دانشکده مهندسی کامپیوتر

گزارش پروژه درس

ارزیابی کارایی

نام دانشجو: مینا فریدی
۸۱۰۱۰۰۴۳۰

نیم‌سال دوم
سال تحصیلی ۱۴۰۱-۱۴۰۰

شناسایی رویدادها به این صورت است که ورود یک رویداد نوع ۱ است و یک خروج (تکمیل خدمات) یک رویداد نوع ۲ است. در مرحله بعد، لیست های Simlib و ویژگی ها را در رکوردهای آنها تعریف می کنیم. لیست ۱ (نماینده صف) کاملاً شبیه آرایه timearrival استفاده شده در قسمت قبلی، با این تفاوت که اکنون ما از قابلیت های پردازش لیست Simlib استفاده می کنیم. لیست ۱ فقط یک ویژگی دارد. لیست ۲ نشان دهنده سرور است و یا خالی خواهد بود (اگر سرور بیکار باشد) یا حاوی یک رکورد واحد است (اگر سرور مشغول است). یک رکورد در لیست ۲ زمانی که سرور مشغول است یک رکورد "ساختگی" است، زیرا هیچ ویژگی واقعی ندارد. هدف از تعریف چنین لیستی، اجازه استفاده از filest در پایان شبیه سازی برای به دست آوردن استفاده از سرور است. همچنین، می‌توانیم با پرسیدن اینکه [۲]listsize برابر با ۱ است، متوجه می‌شویم که سرور مشغول است یا خیر.

این استفاده از لیست ساختگی از این جهت راحت است که متغیر serverstatus حذف شده است، و ما نیازی به استفاده از timest در طول یا در پایان شبیه سازی برای دریافت استفاده نداریم. با این حال، از نظر محاسباتی کارآمدترین رویکرد نیست، زیرا همه ماشین آلات لیست های پیوندی هر زمان که سرور وضعیت را تغییر می دهد، فراخوانی می شود، نه اینکه صرفاً مقدار یک متغیر serverstatus را تغییر دهد. (این نمونه خوبی از مبادله بین زمان محاسبه و زمان تحلیلگر در کدنویسی یک مدل است.) در نهایت، لیست ۲۵ لیست رویداد است، با ویژگی ۱ زمان رویداد و ویژگی ۲ نوع رویداد است. این در همه برنامه های simlib مورد نیاز است، اما برای برخی از مدل ها از ویژگی های اضافی برای رکورد رویداد در لیست استفاده می کنیم. در مرحله بعد، ما باید تمام متغیرهای Sampst و timest استفاده شده را شناسایی کنیم. تنها متغیر sampst ما به شرح زیر است: از آنجایی که می‌توانیم هم آمار تعداد در صف و هم آمار استفاده را با استفاده از Filest بدست آوریم برای این مدل به هیچ متغیری نیاز نداریم. در نهایت، جریان‌های اعداد تصادفی جداگانه را برای تولید زمان‌های بین ورود و سرویس اختصاص می‌دهیم.

اولین کاری که انجام می‌دهیم این است که کتابخانه simlib را صدا می¬زنیم.که برای همه برنامه‌هایی که از simlib استفاده می‌کنند لازم است. برای اینکه کد قابل خواندن تر و کلی تر باشد، ثابت های نمادین را برای انواع رویداد، اعداد لیست، متغیر sampst و جریان های اعداد تصادفی تعریف می کنیم. ما همچنان باید برخی از متغیرهای غیر simlib را برای مدل اعلام کنیم و همچنان باید توابع خودمان را داشته باشیم، initmodel برای مقداردهی اولیه این مدل خاص، رسیدن و خروج برای رویدادها، و یک تولید کننده گزارش، اما دیگر نیازی به تابع زمان بندی یا تابع expon نداریم، زیرا آنها توسط simlib ارائه می شوند.

تابع اصلی هنوز باید توسط کاربر نوشته شود. پس از باز کردن فایل های ورودی و خروجی، پارامترهای ورودی را می خوانیم و بلافاصله آنها را می نویسیم تا تأیید کنیم که آنها به درستی خوانده شده اند (و خروجی خود را مستند کنیم). فراخوانی simlib initsimlib را مقدار دهی اولیه می کند و پس از آن maxatr را روی ۴

قرار می دهیم. در حالی که هیچ رکوردی با بیش از دو ویژگی نداریم، maxatr باید حداقل ۴ باشد تا Simlib به درستی کار کند. ما از هیچ لیست رتبه بندی شده ای استفاده نمی کنیم (به غیر از لیست رویداد)، بنابراین نیازی به تنظیم چیزی در آرایه listrank نداریم. همچنین، هر دو تابع زمان پیوسته (طول صف و وضعیت سرور) در ابتدا صفر هستند، بنابراین نیازی نیست مقادیر پیش‌فرض آنها را نادیده بگیریم. سپس تابع نوشته شده توسط کاربر initmodel برای تنظیم متغیرهای مدل سازی غیر simlib خودمان فراخوانی می شود. بقیه تابع اصلی ما نیازی به به روز رسانی انباشته کننده های آماری زمان پیوسته نداریم زیرا simlib به صورت داخلی از آن مراقبت می کند. در تابع initmodel را نشان می دهد که با تنظیم شمارنده numcustsdelayed روی ۰ برای تعداد تاخیرهای مشاهده شده شروع می شود. اولین رویداد ورود سپس با فراخوانی eventschedule با زمان رویداد مورد نظر a) (float به عنوان آرگومان اول و نوع رویداد (int) به عنوان آرگومان دوم برنامه ریزی می شود. افزودن simtime به زمان میان رسیدن نمایی تولید شده در آرگومان اول در اینجا کاملاً ضروری نیست زیرا simtime اکنون صفر است، اما ما آن را به این صورت می نویسیم تا شکل کلی را نشان دهیم و تاکید کنیم که اولین آرگومان eventschedule (مطلق) است. زمان در آینده شبیه سازی شده زمانی که رویداد قرار است رخ دهد، نه فاصله زمانی از الان تا آن زمان. در فصل ۱ باید زمان رویدادهای غیرممکن را روی (در واقع ۱۰۳۰) تنظیم می‌کردیم، اما اکنون آنها را به سادگی از فهرست رویدادها حذف می‌کنیم و اطمینان می‌دهیم که نمی‌توان آنها را برای اتفاق بعدی انتخاب کرد. بنابراین، ما اصلاً یک رویداد خروج را در اینجا برنامه ریزی نمی کنیم. در ورود تابع رویداد که با استفاده از eventschedule برای برنامه‌ریزی رویداد ورود بعدی به روشی مشابه initmodel شروع می‌شود (در اینجا، افزودن simtime به زمان بین ورود نمایی تولید شده ضروری است زیرا simtime مثبت خواهد بود). سپس بررسی می کنیم که آیا سرور مشغول است یا خیر، با پرسیدن اینکه آیا لیست سرور حاوی یک رکورد (ساختگی) است یا خیر. این کار با بررسی اینکه [LISTSERVER]listsize برابر با ۱ است یا خیر انجام می شود. در این صورت، مشتری ورودی باید به انتهای صف بپیوندد، که با قرار دادن زمان ورود (مقدار ساعت فعلی، simtime) در اولین صف انجام می شود. ما مجبور نیستیم سرریز صف را در اینجا بررسی کنیم زیرا Simlib به صورت خودکار ذخیره سازی را به صورت پویا برای لیست ها در صورت نیاز اختصاص می دهد. از طرف دیگر، اگر سرور بیکار باشد، مشتری تاخیر ۰ را تجربه می کند که با فراخوانی sampst مشخص می شود. این لازم است حتی اگر تاخیر ۰ باشد، زیرا sampst تعداد مشاهدات را نیز ۱ افزایش می دهد. توجه داشته باشید که ما جریان EVENTDEPARTURE را به تولید زمان خدمات اختصاص می دهیم. خروج تابع رویداد، خالی بودن صف را با نگاه کردن به طول لیست صف که توسط simlib در [LISTQUEUE]listsize نگهداری می شود، بررسی می کند. اگر چنین است، سرور با حذف رکورد (ساختگی) از لیست LISTSERVER ، تنها اقدام مورد نیاز، بیکار می شود. ما اولین رکورد را در لیست حذف می کنیم، اما می توانیم آخرین رکورد را نیز حذف کنیم زیرا تنها یک رکورد در آنجا وجود دارد. از سوی دیگر، در صورت وجود صف، اولین مشتری از آن حذف می‌شود و زمان ورود آن مشتری در انتقال توسط listremove قرار می‌گیرد. بنابراین تأخیر در صف آن مشتری، انتقال simtime ۲ [۱] است که

در sampst محاسبه و محاسبه می‌شود و تعداد تأخیرهای مشاهده‌شده افزایش می‌یابد. اگر قرار بود شبیه‌سازی برای یک دوره طولانی از زمان شبیه‌سازی شده اجرا شود، ممکن است لازم باشد که هم simtime و هم انتقال نوع دو برابر شود تا از کاهش دقت در تفریق برای محاسبه تاخیر در صف جلوگیری شود.

در نهایت، تکمیل خدمات این مشتری با فراخوانی eventschedule زمان‌بندی می‌شود. توجه داشته باشید که دیگر نیازی به جابجایی صف به بالا نداریم، زیرا این کار به صورت داخلی توسط simlib و با استفاده از لیست‌های پیوندی انجام می‌شود. تابع تولید گزارش از قالب بندی خروجی استاندارد در outsampst برای اندازه گیری تاخیر در صف و outfi برای اندازه گیری تعداد در صف و استفاده استفاده می کند. توجه داشته باشید که قبل از فراخوانی outsampst و outfi هدرهای مختصری می نویسیم تا مبادا گزارش را کمی خواناتر کنیم. از قالب بندی کلی نتایج عددی برای جلوگیری از امکان سرریز شدن به دلیل عرض فیلد استفاده می کنیم. ما تمام ویژگی های معیارهای خروجی، یعنی میانگین، حداکثر و حداقل، و همچنین تعداد مشاهدات متغیرهای زمان گسسته مورد استفاده توسط Sampst را دریافت می کنیم. همچنین مقدار ساعت نهایی را به عنوان چک می نویسیم. در حالی که استفاده از simlib کدگذاری این مدل را به طور قابل توجهی ساده کرد، ارزش چنین بسته ای در مدل های پیچیده با ساختارهای فهرست غنی تر آشکارتر می شود. چنین مدل هایی در مرحله بعدی، در ثانیه های ۵.۲ تا ۷.۲ در نظر گرفته می شوند.