دانشکده مهندسی کامپیوتر

گزارش پروژه درس

ارزیابی کارایی

نام دانشجو: مینا فریدی
۸۱۰۱۰۰۴۳۰

نیم‌سال دوم
سال تحصیلی ۱۴۰۱-۱۴۰۰

فصل ۱

گزارش شبیه سازی ۵-۲

در این پروژه، تعدادی از مثال‌های کتاب Law simulation modeling به زبان متلب شبیه‌سازی شده‌اند.

رویداد پایانی شبیه‌سازی در زمانی برنامه‌ریزی می‌شود که هزارمین زمان پاسخ مشاهده می‌شود، و برنامه‌ریزی شده است که بلافاصله رخ دهد، یعنی در آن زمان. بدیهی است که روش های دیگری نیز وجود دارد که در زیر می توان این قانون توقف را اجرا کرد.

n مقداردهی اولیه جداگانه رویداد ورود (یعنی پایان فکر–زمان) به این واقعیت اشاره دارد که هر یک از n پایانه در ابتدا برای داشتن چنین رویدادی برنامه ریزی می شود. همچنین رویدادهای ورود و پایان اجرای CPU به طور بالقوه می توانند یکدیگر را برنامه ریزی کنند. یک ورود می‌تواند پایان اجرای CPU را برنامه‌ریزی کند اگر کار ورودی CPU را بی‌حرکت بیابد، و یک رویداد پایانی CPU می‌تواند در صورتی که کار خروج از CPU تمام شده و به ترمینال خود بازگردد، ورود را برنامه‌ریزی کند. همچنین، یک رویداد پایانی–اجرای CPU می‌تواند در صورت خروج یک کار از CPU قبل از اتمام کل پردازش، خود را برنامه‌ریزی کند، و تنها برای اینکه بفهمد صف خالی است و CPU بی‌کار است، به صف بازگشته است. مشاغل دیگر در پایانه های خود هستند. در نهایت، توجه داشته باشید که رویداد پایانی شبیه‌سازی را می‌توان فقط از رویداد پایانی CPUrun و در زمان صفر برنامه‌ریزی کرد، در صورتی که یک کار تمام شده از CPU خارج شود و آخرین زمان پاسخ مورد نیاز را تامین کند. سه لیست از رکوردها استفاده خواهد شد، یکی مربوط به کارهای موجود در صف (فهرست ۱)، یکی برای کاری که توسط CPU ارائه می شود (لیست ۲)، و دیگری برای لیست رویداد (لیست ۲۵، طبق

معمول). این لیست ها دارای ویژگی های زیر هستند: ویژگی‌های این فهرست «سرور» ساختگی نیستند. آنها اطلاعات لازم را در مورد کار در حال سرویس حمل می کنند، زیرا ممکن است مجبور باشد چندین بار CPU را بازبینی کند و زمان رسیدن و زمان باقیمانده سرویس باید همراه باشد تا به درستی پردازش شود. همچنین توجه داشته باشید که ما ویژگی‌های لیست‌های ۱ و ۲ را با هم تطبیق داده‌ایم، به طوری که انتقال یک رکورد بین این لیست‌ها به سادگی با فراخوانی listremove و سپس listfile بدون نیاز به تنظیم مجدد ویژگی‌ها در آرایه انتقال انجام می‌شود.

در نهایت، صراحتاً ترمینال مبدأ یک کار را دنبال نمی‌کنیم، به طوری که وقتی پردازش آن در رایانه کامل شد، نمی‌دانیم آن را به چه پایانی برگردانیم تا زمان فکر بعدی آن شروع شود. این قطعاً در واقعیت انجام نمی‌شود، زیرا اپراتورهای پایانه خروجی یکدیگر را پس می‌گیرند، اما در شبیه‌سازی، ما نیازی به نشان دادن مالکیت هر شغل توسط یک پایانه خاص نداریم، زیرا ما فقط به طور کلی می‌خواهیم (یعنی روی همه پایانه‌ها).) معیارهای عملکرد برای زمان پاسخ. علاوه بر این، همه پایانه‌ها از نظر احتمال یکسان هستند، زیرا توزیع‌های زمان فکر و زمان CPU یکسان هستند.

از آنجایی که تنها یک آمار زمان گسسته مورد علاقه (زمان پاسخ) وجود دارد، ما تنها به یک متغیر sampst نیاز داریم: برای هر یک از آمارهای زمان پیوسته مورد نظر (تعداد در صف و استفاده از CPU)، لیست مربوطه وجود دارد که طول مقدار مورد نظر را نشان می دهد، بنابراین ما می توانیم دوباره خروجی را از طریق filest بدست آوریم و به هیچ یک از متغیرهای timest خود نیاز نداریم. دو نوع متغیر تصادفی برای این مدل وجود دارد و ما از تخصیص جریان زیر استفاده می کنیم: پس از کتابخانه simlib ، ثابت های نمادین را برای انواع رویداد، اعداد لیست، تعداد متغیر sampst و جریان های اعداد تصادفی تعریف می کنیم.

متغیرهای غیر simlib اعلام شده‌اند، از جمله ints برای حداقل و حداکثر تعداد پایانه‌ها در سراسر شبیه‌سازی (minterm و ،(maxterms افزایش تعداد پایانه‌ها در سراسر شبیه‌سازی (incrterms) ، تعداد پایانه های یک شبیه سازی خاص (numterms) ، تعداد دفعات پاسخ مشاهده شده در شبیه سازی فعلی (numresponses) ، و تعداد پاسخ های مورد نیاز numresponsesrequired . شناورها فقط برای پارامترهای ورودی برای میانگین زمان فکر و سرویس، کوانتوم Q و زمان مبادله T مورد نیاز هستند. توابع غیر Simlib نوشته شده توسط کاربر، از جمله توابع رویداد برای ورود و رویدادهای پایانی CPU اعلام شده است. ما یک تابع غیر رویدادی، ،startCPUrun نوشته‌ایم تا فعالیت خاصی را پردازش کنیم که ممکن است هنگام ورود یا پایان اجرای CPU رخ دهد، و از تکرار همان بلوک کد در هر یک از آن توابع رویداد اجتناب می‌کنیم. تابع راه‌اندازی مدل جداگانه‌ای وجود ندارد زیرا برای مقداردهی اولیه این مدل (فراتر از آنچه simlib در initsimlib انجام می‌دهد) نیازی نیست، بنابراین این فعالیت به سادگی در تابع اصلی قرار داده شد. برای این مدل، ما انتخاب کردیم که از گزینه قالب‌بندی خروجی استاندارد استفاده نکنیم، زیرا ما واقعاً هشت شبیه‌سازی

جداگانه انجام می‌دهیم و می‌خواهیم داده‌های خروجی را در یک جدول سفارشی، با یک خط در هر شبیه‌سازی، مرتب کنیم. همچنین، ما می‌خواهیم فقط میانگین معیارهای عملکرد خروجی را به‌جای همه ویژگی‌های آنها (حداکثر و غیره) بدست آوریم.

در تابع اصلی فایل های ورودی و خروجی را باز می کنیم، پارامترهای ورودی را می خوانیم و سپس آنها را در عنوان گزارش می نویسیم. از آنجایی که ما هشت شبیه سازی را انجام خواهیم داد، با یک خط خروجی در هر شبیه سازی، در این زمان عناوین ستون ها را نیز برای خروجی می نویسیم. هر شبیه‌سازی با یک مقدار اولیه اولیه simlib با فراخوانی initsimlib آغاز می‌شود، سپس maxatr را روی ۴ تنظیم می‌کند، numresponses را روی ۰ مقداردهی می‌کند، و اولین ورود به CPU را از هر ترمینال با فراخوانی $_{s}chedulenum_{t}erms.num_{t}erms1."dowhile".()dowhilefor.endCPUrun."for"$.event

فایل خروجی، tscomp.out است. ازدحام درکامپیوتر با افزایش تعداد پایانه ها بدتر می شود، همانطور که با میانگین زمان پاسخ، متوسط طول صف و استفاده از CPU اندازه گیری می شود. به طور خاص، به نظر می رسد که این سیستم می تواند حدود ۶۰ پایانه را قبل از اینکه میانگین زمان پاسخ به مقدار بسیار بدتر از ۳۰ ثانیه کاهش دهد، اداره کند. در این سطح، می بینیم که میانگین طول صف حدود ۳۰ شغل خواهد بود، که می تواند برای تعیین میزان فضای مورد نیاز برای نگهداری این مشاغل مفید باشد (حداکثر طول صف ممکن است اطلاعات بهتری برای این منظور باشد). علاوه بر این، CPU تقریباً همیشه با چنین سیستمی مشغول است. با این حال، اخطار معمول ما در مورد این نتیجه‌گیری‌ها صدق می‌کند: داده‌های خروجی که آنها بر اساس آن‌ها هستند، تنها از یک اجرای سیستم (با طول تا حدودی دلخواه) ناشی می‌شوند و بنابراین دقت ناشناخته‌ای دارند.