CS330 : Assignment 2

Siddharth Agrawal (150716)
K Karthikeyan (150311)
Shubham Kumar Bharti (150702)
Shubhojyoti Nath (150708)

October 26, 2017

Part I:

Table 1: Batch 1 Data

No.	Algorithm	CPU Utilization	Average Wait Time
1	Default NachOS	56.214733%	24464
2	Shortest CPU Burst First	56.214733%	24464
3	Round Robin (Time Quantum : 31)	65.580116%	106229
4	Round Robin (Time Quantum : 62)	61.699043%	81907
5	Round Robin (Time Quantum : 93)	60.515781%	76763
6	Round Robin (Min. Quantum : 20)	71.755157%	143174
7	UNIX Scheduler (Time Quantum : 31)	66.102104%	105811
8	UNIX Scheduler (Time Quantum : 62)	62.031017%	81988
9	UNIX Scheduler (Time Quantum : 93)	60.382980%	77415
10	UNIX Scheduler (Min. Quantum : 20)	72.567963%	142492

Table 2: Batch 2 Data

No.	Algorithm	CPU Utilization	Average Wait Time
1	Default NachOS	82.866562%	24410
2	Shortest CPU Burst First	82.866562%	24410
3	Round Robin (Time Quantum : 31)	89.966286%	105370
4	Round Robin (Time Quantum : 62)	89.172951%	80388
5	Round Robin (Time Quantum : 93)	87.424133%	76454
6	Round Robin (Min. Quantum : 20)	92.221581%	142987
7	UNIX Scheduler (Time Quantum : 31)	89.312660%	106574
8	UNIX Scheduler (Time Quantum : 62)	89.537804%	82234
9	UNIX Scheduler (Time Quantum : 93)	86.887054%	77939
10	UNIX Scheduler (Min. Quantum : 20)	92.961815%	142003

Table 3: Batch 3 Data

No.	Algorithm	CPU Utilization	Average Wait Time
1	Default NachOS	94.724182%	24410
2	Shortest CPU Burst First	94.724182%	24410
3	Round Robin (Time Quantum : 31)	98.815773%	105307
4	Round Robin (Time Quantum : 62)	99.319725%	80260
5	Round Robin (Time Quantum : 93)	99.016510%	76517
6	Round Robin (Min. Quantum : 20)	98.847183%	163024
7	UNIX Scheduler (Time Quantum : 31)	98.02366%	106496
8	UNIX Scheduler (Time Quantum : 62)	99.070992%	81635
9	UNIX Scheduler (Time Quantum : 93)	98.491173%	77974
10	UNIX Scheduler (Min. Quantum : 20)	99.351433%	142012

Table 4: Batch 4 Data

No.	Algorithm	CPU Utilization	Average Wait Time
1	Default NachOS	99.864647%	36632
2	Shortest CPU Burst First	99.864647%	36632
3	Round Robin (Time Quantum : 31)	99.899818%	107646
4	Round Robin (Time Quantum : 62)	99.876007%	86806
5	Round Robin (Time Quantum : 93)	99.868073%	81425
6	Round Robin (Min. Quantum : 20)	99.919846%	145849
7	UNIX Scheduler (Time Quantum : 31)	99.899926%	10702
8	UNIX Scheduler (Time Quantum : 62)	99.876007%	86866
9	UNIX Scheduler (Time Quantum : 93)	99.868088%	81519
10	UNIX Scheduler (Min. Quantum : 20)	99.919853%	145902

Method for selecting the quantum (Q4) yielding max CPU Utilization for Round-Robin Algorithm:

We ran the program for various values of time quantum (values = 24, 34, 44 ,.... ,124). The CPU utilization came out to be decreasing with increasing value of quanta. So we ran for 20 ticks which is the lower bound we are allowed to go and it turned out to work best at 20 ticks (yielded max CPU Utilization).

Explanation for Q4=20:

For different values of the quantum, I/O burst time remains same.
But, for smaller quantum values, there will be more *context* and *mode* switches which adds to total CPU Burst time. Thus the total CPU Burst Time increases.
Hence CPU Utilization is higher - smaller the value of quanta.

Downside of the above method for selecting Q4:

With smaller quantum values, we spend most of the time in *context* and *mode* switches rather than computing the original process. So, this method is not a good method for choosing the ideal value of the time quantum. Instead we could have used average process completion time (which was observed to be large for small quantum values).

Note: *For UNIX Scheduling Algorithm, we observe almost same behaviour (for CPU Utilization) as in Round-Robin. This result can be explained in the same way as above.*

Quantum Values:

$$
\begin{aligned}
\mathrm{A} &= 124\\
\mathrm{Q1} &= 31\\
\mathrm{Q2} &= 62\\
\mathrm{Q3} &= 93\\
\mathrm{Q4} &= 20
\end{aligned}
$$

Part II:

The default NachOS Scheduling algorithm 1 runs the processes in non-preemptive first come first serve basis. Hence each process in batch 5 will yield after running a outer loop. The order of running process are thus fixed and its surely less efficient way for getting average wait time. The shortest burst first algorithm would be near ideal choice which is not happening here. However the second algorithm (SJF) tries to estimate next burst time of process using exponential averaging technique and runs the one with shortest estimated burst time. Thus it will perform better in reducing average waiting time.

Table 5: Batch 5 Data

No.	Algorithm	CPU Utilization	Average Wait Time
1	Default NachOS	99.822090 %	55505
2	Shortest CPU Burst First	99.822090 %	40250

Explanation:

The default NachOS Scheduling algorithm 1 runs the processes in non-preemptive first come first serve basis. Hence each process in batch 5 will yield after running a outer loop. The order of running process are thus fixed and its surely less efficient way for getting average wait time. The shortest burst first algorithm would be near ideal choice which is not happening here. However the second algorithm (SJF) tries to estimate next burst time of process using exponential averaging technique and runs the one with shortest estimated burst time. Thus it will perform better in reducing average waiting time.

Part III:

Table 6: CPU Burst Estimation Error Ratio

No.	Batch	Outer Bound = 4	Outer Bound = 8
1	Batch1	0.703194	0.474806
2	Batch2	0.743443	0.486972
3	Batch3	0.742332	0.482975
4	Batch4	1.0	1.0
5	Batch5	0.693535	0.369800

Explanation:

The overall CPU burst estimation error for non-preemptive shortest burst first algorithm is given below in table with required ratios.
On increasing the value of OUTER_BOUND from 4 to 8, the value of the error drops by good margin. Higher are the number of iterations, better is the algorithm able to better predict the correct next CPU Burst using exponential averaging technique. Thus we are able to see such good decrease in error ratio.
In the case of the testloop3 we are not able to witness this improvement as it does not yields between iterations and thus complete individual programs runs in a single CPU burst and the scheduling algorithm doesn't get to choose between the different threads and thus does not affects anything (Hence the error ratio = 1.0).
Note: *testloop has syscall_wrapper_PrintInt which make the thread to yield because of the I/O instruction and thus its behavior is similar to other programs with yield in their loop iteration.*

Part IV:

The completion time statistics data is given below in table.

Table 7: Thread Completion Time(TCT) data for Round-Robin and Unix Scheduler

	Round-Robin	Unix Scheduler
Maximum TCT	163386	135570
Minimum TCT	159704	84668
Average TCT	162034.20	118798.40
Variance	1885300.76	303284016.64

Explanation:

Round Robin algorithm executes the the processes one by one in circular fashion using a 100 tick time quantum.
Unix schedular model uses CPUUsage value to decide priorities of processes and schedules them on basis of them.
A process having a very large priority has a high probability of getting scheduled early while the process with very less priority may get scheduled at a very later time. The former program terminates early while the latter ends at a very late stage. Thus there is a significant difference between the minimum and maximum waiting times. Thus the Completion Time for the threads vary very much resulting in a high variance.
Round robin handles all processes with equal priority by scheduling them in a circular fashion. Thus threads exit nearly after the same time interval. Hence the max-min value is small for Round-Robin scheduler when compared to Unix Scheduler model. Since Completion Times are almost same for the threads, hence variance is significantly less.