CS 342 – Operating Systems

Project 2

Maryam Shahid

21801344

Section 03

March 27, 2021

1. Introduction

This report consists of an experiment performed on a scheduler which makes use of threads and synchronization. The experiment aims to answer the following question – what is the average waiting time for each thread when we use the FCFS, SJF, PRIO and VRUNTIME algorithm on an exponentially distributed burst length and interarrival time (with a constant mean).

2. Experiment Result and Interpreatations

The experiment conducted aimed to calculate and compare the average waiting time for each thread using the following algorithms - FCFS, SJF, PRIO and VRUNTIME, on an exponentially distributed burst length and interarrival time (with a constant mean). Values used for the experiment were kept constant as N = 4, Bcount = 4, minB = 100, avgB = 200, minA = 1000, avgA = 1500. The waiting for every burst for each algorithm is recorded in the table below.

T-Index	B-Count	FCFS (ms)	SJF (ms)	PRIO (ms)	VRUNTIME
1	1	1797	1770	195	0
	2	3053	230	0	923
	3	3172	350	142	1855
	4	3644	505	215	3019
2	1	0	0	233	216
	2	205	208	169	1253
	3	1115	1006	450	1997
	4	1351	147	448	3165
3	1	1935	480	513	491
	2	3915	607	699	1135
	3	4466	1007	513	2409
	4	4642	107	699	3759
4	1	4830	399	958	707
	2	304	1904	1929	1368
	3	728	1488	1348	3320

	4	1206	237	1085	3892

Table 1 – Wait time (ms) for each burst using various scheduling algorithms

T-Index	B-Count	B-Length	SJF (ms)
1	1	580	1770
	2	114	230
	3	201	350
	4	266	505
2	1	100	0
	2	105	208
	3	402	1006
	4	145	147
3	1	231	480
	2	395	607
	3	440	1007
	4	129	107
4	1	203	399
	2	577	1904
	3	451	1488
	4	157	237

Table 2 – Burst length of each burst generated – for SJF

T-Index	B-Count	vruntime	VRUNTIME
1	1	0	0
	2	212	923
	3	419	1855
	4	556	3019
2	1	0	216

	2	350	1253
	3	495	1997
	4	1024	3165
3	1	0	491
	2	339	1135
	3	519	2409
	4	1487	3759
4	1	0	707
	2	402	1368
	3	1317	3320
	4	2141	3892

Table 3 – vruntime of each burst generated – for VRUNTIME

- **FCFS**: In the first come, first serve scheduling algorithm, the first burst to be generated is scheduled and executed first. However, since thread generation is non-deterministic, bursts are generated in a random order and added to the ready queue in the same way. As noted in table 1, the first burst to be generated was thread 2, burst count 1. Thus, it has 0 ms wait time. The second burst to be generated and added to run queue was thread 2, burst count 2 – as it has a waiting of 204 ms., and so on. The last thread to be generated and added to the queue, and thus the thread with the largest wait time was thread 4, burst count 1, and it has a wait time of 4830 ms.
- **SJF**: In shortest job first algorithm, the burst with the shortest burst length (calculated by random and exponential distribution of the mean specified above is simulated first. Table 2 shows the burst length of each burst against its wait time. As expected, the burst length with the shortest burst length is scheduled first. In the experiment above, thread 2 burst count 1 has the shorted burst length – 100, thus it is executed first and has a wait time of 0 ms. Thread 1 burst count1 has the largest burst length of 580 and thus has the highest wait time – 1770 ms. In the experiment above, the bursts of a particular thread are schedules in the order they are generated and this since generation is non deterministic, the burst with the shortest

burst length is chosen from only the earliest generated bursts. Due to this, the result may have some inconsistencies.

- **PRIO**: Priority based scheduling algorithm assigns each thread i with i priority, where the smaller the number, the higher the priority. The burst with the highest priority is selected for simulation. As shown in table 1, the bursts generated by thread 1 have the highest priority and the lowest waiting time. This is followed by thread 2 and so on. However, since the bursts are not reordered, and instead served in the order they are added to the run queue, a thread with a higher priority may be added to the run queue after a thread with a lower priority has already been simulated, thus the specific thread will have a higher waiting time. This is also portrayed in table 1.
- **VRUNTIME**: In virtual runtime scheduling, each thread is associated with a virtual runtime which is advanced by t(0.7 + 0.3i) where thread i runs t ms in cpu, and the thread with the smallest virtual runtime is selected for simulation. Table 4 displays the virtual runtime against wait time of each burst. As noted, all threads initially have virtual runtime of zero, and is advanced by its time in the CPU. Thus, the first burst of each thread has the shortest wait time, whereas the last burst of each thread has the highest wait time. For example, for four bursts of thread 1, its virtual runtime goes from 0, 212, 419, 556. And thus, wait time increases from 0, 923, 1855, 3019.

3. Conclusions

4. T-Index	FCFS	SJF	PRIO	VRUNTIME
1	667	713	1325	1450
2	2916	340	1621	1657
3	3739	550	2016	1948
4	1767	1007	2234	2321
Total	2266	652	1982	1844

Table 4 – average wait time of each thread using various algorithms in ms

The table above records the average wait time of each thread using various scheduling algorithms. As noted, SJF algorithm has the shortest waiting time for each thread. This is because, the shortest job in the run queue is executed first. Thus, minimizing the convoy effect and reducing waiting time of each burst present in the run queue. This is followed by the VRUNTIME scheduling algorithm. Since virtual runtime is based on the threads' index and burst length, this algorithm has attributes similar to SJF and PRIO, both. The first burst of each thread has the shortest virtual runtime and is executed first, followed by the rest. Thus, all threads have very similar average wait time. After this, PRIO algorithm takes the line. Since, for each thread i, the smallest i has the highest priority, thread 1 has the smallest average wait time, followed by thread 2, 3 and 4 respectively. Lastly, FCFS has the largest average wait time of all threads. Since bursts are generated randomly, it can be inferred from table 4 that thread 1 was placed in the run queue first, followed by thread 4, then 2 and then 3. This is reflected in their average wait times.

In conclusion, SJF scheduling algorithm has the lowest total wait time, followed by VRUNTIME, PRIO and FCFS respectively.

5. References

[1] https://stackoverflow.com/questions/34558230/generating-random-numbers-of-exponential-distribution

[2] https://cboard.cprogramming.com/c-programming/140392-reading-two-numbers-same-line-file.html

[3] Operating System Concepts, 9th edition, Silberschatz et al. Wiley