گزارش پروژه

محمد مهدی صفت زاده

مهدی ناصری مجد

مقدمه

این پروژه یک شبیه ساز برای زمان بند CPU یا همان CPU scheduling هست که می تواند شش الگوریتم FIFO، Non-Preemptive SJF، Preemptive SJF، Non-Preemptive Priority، Preemptive Priority و Round Robin را برای زمان بندی CPU شبیه سازی کند.

نحوه اجرای شبیه ساز

در این شبیه ساز هر فرآیند دارای پنج پارامتر است که اولی ID آن فرآیند است که آن فرآیند را از بقیه فرآیند ها متمایز می کند، دومی زمان ورود فرآیند به صف ready است(همان Arrival Time)، سومی اولویت اجرای فرآیند نسبت به بقیه فرآیند ها است و چهارمین پارامتر آن مدت زمانی است که آن فرآیند باید توسط Cpu اجرا شود (همان Burst Time). این شبیه ساز به گونه ای پیاده سازی شده است که کاربر می تواند بنا بر انتخاب خود یا بوسیله یک command line برنامه را اجرا کند و یا به وسیله یک GUI یا همان یک رابط گرافیکی برنامه را اجرا کند.

نحوه اجرا بوسیله GUI (رابط گرافیکی)

با زدن دکمه چرخ دنده در کنار دکمه Generate The Processes در صفحه اصلی، تعداد فرآیند هایی که می خواهد توسط الگوریتم ها شبیه سازی بشود را انتخاب کند و همچنین می تواند برای پارامترهایی که هر فرآیند داراست، بازه هایی را تعیین کند که در این صورت شبیه ساز با استفاده از کلاس Request Generator مقادیری تصادفی برای هر پارامتر یک فرآیند در آن بازه هایی که کاربر مشخص کرده است انتخاب می کند و

وقتی تعداد فرآیند های تولید شده به آن عدد مورد نظر کاربر رسید، آنگاه یک فایل csv که شامل همه ی فرآیندهای تولید شده است به همراه تمام پارامترهای هر فرآیند به کاربر تحویل می دهد. همچنین اگر کاربر نخواهد تعداد فرآیندها و یا هرکدام از بازه های پارامترها را مشخص کند، این شبیه ساز شامل بازه هایی پیش فرض است که به وسیله آنها آن فایل csv را تولید می کند. پس از اینکه فرآیندها تولید شدند حال کاربر اگر در صفحه اصلی گزینه start simulation را بزند، آنگاه تمامی شش الگوریتم شبیه سازی می شوند، اما کاربر می تواند با زدن چرخ دنده کنار دکمه start simulation انتخاب کند که می خواهد کدام الگوریتم ها شبیه سازی بشوند. حال پس از اینکه شبیه سازی تمام شد، شبیه ساز در یک جدول خروجی هایی را برای هر الگوریتم نشان می دهد که این جدول شامل ستون های زیر است:

1. Algorithm Name: نام الگوریتم هایی که شبیه سازی شده اند.
2. Number of Processes: تعداد فرآیندهایی که شبیه سازی شده اند.
3. Time of Simulation: مدت زمان سپری شده برای شبیه سازی.
4. Throughput: تعداد فرآیندهایی که cpu در واحد زمان انجام داده اند.
5. CPU Utilization: میزان بهره وری CPU
6. Avg Waiting Time: میانگین زمانی که فرآیند در صف ready برای اجرا شدن توسط CPU منتظر بوده است.
7. Avg Turnaround Time: میانگین مدت زمانی است که برای هر فرآیند از لحظه ورود طول کشیده است که کارش توسط cpu پایان یابد.
8. Avg Response Time: میانگین مدت زمانی است که هر فرآیند از زمان ورود به صف صبر کرده است تا برای اولین توسط cpu اجرا شود.

پس از اینکه این جدول به کاربر نمایش داده شد، کار شبیه سازی پایان می یابد و با کاربر با زدن دکمه Home می تواند دوباره مراحل شبیه سازی را شروع کند. همچنین در صفحه اصلی یک دکمه با لوگو i می باشد که با زدن آن می توانید نام برنامه نویسان این شبیه ساز را ببینید.

نحوه اجرا بوسیله CLI

در شبیه سازی بوسیله CLI یک سری دستورات برای این شبیه ساز نوشته شده است که با اجرای هر یک از این دستورات کاربر می تواند دقیقا به همان نتایجی برسد که در GUI می رسید. در این رابط، یک دستور help– وجود دارد که به کاربر می تواند راهنمایی کند که دستورات این برنامه چگونه است.

عملکرد کلاس های پیاده سازی شده

در اینجا به بررسی هر یک از کلاس های این شبیه ساز که در پکیج CPU Scheduling هستند می پردازیم. پکیج Controller پکیجی هستند که در آن کلاس های کنترلی برای interface است. کلاس Command Line Interface برای کنترل کردن کردن رابط Command Line است و باقی کلاس های داخل پکیج Controller برای کنترل کردن هر یک از صفحات رابط کاربری گرافیکی یا GUI است.

حال به بررسی پکیج model می پردازیم که کلاس های داخل آن کلاس های اصلی شبیه ساز هستند که هسته شبیه ساز را تشکیل می دهند. در زیر به بررسی هر یک از کلاس ها می پردازیم:

1. Process: این کلاس که فرآیند ها را شبیه سازی می کند، شامل چند پراپرتی است که فرآیندها آن ها را شامل می شوند و این پراپرتی ها در واقع همان پارامترهایی هستند که در بالاتر گفتیم که هر فرآیند آن ها را شامل می شد. همچنین در اینجا متدهایی وجود دارند که یک سری پارامترهای خاص هر فرآیند را مانند کل مدت زمانی که منتظر بوده است، و یا مدت زمانی که از وقتی یک فرآیند برای اولین بار وارد صف شده است را تا وقتی که کارش با cpu تمام شده است را محاسبه می کند و یا نظایر این ها. برای

مثال متد execute کارش این است که متد زمانی که هر فرآیند توسط cpu اجرا شده است را بررسی کند و اگر تمام شده باشد زمانی که اجرای فرآیند تمام شده است را ثبت می کند. همچنین در این کلاس یک سری کلاس های هستند که از اینترفیس Comparator ارث بری کرده اند. برای مثال کلاس Arrival Time Comparator دو فرآیند را بر اساس زمانی که وارد صف شده اند مقایسه می کند. بقیه کلاس ها نیز بدین صورت بر اساس هر یک از پارامترهای مختص فرآیند ها، دو فرآیند را با هم مقایسه می کنند.

2. Scheduling Algorithm: این کلاس همان شش الگوریتم برای زمان بندی CPU را که در بالا به آن ها اشاره کردیم پیاده سازی می کند. در اینجا صف Ready به وسیله ساختمان داده صف اولویت دار پیاده سازی شده است. سازنده این کلاس به عنوان پارامتر اسم الگوریتم مد نظر را می گیرد و سپس بر اساس آن اولویت در صف را پیاده سازی می کند که این الویت بر اساس همان کلاس های مقایسه ای هستند که در کلاس Process پیاده سازی شده اند. همچنین اگر الگوریتم مد نظر Round Robin باشد، آنگاه سازنده آن می تواند پارامتر دومی که همان Quantum time است را بگیرد. همچنین نحوه اینکه در هر لحظه چه فرآیندی برای اجرا انتخاب شود توسط متد Select process و بر اساس همان اولویتی که در صف اولویت ما پیاده سازی کردیم مشخص می شود. پارامتر های خروجی مانند میانگین مدت زمان انتظار نیز در این کلاس محاسبه می شوند.
3. Request Generator: این کلاس دو کار مهم انجام می دهد. با استفاده از متد Generate Process Data این کلاس بر اساس مقادیر پیش فرض و یا مقادیری که کاربر مشخص کرده است یک سری اعداد تصادفی می سازد و به وسیله آنها فرآیندهایی را می سازد که شامل همان پارامترهایی هستند که در بالا گفته شد. سپس بعد از اینکه به تعداد مشخصی فرآیند ساخت، یک فایل csv می سازد که شامل همه ی فرآیند ها می باشد. متد مهم بعدی متد read Process Data هست که این متد یک صف اولویت دار را بر می گرداند. به وسیله این متد، تمام فرآیندهایی را که در فایلcsv ساخته شده

هستند را می خوانیم و هر از فرآیند ساخته شده یک شی از کلاس Process می سازیم و هر پارامتر فرآیندها را به پراپرتی معادل در کلاس Process نسبت می دهیم و سپس آن فرآیند را وارد یک صف می کنیم. این کار آنقدر انجام می شود تا تمام فرآیندهای داخل فایل csv خوانده شده و به صف وارد شوند.

4. Operating System: این کلاس در واقع شروع شبیه ساز ما است و متد های داخل آن برای راه اندازی شبیه ساز هستند. همچنین متد check process هر لحظه چک می کند و بر اساس پراپرتی time (که مدت زمان سپری شده از شبیه ساز را می گوید) هر کدام از پروسه ها که زمان ورودشان به صف رسیده است را مشخص می کند و سپس ان ها را داخل صف Ready می گذارد.

تحلیل الگوریتم ها

در اینجا به بررسی هر یک از الگوریتم ها می پردازیم:

1. FIFO: این الگوریتم بدین صورت است که هر فرآیندی که زودتر وارد صف بشود زودتر اجرا می شود حتی اگر زمان اجرای طولانی داشته باشد. یکی از نکات منفی این الگوریتم این است که میانگین مدت زمان انتظار فرآیندها می تواند بسیار طولانی باشد. مثلا اگر فرض کنیم که یک فرآیندی که CPU را برای مدت زمان زیادی نیاز دارد زودتر از بقیه فرآیندها وارد صف بشود، آنگاه مدت زمان انتظار فرآیندهای کوچک نر خیلی بالا می رود از آنجا کخ باید صبر کنند که آن فرآیند اول کارش تمام بشود. پس می توان گفت که الگوریتم بهینه ای نیست.
2. Non-Preemptive SJF: این الگوریتم بدین صورت است که چک می کند و فرآیندی را که از بقیه فرآیندها کمتر به CPU نیاز دارد را انتخاب می کند، سپس از آنجا که این الگوریتم Non-Preemptive است، پس باید صبر کند تا آن فرآیند کارش تمام شود و سپس دوباره کوچک ترین

فرآیند از نظر burst time را انتخاب کند. در این الگوریتم چون کوچک تزین کارها زودتر انجام می شوند میانگین مدت زمان انتظار کمتر از FIFO است.

3. Preemptive SJF: دقیقا مانند Non-Preemptive SJF است ولی با این تفاوت که باید هر لحظه چک شود و اگر فرآیندی مدت زمان اجرای کمتری نسبت به فرآیند در حال اجرا داشت، در این صورت CPU را در اختیار آن فرآیند بگذارد.
4. Non-Preemptive Priority: در این الگوریتم بر اساس اولویتی که فرآیند دارد فرآیند به CPU داده می شود. مشکل این الگوریتم این است که از آنجا که الگوریتم ها در هر لحظه وارد می شوند، اگر فرآیندی اولویت پایینی داشته باشد ممکن است هیچ وقن نوبتش مشود و به اصطلاح Starvation رخ دهد. همچنین این الگوریتم وقتی فرآیندی CPU را گرفت دیگر نمی توان CPU را از آن گرفت تا وقتی که کارش تمام بشود. در این الگوریتم نیز بر اساس اولویت ها ممکن است میانگین مدت زمان انتظار تغییر کند.
5. Preemptive Priority: این الگوریتم نیز مانند الگوریتم Non-Preemptive Priority است با این تفاوت که باید هر لحظه چک شود و اگر فرآیندی وارد صف شد که اولویتش از فرآیند در حال اجرا بیشتر بود باید CPU را به ان فرآیند اختصاص دهیم.
6. Round Robin: این الگوریتم مانند FIFO است ولی با این تفاوت که یک پارامتری به اسم Quantum Time دارد. هر فرآیندی که زودتر به صف برسد به اندازه این پارامتر اجرا می شود و سپس CPU را از آن می گیریم و به فرآیند بعدی می دهیم و فرآیند بعدی نیز به همین مقدار اجرا می شود. این پرئسه آنقدر اجرا می شود تا دوباره به فرآیند اول می رسد، حال دوباره فرآیند اول به اندازه پارامتر اجرا می شود سپس نوبت به بقیه فرآیندها می رسد. مقدار میانگین مدت زمان انتظار هر فرآیند در این الگوریتم به پارامتر کوانتوم بستگی دارد. اگر مقدار آن خیلی بزرگ باشد آنگاه مقدار این میانگین نیز زیاد می شود، همجنین اگر کوانتوم خیلی کم باشد آنگاه Context

Switching های زیادی رخ می دهد که باعث کندی عملکرد می شود. پس باید یک کوانتوم بهینه ای را انتخاب کرد که بتوان بهترین عملکرد را گرفت.