CS330 Assignment 2
Group 8

Archit Rathore 12152
Lucky Sahani 12383
Hardik Bansal 12274

Course Instructor - Prof. Mainak Choudhary

PART 0

The output of testloop.c when we run nonpreemptive scheduling algorithm is:

Max CPU Burst :2068
Min CPU Burst :10
Mean CPU Burst :124
CPU Burst count :725
CPU utilization: 0.562833
Total CPU busy time: 90270
Number of context switches through yield or preemption: 0
Number of non-preemptive context switches: 725
Total time for which the ready queue is empty: 70115
Total Wait time in ready queue: 244093
Average Wait time in ready queue: 22190.27
Total Number of Ticks: 160385
Idle ticks :70115
System Ticks : 7750
User Ticks :82520

Let A be the average CPU burst time . From above, A=124 . So,
Q1=A/4=31
Q2=A/2=62
Q3=3A/4=93

The minimum value of the quantum that offers the maximum achievable CPU utilization for testloop is Q4. We can infer that CPU utilization is constantly decreasing and is maximum at the start. Therefore, Q4=20.

(When we decrease the time quantum, the probability of I/O overlapping with CPU burst increases and hence the CPU is utilized more effectively. The CPU utilization is maximum if the scheduling quantum is equal to CPU burst and when the ready queue always has a “ready to run” process at the end of the scheduling quantum.)

PART-A

Batch-1

Scheduling Algorithm	CPU Utilization	Quantum
Non-preemptive FCFS	0.562833	No Effect
Non-preemptive SJF	0.562833	No Effect
Round-robin	0.660031	31
Round-robin	0.621160	62
Round-robin	0.605791	93
Round-robin	0.724455	20
UNIX scheduler	0.660031	31
UNIX scheduler	0.621951	62
UNIX scheduler	0.605791	93
UNIX scheduler	0.724455	20

Scheduling Algorithm	Average Waiting Time	Quantum
Non-preemptive FCFS	22190.27	No Effect
Non-preemptive SJF	22190.27	No Effect
Round-robin	95076.09	31
Round-robin	73092.18	62
Round-robin	69932.36	93
Round-robin	128748.36	20

UNIX scheduler	95076.09	31
UNIX scheduler	73420.09	62
UNIX scheduler	69932.36	93
UNIX scheduler	128748.36	20

Batch-2

Scheduling Algorithm	CPU Utilization	Quantum
Non-preemptive FCFS	0.829748	No Effect
Non-preemptive SJF	0.829748	No Effect
Round-robin	0.905610	31
Round-robin	0.891508	62
Round-robin	0.877934	93
Round-robin	0.930628	20
UNIX scheduler	0.905610	31
UNIX scheduler	0.887449	62
UNIX scheduler	0.877934	93
UNIX scheduler	0.930628	20

Scheduling Algorithm	Average Waiting Time	Quantum
Non-preemptive FCFS	22141.18	No Effect
Non-preemptive SJF	22141.18	No Effect
Round-robin	95077.55	31
Round-robin	73233.82	62
Round-robin	69251.27	93
Round-robin	129091.82	20
UNIX scheduler	95077.55	31

UNIX scheduler	73850.73	62
UNIX scheduler	69251.27	93
UNIX scheduler	129091.82	20

Batch-3

Scheduling Algorithm	CPU Utilization	Quantum
Non-preemptive FCFS	0.948502	No Effect
Non-preemptive SJF	0.948502	No Effect
Round-robin	0.989470	31
Round-robin	0.993718	62
Round-robin	0.991604	93
Round-robin	0.993915	20
UNIX scheduler	0.989470	31
UNIX scheduler	0.990861	62
UNIX scheduler	0.991604	93
UNIX scheduler	0.993915	20

Scheduling Algorithm	Average Waiting Time	Quantum
Non-preemptive FCFS	22190.27	No Effect
Non-preemptive SJF	22141.18	100
Round-robin	94821.00	31
Round-robin	73215.64	62
Round-robin	69091.27	93

Round-robin	128986.18	20
UNIX scheduler	94821.00	31
UNIX scheduler	73821.27	62
UNIX scheduler	69091.27	93
UNIX scheduler	128986.18	20

Batch-4

Scheduling Algorithm	CPU Utilization	Quantum
Non-preemptive FCFS	1	No Effect
Non-preemptive SJF	1	No Effect
Round-robin	1	31
Round-robin	1	62
Round-robin	1	93
Round-robin	1	20
UNIX scheduler	1	31
UNIX scheduler	1	62
UNIX scheduler	1	93
UNIX scheduler	1	20

Scheduling Algorithm	Average Waiting Time	Quantum
Non-preemptive FCFS	33251.82	No Effect
Non-preemptive SJF	33251.82	No Effect
Round-robin	97465.18	31
Round-robin	78914.55	62

Round-robin	73994.82	93
Round-robin	131687.27	20
UNIX scheduler	97465.18	31
UNIX scheduler	78693.64	62
UNIX scheduler	73994.82	93
UNIX scheduler	131687.27	20

There are some observations to note down:

1. CPU utilisation in preemptive algorithm is more than that in CPU utilisation in non-preemptive algorithm. In preemptive algorithm increasing time quantum decreases CPU utilisation and make it close to non-preemptive one. This is due to that fact that preemption (for most general cases) ensures that for most of the time CPU and I/O works parallely. Empirically if the quantum is such that single CPU bursts of 80% process is approximately equal or less than the quantum higher CPU utilization is achieved. A very large scheduling quantum is practically equivalent to a non-preemptive scheduling.
2. Batch 4 has no I/O burst so it has 100% CPU utilisation as there can be no time when CPU is just doing I/O. Number of I/O calls are more in Batch1 than in Batch2 than in Batch3 and so CPU utilisation is more in Batch3 than in Batch2 than in Batch1.

Downside:

1. In round robin and unix scheduling algorithm waiting time increases with decrease in time quantum because it preempts each and every process very quickly and thus increases the time they have to wait to complete.
2. Also with low quantum there will be more context switches which in then increases the waiting time of the process and adds the context switch overhead to wait time.
3. Preemptive algorithm increase waiting time as compared to non-preemtive ones. This is due to the fact that processes get preempted before completion resulting in increased waiting time in the ready queue for all the processes.

PART-B

Scheduling Algorithm	Average Wait time in ready queue
Non-preemptive FCFS	50409.09
Non-preemptive SJF	36540.91

In this part , we compare the behaviour of Non-Preemptive algorithm First come First serve with Non-Preemptive algorithm SJF algorithm.
SJF gives a fair estimation of next cpu bursts for different threads and by using this estimation ,threads with lower value of next CPU burst are given higher priority over the others. Thus the order of execution of iterations changes, leading to reduced average waiting time in ready queue.

PART-C

Batch Number	Estimation Error
1	0.86
2	0.91
3	0.81
4	1.00
5	0.69

Larger number of CPU bursts leads to better estimation as the algorithm has more data to make estimation and hence lower error in estimation. In case of Batch 4 there is no scope of estimation as every process gets scheduled for exactly one time and hence large error in estimation.

While in case of Batch 1,2,3,5 there are more number of CPU bursts than in Batch 4 so better estimation and lower Estimation Error. Also Batch2 has less number of CPU bursts than in Batch1 so high error in estimation but even though Batch3 has even less number of CPU bursts it gives us better estimation because of presence of 4 big CPU bursts which dominates other CPU bursts and also they occupy most of CPU time, so our estimate will be close to these values and thus better estimation.

Now, on increasing the OUTER_BOUND estimation gets better because as we increase the OUTER_BOUND, the given threads will get scheduled for more number of times as thus a better estimation. Also the time quantum is same for each loop so estimation will increase with each iteration. This is because it will have more data to make estimation and thus lead to better results.

PART-D

Completion time statistics

Scheduling Algo	Maximum	Minimum	Average	Variance
Unix	163476	159626	162101.00	1948704.75
Round Robin	133354	59143	112657.20	594828224.00

In Round-Robin , all processes are given equal priority and hence the minimum , maximum and average completion times are very close to each other. Also , when we run round robin , the average completion time is higher due to the presence of a lot of I/O processes which creates a bottleneck and the processes end up spending most of their time in I/O and the CPU is idle in such situations.

In Unix, the maximum and minimum completion times are far apart and the variance is hence higher. However, the bottleneck is relieved in Unix as we assign different priorities to the processes and hence , all processes don’t reach to the I/O at the same time. Thus , the average CPU utilization is improved .