Assignment 2, CS330a
Semester I 2014-15

Group 21
Roll - 12633, 12640, 12714
Course Instructor - Prof. Mainak Choudhary

October 1, 2014

1 Assumptions

- In calculation of average CPU Burst for evaluation of Q1,Q2,Q3,Q4 we use a single instance of a testloop program submitted as a batch program with flag -F, Thus in calculation of average CPU Burst we considered CPU Burst of the main thread as a part of overall calculation
- We include the statistic corresponding to main thread in the calculation of all statistic except for completion time
- For flag -x we have defined the type algorithm equal to 11 which implements the round robbin algorithm with time quantam equal to 100 (As was used in assignment 1)
- We have included the main thread in calculation of Completion time when the flag is -x

2 Part I

2.1 PART I : Batch 1

Algorithm Type	Scheduling Policy	CPU Utilization	Average waiting time
1	Non-Preemtive default Nachos scheduling	56.283318%	22190.272727
2	Non-Preemtive shortest next CPU burst first algorithm	56.283318%	22190.272727
3	Round-robin with quanta 32	65.647133%	94176.636364
4	Round-robin with quanta 65	61.818074%	71574.545455
5	Round-robin with quanta 97	60.533606%	68156.818182
6	Round-robin with quanta 20	72.194768%	129372.090909
7	UNIX scheduler with quanta 32	65.779925%	93665.909091
8	UNIX scheduler with quanta 65	61.713874%	74357.181818
9	UNIX scheduler with quanta 97	60.439560%	69830.909091
10	UNIX scheduler with quanta 20	71.786087%	130009.000000

2.2 PART I:Batch 2

Algorithm Type	Scheduling Policy	CPU Utilization	Average waiting time
1	Non-Preemtive default Nachos scheduling	82.974849%	22141.181818
2	Non-Preemtive shortest next CPU burst first algorithm	82.974849%	22141.181818
3	Round-robin with quanta 32	90.327935%	93101.000000
4	Round-robin with quanta 65	89.114465%	71464.181818
5	Round-robin with quanta 97	87.536771%	68873.909091
6	Round-robin with quanta 20	93.066826%	129072.090909
7	UNIX scheduler with quanta 32	90.662519%	92985.000000
8	UNIX scheduler with quanta 65	89.267740%	73962.363636
9	UNIX scheduler with quanta 97	87.878073%	71015.727273
10	UNIX scheduler with quanta 20	92.147257%	129769.545455

2.3 PART I:Batch 3

Algorithm Type	Scheduling Policy	CPU Utilization	Average waiting time
1	Non-Preemtive default Nachos scheduling	94.850203%	22141.181818
2	Non-Preemtive shortest next CPU burst first algorithm	94.850203%	22141.181818
3	Round-robin with quanta 32	99.261069%	93004.454545
4	Round-robin with quanta 65	99.086236%	71428.090909
5	Round-robin with quanta 97	99.417615%	68837.727273
6	Round-robin with quanta 20	99.417372%	128995.636364
7	UNIX scheduler with quanta 32	99.328602%	92854.636364
8	UNIX scheduler with quanta 65	98.515273%	73427.272727
9	UNIX scheduler with quanta 97	99.083719%	70550.909091
10	UNIX scheduler with quanta 20	98.965919%	129834.363636

2.4 PART I:Batch 4

Algorithm Type	Scheduling Policy	CPU Utilization	Average waiting time
1	Non-Preemtive default Nachos scheduling	100.000000%	33251.818182
2	Non-Preemtive shortest next CPU burst first algorithm	100.000000%	33251.818182
3	Round-robin with quanta 32	100.000000%	96182.000000
4	Round-robin with quanta 65	100.000000%	78090.909091
5	Round-robin with quanta 97	100.000000%	73548.818182
6	Round-robin with quanta 20	100.000000%	132088.181818
7	UNIX scheduler with quanta 32	100.000000%	96462.909091
8	UNIX scheduler with quanta 65	100.000000%	78359.545455
9	UNIX scheduler with quanta 97	100.000000%	73628.181818
10	UNIX scheduler with quanta 20	100.000000%	132588.181818

2.5 Observation

- Waiting time increases as the Time quantam decreases in Preemtive Algorithms

- Waiting time for First come first serve is same as the waiting time for shortest job first
- Waiting time of Non Preemtive algorithm is less than waiting time of Preemtive Algorithm
- The CPU utilization increases as we go from Batch 1 to Batch 4
- CPU Utilization of Shortest Job first and First come First serve are same for Batch 1,2,3,4
- Preemtive Algorithm have a higher Cpu Utilization then the Non - Preemtive Algorithm

2.6 Explanation

2.6.1 Waiting Time analysis for Preemtive Algorithms for different Time quantams

As we increase the time quantam the waiting time decreases. This is because as the quantam increases processes tends to finish early leading to less waiting time . As mentioned in the assignment we can consider IO Burst time to be negligible. So let us consider a example of n processes each having CPU_BURST time as cb.

Waiting Time when time quantam t

Waiting Time = 0+ t + 2*t +..........(n-1)*t + (n-1)*t*n*(cb/t-1)

Waiting Time = (n-1)*n*cb - (n*(n-1))/2 * t

So more the value of t less is the waiting time

2.6.2 Waiting Time analysis for Non-Preemtive Algorithm

The Waiting time of Shortest Job first is same as the Waiting time for the First come First serve for Batch 1,2,3,4 because the batches have 10 copies of the same program. So no matter in which order the algorithm are schedule the waiting time will remain the same.

2.6.3 Waiting Time comparison for Non - Preemtive Algorithm VS Preemtive Algorithm

We can always say that the waiting time for the non preemtive process will be always less than the preemtive process because non preemtive ones are similar to preemtive with a large time quantam and as we proved above that more the time quantam lesser is the waiting time

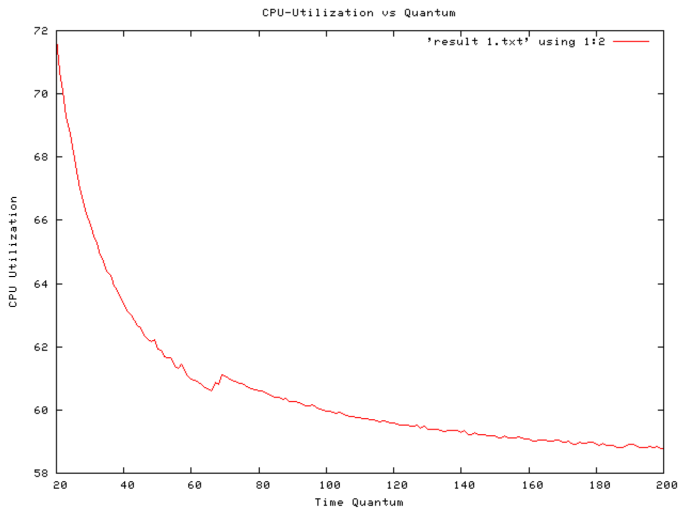

Figure 1: Cpu utilization vs Time Quantam

2.6.4 Cpu Utilization between Different Batches

The CPU utilization increases as we go from Batch 1 to Batch 4. This is because as we go from testloop.c to testloop3.c, the number of print calls decrease monotonously(no print call in testloop3.c). So the no of I/O bursts decrease monotonously. Hence the time for which all the active processes will be sleeping at the same time decreases, and hence the chances that the CPU will be busy increases. Therefore the CPU utilization increases as we go from Batch1 to Batch4.

2.6.5 Deciding time quantam

When we submit testloop as a single job using -F, the average CPU burst observed (A) is 130, so we take Q1 as 32, Q2 as 65, Q3 as 97. From the above figure we can conclude that the CPU Utilization decreases with increase in time quantam for the testloop program. Hence the minimal quantam that has maximum CPU Utilization is 20. As the batch contains only one process, Hence the behavior of both Unix Scheduler and Round Robin is same. This is because lesser the time quantam more the number of context switches, which leads to more system ticks and since the idleticks nearly remain the same, the ratio of idleticks to total execution time more. Hence the CPU Utilization increases increases with lower time quantam

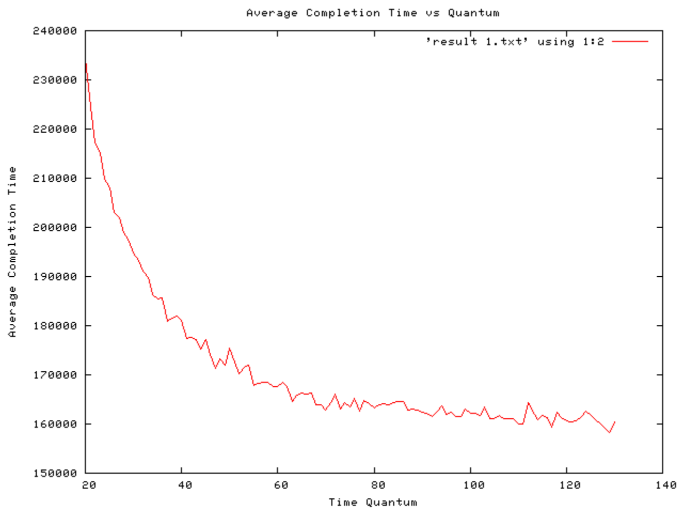

Figure 2: Completion Time vs Time Quantam

2.7 Drawback of Time Quantam

Average waiting time, Average completion time, Average execution time is more for time quantam 20 as compared to other time quantam which could be considered as a downside of the time quantam . This is so because less the time quantam more will be the context switches, hence more will be the system Ticks , So more total Ticks .

2.7.1 Cpu Utilization analysis for Non-Preemtive Algorithm

The CPU Utilization of Shortest Job first is same as the CPU Utilization for the First come First serve for Batch 1,2,3,4 because the batches have 10 copies of the same program, So no matter in which order the algorithm are schedule the Cpu Utilization will remain the same

2.7.2 Cpu Utilization comparison for Non - Preemtive Algorithm VS Preemtive Algorithm

The preemtive Algorithm have a higher Cpu Utilization then the Non - Preemtive Algorithm as due to preemtion the number of context switch increases, so the number of System Ticks increases.

CPU Utilization = 1-(idle Ticks)/(user Ticks + system Ticks + idle Ticks)

As the system ticks the CPU Burst time increases. The no of idle Ticks and user Ticks nearly constant for both pre-emptive and non-premeptive algorithms.

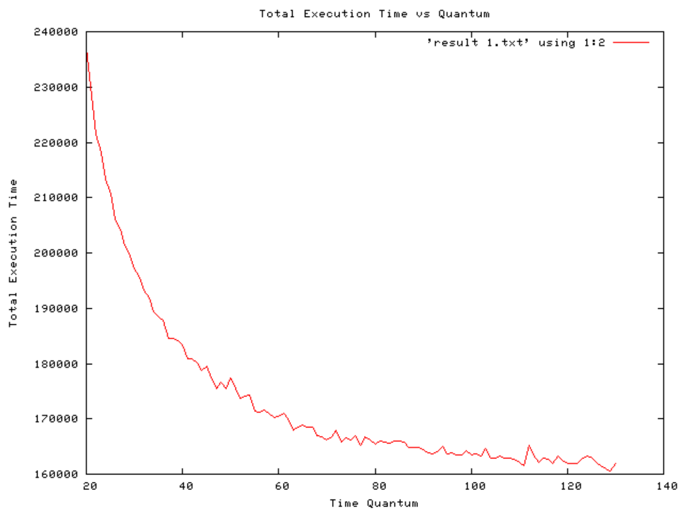

Figure 3: Execution Time vs Time Quantam

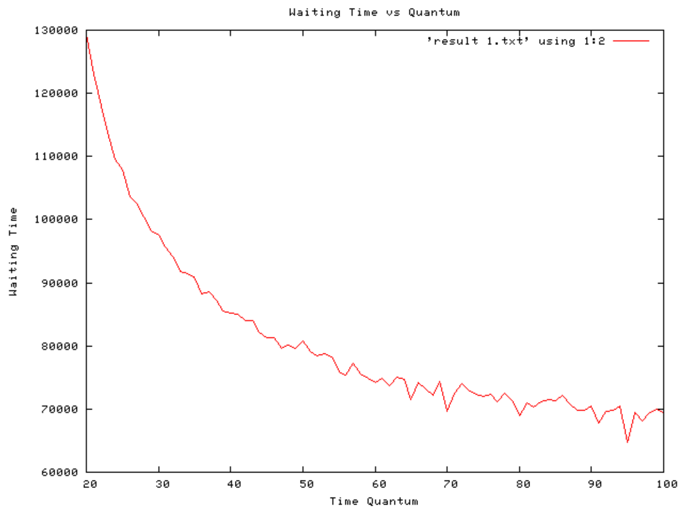

Figure 4: Waiting Time vs Time Quantam

Hence Preemtive Algorithm have a higher Cpu Utilization then the Non - Preemtive Algorithm

3 PART II:Batch 5

Algorithm Type	Scheduling Policy	Average waiting time
1	Non-Preemtive default Nachos scheduling	50409.090909
2	Non-Preemtive shortest next CPU burst first algorithm	36540.909091

3.1 Observation

- Waiting time of Shortest Job first is less than First come first serve for Batch 5

3.2 Explanation

The waiting time of Shortest Job first is less than First come first serve for Batch 5 .Batch 5 consist of 5 testloop4 followed by 5 testloop5. As the cpu burst time of testloop5 is less than cpu burst time of testloop 4 due to less value of Size (defined in testloop4,testloop5). So in Shortest Job first the testloop 5 process will be evaluated first whereas in First Come First serve we will process testloop 4 programs first. As the cpu burst time of the current running process is added to the waiting time of every subsequent process in the ready queue . So if we schedule a job with greater cpu burst time earlier then the overall waiting time will increase .

4 PART III

Algorithm Type	Scheduling Policy	Average Estimation Error
2	Shortest next CPU burst first algorithm on BATCH 1	0.857871
2	Shortest next CPU burst first algorithm on BATCH 2	0.908886
2	Shortest next CPU burst first algorithm on BATCH 3	0.805894
2	Shortest next CPU burst first algorithm on BATCH 4	0.997293
2	Shortest next CPU burst first algorithm on BATCH 5	0.687935

OUTER_BOUND	Batch 1	Batch 2	Batch 3	Batch 4	Batch 5
2	1.266301	1.421734	1.280240	0.994622	1.092456
4	0.857871	0.908886	0.805894	0.997293	0.687935
5	0.720880	0.755081	0.667312	0.997831	0.569651
7	0.538715	0.556119	0.489228	0.998449	0.417524
10	0.385898	0.393761	0.345345	0.998913	0.294697
13	0.299909	0.303826	0.266049	0.999163	0.227050
17	0.231112	0.232901	0.203690	0.999360	0.173876
20	0.197211	0.198226	0.173263	0.999456	0.147932

Estimated error vs Outer Bound for different Batches

4.1 Observation

- As the Outer_Bound value increases the error in the estimated value of Cpu Burst decreases if the No of CPU Burst are greater than one
- As the Outer_Bound value increases the error in the estimated value of Cpu Burst increases if the No of CPU Burst is one

4.2 Explanation

We notice that as we increase the Outer_Bound value the error in the estimated value of Cpu Burst decreases if the No of CPU Burst are greater than one because the Cpu burst time over different values of Outer_Bound remains same . As the estimated CPU burst depends on the previous CPU burst and previous estimated CPU burst .

$\Rightarrow$ New Estimated CPU Burst = (Old Estimated Cpu Burst + Old CPU burst)/2
$\Rightarrow$ Error in Estimated CPU burst = absolute value of (New Estimated CPU Burst - New CPU Burst)
$\Rightarrow$ Error in Estimated CPU burst = absolute value of ((Old Estimated Cpu Burst + Old CPU burst)/2 - New CPU Burst)
But Old CPU Burst = New CPU Burst = CPU BURST
$\Rightarrow$ Error in Estimated CPU burst = absolute value of (Old Estimated Cpu Burst/2 - CPU Burst/2)
$\Rightarrow$ New estimated error = Old estimated error /2
Hence the error over different iterations decreases

As the Outer_Bound value increases the error in the estimated value of Cpu Burst increases if the No of CPU Burst is one as the starting estimation remains fixed for different CPU Burst time. The error in estimation increases as the Outer Bound increases.

So the error in Estimated CPU Utilization increases for Batch 4 and decreases with Batch 1,2,3,5 with increase in Outer_Bound

5 PART IV:Batch 6

Index	Scheduling Policy	Maximum	Minimum	Average	Variance of job completion times
1	Round-robin with quanta 100	163476	159626	162101.000000	1948705.000000
2	UNIX scheduler with quanta 100	133354	59143	112657.200000	594828208.760000

Completion time statistic for Round Robbin and UNIX

5.1 Observation

- Minimum Completion time of Unix Scheduler is less than Round Robbin

- Maximum Completion time of Unix Scheduler is less than Round Robbin
- Average Completion time of Unix Scheduler is less than Round Robbin
- Variance in Completion time of Unix Scheduler is more than Round Robbin

5.2 Explanation

As in the Unix Scheduler we assign priority to each process based on the CPU Utilization and the Priority of different process. UNIX takes care that process with high priority get schedule first .We have also ensured that if two processes have the same priority, the one that has been waiting for a longer time is scheduled first.

Minimum Completion time : Decreases for the Unix Scheduler because the process with high priority finishes first whereas in the Round Robbin Scheduling all process almost ends at the same time .

Maximum Completion Time : Decreases for the Unix Scheduler because in Unix Scheduler it may happen that the same process is Scheduled to run in the next quantam again leading to less number of context switch, So the overhead of the context switch is less in unix scheduler than Round Robbin Scheduler

Average Completion Time : Decreases for the Unix Scheduler because the completion time of all the process will be distributed over the whole range whereas in round robbin the completion time of all the process will be towards the end and as Unix scheduler is expected to finish early as compared to Round Robbin. So the average completion time of Round Robbin is greater than average completion time of UNIX

Variance : Increases for the Unix Scheduler because the completion time of all the process will be distributed over the whole range whereas in round robbin the completion time of all the process will be towards the end and as both Unix and Round Robbin are expected to end at nearly same time. So the deviation from the mean CPU Utilization is more in the case of Unix Scheduler as compared to Round Robbin Scheduling