Operating Systems (CS330A)

Assignment-2 Report Group-16

Shivansh Rai(14658)
Sunil K. Pandey(14723)
Tushar Vatsal(14766)
Deepanshu Bansal(150219)

PART I

We got average CPU burst to be A = 124 for the program `testloop.c`. Pertaining to this value, the following values were provided during execution:

- A = 124
- Q_1 = 31
- Q_2 = 62
- Q_3 = 93
- We found that Q_4 = 20 as on increasing the scheduling quanta the CPU utilization decreases and CPU utilization maximises at the lowest quanta allowed.

And we have assigned numbers between 1 to 10 for scheduling algorithm, where 1 for non preemptive default scheduling, 2 for non preemptive SJF with quanta 100 (average as found), 3-6 for Round Robin scheduling with quanta Q_1, Q_2, Q_3, Q_4 and 7-10 for Unix scheduling with quanta Q_1, Q_2, Q_3, Q_4 .

A few observations

- As quanta increases, average wait time in ready queue decreases as well as CPU utilization decreases.
- Non-preemptive default scheduling algorithm and non preemptive SJF have similar behaviour. The same is true for Round Robin and UNIX scheduling algorithms.

CPU Utilization Table

Sch. Algo. No. / Batch N	1	2	3	4
1	0.56	0.83	0.95	1.00
2	0.56	0.83	0.95	1.00
3 ($Q = Q_1$)	0.66	0.90	0.99	1.00
4($Q = Q_2$)	0.62	0.89	0.99	1.00
5($Q = Q_3$)	0.61	0.88	0.99	1.00
6($Q = Q_4$)	0.72	0.93	0.99	1.00
7($Q = Q_1$)	0.77	0.98	0.99	1.00
8($Q = Q_2$)	0.73	0.98	0.99	1.00
9($Q = Q_3$)	0.67	0.97	0.99	1.00
10($Q = Q_4$)	0.79	0.99	0.99	1.00

Average Wait Time in Ready Queue Table

Sch. Algo. No. / Batch No.	1	2	3	4
1	24409	24355	24355	36577
2	24409	24355	24355	36577
3 ($Q = Q_1$)	106173	105314	105251	107590
4($Q = Q_2$)	81851	80332	80204	86750
5($Q = Q_3$)	76711	76398	76461	81369
6($Q = Q_4$)	143138	142951	142988	145813
7($Q = Q_1$)	84592	91744	91428	80376
8($Q = Q_2$)	62316	66659	64789	64915
9($Q = Q_3$)	62426	61227	61921	61124
10($Q = Q_4$)	119703	119566	117490	120078

PART II

Running non-preemptive algorithms on Batch5 having 5 testloop4.c and 5 testloop5.c programs.

Sch. Algo. No.	1 (Default NachOS)	2 (SJF)
Average Waiting Time	55450.0	40195

Since the inner loop size of testloop5 is smaller compared to testloop4 it has smaller CPU burst. Hence SJF will first schedule process with less CPU burst (testloop5) because of which it will take less average waiting time for testloop4 processes to get scheduled as testloop5 will exit early as compared to default non-preemptive algorithm. This is evident from the data in the above table. Also testloop5 process will finish early thus contributing less to average wait time.

PART III

Batch No.	Error for algo. 2
1	1.13
2	1.19
3	1.15
4	.99
5	.69

If OUTER_BOUND is increased the number of (I/O calls + calls to yieldCPU()) will increase (there are two calls to PrintInt()). This will result in increase in the number of CPU bursts, thus more number of estimations of expected CPU burst (the expected CPU burst is estimated at the end of every CPU burst). The CPU burst are same over different iterations, thus estimation will become better after each iteration. Therefore, the overall estimation will improve, decreasing the ratio.

In case of Batch4 the whole thread get executed in one quantum(8000 ticks) as the algorithm is non pre-emptive and hence the error is closer to 1

PART IV

Feature / Scheduling Algo	Round Robin	UNIX
Maximum Thread Execution Time	163386	131287
Minimum Thread Execution Time	159616	59029
Average Wait time in Ready Queue	76334	47383
Variance of Thread Execution Time	1910368	550535360

We observe that

1) Because of priority rule, higher priority processes tend to complete their execution faster, thus completing earlier than they would in round robin. This results in overall decrement in average waiting time in case of Unix scheduling algorithm as compared to round robin.
2) For UNIX algorithm, the time values are widely spreaded as can be seen from the difference between Maximum and Minimum Thread Execution Times as compared to the Round Robin algorithm. Therefore, UNIX algorithm has very high variance.
3) Time values for UNIX algorithm are widely spreaded because of difference in priorities of processes.