پروژه شبیه‌سازی کامپیوتری

دکتر بردیا صفایی

دانشکده مهندسی کامپیوتر

دانشگاه صنعتی شریف

نیم‌سال اول ۰۱-۰۲

امیرحسین باقری - ۹۸۱۰۵۶۲۱

محمدرضا مفیضی - ۹۸۱۰۶۰۵۹

۱ مقدمه

در این گزارش نحوه پیاده‌سازی و نتایج شبیه‌سازی یک زمان‌بند CPU را شرح می‌دهیم. [۱] در این سیستم پردازه‌ها به‌صورت تسک‌هایی درنظر گرفته شده‌اند که بین صف‌های مختلف جابه‌جا می‌شوند. توضیحات پیاده‌سازی در ادامه آمده است.

۲ پیاده‌سازی

۱.۲ کلاس Task

کلاس Task به عنوان پردازه‌هایی که در صف‌ها قرار می‌گیرند تا به CPU برسند طراحی شده است. نمونه‌های این کلاس مقادیر X و Y و Z را برای تولید نرخ ورود (یا همان زمان inter-arrival)، زمان سرویس و اندازه زمان انقضا ورودی می‌گیرند. همچنین هنگام ساخت به‌صورت تصادفی و با احتمال‌های گفته‌شده یک اولویت برای تسک ساخته‌شده درنظر گرفته می‌شود.

۲.۲ کلاس JobCreator

کلاس JobCreator همان‌طور که در دستور پروژه آمده است تسک‌های ورودی سیستم را با مشخصات گفته‌شده تولید می‌کند و آن‌ها را در Priority Queue لایه اول قرار می‌دهد.

۳.۲ صف‌های لایه دوم

برای لایه دوم صف‌های First Come First Serve و Round Robin به‌ترتیب در کلاس‌های FCFS و RR پیاده‌شده اند. صف‌های RR مقدار کوانتوم تایم را ورودی می‌گیرند. تابع `insert` همواره تسک‌های جدید را به اول لیست اضافه می‌کند تا تسک با اولیت بیشتر (یا تسکی که زودتر آمده) آخر لیست در دسترس باشد. تابع `pop_task` برای حذف تسک از لیست و `get_task` هم برای گرفتن اولین تسک از لیست استفاده می‌شود. توابع دیگری هم برای استفاده در صف‌ها پیاده‌سازی شده‌اند.

۴.۲ کلاس Processor

کلاس Processor برای شبیه‌سازی پردازنده طراحی شده است. این کلاس تعداد پردازه‌ها، زمان شبیه‌سازی، صف‌های لایه دوم، و بقیه پارامترها را به‌عنوان ورودی می‌گیرد. تابع `job_loader` هنگامی که محموع تعداد تسک‌ها در صف‌ها از k کمتر می‌شود k تسک با بالاترین اولویت را به اولین صف لایه دوم منتقل می‌کند.

فرایند اصلی شبیه‌سازی در تابع `dispatcher` انجام می‌شود. در این متد یکی از صف‌های لایه دوم به‌صورت تصادفی با احتمال داده‌شده انتخاب می‌شود. سپس باتوجه به سیاست صف یک تسک انتخاب می‌شود و در پردازنده اجرا می‌شود. فرایند اجراشدن یک تسک در تابع `process` انجام می‌شود. به این صورت که واحد زمانی یک واحد به جلو برده می‌شود. سپس بررسی می‌شود که اگر تسک به‌اندازه کوانتوم زمانی در صف فعلی سپری کرده است به صف بعدی منتقل شود. همچنین اگر زمان سرویس آن به پایان رسیده است از صف حذف می‌شود.

۳ نتایج

شبیه‌سازی را به‌ازای ورودی‌ها و پارامترهای مختلف انجام می‌دهیم. ورودی‌های درنظر گرفته‌شده در جدول ۱ قرار داده شده‌اند.

[۱] کدهای پیاده‌سازی و نتایج کامل شبیه‌سازی در فایل ژوپیتری که همراه با این گزارش ضمیمه شده آمده است.

Input	X	Y	Z	Num Processes	Len of Simulation	K	T1	T2
1	8	8	32	100	800	10	2	4
2	8	4	32	50	800	10	2	4
3	8	6	16	100	800	10	2	4
4	8	8	32	100	800	10	6	8

جدول ۱: ورودی‌های شبیه‌سازی

۱.۳ طول صف‌ها

نمودار طول صف‌های لایه اول و دوم در شکل ۱ آمده است.
میانگین طول صف‌ها به‌ازای ورودی‌های مختلف نیز در جدول ۲ آمده است.

Input	RR-T1	RR-T2	FSFS	Priority Queue
1	0.30	2.03	0.87	0.12
2	0.25	0.33	0.02	0.02
3	0.26	0.67	0.13	0.06
4	0.98	2.23	0	0.12

جدول ۲: میانگین طول صف‌های مختلف به‌ازای ورودی‌های متفاوت

۲.۳ میانگین زمان صرف‌شده در صف‌ها

نمودار زمان صرف‌شده در صف‌ها در شکل ۲ آمده است.

۳.۳ میزان بهره‌وری پردازنده

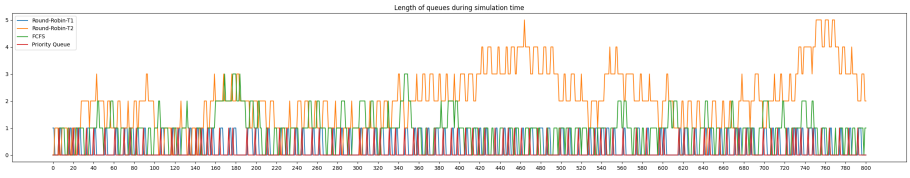
میزان بهره‌وری پردازنده در ورودی اول 100%، در ورودی دوم 78.0%، در ورودی سوم 80.2% و در ورودی چهارم 98.5% است.

۴.۳ بهبود میانگین زمان صرف‌شده در صف‌ها

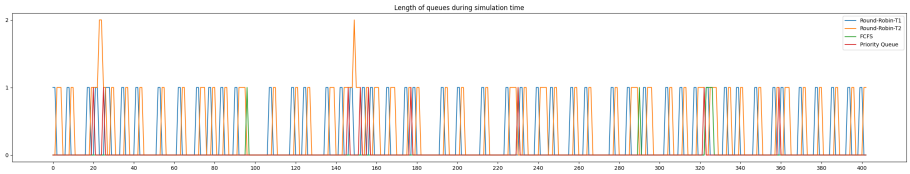
از روی مشاهدات و همچنین با استناد به این‌که هرچه مقدار کوانتوم‌ها کمتر باشد تسک‌ها در صف‌های RR (مخصوصا RR-T1) کمتر باقی می‌مانند و می‌دانیم احتمال انتخاب این صف‌ها توسط dispatcher بیشتر است، بنابراین تسک‌ها کمتر در صف RR و بیشتر در FCFS باقی می‌مانند و به‌طور میانگین هم کمتر زمان خود را در صف سپری می‌کنند.

۵.۳ درصد پردازه‌های منقضی‌شده

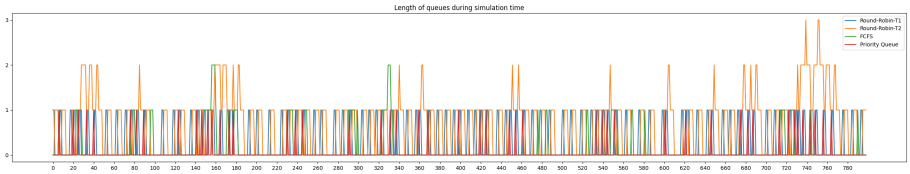
درصد پردازه‌های منقضی‌شده در ورودی اول 20.0%، در ورودی دوم 0.0%، در ورودی سوم 1.0% و در ورودی چهارم 28.0% است.



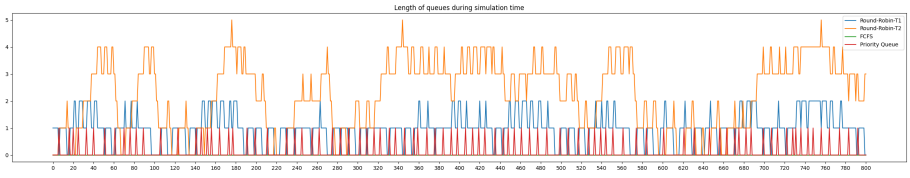
(آ) ورودی اول



(ب) ورودی دوم

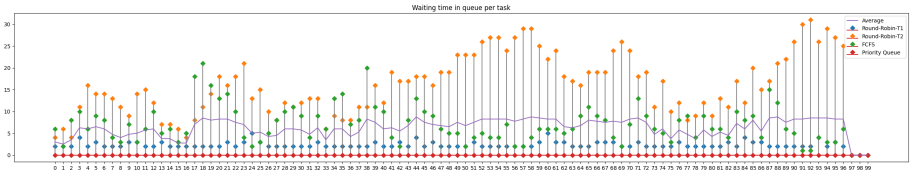


(ج) ورودی سوم

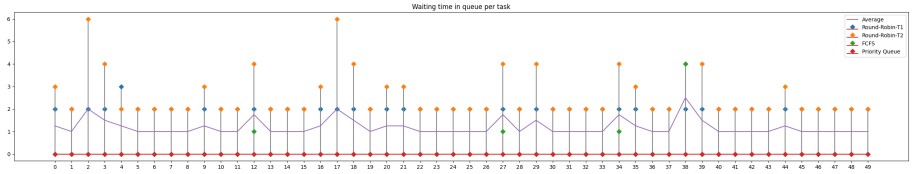


(د) ورودی چهارم

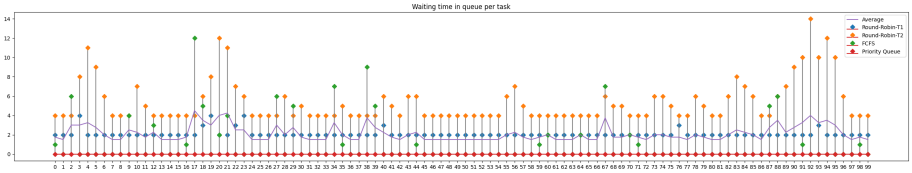
شکل ۱: طول صف‌های لایه اول و دوم به‌ازای ورودی‌های مختلف



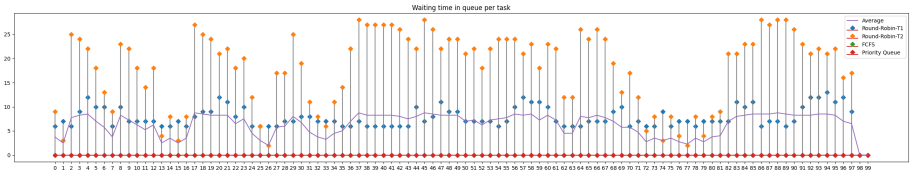
(آ) ورودی اول



(ب) ورودی دوم



(ج) ورودی سوم



(د) ورودی چهارم

شکل ۲: زمان صرف‌شده در صف‌های مختلف به‌ازای ورودی‌های مختلف