DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

CS330 : Assignment - 2

Ayush Sekhari (12185), Kundan Kumar (12375), Vicki Anand (12793)

October 1, 2014

1 SELECTION OF TIME QUANTAS

We wrote a bash script to run `testloop.c` with different values of Q4 in a pre-emptive round-robin fashion and recorded the CPU utilization for each case. The script `run_for_all.sh` in the userprog folder is used. We ran the script first with an interval of 5 on the choice of Q4 and then with an interval of 1 in the choice of Q4 to get the accurate answer. The values observed may be seen in Table 1.1 and Table 1.2. We observe that smaller the Q4, more is the CPU Utilization. The maximum Utilization was obtained at Q4 = 11, Since, we are allowed to choose Q4 to be at least 20, we decided to choose Q4 to be the minimum possible Q4 allowed i.e. 20.

Q4 chosen = 20

Explanation : We are calculating the CPU efficiency on the program `testloop`. We are using round-robin algorithm with different values of Q4. When we are calculating it, `testloop` is the only program which is running. So, if it goes to sleep, there would be no program to run and the CPU would be idle. If this program gets pre-emptied, then it would enter into the ready queue and would run again. This would involve context switches and mode switches which would be counted in the CPU bursts. Now, more are the context switches, more would be the total CPU burst but the total I/O burst would remain constant. Thus, the CPU utilization would increase as the share of time for which the CPU would remain idle would decrease (Because the net time increases as there are more context switches). Thus, lesser is the Quantum, more are context switches and thus more is the CPU utilization. Therefore, we were

Q4	CPU Utilization
19	72.68
24	68.61
29	66.25
34	64.65
39	63.52
44	62.66
49	62.16
54	61.47
59	61.13
64	60.77
69	61.08
74	60.88
79	60.68
84	60.45
89	60.32
94	60.14
99	60.00

Table 1.1: CPU Utilization for different values of Q4 for testloop.c

getting maximum Utilization when time quantum = 11 (time quantum less than this is not accepted by the system). Following this trend, we set Q4 = 20 which is the minimum possible time quantum allowed.
Problems: This is not a good way to find the value of Q4 as the optimization of CPU Utilization would not give any results because we are running only one program. The above explanation shows that CPU Utilization can be maximised in a fake way. We must consider other parameters also like Average Completion time , etc. to choose appropriate time quantum. In this case , the average completion time is increasing as I decrease the quantum interval (due to more context switches).

The value of A obtained when running `looptest` through a batch input is 130.32. However, if looptest is run directly using the command `./nachos -x ../test/testloop`, we obtain the value of A as 119.11. We would be using the value of A = 119 and would be rounding it to 120 for obaining Q1 , Q2 and Q3. The values obtained are:

A = 120
Q1 = 30
Q2 = 60
Q3 = 90

Q4	CPU Utilization
19	72.68
20	71.73
21	70.70
22	70.08
23	69.33
24	68.61
25	68.10

Table 1.2: CPU Utilization for different values of Q4 for testloop.c

1.1 WHY THE VALUES OF A ARE DIFFERENT FOR THE TWO CASES?

When the `testloop` is run directly using `./nachos -x ../test/testloop`, there is a single thread in the processor, main thread is overwritten by the testloop thread , Here the PID is 0. However, if we run `testloop` through batch input using -F flag, first `main` is created and then it created the `testloop` thread. Thus, In second case , `main` had PID 0 and `testloop` has PID 1.
Thus, In the two cases, the PID of `testloop` is different. If we look at the testloop, we are printing PID - 1. Therefore, in the first case, we would be printing -1 and in the second case, we would be printing 0. "`-1`" has more characters than "`0`". So, printing "`-1`" would involve more I/O and CPU bursts as compared to printing "`0`". These are short length bursts and therefore would decrease the average in the first case. This explains the difference in average CPU bursts in the two cases.

2 PART - I

The algorithms used for this part with the time Quantum lengths (if required) are listed in the Table 2.1.

Algorithm No.	Algorithm
1	Non-preemptive default NachOS scheduling
2	Non-preemptive shortest next CPU burst first algorithm
3	Round-robin with Quanta = 30
4	Round-robin with Quanta = 60
5	Round-robin with Quanta = 90
6	Round-robin with Quanta = 20
7	UNIX scheduler with Quanta = 30
8	UNIX scheduler with Quanta = 60
9	UNIX scheduler with Quanta = 90
10	UNIX scheduler with Quanta = 20

Table 2.1: Algorithms Used

2.1 Batch1

The CPU Utilization and Average Waiting Times are listed in Table 2.2.

Scheduling Algorithm Used	CPU Utilization	Average Waiting Time
1	56.28	22190.27
2	56.28	22190.27
3	66.35	97458.00
4	62.09	74276.91
5	60.71	70402.18
6	72.19	129372.09
7	66.30	96700.91
8	61.39	76208.36
9	60.69	71276.82
10	71.79	130009.00

Table 2.2: Output For Batch 1

2.2 Batch2

The CPU Utilizations and Average Waiting Times are listed in Table 2.3.

Scheduling Algorithm Used	CPU Utilization	Average Waiting Time
1	82.97	22141.18
2	82.97	22141.18
3	90.44	97043.09
4	89.16	75839.36
5	87.62	69809.27
6	93.07	129072.09
7	89.71	97200.36
8	88.88	76572.09
9	87.73	69933.09
10	92.15	129769.55

Table 2.3: Output For Batch 2

2.3 Batch3

The CPU Utilizations and Average Waiting Times are listed in Table 2.4.

Scheduling Algorithm Used	**CPU Utilization**	**Average Waiting Time**
1	94.85	22141.18
2	94.85	22141.18
3	99.14	96898.45
4	99.21	75380.18
5	99.10	69837.91
6	99.42	128995.64
7	98.77	97241.00
8	98.77	76172.91
9	99.53	69982.55
10	98.97	129834.36

Table 2.4: Output For Batch 3

2.4 Batch4

The CPU Utilizations and Average Waiting Times are listed in Table 2.5.

Scheduling Algorithm Used	**CPU Utilization**	**Average Waiting Time**
1	100.00	33251.82
2	100.00	33251.82
3	100.00	99033.64
4	100.00	79179.09
5	100.00	74217.27
6	100.00	132088.18
7	100.00	99388.18
8	100.00	79370.00
9	100.00	74422.73
10	100.00	132588.18

Table 2.5: Output For Batch 4

3 Part - II

The CPU Utilizations and Average Waiting Times are listed in Table 3.1.

Both, the processes `testloop4` and `testloop5` yield after an iteration of the outer-loop.
The scheduling algorithm 1 would run the processes on a first-come-first-serve basis in a non-preemptive fashion. Thus, each of the ten processes in the batch file would appear to run an iteration of the outerloop in a round robin fashion before yielding. Thus the order in which processes are run is fixed i.e. first an outer iteration of all five `testloop4` followed by an outer iteration of all five `testloop5` and then again `testloop4` and so on. But, this is not the most efficient way for average waiting time. Ideally, you would like to schedule the

shorter burst first which is not happening in this case.

On the other hand, the scheduling algorithm 2 i.e. non-preemptive shortest next CPU burst first algorithm always picks the thread with the shortest predicted next CPU Burst to be run. These processes are iterative i.e. the CPU burst for an outer iteration is constant. Thus, after some iterations, it would be able to predict the CPU burst required by the programs by exponential averaging technique. In this case, it will learn that it is better w.r.t. average waiting time to run an iteration in `testloop5` over `testloop4`.

Scheduling Algorithm Used	CPU Utilization	Average Waiting Time
1	100.00	50409.09
2	100.00	36540.91

Table 3.1: Output for Batch 5 for the non-preemptive algorithms

4 Part - III

The results for the overall estimation error for non-preemptive shortest next CPU burst first algorithm are listed in Table 4.1. We have calculated percentages instead of ratio.

We can observe that if we increase the value of OUTER_BOUND (Here, we increased it from 4 to 10), the value of the error reduces drastically. This is because these programs are loop programs. For all these programs (except `testloop3`) , there is a yield in every iteration. More are the number of iterations, more does the algorithm gets the chance to adapt to predict the correct next CPU Burst using exponential averaging. This improvement in prediction is reflected in drastic decrease in overall estimation error.
No such improvement is seen in the case of the `testloop3` as it does not yields between iterations and thus whole of the programs runs in a single CPU burst and the scheduling algorithm does not gets a chance to choose among programs and thus does not affects anything.
`testloop` has `sys_PrintInt` which make the thread to yield because of the I/O burst and thus it shows similar behaviour to other programs with yield in their loop iteration.

Batch	OUTER_BOUND = 4	OUTER_BOUND = 10
Batch1	84.80	38.12
Batch2	89.82	38.88
Batch3	79.49	34.04
Batch4	99.09	99.63
Batch5	67.31	28.82

Table 4.1: Overall Estimation Error for all the five batches for OUTER_BOUND = 4 and OUTER_BOUND = 10

5 Part - IV

The completion time statistics may be seen in the Table 5.1. The quantum length and timer interval are both set to 100.

Explanations:

- Round Robin algorithm executes the ten programs one by one with a fixed quantum of 100.
- Unix schedular decides among the programs of which one to run based on the priorities.
- One of the programs has a very large priority and thus there is a high probability of it being selected. This program terminates early and thus there is a significant difference in the minimum waiting time (In fact the thread with priority value 10 is the one that terminates earliest).
- Round robin handles all the threads with equal priorities i.e. gives fair chance for all to run. Threads are run in a round one by one. Thus, we can expect the threads to terminate nearly at the same time. This is the reason why max - min is very small for Round-Robin as compared to Unix Schedular. This also explains why the variance for Round Robin is very small as compared to Unix schedular.

Data	**Round-Robin**	**Unix Scheduler**
Avg. Completion Time	162101.00	112657.20
Max. Completion Time	163476	133354
Min. Completion Time	159626	59143
Variance of Completion Times	1948705.00	594828208.76

Table 5.1: Completion time data for Round-Robin and Unix Scheduler with time quantum and timer interval both set to 10