نویسندگان پائول فورتیر
هووارد میشل

مترجم طیبه محمدی

پیش بینی و ارزیابی کارایی سیستم های کامپیوتری


بسم الله الرحمن الرحيم

ارزیابی و پیش‌بینی کارایی سیستمهای کامپیوتری

تألیف

Paul J. Fortier

Howard E. Michel

مترجم

طیبه محمدی

سرشناسه

عنوان و نام پدیدآور

مشخصات نشر

مشخصات ظاهری

شابک

وضعیت فهرست‌نویسی

یادداشت

یادداشت

موضوع

موضوع

موضوع

موضوع

شناسه افزوده

رده‌بندی کنگره

رده‌بندی دیویی

شماره کتابشناسی ملی

عنوان کتاب

مؤلف

مترجم

نوبت چاپ

شمارگان ۱۰۰۰ نسخه

ناشر

قیمت

تقدیم به پدر و مادر عزیز و مهربانم

که در سختی‌ها و دشواری‌های زندگی همواره یاوری دلسوز و فداکار و پشتیبانی محکم و مطمئن

برایم بوده‌اند.

فهرست مطالب

پیشگفتار

این کتاب یک رویکرد به‌روز درباره مفاهیم و تکنیک‌های اعمال‌شده برای ارزیابی کارایی سیستمهای کامپیوتری را در اختیار قرار می‌دهد. سیستمهای کامپیوتری در این زمینه دربرگیرنده اجزاء سخت‌افزاری و نرم‌افزاری سیستمهای کامپیوتری، معماری کامپیوتر، شبکه‌های کامپیوتری، سیستمهای عامل، سیستمهای پایگاه داده و میان‌افزارها می‌شوند. انگیزه نگارش این کتاب نشأت‌گرفته از این است که پیدا کردن کتابی که به شکلی مناسب، تکنیک‌های تست تجربی اعمال‌شده برای ارزیابی نرم‌افزار سیستم‌ها و سیستم‌های کامپیوتری که پشتیبان آن‌ها هستند را تحت پوشش قرار داده باشد ناممکن بود. این کتاب می‌تواند به‌عنوان کتابی برای یک‌ترم یا چندترم درباره ارزیابی عملکرد سیستم‌های کامپیوتری یا به‌عنوان یک متن مرجع برای محققان و متصدیان در زمینه‌های ارزیابی عملکرد و مهندسی سیستم‌های کامپیوتری استفاده شود.

در طی ۱۰ تا ۲۵ سال اخیر، مجموعه وسیعی از اطلاعات تجمیع شده که با ارزیابی عملکرد سیستم‌های کامپیوتری سروکار داشته است. ابزارهای سنجشی ویژه، هم سخت‌افزاری و هم نرم‌افزاری، برای کمک در آزمودن و مانیتورینگ عملکرد یک سیستم کامپیوتری در دسترس قرارگرفته‌اند که دارای زبان‌ها و ابزارهای شبیه‌سازی متعددی هستند که هدفشان اجزاء خاص یک سیستم کامپیوتری یا برای مطالعات مدل‌سازی تعمیم داده‌شده است. تکنیک‌ها و ابزارهای تحلیلی را به‌راحتی می‌توان کسب نموده و برای تحلیل سطح بالای سیستم‌های کامپیوتری و کاربردهایشان اعمال نمود. بااین‌حال، بسیاری از این تلاش‌ها منجر به راهکارهای مختلفی شده‌اند که اعمال نتایجشان برای مسائل جدید دشوار و چه‌بسا ناممکن است. علاوه بر این، بیشتر مسائل واقع‌گرایانه نیاز به استفاده از تمام این تکنیک‌ها در سطحی دارند تا عملکرد یک سیستم و تمام عناصر تشکیل‌دهنده آن برای پشتیبانی از طراحی، توسعه و پرداختن سریع به محصول معلوم شود.

برای در نظر گرفتن عملکرد در مراحل طراحی و توسعه شکل‌گیری یک سیستم، مدل‌سازی باید استفاده شود، چراکه سیستم محصول موردنظر هنوز برای ابزار دقیق و تست تجربی در دسترس نیست. مدل‌سازی نسبتاً به‌خوبی توسط متصدیان این رشته که پس‌زمینه مناسبی داشته باشند درک می‌شود، بااین‌حال این تکنیک‌ها به همین سادگی به دیگر اعضای یک گروه طراحی که آن‌ها نیز ممکن است از چنان دانشی منتفع شوند انتقال نمی‌یابند. هدف این کتاب انجام مدل‌سازی مبتنی بر ابزار دقیق، تحلیلی، شبیه‌سازی و

ارزیابی اجزاء کارایی سیستم‌های کامپیوتری امکان‌پذیر و قابل‌درک برای طیف گسترده‌تری از مخاطبان طراحان، توسعه‌دهندگان، مدیران، مجریان و کاربران سیستم‌های کامپیوتری است. فرض این کتاب بر آن است که خواننده با مفاهیم معماری سیستم‌های کامپیوتری، نرم‌افزار سیستم‌های کامپیوتری، شبکه-های کامپیوتری و ریاضیات مقدماتی دربرگیرنده حساب و جبر خطی آشنایی دارد.

محرکه این کتاب بررسی ابزارهایی برای ارزیابی کارایی سیستم‌های کامپیوتری و اجزایشان و فراهم‌سازی مروری اجمالی بر برخی از ابزارهای مورداستفاده در عمل است.

در فصل ۱ درباره ارزیابی و پیش‌بینی عملکرد سیستم‌های کامپیوتری و علت ضروری بودن این تکنیک‌ها در دنیای کنونی که هزینه سیستم‌های کامپیوتری مدام در حال کاهش است بحث شده است.

در فصل ۲، اجزائی که سیستم‌های کامپیوتری را شکل می‌دهند با جزئیات بیشتر در رابطه با معماری‌هایشان، ساختار عناصر سخت‌افزاری پایه، شبکه‌ها و توپولوژی‌ها، پروتکل‌ها و معماری سیستم‌های عامل، اجزاء و تکنولوژی‌های سیستم‌های مدیریت پایگاه داده، سیستم‌های کلاینت/سرور و دیگر پیکربندی‌های سیستم‌های کامپیوتری موردبررسی قرار می‌گیرند.

فصل ۳ به بررسی مجدد موضوع مدل‌سازی از جنبه مدل‌سازی سیستم‌های کامپیوتری، نحوه مفید واقع‌شدن ابزارهای مختلف در سیستم‌های قبلی، نحوه استفاده از آن‌ها در تلاش‌های آتی می‌پردازد. مفاهیم اساسی زمان، رویدادها، سنجش‌ها، بازه‌ها، پاسخ و استقلال به شکلی که به سیستم‌های کامپیوتری ارتباط بیابند موردبحث قرار می‌گیرند.

فصل ۴ تعاریف اساسی ذکرشده در فصل ۳ را بسط می‌دهد.مفاهیم مربوط به فرآیندهای سنجش کلی، توزیع زمان‌های سرویس، زمان‌بندی و زمان پاسخ مرتبط با کاربردهای سیستم‌های کامپیوتری ارائه می-شوند.

در فصل ۵ مفاهیم مربوط به قابلیت سوددهی رویدادها ارائه‌شده‌اند. مفهوم فضای نمونه و کاربرد آن در محاسبه احتمال اساسی وقوع رخداد در یک فضای نمونه موردبررسی قرار می‌گیرند. پس‌ازاین نوبت به بحث درباره تصادفی بودن رویدادها و ارتباط این پدیده با احتمال می‌رسد. سپس مفاهیم احتمال شرطی و مشترک و این مفهوم متغیرهای تصادفی و توزیع‌های احتمال ارائه می‌شوند.

فصل ۶ مبتنی بر مبانی احتمال در فرآیندهای احتمالاتی است. تعریف اساسی یک فرآیند احتمالاتی ارائه می‌شود و سپس رابطه‌اش با فرآیند پواسون موردبررسی قرار می‌گیرد. با این تعاریف، مفهوم یک فرآیند

تولد-مرگ محض، همانند تکنیک‌های تحلیل ارائه می‌شود. این فصل سپس به کنکاش در فرآیند مارکوف و زنجیره‌های مارکوف که به تحلیل سیستم‌های کامپیوتری ارتباط می‌یابند می‌پردازد.

در فصل ۷، ما مفهوم یک صف را معرفی می‌کنیم و تکنیک‌های موردنیاز برای ارزیابی صف‌های واحد و شبکه‌های صف‌ها را مورد ارزیابی قرار می‌دهیم. این تکنیک‌ها سپس به مدل‌سازی تکنیک‌های اعمال‌شده به ارزیابی سیستم‌های کامپیوتری توسعه داده می‌شوند.

در فصل ۸ مفهوم مدل‌سازی شبیه‌سازی معرفی‌شده است. روش‌ها برای مدل‌های شبیه‌سازی ساختاربندی از توصیفی از یک سیستم مدل‌سازی شده موردنظر ارائه می‌شوند. مفاهیم رویدادهای شبیه‌سازی وقت پایی[۱] و سپس اعمال تکنیک‌ها برای تحلیل سیستم‌های کامپیوتری ارائه می‌شوند.

در فصل ۹ یک تکنیک تحلیل دیگر را معرفی می‌کند: شبکه‌های پتری. عناصر اساسی شکل‌دهنده شبکه‌های پتری توسعه داده‌شده و سپس به جنبه‌های مدل‌سازی سیستم‌های کامپیوتری اعمال می‌شوند. شبکه‌های پتری اساسی و این شبکه‌های زمانی و عمومی توضیح داده می‌شوند.

در فصل ۱۰ به طراحان یا معماران آینده نحوه مدل‌سازی پیکربندی سیستم‌های آتی را نشان داده است. این فصل نحوه سنجش دقیق یک سیستم را به‌منظور استخراج و اندازه‌گیری میزان کارایی سیستم‌ها نشان می‌دهد. سپس این سنجش و داده‌ها در توسعه فرآیندهای تحلیل برای تعریف عملکرد فعلی و ترسیم عملکرد آتی سیستم‌های کامپیوتری و اجزائشان استفاده می‌شوند.

فصل ۱۱ به خواننده در تشخیص اینکه کدام ابزار تجزیه‌وتحلیل خاص بهتر است برای ارزیابی سیستم کامپیوتری یا جزء موردنظر استفاده شود، کمک می‌کند. طراح اصولی را ارائه می‌دهد که در تشخیص اینکه چه زمانی تکنیک‌های تحلیلی را استفاده کنیم، کدام تکنیک را استفاده کنیم و چه زمانی آن را استفاده کنیم، کمک می‌کند. اگر تکنیک‌های تحلیلی بهترین تکنیک‌های کاربردی نباشند، توصیه‌هایی به مخاطب در مورد نحوه انتخاب یک ابزار مدل‌سازی و زمان اعمال آن در تحلیل یک سیستم کامپیوتری ارائه می‌شود. درنهایت، اطلاعاتی در رابطه بازمان و نحوه انتخاب ابزار تحلیل عملیاتی مناسب برای سنجش و مدل‌سازی اجزاء و سیستم‌های کامپیوتری موجود به مخاطب داده می‌شود.

---

[۱] timekeeping

در فصل‌های ۱۲ تا ۱۵ نمونه‌های تحلیلی برای اجزاء سیستم‌های کامپیوتری معین ارائه می‌شوند. ارزیابی اجزاء و معماری کامپیوتر همانند تحلیل و مدل‌سازی سیستم‌های عامل، سیستم‌های پایگاه داده و سیستم‌های شبکه توضیح داده می‌شوند.

فصل اول

مقدمه

مقدمه

ارزیابی و پیش‌بینی عملکرد سیستم‌های کامپیوتری چیست و چرا این تکنیک‌ها در دنیای امروز که هزینه سیستم‌های کامپیوتری به‌طور مداوم در حال کاهش است ضروری هستند؟ پاسخ دادن به این پرسش‌ها نیازمند آن است که مهندس کامپیوتر درک کند که چگونه تمام عناصر یک سیستم کامپیوتری در تحقق استفاده‌های یک کاربر و پیاده‌سازی، اداره و نگهداری آن نقش‌آفرین می‌شوند. تمام جنبه‌های طول عمر یک سیستم کامپیوتری در زمان تلاش برای درک مسائل کارایی مهم هستند. صرف خرید "بهترین" ماشین همه‌منظوره محاسباتی که امروز بتوان یافت و سپس پیاده‌سازی برنامه‌های کاربردی موردنظر بر روی آن کفایت نمی‌کند. لازم است که نحوه جای گرفتن سیستم در یک ساختار محاسباتی موجود و این‌گونه امروز چه الزاماتی برای سیستم کامپیوتری وجود دارد و این الزامات در طی دوران عمر سیستم کامپیوتری چه خواهند بود ضروری است.

مهم‌ترین فاکتورهای محرکه در زمان طراحی، ساخت و مدیریت یک سیستم کامپیوتری عبارت‌اند از اجرای درست عملکرد موردنظر، انجام مؤثر کارکرد موردنظر، و انجام این کار به شکلی مقرون‌به‌صرفه. ازاین‌رو، درست بودن طراحی داخلی اغلب ممکن است مهم‌تر از محرکه‌ای قوی‌تر از عملکرد و هزینه باشد. با این تفاصیل، اغلب مهم است که طراحان سیستم‌های کامپیوتری عملکرد، هزینه و صحت کارایی را هم‌سطح باهم در نظر بگیرند. بااین‌حال، آن‌ها متفاوت هستند. یک طراحی درست ممکن است متضمن طراحی که به‌شدت سریع عمل کند یا خیلی مقرون‌به‌صرفه باشد نباشد. این می‌تواند به علت ملاحظات دیگر باشد، مثلاً ما می‌توانیم برای صرفه‌جویی در هزینه‌ها توازنی بین عملکرد یا صحت کامل ایجاد نماییم. این در طراحی‌های مهندسی رواج بیشتری دارد. ما هیچ زمان و بودجه بینهایت نخواهیم داشت، که امکان طراحی، ساخت و مدیریت بهینه‌ترین عملکرد سیستم کامپیوتری را فراهم بیاورد. ازاین‌رو، ما نیاز به روش‌هایی داریم تا به ما در توسعه سیستم‌هایی کمک کنند که در آن‌ها بتوانیم به شکلی مصالحه-ای بین این آیتم‌های متنازع بیابیم. ارزیابی عملکرد سیستم‌های کامپیوتری به این صورت است و این کتاب کلاً به آن می‌پردازد.

هدف این کتاب توصیف روش‌های مختلف تحلیلی است که برای مراحل مختلف طراحی، ساخت، مدیریت و نگهداشت چرخه حیات قابل‌اعمال باشند. هدف ما فراهم‌سازی درکی از این است که چه ابزارها یا تکنیک‌هایی هستند که به بهترین شکل در چرخه عمر یک سیستم کامپیوتری اعمال می‌شوند

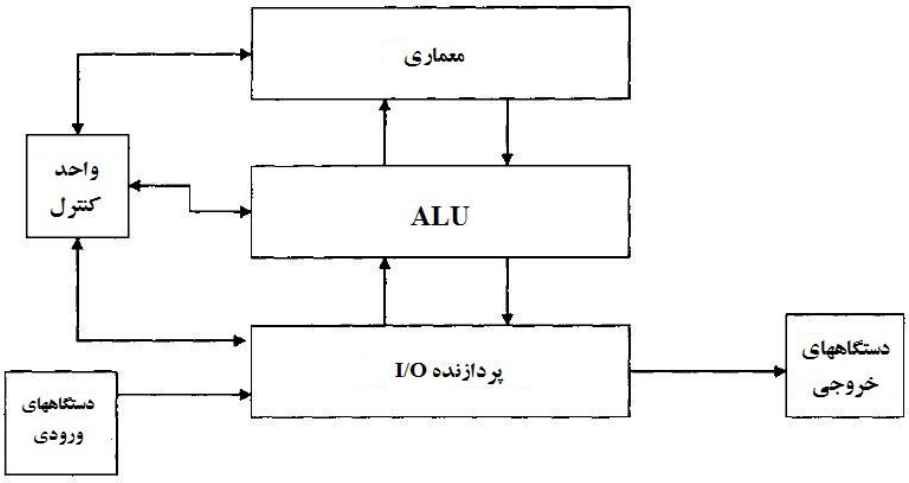
تا طراح بتواند گزینه‌های جایگزین را مورد تجزیه‌وتحلیل قرار داده و راهکارهای تقریباً بهینه را برای هر مرحله از این فرایند انتخاب نماید. ما نمی‌توانیم امیدوار باشیم که بتوانیم به‌طور کامل تمام جنبه‌های یک طراحی سیستم کامپیوتری را پوشش دهیم، و این نمی‌توانیم چنین کاری را برای هر تکنیک تحلیل در دسترس انجام دهیم. هدف ما ارائه جزئیات، نمونه‌ها و ارجاعات کافی است تا یک مخاطب علاقه‌مند بتواند بداند که بهترین تکنیک ارزیابی کارایی چیست، چگونه این تکنیک در سطحی از پیچیدگی اعمال شود، و در صورت نیاز در کجا به دنبال اطلاعات مشروح بیشتر بگردد. هدف ما فراهم‌سازی جزئیات، نمونه‌ها، و ارجاعات کافی است تا مخاطب بتواند بداند که بهترین تکنیک ارزیابی کارایی که باید اعمال شود چیست، چگونه این تکنیک به سطحی از پیچیدگی اعمال می‌شود، و کجا باید به دنبال اطلاعات بیشتر، در صورت نیاز، درباره یک مبحث گشت. هدف ما فراهم‌سازی یک تحقیق عمیق‌تر است تا مخاطب بتواند درک نماید که چگونه تمام جنبه‌ها و تکنیک‌های مختلف برای تحلیل ایجاد توازن بین سیستم‌های کامپیوتری اعمال می‌شوند.

۱-۱ تکامل معماری‌های سیستم‌های کامپیوتری

کامپیوترها با توسعه سیستم کامپیوتری ENIAC در اواخر دهه ۱۹۴۰ پا به عرصه وجود گذاشتند. ENIAC و کامپیوترهای متعاقب آن از لوله‌های خلأ ساخته‌شده بودند و فضای زیادی را اشغال نموده بودند. این سیستم‌های کامپیوتری اولیه مختص یک کار واحد بودند و سیستم‌عامل نداشتند. قدرت این کامپیوترهای اولیه کمتر از قدرت ماشین‌حساب‌های دستی امروز بود. این کامپیوترها اصولاً برای طرح‌های مسیر موشک بالستیک و تحقیقات نظامی استفاده می‌شدند. معماری این کامپیوترهای اولیه مبتنی بر برنامه ذخیره‌شده فون نویمان، معماری جریان آموزشی تک جریانی بود (شکل ۱-۱). این معماری و فلسفه اساسی همچنان امروز در بیشتر سیستم‌های کامپیوتری مورداستفاده قرار می‌گیرد.

این سیستم‌های اولیه کامپیوتری سیستم‌های عامل، پایگاه داده، شبکه، و یا زبان برنامه‌نویسی سطح بالایی برای ساده‌سازی عملیاتشان نداشتند. دستورالعمل‌ها و داده‌های ذخیره‌شده برنامه برای محاسبه در یک مکان موردنیاز بودند. دستورالعمل‌ها در هر بار از حافظه خوانده می‌شدند و عمدتاً مرتبط با بارگیری و ذخیره‌سازی داده‌های برنامه از حافظه در ثبات‌هایی بودند که در آن‌ها داده‌ها باید بر رویشان آماده می‌شدند. داده‌ها در این سیستم‌های اولیه توسط برنامه‌ها به اشتراک گذاشته نمی‌شدند. اگر یک برنامه

به داده‌های تولیدشده توسط یک برنامه دیگر نیاز داشت، این آیتم‌های داده‌ها نوعاً در یک بخش نزدیک به انتهای فضای برنامه کپی می‌شدند، و نشانی‌های نهایی برای استفاده توسط برنامه کاربردی که در آن گنجانده‌شده بودند هاردکد[۲] شدند.



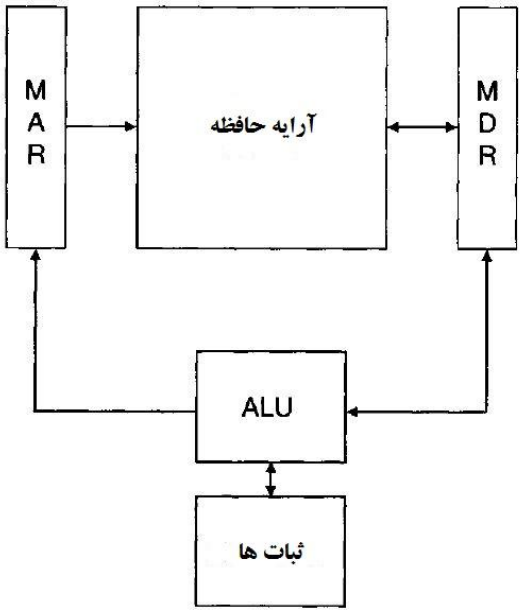
شکل (۱-۱): سیستم کامپیوتری پایه

یک برنامه کاربردی کاربری بر روی یک سیستم کامپیوتری قرار می‌گیرد. سیستم کامپیوتری واسطه فیزیکی را فراهم می‌آورد که داده‌ها و برنامه‌ها و ظرفیت پردازشی برای دست‌کاری داده‌های ذخیره‌شده بر روی آن ذخیره می‌شود. یک واحد پردازشی یک سیستم کامپیوتری از پنج عنصر اصلی تشکیل یافته است: حافظه، یک واحد حساب و منطق، یک واحد ورودی، یک واحد خروجی، و یک عنصر کنترل. واحد حافظه هم داده‌ها برای برنامه‌ها و هم دستورالعمل‌های برنامه‌ای را که داده‌های ذخیره‌شده را دست‌کاری می‌کند ذخیره می‌نماید.

[۲] hard- code

عناصر یا دستورالعمل‌های فردی برنامه هر یک‌بار از حافظه گرفته می‌شوند و توسط واحد کنترل تفسیر می‌شوند. واحد کنترل، بسته به تفسیر دستور، تعیین می‌کند که چه عملیات کامپیوتری بعداً باید انجام شود. اگر این دستور نیازی به داده‌های اضافی نداشته باشد، این کنترل به واحد منطق حسابی نشان می‌دهد که چه عملیاتی و با چه ثبات‌هایی باید انجام شود. (شکل (۱-۱) را ببینید)

اگر این دستورالعمل نیاز به داده‌های اضافی داشته باشد، واحد کنترل فرمان مناسب را به حافظه انتقال می‌دهد (MAR، ثبات نشانی حافظه[۳]) تا یک آیتم داده‌ها را از حافظه بگیرد (MDR، ثبات داده‌های حافظه[۴]) و آن را در یک ثبات مناسب در ALU (بانک ثبات داده‌ها) قرار می‌دهد (شکل ۱-۲).



شکل (۱-۲): دسترسی حافظه سطح پایین

[۳] memory address register
[۴] memory data register

این تا زمانی ادامه می‌یابد که تمام عملوندهای موردنیاز در ثبات‌های مناسب ALU قرار بگیرند. وقتی تمام عملوندها درجایشان قرار گرفتند واحد کنترل به ALU فرمان می‌دهد که دستورالعمل مناسب را انجام دهد – به‌عنوان‌مثال، ضرب، جمع یا تفریق. اگر این دستورالعمل نشانگر آن بود که نیاز به یک ورودی یا خروجی است، عنصر کنترلی می‌توانست کلمه‌ای را، بسته به دستورالعمل، از واحد ورودی به حافظه یا ALU انتقال دهد. اگر یک دستورالعمل خروجی رمزگشایی[۵] می‌شد، واحد کنترل می‌توانست فرمان انتقال واژه حافظه مناسب یا ثبات را به کانال خروجی ذکرشده بدهد. این پنج عنصر متشکل از بلوک‌های اساسی در سیستم کامپیوتر فون نویمان اصلی بنیادین هستند و به شکلی در بیشتر سیستم‌های موقتی یافت می‌شوند.

یک سیستم کامپیوتری از پنج بلوک ساختاری که قبلاً توضیح داده شدند و این دستگاه‌های پشتیبان پیرامونی اضافی تشکیل یافته است، که به انتقال داده‌ها و پردازش کمک می‌کنند. این بلوک‌های ساختاری پایه برای شکل‌دهی واحدهای پردازش کلی، کنترل، ذخیره‌سازی، ورودی و خروجی استفاده می‌شوند که سیستم‌های کامپیوتری مدرن را شکل می‌دهند. دستگاه‌ها نوعاً به شکلی سازمان‌دهی می‌شوند که از پردازش برنامه‌های کاربردی که سیستم کامپیوتری برای آن مدنظر است پشتیبانی می‌کند – به‌عنوان‌مثال، اگر مقادیر عظیمی از داده‌ها برای ذخیره‌سازی لازم باشند، دستگاه‌های پیرامون اضافی همچون دیسک‌ها یا واحدهای نوار، در کنار کنترلرهای موردنیاز یا کانال‌های داده‌های موردنیازشان لازم خواهند بود.

برای توصیف بهتر تغییرات درون معماری، ما درباره برخی از جزئیات بحث خواهیم نمود – به‌عنوان‌مثال واحد منطقی حسابی (ALU) واحد کنترل باهم، در یک واحد پردازش مرکزی، یا CPU، ادغام می‌شوند. CPU جریان دستورات و داده‌ها در سیستم کامپیوتری را کنترل می‌کند. حافظه‌ها قابل‌تقسیم بندی به سلسله‌مراتب بر اساس نزدیک بودن به CPU و سرعت هستند – به‌عنوان‌مثال، حافظه کش حافظه کوچک و فوق‌العاده سریع مورداستفاده برای دستورالعمل‌ها و داده‌هایی است که به شکلی فعالانه اجرا می‌شوند و توسط CPU مورداستفاده قرار می‌گیرند. حافظه اولیه کندتر است، ولی ارزان‌تر هم هست و حاوی مکان‌های حافظه بیشتری است. از آن برای ذخیره‌سازی داده‌ها و دستورالعمل‌هایی استفاده می‌شود که در جریان اجرای برنامه‌های کاربردی استفاده خواهند شد که در حال حاضر بر روی CPU

---

[۵] decoded

اجرا می‌شوند – به‌عنوان‌مثال، اگر شما برنامه واژه‌پردازتان را بر روی کامپیوتر شخصیتان اجرا نمایید، سیستم‌عامل تلاش خواهد نمود تا برنامه را به چند بخش تقسیم نموده و آن‌ها را در صورت نیاز برداشت نماید.

آن بخشی از برنامه که در حافظه قابل ذخیره‌سازی نیست بر روی یک دستگاه ذخیره‌سازی ثانویه، که نوعاً یک دیسک درایو است، نگهداری می‌شود. این دستگاه یک ظرفیت ذخیره‌سازی به‌مراتب بزرگ‌تر از حافظه اولیه دارد، که نوعاً هزینه به ازای واحد به‌مراتب کمتری برای ذخیره‌سازی دارد، و زمان‌های دسترسی داده‌هایی دارد که به‌مراتب کندتر از حافظه اولیه هستند. یک دستگاه ذخیره‌سازی ثانویه اضافی واحد درایو نواری است. یک درایو نواری یک دستگاه ذخیره‌سازی ساده است که می‌تواند حجم عظیمی از داده‌ها را ذخیره نماید، که در اینجا هم با هزینه‌ای کمتر از واحدهای دیسک ولی با سرعت دسترسی کمتر است. دیگر اجزاء یک سیستم کامپیوتری واحدهای ورودی و خروجی هستند. این‌ها برای استخراج داده‌ها از کامپیوتر استفاده می‌شوند و این داده‌ها را در اختیار دستگاه‌های خارجی یا داده‌های ورودی از دستگاه خارجی قرار می‌دهند. دستگاه‌های خارجی می‌توانند پایانه‌های کاربر نهایی، سنسورها، پورت‌های شبکه اطلاعاتی، ویدئو، صوت یا کامپیوترهای دیگر باشند.

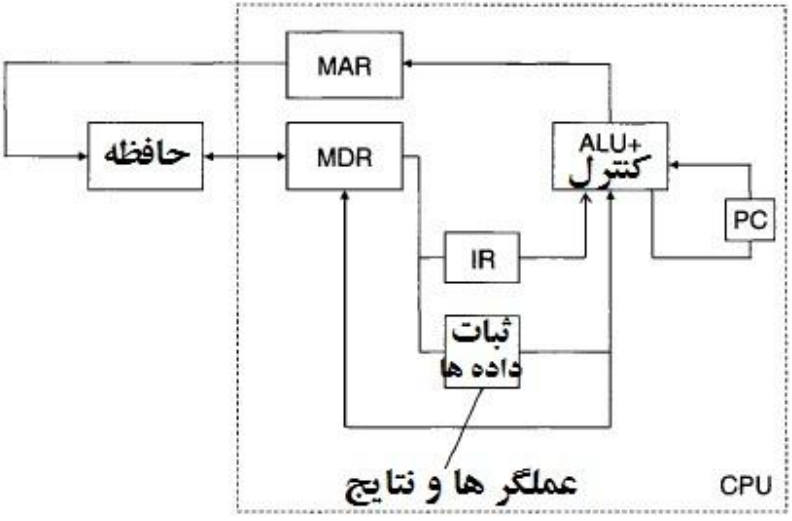
یک معماری سیستم کامپیوتر با استفاده از بلوک‌های ساختاری پایه، همچون CPUها، دیسک‌ها، I/O، و دیگر دستگاه‌های موردنیاز ساخته می‌شود.

در بخش‌های بعدی ما هر یک از اجزاء یک سیستم کامپیوتری را با جزئیات بیشتر موردبررسی قرار خواهیم داد، و بررسی می‌کنیم که چگونه این دستگاه‌ها می‌توانند برای پشتیبانی از برنامه‌های کاربردی پردازش داده‌ها به هم متصل شوند.

۱-۱-۱ معماری‌های CPU

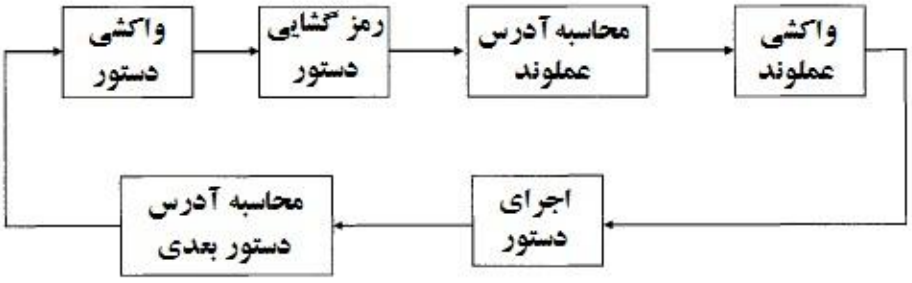
واحد پردازش مرکزی (CPU) هسته یک سیستم کامپیوتری است و از واحد منطق حسابی (ALU) واحد کنترل تشکیل یافته است. ALU می‌توانند با پیکربندی‌های مختلفی ارائه شود – از یک واحد ساده تکی، تا واحدهای فوق‌العاده پیچیده‌ای که عملیات پیچیده را انجام می‌دهند. عملیات اولیه ALU صفر دادن به عملوندهای بیشتر و انجام کارکرد خواسته‌شده در دستورالعمل است. علاوه بر ALU، CPU از مجموعه‌ای از ثبات‌ها برای ذخیره‌سازی عملوندها و نتایج میانی عملیات و

نگهداری اطلاعات مورداستفاده CPU برای تعیین وضعیت محاسباتش تشکیل یافته است. به‌عنوان‌مثال، ثبات‌هایی برای وضعیت عملیات ALU وجود دارند، برای زیر نظر داشتن دستوری که باید بعداً انجام شود، برای حفظ جریان ورودی و خروجی به حافظه، برای دستوری که در حال اجرا است، و برای مکان عملوندهایی که با CPU بر رویشان عملیات انجام می‌شود. هر یک از این ثبات‌ها دارای یک کارکرد منحصربه‌فرد درون CPU هستند، و هر یک برای کلاس‌های مختلف معماری‌های کامپیوتری ضروری است. یک معماری حداقلی نوعی برای یک CPU و ثبات‌های آن در شکل (۱-۳) نشان داده‌شده است و از یک حافظه اولیه متصل به CPU از طریق باس ها تشکیل یافته است. ثبات‌هایی در CPU برای نگهداری ثبات‌ها و نتایج عملیات، عملوندهای دستور، یک شمارنده مکان برنامه، حاوی مکان در حافظه برای دستورالعمل‌ها یا عملوندها، بسته به رمزگشایی دستورالعمل‌ها، و یک شمارنده برنامه حاوی مکان دستورالعمل بعدی که باید انجام شود وجود دارند.



شکل (۱-۳): معماری ریزپردازنده معمولی

CPU حاوی واحد کنترل نیز هست. واحد کنترل از ثبات‌ها و دستورالعمل‌های وضعیت در ثبات دستورالعمل برای تعیین اینکه CPU باید چه کارکردهایی برای ثبات‌ها، ALU، و مسیرهای داده‌هایی که CPU را شکل می‌دهند انجام دهد استفاده نماید. عملیات اساسی CPU از یک حلقه مشابه تبعیت می‌کند، که چرخه اجرا نامیده می‌شود (شکل ۱-۴). شش تابع اساسی وجود دارد که در حلقه دستورالعمل به اجرا درمی‌آیند: کسب دستور، رمزگشایی دستور، محاسبه نشانی مؤثر عملوند، بررسی عملوند، اجرای عملیات، و محاسبه نشانی بعدی. این توالی اجرا کارکردهای اساسی یافته شده در تمام سیستم‌های کامپیوتری را نشان می‌دهد. تغییرات در تعداد گام‌ها بر مبنای نوع و طول این دستورالعمل یافته می‌شوند.



شکل (۱-۴): اجرای چرخه دستور

۱-۱-۲ معماری‌های دستورالعمل

ایده‌های متعددی درباره نحوه سازمان‌دهی سیستم‌ها حول مجموعه دستورالعمل وجود دارند. یک‌شکل، که با این کامپیوترهای قدرتمند[۶] در طی دوره‌ها تحول‌یافته است، کامپیوتری با مجموعه دستورالعمل کاهش‌یافته (RISC)، که در آن ساده، ولی بسیار بهینه‌سازی شده است. در یک سمت این طیف کلمه بسیار طولانی معماری دستورالعمل قرار دارد، که در آن هر دستورالعمل می‌تواند یک تابع پردازشی عظیم را نمایندگی کند. یک زمینه میانی کامپیوتر با مجموعه دستورالعمل ترکیبی و پیچیده (CISC) است.

[۶] powerfill workstations

طرح‌های آدرس‌دهی حافظه

این راه‌های متعددی برای تعیین آدرس یک عملوند[۷] از یک دستورالعمل وجود دارند. هر روش محاسبه آدرس از جنبه انعطاف‌پذیری طراحی دستورالعمل منافع خودش را دارد. شش نوع عمده از طرح‌های محاسبه آدرس‌دهی در کامپیوترها یافت می‌شوند: میانی، مستقیم، شاخص، مبنا، غیرمستقیم، و دو عملوندی. ما این را در فصل ۲ بیشتر بررسی خواهیم نمود.

۱-۱-۳ معماری‌های حافظه

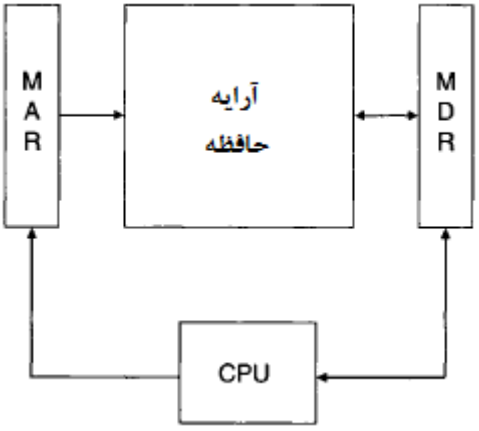
به‌طورکلی، حافظه یک سیستم کامپیوتری یک ساختار معمول است، که با استفاده از محتوای یک ثبات آدرس حافظه و با داده‌های انتقال داده‌شده از طریق یک ثبات داده‌های حافظه مورد چاره-جویی قرار می‌گیرد (شکل ۱،۵). بسیاری از معماری‌ها مبتنی بر سازمان کلمات حافظه هستند. ساده-ترین شکل یک ساختار دوبعدی خطی است. یک سازمان ثانویه یک معماری دو نیم بعدی است.

۱-۱-٤ معماری‌های I/O

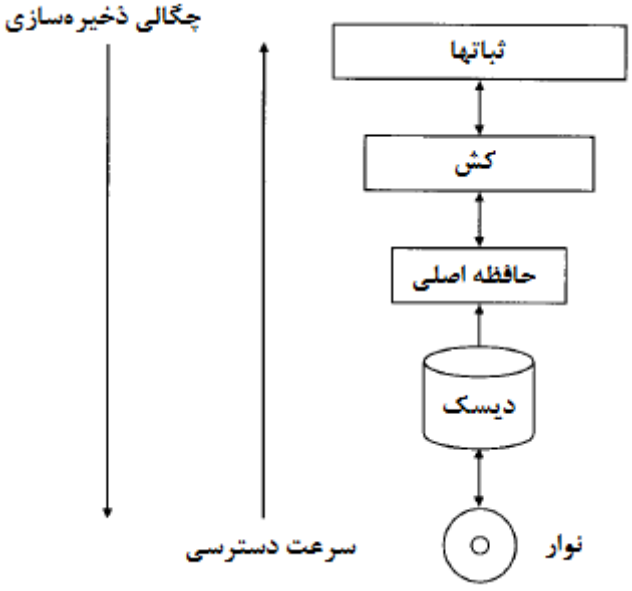
معماری‌های ورودی و خروجی توسط سیستم‌های کامپیوتری برای انتقال اطلاعات به داخل و خارج از حافظه اصلی کامپیوتر استفاده می‌شوند و به شکل‌های بسیاری تکامل‌یافته‌اند. معماری‌های I/O معمولاً متکی بر استفاده از یک عنصر کامپیوتر به‌عنوان روتر انتقال‌های I/O هستند. این روتر می‌تواند CPU، حافظه، یا یک کنترلر ویژه باشد. فصل ۲ این معماری‌ها را با جزئیات بیشتر موردبحث قرار می‌دهد.

---

[۷] operand



شکل (۱-۵): دسترسی حافظه CPU



شکل (۱-۶): سلسله‌مراتب حافظه

۱-۱-۵ معماری‌های دستگاه پیرامونی و ذخیره‌سازی ثانویه

دستگاه‌های I/O به دستگاه‌های ذخیره‌سازی ثانویه متصل شده و آن‌ها را کنترل می‌کند. حافظه اولیه تا یک حجم نسبتاً بالا رشد یافته است، ولی نه هنوز تا نقطه‌ای که در آن داده‌های اضافی و ذخیره برنامه لازم است. سلسله‌مراتب ذخیره‌سازی (شکل ۱-۶) از چند نوع ذخیره داده‌ها استفاده می‌کند. از عنصری از حافظه که دارای بالاترین سرعت است، یعنی کش، تا عناصری که دارای کمترین سرعت هستند، همچون درایوهای نواری، مصالحه‌ای که معمار سیستم‌ها باید انجام دهد هزینه و سرعت واسطه ذخیره‌سازی به ازای واحد حافظه است. دستگاه‌های ذخیره‌سازی ثانویه نوعی شامل درایوهای نوار مغناطیسی، درایوهای دیسک مغناطیسی، درایوهای دیسک نوری فشرده، و دستگاه‌های ذخیره‌سازی آرشیوی همچون جعبه‌های دیسک هستند.

ذخیره‌سازی (مخزن) اطلاعات نوار مغناطیسی یک واسطه ذخیره‌سازی کم‌هزینه با چگالی بالا را برای داده‌های دسترسی پایین یا دسترسی کند فراهم می‌آورد. یکی از پیشرفت‌های رخ‌داده در حوزه ذخیره‌سازی نوار واحدهای دیسک دسترسی تصادفی است، که می‌توانند واسطه ذخیره-سازی قابل‌خروج یا ذخیره‌سازی ثابت داخلی را داشته باشد. دستگاه‌های ذخیره‌سازی آرشیوی نوعاً متشکل از واسطه قابل‌خروج پیکربندی‌شده در چند آرایه دستگاه‌ها هستند.

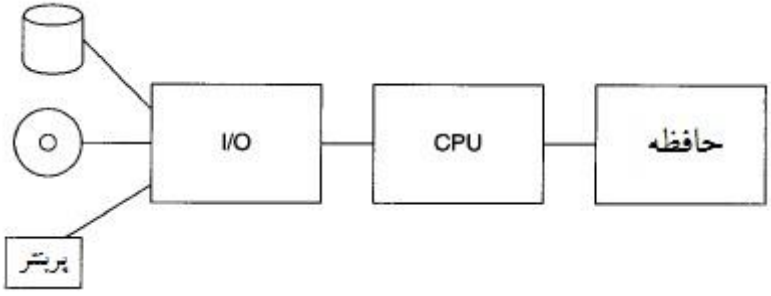
۱-۱-٦ معماری‌های شبکه

شبکه‌های از نیازهای برنامه‌های کاربردی و سازمان‌ها برای اشتراک‌گذاری اطلاعات و ظرفیت پردازشی به‌طور بی‌درنگ تکامل یافتند. شبکه‌های کامپیوتری یک مسیر ورودی و خروجی دیگر را برای کامپیوتر برای دریافت یا ارسال اطلاعات فراهم می‌آورند. شبکه‌ها به طرق متعددی معماری می‌شوند: آن‌ها می‌توانند یک عنصر سوئیچینگ مرکزی داشته باشند، یک مخزن ذخیره‌سازی مرکزی اشتراکی داشته باشند، یا می‌توانند با استفاده از واحدهای سطح هوشمند بر روی یک واسطه ارتباطی به‌صورت سیم‌های تلفن یا کابل‌های دیجیتالی متصل شوند. پیکربندی استفاده‌شده به درجه همگام‌سازی و کنترل موردنیاز و این توزیع فیزیکی بین کامپیوترها بستگی دارد. در فصل ۲ برخی از معماری‌ها و پیکربندی‌های توپولوژی برای سیستم‌های کامپیوتری شبکه‌بندی شده موردبررسی قرار خواهند گرفت.

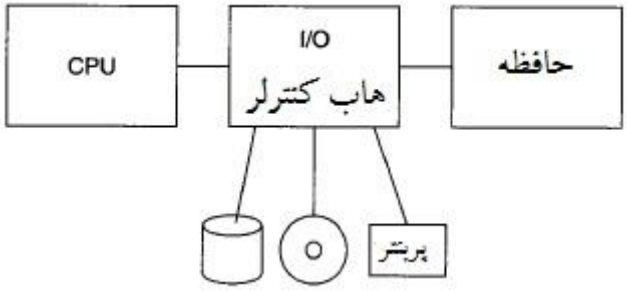
۱-۱-۷ معماری‌های کامپیوتری

معماری‌های کامپیوتر ابزارهای اتصال را برای اجزاء سخت‌افزار کامپیوتری و این حالت انتقال داده‌ها و پردازش نشان داده‌شده نمایندگی می‌کنند. پیکربندی‌های معماری‌های مختلف کامپیوتر برای سرعت بخشیدن به انتقال داده‌ها توسعه داده‌شده‌اند، که امکان افزایش پردازش داده‌ها را فراهم می‌آورند. محور معماری پایه CPU است که یک حافظه و سیستم ورودی/خروجی اصلی در هر سمت CPU قرار دارند (شکل (۱-۷) را ببینید). یک پیکربندی کامپیوتری ثانویه کنترلر ورودی/خروجی مرکزی است (شکل (۱-۸) را ببینید). یک معماری کامپیوتر ثالث از حافظه اصلی به‌عنوان مکان در سیستم کامپیوتری استفاده می‌کند که تمام داده‌ها و دستورالعمل‌ها به آن ورود و خروج می‌کنند. یک معماری کامپیوتر چهارم از یک باس مشترک داده‌ها و کنترل برای ارتباط دادن تمام دستگاه‌هایی که یک سیستم کامپیوتری را شکل می‌دهند استفاده می‌کند (شکل۱-۹ را ببینید). یک پیشرفت در معماری باس مرکزی اشتراکی واحد معماری باس دوگانه است. این معماری یا داده‌ها را جداسازی نموده و یا باس ها را کنترل نموده یا آن‌ها را به اشتراک می‌گذارد تا عملکرد کلی افزایش یابد (شکل (۱-۱۰) را ببینید).

ما در ادامه بحثمان درباره معماری‌ها و عملیات سیستم، نحوه استفاده از این معماری‌ها و عناصر را خواهیم دید.

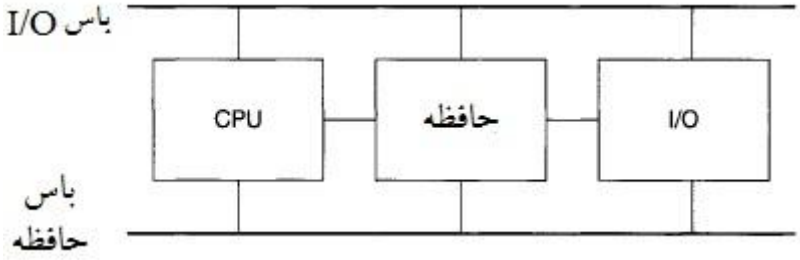


شکل (۱-۷): معماری کامپیوتر پایه



شکل (۱-۸): معماری کامپیوتر ثانویه

شکل (۱-۹): معماری باس مشترک



شکل (۱-۱۰): معماری باس دوگانه

۱-۲ تکامل سیستم‌های پایگاه داده

سیستم‌های پایگاه داده از دهه ۱۹۶۰ به‌عنوان ابزارهای تحقیقاتی (محصولات نسل اول حول مدل‌های داده‌های شبکه و سلسله‌مراتبی شکل‌گرفته بودند) با ما بوده‌اند و از اواسط دهه ۱۹۸۰ به‌صورت محصولات کاملاً کاربردی با استفاده از مدل داده‌های رابط‌های بوده‌اند. از زمان این شروع‌های اولیه، سیستم‌های پایگاه داده از مخزن‌های ساده برای اطلاعات ماندگار به ابزارهای بسیار قوی برای مدیریت و استفاده از اطلاعات تکامل‌یافته‌اند.

سیستم‌های پایگاه داده از نخستین روزهای عرضه کامپیوترهای تجاری موردتوجه تحلیلگران عملکرد سیستم‌های کامپیوتری و توسعه‌دهندگان برنامه‌های کاربردی سیستم‌های کامپیوتری بوده‌اند. سیستم‌های کامپیوتری اولیه فاقد مخزن گسترده آنلاین داده‌ها (حافظه اولیه و این مخزن دیسک ثانویه) بودند، که باعث ناچار شدن معماران و توسعه‌دهندگان سیستم‌ها به اتکای شدید بر اطلاعاتی شده بود که از خارج کسب‌شده بودند (و نوعاً در درایوهای نوار ذخیره شدند). مخزن‌های ذخیره‌سازی اولیه داده‌ها با استفاده از طرح‌های آدرس‌دهی مستقیم ساده ساخته‌شده بودند که مخزن خاص را به یک دستگاه خاص و مکان خاص بر روی آن دستگاه ارتباط می‌دادند. به‌عنوان‌مثال، به‌منظور استخراج اطلاعات یک برنامه کاربردی نیاز به این داشت که بداند یک اطلاعات مشخص بر روی چه دستگاهی ذخیره‌سازی شده است (مثلاً دیسک ۱) و راهکار دقیق را برای آن دستگاه بیابد (مثلاً بخش ۰۵، ترک ۲۲، آفست ۲۵۵، طول ۱۰۲۴). هر دستگاه شیوه منحصربه‌فرد خودش را برای ذخیره‌سازی، دسترسی، و بازیابی اطلاعات داشت، که باعث شد تا انتقال برنامه‌های کاربردی از یک مکان به مکان دیگر بسیار دشوار شود.

این مخزن‌های اولیه به سیستم‌های مدیریت فایل قوی‌تر تکامل یافتند، که نشأت‌گرفته از حرکتی به سمت ساده‌سازی رابط برنامه‌های کاربردی سیستم بود. حرکت به سمت ساده‌سازی، نشأت‌گرفته از توسعه-دهندگان برنامه‌های کاربردی و تکامل‌های سیستم‌های عامل برای حذف پیچیدگی سلسله‌مراتب ذخیره-سازی نوعی از سمت کاربر/توسعه‌دهنده و قرار دادن آن در سمت سیستم‌های عامل بود. انگیزه این کار اجرای رابط کاربری در سطح سیستم‌عامل به‌منظور ساده‌سازی رابط بود، که در آن برنامه‌های کاربردی امکان دسترسی مداوم منطقی به اطلاعات ذخیره‌شده را، به‌جای حالت فیزیکی توسط مسیرهای آدرس

خاص داشتند. این سیستم‌های مدیریت فایل اولیه ابزارهایی برای یک برنامه کاربردی ارائه نمودند تا به‌صورت منطقی به‌طور مداوم اطلاعات را برای بازیابی و استفاده آتی ذخیره نماید. سیستم‌های فایل اولیه یک رابط و پیاده‌سازی ساده را برای ذخیره‌سازی و بازیابی اطلاعات با استفاده از ابزارهای معنایی درشت پیشنهاد دادند. فرد می‌تواند یک فایل را باز کند، محتوای جهت‌دهی شده بر اساس سوابق یک فایل را بخواند، یک سابقه یا کل فایل را بنویسد، و فایل را ببندد. اطلاعات درون فایل هیچ معنایی برای نرم‌افزار کنترلی سیستم‌عامل یا سیستم پایگاه داده نداشت. نرم‌افزار مدیریت فایل درباره همه نقاط تا یک فایل، یا زیرمجموعه‌ای یک فایل، اطلاع داشت، ولی هیچ‌چیز در رابطه با جزئیات محتوای اطلاعات درون فایل نمی‌دانست. این سیستم‌های فایل اولیه و طرح‌های دسترسی خام آن‌ها در خدمت نیازهای ماشین‌های مین فریم[۸] اولیه بودند، که در آن‌ها کارها به‌طور متوالی اجرا می‌شدند و هیچ اشتراک‌گذاری بین کارها صراحتاً در زمان اجرا لازم نبود.

ابداع سیستم‌های عامل چندکاربره، و نیازهای رو به تکامل برنامه‌های کاربردی چندکاربره برای دسترسی همزمان به اطلاعات ذخیره‌شده در فایل سیستم‌ها، باعث بروز نیاز به تکامل سیستم‌های پایگاه داده از مخزن‌های مداوم تک کاربره به سیستم‌های پایگاه داده همزمان چندکاربره شد. سیستم‌های کامپیوتری چند پردازشی و چندکاربره ایجاب می‌کردند که اطلاعات ذخیره‌شده درون فایل سیستم کامپیوتری برنامه‌های کاربردی باید برای اشتراک‌گذاری در دسترس باشد. این این اطلاعات علاوه بر اینکه باید به اشتراک گذاشته می‌شد، بلکه باید چنین کاری به شیوه‌ای دینامیک انجام می‌شد. ذخیره‌سازی، دسترسی و بازیابی – درون چنان سیستم‌های رو به تکاملی – نیاز به کنترل‌های بیشتری داشت تا اطلاعات را بتوان به اشتراک گذاشت، بااین‌حال برای چشم‌انداز تمام برنامه‌های کاربردی که از آن استفاده می‌کنند درست و همخوان بود.

یک مشکل اشتراک‌گذاری اطلاعات در چارچوب زمینه این سیستم‌های جدید امنیت بود – چگونه اجازه می‌دهید که فقط مالک، یا گروه کاربران، به یک فایل دسترسی یافته یا آن را تغییر دهند درحالی‌که همچنان امکان دسترسی توسط دیگران را فراهم می‌آورند؟ دستیابی به یکپارچگی در همخوانی با این مسئله بود – چگونه داده‌ها را دست‌نخورده و درست نگه‌داریم، درحالی‌که کاربران متعدد به اطلاعات دسترسی می‌یابند، آن‌ها را دست‌کاری نموده، می‌افزایند یا حذف می‌کنند. در ابتدا،

[۸] mainframe

فایل سیستم‌ها با افزودن کنترل‌های دسترسی برای بیشتر این دغدغه‌ها، همچون قفل‌ها و لیست‌های دسترسی به مدیران فایل به‌منظور کنترل چنان دسترسی چاره‌جویی نمودند، ولی این‌ها اهداف موردنظر را محقق ننمودند. اگرچه این‌ها ارتقاهای قابل‌ستایشی بودند، ولی به‌مراتب خام‌تر از آن بودند که اجازه اشتراک‌گذاری درست داده‌های آنلاین را بدهند. فایل‌ها لازم است که به عناصر ریزساختار تجزیه شوند اگر همزمانی ظریف‌تری از دسترسی حاصل شود. قفل‌سازی سطح فایل ساده منجر به انتظارهای طولانی‌تر شده و دسترسی‌پذیری داده‌ها را برای استفاده توسط کاربردهای دیگر کاهش داد.

به‌منظور کاستن از این مشکلات، فایل سیستم‌ها تعاریف جزئی‌تری را از اطلاعات ذخیره‌شده را افزودند. به‌عنوان‌مثال، فایل‌ها از داده‌های ساختار نیافته به ساختاریافته، مجموعه‌های رکورد محور اطلاعات، که در آن‌ها هر رکورد یک سروته ویژه دارد، و این معنای مبتنی بر معناشناسی برای فایل سیستم و سازمانش تکامل یافتند. در ابتدا، معنای مبتنی بر معناشناسی به‌سادگی ترتیب همزمانی در یک فایل سیستم را نشان داده است. معناشناسی‌های داده‌هایی که با ساختار سروکار دارند منتج به سازمان‌دهی اضافی فایل‌ها با استفاده از رکوردها به‌عنوان واحدهای اساسی سازمان برای اطلاعات موردنیاز به برنامه‌های کاربردی و برای ذخیره‌سازی محیطی شده است. رکوردها مکانیسمی را فراهم آوردند که از آن ساختارهای ذخیره-سازی پیچیده‌تر ساخته می‌شود. رکوردها به دانه‌بندی مخزن مورداستفاده برای ساخت سازمان فایل و این طرح‌های دسترسی تبدیل شدند. یافتن یک رکورد درون یک فایل آسان شد، زیرا فایل‌ها متشکل از مجموعه‌ای از سوابق بودند. از طریق چنان ابزارهایی، کنترل‌های دسترسی همچون تکنیک‌های قفل رکورد به شکلی تکامل یافتند که نحوه یافتن امکان دسترسی به این فایل‌ها و سوابق لحاظ شده را کنترل کنند.

به‌مرور رکوردها، که در فایل‌هایی گروه‌بندی شده بودند، معنای معناشناسی اضافی گرفتند و به نقطه محوری برای سازمان‌دهی اطلاعات تبدیل شدند. به‌عنوان‌مثال، برای تعریف گروهی از دانش‌آموزان، مجموعه‌ای از رکوردها را می‌توان تعریف نمود به‌طوری‌که هر رکورد اطلاعات موردنیاز برای تعریف یک دانش‌آموز واحد را داشته باشد. به‌منظور سازمان‌دهی دانش‌آموزان به شکلی که برنامه‌های کاربردی بتواند از آن‌ها استفاده نماید، یک فایل سیستم می‌تواند یک بخش از یک فایل را برای ذخیره‌سازی این رکوردها تخصیص دهد یا می‌تواند ابزاری برای ارتباط دهی رکوردهای مرتبط در یک زنجیره با استفاده از یک استراتژی کنترل فراهم بیاورد.

این مفهوم ساختاری متمرکز حول اطلاعات منتج به یکی از اولین مفاهیم ذخیره‌سازی سیستم پایگاه داده و طرح‌های دسترسی گردید، که با عنوان مدل پایگاه داده شبکه شناخته می‌شود. مدل پایگاه داده شبکه داده‌ها را به‌صورت لیست‌های مرتبط یا زنجیره‌هایی از اطلاعات مرتبط سازمان‌دهی می‌کند. در مدل داده‌های شبکه هر اطلاعاتی که رابط‌های با یک اطلاعات ذخیره‌شده دیگر داشته باشد باید یک رابطه فیزیکی با اطلاعات مرتبط داشته باشد. مدل ساختاربندی پایگاه داده شبکه در استاندارد زبان پایگاه داده CODASYL فرمالیته شده بود و به شکلی وسیع پیاده‌سازی شد ولی هرگز به‌عنوان یک استاندارد واقعی موردپذیرش قرار نگرفت. سیستم‌های پایگاه داده شبکه به‌تدریج در این میان، از اواسط تا اواخر دهه ۱۹۷۰ و اوایل دهه ۱۹۸۰، به خاطر پیچیدگی و محدودیت‌های ذاتی‌شان رونق خود را از دست دادند. این مدل شبکه نیازمند آن است که درصورتی‌که یک رابطه منطقی بین اطلاعات نیاز باشد اطلاعات به‌صورت فیزیکی ارتباط دهی شوند. این به‌طور ضمنی اشاره به این داشت که با افزایش روابط منطقی بین آیتم‌های اطلاعاتی تعداد موردنیاز ارتباطات فیزیکی برای ثبت این روابط منطقی نیز افزایش‌یافته است.

این نیاز به افزایش ابر داده‌ها موجب افزایش تصاعدی ابعاد پیچیدگی شد، و باعث گردید که این مدل گزینه‌ای ضعیف برای هر سیستمی باشد که می‌تواند به‌مرورزمان رشد یافته و تغییر کند. فقدان یک ارتباط واحد می‌توانست باعث شود که پایگاه داده برای کاربرد اصلی که برای آن توسعه داده‌شده بود بی‌فایده شود. پیچیدگی زنجیره‌های ساخته‌شده درون یک برنامه کاربردی به‌مرورزمان باعث بسیار پیچیده‌تر شدن چنان سیستم‌هایی می‌شود. با یک عامل تعیین‌کننده دیگر برای این مدل پایگاه داده وقتی برخورد می‌کنیم که فرد برای دسترسی به اطلاعات ذخیره‌شده درون این مدل داده‌ها تلاش نماید. به‌منظور دسترسی یافتن به اطلاعات، پایگاه داده باید در یک نقطه ورود خاص مورد دسترسی قرار بگیرد، که پس‌ازآن نوبت به پیمایش زنجیره‌های (مسیرهای) داده‌های تعریف‌شده توسط روابط رمزنگاری‌شده توسط روابط رمزنگاری‌شده بین آیتم‌های داده‌ها می‌رسد. این بدان معنا نیست که اطلاعات موردنیاز یافته خواهند شد، این مسیرها ممکن است پیموده شوند و درنهایت مسیر هیچ داده‌ای یافت نشود. هیچ راهی برای دور زدن مسیرها وجود ندارد. برای یافتن آیتم‌های خاص داده‌ها، اگر قرار بر یافتن اطلاعات باشد، باید مسیری که منتج به این آیتم می‌شود را بپیماید نه مسیر دیگری را.

این‌ها و دیگر محدودیت‌های پایگاه داده شبکه برای انتقال تدریجی مدل، مدل‌سازی شدند. یک مسئله که باید در رابطه با مدل شبکه در نظر گرفته شود قدیمی بودن آن است. اگرچه این مدل، مدل غالب کاربردهای جدید در طی ۲۰ سال اخیر نبوده است، ولی همچنان پایگاه داده‌های بسیاری از این مدل ساخته می‌شوند که علت آن ورود زودهنگام و استفاده طولانی‌مدت در جامعه اطلاعاتی است. بسیار بعید است که همه یا حتی اکثر این اطلاعات در یک مدل داده‌های جدید همچون مدل رابطه‌ای میزبانی مجدد شوند. به علت حجم بالای اطلاعات قدیمی، این مدل باید از اثراتش برگذشته، حال و آینده سیستم‌های مدیریت اطلاعات درک شود. سیستم‌های جدید، درصورتی‌که به برای سیستمشان دسترسی داشته باشند، احتمالاً ملزم به تراکنش با چنان سیستم‌های قدیمی خواهند بود، که درک تأثیرشان بر عملکرد را الزامی می‌سازد.

انتقال سیستم پایگاه داده شبکه با توسعه و انتشار مدل پایگاه داده رابطه‌ای Codd و اولین مقاله منتشرشده در اوایل دهه ۱۹۷۰ آغاز شد. پیش‌فرض اساسی این مقاله این بود که تمام اطلاعات در سیستم پایگاه داده را می‌توان در جداولی شکل داد که روابط نامیده می‌شوند. این روابط ساختاری منظم دارند، که در آن هر ردیف جدول فرمت یکسانی دارد. روابط بین جداول با استفاده از مفاهیم یکپارچگی و محدودیت‌های ارجاعی تعریف‌شده‌اند. شیوه اساسی عملکرد بر روی این جداول از طریق تکنیک‌های حساب و جبر رابطه‌ای[۹] است. انتشار این مقاله با تبعیت از یک سیستم آزمایشی ساخته‌شده توسط IBM به نام سیستم R و یک سیستم دیگر توسعه داده‌شده توسط تحقیق دانشگاهی به نام Ingress بود. این توسعه‌های اولیه هدفشان اثبات تئوری‌های پایگاه داده رابطه‌ای است. این مدل رابطه‌ای روی کاغذ بسیار نویدبخش بود، ولی ساخت نرم‌افزار برای واقعی نمودن آن کاری ترسناک بود. یک تفاوت عمده اساسی در این دو مدل در مدلشان برای کسب داده‌ها یافت می‌شود. مدل شبکه یک مدل رویه‌ای است، که در آن‌یک کاربر به سیستم اطلاع می‌دهد که چگونه اطلاعات موردنیاز را بیابد، درحالی‌که مدل رابطه‌ای غیر رویه‌ای است، که در آن فرد ذکر می‌کند که چه چیزی را می‌خواهد و اجازه می‌دهد که "سیستم" اطلاعات را بیابد.

[۹] relational algebra

این تغییر در شیوه اساسی که پایگاه داده اطلاعات را می‌یابد یک مورد بسیار قابل‌توجه بوده است، که انشعابات آن همچنان بر اساس آن بهبود می‌یابد. این سرویس سیستم "پردازش پرس‌وجو" نامیده می‌شود. کارکرد اساسی پردازش پرس‌وجو، با داشتن پرس‌وجو یک کاربر، تعیین این است که چگونه باید درباره کسب اطلاعات درخواستی از روابط ذخیره‌شده در پایگاه داده اقدام شود. پردازش پرس‌وجو[۱۰] منتج به بهبود بیشتر در دسترسی به اطلاعات از پایگاه داده شد. یکی از پیشرفت‌های اولیه در بهینه‌سازی پرس‌وجو بود. هدف بهینه‌سازی پرس‌وجو یافتن راه‌هایی برای بهبود هزینه استخراج اطلاعات از پایگاه داده و انجام بی‌درنگ این کار است.

این سیستم‌های پایگاه داده رابطه‌ای در توسعه بسیاری از مفاهیم اساسی حول همزمانی دسترسی به سیستم‌های پایگاه داده نقشی اساسی داشتند. مفهوم دسترسی همزمان در پایگاه داده‌های اولیه مبتنی بر شبکه موجود نبود. نظریه تسلسل پذیری و کنترل همزمانی منتج به پیشرفت‌های بیشتری در تکنولوژی پایگاه داده شد. به‌طور خاص، مفاهیم برای تراکنش‌ها هم‌راستا با نظریه‌ها و مفاهیم بهبودی بودند. اصول بنیادین تراکنش‌ها و پردازش تراکنش این است که آن‌ها تحت کنترل ویژگی‌های "ACID" اجرا می‌شوند. این ویژگی‌ها الزام می‌کنند که تراکنش‌ها "به‌صورت خودکار" (همه‌چیز یا هیچ‌چیز)، "به شکلی یکدست" (تمام محدودیت‌ها بر صحت داده‌ها معتبر هستند)، "مجزا سازی شده" (تراکنش‌ها به صورتی اجرا می‌شوند که انگار به شکلی مجزا انجام‌شده‌اند)، و "بادوام" (اثرات اجرای تراکنش قابل‌تغییر نیستند مگر با اجرای یک تراکنش دیگر) اجرا شوند. تضمین این ویژگی‌ها نیاز به کنترل و بازیابی همزمان دارد.

مدل رابطه‌ای و پایگاه داده‌های رابطه‌ای در طی دهه ۱۹۸۰ ازنظر نوآوری و رشد درون صنعت پایگاه داده پیشتاز مسیر بودند. بخش عمده دهه ۱۹۸۰ صرف پالایش صحت برای پایگاه داده‌ها و برای عملکرد اساسی‌شان، یعنی تراکنش، شد. علاوه بر این پیشرفت‌های بنیادین، دهه ۱۹۸۰ شاهد پیشرفت قابلیت مدل‌سازی مدل بود.

پس‌ازاین دوره نوبت به دوره دیگری رسید، که ما دوره شیءمحور ذکر خواهیم کرد. در طی این دوره زمانی، یعنی اواخر دهه ۱۹۸۰ و اوایل دهه ۱۹۹۰، نیاز توسعه‌دهندگان برنامه‌های کاربردی برای تطابق دادن دقیق‌تر انواع داده‌های برنامه‌های کاربردیشان با موارد فراهم‌سازی شده توسط پایگاه داده باعث

[۱۰] Query Processing

بروز نیاز به غنای معنایی بیشتر مشخصه داده‌ها و عملیات برای این داده‌ها شد. پایگاه داده‌های شیء محور این دوره باید با این نیاز تطابق داشته باشند. مشکل این پایگاه داده‌های اولیه شیء محور این بود که آن‌ها برخی از مفاهیم بنیادین توسعه داده‌شده در طی تکامل و رشد سیستم‌های پایگاه داده رابطه‌ای‌شان را نداشتند.

در اواخر دهه ۱۹۹۰ و ابتدای قرن بیست و یکم مدل رابطه‌ای با مدل پایگاه داده شیءمحور نمودار شدند، که باعث شکل‌دهی مدل پایگاه داده رابطه‌ای شیی می‌شود. این مدل به‌عنوان مدلی که شایسته پالایش و پشتیبانی برای رشد است مورد استقبال نهادهای استانداردهای بین‌المللی و آمریکا قرار گرفت. فروشندگان عمده ملی و بین‌المللی از این مدل به‌عنوان تکامل پایگاه داده بعدی استقبال نموده بودند و در حال حاضر در حال شکل‌دهی محصولات حول این استاندارد جدیداً تصویب‌شده به‌وسیله برخی توسعه‌های خودشان هستند.

پس‌ازاین تکامل مشخص می‌شود که تغییر عمده بعدی در حوزه پایگاه داده احتمالاً در حوزه تراکنش‌ها و پردازش تراکنش خواهد بود. مدل متداول پیچیده حول مفهوم یک بخش اجرای تراکنش مسطح یا تک لایه‌ای که با ویژگی‌های ACID به‌شدت کنترل می‌شد ممکن است تغییر یابند. تحقیق و توسعه-های بسیاری وجود دارند که به دنبال مدل‌ها و نظریه‌های اجرایی جایگزین صحت هستند که می‌توانند ما را به دهه بعدی بهبودهای پایگاه داده سوق دهند.

۱-۳ تکامل سیستم‌های عامل

یک سیستم‌عامل مدرن عبارت است از نرم‌افزار کامپیوتر، سیستم‌عامل، و احتمالاً سخت‌افزاری که در یک سطح پایین با اجزاء سخت‌افزاری سیستم برای مدیریت اشتراک‌گذاری منابع کامپیوتری در میان برنامه‌های کاربردی نرم‌افزاری تراکنش می‌کند. هدف نرم‌افزار این مجموعه از سیستم‌ها فراهم‌سازی امکان اشتراک‌گذاری این منابع در میان هر یک و تمام مشاغل فعال درون سیستم است. یک سیستم‌عامل به‌صورت خاص‌ترین عناصر نرم‌افزاری بر روی سیستم اجرا می‌شود و نیاز به پشتیبانی سخت‌افزاری اساسی برای وقفه‌ها و زمان‌بندها برای تأثیرگذاری بر برنامه‌های اجرایی دارد.

سیستم‌های عامل تکامل‌یافته در یک دوره زمانی طولانی، عمدتاً با سخت‌افزار در دسترس به‌عنوان نیازهای برنامه‌های کاربردی که بر روی ماشین اجرا می‌شوند اجراشده‌اند. در ابتدا، ابزارهای معدودی

برای ارتقاء کارایی یک کامپیوتر برای عمومیت مردم وجود داشت، و آن‌ها به سطح استفاده توسط معدود افراد منتخبی تنزل داده‌شده بودند که ممکن بود در برگردان مسئله‌های واقعی به توالی‌های دستورالعمل‌های ماشین ساده با مشکل مواجه شوند. این دستورالعمل‌های ماشین در ابتدا در میکرو کد (پایین‌ترین شکل نرم‌افزار) یا کد اسمبلی بودند. در هر دو حالت، هیچ کنترلی بر آنچه کدر (رمز کننده) با سیستم‌عامل انجام می‌دهد وجود نداشت. این کدرهای اولیه نیاز به افرادی بسیار مستعد در کار تغییر یک مسئله همچون هدایت موشک به سمت نرم‌افزار موردنیاز برای اجرای آن داشتند. این کدرهای اولیه مشخصات نرم‌افزار را در یک مکان مشخص حافظه بارگیری می‌کردند و به سخت‌افزار اشاره می‌کردند که پردازش کار را آغاز کند. این ماشین نیاز به ادامه پردازش همین کار تا زمانی داشت که ماشین یک خطا را شناسایی می‌نمود (همچون یک سرریز) یا یک فرمان توقف وجود داشت که برای ماشین صادرشده است. هیچ ابزار خودکار دیگری برای رو آوردن از یک شغل به شغل دیگر وجود نداشت.

نخستین مسئله سیستم‌عامل که برنامه‌نویسان سیستم برای تغییر این وضعیت آن را رفع نمودند توسعه ابزاری برای انتقال از یک کار به یک کار پردازشی دیگر بدون نیاز به توقف ماشین، وارد کردن برنامه جدید، و شروع مجدد آن، به‌صورت مطرح در گذشته بود. مفهوم نمایشگر یا سیستم‌عامل دسته‌ای راهکاری برای این مسئله در ابتدا تعریف‌شده ارائه نمود. این سیستم‌های اولیه ابزارهایی را برای بارگیری چند کار در یک کار ارائه نمودند، سپس سیستم کامپیوتری آن‌ها را به شکلی متوالی اجرا نمود. با تکمیل یک کار، نرم‌افزار سیستم‌های عامل می‌توانند کنترل سخت‌افزار ماشین را بر عهده بگیرند، و آن را برای کار بعدی تنظیم نمایند، و سپس کنترل را مجدداً به کار جدید برگردانند، که سپس منتج به تکمیل شد. اگرچه این گامی در جهت مناسب بود، سیستم‌های کامپیوتری گران‌قیمت روز به شکلی مؤثر به‌کارگیری نشده بودند. دستگاه‌های جدید برای کمک به ورودی و خروجی (پایانه‌های اولیه) و مخزن (بهبود دیسک درایو، واحدهای نوار) در دست توسعه بودند، ولی مکانیسم‌های کنترلی برای استفاده از آن‌ها به شکلی مؤثر موجود نبودند.

این دستگاه‌های پیرامونی کامپیوتری، که در محل خودشان در دهه ۱۹۷۰ ارائه‌شده بودند، و محرکه‌ای برای طراحان سیستم‌ها فراهم آورد که راه‌هایی برای کاملاً کاربردی کردن آن‌ها درون سیستم بیابند. یکی از بزرگ‌ترین محرکه‌ها پایانه ورودی/خروجی بود. این نیازمند آن بود که این سیستم مکانیسم‌هایی برای اپراتورها برای وارد کردن کد و داده‌ها و درخواست گردآوری، ارتباط دهی، بارگیری و اجرای

کارهایشان را به صورتی فراهم بیاورد که به‌تنهایی بر روی ماشین اجرا شوند فراهم بیاورد، در زمانی که در واقع ممکن است کاربران بسیاری به طور همزمان بر روی ماشین کار کنند. این سرویس مدیریت سیستم که برای تحقق این مطالبات توسعه داده‌شده بود برنامه اجرائی نامیده شد.

این برنامه اجرائی خط‌مشی‌ها و مکانیسم‌هایی را برای برنامه‌ها و دستگاه‌هایی همچون پایانه‌ها فراهم آورد که به طور همزمان تحت کنترل چشم بینای اجرائی قرار بگیرند. کارکرد آن کنترل تراکنش به شکلی بود که دستگاه‌ها در اجرای کارهایشان بر روی ماشین تداخل نداشتند. بااین‌حال، آن‌ها همچنان تا حدود زیادی هرکدام در هر بار بر روی ماشین اجرا می‌شوند. این سیستم‌عامل خام بسیاری از سرویس‌های ابتدایی مورد انتظار از یک سیستم‌عامل را فراهم‌سازی نمود و به ابزاری تبدیل شد که بسیاری از ابداعات بر روی آن توسعه داده‌شده بودند.
تحقیق انجام‌شده بر روی این برنامه‌های اجرائی اولیه منتج به برنامه‌های ناظر گردید، که کارکردهای بیشتری را از اپراتورها و کدرهای سیستم‌ها بر عهده گرفت. برنامه‌های ناظر خدمات ابتدایی را برای "تبادل" برنامه‌ها از حافظه و کنترل اولیه بر روی CPU بر مبنای مفهوم برش‌های زمانی فراهم آوردند. پس از موفقیت این تحولات نخستین سیستم‌های عامل در دهه ۱۹۶۰ عرضه شد. بسیاری از سرویس‌های یافته شده در سیستم‌های عامل امروزی در این سیستم اولیه ریشه‌دارند.
از جنبه عمومی، یک سیستم‌عامل خدمات ذیل را ارائه می‌کند:

۱. مدیریت سخت‌افزار (بررسی وقفه، مدیریت تایمر)

۲. همگام‌سازی و ارتباطات بین مراحل

۳. مدیریت فرآیند

۴. تخصیص منابع (زمان‌بندی، توزیع)

۵. مدیریت ذخیره‌سازی و دسترسی (I/O)

۶. مدیریت حافظه

۷. مدیریت فایل

۸. حفاظت از سیستم و منابع کاربری

یک سیستم‌عامل با مدیریت سخت‌افزار یک سیستم کامپیوتری شروع می‌شود. مدیریت سخت‌افزاری نیاز به قابلیت تعیین محدودیت‌ها بر نگهداری منابع و توانایی کنترل از یک برنامه اجرائی به سیستم‌عامل است. این کارکردها از طریق استفاده از تایمرهای سخت‌افزاری و سرویس‌های وقفه محقق می‌شوند. یک تایمر سخت‌افزاری شمارنده‌ای است که می‌تواند برای یک شمارش مشخص تنظیم شود (دوره زمانی). وقتی زمان منقضی شد، یک سیگنال وقفه منتشر می‌شود، که پردازنده را متوقف می‌کند، وضعیت پردازنده را ذخیره می‌کند (تمام محتوای ثبات، ثبات‌های ALU، ثبات‌های وضعیت، اشاره-گرهای پشته[۱۱]، شمارنده‌های برنامه، ثبات‌های دستورالعمل و غیره) را ذخیره می‌کند، و کنترل را به یک روتین سرویس وقفه[۱۲] انتقال می‌دهد. روتین سرویس وقفه محتوای ثبات‌های از پیش تعریف‌شده (مثلاً ثبات وضعیت CPU یا یک ثبات وقفه از پیش تعریف‌شده) را بررسی می‌کند یا مکان‌های حافظه را تعیین می‌کند و تعیین می‌کند که چه عملیاتی باید بعداً انجام شوند. نوعاً، کنترل بلافاصله به هسته سیستم‌عامل برای سرویس‌دهی به وقفه تبدیل می‌شود.

اهداف این سرویس‌ها و توسعه‌ها یک موضوع[۱۳] مشترک داشتند: استفاده مؤثرتر از امکانات محاسباتی. آن‌ها باید رابط‌های آسانی را برای کاربران فراهم می‌آوردند و درعین‌حال جزئیات ماشین لخت[۱۴] (کامپیوتری که بدون مداخله سیستم‌عامل دستورات را مستقیماً روی سخت‌افزار منطقی اجرا می‌کند) را از آن‌ها مخفی می‌نمودند. این سیستم‌عامل امکان استفاده شفاف از منابع محاسباتی را فراهم می‌آورد، و کاربران و اپراتورها را از مشکل نیاز به اطلاع داشتن از پیکربندی سیستم خاص رها می‌کند. سیستم‌عامل این به کاربران و برنامه‌نویسان سیستم‌ها امکان حفاظت از تخریب، سرقت یا افشای غیرمجاز تصادفی یا بدخواهانه را داد.

---

[۱۱] stack pointers
[۱۲] interrupt service routine
[۱۳] thread
[۱۴] bare machine

بدیهی‌ترین دستاورد یک سیستم‌عامل مخفی نمودن جزئیات پلتفرم محاسباتی و استفاده بهینه از منابع بود. کاربران نیازی ندارند که چه دستگاه خاصی را استفاده می‌کنند، بلکه فقط به یکی از کلاس‌های مشخص دستگاه‌ها (مثلاً یک نوار یا دیسک) نیاز دارند. این کاربران را از مشکلات اجزاء پایین[۱۵] حفاظت می‌کند. اگر مشخص نمودن یک دستگاه خاص ضروری بوده باشد که در دسترس نبوده است، کار ممکن بود نتواند ادامه یابد. اگر کاربر بتواند کلاسی از دستگاه را مشخص نماید، هر یک از این موارد می‌تواند این نیاز را محقق نماید، و توانایی کاربر برای انجام موفق کار را افزایش دهد.

برنامه‌نویسان سیستم‌ها و پژوهشگران سخت‌افزار و نرم‌افزار جستجویشان برای کمال در این مقطع را تمام نکردند. جاهای بیشتری برای توجه کردن وجود داشت، و بخش‌های مسئله سیستم نیاز به داشتن راهکارهایی بودند که برای آن‌ها توسعه‌یافته باشد. اشتراک‌گذاری منابع مجموعه مشکلات خودش را به دنبال داشت. با کاربری‌پذیرتر شدن سیستم‌ها، استفاده‌های بیشتری متصور بود و پیاده‌سازی شد. سیستم‌ها به‌تدریج ظرفیت پردازشی خام ماشین‌ها را تأمین نمودند. طراحان نیاز به یافتن راه‌هایی برای برنامه‌های کاربردی کاربری داشتند. توسعه‌دهندگان نرم‌افزار به دنبال رفع پیچیدگی محاسباتی سیستم‌عامل بوده‌اند، که قدری مؤثر واقع شده است. طراحان سخت‌افزار ظرفیت محاسباتی سیستم‌ها را از طریق بهبود معماری‌ها و طرح‌های اجرای دستورالعمل بهبود دادند. بااین‌حال، تمام چنین پیشرفت‌هایی، فقط موقتی بودند.

جامعه توسعه و تحقیق به‌تدریج به دنبال راه‌هایی برای بهبود عملکرد پردازنده به‌صورت ثابت یا حاشیه‌ای گشته است. مسئله این است که اهمیتی ندارد که ما چقدر عملکرد یک پردازنده را بهبود بدهیم، چون همچنان مقدار محدودی از چرخه‌های در دسترس را برای برنامه‌های کاربردی و سرویس‌های سیستم‌های موردنیاز خواهد داشت. مفهوم اولیه مدنظر رشد تنها یک توان پردازشی واحد، که محدود است، نبود، بلکه در عوض افزودن پردازنده‌های کاملاً جدید بود. این مفهوم در ابتدا به‌عنوان بخشی از بهبودهای معماری در دهه ۱۹۸۰ موردبررسی قرار گرفت. هر یک از پردازنده‌های متعدد را می‌توان به شکلی تنظیم نمود که سیستم‌های عامل خودشان را اجرا نمایند، می‌توان هر یک از سیستم‌های عامل را به شکلی طراحی نمود که سیستم‌های عامل خودشان را با سرویس‌های بیشتر اجرا نمایند تا سیستم‌های دوردست بتوانند سرویس‌ها (منابع) را از ماشین دیگری درخواست نمایند که مشغول نبود یا به طور کامل

[۱۵] down components

مورداستفاده قرار نمی‌گرفت. با استفاده از این تماس‌های سیستم‌ها، پردازنده‌های متعددی که سیستم‌های عامل مجزا را اجرا می‌کنند را می‌توان به شکلی همگام‌سازی نمود که یک وظیفه پردازشی واحد بزرگ‌تر را در زمانی به‌مراتب کمتر انجام دهند. بااین‌حال، این پیشرفت مؤثر در عملکرد صرفاً، فاکتور ضرب تعداد ماشین‌ها نیست بلکه کسری از این محاسبه است. همگام‌سازی عملیات سیستم‌هایی که ارتباط چندانی باهم ندارند به علت بالاسری اضافی است.

این سیستم‌ها منتج به تحقیق و تجربه‌اندوزی بیشتری شدند. اگر بتوان ماشین‌هایی که ارتباط تنگاتنگی باهم ندارند را بتوان گردآوری نموده و باهم گروه‌بندی نمود تا کارکردهای بزرگ‌تر را انجام دهند که در تنها یک ماشین قابل انجام نیستند؟ این سیستم‌های جدید "پردازنده‌های توزیع‌شده" نامیده می‌شدند. آنچه این کلاس‌های سیستم‌ها را از همتایان چندپردازنده‌ای‌شان که ارتباط تنگاتنگی ندارند متمایز می‌کند درجه انسجامی است که پردازنده‌ها از خود نشان می‌دهند. سیستم‌عامل پردازنده یک سیستم‌عامل فراگیر واحد است، که به طرق مختلفی بر روی ماشین‌ها گسترده شده است. در یک مورد، کل سیستم‌عامل را می‌توان بر روی هر مکان بازتولید نمود، که در آن هر یک پردازنده‌ها تنها نیاز به اطلاعات حالت بیشتری دارند تا نشان دهند که کارکردشان چیست وضعیتشان در حال حاضر در رابطه با وضعیت کل سیستم‌های توزیع‌شده چیست. پیکربندی دوم از مفهوم تقسیم‌بندی اجزاء سیستم‌های عامل بر روی بخش‌های مختلف سیستم کامپیوتری توزیع‌شده استفاده می‌کند. ازاین‌رو هر پردازنده کارکردی ویژه دارد – به‌عنوان‌مثال، زمان‌بندی فرآیند یا دسترسی دستگاه.

این سیستم‌های عامل جدید همچنان در حوزه تحقیق موردبررسی قرارگرفته‌اند و هنوز راهشان را به سمت سیستم‌های جریان اصلی نگشوده‌اند. از سوی دیگر، پردازش کلاینت/سرور راداریم، که از شکلی از سیستم‌های عامل چندپردازنده‌ای برای فراهم‌سازی دسترسی از راه دور به منابع استفاده می‌کند. بااین‌حال، تفاوت آن‌ها در این است که الزامات همگام‌سازی شدیدی را بر پردازش کلاینت/سرور اعمال نمی‌کنند. بسیاری از پروتکل‌های اضافی برای فراهم‌سازی این شکل از پردازش توسعه داده‌شده‌اند، که در بیشتر محصولاتی که امروزه برای محاسبه از راه دور در محیط وب استفاده می‌کند رواج دارد.

۱-۴ تکامل شبکه‌های کامپیوتری

واژه شبکه می‌تواند به معنای چیزهای متفاوتی باشد. این می‌تواند به طور ضمنی به یک اتصالات داخلی مسیرهای راه‌آهن برای شبکه ریلی، بزرگراه‌ها و خیابان‌ها برای شبکه‌های حمل‌ونقل، خطوط تلفن و مراکز سوئیچینگ برای شبکه تلفن، خطوط هم‌محور برای شبکه تلویزیون کابلی، خطوط فیبر برای شبکه‌های ارتباطات کابلی، یا پیوستگی مراکز سرویس، کسب‌وکار، و امثالهم برای شکل‌دهی یک شبکه اشاره داشته باشد. تمام این پیکربندی‌ها به ابزارهایی برای به هم ارتباط دادن منابع مختلف اشاره دارند تا بتوانند به‌صورت یک گروه عمل نمایند، و منافع اعداد، اشتراک‌گذاری و ارتباطات در چنان گروهی را محقق نمایند.

در سیستم‌های کامپیوتری، مجموعه اصطلاحات یک شبکه ترکیبی از تجهیزات و برنامه‌های محاسباتی به‌هم‌پیوسته مورداستفاده برای اطلاعات (و محاسبه‌ها) بین نقاط (گره‌ها) در شبکه‌ای است که می‌تواند تولیدشده، پردازش شود، ذخیره شود یا به هر شکلی که مناسب فرض شود مورداستفاده قرار بگیرد. این به‌هم‌پیوستگی می‌تواند شکل‌های بسیاری را به خود بگیرد، همچون پیوندهای اختصاصی، پیوندهای اشتراک گذاشته‌شده، خطوط تلفنی، پیوندهای ماکروویو، و پیوندهای ماهواره‌ای. شبکه‌ها از این جنبه یک همبستگی ضعیف دستگاه‌هایی را شک می‌دهند که اطلاعات را به اشتراک می‌گذارند. این‌یکی از نخستین کاربردهای یک شبکه بود، هرچند که آخرین آن‌ها نبود. کاربران دریافتند که این شبکه می‌تواند چیزی بیش از صرفاً اشتراک‌گذاری اطلاعات را ارائه نماید، و می‌تواند سرویس‌های دیگری را برای دورکاری و نهایتاً محاسبه توزیع‌شده (پراکنده) فراهم بیاورد.

اولین مفهوم یک شبکه پیوند نه‌چندان محکم دستگاه‌ها به یکدیگر یا منابع برای اشتراک‌گذاری بود. یک شبکه ارتباطات کامپیوتری اولیه که این ویژگی‌ها را از خود نشان داد ARPANET بود. ARPANET نخستین بار در سال ۱۹۶۹ به‌عنوان یک ابزار تحقیقاتی برای بررسی مسائل شبکه دوربرد و فراهم‌سازی ابزاری برای راهکارهای تحقیق و توسعه خریداری شد. این شبکه به اینترنت تکامل‌یافته است، میلیون‌ها کامپیوتر را از طریق شبکه‌های ناحیه محلی، شبکه‌های شهری، و دیگر شبکه‌های گسترده به هم مرتبط نموده است. ARPANET ابزار لازم برای تحقیقات اولیه در زمینه پروتکل‌های ارتباطی که با تراکم، کنترل، مسیریابی، آدرس‌دهی، فراخوانی از راه دور، محاسبه

توزیع‌شده، سیستم‌های عامل و خدمات توزیع‌شده و بسیاری از حوزه‌های دیگر سروکار دارند را فراهم آورده است.

دلایل استفاده از شبکه‌هایی همچون invocation فراهم‌سازی دسترسی‌پذیری وسیع به طیف وسیعی از دستگاه‌ها بود. برنامه‌های کاربردی اولیه کامپیوتری با انجام امور مهندسی و کارکردهای عمده پردازش داده‌ها سروکار داشتند. با تغییر تکنولوژی کامپیوترها، و با افزودن بیشتر و بیشتر برنامه‌های کاربردی توسط محققان و کاربران به شکلی مشابه، دسترسی به اطلاعات و دست‌کاری آن مورد تأکید بیشتری قرار گرفت.

شبکه‌های اولیه سرویس‌های تبادل اطلاعات ضروری را فراهم‌سازی نمودند ولی اساساً فقط این سرویس را ارائه نمودند. این دسترسی‌پذیری اطلاعات استفاده‌های متصور بیشتر از این اطلاعات را برمی‌انگیزد. با وقوع این و پیشرفت تکنولوژی شبکه‌ها، برنامه‌های کاربردی جدیدی ظهور یافتند. این برنامه‌های کاربردی جدید علاوه بر استفاده از تبادل اطلاعات از دورکاری نیز استفاده می‌کردند. این به‌سادگی به‌صورت ارسال کار به لینک به یک میزبان کم‌کارتر آغاز می‌شد، کار در آنجا تکمیل می‌شد، و سپس نتایج به مبدأ برگردانده می‌شدند.

این تا مدتی کفایت می‌کرد، ولی همچنان محیط‌های بی‌درنگ و تراکنشی را فراهم نمی‌آورد که کاربران به‌تدریج به آن خو گرفته بودند، از جمله پروتکل‌ها و سیستم‌های عامل شبکه پیشرفته‌تر تا سرویس‌های بعدی برای فراخوانی کار از راه دور و همگام‌سازی فراهم بیاید. دوران شبکه محلی رو به آغاز بود. بزرگ‌ترین نقیصه شبکه‌های گسترده[۱۶] ازنظر بازدهی یا دوره بازگشت برای کارها و ارتباطات بین‌پردازنده‌ای بود. به علت فواصل وسیع، تأخیرهای چندثانیه‌ای رواج داشتند و موجب بالاسری اضافی در اجرای کارهای غیر ساده می‌شدند. طراحان شبکه شاهد نیازی به فراهم‌سازی پیوندی دیگر در این شبکه بودند: شبکه محلی[۱۷].

شبکه‌های محلی، همانند بسیاری از فعالیت‌های تحقیقاتی در دانشگاه‌ها و لابراتوارهای دولتی، در اوایل تا اواسط دهه ۱۹۷۰ نمودار شدند. از اواسط دهه ۱۹۷۰ که اترنت[۱۸] عرضه شد بود که LAN ها به طور

---

[۱۶] wide area networks

[۱۷] local area network

[۱۸] Ethernet

گسترده در دسترس قرار گرفتند. از آن زمان، طراحی‌های متعدد LAN برای جای گرفتن در طیف فوق‌العاده وسیعی از الزامات کاربری تولید شده‌اند، به‌عنوان‌مثال، حلقه فیبر[۱۹]. علاوه بر این، استانداردهایی تکامل یافته‌اند که توپولوژی‌های LAN اساسی و سرویس‌هایشان را برای تعداد بیشتری از کاربران فراهم آورده‌اند.

شبکه‌های محلی در حال راه یافتن به تمام جنبه‌های جامعه مدرن هستند. ما آن‌ها را در خانه‌هایمان از طریق مودم‌های کابلی و مودم‌های تلفن، خودروها از طریق تکنولوژی‌های بی‌سیم، بانکداری (مثلاً ATM ها)، مدارس از طریق پیوندهای اینترنتی، کسب‌وکارها، دولت و صنعت می‌یابیم. جنبه‌های تبادل اطلاعاتی و پردازش داده‌ای چندانی وجود ندارد که در آن‌ها نتوان LAN را یافت. شبکه‌های محلی و تکنولوژی‌های مرتبطشان یکی از حوزه‌های رشد عظیم دهه ۱۹۹۰ و اوایل دهه ۲۰۰۰ را به خود اختصاص داده‌اند. با قابل دسترس شدن هر چه بیشتر LAN ها، محصولات و استفاده‌های جدید برای آن‌ها هم قابل دسترسی می‌شود. LAN ها برای اتصال تمام کامپیوترهای شخصی در ادارات، کلاس‌ها، کارخانه‌ها، خرده‌فروشی‌ها، و حتی اکنون در بسیاری از خانه‌ها استفاده می‌شوند. آن‌ها در این محیط‌ها برای ارسال تفاهم‌نامه‌ها، صدور رهنمودها، برنامه‌ریزی برای جلسات، انتقال اسناد، ارسال ایمیل، کشف اطلاعات جدید، و پردازش حجم بالایی از داده‌ها به طور همزمان در بسیاری از جاها استفاده می‌شوند. LAN ها برای ارتباط دادن ربات‌های کارخانه‌ها به هم با کنترلرهای محلی و کارخانه‌ها استفاده می‌شوند. آن‌ها داده‌های حسگر، کنترل داده‌ها، و بازخورد به مراکز کنترلی را در اختیار قرار می‌دهند، و همزمان ابزاری برای صدور تغییرات و اعلان‌های تولیدی برای کاربران و ربات‌ها به طور مشابه فراهم می‌آورند. نمونه خوبی از یک شبکه محلی که سرویس‌های مواریانسی را برای کاربران فراهم می‌آورد در "دنیای والت دیسنی" دیده می‌شود. دیسنی از LAN ها و کامپیوترها برای مانیتور تمام جنبه‌های سرویس‌ها، از جمله حفاظت، زمان‌بندی، مدیریت حرکت، اطلاعات آنلاین، امنیت، خدمات پرسنلی، و مجموعه وسیعی از دیگر امور مربوط به مدیریت پارک استفاده می‌کند. بانک‌های بزرگ، همچون بانک جهانی، LAN ها را به‌عنوان ابزارهایی برای به‌هم‌پیوستگی سایت‌های محلی‌شان به شبکه‌های کوچک‌تر که توسط شبکه‌های گسترده به هم ارتباط داده‌شده‌اند به‌کارگیری نموده است. بااین‌حال، LAN برای همگان نیست.

[۱۹] fiber ring

تکامل شبکه در اینجا متوقف نشده است. با پیشرفت تکنولوژی علاقه ارائه‌دهندگان خدمات شبکه‌ای برای ارائه سرویس‌هایشان به کاربران این حوزه‌ها نیز افزایش‌یافته است. شبکه‌های بی‌سیم نسبتاً بی‌سروصدا در دهه ۱۹۷۰ با شبکه آلوها[۲۰] به‌عنوان مبنا در دسترس قرار گرفتند. از آن زمان، توسعه شبکه تلفن بی‌سیم در را به روی شبکه‌های کامپیوتری گشوده است. امروز یکی از حوزه‌های رشد بزرگ در شبکه توسعه بیشتر شبکه‌های بی‌سیم و تلفیق این‌ها در شبکه‌های موجود LAN و WAN به‌منظور فراهم‌سازی محدوده وسیع‌تری از کارکردها برای مجموعه تلفن همراه بی‌سیم است.

۱-۵ نیاز به ارزیابی کارایی

انتخاب یک معماری کامپیوتر خاص برای یک برنامه کاربردی، یک سیستم‌عامل، یک سیستم پایگاه داده یا یک شبکه گسترده یا محلی که خدمات بهینه را برای کاربران فراهم خواهد آورد نیازمند تحلیل پیشاپیش و دانش است. همان‌گونه که اشاره شد، یک معماری کامپیوتر خاص، سیستم‌عامل، پایگاه داده و / یا LAN ابزارهای ارتقادهنده بهره‌وری هستند، ولی همانند ابزارهای دیگر، اگر به‌درستی استفاده نشوند، درواقع می‌توانند بهره‌وری را کاهش دهند. یک سیستم‌عامل می‌تواند یک ابزار را برای افزایش همزمانی دسترسی فراهم بیاورد و کاربری کلی منبع سیستم را افزایش داده یا می‌تواند با ممانعت از دسترسی به تنگنایی تبدیل شود. یک سیستم پایگاه داده می‌تواند ابزاری برای اشتراک‌گذاری کاراتر اطلاعات میان برنامه‌های کاربردی به شکلی درست و همزمان فراهم بیاورد یا می‌تواند موجب مسدودسازی گسترده اطلاعات از طریق حذف دسترسی‌پذیری داده‌ها شود. یک LAN می‌تواند ابزاری برای تسهیل پردازش اطلاعات فراهم آورده و موارد اضافی را حذف کند، ولی این می‌تواند به خاطر مشکلات پروتکل یا ارتباطی مانع از لاگینگ کاربران شود. برای کاربران عادی، مدیریت منابع سیستم‌های عامل، پردازش داده‌ها، استخراج داده‌ها، ارتباطات داده‌ها، و شبکه‌های محلی یک سیاه‌چاله پروتکل‌ها، طرح‌های دسترسی، الگوریتم‌های روتینگ، مسائل کابل بندی و توپولوژی، و مسائل خدماتی هستند. به‌منظور کاستن از این مسائل، باید به کاربران درباره مبانی معماری کامپیوتر، سیستم‌های عامل، سیستم‌های پایگاه داده و تکنولوژی شبکه محلی آموزش داده شود و معیارها و ابزارهایی در اختیارشان

[۲۰] Aloha

قرار داده شود که با آن‌ها می‌توانند به شکلی مناسب برای عبور از مشکلات متعدد و انتخاب یک سیستم کامپیوتری نقشه‌برداری شده برای نیازهایشان کمک نماید.

وقتی شما به گزینه‌های متعدد در دسترس برای خریداران احتمالی آتی سیستم‌های کامپیوتری برای ارزیابی نگاه می‌کنید، می‌توانید دلایل پریشانی آن‌ها را ببینید. یک سیستم کامپیوتری می‌تواند بسیار ساده باشد، و تنها یک واحد پردازنده مرکزی، یک بانک حافظه اولیه، یک کانال I/O برای دسترسی به دستگاه‌های جانبی، و یک رابط شبکه را برای اتصال به ماشین دیگر فراهم بیاورد. در مقابل، سیستم کامپیوتری می‌تواند بسیار پیچیده باشد، و دارای پردازنده‌های متعدد، حافظه کش، یک سیستم حافظه انجمنی با تکنولوژی بالا، بانک‌های دیسک کنترل‌شده SCSI، موتورهای گرافیکی ویژه، و احتمالاً سیستم‌عامل، پروتکل‌ها و سرویس‌های توزیع‌شده خودش باشد. خریدار موردنظر سیستم کامپیوتری باید تصمیم بگیرد که چه انواعی از مادر بورد(ها) لازم است و به چه تعداد، چه نوع از حافظه و معماریش، چه شکلی از سیستم‌عامل، سیستم پایگاه داده، و کابل بندی شبکه لازم است، و چه انواعی از مشخصه‌های الکتریکی، طرح سیگنالینگ، پروتکل انتقال کنترلینگ، طرح‌های روتینگ، توپولوژی پیوستگی، الزامات پایایی، تحمل خطای سیستم و اجزاء در صورت لزوم، سرویس‌ها، مشخصه‌ها و الزامات رابط، و جنبه‌های متعددی دیگر. گستره کنترل، درک و تطابق‌پذیری با تجهیزات دیگری که یک کاربر به آن‌ها نیاز دارد مشخص خواهند کرد که کدام‌یک از این و سایر مسائل باید پیش از خرید یک سیستم کامپیوتری مورد چاره‌جویی قرار بگیرند.

۱-۶ نقش ارزیابی در مهندسی کامپیوتر

در حال حاضر توجه و فعالیت زیادی در زمینه طراحی و استفاده از سیستم‌های کامپیوتری، همچون و معماری‌های متمرکز، بردار، موازی، توزیع‌شده، کلاینت / سرور. این سیستم‌های کامپیوتری در دست تحقیق، توسعه، تولید و عرضه توسط افراد و سازمان‌های دولتی، صنعتی و دانشگاهی هستند. این فعالیت‌های تحقیق و توسعه انگیزش یافته از تکنولوژی‌های به‌سرعت در حال تغییر دستگاه‌ها، نرم‌افزارها و سیستم‌ها، افزایش الزامات کارایی، افزایش پیچیدگی آن‌ها، دستگاه‌های جانبی، اتصالات و کنترل، درخواست ثابت برای بهبود قابلیت اطمینان و دسترسی‌پذیری، و افزایش اتکای سازمان‌ها بر استفاده از امکانات کامپیوتری در تمام جنبه‌های کسب‌وکار هستند.

سیستم‌های کامپیوتر دسکتاپ فعلی قابلیت‌های بیشتری را نسبت به آنچه قبلاً در یک سیستم واحد اشتراک زمانی واحد وجود داشت فراهم می‌آورند. به‌طور خاص، برخی از ویژگی‌ها دربرگیرنده اشتراک‌گذاری منابع بر روی یک مقیاس به‌مراتب جهانی‌تر و این تحقق الزاماتی همچون بسط پذیری، انعطاف‌پذیری، دسترسی‌پذیری، پایایی، قابلیت تغییر پیکربندی، تحمل خطا، زوال برازنده، پاسخ‌دهی، سرعت، ظرفیت بازدهی، پیچیدگی منطقی، و *سهولت* توسعه (مدولاریته) هستند. یک ویژگی جذاب دیگر سیستم‌های کامپیوتری معاصر قابلیت آن‌ها برای ارائه توان محاسباتی به *کاربر* بدون قربانی کردن قابلیت لازم برای کسب تمام دارایی‌های اطلاعاتی در دسترس از کسب‌وکار است.

ازاین‌رو، درصورتی‌که قرار بر آن باشد که تأسیسات محاسباتی هدف خدمات جدید و بهبودیافته‌ای را برای آنچه در حال حاضر برای کاربرد حوزه هدف در دسترس است فراهم بیاورد، طراحی و/یا انتخاب بهینه یک سیستم کامپیوتری واجد بیشترین اهمیت خواهد بود. ولی چگونه می‌توان این کار را انجام داد؟ برای این هدف چه تکنیک‌ها و ابزارهایی در دسترس هستند؟ این‌ها برخی از پرسش‌هایی هستند که در این کتاب برای معمار، محقق، طراح، خریدار یا دانشجوی سیستم‌های کامپیوتری موردبررسی قرار خواهند گرفت. این برای پوشش دادن موارد ضروری مدل‌سازی و تحلیل محیط‌های سخت‌افزار و نرم‌افزار سیستم‌های کامپیوتری است. مباحث *تحت پوشش* قرارگرفته دربرگیرنده تکنولوژی‌های پایه مرتبط با سخت‌افزار، نرم‌افزار و شبکه‌بندی سیستم‌های کامپیوتری، جزئیات تکنیک‌های مدل‌سازی مورداستفاده برای پیکربندی‌های مطالعه سخت‌افزار و نرم‌افزار کامپیوتری، و توصیف ابزارهای نرم‌افزاری که برای مدل‌سازی چنین سیستم‌هایی استفاده‌شده‌اند می‌شوند.

۱-۷ مرور اجمالی روش‌های ارزیابی عملکرد

مدل‌ها ابزاری برای تعریف یک سیستم و مسائل آن به شکلی مختصر فراهم می‌آورند، آن‌ها ابزارهایی برای تعیین عناصر حیاتی، قطعات، و مسائل مربوط فراهم می‌آورند، آن‌ها ابزاری برای ارزیابی طراحی‌ها یا ساخت و ارزیابی راهکارهای پیشنهادی فراهم می‌آورند، و آن‌ها را می‌توان به‌عنوان پیش‌بینی‌هایی برای آینده‌نگری و کمک به ارتقاء یا توسعه طراحی آتی استفاده نمود. به بیانی مختصر، آن‌ها یک محیط آزمایشگاهی را فراهم می‌آورند که در آن‌یک سیستم حتی *پیش‌ازآنچه* موجود باشد یا بدون درواقع اثرگذاری بر یک پیاده‌سازی واقعی یک محیط آزمایشگاهی را فراهم می‌آورد. مدل‌ها نوعاً بر مبنای

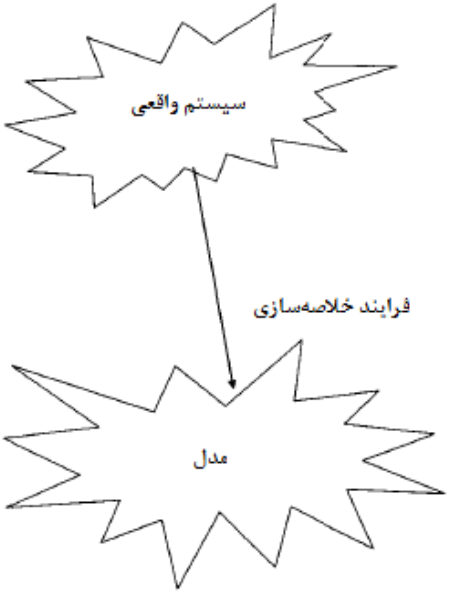
قوانین و اصول تئوریک توسعه داده می‌شوند. آن‌ها می‌توانند مدل‌های فیزیکی (نسخه‌های مشابه مقیاس بندی شده)، روابط و معادلات ریاضی (انتزاعات)، یا نمایش‌های گرافیکی را شامل شوند. مدل‌ها فقط به‌اندازه اطلاعاتی مفید واقع می‌شوند که در اختیارشان قرار می‌گیرند. بدین معنی که، مدل‌سازی یک سیستم در صورتی آسان‌تر و نوعاً *بهتر* است که:

- قوانین فیزیکی در دسترس هستند که برای توصیف آن در دسترس هستند.
- نمایش‌های تصویری (گرافیکی) را می‌توان برای دستیابی به درکی *بهتر* از مدل ارائه نمود.
- ورودی‌ها، عناصر و خروجی‌های سیستم بزرگی قابل مدیریتی دارند.

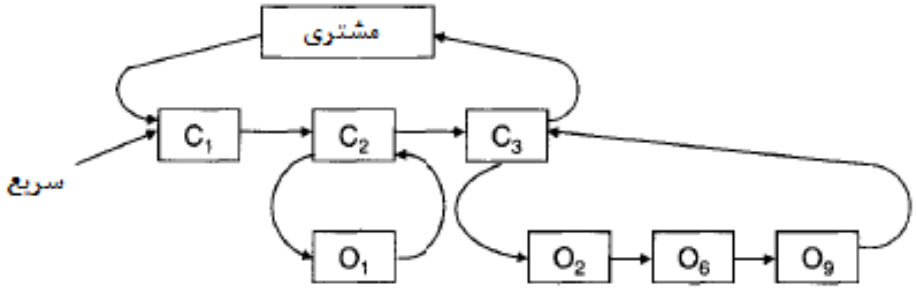
همه این‌ها ابزاری برای ساخت و تحقق مدل‌ها فراهم می‌آورند، ولی مسئله معمولاً این است که ما قوانین فیزیکی مشخصی برای تبعیت نمودن نداریم، توصیف تراکنش‌ها ممکن است *بسیار* دشوار باشد، تصادفی بودن سیستم، محیط یا کاربران موجب بروز مشکلاتی می‌شود، و خط‌مشی‌هایی که محرکه فرآیندها هستند به‌سختی قابل‌سنجش هستند. آنچه معمولاً انتشار می‌یابد این است که یک مدل "وفادار" یک سیستم ساخته‌شده است: مدلی که بینش‌هایی درباره جنبه حیاتی یک سیستم، نه تمام اجزائش، فراهم می‌آورد. بدین معنی که ما نوعاً برشی از سیستم واقعی را مدل‌سازی می‌کنیم. مفهوم ضمنی آن این است که این مدل خلاصه از سیستم دنیای واقعی تحت مطالعه است. در مورد تمام انتزاعات، باید تصمیم گرفته شود که چه عناصری از دنیای واقعی را در انتزاع بگنجاند – بدین معنی که کدام‌یک برای در نظر گرفتن به‌عنوان یک مدل "وفادار" مهم هستند. آنچه ما در اینجا درباره‌اش صحبت می‌کنیم شهود است – یعنی یک مدل کننده تا چه حدی می‌تواند عناصر مهم را انتخاب نماید، این عناصر را تا چه میزان می‌توان تعریف نمود، و تراکنش این عناصر قابل‌توجه درون خودش، میان خودشان، و با دنیای خارج به چه میزان است.

۱-۷-۱ مدل‌ها

همان‌گونه که قبلاً ذکر شد، یک مدل خلاصه از یک سیستم است (شکل (۱-۱۱) را ببینید). واقع‌گرایی این مدل مبتنی بر سطح انتزاع اعمال‌شده است. بدین معنی که اگر ما بدانیم که همه‌چیز درباره یک سیستم است و مایل به پرداخت برای پیچیدگی ساختمان یک مدل حقیقی باشیم، این انتزاع نزدیک به صفر است. از سوی دیگر، در بیشتر موارد، ما تمایل داریم که دیدی که از یک سیستم داریم را برای ساده‌سازی پیچیدگی‌ها خلاصه‌برداری کنیم. ما مایلیم تا مدلی را ایجاد نماییم که بر برخی از عناصر موردتوجه تمرکز دارد و مابقی سیستم تنها رابطی بدون جزئیات برای ورودی‌ها و خروجی‌های مناسب بماند.



شکل (۱-۱۱): پردازش مدل‌سازی



شکل (۱-۱۲): خلاصه‌سازی یک سیستم

"سیستم" به شکلی که ما آن را ذکر نموده‌ایم، دنیای واقعی است که تمایل به مدل‌سازی آن داریم (مثلاً یک دستگاه خودپرداز، یک کارواش، یا یک آیتم یا فرایند ملموس دیگر). در شکل (۱-۱۲) یک سیستم گروهی یکپارچه‌ای از اشیاء لحاظ می‌شود که برای اجرای برخی از کارکردها یا فرایندهای تنظیم‌شده یکپارچه‌سازی شده‌اند، درحالی‌که یک مدل خلاصه‌ای از سیستم است که آیتم‌های مهم و تراکنش‌های آن را استخراج می‌کند.

مفهوم اساسی این بحث این است که یک مدل دیدگاه ذهنی یک مدل کننده از سیستم است. این دیدگاه آنچه مهم است، هدف، جزئیات، مرزها و دیگر موارد آن را تعریف می‌کند. مدل کننده باید سیستم را درک کند تا یک نظرگاه وفادارانه درباره ویژگی‌های مهم آن فراهم بیاورد و مدل را قابل‌استفاده نماید.

۱-۷-۲ ساخت مدل

به‌منظور ساخت یک مدل، ما به‌عنوان مدل کننده‌ها باید از متدلوژی‌های قابل پیش‌بینی برای استخراج نمایش‌های صحیح تبعیت نماییم. متدلوژی مورداستفاده رایج از تجزیه بالا به پایین تشکیل یافته است و به هدف امکان تعریف هدف مدل یا جزء آن در هر گام و، بر اساس این هدف، به استخراج مرزهای سیستم یا جزء و توسعه سطح جزئیات مدل‌سازی تعلق دارد. این روش تکراری توسعه سطح هدف، مرزها و مدل‌سازی لبه‌های ناهموار یا غیرقابل تعریف سیستم یا جزء واقعی را نرم‌سازی می‌کند و بدین طریق بر عناصر حیاتی آن تمرکز دارد.

ورودی‌های این مدل از سیستم تحت مطالعه و این معیارهای کارایی استخراج می‌شوند که مایل به استخراجشان هستیم. بدین معنی که نوع ورودی‌هایی که به‌طور مفصل ذکرشده‌اند نه‌تنها از سیستم فیزیکی هستند بلکه از طریق استفاده موردنظر از سیستم نیز هستند (این طبیعت تجربی این مدل را در اختیار قرار می‌دهد). به‌عنوان‌مثال، در یک دستگاه خودپرداز، ما مایلیم تا کارایی ویژگی‌های سرویس سریع، همچون دادن پول یا تنظیم مبالغ را موردمطالعه قرار دهیم. ما ممکن است تصمیم بگیریم که از جزئیات عملیات داخلی ATM یا تغییرات کاربری صرف‌نظر نمایید. این مدل می‌تواند دستگاه خودپرداز، رابط آن، و یک مدل (تحلیلی، شبیه‌سازی) فرایند داخلی باشد. این آزمایش می‌تواند بر این مبنا باشد که از کاربران (آزمایش گیرندگان) خواسته شود که از این مدل همانند یک سیستم واقعی استفاده نمایند و کارایی آن را در موردسنجش قرار دهند. این معیارها می‌توانند به نیت طراحی بپردازند. بدین معنی که ما می‌توانیم رصد کنیم که چه نوع از ویژگی دسترسی به پول نقد نسبت به دیگری استفاده‌شده است، که در سطحی بالاتر انجام‌شده است، یا قابلیت‌های آن خیلی کاربردی نبودند.

تعریف معیارهای کارایی موردنیاز محرکه طراحی و/یا بازطراحی مدل هستند. درواقع، کل فرایند فرموله سازی و شکل‌دهی یک مدل از یک سیستم واقعی به شکلی تراکنشی رخ می‌دهد. ازآنجاکه بینش درباره سیستم واقعی از طریق مطالعه آن برای اهداف مدل‌سازی کسب می‌شود، رویکردهای جدید طراحی و مدل‌ها و اجزاء بهتری استخراج می‌شوند. این فرایند تکرار تا زمانی ادامه می‌یابد که مدل کننده به سطحی از جزئیات همخوان با دیدگاه سیستم واقعی موردنظر در فاز توسعه مدل،هدف دست‌یافته باشد. سطح جزئیات نشانگر اهمیت هر جزء از چشم مدل کننده به‌عنوان نکاتی است که باید مورد ارزیابی قرار بگیرند.

برای تأکید بیشتر می‌توان گفت که روش توسعه و استفاده از یک مدل یک سیستم از قرار ذیل است:

۱. مسئله‌ای که باید موردمطالعه قرار بگیرد و این معیارهای تحلیل را تعریف کنید.

۲. مدل سیستم را تعریف نموده و/یا پالایش کنید (شامل توسعه انتزاعی سیستم در روابط ریاضی، منطقی و رویه‌ای).

۳. داده‌ها را برای ورود به مدل گردآوری نمایید (آنچه برای "شبیه‌سازی آن دنیا" در خارج می‌تواند باشد و یا آنچه باید از مدل گرفته شود یا تغذیه شود را تعریف کنید).

۴. یک ابزار مدل‌سازی را انتخاب نموده و این مدل را برای پیاده‌سازی ابزار آماده‌سازی و تقویت نمایید.

۵. راستی آزمایی کنید که پیاده‌سازی ابزار انعکاس دقیقی از این مدل باشد.

۶. بررسی کنید که پیاده‌سازی ابزار دقت یا تناظر مطلوب را با سیستم واقعی که باید مدل‌سازی شود فراهم‌سازی می‌نماید.

۷. با مدل آزمایش انجام دهید تا معیارهای کارایی حاصل شوند.

۸. نتایج ابزار را مورد تجزیه‌وتحلیل قرار دهید.

۹. از این یافته‌ها برای استخراج طراحی‌ها و بهبودها برای سیستم واقعی استفاده کنید.

اگرچه برخی از این گام‌ها قبلاً تعریف‌شده بودند، ولی آن‌ها را می‌توان در زمینه این روش موردبررسی مجدد قرارداد.

اولین کار در این روش تعیین این است که حوزه عمل این مسئله چیست و آیا این سیستم واقعی برای مدل‌سازی قابل تطابق دهی هست یا نه. این وظیفه از مسئله‌ای تشکیل یافته است که به‌روشنی مسئله را تعریف می‌کند و صراحتاً اهداف تحقیق را بیان می‌کند. این وظیفه ممکن است در طی کل فاز ساخت مدل، به خاطر طبیعت مدل‌سازی، نیاز به ارزیابی مجدد داشته باشد. بدین معنی که با ورود بینش‌های بیشتر به این فرایند، یک مدل بهتر، البته متفاوت، را می‌توان توسعه داد. این دربرگیرنده یک بازتعریف پرسش‌ها و تکامل یک تعریف مسئله جدید است.

وقتی یک تعریف مسئله مشخص گردید، کار تعریف و بازتعریف مدلی از این فضای مسئله دنیای واقعی می‌تواند در ادامه بیاید. این مدل نوعاً از بخش‌های متعددی تشکیل یافته است که هم استاتیک و

هم‌دینامیک هستند. آن‌ها عناصر سیستم (استاتیک)، مشخصه‌هایشان، و شیوه‌هایی را تعریف می‌کنند که در آن‌ها این عناصر به‌مرورزمان تراکنش می‌نمایند تا وضعیت سیستم واقعی به‌مرورزمان تعیین‌شده یا بازتاب داده شود. همان‌گونه که قبلاً ذکر شد، این فرایند تنظیم یک مدل تا حدود زیادی به دانش، درک و تجربه مدل کننده (هنر در مقایسه با علم) بستگی دارد. مدل کننده شالوده سیستم واقعی را بدون پوشاندن جزئیات اضافی استخراج می‌کند. این مفهوم دربرگیرنده ثبت (مهم‌ترین) جنبه‌های سیستم بدون رفع پیچیدگی ولی با مقدار کافی برای بازتاب دادن واقع‌گرایانه جنبه‌های وابسته سیستم واقعی است. مقدار جزئیاتی که باید در یک مدل گنجانده شود عمدتاً مبتنی بر هدفش است. به‌عنوان‌مثال، اگر ما مایل به مطالعه انواع و نسبت تراکنش کاربری بر روی یک ماشین خودپرداز باشیم، تنها کافی است که این ماشین را به‌عنوان یک مصرف‌کننده تمام دفعات تراکنش و انواعشان مدل‌سازی نماییم. لزومی ندارد که ما ماشین و تراکنش‌های آن با یک پایگاه داده والد را با هیچ جزئیاتی تنها از یک سطح کاربری خروجی ناخالص مدل‌سازی کنیم.

فرایند توسعه مدل از صورت‌مسئله فعال و زمان‌بر است. بااین‌حال، یکی از عواقب این فاز تعریف الزامات داده‌های ورودی است. نوعاً برای گردآوری مقادیر داده‌های تعریف‌شده به‌منظور استخراج مدل به کار اضافی نیاز خواهد بود. در بسیاری از موارد در توسعه مدل، برای ورود داده‌ها باید فرضیه ارائه شود یا مبتنی بر تحلیل مقدماتی باشند، یا این داده‌ها ممکن است نیازی به مقادیر دقیقی برای مدل‌سازی خوب نداشته باشند. حساسیت این مدل به یک‌شکل قابل‌اجرا یا تحلیلی تبدیل می‌شود و داده‌ها را می‌توان درباره اثراتشان مورد تجزیه‌وتحلیل قرارداد.

وقتی طرح‌ریزی و توسعه یک مدل ورودی داده‌ها انجام شد، کار بعدی تبدیل نمودن آن به یک‌شکل تحلیلی یا قابل‌اجرا است. ابزار مدل‌سازی منتخب محرکه بخش عمده مابقی این کار است. ابزارهای در دسترس شامل شبیه‌سازی، مدل‌سازی تحلیلی، بسترهای آزمودن، و تحلیل عملیاتی می‌شوند. هر یک از این ابزارهای مدل‌سازی نقاط قوت و ضعف خودشان رادارند. شبیه‌سازی امکان طیف وسیعی از بررسی‌ها را بر مبنای تجربه مدل کننده فراهم می‌آورد، آنالیز تحلیلی ابزاری برای آزمودن مدل بر روی قطعات سخت‌افزاری واقعی یک سیستم را فراهم می‌آورد. تحلیل عملیاتی الزام می‌کند که ما سیستم واقعی را در دسترس داشته باشیم و اینکه ما بتوانیم آن را برای اجرای مطالعه موردنظر به کار بگیریم. این

همیشه یک گزینه در دسترس در سیستم‌های پیچیده نیست. درهرحال، ابزار منتخب تعیین خواهد نمود که مدل کننده چگونه مدل، ورودی‌هایش، و بازدهی تجربی‌اش را توسعه می‌دهد.

وقتی یک مدل و یک ابزار مدل‌سازی برای پیاده‌سازی آن توسعه داده شد، مدل کننده مدل قابل‌اجرا را توسعه می‌دهد. این مدل، به‌محض توسعه داده شدن، باید راستی آزمایی شود تا تعیین شود که آیا دقیقاً سیستم واقعی موردنظر تحت مطالعه را بازتاب می‌دهد یا نه. راستی آزمایی نوعاً با بررسی دستی این انجام می‌شود که نتایج محاسباتی مدل مطابق با نتایج پیاده‌سازی باشند. بدین معنی که آیا مدل اختصاری و مدل پیاده‌سازی شده کار یکسانی را انجام داده و نتایج منطبقی ارائه می‌کنند؟

اعتبارسنجی مشابه با راستی آزمایی است. اعتبارسنجی با تعیین این سروکار دارد که آیا پیاده‌سازی مدل نمایش دقیقی از سیستم واقعی که باید مدل‌سازی شود را ارائه می‌کند یا نه. بررسی دقت معمولاً از یک مقایسه مدل و ساختارهای سیستم در قبال یکدیگر و یک مقایسه ورودی‌ها، خروجی‌ها و فرایندهای ابزار مدل در قبال سیستم واقعی برای برخی از مرزهای مشخص تشکیل یافته است. اگر آن‌ها منطبق با برخی از معیارهای تغییرات مدل‌سازی یا آزمایشی باشند، ما مدل را تخمینی دقیق از سیستم فرض می‌کنیم. در غیر این صورت، نقطه‌ضعف‌ها باید مشخص‌شده، اصلاح‌شده و مدل تا زمان حصول همزمانی مورد اعتبارسنجی مجدد قرار بگیرد.

وقتی پیاده‌سازی مدل راستی آزمایی شده و اعتبار سازی شد، مدل کننده‌ها می‌توانند آزمایش‌های موردنظر پروژه را انجام دهند. این فاز مرحله‌ای است که در آن محدودیت‌های اصلی مدل را می‌توان بسط داد و بینش‌های جدیدی درباره پیچیدگی‌های سیستم واقعی کسب نمود. محدودیت‌های آزمایش مستقیم به ابزار منتخب ارتباط می‌یابند: شبیه‌سازی در انعطاف‌پذیرترین حالت است، و پس‌ازآن بسترهای آزمودن، آنالیز تحلیلی، و تحلیل عملیاتی قرار دارند.

وقتی تجربه‌اندوزی کامل شد، یک تحلیل پیوسته از نتایج به شکلی فعالانه به اجرا درمی‌آید. این فاز با گردآوری و آنالیز داده‌های ارائه‌شده تجربی برای کسب بینش بیشتر درباره سیستم تحت مطالعه سروکار دارد. بر مبنای نتایج کسب‌شده، مدل کننده این نتایج را برای سیستم واقعی به فرایند تصمیم‌گیری تزریق می‌کند، و به‌طور بالقوه ساختار و عملیات آن را بر مبنای یافته‌های مدل تغییر می‌دهد. یک مطالعه وقتی موفق فرض می‌شود که تلاش برای مدل‌سازی داده‌های مفیدتری را برای استخراج محصول نهایی ارائه نماید. این خروجی‌ها می‌توانند یک مفهوم را درباره سیستم تثبیت نمایند، یک کاستی را تعریف نمایند،

بینشی را درباره اصلاحات ارائه نماید، یا اطلاعات دیگر درباره سیستم را تقویت نماید. مدل‌سازی ابزاری مفید است که محیط‌های پیچیده با آن مورد تجزیه‌وتحلیل قرار می‌گیرند.

۱-۷-۳ ابزارهای مدل‌سازی

همان‌گونه که مختصراً در بخش قبلی ذکر گردید، کلاس‌های عمده‌ای از ابزارهای مدل‌سازی امروز مورداستفاده قرار می‌گیرند: تحلیلی، شبیه‌سازی، بستر تست، و تحلیل عملیاتی. هر یک مزایای خودشان را در مجموعه ابزارهای مدل کننده دارند و به دلایل مختلفی استفاده می‌شوند، که در بخش‌های بعدی این کتاب موردبحث قرار خواهند گرفت.

ابزارهای مدل‌سازی تحلیلی

مدت‌هاست که ابزارهای مدل‌سازی تحلیلی به‌عنوان یک تکنیک پیاده‌سازی مورداستفاده بوده‌اند، که دلیل اصلی کارکردشان است. پیاده‌سازی‌های تحلیلی مدل‌ها بر قابلیت مدل کننده برای توصیف یک مدل از جنبه‌های ریاضی متکی هستند. نوعاً، اگر یک سیستم را بتوان به‌عنوان مجموعه‌ای از صف‌ها، با سرویس، انتظار و زمان‌های تحلیلی تعریف‌شده به‌صورت تحلیلی با سرویس لحاظ نمود، تحلیل صف‌بندی را می‌توان برای حل مسئله مورداستفاده قرارداد. دیگر ابزارهای تحلیلی همچون شبکه‌های پتری را نیز می‌توان برای حل چنان مسائلی اعمال نمود.

برخی از دلایل علت انتخاب مدل‌های تحلیلی به‌عنوان یک ابزار مدل‌سازی از قرار ذیل است:

۱. مدل‌های تحلیلی ویژگی‌های برجسته‌تری از سیستم‌ها را در برمی‌گیرند، بدین معنی که بیشتر سیستم‌ها را می‌توان به‌عنوان تأخیرهای صف‌بندی، زمان‌های سرویس، زمان‌های ورود، و امثالهم نمایش داد، و، درنتیجه، ما با کنار گذاشتن جزئیات می‌توانیم از این دیدگاه مدل‌سازی را انجام دهیم.

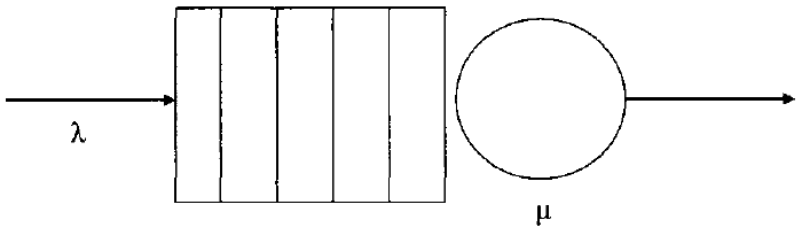
۲. مفروضات یا تحلیل واقع‌گرایانه است.

۳. الگوریتم‌ها برای حل معادلات صف‌بندی در شکل ماشینی در دسترس هستند تا تحلیل تسریع شود.

مفهوم ضمنی قابل‌برداشت از آن این است که مدل‌های صف‌بندی یک ابزار آسان و دقیق سیستم‌های مبتنی بر صف را در اختیار قرار می‌دهند. صف‌ها خطوط انتظار هستند، و نظریه صف‌بندی مطالعه محرکه‌های خط انتظار است.

در تحلیل صف‌بندی در ساده‌ترین سطح (یک صف)، یک صف (خط انتظار) وجود دارد که با مشتریان جدید تغذیه می‌شود (نرخ ورود)، این صف توسط یک سرور عملیاتی می‌شود، که مشتریان را بر طبق یک نرخ سرویس از صف استخراج می‌کند (شکل ۱-۱۳ را ببینید).

این صف به‌صورت ذیل عمل می‌کند: یک ورود در صف قرار می‌گیرد، و، در صورت مشغول بودن سرور، مشتری در یک ساختار انتظار (صف) قرار می‌گیرد، مگر آنکه صف پر باشد، که در این صورت مشتری رد (ریجکت) می‌شود (جایی برای انتظار وجود ندارد). از سوی دیگر، اگر صف خالی باشد، مشتری به مکان سرویس برده می‌شود و زمان نرخ سرویس در آن با تأخیر مواجه می‌شود. مشتری سپس از صف خارج می‌شود.



شکل (۱-۱۳): صف سرور واحد

به‌منظور تجزیه‌وتحلیل این پدیده، ما باید ابزارهای (تئوری‌های) تصوری و تحلیلی را داشته باشیم که با آن‌ها این تصور را دست‌کاری نماییم. علاوه بر این، به‌منظور تعیین مفید بودن این تکنیک، لازم است که بدانیم چه چیزی را می‌توان مورد تجزیه‌وتحلیل قرارداد و چه نوع سنجشی از این صف استخراج می‌شود.

مفهوم مورداستفاده (شکل ۱-۱۳را ببینید) برای توصیف پدیده صف از قرار ذیل است: توزیع ورودی الگوهای ورودی مشتریان در صف را تعریف می‌کند. این‌ها توسط یک متغیر تصادفی تعریف می‌شوند

که زمان بین ورود را تعریف می‌کند. یک معیار پرکاربرد فرایند ورود پواسون است که به این صورت تعریف می‌شود:

$$P\ (arrival\ time) = ۱ - e^{-\lambda} \qquad (۱-۱)$$

که در آن نرخ ورود متوسط λ است. این صف به‌عنوان یک مخزن ذخیره‌سازی برای مشتریان تعریف می‌شود. علاوه بر این، خط‌مشی که برای پذیرش و حذف مشتریان استفاده می‌کند نیز تعریف می‌شود. نمونه‌هایی از ساختارهای صف‌بندی پرکاربرد عبارت‌اند از خروج به ترتیب ورود[۲۱] (FIFO) و خروج به ترتیب ورود[۲۲] (LIFO). آخرین جزء اصلی توصیف صف خط‌مشی سرویس است،که روشی است که از طریق آن مشتریان برای سرویس پذیرفته می‌شوند و طول سرویس است. این زمان سرویس با یک توزیع، یک متغیر تصادفی، توصیف می‌شود. یک توزیع زمانی سرویس نوعی سرویس تصادفی است که از این رابطه به دست می‌آید:

$$W_s(t) = ۱ - e^{-\mu t} \qquad (۲-۱)$$

که در آن $t \gg ۰$ است، و نماد μ برای توصیف این توزیع مشترک برای نرخ سرویس متوسطش رزرو می‌شود. توزیع‌های مورداستفاده برای توصیف نرخ ورود و نسبت‌های سرویس متعدد و متغیر هستند، به‌عنوان‌مثال، نمایی، به‌طورکلی، ارلانگ، قطعی، فوق نمایی را می‌توان استفاده نمود. نماد کندال برای توصیف این توسعه داده‌شده بود که چه نوع از صف موردبررسی قرار می‌گیرد. شکل این نماد از قرار ذیل است:

$$A/B\ /\ c/K\ /\ m/Z \qquad (۳-۱)$$

[۲۱] first-in first-out
[۲۲] last-in first-out

که در آن A توزیع زمانی بین ورودی را مشخص می‌کند، B توزیع زمان سرویس است، c تعداد سرورها است، K ظرفیت سیستم است، m عدد منبع است، و Z ساختار صف است.

این نوع از تحلیل را می‌توان برای تولید آمارها در زمان انتظار متوسط، طول متوسط صف، زمان سرویس متوسط، شدت ترافیک، استفاده از سرور، زمان متوسط در سیستم، و احتمال‌های مختلف زمان‌های انتظار و سرویس مورد انتظار و زمان‌های انتظار استفاده نمود. جزئیات بیشتر درباره این تکنیک تحلیل و مدل‌سازی در فصل ۷ ارائه خواهد شد.

ابزارهای مدل‌سازی شبیه‌سازی

شبیه‌سازی به‌عنوان یک ابزار مدل کننده مدت‌هاست که برای مدل‌سازی و تحلیل بسیاری از سیستم‌ها مورداستفاده قرارگرفته است، به‌عنوان‌مثال، کسب‌وکار، اقتصاد، بازاریابی، آموزش، سیاست، علوم اجتماعی، علوم رفتاری، روابط بین‌الملل، حمل‌ونقل، اعمال قانون، مطالعات شهری، سیستم‌های جهانی، کامپیوترها، کارخانه‌ها و خیلی موارد دیگر. شبیه‌سازی به خاطر انعطاف‌پذیری‌اش برای مسائل مختلفی کاربرد می‌یابد. این‌یک ابزار دینامیک است که قابلیتی برای تعریف مدل‌های سیستم‌ها در اختیار مدل کننده قرار می‌دهد و آن‌ها را در معرض اقدام قرار می‌دهد. این آزمایشگاهی را در اختیار قرار می‌دهد که در آن مسائل متعددی در رابطه با یک سیستم بدون اختلال در سیستم واقعی موردمطالعه قرار می‌گیرد. طیف وسیعی از آزمایش‌ها را می‌توان در یک محیط بسیار کنترل‌شده به اجرا درآورد: زمان را می‌توان فشرده‌سازی نمود، تا بدین‌صورت امکان مطالعه پدیده‌هایی که به شکلی دیگر غیرقابل مشاهده هستند فراهم بیاید، و تحلیل حساسیت را می‌توان بر روی تمام اجزاء انجام داد.

بااین‌حال، مدل‌سازی شبیه‌سازی ممکن است نقایص خودش را داشته باشد. توسعه مدل ممکن است پرهزینه باشد و برای اجرا نیاز به زمان زیادی داشته باشد، فرض لحاظ شده ممکن است بحرانی شده و موجب یک سوگیری در مدل یا حتی سبب خروج آن از محدوده‌های واقعیت شود، و، نهایت اینکه، استفاده از این مدل ممکن است، درصورتی‌که امکان رشد نامحدود را بیابد، بیش‌ازحد طاقت‌فرسا شود و به شکلی مؤثر مقداردهی اولیه را انجام دهد. به‌منظور جلوگیری از بسیاری از این اثرات نامناسب، مدل

کننده باید خط‌مشی‌های دقیق فرمولاسیون، ساخت و استفاده را دنبال نماید. این‌ها اثرات بستر را حداقل سازی خواهند نمود و درعین‌حال مزایای شبیه‌سازی را حداکثر سازی می‌کنند.

شکل‌های شبیه‌سازی بسیاری بر مبنای سیستم در دست مطالعه وجود دارند. اساساً، چهار کلاس از مدل‌های شبیه‌سازی وجود دارند: پیوسته، گسسته، و ترکیبی. این چهار تکنیک قوت ضروری روش‌ها را برای مدل‌سازی بیشتر سیستم‌های موردنظر فراهم می‌آورند. یک مدل پیوسته مدلی است که حالت پردازشی آن بر مبنای سیگنال‌ها یا متغیرهای متغیر بازمان تغییر می‌یابد. شبیه‌سازی گسسته در عوض متکی بر شرایط رویدادی و انتقال‌های رویدادی برای تغییر وضعیت است. شبیه‌سازی‌های مبتنی بر صف ابزارهای دینامیکی را برای ساخت و تحلیل سیستم‌های مبتنی بر صف فراهم می‌آورند. آن‌ها به‌صورت دینامیک رخدادهای ریاضی آنالیز شده در تکنیک‌های تحلیلی را مدل‌سازی می‌کنند.

مدل‌های شبیه‌سازی در تحلیل یک سیستم مبتنی بر تطابق سیستم در شبیه‌سازی ساخته‌شده و پیگیری می‌شوند. بدین معنی که، پیش از آنکه ما یک‌نهاد دنیای واقعی را شبیه‌سازی کنیم، باید تعیین کنیم که مسئله نیاز و تطابق با شبیه‌سازی دارد یا نه. فاکتورهای مهمی که باید در نظر گرفته شوند عبارت‌اند از هزینه، امکان‌پذیری اجرای آزمایش‌های مفید، و امکان شکل‌های ریاضی و دیگر شکل‌های آنالیز. یک شبیه‌سازی گزینه‌ای مناسب برای پیاده‌سازی مدل لحاظ می‌شود، یک مدل رسمی با شکل ابزارهای شبیه‌سازی در دسترس باید به اجرا دربیاید. پس از تکمیل مشخصه یک مدل، برنامه کامپیوتری که این مدل را به شکل قابل‌اجرا تبدیل می‌کند باید توسعه داده شود. نهایتاً ، وقتی‌که مدل کامپیوتری راستی آزمایی و اعتبارسنجی شد، مدل کننده می‌تواند با شبیه‌سازی تجربه کند تا به مطالعه سیستم واقعی کمک شود.

بسیاری از زبان‌ها برای استفاده در ویرایش درحال‌توسعه و قابل‌اجرای کامپیوتری یک مدل در دسترس هستند – به‌عنوان‌مثال، GPSS، Q،gert، Simscript، Slam، AWSIM و Network ۲,۵. انتخاب زبان شبیه‌سازی مبتنی بر نیازها و ترجیحات کاربران است، زیرا هر یک از این‌ها یک ابزار مدل‌سازی مفید را برای پیاده‌سازی یک شبیه‌سازی فراهم خواهند آورد. جزئیات این‌ها و مزایای جنبه-های دیگر شبیه‌سازی در فصل ۸ موردبررسی قرار می‌گیرند.

بسترهای تست به‌عنوان ابزارهای مدل‌سازی

بسترهای تست، به صورتی که قبلاً ذکر شد، جزییات ترکیبی از سیستم‌ها هستند و برای مطالعه اجزاء و تراکنش‌ها برای دستیابی به بینش بیشتر درباره شالوده سیستم واقعی استفاده می‌شوند. آن‌ها از نمونه‌های اولیه و اجزاء سیستم واقعی ساخته‌شده‌اند و برای ارائه بینش درباره کارکردهای یک عنصر (یا عناصر) یک سیستم استفاده می‌شوند. ویژگی مهم یک بستر تست این است که تنها بر زیرمجموعه‌ای از سیستم کلی متمرکز است. بدین معنی که جنبه مهمی که ما مایل به مطالعه، پالایش، یا توسعه آن هستیم جنبه پیاده‌سازی شده در بستر تست است. تمام جنبه‌های دیگر ریشه‌هایی دارند که محرکه‌شان را فراهم می‌آورند یا بارشان را استخراج می‌کنند ولی خودشان اجزاء کاملی نیستند، بلکه صرفاً اجزاء شبیه‌سازی‌شده‌ای هستند. بستر تست یک محیط واقع‌گرایانه سخت‌افزاری،نرم‌افزاری فراهم می‌آورد که با آن اجزاء تست می‌شوند بدون اینکه به سیستم نهایی نیازی باشد. بستر تست ابزاری برای بهبود درک از الزامات کارکردی و رفتار عملیاتی سیستم فراهم می‌آورد. این بستر امکان سنجشی را فراهم می‌آورد که از آن نتایج کمی درباره سیستم را می‌توان استخراج نمود. در آن محیطی یکپارچه فراهم‌سازی شده است که در آن روابط راهکارها برای مسائل سیستم را می‌توان مورد ارزیابی قرارداد. نهایتاً اینکه، محیطی را فراهم می‌آورد که در آن تصمیمات درباره طراحی ممکن است مبتنی بر هر دو مطالعه تئوریک و تجربی باشند.

در اینجا نیز آنچه از این بحث پیداست اینکه بستر تست، همانند ابزارهای شبیه‌سازی و تحلیلی، یک محیط آزمایشگاهی را فراهم می‌آورد که در آن اجزاء سیستم واقعی مدل‌سازی شده را می‌توان تجربه نمود، مطالعه کرد، و از بسیاری از زوایا ارزیابی کرد. بااین‌حال، بسترهای تست محدودیت‌های خودشان رادارند ازاین‌جهت که هزینه‌شان برای توسعه بیشتر است و ازنظر کاربرد به فقط سیستم‌ها و اجزاء مدل‌سازی مناسب برای چنان محیط‌هایی محدود هستند. به‌عنوان‌مثال، ما احتمالاً نمی‌توانستیم یک سیستم محاسباتی توزیع‌شده پیچیده را در یک بستر تست مدل‌سازی کنیم. در عوض مدل‌های تحلیلی یا شبیه‌سازی را به‌عنوان یک پاس اولیه در نظر گرفتیم و از یک بستر تست بین مفهوم اولیه و طراحی نهایی استفاده نمودیم. این می‌تواند در ادامه بحثمان در اینجا و در فصل ۸ مشهودتر شود، که در آن بسترهای تست با جزئیات به‌مراتب بیشتری موردبحث قرار می‌گیرند.

یک بستر تست از سه جزء تشکیل یافته است: یک زیرسیستم تجربی، یک زیرسیستم مانیتورینگ، و یک زیرسیستم شبیه‌سازی،تحریک. زیرسیستم آزمایشی مجموعه اجزاء و یا نمونه‌های اولیه سیستم واقعی است که ما مایل به مدل‌سازی و آزمودنشان هستیم. زیرسیستم مانیتورینگ از رابط‌هایی به سیستم تجربی برای استخراج داده‌های خام و یک جزء پشتیبانی به‌منظور تطبیق و آنالیز اطلاعات گردآوری‌شده تشکیل یافته است. زیرسیستم شبیه‌سازی،تحریک ابزارهای لازم برای ارائه ورودی‌ها و خروجی‌های سیستم واقعی به‌منظور تأمین یک محیط آزمایش واقع‌گرایانه را فراهم می‌آورد.

با این عناصر، یک بستر تست می‌تواند یک ابزار انعطاف‌پذیر و مدولار را فراهم بیاورد که با آن طیف وسیعی از محرک‌ها، پیکربندی‌ها و کاربردهای مختلف سیستم مورد آزمایش قرار می‌گیرند. رویکرد بستر تست روشی برای بررسی جنبه‌های سیستم فراهم می‌آورد که مکمل روش‌های شبیه‌سازی و تحلیلی هستند.

تصمیمات درباره استفاده از یک بستر تست نسبت به روش‌های دیگر عمدتاً نشأت‌گرفته از هزینه مرتبط با توسعه و مزایای واقعی هستند که می‌توانند با چنان پیاده‌سازی‌هایی محقق شوند. علاوه بر این، نتایج بستر تست تنها به‌خوبی قابلیت مانیتور برای استخراج و تحلیل پدیده‌های واقعی رخ‌داده و قابلیت اجزاء شبیه‌سازی،تحریک برای بازتاب دادن یک رابط واقع‌گرایانه با محیط هستند.

بسترهای تست زمینه شبکه‌های محلی برای تحلیل طیف وسیعی از اجزاء می‌توانند استفاده شوند و استفاده‌شده‌اند. محدودیت انعطاف‌پذیری در تجزیه‌وتحلیل ساختارها و پیاده‌سازی‌های بسیار واریانس هزینه مرتبط با ساخت یک بستر تست بوده و خواهد بود. در زمینه یک سیستم کامپیوتری، بستر تست باید نسبت بزرگی از سخت‌افزار کامپیوتری سیستم کامپیوتری، سخت‌افزار ذخیره‌سازی داده‌ها، سخت-افزار انتقال داده‌ها، و احتمالاً سخت‌افزار و نرم‌افزار شبکه را پیاده‌سازی نماید که مفید هستند. بااین‌حال، با این کار، مدل کننده محدود به مطالعه این پیکربندی واحد می‌شود. در بخش‌های بعدی مشاهده خواهد شد که این محدودیت‌ها و مزایای مدل‌سازی چه هستند و چگونه می‌توانند بر رویکرد ما برای مطالعه یک سیستم تأثیر بگذارند.

تحلیل عملیاتی به‌عنوان یک ابزار مدل‌سازی

ابزار نهایی از چشم‌انداز یک مدل کننده تحلیل عملیاتی است، که گاهی با عنوان تحلیل تجربی نیز مورد ارجاع قرار می‌گیرد. در این تکنیک، مدل کننده چندان به خلاصه‌ای از سیستم توجه ندارد، بلکه به شیوه استخراج از اطلاعات سیستم واقعی توجه دارد که بر روی آن همان تحلیل راهکارهای بالقوه‌ای را توسعه دهد که در اختیار مدل‌های دیگر قرار می‌گیرند.

تحلیل عملیاتی به استخراج اطلاعات از یک سیستم‌عامل مربوط می‌شود که برای توسعه طرح‌ها درباره عملیات آتی سیستم توجه دارد. افزون بر این، این روش مدل‌سازی را می‌توان توسط دو تکنیک مدل‌سازی دیگر برای استخراج اطلاعات معناداری استفاده نمود که می‌توانند به فرایندهای تحلیلشان تزریق شوند یا عملیات تحلیلشان را راستی آزمایی یا اعتبارسنجی نمایند.

تحلیل عملیاتی با سنجش و ارزیابی یک سیستم واقعی در عملیات سروکار دارد. سنجش به سنجش دقیق سیستم برای استخراج اطلاعات توجه دارد. ابزارهای اجرای این از مانیتورهای سخت‌افزاری و/یا نرم-افزاری استفاده می‌کنند.

مانیتورهای سخت‌افزاری از مجموعه‌ای از پروب ها یا سنسورها، یک دستگاه حسگری منطقی، مجموعه‌ای از شمارنده‌ها، و یک نمایشگر یا واحد ضبط تشکیل یافته‌اند. این پروب ها وضعیت نقاط سیستم منتخب را مانیتور می‌کنند. نوعاً، پروب ها را می‌توان به شکلی برنامه‌ریزی نمود که در یک رخداد خاص وارد عمل شوند، و بدین طریق امکان پیگیری رخدادهای خاص درون یک سیستم فراهم بیاید.

زیرسیستم حسگری منطقی برای تفسیر داده‌های ورودی خامی که باید در آیتم‌های اطلاعاتی معنادار پروب شوند استفاده می‌شود. این شمارنده‌ها برای تنظیم نرخ‌های نمونه‌برداری بر روی فعالیت‌های دیگری استفاده می‌شوند که نیاز به بازه‌های زمان‌بندی‌شده دارند. آخرین جزء اطلاعات را به‌صورت حسگری و کاهش داده می‌شوند ثبت نموده و نمایش می‌دهد. کمک بیشتری را می‌توان برای آنالیز بیشتر اطلاعات افزود. قابلیت اجرای تحلیل عملیاتی مؤثر مستقیماً وابسته به قابلیت مانیتور سخت‌افزار و نرم‌افزار برای استخراج اطلاعات است. مانیتور سخت‌افزاری تنها به‌اندازه قابلیتش برای اتصال به سیستم بدون ایجاد اعوجاج بی‌مورد مؤثر است.

مسئله این است که مانیتور مبتنی بر سخت‌افزار نمی‌تواند، در یک سیستم کامپیوتری، رخدادهای مرتبط با نرم‌افزار را لمس کند. تراکنش نرم‌افزار و سخت‌افزار سیستم باهم داده‌های به‌مراتب مؤثرتری را برای

تحلیل عملیاتی که باید به اجرا دربیاید فراهم خواهد آورد. مانیتورهای نرم‌افزاری نوعاً سبک‌های نمونه-برداری یا ردیابی رخدادها را فراهم می‌آورند. مانیتورهای ردیابی رخدادها از مجموعه‌ای از روال‌های سیستم تشکیل یافته‌اند که بر روی رخدادهای نرم‌افزاری خاص، همچون وقفه‌های CPU، فازهای زمان‌بندی، توزیع، جستارها، دسترسی ۱۱۰، و امثالهم فراخوانی می‌شود. مانیتور نرم‌افزاری بر روی این رویدادها به اجرا درمی‌آید و اطلاعات مرتبط را بر روی وضعیت سیستم ثبت می‌کند. این اطلاعات می‌توانند دربرگیرنده رویدادی که در آن زمان برانگیخته می‌شود، اینکه چه فرایندی پیش از رویداد بر CPU کنترل داشته است، وضعیت CPU (ثبات‌ها، شرایط و غیره) باشد. این داده‌ها می‌توانند بینش‌های بسیاری را در این مورد فراهم بیاورند که چه برنامه‌هایی دارای بیشترین دسترسی به CPU هستند، چقدر زمان در بالاسری سرویس سیستم صرف می‌شود، طول صف دستگاه‌ها، و بسیاری از رویدادهای قابل‌توجه.

ترکیب مانیتورهای سخت‌افزار و نرم‌افزار مجموعه وسیعی از داده‌ها را در اختیار تحلیلگر قرار می‌دهد که آنالیز را بر روی آن‌ها انجام دهد. محاسبات معمول با ابزارهای مختلف محاسباتی و انواع استفاده‌های دستگاه‌ها و نرم‌افزار و ترسیم تناوب‌های نسبی دسترسی و استفاده سروکار دارند.

سنجش‌ها و محاسبات انجام‌شده در این سطح فقط عملکرد سیستم حاضر را مدل‌سازی می‌کنند. تحلیلگر عملیاتی باید از این معیارها برای بسط عملکرد استفاده نموده و مرزهای جدیدی را بر مبنای بسط داده‌ها به بخش‌های نامعلوم فرض نماید و محاسبات را بر مبنای داده‌های مطرح‌شده به اجرا دربیاورد. با استفاده از این تکنیک‌ها، تحلیلگر می‌تواند تغییرات و اصلاحات را پیشنهاد دهد و تأثیرشان را بر مبنای اطلاعات واقعی پیش‌بینی نماید.

۱-۸ معیارهای کارایی و معیارهای ارزیابی

انتخاب یک معماری سیستم کامپیوتری و نرم‌افزار پشتیبانی سیستم نیازمند معیارهای کارایی و ارزیابی است. به‌منظور تولید چنان اطلاعاتی، یک کاربر باید از متدلوژی انتخابی پیروی نماید که نیازها، انگیزه-ها، و مرزهای محیطی و فنّاورانه کاربر را تعریف می‌کند. در رابطه با خرید هر محصول، خریدار باید مشخص نماید که چگونه محصول (در این مورد، یک سیستم کامپیوتری) مورداستفاده قرار خواهد گرفت. این عنصر اول فرایند انتخاب از همه مهم‌تر است، زیرا اگر ما نیازها و کاربردها را به‌درستی

تعریف نکنیم، وظایف باقیمانده یک خطای از پیش تعریف‌شده داخلی خواهند داشت. ازاین‌رو، خریدار بالقوه باید یک لیست موردنظر از تمام استفاده‌های بالقوه را داشته باشد. به‌عنوان‌مثال، این لیست می‌تواند دربرگیرنده موارد ذیل باشد:

- پردازنده‌های متعدد
- سرور فایل توزیع‌شده
- دیسک درایوهای اضافی
- واژه‌پردازی
- تحلیل برگه گسترده
- پست الکترونیکی
- ورود شغل از راه دور
- کنترل بی‌درنگ
- لاگین و اجرا یا نتایج تراکنشی
- چیدمان نصب فیزیکی
- شمارش و انواع گره ماکسیمم
- ملاحظات پایایی
- مدیریت شبکه

- اتوماسیون کارخانه
- انواع کامپیوتر
- ویدئو، صوت یا هر دو
- ارتباط با MANMAN ها یا W WANهای موجود
- اشتراک‌گذاری منابع
- محاسبه توزیع‌شده
- پایگاه داده بسیار عظیم

از این لیست موردنظر، کاربر باید الزامات پردازشی، انتقال ارتباطات و الزامات مدیریتی را ارائه نماید. به‌عنوان‌مثال، با توجه به اینکه ما N کامپیوتری داریم، که باید بتواند به‌طور همزمان داده‌ها را به سایت‌های دیگر انتقال دهد، الزامی برای پهنای باند (یا یک ماکسیمم نرخ ۱۱۰) و همزمانی دسترسی را دریافت نموده‌ایم، که هردوی آن‌ها، ازجمله، بر پروتکل‌ها، توپولوژی، و الزامات رساندن آن‌ها تأثیرگذار هستند. این مجموعه از الزامات پردازشی، انتقال ارتباطات، و الزامات مدیریتی را اکنون می‌توان استفاده نمود تا به ما در فازهای دیگر کمک نماید. بخش دوم این متدلوژی توسعه یک هدف انگیزشی برای سیستم کامپیوتری است: تعریف اینکه اصلاً چرا ما به آن نیاز داریم. به‌عنوان‌مثال، ممکن است بخواهیم که با رقبایی رقابت کنیم که با استفاده از یک صفحه پشتی ارتقاءیافته[۲۳]، برخورداری از مزیتی در دسترسی‌پذیری اطلاعاتی برای ارتقاء قابلیت تصمیم‌گیری شرکتی، یا برای فراهم‌سازی کنترل بهتر یا استفاده از منابع محاسباتی شرکت، سرویس بهتر یا بسیط‌تری را به مشتریانشان ارائه می‌کنند. انگیزه انتخاب سیستم محاسباتی معیارهای کارایی و ارزیابی بیشتری در اختیار خریدار یا طراح ما قرار خواهد داد که تصمیمان را مبتنی بر آن نماییم.

[۲۳] enhanced backplane

مرحله بعدی درون متدلوژی ارزیابی سیستم‌های کامپیوتری ارزیابی جنبه‌های محیطی و تکنولوژی است که سیستم کامپیوتری باید در آن‌ها جای بگیرد. به‌عنوان‌مثال، آیا سیستم کامپیوتری و زیرسیستم پیوستگی آن برای پیاده‌سازی در محیط‌های کثیف، گرم، سرد یا متغیر موردنظر است؟ آیا سیستم کامپیوتری یا برخی از اجزاء آن در معرض تنش یا فشار از سوی عناصر طبیعی همچون باد، باران، برف یا آذرخش قرار خواهند داشت؟ آیا سیستم کامپیوتری یا اجزاء آن در یک اتاق کامپیوتر که تهویه هوایی می‌شود قرار داده خواهند شد یا در نقاط مختلف ساختمان پراکنده خواهند بود؟ آیا این ساختمان ساختار جدیدی خواهد داشت و یا قدیمی خواهد بود؟ آیا لازم خواهد بود که ارتباطات سیستم‌های کامپیوتری به طبقات برسند و به بالا بروند؟ در این صورت، کد آتش غالب چیست؟ آیا سیستم کامپیوتری ساختمان‌های بسیاری را به یکدیگر پیوند خواهد داد؟ در این صورت، آیا پیوستگی‌ها به‌صورت بالاسری شکل خواهند گرفت یا از ساختمانی به ساختمان دیگر کشیده می‌شوند؟ آیا سیم‌کشی مخفی خواهد بود؟ آیا زیرآب خواهد بود یا درون لوله‌های حامل آب؟

از یک نظرگاه فنّاورانه، سیستم کامپیوتری ممکن است نیاز به مجموعه واریانسی از دارایی‌های شرکت حاضر خواهد داشت و قادر به طرح‌ریزی منابع جدید طرح‌ریزی‌شده خواهد بود. این منابع ویژگی‌های خودشان را از جنبه مشخصه‌های الکتریکی، شمارش پین، و ساختار خواهند داشت. این ویژگی‌ها در الزامات روی تجهیزات و نرم‌افزارهای رابط نیز نقشه‌برداری خواهند شد. اجزاء پیوستگی سیستم کامپیوتری باید قادر به ارتباط دادن مستقیم این دستگاه‌ها یا از طریق یک دستگاه میانی باشند، که باید در صورت امکان یک بخش به بازار عرضه‌شده باشد.

وقتی تمام این آنالیزهای اولیه تکمیل شدند و داده‌هایشان گردآوری شد، خریداران یا طراحان باید حجم بالایی از داده‌ها را داشته باشند که الزامات سیستم‌های کامپیوتری را از آن‌ها استخراج نمایند.

پرسش بعدی این است که چگونه از این داده‌ها برای کمک به انتخاب استفاده کنیم؟ آیا شما این داده‌ها را در یک مدل سیستم کامپیوتری مدنظر تدوین می‌کنید و از این اطلاعات برای استخراج آنالیز تحلیلی و کیفی سیستم محاسبه موردنظر و سپس مقایسه این نتایج با دیگر پارامترهای معلوم محصول استفاده می‌کنید؟ یا آیا یک مدل شبیه‌سازی همخوانی بیشتری دارد؟ در هر صورت یک ابزار ارزیابی این داده‌ها باید فراهم‌سازی شود و باید امکان استفاده از داده‌هایی وجود داشته باشد که گردآوری‌شده‌اند.

داده‌های گردآوری‌شده را می‌توان به کلاس‌های کمی و کیفی تقسیم نمود. بدین معنی که یک مجموعه از داده‌ها وجود دارد که از آن‌ها می‌توان معیارهای کارایی ویژه را استخراج نمود و یک معیار دیگر که فقط معیارهای ذهنی را می‌توان از آن استخراج کرد. دیتاستهای کمی باید برای ساخت مدلی از سیستم پیشنهادی و استخراج سنجش‌های ترکیبی برای ارزیابی معماری‌ها و پیکربندی‌های سیستم‌های کامپیوتری موردنظر استفاده شوند. روش‌های مورداستفاده برای این آنالیز مدل‌های تحلیلی و شبیه‌سازی هستند. روش‌های آنالیز بستر تست و عملیاتی ممکن است برای تست گزینه‌های جایگزین در اوایل تحلیل سیستم ها مناسب نباشند.

فصل دوم

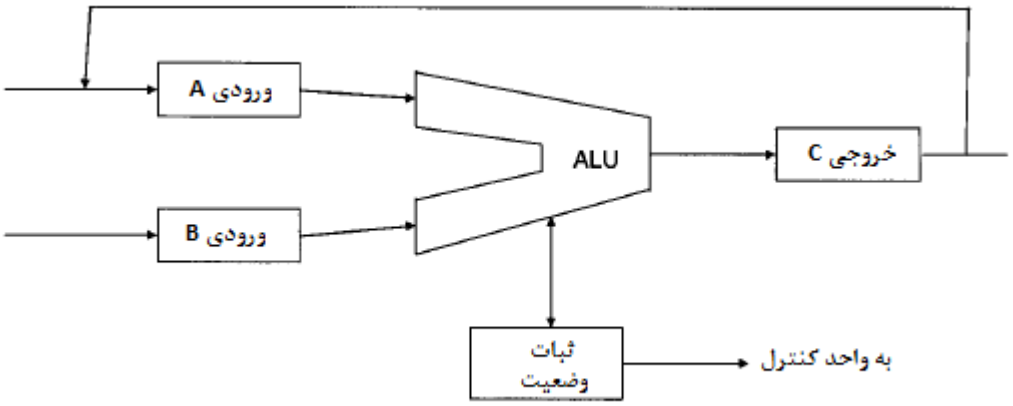
معماری سخت‌افزار پردازش داده‌های کامپیوتری

در این فصل اجزاء سخت‌افزاری و نرم‌افزاری مورداستفاده در کاربردهای مبتنی بر کامپیوتر تعریف‌شده است. در اینجا ترکیب اساسی کامپیوترها (CPUحافظه، ۱۱۰)، دستگاه‌های حافظه ثانویه، دیگر دستگاه‌های ورودی و خروجی پیرامونی، معماری‌های چندپردازنده‌ای، و شبکه‌ها در این فصل گنجانده‌شده‌اند. مباحث ما برای تمرکز بر معماری و استفاده از این اجزاء تنظیم‌شده‌اند چراکه آن‌ها به مدیریت کامپیوتری داده‌های پایا ارتباط می‌یابند.

۲-۱ مقدمه

یک برنامه کاربردی مبتنی بر کامپیوتر بر روی یک سیستم کامپیوتری قرار می‌گیرد. سیستم کامپیوتری واسطه فیزیکی که در آن داده‌های برنامه‌های کاربردی ذخیره می‌شوند و ظرفیت پردازشی برای دست‌کاری داده‌های ذخیره‌شده را ارائه می‌نماید. یک واحد پردازنده یک سیستم کامپیوتری از پنج عنصر اصلی تشکیل یافته است: حافظه، یک واحد منطقی حسابی، یک واحد ورودی، یک واحد خروجی، و یک عنصر کنترلی. واحد حافظه هم داده‌ها برای برنامه‌ها و هم دستورات یک برنامه را ذخیره نموده است که داده‌های ذخیره‌شده را دست‌کاری می‌کنند.

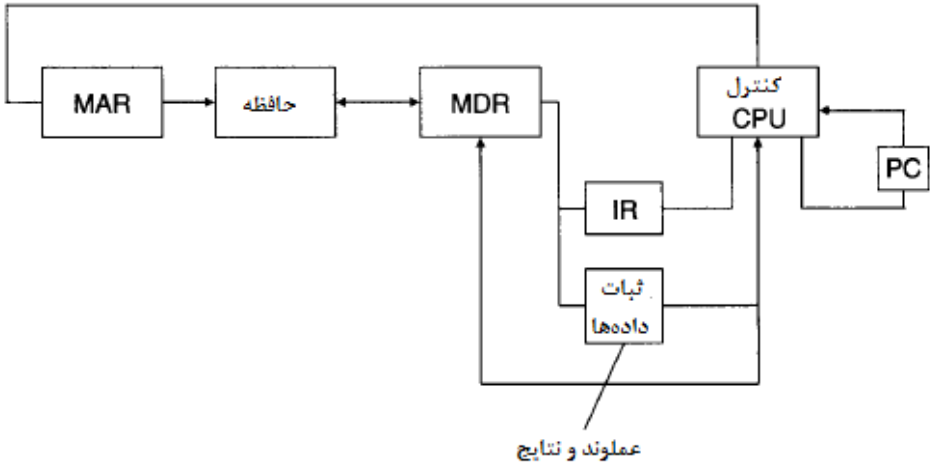
هر یک از عناصر یا دستورات برنامه یک‌به‌یک از حافظه واکشی می‌شوند و توسط واحد کنترل تفسیر می‌شوند. واحد کنترل بسته به تفسیر دستور، تعیین می‌کند که بعداً چه عملیات کامپیوتری انجام شود. اگر دستورالعمل نیاز به داده‌های اضافی نداشته باشد، این کنترل به واحد منطق حسابی اشاره می‌کند که چه عملیاتی را انجام دهد و با چه ثبات‌هایی. (شکل ۲-۱ را ببینید.)



شکل (۲-۱): واحد پردازنده پایه یک کامپیوتر

اگر این دستورالعمل نیاز به داده‌های اضافی داشته باشد، واحد کنترل فرمان مناسب را به حافظه انتقال می‌دهد (MAR، ثبات آدرس حافظه) تا یک آیتم داده‌ها از حافظه واکشی شود(MDR ثبات داده‌های حافظه) و آن را در یک ثبات مناسب در ALU (بانک ثبات داده‌ها) قرار می‌دهد (شکل ۲-۲). این تا زمانی ادامه می‌یابد که تمام عملوندهای موردنیاز در ثبات‌های مناسب ALU باشند. وقتی تمام عملوندها موجود باشند، واحد کنترل به ALU فرمان می‌دهد که دستورالعمل مناسب را به اجرا دربیاورد – به‌عنوان‌مثال، ضرب، جمع یا تفریق. اگر این دستورالعمل ذکر نماید که به یک ورودی یا خروجی نیاز بوده است، عنصر کنترلی می‌تواند یک کلمه را، بسته به دستور، از واحد ورودی به حافظه یا ALU انتقال دهد. در صورت رمزگشایی شدن یک دستورالعمل خروجی، واحد کنترل می‌تواند فرمان انتقال کلمه یا ثبات حافظه مناسب به کانال خروجی ذکرشده را بدهد. این پنج عنصر متشکل از مبانی اساسی مورداستفاده در سیستم کامپیوتری اصلی فون نویمان هستند به شکلی در بیشتر سیستم‌های کامپیوتری معاصر یافته می‌شوند.

در این فصل ما این مبانی را موردبررسی قرار خواهیم داد و خواهیم دید که چگونه برای شکل دادن معماری‌های کامپیوتری مختلف استفاده می‌شوند.



شکل (۲-۲): *دسترسی حافظه* CPU

۲-۲ معماری سخت‌افزار کامپیوتر

یک سیستم کامپیوتری از پنج بخش سازنده که قبلاً توضیح داده شدند و این دستگاه‌های پشتیبانی پیرامونی اضافی تشکیل یافته است، که به انتقال و پردازش داده‌ها کمک می‌کنند. این بخش‌های سازنده اساسی برای شکل‌دهی واحدهای پردازش، کنترل، ذخیره‌سازی، ورودی و خروجی استفاده می‌شوند که سیستم‌های کامپیوتری مدرن را شکل می‌دهند. دستگاه‌ها نوعاً به شکلی سازمان‌دهی می‌شوند که پشتیبان پردازش برنامه‌های کاربردی باشد که سیستم کامپیوتری برای آن مدنظر بوده است – به‌عنوان‌مثال، اگر نیاز به ذخیره‌سازی حجم بالایی از داده‌ها باشد، دستگاه‌های ذخیره‌سازی جانبی اضافی همچون دیسک‌ها یا واحدهای نواری، در کنار کانال‌های داده‌ها و یا کنترلرهای موردنیازشان لازم خواهند بود.

یک معماری سیستم کامپیوتری ساخته‌شده با استفاده از ساختارهای پایه، همچون CPUها، حافظه‌ها، دیسک‌ها، I/O و دستگاه‌های دیگر موردنیاز است.

به‌منظور توصیف بهتر تغییرات درون معماری‌ها، ما مختصراً برخی از جزئیات را شرح خواهیم داد – به‌عنوان‌مثال، واحد منطق حسابی (ALU) واحد کنترلی با یکدیگر در یک واحد پردازنده مرکزی یا CPU ادغام می‌شوند. CPU جریان دستورات و داده‌ها در سیستم کامپیوتری را کنترل می‌کند. حافظه‌ها را می‌توان بر مبنای نزدیک بودن به CPU و سرعت دسترسی در سلسله‌مراتب‌هایی قرار داد – به‌عنوان‌مثال، حافظه کش حافظه کوچک و فوق‌العاده سریعی برای دستورالعمل‌ها و داده‌هایی است که به شکلی فعالانه توسط CPU اجرا و استفاده می‌شوند و معمولاً بر روی همان بورد یا تراشه CPU قرار می‌گیرند. حافظه اولیه کندتر است، ولی ارزان‌تر نیز هست و دربرگیرنده مکان‌های حافظه بیشتری نیز می‌شود. از آن برای ذخیره‌سازی داده‌ها و دستوراتی استفاده می‌شود که در طی اجرای برنامه‌های کاربردی استفاده خواهند شد که در حال حاضر بر روی CPU اجرا می‌شوند – به‌عنوان‌مثال، اگر شما برنامه واژه‌پردازتان را بر روی کامپیوتر شخصیتان بوت کنید، سیستم‌عامل تلاش خواهد نمود تا تمام برنامه واژه‌پرداز را در حافظه اولیه قرار دهد. اگر فضای کافی وجود داشته باشد، سیستم‌عامل برنامه را به چند قسمت تقسیم‌بندی خواهد نمود در صورت نیاز آن‌ها را به داخل می‌کشد.

بخشی از برنامه که در حافظه قابل ذخیره‌سازی نباشد بر روی دستگاه ذخیره‌سازی ثانویه نگهداری می‌شود، که نوعاً یک دیسک درایو است. این دستگاه یک ظرفیت ذخیره‌سازی به‌مراتب بالاتر از حافظه اولیه دارد، هزینه آن به ازای واحد ذخیره‌سازی کمتر است، و دارای زمان‌های دسترسی به داده‌هایی است که به‌مراتب کندتر از حافظه اولیه هستند. یک دستگاه جدیدتر برای ذخیره‌سازی خارجی درایو CD-ROM است. این دستگاه، در حالت فقط خواندنی (ROM)، تنها به کاربران اجازه استخراج اطلاعات از درایو را می‌دهد. در جدیدترین نوع خواندن/نوشتن، دستگاه را می‌توان تا حدودی مانند درایو نواری سنتی استفاده نمود. یک دستگاه ذخیره‌سازی ثانویه دیگر واحد درایو نواری است. یک درایو نواری یک دستگاه ذخیره‌سازی ساده است که می‌تواند حجم عظیمی از داده‌ها را ذخیره نماید – که در اینجا نیز درازای هزینه کمتری نسبت به واحدهای دیسک ولی با سرعت دسترسی کمتر است. دیگر اجزاء یک کامپیوتر واحدهای ورودی و خروجی هستند. این‌ها برای استخراج داده‌ها از کامپیوتر و قرار دادن این داده‌ها در اختیار دستگاه‌های خروجی یا داده‌های ورودی از دستگاه خارجی

استفاده می‌شوند. این دستگاه‌های خروجی می‌توانند پایانه‌های کاربر نهایی، سنسورها، پورت‌های شبکه اطلاعاتی، ویدئو، صوت یا دیگر کامپیوترها باشند.

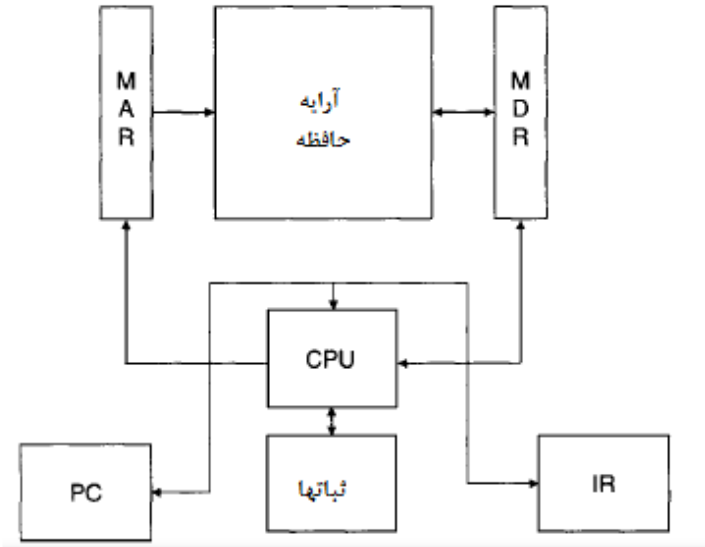
در بخش‌های بعدی، که به نحوه پیوستگی این دستگاه‌ها برای پشتیبانی از برنامه‌های کاربردی پردازش داده‌ها می‌پردازیم، ما هر یک از اجزاء یک سیستم کامپیوتری را به‌طور مفصل موردبررسی قرار خواهیم داد.

۲-۳ معماری‌های CPU

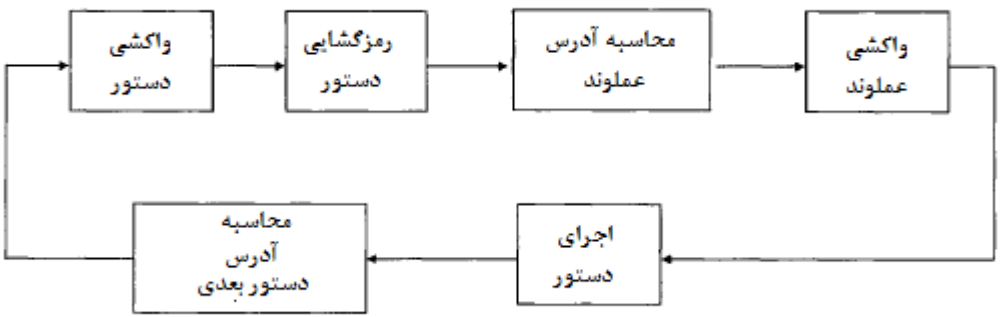
واحد پردازشگر مرکزی (CPU) مغز یک سیستم کامپیوتری است. CPU، همان‌گونه که قبلاً ذکر شد، از واحد منطقی حسابی (ALU) واحد کنترل تشکیل یافته است. ALU می‌تواند در پیکربندی‌های مختلفی از یک واحد ساده تکی، نشان داده‌شده در شکل (۲-۱) ارائه شود که جمع، تفریق، افزایش تدریجی، کاهش تدریجی، بار و ذخیره‌سازی، را انجام می‌دهند تا واحدهای فوق‌العاده پیچیده‌ای که عملیاتی همچون ضرب، تقسیم، به توان رساندن، سینوس گرفتن، کسینوس گرفتن و غیره را اجرا می‌کنند. عملکرد اولیه ALU گرفتن صفر یا عملوندهای دیگر و اجرای عملکرد ذکرشده در دستورالعمل است. علاوه بر ALU، CPU از مجموعه‌ای از ثبات‌ها برای ذخیره‌سازی عملوندها و نتایج میانی و حفظ اطلاعات مورداستفاده CPU برای تعیین وضعیت محاسباتش تشکیل یافته است. ثبات‌هایی برای وضعیت عملکرد ALU، برای حفظ شماره دستوری که باید بعداً اجرا شود، حفظ جریان داده‌ها به داخل یا خارج، حفظ دستوری که در حال اجرا است، و مکان عملوندهایی که توسط CPU وجود دارند.

هر یک از این ثبات‌ها دارای کارکردی منحصربه‌فرد درون CPU است، و هر یک برای کلاس‌های مختلف معماری‌های کامپیوتری ضروری هستند. یک معماری حداقلی نوعی برای یک CPU و ثبات‌های آن در شکل (۲-۳) نشان داده‌شده است. این معماری از یک حافظه اولیه تشکیل یافته است که از طریق باس هایی به CPU متصل شده است برای آدرس‌دهی یک مکان در حافظه و انتقال محتوای مکان از حافظه به ثبات داده‌های حافظه استفاده می‌کند. ثبات‌هایی در CPU برای دستورات (دستورالعمل یا ثبات IR)، عملوندهای دستور، و نتایج عملیات، یک شمارنده مکان (که دربرگیرنده

مکان در حافظه برای دستورات یا عملوندها است، بسته به رمزگشایی دستورات)، یک شمارنده برنامه یا PC (که مکان دستورالعمل بعدی که باید اجرا شود) و ثبات‌های وضعیت وجود دارند.



شکل (۲-۳): CPU و ثبات‌های مرتبط با آن



شکل (۲-۴): چرخه اجرای دستورالعمل

CPU این دربرگیرنده واحد کنترل است. واحد کنترل از ثبات‌ها و دستورات وضعیت در ثبات آموزش برای تعیین اینکه چه کارکردهایی باید CPU بر روی ثبات‌ها انجام دهد، ALU، و مسیرهای داده‌هایی که CPU را شکل می‌دهند استفاده می‌کند. عملکرد اساسی CPU از یک حلقه ساده پیروی می‌کند (مگر آنکه وقفه‌هایی رخ دهند که جریان اجرا را تغییر دهند). این حلقه چرخه اجرای دستورالعمل نامیده می‌شود (شکل ۲-۴). شش کارکرد اساسی برای اجرا در حلقه دستورالعمل وجود دارد: واکشی دستور، رمزگشایی دستور، محاسبه آدرس مؤثر عملوند، واکشی عملوند، اجرای عملیات و محاسبه آدرس بعدی.

واکشی دستورالعمل از ثبات شمارنده برنامه برای اشاره به دستورالعمل بعدی ذخیره‌شده در حافظه استفاده می‌کند. این آدرس در ثبات آدرس حافظه قرار داده می‌شود و سپس دستورالعمل (به‌صورت الکترونیکی علامت‌دهی شده توسط عنصر کنترل CPU برای انتقال داده‌ها) از حافظه داده‌ها به ثبات داده‌های حافظه تحت جهت‌دهی واحد کنترل.

وقتی یک دستورالعمل در ثبات دستورالعمل[۲۴] باشد، سیکل دوم در اجرای دستورالعمل، یعنی رمزگشایی، را می‌توان به اجرا درآورد. به‌منظور رمزگشایی[۲۵] دستورالعمل، واحد کنترل باید تشخیص دهد که چه نوع از دستورالعمل درخواست شده است —به‌عنوان‌مثال، آیا این دستورالعمل نیاز به داده‌های اضافی از حافظه برای اجرای کارکرد موردنظرش دارد، یا آیا این دستورالعمل فقط دربرگیرنده ثبات‌های مقیم ALU است؟

سیکل سوم در اجرای دستورالعمل محاسبه آدرس مؤثر عملوند است. این فاز از اجرای دستورالعمل با استخراج اطلاعات آدرس عملوند از دستورالعمل و سپس اجرای شکلی از محاسبه (مثلاً مبنا به‌علاوه آفست) با این اطلاعات برای شکل‌دهی یک آدرس فیزیکی در حافظه اجرا می‌شود. ما درباره انواع مختلف آدرس‌دهی در بخش‌های بعدی این فصل بحث خواهیم نمود. وقتی نوع و تعداد عملوندها تعیین شد، ALU می‌تواند عملوندها را کسب نماید و سپس مقرر نماید تا دستورالعمل رمزگشایی‌شده به اجرا دربیاید.

---

[۲۴] instruction register

[۲۵] decode

وقتی ما یک آدرس فیزیکی کسب نمودیم، می‌توانیم واکشی عملوند را انجام دهیم (تابع چهارم چرخه اجرای دستورالعمل). برای واکشی عملوند، آدرس مؤثر در ثبات آدرس حافظه قرار داده می‌شود، و این کنترل محتوای اشاره‌شده توسط ثبات آدرس حافظه را در ثبات داده‌های حافظه جهت‌دهی می‌کند. عملوند استخراج‌شده سپس از ثبات داده‌های حافظه به یک ثبات ALU جهت‌دهی می‌شود. اگر یک عملوند اضافی موردنیاز باشد، دو گام چرخه برای واکشی عملوند ممکن است برای دستیابی به عملوند باقیمانده تکرار شوند. با تمام عملوندهای موردنیاز در ثبات‌های ALU، دستورالعمل درخواستی را اکنون می‌توان به اجرا درآورد. اجرای دستور توسط واحد کنترل CPU کنترل می‌شود. واحد کنترل به ALU برای اجرای دستورالعمل اشاره می‌کند – به‌عنوان‌مثال، اگر یک جمع درخواست شود، ALU می‌تواند ثبات‌های A و B را جمع کند و نتیجه حاصله را در ثبات C قرار دهد. پس از تکمیل دستورالعمل گام آخر در سیکل اجرای دستورالعمل را می‌توان به اجرا درآورد.

محاسبه آدرس بعدی از شمارنده برنامه و/یا هرگونه نتیجه مرتبط (همچون یک دستورالعمل نوع go to) برای تعیین اینکه در دستورالعمل بعدی را در کجای حافظه باید یافت استفاده می‌کند. حالت نرمال محاسبه آدرس افزایش تدریجی محتوای شمارنده برنامه است. با این آدرس جدید، چرخه دستورالعمل یک‌بار دیگر شروع می‌شود.

این توالی اجرا توابع اساسی یافته شده در تمام سیستم‌های کامپیوتری را نمایش می‌دهد. انواع مختلف تعداد گام‌ها بر مبنای نوع و طول دستورالعمل یافت می‌شوند.

۲-۳-۱ انواع دستورالعمل

بر مبنای تعداد ثبات‌های در دسترس و پیکربندی این ثبات‌ها چند نوع از دستورالعمل‌ها امکان‌پذیر هستند – به‌عنوان‌مثال، اگر ثبات‌های بسیاری در دسترس باشند، همانند آنچه در یک کامپیوتر پشته‌ای موجود است، هیچ محاسبه آدرسی لازم نیست و، ازاین‌رو، این دستورالعمل می‌تواند، از هر دو جهت فرمت و زمان اجرا، به‌مراتب کوتاه‌تر باشد. از سوی دیگر، اگر هیچ ثبات کلی وجود نداشته باشد و تمام محاسبات توسط انتقال داده‌ها به حافظه انجام شوند، دستورالعمل‌ها طولانی‌تر خواهند بود و به خاطر واکشی و ذخیره‌سازی عملوند نیاز به زمان بیشتری دارند. موارد ذیل نشانگر انواع دستورالعمل هستند:

دستورالعمل‌های صفر آدرسه این نوع از دستورالعمل در ماشین‌هایی یافت می‌شود که در آن‌ها بسیاری از ثبات‌های چندمنظوره در دسترس هستند. این در ماشین‌های پشته‌ای و در برخی از ماشین‌های مجموعه دستور کاهش‌یافته مطرح است. دستورالعمل‌هایی از این نوع کارشان را به‌طور کامل با استفاده از ثبات‌ها به اجرا درمی‌آورند. اگر ما سه ثبات کلی A، B و C داشته باشیم، یک فرمت نوعی می‌تواند از شکل ذیل باشد:

$$R[A] <-- R[B]\ operator\ R[C] \qquad (۱-۲)$$

که نشانگر این است که محتوای ثبات‌های B و C از عملوند (همچون جمع، تفریق، ضرب و غیره) می‌خواهند که بر رویشان عملیات به اجرا دربیاورند، که نتیجه آن در ثبات کلی C ذخیره می‌شود. به‌طور مشابه، ما می‌توانیم دستورالعمل‌هایی را توصیف کنیم که از تنها یک یا دو ثبات از قرار ذیل استفاده می‌کند:

$$R[B] <-- R[B]\ operator\ R[C] \qquad (۲-۲)$$

یا

$$operator\ R[C] \qquad (۳-۲)$$

که دستورالعمل‌های دو ثبات و یک ثبات را، به ترتیب، نشان می‌دهد. در حالت دو ثبات، یکی از ثبات‌های عملوند نیز درنتیجه ثبات استفاده می‌شود. در حالت یک ثبات، ثبات عملوند نیز ثبات نتیجه است. دستورالعمل افزایش تدریجی نمونه‌ای از دستورالعمل یک ثبات است. این نوع از دستورالعمل در تمام ماشین‌ها یافت می‌شود.

دستورالعمل‌های یک آدرسه، در این نوع از دستورالعمل یک آدرس حافظه واحد در دستورالعمل یافته می‌شود. اگر یک عملوند دیگر استفاده شود، نوعاً یک انباره یا بالای یک پشته در یک کامپیوتر پشته‌ای است. فرمت معمول این دستورالعمل‌ها از این شکل است:

$$operator\ M[address] \qquad (۴-۲)$$

که در آن نام محتوای آدرس حافظه نام‌برده شده در کنار یک ثبات ویژه ضمنی بر روی آن‌ها اجرا می‌شود. نمونه‌ای از چنان دستورالعملی می‌تواند از قرار ذیل باشد:

$$Move\ M[۱۰۰] \qquad (۵-۲)$$

یا

$$Add\ M[۱۰۰] \qquad (۶-۲)$$

که محتوای مکان حافظه ۱۰۰ را به انباره ALU انتقال می‌دهد یا محتوای آدرس حافظه ۱۰۰ را به انباره انتقال داده و نتیجه حاصله را در انباره ذخیره می‌کند. اگر لازم باشد که این نتیجه در حافظه ذخیره شود، ما ممکن است نیاز به یک دستورالعمل ذخیره‌سازی داشته باشیم:

$$Store\ M[۱۰۰] \qquad (۷-۲)$$

دستورالعمل‌های یک و نیم آدرسه، وقتی ما یک معماری داشته باشیم که دارای چند ثبات چندمنظوره است، می‌توانیم عملیات پیشرفته‌تری را با ترکیب محتوای حافظه و ثبات‌های عمومی فراهم بیاوریم. این دستورالعمل نوعی عملیاتی را بر روی محتواهای یک مکان حافظه با عملیات یک ثبات عمومی به اجرا درمی‌آورد – به‌عنوان‌مثال، ما می‌توانیم محتوای یک مکان حافظه را به محتوای یک ثبات عمومی، A، به‌صورت نشان داده‌شده اضافه نماییم:

$$Add\ R[A], M[۱۰۰] \qquad (۸-۲)$$

این دستورالعمل نوعاً نتیجه را در اولین مکان یا ثبات نام‌برده شده در دستورالعمل ذخیره می‌کند. در این مثال، این ثبات A است.

دستورالعمل‌های دو آدرسه، دستورالعمل‌های دو آدرسه از دو مکان حافظه برای انجام یک دستورالعمل استفاده می‌کنند – به‌عنوان‌مثال، یک حرکت بلوکی N کلمه‌ای از یک مکان در حافظه به مکان دیگر، یا یک جمع بلوکی. این حرکت می‌تواند به‌صورت ذیل نمود یابد:

$$Move\ N, M[۱۰۰], M[۱۰۰] \qquad (۹-۲)$$

دستورالعمل‌های دو نیم آدرسه، این فرمت از دو مکان حافظه و یک ثبات کلی در دستورالعمل استفاده می‌کند. آنچه در این نوع از دستورالعمل معمول است عملیاتی است که دربرگیرنده دو مکان در حافظه است که نتیجه را به‌عنوان یک ثبات یا یک عملیات با یک ثبات عمومی ذخیره می‌کند و یک مکان حافظه که نتیجه را بر روی مکان حافظه دیگر ذخیره می‌کند، که به این صورت نشان داده‌شده است:

$$R[A] - -\gg M[۱۰۰]\ operator\ M[۱۰۰۰] \qquad (۱۰-۲)$$

$$M[۱۰۰۰] - -\gg M[۱۰۰]\ operator\ R[A]$$

دستورالعمل‌های سه آدرسه، دستورالعمل سه آدرسه یک‌شکل کمتر رایج فرمت دستورالعمل است. این دستورالعمل‌ها دربرگیرنده سه مکان حافظه هستند – دو تا که برای عملوندها استفاده می‌شود و یکی که به‌عنوان مکان نتایج استفاده می‌شود. یک فرمت معمول آن ازاین‌قرار است:

$$M[۲۰۰] - -\gg M[۱۰۰]\ operator\ M[۳۰۰] \qquad (۱۱-۲)$$

۲-۳-۲ معماری‌های دستورالعمل

ایده‌های متعددی درباره نحوه سازمان‌دهی سیستم‌های کامپیوتری حول مجموعه دستورالعمل وجود دارد. یک‌شکل، که با کامپیوترهای بزرگ جدید نمود یافته است، کامپیوتر کم دستور[۲۶] (RISC) است. این ماشین‌ها نوعاً دارای تعداد اندکی از دستورالعمل‌ها هستند که ساده‌اند و تعداد نسبتاً برابر اندکی از سیکل‌های کلاک به ازای هر دستورالعمل را به خود اختصاص می‌دهند. هر یک از این دستورالعمل‌ها بسیار بهینه‌سازی شده است و به شکلی مؤثر عملیاتی می‌شود. برنامه-های ماشین‌آلات معمولاً طولانی‌تر هستند، ولی کد واقعی می‌تواند به علت کد بسیار بهینه‌سازی شده و معمول سریع‌تر اجرا شود.

در سمت دیگر این طیف معماری‌های شکل‌گرفته حول دستورالعمل‌های پیچیده قرار دارند. این کامپیوترها با عنوان کامپیوترهای پر دستور، یا CISC، مورد ارجاع قرار می‌گیرند. این ماشین‌ها از دستورالعمل‌هایی استفاده می‌کنند که هرکدامشان کارکرد پیچیده‌ای را به اجرا درمی‌آورند – به‌عنوان‌مثال، یک‌ضرب ماتریسی یا یک تابع مثلثاتی دست‌کاری پرتعداد. هر دستورالعمل می‌تواند سیکل‌های ماشین متعددی را برای اجرا به خود اختصاص دهد و خودش می‌تواند در ریز کد سطح پایین‌تر کدگذاری شود. برنامه‌های نوشته‌شده در این نوع معماری ممکن است کوتاه‌تر باشند، ولی نمی‌توانند زمان کمتری را به خود اختصاص دهند و در برخی از موارد ممکن است به خاطر پیچیدگی‌شان زمان حتی بیشتری را به خود اختصاص دهند.

---

[۲۶] reduced instruction set computer

۲-۳-۳ طرح‌های آدرس‌دهی حافظه

همان‌گونه که فرمت‌های مختلفی برای دستورالعمل‌ها وجود دارند، راه‌های متعددی نیز وجود دارند که از طریقشان آدرس یک عملوند از یک دستورالعمل تعیین می‌شود. هر شکل از محاسبه آدرس مزایای خودش را از جنبه انعطاف‌پذیری طراحی دستورالعمل دارد. شش نوع عمده از طرح‌های محاسبه آدرس‌دهی در کامپیوترها یافت شده‌اند: بلافاصله، مستقیم، شاخص، پایه، غیرمستقیم، و دو عملوندی. ما مختصراً این موارد را موردبررسی قار خواهیم داد.

بلافاصله ، آدرس‌دهی بلافاصله (فوری) درواقع یک حالت آدرس‌دهی در حافظه نیست، بلکه، این‌یک فرمت دستورالعمل است که مستقیماً دربرگیرنده داده‌هایی است که باید به‌عنوان بخشی از دستورالعمل بر رویشان اقدام صورت پذیرد. این شکل از دسترسی به عملوند چرخه اجرای دستورالعمل را ساده‌سازی می‌کند چراکه نیاز به هیچ واکشی دیگری نیست.

مستقیم – برای آدرس‌دهی مستقیم، نیازی به هیچ رمزگشایی آدرس عملوندی لازم نیست. فیلد آدرس عملوند دستورالعمل دربرگیرنده آدرس فیزیکی عملوند است. این کنترل صرفاً فیلد آدرس عملوند را در فیلد آدرس حافظه قرار می‌دهد و عملوند از حافظه واکشی می‌شود.

شاخص – پالایشی از آدرس‌دهی مستقیم آدرس‌دهی شاخص گذاری شده است. در این شکل از رمزگشایی آدرس عملوند، فیلد آدرس عملوند به محتوای یک ثبات تخصیص داده‌شده افزوده می‌شود تا آدرس فیزیکی مؤثر محاسبه شود.

پایه – آدرس‌دهی پایه بر روی این مفهوم بسط می‌یابد. یک ثبات پایه دربرگیرنده یک‌پایه آدرس است، که به آدرس شاخص گذاری شده افزوده می‌شود تا یک آدرس فیزیکی مؤثر شکل بگیرد. این طرح در سیستم‌های کامپیوتری برای آدرس‌دهی و تقسیم‌بندی حافظه به چند بخش استفاده می‌شود. وقتی بیش از یک ثبات پایه در یک معماری در دسترس باشد، ما می‌توانیم به شکلی آسان‌تر حافظه قسمت‌بندی شده را برای کاربران متعدد و نرم‌افزار کنترل دستگاه‌ها تقسیم‌بندی کنیم.

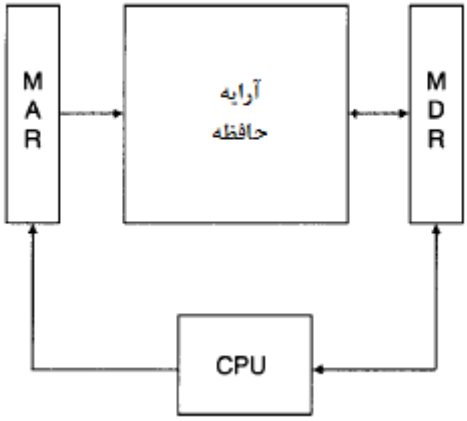
غیرمستقیم – برای این طرح محاسبه آدرس، ما از محتوای یک مکان حافظه مشخص‌شده به‌عنوان آدرس مؤثر استفاده می‌کنیم. کنترل محتوای مکان حافظه نام‌برده شده را واکشی می‌کند و از این به‌عنوان نشانگر ثبات آدرس حافظه برای استخراج عملوند واقعی استفاده می‌کند.

آدرس‌دهی دو عملوندی – در آدرس‌دهی دو عملوندی، هر ترکیبی از طرح‌های فوق را می‌توان باهم برای دسترسی به چند عملوند برای یک دستورالعمل استفاده نمود.

۲-۳-٤ معماری‌های حافظه

مخزن حافظه این می‌تواند دارای یک معماری (پیکربندی) باشد که می‌تواند به ذخیره‌سازی واکشی محتوای حافظه کمک نماید. عموماً یک حافظه به صور یک ساختار معمول سازمان‌دهی می‌شود، که با استفاده از ثبات آدرس حافظه قابل آدرس‌دهی است و داده‌های آن از طریق ثبات داده‌های حافظه انتقال می‌یابند (شکل ۲-۵). این حافظه از طریق ترکیبی از آدرس‌دهی و محرکه‌ها یا سنسورها برای نوشتن و خواندن داده‌ها از ثبات داده‌های حافظه یا بر روی آن مورد دسترسی قرار می‌گیرد. ساختارهای حافظه بر مبنای سازمان‌دهی کلمات حافظه ساخته می‌شوند. ساده‌ترین شکل یک ساختار دوبعدی خطی است. هر مکان حافظه یک خط کلمه منحصربه‌فرد دارد، که، وقتی انرژی دهی شود، محتوای خطوط N بیتی را جهت‌دهی می‌کند (که در آن N اندازه کلمه داده‌ها در کامپیوتر است).

یک سازمان دیگر عبارت است از معماری دو نیم بعدی. در این ساختار حافظه، کلمات حافظه به صفحات داده‌های مجزا تفکیک می‌شوند، که هر یک از یک بیت برای تمام مکان‌های حافظه تشکیل یافته‌اند. برای دسترسی به یک کلمه، n صفحه باید با مختصات X و Y انرژی دهی شود، که متناظر با کلمه حافظه مطلوب است. درایورهای هر یک از صفحات بیت مناسب را به ثبات داده‌های حافظه برای کلمه حافظه آدرس‌دهی شده جهت‌دهی می‌کنند. دیگر سازمان‌های داده‌ها استخراج‌شده‌اند و ما بررسی این موارد را بر عهده خواننده قرار می‌دهیم.



شکل (۲-۵): مکانیسم دسترسی حافظه

۲-۴ معماری‌های I/O

مکانیسم‌های ورودی و خروجی توسط سیستم‌های کامپیوتری برای انتقال اطلاعات به درون یا بیرون حافظه اصلی کامپیوتر استفاده می‌شوند. یک توالی نوعی برای اجرای این انتقال اطلاعات از یک دستگاه ورودی و خروجی یا به آن از قرار ذیل است:

۱. انتخاب یک دستگاه I/O

۲. مشغول – تا زمانی که دستگاه آماده شود منتظر بمانید.

۳. یک کلمه را از بافر دستگاه I/O به انباره CPU انتقال دهید.

۴. محتوای انباره را به یک مکان حافظه انتقال دهید.

۵. مکان حافظه بعدی را برای داده‌های CPU محاسبه کنید.

۶. به گام ۲ برگردید و تا زمانی که تمام داده‌ها انتقال داده شوند آن را تکرار نمایید.

توالی فوق این‌گونه فرض نموده است که تمام داده‌ها باید از CPU عبور داده شوند تا جریان کنترل شود.

اگر در عوض، ما قابلیت قرار دادن استخراج داده‌ها را به‌طور مستقیم در حافظه یا از آن بدون عبور از CPU نداشته باشیم، می‌توانیم به پیشرفت‌های بیشتری ازنظر عملکردها و پالایش معماری دست‌یابیم. برای اینکه CPU داده را بتوان از حلقه I/O خارج نمود، ما به یک عنصر کنترلی اضافی نیاز داریم. امکان کنترل I/O به‌طور مستقیم و دور زدن CPU در مسیر حافظه نیاز به افزایش کنترل دارد، این کنترلر با عنوان یک دستگاه دسترسی حافظه مستقیم (DMA) مورد ارجاع قرار می‌گیرد. دستگاه DMA برای ما امکان آنچه CPU باید انجام دهد را فراهم می‌آورد. CPU یک فرمان آغاز I/O را برای واحد کنترل DMA با آدرس بلوک داده‌هایی که باید انتقال داده شوند صادر می‌کند. CPU اکنون عاری از بالاسری اضافی ورودی و خروجی است و می‌تواند آزاد شود تا چند پردازش دیگر را انجام دهد یا صرفاً منتظر بماند تا زمانی که DMA پاسخ دهد که انتقال کامل شده است. به‌منظور فراهم‌سازی فعالانه این اعلان، یک قابلیت اضافی برای CPU لازم بود: یک قابلیت وقفه. این وقفه‌ها می‌توانند از چند نوع باشند، به‌صورت ذیل:

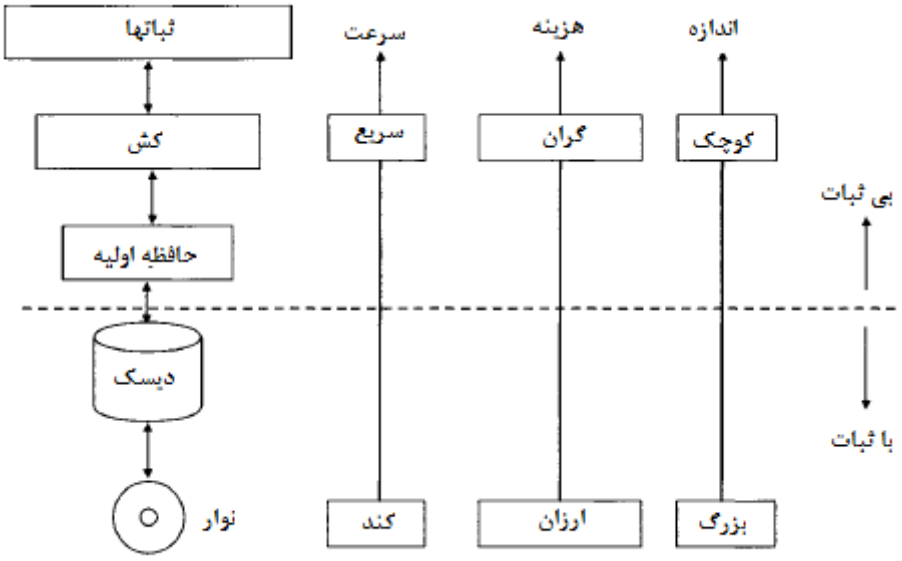
- وقفه‌ها می‌توانند میانی باشند، که موجب وقفه در عملکرد CPU و توقف سرویس شود.
- وقفه‌ها را می‌توان به تعویق انداخت، تا بدین‌صورت CPU بتواند وقتی آماده است به آن‌ها سرویس بدهد.
- وقفه‌ها را می‌توان اولویت‌بندی کرد، و این امکان را فراهم آورد که سرویس برای اقدامات بحرانی که در سیستم رخ می‌دهند ارائه شود.

۲-۵ حافظه ثانویه و معماری‌ها و دستگاه‌های جانبی

در زمان بررسی تهیه یک سیستم کامپیوتری، حجم مخزن حافظه همیشه به‌عنوان یک ویژگی مهم لحاظ می‌شود . فارغ از اینکه سیستم یک کامپیوتر شخصی رومیزی باشد، یا یک کامپیوتر بزرگ[۲۷]، یا پردازنده چندمنظوره بزرگ، یک مکان ذخیره‌سازی داده‌ها همیشه واجد اهمیت بالایی بوده و مطلوب بوده است. با کاهش بهای حافظه، اندازه حافظه خریداری‌شده برای تمام کلاس‌های کامپیوترها افزایش‌یافته است. یک ویژگی غیر شارژی ساختار کلی سلسله‌مراتب حافظه است. فارغ از اینکه سیستم‌ها چقدر پیچیده و یا ساده باشند، ما متوجه خواهیم شد که همه آن‌ها، چیزی مشترک دارند. طراحان سیستم‌ها مخزن داده‌ها را برای حداکثر سازی عملکرد و فراهم‌سازی ذخیره حجم اطلاعاتی مناسب سازمان‌دهی کرده‌اند.

سلسله‌مراتب مخزن (شکل ۲-۶) از مجموعه‌ای از انواع مخزن داده‌ها تشکیل یافته است که به نیازهای اطلاعاتی سیستم پاسخ می‌دهند. از عنصری که بالاترین سرعت را دارد (یک کش) تا عناصری که کمترین سرعت رادارند (دستگاه‌های آرشیوی)، مصالحه‌ای بین هزینه و سرعت واسطه ذخیره‌سازی به ازای واحد حافظه برقرار می‌شود. تلاش بر این است که سرعت پردازنده کامپیوتر با دستگاه‌هایی که بالاترین سرعت را درون یک منحنی هزینه معقولانه دارند تطابق داده شود. در بخش‌های بعدی ما دستگاه‌های ذخیره‌سازی اطلاعات خارج از حوزه واحد پردازنده مرکزی را موردبررسی قرار خواهیم داد. در این بررسی، حافظه‌های کش گران‌قیمت پرسرعت و حافظه اولیه نادیده گرفته می‌شوند. ما مرورمان را با نگاهی به دستگاه‌های نواری، دیسک‌های مغناطیسی، و دستگاه‌های آرشیوی آغاز خواهیم نمود.

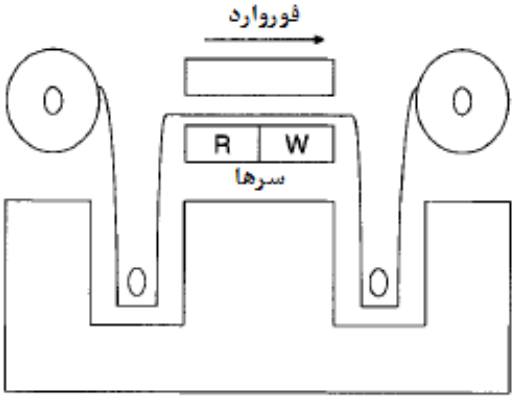
[۲۷] workstation



شکل (۲-۶): سلسله‌مراتب حافظه

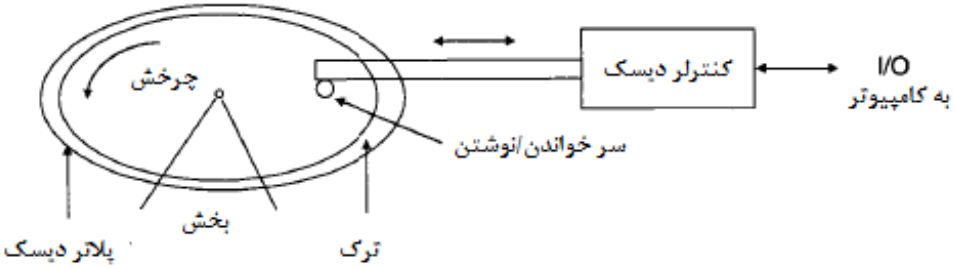
۲-۵-۱ دستگاه‌های ذخیره‌سازی نواری

مخزن اطلاعات نوار مغناطیسی یک واسطه ذخیره‌سازی چگالی بالا و کم‌هزینه را برای داده‌های دسترسی پایین یا دسترسی بالا فراهم می‌آورد. یک واحد نواری از یک واسطه ذخیره‌سازی (یک قرقره مغناطیسی شکل‌گرفته در یک نوار)، لوازم الکترونیکی دسترسی، و اجزاء مکانیکی تشکیل یافته است (شکل ۲-۷). یک واحد نواری به شکلی ساده عمل می‌کند. داده‌های روی یک نوار تنها در شکل متوالی قابل‌دسترسی هستند. داده‌ها باید بر روی نوار مکان‌یابی شده و سپس از نوار حذف شوند. یک درایو نواری می‌تواند به‌صورت مکانیکی یک نوار را باز پیچانی کند، به‌طور متوالی نوار را جستجو نماید، و نوار را متوقف نماید. به‌منظور دسترسی یافتن به داده‌های ذخیره‌شده بر روی یک نوار، یک برنامه I/O می‌تواند به واحد نواری دستور باز پیچی نوار را داده و سپس به‌صورت متوالی نوار را از ابتدا جستجو کند تا زمانی که تطابق یافته شود. پس از یافته شدن آدرس، داده‌ها را می‌توان حذف نمود.



شکل (۲-۷): دیاگرام شماتیک یک سیستم مخزن نوار مغناطیسی

به‌منظور بهبود عملکرد واحدهای نواری، طرح‌های دسترسی معنایی مخزن اضافی ارائه‌شده‌اند. ابتدای نوار برای نگهداری نشانگرها در نقاط شروع فایل‌های ذخیره‌شده بر روی نوار بود. به‌جای جستجوی متوالی کل نوار، کنترلر دایرکتوری نوار را جستجو می‌کند، محل ذخیره شدن داده‌ها بر روی نوار را می‌یابد (مثلاً در فاصله چند فوت از بخش دایرکتوری)، و سپس از این اطلاعات برای فست فوروارد به مکان عمومی استفاده می‌کند که در آن می‌توان جستجوی خطی را ادامه داد. این امکان افزایش سرعت را در دسترسی و انتقال داده‌های ذخیره‌شده بر روی دستگاه فراهم می‌آورد، یک ویژگی مهم درزمانی که سیستم مدیریت پایگاه داده دخیل باشد.



شکل (۲-۸): دیاگرام شماتیک یک سیستم دیسک نوری یا مغناطیسی

۲-۵-۲ دستگاه‌های ذخیره‌سازی دیسک نوری و مغناطیسی

یک پیشرفت رخ‌داده در حوزه ذخیره‌سازی نواری واحدهای دیسک است، که بیشتر کاربران کامپیوتر از آن مطلع‌اند. این دیسک‌ها می‌توانند قابل درآوردن بوده یا از شکل‌های ثابت داخلی باشند. یک واحد دیسک نوعاً متشکل از یک یا چند تا از موارد ذیل است: یک کنترلر، یک بازوی دسترسی قابل حرکت، و یک واسطه ذخیره‌سازی مغناطیسی در شکل یک پلاتر چرخنده (شکل ۲-۸ را ببینید). این پلاتر(ها) بر روی یک دوک نصب می‌شوند، که با سرعتی مشخص می‌چرخد. این پلاتر در مجموعه‌ای از حلقه‌ها سازمان‌دهی شده است که ترک‌ها[۲۸] نامیده می‌شوند و یک تقسیم‌بندی از این ترک‌ها بخش[۲۹] نامیده می‌شود.

بازوی قابل برداشتن دربرگیرنده سخت‌افزار حسگری و حرکتی است تا امکان خواندن و نوشتن داده‌های مغناطیسی یا نوری ذخیره‌شده بر روی پلاتر فراهم بیاید. این کنترلر دسترسی داده‌های ذخیره‌شده را بر مبنای الگوریتم‌های دسترسی مختلف تنظیم می‌کند، که در اینجا فقط درباره ساده‌ترین آن‌ها بحث می‌کنیم. ساده‌ترین شکل دسترسی دیسک شکلی است که در پارادایم جستجوی متوالی یافته می‌شود. کنترلر دیسک می‌داند که بر روی چه بخش و ترکی یک فایل داده‌ها ذخیره‌شده است و با استفاده از این کنترلر دیسک باید برخی از کارکردهای ساده را به اجرا دربیاورد، مثلاً انتقال بازوی دسترسی به بیرون برای چک کردن اینکه داده‌ها بر روی آن ذخیره‌شده‌اند (این جستجو نامیده می‌شود و زمانی که صرف می‌کند زمان جستجو نام دارد).

به‌محض قرارگیری بر روی ترک مناسب، کنترلر باید بخش مناسبی را بیابد که در آن داده‌ها ذخیره‌شده‌اند. این الزام می‌کند که کنترلر شروع نشانگرهای بخش‌بر روی ترک را تشخیص دهد و بخش مناسب را در حین عبور از زیر سنسورهای بازوی دسترسی بیابد. زمان موردنیاز برای این زمان چرخش نامیده می‌شود. وقتی این بازو بر روی بخش و ترک مناسب قرار گرفت، داده‌ها از واسطه به کنترلر قابل‌انتقال هستند. این زمان، زمان انتقال نامیده می‌شود.

---

[۲۸] tracks
[۲۹] sectors

ازاین‌رو، برای دسترسی متوسط یک فایل داده‌ها بر روی یک دیسک ما باید زمان ذیل را داشته باشیم:

$$T = t\{seek\} + t\{rotate\} + t\{transfer\} \qquad (۱۲-۲)$$

به‌راحتی می‌توان از این دید که زمان دسترسی به داده‌ها بر روی یک واحد دیسک بیشتر از زمان حافظه اولیه است و معمولاً ممکن است کمتر از زمان لازم برای استخراج مقدار مشابهی از داده‌ها از یک واحد نواری باشد.

چگالی دیسک بر مبنای واسطه استفاده‌شده برای ذخیره‌سازی داده‌ها است. واحدهای دیسک ساخته‌شده بر روی یک واسطه مغناطیسی نسبتاً در حال متراکم‌تر شدن هستند، ولی در حال نزدیک شدن به محدوده‌هایشان هستند. علاوه بر این، این واسطه به خاطر آلاینده‌هایی که از طریق هوا منتقل می‌شوند و میدان‌های مغناطیسی در معرض خرابی است. به‌منظور بهبود این وضعیت، این صنعت تکنولوژی دیسک‌های نوری را توسعه داده است. این تکنولوژی جایگزین واسطه مغناطیسی با یک واسطه نوری می‌شود که در آن داده‌ها به‌عنوان واسطه نوری انعکاسی ذخیره می‌شوند. این واسطه مشابه با آن چیزی است که در اجرا گرهای دیسک نوری تلویزیونی مشاهده می‌شود.

۲-۵-۳ دستگاه‌های ذخیره‌سازی و آرشیو اطلاعات

حتی با دسترسی به تمام تکنولوژی نوار و دیسک، نمی‌توان تمام داده‌های موردنیاز را آنلاین نگه داشت. به‌منظور نگهداری داده‌هایی که فقط از جنبه شغلی موردنیاز هستند، ما نیازمند دستگاه‌های ذخیره‌سازی و آرشیو اطلاعات هستیم. دستگاه‌های ذخیره‌سازی و آرشیو اطلاعات دارای واسطه قابل برداشتن هستند. اگر شما به سیستم‌های چندرسانه‌ای جدید دسترسی دارید یا یک کامپیوتر شخصی یا کامپیوتر بزرگ برای استفاده دارید، با شکلی از دستگاه آرشیو اطلاعات تراکنش نموده-اید: دیسک قابل برداشتن، دیسک فشرده، یا کارتریج نوار. این قابل‌رؤیت‌ترین شکل از دستگاه ذخیره‌سازی آرشیوی است. داده‌ها در صورت لزوم در سیستم بارگیری می‌شوند و پس از انجام کار برداشته می‌شوند. تازه‌ترین دستگاه ذخیره‌سازی آرشیوی توسعه داده شد، درایو خواندن/نوشتن

CD، به‌تدریج تمایز بین ذخیره‌سازی آنلاین و آرشیو اطلاعات را کمرنگ کرده است. بسیاری از سیستم‌ها از CD درایوها به‌عنوان مخزن ارتقاءیافته برای حافظه برنامه‌های کاربردی بلندمدت استفاده می‌کنند. برخی از سیستم‌ها حتی به‌جایی رسیده‌اند که در آن‌ها این‌ها نشانگر مخزن آنلاین اولیه هستند.

سیستم‌های آرشیو اطلاعات پیچیده‌تر دیگری توسعه داده‌شده‌اند که از ترکیبی از سیستم‌های مکانیکی و الکتریکی برای اتصال آنلاین و آفلاین واسطه‌ها استفاده می‌کنند. این‌ها مشابه با مجلات دیسک فشرده هستند و شبیه جوک باکس‌ها هستند. وقتی یک آیتم داده‌ای خاص لازم باشد، مکان ذخیره‌سازی فیزیکی آن یافته می‌شود، و این واسطه به‌صورت آنلاین در سلسله‌مراتب ذخیره‌سازی فعالانه قرار داده می‌شود که در آن داده‌های آرشیوشده اکنون قابل‌دسترسی خواهند بود. در اینجا نیز، وقتی درباره یک پایگاه داده بسیار بزرگ صحبت کنیم، این‌یک ویژگی مفید به شمار می‌رود.

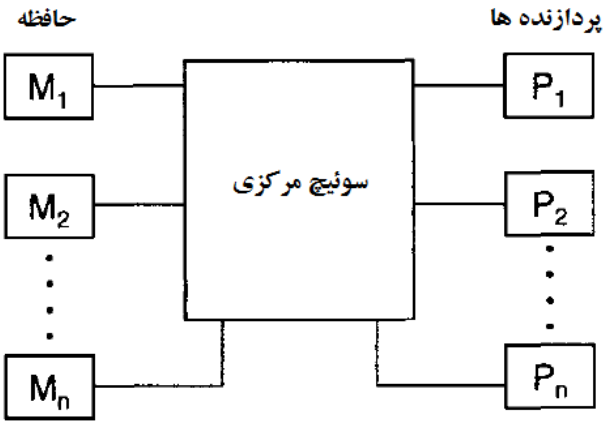
۲-۶ معماری‌های توزیع‌شده و شبکه

همه سیستم‌ها از یک کامپیوتر تشکیل نشده‌اند. سیستم‌های مدرن مورداستفاده در دانشگاه‌ها، کسب‌وکار و دولت به تعداد دفعات بیشتری به هم متصل می‌شوند تا سیستم‌های اشتراک اطلاعات یا سیستم‌های چندپردازنده‌ای شکل بگیرند. این شبکه‌ها و پیوستگی‌های کامپیوتری با فراهم‌سازی یک مسیر ورودی و خروجی دیگر برای کامپیوتر برای دریافت یا ارسال اطلاعات ساخته می‌شوند. واحد ورودی و خروجی و کنترلر برای دستگاه جانبی شبکه یک واحد رابط شبکه (NIU) یا باس پردازنده نامیده می‌شوند. کارکرد این واحدها و باس های رابط فراهم‌سازی یک شیوه (نوعاً) یکپارچه برای یک کامپیوتر به‌منظور تراکنش با کامپیوتر دیگر به شکلی است که انگار آن‌ها در ماشین یکسانی قرارگرفته‌اند. شبکه‌ها در پیکربندی‌های مختلفی ارائه می‌شوند – به‌عنوان‌مثال، NIUها را می‌توان به‌عنوان یک توپولوژی باس جهانی واحد، به‌صورت یک توپولوژی ستاره مرکزی یا هاب، به‌صورت یک توپولوژی حلقه‌ای، یا به‌عنوان یک روش ترکیبی پیکربندی کرد. وقتی پیوستگی به چنان طرقی بر روی یک‌فاصله نسبتاً اندک انجام شود (یک طبقه، ساختمان، یا سازمان کوچک)، آنچه با عنوان یک شبکه محلی، یا LAN، نامیده

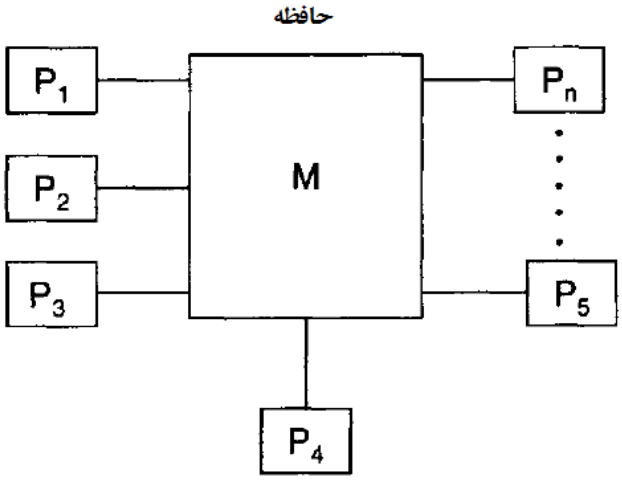
می‌شود را دارا خواهیم بود. یک LAN برای اتصال دادن یک زیر واحد از یک سازمان بزرگ‌تر یا ارتباط دادن تعداد کمتری از کاربرانی استفاده می‌شود که نیاز به اشتراک‌گذاری اطلاعات دارند. برای یک LAN، ما شبکه‌های گسترده و اینترنتی راداریم. سیستم‌های چندپردازنده‌ای با استفاده از مفاهیم مشابه به هم متصل می‌شوند. آن‌ها با استفاده از باس های مشترک یا حافظه مشترک ترکیب می‌شوند.

۲-۶-۱ عناصر رابط کامپیوتر به شبکه

این شبکه می‌تواند به طرق بسیاری شکل بگیرد: می‌تواند یک عنصر سوئیچینگ مرکزی داشته باشد، که می‌تواند یک کامپیوتر مستقل باشد که به‌عنوان یک روتر عمل می‌کند (شکل (۲-۹) (a) را ببینید)، این می‌تواند یک مخزن ذخیره‌سازی مرکزی مشترک داشته باشد، یا می‌توان آن را با استفاده از واحدهای رابط هوشمند به یک واسطه ارتباطات متصل نمود.



شکل (۲-۹) (a) سیستم کامپیوتری چندپردازنده‌ای با حافظه توزیع‌شده

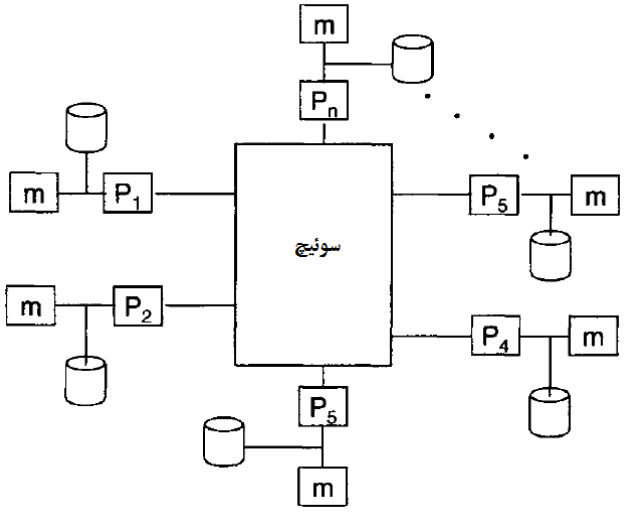


شکل (۲-۹) (b) سیستم کامپیوتری چندپردازنده‌ای با حافظه Simms

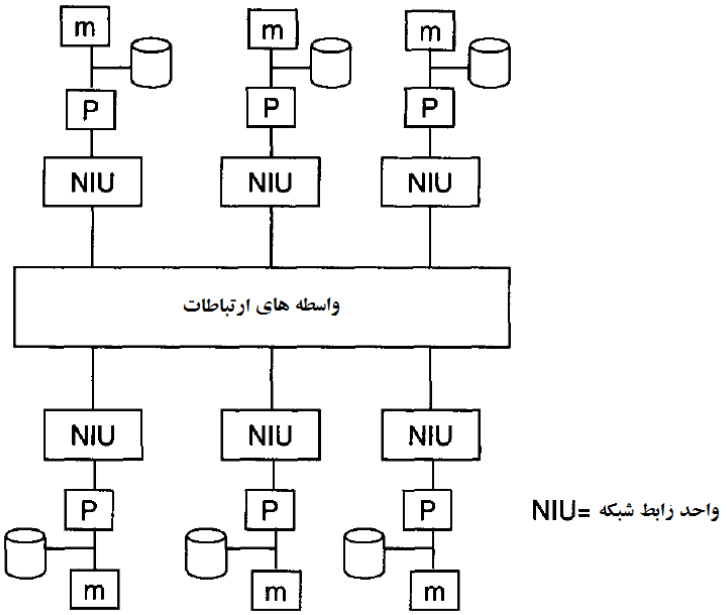
پیکربندی مورداستفاده به درجه همگام‌سازی و کنترل موردنیاز، و این توزیع بین کامپیوترها بستگی دارد.

این چندپردازنده به‌هم‌پیوسته از یک حافظه مرکزی اشتراکی به‌عنوان دستگاه پیوستگی استفاده می‌کند (شکل ۲،۹ (b) را ببینید). تمام پردازنده‌ها بر روی شبکه از حافظه مرکزی برای دسترسی انتقال داده‌ها میان پردازنده‌های به‌هم‌پیوسته استفاده می‌کنند. این معماری توزیع‌شده ابزاری آسان برای هماهنگ‌سازی بین پردازنده‌ها فراهم می‌آورد. یک تمایز این است که هر پردازنده هیچ حافظه محلی ندارد، تمام دستورالعمل‌ها و داده‌ها از بانک حافظه اشتراکی کسب می‌شوند و از طریق یک سیستم مخزن ثانویه اشتراکی به هم ارتباط می‌یابند. هر پردازنده سیستم‌عامل و مخزن محلی خودش را برای برنامه‌ها و داده‌های محلی دارد. هماهنگ‌سازی از طریق انتقال داده‌ها از یک سیستم کامپیوتری به سیستم دیگر از طریق دستگاه ذخیره‌سازی مشترک انجام می‌شود. تبادل داده‌ها و سیگنالینگ انتقال‌ها از طریق مکانیسم‌هایی همچون پیام یا هماهنگ‌سازی بخش‌های ذخیره‌سازی مشترک در واسطه ذخیره‌سازی ثانویه مدیریت می‌شوند.

یک پالایش مضاعف باعث حذف دستگاه ذخیره‌سازی ثانویه مشترک می‌شود و آن را با یک عنصر سوئیچینگ ارتباطات جایگزین می‌کند. این سوئیچ این امکان را فراهم می‌آورد که هر یک از سیستم‌های گسسته کامپیوتری اطلاعات را میان خودشان آدرس‌دهی نموده و ارسال کنند.



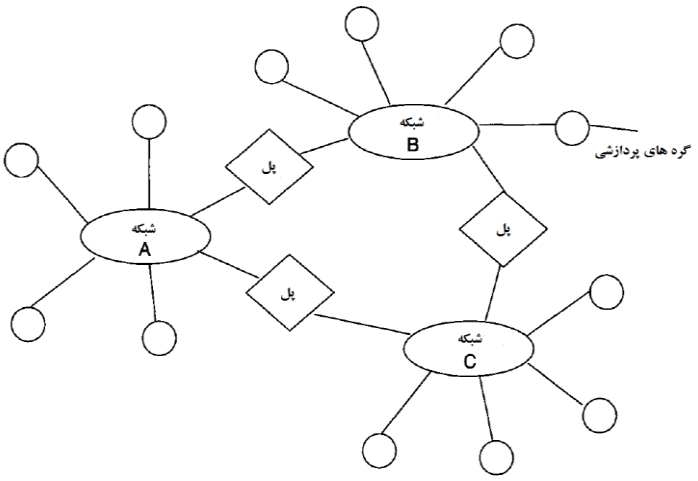
شکل (۲-۹) (c) : سیستم کامپیوتری چندپردازنده‌ای با حافظه محلی خصوصی



شکل (۲-۹) (d): سیستم کامپیوتری چندپردازنده با یک زیرسیستم ارتباطات

هر سیستم کامپیوتری حافظه محلی خودش را دارد و می‌تواند دستگاه‌های ذخیره‌سازی ثانویه اضافی داشته باشد (شکل ۲،۹ (c) را ببینید). هر کامپیوتر با آدرس‌دهی سیستم فراخوانی شده با سیستم‌های به‌هم‌پیوسته، شکل‌دهی یک ارتباط، و سپس آغاز یک محاوره، ارتباط برقرار می‌کند. این با نحوه محاوره ما از طریق یک سیستم تلفنی قابل‌مقایسه است. این سیستم توزیع‌شده مبتنی بر سوئیچینگ برای هماهنگ‌سازی دسترسی نیاز به نرم‌افزار اضافی در هر سمت دارد.

یک ارتقاء اضافی حذف سوئیچ مرکزی و جایگزینی آن با یک مسیر ارتباطی مشترک است (شکل (۲-۹) (d) را ببینید). این مسیر می‌تواند یک باس اشتراکی، یک حلقه، یک یا واسطه ستاره‌ای باشد. کامپیوترهای به‌هم‌پیوسته هرکدام باید یک واحد به‌هم‌پیوسته واسطه‌ای باشند، که دسترسی به‌واسطه را کنترل می‌کند. این معماری نیاز به نرم‌افزار کنترل و خط‌مشی‌های اضافی دارد تا امکان کنترل بر واسطه مشترک فراهم بیاید. تنها یک کامپیوتر در هر بار می‌تواند به‌واسطه دسترسی داشته باشد و اطلاعات ارسال کند. ما در بخش‌های بعدی خواهیم دید که این نرم‌افزار چگونه عمل می‌کند.



شکل (۲-۱۰): ارتباط دادن شبکه‌ها از طریق یک پل

۲-۶-۲ پل‌های شبکه

ما می‌توانیم با معرفی یک واحد کنترل شبکه‌بندی دیگر شبکه محلی یا سیستم‌های چند پردازشی را بسط دهیم. اتصال دادن شبکه‌های متعدد یا سیستم‌های چند پردازشی نیاز به یک پل دارد (شکل ۲،۱۰ را ببینید). یک پل را می‌توان به‌عنوان یک دستگاه تطبیق سرعت برای همگام‌سازی ترافیک بین شبکه‌ها لحاظ نمود. پل‌ها معمولاً دربرگیرنده نرم‌افزار و سخت‌افزار برای واسطه‌گری پیام‌های ورودی، تعیین و یک‌سوسازی انواع در آدرس‌ها بر روی شبکه‌های به‌هم‌پیوسته، و فوروارد کردن پیام‌ها به واحد آدرس‌دهی شده هستند. روترها و سوئیچ‌هایی که در بیشتر پیکربندی‌های متوسط تا بزرگ یافت می‌شوند در این طبقه‌بندی دستگاهی قرار می‌گیرند.

۲-۷ توپولوژی‌های شبکه

همان‌گونه که قبلاً ذکر شد، توپولوژی‌های پیوستگی مختلفی در شبکه‌های محلی وجود دارند. آن‌ها عبارت‌اند از توپولوژی‌های باس جهانی، حلقه، و ستاره.

۲-۷-۱ توپولوژی باس جهانی

یک باس جهانی یک واسطه اشتراکی واحد است، که در هر بار تنها توسط یک دستگاه قابل‌استفاده است. باس جهانی با انواع مختلفی از طرح‌ها کنترل می‌شود. یکی از ساده‌ترین آن‌ها طرح دسترسی چندگانه حسگری حامل است. این پروتکل با استفاده از دو اصل کار می‌کند: اول، تأخیر ناشی از ارسال یک بیت از یکسر باس به سر دیگر و، دوم، توانایی ارسال و سپس گوش دادن به‌واسطه. این پروتکل در ساده‌ترین شکلش به‌صورت ذیل عمل می‌کند:

- به باس گوش دهید – اگر مشغول بود، منتظر بمانید، اگر رفع شد، دیتا را ارسال کنید.
- وقتی دیتا ارسال شد، به گوش دادن و مقایسه آنچه شنیده‌شده با آنچه ارسال‌شده ادامه دهید.
- اگر، در تمام مدت ارتباط سربه‌سر، آنچه ارسال‌شده بود با آنچه شنیده‌شده تطابق داشته باشد، من باس را کنترل می‌کنم و می‌توانم ارسال یک پیام را ادامه دهم (فرض در اینجا این است که اگر من برای یک‌زمان انتقال سربه‌سر منتظر بمانم، آنگاه تمام گره‌های دیگر باید پیام مرا شنیده باشند و درصورتی‌که مایل به انتقال هستند اکنون تأخیر داشته باشند).
- پس از تکمیل، به حالت گوش دادن برگردید.
- اگر من همان پیامی که ارسال کرده‌ام را نشنوم، یک تصادم بر روی باس رخ‌داده است. من بلافاصله انتقال را متوقف می‌کنم و تأخیر می‌کنم و سپس سعی می‌کنم مجدداً ارسال را انجام دهم.

با استفاده از این پروتکل ساده، دستگاه‌ها بر روی شبکه می‌توانند پیام‌ها را به شکلی نسبتاً مؤثر ارسال و دریافت کنند. مشکل این پروتکل این است که ذاتاً پهنای باند واسطه را در فرایند ارسال و حسگری هدر می‌دهد.

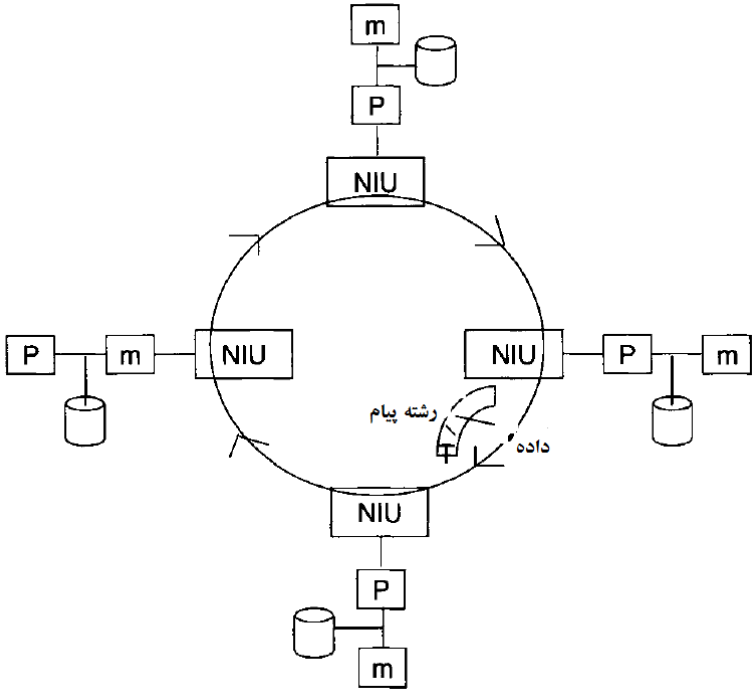
یک رویکرد متفاوت به کنترل دسترسی به یک باس جهانی مبتنی بر یک طرح حفاظتی است. در یک طرح حفاظتی[۳۰] پهنای باند در دسترس به چند بخش تقسیم می‌شود، که سپس به دستگاه‌های مختلف روی شبکه تخصیص داده می‌شوند. برای دسترسی به‌واسطه برای ارسال داده‌ها، یک دستگاه ابتدا باید منتظر بماند تا اسلات حفاظتی‌اش قابل‌دسترسی شود. چند طرح وجود دارد که از طریقشان اسلاتها را می‌توان تخصیص داده و کنترل کرد. مشکلی که با این رویکرد وجود دارد این است که ذاتاً استاتیک است. اسلاتها را نمی‌توان به‌راحتی از یک سیستم به سیستم دیگر باز تخصیص نمود. انواع متعددی از این پروتکل در سیستم‌هایی با درجات مختلف موفقیت توسعه داده‌شده‌اند.

۲-۷-۲ توپولوژی حلقه‌ای

توپولوژی حلقه‌ای سیستم‌های کامپیوتری در شبکه در یک حلقه پیوسته را به هم مرتبط می‌کند. پیام‌ها در شبکه از یک سیستم کامپیوتری به سیستم دیگر جریان می‌یابند تا زمانی که به فرستنده برگردند. (شکل ۲-۱۱ را ببینید). این توپولوژی امکان استفاده بهتر از واسطه را فراهم می‌آورد. این واسطه را می‌توان به شکاف‌هایی تقسیم نمود که در شبکه جریان بیابند. این شکاف‌ها بسته به وجود داشتن یا نداشتن یک پیام در شکاف (اسلات) به‌عنوان خالی یا پر نشان گزاری می‌شوند. برای ارسال یک پیام یک کامپیوتر ابتدای شکاف را سنس می‌کند و چک می‌کند که آیا پر است یا خالی. اگر اسلات پر باشد، فرستنده منتظر شکاف بعدی می‌ماند. اگر شکاف خالی باشد، فرستنده پیامش را قرار می‌دهد. مشکلی که با این طرح وجود دارد این است که اندازه اسلات اندازه پیام‌هایی می‌توانند در یک اسلات واحد ارسال شوند را محدود می‌کند. وجود انواع مختلفی از این پروتکل باعث کاهش این مسئله شده است، ولی آن‌ها مجموعه مسائل خودشان رادارند. یک پروتکل متفاوت، که امکان پیام‌هایی با اندازه متغیر را فراهم می‌آورد، پروتکل حلقه قرار دهی است. این پروتکل نیاز به پشتیبانی سخت‌افزاری برای واسطه‌گری پیام‌های ورودی دارد که ممکن است با پیام یک فرستنده تداخل داشته باشند. کامپیوتری که خواستار ارسال پیامی بر روی شبکه است در صورتی می‌تواند پیام را ارسال کند که هیچ ترافیک پیام دیگری توسط فرستنده سنس نشود. اگر

[۳۰] reservation scheme

پیام دیگری در ورودی فرستنده در طی ارسال پیام خودش از راه برسد، فرستنده صرفاً پیام ورودی را صف‌بندی می‌کند و، پس از تکمیل، آن را به پیام ارسال ضمیمه می‌نماید.



شکل (۲-۱۱): توپولوژی حلقه

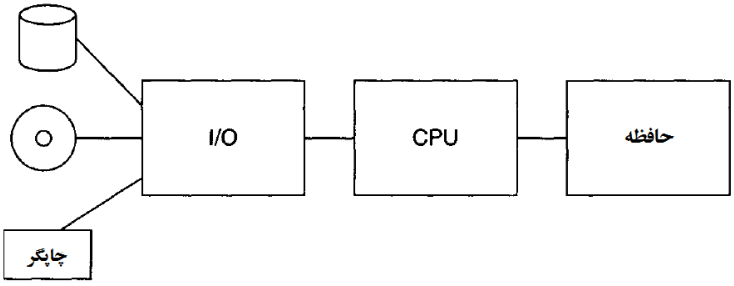
۲-۷-۳ توپولوژی ستاره‌ای

توپولوژی ستاره‌ای چیدمان فیزیکی یک ستاره را دارد. در مرکز آن‌یک پردازنده شبکه مرکزی قرار دارد، که گره‌ها در اطرافش ترتیب یافته و به نقطه مرکزی متصل شده‌اند. هزینه‌های سیم بندی با این توپولوژی می‌تواند به طرز قابل‌توجهی بالاتر باشد.

۲-۸ معماری‌های کامپیوتر

در ادامه بحث قبلی‌مان درباره پیکربندی‌های کامپیوتری ما بررسی خواهیم نمود که چگونه اجزاء مختلف را می‌توان به هم پیوند زد تا یک سیستم کامپیوتری شکل بگیرد. پیش‌فرض مبنایی این معماری تسریع

حرکت داده‌ها به‌منظور فراهم‌سازی امکان افزایش پردازش است. در کانون این معماری پایه CPU با یک حافظه اصلی و سیستم ورودی/خروجی در هر سمت از CPU وجود دارد (شکل ۲-۱۲ را ببینید). در این معماری، تمام داده‌ها تحت کنترل CPU به درون، بیرون و از طریق CPU عبور می‌کنند. این نشانگر معماری فون نویمان پایه است که قبلاً توضیح داده شد. تغییرات در این معماری برای برداشتن بار کنترل تمام حرکت‌های داده‌ها از روی دوش CPU طراحی‌شده‌اند.



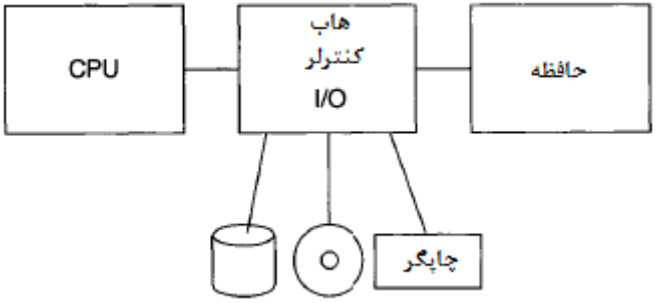
شکل (۲-۱۲): معماری کامپیوتر پایه

۲-۸-۱ معماری‌های کنترلر I/O مرکزی

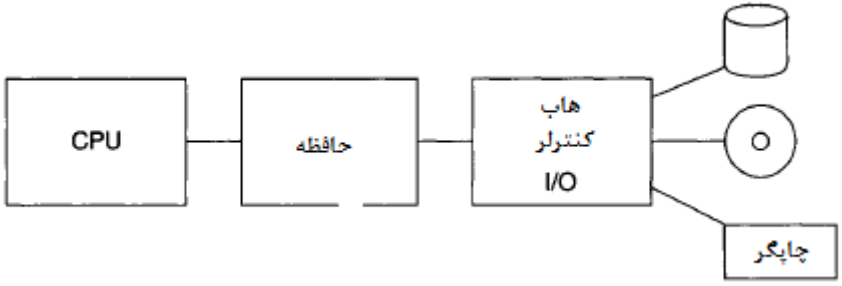
برای به‌منظور برداشتن کارکرد مرکزی هماهنگ‌سازی تمام گردش داده‌ها از عهده CPU، معماری کنترلر ورودی/خروجی مرکزی توسعه داده شد (شکل ۲-۱۳ را ببینید). محوریت سیستمی این معماری با IOC است، و CPU، حافظه اصلی، و دستگاه‌های ۱۱۰ به هاب IOC متصل می‌شوند. به‌منظور انتقال داده‌ها از حافظه اصلی به یک دستگاه ۱۱۰، CPU می‌تواند به IOC فرمان دهد که انتقال را آغاز کند. این داده‌ها می‌توانند تحت کنترل IOC از حافظه اصلی از طریق IOC به دستگاه خروجی نام‌برده شده جریان یابند. مشکلی که در رابطه با این معماری وجود دارد این است که CPU باید از IOC نیز برای انتقال داده‌ها از حافظه اصلی به CPU استفاده کند. این منجر به کاهش بالقوه در عملکرد CPU می‌شود. انواع مختلف این معماری دارای یک مسیر ثانویه به حافظه اصلی برای سرویس بهتر به CPU هستند.

۲-۸-۲ معماری‌های نقشه‌برداری شده حافظه

حافظه اصلی مکانی در سیستم کامپیوتر است که در آن تمام داده‌ها و دستورالعمل‌ها به داخل و خارج جریان داشته باشند. درنتیجه این، یک معماری مطرح گردید که حافظه اصلی عنصر مرکزی آن بود (شکل ۲-۱۴ را ببینید). حافظه اصلی بین CPU و ۱۱۰ قرار می‌گیرد. تمام جریان داده‌ها بین I/O و CPU از حافظه می‌گذرند. چند طرح کنترلی برای کنترل دسترسی به حافظه اشتراکی تهیه‌شده است. یکی از این طرح‌ها تقسیم‌بندی حافظه به چند بخش است: یک بخش برای CPU برای استفاده و یکی برای هر یک از دستگاه‌های I/O در سیستم. CPU برای فرستادن داده‌ها به یک دستگاه I/O صرفاً مکان حافظه را برای دستگاه آدرس‌دهی می‌کند. با این کار، ثبات ورودی دستگاه مستقیماً با داده‌ها پر می‌شود. برای ، انتقال I/O همانند یک نوشتن در حافظه اصلی است.



شکل (۲-۱۳): معماری کامپیوتر با استفاده از یک کنترلر I/O

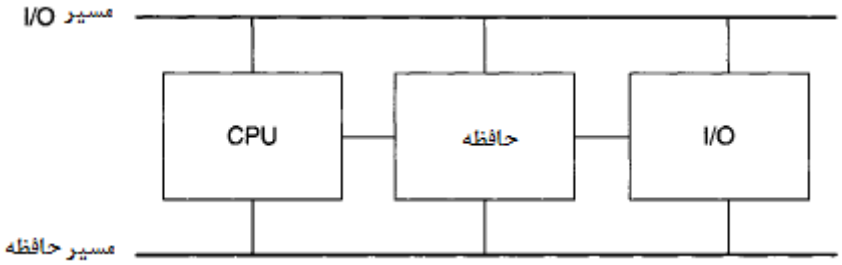


شکل (۲-۱۴): یک سیستم کامپیوتری سازمان‌دهی شده حول حافظه

۲-۸-۳ معماری باس مشترک

یک معماری که مشابه با معماری شبکه جهانی قبلاً توضیح داده‌شده است معماری تک مسیر[۳۱] است. معماری تک مسیر یا جهانی از یک باس ارتباطات واحد برای ارتباط دادن حافظه، CPU، و دستگاه‌های I/O استفاده می‌کند (شکل (۲-۱۵) را ببینید). این عناصر به باس مرتبط‌اند و با استفاده از آدرس‌های روی باس با یکدیگر ارتباط برقرار می‌کنند. همانند مورد مربوط به شبکه، این طراحی منجر به کاهش کاربرد می‌شود، اگر منازعات بین دسترسی‌های باس متناوب باشند. این معماری با موفقیت در چند کامپیوتر تجهیزات دیجیتالی اولیه استفاده می‌شد و همچنان در بسیاری از سیستم‌ها مورداستفاده قرار می‌گیرد.

شکل (۲-۱۵): معماری تک مسیره



شکل (۲-۱۶): معماری دومسیره

[۳۱] unibus

۲-٤-۸ معماری دومسیره

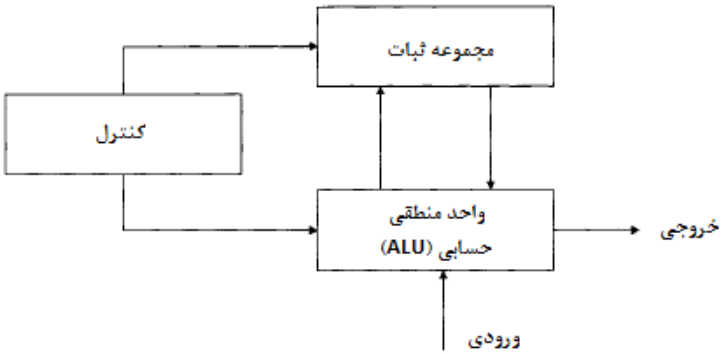
یک اصلاح بر روی معماری تک مسیره معماری دومسیره (شکل ۲-۱۶) است. در این معماری، هاب مرکزی کامپیوتر یک پیکربندی دومسیره است: یک باس (مسیر) برای ترافیک حافظه و یکی برای ترافیک I/O. تمام دستگاه‌ها، حافظه اصلی، دیسک‌ها، نوارها، پایانه‌ها، و دستگاه‌های دسترسی حافظه مستقیم به هر دو مسیر متصل می‌شوند. این معماری یکی از ارتباطات بین دسترسی‌های حافظه CPU و انتقال‌های I/O را حذف نمود. CPU و حافظه برای انتقال فعالانه داده‌ها به حافظه یا از آن، همانند دستگاه‌های ۱۱۰، بدون تداخل، آزاد بودند. یک دستگاه ۱۱۰ می‌تواند در یک بخش از حافظه بنویسد درحالی‌که CPU به‌طور همزمان در حال ارزیابی بخش دیگر بوده است. معماری‌های استخراج‌شده از این فلسفه در سیستم‌های کامپیوتری مدرن رواج بیشتری دارند. ما در ادامه بحثمان درباره معماری‌ها و عملیات سیستم مدیریت، نحوه استفاده از این معماری‌ها و عناصر سیستم کامپیوتری توسط سیستم‌های مدیریت پایگاه داده را موردبررسی قرار خواهیم داد.

۲-۹ سیستم‌های کامپیوتری معماری نرم‌افزاری را پشتیبانی می‌کنند

یک برنامه کاربردی مبتنی بر سیستم کامپیوتری نیاز به سرویس‌ها و پشتیبانی شرکتی از مجموعه‌ای از سخت‌افزار و نرم‌افزار کامپیوتری برای اجرای کارکرد تعیین‌شده‌اش نیاز دارد. این برنامه‌های کاربردی نیاز به یک پلتفرم محاسباتی دارد که متشکل از یک CPU، حافظه و مخزن داده‌های ثانویه و این یک زیرساخت عملیاتی پشتیبان متشکل از یک سیستم‌عامل، سیستم مدیریت پایگاه داده، مدیریت شبکه، و اجزاء دیگر مدیریت فرایند و منابع است. برای درک نحوه استفاده یک برنامه کاربردی مبتنی بر کامپیوتر از این اجزا، ما ابتدا باید عملیات این عناصر زیرساختی نرم‌افزاری را درک کنیم.

واحد پردازنده مرکزی (CPU) و حافظه اصلی موتور محاسباتی اولیه را شکل داده و از اجرای تمام نرم‌افزارها درون کامپیوتر پشتیبانی می‌کنند. CPU از مجموعه‌ای از ثبات‌ها، زیر واحدهای محاسباتی، مسیرهای داده‌ها، و ثبات‌های وضعیت تشکیل یافته است که برای انتقال داده‌ها و انجام دست‌کاری‌های اساسی بر روی این داده‌ها استفاده می‌شوند (شکل ۲-۱۷). به‌عنوان‌مثال یک CPU می‌تواند جمع و

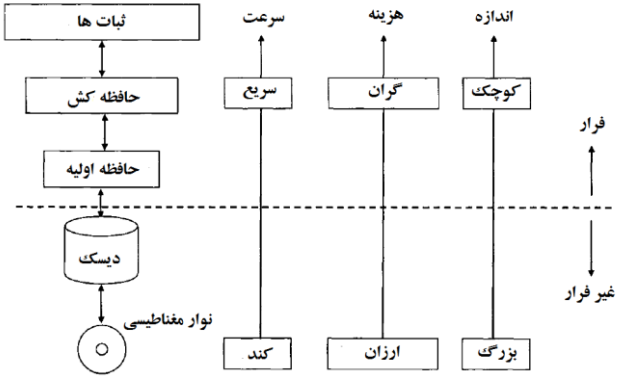
تفریق، ضرب، تقسیم و مقایسه مقادیر را انجام دهد یا انتقال از یک مکان به مکان دیگر را ساده‌سازی نماید. عملیات پایه‌ای وجود دارند، که مابقی زیرساخت سیستم مبتنی بر آن‌ها است و در آن مستقر می‌شود. CPU این دربرگیرنده برخی سخت‌افزارهای پشتیبان اضافی، همچون تایمرها، چفت‌ها و ثبات‌های وقفه، ثبات‌های ورودی و خروجی، و پیوستگی‌ها است. برای کسب جزئیات بیشتر درباره این عناصر به بخش‌های بعدی این فصل مراجعه کنید.



شکل (۲-۱۷): موتور محاسباتی

علاوه بر CPU، عنصر اولیه دیگر درون سیستم پایه حافظه است. یک سلسله‌مراتب حافظه نوعاً متشکل از ثبات‌های داده‌های پرسرعت، حافظه کش سریع، حافظه اولیه، و مخزن ثانویه می‌شود (شکل ۲-۱۸). سلسله‌مراتب حافظه در نزدیک‌ترین نقطه به سخت‌افزار CPU مملو از ثبات‌های بسیار گران‌قیمت و محدود پرسرعت است. این ثبات‌ها برای انتقال تعداد بسیار محدودی از آیتم‌های داده‌ها به درون و بیرون CPU برای پردازش واقعی استفاده می‌شوند. سطح دوم این سلسله‌مراتب حافظه کش است. یک حافظه کش بانکی از حافظه پرسرعت است که به شکلی سازمان‌دهی شده امکان بازیابی سریع داده‌ها را فراهم می‌آورد، اجرای آن تقریباً در سرعت روی تراشه یا ثبات‌های CPU است. یک حافظه کش برای این استفاده می‌شود که داده‌ها را در محتمل‌ترین حالت برای استفاده بعدی در مجاورت CPU و در ذخیره-سازی سریع نگه دارد. مشکلی که با حافظه کش و ثبات‌ها وجود دارد این است که آن‌ها بسیار گران هستند، و ازاین‌رو احتمال اینکه در یک معماری یافته شوند کاهش می‌یابد. این نوع از سخت‌افزار

ذخیره‌سازی نیاز به پشتیبانی زیرساختی اضافی از سیستم‌عامل و سخت‌افزار دارد تا مناسب‌ترین داده‌ها را در مناسب‌ترین سطح سلسله‌مراتب حفظ نماید.



شکل (۲-۱۸): سلسله‌مراتب حافظه

این کنترل نوعاً توسط ترکیبی از سخت‌افزار و نرم‌افزار مدیریت حافظه انجام‌شده است که از اصول مکان ارجاع و مکان استفاده برای تعیین اینکه چه اطلاعاتی باید در سطح مخزن مناسب قرار داده شود و چه اطلاعاتی حذف شود بهره می‌برد.

عنصر سوم سلسله‌مراتب حافظه، حافظه اولیه است. محدوده ذخیره‌سازی حافظه اولیه در بیشتر ماشین‌های امروز صدها مگابایت است. این حجم از حافظه امکان مقیم حافظه بودن نسبت‌های بالایی از یک کار پردازش داده‌ها را برای برنامه‌های کاربردی پردازش داده‌های کوچک فراهم می‌آورد. این بدان معنا نیست که هیچ تبادل اطلاعاتی بین حافظه اولیه واحدهای دیسک ذخیره‌سازی ثانویه وجود ندارد. حجم ذخیره‌سازی بر روی چنان واحدهایی اکنون در محدوده ده‌ها گیگابایت است. تأکید اصلی در یک سیستم کامپیوتری اکنون بر این است که چگونه و چه چیزی مدیریت این سلسله‌مراتب را انجام می‌دهد. مدیر حافظه سیستم می‌تواند بهترین کار ممکن را برای این برنامه‌های کاربردی نوعی انجام دهد، ولی به قیمت تمام برنامه‌های کاربردی عملکرد بالای دیگر سیستم‌عامل.

برای گردآوری عناصر سخت‌افزاری کامپیوتری که قبلاً توضیح داده شد، یک سیستم کامپیوتری در حال کار نیازمند خط‌مشی‌ها و مکانیسم‌هایی برای کنترل این منابع و هماهنگ‌سازی بین آن‌ها برای وجود داشتنشان است. این نوعاً کارکرد سیستم‌عامل یک کامپیوتر بوده است. یک سیستم‌عامل از نرم‌افزار ویژه

با پشتیبانی سخت‌افزاری برای مدیریت تراکنش CPU و تمام عناصر سخت‌افزاری دیگر پشتیبان نرم-افزار برنامه‌های کاربردی اجرائی بر روی سیستم‌عامل است.

۲-۹-۱ معماری سیستم‌های عامل

یک سیستم‌عامل نرم‌افزار کامپیوتری است که در یک سطح پایین با سخت‌افزار سیستم کامپیوتری برای مدیریت اشتراک‌گذاری منابع کامپیوتر میان برنامه‌های کاربردی نرم‌افزاری مختلف تراکنش می‌نماید. یک سیستم‌عامل به‌عنوان ممتازترین مورد در میان عناصر نرم‌افزاری بر روی سیستم اجرا می‌شود و برای وقفه‌ها و تایمرها نیاز به پشتیبانی سخت‌افزاری پایه دارد تا بر کنترل بر اجرای برنامه‌ها تأثیر بگذارد. یک سیستم‌عامل معمولاً سرویس‌های ذیل را ارائه می‌نماید:

۱. مدیریت سخت‌افزار (وقفه هندلینگ، مدیریت تایمر)

۲. همگام‌سازی و ارتباطات بین فرایندی

۳. مدیریت فرایند

۴. تخصیص منابع (زمان‌بندی، توزیع)

۵. مدیریت و دسترسی ذخیره‌سازی (۱۱۰)

۶. مدیریت حافظه

۷. مدیریت فایل

۸. حفاظت از سیستم و منابع کاربر

۹. مدیریتی ارتباطات بین فرایندی

۱۰. مدیریت شبکه

یک سیستم‌عامل با مدیریت سخت‌افزار یک سیستم کامپیوتری آغاز می‌شود. مدیریت سخت‌افزاری نیاز به قابلیتی برای تنظیم محدودیت‌ها بر نگهداری منابع و قابلیتی برای برگرداندن کنترل از یک برنامه اجرائی به سیستم‌عامل است. این کارکردها از طریق استفاده از تایمرهای سخت‌افزاری و سرویس‌های وقفه محقق می‌شوند. یک تایمر سخت‌افزاری شمارنده‌ای است که می‌تواند برای مقدار مشخصی (دوره زمانی) تنظیم شود. وقتی این زمان پایان یافت، یک سیگنال وقفه منتشر می‌شود، که پردازنده را متوقف می‌کند، وضعیت پردازنده را ذخیره می‌کند (تمام محتوای ثبات فعال، ثبات‌های ALU، ثبات‌های وضعیت، نشانگرهای پشته، شمارنده‌های برنامه، ثبات‌های دستورالعمل و غیره را ذخیره می‌کند)، و کنترل را بر روی یک‌روال سرویس وقفه انتقال می‌دهد. روال سرویس وقفه محتوای ثبات‌های از پیش تعریف‌شده (مثلاً ثبات وضعیت CPU یا یک ثبات وقفه از پیش تعریف‌شده) را موردبررسی قرار می‌دهد یا مکان‌های حافظه را تعیین نموده و مشخص می‌کند که بعداً چه عملیاتی باید انجام شوند. نوعاً، کنترل بلافاصله بر روی هسته[۳۲] سیستم‌عامل برای سرویس‌دهی وقفه قرار داده می‌شود.

مدیریت وقفه و سمافورها

استفاده از وقفه‌ها یکی ابزارهای یک سیستم‌عامل برای اعمال کنترل بر سخت‌افزار سیستم است. یک ابزار دیگر استفاده از اقدامات و دستورالعمل‌های کنترلی و نرم‌افزاری همکار است. مفهومی که در اینجا توصیف‌شده است انحصار متقابل[۳۳] است. یک سیستم‌عامل، برای تضمین دسترسی واحد غیر انتقالی به یک منبع، باید ابزاری برای محدودسازی دسترسی به یک منبع و مکانیسم تخصیص منبع از طریق یک عملگر انحصاری متقابل داشته باشد. یک مبنای انحصار متقابل باید دارای قابلیتی برای محدودسازی دسترسی به یک بخش یا منبع از طریق فقط یک عنصر در هر بار باشد، حتی درزمانی که برای دسترسی همزمان تلاش می‌شود (اقدام اتمی). عملیات همه‌چیز یا هیچ‌چیز یک اقدام اتمی برای دسترسی و کنترل تضمین‌شده و غیرمتداخل بر منابع سیستم

[۳۲] kernel
[۳۳] mutual exclusion

توسط سیستم‌عامل لازم است. یک دستورالعمل سخت‌افزاری ویژه به نام تست و تنظیم در بسیاری از سیستم‌های کامپیوتری برای پشتیبانی از این عمل اولیه انحصار متقابل ارائه می‌شود. این دستورالعمل در یک سیکل دستورالعمل اتمی یک متغیر مشخص‌شده را می‌خواند، مقدار آن را در قابل یک مقدار پایه تست می‌کند، و درصورتی‌که شرایط تست‌شده معتبر باشد متغیر را در یک مقدار جدید تنظیم می‌نماید.

دستورالعمل تست و تعیین مبنا را برای ساخت سمافورها شکل می‌دهد. یک سمافور یک متغیر سیستم است که تنها دریکی از دو حالت وجود دارد، یا درست و یا نادرست، و هیچ حالت معتبر دیگری برای این متغیر وجود ندارد. متغیرهای سمافور عملیات اتمی دارند که بر رویشان قابل انجام است، و هیچ عملیات دیگری خارج از این موارد عملیات معتبر نیست. عملیات معتبر از دو نوع هستند. عملیات اول درخواستی برای تنظیم متغیر است، که گاهی با عنوان $P(۵)$ مورد ارجاع قرار می‌گیرد. عملیات دوم درخواستی برای ریست کردن متغیر است و گاهی با عنوان $V(۵)$ مورد ارجاع قرار می‌گیرد. این‌ها بیشتر مانند یک فیلپ فلاپس در یک مدار منطقی عمل می‌کنند. این فلیپ‌فلاپس را می‌توان ست یا ریست کرد، که دارای یک مقدار صفر یا یک باشد. عملیات ست یا ریست یک متغیر سمافور برای ساخت عملیات لاک و آنلاک بر روی منابع یا نگهداری یا آزادسازی عملیات بر روی منابع استفاده می‌شوند. سمافورها برای ساخت مانیتورها استفاده می‌شوند، که دربرگیرنده کنترل منبع کنترل‌شده یک سیستم‌عامل است. به‌عنوان‌مثال، یک مانیتور را می‌توان به‌عنوان ابزاری برای محدودسازی دسترسی به واحد نواری به یک فرایند در یک‌زمان با ساخت یک صف از فرایندهای در حال انتظار و یک‌روال سرویس استفاده نمود. این عملیات می‌تواند برای ساخت یک پوسته بیرونی حول روال سرویس نواری باشد تا در هر بار تنها امکان دسترسی به یک فرایند وجود داشته باشد. اپراتورهای P و V را می‌توان برای این تابع استفاده نمود.

$$P(S) if\ S = ۰\ THEN\ S := ۱\ ELSE\ Enqueue\ requester$$
$$V(S) S := ۰, If\ Queue \qquad (۱۳-۲)$$
$$V(S) S := ۰, If\ Queue <> null\ then\ dequeue$$

فرایندهایی که مایل به استفاده از روال سرویس نواری هستند سرویس را با ابتدا درخواست تابع ست $P(S)$ درخواست می‌کنند. اگر هیچ فرایندی در حال حاضر از نوار استفاده نکند، متغیر S صفر خواهد بود. اگر این متغیر آزاد باشد، متغیر ست می‌شود، و فرایند مجاز به ورود به بخش بحرانی کد معکوس

شده برای روال سرویس نوار و استفاده از روال می‌شود. اگر روال نوار در حال استفاده باشد (که با متغیر سمافور S تنظیم‌شده در یک نشان داده می‌شود)، درخواست در صف قرار می‌گیرد، و منتظر انتشار منبع خواهد بود. وقتی یک فرایند با روال سرویس نوار تمام شد، عملیات ریست یا V(S) درخواست می‌شود. عملگر ریست مقدار سمافور را به مجدداً به صفر ریست می‌کند و صف فرایندهای در حال انتظار را تست می‌کند تا ببیند که آیا فرایندی هنوز وجود دارد که نیاز به سرویس داشته باشد یا نه. اگر فرایندهایی در انتظار باشند، صدر صف حذف‌شده و یک درخواست V(S) برای این فرایند منتشر می‌شود، و فرایند از ابتدا آغاز می‌گردد.

بدین طریق، با استفاده از سمافورها، مانیتورهای پیچیده‌ای را می‌توان برای کنترل دسترسی به منابع نرم‌افزاری و سخت‌افزاری مختلف ساخت. مانیتورها و سمافورها به‌عنوان ابزاری برای ساخت مکانیسم‌های همگام‌سازی برای هماهنگی اقدامات منابع همکاری استفاده‌شده‌اند. به‌عنوان‌مثال، با استفاده از سمافورهای P و V توصیف‌شده، می‌توان دو روال مدیریت منبع همکار را با استفاده از سه متغیر سمافور و عملگرهای P و V، به‌صورت ذیل، ساخت:

$$P(S)if\ S = ۰\ THEN\ S := ۱ \qquad P(S۱)If\ S۱ = ۰\ THEN\ S۱ := ۱$$
$$P(M)if\ M = ۰\ THEN\ M := ۱ \qquad P(M)If\ M = ۰\ THEN\ M := ۱$$
$$Resource\ A\ Service\ Routine \qquad Resource\ B\ Service\ Routine \qquad (۱۴-۲)$$
$$V(M)\ M := ۰ \qquad V(M)\ M := ۰$$
$$V(S۱)\ S۱ := ۰ \qquad V(S)\ S := ۰$$

این دو سمافور (S۱ و S) می‌توانند امکان عملکرد همزمان دو منبع را به شکلی فراهم بیاورند که بتوانند به عقب و جلو تغییر وضعیت بدهند، چه روال سرویس Aی منبع ابتدا با روال سرویس B منبع دنبال شود و چه روال سرویس B با روال Aی منبع دنبال شود. بااین‌حال، آن‌ها را نمی‌توان به علت استفاده از سمافور M به‌طور همزمان اجرا نمود. می‌توان از این مثال برخی از نیازهای ابتدائی توابع سیستم مدیریت پایگاه داده پیاده‌سازی شده با استفاده از مفاهیم مشابه برای تضمین دسترسی محدود به اطلاعات ذخیره‌شده در پایگاه داده و روال‌های مدیریتی را مشاهده نمود.

مدیریت فرایند

یک فرایند نوعاً به‌عنوان پایین‌ترین سطح قابل‌اجرا نرم‌افزار شناسایی‌شده توسط سیستم‌عامل لحاظ می‌شود. فرایندها می‌توانند دارای لایه‌های مدیریت داخلی اضافی باشند که خارج از حوزه سیستم‌عامل هستند. بااین‌حال، این فرایند معادل با یک برنامه کاربری نیست. یک برنامه کاربری را می‌توان به چند فرایند تقسیم نمود، یا می‌تواند یک فرایند واحد باشد. لزومی ندارد که این فرایند یک تصویر اندازه ثابت در سیستم باشد. [بلکه] می‌تواند شکل‌ها و فرم‌های مختلفی به خود بگیرد. جنبه مهم یک فرایند این است که یک‌نهاد قابل‌سنجش وجود دارد که سیستم‌عامل درباره آن اطلاع دارد و درباره آن اطلاعاتی دارد، حداقل از جنبه نحوه تراکنش این فرایند و تطابقش با منابع در حال مدیریت.

مدیریت فرایند کار مدیریت فرایندهای نرم‌افزاری بر روی یک سیستم کامپیوتری را انجام می‌دهد. سیستم‌عامل سرویس‌هایی را برای ایجاد یک فرایند (ایجاد و تکمیل یک بلوک کنترل فرایندی برای یک فرایند جدید)، برای لغو یک فرایند (حذف بلوک کنترل فرایند آن)، تقسیم‌بندی یک فرایند به چندوظیفه، الحاق وظایف، و تقسیم فرایندها فراهم می‌آورد. یک فرایند با استفاده از یک بلوک کنترل فرایند، یا PCB، در سیستم‌عامل توصیف می‌شود. PCB برای یک فرایند، بر مبنای نمونه‌سازی اولیه‌اش در سیستم، ایجاد می‌شود. یک بلوک کنترل نوعی دربرگیرنده اطلاعاتی همچون مشخص‌کننده فرایند، یک نوع فرایند (کاربر، سیستم، پایگاه داده، شبکه و غیره)، اولویت فرایند، اطلاعات حالت فرایند، الزامات منبع فرایند (حافظه، دیسک‌ها، لوازم جانبی، فرایندهای دیگر، و غیره)، وضعیت منابع موردنیاز، اندازه فرایند، و مکان بار حافظه پردازشی حاضر می‌شود. این تنها مجموعه نمونه‌ای از اطلاعات است و به‌هیچ‌عنوان کامل نیست.

سیستم‌عامل از اطلاعات بلوک کنترل فرایند از تمام فرایندهای درون سیستم برای هماهنگ‌سازی اجرای تمام فرایندها به‌منظور تحقق برخی از اهداف سیستم‌عامل، همچون اجرای مناسب، زمان‌های اجرای معادل، یا برخی زمان‌های اجرای متوسط مینیموم استفاده می‌کند. فرایندها درون سیستم‌عامل در سطوح مختلفی اجرا می‌شوند. برخی از فرایندها تقدم دارند و، درنتیجه، می‌توانند به بخش‌های حفاظت‌شده حافظه یا روال‌های مخفی دسترسی داشته باشند. فرایندهای برنامه‌های کاربردی ممکن است هیچ دسترسی بیرونی به‌جز تصویر بار فوری برنامه‌نویس نداشته باشند. دیگران، همچون سیستم مدیریت پایگاه

داده، دارای شکلی از حقوق دسترسی در بین این دو سمت هستند. فرایندها درون این سیستم از سیاری از جهات مختلف کمال موجود هستند، که حالت‌ها نامیده می‌شوند. یک فرایند درون سیستم می‌تواند یکی از این چهار حالت باشد: آماده اجرا، در حال اجرا، معلق یا مسدود، و خاتمه یافته یا مرده (شکل ۲-۱۹).

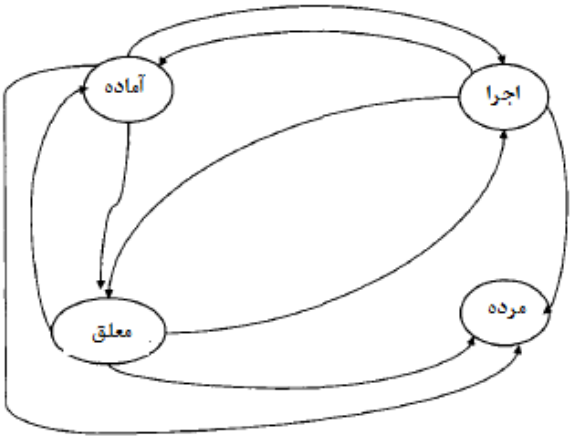
حالت آماده به حالتی ارجاع دارد که یک فرایند درزمانی که در انتظار انتقال از سیستم‌عامل است در آن قرار دارد. این فرایند برای آنکه در این حالت باشد باید تمام منابعی که برای اجرای تخصیص‌یافته یا حداقل به‌طور کامل مشخص‌شده به آن‌ها نیاز دارد را دارا باشد، و باید دارای یک حالت معلوم برای منابع ذخیره‌شده در PCB داشته باشد. انتقال‌ها از حالت آماده دربرگیرنده خاتمه، توزیع یا مسدودسازی می‌شوند.

خاتمه فرایند ممکن است حاصل یک اقدام کاربری برای لغو فرایند یا فرمانی از یک فرایند دیگر، همچون یک فرایند سیستم‌عامل، ناشی از حذف منبع یا یک شرایط خطا (مثلاً یک کلمه کنترلی بد برای یک چاپگر) باشد. توزیع یک فرایند باعث انتقال یک فرایند از حالت آماده به حالت اجرائی به علت یک اقدام زمان‌بندی می‌شود. انتقال بلوک باعث انتقال یک فرایند از حالت آماده به حالت انتظار می‌شود و ناشی از حذف یک منبع کسب‌شده توسط سیستم‌عامل یا برای یک کاستی دیگر است که امکان جلو رفتن فرایند را فراهم نمی‌آورد.

حالت اجرا به مقطعی اشاره دارد که در آن فرایند کنترل CPU را دارد و دستورالعمل‌هایش را بر روی ماشین عریان به اجرا درمی‌آورد. این فرایند در این سطح بر سخت‌افزار کنترل دارد و تنها با وقفه‌ای از سیستم‌عامل یا یک شرایط خطا حذف می‌شود. انتقال‌ها به این وضعیت تنها تحت کنترل سیستم‌عامل رخ می‌دهند و به علت اقدامات زمان‌بندی هستند. انتقال‌های خارج از این حالت به علت شرایط مختلفی هستند. یک فرایند می‌تواند به‌محض خاتمه اجرا از حالت اجرا به حالت خاتمه انتقال یابد، یا یک فرایند می‌تواند به علت یک درخواست ورودی/خروجی (که تحت سرویس یک فرایند دیگر است) یا به حالت آماده، به علت وقفه‌ای از سیستم‌عامل برای برخی شرایط دیگر، به حالت انتظار برگردد.

حالت انتظار یا معلق برای یک فرایند برای نگهداری فرایندهایی استفاده می‌شود که منابع موردنیاز را برای اجرا کسب ننموده‌اند یا اینکه به علت یک اقدام مسدودسازی از اجرای فعالانه حذف‌شده‌اند. یک اقدام انتظاری می‌تواند به علت انتقال داده‌ها از دیسک به حافظه یا تکمیل یک فرایند همکارانه باشد.

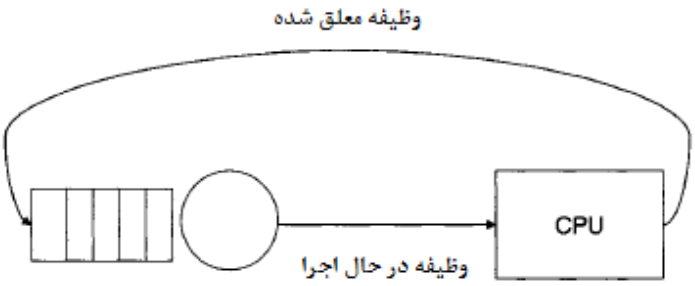
انتقال‌ها به حالت انتظار نوعاً ناشی از درخواست‌ها برای افزوده شدن منابع، حذف یا تغییر مکان برخی از منابع موردنیاز در انتظار یک فرایند همکارانه برای خاتمه سرویسش ، یا انتظار برای اینکه منابع برای درخواست‌های دیگر آزاد شوند.



شکل (۲-۱۹): حالات فرآیند

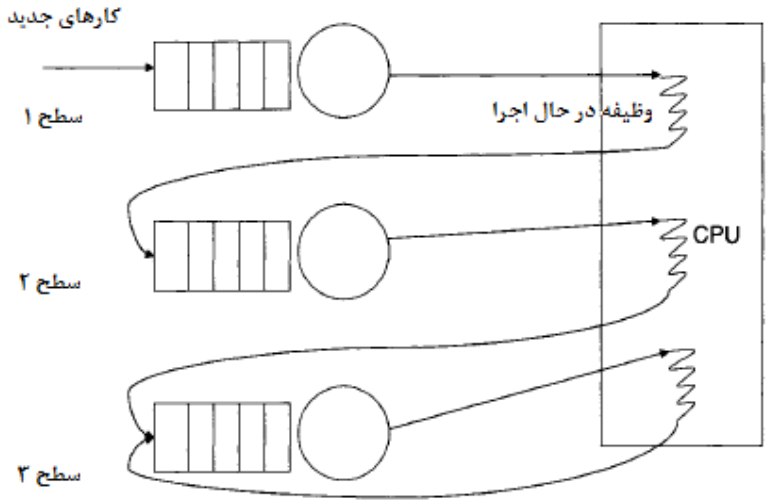
حالت خاتمه یا مرده حالتی است که تمام فرایندها از آن نشأت می‌گیرند و نهایتاً برای سیستم موجود به آن برمی‌گردند. این حالت درجایی است که اصولاً دارایی‌های اساسی، همچون یک بلوک کنترل فرایند، فضای بار حافظه اولیه و امثالهم به یک فرایند داده شود. افزون بر این، این حالتی است که در آن فرایندهایی که به هر دلیلی خاتمه یافته‌اند برگردانده می‌شوند. کارکردها در اینجا منابع حفظ‌شده را آزاد می‌کنند و فرایند را از سیستم حذف می‌نمایند.

فرایندها بر مبنای اقدامات روال‌های مختلف پشتیبانی سیستم‌عامل، همچون زمان‌بندی، توزیع‌کننده و روال‌های تخصیص، از حالتی به حالت دیگر نقل‌مکان می‌کنند. کار این روال‌ها تعیین این است که چه زمانی یک فرایند را از یک حالت به حالت دیگر انتقال دهند، چه فرایندی از یک حالت به حالت دیگر انتقال می‌یابد، فرایند چگونه انتقال یابد، و به کجا انتقال یابد. تمام این تصمیمات مبتنی بر تفسیر بلوک کنترل فرایند مدیریت‌شده سیستم‌عامل و حالت منابع سیستم هستند.



شکل (۲-۲۰): جریان فرایند در یک زمان‌بند نوبت گردشی

تعیین اینکه کدام‌یک از مجموعه‌های آماده از حالت آماده به حالت اجرا انتقال یابند نیاز به یک خط‌مشی زمان‌بندی و مکانیسم پشتیبانی برای پیاده‌سازی این خط‌مشی دارد. در ابتدا، سیستم‌های کامپیوتری از زمان‌بندی ساده FIFO استفاده می‌کردند، که در آن فرایند بعدی در یک لیست (صف، لیست ارتباط دهی شده، یا یک ساختار دیگر از داده‌های PCBها) فرایند زمان‌بندی‌شده برای انتقال از حالت آماده به حالت اجرا است. دیگر تکنیک‌های زمان‌بندی تلاش می‌کنند تا عادلانه‌تر باشند و فرایندهای اجرائی را به دسته‌های زمانی تقسیم می‌کنند که کوانتوم‌ها نامیده می‌شوند. یکی از چنان زمان‌بندهایی تکنیک نوبت گردشی است، که در آن فرایندها، وقتی‌که از کوانتوم زمانی تخصیص داده‌شده‌شان (یک برش یا دوره زمانی) فراتر رفتند، از حالت اجرا به حالت‌های مسدود شد یا معلق انتقال می‌یابند. فرایندهای تعلیق شده بر روی صف مدور قرار داده می‌شوند، و در آنجا منتظر می‌مانند تا زمانی که به جلوی صف نقل‌مکان کنند تا یک‌بار دیگر سرویس را دریافت نمایند. بدین طریق، زمان CPU به‌طور برابر در میان تمام فرایندهای فعال به اشتراک گذاشته می‌شود (شکل ۲-۲۰). این نوع از زمان‌بندی برای یک سیستم اشتراک زمانی رایج است.



شکل (۲-۲۱): زمان‌بندی برش زمانی چند سطحی

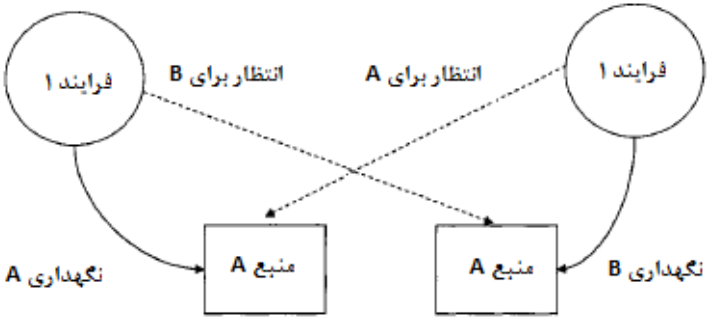
تکنیک‌های دیگری وجود دارند که در آن‌ها زمان کوانتوم برابر نیست و در آن‌ها فرایند انتخاب صرفاً مورد بعدی در صف را انتخاب نمی‌کند. برش‌های زمانی به سطوح متغیری تقسیم‌بندی می‌شوند که در آن‌ها سطح بالایی برش‌های زمانی کوتاه و کوچک است، واسطه برش‌های طولانی‌تر ولی با یک‌زمان انتظار طولانی‌تر بین دستیابی به سرویس است، و نهایتاً ، یک زمان‌بند بلندمدت، که در آن‌یک برش زمانی بزرگ تخصیص داده می‌شود ولی درجایی که زمان بین بازه‌های سرویس حتی بزرگ‌تر است (شکل ۲-۲۱).

انواع دیگری از زمان‌بندها برای تقریباً تمام کمیت‌های سیستم سنجش پذیر قابل‌درک ساخته‌شده است. به‌عنوان‌مثال، زمان‌بندهایی ساخته‌شده‌اند که از اولویت (از چند سطح تا هزاران سطح)، زمان اجرای باقیمانده، زمان‌بندی زمان سررسید ثابت، سقف اولویت، و تکنیک‌های دیگر برای انتخاب اینکه چه فرایندی بعد سرویس‌دهی خواهد شد استفاده می‌کنند.

وقتی یک فرایند برای سرویس زمان‌بندی شد، همچنان باید از حالت بلوک کنترل فرایند غیرفعال به حالتی انتقال یابد که در آن برای اجرا بر روی سخت‌افزار آماده‌سازی شده است. وظیفه آماده‌سازی فرایند برای اجرای واقعی بر عهده توزیع‌کننده سیستم‌عامل است. این توزیع‌کننده بلوک کنترل فرایند

را از زمان‌بند می‌پذیرد و به اجرای وظایفی ادامه می‌دهد که برای آماده‌سازی CPU برای اجرای فرایند فراهم‌سازی شده لازم هستند. توزیع‌کننده مقادیر ثبات CPU ذخیره‌شده را برای فرایند در ثبات‌های مناسب بارگیری می‌کند و ثبات‌های وضعیت CPU را بازیابی می‌کند. شمارنده برنامه ذخیره‌شده برای این فرایند در ثبات شمارنده برنامه CPU ذخیره شد، و اطلاعات آدرس‌دهی فیزیکی مناسب برای این فرایند در ثبات‌های آدرس‌دهی حافظه و ساختارهای داده‌های مناسب بارگیری می‌شود. وقتی تمام پارامترها در جای خود موجود باشند، توزیع‌کننده، با تبدیل کردن شمارنده برنامه برای فرایند به آدرس جهش بعدی که دستورالعمل بعدی از آن کسب می‌شود، کنترل را به فرایند انتقال می‌دهد. توزیع‌کننده می‌تواند وظیفه ریست کردن تایمر وقفه دادن به پرچم‌ها را پیش از انتقال کنترل اجرای CPU بر عهده داشته باشد. اگر قرار بر آن باشد که سیستم‌عامل مجدداً کنترل CPU را بعداً بر عهده بگیرد، تنظیم تایمرهای وقفه ضروری است.

یک کارکرد دیگر سیستم‌عامل که مسئول انتقال فرایندها از یک حالت به حالت دیگر است سرویس تخصیص حافظه است. این در بخش‌های بعدی این فصل با جزئیات بیشتری موردبحث قرار خواهد گرفت. ویژگی‌های دیگری که سیستم‌عامل باید برای مدیریت فرایند فراهم بیاورد دربرگیرنده مدیریت خطا و تشخیص بن‌بست می‌شوند، که هردوی آن‌ها برای یک سیستم مدیریت پایگاه داده نیز مهم هستند ولی نه در شکل مورداستفاده در یک سیستم‌عامل. سرویس‌های مدیریت خطا، بسته به کلاس سرویس موردنیاز و قیمتی که سیستم‌عامل و برنامه‌های کاربردی که سرویس دریافت می‌کنند تمایل به پرداختش داشته باشند، کارکردهایی را برای شناسایی، اصلاح، اجتناب، و جلوگیری از خطاها فراهم می‌آورند.



شکل (۲-۲۲): یک بن‌بست

تشخیص بن‌بست برای فرایندها و برای منابع موردنیاز فرایندهایی به اجرا درمی‌آید که در سیستم اجرا می‌شوند. بن‌بست وقتی رخ می‌دهد که یک فرایند دارای منبعی باشد که منبع دیگر الزامی نموده است و منبعی که این فرایند به آن نیاز دارد تحت نگهداری دیگری است (شکل ۲-۲۲۲). مدیریت بن‌بست می‌تواند شکل‌های بسیاری به خود بگیرد. ما ممکن است مایل به شناسایی بن‌بست و اصلاح آن از طریق حذف برخی از متخلفان باشیم. ما ممکن است مایل به جلوگیری از وقوع بن‌بست از طریق تضمین زودهنگام این باشیم که تخصیص منابع درخواست شده نمی‌تواند منجر به یک بن‌بست شود. یک‌راه تحقق این پیش تخصیص تمام منابع موردنیاز برای یک فرایند اجرائی پیش از فراهم آمدن امکان آغازش است. این‌یک الگوریتم ایمن است ولی الگوریتم که دارای مقدار بالایی از زمان نگهداری داخلی بر روی منابع است و الگوریتم که مستقیماً منتج به افزایش مدت‌زمان انتظار توسط فرایندها می‌شود، که منجر به زمان‌های اجرای کلی طولانی‌تر و کاهش بازدهی فرایند سیستم شود. یک ابزار دیگر مدیریت بن‌بست اجتناب کامل از بن‌بست است. اجتناب را می‌توان با تنظیم منابع در یک‌مرتبه ویژه دسترسی حاصل نمود، که باید تمام فرایندها از آن تبعیت نمایند. بدین طریق، فرایندها فقط می‌توانند به‌صورت مرتب به منابع دسترسی بیابند و نمی‌توانند منبعی را در اختیار داشته باشند که در اختیار دیگری است که شما منتظرش هستید. انتظار چرخشی در این رویکرد حذف می‌شود.

مدیریت منابع

مدیریت منابع الزام می‌کند که سیستم‌عامل دسترسی و انتقال اطلاعات از منابع مرتبط به کامپیوتر را هماهنگ‌سازی کند. برخی از کارکردهای معمول اجرائی توسط کارکرد مدیریتی منبع سیستم‌عامل عبارت‌اند از مدیریت حافظه، مقداردهی اولیه به دستگاه جانبی، تنظیمات دستگاه، کنترل بر انتقال داده‌ها، و دوز بندی دستگاه جانبی هستند. در سیستم‌های اولیه، سیستم‌عامل این دستگاه‌ها را تا یک سطح پایین کنترل می‌کرد. در سیستم‌های مدرن‌تر، سیستم‌عامل پارامترهای یک انتقال را تعیین می‌کند و جزئیات انتقال داده‌ها را بر عهده کنترلرهای دستگاه می‌گذارد و دستگاه‌های کنترل انتقال حافظه را جهت‌دهی می‌نماید. این باعث می‌شود که سیستم‌عامل و CPU آزادانه بتوانند دیگر وظایف ضروری مدیریت منبع را انجام دهند.

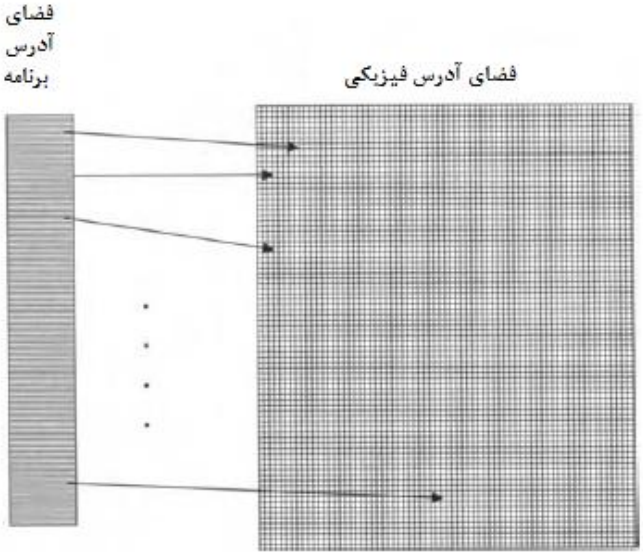
مدیریت حافظه

یک مدیر ذخیره‌سازی سیستم‌عامل سلسله‌مراتب حافظه کامپیوتر را مدیریت می‌کند. سیستم‌عامل، به‌طور خاص، باید انتقال اطلاعات به درون و بیرون از حافظه اولیه و این نگهداری فضای آزاد حافظه را انجام دهد. به‌منظور اجرای این کارکردها، یک سیستم‌عامل معمولاً از طرحی استفاده می‌کند که در آن حافظه اولیه به قطعاتی با اندازه ثابت به نام صفحات یا قطعاتی با اندازه متغیر به نام بخش‌ها[۳۴] شکسته می‌شود. سیستم‌عامل سپس انتقال صفحات یا بخش‌ها در حافظه را بر مبنای خط‌مشی‌های مورداستفاده مدیریت می‌کند. مدیر حافظه باید فضا را برای فرایندها تخصیص دهد، وقتی یک فرایند کامل شد فضا را آزاد کند، وقتی حافظه به خاطر تخصیص و آزادسازی نسبت‌های نابرابر چندتکه شد، به‌طور دوره‌ای فضای حافظه را پاک‌سازی نماید. مسئله تخصیص حافظه ارتباطی مستقیم با نقشه حافظه دارد. (شکل (۲-۲۳) را ببینید.)

نقشه حافظه نشانگر آن است که چه بخش‌هایی در حافظه به یک فرایند تخصیص داده می‌شوند و چه بخش‌هایی آزادند تا به فرایند جدید تخصیص داده شوند. این نقشه حافظه را می‌توان برای کمک به مدیریت تخصیص به طرق مختلفی مدیریتی نمود. لیست بخش‌های آزاد را می‌توان در یک لیست آزاد سازمان‌دهی نمود، که در آن بلوک‌ها به‌صورت درختی با اندازه بلوک افزایشی، یا به‌صورت یک پشته، ساختاربندی می‌شوند، که در آن بزرگ‌ترین بلوک همیشه به سمت بالای پشته است. ازاین‌رو تخصیص حافظه به تابعی از انتخاب بلوکی با اندازه مناسب بر مبنای خط‌مشی انتخابی ارائه‌شده تبدیل می‌شود. برخی از خط‌مشی‌ها شامل تطابق اولیه می‌شوند، که در آن اولین بلوکی که منطبق با این فرایند تشخیص داده می‌شود انتخاب می‌شود. یک خط‌مشی دیگر بهترین تطابق است، که در آن‌ها بلوک‌ها تا زمانی اسکن می‌شوند که یک مورد پیدا شود که بهترین تطابق را با اندازه فرایندی که باید در حافظه بارگیری شود داشته باشد. طرح‌های متعدد دیگری وجود دارند، ولی آن‌ها برای حوزه عمل این فصل هستند.

---

[۳۴] segments



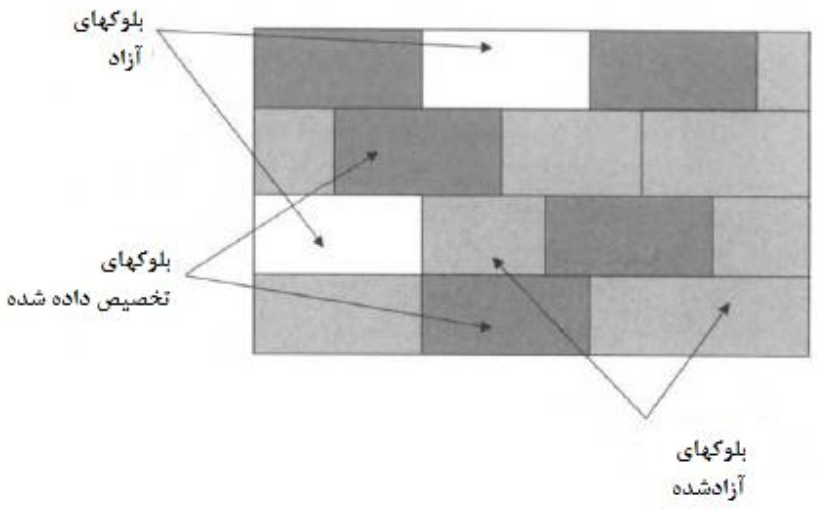
شکل (۲-۲۳): نقشه حافظه

تخصیص حافظه هم‌راستا با آزادسازی حافظه پیش می‌رود. وقتی صفحات یا بخش‌ها توسط فرایندهایی آزادسازی می‌شوند که از حالت اجرا خارج می‌شوند، آن‌ها باید از لیست تخصیص داده‌شده خارج‌شده و در لیست آزاد صفحات یا بخش‌ها جایگزین شوند. این بخش‌های آزادشده در بلوکی معادل با اندازه فرایند تخصیص داده‌شده بازیابی می‌شوند که آن‌ها را نگهداری می‌کند. این بخش‌های آزاد سپس در لیست آزاد در مکانی مناسب با اندازه بخش‌های آزادی قرار داده می‌شوند که بازیابی می‌شوند.

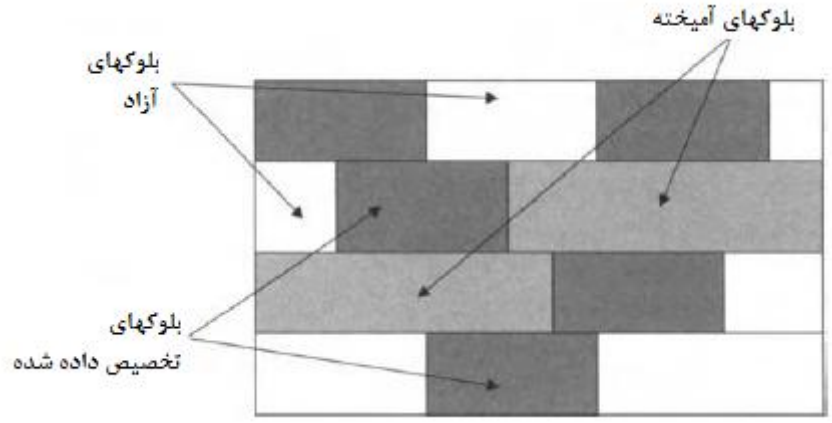
بااین‌حال، تمام جایگزینی‌ها به چنان شکل خوبی بر روی مرزهای اجرای فرایند انجام نمی‌شوند. بیشتر آن‌ها بر روی یک حافظه اولیه پر یا تقریباً پر انجام می‌شوند. برای اینکه این امکان فراهم شود که فرایندها در اجرایشان به جلو حرکت کنند، ما باید صفحات فعال را توسط خط‌مشی مرتب‌سازی مجدد نماییم که برایمان امکان حذف برخی از صفحات فعال را فراهم خواهد آورد و اجازه دهیم که آن‌ها به دیگر فرایندهای نیازمند یا سنگین باز تخصیص شوند. رایج‌ترین الگوریتم جایگزینی صفحه و خط‌مشی آزادسازی مبتنی بر اصل کمترین استفاده اخیر[۳۵] (LRU) است. این اصل نشانگر آن است که جدیدترین صفحه‌ای که استفاده‌شده است بیشترین احتمال را دارد که در آینده قابل پیش‌بینی به همان

[۳۵] least recently used

صورت بماند و، ازاین‌رو، یک نامزد اولیه برای حذف و جایگزینی توسط یک فرایند انتظار است. دیگر طرح‌های مورداستفاده برای تعویض صفحه شامل جدیدترین مورداستفاده شده، موردی که اخیراً کمترین استفاده را داشته است، و حذف تصادفی هستند. تمام این خط‌مشی‌ها درگذشته به‌طور مفصل موردبررسی قرارگرفته‌اند و مزایایی برای فعالیت‌های پردازشی مشخص دارند، اگرچه برای سیستم‌های پایگاه داده برخی از این‌ها کاملاً فاجعه‌بار هستند. فرایند پایگاه داده به شکلی عمل می‌کند که در بیشتر برنامه‌های کاربردی رواج ندارد و، درنتیجه، برای یک خط‌مشی مشخص واکنشی یکسان نخواهد داشت.



شکل (۲-۲۴): حافظه قطعه‌قطعه



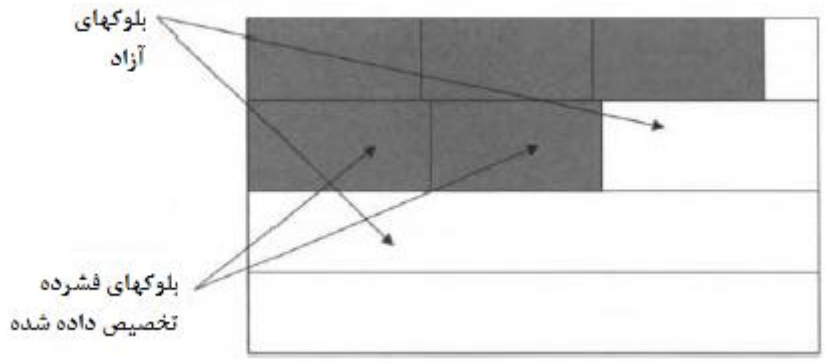
شکل (۲-۲۵): علامت‌گذاری بلوک‌های آزاد در حافظه

یک کار دیگر برای مدیریت حافظه حفظ یک نقشه بخش‌های آزاد حافظه و پاک‌سازی دوره‌ای حافظه برای پاک‌سازی قطعه‌های پیوسته بزرگ‌تر است تا تخصیص آسان‌تر شود. این فرایند با عنوان جمع‌آوری زباله و باز تخصیص شناخته می‌شود. خط‌مشی‌های تخصیص و باز تخصیصی که قبلاً موردبحث قرار گرفتند سبب می‌شوند که حافظه به‌طور دوره‌ای قطعه‌قطعه شود. وقتی حافظه به قطعه‌های بسیار ریز قطعه‌قطعه شد، ممکن است یافتن بلوک‌های پیوسته حافظه آزاد برای تخصیص به فرایندهای ورودی ناممکن شود (شکل ۲-۲۴). به‌منظور رفع این مشکل، سرویس‌های مدیریت حافظه به‌طور دوره‌ای نقشه حافظه را چک می‌کنند تا مشخص شود که آیا پاک‌سازی بلوک‌های آزاد قطعه‌قطعه شده به بخش‌های بزرگ‌تر منجر به افزایش قابل‌توجهی در بلوک‌های پیوسته آزاد با اندازه کافی خواهد شد.

یک تکنیک تمام بلوک‌های آزاد را اسکن می‌کند و حفره‌های مجاور را با بخش‌های آزاد بزرگ‌تر علامت‌گذاری شده ادغام می‌کند. این‌ها سپس به لیست آزاد با حفره‌های گسسته درآمیخته حذف‌شده از لیست آزاد افزوده می‌شوند (شکل ۲-۲۵).

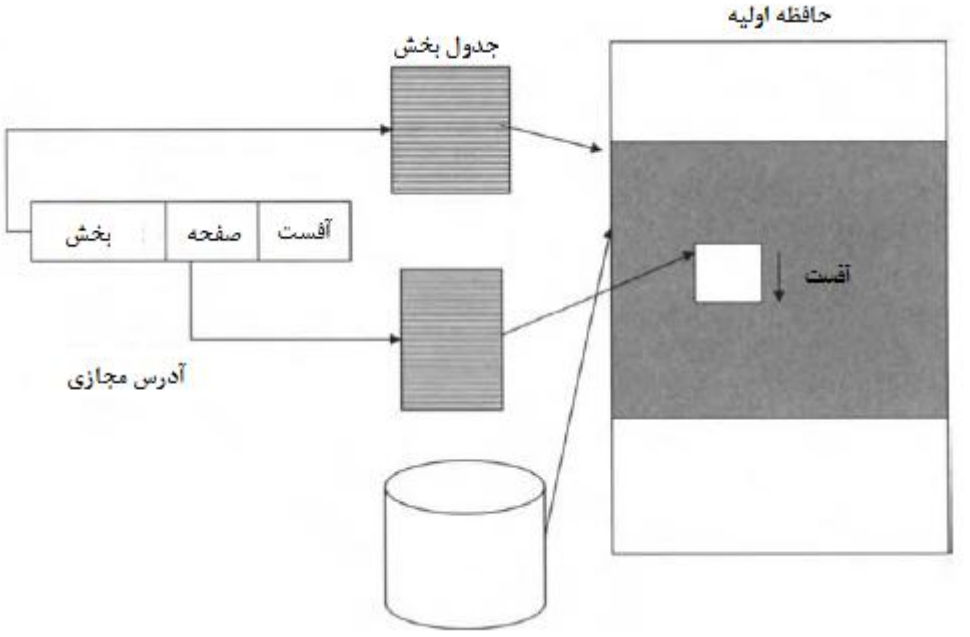
این به‌خودی‌خود نمی‌تواند منجر به فضای آزاد با اندازه کافی شود. برای دستیابی به بلوک‌های آزاد بزرگ‌تر، ممکن است اسکن دوره‌ای کل حافظه و آزادسازی درجایی که فرایندها ذخیره‌شده‌اند ضروری باشد تا نقشه تخصیص حافظه به دو بخش تمیزکاری شود – یکی شامل یک بخش پیوسته متشکل از تمام بلوک‌های حافظه تخصیص داده‌شده و دیگری تمام بلوک‌های آزاد حافظه می‌شود.

فرایندی که از طریق آن تمام بلوک‌های تخصیص داده‌شده به یکسر حافظه انتقال داده‌شده و باز تخصیص می‌شوند فشرده‌سازی نامیده می‌شود، و فرایند لازم برای باز تخصیص تمام فضای تازه تخصیص داده‌شده به لیست آزاد با عنوان جمع‌آوری زباله شناخته می‌شود. فرایندی که از طریق آن تمام بلوک‌های تخصیص داده‌شده انتقال داده‌شده به یکسر حافظه انتقال داده می‌شوند و باز تخصیص می‌شوند فشرده‌سازی نامیده می‌شود، و فرایند لازم برای باز تخصیص تمام فضای تازه آزادشده به لیست آزاد با عنوان جمع‌آوری زباله مورد ارجاع قرار می‌گیرد (شکل ۲-۲۶). همانند یک ماشین جمع‌آوری زباله، این فشرده‌سازی تلاش می‌کند تا محتوا را در یکسر محفظه فشرده‌سازی نماید، و مابقی فضا را برای زباله بیشتر آزاد نماید. این فرایند باز تخصیص و انتقال فرایندها و آدرس‌هایشان را الزامی می‌سازد (تمام ارجاعات باید در بخش‌های بار فیزیکی و PCB تغییر یابند).



شکل (۲-۲۶): حافظه پس از جمع‌آوری زباله

برای این طرح‌های پایه مدیریت حافظه، برخی از سیستم‌های عامل، در کنار سخت‌افزار و نرم‌افزار پشتیبان، هم از صفحه‌بندی و هم از بخش‌بندی پشتیبانی می‌کنند. در این طرح، حافظه به چند بخش تقسیم می‌شود. یک بخش دارای چند صفحه است، و یک صفحه اندازه‌ای ثابت دارد. این بخش‌ها به‌صورت صفحات در طرح اول در داخل و خارج از حافظه نگاشت می‌شوند (شکل ۲-۲۷ را ببینید).



شکل (۲-۲۷): حافظه هم با صفحه‌بندی و هم با بخش‌بندی

مدیریت فایل

مدیریت فایل تابعی از سیستم‌عامل است که ساختار و ذخیره‌سازی اطلاعات بر روی منابع ذخیره-سازی غیر اولیه را کنترل می‌کند. یک کاربرد معمول، مدیریت فایل‌های ذخیره‌شده بر روی دیسک درایو یا درایو نواری است. فایل‌ها مجموعه‌ای از داده‌ها و/یا برنامه‌ها هستند که به حافظه انتقال داده‌شده یا از آن خارج می‌شوند. اجرای این حرکت نیازمند آن است که ساختار، فرمت و مکان فایل برای سیستم‌عامل معلوم باشد. مدیریت فایل از این اطلاعات برای درخواست فضای حافظه از مدیر حافظه برای انتقال یک فایل از مخزن به حافظه استفاده می‌کند. وقتی فایل سیستم آماده بازگشت به حافظه باشد، از اطلاعات مدیر حافظه برای تعیین این استفاده می‌کند که آیا تغییری در فایل انجام‌شده است یا نه. اگر تغییری انجام‌نشده باشد، فایل دور ریخته می‌شود. اگر تغییراتی انجام‌شده باشد، مدیر فایل باید تعیین کند که آیا به فضایی بیشتر ازآنچه فایل در ابتدا

اشغال نموده بوده نیاز هست یا نه. اگر جواب مثبت باشد، فایل می‌تواند در یک مکان متفاوت ذخیره شود یا نیاز به قطعه‌قطعه سازی بر روی دستگاه دارد. مشابه با مدیر حافظه، مدیر فایل ممکن است به‌طور دوره‌ای نیازمند باز تخصیص فضای ذخیره‌سازی و انتقال فایل‌ها به‌منظور آزادسازی بخش‌های بزرگ‌تر پیوسته باشد.

مدیر فایل سرویس‌های اضافی را در اختیار برنامه‌های کاربردی قرار می‌دهد. مدیریت فایل کارکردهایی برای ایجاد، حذف و قرار دهی اطلاعات در فایل‌ها فراهم می‌آورد. مکانیسم‌های کنترل فایل از اشتراک‌گذاری فایل‌ها میان کاربران به‌منظور کنترل شکل دسترسی مجاز، برای ساختاربندی فایل‌ها برای استفاده بهینه از زمان و فضا، نام‌گذاری یا تغییر نام فایل‌ها، و کپی کردن و نسخه‌برداری فایل‌ها به‌صورت موردنیاز برای پشتیبانی از سیستم پشتیبانی می‌کنند.

مدیریت مکان و محتوای یک فایل سیستم از طریق استفاده از یک سرویس فایل دایرکتوری کنترل می‌شود. یک فایل دایرکتوری را می‌توان به‌عنوان ابزاری برای تسهیل دسترسی به فایل‌ها و محدودسازی استفاده از یک فایل به‌صورت مشخص‌شده توسط مالک یا سیستم‌عامل استفاده نمود. برخی از مدیریت کننده‌های فایل، فایل‌ها را بر اساس نوعشان سازمان‌دهی می‌کنند، مثلاً EXE. برای فایل‌های قابل‌اجرا، TXT. برای فایل‌های متنی، FOR. برای فایل‌های FORTRAN، PAS. برای فایل‌های پاسکال، و C. برای فایل‌های C. برای کمک به مدیریت فایل‌ها، مدیر فایل یک بلوک کنترل فایل را با اطلاعاتی درباره فایل‌های تحت کنترلش حفظ می‌کند. این اطلاعات می‌توانند نگهداری و استفاده از فایل‌ها را تسهیل نمایند.

حفاظت

حفاظت یک تابع سیستم‌عامل است که دسترسی به منابع کنترل‌شده را مدیریت می‌کند. حفاظت نوعاً متشکل از اجازه دسترسی[۳۶]، احراز هویت دسترسی[۳۷] و محدودیت دسترسی. سیستم‌عامل حقوق احراز هویت یک درخواست‌کننده سرویس را پیش از اجرای سرویس چک می‌کند. اگر

[۳۶] access authorization
[۳۷] access authentication

حقوق مناسبی موجود باشند، اجازه دسترسی داده می‌شود، در غیر این صورت، دسترسی درخواست‌کننده مسدود می‌شود.

احراز هویت دسترسی فرایندی است که از طریق آن سیستم‌عامل تعیین می‌کند که یک فرایند حق اجرا بر روی این سیستم را دارد. رایج‌ترین شکل این کنترل نام کاربری است، که همه ما با آن در زمان لاگین کردن در یک کامپیوتر آشنایی داریم. شکل دوم حفاظت سیستم‌عامل احراز هویت است. احراز هویت با مسئله راستی آزمایی هویت یک فرد سروکار دارد. رایج‌ترین شکل احراز هویت گذرواژه است. مشخص‌شده است که ترکیب احراز هویت کاربر از طریق مجوز دهی یک نام کاربری و احراز هویت کاربر از طریق یک گذرواژه برای بیشتر موارد غیر حساس مدیریتی محدودسازی دسترسی به سیستم‌های کامپیوتری کفایت می‌کند. در صورت لزوم، این دو روش را می‌توان برای دسترسی به هر منبعی به‌منظور محدودسازی دسترسی به آن مورداستفاده قرارداد. مسئله‌ای که باید برای آن چاره‌جویی شود درجه حفاظت موردنیاز و مقدار بالاسری است که ما برای پرداختن برای آن تمایل داریم.

کنترل دسترسی یک مسئله دخیل‌تر است و با نحوه کنترل استفاده از اطلاعات و برنامه‌ها توسط کاربری که دارای مجوز ورود به سیستم رادارند سروکار دارد. برای کنترل اینکه چه کسی از نرم‌افزار استفاده نماید و نحوه استفاده‌اش چگونه باشد، یک سیستم‌عامل باید مکانیسم‌هایی برای محدودسازی حقوق اجرای نرم‌افزار کنترل‌شده فراهم بیاورد. برای این کار، سیستم‌های عامل از شکلی از کنترل دسترسی استفاده می‌کنند. رایج‌ترین آن‌ها لیست‌های کنترل دسترسی، ماتریس‌های کنترل دسترسی، و قابلیت‌ها هستند. لیست‌های کنترل دسترسی ابزاری برای لیست کردن تمام عناصر نرم‌افزاری که باید در سیستم کنترل شوند فراهم می‌آورند و لیستی از کاربران یا فرایندها فراهم می‌آورند که حق استفاده از این عناصر نرم‌افزاری رادارند. این کنترل این می‌تواند نوع حقوق اجرائی که فرایند یا کاربر می‌توانند داشته باشند را نیز محدود نماید. به‌عنوان‌مثال، ما تنها می‌توانیم اجازه فراخوانی یک فرایند را داشته باشیم، نه آزادسازی CPU برای فرایند فراخوانی را. ما تنها می‌توانیم اجازه فقط اجازه دسترسی خواندن را به یک منطقه یک فرایند نرم‌افزاری یا قرار دهی حقوق بدهیم، یا می‌توانیم حقوق نامحدودی بدهیم. مکانیسم اصلی (ترکیب مشخصه یک کاربر در قبال لیستی از حقوق) برای یک لیست کنترل دسترسی در یک مکان متمرکز انجام می‌شود، که احتمالاً درون یک سرویس سیستم‌عامل مجزا یا درون خود نرم‌افزار کنترل‌شده است. قابلیت‌ها عملکرد مشابهی دارند ولی این کار را به شکلی توزیع‌شده انجام

می‌دهند. قابلیت‌ها برای هر عنصر کنترل‌شده ایجاد می‌شوند و توسط فرایندهایی درخواست می‌شوند که به استفاده از عنصر کنترل‌شده تمایل دارند. اگر قابلیت برای یک فرایند مناسب باشد، به فرایند اعطا می‌شود. این فرایند سپس می‌تواند از این قابلیت مانند یک بلیت برای دسترسی و استفاده از عنصر کنترل‌شده استفاده نماید.

مدیریت دستگاه جانبی

سرویس‌های مدیریت دستگاه جانبی ورودی/خروجی برای حذف جزئیات فیزیکی استفاده از فرایندهای کاربری و فراهم‌سازی امکان مدیریت یکپارچه‌تر و عادلانه‌تر منابع ایجاد می‌شده‌اند. هدف سرویس‌های مدیریت دستگاه‌های جانبی روشن، تمیز و شفاف کردن دسترسی برای کاربران است. مدیریت باید تمام وابستگی‌های فیزیکی را از الزامات دسترسی کاربران حذف نموده و این‌ها را با مکانیسم‌های منطقی رایج در محیط‌های برنامه‌نویسی جایگزین کند. این کنترل برای مستقل نمودن دستگاه دسترسی است. کاربر نباید بداند که چه نوع از دستگاه یا در کجا برای دسترسی به داده‌ها و نرم‌افزار سرویس قرار دارد.

مدیریت برای دستگاه‌های پیرامونی محدود به دو کلاس روال‌های سرویس سیستم‌های عامل می‌شود: مدیران دستگاه و ۱۱۰. سیستم‌عامل نهایت تلاش را برای یکسان به نظر رسیدن تمام دسترسی‌ها انجام می‌دهد. روش معمول این است که تمام دسترسی‌ها نما و ظاهر یک دسترسی فایل را داشته باشند. کارکرد فرایند مدیریت ۱۱۰ تنظیم و حفظ مسیرها یا کانال‌های منطقی بین فرایندهای مقیم در حافظه CPU و دنیای خارج است. کارکردهای فراهم‌سازی شده توسط این عنصر شامل تخصیص و آزادسازی کانال می‌شوند. تشخیص و اصلاح خطا بر روی کانال می‌تواند بدین طریق جای‌داده شود. کارکرد مدیریت دستگاه با این کارکرد هم‌راستا است. سرویس‌های مدیریت دستگاه مکانیسم‌هایی برای اجرای تنظیمات، تخصیص، کنترل، همگام‌سازی، آزادسازی و انتقال داده‌های وابسته به دستگاه در اختیار قرار می‌دهند.

مدیریت دستگاه و ۱۱۰ پیوند فیزیکی را ایجاد نموده و انتقال را کنترل می‌کنند. در این کارکرد درخواست برای دارایی‌های میانجی برای کانال به‌منظور به‌کارگیری در انتقال اطلاعات از مخزن ثانویه به حافظه داخلی کامپیوتر گنجانده‌شده است. این بافرها به‌عنوان واسطه‌ای میان دستگاه‌ها و CPU

استفاده می‌شوند. آن‌ها امکان عملکرد همزمان ۱۱۰ با پردازش برنامه‌های کاربردی درون سیستم را فراهم می‌آورند. کنترل کانال ۱۱۰ و کنترل دستگاه نوعاً در یک سیستم‌عامل به‌عنوان یک فرایند مستقل هندل می‌شوند. سیستم‌عامل عملیات دستگاه یا ۱۱۰ را مقداردهی اولیه می‌کند و خارج می‌شود، و بدین طریق به مدیران ۱۱۰ و دستگاه اجازه اجرای وظیفه را می‌دهد و، پس از تکمیل شدن، سیستم‌عامل را وقفه می‌دهد تا نشانگر تکمیل وظیفه باشد. وقفه می‌تواند فعال باشد، که در آن سیستم‌عامل را برای سرویس بلافاصله متوقف می‌کند، یا می‌تواند پیام محور باشد، که در آن‌یک شاخص وضعیت را تعیین می‌نماید، که سیستم‌عامل در زمان فراغت آن را چک می‌کند.

این روال‌ها، وقتی با مدیر فایل سیستم‌عامل تلفیق شوند، یک رابطه یکپارچه را بین برنامه‌های ذخیره‌شده، داده‌ها و سیستم زمان اجرا شکل می‌دهند. مدیر فایل برای دسترسی مستقیم عناصر ذخیره‌سازی منطقی توسط سیستم‌عامل و فرایندهای کنترل‌شده استفاده می‌شود. این مدیر فایل سرویس‌ها را در اختیار فایل نام‌ها[۳۸] قرار می‌دهد، فایل‌ها را آدرس‌دهی می‌کند، دسترسی را کنترل می‌کند، مسیرهای دسترسی را انتخاب و هماهنگ‌سازی می‌کند، کپی‌سازی و پشتیبان گیری پس‌زمینه را برای بازیابی انجام می‌دهد، تخصیص و آزادسازی منابع را درجایی که اطلاعات فایل واقع است هماهنگ‌سازی می‌کند، و قرارگیری (منطقی) اطلاعات ذخیره‌شده را مدیریت می‌کند. یک کارکرد مهم سیستم مدیریت فایل مدیریت قفل است. مدیران فایل لاگینگ و آنلاگینگ فایل‌ها و رکوردهای درون فایل‌ها را ایجاد نموده، منتشر نموده و کنترل می‌کنند. این سرویس فوق‌العاده برای کنترل همزمانی مهم است.

۲-۹-۲ نرم‌افزار کنترل شبکه

نرم‌افزار مدیریت شبکه ارسال و دریافت اطلاعات بر روی یک واسطه ارتباطی را مدیریت می‌کند. کارکردهای معمول عبارت‌اند از روتینگ (مسیریابی) پیام، نام‌گذاری، آدرس‌دهی، حفاظت، دسترسی، تشخیص و اصلاح خطا، تنظیم ارتباطات، و مدیریت.

روتینگ یک کارکرد مدیریت شبکه سیستم موردنیاز برای هماهنگ‌سازی انتقال اطلاعات بر روی یک یا چند شبکه است. در یک شبکه محلی این کارکرد در تمام موارد موردنیاز نیست. روتینگ می‌تواند صرفاً نیازمند ارسال داده‌ها در یک‌جهت مشخص بر روی واسطه باشد، یا ممکن

[۳۸] name files

است نیاز به خط‌مشی‌های پیچیده‌تری برای انتخاب یک کانال یا سیستم برای ارسال پیام، بر مبنای مکان فرستنده و ترافیک شبکه، داشته باشد. روتینگ یک کارکرد موردنیاز در شبکه‌های گسترده‌ای همچون اینترنت است.

نام‌گذاری برای تسهیل دسترسی شفاف به تمام منابع سیستم، محلی یا دوردست، لازم است. یک طرح نام‌گذاری باید ویژگی‌های ذیل را دارا باشد: امکان اشتراک‌گذاری اشیاء را فراهم بیاورد، امکان دسترسی به موارد بازتولیدی را فراهم بیاورد، و امکان دسترسی کاملاً شفاف را فراهم بیاورد. تابع نام‌گذاری باید دو نوع از نام‌ها را برای هر آیتم مدیریت‌شده پشتیبانی کند: یک نام داخلی (سیستم‌ها) و یک نام خارجی (کاربری). تابع نام‌گذاری باید ترجمه و مدیریت نام‌های خارجی را بانام‌های داخلی (منحصربه‌فرد) مدیریت نماید.

آدرس‌دهی ابزاری است که از طریق آن سیستم مکان یک آیتم نام‌گذاری شده را تعیین می‌کند. طرح‌های آدرس‌دهی را می‌توان به سلسله‌مراتب‌هایی تقسیم نمود، که در آن کامپیوترهای محلی مجموعه نام‌های خودشان رادارند، که ممکن است بین سیستم‌ها منحصربه‌فرد نباشد. ترکیب آدرس سیستم (یک گره روی شبکه) و نام محلی برای فراهم‌سازی یک نام منحصربه‌فرد شبکه کفایت می‌کند. به همین صورت، ما می‌توانیم یک نام و آدرس منحصربه‌فرد در مجموعه شبکه‌های به‌هم‌پیوسته برای هر شبکه داشته باشیم.

کنترل دسترسی بر روی یک شبکه با خط‌مشی‌ها و مکانیسم‌هایی برای محدودسازی حالت دسترسی ارائه‌شده به کاربران شبکه سروکار دارد. محدودیت‌های دسترسی باید به‌سادگی امتیاز لاگین یا پیچیده‌تر، همچون محدودسازی نوع ارتباطاتی که می‌توان کسب نمود یا نوع دسترسی به اطلاعات دوردست، باشند. مکانیسم محدودسازی دسترسی را می‌توان در نرم‌افزار دسترسی به شبکه گنجاند یا می‌تواند صراحتاً توسط کاربر نرم‌افزاری که به شبکه دسترسی می‌یابد گنجانده شود.

حفاظت تابعی از سیستم‌عامل است که با مدیریت منابع از دسترسی بدخواهانه یا تصادفی سروکار دارد که می‌تواند سیستم را قفل نماید. دو کلاس عمده از طرح‌های حفاظتی وجود دارند: اولی تلاش می‌کند تا با پیش تخصیص منابع از بروز مسئله جلوگیری نماید، دومی اجازه وقوع رخداد را می‌دهد ولی ابزارهایی برای شناسایی و اصلاح مسائل فراهم می‌آورد. اجتناب باعث ایجاد امکان دسترسی به منابع به شکلی متدولوژیک می‌شود. یک طرح نیاز فرایندی برای دستیابی زودهنگام به تمام منابع موردنیازش

دارد و آن‌ها را در طی مدت دسترسی حفظ می‌کند. این بسیار محدودکننده است و می‌تواند بروز تأخیرهای فوق‌العاده‌ای برای منابع دیگری شود که ممکن است به منابع نگهداری شده نیاز داشته باشند. تشخیص بن‌بست امکان دسترسی همزمان‌تر به منابع را فراهم می‌آورد ولی به قیمت بن‌بست‌های بالقوه. یک طرح نیاز به ساخت گراف‌های انتظار دارد، که امکان تشخیص بن‌بست‌های بالقوه واقعی را فراهم بیاورد و مکانیسم‌هایی برای از بین بردن بن‌بست از طریق از بین بردن فرایندهای متداخل فراهم می‌آورد. می‌توان از این توصیف ساده مسائل محتمل ازنظرگاه یک پایگاه داده را دید. سیستم‌عامل می‌تواند اشتراک‌گذاری منابع بین فرایندها را محدود نماید، حتی درصورتی‌که پایگاه داده اجازه آن را بدهد. نرم‌افزار دسترسی رسان‌های تراکنش کاربران و فرایندهای نرم‌افزاری با شبکه را کنترل می‌کند. مکانیسم‌های معمول با تراکنش لاگین و شناخت با یک گره شبکه تراکنش می‌کنند. نرم‌افزار دسترسی رسان‌های با ارتباط باواسطه ارتباطی و تنظیم نشست‌های ارتباطی سروکار دارد. دسترسی به یک فرایند امکان لاگین در شبکه و شناسایی شدن توسط دیگران بر روی شبکه را فراهم می‌آورد.

تنظیم و مدیریت ارتباطات در همراهی با نرم‌افزار دسترسی رسان‌های برای تراکنش با گره‌های دوردست و برقراری یک لینک عمل می‌کند. نوعاً، یک گره درخواست ارتباط[۳۹] با یک گره دوردست را ارائه می‌کند. اگر گره دوردست بتواند یک نشست اضافی را پشتیبانی نماید، یک بلوک کنترلی را برای نگهداری اطلاعات درباره تنظیمات[۴۰] ایجاد می‌نماید. گره درخواست‌کننده علامت می‌دهد که یک نشست با موفقیت ایجاد شده است. فرایندهای تراکنشی پس از ایجاد می‌توانند اطلاعات را با استفاده از پارامترهای از پیش تخصیص داده‌شده‌شان ارسال و دریافت کنند.

خط‌مشی‌ها و مکانیسم‌های کلاینت/سرور

حالت کلاینت/سرور دسترسی و کنترل منبع دوردست رواج دارد. به‌آسانی می‌توان در یک مجله تجاری تبلیغاتی برای سیستم‌های دارای پردازش کلاینت/سرور یافت. این تکنیک برخی از مزایای سیستم‌های توزیع‌شده را در اختیار قرار می‌دهد، ولی بدون بالاسری کنترل اضافی.

---

[۳۹] linkup
[۴۰] setup

شرکت‌کنندگان کلاینت/سرور با درخواست و دریافت سرویس‌ها در صورت نیاز فعالیت می‌کنند. کلاینت‌ها نیاز به نگهداری منابع دارند و می‌توانند سرویس را از سرور درخواست نمایند. سرور بر مبنای استفاده فعلی سرویس و خط‌مشی اشتراکی موجود در سرور را به کلاینت‌ها ارائه می‌کند. این روش، همگام‌سازی تنگاتنگی که ممکن است در سیستم‌های توزیع‌شده یافته شود ارائه نمی‌کند، ولی ابزاری ساده برای دسترسی و اشتراک منابع دوردست به شکلی یکنواخت فراهم می‌آورد. سادگی آن باعث افزایش رواج و رشد آن شده است.

خط‌مشی‌ها و مکانیسم‌های فراخوانی رویه دوردست

یک مکانیسم دسترسی دوردست مشابه، مکانیسم فراخوانی رویه دوردست است. همانند رویه‌های محلی، یک درخواست‌کننده باید نام رویه و پارامترهای مناسب را بداند. درخواست‌کننده رویه دوردست و بلوک‌های منتظر پاسخ رویه دوردست را فراخوانی می‌کند. رویه فراخوانی سرویس درخواست شده را به اجرا درمی‌آورد، و، در عوض کنترل با فراخوان کننده، فراخوان کننده پردازش را ادامه داده و آنلاک می‌کند. این رویه دقیقاً به همان صورت رویه متداول فراخوانی است، به‌جز اینکه این فراخوانی بر روی یک کانال دوردست تا یک سایت دیگر است. جزئیات بیشتر نرم‌افزار شبکه ویژگی‌های مرتبط با پایگاه داده‌ها در فصل‌های بعدی توضیح داده خواهند شد.

۲-۹-۳ شناسایی و بازیابی خطا

یک سیستم‌عامل دارای الزامی برای مانیتور سیستم از بابت خطاها و خرابی‌ها و فراهم‌سازی مکانیسم‌هایی برای اصلاح این شرایط یا پیکربندی مجدد حول آن‌ها است. به‌منظور شناسایی خطاها، در وهله اول از یک سیستم‌عامل برای چند کارکرد اساسی استفاده می‌کند. اولی بر تشخیص سخت‌افزاری خطاها متکی است – به‌عنوان‌مثال، بیت‌های بررسی توازن زوج[۴۱]، بررسی‌های افزونگی دوره‌ای[۴۲] و محاسبات محاسباتی همچون سرریزها و تقسیم بر صفر. این‌ها

[۴۱] parity check bits
[۴۲] cyclic redundancy

امکان شناسایی خطاهای متناوب یا سخت درون زیرساخت ارتباطی و محاسباتی ماشین را فراهم می‌آورند. بررسی خطاهای ظریف‌تر یا مخفی نیاز به شروع دوره‌ای نرم‌افزار مانیتورینگ خطا دارد. این نرم‌افزار اطلاعات را از این عناصر سخت‌افزاری اساسی و نرم‌افزار اجرائی با استفاده از نقاط تست از پیش تعریف‌شده فراهم می‌آورد. این داده‌های گردآوری‌شده سپس به‌طور دوره‌ای برای یافتن الگوهایی که می‌تواند نشانگر وجود خطاهای نرم‌افزاری یا سخت‌افزاری در سیستم باشد آنالیز می‌شوند. این نرم‌افزار با عنوان نرم‌افزار مانیتورینگ برنامه مورد ارجاع قرار می‌گیرد.

وقتی یک وضعیت خطا با استفاده از مکانیسم‌های مانیتورینگ خطا شناسایی شد، کار بعدی تعیین محل ورود خطا و سپس مجزا سازی خطا در یک دانه‌بندی نرم‌افزاری یا سخت‌افزاری از پیش تعیین‌شده است – به‌عنوان‌مثال، برای سخت‌افزار تا یک بورد قابل تعویض یا یک قطعه همچون یک مدار مجتمع، برای نرم‌افزار تا یک ماژول، فرایند، کارکرد یا احتمالاً یک بلوک یا خط کد، برای داده‌های درون فایل، تا سطح آیتم داده‌ها یا رکورد. سطح جداسازی ارائه‌شده بستگی به بالاسری و بهایی خواهد داشت که سیستم مایل به پرداخت به‌منظور شناسایی و جداسازی باشد. این مکانیسم معمولاً تشخیص محل خطا[۴۳] نامیده می‌شود. تشخیص محل خطا با استفاده از محرکه-های تست معلوم و پاسخ‌های معلوم برای بررسی عناصر سخت‌افزاری و نرم‌افزاری از بابت خروجی‌های خطادار عمل می‌کند. بااین‌حال، صرف شناسایی یک وضعیت خروجی خطادار و این فرض که این بخش خطادار است کفایت نمی‌کند. خطاها می‌توانند از طریق لایه‌های متعدد سخت‌افزاری و نرم‌افزاری انتشار یابند، که فقط در مراحل بعدی به نمایش درمی‌آید. هدف تشخیص محل خطا شناسایی یک خطا و سپس تست مجدد تمام عناصر تراکنشی برای جداسازی خطا از مقصر واقعی است.

در زمان تفکیک یک عنصر سخت‌افزاری یا نرم‌افزاری خطادار، سیستم‌عامل باید یک اقدام مناسب را برای رفع خطا از سیستم تعیین نماید. فرایند اجرای این کارکرد بازیابی و پیکربندی مجدد نامیده می‌شود. رایج‌ترین روش اجرا برخی از اقدامات بازیابی از اول است.

بازیابی می‌تواند به‌سادگی بارگیری مجدد و شروع مجدد یا صرفاً ریست کردن نرم‌افزاری باشد که از قبل بارگیری شده است. تکنیک‌های پیشرفته‌تر شامل حفظ سابقه اجرائی جزئی (وضعیت ثبت، حالت

[۴۳] fault localization

محاسباتی) و ریست و شروع مجدد از یک نقطه میانی در نرم‌افزار می‌شوند. اگر یک خطا پیچیده‌تر باشد، ممکن است نیاز به حذف و جایگزینی عنصر نرم‌افزار و سخت‌افزاری برای اثرگذاری بر بازیابی داشته باشد.

اگر سخت‌افزار و نرم‌افزار اضافی در دسترس باشد، بازیابی می‌تواند یک چشم‌انداز جهانی‌تر به خود بگیرد. بازیابی ممکن است شبیه دارایی‌های دیگر درون سیستم برای کار حول خطاها به نظر برسد. این شکل از بازیابی نیازمند باز تخصیص منابع (هم نرم‌افزاری و هم سخت‌افزاری) برای پر کردن خلأ بر جامانده از عناصر خطادار است. این شکل از بازیابی با عنوان پیکربندی مجدد مورد ارجاع قرار می‌گیرد. پیکربندی مجدد در بخش‌های بعدی با جزئیات بیشتری موردبحث قرار خواهد گرفت.

۲-۹-٤ سیستم‌های مدیریت پایگاه داده

سیستم مدیریت پایگاه داده از پنج عنصر تشکیل یافته است: سخت‌افزار و نرم‌افزار کامپیوتری، داده‌ها، افراد (کاربران)، و رویه‌های عملیاتی. سخت‌افزار کامپیوتری از عناصر پردازشی، حافظه فرار، اجزاء ذخیره‌سازی ثانویه، دستگاه‌های ذخیره‌سازی آرشیوی، دستگاه‌های ورودی و خروجی، و احتمالاً دستگاه‌های محاسباتی ویژه و سنسورهای ورودی تشکیل یافته است. نرم‌افزار مناسب برای یک پایگاه داده را می‌توان به سه طبقه‌بندی تقسیم نمود: نرم‌افزار پشتیبانی زیرساختی، نرم‌افزار پایگاه داده، و نرم‌افزار برنامه‌های کاربردی. نرم‌افزار پشتیبانی سخت‌افزاری دربرگیرنده سیستم‌عامل و نرم‌افزار ارتباطات شبکه می‌شود. نرم‌افزار سیستم مدیریت پایگاه داده دربرگیرنده اجزائی برای مدیریت حافظه، کنترل همزمانی، پردازش تراکنش، رابط دست‌کاری پایگاه داده، رابط تعریف پایگاه داده، و رابط کنترل پایگاه داده است. نرم‌افزار و برنامه‌های کاربردی به نیازهای کاربر بستگی دارد. داده‌ها کالای سیستم پایگاه داده هستند که در حال مدیریت شدن است. افراد و برنامه‌های کاربری، به‌عنوان کاربران، داده‌های ذخیره‌شده را دست‌کاری می‌کنند و، به‌عنوان مدیران پایگاه داده، پایگاه داده را برای کاربران موردبررسی و نگهداری قرار می‌دهند. رویه‌های عملیاتی برای فراهم‌سازی پشتیبانی اضافی از سیستم پایگاه داده توسعه داده‌شده و به اجرا درمی‌آیند. رویه‌های عملیاتی شامل بکاپ‌گیری از پایگاه داده بر روی حافظه انبوه غیر فرار، همچون نوارها، به‌صورت

زمان‌بندی‌شده، و گردآوری آمارهای عملیاتی برای استفاده در تنظیم ساختار و عملکرد پایگاه داده هستند.

یک سیستم پایگاه داده به‌صورت یک فرایند برنامه‌های کاربردی تحت کنترل سیستم‌عامل عمل می‌کند. مدیر پایگاه داده از سرویس‌های مدیریت حافظه و مدیریت فایل سیستم‌عامل برای ذخیره-سازی و بازیابی داده‌ها در پایگاه داده استفاده می‌کند. ارتباط با سیستم مدیریت پایگاه داده از طریق سه مسیر متمایز است: زبان تعریف پایگاه داده، زبان دست‌کاری پایگاه داده، و زبان کنترل پایگاه داده.

زبان تعریف پایگاه داده

یک پایگاه داده برای مدیریت داده‌هایی ساخته می‌شود که باید برای استفاده آتی حفظ شوند. داده‌های موجود در این پایگاه داده بر مبنای نیازهای اطلاعاتی برنامه‌های کاربردی در مجموعه-های ساختاربندی شده سازمان‌دهی می‌شوند. داده‌ها در این ساختارهای داده‌های از پیش تعریف‌شده در پایگاه داده قرار داده می‌شوند. این ساختارهای داده‌ها با استفاده از الزامات اولیه تعریف داده‌ها درون زبان پایگاه داده تعریف می‌شوند. نیازهای اولیه تعریف داده‌ها برای طراح پایگاه داده امکان مشخص‌سازی ترکیب هر یک از آیتم‌های داده‌ها و این ساختارهای پیچیده‌تر متشکل از آیتم‌های داده‌های سطح پایین را فراهم می‌آورند.

یک آیتم داده‌ها کوچک‌ترین بخش قابل‌تشخیص اطلاعات مدیریت‌شده درون پایگاه داده را نمایندگی می‌کند. یک نام منحصربه‌فرد به این شاخصه‌ها یا آیتم‌های داده‌ها تخصیص داده می‌شود، و نوع و ساختار فیزیکی‌شان با استفاده از انواع داده‌های در دسترس درون آن زبان مشخص می‌شوند. در زبان ساختارمند پرسش‌ها[۴۴] (SQL) مورداستفاده برای تعریف پایگاه داده‌های رابطه‌ای، یک آیتم داده‌ها همزمان با تعریف یک رابطه تعریف می‌شود. به‌عنوان‌مثال، برای تعریف رابطه یک فرد در SQL ما می‌توانیم از کد ذیل استفاده کنیم:

```
CREATE TABLE person
```

[۴۴] Structured Query Language

```
(name        VARCHAR (۳۰)  NOT NULL
 ssnum       INT (۹)  NOT NULL,
 bdate       DATE   NOT NULL,
 saddr       VARCHAR (۲۰)  NOT NULL,
 city        VARCHAR (۲۰)  NOT NULL,
 state       VARCHAR (۲۰)  NOT NULL,
 zcode       INT (۹)  NOT NULL,
             PRIMARY KEY (ssnum))
```

این مثال یک‌نهاد داده‌های فردی را تعریف می‌کند که از هفت آیتم داده‌ای متمایز تشکیل یافته است. یک نوع داده صریح و اندازه ماکسیمم برای آیتم داده‌ها به هر آیتم داده‌ها داده می‌شود – به‌عنوان‌مثال، طول این نام می‌تواند از ۱ تا ۳۰ کاراکتر باشد، تاریخ تولد از نوع تاریخ است. تاریخ در SQL با شکل سال،ماه،روز[۴۵] تعریف می‌شود و از چهار عدد اعشاری برای سال و دو عدد اعشاری برای هر دو نهاد ماه و روز تشکیل یافته است.

تعریف پایگاه داده نوعاً از یک فرایند تدوین برای ساخت و تولید طرح‌های پایگاه داده یا مدل توصیف داده‌ها استفاده می‌کند. فرایند تعریف پایگاه داده در توصیفات پایگاه داده از هر دو جنبه منطقی و فیزیکی و تولید نگاشت بین این دو حاصل می‌شود، که در بخش کد قبلی نشان داده‌شده است:

```
CREATE TABLE customer
 (cname       VARCHAR (۱۰)  NOT NULL
  cnum        INT (۳)  NOT NULL,
  credlim     DECIMAL (۶,  ۲),
              PRIMARY KEY (cnum))

CREATE TABLE order
```

[۴۵] year-month-day

```
(onum       DECIMAL (۵)  NOT NUL,
 cnum       DECIMAL (۳)  NOT NUL,
 spnum      SMALL INT  NOT NUL,
 date       DECIMAL (۶),
 amount     DECIMAL (۶,  ۲),
            PRIMARY KEY (onum))
```

این ساختارهای تعریف داده‌ها از روابط زبان ساختارمند پرسش‌ها (SQL) هستند و دو رابطه را مشخص می‌کنند. یکی رابطه مشتری است و دیگری یک رابطه سفارشی است. رابطه مشتری دارای سه مشخصه است: یک نام مشتری، یک شماره مشتری، و یک محدوده اعتباری. شاخصه کلیدی برای این رابطه به‌عنوان شاخصه شماره مشتری تعریف می‌شود. رابطه دوم یک رابطه سفارش مشتری است. رابطه سفارش مشتری از پنج شاخصه تشکیل یافته است: شماره سفارش، شماره مشتری، شماره بخش تأمین‌کننده، تاریخ سفارش، و مبلغ سفارش به دلار. کلید اولیه برای این رابطه به‌صورت شماره سفارش تعریف می‌شود. این توجه داشته باشید که ازآنجاکه شماره مشتری در رابطه سفارش به همان صورت شماره مشتری در رابطه مشتری است، این شاخصه یک کلید خارجی را در رابطه مشتری شکل می‌دهد. با استفاده از تکنیک‌هایی همچون این، این روابط از جنبه فوق به یکدیگر ارتباط داده می‌شوند.

برای تمام مدل‌های پایگاه داده، زبانی برای مشخصه ساختار و محتوای پایگاه داده وجود دارد. این مشخصه، طراحی طرح[۴۶] نامیده می‌شود و نشانگر دید منطقی اطلاعاتی است که باید توسط یک سیستم مدیریت پایگاه داده مدیریت شود. این مشخصه به طراح قابلیتی برای نگاشت دیدگاه‌های کاربر منطقی گسسته درباره اطلاعات در یک دید جهانی فراگیر از اطلاعات و نهایتاً درباره نگاشتی در ساختارهای ذخیره‌سازی فیزیکی می‌دهد. این تفکیک ساختارهای منطقی و فیزیکی منجر به شفافیت از وابستگی‌های فیزیکی و منطقی داده‌های کاربران می‌شود. با این کار، طراح پایگاه داده امکان تغییر ساختار و سازمان ذخیره‌سازی فیزیکی را به‌منظور بهینه‌سازی ذخیره‌سازی سطح پایین و راندمان بازیابی بدون نیاز به تغییر دید کاربر منطقی و کد برنامه‌های کاربردی آن می‌یابد.

---

[۴۶] schema design

زبان طراحی پایگاه داده، برای قابلیت اساسی برای تعریف داده‌ها، باید قابلیتی برای تغییر ساختارهای داده‌های مشخص‌شده و نمایش‌های فیزیکی‌شان پس از مشخص شدن پایگاه داده داشته باشند. ویژگی‌های مربوط به حذف یک ساختار از پایگاه داده، قرار دادن یک ساختار جدید، یا تغییر یک ساختار موجود باید برای تکمیل و برای نگهداری یک پایگاه داده در زبان گنجانده شوند، به خدمت گرفته شوند و در یک دوره زمانی کوتاه حذف گردند. وقتی سازمان‌ها یک پایگاه داده را ساخته و پر می‌کنند، این کار را به‌طور مداوم در طی عمر سیستمشان انجام می‌دهند. طول عمر یک سیستم پایگاه داده در چنان سازمانی ممکن است چندین دهه طول بکشد، که به‌طور ضمنی نشانگر غیرقابل‌اجتناب بودن رشد و تغییر است و باید برای نیازهای بعدی طراحی شوند. یک پایگاه داده درون چنان محیطی در ابتدا مشخص‌شده و به اجرا درمی‌آید. پس از استفاده از این پایگاه داده، برخی از تعدیلات اولیه لازم خواهد بود. علاوه بر این، با رشد سازمان و تغییر احتمالی تمرکز فعالیت‌هایش، مبنای اطلاعاتی آن نیز باید تغییر یابد تا رقابتی باقی بماند. باید دید که یک ساختار مشخصه و عملیاتی سخت و غیرقابل تغییر منتج به منسوخ‌شدگی و زوال عملکرد در خود برنامه‌های کاربردی که پایگاه داده در ابتدا برای پشتیبانی از آن‌ها مشخص‌شده بود می‌شود یا نه. یک‌زبان و پیاده‌سازی مشخصه پایگاه داده باید انعطاف‌پذیر باشد تا بتواند مفید و باثبات باقی بماند.

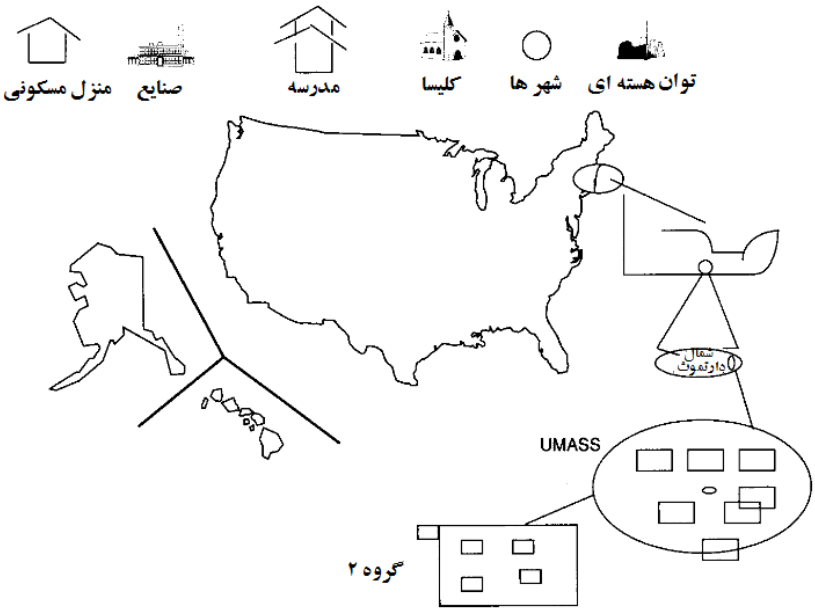
زبان دست‌کاری پایگاه داده

بخشی از پایگاه داده که برای متخصصان پایگاه داده، و این توسعه‌دهندگان برنامه‌های کاربردی و احتمالاً کاربران برنامه‌های کاربردی از همه قابل‌رؤیت‌تر و قابل‌تشخیص‌تر است زبان دست‌کاری داده‌ها است. این جزء از پایگاه داده می‌تواند شکل‌های بسیاری به خود بگیرد، که رایج‌ترین آن‌ها یک رابط شبیه زبان برنامه‌نویسی است، که فرصتی برای بازیابی و ذخیره‌سازی اطلاعات درون پایگاه داده فراهم می‌آورد که قبلاً توسط زبان طراحی پایگاه داده مشخص‌شده است.

بااین‌حال، لزومی ندارد که زبان دست‌کاری داده‌ها یک دید زمینه‌ای و رویه‌ای به خود بگیرد. زبان دست‌کاری داده‌ها می‌تواند، همانند آنچه در سیستم مدیریت داده‌های فضایی وجود دارد، بصری باشد، که در آن اطلاعات با استفاده از آیکون‌ها توصیف می‌شود و با استفاده از تصاویری بازیابی می‌شود که

می‌تواند برای کسب اطلاعات بیشتر درباره یک آیتم بر رویشان زوم کرد – به‌عنوان‌مثال: با توجه به اینکه ما نقشه‌ای از ایالات‌متحده داریم که به‌عنوان دید سطح بالا برای جست‌وجوی اطلاعات کسب‌وکار استفاده می‌شود، ممکن است مایل باشیم بفهمیم که چه دانشگاه‌هایی را در بخش جنوبی ماساچوست به Cape Cod نزدیک‌تر هستند. ما ابتدا می‌توانیم انواعی از آیکون‌ها را انتخاب کنیم که مایلیم نمایش داده‌شده باشند – به‌عنوان‌مثال، با انتخاب آیکون دانشگاه‌ها فقط مناطقی نشان داده شوند که دانشگاه دارند. نمایش بصری سپس می‌تواند شهرهایی را نشان دهد که در آن‌ها دانشگاه‌های عمده واقع‌اند. برای جداسازی یک دانشگاه مشخص یا یافتن اطلاعات بیشتر درباره جایی که دانشگاه در آن واقع است، ما کار را با انتخاب بخش شروع می‌کنیم، مثلاً نیو انگلند جنوب شرقی حول کیپ کد، با کشیدن دایره حول این منطقه. سپس نمایشگر این ناحیه را نمایش می‌دهد، که در این مورد هم فقط با نمایش دانشگاه‌ها است. برای انتخاب یک دانشگاه مشخص، یک آیکون دانشگاهی را انتخاب کنید (شکل ۲-۲۸). اگر ما دانشگاه ماساچوست در دارتموث را انتخاب می‌کردیم، متعاقباً می‌توانستیم یک نمای هوایی از دانشگاه نیز داشته باشیم. برای کشف اطلاعات بیشتر، ما می‌توانیم یک ساختمان، سپس یک دپارتمان، یا احتمالاً حتی یک استاد مشخص یا دوره پیشنهادی را انتخاب کنیم. به چنین طریقی، اکثر اطلاعات موردنیاز را می‌توان در شکل بصری استخراج نموده و نمایش داد. بااین‌حال، این روش محدودیت‌هایی دارد. تمام اطلاعات برای نمایش فقط بصری مناسب نیستند. ممکن است ما ناچار به قرار دادن تنها یک زیرمجموعه از اطلاعات کاملاً در دسترس در چنان سیستمی شویم و از یک رابط پایگاه داده مجزا برای اطلاعات متنی بیشتر استفاده کنیم.

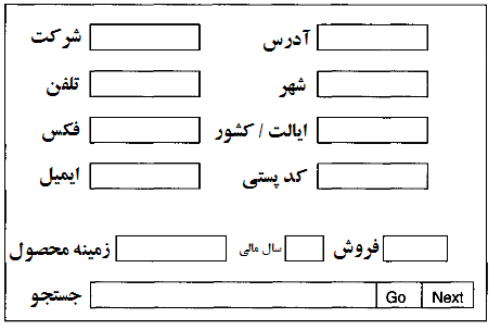
رابط نوع دومی بیشتر به استفاده‌های شغلی پایگاه داده‌ها ارتباط دارد. این نوع از اطلاعات بیشتر به سمت استفاده‌های شغلی از پایگاه داده‌ها ارتباط می‌یابد. این نوع از رابط از شکل‌های کاغذی معمول یک شرکت برای اطلاعات درباره موجودی، فروش، سوابق کارکنان و امثالهم به‌عنوان رابط ارائه شده به کاربران پایگاه داده استفاده می‌کند. یک برنامه کاربردی یا کاربر به راحتی شکل مناسب، مثلاً یک شکل سابقه کارکنان، را انتخاب می‌کند، و انتخاب می‌کند که، با تایپ کردن اطلاعات در فرم، به چه سوابقی از کارمندان یا گروهی از آنان نگاه کند.



شکل (۲-۲۸): سیستم مدیریت داده‌های فضایی

شکل (۲-۲۹) فرمی را نشان می‌دهد که توسط یک کسب‌وکار برای نشان دادن مشتریان یا تأمین‌کنندگان می‌تواند استفاده شود. این فرم اطلاعات عمده شرکت، همچون نام، نشانی، شماره تلفن، شماره نمابر، نشانی ایمیل، و احتمالاً برخی اطلاعات درباره نوع محصول یا سرویسی که ارائه می‌کند را نشان می‌دهد. علاوه بر این، این فرم می‌تواند دربرگیرنده چند فیلد دیگر هم باشد، که می‌تواند برای کمک به یافتن اطلاعات استفاده شود. در صفحه مثال شکل (۲-۲۹)، یک فیلد مجزا به نام Find در پایین صفحه وجود دارد. در این فیلد، یک کاربر می‌تواند پارامترهایی که باید موردبررسی قرار بگیرند یا ملاک‌های کمکی در یک جستجو را وارد نماید – به‌عنوان‌مثال، اگر ما مایل به انتخاب تمام شرکت‌ها در بوستون ماساچوست بودیم، که در پایگاه داده ما ذخیره شده‌اند، دو راه بالقوه برای این کار وجود دارد. راه اول وارد کردن نام‌های بوستون و ماساچوست در فیلدهای مناسب شهر و ایالت و انتخاب Go در سمت راست پایین صفحه است. این می‌تواند به پایگاه داده اشاره نماید که با هرگونه سوابقی که این ملاک‌ها را در این زمینه‌های مشخص دارند تطابق داشته باشد. برای یافتن ورودی بیشتر با همین فیلدها،

می‌توان فیلد Next در گوشه سمت راست بالا را انتخاب کرد. یک ابزار دیگر برای بازیابی همین سوابق نوع همه (All)، بوستون، و ماساچوست است.



شکل (۲-۲۹): فرم نمونه

شکل سومی از زبان دست‌کاری داده‌های غیر سنتی نوع امکان سؤال از طریق مثال[۴۷]، یا QBE، است. در یک محیط سؤال از طریق مثال، کاربر اطلاعات اساسی را درباره یک سابقه مورد توجه درخواست می‌کند – به‌عنوان‌مثال، نام یک کمپانی. سیستم سپس یک الگو را برمی‌گرداند، که ممکن است با آنچه درخواست شده تطابق داشته باشند و یا نداشته باشد. این الگو سپس می‌تواند برای پالایش بیشتر سؤال و دریافت مثال‌های بیشتر برای استفاده در تنظیم یک سؤال دقیق‌تر استفاده شود. رابط QBE توسعه داده‌شده برای مدل رابطه‌ای ارتباطی تنگاتنگ با رابط مبتنی بر فرم دارد. این مثال‌ها در شکل جداول برمی‌گردند، و کاربر کمیت‌های شناخته شده را پر می‌کند. پایگاه داده سپس تلاش می‌کند که با استفاده از این اطلاعات یک جدول پاسخ را به‌صورت اطلاعات محدودسازی پر کند.

دیگر زبان‌های دست‌کاری داده‌ها مبتنی بر ارزیابی کارکردی هستند. در این نوع از زبان‌ها، کاربران اطلاعات را از طریق استفاده از فراخوان‌های کارکردی از پایگاه داده درخواست می‌کنند. فراخوان‌های کارکردی می‌توانند متن، گرافیک، ویدئو، صدا یا انواع مختلفی از فرمت‌های داده‌ها را برگردانند. فرم برگردانده شده به فرمت‌های داده‌های کارکرد فراخوانی شده و انواع داده‌های پارامترها بستگی دارد. این شکل از رابط سؤال در پایگاه داده‌های شیءمحور و در پایگاه داده‌های چندرسانه‌ای و ابر رسانه‌ای بیشترین شیوع را دارد. اطلاعاتی که بین پایگاه داده و برنامه‌های کاربردی انتقال داده می‌شود در شکل

[۴۷] query by example

بومی برنامه‌های کاربردی است، نه در شکل پایه پایگاه داده. این نوع از رابط در برنامه‌های کاربردی مطلوب است که در آن‌ها داده‌ها در شکل‌های غیرمتنی ارائه می‌شوند که در غیر این صورت توسط یک سیستم مدیریت پایگاه داده ذخیره و مدیریت می‌شده‌اند.

شایع‌ترین شکل زبان دست‌کاری داده‌ها در حال حاضر همچنان با فاصله زیادی نسبت به دیگران زبان‌های متنی و رویه‌ای، همچون زبان ساختارمند پرسش‌ها (SQL) و زبان پرسمان شیء (OQL) است. در این زبان‌ها، سؤال‌ها تا حدود زیادی بیشتر شبیه یک برنامه در هر زبان برنامه‌نویسی شکل می‌گیرند. نویسنده سؤال چند کلمه ذخیره‌شده دارد که برخی کارکردهای معین را ارائه می‌نماید. این سؤال نوعی دربرگیرنده کلمات معکوس شده برای انتخاب رکوردهای متعدد، یک رکورد واحد، یا زیرمجموعه‌ای از یک رکود می‌شود، برای اینکه رکورد باید از کجا بیاید، و هرگونه ملاک در زمینه دسترسی و بازیابی اطلاعات درخواست شده. در زبان‌های این شکلی، سؤال‌ها ساختار و جریان اجرائی برنامه را به خود می‌گیرند – به‌عنوان‌مثال، اگر ما به دنبال یک سفارش (Order) رابط‌های، از شکل شماره سفارش، تاریخ سفارش، شماره مشتری، محصول سفارش داده‌شده، کمیت سفارش داده‌شده، هزینه واحد، و هزینه کل بگردیم، و می‌خواهیم که تمام سفارش‌ها (کل چندگانه) را از شرکت *XYZ* برای صحافی‌ها از مه ۱۹۹۵ بیابیم، می‌توان از سؤال ذیل استفاده نمود، یا توجه به اینکه شرکت *XYZ* شماره مشتری I۱۰۱ را داشته باشد:

```
Range of O is order;
Select  O.onum, O.odate, O.cnum, O.pname, O.qty, O.uic,
O.ttl
From Orders
WhERE O.cnum := 'C۱۰۱' and O.odate > '۴،۳۰،۹۵' and
O.pname := 'bindings' ;
```

در سؤال، ما ابتدا یک متغیر داخلی را برای دربرگیری تمام مقادیر رابطه جستجو تعیین می‌کنیم. سپس، به جستجو می‌پردازیم تا تمام شاخصه‌های "سفارش" رابطه را در همان سفارشی که در آن ذخیره شده‌اند، یعنی در سفارش رابط‌های، بازیابی کنیم. و، نهایتاً اینکه، ما انتخاب را به یافتن محدود می‌کنیم و

تنها آن دسته از چندتایی‌هایی را در رابطه نتیجه‌مان کپی می‌کنیم که در آن‌ها شاخصه شماره شرکت با مقدار 'C۱۰۱' در آن‌ها ذخیره شده باشد، مقدار برای تاریخ سفارش شاخصه بیشتر از انتهای آوریل ،(□۴،۳۰،۹۵□ $< odate$)و فقط بخش‌هایی که نام اتصال ('bindings') را دارند.

زبان‌های رویه‌ای همچون SQL نیز عملگرهایی برای قرار دادن چندتایی‌های جدید در پایگاه داده، برای ایجاد روابط جدید، برای اصلاح روابط موجود، برای حذف روابط، و به‌روزرسانی محتوای یک رابطه دارند. قرار دادن یک چندتایی در رابطه قبلی را به راحتی می‌توان با صدور دستورالعمل ذیل به اجرا در آورد:

```
Range of ٠ is order;
INSERT INTO Orders
VALUES ('٠١٠٠', '٥،٢١،٩٥', 'C١٠١', 'binding', '١٠٠',
'١,٢٥', '١٢٥,٠٠');
```

حذف یک چندتایی یکسان از پایگاه داده الزام می‌کند که ما ابتدا ما چندتایی مناسب را بیابیم، و سپس آن را از رابطه حذف کنیم. این کد می‌تواند به‌صورت ذیل ظاهر شود:

```
Range of ٠ is order;
DELETE FROM  Orders
Where O.cnum:= 'O١٠٠' AND O.odate := '٥،٢١،٩٥', AND
O.cnum, := 'C١٠١' AND O.pname := 'binding' AND O.qty :=
'١٠٠' AND O.uic := '١,٢٥' AND
O.ttl := '١٢٥,٠٠';
```

یک ابزار ساده‌تر می‌تواند ارجاع به چندتایی مورد نظر توسط فقط کلید اولیه‌اش باشد. از آنجا که در یک پایگاه داده رابطه‌های کلید اولیه برحسب تعریف باید به شکلی منحصر به فرد یک چندتایی را در یک رابطه تعریف کند، می‌توان از این به تنهایی برای یافتن و حذف چندتایی مناسب استفاده نمود. عملیات حذف اصلاح شده می‌تواند به‌صورت ذیل باشد:

```
Range of ٠ is order;
```

```
DELETE FROM  Orders
Where O.cnum:= 'O١٠٠'
```

آنچه خواننده باید از این بحث درک کند این است که هیچ ابزار درست واحدی برای بازیابی اطلاعات از یک پایگاه داده وجود ندارد. بااین‌حال ابزارهای استانداردی وجود دارند [١]. مفهوم مهم این است که بازیابی پایگاه داده با فرایندهای زبان برنامه‌نویسی متداول متفاوت است. یک زبان دسترسی اطلاعاتی وجود دارد، که در کنار تکنولوژی پایگاه داده تکامل یافته و به تکامل ادامه می‌دهد. بااین‌حال این زبان همچنان با همتایان زبان برنامه‌نویسش متفاوت است که در ابتدا به علت تفاوت‌ها در الزامات برای داده-های مداوم برای نقطه وجود یک برنامه و الزامات برای ثبات و درست بودن اطلاعات برای حوزه عمل هر فرایند یا برنامه‌ای است.

زبان کنترل پایگاه داده

آخرین بخش رابط زبان به یک سیستم مدیریت پایگاه داده زبان کنترل داده‌ها است. این گاهی به‌عنوان بخشی از زبان تعریف داده‌ها در برخی از توصیفات نیز گنجانده می‌شود. ما در اینجا آن را تجزیه می‌کنیم تا به تمرکز بر سرخی از تفاوت‌ها کمک شود. به طور خاص، این جزء از یک رابط پایگاه داده معمولاً توسط مدیر پایگاه داده استفاده می‌شود. کارکردهای معمول فراهم‌سازی شده در این لایه ابزارهایی برای مانیتور عملیات پایگاه داده، ساختاربندی مجدد مخزن فیزیکی، ریست کردن مقادیر محدوده بر روی آیتم‌های داده‌ها، ریست کردن مقادیر داده‌ها بر روی آیتم‌های داده‌ها، تغییر نام روابط، ایجاد شاخص‌های اضافی، داده‌های محدودکننده، و اعطا، تغییر یا ابطال مزایا هستند. رابط نوعی در این سطح به‌صورت متنی با ابزارهای تحلیل ویژه مورداستفاده برای تحلیل اطلاعات گردآوری شده جهت‌دهی می‌شود.

مدیر پایگاه داده، به‌عنوان‌مثال، می‌تواند مجموعه‌ای از محدودیت های کاربری بر روی یک مقدار ویژگی را بررسی نموده و، بر مبنای الزامات کاربر، به‌منظور افزایش حوزه محتمل مقادیر درست لحاظ شده توسط این شاخصه تصمیم به تغییر آن‌ها بگیرد – به‌عنوان‌مثال، اگر مدیر پایگاه داده احساس کند

که طیف کافی از مقادیر برای نمایش طبقه‌بندی‌های شغلی و عناوینشان به‌صورت موردنیاز وجود ندارد. اگر در ابتدا تنها سه عنوان در این شرکت وجود داشته باشد:

Jobtitle IN (welding, management, sales)

ولی مقرر می‌شود که ساختار داده‌ها باید به شکلی بسط داده شود که به شکلی کامل‌تر نیاز طبقه‌بندی‌های شغلی اضافی را برطرف نماید، مدیر داده‌ها صرفاً لیست آیتم‌های معتبر را بسط می‌دهد. این دستورالعمل مشخصاً سه طبقه‌بندی جدید را به لیست عناوین شغلی مجاز می‌افزاید. این عناوین جدید اکنون از طریق سؤال برنامه‌های کاربردی یا اصلاح پایگاه داده قابل استفاده هستند.

Jobtitle IN (welding, management, metal cutter,
Machinist, glass cutter, sales)

محدودیت‌ها برای محدوده مقادیر یک آیتم داده‌ها با افزایش مقادیر تخصیص داده‌شده به مقادیر مرزی قابل تغییر است – به‌عنوان‌مثال، اگر یک محدودیت اضافی نشانگر آن باشد که شماره مشتری در محدوده ۱ ت ۵۰۰ است، ولی ما اکنون خود را با ۵۰۱ مشتری بیابیم، این محدودیت باید تغییر داده شود تا امکان ذخیره‌سازی رکورد مشتری جدید فراهم بیاید. به‌منظور تغییر این محدودیت، کافیست RANGE OF Customer.cnum ۱...۷۵۰ تنظیم شود. محدودیت‌ها درباره زمان اجرای کارکردهای آزمایش را نیز می‌توان تغییر داد –به‌عنوان‌مثال، محدودیت های تست بر خواندن ها، نوشتن ها یا تعهد.

برای تغییر محدودیت‌ها، زبان‌های کنترل داده‌های پایگاه داده دستورالعمل‌ها و محدودیت‌هایی برای اعطای مزایا اضافی به کاربران یا لغو مزایا فراهم می‌آورند. عبارت GRANT برای این استفاده می‌شود که یک کاربر بتواند دست‌کاری‌های مشخصی را انجام دهد – دادن اجازه به کاربر Tom برای اینکه مقادیر را از رابطه Customer بخواند بدین طریق قابل انجام است:

GRANT SELECT ON Customer TO Tom;

این می‌توان حق عملیات متعدد در یک عبارت را نیز داد:

GRANT SELECT, UPDATE, INSERTION ON Customer TO Tom;

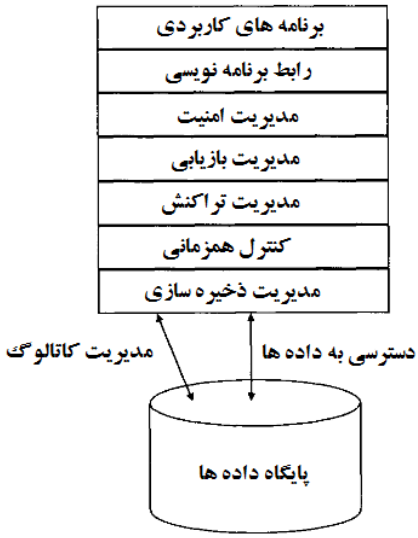
این عبارت حقوق انتخاب، آپدیت و قراردهی را به کاربر Tom در رابطه Customer اعطا می‌نماید. بدین طریق، مدیر پایگاه داده می‌تواند حقوق دسترسی به هر آیتم درون پایگاه داده را تغییر داده، اضافه نماید یا حذف کند. همه مدل‌ها و سیستم‌های پایگاه داده از طیف وسیعی از ویژگی‌های زبان کنترل داده‌ها پشتیبانی نمی‌کنند. در چند تا از زبان‌ها، بسیاری از این ویژگی‌ها ممکن است آفلاین کردن پایگاه داده برای تغییرات را الزامی نمایند.

۲-۱۰ اجزاء یک معماری سیستم پایگاه داده

یک سیستم پایگاه داده متشکل از چیزی به‌مراتب بیش از صرفاً زبان تعریف داده‌ها، زبان دست‌کاری داده‌ها، و زبان کنترل داده‌ها است. این مشخصاً رابط در سیستم پایگاه داده واقعی را نمایندگی می‌کند. هسته یک سیستم مدیریت پایگاه داده مجموعه سرویس‌هایی است که امکان تداوم داده‌ها در پایگاه داده و کارکردی برای تضمین همخوانی و صحت داده‌ها و تبعیت از ویژگی‌های ACID توسط تراکنش‌ها را فراهم می‌آورند (شکل ۲-۳۰). ویژگی‌های ACID شامل اجرای اتمی، همخوان، مستقل و بادوام یک تراکنش بر روی پایگاه داده می‌شوند. ما در ادامه این فصل با جزئیات بیشتری درباره این‌ها بحث خواهیم نمود.

معماری یک سیستم پایگاه داده مجموعه‌ای از سرویس‌های ایجاد شده بر روی سرویس‌های سیستم‌عامل پایه، سرویس‌های ذخیره‌سازی فایل سیستم، و سرویس‌های مدیریت بافر حافظه اولیه است. مدیر فایل رابط پایگاه داده با اطلاعات ذخیره شده پایا است. اطلاعات مدیریت شده برای دیتابیس توسط فایل سیستم دربرگیرنده طرح‌های داخلی، مفهومی و خارجی برای اطلاعات ذخیره شده (ابر پایگاه داده[۴۸])، پایگاه داده واقعی، و لاگ فایل پایگاه داده است. لاگ فایل‌ها شامل تصاویر قبلی (مقادیر بافر)، تصاویر بعدی، رکوردهای اجرای مجدد[۴۹]، (اقدامات تراکنش‌ها متعهد)، رکوردهای بازگشتی (اقدامات تراکنش‌ها غیرمتعهد)، رکوردهای تعهدی، رکوردهای توقفی[۵۰]، و رکوردهای آغاز تراکنش (تراکنش) می‌شوند.

[۴۸] metadatabase
[۴۹] redo records
[۵۰] abort records



شکل (۲-۳۰): معماری برای پشتیبانی یک سیستم پایگاه داده

از طریق ویژگی‌های اساسی مدیریت فرایند، ارتباطات بین فرایندی، همگام‌سازی، مدیریت بافر، و مدیریت فایل سرویس‌های سیستم‌های پایگاه داده را می‌توان ساخت. این سرویس‌ها شامل مدیریت کاتالوگ، مدیریت تلفیق، مدیریت تراکنش، کنترل همزمانی، مدیریت امنیت، پردازش سؤال، مدیریت ارتباطات، و مدیریت لاگ. بالای سرویس‌های پایگاه داده، برنامه‌های کاربردی کاربری از طریق مدیر دید ورودی/خروجی و مدیر دست‌کاری داده‌ها عمل می‌کنند. در پاراگراف‌های بعدی ما مختصراً هر یک از این موارد را مرور خواهیم نمود. پس‌ازاین مرورهای کوتاه، ما برخی از این‌ها را با جزئیات بیشتر مرور خواهیم کرد.

۲-۱۰-۱ مدیر کاتالوگ

مدیر کاتالوگ اطلاعاتی درباره اطلاعات پایگاه داده را داراست. این ابر داده‌ها طرح‌ها برای پایگاه داده را شکل می‌دهند. مدیر پایگاه داده، با استفاده از زبان کنترل داده‌ها و رابط‌های زبان تعریف داده‌ها، می‌تواند این طرح‌ها را تغییر دهد. به‌عنوان‌مثال، در SQL این سهم از پایگاه داده می‌تواند

این تعریف را برای تمام روابط، محدودیت‌ها، تأکیدات امنیتی، و نگاشت‌ها در مخزن فیزیکی حفظ نماید.

۲-۱۰-۲ مدیر یکپارچگی

مدیر یکپارچگی به نگهداری دقت، صحت، و اعتبار آیتم‌های داده‌های پایگاه داده کمک می‌کند – به‌عنوان‌مثال، مدیر یکپارچگی می‌تواند از طریق مکانیسمی که زمان انجام بررسی را تعیین می‌کند چک کند که یک آیتم داده‌ها از نوع مناسبی است یا نه، چگونه این چک انجام شود، وقتی رخ داد چگونه بازیابی، رد یا تعمیر شود. مدیر یکپارچگی می‌تواند چک کند که یک آیتم داده‌ها در یک حوزه از پیش تعریف‌شده مقادیر صحیح قرار دارد یا نه، مثلاً DOMAIN FIXED (۵) یا Weight GREATER THAN ۰ AND Weight LESS THAN ۲۰۰۰. این‌ها می‌توانند محدوده‌ای از داده‌ها که یک آیتم داده‌ها می‌تواند تحت پوشش قرار دهد را تست کنند. بررسی‌های یکپارچگی می‌توانند نهادها یا روابط متعددی را تحت پوشش قرار دهند – به‌عنوان‌مثال، یک چک یکپارچگی ارجاعی در SQL را می‌توان مورداستفاده قرار داد تا مشخص شود که رابطه بسیاری از اشیاء دارای این ویژگی است که برای معتبر ماندن باید برای آن‌ها برقرار بماند. چنان چک کردنی می‌تواند به این صورت باشد که مجموع تمام مانده حساب‌ها در یک بانک باید برابر با مانده بانک باشد. یک جنبه مهم این مدیریت زمان اجرای بررسی‌های مشخص‌شده است – به‌عنوان‌مثال، اگر چک کردن در زمان تعریف پایگاه داده انجام شوند، در زمان دسترسی به آیتم داده‌ها، در زمان آپدیت یک آیتم داده‌ها، در رخدادی همچون یک تایمر، یا در زمان تعهد یک تراکنش هزینه‌های متفاوت وجود دارد. چک‌های انجام‌شده در طی زمان اجرا بازدهی پردازش پایگاه داده را کند خواهند نمود.

۲-۱۰-۳ مدیر تراکنش

مدیر تراکنش اجرای تراکنش‌ها درون پایگاه داده را هماهنگ‌سازی و کنترل می‌کند. در حال حاضر فقط فرض کنید که یک تراکنش مجموعه‌ای از عملیات بر روی پایگاه داده‌ای باشد که در یک واحد زمان اجرای واحد به یکدیگر پیوند داده‌شده‌اند. مدیر تراکنش باید کارهایی را برای

این موارد انجام دهد: شروع تراکنش‌ها (زمان‌بندی)، همگام‌سازی اجرای تراکنش با پایگاه داده، تراکنش‌های دیگر، و سیستم‌عامل، هماهنگ‌سازی ارتباطات درون تراکنشی، تعهد (تکمیل) پردازش، و لغو (شکست) پردازش، چک کردن محدوده تراکنش، و هندلینگ شرایط، و این مدیریت بازیابی تراکنش (خطا). یک تراکنش معمولاً دارای شکل ذیل است:

TRANACTION T (Optional Input Parameters)
Specification Part
BEGIN
BODY of T
COMMIT or ABORT of T
RECOVERY PART of T
END
END TRANSACTION T

عبارت اول تراکنش را نام می‌برد، و این امکان را فراهم می‌آورد که از پیش تدوین شده و برای اجرای بعدی ذخیره شود. عبارت اول این مجال را برای انتقال پارامترهای ورودی به تراکنش، همچون مکان داده‌هایی که باید اجرا شوند، فراهم می‌گذارد. بخش مشخصه تراکنش جایی است که در آن متغیرهای محلی برای محل کار تراکنش، همانند پیش شرط‌ها و پس شرایط اجرای تراکنش، شرایط بازیابی، سطح جداسازی، حالت‌های دسترسی، و اندازه تشخیصی برای تخصیص مشخص می‌شوند. هر دو دربرگیرنده کد قابل اجرا برای تراکنش هستند. عبارات تعهد و لغو نشانگر موفقیت یا شکست تراکنش هستند. نهایتاً اینکه، بخش بازیابی تأمین شده توسط کاربر یا سیستم با هندلرهای وضعیت برای پردازش خطا و پردازش تکمیل تراکنش را مشخص می‌کند.

۲-۱۰-٤ مدیر کنترل همزمانی

مدیر کنترل همزمانی اقدامات دسترسی تراکنشی با پایگاه داده را با اجرای همزمان تراکنش‌ها هماهنگ‌سازی می‌کند. هدف کنترل همزمانی هماهنگ‌سازی اجرا به شکلی است که VIEW یا اثر ازنظرگاه پایگاه داده به همان صورتی باشد که انگار تراکنش‌ها اجرای همزمان به شکل

سریالی اجرا شده‌اند. این طرح با عنوان اجرای قابل سریالی‌سازی تراکنش‌ها مورد ارجاع قرار می‌گیرد. نظریه سریال پذیری کنترل همزمانی دو حالت پایه دارد: ساده‌ترین آن‌ها به اجرای قابل سریالی‌سازی مجموعه‌های خواندن و نوشتن از تراکنش‌ها متداخل توجه دارد و مبتنی بر لاکینگ (قفل‌سازی)، سفارش دهی تمبر زمانی[۵۱]، یا حل و فصل تداخل خواندن و نوشتن خوش‌بینانه است. دومین مفهوم کنترل همزمانی پیچیده‌تر است و از دانش معنایی اجرای یک تراکنش برای کمک به هماهنگ‌سازی استفاده می‌کند. تفاوت عمده آن این است که دانه‌بندی عملگر سریال بندی خواندن و نوشتن نیست بلکه عمدتاً کارکردها و رویه‌های پیچیده و این اشیاء داده‌های پیچیده است. بااین‌حال، معیار اجرای درست سریال بندی بر روی تراکنش‌ها همزمان است.

۲-۱۰-۵ مدیر قفل

مدیر قفل برای کنترل دسترسی به جدول قفل پایگاه داده طراحی شده است. جدول قفل پایگاه داده وضعیت قفل‌ها (قفل خواندن، قفل نوشتن، اشتراک‌گذاری قفل، قفل معنایی و غیره) را برای هر آیتم پایگاه داده که مورد دسترسی قرار گرفته باشد حفظ می‌کند. مدیر قفل کاربران را از دسترسی مستقیم به جدول قفل حفاظت می‌کند. به‌منظور کسب دسترسی به وضعیت قفل، مدیر قفل مقدمات قفل و باز کردن را برای پایگاه داده و کد کاربر فراهم می‌آورد. قفل می‌تواند یک قفل خواندن باشد، که وقتی اعطا می‌شود که یک تراکنش تلاش کند یک آیتم داده‌ها را بخواند. یک قفل خواندن تنها زمانی قابل اعطا است که هیچ تراکنش برای خواندن یک آیتم داده‌ها تلاش نماید (در صورتی که هیچ کس یک قفل نوشتن متداخل را نداشته باشد). یک قفل نوشتن تنها در صورت قابل اعطا است که هیچ تراکنش دیگری یک قفل خواندن یا نوشتن بر روی آیتم داده‌ها نداشته باشد. قفل‌ها در یک پایگاه داده را می‌توان مانند سمافورها در یک سیستم‌عامل دید، آن‌ها به‌عنوان ابزاری برای تضمین استفاده انحصاری از یک آیتم در کنترل پایگاه داده استفاده می‌شوند.

---

[۵۱] timestamp ordering

۲-۱۰-٦ مدیر بن‌بست

وقتی یک پروتکل قفل‌سازی مورداستفاده قرار گیرد، یک قفل نگهداری شده توسط یک تراکنش می‌تواند یک درخواست قفل را از تراکنش دیگر قفل نماید. اگر هیچ انتظار چرخه‌ای برای یک قفل وجود نداشته باشد، قفل نهایتاً اعطا خواهد شد. اگر انتظارهای چرخه‌ای وجود داشته باشند، بن‌بست رخ می‌دهد. بن‌بست وضعیتی است که در آن دو یا چند معامله برای منابع تحت نگهداری تراکنش دیگری که منتظر منبعی هستند که شما در اختیار دارید منتظر می‌مانند. از آنجا که هیچ‌کس نمی‌تواند جلو برود، سیستم نمی‌تواند هیچ کار مفیدی را به انجام برساند. مدیر بن‌بست باید زمانی که یک وضعیت قفل برقرار است را شناسایی نماید و نحوه مدیریت وضعیت را تعیین نماید. معمولاً، یکی از تراکنش‌ها دخیل ناتمام گذاشته می‌شود و قفل‌های آن آزاد می‌شوند، و بدین طریق امکان ادامه تراکنش‌ها دیگر فراهم می‌شود.

۲-۱۰-۷ مدیر بازیابی

مدیر بازیابی باید اطمینان حاصل کند که پایگاه داده همیشه در وضعیتی است که به طور مداوم و درست قابل بازیابی است. این کار با کسب اطمینان از اینکه پایگاه داده دربرگیرنده تمام یا هیچ کدام از اثرات تراکنش‌ها تعهد داده‌شده و هیچ مورد از تراکنش‌ها ناتمام رهاشده یا اجرائی است انجام می‌شود. مدیر بازیابی از مفهوم یک نقطه بررسی (تصویر لحظه‌ای از وضعیت فعلی پایگاه داده) و یک لاگ فایل (فایل عملیات روی پایگاه داده) برای کمک در بازیابی استفاده می‌کند. برای پایگاه داده‌های شرطی، بازیابی برای برگرداندن پایگاه داده به یک وضعیت قدیمی پایگاه داده و شروع پردازش از آنجا تلاش می‌کند. برای برگرداندن پایگاه داده به یک وضعیت قبلی، مدیر بازیابی هم برگرداندن به عقب[۵۲]، جایی که دیدگاه‌های قبلی تراکنش فعال یا نامتعهد برگردانده می‌شوند، و اجرای مجدد[۵۳]، در جایی که حالت‌های جدید تراکنش‌ها تعهدیافته نوشته نشده در پایگاه داده به مخزن مداوم برگردانده می‌شوند را انجام می‌دهد. این رکوردهای آندو

---

[۵۲] undo
[۵۳] redo

ریدو برای یک حالت چک‌پوینت اعمال می‌شوند تا پایگاه داده به یک وضعیت باثبات میانی قابل قبول برگردد. شکل ثانویه‌ای از بازیابی تلاش می‌کند که پایگاه داده را با اعمال تراکنش‌ها جبران-کننده (تغییر اثرات متعهد شده به شکل‌های قابل قبول بر مبنای نیازهای معنایی)، با اعمال برونیابی (برای محاسبه حالت‌های آتی باثبات و صحیح قابل قبول)، و با اعمال هندلرهای وضعیت برای اقدامات معنایی سیستم یا کاربری به سطوح متفاوتی درون پایگاه داده انتقال دهد.

۲-۱۰-۸ مدیر امنیت

مدیر امنیت وظیفه محدودسازی دسترسی، اصلاحات، و نفوذ بدخواهانه به پایگاه داده را دارد. برای اجرای این اقدامات کنترلی، مدیر امنیتی الزام می‌کند که کاربران تعیین هویت شوند، احراز هویت شوند و برای دسترسی به کنترل در رابطه با یک آیتم داده‌های درخواست شده مجوز دهی شوند. تعیین هویت مشابه باقابلیت‌های لاگین نوعی است، که در آن‌ها مدیر امنیتی از کاربران می‌خواهد که خودشان را تعیین هویت کنند. برای کسب اطمینان از اینکه هیچ کسی برای دسترسی به پایگاه داده تلاش نکند می‌توان از یک کاربر خواست که هویتش را احرازهویت کند. این کار با یک گذرواژه یا با انواع مختلفی از مکانیسم‌های نسبتاً پیچیده قابل انجام است. وقتی کاربر مجاز به دسترسی باشد، وی به آنچه قابل دیدن یا تغییر باشد محدود می‌شود. احرازهویت کارکرد محدودسازی دسترسی به تنها یک سطح مطلوب از پیش تعریف‌شده را انجام می‌دهد – به‌عنوان‌مثال فقط خواندن، فقط نوشتن، تغییر قابلیت، دیدن محدودیت، و محدودیت های متعدد دیگر.

۲-۱۰-۹ مدیر پشتیبانی از پردازش سؤال

پردازنده سؤال یک سیستم پایگاه داده وظیفه تعیین نحوه پاسخ‌دهی به درخواست‌ها برای اطلاعات از یک کاربر به بهینه‌ترین شکل را دارد. ایده این است که بتوان به یک سؤال به طرق مختلفی توسط یک سیستم پایگاه داده جواب داد. سرراست‌ترین آن‌ها رویکرد جامع[۵۴] است. بااین‌حال، این نوعاً گران‌ترین حالت از جنبه مصرف زمان و منابع است – به‌عنوان‌مثال، هزینه اتصال دو جدول هزینه

[۵۴] brute-force

اسکن کردن هر آیتم اولی به هر آیتم دومی است، یا از مرتبه N ضرب در N یا مربع N در صورتی است که ما فرض کنیم آن‌ها اندازه یکسانی دارند. از سوی دیگر اگر ما بتوانیم اندازه هر یک را با یک ضریب ۲ کاهش دهیم، هزینه به میزان نصف کاهش می‌یابد. این به راحتی حاصل می‌شود اگر ما یک انتخاب را اول پیش از یک الحاق بر روی هر یک پیش از یک الحاق انجام دهیم. اگر اندازه N بزرگ باشد، این کاهش ممکن است قابل توجه شده و دارای نتیجه‌ای معنادار بر عملکرد پایگاه داده باشد. به‌منظور کاستن از هزینه سؤال‌ها، ما به دنبال اکتشاف در مرتبه دسترسی به روابط و ترکیباتشان، کاهش‌های رابط‌های از طریق انتخاب‌ها و طرح‌ها، پیش‌پردازش (مرتب‌سازی)، ترتیب تکرار، مرتب‌سازی تقدم عملگر رابطه، و فاکتورهای متعدد دیگر می‌گردیم.

۲-۱۰-۱۰ مدیر ارتباطات

مدیر ارتباطات نقش پلیس راهنمایی و رانندگی را دارد. این سرویس باید انتقال داده‌های پایگاه داده و این اطلاعات وضعیت را برای کمک به پردازش داده‌ها هماهنگ‌سازی نماید. ارتباطات می‌توانند بین سرویس‌های پایگاه داده، پایگاه داده‌های مختلف، پردازنده‌های مختلف، تراکنش‌ها مختلف، یا درون یک تراکنش باشند. مکانیسم‌هایی همچون طرح‌های انتقال پیام ساده، پروتکل‌های کلاینت/سرور، و دیگر موارد پیاده‌سازی شده‌اند.

۲-۱۰-۱۱ مدیر لاگین

مدیر لاگین کار هماهنگ‌سازی انتقال اطلاعات از پایگاه داده فعال به مخزن باثبات ثانویه را برای کمک به بازگشت‌پذیری پایگاه داده و کاهش مؤثر مسئله انتقال زودهنگام اطلاعات از سیستم‌عامل بر عهده دارد. این لاگین سوابق جریان داده‌ها به داخل و خارج پایگاه داده و این اقداماتی که می‌توانند بر وضعیت پایگاه داده تأثیرگذار باشند را حفظ می‌کند. این شامل تراکنش‌ها تصاویر قبلی، تصاویر بعدی، رکوردهای اندو، و رکوردهای ریدو می‌شود.

مدیریت تراکنش

مدیر تراکنش[۵۵] کار فراهم‌سازی یک چارچوب محدود را بر عهده دارد که حول آن تضمین می‌کند که پایگاه داده باثبات می‌ماند و در عین حال عملیات همزمان بر روی پایگاه داده اجرا می‌شوند. مدیر پایگاه داده، بدون همزمانی، این را بدون مشکل تضمین می‌کند – ولی این نه برای مطالعه جذابیت دارد و نه در دنیای واقعی عملی است. دنیای واقعی پردازش پایگاه داده نوعاً با یک پایگاه داده بزرگ با درجه بالایی از چندپردازشی سروکار دارد (تراکنش‌های که در حال حاضر در حال اجرا هستند).

اجرای یک تراکنش شبیه بستن یک قرارداد است، هر دو طرف در قرارداد دخیل هستند، آن‌ها مدتی مذاکره می‌کنند، و بعداً یا به اجماع می‌رسند و قرارداد را امضا می‌کنند، و یا هر برمی‌گردند. ازاین‌رو یک تراکنش (معامله) یا همه چیز است و یا هیچ چیز. یک تراکنش باید یا به طور کامل تکمیل شود و یا اصلاً کامل نشود. این اکنون یک مفهوم است.

یک تراکنش باید به‌عنوان یک واحد باثبات و قابل‌اتکای کار برای سیستم پایگاه داده استفاده شود. یک تراکنش با محیط برنامه‌های کاربردی در تراکنش است و همزمانی پایگاه داده پروتکل‌ها را برای اجرای کارکرد مورد نظر آن کنترل می‌کند (شکل (۲-۳۱) (a)). لازم است که یک تراکنش چهار ویژگی را در زمان اجرا بر روی یک پایگاه داده باثبات تضمین نماید. این چهار خصوصیت ویژگی‌های ACID تراکنش نامیده می‌شوند، آن‌ها دربرگیرنده اجراهای اتمی، باثبات، مستقل و بادوام تراکنش‌ها بر روی پایگاه داده هستند.

یک تراکنش ACID تضمین می‌کند که پایگاه داده‌ای که این تراکنش با آن آغاز می‌شود و پایگاه داده‌ای که با آن خاتمه می‌یابد همخوانی دارند، اینکه این داده‌ها بادوام هستند، اینکه این تراکنش به تنهایی بر روی پایگاه داده عمل نموده، و اینکه این تراکنش به طور کامل اقدامات بر روی پایگاه داده را تکمیل نمود (شکل ۲,۳۱b).

ویژگی‌های ACID تراکنش از قرار ذیل هستند:

[۵۵] transaction manager

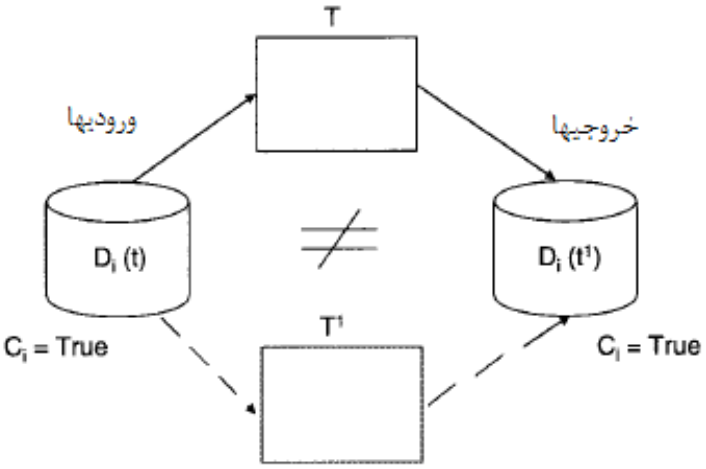
اتمیک – ویژگی اتمی به طور ضمنی اشاره به این دارد که یک تراکنش یک واحد غیرقابل تقسیم اجرائی است که یا به طور کامل کارکرد تعیین‌شده‌اش را به اجرا درمی‌آورد و در غیر این صورت اثر آن بر پایگاه داده به شکلی است که انگار تراکنش هرگز شروع نشده است، بدین معنی که، وضعیت پایگاه داده‌ای که از یک تراکنش خارج می‌شود تراکنشی است که کاملاً متعهد نشده است به همان حالت پایگاه داده‌ای باشد که تراکنش با آن شروع شده است. از سوی دیگر، اگر یک تراکنش اتمی تکمیل شود، حالت پایگاه داده‌ای که ترک می‌کند تمام تغییراتی را داراست که تراکنش محاسبه شده بدون هیچ مورد نصب شده دیگری داراست.

سازگاری (مداومت)[۵۶] – اجرای پیوسته یک تراکنش الزام می‌کند که یک تراکنش یک وضعیت پایگاه داده سازگار و پایدار اولیه را به یک حالت پایگاه داده سازگار و پایدار جدید تغییرشکل دهد. مفهوم اساسی پشت این ویژگی تراکنشی این است که پایگاه داده از مجموعه‌ای از آیتم‌های داده‌ها تشکیل یافته است که، که محدودیت‌هایشان بر رویشان تعریف‌شده است. این پایگاه داده، که در هر مقطع زمانی سازگار لحاظ می‌شود، الزام می‌کند که این محدودیت‌ها بر آیتم‌های داده‌های درون پایگاه داده در حالت درست ارزیابی شوند، بدین معنی که، در صورتی که بخواهیم یک حالت پایگاه داده سازگار داشته باشیم، هیچ یک از این محدودیت‌ها نباید نقض شوند. یک تراکنش معتبر، که در ابتدا پایگاه داده‌ای را می‌بیند که سازگار و پایدار است، باید، به محض تعهد، پایگاه داده‌ای را به جا بگذارد که همچنان سازگار و پایدار است.

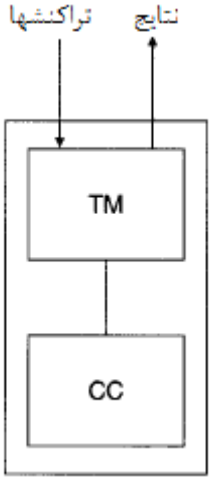
مستقل بودن – استقلال، که گاهی با عنوان ویژگی جداسازی تراکنش‌ها مورد ارجاع قرار می‌گیرد، الزام می‌کند که هر تراکنشی که به داده‌های اشتراکی دسترسی دارد به تنهایی عمل کند، بدون اینکه متأثر از دیگر تراکنش‌ها در حال اجرای همزمان قرار بگیرد. این ویژگی اساساً به این اشاره دارد که اثر یک تراکنش بر پایگاه داده به صورتی است که موجود است، و این به تنهایی، بر روی پایگاه داده اجرا می‌شده است. کارکرد این ویژگی الزام نمودن به حذف هرگونه وابستگی اجرای یک تراکنش به اجرای هر تراکنش دیگری است.

---

[۵۶] Consistent



شکل (۲-۳۱) (a) تراکنش پایگاه داده



شکل (۲-۳۱) (b) تراکنش ACID.

دوام پذیر – دوام‌پذیری اجرای یک تراکنش الزام می‌کند که وقتی که یک تراکنش متعهد شد، اثرات آن در پایگاه داده دائمی بمانند. آنچه این ویژگی متضمن آن است این است که تغییراتی که یک تراکنش در پایگاه داده انجام می‌دهد وقتی که تراکنش خاتمه یافت محو نشوند. داده‌های تولید شده توسط یک تراکنش و نوشته شده در پایگاه داده دائمی می‌شوند. وقتی داده‌ها در پایگاه داده نوشته شدند تنها با تراکنش دیگری که بر روی این داده‌ها خوانده یا نوشته می‌شود قابل تغییر هستند.

این ویژگی‌های تراکنشی باید برای تمام تراکنش‌های که درون یک سیستم مدیریت پایگاه داده اجرا می‌شوند برقرار باشند، اگر قرار بر حفظ همخوانی، صحت و اعتبار پایگاه داده باشد. این ویژگی‌ها باید حتی درزمانی که تراکنش‌ها دیگر به طور همزمان اجرا شوند به‌قرار باشد. علاوه بر این، این ویژگی‌ها، در صورت مورد تبعیت قرار گرفتن یک پایگاه داده درست و همخوان را حتی در مواجهه با شکست‌ها یا خطاها تضمین خواهند نمود. این در زمانی است که ما به‌تدریج به این دید برسیم که خط‌مشی‌ها و مکانیسم‌ها برای اجرای تراکنش و عملیات را می‌توان برای تضمین این ویژگی‌ها توسعه داد که مسائل رخ می‌دهند.

مبانی تراکنش

یک تراکنش مجموعه‌ای از کد برنامه‌های کاربردی و کد دست‌کاری پایگاه داده محدود به یک واحد اجرائی غیر قابل تقسیم است، مثالی از آن در بخش کد ذیل نشان داده‌شده است:

TRANSACTION Name،BEGIN

Applications Code

Code،DB

Applications Code

Code،DB

Code،DB

.

.

Applicatins Code
END TRANSACTION Name

یک تراکنش با شاخص‌های BEGIN TRANSACTION و END TRANSACTION معین می‌شود که مرزهای تراکنش را مشخص می‌کنند – به‌عنوان‌مثال، اگر ما سه رابطه ذیل را داشته باشیم که یک سیستم حفظ خط هوایی را توصیف نمایند:

FLIGHT (Fno, Date, Source, Destination, Seats،Sold, Capacity)
CUSTOMER (Cname, Address, Balance)
FlghtCust (Fno, Date, Cname, Special)

رابطه اول اطلاعات پرواز را نشان می‌دهد – شماره پرواز، تاریخ پرواز، شهر مبدأ، شهر مقصد، تعداد صندلی‌های فروخته شده برای این پرواز، و ظرفیت این هواپیما. رابطه دوم مشتریانی را توصیف می‌کند که سوار بر یک پرواز هستند، در این رابطه نام، نشانی‌ها و توازن‌های تعهد شده بر روی بلیت‌ها را ارائه می‌کند. رابطه سوم رابطه بین پروازها و مشتریان را توصیف می‌کند.این رابطه به طور خاص نشانگر آن است که شماره پرواز چیست، چه مسافرانی در چه تاریخی سوار پرواز بوده‌اند، و هرگونه الزامات ویژه برای این مسافران – به‌عنوان‌مثال، ممکن است فردی یک "غذای شاد" مک‌دونالد یا یک وعده‌غذای گیاهی را بخواهد.

برای تولید یک تراکنش ساده بر روی این روابط پایگاه داده، که ملاحظه‌ای را برای یک مشتری ایجاد می‌کند ما می‌توانیم درخواست یا سؤال (کوئری) "زبان ساختارمند پرسش‌ها" را بنویسیم:

BEGIN TRANSACTION Reservation
BEGIN
Input (FlightNo, date, customer, specl)
EXEX SQL UPDATE FLIGHT
SET Seats،Sold=Seats،Sold+۱
WHERE Fno='FlightNO' AND Date='date';

```
EXEX SQL INSERT INTO FlightCust (FNO, Date, Cname,
Special)
VALUES (FlightNo, date, customer, Specl)
OUTPUT ("Transaction Completed")
End Transaction Reservation;
```

این تراکنش به دنبال ورودی از صفحه کلید برای شماره پرواز، تاریخ پرواز، نام مشتری، و هرگونه الزام ویژه‌ای است که مشتری ممکن است داشته باشد. این‌ها ورودی به متغیرهای تراکنش هستند: به ترتیب، شماره پرواز (.FlightNo)، تاریخ، مشتری، و specl. محتواهای این متغیرها سپس از طریق تابع VALUES در مکان‌های مناسب درون این رابطه قرار داده می‌شوند. ما تعداد صندلی‌های فروخته شده برای این پرواز را با افزایش تدریجی یکی یکی این مقدار و سپس آپدیت کردن مقدار در این رابطه آپدیت می‌کنیم. این تراکنش سپس رابطه FlightCust را با اطلاعات جدید آپدیت می‌کند. برای تکمیل کار ما باید رابطه مشتری را آپدیت کنیم، این به‌عنوان یک تمرین بر عهده خواننده خواهد بود. این نشانگر یک تراکنش ساده است، بااین‌حال، همان‌گونه که مشهود است این به تنهایی تضمین گر ویژگی‌های ACID تراکنش نخواهد بود.

برای تضمین ویژگی‌های ACID تراکنش، ما به چند ویژگی اضافی در این مدل تراکنش ساده نیاز داریم. به‌منظور تحقق نیازهای اجرای اتمی، ما نیاز به ابزاری برای تعیین شرایط برای خاتمه یک تراکنش، اصلاح آن یا دیگر موارد داریم. مفهوم اول موردنیاز برای اجرا و خاتمه مناسب تعهد است. تعهد برای اشاره به خاتمه درست و اتمی یک تراکنش استفاده می‌شود. این دربرگیرنده پردازش ضروری برای تضمین آپدیت کردن مناسب و علامت‌گذاری پایگاه داده است. مفهوم دوم، که بی‌نتیجه گذاشتن[۵۷] نامیده می‌شود، برای تراکنش‌های که به دلیلی دچار مشکل می‌شوند یا اجرایشان متوقف می‌شود ضروری است. شرایط بی‌نتیجه گذاشتن دربرگیرنده عملیات خطادار، تداخل در دسترسی به اطلاعات ذخیره شده، یا ناتوانی در تحقق الزامات ACID برای پردازش تراکنش می‌شوند. یک بی‌نتیجه گذاشتن الزام می‌کند که تمام اثرات یک تراکنش پیش از آنکه هر تراکنش دیگری شانسی برای دیدنشان داشته

[۵۷] abort

باشد از پایگاه داده حذف شوند. این دو ویژگی اضافی برای تسهیل اجرای اتمی ضروری هستند، هرچند نه به طور مجزا.

اقدام تعهدی برای همگام‌سازی اقدامات عناصر دیگر سیستم مدیریت پایگاه داده برای دائمی تغییرات در پایگاه داده ضروری است – به‌عنوان‌مثال، این فرمان را می‌توان برای این استفاده نمود که باعث شود که بافرهای پایگاه داده و لاگ فعالیت به‌صورت اجباری در زیرسیستم ذخیره‌سازی دائمی شوند، و بدین طریق تغییرات بادوام شوند، به صورتی که در بخش کد ذیل نشان داده‌شده است:

```
BEGIN TRANSACTION Reservation
BEGIN
Input (FlightNo, date, customer, specl)
SELECT Seat،Sold, Capacity  FROM FLIGHT;
IF Seat،Sold > Capacity THEN
BEGIN
EXEX SQL UPDATE FLIGHT SET  Seats،Sold=Seats،Sold+١
WHERE Fno='FlightNO'
AND Date='date';
EXEX SQL INSERT INTO FlightCust (FNO, Date, Cname,
Special)
VALUES (FlightNo, date, customer, Specl)
OUTPUT ("Transaction Completed")
COMMIT Reservation;
ELSEABORT Reservation;
END;
End Transaction Reservation;
```

این تراکنش هشدار دهی شده اکنون به ما این امکان را می‌دهد که به تراکنش در صورتی که شانسی برای موفقیت داشته باشد ادامه دهیم، یا تراکنش را ناتمام رها کنیم. در این مثال، ما می‌توانیم در صورتی

که هیچ صندلی در هواپیما برایمان باقی نمانده باشد که به مشتری بدهیم تراکنش را نیمه‌تمام رها کنیم. اگر صندلی وجود داشته باشد، یک صندلی به این مشتری می‌فروشیم و معامله را متعهد می‌کنیم.

رسمی‌سازی تراکنش

یک تراکنش، T_i، از مجموعه‌ای از عملیات تشکیل یافته است، $O_j \in \{Read, Write\}$، که در آن O_j یک عملیات از یک تراکنش i بر روی آیتم‌های داده‌ها از پایگاه داده D است.

فرض کنید که $O_{si} = \cup O_{ij}$ نشانگر یکی بودن مجموعه تمام عملیات j از یک تراکنش i باشد.

نهایتاً اینکه، فرض کنید که $N_i \in \{Abort, Commit\}$ نشانگر مجموعه شرایط خاتمه‌ای بر روی یک تراکنش، چه تعهد و چه بی‌نتیجه رها کردن، باشد.

یک تراکنش به‌صورت یک مرتب‌سازی جزئی بر روی عملیات و شرایط انتهائی‌اش مدل‌سازی می‌شود. مرتب‌سازی جزئی توسط P نشان داده می‌شود که نشانگر آن است که ترتیب جزئی P از مجموعه-ای از عملیات تشکیل یافته است، که با S نشان داده می‌شود، و یک رابطه مرتب‌سازی که بین عناصر در S برقرار است و با $\ll$ نشان داده می‌شود.

با این تعاریف ما می‌توانیم به طور رسمی یک تراکنش، T_i ، را به‌صورت یک مرتب‌سازی جزئی از عملیات ترکیبی، به‌صورت ذیل، توصیف نماییم:

$$T_i = \{S_i \ll i\} \qquad (۱۵-۲)$$

که در آن:

$$۱- \quad S_i = OS_i \cup N_i \qquad (۱۶-۲)$$

$$۲- \quad For\ any\ two\ operations\ from\ T_i = O_{ij}, O_{ik} \in O_{si}$$
$$If\ O_{ij} = R(X) and\ O_{ik} = W(X), then\ for\ any\ X \qquad (۱۷-۲)$$
$$Either\ O_{ij} \ll i\ O_{ik}\ \ or\ \ O_{ik} \ll i\ O_{ij}$$

$$۳.\ And\ for\ all\ O_{ij} \in OS_i, O_{ij} \ll i\ N_i \qquad (۱۸-۲)$$

تمام آنچه از این برداشت می‌شود این است که یک تراکنش از خواندن‌ها، نوشتن‌ها، و یک تعهد یا یک عملیات بی‌نتیجه رها کردن می‌شود، و یک مرتب‌سازی صریح در یک تراکنش وجود دارد که تا در

صورتی که یک خواندن متنازع مقدم بر یک خواندن متنازع در تاریخ قرار بگیرد، یک مرتب‌سازی متوالی دقیق همیشه باید در این تراکنش برای این عملیات متنازع برقرار باشد. علاوه بر این، تمام عملیات تراکنش‌ها باید مقدم بر عبارات تعهد یا بی نتیجه‌گذاری قرار بگیرند. این مفهومی مهم برای توسعه معیارهای صحت برای اجرای تراکنش‌ها است، به ویژه در زمانی که همزمانی وارد عمل شود. مرتب‌سازی تراکنش نباید نقض شود، تا اطمینان حاصل شود که تراکنش می‌تواند عملیات مورد نظر را به انجام در بیاورد.

پردازش تراکنش در یک سیستم پایگاه داده تلاش‌های گسترده‌ای برای تضمین ویژگی‌های ACID انجام می‌دهد، و در عین حال درجه بالایی از دسترسی داده‌ها را در اختیار قرار می‌دهد، که کمتر از آپدیت‌ها، اجتناب از ناتمام گذاشتن آبشاری، و قابلیت بازیابی پایگاه داده و تراکنش‌ها نیست. درجه بالایی از دسترسی‌پذیری داده‌ها از طریق کاهش مسدودسازی درخواست‌های خواندن و نوشتن محقق می‌شود. چیزی کمتر از آپدیت‌ها با پردازش تعهدی درست تضمین نمی‌شود. اجتناب از بی نتیجه رها کردن‌های آبشاری از طریق پروتکل‌های بازیابی قوی فراهم‌سازی می‌شود. نهایتاً اینکه، بازیابی از طریق افزونگی و قواعدی امکان‌پذیر می‌گردد که بر تعهد حاکم هستند.

۲-۱۰-۱۲ عدم تطابق پایگاه داده و سیستم

سیستم‌عامل مخزن را، بر مبنای چشم‌انداز سیستم‌عامل درباره اینکه این چه زمانی باید انجام شود، از حافظه اولیه به حافظه ثانویه کوچ می‌دهد. صفحه‌بندی تقاضا و ذخیره‌سازی محدود الزام می‌کنند که این بر یک مبنای خطای صفحه انجام شود. بااین‌حال، ممکن است پایگاه داده، به دلایل کنترل همزمانی و مسائل اتمیسیته، مایل به این نباشد که صفحه در حافظه ثانویه بازنویسی شود. پایگاه داده ممکن است تمایل به حفظ صفحات در حافظه داشته باشد تا زمانی که تراکنش زمان را متعهد نماید و سپس وارد مخزن ثانویه شود. این ممکن است به پایگاه داده کمک کند که نیازی به برگرداندن تراکنش‌ها در زمان خرابی نداشته باشد، صرفاً کار را بی نتیجه رها کند، و ریستارت کند.

مدیریت ۱۱۰ و مدیریت دستگاه به این مرتبط است. این پایگاه داده ممکن است تمایل داشته باشد که دسترسی را بر مبنای کوئریهای ارائه شده به آن مرتب‌سازی کند تا اجرای ACID را حفظ کند، در حالی که سیستم‌عامل به‌سادگی دسترسی‌ها را مرتب‌سازی خواهد نمود تا بیشترین بازدهی داده‌ها را به

CPU برگرداند. ترتیبی که در آن اطلاعات را برمی‌گرداند می‌تواند برای پایگاه داده غیر سازنده باشد، تا نقطه‌ای که در آن پایگاه داده تا جایی برای داده‌های موردنیاز انتظار کشیده باشد که داده‌ها صفحات سیستم‌عامل را از نرم‌افزار پایگاه داده خارج نموده باشند تا جا برای داده‌ها باز شود، یا داده‌هایی که این اطلاعات جدید باید در قبالشان پردازش شود را حذف نماید. در هر حالت، این منجر به پردازش بهینه پایگاه داده نمی‌شود.

مشکلی که با سیستم‌عامل برای این نوع از مسئله وجود دارد مکانیسم‌ها و خط مشی‌های مدیریت بافر I/O است. پایگاه داده می‌خواهد که بافرها را مورداستفاده قرار داده و بهینه‌سازی کند تا بازدهی تراکنش را حداکثرسازی نماید، در حالی که سیستم‌عامل خواهان حداکثرسازی پاسخ پردازش متوسط است.

کنترل خود پردازنده توسط سیستم‌عامل می‌تواند مانع از کارکردهای اساسی شود که پایگاه داده باید به اجرا در بیاورد – به‌عنوان‌مثال، این پایگاه داده الزام می‌کند که مجموعه کارکردهای پایگاه داده در نقاط مشخص و به شکلی بدون وقفه به مخزن ثانویه انتقال داده شود. به همین صورت، همخوان نگه داشتن هر چه بیشتر پایگاه داده نیازمند آن است که پایگاه داده در زمان لازم و در یک عملیات اتمی داده‌های متعهد شده را به مخزن همخوان انتقال دهد. سیستم‌عامل در تمایلش به رفتار منصفانه می‌توان یک کارکرد پایگاه داده را به طور خاص با اجرای این عملیات منقضی نماید. در یک مسئله مرتبط دیگر، اگر یک پایگاه داده دو فایل داده‌های بزرگ را در قبال یکدیگر مرتب‌سازی و پردازش کند، ممکن است مایل به حفظ کنترل مستقیم بر این باشد که چگونه و چه زمانی مرزها را از مخزن به پردازنده می‌پیماید و بر می‌گردد. بدون کنترل مستقیم بر مکانیسم‌های تخصیص و رهاسازی، پایگاه داده می‌تواند از یک منبع انتقال داده شود در حالی که هنوز دیگری را دارد، و موجب از دست رفتن تداوم عملیات مورد نظر شود.

مکانیسم قفل‌سازی سیستم‌عامل برای مدیریت فایل ساده خوب عمل می‌کند، و این برای اکثر برنامه‌های کاربردی کفایت می‌کند ولی یک پایگاه داده نیاز به کنترل بهتری بر قفل‌سازی دارد تا امکان قفل‌سازی در احتمالاً فقط یک سطح آیتم داده‌ها باقی بماند. دلیل آن فراهم‌سازی امکان همزمانی بیشتر و مسدودسازی کمتر داده‌ها است. هدف ما افزایش دسترسی‌پذیری داده‌ها با صرفاً قفل‌سازی آنچه در حال استفاده است، و نه کل یک فایل، است. برای یک طرفه کردن، این پایگاه داده‌ها مجبور به استفاده از

آدرس‌دهی مستقیم ویژگی‌های مدیریت فایل مستقیم هستند تا اجازه کنترلشان بر سطح فایل قفل‌سازی داده شود. بااین‌حال، در برخی از سیستم‌های عامل، پایگاه داده همچنان تحت کنترل مدیر قفل سیستم‌عامل، فارغ از حالتی که در حال استفاده است، دچار مشکل است.

یک مکانیسم ارتباط بین فرایندی سیستم‌عامل ممکن است به‌مراتب برای استفاده شدن درون یک سیستم‌عامل بیش از حد گران تمام شود. بسیاری از سیستم‌های عامل از شکلی از انتقال پیام دربرگیرنده پردازش وقفه استفاده می‌کنند. چنان مکانیسم‌هایی ممکن است هزینه بالایی از جنبه بالاسری داشته باشند. یک پایگاه داده ممکن است تمایل به ارائه مکانیسم‌های IPC ساده‌ای با استفاده از حافظه یا سفامورهای اشتراکی داشته باشد، به ویژه از این جهت که یک پایگاه داده تنها یک فرایند دیگر درون سیستم‌عامل است.

زمان‌بندی در یک سیستم‌عامل به دنبال حداکثرسازی زمان پاسخ متوسط کلی و اشتراک‌گذاری مناسب منابع باشد. زمان‌بندی تنها با انتخاب یک فرایند برای قرار دادن بر روی یک سخت‌افزار اجرائی سروکار دارد. از سوی دیگر، یک پایگاه داده دارای یک مسئله زمان‌بندی چند سطحی است —نه‌تنها باید انتخاب کند که چه تراکنشی را در هر مقطع زمانی وارد سرویس نماید، بلکه این باید زمان‌بندی کند که چه عملیاتی را بر روی پایگاه داده اساسی به اجرا در بیاورد تا الزامات کنترل همزمانی محقق شوند. زمان‌بند یک سیستم‌عامل چنان سرویسی را فراهم نخواهد آورد و نمی‌آورد.

یک پایگاه داده استفاده از سرویس‌های کپی کردن، بکاپ، و بازیابی زیرساخت اساسی را برای کمک به ساخت پروتکل‌های بازیابی پایگاه داده الزامی می‌کند. مشکل اینجاست که بسیاری از عملیات دیگر یک سیستم‌عامل ممکن است وارد عمل شوند و مانع از عملکرد آسان بازیابی پایگاه داده شوند. این پایگاه داده مایل است تا دیکته نماید که چگونه و چه زمانی اطلاعات را به مخزن باثبات و همخوان انتقال خواهد داد. این به‌منظور حداقل سازی کاری (UNDO و REDO) انجام می‌شود که باید برای بازیابی پایگاه داده به یک حالت همخوان معلوم انجام شود. سیستم‌عامل، از سوی دیگر، این کار را بر مبنای نیازهای خودش برای تخصیص مجدد مخزن برای فرایندهای در حال اجرا انجام خواهد داد. سیستم‌عامل این‌گونه در نظر نمی‌گیرد که این صفحه که کمترین استفاده از آن شده است در واقع صفحه بعدی خواهد بود که توسط پایگاه داده استفاده می‌شود. بلکه صرفاً این صفحه را انتخاب می‌کند و بلافاصله آن را، بر مبنای نیازهایش، به بیرون می‌راند.

برای تطابق‌پذیرتر کردن سیستم‌عامل و رابط پایگاه داده، مطلوب است که سیستم‌عامل از اطلاعات معنایی استفاده کند، که توسط پایگاه داده برای ارائه تصمیمات مناسب و آگاهانه قابل ارائه هستند. منظور این نیست که پایگاه داده باید حرکت‌های سیستم‌عامل را بر عهده گرفته و دیکته کند. بلکه در عوض باید به شکلی همکارانه عمل نماید تا نیازهای سیستم محور یک پایگاه داده حداکثرسازی شود، که مواریانستر از نیازهای یک برنامه کاربردی معمول هستند. برای کسب اطلاعات بیشتر درباره سیستم‌های پایگاه داده به منبع [۱] مراجعه کنید.

۲-۱۱ خلاصه

یک سیستم کامپیوتری از عناصر بسیاری تشکیل یافته است. این‌ها در ابتدا شامل واحد پردازشگر، واحد حافظه، واحد مخزن ثانویه، واحد ورودی و خروجی، و سخت‌افزار در کنار هم می‌شوند. هر یک از این عناصر را می‌توان به شیوه‌های مختلفی معماری نمود، که هر یک نقاط ضعف و قوت خود را دارند. سیستم‌های کامپیوتری نوعاً به‌صورت واحدهای پردازنده واحد، واحدهای پردازنده چندگانه، واحدهای پردازنده توزیع‌شده، یا واحدهای شبکه‌ای شده ارائه می‌شوند. خط مشی‌های ارتباط دهی این دستگاه‌ها برای تحقق نیازهای یک برنامه کاربردی شکل نهایی معماری سیستم را تعیین خواهند کرد.

سیستم‌عامل و سرویس‌های زیرساخت پشتیبانی مرتبط توسط یک برنامه کاربردی برای سازماندهی، نگهداری و دست‌کاری اطلاعات بر روی یک معماری کامپیوتر مشخص استفاده می‌شوند. نیازها و اولویت‌های برنامه‌های کاربردی و سیستم‌عامل همیشه منطبق نیستند. به علت این عدم تطابق امپدانس، برنامه‌های کاربردی، درگذشته، تلاش نموده‌اند که به‌جای کار با آن حول سیستم‌عامل کار کنند. بدنام‌ترین آن‌ها در روزهای آغازین سیستم‌ها عامل DOS و IBM PC بود. برنامه‌های کاربردی نوعاً سیستم‌عامل را دور زدند و مستقیماً بر روی سخت‌افزار اساسی کارکردند. نتیجه آن این بود که برنامه‌ها نوعاً بر روی ماشین‌های "IBM PC یا ٪۱۰۰ منطبق" اجرا می‌شدند. یک مثال دیگر این است که مدیریت سیستم‌عامل و سلسله‌مراتب حافظه ممکن است برای تطابق با نیازهای سیستم مدیریت پایگاه داده منصفانه و به شکلی معقولانه بهینه باشد. سیستم‌عامل تلاش می‌کند تا مجموعه معقولانه‌ای از صفحات داده‌ها را در حافظه برای استفاده برنامه‌های کاربردی نگه دارد، ولی تلاش نمی‌کند که از

معیارهای کارایی خودش فراتر برود. مفهوم مطرح در اینجا این است که سیستم‌ها به شکلی مهندسی شوند که، بر مبنای نیازهای معنایی و هدف برنامه‌های کاربردی، به شکلی بهینه، بر مبنای نیازهای معنایی و هدف برنامه‌های کاربردی، فعالیت نمایند، که ممکن است برخلاف زمان پاسخ متوسط سیستم‌عامل و اهداف منصفانه باشد.

فصل سوم

مفاهیم بنیادین و معیارهای کارایی

۳-۱ مقدمه

معماران و طراحان سیستم‌های کامپیوتری به دنبال پیکربندی‌های عناصر سیستم‌های کامپیوتری می‌گردند تا عملکرد سیستم منطبق با معیارهای مطلوب باشد. معنای آن این است که سیستم‌عامل کیفیت سرویسی را ارائه می‌کند که منطبق با نیازهای برنامه‌های کاربردی کاربری است. ولی معیارهای این کیفیت سرویس و انتظار کارایی بسته به نوع کاربری شما متغیر هستند. در وسیع‌ترین حوزه، ما ممکن است به دنبال زمان متوسط پاسخ کاربر، سهولت کاربری، پایایی، تحمل خطا، و دیگر کمیت‌های کارایی این‌چنینی باشیم. مشکلی که با برخی از این موارد وجود دارد این است که آن‌ها معیارهای کمی در مقایسه با کیفی هستند. برای اینکه در مطالعات عملکرد سیستم‌های کامپیوتری‌مان را علمی و دقیق عمل کنیم، باید بر معیارهای کیفیت یا کمی قابل‌سنجش یک سیستم تحت مطالعه تمرکز نماییم.

گزینه‌های ممکن بسیاری برای سنجش عملکرد وجود دارند، ولی بیشتر آن‌ها در یکی از این دو طبقه-بندی جای می‌گیرند: معیارهای سیستم‌محور یا کاربرمحور. معیارهای سیستم‌محور نوعاً حول مفاهیم بازدهی و کارایی تعریف می‌شوند. بازدهی به‌عنوان متوسط تعداد آیتم‌های (مثلاً تراکنش‌ها، فرایندها، مشتریان، مشاغل، و غیره) پردازش شده به ازاء واحد زمان سنجیده شده تعریف می‌شود. بازدهی وقتی معنادار است که ما اطلاعاتی درباره ظرفیتی نهاد مورد سنجش قرار گرفته و بار کاری ارائه شده آیتم‌ها در نهاد در دوره زمانی تعیین‌شده بدانیم. ما می‌توانیم از معیارهای بازدهی برای تعیین ظرفیت سیستم‌ها،

با مشاهده اینکه چه زمانی تعداد آیتم‌های در انتظار هرگز صفر نمی‌شود و تعیین اینکه در چه سطحی بر مبنای بار کاری ارائه شده سیستم آیتم‌ها هرگز منتظر نمی‌مانند، استفاده کنیم. بهره‌برداری یک معیار کسری از زمان است که یک منبع مشخص مشغول است. یک مثال آن بهره‌برداری از CPU است. این می‌تواند زمان بیکار ماندن CPU و زمان عملکرد آن را برای اجرای برنامه ارائه شده مورد سنجش قرار دهد.

معیارهای کارایی کاربرمحور نوعاً دربرگیرنده زمان پاسخ یا زمان انجام کار[۵۸] هستند. زمان پاسخ و زمان انجام کار به دیدی از زمان باقیمانده سیستم از نقطه‌ای ارجاع دارند که یک کاربر یا برنامه‌های کاربردی یک کار را بر روی سیستم شروع می‌کند و زمانی که پاسخ کار به کاربر برگردانده می‌شود. می‌توان به راحتی از این تعریف دید که این‌ها معیارهای روشن و نامبهمی نیستند، زیرا متغیرهای بسیاری در آن‌ها دخیلند. به‌عنوان‌مثال ترافیک کانال ۱۱۰ می‌تواند موجب تغییراتی در این معیار برای یک کار واحد شود، همان‌گونه که برای بار سیستم‌های عامل یا بارهای CPU برقرار است. ازاین‌رو، ضروری است که اگر قرار بر استفاده از این معیار باشد، مدل کننده عملکرد باید در تعریفش از معنای این معیار فاقد ابهام باشد. این معیارهای کاربری تصادفی لحاظ می‌شوند، و، ازاین‌رو، معیارهای آن‌ها نوعاً از جنبه مقادیر مورد انتظار یا متوسط و اینی انواع مختلفی از این مقادیر موردبحث قرار می‌گیرند.

بااین‌حال، در این موارد، برای انجام چنین سنجش‌هایی ما نیاز به یک درک اساسی از محیط و پارامترهایی از آن که با آن‌ها کار می‌کنیم داریم. یک مفهوم اساسی به زمان مربوط می‌شود. برای سنجش یک پدیده فیزیکی، ما به یک معیار برای سنجش در قبال آن نیاز داریم. در سیستم‌های کامپیوتری این معیار معمولاً زمان است. بااین‌حال، زمان به تنهایی کفایت نمی‌کند، ما باید جایی داشته باشیم که زمان را از آن معین نماییم. این نقطه گاهی نشأت گرفته از رویدادی در سیستم است که باید مورد سنجش قرار بگیرد یا صرفاً یک زمان مشخص‌شده است. به‌عنوان‌مثال، در یک سیستم کامپیوتری ممکن است ما قصد سنجش زمانی را داشته باشیم که یک تراکنش برای اجرا در یک سیستم پایگاه داده به آن نیاز دارد. ما باید رویدادهای مورد علاقه را برای این سیستم تراکنش تعریف کنیم – به‌عنوان‌مثال، شروع تراکنش، اجرای تراکنش، و خاتمه دادن یا شروع تراکنش. با توجه به اینکه ما زمان و رخدادها

[۵۸] turnaround time

را داریم، در ادامه لازم است که تعریف کنیم که چه زمانی و چگونه این رویدادها و بازه‌های مورد نظر برای این رویدادها را مورد سنجش قرار دهیم.

دیگر مفاهیم اساسی موردنیاز برای بحث درباره عملکرد سیستم‌های کامپیوتری دربرگیرنده ابزارهایی می‌شوند که به وسیله آن‌ها از یک سیستم نمونه‌برداری می‌شود و مورد سنجش قرار می‌گیرد. سنجش‌ها می‌توانند درون یک پروژه ارزیابی شکل‌های بسیاری به خود بگیرند، که در ادامه دیده خواهد شد. یک جنبه دیگر زمان، که در مطالعات عملکرد سیستم‌های کامپیوتری مهم است، به بازه‌ها مربوط می‌شود. یک بازه فاصله سنجیده شده را نمایندگی می‌کند که یک‌فاصله اندازه‌گیری شده را در یک دوره زمانی نشان می‌دهد. به‌عنوان‌مثال، یک روز، هفته یا ماه نشانگر بازه‌های زمانی است. مهم‌ترین چیز در ارزیابی سیستم‌های کامپیوتری مفهوم پاسخ است. پاسخ نشانگر یک رویداد تکمیل برای یک نهاد مورد سنجش قرار گرفته است – به‌عنوان‌مثال، زمان بین زدن یک کلید بر روی پایانه یک کامپیوتر و نتیجه گرفتن توسط کاربر.

به‌منظور بهره‌برداری از کمیت‌های اساسی زمان، رویدادها، بازه‌ها و پاسخ، ما نیاز به یک مفهوم اضافی در رابطه با روابط بین تمام این آیتم‌ها داریم. دغدغه‌های نوعی که ما باید برای آن‌ها چاره‌جویی کنیم به مفاهیم استقلال و تصادفی بودن مربوط می‌شود که با آیتم‌های یک سیستم کامپیوتری ارتباط دارند. نکته آخر، اما مهم، اینکه مفهوم یک‌بار کاری و رابطهای که این با پروژه مدل‌سازی دارد باید تعریف شود.

۳-۲ زمان

زمان اساسی‌ترین مفهوم موردنیاز برای تحلیل عملکرد سیستم‌های کامپیوتری است. بدون مفهوم روشنی از زمان مطالعات کارایی ما نمی‌توانند کیفیت‌های کمی به خود بگیرند. وقتی که عملکرد یک سیستم کامپیوتری موردبررسی بگیرد زمان به‌عنوان یک کمیت به چند طریق نمود می‌یابد. به‌عنوان‌مثال، ما درباره مفاهیمی همچون زمان ورود یک نهاد، زمان سرویس برای یک نهاد، زمان بین خرابی‌ها، زمان لازم برای تعمیر، طول عمر نهاد، و چند کمیت دیگر ارتباط‌دهنده زمان با عملکرد سیستم‌های کامپیوتری می‌شنویم. هر یک از این کمیت‌ها ما را ملزم به برخورداری از یک نقطه مرجع می‌کند که معنایشان را از آن تعیین کنیم.

در عملکرد سیستم‌های کامپیوتری، ما به سنجش زمان مرتبط با رخدادهای عملیاتی مختلف در سیستم کامپیوتری تحت مطالعه علاقه‌مند خواهیم بود. این رخدادها با تمبر زمانی علامت‌گذاری خواهند شد، و با استفاده از این تمبر زمانی ما قابلیتی برای تعیین مرتب‌سازی نسبی این رخدادها به نسبت یکدیگر خواهیم داشت. تمبر زمانی یک رویداد، E، را می‌توان به‌صورت $E(t)$ نشان داد. سنجش علامت‌گذاری زمان، t، فقط به خوبی ساعتی که ما در نمایش زمان یک رویداد استفاده می‌کنیم و توانائی‌مان برای تطابق نمایش زمانی با رویداد خواهد بود.

زمان در یک سیستم واقعی به دو طریق عمده نمایش داده می‌شود: یا به‌صورت یک دنباله و یا به‌صورت بازه‌ها یا گام‌های گسسته. بهترین راه لحاظ نمودن این دو معیار زمانی این است که زمان گسسته یک نمونه واحد از معیارهای ساعت زمانی را نمایندگی می‌کند، در حالی که بازه‌ها به صفر میل می‌کنند یا بینهایت کوچک می‌شوند.

سیستم‌های کامپیوتری با استفاده از مفهوم بازه‌های زمانی ثابت عمل می‌کنند. این بازه‌ها چارچوب یا محدودیت زمانی را نمایندگی می‌کنند که ساعت یک سیستم کامپیوتری در یک ثانیه انجام می‌دهد. زمان‌های سیکل یا ساعت‌های سیستم‌های کامپیوتری معمول بر حسب نانوثانیه ($۱۰^{-۹}$ ثانیه) یا در برش‌های حدود یک میلیاردم ثانیه سنجیده می‌شوند. چنان درجه‌بندی‌های ظریف یک ثانیه به ما برای درک سرعت سیستم‌های کامپیوتری و اجزاء مرتبط کمک می‌کند.

اطمینان دارم که همه شما چیزهایی درباره سرعت یک پردازنده شنیده‌اید. وقتی ما قصد خرید یک کامپیوتر شخصی جدید را داریم، چند معیار آن به نسبت زمان در اختیارمان قرار داده می‌شود. به‌عنوان‌مثال، پردازنده ۱٫۵GHz به طور ضمنی به این اشاره دارد که پردازنده یک سیکل کلاک حدود ۰٫۶۷ نانوثانیه، یا $۶٫۶T^{۱۰}$ ثانیه، دارد. این تنها معیاری نیست که در زمان انتخاب یک کامپیوتر شخصی برای خرید باید بداند. سرعت CPU مهم است ولی تنها یکی از معیارهاست. ما نیاز داریم که بدانیم سرعت انتقال داده‌ها و دستورالعمل‌ها از دستگاه‌های خارجی به حافظه اولیه داخلی چقدر است، و این اینکه حافظه اولیه با چه سرعتی می‌تواند این اطلاعات و دستورالعمل‌ها را برای اجرای واقعی به پردازنده انتقال دهد. اگرچه سرعت CPU ذکر می‌شود، ولی این معیار واقعی برای عملکرد ماشین نیست. ما شیوه پیوستگی دستگاه‌ها و نحوه تأثیرگذاری تراکنش آن‌ها بر سرعت کلی را موردبررسی قرار خواهیم داد. در این مثال ساده، کندترین دستگاه در سیستم نهایتاً سرعت واقعی را دیکته خواهد کرد.

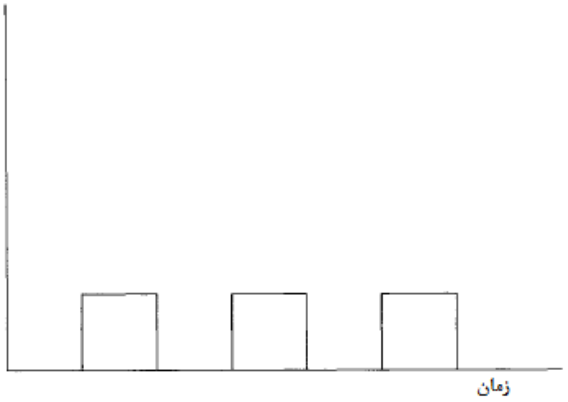
۳-۳ رویدادها

زمان یک معیار مهم است، ولی زمان تنها وقتی برای ما می‌تواند مفید واقع شود که ما ابزاری برای استفاده از آن در سنجش چیزی درون سیستم کامپیوتری تحت ارزیابی‌مان داشته باشیم. یک رویداد یک نهاد مورد علاقه را در سیستممان توصیف می‌کند. رویدادها معمولاً نشانگر یک اقدام هستند، به‌عنوان‌مثال، آغاز یک سیکل کلاک (شکل ۳-۱) یا انتهای یک سیکل کلاک. آغاز سیکل اجرای دستورالعمل یک کامپیوتر یک رویداد دیگر است، و رویدادهای دیگر شامل انتهای سیکل، خواندن یک مکان حافظه، آغاز انتقال یک بلوک داده‌ها از یک دستگاه ذخیره‌سازی ثانویه، و شروع یک فرایند یا وظیفه می‌شوند. تمام این‌ها نشانگر رویدادها مورد توجه برای مهندس کامپیوتر یا معمار کامپیوتر هستند.

رویدادها، که نشانگر اقدامات درون سیستم کامپیوتری ما هستند، همه باید کنترل‌شده باشند، تا توالی بندی یا مرتب‌سازی (مرتب‌سازی جزئی یا کلی) این اقدامات در تحقق یک رویداد بزرگ‌تر نقش‌آفرین باشد. به‌عنوان‌مثال، این سیکل کلاک کامپیوتری ساده برای نشان دادن آغاز اجرای یک دستورالعمل در یک کامپیوتر استفاده می‌شود. لبه بالارونده کلاک توسط پردازنده برای شروع اجرای دستورالعمل فعلی در ثبات دستورالعمل و آماده‌سازی دستورالعمل بعدی برای اجرا استفاده می‌شود. رویدادهای متعدد موازی در حال اجرا در طی هر سیکل کلاک از کلاک سیستم‌عامل باید به شکلی همگام‌سازی شوند که نیات طراحی شده محقق شوند. به‌عنوان‌مثال، دستورالعملی که باید به اجرا در می‌آمد در طی توالی آخر در ثبات دستورالعمل بارگیری می‌شد، و در عین حال آدرس دستورالعمل بعدی محاسبه می‌شد و احتمالاً برخی پارامترها برای دستورالعمل در محل قرار می‌گرفتند. هر اقدام باید طراحی شود و توالی بندی آن به نسبت اقدامات دیگر تعریف شود تا کامپیوتر به‌صورت مورد نظر عمل نماید.

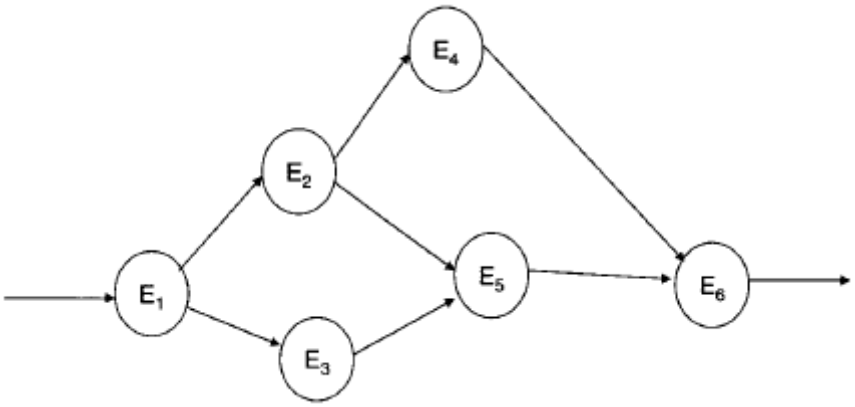
هر اقدام ساده، از تیکهای ساعت تا کنش‌های پیچیده‌تری همچون اجرای یک دستورالعمل همه به بخشی از کنش‌های سیستم‌های بزرگ‌تر تبدیل می‌شوند – به‌عنوان‌مثال، آغاز یک انتقال حافظه مستقیم داده‌ها از یک دیسک درایو مخزن ثانویه، انتقال DMA به‌عنوان بخشی از الگوریتم‌های صفحه،بندی سیستم مدیریت حافظه سیستم‌ها مورداستفاده قرار می‌گیرد، و رابطه الگوریتم صفحه‌بندی با حرکت یک فرایند که به طور فعالانه بر روی DMA اجرا می‌شود به علت یک سوئیچ فرایند هندل شده توسط سیستم‌عامل با یک مورد دیگر جایگزین می‌شود. همه این‌ها اقدامات مورد توجه را برای تحلیلگر کامپیوتر نمایندگی می‌کنند. بااین‌حال هر یک از آن‌ها دارای یک رابطه زمانی متفاوت با معیار زمانی هستند. سیکل کلاک

به‌صورت کسری از نانوثانیه، انتقال حافظه اصلی در محدوده ۱۰۰ نانوثانیه، انتقال دیسک در محدوده میلی‌ثانیه، انتقال فایل‌های سیستم‌های عامل در محدوده ده‌ها میلی‌ثانیه و امثالهم مورد سنجش قرار می‌گیرد.



شکل (۳-۱): نمونه‌ای از یک کلاک کامپیوتر

از جنبه ارزیابی عملکرد، تحلیلگر سیستم باید درکی از درون سیستم تحت مطالعه و رابطه این رویدادها در میان یکدیگر داشته باشد. به‌عنوان‌مثال، ما باید بدانیم که یک رویداد دسترسی فایل از رویداد دسترسی دیسک، رویداد الگوریتم جایگزینی صفحه حافظه، و بار حافظه اصلی و رویدادهای مخزن تشکیل یافته است. علاوه بر اطلاع از رویدادهای دخیل در یک رویداد مرتبه بالاتر، مرتب‌سازی رویداد نیز باید درک شود. به‌عنوان‌مثال، دانستن این مهم است که الگوریتم جایگزینی صفحه باید ابتدا مورد دسترسی قرار بگیرد، تا مشخص شود که، پیش از اینکه صفحه جدیدی در حافظه اولیه بارگیری شود، چه صفحه‌ای باید انتقال یابد. این مرتب‌سازی رویداد را می‌توان با لیست رویدادهای ساده یا با مرتب‌سازی‌های جزئی پیچیده‌تر نشان داد (شکل ۳-۲). این مرتب‌سازی الزام می‌کند که چه رویدادهایی باید در نظر گرفته شوند و چگونه ممکن است لازم باشد که رویدادها مورد سنجش قرار بگیرند، تا تصویری دقیق از عملکرد سیستم را بتوان تعیین نمود.



شکل (۳-۲): مرتب‌سازی جزئی رویداد

۳-۴ سنجش‌ها (نمونه‌برداری)

چگونه می‌توان عملکرد یک سیستم یا جزء را مورد سنجش قرار داد؟ این مسئله اصلی است که تحلیلگر عملکرد سیستم‌های کامپیوتری با آن مواجه است. تحلیلگر، برای اینکه نحوه سنجش، زمان سنجش، یا چیزی که باید مورد سنجش قرار بگیرد را تعیین نماید، باید ابتدا تمام رویدادهای مورد علاقه در سیستم و رابط‌های که این رویدادها با یکدیگر دارند را بداند. این رویدادها، همان‌گونه که قبلاً دیدیم، تمام کنش‌های واقعی را نمایندگی می‌کنند که در سیستم کامپیوتری تحت مطالعه رخ می‌دهند. این رویدادها سلسله مراتبی از روابط را شکل می‌دهند، که در آن رویدادهای دانه بندی شده ظریف‌تر برای ساخت رویدادهای دانه درشت‌تر در سیستم استفاده می‌شوند. بااین‌حال، حتی با این تعاریف، ما اطلاعات کافی برای شروع سنجش‌هایی نداریم که دارای معناهایی هستند. ما باید تمام شرایط محتملی که برای رویدادها در سیستممان برقرار هستند و زمان معتبر بودن احتمالی‌شان را بدانیم. با داشتن مجموعه‌ای از رویدادهای محتمل و مقادیرشان، ما می‌توانیم یک "وضعیت[۵۹]" معتبر را برای سیستم کامپیوتری تحت مطالعه توصیف کنیم.

[۵۹] state

وضعیت یک جنبه مهم در زمان لحاظ کردن هر سیستم کامپیوتری است. وضعیت، S، به‌صورت مجموعه‌ای از تمام رویدادها در سیستم ما در کنار مقادیر معتبر برای شرایطشان در حالت تعریف‌شده تعریف می‌شود. این را می‌توان به‌صورت ذیل توصیف نمود:

$$S = \{E_1(value), E_2(value), E_3(value), \dots, E_n(value)\} \quad (۱-۳)$$

که در آن هر یک از رویدادها باید تمام رویدادهای مؤلفه معتبر را داشته باشند، و مقادیرشان باید حالت‌های معتبر را تعریف کند. به‌عنوان‌مثال، یک وضعیت برای یک واحد پردازنده مرکزی را می‌توان متشکل از رویدادها و مقادیر ذیل تعریف نمود:

- آدرس شمارنده برنامه موجود در شمارنده برنامه
- دستورالعمل موجود در ثبات دستورالعمل
- وضعیت و مقدار ثبات شاخص
- وضعیت و مقادیر در ثبات کنترل وضعیت
- مقدار موجود در ثبات‌های موقتی واحد منطقی حسابی
- مقدار موجود در گذرگاه داده[۶۰]
- مقدار موجود در گذرگاه آدرس[۶۱]

وقتی ما تعاریفی برای رویدادها وضعیت سیستم داشته باشیم، می‌توانیم شروع به بحث درباره مفهوم سنجش کمیت‌های درون سیستم کنیم. سه نوع اولیه از معیارها وجود دارد: A، B و C. آن‌ها را می‌توان به‌صورت ذیل توصیف نمود:

[۶۰] data bus
[۶۱] address bus

نوع A به دنبال شمارش چند آیتم در یک دوره زمانی مشخص است. به‌عنوان‌مثال، ممکن است ما علاقه‌مند باشیم بدانیم که CPU در چه بازه‌های زمانی یک دستورالعمل را در طی هر ثانیه دریافت می‌کند. این می‌تواند نشانگر سرعت دستورالعمل پردازنده، با توجه به ترکیب دستورالعمل‌های ارائه شده باشد.

نوع B تمام متغیرهای حالت را مورد سنجش قرار می‌دهد (رویدادهای معتبر و مقادیرشان). نمایشی از این نوع از سنجش می‌تواند استخراج تمام مقادیر برای تمام ثبات‌ها و دستگاه‌های داخلی در ابتدای یک سیکل اجرای دستورالعمل باشد.

نوع C کسری از زمان که سیستم در یک وضعیت (حالت) است را مورد سنجش قرار می‌دهد. نمونه‌ای از این معیار می‌تواند این باشد که ببینیم اجرای دستورالعمل‌های بار در چه کسری از زمان در مقایسه با تمام انواع دستورالعمل‌ها در طی دوره زمانی مورد سنجش قرار گرفته اجرا می‌شود.

صرف تعیین نوع رویداد مورد نظر برای سنجش و مقادیری که این رویداد را نمایندگی می‌کنند کفایت نمی‌کند. این باید بتوان تأیید نمود که یک وضعیت مشخص حاصل شده است و اینکه تمام رویدادها و متغیرهای وضعیت برای حالت مورد نظر معتبر هستند. علاوه بر تأیید اینکه یک حالت معتبر حاصل شده است، باید این بتوان تعیین نمود که آیا ما در یک سر انتهایی یا گذرای درون یک حالت هستیم یا نه.

به‌منظور اطلاع یافتن از اینکه که در کجا در یک حالت هستیم، ما باید ابزارهایی برای سنجش رویدادهای سیستم‌هایی که به آن‌ها علاقه‌مندیم داشته باشیم. چندراه برای سنجش این رویدادها وجود دارد، که هر یک مسائل خودشان را دارند. ما می‌توانیم از مانیتورینگ سخت‌افزاری، مانیتورینگ نرم‌افزاری، یا مانیتورینگ ترکیبی استفاده کنیم. تصمیم درباره اینکه کدام‌یک از این تکنیک‌ها باید استفاده شوند به فاکتورهای بسیاری بستگی دارد، از جمله دسترسی‌پذیری، تناوب رویداد، بررسی محصولات و نتایج، بالاسری مانیتورینگ، و انعطاف‌پذیری تکنیک مورداستفاده قرار گرفته.

مانیتورینگ سخت‌افزاری الزام می‌کند که تحلیلگر سیستم قابلیت افزودن ابزار دقیق را به سیستم مورد سنجش قرار گرفته داشته باشد. به‌عنوان‌مثال، ما می‌توانیم یک تحلیلگر منطقی را برای سنجش سیگنال‌های درون سیستم یا قرار دادن یک سخت‌افزار دارای طراحی ویژه برای سنجش سیگنال‌های درون سیستم یا قرار دادن یک کارت سخت‌افزار دارای طراحی ویژه برای استخراج برخی سیگنال‌ها از

یک سیستم ضمیمه نماییم. این حالت سنجش برای ما امکان سنجش چند زیرمجموعه از عناصر سخت-افزار سیستم کلی را فراهم خواهد آورد. ما تنها می‌توانیم آنچه برای مانیتورینگ در معرض دید و دسترسی است را مورد سنجش قرار دهیم. اگر آیتم یا کنشی که ما مایل به رصد آن هستیم به آسانی در دسترس نباشد، ممکن است ما نتوانیم با استفاده از یک مانیتور سخت‌افزاری آن را مورداستفاده قرار دهیم. ممکن است لازم شود که ما از ابزارهای دیگری برای استخراج اطلاعات از سیستم استفاده نماییم. شکل دیگری از مانیتورینگ سخت‌افزاری از سخت‌افزار تست یکپارچه استفاده می‌کند، که در سیستم در حال رصد در طی طراحی سیستم‌ها طراحی می‌شود. شکل رایجی از این طرح مانیتورینگ در دستگاه‌های یکپارچه‌سازی مقیاس بسیار بزرگ (VLSI) یافت می‌شود. بسیاری از دستگاه‌های VLSI به شکلی طراحی می‌شوند که تمام آیتم‌های داده‌های مورد نظر را بتوان در خود دستگاه تست نمود، یا، حداقل، نقاط داده‌های تست از چیپ خارج می‌شوند تا دستگاه‌های اضافی را بتوان برای گردآوری این اطلاعات و محاسبه سلامت دستگاه استفاده نمود.

در تمام این موارد، ضروری است که مانیتورینگ سخت‌افزاری به‌صورت بخشی لاینفک از سیستم طراحی شود، به صورتی که تداخلی با سیستم‌عامل نداشته باشد. مطلوب نیست که تجهیزات مانیتورینگ با سیستم تحت رصد تداخل داشته باشند. اگر این مطرح باشد، نتایج حاصل از مانیتورینگ مورد تردید هستند و ممکن است منتج به جمع بندی‌های اشتباهی شوند. سخت‌افزار مانیتورینگ باید با در نظر گرفتن دستگاه مورد سنجش انتخاب و طراحی شود. تعیین مکان‌های نمونه‌برداری و تناوب سنجش‌ها باید جلوتر از زمان طراحی شود، نه پس‌ازآنکه مانیتور در جایگاهش قرار گرفت. روش مانیتورینگ هم باید جلوتر از زمان به اجرا دربیاید. بدین معنی که ما باید تعیین کنیم که آیا مانیتور قرار است به طور همزمان با سیستم مورد سنجش عمل نمایند یا ناهمزمان با آن. ما باید تمام جنبه‌های وجود مانیتور را در سیستم مورد سنجش قرار گرفته تعیین و تعریف کنیم. اگر بخواهیم به داده‌هایی دست یابیم که شایان اعتماد باشند هیچ چیز نباید از قلم بیفتد.

مانیتورینگ نرم‌افزاری برای موفقیت نیازمند پشتیبانی از سیستم تحت مطالعه خواهد بود. مانیتورینگ نرم‌افزاری الزام می‌کند که ابزاری برای طراحان رصد برای دستیابی به عناصر سخت‌افزاری و این عناصر نرم‌افزاری سطح پایین وجود داشته باشد – به‌عنوان‌مثال، کلاکهای سیستم، تایمرهای قابل برنامه‌ریزی، ثبات‌های وقفه، و ثبات‌های وضعیت سیستم‌ها. مانیتور نرم‌افزاری نوعی برای مانیتورینگ و نمونه‌برداری

دنباله‌ها طراحی شده است، نه برای مانیتورینگ همزمان. در مانیتورینگ دنباله، تحلیلگر کد اضافی را به یک توالی کد اضافه می‌کند تا زمان اجرای کد را بتوان رصد نمود. معمولاً ما به دانستن تعداد دفعات وارد کردن یک بخش کد، مدت اجرای بخش کد، و اینکه سیستم کلی چقدر از بخش کد را مورداستفاده قرار می‌دهد علاقه‌مند خواهیم بود.

مانیتورینگ نرم‌افزاری، همانند مانیتورینگ سخت‌افزاری، همچنان نیازمند آن است که ما پیش از موعد اطلاع داشته باشیم که معیارهای نمونه‌برداری در کجا درون سیستم رخ می‌دهند و از تناوب این نمونه-برداری در صورتی که قرار بر این باشد که سنجشهایمان معنادار باشند مطلع شویم.

در مانیتورینگ نرم‌افزاری، که در آن از تکنیک‌های مانیتورینگ نمونه‌برداری شده استفاده می‌شود، ما نیاز به دسترسی به فراخوانی‌های سیستم‌های عامل سطح پایین داریم. این نوع از دسترسی از این بابت لازم است که بتوانیم موجب یک وقفه سیستم شده و کنترل سیستم را بر عهده بگیریم. کنترل وقفه می‌تواند امکان ورود به سیستم و گردآوری اطلاعات حالت سیستم‌ها همچون محتوای ثبات‌ها و پرچم‌های وضعیت را فراهم بیاورد. یک جنبه مثبت این شکل از مانیتورینگ نرم‌افزاری این است که نمی‌تواند منجر به تغییر کدی شود، چرا که تمام اطلاعات موردنیاز را می‌توان با استفاده از اطلاعات در دسترس گردآوری نمود.

یک شکل رایج‌تر سنجش از مانیتورینگ ترکیبی[۶۲] استفاده می‌کند. این شکل از مانیتورینگ سیستم‌ها از مفاهیم و مکانیسم‌هایی از هم مانیتورینگ نرم‌افزاری و هم مانیتورینگ سخت‌افزاری استفاده می‌کند. برای استفاده از مانیتورینگ سخت‌افزاری ما باید همان مجموعه از مسائل را طی کنیم که برای مانیتورینگ نرم‌افزاری هم وجود داشت. این ساختار ممکن است نیازمند همگام‌سازی چند ساختار سخت‌افزاری و نرم‌افزاری باشد. ما باید برنامه‌های کنترلی را برقرار نماییم تا مشخص شود که چه زمانی و چگونه سیستم تحت تست را مانیتور می‌کنیم. ما باید تعیین کنیم که چه چیزی را با دستگاه‌های سخت-افزاری قرار است ثبت کنیم و چه چیزهایی را با ابزارهای نرم‌افزاری. به محض اجرای زیرسیستم مانیتورینگ، ما باید تعیین کنیم که چگونه و به چه تعداد دفعاتی اطلاعات گردآوری شده را بازیابی می‌کنیم. این، ما این باید تعیین کنیم که چگونه و چه زمانی سیستم‌های سنجشی و اندازه‌گیری شده را همگام‌سازی می‌کنیم.

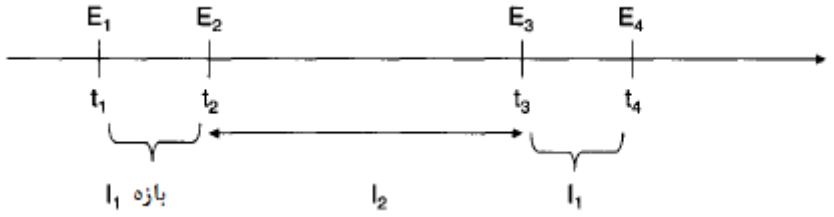
---

[۶۲] hybrid monitoring

مانیتورینگ ترکیبی مشکلات خودش را دارد. همانند مانیتورینگ سخت‌افزاری، ما باید ابزاری برای استخراج سیگنال‌های مورد نظر از سیستممان داشته باشیم. ما باید تعیین کنیم که چه عناصری که مایل به تستشان هستیم بهتر با سخت‌افزار تست می‌شوند و کدامها با نرم‌افزار. ما باید تأثیری که مانیتورینگ نرم‌افزار بر سیستم‌های رصد شده دارد بشناسیم و تأثیرشان را درک کنیم، تا سنجش‌های صحیح انجام شوند. نهایتاً اینکه، همیشه باید به خاطر داشته باشیم که سنجشها فقط به‌اندازه نقاط اندازه‌گیری در دسترس تناسب دارند.

۳-۵ بازه‌ها

سنجش نیازمند آن است که ما حوزه یا محیطی را در اختیار داشته باشیم که در آن سنجش را انجام دهیم. در سیستم‌های کامپیوتری، این محیط عبارت است از کلاکهای سیستم و سیکل اجرای دستورالعمل. یک محیط عمده دیگر کارکردهای سیستم‌های مرتبه بالاتر است. برای سنجش این آیتم‌ها، ما باید بر بازه اجرایشان تمرکز کنیم. یک بازه دوره‌ای از زمان را نمایندگی می‌کند که آغاز یک توالی رویداد مورد نظر و انتهای این توالی رویداد را محدود می‌کند (شکل ۳-۳).



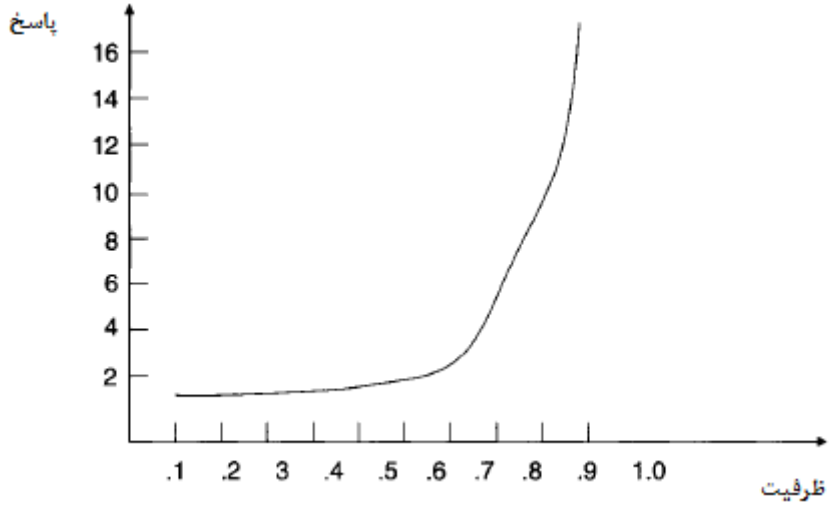
شکل (۳-۳): نمونه‌ای از بازه‌ها

در شکل (۳-۳)، بازه I_1 از تأخیر زمان "رویداد ۱" و تأخیر زمانی "رویداد ۲" تشکیل یافته است. بازه‌ها به‌عنوان ابزاری برای سنجش دوره توالی‌های اجرای یک رویداد یا دوره زمانی بین چنان اجراهایی استفاده می‌شوند (در شکل ۳،۳، بازه ۲). دو بازه در صورتی یکسان هستند که نشانگر توالی یکسانی از

رویدادها باشند (مرتبط باشند) و بازه زمانی بین رویدادها معادل باشد. دو بازه نشانگر دو توالی مجزا از رویدادهای مختلف می‌توانند بازه‌های معادلی هم داشته باشند، ولی آن‌ها نامرتبط هستند.

۳- ۶پاسخ

پاسخ یک مفهوم مهم در مطالعات عملکرد سیستم‌های کامپیوتری است. زمان پاسخ نشانگر یک معیار دوره زمانی است که یک کاربر یا برنامه‌های کاربردی باید از نقطه انتشار یک اقدام یا فرمان تا تکمیل و برگرداندن کنترل برای فرمان درخواست شده انتظار بکشد. معیار معمول مورداستفاده ممکن است زمان پاسخ (یک بازه) را در برابر بار سیستم (سیستم کارها) قرار دهد. این منحنی می‌تواند مشابه با مورد نشان داده‌شده در شکل۳،۴ باشد.



شکل (۳-۴): زمان پاسخ در مقایسه با بار سیستم

تفسیر این منحنی به ابزاری مهم برای ارزیابی سیستممان تبدیل می‌شود. در شکل (۳-۴)، ما می‌بینیم که زمان پاسخ توالی کنش سنجیده شده ما برای بارهای زیر تقریباً ۶۰ درصد ظرفیت سیستم مورد سنجش قرار گرفته در محدوده قابل‌تحمل می‌ماند (بین ۱,۰ و ۳,۰). با افزایش بار به بالای این نقطه، پاسخ به طور

نمایی افزایش می‌یابد –وقتی که سیستم به طور کامل بارگیری شده باشد به یک سطح اشباع می‌رسد، و یک زمان پاسخ مجانبی می‌دهد که به بینهایت میل می‌کند. می‌توان از این مثال ساده اهمیت زمان پاسخ را به‌عنوان یک ابزار سنجشی در سیستم‌های مدل‌سازی و ارزیابی مشاهده نمود.

مشکل به تلاش برای تعیین پاسخ چه چیزی مربوط می‌شود. ممکن است نگاه به عملکرد سیستم‌هایمان از تنها یک معیار یا توالی اقدام کفایت نکند. شاید ما به خانواده‌ای از چنان توالی‌های اقدامی علاقه‌مند باشیم، که نیاز به یک سری از تست‌های مجزا برای مطالعه اثر هر یک از این توالی‌های قابل سنجش در قبال بار سیستم دارد. علاوه بر این شکل از معیار، ما ممکن است به این نیز علاقه‌مند باشیم که چگونه این توالی‌های مختلف اقدام، با افزایش بار، بر یکدیگر تأثیر می‌گذارند نیز علاقه‌مند باشیم. این می‌تواند منجر به خانواده‌ای از منحنی‌های پاسخ شود، که ممکن است نیاز به تفسیر در قبال یکدیگر و بارها داشته باشند.

۳-۷ استقلال

استقلال یک مفهوم بسیار مهم دیگر در مدل‌سازی و تحلیل کارایی است. یک اقدام یا رویداد در صورتی مستقل از رویداد یا اقدام دیگر لحاظ می‌شود که بر نتیجه یکدیگر تأثیر نگذارند. به‌عنوان‌مثال، بالا انداختن یک سکه پس از غلتاندن یک تاس مستقل از هم هستند، زیرا انداختن سکه تاثیری بر نتیجه آن ندارد. اگر ما این دو رویداد را به‌عنوان دو فضای نمونه مجزا ببینیم، ارتباط این رویدادها روشن‌تر می‌شود. فضای نمونه برای انداختن مشخصاً مجموعه $\{H,T\}$ است، و فضای نمونه برای تاس صرفاً مجموعه $\{۱،۲،۳،۴،۵،۶\}$ است. فضای نمونه در شکل (۳-۵) حاصل‌ضرب دکارتی این دو فضای مستقل است.

استقلال رویدادها در یک سیستم یک مفهوم مهم است که باید در زمان ارزیابی سیستم‌ها در نظر گرفته شود. اگر دو رویداد مستقل باشند، لزومی ندارد که این‌ها را به‌عنوان آیتم‌های مرتبطی لحاظ کنیم که ما را ملزم به بررسی پاسخشان در رابطه با یکدیگر و محیطشان می‌کنند.

	1	2	3	4	5	6
H	H1	H2	H3	H4	H5	H6
T	T1	T2	T3	T4	T5	T6

شکل (۳-۵): حاصل‌ضرب دکارتی دو فضای نمونه مستقل

در یک سیستم کامپیوتری، دو برنامه‌ای که نتوانند همزمان با یکدیگر اجرا شوند را می‌توان آیتم‌های مستقل لحاظ نمود و بر این اساس مورد تجزیه‌وتحلیل قرار داد. اگرچه آن‌ها بر روی سخت‌افزار یکسانی اجرا می‌شوند و احتمالاً از نرم‌افزار سیستم‌های عامل یکسانی استفاده می‌کنند، ولی از آنجا که نمی‌توانند با یکدیگر تداخل داشته باشند و از جنبه توالی بندی مستقل نیستند، آن‌ها را می‌توان به‌صورت آیتم‌های مجزا و نامرتبط مورد ارزیابی قرار داد. این به بخش مهمی از مدل‌سازی و تحلیل یک سیستم برای تعریف تمام عناصر و رابطه‌شان با یکدیگر تبدیل می‌شود. از این‌رو از این تعاریف می‌توان برای کمک به تعیین استقلال استفاده نمود. ما بعداً وقتی که به دنبال احتمال و سپس نگاشت این در عناصر سیستم‌های کامپیوتری برویم درباره این ویژگی رویدادها بیشتر بحث خواهیم کرد.

۳-۸ تصادفی بودن

مفهوم تصادفی بودن رویدادها به‌اندازه مفهوم استقلال برای مدل‌سازی اهمیت دارد. تصادفی بودن یک ویژگی یک رویداد وقوع مجددش است. اگر یک رویداد تصادفی باشد، متضمن آن است که الگویی وجود ندارد که بتوان آن را برای تعیین زمان وقوع مجدد آن در رویدادها نگاشت کرد. اثبات تصادفی بودن دشوار، و چه‌بسا ناممکن، است. بااین‌حال، می‌توان عکس آن را نشان داد، یعنی اینکه یک رویداد تصادفی نیست. ما می‌توانیم از این ادعا استفاده کنیم که یک الگو به‌عنوان راهی برای نشان دادن اینکه گذشته کمکی به تعریف آینده یک رویداد نمی‌کند وجود ندارد. یک توالی تصادفی از آزمایش‌ها تحقق ویژگی استقلال تعریف‌شده در بخش قبلی است.

تصادفی بودن یک مفهوم ریاضی است. در ریاضیات، ما اعداد تصادفی را نشأت گرفته از منبع نامتناهی تصادفی تصور می‌کنیم. در عمل، توالی‌های محدودی از اعداد در دسترس وجود دارند، وقتی آن‌ها تولید شدند الگو دار خواهند بود. به‌عنوان‌مثال، اگر ما یک تاس را بغلتانیم، پیش از آنکه آن را بغلتانیم نمی‌دانیم که چه عددی خواهد آمد، ولی پس از غلتانده شدنش فقط یک نتیجه وجود خواهد داشت.

در یک سیستم کامپیوتری، رویدادهای ناشی از منابع خارجی (مثلاً فشار دادن دکمه‌های صفحه کلید توسط کاربر، فراخوانی‌های از راه دور برای یک سرور) را می‌توان رویدادهای تصادفی لحاظ نمود. از این‌رو، وقوع آن‌ها را نمی‌توان به طور زودهنگام پیش‌بینی نمود و پس از آخرین ورود هم نمی‌توان

آینده را پیش‌بینی کرد. این مفهوم وقتی اهمیت بالایی می‌یابد که ما مایل به آنالیز سیستم‌های کامپیوتری‌مان با استفاده از مفاهیم ریاضی باشیم.
در فصل‌های بعدی، در زمان بررسی متغیرها تصادفی و استفاده‌شان در مدل‌سازی سیستم‌های محاسباتی به‌صورت زنجیره‌های مارکوف و فرایندهای مارکوف برویم، بحث بیشتری درباره مفهوم تصادفی بودن انجام خواهیم داد.

۳-۹ بارهای کاری

یکی از مفاهیمی که تا این مقطع در این کتاب درباره آن بحث نموده‌ایم به بار کاری، یا، به عبارتی ساده‌تر، بار مربوط می‌شود. این عبارات یک عنصر فوق‌العاده مهم در مسئله مدل‌سازی سیستم‌های کامپیوتری را به خود اختصاص می‌دهند. بار کاری یا بار نشانگر رویدادها یا توالی‌های رویدادهای ارائه شده به سیستم برای مدل‌سازی و پیشبرد سیستم تحت مطالعه است. این بار نشان می‌دهد که چه تعداد از برخی از توالی‌های رویداد برای اجرا در طی یک دوره زمانی مشخص ارائه می‌شوند. یک مثال آن می‌تواند تعداد دستورالعمل‌ها در ثانیه و ترکیب انواع دستورالعمل‌های ارائه شده برای اجرا در ثانیه باشد. ترکیب حجم و ترکیب، و این مدت‌زمان بار، مهم است.
این مدت می‌تواند کلاً یک‌بار باشد، و سیستم را ملزم به‌صف بندی درخواست‌ها و اجرای آن‌ها پس از قابل‌دسترسی شدن منابع نماید. این نوع از ترکیب و بار می‌تواند سیستم را پیشاپیش اشباع نماید و سپس، در طی پردازش آیتم‌ها، به بی‌باری کاهش یابد. این مدت ممکن است بی‌پایان باشد، به‌طوری‌که بار به طور متوالی تازه‌سازی شود تا یک‌بار تعادلی یا اشباع ثابت را برای سیستم فراهم بیاورد. بارها می‌توانند دوره‌ای باشند، که در آن دستورالعمل‌ها همه یک‌باره برای سرویس ارائه می‌شوند و سپس امکان مورد پردازش قرار گرفتن را پیدا می‌کنند. این بار سپس پس از گذشتن دوره زمانی تجویزشده مجدداً وارد می‌شود، و یک پیشرفت چشمگیر دیگر را در الزامات پردازشی فراهم می‌آورد.
دانشی برای توسعه و انتخاب بار کاری برای مدل کننده سیستم‌های کامپیوتری وجود دارد. به‌عنوان‌مثال، جامعه پایگاه داده مجموعه‌ای از بارهای کاری تراکنشی را با هدف آزمودن پیکربندی‌های مختلف سیستم‌های پایگاه داده توسعه داده است. این مجموعه از تراکنش‌ها و سیستم پایگاه داده اساسی در طی چند سال از طریق سنجش سیستم‌های پایگاه داده واقعی و نیاز به ارزیابی پایگاه داده‌ها از طریق یکدیگر

با مجموعه متوازن شناخته‌شده‌ای از بارها توسعه داده‌شده است. به همین صورت، صنعت کامپیوترهای شخصی این مجموعه‌ای از بارهای کاری سیستم‌ها را با هدف فراهم‌سازی امکان ارزیابی عملکرد یک معماری کامپیوتر با معماری کامپیوتر دیگر برای مشتریان توسعه داده است.

۳-۱۰ مسائلی که در توسعه و استفاده از مدل با آن‌ها مواجه می‌شویم

توسعه یک پروژه ارزیابی کارایی برای یک سیستم مشخص مشکلات خودش را دارد. ما باید کار را با توسعه مفهومی برای اینکه برای چه چیزی و چرا در حال ارزیابی هستیم شروع کنیم. بدین معنی که آیا هدف مطالعه کارایی ما سنجش کارایی جاری یک سیستم است یا احتمالات آن در آینده؟ آیا ما در حال سنجش هزینه سیستم در حال حاضر هستیم یا در آینده؟ آیا ما در حال سنجش صحت سیستم هستیم یا کفایت آن؟ چگونه ما این عبارات را تعریف می‌کنیم؟ چه چیزی صحت یا کفایت را دیکته می‌کند؟ چرا ما نیاز به اجرای این مطالعه داریم؟

یک تحلیلگر نوعی ابتدا کار را با این دغدغه اولیه شروع می‌کند که آیا سیستم به درستی مطابق با هدف در نظر گرفته شده‌اش عمل می‌کند یا نه. به‌عنوان‌مثال، اگر یک سیستم کامپیوتری بتواند عملیات همزمان را به اجرا در بیاورد، یک معیار اولیه این است که می‌تواند دقیقاً این کار را انجام دهد. یک نگرانی ثانویه مدل‌کننده این است که سیستم کارایی مناسب و با هزینه‌ای معقولانه داشته باشد. این به طور ضمنی اشاره دارد که ما به راهی برای سنجش و پیش‌بینی اینکه چه چیزی عملکرد مناسب است و چه چیزی هزینه معقولانه است نیاز داریم.

برای درک این ضوابط ما ابتدا باید آن‌ها را در زمینه محیطی قرار دهیم که در آن سیستم باید عملیاتی باشد. با این وجود حتی پیش از این، ما باید کار را با تعیین معنای این سیستم شروع کنیم. به‌عنوان‌مثال، اگر این یک کامپیوتر شخصی است، ما باید بدانیم که معنای این واژه چیست. آیا ما مایلیم که مادربورد، نوع پردازنده، حجم و نوع حافظه، بوردهای I/O، کارت‌های گرافیکی، دیسک درایوها، و احتمالاً رابط‌های شبکه استفاده کنیم؟ یا صرفاً منظورمان جعبه سیاه است، بدون اینکه برایمان اهمیت داشته باشد که درون آن چیست؟ وقتی سیستم مورد نظر تعریف شد، مدل‌کننده باید تعریف کند که چه اجزائی این سیستم را شکل می‌دهند و اهمیتشان در زمینه کل سیستم چیست.

با توجه به تعریف سیستم‌ها و تعریف اجزا، ما سپس باید محیطی که سیستم در آن فعالیت می‌کند را تعریف کنیم. این محیط فقط باید دربرگیرنده فاکتورهای مهم تعریف‌کننده آن است، نه همه چیز. به‌عنوان‌مثال، اگر ما در حال مطالعه یک معماری PC باشیم، ممکن است مایل باشیم که بدانیم آیا در معرض عناصر، دماهای فوق‌العاده، رطوبت و امثالهم قرار خواهد داشت یا نه.

وقتی این محیط تعریف شد، ما باید تعیین کنیم که چه پارامترهایی برای ما جالب توجه هستند. این‌ها می‌توانند دربرگیرنده پارامترهایی باشند که سیستم بر اساس آن استفاده می‌شود یا سنجیده می‌شود. برخی از پارامترها ممکن است دربرگیرنده چیزهایی همچون سرعت پردازنده PC، اندازه حافظه اولیه و امثالهم باشند.

پاسخ مشترک کاربران PC و کاربران سیستم‌های کامپیوتری به طور کلی این است که آن‌ها نمی‌توانند به راحتی عبارت فوق را تعریف کنند. آن‌ها نوعاً به ارزیابی عملکرد سیستم‌های کامپیوتری به‌عنوان پاسخ دادن به تنها یک پرسش نگاه می‌کنند: اگر کامپیوتر من به درستی عمل نکند، آیا من نمی‌توانم "هر چه" بیشتری را اضافه کنم تا بهتر عمل کند؟ مشکل در این نهفته است که چگونه بدانیم که چه "هر چه" لازم است. چه مقدار از این "هر چه" لازم است؟ مشکل اینجاست که فرد به راحتی نمی‌داند که چه زمانی "هر چه" را اضافه کند که یک کمیت مشخص نتیجه مورد نظر را ارائه نماید. مهم‌تر از همه اینکه، بدون ارزیابی کارایی، ما چطور از تکمیل کارمان مطلع شویم؟

مسئله‌ای که ارزیاب عملکرد با آن مواجه است این است که چگونه اینکه چه چیزی باید مورد سنجش قرار بگیرد و چگونه این کار باید انجام شود را تعیین کند. دو کلاس اصلی تکنیک‌ها برای ارزیابی عملکرد سیستم‌های کامپیوتری وجود دارد. کلاس اول لحاظ کردن یک سیستم موجود و طراحی برخی از آزمایش‌های دربرگیرنده احتمالاً سخت‌افزار، نرم‌افزار یا هر دو است. سپس نتیجه مورد سنجش قرار بگیرد تا مشخص شود که چه چیزی موردنیاز است. کلاس دوم مدلینگ ابزار مدل‌سازی از ابزارهای انتزاعی‌تری استفاده می‌کند. این‌ها دربرگیرنده مدل‌سازی یا شبیه‌سازی تحلیلی می‌شوند. مدل‌سازی تحلیلی معمولاً از تئوری صف بندی یا شبکه‌های پتری استفاده می‌کند و می‌تواند تحلیل درشتی از سیستم‌های تحت مطالعه ارائه نماید. شبیه‌سازی‌ها می‌توانند باعث افزایش اعتبار شوند و به قیمت افزایش زمان و تحلیل طراحی. شبیه‌سازی‌ها را می‌توان به‌عنوان مدل‌های گسسته مبتنی بر رویداد، مدل‌های مبتنی بر پیوستگی یا مدل‌های ترکیبی طراحی نمود.

معیارهای کارایی مورداستفاده تحلیلگر در تعیین عملکرد شامل پاسخگویی، سطوح استفاده، مأموریت‌پذیری، اتکاپذیری، بهره‌وری، و پیش‌بینی پذیری می‌شوند. پاسخگویی نشانگر قابلیت سیستمی است که باید فرمان‌ها را ارائه نماید و پاسخ‌ها را در یک دوره زمانی معقولانه ارائه نماید. سطح استفاده نشانگر درجه سیستمی است که در آن بارگیری شده است – به‌عنوان‌مثال، آیا سیستم ۵۰ درصد بارگیری شده است یا ۱۰۰ درصد اشباع شده است؟ مأموریت‌پذیری به قابلیت سیستم برای اجرا به‌صورت مورد نظر برای مدت تقاضا شده اشاره دارد. به‌عنوان‌مثال، یک فضاپیما باید بسیار مأموریت‌پذیر باشد. اتکاپذیری به معیار آخر مرتبط است ولی نشانگر قابلیت سیستم برای مقاومت در برابر شکست یا عملیاتی ماندن است. بهره‌وری نشانگر یک معیار بازدهی سیستم ارائه شده است. و پیشبینی‌پذیری نشانگر یک معیار قابلیت یک سیستم برای عملکرد در صورت نیاز تحت تمام یا بیشتر شرایط است.

تمام این معیارها، با توجه به کلاس‌های مشخص سیستم‌ها، جایگاهی دارند. به‌عنوان‌مثال، یک ساختار محاسباتی همه‌منظوره باید دارای ویژگی‌های پاسخگو بودن، برخورداری از سطوح خوب استفاده و بهره‌ور بودن باشد. سیستم‌های دسترسی بالا، همچون پردازش تراکنش یا سیستم‌های پایگاه داده، نه‌تنها باید پاسخگو باشد بلکه باید دارای درجه بالایی از اتکاپذیری نسبت به محیط محاسبه همه‌منظوره باشد. سیستم‌های کنترل بی‌درنگ نیاز به پاسخگویی، اتکاپذیری و پیشبینی‌پذیری بالایی دارند. سیستم‌های ماموریت‌محور، همچون سیستم‌های کنترل هوانوردی، نیاز به اتکاپذیری فوق‌العاده بالایی در مدت‌های کوتاه دارد و باید پاسخگو باشد. کاربردهای بلندمدت، همچون فضاپیما وسایل نقلیه زیرآبی مستقل، باید بسیار اتکاپذیر، مأموریت‌پذیر و پاسخگو باشند.

خطاها یا اشتباه‌های رایجی وجود دارند که تحلیلگران عملکرد سیستم‌های کامپیوتری مرتکب می‌شوند یا باید در زمان اجرای کارهایشان از آن اجتناب نمایند. اولین و رایج‌ترین آن‌ها، در زمینه مطالعه عملکرد، هدف نداشتن یا داشتن اهدافی است که به خوبی تعریف نشده‌اند. این اهداف باید دربرگیرنده مشخصه-ای برای یک مدل از سیستم یا بخش تحت مطالعه و تعریف تکنیک‌ها، معیارها و بار کاری مورداستفاده در ارزیابی‌ها باشد. مسئله عمده دوم تعیین اهداف جهت‌دار است. این یک اشتباه بسیار رایج توسط مدل‌کننده است. هدف اثبات این خواهد بود که "سیستم من برتر از سیستم فرد دیگر است". این باعث می‌شود که تحلیلگر به هیئت‌منصفه تبدیل شود، که این منتج به قضاوت‌های بد خواهد شد.

اگر تحلیلگر از یک رویکرد نامتقارن برای توسعه مدل استفاده کند یا پیش از درک کامل مسئله تحت مطالعه به تحلیل پرش کند، نتایج حاصله ناقص خواهند بود. انتخاب معیارهای کارایی نادرست یا معیارهای گمراه‌کننده منجر به نتایج و جمع بندی‌های خطادار خواهد شد. انتخاب یک بار کاری نامتناسب یا بدون استرس منتج به تفسیر نادرست از مرزهای عملکرد سیستم خواهد شد. انتخاب تکنیک ارزیابی نادرست ، به‌عنوان‌مثال، مدل‌سازی تحلیلی، در زمانی یکه بستر تست انتخاب درست است – منتج به رویکرد بیش از حد ساده‌انگارانه یا پیچیده‌ای خواهد شد. نادیده گرفتن پارامترهای مهم سیستم می‌تواند منجر به جمع بندی‌های اشتباهی درباره حساسیت‌ها و اتکاپذیری‌های میان عناصر سیستم داشته باشد. طراحی تجربی نامناسب یا انتخاب بد سطح جزئیات می‌تواند موجب رسیدن به جمع بندی‌های گمراه‌کننده‌ای شود. تحلیل خطادار، عدم انجام تحلیل حساسیت، یا حتی عدم تحلیل منتج به عدم موفقیت خواهد شد. نادیده گرفتن ورودی، خطاهای ورودی یا خروجی، یا تغییرپذیری این‌ها می‌تواند موجب تفسیرهای گمراه‌کننده‌ای از نتایج شود. عدم اجرای تحلیل حساسیت، تحلیل نقطه پرت، یا نادیده گرفتن تغییر نیز می‌تواند موجب بروز مشکلاتی در تفسیر یا اعتماد به نتایج شود. اجرای بیش از حد پیچیده یک تحلیل یا ارائه یا تفسیر نامناسبی از نتایج، و این حذف مفروضات و محدودیت‌ها، باعث رسیدن به تحلیلی نادرست خواهد شد.

به‌منظور تلاش برای کاستن از این مشکلات تحلیلگر باید قبل، در جریان و بعد از انجام یک تحلیل پرسش‌های ذیل را مطرح نماید:

۱. آیا سیستم به درستی تعریف‌شده است و اهداف تحلیلی به‌روشنی بیان‌شده‌اند؟

۲. آیا این اهداف به شکلی عاری از جهت‌گیری بیان شده‌اند؟

۳. آیا تمام گام‌های تحلیل به شکلی متقارن دنبال شده‌اند؟

۴. آیا مسئله پیش از آغاز تحلیل به روشنی درک شده است؟

۵. آیا معیارهای کارایی با این مسئله مرتبط هستند؟

۶. آیا بار کاری برای این مسئله درست است؟

۷. آیا تکنیک ارزیابی مناسب است؟

۸. آیا لیست پارامترهایی که بر عملکرد تأثیرگذار هستند کامل است؟

۹. آیا تمام پارامترهایی که بر عملکرد مؤثر هستند به‌عنوان فاکتورهایی انتخاب شده‌اند که در طراحی تجربی قرار است استفاده شوند؟

۱۰. آیا طراحی تجربی از جنبه زمان و نتایج مورد انتظار کارا است؟

۱۱. آیا سطح جزئیات مدل کفایت می‌کند؟

۱۲. آیا داده‌های مورد سنجش قرار گرفته با تحلیل و تفسیر ارائه شده‌اند؟

۱۳. آیا تحلیل ازنظر آماری درست است؟

۱۴. آیا تحلیل حساسیت انجام‌شده است؟

۱۵. آیا خطاها در ورودی موجب تغییری قابل صرف نظر در نتایج می‌شوند؟

۱۶. آیا نقاط پرت در ورودی یا خروجی‌ها به درستی درمان شده‌اند؟

۱۷. آیا تغییرات آتی در سیستم و بار کاری مدل‌سازی شده‌اند؟

۱۸. آیا تغییرات ورودی لحاظ شده است؟

۱۹. آیا تغییرات در نتایج مورد آنالیز قرار گرفته است؟

۲۰. آیا توضیح این تحلیل آسان و بدون ابهام است؟

۲۱. آیا سبک ارائه برای مخاطبان مورد نظر آن مناسب است؟

۲۲. آیا نتایج تا جای ممکن به‌صورت گرافیکی ارائه شده‌اند؟

۲۳. آیا مفروضات و محدودیت های تحلیل مشخصاً مستندسازی و لحاظ شده‌اند؟

در زمان توسعه یک مطالعه عملکرد، تحلیلگر عملکرد خردمندانه می‌تواند یک رویکرد سیستماتیک را دنبال کند، که اجزاء آن از قرار ذیل هستند:

۱. بیان اهداف و تعریف سیستمی که باید مطالعه شود.

۲. لیست کردن مشخص و کامل سرویس‌ها و نتایج.

۳. انتخاب معیارهای کارایی.

۴. لیست کردن تمام پارامترهای مورد توجه سیستم‌ها.

۵. انتخاب فاکتورها برای مطالعه.

۶. انتخاب تکنیک ارزیابی که باید اعمال شود.

۷. انتخاب بار کاری.

۸. طراحی آزمایش‌ها.

۹. تحلیل و تفسیر نتایج.

۱۰. ارائه روشن و بدون ابهام نتایج

۱۱. تکرار در صورت لزوم.

۳-۱۱ یک مطالعه موردی

اگر ما مایل به مطالعه موضوع لوله‌های دوردست در مقایسه با فراخوان‌های رویه دوردست بودیم، می‌توانستیم تلاش ذیل برای مدل‌سازی را دنبال کنیم. گام نخست تعریف سیستمی است که قصد مطالعه آن را داریم. این متضمن توسعه مدلی است که دربرگیرنده تمام اجزاء عمده مورد علاقه است. در شکل (۳-۶)، ما چنین تعریفی را لحاظ می‌کنیم.

سرویس‌هایی که ما قصد تمرکز بر آن‌ها را داریم انتقال داده‌های کوچک و بزرگ هستند. ما توجهی به دیگر جزئیات سرویس‌ها نخواهیم داشت.

معیارهایی که ما مایل به تمرکز بر آن‌ها در کنار برخی از مفروضات هستیم دربرگیرنده این می‌شوند که هیچ خطا و هیچ اشکالی در سیستم وجود نداشته باشد. ما تمایل به تمرکز بر روی تعریف نرخ‌های موفقیت، زمان عملکرد، و الزامات منابع به ازاء هر سرویس داریم. منابعی که ما بر آن‌ها تمرکز خواهیم نمود عبارتند از کلاینت، سرور، و عناصر شبکه.

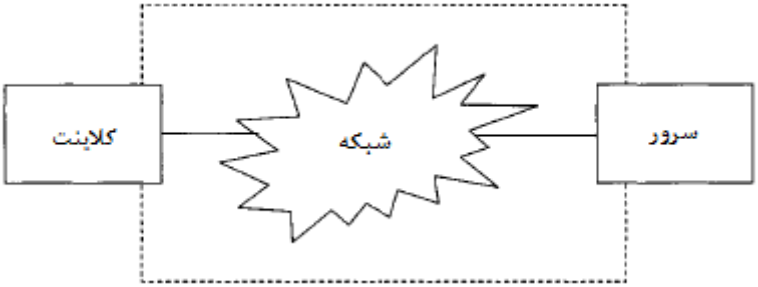
این معیارها و مفروضات می‌توانند ما را به تمرکز بر سنجش‌هایی که گردآوری می‌شوند سوق دهند، از جمله زمان صرف شده به ازاء هر فراخوانی، نرخ فراخوانی ماکسیمم به ازاء واحد زمان، زمان موردنیاز برای تکمیل بلوکی از N فراخوانی متوالی، زمان CPU محلی به ازاء هر فراخوان، زمان CPU دوردست به ازاء هر فراخوان، تعداد بایت‌های ارسال‌شده بر روی لینک به ازاء هر فراخوانی و امثالهم.

این‌ها به‌نوبه خود ما را ملزم به تمرکز بر تعریف پارامترهای سیستم خواهند نمود، به‌عنوان‌مثال، سرعت CPU های محلی و دوردست، سرعت شبکه، بالاسری سیستم‌عامل برای تداخل با کانال‌ها، بالاسری سیستم‌عامل برای تداخل با شبکه، اتکاپذیری شبکه، و امثالهم.

پارامترهای بار کاری مورداستفاده برای تعریف بار کاری ارائه شده می‌توانند دربرگیرنده زمان بین فراخوانی‌های متوالی، تعداد و اندازه‌های پارامترهای فراخوانی، تعداد و اندازه فراخوانی‌ها، نوع کانال استفاده‌شده، و دیگربارهای پس‌زمینه بر روی سایت محلی و دوردست و این شبکه باشند.

فاکتورهایی که ممکن است ما مایل به مطالعه‌شان باشیم شامل نوع کانال (RPC یا لوله‌های دوردست)، اندازه شبکه (فاصله طولانی، شبکه محلی)، اندازه تماس‌ها (کوچک، بزرگ)، و تعداد فراخوانی‌های متوالی باشند (که می‌تواند از یک، پنج، ده تا یک بار اشباع متغیر باشد) هستند.

مفروضات ارائه شده ممکن است دربرگیرنده تثبیت نوع CPU و سیستم‌عامل، صرف نظر از ارسال‌های مجدد ناشی از خطاهای شبکه، و انجام سنجشها با بارهای دیگر روی میزبان‌ها و شبکه‌ها شوند.



شکل (۳-۶): تعریف سیستم

تکنیک ارزیابی را می‌توان به‌عنوان یک نمونه اولیه در امتداد مدل‌های تحلیلی برای اعتبارسنجی یا مقید نمودن نتایج مورد انتظار انتخاب نمود. این بار کاری با استفاده از ساختارهای ترکیبی ساخته می‌شود. طراحی آزمایش تمام فاکتورها را تغییر خواهد داد، که منجر به یک طراحی تجربی عاملی کامل با استفاده از ۸۸ آزمایش می‌شود. این نشانگر تغییر تمام فاکتورهای توصیف‌شده در کل طیف مقادیر مفروضشان است. تحلیل داده‌ها دربرگیرنده تعیین تغییرات نتایج و مقایسه این‌ها در قبال هر فاکتور می‌شود. پس از آن نوبت به ترسیم تمام نتایج در شکل گرافیکی برای نمایش بهتر تغییرات کارایی می‌رسد.

۳-۱۲ خلاصه

در این فصل ما مختصراً برخی از مفاهیم اساسی موردنیاز برای شروع تحلیل سیستم‌های کامپیوتری را توصیف می‌کنیم. اولی با مفهوم اساسی زمان و نحوه استفاده از این واحد به‌عنوان یک ابزار اساسی برای سنجش عملکرد سروکار داشت. پس از این توصیف نوبت به توصیفی از تعریف رویدادها یا اقدامات درون یک حوزه رسید. مفاهیم زمان و رویداد سپس مدل‌سازی شدند تا ابزاری برای شناسایی نقاطی حاصل شود که سنجش از آن‌ها شروع می‌شود. روش‌های سنجش یک سیستم سپس، با تمرکز بر مانیتورینگ سخت‌افزاری و نرم‌افزاری و مسائل مرتبط با هر یک، توصیف شدند.

گام بعدی در تحقیق ما توسعه مفهوم اقدامات مرتبطی که اقدامات بزرگ‌تر را شکل می‌دهند و مدت این فعالیت‌ها است. این مدت به‌صورت بازه‌ای از یک اقدام پیچیده یا زمان بین تکرارهای متوالی یک توالی مشخص تعریف‌شده بود.

با مفهوم بازه‌ها ما سپس می‌توانیم بر سنجش یک توالی از اقدامات مرتبط تمرکز نماییم. تمرکز این بخش تعریف زمان پاسخ در رابطه با یک تحلیل عملکرد و مدل‌سازی سیستم کامپیوتری بود.

مفهوم این تراکنش‌ها پیچیده بعداً موردبررسی قرار گرفت. مفاهیم اقدامات وابسته و مستقل توسعه داده شد. پس از این‌ها نوبت به توسعه یک تعریف برای تصادفی بودن چنان رویدادهایی درون یک سیستم کامپیوتری رسید. این‌گونه ذکرشده است که این مفهوم تصادفی بودن در ساده‌سازی برخی از تکنیک‌های تحلیل یک مفهوم مهم است.

این بحث سپس با مقدمه‌ای بر مفهوم یک بار کاری و آنچه در عملکرد سیستم‌های کامپیوتری نمایندگی می‌کند دنبال شد. در بخش پایانی این فصل برخی از موانعی موردبحث قرارگرفته‌اند که تحلیل سیستم‌های کامپیوتری در طراحی، توسعه، عملیات، و تکمیل مطالعه عملکرد به آن‌ها نیاز دارند.

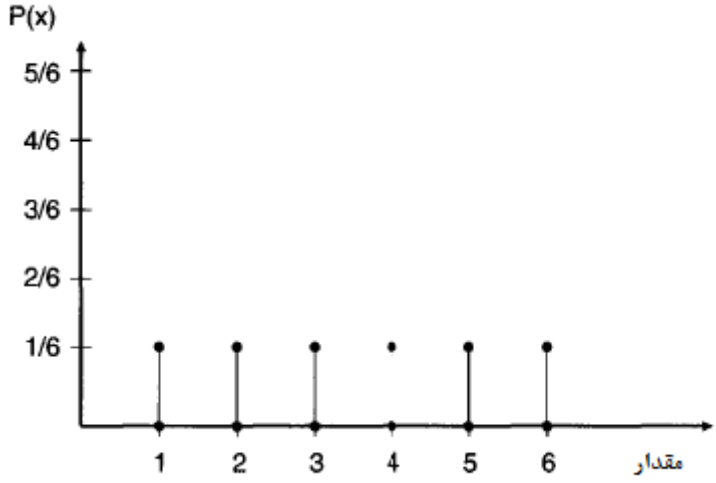
فصل چهارم

اصول کلی سنجشی

در مدل‌سازی سیستم‌های کامپیوتری، ما نوعاً به زمان‌های سرویس نهادهایی علاقه‌مند هستیم که از منابع سیستم استفاده می‌کنند. نهادها در بحث ما می‌توانند عملیات مختلفی را بر روی یک سیستم کامپیوتری نمایندگی کنند. به‌عنوان‌مثال، ممکن است ما علاقه‌مند به اطلاع یافتن از مدتی باشیم که برای سرویس یک وقفه لازم است یا، در یک سیستم پایگاه داده، به زمان برای قفل‌کردن یک آیتم داده‌ها در پایگاه داده لازم است. منابعی که به آن‌ها علاقه‌مندیم عناصر سخت‌افزار کامپیوتری و منابع نرم‌افزاری هستند. این نهادها عملیاتی را نمایندگی می‌کنند که با استفاده از منابع سیستم کامپیوتری به اجرا درمی‌آیند. به‌عنوان‌مثال، اگر منبع یک واحد پردازش مرکزی باشد، یک برنامه فعال بر روی CPU می‌تواند یک زمان سرویس متشکل از اجرای دستورالعمل (احتمالاً نشأت گرفته از ترکیب دستورالعمل)، مدیریت حافظه، مدیریت I/O، و دسترسی دستگاه ثانویه و تأخیرهای انتقال داشته باشد.

این اجزاء سیستم تحت آنالیز قابل‌مشاهده و احتمالاً قابل سنجش هستند. این بدان معنا نیست که لازم است که ما تمام اجزاء را با دقت و کامل مورد سنجش قرار دهیم. ممکن است مطلوب‌تر آن باشد که از زمان‌های متوسط و سرویس تصادفی ورودها برای مدل‌سازی این منابع و برنامه‌ها استفاده شود. این در واقع ممکن است مطلوب‌تر باشد که از زمان‌های متوسط ورودها و سرویس تصادفی برای مدل‌سازی این منابع و برنامه‌ها استفاده کنیم. اگر تمرکز مرور عملیات کلی برنامه باشد، و نه اجزاء این عملیات، زمان‌های سرویس غیر قابل پیش‌بینی به نظر خواهند رسید و، درنتیجه، می‌توان آن‌ها را تصادفی فرض نمود. بدون چنان مفروضاتی، مدل‌سازی یک سیستم کامپیوتری ممکن است در استخراج و تعیین جزئیات دقیق گرفتار شود، که چنین چیزی می‌تواند تحلیل کلی ما را مبهم نماید.

اگرچه زمان‌های سرویس برای رویدادها ممکن است غیر قابل پیش‌بینی باشند، ولی همچنان می‌توانیم آن‌ها را به شکلی مناسب برای مدل‌سازی و تحلیل با دقت نسبتاً خوب توصیف نماییم. به‌عنوان‌مثال، ما می‌توانیم بسیاری از رخدادهای رویدادی را در یک دوره زمانی طولانی زمانی مشاهده نموده و زمان سرویس متوسط ترکیبی را از این اطلاعات با درجه‌ای از دقت استنتاج نماییم. چنان تقریب‌هایی برای بسیاری از مدل‌ها و برای تحلیلشان کفایت می‌کنند، که در فصل‌های بعدی خواهیم دید.



شکل (۴-۱): توزیع احتمال برای یک تاس معمولی

یکی از مهم‌ترین تخمین‌های مرتبط با رویدادها و توزیع‌های سرویس در یک سیستم مدل‌سازی شده به توزیع احتمال مربوط می‌شود. در سیستم‌های مدل‌سازی شده مهم است که یک معیار احتمال از یک رویداد وجود داشته باشد که در رابطه با رویدادهای دیگر رخ می‌دهد. این توزیع احتمال به دنبال تخصیص مقادیر احتمال گسسته یا بازه‌های پیوسته مقادیر احتمال به رویدادها است. فرض بر این است که رویدادها و زمان‌های سرویس مجزا مستقل هستند و به شکلی یکسان توزیع‌شده‌اند (فصل ۳ را ببینید). این یک تخمین معقولانه برای واقعیت تحت بیشتر شرایط است.

ساده‌ترین شکل توزیع وقتی یافته می‌شود که ما مجموعه‌ای متناهی از مقادیر ممکن را داشته باشیم. به‌عنوان‌مثال، با غلتاندن یک تاس معمولی تنها می‌توان به مقادیر $\{۱،۲،۳،۴،۵،۶\}$ دست‌یافت، و نه چیز دیگری. این، احتمال اینکه هر یک از این مقادیر با غلتاندن حاصل شوند، با داشتن یک تاس معمولی و تعداد زیادی از آزمایش‌ها، برای هرکدام یک‌ششم است. مقادیر ممکن و یک نمایش گرافیکی در شکل (۴-۱) نشان داده‌شده است.

در معادله (۴-۱)، $P(x)$ احتمال (یا تناوب نسبی) وقوع مقدار x را نشان می‌دهد. در فصل ۵، ما خواهیم دید که $P(x)$ باید ویژگی‌هایی داشته باشد که $0 \ll P(x) \ll 1$ برای تمام مقادیر ممکن x از مجموعه مقادیر محتمل ما داشته باشد، و $\sum P(x) = 1$ باشد.

در زمان استفاده از چنان معیارهایی، مهم‌ترین پارامتر در زمان مدل‌سازی مقدار متوسط یا مورد انتظار است. این مقدار متناظر با مقدار متوسط است و به این صورت نمایش داده می‌شود:

$$E[X] = \sum_x xP(x) \qquad (۴-۱)$$

با داشتن توزیع معادله ۴،۱، $E[X]$ را می‌توان به این صورت محاسبه نمود:

$$E[X] = ۱(۱/۶) + ۲(۱/۶) + ۳(۱/۶) + ۴(۱/۶) + ۵(۱/۶) + ۶(۱/۶) = ۳.۵ \qquad (۴-۲)$$

یک معیار سنجشی تعمیم‌یافته اضافی که معمولاً استفاده می‌شود n امین لحظه است و به‌صورت مجموع مقدار x که به توان n ام رسیده است ضرب در احتمال وقوع این مقدار x محاسبه می‌شود، یا:

$$E[X^n] = \sum_x x^n P(x) \qquad (۴-۳)$$

برای مثال تاس معمولی، لحظه دوم را می‌توان به این صورت یافت:

$$E[X^۲] = ۱^۲(۱/۶) + ۲^۲(۱/۶) + ۳^۲(۱/۶) + ۴^۲(۱/۶) + ۵^۲(۱/۶) + ۶^۲(۱/۶) = ۱۵.۱۶۷ \qquad (۴-۴)$$

یک نوع مفیدتر از معیارها لحظه مرکزی n ام است، که با بررسی تفاوت بین مقادیر مورد سنجش قرار گرفته و مقدار مورد انتظار پیدا می‌شود. لحظه مرکزی از این فرمول یافته می‌شود:

$$E[(X - E[X])^n] = \sum_x (X - E[X])^n P(x) \qquad (۴-۵)$$

برای این مثال تاس معمولی،لحظه مرکزی دوم را می‌توان به این صورت یافت:

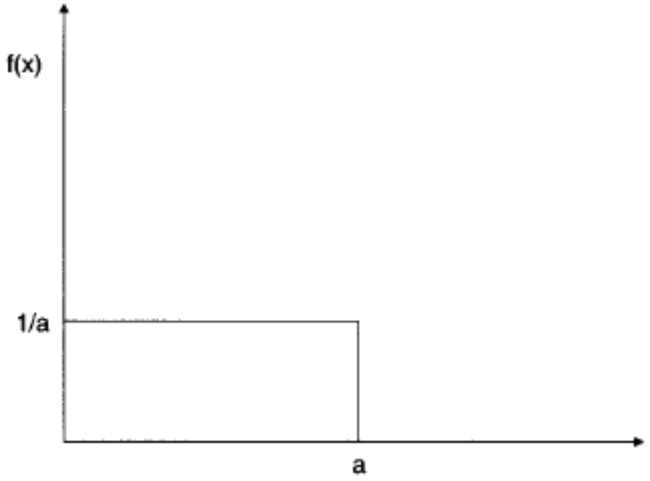
$$E[(X - E[X])^۲] = \qquad (۴-۶)$$

$$۱/۶\,[(-۲.۵)^۲ + (-۱.۵)^۲ + (-۰.۵)^۲ + (۰.۵)^۲ + (۱.۵)^۲ + (۲.۵)^۲] = ۲.۹۲$$

این معیار لحظه مرکزی دوم نام دیگری دارد: واریانس. واریانس را می‌توان به شکلی پالایش نمود که یک معیار مهم به نام انحراف استاندارد، با گرفتن ریشه دوم واریانس در اختیارمان قرار دهد. معمولاً این واریانس به‌صورت $\sigma^۲$ نوشته می‌شود. در مثال ما، برای تاس معمولی، انحراف استاندارد ۱،۷ مشخص می‌شود. انحراف استاندارد به ما اطلاع می‌دهد که فاصله متوسطی که مقادیر مورد سنجش قرار گرفته ما

تغییر می‌کنند چقدر با میانگین تفاوت دارد و می‌توان به ما کمک کند که بدانیم داده‌هایمان چقدر متغیر هستند. یک معیار اضافی در رابطه با رابطه مقادیر واقعی در مقایسه با مقادیر مورد انتظار ضریب تغییرات C_x است. ضریب تغییرات به این صورت تعریف می‌شود:

$$C_x = \sigma_x / E[X] \qquad (۷-۴)$$



شکل (۴-۲): تابع چگالی احتمال

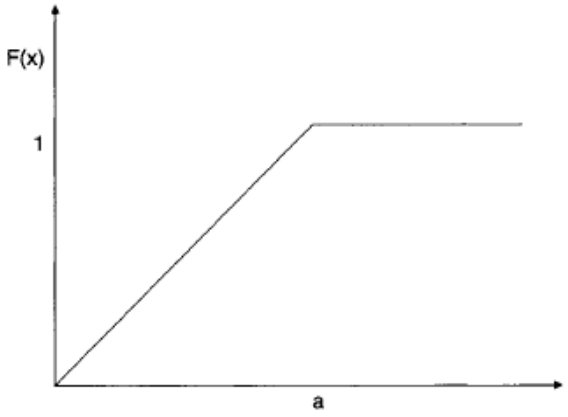
در سیستم‌های کامپیوتری مدل‌سازی آن برای دیدن ضریب معیارهای تغییرات از زیر ۱ تا ۱۰ و بالاتر امکان‌پذیر است. بااین‌حال بیشتر معیارها عمدتاً در جایی بین این مقادیر قرار می‌گیرند.

در مدل‌سازی سیستم‌های کامپیوتری ما اغلب باید نرخ‌های ورود و نرخ‌های سرویس را با استفاده از توابع توزیع مشخص‌سازی نماییم. توزیع‌های نوعی مورداستفاده قرار گرفته شامل توزیع نمایی، توزیع نرمال، توزیع یکنواخت، و توزیع‌های هندسی می‌شوند. ما در مرور انجام‌شده در این فصل به آن‌ها اشاره خواهیم نمود، و درباره جزئیات بیشتر آن در فصل ۵ بحث می‌کنیم.

در زمان نگاهی به مقادیر برای یک نهاد مورد نظر، ما بررسی می‌نمودیم که این مقدار به چه تعداد دفعاتی در مقایسه با تمام مقادیر محتمل رخ می‌دهد. ما تا این مقطع از توزیع احتمال گسسته استفاده نموده‌ایم، زیرا مثال‌های ما مقادیر گسسته‌ای را داشته‌اند. اغلب در سیستم‌های کامپیوتری مقادیر برای یک نهاد

مورد نظر گسسته نخواهند بود، آن‌ها پیوسته خواهند بود. به‌عنوان‌مثال، مقدار زمانی که CPU برای هر کاری که پردازش می‌کند صرف می‌نماید نوعاً از مقادیر واقعی، و نه مقادیر گسسته، تشکیل خواهد یافت. چنان معیارهایی نیازمند آن هستند که احتمال یک مقدار مشخص که ما به آن علاقه‌مندیم بر روی طیف کامل مقادیر احتمالی متغیر خواهد بود. چنان توابع احتمالی پیوسته هستند و توسط توابع توصیف می‌شوند. تابعی که مقادیر احتمال ممکن را برای نهاد مورد نظر ما توصیف می‌کنند تابع چگالی احتمال نامیده می‌شوند (شکل ۲-۴)، در حالی که معیاری که احتمال سیستم‌ها را نشان می‌دهد توسط تابع توزیع احتمال توصیف می‌شود (شکل ۴-۳). تابع تراکم احتمال مقدار واقعی برای احتمال یک نهاد در یک مقطع مشخص در فضای حالت برای آن آیتم را به ما می‌دهد. تابع توزیع یک معیار احتمال در اختیار ما قرار می‌دهد که نشانگر این است که احتمال اینکه یک مقدار کمتر یا برابر با یک مقدار مشخص باشد چقدر است.

برای معیارهایی که ما برای مقادیر مورد انتظار، واریانس و لحظه مرکزی ارائه نمودیم، تغییرات ذیل در فرمول‌ها برقرار خواهند بود.



شکل (۴-۳): تابع توزیع احتمال

برای متوسط:

$$E[X] = \int_{-\infty}^{\infty} xf(x.)dx. \qquad (۸-۴)$$

برای واریانس:

$$E[X^۲] = \int_{-\infty}^{\infty} x^۲ f(x.)dx. \qquad (۹-۴)$$

و برای لحظه مرکزی:

$$\sigma^۲ = \int_{-\infty}^{\infty} (x - E[X])^۲ f(x.)dx. \qquad (۱۰-۴)$$

برای توزیع نشان داده‌شده در شکل (۴-۲)، تابع تراکم احتمال را می‌توان به این صورت توصیف نمود:

$$f(x) = \begin{cases} ۱/a & ۰ \le x. \le a \\ ۰ & \text{در غیر ینصورت} \end{cases} \qquad (۱۱-۴)$$

و برای تابع توزیع احتمال به‌صورت:

$$F(x) = \begin{cases} ۰ & x. < ۰ \\ x./a & ۰ \le x. \le a \\ ۱ & x. > a \end{cases} \qquad (۱۲-۴)$$

مقدار مورد انتظار برای مثال ما را می‌توان به این صورت یافت:

$$E[x] = a/۲ \qquad (۱۳-۴)$$

لحظه دوم برای مثال ما به این صورت مشخص است:

$$E[X^۲] = a^۲/۳ \qquad (۱۴-۴)$$

واریانسی که لحظه مرکزی نامیده می‌شود را می‌توان از این رابطه به دست آورد:

$$\sigma^۲ = a^۲/۱۲ \qquad (۱۵-۴)$$

یکی از مهم‌ترین توزیعات برای مدل‌سازی سیستم‌های کامپیوتری توزیع نمایی (به ویژه نمایی منفی) است. برای توزیع نمایی تابع چگالی احتمال به‌صورت زیر تعریف می‌شود:

$$f(x) = \begin{cases} ۰ & x. < ۰ \\ \lambda e^{-\lambda x.} & x. \ge ۰ \end{cases} \qquad (۱۶-۴)$$

و برای تابع توزیع احتمال به این صورت به دست آورد:

$$F(x_.) = \begin{cases} ۰ & x_. < ۰ \\ ۱ - e^{-\lambda x_.} & x_. \geq ۰ \end{cases} \qquad (۱۷-۴)$$

مقدار مورد انتظار برای توزیع نمایی به این صورت توصیف می‌شود:

$$E[X] = ۱/\lambda \qquad (۱۸-۴)$$

لحظه دوم به این صورت یافته می‌شود:

$$E[X^۲] = ۱/\lambda^۲ \qquad (۱۹-۴)$$

لحظه مرکزی به این صورت یافته می‌شود:

$$\sigma^۲ = ۱/\lambda^۲ \qquad (۲۰-۴)$$

و ضریب تغییرات به این صورت یافته می‌شود:

$$C_x = ۱ \qquad (۲۱-۴)$$

در فصل‌های بعدی، در زمان بررسی سیستم‌های کامپیوتری، ما اهمیت این توزیع را خواهیم دید. این توزیع را می‌توان به طرقی استفاده نمود که ما به تخمین‌های بسیار نزدیکی از عملیات سیستم‌های عمومی دست بیابیم.

۴-۱ الگوریتم‌های زمان‌بندی

در زمان تجزیه‌وتحلیل سیستم‌های کامپیوتری لازم است که نهایتاً به الگوریتم‌های زمان‌بندی اعمال‌شده برای تخصیص منابع توجه شود. در ارزیابی عملکرد یک سیستم کامپیوتری شیوه‌های تخصیص و سپس مصرف منابع واجد اهمیت بسیار بالایی در ارزیابی عملکرد یک سیستم کامپیوتری هستند. به‌عنوان‌مثال، الگوریتم‌های زمان‌بندی در زمان انتخاب اینکه چه برنامه‌ای بر روی یک CPU اجرا می‌شود، به چه دستگاه‌هایی سرویس‌دهی می‌شود، و چه زمانی و چگونه یک دستگاه مشخص درخواست‌های متعدد را هندل می‌کند اعمال می‌شوند. در زمان بررسی الگوریتم‌های زمان‌بندی، باید برای دو مفهوم چاره‌جویی شود. اولی کار عمده الگوریتم زمان‌بندی است، که عبارت است از اینکه چه‌کاری باید برای اجرا در مرحله بعدی انتخاب شود. دومی تعیین این است که آیا وظیفه‌ای که در حال حاضر در حال اجرا است برای اجرا از همه مناسب‌تر است و اگر نه، آیا باید از سرویس خارج شود یا نه.

اساسی‌ترین شکل از الگوریتم زمان‌بندی سرویس‌دهی به ترتیب ورود[۶۳] (FCFS) است. در این الگوریتم زمان‌بندی، کارها وارد سیستم می‌شوند و بر اساس زمان ورودشان بر روی آن‌ها عملیات انجام می‌شود. وظیفه‌ای که اولین ورود را داشته باشد سرویس بعدی را دریافت می‌کند. این الگوریتم پیش‌دستی[۶۴] را به یک وظیفه در حال اجرا اعمال نمی‌کند، زیرا وظیفه در حال اجرا همچنان ممکن است دربرگیرنده معیار برخورداری از اولین زمان ورود باشد. الگوریتم زمان‌بندی‌ای که در مقابل FCFS عمل نماید الگوریتم اولین سرویس‌دهی برای آخرین ورود[۶۵] (LCFS) است. در این الگوریتم، ای که جدیدترین برچسب را دارد برای عملیات انتخاب می‌شود. با داشتن این معیار انتخاب الگوریتم، این امکان وجود دارد که این الگوریتم بتواند پیشدستانه باشد. کاری که سرویس‌دهی می‌شود دیگر آخرین مورد برای ورود برای سرویس نیست. تصمیم پیش‌دستی باید بر مبنای قابلیت منبع برای توقف در بطن جریان باشد و در یک مقطع زمانی دیگر از نو شروع شود. پردازنده‌ها نوعاً می‌توانند پیش‌دستانه باشند، زیرا تسهیلاتی برای ذخیره‌سازی ثبات‌ها و اطلاعات دیگر موردنیاز برای راه‌اندازی مجدد یک شغل در یک زمان دیگر وجود دارد. دستگاه‌های دیگری همچون یک دیسک درایو یا کانال ۱۱۰، ممکن است قابلیتی برای توقف یک کار نداشته باشند و آن را در یک مقطع زمانی دیگر انتخاب نمایند.

یک عدد و نوع از الگوریتم‌های زمان‌بندی به زمان‌بندی پردازنده ارتباط داده می‌شوند. یکی از رایج‌ترین الگوریتم‌های زمان‌بندی پردازنده "نوبت برگشتی[۶۶]" است. زمان‌بندی نوبت برگشتی یک الگوریتم ترکیبی است، که از زمان‌بندی FCFS، در کنار پیش‌دستی، استفاده می‌کند. سرویس این پردازنده به دسته‌های زمانی به نام کوانتوم تقسیم می‌شود. اگر زمان سرویس موردنیاز آن‌ها از این فراتر رود، این کار پیش‌دستی شده و در پست مجموعه کارهای معلق قرار می‌گیرد. حرکت قرار دادن یک کار در پست لوله زمان‌بندی FCFS تا زمانی ادامه می‌یابد که کار نهایتاً تمام شود. از این‌رو، زمان سرویس کار به چند برش زمانی برابر با اندازه ثابت تقسیم می‌شود. مسئله عمده‌ای که با زمان‌بندی نوبت برگشتی وجود دارد انتخاب اندازه کوانتوم است. دلیل اهمیت بالای انتخاب اندازه کوانتوم به طبیعت پیش‌دستی

[۶۳] first-come first-served
[۶۴] preemption
[۶۵] last-come first-served
[۶۶] round robin

برمی‌گردد. پیش‌دستی یک کار نیاز به بالاسری از سیستم‌عامل برای توقف کار، ذخیره‌سازی حالت آن، و نصب یک کار جدید دارد. اگر زمان برای کار وظایف در مقایسه با اندازه کوانتوم قابل توجه باشد، عملکرد آسیب خواهد دید. بسیاری از قواعد مختلف کلی در طراحی چنان سیستم‌هایی توسعه داده‌شده‌اند. بیشتر آن‌ها به دنبال این هستند که بالاسری را به کسر بسیار کوچکی از اندازه کوانتوم تبدیل نمایند که معمولاً چند مرتبه کوچک‌تر است. یک روش مورداستفاده برای تخمین زمان‌بندی نوبت برگشتی در زمانی که کوانتوم در مقایسه با بالاسری خیلی بزرگ باشد اشتراک‌گذاری پردازنده (PS) است. این مدل از زمان‌بندی نوبت برگشتی در تحلیل تئوریک استفاده می‌شود، که ما در فصل‌های بعدی خواهیم دید.

یک الگوریتم دیگر کوتاه‌ترین زمان باقیمانده[۶۷] (SRTF) است. در این الگوریتم، کاری که نیاز به کمترین مقدار از مان منبع دارد به‌عنوان کار بعدی برای سرویس انتخاب می‌شود. الگوریتم زمان‌بندی CPU، SRTF پیش‌دستی را پشتیبانی نمی‌کند. وقتی مشخص شود که یک کار ورودی دارای یک زمان اجرای تخمینی کوچک‌تر از کاری باشد که در حال حاضر در حال اجرا است، کار در حال اجرا پیش‌دستی شده و با کار جدید جایگزین می‌شود. مشکلی که با این الگوریتم زمان‌بندی وجود دارد این است که باید الزامات پردازشی هر کار به‌صورت پیش از موعد معلوم شود، که نوعاً پیش نمی‌آید، و به علت محدودیتش اغلب استفاده نمی‌شود. بااین‌حال، این الگوریتم بهینه است و به‌صورت مقایسه‌ای با دیگر الگوریتم‌های کاربردی‌تر استفاده می‌شود.

یک الگوریتم مفید مرتبط با SRTF الگوریتم ارزش محور است، که در آن هم زمان اجرا و هم ارزش تکمیل کار درون چارچوب زمانی به‌صورت زودهنگام معلوم هستند. این کلاس از الگوریتم در سیستم‌های بی‌درنگ سررسیدمحور یافته می‌شود. این الگوریتم کار بعدی‌ای که باید انجام دهد را بر مبنای نزدیک بودن به سررسیدش و محاسبه مقداری که در صورتی که اکنون انجام دهد انتخاب می‌کند. این الگوریتم این از این نظر پیشدستانه است که یک کار در حال اجرا را، در صورتی که کار رقیب به سررسیدش نزدیک‌تر باشد و یک ارزش نسبی بالاتر داشته باشد، از پردازنده حذف می‌کند. نکته مورد توجه در این کلاس‌های الگوریتمهای زمان‌بندی این است که آن‌ها پشتیبانی را برای بحرانی‌ترین عملیات به قیمت بازدهی کلی ارائه می‌نمایند.

---

[۶۷] shortest remaining time first

٤-١-١ رابطه بین زمان‌بندی و توزیع‌ها

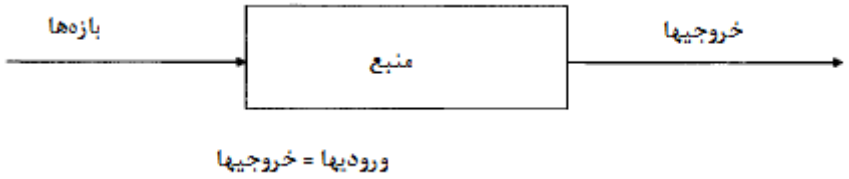
در تعیین عملکرد یک سیستم کامپیوتری، معیار معمول بازدهی است. در بحث‌های بعدی ما این را تعداد متوسط کارهایی لحاظ می‌کنیم که از یک مقطع مورد نظر در معماری ما در طی یک بازه زمانی عبور می‌کنند، به‌عنوان‌مثال، تعداد کارهایی که در هر دقیقه از CPU خارج می‌شوند. در بیشتر موارد ما مقدار ماکسیمم را در زمانی که منابعمان به طور کامل مورداستفاده قرار گرفته باشند (مشغول) برای بازدهی درک می‌کنیم.

در بخش قبلی، ما معیارهایی را ارائه نمودیم که اکنون می‌توانیم استفاده نماییم. ضریب تغییراتی که قبلاً تعریف شد یک راه خوب بررسی تغییرپذیری در داده‌های ما است. اگر زمان‌های سرویس بسیار متغیر باشند، $C > ۱$، بیشتر معیارها هوشمندتر از متوسط خواهند بود و برخی بزرگ‌تر خواهند بود. به‌عنوان‌مثال، در توزیع نمایی، $C = ۱$، می‌توان از تابع تراکم احتمال دریافت که حدود ۶۳ درصد از مقادیر زیر متوسط هستند. چنان تغییرپذیری‌ای می‌تواند باعث مشکلاتی با الگوریتم‌های زمابندی مشخص شود – به‌عنوان‌مثال، الگوریتم زمان‌بندی FCFS، زیرا کارهایی که نیاز به الزامات قابل توجه منابع هستند موجب افزایش تأخیرها برای اکثر کارهایی خواهند شد که کوچک‌تر از متوسط خواهند بود. این اثر ممکن است با دیگر منابع وابسته به منبع زمان‌بندی‌شده FCFS آمیخته‌تر شود. به‌عنوان‌مثال، اگر کارها در انتظار سرویس CPU جمع شوند، منابع دیگر همچون دیسک درایوها و دستگاه‌های I/O ممکن است بیکار بمانند.

یک الگوریتم زمان‌بندی که به این پدیده خیلی حساس نیست پروتکل زمان‌بندی نوبت برگشتی است. از آنجا که هیچ کاری، چه بزرگ و چه کوچک، نمی‌تواند منابعی بزرگ‌تر از یک کوانتوم واحد را در هر زمان نگه دارد، کارهای بزرگ‌تر کارهای کوچک‌تر را از رده خارج نخواهد کرد. این واقعیت باعث می‌شود که پروتکل زمان‌بندی نوبت برگشتی یک الگوریتم مناسب برای سنجش استفاده از منابع با بارهای متغیر شود. اگر قرار بر این باشد که ما پروتکل‌های زمان‌بندی نوبت برگشتی و FCFS، برای بارهای بسیار متغیر و بسیار مرتبط، را با یکدیگر مقایسه کنیم، مشاهده خواهیم نمود که با به هم پیوسته‌تر شدن بارها الگوریتم‌ها به شکلی مشابه‌تر به اجرا در می‌آیند. از سوی دیگر، با متغیرتر شدن داده‌ها، پروتکل زمان‌بندی نوبت برگشتی کارایی بهتر از FCFS خواهد داشت.

٤-١-٢ رابطه با عملکرد سیستم‌های کامپیوتری

برای مدل‌سازی سیستم‌های کامپیوتری و اجزائشان ما معمولاً به تعیین بازدهی، استفاده، و زمان‌های متوسط سرویس برای هر یک از عناصر مورد علاقه بر روی طیف وسیعی از بارها علاقه‌مند خواهیم بود. تحلیل از یک چشم‌انداز تئوریک همیشه تعادلی را کسب خواهد نمود که حاصل شده است، که به طور ضمنی نشانگر آن است که تعداد ورودی‌ها به یک منبع برابر با تعداد خروجی‌ها از آن منبع است (شکل ۴-۴).



شکل (۴-۴): منابع در تعادل

جریان خروجی از بازدهی‌اش نامیده می‌شود. زمان سرویس متوسط، همان‌گونه که قبلاً تعریف شد، $E[X]$ است، و نرخ سرویس متوسط $1/E[X]$ است. بهره‌برداری برای این منبع به‌صورت کسری از زمانی تعریف می‌شود که منبع مشغول است (U). بازدهی منبع باید برابر با نرخ سرویس منبع، در زمانی که مشغول است، ضرب در کسر زمانی که مشغول است باشد.

این را می‌توان به این صورت نمایش داد:

$$\text{بازدهی} = U/E[X] \qquad (۲۲-۴)$$

اگر ما n دستگاه یکسان در سیستممان داشته باشیم، به‌عنوان‌مثال، چند CPU با خصوصیات یکسان – بازدهی برای این‌ها را می‌توان به این صورت توصیف نمود:

$$\text{بازدهی} = nU/E[X] \qquad (۲۳-۴)$$

این روابط ساده بین استفاده و زمان مورد انتظار سرویس معیارهای مهم در تحلیل عملکرد سیستم‌ها خواهد بود، که در فصل‌های بعدی مشاهده خواهیم نمود.

مدل‌کننده اندازه مجموعه سرویس انتظار کارها در یک منبع و زمانی که این کارها صرف انتظار برای سرویسشان می‌کنند نیز برای قابل توجه است. اول، ما نیاز به تعریف طول صف منبع داریم. این به‌صورت متوسط تعداد کارهایی تعریف می‌شود که در طی طول عمر این منبع منتظر سرویس هستند. از جنبه تئوریک، این را می‌توان با استفاده از احتمال داشتن n کار در انتظار ضرب در تعداد کارها برای تمام مقادیر n یافت:

$$L_q = E[L_q] = \sum_{n=1}^{\infty} n\, P(n) \qquad (۲۴-۴)$$

که در آن $P(n)$ نشانگر این احتمال است که طول صف منبع n است. زمان صف بندی متوسط (زمان انتظار منبع) را می‌توان از این رابطه یافت:

$$T_q = E[q] = \int_{.}^{\infty} q.\, f_q(q.)dq. \qquad (۲۵-۴)$$

که در آن $f_q(q.)$ تابع تراکم احتمال را برای زمان‌های صف بندی منبع نشان می‌دهد. از این مشاهدات، می‌توان نشان داد که:

$$L_q = \lambda T_q \qquad (۲۶-۴)$$

این فرمول نشانگر آن است که ما می‌توانیم طول صف را بیابیم چرا که ما زمان صف بندی متوسط و نرخ ورودی‌های لاندا (یا آیتم‌های سرویسدهی‌شده) را برای منبع مورد نظر می‌دانیم. این مشاهده ساده توسط J.D. Little کشف شد و با عنوان قانون لیتل مورد ارجاع قرار می‌گیرد. در فصل‌های بعدی بیشتر درباره این فرمول و کاربرد آن برای تحلیل عملکرد سیستم‌های کامپیوتری صحبت خواهد شد.

۴-۲ بارهای کاری

برای اینکه بتوانیم سیستم‌ها را مدل‌سازی و آنالیز کنیم، لازم است تا ابزارهایی برای تست و/یا تحت تنش قرار دادن سیستم‌های مورد نظر توسعه دهیم. این ابزارهای تست یا استرس سیستم‌ها بارهای کاری نامیده می‌شوند. یک بار کاری باید به شکلی توسعه داده شود که به درستی طبیعت بار درست را بر روی سیستم مورد نظر مدل‌سازی کند. بارهای کاری بر مبنای تمرکز تحلیل ساخته می‌شوند. به‌عنوان‌مثال، اگر ما علاقه‌مند به بررسی سخت‌افزار یک سیستم کامپیوتری در مقایسه با دیگران باشیم، تمرکزمان می‌تواند بر دستورالعمل‌های سطح پایین باشد. ممکن است لازم شود که ما ترکیب دستورالعملی مشاهده‌شده بر روی سیستم در حال کار را مورد سنجش قرار دهیم و سپس یک ترکیب سنتزی از دستورالعمل‌ها را بر

مبنای این معیارها توسعه دهیم. یک مثال دیگر سنجش بازدهی تراکنشی از طریق یک سیستم پایگاه داده است. این بار کاری می‌تواند دارای واحدهای تراکنشی کاری باشد که آیتم‌های داده‌ها را از سیستم پایگاه داده خوانده و می‌نویسند، و چند کار محاسباتی اضافی شبیه سیستم واقعی انجام می‌دهند. جامعه پایگاه داده چنان بارهای کاری را برای پایگاه داده‌های متداول، پایگاه داده‌های رابطه‌ای شیی، انبارهای داده‌ها، و داده‌کاوی توسعه داده است. این بارهای کاری معیارهای TP نامیده می‌شوند.

در زمان طراحی یک بار کاری، مهم است که ما درک کنیم که چگونه بار کاری بار سیستم مورد نظر را کاهش خواهد داد. صرف ارائه یک بار مشخص کفایت نمی‌کند، بار باید ابزارهایی برای استرس وارد کردن به سیستم در حال آنالیز فراهم بیاورد. ما بارهای کاری را می‌خواهیم که موجب به اشباع رسیدن سیستم‌های سنجیده شوند. ما می‌خواهیم که ببینیم سیستم در کجا وارد استفاده ۱۰۰ درصدی از منابع می‌شود و چنان بارهایی برای یک دوره زمانی حفظ می‌شوند.

وقتی یک بار کاری توسعه داده شد، ما نیاز به استفاده از شکلی از تابع توزیع برای انتخاب آیتم‌ها از بارکاری‌مان و ارائه آن‌ها به سیستم برای سرویس خواهیم داشت. به‌عنوان‌مثال، اگر ما n نوع تراکنش را برای تحلیل سیستم پایگاه داده‌مان توسعه داده باشیم، ممکن است مایل باشیم که از یک توزیع یکنواخت برای انتخاب آیتم‌های داده‌های پایگاه داده برای تراکنش‌هایی که باید بر رویشان عملیات انجام شود استفاده کنیم و از توزیع نمایی برای ارائه تراکنش‌ها به سیستم برای پردازش استفاده کنیم. بار کاری ما، با استفاده از این ابزارها به شکلی برای سیستممان ارائه خواهد شد که شبیه وضعیت دنیای واقعی باشد، ولی سیستمی که ما کنترل کاملی بر آن داریم.

این لازم است که بارهای کاری به شکلی توسعه داده شوند که ما اجزائی را تست کنیم که خواستار تست کردنشان هستیم. به‌عنوان‌مثال، اگر ما سیستم پایگاه داده را تست کنیم، یک بار کاری ترکیب دستورالعمل ساده نوعی از اطلاعات که برایمان جالب توجه باشد را ارائه نخواهد نمود. دستورالعمل‌ها به تنهایی نشانگر بار مطلوب نیستند. تراکنش‌ها از مرزهای تراکنش، دسترسی پایگاه داده و فرمان‌های تغییر[۶۸] و فرمان‌های دست‌کاری داده‌ها تشکیل یافته‌اند. در فصل‌های بعدی اطلاعات بیشتری درباره بارهای کاری ارائه خواهد شد.

[۶۸] alteration commands

۳-۴ خلاصه

در این فصل، ما چند مفهوم اساسی موردنیاز برای ارزیابی عملکرد سیستم‌های کامپیوتری را ارائه نمودیم. مفهوم منابع مرتبط با زمان سرویس، به همان صورت موجود برای برخی از مفاهیم پایه برای استفاده از این معیارها معرفی می‌شود. مفاهیم اساسی دخیل در توابع توزیع احتمال و تراکم احتمال توسعه داده می‌شوند. این‌ها سپس برای توسعه تعاریف اساسی برای مقادیر مورد انتظار یا متوسط، n امین لحظه توزیع، استفاده می‌شوند.لحظه دوم ویژه، که لحظه مرکزی نامیده می‌شود، همانند تعریف و فرمولاسیون برای واریانس و ضریب تغییرات توسعه داده می‌شود. این بحث سپس به الگوریتم های زمان‌بندی مورداستفاده در سیستم‌های کامپیوتری تغییر یافت. ما الگوریتم های زمان‌بندی تابع مقدار سررسیدمحور، زمان‌بندی اولویتی، نوبت برگشتی، سرویس دهی به ترتیب ورود، و سرویس دهی از آخرین ورودی را معرفی می کنیم.

پس از آن نوبت به بحث درباره نحوه ارتباط یافتن این آیتم ها با مدل‌سازی سیستم‌های کامپیوتری می‌رسد. رابط‌های که الگوریتم‌های زمان‌بندی با توابع توزیع دارند و این رابطه‌شان با کارایی ارائه می-شود. آخرین مبحث ارائه شده به بارهای کاری مربوط می‌شود. مفهوم اینکه یک بار کاری چیست و نحوه استفاده از آن در مدل‌سازی سیستم‌های کامپیوتری از یک نظرگاه مقدماتی ارائه می‌شود.

فصل پنجم

احتمال

تئوری صف بندی و تحلیل صف بندی مبتنی بر استفاده از تئوری احتمال و مفهوم متغیرهای تصادفی هستند. ما مفاهیم موجود در احتمال را به چند شیوه مختلف مورداستفاده قرار می‌دهیم. به‌عنوان‌مثال، ما می‌توانیم این احتمال را بپرسیم که احتمال اینکه بوستون برونز امسال برنده استنلی کاپ شود چقدر است. احتمال اینکه جورج دبلیو بوش پس از رویدادهای امسال مجدداً انتخاب شود چقدر است؟ احتمال بارش برف در کوه واشینگتن در نیوهمپشایر در ژانویه امسال قدر است؟ در بیشتر اوقات یک پاسخ کلی کفایت می‌کند. به‌عنوان‌مثال، احتمال زیادی وجود دارد که در مقطعی در ژانویه در کوه واشینگتن ببارد. در مقابل، بر مبنای وضعیت بد ۳۰ سال اخیر بوستون برونز هم احتمال اینکه استنلی کاپ را ببرد بسیار بعید است. تئوری احتمال برای ما امکان ارائه تعاریف دقیق‌تر را برای احتمال وقوع یک رویداد بر مبنای سوابق گذشته یا سنجش‌های خاص در دسترس، به شکلی خواهیم دید، فراهم خواهد آورد. در این فصل، ما مفاهیم احتمال، احتمال مشترک، احتمال مشروط و استقلال را ارائه خواهیم نمود. سپس به سراغ توزیع‌های احتمال می‌رویم، و، نهایتاً ، مبانی صف بندی را بررسی خواهیم نمود.

پیش از بحث درباره تحلیل صف بندی ، ارائه برخی از مفاهیم از تئوری احتمال و آمار ضروری است. در تئوری احتمال اساسی، ما کار را با ایده‌های رویدادهای تصادفی و فضاهای نمونه آغاز می‌کنیم. به‌عنوان‌مثال آزمایشی را در نظر بگیرید که دربرگیرنده غلتاندن تاس معمولی باشد (آزمایشی که نوعاً رویه‌ای را تعریف می‌کند که یک نتیجه ساده را حاصل می‌کند، که می‌توان یک احتمال وقوع به آن اختصاص داد). فضای نمونه یک آزمایش صرفاً مجموعه‌ای از نتایج محتمل است – که در این حالت مجموعه $\{۱،۲،۳،۴،۵،۶\}$ برای تاس است. یک رویداد به‌صورت زیرمجموعه‌ای از یک فضای نمونه تعریف می‌شود و ممکن است هیچیک، یکی، یا چند تا از عناصر فضای نمونه باشد. در آزمایش‌های تاس، یک رویداد می‌تواند وقوع یک ۲ باشد یا اینکه عددی که ظاهر می‌شود فرد باشد. از این‌رو، فضای

نمونه دربرگیرنده تمام نتایج منفرد یک آزمایش است. برای اینکه جمله قبلی برقرار باشد، ضروری است که تمام نتایج احتمالی یک آزمایش معلوم باشند.

این اصل اساسی احتمال بیان می‌کند که شانس وقوع یک نتیجه خاص با نسبت تعداد نتایج مطلوب (موفقیت‌ها) به تعداد کل نتایج (فضای نمونه) تعیین می‌شود. A که در فرمول برخی از رویدادها به کار می‌رود نتیجه می‌دهد:

$$P(A) = \frac{\text{تعداد نتایج موفق}}{\text{تعداد کل نتایج محتمل}}$$

(۱-۵)

از مثال قبلی می‌دانیم که احتمال غلتاندن یک تاس، که به‌صورت $P(۲)$ بیان می‌شود، با استفاده از یک تاس معمولی، معادل با نسبت ۱/۶ یا ۰,۱۶۷ است. در این آزمایش، ۱ نشانگر نتیجه مطلوب یا موفقیت آزمایش ما است، و ۶ نشانگر تعداد کل نتایج ممکن از غلتاندن تاس است. به همین صورت، ما می‌توانیم احتمال غلتاندن یک عدد فرد را (۱، ۳ یا ۵) را بیان کنیم، که به‌صورت $P(\text{odd})$، به شکل ۳/۶ یا ۰,۵، بیان می‌شود. در این آزمایش ۳ نشانگر تعداد نتایج ممکنی است که نشانگر نتایج مطلوب برای این آزمایش است، و ۶ نشانگر تعداد کل نتایج احتمالی از غلتاندن تاس است. یک مثال دیگر با استفاده از کارت‌های بازی می‌تواند بیشتر این تعریف را پالایش نماید. اگر ما کارت‌ها را خوب بر زده باشیم و ۵۲ کارت ممکن داشته باشیم که می‌توان هر یک از آن‌ها را بیرون کشید، و ما بخواهیم که احتمال بیرون کشیده شدن شاه را بدانیم، این را می‌توان به این صورت بیان نمود:

$$P\left(\text{شاه آمدن}\right) = \frac{\text{تعداد کل نتایج موفق (مثلا یک شاه=۴)}}{\text{تعداد کل نتایج (مثلا ۵۲ کارت در دسته)}} = ۴/۵۲ \text{ یا } ۰.۰۷۷ \qquad (۲-۵)$$

می‌توان احتمال آمدن شاه دل را به این صورت پرسید:

$$P\left(\text{شاه آمدن}\right) = \frac{\text{تعداد کل نتایج موفق (مثلا تعداد شاه دل=۱)}}{\text{تعداد کل نتایج (مثلا ۵۲ کارت در دسته)}} = ۱/۵۲ \text{ یا } ۰.۰۱۹ \qquad (۳-۵)$$

در تمام موارد، مقدار برای احتمال وقوع یک رویداد در یک محدوده رویدادهای احتمالی باید از ۰، که در آن این رویداد رخ نمی‌دهد (مثلاً غلتاندن یک صفر با یک تاس، زیرا هیچ صفری روی تاس وجود ندارد، این امکان‌پذیر نیست)، تا ماکسیمم ۱ (مثلاً احتمال آمدن یک عدد فرد یا زوج با تاس) را در بر بگیرد نشانگر آن است که رویداد همیشه رخ می‌دهد.

در مثال‌های ذکر شده، هر عدد روی تاس باید یک شانس برابر غلتانده شدن داشته باشد. به همین صورت، در مثال کارتها، هر کارت باید یک شانس برابر بیرون کشیده شدن داشته باشد. هیچ عددی روی صورت تاس یا کارت در دسته را نمی‌توان به شکلی متمایزسازی نمود که احتمال بیشتری برای انتخاب شدن یا غلتانده شدن آن وجود داشته باشد (مثلاً یک تاس موزون یک تاس معمولی نیست).

تئوری احتمال بر مبنای مفهوم انحصار متقابل (منفصل بودن) رویدادها ساخته شده بود. بدین معنی که رویدادها نمی‌توانند به همزمان در یک آزمایش رخ دهند اگر متقابلاً منحصر باشند. به‌عنوان‌مثال، غلتاندن یک تاس معمولی می‌تواند منجر به یکی از شش نتیجه محتمل شود، ولی نه دو یا چند تا از آن‌ها به طور همزمان. یک مثال دیگر یک سکه معمولی است، انداختن سکه منتج به آمدن شیر یا خط می‌شود ولی امکان اینکه هر دو همزمان بیایند وجود نخواهد داشت. از این‌رو، نتیجه رویداد غلتاندن یک تاس و دستیابی به یک ۱ در مقایسه با تمام اعداد دیگر منحصراً متقابل نامیده می‌شود، همان‌طور که انداختن سکه منجر به آمدن شیر یا خط می‌شود و نه هر دو باهم.

یک ویژگی مهم دیگر در حوزه احتمال به استقلال مربوط می‌شود. به‌عنوان‌مثال اگر ما یک سکه و یک تاس داشته باشیم، به طور شهودی درک خواهیم کرد که رخداد انداختن سکه و غلتاندن تاس نتایجی مستقل از هم دارند. بدین معنی که نتیجه یکی بر نتیجه دیگری مؤثر نخواهد بود. فضای نمونه برای این دو رویداد از محصول کارتیزین دو فضای مستقل تشکیل یافته است، زیرا همه رویدادهای هر دو به طور مستقل امکان‌پذیر هستند، فضای نمونه حاصله باید دربرگیرنده تمام ترکیبات ممکن از دو فضای مستقل باشد.

اگر ما بر این باور باشیم که این ویژگی احتمال برابر وجود دارد، آنگاه احتمال هر یک از این نتایج باید برابر بوده و متشکل از احتمال ضربی هر یک به طور مستقل باشد. در مثال قبلی، انداختن سکه معمولی دارای یک فضای نمونه متشکل از عناصر مجموعه $\{H, T\}$ است، که هر یک دارای یک احتمال ۱/۲ هستند، و تاس معمولی یک فضای نمونه متشکل از عناصر مجموعه $\{۱،۲،۳،۴،۵،۶\}$, باشد،که هر یک دارای یک احتمال ۱/۶ هستند.

احتمال وقوع ترکیبی از هر یک از این رویدادها را می‌توان از مجموعه حاصل‌ضرب دکارتی $\{H۱, H۲, H۳, H۴, HS, H۶, T۱, T۲, T۳, T۴, TS, T۶\}$, با هر یک از این رویدادهای ترکیبی استخراج کرد، که یک معادل احتمال برابر با ۱/۶ × ۱/۲ یا ۱/۱۲ را می‌دهد.

به طور کلی، وقتی رویدادها مستقل باشند، فضاهای نمونه که در آن‌ها هر رویداد رخ می‌دهد به‌اندازه‌ای برابر محتمل خواهد بود. اگر n_1 آیتم در فضای رویداد اول، n_2 در دومی، و n_m در فضای نمونه آخر وجود داشته باشد، آنگاه فضای نمونه فضای رویدادهای ترکیبی برابر با مجموع اندازه هر یک از این فضاها است:

$$\text{فضای نمونه ترکیبی} = (n_1 + n_2 + \cdots + n_m) \qquad (۴-۵)$$

از این‌رو احتمال وقوع هر اتفاق برابر با $1/(n_1 + n_2 + \cdots + n_m)$ است، که در مثال قبلی، ۱/۱۲ مشخص گردید.

ما همیشه به احتمال تنها یک رویداد از وقوع یک فضای نمونه کامل علاقه‌مند نخواهیم بود بلکه در عوض به زیرمجموعه‌ای از رویدادها از فضای کلی توجه خواهیم داشت. به‌عنوان‌مثال، ما ممکن است به احتمال وقوع تنها یک شیر در طی انداختن چهار سکه علاقه‌مند باشیم. فضای نمونه کامل این آزمایش دقیقاً ۱۶ نتیجه محتمل دارد:

$$\{HHHH, HHHT, HHTH, HTHH, THHH, TTHH, THHT, HHTT, THTH, HTHT, HTTH, HTTT, THTT, TTHT, TTTH, TTTT\}$$

از این فضا، ما می‌توانیم ببینیم که رویدادهایی که نتیجه مطلوب فقط یک شیر را حاصل می‌کنند منتج به این زیرمجموعه می‌شوند:

$$\{HTTT, THTT, TTHT, TTTH\}$$

که در آن احتمال وقوع هر آیتم در این زیرمجموعه از مجموعه اصلی به یک اندازه است، از این‌رو هر یک دارای احتمال برابر ۱/۱۶ هستند. احتمال ترکیبی آن‌ها می‌تواند احتمال مطلوب وقوع تنها یک شیر را نمایندگی کند و می‌تواند برابر با مجموع احتمالشان باشد: ۴/۱۶ یا ۱/۴. در این مثال، ما از احتمالات افزایشی این رویدادها برای دیدن احتمال وقوع یکی از این این‌ها از مجموعه اصلی استفاده می‌کنیم. برای محاسبه احتمال زیرمجموعه، ما تنها باید اندازه مجموعه اصلی و اندازه زیرمجموعه را بدانیم.

در بسیاری از موارد، ما با مسئله تعیین احتمال وقوع یک رویداد با توجه به اینکه یک رویداد قبلی پیش از این رخ داده است مواجهیم. به‌عنوان‌مثال ممکن است از ما احتمال از کار افتادن سیستم کامپیوتریمان، با توجه به اینکه یک تراشه حافظه خراب شده باشد، پرسیده شود. این مفهوم رویدادهای مرتبط یا وابسته احتمال شرطی نامیده می‌شود. اثر اعمال این ویژگی بر دو فضای رویداد مستقل حذف برخی از ترکیبات

محتمل از فضای ترکیبی نهایی ارزش‌های محتمل است. به‌عنوان‌مثال، ممکن است از ما پرسیده شود که احتمال آمدن دقیقاً یک شیر پس از آنکه مشخص شد که عنصر اول، در چهار بار انداختن سکه، خط است چقدر است. فضای نمونه اولیه ۱۶ بود، ولی با توجه به اینکه ما از رویدادهایی خارج شده‌ایم که در آن‌ها پرتاب اولیه منجر به یک خط می‌شود فضای حاصله اکنون تنها هشت نتیجه احتمالی دارد و از این‌ها تنها سه تایشان در زیرفضای منطبق با نتیجه مطلوب نهایی ما هستند.

برای محاسبه اینکه در این حالت چه احتمالی وجود دارد، ما می‌توانیم چند کار انجام دهیم. اول، ما می‌توانیم احتمال خط آمدن در پرتاب اول را ۸/۱۶ و احتمال وجود یک شیر در سه پرتاب آخر را ۳/۸ محاسبه کنیم. در بسیاری از موارد محاسبه رخداد مقابل آسان‌تر است. بدین معنی که بیایید این احتمال که رویداد نهایی رخ ندهد را محاسبه کنیم. ابتدا ما می‌توانیم این‌گونه استدلال کنیم که فضای نمونه اکنون تمام رویدادهایی را نادیده می‌گیرد که در آن‌ها آیتم اول شیر بوده است، یا هشت رویداد. اکنون تنها ۸ — ۱۶ یا ۸ رویداد با احتمال برابر وجود دارند که در فضایمان باقی می‌مانند. از این ۸ تای باقیمانده، پنج راه وجود دارد که در آن، با خط آمدن در دفعه اول، ما دقیقاً یک شیر نمی‌آوریم. از این‌رو احتمال آمدن حداقل یک شیر نسبت تعداد موفقیت‌ها (۳) به تعداد کل رویدادهای محتمل (۸/۱۶) یا ۸ است، که منتج به یک احتمال ۳/۸ می‌شود.

چند عملیات در رویدادها در فضای نمونه ویژگی‌های مهم رویدادها را نتیجه می‌دهند. بر حسب تعریف، تقاطع دو رویداد مجموعه‌ای است که دربرگیرنده تمام عناصر مشترک برای هر دو رویداد است. تقاطع مجموعه‌های A و B به‌صورت AB نوشته می‌شود. با بسط آن می‌توان گفت که تلاقی چند رویداد دربرگیرنده عناصر مشترک برای هر رویداد است. دو رویداد در صورتی متقابلاً منحصر به فرد خوانده می‌شوند که تلاقی‌شان مجموعه خنثی را نتیجه دهد. اجتماع دو رویداد مجموعه تمام عناصری را نتیجه می‌دهد که در یکی از رویدادها یا در هر دو رویداد هستند. اجتماع مجموعه‌های A و B به‌صورت $A \cup B$ نوشته می‌شود.

مکمل یک رویداد، که با $\overline{A}$ نشان داده می‌شود، نشانگر تمام عناصر به‌جز موارد تعریف‌شده در رویداد A است. تعریف ذیل، که با عنوان قانون دمورگان شناخته می‌شود، برای ارتباط دهی مکمل‌های دو رویداد مفید است:

$$\overline{AB} = \overline{A} \cup \overline{B} \qquad (۵-۵)$$

جایگشت‌ها و ترکیب‌های عناصر در یک فضای نمونه ممکن است شکل‌های متفاوت بسیاری به خود بگیرد. اغلب، ما می‌توانیم معیارهای احتمالی را درباره ترکیبات نقاط نمونه شکل دهیم، و جایگشت‌ها و ترکیب‌های اساسی مورد بحث قرار گرفته در نوشتار بعدی در این کتاب کاربرد می‌یابند. بر حسب تعریف، یک ترکیب عبارت است از یک انتخاب نامرتب از آیتم‌ها، که در آن یک جایگشت یک انتخاب مرتب از آیتم‌ها است. اساسیترین ترکیب دربرگیرنده وقوع یکی از $n_۱$ رویداد است، که پس از آن $n_۲$ رویداد می‌آید، و به همین صورت تا یکی از n_k ادامه می‌یابد. از این‌رو، برای هر مسیر در پیش گرفته شده برای رسیدن به آخرین رویداد، یعنی k، n_k انتخاب ممکن وجود دارد. با پشتیبان گیری از یک سطح، $n_{k-۱}$ انتخاب در آن سطح وجود داشت، و بدین طریق n_k $n_{k-۱}$ انتخاب برای دو رویداد آخر وجود دارد. تبعیت از پشتیبان گیری منطقی مشابه با سطح اول این تعداد مسیر ممکن را نتیجه می-دهد:

$$n_۱n_۲ \dots n_{k-۱}n_k = \prod_{i=۱}^{k} n_i \qquad (۶-۵)$$

به‌عنوان‌مثال، در یک رشته پنج رقمی، که هر یک از آن‌ها ممکن است مقادیر ۰ تا ۹ به خود بگیرند، ۱۰×۱۰×۱۰×۱۰×۱۰=۱۰۰,۰۰۰ ترکیب محتمل، یا n^k نتیجه محتمل، وجود دارد، که در آن $n = ۱۰$ و $k = ۵$ در فضای نمونه هستند. فرض بر این است که وقتی یک آیتم نمونه‌برداری شد، برای نمونه‌برداری احتمالی به فضا برگردانده می‌شود. این گاهی با عنوان نمونه‌برداری با جایگزینی مورد ارجاع قرار می‌گیرد. توزیع احتمال برای یک انتخاب تصادفی آیتم‌ها یکدست و یکنواخت است، از این‌رو، هر آیتم در فضای نمونه احتمال $۱/n^k$ خواهد داشت.

در زمان پرداختن به عناصر منحصربه‌فردی که ممکن است به طرق متفاوتی ترتیب بیابد، ما درباره جایگشت‌ها صحبت می‌کنیم. وقتی ما n چیز داشته باشیم و از n آیتم نمونه‌برداری کنیم، ولی آیتم‌های نمونه را از انتخاب بعدی جایگزین نکنیم، اکنون منتخبی بدون جایگزینی خواهیم داشت، که با عنوان جایگشت نیز مورد ارجاع قرار می‌گیرد. برای n شیء گسسته، اگر ما یکی را انتخاب کنیم و آن را کنار بگذاریم، در این صورت $n - ۱$ انتخاب برایمان وجود خواهد داشت. تکرار این اقدام باعث می‌شود که $n - ۲$ انتخاب وجود داشته باشد، و به همین صورت تا ۱. تعداد جایگشت‌های مختلف این nعنصر تعداد انتخاب‌هایی است که شما می‌توانید در هر گام در فرایند انتخاب و کنار گذاشتن انجام دهید و برابر با این رابطه است:

$$p(n,n) = (n)(n-۱)(n-۲)\dots(۲)(۱) = n! \qquad (۷-۵)$$

نمادگذاری مشترک $p(n,k)$ نشانگر تعداد جایگشت‌های n آیتمی است که در هر بار k را بر می-دارند. برای یافتن مقدار عددی برای یک انتخاب تصادفی ما می‌توانیم از منطق مشابهی استفاده کنیم. آیتم اول را می‌توان به n طریق انتخاب نمود، دومی به $n-۱$ حالت، سومی در $n-۲$ حالت، و k امین یا آخرین آیتم را ما به $n-k-۱$حالت انتخاب می‌کنیم. انتخاب از n آیتم متمایز که در گروه‌های k در هر بار اخذشده است تعداد جایگشت‌ها یا فضای محصول ذیل را برای این آزمایش نتیجه می‌دهد:

$$p(n,k) = (n)(n-۱)(n-۲)\dots(n-k-۱) = \{n!\}/\{n-k\}! \qquad (۸-۵)$$

این جایگشت تعداد گروه‌های متمایز ممکن از k آیتم را در زمان انتخاب از مجموعه‌ای از n نتیجه می‌دهد. ما می‌توانیم ببینیم که این کلی‌ترین حالت بیان قبلی است و معادله (۵-۷) را در زمانی که $k = n$ باشد کاهش می‌دهد (این، بر حسب تعریف،). به‌عنوان‌مثال، اگر ما مایل بودیم که ببینیم به چند طریق می‌توانیم سه سرور کامپیوتری را بر روی یک میز کار از پنج سرور متمایز ترتیب دهیم، می‌توانستیم این‌گونه فرض کنیم که ترتیب سرورها معنایی دارد، از این‌رو، داریم:

$$p(۵,۳) = (۵)(۵-۱)(۵-۲) = ۶۰ \text{ طریق} \qquad (۹-۵)$$

یک ترکیب یک جایگشت است در زمانی که ترتیب نادیده گرفته شده باشد. یک حالت خاص وقتی رخ می‌دهد که قرار باشد تنها دو نوع از آیتم‌ها انتخاب شوند. این‌ها ضرایب دوجمله‌ای یا $C(n,k)$ نامیده می‌شوند. تعداد ترکیبات n آیتم لحاظ شده k در یک زمان (که با $C(n,k)$ نشان داده می‌شود) معادل با $P(n,k)$ است که با تعداد کل k گروه عنصر کاهش می‌یابد که عناصری یکسان ولی در مرتبه‌های مختلف دارند (مثلاً $P(k,k)$). این به‌صورت شهودی درست است، زیرا ترتیب برای یک ترکیب بی‌اهمیت است و از این‌رو، ترکیبات منحصر به کمتری نسبت به جایگشت‌ها برای هر مجموعه مشخص از k آیتم وجود خواهد داشت. $P(k,k)$ در معادله (۵،۷) ارائه شده است، از این‌رو:

$$C(n,k) = \{n!\}/k!\{n-k\}! \qquad (۱۰-۵)$$

این، می‌توان بیان نمود که هر مجموعه از k عنصر می‌توانند $P(k,k) = k!$ جایگشت را شکل دهند، که وقتی ضرب در $C(n,k)$ شود $P(n,k)$ را نتیجه می‌دهد. تقسیم بر $P(k,k)$ نتیجه می‌دهد:

$$P(k,k)C(n,k) = P(n,k) \qquad (۱۱-۵)$$

$$C(n,k) = P(n,k)/P(k,K) = n!/(n-k)!\,k! \qquad (۱۲-۵)$$

اکنون‌که ما برخی از شیوه‌هایی که می‌توانیم فضاهای نمونه را بسازیم را مشخص نموده‌ایم، می‌توانیم نحوه اعمال احتمالات را برای رویدادها در فضای نمونه تعیین کنیم. گام نخست برای دستیابی به این هدف تخصیص مجموعه‌ای از وزن‌ها به رویدادها در فضای نمونه است. انتخاب اینکه چه فاکتور وزندهی ای باید به چه رویدادی در فضای نمونه اعمال شود کار آسانی نیست. یک روش به‌کارگیری مشاهدات در یک دوره به‌اندازه طولانی است تا تعداد زیادی از تمام نتایج ممکن حاصل شود. این چیزی است که اصطلاحاً "سیکل مشاهده، استنتاج و پیش‌بینی" نامیده می‌شود، و این برای توسعه وزن‌ها برای فرایندهایی مفید است که در آن‌ها یک مدل اساسی از فرایند یا وجود ندارد و یا پیچیده‌تر از آن است که وزن‌های رویداد را نتیجه دهد. این روش، که گاهی "تعریف احتمال کلاسیک" نامیده می‌شود، احتمال هر رویداد را به‌صورت ذیل تعریف می‌کند:

$$P(A) = N_A/N \qquad (۱۳-۵)$$

که در آن $P(A)$ نشانگر احتمال رویداد A است، N_A تعداد کل مشاهدات در جایی است که رخداد A اتفاق افتاده است، و N تعداد کل مشاهدات انجام‌شده است. بسطی به تعریف کلاسیک، که "تعریف تناوب نسبی" نامیده می‌شود، از این رابطه به دست می‌آید:

$$P(A) = \lim_{n\to\infty} N_A/N \approx N_A/N \qquad (۱۴-۵)$$

دو رویکرد قبلی، یعنی ترکیب‌ها/جایگشت‌ها و تناوب نسبی، اغلب در عمل به‌عنوان یک ابزار ارائه یک فرضیه درباره نحوه یک رفتار یک فرایند استفاده‌شده‌اند. این روش‌ها در واقع به این خاطر فرضیه‌ها را تعریف می‌کنند که هر دو مبتنی بر مشاهده تعداد محدودی از مشاهدات هستند. این واقعیت برانگیزاننده تمایل به توسعه تعاریف بدیهی برای قواعد اساسی احتمال است.

از این‌رو، تئوری احتمال، مبتنی بر مجموعه‌ای از سه اصل است. بر حسب تعریف، احتمال یک رویداد با یک عدد مثبت به دست می‌آید. بدین معنی که:

$$P(A) \geq ۰ \qquad (۱۵-۵)$$

رابطه ذیل نیز تعریف می‌شود:

$$P(S) = ۱ \qquad (۱۶-۵)$$

یعنی، مجموع تمام احتمال‌های رویدادها در فضای نمونه کلی S برابر با ۱ است. این گاهی "رویداد مشخص" نامیده می‌شود. دو تعریف قبلی دو اصل اول را نمایندگی می‌کنند و لزوماً احتمال اینکه هر رویداد به ترتیب بین ۰ و ۱ باشد را محدود می‌نمایند. سومی مبتنی بر ویژگی انحصار متقابل است، که بیان می‌کند که دو رویداد متقابلاً انحصاری هستند اگر، و تنها اگر، که رخداد یکی از این رویدادها به شکلی مثبت وقوع دیگری را نادیده بگیرد. در تنظیم واژگان، این بیان می‌کند که تلاقی این دو رویداد دربرگیرنده هیچ عنصری نمی‌شود، بدین معنی که این مجموعه تهی[۶۹] است. به‌عنوان‌مثال، دو رویداد در آزمایش شیر یا خط متقابلاً منحصر هستند. سپس، اصل سوم بیان می‌کند که احتمال ترکیبی وقوع رویدادهای A یا B برابر با مجموع هر یک از احتمال‌های آن‌ها است. بدین معنی که:

$$P(A\ or\ B) = P(A) + P(B) \qquad (۱۷-۵)$$

از این‌رو، به‌عنوان‌مثال، احتمال شیر آمدن یا آمدن عدد ۶ برابر با احتمال آمدن یک شیر یا ۱/۲ است، به علاوه احتمال آمدن یک ۶، یا ۱/۶، که ۴/۶ است.

به خواننده هشدار داده می‌شود که عبارت موجود در سمت چپ علامت مساوی احتمال رویداد A یا رویداد B را می‌خواند، در حالی که عبارت سمت راست احتمال رویداد A به‌علاوه احتمال رویداد B را می‌خواند. این یک رابطه مهم بین تئوری مجموعه و عبارت متقابل احتمالات است.

سه اصل احتمال از قرار ذیل هستند:

$$۱ - \ P(E) \geq ۰ \qquad (۱۸-۵)$$

$$۲ - \ P(S) = ۱ \qquad (۱۹-۵)$$

---

[۶۹] null set

$$۳-\ If\ AB = ۰, then\ P(A+B) = P(A) + P(B) \qquad (۲۰-۵)$$

در معادله (۵-۲۰) واژگان AB به‌صورت مجموعه A در تلاقی با مجموعه B به دست می‌آید. فضای نمونه بر روی مجموعه کلی $\{A_۱ \dots A_k\}$ تعریف می‌شود:

$$S = A_۱ + A_۲ + \cdots + A_k \qquad (۲۱-۵)$$

یک مبحث بسیار مهم در تئوری احتمال به احتمال شرطی مربوط می‌شود. آزمایش ذیل را در نظر بگیرید که در آن ما رویدادهای A، B و AB را داریم. به‌عنوان‌مثال، یک خرابی دیسک و اشکال در حافظه می‌توانند به ترتیب رویداد Aو B باشند، و یک خرابی دیسک و حافظه به طور همزمان رویداد AB است. رویداد AB دربرگیرنده رویدادهایی است که در A و B هستند. اجازه دهید بگوییم که این رویداد (AB) N_{AB} رخ می‌دهد. فرض کنید N_B نشانگر تعداد دفعات وقوع رویداد B به تنهایی باشد. اگر ما بخواهیم تناوب نسبی رویداد A را با فرض وقوع رویداد B بدانیم، می‌توانیم آزمایش و محاسبه ذیل را انجام دهیم:

$$\text{تناوب نسبی } (A) = N_{AB}/N_B \qquad (۲۲-۵)$$

بدین معنی که اگر هر دو رویداد A و B رخ دهند (رویداد AB)، تعداد دفعات وقوع رویداد A در زمانی که B نیز رخ دهد به‌صورت کسری از فضای رویداد B یافته می‌شود که در آن رویداد A تلاقی دارد (شکل ۵-۱ را ببینید). نمادگذاری برای این تناوب نسبی به‌صورت $P(A|B)$ است و به‌صورت احتمال شرطی رویداد A با داشتن این وقوع رویداد B خوانده می‌شود. اگر ما عبارت ذیل را از معادله (۴،۱۶) شکل دهیم:

$$P(A|B) = (N_{AB}/N)/(N_B/N) = N_{AB}/N_B \qquad (۲۳-۵)$$

و با اعمال معادله (۵-۱۴)، ما تعریف احتمال شرطی سنتی را به دست می‌آوریم:

$$P(A|B) = P(AB)/P(B) \qquad (۲۴-۵)$$

یک ساده‌سازی تلاقی معادله (۵-۲۳) وقتی رخ می‌دهد که رویداد A در رویداد B موجود باشد، یا زیرمجموعه‌ای از آن باشد، به‌طوری‌که $AB = A$ باشد. در این حالت، معادله (۵-۲۳)، به این صورت تبدیل می‌شود:

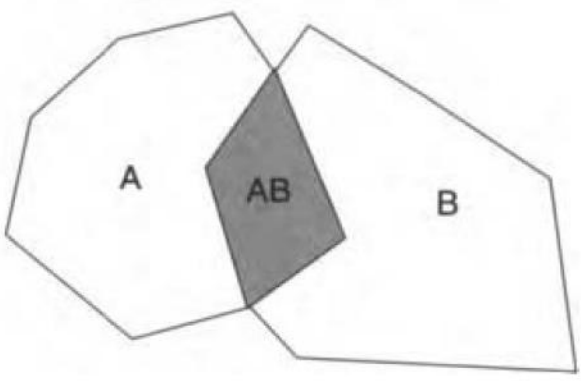
$$If\ AB = A, then\ P(A|B) = P(A)/P(B) \qquad (۲۵-۵)$$

دو رویداد، A و B، در صورتی مستقل هستند که دیاگرام Venn آن‌ها تلاقی نداشته باشد. استقلال دو رویداد با فرمول ذیل تعریف می‌شود:

$$P(AB) = P(A) \times P(B) \qquad (۲۶-۵)$$

یا

$$P(A) = P(AB)\ /P(B) = P(A|B) \qquad (۲۷-۵)$$



شکل (۵-۱): دیاگرام فضای احتمال شرطی Venn

این تعریف بیان می کند که تعداد نسبی وقوع رویداد A برابر با تعداد نسبی وقوع‌های رویداد A با فرض رخداد رویداد B است. به عبارتی ساده‌تر، استقلال دو یا چند رویداد نشانگر آن است که وقوع یک رویداد امکان استنتاج چیزی درباره وقوع دیگری را فراهم نمی‌آورد.

قضیه بیز به‌صورت ذیل بیان می‌شود: اگر ما چند رویداد متقابلاً انحصاری داشته باشیم، $B_۱, B_۲, \dots, B_N$، که اجتماع آن‌ها فضای رویداد را تعریف کند (یا، به طور رسمی‌تر، زیرمجموعه‌ای از فضای نمونه)، را برای یک آزمایش، و یک رویداد اختیاری A از فضای نمونه تعریف کند، احتمال شرطی هر رویداد B_K در مجموعه $B_۱, B_۲, \dots, B_N$ با توجه به اینکه رویداد A رخ دهد، از این رابطه به دست می‌آید:

$$P(B_k|A) = (P(B_k) \times P(A|B_K)/\sum_{i=۱}^{N} P(B_i) \times P(A|B_i) \qquad (۲۸-۵)$$

این نتیجه یک عبارت مهم است، زیرا احتمال شرطی هر رویداد یک زیرفضای رویدادهای مرتبط با یک رویداد دلخواه فضای نمونه را به احتمال شرطی رویداد دلخواه به نسبت تمام رویدادهای دیگر در زیر فضا ارتباط می‌دهد. این قضیه حاصل قضیه احتمال کلی است، که بیان می‌کند که:

$$P(A) = P(A|B_۱)P(B_۱) + \cdots + P(A|B_k)P(B_k) \qquad (۲۹-۵)$$

این قضیه به این خاطر برقرار است که رویدادهای $B_۱ \dots B_k$ متقابلاً انحصاری هستند، و از این‌رو، رویداد $A = AS = A(B_۱, B_۲, \dots, B_k) = AB_۱ + AB_۲ + \cdots + AB_k \dots$ است به‌طوری‌که:

$$P(A) = P(AB_۱) + P(AB_۲) + \cdots + P(AB_k) \qquad (۳۰-۵)$$

از آنجا که رویدادهای $B_۱, B_۲, \dots, B_k$ متقابلاً انحصاری هستند، رویدادهای $AB_۱, AB_۲, \dots, AB_k$ نیز به همین صورت هستند. اعمال تعریف احتمال شرطی معادله (۵-۲۴) به معادله (۵-۳۰) معادله (۵-۲۸) را نتیجه می‌دهد.

یک رابطه دیگر که اغلب در زمان بررسی احتمال شرطی دو رویداد مفید واقع می‌شود از قرار ذیل است:

$$P(A + B) = P(A) + P(B) - P(AB) \qquad (۳۱-۵)$$

این فرمول بیان می‌کند که برای هر دو رویداد، یعنی A و B، اگر ما مایل به تعیین وقوع رویداد A یا B یا وقوع هر دو باشیم، لازم خواهد بود تا آن‌ها را به طور مجزا و در اتحاد بررسی کنیم. این از بحث‌ها درباره مجموعه‌های رویدادهایی تبعیت می‌کند که قبلاً در این فصل به انجام رسیدند. از آنجا که اجتماع رویدادهای A و B تمام نقاط نمونه در A و B را در صورتی که باهم در نظر گرفته شوند نتیجه می‌دهد، مجموع رویدادها به طور مجزا کمیت یکسانی را به‌علاوه یک عنصر اضافی برای هر عنصر در تقاطع دو رویداد نتیجه می‌دهد. از این‌رو، ما باید این تقاطع را از برابری کسر کنیم، و بدین طریق معادله (۵-۳۰) حاصل می‌شود. توجه داشته باشید که معادله (۵-۳۱) اساسی یک ویرایش بسط یافته از معادله (۵-۲۰) است، که در آن AB برابر با مجموعه تهی نیست.

۵-۱ متغیرهای تصادفی

تاکنون، ما درباره آزمایش‌ها، در کنار فضای رویداد مرتبطشان، در زمینه احتمالات وقوع رویدادها، بحث نموده‌ایم. ما اکنون به سراغ مبحثی با اهمیت بالا خواهیم رفت، که به معیارهای احتمال اساسی را به کمیت‌های واقعی ارتباط می‌دهد. مفهوم متغیر تصادفی احتمالات نتایج یک آزمایش را به یک محدوده یا مجموعه از اعداد ارتباط می‌دهد. از این‌رو یک متغیر تصادفی به‌صورت تابعی تعریف می‌شود که مقادیر ورودی آن رویدادهای فضای نمونه هستند و نتیجه آن یک عدد حقیقی است. به‌عنوان‌مثال، ما می‌توانیم آزمایشی داشته باشیم که در آن نتیجه طول هر پیامی باشد که بر روی یک خط ارتباطی دریافت می‌شود. یک متغیر تصادفی تعریف‌شده بر روی این آزمایش می‌تواند تعداد پیام‌هایی باشد که برابر با یک تعداد مشخص از کاراکترها هستند. اغلب، ما می‌خواهیم که طیفی از مقادیر متغیرهای تصادفی را در نظر بگیریم ، به‌عنوان‌مثال، طیف پیام‌های بزرگ‌تر از x_1 . این به‌صورت $\{X \leq x_1\}$ نمایش داده می‌شود، که در آن X نشانگر متغیر تصادفی است و x_1 یک مقدار برای متغیر تصادفی در یک نقطه مشخص است. ما می‌توانیم این را در جایی که متغیر تصادفی X یک مقدار بیشتر از x_1 را نتیجه دهد یک رویداد بنامیم.

در ادامه مثال قبلی، فرض کنید که ما نتایج ذیل را از آزمایش طول پیام داریم: متغیر تصادفی توسط تعداد دفعاتی که یک طول پیام مشخص مشاهده می‌شود تعریف می‌شود. در ارجاع به شکل (۵-۲) می‌توان گفت که رویداد $\{X > ۲۰۰۰\}$ دربرگیرنده نتایج پیام‌های ۱، ۴، ۵ و ۶ است.

نتیجه	طول پیام
1	2,097
2	500
3	1,259
4	5,794
5	4,258
6	5,205

شکل (۵-۲): نتایج آزمایش طول پیام

متغیرهای تصادفی ممکن است گسسته یا پیوسته باشند. یک متغیر تصادفی گسسته متغیری است که در آزمایشی تعریف می‌شود که در آن تعداد رویدادها در مجموعه نتایج محدود یا نامحدود است (یعنی می‌توان یک عدد صحیح مثبت را به هر رویداد تخصیص داد، حتی در صورتی که تعدادی نامحدود از نتایج وجود داشته باشند). یک متغیر تصادفی پیوسته متغیری است که در آزمایشی تعریف می‌شود که در آن تعداد نتایج ممکن نامحدود باشد (یعنی بر روی خط واقعی تعریف شود). مفهوم متغیرهای تصادفی مبنا را برای بحث درباره توزیع‌های احتمال و توابع تراکم شکل می‌دهد.

۵-۲ متغیرهای تصادفی با توزیع مشترک

فرض کنید ما یک آزمایش داریم که دارای دو یا چند متغییر تصادفی در فضای رویداد است و ما مایل به ایجاد یک متغیر تصادفی هستیم که در هر بار، هر یک از متغیرهای تصادفی فرد را در نظر می گیرد. اینها متغیرهای تصادفی توزیع شده مشترک نامیده می شوند و آنها تقاطع های فضاهای رویداد متغیر تصادفی فرد را نشان می دهند. متغیرهای تصادفی توزیع شده مشترک همراه با نماد زیر نمایش داده می شوند:

$$\{X \le x_1, Y \le y_1\} \qquad (۳۲-۵)$$

به سادگی بیان شده است که متغیرهای تصادفی مشترک، خروجی خود را از یک تابع که دامنه مجموعه نتایج برای تمام دامنه های متغیر تصادفی فرد است، محاسبه می کنند. در اینجا باید اشاره کرد که ترکیبیات و شرایط پیچیده تر برای تابع متغیر تصادفی ممکن است ساخته شود. به عنوان مثال، متغیرهای تصادفی زیر را در نظر بگیرید:

$$\{x_1 \le X \le x_2\}\, where\ x_1 < x_2 \qquad (۳۳-۵)$$

$$\{x_1 > X, x_2 < X\}\, where\ x_1 > x_2 \qquad (۳۴-۵)$$

$$\{x_1 \le X \le x_2, y_1 \le Y \le y_2\}\, where\ x_1 < x_2\ and\ y_1 < y_2 \qquad (۳۵-۵)$$

۵-۳ توزیع‌های احتمال

مفهوم یک متغیر تصادفی به‌خودی‌خود استفاده عملی گسترده‌ای ندارد. برای یافتن راهکاری برای این، ما یک تابع توزیع را برای هر متغیر تصادفی X تعریف می‌کنیم. تابع توزیع معمولاً به این صورت نمایش داده می‌شود:

$$F(x) = P(X \leq x) \qquad (۳۶-۵)$$

بر حسب تعریف، تابع توزیع مقادیری از ۰ تا ۱ را به خود می‌گیرد. این، تابع توزیع با افزایش مقدار x غیر کاهشی است. این ویژگی‌ها به‌صورت ذیل به طور خلاصه بیان می‌شوند:

خصوصیت I: $\lim_{x\to\infty} F(x) = ۰$ (۳۷-۵)

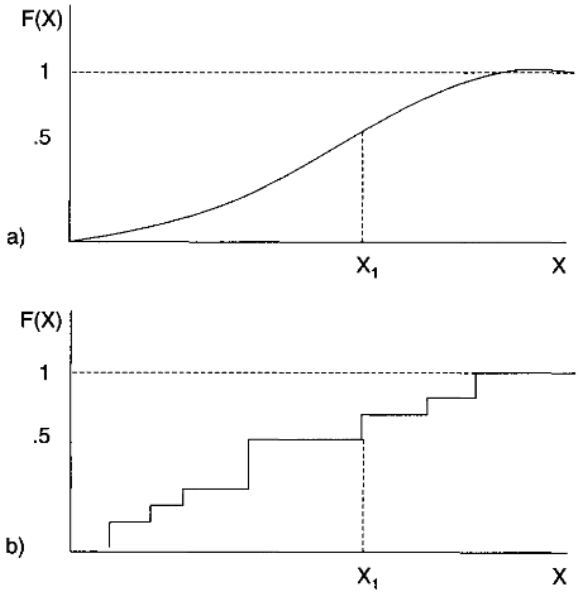
خصوصیت Π: $\lim_{x\to\infty} F(x) = ۱$ (۳۸-۵)

خصوصیت ΙΙΙ: $\lim_{x\to\infty} F(x_۱) \leq F(x_۲)\ if\ x_۱ \leq x_۲$ (۳۹-۵)

توابع توزیع توابع توزیع تجمعی نامیده می‌شوند، زیرا در هر x در امتداد توزیع، ناحیه زیر منحنی در سمت چپ x جمع تجمیعی احتمالات متغیرهای تصادفی $\{x \leq X\}$ را نمایندگی می‌کند. شکل ۵،۳ چند تا از توابع توزیع نمونه را نشان می‌دهد.

از شکل ۵،۳، روشن است که این توابع به این خاطر توابع توزیع نامیده می‌شوند که دقیقاً نشان می‌دهند که چگونه احتمال متغیر تصادفی بر روی طیف مقادیر متغیر تصادفی توزیع می‌شود.

تابع توزیع نشان داده‌شده در شکل (۵-۳) (a) یک تابع پیوسته است، زیرا مبتنی بر یک متغیر تصادفی پیوسته است. شکل (۵-۳) (b) یک تابع توزیع گسسته را نشان می‌دهد که مبتنی بر متغیر تصادفی گسسته است.



شکل (۵-۳): توابع توزیع نمونه

یک توزیع مشترک توزیعی است که نحوه ارتباط احتمال با هر یک از چند متغیر تصادفی را تعریف می‌کند. از این‌رو، ما می‌توانیم تابعی را بیان کنیم که یک توزیع مشترک را به این صورت بیان کند:

$$F(x,y) = P(X \leq x, Y \leq y) \qquad (۴۰-۵)$$

این تابع را می‌توان به این صورت تفسیر نمود:

$$F(x,y) = P(X \leq x \; intersected \; with \; Y \leq y) \qquad (۴۱-۵)$$

ما با داشتن توزیع مشترک معادله (۵،۴۰) به توابع توزیع منفرد X و Y نیز علاقه‌مند هستیم. به‌عنوان‌مثال، توزیع X با داشتن $F(x,y)$، که تابع توزیع حاشیه‌ای X متناظر با $F(x,y)$ نیز تعریف می‌شود، به این صورت به دست می‌آید:

$$F_x(x) = \lim_{y \to \infty} F_{x,y}(x,y) \qquad (۴۲-۵)$$

یا

$$F_X(x) = F_{X,Y}(x, \infty) \qquad (۴۳-۵)$$

همین برای توزیع حاشیه Y صدق می‌کند:

$$F_Y(y) = F_{X,Y}(\infty, y) \qquad (۴۴-۵)$$

توزیع‌های حاشیه‌ای متغیرهای تصادفی قبلاً از تعاریف متغیرهای تصادفی و توابع توزیع نتیجه گرفته‌اند. به خاطر داشته باشید که یک متغیر تصادفی با طیفی از مقادیر در امتداد محور واقعی تعریف می‌شود و اینکه تابع توزیع به‌صورت احتمال تجمیعی تعریف می‌شود که متغیر تصادفی حداقل به یک مقدار مشخص دست خواهد یافت. احتمال اینکه متغیر تصادفی مقداری کمتر از بینهایت کسب نماید برابر با یک است. از این‌رو، توزیع حاشیه‌ای برای یک متغیر تصادفی با داشتن یک توزیع مشترک با داشتن این رابطه روشن است:

$$(X \leq x) = (X \leq x, Y \leq \infty) \qquad (۴۵-۵)$$

۵-۴ تراکم‌ها

یک تابع تراکم مشتق تابع توزیع را تعریف می‌کند، که نشانگر نرخ تغییر توزیع احتمال است:

$$f(x) = dF(x)/dx \qquad (۴۶-۵)$$

این تعریف برای متغیرهای تصادفی پیوسته برقرار است. برای متغیرهای تصادفی گسسته، تابع تراکم به‌صورت احتمال‌های گسسته‌ای تعریف می‌شود که متغیر تصادفی برای محدوده مقادیر احتمالی‌اش برابر با یک مقدار مشخص است. بدین معنی که:

$$f_X(x) = P[X = x] = \sum P(X)\delta(x - X) \qquad (۴۷-۵)$$

که در آن $\delta(x - X)$ یک تابع دلتا است که وقتی $x = X$ باشد ۱ است و در غیر این صورت ۰ است.

از روابط قبلی، ما می‌توانیم ببینیم که چگونه تابع توزیع شکل می‌گیرد. برای هر مقدار متغیر تصادفی، ما می‌توانیم (برای یک تابع پیوسته) تا آن نقطه انتگرال‌گیری کنیم تا احتمال تجمیعی تا آن نقطه را بیابیم. این احتمال‌ها برای توابع گسسته جمع زده می‌شوند:

$$F(x) = \int_{-\infty}^{x} f(t)dt \qquad (۴۸-۵)$$

$$F(x) = \sum_{n\to\infty}^{x} f(n) \qquad (۴۹-۵)$$

ما از بحث قبلی می‌دانیم که $F(\infty) = ۱$ است، به‌طوری‌که:

$$\int_{-\infty}^{x} f(t)dt = ۱ \qquad (۵۰-۵)$$

و

$$\sum_{t=-\infty}^{\infty} f(t) = ۱ \qquad (۵۱-۵)$$

به شکلی مشابه با آنچه قبلاً برای یافتن تابع توزیع از تابع تراکم برای یک متغیر تصادفی واحد نشان داده شد، ما می‌توانیم توزیع مشترک را از تراکم مشترک بیابیم. این رابطه به این صورت به دست می‌آید:

$$F(x,y) = \int_{-\infty}^{x}\int_{-\infty}^{y} f(t,u)\,dt\,du \qquad (۵۲-۵)$$

به طور مشابه، یک توزیع گسسته را می‌توان از این تراکم گسسته یافت:

$$F(x,y) = \sum_{i=-\infty}^{x}\sum_{j=-\infty}^{y} f(i,j) \qquad (۵۳-۵)$$

همانند تراکم‌های توزیع‌شده تکین، مساحت کل زیر تابع تراکم از این رابطه به دست می‌آید:

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\,dx\,dy = ۱ \qquad (۵۴-۵)$$

به دست آوردن تابع تراکم از تابع توزیع برای یک حالت پیوسته از این رابطه به دست می‌آید:

$$f(x,y) = \partial^{۲} F(x,y) \,/\, \partial x\,\partial y \qquad (۵۵-۵)$$

ما تراکم حاشیه‌ای یک متغیر تصادفی توزیع‌شده مشترک را به این صورت تعریف می‌کنیم:

$$f_Y(y) = \int_{-\infty}^{\infty} f(x,y)\,dx \qquad (۵۶-۵)$$

ویژگی استقلال بر روی توزیع‌های مشترک به این صورت تعریف می‌شود:

$$F(x,y) = F_x(x)\,F_Y(y) \qquad (۵۷-۵)$$

و برای تراکم‌های مشترک به این صورت:

$$f(x,y) = f_x(x)\,f_Y(y) \qquad (۵۸-۵)$$

در برخی از موارد، تعریف توزیع‌های مشترک ترکیبی که در آن‌ها یکی از متغیرها گسسته و دیگری پیوسته باشد ضروری است. تراکم مشترک، که در آن y نشانگر متغیر پیوسته است و i تراکم گسسته را نمایندگی می‌کند، به این صورت نوشته می‌شود:

$$f(i,y) = f_{x|Y}(y|i)\, P_x(i) \qquad (۵۹-۵)$$

این عبارت یک نکته مهم دیگر را معرفی می‌کند: توزیع‌های شرطی. برای متغیرهای تصادفی گسسته، تابع شرطی را می‌توان به‌صورت ذیل تعریف نمود:

$$f_{X|Y}(x|y) = f(x,y) / f_Y(y) \qquad (۶۰-۵)$$

به طور مشابه، ما می‌توانیم تراکم شرطی y را با داشتن x از معادله (۵-۶۰) تعریف کنیم. نتایج ذیل:

$$f(x,y) = f_{X|Y}(x|y)\ f_Y(y) = f_{x|y}(x|y)\ f_X(x) \qquad (۶۱-۵)$$

این یک راه آسان برای ارتباط دادن تراکم‌های شرطی برای دو متغیر تصادفی است. اگر متغیرهای X و Y تصادفی مستقل باشند، معادله (۵-۶۰) به این صورت تبدیل می‌شود:

$$f(x,y) = f_X(x)\, f_Y(y) \qquad (۶۲-۵)$$

و رابطه ذیل حاصل می‌شود:

$$f_{Y|X}(y|x) = f_y(y) \qquad (۶۳-۵)$$

از معادلات (۵-۵۶) و(۵-۶۰)، ما می‌توانیم با جایگزینی به این رابطه برسیم:

$$f_Y(y) = \int_{-\infty}^{\infty} f_X(x)\, f_{Y|X}(y|x)\ dx \qquad (۶۴-۵)$$

و این (برای تراکم حاشیه‌ای X):

$$f_Y(x) = \int_{-\infty}^{\infty} f_Y(x)\, f_{X|Y}(x,y)\ dy \qquad (۶۵-۵)$$

با ترکیب معادلات (۵-۶۰)، (۵-۶۱) و (۵-۶۴)، ما قاعده بیز را برای متغیرهای تصادفی پیوسته به دست می‌آوریم:

$$F_{X|Y}(x|y) = \frac{f_X(x)\, f_{Y|X}(y|x)}{\int_{-\infty}^{\infty} f_X(x)\, f_{Y|X}(y|x)\ dx} \qquad (۵-۶۶)$$

این تکمیلی بر بحث ما درباره ویژگی‌های توزیع‌ها و تراکم‌های توزیع است. در بخش بعدی، ما به بررسی برخی از روش‌ها برای دستیابی به آمارهای پرکاربرد درباره متغیرهای تصادفی با استفاده از توزیع‌ها و تراکم‌هایشان خواهیم پرداخت.

۵-۵ امید ریاضی

اگرچه هر دو تابع توزیع و چگالی یک متغیر تصادفی تمام اطلاعات ضروری برای توصیف رفتارش را فراهم می‌آورند، ولی ما اغلب مایلیم تا یک کمیت واحد (یا تعداد اندکی از آن‌ها) را داشته باشیم که اطلاعات خلاصه‌ای از متغیر تصادفی در اختیار قرار دهد. چنان معیاری امید ریاضی یک متغیر تصادفی است. امید ریاضی میانگین نیز نامیده می‌شود. امید ریاضی یا ارزش مورد انتظار برای یک متغیر تصادفی گسسته X به این صورت تعریف می‌شود:

$$E[X] = \sum_x xP(x) \qquad (۶۷-۵)$$

و برای یک متغیر تصادفی پیوسته X با تابع تراکم $f(x)$ به این صورت است:

$$E[X] = \int_{-\infty}^{\infty} x\ f(x)\,dx \qquad (۶۸-۵)$$

اکنون فرض کنید که ما یک تابع از یک متغیر تصادفی X داشته باشیم، مثلاً $g(x)$، امید ریاضی از این رابطه به دست می‌آید:

$$E[g(X)] = \int_{-\infty}^{\infty} g(x)\, f(x)\,dx \qquad (۶۹-۵)$$

برای متغیرهای تصادفی پیوسته، و به این صورت:

$$E[g(X)] = \sum_x g(x)\, P(x) \qquad (۷۰-۵)$$

برای متغیرهای تصادفی گسسته است.

اگر ما متغیرهای تصادفی دارای توزیع مشترک داشته باشیم، امید ریاضی برای متغیرهای تصادفی گسسته به این صورت تعریف می‌شود:

$$E[g(X,Y)] = \sum_x \sum_y g(x,y)\, f(x,y) \qquad (۷۱-۵)$$

و برای متغیرهای تصادفی پیوسته به این صورت:

$$E\left[g\left(X,Y\right)\right) \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g\left(x,y\right) f(x,y)\, dx\, dy \qquad (۷۲-۵)$$

برای تابع $g\left(X,Y\right)$ است. این فرمولاسیون‌ها برای مقادیر مورد انتظار (امید ریاضی) در صورتی معتبرند که سمت راست معادلات مربوطه کمتر از بینهایت باشند.

چند قانون مفید در رابطه با امید ریاضی وجود دارد که اکنون به بحث درباره آن خواهیم پرداخت. فرض کنید که می‌خواهیم رابطه ذیل را بیابیم:

$$E\left[aX+b\right] = \int_{-\infty}^{\infty}(ax+b)\, f(x)\, dx \qquad (۷۳-۵)$$

امید ریاضی در سمت راست به این صورت تبدیل می‌شود:

$$a\int_{-\infty}^{\infty} x\, f\left(x\right) dx + b\int_{-\infty}^{\infty} f\left(x\right) dx \qquad (۷۴-۵)$$

از معادلات (۵۱-۵) و (۶۹-۵)، معادله (۷۴-۵) به این صورت تبدیل می‌شود:

$$E\left[aX+b\right] = a\, E\left[X\right] + b \qquad (۷۵-۵)$$

تنظیم a یا b در صفر منجر به رابطه ذیل می‌شود:

$$E\left[aX\right] = aE\left[X\right] \qquad (۷۶-۵)$$

$$E\left[b\right] = b \qquad (۷۷-۵)$$

اکنون فرض کنید که داشته باشیم:

$$E\left[g\left(X\right)+h\left(x\right)\right] = \int_{-\infty}^{\infty}(g\left(x\right)+h\left(x\right))\, f\left(x\right) dx \qquad (۷۸-۵)$$

این انتگرال به این صورت تبدیل می‌شود:

$$\int_{-\infty}^{\infty} g\left(x\right) f(x)\, dx + \int_{-\infty}^{\infty} h\left(x\right) f(x)\, dx \qquad (۷۹-۵)$$

از معادله (۶۸-۵)، داریم:

$$E\left[g\left(X\right)+h\left(X\right)\right] = E\left[g\left(X\right)\right] + E\left[h\left(X\right)\right] \qquad (۸۰-۵)$$

به طور مشابه، برای توابع دو متغیر تصادفی، داریم:

$$E\left[g\left(X,Y\right)+h\left(X,Y\right)\right] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(g\left(x,y\right)+h\left(x,y\right))\, f\left(x,y\right) dy\, dx \qquad (۸۱-۵)$$

که به این صورت تبدیل می‌شود:

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g\left(x,y\right) f(x,y)\, dy\, dx + \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h\left(x,y\right) f(x,y)\, dy\, dx \qquad (۸۲-۵)$$

از معادله (۷۲-۵)، به دست می‌آوریم:

$$E\,[g\,(X,Y)+h\,(X,Y)]=E\,[g\,(X,Y)]+E\,[h\,(X,Y)] \qquad (۸۳-۵)$$

به طور مشابه،

$$E\,[X+Y]=E\,[X]+E\,[Y] \qquad (۸۴-۵)$$

وضعیت دو متغیر تصادفی مستقل، X و Y، را با معادله (۵-۷۲) در نظر بگیرید:

$$E\,[XY]=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} xy\,(x,y)\,dy\,dx \qquad (۸۵-۵)$$

که، به خاطر معادله (۵-۵۸)، به این صورت تبدیل می‌شود:

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} xy\,f_X(x)\,f_Y(y)dy\,dx \qquad (۸۶-۵)$$

جداسازی انتگرال‌ها توسط انتگرال ده‌ها نتیجه می‌دهد:

$$\int_{-\infty}^{\infty} x\,f_X\,(x)\,dx\;\;\int_{-\infty}^{\infty} y\,f_Y\,(y)\,dy \qquad (۸۷-۵)$$

از معادلات (۵-۶۹) و (۵-۸۵)، داریم:

$$E\,[XY]=E\,[X]\,E\,[Y] \qquad (۸۸-۵)$$

برای متغیرهای تصادفی مستقل X و Y، برای یک تابع ویژه یک متغیر تصادفی، یعنی $g(X)=x^n$، امید ریاضی $g(X)$ به‌صورت "nمین لحظه" متغیر تصادفی X شناخته می‌شود.لحظه اول $g(X)$ به‌صورت متوسط متغیر تصادفی X برای $g(X)=X$ شناخته می‌شود. گشتاور، به صورتی که قبلاً تعریف‌شده، در مبدأ متمرکز است و از این‌رو " گشتاور نسبت به مبدأ [۷۰]" نامیده می‌شود. یک تعریف رایج‌تر و مناسب‌تر از گشتاور دربرگیرنده انتقال تابع چگالی است به‌طوری‌که میانگین در مبدأ متمرکز است. گشتاور که در این راستا تعریف‌شده‌اند " گشتاور مرکزی " نامیده می‌شوند، زیرا بر روی توابع تراکمی تعریف می‌شوند که در مبدأ متمرکز بوده‌اند. از این‌رو، تابع متغیر تصادفی به این صورت تبدیل می‌شود:

$$g\,(X)=(x-\mu)^n \qquad (۸۹-۵)$$

که در آن متوسط از این رابطه به دست می‌آید:

$$\mu=E\,[X] \qquad (۹۰-۵)$$

از این رو گشتاورمرکزی، یا گشتاور میانگین ، به این صورت تعریف می‌شود:

---

[۷۰] moments about the origin

$$\mu_n = E\left[(X-\mu)^n\right] = \sum_n (x-\mu)^n f(x) \qquad (۹۱-۵)$$

برای متغیر تصادفی گسسته X، و به‌صورت:

$$\mu_n = \int_{-\infty}^{\infty} (x-\mu)^n f(x)\, dx \qquad (۹۲-۵)$$

برای متغیر تصادفی پیوسته X است.

یک معیار مهم تغییرپذیری توزیع یک تابع حول متوسط "واریانس" نامیده می‌شود. این معیار، به بیانی باز، مقادیر توابع به نسبت متوسط چقدر متمرکز هستند. از این‌رو، یک واریانس کوچک نشانگر آن است که محدوده مقادیر تابع در نزدیکی متوسط متمرکز است، در حالی که یک واریانس بزرگ حکایت از آن دارد که مقادیر گسترده‌تر هستند. واریانس یک متغیر تصادفی توسط گشتاور مرکزی دوم آن تعریف‌شده و به این صورت نمایش داده می‌شود:

$$\sigma^2 = Var[X] = \mu^2 = E\left[(X-\mu)^2\right] = \int_{-\infty}^{\infty} (x-\mu)^2 f(x)\, dx \qquad (۹۳-۵)$$

استفاده از چند مفهوم مختلف را در نظر داشته باشید، همه مشترک هستند. برای برخی از توابع، $f(x)$، ارزیابی انتگرال معادله (۵-۹۳) ممکن است به‌سختی قابل ارزیابی باشد. خوشبختانه، ما می‌توانیم یک عبارت جایگزین را برای واریانس استخراج کنیم، به‌صورت ذیل:

$$\sigma^2 = E\left[(X-\mu)^2\right]$$
$$= \left[X^2 - 2X\mu + \mu^2\right]$$
$$= E\left[X^2\right] - 2\mu E[X] + \mu^2$$

با استفاده از معادله (۵-۹۰)

(۵-۹۴)

$$= E\left[X^2\right] - 2\mu^2 + \mu^2$$

$$\sigma^2 = E\left[X^2\right] - \mu^2$$

انحراف استاندارد یک متغیر تصادفی به‌صورت ریشه دوم واریانس تعریف می‌شود و به این صورت نشان داده می‌شود:

$$\sigma = \sqrt{\sigma^2} = \sqrt{E[X^2] - \mu^2} \qquad (۹۵-۵)$$

کوواریانس دو متغیر تصادفی معیاری از درجه وابستگی خطی است، که "همبستگی" دو متغیر نیز نامیده می‌شود. این کوواریانس به این صورت تعریف می‌شود:

$$Cov[X, Y] = E[(X - \mu_x)(Y - \mu_y)] \qquad (۹۶-۵)$$

اگر X و Y مستقل باشند، کوواریانس برابر با صفر است. این حاصل رابطه ذیل است:

$$E[(X - \mu_x)(Y - \mu_y)] = E[XY - X\mu_y - Y\mu_y + \mu_x \mu_x]$$
$$= E[XY] - \mu_y E[X] - \mu_x E[Y] + \mu_x \mu_x \qquad (۹۷-۵)$$

از معادله (۵-۹۰):

$$= E[XY] - ۲\mu_y \mu_x + \mu_x \mu_x$$
$$= E[XY] - \mu_x \mu_y \qquad \text{ادامه (۵-۹۷)}$$

با استفاده از معادله (۵-۸۸) به دست می‌آوریم:

$$Cov[XY] = E[X]\,E[Y] - \mu_y \mu_x \qquad \text{ادامه (۵-۹۷)}$$

معادله (۵-۹۷) یک ابزار راحت‌تر برای محاسبه کوواریانس می‌دهد. دو متغیر تصادفی اگر $Cov[X, Y] = ۰$ باشد همبسته نامیده می‌شوند.

چند ویژگی مفید واریانس وجود دارد، که ما اکنون درباره آن‌ها بحث خواهیم کرد، این، با داشتن فاصله‌ای از میانگین سنجیده شده در انحراف‌های استاندارد، با روشی برای توسعه یک کران پایین بر روی احتمال برای هرگونه متغیر تصادفی دنبال خواهد شد.

از معادلات (۵-۷۵، ۵-۷۶، ۵-۷۷، ۵-۹۴)، ما به راحتی می‌توانیم نشان دهیم که:

$$Var[x + b] = E[(X + b - E[X + b])^2]$$

$$= E\left[(X + b - E[X] - b)^{2}\right] \qquad (۹۸-۵)$$

با استفاده از(۵-۹۳):

$$= E\left[(X - \mu_x)^{2}\right] = Var[X] \qquad \text{ادامه } (۹۸-۵)$$

از معادلات (۵-۷۶) و (۵-۹۳):

$$\begin{aligned} Var[aX] &= E\left[(aX - E[aX])^{2}\right] \\ &= E\left[a^{2}(X - E[X])^{2}\right] \\ &= a^{2}E\left[(X - \mu_x)^{2}\right] = a^{2}Var[X] \end{aligned} \qquad (۹۹-۵)$$

برای دو متغیر تصادفی توزیع‌شده مشترک، یعنی X و Y، این واریانس به این صورت تعریف می‌شود:

$$\begin{aligned} Var[X + Y] &= E\left[(X + Y - E[X + Y])^{2}\right] \\ &= E\left[(X - \mu_x + Y - \mu_y)^{2}\right] \\ &= E\left[(X - \mu_x)^{2}\right] + E\left[(Y - \mu_y)^{2}\right] \\ &+ 2 = E\left[(X - \mu_x)(Y - \mu_y\right] \\ &= Var[X] + Var[Y] + 2\,Cov[x, y] \end{aligned} \qquad (۱۰۰-۵)$$

با داشتن هرگونه متغیر تصادفی، امکان استخراج عبارتی وجود دارد که احتمال مینیمم یک متغیر تصادفی که در انحراف‌های استاندارد k میانگینش قرار گرفته باشد وجود دارد. این قضیه با عنوان قضیه Chebyshev شناخته می‌شود و به‌صورت ذیل بیان می‌شود:

$$P(\mu - k\sigma) < X < (\mu + K\sigma)) \geq 1 - (1 / k^{2}) \qquad (۱۰۱-۵)$$

معادله (۵-۱۰۱) را می‌توان به‌صورت ذیل استخراج نمود. از معادله (۵-۹۳) داریم:

$$\sigma^{2} = \int_{-\infty}^{\infty}(x - \mu^{2})\, f(x)\, dx \qquad (۱۰۲-۵)$$

$$\sigma^2 = \int_{-\infty}^{\mu - K\sigma} (x-\mu)^2 f(x)\,dx + \int_{\mu-K\sigma}^{\mu+K\sigma} (x-\mu)^2 f(x)\,dx + \int_{\mu+K\sigma}^{\infty} (x-\mu)^2 f(x)\,dx \qquad (۱۰۳-۵)$$

از آنجا که انتگرال میانی مثبت یا صفر است، ما می‌توانیم آن را از این عبارت حذف کنیم تا به این رابطه برسیم:

$$\sigma^2 \geq \int_{-\infty}^{\mu-K\sigma} (x-\mu)^2 f(x)\,dx + \int_{\mu+K\sigma}^{\infty} (x-\mu)^2 f(x)\,dx \qquad (۱۰۴-۵)$$

در این محدوده:

$$x \geq \mu + k\sigma \quad \text{و} \quad x \leq \mu - k\sigma \qquad (۱۰۵-۵)$$

داریم:

$$|x-\mu| \geq k\sigma \qquad (۱۰۶-۵)$$

از این‌رو:

$$(x-\mu)^2 \geq (k\sigma)^2 \qquad (۱۰۷-۵)$$

از این‌رو، ما می‌توانیم با جایگزینی در معادله (۵-۱۰۴) به دست بیاوریم:

$$\sigma^{۲} \geq \int_{-\infty}^{\mu-k\sigma} f(x)\,dx + \int_{\mu+k\sigma}^{\infty} (k\,\sigma)^{۲} f(x)\,dx \qquad (۱۰۸-۵)$$

و تقسیم کنیم تا رابطه ذیل را به دست بیاوریم:

$$\frac{\sigma^{۲}}{(k\sigma)^{۲}} \geq \int_{-\infty}^{\mu-k\sigma} f(x)dx + \int_{\mu+k\sigma}^{\infty} f(x)\,dx \qquad (۱۰۹-۵)$$

با بازنویسی معادله (۱۰۷-۵) داریم:

$$\int_{\mu-k\sigma}^{\mu-k\sigma} f(x)\,dx + \int_{\mu+k\sigma}^{\infty} f(x)\,dx = \int_{-\infty}^{\infty} f(x)dx - \int_{\mu-k\sigma}^{\mu+k\sigma} f(x)\,dx \leq ۱/k^{۲} \qquad (۱۱۰-۵)$$

و از معادله (۵-۵۰) داریم:

$$۱ - \int_{\mu-k\sigma}^{\mu+k\sigma} f(x)\,dx \leq ۱/k^{۲} \qquad (۱۱۱-۵)$$

از معادله (۵-۴۸) داریم:

$$P(\mu - k\sigma) < X < (\mu + k\,\sigma) = \int_{\mu-k\sigma}^{\mu+k\,\sigma} f(x)\,dx \qquad (۵-۱۱۲)$$

از این‌رو، معادله (۱۰۱-۵) نتیجه می‌دهد:

$$P(\mu - k\sigma) < X < (\mu + K\sigma) = \int_{\mu-k\sigma}^{\mu+k\sigma} f(x)\,dx \geq ۱ - ۱/k^{۲} \qquad (۵-۱۱۳)$$

۵-۶ برخی مثال‌ها از توزیع‌های احتمال

در این بخش، ما چند توزیع احتمال پیوسته و چند مورد گسسته را بررسی خواهیم نمود که به تثبیت تئوری احتمال اساسی بخش‌های قبلی کمک خواهد نمود. بسیاری از این توزیع‌ها به‌طورمعمول برای مدل‌سازی فرایندهای واقعی و کمک به رسیدن به تخمین‌هایی برای کمیت‌های مورد نظر در سیستم‌های واقعی استفاده می‌شوند. ما درباره ویژگی‌های هر توزیع و فرایند استخراج انحراف‌های تصادفی، با داشتن یک توزیع مشخص که یک فرایند واقعی را مدل‌سازی کند، بحث خواهیم نمود.

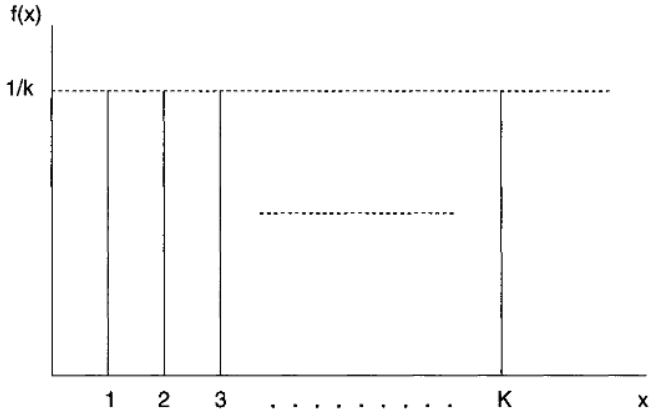
۵-٦-۱ توزیع یکنواخت

ساده‌ترین مورد از تمام توزیع‌های احتمال توزیع یکنواخت گسسته است. چنان توزیعی بیان می‌کند که تمام مقادیر متغیر تصادفی دارای احتمال برابر هستند و تنها به تعداد خروجی‌های ممکن آزمایش بستگی دارند. تابع تراکم برای توزیع یکنواخت از این رابطه به دست می‌آید:

$$f(x)\ ۱/k \qquad x = x_۱, x_۲, \ldots, x_k \qquad (۱۱۴-۵)$$

که در آن k تعداد خروجی‌های ممکن است. آزمایشی که در آن متغیر تصادفی $X = P(n)$ است، $n = ۱$ تا ۶، و این رویداد انداختن تاس است که منتج به یک توزیع احتمال یکنواخت گسسته می‌شود. طرحی از تابع تراکم برای توزیع یکنواخت در شکل (۵-۴) نشان داده‌شده است. متوسط توزیع یکنواخت با معادله (۵-۶۶) یافت می‌شود و از این رابطه به دست می‌آید:

$$E[X] = \sum_{i=۱}^{k} x_i\ (۱/k) \qquad (۱۱۵-۵)$$



شکل (۵-۴) تابع تراکم یکنواخت

انحراف استاندارد از این رابطه به دست می‌آید:

$$\sigma^2 = E\left[(X-\mu)^2\right] = \sum_{i=1}^{k}(x_i-\mu)^2 \; f(x_i)$$

$$= \sum_{i=1}^{k}\frac{(x_i-\mu)^2}{k} \qquad (۱۱۶-۵)$$

۵-۶-۲ توزیع دوجمله‌ای

مفهوم یک آزمایش برنولی در بسیاری از توزیع‌های گسسته اهمیت دارد. یک آزمایش برنولی آزمایشی است که در آن نتیجه در آن تنها می‌تواند موفقیت یا شکست باشد. متغیرهای تصادفی تعریف‌شده در آزمایش‌های برنولی متوالی چند تا از توابع تراکم گسسته‌ای را شکل می‌دهند که در موردشان بحث خواهیم نمود.

یک توزیع احتمال گسسته توزیع دوجمله‌ای است. این توزیع ناشی از آزمایش‌هایی است که در آن‌ها تنها دو نتیجه ممکن از یک آزمایش وجود دارد، همچون انداختن یک سکه. برای این توزیع، یک نتیجه برای نشان دادن موفقیت و دیگری برای شکست انتخاب‌شده است. یک

آزمایش دوجمله‌ای نیز موردنیاز است که احتمال موفقیت برای آزمایش‌های متوالی ثابت می‌ماند، که آزمایش‌ها مستقل هستند، و اینکه هر نتیجه تجربی منجر به موفقیت یا شکست می‌شود. از آنجا که آزمایش‌ها مستقل هستند، احتمال کلی برای یک آزمایش با x موفقیت و n آزمایش را می‌توان با صرفاً ضرب احتمال هر رویداد یافت (معادله (۵-۲۶) را ببینید). اگر احتمال موفقیت به‌صورت p داده شود و اگر $p, q = ۱$ باشد، داریم:

$$P\left(\text{x موفقیت در n آزمایش}\right) = p^{x}\, q^{(n-x)} \qquad (۵-۱۱۷)$$

برای توزیع دوجمله‌ای، ما می‌خواهیم که تعداد موفقیت‌ها در n آزمایش مستقل را بیابیم، چرا که ما احتمال موفقیت را برای هر آزمایش منفرد می‌دانیم. تعداد موفقیت‌ها در n آزمایش یک ترکیب است، به صورتی که در معادله (۵-۱۰) ارائه شده است. از این‌رو احتمال x موفقیت در n آزمایش این عبارت برای این توزیع دوجمله‌ای است:

$$f(x) = X(n, x)\, p^{x}\, q^{n-x} \qquad x = ۱, ۲, ۳, \ldots, n \qquad (۵-۱۱۸)$$

فرض کنید که ما یک آزمایش دوجمله‌ای داشته باشیم که در آن نتایج n آزمایش را بتوان برای نمایش متغیر تصادفی X استفاده نمود، که نشانگر موفقیت‌ها در n آزمایش است. از این‌رو، به وسیله معادله (۵-۸۴) و به وسیله تعریف متغیرهای تصادفی دوجمله‌ای:

$$E[X] = E[x_۱] + E[x_۲] + \cdots + E[x_n]$$
$$E[X] = np \qquad (۵-۱۱۹)$$

از آنجا که واریانس هر آزمایش منفرد pq است، به وسیله معادله (۵-۱۰۰) واریانس یک تراکم دوجمله‌ای را می‌توان به این صورت یافت:

$$Var[X] = Var[x_۱] + Var[x_۲] + \cdots + Var[x_n]$$
$$Var[X] = npq \qquad (۵-۱۲۰)$$

فرض کنید که ما می‌خواهیم که بدانیم چه تعداد آزمایش برنولی پیش از وقوع اولین موفقیت در یک توالی از آزمایش‌ها رخ می‌دهد. اگر آزمایش اول یک موفقیت را نتیجه دهد و احتمال موفقیت برای هر آزمایش p باشد، احتمال متغیر تصادفی X برابر با p خواهد بود. اگر احتمال شکست به‌صورت $q = ۱ - p$ داده شود و ما در آزمایش دوم موفقیت داشته باشیم، به یک احتمال pq می‌رسیم. با بسط به $k - ۱$ شکست پیش از یک موفقیت نهایی، ما آنچه با عنوان "توزیع هندسی" شناخته می‌شود را به دست می‌آوریم، که در آن:

$$f(k) = pq^{k-۱} \quad k = ۱, ۲, \ldots \qquad (۱۲۱-۵)$$

یافتن مقدار مورد انتظار تابع تراکم هندسی اندکی دشوار است ولی می‌تواند به‌صورت ذیل محقق شود. به وسیله معادله (۵-۶۷) داریم:

$$E[X] = \sum_{i=۱}^{\infty} i \; pq^{(i-۱)}$$

$$E[X] = P \sum_{i=۱}^{\infty} i \, q^{(i-۱)} \qquad (۱۲۲-۵)$$

$$E[X] = P \sum_{i=۰}^{\infty} \frac{d}{dq} \; q^i$$

$$E[X] = \frac{Pd}{dq} \sum_{i=۰}^{\infty} q^i$$

$$E[X] = \frac{pd}{dq} \left[\frac{۱}{۱-q}\right] \qquad \text{ادامه (۵-۱۲۲)}$$

$$E[X] = \frac{p}{(۱-q)^۲}$$

$$E[X] = ۱/p$$

خط پنجم مشتق بالا به این خاطر حاصل می‌شود که مقدار q کمتر از ۱ یا برابر آن است، از این‌رو، این تجمیع به $1/(1-q)$ همگرا می‌شود. واریانس تراکم هندسی در اینجا استخراج نمی‌شود بلکه به این صورت ارائه می‌شود:

$$\mathrm{Var}\,[X] = q/p^{2} \qquad (۱۲۳-۵)$$

۵-۶-۳ توزیع پواسون

یک تابع تراکم پرکاربرد که برای استخراج آمارها درباره تعداد موفقیت‌ها در طی یک دوره زمانی مشخص مفید است توزیع پواسون است. تابع تراکم پواسون عمدتاً به این خاطر رواج دارد که بسیاری از فرایندهای واقعی را خیلی خوب توصیف می‌کند. در سیستم‌های کامپیوتری، درخواست‌ها برای مشاغل در یک CPU اغلب با یک فرایند پواسون نشان داده می‌شوند. تابع تراکم پواسون به این صورت تعریف می‌شود:

$$f(x) = \frac{(e^{-\mu}\mu^{x})}{x!} \qquad x = 0, 1, 2, \ldots \qquad (۱۲۴-۵)$$

که در آن پارامتر μ به‌صورت تعداد متوسط موفقیت‌ها در طی بازه تعریف می‌شود. چند شرط باید برای وجود یک تابع تراکم غالب باشند. این‌ها عبارتند از اینکه موفقیت‌ها برای یک بازه مستقل از موفقیت‌ها در هر بازه دیگری هستند، اینکه احتمال یک موفقیت در طی یک بازه فوق‌العاده کوتاه قریب به صفر است، و اینکه احتمال تنها یک موفقیت در طی یک بازه کوتاه تنها به طول بازه بستگی دارد. جالب اینکه، متوسط توزیع پواسون بخشی از تعریفش است. مقدار مورد انتظار را می‌توان به این صورت یافت:

$$E[X] = \sum_{x=0}^{\infty} x\,\frac{e^{-\mu}\mu^{x}}{x!}$$

$$E[X] = \sum_{x=1}^{\infty} \frac{x e^{-\mu}\mu^{x}}{x!} \qquad (۱۲۵-۵)$$

$$E[X] = \mu \sum_{x=1}^{\infty} \frac{e^{-\mu}\mu^{x-1}}{(x-1)!}$$

اکنون، اگر ما فرض کنیم که $y = x - ۱$ باشد، به مجموعی از تابع تراکم از ۱ تا بینهایت می‌رسیم، که به وسیله معادله (۵-۵۱)، برابر با ۱ است:

$$E[X] = \mu \sum_{y=۰}^{\infty} \frac{e^{-\mu}\mu^{x}}{y!}$$

$$E[X] = \mu \qquad (۱۲۶-۵)$$

واریانس توزیع پواسون را می‌توان با ابتدا یافتن $E[X(X-۱)]$ و سپس استفاده از نتیجه آن در معادله (۵-۹۴) یافت:

$$E[X(X-۱)] = \sum_{x=۰}^{\infty} \frac{(x(x-۱)e^{-\mu}\mu^{x}}{x!} \qquad (۱۲۷-۵)$$

دو عبارت اول این تجمیع صفر هستند، به‌طوری‌که داریم:

$$E[X(X-۱)] = \sum_{x=۲}^{\infty} \frac{x(x-۱)e^{-\mu}\mu^{x}}{x!}$$

$$E[X(X-۱)] = \sum_{x=۲}^{\infty} \frac{e^{-\mu}\mu^{x-۲}\mu^{۲}}{(x-۲)!} \qquad (۱۲۸-۵)$$

به وسیله معادله (۵-۵۱)، و، در صورتی که فرض کنیم $x = y + ۲$ باشد، داریم:

$$E[X(X-۱)] = \mu^{۲} \sum_{y=۰}^{\infty} \frac{e^{-\mu}\mu^{y}}{y!} \qquad (۱۲۹-۵)$$

از معادله (۵-۹۴) ما داریم:

$$\sigma^{۲} = E[X^{۲}] - \mu^{۲}$$

$$\sigma^{۲} = E[X^{۲}] - E[X] + E[X] - \mu^{۲} \qquad (۱۳۰-۵)$$

از معادله (۵-۸۴) به دست می‌آوریم:

$$\sigma^{۲} = E[X^{۲} - X] + E[X] - \mu^{۲}$$

$$\sigma^{۲} = E\,[X\,(X - ۱)] + E\,[X] - \mu^{۲} \qquad (۱۳۱-۵)$$

به وسیله معادله (۵-۱۲۷)، ما به دست آوردیم:

$$\sigma^{۲} = \mu^{۲} + \mu - \mu^{۲} = \mu \qquad (۱۳۲-۵)$$

تابع تراکم قبلی چند نمونه از متغیرهای تصادفی گسسته رایج‌تر را ارائه می‌نماید. توزیع‌هایی همچون این‌ها برای مدل‌سازی فرایندهای واقعی که در آن‌ها کمیت‌های مورد نظر آیتم‌های قابل‌شمارشی هستند مفید هستند.

علاوه بر توابع تراکم گسسته اساسی که قبلاً توصیف‌شده‌اند، چند تراکم پیوسته وجود دارد. توابع تراکم پیوسته با این واقعیت مشخص می‌شوند که مقدار $f(x)$ در هر نقطه مشخص x صفر است. بااین‌حال، این احتمال که هر مقدار x بین x و یک دلتای کوچکی باشد تقریباً $f(x)$ ضرب در مقدار دلتا است.

۵-٦-٤ توزیع گوسی

یکی از مهم‌ترین توزیع‌های احتمال پیوسته، و احتمالاً پرکاربردترین آن‌ها، توزیع "نرمال" یا گوسی است.

تابع تراکم یک متغیر تصادفی نرمال X به این صورت به دست می‌آید:

$$f(x) = \frac{۱}{\sigma\sqrt{۲\pi}}\, e^{-۱/۲\left(\frac{x-\mu}{\sigma}\right)^{۲}} \qquad (۱۳۳-۵)$$

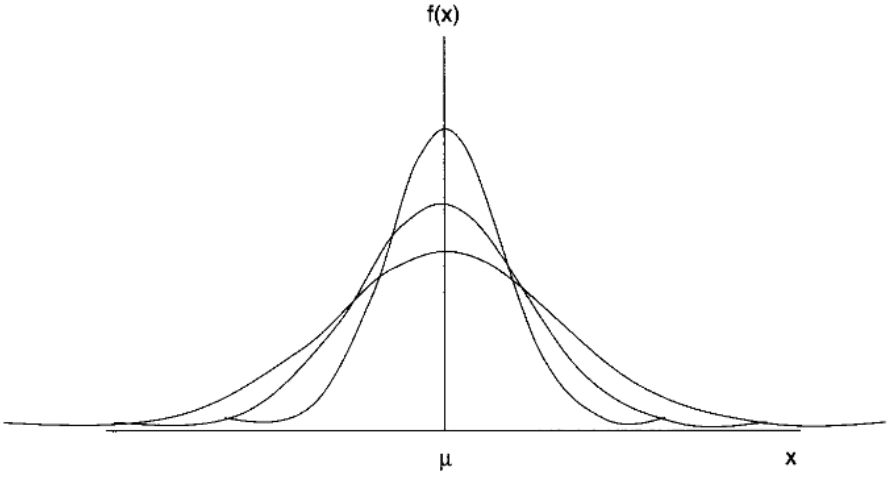
شکل (۵-۵) چند منحنی نرمال را نشان می‌دهد (که به خاطر شکل زنگ مانندشان با عنوان منحنی‌های رنگی شناخته می‌شوند). منحنی مسطح یک انحراف استاندارد بزرگ‌تر از منحنی‌های نازک‌تر دارد. مقدار مورد انتظار منحنی نرمال به‌صورت ذیل یافته می‌شود. از معادله (۵-۶۷) داریم:

$$E\,[X] = \int_{-\infty}^{\infty} \frac{x}{\sigma\sqrt{۲\pi}}\, e^{-۱/۲\left(\frac{x-\mu}{\sigma}\right)^{۲}}\, dx \qquad (۱۳۴-۵)$$

اگر ما رابطه ذیل را جایگزین کنیم:

$$y = \frac{x - \mu}{\sigma} \quad \text{یا} \quad x = \sigma y + \mu$$

$$dx = \sigma dy \qquad (۱۳۵-۵)$$



شکل (۵-۵) چند منحنی نرمال

ما داریم:

$$E[X] = \frac{۱}{\sqrt{۲\pi}} \int_{-\infty}^{\infty} (y\sigma + \mu)\ e^{-\frac{y^۲}{۲}}\ dy$$

$$E[X] = \frac{\sigma}{\sqrt{۲\pi}} \int_{-\infty}^{\infty} y\ e^{-\frac{y^۲}{۲}}\ dy + \frac{\mu}{\sqrt{۲\pi}} \int_{-\infty}^{\infty} e^{-\frac{y^۲}{۲}}\ dy \qquad (۱۳۶-۵)$$

اگر در انتگرال دوم ما y را با رابطه ذیل جایگزین کنیم:

$$y = (x - \mu)/\sigma$$

$$dy = ۱/\sigma\ dx \qquad (۱۳۷-۵)$$

مشخصات انتگرال یک تابع تراکم که برابر با ۱ است را مشاهده می‌کنیم:

$$E[X] = \frac{\sigma}{\sqrt{2\pi}} \int_{-\infty}^{\infty} y\, e^{-\frac{y^2}{2}}\, dy + \frac{\mu}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}\, dx$$

$$E[X] = \frac{\sigma}{\sqrt{2\pi}} \int_{-\infty}^{\infty} y\, e^{-\frac{y^2}{2}}\, dy + \mu \qquad (۱۳۸-۵)$$

عبارت موجود در انتگرال باقیمانده به خاطر وجود y یک تابع فرد است. از آنجا که یک تابع فرد که بر روی محدوده‌های متقارن انتگرال‌گیری می‌شود صفر است، متوسط آن به این صورت تبدیل می‌شود:

$$E[X] = \mu \qquad (۱۳۹-۵)$$

ما می‌توانیم نوع توزیع نرمال را به‌صورت ذیل بیابیم:

$$E[(X-\mu)^2] = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} (x-\mu)^2\, e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}\, dy \qquad (۱۴۰-۵)$$

با انجام همان جایگزینی قبلی به دست می‌آوریم:

$$E[(X-\mu)^2] = \frac{\sigma^2}{\sqrt{2\pi}} \int_{-\infty}^{\infty} y^2\, e^{\left(-\frac{y^2}{2}\right)}\, dy \qquad (۱۴۱-۵)$$

فرض کنید:

$$u = y, \qquad v = -e^{-\frac{y^2}{2}}$$

و

$$du = dy\,,\, dv = ye^{-\frac{y^2}{2}} \qquad (۱۴۲-۵)$$

آنگاه:

$$E\left[(X-\mu)^2\right] = \frac{\sigma^2}{\sqrt{2\pi}}\left[ye^{-\frac{y^2}{2}}\Big|_{y=-\infty}^{\infty} + \int_{-\infty}^{\infty} e^{-\frac{y^2}{2}}\,dy\right]$$

همانند قبل، بازه اول برابر با صفر است، و انتگرال دوم را می‌توان نشان داد که به این صورت است:

$$\int_{-\infty}^{\infty} e^{-\frac{y^2}{2}}\,dy = \sqrt{2\pi} \qquad (۱۴۳-۵)$$

از این‌رو داریم:

$$E\left[(X-\mu)^2\right] = \frac{\sigma^2}{\sqrt{2\pi}}\left[0 + \sqrt{2\pi}\right]$$
$$E\left[(X-\mu)^2\right] = \sigma^2 \qquad (۱۴۴-۵)$$

برای منحنی نرمال، این میانگین در این حالت رخ می‌دهد، که به‌صورت مقداری تعریف می‌شود که بیشتر در توزیع به نظر می‌رسد.

یافتن این احتمال که یک متغیر تصادفی دارای توزیع نرمال بین دو مقدار قرار می‌گیرد نیاز به حل انتگرال دارد:

$$P\left(x_1 < X < x_2\right) = \frac{1}{\sigma\sqrt{2\pi}}\int_{x_1}^{x_2} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)}\,dx \qquad (۱۴۵-۵)$$

این انتگرال به راحتی قابل‌حل نیست و بهترین راه رویکرد به آن با استفاده از میانگین‌های عددی است. بااین‌حال، برای مفید بودن، ما نیاز به تولید جدولی برای هر مقدار انحراف میانگین و استاندارد داریم. ما مایلیم تا با داشتن تنها یک منحنی نرمال استاندارد از این اجتناب کنیم. اگر ما این جایگزینی را انجام دهیم:

$$z = (x - \mu) / \sigma, \quad dx = \sigma\, dz \qquad (۱۴۶-۵)$$

در معادله قبلی، به دست می‌آوریم:

$$\frac{۱}{\sqrt{۲\pi}} \int_{z_۱}^{z_۲} e^{-\frac{z^۲}{۲}}\, dz \qquad (۱۴۷-۵)$$

که در آن:

$$z = (z - \mu)/ \sigma\ , sx = \sigma\, dz \qquad (۱۴۸-۵)$$

این عبارت معادل با یک توزیع نرمال میانگین برابر با صفر و انحراف استاندارد یک است. از این‌رو، ما می‌توانیم هر توزیع نرمال را با میانگین صفر و یک انحراف استاندارد به منحنی نرمال استاندارد تبدیل کنیم. به‌عنوان‌مثال، اجازه دهید این احتمال را محاسبه کنیم که هر متغیر تصادفی دارای توزیع نرمال در یک انحراف استاندارد متوسط قرار می‌گیرد. برای این کار، ما باید نوعی از جدول را برای توزیع نرمال استاندارد تولید کنیم. جدول (۵-۱) مقادیر را برای توزیع نرمال استاندارد انتگرال‌گیری شده از منفی بینهایت تا x ارائه نموده است.

$$\frac{۱}{\sqrt{۲\pi}} \int_{-\infty}^{x} e^{-\frac{x^۲}{۲}}\, dx$$

فرض کنید:

$$x_۱ = \mu - \text{sigma} \quad \text{and} \ x_۲ = \mu + \sigma$$

آنگاه:

$$z_۱ = (\mu - \sigma - \mu) / \sigma \text{ and } z_۲ = (\mu + \sigma - \mu) / \sigma$$

$$z_۱ = -۱ \ , \ z_۲ = ۱ \qquad (۱۴۹-۵)$$

سمت چپ معادله (۵-۱۴۳) را می‌توان به این صورت بازنویسی کرد:

$$P(x_۱ < X < x_۲) = P(X < x_۲) - P(X < x_۱) \qquad (۱۵۰-۵)$$

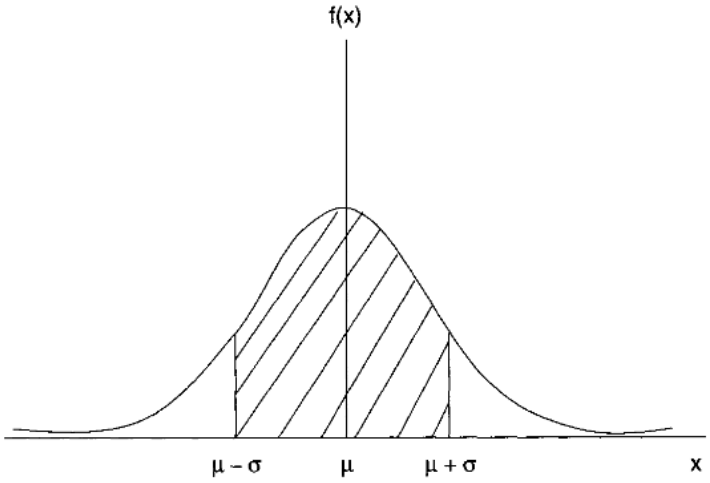
جدول (۵-۱) مقادیر منحنی نرمال استاندارد

X+۰,۰	X+۰,۰۱	X+۰,۰۲	X+۰,۰۳	X+۰,۰۴	X+۰,۰۵	X+۰,۰۶	X+۰,۰۷	X+۰,۰۸	X+۰,۰۹
۰,۵۰۰۰۰	۰,۵۰۴۰۱	۰,۵۰۸۰۰	۰,۵۱۱۹۹	۰,۵۱۵۹۷	۰,۵۱۹۹۶	۰,۵۲۳۹۴	۰,۵۲۷۹۴	۰,۵۳۱۹۴	۰,۵۳۵۹۴
۰,۵۳۹۹۳	۰,۵۴۳۹۱	۰,۵۴۷۹۰	۰,۵۵۱۸۷	۰,۵۵۵۸۵	۰,۵۵۹۸۱	۰,۵۶۳۷۸	۰,۵۶۷۷۳	۰,۵۷۱۶۸	۰,۵۷۵۶۲
۰,۵۷۹۵۵	۰,۵۸۳۴۸	۰,۵۸۷۴۰	۰,۵۹۱۳۰	۰,۵۹۵۲۰	۰,۵۹۹۰۹	۰,۶۰۲۹۷	۰,۶۰۶۸۴	۰,۶۱۰۷۰	۰,۶۱۴۵۵
۰,۶۱۸۳۹	۰,۶۲۲۲۱	۰,۶۲۶۰۳	۰,۶۲۹۸۳	۰,۶۳۳۶۲	۰,۶۳۷۳۹	۰,۶۴۱۱۶	۰,۶۴۴۹۰	۰,۶۴۸۶۴	۰,۶۵۲۳۶
۰,۶۵۶۰۷	۰,۶۵۹۷۶	۰,۶۶۳۴۳	۰,۶۶۷۰۹	۰,۶۷۰۷۴	۰,۶۷۴۳۶	۰,۶۷۷۹۸	۰,۶۸۱۵۷	۰,۶۸۵۱۵	۰,۶۸۸۷۱
۰,۶۹۲۲۵	۰,۶۹۵۷۸	۰,۶۹۹۲۹	۰,۷۰۲۷۷	۰,۷۰۶۲۴	۰,۷۰۹۷۰	۰,۷۱۳۱۳	۰,۷۱۶۵۴	۰,۷۱۹۹۳	۰,۷۲۳۳۱

٠,٧٢٦	٠,٧٢٩	٠,٧٣٣	٠,٧٣٦	٠,٧٣٩	٠,٧٤٣	٠,٧٤٦	٠,٧٤٩	٠,٧٥٢	٠,٧٥٥
٦٦	٩٩	٣١	٦٠	٨٧	١٢	٣٥	٥٦	٧٤	٩١
٠,٧٥٩	٠,٧٦٢	٠,٧٦٥	٠,٧٦٨	٠,٧٧١	٠,٧٧٤	٠,٧٧٧	٠,٧٨٠	٠,٧٨٣	٠,٧٨٦
٠٥	١٧	٢٧	٣٤	٣٩	٤٢	٤٣	٤١	٣٧	٣١
٠,٧٨٩	٠,٧٩٢	٠,٧٩٤	٠,٧٩٧	٠,٨٠٠	٠,٨٠٣	٠,٨٠٦	٠,٨٠٨	٠,٨١١	٠,٨١٤
٢٣	١٢	٩٨	٨٣	٦٥	٤٤	٢٢	٩٧	٦٩	٣٩
٠,٨١٧	٠,٨١٩	٠,٨٢٢	٠,٨٢٤	٠,٨٢٧	٠,٨٣٠	٠,٨٣٢	٠,٨٣٥	٠,٨٣٧	٠,٨٤٠
٠٧	٧٢	٣٤	٩٥	٥٣	٠٨	٦١	١٢	٦٠	٠٦
٠,٨٤٢	٠,٨٤٤	٠,٨٤٧	٠,٨٤٩	٠,٨٥١	٠,٨٥٤	٠,٨٥٦	٠,٨٥٨	٠,٨٦٠	٠,٨٦٣
٤٨	٨٧	٢٤	٥٩	٩٢	٢١	٤٩	٧٤	٩٧	١٧
٠,٨٦٥	٠,٨٦٧	٠,٨٦٩	٠,٨٧١	٠,٨٧٣	٠,٨٧٥	٠,٨٧٧	٠,٨٧٩	٠,٨٨١	٠,٨٨٣
٣٥	٥١	٦٤	٧٤	٨٣	٨٩	٩٣	٩٤	٩٣	٨٩
٠,٨٨٥	٠,٨٨٧	٠,٨٨٩	٠,٨٩١	٠,٨٩٣	٠,٨٩٥	٠,٨٩٧	٠,٨٩٨	٠,٩٠٠	٠,٩٠٢
٨٤	٧٥	٦٥	٥٢	٣٧	٢٠	٠١	٧٩	٥٥	٢٨
٠,٩٠٤	٠,٩٠٥	٠,٩٠٧	٠,٩٠٩	٠,٩١٠	٠,٩١٢	٠,٩١٣	٠,٩١٥	٠,٩١٦	٠,٩١٨
٠٠	٦٩	٣٦	٠١	٦٣	٢٤	٨٢	٣٨	٩٢	٤٤
٠,٩١٩	٠,٩٢١	٠,٩٢٢	٠,٩٢٤	٠,٩٢٥	٠,٩٢٧	٠,٩٢٨	٠,٩٢٩	٠,٩٣١	٠,٩٣٢
٩٤	٤٢	٨٧	٣١	٧٢	١٢	٤٩	٨٥	١٨	٥٠
٠,٩٣٣	٠,٩٣٥	٠,٩٣٦	٠,٩٣٧	٠,٩٣٨	٠,٩٣٩	٠,٩٤١	٠,٩٤٢	٠,٩٤٣	٠,٩٤٤
٧٩	٠٧	٣٣	٥٦	٧٨	٩٨	١٧	٣٣	٤٨	٦٠
٠,٩٤٥	٠,٩٤٦	٠,٩٤٧	٠,٩٤٨	٠,٩٤٩	٠,٩٥١	٠,٩٥٢	٠,٩٥٣	٠,٩٥٣	٠,٩٥٤
٧١	٨١	٨٨	٩٤	٩٨	٠٠	٠١	٠٠	٩٧	٩٣
٠,٩٥٥	٠,٩٥٦	٠,٩٥٧	٠,٩٥٨	٠,٩٥٩	٠,٩٦٠	٠,٩٦١	٠,٩٦٢	٠,٩٦٢	٠,٩٦٣
٨٧	٧٩	٧٠	٦٠	٤٧	٣٤	١٩	٠٢	٨٤	٦٤

۰,۹۶۴	۰,۹۶۵	۰,۹۶۵	۰,۹۶۶	۰,۹۶۷	۰,۹۶۸	۰,۹۶۸	۰,۹۶۹	۰,۹۷۰	۰,۹۷۰
۴۳	۲۱	۹۷	۷۲	۴۵	۱۷	۸۸	۵۷	۲۶	۹۳
۰,۹۷۱	۰,۹۷۲	۰,۹۷۲	۰,۹۷۳	۰,۹۷۴	۰,۹۷۴	۰,۹۷۵	۰,۹۷۵	۰,۹۷۶	۰,۹۷۶
۵۸	۲۳	۸۶	۴۸	۰۹	۶۸	۲۷	۸۴	۴۰	۹۵
۰,۹۷۷	۰,۹۷۸	۰,۹۷۸	۰,۹۷۹	۰,۹۷۹	۰,۹۸۰	۰,۹۸۰	۰,۹۸۱	۰,۹۸۱	۰,۹۸۱
۴۹	۰۳	۵۵	۰۶	۵۶	۰۵	۵۳	۰۰	۴۶	۹۱
۰,۹۸۲	۰,۹۸۲	۰,۹۸۳	۰,۹۸۳	۰,۹۸۴	۰,۹۸۴	۰,۹۸۴	۰,۹۸۵	۰,۹۸۵	۰,۹۸۵
۳۵	۷۹	۲۱	۶۳	۰۳	۴۳	۸۲	۲۰	۵۷	۹۳
۰,۹۸۶	۰,۹۸۶	۰,۹۸۶	۰,۹۸۷	۰,۹۸۷	۰,۹۸۷	۰,۹۸۸	۰,۹۸۸	۰,۹۸۸	۰,۹۸۹
۲۹	۶۴	۹۸	۳۱	۶۴	۹۵	۲۷	۵۷	۸۷	۱۶
۰,۹۸۹	۰,۹۸۹	۰,۹۸۹	۰,۹۹۰	۰,۹۹۰	۰,۹۹۰	۰,۹۹۱	۰,۹۹۱	۰,۹۹۱	۰,۹۹۱
۴۴	۷۲	۹۹	۲۵	۵۱	۷۷	۰۱	۲۵	۴۹	۷۲
۰,۹۹۱	۰,۹۹۲	۰,۹۹۲	۰,۹۹۲	۰,۹۹۲	۰,۹۹۲	۰,۹۹۳	۰,۹۹۳	۰,۹۹۳	۰,۹۹۳
۹۴	۱۶	۳۷	۵۸	۷۹	۹۸	۱۸	۳۷	۵۵	۷۳
۰,۹۹۳	۰,۹۹۴	۰,۹۹۴	۰,۹۹۴	۰,۹۹۴	۰,۹۹۴	۰,۹۹۴	۰,۹۹۵	۰,۹۹۵	۰,۹۹۵
۹۱	۰۸	۲۴	۴۱	۵۶	۷۲	۸۷	۰۲	۱۶	۳۰
۰,۹۹۵	۰,۹۹۵	۰,۹۹۵	۰,۹۹۵	۰,۹۹۵	۰,۹۹۶	۰,۹۹۶	۰,۹۹۶	۰,۹۹۶	۰,۹۹۶
۴۳	۵۶	۶۹	۸۲	۹۴	۰۶	۱۸	۲۹	۴۰	۵۰
۰,۹۹۶	۰,۹۹۶	۰,۹۹۶	۰,۹۹۶	۰,۹۹۷	۰,۹۹۷	۰,۹۹۷	۰,۹۹۷	۰,۹۹۷	۰,۹۹۷
۶۱	۷۱	۸۱	۹۰	۰۰	۰۹	۱۷	۲۶	۳۴	۴۲
۰,۹۹۷	۰,۹۹۷	۰,۹۹۷	۰,۹۹۷	۰,۹۹۷	۰,۹۹۷	۰,۹۹۷	۰,۹۹۸	۰,۹۹۸	۰,۹۹۸
۵۰	۵۸	۶۵	۷۳	۸۰	۸۷	۹۳	۰۰	۰۶	۱۲
۰,۹۹۸	۰,۹۹۸	۰,۹۹۸	۰,۹۹۸	۰,۹۹۸	۰,۹۹۸	۰,۹۹۸	۰,۹۹۸	۰,۹۹۸	۰,۹۹۸
۱۸	۲۴	۲۹	۳۵	۴۰	۴۵	۵۰	۵۵	۵۹	۶۴

٠,٩٩٩	٠,٩٩٨	٠,٩٩٨	٠,٩٩٨	٠,٩٩٨	٠,٩٩٨	٠,٩٩٨	٠,٩٩٨	٠,٩٩٨	٠,٩٩٨
٠٦	٩٩	٩٦	٩٢	٨٠	٨٥	٨١	٧٧	٧٣	٦٨
٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩
٣٠	٢٨	٢٦	٢٣	٢٠	١٨	١٥	١٢	٠٩	٠٦
٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩
٥١	٤٩	٤٧	٤٥	٤٤	٤٢	٣٩	٣٧	٣٥	٣٣
٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩
٦٦	٦٤	٦٣	٦٢	٦٠	٥٩	٥٧	٥٦	٥٤	٥٣
٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩
٧٦	٧٥	٧٤	٧٣	٧٢	٧١	٧٠	٦٩	٦٨	٦٧
٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩
٨٣	٨٣	٨٢	٨١	٨١	٨٠	٧٩	٧٨	٧٨	٧٧
٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩
٨٨	٨٨	٨٨	٨٧	٨٧	٨٦	٨٦	٨٥	٨٤	٨٤
٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩
٩٢	٩٢	٩١	٩١	٩١	٩٠	٩٠	٩٠	٨٩	٨٩
٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩
٩٤	٩٤	٩٤	٩٤	٩٣	٩٣	٩٣	٩٣	٩٢	٩٢
٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩
٩٦	٩٦	٩٦	٩٥	٩٥	٩٥	٩٥	٩٥	٩٥	٩٤
٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩	٠,٩٩٩
٩٧	٩٧	٩٧	٩٧	٩٧	٩٧	٩٦	٩٦	٩٦	٩٦



شکل (۵-۶) ناحیه منتخب زیر یک منحنی توزیع نرمال

مقادیر برگرفته از جدول برای $z = -۱$ و ۱، به ترتیب، برابرند با:

$$P(z < -۱) = ۰.۱۵۸۷$$

$$P(z < ۱) = ۰.۸۴۱۳ \qquad (۱۵۱-۵)$$

از این‌رو، داریم:

$$P(x_۱ < X < x_۲) = P(X < x_۲) - P(X < x_۱) \qquad (۱۵۲-۵)$$

شکل (۵-۶) ناحیه منتخب زیر منحنی نرمال را نشان می‌دهد.

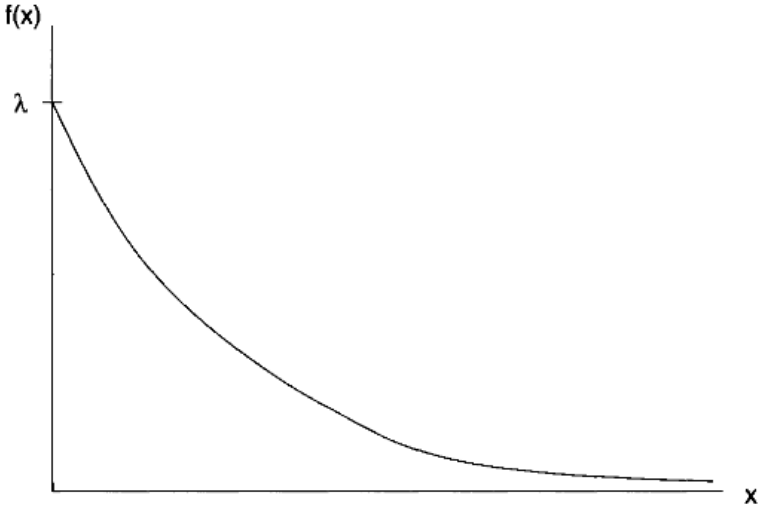
۵-۶-۵ توزیع نمایی

یک توزیع پیوسته ساده‌تر، یعنی توزیع نمایی، در تئوری صف بندی مهم است و از این‌رو در اینجا مورد بحث قرار می‌گیرد. جذابیت اصلی آن این است که این دارای خصوصیت مارکوین است، که بیان می‌کند که احتمال وقوع یک رویداد کاملاً مستقل از تاریخچه آزمایش است. این مشخصه "بدون حافظه" نیز نامیده می‌شود. این عبارت برای یک توزیع نمایی به این صورت به دست می‌آید:

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & x > 0 \\ 0 & \text{در غیر این صورت} \end{cases} \qquad (۱۵۳-۵)$$

گراف یک منحنی نمایی در شکل (۵-۷) نشان داده‌شده است.

ما بعداً خواهیم دید که توزیع نمایی، به خاطر خصوصیت مارکوین آن، برای ارائه توزیع‌های زمانی سرویس در سیستم‌های صف بندی مفید خواهد بود.



شکل (۵-۷): تابع تراکم احتمال نمایی

فرض کنید که زمانی که یک کاربر کامپیوتر در یک پایانه سیستم صرف می‌کند دارای توزیع نمایی در طی زمان است. این احتمال که کاربر به مدت n دقیقه در یک نهایی خواهد بود از این رابطه به دست می‌آید:

$$P(X \geq n) = \int_{n}^{\infty} f(x)\, dx$$

$$P(X \geq n) = \int_{n}^{\infty} \lambda e^{-\lambda x} dx \qquad (۱۵۴-۵)$$

$$P(X \geq n)\, e^{-n\lambda}$$

تابع توزیع احتمال برای این تابع‌نمایی در شکل (۵-۸) نشان داده‌شده و به این صورت به دست می‌آید:

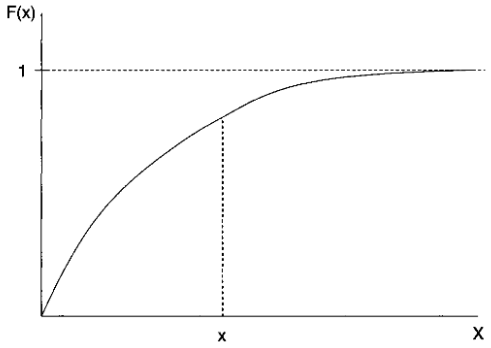
$$F(x) = P(X < x) = \begin{cases} ۱ - e^{-\lambda x} & x > ۰ \\ ۰ & \text{در غیر این صورت} \end{cases} \qquad (۱۵۵-۵)$$

همانند هر تابع توزیعی، ما می‌توانیم نتیجه یکسانی را با انتخاب نقطه n بیابیم، که نشانگر این احتمال است که کاربر در کمتر از n دقیقه در یک پایانه خواهد بود، و از معادله (۵-۱۹) برای یافتن احتمال رویداد مکمل استفاده کنید (شکل (۵-۸) را نیز ببینید):

$$P(X \geq n) = ۱ - F(n)$$

$$P(X \geq n) = ۱ - (۱ - e^{-nx}) \qquad (۱۵۶-۵)$$

$$P(X \geq n) = e^{-nx}$$



شکل (۵-۸) تابع توزیع احتمال نمایی

متوسط متغیر تصادفی نمایی به این صورت یافت می‌شود:

$$E[X] = \int_{\cdot}^{\infty} x\,\lambda\, e^{-\lambda x}\ dx \qquad (۱۵۷-۵)$$

اگر ما فرض کنیم:

$$u = x \quad , \qquad v = -e^{-\lambda x}$$

(۱۵۸-۵)

$$du = dx \quad , \quad dx = \lambda e^{\lambda x}\ dx$$

و با اجزاء انتگرال‌گیری کنید که در آن:

$$\int_a^b u\ dv = uv\ \Big|_a^b - \int_a^b v\, du \qquad (۱۵۹-۵)$$

به دست می‌آوریم:

$$E[X] - xe^{-\lambda x}\ \Big|_{\cdot}^{\infty} - \int_a^b - e^{-\lambda x}\ dx \qquad (۱۶۰-۵)$$

زیرا:

$$\lim_{x\to\infty} xe^{-\lambda x} = ۰$$

(۱۶۱-۵)

$$E[X] = ۰ - \int_{۰}^{\infty} -e^{-\lambda x}\, dx$$

$$E[X] = -۱/\lambda\, e^{-\lambda x} \Big|_{۰}^{\infty}$$

$$E[X] = \lim_{x\to\infty} ۱/\lambda\, e^{-\lambda x} + ۱/\lambda$$ ادامه (۱۶۱-۵)

$$E[X] = ۱/\lambda$$

ما ممکن است واریانس متغیر تصادفی نمایی را به‌صورت ذیل بیابیم:

$$var[X] = E[(X-\mu)^{۲}] = \int_{-\infty}^{\infty} (x-\mu)^{۲}\, \lambda e^{-\lambda x}\, dx$$

(۱۶۲-۵)

$$var[X] = \int_{-\infty}^{\infty} \lambda x^{۲} e^{-\lambda x}\, dx - ۲\mu \int_{-\infty}^{\infty} \lambda x e^{-\lambda x}\, dx + \mu^{۲}\lambda \int_{۰}^{\infty} e^{-\lambda x}\, dx$$

می‌توانیم ببینیم که توسط معادلات (۵-۱۵۷) تا (۵-۱۶۱)، بازه دوم تا $۱/\lambda$ ارزیابی می‌کند. بازه سوم تا $۱/\lambda$ ارزیابی می‌شود (همان‌گونه که در (۵-۱۶۱) نشان داده‌شده است). برای حل این بازه اول، ما تابع گاما را معرفی می‌کنیم، که به این صورت نشان داده می‌شود:

$$\Gamma [t] = \int_{۰}^{\infty} x^{t-۱} e^{-x} \, dx \qquad (۱۶۳-۵)$$

تابع گاما را می‌توان برای یک مقدار مثبت پارامتر حل کرد تا این رابطه حاصل شود:

$$\Gamma [n] = (n - ۱) ! \qquad (۱۶۴-۵)$$

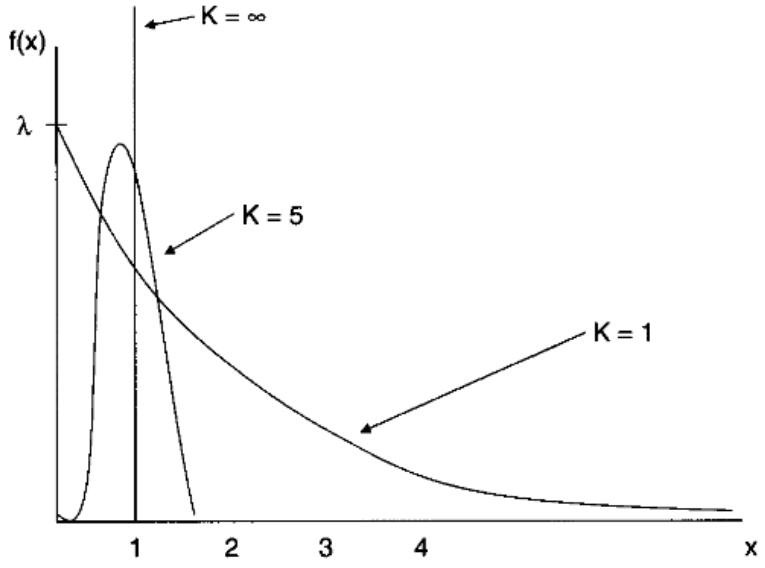
ما اکنون می‌توانیم از معادله (۵-۱۶۴) برای کمک به یافتن راه‌حل معادله (۵-۱۶۲) و رسیدن به واریانسی برای یک توزیع نمایی استفاده کنیم:

$$var [X] = ۱ / \lambda^{۲} \qquad (۱۶۵-۵)$$

۵-٦-٦ توزیع ارلانگ

تابع تراکم نمایی اغلب برای نمایش زمان سرویس یک سرور در انتهای یک خط انتظار استفاده می‌شود. در برخی از موارد، مطلوب است که چند سرور همسان با یک تابع تراکم واحد ارائه شوند که آمارهایشان برای یک سرور نمایی معادل یکسان است. توزیعی که این شرایط را فراهم می‌کند "توزیع ارلانگ" نامیده می‌شود و به این صورت ارائه می‌شود:

$$f (x) = \begin{cases} \frac{\lambda k (\lambda kx)^{K-۱} e^{-\lambda kx}}{(k-۱)!} & x > ۰ \end{cases} \qquad (۱۶۶-۵)$$



شکل (۵-۹) توابع تراکم ارلانگ برای مقادیر منتخب k.

که در آن پارامترهای λ و k وجود دارند. شکل (۵-۹) یک گراف تابع تراکم ارلانگ را برای مقادیر مختلف k برای یک مقدار مشخص λ نشان می‌دهد. انحراف‌های متوسط و استاندارد تابع تراکم ارلانگ به این صورت ارائه می‌شوند:

$$E[X] = ۱ / \lambda \qquad (۱۶۷-۵)$$

$$var[X] = ۱ / k\lambda^{۲} \qquad (۱۶۸-۵)$$

تابع توزیع احتمال به این صورت ارائه می‌شود:

$$F(x) = ۱ - e^{-\lambda kx}\left[۱ + \sum_{i=۰}^{k-۱}\frac{(k\lambda x)^{i}}{i!}\right] \qquad (۱۶۹-۵)$$

ذکر این نکته مهم است که مقدار مورد انتظار برای یک نمایی با همان پارامتر λ یکسان است، و مستقل از تعداد سرورهای ارلانگی است (مثلاً سرورهای موازی).

۵-۷ خلاصه

این فصل برخی از مفاهیم احتمال پایه که برای درک و تحلیل مدل‌های شبکه صف بندی مفید هستند را معرفی نمود. بسیاری از تراکم‌های احتمال اضافی پیچیده‌تر مشخص‌شده است که برای ارائه انواع مشخصی از فرایندهای واقعی مفید هستند. این‌ها برای حوزه عمل این کتاب هستند ولی ممکن است در بسیاری از متون احتمالات و آمار یافت شوند (مراجع [۶-۲] را ببینید). بااین‌حال، چگالی‌های ارائه شده در این کتاب به شکلی رایج به خاطر کاربردپذیری‌شان برای بسیاری از فرایندهای ورودی و سرویس و به خاطر سادگی محاسباتی نسبی‌شان در تحلیل صف بندی استفاده می‌شوند.

فصل ششم

فرآیندهای احتمالی

۶-۱ مقدمه

فرآیندهای مارکوف ابزارهای تحلیلی قدرتمندی هستند که برای تحلیل سیستم‌های کامپیوتری قابل استفاده هستند. آن‌ها ابزارهای دقیق، و در عین حال نسبتاً ساده‌ای را برای ایجاد نمایشی از سیستم‌ها و برای تحلیل سیستم کامپیوتری ارائه می‌دهند. فرآیندهای مارکوف نیازمند درک عمیقی از فرآیندهای احتمالی و تحلیل آن‌ها هستند. این فصل پیش‌زمینه لازم برای اجرای مدل‌سازی و تحلیل این‌چنین سیستم‌هایی را ارائه می‌دهد.

۶-۲ تعاریف پایه

یک فرآیند احتمالی شامل نمایش خانواده‌ای از متغیرهای تصادفی است. یک متغیر تصادفی به‌عنوان تابعی از یک متغیر، $f(x)$ نشان داده می‌شود که مقدار را با نتایج برخی از آزمایش‌ها تقریب می‌زند. متغیر X یکی از مقادیر ممکن از یک خانواده از متغیرها، از یک فضای نمونه که با Ω نشان داده‌شده، است. برای مثال، برای پرتاب سکه کل فضای نمونه Ω = {پشت، رو} است، و متغیر تصادفی X ممکن است برابر نگاشت $x = \{۱,۰\}$ باشد، که نمایش‌دهنده نگاشت کاربردی مجموعه رخداد {پشت، رو} برای مجموعه نگاشت متغیر تصادفی رخداد $\{۱,۰\}$ است. (جدول ۶-۱)

یک فرآیند احتمالی به‌عنوان خانواده‌ای از متغیرهای تصادفی نشان داده می‌شود که با $X(t)$ نشان داده می‌شود، که در آن مقدار متغیر تصادفی X برای هر مقدار t وجود دارد. متغیر تصادفی، X، یک مجموعه از مقادیر ممکن تعریف‌شده با فضای حالت، $X(t)$ است، برای متغیر تصادفی X، با مقدار انتخاب‌شده با مجموعه پارامتر، T (که مجموعه شاخص گویند)، که مقادیر آن‌ها از زیرمجموعه‌ای از مجموعه شاخص T استنتاج شده است، تعریف‌شده است.

Events	=	Heads	Tails
		↓	↓
X	=	۰	۱

جدول (۶-۱): نگاشت کاربردی

با متغیرهای تصادفی، فرایندهای احتمالی به فرآیندهای پیوسته و گسسته دسته‌بندی‌شده است. فرآیندهای احتمالی می‌توانند فضای حالت پیوسته و گسسته و یا مجموعه شاخص پیوسته یا گسسته داشته باشند. برای مثال، تعداد دستورات، $X(t)$، با سیستم کامپیوتری با اشتراک زمانی در طول بازه زمانی $(۰, t)$ می‌تواند با داشتن پارامتر شاخص پیوسته و یک فضای حالت گسسته نشان داده‌شده است. مثال دوم، می‌تواند تعداد دانشجویانی باشد که در دوره دهم یک سخنرانی حضور داشتند. این مقدار می‌توانند با داشتن مجموعه شاخص گسسته و فضای حالت گسسته نشان داده شود. به طور کل، اگر تعداد حالات در فضای حالت محدود باشد، سپس فرآیند احتمالی دارای یک فضای حالت گسسته است. به همین ترتیب، اگر مجموعه شاخص برای فضای حالت محدود و قابل‌شمارش باشد، سپس مجموعه شاخص نیز گسسته است. برای سیستم‌های پیوسته، تعداد مقادیر ممکن برای متغیرها گسسته نمی‌باشد (برای مثال مقدار واقعی). برای اینکه مجموعه شاخص پیوسته باشد، مجموعه مقادیر ممکن باید حقیقی باشد و می‌تواند به بی‌نهایت برود.

یک فرم مهم فرآیند احتمالی فرآیندهای پیوسته هستند. یک فرآیند احتمالی پیوسته موردی است که می‌خواهیم تعداد رخدادهایی را که در برخی از بازه‌های زمانی رخ می‌دهند را بشماریم، که با $N(t)$ نشان داده می‌شود، که در آن N از مجموعه گسسته اعداد صحیح مثبت پیوسته از مجموعه $\{ ۰,۱,۲,۳.... \}$ استنتاج می‌شود. علاوه بر این، مجموعه شاخص برای این فرآیند احتمالی پیوسته‌ای از فضای زمانی پیوسته استنتاج می‌شود، جایی که زمان از برخی از نقاط مرجع است $\{t >= ۰\}$. الزامات این فرآیند پیچیده این است که برای مقدار مجموعه شاخص ۰، متغیر تصادفی $N(۰) = ۰$ است. برای مقادیر $t < ۰$، مقدار $N(t)$ تعریف‌نشده است. این حاکی از این است که $N(t)$ تنها برای مقادیر

مجموعه شاخص‌های بالای ۰ وجود دارد، و مقادیر $N(t)$ برای همه دامنه‌های t بالای ۰ مقادیر غیر منفی مثبت هستند. خاصیت دوم برای یک فرآیند احتمالی پیوسته به رابطهای که مقادیر گسسته در فضای حالت با یکدیگر دارند، می‌پردازد. برای هر دو مقدار مجموعه شاخص، برای مثال، شاخص‌های s و t، در جایی که $s < t$ است، مقادیر متغیرهای تصادفی باید رابطه $X(s) =< X(t)$ را داشته باشند. سرانجام، اگر به دنبال بازه‌هایی از مقادیر برای مجموعه شاخص بگردیم، برای مثال، مقادیر s و t، $N(t) - N(s)$ تعداد رخدادها ا از رخدادهای ما نشان می‌دهند که در زمان t بعد از زمان s رخ می‌دهند و به t محدود می‌شوند.

در زمان بحث بر فرآیندهای احتمالی، اغلب توانایی تعیین مرتبه یک تابع مهم است، چرا که می‌توانیم بر مؤلفه اصلی تمرکز کنیم. یک روش انجام این کار استفاده از مفاهیمی از تحلیل و علوم کامپیوتر الگوریتم‌ها است. تعریف "مرتبه" محاسباتی برای یک الگوریتم اغلب مرتبه یک تابع گفته می‌شود. دو مورد متداول $little - oh$، است، که $o(h)$، و $big - oh$ ، که $O(h)$ نوشته می‌شود، که در آن h متغیر تابع را نشان می‌دهد. oh کوچک مرتبه یا اندازه یک تابع را به‌عنوان مقدار یک تابع تشریح می‌کند، زمانی که بر مقدار h تقسیم می‌شود، به مقدار حد o نزدیک می‌شود، و آن نیز به ۰ نزدیک می‌شود. که با رابطه زیر نشان داده می‌شود:

$$\lim_{h\to ۰} \frac{f(h)}{h} = ۰ \qquad (۱-۶)$$

اگر یک تابع h، در زمانی که بر h تقسیم می‌شود، نتیجه آن ۰ نباشد، سپس تابع $o(h)$ نیست. اگر با نزدیک شدن حد به ۰ به صفر نزدیک شود، سپس تابع $o(h)$ است. برای مثال:

$f(x) = x^۲$ is $o(h)$ زیرا

$$\lim_{x\to ۰} \frac{x^۲}{x} = \lim_{x\to ۰} x = ۰ \qquad (۲-۶)$$

$f(x) = x \quad \text{is not } o(h)$ زیرا

$$\lim_{x \to 0} \frac{x}{x} = 1 \neq 0$$

این مفهوم مرتبه یک تابع می‌تواند در درک فرآیندهای احتمالی و ساده‌سازی تحلیل آن‌ها استفاده شود، همان‌طور که نشان داده‌شده است. برای مثال، فرض کنید x یک متغیر تصادفی نمایی با پارامتر λ است و با تابع احتمال زیر تشریح شده است:

$$P[x \leq h] = 1 - e^{-\lambda h} \qquad (۳-۶)$$

ما می‌خواهیم تعیین کنیم که با چه احتمالی x کمتر از $t + h$ است و با چه احتمالی بزرگ‌تر از t است (شکل ۶-۱).

$$P[x \leq t + h \,|x > t] = P[x \leq h] \qquad (۴-۶)$$

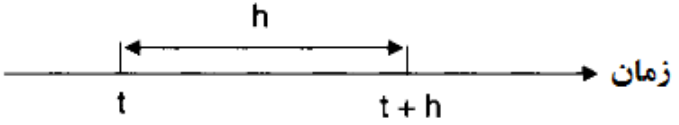
با یک پرش کوتاه به جلو استفاده از یک مفهومی که تشریح نشده است، از ویژگی مارکوف توزیع‌های نمایی، می‌توانیم نشان دهیم که روابط زیر برقرار هستند:

$$= 1 - e^{-\lambda h}$$

$$= 1 - [1 - \lambda h] + \sum_{n=2}^{\infty} \frac{(-\lambda h)^n}{n!} \qquad (۵-۶)$$

$$= \lambda h + o(h)$$

این نشان می‌دهد که مرتبه تابع $o(h)$ است.



شکل (۶-۱): فرآیند احتمالی $P\,[x \le t + h \,|x > t]$

خاصیت مهم دیگر استقلال و رشد ثابت است. یک فرآیند احتمالی رشد مستقل دارد ، اگر رخدادها در فضای نمونه $\{x(t), t \ge ۰\}$، در بازه‌های غیر همپوشانی رخ دهند، که مقادیر یکسانی ندارند. برای مثال، با توجه به شکل (۶-۲)، $x(b۱) - x(a۱) \neq x(b۲) - x(a۲)$ برقرار است.

یک فرآیند احتمالی دارای یک رشد ثابت است، اگر مقادیر یک متغیر تصادفی بر روی یک محدوده مشابه معادل باشد. برای مثال، اگر بر روی دو بازه $x(t)$، $x(s)$ و $x(t + h)$، $x(s + h)$، مقادیری برای $x(t + h) - x(s + h)$ دارای توزیع مشابهی مانند $x(t) - x(s)$ برای همه مقادیر $h >$ ۰ باشد، سپس فرآیندهای احتمالی رشد ثابت دارند. (شکل ۶-۳).

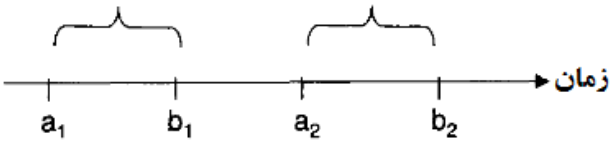
روش دیگر جستجو در این تعریف این است که اگر $x(t) - x(s) = x(t + h) - x(s + h)$ برقرار باشد، سپس این فرآیند احتمالی دارای رشد ثابت است.

به‌عنوان یک مثال، فرض می‌کنیم که $N(t)$ تعداد تماس‌های تلفنی هندل شده توسط دفتر مرکزی خاص بین اواسط شب، و زمان t، در یک روز کاری است (شکل ۶-۴).

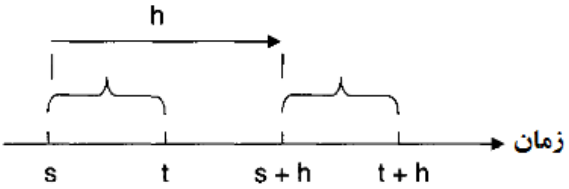
این فرآیندها می‌تواند با رشد مستقل دیده شوند، نه رشد ثابت. اگر به دو مقدار با توجه به زمان نگاه کنیم، ۸:۰۰ صبح و ۱۲:۰۰ ظهر، مقادیر از ۸:۰۰ صبح تا ۱۰:۰۰ صبح و از ۱۲:۰۰ ظهر تا ۲:۰۰ بعدازظهر است، که مقادیر مشابهی را برای متغیر $N(t)$ نشان نمی‌دهند. بنابراین، این فرآیند احتمالی دارای بازه‌های ثابتی نیستند بنابراین دارای رشد مستقل هستند.

این مفاهیم فضای حالت پیوسته و گسسته و مجموعه شاخص پیوسته و گسسته، همراه با مفاهیم رشد مستقل و ثابت، می‌توانند برای درک بیشتر مشخصه‌های سیستم‌های گوناگون استفاده شوند، برای مثال، به فرآیند احتمالی دیگری مانند پرتاب یک سکه نگاه کنید. می‌توانیم یک فرآیند احتمالی را با شمارش

تعداد پشت‌وروها در طول n پرتاب سکه متوازن تشریح کنیم. این فرآیندهای احتمالی را فرآیند برنولی گویند.



شکل (۶-۲): فرآیند احتمالی مستقل



شکل (۶-۳): فرآیند احتمالی ثابت

اگر فرض کنیم که $X_1, X_2, X_3, \ldots$ متغیرهای تصادفی برنولی باشند که به‌صورت یکسان و مستقلی توصیف‌شده‌اند، مشخصه هر مقدار xبه شکل زیر است:

$$X_i = \begin{cases} ۱ & \text{با احتمال } P \\ ۰ & \text{با احتمال } ۱ - p \end{cases}$$

مقدار ۱ یک نتیجه موفقیت‌آمیز را نشان می‌دهد (پرتاب به رو) و ۰ یک شکست را نشان می‌دهد.

$$S_n = X_۱ + X_۲ + X_۳ + \ldots + X_n \qquad (۶-۶)$$

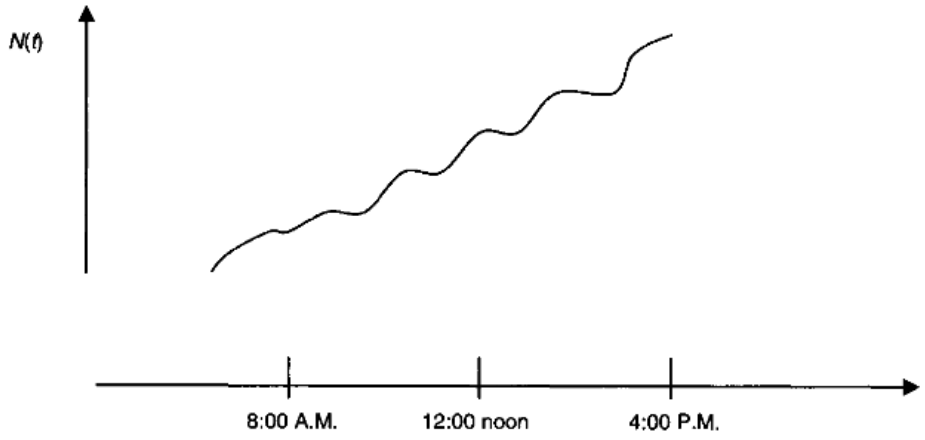
بنابراین S_n یک فرآیند برنولی است، (پارامتر گسسته، حالت گسسته). برای هر n، S_n یک توزیع دوجمله‌ای است:

$$P\,[S_n = k] = \binom{n}{k} P^K (۱ - P)^{n-k} \qquad (۷-۶)$$

$$K = ۰, ۱, ۲, \ldots, n$$

با شروع در هر نقطه‌ای درون فضای نمونه، تعداد دنباله‌ها، y، قبل از موفقیت بعدی دارای توزیع هندسی با احتمال زیر هستند:

$$P\,[y = k] = (۱ - P)^k P \qquad K = ۰, ۱, \ldots \qquad (۸-۶)$$



شکل (۶-۴): مثال حجم تماس‌های تلفنی

۶-۳ فرآیند پواسون

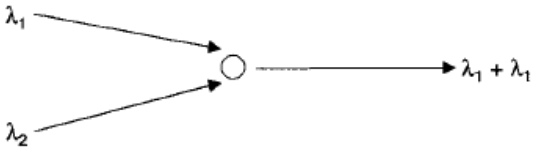
یک فرآیند احتمالی مهم استفاده‌شده در ارزیابی عملکرد سیستم کامپیوتری فرآیند پواسون است. یک فرآیند احتمالی پواسون دارای ویژگی‌هایی است که در آن رخدادها مستقل هستند، و بازه زمانی بین رخدادها می‌تواند با استفاده از توزیع نمایی $F(t) = ۱ - e^{-\lambda t}$ تشریح شود. برای مثال، رخدادهای تشریح شده می‌توانند ورودی یک تراکنش برای سرویس، تکمیل پردازش تراکنش، یا زمان سرویس‌دهی به تراکنش باشد. با توجه به اینکه میانگین زمانی بین برخی از رخدادها $۱/\lambda$ است، این است که نرخ وقوع رخدادها λ خواهد بود. فرآیند پواسون دارای ویژگی‌های زیر است:

۱. وقوع رخدادها در طول بازه‌های زمانی غیر همپوشان به‌صورت مستقل صورت می‌گیرد.
۲. برای افزایش زمانی کوچک، احتمال صفر رخداد برابر $۱ - \lambda\Delta t$ است، احتمال وقوع یک رخداد در همان زمان $\lambda\Delta t$ است.

فرآیند احتمالی پواسون دارای ویژگی‌های بسیار مطلوبی است. اگر دو جریان ورودی پواسون در هم ادغام شوند، جریان نتیجه بازهم جریان پواسون با نرخ برابر با مجموع نرخ‌های ورودی است. برای مثال، شکل (۶-۵) با $\{N_۱(t), t \geq ۰\}$ و نرخ $\lambda_۱$، $\{N_۲(t), t \geq ۰\}$ و نرخ $\lambda_۲$، جریان نتیجه، $\{N_{sum}(t), t \geq ۰\}$، نرخ $\lambda_۱ + \lambda_۲$ را دارد.

اگر یک جریان پواسون به دو جریان تقسیم شود، با هر رخداد در جریان A با احتمال P_A و جریان B با احتمال P_B، جریان نتیجه پواسون با نرخ‌های $P_A\lambda$ و $P_B\lambda$ همراه است.

بیایید بابا استفاده از ویژگی‌های پایه فرآیند پواسون و برخی از مفاهیم اصلی احتمال به چند مثال نگاهی بی‌اندازیم.در این مثال، یک مرکز کامپیوتر تعداد زیادی بخش سیستمی مجزا دارد که احتمال شکست آن‌ها وجود دارد، برای مثال، ترمینال‌ها، درایورها، دیسک ها، پرینترها، حسگرها، CPU ها و غیره. زمانی که این آیتم‌ها با شکست مواجه شوند، باعث خرابی کل سیستم کامپیوتری نمی‌شوند. می‌دانیم که برای این سیستم به‌صورت میانگین ۰/۶ خطا در هرروز وجود دارد. خطاها و خرابی‌ها مستقل هستند. این خرابی می‌توانند با فرآیند پواسون با نرخ $\lambda = ۰.۶$ (در روز) نشان داده شوند.



شکل (۶-۵): ادغام دو جریان پواسون ورودی

علاوه بر این، زمان بین شکست‌ها به‌صورت توزیع نمایی دیده شده است. میانگین زمانی بین خرابی‌ها چقدر است؟

$$P[\tau_n \le s] = ۱ - e^{-\lambda s} \quad \lambda = ۰.۶ \text{ / روز}$$

(۹-۶)

$$E[\tau_n] = ۱/\lambda = ۱.۶۶۶ \text{ روزها} = ۳۹.۹۹ \text{ ساعات بین خرابی ها}$$

با استفاده از شرایط اولیه، می‌توانیم تعداد خرابی‌ها را در یک بازه t روزه با توزیع پواسون با میانگین $۰.۶t$ ببینیم. می‌توانیم از این برای تعیین تعداد خرابی‌هایی که می‌توانیم در طول هر دوره زمانی خاص انتظار داشته باشیم استفاده کنیم. برای مثال، می‌توانیم در مورد احتمال وقوع یک خرابی در یک دوره ۲۴ ساعته سؤال بپرسیم:

$$P[y_t = k] = e^{-\lambda r}\frac{(\lambda t)^k}{k!}$$

(۱۰-۶)

$$P\left[y_t = ۱ \text{ روز}\right] = e^{-۰.۶}\frac{(۰.۶)}{۱!} = (۰.۵۴۸۸)(۰.۶) = ۰.۳۲۹۲۸$$

می‌توانیم در مورد احتمال اینکه کمتر از پنج خطا در یک هفته رخ می‌دهد یا نه، سؤال بپرسیم:

$$P\,[y_{۷} < ۵] = \sum_{k=۰}^{۴} P\,[y = k]$$

$$\sum_{k=۰}^{۴} e^{-\lambda t}\ \frac{(\lambda t)^k}{k!}$$

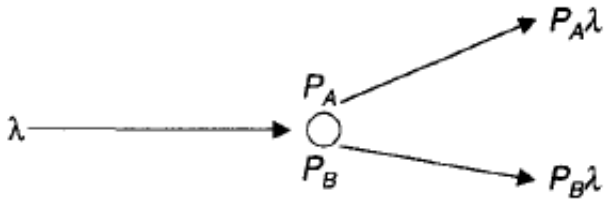
(۱۱-۶)

$$= \sum_{k=۰}^{۴} e^{(-۰.۶ \times ۷)}\ \frac{(۰.۶ \times ۷)^k}{k!}$$

$$= e^{-۴.۲}\left[\begin{array}{l} ۱ + (۴.۲ / ۱) + ((۴.۲)^{۲} / ۲!) \\ + ((۴.۲)^{۳} / ۳!) + ((۴.۲)^{۴} / ۴!) \end{array}\right]$$

$$= ۰.۵۸۹۸$$

این حاکی از این است که با احتمال ۰,۵۸۹۸ در این دوره زمانی کمتر از پنج خطا وجود دارد.



شکل (۶-۶): تقسیم جریان پواسون

در مقابل، می‌توانیم از نقطه زمان تصادفی شروع کنیم و تعیین کنیم که چقدر احتمال دارد که در ۲۴ ساعت آینده هیچ خطایی رخ ندهد:

$$P[\tau_n > ۱ \text{ day}] = P[y_۱ = ۰] = e^{-\lambda t} = e^{-۰.۶} = ۰.۵۴۸۸ \qquad (۱۲-۶)$$

با استفاده از ویژگی‌های اصلی فرآیند پواسون می‌توانیم سؤال دیگری مطرح کنیم که در سیستم کشف یا سنجیده می‌شود. برای مثال، فرض کنید که دقیقاً ۲۴ ساعت بدون هیچ خرابی سپری‌شده است. تا خرابی بعدی چقدر زمان انتظار می‌رود؟ فرآیند پواسون از ویژگی بدون حافظه بودن استفاده می‌کند – به همین ترتیب، سابقه گذشته به‌پیش بینی آینده کمک نمی‌کند (افزایش‌های مستقل). نتایج این سؤال مانند سؤال اول است، که در مورد احتمال خطای بعدی سؤال پرسیده بود.

$$P[\tau_n \le S] = ۱ - e^{-\lambda s} \qquad \lambda = ./۶ \text{ روز}$$

(۱۳-۶)

$$E[\tau_n] = ۱/\lambda = ۱.۶۶۶ \text{ روزها} = ۳۹.۹۹ \text{ ساعات بین خرابی ها}$$

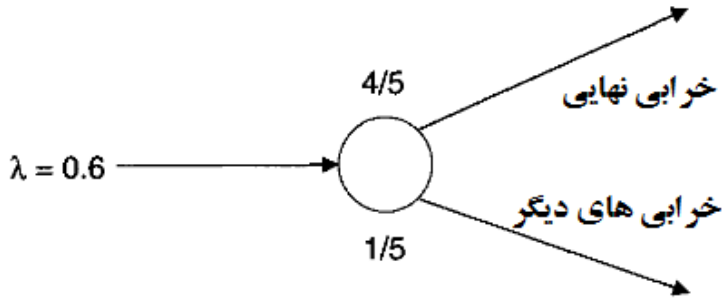
به‌عنوان‌مثال دیگر، می‌دانیم که چهار خطا از هر پنج خطا یک مشکل نهایی است، در حالی که این خرابی‌ها با احتمال برابر در هر خطا رخ می‌دهند (شکل ۶-۷). می‌خواهیم تعیین کنیم که کدام فرآیند تشریح کننده خرابی نهایی است. باید ابتدا تشخیص دهیم که این می‌تواند با فرآیند پواسون مدل شود، که در آن مجموع جریان (نمایش خرابی) می‌تواند به یک جریان مجزا شکسته شود، که هردوی آن فرآیند پواسون هستند. با توجه به این فرضیات، می‌توانیم بیان کنیم که این یک فرآیند پواسون با نرخ $P_A \lambda = (۴/۵)۰.۶ = ۰.۴۸$ روز/ برای هر خطای نهایی است.

میانگین زمانی بین خرابی‌های نهایی ۲٫۰۸۳ روز است. می‌توانیم تعداد خرابی‌های نهایی را در t روز با رابطه زیر به دست آوریم:

$$P[y = k] = e^{-\lambda t} \frac{(\lambda t)^k}{k!}$$

(۱۴-۶)

$$= e^{-.۴۸\,t}\ \frac{(.۴۸t)^k}{K!}$$



شکل (۶-۷): احتمال یک خرابی نهایی

۶-۴ فرآیند تولد-مرگ

فرآیندهای احتمالی پواسون به یک خانواده عمومی از فرآیندهای احتمالی به نام فرآیند مرگ،تولد مربوط است. فرآیند احتمالی مرگ،تولد به فضای حالت تصادفی مربوط است که در آن دامنه مقادیر از ۰، به معنی این که هیچ عضوی در جمعیت نیست، تا عدد به‌صورت بالقوه نامحدود، که نشان دهنده جمعیت با رشد ثابت است، می‌باشد. به‌صورت واقع‌بینانه‌ای ما به جمعیت با اندازه ثابت علاقه داریم که در طول افزایش جمعیت (تولد) و کاهش جمعیت (مرگ) ثابت می‌ماند.

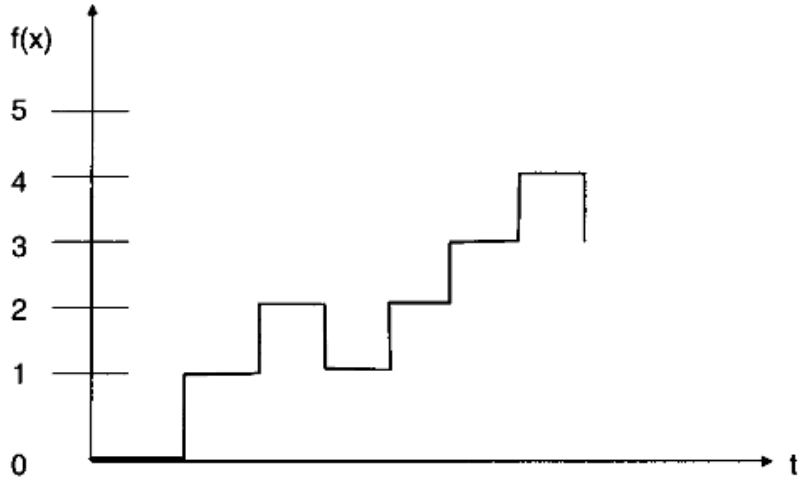
برای هر سطح خاصی (دامنه ممکن مقادیر یا عدد خاص) در جمعیت یک نرخ تولد و نرخ مرگ مربوطه وجود دارد. این نرخ ممکن است برای هر سطح ثابت باشد اما نیازی هم نیست که ثابت باشد. فرآیند احتمالی تولد،مرگ با پارامتر پیوسته (مجموعه شاخص) فرآیند احتمالی فضای حالت گسسته تشریح شده است (شکل ۶-۸).

$$\{x(t), t \geq ۰\} \qquad (۱۵-۶)$$

$E(n)$، $n = ۰, ۱, ۲, \ldots$ حالت را تشریح می‌کند و $x(t) = n$ به معنی این است که $x(t)$ در زمان t در حالت $E(n)$ است.

برای هر فرآیند احتمالی، $t \geq ۰$ $x(t)$ ، فرآیند احتمالی تولد-مرگ، فرآیند دارای یک پارامتر پیوسته از فضای حالت گسسته فرآیند احتمالی است، و باید دارای مشخصه‌های اضافی زیر باشد:

۱. تغییر حالت تنها با ۱± صورت می‌گیرد و مقدار E_n هیچ‌گاه منفی نیست.



شکل (۶-۸): مثالی از فرآیند تولد-مرگ

۲. اگر سیستم در زمان t در حالت E_n باشد، احتمال یک تراکنش E_{n+1} در طول بازه $(t, t+h)$ به شرح زیر است:

۳.

$$\lambda_n h + o(h)$$

و تا E_{n-1} برابر است با

$$\mu_n h + o(h)$$

۴. احتمال بیش از یک تراکنش در طول بازه h برابر $o(h)$ است.

اگر یک فرآیند مرگ،تولد را از هر حالت خاص دیگر، E_n، بررسی کنیم، می‌توانیم ببینیم که به حالتی از دو محل دیگر وارد می‌شویم: حالت E_{n+1} یا E_{n-1} (شکل ۶-۹).

با دانستن این موضوع می‌توانیم انواع معیارهای کارایی مهم را محاسبه کنیم. همه این معیارها از مبنای محاسباتی معامله دیفرانسیلی تفاضل به‌دست‌آمده است. این معادلات فرآیند احتمالی مرگ،تولد را بین حالات بررسی می‌کند. ما این موارد را با تمرکز بر یک گره یا حالت محاسبه می‌کنیم، مانند شکل (۶-۹).

$$P_n(t) = P[X(t) = n] \qquad (۱۶-۶)$$

احتمال اینکه سیستم در زمان t در حالت E_n باشد، است.

$P_n(t+h)$ برای P کوچک چیست؟

$$P_n(t+h) = P_n(t)\left(۱ - \left(\lambda_n h + ۰(h)\right)\right)\left(۱ - \left(\mu_n h + ۰(h)\right)\right)$$

$$+ P_{n-۱}(t)\left(\lambda_{n-۱} h + ۰(h)\right)$$

$$+ P_{n+۱}(t)\left(\mu_{n+۱} h + ۰(h)\right)$$

$$+ ۰(h) \qquad (۱۷-۶)$$

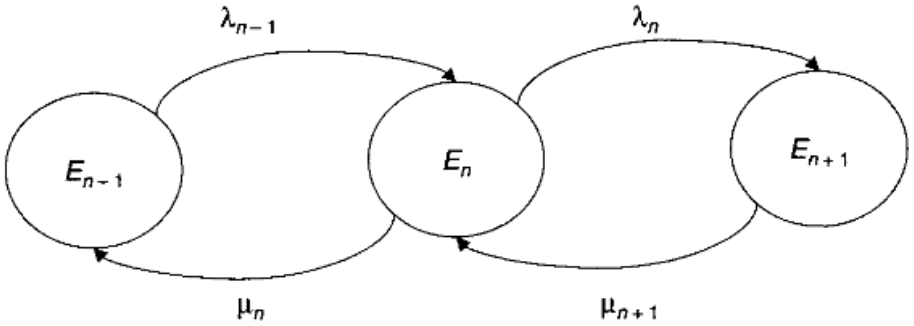
$$= [۱ - \lambda_n h - \mu_n h]\, P_n(t) + \lambda_{n-۱} h\, P_{n-۱}(t)$$

$$+ \mu_{n+۱} h\, P_{n+۱}(t) + ۰(h)$$

با جابجایی از $P_n(t)$ و تقسیم بر h داریم:

$$\frac{P_n(t+h) - P_n(t)}{h} = -(\lambda_n + \mu_n) P_n(t)$$

$$+ \lambda_{n-1} P_{n-1}(t) + \mu_{n+1} P_{n+1}(t) + \frac{o(h)}{h} \qquad (۱۸-۶)$$



شکل (۶-۹) نمونه دیاگرام انتقال حالت فرآیند احتمالی

با در نظر گرفتن حد $h \to 0$ داریم:

$$\frac{d P_n(t)}{dt} = -(\lambda_n + \mu_n)P_n(t) + \lambda_{n-1} P_{n-1}(t) + \mu_{n+1} P_{n+1}(t) \text{ for } n \geq 1$$

$$\frac{d P_0(t)}{dt} = -\lambda_0 P_0(t) + \mu_1 P_1(t) \text{ for } n = 0 \qquad (۱۹-۶)$$

معادله (۶-۱۹) رابطه حالت آغازین با حالت اولیه و نرخ مرگ و میر اولیه را می‌دهد. ما اهمیت این ویژگی ساده فرآیند احتمالی مرگ،تولد با رشد این فرآیند احتمالی و استفاده از آن برای تحلیل سیستم کامپیوتری به‌عنوان صف‌های ساده و شبکه‌ای از صف‌ها خواهیم دید.

مثال خاص ما از فرآیند تولد،مرگ مانند شرایطی است که تنها تولد وجود دارد و هیچ مرگی وجود ندارد، و نرخ تولد مستقل از حالت و ثابت است. به‌صورت ویژه تر، ما فرضیات زیر را داریم:

۱. نرخ تولد با میانگین نرخ $\lambda_n = \lambda > ۰$ وجود دارد.

۲. هیچ مرگی وجود ندارد، بنابراین نرخ مرگ برابر است با $\mu_n = ۰$.

با استفاده از این اطلاعات و تحلیل تولد،مرگ پایه که قبلاً تشریح کردیم، می‌توانیم نشان دهیم که:

$$\frac{d\,P_n(t)}{dt} = -\lambda P_n(t) + \lambda P_{n-۱}(t) \text{ for } n \geq ۱$$

$$\frac{d\,P_۰(t)}{dt} = -\lambda P_۰(t) \text{ for } n = ۰ \qquad (۲۰-۶)$$

احتمال بودن در هر حالت برابر زیر است:

$$P_n(t) = e^{-\lambda t}\ \frac{(\lambda t)^n}{n!}\ , n \geq ۰ \text{ and } t \geq ۰ \qquad (۲۱-۶)$$

که نشان می‌دهد که این یک فرآیند پواسون است. فرآیند پواسون می‌تواند به‌عنوان بخشی از فرآیند تولد با نرخ تولد ثابت مدل شود.

مورد عمومی فرآیند تولد،مرگ در زمانی که راه حل‌های وابستگی زمانی داشته باشند کمی پیچیده تر است. بااین‌حال، اگر به نقطه‌ای نگاه کنیم که سیستم نزدیک به مقادیر محدود کننده باشد، سپس سیستم می‌تواند ثابت فرض شود، و بنابراین راه حل‌های تعادلی برای سیستم وجود دارد. در این تعادل یا راه حل‌هایی با حالت پایدار، موارد زیر را فرض می‌کنیم:

$$\lim_{t\to\infty} \frac{d\,P_n(t)}{dt} = ۰ \text{ for each } n \qquad (۲۲-۶)$$

و

$$\lim_{t\to\infty} P_n(t) = P_n \text{for each } n \qquad (۶-۲۳)$$

می‌توانیم بر حالات گوناگون تمرکز کنیم و معادله دیفرانسیلی تفاضل از یک گره عمومی $(n \geq ۱)$ برای حالت اولیه $n = ۰$ تمرکز می‌کنیم، بنابراین داریم:

$$۰ = \lambda_{n-۱}\ P_{n-۱} + \mu_{n+۱}\ P_{n+۱} - (\lambda_n + \mu_n)\ P_n,\ n \geq ۱ \qquad (۶-۲۴)$$

$$۰ = \mu_۱ P_۱ - \lambda_۰ P_۰ \qquad (۶-۲۵)$$

راه حل برای این معادله دیفرانسیلی تفاضلی، با استفاده از جبر، به شکل زیر نشان داده می‌شود:

$$P_{n+۱} = \frac{\lambda_۰}{\mu_۱} P_n, \qquad n \geq ۱$$

$$\therefore\ P_۱ = \frac{\lambda_۰}{\mu_۱} P_۰$$

$$P_۲ = \frac{\lambda_۱}{\lambda_۲} P_۱ = \frac{\lambda_۰ \lambda_۱}{\mu_۱ \mu_۲} P_۰ \qquad (۶-۲۶)$$

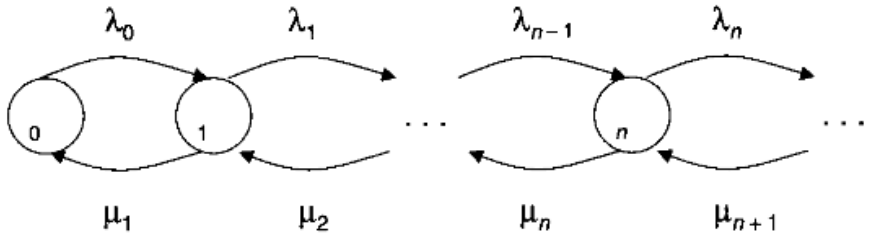
$$\vdots$$

$$P_n = \frac{\lambda_۰ \lambda_۱ \lambda_۲ \dots \lambda_{n-۱}}{\mu_۱ \mu_۲ \mu_۳ \dots \mu_۴} P_۰,\ n \geq ۱$$

و

$$\sum_{n=۰}^{\infty} P_n = ۱$$

راه حل تشریح شده در اینجا بر استفاده از معادلات تعادل برای حل احتمال حالت گوناگون تمرکز می‌کند. معادلات تعادل می‌توانند استفاده شود، ازآنجایی‌که فرض می‌کنیم که سیستم به تعادل رسیده است، بنابراین بین حالات پایدار جا به جا می‌شود، و اهمیتی ندارد که در کدام حالت پایدار باشد.



شکل (۶-۱۰): نمایش گرافیکی برای فرآیند تولد - مرگ

معادلات تعادل هر حالت، E_n را بررسی می‌کند، زمانی که تعادل حاصل می‌شود، نرخ انتقال به حالت E_n و نرخ انتقال به خارج از حالت E_n به‌صورت زیر محاسبه می‌شود :

نرخ خروجی E_n = نرخ ورودی E_n

از فرآیند تولد-مرگ داریم:

$$۱- \lambda_{n-۱} P_{n-۱} + \mu_{n+۱} P_{n+۱} = (\lambda_n + \mu_n) P_n, \quad n \geq ۱ \qquad (۶-۲۷)$$

$$۲- \mu_۱ P_۱ = \lambda_۰ P_۰ \qquad (۶-۲۸)$$

همچنین

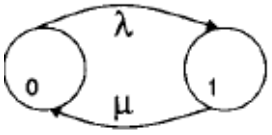
$$\sum_{n=۰}^{\infty} P_n = ۱ \qquad (۶-۲۹)$$

یک نمایش گرافیکی برای فرآیند تولد،مرگ در شکل (۶-۱۰) نشان داده‌شده است.
نمودار انتقال نرخ یک حالت تحلیل برای یک سرور واحد بدون خط انتظار در شکل (۶-۱۱) نشان داده‌شده است. مثال دارای زمان ورود پواسون با نرخ λ و خدمات نمایی با نرخ μ است. معادلات تعادل برای این مثال به شرح زیر است:

$$\mu P_1 = \lambda P_0$$

و

$$P_1 + P_0 = 1 \qquad (۳۰-۶)$$



شکل (۶-۱۱) نمودار نرخ انتقال

با حل معادلات تعادل داریم:

$$P_1 = (\lambda/\mu)P_0 \text{ and } p_0 + (\lambda/\mu)P - 0 = 1 \qquad (۳۱-۶)$$

بنابراین داریم:

$$P_0 = \frac{\mu}{\lambda+\mu} \qquad (۳۲-۶)$$

اهمیت این بررسی اولیه فرآیند احتمالی مرگ و تولد، نمایش این فرآیند با استفاده از دیاگرام‌های نرخ انتقال، و فرض تکنیک تعادل و حل با استفاده از تعادل با آغاز نگاه به نمایش عمومی و نگاشت به سیستم کامپیوتری به نظر معقول تر است.

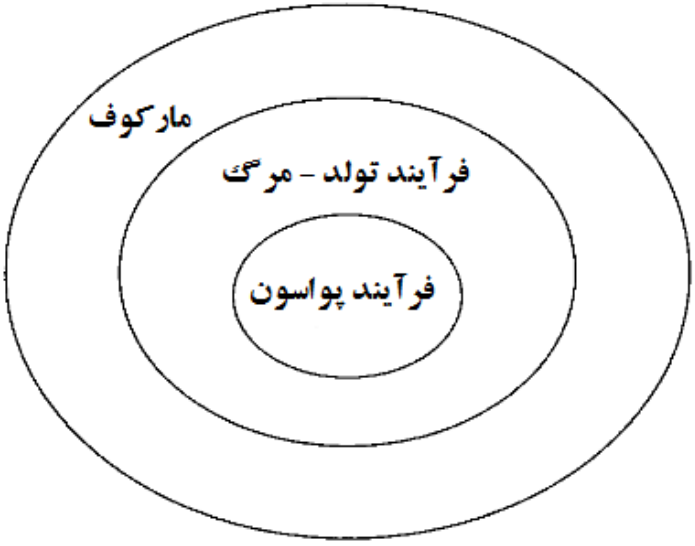
۶-۵ فرآیند مارکوف

یک فرآیند مارکوف یک فرآیند احتمالی با مشخصه‌های اضافی است. اگر احتمال حالت آینده یک فرآیند احتمالی فقط به احتمال حالت فعلی بستگی داشته باشد و به چگونگی رسیدن به آن حالت در دسترس نباشد، سپس این یک فرآیند مارکوف است.

به طور رسمی تر، یک فرایند پیچیده $\{X(t), t \in T\}$ یک فرآیند مارکوف است اگر برای هر مجموعه $n+1$ از مقادیر $t_1 < t_2 < \cdots < t_n < t_{n+1}$ در مجموعه شاخص در هر مجموعه حالات دیگر $\{x_1, x_2, \ldots, x_n, x_{n+1}\}$ روابط زیر را داشته باشیم:

$$P\left[X\left(t_{n+1}\right) = x_{n+1} \mid X\left(t_1\right) = x_1, X\left(t_2\right) = x_2, \ldots X\left(t_n\right) = x_n\right] \quad (۳۳-۶)$$

$$= P\left[X\left(t_{n+1}\right) = x_{n+1} \mid X\left(t_n\right) = x_n\right]$$



شکل (۶-۱۲): نگاشت فرآیند مارکوف به دیگر فرآیند احتمالی

همه فرآیندهای تولد،مرگ فرآیندهای مارکوف هستند، فرآیند پواسون نیز یک فرآیند مارکوف است.(شکل ۶-۱۲).

یک فرآیند مارکوف با حالات گسسته را زنجیره مارکوف گویند، زنجیره‌های مارکوف شامل حالات گسسته $\{E_0, E_1, E_2, \dots\}$ است. این حالتها معمولاً با استفاده از اعداد صحیح غیر منفی یا غیر انتزاعی $\{0,1,2,3,4 \text{ و} \dots\}$ به‌جای توصیف رسمی که قبلاً ارائه شده است، توصیف می‌شوند. یک زنجیره مارکوف گسسته زمانی حالات انتقالی را در زمان t_n ، $n = 1,2,3, \dots$ ، (احتمالاً در یک حالت) مشخص می‌کند. نشانه‌گذاری این انتقال و حالت نتیجه به شکل زیر است:

$$\{X(t_n), t_n = 0,1,2, \dots\} \rightarrow \{X_n\} \qquad (۶-۳۴)$$

ما معمولاً تنها به زنجیره‌های مارکوفی علاقه داریم که دارای احتمال حالت انتقال ثابت هستند:

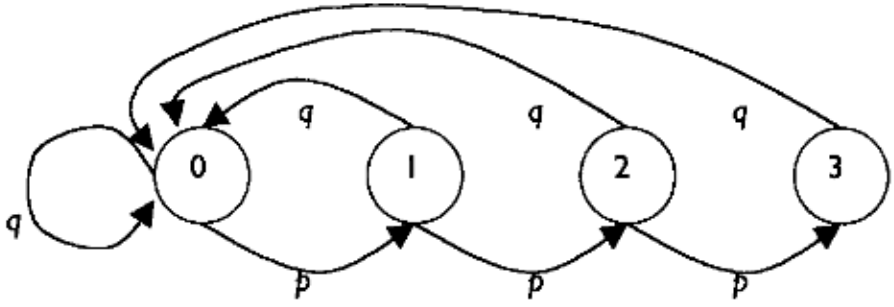
$$P[X_{n+1} = j \mid X_n = i] = P[X_{m+1} = j \mid X_m = i]$$

$$P_{ij} \quad \forall \; m, n, i, j \qquad (۳۵-۶)$$

احتمال انتقال حالت، P_{ij} ، احتمال انتقال از حالت i به حالت j را نشان می‌دهد. برای یک زنجیره مارکوف کامل، ما مجموعه‌ای از این احتمال انتقالی را به‌عنوان یک ماتریس انتقال حالت P، مانند شکل (۶-۱۳)، نشان می‌دهیم.

$$P = \begin{bmatrix} P_{۰۰} & P_{۰۱} & P_{۰۲} & \cdots & P_{۰j} & \cdots \\ P_{۱۰} & P_{۱۱} & P_{۱۲} & \cdots & P_{۱j} & \cdots \\ P_{۲۰} & P_{۲۱} & P_{۲۲} & \cdots & P_{۲j} & \cdots \\ P_{۳۰} & P_{۳۱} & P_{۳۲} & \cdots & P_{۳j} & \cdots \\ \vdots & \vdots & \vdots & & \vdots & \\ P_{i۰} & P_{i۱} & P_{۲i} & \cdots & P_{ij} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

شکل (۶-۱۳) ماتریس انتقال حالت احتمالی نمونه



شکل (۶-۱۴) نمودار انتقال حالت

شرایط ورودی‌های ماتریس احتمال انتقال حالت به شکل زیر است:

$$P_{ij} \geq ۰ \quad i,j = ۰,۱,۲,\ldots \qquad (۳۶-۶)$$

$$\sum_{j=۰}^{\infty} P_{ij} = ۱ \quad i = ۱,۲,۳,\ldots \qquad (۳۷-۶)$$

یک مثال با استفاده از این ماتریسی شامل توالی از آزمایش‌های برنولی است. در یک آزمایش برنولی تنها موفقیت یا شکست وجود دارد. یک آزمایش با احتمال p موفق می‌شود و با احتمال $q = ۱ - p$ شکست می‌خورد. در این مثال، فرض می کنیم که حالت در آزمایش n، با مقدار X_n، تعداد موفقیت‌های پی‌درپی است (طول موفقیت متوالی)، در مثال فرض کنید که آزمایش زیر رخ می‌دهد. مقادیر برای فضای نمونه به شکل زیر است، شاخص n و X_n، به شکل زیر هستند:

آزمایش:	F	S	S	F	F	S	S	S	F
$n =$	۰	۱	۲	۳	۴	۵	۶	۷	۸
$X_n =$	۰	۱	۲	۰	۰	۱	۲	۳	۰

نمودار انتقال حالت در شکل (۶-۱۴) نشان داده‌شده است.

ماتریس احتمالی انتقال حالت نتیجه از عناصر زیر تشکیل شده است، که در شکل (۶-۱۵) آورده شده است.

$$P = (P_{ij}) = \begin{bmatrix} q_{۰۰} & P_{۰۱} & ۰ & ۰ \\ q_{۱۰} & ۰ & P_{۱۲} & ۰ \\ q_{۲۰} & ۰ & ۰ & P_{۲۳} \\ q_{۳۰} & ۰ & ۰ & ۰ \end{bmatrix}$$

شکل (۶-۱۵) ماتریس احتمال انتقال

با استفاده از این احتمالات حالت می‌خواهیم که احتمال حالت را برای کل گراف محاسبه کنیم، فرض می‌کنیم از قبل تعادل وجود دارد.

اگر فرض کنیم $\pi_i^{(n)}$ احتمال بودن در حالت j را بعد از n امین مرحله نشان می‌دهد داریم:

$$\pi_i^{(n)} = P\,[X_n = j] \qquad (۳۸-۶)$$

پس:

$$\pi_j^{(n+۱)} = \sum_{i=۰}^{\infty} \pi_i^{(n)}\, P_{ij}^{(n)}, \forall j \qquad (۳۹-۶)$$

برای حالات $i = ۰, ۱, ۲, \dots, n - ۱$

$$\pi_i^{(n+۱)} = \sum_{i=۰}^{n-۱} \pi_i^{(n)}\, P_{ij}^{(n)}, \forall j \qquad (۴۰-۶)$$

$P^{(n)} = \left[P_{ij}^{(n)}\right], N \times N$ ماتریس :

$$\pi^{(n)} = (\pi_۰^{(n)}, \pi_۱^{(n)}, \dots, \pi_{n-۱}^{(n)}) \qquad (۴۱-۶)$$

سپس در بردار داریم:

$$\prod_{\sim} n + ۱ = \prod_{\sim}(n).\, P^{(n)} \qquad (۴۲-۶)$$

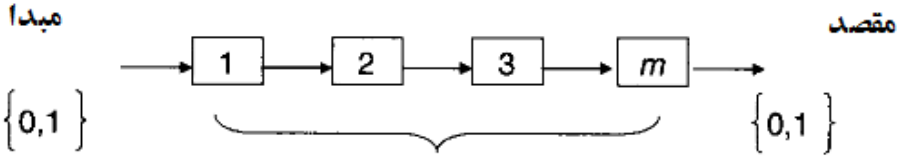
احتمال انتقال ثابت (همگن)

$$P^{(n)} = P^{(m)}, \forall n, m$$

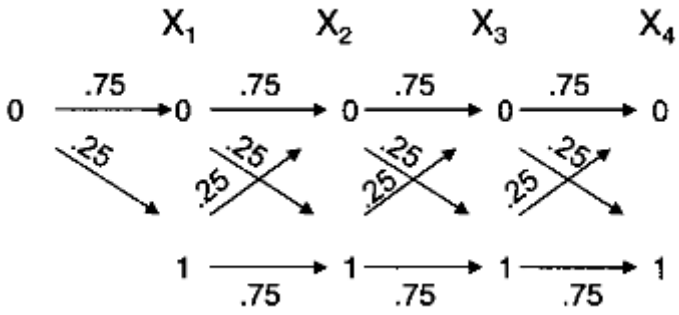
$$= P \qquad (۴۳-۶)$$

$$\prod_{\sim}(n+۱) = \prod_{\sim}(n) \; . P$$

یک مثال، فرض کنید که یک سیستم ارتباطی داریم که اعداد ۰ و ۱ را از طریق چند مرحله انتقال می‌دهد (شکل ۶-۱۶) و فرض می کنیم که در هر مرحله یک احتمال ۰,۷۵ وجود دارد که خروجی همان ورودی باشد.



شکل (۶-۱۶) مراحل سیستم ارتباطی



شکل (۶-۱۷): احتمال انتقال برای سیستم‌های ارتباطی شکل (۶-۱۶)

یک سؤال که ممکن است پیش آید این است که با چه احتمالی ۰ وارد شده در اولین مرحله همان خروجی ۰ از مرحله چهارم است. راه حل نیازمند نمایش مسئله به‌عنوان زنجیره مارکوف و احتمال یک راه حل ماتریسی است:

فرض کنید، حالت در مرحله n، X_n، به مقدار خروجی n امین مرحله اشاره کند.

فرض کنید ۰ ورودی به مرحله ۱ است که در شکل (۶-۱۷) نشان داده‌شده است.

احتمال $X_۴ = ۰$ چقدر است؟

$$\prod_{\sim}(n+۱) = \prod_{\sim}(n)\, P \qquad (۴۴-۶)$$

هرگاه:

$$\prod_{\sim}(۰) = (۱,۰)$$

$$\prod_{\sim}(۱) = \prod_{\sim}(۰)P$$

$$\prod_{\sim}(۲) = \prod_{\sim}(۱) = \prod_{\sim}(۰)P_۲ \qquad (۴۵-۶)$$

$$\prod_{\sim}(۳) = \prod_{\sim}(۲)P = \prod_{\sim}(۰)P_۳$$

$$\prod_{\sim}(۴) = \prod_{\sim}(۳)P = \prod_{\sim}(۰)P_۴$$

با توجه به ماتریس احتمال انتقال حالت داریم:

$$P = \begin{bmatrix} ۰.۷۵ & ۰.۲۶ \\ ۰.۲۵ & ۰.۷۵ \end{bmatrix}$$

$$P^۲ = \begin{bmatrix} ۰.۶۲۵ & ۰.۳۷۵ \\ ۰.۳۷۵ & ۰.۶۲۵ \end{bmatrix}$$

$$P^{۴} = \begin{bmatrix} ۰.۵۳۱۲۵ & ۰.۳۷۵ \\ ۰.۴۸۸۷۵ & ۰.۵۳۱۲۵ \end{bmatrix}$$

$$\therefore \left(\prod_{۰}(۴), \prod_{۱}(۴) \right) = (۱,۰)P^{۴}$$

$$= (۰.۵۳۱۲۵ , ۰.۴۶۸۷۵)$$

راه حل عمومی با استفاده از زنجیره‌های مارکوف ثابت عبارت زیر را حاصل می‌کند:
$\prod(n) = \prod(۰)\, P^{n}$ ، n مرحله ماتریس احتمال انتقال است.

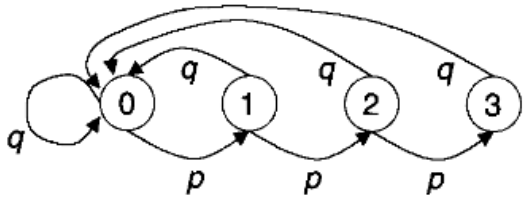
۶-۵-۱ تعریف زنجیره مارکوف

حالت j از حالت i قابل دسترسی است اگر برای زنجیره دنبال کردن حالت i به j در تعداد متناهی حالت امکان پذیر باشد، داریم:

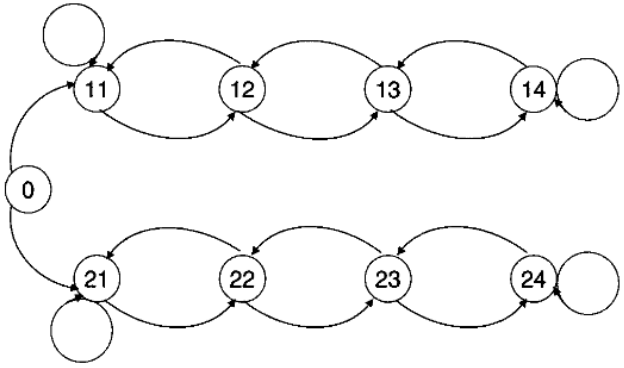
$$P_{ij}^{n} > ۰ \text{ , for some } n > ۰ \quad (۶-۴۶)$$

اگر هر حالت از هر حالت دیگری قابل دسترسی باشد، زنجیره غیر قابل تقلیل است. با استفاده از آزمایش پرتاب سکه برنولی مانند قبل، ما نمودار حالت انتقال را در شکل (۶-۱۸) رسم کردیم.
در شکل (۶-۱۸)، می‌توانیم ببینیم که اگر در هریک از حالات باشیم، می‌توانیم در طی چند مرحله به برخی حالات دیگر برسیم. برای مثال، اگر در حالت ۳ باشیم، می‌توانیم با انتقال از طریق کمان ۰،۳ به حالت صفر برسیم. می‌توانیم با یال ۱،۰ به حالت یک برسیم و سپس با انتقال از کمان ۲،۱ به حالت ۲ برسیم. نکته جالب توجه دیگر این است که اگر در حالت ۰ باشیم، می‌توانی با کمان ۰،۰ دوباره به حالت صفر برگردیم. بعد از بررسی همه مسیرها از همه جفت‌ها می‌توانیم ببینیم که زنجیره مارکوف غیر قابل تقلیل است.
در مثال دوم (شکل ۶-۱۹)، این گراف قابل تقلیل است، چرا که حداقل یک مسیر (کمان ۰،۱۱) وجود دارد که اجازه نمی‌دهد که عناصر یک زیر زنجیره (شامل گره‌های ۱۱،۱۲،۱۳ و ۱۴) به زیر زنجیره ۲۱،۲۲،۲۳ و ۲۴ وصل شود.

ما معمولاً به رفتار فرآیندی علاقه داریم که می‌تواند با زنجیره‌های غیر قابل تقلیل نشان داده شود، چرا که آن‌ها ساده‌تر حل می‌شوند و تعادل می‌تواند در این سیستم حاصل شود یا فرض شود.



شکل (۶-۱۸): نمودار حالت انتقال (آزمایش برنولی، شیر یا خط سکه)



شکل (۶-۱۹): نمودار انتقال غیر قابل تقلیل

خصوصیات جالب دیگر در زنجیره‌های مارکوف مفهوم زنجیره ergotic است. گفته می‌شود یک زنجیره مارکوف زمان گسسته ergotic است اگر (۱) بتوانید از هر حالتی به حالت دیگر بروید (برای مثال غیر قابل تقلیل)، (۲) برای هریک از این حالات با مسیرهایی با طول‌های گوناگون به این حالت برگردد، (۳) با ترک یک حالت با احتمال ۱ درون یک میانگین زمانی محدود بازمی‌گردد (بازگشت مثبت). این آخرین ویژگی حاکی از این است که، یک مسیر باید وجود داشته باشد که همه کمان‌ها را ملاقات کند.

یک زنجیره مارکوف دارای یک توزیع ثابت است اگر :

$$\prod_{\sim} = (\prod_{\cdot}, \prod_{1}, \prod_{2}, \dots, \prod_{n-1}) \qquad (۶-۴۷)$$

n حالت

و اگر یک بردار $\prod_{\sim}$ وجود داشته باشد به‌طوری‌که روابط زیر برقرار باشد:

$$\prod_{\sim} = \prod_{\sim} P \qquad (۴۸-۶)$$

با

$$\prod_i \geq ۰ \; \forall_i \qquad (۴۹-۶)$$

و

$$\sum \prod_i = ۱ \qquad (۵۰-۶)$$

به طور معادل:

$$\lim_{n\to\infty} \prod_j(n) = \prod_j, \text{for } j = ۰, ۱, \ldots \; (۶.۵۱) \qquad (۵۱-۶)$$

برای یک زنجیره مارکوف ergotic، حد زیر را داریم:

$$\prod_{\sim} \lim_{n\to\infty} \prod_{\sim}(n) = \lim_{n\to\infty} \prod_{\sim}(۰) \, P^n \qquad (۵۲-۶)$$

همیشه برقرار است و یک توزیع احتمال ثابت را شکل می‌دهد که از حالت اولیه مستقل است:

$$\prod_{\sim}(۰) \qquad (۵۳-۶)$$

توزیع محدود کردن یک راه‌حل مؤثر برای معادلات است.

۱- معادله تعادل

$$\prod_{\sim} = \prod_{\sim} P \qquad (۵۴-۶)$$

۲- مجموع احتمالات

$$\sum_{j} \prod_{j} = ۱$$

علاوه بر این، برای هر حالت

$$\prod_{j} = ۱/m_i \qquad (۵۵-۶)$$

در این رابطه m_i متوسط زمان بازگشت حالت i است، میانگین تعداد مراحل برای بازگشت به حالت بعد از ترک حالت، در نظر گرفته می‌شود.

مثال: سیستم ارتباطی

با چه احتمالی یک ۰ که به اولین مرحله واردشده است به‌عنوان ۰ از مرحله n ام با $n \to \infty$ خارج می‌شود (شکل ۶-۲۰) ؟

۱. معادله تعادل

$$\prod_{\sim} = \prod_{\sim} P$$

$$= \prod \begin{bmatrix} ۰.۷۵ & ۰.۲۵ \\ ۰.۲۵ & ۰.۷۵ \end{bmatrix}$$

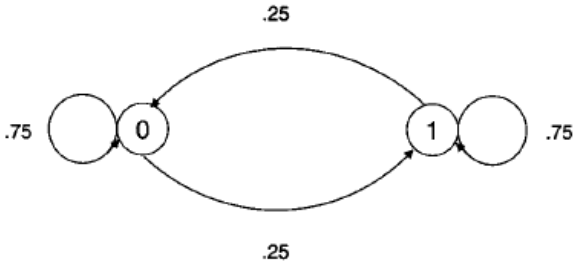
نرخ ورود= نرخ خروج

$$\prod_{۰} \times ۰.۷۵ + \prod_{۰} \times ۰.۲۵ = \prod_{۰} \times ۰.۲۵ + \prod_{۰} \times ۰.۷۵$$

$$\Pi_{۰} = \Pi_{۱} \qquad (۵۶-۶)$$

۲. مجموع احتمالات

$$\prod_{۰} + \prod_{۱} = ۱$$



شکل (۶-۲۰): نمودار حالت

ازاین‌رو داریم

$$\prod_{۰} = ۰.۵ \text{ and } \prod_{۱} = ۰.۵$$

$$\prod_{\sim} = (۰.۵ , ۰.۵)$$

که یک توزیع ثابت است.

۶-۶ خلاصه

در این فصل، برخی از مفاهیم پایه مربوط به متغیرهای تصادفی و فرآیندهای احتمالی را بیان کردیم. این نشان می‌دهد که فرآیندهای احتمالی دارای برخی از خصوصیات اصلی هستند که به آن‌ها اجازه می‌دهد که به سهولت برای مطالعه سیستم‌های کامپیوتری استفاده شوند. یکی از این مفاهیم فرآیند پواسون و کاربردهای آن برای مفهوم نرخ ورود مورد انتظار یا نرخ‌های خدمات برای رخدادهایی درون فرآیندهای احتمالی است. یک فرآیند احتمالی خاص فرآیند تولد،مرگ است. این فرآیند برای توسعه مفهوم حالات تعادل و معادلات تعادل استفاده می‌شود. این برای تعیین احتمالات حالت استفاده می‌شود. اصلاحات بیشتر فرآیند تولد،مرگ زنجیره مارکوف است. زنجیره مارکوف ویژگی‌های اضافی دارند که آن را برای مدل‌سازی برنامه‌های کاربردی سیستم‌های کامپیوتری مناسب می‌سازد.

فصل ۷

نظریه صف

در این فصل، بر اساس نظریه احتمالی پایه پوشش داده‌شده در فصل ۵ و فرآیندهای احتمالی پوشش داده‌شده در فصل ۶ عمل می‌کنیم. بحث‌ها به تعریف و تحلیل از چندین مدل صف سودمند برای رفتار بسیاری از سیستم‌های خدمات منجر خواهد شد. روش‌های بحث شده موارد ارائه شده توسط تحلیل شبیه‌سازی را تکمیل می‌کند. غالبا، رشد یک مدل صف عمومی برای یک سیستم خاص به رشد مدل شبکه پتری یا یک شبیه‌سازی دقیق بخش‌های خاص سیستم کمک می‌کند. این، نتایج و رفتارهای مشاهده‌شده از شبیه‌سازی به تنظیم مدل‌های تحلیلی کمک می‌کند.

این فصل در سه موضوع سازمان‌یافته است: مدل‌های صف، روش‌های تخمین و محاسباتی برای تحلیل سیستم‌های نظری. فرآیندهای احتمالی مبنایی را برای بسیاری از تکنیک‌های تحلیلی تشکیل می‌دهند که برای اعمال به سیستم‌های صفی که بر آن‌ها بحث خواهیم کرد استفاده می‌شوند. بخش مربوط به تخمین

روش‌هایی را برای تعریف مقادیری که مدل‌های صف را با داده جهان واقعی پارامتر بندی می‌کند، ارائه می‌دهد.

۷-۱ سیستم صف

در این بخش، ما تکنیک تحلیل پایه مربوط به سیستم‌های صف را پوشش دادیم. انگیزه نخست برای اجرای تحلیل صف ارزیابی رفتار سیستم محلی تحت انواع فرضیات، شرایط اولیه، سناریوهای عملیاتی است. جنبه مدل‌سازی به دنبال نمایش رفتار بخش‌هایی از سیستم به‌عنوان فرآیندهایی است که دارای آمار قابل‌ملاحظه و منعکس‌کننده واقعیت هستند. بنابراین، استفاده از تحلیل صف به ما مجموعه‌ای از تکنیک‌ها برای محاسبات مقادیر، مانند زمان انتظار برای خدمات، توان عملیاتی سرور، و اثر سرورهای متفاوت یا استراتژی‌های صف، تأثیر بر شبکه‌های بسته‌شده صف‌ها، ارائه می‌دهد.

فرض این است که ما باید مزیت این تکنیک را در سیستم تحت مشاهده می‌توانند که توسط یک سیستم صف نشان داده می‌شود، را اتخاذ کنیم. در باقی این فصل، ابتدا به مدل‌سازی تحلیلی به طور کل، در مشخصه سیستم‌هایی که به مدل‌سازی مربوط هستند نگاهی می‌اندازیم، و سپس مناسب بودن مدل‌های صف را در کل و استفاده خاص از آن‌ها را بررسی می‌کنیم.

در زمان تنظیم مدل یک سیستم، چه مواردی را مقداردهی می‌کنیم؟ پاسخ می‌تواند تنها یک کلمه باشد: عملکرد. این یک کلمه، ممکن است برای افراد متفاوت معانی متفاوت داشته باشد. برای نمونه، عملکرد اتومبیل را در نظر بگیرید. برای علاقمندان سرعت، عملکرد این است که یک اتومبیل تا چقدر سریع می‌تواند برود و چقدر طول می‌کشد که به این سرعت برسد. برای راننده جاده، عملکرد به معنی توانایی حرکت بدون سختی تحت شرایط خاص است. برای اقتصاددانان، عملکرد بالا به معنی بهره وری سوخت و پایین نگه داشتن هزینه‌های تعمیر و نگه داری است. و به همین ترتیب، حال عملکرد را برای یک سیستم کامپیوتری در نظر می‌گیریم. مسئله در اینجا سنجش عملکرد است، مانند استفاده از مؤلفه‌های

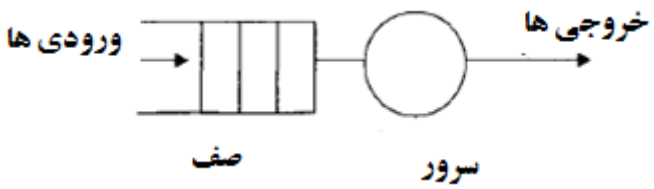
سیستم، توان عملیاتی مؤثر، میانگین زمان انتظار برای کاربر بالقوه، میانگین تعداد کاربران در سیستم در هر زمان مشخص، و توانایی سرویس‌دهی منابع.

علاوه بر این، تحلیل موازنه و مطالعه "چه می‌شود اگر" می‌تواند برای ایجاد سنجش عملکرد مانند افزایش سرعت و بهبود قابلیت دسترسی اجرا شود. در کل، این مطالعاتی توانایی تحلیل حساسیت معیارهای ذکر شده قبلی را برای تغییر در سیستم تحت مطالعه فراهم می‌کند.

فرآیند عمومی مدل‌سازی تحلیلی شامل نگاشت رفتار یک سیستم پیچیده در یک سیستم نسبتاً ساده، حل سیستم ساده‌تر برای معیارهای بهره، و سپس استقرای نتایج در سیستم پیچیده است. گاهی این فرآیند سطوح متعددی دارد، که در آن مدل‌ها به زیر مدل‌ها تقسیم می‌شوند. در اینجا، سطح پایین‌ترین مدل‌ها ابتدا حل می‌شوند (یا تا حدی حل می‌شوند)، نتایج آن‌ها به لایه بالاتر برای شامل شدن راه‌حل کامل‌تر منتشر می‌شود، و همین‌طور تا لایه بالا ادامه می‌یابد.

در برخی از موارد، بخش‌های یک مدل می‌تواند با یک تکنیک به نام تجزیه، یا ایزوله سازی جایگزین شود. در اینجا، یک زیرسیستم صف با یک سرور با جریان معادل جایگزین می‌شود، که در آن خروجی سرور برای هر تعداد واحد (یا مشتری) در سیستم از پیش محاسبه می‌شود. بنابراین، جریان کار از طریق سرور جریان معادل می‌تواند با استفاده از یک جدول جستجوی ساده شاخص بندی شده با تعداد مشتریانی که در حال حاضر در سیستم وجود دارد اجرا شوند. این تکنیک مناسب است اگر تأثیر زیرسیستم حذف‌شده در مقایسه با تأثیر دیگر زیرسیستم‌های مدل حداقل باشد.

فرض اصلی پشت استفاده از این مدل‌های صف برای تحلیل سیستم‌های کامپیوتری این است که مؤلفه‌های یک سیستم کامپیوتری می‌توانند با یک شبکه از سرورها (یا منابع) یا خطوط انتظار نمایش داده شوند.



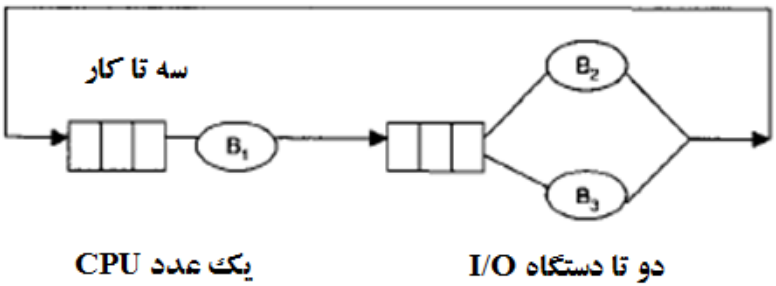
شکل (۷-۱): مدل سرور منفرد

یک سرور به‌عنوان نهادی تعریف می‌شود که می‌تواند، جریان‌های کار سیستم را متوقف کند یا بر آن‌ها تأثیر گذارد. در یک سیستم کامپیوتری، یک سرور ممکن است CPU، کانال I/O، حافظه، یا یک پورت ارتباطی باشد. یک خط انتظار محلی است که کارها برای سرویس گرفتن صف می‌شوند. برای به‌کارگیری مدل صف، کارها (یا مشتریان یا بسته پیام یا هر چیز دیگری نیازمند این است که برای سرویس گرفتن در صف قرار گیرد) در شبکه درج می‌شود. یک مثال ساده، مدل سرور واحد است، که در شکل (۷-۱) نشان داده‌شده است. در سیستم،کارها با یک نرخ وارد می‌شوند، و برای سرویس بر مبنای اولین ورود اولین خدمات سرویس داده می‌شوند، سرویس دریافت می‌کنند، و سیستم خاتمه می‌یابد. این نوع مدل، باکارهای واردشده به سیستم و خارج‌شده از سیستم، مدل سیستم صف بازگویند. با مدل‌های صف بندی ساده آبشاری وجود سرورهای موازی، شبکه‌های صف‌ها و سرورها ممکن است تشکیل شود. این ترکیبات را به‌صورت رسمی شبکه‌های صف گویند، اگرچه ما آن‌ها را مدل‌های شبکه و سیستم صف گوییم. شکل (۷-۲) یک این مدلی از یک سیستم کامپیوتری با تعداد ثابت کار که برای یک CPU و دو پردازنده I/O رقابت می‌کنند را نشان می‌دهد.

در شکل (۷-۲)، کارهایی که سرویس I/O را به اتمام رسانده‌اند برای چرخه دیگر محاسبات و I/O به‌صف CPU برمی‌گردند. به این سیستم، در جایی که تعداد مشتریان ثابت است، یک مدل سیستم شبکه با صف بسته گویند.

یک ترکیب از مفاهیم باز و بسته اساساً امکان‌پذیر است، اگر هر کاری دارای کلاس مربوطه باشد. برای مثال، یک سیستم کامپیوتری ممکن است شامل دو کلاس کاری، تراکنشی و سیستم باشد، که در آن کارهای تراکنشی می‌آیند و با ورود و خروج کاربر به سیستم کارها به‌صورت پیوسته اجرا می‌شوند. یک سیستم که شامل مشتریان از کلاس بسته و باز است را ترکیبی گویند.

مفهوم کلاس‌های مشتری اجازه می‌دهد که کلاس‌های متفاوت اصلاحات متفاوتی را در یک سرور دریافت کنند، و یک گروه از مشتریان را به‌صورت باز یا بسته تعریف کنند. یک سیستم با بیش از یک کلاس مشتری را چند کلاسی گویند، و ممکن است باز، بسته یا ترکیبی باشد.



شکل (۷-۲): مدل شبکه صف

زمانی که یک مدل شبکه ایجاد می‌شود، مجموعه‌ای از n_1 مشتری در سرور ۱، n_2 در سرور ۲، و غیره برای کل مجموعه صف در سیستم شبکه، حالت مدل شبکه را تعریف می‌کنند. یک مدل تحلیلی برای یک شبکه صف روشی را برای محاسبه این احتمال که شبکه در یک حالت خاص است فراهم می‌کند (برای مثال تعداد مشتریان در سطح اصلی برای هر صف و مرکز سرویس). علاوه بر این، توان عملیاتی شبکه، میانگین طول صف برای هر سرور، و میانگین زمان پاسخ (زمان انتظار و زمان سرویس) برای هر سرور می‌توانند با انواع روش‌ها یافت شود. در یک مدل شبکه، یک سرور معمولاً با توزیع زمانی سرویس، از زمان سرویس مشتری استنتاج شده، رابطه دارد. بر اساس زمان ورود به سرور، یک مشتری

سرویس دریافت می‌کند، و مدت زمان آن با توزیع زمانی سرویس تعیین می‌شود. علاوه بر این، میانگین طول صف برای هر سرور، و میانگین زمان پاسخ (زمان انتظار و زمان سرویس) برای هر سرور می‌تواند با انواع روش‌ها یافت شود.

در یک مدل شبکه، یک سرور معمولاً با توزیع زمانی یک سرویس در ارتباط است، که از آن زمان‌های سرویس‌دهی به مشتری استنتاج می‌شود. در زمان ورودی به یک سرور، مشتری سرویس دریافت می‌کند، و مدت زمان هر یک با توزیع زمان سرویس تعیین می‌شود.

ما حال توجه خود را به برخی از سیستم های صف شناخته شده تر، مفاهیم استفاده شده برای نمایش آن‌ها، کمیت سنجی عملکرد مورد نظر، و روش‌های محاسباتی آن‌ها جلب می‌کنیم. ما تا به حال مفاهیم بسیاری را برای مقادیر مورد نظر متغیرهای تصادفی و فرآیند احتمالی معرفی کردیم. شکل (۷-۳) این موارد را بررسی می‌کند و موارد سودمند دیگری که برای تحلیل سیستم صف سودمند هستند را به آن می‌افزاید. متن زیر به‌صورت خلاصه بر پارامترهای مهم تر بحث می‌کند.

نرخ ورود برای یک سیستم صف با جریان ورودی به صف از یک منبع خارجی تعریف می‌شود. این نرخ به‌عنوان میانگین نرخ تعریف می‌شود، که از یک فرآیند ورودی حاصل شده است. میانگین زمان بین دو ورود برای یک فرآیند ورود به شرح زیر است:

$$E[\tau] = ۱/\lambda \qquad (۱-۷)$$

پارامتر نرخ سرویس به روشی تعریف می‌شود که مشابه با نرخ ورود است. این نرخ یک نرخ میانگین است، که تعریف می‌کند که چند مشتری در هر واحد زمانی در زمانی که سرور مشغول است، پردازش شده اند. نرخ سرویس می‌تواند ازنظر متغیر تصادفی زمان سرویس‌دهی تغییر کند:

$$\mu = ۱ / E[s] \qquad (۲-۷)$$

اغلب، می‌خواهیم احتمال اینکه سیستم شامل دقیقاً n مشتری در یک حالت پایدار باشد را بدانیم. با در نظر گرفتن همه احتمالات (n) از صفر تا نامحدود، توزیع احتمال برای تعداد مشتریان در سیستم تعریف می‌شود.

λ = نرخ ورود به صف

M = نرخ سرویس‌دهی یک سرور (میانگین)

P_n = احتمال اینکه n مشتری در یک حالت پایدار در سیستم وجود داشته باشند.

C = تعداد سرورهای شناخته شده در سیستم صف

N = متغیرهای تصادفی برای تعدادی مشتری در حالت پایدار

L = E[N]، تعداد مشتریان مورد انتظار در حالت پایدار سیستم.

W_q = متغیر تصادفی برای زمان انتظار مشتری در صف

S = متغیر تصادفی برای زمان سرویس‌دهی مشتری

N_q = متغیر تصادفی برای تعداد مشتریان در یک صف در حالت پایدار

L_q=$E[N_q]$، تعداد مورد انتظار مشتریان در یک صف در حالت پایدار

N_s = متغیر تصادفی برای تعدادی مشتری در یک سرور در حالت پایدار

w = $W_q + S$ ، متغیر تصادفی برای مجموع زمان در یک سیستم

شکل (۷-۳): نماد متغیرهای تصادفی و فرآیند احتمالی

با توجه به هر سیستم صف، میانگین تعداد مشتریان (N) در سیستم، در حالت پایدار است. این مقدار می‌تواند مجموع مشتریان در یک صف (N_q) و در سرورها (N_s) باشد:

$$N = N_q + N_i$$

$$L = E[N] = E[N_q] + E[N_s] \qquad (۳-۷)$$

مجموع زمانی که یک مشتری در سیستم سپری می‌کند می‌تواند مجموع زمان انتظار در صف (q_t) و زمان در سرور (s_t) باشد. مجموع زمان، مجموع زمان مورد انتظار در حالت پایدار، به شکل زیر است:

$$W = W_q + S$$

$$E[W] = E[W_q] + E[S] \qquad (۴-۷)$$

$A/B/c/K/m/Z$
A=تعریف فرآیند ورود
B= توزیع زمان سرویس‌دهی
C= تعداد سرورهای یکسان
K= حداکثر تعداد مشتریان مجاز در سیستم (پیش فرض= ∞)
M=تعداد مشتریان مجاز برای اینکه قبل از توقف فرآیند ورود به سیستم برسند (پیش فرض= ∞)
Z= اصول استفاده شده برای ترتیب مشتریان در صف (پیش فرض= FIFO)
D= زمان سرویس‌دهی قطعی یا نرخ ورود
G= زمان سرویس‌دهی عمومی یا نرخ ورود
M= زمان سرویس‌دهی مارکوف یا نرخ ورود

شکل (۷-۴): مفاهیم کندال

علاوه بر مقادیر تشریح شده قبلی برای مقادیر مربوط به سیستم‌های صف، معرفی یک نشانه گذاری برای پارامترهای سیستم صف سودمند است. نشانه گذاری که در اینجا استفاده می‌کنیم را مفاهیم کندال گویند، که در شکل (۷-۴) آورده شده است.

نمادهای استفاده شده در یک تشریح مفهوم کندل نیز دارای برخی از تعاریف استاندارد هستند. شکل(۷-۵) موارد شایع تری را برای فیلدهای A و B نشانه گذاری نشان می‌دهد.

اصول سرویس استفاده شده برای مرتب کردن مشتریان در صف می‌تواند انواع مختلفی داشته باشد، مانند اول ورود، اول خروج (FIFO)، آخرین مورد وارد شده (LIFO)، به ترتیب اولویت، ترتیب تصادفی و سایر موارد. سپس، ما سیستم‌های صف بندی متعددی را بررسی می‌کنیم و برای مقادیر کارایی مهم تر عباراتی ارائه می‌دهیم.

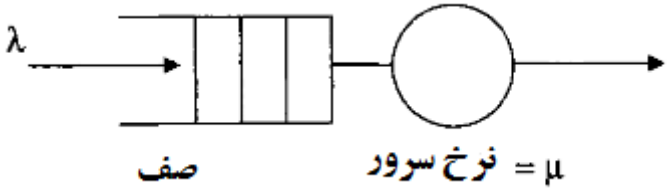
۷-۱-۱ سیستم صف M/M/۱

سیستم صف $M/M/1$ با یک فرآیند ورود پواسون و توزیع زمان سرویس نمایی، با یک سرور، و یک صف با ترتیب FIFO مشخص می‌شود. سیستم نشان داده‌شده در شکل(۷-۶) ممکن است یک بافر ورودی را نشان دهد که شامل بایت‌های ورود، با پردازنده I/O به‌عنوان سرور باشد. چند مورد از مقادیری که برای این نوع سیستم صف مورد توجه هستند میانگین طول صف، زمان انتظار یک مشتری در صف، مجموع زمانی که یک مشتری در سیستم صرف می‌کند، و استفاده از سرور است.

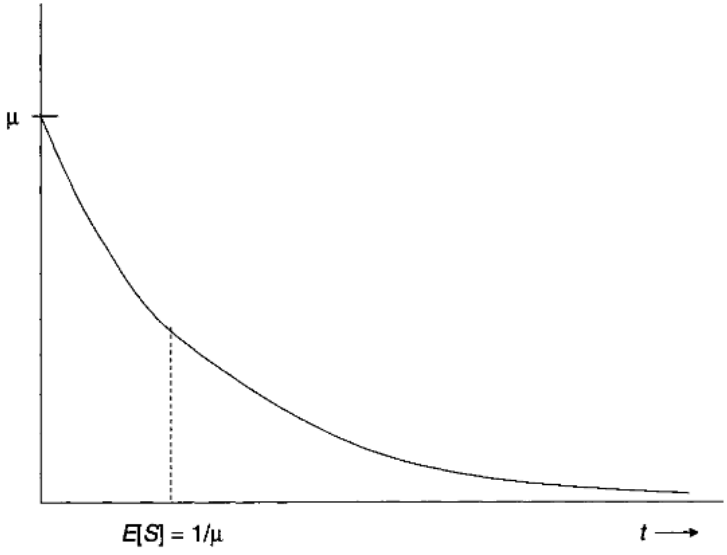
بیایید به توزیع نمایی سرویس نگاه کنیم. داریم:

$$S = \mu e^{-\mu t} \qquad (۵-۷)$$

همان‌طور که در شکل (۷-۷) نشان داده شد. در شکل، $E[S]$ میانگین زمان سرویس‌دهی به یک مشتری در سرور است. سپس، بیایید معادله حالت پایدار را برای سیستم $M/M/1$ به دست اوریم.



شکل (۷-۶): مدل سیستم صف $M/M/1$



شکل (۷-۷): توزیع نمایی سرویس‌دهی

سیستم $M/M/1$ یک فرآیند تولد،مرگ است، که در فصل ۶ بحث شد. فرض کنید که داریم:

(۶-۷)

P_n^t = احتمال این است که n مشتری در زمان t در سیستم باشند.

از بحث‌های اولیه در مورد فرآیند تولد،مرگ دریافتیم که:

$$P_n(t+h) = P_n(t)[1 - \lambda_n h - \mu_n h]$$

$$+ P_{n-1}(t)\lambda_{n-1} h + P_{n+1}(t)\mu_{n+1} h + o(h) \qquad (۷-۷)$$

و

$$P_0(t+h) = P_0(t) - P_0(t)\,\lambda_0 h + P_1(t)\,\mu_1 h + o(h) \qquad (۸-۷)$$

به دنبال برخی از استدلال برای استنتاج احتمال حالت پایدار که برای فرآیند تولد،مرگ عمومی انجام دادیم، ما معادله حالت پایدار را برای سیستم $M/M/1$ به دست آوردیم:

$$\lambda P_0 = \mu P_1 \qquad (۹-۷)$$

$$(\lambda + \mu)P_n = \lambda P_{n-1} + \mu P_{n+1} \quad \text{for } n > 0 \qquad (۱۰-۷)$$

حال اگر فرض کنیم که u به میانگین استفاده از سرور اشاره دارد، ما این مقدار را به‌عنوان میانگین زمان سرویس‌دهی به یک مشتری تقسیم بر میانگین زمان ورود تعریف می‌کنیم (معادله (۷-۱) و (۷-۲)، سپس داریم:

$$u = (1/\mu)\,(1/\lambda) = \lambda/\mu \qquad (۱۱-۷)$$

با حل معادلات حالت پایدار (۷-۹) و (۷-۱۰) داریم:

$$P_1 = \frac{\lambda}{\mu} P_0 \quad \text{برای } n = 0 \qquad (۱۲-۷)$$

به طور مشابه برای $n = 1$

$$(\lambda + \mu)\,P_1 = \lambda P_0 + \mu P_2$$

$$\mu P_2 = (\lambda + \mu)\,P_0 - \lambda P_0$$

$$P_2 = (1 + \lambda/\mu)P_1 - (\lambda/\mu)P_0 \qquad (۱۳-۷)$$

$$P_2 = (\lambda/\mu)^2 P_0$$

برای $n = 1$

به طور مشابه:

$$P_۳ = (\lambda/\mu)^۲ P_۰$$

$$P_n = (\lambda/\mu)^n P_۰ \quad n > ۰ \text{ برای} \qquad (۱۴-۷)$$

$$P_n = \mu^n P_۰ \quad n > ۰ \text{ برای}$$

کمتر از ۱ است و دارای یک طول صف محدود هستیم. حال، می‌دانیم u در اینجا فرض می‌کنیم که

که روابط زیر برقرار است:

$$\sum_{n=۰}^{\infty} P_n = ۱$$

و

$$\sum_{n=۰}^{\infty} P_n = P_۰ \sum_{n=۰}^{\infty} u^n = ۱ \qquad (۱۵-۷)$$

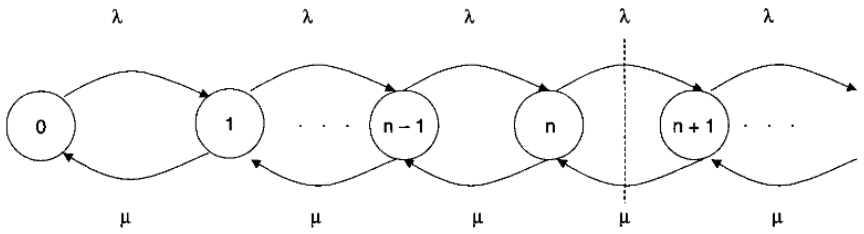
پس:

$$P_۰ = ۱ / \left(\sum_{n=۰}^{\infty} u^n\right)$$

سمت راست معادله (۷-۱۵) به‌عنوان تصاعد هندسی تشخیص داده‌شده است که راه‌حل زیر را دارد:

$$P_۰ = \frac{۱}{۱ / (۱ - \mu)}$$

$$P_{\cdot} = ۱ - u = ۱ - (\lambda / \mu) \qquad (۱۶-۷)$$



شکل (۷-۸): نمودار انتقال حالت سیستم $M/M/۱$

با ترکیب معادلات (۷-۱۴) و (۷-۱۶)، به احتمال حالت پایداری می‌رسیم که در آن n مشتری در یک سیستم $M/M/۱$ وجود دارند:

$$P_n = (۱ (\lambda / \mu)) ((\lambda / \mu)^{۲} \qquad (۱۷-۷)$$

حال به میانگین تعداد مشتریان در سیستم در حالت پایدار نگاه کنید. این یک مقدار مورد انتظار N را می‌دهد،که از روابط زیر حاصل می‌شود:

$$E[N] = \sum_{n=۰}^{\infty} n p_n$$

$$= \sum_{n=۰}^{\infty} n (۱ - (\lambda / \mu)) ((\lambda / \mu)^n$$

$$= (۱ - (\lambda / \mu) \sum_{n=۰}^{\infty} n \ ((\lambda / \mu)^n$$

$$= (۱ (\lambda /\mu)) ((\lambda /\mu) + ۲(\lambda /\mu)^{۲} + ۳(\lambda /\mu)^{۳} + \cdots)$$

$$= (۱ (\lambda /\mu)) (\lambda /\mu)(۱ + ۲(\lambda /\mu)^{۱} + ۳(\lambda /\mu)^{۲} + \cdots) \qquad (۱۸-۷)$$

$$= (۱ (\lambda /\mu)) ((\lambda /\mu) \sum_{n=۱}^{\infty} + n (\lambda /\mu)^{n-۱}$$

$$= \frac{(۱ - (۱ (\lambda /\mu)) ((\lambda /\mu)}{(۱ - (۱ (\lambda /\mu)^{۲}}$$

$$E [N] = \frac{\lambda /\mu}{۱ - (\lambda /\mu)}$$

مقدار میانگین زمانی که مشتری باید در صف منتظر بماند، با فرض اینکه مشتریان دیگری نیز در صف هستند، با تعداد مشتریان تقسیم بر میانگین زمان سرویس‌دهی سرویس به دست می‌آید:

$$E \left[w_q \mid n = i\right] = i /\mu \qquad (۱۹-۷)$$

زمان مورد انتظار در صف، تابعی از میانگین زمان انتظار و احتمال حالت پایداری است که i مشتری در سیستم وجود دارند:

$$E \left[W_q\right] = \sum_{i=۱}^{\infty} (i/ \mu) P_i$$

$$= (۱ / \mu) E [N] \qquad -۷)$$

(۲۰

$$E \left[W_q\right] = \frac{(\lambda /\mu^{۲})}{۱ - (\lambda /\mu)}$$

در ترکیب زمان انتظار صف و زمان سرویس‌دهی مورد انتظار $E[s]$ (معادله (۷-۲)) مجموع زمانی که مشتری در سیستم می‌گذراند به دست می‌اید، که زمان مورد انتظار می‌گویند:

$$E[W] = E[W_q] + E[s]$$

$$= \frac{(\lambda/\mu^{۲})}{۱-(\lambda/\mu)} + \frac{۱}{\mu}$$

$$= \frac{۱}{\mu}\left(\frac{(\lambda/\mu)}{۱-\lambda/\mu} + ۱\right) \quad (۷-۲۱)$$

$$= \frac{۱}{\mu}\left(\frac{۱}{۱-\lambda/\mu}\right)$$

$$E[W] = ۱/(\mu-\lambda)$$

اگر معادله (۷-۱۸) را به شکل زیر بازنویسی کنیم:

$$E[N] = \lambda/(\mu-\lambda) \quad (۷-۲۲)$$

با استفاده از معادله (۷-۲۱)، نتایج Little را به دست می‌اوریم:

$$E[N] = \lambda E[W] \quad (۷-۲۳)$$

نتایج Little در کل برای هر سیستم صف در حالت پایدار که فرض تعادل جریان که در ابتدا بر آن بحث شد را تأیید می‌کند، برقرار است. به همین ترتیب، این به ما یک رابطه مهم برای تأثیر نرخ ورود و طول صف بر مجموع زمان انتظار مشتری را می‌دهد. یک نتیجه مربوطه، که به Little نسبت داده‌شده است، تعادل طول صف و زمان انتظار صف را بیان می‌کند و برای سیستم صف در حالت پایدار برقرار است:

$$E\left[N_q\right] = \lambda E\left[W_q\right]$$

(۲۴-۷)

این نسخه دوم نتایج Little بیان می‌کند که طول مورد انتظار صف می‌تواند مستقیماً از زمان نرخ ورود و زمان انتظار صف حاصل شود.

مجموع زمان انتظار در سیستم، می‌تواند با استفاده از نتایج Little یا با جمع‌بندی زمان انتظار صف و زمان سرویس مورد انتظار یافت شود.

استفاده از سرور یک مقدار مفید برای تعیین این است که چند سرور معادل باید برای سرویس‌دهی به فرآیند ورودی مشخص تدارک دیده شوند. روش سرراست است و شامل حل یک سیستم صف M/M/۱ با استفاده از روش‌هایی که قبلاً نشان داده‌شده، است. برای نمونه، فرض کنید که، دارای یک سیستم M/M/۱ با یک نرخ ورودی شش مشتری در هر دقیقه و زمان انتظار ده ثانیه هستیم. سپس استفاده از سرور، همان‌طور که توسط معادله (۷-۱۱) داده‌شده است، ۱ است. این به معنی این است که سرور می‌تواند ۱۰۰ درصد مواقع مشغول باشد، اما، در حقیقت، می‌توانند مشتریان کافی را پردازش کند لذا از صف نامحدود جلوگیری می‌کند. حال فرض کنید که نرخ ورود به ۶۰ مشتری در هر دقیقه افزایش یابد لذا استفاده از سرور، در شرایط سربار، ۱۰ می‌باشد. اگر سرعت سرور را ده برابر کنیم، یا ده سرور با سرعت اصلی ارائه دهیم، بهره وری ۱ خواهد بود. در کل، اگر بهره وری کمتر از ۱ باشد، سرور می‌تواند جریان مشتریان را حفظ کند و صف نامحدود حاصل نمی‌شود. اگر، بهره وری بزرگ‌تر از ۱ باشد، بهره وری، به بزرگ‌ترین عدد صحیح بعدی گرد می‌شود، و نشان‌های از تعداد سرورهای یکسانی است که برای حفظ جریان مشتریان لازم هستند.

یک مشخصه جالب نهایی سیستم صف M/M/۱ این حقیقت است که زمان انتظار صف و مجموع زمان انتظار هر دو توزیع نمایی دارند. برای نمونه، زمان انتظار صف به شرح زیر است:

$$P\left[0 < W_q \leq t\right] = \sum_{n=1}^{\infty} P_n \int_0^t f_{W_q|n}(X|n)\, dx \qquad (۲۵-۷)$$

از معادله (۱۴-۷) و توزیع (پواسون)، داریم:

$$P\left[0 < W_q \leq t\right] = \sum_{n=1}^{\infty} (\lambda/\mu)^n \left(1 - (\lambda/\mu)\right) \int_0^t \frac{\mu^n x^{n-1}}{(n-1)!} e^{-\mu x}\, dx$$

$$= \int_0^t \lambda e^{\mu x} (1 - \lambda/\mu)) \sum_{n=1}^{\infty} \frac{(\lambda x)^{n-1}}{(n-1)!}\, dx$$

$$= \int_0^{\infty} \lambda e^{\mu x} (1 - \lambda/\mu))\, e^{\lambda x}\, dx \qquad (۲۶-۷)$$

$$= \lambda/\mu \int_0^t (\mu - \lambda)\, e^{-(\mu-\lambda)x}\, dx$$

$$= \lambda/\mu \left[1 - e^{-t(\mu-\lambda)}\right]$$

از معادله (۲۱-۷)، با جایگزینی داریم:

$$P\left[0 < W_q \leq t\right] = \lambda/\mu\left[1 - e^{-t/E[W]}\right] \qquad (۲۷-۷)$$

شامل $P(W_q = 0]$:

$$P\left[W_q \leq t\right] = P_0 + \lambda/\mu \left[1 - e^{-t/E[W]}\right] \qquad (۲۸-۷)$$

با جایگزینی $w_q [۰] = ۱ - \frac{\lambda}{\mu}$ داریم:

$$P [W_q \leq t] = ۱ - \lambda / \mu [e^{-t/ E [W]}]$$

(۲۹-۷)

از این توزیع، می‌توانیم درصد زمان مورد انتظار برای r درصد از مجموع تعداد مشتریان را پیدا کنیم. درصد هر متغیر تصادفی به شکل زیر تعریف می‌شود:

$$P [x \leq \pi (r)] = r / ۱۰۰$$

-۷)

(۳۰

در مورد زمان انتظار صف، برای مثال، اگر بخواهیم زمان انتظاری پیدا کنیم که از ۹۰ درصد مشتریان در سیستم بیشتر نشود، داریم:

$$۱ - e^{- \pi (۹۰)/ E [W]} = ۰.۹$$

$$۰.۱ = e^{- \pi (۹۰)/ E [W]}$$

$$In ۰.۱) = \frac{- \pi / (۹۰)}{E [W]}$$

$$\pi (۹۰) = ۲.۳ E [W]$$

(۳۱-۷)

۷-۱-۲ سیستم M/M/I/K

یک تغییر جالب واقع‌بینانه در سیستم M/M/I پایه یک سیستم با اندازه صف محدود است. در این سیستم، زمانی که صف پر می‌شود، ورودی‌های جدید گم می‌شوند و به آن‌ها هیچ سرویسی داده نمی‌شود. برای مثال، در یک سیستم ورودی با فضای بافر ورودی محدود و بدون پروتکل

کنترل جریان، این کاملاً واقع بینانه است. نمودار انتقال حالت تولد،مرگ برای سیستم M/M/I/K در شکل (۷-۹) نشان داده‌شده است.

مانند سیستم M/M/I داریم:

$$P_n = (\lambda/\mu)^n P_. \quad \text{for } n \geq n \geq ۰ \qquad (۳۲-۷)$$

با استفاده از قوانین احتمال کل، داریم:

$$\sum_{i=۰}^{K} P_۱ = ۱$$

$$\sum_{i=۰}^{K} (\lambda/\mu)^i P_. = ۱ \qquad (۳۳-۷)$$

$$P_. \sum_{i=۰}^{K} (\lambda/\mu)^i = ۱$$

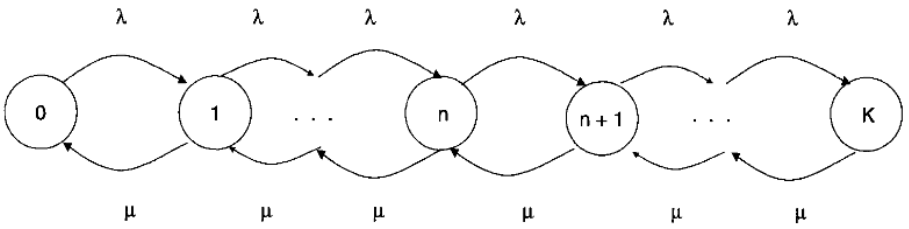
در کل یک سری هندسی است، که رابطه زیر را حاصل می‌کند:

$$P_. \frac{۱ - (\lambda/\mu)^{K+۱}}{۱ - (\lambda/\mu)}$$

پس:

$$P_. = \frac{۱-(\lambda/\mu)}{۱-(\lambda/\mu)^{K+۱}} \qquad (۷-$$

۳۴)



شکل (۷-۹): نمودار حالت برای سیستم M/M/I/K

با جایگزینی در معادله (۷-۳۲) داریم:

$$P_n = \frac{۱-(\lambda/\mu)}{۱-(\lambda/\mu)^{K+۱}} \left(\frac{\lambda}{\mu}\right)^n \quad \text{for } K \geq n \geq ۰ \qquad (۳۵-۷)$$

اگر نرخ ورودی برابر با نرخ سرویس‌دهی باشد، داریم:

$$P_۰ \sum_{i=۱}^{K} (\lambda/\mu)^i = ۱ \quad \text{for } \lambda = \mu$$

$$P_۰ = ۱ / (K+۱) \quad \text{for } \lambda = \mu \qquad (۳۶-۷)$$

و

$$P_n = ۱ / (K+۱) \quad \text{for } K \geq n \geq ۰ \text{ and } \lambda = \mu \qquad (۳۷-۷)$$

تعداد مشتریان مورد انتظار در سیستم، برای یک سیستم با ورودی و نرخ‌های سرویس‌دهی نابرابر، به شکل زیر است:

$$E[N] = \sum_{i=0}^{K} i\, P_i$$

$$= \sum_{i=0}^{K} \left[\frac{1-(\lambda/\mu)}{1-(\lambda/\mu)^{K+1}}\right] \left(\frac{\lambda}{\mu}\right)^i \qquad (۳۸-۷)$$

$$E[N] = \frac{1-(\lambda/\mu)}{1-(\lambda/\mu)^{K+1}} \sum_{i=0}^{k} i \left(\frac{\lambda}{\mu}\right)^i$$

بعد از برخی از عملیات جبری و ساده سازی جمع، داریم:

$$E[N] = \frac{(\lambda/\mu)}{1-(\lambda/\mu)} - \frac{(K+1)(\lambda/\mu)^{K+1}}{1-(\lambda/\mu)^{K+1}} \quad \text{for } \lambda \neq \mu \qquad (۳۹-۷)$$

می‌توانیم ببینیم که، برای هر مقدار بزرگ K، عبارت دوم تقریباً صفر است و عبارتی مانند عبارت M/M/I حاصل می‌شود.

برای موردی که نرخ ورودی و سرویس‌دهی برابر است داریم:

$$E[N] = \sum_{i=0}^{K} i p_i$$

$$E[N] = \frac{1}{K+1} \sum_{i=0}^{K} i \qquad (۴۰-۷)$$

$$E[N] = K/2 \quad (\text{for } \lambda = \mu)$$

برای محاسبه توزیع زمان انتظار برای سیستم M/M/I/K، ما باید احتمال برای تعداد مشتریان در صف را در زمان ورود مشتری حساب کنیم، با توجه به اینکه مشتری در سیستم پذیرفته می‌شود. رابطه آن به شکل زیر است:

$$P\ (n\ customers\ in\ system\ |N < K)\ \ P_n\ /\ (۱ - P_k) \qquad (۴۱-۷)$$

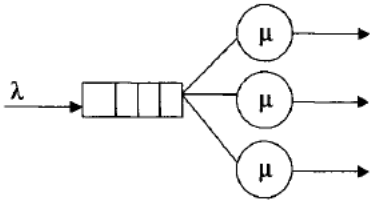
برای این، می‌توانیم توزیع زمان انتظار زیر را داشته باشیم:

$$P\ (w \le t) = ۱ - \sum_{N=۰}^{K-۱} P_n\ \ (۱ - P_K)\ \sum_{K=۰}^{N} e^{-\mu t}\frac{(\mu t)^K}{K!} \qquad (۴۲-۷)$$

این مقدار می‌تواند به روش مشابهی مانند متغیرهایی برای سیستم $M/M/I$ باشد.

۳-۱-۷ سیستم M/M/C

سیستم $M/M/C$، نشان داده‌شده در شکل (۷-۱۰)، شامل یک خط انتظار واحد است که از C سرور یکسان تغذیه می‌کند. فرآیند ورود پواسون است، و سرور دارای زمان سرویس‌دهی نمایی است. نمودار انتقال حالت برای یک سیستم $M/M/C$ در شکل (۷-۱۱) نشان داده‌شده است. حال بیایید معادلات تعادل جریان را برای نمودار انتقال حالت بنویسیم.



شکل (۷-۱۰): یک سیستم $M/M/C$ برای $C = ۳$

$$P_۱ = \frac{\lambda}{\mu} P_۰$$

$$P_۲ = \frac{\lambda^۲}{۲\mu^۲} P_۰$$

$$P_۳ = \frac{\lambda^۳}{۶\mu^۳} P_۰$$

⋮

(۴۳ -۷)

$$P_C = \frac{\lambda^C}{C!\ \mu^C} P_۰$$

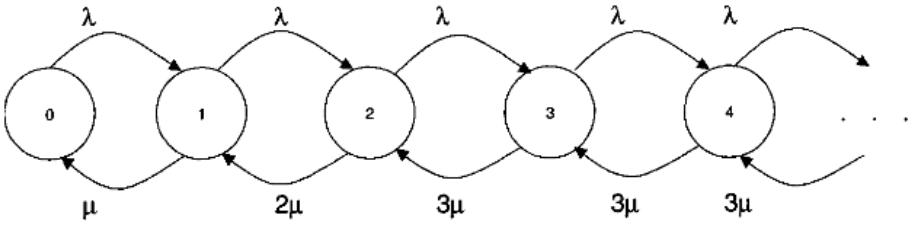
$$P_{C+۱} = \frac{\lambda^{C+۱}}{CC!\,\mu^{C+۱}} P_۰$$

⋮

$$P_N = \frac{\lambda^N}{C^{N-C}C!\ \mu^N} P_۰$$

بنابراین:

$$P_N = \begin{cases} \dfrac{\lambda^N P_۰}{N!\ \mu^N} & \text{for } N \le C \\ \dfrac{\lambda^N P_۰}{C^{N-C}\ C!\ \mu^N} & \text{for } N > C \end{cases}$$



شکل (۷-۱۱): نمودار انتقال حالت برای یک سیستم $M/M/C$، $C = ۳$

برای یافتن احتمال اینکه هیچ مشتری در سیستم نیست، ما همه احتمالات را باهم جمع می‌کنیم:

$$۱ = P_۰ \left[\begin{matrix} ۱ + \ +(\lambda/\mu)^{۲}/۲ + \cdots + (\lambda/\mu)^{C}\, C\,! + \\ (\lambda/\mu)^{C+۱}\ CC! + \cdots + (\lambda/\mu)^{N}\ C^{N-C}\, C! + \cdots \end{matrix} \right]$$

$$۱ = P_۰ \left[\sum_{i=۱}^{C} (\lambda/\mu)^{i}\, i! \ + \ \sum_{i=C}^{\infty} \left(\frac{\lambda}{\mu}\right)^{i} / \left(C^{i-C}\ C!\right) \right] \qquad (۴۴-۷)$$

$$۱ = P_۰ \left(\sum_{i=۱}^{C} (\lambda/\mu)^{i}\, i! + (\lambda/\mu)^{C}\ C!\ \left(۱ - (\lambda/C\mu)\right) \right)$$

طول صف مورد انتظار می‌تواند با کسر تعداد مشتریان در سرویس از تعداد مشتریان مورد انتظار در سیستم به دست آید:

$$E\left[N_q\right] = \sum_{N=C}^{\infty} (N - C)\, P_N$$

$$E\left[N_q\right] = \frac{P_0 (\lambda/\mu)^c (\lambda/\mu)}{c!\ (1-(\lambda/\mu))^2} \quad (7-45)$$

با استفاده از نتایج Little، می‌توانیم زمان انتظار صف را محاسبه کنیم:

$$W_q = E\left[N_q\right] / \lambda \quad (7-46)$$

مجموع زمان انتظار:

$$w = W_q + E[s]$$

$$w = \left(E\left[N_q\right]/\lambda\right) + (1/\mu) \quad (7-47)$$

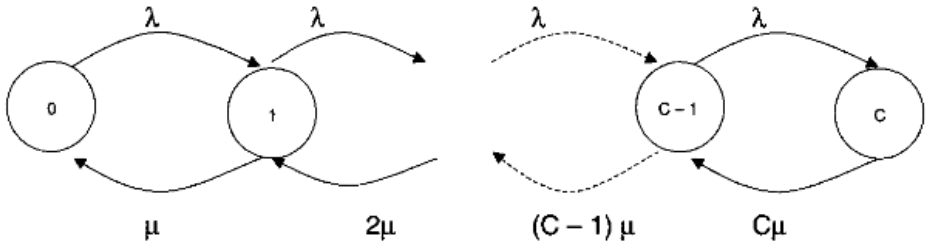
مجموع تعداد مشتریان در سیستم:

$$E[N] = \lambda w$$

$$E[N] = E\left[N_q\right] + (\lambda/\mu) \quad (7-48)$$

برای برخی از سیستم‌های چند سروری، هیچ صفی وجود ندارد که در آن مشتریان منتظر سرویس باشند. در این مورد، مشتری که در زمانی می‌رسد که همه سرورها مشغول هستند دور انداخته می‌شود، تا بار دیگر مجدد تلاش کند. نمودار انتقال حالت در شکل (۷-۱۲) نشان داده‌شده است. این سیستم اغلب

سیستم افت M/M/C گفته می‌شود، چرا که مشتریانی که در زمانی که همه سرورها مشغول هستند به سیستم وارد می‌شوند، گم می‌شوند.



شکل (۷-۱۲): سیستم افت M/M/C

با نوشتن معادلات تعادل جریان، یک احتمال حالت پایدار را، همان‌طور که برای سیستم M/M/C به دست آوردیم، حاصل می‌کنیم:

$$P_N = (\lambda/\mu)^N N! = ۱ + (\lambda/\mu) + (\lambda/\mu)^۲ ۲! + \cdots (\lambda/\mu)^C C! \qquad (۴۹-۷)$$

احتمال اینکه یک مشتری دور انداخته شود، را می‌توان از عبارت قبلی با $N = C$ یافت. ازآنجایی‌که هیچ صفی وجود ندارد، طول صف و زمان انتظار صف صفر هستند، و مجموع زمان انتظار برابر زمان مورد انتظار سرویس‌دهی است.

۷-۱-٤ سیستم M/G/I

سیستم‌های صفی که تابه‌حال بر آن‌ها بحث کردیم برای فرآیندهای ورود و سرویس‌دهی دارای مشخصه مارکوف بودند، که مدل کردن سیستم را به‌عنوان یک فرآیند تولد،مرگ و نوشتن معادلات جریان را با بازرسی، ممکن می‌ساختند. سپس، به سیستمی نگاه می‌کنیم که در آن زمان سرویس دارای مشخصه مارکوف است. در سیستم M/G/۱، هر مشتری دارای زمان‌های سرویس‌دهی متفاوت و مستقلی است. به دلیل اینکه تضمین نشده است که زمان سرویس‌دهی مارکوفی باشد، سیستم زنجیره مارکوف را نشان نمی‌دهد و باید برای استنتاج آمارهای معنی‌دار به روش‌های دیگر رو آوریم. یک رویکرد به طور متداول استفاده شده نگاه به فرآیندی است که

کارهایی که از سیستم خارج می‌شوند را تشریح می‌کند، که یک فرآیند احتمالی است که در زنجیره مارکوف رخ می‌دهد. این نشان می‌دهد که، توزیع تعداد کارها در سیستم در هر نقطه زمانی و هر کاری در سیستم در زمانی که مشتری از سیستم جدا شود، یکسان است.

با خلاصه‌سازی رویه، می‌توانیم جنبه‌های اصلی سیستمی را تحلیل کنیم که توسط فرآیند غیر مارکوفی با زیر بخش‌های مارکوفی سیستم (در مورد فرآیند خروج) و استقرا به سیستم اصلی تشریح شده است. این نوع تحلیل به آنچه به‌عنوان زنجیره مارکوف شناخته شده، متکی است. انحراف متغیرها برای سیستم $M/G/1$ فراتر از دامنه این کتاب است.

سیستم $M/G/1$ در بسیاری از شرایط سودمند است، چرا که می‌توانیم فرآیند سرویس‌دهی شناخته شده را ازنظر اهمیت آن مشخص کنیم و سپس عملکرد آن را در حضور فرآیند ورودی تصادفی ارزیابی کنیم.

۷-۱-۵ سیستم $G/M/I$

در بخش قبلی، ما بر شرایطی بحث کردیم که در آن یک سیستم دارای یک فرآیند سرویس‌دهی غیر مارکوفی باشد. سپس، موردی را در نظرخواهیم گرفت که در آن زمان سرویس‌دهی تصادفی است، و فرآیند ورود غیر مارکوفی است. ما فرض می‌کنیم که زمان‌های بین دو ورودی مستقل هستند و به‌صورت یکسانی توزیع‌شده‌اند. بار دیگر، می‌توانیم زنجیره مارکوف تعبیه‌شده در این سیستم را بیابیم که رفتار آن ضرورتاً معادل با رفتار سیستم در حالت پایدار است. در این مورد، متغیر تصادفی تعداد مشتریان در سیستم را تعریف می‌کند، زمان دقیقی که ورود دیگری صورت می‌گیرد، یک فرآیند زنجیره مارکوف شکل می‌گیرد. مانند سیستم‌های دیگری که بر آن‌ها بحث کردیم، آمارهای مورد نظر از احتمال داشتن یک سیستم خالی در محاسبات ارزش استفاده می‌کنند.

۷-۲ شبکه‌هایی از صف‌ها

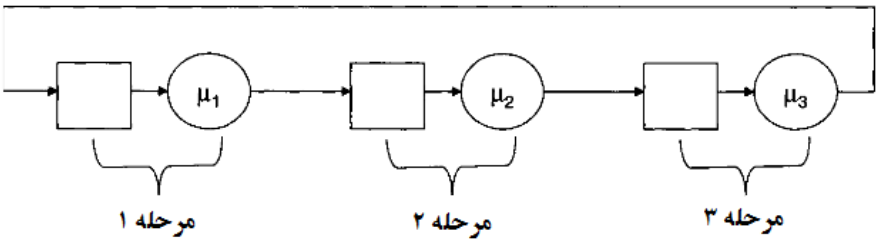
تابه‌حال، ما سیستم‌های صفی را در نظر گرفتیم که تنها شامل یک حالت بودند. به همین ترتیب، سیستم‌هایی که تنها یک صف دارد و یک سرور یا مجموعه سرور دارند، و مشتریان تنها به آن صف وارد می‌شوند و تنها از سرویس‌های گفته شده خارج می‌شوند را در نظر گرفتیم. این شرایط برای سیستم‌های نسبتاً ساده‌ای خوب است که نه سیستم دیگری متصل هستند و از دیگر سیستم‌های متصل، ایزوله هستند. ما، حال موردی را در نظر می‌گیریم که در آن چندین سیستم صف باهم در ارتباط هستند و برای یافتن متغیرهای معنی‌داری مانند رفتار سیستم تلاش می‌شود.

با اشاره به شکل (۷-۱)، به یاد می‌آوریم که یک شبکه از صف‌ها از ارتباط جریان خروجی یک سیستم صف به ورودی صف دیگر، برای هر تعداد سیسم صف دلخواه که به روش دلخواه به هم متصل شده‌اند، نتیجه می‌شود. این، ما بر مفهوم شبکه‌های باز و بسته‌ای بحث کردیم که آن یک شبکه باز به‌عنوان شبکه‌ای تعریف شد که در آن ورودی از سمت جهان خارج و خروج به سمت جهان خارج مجاز است، و شبکه بسته شبکه‌ای است که در آن این موارد مجاز نیستند. ما بر کلاس‌های عمومی هر دو نوع در اینجا بحث می‌کنیم.

۷-۲-۱ شبکه‌های بسته

یک شبکه سه مرحله‌ای بسته از صف‌های نشان داده‌شده در شکل (۷-۱۳) را در نظر بگیرید. فرض کنید که زمان سرویس‌دهی برای هر سرویس به‌صورت نمایی توزیع‌شده باشد و برای آن سرور منحصربه‌فرد باشد و سیستم شما دو مشتری داشته باشد. ما می‌توانیم این سیستم را با فرآیند مارکوف با هر حالت در فرآیند که با سه‌تایی تعریف می‌شود، تشریح کنیم:

$$\text{حالات} = \{N_1, N_2, N_3\} \qquad (۵۰-۷)$$



شکل (۷-۱۳): شبکه صف بسته سه مرحله‌ای

در این معادله N_i تعداد مشتریان در صف i است. زمانی که ما دو مشتری داریم:

$$\sum_{i=1}^{3} N_i = ۲ \qquad (۵۱-۷)$$

نمودار انتقال حالت برای سیستم، با حالاتی که مانند آنچه در معادله (۷-۵۰) تعریف‌شده است برچسب خورده اند، در شکل (۷-۱۴) نشان داده‌شده است. برچسب‌ها بر روی یال‌ها حرکت مشتریان از مرحله‌ای به مرحله دیگر را نشان می‌دهند و به نرخ سرویس‌دهی در مرحله که مشتریان خارج می‌شوند، وابسته است.

برای یافتن احتمالات حالت پایدار برای هر حالت در سیستم، می‌توانیم معادلات تعادل جریان را بنویسیم. همان‌طور که در ابتدا بحث شد، فرض تعادل جریان بیان می‌کند که می‌توانیم احتمال حالات پایدار یک فرآیند مارکوف را با نوشتن معادلاتی که جریان مشتریان به حالت و خارج از حالت در شبکه متعادل می‌کند، پردازش کنیم. برای هر حالت تکی، سپس، می‌توانیم معادله تعادلی را بنویسیم که جریان به درون یک حالت با جریان خروجی از حالت تعادل دارد. برای حالات شکل (۷-۱۴)، معادلات تعادل زیر را می‌نویسیم:

$$\pi(۲,۰,۰)\mu_۱ = \pi(۱,۰,۱)\mu_۳ \qquad (۵۲-۷)$$

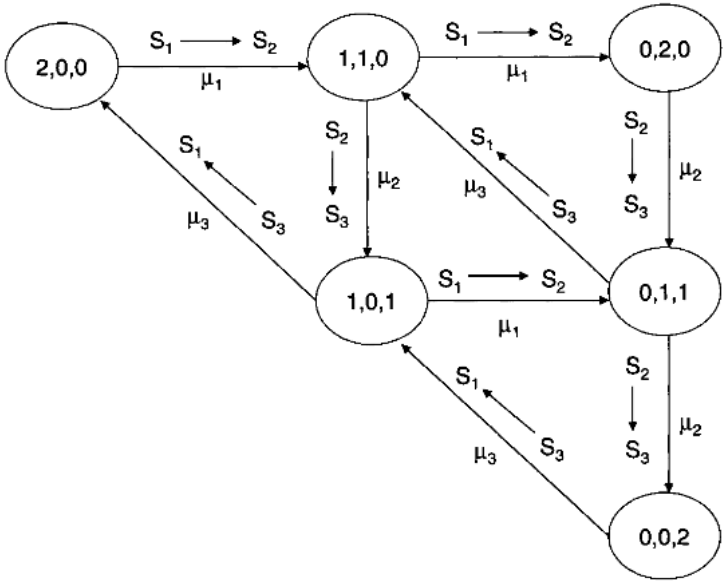
$$\pi(۱,۱,۰)(\mu_۱ + \mu_۲) = \pi(۲,۰,۰)\mu_۱ + \pi(۰,۱,۱)\mu_۳ \qquad (۵۳-۷)$$

$$\pi(۰,۲,۰)\mu_۲ = \pi(۱,۰,۰)\ \mu_۱ \qquad (۵۴-۷)$$

$$\pi\ (۱,۰,۱)\ (\mu_۳ + \mu_۱) = \pi(۱,۱,۰)\ \mu_۲ + \pi(۰,۰,۲)\ \mu_۳ \qquad (۵۵-۷)$$

$$\pi(۰,۱,۱)\ (\mu_۲ + \mu_۳) = \pi(۰,۲,۰)\ \mu_۲ + \pi\ (۱,۰,۱)\mu_۱ \qquad (۵۶-۷)$$

$$\pi(۰,۰,۲)\ \mu_۳ = \pi(۰,۱,۱)\mu_۲ \qquad (۵۷-۷)$$



شکل (۷-۱۴): نمودار نرخ انتقال حالت برای یک سیستم بسته ساده

زمانی که:

$$\pi(N_۱, N_۲, N_۳) = \text{احتمال حالت های } \{N_۱, N_۲, N_۳\}$$

برقرار باشد، در ذهن داشته باشید که مجموع همه احتمالات حالت باید برابر ۱ باشد، این شبکه یک راه‌حل دارد:

$$\pi(N_1, N_2, N_3) = K\,(1/\mu_1)^{N_1}\,(1/\mu_2)^{N_2}\,(1/\mu_3)^{N_3} \qquad (۵۸-۷)$$

در جایی که K یک ثابت نرمال‌سازی برای تضمین مجموع احتمال به یک است، داریم:

$$K = \frac{1}{\sum_{i=0}^{2}\sum_{j=0}^{2}\sum_{k=0}^{2}\pi(i,j,k)} \qquad (۵۹-۷)$$

حال، با استفاده از معادلات (۷-۵۲) تا (۷-۵۹)، و این حقیقت که مجموع همه احتمالات برابر یک است، می‌توانیم معادلات تعادل جریان را برای احتمالات هر حالت تکی حل کنیم. زمانی که دارای احتمال حالت هستیم، می‌توانیم طول مورد انتظار را در هر سرور بیابیم، به شرح زیر :

$$E[N_q] = \sum_{i=1}^{\text{تعداد حالات}} iP\,[N = i \text{ at queue K}] \qquad (۶۰-۷)$$

به دلیل اینکه احتمال حالت در صف‌ها مشابه با سیستم‌های صف در شبکه ایزوله نیست، ما نمی‌توانیم زمان انتظار مورد انتظار را در یک صف با ضرب تعداد مشتریان در زمان سرویس‌دهی در صف به دست آوریم. در عوض، ابتدا بر محاسبه توان عملیاتی برای هر سیستم صف تمرکز می‌کنیم و سپس از نتایج Little با استفاده از توان عملیاتی به‌عنوان یک معیار نرخ ورود در صف خاص استفاده می‌کنیم. بنابراین، توان عملیاتی در یک صف خاص می‌تواند با ضرب احتمال داشتن مشتریان در آن صف (برای مثال سرور مشغول است) ضرب در نرخ سرویس‌دهی مورد انتظار به دست آید:

$$\lambda_i = P(\text{server is busy})\,\mu \qquad (۶۱-۷)$$

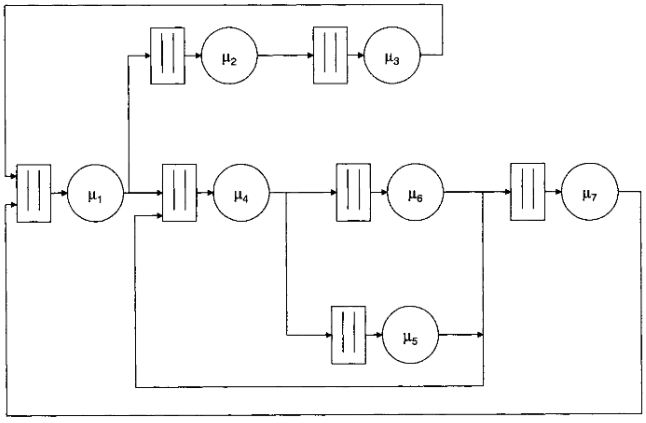
حال، با استفاده از نتایج Little، می‌توانیم زمان سپری‌شده در هر صف را با یک مشتری در صف مربوطه به دست آوریم:

$$W_q = E[N_k] / \lambda_i \qquad (۶۲-۷)$$

مجموع زمان مورد انتظار رفت‌وبرگشت برای یک مشتری در سیستم را می‌توان با جمع‌بندی همه زمان‌های انتظار صف به دست آورد. این می‌تواند مستقیماً با استفاده از نتایج Little و میانگین توان عملیاتی برای سیستم حاصل شود. بنابراین، برای دو مشتری، داریم:

$$W_q = ۲ / \lambda_{avg} \qquad (۶۳-۷)$$

سپس، بیایید یک شبکه بسته دلخواه را با M صف و N مشتری بیابیم. فرض کنید که همه سرورها دارای توزیع زمانی سرویس نمایی خود هستند. برای سادگی بحث، شبکه در شکل (۷-۱۵) یک شبکه دلخواه ما را نشان می‌دهد. بیایید یک احتمال انشعاب را به‌عنوان احتمال داشتن هر مشتری در یک شاخه خاص در زمانی که به نقطه انشعاب می‌رسد تعریف کنیم.



شکل (۷-۱۵): سیستم بسته دلخواه

بنابراین، داریم:

P_{ij} = احتمال اینکه یک مشتری از سرور i خارج شود می به صف J می رود (۶۴-۷)

برای هر سرور i:

$$\sum_{all\ j} P_{ij} = ۱ \qquad (۶۵-۷)$$

برای حفاظت از جریان در سیستم نیاز است که رابطه زیر برقرار باشد:

$$\lambda_j = \sum_{i=1}^{M} \lambda_i P_{ij} \quad (۶۶-۷)$$

توان عملیاتی یک سرور، i، را به شکل زیر تعریف کنید :

$$B(j) = \sum_{i=1}^{M} B(i) P_{ij} \quad (۶۷-۷)$$

ازآنجایی‌که عبارت B نسبی است، می‌توانیم به‌صورت دلخواه یکی از آن‌ها را برابر یک تنظیم کنیم و باقی را حل کنیم. زمانی که همه عبارات را داشته باشیم، احتمال حالت پایدار به‌صورت زیر است:

$$P(N_1, N_2, N_3, N_4, \dots, N_M) = K \prod_{i=1}^{M} \left(\frac{B(i)}{\mu_i}\right)^{N_i} \quad (۶۸-۷)$$

معادله (۶۷-۷) می‌تواند با فرض اینکه حفاظت از جریان برای یک حالت خاص صورت می‌گیرد و سپس با حل سیستم معادلاتی که در مثال قبل داشتیم، به دست آید. حال، حالت S، را در نظر بگیرید:

$$S = (K_1, k_2, k_3, \dots . k_M) \quad (۶۹-۷)$$

و اثر ورود و خروج مشتری از صف J را بررسی کردیم. حالت دیگر A را تعریف کنید، که با S یکسان است، جز اینکه در صف i یک مشتری بیشتر و در صف J یک مشتری کمتر از S دارد. بنابراین، A یک حالت همسایه S است. ما فرض می‌کنیم که نرخ ورود حالت S با توجه به ورود به صف J با نرخ ترک حالت S با توجه به خروج از J متعادل است. ازآنجایی‌که ممکن است بیشتر از یک حالت A وجود داشته باشد، جایی که بیش از یک مشتری در صف i باشد و یک مشتری کمتر در صف J باشد، باید همه حالات را باحالت S متعادل کنیم. تعادل جریان‌ها از رابطه زیر نتیجه می‌شود:

$$\sum_{i=1}^{M} P[A_i] \mu_i P_{ij} = P[S] \mu_j \quad (۷۰-۷)$$

از معادله (۶۸-۷) داریم:

$$P[A_i] = K \prod_{j=1}^{M} \left(\frac{B(j)}{\mu_j}\right)^{N_j} \frac{B(i)}{\mu_i}$$

$$P[s] = K \prod_{j=1}^{M} \left(\frac{B(j)}{\mu_j}\right)^{N_j} \frac{B(j)}{\mu_j} \qquad (۷۱-۷)$$

آخرین عبارت در هر یک از دو عبارت از داشتن یک مشتری در سرویس در سرور مربوطه ناشی می‌شود. با جایگزینی این عبارات در معادله (۷-۷۰) و ساده سازی آن، داریم:

$$\sum_{i=1}^{M} B(i) P_{ij} = B(j) \qquad (۷۲-۷)$$

که آن چیزی است که در معادله (۷-۶۷) فرض شده است.

حال که معادله (۷-۷۲) را داریم، می‌توانیم مجموعه‌ای از معادلاتی را تولید کنیم که به‌صورت همزمان با تنظیم عبارت B برابر ۱ حل می‌شوند. در یک روش ساده مانند مثال قبل، ثابت نرمال سازی K را یافتیم و بنابراین معادله (۷-۶۸) را برای احتمال حالت پایدار سیستم و برای طول صف مورد انتظار با معادله (۷-۶۰) یافتیم.

اگر یک شبکه بسته را در طول دوره زمانی طولانی بیابیم، عبارت توان عملیاتی نسبی می‌تواند به‌عنوان یک شاخص از تعداد دفعات نسبی که مشتری سرور مربوطه را بازدید می‌کند فرض شود، که نسبت بازدید گفته می‌شود. این تفسیر برای تعیین اینکه کدام سرور بیش از همه استفاده شده است، سودمند است، و به‌عنوان تحلیل گلوگاه شناخته شده است. مقدار نسبی کار انجام‌شده توسط یک سرور، i، را تعریف کنید:

$$\text{Relative work by the server } i = B(i) / \mu_i \qquad (۷۳-۷)$$

این مقدار استفاده نسبی از سرور است، سرور با بالاترین نرخ گلوگاه سیستم را نشان می‌دهد.

۷-۲-۲ شبکه‌های باز

سپس، ما بر کلاس دیگر شبکه‌های صف، آن‌هایی که شامل سورس و حفره هستند، بحث می‌کنیم. ما فرض خواهیم کرد که مشتریان ممکن است از هر سورس خارجی بر طبق فرآیند پواسون که برای صف مناسب است، به صف برسند. ما می‌توانیم به این فرایندهای ورود با منشأ گرفتن از هر فرآیند ورود منحصر به انشعاب‌ها فکر کنیم، هر یک با یک احتمال انشعاب مرتبط در ارتباط هستند. شکل (۷-۱۶) نشان می‌دهد که این فرآیندهای ورودی و یک شبکه باز فرضی با صف M و سرورهای مرتبط را نشان می‌دهد.

ما این نرخ سرویس نمایی را برای همه سرورها در سیستم فرض می‌کنیم. در این مورد، نرخ ورود تجمعی معادل با مجموع همه نرخ‌های ورود تکی است که در ابتدا بحث شد. اگر هر نرخ ورود به شکل زیر تعریف‌شده باشد، نرخ تجمعی به شکل زیر است:

نرخ ورودی صف i برابر است با γ_i (۷۴-۷)

نرخ مجموع یا تجمعی برابر است با:

$$\lambda = \sum_{i=1}^{M} \gamma_i$$ (۷۵-۷)

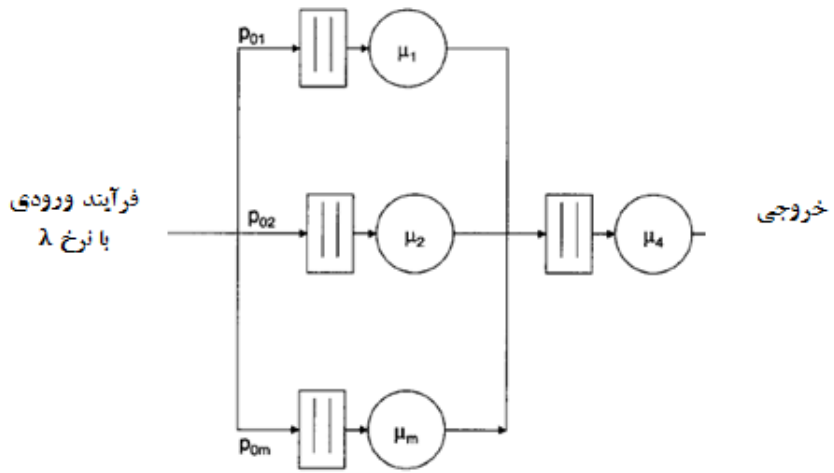
احتمال انشعاب زیر را داریم:

$$P_{.i} = \gamma_i \ / \lambda$$ (۷۶-۷)

مشتریان سیستم را با احتمال تعریف‌شده زیر ترک می‌کنند:

$$P_{i.} = 1 - \sum_{j=1}^{M} P_{ij}$$ (۷۷-۷)

این تعریف بیانگر این است که احتمال یک ترک کار سیستم برابر با مکمل این احتمال است که یک کار در سیستم باقی خواهد ماند.



شکل (۷-۱۶): مدل سیستم باز

با شبکه بسته بحث شده در ابتدا، می‌توانیم یک مجموعه از عبارات توان عملیاتی را فرض کنیم، که با $B\ (i)$ برای هر صف و سرور i نشان داده‌شده است. بنابراین:

$$B\ (i) = \sum_{j=0}^{M} B\ (j)\ P_{ij}$$

$$B\ (i)\ \sum_{j=1}^{M} B\ P_{ij} + \gamma_i \qquad (۷۸-۷)$$

ازآنجایی‌که توان عملیاتی ورود از سمت سورس خارجی را می‌دانیم، می‌توانیم رابطه زیر را داشته باشیم:

$$B\ (0) = \lambda \qquad (۷۹-۷)$$

و عبارت B باقی مانده را حل کنید. در مورد یک شبکه باز، عبارت B یک توان عملیاتی واقعی، و نه نسبی را در سرور، i نشان می‌دهد، چرا که آن‌ها از نرخ ورود تجمعی به‌دست‌آمده اند. به همین دلیل، می‌توانیم بهره وری هر سرور را به شکل زیر تعریف کنیم:

$$U_i = B\ (i)\ /\ \mu_i \qquad (۸۰-۷)$$

بعد از حل همه عبارات $B\ (i)$، احتمال حالت پایدار به شکل زیر است:

$$P\ (N_1\ , N_2, N_3, N_4\ , \dots , N_M) = K \prod_{i=1}^{M} \left(\frac{B\ (i)}{\mu_i}\right)^{N_i}$$

$$= K \prod_{i=1}^{M} U_i^{N_i} \qquad (۸۱-۷)$$

در این رابطه، K ثابت نرمال سازی است. ما می‌توانیم همه احتمالات حالات را جمع کنیم و برای K رابطه زیر را حل کنیم:

$$K = \prod_{i=1}^{M} (I - U_i) \qquad (۸۲-۷)$$

بنابراین، عبارت برای احتمال حالت پایدار، به رابطه زیر تبدیل شد:

$$P\ (N_1\ , N_2, N_3, N_4\ , \dots , N_M) = K \prod_{i=1}^{M} (1 - U_i)\ U_i^{N_i} \qquad (۸۳-۷)$$

اگر به معادله (۷-۱۷) نگاهی بی اندازیم، می‌بینیم که عبارت حاصل شده حاصل ضرب عباراتی است که می‌تواند با اصلاح هر صف و سرور به‌عنوان یک سیستم صف $M/M/I$ در ایزوله سازی حاصل شود. این نتایج به‌عنوان نظریه جکسون شناخته می‌شود، و بیان می‌کند که، اگر چه نرخ ورود در هر سرور در یک سیستم باز ممکن است پواسون نباشد، اما اگر فرآیند ورود پواسون باشد، می‌توانیم تابع احتمال توزیع را برای تعدادی مشتری در هر صف پیدا کنیم. قضیه جکسون بیان می‌کند که هر سیستم صف در شبکه مانند یک سیستم $M/M/1$، با نرخ ورود تعریف‌شده با عبارت زیر، رفتار می‌کند:

$$\lambda_i = \gamma_i + \sum_{j=1}^{M} P_{ij}\,\lambda_i \qquad (۷-۸۴)$$

که یک معادله ساده (۷-۷۸) است که به عبارت آشناتر تبدیل شده است.

شایان ذکر است که قضیه جکسون بر سیستم‌های بازی دلالت دارد که هر سیستم صف در آن $M/M/C_i$ است. به همین ترتیب، هر سرور ممکن است در واقع از تعداد (i) سرورهای یکسان تشکیل شده باشد. بنابراین، احتمالات حالت پایدار برای هر سیستم صف در شبکه با معادله‌ای برای این سیستمی در ایزوله سازی با نرخ ورود، همان‌طور که در معادله (۷-۸۴) تشریح شده است، داده‌شده است. اثبات کامل قضیه جکسون در [۸] آمده است.

۷-۳ تخمین توزیع‌ها و پارامترها

حال می‌توانیم بر جنبه‌های گوناگون نظریه صف بحث کنیم، ما باید برخی از روش‌هایی را که می‌توانیم مدل‌ها را پارامتر بندی کنیم را انتخاب کنیم. در این بخش، بر روش‌های گوناگونی بحث می‌کنیم که می‌توانند برای تعیین اینکه آیا یک آمار اصلی یا توزیع به‌صورت مناسبی فرآیند مشاهده شده را تشریح می‌کند یا نه، استفاده کنیم. به خصوص، ما تست فرضیه را، تخمین‌هایی برای برخی از آمارها، آزمون‌های برازش نکویی را پوشش خواهیم داد. ما با تست فرضیه کلی آغاز می‌کنیم.

یک آزمون فرضیه یک تکنیک استفاده شده برای تعیین این است که آیا یک عبارت اصلی را در مورد یک پدیده جهان واقعی باور کنیم یا نه و معیارهایی برای حد باور ارائه می‌دهد. یک فرضیه معمولاً در دو بخش بیان می‌شود: ابتدا به آمارها و مشخصه‌هایی مربوط است که در مورد ارزش‌ها فرض می‌کنیم و به آن ربط می‌دهیم که برای آمارها فرض می‌شود. برای مثال، ممکن است فرض کنیم که مقدار

میانگین فرآیند مشاهده شده کمتر از ۱۰ است یا فرآیند مشاهده شده گاوس است. صورت مثبت یک فرضیه معمولاً فرضیه null است و با H_0 نشان داده می‌شود. همراه با فرضیه null یک جایگزین هست، که با H_1 نشان داده می‌شود. ایده در اینجا داشتن دو فرضیه مکمل است که تنها یک مورد به‌عنوان محتمل انتخاب می‌شود. دو فرضیه، H_0 و H_1، مبنایی را برای متدلوژی تست فرضی نشان داده‌شده در گراف زیر تشکیل می‌دهد.

یک تست فرضی معمولاً در چهار مرحله عمومی اجرا می‌شود، که به پذیرش یا رد فرضیه اولیه منجر می‌شود. اولین مرحله فرمول کردن فرضیه‌های null ،H_0 و فرضیه جایگزین H_1 است. سپس، بر یک آماره برای تست تصمیم می‌گیریم. متغیرها معمولاً میانگین یا واریانس نمونه هستند. سوم، یک مجموعه از نتایج برای تست آماره انتخاب می‌شود لذا نتایج آماره تست در مجموعه با احتمال خاص قرار می‌گیرد، با توجه به اینکه H_0 درست است. به همین ترتیب، اگر H_0 درست باشد، ما می‌گوییم که مقدار آماره تست درون مجموعه منتخب (گاهی منطقه بحرانی می‌گویند) با احتمال P قرار گرفته است (سطح اهمیت تست گفته می‌شود). هدف انتخاب منطقه بحرانی به‌طوری‌که مقدار آماره تست درون منطقه کوچک، معمولاً بین ۰,۰۱ و ۰,۰۵ قرار گیرد، است. یک وقوع این رخداد، سپس، نشان می‌دهد که فرضیه H_0 یک انتخاب خوب نیست و باید رد شود. در مقابل، می‌توانیم احتمال بزرگی را انتخاب کنیم، در مواردی که وقوع رخداد نشان می‌دهد که فرضیه null باید پذیرفته شود تقریباً ۰,۹ است. مرحله نهایی، فرآیند جمع آوری داده نمونه و محاسبه آماره تست است.

مسئله بلافصل بعدی برای اجرای یک تست فرضیه، تعریف عبارتی است که متغیرهای نمونه‌ای را تشریح می‌کند که مورد توجه هستند. این موارد معمولاً برآوردگر گفته می‌شوند، چرا که آن‌ها متغیرهایی را تخمین می‌زنند که می‌تواند از یک توزیع به دست آید که دقیق فرآیند واقعی را مدل می‌کند. متداول‌ترین برآوردگرهای استفاده شده میانگین نمونه واریانس نمونه است.

به‌منظور محاسبه آماره نمونه، باید ابتدا یک نمونه تصادفی را از جمعیت آزمایشی به دست آوریم. یک نمونه تصادفی تعریف‌شده در اینجا یک توالی از مشاهدات فرآیندهای جهان واقعی است، جایی که هر مقدار مشاهده‌شده دارای یک احتمال برابر انتخاب‌شده است و مشاهدات از دیگر موارد در نمونه مستقل هستند. بنابراین، یک نمونه تصادفی یک دنباله از متغیرهای تصادفی است که مستقل هستند و به‌صورت یکسانی توزیع‌شده اند.

برای یک نمونه تصادفی با اندازه n، که n تعداد نمونه‌های حاصل شده است، میانگین نمونه به شکل زیر تعریف‌شده است:

$$\overline{X} = \frac{1}{n} \sum_{i=1}^{n} X_i \qquad (۸۵-۷)$$

واریانس نمونه به شکل زیر تعریف‌شده است:

$$S^2 = \frac{1}{n-1} \sum_{i=1}^{n} (X_i - \overline{X})^2 \qquad (۸۶-۷)$$

انحراف معیار نمونه مانند انحراف معیار یک توزیع تعریف‌شده است و در اینجا به شکل زیر تکرار شده است:

$$S = \sqrt{S^2} \qquad (۸۷-۷)$$

در سه عبارت بالا، متغیر تصادفی X_i i امین مشاهده در نمونه تصادفی است.

حال می‌توانیم متغیرهایی را برای یک نمونه تصادفی برخی از پدیده‌ها بیابیم، حال می‌توانیم این تخمین‌ها را به آمارهای واقعی فرآیند اصلی ربط دهیم؟ به همین ترتیب، ما از یک نظریه شناخته شده به نام نظریه نمونه‌گیری استفاده می‌کنیم. این بیان می‌کند که، برای یک نمونه تصادفی، همان طور که از قبل تشریح شد، با یک میانگین محدود، میانگین نمونه و مقدار مورد انتظار می‌تواند معادل باشد و از طرفی واریانس نمونه واریانس معادل هستند. به همین ترتیب، متغیرهای نمونه برآوردگرهای سازگار و بدون بایاس دیده می‌شوند. نظریه نمونه‌گیری نیز چندین رابطه مهم دیگر را بیان می‌کند، از جمله عبارت مربوط به واریانس میانگین نمونه واریانس متغیر تصادفی که تشریح کننده فرآیند است. این عبارت به‌صورت زیر است:

$$\mathrm{Var}[\overline{X}] = \mathrm{Var}[X] / n \qquad (۸۸-۷)$$

بیان می‌کند که اندازه نمونه هرچقدر بزرگ‌تر شود، واریانس میانگین نمونه کوچک‌تر می‌شود، که نشان‌دهنده این است که به میانگین حقیقی X نزدیک‌تر است.

این تخمین‌ها به سؤال دیگری منجر می‌شود: تا آنجا که می‌دانیم (یا فکر می‌کنیم می‌دانیم) انواع توزیع نمونه تصادفی وجود دارد، چگونه پارامترهای این توزیعی را از داده نمونه تصادفی تخمین بزنیم؟ برای انجام این کار دو روش به‌صورت متداول استفاده شده: روش لحظه و روش تخمین حداکثر احتمال است.

روش لحظه زمانی که فکر می‌کنیم توزیع نمونه را می‌دانیم اما نمی‌دانیم که چه پارامترهای توزیعی وجود دارد، سودمند است. توزیعی را فرض کنید که تخمین پارامترهای آن برابر n پارامتر باشد. در این روش، ما ابتدا n لحظه توزیع را می‌یابیم، که در فصل ۵ تشریح شده است.در این روش، n لحظه توزیع اولیه را پیدا می‌کنیم، همان‌طور که در فصل ۵ تشریح شده است. سپس، K لحظه نمونه اول را پیدا می‌کنیم و برابر با نتایج لحظهی که قبلاً پیدا کردیم قرار می‌دهیم. از این دیدگاه ما n معادله با n مجهول داریم، که می‌توانند همزمان برای پارامترهای مطلوب حل شوند. ما k امین لحظه نمونه را برای یک اندازه نمونه m مانند زیر حاصل کردیم:

$$M_k = \frac{1}{n} \sum_{i=1}^{n} X_i^k \qquad (۸۹-۷)$$

در اینجا x_i نقطه نمونه i در نمونه تصادفی است.

در تخمین حداکثر احتمال، ما برای اتخاذ پارامترهای توزیعی تلاش می‌کنیم که مقادیر مشاهده‌شده را در نمونه تصادفی حداکثر می‌سازد. برای انجام این کار، ما ابتدا آنچه که تابع احتمال گفته می‌شود را تشکیل می‌دهیم. سازگار با مقادیر فرض شده توسط تابع توزیع احتمال در نقاط مشاهده‌شده، در نمونه تصادفی نیز این توزیعی رخ می‌دهد. این تابع، برای یک متغیر تصادفی پیوسته که توزیع آن‌ها تنها یک پارامتر دارد به‌صورت زیر است:

$$L(\theta) = f(x_1) f(x_2) f(x_3) \dots f(x_m) \qquad (۹۰-۷)$$

برای یک متغیر تصادفی که توزیع آن n پارامتر دارد، ما n معادله داریم، که مشابه با معادله (۹۰-۷) است. سپس حداکثر هر معادله را با توجه به هر پارامتر پیدا می‌کنیم. سرانجام، مجموعه n پارامتر در n مجهول می‌تواند برای پارامترهای لازم حل شود.

حال ما چندین روش مطرح‌شده برای تخمین متغیرهایی برای یک توزیع داریم که فرآیندهای جهان واقعی را تشریح می‌کند، ما توجه خود را به قابلیت اطمینان تخمین جلب می‌کنیم. یک معیار این قابلیت اطمینان را بازه اطمینان گویند است. یک بازه اطمینان به‌عنوان طیفی از مقادیر تعریف می‌شود، که بر تخمین اماره مورد نظر متمرکز است، جایی که مقدار واقعی آماره در احتمال ثابت قرار می‌گیرد. برای مثال، یک بازه اطمینان ۹۰ درصد برای میانگین یک متغیر تصادفی، بر اساس نمونه مشخص ممکن است با مقادیری درون یک فاصله، r، میانگین تخمین زده شده، تعریف می‌شود. در این مورد، r انتخاب شده است به طوری که کسری از زمانی که میانگین واقعی درون بازه قرار می‌گیرد ۹۰ درصد است. رویه

عمومی برای تعریف یک بازه اطمینان نیازمند ساختار یک توزیع شناخته شده، C، از تخمین متغیرهای تخمین زده شده است. سپس، ما، بازه را به‌صورت زیر در نظر می‌گیریم:

$$P\,(a < C < b) = z \qquad (۹۱-۷)$$

در این معادله z سطح اطمینان مطلوب است. سرانجام، ما C را با استفاده از مقدار X_i ارزیابی می‌کنیم، به طوری که رابطه زیر برقرار باشد:

$$a < C\,(X_i) < b \qquad (۹۲-۷)$$

می‌توانیم به‌صورت جایگزینی C را برای نقاط X_a و X_b حل کنیم، جایی که $C\,(x_a) = a$ و $C\,(X_b) = b$ برقرار است. این‌ها نقاط انتهایی ۱۰۰ بازه اطمینان هستند.

این رویه فرض می‌کند که ما توزیع C را قبل از اینکه بازه اطمینان را بیابیم می‌دانیم. اگر این مورد برقرار نباشد، و اندازه نمونه بزرگ باشد، می‌توانیم فرض کنیم که توزیع نمونه نرمال است و می‌تواند بازه اطمینان قابل اعتمادی را برای مقدار میانگین حاصل کند. در این مورد، ما ابتدا آماره زیر را تشکیل می‌دهیم:

$$T = (\overline{X} - \mu) \,/\, \left(\sigma \sqrt{n}\right) \qquad (۹۳-۷)$$

از آنجایی که X نرمال فرض می‌شود، T در این مورد با میانگین ۰ و انحراف معیار ۱ نرمال است. بار دیگر، ما درصد بازه اطمینان و عوامل تعیین کننده a و b را تعریف می‌کنیم، به طوری که رابطه زیر برقرار باشد:

$$P\,(a < T < b) = z \qquad (۹۴-۷)$$

بازه اطمینان مطلوب برای میانگین سپس با رابطه زیر داده می‌شود:

$$(\overline{X} - b)\,\sigma \,/\, \sqrt{n} \;< \mu < (\overline{X} + a)\,\sigma \,/\, \sqrt{n} \qquad (۹۵-۷)$$

بازه اطمینان برای واریانس در زمانی که توزیع جمعیت ناشناخته است با استفاده از روش‌های تشریح شده قبلی یافت می‌شود، اگر چه نتایج ضعیف خواهد بود اگر توزیع جمعیت فعلی از نرمال فاصله داشته باشد.

حال ما چندین تکنیک را برای تخمین پارامترهای توزیع کشف کردیم، ما به روش‌هایی برای یافتن توزیعی که با داده نمونه متناسب است نگاهی انداختیم. معمولاً ما میانگین نمونه و انحراف معیاری را پیدا می‌کردیم و حال می‌خواهیم متغیر تصادفی را پیدا کنیم که به اندازه کافی جمعیت نمونه را نشان می‌دهد. تست استفاده شده در اینجا معمولاً آزمون برازش نکویی گفته می‌شود. ما بر دو تست بحث خواهیم کرد، تست chi-square (آزمون مربع کای-دو) و تست Kolmogorov-Smirnov. این آزمون‌ها تحت عنوان کلی تست فرضیه قرار می‌گیرد، و بنابراین ما از فرضیه مشابهی برای تشکیل تکنیک تشریح شده در ابتدا استفاده می‌کنیم. در هر دو تست، ما با فرضیه null شروع می‌کنیم که دارای یک توزیع اصلی است، و سپس متغیرهایی را به دست می‌اورد که نشان می‌دهد که آیا ما باید فرضیه null را بپذیریم یا نه.

در آزمون مربع کای - دو، ما تعیین می‌کنیم که آیا توزیع فرضیه null به خوبی با جمعیت در مقایسه با گروه نمونه جمع‌آوری‌شده با آنچه که توزیع فرضی است متناسب است یا نه. امید می‌رود که بتوانیم $k\ bins$، $B_1, \dots, B_k$ را پیدا کنیم، لذا هر مقدار در نمونه تصادفی در یک، فقط، یک bin قرار می‌گیرد. بعد از یافتن یک مجموعه متناسب از bin ها، ما نمونه را به آن‌ها تقسیم می‌کنیم و تعداد نمونه‌هایی که در هر مورد قرار می‌گیرند را ثبت می‌کنیم. سپس، تعداد متناظر نمونه‌ها از توزیع جمعیت فرضی را در نظر می‌گیریم و آن‌ها را به یک bin مشابه تخصیص می‌دهیم. اگر هر مجموعه دوم نمونه (مواردی که از توزیع در نظر گرفته شده اند) در قرار گرفتن فقط یک bin با شکست مواجه شوند، ما مجموعه مناسبی از binها را انتخاب نمی‌کنیم و باید مجموعه دیگری را انتخاب کنیم. برای هر نوع توزیعی که ما تست کردیم، پارامترهای توزیع مناسب با استفاده از یک تکنیک تخمین تشریح شده اولیه می‌تواند یافت شود. با ادامه تست، حال آماره زیر را محاسبه می‌کنیم:

$$C = \sum_{i=1}^{K} \frac{(NS_i - ND_i)^2}{ND_i} \qquad (۹۶-۷)$$

در این رابطه NS_i به تعداد عناصر در bin_i با توجه به نمونه تصادفی اشاره دارد، و ND_i تعداد در bin_i با توجه به توزیع فرض شده است. مبنای این تست متغیرهای معادله (۹۶-۷) است که دارای یک توزیع مربع کای-دو است. درجه آزادی مربع کای-دو به‌عنوان یک مورد کمتر تعداد نمونه bin ها منهای تعداد پارامترها در توزیع فرضی تعریف‌شده است:

$$M = k - 1 - \text{تعداد پارامتر ها} \qquad (۹۷-۷)$$

سپس، بر سطح اهمیتی تصمیم می‌گیریم که می‌خواهیم آن را تست کنیم. با استفاده از عملیات زیر، می‌توانیم تابع چگالی احتمال را برای توزیع مربع کای با n درجه آزادی محاسبه کنیم:

$$f_x(x) = \begin{cases} \left(1/\left((n/2)-1\right)!\right)\left(2^{-n/2}\right)\left(x^{(n/2)-1}\right)\left(e^{-e(x/2)}\right) \text{ for } x > 0 \\ 0 \quad \text{در غیر اینصورت} \end{cases}$$

(۹۸-۷)

مرحله نهایی برای یافتن مقدار X برای انتگرال با توجه به x معادله (۷-۹۸)، که از X تا بی‌نهایت ارزیابی شده است، برابر با سطح اهمیت مطلوب است. تست نهایی بیان می‌کند که اگر مقدار x بزرگ‌تر برابر آماره مربع کای محاسبه شده در معادله (۷-۹۶) باشد، توزیع فرض شده برازش نکویی در سطح اهمیت مطلوب نیست. به همین ترتیب، ما فرضیه null را رد می‌کنیم اگر رابطه زیر برقرار باشد:

$$X \geq C \qquad (۹۹-۷)$$

یک رویکرد متداول برای آزمون مربع کای تشکیل مقدار $C - \varepsilon$ است، در جایی که ε یک مقدار کوچک است. ما سپس از نتایج برای یافتن احتمال x استفاده می‌کنیم که بزرگ‌تر از $C - \varepsilon$ است. احتمال حاصل به ما شاخصی را برای تقریب سطح یا اهمیتی می‌دهد که ممکن است فرضیه null را بپذیرد. چندین مرجع جداولی را برای مقادیر مهم توزیع مربع کای دادند. این جدول‌ها ممکن است برای محاسبه مقدار توزیع استفاده شوند.

دیگر آزمون برازش نکویی آزمون Kolmogorov- Smirnov است. آزمون بر اساس ترتیب بزرگی نمونه، محاسبه حداکثر تفاوت بین نقاط نمونه و توزیع فرض شده، تعیین سطح برازش توزیع فرض شده است. یک تشریح رسمی از این تست به نظر می‌رسد که در تعدادی از متون وجود داشته باشد. در اینجا یک رویکرد بصری تر را توضیح خواهیم داد، که تا حدی آزمایش آن ساده تر خواهد بود.

همانطور که در ابتدا گفته شد، اولین مرحله این آزمون مرتب سازی مقادیر نمونه در یک ترتیب صعودی بر طبق بزرگی است. برای هر نقطه x_i در نمونه مرتب شده، کسر، f_i، تعداد مجموع نمونه‌ها است که کمتر از مقدار داده‌شده است. سپس، برای توزیع فرض شده، مقدار، k_i را پیدا می‌کنیم، که کسر مشابه، f_i را برای تعداد نمونه مشخصی حاصل می‌کند. سرانجام، ما k_i را در برابر x_i برای همه i ها پیدا می‌کنیم. نمودار نتیجه شده برازش نکویی را نشان می‌دهد اگر داده تقریباً یک خط راست با شیب یکنواخت باشد.

اگر برازش خط راستی با شیب غیر یکنواخت باشد، پارامترهای توزیع فرض شده ممکن است برای کسب نتایج مطلوب دچار تغییر شوند. در غیر این صورت، ما باید برای توزیع مفروض دیگری تلاش کنیم.

۴-۷ روش‌های محاسباتی برای راه حل‌های شبکه صف

در فصل ۵ و فصل ۶ و بخش‌های گذشته این فصل، ما نظریه احتمال و تکنیک تحلیل را برای اجرای تحلیل سیستم صف کلاسیک معرفی کردیم. این تحلیل‌ها، حتی برای سیستم‌های ساده هم به سمت پیچیدگی می‌روند. در تلاش برای اصلاح این شرایط، سه روش تحلیل جایگزین ظاهر شدند.

اولین روش، از Buzen [۹] است که یک تکنیک برای یافتن ثابت نرمال سازی ارائه می‌دهد که برای راه حل حاصل اصلی شبکه‌ها نیاز است. روش به راه حل نرمال سازی جمع تشریح شده در فصل ۵ نیازی ندارد. در عوض، این از یک راه حل تکراری استفاده می‌کند، که ساده تر اجرا می‌شود.

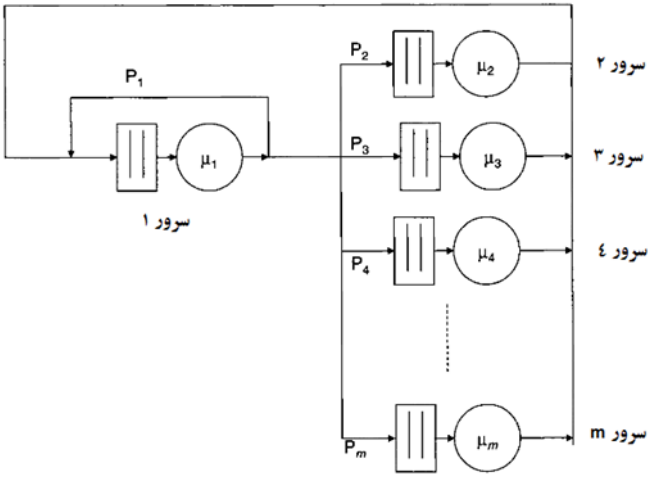
روش بعدی، از Buzen و Dening [۱۰]، متدلوژی را برای ارزیابی تطبیق فرضیات مشخص برای سیستم تحت تحلیل معرفی کردند. علاوه بر این، آن‌ها مقدار مورد نظر عملکرد را ازنظر روابط عملیاتی آن‌ها در سیستم تحت بررسی تعریف کردند. به همین دلیل، این نوع تحلیل به‌عنوان تحلیل عملیاتی شناخته شده است.

روش تحلیل سوم، از Reiser و Lavenberg [۱۱]، برای ساده سازی تحلیل شبکه‌های صف تلاش کردند. با استفاده از میانگین زمان انتظار و میانگین اندازه صف، در ارتباط با نتایج Little، راه حل یک سیستم صف‌ها می‌تواند به‌صورت مجانبی به راه حل دقیق نزدیک شود، با محاسبات موردنیاز ساده تر. این نوع تحلیل را تحلیل مقدار میانگین گویند.

در این بخش، ما بر این روش‌ها و مدل‌ها بحث می‌کنیم. برخی از نتایج برای نوع خاص مدل استفاده شده خاص هستند، در حالی که سایر موارد عمومی هستند. انواع مدل‌های خاص، می‌توانند برای تقریب سیستم مشخص یا بخشی از سیستم و برای کسب احساس اولیه از رفتار واقعی سیستم استفاده شوند.

۷-٤-۱ مدل سرور مرکزی

مدل سرور مرکزی، نشان داده‌شده در شکل (۷-۱۷)، در اصل به‌عنوان مدلی برای کارها در کامپیوترهای چند برنامه‌ای پیشنهاد شدند. این یک شبکه بسته است و فرض می‌کنیم که تعداد ثابت (K) کارها همیشه در پردازش هستند. در مدل اصلی، برنامه‌ها با CPU سرویس‌دهی می‌شوند (سرور ۱) و سپس به یکی از $M-1$ I/O دستگاه مسیر یابی می‌شود (سرور ۲ از طریق M). بعد از دریافت سرویس I/O، برنامه برای زمان CPU صف‌هایی را ایجاد می‌کند. اگر یک برنامه زمان اجرا را تکمیل کند، این به صف CPU برای شروع کار دیگر مسیر یابی می‌شود. در نتیجه تعداد کارها در سیستم برابر K باقی می‌ماند.



شکل (۷-۱۷): مدل سرور مرکزی

این می‌تواند سیستمی باشد که در آن همیشه یک کار در حال انتظار برای ورود به سیستم در صف CPU وجود دارد، اما نمی‌تواند تا زمان تکمیل کار انجام شود. کارهای واقعی در سیستم، ممکن است در طول زمان تغییر کند، اما تعداد در برنامه در هر زمان مشخص ثابت باقی می‌ماند.

مدل سرور مرکزی می‌تواند برای نمایش سیستم‌های دیگر در کنار یک CPU و دستگاه‌های I/O مربوطه پذیرفته شود. برای نمونه، می‌توانیم سرور ۱ را برای نمایش کنترلر DMA چند کاناله و سرور

۲ را از طریق M برای نمایش کانال خروجی نشان می‌دهد. یا، می‌توانیم احتمال انشعاب را برای نمایش یک سیستمی که در آن کارهای ثابت می‌توانند و هیچ گاه تکمیل نمی‌شوند تنظیم شود. $(i.e., P_1 = 0)$. این می‌تواند برای یک سرور I/O اختصاصی سودمند باشد. به روش دیگر، می‌توانیم یکی از سرورهای ۲ را از طریق M برای نمایش $idle$ برای یک کار یا کانال I/O انتخاب کنیم.

اگر چه ممکن است قادر به فرمول کردن مدل سرور مرکزی باشیم که تا حدی شرایط واقعی را منعکس می‌کند، تطبیق دقیق نخواهد بود. مزیت این مدل، به هر حال، سادگی محاسباتی بسیاری از پارامترهای کارایی مهم است. سپس، مدل محاسباتی را برای این شبکه صف توسعه می‌دهیم.

در مدل سرور مرکزی، سرورها فرض می‌شود که دارای توزیع زمان سرویس نمایی است. همانطور که در شکل (۷-۱۷) نشان داده شد، در سرور مرکزی چندین انشعاب وجود دارد، که هریک با احتمال انشعاب مربوطه، P_i هستند. مجموع k مشتری (کار) در سیستم در هر زمانی وجود دارد. اجازه دهیم حالت سیستم را تعریف کنیم:

$$S = (k_1, k_2, \dots, k_m) \qquad (۷-۱۰۰)$$

در اینجا k_i به تعداد مشتریان در صف i اشاره دارد. بنابراین داریم:

$$\sum_{i=1}^{M} k_i = k \qquad (۷-۱۰۱)$$

اگر $B(i)$ را به‌عنوان احتمال رفتن به سرور i بعد از سرویس‌دهی به سرور ۱ انتخاب کنیم، اجازه می‌دهیم که $B(1) = 1$ برقرار باشد، سپس:

$$B_i = P_i \text{ , for } i = 2, 3, \dots, M \qquad (۷-۱۰۲)$$

با استفاده از تکنیک‌های مشابهی که قبلاً برای شبکه‌های صف بندی تشریح شد، می‌توانیم احتمال حالت را به شکل زیر به دست اوریم:

$$P(k_1, k_2, \dots, k_M) = \text{norm} \prod_{i=1}^{M} \left(\frac{P_i \mu_i}{\mu_i}\right)^{K_i} \qquad (۷-۱۰۳)$$

این معادله برای احتمال حالت را فرم ضرب گویند، و می‌تواند با یافتن ثابت نرمال سازی، نرم، که در ابتدا بخش نشان داده‌شده است، حل شوند. روش‌های تشریح شده، به راه حل M معادله هم زمان نیاز دارند. یک روش جایگزین با محاسبات کمتر به شکل زیر است:

فرض کنید:

$$G(k) = \sum_{\text{all states}} \frac{P(k_1,k_2,k_3,\ldots k_M)}{\text{norm}} = \frac{1}{\text{norm}} \qquad (۱۰۴-۷)$$

$$G(k) = \sum_{\text{all states}} \prod_{i=1}^{M} \left(\frac{p_i \mu_i}{\mu_i}\right)^{k_i} \qquad (۱۰۵-۷)$$

برخی از مجموع‌ها شامل حالاتی هستند که در معادله (۷-۱۰۱) برقرار است. این، ما تصریح می‌کنیم که $k_i \geq 0$ برقرار است. تابع دیگر $g(k,m)$ را تعریف کنید در جایی که m صف در سیستم M وجود دارد. پس رابطه زیر درست است:

$$g(k,m) = G(k) \qquad (۱۰۶-۷)$$

در اینجا k مجموع تعداد مشتریان در سیستم با M صف است. بنابراین، می‌توانیم $g(k,m)$ را به شکل زیر تعریف کنیم:

$$g(k,m) = \sum_{\text{تمام حالات}} \prod_{i=1}^{M} \left(\frac{p_i \mu_i}{\mu_i}\right)^{k_i} \qquad (۱۰۷-۷)$$

می‌توانیم جمع سمت راست را به شکل زیر تجزیه کنیم:

$$g(k,m) = \sum_{\substack{\text{تمام حالات} \\ \text{با } k_m>0}} \prod_{i=1}^{M} \left(\frac{p_i \mu_i}{\mu_i}\right)^{k_i} + \sum_{\substack{\text{all states} \\ \text{with } K_m=0}} \prod_{i=1}^{M} \left(\frac{p_i \mu_i}{\mu_i}\right)^{k_i} \qquad (۱۰۸-۷)$$

برای اولین جمع در معادله (۷-۱۰۸)، اگر همیشه حداقل یک مشتری در صف m داشته باشیم، می‌توانیم فرض کنیم سیستم $k-1$ مشتری در m صف دارد. ما باید عبارت ضرب مربوط به مشتریان در صف m را حذف کنیم. به طور مشابه، در جمع دوم، اگر صف m همیشه خالی باشد، سپس می‌توانیم فرض کنیم سیستم $m-1$ صف و k مشتری دارد. بنابراین، معادله (۷-۱۰۸) به رابطه زیر تبدیل می‌شود:

$$g(k,m) = \frac{P_m \mu_1}{\mu_m} \sum_{\substack{\text{تمام حالات} \\ \text{با } k_m>0}} \prod_{i=1}^{M} \left(\frac{p_i \mu_i}{\mu_i}\right)^{k_i} + \sum_{\substack{\text{all states} \\ \text{with } K_m=0}} \prod_{i=1}^{M} \left(\frac{p_i \mu_i}{\mu_i}\right)^{k_i}$$

(۷-۱۰۹)

دو جمع می‌تواند، با استفاده از معادله (۷-۱۰۵) و (۷-۱۰۶) بازنویسی شود:

$$g(k,m) = \frac{P_m \mu_1}{\mu_m} g(k-1,m) + g(k,m-1) \qquad (۷-۱۱۰)$$

برای $k = 0$، و برای $m = 1$، معادله (۷-۱۱۰) به روابط زیر تبدیل می‌شود:

$$g(0,m) = 1 \quad \text{for } m = 1,2,\ldots,M \qquad (۷-۱۱۱)$$

$$g(k,1) = 1 \quad \text{for } k = 0,1,\ldots,k \qquad (۷-۱۱۲)$$

ما می‌دانیم که یک مجموعه شرط اولیه ([۷-۱۱۱]، [۷-۱۱۲]) و روابط بازگشتی، معادله (۷-۱۱۰)، برای محاسبه مقادیر تا $g(K,M) = G(K)$ داریم و سپس می‌توانیم معادله (۷-۱۰۳) و (۷-۱۰۴) را برای محاسبه احتمال حالت استفاده کنیم. محاسبات به شرح زیر است:

فرض کنید یک شبکه مشابه با آنچه در شکل (۷-۱۷) نشان داده‌شده است داریم، در آن $M = 3, \mu_1 = 0.9, \mu_2 = 0.5\ \mu_3 = 0.9, p_1 = 0.7, p_2 = 0.2, p_3 = 0.1$ برقرار است. علاوه بر این، فرض کنید که $k = 2$ مشتری در سیستم وجود دارد. از معادله (۷-۱۱۱)، می‌دانیم که»

$$g(0,1) = 1$$

$$g(0,2) = 1$$

$$g(0,3) = 1$$

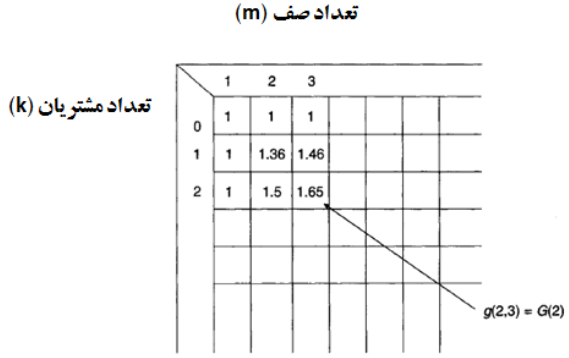
از معادله (۷-۱۱۲) می‌دانیم که:

$$g(0,1) = 1$$

$$g(1,1) = 1$$

$$g(۲,۱) = ۱$$

می‌توانیم این مقادیر را در یک شبکه مرتب کنیم،که در شکل (۷–۱۸) نشان داده‌شده است. محاسبات حال یک ردیف در زمان دارند و با مقداری برای $g(۲,۳) = G(۲)$ پایان می‌یابد.



شکل (۷–۱۸): محاسبه شبکه $G(K)$

برای مثال، برای محاسبه $g(۲,۱)$ ، روابط زیر را داریم:

$$g(۱,۲) = \frac{P_۲ \mu_۱}{\mu_۲} g(۰,۲) + g(۱,۱)$$

$$g(۱,۲) = \frac{(۰.۲)(۰.۹)}{(۰.۵)} (۱) + (۱)$$

$$g(۱,۲) = ۱.۳۶$$

بنابراین، برای شکل (۷–۱۸)، ثابت نرمال سازی برابر ۰/۶ است. با دو مشتری، می‌توانیم احتمال حالت را با استفاده از معادله (۷–۱۰۳) به دست آوریم.

Buzen [۹] عبارات متعددی را برای معیار عملکرد می‌دهند که بر اساس ساختار محاسباتی عمومی است. یکی از این معیارهای بهره وری دستگاه است، U_i برای سرور i است، ما بهره وری دستگاه را به‌عنوان مجموع احتمالات حالات تعریف می‌کنیم که در آن حداقل یک مشتری در سرور i قرار دارد:

$$U_i = \sum_{\substack{\text{تمام حالات} \\ \text{با } k_i>۰}} P\left(k_۱, k_۲, \dots, k_\mu\right) \qquad (۱۱۳-۷)$$

$$U_i = ۱/\, G\,(k) \sum_{\substack{\text{تمام حالات} \\ \text{با } k_i>۰}} \prod_{j=۱}^{m} \frac{(p_j \mu_۱)^{k_j}}{\mu_j} \qquad (۱۱۴-۷)$$

با استفاده از استدلال مشابه، ما برای معادله (۷-۱۰۹)، می‌توانیم معادله (۷-۱۱۴) را به‌عنوان سیستمی با یک مشتری کمتر، ضرب در فاکتوری که همیشه دارای یک مشتری در صف i است، در نظر بگیریم. بنابراین، داریم:

$$U_i = \frac{۱}{G\,(k)} \frac{(p_j \mu_۱)^{k_j}}{\mu_j} \sum_{\substack{\text{تمام حالات} \\ \text{با } k-۱ \\ \text{مشتری}}} \prod_{j=۱}^{m} \frac{(p_j \mu_۱)^{k_j}}{\mu_j} \qquad (۱۱۵-۷)$$

پس:

$$U_i = \frac{p_j \mu_۱}{\mu_j} \frac{G\,(k-۱)}{G\,(k)} \qquad (۷-۱۱۶)$$

ما تا به حال مقادیری را برای $G\,(k)$ و $G\,(k-۱)$ محاسبه می‌کنیم، مانند شکل (۷-۱۸)، لذا بهره وری یک دستگاه سرراست است. از این، می‌توانیم توان عملیاتی دستگاه i را به شکل زیر بیابیم:

$$\lambda_i = U_i \mu_i = P_i \mu_۱ \frac{G\,(k-۱)}{G\,(k)} \qquad (۷-۱۱۷)$$

با برگشت به معادله (۷-۱۱۴)، می‌توانیم معادله (۷-۱۱۶) را برای یافتن احتمال اینکه طول صف در سرور i بزرگ‌تر یا برابر مقادیر n است، بسط می‌دهیم:

$$P(N_i \geq n) = ۱\, G(k) \sum_{\substack{\text{تمام حالات} \\ \text{با } k_i \geq n \\ \text{مشتری}}} \prod_{j=۱}^{m} \left(\frac{p_j \mu_۱}{\mu_j}\right)^{k_j} \qquad (۱۱۸-۷)$$

بنابراین:

$$P(N_i \geq n) = \left(\frac{p_i \mu_۱}{\mu_i}\right)^n \frac{G(k-n)}{G(k)} \qquad (۱۱۹-۷)$$

با استفاده از معادله (۷-۱۱۸) برای موردی که طول صف برابر $n+۱$ است، می‌توانیم احتمال n مشتری در صف i را به‌صورت زیر حاصل کنیم:

$$P(N_i = n) = \left(\left(\frac{P_i\, \mu_۱}{\mu_i}\right) \frac{G(k-n)}{G(k)}\right) - \left(\left(\frac{p_i \mu_۱}{\mu_i}\right)^{n+۱} \frac{G(K-n-۱)}{G(k)}\right) \qquad (۱۲۰-۷)$$

$$P(N_i = n) = \left(\frac{P_i \mu_۱}{\mu_i}\right)^n \frac{۱}{G(k)} \left(g(k-n) - \frac{P_i \mu_۱}{\mu_i}\, G(K-n-۱)\right) \qquad (۱۲۱-۷)$$

حال یک عبارت را برای احتمال داشتن n مشتری در صف استنتاج می‌کنیم، ما از معادله (۵-۶۶) برای کسب عبارتی برای طول صف مورد انتظار استفاده می‌کنیم:

$$E[N_i] = \sum_{j=۱}^{K} jp \quad \text{for } N_i = j$$

$$= \sum_{j=1}^{k} \left(\left(\frac{P_i \mu_1}{\mu_i} \right)^j \frac{1}{G(k)} \left(G(k-j) - \frac{P_i \mu_1}{\mu_i} G(k-n-1) \right) \right)$$

$$= \frac{1}{G(k)} \left(\frac{P_i \mu_1}{\mu_i} \left(G(k-1) - \frac{P_i \mu_1}{\mu_i} G(k-2) \right) \right) \quad (۱۲۲-۷)$$

$$+2 \left(\left(\frac{P_i \mu_1}{\mu_i} \right)^2 \left(G(k-2) - \frac{P_i \mu_1}{\mu_i} G(k-3) \right) \right)$$

$$+ \ldots + K \left(\left(\frac{P_i \mu_1}{\mu_i} \right)^k \left(G(0) - \frac{P_i \mu_1}{\mu_i} G(-1) \right) \right)$$

حال می‌توانیم عبارت را بسط دهیم، و با در ذهن داشتن $G(K < 0) = 0$، داریم:

$$E[N_i] = \frac{1}{G(k)} \sum_{j=1}^{k} \left(\frac{P_i \mu_1}{\mu_i} \right)^j G(k-j) \quad (۱۲۳-۷)$$

تأخیر مورد انتظار یک صف، می‌تواند از نتایج $Little$، با استفاده از توان عملیاتی و مقدار طول صف مورد انتظار یافت شود:

$$E[W_i] = E[N_i] / \lambda_i \quad (۱۲۴-۷)$$

این تکنیک‌ها محاسبات کارآمد متغیرهای مهم را برای مدل‌های نشان داده‌شده در شکل (۷-۱۷) اجازه می‌دهد. سپس، بر روش محاسباتی عمومی‌تر تحلیل مقدار میانگین بحث می‌کنیم.

۷-٤-۲ تحلیل مقدار میانگین

تحلیل تشریح شده تا کنون همه محاسبه‌ای بودند، یا در یک نقطه، عبارت برای توزیع طول صف یا دیگر احتمالات حالت محاسبه شده اند. این روش‌ها به‌صورت مناسبی برای سیستم‌های نسبتاً ساده شامل چند مشتری و سیستم‌های صف قابل قبول هستند. در بحث زیر، ما یک روش تکراری را برای یافتن معیارهای کارایی مورد نظر بدون محاسبه توزیع‌های فوق استفاده کردیم. عیب این روش این است که تحلیل تنها به مقدار میانگین معیار عملکرد اصلی اشاره دارد.

تکنیک‌هایی که به آن توجه داریم برای شبکه‌های صف بسته‌ای استفاده می‌شوند که دارای یک راه حل به فرم ضرب برای احتمالات حالت هستند. راه حل بر اساس این فرض است که یک مشتری، که در سیستم بسته در یک حالت پایدار به صف رسیده است، یک انتظار را در صف تجربه می‌کند که معادل با زمان انتظار حالت پایدار برای آن صف با حذف مشتریان رسیده است. بنابراین، رفتار شبکه با یک مشتری یا بیشتر بر اساس رفتار قبل از ورود آن‌ها است. این فرض به یک الگوریتم تکراری منجر می‌شود که در آن مشخصه عملکرد حالت پایدار برای سیستم با $n+۱$ مشتری از مشخصه‌هایی با n مشتری حاصل شده است، که از یک سیستم با $n-۱$ مشتری، استنتاج شده، و تا یک مشتری پایین می‌آید.

الگوریتم عمومی، سپس، به ما اجازه می‌دهد که مقدار میانگین را برای طول صف، توان عملیاتی، بهره وری سرور و زمان انتظار با یک حالت برای یک مشتری در سیستم و کار بر هر تعداد مشتری آغاز کنیم. نظریه اصلی پشت تحلیل مقدار میانگین بیان می‌کند که میانگین زمان انتظار برای مشتریان در هر سرور در شبکه به راه حل همان شبکه با یک مشتری کمتر مربوط است. در ارتباط با زمان انتظار، ما از نتایج $Little$ برای کسب مجموع توان عملیاتی شبکه استفاده می‌کنیم و سپس آن را به هر سرور منفرد برای اتخاذ میانگین طول صف در هر سرور اعمال می‌شود.

عبارت برای زمان انتظار به یک شبکه با مشتریان کمتر مربوط است که به شکل زیر می‌باشد:

$$W(k) = ۱ / \mu[۱ + N_q(k-۱)] \qquad (۱۲۵-۷)$$

در این رابطه μ میانگین زمان سرویس موردنیاز مشتری در سرور است. مقدار $W(k)$ و $N_q(k-۱)$ به میانگین زمان انتظار برای یک سیستم با K مشتری در صف و میانگین تعداد مشتریان در صف با $k-۱$ مشتری در سیستم به ترتیب اشاره دارد. این عبارت برای سیستم‌هایی با اصول صف اولین ورود اولین خروج با یک سرور تنها، با نرخ ثابت در هر صف برقرار است.

سپس، ما از نتایج $Little$ برای یافتن توان عملیاتی میانگین برای شبکه استفاده می‌کنیم:

$$\lambda(k) = k \,/\, \text{میانگین زمان انتظار} = k \,/\, \sum_{\text{all i}} \emptyset_i \, W_i \, (k) \qquad (۱۲۶-۷)$$

در اینجا $\emptyset$ نرخ ویزیت برای سرور با در نظر گرفتن همه سرورها است. مقدار نرخ ویزیت بعدها در این فصل بحث می‌شود.

سرانجام، ما از نتایج $Little$ برای سرورها برای مقایسه میانگین طول صف استفاده می‌کنیم:

$$N_q \, (k) = \text{میانگین زمان انتظار} \times \text{نرخ ورودی}$$

$$N_q \, (k) = \emptyset_i \, \lambda \, (k) \, W_i \, (k) \qquad (۱۲۷-۷)$$

حالا ما یک عبارت جدید برای میانگین طول صف داریم که در معادله (۷-۱۲۵) بری آغاز تکرار دیگر استفاده می‌شود.

رویه عمومی، سپس، برای آغاز با یک سیستم تهی $(K = ۰)$. تکرار معادله (۷-۱۲۵) از طریق (۷-۱۲۷) تا زمانی که به مقدار مطلوب k برسد، است. برای یک تکرار، ما مقادیری را برای هر سیستم صف در شبکه قبل از دادن آن به معادله بعدی محاسبه می‌کنیم. شکل (۷-۱۹) یک شبکه ساده را برای تشریح تکنیک نشان می‌دهد. در مثال، اگر ما با ۰ مشتری شروع کنیم، ما مقادیر زیر را از معادله (۷-۱۲۵) با استفاده از (۷-۱۲۷) حاصل کردیم. در عبارت زیر، زیرنویس‌ها به جفت صف/سرور اشاره دارند که معیار به آن مربوط است. الگوریتم تکرار کلی به شرح زیر است:

اولین تکرار:

$$W_۱(۱) = ۱/\mu_۱ \, (۱ + ۰) = ۱ \,/\, \mu_۱$$

$$W_۲(۱) = ۱/\mu_۲ \, (۱ + ۰) = ۱/\mu_۲$$

$$W_۳(۱) = ۱/\mu_۳ \, (۱ + ۰) = ۱/\mu_۳$$

$$\lambda(۱) = ۱ / \left(\frac{\emptyset_۱}{\mu_۱} + \frac{\emptyset_۲}{\mu_۲} + \frac{\emptyset_۳}{\mu_۳}\right) \quad (۱۲۸ -۷)$$

$$N_۱(۱) = \emptyset_۱ \lambda(۱)\, W_۱(۱)$$

$$N_۲(۱) = \emptyset_۲ \lambda(۱)\, W_۲(۱)$$

$$N_۳(۱) = \emptyset_۳ \lambda(۱)\, W_۳(۱)$$

تکرار دوم:

$$W_۱(۲) = ۱/\mu_۱ \left(۱ + N_۱(۱)\right)$$

$$W_۲(۲) = ۱/\mu_۲ \left(۱ + N_۲(۱)\right)$$

$$W_{۳۱}(۲) = ۱/\mu_۳ \left(۱ + N_۳(۱)\right)$$

$$\lambda(۲) = ۱ / \left(\emptyset_۱ W_۱(۲) + \emptyset_۲ W_۲(۲) + \emptyset_۳ W_۳(۲)\right) \quad (۱۲۹ -۷)$$

$$N_۱(۲) = \emptyset_۱ \lambda(۲)\, W_۱(۲)$$

$$N_۲(۲) = \emptyset_۲ \lambda(۲)\, W_۲(۲)$$

$$N_۳(۲) = \emptyset_۳ \lambda(۲)\, W_۳(۲)$$

نرخ ویزیت، $\emptyset_i$ به شرح زیر حاصل می‌شود. یک سرور انتخاب کنید و نرخ ویزیت $\emptyset_i$ آن را برابر ۱ قرار دهید. سپس معادله‌ای را که در نرخ ویزیت صف با جستجوی همه صف‌ها سهم دارد، فرمول کنید. معادله نرخ ویزیت، ضرب در احتمال انشعاب مربوطه، در خط ($\emptyset_i$) است. برای هر سیستم صف تا زمانی که m رابطه در m مجهول داشته باشیم ادامه می‌یابد، که در آن m تعداد سیستم‌های صف است. توجه کنید که نرخ‌های ویزیت با میزان نسبی همراه هستند.برای شکل (۷-۱۹)، نرخ‌های ویزیت به شکل زیر محاسبه می‌شود:

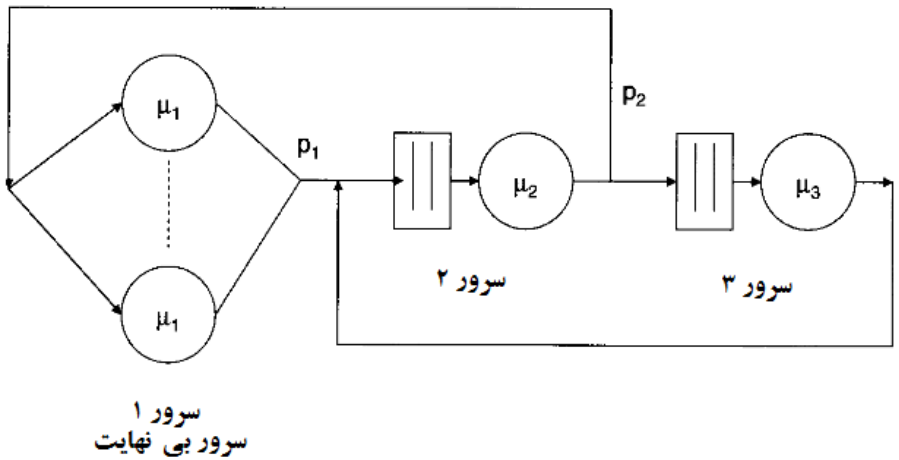
$$\emptyset_1 = 1$$

$$\emptyset_1 = P_2 \emptyset_2$$

$$\emptyset_2 = \emptyset_1 + \emptyset_3 \qquad (7-130)$$

$$\emptyset_3 = P_2 \emptyset_2$$

الگوریتم تا زمانی که به جمعیت شبکه مطلوب برسد تکرار می‌شود، در عین حال میانگین معیار عملکرد را برای شبکه محاسبه می‌کنیم.



شکل (۷-۱۹): شبکه برای تحلیل متغیر میانگین

۷-٤-۳ تحلیل عملیاتی

روش‌هایی برای اجرای تحلیل صف در آغاز فصل ارائه شده است که تقریباً مشابه با سیستم واقعی است. ادعا شده است که، این فرض که روابط و توزیع‌های گوناگون که برای نمایش جهان واقعی استفاده می‌کنیم نمی‌توانند کاملاً دقیق باشند. علاوه بر این، مدل‌های احتمالی مطالعه شده در ابتدا روابطی را حاصل می‌کند که نمی‌توانند در طول هر دو مشاهده اثبات شود.

تحلیل عملیاتی، از طرف دیگر، بر اساس مشاهدات پایه، مقدارهای قابل اندازه گیری هستند که می‌توانند در روابط عملیاتی ترکیب شوند. علاوه بر این، دوره مشاهده برای هر سیستم تحلیل شده محدود است. فرض این است که مقدار پاسخ، یعنی متغیرهای عملیاتی، قابل اندازه گیری هستند. متغیرهای عملیاتی پایه که معمولاً در تحلیل عملیاتی می‌بینیم در زیر آورده شده است:

T= طول دوره مشاهده

A= تعداد ورودی در طول دوره مشاهده

B= زمان مشغول بودن سرور در طول دوره مشاهده

C= تعداد کارهای تکمیل شده در طول دوره مشاهده

علاوه بر مقدارهای قابل سنجش، چندین رابطه مبنا وجود دارد که دیگر مقدارهای سودمند را تعریف می‌کند. آن‌ها به شرح زیر هستند:

$$\lambda = \text{نرخ ورودی} = A/T$$

$$X = \text{سرعت تکامل} = C/T$$

$$U = \text{بهره برداری از سرور} = B/T$$

$$S = \text{زمان سرویس اصلی برای هر کار} = B/C$$

چندین نهاد عملیاتی تعریف‌شده تحت فرض عملیاتی اصلی برقرار هستند. ابتدا، استفاده از سرور مربوط به نرخ تکمیل و میانگین زمان سرویس، که به شکل زیر حاصل شده است:

$$X.S = (C/T)(B/C) = B/T$$

$$X.S = U \qquad (۱۳۱-۷)$$

این روابط در همه موارد برقرار است و به آن قانون عملیاتی گفته می‌شود.

اگر فرض کنیم که سیستم در تعادل حالت پایدار است، می‌توانیم بیان کنیم که تعداد ورودی‌ها و تعداد تکمیل‌ها در طول یک دوره مشاهده شده برابر خواهد بود (فرض تعادل جریان). این جملات ممکن است همیشه درست نباشند، اما می‌توانند سنجیده شوند و برای هر در مشاهده مورد نظر اعتبارسنجی شوند. بنابراین، این را نظریه عملیاتی گویند. به همین دلیل، می‌توانیم روابط دیگری حاصل کنیم، که در آن سیستم در تعادل جریان است:

$$A = C$$

$$A/T = C/T$$

$$\lambda = X \qquad (۱۳۲-۷)$$

$\lambda S = X S$

$\lambda S = U$

یک ویژگی سودمند تحلیل عملیاتی این است که تکنیک می‌تواند برای شبکه‌های باز و بسته استفاده شود. یک شرط که باید برآورده شود این است که شبکه تحت بررسی باید ازنظر عملیاتی همبند باشد. به همین ترتیب، هیچ سروری ممکن نیست که در طول کل دوره مشاهده idle باشد.

برای یک سیستم بسته، تعداد کارها در برنامه در شبکه را می‌دانیم و توان عملیاتی سیستم را از طریق یک نقطه خاص در شبکه پیدا می‌کنیم. مقادیر دیگر می‌توانند با استفاده از توان عملیاتی به‌عنوان یک نقطه شروع حاصل کنند. در یک سیستم باز، توان عملیاتی در پایان از شبکه فرض می‌شود که شناخته شده است، و ما از این برای یافتن تعداد مشتریان در صف استفاده می‌کنیم.

بیایید به برخی از مقادیر عملیات پایه نگاه کنید. فرض کنید که ما دارای شبکه دلخواهی هستیم که در طول دوره T تحت مشاهده قرار گرفته است. برای هر سیستم صف در شبکه، ما داده‌های زیر را به دست می‌آوریم:

A_i= تعداد ورودی در صف سیستم i

B_i= زمان مشغول بودن سرور i

C_{ij}= تعداد دفعاتی که یک گاز مستقیماً از سرور i به صف سرور j می‌رود است.

کارهایی که از سورس خارجی می‌رسند یا حفره خارجی را ترک می‌کند با A_{oi} و C_{io} نشان داده‌شده است. تعداد ورود و خروجی از سیستم با رابطه زیر داده‌شده است:

$$A_{\cdot} = \sum_{j=1}^{m} A_{oj} \qquad \text{تعداد ورودی ها} \qquad (۱۳۳-۷)$$

$$C_{\cdot} = \sum_{i=1}^{m} C_{io} \qquad \text{تعداد خروجی ها} \qquad (۱۳۴-۷)$$

مجموع تعداد تکمیل در سرور i به شرح زیر است:

$$C_i = \sum_{j=1}^{m} C_{ij} \qquad (۱۳۵-۷)$$

از مقدار اندازه گیری شده پایه که قبلاً تعریف‌شده است، چندین مقدار کارایی دیگر می‌توانند به شرح زیر استنتاج شوند:

بهره وری سرور i:

$$U_i = B_i/T$$

میانگین زمان سرویس‌دهی سرور i:

$$S_i = B_i \ /C_i$$

نرخ خروجی سرور i:

$$X_i = X_i \ / \ T$$

فرکانس مسیریابی از سرور i به j:

$$q_{ij} = \ C_{ij} \ / \ C\_i$$

می‌توانیم نرخ تکمیل کار این سیستمی را به شکل زیر نشان دهیم:

$$X_{.} = \sum_{i=1}^{m} X_i \ q_{io} \qquad (۱۳۶-۷)$$

و بهره وری هر سرور i به شکل زیر است:

$$U_i = X_i S_i \qquad (۱۳۷-۷)$$

اگر زمان انتظار در سرور خاص i را در هر بازه زمانی در طول دوره مشاهدات به‌عنوان مجموع زمان سرویس‌دهی مشتریان با ورودی جدید در نظر بگیریم، مجموع زمان انتظار برای همه کارها در سیستم در طول دوره جمع می‌شود:

$$W_i = \int_{.}^{T} N_i \ (t) dt \qquad (۱۳۸-۷)$$

میانگین طول صف در سرور در سؤال به شرح زیر است:

$$N_i = W_i \ / \ T \qquad (۱۳۹-۷)$$

زمان پاسخ سیستم سرور به شرح زیر است:

$$R_i = W_i / C_i \qquad (۱۴۰-۷)$$

با ترکیب معادلات (۷-۱۳۹) و (۷-۱۴۰)، معادله عملیاتی نتایج Little را کسب می‌کنیم:

$$N_i = (W_i / C_i)\,(C_i / T) = R_i\,X_i \qquad (۱۴۱-۷)$$

گر سیستم تحت مطالعه در حالت پایدار باشد، ما دارای توازن جریان هستیم، فرض می‌کنیم که نرخ ورود به سیستم صف برابر با نرخ تکمیل آن سیستم است. ما می‌توانیم نرخ توان عملیاتی سرور را برای هر سرور j به‌صورت زیر حاصل کنیم:

$$C_j = \sum_{i=۰}^{m} C_{ij}$$

$$C_j = \sum_{i=۰}^{m} \left(C_i\,C_{ij}\right) / C_i \qquad (۱۴۲-۷)$$

$$C_j = \sum_{i=۰}^{m} C_i\,q_{ij}$$

می‌توانیم عبارت مشابهی را که در معادله (۷-۱۳۶) بیان شده است را به دست آوریم، اما برای هر مورد، آن را تعمیم دهیم:

$$X_j = \sum_{i=۰}^{m} X_i\,q_{ij} \quad \text{for } j = ۰, ۱, \ldots, m \qquad (۱۴۳-۷)$$

روابط مشتق شده یک راه حل منحصر را حاصل می‌کنند اگر به یک سیستم باز اعمال شوند، چرا که توان عملیاتی ورودی، X، شناخته شده است. در یک سیستم بسته، معادله (۷-۱۴۳) نرخ توان عملیاتی نسبی را حاصل می‌کند، چرا که مقدار مطلق X را نمی‌داند.

Buzen [۹] نرخ ویزیت V_i را به‌عنوان تعداد دفعاتی که یک سرور خاص i ویزیت شده اند، نسبت به تعداد ورودی داده‌شده، تعریف می‌کند. می‌توانیم این مقدار را به‌عنوان نرخ توان عملیاتی در سرور i نسبت به مجموع توان عملیاتی ورودی بیان کنیم:

$$V_i = X_i / X_0 \qquad (۱۴۴-۷)$$

اگر فرض کنیم که جریان کارها در شبکه متوازن است، می‌توانیم $V_0 = ۱$ را تنظیم کنیم (چرا که همه کارها از ورودی شبکه عبور می‌کنند) و همه نرخ‌های ویزیت دیگر با استفاده از عبارت زیر حل می‌شوند:

$$V_j = q_0 \sum_{i=0}^{m} V_i q_{ij} \qquad (۱۴۵-۷)$$

این، می‌دانیم که استفاده از هر سرور در شبکه به ما اجازه می‌دهد که توان عملیاتی هر سرور دیگر را با استفاده از ترکیب معادلات (۷-۱۴۴) و (۷-۱۴۵) پیدا کنیم.

حال به مجموع زمانی که یک کار در سیستم مانده است به‌عنوان تابعی از هر توان عملیاتی سرور و میانگین طول صف نگاه کنید. مجموع زمان انتظار برای هر کار رسیده به سرور به این بستگی دارد که چقدر کار در سر صف جدید هستند و به نرخی که کارها توسط سرور تکمیل می‌شوند اشاره دارد. در هر سرور، می‌توان از معادله نتیجه Little (۷-۱۴۱) در ترکیب با معادله (۷-۱۴۴) برای کسب رابطه زیر استفاده کنیم:

$$N_i / X_0 = V_i R_i \qquad (۱۴۶-۷)$$

اگر ما معادله (۷-۱۴۶) را با همه سرورها جمع کنیم، یک عبارت کلی را به دست می آوریم که می‌تواند به‌عنوان نتیجه Little استفاده شده برای کل سیستم تفسیر شود»

$$\sum_{i=1}^{m} N_i / X_o = \sum_{i=1}^{m} V_i R_i \qquad (۱۴۷-۷)$$

در اینجا تعداد کارها در هر سیستم در زمان داده‌شده به‌سادگی مجموع کارها در سرورهای شبکه حاصل می‌شود.

$$N = \sum_{i=0}^{m} N_i \qquad (۱۴۸-۷)$$

بنابراین داریم:

$$N / X_0 = \sum_{i=1}^{m} V_i R_i \qquad (۱۴۹-۷)$$

سمت چپ معادله (۷-۱۴۹) می‌تواند یک کاربرد نتیجه Little برای سیستم در کل دیده شود، بنابراین، زمان پاسخ سیستم را تعریف می‌کنیم:

$$R = N / X_0 = \sum_{i=1}^{m} V_i R_i \qquad (۱۵۰-۷)$$

موضوع نهایی که تحت تحلیل عملیاتی پوشش می‌دهیم تحلیل گلوگاه در سیستم بسته است. در هر شبکه، یکی از سیستم‌های صف در نهایت قادر به حفظ تقاضای افزایش‌یافته سرویس با افزایش تعداد کارها در شبکه نیستند. این سرور خاص به طور متعاقبی حداکثر میزان توان عملیاتی شبکه را در کل تعیین می‌کند. اگر بهره وری آن به یک برسد، یک دستگاه اشباع شده دیده می‌شود (قادر به پردازش سریعتر کارها نیست)، در این مورد، توان عملیاتی به‌صورت معکوسی با زمان سرویس متناسب است، چرا که همیشه یک کار در سرویس وجود دارد.

$$X_{\cdot} = ۱/ S_{\cdot} \qquad (۱۵۱-۷)$$

اگر معادلات را ترکیب کنیم (۷-۱۳۷) و (۷-۱۴۴)، می‌توانیم بهره وری نسبی هر دو سرور را در شبکه به‌صورت زیر بیان کنیم:

$$U_i / U_j = V_i S_i \ / V_j S_j \qquad (۱۵۲-۷)$$

توجه داشته باشید که میزان بهره وری سرور به توان عملیاتی سرور بستگی ندارد، میزان مستقل از بار سیستم ثابت باقی می‌ماند. بنابراین، دستگاه با بزرگ‌ترین مقدار $V_i S_i$ با افزایش بار به گلوگاه شبکه تبدیل می‌شود.

سپس، یافتن حداکثر مقدار توان عملیاتی سیستم در زمانی که گلوگاه در حالت اشباع است، ممکن است. برای گلوگاه، در سرور b، توان عملیاتی برابر با معکوس زمان سرویس است، که می‌توانیم معادله (۷-۱۴۴) و (۷-۱۵۱) را برای کسب حداکثر توان عملیاتی سیستم ترکیب کنیم:

$$V_b = X_b / X_{\cdot} = ۱ / X_{\cdot} s_b$$

$$X_{\cdot} = ۱ / V_b S_b \qquad (۱۵۳-۷)$$

زمان واکنش شبکه، در مورد شبکه، با معادله (۷-۱۵۰) داده‌شده است:

$$R = N / X_{\cdot} = N V_b S_b \qquad (۱۵۴-۷)$$

و این توسط سرور گلوگاه محدود شده است.

Buzen و Denning [۱۰] نتایج عملیاتی بحث شده را به سیستم‌هایی با رفتار مستقل از بار، بسط دادند. این، پروپوزال اولیه برای تحلیل عملیاتی شبکه صف بندی شده را می‌توان در [۱۲] یافت.

۵-۷ خلاصه

عرصه کشف شده در این فصل، از فرآیند احتمالی تا نظریه صف بندی تا تخمین پایه، یک دامنه وسیع از موضوعات هستند، هریک با تعداد زیادی تخصص و تکنیک همراه هستند. اصلاحات ارائه شده، باورها، به تشریح سودمندی تحلیل آماری و نظریه صف بندی، برای ارائه مبنایی برای درک برخی از تکنیک‌ها و روش‌های استفاده شده در شبیه سازی اشاره دارد. بحث‌های دقیق‌تر بر مسائل مربوطه با اماره و احتمال‌های پایه در بسیاری از متون احتمالی پایه، از جمله [۴،۵]، برای نظریه صف بندی، در مرجع [۳] آورده شده است. موضوعات نظریه صف در مرجع [۳] و این مرجع [۲،۶،۷،۱۳] بحث شده‌اند. تخمین، که به سیستم صف بندی مربوط است، در طی مراحلی اصلاح شده است [۲،۳]. کاربرد تکنیک‌های بحث شده در این فصل ما را قادر به محاسبه، تحت فرضیات و شرایط اصلی می‌کند، بسیاری از مقادیر کارایی مورد توجه می‌توانند در تحلیل نظریه صف سنتی یافت شوند. نتایج به‌دست‌آمده، اغلب شهودی‌تر هستند و می‌توانند به‌صورت ساده‌تری به سیستم واقعی برای موارد مورد نظر ربط داده شوند.

فصل ۸

تحلیل شبیه‌سازی

شبیه‌سازی یک تحقق از یک مدل برای یک سیستم در فرم قابل‌اجرای کامپیوتری است. به همین ترتیب، مدل سیستم جهان واقعی به زبان شبیه سازی کامپیوتری تفسیر می‌شود. تحقق کامپیوتر یک دستگاه برای انجام آزمایش با مدلی به‌منظور کسب بینشی از رفتار سیستم یا ارزیابی گزینه‌ها انجام‌شده است. شبیه سازی ها، برای مؤثر بودن، نیازمند فرمول سازی دقیق سیستم مطالعه شده، تفسیر صحیح این فرمولاسیون در یک برنامه کامپیوتری،، و تفسیر نتایج هستند.

شبیه سازی معمولاً به دلیل پیچیدگی بیشتر سیستم کامپیوتری که استفاده از ابزارهای ریاضی ساده تر را برای مطالعه واقع بینانه عملکرد به چالش می‌کشد انجام می‌شود. این پیچیدگی ممکن است از فرآیند احتمالی ذاتی در سیستم، تراکنش‌ها پیچیده عناصری که فاقد فرمولاسیون ریاضی هستند، یا مشکلات ذاتی روابط ریاضی که از محدودیت‌ها و معادلات سیستم ناشی می‌شود، رخ دهد. به دلیل این محدودیت‌ها و دلایل دیگر، شبیه سازی اغلب ابزاری برای ارزیابی است. شبیه سازی مزایای بالقوه بسیاری را برای مدل ساز ارائه می‌دهد. این آزمایش‌ها و مطالعه تراکنش داخلی پیچیده ترکیبی یک سیستم خاص، با پیچیدگی به عهده مدل‌ساز گذاشته‌شده را ترکیب می‌کند.

شبیه‌سازی تحلیل حساسیت سیستم را با ارائه ابزاری برای تغییر مدل و مشاهده اثر آن بر رفتار سیستم اجازه می‌دهد. با استفاده از شبیه‌سازی می‌توانیم درک بهتری از سیستم واقعی حاصل کنیم. این به این دلیل است که جزییات مدل و نیازهای مدل‌سازی به‌صورت مستقلی سیستم کامپیوتری را به‌منظور ساخت صادقانه یک شبیه‌سازی درک می‌کند. فرآیند یادگیری در مورد سیستم به‌منظور شبیه‌سازی آن اغلب به پیشنهادهایی برای تغییر و بهبود منجر می‌شود. شبیه‌سازی ابزاری را برای تست این فرضیه‌ها ارائه می‌دهد. شبیه‌سازی اغلب به درک بهتر اهمیت عناصر گوناگون سیستم و اینکه چگونه آن‌ها با یکدیگر در تراکنش هستند منجر می‌شود. این روش یک محیط آزمایشگاهی را ارائه می‌دهد که می‌توانیم در آن گزینه‌ها بسیاری و اثرات بسیاری را قبل از اینکه سیستم به واقع به وجود آید، بدون اخلال و آشفتگی بررسی کنیم. شبیه سازی مدل ساز را قادر می‌کند که سیستم‌های پویا را در واقعیت مطالعه کند، و ابزارهایی را برای بررسی جزئیات شرایط و فرآیندهایی که در غیر این صورت نمی‌توانند اجرا شوند ارائه دهند. سرانجام، این ابزاری را برای مطالعه اثرات بر سیستم موجود افزودن مؤلفه جدید، خدمات و غیره بدون تست آن‌ها در سیستم ارائه می‌دهد. این ابزاری برای کشف گلوگاه‌ها و دیگر مسائل قبل از اینکه به واقع پیاده سازی شود و تغییرات اجرا شود، ارائه می‌دهد.

شبیه سازی برای انواع اهداف گوناگون استفاده می‌شوند، همان‌طور که در انواع موضوعات پوشش داده‌شده در فرضیات شبیه سازی سالانه می‌بینیم. زمینه‌های بسیاری، از جمله تجاری، اقتصادی، بازاریابی، آموزشی، سیاسی، علوم اجتماعی، علوم رفتاری، علوم طبیعی، روابط بین المللی، حمل و نقل، جنگ، تصویب قانون، مطالعات شهری، سیستم‌های جهانی، سیستم‌های فضایی، طراحی کامپیوتر و عملیات مدیون شبیه سازی هستند.

تا اینجا از "سیستم" برای تشریح نهاد مدل شده مورد نظر استفاده کردیم. در زمینه شبیه سازی، طراحی یک مجموعه از اشیا با مجموعه به خوبی تعریف‌شده از تراکنش‌ها بین آن‌ها استفاده می‌شود. یک صندوقدار با خطی از مشتریان در تراکنش هستند، کار کارمند در این زمینه ممکن است یک سیستم دیده شود، مشتریان و کارمند شی هستند و توابع توسط هریک (واریز و برداشت) به‌عنوان مجموعه تراکنش اجرا می‌شود.

سیستم در ماهیت خود معمولاً پیوسته یا گسسته است، در حالی که این عبارت حاکی از رفتار متغیرهای مربوط به سیستم است. آن‌ها برای مدل ساز زمینه‌ای را فراهم می‌کنند که از مدل خود استفاده کند و آن را شکل دهد. در هر دو مورد، روابط معمولی متغیرها حول زمان صورت می‌گیرد. در مورد مدل گسسته، زمان به صورت مرحله ای در بازه‌های ثابت تعیین شده توسط رخدادها در برابر برخی از تدوین‌ها رو به جلو می‌رود، و در مدل پیوسته، متغیرها به‌صورت پیوسته با زمان تغییر می‌کنند. برای مثال، در سناریوی بانک، اگر متغیر مورد نظر تعداد مشتریان مورد انتظار برای سرویس باشند، ما یک دنباله "شمارش" گسسته داریم. از طرفی دیگر، اگر ما به دنبال یک صندوقدار drive،up باشیم می‌توانیم یک متغیر پیوسته وابسته به زمان را در خط را تا زمانی که وجود دارد مدل کنیم.

سیستم‌ها می‌توانند هر دو متغیرهای پیوسته و گسسته را داشته باشند و باز هم مدل شوند. در واقعیت، این موردی است که مکرر تکرار شده است. دیگر ملاحظه در تعریف یک سیستم ماهیت پردازش آن است. پردازش، خواه پیوسته باشد یا گسسته، می‌تواند ویژگی دیگری داشته باشد، و آن ویژگی قطعی بودن یا تصادفی بودن است. یک سیستم قطعی سیستمی است که، با توجه به ورودی x و شرط اولیه i، شما همیشه یک ورودی می‌گیرید: $y = f(x, i)$. به همین ترتیب، اگر بخواهیم همان پردازش را به دفعات نامحدودی اجرا کنیم، با همان ورودی و همان حالت اولیه فرآیند، ما نتایج یکسانی را تحقق می‌بخشیم.

از طرفی دیگر، اگر سیستم احتمالی باشد، آنچه گفته شد برقرار نیست. برای سیستم مشابه با ورودی X و حالت اولیه I، می‌توانیم خروجی Y را در یکی از بسیار خروجی‌های ممکن اتخاذ کنیم. این بر اساس ماهیت تصادفی فرآیند احتمالی است. به همین ترتیب، آن‌ها به‌صورت تصادفی بر نتایج ممکن توزیع می‌شوند. برای مثال، اگر سیستم صندوقدار به‌عنوان یک سیستم گسسته تشریح شود، فرض می‌کنیم که زمان سرویس‌دهی سرور دقیقاً مشابه است و نرخ ورود مشتری ثابت و بدون تغییر است. بااین‌حال، اگر همان سیستم در واقعیت داده شود، بر اساس کاری که صندوقدار باید انجام دهد و چگونگی اجرای آن، می‌دانیم که سرویس تصادفی است. به همین ترتیب مشتریان به یک ترتیب مناسب نمی‌رسند، آن‌ها به‌صورت تصادفی به سیستم می‌رسند. در هر دو مدل ما نتایج بسیار متفاوتی حاصل می‌کنیم.

۸-۱ فرآیند شبیه سازی

برای استفاده از کامپیوتر دیجیتال برای اجرای مدل‌سازی و اجرای آزمایش‌ها تکنیک مشهوری وجود داشت. در این محیط شبیه سازی می‌توان مطالعات سیستماتیکی از مسائل را ایجاد کرد که نمی‌تواند توسط تکنیک‌های دیگر مطالعه شود. مدل شبیه سازی، سیستم را ازنظر عناصر مورد نظر و روابط آن‌ها تشریح می‌کند. در زمان تکمیل، این یک آزمایشگاه را ارائه می‌دهد که آزمایش‌های بسیاری بر روی این عناصر و تراکنش‌های آن‌ها صورت گرفته است.

برنامه‌های شبیه سازی، با مدل‌سازی عمومی، نیازمند این است که فاز گسسته به‌منظور تحقق پتانسیل کامل آن اجرا شود. آن‌ها به شرح زیر هستند:

۱. مسئله نیازمند شبیه سازی را تعیین کنید.
۲. یک مدل برای حل مسئله فرمول کنید.
۳. مدل شبیه سازی مسئله را فرمول کنید.
۴. مدل را در زبان مناسب اجرا کنید.
۵. آزمایش‌های شبیه سازی را طراحی کنید.
۶. مدل را اعتبارسنجی کنید.
۷. آزمایش‌ها را اجرا کنید.

پروژه مدل شبیه سازی معمولی بیشتر زمان خود را، به دلیل پیچیدگی‌های مربوط به فرمول کردن مدل و تبدیل به فرمت‌های شبیه سازی و اجرا در یک زبان، در فازهای ۲،۳ و ۴ صرف می‌کند. فرمولاسیون مدل به تعاریف عناصر مهم سیستم جهان واقعی و تراکنش‌های آن‌ها مربوط است. زمانی که این عناصر مهم شناخته و تعریف می‌شوند (به‌صورت ریاضی، رفتاری، تابعی) و رفتار آن‌ها (علت و معلول، قبلی و بعدی، وابستگی‌ها و غیر وابستگی ها، جریان‌های داده، و جریان کنترل) ازنظر ضرورت آن‌ها تعریف می‌شوند، مدل شبیه سازی به درون و همراه با تعریف مدل سیستمی جریان می‌یابد. به همین ترتیب، همان‌طور که ما یک مدل سیستمی را توسعه دادیم، می‌توانیم مستقیماً ساختار مدل شبیه سازی را تعریف کنیم.

یک جنبه مهم رشد مدل انتخاب سطح مناسب شبیه سازی است، که مستقیماً متناسب با هدف مورد نظر ارزیابی عملکرد، درجه درک سیستم، محیط آن، متغیرهای خروجی موردنیاز است. از یک سو، برای مثال، می‌توانیم سیستم صندوقدار خود را تا سطح مدل‌سازی همه اقدامات او پایین آوریم. یا، از طرف دیگر، سرویس کارمند را با تخمین زمان ناخالص برای اجرای سرویس صرف نظر از نوع سرویس مدل کنیم. سطح انتخابی به آنچه بررسی می‌شود بستگی دارد. در اولین مثال، ممکن است بخواهیم که بیشتر جنبه‌های زمان بر تابع آن‌ها را ایزوله کنیم لذا آن‌ها می‌توانند روشی برای بهبود آن‌ها توسعه دهند. در سطح دوم احتمالاً همه آنچه که می‌خواهیم تعیین کنیم بر اساس بار مشتری، میانگین زمان سرویس کارمند، و تعداد بانکداران بهینه‌ای که وظیفه این کار دارند، است.

هدف از اندازه‌گیری عملکرد ما را مستقیماً به سطح شبیه‌سازی جزییات تحریک می‌کند، که معمولاً در جایی بین این دو قطب قرار می‌گیرد: خیلی پایین و خیلی بالا سودمند است. در بیشتر موارد، ما به‌عنوان مدل کننده نمی‌توانیم و نمی‌خواهیم که همیشه اینکه چگونه سطح جزییات همه بخش‌ها می‌توانند بر سودمندی نهایی مدل تأثیر گذارند را پیش‌بینی کنیم. یک راه حل که معمولاً برای مقابله با این عدم قطعیت‌هایی استفاده می‌شود ساخت مدل به روش ماژولار است، که به هر مؤلفه اجازه می‌دهد که سطح سازگاری و تأثیر کلی بر شبیه سازی و سیستم را کاهش دهیم. آنچه که معمولاً ما را تحریک می‌کند رشد مدل بالا به پایین، با هر لایه که در صورت لزوم اصلاح شده است می‌باشد.

شبیه سازی‌ها، فراتر از ساختار آن‌ها (عناصر و تراکنش‌ها)، نیازمند ورودی داده و استخراج داده برای سودمند کردن آن‌ها است. معمول ترین شبیه‌سازی‌ها خودمحور یا ردیابی محور هستند. در

شبیه‌سازی‌های خودمحور خود مدل (برای مثال برنامه) محرک‌های تعبیه شده ای برای ارائه داده موردنیاز برای تحریک شبیه سازی دارد. این داده معمولاً با انواع توزیع‌های تحلیلی تحریک می‌شود و به مولد عدد تصادفی پیوند خورده است. در مثال سیستم صندوقدار، ما از شبیه سازی خود محور استفاده می کنیم. ما ممکن است از یک توزیع ورود پواسون برای تشریح ماهیت تصادفی مشتریانی که به سیستم می‌رسند استفاده کنیم. این استفاده‌ای نشانی از برخی از جریان‌های تولید شده برای ورودی‌های سیستم مدل است. در مورد دیگر، زمانی که از داده ردیابی محور یا داده بر اساس مستندات استفاده می کنیم، شبیه سازی با محرک خارجی تحریک می‌شود. معمولاً این‌ها استخراج شده‌اند، کاهش‌یافته‌اند، و داده به یک سیستم در حال اجرای واقعی همبسته است. برای مثال، در مورد صندوقدار، می‌خواهیم که یک پایگاه واقع‌بینانه‌تر داشته باشیم تا تعداد بهینه کارمندان و ساعات آن‌ها را محاسبه کنیم. در این موردی می توانیم دوره زمانی که مشتریان به‌صورت پویا برای سرویس گرفتن به بانک می‌رسند را بسنجیم. این اطلاعات جمع‌آوری‌شده سپس برای یک دنباله ورودی ذخیره شده استفاده می‌شود، که شبیه‌سازی را بر اساس این داده‌های واقع بینانه تحریک می کند. این نوع مدل‌سازی به سیستم جهان واقعی نزدیک‌تر است اما عیب نیاز به جمع‌آوری داده‌های مقدماتی و تحلیلی برای در دسترس ساختن این داده‌ای برای استفاده دارند.

۸-۲ کنترل زمانی

در شبیه سازی پیوسته و گسسته، نگرانی‌های اصلی در اجرای شبیه سازی مدیریت زمان و استفاده آن در تحت تأثیر قرار دادن متغیرهای وابسته است. زمان بندی در برنامه شبیه سازی برای همگام‌سازی رخدادها، محاسبه تغییرات حالت، و کنترل تراکنش‌های سراسری استفاده می‌شود. زمان بندی می‌تواند دو حالت به خود بگیرد: همگام و ناهمگام.

زمان بندی سنکرون به الگوی زمان بندی اشاره دارد که در آن پیشرفت زمانی ثابت است، به‌صورت مناسبی واحدهای زمانی، t، انتخاب می‌شوند. در هر به‌روزرسانی، حالت سیستم برای انعکاس این تغییر زمانی به‌روز می‌شود. به همین ترتیب، همه رخدادها در طول این دوره زمانی تعیین‌شده‌اند و حالت آن‌ها منطبق با آن تنظیم می‌شود. این فرآیند پیشبرد زمان (در مراحل) و به روز رسانی حالت عناصر تا زمانی که شبیه سازی به شرایط مرزی برسد رخ می‌دهد (زمان تا حداکثر، که رخدادهایی وقوع یابند پیش

میرود). در سیستم صندوقدار، زمان بندی برای تعیین ورود و سرویس‌دهی نیاز است. برای سازمان دهی t مرحله‌ای در هر توقف، ما باید بررسی کنیم که آیا یک ورود باید رخ دهد، آیا یک سرویس باید تکمیل شود، یا آیا یک مورد جدید باید آغاز شود یا نه. در ذهن داشته باشید که استفاده از زمان سنکرون جزئی از انتخاب گام است. اگر یک مرحله بیش از حد بزرگ انتخاب شود، رخدادها به‌صورت همزمان رخ می‌دهند که در واقعیت این چیزی امکان پذیر نیست. از طرفی دیگر، از گام بیش از حد کوچک باشد باعث می‌شود که مراحل بسیاری حاصل شود که در حقیقت این چیزی رخ نمی‌دهد. مورد دوم باعث زمان اجرای زیاد کامپیوتر می‌شود اما تمایز بسیار کمی بین رخدادها وجود دارد. مورد اول، از طرف دیگر، باعث اعوجاج همه چیز و احتمال از دست دادن سودمندی است. کار مهم مدل‌سازی انتخاب زمان مرحله مناسب برای اینکه مدل سودمند باشد اما از زمان کامپیوتر بیشتر نشود است.

آسنکرون، یا زمان بندی رخداد، از زمان بندی سنکرون در مواردی که زمان در یک مقدار متغیر به جای مقدار ثابت پیشرفت می‌کند تفاوت دارد. به معنی حفظ دنباله‌ای از رخدادها در برابر مراحل زمانی است. زمان بر اساس رویداد زمانی بعدی که رخ می‌دهد، پیشرفت می‌کند. این رخدادهای به ترتیب زمان معمولاً در یک لیست پویا نگه داشته می‌شوند، که بر هر رخداد برای انعکاس رخدادهایی جدیدی که در آینده رخ می‌دهند به روز رسانی و تنظیم می‌شوند، است.

در مثال صندوقدار صف رخداد، یا لیست، از دو رخداد تشکیل شده است: ورودی بعدی و تکمیل سرویس بعدی. به‌صورت انتزاعی، این روش ساده‌تر بصری سازی می‌شود. رخدادها باید به ترتیب وقوع باشند و با رخداد تازه رسیده تنظیم شوند. مسئله این است که، در مورد اول، چگونه درج یا زمان بندی رخداد جدید یا شرایط جدید در شبیه سازی رخ می‌دهد. در بخش بعدی این و جنبه‌های دیگر استفاده زمانی در شبیه سازی‌ها را بررسی خواهیم کرد.

۸-۳ شبیه سازی‌ها و مدل‌سازی

تا این نقطه، ما بر ویژگی‌های عمومی بحث شده مربوط به مدل‌سازی شبیه سازی بحث کردیم. ما بر کلاس تکنیک‌های مدل‌سازی موجود یا دسته بندی تکنیک‌های اجرای شبیه سازی (زبان شبیه سازی) بحث نکردیم. تکنیک‌های شبیه سازی شامل رخداد گسسته، تغییر پیوسته، صف، ترکیب، و تکنیک

ترکیبی است. هر مورد نقطه نظر خاصی را برای اعمال بر مسئله شبیه سازی ارائه می‌دهد. آن‌ها مدل‌سازی را برای برازش مدل‌ها با خصیصه‌های ذاتی تکنیک‌ها تشویق می‌کنند.

۸-۳- ۱ مدل‌های گسسته

در مدل‌های شبیه سازی گسسته، اشیای سیستم واقعی معمولاً موجودیت گفته می‌شوند. موجودیت‌ها با خود صفاتی دارند که آن‌ها را تشریح می‌کند. اقدامات بر این موجودیت‌ها بر شرایط و نقاط مرزی رخ می‌دهد. به این شرایط رخداد گفته می‌شود. رخدادهایی مانند ورود، سرویس‌دهی، نقاط توقف، دیگر سیگنالینگ رخداد، زمان انتظار و غیره معمول هستند.

موجودیت‌ها صفاتی دارند که اطلاعاتی را در مورد آنچه که باید بر اساس دیگر رخدادها و یا شرایط رخ دهد، را ارائه می‌دهند. تنها در این وقوع‌های شرطی و مرزهای رخداد می‌تواند حالت موجودیت تغییر کند. برای مثال، در شبیه سازی صندوقدار، تنها یک ورود مشتری (رخداد ورود) می‌تواند رخداد سرویسی باشند که زمان بندی می‌شود، یا تنها یک رخداد سرویسی باشد که می‌تواند به رخداد سرویس زمان بندی شده پایان دهد. این حاکی از این است که بدون رخدادها شبیه سازی کاری انجام نمی‌دهد. این تکنیک مدل‌سازی تنها بر مفهوم رخدادهای زمان بندی و عمل بر روی آن‌ها کار می‌کند. بنابراین، ضروری است که قابلیت‌هایی وجود داشته باشد که رخدادها را در صف برنامه نویسی یا لیست قرار دهد و آن‌ها را بر اساس برخی از شرایط مورد نظر حذف کند.

این تکنیک حاکی از این است که همه اقدامات درون شبیه سازی با مرزهای رخداد تحریک می‌شوند. به همین ترتیب، شروع و پایان رخداد می‌تواند رخدادهای دیگری باشند که شبیه سازی می‌شوند. همه چیز بین این مرزهای رخداد، یا نقاط جمع آوری داده، در حال تغییر هستند. یک مدل شبیه سازی که از این تکنیک استفاده می‌کنند نیاز دارد که مدل ساز همه رخدادهای ممکن را درون سیستم واقعی و اینکه چگونه این رخدادها بر حالت رخدادهای دیگر در سیستم تأثیر می‌گذارد را تعریف کند. این فرآیند شامل تعریف رخدادها و تعریف تغییرات برای دیگر حالات در همه مرزهای رخداد است، از همه فعالیت‌هایی که می‌توانند اجرا شوند، و از تراکنش بین موجودیت‌ها درون سیستم شبیه سازی شده. در این نوع مدل‌سازی هر رخداد باید برخی از رخدادهای دیگر درون سیستم را هدف قرار دهد. اگر این شرط برقرار نباشد، نمی‌توانیم یک شبیه سازی در حال کار واقع بینانه را ایجاد کنیم. این تریگر تراکنش

رخدادها و رابطه رخدادها با یکدیگر را فراهم می‌کند. برای مثال، برای مدل کردم یک خودپرداز خود سرویس، حداقل به تعریف موجودیت‌ها و نهادهای زیر نیاز داریم:

- رخدادهای ورودی
- رخدادهای سرویس
- رخدادهای خروج
- رخدادهای جمع آوری
- موجودیت‌های مشتری
- موجودیت‌های سرور

رخدادها چگونگی وقوع فرآیند را هدایت می‌کنند و موجودیت‌ها واسطی را برای عمل ارائه می‌دهند، همه آن‌ها با رخداد جمع آوری نظارت می‌شوند که دیدگاه "اسنپ شاتی" از سیستم ارائه می‌دهند. این ابزاری را برای استخراج متغیرهایی از موجودیت‌ها ارائه می‌دهد. در این مثال، تشریح زیر می‌تواند برای ایجاد یک مدل ساده استفاده شود:

۱. رخداد ورودی

- زمان بندی ورود بعدی (زمان حاضر + T)
- اگر همه کارمندان مشغول باشد، تعداد انتظار = تعداد انتظار + ۱
- اگر هر کارمندی آزاد باشد و هیچ کس قبل از مشتری منتظر نباشد، رخداد سرویس را زمان بندی کن.

۲. رخداد سرویس

- تعداد کارمندان مشغول = تعداد کارمندان مشغول + ۱
- سرویس و رخداد را بر اساس نوع سرویس زمان بندی کن.
- شروع آماره سرویس

۳. پایان رخداد سرویس

- تعداد کارمندان مشغول = تعداد کارمندان مشغول ،۱

- زمان بندی ورودی مشتری
- پایان اماره سرویس

۴. موجودیت ها

- کارمندان
  - تعداد کارمندان
  - نوع و نرخ سرویس
  - نوع سرویس
- مشتریان
  - نرخ ورود
  - پویایی (نوع سرویس موردنیاز)

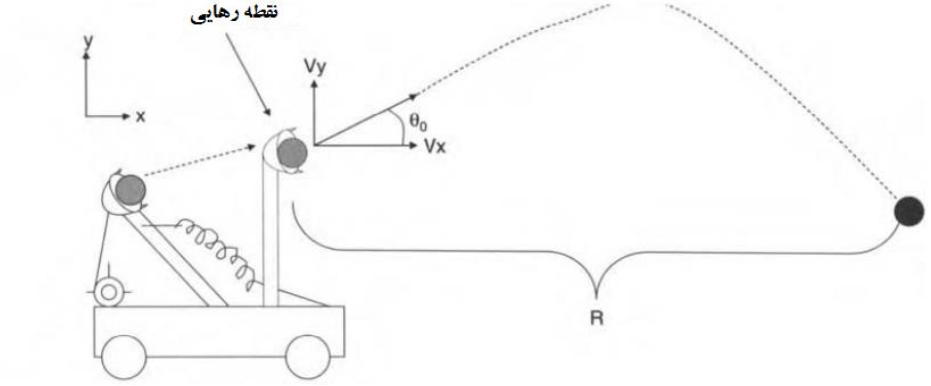
یک شبیه سازی رخداد گسسته (با زبان مناسب) می‌تواند با استفاده از این رخدادها و موجودیت‌ها برای مبنا صورت گیرد. یک مدل بر اساس استفاده از این شرایط برای زمان بندی برخی از تعداد ورودی‌ها و برخی از شرط‌ها استفاده شوند. روابطی که بین موجودیت‌ها وجود دارند باعث اجرای مدل، با متغیرهای اتخاذ شده تا زمانی که شرط نهایی برقرار شود می‌شوند. این مثال بسیار ساده است و به هیچ وجه کامل نیست، اما تشریحی از برخی از مفاهیم پایه مربوط به شبیه سازی رخداد گسسته را ارائه می‌دهد.

۸-۳-۲ مدل‌سازی پیوسته

شبیه سازی پیوسته با مدل‌سازی رخدادهای فیزیکی (فرآیندها، رفتارها، شرایط) می‌پردازد که می‌تواند توسط برخی از مجموعه‌های متغیرهای وابسته درحال تغییر پیوسته تشریح شود. این به نوبه خود در معادلات دیفرانسیلی یا تفاضلی گنجانده شده است که فرآیند فیزیکی را تشریح می‌کند. برای مثال، می‌خواهیم نرخ تغییر سرعت پرتاب شی گلوله از منجنیق و فاصله R آن از catapault را تعیین کنیم. با صرف نظر از مقاومت باد، معادله برای آن به شکل زیر است. سرعت، c، در هر زمانی به‌صورت زیر است:

$$v_x = v. \cos\theta$$

$$v_y = V_{\cdot} \sin\theta_{\cdot} - gt \qquad (۱-۸)$$



شکل (۸-۱): حرکت پرتابه

و فاصله در جهت x به‌صورت زیر است:

$$R = V_x t = t v_o \cos\theta \qquad (۲-۸)$$

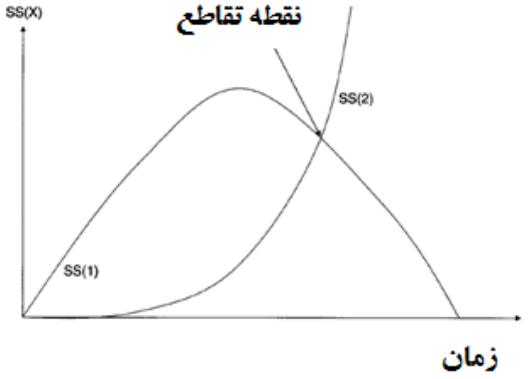
این معادلات می‌توانند در معادلاتی فرمول شوند که می‌تواند در یک زبان پیوسته برای تعیین حالات آن‌ها در هر دوره زمانی t فرمول شود.

با استفاده از این معادلات حالت، ما می‌توانیم شبیه سازی تغییر مبتنی بر حالت را ایجاد کنیم که به ما ابزاری برای هدف قرار دادن رخدادهای خاص ارائه می‌دهد. برای مثال، در این معادلات ممکن است بخواهیم یک رخداد را هدف قرار دهیم (شلیک در زمانی که vy برابر ۰ است). به همین ترتیب، زمانی که پرتابه بیشتر از آن بالا نیست، و به حداکثر ارتفاعش رسیده است، پرتاب صورت می‌گیرد. در این رخداد معادله ممکن است به شکل زیر باشد:

$V_y = ۰$; اجرای بازخورد را شروع کنید (۳-۸)

مثال دیگر این نوع تریگر در شکل (۸-۲) نشان داده‌شده است. در این مثال، دو فرمول پیوسته در طول زمان محاسبه می‌شوند، زمانی که نتایج آن‌ها معادل هستند (رخداد متقاطع)، برخی از رخدادهای دیگر برای وقوع زمان بندی می‌شوند. این نوع عملیات به ما اجازه می‌دهند که محاسبات جدید را هدف قرار دهیم یا مقادیر فعلی را بر اساس روابط معادلات پیوسته با یکدیگر تنظیم کنیم.

با استفاده از ترکیب خود تریگر و تریگرهای مقایسه‌ای (کمتر، بیشتر، مساوی) می‌توانیم شبیه سازی‌های دخیل تری از سیستم‌های پیچیده را ایجاد کنیم. کار اصلی شبیه ساز در این نوع مدل شبیه سازی توسعه یک مجموعه از معادلاتی است که پویایی سیستم تحت مطالعه را تعریف می‌کند و چگونگی تراکنش آن‌ها را تعیین می‌کند.



شکل (۸-۲): نمودار متغیر پیوسته

۸-۳-۳ مدل‌سازی صف

کلاس دیگر مدل عمومی، مدل صف بندی است. زبان شبیه سازی مبتنی بر صف وجود دارد (gert, Slam II،AWESIM, GPSS, Q) و برای حل انواع مسئله‌ها استفاده می‌شود. همان‌طور که در ابتدا نشان داده شد، بسیاری از مسائل مدل شده می‌توانند به آسانی به‌عنوان ارتباطات داخلی صف‌ها، با اصول صف بندی و نرخ‌های سرویس گوناگون مدل شوند. به همین

ترتیب، یک زبان شبیه‌سازی که مدل‌های صف بندی و تحلیل آن‌ها را حمایت می‌کند تا حد زیادی مسئله مدل‌سازی را ساده می‌سازند. در این زبانی امکان پشتیبانی از تعریف صف ازنظر اندازه صف، تعداد سرورها، نوع صف، اصول صف، نوع سرور، اصول سرور، ایجاد مشتری، نظارت بر عملیات، نقطه جمع‌آوری خروج، جمع‌آوری آماره، و همبستگی، ارائه عملیات وجود دارد. علاوه بر سرویس‌های پایه موارد دیگری برای کاستن از سرعت مشتریان یا مسیریابی آن‌ها به محل‌های گوناگون در شبکه صف وجود دارد. جزییات این ابزار مدل‌سازی بعدها در این فصل گفته می‌شود.

۸-۳-٤ مدل‌سازی ترکیب‌شده

هریک از تکنیک‌های تشریح شده قبلی به مدل‌ساز دیدگاه واضحی از تناسب مدل سیستم ارائه می‌دهد. مدل رخداد محور گسسته به ما دیدگاهی در مورد اینکه کدام سیستم‌ها از نهادها تشکیل شده است و کدام رخدادها برای تغییر حالت این نهادهایی وقوع می‌یابند، ارائه می‌دهد. مدل‌های پیوسته ابزاری برای اجرای شبیه‌سازی بر اساس معادلات دیفرانسیل یا فرمول‌های تفاضلی ارائه می‌دهد که پویایی متغیر زمانی یک عملیات سیستم را تشریح می‌کند. مدل‌سازی صف به مدل‌ساز، یک بررسی از سیستم‌هایی که از صف و سرویس تشکیل شده است ارائه می‌دهد. ساختار از چگونگی ارتباط آن‌ها و اینکه چگونه ارتباط آن‌ها با خروجی سرورهای صف تحریک می‌شود می‌آید.

مشکل همه سه تکنیک این است که به‌منظور استفاده از آن‌ها، مدل‌سازی باید مسئله را ازنظر ساختار موجود تکنیک فرمول کند. این نمی‌تواند به یک روش طبیعی فرمول شود و سپس به آسانی تفسیر شود. بار حاصل از ارتباطات یک چارچوب بر عهده مدل‌ساز و زبان شبیه‌سازی است. راه حل ارائه زبان ترکیبی است که ویژگی‌های هر سه تکنیک را دارد. در این زبانی مدل‌سازی می‌تواند شبیه سازی‌هایی را به یک روش بالا به پایین حاصل کند، و جزییات به سطوح پایین‌تر واگذار کند. برای نمونه، در سیستم صندوقدار، می‌توانند در ابتدا آن را به‌عنوان صف واحد با n سرور (کارمند) مدل کنیم. اصل صف اول ورود اول سرویس است، و اصل سرویس می‌تواند هر توزیع ساده‌ای ، مانند توزیع نمایی باشد. این مدل ساده به ما یک بررسی عاقلانه بر صحت مدل خود با مرزهایی برای تعیین سریع حد سیستم می‌دهد. ما

می‌توانیم سپس بر مدل کردن سرویس کارمند در جزییات بیشتر با کاهش آن به سطح مؤلفه‌هایی در سطح مدل‌سازی رخداد، تصمیم بگیریم.

در این نقطه می‌توانیم فعالیت کارمند را به‌عنوان مجموعه از رخدادهایی که نیاز است ترتیبی از خدمات را رو به سمت تکمیل شدن بپیمایند، مدل کنیم. در صورت امکان، می‌توانیم جنبه مدل پیوسته را در اصلاحات بیشتر برخی از ویژگی‌های دیگر ترکیب کنیم. جنبه اصلی جمع‌آوری این‌ها از فرم مدل‌سازی است که می‌تواند به مدل‌سازی توانایی مدل کردن ساده سطح جزییات لازم را برای تحریک سیستم تحت مطالعه ارائه دهد.

۸-۳-۵ مدل‌سازی ترکیبی

مدل‌سازی ترکیبی به شبیه‌سازی مدل‌سازی که در آن ویژگی‌ها در تکنیک‌های قبلی با زبان برنامه نویسی معمولی ترکیب می‌شوند اشاره دارد. این فرم مدل‌سازی می‌تواند تا جایی که همه چیزها را در یک زبان معمولی انجام دهد و به سطوح پایین‌تر مدل‌سازی ارائه واسط زبان معمولی را اجازه دهد ساده کند. بیشتر زبان‌های شبیه‌سازی ابزارهایی را برای درج کد زبان برنامه‌نویسی منظم در خود ارائه می‌دهند. بنابراین، آن‌ها می‌توانند انواع تکنیک‌ها را در نظر بگیرند.

۸-۴ زبان شبیه سازی

با افزایش استفاده از شبیه سازی، رشد زبان‌های شبیه سازی جدید نیز افزایش یافت. زبان شبیه سازی توسعه یافت چرا که نیازهای منحصربه‌فرد جوامع مدل‌سازی روتین‌های سیستمی برای حفظ دنباله زمانی، حفظ حالت شبیه سازی، جمع آوری آماره، ارائه محرک و تراکنش کنترل است. همه این موارد توسط هر فرد برنامه نویسی صورت می‌گیرد.

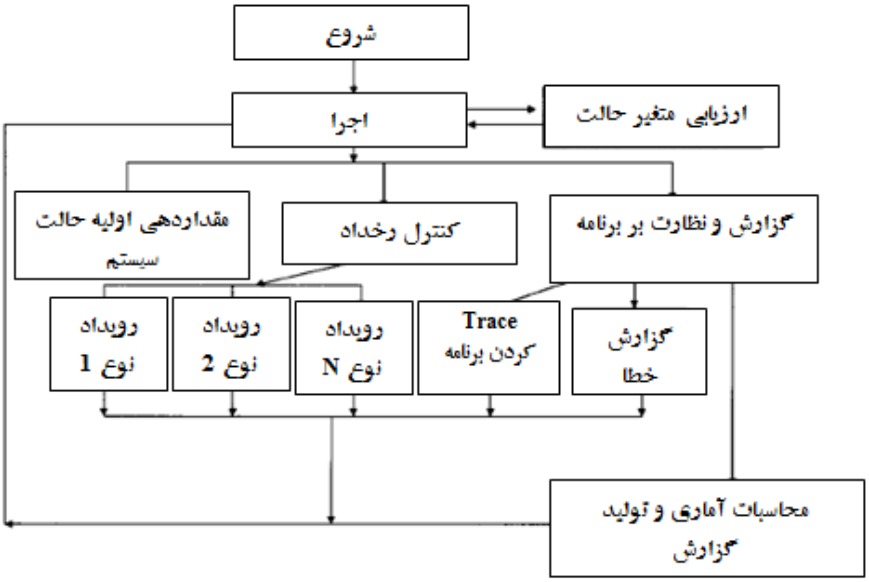
زبان‌های اولیه سرویس‌های پایه‌ای را با افزودن یک روتین قابل فراخوانی از زبان برنامه نویسی فراهم می‌کند. این زبان‌های اولیه مدیریت زبان و رخداد را فراهم می‌کنند. این فصل به چهار زبان می‌پردازد و بر جنبه‌هایی که فرآیند شبیه سازی را هدف قرار می‌دهد بحث می‌کند.

GASP IV ۱-٤-۸

GASP IV در اوایل دهه ۱۹۷۰ به‌عنوان زبان شبیه سازی با هدف عمومی توسعه‌یافته بود و هنوز با وجود تغییرات امروزی مورداستفاده است. از آن، امروزه به‌عنوان مدل پایه برای بیشتر زبان‌های امروزی استفاده می‌کنیم. GASP IV زبان شبیه سازی مبتنی بر FORTRAN است که روتین‌ها و ساختارهایی را برای حمایت از نوشتن رخدادهای گسسته، رخدادهای گسسته ترکیب شده و پیوسته، مدل‌های شبیه سازی پیوسته ارائه می‌دهد. مدل‌های رخداد گسسته در GASP IV به‌عنوان مجموعه از سیستم هر کاربر زیر روتین FORTRAN نوشته شده‌اند. GASP IV برای کاربران روتین‌های زیر را فراهم می‌کند: مدیریت زمان، مدیریت فایل(فایل‌های رخداد، ذخیره و بازیابی، مقابله با رخدادها، یافتن رخدادها)، و تحلیل و جمع‌آوری داده (متغیرهای مبتنی بر زمان و مشاهدات). کاربر باید یک مجموعه از روتین‌های رخداد را توسعه دهد که روابط منطق ریاضی را برای مدل‌سازی تغییرات در حالات متناظر با هر رخداد و تراکنش آن‌ها تشریح و تعریف می‌کند.

به‌عنوان یک مثال از ساختار و استفاده از GASP IV، مسئله صندوقدار بار دیگر بررسی می‌شود. به‌منظور مدل کردن این مسئله در GASP ما باید رخدادهای مورد نظر، ساختار آن‌ها، مرزها، را که هدف قرار داده‌شده‌اند را تعیین کنیم. برای ساده سازی مثال، فرض می‌شود که هیچ تأخیر زمانی بین پایان سرویس برای یک مشتری و آغاز دیگری (اگر هیچ انتظاری صورت نگیرد) وجود ندارد. از این دو رخداد سیستم یک ورود و یک پایان سرویس کارمند رخ می‌دهد. این‌ها به‌عنوان نقطه‌ای انتخاب‌شده اند که در آن تغییرات مهمی برای یک وضعیت سیستم رخ می‌دهد. فعالیتی که رخ می‌دهد آغاز سرویس است، این می‌تواند فرض شود که در زمانی که مشتری به خط تهی می‌رسد یا زمانی که کارمند سرویس را برای مشتری تکمیل میکند، رخ می‌دهد.

موجودیت‌ها در GASP با آرایه‌ای از صفات نشان داده می‌شوند، که در آن صفات اطلاعات توصیفی را در مورد موجودیتی نشان می‌دهند که مدل‌ساز می‌خواهد حفظ کند.



شکل (۸-۳): مدل پایه کنترل GASP IV

موجودیت‌ها عناصری هستند که در طول شبیه‌سازی بر روی آن‌ها کار می‌شود. صفات آن‌ها بر اساس وقوع منافع تنظیم می‌شود. یک متغیر "busy" برای نشان دادن وضعیت‌های کارمند، و صفات (۱) مشتری برای نشان گذاری زمان ورود به خط کارمند استفاده می‌شود. برای عملی کردن شبیه سازی، حالت سیستم باید برای برخی از مقادیر شناخته شده مقدار دهی شود، در این مورد کارمند به busy مقدار دهی نمی‌شود و اولین مشتری رسیده باید برای ورود زمان بندی شود. علاوه بر این، برای حفظ حالت اجرای مدل، مشتری ورودی باید دیگر ورود مشتریان را در آینده بر اساس توزیع زمان تصادفی انتخاب شده زمان بندی کند. متغیرها در زمانی که سرویس بر اساس طول زمان انتظار، تعداد مشتریان در حال انتظار، در حال سرویس‌دهی، و در کل تکمیل می‌شود. زمانی که به کد GASP نگاه می‌کنیم نیازمند بررسی ساختار برنامه GASP معمولی (شکل ۸-۳) هستیم. همان‌طور که در شکل نشان داده‌شده است، GASP IV به‌عنوان یک برنامه منحصربه‌فرد در FORTRAN وجود دارد.

Dimension nset (۱۰۰۰)

```
common/gcom/atrib (۱۰۰, DP(۱۰۰), DDL (۱۰۰, DTNOW,
II,MFA,MSTOP,NCLNR,NCRDR,NPRINT, NNRUN,
NNSET, NTAPE,SS(۱۰۰),SSL(۱۰۰),TNEXT,TNOW,XX(۱۰۰)
Common Q Set
Equivalence Nset(۱ ),Qset(۱))
NNSet=۱۰۰۰
NCRDR=۵
NPRINT=۶
NTAPE=۷
Call GASP
Stop
End
```

شکل (۸-۴): برنامه GASP IV main FORTRAN

```
Subroutine Event (I)
Goto (۱،۲), I
 ۱Call arrival
   return
 ۲Call end SRU
   return
End
```

شکل (۸-۵): رخداد زیر روتین برای مسئله صندوقدار

بنابراین، ساخت تابع نیازمند یک برنامه اصلی و فراخوانی برنامه GASP است که با شبیه‌سازی آغاز می‌شود. تابع دیگر برنامه اصلی تنظیم محدودیت‌های سیستم، مانند تعداد فایل ها، ورودی، خروجی، محدودیت بر رخدادها و غیره است. شکل (۸-۴) برنامه اصلی را برای مثال نشان می‌دهد.

زمانی که GASP فراخوانی شود، برنامه باید تحت کنترل زیر برنامه GASP با یک سری فراخوانی‌ها، اجرا شود. اجرای سیستم GASP و مقداردهی اولیه شبیه سازی با استفاده از روتین نوشته شده توسط کاربر I n t l c. صورت می‌گیرد. در زمان مقدار دهی اولیه، این شبیه سازی با بررسی لیست رخدادها آغاز می‌شود، اگر هر رخداد موجودی، انتخاب شود، اجرا شود، و متغیرها را اتخاذ کند. حلقه تا زمانی که شرط انتهایی برآورده شود یا خطا رخ دهد ادامه می یابد. برای کنترل رخدادها یا بررسی صحت آن‌ها در FORTRAN نیازمند این هستیم که کاربر یک روتین توالی رخداد به نام Event را تأمین کند. این روتین کنترل رخداد با تعداد صفات رخداد مورد نظر فراخوانی می‌شود. این از این برای فراخوانی رخداد واقعی استفاده می‌کند. برای مثال صندوقدار ما این روتین در شکل (۸-۵) را با دو رخداد نشان داده‌شده تشریح می‌کند. زمانی که این روتین در تعداد مناسبی فراخوانی می‌شود، رخداد مورد نظر نیز فراخوانی، اجرا می‌شود و کنترل به روتین رخداد باز می‌گردد، که کنترل را به روتین اجرا باز می‌گرداند.

این رخدادها بر اساس آنچه Filem (۱) در آن ذخیره کرده است فراخوانی می‌شوند. Filem (۱) در مبنای اول ورود اول خدمت، عمل می‌کند و آیتم‌ها را در یک زمان حذف می‌کند. رخدادها را در Filem (۱) به‌عنوان صفت ذخیره میکند. صفت (۱) زمان رخداد، صفت (۲) نوع رخداد، و همه صفات دیگر توسط کاربر برای هر موجودیت اضافه شده است. شکل (۸-۶) مثالی در مورد اینکه چگونه فایل مقدار دهی اولیه می‌شود ارائه می‌دهد.

```
Subroutine INTLC
common/gcom/atrib (۱۰۰, DP(۱۰۰), DDL (۱۰۰, DTNOW,
I I,MFA,MSTOP,NCLNR,NCRDR,NPRINT, NNRUN,
NNSET,NTAPE,SS(۱۰۰),SSL(۱۰۰),TNEXT,TNOW,XX(۱۰۰)
Equivalence (xx(۱), Busy)
```

```
Busy=۰
Atrib(۲)=l
Call filem
return
end
```

شکل (۸-۶): زیر روتین Intlc برای مسئله صندوقدار

```
Subroutine Arrival
common/gcom/atrib (۱۰۰, DP(۱۰۰), DDL (۱۰۰, DTNOW,
II,MFA,MSTOP,NCLNR,NCRDR,NPRINT, NNRUN,
NNSET,NTAPE,SS(۱۰۰),SSL(۱۰۰),TNEXT,TNOW,XX(۱۰۰)
Equivalence (xx(۱), Busy) TIMST(Busy,TNOW,ISTAT)
Att rib(۱)=TN OW+expon(۲۰., ۱)
Attrib(۲)=l
Call filem(۱)
Call filem(۲,attrib(۱))
if (Busy=۰) go to return
۱۰ Busy =۱
attrib (۱)=TNOW+u nfrm (۱۰.,۲۵،۱)
attrib(۲)=l
attrib(۳)=TNOW
Call filem(۱)
return
end
```

شکل (۸-۷): کد رخداد روتین Arrival

شکل (۸-۶) روتین مقداردهی اولیه برای شبیه‌سازی صندوقدار را تشریح می‌کند. Filem (۱)، فایل رخداد، با اولین رخداد ورودی (مشتری) بارگذاری می‌شود، و کارمند به not busy تنظیم می‌شود. زمانی که Filem (۱) یک رخداد ذخیره شده دارد، یک شبیه‌سازی می‌تواند آغاز شود. اولین رخداد یک ورودی مشتری است که با محتوای صفت (۲) از Filem (۱) نشان داده‌شده است، که تنها رخداد در این زمان است. رخداد ورود (شکل ۸-۷) زمان بندی برای ورود بعدی را اجرا می‌کند.

بعد از اینکه ورود بعدی زمان بندی شد، ورود حاضر در صف قرار می‌گیرد. سپس یک تست برای دیدن اینکه آیا کارمند busy است یا نه انجام می‌شود. اگر این چنین باشد، ما به برنامه اصلی باز می گردیم یا در غیر این صورت پایان رخداد سرویس را برای زمان حاضر به علاوه تعداد انتخاب یکنواخت بین ۱۰ و ۲۵ زمان بندی می‌کنیم.

رخداد دوم، پایان سرویس، در شکل (۸-۸) نشان داده‌شده است. این کد متغیر زمان در سیستم و متغیر busy را تعیین می‌کند. کد این بررسی می‌کند که آیا هیچ کاربری در صف وجود دارد، اگر وجود داشته باشد یکی بر می‌دارد، و انتهای سرویس دیگری را برای این کار زمان بندی می‌کند.

```
Subroutine end SRU
common/gcom/atrib (۱۰۰, DP(۱۰۰), DDL (۱۰۰, DTNOW,
II,MFA,MSTOP,NCLNR,NCRDR,NPRINT, NNRUN,
NNSET, NTAPE,SS(۱۰۰),SSL(۱۰۰),TNEXT,TNOW,XX(۱۰۰)
TIMST(subusy,TNOW,T, ISTAT)
TSYS=TNOW-Attrib(۳)
Call col CT(TSYS, ۱ )
if(NNQ(۲), ۶T,۰) go to ۱۰
busys=۰
return
```

```
۱۰Call schd((۲,unfrm(۱۰.,۲۵،۱ ) attrib)
return
end
```

شکل (۸-۸): پایان روتین service

GPS ۲-٤-۸

سیستم شبیه سازی با هدف عمومی (GPSS)یک زبان شبیه سازی فرآیند گرا برای مدل‌سازی سیستم‌های گسسته است. این سیستم از یک مفهوم ساختار بندی بلوکی برای ساخت مدل استفاده می‌کند. مجموعه‌ای از بلوک‌های استاندارد (شکل ۸-۹) است که کنترل و عملیاتی را برای تراکنش‌ها (موجودیت ها) فراهم می‌کند. یک مدل برای برنامه GPSS با انتخاب بلوک‌هایی برای نمایش مؤلفه‌های مدل و پیوند آن‌ها در یک نمودار بلوکی با تعریف ساختارهای منطقی سیستم تفسیر می‌شود. GPSS دیاگرام بلوکی تعریف‌شده توسط کار را تفسیر و اجرا میکند، در نتیجه شبیه سازی‌هایی را ارائه می‌دهد. این تفسیر آهسته است، بنابراین، زبان نباید برای حل مسائل بزرگ‌تر استفاده شود.

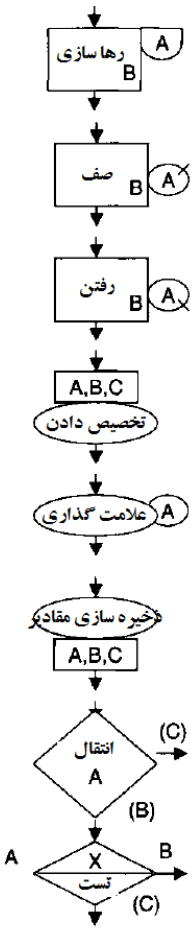
برای تشریح GPSS ما بار دیگر سیستم صندوقدار را بررسی می‌کنیم. این به‌عنوان یک سیستم صف تک سرور با کارمند ما و n مشتری ورودی دیده میشود (شکل ۸-۱۰). مشتریان رسیده با یک میانگین زمان بین ورودی‌های ۱۰ دقیقه ای به‌صورت نرمال توزیع‌شده‌اند. کارمند برای مشتریان در یک زمان یکنواخت بین ۵ و ۱۵ دقیقه سرویسی را فراهم می‌کند. شبیه سازی متغیرهایی را در مورد طول صف، بهره وری کارمند، و زمان در سیستم اتخاذ می‌کند. شبیه ساز نمودار نشان داده‌شده در شکل (۸-۱۰) را از مدل سیستم صف بر اساس متغیرهای در نظر گرفته شده نشان می‌دهد. این ساختار سپس به آنچه در شکل (۸-۱۱) نشان داده‌شده، تفسیر می‌شود. کد به چهار بخش تقسیم می‌شود. بخش بالایی برای تعریف داده موردنیاز برای تقریب یک توزیع نمایی و راه اندازی نشانه گذارهایی برای متغیرهای وابسته زمانی استفاده می‌شود.

بخش دوم کد شبیه سازی اصلی است، وظایف تولید مشتری (۱۴)، اتخاذ متغیرهای زمان ورود (۱۵)، صف کردن مشتریان ورودی (۱۶)، زمان بندی سرویس (۱۷)، خروج از خط انتظار (۱۸)، به تأخیر انداختن خروج با زمان سرویس‌دهی مناسب به کارمند (۱۹)، آزادسازی کارمند برای مشتری بعدی (۲۰)، اتخاذ متغیرها برای زمان مشتری در سیستم (۲۱)، و پایان سیستم (۲۷) وجود دارد.

بخش سوم یک بخش زمان بندی است و برای زمان بندی پایان روتین سرویس استفاده می‌شود. مدل یک تراکنش جعلی را در زمان ۴۸۰ زمان بندی می‌کند، که باعث پایان دادن دستورالعمل‌ها برای اجرا می‌شود (تنظیم شمارنده به صفر). بخش چهارم، بخش کنترل، با شبیه سازی شمارنده و کنترل بر بخش مدل آغاز می‌شود.

نماد بلوک	توصیف عملکرد بلوک
ایجاد A,B,C,D.E.F,G	ایجاد تراکنش تعیین شده توسط عملوندهای A، B، C، D، E، F و G
پایان A	حذف تراکنش رسیده و کاهش شمارنده پایانی با A
تسریع کردن A,B	تسریع کردن زمان شبیه سازی تعیین شده با عملوندهای A و B
تصاحب کردن A	باعث می شود تراکنش ها منتظر بمانند و امکان A را بگیرند.
آزاد سازی A	آزاد سازی امکان (فرصت) A
ورود B A	باعث می شود تراکنش منتظر بماند و B واحد از مخزن A را تسخیر کند.

شکل (۸-۹): بلوک مؤلفه‌های مدل‌سازی GPSS پایه



آزاد سازی B واحد از مخزن A

افزایش B واحد به تعداد صف A

کاهش B واحد از تعداد صف A

تخصیص مقدار مشخص B با اصلاح کننده C به‌پارامتر شماره A تراکنش

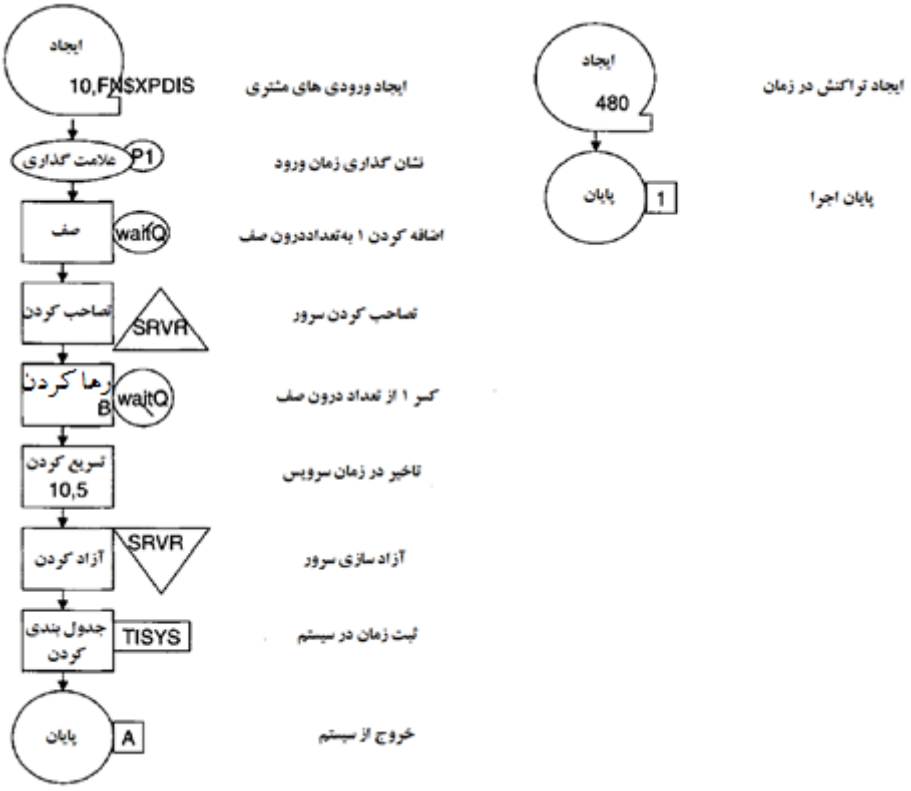
تخصیص زمان کلاک کنونی به پارامتر شماره A

تخصیص مقدار مشخص شده B برای ذخیره محل مقدار A

موجب انتقال به محل C با احتمال A، و محل B با احتمال 1-A

موجب انتقال به محل C اگر A مطابق با عملوند X به B مربوط نباشد

ادامه شکل (۸-۹): بلوک مؤلفه‌های مدل‌سازی GPSS پایه



شکل (۸-۱۰): مدل GPSS برای مسئله صندوقدار

۱ *

۲ Simulate

۳ *

۴ XPDIS function RN۱ ,C۲۴

۵ ۰,۰, ۰/۰,۱, ۰,۱۰۴/۰,۲, ۰,۲۲۲/۰,۳, ۰,۳۵۵/۰,۴, ۰,۵۰۹/۰,۵, ۰,۶۹

۶ ۰,۶, ۰,۹۱۵/۰,۷, ۱,۲/۰,۷۵, ۱,۳۸/۰,۸, ۱,۶/۰,۸۴, ۱,۸۳/۰,۸۸, ۲,۱۲/۰,۹

```
۷    ۲,۳/۰,۹۲, ۲,۵۲/۰,۹۴, ۲,۸۱/۰,۹۵, ۲,۹۹/۰,۹۶, ۲,۳/۰,۹۷, ۳,۵/۰,۹۸
۸    ۴,۰/۰,۹۹, ۴,۶/۰,۹۹۵, ۵,۳/۰,۹۹۸, ۶,۲/۰,۹۹۹, ۰,۷/۰,۹۹۹۷, ۸
۹    *
۱۰   TISYS table MP۱،۰،۵،۲۰
۱۱   *
۱۲   *model segment
۱۳   *
۱۴   Generate ۱۰, FN$XPDIS
۱۵   Mark P۱
۱۶   Queue Waitq
۱۷   Seize SRVR
۱۸   Depart Waitq
۱۹   Advance ۱۰،۵
۲۰   Release SRVR
۲۱   Tabulate TISYS
۲۲   Terminate
۲۳   *
۲۴   *timing segment
۲۵   *
۲۶   Generate ۴۸۰
۲۷   Terminate ۱
```

شکل (۸-۱۱): کد GPSS برای مسئله صندوقدار

این مثال برخی از ویژگی‌های GPSS را نشان می‌دهد. GPSS یک روش مدل‌سازی ساده است که به‌صورت گسترده استفاده شده است. به‌هرحال، این زبان به واسطه عملیات تفسیری خود با شکست مواجه شد، که آن را بی نهایت آهسته ساخته بود.

۸-٤-۳ Simscript

Simscript در اواخر دهه ۱۹۶۰ به‌عنوان زبان شبیه سازی عمومی توسعه یافت. این زبان فریم ورک مدل‌سازی شبیه سازی گسسته‌ای را با سینتکس سازی آزادانه مانند انگلیسی، برای هر مدل خوانا و خود مستند ساز فراهم کرد. Simscript دو نوع موجودیت را حمایت می‌کند: موقتی و دائمی. برای مثال در مسئله صندوقدار، کارمند دائمی است و مشتری موقتی است. موجودیت دائمی برای کل مدت زمان شبیه سازی وجود دارد، در حالی که موجودیت موقتی در طول شبیه سازی می‌آید و می‌رود. صفات موجودیت نام دارند، قابلیت خوانایی و معانی آن‌ها افزایش می‌یابد.

یک شبیه‌سازی Simscript بر اساس این سه بخش است: یک مقدمه، یک برنامه اصلی، یک زیر برنامه رخداد. مقدمه مؤلفه‌های مدل (موجودیت، متغیر، آرایه). را تعریف می‌کند. برنامه اصلی همه عناصر را برای آغاز شبیه سازی مقدار دهی میکند. رخدادها، رخدادهای کاربر استفاده شده برای مدل کردن سیستم را تعریف می‌کنند. برای تعریف این مؤلفه‌ها ما از مسئله صندوقدار استفاده می‌کنیم. ما فرض می‌کنیم ورودها به‌صورت میانگین ده دقیقه‌ای هستند و به‌صورت نمایی توزیع‌شده اند، و زمان سرویس‌دهی کارمند به‌صورت یکنواختی بین ۵ و ۱۵ دقیقه توزیع‌شده است. شکل (۸-۱۲) کد این مسئله را نشان می‌دهد. این شکل بسیاری از ویژگی‌های Simscript را به شرح زیر نشان می‌دهد:

- خط ۲ زمان انتظار را به‌عنوان موجودیت سیستمی نشان می‌دهد که دارای متغیرهای مرتبط با آن است، و این یک موجودیت دائمی است چرا که به‌صورت موقتی نشان داده نشده است. بنابراین، می‌توانیم متغیرهایی را بر روی عمر مدل پیدا کنیم.
- خط ۳ یک موجودیت موقتی مشتری را تعریف می‌کند و نشان می‌دهد که به زمان انتظار تعلق دارد.
- خط ۶ و ۷ نام و صفات رخدادها را تعریف می‌کند.

■ خط ۹ و ۱۴ متغیرهای در نظر گرفته شده بر این نهاد را تعریف می‌کند.

برنامه یا بخش اصلی در بخش B نشان داده‌شده است. این بخش شرایط اولیه را راه‌اندازی میکند. سه بخش بعدی رخدادهای ورود، خروج و توقف را تعریف می‌کند. رخداد ورود، ورود بعدی را برای حفظ جریان رخداد زمان بندی میکند، و یک مشتری ایجاد می‌کند، به آن اطلاعات زمانی می‌دهد، و آن را در خط انتظار می‌گذارد، و یک سرویس کارمند را از خط تهی برنامه ریزی می‌کند. رخداد خروجی زمان مشتری در بانک را محاسبه می‌کند، مشتری را حذف میکند، و مشتری بعدی را زمان بندی میکند. خروجی رخداد توقف متغیرهای جمع آوری شده است.

A:
۱ Preamble
۲ the system owns a wait line and has a status temporary entities
۳ every customer has an enter time and may belong to the wait line
۴ event notices include arrival and stop simulation
۵ every departure has a teller
۶ define busy to mean ۱
۷ define idle to mean ۰
۸ define time in bank as a real variable
۹ tally no customers as the number, AV time and the mean,

D:
۱ event departure given customer
۲ define customer as an integer variable
۳ let time in back=۱۴۴۰.*(time v-enter time(customer)
۴ destroy the customer
۵ if the wait line is empty
۶ let status=idle
۷ return
۸ else
۹ remove the first customer from the wait line
۱۰ schedule a departure given customer in Uniform F(۵., ۱۵., ۱) minutes

۱۰ and Var time as the variance of time in bank
۱۱ accumulate avg util as the mean, and Var util as the
۱۲ Variance of status
۱۳ accumulate Ave waitline length as the mean, and
۱۴ var waitline length as the variance of N wait line
۱۵ end
B:
۱ main
۲ let status=idle
۳ schedule an arrival now
۴ schedule a stop simulation in ۸ hours
۵ start simulation
۶ end
C:
۱. Event arrival
۲ schedule an arrival in exponential F(۱۰.,۱) minutes
۳ create a customer
۴ let enter time (customer)=time V
۱۱ return
۱۲ end
E:
۱ event stop simulation
۲ start new page
۳ skip ۵ lines
۴ print ۱ line thus
۵ single teller wait line example
۶ skip ۴ lines
۷ print ۳ lines with no customers, av time, and var time thus
۸ Number of customers = * . . . .
۹ Average time in bank = ****. ....
۱۰ Variance of time in bank = ****. ....
۱۱ skip ۴ lines
۱۲ print ۲ lines with avg util and var util thus
۱۳ Average teller utilization = ****. ....
۱۴ Variance of utilization = ****. ....

```
۵ if status=busy
۶ file the customer in the wait
line
۷ return
۸ else
۹ let status=busy
۱۰ schedule a departure given
customer in Uniform F(۵۶.,۱۵.,۱)
minutes
۱۱ return
۱۲ end
۱۵ Skip ۴ lines
۱۶ print ۲ lines with avg queue
length and var queue length thus
۱۷ Average wait line length =
****. ....
۱۸ Variance of wait time = ****.
....
۱۹ stop
۲۰ end
```

شکل (۸-۱۲): کد Simscript صندوقدار

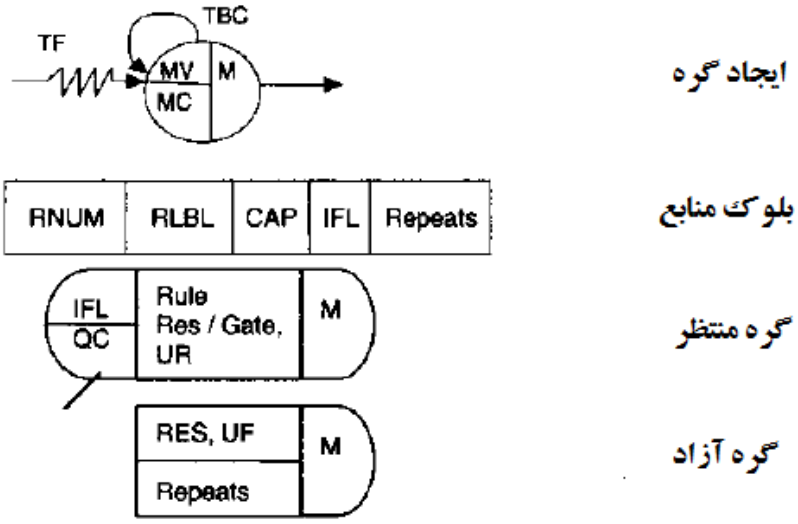
Slam II ٤-٤-٨

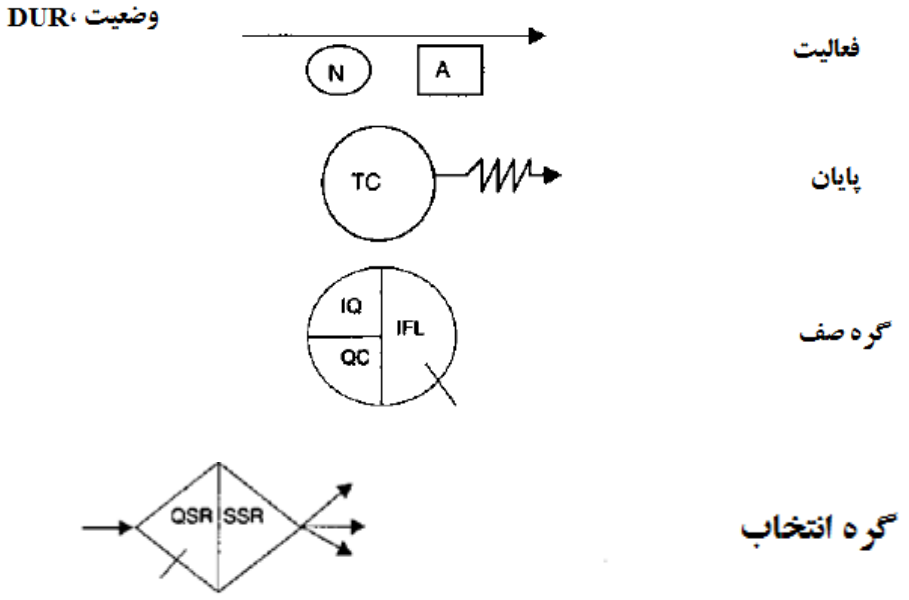
Slam II، یک زبان شبیه سازی برای مدل‌سازی جایگزین است، که توسط Pritsker and Associates، West Lafayette، Indiana، در اواخر دهه ۱۹۷۰ توسعه داده شد. این یک زبان مدل‌سازی ترکیبی است که برای تحلیل شبکه صف بندی، مدل‌سازی رخداد گسسته و پیوسته در یک فرم یکپارچه ارائه شد. Slam II ویژگی‌هایی را برای ادغام سه فرم ارائه می‌دهد.

در بالاترین هدف مدل‌ساز می‌تواند از یک ساختار شبکه شامل گره‌ها و انشعاب‌های نشان‌دهنده صف، سرور، و نقاط تصمیم‌گیری برای ساخت یک مدل از سیستم شبیه‌سازی‌شده استفاده کند. این، به نوبه خود، می‌توانند به آسانی به کد Slam II تفسیر شود. علاوه بر این، Slam توانایی ترکیب رخدادها و مدل‌های پیوسته با مدل‌های شبکه را با استفاده از گره‌های رخداد فراهم کند که کد رخداد را برای مدل‌های گسسته یا پیوسته فراخوانی میکند. مانند زبان قبلی، مدل‌های Slam رخداد گرا مجموعه‌ای از رخدادها و تغییرات بالقوه را ایجاد کرده‌اند که می‌تواند با هریک از آن‌ها رخ دهد. این رخدادها تعریف

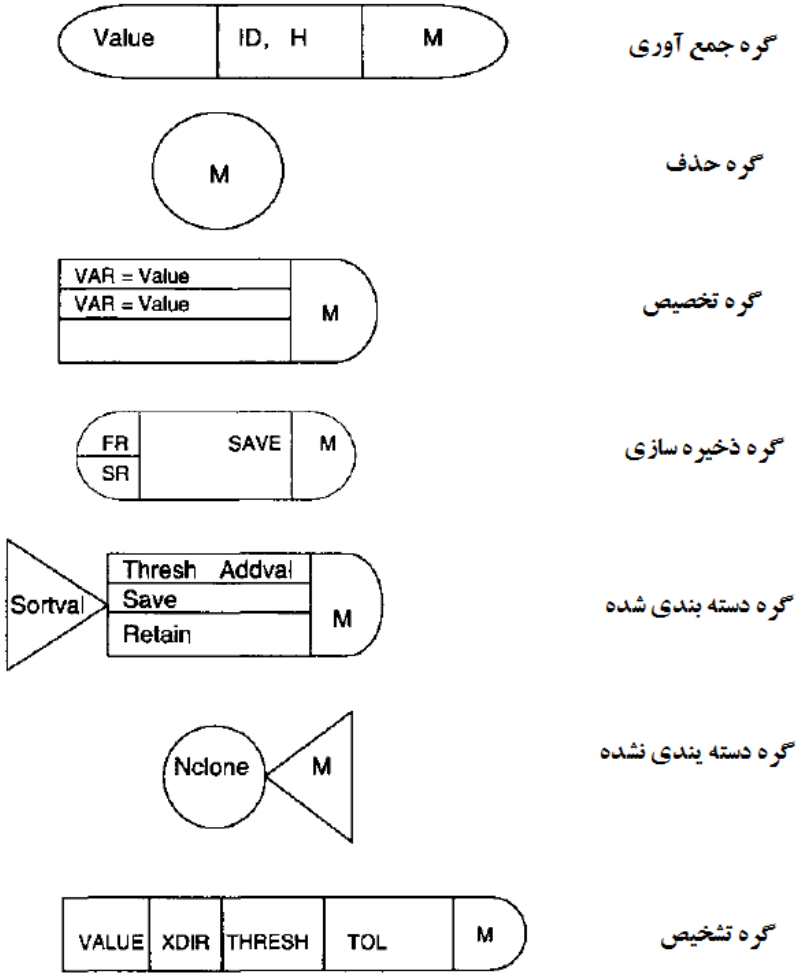
می‌کنند که چگونه مدل رخداد و تغییر حالت را تفسیر می‌کنند. Slam مجموعه‌ای از زیر نامه‌ها را برای کمک به مدل‌ساز رخداد گرا ارائه می‌دهد. همان‌طور که در GASP دیدیم، مدل‌های پیوسته Slam با مشخص کردن مجموعه‌ای از معادلات دیفرانسیلی یا تمایزی که رفتار پویایی متغیرهای حالت را تشریک می‌کند، ساخته می‌شود. این معادلات در FORTRAN با استفاده از متغیرهای حالت کد می‌شود.

Slam II از یک مجموعه نماد پایه برای تشریح سیستم مدل شده، مانند GPSS استفاده می‌کند. شکل (۸-۱۳) نمادهای Slam II پایه و عبارت کد مربوطه را نشان می‌دهد. تنها سه کاراکتر اول جمله نام‌ها و چهار کاراکتر اول برچسب‌های گره مهم هستند. آن‌ها در مثالی از صندوقدار استفاده خواهند شد. مانند قبل، ما می‌خواهیم از مشتریان رسیده به‌صورت میانگین هر ده‌دقیقه‌ای با یک توزیع نمایی استفاده کنیم که اولی در زمان ۰ شروع می‌شود. علاوه بر این، کارمند به مشتریان با توزیع یکنواخت از ۵ تا ۱۵ دقیقه سرویس میدهد. مدل شبکه Slam نتیجه در شکل (۸-۱۴) نشان داده‌شده است. ارجاع به گره از طریق برچسب گره صورت می‌گیرد. زمانی که برچسب گره نیاز است، در یک مستطیل قرار می‌گیرد و به مبنای نماد اضافه می‌شود.

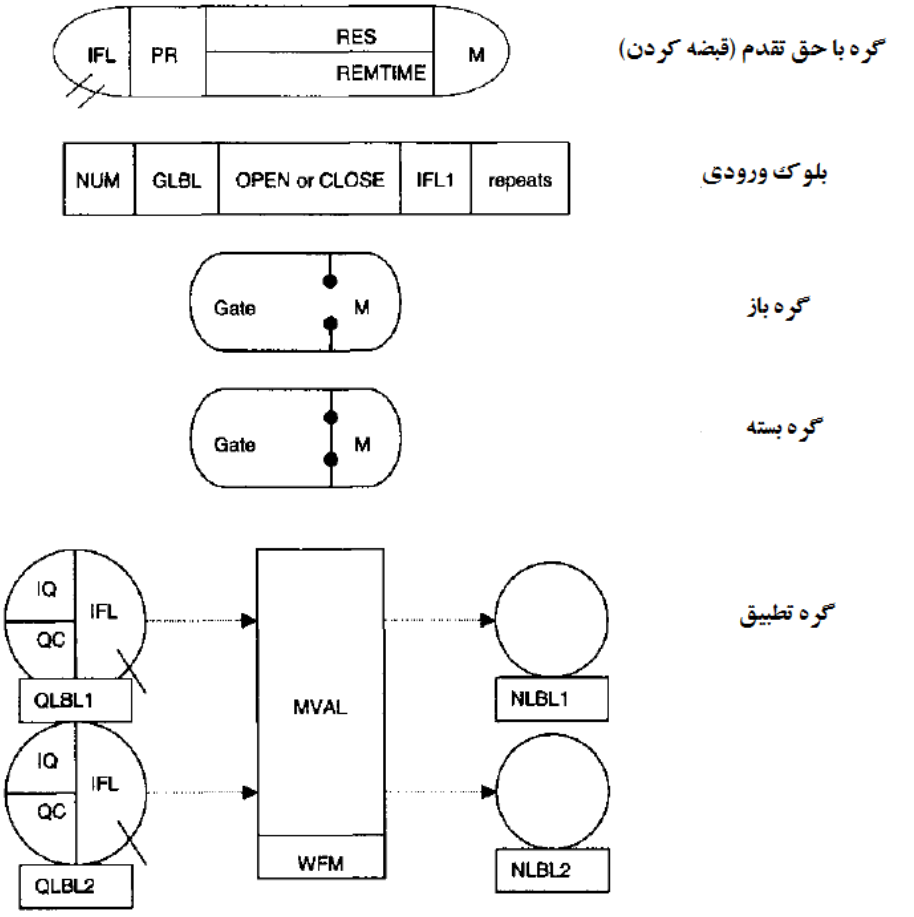




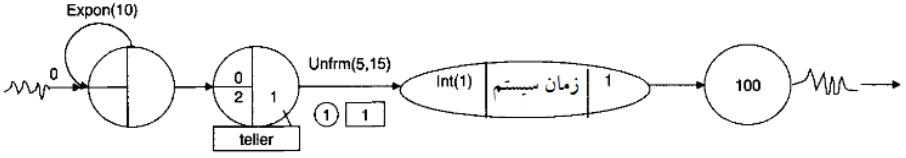
شکل (۸-۱۳): عبارات و نمادهای پایه برای مدل‌های Slam



ادامه شکل (۸-۱۳): عبارات و نمادهای پایه برای مدل‌های Slam



ادامه شکل (۸-۱۳): عبارات و نمادهای پایه برای مدل‌های Slam



شکل (۸-۱۴): مدل شبکه مسئله صندوقدار بانک Slam II

```
۱    Gen, Fortier, Bankteller, ۵/۲۲/۲۰۰۲،۱;
۲    Limits, ۲،۱،۱۰۰
۳    Network
۴    Create, Expon(۱۰.), ۰, ۱;
۵    Teller Queue(۱ ),۰,-;
۶    Activity (۱)/۱, Unifrm(۵., ۱۵.);
۷    Term ۱۰۰; Colct, Ini(۱ ), system
                time,,۱;
۸    end networks;
```

شکل (۸-۱۵): کد مسئله صندوقدار Slam II

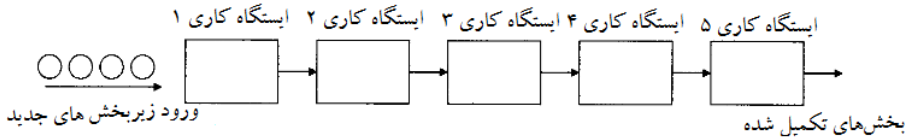
کد برای این شبکه در شکل (۸-۱۵) نشانه داده‌شده است. خط اول کد مدل ساز ، نام مدل، و تاریخ و نسخه آن را تعیین می کند. خط دوم محدودیت های مدل و فایل‌هایی بر صفت USR و ۱۰۰ موجودیت همزمان را در سیستم تعریف می کند. خط ۳ این کد را به‌عنوان کد شبکه شناسایی میکند، و خط ۴ به مشتریان میانگین ده دقیقه‌ای به‌صورت نمایی توزیع‌شده را ارائه می‌دهد. خط ۵ صف ۱ را به‌عنوان یک کارمند بدون مشتری اولیه در صف خود، یک صف نامحدود با سرویسی که به‌صورت یکنواخت از ۵ تا ۱۵ دقیقه توزیع‌شده است،تعریف می کند. خط ۷ متغیرها و زمان را در سیستم از موجودیت‌ها در زمانی که سرور را تکرار می کنند در نظر میگیرند. خط ۸ نشان می‌دهد که شبیه سازی برای ۱۰۰ موجودیت انجام‌شده است و سپس خاتمه یافته است.

این کد بی‌نهایت ساده است و انعطاف‌پذیری زیادی در نشان دادن چگونگی بسط سیستم دارد. برای نگاه به عملیات کارمند در جزییات بیشتر، صف می‌تواند با یک گره رخداد و کد برای رخداد کارمند آمده در مدل جایگزین شود.

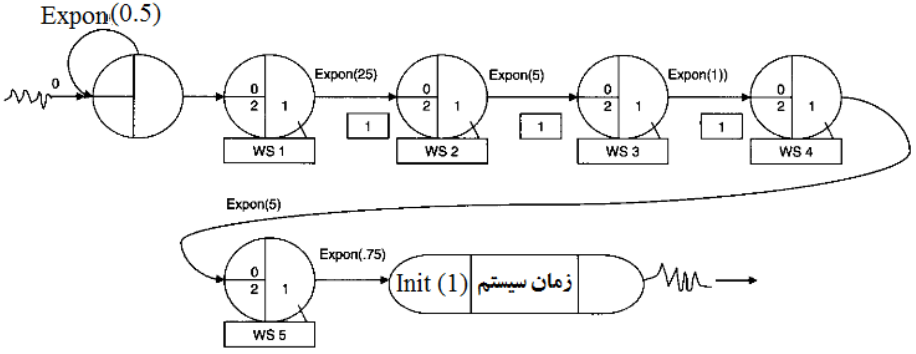
۸-۵ کاربرد شبیه‌سازی

برای تشریح استفاده از شبیه‌سازی چند مسئله نمونه داده‌شده است و مدل‌ها در زبان شبیه‌سازی Slam II توسعه داده‌شده است. نمونه اول یک کارخانه صنعتی با پنج ایستگاه تولیدی به روش خط مونتاژ است. مسئله می‌تواند به شکل (۸-۱۶) دیده شود.

کارخانه مونتاژهای اولیه را انجام می‌دهد و بعد از پنج مرحله خاتمه می‌یابد. یک اتاق ذخیره‌سازی در آغاز خط وجود دارد. متغیرهایی که می‌خواهیم تعیین کنیم استفاده از ایستگاه کاری، زمان برای پردازش در ایستگاه‌ها، تعداد واحدهای منتظر، و مجموع تولید شده است.



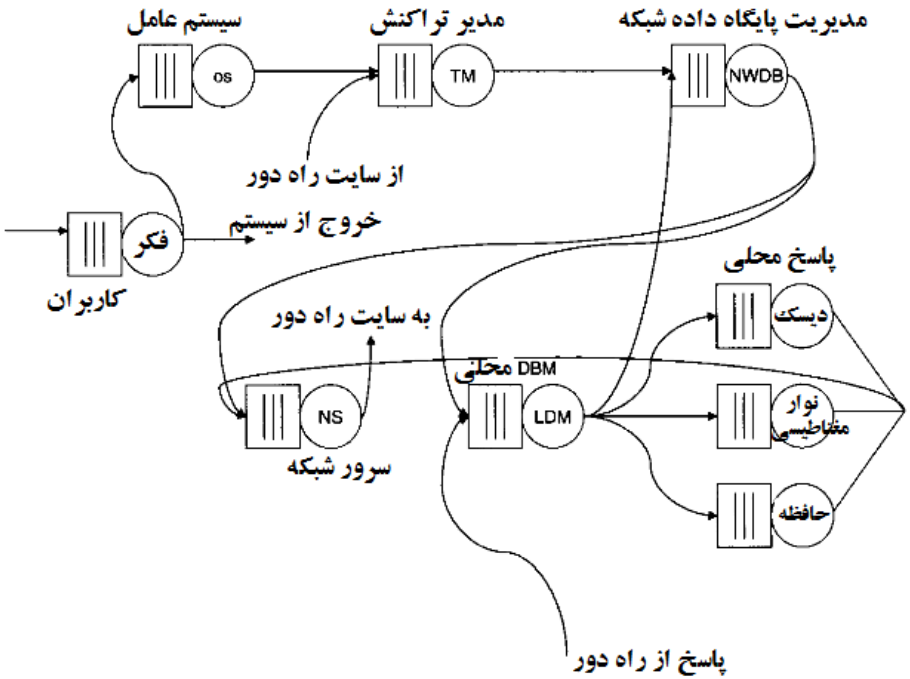
شکل (۸-۱۶): مثال خط مونتاژ



شکل (۸-۱۷): مدل شبکه Slam II برای مسئله خط مونتاژ

شبکه Slam نتیجه در شکل (۸-۱۷) نشان داده‌شده است. کد به ما اجازه می‌دهد که آیتم‌های مورد نظر را بدون در نظر گرفتن علت هر افتی از مدل مورد نظر بررسی کنیم.

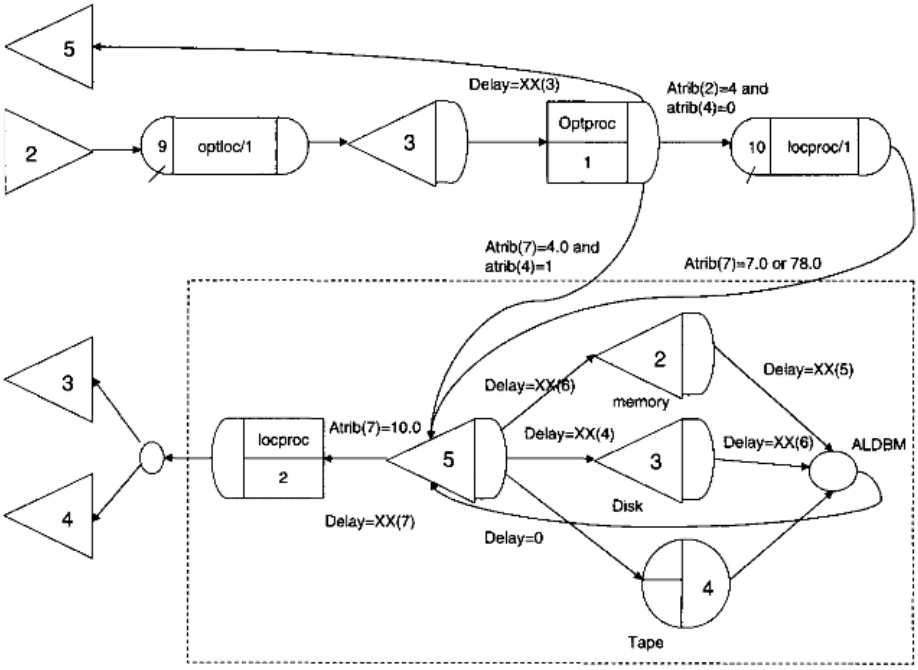
یک مثال دقیق‌تر اینکه چگونه شبیه‌سازی می‌تواند برای مدل کردن یک سیستم مدیریت پایگاه داده توزیع‌شده، استفاده شود. مدل در شکل (۸–۱۸) نشان داده‌شده است. با توجه به آن فرآیند یا سرورها در یک گره به تراکنش پایگاه داده کاربر سرویس می‌دهند. کاربران درخواستی را ارائه می‌دهند، و سیستم‌های عملیاتی با پایپ لاین درخواست‌ها پایگاه داده برای مدیران تراکنش سرویس می‌دهند، که به نوبه خود درخواست‌های کاهش یافته را برای پایگاه داده سرور شبکه ارائه می‌دهند، که تعیین می‌کند که دسترسی واقعی در کجا صورت می‌گیرد.

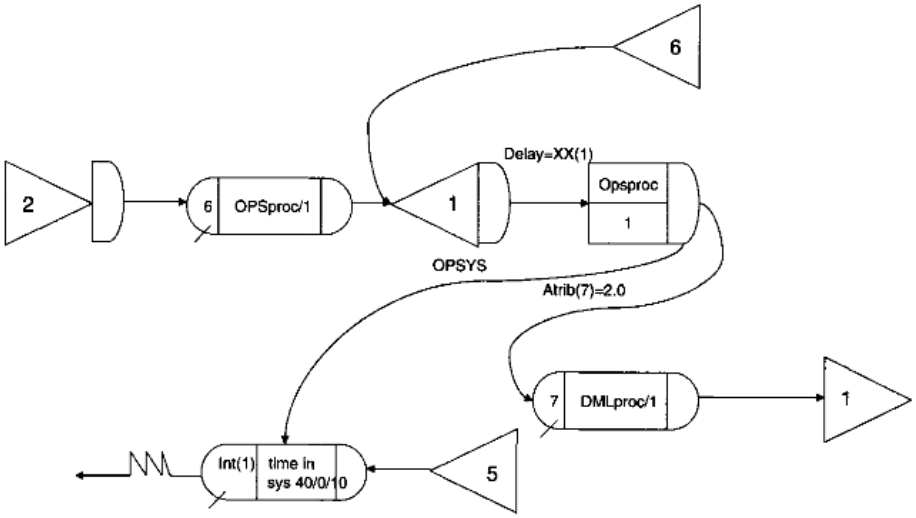


شکل (۸–۱۸): مدل صف بندی یک سیستم پایگاه داده توزیع‌شده

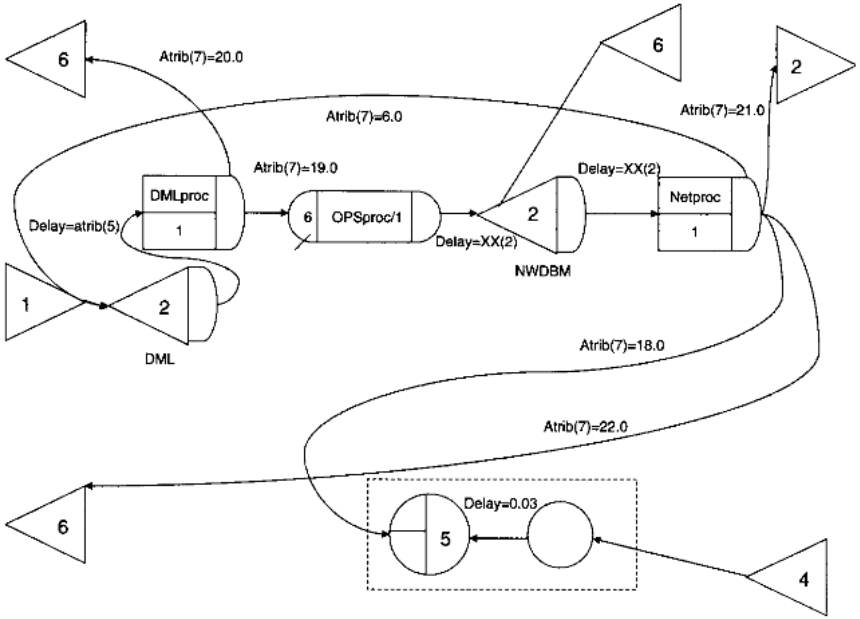
سایت محلی انتخاب شده سپس به اطلاعاتی از دستگاه مناسب دسترسی دارد. جزییات در هر سطح متناسب با مدل است. صف‌ها همه به‌عنوان رخدادها مدل شدند و کد لازم برای شبیه سازی آن‌ها توسعه

یافت. این شبیه سازی برای تحلیل الگوریتم بهینه سازی برای سیستم‌های پایگاه داده توزیع‌شده استفاده می‌شود. شبکه Slam در شکل (۸-۱۹) نشان داده‌شده است.





شکل (۸-۱۹): مدل شبیه سازی شبکه Slam II برای یک سیستم پایگاه داده توزیع‌شده



شکل (۸-۱۹): ادامه شکل

۸-۵-۱ برنامه شبیه سازی

برنامه شبیه سازی از تحقق مدل شبیه سازی تشکیل شده است. این به‌عنوان یک پکیج نرم افزار ماژولار ساخته شده است تا تبادل مؤلفه‌های مدیریت پایگاه داده شبیه سازی شده را بدون فشار اضافی بر دیگر مؤلفه‌های مدل اجازه دهد. برنامه شبیه سازی از یک مجموعه از شبکه Slam II و کد رخداد گسسته تشکیل شده است، که جنبه‌های محاسباتی اصلی یک سیستم مدیریت پایگاه داده توزیع‌شده را، همان‌طور که از قبل مشخص شد مدل می‌کند. برای ارائه قابلیت مدل کردن طیف وسیعی از توپولوژی‌های و معماری مدیریت پایگاه داده، مدل با مجموعه‌ای از جدول‌های اطلاعاتی که مشخصه توپولوژی شبکه و ارتباطات آن، محل آیتم داده، محتوای آیتم داده، و آماره استفاده را دارد به دست می‌آید. کد شبکه Slam II برای تحقق این مدل در شکل (۸-۲۰) نشان داده‌شده است. این کد به وضوح مؤلفه‌های اصلی برنامه شبیه سازی را نشان می‌دهد. علاوه بر این، توجه داشته باشید که EVENT۵ نشان می‌دهد که گره خاص یک نمایش صف ساده نیست، این یک کاهش در کد شبیه سازی رخداد گسسته را نشان می‌دهد. این رخداد بسط جزییات جنبه مدل را اجازه می‌دهد.

```
his is a list of the activities in
the network
۱ Enter to OPSYS ۲٠PSYS to
DML
۳ ٠PSYS to User ۴ Optimize
to LDBM
۵ Optimize to User ۶
NMMDBM to DML
۷ LDBM to memory ۸
LDBM to disk
```

```
Enter, ۱;
act/l;
Opsys await(۶),opsproc/۱;
event,۱; operating system
act,XX(۱);
free, opsproc/۱;
act/۳,,atrib(۷).eq.۳,user; service
completed
act/۲,,atrib(۷).eq.۲;
DML await(۷),DMLproc/۱;
event,۴;
```

۹ LDBM to tape ۱۰ LDBM to DLOC
۱۱ Memory to RLDBM ۱۲ disk to RLDBM
۱۳ Tape to RLDBM ۱۴ RLDBM to LDBM
۱۵DLOC to NMMDBM ۱۶ DLOC to REQN
۱۷REQN to NWDEL ۱۸ NWDEL to NWDBM
۱۹DML to NWDBM ۲۰ DML to OPSYS
۲۱NWDBM to Optimize ۲۲ NWDBM to User
۲۳LDBM to REQN ۲۴ NWDBM to User

The following statements are network input statements :

Gen, P. Fortier, DBMS Queue SIMPROG, ۵،۲۲،۲۰۰۲،۱;
Umits, ۱۰،۲۰،۵۰۰;
Stat,۱ ,hits on directory,۱۰،۱ .,۱;.
Stat,۲ hits on dictionary, ۱۰،۱ .,۱;.
act,atrib(۵);
free, DMLprocJ۱;
act/۲۰,,atrib(۷).eq.۲۰,opsys service completed
act/۱۹,,atrib(۷).eq. ۱۹,NWDBM;
NWDBM await(۸), NetprocJ۱;
event,۲; Network database manager
act, XX(۲);
free, NetprocJ۱;
act/۶,,atrib(۷).eq.۶.DML processing completed
act/۲۲,,atrib(۷).eq.۲۲,user; data doesn't exist
act/۲۱, ,atrib(۷).eq.۲۱;
optim await(۹), optprocJ۱ ;
event,۳; Query optimization
act,XX(۳);
free, optprocJ۱;
act/۴,,atrib(۷).eq.۴,۰and atrib(۴).eq.۰,WLDBM;
act/۲۶,,atrib(۷).eq.۴ and atrib(۴).eq. ۱ ,LDBM;
act/۵,,atrib(۷).eq.۵, user; illegal query
WLDBM await(۱۰), Iocproc/۱;

Stat,٣,processing time,٢٠،٠.,١٠;.
Stat,٤, remote time, ١٠،٠.,.٠٥;
Stat,٥,failure rate,١٠،١.,١;.
Stat,٦,optimizer time,١٠،٠.,١٠;.
Stat,٧,Optimizer algorithm delay, ١٠،٠., ١٠;.
Stat,٨,parsing delay, ١٠،٠.,.٠٠١٥;
Stat,٩,illegal operations, ١٠،١., ١;.
Stat,١٠, translate delay, ١٠،٠.,.٠١;
Stat, ١١ ,dictionary search, ١٠،٠.,.٠٠٠٠٢;
Network
Resource/Opsproc(١),٦;
Resource/DMLproc(١),٧;
Resource/Netproc(١),٨;
Resource/Optproc(١),٩;
Resource/Locproc(١),١٠ ;
Tape queue(٤);
act/١٣,,,RLDBM;
Mem queue(٢);
act/١١ ,XX(٥),,RLDBM;
LDBM event,٥; local database manager
act/٧,XX(٤),atrib(٧).eq.٧ or atrib(٧).eq.٧٨, rnem;
act/٨,XX(٤),atrib(٧).eq.٨ or atrib(٧).eq.٧٨, disk;
act/٩،٩٩٩٩٩٩,atrib(٧).eq.١٠,DLOC;
act/٢٣,,atrib(٧).eq.٢٣,REQN;

```
Disk queue(۳);
act/۱۲,XX(۶),,RLDBM ;
RLDBM GOON; Request
LDBM
act/۱۴,,,LDBM ;
DLOC Free, Iocproc/۱;
act;
goon;
act/۱۵,,atrib(۸).eq.۱۵,
NWDBM ret route, local
source
act/۱۶،۰,۰۲,atrib(۸).eq.۱۶; ret
route, remote source
reqn goon;
act/۱۷،۰,۰۲;
NWDEL queue(۵); Network
delay
act/۱۸،۰,۰۳,,NWDBM
user colct,int(۱), tim in sys,
۴۰،۰۰,۱۰;.
act/۲۰;
terminate;
endnetwork;
init,۰;
fin;
```

شکل (۸-۲۰): کد مدل شبکه Slam II

۸-۶ خلاصه

این فصل استفاده از شبیه سازی را در ساخت و تحلیل طیف وسیعی از سیستم‌ها معرفی کرد. شبیه سازی‌ها ازنظر توانایی برای سیستم‌ها مدل در سطوح متغیر جزییات بسیار مواریانس هستند. آن‌ها مدل‌های سریع و دقیقی از سیستم‌ها را ارائه دادند که به هر مطالعه‌ای اجازه اجرای آن‌ها داده می‌شود. تکنیک شبیه سازی اصلی رخداد گسسته، پیوسته، صف بندی، ترکیبی، و روش‌های ترکیبی تشریح شده است، که به‌صورت گسترده از زبان‌های GASP, GPSS, Simscript, and Slam استفاده می‌کند. این توسط دو مثال ساده دنبال شد که اینکه چگونه شبیه سازی می‌تواند برای مطالعه سیستم جهان واقعی استفاده شود را نشان می‌دهد.

فصل نهم

شبکه‌های پتری

۹-۱ مقدمه

هر ابزار استفاده شده برای مدل‌سازی و تحلیل سیستم کامپیوتری جایگاه خود را دارد. شبکه‌های پتری در ارزیابی عملکرد سیستم‌های کامپیوتری جای دارند، که جایی بین نظریه صف بندی تحلیلی و شبیه سازی کامپیوتر دامنه آن است. با توجه به ماهیت شبکه‌های پتری و توانایی آن‌ها برای مدل کردن همزمانی، هماهنگی، انحصار متقابل، تصادفات و حالات سیستم است که از مدل‌های تحلیلی کامل تر است اما به اندازه شبیه سازی کامل نیست. آن‌ها نظریه اصلی خود را دارند که به تحلیل اختصاص داده‌شده است، اما آن‌ها در مواردی که به مدل‌ساز اجازه داده می‌شود که موجودیت‌های تکی را درون سیستم، و حرکت آن‌ها و تأثیر بر حالت کل سیستم را شبیه‌سازی کنند، بیشتر شبیه‌سازی هستند.

شبکه‌های پتری (PN) یک ابزار گرافیکی و این روش نوشتاری برای تعیین مشخصات رسمی سیستم‌ها ارائه می‌دهد. سیستم‌ها روندی را مدل می‌کنند که شامل چیزی بیشتر از یک نرخ ورود و نرخ سرویس‌دهی ساده است. آن‌ها در شرایطی استفاده می‌شوند که هر موجودیت که از سیستم گذر می‌کنند می‌توانند اطلاعات حالت خود را داشته باشد، که می‌تواند برای مدل کردن دقیق‌تر و کامل‌تر تراکنش‌ها پیچیده مانند مشاجره و همزمانی استفاده شوند.

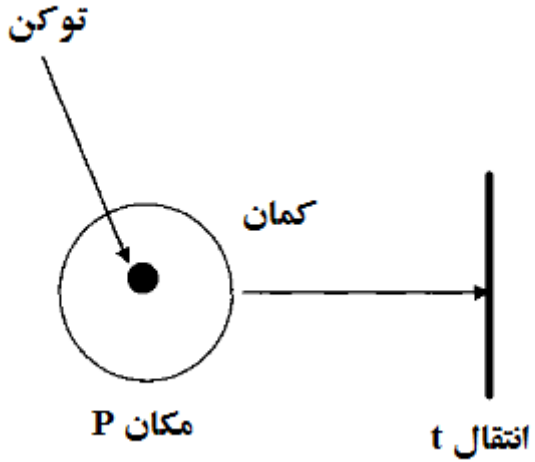
شبکه‌های پتری اولین بار در سال ۱۹۶۶ برای تشریح سیستم‌های همزمانی معرفی شدند. این مقدمه اولیه با بهبود مستمر دنبال شد – برای مثال، افزودن زمان بندی برای انتقال، اولویت انتقال، انواع نشانه[۷۱] ها،

[۷۱] token

شرایط نشان دادن رنگ در محل[۷۲] و نشانه ها. این موارد با رشد ابزارهای نرم افزاری برای کمک به مدل‌سازی و تحلیل سیستم‌های استفاده شده در مفهوم شبکه‌های پتری دنبال شده اند.

۹-۲ نماد پایه

شبکه‌های پتری به سیستم‌های کامپیوتری ابزاری برای جذب عناصر پایه سیستم و جریان اطلاعاتی آن با استفاده از فقط چهار مؤلفه اصلی ارائه می‌دهد.

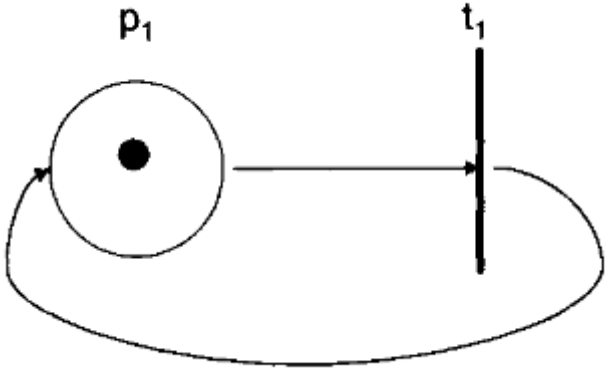


شکل (۹-۱): مؤلفه شبکه پتری پایه

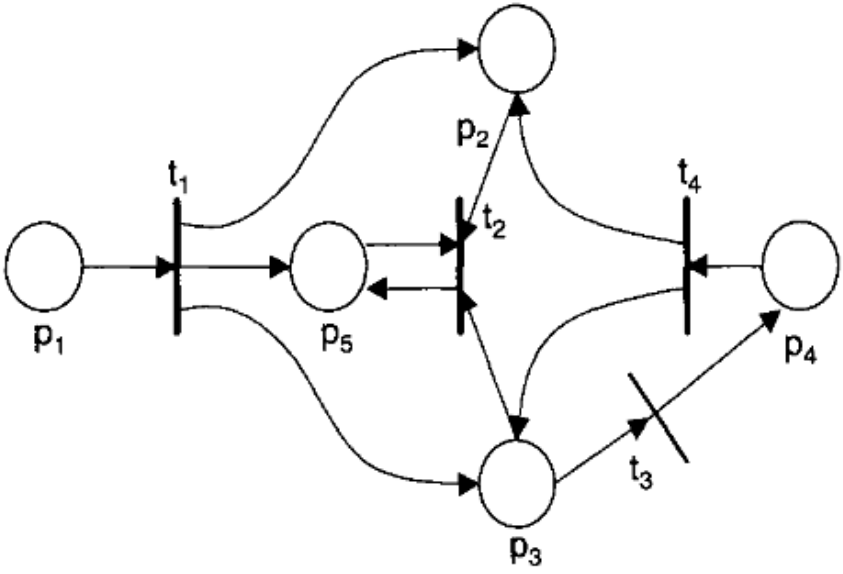
مؤلفه‌های مدل‌سازی شبکه پتری مکان[۷۳]، گذرگاه[۷۴]، کمان[۷۵]، نشانه[۷۶] هستند. مکان‌ها به‌صورت گرافیکی با دایره نشان داده میشوند، گذرگاه با میله نشان داده میشود، کمان یک قطعه خط مستقیم

[۷۲] palce
[۷۳] palce
[۷۴] transition
[۷۵] arc
[۷۶] token

هستند، و نشانه‌ها یا توکن‌ها با نقطه نشان داده می‌شوند (شکل ۹-۱). مکان‌ها برای نمایش مؤلفه‌های محتمل سیستم و حالت آن‌ها استفاده می‌شوند. برای مثال، یک دیسک درایو می‌تواند با استفاده از یک محل نشان داده شود، و برنامه یا دیگر منابع نیز می‌توانند نشان داده شوند. انتقال برای تشریح رخدادهایی استفاده می‌شود که در حالات سیستمی متفاوت نتیجه می‌دهد. گذارها یا انتقال‌ها برای تشریح رخدادهایی استفاده می‌شوند که ممکن است در حالات متفاوت سیستم نتیجه دهند. برای مثال، عمل خواندن یک آیتم از دیسک درایو یا عمل نوشتن یک آیتم بر دیسک درایو می‌تواند به‌عنوان انتقال‌های جدا مدل شوند. کمان‌ها روابطی که بین انتقال‌ها و محل‌ها وجود دارند را نشان می‌دهند. برای مثال، درخواست خواندن از دیسک در یک محل گذاشته می‌شود، و آن محل ممکن است با گذرگاه، "برداشتن آیتم از یک دیسک" در ارتباط باشد، بنابراین نشان می‌دهد که این محل به گذرگاه مربوط است. می‌توانید فکر کنید که کمان مسیری را برای فعال سازی گذرگاه ارائه می‌دهد. سرانجام، توکن‌ها برای تعریف حالت شبکه پتری استفاده می‌شوند. توکن‌ها در مدل شبکه پتری پایه نشانه گذارهایی غیرتوصیفی هستند، در محل‌ها ذخیره می‌شوند و در تعریف نشانه گذاری شبکه پتری استفاده می‌شوند. نشانه گذاری محل شبکه پتری با جایگذاری یک توکن می‌تواند به‌عنوان عبارت شرطی محل دیده شود. برای مثال، شکل (۹-۲) یک شبکه پتری ساده را با یک محل و یک گذرگاه نشان می‌دهد. محل به گذرگاه با یک کمان مرتبط شده است. کمان اول یک کمان ورودی است، در حالی که کمان دوم یک کمان خروجی است. مکان یک نشانه گذاری فعال حالت شبکه پتری را نشان می‌دهد. شبکه پتری نشان داده‌شده در شکل (۹-۲) یک شبکه را نشان می‌دهد که یک چرخه را تا همیشه ادامه می‌دهد.



شکل (۹-۲): نمونه‌ای از حرکت ادراکی شبکه پتری



شکل (۹-۳): نمونه شبکه پتری

شبکه‌های پتری به‌صورت گرافیکی تشریح شده‌اند و از مجموعه نمادها استفاده کرده‌اند. همان‌طور که از قبل گفته شد، یک شبکه پتری از مکان‌ها (P)، گذرگاه‌ها (T)، کمان‌ها (شامل تابع ورودی، I، و تابع خروجی، O)، نشانه ها، تشکیل شده است که نشانه گذاری شبکه را (MP) تشکیل می‌دهند. با استفاده از این مفهوم، می‌توانیم یک شبکه پتری را به‌عنوان یک پنج‌تایی $M = (P, T, I, O, MP)$ تعریف کنیم، که در آن P یک مجموعه از مکان‌ها، $P = \{p_1, p_2, \dots, p_n\}$ است، با یک مکان برای هر دایره در گراف شبکه پتری، T یک مجموعه از گذرگاه‌ها است، $T = \{t_1, t_2, \dots, t_m\}$ برای هر میله در گراف شبکه پتری، I یک بسته از مجموعه توابع ورودی برای همه گذرگاه‌ها است $I = \{I_{t1}, I_{t2}, \dots, I_{tm}\}$، که مکان را به گذرگاه نگاشت می‌کنند، O یک مجموعه از مجموعه توابع خروجی برای همه گذرگاه‌ها است $O = \{O_{t1}, O_{t2}, \dots, O_{tm}\}$، نگاشت انتقال به مکان، و MP نشانه گذاری مکان با توکن را نشان می‌دهد. نشانه گذاری اولیه با MP_0 نشان داده می‌شود. MP_0 یک چندتایی مرتب از مقدار n را نشان می‌دهد،که در آن n تعداد مکان‌ها در شبکه پتری ما است. هر مکان یا هیچ توکنی ندارد یا چند تا دارد. برای مثال، گراف شبکه پتری نشان داده‌شده در شکل (۹-۳) می‌تواند با استفاده از مفاهیم قبلی تشریح شود:

$$M = \{P, T, I, O, MP\}$$

$$P = \{P_1, P_2, P_3, P_4, P_5\}$$

$$T = \{t_1, t_2, t_3, t_4\}$$

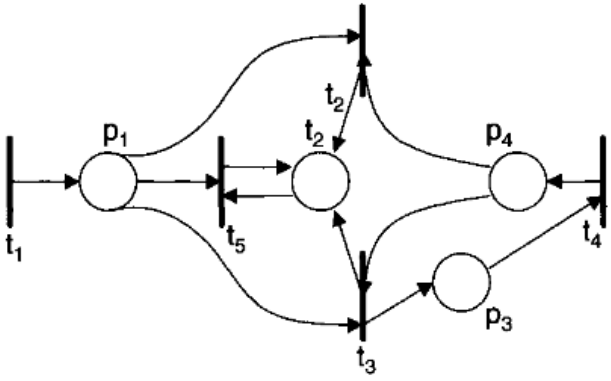
$$I(t_1) = \{P_1\}$$

$$I(t_2) = \{p_2, p_3, p_5\} \quad (9-1)$$

$$I(t_3) = \{P_3\}$$

$I\ (t_۴) = \{P_۴\}$

$O\ (t_۱) = \{P_۲\ , P_۳, P_۴\}$



شکل (۹-۴): دوگان شبکه پتری از شکل (۹-۳)

$O\ (t_۲) = \{P_۵\}$

$O\ (t_۳) = \{P_۴\}$

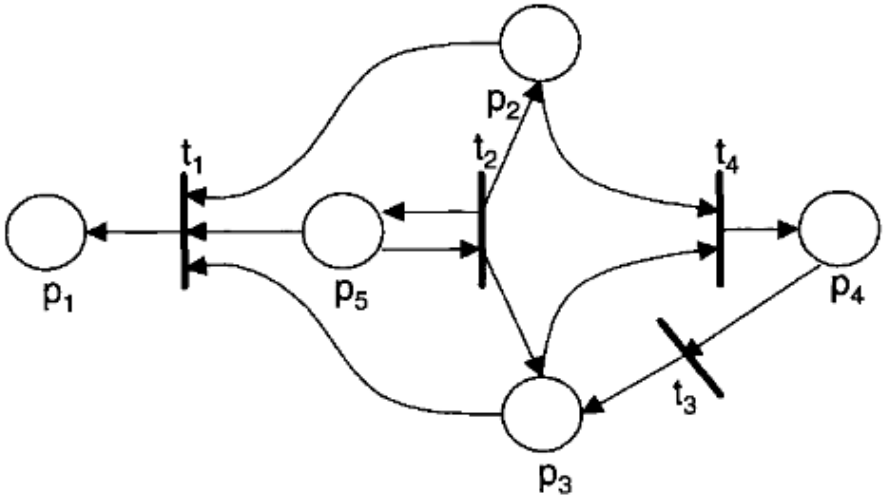
$O\ (t_۴) = \{P_۲, P_۳\}$ ادامه (۹-۱)

$MP = (۰, ۰, ۰, ۰, ۰)$

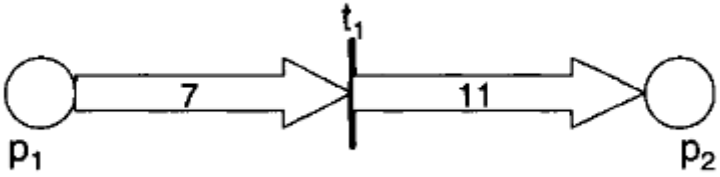
نمادهای گرافیکی شبکه‌های پتری را به‌عنوان یک گراف دو بخشی جهت‌دار نشان می‌دهند. گراف، G، با یک دوتایی، $\mathrm{G} = (\mathrm{V}, \mathrm{A})$ تشریح شده است، V یک مجموعه از رئوس را نشان می‌دهد، $V = \{v_۱, v_۲, v_۳, \dots, v_s\}$ ، و A یک مجموعه از کمان‌های جهت دار را نشان میدهد $A = \{a_۱, a_۲, a_۳, \dots, a_i\}$. یک کمان، که یک عنصر از A است، از یک چندتایی با دو راس، $a_i = (v_j, v_k)$ تشکیل شده است، که در آن $v_j, v_k \in V$ برقرار است. مجموعه رئوس گراف می‌توانند

به دو مجموعه منفصل P و T بخش‌بندی شوند، که در آن این مجموعه‌ها ویژگی $V = P \cup T$ و $P \cap T = ۰$ را دارد. علاوه بر این، اگر برای یک کمان $a_i \in A$، اگر $a_i = (v_j, v_k)$ برقرار باشد، سپس $v_j \in P$ و $v_k \in T$ یا برعکس برقرار است. به همین ترتیب، دو انتهای کمان نمی‌توانند از یک مجموعه استنتاج شوند، اگر $v_j \in P$ برقرار باشد، سپس $v_k \in T$ برقرار است، و نمی‌تواند عنصر P باشد.

یک مدل شبکه پتری، با بسیاری از مدل‌های ریاضی، یک دوگان دارد. دوگان شبکه پتری به‌عنوان شبکه پتری تعریف می‌شود که در آن گذرگاه‌ها به مکان و مکان‌ها به گذرگاه تغییر می‌کنند. (شکل ۹-۴) توابع ورودی و خروجی به ورودی‌هایی برای گذرگاه‌ها تغییر می‌کند حال ورودی‌هایی برای مکان‌ها نشان داده می‌شود ازآنجایی‌که این کار ممکن نیست، ورودی‌ها به توابع خروجی تبدیل می‌شوند و توابع خروجی به توابع ورودی تبدیل می‌شوند.



شکل (۹-۵): معکوس شبکه پتری از شکل (۹-۳)



شکل (۹-۶): کمان چند مسیره

یک شبکه پتری نیز دارای یک معکوس تعریف‌شده است. معکوس یک شبکه پتری همه مکان‌ها. گذرگاه‌ها را یکسان نگه می‌دارد و توابع ورودی را با توابع خروجی جابجا می‌کند. (شکل ۹-۵).
شبکه‌های پتری تعریف‌شده چند گرافی هستند، چرا که یک مکان می‌تواند ورودی‌های متعدد و یا خروجی‌هایی را از یا به گذرگاه نشان دهد. ما می‌توانیم این‌ها را به‌عنوان کمان‌های منحصر مدل کنیم اما شماره دهی آن‌ها با رشد کمان‌ها پیچیده می‌شود. یک روش بهتر برای مدل کردن کمان‌های متعدد نمایش کمان‌های متعدد به‌عنوان یک کمان نازک با عدد نشان‌دهنده کمان در کنار آن است (شکل ۹-۶)، یا کمان پررنگ با یک عدد ضمیمه‌شده به‌عنوان یک برچسب (شکل ۹-۷).
شبکه‌های پتری یک حالت دارند. این حالت با کاردینالیتی توکن‌ها و توزیع آن‌ها از طریق مکان‌ها در شبکه پتری تعریف می‌شود. این نشانه گذاری به عنوان یک تابع، μ، در هر مکان P تعریف می‌شود و در مقدار N از مجموعه شمارش اعداد صحیح ۰,۱, ... , ∞ نتیجه می‌شود.

$$\mu: P \to N \qquad (۹-۲)$$

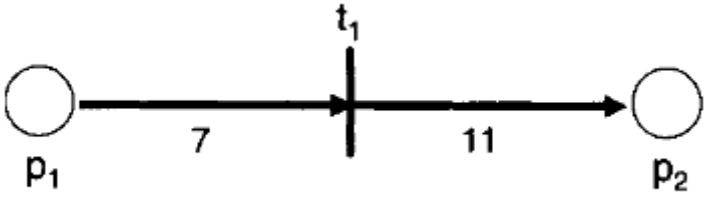
نشانه گذاری، μ به‌عنوان بردار n تایی تعریف شود. بردار μ اطلاعات نشانه را برای هر مکان p_i، در شبکه پتری فراهم میکند. اطلاعات توکن تعداد توکن را در محل خاصی نشان می‌دهند (تعداد توکن در محل p_i، μ_i است).

$$\mu = (\mu_۱ , \mu_۲ , \mu_۳ , \dots , \mu_n)$$

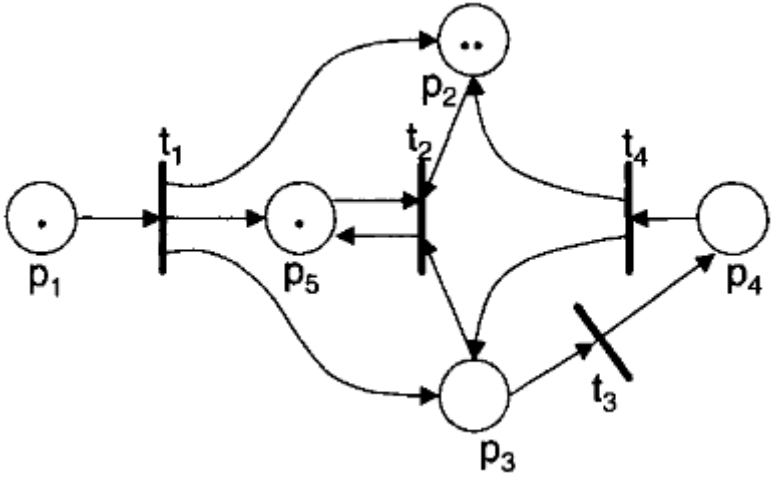
جایی که

(۹-۳)

$$n = |P| \; and \; each \; \mu_i \in N, \qquad i = ۰, .., n$$



شکل (۹-۷): کمان‌های چندمسیره با خط پررنگ



شکل (۹-۸): شبکه پتری توکن دار

نشانه گذاری مکان به‌عنوان یک تابع نشان داده می‌شود و نشانه‌گذاری‌های مکان به‌عنوان یک بردار مربوطه با $\mu(P_i) = \mu_i$ نشان داده می‌شود. نشانه‌گذاری‌ها در یک نقطه زمانی خاص حالت شبکه پتری را در زمان نشان می‌دهند. یک شبکه پتری نشانه گذاری شده $M = (C, \mu)$ ، به‌عنوان ساختار شبکه پتری نشان داده می‌شود، $M = (P, T, I, O)$ و نشانه گذاری آن، μ یا MP معمولاً با $M = (P, T, I, O, \mu)$ نشان داده می‌شود. نشانه گذاری μ با تغییر حالت شبکه پتری تغییر می‌کند و معمولاً با زیرنویس t نشان داده می‌شود. بنابراین، نمایش درست به شکل زیر است:

$$M = (P, T, I, O, \mu_t) \quad (۹-۴)$$

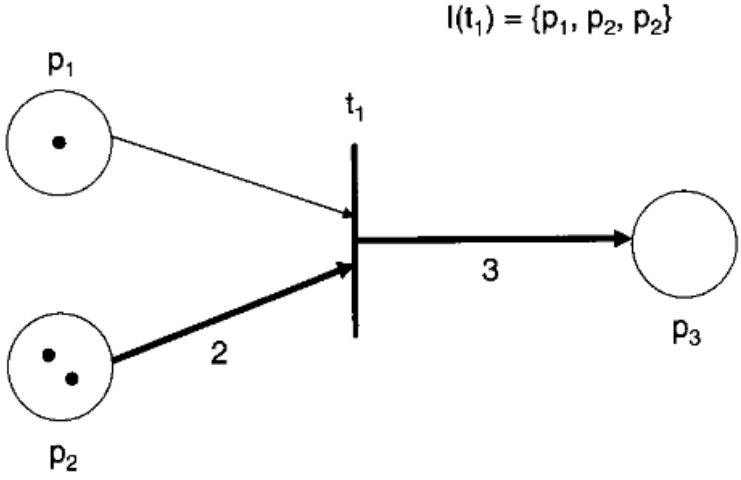
در اینجا حالت شبکه پتری در زمان t نشان داده‌شده است، که در آن t از مجموع مقادیر اعداد صحیح غیر منفی استنتاج شده است.

نشانه گذاری یک شبکه پتری با جایگذاری توکن‌ها مشخص‌شده است، که با نقطه‌هایی در نمادهای گرافیکی در مکان‌ها (شکل ۹-۸) نشان داده‌شده است. نشانه‌گذاری شبکه پتری در شکل (۹-۸) به‌عنوان یک بردار $\mu_t = (۱,۲,۰,۰,۱)$ نشان داده‌شده است. اگر فرض کنیم که این نشانه گذاری اولیه این شبکه پتری است، سپس تعریف به $\mu_. = (۱,۲,۰,۰,۱)$ تبدیل می‌شود، چرا که این صفرمین حالت شبکه پتری است که بازدید شده است. تعداد نشانه‌های تخصیص داده‌شده به یک محل نامحدود است (اگر مدل اصلاح شده دوم این را محدود می‌بیند). مجموع همه نشانه‌گذاری‌های ممکن برای شبکه پتری با n محل به‌سادگی مجموعه‌ای از n بردار است، N^n، N همه حالات ممکن است و n تعداد مکان‌ها است.

۹-۳ شبکه‌های پتری کلاسیک

با توجه به مفاهیم و تعاریف پایه از بخش قبلی، می‌توانیم بررسی اینکه چگونه این عناصر اصلی می‌توانند در مدل‌سازی، جنبه‌هایی از سیستم کامپیوتری و کل سیستم نهایی استفاده شود را بررسی کند. ابتدا مفاهیم جدید موردنیاز از گذرگاه‌های حالت شبکه پتری است. برای حرکت از یک حالت به حالت

دیگر، یک شبکه پتری همه گذرگاه‌هایی که فعال هستند را به کار می‌گیرد.لحظه دقیق شلیک می‌تواند با وقوع سیگنال کلاک در یک سیستم کامپیوتری تصور شود. زمانی که کلاک یک سیکل را شروع می‌کند همه گیت‌هایی که دارای سیگنال‌های توانمند ساز اجرا هستند در طول سیکل فعال می‌شوند. به طور مشابه، در یک شبکه پتری همه گذرگاه‌ها فعال هستند که در طول سیکل مربوطه به کار می‌افتند.

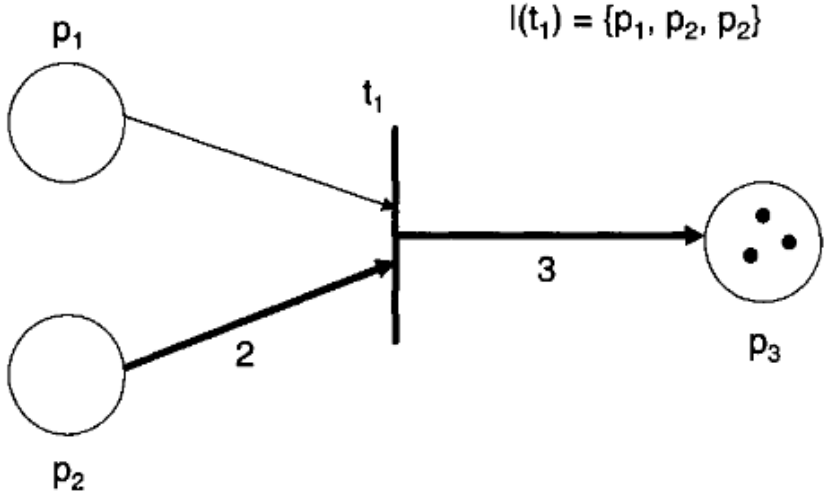


شکل (۹-۹): گذرگاه فعال

قبل بررسی شلیک، ما نیاز داریم که شرایط موردنیاز را بررسی کنیم. داشتن نشانه‌هایی در مکان، مبنایی برای مفاهیم توانمندی گذرگاه است. بنابراین، مهم است حالت کل شبکه پتری را قبل از آماده‌سازی برای گذرگاه‌های آماده شلیک بدانیم. فعال کردن گذرگاه‌ها باعث در دسترس ساختن نشانه‌ها در مکانی می‌شود که عضوی از تابع ورودی گذرگاه است. تنها اگر همه مکان‌ها نامگذاری شده باشند، در یک تابع ورودی گذرگاه نشانه‌های در دسترس قرار گرفته یک گذرگاه فعال هستند (شکل ۹-۹).

شبکه پتری در شکل (۹-۹) با $\mu_{\cdot} = (۱,۲,۰)$، تابع ورودی $I(t_۱) = \{P_۱, P_۲, P_۳\}$، و تابع خروجی $O(t_۱) = \{P_۳, P_۳, P_۳\}$ نشانه گذاری شده است. با توجه به اینکه نشانه‌گذاری‌های اولیه ورودی تعریف‌شده و توابع خروجی، گذرگاه $t_۱$ فعال است، از آنجا که این یک نشانه نیاز دارد که در $P_۱$ سهم داشته باشد و دو نشانه که در $P_۲$ سهم داشته باشد همه توابع ورودی آن با توکن‌های موجود در مکان‌های

نام‌گذاری شده و مقدارهای موردنیاز در دسترس هستند. توکن‌ها در این مکان‌ها به‌عنوان توکن‌های توانمند ساز گذرگاه‌ها مورداشاره قرار گرفته اند. با توجه به اینکه یک گذرگاه در آغاز چرخه شلیک شبکه پتری فعال شده است، این گذرگاه را فعال می‌کند، باعث حرکت تعداد نشانه از مکان ورودی آن به مکان خروجی می‌شود، که توسط آرایه‌ای از توابع خروجی مدل شده است. نتایج این شلیک یک حالت شبکه پتری جدید، μ_1 است. برای مثال، در شکل (۹-۹)، با توجه به حالت اولیه μ_0، گذرگاه P_1 در آغاز چرخه شلیک فعال می‌شود، یک توکن را از مکان P_1 برمی‌دارد و دو توکن را از P_2 برمی‌دارد و این سه توکن را در P_3 قرار می‌دهد. حالت شبکه پتری جدید با نشانه گذاری $\mu_1 = (0,0,3)$، در شکل (۹-۱۰) نشان داده است. عمل شلیک به‌عنوان یک عمل اتمیک در نظر گرفته می‌شود، که در آن به نظر می‌رسد که آیا همه نشانه‌ها از مکان‌های ورودی حذف شده‌اند و به مکان‌های خروجی سپرده‌شده‌اند یا نه.



شکل (۹-۱۰): حالت شبکه پتری جدید

شلیک شبکه پتری حرکت‌هایی را از یک حالت به حالت جدید فراهم می‌کند. حالت جدید تنها حالت قابل دسترس در یک شلیک مجموع گذرگاه‌ها از همه گذرگاه‌های فعال است. مجموع همه حالات ممکن می‌تواند با شبکه پتری نشان داده شود و نشانه گذاری اولیه فضای حالت گفته می‌شود. مجموعه

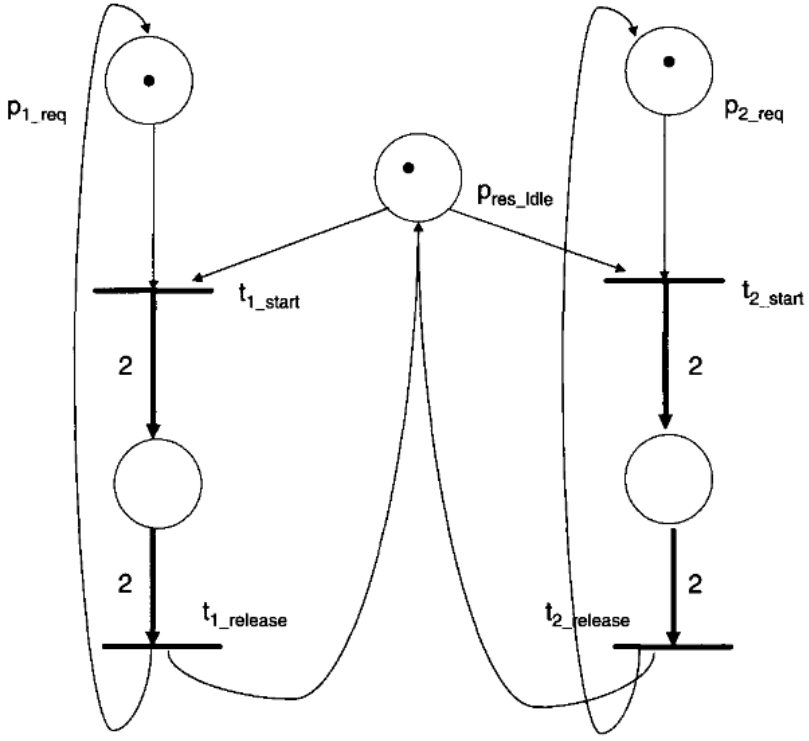
همه حالات در فضای حالت این شبکه پتری می‌تواند در دنباله حالت اولیه به حالت انتهایی به‌صورت یک مرحله یک مرحله، با تغییرات فضای حالت برسد. این حاکی از این است که یک شبکه پتری می‌تواند تنها گذرگاه‌های فعال شده را طی کند، و بعد از شلیک، آن‌ها نمی‌توانند هر گذرگاه به تازگی فعال شده را تا سیکل بعدی شلیک کنند. این حالت منحصربه‌فرد تغییر حالت شبکه پتری می‌تواند با استفاده از یک تابع حالت بعدی، δ تشریح می‌شود. این تابع، δ، در زمان استفاده برای یک حالت شبکه پتری، μ_i، باعث می‌شود که شبکه پتری از حالت μ_i به حالت جدید μ_{i+1} منتقل شود. تابع δ به شکل زیر تعریف می‌شود:

$$\delta\left(\mu_i\{t\}\right) = \mu_{i+1} \quad (۹-۵)$$

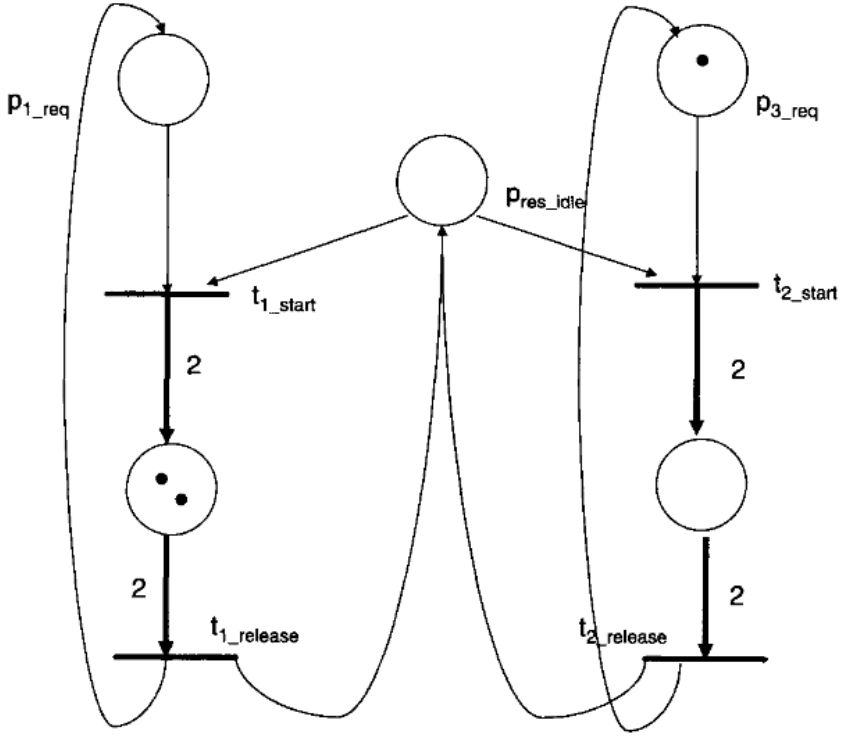
مجموعه $\{t\}$ مجموعه همه گذرگاه‌های فعال شده درون شبکه پتری است. اگر یک گذرگاه فعال نشده باشد، سپس این تابع تعریف نشده است.

فراتر از مبناهای تعریف‌شده در اینجا، شبکه‌های پتری برای مدل‌سازی شرایطی که معمولاً در نظریه صف بندی موجود نیستند توسعه داده‌شده اند. برای مثال، همگام‌سازی، تضادها، همزمانی مفاهیمی هستند که به‌سادگی توسط نظریه صف تعریف‌شده و مدل شده اند.

برای شبیه سازی برخی از مفهوم‌های پوشش داده‌شده ما به یک مثال ساده نگاه می‌کنیم. در این مثال، نشان داده‌شده در شکل (۹-۱۱)، ما و فرایند کاربر را با درخواست سورس خاص (P_{res_idle}) مدل میکنیم. اگر منبع idle باشد، یک نشانه در مکان P_{res_idle} وجود دارد. اگر یک فرآیند بخواهد که این منابع یک توکن در مکان $(e.g., P_{1\_req})$ داشته باشد، و منبع idle باشد (یک توکن در P_{res_idle})، سپس گذرگاه $(e.g., t_{1\_start})$ فعال می‌شود و می‌تواند در سیکل بعدی شلیک کند. زمانی که گذرگاه شلیک کند، منابع از دسترس خارج می‌شوند (مشغول می‌شوند)، و فرآیند دوم باید تا زمانی که منابع آزاد شوند، منتظر بماند (شکل ۹-۱۲).



شکل (۱۱-۹): مثال اشتراک گذاری منابع

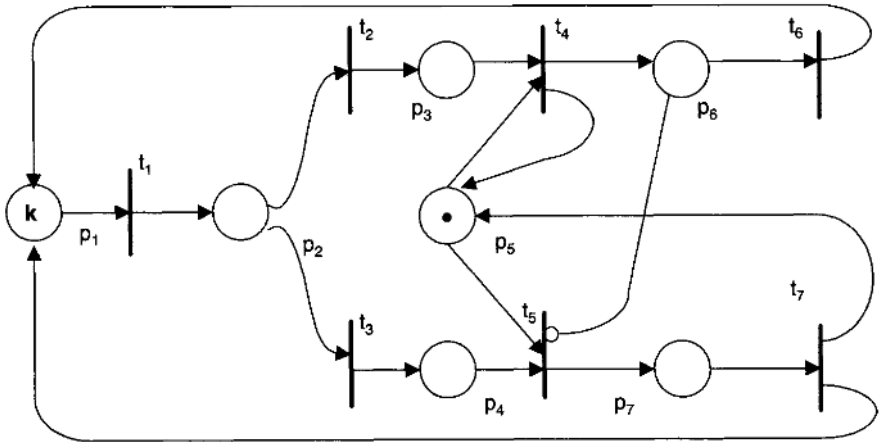


شکل (۹-۱۲): منابع تخصیص داده‌شده

با این مفاهیم پایه می‌توانیم به سمت موارد پیشرفته تر برویم. اغلب اوقات در یک سیستم کامپیوتری، زمانی که یک فرآیند به برخی از منابع دسترسی می یابد یا برای آن تلاش می کند، دیگر فرآیند از تلاش برای ورود منع می‌شوند. این مفهوم یک گیت، قفل، یا سمافور است. در مورد این آیتمی، زمانی که یک کار بر روی منابع دیگر کنترل دارد، دیگران از تلاش برای دسترسی به منابع منع می‌شوند، حتی اگر آن‌ها همه منابع دیگر را داشته باشند و نیاز داشته باشند که اجرای خود را ادامه دهند. برای مدل کردن این مفهوم قفل یا سمافور، مدل‌سازی شبکه پتری مفهوم مهارکننده را شبیه سازی کرده است. یک مهار کننده یک تابع است که یک مکان (به‌عنوان مسدود کننده) را به یک گذرگاه ربط می‌دهد. اگر یک مکان، p، دارای یک رابطه مسدود کننده با یک گذرگاه،t، باشد، سپس زمانی که مکان P دارای توکن باشد، گذرگاه t نمی‌تواند شلیک کند، حتی اگر همه توابع ورودی برآورده شده باشند و گذرگاه‌ها در

غیر این صورت فعال می‌شوند. برای مثال، شکل (۹-۱۳) مسئله خواننده و نویسنده را نشان می‌دهد. مسئله بیان می‌کند که یک نویسنده در حالت نوشتن هست، همه خوانندگان باید از دسترسی به منبع مربوطه منع شوند. در مثال، مهار کننده به‌عنوان یک کمان بدون جهت با یک دایره کوچک در گذرگاه مهار شده با مکانی در دیگر انتهای کمان نشان داده می‌شود. مثال نشان می‌دهد که مکان، $p_۶$، به‌عنوان مهارکننده‌ای برای گذرگاه $t_۵$ است. شبکه پتری حال با استفاده از شش تایی، $M = (P, T, I, O, N, H)$ نشان داده‌شده است. در مجموعه مفاهیم مهار کننده به‌صورت $H\{t_۵\} = \{p_۶\}$ تشریح شده است.

یک حالت شبکه پتری، μ ، از حالت دیگر، μ' قابل دسترسی است، اگر برخی از تعداد محدود شلیک شبکه پتری در حالت μ آغاز شوند، که در نشانه گذاری نهایی μ' نتیجه می‌دهد. یک قابلیت دسترسی $[RS\ (\mu)]$ از هر نشانه گذاری شبکه پتری ما تنظیم می‌شود، M در حالت N شروع می‌شود، که به‌عنوان مجموعه‌ای از همه نشانه گذاری‌های ممکن قابل دسترسی از طریق هر مجموعه شلیک تعریف می‌شود. هیچ قابلیت دسترسی برای یک شبکه پتری با یک نشانه گذاری null اولیه به حالت امکان‌پذیر تنظیم نشده است.



شکل (۹-۱۳): شبکه پتری

مجموعه قابلیت دسترسی برای یک شبکه پتری با یک نشانه گذاری اولیه μ_0، با $RS(\mu_0)$ اشاره می‌شود، و به‌عنوان کوچک‌ترین مجموعه نشانه گذاری است، به‌طوری‌که روابط زیر برقرار است:

$$\mu_0 \in RS(\mu_0) \; and \; \mu_1 \in RS(\mu_0)$$

$$\exists\, r \in T: \delta\left(N,(t)\right) \rightarrow \mu_2(t+1) \qquad (۶-۹)$$

$$\rightarrow \mu_2 \in Rs(N_0)$$

برای تعیین تنظیمات قابلیت دسترسی، باید از حالت اولیه، μ_0، آغاز کنیم، و به‌صورت افزایشی، هر مرحله ممکن ناشی شده از حالت اولیه و همه حالت قابل استنتاج از این حالت را تعریف کنیم. زمانی که یک نشانه گذاری در نظر گرفته شود، در طول هر تکرار، نمی‌تواند بار دیگر در نظر گرفته شود. تنظیم قابلیت دسترسی برای گراف خواننده و نویسنده در شکل (۹-۱۳)، برای $k = ۲$ نشان داده‌شده است.

$$\mu_0 = ۲P_1 + P_5$$

$$\mu_1 = P_1 + P_2 + P_5$$

$$\mu_2 = ۲P_2 + P_5$$

$$\mu_3 = P_1 + P_3 + P_5$$

$$\mu_4 = P_1 + P_4 + P_5$$

$$\mu_5 = P_2 + P_3 + P_5$$

$$\mu_6 = P_2 + P_4 + P_5$$

$$\mu_7 = P_1 + P_5 + P_6$$

$$\mu_{۸} = P_{۱} + P_{۷}$$

$$\mu_{۹} = ۲P_{۳} + P_{۵}$$

$$\mu_{۱۰} = P_{۳} + P_{۴} + P_{۵}$$

$$\mu_{۱۱} = P_{۲} + P_{۵} + P_{۶}$$

$$\mu_{۱۲} = ۲P_{۴} + P_{۵}$$

$$\mu_{۱۳} = P_{۲} + P_{۷}$$

$$\mu_{۱۴} = P_{۳} + P_{۵} + P_{۶}$$

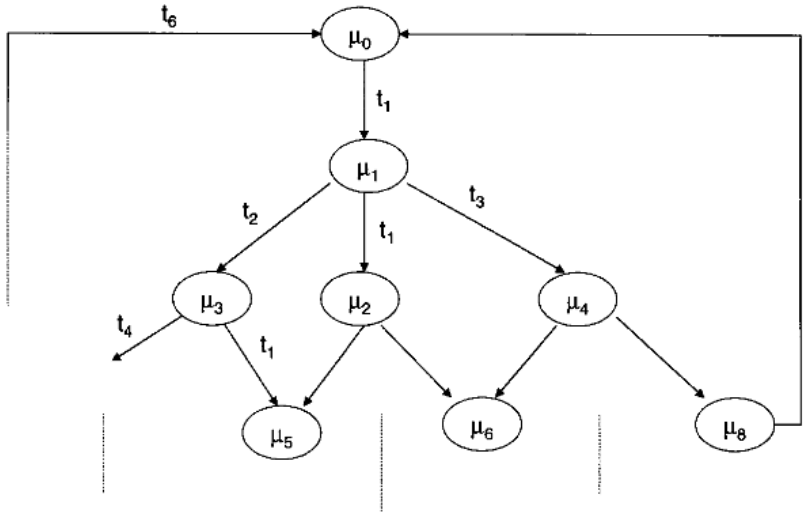
$$\mu_{۱۵} = P_{۴} + P_{۵} + P_{۶}$$

$$\mu_{۱۶} = P_{۳} + P_{۷}$$

$$\mu_{۱۷} = P_{۴} + P_{۷}$$

$$\mu_{۱۸} = P_{۵} + ۲P_{۶}$$

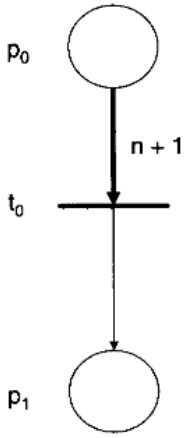
تنظیم قابلیت دسترسی شامل هیچ اطلاعاتی در مورد اینکه کدام گذرگاه‌ها برای رسیدن به نشانه‌گذاری‌های حالت شلیک میشوند، است. این اطلاعاتی می‌تواند در یک گراف قابلیت دسترسی نشان داده‌شده در شکل (۹-۱۴)، دیده شود. در این گراف هر گره یک حالت پتری را نشان می‌دهد و هر کمان یک گذرگاه مستقیم را نشان می‌دهد، که از یک انتهای کمان جهت‌دار به انتهای دیگر با توجه به یک فعال سازی گذرگاه واحد امکان‌پذیر است. برای مثال، شما می‌توانید ببینید که اگر ما گذرگاه $t_{۱}$ را از $\mu_{۰}$ فعال کنیم، می‌توانیم به حالت $\mu_{۱}$ برسیم، که نشانه از مکان ۱ به مکان ۲ می‌رود.



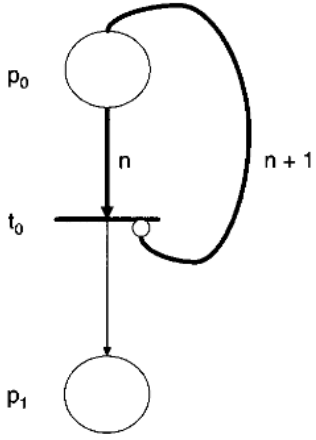
شکل (۹-۱۴): گراف قابلیت دسترسی برای مسئله خواننده و نویسنده

اغلب مدل کردن شرایط منطقی مطلوب است. برای مثال، تنها برای فعال کردن یک گذرگاه زمانی که بیش از n نشانه در یک مکان وجود دارد، ما به‌سادگی نیازمند یک کمان با arity_{n+1} هستیم. از آنجا که شرط پایه بر یک فعال‌سازی گذرگاه این است که مکان‌های ورودی مرتبط آن شرایط تابع ورودی گذرگاه‌ها را برآورده سازند، شامل $n+1$ آیتم مجموعه افزونه برای مکان ورودی است که می‌تواند آنچه نیاز داریم را تأمین کند (شکل ۹-۱۵).

به‌جای تست شرط برابر و کوچک‌تر از مقدار، می‌توانیم از مهارکننده arity_{n+1} برای مسدودسازی یک گذرگاه استفاده کنیم اگر بیش از n نشانه در مکان ۰ وجود داشته باشد.



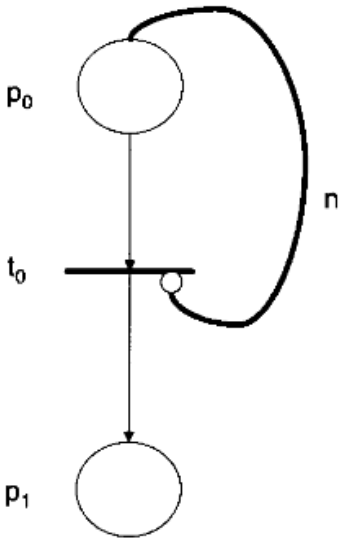
شکل (۹-۱۵): مؤلفه شبکه پتری برای تست شرط بزرگ‌تر از M



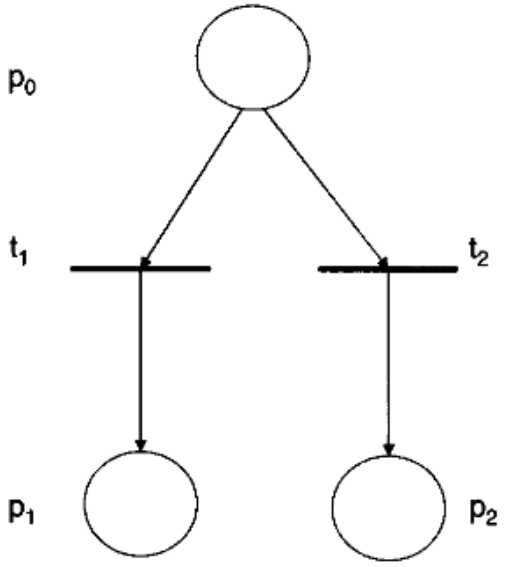
شکل (۹-۱۶): مؤلفه شبکه پتری برای تست شرط برابر اما نه بزرگ‌تر از M

برای برآورده کردن شرط برابری ما از کمان وزن‌دار در n برای حذف n آیتم استفاده می‌کنیم اگر n و فقط n آیتم وجود داشته باشد (شکل ۹-۱۶). اگر بخواهیم که کمتر از n آیتم را تست کنیم و آیتم‌ها را حذف کنیم، می‌توانیم از شبکه پتری شکل (۹-۱۷) استفاده کنیم. بار دیگر، این شبکه پتری از تابع مهارکننده برای مسدودسازی حرکت تعداد مطلوب نشانه استفاده می‌کند.

یک مشخصه مهم موردنیاز در زمان مدل‌سازی سیستم‌های کامپیوتری تضاد و تعارض است (شکل ۹-۱۸). در این مثال، زمانی که یک توکن در محل P_0 باشد، هردو گذرگاه t_1 و t_2 فعال هستند. به‌هرحال، تنها یکی از آن‌ها ممکن است شلیک کند، چرا که تنها یک نشانه در دسترس است. به محضی که اولین شلیک صورت گیرد، مثلاً t_1، این نشانه را از محل P_0 حذف می‌کند و آن را به مکان P_1 منتقل می‌کند. به محضی که گذرگاه t_1 نشانه را برداشت، گذرگاه t_2 دیگر فعال نیست. اگر دو نشانه در محل P_0 وجود داشته باشد، سپس هردو گذرگاه‌ها فعال خواهند بود و می‌تواند در طول این چرخه شلیک، شلیک کنند.



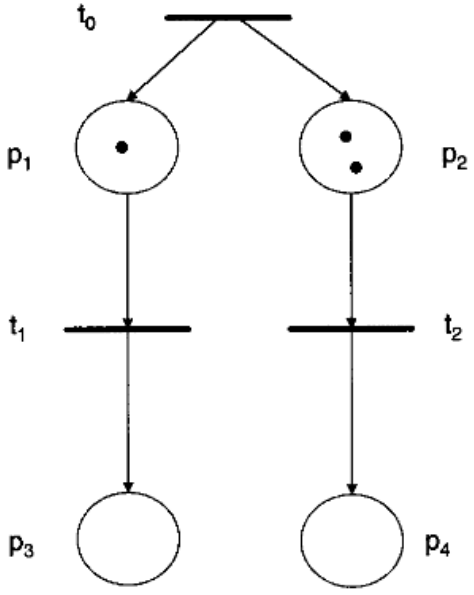
شکل (۹-۱۷): تضاد مدل‌سازی شبکه پتری



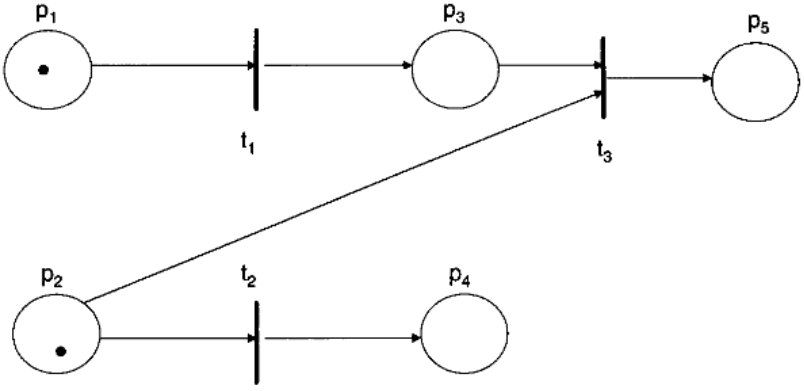
شکل (۹-۱۸): مؤلفه‌های شبکه پتری برای تست شرط کمتر

اغلب می‌توانیم نشان دهیم که کدام‌یک از این گذرگاه تنها در زمانی که یکی می‌تواند شلیک کند، شلیک می‌کند، و به چه ترتیبی هردو آن‌ها قادر به شلیک هستند. ما بر برخی از این کنترل‌های بسط یافته بحث می‌کنیم.

دیگر ویژگی مهم موردنیاز در زمان مدل‌سازی سیستم‌های کامپیوتری و فرآیندهای نرم‌افزاری همزمانی است. همزمانی با اجرای همزمان یا موازی فعالیت‌ها مشخص می‌شود. برای اینکه یک شبکه پتری فعالیت همزمان داشته باشد، نیاز است که گذرگاه‌های فعال‌شده همزمان داشته باشیم، از آنجا که آن‌ها هر دو در یک زمان در یک نشانه گذاری شبکه پتری فعال‌شده هستند، مثال ما، $\mu = (۱,۲,۰,۰)$ دارای گذرگاه‌های فعال $t_۱$ و $t_۲$ است، و آن‌ها می‌توانند همزمان شلیک کنند.



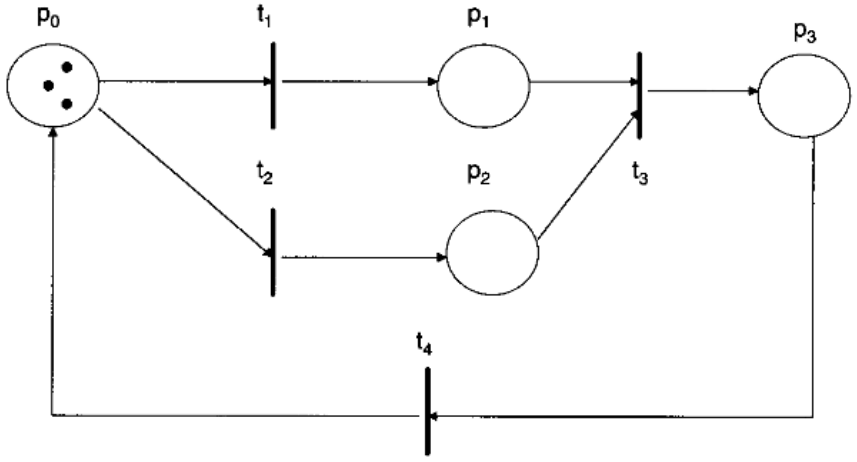
شکل (۹-۱۹): همزمانی شبکه پتری



شکل (۹-۲۰): ابهام مدل‌سازی شبکه پتری

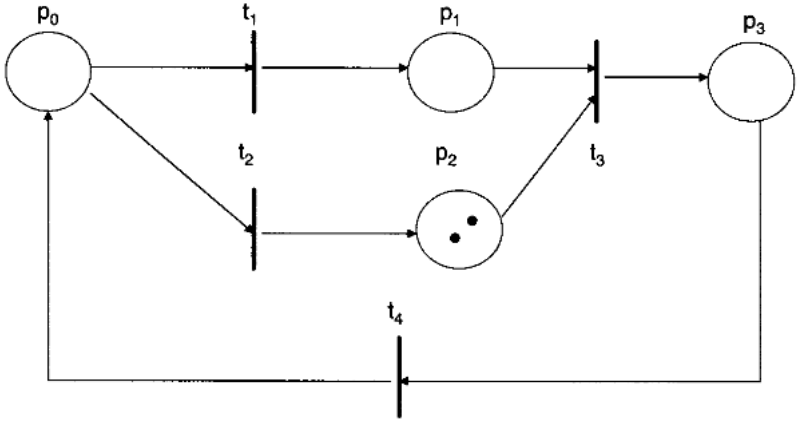
در شکل (۹-۲۰) ما یک نشانه گذاری اولیه داریم، $\mu = (۱,۱,۰,۰,۰)$ را داریم، که در گذرگاه‌های $t_۱$ و $t_۲$ فعال‌شده، در شرایط گذرگاه‌های همزمان نتیجه می‌دهد. اگر ابتدا $t_۱$ شلیک کند، سپس ما دو گذرگاه فعال داریم، $t_۲$ و $t_۳$. این می‌تواند تضادی را نشان دهند، چرا که نشانه از $P_۲$ می‌تواند تنها یکی از شرایط ضروری را برای دو گذرگاه فعال‌شده در این نقطه زمانی را برآورده سازد.

یک شبکه پتری می‌تواند انواع ویژگی‌های دیگر را داشته باشد. برای مثال، یک حالت شبکه پتری، μ ، از حالت دیگر μ'، یک شبکه پتری قابل‌دسترسی است اگر یک عدد صحیح مراحل میانی از μ' وجود داشته باشد که ما را به حالت μ هدایت کند. برای مثال، در شکل (۹-۲۱) حالت اولیه $\mu_۰ = (۳,۰,۰,۰)$، و یک حالت هدف $\mu' = (۱,۱,۰,۱)$ را داریم. از محاسبه گذرگاه‌های حالت می‌بینیم که شبکه ما می‌تواند به حالت هدف خود در شلیک شبکه ما دسترسی داشته باشد. یک ویژگی مربوطه برگشت‌پذیری است. برگشت‌پذیری ویژگی است که، با توجه به حالت شبکه پتری اولیه، μ، می‌توانیم به این حالت ،μ ، در دفعات محدودی بازگردیم. در شکل (۹-۲۱) حالت اولیه، $\mu_۰$ ، برگشت‌پذیر نیست، چرا که نمی‌توانیم از این حالت در مراحل محدود بازگردیم.



شکل (۹-۲۱): برگشت‌پذیری و دسترس‌پذیری شبکه پتری

از طرفی دیگر، اگر حالت اولیه $\mu' = (۰,۱,۰,۲)$ باشد، پیدا می‌کنیم که با هر چهار گذرگاه حالت به این حالت بازگردیم. بنابراین، این حالت برگشت‌ناپذیر است.



شکل (۹-۲۲): شبکه پتری بن‌بست دار

یک شبکه پتری با بن‌بست مواجه می‌شود اگر هیچ گذرگاهی در شبکه وجود نداشته باشد که فعال باشد. در مثال نشان داده‌شده در شکل (۹-۲۲)، شبکه دارای نشانه گذاری اولیه، $\mu_0 = (0,0,2,0)$ است. این نشانه گذاری در اینکه هیچ گذرگاهی فعال نشود نتیجه می‌دهد و بعد از یک مقدار نامحدود زمان که فعال‌شده است هیچ امیدی وجود ندارد. به همین ترتیب، یک شبکه پتری زنده در نظر گرفته می‌شود اگر هیچ گذرگاهی در آن فعال نباشد.

یک شبکه پتری محدود به k مکان در نظر گرفته می‌شود اگر برای همه مکان‌ها در شبکه k نشانه یا کمتر در هر مکان برای همه حالات ممکن شبکه وجود داشته باشد. برای مثال، در شبکه پتری نشان داده‌شده در شکل (۹-۲۱)، ما یک شبکه محدود سه‌تایی داریم، چرا که همه مکان‌ها در شبکه حداکثر سه توکن یا کمتر در مکان خود برای همه حالات قابل دسترسی درون شبکه پتری دارند.

انحصار متقابل ویژگی نهایی است که برای شبکه‌های پتری سنتی تعریف خواهد شد. انحصار متقابل برای مکان‌ها و برای گذرگاه‌ها تعریف می‌شود. احتمال برای جفت مکان‌ها یا گذرگاه‌ها درون یک شبکه پتری برقرار است. دو مکان، P_a و P_b، در یک سیستم شبکه پتری انحصار متقابل دارند اگر برای همه حالات در مکان سیستم P_a و P_b هیچ‌گاه همزمان دوتایی با توکن‌ها بارگذاری نشوند. این حاکی از این است که اگر یکی توکن داشته باشد دیگری ندارد. به طور مشابه، برای گذرگاه ها، یک شبکه

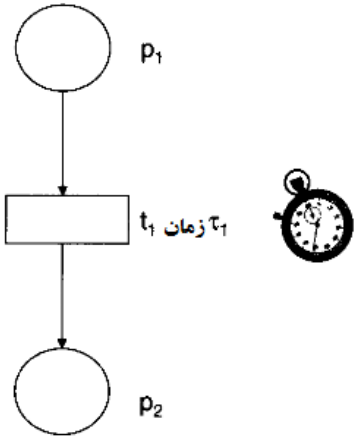
پتری دارای انحصار متقابل گذرگاه‌ها هستند اگر برای همه جفت گذرگاه‌ها در شبکه پتری، t_a و t_b، تنها بتوانند در طول هر قابلیت دسترسی حالت توسط شبکه فعال شوند. ویژگی‌های ارائه شده در این بخش عمومی هستند و می‌توانند بر بیشتر شبکه‌های پتری اعمال شوند.

۹-۴ شبکه‌های پتری زمان بندی شده

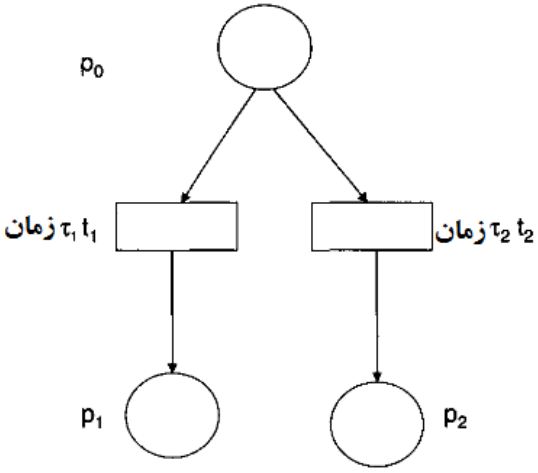
شبکه‌های پتری پوشش داده‌شده در بخش قبلی دارای گذرگاه‌هایی هستند که در زمان شلیک هیچ زمانی برای حرکت توکن از یک مکان به مکان دیگر اتخاذ نمی‌کند. در هر سیستم واقعی یک فعالیت زمان محدودی را برای اجرای عملیاتش در نظر می‌گیرد. برای مثال، برای خواندن یک فایل از یک دیسک، برای اجرای یک برنامه، یا برای برقراری ارتباط با دیگر ماشین‌ها به‌صورت بلادرنگ زمان محدودی در نظر گرفته می‌شود. افزودن زمان به شبکه پتری به مدل‌ساز شبکه پتری ابزار قدرتمند دیگری را برای مطالعه عملکرد سیستم‌های کامپیوتری ارائه می‌دهد. زمان می‌تواند با گذرگاه‌ها، با انتخاب مسیرها، با انتظار در مکان، با مهارکننده‌ها، و با هر مؤلفه دیگری از شبکه پتری در ارتباط باشند.

معمول‌ترین روشی که زمان در مدل‌سازی شبکه پتری استفاده می‌شود با گذرگاه‌ها در ارتباط است. این به دلیل این است که شلیک یک گذرگاه می‌تواند به‌عنوان اجرای یک رخداد مدل شده دیده شود – برای مثال، یک چرخه اجرای CPU. گذرگاه‌هایی که ارتباط زمانی دارند را گذرگاه‌های زمان‌بندی‌شده گویند. این گذرگاه‌های زمان‌بندی‌شده به‌صورت گرافیکی با یک مستطیل یا میله‌های نازک نشان داده می‌شود و با طراحی آغازشده با t شناخته می‌شوند.

در شکل (۹-۲۳)، گذرگاه t_1 یک گذرگاه زمان‌بندی‌شده با زمان t_1 به‌عنوان بازه زمانی برای تکمیل شلیک امکان‌پذیر است. زمانی که یک گذرگاه فعال شود، دوره زمانی آن، تایمر کلاک تنظیم می‌شود و شروع به کاهش می‌کند. زمانی که تایمر به صفر برسد، گذرگاه فعال می‌شود، توکن را از مکان ورودی به مکان خروجی انتقال می‌دهد. در مثال نشان داده‌شده در شکل (۹-۲۳)، زمانی که توکن به محل P_1 برسد، تایمر برای گذرگاه t_1 به τ_1 تنظیم می‌شود و کاهش می‌یابد. زمانی که τ_1 واحد زمانی بگذرد، گذرگاه فعال می‌شود و توکن از P_1 گرفته می‌شود و به P_2 داده می‌شود. کاهش تایمر باید با سرعت ثابتی برای گذرگاه‌های در مدل شبکه پتری صورت گیرد. در این روش گذرگاه‌ها برای مدل کردن عملیات برخی از عناصر درون سیستم مدل شود، ساخته می‌شود.



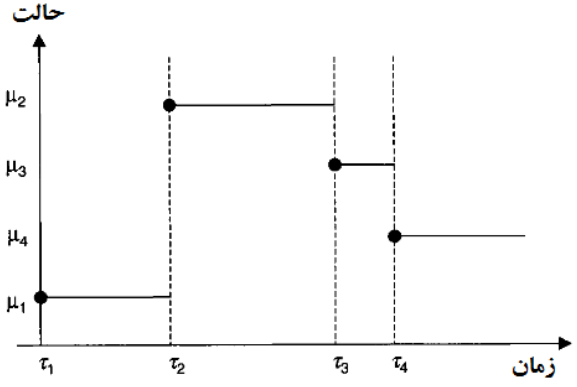
شکل (۹-۲۳): شبکه پتری زمان بندی شده



شکل (۹-۲۴): شبکه پتری زمان بندی شده با تضاد

چیزی که باید در مورد آن فکر کنید این است که چه زمانی گذرگاه با توجه به توکن فعال‌سازی اولیه استفاده شده برای شلیک یک گذرگاه رقابتی به غیرفعال تبدیل می‌شود. این شرط در شکل (۹-۲۴) نشان داده‌شده است. اگر فرض می‌کنیم که زمان برای گذرگاه t_1 کمتر از گذرگاه t_2 باشد، سپس، زمانی که مکان P_1 یک توکن را دریافت کند، دو تایمر شروع به شمارش می‌کنند. در زمان (τ_1) در

آینده، تایمر برای t_1 به مقدار صفر می رسد، که به شلیک گذرگاه t_1 نتیجه می‌دهد. از آنجایی که فعال سازی توکن t_2 هم اکنون صورت می گیرید، t_2 دیگر فعال نیست، و تایمر آن (τ_2) شمارش را متوقف می کند. سؤال حال این است که با گذرگاه تایمر t_2 چه کار باید بکنیم. دو امکان وجود دارد. اول اینکه به‌سادگی تایمر را برای سیکل بعدی که مکان P_1 دارای یک توکن است مجدد تنظیم کنیم و t_2 فعال شود. در این مورد، در صورتی که مکان P_1 حالتی داشته باشد که بیش از توکن داشته است، گذرگاه t_2 هیچ گاه شلیک نمی کند. دومین روش ممکن برای هندل کردن این شرایط این است که به تایمر گذرگاه t_2 اجازه دهیم که مقدار تایمر کلاک را نگه دارد $(\tau_2 - \tau_1)$. در دومین مورد، زمانی که توکن بعدی در مکان P_1 دریافت می‌شود، تایمر برای گذرگاه t_1 تایمر کلاک را مجدد به τ_1 تنظیم می کند، و گذرگاه t_2 از زمان $(\tau_2 - \tau_1)$ شروع به کاهش می کند.



شکل (۹-۲۵): گراف زمان بندی انتقال حالت

اگر زمان باقی‌مانده در تایمر گذرگاه t_2 کمتر از تایمر گذرگاه t باشد، سپس گذرگاه t_2 شلیک خواهد کرد، گذرگاه t_1 را با زمان باقی‌مانده $((\tau_2 - \tau_1) - \tau_1)$ به جا می گذارد. انتخاب اینکه کدام پروتکل استفاده شود به سیستمی که می‌خواهیم مدل کنیم بستگی دارد.

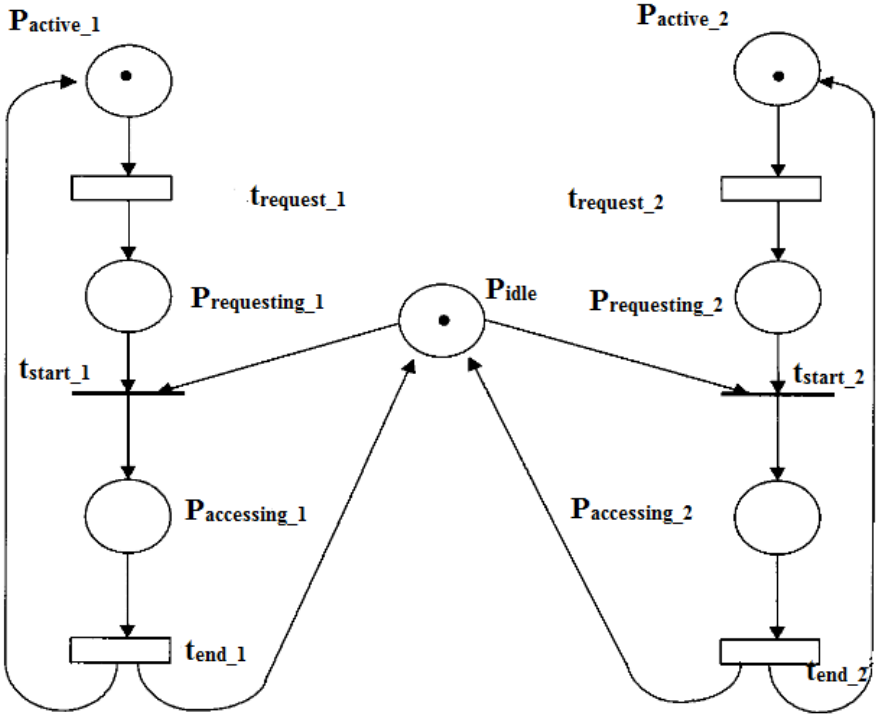
زمان‌بندی نیاز ندارد که بر اساس تایمرها و شمارنده‌ها انحصاری باشد. برخی از مدل‌های شبکه پتری با استفاده از گراف‌های زمان‌بندی گذرگاه حالت پیشنهادشده‌اند. در این مورد، هر حالت ممکن، μ ، شمرده می‌شود، دوره زمانی برای هر حالت منحصر به پیمودن این حالت به حالت بعدی در توالی تنظیم

می‌شود (شکل ۹-۲۵). در این شکل، حالت μ_1 نیازمند واحد زمانی $(\tau_2 \; or \; \tau_1)$ برای حرکت از حالت μ_1 به حالت μ_2 برای همه حالات تعریف‌شده در سیستم هستند. این می‌تواند با استفاده از دنباله گذرگاه‌های زمان‌بندی‌شده نشان داده شود، که ترتیب هر گذرگاه را در رابطه با همه موارد دیگر نشان میدهد. این تشریحی ممکن است مانند زیر باشد:

$$\left[(\tau_1 , t_1); (\tau_2, t_2); \dots , (\tau_j, t_j); \dots \right] \quad (۹-۷)$$

گذرگاه‌ها می‌توانند به‌عنوان گذرگاه‌های فوری نشان داده شوند، به همین ترتیب گذرگاه‌ها بدون هر تأخیر زمانی با آن‌ها در ارتباط هستند. برای مدل کردن این شبکه‌های پتری زمان‌بندی‌شده، ما به‌سادگی از یک خط ممتد برای حالت گرافیکی یا یک تایمر ۰ استفاده می‌کنیم اگر از نماد شبکه پتری استفاده کنیم.

در مثال نشان داده‌شده در شکل (۹-۲۶)، از گذرگاه‌های بلافصل برای اتخاذ یک منبع استفاده می‌کنیم. این عمل مانند سمافور در سیستم واقعی است.

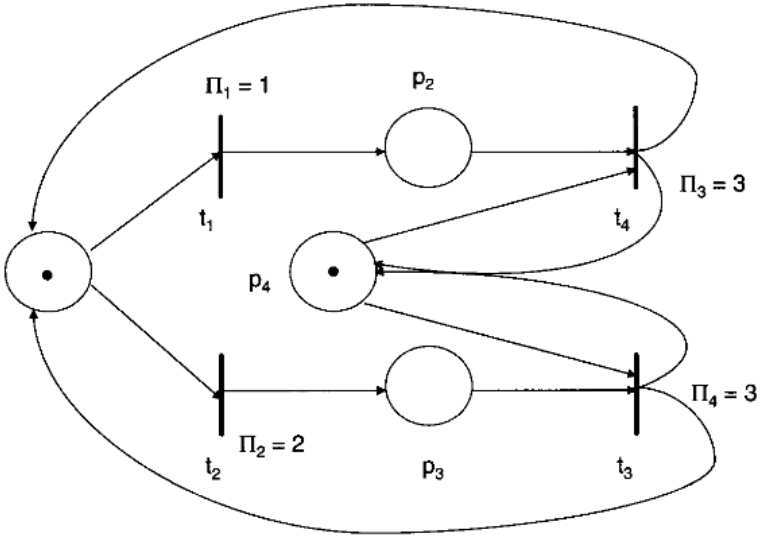


شکل (۹-۲۶): شبکه پتری زمان بندی شده با گذرگاه‌های بلافصل

یک فرآیند فعال برای برد منابع تلاش می‌کند، در حالی که در موارد دیگر مجبور هستند که تا زمانی که منابع بار دیگر آزاد شوند منتظر بمانند.

۹-۵ شبکه‌های پتری مبتنی بر اولویت

شبکه‌های پتری تعریف‌شده در بخش قبل بر اساس شبکه‌های پیاده شده در اولین بخش این فصل تعریف‌شده اند. برای ادامه دادن این روند ما بر اولویت در شبکه پتری نگاهی می کنیم. مدل رسمی حال نیازمند عناصر اضافی است. شبکه پتری با یک نه تایی تشریح شده است، $M = (P, T, I, O, H, \Pi, Par, Pred, \mu)$، p مجموعه مکان‌هایی را برای شبکه پتری M نشان می‌دهد. T مجموعه گذرگاه‌های شبکه پتری M است. Iمجموعه توابع ورودی برای گذرگاه‌های M است. O مجموعه توابع خروجی برای گذرگاه‌های M را نشان می‌دهد. O مجموعه توابع خروجی را برای M نشان می‌دهد. H توابع مهارکننده تعریف‌شده بر مجموعه گذرگاه‌ها در M را نشان می‌دهد. H توابع مهارکننده در مجموعه‌ای از گذرگاه‌های در M را تعریف می کند. Par مجموعه پارامترهایی را برای این شبکه پتری نشان می‌دهد، $Pred$پیش‌بینی‌هایی را نشان می‌دهد که تعریف می کند که چگونه مجموعه پارامترها می‌توانند توزیع شوند. نماد μ مجموعه نشانه گذاری برای این شبکه پتری را نشان می‌دهد، Π تابع اولویت تعریف‌شده در همه گذرگاه‌های شبکه پتری M را نشان می‌دهد. تابع Π اولویت‌ها برای هر گذرگاه برای یک مجموعه از مقادیر عدد صحیح نشان‌دهنده اهمیتی است که در موارد دیگر در شبکه نگاشت می‌شود.



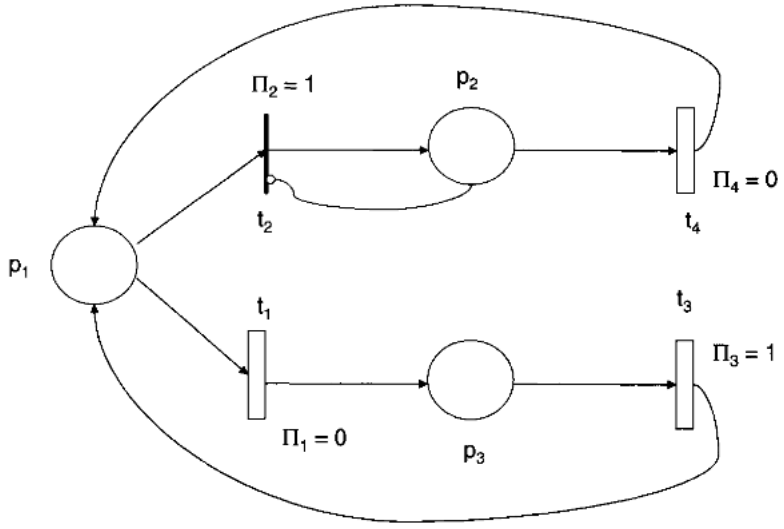
شکل (۹-۲۷): شبکه پتری مبتنی بر اولویت

اگر یک گذرگاه امتیاز انحصاری داشته باشد و هیچ گذرگاه اضافی در این شبکه وجود نداشته باشد که نشانه گذاری با الویت را بزرگ‌تر از الویت گذرگاه ارائه دهند، سپس فعال می‌شود، به‌صورت رسمی این مجموعه از شرایط به شکل زیر هستند:

$$t_i \; is \; enabled \quad if \; and \; t_j \in T only \quad if \; t_j < t_i \qquad (۹-۸)$$

یک گذرگاه که این معیارها را برآورده سازند می‌تواند شلیک کند. نتایج شلیک مشابه با شبکه‌های بدون اولویت است. در مثال نشان داده‌شده در شکل (۹-۲۷)، گذرگاه t_1 دارای پایین ترین اولویت در ۱ است، گذرگاه t_2 داری اولویت بعدی ۲ است، و گذرگاه t_3 و ۴ بالاترین اولویت ۳ را دارند. اگر با یک نشانه گذاری اولیه $\mu = (1,0,0,1)$ شروع کنیم، تنها گذرگاه t_2 در دسترس است، چرا که دارای بالاترین اولویت است و تعدادی توکن از تابع ورودی‌اش در صورت نیاز برای فعال‌سازی در دسترس دارد. اگر حالات ممکن بعدی را حساب کنید، می‌بینید که t_1 هیچ‌گاه فعال نیست، چرا که قدرت غلبه بر اولویت گذرگاه t_2 را ندارد.

با استفاده از اولویت و زمان‌بندی می‌توانیم کامل شدن را با تعریف شرایطی مانند تعرض، ابهام و همزمانی انجام دهیم. برای مثال، شبکه پتری نشان داده‌شده در شکل (۹-۲۸) دارای ویژگی‌های زمان‌بندی و اولویت است. با ترکیب این ویژگی‌ها می‌تواند سیستم را بین دو رخداد تغییر وضعیت دهد. مهارکننده گذرگاه t_2 باعث می‌شود که گذرگاه‌های بلافصل تا زمانی که سرویس تکمیل شود مسدود شوند. در این روش به گذرگاه t_1، با اولویت پایین‌تر اجازه داده می‌شود که سرویس بگیرد در حالی که مکان p_2 توکن دارد.



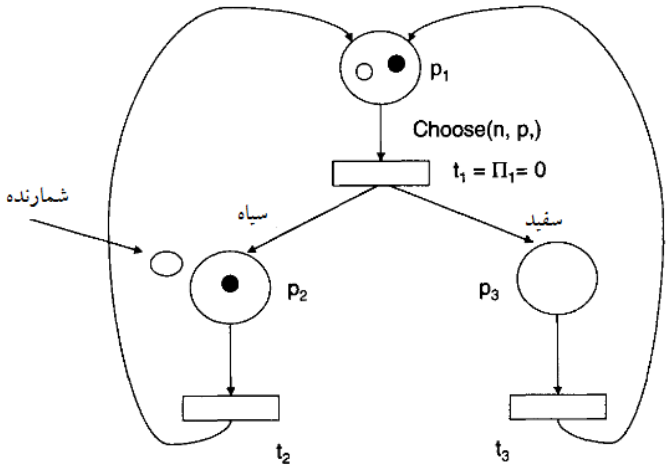
شکل (۹-۲۸): شبکه اولویت دار و زمان بندی شده

۹-۶ شبکه‌های پتری رنگی

در این بخش، ما برخی از مفاهیم پایه شبکه‌های پتری رنگی را معرفی می‌کنیم. شبکه‌های پتری رنگی بعد دیگری را به توکن و به معیارهای استفاده شده در تعیین شلیک با اضافه کردن انواع توکن‌های متفاوت اضافه می‌کند. توکن‌ها حال می‌توانند توابع متفاوتی را نشان دهند. برای مثال، می‌توانیم از توکن‌های متفاوت برای نمایش فراخوانی‌های سیستم‌عامل‌ها یا کلاس‌های متفاوت کاری استفاده کنیم. این توکن‌های متفاوت می‌توانند برای تعیین اینکه کدام گذرگاه از گذرگاه‌های متعددی موجود می‌توانند عملی باشند، استفاده کنند.

برای نمای گرافیکی مورد گفته شده از توکن‌های رنگی استفاده می‌کنیم. مجموعه همه رنگ‌های ممکن برای توکن‌ها کاردینالینی مجموعه توکن را نشان می‌دهد. با استفاده از این مجموعه توکن می‌توان مجدد تعریف شبکه پتری را اصلاح کنیم، به خصوص، برای اصلاح قوانین شلیک (جبر پیوندی) برای همه گذرگاه‌های تعریف‌شده در شبکه ما. برای مثال، در شکل (۹-۲۹)، تنها دو نوع توکن وجود دارد: سیاه وسفید. این می‌تواند انواع متفاوت کار را نشان دهد. گذرگاه‌ها می‌تواند اولویت و زمان مرتبط با آن‌ها

را مانند قبل داشته باشد و می‌تواند برای عمل تنها بر نوع توکن خاص تعریف شود. همان طور که در شکل (۲۹-۹) نشان داده‌شده است، گذرگاه t_1 دارای اولویت و زمان مرتبط با آن است. کمان‌ها نیز دارای جزییات اضافی مرتبط با آن‌ها هستند. کمان از p_1 به t_1 دارای انتخاب شرطی (n, p_1) است، که n توکن را برای آزادسازی در گذرگاه انتخاب میکند. کمان‌های دیگر برای انتخاب نوع خاصی از توکن استفاده می‌شوند. برای مثال، کمان‌هایی که از گذرگاه t_1 خارج می‌شوند و به مکان p_2 و p_3 می‌روند فیلترهایی برای آن‌ها هست که تنها اجازه می‌دهند که توکن‌های مشکی به مکان p_2 بازگردد و توکن سفید به مکان p_3 برگردد. شرایط کمان‌ها می‌تواند تا جایی که می‌خواهیم پیچیده باشد. ما می‌توانیم از شرایط پیچیده‌ای استفاده کنیم که به n_1 از یک نوع توکن، n_2 از نوع دیگر توکن، هیچی از نوع سوم توکن نیاز داشته باشد، قبل از اینکه تنها یک توکن در مسیر مشخص آزاد شود. با استفاده از این روش‌های پیچیده می‌توانیم هر شرطی که ممکن است در سیستم کامپیوتری که در حال مدل‌سازی آن هستیم رخ دهد را مدل کنیم.



شکل (۹-۲۹): شبکه پتری تعمیم یافته

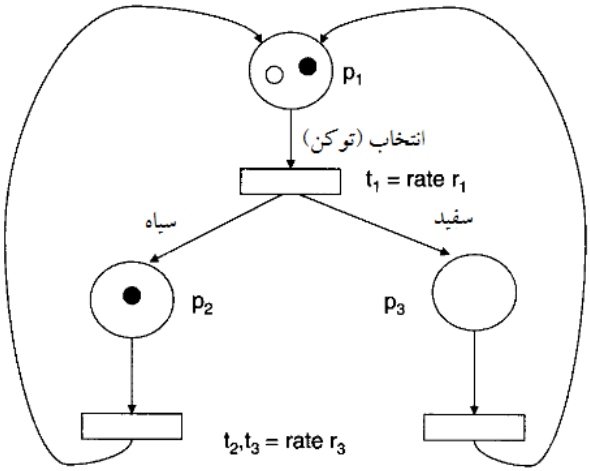
۹-۷ شبکه‌های پتری تعمیم یافته

شبکه‌های پتری تعمیم یافته برای ارائه اصلاحات دیگری برای روش‌های بحث شده استفاده شده اند. همه مدل‌های دیگر گذرگاه‌هایی دارند، که در زمان شلیک، بلافاصله اجرا می‌شوند یا در بازه زمان تعریف‌شده‌ای اجرا می‌شوند. دوره زمانی برای یک شلیک گذرگاه، در زمان تنظیم، ثابت است و در طول دوره عمر مدل تغییر نمی‌کند. اگر زمان بندی سیستم ما قطعی باشد و هیچ قابلیت تغییر وجود نداشته باشد، این کافی است. در واقعیت می‌دانیم که این برای بیشتر سیستم‌های واقع بینانه برقرار نیست. مدل‌های شبکه پتری تعمیم یافته این را با ارائه مکانیزمی برای یک تأخیر آنی تصادفی، به‌صورت نمایی توزیع‌شده با گذرگاه‌های زمان بندی شده تغییر می‌دهد.

علاوه بر این نیاز است که این شرایط برای شلیک برآورده شود که به تابع تعریف‌شده بر گذرگاه‌ها در سیستم نیاز دارد. این تابع جدید را تابع گذرگاه وزن یا نرخ گویند. این تابع $W(t_k, \mu)$، باید برای هر گذرگاه و حالت در شبکه تعریف شود. اگر تابع نیاز نیست که برای همه نشانه‌گذاری‌ها تعریف شود، سپس می‌توانیم به‌سادگی با $W(t_k)$ به تابع اشاره کنیم، در جایی $t_k \in T$ برقرار است. نتایج این تابع، $W(t_k, \mu)$ یا $W(t_k)$، نرخ انتقال t_k در نشانه گذاری μ را فراخوانی می‌کند، اگر t_k زمان بندی شود وزن انتقال t_k در μ صورت می‌گیرد اگر t_k بلافصل باشد. مقدار این نتایج در یک متغیر تصادفی با تابع نمایی تعریف‌شده برای گذرگاه‌ها حول مقدار میانگین انتخاب شده تعریف‌شده است.

شلیک یک گذرگاه در یک شبکه تعمیم یافته مانند یک نرخ زمان بندی شده رخ می‌دهد جز اینکه زمان شلیک گذرگاه محاسبه شده است. هر گذرگاه در شبکه باید یک نرخ، r، تعریف‌شده داشته باشد. نرخ میانگین مقدار استفاده در محاسبه زمان واقعی برای استفاده در شلیک گذرگاه است. زمانی که گذرگاه فعال شود، مقدار تایمر محاسبه شده آن با استفاده از یک سیستم کاهنده از مقدار، کاهش میابد تا زمانی که به ۰ برسد یا توکن‌ها را از دست بدهد. مانند شبکه زمان بندی شده، می‌توانیم از اطلاعات اضافی یا سیاست‌هایی برای تصمیم گیری برای اینکه چگونه از حالت گذرگاه در زمانی که غیر فعال است استفاده کنیم. میتوانیم از زمان باقی مانده در زمان محاسبه شده استفاده کنیم، آن را به مقدار مشابهی تنظیم کنیم، یا زمان جدیدی را بر اساس میانگین مشابه انتخاب کنیم. همه دارای معیارهایی براساس نوع عنصر مدل شده هستند.

تفاوت دیگر در این کلاس سیستم مفهوم حالت سیستم است. حالت سیستم از یک حالت به حالت دیگر بر اساس شلیک همه گذرگاه‌های فعال و آماده در طول بازه زمانی حاضر صورت می‌گیرد. در شکل (۹-۳۰)، نشانه گذاری اولیه $\mu_{\cdot} = (۲,۰,۰)$ است. اگر ما از توابع شبکه رنگی شده استفاده کنیم و به‌سادگی زمان‌های گذرگاه را به نرخ‌ها تغییر ندهیم، حال یک شبکه پتری تعمیم یافته داریم. در مثال، گذرگاه $t_۱$، زمانی که برچسب بخورد، یک نرخ گذرگاه را با استفاده از میانگین نرخ $r_۱$، به‌عنوان مقدار خورانده شده به نمای منفی محاسبه می‌کند. با استفاده از نرخ محاسبه شده تایمر تا زمانی که به نقطه شلیک برسد کاهش می‌یابد. ما سپس باید از توکن انتخاب شده برای تعیین اینکه کدام مسیر در ترک گذرگاه انتخاب شود استفاده می‌کنیم. در مثال، اگر تابع انتقال سفید را انتخاب کند، سپس مسیر سفید انتخاب می‌شود، که در حالت سیستم جدید نتیجه می‌دهد، $\mu_۱ = (۱,۰,۱)$



شکل (۹-۳۰): شبکه پتری تعمیم یافته

یک مفهوم مهم این شبکه‌ها این است که حالت‌های مشابه را زمانی که از یک حالت مشخص اجرا می‌شوند، با توجه به تصادفی بودن زمان شلیک گذرگاه ممکن محاسبه نمی‌کنند. این یک ویژگی مطلوب برای این مدل‌هایی است، چرا که از این برای ویژگی‌های بدون حافظه نرخ‌های ورود و نرخ‌های سرویس در صف استفاده می‌کنیم.

۹-۸ خلاصه

شبکه‌های پتری به‌عنوان ابزار مدل‌سازی از اواخر دهه ۱۹۶۰ در دسترس هستند. از این نقطه در زمان دچار تحول و بهبودهای بسیار شد. در ابتدا آن‌ها بیشتر کنجکاو کننده بودند، چرا که هیچ ابزاری برای ساخت و تحلیل آسان مدل‌ها وجود نداشت. از اولین روزها، بسیاری از ابزارهای کامپیوتری شده در دسترس قرار گرفتند، که به ما اجازه دادند که شبیه سازی ساختار مدل را اجرا کنیم. اطلاعات عملکرد را جمع آوری کنیم. برخی از ابزارهای تحلیل شبکه پتری تخصصی توسعه‌یافته و به‌صورت گسترده در دسترس قرار گرفتند.

در این فصل ما مقدمه پایه‌ای از شبکه‌های پتری، ویژگی‌های آن‌ها، و قابلیت‌های مدل‌سازی آن‌ها را پوشش دادیم. این بررسی پایه با اصلاحات بیشتر بر مدل پایه دنبال شد. ابتدا مفهوم زمان انتقال را به مفهم مدل اولیه اضافه کردیم. سپس آن را با مفهوم اولویت انتقال ترکیب کردیم. سپس، مفهوم نوع توکن و قوانین شلیک انتقال در شبکه‌های پتری را معرفی کردیم. سرانجام بررسی شبکه‌های پتری را با بحث بر شبکه‌های پتری تعمیمی یافته و قابلیت‌های آن‌ها تکمیل کردیم.

فصل دهم

پلتفرم سخت‌افزاری، ابزارها، اندازه گیری، استخراج اطلاعات و تحلیل

در فصل‌های قبل، به مدل‌سازی دیدگاه‌های گذشته – از مدل‌های شبیه سازی به صف بندی تا شبکه‌های پتری به تحلیل عملیاتی – پرداختیم. برای آن دیدگاه‌ها، تنها توانستیم مقدار محدودی از اطلاعات را روی سیستم واقعی اندازه گیری نماییم. اغلب، فرضیه‌ها تسهیل می‌شوند تا مدل‌های حاصل قابل محاسبه شوند، این کار به ما کمک می کند تا تحلیل تقریبی از رفتار سیستم به دست آوریم. علاوه بر این، شرایط بار که برای یک مدل تحلیلی یا شبیه‌سازی ایجاد می‌شود، معمولاً در محیط واقعی آزمایش نمی گردد. این عوامل دو شاخه دارند. مورد اول که یک تحلیل کامل‌تر است به دلیل نداشتن اطلاعات مناسب در محیط واقعی، بسیار دشوار است. مورد دوم، حتی با وجود انجام تحلیل‌های دقیق، اعتبار مدل و نتایج آن در بهترین حالت نیز ضعیف است. مورد دوم برای مدل‌های شبیه سازی همه منظوره درست است مثل مدل‌هایی که در این کتاب به آن پرداختیم. پیش از اینکه از یک شبیه سازی برای پیش بینی عملکرد هر سیستمی استفاده کنیم، نتایج مربوط به اجرای آن باید با خطوط اصلی موجود مقایسه شود و شبیه سازی باید بر همان اساس توجیه گردد. یک روش برای دستیابی به این نتایج، داشتن ابزار و جمع آوری اطلاعات مربوط به عملکرد روی یک سیستم واقعی است. نتایج این ارزیابی با نتایج پیش بینی شده از مدل شبیه سازی همان سیستم مقایسه می‌شود. اگر نتایج با معیار از پیش تعیین شده منطبق باشد، می گوییم که مدل معتبر است.

این فصل درباره استفاده از پلتفرم سخت‌افزاری نمونه به‌عنوان یک ابزار برای بررسی معیارهای واقعی برای مقادیر عملکرد مورد نظر، برای اجرای آزمایش‌های کنترل شده جهت تعیین ویژگی‌های عملیاتی بخش‌های مختلف یک شبکه و برای اعتبار سنجی مدل‌های شبیه سازی نرم افزاری، بحث می‌کند. به ویژه، درباره پیاده سازی یک پلتفرم سخت‌افزاری بحث می‌کنیم و مقادیر قابل ارزیابی مورد نظر خود را تعریف می‌نماییم و روابط عملیاتی برای مقادیر مبهم را به دست می‌آوریم و به نتیجه گیری می‌رسیم. ساخت یک پلتفرم تک منظوره تنها برای برآورد عملکرد سیستم نهایی، پر هزینه است. بااین‌حال، برای آزمایش و تعیین اعتبار طرح مفروضات، برای کسب تجربه در سیستم و برای ایجاد یک وسیله جهت پیشرفت بیشتر، یک اثبات از نمونه اولیه مفهوم ایجاد می‌شود. با توجه به اینکه یک اغلب یک سیستم نمونه وجود دارد، بهتر است که از تجهیزات و قواعد آزمایش در طرح نمونه اولیه استفاده شود. اگر این آزمایش در حوزه نمونه سازی اولیه انجام شود، هزینه مربوط به امکانات اندازه گیری عملکرد ویژه، قابل قبول تر خواهد بود. برخی از امکانات مهمی که برای یک نمونه خاص در این فصل توصیف می‌کنیم، می‌توانند شامل یک اساس زمانی سیستمی برای کسب معیارهای همزمان، سخت‌افزار برچسب زمانی یا نرم افزاری برای رویدادهایی با برچسب زمانی، شمارنده‌هایی برای ثبت تعداد رخدادهای مهم و گردانندگان موضوعاتی باشند که یک بار شناخته شده را در یک سیستم مدل تزریق کنند. البته، مطلوب این است که این امکانات تا حد ممکن از قبل تعریف‌شده باشند به نحوی که کاربرد آن‌ها با عملیات طبیعی شبکه تحت بررسی تداخل نداشته باشد. در برخی موارد، بخش‌هایی از پیکربندی نرم افزاری سیستم نهایی ممکن است با تجهیزات اندازه گیری تک منظوره جایگزین گردد. در ادامه این

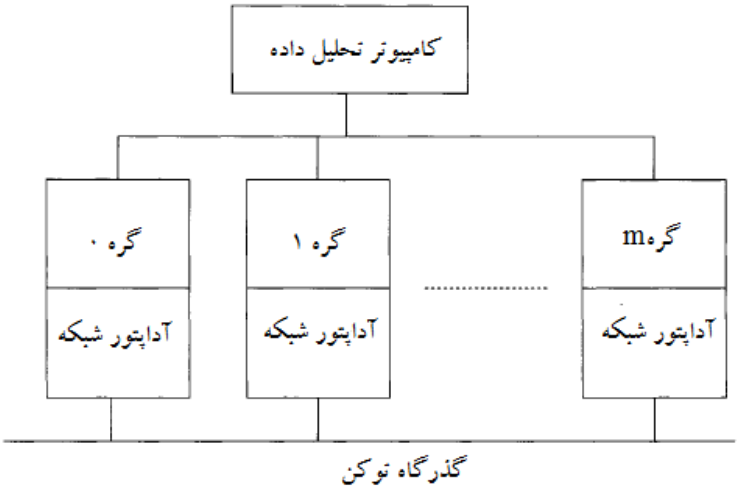
فصل درباره پیکربندی شبکه نمونه بحث خواهیم کرد و تکنیک‌های مستقر برای اندازه گیری مشخصات کارایی آن را نشان خواهیم داد.

شبکه ای که ما به آن می‌پردازیم در یک پلتفرم نمونه قرار دارد که از آن برای جمع آوری اطلاعات استفاده می‌کنیم و می‌تواند در شبکه ترافیک ایجاد نماید. هر گره پلتفرم شامل یک کنترل کننده میزبان است که می‌تواند بار ترافیک شناخته شده را شبیه سازی نماید یا الگوهای خاصی از ترافیک پیام را تولید کند. این آزمایش‌ها آنقدر تکرار می‌شوند تا بتوانیم از معیارهای مختلفی استفاده کنیم یا یک پارامتر مربوط به ارتباط خاص ایجاد شود. بنابراین، مکانیزم‌های بارگذاری و ثبت سیستم نمونه را می‌توان برای مشاهده پدیده عملکرد شبکه‌های مختلف تحت کنترل در آورد. برای ایجاد یک پلتفرم نمونه مثل پلتفرمی که بحث کردیم، بهتر است که میزان هزینه‌های توسعه سخت‌افزاری به حداقل برسند تا یک سیستم منعطف ایجاد گردد که تغییرات موجود در طراحی شبکه با یکدیگر وفق داده شود و قابلیت‌های همه‌منظوره‌ای که پیش از این به آن‌ها اشاره کردیم ارائه گردند. یک راه برای کاهش هزینه توسعه سخت‌افزاری، استفاده از مؤلفه‌های سخت‌افزاری معتبر است. تمام مؤلفه‌های گره که مخصوص شبکه نیستند را می‌توان با استفاده از محصولات سطح سیستم استاندارد پیاده سازی نمود. برای مثال، پردازنده میزبان برای یک گره را می‌توان با یک کامپیوتر تک برد یا حتی با یک کامپیوتر شخصی پیاده سازی نمود.

انعطاف موجود در طراحی مؤلفه‌های مخصوص شبکه، برای به حداقل رساندن تأثیر تغییرات اجتناب ناپذیری که در حین طراحی شبکه و مرحله نمونه‌برداری رخ می‌دهند، ضروری است. یک روش خوب

برای ایجاد یک طرح نمونه منعطف این است که سرعت عملیات شبکه کاهش یابد. این کار اجازه می‌دهد که برخی از عملیات موجود در شبکه با مؤلفه‌های همه‌منظوره پیاده‌سازی شوند، مثل میکروکنترلرهای قابل برنامه نویسی یا ماشین وضعیت. بعد از اینکه طرح نمونه تجزیه و تحلیل شد و بر همان اساس یک پیکربندی تقریباً کامل ایجاد شد، این عملیات را می‌توان به پیاده سازی‌هایی با سرعت زیادتر انتقال داد. در اینجا فرض بر این است که گسترش یکنواخت سرعت عملیات در سراسر مؤلفه‌های حساس شبکه ما را به نتایجی می‌رساند که برای انعکاس عملکرد واقعی سیستم قابل مقیاس دهی است. این ممکن است در موارد دقیق تر درست نباشد، به‌عنوان‌مثال جایی که مشخصات سخت‌افزاری در سرعت‌های بالاتر تغییر می‌کنند، اما ممکن است قابلیت شبکه تغییری نداشته باشد. برای ایجاد یک گرداننده شبکه همه منظوره و قابلیت‌های جمع آوری داده، همیشه باید یک میزبان جداگانه داشته باشیم، که تنها قابلیت آن تولید ترافیک شبکه و جمع آوری نتایج می‌باشد. اگرچه ممکن است لازم باشد منابع اضافه ای در نظر بگیریم که تنها کار آن‌ها کمک به تولید ترافیک مجموعه داده است. لازم است که تا حد ممکن یک استاندارد شبکه لایه ای مثل سازمان بین المللی برای استانداردسازی مدل برای مدل مرجع اتصال سیستم‌های باز (ISO) وجود داشته باشد. با این کار، تغییرات ممکن است کم و بیش به سطحی برسند که بیشترین تأثیر را داشته باشند، درحالی که سطوح دیگر می‌توانند قابلیت‌های خود را به حداقل برسانند. بنابراین، همان استانداردهایی که در میان شبکه‌های مختلف درجه ای از همکاری ایجاد می‌کنند، می‌توانند به ما نمونه ای مفید برای ایجاد یک سیستم نمونه منعطف بدهند. برای مثال، پلتفرم سخت‌افزاری که در این مطالعه استفاده کردیم، شامل گره‌های متعدد شبکه متصل به یک LAN

گذرگاه نشانه بودند. هر گره دو کامپیوتر تک بردی دارد: یک کامپیوتر که قابلیت‌های میزبان شبیه سازی شده را پیاده سازی می‌کند (میزبان) و برای بارگذاری شبکه و جمع آوری داده فراهم شده است و دیگری که کنترل سطح بالایی برای سخت‌افزار شبکه دارد (پردازنده ورودی/خروجی یا IOP). علاوه براین، هر گره یک آداپتور شبکه دارد که عملکرد آن اجرای پروتکل شبکه می‌باشد.



شکل (۱۰-۱): پیکربندی پلتفرم کلی

در این مورد خاص، پلتفرم شبکه یک شبکه ارتباطی سریال همه منظور مدل‌سازی می‌کند. با یک میزبان ثابت و یک طرح IOP و با استفاده از آداپتورهای مختلف شبکه و سخت‌افزار نهایی می‌توان چند نوع شبکه مختلف را پیاده سازی نمود. شبکه ای که ما از آن استفاده خواهیم کرد، یک پروتکل دسترسی به توکن را آزمایش می‌کند. در این پروتکل، گره ای که توکن را نگه می‌دارد، حق انتشار داده را دارد، البته اگر ترافیک پیام برای انتقال توسط پردازنده میزبان به صف شده باشد. اگر گره پیامی برای ارسال

داشته باشد، پیام را در گذرگاه ارتباطی منتشر می‌نماید. تمام گره‌ها منتظر آدرس شناسایی خود در هدر پیام هستند و اگر پیام به مقصد آن‌ها باشد، آن را تأیید می‌کنند. بعد از انتقال کامل پیام، یا اگر پیامی برای ارسال باقی نمانده باشد، توکن به شیوه راند – رابین به گره بعدی شبکه می‌رود. گره بعدی ممکن است به‌صورت فیزیکی به گره فعلی نزدیک شود یا نشود. ساختار کلی پلتفرم شبکه در شکل (۱۰-۱) نشان داده‌شده است.

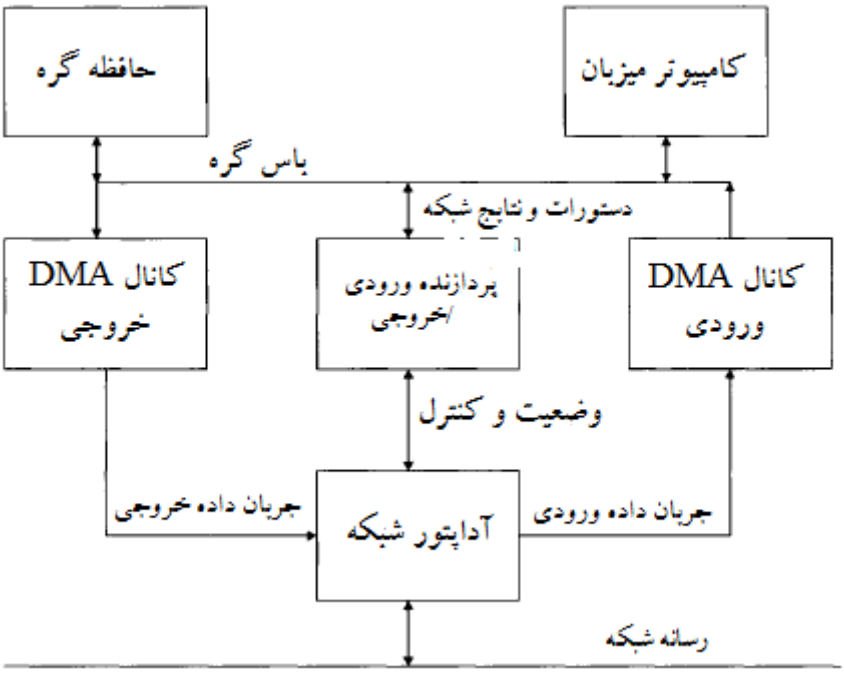
تعدادی از گره ها، که هرکدام یک آداپتور شبکه دارند، به یک گذرگاه توکن خطی و یک کامپیوتر تحلیل داده متصل می‌شوند. در حین اجرای آزمایش، از گذرگاه شبکه برای انتقال بار شبیه سازی شده استفاده می‌شود. بعد از تکمیل آزمایش، هر گره داده جمع آوری شده خود را به کامپیوتر تحلیل داده می‌فرستد تا مورد تجزیه و تحلیل قرار گیرد. هر گره موجود در پلتفرم شبکه یک معماری دارد (شکل ۱۰-۲).

کامپیوتر میزبان در این معماری دو قابلیت دارد. اول، پیاده سازی بخشی از پروتکل لایه ای و ارائه یک بار پیام شبیه سازی شده به آن. دوم، جمع آوری اطلاعات مربوط به عملکرد لازم برای تحلیل بعد از آن است. شکل (۱۰-۳) ساختار کلی نرم افزار میزبان را نشان می‌دهد که این قابلیت‌ها را پیاده سازی می‌نماید.

IOP جریان ترافیک پیام را از طریق آداپتور شبکه، در داخل و خارج شبکه کنترل می‌کند. علاوه بر این کانال‌های DMA را کنترل می‌کند و یک رابط استاندارد برای کامپیوتر میزبان فراهم می‌نماید و

آمار مربوط به عملکرد مخصوص شبکه را جمع‌آوری می‌کند. شکل (۱۰-۴) معماری کاربردی IOP را نشان می‌دهد.

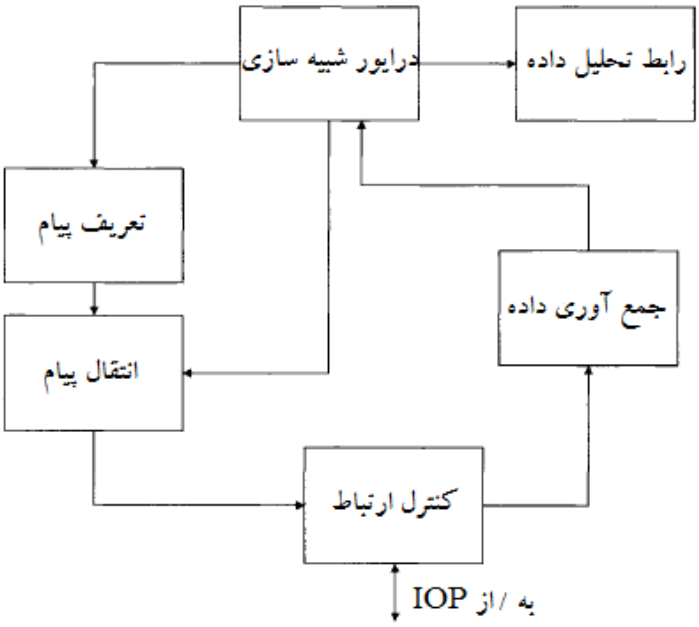
همان‌طور که پیش از این نیز اشاره کردیم، مزیت آن این است که مؤلفه‌های پلتفرم آن با یک استاندارد پروتکل لایه‌ای مواجه می‌شوند.



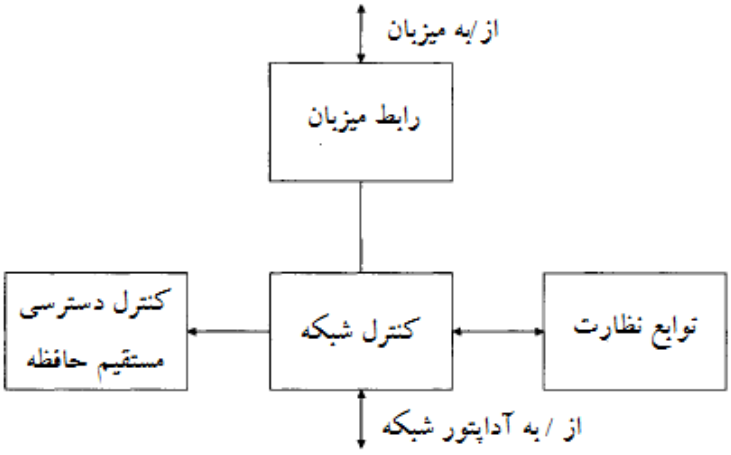
شکل (۱۰-۲): معماری گره بستر آزمایشی

پلتفرمی که در اینجا موردبررسی قرار می‌گیرد سطوح ۱ تا ۴ و بخشی از سطوح ۵ مدل OSI را برای پروتکل‌های لایه‌ای پیاده‌سازی می‌نماید. شکل (۱۰-۵) نشان می‌دهد که چگونه مؤلفه‌های مختلف

استاندارد می‌شوند. در مدل لایه‌ای شکل (۱۰-۵)، سطح فیزیکی توابع فیزیکی و الکتریکی موردنیاز برای پیوند به گره‌ها را پیاده سازی می‌کند.



شکل (۱۰-۳): معماری نرم افزاری میزبان



شکل (۱۰-۴): معماری کاربردی IOP

لایه لینک داده مکانیزم‌های لازم برای انتقال مطمئن داده در لینک فیزیکی را فراهم می‌کند. سطح ۳، سطح شبکه است که سوییچینگ داده در شبکه را کنترل می‌نماید. شبکه‌هایی که چند مسیر انتقال دارند، عملکرد سطح ۳ کنترل لینک‌هایی است که پیام‌ها از آن‌ها عبور می‌کنند. در سطح انتقال، امکان ارتباط بدون خطا میان گره‌ها ایجاد می‌شود. کنترل نشست شامل مقدار دهی اولیه، نگهداری و قطع نشست ارتباط میان فرآیندها است. سطح ۶ امکان ترجمه، فشرده سازی یا خدمات رمزنگاری/رمزگشایی را فراهم می‌نماید که موردنیاز برنامه کاربردی است. در رأس آن‌ها برنامه کاربردی قرار دارد که فرآیندی را گویند که از امکانات ارتباطی استفاده می‌نماید.

برای مثال، سطح ۱ و ۲ شبکه از طریق کابل‌های کوواکسیال، مرتب سازی و بسته بندی داده و هماهنگ‌سازی و تشخیص داده در داخل و خارج شبکه ارتباط فیزیکی ایجاد می‌کند.

بستر آزمایش	سطوح ISO/OSI
درایور شبیه سازی	سطح ۷ : کاربردی
اجرا نشده	سطح ۶ : ارائه / نمایش
تقریبا اجرا شده	سطح ۵ : نشست
IOP	سطح ۴: انتقال
	سطح ۳: شبکه
آداپتور شبکه	سطح ۲ : پیوند داده
	سطح ۱ : فیزیکی

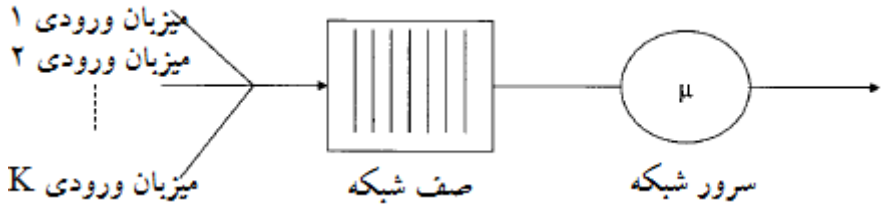
شکل (۱۰-۵): ضمایم پلتفرم ISO

از آنجایی که شبکه مورد نظر ما گذرگاه عمومی است، نیازی به سوییچینگ مرحله ۳ وجود ندارد. اما در موردی که این جزو عوامل محسوب می‌شود، این عملیات در IOP اجرا خواهد شد. کنترل حمل و نقل به خوبی در IOP پیاده سازی شده است، برای اینکه میزبان اجرا کننده تشخیص و بازانتشار خطا را از مدیریت بسته‌های پیام رها می‌کند. بخشی از سطح ۵ در میزبان پیاده سازی می‌شود، به نحوی که پیام‌ها بتوانند برای انتشار به گره‌ها دیگر بازسازی و صف بندی شوند. مکانیزم‌های لازم برای ایجاد اتصالات داخلی پیاده سازی نشدند. شبکه مورد نظر ما باید بعد از هر بسته پیامی که از گیرنده دریافت می‌کند، پیام تأیید بفرستد. عدم ارسال تأیید یا از بین رفتن آن باعث می‌شود که باز انتشار بسته با خطا مواجه شود. پیام‌ها در شناسه‌های فرآیند منطقی رسیدگی می‌شوند که در هر گره مطابق با مقدار اولیه ترسیم می‌گردند. در مدل پلتفرم، با دنباله‌ای از پیام‌ها به‌عنوان مجموعه‌ای از رویدادهایی با ترتیب زمانی

پشت سرهم رفتار می‌شود. تاریخ رویدادها در میزبان مطابق با یک توزیع احتمالی تولید می‌شوند که نماینده ویژگی‌های بارگذاری مطلوب هستند. زمان مربوط به تولید پیام برای تحلیل پس از اجرا ثبت و جمع‌آوری می‌شود. به محض اینکه یک پیام از لایه کنترل و شبکه عبور کند، دوباره برای تحلیل بعدی، برچسب زمانی می‌خورد. در بخش بعد، کاربرد این برچسب‌های زمانی و داده‌های دیگر را نشان خواهیم داد که پارامترهای ارزیابی عملکرد را به ما می‌دهند و برخی نتایج تجربی را نشان می‌دهیم که از این تکنیک‌ها استفاده کرده‌اند.

۱۰-۱ کسب پارامترهای ارزیابی عملکرد

همانطورکه پیش از این اشاره کردیم، ترافیک پیام برای شبکه تحت آزمایش در بخش میزبان تشکیل می‌شود. یک صف از پیام‌هایی که منتظر انتشار هستند، در حافظه گره پیاده سازی می‌شود که در شکل (۱۰-۲) نشان داده‌شده است. بنابراین گفته می‌شود که یک پیام از پردازنده میزبان وارد شبکه می‌شود و از همان نیز خارج می‌گردد. پیام بعد از ورودی به شبکه، به مجموعه ای از بسته‌ها شکسته می‌شود که هر یک به‌صورت سری توسط آداپتور شبکه و تحت کنترل IOP منتقل می‌گردند. تنها یک آداپتور شبکه می‌تواند کنترل گذرگاه را در هر زمانی حفظ کند (یعنی، تنها یکی می‌تواند در هرلحظه منتقل شود). این دسترسی سری تحت کنترل توکن قرار دارد. بنابراین، این شبکه یک سیستم صف بندی تک سروره را نشان می‌دهد که در آن سرویس ارائه شده همان انتقال پیام است. تمام پیام‌های موجود در سیستم دارای اولویت یکسان هستند به نحوی که سیستم تنها یک کلاس مشتری دارد.



شکل (۱۰-۶): سرور شبکه مفهومی

از آنجایی که دسترسی به شبکه توسط گره‌ها به‌صورت سری است و از آنجایی که تنها سرور موجود در شبکه خود شبکه می‌باشد، می‌توانیم تمام ورودی‌های بسته پیام به سرور را به‌عنوان ورودی از یک صف در نظر بگیریم. بنابراین، این صف مفهومی شامل ترکیبی از تمام پیام‌های موجود در صف‌های پیام است که بر حسب تاریخ مرتب شده اند. شکل (۱۰-۶) این مفهوم را نشان می‌دهد.

برای این مثال، فرض خواهیم کرد که پیام‌ها مطابق با یک توزیع پواسون وارد سرور می‌شوند. بنابراین، احتمال اینکه n ورودی از میزبان i در ورودی به طول t داشته باشیم برابر است با:

$$P\left(n \text{ مدت در ورودی } t\right) = \frac{(\lambda_i t)^n \, e^{-\lambda_i t}}{n!} \qquad (۱۰-۱)$$

که در آن λ_i میانگین ورودی در میزبان i است. برای توزیع پواسون، زمان میان ورودی‌ها به‌صورت نمایی توزیع می‌شود و زمان ورودی برای پیام‌ها در میزبان i به‌صورت زیر تولید می‌شود:

$$A_i = ۱ / \lambda_i \qquad (۱۰-۲)$$

میانگین زمان ورودی برای سرور شبکه مفهومی را می‌توان به‌صورت زیر نشان داد:

$$A_i = \sum_{i=1}^{k} 1 / \lambda_i \qquad (۳-۱۰)$$

می توانیم وضعیت سیستم را در دوره مشاهده به‌صورت تعداد پیام‌های منتظر انتقال در شبکه ارائه کنیم. از آنجایی که ویژگی روند ورودی پواسون و به موجب آن احتمال نه بیش از یک ورودی در هر دوره زمانی به صفر می‌رسد، تبدیلات وضعیت فرض یک مرحله ای را برآورده می‌نماید. یعنی، این سیستم تنها به حالات همسایگی منتقل می‌شود. یک وضعیت را با $n(t)$ نشان می‌دهیم و تعداد بسته‌های پیام منتظر برای انتقال در زمان t را تعریف می‌کنیم.

تحلیلی انجام می‌دهیم که مبتنی بر تکنیک‌های تحلیل عملیاتی موجود در فصل ۷ است. مقادیر این ارزیابی به‌صورت زیر نشان داده می‌شود:

W – زمان انتظار برای یک بسته پیام از ورودی به صف شبکه تا اتمام تبدیل اندازه گیری می‌شود.

B – زمان اشغال برای شبکه به‌صورت زمان کلی است که حداقل یک بسته پیام در شبکه وجود دارد.

این مقادیر از طریق ارزیابی سه مقدار اصلی اندازه گیری شده توسط ابزارهای سخت‌افزاری و نرم افزاری به دست می‌آیند. مقادیر اصلی اندازه گیری شده به‌صورت زیر هستند:

$A(n)$ – تعداد ورودی به سیستم در زمانی که n بسته پیام در سیستم وجود دارد.

$C(n)$ – تعداد تکمیل کار هنگامی‌که n بسته پیام در سیستم وجود دارد.

$T(n)$ – تعداد کل زمان وقتی که n بسته پیام در سیستم وجود دارد.

مقدار کل n از هر مقادیر قبل را به‌صورت زیر تعریف می‌کنیم:

$$A = \sum_{i=0}^{K} A\,(i) \quad \left(\text{ورودی‌ها}\right) \qquad (۴-۱۰)$$

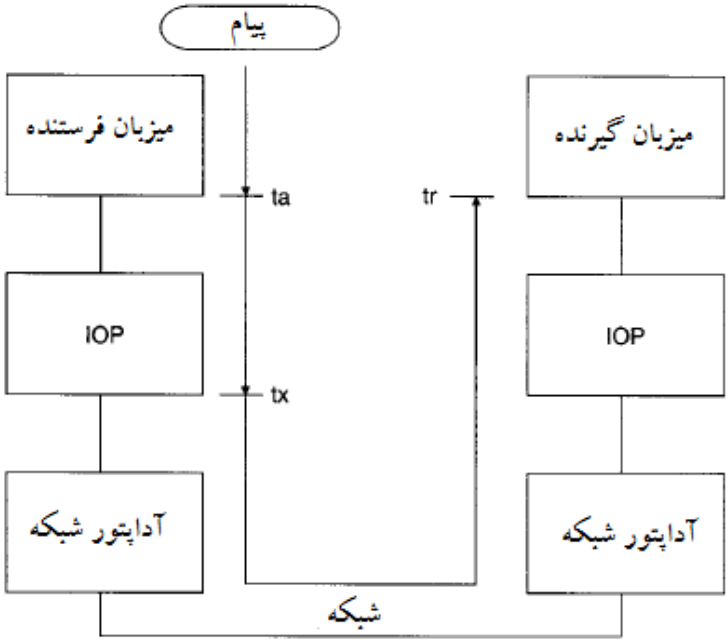
$$C = \sum_{i=0}^{k} C(i) \quad (\text{تکمیل}) \tag{۵-۱۰}$$

$$T = \sum_{i=0}^{k} T(i) \quad (\text{دوره مشاهده}) \tag{۶-۱۰}$$

در جمع‌های بالا، k نشان دهنده بیشترین تعداد بسته‌های پیام است که منتظر انتقال در فاصله مشاهده هستند. اگر تعادل بار برقرار باشد، تعداد کل ورودی‌ها معادل تعداد کل تکمیل پیام در دوره مشاهده خواهد بود. زمان انتظار و اشغال را می‌توان ازنظر این مقادیر به‌صورت زیر تعریف کرد:

$$W = \sum_{i=0}^{k} iT(i) \tag{۷-۱۰}$$

$$B = \sum_{i=1}^{k} T(i) = T - T(0) \tag{۸-۱۰}$$



شکل (۱۰-۷): زمان‌های انتقال پیام

همراه با این معیارها، سه معیار دیگر به دست می‌آوریم: زمان انتقال پیام (t_x)، زمان ورود پیام (t_a) و زمان پذیرش پیام (t_r). این معیارها در رابطه با یک انتقال پیام در شکل (۱۰-۷) نشان داده‌شده است. همانطورکه در فصل ۷ دیدید، برخی از پارامترهای مربوط به عملکرد را ازنظر مقادیر اصلی عملیاتی تعریف کردیم. این پارامترها به‌صورت زیر هستند:

(۱۰- میانگین طول صف : $N = W / T$

۹)

میانگین زمان پاسخ : $R = W / C$ (۱۰ –

۱۰)

کاربرد : $U = B \ / \ T$ (۱۱-۱۰)

میانگین زمان سرویس کار : $S = B \ / \ S$ (۱۲-۱۰)

توان عملیاتی شبکه : $X = C \ / \ T$ (۱۳-۱۰)

زمان سرویس شبکه : $S = \sum_{\text{همه پیام ها}} t_n \ / C \ \ where\ t_n = t_۲ - t_x$ (۱۴-۱۰)

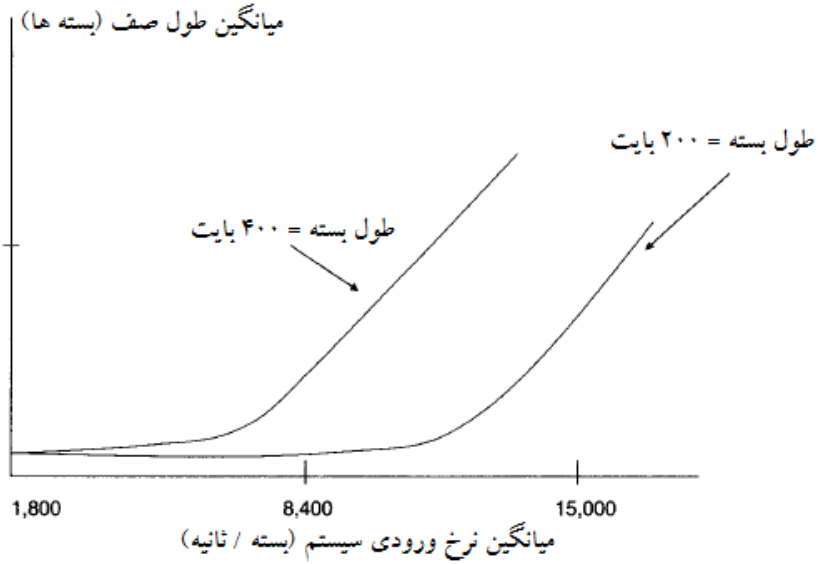
۵ مقدار اول از نتایج تحلیل عملیاتی استاندارد هستند. آخرین مورد مربوط به عملکرد مکانیزم‌های تبدیل، صرف نظر از زمان انتظار صف است.

۱۰-۲ آزمایش‌های مربوط به عملکرد شبکه

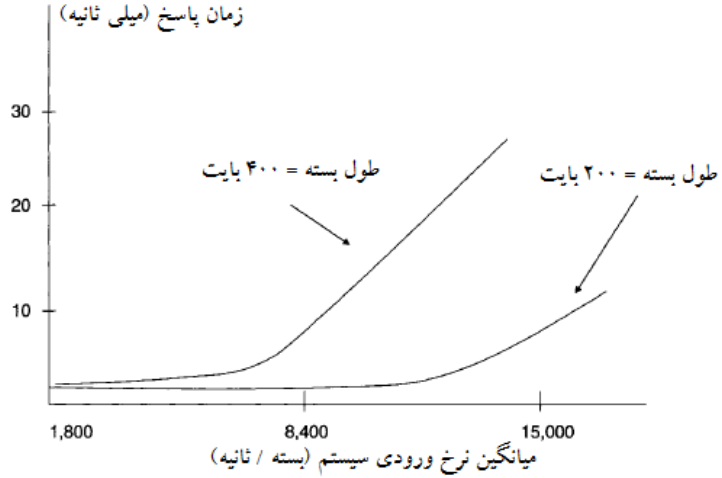
یک تحلیل با کمک تمام میزبان‌های شبکه با یک مولد نسبت ورودی شناخته شده روی این پلتفرم انجام شد و سپس از ترافیک پیام تولیدی برای بارگذاری شبکه استفاده گردید و معیارهای عملیاتی نیز که پیش از این تعریف کرده بودیم، جمع‌آوری گردید. بعد از اجرا، اندازه‌ها با یکدیگر ترکیب شدند و معیار عملکرد مطلوب محاسبه گردید.

آزمون نمونه روی پلتفرم شبکه اجرا شد و برای اینکه یک شاخص از زمانی را بدهد که یک پیکربندی شبکه خاص اشباع می‌گردد، به‌صورت فرمول ارائه گردید (یعنی، رویکرد ۱ کاربرد شبکه). برای مثالی که در اینجا مشاهده کردیم، بسته‌هایی به طول ۲۰۰ و ۴۰۰ بایت با سه گره تولیدکننده ترافیک آزمایش شدند. نسبت ورودی برای تمام سه گره یکسان تنظیم گردید و این نسبت تفاوتی از تقریباً ۶۰۰ بسته در ثانیه تا ۱۵۰۰۰ بسته در ثانیه داشت. اجرای آزمایش برای هر یک از نسبت‌های ورودی با فاصله انجام شد.

متوسط طول صف بسته‌های منتظر برای ارسال در شبکه برای نسبت‌های مختلف ورودی در شکل (۱۰-۸) نشان داده‌شده است. مقادیر لازم برای هر اجرا از مقادیر اندازه‌گیری شده برای $T(i)$ استفاده کردند و طول‌های صف برای هر زمان ورودی از طریق ترکیب معادلات (۱۰-۶)، (۱۰-۷) و (۱۰-۹) به دست آمدند. به طور مشابه، متوسط زمان پاسخگویی با استفاده از معادلات (۱۰-۵)، (۱۰-۷) و (۱۰-۱۱) محاسبه شدند و در شکل (۱۰-۹) ترسیم شدند.

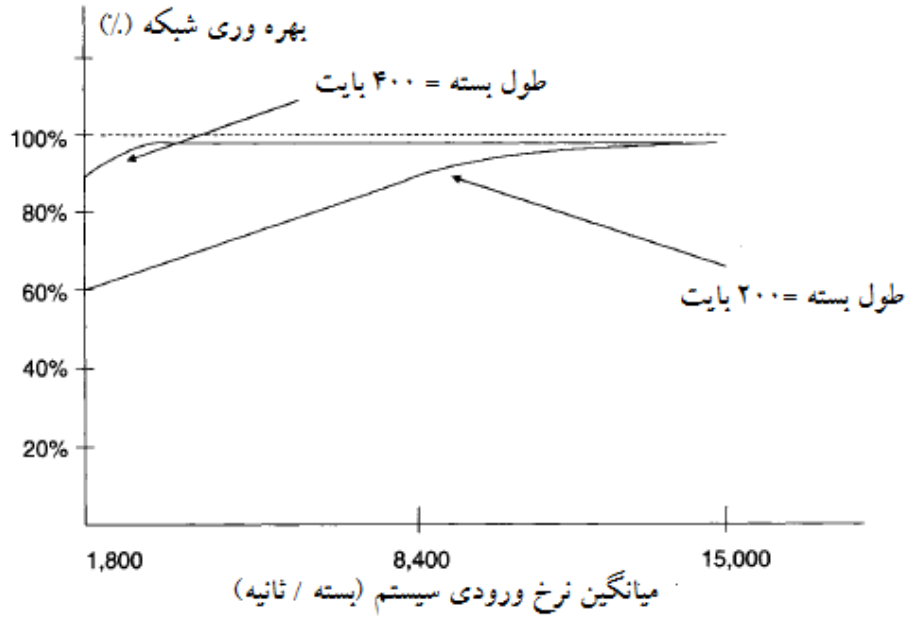


شکل (۱۰-۸): متوسط طول صف

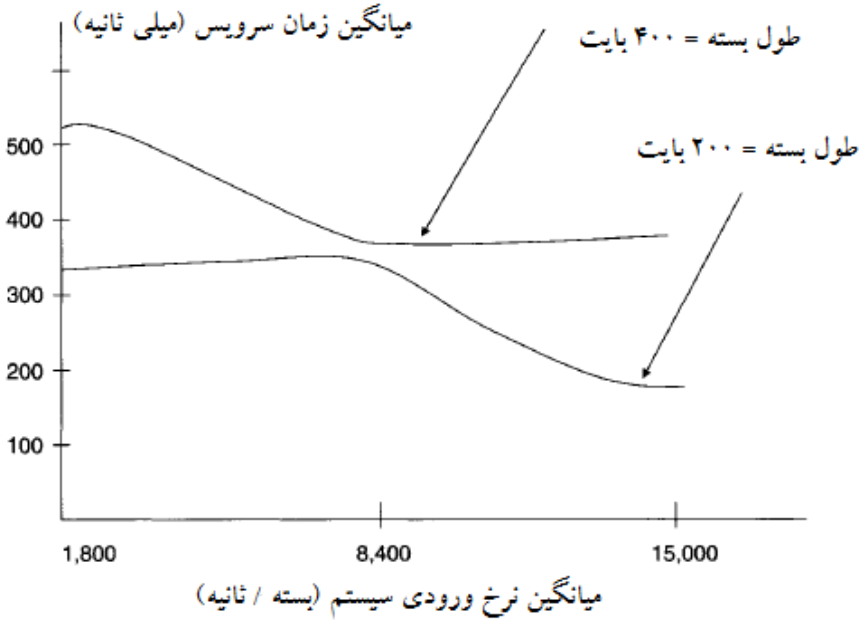


شکل (۱۰-۹): میانگین زمان پاسخگویی

منحنی بهره وری که در شکل (۱۰-۱۰) نشان داده‌شده است، درصد پهنای باند داده موجود را نشان می‌دهد که برای انتقال بسته‌های پیام مورداستفاده قرار می‌گیرد. برای این نمودار، می‌بینیم شبکه مورد نظر ما به حالت اشباع می‌رود (یعنی ۱۰۰ درصد بهره وری) و نسبت ورودی و اندازه بسته‌ها نیز نشان داده‌شده است.

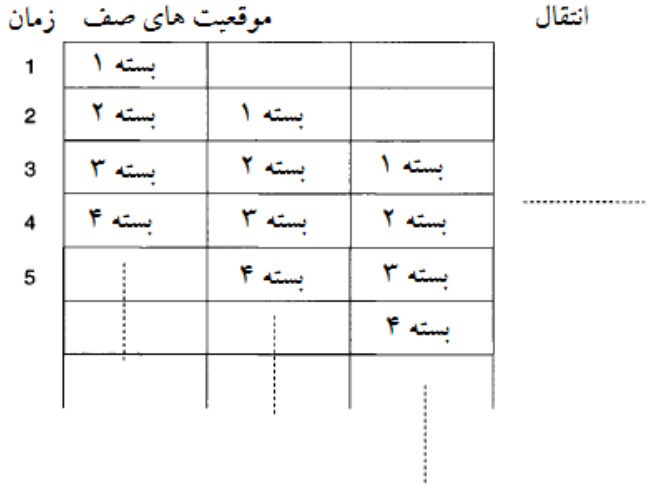


شکل (۱۰-۱۰): بهره وری شبکه

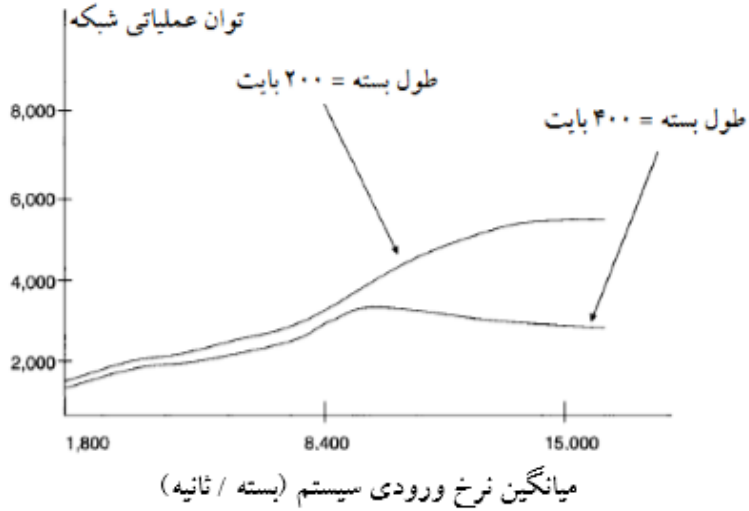


شکل (۱۰-۱۱): میانگین زمان سرویس سیستم.

شکل (۱۰-۱۱): تأثیر یک نسبت ورودی افزایشی را روی زمان سرویس نشان می‌دهد. در این مورد، زمان سرویس را به‌صورت زمان شلوغی سیستم در تکمیل عملیات معرفی می‌کنیم که در آن زمان شلوغی مربوط به زمانی است که یک بسته در صف و زمانی که یک بسته در انتقال می‌گذراند، در نظر می‌گیریم. بااین‌حال، وقتی که سیستم اشباع شود، همیشه بسته‌ای برای انتقال وجود دارد و بنابراین زمان کاهش صف توسط این واقعیت مخفی می‌ماند. شکل (۱۰-۱۲) این تأثیر را نشان می‌دهد که به خط لوله نیز معروف است.

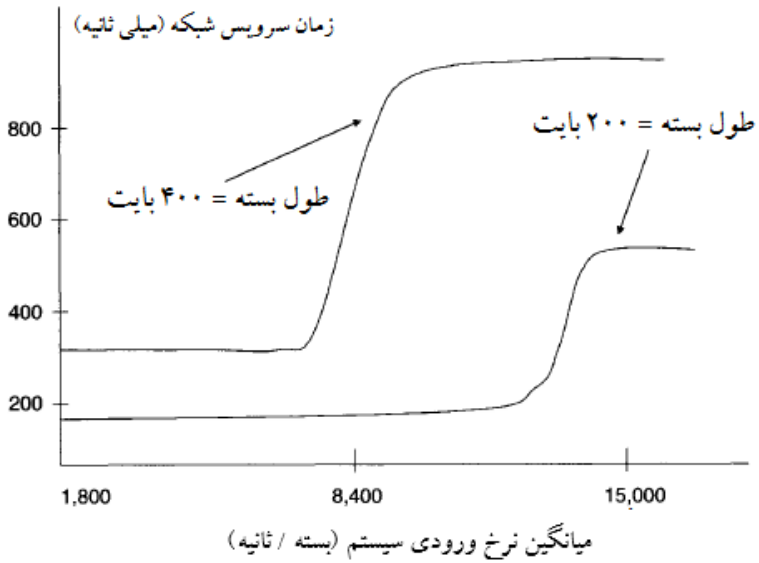


شکل (۱۰-۱۲): تأثیر خط لوله روی زمان کاهش صف



شکل (۱۰-۱۳): توان عملیاتی شبکه

در شکل (۱۰-۱۳)، توان عملیاتی شبکه نسبت به نسبت ورودی ترسیم‌شده است. نتایج نشان می‌دهند که بعد از اشباع، توان شبکه برای این نوع شبکه ثابت باقی می‌ماند. این یکی از ویژگی‌های مهم برخی سیستم‌ها است، به ویژه از آنجایی که برخی پروتکل‌های شبکه تحت بار شدید سیستم تخریب می‌شوند.



شکل (۱۰-۱۴): زمان سرویس شبکه

گراف آخر (شکل ۱۰-۱۴)، زمانی را نشان می‌دهد که یک پیام از شبکه عبور می‌کند. این زمان شامل زمان انتظار صف نیست. هنگامی که سیستم به شدت بارگذاری شده باشد، این زمان شامل زمان عبور توکن در شبکه نیز می‌شود. تحت بار شدید، این زمان شامل تأخیر مربوط به انتظار بسته‌های پیام در گره‌های دیگر است.

این مثال دو منظور دارد: نمایش مفید بودن مدل‌سازی سخت‌افزاری در برخی موارد خاص و نشان دادن کاربرد برخی تکنیک‌های مربوط به تحلیل عملیاتی که در فصل ۷ بیان شد. برای مدل‌سازی سخت‌افزاری، فرضیات مربوط به رفتار شبکه را می‌توان بررسی نمود. تحلیل عملیاتی به ما این امکان را می‌دهد تا مقادیر مورد نظری که به‌صورت مستقیم قابل اندازه گیری نیستند یا اندازه گیری آن‌ها خیلی سخت است را بدون نیاز به عملیات واقعی خود شبکه، محاسبه نماییم.

۱۰-۳ روش‌های کلی استخراج داده

در بخش قبل سیستمی را آزمایش کردیم که برای آزمایش مفاهیم سیستم، قبل از ایجاد سیستم نهایی، مورداستفاده قرار می‌گیرد. ما اغلب با سیستم موجود را تجزیه و تحلیل می‌کنیم. برای این کار تحلیل گر سیستم‌های کامپیوتری باید روش‌هایی را برای استخراج اطلاعات از یک سیستم در حال اجرا توسعه دهد و آزمایش‌هایی روی سیستم موجود انجام دهد.

سه روش برای استخراج اطلاعات از سیستم موجود وجود دارد: نظارت سخت‌افزاری، نظارت نرم‌افزاری و نرم‌افزار حسابداری (یعنی سیستم‌عامل). معیارهای لازم برای تحلیل عملکرد معمولاً با کمک ناظرهای سخت‌افزاری و نرم‌افزاری استخراج می‌شوند که به طور خاص برای معیارهای مورد نظر راه‌اندازی می‌شوند. با توجه به نوع پارامتر، می‌توانیم آن‌ها را به طور مستقیم اندازه‌گیری کنیم یا ممکن است نیاز به اندازه‌هایی از معیارهای واسط داشته باشیم.

بیشتر سیستم‌های کامپیوتری، حتی کامپیوتر شخصی شما، ابزارهایی برای تعیین کاربرد منبع و انواع مختلف معیارهای مفید فراهم می‌کنند. اگر سیستم، یک سیستم با اشتراک زمانی باشد، می‌توانیم تعیین کنیم که برای یک فرآیند چه میزان زمان CPU گرفته می‌شود، چقدر حافظه مصرف می‌شود و احتمالاً چه میزان زمان برای پردازش I/O سپری‌شده است. اطلاعاتی مثل تعداد کاربران متصل به سیستم، تعداد دسترسی‌های I/O انجام‌شده توسط یک کاربر، خطاهای صفحه در حافظه و زمان فعال برای یک فرآیند را نیز می‌توان به دست آورد.

مشکل مربوط به نرم‌افزاری که برای اهداف حسابداری سیستم در نظر گرفته شده این است که ممکن است اطلاعات مربوط به مؤلفه‌های نرم‌افزاری سیستم مثل سیستم‌عامل را نداشته باشد. در بسیاری سیستم ها، زمان سپری‌شده توسط سیستم‌عامل ممکن است واقعاً نشان‌دهنده استفاده از سهم منبع شیر باشد. برای مثال، بیشتر PC ها بخش زیادی از زمان خود را در حالت بیکاری سیستم‌عامل سپری می‌کنند. به دلیل وجود این محدودیت، بیشتر بسته‌های حسابداری سیستم نمی‌توانند به ما کمک کنند تا عملکرد سیستم را ارزیابی نماییم. برخی از سیستم‌عامل‌های جدیدتر به کاربران ابزارهایی برای تعیین میزان بهره وری سیستم‌عامل می‌دهند. برای مثال، بخش task manager محصول مایکروسافت قابلیت‌های نسبتاً خوبی برای نظارت روی استفاده از منبع و عملکرد سیستم دارند. بااین‌حال، این بسته بیشتر شبیه ناظر نرم افزاری است تا نرم افزار حسابرسی.

مانیتورهای نرم افزاری از مجموعه ای از قطعات تعبیه شده در سیستم‌عامل یا برنامه کاربردی استفاده می‌کنند تا اطلاعات مربوط به عملکرد را در یک جا جمع کنند. ناظر نرم افزاری باید در حداقل قرار

گیرد تا تأثیر آن روی سیستم درحال اجرا در کمترین حد ممکن باشد. حالات اصلی برای ایجاد مانیتورهای نرم افزاری از رویکرد نمونه برداری یا رویداد استفاده می‌کند. در رویکرد نمونه برداری، قطعات کد مانیتور به‌صورت دوره ای فراخوانی می‌شوند تا وضعیت سیستم را تعیین نمایند. هر یک از قطعات کد دارای یک قابلیت خاص دارد. یکی ممکن است کاربرد حافظه و دیگری ممکن است CPU یا I/O را آزمایش نماید. با استفاده از اطلاعات به دست آمده، ناظر می‌تواند به مرور زمان یک تصویر نسبتاً دقیق از رفتار سیستم‌ها ایجاد نماید. مشکل مربوط به نمونه برداری این است که برخی پارامترها یا رویدادهای مورد نظر ممکن است به دلیل قرار نگرفتن دوره نمونه برداری در مسیر خود، از دست بروند. بااین‌حال، مزیت این رویکرد ساده بودن و کم بودن اثر آن روی سیستم‌ها است. با تغییر تعداد نمونه برداری، بار روی سیستم را می‌توان در صورت نیاز کم و زیاد کرد. مهم‌ترین مشکل طراحی هنگام توسعه و نمونه برداری، تعیین اندازه مناسب نمونه برداری و نقاط ارزیابی در سیستم است.

طراح در رویکرد طراحی رویداد برای یک ناظر نرم افزاری باید درک درستی از رویدادهای موجود در سیستم داشته باشد که به وسیله آن بتواند نظارت را به‌صورت همزمان انجام دهد. برای مثال، سوییچینگ کار CPU یک رویداد مهم است، برای اینکه تخصیص و عدم تخصیص I/O و حافظه دارد. برای رویدادهایی که باید تحت نظارت قرار گیرند، کد سیستم‌عامل‌ها باید تغییر کند. این کد باید به نحوی تنظیم شود که در زمان بروز رویداد، اطلاعات موردنیاز را بتوان از سیستم‌عامل استخراج نمود و روی یک فایل ذخیره کرد. برای مثال، ممکن است بخواهیم ثبت کنیم که کدام فرآیند حافظه به خود تخصیص داده یا از حافظه جدا شده است، کدام فرآیند CPU را در اختیار دارد و یا زمان‌های مربوط

به این رویدادها را بخواهیم ضبط نماییم. فایل‌های رویداد را می‌توان بعداً پردازش نمود تا معیارهای عملکرد سیستم استخراج شوند. اگر بتوانیم تمام رویدادهای مورد نظر را تعریف کنیم و کد سیستم‌عامل‌ها را برای استخراج اطلاعات مربوط به آن‌ها، تحت کنترل درآوریم، درنتیجه می‌توانیم یک مدل نسبتاً خوب از سیستم تحت مطالعه ایجاد کنیم. با استفاده از آثار مربوط به یک رویداد، می‌توانیم طول مدت هر سرویس CPU و منابع مصرف شده در حین هر یک از این سیکل‌ها را تعیین کنیم.

اگر به کد سیستم‌عامل دسترسی نداشته باشیم درنتیجه نمی‌توان از این روش استفاده نمود. ابتدا باید بتوانیم این کد را بگیریم و زمان بندی این کد را استخراج کنیم. اگر این برنامه کاربردی به خوبی طراحی شده باشد، این حداقل یک معیار از دوره زمانی فراهم می‌کند که طی آن یک برنامه کاربردی منبعی را نگه می‌دارد و از آن می‌توان به‌عنوان یک ابزار برای ارزیابی عملکرد سیستم استفاده نمود. مشکل تمام این رویکردها این است که روی سیستم‌عامل تأثیر می‌گذارند. نرم افزار نمونه برداری از منابع استفاده می‌کند و باعث ایجاد وقفه‌هایی می‌شود که روی اندازه گیری عملکرد تأثیر می‌گذارد و احتمالاً باعث می‌شود آن‌ها مقادیر نادرستی نشان دهند. مطالعات نشان می‌دهند که یک ناظر نرم افزاری می‌تواند به اندازه ۲۰٪ از منابع سیستم‌ها استفاده کند و عملکرد سیستم را زیر سؤال ببرد. اگر نوع رویدادها را با دقت انتخاب کنیم و کد افزوده به حداقل موردنیاز برای ضبط اطلاعات را محدود نماییم، میزان سربار تا حدود ۵٪ افت می‌کند. این معاوضه نسب به سربار نرم افزار معیار کاربر پسندتر است.

علاوه بر مشکل مربوط به تأثیر عملیات سیستمی، مانیتورهای نرم‌افزاری نیز مشکلاتی به همراه دارند. روش ردیابی مجموعه داده، حجم زیادی از اطلاعات ایجاد می‌کند که ذخیره‌سازی و پردازش آن‌ها

باعث می‌شود میزان استفاده کارآمد از آن‌ها بسیار دشوار گردد. ناظران نرم‌افزاری باید طوری تنظیم شوند که با معماری سیستم متناسب شوند و یکی از انواع پیاده سازی‌ها شوند. به دلیل این محدودیت، هیچ معماری عمومی از ناظر نرم افزاری موجود وجود ندارد. علاوه بر این، پیاده سازی ناظران نرم افزاری به تجربه زیادی در زمینه کدنویسی سیستم‌عامل‌ها دارد که البته جزو توانایی‌های روزمره بیشتر برنامه نویسان نمی‌باشد. به دلیل این محدودیت، معمولاً از این تکنیک استفاده نمی‌شود و ما مجبوریم که از قابلیت‌های نظارتی یک سیستم‌عامل استفاده کنیم.

ناظران سخت‌افزاری امکان دیگری برای دسترسی به اطلاعات عملکرد فراهم می‌کنند. یک ناظر سخت‌افزاری از مجموعه ای از سخت‌افزارهای دیجیتالی متصل به سیستم تحت بررسی تشکیل می‌شود. برای مثال، یک اسیلوسکوپ یک سخت‌افزار همه منظوره است که بر فعالیت‌های سخت‌افزاری نظارت دارد. ناظران سخت‌افزاری ممکن است چند گیت ساده باشند یا کل سیستم کامپیوتری با تمام وسایل جانبی آن. ناظران سخت‌افزاری را می‌توان به راحتی از بازار تهیه کرد.

ناظران سخت‌افزاری باید به سیستم ما متصل شوند تا اطلاعات را که به‌صورت سیگنال هستند، جمع آوری کنند. مواردی را به ناظر سخت‌افزاری اضافه می‌کنیم تا نقاط آزمایشی یا ردیابی ما را نشان دهد. نقاط آزمایشی در سیستم کامپیوتر تحت آزمایش قرار می‌گیرد و برای اندازه گیری قابل دسترسی است. برای مثال، ممکن است بخواهیم خطوط وقفه CPU را ردیابی نماییم، بنابراین می‌توانیم تعیین کنیم که چه زمانی کار سوییچ شروع شود. ممکن است بخواهیم که مکان‌های مخصوص حافظه را برای آزمایش زمان عبور مقادیر از برخی نقاط یا زمان تغییر آن‌ها را آزمایش کنیم. با اتصال ردیاب‌های آزمایش ناظر

به این نقاط، رفتار سیستم‌ها را می‌توانیم در برخی قاب‌های زمانی مشاهده کنیم. این می‌توانیم از ترکیب چند نقطه آزمایشی برای هماهنگ‌سازی زمان استخراج سیگنال بر اساس اندازه گیری یا تشخیص برخی نقاط آزمایشی دیگر استفاده کنیم. این اندازه گیری‌ها معمولاً بدون درگیر شدن سربار اضافه به سیستم انجام می‌شوند، که یکی از مزایای رویکرد نظارت نرم افزاری است.

یکی از مشکلات ناظرهای سخت‌افزاری این است که چه زمان باید از نقاط آزمون استفاده کرد و کجا باید آن‌ها را قرار داد. برای مثال، کجا باید متوجه مشغول بودن CPU شویم؟ چگونه بفهمیم که با یک سیستم‌عامل مشغول است یا با یک برنامه کاربردی؟ بیشتر فروشندگان سیستم‌ها اجزاء و سیستم‌های خود را با نقاط نظارتی آماده به کار ایجاد کرده اند تا به اشکال زدایی و تعمیر آن‌ها کمک کنند. این باعث می‌شود که به‌سادگی متوجه شویم که نقاط آزمون خود را در کجا قرار دهیم. اگر این نقاط آزمون در دسترس نباشند، درنتیجه نظارت سخت‌افزار به سختی انجام می‌شود.

دستگاه‌های ناظر باید بتوانند اطلاعات مربوط به نقاط آزمون را جمع آوری کنند و این اطلاعات را برای پردازش و تحلیل آتی حفظ کنند. این کار ضروری است به نحوی که می‌توانیم استفاده از مؤلفه‌های آزمونی موجود در یک سیستم را تعیین نماییم: تعداد واحدهای اندازه‌گیری شده که یک نقطه را در برخی قاب‌های زمانی عبور می‌دهند – برای مثال، تعداد کارهایی که به CPU عرضه می‌شود تا در یک دوره زمانی انجام دهد و اینکه چه درصدی از زمان طول می‌کشد که CPU مشغول یا بیکار باشد. محدودیت مربوط به نظارت سخت‌افزاری اینجاست که تنها می‌توانیم سیگنال‌های سخت‌افزاری و احتمالاً محتوای ثبات‌ها یا حافظه را اندازه‌گیری کنیم (در صورت اجازه). ما معمولاً نمی‌دانیم که در

نقطه ارزیابی ما، سیستم‌عامل دقیقاً چه‌کاری انجام می‌دهد. به دلیل این محدودیت، معمولاً از ناظران سخت‌افزاری همراه با برخی از نرم‌افزارهای ردیاب رویداد استفاده می‌شود تا امکان تفسیر بیشتر عملیات سخت‌افزاری فراهم شود.

۴-۱۰ تراکم کار[۷۷] مدل و پلتفرم

عبارت تراکم کار میزان بارکاری موجود در یک سیستم واقعی را نشان می‌دهد (که معمولاً روی یک سیستم کامپیوتری اندازه‌گیری یا مشاهده می‌شود، درحالی‌که عملیات معمولی اجرا می‌کند)، درحالی‌که عبارت تراکم کاری تست یا مدل نشان دهنده بار کاری سیستم کامپیوتری است که روی یک سیستم برای مطالعه عملکرد ایجاد و اعمال می‌گردد (معمولاً با استفاده از ویژگی‌های یک تراکم کار واقعی هماهنگ می‌گردد). برای بیشتر پروژه‌های مدل‌سازی، استفاده از تراکم کار مصنوعی معقول تر است، برای اینکه ما می‌توانیم بار کاری اعمال شده در آزمایش ها را کنترل کنیم. با کنترل بار کاری اعمال شده در یک سیستم کامپیوتری مورد تحلیل، می‌توانیم خروجی آزمایش را پیش بینی کنیم و یا در آزمایش مؤلفه‌های خاصی از سیستم را آزمایش نماییم. علاوه بر این دلیل، تراکم بار کاری مصنوعی احتمالاً شامل اطلاعات واقعی نمی‌باشد که ممکن است برای سیستم تحت بررسی ما حساس یا باارزش باشد و ضرر و زیان آن نیز قابل توجه خواهد بود. هنگامی که تراکم کار مصنوعی ایجاد شود، از آن می‌توان برای مطالعه سیستم‌های دیگر استفاده نمود. برای مطالعه سیستم‌های پایگاه داده یک مثال در

---

[۷۷] - workload

زمینه تراکم بار کنسرسیوم پردازش انتقال (TPC) ایجاد شده است. این تراکم بار TCP توسط فروشندگان و مشتریان مورداستفاده قرار می‌گیرد تا انواع مختلف سیستم‌های پایگاه داده مطالعه شوند و معلوم شود که کدام یک برای برنامه‌های کاربردی مختلف مناسب تر است. برخی از این تراکم بار برای کاوش داده یا برای پایگاه‌های داده توزیع‌شده در نظر گرفته شده و بقیه مخصوص برنامه‌های کاربردی است.

برای مطالعه در زمینه معماری‌های کامپیوتری، انواع مختلف تراکم بار دستوری ایجاد شده است. این‌ها روی عملیات سطح پایین متمرکز شده اند و شامل ترکیبی از بار کاری، ذخیره سازی، مقایسه، شاخه ها، افزایش، کاهش، عملیات نقطه اعشاری، ضرب، تقسیم، عملیات جابجایی، عملیات منطقی و عملیات ثبات می‌شوند. این تراکم‌های بار ترکیب دستوری برای معماری‌های خاصی مثل PC ها استاندارد شده اند.

تراکم بارهای دیگر روی عملیات سطح پایین تمرکز ندارد بلکه تمایل دارند مفاهیم معماری کلان تری را مورد آزمایش قرار دهند. این مفاهیم با استفاده از زبان‌هایی با ترتیب بالا توسعه می‌یابند و برای آزمایش چیزهایی مثل انتقال فایل، سوییچینگ کار، خط‌مشی‌های مدیریت حافظه و مؤلفه‌های سیستم‌عاملی دیگر طراحی شده اند. برخی معیارهای معروف عبارتند از: معیار TPC که پیش از این برای آزمایش سیستم‌های پایگاه داده توصیف شده است، معیار Sieve که برای آزمایش PC ها و ریزپردازنده‌ها مورداستفاده قرار می‌گیرد، تابع Ackerman که برای آزمایش مکانیزم فراخوان روال در سیستم‌عامل‌ها به کار می‌رود، هسته وتستون که برای آزمایش عملیات سطح پایین اسمبلی توسعه‌یافته

است، بسته Linpack که برای آزمایش عملیات اعشاری ایجاد شده است، معیار Drystone برای آزمایش عملیات عدد صحیح سطح پایین و مجموعه معیار نمونه (Spec) که برای ارزیابی برنامه‌های کاربردی مهندسی در سیستم‌عامل ایجاد شده است (مثل، کامپایل، طراح الکترونیک، شبیه سازی مدار VLSI و کنترل ریاضیات پیچیده مثل ضرب ماتریس).

با توجه به اینکه این تراکم بارها وجود دارند، مدل سازها باید تعیین کنند که از چه روشی برای ساخت تراکم بار خود برای یک پروژه مدل‌سازی استفاده نمایند. ۴ ملاحظه اصلی وجود دارد که در هنگام انتخاب یک تراکم بار برای یک پروژه کاربردی هستند. آن‌ها سیستم‌های کامپیوتری هستند که توسط تراکم بار، سطح کلی قابل اعمال، نزدیکی به بار واقعی و بهنگام بودن اعمال می‌شوند.

مهم‌ترین بخش انتخاب تراکم بار مربوط به تعیین سرویس‌هایی است که قصد آزمایش آن‌ها را داریم. ایجاد فهرستی از این سرویس‌ها ممکن است بسیار دشوار وقت‌گیر باشد اما باید به‌موقع باشد. ابتدا سیستم تحت آزمایش باید انتخاب گردد. این مجموعه کاملی از مؤلفه‌هایی را نشان می‌دهد که مجموعه تحت مطالعه را تشکیل می‌دهد. بیشتر اوقات ممکن است برای مقایسه روی برخی اجزای واحد یا مجموعه‌های کوچکی متمرکز شویم که به آن مؤلفه‌های تحت مطالعه می‌گوییم. برای مثال، تیم طراحی یک سیستم‌عامل ممکن است به الگوریتم‌های زمان‌بندی روند مختلفی برای بررسی عملکرد سیستم‌عامل‌ها گرایش داشته باشند. تعیین سیستم و مؤلفه‌های آن، گام خیلی مهمی در توسعه تراکم بار است و نباید کم‌اهمیت جلوه داده شود. با یک مثال مفهوم سرویس را نشان خواهیم داد: یک سیستم ذخیره‌سازی صفحه‌بندی پشتیبان آفلاین را با استفاده از آرایه‌های درایو مقایسه می‌کنیم (یعنی، مانند نمونه‌ای که در

یک سیستم فرعی پایگاه داده پیدا می‌کنیم). این سیستم از چند سیستم داده دیسک تشکیل می‌شود که هر یک چند درایو دیسک دارد. درایوهای دیسک سیستم‌های فرعی خواندن و نوشتن را از هم جدا می‌کند. هر سیستم فرعی از هدهای مغناطیسی ثابت برای عملیات استفاده می‌کند. اگر این معماری را از بالاترین سطح مشخص کنیم و به پایین ترین سطح برسیم، سرویس ها، عوامل، معیارها و تراکم بار را به‌صورت زیر تعریف می‌نماییم:

۱. سیستم پشتیبان

- سرویس ها: صفحات پشتیبان، صفحاتی با پشتیبان تغییریافته، صفحات بازیابی، صفحات پشتیبان فهرست.
- عوامل: اندازه صفحه، فرآیند پس زمینه یا انشعاب، پشتیبانی کامل یا افزایشی.
- معیارها: زمان پشتیبان، زمان بازیابی.
- تراکم بار: یک سیستم پایگاه داده با صفحات آماری که باید پشتیبانی شوند و دارای میزان ورودی متفاوتی هستند.

۲. سیستم داده دیسک

- سرویس ها: خواندن/نوشتن از دیسک
- عوامل: نوع دیسک درایو
- معیارها: سرعت، قابلیت اطمینان، زمان ما بین خطاها
- تراکم بار: برنامه مصنوعی که درخواست‌های I/O دیسک را تولید می‌کند.

۳. درایوهای دیسک

- سرویس ها: خواندن رکورد، نوشتن رکورد، پیداکردن رکورد
- عوامل: ظرفیت درایو دیسک، تعداد ترک ها، تعداد استوانه ها، تعداد هدهای خواندن/نوشتن
- معیارها: زمان پیدا کردن رکورد، زمان خواندن رکورد، زمان نوشتن رکورد، نسبت اتلاف داده، تعداد درخواست‌ها در واحد زمان
- تراکم بار: درخواست عملیات واقعی تولید برنامه مصنوعی برای درایوهای دیسک

٤. سیستم فرعی خواندن/نوشتن

- سرویس ها: خواندن دادن، نوشتن داده
- عوامل: تکنیک رمزنگاری داده، فناوری پیاده سازی
- معیارها: پهنای باند $\mathrm{I/O}$، تراکم رسانه
- تراکم بار: جریان‌های داده خواندن و نوشتن با الگوهای واریانس

۵. هدهای خواندن و نوشتن

- سرویس ها: سیگنان خواندن و سیگنال نوشتن
- عوامل: ترکیب، فاصله گذاری هد، اندازه شکاف رکورد
- معیارها: توان حوزه مغناطیسی، پسماند.
- تراکم بار: خواندن و نوشتن انواع مختلف قدرت توان، دیسک‌هایی که با سرعت‌های چرخشی مختلفی حرکت می کنند.

بعد از اینکه بررسی مشخصات سیستم و مؤلفه‌های مورد نظر را به پایان بردیم، باید سطح جزئیات موردنیاز برای تولید و ثبت درخواست‌های مربوط به سرویس‌های تعریف‌شده را تعیین کنیم. یک توصیف از تراکم بار می‌تواند به اندازه ایجاد تعریف‌ها برای رویدادهای موجود در سیستم مفصل باشد یا می‌تواند یک تعمیم از این بار محسوب شود. برخی از احتمالات مربوط به جزئیات می‌تواند شامل میانگین تقاضا برای منبع، پر تکرارترین درخواست ها، تکرر انواع درخواست (یعنی، ۲۵٪ خواندن و ۷۵ درصد نوشتن)، یک دنباله با برچسب زمانی از درخواست‌های خاص یا توزیعی از تقاضاهای منبع باشد. پروژه‌های مدل‌سازی معمولی با استفاده از یک نوع مفهوم از پر تکرارترین درخواست سرویس شروع می‌شوند. برای مثال، در سیستم پردازش معاملات، ممکن است از یک معیار بدهی – اعتبار از معیارهای TPC استفاده کنیم. این انتخاب در صورتی معتبر است که یک سرویس بیش از بقیه درخواست شود. یک روش جایگزین انتخاب سرویس‌های خاص، ویژگی‌های آن‌ها و فراوانی است. بسته Linpack چنین تراکم باری دارد. این بسته عملیات مخصوص کامپیوتر را به شیوه تعریف‌شده انتخاب می‌کند تا مؤلفه‌های مخصوص سیستم را آزمایش کند. روش دیگر، ایجاد یک رکورد با برچسب زمانی است، که در آن هر رکورد نشان دهنده یک درخواست خاص برای یک سرویس با امکان دسترسی واقعی است (چنین تعریفی را می‌توان با ردیابی تمام فعالیت‌های یک سیستم موجود ایجاد نمود). در بیشتر موارد این نوع تراکم بار آن‌قدر سخت ایجاد می‌شود تا برای استفاده در تمام پروژه‌های مدل‌سازی پیچیده اعتبار داشته باشد. رویکرد تقاضای انبوه منبع مشابه رویکردی است ما انتظار داریم در یک مدل تحلیلی ببینیم. ما به دنبال این هستیم که هر درخواست را برای سرویس ها، میانگین‌ها یا توزیع‌ها مشخص کنیم. برای

مثال، یک درخواست ممکن است به ۵۰ واحد از یک منبع خاص و ۲۵ واحد از منبع دیگر نیاز داشته باشد و به ازای هر ۱۰۰۰ واحد زمانی این منابع را درخواست نماید.

مهم نیست که از کدام یک از این روش‌ها استفاده کنیم، مدل ساز باید تعیین نماید که آیا بار انتخابی نماینده بار سیستم واقعی است یا خیر. معمولاً دوست داریم تعیین کنیم که آیا بار درخواست سرویس مشابه ویژگی‌های ورودی، تقاضا برای منبع و تقاضا برای بهره برداری از منابع به‌عنوان بار واقعی است یا خیر.

در نهایت، یک تراکم بار پیشرفته باید به درستی تغییرات موجود در هنگام استفاده از الگوها را به‌موقع مدل‌سازی نماید. برای مثال، معیارهای TPC برای تأمین نیازهای مربوط به تغییر طرح و کاربرد سیستم‌های پایگاه داده، باید تکامل یابند. تراکم بار اصلی TPC برای مشکل پایگاه داده "بانکداری" در نظر گرفته شده است. یعنی، آن‌ها به دنبال تأمین بار تراکنش برای مشخصات پایگاه داده رابطه‌ای، ساده و ساختاریافته بودند. آن‌ها به‌صورت همگرا ثبت شدند و بعدی فراتر از مدل ساده رابطه‌ای روز نداشتند. درحال حاضر این موارد تکامل یافته اند و شامل معیارهایی برای پایگاه‌های داده رابطه ای شی ای جدید و برای انبارهای داده و سیستم‌های داده کاوی شده اند. ملاحظات مهم دیگر مربوط به تراکم بار شامل قابلیت تکرار، تاخیر مؤلفه‌های خارجی و طبقه بندی بار است. قابلیت تکرار به دنبال توانایی تراکم بار است تا بتواند با اطمینان با سربار اندکی تولید شود. تأثیر مؤلفه‌های خارجی تأثیرات روی سیستم تحت مطالعه را توسط مؤلفه‌های غیرضروری ضبط و مشخص می‌کند. درنهایت، اگر در این

مطالعه بخواهیم سیستمی را تحت بهترین شرایط یا بدترین شرایط آزمایش کنیم، طبقه‌بندی بار نیز از محبوبیت زیادی برخوردار است.

۱۰-۵ طراحی آزمایشی

هنگامی‌که یک پلتفرم برای مطالعه یک سیستم کامپیوتری و یک تراکم بار برای بارگذاری پلتفرم با آن داشته باشیم، باید آزمایشی انجام دهیم که به ما به‌عنوان مدل‌ساز کمک کند محدودیت های مربوط به عملکرد چیزی که روی آن متمرکز شده ایم را کشف کنیم. یک طرح آزمایشی درست به ما کمک می‌کند تا بیشترین اطلاعات تحلیلی مربوط به حداقل تعداد اجراهای آزمایشی موردنیاز را به دست آوریم. برای معنادار شدن این بحث باید برخی اصطلاحات را معرفی نماییم. یک متغیر عملکرد یک خروجی ارزیابی شده برای یک آزمایش برای یک مؤلفه واحد، یک فرآیند یا احتمالاً کل یک سیستم است. یک عامل متغیری است که ممکن است روی متغیرهای مربوط به عملکرد تأثیر داشته باشد و معمولاً مواردی را نشان می‌دهد که می‌توانند در حین آزمایش متفاوت باشند. مراحل، مقادیری هستند که یک عامل می‌تواند در حین اجرای آزمایشی از آن‌ها استفاده نماید. برای مثال، حافظه یک CPU ممکن است از حداقل تا حداکثر مقدار در برخی مقادیر متمایز از مراحل گسسته تنظیم شود. هر یک از این مراحل برای عامل تحت مطالعه ما نشان دهنده یک مقدار می‌باشد. لازم نیست که عوامل همیشه مهم باشند. معمولاً، آزمایشاتی که روی سیستم‌های کامپیوتری انجام می‌شوند، عوامل متعددی دارند، که برخی از آن‌ها خیلی مهم هستند (مثل سرعت CPU) و برخی از اهمیت جانبی برخوردارند (مثل سرعت

نهایی). این آزمایش‌ها می‌توانند تکرار شوند و سپس به‌عنوان یک مرجع مورداستفاده قرار گیرند. یک مطالعه کامل از عملکرد برای یک سیستم خاص شامل تعدادی از آزمایش‌های گسسته است – هنگامی‌که این آزمایش‌ها باهم انجام شوند، طرح آزمایشی را ایجاد می‌کنند. عوامل ممکن است با یکدیگر مرتبط باشند و باید به‌گونه‌ای تعریف شوند که این ارتباط را نشان دهند.

طراحی آزمایشی به شیوه‌های مواریانسی انجام می‌شود. سه طراحی معمول عبارتند از طراحی ساده، فاکتوریل کسری و فاکتوریل کامل. در طراحی ساده، ما با یک پیکربندی ثابت شروع می‌کنیم و در هر لحظه یک عامل را تغییر می‌دهیم تا نشان دهیم که چگونه روی عملکرد تأثیر می‌گذارد. برای مثال، هنگام ارزیابی عملکرد یک مؤلفه مدیریت حافظه مجازی، ممکن است بخواهیم با تغییر اندازه حافظه اولیه موجود روی طراحی سیستم‌ها مطالعه کنیم. با اجرای آزمایش‌های مختلف، هر آزمایشی تحت هر شرایطی ثابت است به‌جز اندازه حافظه، ممکن است بتوانیم برخی اطلاعات مفید مربوط به عملیات سیستم‌های مدیریت حافظه مجازی را تعیین نماییم. تعداد کل آزمایش‌های موردنیاز برای یک طرح ساده برابر است با مجموع تعداد آزمایش‌ها برای هر عامل. برای مثال، اگر بخواهیم درباره سیستم مدیریت حافظه با سه اندازه حافظه مختلف مطالعه کنیم، باید از سه CPU مجزا و سه درایو دیسک مختلف و سه تراکم بار به‌صورت زیر استفاده کنیم:

$$V = (\text{اندازه حافظه ۳}) + (\text{انواع پردازنده ۳}) + (\text{مدل های درایو دیسک ۳}) + (\text{حجم کاری ۳}) = ۱۲ \quad (۱۰-۱۵)$$

تعداد کل آزمایش‌ها مشخص می‌شود. این شکل از آزمایش به ما اطلاعاتی می‌دهد اما ممکن است نشان ندهد که چگونه عناصر مختلف با یکدیگر در تراکنش هستند. برای نشان دادن تراکنش این عوامل، باید آزمایش‌های فاکتوریل کسری یا کامل را اجرا کنیم. در آزمایش‌ها فاکتوریل کسری، ممکن است بخواهیم که تنها چند عامل را نسبت به بقیه مورد آزمایش قرار دهیم. در این مورد، به آزمایش‌های بیشتری نیاز است. در مثال ما، اگر به نحوه تراکنش CPU ها با حافظه علاقه‌مند باشیم، باید ترکیب آن‌ها را نسبت به یکدیگر آزمایش کنیم. این کار به آزمایش‌های بیشتری نیاز دارد. در مثال ساده ما، اکنون به تعداد زیر آزمایش نیاز داریم:

$$V = (\text{مدل های درایو دیسک ۳}) + (\text{انواع پردازنده ۳}) + (\text{اندازه حافظه ۳}) + (\text{حجم کاری ۳}) = ۵۴ \quad (۱۰-۱۶)$$

برای یک طرح آزمایشی فاکتوریل کامل، تمام عوامل را با نسبت به یکدیگر تغییر می‌دهیم. در این مورد، باید عمل زیر را اجرا کنیم:

$$N = \left(\text{مدل های درایو دیسک ۳}\right) + \left(\text{انواع پردازنده ۳}\right) + \left(\text{اندازه حافظه ۳}\right) + \left(\text{حجم کاری ۳}\right) = ۸۱ \quad (۱۰-۱۷)$$

عملیات بالا آزمایش‌های خاصی هستند که نشان می‌دهند چگونه این عوامل یکدیگر را در کل دامنه مقادیر خود تحت تأثیر قرار می‌دهند. در هنگام آزمایش با طراحی فاکتوریل، باید تعیین کنیم که این

عوامل در رابطه با یکدیگر چقدر لازم هستند. در این روش، اهمیت یک عامل توسط بخشی از کل نوع در متغیر عملکرد اندازه‌گیری می‌شود که توسط این عامل توصیف می‌گردد. برای مثال، اگر دو عامل به ترتیب ۹۰٪ و ۵٪ واریانس عملکرد را توضیح دهند، درنتیجه عبارت دوم را می‌توان به‌عنوان تأثیر کم روی متغیر عملکرد در نظر گرفت. واریانس نمونه برای یک معیار به‌صورت زیر است:

$$Sample\ variance\ (s^2 = \sum_{i=1}^{\#\ exp}(f_i - f)^2\ /\ (\#\ \exp - 1) \qquad (18\text{–}10)$$

که در آن f متوسط زمان پاسخگویی برای تمام آزمایش‌هایی است که برای متغیر عملکرد ارزیابی‌شده است. اگر می‌خواهیم که اطلاعات عملکرد توسط مدل ما منطقی به نظر برسند، روابط میان اطلاعات باید آزمایش و درک شوند. جزئیات مربوط به نحوه وقفه این اطلاعات را می‌توانید در مرجع پیدا کنید.

۱۰–۶ ارائه داده

اگر نتوانیم نتایج مطالعه را ساده و روشن ارائه کنیم، درنتیجه تحقیق ما بیهوده است و مهم نیست که چقدر تلاش برای آن کرده ایم. هدف هر مطالعه با موضوع عملکرد این است که به تحلیلگر و سرویس‌گیرنده مربوطه کمک کند با توجه به سیستم کامپیوتری تحت مطالعه، تصمیم‌گیری کنند. برای کمک به این تحلیل، مدل‌ساز باید توانایی لازم برای تعیین اینکه کدام رسانه برای دسترسی به اطلاعات خاص بهتر است را در اختیار داشته باشد، برای مثال، حروف، تصاویر، گرافیک، نمودارها، انیمیشن یا برخی ابزارهای تابع دیگر برای دامنه تحت مطالعه. یک جمله قدیمی می‌گوید که یک تصویر از هزاران حرف با ارزش تر است، در اینجا یک مدل ساز باید به میدان بیاید و تمایل به درک داشته باشد. تصاویر

گرافیکی از بهترین ابزارها برای درک تفاوت‌های میان مؤلفه‌ها یا سیستم‌های موردبررسی هستند. معمولاً اگر عملکرد CPU ها را به‌صورت گرافیکی ببینیم، به راحتی متوجه می‌شویم که عملکرد کدام‌یک بهتر است، چون تصاویر گرافیکی می‌توانند تفاوت‌های میان عملکردهای بهتر نشان دهند. تصاویر گرافیکی زیادی وجود دارد که می‌توانند این مقایسه‌ها را نشان دهند – برای مثال، گرافیک‌های خطی، نمودارهای میله ای، نمودارهای دایره‌ای، هیستوگرام و نمودار گانت. در تمام این موارد، لازم است چیزی که رسم شده و علت ترسیم را درک کنیم، تا بتوانیم متغیرها و سبک‌های درستی را برای نمایش انتخاب کنیم. یکی از عواملی که روی انتخاب نوع نمودار تأثیر دارد، نوع متغیری است که باید نمایش داده شود. آیا متغیر کمی است یا کیفی، مرتب است یا نامرتب، گسسته است یا پیوسته؟ متغیرهای کیفی متغیرهایی هستند که در آن‌ها هیچ معیار خاصی وجود ندارد و صرفه جزو یک گروه هستند. برای مثال، ریزپردازنده‌ها، سرورها و کامپیوترهای بزرگ همگی از کلاس کامپیوترها هستند اما وقتی که به تنهایی از این عبارات استفاده می‌کنیم هیچ معیار خاصی وجود ندارد. مقادیر کمی مقادیری هستند که می‌توان به‌صراحت آن‌ها را اندازه‌گیری نمود، برای مثال، تعداد دستور در ثانیه یا تعداد درخواست‌های I/O در هر دوره. ما احتمالاً از یک گراف خطی برای نشان دادن رابطه زمانی میان یک مجموعه از متغیرهای پیوسته استفاده می‌کنیم. از سوی دیگر، اگر متغیرهای ما گسسته باشند، می‌توانیم از هیستوگرام یا نمودار میله‌ای برای نشان دادن آن مقادیر استفاده کنیم. هنگامی‌که درباره نوع نمودار تصمیم گرفته می‌شود، مدل‌ساز باید چند مورد مهم را در خاطر داشته باشد. اول، یک مکانیزم گزارشی انتخاب نماید که درک آن برای خواننده خیلی ساده باشد. تفاوت‌هایی که شما قصد نشان دادن دارید باید به روشنی تعریف‌شده

باشند و به نحوی به تصویر کشیده شوند که خواننده به راحتی به نتایج آزمایش‌های مدل‌ساز برسد. اطمینان یابید که تمام اطلاعات مربوطه روی نمودار مشخص است و خواننده نیازی ندارد که برای رسیدن به جواب به‌جای دیگری مراجعه نماید. نمودار باید ساده باشد. حتی اگر موضوع دوم برای قرار دادن تمام اطلاعات مربوطه در گراف باشد، مورد اول باید این باشد که هیچ‌گونه اطلاعات غیرمفیدی در نمودار وجود ندارد. سعی کنید از روش‌های استاندارد توصیف و نمایش اطلاعات استفاده کنید. برای مثال، منشاء نمودار چیزی است که مردم انتظار دارند، به‌عنوان نقطه صفر در هر دو بعد در نظر گرفته شود. درنهایت، از ابهام بر حذر باشید. اطمینان یابید که رئوس به درستی علامت زده‌شده‌اند (یعنی، از نام‌ها استفاده کنید نه علائم) ، مقیاس‌های مورداستفاده را به‌روشنی نشان دهید (یعنی، مقیاس‌های آماری، اعشاری و غیره) و از تفاوت‌های آشکار برای به تصویر کشیدن مقادیر مختلف استفاده کنید (یعنی، CPU نوع ۱ قرمز و نوع دوم سیاه است و ...). مقیاس‌های مورداستفاده باید طوری تنظیم شوند که به‌روشنی تفاوت‌ها را نشان دهند. این مفهوم مهم آخر را نباید نادیده گرفت. انتخاب یک مقیاس نامناسب باعث می‌شود که یک ادعا بهتر یا بدتر ازآنچه در واقعیت هست، باشد. هدف خواننده مورد نظر متن آماری است که روی نمایش داده متمرکزشده است تا بحث‌ها و مثال‌های کاملی از برخی از این مفاهیم ایجاد کند.

۱۰-۷ خلاصه

در این فصل چند مفهوم جدید معرفی کردیم. اولین مفهوم مربوط به استفاده از سیستم‌های پلتفرم است تا به مدل‌سازها در ایجاد مجموعه داده‌ها یا کل سیستم‌هایی کمک کند که باید به‌عنوان محیط‌هایی برای ارزیابی مشکلات عملکرد با طرح‌های جدید یا موجود مورداستفاده قرار گیرند، البته پیش از اینکه ساخته شوند یا پیش از اینکه سیستم موجود تغییر داده شود. به دنبال این ارائه یک مثال از پلتفرم شبکه می‌آید که برای تجزیه‌وتحلیل پروتکل‌ها مورداستفاده قرار می‌گیرد. این مثال به خواننده کمک می‌کند تا روش‌هایی را درک کند که در آن روش‌ها می‌توان از یک پلتفرم به طور مؤثر به‌عنوان بخشی از کار مدل‌سازی بزرگ‌تر استفاده نمود. سپس این بحث به یک بحث عمومی از تکنیک‌های اندازه‌گیری تبدیل می‌شود که در سیستم‌های موجود و پلتفرم اعمال می‌گردند. تکنیک‌های اولیه که توصیف کردیم، مانیتورهای نرم‌افزاری و سخت‌افزاری هستند. معایب و مزایای هر تکنیک به همراه با نمونه‌هایی از نحوه اعمال آن‌ها در یک پروژه مدل‌سازی مورد بحث و بررسی قرار گرفتند.

پس از آن بحث عوض شد و به توسعه تراکم بار برای اداره محیط‌های پلتفرم نگاهی داشتیم. ما روی انواع تراکم‌های بار متمرکز شدیم و به این پرداختیم که چگونه مدل‌ساز تصمیم می‌گیرد که تا چه اندازه روی تراکم بار متمرکز شود و چگونه تعیین کند که سیستم و مؤلفه‌های تراکم بار تحت‌فشار هستند و باید چه نوع خدماتی شبیه‌سازی شوند. سپس به دنبال آن یک ارائه مربوط به ساخت تراکم بار و یا میانگین بار، یک بار واقعی یا چیزی بین آن‌ها انجام شد. به دنبال آن یک بحث از ارائه نتایج پلتفرم و برخی مفاهیم مربوط به اقدام خوب در ارائه خروجی مطالعه عملکرد به مشتری انجام دادیم.

فصل یازدهم

انتخاب و استفاده از یک ابزار برای ارزیابی عملکرد سیستم

هنگامی‌که در ابتدا تصمیم می‌گیریم عملکرد برخی سیستم‌های کامپیوتری را مورد ارزیابی قرار دهیم، باید بدانیم که کدام‌یک از تکنیک‌های گفته شده برای مطالعه عملکرد پیشنهادی مناسب‌تر است. پیش از تصمیم‌گیری، ملاحظات مختلفی باید در نظر گرفته شود.

۱۱-۱ انتخاب ابزار

چهار تکنیک لازم برای ارزیابی عملکرد سیستم‌های کامپیوتری شامل مدل‌سازی تحلیلی، مدل‌سازی شبکه پتری، مدل‌سازی شبیه‌سازی و تحلیل تجربی یا پلتفرم. با توجه به معیار موجود در تحلیل سیستم‌های کامپیوتری، می‌توان یک سری معیار برای انتخاب تعیین نمود. مهم‌ترین معیار با مرحله چرخه عمر سیستم‌های کامپیوتری سروکار دارد. برای مثال، اگر سیستم از قبل وجود داشته باشد یا چیزی مشابه وجود داشته باشد، معیارها تنها به‌عنوان یک احتمال مدل‌سازی در دسترس هستند. از سوی دیگر، اگر یک سیستم کامپیوتری جدید داشته باشیم که ایجاد نشده باشد، درنتیجه، منطقی است که از مدل‌سازی تحلیلی، شبکه‌های پتری یا مدل‌سازی شبیه‌سازی استفاده شود. اگر در مراحل ابتدایی سیکل عمر باشیم، هنگام آزمایش مبادلات میان بسیاری از مؤلفه‌ها، ممکن است بخواهیم که از مدل‌سازی تحلیلی استفاده نماییم، برای اینکه این مدل‌سازی می‌تواند به سؤالات مربوط به مبادلات پاسخ‌های سریعی بدهد و تعیین کند که آیا یک زیرمجموعه از n روش جایگزین برای مدل‌سازی دقیق‌تر مناسب است یا خیر. بعد از

اتمام این تحلیل دشوار و بعد از کاهش تعداد انتخاب روش‌های جایگزین به چند زیرمجموعه مشابه، احتمالاً تمایل داریم که از شبکه‌های پتری برای غربال بیشتر انتخاب‌های خود استفاده کنیم. شبکه‌های پتری باعث دقیق شدن مدل‌سازی و مبادلات، درگیری و هماهنگ‌سازی می‌شوند که برای انجام مدل‌سازی تحلیلی کاری غیرممکن است. هنگامی‌که تحلیل ما با استفاده از شبکه‌های پتری تمام شد و انتخاب ما تنها به چند مؤلفه محدود شد، سپس می‌توانیم شبیه‌سازی انجام دهیم. شبیه‌سازی به ما این امکان را می‌دهد تا مدل‌های دقیق از یک سیستم مورد نظر یا تنها چند مؤلفه مستمر خاص ایجاد کنیم. هدف هر یک از این مراحل اولیه طراحی و توسعه سیستم کامپیوتر، کاهش تعداد انتخاب‌ها است تا بتوانیم بهترین معماری و اجزاء را برای نیازهای کاربردی سیستم کامپیوتری انتخاب کنیم. درنهایت، هنگامی‌که سیستم ایجاد می‌شود، مدل‌سازی تجربی را اعمال می‌کنیم. این کار به ما این امکان را می‌دهد که ببینیم آیا مدل اولیه ما صحیح است یا خیر و احتمالاً مناطقی را شناسایی می‌کنیم که در آن سیستم جدید ما پیش از تحویل به مشتری بیشتر غربال می‌شود و توسعه می‌یابد. معیار بعدی هنگام تصمیم‌گیری در زمینه استفاده از ابزار، زمانی است که برای انجام مدل‌سازی داریم. در بیشتر مواقع یک مدل به این دلیل درخواست می‌شود که مشکلی رخ داده باشد و پاسخی باید به مشکلی که هفته قبل به آن نیاز بود داده شود. گفته می‌شود که زمان، پول است و مدل‌سازی سیستم‌های کامپیوتری، مدل‌سازی آن متفاوت نیست. اگر در دنیا وقت کافی برای انجام ارزیابی‌های خود داشته باشیم، درنتیجه احتمالاً از هر مدل عبور می‌کنیم و تحلیل خود را غربال می‌کنیم و تحت معیار مرحله زمانی آن را تعریف می‌نماییم. مسئله این است که ما معمولاً چنین تجملی نداریم. اگر زمان کوتاه باشد، درنتیجه معمولاً تنها از مدل‌سازی تحلیلی

یا شبکه پتری استفاده می‌نماییم و اگر زمان خیلی کم باشد، مدل‌سازی تحلیلی برنده می‌شود. اگر زمان از اهمیت برخوردار باشد ولی بحرانی نباشد، درنتیجه مدل‌هایی که باید انتخاب کنیم مدل‌های شبکه‌های پتری و شبیه‌سازی هستند. بین این دو مدل، مدل شبکه‌های پتری زمان کمتری نسبت به شبیه‌سازی نیاز دارد و احتمالاً به ما اطلاعات تحلیلی کمتر دقیقی ارائه می‌نماید. اگر سیستم وجود داشته باشد، درنتیجه اگر تعداد روش‌های دیگر خیلی کم باشند، برای مدل‌سازی شبیه‌سازی باید معیارهای مناسبی در نظر گرفته شود. اگر تعداد روش‌های دیگر قابل توجه باشد، درنتیجه شبیه‌سازی برتری داری حتی اگر زمان بیشتری نسبت به معیارها نیاز داشته باشد. سومین معیار انتخاب ابزار مدل‌سازی دسترسی ابزار است. وقتی از دسترسی حرف می‌زنیم، جنبه‌های مختلف زیادی در نظر داریم. اولین جنبه‌ای که به ذهن می‌آید، دسترسی به یک ابزار مبتنی بر کامپیوتر است. برای مثال، اگر ابزاری داشته باشیم که به ما امکان تعریف مدل‌های صف بندی و ایجاد تمایز میان عوامل مدل‌سازی ویژگی‌های مؤلفه سیستمی را بدهد، درنتیجه خیلی ساده‌تر می‌توان از مدل‌سازی تحلیلی استفاده نمود. از سوی دیگر، اگر ابزاری در اختیار نداشته باشیم که از نوع مدل پیشنهادی ما حمایت کند، درنتیجه دسترسی به ما کمک می‌کند تا نشان دهیم مدل‌سازها دانش و توانایی لازم برای ایجاد مدل تحلیلی را دارند و با استفاده از این مدل تحلیل مبادله‌ای را انجام می‌دهند. به همین ترتیب، اگر قصد داریم که از شبکه‌های پتری استفاده کنیم، ابتدا باید بررسی کنیم که آیا ابزارهای مبتنی بر کامپیوتر وجود دارند یا خیر. دوم، مدل‌سازهایی داریم که دانش مربوط به ابزارها را دارند و سوم اینکه اگر ابزاری وجود نداشته باشد، آیا کارکنان بخش مدل‌سازی دانش کافی برای ایجاد یک مدل شبکه پتری از سیستم کامپیوتری هدف را دارند؟ اگر قصد داریم یک مدل

شبیه‌سازی ایجاد کنیم، ابتدا باید ببینیم که آیا ابزارهای شبیه سازی کافی داریم تا کلاس مدل موردنیاز ما را فراهم کنند. برای مثال، یک نفر می‌تواند چند ابزار شبیه سازی با هدف تحلیل شبکه خریداری نماید که احتمالاً برخی از آن‌ها برای معماری‌ها هستند و احتمالاً هیچ کدام برای سیستم‌عامل‌ها نیستند. اگر یک مدل خاص وجود داشته باشد، باید تعیین کنیم که آیا این مدل نیازهای مربوط به کار مدل‌سازی را برآورده می‌سازد و اگر خیر، آیا می‌توان آن را طوری تغییر داد تا خواسته‌های ما را برآورده سازد. اگر ابزارهای موجود مناسب کار مدل‌سازی نباشند، باید یک زبان شبیه سازی همه منظوره یا زبان برنامه نویسی همه منظوره انتخاب نماییم و مدل شبیه سازی خود را از ابتدا بسازیم. این یک کار پر زحمت وقت گیر است و مدل سازان باید دارای مهارت‌های برنامه نویسی و طراحی شبیه سازی باشند. توانایی ابزار مدل‌سازی منتخب برای تحویل اطلاعات دقیق مربوط به سیستم تحت بررسی نیز حائز اهمیت است. در مورد دقت، می‌خواهیم بدانیم که آیا این مدل اطلاعات لازم برای ترسیم دقیق سیستم هدف به ما می‌دهد. در مدل‌های تحلیلی مدل‌ساز باید مبادلات و مفروضات زیادی برای تسهیل توسعه مدل و رام کردن آن انجام دهد. این ساده بودن باعث می‌شود که به نتایج مشکوک شویم. شبکه‌های پتری نیز از همین مشکل رنج می‌برند، اما کار آن‌ها مشابه کار مدل تحلیلی نیست. شبیه‌سازی به مدل‌ساز کمک می‌کند تا جزئیات بیشتری از سیستم کامپیوتری هدف وارد کند و ممکن است به مفروضات کمتری نیز نیاز داشته باشد، درحالی‌که طرح‌ریزی آن به سیستم هدف نزدیک‌تر است. اندازه‌گیری سیستم هدف ممکن است بهترین نتایج را در برداشته باشد اما علاوه بر این در معرض مشکلات احتمالی ناشی از تکنیک اندازه گیری اعمال شده نیز قرار دارد. اگر ما از نظارت نرم افزاری استفاده کنیم، بار ناظر روی

سیستم می‌تواند آن‌قدر قابل توجه باشد که میزان دقت نتایج را تا حد زیادی تنزل دهد. این معیار را نباید نادیده گرفت و باید در زمان تصمیم‌گیری و انتخاب یک ابزار برای مدل‌سازی آن را درک کرد.

چهارمین معیار در هنگام تصمیم‌گیری درباره انتخاب ابزار مدل‌سازی، توانایی مدل برای مقایسه روش‌های ساده و کامل دیگر است. اگر یک مدل توانایی لازم برای تغییر پارامترها و بررسی روش‌های دیگر نداشته باشد، درنتیجه توانایی لازم برای مطالعه مبادله عملکرد را ندارد. کم انعطاف‌ترین ابزار، مدل تحلیلی و پلتفرم است. این ابزارها را به‌سختی می‌توان تغییر داد، برای اینکه ما احتمالاً به ترکیب چند مؤلفه در محیط نیاز داریم تا مؤلفه‌های جایگزین را آزمایش کنیم یا در صورتی که کل سیستم را مقایسه کنیم، تمام این سیستم‌ها در دسترس باشند. مدل‌های تحلیلی را می‌توان به‌سرعت تغییر داد تا پیکربندی‌ها یا مؤلفه‌های مختلفی را آزمایش کنیم و بنابراین، یک ابزار جالب برای تحلیل بسازیم. شبکه‌های پتری نیز مشابه مدل‌های تحلیلی هستند و به خود اجازه می‌دهند تا به‌صورت عادلانه تغییرات ساده‌ای ایجاد کنند. مدل‌های شبیه‌سازی را می‌توان به نحوی ایجاد کرد که امکان مبادله مؤلفه‌های مختلف با یکدیگر را داشته باشند. برای مثال اگر پروتکل‌های حافظ – مدیریت را مبادله می‌کنیم، می‌توانیم آن‌ها را روی یک ماژول پیاده‌سازی کنیم بدون اینکه مؤلفه‌های باقیمانده مدل شبیه‌سازی را تغییر دهیم. چنین رویکردی به ما اجازه می‌دهد تا روی تفاوت‌های هر یک از این پروتکل‌ها در یک سیستم متمرکز شویم. تیم مدل‌سازی به دلیل مشکلات مربوط به هزینه معمولاً یک معیار انتخاب را نادیده می‌گیرند. بیشتر پروژه‌های مدل‌سازی روی هدف موجود متمرکز می‌شوند و همیشه با این پروژه همانند پروژه‌های مهندسی دیگری که در آن‌ها عملکرد و هزینه را باید در نظر داشت، برخورد نمی‌کنند. هزینه می‌تواند

شامل سیستم تحت بررسی، ابزار مورداستفاده در اجرای مطالعات تجاری و کارکنان مدل‌سازی باشد. به طور مستقیم، می‌توانیم متوجه شویم که اگر از سیستم‌های تجربی یا پلتفرم به‌عنوان ابزار خود استفاده کنیم، هزینه شامل بهای واقعی سیستم، هزینه راه‌اندازی این سیستم‌ها برای اندازه‌گیری و هزینه کارکنانی است که ارزیابی عملکرد را انجام می‌دهند. این هزینه‌ها برای پروژه‌های های بزرگ‌تر به بودجه بیشتری نیاز دارند. علاوه بر این، هزینه تغییر سیستم‌های بین اجرای تحلیل ممکن است گران و حتی غیرممکن باشد. به همین دلیل، معمولاً در پروژه‌های تحلیل سیستم‌های بزرگ از شبیه‌سازی‌ها استفاده می‌شود که در آن‌ها به مطالعات گسترده‌ای نیاز است. شبیه‌سازی برای تغییر و اجرا نسبت به سیستم واقعی یا حتی یک پلتفرم خیلی ساده‌تر است. درنهایت، تولید مدل‌های تحلیلی و پتری ممکن است کم‌هزینه‌تر باشند، برای اینکه معمولاً نیاز به توسعه یا پیاده‌سازی نرم‌افزاری بزرگی ندارند. بیشترین هزینه‌های این مطالعات مربوط به حقوق و زمان تحلیلگر است.

۲-۱۱ اعتبار نتایج

ابزارهای منتخب باید نتایجی تولید کنند که درست و سازگار باشند و بنابراین، مشتریان را متقاعد کنند. اگر نتایج و مفروضات مورداستفاده برای رسیدن به آن‌ها از نتیجه دلخواه سیستم خیلی دور باشند، تحلیل فوق مشکوک است و نباید از آن استفاده نمود. نتایج تحلیلی به این حوزه صلاحیت بازمی‌گردد، برای اینکه بیشتر مردم زمانی که نوبت به مفروضات و تسهیل موردنیاز برای اعمال این مدل‌ها می‌رسد، دچار تردید می‌شوند. شبیه‌سازی‌ها نیز به خاطر ماهیت ساخت مدل شبیه‌سازی، از این مشکل رنج می‌برند.

وقتی نوبت به جزئیات خاصی می‌رسد، برای شبیه‌سازی مدل‌ساز باید تبادلاتی انجام دهد. برخی از این مبادلات ممکن است نتایج را برای مشتری غیرواقعی‌تر نشان دهد. این بسیاری از توسعه‌دهندگان شبیه‌سازی از یک نقص بزرگ رنج می‌برند. آن‌ها معمولاً قبل از اعمال مدل‌ها روی مسئله تحت بررسی، صحت آن‌ها را کاملاً بررسی نمی‌کنند. بعد از تعیین کنیم ابزار لازم برای مدل‌سازی و ایجاد مدل، هنوز نمی‌توانیم آزمایش‌های خود را اجرا کنیم. ابزار و مدل منتخب باید به نحوی معتبر شوند که نتایج تولید شده توسط آن‌ها قابل‌باور شوند. اعتبار دهی به یک ابزار با انتخاب ابزار یا ابزارهای دیگر و جمع‌آوری اطلاعات از ابزارهای دیگر آغاز می‌شود. با استفاده از این مجموعه اطلاعات، مدل‌ساز ابزار جدیدی برای همان پیکربندی اجرا می‌نماید و نتایج حاصل از چند ابزار را باهم مقایسه می‌کند. نتایج حاصل از تمام ابزارها باید یکسان باشند. قوانین سخت و سریعی وجود ندارد، برای اینکه بدانیم چگونه نتایج یک ابزار معتبر باید نقطه‌به‌نقطه با ابزار مورداستفاده برای اعتبارسنجی مورد مقایسه قرار گیرد. بسیاری از مطالعات مربوط به شبیه‌سازی از یک معیار استفاده می‌کنند و به دنبال نتایج مشابه هستند که بیش از ۵٪ با یکدیگر اختلاف نداشته باشند.

برای اعتباردهی، مدل‌ساز باید به اجزای مختلف مدل نگاه کند. ابتدا، آیا مدل با سیستم تحت مطالعه ارتباطی دارد؟ یعنی، آیا نماینده قابل‌اطمینانی از سیستم واقعی است؟ برای مثال، اگر مدل دو پردازنده داشته باشد و سیستم واقعی یک پردازنده، نمی‌تواند نماینده قابل‌اطمینانی باشد. دوم، آیا مفروضات مورداستفاده مدل‌ساز ازنظر سیستم واقعی که باید مدل‌سازی شود واقع‌گرایانه است؟ سوم، آیا توزیع‌ها و پارامترهای ورودی مدل، پارامترهای مقادیر سیستم واقعی را ردیابی می‌کند، آیا در دسترس هستند؟

اگر در دسترس نیستند، آیا برخی مدل‌های دیگر ایجاد شده برای یک پروژه مشابه را ردیابی می‌کنند؟ درنهایت، آیا نتایج و نتیجه‌گیری از روی نقشه مدل مطلوب نتایج سیستم اندازه‌گیری شده یا ابزارهای دیگر است؟ علاوه بر این، آیا نتیجه‌گیری مدل‌های معتبر از نتایج سیستم واقعی یا مدل دیگر به طور مداوم و درست پیروی می‌کند؟ هر یک از این سؤالات را می‌توان به شیوه‌های مختلفی پاسخ داد. آن‌ها را می‌توان با استفاده از شهود متخصص، با اندازه‌گیری سیستم و مقایسه نتایج یا از طریق نتایج تحلیلی پاسخ داد. منظور از شهود متخصص مدل‌سازهایی است که آزمایش‌های زیادی درگذشته انجام داده اند. با استفاده از این دانش انبوه، مدل‌ساز می‌تواند نتایج را مورد آزمایش قرار دهد و تعیین کند که آیا مدل فوق ازنظر او نماینده خوبی از سیستم تحت مطالعه است یا خیر. این کارشناسان شامل طراحان، معماران، مجریان، تحلیلگران، بخش نگهداری و اپراتورها و حتی کاربران سیستم تحت بررسی هستند. چیزی که نمی‌خواهیم این است که کارشناس اعتبارسنجی تیم، عمل طراحی مدل فوق را انجام دهد.

سنجش سیستم واقعی قابل‌اطمینان‌ترین نوع اعتبارسنجی مدل است اما به سخت‌ترین شکل ممکن است. علت این است که یک سیستم واقعی تاکنون وجود نداشته است یا اطلاعات جمع‌آوری‌شده تاکنون وجود نداشته اند. احتمالاً در صورت وجود معیارهایی برای یک سیستم موجود، ممکن است طیف کاملی از اطلاعات موردنیاز برای اثبات داده‌های مدل را نشان ندهد. روش آخر برای کسب اطلاعات مربوط به اعتبارسنجی موردنیاز، استفاده از نتایج تحلیلی است. تا زمانی که یک مدل سعی در کسب اعتبار دارد، یک مدل تحلیلی محسوب نمی‌شود، این‌یکی از ابزارهای موجود و قابل قبول برای

اعتبارسنجی اطلاعات است. با تنظیم پارامترها برای شبیه‌سازی به‌صورت پارامترهای یک مدل تحلیلی، در تئوری ما باید بتوانیم همان نتایج را برای مدل تحلیلی به دست آوریم.

۱۱-۳ انجام آزمایش‌ها

با توجه به اینکه ما یک ابزار انتخاب کرده‌ایم و مدل خود را ایجاد و معتبر کرده ایم، در ادامه باید آزمایش‌های خود را برای انجام تابع مورد نظر برای بررسی عملکرد توسعه دهیم. برای توسعه تجربیات خود، باید در نظر داشته باشیم که معیارهای عملکرد (متغیرهای عملکرد) برای این مطالعه کدام‌اند. پیش از این در فصل ۸ دیدیم که برای توسعه مجموعه‌ای از متغیرهای عملکرد، باید فهرستی از سرویس‌ها را توسعه دهیم تا توسط سیستم تحت بررسی ارائه گردد. با توجه به اینکه ما انتخاب و تعریف سرویس‌ها را از قبل انجام دادیم، در ادامه باید تمام خروجی‌های احتمالی را برای سرویس تعیین کنیم. برای مثال، هر سرویس ممکن است یک درخواست در حال بررسی از سرویس داشته باشد، در حال استفاده از سرویس باشد، در حال تکمیل سرویس باشد یا یک درخواست سرویس را رد کند. نتایج درخواست سرویس باید با درخواست سرویس آینده موافقت کنند، سرویس را به‌صورت کامل یا ناقص انجام دهند یا درخواست را به‌عنوان یک درخواست غیرممکن رد کنند. برای مثال، مدیر قفل یک سیستم پایگاه داده را می‌توان به‌عنوان یک درخواست برای قفل‌کردن، به تأخیر انداختن، اجرای درخواست به‌صورت اشتباه پذیرفت یا درخواست قفل را رد نمود.

اگر سیستم درخواست را به درستی انجام دهد، عملکرد به‌صورت زمان سپری‌شده برای انجام سرویس، نسبتی که سرویس طبق آن انجام می‌شود و منابع مصرف‌شده در حین اجرای سرویس مورد درخواست،

در نظر گرفته می‌شود. این سه معیار مربوط به معیار پاسخگویی، بهره وری و تولید هستند، که تمام مؤلفه‌های مهم مربوط به مطالعات عملکرد سیستم کامپیوتری هستند. این معیارها را می‌توان با توجه به‌سرعت، قابلیت اطمینان و دسترس بودن تغییر داد. برای مثال، پاسخگویی یک سیستم پردازش تراکنش توسط زمان پاسخگویی آن سنجیده می‌شود. این شامل زمان بین یک درخواست تراکنش برای سرویس و پاسخ به تراکنش از سرور است. بهره وری سیستم‌های پردازش تراکنش توسط توان عملیاتی آن اندازه‌گیری می‌شود. توان عملیاتی شامل تعداد تراکنش‌های انجام‌شده در یک واحد زمانی تعریف‌شده است (یعنی، در دقیقه یا ثانیه). معیار بهره وری یک معیار از یک کسب‌وکار منبع است. در مثال پردازش تراکنش، چه درصد زمانی را سرور مشغول انجام تراکنش‌ها است و در فواصل زمانی به‌عنوان معیار توان عملیاتی بیکار می‌باشد. با استفاده از این اطلاعات می‌توانیم مشکلات را در سیستم خود تفکیک نماییم. برای مثال، سرویسی که بیشترین کاربرد را دارد، دچار تنگنای سیستمی می‌شود. اگر یک سرویس کار اشتباهی انجام دهد، به اطلاعات مربوط به آن نیز نیاز داریم. تشخیص و ارزیابی خطا برای تعیین تحمل یک سرویس در مقابل خطا بسیار مهم است. برای مثال، در سیستم پردازش تراکنش، ممکن است بخواهیم علت بروز خطای تراکنش‌های تحت پردازش را بدانیم. شاید لازم باشد بدانیم که یک خطا به دلیل آسیب سخت‌افزاری رخ داده یا نرم افزاری یا در اثر درگیری یا یک طراحی ضعیف معامله به وجود آمده است. هر کس به ما چیزی درباره محصولی که ارزیابی می‌کنیم، می‌گوید. اگر یک منبع نتواند عملیات سرویس را اجرا نماید، می‌توان گفت که کارایی ندارد یا ۱۰۰٪ مورد بهره برداری قرار گرفته است. یک بار دیگر متذکر می‌شویم که دانستن وضعیت منبع مربوط به تعیین عملکرد کلی آن است.

۴-۱۱ معیارهای عملکرد

در مطالعات مربوط به سیستم‌های کامپیوتری، ما معمولاً به چنین معیارهایی علاقه‌مند هستیم نه معیارهایی که برای سیستم ساده باشد. سیستم‌های کامپیوتری که ما مدل‌سازی خواهیم کرد، شامل سیستم ها، مؤلفه‌ها و کاربران است. هر کسی دیدگاه خاص خود را از معیارها دارد و هر کسی از چشم انداز متفاوتی به عملکرد سیستم‌ها نگاه می‌کند. برخی معیارها در حد سیستم یا جهانی هستند، درحالی‌که معیارهای دیگر می‌توانند محلی یا فردی باشند. مواردی وجود دارد که در آن‌ها بهینه‌سازی یک معیار جداگانه روی معیار جهانی اثر دارد و زمانی نیز دیده می‌شود که این کار تأثیر اندکی دارد یا تاثیری ندارد. این، سطوح مختلف می‌توانند اهداف اولیه متفاوتی داشته باشند. ممکن است در یک سطح ما به دنبال بیشترین استفاده و در سطح دیگر کمترین استفاده باشیم که این امر بستگی به هدف سیستم و اجزای مختلف آن دارد. این نشان می‌دهد که معیارهای لازم برای مدل‌سازی باید در سطوح مختلفی انتخاب شوند به نحوی که بتوانیم تحلیل مناسبی از عملکرد سیستم ایجاد نماییم. طراح یا مدل‌ساز باید مجموعه کاملی از متغیرهای معیار تعیین نماید. سپس، این معیارها باید در رابطه با واریانس، افزونگی و تکامل خود مورد آزمایش قرار گیرند. اگر یک معیار واریانس کمی داشته باشد، فرض می‌کنیم که ثابت است و آن را از فهرست سرویس‌ها و معیارهای مورد نظر حذف می‌کنیم. اگر یک متغیر اطلاعاتی فراهم کند که متغیر دیگری نیز فراهم می‌کند، یکی از آن‌ها باید حذف شود. برای مثال، طول صف برابر است با تعداد سرویس بعلاوه آن‌هایی که منتظر سرویس هستند، بنابراین نیازی نیست که به آن‌ها توجه کنیم. تکامل یعنی اطمینان یابیم که متغیرها بازتابی از سیستم واقعی هستند. هنگام مدل‌سازی سیستم‌های کامپیوتری،

معیارهای کارایی بسیار مشترکی وجود دارند. رایج‌ترین آن‌ها، زمان پاسخگویی (که گاهی به آن سرعت، زمان چرخش واکنش نیز می‌گویند)، توان عملیاتی (که به آن ظرفیت یا پهنای باند می‌گویند)، استفاده (کارایی یا کسب‌وکار)، قابلیت اطمینان و نسبت هزینه به عملکرد است.

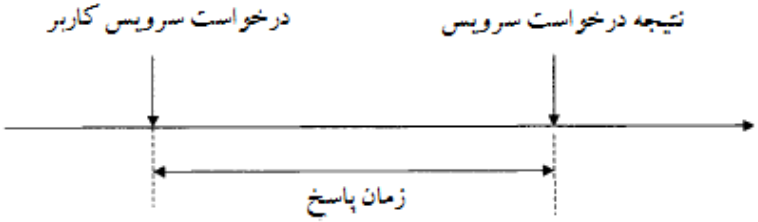
۱۱-٤-۱ زمان پاسخگویی

زمان پاسخ همان فاصله زمانی بین درخواست یک کاربر برای سرویس و دریافت سرویس است (شکل ۱۱-۱). در واقعیت این زمان بیش از اندازه ساده است و چیزی که رخ می‌دهد نمی‌باشد. مؤلفه‌های زیادی در دو سوی درخواست/پاسخ وجود دارد تا معیارهای درستی را ایجاد کنند. اگر درباره سیستم پردازش تراکنش مورداستفاده در مثال قبل فکر کنیم، متوجه می‌شویم که کار با وارد کردن تراکنش از سوی کاربر آغاز می‌شود. فرض می‌کنیم که این تنها یک مرحله باشد اما اگر کاربر از رابط تراکنشی برای سرویس تراکنشی استفاده کند، این زمان می‌تواند طولانی‌تر شود. سیستم پایگاه داده باید ساختمان داده مناسبی داشته باشد و منابعی را برای اجرای تراکنش‌ها فراهم نماید. سپس این تراکنش توسط موتور پایگاه داده اجرا می‌شود. سپس این تراکنش پردازش می‌شود و نتیجه کار ارسال می‌گردد (شکل ۱۱-۲). هر یک از این مراحل درحالی‌که اندکی کامل‌تر از مدل ساده هستند، هنوز نمایش جزئی از یک سیکل پردازش تراکنش کامل در سیستم پایگاه داده تجاری محسوب می‌شوند. هر یک از این اجزای زمان پاسخگویی تراکنش، یک مؤلفه زمان پاسخ محسوب می‌شوند. این مؤلفه‌ها اجزای فرعی کل زمان پاسخگویی هستند، مثل زمان انتظار در صف و زمان لازم برای سرور که نشان‌دهنده زمان کار در یک مدل صف بندی هستند.

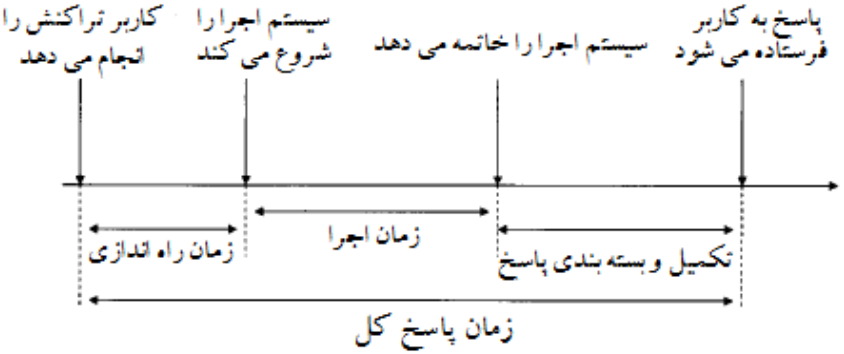
زمان پاسخگویی برای یک سیستم کامپیوتری معمولاً با افزایش بار، زیاد می‌شود. برای ایجاد قاعده انگشت شست در چنین حالاتی، اندازه‌گیری‌های زیادی انجام‌شده است. عاملی به نام عامل کشش وجود دارد که به‌عنوان زمان پاسخگویی مورد انتظار در زمان موردنیاز سرور تعریف می‌شود، یا:

$$\text{عامل کشش} = E[W]/E[S] \qquad (۱-۱۱)$$

این معیار در شکل (۱۱-۳) نشان داده‌شده است. در بیشتر سیستم‌های واقعی، انتظار داریم که این عامل کشش دارای مقداری حدود ۵ باشد. اگر این عامل مقداری بیش از ۵ داشته باشد، نشان می‌دهد که زمان انتظار طولانی‌تری در رابطه با زمان‌های سرویس وجود دارد و بنابراین، میزان دسترسی منابع کمتر و میزان استفاده بالاتر خواهد بود.



شکل (۱۱-۱): اندازه‌گیری معمولی زمان پاسخگویی

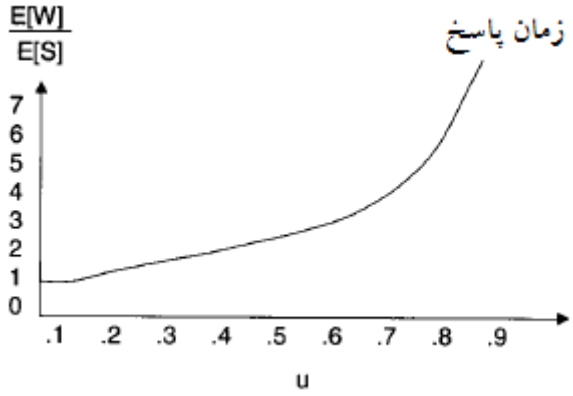


شکل (۱۱-۲): تقسیم‌بندی پاسخ به پردازش تراکنش

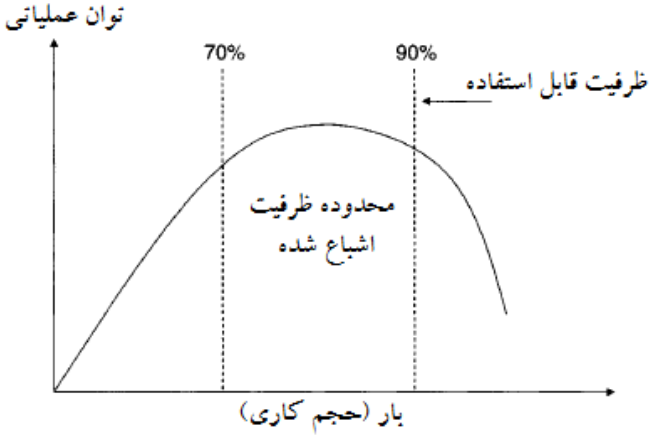
۱۱-٤-۲ توان عملیاتی

توان عملیاتی مربوط به‌اندازه گیری تعداد اقلامی است که باید ارزیابی شوند (مثل تراکنش‌ها) و سرویس (یعنی، اجرای کامل تراکنش) را در دوره‌های زمانی از پیش تعریف‌شده دریافت کند. برای سیستم تراکنشی که بحث کردیم، توان عملیاتی را به‌صورت تعداد تراکنش در هر ثانیه یا TPS اندازه‌گیری می‌کنیم. در CPU سیستم‌های کامپیوتری، معیار MIPS یا میلیون دستورالعمل در ثانیه است. در سیستم‌های ارتباطی، معیار ما می‌تواند MPS برای تعداد پیام در ثانیه یا BPS برای بیت در ثانیه نیز باشد. توان عملیاتی در مقابل زمان پاسخ، به‌صورت بار اضافه موجود در یک سیستم رشد می‌کند. بااین‌حال، برخلاف زمان پاسخ، نقطه‌ای وجود دارد که در آن توان عملیاتی به حداکثر می‌رسد و احتمالاً رو به تخریب می‌گذارد (شکل ۱۱-۴). در این شکل به نظر می‌رسد که توان در محدوده گسترده ای از بار زیاد می‌شود و سپس با رسیدن به نقطه اشباع، کاهش می‌یابد. در مورد توان عملیاتی، توان تا برخی سطوح بیشینه افزایش می‌یابد و سپس

تنزل پیدا می‌کند. در نقطه بحرانی در بار، که در آن زمان پاسخگویی به‌صورت نمایی افزایش می‌یابد، توان عملیات کمتر از بیشینه می‌شود. این منحنی‌ها نمونه ای از سیستم‌های کامپیوتری هستند که در آن ظرفیت سرویس برای بار ارائه شده، کافی نیست. ما می‌خواهیم که توان را نزدیک اوج نگه داریم اما از محدوده اشباع خیلی دور باشد، تا منابع موجود برای خوشه‌های بار همیشه در دسترس باشند.



شکل (۱۱-۳): مقایسه عامل کشش با استفاده



شکل (۱۱-۴): منحنی توان نسبت به منحنی پاسخ

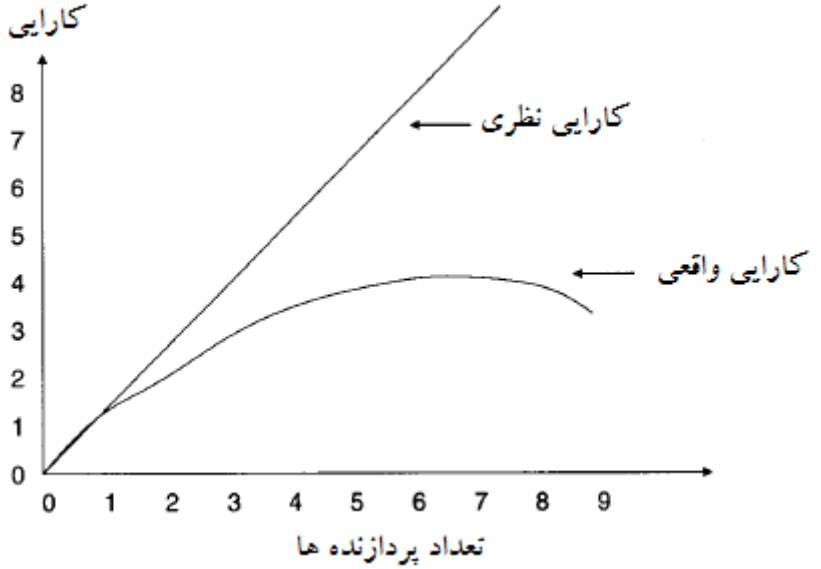
۱۱-٤-۳ کارایی

معیار مهم بعدی کارایی است. این معیار مربوط به بهره برداری و توان عملیاتی می‌باشد. این روابط نگاهی به نسبت حداکثر توان موجود به توان عملیاتی واقعی دارد:

کارایی = توان عملیاتی واقعی تقسیم بر توان عملیاتی نظری (۱۱-۲)

اگر پردازنده ای با سرعت ۱۰۰ مگافلاپس (عملیات نقطه اعشاری) داشته باشیم، هنگام اجرا در یک پلتفرم ما ۹۰ مگافلاپس را اندازه می‌گیریم یعنی میزان کارایی پردازنده ۹۰٪ است. کارایی را می‌توان برای سیستم‌هایی با چند منبع نیز اندازه گیری نمود. یکی از کاربردهای عمومی آن هنگامی است که به سرعت عملکرد یک پردازنده نسبت به n پردازنده نگاه می‌کنیم. کارایی در این کلاس از محیط به‌صورت نسبت توان تئوری به تعداد دستگاه ها، تقسیم بر سرعت یک دستگاه واحد. در شکل (۱۱-۵) می‌بینیم که کارایی نظری افزودن پردازنده‌های بیشتر، یک منحنی خطی با یک کارایی معادل با تعداد

دستگاه‌های اعمال شده است. منحنی واقعی کمی متفاوت است و کارایی خطی نیست و با افزایش تعداد دستگاه‌ها رو به زوال می‌گذارد. علت آن اضافه شدن سربار برای استفاده کارآمد آز پردازنده‌ها می‌باشد.



شکل (۱۱-۵): منحنی کارایی پردازنده چندگانه

۱۱-٤-٤ بهره برداری، قابلیت اطمینان و دسترسی

بهره برداری از یک منبع، معیاری است برای میزان مشغول بودن منبع. این عامل به‌صورت کسری از زمانی که یک منبع مشغول سرویس‌دهی به مشتری است تقسیم بر کل دوره زمانی، تعریف می‌شود:

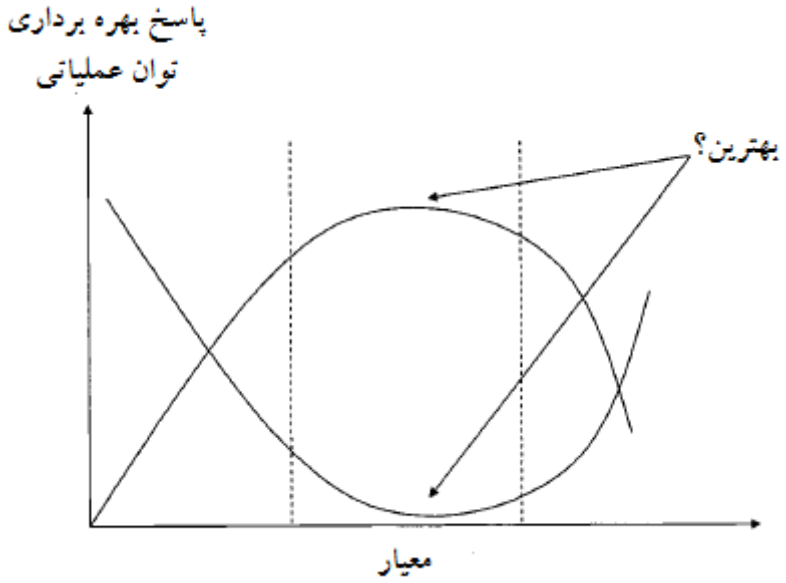
بهره برداری = زمان مشغول بودن تقسیم بر (زمان مشغول بودن + زمان بیکاری)

هدف در بیشتر سیستم‌ها به اشباع رساندن منبع نیست (یعنی، ۱۰۰٪ مشغول نگه داشتن آن‌ها) بلکه استفاده متعادل است به نحوی که هیچ سیستمی نسبت به دیگری سخت تر کار نکند. در اصل هدف همین است، اما در واقع خیلی سخت می‌توان به آن رسید. بهره برداری در هنگام آزمایش سیستم‌ها یک معیار مهم محسوب می‌شود. دستگاه‌های مختلف موجود در سیستم دارای میانگین بهره وری مختلفی هستند. برای مثال، پردازنده‌ها معمولاً کاربرد زیادی دارند در حالی که حافظه، دیسک و ابزارهای جانبی دیگر زمان استفاده کوچک‌تری دارند. معیارهای مهم دیگر در عرصه تحلیل سیستم‌های کامپیوتری شامل قابلیت اطمینان سیستم‌ها و دسترسی سیستم‌ها است. قابلیت اطمینان به معیار احتمال وجود خطا یا یک معیار از زمان معمول میان خطاها گفته می‌شود. بیشتر سیستم‌های کامپیوتری نسبتاً قابل اطمینان هستند یعنی سخت‌افزار آن‌ها از نرم افزار قابل اطمینان تر است. در دسترس بودن یک سیستم در رابطه با قابلیت اطمینان اندازه گیری می‌شود. اگر یک سیستم بسیار قابل اعتماد باشد، امکان دسترسی آن بیشتر از عدم دسترسی است. اما اگر یک سیستم غیرقابل اطمینان باشد، درنتیجه دوره‌های خرابی خواهد داشت که در آن سیستم یا اجرا نمی‌شود و یا همراه با خطا اجرا می‌شود. در صورت بروز عیب، معیار مهم دیگر یعنی زمان تعمیر یا MTTR مطرح می‌شود. MTTR به طور متوسط نشان می‌دهد که بعد از بروز یک خطا، سیستم تا چه مدت از دسترس خارج است. اگر خطاها قابل اندازه گیری و پیش بینی باشند، می‌توانیم معیارهایی مثل زمان میانگین برای بروز خطا یا MTTE را ایجاد کنیم. معیار آخر که توسط تحلیلگران سیستم برای مقایسه سیستم‌ها یا مؤلفه‌ها مورداستفاده قرار می‌گیرد، نسبت هزینه به عملکرد است. این معیار برای تعیین اینکه کدام سیستم‌ها دارای عملکرد نسبی یکسان هستند و خرید کدام یک

بهتر است، استفاده می‌شود. در این مورد، هزینه حداقل شامل هزینه نرم افزار و سخت‌افزار است اما ممکن است شامل صدور مجوز، نصب، نگهداری و حتی عملیات نیز باشد.

۱۱-۵ ارزیابی

تمام این معیارهای کارایی بی معنا هستند مگر اینکه رابطه ای مربوط به معیار در آن‌ها وجود داشته باشد. برای مثال، چگونه بفهمیم که برای یک معیار خاص، میزان هزینه باید حداقل یا حداکثر شود؟ برای اینکه قضاوت ما صحیح باشد، باید معیارهای مورد نظر خود و رابطه آن‌ها با ارزش‌های سیستم را به خوبی درک کنیم (شکل ۱۱-۶). برای مثال، برای یک CPU آیا دوست دارید که تعداد دستورالعمل در ثانیه بالا باشد یا پایین؟ آیا به دنبال میانه هستید یا حالت؟ در تفسیر نتایج تفاوت‌هایی وجود دارد. برای اینکه تصمیم گیری ما پیرامون تفسیر معیارها منطقی باشد باید رابطه آن‌ها با یکدیگر را درک کنیم. برای مثال، کاربرد زیاد از دیسک ممکن است به کاهش توان عملیاتی سیستم منجر شود. یا کاربرد زیاد CPU ممکن است توان را افزایش دهد. باید بدانیم که کدام، کدام است تا بتوانیم تصمیم درستی بگیریم.



شکل (۱۱-۶): معیارها نسبت به مفید بودن

چگونه بفهمیم که کدام عملکرد مناسب است یا خوب – به ویژه اگر برای سیستمی که ما این سؤال برای آن در نظر داریم هنوز وجود نداشته باشد. مشکل تنظیم الزامات عملکرد برای یک سیستم غیرموجود است. ما معمولاً الزامات را به شیوه غیرکمی مشخص می‌کنیم. ما ممکن است با اظهاراتی این‌چنینی مواجه شویم: سیستم بهتر است سربار کمی داشته باشد، سرعت حافظه و پردازنده باید هماهنگ باشد، احتمال خطا باید خیلی کم باشد و غیره. در تمام این موارد، ما به الزامات کیفی اشاره‌کرده این که ممکن است اندازه گیری و درک آن‌ها خیلی سخت باشد. این عبارات غیراختصاصی، غیرقابل ارزیابی و بنابراین، غیرقابل قبول هستند. برای تغییر این، تحلیلگر باید به نوع کاربرد سیستم نگاه کند و اینکه برای یک سیستم معمولی با بار یکسان به چه ظرفیتی نیاز است. این باید برخی عوامل رشد (مثلاً ۱۰۰٪) را

اضافه کنیم. بنابراین، سیستمی را مشخص می‌کنیم که بتواند تمام الزامات پردازشی را برآورده سازد و در عین حال ظرفیت رشدی معادل با ظرفیت سیستم موجود داشته باشد.

۱۱-۶ خلاصه

در این فصل چند قانون ساده معرفی کردیم که در هنگام انتخاب یک ابزار برای مدل‌سازی یک پروژه مدل‌سازی خاص در نظر گرفته می‌شوند. ما نشان دادیم که اگر زمان و پول جزو عوامل نباشند، از تمام روش‌ها استفاده خواهیم کرد. ابتدا، از مدل‌سازی تحلیلی برای حذف فوری طرح‌های جایگزینی که نیازهای سیستم هدف را برآورده می‌سازند، استفاده کردیم. دوم، از مدل‌های شبکه پتری برای مقایسه بیشتر و حذف روش‌های جایگزین مورد نظر استفاده کردیم. سوم، از شبیه سازی برای مطالعه چند مؤلفه یا سیستم جایگزین استفاده کردیم. شبیه سازی‌ها برای مدل‌سازی دقیق مؤلفه‌ها یا عملیات ایجاد شدند. ابزار چهارم همان پلتفرم‌ها بودند. پلتفرم‌ها پیچیدگی بیشتری دارند و ما از این روش جایگزین زمانی استفاده می‌کنیم که تنها چند راه یا یک راه جایگزین در اختیار داشته باشیم که باید اعتبارسنجی شوند. از آنجایی که دنیای ما، کامل نیست باید زمان و پول را در نظر بگیریم، بنابراین انتخاب ابزار مدل‌سازی تحت هدایت این ملاحظات انجام می‌شود. اگر هزینه بالاترین اهمیت را داشته باشد، بهتر است به مدل‌سازی تحلیلی فکر کنیم برای اینکه ازنظر هزینه نسبتاً ارزان است. اگر مشکل هزینه نداشته باشیم، درنتیجه پلتفرم‌های ساخت را می‌توانیم استفاده کنیم. اگر هزینه جایی بین این دو باشد، می‌توانیم شبیه سازی یا شبکه‌های پتری را مورداستفاده قرار دهیم. اگر زمان اصل باشد، تئوری صف بندی بر دیگران ارجح است برای اینکه یک مدل را می‌توان ایجاد و تجزیه و تحلیل نمود. اگر زمان کافی در اختیار

داشته باشیم، شبیه سازی یا پلتفرم ها، انتخاب‌های مناسبی هستند. بعد از این مبحث، برخی از مؤلفه‌های یک پروژه مدل‌سازی را آزمایش می‌کنیم، این مؤلفه‌ها به ما کمک می‌کنند تا درباره انتخاب ابزار مدل‌سازی تصمیم گیری کنیم. معیارهای موردنیاز و راحت و دقیق بودن آن‌ها ما را به سمت برخی ابزارهای خاص هدایت می‌کنند. اگر به اطلاعات دقیقی نیاز داشته باشیم، می‌توانیم از مدل‌های تجربی و پلتفرم‌ها استفاده نماییم برای اینکه می‌خواهیم سیستم واقعی یا یک نمونه اولیه از آن را ارزیابی نماییم. اگر دقت زیاد برای ما مهم نباشد، می‌توانیم از مدل‌های تحلیلی استفاده کنیم، برای اینکه این مدل‌ها به‌سادگی ساخته می‌شوند و می‌توانند تحلیل کلی تری به ما ارائه دهند. در ادامه این فصل به برخی مفاهیم مربوط به مدل‌سازی یک سیستم پرداختیم، برای مثال، به این پرداختیم که چگونه تعیین کنیم داده یک مدل صحیح است یا آیا نتایج ما خوب اند یا بد. تفسیر نتایج بستگی به دانش ما از معیارهایی موجود و رابطه آن‌ها با معیارهای سیستم‌های مهم مثل، توان عملیاتی، کاربرد و زمان پاسخگویی دارد.

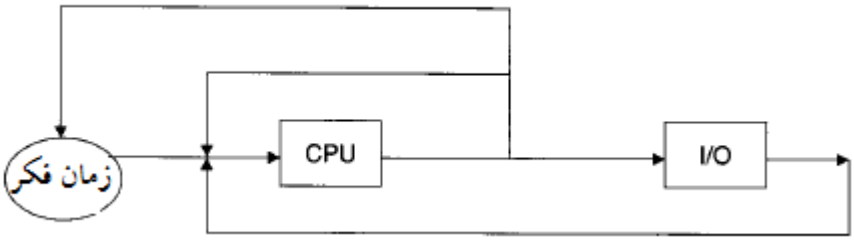
فصل دوازدهم

تحلیل معماری کامپیوتر

مدل‌سازی تحلیلی و شبکه پتری در فصل‌های قبل معرفی شدند. در این مبحث، مفاهیم اصلی سیستم‌های صف بندی و تئوری‌های شبکه پتری، کاربرد آن‌ها در مدل‌سازی سیستم‌های کامپیوتری و مقدمه ای بر مدل‌سازی سیستم‌های کامپیوتری خواهیم داشت. این فصل به استفاده از مدل‌های تحلیلی و شبکه پتری می‌پردازد به ویژه به استفاده از آن‌ها به‌عنوان ابزارهای ارزیابی عملکرد که روی مدل‌سازی معماری‌های کامپیوتری اعمال می‌شوند، می‌پردازیم.

۱۲-۱ مقدمه

در چند سال گذشته، به جای استفاده از روش‌های شبیه سازی آشنا از مدل‌های عملکرد شبکه پتری و تحلیلی به طور گسترده استفاده شده است و علت آن سهولت نسبی پیاده سازی و استحکام نتایج کاربردی آن‌ها بوده است. مدل‌های تحلیلی و شبکه پتری با موفقیت توانستند معیارهای مختلف کارایی مثل توان عملیاتی، میانگین طول صف، متوسط زمان پاسخگویی، رقابت بین منابع و هماهنگ‌سازی برای سیستم‌های واقعی را برآورد کنند. این فصل مقدمه ای بر تکنیک‌های مدل‌سازی پتری و صف بندی است که در معماری‌های کامپیوتری اعمال می‌شوند و تا کنون به‌صورت عمقی بررسی نشده اند. یکی از محبوب ترین مدل‌ها برای طراحان، مدل سرور مرکزی یک سیستم کامپیوتری است، مثل یک کامپیوتر شخصی یا ایستگاه کاری، و یک مدل پردازنده چندگانه معمولاً در یک سرور پیدا می‌شود. به این دلیل، این دو مدل را به‌عنوان یک نمونه اولیه از نحوه اعمال تکنیک‌های مدل‌سازی تحلیلی در روش ارزیابی عملکرد سیستم مورد آزمایش قرار می‌دهیم.



شکل (۱۲-۱): مدل سرور مرکزی

۱۲-۲ مورد ۱: سیستم کامپیوتری سرور مرکزی

مدل سرور مرکزی که در شکل (۱۲-۱) نشان داده‌شده، امروزه از معمول ترین مدل‌های مربوط به کامپیوترهای رومیزی و سرورهای تک پردازنده است. مهم‌ترین عناصر این مدل عبارتند از CPU، حافظه و دستگاه‌های I/O (دیسک ها، اتصالات شبکه و غیره). علاوه بر این اجزای اصلی، این مدل باید نحوه اتصال مؤلفه‌های اصلی به یکدیگر را نیز نشان دهد و معماری سیستم کامپیوتر سرور مرکزی را تشکیل دهد. اتصالات داخلی شامل مسیرهایی است که برای راه‌اندازی برنامه جدید برای برنامه‌های فعال ایجاد می‌شود تا در داخل سیستم به گردش در بیایند. فرض ما بر این است که تعداد کارها (برنامه ها) داخل سیستم ثابت باشد. تعداد ثابتی از کارها برخی از فعالیت‌های CPU را انجام می‌دهند و به دنبال آن فعالیت I/O انجام می‌شود و دوباره برای گرفتن سرویس بیشتر به CPU باز می گردند. سیستم مدل‌سازی شده از کامپیوترهای معمولی تشکیل شده است که می‌تواند شامل میزکار افرادی باشد که در داخل یک دفتر تجاری شلوغ و پر رفت و آمد مشغول به کارند. این سیستم کامپیوتری شامل یک CPU (مثل پنتیوم ۴ با حافظه کش درون برد)، حافظه اصلی با سرعت منطبق (۱۲۸ مگابایت تا ۱

گیگابایت)، حداقل یک درایو دیسک و اجزای جانبی فراوان است. این دستگاه به اینترنت نیز متصل خواهد بود و ممکن است به فراخوان سرویس راه دور نیز سرویس رسانی کند. سیستم‌عامل یکی از استانداردهای صنعت است (میکروسافت XP، لینوکس).

ظرفیت پردازش CPU و هر یک از دستگاه‌های I/O توسط معادله $\mu_i = ۱/T_{S(i)}$ نشان داده می‌شود که در آن $T_{S(i)}$ میانگین زمان سرویس برای یک دستگاه مشخص است. جریان کنترل برای یک کار در سیستم توسط احتمالات انشعابات هدایت شده است $(P_۱, P_۲, P_۳, \dots, P_n)$. در CPU یک کار ممکن است برای دریافت سرویس‌های بیشتر به عقب باز گردد و احتمال آن نیز $P_۱$ در نظر گرفته می‌شود. تفسیر این کار این‌گونه است که برنامه برای کسب سرویس بیشتر به عقب باز می‌گردد یا توسط برنامه جدید دیگر تحت رقابت قرار می‌گیرد و جایگزین می‌گردد.

مقادیر باقیمانده را ساده تر می‌توان تفسیر نمود. احتمالات $P_۲, P_۳, \dots, P_n$ نشان می‌دهند که چه درصدی از درخواست‌های I/O به کدام دستگاه می‌رود. در فصل‌های قبل دیدیم که:

$$\sum_{i=۱}^{n} p_i = ۱ \qquad (۱-۱۲)$$

مقادیر $i = ۱,۲, \dots, n$ ، μ_i سرعت سرویس برای هر سرور موجود در این مدل را نشان می‌دهد. سرعت سرویس برابر است با تعداد دستورات اجرا شده تقسیم بر سرعت خام دستگاه. برای مثال، سرعت سرویس CPU به‌صورت زیر تعریف می‌شود:

$$\mu_۱ = CPU \text{ سرعت سرویس} = \frac{\text{دستور العمل های } CPU \text{ در کار}}{CPU \text{ در دستور العمل ها بر ثانیه سرعت}} \qquad (۲-۱۲)$$

در بیشتر تحلیل ها، فرض ما بر این است که دستورالعمل سرویس FCFS باشد و سرعت ورودی و سروی به‌صورت نمایی توصیف شوند و دستورالعمل صف بندی FCFS است. معیارهای کلیدی که ما سعی در کشف آن‌ها داریم عبارتند از توان عملیاتی و بهره برداری.

روش معمول برای محاسبه آن، استفاده از الگوریتم بوزن می‌باشد. این الگوریتم احتمال مربوط به داشتن تعداد مختلف کارها در سرورهای مختلف را در نقطه یک زمان تعیین می‌کند. تعداد کارها در هر گره مخصوص از باقیمانده گره‌های سیستم‌ها مستقل نیستند، برای اینکه تعداد کل در سیستم باید ثابت بماند. این الگوریتم اثبات می‌کند که احتمال گسترش کارها پیرامون سرورها در یک توزیع خاص برابر است با:

$$Prob\ (k_۱, k_۲, \dots, k_m) = \left(۱ \,/\, G\ (K)\right) \prod_{i=۲}^{M} \left(\mu_۱\ (P_i \,/\, \mu_i)\right)^{k_i} \qquad (۳-۱۲)$$

با کمک این ویژگی و قانون Little می‌توانیم برخی از معیارهای مورد نظر را به‌صورت زیر محاسبه کنیم:

$\rho_۱ = CPU$ کارکرد

$= G\ (K - ۱) \,/\, G\ (K)$

$\rho_i =\ I \,/\, O$ کارکرد دستگاه های

$= \rho_i\ \mu_i\ (P_i \,/\, \mu_i) for\ i = ۲, ۳, \dots, M$

$\lambda =$ توان عملیاتی سیستم (۴-۱۲)

کار در واحد زمان $= P_۱\rho_۱\ \mu_۱\ jobs$

متوسط کار در گره $E\ (K_i) = i$

$$= \sum_{n=۱}^{K} \left(\mu_۱\ (P_i\ /\ \mu_i)\right)^n \frac{G\ (k-n)}{G\ (k)} \quad for\ i = ۱, ۲, \dots, M$$

یکی از مهم‌ترین مؤلفه‌های الگوریتم بوزن، تابع $G(k)$ است ($k\ =\ ۱,۲,۳, \dots, M$). برای انجام این محاسبات یک تابع جزئی $g(k,m)$ تعریف می‌شود. جزئیات این تابع در فصل ۷ آمده است. برخی از جنبه‌های مهم محاسبه $G(k)$ را تکرار می‌کنیم:

$$g\ (k, ۱)\ ۱\ for\ k = ۰, ۱, ۲, \dots, k$$

$$g\ (۰, m) = ۱\ for\ m = ۱, ۲, \dots, M$$

$$g\ (k, m) = g\ (k\ , m-۱) + \left(\mu_۱\ (P_m\ /\ \mu_m)\ g\ (k-۱\ , m)\right) \qquad (۵-۱۲)$$

$$G\ (k) = g\ (k, M)\ for\ k = ۰, ۱, \dots, k$$

جزئیات این راه‌حل را می‌توانید در [۹] پیدا کنید. به‌عنوان مثالی از عملیات این الگوریتم، می‌توانیم برخی از مقادیر را برای یک سیستم آزمون تنظیم کنیم. اگر فرض کنیم که یک سیستم با یک CPU و سه دستگاه I/O داریم، M = ۴ می‌شود. می‌توانیم سرعت سرویس را برای دستگاه‌های I/O

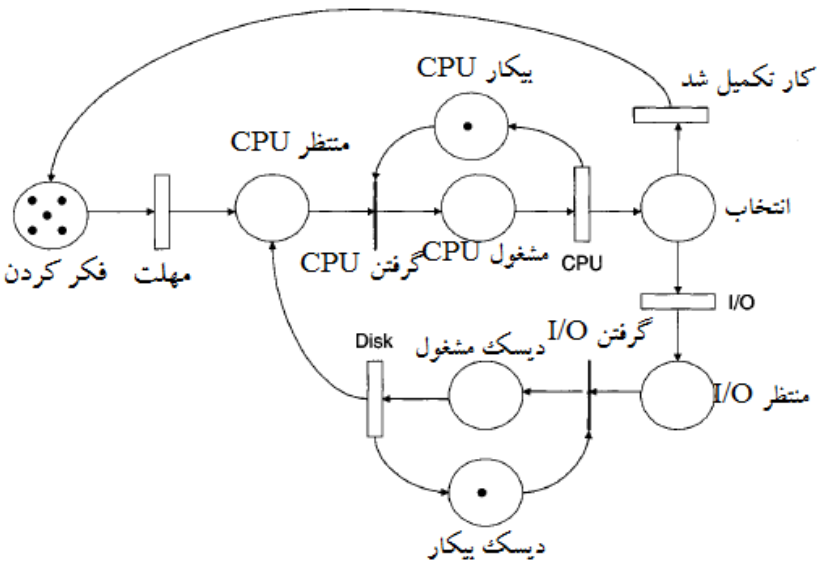
به‌صورت $\mu_2 = \mu_3 = \mu_4 = 10$ I/O در ثانیه تنظیم کنیم. CPU در $\mu_1 = 18.33$ برش کوانتومی در هر کار تنظیم می‌شود. اگر $K = 8$ کار در گردش سیستم باشد و آن را در الگوریتم بوزن اجرا کنیم، نتایج نمودار زیر را پیدا کنیم.

درخواست توان /ثانیه	بهره برداری دیسک	بهره برداری از CPU	سرعت I/O دیسک	زمان CPU در هر درخواست
۱,٦	۰,۵۳	۰,۹۶	۱۰	۰,۶
۲,۲۳	۰,۷۴	۰,۶۷	۱۰	۰,۳
۱,٦٦	۰,۳۷	۱,۰۰	۱۵	۰,۶
۲,۹۲	۰,۶۵	۰,۸۸	۱۵	۰,۳

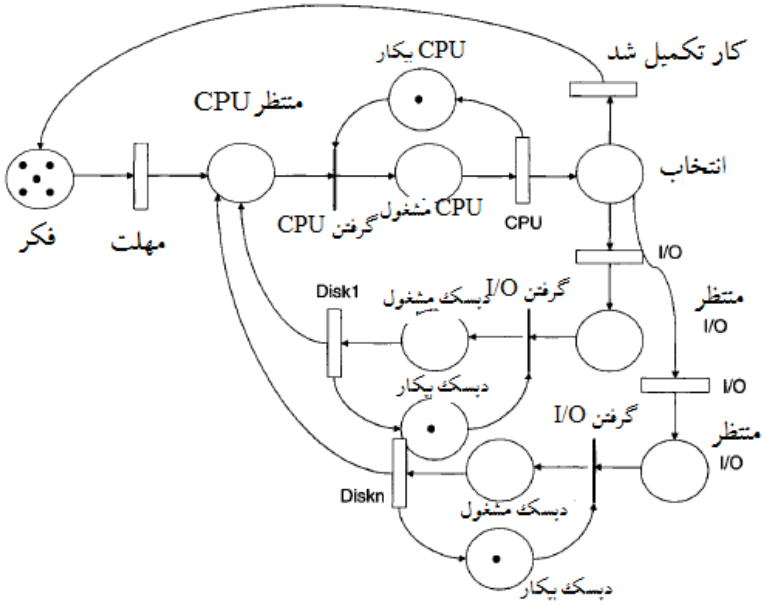
می توانیم به جای مدل صف بندی از مدل شبکه پتری یک تحلیل مشابه ایجاد کنیم. شکل (۱۲-۲) یک مثال از شبکه پتری برای سیستم دستگاه I/O نشان دهیم (شکل ۱۲-۱).

در این مدل، کارها از طریق مجموعه ای از پایانه‌ها ارسال می‌شوند که توسط محل فکر و انتقال نشان داده می‌شوند. هنگامی که یک کار درخواست سرویس کند، وارد محل انتظار CPU می‌شود و منتظر می‌شود تا CPU در دسترس قرار گیرد. بعد از دسترسی، کارهای منتظر CPU را به دست می‌آورند و با حرکت در مکان CPU، مدل‌ها از آن استفاده می‌کنند و به دنبال آن انتقال CPU رخ می‌دهد که سیکل اجرای CPU را مدل‌سازی می‌کند. بعد از اینکه یک استفاده از CPU توسط یک کار به اتمام رسید، وارد مکان انتخاب می‌شود. در این مکان، باید تصمیم گرفته شود که آیا کار انجام‌شده یا به I/O بیشتری نیاز است. اگر تکمیل شده باشد، نشانگر کار به منطقه فکر باز می‌گردد و تبدیل به کار

جدید بعدی می‌شود. اگر به I/O نیاز باشد، نشانگر به مکان انتظار دیسک می‌روند و منتظر منبع دیسک می‌شود. زمانی که منبع آزاد، کار دیسک را به دست می‌آورد و با حرکت در فضای دیسک آن را مدل‌سازی می‌کند و در ادامه انتقال دیسک انجام می‌شود. پس از اتمام استفاده از دیسک، نشانگر (کار) دوباره وارد محل انتظار CPU می‌شود تا CPU را به دست آورد. اگر بخواهیم که دیسک درایوهای بیشتری را مدل‌سازی کنیم، می‌توانیم شاخه‌های بیشتری را از محل انتخاب ایجاد کنیم و حلقه را برای درایو دیسک تکرار کنیم (شکل ۱۲-۳).



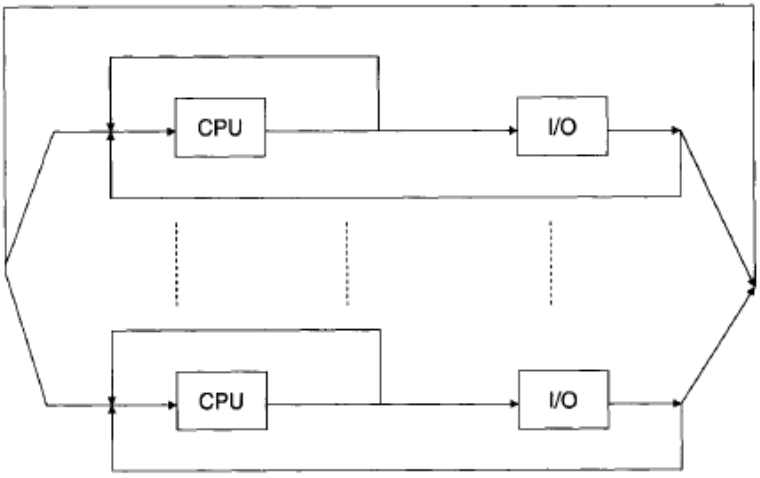
شکل (۱۲-۲): شبکه پتری سرور مرکزی



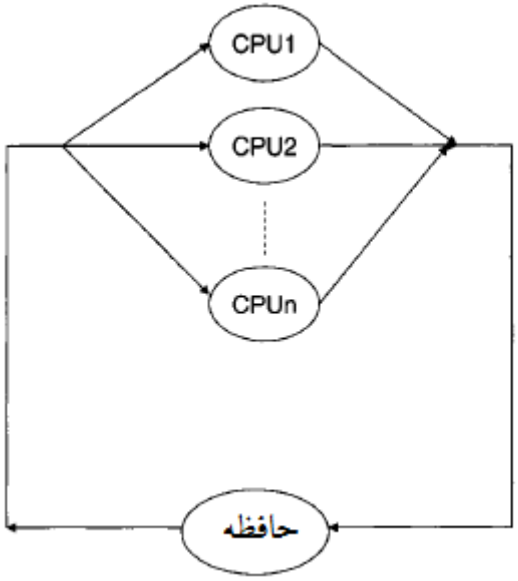
شکل (۱۲-۳): نمونه ای از چند دیسک شبکه پتری

اگر انتقال‌ها را به این صورت تنظیم کنیم: term۰,۱، CPU ۱,۰ و دیسک در ۰,۸، و مکان‌ها در ابتدا با نشانگرها به این صورت تنظیم شوند: think = ۶، CPU_idl =۱ و Disk_idl = ۱، می‌توانیم تحلیل را انجام دهیم. اگر این مدل محصور و زنده باشد، اولین تحلیل را باید تعیین نمود. یعنی باید بگوییم که هیچ‌گونه حالت بن بستی وجود ندارد و شبکه طوری پیکربندی شده است که بتوانیم اخراج‌ها را شروع کنیم. با آزمایش این شبکه، می‌توانیم ببینیم که چهار مسیر عمده در شبکه وجود دارد. Flow۱=think، waitCPU، Use CPU و Choose. اگر think = ۲ Flow، waiteCPU، Use CPU، Choose، diskwait و use disk. Flow۳=CPUidle، useCPU .Flow۴=diskidl، use disk. مسیر اول مربوط به کاری است که از CPU استفاده

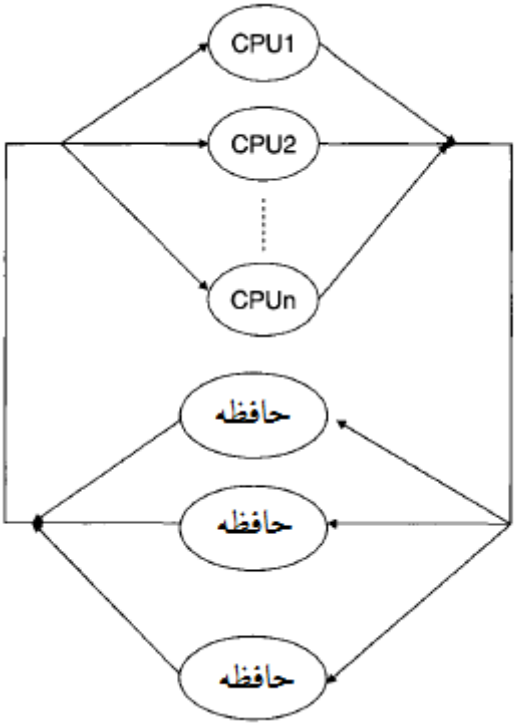
می‌کند و بدون نیاز به I/O کار را تکمیل می‌کند. مسیر دوم نیز کاری را نشان می‌دهد که از CPU استفاده می‌کند اما به دستگاه‌های I/O نیاز دارد و از آن‌ها استفاده می‌کند. حلقه مسیر سوم نشان دهنده سیکل CPU در حالت مشغول و بیکاری است و مسیر آخر نشان دهنده سیکل دیسک در زمان مشغول و بیکاری است. از آنجایی که مکان شبکه پتری تحت پوشش این مسیرها قرار دارد، شبکه محصور می‌شود. این این نشان می‌دهد که این شبکه یک گراف دسترسی محدود دارد و بنابراین، زنجیره مارکوف دارای یک فضای حالت محدود است. با استفاده از این مسیرها می‌توانیم زمان میانگین را برای کارها در مسیر از طریق هر حلقه با اجرای این مدل نشان دهیم. اگر این مدل را با همان اطلاعات موجود در مدل صف بندی اجرا کنیم، به همان نتایج قبل دست پیدا می‌کنیم. جزئیات بیشتر مربوط به این مدل و مدل‌های دیگر را می‌توانید در [۱۰] پیدا کنید.



شکل (۱۲-۴): مدل چندپردازنده ای با استفاده از پردازنده مرکزی



شکل (۱۲-۵) (الف)): مدل حافظه مشترک



شکل (۱۲-۵ (ب)) : مدل حافظه مشترک چند بانکی

۱۲-۳ مورد دوم: سیستم کامپیوتری چندپردازنده ای

تحلیل کلاسیک دیگر مربوط به چندپردازنده است. در این مورد می‌توانیم مدل سرور مرکزی را با مثال‌های متعددی از همان مدل جایگزین کنیم (شکل ۱۲-۴) یا می‌توانیم پیاده سازی‌های بیشتری داشته باشیم – برای مثال، مدل حافظه مشترک (شکل (۱۲-۵ (الف))) یا مدل حافظه مشترک چندبانکه (شکل (۱۲-۵ (ب))).

تحلیل هر یک از این معماری‌ها از روشی مشابه روش تک CPU استفاده می‌کند که پیش از این توصیف نمودیم. سیستمی که ما برای مدل‌سازی انتخاب کردیم، سیستم چندپردازنده ای است. این بیشتر نشان دهنده سیستم‌های واقعی است که در آن‌ها چند سرور به یکدیگر متصل می‌شوند تا به کاربران سرویس رسانی کنند. اگر تصمیم بگیریم سیستم شکل (۱۲-۵ (ب)) را مدل‌سازی کنیم، با مشکل تخصیص حافظه به پردازنده‌ها مواجه می‌شویم. یک پردازنده می‌توان تمام حافظه را در اختیار بگیرد یا حافظه ای در میان نباشد. تخصیصات با استفاده از کل ماژول حافظه انجام می‌شود. یعنی، یک CPU نمی‌تواند یک ماژول حافظه را با CPU دیگر در حین یک سیکل به اشتراک بگذارد. در هر سیکل CPU، هر پردازنده یک درخواست حافظه انجام می‌دهد. اگر برای برآوردن نیاز CPU، حافظه آزاد وجود داشته باشد، آن را پر می‌کند، در غیر این صورت، CPU باید منتظر سیکل بعدی بماند. هر ماژول حافظه ای که برای آن یک درخواست دسترسی به حافظه وجود دارد تنها می‌تواند به یک درخواست پاسخ دهد. هنگامی که چند پردازنده از یک ماژول حافظه درخواست حافظه داشته باشند، تنها یکی از آن‌ها موفق می‌شود (انتخاب درخواست‌های آن‌ها تصادفی است). پردازنده‌های دیگر همان درخواست ماژول را از سیکل بعدی می‌کنند. درخواست حافظه بعدی با استفاده از یک توزیع یکنواخت، برای هر پردازنده به‌صورت تصادفی از M ماژول حافظه انجام می‌شود. فرض می‌کنیم که وضعیت سیستم برابر با تعداد درخواست‌های حافظه برای هر ماژول حافظه باشد:

$$K = (k_1, k_2, k_3, \dots k_m) \qquad (۱۲-۶)$$

که در آن k_i نشان دهنده درخواست حافظه توسط پردازنده‌ها برای بانک حافظه i باشد. در آغاز یک سیکل مجموع تمام درخواست‌ها نمی‌تواند بیش از تعداد پردازنده‌ها در این سیستم باشد، N:

$$k_۱ + k_۲ + k_۳ + \cdots + k_m = N \qquad (-۱۲$$

۷)

تعداد کل حالات احتمالی مربوط به تعداد روش‌های N درخواست پردازنده است که می‌توانند روی M ماژول حافظه توزیع شوند:

$$\binom{M+N-۱}{N} = \binom{M+N-۱}{M-۱} = \frac{(M+N-۱)!}{(M-۱)!N!} \qquad -۱۲)$$

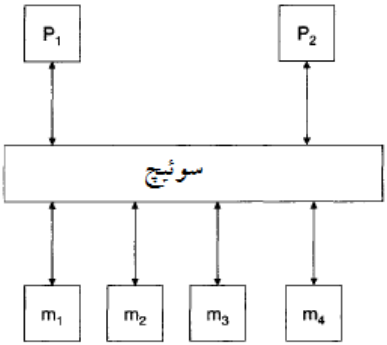
۸)

یا به عبارت دیگر، نحوه تخصیص N شئ به M خانه.

برای $N = ۲$ و $M = ۴$ (شکل ۶-۱۲) روش احتمالی برای تخصیص ۴ ماژول حافظه به پردازنده‌ها (که از یکدیگر غیرقابل تمایزند) در جدول (۱-۱۲) نشان داده‌شده است و توسط رابطه زیر مشخص می‌شود.

$$\frac{(M+N-۱)!}{(M-۱)!N!} = \frac{۵!}{۳!۲!} = \frac{۵*۴}{۲} = ۱۰ \qquad -۱۲)$$

۹)



شکل (۱۲-۶): سیستم چندپردازنده ای با $N = ۲$ و $M = ۴$

می بینیم که اگر تعداد پردازنده‌هایی که ماژول حافظه درخواست می‌کنند و تعداد ماژول‌های حافظه افزایش یابد، تعداد حالات احتمالی رشد سریعی خواهد داشت و تحلیل آن حتی برای مسائل نسبتاً کوچک بسیار دشوار خواهد بود (جدول ۱۲-۲).

جدول (۱۲-۱): روش‌های احتمالی برای تخصیص حافظه

	حافظه			
	۱	۲	۳	٤
۱	۰	۰	۰	**۲**
۲	۰	۰	۱	**۱**
۳	۰	۰	۲	۰
۴	۰	۱	۰	**۱**
۵	۰	۱	۱	۰
۶	۰	۲	۰	۰
۷	۱	۰	۰	**۱**
۸	۱	۰	۱	۰
۹	۱	۱	۰	۰

۱۰	۲	۰	۰	۰

جدول (۱۲-۲): تعداد حالات برای تعداد منتخبی از پردازنده‌ها و ماژول‌های حافظه

N	M	# States
۲	۴	۱۰
۳	۵	۳۵
۴	۷	۲۱۰

فرض کنید $H = (h_1, h_2, \dots, h_m)$ نشان دهنده حالت میانه باشد، هنگامی که دسترسی به حافظه مورد نظر در یک سیکل میسر نباشد و درخواست‌های جدید ایجاد نشده باشند:

$$h_i = \begin{cases} k_i - 1 & if\ k_i > 0, \forall i \\ 0 & if\ k_i = 0 \end{cases} \quad (12-10)$$

فرض می‌کنیم G نشان دهنده یک حالت جدید از سیستم باشد:

$$G = (g_1, g_2, g_3, \dots, g_m) \quad (12-11)$$

ابتدا عبارت زیر را تعریف می‌کنیم:

$d_i = g_i - h_i$ تعداد درخواست های جدید (۱۲-۱۲)

$$x = \sum d_i$$

نکته: وضعیت G را می‌توان از وضعیت K در یک سیکل به دست آورد اگر و تنها اگر به ازای هر مقدار از i، di >= ۰ باشد.

۱-۳-۱۲ ویژگی ها

۱. اگر G از K در یک سیکل قابل دسترسی باشد، احتمال اینکه واقعاً وضعیت بعدی باشد برابر است با:

$$P\ (K\ ,G) = \frac{x!}{d_1!\ d_2!\ \ldots d_m!}\ \left(\frac{1}{m}\right)^x$$

(۱۳-۱۲)

که در آن x نشان دهنده تعداد درخواست‌های جدید است.

۲. این سیستم را می‌توان توسط زنجیره مارکوف توصیف کرد، برای اینکه احتمالات وضعیت بعدی در هر زمانی تنها به حالت فعلی بستگی دارد.

۳. این سیستم نامتناوب است، برای اینکه یک انتقال تک مرحله ای از یک حالت به خودش در هر زمانی میسر است.

۴. سیستم غیرقابل کاهش است برای اینکه می‌توان در طی چند مرحله محدود به دیگری برسد.

از این رو، از آنجایی که این یک فرآیند با وضعیت محدود است، این می‌تواند یک فرآیند مارکوف ارگوتیک نیز باشد. علاوه بر این، از آنجایی که این شرایط حفظ می‌شوند، یک توزیع احتمال وضعیت متعادل نیز وجود دارد، Π، به نحوی که:

(۱٤-۱۲)

$$\underset{\sim}{\prod} = \underset{\sim}{\tilde{\prod}} P$$

که در آن P ماتریس انتقال وضعیت است (در فصل ۶ و ۷ توضیح گفته شد):

$$\underset{\sim}{\prod} = \left(\prod_{۱}, \prod_{۲}, \prod_{۳}, \prod_{۴}, \dots, \prod_{j}\right) \qquad (۱۵-۱۲)$$

یک ارزیابی عملکرد معمولاً در این نوع پیکربندی سیستم ایجاد می‌شود تا نوع توان پردازنده مؤثر N پردازنده با M سیستم حافظه را به‌صورت تعیین نماید:

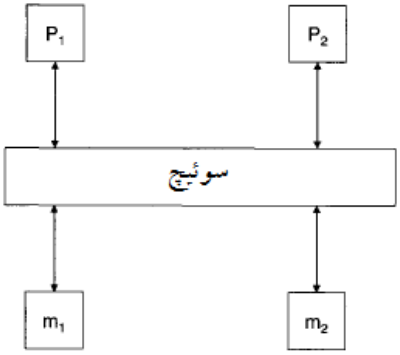
$EP\ (N\ ,M) =$ تعداد مورد انتظار از دستور العمل های اجرایی در هر ثانیه

$N = ۱, M = ۱$ نسبت به یک سیستم (۱۶-۱۲)

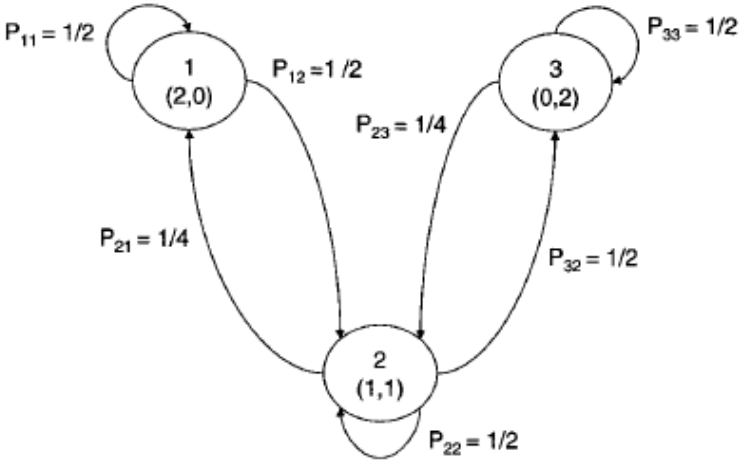
فرض می‌کنیم $Proc(i)$ نشان دهنده تعداد درخواست برای حافظه (دستورات اجرا شده) در زمانی که سیستم در وضعیت i باشد:

$$\therefore EP\ (N,M) = \sum_{i=۱}^{j} Proc\ (i)\ \prod i \qquad (۱۷-۱۲)$$

برای مورد ساده ای که در آن $N = ۲$ و $M = ۲$، سیستمی داریم که در شکل (۱۲-۷) نشان داده‌شده است.



شکل (۱۲-۷): سیستم چندپردازنده ای با $N = ۲$ و $M = ۲$



شکل (۱۲-۸): نمودار انتقال وضعیت احتمالی

وضعیت‌های احتمالی این مدل که نشان دهنده درخواست از حافظه توسط دو پردازنده است به‌صورت زیر توصیف می‌شود (شکل ۱۲-۸):

$State\ (k_1, k_2)$:

۱ (۲, ۰)

۲(۱, ۱)

$$j = \binom{N + M - ۱}{N} = \binom{۳}{۲} = \frac{۳!}{۲!۱!} = ۳ \qquad (۱۲-$$

۱۸)

۳ (۰, ۲)

با استفاده از فرمول عمومی زیر:

$$P\ (K\ ,G) = \frac{x!}{d_۱!\ d_۲!...d_m!} \left(\frac{۱}{m}\right)^x \qquad (۱۲-$$

۱۹)

$$P\ \big((۲,۰),(۱,۱)\big) = \left(\frac{۱!}{۰!۱!}\right) \left(\frac{۱}{۲}\right)^۱ = \frac{۱}{۲} \qquad (۲۰-۱۲)$$

که احتمال بودن در وضعیت (۲, ۰) و انتقال به وضعیت (۱ ,۱) را نشان می‌دهد. به طور مشابه، احتمال بودن در وضعیت (۱ ,۱) و چرخش به وضعیت (۲, ۰) به‌صورت زیر است:

$$P\ \big((۱,۱),(۲,۰)\big) = \left(\frac{۱!}{۲!۰!}\right) \left(\frac{۱}{۲}\right)^۲ = \frac{۱}{۴} \qquad (۲۱-۱۲)$$

معادلات توازن برای این زنجیره مارکوف را می‌توان با استفاده از روابط زیر پیدا کرد:

جریان خروجی = جریان ورودی (۲۲-۱۲)

مرحله یک: $۱/۲ \prod_{۱} + ۱/۴ \prod_{۲} = \prod_{۱}$

جریان خروجی = جریان ورودی (۱۲-۲۳)

مرحله دو: $۱۲ \prod_{۳} + \prod_{۲} = \prod_{۳}$

با حل معادلات همزمان داریم:

$$۲\prod_{۱} + \prod_{۲} = ۴\prod_{۱}$$

$$۲\prod_{۳} + \prod_{۲} = ۴\prod_{۳}$$

$$\therefore \prod_{۲} = ۲\prod_{۱} \quad (۱۲-۲۴)$$

$$\prod_{۲} = ۲\prod_{۳}$$

$$\prod_{۱} = \prod_{۳} = ۱/۲ \prod_{۲}$$

و از آنجایی که:

$$\prod_{۱} + \prod_{۲} + \prod_{۳} = ۱ \quad (۱۲-$$

۲۵)

درنتیجه:

$$\prod_{۱} = ۰.۲۵$$

$$\prod_{۲} = ۰.۵۰ \qquad (۲۶-۱۲)$$

$$\prod_{۳} = ۰.۲۵$$

توان پردازنده مؤثر اکتشافی با استفاده از رابطه زیر محاسبه می‌شود:

$$EP\ (۲,۲) = \sum_i \prod_i Proc\ (i) \qquad (-۱۲$$

۲۷)

$i \quad Proc\ (i)\ i$: تعداد دستورات اجرا شده در وضعیت

I ۱

۲ ۲

۳ ۱

∴

$$EP\ (۲,۲) = ۱\ \prod_{۱} + ۲\ \prod_{۲} + ۱\ \prod_{۳} = ۰.۲۵ + ۱.۰ + ۰.۲۵ = ۱.۵ \qquad (۲۸-۱۲)$$

جدول (۱۲-۳): خلاصه افزایش سرعت برای M حافظه و N پردازنده

M	N پردازنده			
ماژولهای حافظه	۲	۳	۴	۵
۲	۱,۵	–	–	–
۳	۱,۶۶۷	۲,۰۴۸	–	–
۴	۱,۷۵۰	۲,۲۶۹	۲,۶۲	–
۵	۱,۸۰۰	۲,۴۰۹	۲,۸۶۳	**۳,۱۹۹**
۶	۱,۸۳۳	۲,۵۰۵	۳,۰۳۶	**۳,٤٥۳**
۷	۱,۸۵۷	۲,۵۷۵	۳,۱۶۶	**۳,٦٤۸**
۸	۱,۸۷۵	۲,۶۲۷	۳,۲۶۵	**۳,۸۰۱**
۹	۱,۸۸۹	۲,۶۶۸	۳,۳۴۴	**۳,۹۲۵**
۱۰	۱,۹۰۰	۲,۷۰۱	۳,۴۰۷	**٤,۰۲۵**

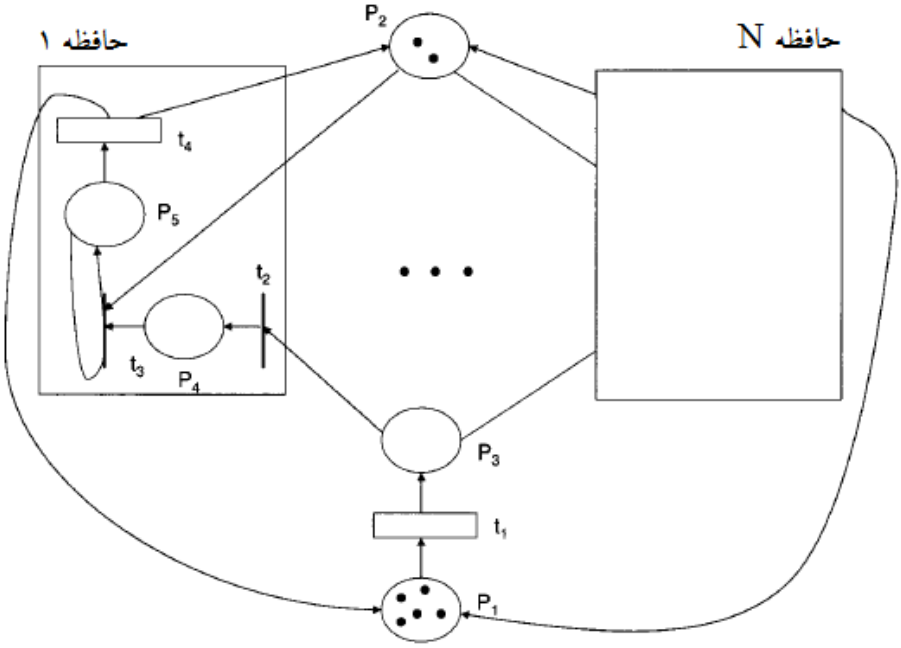
نتایج M ماژول حافظه (۲≤M≤۱۰) و N پردازنده (۲≤N≤۵) به‌صورت خلاصه در جدول (۱۲-۳) آمده است.

محدودیت ها: این مدل رابط حافظه ناشی از عملیات I/O را در نظر نمی‌گیرد. این این مدل فرض می‌کند که پردازنده‌ها و حافظه هماهنگ باهم باشند.

۱۲-۴ مورد سوم: مثال شبکه پتری

از دیدگاه یک شبکه پتری نیز به همان مسئله نگاه می‌کنیم. در این مورد چند فرضیه ایجاد می‌نماییم. فرض کنیم n_p تعداد پردازنده‌های ما، n_m تعداد ماژول حافظه مشترک و n_b گذرگاه داده است. در تحلیل نظری قبل از گذرگاه داده چشم پوشی می‌کنیم. هر یک از پردازنده‌ها حافظه محلی دارند که تا زمان از بین رفتن یک صفحه از آن استفاده می‌شود. در این نقطه، به دسترسی به ماژول حافظه خارجی نیاز داریم که باعث می‌شود صفحه جدید در حافظه پردازنده محلی بارگذاری گردد. نرخ اتلاف به‌صورت نمایی توزیع و در $1/\lambda$ تنظیم می‌شود. زمان دسترسی به حافظه مشترک را نیز نمایی فرض می‌کنیم یعنی، $1/\mu$. اگر در همان ابتدا $n_p = 5$، $n_m = 3$ و $n_b = 2$ تنظیم کنیم، در شکل (۱۲-۹) می‌توانیم پیکربندی اولیه آن را ببینیم. این مدل شامل دو محل در هر ماژول حافظه (یک مکان برای نشانگرهای پردازنده و یک محل برای نشانگرهای گذرگاه) و یک تبدیل زمان‌بندی شده (برای استفاده و تخصیص حافظه) است. علاوه بر این، دو تبدیل میانه مربوط به هماهنگ‌سازی و کنترل دسترسی به حافظه نیز وجود دارد. برای مدل اندازه ما ۹ محل کلی، ۴ تبدیل زمان‌بندی شده و ۶ تبدیل میانه فرض می‌کنیم. نشانگرهای موجود در مکان P_2 نشان دهنده گذرگاه‌های داده موجود برای استفاده هستند. نشانگرهای موجود در محل P_1 پردازنده‌هایی را نشان می‌دهند که روی حافظه محلی خود اجرا می‌شوند. یکی از مفروضات مهم این مدل این است که هر پردازنده و ماژول حافظه به شیوه ای یکسان رفتار می‌کند. هنگامی که یک پردازنده برای دسترسی به حافظه محلی رقابت می‌کند (یک خطا یا اتلاف صفحه ناشی در نتیجه گذار یا انتقال t_1 دارد) و به منابع حافظه مشترک بیشتری نیاز دارد، یک نشانگر از

مکان P_1 به P_3 حرکت می‌کند. یک پردازنده تعیین می‌کند که با خروج تبدیل فوری t_2 در ماژول حافظه ای که با استفاده از شاخه احتمالی انتخاب کرده است، به کدام حافظه نیاز دارد.



شکل (۱۲-۹): مدل شبکه پتری برای سیستم چندپردازنده ای

بعد از خروج t_2، یک نشانگر از مکان ۳ به مکان ۴ حرکت می‌کند. هنگامی که یک نشانگر در مکان ۴ باشد، پردازنده درخواست دسترسی به گذرگاه داده را می‌فرستد. از گذرگاه برای اتصال پردازنده به ماژول حافظه استفاده می‌شود. این پردازنده حافظه مطلوب را کسب می‌کند و سپس یک گذرگاه داده برای بازیابی اطلاعات موردنیاز به دست می‌آورد. هنگامی که پردازنده به گذرگاه دسترسی یابد، با

خروج انتقال t_3 علامت گذاری می‌شود و حافظه را به دست می‌آورد (که توسط نشانگر در محل نشان داده می‌شود، P_5) و با استفاده از ماژول حافظه با آغاز تایمر در انتقال t_4 شروع به مدل‌سازی می‌کند. بعد از اتمام کار با گذرگاه، نشانگر پردازنده و گذرگاه به مکان‌های اولیه خود یعنی P_2 و P_1 باز می‌گردند.

اگر ما این مدل را با ورودی‌های مشابه به چیزی که پیش از این به آن مدل صف بندی می‌گفتیم اعمال کنیم، نتایجی به دست می‌آوریم که بسیار مطابق با مورد مدل صف بندی می‌باشد. یعنی، متوجه می‌شویم که توان مؤثر پردازنده با پیکربندی مشخص، حدود ۲,۰۵ می‌باشد. اگر دسترسی متعادل باشد می‌توانیم این روند را بهبود بخشیم، این نشان می‌دهد که در هرلحظه هیچ پردازنده ای نمی‌تواند بیش از یک حافظه در اختیار داشته باشد. این باعث افزایش ظرفیت پردازنده ما تا حدود ۳,۲ می‌شود.

۵-۱۲ خلاصه

در این فصل برای تحلیل معماری کامپیوتر از ابزارهای تحلیلی که در فصول ۱ تا ۱۱ بیان کردیم، استفاده نمودیم. ما در ابتدا به یک سیستم کامپیوتری با سرور مرکزی پرداختیم که همانند بسیاری از سیستم‌های کامپیوتری رومیزی است و سپس به یک سیستم کامپیوتری چند سروره رسیدیم. ما این تحلیل‌ها را در ابتدا به‌صورت تحلیلی انجام دادیم و سپس از شبکه‌های پتری استفاده کردیم. این فصل به‌عنوان یک راهنما برای تحلیل معماری‌های کامپیوتری ارائه شده است و برای مقایسه معماری‌ها یا مؤلفه‌های مختلف نیست.

فصل سیزدهم

تحلیل اجزای سیستم‌عامل

این فصل از موضوعات زیر تشکیل شده است. بخش (۱۳-۱) مقدمه ای بر ارزیابی عملکرد خاص، مفاهیم اصلی آن، انواع تراکم بار مورداستفاده، طراحی آزمایش برای تحلیل عملکرد و مروری بر ابزار شبیه سازی مورداستفاده جهت ارزیابی دارد. بخش (۱۳-۲) شامل معماری چهار سیستم‌عامل مورداستفاده است. ما سعی کردیم که معماری‌های آن‌ها کاملاً مخصوص به آزمایش‌های ما باشند. بخش (۱۳-۳) مربوط به آمار، تحلیل نتایج آزمایش، تحلیل حساسیت، مشکلات مربوط به هزینه/عملکرد و ارائه در قالب گراف و نمودار است. بخش (۱۳-۴) درباره طرح آزمایشی و شبیه سازی بحث می‌کند. بخش (۱۳-۵) نیز نتیجه گیری تحلیل عملکرد ارائه می‌گردد.

۱-۱۳ مقدمه

کاربران سیستم‌های کامپیوتری، مدیران و طراحان همگی تمایل دارند که عملکرد را مورد ارزیابی قرار دهند، برای اینکه هدف آن‌ها، کسب یا تأمین بالاترین میزان عملکرد با حداقل هزینه است. ارزیابی عملکرد در هر مرحله چرخه عمر یک سیستم کامپیوتری ضروری است. این ارزیابی شامل طراحی، تولید، استفاده، ارتقاء و غیره است. ما باید این ارزیابی را برای پیش بینی کفایت سیستم انجام دهیم. برای انجام این ارزیابی عملکرد باید سیستم، مؤلفه‌های آن، وضعیت محیطی که سیستم در آن بسر می‌برد را

به‌دقت تعریف نماییم و پارامترهایی را تعریف کنیم که بر اساس آن‌ها سیستم ایجاد شده است. سیستم‌های کامپیوتری ستون فقرات یک سازمان محسوب می‌شوند و ممکن است کاربرانی در سراسر دنیا داشته باشند. اگر سیستم هدفی که برای آن در نظر گرفته شده را دنبال نکند، زیرساخت، بهره‌وری و اعتبار سازمان را به خطر می‌اندازد. بنابراین ارزیابی سیستم کامپیوتری از اهمیت بالایی برخوردار است. این نه‌تنها شامل عملکرد سخت‌افزاری/نرم افزاری می‌شود بلکه شامل معیار هزینه به عملکرد نیز می‌گردد. برای ارزیابی عملکرد سیستم کامپیوتری معیارهایی مثل پاسخ گویی، قابلیت مأموریت، قابلیت اطمینان و بهره‌وری اهمیت زیادی دارند. تکنیک‌های مختلفی برای ارزیاب عملکرد وجود دارد. می‌توانیم این تکنیک‌ها را در دو کلاس قرار دهیم. یک کلاس شامل طراحی یک آزمایش (HW/SW/Stimulus) است و دیگری شامل مدل‌سازی است که ممکن است تحلیلی (صف بندی، شبکه‌های پتری) یا با شبیه سازی باشد (گسسته، مستمر، ترکیبی). در این مطالعه از هر دو تکنیک برای مقایسه چهار سیستم‌عامل استفاده کردیم. این فصل عملکرد چهار سیستم‌عامل را مورد ارزیابی قرار می‌دهد (مایکروسافت ویندوز XP، ویندوز ME، ویندوز NT و لینوکس ۷,۲). ارزیابی عملکرد سیستم‌عامل توسط یک کلاس ارزیابی عملکرد سیستم‌های کامپیوتری ارتقاءیافته در UMass Dartmouth در ترم بهار ۲۰۰۲ انجام شد. ارزیابی عملکرد این سیستم‌عامل‌ها روی معماری x۸۶ انجام گردید. عملکرد سیستم‌عامل‌ها با استفاده از سه نوع تراکم بار مورد آزمایش قرار گرفت. این ارزیابی بر اساس نسخه‌های اصلی موجود از این سیستم‌ها است که بدون تنظیم عملکرد می‌باشند. از هر گروه خواسته شد تا مدل‌های سطح بالایی طراحی کنند و این مدل‌ها را با استفاده از ابزار شبیه سازی

AWESIM تبدیل نمایند. گروه‌ها با یک طرح آزمایش مشترک آمدند و انواع خاصی از آزمایش‌های کارایی را برای اندازه گیری پاسخ این چهار سیستم‌عامل انجام دادند. هر گروه یک تحلیل مقایسه ای انجام داد.

۱۳-۲ معماری‌های سیستم

برای تحلیل ارزیابی عملکرد سیستم‌های عامل، معماری سیستم‌های کامپیوتری نقش مهمی ایفا می‌کند. این بخش شامل معماری‌های سیستم‌عامل برای لینوکس ۷,۲، ویندوز ME، ویندوز XP ویندوز NT است.

۱۳-۲-۱ معماری لینوکس

Red Hat LINUX ۷,۲

از آنجایی که بخشی از تلاش‌ها برای ارزیابی عملکرد نسبی لینوکس نسبت به سیستم‌عامل‌های دیگر است، ما مؤلفه‌های کلیدی لینوکس را در نظر می‌گیریم تا بستری برای این مقایسه باشد. در اینجا ما خط‌مشی‌های لینوکس و پارامترهای مورداستفاده در آخرین نسخه ثابت Kernel ۲,۴ را در نظر می‌گیریم. توزیع مورداستفاده برای ارزیابی ما Red Hat LINUX ۷,۲ است. لینوکس از چندوظیفگی حمایت می‌کند یعنی توانایی اجرای همزمان چند برنامه. برای مثال، می‌توان یک دیسک را فرمت کرد، یک فایل از BBS دانلود نمود و در پردازنده word آن را ویرایش کرد، همه این کارها به طور همزمان انجام می‌شود.

ساختار وظیفه و جدول پردازش

لینوکس پردازش‌ها را در سیستم مدیریت می‌کند، هر پردازش توسط یک ساختمان داده $task\_struct$ نشان داده می‌شود. بردار وظیفه نیز آرایه‌ای از اشاره‌گرها به هر ساختمان داده $task\_struct$ موجود در سیستم است. به محض ایجاد پردازش‌ها، یک $task_struct$جدید از حافظه سیستم تخصیص می‌یابد و در $task\_vector$ اضافه می‌شود. ساختارهای $task\_struct$ به دو روش متصل می‌شوند: یک جدول هش (درهم سازی) که توسط پید[۷۸] خرد و درهم می‌شود و به‌صورت دایره‌ای، با استفاده از اشاره‌گرهای $P->prev\_task$ و $P->next\_task$ به‌صورت دوگانه به لیست متصل می‌شوند. این وظایف توسط مقدار pid خود درهم می‌شوند. از جدول هش برای پیدا کردن سریع یک وظیفه توسط pid داده‌شده استفاده می‌شود، $find\_task\_pid\ ()$.

فهرست مدور با پیوند دوگانه که از $P->prev\_task$ و $P->next\_task$ استفاده می‌کند به نحوی حفظ می‌شود که بتوان از تمام وظایف سیستم به راحتی عبور کرد. پرچم‌های وظیفه شامل اطلاعات مربوط به حالات پردازش است که ذاتا منحصربه فرد نیستند. برنامه ریز برای تصمیم گیری درمورد پردازشی که سیستم نیاز دارد، به اطلاعات نیاز دارد. هر فرآیند موجود در سیستم یک شناسه دارد. شناسه فرآیند یک شاخص در task_vector نیست، بلکه فقط یک عدد است. هر پردازش

[۷۸] -pid

شناسه‌های کاربری و گروهی دارد، از این شناسه‌ها برای کنترل دسترسی این پردازش به فایل‌های آن و دستگاه‌های موجود در سیستم استفاده می‌شود.

لینک ها

در یک سیستم لینوکس هیچ فرآیندی مستقل از دیگری نیست. هر فرآیند موجود در سیستم به جز فرآیند اولیه، init، یک فرآیند والد دارد. فرآیندهای جدید ایجاد نمی‌شوند، کپی می‌شوند یا حتی توسط فرآیندهای قبل تکثیر می‌شوند. هر task_struct نشان دهنده فرآیندی است که اشاره‌گرها را به والدین و بستگان (فرآیندهایی با یک فرآیند والد) و این به فرزندان خودشان نگه می‌دارد. علاوه بر این، تمام پردازش‌های موجود در سیستم در یک لیست با پیوند دوگانه نگه داشته می‌شوند که ریشه آن‌ها ساختمان داده task_struct فرآیند init است. این فهرست به هسته لینوکس اجازه می‌دهد تا به همه فرآیندهای موجود در سیستم نگاه کند. این فهرست به این کار نیاز دارد تا برای دستوراتی مثل ps یا kill پشتیبانی فراهم کند.

زمان و زمان‌سنج

هسته زمان ایجاد هر فرآیند و این زمانی که CPU در طول عمر خود صرف می‌کند را نگه می‌دارد. برای هر ضربه ساعت، هسته مقدار زمان موجود در jiffies که فرآیند فعلی در سیستم گذرانده و در وضعیت کاربر است را به روز رسانی می‌کند. لینوکس نیز از زمان سنج‌های فاصله مخصوص پردازش حمایت می‌کند، پردازش‌ها می‌توانند از فراخوان‌های سیستم برای راه اندازی تایمرها استفاده کنند تا در زمان انقضای تایمر، سیگنال را به خودشان بفرستند. این تایمرها ممکن است دوره ای یا تک ضربه باشند.

سیستم فایل

فرآیندها می‌تواند فایل‌هایی که می‌خواهند را باز و بسته کنند و $task\_struct$ این فرآیند حاوی اشاره گرهایی به توصیف کننده‌هایی به هر فایل باز و این اشاره گرها به دو گره $VFS\ i$ است. هر گره $VFS\ i$ به‌صورت یکنواخت یک فایل یا دایرکتوری را در یک سیستم فایل توصیف می‌کند و این یک رابطه یکنواخت برای سیستم‌های فایل پایه ایجاد می‌نماید. فیلد $P->fs$ حاوی اطلاعات سیستم فایل است که تحت لینوکس قرار دارد و سه بخش اطلاعاتی دارد: دایرکتوری ریشه d_entry و نقطه اوج، دایرکتوری ریشه متناوب d_entry و نقطه اوج و دایرکتوری کاری فعلی d_entry و نقطه اوج.

حافظه مجازی

بیشتر فرآیندها اندکی حافظه مجازی دارند (تهدیدات هسته و دومون‌ها ندارند) و هسته لینوکس باید بداند که حافظه مجازی چگونه در حافظه فیزیکی سیستم ترسیم شده است. فیلدهای $P->mm$ و $P->active\_mm$ به ترتیب به فضای آدرس فرآیند توصیف شده توسط ساختمان mm_struct و به فضای آدرس فعال اشاره می‌کنند (به‌عنوان‌مثال، تهدیدات هسته).

صفحه بندی

برای برآورد الگوریتم‌هایی که اخیرا برای جایگزینی صفحه استفاده شده اند (LRU)، لینوکس فرآیندی برای بیشتر صفحات NRU (که به تازگی استفاده نشده اند) پیدا کرده که صفحات را دور می‌کند. برخلاف یک الگوریتم ساعت استاندارد، که چند صفحه از تمام پردازش‌ها می‌گیرد، این الگوریتم تعداد زیادی صفحه از چند فرآیند اندک می‌گیرد. گاهی اوقات لینوکس با حذف موقتی بیشتر

فرآیندهای قربانی از مخزن فرآیندهای درحال اجرا سروکار دارد. هسته‌های مختلف لینوکس این جزئیات را به‌صورت متفاوت کنترل می‌کنند.

kernel ۲,٤

kernel ۲,۴ یک مصالحه بین افزایش عمر کرنل ۲,۰ و فقدان آن در نسخه ۲,۲ جدید ایجاد کرده است. این برنامه با تغییر روش کاهش عمر این کار را انجام می‌دهد. عمر به‌صورت نمایی مخالف حالت خطی کاهش می‌یابد. این مانع این می‌شود که یک فرآیند با نرخ خطای بالای صفحه بیشتر از سهم صفحات بگیرد و درنتیجه به فرآیندهای دیگر صدمه بزند و اجازه نمی‌دهد صفحه ای که تنها یک بار رجوع شده همان زمان انتظاری را داشته باشد که یک صفحه با ۲۰ بازدید دارد.

برای اینکه از حافظه به‌صورت موثرتری استفاده شود، کرنل ۲,۴ روش مورداستفاده در کرنل ۲,۰ را برای انتخاب فرآیندها معرفی می‌کند تا در صفحات NRU مشارکت داشته باشند. با عبور از یک فهرست فرآیند در هر لحظه، تنها حدود ۶٪ فضای آدرس در هر فرآیند جستجو می‌شود تا پیش از اینکه صفحات NRU به فرآیند بعد بروند، جستجو انجام شود. مشابه کرنل ۲,۰، این روش احتمال thrashing را افزایش می‌دهد.

۱۳-۲-۲ معماری ویندوز XP

ویندوز XP حرفه ای بر اساس کد ثابت ویندور NT ویندوز ۲۰۰۰ ساخته شده است که یک معماری ۳۲ بیتی و یک مدل حافظه کاملاً حفاظت شده دارد. ویندوز XP حرفه ای به این منظور

طراحی شده تا امکان اجرای همزمان به چند برنامه کاربردی داده شود وضعیت پاسخگویی و ثبات سیستم را تضمین نماید.

مدیریت دیسک

مایکروسافت ویندوز XP دو نوع ذخیره سازی دیسک دارد: بیسیک (ایستا) و پویا.

مخزن دیسک اصلی (بیسیک)

به دیسکی که برای ذخیره سازی اولیه مقدار دهی شده، دیسک اصلی می‌گویند. یک دیسک اصلی حاوی حجم اصلی، مثل پارتیشن‌های اولیه، پارتیشن‌های پیشرفته و درایوهای منطقی است. علاوه بر این، حجم‌های اصلی شامل حجم چند دیسک است که با استفاده از ویندور NT ۴,۰ یا قبل تر ایجاد شده اند، مثل مجموعه‌های Volume، مجموعه‌های Stripe، مجموعه‌های Mirrored و مجموعه Stripe با بیت توازن. ویندور XP از این ظرفیت پایه چنددیسکی پشتیبانی نمی‌کند.

ذخیره سازی دیسک پویا

به دیسکی که برای ذخیره سازی پویا مقدار دهی می‌شود، دیسک پویا می‌گویند. یک دیسک پویا حاوی حجم های پویا است مثل حجم های Simple (حجم Simple، فضای خالی روی یک دیسک داینامیک است، متداول ترین روش که تقریبا شبیه Primary Partition است و در صورت استفاده از یک هارددیسک، فقط میتوان simple volume داشت. توجه داشته باشید که یک Volume در صورت Active یا Boot بودن حتما simple می‌باشد.)، حجم های Spanned (در این روش اطلاعات به ترتیب در دیسک ها ذخیره می‌شود. ابتدا یک دیسک پر می‌شود، سپس دیسک بعدی و

این عمل تا وقتی‌که کل فضاهای ذخیره سازی به طور کامل پر شود، ادامه پیدا میکند. هدف از این افزایش حجم، فضای ذخیره سازی و دسترسی به کل آن از طریق یک volume است.) به مفهوم دیگر یکی کردن چند هارد دیسک). می‌تواند در بر گیرنده ۳۲ هارددیسک باشد و برای ایجاد این گونه ازVolumeها باید ازVolumeهای NTFS استفاده کنیم). حجم های Stripped (یا RAID۰ ، باید حداقل دو هارددیسک داشته باشیم، این Volume از کوچک‌ترین فضای خالی دیسک ها به‌صورت مساوی استفاده می‌کند، یعنی فضای قابل استفاده در هر هارددیسک محدود می‌شود به حجم کوچک ترین هارددیسک. پس در این روش بهتر است که هارددیسک‌ها، اندازه هم باشند تا هیچ مقداری از فضای هارددیسک بی استفاده نماند. در این روش، اطلاعات به‌صورت قطعه‌های ۶۴ کیلوبایتی در هر دیسک، به‌صورت Striped یا نواری قرار می‌گیرد. خواندن و نوشتن اطلاعات هم به‌صورت قطعه های ۶۴ کیلوبایتی است. این باعث شده که سرعت بالایی داشته باشیم و بیشترین کارایی را در بین انواع Volume داشته باشد.)، حجم های Mirrored و حجم های RAID-۵ توسط ذخیره سازی پویا، مدیریت دیسک و ظرفیت را می‌توان بدون نیاز به راه اندازی مجدد ویندوز انجام داد. حجم های Mirrored (ظرفیت Mirrored یا RAID۱، در این روش به دو هارد دیسک احتیاج داریم. دیسکهایی با حجم یکسان و ترجیحاً مشخصات یکسان. مطابق نظر مایکروسافت، حتی بهتر است مدل و سازنده آن‌ها هم یکسان باشد. اطلاعات همزمان در دو دیسک ذخیره می‌شود، پس مسلما سرعت خوانده شدن اطلاعات بیشتر و سرعت نوشته شدن کمتر میشود. پس بهترین گزینه برای کنترل خطاست.) یا حجم های RAID-۵ (فقط روی ویندوز سرور قابلیت استفاده دارد، نیاز به حداقل ۳ هارددیسک

دارد، مانند حالت Striped عمل میکند، با این تفاوت که اطلاعاتی با نام parity همراه با این قطعه های ۶۴ کیلوبایتی در دیسک ها ذخیره می‌شود تا اگر یکی از دیسک ها آسیب ببیند، سیستم با استفاده از اطلاعات باقی مانده و parity، اطلاعات دیسک آسیب دیده را بازسازی می کند.) را نمی‌توان روی کامپیوترهایی با سیستم‌عامل ویندوز XP حرفه ای ایجاد کرد. بااین‌حال، از یک کامپیوتر با سیستم‌عامل حرفه ای XP می‌توان برای ایجاد یک حجم Mirrored یا حجم RAID-۵ استفاده نمود که روی ویندوز سرور ۲۰۰۰، ویندوز سرور پیشرفته ۲۰۰۰ یا ویندوز سرور مرکز داده ۲۰۰۰ اجرا می‌شوند. انواع ذخیره سازی با نوع سیستم فایل مجزا هستند. یک دیسک پایه یا پویا می‌تواند شامل ترکیبی از پارتیشن‌ها یا حجم‌های FAT۱۶، FAT۳۲ یا NTFS باشد.

سیستم‌های فایل

ویندوز XP از سه سیستم فایل مختلف پشتیبانی می‌کند: جدول تخصیص فایل (FAT)، FAT۱۶، FAT۳۲ و NTFS (سیستم فایل NT)، NTFS سیستم فایل توصیه شده است. NTFS دارای ویژگی‌های پیشرفته‌ای از سیستم فایل است مثل امنیت، عملیات تراکنشی، حجم‌های بزرگ و روی حجم‌های بالا عملکرد بهتری دارد. چنین قابلیت‌هایی را نمی‌توان در FAT۱۶ یا FAT۳۲ مشاهده نمود. ویندوز XP برای حجم‌های NTFS در اندازه‌های بزرگ پشتیبانی محلی ایجاد می‌کند، درحالی که یک حجم FAT۳۲ تنها برای اندازه‌های بیش از ۳۲ گیگابایت پشتیبانی می‌شود. تحت ویندوز XP، NTFS از فایلی با بیش از حداکثر اندازه دیسک پشتیبانی می‌کند. ویندوز XP ویژگی‌های جدیدی (مثل پشتیبانی برای کسب ویرایش فایل‌های ویدئویی) دارد که به طور متناوب باعث ایجاد فایل‌هایی

می‌گردد که اندازه‌ای بیش از ۴ گیگابایت دارند. NTFS یک سیستم فایل با روال‌های رویداد نگاری است. NTFS آماری از تغییرات ایجاد شده را می‌نویسد که مزایای مهمی دارد مخصوصا زمانی که یک سیستم دچار افت برق می‌شود، با ریست ناخواسته مواجه می‌شود یا دچار وقفه می‌گردد. NTFS می‌تواند به سرعت دیسک را بدون اجرای CHKDSK به وضعیت ثابت بازگرداند. این باعث می‌شود که کاربر تجربه خوبی پیدا کند و میزان فراخوانی برای پشتیبانی را نیز کاهش می‌دهد.

مدیریت حافظه

ویندوز XP، مثل بیشتر سیستم‌عامل‌های مدرن، از حافظه مجازی استفاده می‌کند. ویندوز XP معمولاً همیشه بررسی می‌کند که آیا حافظه تخصیص یافته به یک برنامه خاص واقعا مورداستفاده قرار می‌گیرد یا خیر و همیشه برآوردی از این میزان مصرف حافظه را که می‌تواند بدون تأثیرگذاری روی حافظه دور شود، حفظ می‌نماید. ذخیره حافظه در دسترس قرار دارد تا در صورت نیاز مورداستفاده قرار گیرد. هنگامی که این ذخیره رو به کاهش رود، توسط مجموعه‌های در حال کار پیرایش می‌شود. از این برآوردها به‌عنوان یک دستورالعمل برای تعیین جایی که حافظه باید گرفته شود، استفاده می‌شود.

حافظه مجازی در میان فضاهای گرفته شده توسط برنامه‌های کاربردی، کد درایور، داده‌های تخصیص یافته و ترسیم شده توسط سیستم و فضای مورداستفاده توسط سیستم تقسیم می‌شود. در ویندوز، حافظه فیزیکی دارای تخصیص صفحه ذخیره شده و ذخیره نشده است. حافظه صفحه ذخیره نشده[۷۹] برای کدی است که زمان برای آن بحرانی است مثل مدیر حافظه مجازی (VMM). حافظه صفحه ذخیره شده

[۷۹] - Non-page-pooled memory

برای فایل‌های دیسک طرح ریزی شده است و به سیستم‌عامل اجازه می‌دهد تا در صورت نیاز به حافظه فیزیکی در جایی دیگر، صفحات حافظه از دیسک خارج شوند. مخزن حافظه توسط سیستم توصیف گر مدیریت می‌شود که به آن PTE یا ورودی جدول صفحه می‌گویند و شامل تعداد قاب صفحات حافظه است که به صفحات حافظه فیزیکی اشاره می‌کند. علاوه بر تعداد قاب صفحه حافظه، PTE شامل بیت‌های مربوط به وضعیت صفحات در حال استفاده، آلوده، تمیز و مصرف نشده است. مدیر حافظه وضعیت صفحه را برای واکشی و استفاده مجدد با فهرست جدول صفحه بررسی می‌کند.

در مبارزه میان درایورها و پردازنده‌ها بر سر حافظه (تحت شرایطی که حافظه کمی در اختیار است)، این کاربر است که بازنده می‌شود. معمولاً، این شرایط موقتی اند و هنگامی که درایور یا فرآیند بلوک‌های حافظه را آزاد می‌کند، حافظه در اختیار قرار می‌گیرد. هنگامی که یک درایور یا برنامه کاربردی به حافظه نیاز داشته باشد، سیستم را برای تخصیص حافظه در اختیار می‌گیرد. عمل تخصیص یا انجام می‌شود یا نادیده گرفته می‌شود. در نسخه‌های قبلی ویندوز، روال تخصیص که باید انجام شود، مجاز می‌گردد تا سیستم مجبور گردد به درایور کمی حافظه بدهد. متاسفانه، در زمان فقدان حافظه، این می‌تواند باعث خاموشی سیستم شود. برای اینکه این زمآن‌ها رد شوند، ویندوز XP دیگر به درایورها اجازه نمی‌دهد تا درخواست‌های بیشتری را تخصیص دهند. اگر یک برنامه یا درایور از یک درخواست موفق استفاده کند، رد می‌شود. تمام درایورهای ویندوز XP دوباره بازنویسی شده اند تا امکان استفاده از این درخواست‌ها فراهم نشود. درایورهای شخص ثالث نیز باید برای کسب وضعیت درایور تخصیصی پیروی نمایند. مرحله بعدی که باید توسط ویندوز XP گرفته شود تا امکان کنترل بیشتری از حافظه

انجام گردد، نادیده گرفتن I/O است. به دلایل کارایی، ویندوز سعی می‌کند تا جایی که ممکن است بیشتر پردازش‌ها را به‌صورت موازی انجام دهد. بااین‌حال، اگر مصرف حافظه برای جایی باشد که در آن چیزی باقی نمانده است، ویندوز پردازش حافظه خود را به یک صفحه در هر زمان می‌رساند و حتی‌المقدور از منابع استفاده می‌کند. با اینکه این کار سیستم را کند می‌کند اما باعث خاموشی آن نمی‌شود.

۱۳-۲-۳ معماری ویندوز NT

عملیات اجرایی

سیستم‌های جانبی اجرایی NT از مهم‌ترین لایه در وضعیت کرنل تشکیل می‌شوند و بیشتر عملیاتی که به‌صورت سنتی به سیستم‌عامل‌ها تخصیص یافته اند را اجرا می‌کنند. این سیستم‌های جانبی مسئولیت و نام جداگانه دارند. NT سیستم‌های جانبی اجرایی را به فرآیندهای مختلف تخصیص نمی‌دهد، NT این سیستم‌های فرعی اجرایی را در فایل‌های مختلفی قرار نمی‌دهد.

مدیر شئ

مدیر شئ یکی از سیستم‌های جانبی اجرایی NT است. سیستم‌های اجرایی دیگر از مدیر شی برای تعریف و مدیریت اشیایی استفاده می‌کنند که منابع را نشان می‌دهند. مدیر شئ وظایف مربوط به مدیریت شی را که شامل شناسایی و محاسبه مرجع است، انجام می‌دهد.

مدیر حافظه مجازی

مدیر حافظه مجازی دو وظیفه مهم دارد: ایجاد و مدیریت نقشه آدرس برای پردازش‌ها و برای کنترل تخصیص حافظه فیزیکی. NT۴,۰ یک فضای آدرس ۳۲ بیتی (۴ گیگابایتی) را پیاده سازی می‌کند، بااین‌حال، برنامه‌های کاربردی می‌توانند تنها به اولین ۲ گیگابایت دسترسی پیدا کنند. بخش ۲ گیگابایت تا ۴ گیگابایتی فضای آدرس برای بخش‌های وضعیت کرنل NT است و تغییر نمی‌کند. مدیر حافظه مجازی، حافظه مجازی صفحه مورد تقاضا را پیاده سازی می‌کند، یعنی حافظه را در بخش‌ها یا صفحات مجزا مدیریت می‌نماید. در سیستم‌های x۸۶، یک صفحه ۴۰۹۶ بایت است. مدیر حافظه مجازی قابلیت‌های پیشرفته ای دارد و می‌تواند طرح ریزی حافظه فایل، اشتراک گذاری حافظه و حفاظت کپی در نوشتن صفحه را انجام دهد. NT از طرح ریزی حافظه فایل برای بارگذاری تصاویر قابل اجرا و DLL ها استفاده می‌کند. کپی در نوشتن یک روش بهینه مربوط به اشتراک گذاری حافظه ای است که در آن برنامه‌های مختلفی از یک داده استفاده می‌کنند و هر برنامه می‌تواند به طور جداگانه اصلاح شود. هنگامی که یک برنامه روی یک صفحه کپی در نوشتن، نوشته می‌شود که با برنامه دیگری به اشتراک گذاشته می‌گردد، برنامه ای که تغییرات را انجام می‌دهد، نسخه مخصوص به خود از صفحه کپی در نوشتن را به دست می‌آورد. سپس برنامه دیگر مالک صفحه اصلی می‌گردد. NT از بهینه سازی کپی در نوشتن زمانی استفاده می‌کند که چند برنامه بخش‌های قابل نوشتن سیستم DLLs را به اشتراک بگذارند.

مدیر I/O

مدیر I/O مسئول ادغام درایورهای دستگاه اضافه با NT است. درایورهای دستگاه که به‌صورت پویا مؤلفه‌های حالت کرنل را بارگذاری می‌کنند، از سخت‌افزار پشتیبانی می‌کنند. یک درایور دستگاه، نوع خاصی از دستگاه سخت‌افزاری را کنترل می‌کند این کنترل به‌صورت ترجمه فرامینی که NT به دستگاه هدایت می‌کند به فرامین مخصوص دستگاه است و این ترجمه باعث کنترل سخت‌افزار برای انجام فرامین می‌گردد. مدیر I/O ورودی خروجی غیرهمزمان مبتنی بر بسته را پشتیبانی می‌کند. مدیر I/O از آفست‌های فایل ۶۴ بیتی و درایورهای لایه ای دستگاه پشتیبانی می‌کند. استفاده از آفست‌های ۶۴ بیتی به سیستم‌های فایل NT اجازه می‌دهد که به فایل‌های بزرگ بپردازند و به درایورهای دستگاه دیسک نیز اجازه می‌دهد که به دیسک‌های خیلی بزرگ بپردازند. طرح بندی به درایورهای دستگاه اجازه می‌دهد تا کار خود را تقسیم بندی کنند.

مدیر حافظه کش

مدیر حافظه کش نزدیک به مدیر حافظه مجازی و درایورهای سیستم فایل کار می‌کند. مدیر حافظه کش حافظه کش سیستم فایل جهانی NT (که بین تمام سیستم‌های فایل مشترک است) را نگهداری می‌نماید. تقویت کننده مجموعه کاری حافظه فیزیکی را به کش سیستم فایل تخصیص می‌دهد. کش NT بیشتر به فایل گرایش دارد تا بلوک دیسک.

مدیر پردازش

مدیر پردازش در NT موضوع فرآیند کرنل را تحت پوشش قرار می‌دهد و به آن شناسه فرآیند (PID)، نشانگر دسترسی و نشانی نقشه و یک جدول کنترل اضافه می‌کند. مدیر پردازش یک عملیات

مشابه روی موضوع کرنل اجرا می‌کند و به آن یک شناسه موضوع (TID) و آمار اضافه می‌نماید. این آمار شامل زمان آغاز و خروج فرآیند و موضوع و شمارنده‌های حافظه مجازی مختلفی است.

هسته

هسته NT نسبت به بخش اجرایی بیشتر ه سخت‌افزار نزدیک است و شامل کد مخصوص CPU می‌باشد. برنامه موضوع NT، که توسعه دهندگان NT به آن توزیع کننده می‌گویند در هسته باقی می‌ماند. توزیع کننده ۳۲ سطح اولویت را پیاده سازی می‌کند: ۰،۳۱. توزیع کننده سطح اولویت ۰ را برای یک موضوع سیستمی در نظر می‌گیرد که صفحات حافظه را به‌صورت یک کار پس‌زمینه‌ای به صفر می‌رساند. اولویت سطح ۱ تا ۱۵ در دسترس هستند (با برخی سطوح اولویت ثابت) و جایی قرار دارند که برنامه‌ها اجرا می‌شوند، اولویت سطح ۱۶ تا ۳۱ سطوحی با اولویت ثابت هستند که تنها توسط مدیران قابل دسترسی می‌باشند. توزیع کننده NT یک برنامه پیشگیرانه است. زمان CPU به برش‌هایی به نام کوانتوم تقسیم می‌شود. نگامی که یک موضوع در انتهای کوانتوم خود اجرا شود و از CPU استفاده نکند، توزیع کننده پیش‌دستی می‌کند و یا موضوع دیگری با اولویت یکسان که منتظر اجرا بود را در برنامه قرار می‌دهد. NT بیشتر طرح‌های اولیه همگام را در هسته اجرا می‌کند. هسته انواع موضوعات مخصوص به خود را پیاده سازی و مدیریت می‌کند و موضوعات هسته طرح‌های اولیه هماهنگ با NT را ارائه می‌دهد.

۱۳-۲-٤ معماری ویندوز ME

گروه طراح ME معماری آن را بر اساس ویندوز ۹۸ ایجاد کردند، برای اینکه ویندوز میلنیوم بر اساس ویندوز ۹۸ طراحی شده بود و روی ویندوز میلنیوم اطلاعات کافی وجود نداشت.

مدیریت حافظه

در ویندوز ۹۸، دسترسی به حافظه به کمک طرح آدرس دهی خطی ۳۲ بیتی انجام می‌شود. یک سیستم آدرس دهی ۳۲ بیتی می‌تواند تا بیش از ۴ گیگابایت حافظه در اختیار داشته باشد. بنابراین، در ویندوز ۹۸، هنگامی که یک برنامه بخواهد حافظه بگیرد، به‌سادگی می‌تواند یک آدرس حافظه ۳۲ بیتی مشخص کند (حداقل تخصیص حافظه مجازی یک صفحه ۴ کیلوبایتی است).

مدیر حافظه مجازی ویندوز ۹۸ (VMM) تخصیص حافظه فیزیکی و منطقی را کنترل می‌کند. هنگامی که برنامه جدیدی را راه اندازی می‌کنید، مدیر حافظه مجازی فضای آدرس مجازی را مقدار دهی می‌کند. VMM ویندوز ۹۸ می‌تواند تا بیش از ۴ گیگابایت را آدرس دهی کند که شامل فضای موجود در درایوهای هارد سیستم شما نیز می‌شود، بنابراین برنامه نویسان می‌توانند برنامه‌هایی بنویسند که از مقدار زیادی حافظه بدون نگرانی درباره نوع حافظه یا مقدار حافظه موجود، بهره برداری کنند.

مدیر حافظه مجازی ویندوز ۹۸ می‌تواند این فضای حافظه مجازی بزرگ را از طریق دو فرآیند مدیریت حافظه (صفحه بندی و I/O فایل طرح ریزی شده) در اختیار برنامه‌های کاربردی قرار دهد.

صفحه بندی

هر صفحه از حافظه به یکی از سه دسته زیر تقسیم می‌شود: دایرکتوری صفحه، جدول‌های صفحه یا برافروختگی صفحه. برای برنامه‌های کاربردی حساس به زمان و برنامه‌هایی که نیاز به عملکرد خاص حافظه دارند، VMM به کاربر یک سیستم جانبی یا پردازشی با اولویت‌های خاص می‌دهد تا بتواند صفحات مجازی منتخب را در مجموعه کاری خود قفل کند تا این اطمینان حاصل شود که یک صفحه مهم در حین کار از دست نمی‌رود.

در پیاده سازی پردازش حافظه مجازی، ویندوز ۹۸ یک فایل برای تبادل دیسک سخت ایجاد کرده که توسط آن اطلاعاتی که در حافظه فیزیکی (RAM) گنجانده نمی‌شوند، نوشته می‌شوند. فایل تبادل ویندوز ۹۸ پویا است و بر اساس عملیات اجرا شده در سیستم، دارای افت و خیز می‌باشد (اگر فایل تبادل ویندوز ۹۸ از قبل وجود نداشته باشد، باید در حین راه اندازی سیستم ایجاد شود و همین باعث کندی زمان راه اندازی می‌شود).

ورودی/خروجی فایل طرح ریزی شده

اگر یک برنامه کاربردی سعی کند که فایلی بزرگ‌تر از RAM سیستم و فایل صفحه بندی (فاصله مبادله) بارگذاری کند، از سرویس‌های ورودی/خروجی فایل طرح ریزی شده مدیر حافظه مجازی استفاده می‌شود. I/O فایل طرح ریزی شده به مدیر حافظه مجازی این امکان را می‌دهد تا آدرس‌های حافظه مجازی را در یک فایل بزرگ ترسیم نماید و به برنامه کاربردی اطلاع می‌دهد که فایل در دسترس قرار دارد و سپس تنها قطعاتی از فایل که برنامه واقعا نیاز دارد را بارگذاری می‌کند. از آنجایی که تنها بخش‌هایی از فایل بزرگ در حافظه بارگذاری می‌شوند (RAM یا فایل صفحه)، این کار تا حد زیادی

باعث کاهش زمان بارگذاری فایل و تخلیه منبع سیستم می‌شود. این سرویس برای برنامه‌های کاربردی که اغلب به فایل‌های بزرگی نیاز دارند، بسیار مفید است.

حفاظت

در ویندوز ۹۸، هر برنامه کاربردی (۱۶ بیتی، ۳۲ بیتی یا MS-DOS) از دیگری حفاظت می‌شود. سیستم حافظه ویندوز ۹۸ نیز این امکان را می‌دهد که برنامه‌های کاربردی از برنامه‌های دیگر و از بخش‌های حافظه خود نیز جدا شوند. به دلیل پیشرفت حفاظت در ویندوز ۹۸، یک برنامه کاربردی ۱۶ بیتی نمی‌تواند کل یک سیستم یا برنامه‌های کاربردی MS-DOS یا ۳۲ بیتی دیگر را متوقف کند. بااین‌حال، توقف برنامه‌های کاربردی ۱۶ بیتی هنوز می‌تواند روی اجرای برنامه‌های ۱۶ بیتی اثر بگذارد. هر نوع برنامه کاربردی ۱۶، ۳۲ یا MS-DOS یک مدیر ماشین مجازی مرتبط دارد. توسعه حفاظت نیز شامل استفاده از صف‌های پیام جداگانه برای اجرا همه برنامه‌های کاربردی ۳۲ بیتی است.

مروری بر سیستم فایل و دیسک

خوشه‌ها

ویندوز ۹۸ که بعد از DOS آمد، فضای دیسک به در خوشه تخصیص می‌داد. یک خوشه به گروهی از سکتورهای موجود در یک دیسک می‌گویند. تعداد سکتورهای موجود در یک خوشه با توجه به نوع درایو اندازه پارتیشن متفاوت است. هنگامی که ویندوز ۹۸ فایلی را در یک دیسک ذخیره می‌کند، آن

فایل به‌صورت سکتور به سکتور در آن ذخیره نمی‌شود. در عوض، ویندوز ۹۸ خوشه‌های کافی را برای آن تخصیص می‌دهد.

FAT

به دلیل وجود سکتورهای فراوان در یک دیسک، ویندوز ۹۸ به روشی برای ردیابی جایی که هر فایل و دایرکتوری قرار می‌گرفتند، نیاز داشت. در اصل، ویندوز ۹۸ باید خوشه آغازی و پایانی هر فایل را بلد باشد. جدول تخصیص فایل یا FAT این اطلاعات را فراهم می‌کند. FAT برای هر خوشه موجود در هر دیسک یک ورودی دارد ویندوز ۹۸ از FAT برای ردیابی خوشه‌هایی که برای فایل‌ها و دایرکتوری‌ها تخصیص یافته اند، استفاده می‌نماید. FAT کلیدی است که به ویندوز ۹۸ امکان می‌دهد فایل‌ها را روی دیسک تخصیص دهد، از آن بخواند و روی آن بنویسد.

VFAT، CDFS و VCACHE

ویندوز برای کارهای گروهی VFAT را معرفی کرده که یک درایور سیستم فایل قابل نصب است که بین برنامه‌های کاربردی و سیستم فایل یک رابط ۳۲ بیتی ایجاد کرده است. VFAT در حالت محافظت شده کار می‌کند و به ویندوز ۹۸ و برنامه‌های کاربردی ۱۶ یا ۳۲ بیتی امکان دسترسی به سیستم فایل را بدون سوییچینگ پردازنده از حالت محافظت شده به حالت واقعی می‌دهد، که همین امر به طور قابل توجهی باعث بهبود عملکرد شده است. با کار در کنار VFAT به یک حافظه کش مجازی به نام VCACHE رسیدیم که یک کش دیسک ۳۲ بیتی در حالت محافظت شده می‌باشد. یک کش دیسک با ذخیره داده‌هایی که به تازگی مورداستفاده قرار گرفتند و خواندن این داده‌ها از حافظه به جای

دیسک در حالتی که درخواست زیادی برای داده وجود دارد، باعث بهبود عملکرد I/O شده است. ویندوز ۹۸ شامل یک درایور سیستم فایل CD-ROM محافظت شده ۳۲ بیتی به نام CDFS است. CDFS به برنامه‌های کاربردی امکان خواندن از CD-ROM را در وضعیت محافظت شده می‌دهد و نیازی نیست که سیستم برای خواندن CD به حالت واقعی برود که همین مسئله باعث بهبود I/O فایل می‌شود.

FAT۳۲

FAT۳۲ به جای طرح آدرس دهی ۱۶ بیتی، از طرح ۳۲ بیتی آن استفاده می‌کند. FAT۳۲ به دایرکتوری ریشه این امکان را می‌دهد که در هر جایی روی دیسک زندگی کند و تا زمانی که به آن نیاز است، در همان‌جا باشد. FAT۳۲ از اطلاعات ضروری دیسک نیز پشتیبانی می‌کند و باعث می‌شود که پارتیشن‌های FAT۳۲ کمتر در معرض خطا یا تخریب داده قرار گیرند. FAT۳۲ بهره وری بیشتری نسبت به FAT۱۶ دارد. اندازه خوشه‌های آن کوچک‌تر است، برای اینکه طرح آدرس دهی ۳۲ بیتی آن به طور مستقیم می‌تواند بیش از آن‌ها را آدرس دهی نماید. FAT۳۲ در کنار مزایای خود، معایبی نیز دارد. اولین عیب FAT۳۲ نسبت به FAT۱۶ سرعت آن است. FAT۳۲ در انجام عملیات فایل مشترک اندکی کندتر است. دومین عیب قابلیت عقب‌گرد آن می‌باشد. میزبان برنامه‌های کاربردی و روال‌ها با پارتیشن‌های FAT۳۲ نمی‌توانند کار کنند. علاوه بر این، درایوهای فشرده به‌صورت FAT۳۲ قابل فرمت نیستند و درایوهای موقتی را نیز نمی‌توان در FAT۳۲ فرمت کرد. لپ تاپ‌ها نمی‌توانند عملیات مشکوک دیسک را در درایوهای FAT۳۲ اجرا کنند و اگر خواننده PC از حالت

خواب مدیریت توان پشتیبانی کند، اگر درایو در FAT۳۲ فرمت شود، لپ تاپ خاموش می‌شود. درنهایت، درایوی که از FAT۳۲ استفاده می‌کند، نمی‌تواند ویندوز ۹۸ را حذف نصب نماید.

۱۳-۳ تراکم بار

تراکم بار حیاتی ترین بخش مربوط به پروژه ارزیابی عملکرد است. هنگامی که ما تراکم بار را برای ارزیابی انتخاب می‌کنیم، ملاحظات قابل توجهی به آن داریم، مثل سرویس‌هایی که توسط تراکم بار ارائه می‌شوند، سطح جزئیات، نمایش مؤثر و بهنگام بودن.

۱۳-۳-۱ توصیف تراکم بار

تراکم بار را می‌توان به‌صورت زیر توصیف نمود:

۱. **فرآیندها**: برای بررسی توانایی OS در اداره چند فرآیند. این شامل ایجاد، زمان‌بندی، تخصیص منابع و کشتن فرآیند است. این را می‌توان با اجرای یک برنامه C یا جاوا پیاده سازی نمود که تعداد زیادی فرآیند و موضوع را ایجاد خواهد کرد. این برنامه بدون تخریب شدن اجرا خواهد شد – یعنی به‌صورت نامحدود. با استفاده از این روش، سیستم موظف است که بعد از چند فرآیند خاص، خاموش شود. این برنامه در یک فایل، فرآیندها و زمان ایجاد آن‌ها را ثبت خواهد کرد. به این صورت، ما احساس خوبی برای چیزی که OS کنترل می‌کند، خواهیم داشت.

۲. **محاسبات**: برای آزمایش توانایی محاسبات هر OS، با این فرض که سخت‌افزار برای تمام پیکربندی‌ها یکسان باشد. کنترل ALU سیستم‌عامل را می‌توان با اجبار آن برای اجرای تعداد زیادی

عملیات ریاضی آزمایش کرد. بخش‌های زیر شامل توصیف کلی سه تراکم بار است که برای ارزیابی عملکرد سیستم‌عامل‌ها در نظر گرفته شده اند:

۱. برنامه MATLAB، که عملیات روی ماتریس‌ها را انجام می‌دهد (مثل، جمع، ضرب، محاسبه دترمینان و معکوس ماتریس ها).

۲. یک برنامه C برای I/O.

۳. یک برنامه C که فرآیندهای متعددی را ایجاد می‌کند.

برنامه‌های مربوط به تراکم بار که توسط سیستم‌عامل‌های ویندوز ME، ویندوز XP، ویندوز NT و لینوکس مورداستفاده قرار می‌گیرند.

برنامه‌هایی که توسط سه گروه مایکروسافت و یک گروه لینوکس مورداستفاده قرار گرفته اند، در ادامه معرفی می‌شوند. اول، workload۱ یک برنامه C است که تعدادی از فایل‌ها را باز می‌کند و چند بایت در آن‌ها می‌نویسد. این برنامه ۹ آزمایش مختلف را سه بار تکرار می‌کند. دوم، مشابه اولی است بجز اینکه چند فرآیند را با دفعات خواب مختلف بین فرآیندها می‌گشاید تا ببیند که چند سیستم‌عامل قابل کنترل هستند. مورد دوم نیز ۹ آزمایش را سه بار اجرا می‌کند. workload سوم یک برنامه MATLAB است که چند عمل ماتریس را با ماتریس‌هایی با اندازه مختلف اجرا می‌کند. مورد سوم نیز ۹ آزمایش را ۳ بار اجرا می‌کند. این سه برنامه به این منظور نوشته شده اند تا به‌صورت خودکار آزمایشات را اجرا کنند و داده‌ها را در فایل‌های مناسب ثبت کنند. در کنار این سه برنامه، یک برنامه چهارم نیز هست که به آن مانیتور عملکرد می‌گویند و در هر ۵۰۰ میلی ثانیه اطلاعات مربوط به استفاده

از CPU و حافظه را ثبت می‌کند. سپس، استفاده از CPU و حافظه در ابتدا و انتهای آزمایشات با یکدیگر مقایسه می‌شوند و زمان پاسخ این آزمایشات در یک فایل اکسل ذخیره می‌گردد.

اولین برنامه با انتخاب میان ۵۰۰، ۷۵۰ و ۱۰۰۰ بایت برای نوشتن یک فایل و برای نوشتن میان ۱۰۰، ۵۰۰ و ۱۰۰۰ فایل اجرا می‌شود. این در کل ۹ آزمایش نیاز دارد. هر آزمایش سه مرتبه تکرار می‌شود. زمان پاسخ هر یک از ۲۷ آزمایش در یک فایل ثبت می‌گردد. این کار با باز کردن فایل، الحاق کردن به آن و بسته آن بعد از اجرای یک آزمایش جدید، انجام می‌شود. این برنامه تعداد بایت ها، تعداد فایل ها، زمان آغاز آزمایش، زمان پایان آزمایش و زمان پاسخ آزمایش را در یک خط فایل (به نام work۱.txt) می‌نویسد. این فایل نیازی ندارد که در زمان اجرای برنامه، ایجاد شود. دومین برنامه مشابه اولی است. این برنامه میان ۱۰، ۱۰۰ و ۱۰۰۰ پردازش انتخاب می‌کند و زمان خواب آن نیز ۰، ۱۰۰و ۱۰۰ میلی ثانیه در بین هر فرآیند است. مثل workload۱، روی فایل باهمان فرمت work۱.txt نوشته شده است. این برنامه تعداد فرآیندها را به جای تعداد بایت‌ها و سرعت پردازش (زمان خواب) را به جای تعداد فایل‌ها ثبت می‌کند. این در work۲.txt نوشته می‌شود.

برنامه سوم یک برنامه MATLAB است که از دو فایل تشکیل می‌شود. فایل اول matrix.m است که یک برنامه برای گرفتن تعداد ماتریس‌ها و اندازه ماتریس است و چند عملیات را روی این ماتریس‌ها اجرا می‌نماید. فایل matrix.m توسط فایل workload.m اجرا می‌شود. این فایل میان ماتریس‌ها یا اندازه ۱۰×۱۰، ۵۰×۵۰ یا ۱۰۰×۱۰۰ عنصر انتخاب می‌کند و ۱۰، ۱۰۰ یا ۱۰۰۰ ماتریس را مورداستفاده قرار می‌دهد. فایل matrix.m روی یک فایل به نام work.txt در همان فرمت workload۲

نوشته شده است اما اندازه ماتریس به جای تعداد فرآیندها و تعداد ماتریس به جای سرعت پردازش است. فایل matrix.m در یک فایل به نام work۳.txt در همان فرمت workload۲ نوشته می‌شود اما اندازه ماتریس در عوض تعداد پردازش‌ها و تعداد ماتریس به جای سرعت پردازش است. برنامه چهارم مربوط به برنامه ناظر عملکرد است. این برنامه بهره برداری از CPU و حافظه سیستم را می‌خواند و آن‌ها را در یک فایل که توسط آزمایش کننده مشخص می‌شود، می‌نویسد. زمان و بهره برداری از CPU و حافظه در هر ۵۰۰ میلی ثانیه ثبت شده اند.

برای کسب اطلاعات، باید از زمان‌های آغاز و انتهای یک آزمایش در هر یک از تراکم‌های بار استفاده کنیم و سپس بلافاصله یک برچسب زمانی پس از زمان آغاز و بعد از زمان پایان پیدا کنیم. این به ما تعداد قرائت‌های بین زمان آغاز و پایان را می‌دهد. ما سپس میانگین قرائت‌هایی که مورد نظر ماست را با میانگین قرائت‌هایی که اطلاعات موردنیاز برای آن آزمایش لازم است، به دست می‌آوریم.

اجرای این برنامه‌ها روی ویندوز ME

اجرای این برنامه‌ها در ME اثبات شده که کمی سخت است. در ابتدا، این سه workload برای سه سیستم‌عامل یکسان بودند بجز ME که فرآیند cmd.exe را ندارد، این فرآیند در workload۲ برای آزمایش تعداد پردازش‌هایی که سیستم کنترل می‌کند، استفاده می‌شود. در عوض، از گزینه mem.exe استفاده کردیم که یک کنسول کاربردی مشابه است. علاوه بر این، برای استفاده از برنامه ناظر عملکرد، مجبور بودیم که از فراخوان‌های مخصوص سیستم خود استفاده نماییم. در حین اجرای برنامه workload۲، ME نمی‌تواند از برنامه mem.exe بعد از اجرای چند فرآیند استفاده کند.

برخی از آن فرآیندها را اجرا و سپس حافظه را تمام می‌کند. یک پیام خطا ظاهر می‌شود که می‌گوید mem.exe را حتی درست پس اجرا با چند فرآیند بیشتر و کمتر نمی‌تواند پیدا کند. این حدود ۱۰۰ پردازش با ۱۰۰ میلی ثانیه زمان خواب بین فرآیندها رخ می‌دهد.

تراکم بار برنامه MATLAB برای لینوکس ۷,۲

هدف این برنامه MATLAB تعیین میزان استفاده از حافظه و CPU است که در حین عملیات ماتریسی رخ می‌دهد. این عملیات عبارتند از ضرب ماتریس، جمع ماتریس‌ها و عملیات معکوس و دترمینان. برای انجام این آزمایشات، اندازه ماتریس از ۱۰×۱۰ تا ۵۰×۵۰ تا ۱۰۰×۱۰۰ است و تعداد ماتریس‌ها از ۱۰ تا ۱۰۰ تا ۱۰۰۰ می‌باشد. با توجه به مشخصات پروژه، این پارامترها را به‌صورت کوچک، متوسط و بزرگ طبقه بندی می‌کنیم. این برنامه با دادن یک دستور UNIX در فرمان prompt اجرا شد. این دستور به‌صورت زیر نوشته می‌شود:

(experiment number, matrix size, number of matrices) matfinal

برای محاسبه درصد بهره برداری از CPU، از فرمان‌های tic، toc و زمان CPU استفاده کردیم. این دستورات قبل و بعدازاینکه عملیات ماتریس ذکر شوند، فراخوان می‌گردند، نتایج آن‌ها ذخیره می‌شود و میزان درصد استفاده از CPU محاسبه می‌گردد. فرمان tic تایمر stopwatch را آغاز می‌کند و فرمان toc تایمر stopwatch را متوقف می‌نماید. فرمان toc زمان منقضی شده بین فراخوانی آن را باز می‌گرداند و به tic خبر می‌دهد، بنابراین toc زمان منقضی شده برای عملیات ماتریس را اندازه گیری می‌کند. فرمان CPU time زمان مربوط به CPU که از آغاز

MATLAB سپری شده را می‌دهد. مقدار زمان CPU بر مقدار toc تقسیم می‌شود تا میزان استفاده از CPU را به ما بدهد.

برای محاسبه درصد استفاده از حافظه، از یک برنامه C استفاده کردیم (amitfinal.c)که قبل و بعد از عملیات ماتریس از برنامه MATLAB فراخوانی می‌شود. این برنامه C در حین برنامه MATLAB با ایجاد فرمت قابل اجرای MATLAB (mex) فراخوانی می‌گردد – یعنی MATLAB به C توسعه می‌یابد. برنامه C از دستورات سیستم یونیکس استفاده می‌کند تا اطلاعات سیستم درباره حافظه – مقدار RAM، حافظه موجود، حافظه کش و غیره را به دست آورد. برای حفظ این مقادیر حافظه، خروجی برنامه C را در یک فایل متنی به نام mem.txt نوشتیم. محتویات این فایل با استفاده از فرامین ()fid و ()Fopen توسط برنامه MATLAB خوانده می‌شود.

با نگاهی به مقادیر قبل و بعد، با استفاده از فرمول‌های ساده ریاضی میزان حافظه مورداستفاده توسط سیستم برای اجرای عملیات ماتریس را محاسبه می‌کنیم. با دانستن کل میزان حافظه مورداستفاده توسط فرآیندهای دیگر سیستم، متوجه می‌شویم که چند درصد از حافظه برای برنامه MATLAB استفاده شده است. این می‌توانیم با استفاده از فرمان top در دستور prompt این نتایج را بررسی کنیم. فرمان top میزان حافظه مورداستفاده توسط نرم افزار MATLAB را باز می‌گرداند (این نشان دهنده حافظه مورداستفاده برای راه اندازی نرم افزار MATLAB در حین استفاده و حافظه موردنیاز برای عملیات برنامه MATLAB می‌باشد)، در حالی که کد برنامه MATLAB ما تنها میزان حافظه موردنیاز برای عملیات ماتریس را باز می‌گرداند. این بستگی به کاربر دارد تا تصمیم بگیرد از کدام یک برای

رسیدن به یک نتیجه نزدیک‌تر استفاده کند. برای اجرای برنامه C از برنامه MATLAB، باید فرمان قابل اجرای MATLAB را برای کامپایل و اجرای برنامه C بگنجانیم. درنهایت، مقادیر موردنیاز مثل اندازه ماتریس، تعداد ماتریس ها، درصد استفاده از CPU و درصد استفاده از حافظه چاپ شدند. از آنجایی که راهی برای شناسایی درصد حافظه مورداستفاده برای چاپ توسط برنامه MATLAB وجود ندارد، ترجیح می‌دهیم که از مانیتور سیستم استفاده کنیم.

۱۳-۴ طراحی و شبیه سازی آزمایشی

۱۳-٤-۱ مشخصات سخت‌افزار برای سیستم‌های مورداستفاده

برای حصول اطمینان از اینکه سیستم‌عامل‌ها در سطح فیلد تست شده اند، از ۴ کامپیوتر شخصی با سخت‌افزار یکسان استفاده شد. سخت‌افزار هر یک از این کامپیوترها در جدول (۱۳-۱) معرفی شده است.

برای اینکه مطمئن شویم این دستگاه‌ها ازنظر قابلیت‌های کارایی یکسان هستند، یک گروه مستقل تشکیل شد تا معیار عملکرد یکی از کامپیوترها را بررسی نمایند. در بخش بعد به بررسی نتایج می‌پردازیم.

۱۳-٤-۲ معیار PC

Passmark: آزمون عملکرد

این آزمون شامل ۲۲ آزمون معیار جداگانه است که در ۶ بسته آزمونی موجودند. این ۶ بسته به شرح زیرند:

- عملیات ریاضی اعشاری و صحیح
- آزمون توابع گرافیکی دو بعدی استاندارد
- خواندن، نوشتن و کاوش در فایل‌های دیسک
- تخصیص و دسترسی به حافظه
- آزمون MMX (الحاقات چندرسانه ای) در CPU های جدیدتر
- یک آزمون از سیستم گرافیکی DirectX ۳D

گزارش مربوط به نتایج آزمون به‌صورت مقادیر نسبی نشان داده‌شده است. هرچقدر این اعداد و ارقام بزرگ‌تر باشند، یعنی کامپیوتر سریع تر است. برای مثال، کامپیوتری با رقم ۴۰ نسبت به کامپیوتری با رقم ۲۰ می‌تواند دو برابر داده را مورد پردازش قرار دهد. رتبه Passmark یک میانگین از تمام نتایج آزمایشی است و یک نشانه کلی از عملکرد کامپیوتر به ما می‌دهد. نتایج حاصل در جدول (۱۳-۲) دیده می‌شود.

جدول (۱۳-۱): پیکر بندی آزموده شده

CPU	**Pentium III @ ۵۰۰ MHz**
اندازه RAM کل	**۲۵٦ MB**

جدول (۱۳-۲): نتایج مشاهده شده آزمون برای بسته‌های عملکرد passmark

نتایج مشاهده شده آزمون برای بسته‌های عملکرد passmark

S. No.	پارامتر آزموده	Win NT	Win XP	LINUX ۷,۲	Windows ME
۱	ریاضی - جمع	۹۶,۶	۹۶,۲	۹۴,۶	**۹۷,۰**
۲	ریاضی - تفریق	۹۶,۴	۹۷,۱	۹۶,۲	**۹۷,٦**
۳	ریاضی - ضرب	۱۰۱,۱	۱۰۱,۴	۱۰۱,۴	**۱۰۳,۱**
۴	ریاضی - تقسیم	۱۲,۹	۱۲,۸	۱۲,۹	**۱۳,۰**
۵	ریاضی – جمع اعشاری	۸۷,۷	۸۷,۸	۸۷,۶	**۸۸,۷**
۶	ریاضی – تفریق اعشاری	۸۹,۴	۸۹,۵	۸۸,۶	**۹۰,۱**
۷	ریاضی – ضرب اعشاری	۹۱,۷	۹۱,۷	۹۰,۹	**۹۲,۳**
۸	ریاضی – تقسیم اعشاری	۱۴,۸	۱۴,۸	۱۴,۸	**۱٤,۹**
۹	حداکثر مگا فلاپس (عملیات اعداد FlOPS اعشاری در ثانیه)	۱۷۱,۲	۱۷۲,۲	۱۷۰,۷	**۱۷۷,٦**

۱۰	گرافیک دو بعدی –خط	۱۷,۵	۱۷,۶	۱۷,۵	**۱۷,۸**
۱۱	گرافیک دو بعدی – بیت مپ	**۱۲,۹**	**۱۲,۹**	۱۲,۸	**۱۲,۹**
۱۲	گرافیک دو بعدی - اشکال	**٤,۷**	**٤,۷**	**٤,۷**	**٤,۷**
۱۳	گرافیک سه بعدی – دنیاهای چند گانه	**۲۲,۹**	**۲۳,۰**	۲۲,۹	**۲۲,۹**
۱۴	حافظه – بلوک های کوچک تخصیص یافته	۸۶,۶	**۸۷,٦**	۸۷,۰	**۸۷,٦**
۱۵	حافظه – خواندن کش	۶۷,۹	۶۸,۴	۶۸,۰	**٦۸,۵**
۱۶	حافظه – خواندن کش نشده	۴۸,۷	۴۸,۸	**۵۰**	**٤۹,۱**
۱۷	حافظه – نوشتن	۴۰,۸	۴۱,۱	۴۰,۹	**٤۱,٤**
۱۸	دیسک – خواندن ترتیبی	۳,۲	**۳,۸**	۳,۷	**۳,۱**
۱۹	دیسک - نوشتن ترتیبی	۲,۹	**۳,٤**	**۳,٤**	**۲,۹**

٢٠	دیسک - جستجوی تصادفی	١,٢	٢,٣	**٣,٦**	**٢,١**
٢١	MMX-جمع	٩٧,٧	٩٤,٥	٩٧,٨	**٩٩,٤**
٢٢	MMX-تفریق	٩٢,٣	٩٨,٢	٩٣,٣	**٩٦,٠**
٢٣	MMX-ضرب	٩٧,٨	٩٧,٥	٩٦,٩	**٩٩,١**
٢٤	علائم ریاضی	٧٥,٦	٧٥,٨	٧٥,٢	**٧٦,٨**
٢٥	علائم دو بعدی	٤٦,٧	٤٦,٩	٤٦,٧	**٤٧,١**
٢٦	علائم حافظه	٥٨,٧	٥٩,٢	٥٩,٢	**٥٩,٤**
٢٧	علائم دیسک	١٩,٣	٢٥,١	**٢٨,٤**	**٢١,٥**
٢٨	علائم سه بعدی	١٥,٥	**١٥,٧**	١٥,٥	**١٥,٦**
٢٩	علائم MMX	٤٨,٨	٤٩,٢	٤٨,٩	**٥٠**
٣٠	رتبه Passmark	٤٥,٧	٤٧,٢	**٤٧,٨**	**٤٦,٧**

متن پر رنگ نشان دهنده بالاترین مقادیر برای هر گروه است.

۱۳-٤-۳ آزمون Burn-in

کار آزمون Burn-in، یک آزمایش کامل از سخت‌افزار یک PC در دوره کوتاهی است، برای اینکه برنامه‌های معمولی از یک PC برای مدت طولانی استفاده می‌کنند. این آزمون، موضوعات زیر را آزمایش می‌کند:

۱. CPU

۲. درایوهای سخت

۳. CD-ROMs

۴. کارت‌های صدا

۵. گرافیک دوبعدی

۶. گرافیک سه بعدی

۷. RAM

۸. اتصالات و پرینترهای شبکه

نتایج آزمایش در جدول (۱۳-۳) نشان داده‌شده است.

جدول (۱۳-۳): نتایج آزمون برای تست Burn-in

S. No.	اطلاعات سیستم	ویندوز XP	ویندوز NT	ویندوز ME	لینوکس ۷,۲
۲	تعداد CPU	۱	۱	۱	۱

۳	سازنده CPU	Intel	Intel	Intel	**Intel**
۴	نوع CPU	Celeron	Celeron	Celeron	**Celeron**
۵	ویژگی CPU	MMX	MMX	MMX	**MMX**
۶	شماره سریال CPU	N/A or disabled	N/A or disabled	N/A or disabled	**N/A or disabled**
۷	سرعت CPU۱	۵۰۱٫۳ MHz	۵۰۱٫۳ MHz	۵۰۱٫۳ MHz	**MHz ۵۰۱٫۲**
۸	حافظه کش سطح ۲ CPU	۱۲۸ KB	۱۲۸ KB	۱۲۸ KB	**۱۲۸ KB**
۹	RAM	۲۶۷٬۸۲۱٬۰۵۶ Bytes (۲۵۶ MB)	۲۶۷٬۸۲۱٬۰۵۶ Bytes (۲۵۶ (MB	۲۶۷٬۸۲۱٬۰۵۶ Bytes (۲۵۶ (MB	**۲۶۷٬۸۲۱٬۰۵۶ Bytes (۲۵۶ (MB**
۱۰	عمق رنگ	۲۴	۲۴	۲۴	**۲٤**

۱۳-٤-٤ طراحی آزمایشی

عملکرد سیستم‌عامل به عوامل متعددی بستگی دارد. برای انجام یک تحلیل کامل، تأثیر هر عامل باید از عوامل دیگر جدا شود به نحوی که برآورد معناداری از نقش منفی یا مثبت آن‌ها روی

عملکرد ایجاد شود. برای این ارزیابی، ما سه تراکم بار ایجاد کردیم که پیش از این به آن‌ها اشاره نمودیم تا مدیریت پردازش، مدیریت حافظه و سیستم‌های فرعی مدیریت I/O سیستم‌عامل‌های تحت بررسی را آزمایش کنیم. این سه تراکم بار، سیستم‌های فرعی را موردبررسی قرار می‌دهند، اگرچه هر تراکم بار برای فشرده‌سازی سیستم‌های فرعی مخصوصی طراحی شده است. تراکم بار اول که عملیات ماتریس را اجرا می‌کند، ازنظر محاسباتی گسترده است و به حافظه زیادی نیاز دارد. بنابراین، این تراکم بار در درجه اول روی اجرای سیستم‌های جانبی مدیریت حافظه متمرکز می‌شوند. تراکم بار دوم که آرایه ای از فایل‌های مختلف ایجاد می‌کند (یا می‌نویسد و حذف می‌کند) روی مدیریت فایل یا مدیریت I/O سیستم‌های جانبی متمرکز شده است. آخرین تراکم بار، که پردازش پوسته DOS و یونیکس را جدا می‌کند، برای فشرده سازی سیستم فرعی مدیریت پردازش طراحی شده است. طرح آزمایشی این مطالعه به طور خلاصه در جدول (۱۳-۴) آمده است.

ما از یک طرح فاکتوریل کامل برای استفاده از ترکیب احتمالی در تمام سطوح تمام فاکتورها استفاده کردیم. با حذف تعداد فاکتورها و سطح مربوط به آن‌ها، می‌توانیم از مزیت این طرح برای آزمایش ترکیب‌های احتمالی پیکربندی و تراکم بار استفاده کنیم. معیارهایی که برای مقایسه انتخاب شده اند، عبارتند از زمان پاسخگویی، استفاده از CPU، بهره برداری از دیسک، تخصیص حافظه و ثبات. جدول (۱۳-۴) عوامل سطوح آن‌ها را نشان می‌دهد. برای افزایش دقت آماری و حفظ تعداد آزمایشات در

یک سطح معقول، این آزمایشات برای ترکیب عوامل و سطوح سه بار تکرار شدند. تعداد نهایی آزمایشات ۸۱ بود که به‌صورت زیر محاسبه می‌شود:

(سه تراکم بار) × (سه سطح برای فاکتور ۱) × (سه سطح برای فاکتور ۲) × (سه تکرار آزمایش).

جدول (۱۳-۴): طرح آزمایش

تراکم بار	فاکتورها و مقادیر فاکتورها	تعداد تکرار
یک برنامه MATLAB عملیات ماتریس را روی آرایه ای از ماتریس‌هایی با اندازه متفاوت اجرا می‌کند. این برنامه زمان‌های پاسخگویی را نیز ثبت می‌کند.	اندازه ماتریس – کوچک، متوسط و بزرگ (در برنامه تعریف‌شده است) تعداد ماتریس‌ها – ۱۰، ۱۰۰ و ۱۰۰۰	۳
یک برنامه C که تعدادی فایل با اندازه مختلف ایجاد می‌کند و زمان‌های پاسخ را ثبت می‌نماید.	اندازه فایل – کوچک (۱KB)، متوسط (۱۰KB)، بزرگ (۱۰۰KB) تعداد فایل‌ها – ۱۰، ۱۰۰، ۱۰۰۰	۳
یک برنامه C که چند پردازش پوسته UNIX/DOS متعدد را منشعب می‌کند. این برنامه نیز زمان پاسخ را ثبت می‌کند.	تعداد پردازش‌ها – ۱۰، ۱۰۰، ۱۰۰۰ سرعت ورودی پردازش – کند (۱۰۰ p/s)، متوسط (۱۰۰۰ p/s) و سریع (همه باهم)	۳

۱۳-٤-۵ شبیه سازی

یک شبیه سازی شامل فعالیت‌های زیادی است از جمله جمع آوری داده، ایجاد مدل ها، اجرای شبیه سازی، تولید روش جایگزین، تجزیه و تحلیل خروجی‌ها و ارائه نتایج. سیستم‌های شبیه سازی

برای تمام فعالیت‌ها پشتیبانی نرم افزاری دارند. AWESIM یک سیستم شبیه سازی است که از ایجاد مدل، تحلیل مدل‌ها با کمک شبیه سازی و ارائه نتایج شبیه سازی پشتیبانی می‌کند. رابط کاربری AWESIM بر اساس دیدگاه بازدید – هدف – انتخاب ایجادشده است که پنجره‌های متعددی ایجاد می‌کند، مجموعه ای منو نیز در هر پنجره و یک ماوس برای انتخاب گزینه‌های منو دارد. هر گروه برای آماده سازی مدل سطح بالا پیش از استفاده از ابزار شبیه سازی AWESIM ایجاد می‌شود. بخش زیر شامل مدل سطح بالا برای هرگروه، مدل AWESIM مربوطه و گزارشات می‌باشد.

مدل AWESIM برای LINUX ۷,۲

در زمان طراحی مدل AWESIM، گروه لینوکس همکاری نزدیکی با گروه NT داشتند. گروه NT عمدتاً روی زمان‌بندی پردازش کار می‌کردند در حالی که گروه لینوکس بیشتر روی مدیریت حافظه متمرکز بودند. البته در طراحی هر دو بخش تداخل نیز دیده می‌شد و در مدل نهایی کار هر دو تیم اختلافاتی وجود داشت که به دلیل تفاوت میان سیستم‌عامل‌ها و اختلاف نظر بر سر بهترین نوع طراحی بود.

پیش از اینکه نگاهی به نمودار داشته باشیم، بهتر است ابتدا بدانیم که کدام منابع باید مدل‌سازی شوند و کدام ویژگی‌ها را یک نهاد باید داشته باشد. تنها منابعی که مدل‌سازی شدند، حافظه و CPU بود. CPU یک واحد اختصاص دارد، از آنجایی که برای هر نقطه در هر زمان تنها یک فرآیند CPU را

در اختیار دارد، حافظه به‌اندازه‌ای واحدهای تخصیصی دارد که سیستم مدل‌سازی شده، صفحه در اختیار دارد. ویژگی‌ها به شرح زیر هستند:

ATRIB[۱] = زمان کل موردنیاز CPU، ATRIB[۲] = برچسب زمانی (برای تقریب سازی LRU استفاده می‌شود)، ATRIB[۳] = زمان باقیمانده موردنیاز پیش از اتمام پردازش، LTRIB[۱] = اولویت، LTRIB[۲] = تعداد کل صفحاتی که پردازش هرگز به آن‌ها نیاز نخواهد داشت، LTRIB[۳] = تعداد کل باقیمانده خواندن، LTRIB[۴] = تعداد کل نوشتن باقیمانده، LTRIB[۵] = تعداد کل صفحاتی که در حال حاضر نگهداری می‌شوند و LTRIB[۶] نشان می‌دهد که آیا یک نهاد نشان دهنده برنامه MATLAB است یا برنامه انشعابی C. همانطور که گفتیم، ATRIBها واقعی هستند، در حالی که LTRIBها اعداد صحیح هستند. ATRIB[۲] به توضیح بیشتری نیازی دارد. با استفاده از برچسب‌های زمانی که به ازای هر بار مراجعه حافظه ایجاد می‌شود، LRU تقریب زده می‌شود. بااین‌حال، به جای استفاده از یک برچسب زمانی برای هر صفحه (که خیلی بهتر است)، از یک برچسب زمانی برای هر پردازش استفاده می‌شود. منطق می‌گوید که آخرین چیزی که حافظه به آن اشاره می‌کند، آخر از همه مورد استفاده قرار می‌گیرد. هنگامی که این الگوریتم به‌صورت قابل قبول تعیین شود، مسئله دیگری بروز می‌کند: مبادله صفحات. اگر در فرآیند A صفحه ای از دست برود و فرآیند B نیز LRU (آخرین صفحه مورداستفاده) داشته باشد، فرآیند A می‌تواند یک صفحه از فرآیند B با یک گره خالی، بگیرد و از ATRIB[۲] برای نشان دادن اولویت استفاده می‌کند. بااین‌حال مشکل زمانی به وجود می‌آید که پردازش خودش LRU را داشته باشد،

برای اینکه یک پردازش به وضوح نمی‌تواند خودش را مقدم قرار دهد. این مشکل با ترکیب گره HaveLRU و تابع تعریف‌شده توسط کاربر رفع می‌شود. در گره HaveLRU، اگر فرآیند A، LRU را در اختیار داشته باشد، وارد گره swapwself خواهد شد، در غیر این صورت، نهاد فوق به گره مقدم مسیریابی خواهد شد. تابع تعریفی کاربر صرفاً برچسب زمانی تمام نهادهایی را آزمایش می‌کند که حداقل یک صفحه از حافظه را در اختیار داشته باشند. سپس مقدار برچسب زمانی LRU را باز می‌گرداند. اگر برچسب زمانی نهاد برابر برچسب زمانی LRU باشد، می‌توان از یک شاخه نیز استفاده کرد. هر زمانی که یک نهاد به همین شیوه از نهاد دیگر پیشی گرفت، لازم است که LTRIB[۵] یک واحد افزایش یابد (برای اینکه آن می‌تواند بیش از یک صفحه داشته باشد). هنگامی که مفاهیم قبلی را درک کردیم، بقیه مدل را به‌سادگی می‌توانیم درک کنیم.

مدل

فرآیندها در گره create ایجاد می‌شوند، create_proc، اما مشخص نمی‌کنیم که چه زمانی برای شبیه سازی برنامه MATLAB یا برنامه C به آن‌ها نیاز است تا گره goon، Initialize، که به‌صورت تصادفی برای ارسال به گره تخصیص (forking) یا گره تخصیص MATLAB ارسال شوند. بعد از اینکه ویژگی‌های درست را تنظیم کردیم، فرآیند باید یک صفحه از حافظه را با وارد کردن گره انتظار (mem_wait) به دست آورد. بعد از اینکه صفحه حافظه به دست آمد، عامل با افزایش مقدار LTRIB[۵] از طریق گره تخصیص GetFrstPg ثبت می‌شود. بعد از اینکه صفحه به دست آمد، پردازش در صف آماده منتظر می‌شود. از صف آماده، پردازش به گره مقدم

(cpu_wait) می‌رود، که در آن یا پردازش فعلی را در اولویت قرار می‌دهد (ATRIB[۳] ارائه شده به آن اولویتی بالاتر از پردازش فعلی می‌دهد) و پردازش مقدم را به صف آماده باز می‌گرداند یا آنقدر منتظر می‌ماند تا فرآیند درحال اجرای فعلی CPU را ترک کند. هنگامی که نهاد کنترل CPU را در اختیار دارد، دنباله مراحل بعدی توسط پردازشی که نهاد آن را شبیه سازی می‌کند، تعیین می‌شود.

انشعاب شدن

مدل‌سازی نهاد برنامه انشعابی ابتدا به گره goon، یعنی fork می‌رود و منشاء آن جایی در گره آزاد (cpu_free۲) است که CPU را آزاد می‌کند و حافظه آن در گره آزاد (mem_free) قرار دارد. هنگامی که هر دو منبع آزاد شوند، نهاد تفکیک می‌شود، یعنی یک نسخه از خودش (والد) را به گره واسط end می‌فرستد و نسخه دیگر خود (فرزند) را به گره اختصاصی child می‌فرستد که در آن، تعداد صفحات را ثبت خواهد کرد. سپس child سعی می‌کند تا اولین صفحه حافظه را با رفتن به گره منتظر، mem_wait به دست آورد.

MATLAB

مدل‌سازی نهاد برنامه MATLAB با توجه به اینکه کاری برای خواندن یا نوشتن دارد، ابتدا به گره goon (processmatlab) از جایی که انشعاب شده می‌رود. اگر کار خواندن و نوشتن وجود نداشته باشد، واحد به گره اختصاصی Ionotreq مسیریابی می‌شود که در آن ATRIB[۱] توسط دوره زمانی کاهش پیدا کرده است. سپس، CPU سراسر گره آزاد (cpu_free) منتشر می‌شود. اگر

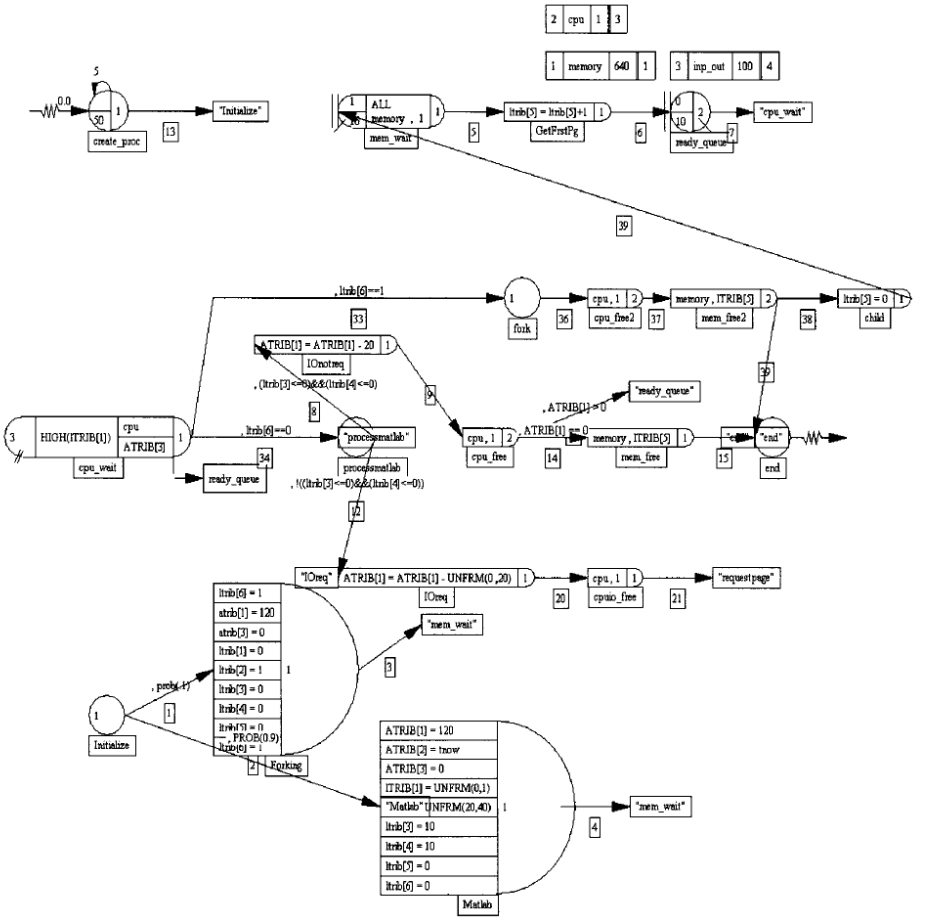
پردازی برای انجام مانده باشد، واحد به صف آماده باز می‌گردد، در غیر این صورت، حافظه خود را از طریق گره آزاد mem_free منتشر می‌کند و با رفتن به گره پایان (end) خاتمه می‌یابد.

بااین‌حال، اگر عملیات خواندن و نوشتن باید انجام شوند، واحد ابتدا به گره اختصاصی Ioreg مسیریابی می‌شود که در آن ارزش ATRIB[۱] بین ۰ و برش زمانی (با توجه به زمانی که I/O درخواست شده است) کاهش می‌یابد که سپس CPU را از طریق گره آزاد cpuio_free از جایی که مسیریابی شده به گره goon (requestpage) مسیریابی می‌کند.

بعد از اینکه درخواست یک صفحه مطرح شد، باید ابتدا تعیین شود که آیا فرآیند متعلق به آن صفحه است یا خیر. این مدل با فرض اینکه احتمال تعلق صفحه برابر با تعداد صفحه حفظ شده تقسیم بر تعداد کل صفحات موردنیاز باشد، با این موضوع سروکار دارد. ارزش این احتمال برای انشعاب‌ها استفاده می‌شود. اگر معلوم شود که این پردازش در واقع صفحه لازم را دارد، واحد به گره goon (GetPageFrame) هدایت می‌شود که یافتن قاب صفحه را شبیه سازی می‌کند و سپس به گره goon (addoffset) مسیریابی می‌شود که نشان می‌دهد به قاب صفحه، آفست افزوده شده است. بعد از addoffset، اگر عمل نوشتن باید انجام شود، واحد به گره goon (write) مسیریابی می‌شود و سپس به گره اختصاصی (DecWrite) می‌رود که در آن برچسب زمانی به دست می‌آید و تعداد نوشتن‌های باقیمانده نیز کاهش می‌یابند. بعد از نوشتن، این واحد به صف حاضر باز گردانده می‌شود. اگر هیچ write باقی نمانده باشد، سپس عمل خواندن باید انجام شود (از آنجایی که اگر عمل خواندن و نوشتن وجود نداشته باشند، ما نیز درباره آن‌ها بحث نمی‌کردیم). این واحد در ابتدا از جایی که به گره

اختصاصی DecRead مسیریابی خواهد شد، به گره goon (read) مسیریابی می‌شود که مشابه گره DecWrite است. بعد از تغییر ویژگی ها، واحد به صف حاضر مسیریابی می‌شود.

اگر در پردازش صفحه ای درخواست نشود، باید LRU پیدا شود و با صفحه موردنیاز مبادله گردد. ابتدا، باید تعیین شود که آیا خود پردازش نیز حاوی LRU است یا خیر، بنابراین واحد ابتدا به گره goon (HaveLRU) مسیریابی می‌شود که در آن از تابع تعریف‌شده توسط کاربر برای تعیین اینکه آیا LRU دارد یا خیر استفاده می کند و بر اساس آن انشعاب بندی انجام می‌شود. اگر LRU را داشته باشد، این واحد تنها باید LRU خود را مبادله کند و صفحه جدیدی بگیرد، این کار از طریق گره goon (swapWzelf) انجام می‌شود که بعد از آن واحد را می‌توان به گره goon (GetPageFrame) مسیریابی نمود. اگر واحد، LRU را نداشته باشد، به یک گره خالی مراجعه می کند تا صفحه را از پردازش با LRU دور کند (این کار به کمک برچسب زمانی انجام می‌شود – ATRIB[۲]) و خودش از آن استفاده نماید. هنگامی که یک صفحه به دست آمد، تعداد صفحاتی که در حال حاضر توسط گره اختصاصی نگهداری می‌شوند، افزایش می‌یابد و نهاد نیز به گره goon (GetPageFrame) باز می گردد. شکل (۱۳-۱) این مدل را برای INUX۷,۲ نشان می‌دهد.



شکل (۱۳-۱): مدلی برای LINUX ۷,۲.

مدل AWSIM برای ویندوز XP

مدل‌سازی سیستم

ما برنامه CPU را بر اساس معماری ویندوز NT مدل‌سازی کردیم. ما یک مدل سطح بالا ایجاد کردیم و مدل AWESIM را بر اساس این مدل سطح بالا پیاده سازی نمودیم. متن زیر به ما نشان می‌دهد که CPU چگونه در سیستم‌عامل برنامه ریزی شده است.

برنامه ریزی CPU

این سیستم‌عامل از سیستم چند منظوره انحصاری استفاده می‌کند. یعنی، به چند فرآیند اجازه می‌دهد تا به‌صورت همزمان اجرا شوند و به سرعت میان آن‌ها سوییچ می‌شود و طوری برنامه ریزی می‌کند که هر پردازش تنها پردازش در حال اجرا روی دستگاه باشد. در این حالت فرض می‌کنیم که یک محیط تک پردازنده داریم.

واحد برنامه ریزی پایه، یک موضوع است. سیستم به هر موضوع یک شماره بین ۱ تا ۳۱ اختصاص می‌دهد که در آن هرچقدر رقم بالاتر باشد، اولویت سیگنال بالاتر است. سیستم اولویت‌های ۱۶ تا ۳۱ را (اولویت‌های بی درنگ) برای عملیاتی که دارای زمان بحرانی هستند اختصاص داده است. یک کلاس اولویت پردازش یک سطح اولویت است که پیرامون آن موضوعات پردازش اجرا می‌شوند. فرآیندهای جدید کلاس اولویت والدین خود را به ارث می‌برند. موضوعات فرآیند در سطح اولویت مربوط به کلاس اولویت فرآیند آن‌ها آغاز می‌شود.

اولویت‌های نسبی که می‌توانند با توجه به موضوع تغییر کنند: یعنی بالاترین اولویت، بالای نرمال، نرمال، زیر نرمال و کمترین اولویت هستند. موضوعات باید در CPU اجرا شوند به نحوی که یک موضوع نمی‌تواند مانع کار موضوعات دیگر شود. یکی از کارهای برنامه ریز، تخصیص واحدهای زمانی CPU

(کوانتوم) به موضوعات است. یک کوانتوم معمولاً دوره زمانی کوتاهی دارد اما موضوعات به‌صورت متناوب کوانتوم را دریافت می‌کنند به نحوی که به نظر می‌رسد سیستم خیلی کند اجرا می‌شود – حتی وقتی که موضوعات زیادی در حال اجرای کار باشند. برنامه ریز باید هر زمانی که یکی از شرایط زیر رخ دهد، برای CPU تصمیم گیری کند.

۱. یک بند کوانتوم روی CPU منقضی شود.

۲. یک بند منتظر رخداد باشد.

۳. یک بند آماده اجرا باشد.

برنامه ریز الگوریتم FindReadyThread را اجرا می‌کند تا تصمیم بگیرد که آیا بند دیگری نیز باید در CPU اجرا شود. اگر بندی با اولویت بالاتر آماده اجرا باشد، با بند در حال اجرا جایگزین می‌شود یا اولویت آن را می‌گیرد. VindReadyThread و ReadyThread الگوریتم‌های کلیدی هستند که برنامه ریزی برای نشان دادن نوبت بندها در CPU استفاده می‌کند. FindReadyThread بندی با بالاترین اولویت برای اجرا را پیدا می‌کند. برنامه ریز بندهای آماده اجرا در فهرست آماده توزیع را ردیابی می‌کند. الگوریتم VindReadyThread لیست آماده توزیع را اسکن می‌کند و بند اول در صف غیرتهی با بالاترین اولویت را پیدا می‌کند. ReadyThread الگوریتمی است که بندها را در فهرست آماده توزیع قرار می‌دهد. هنگامی که ReadyThread یک بند آماده اجرا را پیدا کند، بررسی می‌کند که آیا آن بند اولویتی بالاتر از بند در حال اجرا دارد یا خیر. اگر اولویت بند جدید بالاتر باشد، جای بند فعلی را می‌گیرد و بند فعلی به فهرست آماده توزیع می‌رود.

در غیر این صورت، ReadyThread جای بند آماده اجرا در فهرست آماده توزیع را می‌گیرد. در ابتدای صف، ReadyThread جای بندهایی که برنامه ریز از CPU بیرون آورده را می‌گیرد، بقیه بندهای دیگر (شامل بندهای مسدود شده) به انتهای صف می‌روند.

مدل سطح بالا

شکل (۱۳-۲) مدل سطح بالای برنامه ریز CPU را نشان می‌دهد که در AWESIM اجرا شده است.

مفروضات

ما در حین اجرای مدل شبکه، چند فرضیه ایجاد کردیم که عبارتند از:

۱. پردازش‌هایی که I/O دارند، زمان ثابتی برای انجام آن عملیات دارند.

۲. یک اندازه ثابت کوانتومی.

۳. پردازش‌های مقدم به جای رفتن به ابتدا، به انتهای صف می‌روند.

۴. این مدل تنها عملیات I/O را در نظر خواهد گرفتن. وقفه‌ها و عملیات انشعابی مدنظر نیستند.

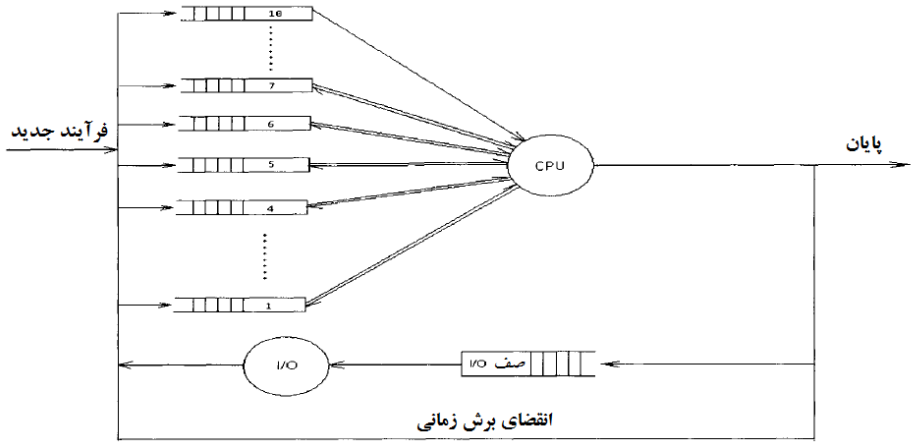
مدل AWESIM

این مدل یک پیاده سازی از مدل سطح بالایی است که پیش از این به آن اشاره کردیم. در ادامه جزئیات کاری مدل را نشان می‌دهیم:

۱. هر پردازش دارای این ویژگی‌ها است: زمان کل پردازش، اولویت ورودی/خروجی و اینکه آیا پردازش یک عمل I/O را انجام می‌دهد یا خیر.

۲. اگر یک پردازش عملیات I/O داشته باشد، در نتیجه زمان بروز و زمان کل برای I/O تخصیص می‌یابند.

۳. پردازش‌هایی که اولویت‌های مختلفی دارند به صف‌های مختلف می‌روند.



شکل (۱۳-۲): مدل سطح بالای برنامه CPU که در AWESIM پیاده سازی شده است

۴. از آنجایی که CPU برای یک کوانتوم به یک پردازش تخصیص می‌یابد، محاسبات برای اندازه گیری زمان باقیمانده پردازش بعد از انقضای کوانتوم انجام می‌شوند.

۵. اولویت بندی بر اساس اولویتی است که از گره دارای اولویت استفاده می‌کند و زمان باقیمانده برای پردازش نیز محاسبه می‌شود.

۶. پردازش‌های اولویت بندی شده به یک صف فرستاده می‌شوند و بعد از آن نیز به صف‌هایی با اولویت مختلف فرستاده می‌شوند.

۷. تمام پردازش‌ها به گره انتظار می‌روند، یعنی جایی که منتظر CPU خواهند شد. هنگامی که CPU در دسترس قرار گیرد، آن‌ها از آن برای کل زمان کوانتوم استفاده می‌کنند یا تا زمانی که یک درخواست I/O یا اولویت بندی رخ دهد.

۸. وقتی که برای I/O درخواست شود، پردازش در گره انتظار I/O منتظر می‌ماند تا یک واحد از منبع I/O به آن تخصیص یابد. پردازش‌هایی که I/O را تمام کنند به صف آماده باز گردانده می‌شوند.

۹. بعد از اینکه از یک منبع (I/O یا CPU) استفاده شود، منبع آزاد می‌شود تا به واحد (پردازش) بعدی تخصیص یابد.

۱۰. بعد از اینکه عملیات I/O انجام شد، وضعیت آن تغییر می‌کند تا نشان دهد که دیگر به عملیات I/O نیازی ندارد.

۱۱. یک پردازش بعد از پایان زمان تخصیص یافته برای آن، پایان می‌یابد.

۱۲. گره‌های جمع آوری برای جمع آوری آماری از قبیل تعداد پردازش‌های اولویت بندی شده، تعداد پردازش‌هایی با اولویت‌های مختلف و تعداد پردازش‌هایی که I/O را انجام می‌دهند، پیاده سازی می‌شوند.

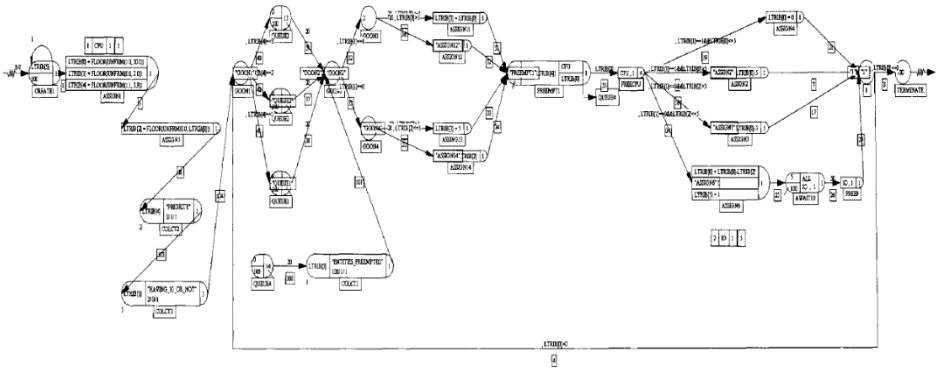
شکل (۱۳-۳) یک مدل AWESIM برای برنامه ریزی CPU را نشان می‌دهد. توصیف طرح مربوط به مدل AWESIM به این صورت است. مدل سطح بالا روی برنامه CPU متمرکز می‌شود، بنابراین منابع موجود در مدل AWESIM عبارتند از CPU و I/O. گره تولید، ۱۰۰ واحد (پردازش) ایجاد می‌کند که با ویژگی‌های مختلفی (و هر ویژگی یک تابع خاص را تعریف می‌کند) تخصیص می‌یابند.

LTRIB[۰] زمان کل اجرای پردازش را به ما می‌دهد، که شامل زمان موردنیاز برای اجرای عملیات I/O نیز هست. اینکه یک پردازش I/O دارد یا خیر، توسط LTRIB[۱] مشخص می‌شود. LTRIB[۲] نیز مربوط به زمانی است که در آن I/O در کل زمان اجرا رخ می‌دهد. هر پردازش یک اولویت دارد که توسط LTRIB[۴] مشخص می‌شود. در مدل ما، پردازشی که به گره تولید می‌رود دارای اولویت ۱، ۲ یا ۳ است. با توجه به اولویت، واحدها به صف‌های مربوط به خود فرستاده می‌شوند. در اینجا واحدها با توجه به مقدار LTRIB[۱] منشعب می‌شوند. زمان موردنیاز پردازش برای اجرا، با برش زمانی موجود در CPU مقایسه می‌شود و به همین ترتیب، منبع برای اجرای فرآیند در دسترس قرار می‌گیرد. اگر در حال حاضر CPU مشغول سرویس‌دهی به پردازشی با اولویت کم باشد و پردازشی با اولویت بیشتر در حال وارد شدن باشد، گره اولویت بندی، پردازش کم اولویت را برای ادامه سرویس‌دهی به صف ۴ می‌فرستد. علاوه بر این، بعد از اینکه پردازش‌هایی با الزامات ورودی/خروجی از CPU سرویس گرفتند، به گره انتظار I/O فرستاده می‌شوند و درنتیجه به صف‌های اولیه می‌روند تا اجرای آن‌ها تکمیل شود. این دو منبع بعد از سرویس، با استفاده از یک گره آزاد، رها می‌شوند.

نتایج مربوط به مدل AWSIM

طی یک گزارش، خلاصه ای از خروجی نشان داده‌شده است که توسط آن می‌توانیم نسبت بهره‌برداری از صف‌های مختلف، طول‌های صف، استفاده از فعالیت‌های سرویس و پارامترهای مربوط به گره‌های دیگر را پیدا کنیم. با توجه به این ارزیابی می‌توانیم میزان بهره برداری از CPU و I/O را نیز پیدا کنیم.

خروجی‌های مربوط به گره جمع‌آوری در هر مرحله از مدل AWESIM ما را به نتایجی می‌رساند که در آن جزئیات مربوط به اولویت‌های مختلف منسوب به پردازش‌ها، تعداد پردازش‌هایی که اولویت‌بندی شده اند و اینکه یک پردازش I/O است یا خیر را به دست می‌آوریم.



شکل (۱۳-۳): مدل AWESIM برای برنامه ریزی CPU

مدل AWESIM برای ویندوز ME

مدل سطح بالا

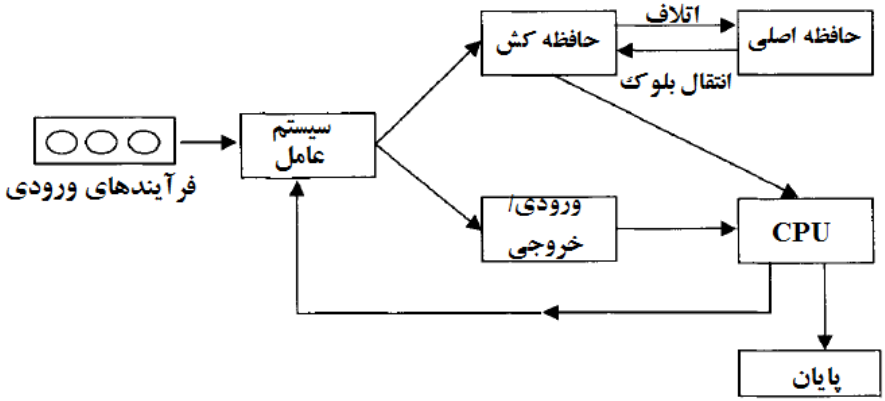
در مدل سطح بالای اصلی برای ویندوز ۹۸، هر فرآیند ورودی به سیستم‌عامل هدایت می‌شود. سیستم‌عامل نیز با توجه به نوع سرویس مورد درخواست، فرآیند را به I/O یا حافظه هدایت می‌کند. اگر درخواست فوق برای حافظه باشد، درنتیجه سیستم‌عامل بررسی می‌کند که آیا داده مورد درخواست در کش وجود دارد یا خیر. اگر اینطور نباشد، بررسی می‌کند که آیا داده در حافظه اصلی است یا خیر و بلوک داده را در کش منتقل می‌کند. اگر درخواست فوق مربوط به حافظه نباشد، پس مربوط به I/O

است. درنهایت، بعد از یک درخواست I/O یا دسترسی به حافظه، در نهایت به CPU می‌رود. بعد از اینکه پردازش در CPU تکمیل شد، می‌تواند وارد زنجیره پایانی I/O یا حافظه برود. شکل (۴-۱۳) مدل شبکه سطح بالا را نشان می‌دهد. مدل شبیه سازی برای ویندوز ME روی قابلیت‌های اصلی دو جنبه متمرکز شده است: دسترسی به حافظه و I/O. این مدل به ساده ترین روش و با جزئیات لازم برای شبیه سازی شرایط واقعی با آن‌ها کار می‌کند. مدل شبیه سازی برای ویندوز روی قابلیت‌های اصلی دو حالت اصلی تأکید دارد. یک گره ایجاد واحد وجود دارد. واحدهای ناشی از آن در سه گره اختصاصی تغذیه می‌شوند که مجموعه ای از ویژگی‌ها را به این واحدها تخصیص می‌دهند تا رفتار آن‌ها در سیستم را از یکدیگر متمایز کنند:

۱. ATRIB[۱] مربوط به نوع سرویس موردنیاز است: دسترسی به حافظه ساده یا I/O.

۲. ATRIB[۲] برای اولویت‌های بین ۵ تا ۱۵ است.

۳. ATRIB[۳] مربوط به تعداد باز نصب‌ها یا تعداد دفعاتی است که واحد برای سرویس‌دهی نیاز دارد.



شکل (۱۳-۴): مدل شبکه سطح بالا

این واحدها از صف گره اختصاصی در صف ورودی عبور می‌کنند و منتظر می‌شوند تا در گره انتظار منابع مورد نظر خود (کش، حافظه اصلی و I/O) را به دست آورند. این منابع بر اساس اولویت پردازش تخصیص می‌یابند. بعد از کسب منابع، آن‌ها در گره انتظار منتظر منبع CPU می‌شوند (پالس ساعت CPU). بعد از سرویس‌دهی CPU، واحدها جابه جا می‌شوند و منابع مربوطه را در گره آزاد، رها می‌کنند و پایان می‌پذیرند. واحدهایی که به چند منبع نیاز دارند، به صف ورودی بازمی گردند. این کار با بررسی ATRIB۳ و تغییر ATRIB۱ برای ایجاد یک رفتار واقعی انجام می‌شود. این این مدل سعی می‌کند که با ایجاد برخی نهادها در داخل پردازش، انشعاب گذاری انجام دهد. این در داخل واحدی که فعالیت‌ها را می‌شمرد دیده می‌شود. شکل (۱۳-۵) مدل AWESIM را برای ویندوز ME نشان می‌دهد.

مدل AWESIM برای ویندوز NT

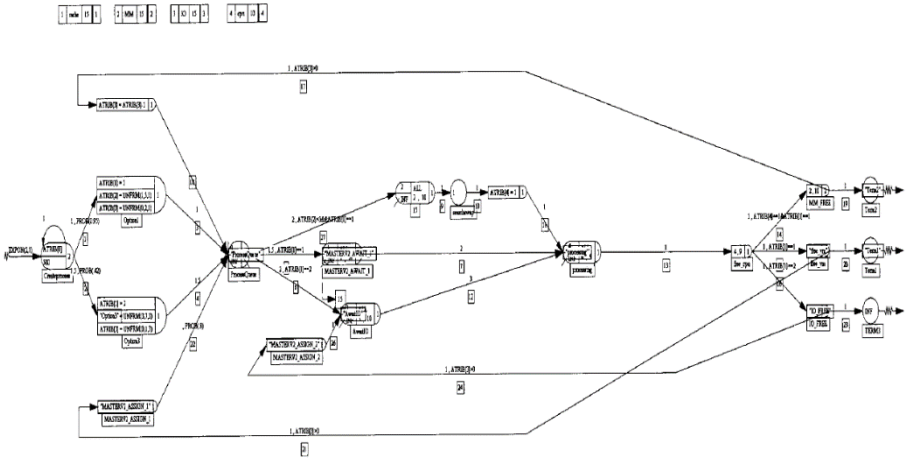
شکل (۱۳-۶) مدل AWESIM را برای ویندوز NT نشان می‌دهد. در هنگام ایجاد مدل AWESIM (بخش حافظه مجازی)، گروه NT همکاری نزدیکی با گروه لینوکس داشتند. گروه NT عمدتاً روی زمان‌بندی پردازش CPU، زمان‌بندی I/O و حافظه مجازی کار می‌کردند. ما وارد جزئیاتی مثل مدیریت فایل و مدیر هدف نشدیم، برای اینکه ورودی‌های سیستم‌عامل در دسترس نبودند. قبل از ورود به مدل واقعی، باید نگاهی به منابعی که مدل‌سازی شده اند ویژگی‌های آن‌ها داشته باشیم. منابع عبارتند از حافظه، CPU و مدیر I/O. از این میان، حافظه و CPU به‌عنوان یک منبع جداگانه از منابع مدل‌سازی شدند، در حالی که مدیر I/O (که دستگاه‌های I/O مختلفی را مدیریت می‌کند) به‌صورت یک منبع گروهی مدل‌سازی شده است، یعنی n منبع دارد. CPU نیز یک واحد اختصاصی دارد، برای اینکه در هر لحظه زمان، تنها یک پردازش می‌تواند CPU را کسب کند، به‌اندازه‌ای که سیستم مدل‌سازی شده صفحه دارد، حافظه واحدهای اختصاصی دارد. صفات آن به شرح زیر است:

ATRIB[۱] = زمان CPU کل موردنیاز توسط یک پردازش.

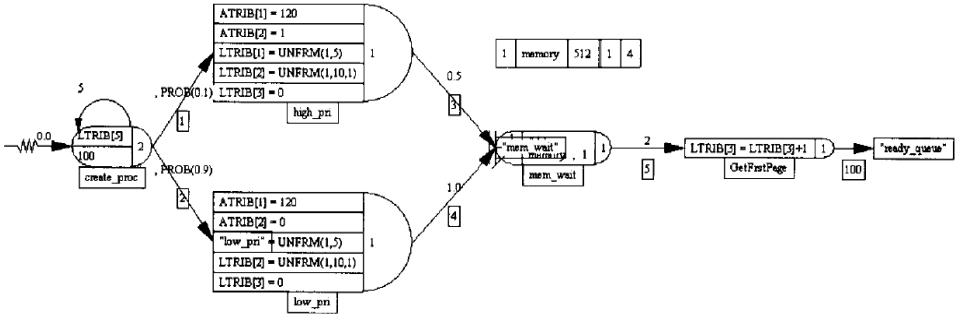
ATRIB[۲] = اولویت یک فرآیند را تعریف می‌کند. این مقدار با استفاده از یک تابع احتمال تخصیص می‌یابد. اگر این مقدار برابر ۱ باشد، درنتیجه فرض می‌شود که این فرآیند متعلق به اولویتی بالاتر از بقیه فرآیندها است و ۰ نیز نشان می‌دهد که اولویت پایین‌تر است. احتمال ۰,۱، نشان می‌دهد که یک فرآیند دارای اولویت ۱ است.

LTRIB[۱] = تعداد کل صفحاتی که یک فرآیند نیاز دارد. دوباره می‌گوییم که برای تخصیص شماره صفحات، از یک تابع دامنه استفاده کردیم (UNFRM) که مقادیری بین ۱ تا ۵ ایجاد می‌کند.

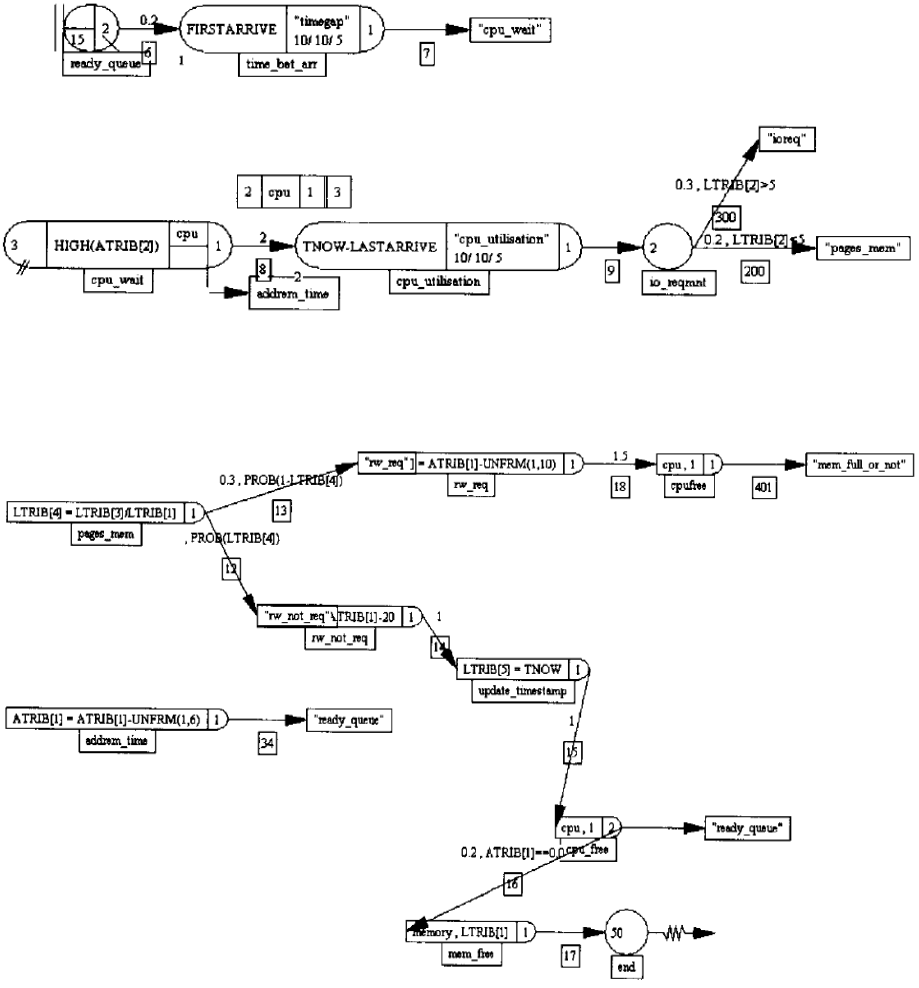
LTRIB[۲] = احتمال اینکه پردازش‌ها می‌تواند به عملیات I/O نیاز داشته باشند مثل، چاپ، انتظار برای یک زیر روال و غیره.



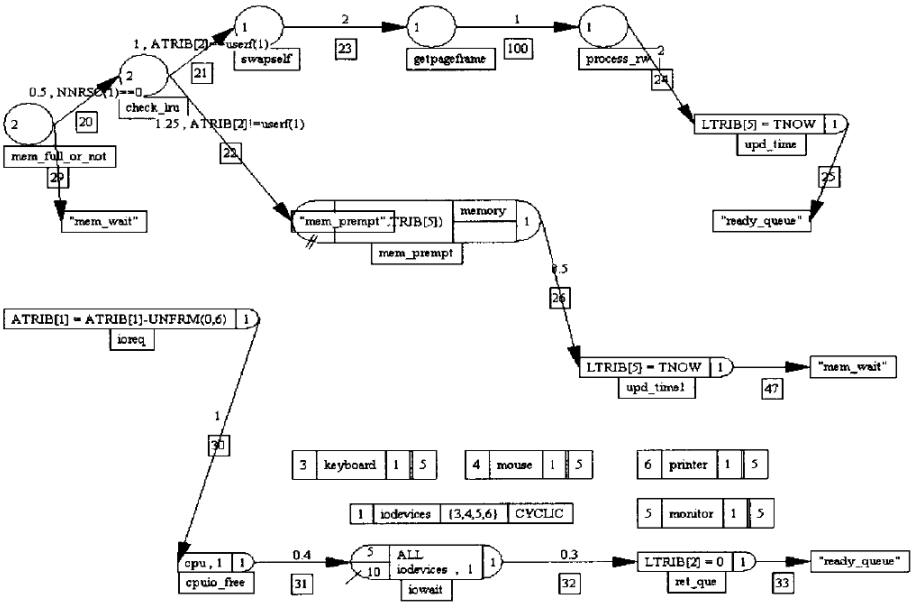
شکل (۱۳-۵): مدل AWESIM برای ویندوز ME



شکل (۱۳-۶): مدل AWSIM برای ویندوز NT



شکل (۱۳-۶) مدل AWSIM برای ویندوز NT



ادامه شکل (۱۳-۶)

LTRIB[۳] = تعداد صفحاتی که حافظه درخواست می‌کند و در ابتدا مقدار آن ۰ است.

LTRIB[۴] = تعداد کل صفحاتی که در حال حاضر در حافظه نگهداری می‌شوند.

LTRIB[۵] = برچسب زمانی است که برای محاسبه LRU استفاده می‌شود.

روش کار LRU به شرح زیر است. هر زمان که مرجع حافظه ایجاد می‌شود، برچسب زمانی به‌روزرسانی می‌شود. بااین‌حال، به جای استفاده از یک برچسب زمانی برای هر صفحه، از یک برچسب برای هر فرایند استفاده می‌شود. منطق می‌گوید که اگر یک فرآیند اخیرا به حافظه مراجعه کرده باشد،

آخرین صفحه مورداستفاده را دارد. فرض کنید X و Y دو فرآیند باشند: X یک صفحه از دست داده و Y نیز LRU دارد. فرآیند X می‌تواند با استفاده از LTRIB[۵]، یک صفحه از فرآیند Y با یک گره اولویت بندی بگیرد تا اولویت را نشان دهد. اما این منطق هر زمان که فرآیند X، LRU را در اختیار دارد دچار مشکل می‌شود برای اینکه خودش نمی‌تواند اولویت بگیرد. این مشکل با استفاده از یک گره HaveLRU و یک تابع تعریف‌شده توسط کاربر رفع شده است که به‌سادگی برچسب زمانی تمام ورودی‌های صاحب حداقل یک صفحه حافظه را آزمایش می‌کند و مقدار برچسب زمانی LRU را باز می‌گرداند. سپس می‌توان از یک شاخه استفاده کرد تا ببینیم برچسب زمانی واحد معادل با برچسب زمانی LRU است یا خیر. در گره HaveLRU، فرآیندها خود را نیز برای داشتن LRU بررسی می‌کند. اگر داشتند، به گره swapself می‌روند، که در آن به‌سادگی عمل مبادله انجام می‌شود، در غیر این صورت، به گره preempt مسیریابی خواهند شد. هر زمان که یک واحد، به همین روش از واحد دیگر تقدم می‌گیرد، باید LTRIB[۳] را یک واحد افزایش دهد (برای اینکه بیش از یک صفحه در اختیار دارد).

مدل شبکه

فرآیندها در گره create ایجاد می‌شوند، create_proc. زمانی که ویژگی‌های صحیح تنظیم شوند، فرآیندها باید با انتظار برای یک گره انتظار، یک صفحه از حافظه را به دست آورند، mem_wait. در ابتدا به تمام پردازش‌ها یک صفحه منفرد از حافظه تخصیص می‌یابد. بعد از کسب صفحه، این عامل با افزایش مقدار LTRIB[۳] از طریق گره اختصاصی GetFrstPage ثبت می‌شود. بعدازاینکه

صفحه به دست آمد، پردازش‌ها در صف حاضر منتظر CPU می‌شوند. بعد از آن وارد گره preempt (cpu_wait) می‌شوند که در آن هم از فرآیند فعلی تقدم می‌گیرند (شمارنده‌های CPU یک فرآیند با اولویت بالاتر از فرآیند فعلی می‌بینند) و آن را به صف انتظار باز می‌گردانند یا آنقدر منتظر می‌شوند تا فرآیند در حال اجرا CPU را آزاد کند. هنگامی که این واحد کنترل CPU را در اختیار بگیرد، مراحل بعدی توسط گره io_reqmnt تعیین می‌شود که نشان می‌دهد به I/O نیاز دارد یا به مقدار LTRIB[۲] بستگی دارد. اینجا دو مشکل مطرح می‌شود. چه می‌شود اگر یک واحد به I/O نیاز داشته باشد یا نداشته باشد؟

اگر واحد به I/O نیاز نداشته باشد، آن‌ها از گره io_reqmnt به سمت pages_mem پیشروی می‌کنند که مقدار LTRIB[۱]/LTRIB[۳] را به LTRIB[۴] تخصیص می‌دهد. منطق انجام این کار، بررسی این نکته است که آیا فرآیند مربوطه تعداد موردنیاز صفحات را با توجه به ارزش LTRIB[۴] دارد یا خیر. اینجا دو مشکل وجود دارد. اگر صفحه دچار خطا شود چه اتفاقی می‌افتد؟ اگر نشود چه؟

اگر خطایی وجود نداشته باشد، درنتیجه این فرآیند پردازش می‌شود و به گره اختصاصی می‌رود که در آن ما ATRIB[۱] و برچسب زمانی را در LTRIB[۵] به‌روزرسانی می‌کنیم، پس از آن، CPU آزاد می‌شود و فرآیندها به صف آماده فرستاده می‌شوند.

اگر خطای صفحه وجود داشته باشد، درنتیجه ما پردازش را از CPU خارج می‌کنیم، برای اینکه CPU منتظر صفحه دیگری می‌شود. بعد از این کار، باید ATRIB[۱] را به روزرسانی کنیم برای

اینکه باید زمانی را در نظر بگیریم که فرآیند در CPU مشغول اجرا شدن است – یعنی، پیش از اینکه خطای صفحه رخ دهد. بعد از بروز رسانی ATRIB[۱]، پردازش به گره mem_full_or_not می‌رود تا پر بودن حافظه موردبررسی قرار گیرد.

اگر حافظه پر نباشد، درنتیجه فرآیند با به‌روزرسانی LTRIB[۳] در یک گره اختصاصی (page_alloc) تعداد موردنیاز صفحه خود را به دست می‌آورد و بعد از آن در گره انتظار (mem_wait) منتظر حافظه می‌شود. اگر حافظه پر باشد، واحد از طریق یک گره check_Iru عبور می‌کند که با استفاده از تابع تعریف‌شده کاربر میزان LRU را محاسبه می‌نماید. اگر این واحد LRU را داشته باشد، این واحد تنها باید صفحه LRU را مبادله کند و در صفحه جدیدی مبادله شود. این کار از طریق گره go-on (SwapSelf) انجام می‌شود که بعد از آن، نهاد می‌تواند از طریق گره GetPageFrame عبور نماید، این کار با شبیه سازی کاوش قاب صفحه و افزودن آفست در آن انجام می‌شود. پس از افزودن آفست، عملیات موردنیاز خواندن/نوشتن با مسیریابی این واحد از طریق گره perform_rw انجام می‌شود. بعد از انجام این کار، واحدها از طریق یک گره اختصاصی به نام upd_time مسیریابی می‌شوند که برچسب زمانی واحدهایی که به حافظه دسترسی دارند را به روز می‌کند. بعد از به روز رسانی برچسب زمانی، واحدها به صف حاضر بازگردانده می‌شوند، برای اینکه عملیات آن‌ها به پایان رسیده است.

اگر واحد فوق LRU را نداشته باشد، به گره preempt یعنی mem_prermpt می‌رود تا صفحه را از پردازش با LRU دور کند و از آن را برای خودش مورداستفاده قرار دهد (توسط برچسب زمانی

LTRIB[۳] انجام می‌شود). بعد از اولویت بندی LRU، برچسب زمانی LTRIB[۵]، دوباره به روز رسانی می‌شود و با انتظارکشیدن برای گره انتظار (mem_wait) درخواست‌های صفحه را از منبع حافظه می‌گیرد.

چه می‌شود اگر واحدها به I/O نیاز داشته باشند؟ واحدهایی که به I/O نیاز دارند، از CPU جدا می‌شوند و از طریق یک گره اختصاصی (ioreq) مسیریابی می‌شوند که زمان کل اجرای سپری شده توسط واحد در CPU را به روز می‌کند. بعد از این، واحدها منتظر یک گره انتظار (iowait) در بلوک I/O منبع گروه می‌شوند که دارای منابع I/O استاندارد مختلفی است مثل مانیتور، ماوس، کیبرد و پرینتر. پس از اینکه واحدها از منبع سرویس گرفتند، ویژگی LTRIB[۲] روی ۰ تنظیم می‌شود، با این فرض که این واحد دیگری نیازی به I/O اضافه ندارد. پس از آن، واحد از یک گره آزاد به نام free_io عبور می‌کند، که در آن تمام منابع به واحدی که آزاد است، تخصیص می‌یابد. درنهایت، واحدها پس از انجام عملیات، به صف حاضر بازگردانده می‌شوند.

چرا مدل AWESIM توسط نتایج آزمایشی معتبر نمی‌شود؟

نتایج AWESIM کاملاً اشتباه بود (به‌عنوان‌مثال، بهره برداری از CPU، ۱۰۰٪ بود)، که بدون شک یک مدل بی اعتبار محسوب می‌شود. با توجه به اینکه بخش‌هایی از این مدل در واقع کنار گذاشته شده بود، جای تعجبی باقی نمی‌ماند. بدون این بخش ها، یک مدل معتبر غیرممکن است، بنابراین به جای تلاش برای متناسب نمودن داده در یک طرح معتبر، در عوض توضیح می‌دهیم که چرا نتایج اینقدر ضعیف هستند و چرا قابل تغییر نمی‌باشند. اولین بخش نادرست داده این است که از CPU به‌صورت

۱۰۰٪ بهره برداری می‌شود. علت این است که ما هیچ دستگاه ورودی/خروجی را شبیه سازی نمی‌کنیم، بنابراین تنها چیزی که باعث می‌شود یک پردازش CPU را رها کند، نابودی یا پایان برش زمانی است. در مورد بعد، فرآیند دیگری به سرعت با آن جایگزین می‌شود، که همین امر باعث می‌شود میزان بهره وری به‌صورت غیرواقعی جلوه کند.

بقیه مسائل نیز در قالب همین مسئله اجرا می‌شوند، برای این ویژگی‌ها نمی‌توانیم مقادیر مناسبی به دست آوریم. به‌عنوان‌مثال ما دوست داریم که تعداد خواندن و نوشتن‌های موجود در برنامه MATLAB را بدانیم اما تمام چیزی که به دست می‌آوریم یک سری عملیات خاص است. این عملیات می‌توانند نمایشگر تعدادی خواندن و نوشتن باشند. این ما را وادار می‌کند که از رقم دلخواهی برای خواندن و نوشتن استفاده کنیم. این، تعداد کل صفحاتی که یک فرآیند درخواست می‌کند یا رقم دلخواه برای آن نیز، برای شبیه ساز شناخته شده نیست. ارقام دلخواه دیگر انتخابی به شرح زیر هستند: دوره برش زمانی، زمان اجرای یک واحد خواندن، زمان اجرای یک واحد نوشتن و طول کل زمان CPU (بدون حساب دفعات I/O) برای انجام پردازش. واضح است که حتی اگر گره‌ها و فعالیت‌های مدل کامل باشند، با این‌همه رقم دلخواه نتایج ما نمی‌توانند معتبر باشند. این مدل از نقطه نظر سطح بالا، برای توضیح نحوه کار سیستم‌عامل لینوکس مفید است و این یک تجربه آموزشی مفید برای طراحی محسوب می‌شود اما در زمینه تعیین رفتار دنیای واقعی این مدل بی فایده است.

۵-۱۳ نتیجه گیری و تحلیل آزمایشی

در این بخش به تحلیل نتایج آزمایشی می‌پردازیم. این نتایج تراکم بار موجود در حالات جداگانه را با یکدیگر مقایسه می‌کند. بر اساس این نتایج، نتیجه گیری نهایی ما مربوط به مبادلات بین سیستم‌عامل‌ها می‌باشد.

۱۳-۵-۱ تراکم بار انتقال فایل

برای این کار، هنگامی که اندازه فایل ثابت است و تعداد فایل نیز خیلی زیاد است، زمان پاسخگویی و استفاده از CPU به طور خطی افزایش می‌یابد، در حالی که استفاده از حافظه نیز برای XP، NT و ME نیز ثابت است. اما در مورد لینوکس، زمان پاسخگویی افزایش می‌یابد اما میزان استفاده از CPU کاهش پیدا می‌کند و میزان استفاده از حافظه نیز ثابت باقی می‌ماند. جدول (۱۳-۵) تا (۱۳-۷) محدوده کاربرد CPU و استفاده از حافظه را با توجه به تعداد فایل برای هر OS نشان می‌دهد.

جدول (۱۳-۵): استفاده از CPU و حافظه برای فایلی با اندازه ۵۰۰

سیستم‌عامل	محدوده CPU	بهره برداری از حافظه
ME	۵۶/۵-۸۸/۸	۱۰
NT	۸۷/۳۷-۹۸/۰۸	۱۰
LINUX	۱۳/۰-۹۹/۹	۰/۳
XP	۱۳/۹۲-۲۶/۷	۱۸

جدول (۱۳-۶): استفاده از CPU و حافظه برای فایلی با اندازه ۷۵۰

سیستم‌عامل	محدوده CPU	بهره برداری از حافظه
ME	۶۲/۰-۸۷/۸	۱۰
NT	۶/۳۵-۹۸/۴۹	۱۰
LINUX	۱۳/۷-۹۹/۹	۰/۴
XP	۱۶/۱۶-۳۲/۷۷	۱۷/۹۹

مشاهدات مربوط به جدول (۱۳-۵): ویندوز ME، NT و XP همگی محدوده استفاده از CPU زیادی دارند، در حالی که در لینوکس این محدود کم است. عملکرد لینوکس برای ۱۰۰ فایل در محدوده ۹۹/۹ و ۷۵/۶ است. برای تعداد ۵۰۰ و ۱۰۰۰ فایل، استفاده از CPU با روندی متفاوت بین ۱۳ و ۱۶/۸ بود.

مشاهدات جدول (۱۳-۶)، ویندوز ME، NT و XP همگی محدوده استفاده از CPU بالایی دارند، درحالی که لینوکس دارای محدوده کمی است، در مورد ۱۰۰ فایل اندازه گیری‌های حاصل بسیار متفاوت بودند. معیارهای استفاده از CPU برای اولین آزمایش ۹۸/۳، برای دومین آزمایش ۲۴/۴ و برای سومین آزمایش ۹۹/۹ بود. بااین‌حال، برای ۵۰۰ و ۱۰۰۰ فایل محدوده آن بین ۱۳/۷ تا ۱۵/۹ بود که واریانس کمتری را نشان می‌دهد. برخی از این تغییرات را می‌توان با انجام آزمایشات بیشتر در هر سطح و میانگین گیری از نتایج کمتر کرد. بااین‌حال، زمانی که در اختیار داشتیم به ما این امکان را نداد.

مشاهدات جدول (۱۳-۷)، ویندوز ME، NT و XP همگی در طیف کاملی از اندازه گیری‌ها دارای محدوده کاربرد CPU بالایی بودند، در حالی که لینوکس محدوده کمی داشت و برای سه اجرای

آزمایشی حدود ۹۸٪ بود. بااین‌حال، برای ۵۰۰ و ۱۰۰۰ فایل، محدوده استفاده از CPU بین ۱۲/۹ تا ۱۵/۶ بود.

جدول (۱۳-۷): استفاده از CPU و حافظه برای فایلی با اندازه ۱۰۰۰

سیستم‌عامل	محدوده CPU	بهره برداری از حافظه
ME	۶۰/۰-۸۷/۸	۱۰
NT	۹۶/۱۱-۹۷/۶۸	۱۰-۱۱
LINUX	۱۲/۹-۹۸/۲	۰/۵
XP	۱۸/۱۵۶-۳۰/۹۱	۱۷/۹۸

جدول (۱۳-۸): استفاده از CPU و حافظه برای سرعت پردازش = ۰

سیستم‌عامل	محدوده CPU	بهره برداری از حافظه
ME	۶۱/۶۷-۹۱/۲	۱۰-۱۲
NT	۳/۳۸-۸۵/۶۲۵	۱۱-۲۶
LINUX	۲۵-۷۵	۰/۱
XP	۴۷-۸۶	۱۷/۲۳-۱۸

۱۳-۵-۲ تراکم بار ایجاد فرآیند

برای تراکم بار ایجاد فرآیند، متوجه شدیم که میزان استفاده از CPU برای ویندوز XP و لینوکس خطی است با شیبی رو به افزایش، در حالی که NT و ME در بهره برداری از CPU

به طور گسترده ای متفاوت هستند. این، برای XP، هنگامی که تعداد نرخ پردازش‌های افزایش پیدا می‌کند، زمان پاسخگویی کاهش می‌یابد. برای هر تعداد از پردازش ها، محدوده عملکرد بالاتر از مقدار قبل است اما با افزایش نرخ پردازش ها، کاهش می‌یابد. جدول (۱۳–۸) تا (۱۳–۱۰) محدوده بهره برداری از CPU و حافظه را با توجه به نرخ پردازش برای هر OS نشان می‌دهد.

مشاهدات جدول (۱۳–۸): میزان بهره برداری از CPU در ویندوز NT برای ۱۰ پردازش حدود ۲۲٪ است اما بین ۷۷ تا ۸۵ درصد برای ۱۰۰ پردازش افزایش می‌یابد و برای ۱۰۰۰ پردازش سقوط ۵۰٪ دارد. این برای ۱۰۰۰ پردازش، میزان بهره برداری از حافظه حتی با وجود کاهش بهره برداری از CPU، افزایش نشان می‌دهد.

جدول (۱۳–۹): استفاده از CPU و حافظه برای نرخ پردازش = ۱۰۰

سیستم‌عامل	محدوده CPU	بهره برداری از حافظه
ME	۶۰/۰۵–۷۳/۲۵	۱۰–۱۴/۵۶
NT	۱۲/۳۳–۳۱/۵۹	۱۱–۱۵
LINUX	۳۷/۱–۷۳/۳	۰/۱
XP	۷۷/۳–۸۵/۲۷	۱۷/۹۸–۱۸/۹۹

جدول (۱۳–۱۰): استفاده از CPU و حافظه برای نرخ پردازش = ۱۰۰۰.

سیستم‌عامل	محدوده CPU	بهره برداری از حافظه

ME	۵۶/۱۳–۵۸/۸۵	۱/۳۲۲–۱۱
NT	۳/۴۲–۴/۶۲	۱۱–۲۰/۹۸
LINUX	۴۱/۶–۷۴/۱	۰/۱
XP	۱۸/۱۵۶–۳۰/۹۱	۱۷/۹۷–۱۷/۹۹

جدول (۱۳–۱۱): استفاده از CPU و حافظه برای ماتریسی با اندازه = ۱۰ در ۱۰.

سیستم‌عامل	محدوده CPU	بهره برداری از حافظه
ME	۵۱/۵–۷۶/۶۶	۲–۲/۲۵
NT	۲–۷۴	۱۶
LINUX	N/A	N/A
XP	۴/۵–۵۶/۶۷	۱۷/۹۷–۱۷/۹۹

جدول (۱۳–۱۲): استفاده از CPU و حافظه برای ماتریسی با اندازه ۵۰ در ۵۰

سیستم‌عامل	محدوده CPU	بهره برداری از حافظه
ME	۵۵–۹۹/۷۸	۴–۱۱/۹۷
NT	۱۴–۹۹/۶۲	۱۶–۱۸
LINUX	۴۶–۴۸/۳۷	۴/۲
XP	۷/۶۶–۹۹/۱۷	۲۸–۳۰/۹۷

مشاهدات جدول (۱۳-۹)، برای مورد یک نرخ پردازش ۱۰۰، ویندوز NT به طور کلی به‌صورت خطی افزایش می‌یابد به جز برخی مقادیر که در آن میزان بهره برداری از CPU خیلی کم است. بااین‌حال، به نظر می‌رسد که بهره برداری از حافظه نیز به‌صورت خطی افزایش یابد. برای لینوکس، نرخ پردازش برای چند پردازش با تعداد ۱۰ و ۱۰۰ بین ۳۷ تا ۴۱ است اما برای تعداد ۱۰۰۰ پردازش، میزان بهره برداری بین ۵۰ تا ۷۰ درصد خیز دارد.

مشاهدات جدول (۱۳-۱۰)، ویندوز NT و XP در بخش CPU تقریبا ثابت هستند و اندکی نوسان دارند (حدود ۳ تا ۴ درصد). تنها سیستم‌عاملی که در میزان بهره برداری از CPU آن واریانس دیده می‌شود، لینوکس است که بین ۴۱/۶ تا ۷۴/۱ می‌باشد. در مقابل، ME با افزایش در تعداد پردازش‌ها، در میزان بهره برداری از CPU آن یک کاهش دیده می‌شود. با افزایش تعداد پردازش‌ها، میزان استفاده از حافظه نیز به طول قابل توجهی افزایش می‌یابد (حداکثر ۸٪ در NT).

۱۳-۵-۳ تراکم بار MATLAB

برای عملیات ماتریسی در تراکم بار MATLAB، با وجود ماتریسی با اندازه ثابت و تعداد متفاوت، زمان پاسخ و بهره برداری از CPU در مورد ویندوز ME، NT و XP افزایش می‌یابد، در حالی که میزان بهره برداری از حافظه نسبتاً ثابت است (به جدول (۱۳-۱۱) تا (۱۳-۱۴) مراجعه کنید).

مشاهدات جدول (۱۳-۱۱)، ویندوز NT و XP عملکرد مشابهی دارند و هنگامی که تعداد ماتریس‌ها از ۱۰ به ۱۰۰ می‌رسد، میزان بهره برداری از CPU آن‌ها به‌صورت نمایی افزایش می‌یابد. این واریانس در ME اندکی کمتر است و بین ۵۱ تا ۷۷ درصد می‌شود. حتی برای ۱۰ ماتریس، توان محاسباتی خیلی زیادی مصرف می‌کند. برای سیستم‌عامل لینوکس، این آزمایش با پارامترهای مختلفی انجام گرفت. اندازه ماتریس از ۵۰ شروع و به بیش از ۱۰۰۰ رسید، در حالی که برای سیستم‌عامل‌های ویندوز، از ۱۰ شروع و بیش از ۱۰۰ ماتریس رسیده بود. مقدار حافظه مصرفی در سیستم ME، نسبت به سیستم‌عامل‌های ویندوز دیگر، به طور قابل توجهی کم بود (۲,۵ درصد). از سویی دیگر، درصد واریانس در تمام موارد نسبتاً ثابت بود.

مشاهدات جدول (۱۳-۱۲)، هنگامی که تعداد ماتریس‌ها از ۱۰ به ۱۰۰ می‌رسد، درصد استفاده از CPU برای ویندوز NT و XP به‌صورت نمایی افزایش می‌یابد. این تغییرات در ME اندکی کمتر است یعنی بین ۵۵ تا ۱۰۰ درصد است. جالب توجه است که ME حتی برای تعداد کمتری ماتریس، CPU بیشتری مصرف می‌کند. برای سیستم‌عامل لینوکس، مصرف CPU تغییرات زیادی نداشت اما برای تعداد کمی ماتریس، توان محاسباتی زیادی (مشابه ME) مصرف می‌کند. مقدار حافظه مصرفی در سیستم‌عامل لینوکس در مقایسه با سیستم‌عامل‌های ویندوز، به طور قابل توجهی (۴,۲٪) کمتر بود. از سوی دیگر، درصد تغییرات در عملکرد در همه آن‌ها نسبتاً ثابت بود.

جدول (۱۳-۱۳): استفاده از CPU و حافظه برای ماتریسی با اندازه = ۱۰۰ در ۱۰۰

سیستم‌عامل	محدوده CPU	بهره برداری از حافظه
ME	۵۹-۹۹,۹۸	۰,۳۰-۴,۲۵
NT	۳۹,۵۰-۱۰۰	۱۶-۲۴,۹۸
LINUX	۸۱,۲۲-۸۱,۹۷	۴,۲
XP	۷,۳-۱۰۰	۲۸-۴۱

جدول (۱۳-۱۴): استفاده از CPU و حافظه برای ماتریسی با اندازه = ۱۰۰۰ در ۱۰۰۰

سیستم‌عامل	محدوده CPU	بهره برداری از حافظه
LINUX	۹۶,۷۶-۹۶,۹۵	۱۷,۹۵-۱۹,۵

مشاهدات جدول (۱۳-۱۳)، هنگامی که تعداد ماتریس حدود ۱۰۰۰ باشد، میزان استفاده از CPU در هر سه دستگاه مبتنی بر سیستم‌عامل ویندوز به حد اشباع می‌رسد. در این حالت نیز، ویندوز ME حتی برای تعداد کمی ماتریس، منابع CPU بیشتری مصرف می‌کند. برای سیستم‌عامل لینوکس، مصرف CPU خیلی متفاوت نبود اما هنوز مقدار قابل توجهی از توان محاسباتی را برای تعداد کمی ماتریس استفاده می‌کند (مشابه ME). مقدار حافظه مصرفی در سیستم‌عامل لینوکس نسبت به سیستم‌عامل‌های ویندوز به طور قابل توجهی کمتر بود (۴,۲٪).

مشاهدات جدول (۱۳-۱۴)، تنها گروه لینوکس یک ماتریس از این اندازه را مورد آزمایش قرار دارند. در این مورد، استفاده از CPU تقریبا در محدوده قرار داشت. استفاده از حافظه نیز نسبت به دستگاه‌هایی با سیستم‌عامل ویندوز کمتر بود.

۱۳-۵-٤ نتیجه گیری نهایی

جدول زیر با محاسبه میانگین مقادیر آماری برای جدول‌های واقعی جمع آوری شده در این مطالعه برای سیستم‌عامل‌های مختلف، به دست آمده است. معیار مورداستفاده، نسبت استفاده از CPU تقسیم بر حاصل مصرف حافظه و زمان پاسخگویی برای هر آزمایش است. جدول (۱۳-۱۵) نتایج لازم برای آزمایش پردازش را نشان می‌دهد.

جدول (۱۳-۱۵): نتایج آزمایشات پردازش

سیستم‌عامل	زمان پاسخگویی (ms)	بهره برداری از ٪ CPU	بهره برداری از حافظه ٪	CPU / (Mem *res)
XP	۳۵,۲۹۰,۹۵	۸۲,۳۴	۱۹,۴۳	۱۲E-۰۵
ME	۱۵۰,۱۱۰,۳۳	۶۸,۹۸	۸,۹۶	۵,۱۳E-۰۵
NT	۱۴۸,۹۵۶,۵۲	۲۷,۰۳	۱۳,۹۱	۱,۳E-۰۵
LINUX	۷,۴۰۰,۵	۴۹,۸۳	۰,۱	۶۷۳۳,۳E-۰۵

جدول (۱۳-۱۶): نتایج برای آزمایشات MATLAB

سیستم‌عامل	زمان پاسخگویی (ms)	بهره برداری از ٪ CPU	بهره برداری از حافظه ٪	CPU / (Mem *res)
XP	۲۱۸,۶۶	۹۹,۵۶	۳۵,۹۷۵	۰,۰۱۲۶۵۷

ME	۲۲۴,۵۵۵	۹۹,۷۱۵	۲۶,۴۵	۰,۰۱۶۷۸۹
NT	۲۲۲,۳۴۵	۹۸,۷	۲۱,۴۶	۰,۰۲۰۶۸۵
LINUX	-	۶۵,۱۷	۴,۲	۰,۰۶۹۹۴۱*

* برای اینکه یک مقدار برای این جدول به دست آوریم، زمان پاسخ برای LINUX لازم است. زمان پاسخ برای سیستم‌عامل‌های دیگر به طور میانگین است. به این ترتیب زمان پاسخ LINUX اساساً مؤثر نمی‌باشد.

جدول (۱۳-۱۶) نتایج مربوط به آزمایشات MATLAB را نشان می‌دهد. جدول (۱۳-۱۷) نیز نتایج مربوط به آزمایشات فایل را نشان می‌دهد.

هیچ سیستم‌عاملی ازنظر عملکرد در زمینه تراکم بارهایی که در این مطالعه مورداستفاده قرار گرفتند، غالب نبودند. هر یک از سیستم‌عامل‌ها به شیوه خود، نسبت به سیستم‌عامل‌های دیگر برتری داشتند. برای تعیین اینکه عملکرد کدام یک برای تراکم بار بهتر بوده است، ما از فرمول ستون آخر استفاده کردیم که برابر است با استفاده از CPU تقسیم بر حاصل استفاده از حافظه و زمان پاسخ. برای انشعاب فرآیندهای جدید، عملکرد لینوکس خیلی خوب بود. این ارزیابی از طریق مقادیر موجود در جدول برای آزمایشات فرآیند حاصل شده است که در آن از حداقل حافظه، میانگین مصرف CPU و حداقل زمان پاسخگویی نسبت به سیستم‌عامل‌های دیگر استفاده شده است و بهترین مقادیر را برای ارزیابی عملکرد به ما داده است. دومین سیستم مبتنی بر این ارزیابی، سیستم‌عامل XP است که با یک فاکتور ۵۶۱، دارای بدترین مقدار است. در ادامه NT مطرح می‌شود که با عامل ۹,۲ دارای بدترین مقدار می‌باشد. در نهایت، ME بدترین است و در مقایسه با NT دارای عامل ۳,۹ می‌باشد.

جدول (۱۳-۱۷): نتایج مربوط به آزمایشات فایل

سیستم‌عامل	زمان پاسخگویی (ms)	بهره برداری از ٪ CPU	بهره برداری از حافظه ٪	CPU / (Mem *res)
XP	۱۶۴,۱۹۴	۶۷,۷۴	۱۷,۹۵	۲,۲۹۸۳۹E-۰۵
ME	۹۶۵,۳۶	۷۴,۰۵	۱۰	۷۶۷,۰۷۱۴E-۰۵
NT	۱۲۹,۰۳۰	۹۶,۰۶	۱۰,۲۹	۷,۲۳۴۹۷E-۰۵
LINUX	۴۲,۶۲۵,۸۸	۸۹,۲۸	۲۲,۶	۹,۲۶۷۷۱E-۰۵

برای آزمایشاتی که شامل عملیات ماتریسی در MATLAB هستند، عملکرد لینوکس از بقیه سیستم‌عامل‌ها بهتر است، برای اینکه در این سیستم‌عامل از حداقل مقدار حافظه استفاده می‌شود و میزان بهره برداری از CPU در عملکرد کلی معادل ۰,۰۶۹۹۴۱ است و این نشان می‌دهد که با ضریب ۳,۳۸ بهتر از NT می‌باشد. NT با ضریب ۱,۲۳ بهتر از ME است و ME با ضریب ۱,۳۳ بهتر از XP است. ویندوز ME در مقایسه با سیستم‌عامل‌های دیگر با معیار کارایی ۰,۳۰۷,۶۷E فایل‌ها را به‌صورت موثرتری مدیریت می‌کند. این معیار روی لینوکس نیز اجرا شد و نشان داد که با داشتن ضریب ۸۲,۷ عملکرد نامطلوبی دارد. در عوض NT نیز با داشتن عامل ۱,۲۸ بدتر از لینوکس است و XP نیز با داشتن عامل ۳,۱۴ بدتر از NT است.

۱۳-۵-۵ نتایج جدولی

جدول‌های (۱۳-۱۸) تا (۱۳-۲۰) نتایج مربوط به تراکم بارهای مختلف را نشان می‌دهند.

جدول (۱۳-۱۸): داده‌های میانی برای تراکم بار ۱

فایل‌ها		ویندوز XP			ویندوز ME			ویندوز NT			لینوکس		
انداز	تعدا	RT	CP	ME	RT	CP	ME	RT	CP	ME	RT	CP	ME
۵۰۰	۱۰۰	۳,۷۰۷	۱۵,۲۶	۱۸	۱۲۵,۳۳	۶۰,۳۳	۱۰	۴۹۰,۳	۸۹,۹۶	۱۰	۹۲۸,۶	۹۱,۴	۱۲
۵۰۰	۵۰۰	۱,۶۶۱	۲۰۸	۱۷,۹۴	۷۸۶,۶۷	۷۵,۰۸	۱۰	,۳۰۹	۹۵,۶۴	۱۰	۱۷۶,۶	۱۵,۷۳	۰,۳
۵۰۰	,۰۰۰	۱,۸۹۹	۲۶,۴	۱۷,۹۹	۱,۹۸۵,۳	۸۷,۶	۱۰	۵۸۲,۳	۹۷,۸۳	۱۰	۵,۸۷۵	۱۴,۷۳	۰,۳
۷۵۰	۱۰۰	۵۸۹,۷	۶۲,۸۳	۱۷,۹۹	۱۲۴,۶۶	۶۲,۸۳	۱۰	۱۲۱,۶	۹۷,۷۵	۱۰	۴۵۰,۶	۷۴,۲	۰,۴
۷۵۰	۵۰۰	۷۴۳,۴	۲۳۷	۱۷,۹۹	۷۹۰	۷۵,۴۷	۱۰	,۲۸۶	۹۶,۷۷	۱۰	۴,۰۸۰	۱۵,۴	۰,۴
۷۵۰	,۰۰۰	۳۷۳,۷	۳۰,۶	۱۷,۹	,۹۷۴,۶۷	۸۴,۶۵	۱۰	۶۶۱,۷	۹۶,۶۵	۱۰	۸,۱۰۰	۱۴,۵	۰,۴
,۰۰۰	۱۰۰	۶,۳۲۴	۱۸,۵	۱۷,۹۷	۱۲۶,۳۳	۶۱,۱۶	۱۰	۴۳۳,۶	۹۶,۷۱	۱۰,۹۸	۶۵۴,۳	۹۷,۷۳	۱۴,۷
,۰۰۰	۵۰۰	۴۸۹,۳	۲۶,۶	۱۷,۹	۷۹۳	۷۶,۰۹	۱۰	,۶۱۸	۹۶,۵۹	۱۰,۹۸	۵,۳۹۳	۱۴,۸۳	۸۲,۶
,۰۰۰	,۰۰۰	۹۵۸,۷	۳۰,۹	۱۷,۹	۱,۹۸۲,۳	۸۳,۲۹	۱۰	,۷۶۷	۹۶,۶۷	۱۰,۶۷	۹۷۴,۶	۱۵	۹۲,۳

جدول (۱۳-۱۹): داده‌های میانی برای تراکم بار ۲

فرآیندها		ویندوز XP			ویندوز ME			ویندوز NT			لینوکس		
تعداد	نسبت	R	CP	ME	RT	CP	ME	RT	CP	ME	RT	CP	ME
۱۰	۰	۶۰۷,	۴۷,۹۱	۱۷,۲	۲۷۱	۶۳,۷۸	۱۰	۳۰۷,۳	۲۲,۳	۱۱	۲۱۲,۶	۴۳,۴	۰,۱
۱۰	۱۰۰	۵۸۴	۶۳,۱۶	۱۸	۰۹۵,۱	۷۱,۰۸	۱۰	,۲۱۷	۱۵,۵۱	۱۱	۲۰۶	۳۵,۹۳	۰,۱
۱۰	۱,۰۰۰	۶۳۰,	۶۱,۴۵	۱۷,۲	۰,۰۶۳	۵۸,۹	۱۰	,۲۳۵	۳,۱۸	۱۱	۱۲۲	۶۶,۶۷	۰,۱
۱۰۰	۰	۹۳۶	۱۰۶,۴	۱۸,۵	,۰۷۲,۳	۸۷,۳	۱۱,۵	,۴۴۵	۷۹,۹۴	۱۲	۳۱۵,۳	۴۱	۰,۱
۱۰۰	۱۰۰	۲۱۲,	۸۳,۴۵	۱۷,۹	,۵۰۲,۶	۶۵,۹۹	۱۰,۶	۵۵۱,۱	۲۸,۷۸	۱۱	۲,۳۰۸	۳۸,۱	۰,۱
۱۰۰	۱,۰۰۰	۶۱۳,	۸۶,۷۶	۱۷,۹	,۰۵۱۳	۵۶,۸۲	۱۱	۳۶۱,۶	۳,۸۲	۱۱,۹	۳۳۵,۷	۶۴,۶۷	۰,۱
,۰۰۰	۰	۰۷۶,	۹۶,۷۲	۲۴	۲,۴۲۲	۹۸,۷	۱,۵۸	۵۲۲,۶	۵۵	۲۴,۲۲	۲,۳۲۹	۴۲,۳	۰,۱
,۰۰۰	۱۰۰	۸۹۴	۹۷,۰۹	۲۰,۹	۱,۸۷۸	۶۱,۶۴	۱۳,۲	,۳۶۱	۳۰,۳۹	۱۳,۸۰	۳۶۱,۳	۴۲,۶	۰,۱
,۰۰۰	۱,۰۰۰	۰۶۴,	۹۸,۱	۲۲,۹	۰,۱۷۶	۵۶,۶۹	,۶۴۷	۶۰۷,۶	۴,۴	۱۹,۳	۴۱۴,۷	۷۳,۸۳	۰,۱

جدول (۱۳-۲۰): داده‌های میانه برای تراکم بار ۳

ماتریس		ویندوز XP			ویندوز ME			ویندوز NT			لینوکس		
انداز	نرخ	RT	CP	ME	RT	CP	ME	RT	CP	ME	RT	CP	ME
۱×۱۰	۱۰	۰,۰۱	۵,۷	۲۹	۰,۰	۵۸,۸۳	۲	۰,۰۱۵	۱۲	۱۶	۰,۰۰۳	–	–
۱×۱۰	۱۰۰	۰,۰۵	۹,۸۳	۲۸	۰,۰۵	۵۳,۵	۲	۰,۰۶	۱۳,۳	۱۶	۰,۰۳۲	–	–
۱×۱۰	۱۰۰۰,	۰,۷۹	۴۷,۷۲	۲۸	۰,۸۴	۷۳,۶۳	۲,۴۴	۰,۸۹۸	۷۱	۱۶	۰,۵۳۸	–	–
۵×۵۰	۱۰	۰,۰۷	۸,۹	۲۸	۰,۰۷	۵۶,۵	۴	۰,۰۶۳	۱۹,۳	۱۶	۰,۰۶	۴۷,۰	۴,۱
۵×۵۰	۱۰۰	۰,۵۷	۴۲,۳۳	۲۸	۰,۶۸	۷۳,۶۳	۴,۳۳	۰,۶۴۰	۷۷,۵	۱۶	۰,۶۸	۴۷,۸۹	۴,۱
۵×۵۰	۱۰۰۰,	۴۸,۵	۹۹,۱۲	۳۰,۹۷	۵۰,۳	۹۹,۴۶	۱۱,۹	۴۹,۹۳	۹۹,۴	۱۷,۹۹	۵۰,۷۵	۴۸,۳۷	۴,۲
×۱۰۰	۱۰	۰,۳۲	۳۴۰,۷	۲۸	۰,۳۳	۶۲,۵	۲,۳۳	۰,۳۲	۳۸,۹	۱۶	۰,۶۱	۸۱,۳۴	۴,۲
×۱۰۰	۱۰۰	۲,۹۲	۷۹,۹۲	۲۸,۸۷	۳,۴۶	۸۹,۳۰	۴,۰۸	۳,۱۴	۱۹,۰	۱۶,۸	۵,۲	۸۱,۷۶	۴,۲
×۱۰۰	۱۰۰۰,	۳۸۸,	۱۰۰	۴۰,۹۸	۳۹۸,	۹۹,۹۷	۴۱	۳۹۴,۷	۹۷,۹	۲۴,۹۳	۴۱۷,۷	۸۱,۹۷	۴,۲

۱۳-۶ خلاصه

این فصل نتایج نتایج دوره‌های مختلف ارزیابی عملکرد سیستم کامپیوتری را نشان می‌دهد که در دانشگاه دارتموث ماساچوست برگزار شده است. این آزمایشات برای نشان دادن مشکلات مرتبط با ارزیابی عملکرد سیستم‌های کامپیوتری دنیای واقعی، به ویژه سیستم‌عامل‌های آن‌ها انجام‌شده اند. این مطالعه سعی دارد یک ارزیابی از چهار سیستم‌عامل نشان دهد. این آزمایشات آزمایشاتی که ما سعی در انجام آن‌ها داریم نیز فراهم کردند اما توانایی ما برای جمع آوری اندازه گیری‌های قابل اطمینان باعث شد که در این مطالعه با هر درجه ای از اطمینان، ناتوان باشیم. اگر قرار باشد که این مطالعه دوباره انجام شود، گروه ما باید از مفاهیم مانیتورینگ سخت‌افزاری استفاده کنند تا در پارامترهای سیستم سطح پایین درک کاملی از نحوه اجرای سیستم‌ها به دست آوریم. این تیم این تجربه ای در زمینه اجرای این آزمایشات و تحلیل نتایج نداشتند که همین امر نیز مشکلاتی به همراه داشت. این مطالعه به این منظور انجام شد تا مشکلات مربوط به انجام این آزمایشات خود را بیشتر از پیش نشان دهند.

فصل چهاردهم

تحلیل عملکرد سیستم‌های پایگاه داده

۱۴-۱ مقدمه

در فصل قبل به مشکلات مربوط به ارزیابی سیستم‌عامل‌ها رسیدگی کردیم. تمرکز آن روی استفاده از پلتفرم‌ها و نرم افزار تجاری برای ارزیابی عملکرد نسبی ۴ سیستم‌عامل بود. این فصل از روشی مشابه استفاده می‌کند، ما از محصولات نرم افزاری صنعتی مورداستفاده در بسیاری از برنامه‌های کاربردی استفاده کردیم. این درباره پردازش معاملات آنلاین پایگاه داده به‌عنوان دامنه کاربردی کلی بحث خواهیم کرد. بااین‌حال، تمرکز اصلی ما روی ارزیابی چهار سیستم پایگاه داده تجاری است که روی مجموعه ثابتی از پلتفرم سخت‌افزاری و نرم افزار سیستم‌ها (سیستم‌عامل) اجرا می‌شود. هر ۴ سیستم پایگاه داده ای که در حال حاضر برای کسب اولین رتبه در بازار پایگاه داده رقابت می‌کنند، ادعا می‌کنند که در دنیا بالاترین سرعت را دارند. این پایگاه‌های داده عبارتند از IBM DB۲، Informix UDB، Microsoft SQL Server و Oracle ۸i. این فصل نشان می‌دهد که کدام یک از این چهار سیستم پایگاه داده هم ازنظر سرعت و هم ازنظر هزینه بهترین است.

این فصل به سه بخش اصلی تقسیم می‌شود. ابتدا، توصیفی از چهار سیستم داده می‌شود که برای آزمایش پایگاه داده مورداستفاده قرار می‌گیرد. در بخش بعد، نتایج معیار عملکرد یک PC نشان می‌دهد که تمام پیکربندی‌های سخت‌افزاری PC یکسان هستند. بخش دوم روش‌های اتخاذ شده توسط گروه ارزیابی هر پایگاه داده برای اجرای یک معیار استاندارد در پایگاه داده آزمایشی آن را به‌صورت مفصل

نشان می‌دهد. درنهایت، نتایج معیار استاندارد و تحلیل هزینه نشان می‌دهند که کدام پایگاه داده بهترین است.

۱۴-۲ سیستم‌های پلتفرم

برای اینکه مطمئن شویم تمام پایگاه‌های داده روی سطحی از زمینه اجرایی مورد آزمایش قرار گرفته اند، از ۴ سیستم کامپیوتری شخصی با سخت‌افزار یکسان استفاده کردیم. سخت‌افزار هر یک از این دستگاه‌ها در جدول (۱۴-۱) نشان داده‌شده است.

جدول (۱۴-۱): پیکربندی پلتفرم

پردازنده	Pentium III @ ۵۰۰ MHz
RAM کلی	۲۵۶ MB
سیستم‌عامل	Windows NT ٤,۰ Service Pack ٥

برای حصول اطمینان از یکسان بودن دستگاه‌ها ازنظر قابلیت‌های کاربردی، یک گروه ارزیابی عملکرد مستقل آزمایشی انجام داد که طی آن معیار عملکرد یک PC برای اعتبارسنجی مشخصات عملکرد دستگاه پیش از اجرای واقعی، موردبررسی قرار گیرد. در بخش بعد، نتایج مربوط به این آزمایش مطرح می‌شود.

۱۴-۲-۱ معیار ارزیابی عملکرد

آزمون بررسی عملکرد معماری کامپیوتر شخصی از ۲۲ آزمون معیار جداگانه تشکیل شده است که در ۶ بسته آزمون در دسترس قرار دارد. ۶ بسته آزمون به شرح زیرند:

- عملیات ریاضی اعشاری و صحیح
- آزمون توابع گرافیکی دو بعدی استاندارد
- خواندن، نوشتن و کاوش در فایل‌های دیسک
- دسترسی و تخصیص حافظه
- آزمون‌های MMX (ضمایم چندرسانه ای) در CPUهای جدیدتر
- یک آزمون از سیستم گرافیکی DirectX ۳D

گزارش حاصل از نتایج آزمون به‌صورت مقادیر نسبی نشان داده‌شده است. هرچقدر این ارقام بزرگ‌تر باشند، کامپیوتر سریع تر است. برای مثال، کامپیوتری با حاصل ۴۰، می‌تواند دو برابر سرعت کامپیوتری با حاصل ۲۰، عمل پردازش را انجام دهد. رتبه بندی Passmark یک میانگین از تمام نتایج آزمایش است و یک شاخص کلی از عملکرد کامپیوتر به ما می‌دهد. هرچقدر مقدار آن بزرگ‌تر باشد، سرعت کامپیوتر بیشتر است. نتایج به دست آمده را می‌توانید در جدول (۱۴-۲) مشاهده کنید.

ارزیابی نتایج

آزمون ارزیابی عملکرد به ما نشان می‌دهد که سیستم کامپیوتری که برای سرورهای DB۲ تنظیم شده است، نسبت به سیستم‌های دیگر در بیشتر آزمایش‌ها عملکرد بهتری دارند. بااین‌حال، رتبه بندی پسمارک (میانگین تمام آزمایشات شاخص کارایی بهتری به ما می‌دهد) سیستم کامپیوتری که برای SQL Server ۲۰۰۰ تنظیم شده است، بالاترین مقدار را دارد.

جدول (۱۴-۲) نتایج عملکرد معماری پلتفرم

پارامتر تست شده	سیستم اوراکل	سیستم Informix	سیستم SQL	سیستم DB۲
ریاضی - جمع	۹۶,۶	۹۶,۲	۹۴,۶	**۹۷,۰**
ریاضی - تفریق	۹۶,۴	۹۷,۱	۹۶,۲	**۹۷,٦**
ریاضی - ضرب	۱۰۱,۱	۱۰۱,۴	۱۰۱,۴	**۱۰۳,۱**
ریاضی - تقسیم	۱۲,۹	۱۲,۸	۱۲,۹	**۱۳,۰**
ریاضی – جمع اعشاری	۸۷,۷	۸۷,۸	۸۷,۶	**۸۸,۷**
ریاضی – تفریق اعشاری	۸۹,۴	۸۹,۵	۸۸,۶	**۹۰,۱**
ریاضی – ضرب اعشاری	۹۱,۷	۹۱,۷	۹۰,۹	**۹۲,۳**
ریاضی – تقسیم اعشاری	۱۴,۸	۱۴,۸	۱۴,۸	**۱٤,۹**
ریاضی – Mega Flops (۱,۰۰۰,۰۰۰ عملیات اعداد اعشاری (ممیز شناور) در هر ثانیه) ماکزیمم	۱۷۱,۲	۱۷۲,۲	۱۷۰,۷	**۱۷۷,٦**
گرافیک ۲ بعدی ، خطوط	۱۷,۵	۱۷,۶	۱۷,۵	**۱۷,۸**
گرافیک ۲ بعدی، تصاویر بیت مپ	۱۲,۹	۱۲,۹	۱۲,۸	**۱۲,۹**
گرافیک ۲ بعدی ، اشکال	۴,۷	۴,۷	۴,۷	**٤,۷**
گرافیک ۳ بعدی – دنیاهای چندگانه	۲۲,۹	۲۳,۰	۲۲,۹	**۲۲,۹**
حافظه، بلوک های کوچک تخصیص داده‌شده	۸۶,۶	۸۷,۶	۸۷,۰	**۸۷,٦**
حافظه – خواندن کش	۶۷,۹	۶۸,۴	۶۸,۰	**٦۸,۵**
حافظه – خواندن غیر کش	۴۸,۷	۴۸,۸	۵۰,۰	**٤۹,۱**
حافظه نوشتن	۴۰,۸	۴۱,۱	۴۰,۹	**٤۱,٤**
دیسک – خواندن ترتیبی	۳,۲	۳,۸	۳,۷	**۳,۱**
دیسک – نوشتن ترتیبی	۲,۹	۳,۴	۳,۴	**۲,۹**

دیسک – جستجوی تصادفی	۱,۲	۲,۳	۳,۶	**۲,۱**
ریزپردازنده‌های پنتیوم MMX، جمع	۹۷,۷	۹۴,۵	۹۷,۸	**۹۹,٤**
ریزپردازنده‌های پنتیوم MMX، تفریق	۹۲,۳	۹۸,۲	۹۳,۳	**۹٦,۰**
ریزپردازنده‌های پنتیوم MMX، ضرب	۹۷,۸	۹۷,۵	۹۶,۹	**۹۹,۱**
علائم ریاضی	۷۵,۶	۷۵,۸	۷۵,۲	**۷٦,۸**
علائم دو بعدی	۴۶,۷	۴۶,۹	۴۶,۷	**٤۷,۱**
علائم حافظه	۵۸,۷	۵۹,۲	۵۹,۲	**۵۹,٤**
علائم دیسک	۱۹,۳	۲۵,۱	۲۸,۴	**۲۱,۵**
علائم گرافیکی سه بعدی	۱۵,۵	۱۵,۷	۱۵,۵	**۱۵,٦**
علائم MMX	۴۸,۸	۴۹,۲	۴۸,۹	**۵۰,۰**
رتبه Passmark	۴۵,۷	۴۷,۲	۴۷,۸	**٤٦,۷**

۱٤-۲-۲ آزمون Burn-in

تست سخت‌افزاری سیستم کامپیوتری Burn-in برای آزمایش سخت‌افزار یک کامپیوتر شخصی در دوره زمانی کوتاهی است، همانند برنامه‌هایی که از یک PC در دوره طولانی مدت استفاده می‌کند. این آزمون به موارد زیر می‌پردازد:

- CPU
- درایوهای سخت
- CD- ROMs
- کارت‌های صدا

- گرافیک دوبعدی
- گرافیک سه بعدی
- RAM
- اتصالات شبکه و چاپگرها

ارزیابی آزمون Burn-in

با توجه به نتایج جدول (۱۴-۳)، متوجه می‌شویم که سرعت CPU ۲۰۰۰ SQL Server نسبت به دستگاه‌های دیگر ۰,۱ مگاهرتز کمتر است. این تفاوت تا زمانی که دوره ارزیابی عملکرد آزمون‌ها تا مدت طولانی به طول نینجامد، قابل توجه نیست. تمام معیارهای دیگری که ۴ دستگاه پلتفرم را نشان می‌دادند، یکسان بودند.

جدول (۱۴-۳): نتایج تست Burn-in

اطلاعات سیستم	Informix	Oracle	DB۲	SQL Server
سیستم‌عامل	Win NT۴	Win NT۴	Win NT۴	**Win NT٤**
تعداد CPU ها	۱	۱	۱	**۱**
سازنده CPU	Intel	Intel	Intel	**Intel**
نوع CPU	Celeron	Celeron	Celeron	**Celeron**
ویژگی های CPU	MMX	MMX	MMX	**MMX**
شماره سریال CPU	موجود نیست یا غیر فعال شده	موجود نیست یا غیر فعال شده	موجود نیست یا غیر فعال شده	موجود نیست یا غیر فعال شده

سرعت CPU۱	۵۰۱,۳ MHz	۵۰۱,۳ MHz	۵۰۱,۳ MHz	**۵۰۱,۲ MHz**
کش سطح ۲ CPU	۱۲۸ KB	۱۲۸ KB	۱۲۸ KB	**۱۲۸ KB**
RAM	۲۶۷،۸۲۱،۰۵۶ Bytes (۲۵۶ MB)	۲۶۷،۸۲۱،۰۵۶ Bytes (۲۵۶ MB)	۲۶۷،۸۲۱،۰۵۶ Bytes (۲۵۶ MB)	**۲٦۷،۸۲۱،۰٥٦ Bytes (۲٥٦ MB)**
عمق رنگ	۲۴	۲۴	۲۴	**۲٤**

۱۴-۳ سیستم‌های پایگاه داده

۱۴-۳-۱ پایگاه داده ۱، ساختار معماری اوراکل

اجزای متشکل سیستم پایگاه داده اوراکل با استفاده از ساختمان‌های حافظه مجازی و پردازش‌های برنامه کاربردی اصلی اجرا می‌شوند. پردازش‌ها عبارتند از کارها یا وظایفی که در حافظه این کامپیوترها کار می‌کنند. اوراکل همیشه تأکید زیادی روی قابلیت حمل و نقل آن‌ها داشته است و به آن‌ها ویژگی‌های یکنواخت و امکاناتی در بیشترین حد ممکن از محیط‌های عملیاتی داده است. اوراکل یک معماری مشترک را پیاده سازی می‌کند که شامل مؤلفه‌های زیر است:

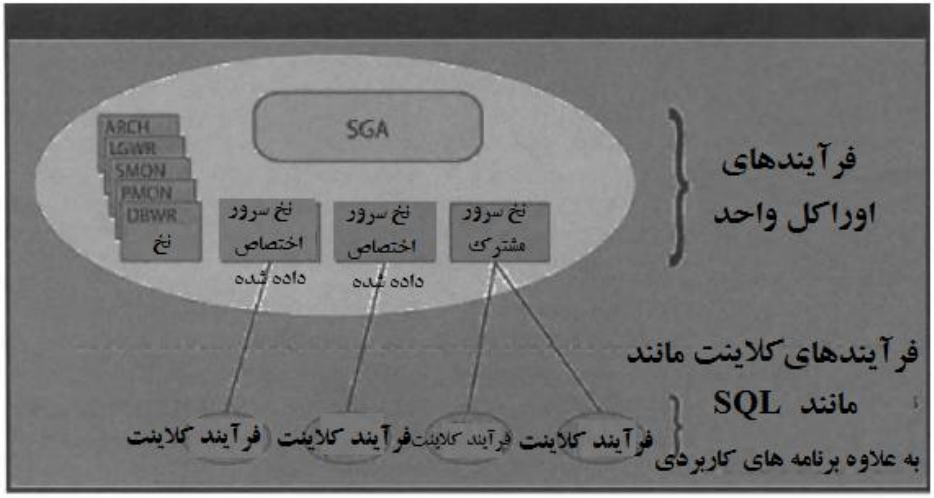
- یک منطقه از حافظه که برای تمام نشست‌های اوراکل موجود است و به آن منطقه جهانی سیستم (SGA) می‌گویند. این منطقه از حافظه شامل بلوک‌های داده ای است که اخیرا مورد

دسترسی قرار گرفته است (کش بافر)، SQL و اشیاء PL/SQL (کش کتابخانه) و اطلاعات مربوط به تراکنش. SGA همیشه ممکن است حاوی اطلاعات نشست باشد.

- وظایف متعددی که فعالیت‌های مربوط به پایگاه داده اختصاصی را اجرا می‌کنند شامل نویسنده پایگاه داده (DBWR)، نویسنده redo log (LGWR)، مانیتور سیستم (SMON)، مانیتور پردازش (PMON) و بایگانی کننده log (ARCH). وظایف دیگر اگر نیاز به حمایت از گزینه‌های اوراکل باشد قابل تنظیم هستند مثل، صف موازی، پایگاه داده توزیع‌شده یا سرورهای چندنخی. ما به این کارها به‌عنوان کارهای پس زمینه نگاه می‌کنیم (اگرچه آن‌ها اغلب به‌عنوان فرآیندهای پس زمینه ای دیده می‌شوند).
- فایل‌های داده اوراکل که حاوی جدول ها، شاخص‌ها و بخش‌های دیگری هستند که اوراکل را تشکیل می‌دهند.
- Redo logs، که اطلاعات تراکنش مهم را برای جلو رفتن در صورت بروز خطا ثبت می‌کند.
- یک وظیفه جداگانه ایجاد شده تا عملیات پایگاه داده را از جانب هر نشست اوراکل انجام دهد و به آن سرور اختصاصی می‌گویند. اگر گزینه سرور چند موضوعی اجرا شده باشد، از نشست‌های زیادی می‌توان توسط تعداد کمی از سرورهای مشترک حمایت کرد.
- اگر گزینه سرور چندموضوعی پیاده سازی شده باشد، توسط تعداد کمی سرورهای مشترک می‌توان نشست‌های زیادی را پشتیبانی کرد.

- یک وظیفه SQL*Net که ارتباط میان سیستم‌های خارجی را برقرار می‌کند.

فایل‌های پایگاه داده و redo log معمولاً با استفاده از سیستم فایل بومی سیستم‌عامل یا پارتیشن‌های دیسک خام پیاده سازی می‌شوند و تفاوت‌های مخصوص پورت در سطح فایل بسیار جزئی هستند. بااین‌حال، ساختمان حافظه و پردازش یک مورد اوراکل با توجه به نحوه پردازش سیستم‌عامل و مدیریت حافظه تفاوت قابل توجهی دارد. معماری اوراکل در محیط ویندوز NT تا حدودی با یونیکس متفاوت است. اوراکل دارای مزیت حمایت قوی NT برای نخها است. تقریبا در همه سیستم‌عامل ها، به یک پردازش اجازه دسترسی به حافظه متعلق به پردازش دیگر داده نمی‌شود. بااین‌حال، نخ های متعلق به یک پردازش یک فضای آدرس حافظه مشترک دارند و می‌توانند به‌سادگی حافظه را به اشتراک بگذارند. در NT، نمونه اوراکل به‌عنوان یک پردازش NT واحد پیاده سازی شده است (شکل ۱۴-۱). این پردازش شامل نخ هایی است که هر یک از کارهای موردنیاز برای نمونه اوراکل را پیاده سازی می‌کند. بنابراین، برای هر پس زمینه وظایف سرور یک نخی و به علاوه یک سربار دو نخی برای هر اتصال سرویس گیرنده وجود دارد. از آنجایی که هر نخ یک فضای حافظه مشترک دارد، نیازی به پیاده سازی SGA در حافظه مشترک نمی‌باشد، اگر شما در حافظه پردازش نمونه، SGA را پیاده سازی کنید، برای تمام نخ های موجود در پردازش در دسترس قرار می‌گیرد. معماری اوراکل، مدل پردازش/ نخ NT را متناسب می‌کند. بااین‌حال، مدل تک پردازشی باعث محدودیت کل حافظه موجود برای نخهای متعلق به نمونه اوراکل در NT می‌شود.



شکل (۱۴-۱): ساختار نخ و پردازش اوراکل در NT

پیش از نسخه ۳,۵۱ از NT، محدودیت حافظه برای یک پردازش تنها ۲۵۶ مگابایت بود – یعنی محدودیت شدید حتی برای نمونه‌های متوسطی از اوراکل. در نسخه ۴,۰ آن یک پردازش می‌توانست بیش از ۴ گیگابایت از حافظه مجازی را آدرس دهی کند. بااین‌حال، ۲ گیگابایت از این حافظه برای سربار سیستم حفظ شد و تنها ۲ گیگابایت برای اوراکل مجاز بود. در نگاه اول، ۲ گیگابایت ممکن است تخصیص حافظه خوبی برای نمونه اوراکل به نظر آید. اما به یاد داشته باشید که این منطقه از حافظه باید برای ذخیره قطعات داده و SGA برای تمام نشست‌های اوراکل کافی باشد. علاوه بر این، ۲ گیگابایت یک محدودیت حافظه مجازی است، این احتمال وجود دارد که اگر حافظه فیزیکی خیلی کمتر از حد نیاز باشد، ۲ گیگابایت از حافظه مجازی به آن افزوده شود. در حال حاضر دو گزینه برای گسترش محدوده ۲ گیگابایتی وجود دارد: در نسخه تخصصی سرور ویندوز NT، شما می‌توانید مولفه سیستم

حافظه پردازش را تا ۱ گیگابایت کاهش دهد و بیش از ۳ گیگابایت از حافظه را برای نمونه اوراکل نگه دارید. در پلتفرم آلفا NT، گزینه حافظه خیلی زیاد (VLM) بیش از ۸ گیگابایت حافظه در اختیار نمونه اوراکل قرار می‌دهد. گزینه سرور چند موضوعی اوراکل این امکان را می‌دهد تا چند پردازش یک مقدار کمی از پردازش‌های سرور اوراکل را به اشتراک بگذارند. این روش می‌تواند نیاز به حافظه و سربار پردازش را کاهش دهد. سرور چند موضوعی در NT نیز وجود دارد اما تنها از اوراکل ۸ به بعد. استفاده از سرور چند موضوعی تحت ویندوز NT می‌تواند تعداد بندهای موجود در پردازش اوراکل و الزامات کلی حافظه را کاهش دهد. این می‌توان از مخزن اتصال اوراکل ۸ استفاده کرد و امکاناتی برای کاهش بیشتر سربار نخ و حافظه ایجاد نمود.

تراکنش‌ها

اوراکل از انواع تراکنش پشتیبانی می‌کند شامل تراکنش‌های فقط خواندنی، خواندن/نوشتن و گسسته. با توجه به تنظیم نوع تراکنش برای تراکنش، پایگاه داده اوراکل یکپارچگی داده را تضمین می‌کند. اگر هیچ نوع تراکنشی برای یک تراکنش تنظیم نشده باشد، به حالت پیش فرض خواندن/نوشتن تنظیم می‌شود. برای هر تراکنش، اوراکل باید تراکنش و تاثیری که تراکنش روی پایگاه داده دارد را ردیابی نماید. این کار برای این انجام می‌شود که اگر تراکنش تمام نشده باشد، می‌تواند بازگردد و تأثیرات تراکنش از پایگاه داده "انجام نشده" باقی می‌ماند. این کار انسجام پایگاه داده را تضمین خواهد کرد. اوراکل از یک نوع خاصی از قطعه برای ثبت مشخصات تراکنش استفاده می‌کند.

توجه: پرسشهایی که برای آزمایش TPC-H اجرا می‌شوند، هیچ نوع تنظیم تراکنشی ندارند، بنابراین آن‌ها به‌صورت تراکنش‌های خواندنی/نوشتنی اجرا شدند.

بهینه سازی پرسش

اوراکل یک ویژگی سیستمی داخلی به نام بهینه ساز دارد. بهینه ساز یک یا چند برنامه اجرایی بیشتر دارد که از آن می‌توان برای اجرای عبارت SQL استفاده کرد. اوراکل 8i سه گزینه دارد: هزینه، قانون و انتخاب. بهینه ساز مبتنی بر هزینه با استفاده از برنامه ای با کمترین هزینه، عبارت SQL را اجرا خواهد کرد. بهینه ساز مبتنی بر قانون با توجه به قوانین تعریف‌شده توسط کاربر که در پایگاه داده راه اندازی شده است، عبارت SQL را اجرا خواهد کرد. بهینه سازی مبتنی بر انتخاب نیز کم هزینه ترین بهینه سازی (هزینه یا قانون) انتخاب می‌کند که می‌تواند عبارت SQL را اجرا کند. به‌منظور تعیین بهترین برنامه‌های اجرایی برای عبارات SQL، اوراکل از آمار موجود در پایگاه داده استفاده می‌کند. این آمار باید به‌صورت دوره ای در ضمایم و جدول‌های پایگاه داده و اشیاء پایگاه داده دیگر به روز شوند. اگر پایگاه داده بعد از تولید آمار اصلاح شود (بعد از تجزیه و تحلیل جدول ها)، بهینه ساز ممکن نیست با کمترین هزینه اجرا شود، بنابراین، آمار باید در جدول‌های پایگاه داده به‌صورت منظم ایجاد شود. یکی از برنامه‌های اجرایی پرهزینه، اسکن جدول کامل است. در اسکن‌های جدول کامل، اوراکل باید هر ردیف جدول را بخواند. برنامه اجرایی دیگر برای پیدا کردن ردیف‌های یک جدول، جستجوی شاخصی از جدول است. بهینه سازها می‌توانند از امتیازات عبور کنند و بهترین مسیر اجرایی برای یک عبارت SQL را انتخاب نمایند. یکی از این امتیازها می‌تواند مربوط به استفاده از شاخص‌ها باشد. امتیازات

دیگر بهینه ساز شامل اولین ردیف، تمام ردیف ها، اسکن کامل جدول، حلقه تودرتو، اتصال یکپارچه، استفاده از اتصال هش و غیره می‌باشد. این امتیازات می توانند به عبارات SQL اضافه شوند تا این اطمینان حاصل شود که بهینه ساز با استفاده از برنامه اجرایی خاص عبارت SQL را اجرا می‌نماید.

نکته: پرسشهای TPC-H با تنظیم انتخاب بهینه ساز پایگاه داده اوراکل اجرا شدند. هیچ شاخصی اجرا نمی‌شود، تمام جدول‌ها پیش از اجرای پرسشها مورد تجزیه و تحلیل قرار می گیرند. هیچ امتیازی به پرسشهای TPC-H افزوده نمی‌شود، برای اینکه انجام این کار دستورالعمل‌های مقایسه را نقض می کند.

کنترل همزمانی و قفل گذاری

اوراکل از مکانیزم‌های قفل گذاری برای حفاظت از داده از آسیب دربرابر تراکنش‌های همزمان استفاده می کند. اوراکل قابلیت قفل خودکار و ضمنی را ارائه می‌دهد. به‌صورت پیش فرض، برای تراکنش‌های داخل پایگاه داده، امکان قفل گذاری برای منابع پایگاه داده را ایجاد کرده است. این سیستم به‌صورت خودکار روی جدول‌ها و ردیف‌ها قفل ایجاد می کند.، سطح قفل‌ها بستگی به قابلیت تراکنش (خواندن، درج، به روزرسانی و حذف) دارد. اوراکل می‌تواند در دو حالت قفل را تنظیم کند: اشتراکی و انحصاری. قفل‌های اشتراکی روی منابع پایگاه داده تنظیم می‌شوند به طوری که بسیاری از تراکنش‌ها بتوانند به منبع دسترسی داشته باشند. قفل‌های انحصاری روی منابع سرور پایگاه داده تنظیم می گردند و این ضمانت را می‌دهند که یک تراکنش دسترسی انحصاری به منبع پایگاه داده را دارد. قفل‌های انحصاری ترتیب تراکنش را تضمین می کنند. قفل‌های DML، قفل‌های اوراکل هستند که به‌صورت

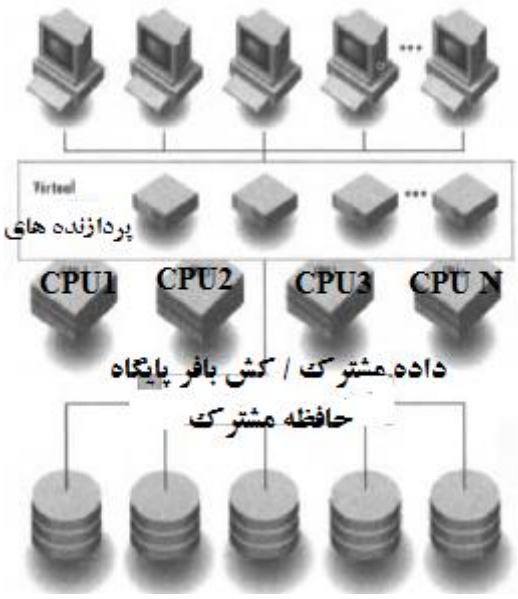
خودکار با استفاده از عملیات DML (بروز رسانی، درج، حذف)، روی جدول‌ها و شاخص‌ها برای تراکنش‌ها تنظیم می‌شوند. علاوه بر این، هنگامی که از عملیات DDL (ایجاد، تغییر و افت) استفاده می‌شود، اوراکل به طور خودکار قفل‌های DDL را روی منابع اوراکل تنظیم می‌کند.

۱۴-۳-۲ پایگاه داده ۲ – ساختار معماری سرور پویای Informix

سرور پویای Informix یک سرور پایگاه داده مربوط به شی است که داده‌های ذخیره شده در ردیف‌ها و ستون‌های یک جدول را مدیریت می‌کند. این سرور از یک پردازنده یا سیستم‌های چندپردازشی متقارن (SMP) و معماری قابل گسترش پویا برای ارائه قابلیت گسترش، مدیریت و عملکرد پایگاه داده استفاده می‌کند. از سرور پویا می‌توان برای پردازش تراکنش آنلاین (OLTP)، برنامه‌های کاربردی بسته بندی، برنامه‌های کاربردی انبار داده و روش‌های وب استفاده نمود.

معماری قابل گسترش پویا

اساس عملکرد برتر، مقیاس پذیری و قابلیت اطمینان سرور پویای Informix، معماری پایگاه داده موازی، معماری قابل گسترش پویای (DSA) آن است که برای بهره برداری کامل از توان پردازش ذاتی هر سخت‌افزار ایجاد شده است (شکل ۱۴-۲).



شکل (۱۴-۲): سرور پایگاه داده مخزن قابل تنظیم

DSA به تمام عملیات پایگاه داده مثل I/O، پرسشهای پیچیده، ساختمان‌های شاخص، بازیابی آمار و پشتیبانی‌ها و مخازن این امکان را می‌دهد تا در تمام منابع سیستم موجود به‌صورت موازی اجرا شوند. معماری هسته سرور پویای Informix به صورتی طراحی شده که دارای قابلیت پردازش موازی و چند موضوعی باشد. پردازش موازی از طریق تقسیم کارهای کاربردی بزرگ به کارهای فرعی به دست می‌آید، بنابراین پردازش می‌توان در تمام منابع موجود توزیع شود. مزایای مهم این سرور پویا به شرح زیر است:

- حداکثر عملکرد و قابلیت گسترش به دلیل داشتن معماری پردازش موازی چند بندی برتر.
- کاهش سربار سیستم‌عامل به دلیل رفع محدودیت های سیستم‌عامل

- پارتیشن بندی جدول محلی برای عملیات I/O موازی برتر و دسترسی بالای مدیریت پایگاه داده.
- قابلیت SQL موازی باعث افزایش عملکرد می‌شود و به تمام عملیات پایگاه داده اجازه می‌دهد تا به‌صورت موازی اجرا شوند، بنابراین، تنگناهای بالقوه از بین می‌روند.
- دسترسی بالای پایگاه داده برای پشتیبانی از طیف انبوهی از برنامه‌های کاربردی ضروری برای کسب‌وکار روی پلتفرم‌های سیستم‌های باز.
- مدیریت سیستم آنلاین توزیع‌شده و پویا برای نظارت کارها در تراکم بار توزیع‌شده.
- برابری کامل روی سیستم‌عامل‌های ویندوز NT و یونیکس.
- قابلیت بالای RDBMS در تمام معماری‌های سخت‌افزاری (تک پردازنده، چندپردازشی متقارن و سیستم‌های خوشه) و مدل‌های پایگاه داده (رابطه ای و شی رابطه ای) باعث مهاجرت کامل برنامه‌های کاربردی، داده و مهارت‌ها می‌شود.
-

قفل گذاری – انسجام داده، جداسازی و بازیابی

در حالی که دردسترس بودن زیاد به ما از انسجام در سطح سیستم خبر می‌دهد، یکنواختی داده به ما یکنواختی در سطح تراکنش را ضمانت می‌کند. سرور پویای Informix، از طریق ورود به تراکنش و بررسی یکنواختی داخلی و با ایجاد و انجام روال‌های قفل گذاری، سطوح تفکیک کنند و قوانین کسب‌وکار، یکنواختی داده را تضمین می‌کند.

هنگامی که یک عملیات قادر به تکمیل نباشد، تراکنشی که ناقص است باید از پایگاه داده حذف شود و یکنواختی داده را از بین نبرد. برای حذف عملیات ناکامل، سرور پویای Informix یک سابقه تاریخی از تمام تراکنش‌های موجود در لاگ‌های منطقی نگه می‌دارد و به‌صورت خودکار از این سوابق به‌عنوان مرجعی برای ذخیره سازی پایگاه داده در وضعیت پیش از تراکنش استفاده می‌نماید. بررسی ثبات داخلی برای تغییر مدیر سرور پویای Informix به عدم ثبات سیستم و داده در نظر گرفته شده است. سرور پویای Informix شامل یک لایه سطح داده از بررسی‌ها است که می‌تواند بی ثباتی‌های داده ناشی از خطاهای سخت‌افزاری و سیستم‌عاملی را پیدا کند. اگر بی ثباتی‌ها پیدا شوند، این مکانیزم داخلی به طور خودکار پیام‌ها را در لاگ پیام سرور پویای Informix می‌نویسد. از ویژگی‌های مهم دیگر برای حفظ ثبات داده می‌توان به روال‌های قفل گذاری و تفکیک پردازش اشاره کرد. این معیارهای امنیتی به کاربران دیگر اجازه نمی‌دهند که داده‌های قابل خواندن را تغییر دهند یا اصلاح کنند و این به سیستم کمک می‌کنند تا متوجه شود چه زمانی قفل‌های متضاد نگه داشته می‌شوند. قفل گذاری سطح صفحه و ردیف در زمان ایجاد یا تغییر جدول مشخص می‌شود. قفل گذاری سطح پایگاه داده و جدول در برنامه کاربردی کاربر مشخص می‌شود. سطح تفکیک درجه ای است که طبق آن عملیات خواندن شما از اقدامات همزمان دیگر پردازش‌های سرور پایگاه داده جدا می‌شوند: تغییراتی که فرآیندهای دیگر می‌توانند روی سوابقی که شما در حال خواندن هستید ایجاد کنند و آن سوابقی را که می‌توانید در حین اینکه پردازش‌های دیگر مشغول خواندن یا اصلاح آن‌ها هستند، بخوانید. سرور پویای

Informix دارای چهار سطح مجزا است: خواندن کثیف، خواندن با تعهد، ثبات مکان نما و خواندن قابل تکرار.

روش‌های اتصال

هنگامی که Informix باید به جدول‌ها بپیوندد، می‌تواند هر یک از این الگوریتم‌ها را انتخاب کند. تمام اتصالات حداقل، اتصالات دو جدولی هستند، چند اتصال با اتصال مجموعه‌های اولیه به جدول‌های بعدی انجام می‌شود. بهینه ساز انتخاب می‌کند که از کدام روش اتصال بر اساس هزینه ها، استفاده نماید، بجز زمانی که با تنظیم OPTCOMPIND در این تصمیم زیاده روی می‌کنید. اتصالات به‌صورت زیر توصیف می‌شوند:

اتصال حلقه تودرتو: هنگامی که ستون‌های هر دو جدول شاخص گذاری شوند، از این روش می‌توان استفاده نمود. جدول اول به هر ترتیب اسکن می‌شود. ستون‌های متصل از طریق شاخص‌ها تطابق داده می‌شوند تا ردیف حاصل را تشکیل دهند. یک ردیف از جدول دوم نیز از طریق شاخص، مورد جستجو قرار می‌گیرد. گاهی اوقات، Informix یک شاخص پویا در جدول دوم تشکیل می‌دهد تا این اتصال برقرار شود. این اتصالات اغلب برای برنامه‌های کاربردی OLTP بسیار کارآمد هستند.

- اتصال ادغام مرتب: بعد از اعمال فیلترها، موتور پایگاه داده هر دو جدول را برای اتصال فیلتر اسکن می‌کند. هر دو جدول ممکن است ابتدا نیاز به مرتب سازی داشته باشند. اگر یک شاخص روی ستون اتصال وجود داشته باشد، دیگر نیازی به مرتب سازی نیست. این روش

معمولاً زمانی انتخاب می‌شود که هر یک از ستون‌ها دارای شاخص نباشند. بعد از مرتب سازی جدول ها، عمل اتصال روش ساده ای برای ادغام مقادیر مرتب است.

- اتصال هش: از نسخه ۷ در دسترس بود، اتصال ادغام هش ابتدا یک جدول را اسکن می‌کند و مقادیر کلیدی هش آن را در یک جدول هش قرار می‌دهد. جدول دوم نیز ابتدا اسکن می‌شود و مقادیر اتصال آن در جدول هش پیدا می‌شود. اتصالات هش اغلب سریع تر از اتصالات ادغام مرتب است برای اینکه نیازی به مرتب سازی ندارند. حتی اگر ایجاد جدول هش به سربار نیاز داشته باشد، با وجود برنامه‌های کاربردی DSS که در آن جدول‌های بزرگی وجود دارد، معمولاً از این روش استفاده می‌شود.

بهینه ساز پرسش مبتنی بر هزینه

بهینه ساز مبتنی بر هزینه سرور پویای Informix به‌صورت خودکار سریع ترین راه را برای بازیابی داده از جدول پایگاه داده بر اساس اطلاعات مربوط به توزیع این داده‌ها در ستون‌های جدول تعیین می‌کند. این بهینه ساز آمار مربوط به این توزیع داده را جمع آوری و محاسبه می‌کند و یک مسیر بازگشت با کمترین اثر روی منابع سیستم را انتخاب می‌نماید – در برخی موارد این مسیر یک مسیر بازگشت موازی است اما در موارد دیگر می‌تواند یک پردازش ترتیبی (سری) باشد. تمام آنچه که برای کنترل درجه موازی سازی نیاز است، یک مدیریت حافظه می‌باشد. برای اینکه کاربران درجه ای از

کنترل داشته باشند، سرور پویای Informix به بهینه ساز جهت‌هایی می‌دهد که برای کاربران امکان دور زدن بهینه ساز را فراهم می‌نماید. بخش‌هایی که کاربران قادر به کنترل هستند شامل موارد زیر است:

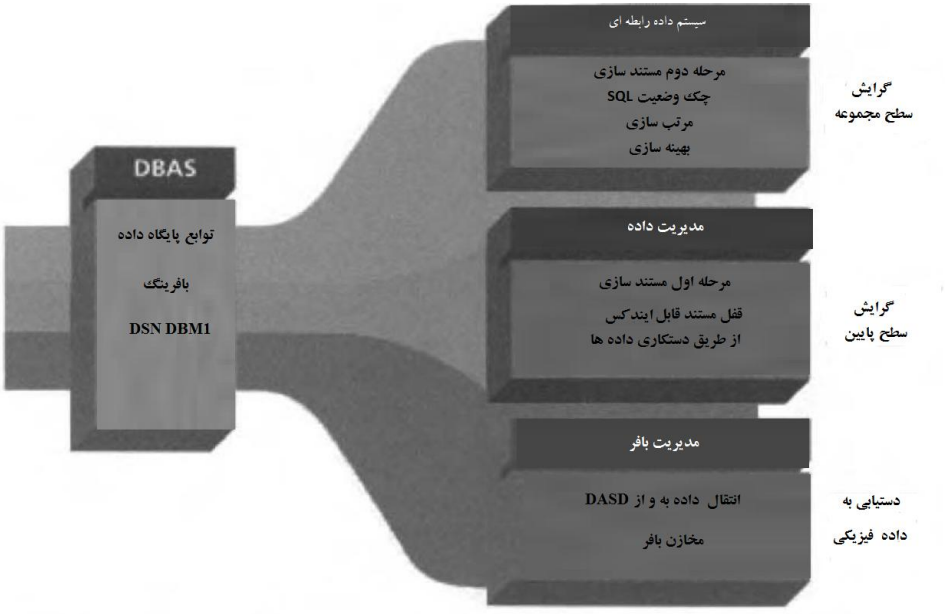
- روش‌های دسترسی: این به کاربران اجازه می‌دهد تا نحوه دسترسی به یک جدول را مشخص کنند. برای مثال، یک کاربر می‌تواند به بهینه ساز جهت دهد تا از یک شاخص مشخص استفاده نماید.
- روش‌های اتصال: این روش به کاربران اجازه می‌دهد تا یک جدول را به جدول‌های دیگر یک پرسش متصل کنند. برای مثال، کاربران می‌توانند مشخص کنند که بهینه ساز از یک اتصال هش استفاده می‌نماید.
- ترتیب اتصال: این به کاربران اجازه می‌دهد تا بهینه ساز را برای اتصال جدول‌ها به شیوه ای مشخص هدایت کنند.
- هدف بهینه سازی: این به کاربران اجازه می‌دهد تا مشخص کنند که یک پرسش توسط زمان پاسخ (که اولین مجموعه ردیف‌ها را باز می‌گرداند) باید بهینه سازی شود یا توسط زمان کل (که تمام ردیف‌ها را باز می‌گرداند).

کنترل حافظه توسط Informix

تمام حافظه مورداستفاده توسط سرور پویای Informix در میان مخزن پردازنده‌های مجازی به اشتراک گذاشته می‌شود. به این صورت، سرور پویای Informix را می‌توان طوری تنظیم نمود تا حافظه بیشتری به مخزن حافظه مشترک آن افزوده شود تا در اسرع وقت به درخواست مشتریان پاسخ

داده شود. داده‌های مربوط به دایرکتوری داده (کاتالوگ سیستم) و روال‌های ذخیره سازی به جای کپی، در میان کاربران به اشتراک گذاشته می‌شوند، که این امر باعث بهبود کارایی حافظه و اجرای سریع روال‌های مورداستفاده می‌شود. این ویژگی می‌تواند مزایای زیادی برای بسیاری از برنامه‌های کاربردی داشته باشد، به ویژه برای برنامه‌هایی که به جدول‌های بزرگ و یا روال‌های ذخیره سازی شده دسترسی دارند. سرور پویای Informix نیز یک منطقه تخصیص می‌دهد که نام آن پشته بند است و در بخش مجازی حافظه اشتراکی قرار دارد تا داده‌های غیراشتراکی برای توابعی که یک بند اجرا می‌کند، ذخیره شوند. پشته بند وضعیت نشست کاربر را بررسی می‌کند و یک پردازنده مجازی برای حفاظت از داده‌های غیراشتراکی یک بند در مقابل بازنویسی شدن توسط بندهایی که به‌صورت همزمان همان کد را اجرا می‌کنند، فعال می‌نماید. سرور پویای Informix به‌صورت پویا پشته را برای عملیات خاصی مثل روال‌های ذخیره سازی شده بازگشتی رشد می‌دهد. حافظه اشتراکی سرور پویای Informix عمل تکه تکه شدن را به حداقل می‌رساند به نحوی که استفاده از حافظه به مرور زمان به زوال نمی‌رود. بجز تخصیص اولیه، قطعات حافظه مشترک به‌صورت خودکار در قطعات بزرگی افزوده می‌شوند اما علاوه بر این می‌توانند در زمان اجرای پایگاه داده توسط مدیر نیز اضافه شوند. سیستم مدیریت حافظه نیز سعی دارد که در زمان اتمام حافظه، قطعه حافظه را به‌صورت خودکار رشد دهد. هنگامی که نشست کاربر به پایان می‌رسد، حافظه مورداستفاده آزاد می‌شود و توسط نشست بعدی مورداستفاده قرار می‌گیرد. حافظه را می‌توان با آزاد سازی حافظه تخصیص یافته به پایگاه داده توسط سیستم‌عامل اصلاح نمود. بنابراین،

بندهای کاربر به‌سادگی می‌توانند با افزایش تعداد کاربر، در میان پردازنده‌های مجازی که در گسترش سرور پویای Informix شرکت دارند، مهاجرت کنند.



شکل (۱۴-۳): مؤلفه‌های فضای آدرس سرویس‌های پایگاه داده.

۱۴-۳-۳ پایگاه داده ۳ – ساختمان معماری IBM DB۲

از لحاظ مفهومی، DB۲ یک سیستم مدیریت پایگاه داده رابطه ای است. ازنظر فیزیکی نیز DB۲ یک ادغام از فضاهای آدرس و لینک‌های ارتباطی درون سیستمی است که وقتی با یکدیگر ترکیب می‌شوند، سرویس‌های یک سیستم مدیریت پایگاه داده رابطه ای را ایجاد می‌کنند. در

نسخه سوم DB۲، هر زیرسیستم DB۲ شامل سه یا چهار کار است که از کنسول ۱ اپراتور آغاز می‌شوند. هر کار در بخشی از CPU به نام فضای آدرس اجرا می‌شود. نسخه چهارم DB۲ یک فضای آدرس اضافه برای روال‌های ذخیره شده فراهم می‌کند. توصیفی از این پنج فضای آدرس را در ادامه خواهید دید (شکل ۱۴-۳).

- DBAS یا فضای آدرس سرویس‌های پایگاه داده امکان کنترل ساختمان داده DB۲ را فراهم می‌کند. نام پیش فرض این فضای آدرس DSNDBM۱ است اما هر فروشگاه ممکن است هر فضای آدرس DB۲ را تغییر نام دهد. DBAS مسئول اجرای بندهای SQL و مدیریت بافرهای داده است و حاوی منطق هسته‌ای سیستم مدیریت پایگاه داده است. سه مؤلفه جداگانه DBAS را تشکیل می‌دهد: سیستم داده رابطه ای، مدیر داده و مدیر بافر. هر یک از این مولفه‌ها کارها بخصوصی را انجام می‌دهند.
- SSAS یا فضای آدرس سرویس‌های سیستم، ضمایم DB۲ را با سیستم‌های جانبی دیگر (CICS، IMS/DC یا TSO) هماهنگ می‌کند. SSAS نیز مسئول اتصال فعالیت‌ها است (اتصال فیزیکی، بایگانی اتصال و BSDS). DSNMSTR یک نام پیش فرض برای این فضای آدرس است.
- سومین فضای آدرس موردنیاز DB۲، IRLM یا مدیر قفل منبع درون سیستم است. IRLM مسئول مدیریت قفل‌های DB۲ (شامل تشخیص بن‌بست) است. نام پیش فرض این فضای آدرس IRLMPROC است. نسخه سوم چهارمین DB۲ فضای آدرس، DDF

یا امکان داده توزیع‌شده، تنها گزینه اختیاری است. DDF تنها زمانی موردنیاز است که به قابلیت پایگاه داده توزیع‌شده نیاز باشد.

- جدیدترین فضای آدرس، SPAS یا فضای آدرس روال ذخیره شده نام دارد که به نسخه چهارم DB۲ اضافه شده است تا از روال‌های ذخیره شده و فراخوان‌های روال راه دور (RPCs) پشتیبانی نماید. SPAS به‌صورت یک فضای آدرس متحد اجرا می‌شود و یک محیط مستقل برای روال‌های ذخیره شده برای اجرا فراهم می‌کند. این کار می‌تواند به‌صورت مؤثر کد روال ذخیره شده نوشته شده توسط کاربر را در دنیای کوچک خود ایزوله نماید، به نحوی که خودش نتواند با کد سیستم DB۲ تداخل داشته باشد.

این ۵ فضای آدرس دهی شامل منطقی است که به‌صورت مؤثر تمام قابلیت‌های DB۲ را اداره می‌کند.

قابلیت‌های DBAS

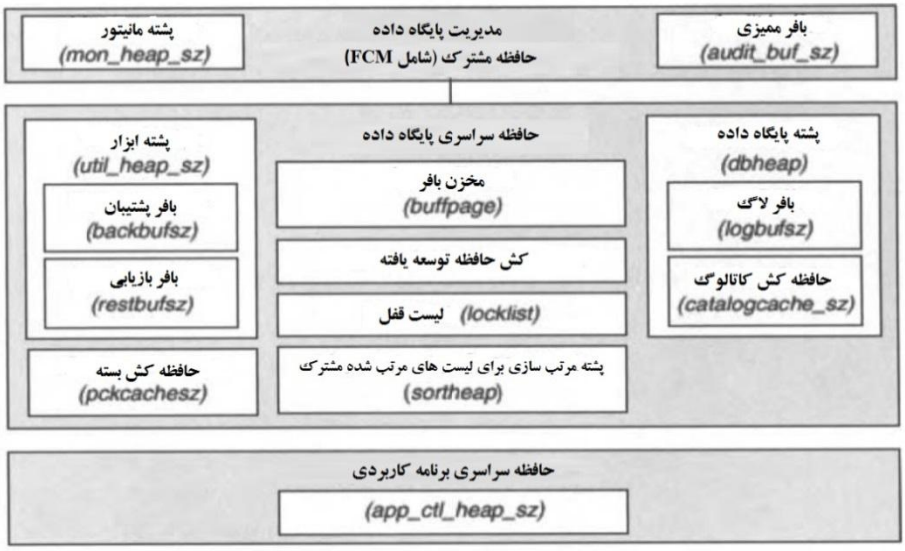
به یاد داشته باشید که DBAS مسئول اجرای SQL است و از سه مؤلفه مجزا تشکیل می‌شود: سیستم رابط‌های، مدیر داده و مدیر بافر. هر مؤلفه یک بند SQL را به مؤلفه بعدی ارجاع می‌دهد و زمانی که نتایج بازگردانده شوند، هر مؤلفه نتایج را بازمی‌گرداند. سیستم داده رابطه‌ای (RDS) مؤلفه‌ای است که به DB۲ گرایشات خود را می‌دهد. هنگامی که یک عبارت SQL مجموعه‌ای از ستون‌ها و سطرها را درخواست می‌کند، این درخواست به RDS می‌رسد و بهترین مکانیزم برای رسیدگی به درخواست تعیین می‌شود. توجه داشته باشید که RDS می‌تواند یک عبارت SQL را منتشر کند و نیازهای آن را نیز تعیین نماید. این نیازها می‌توانند شامل ویژگی‌هایی باشند که توسط یک پایگاه داده رابطه‌ای حمایت

می‌شوند (مثل انتخاب، طرح‌ریزی یا اتصال). هنگامی که یک عبارت SQL توسط RDS دریافت می‌شود، هویت را بررسی می‌نماید، نام عناصر داده که در دسترس شناسه‌های داخلی قرار دارد را ترجمه می‌کند، املای SQL را بررسی می‌نماید و SQL را بهینه‌سازی می‌کند و یک مسیر دسترسی تشکیل می‌دهد. سپس، RDS از عبارت SQL بهینه به مؤلفه مدیر داده (DM) می‌رود. DM در داده مورد دسترسی عمیقا کاوش می‌کند. به عبارت دیگر، DM مؤلفه DB۲ است که ردیف‌های داده (کل ردیف‌ها یا ردیف‌های شاخص جدول) را مورد تجزیه و تحلیل قرار می‌دهد. DM درخواست داده را تحلیل می‌نماید و سپس از مدیر بافر می‌خواهد که به درخواست رسیدگی نماید. مدیر بافر (BM) برای مولفه‌های دیگر DB۲ به داده دسترسی پیدا می‌کند. بافر داده اغلب به حافظه کش در DBMSs گفته می‌شود. BM از مخازن مجموعه حافظه برای ذخیره سازی داده‌هایی که اغلب به آن‌ها مراجعه می‌شود استفاده می‌کند تا یک محیط دسترسی به داده کارآمد ایجاد نماید. هنگامی که یک درخواست به BM برسد، باید تعیین شود که آیا داده موجود در مخزن بافر مناسب است یا خیر. اگر مناسب بود، BM به این داده دسترسی پیدا می‌کند و آن‌ها را به DM می‌فرستد. اگر این داده‌ها در مخزن بافر نباشند، BM از مدیر رسانه VSAM می‌خواهد تا داده را بخواند و به BM باز گرداند که آن هم به نوبه خود آن‌ها را به DM می‌فرستد. DM داده فرستاده شده به آن را توسط BM دریافت می‌کند و تا حد ممکن برای کاهش تنظیمات پاسخ گزاره‌های زیادی اعمال می‌کند. تنها گزاره‌های مرحله ۱ در DM اعمال می‌شوند. درنهایت، RDS داده را از DM دریافت می‌کند. تمام گزاره‌های مرحله دوم اعمال می‌شوند، مرتب سازی لازم انجام می‌گیرد و نتایج به درخواست‌کننده بازگردانده می‌شود. در هنگام

توسعه یک برنامه کاربردی DB۲، بهتر است درک درستی از مؤلفه‌های داخلی DB۲ داشته باشیم. برای مثال، گزاره‌های مرحله ۱ و ۲ را در نظر بگیرید. به‌سادگی می‌توان درک کرد که گزاره مرحله ۱ کارایی بیشتری نسبت به گزاره‌های مرحله ۲ دارد، برای اینکه شما می‌دانید آن‌ها پیش از این در پردازش مورد ارزیابی قرار گرفته‌اند (به‌جای RDS در DM). بنابراین، آن‌ها مانع سربار ناشی از عبور داده‌های اضافه از یک مؤلفه به مؤلفه دیگر می‌شوند.

مدیریت حافظه DB۲

حافظه اشتراکی مدیر پایگاه داده زمانی تخصیص می‌یابد که مدیر پایگاه داده شروع به استفاده از فرمان db۲start می‌کند و تا زمانی که مدیر پایگاه داده دیگر از این فرمان استفاده نمی‌کند، به‌صورت تخصیص‌یافته باقی می‌ماند. از این حافظه برای مدیریت فعالیت در سراسر اتصالات پایگاه داده استفاده می‌شود. از نقطه نظر حافظه اشتراکی مدیر پایگاه داده، تمام حافظه‌های دیگر الحاق و یا تخصیص می‌یابند. حافظه عمومی پایگاه داده (که به آن حافظه اشتراکی پایگاه داده نیز می‌گویند) هنگامی که پایگاه داده با استفاده از فرمان ACTIVATE DATABASE فعال می‌شود یا هنگامی که اولین برنامه کاربردی به پایگاه داده متصل می‌شود، به هر پایگاه داده تخصیص می‌یابد.



شکل (۱۴-۴): مرور کلی حافظه اشتراکی مدیر پایگاه داده

حافظه جهانی پایگاه داده تا زمانی که پایگاه داده با استفاده از فرمان DEACTIVATE DATABASE غیرفعال شود و یا تا زمانی که آخرین برنامه کاربردی از پایگاه داده قطع گردد، تخصیص‌یافته باقی می‌ماند. حافظه سراسری پایگاه داده شامل حوزه حافظه مثل مخازن بافر، فهرست قفل، حافظه آزاد پایگاه داده و حافظه آزاد کاربردی است. پارامتر پیکربندی مدیر پایگاه داده، NUMDB، حداکثر مقدار پایگاه داده‌های فعال همزمان را تعریف می‌کنند. اگر مقدار این پارامتر افزایش یابد، تعداد قطعات حافظه سراسری پایگاه داده نیز ممکن است با توجه به تعداد پایگاه داده فعال افزایش پیدا کند. شکل (۱۴-۴) نشان می‌دهد که چگونه از حافظه برای حمایت از برنامه‌های کاربردی استفاده می‌شود. در بخش قبل ما برخی از پارامترهای پیکربندی را معرفی می‌کنیم که ممکن است روی

تعداد قطعات حافظه تأثیر داشته باشند. در حال حاضر، پارامترهای پیکربندی را معرفی می‌کنیم که با محدود کردن اندازه حافظه، به شما امکان کنترل اندازه هر حافظه را می‌دهند. حافظه اشتراکی مدیر پایگاه داده به مدیر پایگاه داده برای اجرا کمک می‌کند. اندازه این حافظه تحت تأثیر پارامترهای پیکربندی زیر قرار دارد:

- اندازه حافظه آزاد مانیتور سیستم پایگاه داده (MON_HEAP_SZ)
- اندازه بافر حسابرسی (AUDIT_BUF_SZ)
- بافرهای FCM (FCM_NUM_BUFFERS)
- مجریان پیام FCM (FCM_NUM_ANCHORS)
- ورودی‌های اتصال FCM (FCM_NUM_CONNECT)
- بلوک‌های درخواست FCM (FCM_NUM_RQB)

مدیر پایگاه داده هنگامی که موازی سازی فرا پارتیشن فعال می‌شود، از مؤلفه مدیر ارتباطات سریع (FCM) برای انتقال داده میان عامل‌های DB۲ استفاده می‌کند. بنابراین، اگر موازی‌سازی فرا پارتیشن را فعال نکنید، به مناطق حافظه موردنیاز برای بافرهای FCM، مجریان پیام، ورودی‌های اتصال و بلوک‌های درخواست تخصیص نمی‌یابند. حداکثر اندازه قطعه حافظه عمومی پایگاه داده توسط پارامترهای پیکربندی زیر تعیین می‌شوند:

اندازه مخزن بافر وقتی که مخازن بافر ایجاد می‌شوند یا تغییر داده می‌شوند (مقدار پارامتر پیکربندی پایگاه داده BUFFPAGE اگر ۱ شود، گرفته می‌شود) مشخص می‌شوند.

- حداکثر ذخیره سازی برای فهرست قفل (LOCKLIST)
- حافظه آزاد پایگاه داده (DBHEAP)
- اندازه حافظ آزاد کاربردی (UTIL_HEAP_SZ)
- اندازه قطعه حافظه ذخیره سازی گسترش یافته (ESTORE_SEG_SZ)
- تعداد قطعات حافظه ذخیره سازی گسترش یافته (NUM_ESTORE_SEGS)
- اندازه کش بسته (PCKCACHESZ)
- حافظه عمومی برنامه کاربردی توسط پارامترهای پیکربندی زیر تعیین می‌شود: اندازه حافظه آزاد کنترل برنامه کاربردی (APP_CTL_HF_akP_SZ)

بهینه سازی پرسش

بهینه سازی پرسش بخشی از فرآیند پرسش است که در آن سیستم پایگاه داده استراتژی‌های پرسشی مختلف را با یکدیگر مقایسه می‌کند و یک استراتژی با حداقل هزینه مورد انتظار را انتخاب می‌نماید. بهینه ساز پرسش که این قابلیت را دارد، بخش کلیدی پایگاه داده رابطه ای است و موثرترین روش برای دسترسی به داده را تعیین می‌نماید. این قابلیت به کاربران این امکان را می‌دهد تا بدون مشخص کردن نحوه بازیابی این اطلاعات، آن‌ها را درخواست نمایند. هزینه دسترسی به یک پرسش یک ترکیب اندازه گیری شده از I/O و هزینه‌های پردازش است. هزینه I/O، هزینه دسترسی به شاخص و صفحات داده از دیسک می‌باشد. هزینه پردازش با تخصیص یک شمارنده دستور به هر مرحله در محاسبه نتایج پرسش اندازه گیری می‌شود. دو روش برای بهینه سازی وجود دارد:

۱. بهینه سازی مبتنی بر هزینه: این عمل توسط IBM توسعه‌یافته است. بهینه ساز هزینه هر روش پردازش پرسش را برآورد می‌کند و روشی با کمترین هزینه را برمی‌گزیند. در حال حاضر، بیشتر سیستم‌ها از این روش استفاده می‌کنند.

۲. روش بهینه سازی اکتشافی: قوانین بر اساس شکل پرسش هستند. اوراکل از این روش استفاده می‌کرد. در حال حاضر، هیچ سیستمی از این روش استفاده نمی‌نماید.

بهینه ساز پرسش وظیفه انتخاب شاخص‌های مناسب برای کسب داده، طبقه‌بندی گزاره‌های مورداستفاده در یک پرس وجو، اجرای کاهش ساده داده، انتخاب مسیرهای دسترسی، تعیین ترتیب یک اتصال، اجرای تبدیلات گزاره، اجرای تبدیلات منطق بولی و اجرای تبدیلات پرس وجوهای فرعی را بر عهده دارد، همه این‌ها برای کارآمدتر کردن پردازش پرسش انجام می‌شوند.

کنترل همزمان و قفل‌گذاری در DB۲

گرانولیته قفل‌گذاری در یک سیستم مدیریت پایگاه داده یک مبادله مشخص میان همزمانی و سربار CPU را نشان می‌دهد. هر زمان که به گرانولیته ریزتری از قفل گذاری نیاز باشد، ممکن است نیاز به افزایش استفاده از منابع CPU موجود نیاز داشته باشیم، برای اینکه به طور کلی، قفل گذاری طول مسیر CPU را افزایش می‌دهد. هیچ عمل ورودی/خروجی انجام نمی‌شود اما هر درخواست قفل به ارتباط دو طرفه میان DB۲ و مدیر قفل منبع داخلی (IRLM) نیاز دارد. بااین‌حال، این ممکن است یک افزایش در تعداد درخواست‌های قفل بالقوه وجود داشته یا نداشته باشد. برای مثال، برای SQL فقط خواندنی با امتناع خیلی مؤثر قفل، ممکن است در تعداد درخواست‌های قفل DB۲ به IRLM افزایشی

مشاهده نشود. یک DB۲ درخواست‌های قفل را به سرویس‌های IRLM می‌رساند. قفل‌های مبادلات تحت مالکیت واحد کاری یا بند قرار دارند و توسط IRLM مدیریت می‌شوند. اشیاء DB۲ که گزینه‌های قفل گذاری مبادلات هستند عبارتند از:

- فضای جدول
- پارتیشن
- جدول
- صفحه
- ردیف

مکانیزم‌های قفل گذاری باید عملیات زیادی را به نام قفل گذاری انجام دهند – برای مثال، سلسله مراتب قفل، دوره قفل، حالت قفل ها، تشدید و تعلیق قفل و تشخیص و بازیابی بن بست را مدیریت کنند.

روش‌های اتصال

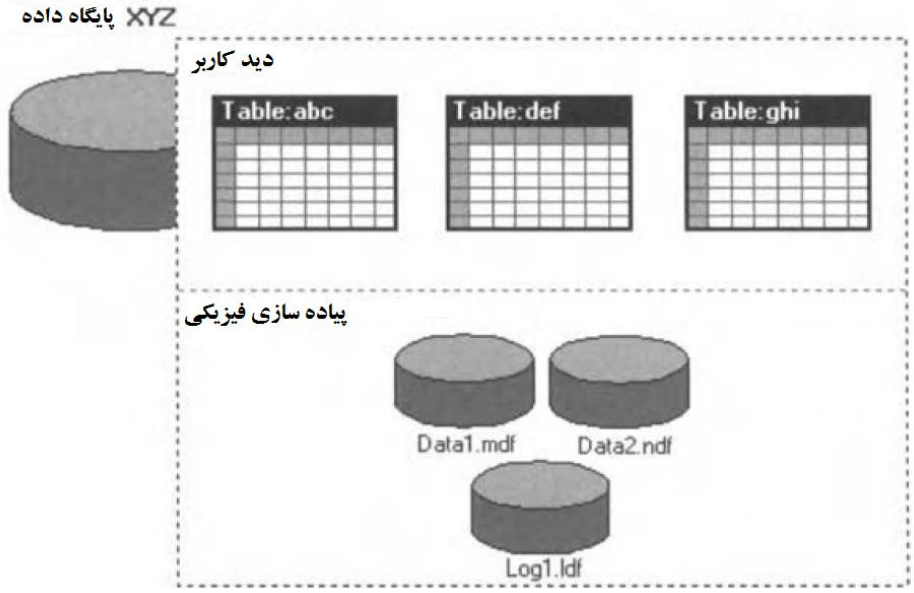
هنگامی که چند جدول در یک بند SQL درخواست شود، DB۲ باید یک اتصال انجام دهد. هنگام اتصال جدول ها، نوع دسترسی (اسکن فضای جدول یا اسکن شاخص) تعیین می‌نماید که چگونه هر جدول قابل دسترسی است، درک روش اتصال تعیین می‌کند که چگونه نتایج از جدول‌های مختلف با یکدیگر ترکیب می‌شوند و یک نتیجه منسجم به درخواست دهنده می‌فرستند. درحالی که بیش از دو جدول را می‌توان در یک عبارت SQL به هم متصل نمود، DB۲ همیشه عملیات اتصال را در چند مرحله انجام می‌دهد. هر مرحله تنها دو جدول را به هم متصل می‌کند و جدول ترکیبی به مرحله بعد

وارد می‌شود. جدول‌های برنامه نحوه اتصال این جدول‌ها به یکدیگر و نحوه دسترسی هر جدول را توصیف می‌نمایند.

۱۴-۳-٤ پایگاه داده ٤ – سرور SQL مایکروسافت

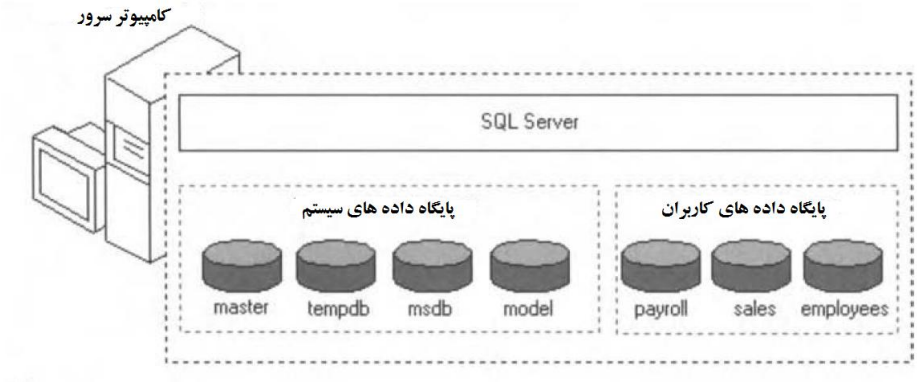
ساختار معماری

Microsoft SQL Server ۲۰۰۰ داده‌ها را در پایگاه داده ذخیره می‌نماید – جدول‌های کنترل شده به‌صورت روابط مدیریت شده در فایل‌های فیزیکی سازمان دهی می‌شوند (شکل ۱۴-۵). در هنگام استفاده از یک پایگاه داده، کار در ابتدا با مؤلفه‌های منطقی انجام می‌شود، مثل جدول‌ها، دیدگاه‌ها، روال‌ها و فضای کاربری. پیاده سازی فیزیکی روابط و درک آن‌ها به‌عنوان یک فایل کاملاً مشخص است.



شکل (۱۴-۵): مقایسه دیدگاه منطقی و دیدگاه فیزکی پایگاه داده

هر نمونه از یک سرور SQL چهار پایگاه داده سیستمی (master، model، tempdb و msdb) و یک یا چند پایگاه داده کاربردی دارد (شکل ۱۴-۶). بعضی از سازمان‌ها تنها یک پایگاه داده کاربری دارند که تمام اطلاعات موردنیاز سازمان را شامل می‌شود. برخی سازمان‌ها پایگاه داده مختلفی برای هر گروه در سازمان خود دارند و گاهی اوقات از یک پایگاه داده برای یک برنامه کاربردی استفاده می‌شود. نیازی نیست که نسخه‌های متعددی از یک موتور پایگاه داده سرور SQL اجرا گردد تا چند کاربر امکان دسترسی به پایگاه داده‌های موجود در یک سرور را پیدا کنند. یک نمونه از نسخه شرکتی یا استاندارد سرور SQL می‌تواند هزاران کاربری را که در چند پایگاه داده کار می‌کنند، به‌صورت همزمان کنترل و اداره نماید. هر نمونه از سرور SQL می‌تواند تمام پایگاه داده‌ها در نمونه را در دسترس کاربرانی قرار دهد که به آن نمونه متصل هستند و یا در معرض مجوز امنیتی تعریف‌شده قرار دارند.



شکل (۱۴-۶): ساختار فضای جدول منطقی

هنگام اتصال به یک نمونه از سرور SQL، اتصال شما به یک پایگاه داده خاص روی سرور مرتبط می‌شود. این پایگاه داده به پایگاه داده جاری معروف است. شما معمولاً به پایگاه داده ای متصل می‌شوید که توسط مدیر سیستم به‌عنوان پایگاه داده پیش فرض شناخته شده است، اگرچه می‌توانید از گزینه‌های اتصال موجود در API های پایگاه داده نیز برای مشخص کردن پایگاه داده دیگر استفاده کنید. شما می‌توانید از یک پایگاه داده با استفاده از عبارت Transact-SQLUSEdatabase_name یا یک تابع API به پایگاه داده دیگر سوییچ شوید که همین باعث تغییر زمینه پایگاه داده جاری شما می‌شود.

SQL Server ۲۰۰۰ به شما اجازه می‌دهد تا پایگاه‌های داده از یک نمونه از سرور SQL را تفکیک کنید و سپس آن‌ها را در نمونه دیگر به هم متصل نمایید یا حتی پایگاه داده را دوباره به همان نمونه ترکیب نمایید. اگر یک فایل پایگاه داده سرور SQL داشته باشید، می‌توانند به سرور SQL بگویید که چه زمانی آن فایل پایگاه داده را با یک نام مشخص از پایگاه داده الحاق نمایید. از تغییرات عمده موجود در سرور ۷,۰ نسبت به سرور ۶,۵، الگوریتم حافظه و استفاده از حافظه توسط اشیاء سرور SQL است که باعث بهبود عملکرد پایگاه داده می‌شود و می‌تواند کاری که مدیر پایگاه داده برای تنظیم حافظه انجام دهد را به حداقل برساند.

MicrosoftSQLServer۷,۰ روش تخصیص و دسترس به حافظه را تا حد زیادی بهبود بخشیده است.

برخلاف SQL Server ۶,۵، که در آن حافظه توسط مدیر پایگاه داده با تنظیمات پیکربندی مدیریت شده است، SQL Server ۷.۰ یک مدیر حافظه دارد که می‌تواند مدیریت حافظه دستی را حذف نماید. SQL Server ۶,۵ یک گزینه پیکربندی حافظه دارد که یک مقدار ثابتی از حافظه را در زمان راه‌اندازی تخصیص می‌دهد – یعنی، حافظه تقسیم بندی می‌شود و به‌صورت دستی مدیریت می‌گردد. اگر پارامتر فوق خیلی بزرگ باشد، سرور SQL نمی‌تواند آغاز به کار کند. مدیر پایگاه داده در ابتدا باید تعیین نماید که سرور SQL چه میزان حافظه باید مصرف نماید. برای مثال، با یک حافظه ۲۵۶ مگابایتی، سرور SQL ممکن است ۲۰۰ مگابایت بگیرد و ۵۶ مگابایت نیز برای سیستم‌عامل تخصیص دهد. این کار به نوبه خود یک هنر است نه علم. به سختی می‌توان برنامه ریزی نمود که یک پایگاه داده به تنهایی چه میزان پایگاه داده نیاز دارد یا چه چقدر برنامه کمتری یک سیستم‌عامل و برنامه‌های کاربردی دیگر نیاز دارند مثل سرورهای وب که روی همان کامپیوتر اجرا می‌شوند. استفاده از حافظه ثابت نیست، این احتمال وجود دارد که سرور SQL ممکن است از ساعت ۸ تا ۵ به حافظه بیشتری نیاز داشته باشد و سیستم‌عامل نیز ممکن است از ۵ تا ۸ به حافظه بیشتری برای کارهای شبانه نیاز داشته باشد. برای تغییر پیکربندی حافظه باید SQLServer۶,۵ خاموش و دوباره راه اندازی شود. با آغاز یک کار SQLServer۷,۰، تخصیص حافظه پویای آن تعیین می‌کند که بر اساس مقدار حافظه ای که سیستم‌عامل ویندوز NT و برنامه‌های کاربردی برای اجرا در ویندوز NT نیاز دارند، چه میزان حافظه باید تخصیص یابد. برای مثال، فرض کنید که ویندوز NT کلا ۵۱۲ مگابایت حافظه داشته باشد. هنگامی که سرور SQL راه اندازی می‌شود، ویندوز NT و برنامه‌های کاربردی روی آن از ۷۲

مگابایت حافظه استفاده می‌کنند. سرور SQL از حافظه موجود استفاده می‌نماید و ۵ مگابایت از آن را آزاد می‌گذارد. بنابراین، سرور SQL از ۴۳۵ مگابایت از ۵۱۲ مگابایت کل حافظه استفاده می‌کند ، ۷۲ مگابایت برای ویندوز NT و ۵ مگابایت حافظه آزاد. اگر برنامه کاربردی دیگر در ویندوز NT شروع به کار کند و ۵ مگابایت از حافظه آزاد را مورداستفاده قرار دهد، سرور SQL حافظه را به آن تخصیص می‌دهد تا مطمئن شود که ۵ مگابایت از فضای آزاد همیشه آزاد باقی می‌ماند. در مقابل، اگر ویندوز NT حافظه را به نحوی آزاد بگذارد که حافظه آزاد بیش از ۵ مگابایت باشد، سرور SQL از آن حافظه برای عملیات پایگاه داده استفاده می‌کند.

این الگوریتم حافظه پویا مزایای زیادی دارد. شما نیازی ندارید که درصد حافظه برای ویندوز NT، برنامه‌های کاربردی مبتنی بر ویندوز NT و سرور SQL را حدس بزنید. این می‌توانید در زمان استفاده شدید از ویندوز NT از صفحه بندی آن خودداری کنید و می‌توانید از حافظه آزاد NT در زمان کم کاری استفاده نمایید. الگوریتم حافظه برای نسخه دسکتاپ SQL Server ۷,۰ کارایی خوبی دارد. به جای استفاده از حافظه در زمان آزادی حافظه، حافظه را به سیستم‌عامل باز می‌گرداند. علت این است که نسخه دسکتاپ برنامه‌های دیگری را نیز اجرا می‌کند.

معماری قفل گذاری

Microsoft SQL Server ۲۰۰۰ از قفل برای کنترل همزمانی میان کاربرانی که به‌صورت همزمان در حال تغییر پایگاه داده هستند، استفاده می‌کند. به‌صورت پیش فرض، SQL Server هم تراکنش‌ها و هم قفل‌ها را بر اساس یک اتصال مدیریت می‌نماید. برای مثال، اگر یک برنامه کاربردی دو اتصال

سرور SQL را باز کند، قفل‌های موردنیاز یک اتصال را نمی‌توان با اتصال دیگر اشتراک گذاشت. اتصال نمی‌تواند قفل‌هایی که با قفل اتصالات دیگر درگیر هستند، به دست آورد. تنها اتصالات محدود تحت تأثیر این قاعده نیستند. قفل‌های سرور SQL در سطوح مختلف گرانولیته در پایگاه داده اعمال می‌شوند. قفل‌ها را می‌توان روی ردیف ها، کلیدها، چندین کلید، شاخص، جدول یا پایگاه داده به دست آورد. سرور SQL سطح مناسبی که در آن می‌توان قفل برای هر بند ،Transact-SQL قرار داد، تعریف می‌کند. سطحی که در آن قفل‌ها را می‌توان به دست آورد، برای اشیاء مختلف متفاوت هستند که معمولاً توسط یک پرسش به آن‌ها مراجعه می‌گردد – برای مثال، یک جدول ممکن است خیلی کوچک باشد و روی آن یک قفل اعمال شده باشد، در حالی که در جدول بزرگ‌تر ممکن است قفل‌های ردیفی اعمال شده باشد. سطحی که در آن قفل گذاشته می‌شود، لزوما توسط کاربران مشخص نمی‌شود و نیازی به تنظیمات مدیر ندارند. هر نمونه از سرور SQL تضمین می‌کند که قفل‌های هر سطح از گرانولیته در سطح دیگری نیز مجاز هستند. چند حالت قفل وجود دارد: قفل مشترک، به روز، منحصربه فرد، دقیق و طرح.

اگر چند اتصال در انتظار برای قفل‌های درگیر روی یک منبع مسدود شوند، قفل‌ها روی اولین اتصال می‌روند، اولین پایه سرویس‌دهی شده به‌عنوان اتصال بعدی قفل‌های آن‌ها را آزاد می‌کند. در حمایت از عملیات همزمان، سرور SQL یک الگوریتم برای تشخیص بن بست‌ها دارد. اگر یک نمونه از سرور SQL بن‌بستی را شناسایی کند، یک تراکنش را به پایان می‌برد و به دیگری اجازه می‌دهد که ادامه یابد. سرور SQL می‌تواند به‌صورت پویا گرانولیته یا نوع قفل‌ها را تشدید یا تخفیف دهد. برای مثال

اگر برای به روزرسانی به چندین ردیف قفل نیاز باشد، و درصد زیادی از جدول نیز قفل باشد، قفل‌های ردیفی به یک قفل جدولی تشدید می‌شوند. اگر به یک قفل جدولی دست یابیم، قفل‌های ردیفی رها می‌شوند. SQL Server ۲۰۰۰ به ندرت قفل‌ها را تقویت می‌کند، بهینه ساز پرسش معمولاً گرانولیته قفل صحیح را در زمان کامپایل برنامه اجرا انتخاب می‌نماید.

زبان پرسش ساختاریافته

برای کار با اطلاعات موجود در پایگاه داده، باید از مجموعه ای از فرامین و بندها (زبان‌ها) استفاده کنید که توسط نرم افزار DBMS تعریف‌شده باشند. می‌توان از زبان‌های مختلفی با پایگاه‌های داده رابطه ای استفاده نمود، رایج ترین آن‌ها SQL است. موسسه استاندارد ملی آمریکا (ANSI) و سازمان بین المللی استاندارد (ISO) استانداردهای نرم افزار را تعریف می‌کنند که شامل استانداردهای زبان SQL می‌شود. SQL Server ۲۰۰۰ از سطح ورودی SQL،۹۲ پشتیبانی می‌کند، استاندارد SQL توسط ANSI در سال ۱۹۹۲ منتشر شده است. گویش SQL تحت حمایت سرور SQL مایکروسافت قرار دارند که به نام Transact-SQL (SQL) شناخته می‌شود. T-SQL زبان اولیه مورداستفاده برنامه‌های کاربردی سرور SQL مایکروسافت می‌باشد.

خلاصه ای از ویژگی‌های خاص

Microsoft SQL Server ۲۰۰۰ به کاربران یک پلتفرم پایگاه داده بدون درز و عالی برای پردازش تراکنش آنلاین (OLTP)، انبار داده و برنامه‌های کاربردی تجارت الکترونیک فراهم می‌کند. پیشرفت‌هایی که در نسخه ۷,۰ سرور SQL اعمال شده، یک محیط XML منسجم ایجاد کرده که

ویژگی داده کاوی جدیدی در سرویس‌های تحلیلی می‌افزاید و فناوری ذخیره سازی را با سرویس‌های متاداده تقویت می‌کند. SQL Server ۲۰۰۰ باعث تقویت عملکرد، قابلیت اطمینان، کیفیت و سهولت استفاده از SQL Server ۷,۰ می‌شود.

۴-۱۴ آزمایش تحلیل عملکرد پلتفرم

برای انجام یک مقایسه درست بین این چهار پایگاه داده به یک معیار ساده نیاز داریم که از مزیت ویژگی‌های خاص موجود در پایگاه‌های داده بهره نگیرد. برای انجام این کار، گروه کاری ما روی آخرین معیارهای انجمن پردازش تراکنش (www.tpc.org،TPC) تحقیق کرده است. سه معیار وجود داشت که به ما امکان آزمایش OLTP را می‌دادند. این معیارها عبارتند از: TPC-C ، H - TPC و TPC-R. از این بین، نسخه پنجم TPC-C به‌عنوان آخرین معیار OLTP طراحی‌شده است. بااین‌حال، TPC تا کنون نتوانسته این معیار را در اختیار عموم قرار دهد. به این ترتیب، H - TPC و TPC-R بازبینی شدند. این دو معیار برای پایگاه‌های داده حامی تصمیم در انبار داده هستند. مشخص‌شده که TPC - H نسخه بازبینی شده از TPC_R است. تنها تفاوت میان این دو معیار قوانین مربوط به پیاده سازی است. تصمیم با تیم ارزیابی عملکرد بود که به دلیل نداشتن زمان کافی، گروه باید معیارها را فرا می‌گرفتند، پایگاه داده را بلد می‌شدند و تحلیل واقعی انجام می‌دادند تا برای استفاده از TPC - H بهترین روش ایجاد می‌شود. این تصمیم به هر صورت گرفته شد، برای اینکه ما در حین پیاده‌سازی معیار، گزینه‌های کمتری پیش رو داشتیم. این معیار شامل یک بسته از پرس وجوهای

ادهاک کسب‌وکار گرا و اصلاحات داده همزمان بود. هدف اصلی آن، کمک به آزمایش حجم کثیری از داده و اجرای پرس وجوهای خیلی پیچیده بود. کار بعدی تعیین نوع تراکم بار و پایگاه‌های داده ای که اجرا می‌شدند و نحوه اجرای این بارها بود.

۱۴-۴-۱ تراکم بار

نگرانی اصلی تعیین معیار یک سیستم، مشخص کردن تراکم بار است. تراکم بار یک کامپیوتر به‌صورت مجموعه ای از تمام ورودی‌های که سیستم از محیط دریافت می‌کند تعریف می‌شود. گروه‌ها از پرس وجوهای تعریف‌شده در معیار TPC - H (جدول ۱۴-۴) به‌عنوان تراکم بار اصلی استفاده کردند.

۱۴-۴-۲ آمادگی برای آزمایش

برای اینکه از استاندارد بودن آزمایش مطمئن شویم، یکی از آزمون‌های کارایی در معیار TPC - H انتخاب و اصلاح شدند. اصلاحات برنامه ریزی شده عبارت بودند از درج تازگی طبق نیاز مشخصات TPC - H و استفاده از شاخص گذاری. بنابراین، دو اجرا انجام شد: یکی بدون شاخص و رفرش و دیگری با شاخص و رفرش. رفرش کردن موردنیاز مشخصات TPC - H است اما موقعیت این رفرشها در پرسش‌ها به آزمون کننده واگذار شده است. برای حصول اطمینان از اینکه تمام پایگاه‌های داده پرسش را به یک شیوه انجام می‌دهند، از آزمون شماره ۱ (پیوست A از معیار TPC - H) با سه روش رفرش از پیش تعریف‌شده استفاده شد.

جدول (۱۴-۴) معیارهای TPC – H

پرسش ۱-گزارش مختصر قیمت‌گذاری	این پرسش یک گزارش مختصر قیمت‌گذاری از تمام اقلامی را که در یک تاریخ به خصوصی منتقل شده اند خواهد داد (متغیر جایگزین). زمان جابه جایی بین ۶ تا ۱۲۰ روز برای بزرگترین کشتی موجود در پایگاه داده می‌باشد. در هر گروه تعدادی از اقلام گنجانده شده است.
پرسش۲- تامین کننده با حداقل هزینه	این پرسش برای هر بخش از یک نوع و اندازه خاص مشخص می‌کند که کدام عرضه‌کننده می‌تواند آن محصول را با کمترین هزینه تهیه کند. اگر چند عرضه‌کننده در آن منطقه، قیمت پایین یکسان را برای آن بخش سفارش ارائه بدهند، این پرسش محصولات را تامین کنندگانی با یکصد گردش حساب بالا فهرست می‌کند.
پرسش۳- اولویت حمل و نقل	این پرسش اولویت حمل و نقل و درآمد بالقوه را تعیین می‌کند و به‌صورت مجموع قیمت افزوده سفارشاتی که بالاترین درآمد را میان کالاهایی که در یک تاریخ مشخص بارگیری نشده بودند، دارد. اگر بیش از ۱۰ کالای بارگیری نشده وجود داشته باشد، تنها ۱۰ سفارش با بیشترین درآمد فهرست می‌شود.
پرسش۴- بررسی اولویت سفارش	این پرسش تعداد سفارشاتی که در یک فصل سال سفارش داده‌شده اند را نشان می‌دهد که در آن حداقل یک کالا بعد از تاریخ تصویب شده دریافت می‌شود.
پرسش۵- ظرفیت تامین کنندگان محلی	این پرسش برای هر کشور یک منطقه، میزان درآمد به دست آمده از تراکنش‌های کالا را فهرست می‌نماید، در این تراکنش‌ها مشتریان کالا سفارش می‌دهند و عرضه کنندگان آن‌ها را در همان کشور عرضه می‌نمایند. این پرسش ها شامل کالاهای سفارشی در یک سال مشخص می‌شود.

پرسش۶-پیش بینی تغییر درآمد	این پرسش مقدار افزایش درآمد را حساب می‌کند که ناشی از حذف تخفیف در گستره یک شرکت خاص در محدوده مشخصی از سال می‌باشد.
پرسش۷- حجم حمل و نقل	این پرسش مقدار کالاهای مبادله شده میان کشورهای خاص را تعیین می‌کند که به مذاکرات دوباره قراردادهای حمل و نقل کمک می‌کند.
پرسش۸- سهم بازار ملی	این پرسش تعیین می‌کند که چگونه سهم بازار از یک کشور معلوم در یک منطقه معلوم در طی دو سال برای یک نوع کالای خاص تغییر کرده است.
پرسش ۹- اندازه گیری سود نوع محصول	این پرسش تعیین می‌کند که برای یک خط تولید چقدر سود به دست می‌آید و توسط سال و کشور عرضه کننده تقسیم می‌شود.
پرسش ۱۰- گزارش کالاهای بازگشتی	این پرسش مشتریانی که ممکن است با کالاها مشکل داشته باشند را شناسایی می‌کند.
پرسش ۱۱- شناسایی مهم موجودی	این پرسش مهمترین زیرمجموعه موجودی تامین کنندگان در یک کشور خاص را پیدا می‌کند.
پرسش ۱۲- حالات حمل و نقل و اولویت سفارش	این پرسش بررسی می‌کند که آیا انتخاب حالات ارزان‌تر حمل و نقل می‌تواند تأثیر منفی روی سفارشاتی با اولویت بالا داشته باشد و باعث شود که کالاهای بیشتری بعد از تاریخ معلوم به دست مشتری برسد یا خیر.
پرسش ۱۳- توزیع مشتری	این پرسش رابطه میان مشتریان و اندازه سفارشات آن‌ها را تعیین می‌کند.
پرسش ۱۴- تاثیر توسعه	این پرسش درصد درآمد سالانه برای کالاهای توسعه‌یافته را پیدا خواهد کرد (دوره زمانی به‌عنوان پارامتر جایگزین در زمان ایجاد پرسش با برنامه QGEN است، از متغیر Seed استفاده می‌کند).

پرسش ۱۵-برترین تامین کنندگان	این پرسش به ما نشان می‌دهد که تامین کنندگان بیشتر درآمد را در دوره زمانی خاصی به دست آورده است (دوره زمانی یک پارامتر جایگزین است که هنگام ایجاد پرسش با برنامه QGEN با کمک متغیر Seed انتخاب می‌شود.)
پرسش ۱۶- رابطه کالاها و تامین کنندگان	این پرسش تعداد تامین کنندگانی که می‌توانند کالاهای موردنیاز مشتریان را تهیه کنند، پیدا می‌کند. برند، نوع و اندازه محصول پارامترهای جایگزین در هنگام ایجاد پرسش با برنامه کاربردی QGEN با متغیر Seed انتخاب می شوند.
پرسش۱۷-درآمد سفارش/تعداد کم	این پرسش اقلام خطی را برای یک برند و نوع کالا پیدا می‌کند و میانگین مقدار کالاهای سفارشی را اگر مقدار کمتر از ۲۰ باشد، برای یک دوره هفت ساله تعیین می‌نماید (برند و کشتی کانتینر پارامترهای جایگزینی هستند که هنگام ایجاد پرسش با برنامه QGEN با استفاده از متغیر Seed انتخاب می‌شوند).
پرسش ۱۸-مشتریان با سفارش انبوه	این پرسش ۱۰۰ مشتری برتر که سفارش انبوه داشته اند را پیدا خواهد کرد (این مقدار پارامتر جایگزین منتخب در زمان ایجاد پرسش با برنامه کاربردی QGEN با استفاده از متغیر Seed است).
پرسش ۱۹-بازده تخفیف	این پرسش بازده یا سود تخفیف برای همه سفارشات مربوط به تمام کالاها را پیدا خواهد کرد (نوع کالا، کانتینر، مقدار، وضعیت حمل و نقل و دستورالعمل حمل و نقل پارامترهای منتخب در هنگام ایجاد پرسش با برنامه QGEN با استفاده از متغیر Seed است).
پرسش ۲۰-توسعه بخش بالقوه	این پرسش تامین کنندگانی را پیدا می‌کند که به مدت یک سال مشخص مقدار زیادی کالا در دسترس داشتند (نام و تاریخ کالا

	پارامترهای جایگزین منتخب در هنگام ایجاد پرسش با برنامه QGEN با استفاده از متغیر Seed می‌باشد).
پرسش ۲۱- تامین کنندگانی که سفارشات را در حالت انتظار نگه می‌دارند	این پرسش عرضه کنندگان یک کشور را پیدا می‌کند که محصول آن بخشی از سفارش چند عرضه کننده است که در آنجا آن‌ها قادر به ارسال به موقع محصول نیستند (این کشور یک پارامتر جایگزین در زمان ایجاد پرسش با برنامه QGEN با استفاده از متغیر Seed می‌باشد).
پرسش ۲۲-فرصت فروش جهانی	این پرسش مشتریان موجود در یک کد کشوری که به مدت ۷ سال سفارشی نداشتند اما بودجه مثبت دارند را پیدا خواهد کرد (کدهای کشورها پارامترهای جایگزین در هنگام ایجاد پرسش با استفاده از متغیر Seed در برنامه QGEN می‌باشد)

بدون نمایه سازی

اجرای بدون (نمایه) شاخص به‌عنوان یک پایه مورداستفاده قرار می‌گیرد که با آن می‌توان اجرای با شاخص را مقایسه نمود. گروه می‌داند که نمایه سازی تا حدود زیادی زمان لازم برای انجام آزمون عملکرد را کاهش می‌دهد اما نتیجه مطلوبی دارد. در هنگام نگارش این متن، هیچ گروهی نتوانست یک اجرای بدون نمایه سازی روی سیستم‌های پایگاه داده تکمیل نماید. در ادامه این فصل روال‌های مورد نظر بررسی می‌شوند. دلیل اصلی عدم انجام یک آزمون بدون نمایه فقدان زمان بود. می‌توانستیم با یک آزمون جداگانه از هر پرسش در آزمون عملکرد بدون تازگی انجام دهیم. این نتایج به‌صورت مفصل در بخش (۱۴-۵) تعریف‌شده اند.

نمایه سازی

با در نظر گرفتن بخش قبل، اگر عدم نمایه سازی انجام نشود، نمایه سازی نیز قابل انجام نیست. بااین‌حال، ارائه مربوط به هر گروه انجام شد تا تعیین شود چه چیزی باید شاخص گذاری گردد. کارهای آن‌ها در جدول (۱۴-۵) ارائه شده که می‌توانند برای محققان آینده قابل استناد باشند.

اجرای تمام پرسش‌ها با همدیگر

یک انجمن ماژول‌های ذخیره شده مداوم (PSM) مسئولیت ایجاد فایلی را به عهده گرفته که با استفاده از TPC-H که در پیوست A تعریف‌شده، پرسش‌های تمام گروه‌ها را اجرا خواهد کرد.

جدول (۱۴-۵): شاخص‌های پیشنهادی برای آزمون معیار

کلید های خارجی
CREATE INDEX tpch. c_nk ON tpch. customer(c_nationkey ASC)
CREATE INDEX tpch. s_nk ON tpch. supplier(s_nationkey ASC
CREATE INDEX tpch. ps_pk ON tpch. partsupp(ps_suppkey ASC)
CREATE INDEX tpch. ps_sk ON tpch. partsupp(ps_suppkey ASC)
CREATE INDEX tpch. l_ok ON tpch. lineitem(l_orderkey ASC)
کلید های اولیه
CREATE UNIQUE INDEX tpch. c_ck ON tpch. customer(c_custkey
CREATE UNIQUE INDEX tpch. p_pk ON tpch. part(p_partkey ASC
CREATE UNIQUE INDEX tpch. s_sk ON tpch. supplier(s_suppkey
CREATE UNIQUE INDEX tpch. o_ok ON tpch. orderd(o_orderkey
CREATE UNIQUE INDEX tpch. ps_pk_sk ON tpch. partsupp(ps_pa
CERATE UNIQUE INDEX tpch. ps_sk_pk ON tpch. partsupp(ps_su
فیلدهای تاریخ مؤثر و مفید

```
CREATE INDEX tpch. o_od ON tpch. orders(o_orderdate ASC)
CREATE INDEX tpch. l_sd ON tpch. lineitem(l_shipdate ASC)
```

این کمیته باید مشخص کند که چگونه زمان هر اجرا، زمان کل و نحوه اداره هر تازه سازی را داشته باشد. متاسفانه، تنها گروهی که توانست از فایل استفاده کند، گروه سرور مایکروسافت SQL بود. گروه‌های دیگر باید فایل را تغییر می‌دادند تا بتوانند با آن کار کنند. به دلیل نداشتن زمان کافی، تصمیمی برای آن گرفته نشد در عوض به‌صورت جداگانه پرسش‌ها اجرا شدند.

۱۴-۴-۳ روال‌های پلتفرم برای هر پیکربندی

برای روال‌های اصلی باید روی هر پیکربندی معیار را اجرا می‌کردیم. ابتدا، لازم بود که برای پایگاه‌های داده، جدول ایجاد شود. دوم، جدول‌هایی که به تازگی ایجاد شدند با داده آزمون معیار شناخته شدند. سوم، روی صف‌ها جداگانه نمونه‌های متعددی ایجاد شد تا تضمین کند که سیستم‌ها به‌صورت کامل اجرا می‌شوند و به هر تیمی روشی برای بهینه سازی سیستم پیش از آزمایش فراهم شود. در نهایت، روی هر سیستم از آزمون‌های عملکرد استفاده شد.

۱۴-۵ نتایج

جدول (۱۴-۶) نتایج آزمایشات مربوط به هر گروه را برای چهار پایگاه داده تحت مطالعه نشان می‌دهد. توجه کنید که زمان مربوط به هر پرسش به ثانیه محاسبه شده است. همانطور که از جدول پیداست، مقایسه یک به یک پایگاه داده میسر نمی‌باشد. علت این است که بسیاری از عوامل به‌صورت مختصر

توصیف شده اند. به نظر می‌رسد که مهمترین دلیل مربوط به استفاده از حافظه توسط هر پایگاه داده باشد. تنها مایکروسافت و DB۲ توانستند از ۱۰۰٪ حافظه موجود خود استفاده کنند، در حالی که اوراکل و Informix تنها از یک سوم حافظه موجود استفاده کردند. برای مقایسه سیستم‌های پایگاه داده با یکدیگر یک سری مقایسه انجام گرفت. اول، با توجه به جدول (۶-۱۴)، کاملاً مشخص است که مایکروسافت SQL ازنظر عملکرد بهتر از DB۲ است. به دلیل دیدگاه منسجم داده موجود در جدول (۱۴-۶)، ما متوجه شدیم که DB۲ حدود ۱,۰۳ برابر سریع تر از سرور SQL اجرا می‌شود. این توجه داشته باشید که DB۲ تنها برای یک آزمون عملکرد بهتری نسبت به SQL داشته است که به مشتریانی با سفارش انبوه نگاه می کند. در تمام موارد عملکرد سرور SQL بهتر از DB۲ بوده است (حدود ۵۳٪).

با توجه به این نتایج، می‌توانیم مایکروسافت SQL را با Informix و اوراکل مقایسه کنیم.

جدول (۱۴-۶): نتایج مربوط به آزمایشات پلتفرم TPC-H

شماره پرسش	Informix	DB۲	SQL	Oracle
۱	۵۱۰	۵۱۰	۴۳۱	**۳۳۵**
۲	۲۱,۶۰۰	۳,۱۸۰	۴۴	**۸۱,۸۴۰**
۳	۳,۱۸۰	۸۴۲	۴۶۵	**۵۳۲**
۴	۱,۵۳۰	خطا: ناشناخته	۳۰۰	**۱,۸۱۸**
۵	خطا: حافظه	خطا: ناشناخته	۳۱۴	**۲۰,۰۴۰**
۶	۲۵۰	۳۸۸	۲۴۵	**۲۶۹**
۷	خطا: ناشناخته	خطا: ناشناخته	۳۱۱	**۴۶۶**
۸	خطا: نحوی	۲۷,۱۵۷	۳۰۹	**خطا: نحوی**
۹	خطا: نحوی	۱,۲۸۶	۴۰۹	**خطا: نحوی**

۵۲۹	۳۱۶	۳۰,۶۷۲	۲,۷۶۰	۱۰
۱۹۹	۴۱	۱۴۴	۹۵	۱۱
٤٦٤	۲۹۸	۴۵۳	۴۸۰	۱۲
خطا: نحوی	۸۹	۱۲۶	عدم اجرا	۱۳
۳۳۱	۲۵۳	۴۶۰	۶۶۰	۱۴
٦۰۰	۲۴۷	۳۳۸	۶۰۰	۱۵
۲,۸٤۸	۵۴	۹۱۵	۲۴۰	۱۶
۲,۷۰۰	۵۰۴	۷۹۷	عدم اجرا	۱۷
۸,۱۰۰	۱,۱۶۹	۱,۱۲۷	۱۲,۶۶۰	۱۸
۱,۰۹۱	۲۵۰	۳۵۴	۲۴۰	۱۹
عدم اجرا	۲۸۸	۴۶۷	عدم اجرا	۲۰
۱,۵٦۰	۸۰۹	۱,۵۶۹	خطا: ناشناخته	۲۱
خطا: نحوی	۶۱	۵۹۴	خطا: ناشناخته	۲۲
۱۰۰٪	۱۰۰٪	۱۰۰٪	۱۰۰٪	میزان CPU استفاده شده
۳۸٪	۱۰۰٪	۱۰۰٪	۳۰٪	میزان حافظه استفاده شده

فرض ما کاهش مقدار حافظه ای است که مایکروسافت SQL برای همان سطح در Informix و اوراکل استفاده کرده است. برای این کار، ما از یک روش خطی برای پیدا کردن زمان‌های جدید استفاده می‌کنیم. فرض بر این است که اگر مقدار حافظه حدود ۵۰٪ کاهش یابد، زمان دو برابر می‌شود. در این آزمایشات، از آنجایی که Informix و اوراکل از یک سوم حافظه سرور SQL استفاده می‌کنند،

فرض می‌کنیم که زمان نیز باید ۳,۳۳ برابر زمان یک سیستم با یک سوم حافظه باشد. لازم به ذکر است که این فرض هنوز آزمایش نشده است. این، از آنجایی که تمام پرسشها را نمی‌توان روی Informix یا اوراکل اجرا نمود، تنها پرسشهایی که روی این سیستم‌ها اجرا شده اند را می‌توانیم در نظر بگیریم. بقیه پرسشها را باید فراموش کنیم. توجه کنید که ما دو جدول جدید داریم، جدول (۱۴–۷) و (۱۴–۸)، برای Informix و اوراکل.

جدول (۱۴–۷): سرور SQL اندازه گیری شده نسبت به Informix

شماره پرسش	Informix	SQL
۱	۵۱۰	۱,٤۳۷
۲	،	۱٤۷
۳	۳,۱۸۰	۱,۵۵۰
۴	۱,۵۳۰	۱,۰۰۰
۵	،	۱,۰٤۷
۶	۲۵۰	۸۱۷
۷	،	۱,۰۳۷
۸	،	۱,۰۳۰
۹	،	۱,۳٦۳
۱۰	۲,۷۶۰	۱,۰۵۳
۱۱	۹۵	۱۳۷
۱۲	۴۸۰	۹۹۳
۱۳	،	۲۹۷
۱۴	۶۶۰	۸٤۳
۱۵	۶۰۰	۸۲۳

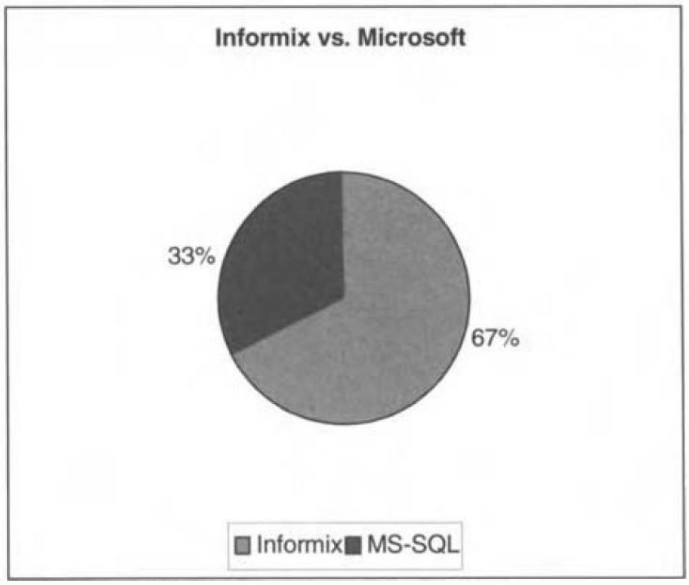
۱۸۰	۲۴۰	۱۶
۱,۶۸۰	،	۱۷
۳,۸۹۷	۱۲,۶۶۰	۱۸
۸۳۳	۲۴۰	۱۹
۹۶۰	،	۲۰
۲,۶۹۷	،	۲۱
۲۰۳	،	۲۲
٪۳۰	٪۳۰	میزان حافظه استفاده شده

جدول (۱۴-۸): سرور SQL نسبت به اوراکل

SQL	Oracle	شماره پرسش
۱,۴۳۷	۳۳۵	۱
۱۴۷	۸۱,۸۴۰	۲
۱,۵۵۰	۵۳۲	۳
۱,۰۰۰	۱,۸۱۸	۴
۱,۰۴۷	۲۰,۰۴۰	۵
۸۱۷	۲۶۹	۶
۱,۰۳۷	۴۶۶	۷
۱,۰۳۰	،	۸
۱,۳۶۳	،	۹
۱,۰۵۳	۵۲۹	۱۰
۱۳۷	۱۹۹	۱۱
۹۹۳	۴۶۴	۱۲

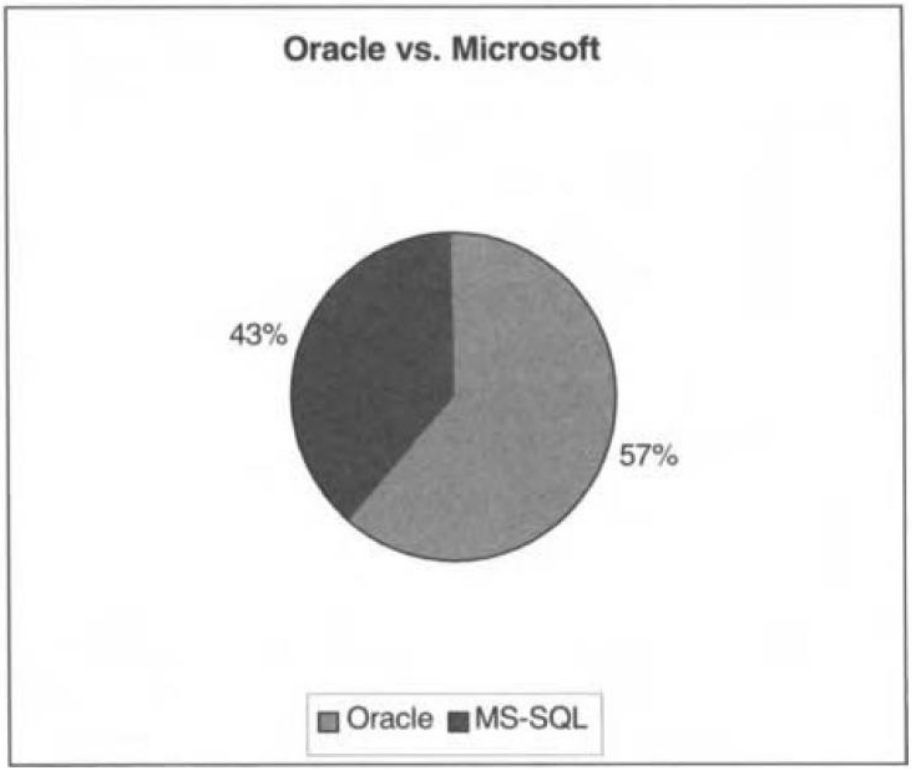
۲۹۷	،	۱۳
۸٤۳	۳۳۱	۱۴
۸۲۳	۶۰۰	۱۵
۱۸۰	۲,۸۴۸	۱۶
۱, ٦۸۰	۲,۷۰۰	۱۷
۳,۸۹۷	۸,۱۰۰	۱۸
۸۳۳	۱,۰۹۱	۱۹
۹٦۰	،	۲۰
۲, ٦۹۷	۱,۵۶۰	۲۱
۲۰۳	،	۲۲
٪۳۸	٪۳۸	میزان حافظه استفاده شده

با مقایسه سرور مایکروسافت SQL و Informix توسط مقادیر عملکرد مشاهده می‌کنیم که ۷ مورد از ۱۲ مورد پرسش روی Informix سریع تر از سرور مایکروسافت SQL اجرا می‌شوند. طبق شکل (۱۴–۷)، Informix حدود ۶۷٪ سریع تر است. اگر با دقت بیشتری به این اطلاعات نگاه کنیم، متوجه می‌شویم که پایگاه داده Informix بین ۳,۴ برابر سریع تر از سرور SQL اجرا می‌شود. تفاوت کلی اندازه گیری شده باعث می‌شود که عملکرد Informix ۱,۴۸ برابر عملکرد سرور SQL باشد.



شکل (۱۴-۷): مقایسه سرور Informix و مایکروسافت SQL

با مقایسه سرور SQL مایکروسافت با اوراکل با استفاده از مقادیر عملکرد محاسبه شده، می‌بینیم که ۹ مورد از ۱۷ پرسش در اوراکل، سریع تر از سرور SQL مایکروسافت اجرا می‌شوند. با توجه به شکل (۱۴-۸)، اوراکل به طور میانگین ۵۷٪ سریع تر است.



شکل (۱۴-۸): مقایسه سرور Oracle و مایکروسافت SQL

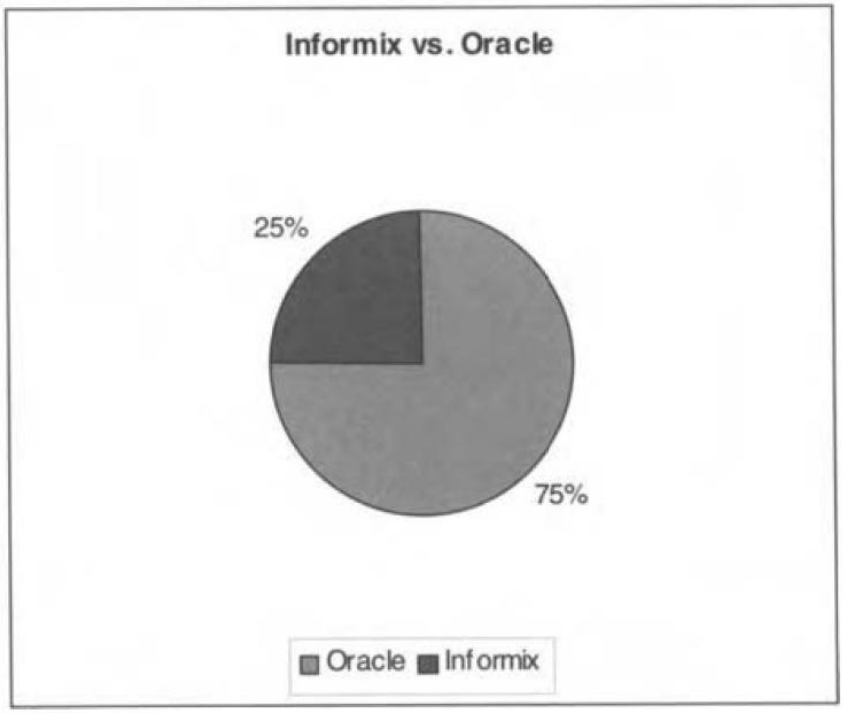
اگر با دقت بیشتری به این اطلاعات نگاه کنیم، متوجه می‌شویم که سیستم پایگاه داده اوراکل بین ۴,۲ برابر سریع تر از سرعت سرور SQL است. تفاوت کلی نتایج نشان می‌دهد که عملکرد اوراکل ۱,۷۶ برابر عملکرد سرور SQL است. بنابراین تحت مفروضاتی که برای اجرای این تحلیل استفاده کردیم، Informix و اوراکل در بیشتر اوقات عملکرد بهتری نسبت به سرور SQL دارند. بااین‌حال هنوز لازم است که بدانیم آیا پایگاه داده Informix بهتر از پایگاه داده اوراکل است یا خیر. برای انجام این

تحلیل اطلاعات هر دو پایگاه داده را نرمال سازی کردیم تا در ۱۰۰٪ استفاده از حافظه آن‌ها را اجرا کنیم. جدول (۱۴-۹) نتایج جدیدی را نشان می‌دهند. مقایسه اوراکل و Informix با استفاده از مقادیر عملکرد محاسبه شده، متوجه می‌شویم که ۹ مورد از ۱۲ پرسش روی اوراکل سریع تر از Informix اجرا می‌شوند. با توجه به شکل (۱۴-۹)، اوراکل به طور متوسط نسبت به Informix، ۷۵٪ سریع تر است. اگر با دقت بیشتری به این اطلاعات نگاه کنیم، متوجه می‌شویم که پایگاه داده اوراکل بین ۷,۷ سریع تر از سرور Informix برای دو پرسش اجرا می‌شود. برای پرسشی ۱۶ (کالا/عرضه کننده) و پرسشی ۱۸ (مشتریانی با سفارش انبوه)، پایگاه داده Informix به طور قابل توجهی سریع تر از اوراکل کار می‌کند: به طور دقیق ۳,۵ تا ۹,۳۶ برابر سریع تر. بااین‌حال، از آنجایی که این پرسشها به نظر پرت هستند، ما با حذف تأثیر این دو بهترین پرسشی اوراکل معیار کلی عملکرد را محاسبه کردیم. تفاوت کلی عملکرد اوراکل ۱,۳۸ برابر عملکرد سرور پویای Informix است. بنابراین، با استفاده از این مفروضات و آزمون ها، اوراکل ازنظر عملکرد برنده است.

جدول (۱۴-۹): مقایسه Informix و Oracle

شماره پرسش	Informix	Oracle
۱	۱,۷۰۰	۸۷۲
۲	،	،
۳	۱۰,۶۰۰	۱,٤۰۰
۴	۵,۱۰۰	٤,۷۸٤
۵	،	،
۶	۸۳۳	۷۰۸

٧	،	،
٨	،	،
٩	،	،
١٠	٩,٢٠٠	١,٣٩٢
١١	٣١٧	٥٢٤
١٢	١,٦٠٠	١,٢٢١
١٣	،	،
١٤	٢,٢٠٠	٨٧١
١٥	٢,٠٠٠	١,٥٧٩
١٦	٨٠٠	٧,٤٩٥
١٧	،	،
١٨	٤٢,٢٠٠	٢١,٣١٦
١٩	٨٠٠	٢,٨٧١
٢٠	،	،
٢١	،	،
٢٢	،	،
ميزان حافظه استفاده شده	٪١٠٠	٪١٠٠



شکل (۱۴-۹): هزینه و عملکرد

البته عملکرد همه چیز نیست بلکه هزینه را نیز باید در نظر داشت. برای در نظر گرفتن هزینه ما یک مقدار برای هزینه خرید در هر سیستم پایگاه داده به دست آوردیم و سپس هزینه هر ثانیه برای عملکرد را محاسبه کردیم. برای اینجاد یک مقایسه دقیق در هزینه ما میانگین زمانی که هر پایگاه داده برای اجرای پرسش‌های مورداستفاده در مدل‌ها نیاز دارد را به دست آوردیم و سپس از آن مقدار برای تقسیم هزینه خرید استفاده کردیم (جدول ۱۴-۱۰).

جدول (۱۴-۱۰): مقایسه هزینه/عملکرد

شماره پرسش	Informix	DB۲	SQL	Oracle
۱	۵۱۰	۱,۷۰۰	۱,۴۳۷	٤٢٤
۲	۲۱,۶۰۰	۱۰,۶۰۰	۱۴۷	١٠٣,٦٦٤
۳	۳,۱۸۰	۲,۸۰۷	۱,۵۵۰	٦٧٤
۴	۱,۵۳۰	۰	۱,۰۰۰	٢,٣٠٣
۵	۰	۰	۱,۴۰۷	٢٥,٣٨٤
۶	۲۵	۱,۲۹۳	۸۱۷	٣٤١
۷	۰	۰	۱,۰۳۷	٥٩٠
۸	۰	۹۰,۵۲۳	۱,۰۳۰	٠
۹	۰	۴,۲۸۷	۱,۳۶۳	٠
۱۰	۲,۷۶۰	۱۰۲,۲۴۰	۱,۰۵۳	٦٧٠
۱۱	۹۵	۴۸۰	۱۳۷	٢٥٢
۱۲	۴۸۰	۱,۵۱۰	۹۹۳	٥٨٨
۱۳	۰	۴۲۰	۲۹۷	٠
۱۴	۶۶۰	۱,۵۵۳	۸۴۳	٤١٩
۱۵	۶۰۰	۱,۱۲۷	۸۲۳	٧٦٠
۱۶	۲۴۰	۳,۰۵۰	۱۸۰	٣,٦٠٧
۱۷	۰	۲,۶۵۷	۱,۶۸۰	٣,٤٢٠
۱۸	۱۲,۶۶۰	۳,۷۵۷	۳,۸۹۷	١٠,٢٦٠
۱۹	۲۴۰	۱,۱۸۰	۸۳۳	١,٣٨٢
۲۰	۰	۱,۵۵۷	۹۶۰	٠
۲۱	۰	۵,۲۳۰	۲,۹۶۷	١,٩٧٦
۲۲	۰	۱,۹۸۰	۲۰۳	٠
زمان کل	۴۴,۸۰۵	۲۳۷,۹۳۰	۲۴,۰۲۳	١٥٦,٧١٥

زمان میانگین	۳,۴۴۷	۱۲,۵۲۳	۱,۰۹۲	**۹,۲۱۹**
هزینه پایگاه داده	۱۲۸,۰۰۰	۱۰۵,۰۰۰	۸۶,۰۰۰	**۱۰۵,۰۰۰**
ثانیه/ دلار	$۳۷,۱۴	$۸,۳۸	$۷۸,۷۶	**$۱۱,۳۰**

توجه کنید وقتی که برای هر پایگاه داده میانگین زمان را در نظر می گیریم، پرسشهایی که مقدار ۰ دارند در نظر گرفته نمی‌شوند. جدول (۱۴-۱۰) نشان می‌دهد که ارزانترین DB ازنظر هزینه و عملکرد DB۲ است و به دنبال آن اوراکل، Informix و سپس مایکروسافت قرار دارد.

۱۴-۶ خلاصه

بر اساس مفروضاتی که ما برای مقایسه IBM DB۲ و Microsoft SQL Server ۲۰۰۰ با Informix UDB و Oracle ۸i انجام دادیم، واضح است که بهترین آن‌ها ازنظر سطح عملکرد و هزینه IBM DB۲ است. البته، این نیز در معرض تکرار قرار دارد. اگر شما اهمیتی به مصرف حافظه روی سیستم خود یا پیکربندی سیستم خود نمی‌دهید، پس می‌توانید از سرور SQL مایکروسافت استفاده کنید برای اینکه کار با آن ساده و نتیجه بخش است. قبل از اینکه عملکرد موجود در این فصل را به دست آوریم باید بدانیم که استفاده از IBM DB۲ به مدیریت بیشتری نیاز دارد. در نهایت، عملکرد بستگی به چیزی دارد که یک نفر برای پایگاه داده برنامه ریزی می کند که می تواند عامل تصمیم گیری محسوب شود.

فصل ۱۵

تحلیل مؤلفه‌های شبکه های کامپیوتری

در فصل های قبل مفاهیم اصلی و تئوری های موجود در مدل‌سازی تحلیلی بررسی شدند. آدرس دهی در نظریه سیستم‌های صف بندی، کاربرد آن در مدل‌سازی سیستم‌های کامپیوتری و مقدمه ای بر

مدل‌سازی شبکه، مفهوم اصلی دارد. این فصل به کاربرد مدل های تحلیلی و شبیه سازی می پردازد، به ویژه از دیدگاه استفاده از ابزارهای ارزیابی عملکرد.

۱۵-۱ مقدمه

در چند سال گذشته، استفاده از مدل های عملکرد تحلیلی به جای روش های آشنا و پرکاربرد از محبوبیت زیادی برخوردار بوده است و علت آن سهولت نسبی پیاده سازی و قدرت برنامه‌های کاربردی می باشد. این مدل های تحلیلی در برآورد چنین معیارهای کارایی (مثل توان عملیاتی، میانگین طول صف و متوسط زمان پاسخگویی برای سیستم‌های واقعی) موفق بوده اند. در این فصل یک معرفی از تکنیک های صف بندی برای مدل‌سازی شبکه های ارتباطی کامپیوتر خواهیم داشت.

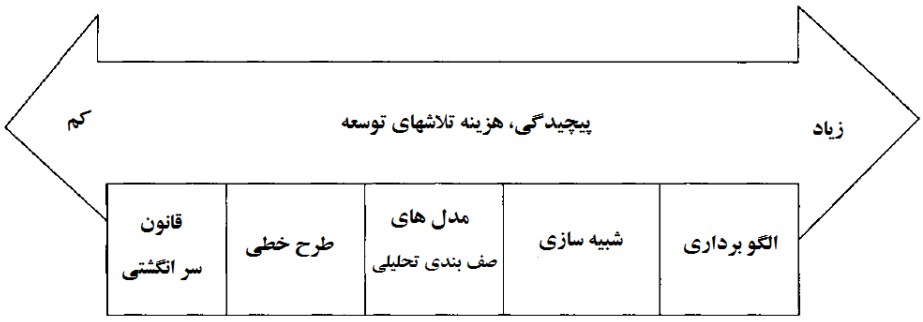
استفاده از مدل‌سازی برای توصیف و شبیه سازی یک سیستم واقعی از آغاز تکامل عصر اطلاعات با ما بوده است. این مدل ها نه‌تنها برای اندازه گیری عملکرد سیستم‌های موجود مورداستفاده قرار می گیرند بلکه به‌عنوان بخشی از طراحی و توسعه سیستم‌های جدید نیز مطرح هستند. هدف بعدی استفاده از مدل های صف بندی تحلیلی است که ما در بحث پیرامون روش های ارزیابی عملکرد شاهدیم. به جز مدل های صف بند، مهم‌ترین ابزارهای ارزیابی عملکرد (شکل ۱۵-۱) عبارتند از قاعده انگشت شست، طرح ریزی خطی، شبیه سازی و معیارگذاری. این روش ها به ترتیب افزایش پیچیدگی و دشواری پیاده سازی مرتب می شوند. قاعده شست توسط مشاهده سیستم‌های عملیاتی تعریف‌شده اند و معمولاً روی سیستم‌های محلی اعمال می گردند و می توان آن‌ها را روی سیستم‌های توزیع‌شده و شبکه ها توزیع نمود. این قوانین به اشکال زیر هستند:

۱. به طور کلی، استفاده از کانال در دستگاه های ذخیره سازی با امکان دسترسی مستقیم (DASD) نباید برای برنامه‌های کاربردی آنلاین بیش از ۳۵٪ و برای برنامه‌های گروهی بیش از ۴۰٪ باشد.

۲. کاربرد دستگاه های DASD جداگانه نباید بیش از ۳۵٪ باشند.

۳. میانگین زمان جستجو روی یک دستگاه DASD نباید بیش از ۵۰ استوانه باشد.

۴. اندازه بلوک برای ذخیره سازی کمکی نباید بیش از ۴ کیلوبایت باشد.



شکل (۱۵-۱): تکنیک های مدل‌سازی سیستم کامپیوتر

این نتایج از این نظر مفید هستند که کار کردن با آن‌ها ساده است، استفاده از آن‌ها اقتصادی است و روی عملیات روزمره قابل اعمال هستند. آن‌ها از این نظر که برای پیش‌بینی سودمندی ارتقاء نرم افزاری یا سخت‌افزاری قابل استفاده نیستند، دارای محدودیت می باشند. از روش طرح ریزی خطی می توان برای انتخاب قاعده شست در نقطه محدودیت پیش بینی استفاده نمود. اگرچه نتایجی به دست می آید اما دقت نتایج محدود به این واقعیت است که از یک طرح خطی باید برای پیش بینی رفتار سیستم‌های غیرخطی استفاده نمود. علاوه بر این، این روش نیاز به دسترسی سیستم موجود دارد تا معیار عملکرد

مربوط را بتوانیم به‌عنوان یک اصل برای طرح ریزی و برآورد نیازهای منابع آینده استفاده کنیم. برای شبیه سازی و معیارگذاری، تمایز مطلقی میان هزینه پیاده سازی و توسعه وجود ندارد. شبیه سازی به این مدل اجازه می دهد تا جزئیات بیشتری از روش های دیگر داشته باشد، اما این در مقایسه با روش های صف بندی که در آن اطلاعات زیادی وجود دارد یک مزیت محسوب نمی شود. برخی مدل های شبیه سازی آنقدر بزرگ و دست و پا گیر هستند که قابل مدل‌سازی نمی باشند. روش معیارگذاری قدیمی ترین و پرکاربردترین روش است اما معمولاً تنها برای انتخاب بهترین سخت‌افزار برای پردازش یک بار ناشناخته مفید می باشد. گفته می شود که این روش به سخت‌افزار موجود نیاز دارد و بنابراین در ارزیابی، به روز رسانی سخت‌افزار مفید نمی باشد. با توجه به نظراتی که درباره ابزارهای ارزیابی عملکرد موجود وجود دارد، می توانیم لحظه مدل های صف بندی و سودمندی کلی آن‌ها را ارزیابی کنیم. مدل های صف بندی بین طرح های خطی و شبیه سازی ازنظر هزینه و پیچیدگی پیاده سازی قرار دارند. مدل های صف بندی ممکن است ساده تر از سیستم‌هایی باشند که مدل‌سازی می شوند، برای اینکه تنها پارامترهای کارایی مرتبط تر را باید در نظر داشته باشیم. نه‌تنها مدل های صف بندی یک مکان در ارزیابی سیستم‌های موجود دارند، بلکه از آن‌ها می توان برای طراحی و توسعه سیستم‌های جدید استفاده نمود و به انتخاب سخت‌افزار کمک نمود و از تراکنش سخت‌افزار – نرم افزار برای پیشگیری از تنگنا خودداری کرد. پیشرفت هایی که اخیراً در تکنیک های مدل‌سازی تحلیلی رخ داده، باعث شدند که مدل های تحلیلی به طور فزاینده‌ای بتوانند جنبه‌های بیشتری از سیستم مدل‌سازی شده را نشان دهند. درنتیجه، این تکنیک ها رو به رشد هستند. یکی از روش هایی که در طراحی سیستم به کار می رود، تحلیل صف

بندی می باشند. مدل های صف بندی دقیق تر از تکنیک های تحلیلی دیگری هستند که عملکرد را بر اساس مقادیر میانگین پیش بینی می نمایند [۲۱]. یک دلیل این است که مدل های صف بندی امکان استفاده از جزئیات را برای توصیف سیستم ها فراهم می کنند و از این رو، ویژگی های مهم سیستم را در خود دارند. بیشتر اوقات، به چند مدل جانبی دیگر نیز نیاز داریم:

۱. مدل تراکم بار. ویژگی های تقاضای منبع را روی تجهیزات مختلف موجود در سیستم مشخص می کند.

۲. پیکربندی یا مدل ساختار سیستم. مشخصات سخت‌افزاری سیستم را نشان می دهد.

۳. مدل زمان‌بندی. الگوریتم های زمان‌بندی که در آن منابع تخصیص یافته اند را مشخص می کند.

مدل های صف بندی را می توان به‌صورت قطعی یا اکتشافی در طبیعت دسته بندی نمود. اگر پارامترهای طراحی برای مدل از طریق تجربه یا آزمایش‌های قبلی شناخته می‌شوند، می‌توان از یک تحلیل قطعی از سیستم استفاده نمود. در مقابل، اگر پارامترهای طراحی شناخته شده نباشند، معمولاً به یک تحلیل اکتشافی نیاز است که از توزیع های مختلف احتمال استفاده می کند. پارامترهای طراحی معمولی می توانند شامل موارد زیر باشند:

۱. میزان زمان بین دو ورود رویدادها

۲. دفعات سرویس‌دهی این رویدادها

۳. تعداد سرورهای مدل‌سازی شده

۴. ظرفیت سیستم (یعنی، تعداد رویدادهایی که در حال حاضر پردازش می شوند و در صف ها هستند)

۵. روش صف بندی اعمال شده (به‌عنوان‌مثال، FIFO، LIFO و غیره)

معمولاً، مدل های صف بندی برخی از ویژگی های کارایی زیر را ارائه می کنند:

۱. میانگین طول صف

۲. میانگین زمان انتظار در صف ها

۳. آمار کاربردی

۴. میانگین زمان پاسخگویی

اگرچه مدل های صف بندی ارزان هستند اما محدودیت های قابل توجهی برای این روش وجود دارد:

۱. برای اینکه این مدل ها فرض می کنند که سیستم به حالت ثابت یا متعادل رسیده است، شرایط اوج یا گذرا مدل‌سازی نشده اند.

۲. این مدل ها محدود به پیچیدگی مسئله هایی هستند که باید حل شوند. با پیچیده تر شدن مسائل یا نیاز به جزئیات بیشتر، باید برای مدل‌سازی سیستم ها از روش های دیگری استفاده شود.

۳. بدون اندازه گیری پارامترهای طراحی مختلف، به سختی می توان گفت که آیا مشخصات داده مورداستفاده می تواند سیستم تحت بررسی را نشان دهد یا خیر.

۲-۱۵ نمونه های مربوط به مدل‌سازی تحلیلی

برای درک بهتر نحوه استفاده از این تکنیک ها برای مدل‌سازی و تحلیل یک سیستم، دو بررسی را انجام خواهیم داد که یکی برای سیستم پردازش توزیع‌شده تجربی هانیول (HXDP) و دیگری برای گذرگاه

توکن. در هر دو مورد، مقادیر مشابهی موردبررسی قرار گرفتند – یعنی، میانگین زمان اسکن (زمان لازم برای کنترل برای دنبال کردن دیگری) و اندازه پیام. هدف این است که بهره وری پروتکل کنترل ویژگی های شبکه تجزیه و تحلیل شوند.

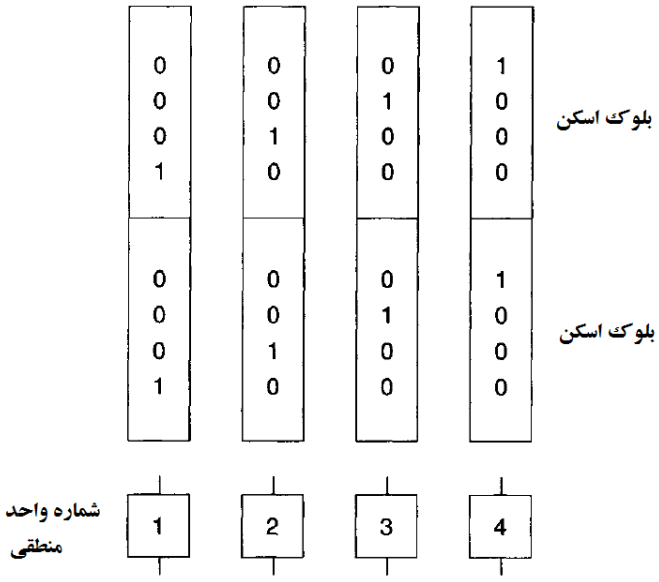
۱۵-۲-۱ مدل HXDP

مقدمه

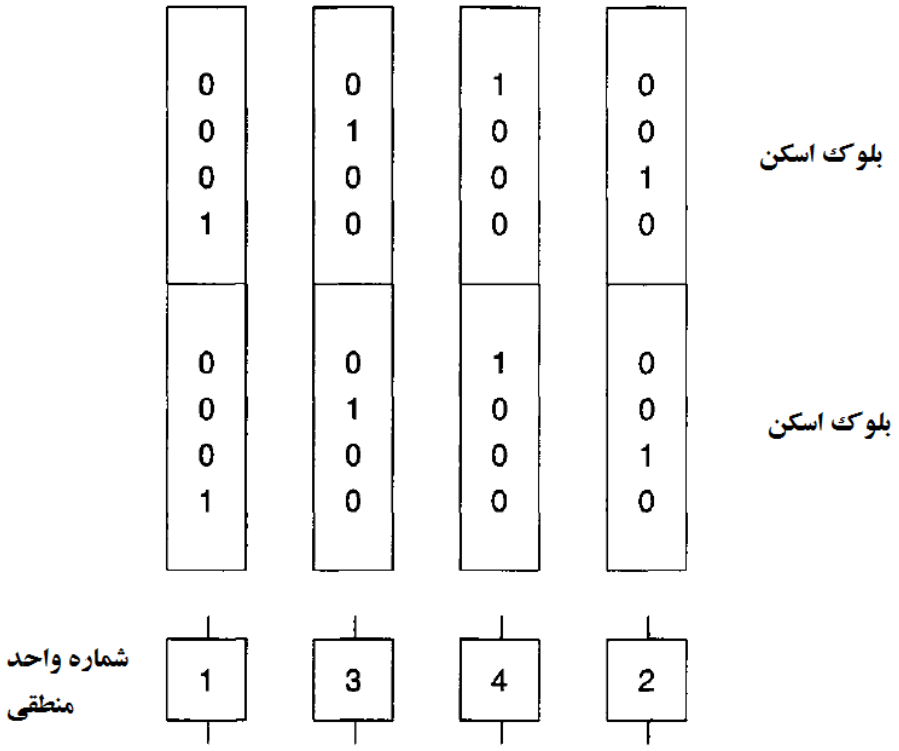
سیستم HXDP شامل پردازنده های متصل به واحدهای رابطی است که توسط یک گذرگاه عمومی با بیت های سریال به هم متصل می شوند. تخصیص گذرگاه توسط مکانیزم دسترسی بردار محور اداره می شود. قبل از مقدار دهی سیستم، بردارهای ۲۵۶ بیتی برای هر پردازنده به نحوی تنظیم می شوند که برای هر پردازنده تنها یک قطعه وجود داشته باشد، واحد رابط در شاخص خود یک مقدار ۱ دارد. تعداد برنامه‌های مختلف (ترکیبات احتمالی ۱ ها و ۰ ها) برای یک سیستم حاوی N واحد رابط است، بنابراین، ازنظر تئوری معادل $N \times N^{۲۵۶}$ می باشد که اگر $N = ۲$ باشد، مقدار زیادی خواهد بود. بااین‌حال، این طرح از همان الگو بارها و بارها عبور می کند. بااین‌حال، به جای توسعه یک مدل که هر برنامه احتمالی را میسر سازد، تصمیم گرفته می شود که برنامه‌های مجاز برای سهولت محاسبه محدود شود. درنتیجه، فرض می شود که مکانیزم برنامه به‌صورت زیر باشد: هر واحد رابط تنها یک بار در شاخص، ۱ می شود، بعد از اینکه الگوی اولیه خود را تا ۲۵۶ مین شاخص برای پردازنده N تکرار کند، این الگو را یک بلوک اسکن می نامند.

در عوض، واحدهای رابط به ترتیب شماره گذاری می شوند و به آن‌ها یک عدد واحد منطقی داده می شود.

نمایی از چنین برنامه ای با بلوک های اسکن مربوطه برای چهار IU خواهد بود (شکل ۱۵-۲). یک برنامه می تواند شامل ۲۵۶/۴ بلوک اسکن باشد. برنامه دیگر را می توان در شکل (۱۵-۳) نشان داد. ترتیب رویدادها در سیستم محدود نیز به‌صورت زیر است: بازتخصیص سیگنال در واحد منطقی ۱ وارد می شود (IU با اولین بیت ۱)، اگر پیامی منتظر دریافت سرویس باشد، واحد رابط به گذرگاه اجازه می دهد و پیام سرویس را دریافت می کند. هنگامی که پیام به مقصد رسید، یک پیام تایید به مبدا فرستاده می شود که بعد از آن یک سیگنال بازتخصیص پس فرستاده می شود، شاخص نیز به روز رسانی می شود و IU منطقی بعدی امکان دسترسی به گذرگاه را پیدا می کند. این دنباله تا زمانی که IU_N سرویس دریافت کند، ادامه پیدا می کند بعد از آن (به خاطر مفروضات) واحد منطقی ۱ سرویس‌دهی می شود. و این همچنان تا بینهایت ادامه می یابد.



شکل (۱۵-۲): بلوک های اسکن.



شکل (۱۵-۳): بلوک اسکن دیگر

در نتیجه، زمان اسکن، زمانی است که برای اسکن از ترتیب منطقی IU ها دریافت می کند، یعنی، از طریق بلوک اسکن. جدول (۱۵-۱) عبارات و توصیفات مربوط به آن‌ها را نشان می دهد.

جدول (۱۵-۱): نمادها و تعاریف آن‌ها

نماد	تعریف
N	تعداد واحدهای رابط موجود در سیستم
T_{si}	الزام زمانی برای سرویس‌دهی به یک پیام در واحد رابط i

T_{ri}	تأخیر زمانی مربوط به تخصیص دوباره سیگنالی که از واحد رابط با ترتیب منطقی i به ۱ + i عبور می کند.
λ_i	میانگین نرخ ورودی پیام در واحد رابط i
τ	زمان لازم برای اسکن از ترتیب کامل IU ها
$\overline{\tau}$	اسکن متوسط یا مورد انتظار
∂_i	با توجه به اینکه IU پیام در حال انتظار است یا خیر مقدار ۱ یا ۰ را تنظیم می کند.
T_s	زمان برای ارسال پیامی با اندازه معین از IU_i به ۱ + IU_i ارسال می شود و فاصله میان آن‌ها نیز d است.
T_{ack}	زمان برای ارسال پیام تایید از IU_i به ۱ + IU_i با فاصله d گرفته می شود.
T_R	زمان برای ارسال سیگنال بازتخصیص از IU_i به ۱ + IU با فاصله d گرفته می شود.
P	احتمال اینکه یک رابط دلخواه در حین یک اسکن به سرویس نیاز داشته باشد.

مدل‌سازی تحلیلی گذرگاه HXDP

فرض می شود که پیام ها با یک نرخ نمایی وارد شوند. احتمال اینکه یک واحد رابط اختیاری در حین اسکن به سرویس نیاز داشته باشد برابر است با:

$$P = \int_{-\infty}^{\infty} \left(۱ - e^{-\lambda t}\right) f_{\tau}(t)\, dt \qquad (۱-۱۵$$

(۱

که در آن $f_\tau(t)$ تابع احتمال تراکم زمان اسکن است. برای λt کوچک، از تقریب $e^{-\lambda t} \cong ۱ - \lambda t$ استفاده می کنیم. بنابراین:

$$P = \int_{-\infty}^{\infty} \left(۱ - e^{-\lambda t}\right) f_\tau(t)\, dt = \lambda \bar{\tau} \quad (۱۵-۲)$$

برای نسبت های موازیانس ورود پیام = λi، این ارزیابی به‌صورت زیر است:

$$P_i = \int_{-\infty}^{\infty} \lambda_i t\, f_\tau(t) dt = \lambda_i\, \bar{\tau} \quad (۱۵-۳)$$

$\nabla\, \tau_i$ که قطعه زمان اسکنی است که آن را می توان به IU_i نسبت داد، عبارت است از:

$$\nabla\, \tau_i = \xi_i\, (T_{si} + T_{ack}) + T_r \quad (۱۵-۴)$$

که در آن T_{ack} زمانی است که برای پیام ack لازم است تا به i بازگردد، اگر یک پیام دریافت شده وجود داشته باشد، T_r زمانی است که برای تخصیص دوباره سیگنال لازم است تا از شماره منطقی i به ۱ + i برود. اما ξ_i تنها می تواند مقدار ۰ یا ۱ را بگیرد و ما فرض می کنیم که یک توصیف مقصد یکنواخت و یک IU در سیستم HXDP با خود از طریق گذرگاه ارتباط برقرار می کند:

$$E\left(\xi_i\, (T_{si} + T_{ack})\right) = \frac{۱}{N} \sum_{k=۱}^{N} \left(|i - k|\, (T_s + T_{ack})\right) P_i \quad (۱۵-۵)$$

این اثبات می کند که:

$$\tau = ۱ N \sum_{i=۱}^{N} \sum_{k=۱}^{N} \left(|i-k| \left(t_s + T_{ack}\right)\right) P_I + \sum_{i=۱}^{N} |i-(i+۱)| \, T_R$$

(۶-۱۵)

که در آن $|I - k|$ نشان دهنده تعداد واحدهای رابط است که دور از واحد رابط مبدا i قرار دارد و در آن واحد رابط k قرار دارد. با جایگزینی $\lambda_i \bar{\tau} = P_i$ داریم:

$$\bar{\tau} = \frac{\tau \left(T_s + T_{ack}\right)}{N} \sum_{i=۱}^{N} (i_۲ + (۱ + N) \left(-i + ۱ / ۲\right)N)) \, \lambda_i$$

$$+ \sum_{i=۱}^{N} |i+۱| \, T_R \qquad -۱۵)$$

(۷

بنابراین:

$$\tau = \frac{\sum_{i=۱}^{N} |i+۱| \, T_R}{۱ - \left(\frac{(T_S + T_{ack})}{N}\right)\left(\sum_{k=۱}^{N} (k^۲ + (۱+N) \left(-k+۱ / ۲\right)N)) \, \lambda_k\right)} \qquad (۸-۱۵)$$

که در آن شاخص i نشان دهنده شماره منطقی واحد رابط است. مهم ترین محدودیت این مدل این است که:

$$\left((T_s + T_{ack})/N\right)\left(\sum_{k=۱}^{N} (k^۲ + (۱ + N) \left(-k + ۱ / ۲\right)N)) \right)\lambda_k \ll ۱ \quad (۹-۱۵)$$

تأثیر آن روی میانگین زمان اسکن، τ را می توان اکنون با تغییر ترکیب پارامترهای زیر تعیین نمود:

۱. تعداد واحدهای رابط، N

۲. نرخ ورودی λ_k

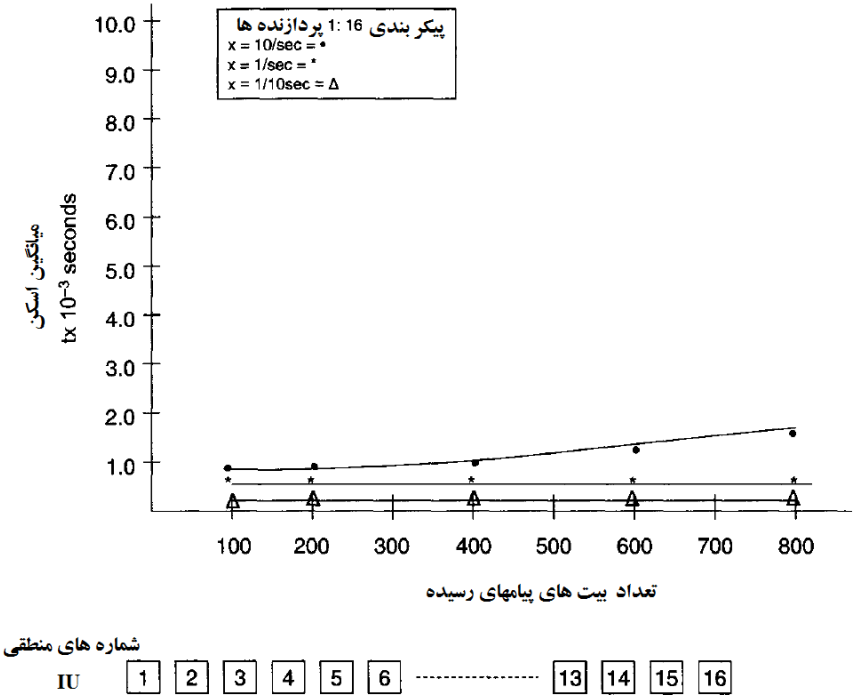
۳. شماره گذاری منطقی واحدهای رابط

۴. فواصل میان همسایگی واحدهای رابط

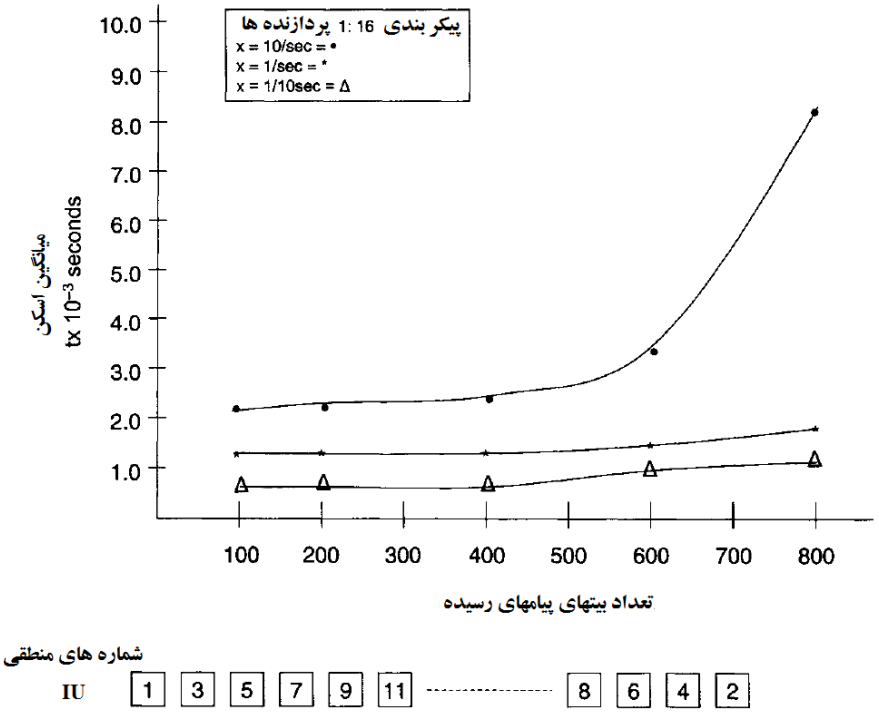
۵. میانگین اندازه پیام های ورودی

خروجی گرافیک

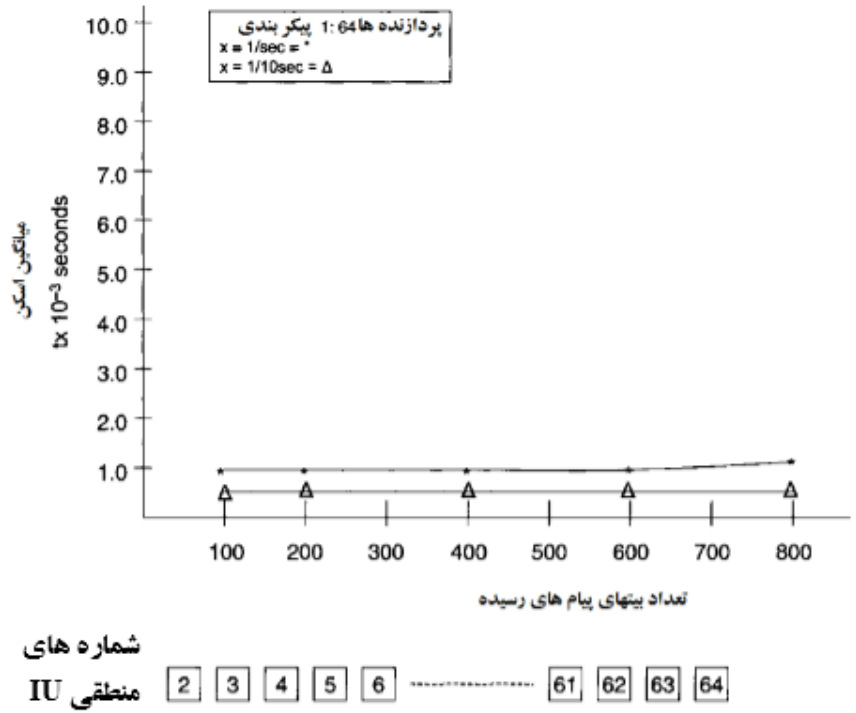
شکل (۱۵-۴) تا (۱۵-۷) محاسبات حاصل را برای زمان اسکن و اندازه پیام برای تغییرات مختلف در نرخ های ورودی، موقعیت پردازنده و اندازه آن نشان می دهند. این نتایج با نتایج شبیه سازی که بعداً در این فصل توصیف خواهیم کرد، مقایسه می شود.



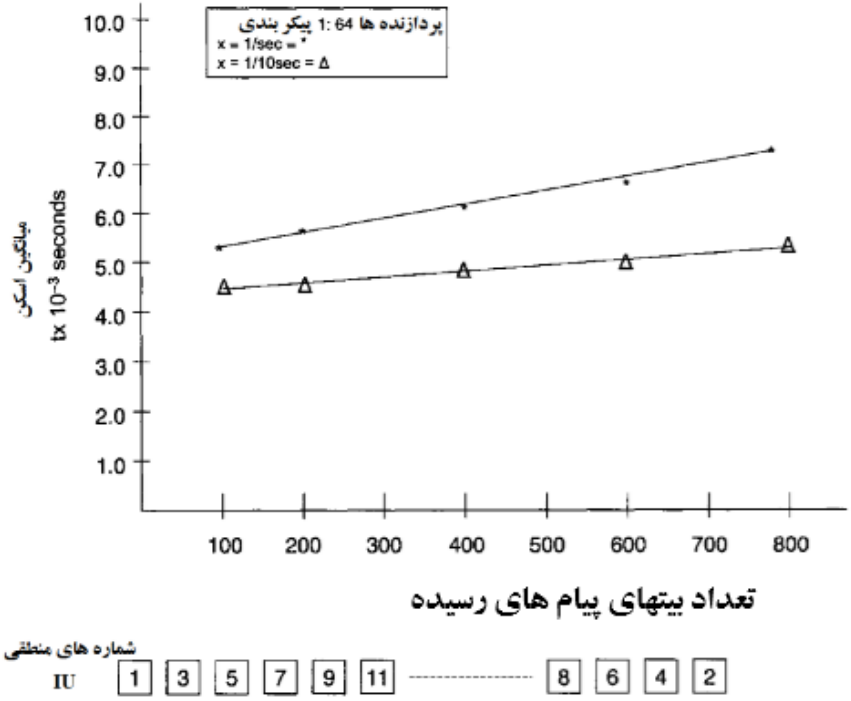
شکل (۱۵-۴): زمان اسکن نسبت به اندازه پیام، پیکربندی ۱



شکل (۱۵-۵): زمان اسکن نسبت به اندازه پیام، پیکربندی ۲



شکل (۱۵-۶): زمان اسکن نسبت به اندازه پیام، پیکربندی 1b



شکل (۱۵-۷): زمان اسکن نسبت به اندازه پیام، پیکربندی ۲b

۱۵-۲-۲ سیستم توزیع‌شده گذرگاه توکن

مقدمه

سیستم پردازش توزیع‌شده گذرگاه توکن (یک شبکه کامپیوتری محلی) شامل پردازنده های متصل به واحدهای رابط است که آن‌ها نیز به نوبه خود توسط یک رسانه ارتباطی مشترک (گذرگاه جهانی) به یکدیگر متصل هستند. تخصیص گذرگاه تحت کنترل چرخه هایی است که از توکن ها به شیوه ترتیبی از پایین‌ترین واحد رابط شماره گذاری شده (IU) به بالاترین آن‌ها

متصل می شود و سرویس می گیرد. شماره گذاری ترتیبی در حین افزایش توان تعیین می شود و بعد از رسیدن به ثبات، فرض می شود که برای اهداف مدل‌سازی به همان حالت باقی بماند. اگر یک IU به سرویس نیاز نداشته باشد، کنترل با تأخیر کمی به IU بعد می رود. برای کنترل زمانی صرف می شود تا به طور کامل از دنباله عبور کند که به آن زمان اسکن می گویند (پیش از این اشاره شد). به جای اینکه درباره کل سیستم گذرگاه توکن بررسی کنیم، به گذرگاه یا IU و لایه گذرگاه می پردازیم. مهم ترین بدنه این مثال مربوط به توسعه مدل های تحلیلی است، به ویژه، روش مدل ها برای مقدار زمان میانگین اسکن. محاسبه تحلیلی زمان اسکن به ما اجازه می دهد تا پارامترهای گذرگاه عملی و مورد نظر را به‌صورت میانگین زمان انتظار پیام، میانگین طول صف و استفاده از گذرگاه تعیین کنیم. با توجه به فرمول های به دست آمده، می توانیم درباره اثر روی پارامترهای گذرگاه تعداد پردازنده ها بررسی کنیم و مکان ترتیبی پردازنده ها را تغییر دهیم یا نسبت ورودی پیام را واریانس بخشیم.

تعاریف و فرمولاسیون مقدماتی

فرض می کنیم پیامی که وارد پردازنده می شود از توزیع پواسون پیروی می کند – به عبارت دیگر:

$$P(r,t) = \frac{(\lambda t)^R e^{-\lambda t}}{R!} \qquad (R = ۰,۱,۲,\dots) \qquad (۱۰-۱۵)$$

که در آن $P(r,t)$ احتمال این است که r پیام در زمان t وارد شود (همه پیام ها دارای طول یکسان هستند). اگر یک یا چند پیام در زمان t وارد شود به سرویس نیاز است:

$$p\left(۱ \text{ ، ورودی های بیشتر یا } ۱ , t\right) = ۱ - P(۰,t),$$

به طور معادل $= ١ - ((\lambda t)^{٠}/٠!)e^{-\lambda t} = ١ - e^{-\lambda t}$ (١١-١٥)

از آنجایی که فرض بر این است که به حالت پایدار رسیده ایم، فرض می کنیم $F\lambda(t)$ نشان دهنده تابع تراکم احتمال زمان اسکن τ باشد. بنابراین توزیع تجمعی مربوطه به‌صورت زیر خواهد بود:

$$F\lambda(t) = P(\tau \leq t);$$
$$dF_{\tau}/dt = f_{\tau}(t);$$
$$F\lambda(t) = \int_{u=-\infty}^{t} f_{\tau}(u)du$$

(١٢-١٥)

فرض می کنیم P نشان دهنده احتمال این باشد که یک پردازنده دلخواه در حین اسکن به سرویس نیاز داشته باشد:

$$P = \int_{-\infty}^{\infty}\left(١ - e^{-\lambda t}\right)f_{\tau}(t)dt \quad (١٣-١٥)$$

اگر در هر پردازنده یا اسکن بیش از یک ورودی رخ دهد، آن ورودی را می توان مسدود در نظر گرفت. این پیام سپس باید حداقل به اندازه یک زمان اسکن پیش از قرار گرفتن در گذرگاه منتظر بماند. این نشان می دهد که:

$$P\left(\text{انسداد}\right) = P\left(\text{بیش از یک ورودی در زمان اسکن}\right) \quad (١۴-١٥)$$

$$P\left(\text{بیش از یک پیام در زمان } t \text{ به اسکن نیاز دارند}\right) = ١ - [P(٠,t) + P(١,t)] = ١ - \left[e^{-\lambda t} + \lambda t e^{-\lambda t}\right] \quad (١٥-$$

(١٥

برای ارزیابی O، برای $e^{-\lambda t} \cong ۱ - \lambda t$، داریم:

$$P = \int_{-\infty}^{\infty} (۱ - e^{-\lambda t}) f_\tau(t) dt = \int_{-\infty}^{\infty} \lambda f_\tau(t) dt \qquad (۱۵-۱۶)$$

که با تعریف مقدار مورد انتظار، معادل با $\lambda\overline{\tau}$ است.

رابطه $P \approx \lambda\overline{\tau}$ برای تمام مدل های ساده سازی استفاده می شود. برای روشن تر شدن موضوع، جدول (۱۵-۲) حاوی نمادها و تعاریف مربوط به آن‌ها است که در توسعه مدل های تحلیلی مورداستفاده قرار می گیرند.

جدول (۱۵-۲): نمادها و تعاریف آن‌ها

نماد	تعاریف
N	تعداد پردازنده ها و تعداد واحدهای مربوطه به دلیل ارتباط یک به یک موجود در سیستم
T_s	زمان موردنیاز برای سرویس رسانی به یک پیام در یک واحد رابط
T_{si}	زمان موردنیاز برای سرویس رسانی به یک پیام در یک واحد رابط i
T_c	تأخیر زمانی مربوط به کنترل (توکن) که از واحد رابط به نزدیکترین واحد رابط همسایه می رود.
T_{ci}	تاخیر زمانی مربوط به عبور کنترل (توکن) از یک واحد رابط با تعداد دنباله منطقی i به واحد رابط همسایگی آن $i + ۱$
λ	میانگین نرخ ورودی پیام در واحد رابط
λ_i	میانگین نرخ ورودی پیام در واحد رابط i
τ	زمان لازم برای اسکن از طریق کل دنباله IUs
$\overline{\tau}$	میانگین یا اسکن مورد انتظار
∂_i	با توجه به اینکه آیا IU پیامی برای ارسال دارد یا خیر روی ۰ یا ۱ تنظیم می شود
d	فاصله میان واحدهای رابط i و $i + ۱$
t_s	زمانی که برای ارسال پیام با اندازه معین از IU_i به $IU_i + ۱$ با فاصله d صرف می شود

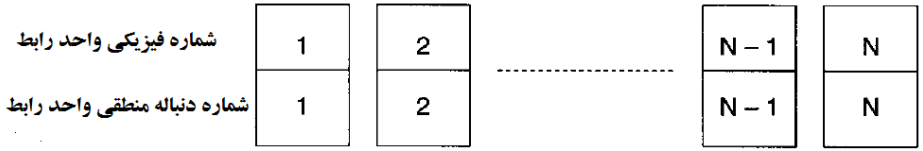
UF	عامل استفاده از گذرگاه
P	احتمال اینکه یک واحد دلخواه در حین یک اسکن به سرویس نیاز دارد.

مدل‌سازی تحلیلی گذرگاه توکن

مدل های تحلیلی که برای گذرگاه توکن ایجاد شده اند، به ترتیبی ارائه می شوند که درجه افزایش پیچیدگی و درنتیجه اثبات مفروضات ریاضی مرتبط را منعکس کنند. در هر یک از مدل ها، یک حالت ثابت، با اندازه ثابت پیام و فواصل یکسان میان پردازنده ها فرض می شود. علاوه براین، هنگامی که IU کنترل گذرگاه را در اختیار بگیرد، فرض می شود که بافر پیام برای واحد رابط بدون فاصله در گذرگاه خالی خواهد شد. مفروضات اصلی در هر مورد به طور واضح بررسی شده اند.

مورد ۱

در مدل تحلیلی اصلی، فرض می کنیم که نرخ ورودی پیام ها در هر واحد رابط N برابر باشد و با k نشان داده شود. علاوه بر این، فرض می کنیم که به حالت پایدار رسیده باشیم، شماره گذاری مرتب (منطقی) واحدهای رابط با شماره گذاری فیزیکی یکسان است (شماره گذاری از چپ به راست)، که در شکل (۱۵–۸) نشان دادهشده است.



شکل (۱۵-۸): شماره گذاری منطقی و فیزیکی واحدهای رابط

اگر T_c نشان دهنده تاخیر مربوط به توکن (کنترل) عبوری از یک واحد رابط به دیگری باشد، T_{ci} برای هر واحد رابط نیز می تواند همان باشد، برای اینکه فرض می کنیم که پردازنده ها با فاصله یکسان از یکدیگر قرار دارند و درنتیجه، کنترل نیز از همان فاصله از یک پردازنده به پردازنده همسایه بعدی خواهد پیمود (با بالاترین شماره همسایگی بعدی).

پارامتر زمانی اساسی بعدی، زمانی است که برای سرویس‌دهی یک پیام برای واحد رابط نیاز است. در یک توپولوژی حلقوی با طرح عبور توکن می توانیم T_s را در نظر بگیریم که برابر است با زمان موردنیاز برای سرویس رسانی به یک پردازنده تا به طور میانگین برای تمام پردازنده ها همان مقدار لحاظ گردد. مرجع [۲۲] نشان می دهد که برای توپولوژی گذرگاه نیز نمی توان به همین نتیجه رسید. زمان لازم برای سرویس رسانی به یک پیام، تابعی از مقصد IU است. بنابراین، قرار دادن مبدا IU در توپولوژی گذرگاه روی میانگین زمانی که برای سرویس رسانی به یکی از پیام های آن لازم است تأثیر می گذارد. برای مثال، اگر $N = ۳$ باشد، پیکربندی آن را در شکل (۱۵-۹) ملاحظه خواهید کرد. برای مثال، واحد ۱ برای انتشار یک پیام به پردازنده ۲، پیام باید فاصله را از ۱ به ۲ بپیماید، برای مثال واحد ۱ برای ارسال یک پیام به ۳، باید فاصله را از ۱ تا ۳ بپیماید. اگر فاصله معادل بین دو واحد رابط همسایگی را با d نشان

دهیم و فرض کنیم که همه IU ها به‌صورت بالقوه مشابه مقاصد پیام باشند (یعنی، یک توزیع یکنواخت برای مقاصد پیام را فرض می کنیم) داریم:

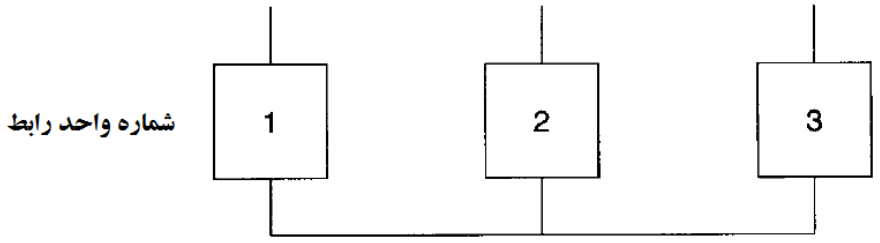
$$E((T_s|source = ۱) = {}^{۱}/_{۲}\Big((d/velocity\ estimate) + {}^{۱}/_{۲}\,(۲d/velocity\ estimate)\Big) \quad (۱۵-۱۷)$$

که در آن $d/velocity$ زمان را برابر t_s برآورد می کند تا پیامی با اندازه منتخب را از i به $i + ۱$ ارسال نماید:

$$= {}^{۱}/_{۲}\,t_s + t_s = {}^{۳}/_{۲}\,t_s \quad (۱۵-۱۸)$$

برای پردازنده ۲، همانند پردازنده مبدا، معادله مربوطه به‌صورت زیر خواهد بود:

$$E((T_s|source = ۲) = {}^{۱}/_{۲}\,t_s + {}^{۱}/_{۲}\,t_s = t_s \quad (۱۵-۱۹)$$



شکل (۱۵-۹): گذرگاه توکن با سه واحد رابط

فرض می کنیم که T_{si} نشان دهنده زمان لازم برای سرویس رسانی به یک پیام در واحد رابط i باشد. این زمان برای عبور توکن از i امین IU به ۱ + i امین واحد رابط باشد که آن را می توان به‌صورت زیر بیان کرد:

$$\nabla\tau_i = \xi_i T_{si} + T_c$$

$$\begin{cases} ۱, if & \text{اگر } IU \text{ یک پیام در حال انتظار برای پردازش داشته باشد} \\ ۰, & \text{در غیر اینصورت} \end{cases} \qquad (۲۰-۱۵)$$

زمان کل اسکن به‌صورت زیر است:

$$\tau = \sum_{i=۱}^{N} \nabla\tau_i \; or \; \tau = \sum_{i=۱}^{N} E(\xi_i T_{si} + T_c) \qquad (۲۱-۱۵)$$

در حال حاضر، هر دو طرف معادله (۱۵-۲۱) را در نظر می گیریم و میانگین مقدار زمان اسکن $\overline{\tau}$ را به‌صورت زیر به دست می آوریم:

$$\overline{\tau} = \sum_{i=۱}^{N} E(\xi_i T_{si}) + \sum_{i=۱}^{N} E(T_c) = \sum_{i=۱}^{N} E(\xi_i T_{si}) + NT_c \qquad (۲۲-۱۵)$$

سپس با تعریف مقدار مورد انتظار حاصل:

$$E(\xi_i T_{si}) = \sum_{\xi_i} \sum_{T_{si}} \xi_i T_{si} \, P\,(\xi_i T_{si}) \qquad (۲۳-۱۵)$$

همانطور که پیش از این گفتیم، T_{si} تابعی از فاصله ای است که پیام باید بپیماید و از آنجایی که ξ_i تنها می تواند دو مقدار ۰ و ۱ داشته باشد:

$$E(\xi_i T_{si}) = \frac{۱}{N-۱} \sum_{k=۱}^{N} |i-k| t_s P \qquad (۲۴-۱۵)$$

که در آن k،i نشان دهنده تعداد واحدهای رابط دور از واحد رابط مبداi است، واحد رابط مقصد k پیدا می شود و هر واحد رابط دیگر به جز i این احتمال را دارد که واحد رابط مقصد باشد. یعنی، یک احتمال ۱ − ۱/N.

P احتمال به دست آمده در فرمولاسیون مقدماتی است. به طور خلاصه:

$$\overline{\tau} = \frac{۱}{N-۱}\sum_{i=۱}^{N}\sum_{k=۱}^{N}|i-k|t_sP + NT_c$$

$$= \frac{t_sP}{N-۱}\sum_{i=۱}^{N}\sum_{k=۱}^{N}|i-k|NT_c \qquad (۲۵-۱۵)$$

$$= \frac{t_sP}{N-۱}\left(\frac{N^۳-N}{۳}\right) + NT_c$$

$$= t_sP\big(N(N+۱)\big)/۳ + NT_c$$

با قرار دادن $P \cong \lambda_i\overline{\tau}$ در (۱۵-۲۵) داریم:

$$\overline{\tau} = t_s\lambda\overline{\tau}\big(N(N+۱)\big)/۳ + NT_c$$

$$\overline{\tau} = \Big(۱ - \big(t_s\lambda\big(N(N+۱)\big)/۳\big)\Big) + NT_c \qquad (۲۶-۱۵)$$

$$\overline{\tau} = (NT_c)/(۱ - \lambda(N(N+۱)t_s)/۳)$$

این معادله بر اساس فرض و تقریب های مربوطه معتبر است، اگر و تنها اگر:

$$(\lambda(N(N+۱)t_s)/۳) \ll ۱ \qquad (۲۷-۱۵)$$

مورد دوم:

در این مدل، ما مفروضاتی که تمام واحدهای رابط دارای نرخ ورودی پیام λ هستند را کنار می گذاریم و به نرخ ورودی پیام λ_i اجازه می دهیم تا واحد رابط i داشته باشند. بااین‌حال، فرض می کنیم پیکربندی فیزیکی یا منطقی باشد که به نوبه خود در مورد مربوطه حذف خواهد شد. رها سازی مفروضات به شیوه ای انجام می شود که روی ماهیت تکامل توسعه مدل های تحلیلی تأکید داشته باشد. رهاسازی نرخ ورودی پیام باعث افزایش قابلیت و درنتیجه، آزمایش می شود اما به طور طبیعی فرمول نهایی برای τ را پیچیده می کند. از آنجایی که هر واحد رابط دارای نرخ ورودی پیام مشخصی است λ_i، برای مثال واحد رابط i، P به‌صورت زیر خواهد بود:

$$P_i = \int_{-\infty}^{\infty}\left(1 - e^{-\lambda_i t}\right) f_\tau(t)dt \qquad (۲۸-۱۵)$$

$$P_i \cong \lambda_i \overline{\tau}$$

معادله (۱۵-۲۱) نیز کاربردی است – یعنی:

$$\tau = \sum_{i=2}^{N} \nabla\tau_i = \sum_{i=1}^{N}(\xi_i T_{si} + T_c) \qquad (۲۹-۱۵)$$

علاوه بر این، τ نیز به‌صورت زیر است:

$$\tau = \sum_{i=1}^{N} E(\xi_i T_{si}) + NT_c \qquad (۳۰-۱۵)$$

که در آن:

$$E(\xi_i T_{si}) = \left(\frac{1}{N-1}\right)\sum_{k=1}^{N}|i-k|t_s P_i \qquad (۳۱-۱۵)$$

$$\bar{\tau} = \sum_{i=1}^{N}\left(\frac{1}{N-1}\right)\sum_{i=1}^{N}(|i-k|t_s P_i)NT_c \qquad (۳۲-۱۵)$$

$$\tau = \left(\frac{t_S}{N-١}\right)\sum_{i=١}^{N}\left(i^{٢} + (١+N)\left(-i + \left((١/٢)N\right)\right)P_i\right) + NT_c \qquad (٣٣-١٥)$$

با جایگزینی $\lambda_i\bar{\tau} \cong P_i$

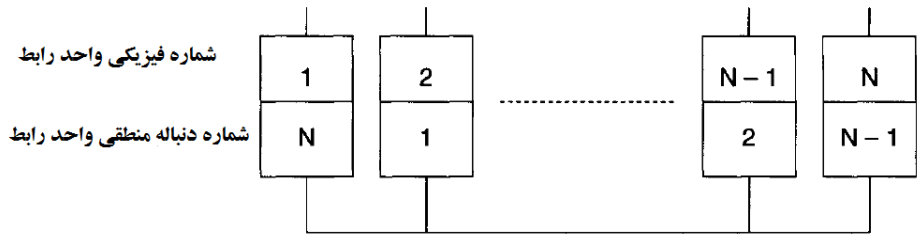
$$\bar{\tau} = \left(\frac{\tau t_S}{N-١}\right)\sum_{i=١}^{N}\left(i^{٢} + (١+N)\left(-i + \left((١/٢)N\right)\right)\lambda_i\right) + NT_c \qquad (٣٤-١٥)$$

بنابراین،

$$\bar{\tau} = \frac{NT_C}{١-\left(\frac{t_S}{N-١}\right)\left(\sum_{k=١}^{N}(i^{٢}+(١+N)(-i+(١/٢)N)\lambda_i)\right)} \qquad (٣٥-١٥)$$

مورد سوم

این مدل اصلاحات زیادی را در بر می گیرد که به این مدل اجازه می دهد سیستم واقعی را بهتر منعکس سازد. به ویژه، فرض می شود که به حالت پایدار رسیده باشیم، شماره گذاری منطقی نباید موقعیت فیزیکی را منعکس کند، بلکه در واقع پیکربندی حالت پایدار در شکل (۱۵-۱۰) نشان داده‌شده است. شماره گذاری دنباله منطقی سیستم گذرگاه توکن ممکن است فرض کند که هر احتمالی از $N!$ میسر باشد، انتخاب های مختلفی از حالت پایدار با یک احتمال معادل.



شکل (۱۵-۱۰): نگاشت منطقی به موقعیت فیزیکی

بنابراین، خیلی مهم است تا مدلی ایجاد کنیم که بتواند تمام $N!$ حالات ترکیب را منعکس سازد. درنتیجه، T_c یک ثابت نیست بلکه از برخی لحاظ زمان لازم برای گردش توکن از واحد رابط i به واحد رابط $i+1$ را منعکس می کند.

فرض کنید i نشان دهنده تعداد متوالی از واحد رابط باشد، درنتیجه، $\nabla\tau_i = \xi_i T_{si} + T_{ci}$ که در آن T_{ci} زمان سپری شده برای یک توکن برای طی فاصله از IU با دنباله منطقی i به IU با دنباله منطقی $i+1$ است ($N+1$ تبدیل به IU_1 می شود). در مدل های قبل، شماره دنباله منطقی و شماره فیزیکی واحدهای رابط یکسان بودند، بنابراین، نیازی نبود که به صراحت ارتباط شاخص i را بیان کنیم. زمان کل اسکن نیز به‌صورت زیر بیان می شود:

$$\tau = \sum_{i=1}^{N} \nabla\tau_i \ \ or\ \tau = \sum_{i=1}^{N} E(\xi_i T_{si}) + T_{ci} \qquad (۱۵-۳۶)$$

مقدار میانگین زمان اسکن برابر است با:

$$\overline{\tau} = \sum_{i=1}^{N} E(\xi_i T_{si}) + \sum_{i=1}^{N} E(T_{ci}) \qquad (۱۵-۳۷)$$

که برای یک پیکربندی شناخته شده برابر است با:

$$\overline{\tau} = \sum_{i=1}^{N} E(\xi_i T_{si}) + \sum_{i=1}^{N} |i-(i+1)|\, t_c \qquad (۱۵-۳۸)$$

$N+1$ به دلیل گردش مقدار ۱ را نشان می دهد که در آن t_c زمانی است که برای کنترل صرف می کند تا از هر واحد رابط k به واحد فیزیکی $k+1$ برود. پیش از این محاسبه کردیم که:

$$\tau = \sum_{i=1}^{N} E(\xi_i T_{si}) \qquad (۳۹-۱۵)$$

و برای بهره مند شدن از نتایج، شاخص k شماره فیزیکی واحد رابط را نشان می دهد، درحالی که شاخص i شماره منطقی آن را نشان می دهد. با ترکیب نتایج و جایگزین کردن عبارات داریم:

$$\lambda_k \overline{\tau} = P_k \qquad (۴۰-۱۵)$$

که به رابطه زیر می رسیم:

$$\bar{\tau} = \frac{\sum_{i=1}^{N} |i-(i+1)| t_c}{1-\left(\frac{t_s}{N-1}\right)\left(\sum_{k=1}^{N}(k^2+(1+N)(-k+(1/2)N)\lambda_k)\right)} \qquad (۴۱-۱۵)$$

با توجه به معادله (۱۵-۴۱)، تأثیر روی میانگین زمان اسکن را می توان با تغییر ترکیب متغیرهای زیر تعیین نمود:

۱. تعداد واحدهای رابط، N

۲. نرخ ورودی، λ_k پیام ها در واحدهای رابط با شماره های منطقی k.

۳. تغییر شماره گذاری ترتیبی منطقی واحدهای رابط

۴. تغییر فواصل میان واحدهای رابط همسایه.

۵. تغییر میانگین اندازه پیام های ورودی به واحدهای رابط

$$\left(\frac{t_s}{N-1}\right)\left(\sum_{k=1}^{N}\left(k^2+(1+N)\left(-k+\left((1/2)N\right)\lambda_k\right)\ll 1\right)\right) \qquad (۴۲-۱۵)$$

۳-۱۵مدل‌سازی شبیه سازی شبکه های منطقه محلی

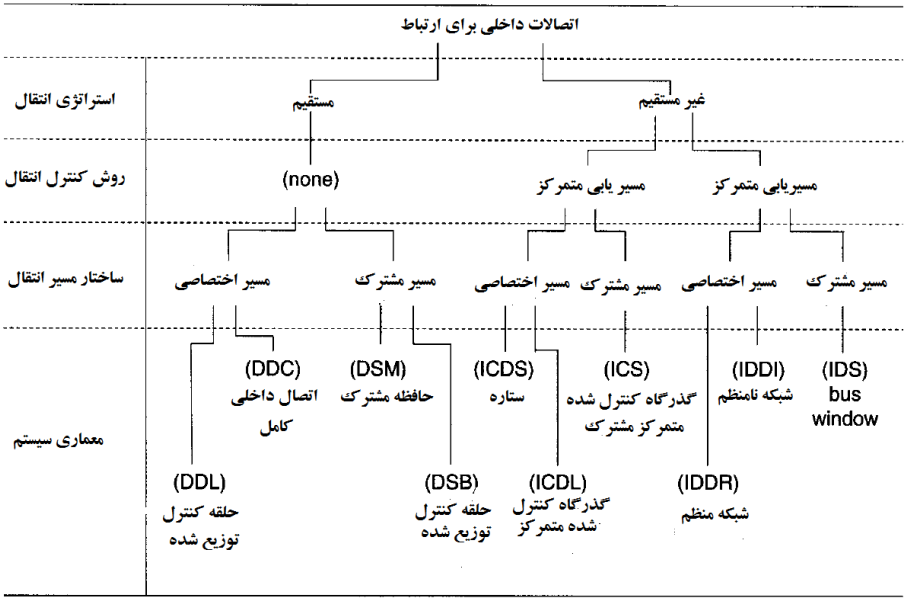
۱۵-۳-۱ شبکه های کامپیوتری (مدل)

یک شبکه کامپیوتری می تواند هر گونه اتصال عناصر محاسباتی (سیستم ها، پایانه ها و غیره) توسط امکانات ارتباطی باشد که در نهایت ارتباط فراشبکه ای و درون شبکه ای ایجاد می شود. این شبکه ها در داخل سازمان می تواند به‌صورت دو پردازنده با یک حافظه مشترک باشد که کامپیوترهای نسبتاً مستقل زیادی با فاصله جغرافیایی زیاد به آن‌ها متصل اند. عناصر محاسباتی نیز می تواند شامل شبکه ها باشد که در آن می توان سیستم‌های نامحدودی از شبکه ها را تصور کرد. ویژگی های اصلی یک شبکه که معماری آن را متمایز می کند شامل توپولوژی یا سازمان، ترکیب، اندازه، نوع کانال و استراتژی کاربردی و مکانیزم کنترل است.

با استفاده از نام‌گذاری و طبقه بندی که در ساختارهای به هم متصل کامپیوتری مورد بحث قرار گرفت، یک سیستم خاص را می توان توسط استراتژی انتقالی (مستقیم یا غیرمستقیم)، مکانیزم کنترل انتقال (متمرکز یا غیرمتمرکز) و ساختار مسیر انتقال آن (اختصاصی یا مشترک) شناسایی نمود. توپولوژی های شبکه ای مختلف مثل حلقوی خطی و ستاره ای را می توان به‌عنوان بدنه ای از ترکیبات منحصربه فرد این ویژگی ها در نظر گرفت (شکل۱۵-۱۱).

ترکیب شبکه می تواند با توجه به تشابه گره ها یا عناصر به هم متصل محاسباتی، همگن یا غیرهمگن باشد. اندازه شبکه معمولاً اشاره به تعداد گره یا عناصر محاسباتی دارد. با توجه به کانال های ارتباطی آن، یک شبکه می تواند همگن باشد یا ممکن است از رسانه های زیادی استفاده نماید. کنترل یا مدیریت

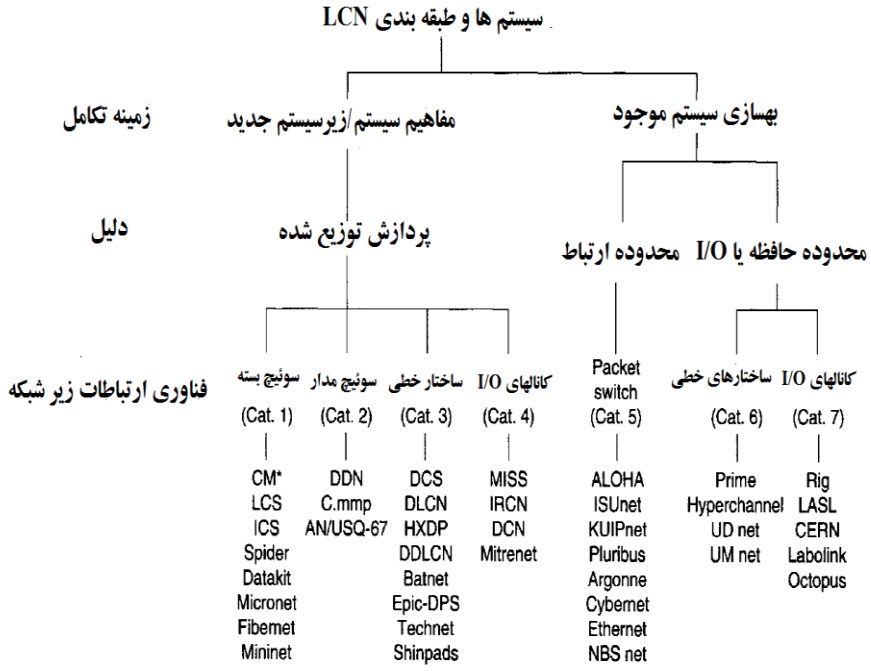
کلی شبکه معمولاً به شدت متمرکز است یا به طور کامل توزیع‌شده می باشد. اگر سخت‌افزار مورداستفاده برای عبور خط کنترل از یک دستگاه به دستگاه دیگر به شدت در یک لحظه متمرکز شده باشد، به آن کنترل متمرکز می گویند.



شکل (۱۵-۱۱): طبقه بندی ساختار اتصال داخلی کامپیوتر

محل سخت‌افزار می تواند در داخل یکی از دستگاه های متصل به شبکه باشد و یا می تواند یک واحد سخت‌افزاری مجزا باشد. اگر منطق کنترل به طور گسترده در دستگاه های مختلف متصل به شبکه توزیع‌شده باشد، به آن کنترل غیرمتمرکز می گویند. مشکلات مستقل از پیاده‌سازی که وابسته به ویژگی های سیستم هستند عبارتند از مدولاریته، انعطاف پذیری اتصال، تأثیر خطا، پیکربندی دوباره خطا، تنگنا و پیچیدگی منطقی. یکی از زیرمجموعه های سیستم‌های کامپیوتری احتمالی، شبکه های کامپیوتری

محلی هستند (LCNs). اگرچه هیچ تعریف استانداردی از این عبارت وجود ندارد، اما به طور کلی یک LCN را می توان به‌صورت شبکه ای ساختاریافته برای ترکیب منابع شبکه های راه دور و برای انجام عملیات چند پردازشی موازی در نظر گرفت. یک معیار معتبر برای ایجاد یک شبکه به‌عنوان یک LCN این است که فواصل بین گره آن در محدوده ۰٫۱ تا ۱۰ کیلومتری با نرخ انتقال ۱۰۰Mbps باشد.



شکل (۱۵-۱۲): طبقه بندی شبکه های کامپیوتری محلی

LCN با ساختار خطی

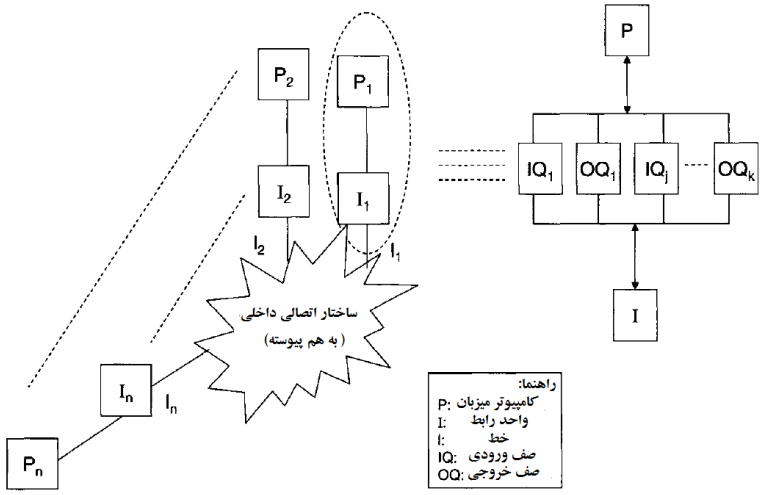
محدوده سیستم‌هایی که باید بررسی شوند را به گروه سوم سیستم‌های دارای ساختار خطی تعریف می‌کنیم (شکل۱۵-۱۲). برخلاف فناوری های سیستم یا رسانه نقطه‌به‌نقطه مثل مدارها و سوییچینگ مدار، یک سیستم با ساختار خطی شامل مجموعه ای از خطوط مشترک است که از آن‌ها در هر زمان می توان تنها به‌عنوان یک واحد استفاده نمود. این نشان می دهد که برای پیشگیری از درگیری های مربوط به گذرگاه باید از طرح هایی با کنترل خطی استفاده کنیم.

مولفه های شبکه

به‌عنوان اولین مرحله در ایجاد یک شبیه سازی کلی LCN، یک مدل شبکه ایجاد کردیم که در شکل (۱۵-۱۳) نشان داده‌شده است. یک شبکه شامل تعداد دلخواهی از گره های شبکه با اتصالات داخلی است. هر گره شامل یک یا چند عنصر محاسباتی میزبان یا پردازنده است که به یک پردازنده مستقل (IU) متصل می شود.

میزبانان عبارتند از تولیدکنندگان و مصرف کنندگان تمام پیام ها و سیستم‌های مستقل، پایانه ها، دروازه به شبکه های دیگر و نمونه های دیگری از عناصر محاسباتی را شامل می شوند. IU ها عملکرد تمام بخش ها و گره های شبکه مثل اداره پیام، کنترل جریان و پیکربندی سیستم را اداره می کنند. این خطوط رسانه پخش فیزیکی را نشان می دهند که گره ها را به یکدیگر متصل می کنند. IU ها همراه با ساختار اتصال داخلی زیرشبکه ارتباطی را تشکیل می دهند. مدل شکل (۱۳-۱۵) واحدهای اصلی سخت‌افزاری موجود برای انتقال اطلاعات میان پردازش های میزبانان مختلف را از یکدیگر جدا می کند. در این

سطح، میان نمونه پیام هایی مثل بلوک های داده و پیام های تایید تمایزی وجود ندارد. برای توسعه و تصفیه این مدل، عناصر اصلی، ساختمان‌ها و فعالیت ها باید بیشتر تعریف شوند.



شکل (۱۵-۱۳): شبکه کامپیوتری توزیع‌شده تعمیم یافته

پردازنده های میزبان

مؤلفه های میزبان یا پردازنده معمولاً شامل عناصر محاسباتی و کنترل، سطوح مختلف حافظه و اجزای خروجی ورودی هستند. در مورد یک سیستم، رفتار هر پردازنده می تواند منعکس کننده توابع توزیع مناسبی باشند که نسبت تولید و مصرف پیام های درون پردازنده را نشان می دهند. این توابع توان پردازش ذاتی یک پردازنده و بارگذاری مبتنی بر پارامترهای پردازنده، سطوح ارتباطی بیرونی و ارتباطات درون شبکه ای را منعکس می‌کنند.

صف ها

صف ها ساختمان‌های بافر حافظه مشترکی هستند که از طریق آن‌ها اطلاعات بین یک پردازنده و مکانی که IU رخ می دهد، انتقال داده می شوند. برای هر گره یک صف (خط) خروجی برای انتشار پیام در حال انتظار وجود خواهد داشت و این یک یا چند صف (پیام) ورودی بیشتر حاوی پذیرش‌های پردازش نشده هستند. منطقه حافظه صف را می توان در پردازنده یا با توجه به پیاده سازی در IU پیدا نمود. در عمل هر دو یکسان هستند. موارد مرتبط با صف ها، متغیرهای کنترل هستند که توسط پردازنده ها و IU ها حفظ و نظارت می‌شوند تا امکان دسترسی همزمان را برای صف ها فراهم نمایند. انواع رایج صف ها، خطی، دایره ای و به هم متصل هستند. صف های خطی (بافرها) زمانی مورداستفاده قرار می گیرند که اندازه پیام مشخص نباشد و ساختمان بافر را بتوان به‌صورت گسترده تخصیص داد. استفاده از بافرهای دایره ای زمانی مناسب است که پیام های متعددی با طول های تعیین نشده پیش از اینکه یکی از آن‌ها پردازش شوند، در بافر قرار گیرند. زمانی از مخزنی از صف استفاده می شود که اندازه پیام و زمان ورودی در محدوده گسترده ای متفاوت باشد و قابل پیش بینی نیز نباشند و پیام ها به ترتیب ورود خود نیز حذف نشده باشند.

پیام ها با استفاده از استراتژی های مختلفی مثل FIFO، LIFO و پردازش طولانی ترین پیام در ابتدا، در حالت برداشت (خواندن) سپرده (نوشته) می شوند. دسترسی به صف برای پیشگیری از نوشتن در یک صف کامل، خواندن از یک صف خالی و خواندن اطلاعاتی که نوشته شده اند، کنترل می شود.

واحدهای رابط

تا آنجایی که به نقش آن در شبکه مربوط می شود، واحد رابط ازنظر سخت‌افزاری و نرم افزاری پیچیده ترین رابط است. عملکرد اصلی IU این است که پردازنده را برای برقراری ارتباط با دیگران در شبکه فعال کند و این عملکرد کلی شبکه را بهبود بخشد. این کار شامل مقداردهی اولیه سیستم، کنترل جریان، تشخیص خطا و مدیریت می شود.

هنگامی که IU متوجه شود که پردازنده آن یک پیام آماده ارسال دارد، پیام را برای انتشار فرمت می کند و برای استفاده انحصاری از کانال های ارتباطی مدعی می شود. با توجه به تخصیص کنترل، کنترل کننده پیام را با توجه به پیاده سازی منتشر می کند و ممکن است از پردازنده مقصد انتظار پاسخ داشته باشد. با توجه به تکمیل بهره برداری از منبع (گذرگاه)، IU باید بتواند کنترل را با توجه به طرح تخصیص به گزینه بعدی بفرستد. اگر خطای IU داشته باشیم، IU های دیگر باید بتوانند تا جایی که به مسئولیت پذیری کنترل شبکه مربوط است، با آن جایگزین شوند.

خطوط ارتباطی

این خطوط ارتباطات فیزیکی میان گره های شبکه هستند که به وسیله آن‌ها انتشار پیام ها صورت می گیرد. اصطلاح هم ارز خطوط ارتباطی، کانال و مدار هستند. یک مدار خاص توسط تکنیک های آنالوگ یا دیجیتال به‌صورت تک جهت یا دو جهت هستند. مدارها با استفاده از انواع رسانه مثل کابل های کواکسیال، جفت های به هم تابیده، فیبر نوری، اتصال مایکروویو، پیوند لیزری و غیره حمایت می شوند. برای شبیه سازی نیازی نیست که به مشخصات سطح پایین فکر کنیم به

جز مواردی که توسط مجموعه ای از مشخصات کانال نشان داده می شوند: حداکثر نرخ داده، پارامترهای تأخیر و خطا و محدودیت های جهتی.

اگر یک مدار نقطه به نقطه همیشه به یک شبکه تخصیص داده نشود، بهتر از مشخصات راه اندازی نیز در نظر گرفته شود. مشخصات راه اندازی می تواند شامل مکانیزم سیگنال رسانی و تاخیر، تاخیر راه اندازی مدار و تاخیر شکست مدار نیز باشد. در سیستم‌هایی که بعداً در همین فصل موردبررسی قرار می گیرد، مشخصات راه اندازی یک عامل محسوب نمی شود. حداکثر سرعت داده از ۵۰Kbps (جفت به هم تابیده) تا بیش از ۱۵۰Mbps (کابل های نوری) است. این نرخ قابلیت انتشار خط را نشان می دهد و همانند نرخ شبکه ای نیست که در آن اطلاعات منتقل می شود. همیشه یک سربار وجود دارد. عوامل مختلفی مثل خطاهای منطقی، رابط های الکترونیکی و آسیب فیزیکی برای انتشار آسیب به وجود می آیند که از خطاهای تک بیت تا خطاهای خطی را شامل می شود. با توجه به نوع خط مورداستفاده، نرخ خطای معمول بین ۱۰۰۰۰ تا ۱۰۰۰۰۰۰۰ بیت است.

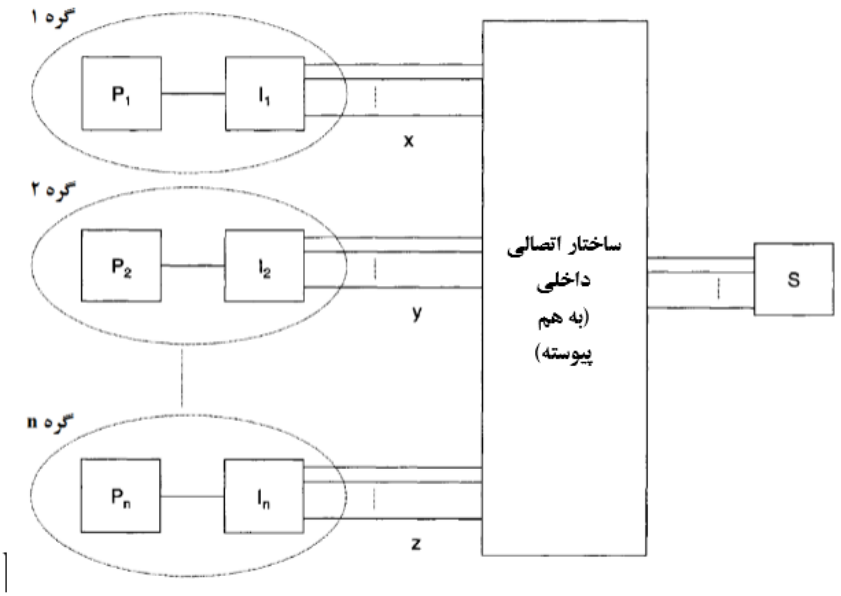
ساختارهای به هم متصل

جنبه های مختلف شبکه، مثل زمان‌بندی، مسیریابی پیام و پیکربندی مربوط به ساختار اتصال فیزیکی شبکه هستند. برای مثال، واجد شرایط بودن برای کنترل گذرگاه می تواند وابسته به لحظه باشد، زمان لازم برای انتقال یک پیام نیز می تواند وابسته به مکان پردازنده های درگیر باشد و یا قابلیت تداوم کار شبکه ممکن است به دلیل وجود لینک های زیاد در آن باشد. این ساختار را می

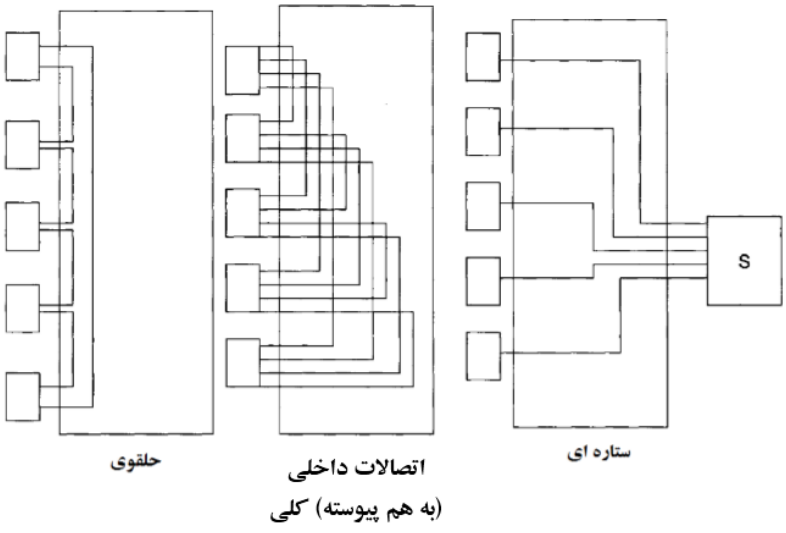
توان توسط یک سازمان منظم از سه سخت‌افزار (گره، مسیرها و سوییچ ها) نشان داد که در انتقال اطلاعات میان فرآیندها در گره های مختلف درگیر هستند.

این انتقال ها را انتقال پیام می گویند و میان نمونه های پیام متمایز نیستند، مثل بلوک های داده، درخواست های سرویس، تیرهای راهنما و غیره. به این ترتیب، در مورد مشکلات ساختاری می توان گفت که نیازی نیست میان یک عنصر محاسباتی و رابط آن تمایزی ایجاد نمود. این عناصر به‌صورت یک گره به همدیگر فشرده شده اند. عناصر سوییچینگ روی مسیریابی یا مقصد تأثیر زیادی دارند.

شکل (۱۵-۱۴) یک مدل کلی از یک سیستم به هم پیوسته (سیستم اتصالات داخلی) را نشان می دهد. برای سادگی، کلاس سیستم‌هایی با یک سوییچ تنها نشان داده‌شده است.



شکل (۱۵-۱۴): مدل به هم پیوسته کلی (سیستم اتصالات داخلی)



شکل (۱۵-۱۵): مثال هایی از ساختارهای به هم پیوسته (ساختارهایی با اتصالات داخلی)

در ارتباط با هر گره و سوییچ، چند مسیر یا لینک وجود دارد. هر گره می تواند از طریق یک، دو یا چند لینک به بقیه شبکه، به یک سیستم گذرگاه، یک ساختار حلقوی یا دایره ای یا یک شبکه کاملاً به هم پیوسته که لینک های مستقیمی میان هر جفت گره وجود دارد، متصل شود. شکل (۱۵-۱۵) مثال های خاصی از ساختارهای به هم متصل (ساختارهایی با اتصالات داخلی) را نشان می دهد.

این نمودارها پیشنهاد می کنند که ساختمان به هم پیوسته را می توان با الگوریتم ها و یا جدول ها ارائه نمود که همین امر امکان تعیین چنین چیزهایی را به‌عنوان گره واجد شرایط بعدی برای استفاده از منبع، طول های درون گره ای، پیکربندی دوباره پارامترها و مسیرهای بهینه میسر می سازد. ممکن است که در شبیه سازی فعلی نمایش کاملی تحت ساخت باشد اما در آینده پیشرفت بیشتری خواهد داشت.

اگر گره در مؤلفه های خود تحلیل شود (یعنی، عنصر محاسباتی و IU)، مشاهده می کنیم که این مدل می تواند جنبه های اتصال طرح های کنترلی مختلف را برای شبکه های توزیع‌شده ارائه دهد. از آنجایی که هیچ فرضی درباره ماهیت مولفه های یک گره و یا ارتباطات آن با بقیه شبکه وجود ندارد، یک گره خاص به جای یک پردازنده میزبان در حالت معقول، می تواند یک گره کنترل کننده متمرکز را به مدیریت شبکه ارائه دهد.

۱۵-۳-۲ پروتکل ها

فعالیت های شبکه در یک محیط خصمانه رخ می دهند برای اینکه این مولفه ها به‌صورت مولفه های ناهمگن با پهنای باند محدود، تاخیر و انتشارهای غیرقابل اطمینانی هستند که برای منابع با یکدیگر رقابت می کنند. برای ایجاد هماهنگی مرتب و کنترل فعالیت ها، تراکنش‌ها یا پروتکل های ارتباطی ایجاد شده اند که ویژگی های الکتریکی، مکانیکی و کاربردی شبکه ها را شامل می شوند. این پروتکل ها معمولاً همیشه پیچیده هستند و ساختارهای چندلایه ای مربوط به ساختار فیزیکی و کاربردی شبکه ها را دارند. هر لایه پایین تر، ازنظر کاربردی مستقل از کل لایه های سطح بالاتر می شود. بااین‌حال، برای کارکرد، لایه های سطح بالاتر به عملیات صحیح سطوح پایین بستگی دارند.

هر زمان که یک پروتکل به وسیله یک پروتکل در یک سطح پایین تر ارتباط برقرار می کند، پروتکل سطح پایین تمام اطلاعات کنترلی پروتکل سطح بالاتر را تایید می کند و سپس با توجه به آن اطلاعات چند عملیات را انجام می دهد. در بیشتر موارد، پروتکل سطح پایین تر، تمام اطلاعات و داده‌های کنترلی

را می گیرد و با آن‌ها به‌عنوان داده رفتار می کند و اطلاعات کنترلی مخصوص به خود را به آن‌ها اضافه می کند. این در قالب پیام هایی است که از یک شبکه عبور می کنند که مفهوم سلسله مراتبی یک پروتکل خود را نشان می دهد. قالب پیام های منتشر شده طرح بندی توابع را نشان می دهد، مثل پرانتز در یک عبارت ریاضی یا کاری که بند زبان برنامه نویسی انجام می دهد.

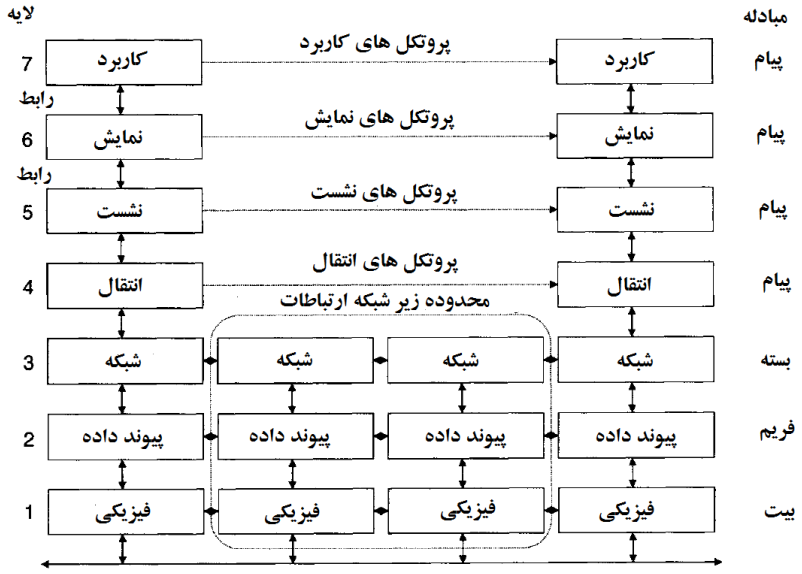
در میان توابعی که توسط پروتکل ها ایجاد شده اند، ایجاد و نگهداری مدار، مدیریت منبع، کنترل پیام و تشخیص و اصلاح خطا نیز دیده می شود. عملکرد توابعی که توسط پروتکل ها ایجاد می شوند عبارتند از ایجاد و نگهداری مدار، مدیریت منبع، کنترل پیام و تشخیص و اصلاح خطا. عملکرد این توابع نشان می دهد که در انتشار داده تاخیر وجود دارد و باید سربرگ و فیلدهای داده به پیام ها افزوده شوند و این پیام تایید برای پذیرش یا بازنشر پیام ها در صورت بروز خطا باید فرستاده شود. این باعث کاهش نرخ داده‌های مفید در یک شبکه می شود. این جنبه های سربار انتشار تبدیل پیام را باید در هنگام ارزیابی بهره وری پروتکل ها در نظر داشت. به طور کلی، یک پروتکل مجموعه‌ای قرارداد است که برای اداره مبادله اطلاعات میان عناصر محاسباتی تایید می‌شوند. اگرچه این عناصر می توانند مدارها، مودم ها، نهایی ها، میزبان‌ها، پردازنده ها یا افراد باشند، اما ما در این بخش تنها به جنبه میزبان و پردازنده تعبیه شده در تجهیزات دیگر می پردازیم.

معمای حفظ یک محیط توزیع‌شده مناسب بستگی به پذیرش عدم اطمینان ذاتی مکانیزم پیام برای طراحی فرآیندها و مقابله با آن دارد. در سیستم‌های قبل، پروتکل هایی در طرح ادهاک طراحی شده بود. معمولاً این پروتکل ها مخصوص برنامه‌های کاربردی بوده و به این صورت پیاده سازی می شوند. تمام کارهای

پروتکل اخیر در جهت ساختار چندلایه و سلسله مراتبی با پیاده سازی جزئیات هر لایه در تمام لایه ها و سلسله مراتب های دیگر حرکت کرده اند.

اگرچه هیچ توافق جهانی روی نام و تعداد لایه های پروتکل وجود ندارد، اما یک استاندارد تایید شده به نام ISO (سازمان ملی استاندارد)، مدل اتصال سیستم باز (OSI) وجود دارد که در شکل (۱۵-۱۶) نشان داده‌شده است. با استفاده از این سازمان، پروتکل های سطح ۱ (لایه فیزیکی) شامل استانداردهای RS-۲۳۲ و کنترل خطی X.۲۱، رمزنگاری منچستر ۲، رمزگذاری، نظارت و کنترل زمان استفاده از لینک، کنترل نرخ انتشار و هماهنگ‌سازی هستند.

سطح ۲ (لینک داده) برای مبادله قابل اطمینان داده میان گره های متصل به یک لینک داده فیزیکی فراهم شده است. عملکردها شامل تأمین شفافیت داده (به‌عنوان‌مثال تأمین ابزارهایی برای برقراری تمایز میان بیت های کنترل و داده در یک انتشار)، نظارت وضوح درگیری، ایجاد، نگهداری و خاتمه دادن به تراکنش‌ها (تراکنش‌ها)، تشخیص و اصلاح خطا و بازیابی خطای گره می شوند.



شکل (۱۶-۱۵): مدل ISO OSI.

یک توصیف از جنبه های عملیاتی شبکه عمومی به خوبی در زمینه ساختار پروتکل که پیش از این تعریف‌شده بود ارائه شده، برای اینکه تمام رویدادها و فعالیت های احتمالی شبکه، عمدی و غیرعمد باید تحت این ساختار مدیریت شوند. ساختار این پروتکل نیز ساختارهای اصلی و مکانیزم های کاربردی حامی عملیات شبکه را بیان می کند. قبل از اینکه کنترل یا ارتباطات داده ای انجام شوند، ابزارهای اصلی سیگنال رسانی و انتشار بیت در یک رسانه فیزیکی باید ایجاد گردند. لینک های فیزیکی باید همراه با توپولوژی مخصوص شبکه و پارامترهای خطی ایجاد گردند. غالباً روی این سطح از یک طرح کدگذاری مثل منچستر۲ برای هماهنگ‌سازی و تشخیص خطا استفاده می‌شد. در این طرح، هر یک از بیت های اصلی داده به دو بیت انتشار تبدیل می شوند به نحوی که نمی توان در پیام کدگذاری شده سه

بیت متوالی یکسان پیدا کرد. این نشان می دهد که گیرنده پیام می تواند با تماشای این رخداد، متوجه بروز خطاها شود. این، این کدگذاری را می توان به‌صورت انتخابی غیرفعال نمود تا شکل موج نامعتبر و منحصربه فردی ایجاد نمود که از آن به‌عنوان سیگنال های هماهنگ‌سازی استفاده می‌شود.

با توجه به قابلیت سرویس‌دهی لایه فیزیکی برای مبادله سیگنال در رسانه فیزیکی، از لایه پیوند داده برای تبادل دنباله منطقی از پیام ها در سراسر لینک فیزیکی استفاده می‌شود. قابلیت های اساسی این لایه شامل تأمین شفافیت داده، کنترل پیام، مدیریت خط و کنترل خطا می باشد. از آنجایی که در مورد اصلی، داده و اطلاعات کنترل از همان مسیر در حین انتشار عبور می کنند، برای برقراری تمایز میان این دو باید از تکنیک های خاصی استفاده شود. این کار با تخصیص ابزارهای کنترل به برخی الگوهای خاص بیتی انجام می گیرد که مانع بروز جریان داده از طریق استفاده از چنین تکنیک هایی به‌عنوان بیت و بایت و طرح یا رویه منچستر می شود. در این روش، دنباله های کنترل می توانند آغاز و پایان پیام هایی با طول متغیر که به‌صورت غیرهمزمان منتشر می‌شوند را محدود سازند. عبارات مصطلح برای این دنباله ها عبارتند از BOM، EOM و پرچم. واحد ابتدایی انتشار داده معمولاً حرف است. تعداد حروف داده در یک پیام معمولاً بیش از حداکثر طول پیام (MML) متغیر است و یک بیت توازن معمولاً به داده و حروف کنترل افزوده می شود. هر زمان که فرستنده و گیرنده به‌صورت مستقیم به هم متصل نباشند، هر پیام باید شامل آدرس دهی به اطلاعات شود. آدرس ها باید فیزیکی باشند، در هر مورد هر گره یک آدرس منحصر به فرد دارد یا ممکن است این آدرس منطقی باشد، که در این صورت هر گره با توجه به آن یک یا چند دنباله کدگذاری شده دارد که نهادهای کاربردی را نشان می دهند. یک

آدرس منطقی خاص می تواند مربوط به تعداد دلخواهی از گره های فیزیکی باشد، بنابراین برای یک یا چند انتشار به کار می رود. اطلاعات آدرس ممکن است در بخش داده یک پیام اعمال شوند یا می تواند بخشی از اطلاعات کنترل محسوب شود. هر تراکنش با توجه به اینکه روند ارسال پاسخ از مقصد نیاز به موفقیت انتشار دارد، را می توان یک طرفه یا دو طرفه در نظر گرفت. در سیستم‌هایی که در آن‌ها می توان میان این روش ها یک انتخاب انجام داد، پیام باید حاوی اطلاعاتی پیرامون این انتخاب باشد. انواع پاسخ‌ها را می توان دید اما محدود به موارد زیر نیستند:

- NO REPLY REQUESTED، اگر یک پیام به روندی که در آن نسخه های متعددی وجود دارد ارسال شود، تضمین ارسال پیام تایید مطلوب نیست برای اینکه در این حالت امکان بروز برخورد زیاد است.
- STATUS REQUESTED، اطلاعات مربوط به موفقیت یا خطای انتشار درخواست می شود.
- LOOPBACK REQUESTED، حلقه بازگشتی حالتی است که در آن یک گره مقصد می تواند گره مبدا نیز باشد.

۱۵-۳-۳ تشخیص خطای انتشار

برای حصول اطمینان از بدون خطا بودن انتشار، باید در سطح کنترل لینک مجموعه‌ای از تراکنش‌ها میان فرستنده و گیرنده وجود داشته باشد تا بروز خطا را تشخیص و اصلاح کنند. روش‌های احتمالی زیادی برای کنترل خطا در یک لینک انتشار وجود دارد. دو نوع کلی کنترل خطا عبارتند از کنترل

خطای ارسال و کنترل خطای بازخوردی. عملی ترین و شایع ترین روش، کنترل بازخوردی است. ساده‌ترین شکل تشخیص یک بررسی توازن روی هر کاراکتر انتقالی است. به این اغلب بررسی فراوانی عمودی گفته می شود که از آن برای حفاظت در مقابل خطاهای بیتی موجود در بین کاراکترها استفاده می شود. بررسی فراوانی افقی یا عمودی (LRC) در کل یک پیام انجام می شود. این کار با محاسبه یک بیت توازن برای هر لحظه بیت تمام کاراکترهای موجود در پیام انجام می شود. قدرتمندترین شکل بررسی، بررسی کد فراوانی چرخه ای (CRC) است که فرآیند جبری منسجم تری می باشد و می تواند تعداد انبوهی از بیت های حامل خطا را شناسایی نماید. این احتمال وجود دارد که یک پیام یا پاسخ به مقصد خود نرسد و ربطی به خوب یا بد بودن اطلاعات نیز ندارد. علت آن می تواند یک خطای فیزیکی باشد، مثل خطای لینک یا گره مقصد، یا یک خطای منطقی مثل استفاده از نام غلط مقصد. این احتمالات را می توان با انجام یک مکانیزم توقف پیدا نمود که باعث می شود اگر پیام بعد از تایید ناشی از تأخیر (وقفه پاسخ) و عدم دریافت پیام تایید، دوباره منتشر شود. بسیاری از سیستم ها به تنهایی وابسته به قرارداد وقفه و تایید مثبت هستند و از پیام تایید منفی استفاده نمی کنند. هنگامی که چند دستگاه از یک گذرگاه استفاده می کنند، حتما باید روشی وجود داشته باشد که توسط آن یک واحد خاص بتواند کنترل گذرگاه را به دست آورد و امکان انتشار داده را پیدا کند. مشکل اصلی این حوزه، اجتناب از درگیری درخواست ها در حین استفاده از طرح های برنامه ریزی و داوری است به نحوی که تنها یک واحد در هر موقعیت می تواند گذرگاه را به دست آورد. مکانیزم ها باید برای مقداردهی مجدد تأمین گردند و در

صورت راه اندازی سیستم، افزایش و حذف گره، خطاهای خط و انتشار امواج مزاحم در سیستم مجهز شوند.

در تمام سیستم‌ها هنگامی که بیش از یک انتشار داده یا کنترل به طور همزمان رخ می‌دهد، تصادم یا برخورد ایجاد می شود. این می تواند به علت استفاده از چند تکنیک تصادفی برای تولید توالی پیام با چند مقصد باشد که به‌صورت ناقص درخواست تاییدیه دارند. تصادم ها یا برخوردها معمولاً توسط حذف دائمی یا موقتی گره های درگیر یا انتشار دوباره پیام های مجاز کنترل می شوند.

محدودیت معمولاً در زمان استفاده از خط اعمال می گردد تا از تداخل گره در گذرگاه به‌صورت عمد یا خطای گره ای پیشگیری شود. این حالت معمولاً با اعمال یک محدودیت روی حداکثر اندازه پیام و یا نظارت استفاده از خط انجام می شود تا تعیین شود که چه زمانی یک گره فراتر از نیاز به ارسال بزرگ‌ترین پیام مجاز، حالت انتشار فعال دارد. این قابلیت نظارت با استفاده از زمان سنج loud-mouth به دست آمده است که با توجه به تخصیص گره فعال می شود و اگر اجازه اجرا داشته باشد یک سیگنال وقفه (یا یک تصادم) ایجاد می کند. خروجی معمول حذف برق از مدار منتشرکننده (به‌صورت موقت یا دائم) و تشکیل پردازنده میزبان در صورت امکان است. هنگامی که از کنترل صحبت می کنیم، بهتر است به یاد داشته باشید که این مورد معمولاً مربوط به دستگاه یا لحظه فیزیکی خاصی نمی باشد اما یک نهاد کاربردی است که در سراسر شبکه توزیع (تکثیر) شده است.

۱۵-۳-٤ رویدادها

برای شبیه سازی دقیق رفتار عملی شبکه ها باید از یک روش تحلیلی رسمی تر و کمی تر استفاده نمود. برای این کار، باید مفاهیم زیر را تعریف کنیم. تمام کارها و فعالیت ها، عمدی و غیره، که در یک سیستم رخ می دهند را می توان به‌صورت رویدادهایی طبقه بندی نمود. یک رویداد یعنی هر رخدادی صرف نظر از دوره آن. رویدادها چند ویژگی دارند که به‌صورت زیر هستند:

- یک رویداد دارای آغاز، پایان و یک دوره زمانی است.
- یک رویداد می تواند ساده یا پیچیده باشد. یک رویداد ساده، رویدادی است که برای شبیه سازی نیازی به تجزیه شدن به رخدادهای متوالی ساده تر ندارد. در مقابل، یک رویداد پیچیده رویدادی است که از رویدادهای ساده تشکیل می شود.
- یک رویداد می تواند رخدادی تصادفی باشد و یا ماهیتی تصادفی داشته باشد یا ممکن است نتیجه قطعی (اثر) یک علت قابل اکتشاف باشد.
- یک رویداد الگوی خاصی از رخداد دارد (به‌عنوان‌مثال، الگوی دوره ای، غیر دوره ای، همزمان و غیرهمزمان و غیره).
- رویدادها به کلاس تعلق دارند. اهمیت کلاس یک رویداد این است که هر عضو دارای همان اثر روی دیگری در یک زمینه خاص است. برای مثال، در سیستم‌های قطعی، ایجاد یک پیام توسط یک صدا معادل با اثر آن روی مشخصات نادرست یک نام مقصد می باشد – که هر

دو باعث بروز وقفه در پاسخ و باز نشر می شوند. بنابراین، این دو رویداد در یک کلاس قرار دارند.

- رویدادها ممکن است همزمان یا منفصل (متوالی) باشند. رویدادهایی که در یک زمان تداخل دارند، همزمان هستند، در غیر این صورت منفصل اند. مفهوم همزمانی را در اینجا معرفی می کنیم. ساختار برنامه متوالی در صورتی همزمان در نظر گرفته می شود که بتواند با موفقیت ارائه شود و یا رویدادهایی که واقعا همزمان هستند را مدل‌سازی کند.

یک مثال می‌تواند عمل درخواست سرویس از پردازنده‌ها باشد. در حالی‌که این مثال درباره یک رویداد غیرهمزمان است، رویدادهای غیرقابل پیش بینی که توسط یک گره خاص از کانال استفاده می کنند، این دو جنبه از سیستم را می توان از یکدیگر جدا کرد، برای اینکه (به جز مورد وقفه ها) درخواست تا زمان تکمیل بهره وری پاسخی دریافت نمی کند. تا زمانی که یک رکورد از دوره استفاده وجود داشته باشد، درست قبل از تفکیک پذیری درگیری توسط ماژول کنترل برنامه شبیه سازی یک سابقه مؤثر از درخواست های گره را می توان ایجاد کرد. در ادامه فهرستی از رویدادهای اصلی معرفی می شوند که برای بروز عملیات LCN های مختلف قابل یافتن هستند. تمام رفتار سیستم را در نهایت می توان به دنباله ای از این رویدادهای ساده تقلیل داد. رویدادها تحت مولفه ای که رخ می دهد، فهرست شده اند.

پردازنده

- تولید. این مربوط به تولید یک پیام توسط یک عنصر محاسباتی میزبان است. پارامترهای مربوط به این رویداد عبارتند از زمان تولید، اندازه پیام و مقصد(ها).

- بررسی صف. تعیین توسط پردازنده وضعیت یک صف ورودی یا خروجی پیش از خواندن یا نوشتن آن.
- وضعیت پیام خروجی. با توجه به استراتژی دسترسی به بافر، یک پیام تولید شده را می توان به حالت نرمال موردبررسی قرار داد، این پیام ممکن است در داده بررسی شده موجود نوشته شود، این پیام ممکن است توسط پیام تا زمان بررسی نگه داشته شود و یا کنار گذاشته شود.
- مصرف (خواندن پیام). این مربوط به خواندن یک پیام در صف ورودی توسط عضو محاسباتی میزبان است. اگر یک پیام در صف ورودی موجود باشد، بدون درنگ قابل مصرف خواهد بود. مصرف با توجه به یک صف، مشابه تولید است.
- گره. بهتر است که رویدادهای پی در پی به جای عضو محاسباتی یا IU به گره نهاد منسوب شوند.
- جمع. هنگامی یک گره به یک مدار افزوده می شود که شبکه از ادغام آن به سیستم مطلع گردد. این ممکن است به دلیل افزایش توان اولیه گره یا بازیابی خطا رخ دهد.
- ادغام. این زمانی است که گره واقعا تبدیل به یک بخش عملی از شبکه گردد.
- خطا. خطای یک گره به‌عنوان عضو کاربردی شبکه در نظر گرفته می شود.

واحد رابط

- بررسی صف. این مشابه رویداد پردازنده است.
- خواندن پیام. IU پیام را از یک صف خروجی به دست می آورد.

- پیش پردازش پیام. این مربوط به فرمت یا بسته بندی یک پیام برای انتقال است. این عمل باید از فرمتی که توسط پردازنده انجام می شود، تفکیک شود.
- درخواست. IU به شبکه اطلاع می دهد که قصد دارد از گذرگاه استفاده کند. این می تواند شامل انتشار سیگنال کنترل باشد یا نباشد.
- اتصال. این به معنی کسب واقعی کانال برای بهره برداری است.
- انتشار(مجدد). این زمانی رخ می دهد که اولین حرف از یک پیام (در صورت وجود زنجیره ای از پیام) مثلاً در یک ساختار حلقوی اولین حرف از اولین پیام، در گذرگاه قرار گیرد.
- فعال سازی وقفه پاسخ. در سیستم‌هایی که به پاسخ انتشار پیام در مقدار زمان خاصی نیاز است، در حین انتشار در برخی نقاط یک زمان سنج برای پاسخگویی فعال می شود.
- تشخیص (شناسایی). این یک شناخت توسط IU توسط پیامی است که به آن آدرس دهی می شود.
- پذیرش. این حالتی است که در آن پیام کاملی دریافت می شود، پردازش روی آن انجام می گیرد و آماده صف بندی می گردد. علاوه بر این به این زمان، زمان بررسی صف یا زمان انتشار پاسخ نیز می گویند.
- نوشتن پیام. این زمانی رخ می دهد که پیام دریافتی در صف ورودی قرار گیرد.
- پذیرش پاسخ. این زمانی است که مبدا پیام اطلاعات را از مقصد بدون توجه به انتشار دریافت می کند.

- حذف پیام. یعنی حذف پیام از یک صف خروجی ناشی از یک انتشار موفق.
- رها کردن. با توجه به اتمام استفاده، سیگنال های IU که در آن بازتخصیص رخ می دهد، رها می شوند.

با استفاده از این مفاهیم، فعالیت کلی سیستم یا جریان را می توان توسط دنباله رویدادهای پیچیده زیر نشان داد:

فعالیت گره ← کنترل/داوری ← استفاده

این ساختار ساده به این دلیل وجود دارد که مفهوم همزمانی مؤثر در مورد سیستم‌های گذرگاه عمومی معتبر است.

فعالیت گره

فعالیت گره رفتار تمام گره ها واحدهای رابط را در طول دوره استفاده از یک گره خاص شبیه سازی می کند. در طول این دوره فعالیت گره ای شامل موارد زیر است:

- تولید و مصرف پیام ها
- فعالیت صف
- پردازش ریشه IU
- حذف و اضافه گره ها از شبکه

کنترل

کنترل ممکن است با توجه به شرایطی که تحت آن کنترل ایجاد می شود، تعدادی دنباله احتمالی باشد.

استفاده

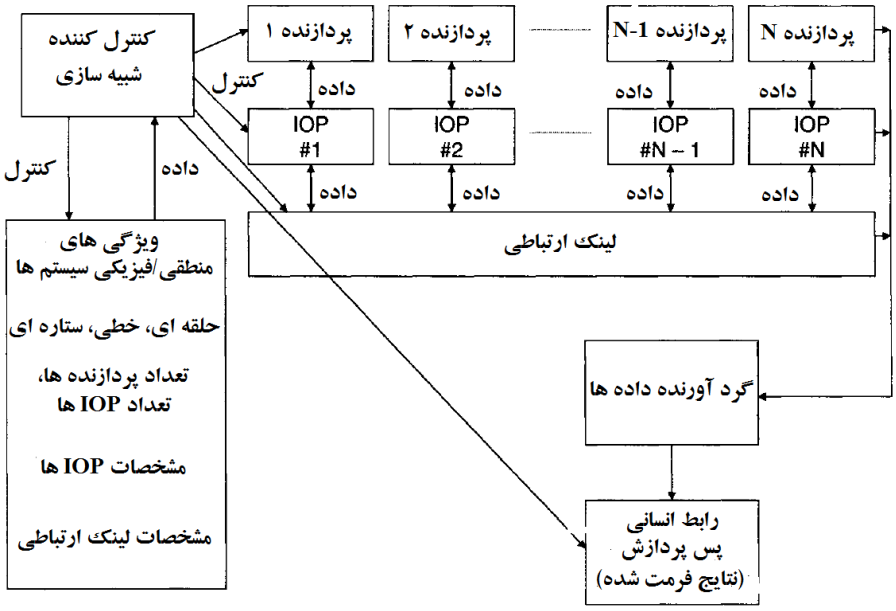
رویداد استفاده شامل تمام فعالیت های مربوط به استفاده از یک گره ذر گذرگاه برای انتشار یک پیام باشد. استفاده با اتصال به گذرگاه شروع می شود و به آرامی نیز خاتمه می یابد، در صورت موفق بودن تراکنش در پی یک خروجی کنترل که در نتیجه تناقض یک پروتکل با مداخله کنترل انجام می شود (یعنی پیام خیلی طولانی است، هیچ سیگنال تخصیص یافته ای منتشر نمی شود و ...).

۱۵-۳-۵ ساختمان مدل شبیه ساز LAN

بر اساس بحث های قبلی درباره رویدادها و ساختمان LAN، می توانیم بگوییم که یک LAN از سه کلاس سخت‌افزاری اصلی تشکیل می شود: دستگاه های میزبان، واحدهای رابط و لینک های ارتباطی. علاوه بر این، این کلاس های سخت‌افزاری سطوح مختلف نرم افزاری و کاربردی نیز دارند. سطح لینک مربوط به مدیریت و عملکرد انتقال های فیزیکی سطح بیت می باشد. این شامل زمان بندی و کنترل و این رابط مکانیکی و الکتریکی می باشد. واحدهای رابط سرویس های اصلی را برای ایجاد رسانه ارتباطی و پروتکل هایی برای یک شبکه واقعی ارائه می دهند. این مولفه و سرویس های آن باید برای پروتکل های گره به گره، میزبان به گره و انتها به انتها

فراهم شوند. این روند شامل تشخیص و اصلاح خطا، کسب و کنترل رسانه، مسیریابی، کنترل جریان، فرمت پیام، شفافیت شبکه، نگهداری از اتصالات و سرویس های دیگر می شود. کلاس میزبان دستگاه برای LAN سایت هایی ایجاد می کند که از سوی میزبانان دیگر نیاز به سرویس های راه دور دارد. سرویس هایی که در این سطح ارائه می شوند از نوع رابط میزبان به گره، پروتکل های میزبان به میزبان و پروتکل های اشتراک منبع هستند. پیاده سازی در این سطح از LAN باعث می شود که کاربر قادر به مشاهده آن ه باشد مثل سیستم عامل توزیع‌شده، یک سیستم مدیریت پایگاه داده توزیع‌شده، سرویس های ایمیل و غیره. مدلی از این ساختار، حداقل یک مولفه برای هر یک از این اقلام را نشان می دهد. بنابراین، شبیه ساز باید مولفه هایی با عمومیت و انعطاف پذیری کافی برای مدل‌سازی این مولفه ها و ایجاد تجزیه و تحلیل داشته باشد.

شکل (۱۵-۱۷) این مولفه های اصلی و مولفه های شبیه سازی لازم برای اندازه گیری عملکرد و عملیات یک شبیه سازی را نشان می دهد. با کمک این ساختار، مشاهده می شود که شبیه ساز شامل یک ساختمان مدولار با مولفه هایی است که برای مدل‌سازی اختلافات جزئی LAN قابل تبدیل هستند.



شکل (۱۵-۱۷): مولفه های اصلی یک مدل شبیه سازی

ویژگی های فیزیکی و منطقی عجیب و غریب هر طراحی سیستم در روال های نرم افزاری مستقل و یا جدول های داده گنجانده شده اند. طراحی سطح بالای مدل LAN نیاز به توابع زیر را به ما نشان می دهد:

- یک کنترل کننده شبیه سازی که مسئول هماهنگی و عملیات به موقع باقیمانده ماژول های نرم افزاری است. این قابلیت معماری سیستم و تکنیک های محاسبات توزیع‌شده را همراه با داده ورودی کاربر مقدار دهی می کند، رویدادهای را برنامه ریزی می نماید، حالت سیستم را محاسبه می کند، جدول های زمانی رایج را نگه می دارد و پردازش روال های نرم افزاری دیگر که توسط تنظیمات منطقی و فیزیکی مخصوصی توصیه می شوند را آغاز می نماید.

- روال پردازنده یک سیستم که قادر به شبیه سازی فعالیت وابسته به زمان (داده ورودی، داده خروجی و زمان پردازش) هر حالت محاسباتی پیشنهادی است.
- یک روال پردازش رابط که می تواند فعالیت (تاخیر زمان، اداره پیام، تعیین اولویت، تکنیک آدرس دهی، تخصیص منبع و غیره) هر پردازنده پشتیبان که توسط ویژگی های منطقی و فیزیکی خاصی موردنیاز است را شبیه سازی کند.
- یک روال لینک ارتباطی که می تواند تاخیرات زمانی ویژگی های انتقال داده و کنترل رسانه انتقالی پیشنهادی را شبیه سازی نماید.
- یک روال گردآوری داده که مسئول جمع آوری، فرمت و تلفیق پارامترهای ارزیابی سیستم مورد درخواست خواهد بود.

اقلام داده جمع آوری شده شامل حداقل، حداکثر و میانگین موارد زیر است:

- زمان برای انتشار پیام از A به B.
- زمان انتظار پیام
- تعداد پیام ها در صف یا سیستم
- اندازه پیام
- استفاده از گذرگاه
- زمان‌بندی واحد رابط (که پیش از این ارائه گردید)

این یک روال پس پردازشی نیز وجود دارد که مسئول ارائه داده به صورتی است که توسط انسان قابل قرائت باشند (گراف، نمودار، جدول و ...).

۱۵-۳-٦ مروری بر شبیه ساز LAN

این شبیه ساز برای ایجاد یک ابزار تحلیلی انعطاف پذیر و پیشرفته برای معماری های شبکه محلی در نظر گرفته شده است. از این ابزار برای کمک به توسعه، انتخاب و ارزیابی معماری های شبکه محلی استفاده می شود و از محیط های ارتباطی و کنترل فرمان بی درنگ توزیع‌شده و گسترده $C_۳$ نیز پشتیبانی می کند. این شبیه ساز با هدف مقایسه طیف انبوهی از تنظیمات $C_۳$ توزیع‌شده احتمالی در نظر گرفته شده است.

۱. یک ساختار مدولار، که اجازه می دهد یک مدل با انواع مشخصات سیستمی خود را وفق دهد.

۲. یک روال گرداننده استاندارد که از ارتباطات موجود در یک سیستم $C_۳$ تقلید می کند.

۳. یک روال استاندارد که سیستم توزیع‌شده را بر اساس معیار ارزیابی کلی تجزیه و تحلیل می کند.

چنین طراحی امکان پیاده سازی، آزمایش و ارزیابی استراتژی های جدید برای بهبود عملکرد سیستم با اندکی تلاش را میسر می سازد.

۱۵-۳-۷ شبیه سازی رویداد بعدی

به تکنیک مدل‌سازی مورداستفاده در شبیه سازی نرم افزار LAN، شبیه سازی رویداد بعدی گفته می شود. این شبیه سازی دنیا را به‌صورت دنباله ای از رویدادهای پشت سر هم می بیند. اگر

یک خط بررسی یک فروشگاه اداری در این شبیه ساز، شبیه سازی شود، روند پرداخت به‌صورت دنباله زیر خواهد بود: (۱) خریدار وارد خط پرداخت می شود، ۲) خریدار شروع به پرداخت می کند، ۳) خریدار پرداخت را تکمیل می نماید. در میان این رویدادها خریدار فعالیت های دیگری نیز انجام می دهد اما اگر توجه ما روی زمان انتظار باشد، این فعالیت ها اهمیتی ندارند. برای درک طراحی و پیاده سازی شبیه سازی LAN، بهتر است با دیدگاه زمان صرف شده توسط شبیه سازی رویداد بعدی آشنا شویم. زمان در شبیه سازی رویداد بعدی به‌عنوان ابزاری از رویدادهای متوالی و محاسبه آمار مربوط به زمان دیده می شود. با رویدادها در واحد ساعت یا ثانیه در آینده به‌صورت یکسان برخورد می شود. شبیه سازی با ایجاد فایلی شروع می شود که حاوی رویدادهای آینده با زمان رخداد آن‌ها است. یک برنامه با حلقه ساده این فایل را اسکن می کند و رویدادی با کمترین زمان را انتخاب می نماید. در این حال، یک لحظه حافظه داخلی که شامل زمان شبیه سازی است، برای زمان بروز رخداد به روزرسانی می شود. بعد از بروز رخداد، برای زمان‌بندی رویدادهای وابسته دیگر می توان از محاسبات ریاضی یا منطقی استفاده نمود. در مثال خط پرداخت، این بدان معناست که وقتی پرداخت آغاز می شود، انتهای رویداد پرداخت برنامه ریزی می گردد. به این ترتیب، شبیه سازی از یک رویداد به رویداد دیگر پیش می رود و زمان نیز به‌صورت ثابت پیشرفت می کند. ماهیت تصادفی رویدادهای زمان‌بندی شده باعث می شود که شبیه سازی دارای مشخصات یک سیستم واقعی باشد. در مثال خط پرداخت، زمان صرف شده برای سرویس‌دهی به یک مشتری ثابت نیست، می تواند یک دقیقه یا ده دقیقه

باشد. زمان سرویس نیز به‌صورت تصادفی توزیع می شود. می توان گفت، زمان سرویس مشتریان قبلی تاثیری روی مشتریان آینده ندارد. زمان سرویس معمولاً دارای یک الگو است، یعنی، این احتمال زیاد است که زمان سرویس ۵ دقیقه باشد و بعید است که به ۲۰ دقیقه برسد. احتمال زمان یک سرویس خاص را می توان توسط الگوهای تئوری توصیف نمود. از این توزیع ها می توان برای تولید زمان سرویس یا الگوهای ورودی استفاده نمود که رخدادهای موجود در سیستم‌های واقعی را شبیه سازی می کند.

۱۵-۳-۸ پیاده سازی مدل LAN

شبیه ساز LAN یک بسته شبیه سازی همه منظوره است که برای مدل‌سازی انبوهی از معماری های موجود در شبکه محلی مورداستفاده قرار می گیرد. ساختار اصلی آن در شکل (۱۵-۱۸) نشان داده‌شده که شامل ۵ مولفه زیر است:

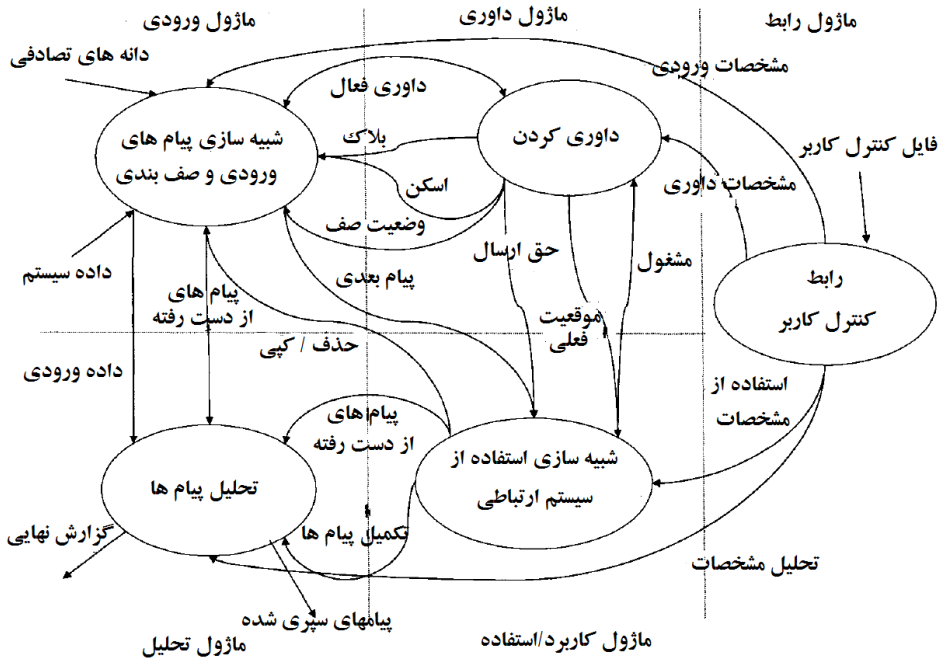
۱. ماژول ورودی. ماژول ورودی مربوط به تولید پیام هایی است که باید در یک صف واحد رابط قرار گیرند و با یکدیگر ارتباط برقرار کنند. این ماژول این باید مشکل سربار صف و احتمال عدم دسترسی یک پردازنده را رفع کند.

۲. ماژول داور. ماژول داور مربوط به تعیین واحد ارتباطی است که با لینک بعدی بر اساس خط مشی کنترل لینک ارتباطی ارتباط برقرار می کند.

۳. ماژول کاربرد/استفاده. این ماژول مربوط به مدل‌سازی عبور پیام ها از گره مبدا به گره مقصد با لینک ارتباطی است.

۴. ماژول تحلیل. این ماژول تحلیل آماری را روی پیام های موجود در سیستم انجام می دهد.

۵. ماژول رابط. این ماژول کنترل کلی مشخصات شبیه سازی را انجام می دهد.



شکل (۱۵-۱۸): ساختار اصلی شبیه ساز LAN

یکی از ماژول‌های شاخص MALAN ازنظر ویژگی‌های عمومی مدل‌سازی، ماژول استفاده است. مهم‌ترین مؤلفه‌های آن عبارتند از:

۱. ADD NEXT MESSAGE TO THE FILE (افزودن پیام بعدی به فایل): این فرآیند پیام بعدی را به پردازنده ای می برد که حق ارسال دارد و آن را در یک فایل ذخیره می کند تا اینکه به مقصد برسد.

۲. SIMULATE TOPOLOGY (شبیه سازی توپولوژی): این فرآیند از طریق توپولوژی شبیه سازی، پیام را مسیریابی می کند.

۳. SIMULATE TRANSACTION DELAY (شبیه سازی تاخیر تراکنش): این فرآیند عبور یک پیام از یک خط انتشار فیزیکی (لینک) را شبیه سازی می کند و شامل پیام های بازنشر شده نیز می شود.

۴. SIMULATE MESSAGE ERRORS (شبیه سازی خطاهای پیام). این مورد تعداد انتشارهای دوباره و پیام های از دست رفته ناشی از اطلاعات خط انتشار فیزیکی را تعیین می کند.

۵. UNFILE MESSAGE (خارج کردن از فایل پیام). این روند پیام ها را با توجه به تکمیل انتشار پیام از فایل پیام با حذف می کند.

۶. SIMULATE STATUS CHANGE (شبیه سازی تغییر وضعیت). این مورد زیان‌های یک لینک یا گره را شبیه سازی می نماید.

جدول (۱۵-۳) ترکیب نهادهای ارائه شده در این مدل را توصیف می کند. هر یک از اقلام موجود در جدول یکی از ویژگی های مربوط به نهادهای مدل می باشند. ویژگی ۱ زمان رویداد است، که شامل زمانی است که در آن پیام می رسد. ویژگی ۲ مربوط به نوع رویداد است که آن را به‌صورت رویداد

ورودی نشان می دهد، برای اینکه رویدادهای دیگری نیز در این سیستم رخ می دهند. ویژگی ۳ شماره IU مبدأ یا IU معلوم است که پیام را تولید می کند. ویژگی ۴ شماره IU مقصد است و مقصد نهایی پیام را توصیف می نماید. ویژگی ۷ اندازه پیام است که به‌صورت حرف بیان می شود. ویژگی ۱۷ و ۱۸ نیز زمانی مورداستفاده قرار می گیرند که یک پیام برای ارسال در یک بسته خیلی بزرگ باشد. در این مورد، پیام باید به چند پیام کوچک‌تر تقسیم شود، که هر یک دارای همان زمان تولید، مبدأ و مقصد هستند. از ویژگی ۱۷ برای شناسایی تعداد دنباله هر پیام چند بسته ای استفاده می شود. ویژگی ۱۸ نیز برای شناسایی تعداد کل بسته های موجود در کل پیام مورداستفاده قرار می گیرد. با ویژگی ۱۷ و ۱۸ می‌توان زمان دریافت پیام کامل را تعیین نمود. همراه با زمان‌های ورودی، اندازه پیام و مقصد پیام باید توسط سیستم تولید شود. در اینجا دوباره اشاره می کنیم که داده را می‌توان توسط اندازه گیری در سیستم واقعی یا با استفاده از توزیع های نظری تولید نمود. بنابراین، بخشی از ماژول ورودی به ترسیم از توزیع های اندازه پیام و مقصد پیام اختصاص دارد ویژگی های مناسب را مقدار دهی می نماید. بخش دیگری که باقی می ماند، تقسیم پیام هایی است که بیش از حداکثر اندازه پیام هستند و به‌صورت دنباله ای از پیام های کوچک‌تر می آیند.

جدول (۱۵-۳): ورودی ها در مدل شبیه ساز LAN

شماره ویژگی	نام داده	نوع داده	حروف اختصاری
۱	زمان رویداد	حقیقی	ETIM
۲	نوع رویداد	حقیقی	ETYP

۳	شماره واحد رابط مبدا (IU)	صحیح	SP
٤	شماره واحد رابط مقصد (IU)	صحیح	DP
٥	شماره واحد رابط فعلی (IU)	صحیح	PP
٦	زمان تولید	حقیقی	GT
۷	اندازه پیام (حروف)	صحیح	MS
۸	طول سربار پیام (بیت)	صحیح	MO
۹	زمان انتظار پیام Δt_1 (در صف پردازنده)	حقیقی	WT۱
۱۰	زمان انتظار پیام Δt_2 (تبدیل از صف به IU)	حقیقی	WT۲
۱۱	زمان انتظار پیام Δt_3 (در IU)	حقیقی	WT۳
۱۲	زمان انتقال پیام	حقیقی	TT
۱۳	زمان انتقال پیام	حقیقی	XFER
١٤	تعداد توقف ها	صحیح	NS
١٥	تعداد انتشارهای دوباره	صحیح	RT
١٦	از دست رفتن پیام	صحیح	ML
١٧	تعداد متوالی از پیام های چند بسته ای	صحیح	NMES
١٨	تعداد بخش های یک پیام بسته بندی شده	صحیح	PARTS
١٩	زمان پیام برای تکمیل	صحیح	MTTC
٢٠	اولویت پیام	صحیح	MP
٢١	شماره شناسایی پیام (ID)	صحیح	MI

تعداد کل پیام های تولید شده در ویژگی ۱۸ هر پیام قرار می گیرد. هر پیام یک شماره متوالی دریافت می کند، ویژگی ۱۷. هنگامی که پیام وارد سیستم شد، باید در خط انتظار بماند یا تا زمانی که IU قادر به انتشار آن باشد در صف قرار گیرد. برای سهولت این کار، یک صف FIFO تشکیل می شود تا

حاوی پیام های انتظار باشد. در یک سیستم واقعی، این صف اندکی حافظه حقیقی مصرف می نماید که معمولاً محدود است. در سیستم شبیه سازی شده، این محدودیت را نیز باید مدنظر قرار داد. هنگامی که حافظه صف از حد بیشتر شد، باید اقدامات مناسبی اتخاذ شود. این اقدامات شامل: ۱) انتظار تا زمانی که حافظه کافی برای پیام آزاد باشد، ۲) دور انداختن پیام و ۳) بازنویسی یک پیام قدیمی تر. هر یک از این موارد را می توان در این حالت انتخاب نمود.

توصیف ماژول توپولوژی

ماژول توپولوژی عمدتاً شامل گروهی از زیرروال هایی است که بازیابی و اصلاح داده‌هایی را تسهیل می نماید که به‌صورت فیزیکی شبکه کامپیوتری محلی را توصیف می کنند. این روال ها در بدنه برنامه شبیه سازی به جز گروه اولیه باقی می مانند که یک نهاد جداگانه است.

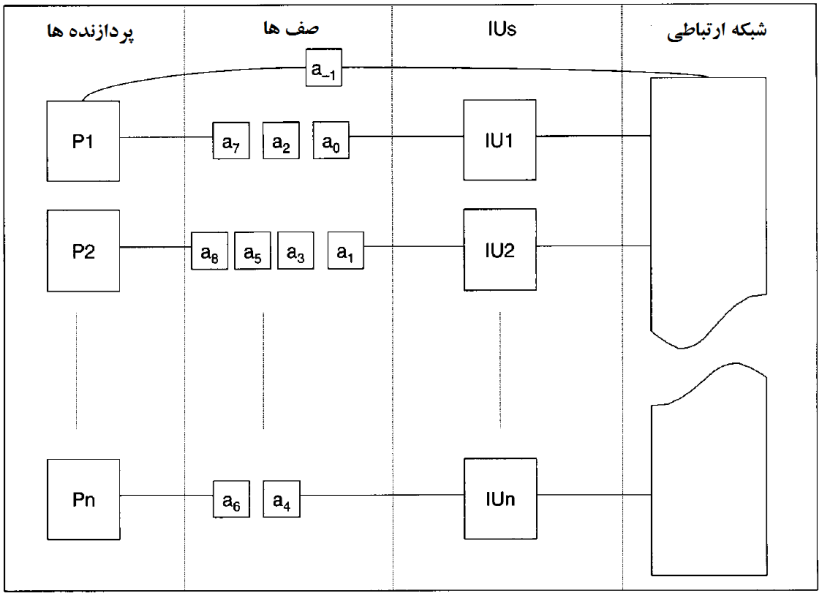
۱۵-۳-۹ ماژول تحلیل

هدف ماژول تحلیل، ارائه معیارهای کمی است که اثربخشی سیستم‌های پردازش توزیع‌شده را در بر دارد و اندازه های آماری ایجاد می نماید و از آن‌ها می توان برای مقایسه سیستم‌های پردازش توزیع‌شده با طرح های موازیانس استفاده نمود. برای رسیدن به این اهداف، باید عوامل ثابتی که سیستم‌های پردازش توزیع‌شده را یکنواخت می کنند شناسایی کنیم و اندازه آماری در دست داشته باشیم که توسط آن‌ها این عوامل را با یکدیگر مقایسه کنیم. به همین دلیل می توان گفت که یک زبان مشترک تحلیل باید داشته باشیم که توسط آن طیف انبوهی از سیستم‌های توزیع‌شده توصیف شوند.

ایجاد معیار تحلیل

برای توسعه معیار تجزیه و تحلیل در سطح گسترده، باید ویژگی هایی شناسایی شوند که در میان سیستم‌های پردازش توزیع‌شده مشترک هستند. این ویژگی های مشترک در معیارهای تحلیلی توسعه خواهند یافت و معیارهای نسبی سیستم پایه را تحلیل خواهند نمود. در توسعه ویژگی های مشترکی، این مناطق موردبررسی قرار می گیرند: ۱) ساختار فیزیکی اصلی شبکه های پردازش توزیع‌شده، ۲) دنباله اصلی رویدادها، ۳) کارکرد کلی شبکه های پردازش توزیع‌شده.

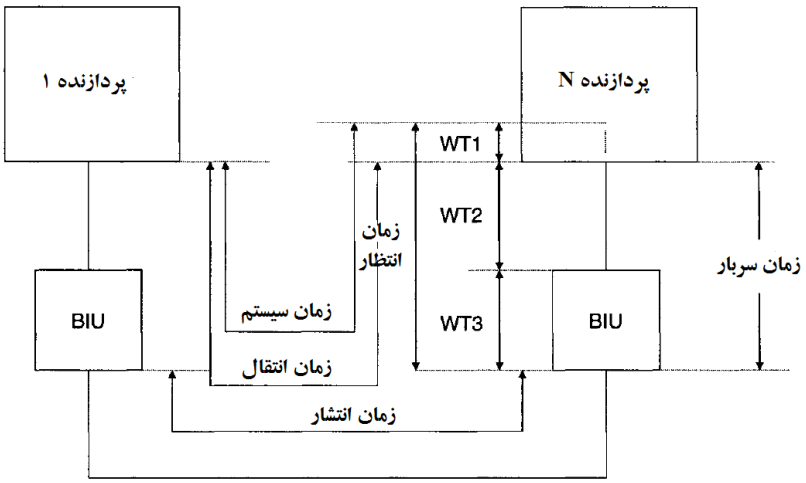
برای ایجاد انعطاف پذیری و سهولت، بیشتر سیستم‌های پردازش توزیع‌شده از یک فلسفه طراحی مدولار استفاده می کنند. مدولاریته منجر به یک ساختار فیزیکی مشترک می شود که به سیستم‌های پردازش توزیع‌شده اجازه می دهد تا در مولفه های کاربردی متعددی تقسیم شوند. این مولفه ها را می توان به‌صورت جداگانه آزمایش و ارزیابی نمود. تقسیم ارزیابی یک سیستم در مولفه های کاربردی باعث می شود که تحلیل دقیقی از عوامل واسط داشته باشیم که روی نقاط قوت و ضعف یک سیستم نقش دارند.



شکل (۱۵-۱۹): مولفه های عملی یک سیستم پردازش توزیع‌شده.

مؤلفه‌های کاربردی اصلی که ساختار فیزیکی یک سیستم پردازش توزیع‌شده را تشکیل می‌دهند، در شکل (۱۵-۱۹) نشان داده شده اند. این شکل هر سیستم پردازش توزیع‌شده را ازنظر مولفه های زیر نشان می دهد: ۱) تعدادی از پردازنده ها که پیام ها را تولید و از آن‌ها استفاده می کنند، ۲) یک خط انتظار یا صف که حاوی پیام هایی است که نمی توانند بلافاصله سرویس دریافت کنند، ۳) یک واحد رابط که پیام ها را برای انتشار آماده می کند و ۴) یک شبکه ارتباطی که انتشار فیزیکی واقعی داده را انجام می دهد. پیاده سازی فیزیکی این مولفه ها به‌صورت گسترده از یک سیستم تا سیستم دیگر متفاوت است. طرح شکل (۱۵-۱۹) یک تصویر دقیق و کلی از سیستم‌های پردازش توزیع‌شده را نشان می دهد. طرح فیزیکی شکل (۱۵-۱۹) امکان شناسایی ویژگی های مشترک خاصی را فراهم می کند که در گراف

های حاصل قابل بحث و بررسی هستند. ساختار اولیه مورد کمی شکل (۱۵-۱۹)، صف یا خط انتظار است. طول این صف ها به ما اطلاعات کمّی مربوط به اثرپذیری سیستم ارتباطی پایه را می دهد. صف های خیلی طولانی یا غیرمتعادل می توانند نشان دهنده حضور تنگناهای سیستم باشند. صف هایی که رشد سریع و گسترده ای دارند می توانند بارهای اوج و پاسخ ضعیف را نشان دهند.



شکل (۱۵-۲۰): معیارهای ارزیابی LAN

مولفه های اصلی که ساختار رویداد کاربردی را در شبکه پردازش توزیع‌شده معمولی تشکیل می دهند، در شکل (۱۵-۲۰) نشان داده‌شده اند. این شکل همان طرح فیزیکی عمومی شکل (۱۵-۱۹) را به تصویر می کشد اما عبور پیام از سیستم فیزیکی به مراحل یا فازهای خاص را از تفکیک می نماید. رویدادهای اصلی مورد نظر در امتداد مسیر پیام شکل (۱۵-۲۰) به این صورت هستند: ۱) یک پیام وارد می شود، ۲) یک پیام وارد صف می شود، ۳) یک پیام صف را ترک می کند، ۴) یک پیام برای واحد رابط در

دسترس قرار می گیرد، ۵) یک پیام شروع به انتشار می نماید، ۶) یک پیام انتشار را به پایان می برد و ۷) یک پیام در دسترس پردازنده گیرنده قرار می گیرد.

این نقاط بررسی مشترک قابل توجه هستند، برای اینکه امکان اندازه گیری زمان را دارند که عبور پیام از سیستم را می تواند ترسیم کند. تا زمانی که یک سیستم خاص با دقت ارتباط را پیاده سازی می کند، زمان بندی یکی از عوامل حیاتی به شمار می رود. یعنی، سرعتی که در آن پیام های منتشر شده به دقت تکمیل می شوند، از اهمیت ویژه برخوردار است. این مجموعه پست های بازرسی امکان تحلیل کلی و تحلیل تاخیرات واسط مربوط به فرآیند ارتباطی را می دهند. در شکل (۲۰-۱۵) زمان بین نقاط بازرسی اصلی و ترکیبات نقاط بازرسی برای مقادیر توصیفی خاصی مطرح شده است. این مقادیر برای هر اجرای شبیه سازی روی شبکه های پردازش توزیع‌شده خاص کامپایل شده اند. در ادامه به طور کلی تری به آن‌ها می پردازیم:

۱. زمان سیستم، زمان میان تولید پیام در پردازنده مبدا و پذیرش پیام توسط پردازنده مقصد است. زمان سیستم، تاخیر کلی سیستم پردازش توزیع‌شده که در معرض یک پیام قرار دارد را پیدا می کند.

۲. زمان انتقال زمانی است بین زمانی که پیام صف را ترک می کند و در مقصد پذیرفته می شود. زمان انتقال نشان دهنده زمان موردنیاز برای سیستم برای برقراری ارتباط است، صرفنظر از زمان سپری شده برای مبادله فرآیند انتقال.

۳. زمان انتشار، زمانی است که یک پیام در پردازش انتشار اطلاعات فیزیکی سپری می کند.

۴. زمان انتظار نیز زمانی است که باید پیش از انتشار پیام صرف شود. این مقدار به چهار کمیت کوچک‌تر تقسیم می شود: wt_1، wt_2، wt_3 و wt_4 که به‌صورت زیر توصیف می شوند:

- wt_1 زمانی است که یک پیام در صف می گذراند.
- wt_2 فاصله بین حذف پیام از صف تا قرار گرفتن در جایی است که IU شروع به آماده سازی پیام برای انتشار می کند.
- wt_3 زمان موردنیاز برای آماده سازی یک پیام برای انتشار را محاسبه می کند.
- wt_4 شامل زمان موردنیاز برای در دسترس قرار دادن پیام برای پردازنده مقصد در زمان تکمیل انتشار است.

۵. زمان سربار برابر است با مجموع wt_2، wt_3 و wt_4. زمان سربار زمانی است که IU برای انتشار پیام هزینه می کند.

معیار تحلیل را می توان از نقطه نظر کاربردی در نظر گرفت. معیار کاربردی امکان ارزیابی و مقایسه عملکرد سیستم‌های پردازش توزیع‌شده را می دهد. این معیار را می توان به گروه های زیر تقسیم کرد: ۱) مقدار اطلاعاتی که توسط سیستم ارتباطی در واحد زمان حمل می شود (یعنی توان عملیاتی)، ۲) مقدار اطلاعات مفیدی که در واحد زمان منتقل می شود و شامل سربار نیز می شود (یعنی، توان عملیاتی اطلاعات)، ۳) از دست رفتن اطلاعات در پردازش اطلاعات، ۴) مقدار سربار اطلاعات و ۵) بخشی از داده که دیر می رسد.

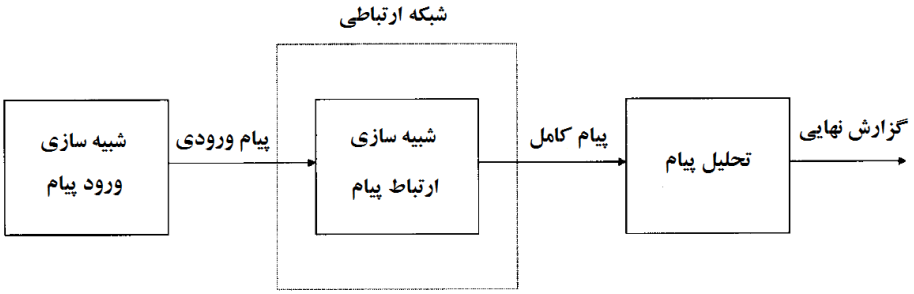
آمار توان عملیاتی حجم کلی اطلاعات حمل شده توسط یک سیستم پردازش توزیع‌شده را در زمان واحد محاسبه می کند. این مقدار یک شاخص کلی از ظرفیت و بهره وری سیستم پردازش توزیع‌شده است. متاسفانه، حجم اطلاعات حقیقی منتقل شده تا بخشی از سربار روی پیام کاهش یافته است. اطلاعات سربار نیز بخشی از یک پیام است که توسط سیستم پردازش توزیع‌شده متصل شده تا ارتباطات را تسهیل بخشد. اندازه گیری توان که به‌عنوان سربار در نظر گرفته می شود، توان عملیاتی اطلاعات نامیده می شود. علاوه بر جریان کلی اطلاعات در سیستم، به از دست رفتن اطلاعات نیز باید توجه کنیم. این اتلاف می تواند در اثر این سه عامل رخ دهد:

۱) از دست رفتن پیام به دلیل وجود یک صف تکمیل، ۲) از دست رفتن پیام به دلیل وجود بیت معیوب در حین انتشار و ۳) از دست دادن پیام به دلیل عیب بخشی از سیستم. این مقادیر به‌صورت درصدی از کل تعداد پیام های منتشر شده محاسبه می شوند. آمار مربوط به کل تلفات پیام نیز محاسبه خواهد شد. این مقدار سرباری که به هر پیام متصل می شود برای ما جالب توجه است. این مقدار امکان تحلیل تخریب عملکرد سیستم ناشی از سربار را فراهم می کند. این آمار سربار را به‌صورت درصدی از کل اطلاعات منتقل شده بیان می کند. یک معیار پیچیده تر برای ارزیابی عملکرد، آمار تاخیر داده است. این آمار بخشی از پیام ها را که بعد از زمان انقضا می رسند، ارزیابی می کند. ازنظر عملی، این معیار یکی از موارد مربوط به محیط های واقعی است. اگر تمام پیام ها در زمان اختصاصی وارد شوند، سیستم در ظرفیت کار می کند و به اوج نیازها پاسخ می دهد. به طور خلاصه، معیار مورداستفاده برای تحلیل توسط سه شاخصه سیستم‌های پردازش توزیع‌شده تولید می شوند:

۱. ساختار فیزیکی اصلی. این ویژگی معیاری مثل طول صف ایجاد می کند که ناشی از لینک فیزیکی میان واحد رابط و پردازنده است.

۲. ساختار رویداد. این ویژگی معیاری مثل زمان انتشار ایجاد می کند که نتیجه نیاز به انتشار فیزیکی داده در برخی نقاط سیکل ارتباطی است.

۳. عملکرد کلی. این ویژگی معیاری مثل توان عملیاتی ایجاد می کند که ناشی از عملکرد کلی سیستم (به‌عنوان‌مثال، ارتباط) است.



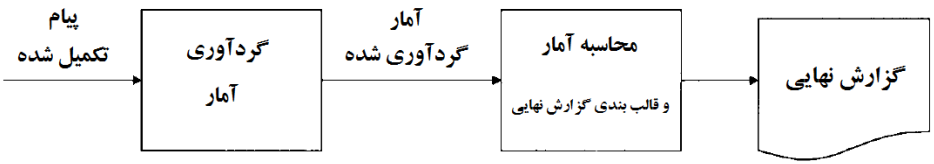
شکل (۱۵-۲۱): گراف جریان پیام شبیه سازی شده

معیار تحلیل و شبیه سازی

از نقطه نظر تحلیل آماری، شکل (۱۵-۲۱) ماهیت اصلی مدل را نشان می دهد. بلوک سمت چپ پیام های شبیه سازی را نشان می دهد که از سیستم واقعی به شبکه ارتباطی می روند. پیام های شبیه سازی از پیام های سیستم واقعی نیستند اما بافرهای حافظه کامپیوتری هستند که حاوی اطلاعات بدون در نظر گرفتن ماهیت و تاریخچه پیام می باشند. بلوک مرکزی شبیه سازی پیام های عبوری از شبکه ارتباطی را نشان می دهد. در حین این عبور، داده که حاوی سابقه پیام است، به پیام شبیه

سازی شده افزوده می شود. هنگامی که پیام تکمیل گردد (به مقصد برسد) یا از دست برود (به مقصد نرسد)، پیام توسط ماژول تحلیل که در سمت راست نشان داده‌شده، تجزیه و تحلیل می گردد.

در طول شبیه ساز، پیام های زیادی مسیر موجود در شکل (۱۵-۲۲) را از آن خود می کنند. به تعداد زیادی از پیام ها نیاز است تا اهمیت آماری ایجاد گردد. این اشاره به این واقعیت دارد که نمونه زیادی از رخدادها را باید در نظر گرفت تا انحراف معیاری که ممکن است در اثر انتخاب یک نمونه تصادفی خیلی کوچک شود، ایجاد گردد. ساختار نرم افزار برای جمع آوری آمار و تشکیل گزارش نهایی نیز در شکل (۱۵-۲۲) نشان داده‌شده است. در حین اجرای شبیه سازی، اطلاعات مربوط به تعداد زیادی از پیام های تکمیل شده گردآوری و ذخیره می شوند و حافظه اشغال شده توسط این پیام ها آزاد می شود. در انتهای اجرای شبیه سازی، این داده‌های گردآوری شده برای محاسبات آماری مورداستفاده قرار می گیرند که به‌صورت گزارش نهایی فرمت و ارائه می شوند.

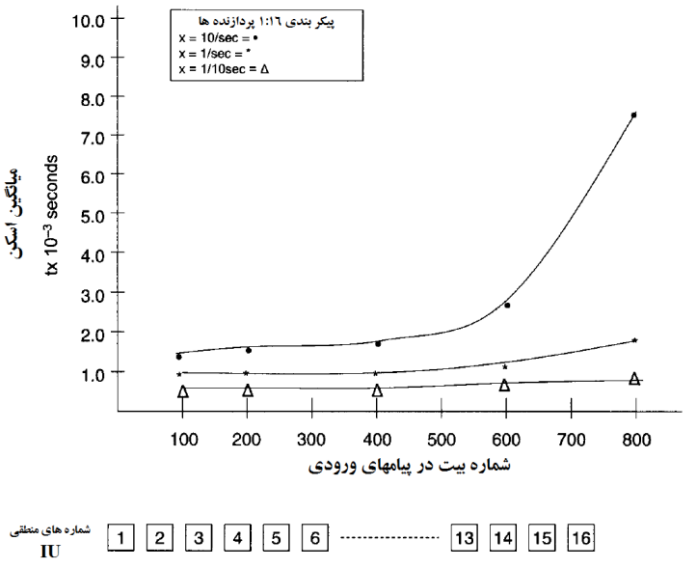


شکل (۱۵-۲۲): نمودار عملکرد ماژول تحلیل

خروجی آماری

آمار سیستم را می توان به سه گروه اصلی تقسیم کرد: ۱) مستقل از زمان، ۲) تداوم زمان و ۳) دوره ای. آمار مستقل از زمان از انتزاع های مستقل به دست می آید. میانگین سنتی و انحراف استاندارد را می توان از این گروه محاسبه نمود. این داده‌ها، که در حین اجرای شبیه سازی گردآوری می شوند، به‌صورت زیر هستند:

۱) مجموع هر قطعه از داده مشاهده شده، ۲) مجموع هر قطعه از مربع داده، ۳) تعداد مشاهدات و ۴) حداکثر مقدار مشاهده شده. از داده گردآوری شده، میانگین، انحراف استاندارد، حداکثر مقدار مشاهده شده برای گزارش نهایی محاسبه و تنظیم خواهد شد. این آمار برای تمام نقاط داده مستقل از زمان ارائه خواهد شد. آمار مستقل از زمان هنگامی مهم است که زمانی که یک پارامتر ارزش خود را حفظ می کند، حیاتی می‌شود. به‌عنوان‌مثال می توان از خط انتظار نام برد. اگر خط دارای ۱۰ عضو برای ۲۰ دقیقه و یک عضو برای ۱ دقیقه باشد، میانگین برابر ۲/(۱۰+۱) یا ۵,۵ نخواهد بود. این مقدار نشان می‌دهد که تقریباً ۵ عضو در خط برای ۲۱ دقیقه حاضر است. بنابراین می‌توان گفت که میانگین درست به‌صورت ۱×۱/۲۰+۱۰×۲۰/۲۱ یا ۹,۵۷ یا تقریباً ۱۰ است. این میانگین تداوم زمانی است.



شکل (۱۵-۲۳): نقاط ارزیابی عملکرد مثال.

همانطورکه در این مورد دیده می شود، میانگین توسط دوره زمانی محاسبه می شود که مقدار ثابت باشد. برای انحراف استاندارد ثبات زمانی می توان یک استدلال مشابه ایجاد کرد. این داده‌ها، که در حین یک شبیه سازی برای مورد تداوم زمانی جمع آوری می‌شوند، به‌صورت زیرند: ۱) مجموع تعداد مقادیر دوره ای که آن مقدار حفظ می‌شود، ۲) مجموع مقدار مربع مشاهده شده ای که طی آن دوره مقدار حفظ می شود، ۳) حداکثر مقدار مشاهده شده و ۴) دوره کلی مشاهدات. از طریق این داده‌های گردآوری شده، میانگین دوام زمانی، انحراف استاندارد دوام زمانی و حداکثر مقدار مشاهده شده برای گزارش نهایی محاسبه و فرمت خواهند شد. این آمار برای تمام نقاط داده با تداوم زمانی ارائه خواهند شد.

آمار دوره ای برای ایجاد یک ترسیم از مشاهدات و به‌صورت تابعی از زمان ایجاد شده اند. این گروه از آمار را می توان به‌صورت سیستمی دید که به موقع اجرا می شود. داده‌ها همانند مثال قبل گردآوری شده اند، به جز اینکه بجای مجموع آمار، یک گراف نقطه داده مجزا از زمان نسبت به مقدار نقاط داده ترسیم خواهد شد. این نقاط برای تمام گروه های آمار دوره ای تولید خواهد شد. برخی نقاط مثال برای ارزیابی ها روی HXDP اجرا خواهند شد و شبکه های حلقه توکن نیز در شکل (۱۵-۲۳) نشان داده‌شده اند.

۱۵-۴ خلاصه

این فصل سودمندی مدل‌سازی شبیه سازی و تحلیلی را برای مطالعه یک مولفه از یک شبکه منطقه محلی نشان می دهد که در این مورد زمان اسکن کنترل برای صف بندی مدل و توان عملیاتی برای شبیه سازی است. این مدل ها یک ابزار نسبتاً ساده برای استخراج این اطلاعات هستند. از آن‌ها برای انواع دیگری از مسائل LAN استفاده می شود. خوانندگانی که تمایل بیشتری در زمینه این مدل‌سازی دارند، به مرجع این فصل مراجعه نمایند.

نمایه ها