Research Journal of Information Technology 1(1): 22-26, 2009
ISSN: 2041-3114
© Maxwell Scientific Organization, 2009
Submit Date: June 03, 2009 Accepted Date: July 21, 2009 Published Date: August 29, 2009

Performance Assessment of Some CPU Scheduling Algorithms

[1]E.O. Oyetunji and [2]A. E. Oluleye
[1]Department of Applied Mathematics and Computer Science,
University for Development Studies, Ghana
[2]Department of Industrial and Production Engineering, University of Ibadan, Nigeria

Abstract: The problem of scheduling which computer process run at what time on the central processing unit (CPU) or the processor is explored. Three basic CPU scheduling algorithms (namely first come first serve, priority scheduling and shortest job first) were discussed and evaluated on four CPU scheduling objectives or goals (average waiting time, average turnaround time, average CPU utilization and average throughput) to determine which algorithm is most suitable for which objective. Experimental results were provided.

Key words: CPU scheduling, Algorithms, Processes, CPU/IO burst

INTRODUCTION

The Central Processing Unit (CPU) is an important component of the computer system, hence it must be utilized efficiently. This can be achieved through what is called CPU scheduling. The CPU scheduling can be defined as the art of determining which processes run on the CPU when there are multiple runnable processes (1, 2). Also, it is the problem of deciding which computer process in the ready queue (in other words, which particular programs need some processing and are ready and waiting for it) is to be allocated to the CPU for processing. It is a fundamental problem in operating systems (OS) in terms of minimizing the wait for the user when he or she simply wants to execute a particular set of tasks. It is important because it has a big effect on resource utilization and the overall performance of the system.

In the computer system, all processes consist of a number of alternating two burst cycles (the CPU burst cycle and the Input & Output (IO) burst cycle) (3). Normally, a process will run for a while (the CPU burst), perform some IO (the IO burst), then run for a while more (the next CPU burst), again perform some IO (the IO burst). These cycles continues until the execution of the process is completed. An IO Bound process is a process that performs lots of IO operations such as reading from and writing to disks. Each IO operation is followed by a short CPU burst to process the IO, and then more IO happens. A CPU bound process is a process that performs lots of computation and do little IO. A typical system has a few long CPU bursts. One of the things a scheduler will typically do is switch the CPU to another process when one process does IO operations. This is because the IO usually take a long time, and we don't want to leave the CPU idle while wait for the IO to finish.

There are three possible process states: running state (process is running on the CPU), ready state (process is ready to run but not actually running on the CPU) and waiting state (process is waiting for some event like IO to happen) (4). The CPU chooses (schedules) which process to run when any of the following occur:

- When process switches from running to waiting. This could be as a result of IO request, waiting for child to terminate, or waiting for synchronization operation to complete.
- When process switches from running to ready. This could be as a result of completion of interrupt handler. If scheduler switches processes in this case, it has preempted the running process.
- When process switches from waiting to ready state (on completion of IO).
- When a process terminates.

A number of basic assumptions are often being made in CPU scheduling. These are as follows (1, 5):

1 There is a pool of runnable processes contending for the CPU.
2 The processes are independent and compete for resources.
3 The job of the scheduler is to distribute the scarce resource of the CPU to the different processes fairly (according to some definition of fairness) and in a way that optimizes some performance criteria.

In this paper, we discussed five CPU scheduling goals that can be used to evaluate the performances of CPU scheduling algorithms. Three CPU scheduling algorithms commonly in use were discussed and evaluated. A number of randomly generated problems were solved and experimental results were provided.

CPU Scheduling Criteria: CPU scheduling criteria are the basis on which the performance of CPU scheduling

Corresponding Author: E.O. Oyetunji, Department of Applied Mathematics and Computer Science,University for Development Studies, Ghana

algorithms is evaluated. There are many possible criteria (1, 3):

CPU Utilization: This is a measure of how busy the CPU is. Usually, the goal is to maximize the CPU utilization.

Throughput: This is the number of processes completed per unit time. Usually, the goal is to maximize the throughput.

Turnaround Time: This is the amount of time from submission to completion of process. Usually, the goal is to minimize the turnaround time.

Waiting Time: This is the amount of time spent ready to run but not running. It is the difference in start time and ready time. Usually, the goal is to minimize the waiting time

Response Time: This is the amount of time between submission of requests and first response to the request. Usually, the goal is to minimize the response time

CPU Scheduling Algorithms: Four algorithms commonly used in CPU scheduling are discussed below.

First-Come, First-Served (FCFS): This algorithm allocates the CPU to the process that requests the CPU first. This algorithm is easily managed with a FIFO queue. New process enters the queue through the tail of the queue and leaves through the head of the queue (when the process is allocated to the CPU) (1). The processes are allocated to the CPU on the basis of their arrival at the queue. Once a process is allocated to the CPU, it is removed from the queue. A process does not give up CPU until it either terminates or performs IO.

Shortest-Job-First (SJF): The SJF algorithm associates the length of the next CPU burst with each process such that that the process that have the smallest next CPU burst is allocated to the CPU(1). The SJF uses the FCFS to break tie (a situation where two processes have the same length next CPU burst). The SJF algorithm may be implemented as either a preemptive or non-preemptive algorithms. When the execution of a process that is currently running is interrupted in order to give the CPU to a new process with a shorter next CPU burst, it is called a preemptive SJF. On the other hand, the non-preemptive SJF will allow the currently running process to finish its CPU burst before a new process is allocated to the CPU.

Priority Scheduling (PS): The PS algorithm associates with each process a priority and the CPU is allocated to the process based on their priorities. Usually, lower numbers are used to represent higher priorities. The process with the highest priority is allocated first. If there are multiple processes with same priority, typically the FCFS is used to break tie (1).

Round Robin (RR): The RR algorithm is designed especially for time-sharing systems and is similar to the FCFS algorithm. Here, a small unit of time (called time quantum or time slice) is defined. A time quantum is generally from 10-100 milliseconds. So, the RR algorithm will allow the first process in the queue to run until it expires its quantum (i.e. runs for as long as the time quantum), then run the next process in the queue for the duration of the same time quantum. The RR keeps the ready processes as a FIFO queue. So, new processes are added to the tail of the queue. Depending on the time quantum and the CPU burst requirement of each process, a process may need less than or more than a time quantum to execute on the CPU. In a situation where the process need more than a time quantum, the process runs for the full length of the time quantum and then it is preempted. The preempted process is then added to the tail of the queue again but with its CPU burst now a time quantum less than its previous CPU burst. This continues until the execution of the process is completed (1). The RR algorithm is naturally preemptive.

Data Analysis: Four CPU algorithms have been discussed but only three were used in performance evaluation. The Round Robin (RR) is traditionally a pr-emptive algorithm, hence it will be improper to compare its performance with non-preemptive algorithms (FCFS, PS and SJF). The non-preemptive version of SJF was implemented. It was assumed that when the execution of a process start a response has been received, therefore, the waiting time and response time will be equal.

In order to assess the performance of the three non-preemptive algorithms, 50 problems each were randomly generated for 10 different problem sizes ranging from 5 to 50 jobs/processes. In all, a total of 500 randomly generated problems were solved. The processing time of jobs were randomly generated with values ranging between 1 and 15 inclusive. The ready time of jobs were also randomly generated with values ranging between 0 and 10 inclusive. The priorities of jobs/processes were also randomly generated with values ranging from 0 to 9 (0 represent highest priority).

A program was written in Microsoft visual basic 6.0 to apply the solution methods (FC, PS and SJF) to the problems generated. The program computes the values of Waiting time, Turnaround time, CPU Utilization and Throughput for each job/process. Average values of these quantities were computed by dividing the sum of each quantity by the number of jobs/processes. For example, Average Waiting Time is obtained by summing the Waiting Time of all the jobs/processes divide by the number of jobs/processes. In a similar manner, the Average Turnaround time, Average CPU Utilization and Average throughput were computed. Also, the program computes the execution time (seconds) required to solve an instance of a problem for each solution method and for each problem.

The data was exported to Statistical Analysis System (SAS version 9.1) for detailed analysis. SAS is a very versatile statistical package and was employed to enable credible conclusions to be drawn from the results. The hardware used for the experiment was a T2080 processor running at 1.73 GHz and with 1GB of main memory.

The general linear model (GLM) procedure in SAS was used to compute both the mean value of each of the four CPU scheduling goal and the mean value of time required for each problem size (50 problem instances were solved under each problem size) and by solution methods. The test of means was also carried out using the GLM procedure so as to determine whether or not the differences observed in the mean value of each of four CPU scheduling goal and mean value of time required by various solution methods are statistically significant.

RESULTS AND DISCUSSION

The mean values of average waiting time, average turnaround time, average CPU utilization and average throughput obtained from the various CPU scheduling algorithms and various problem sizes are shown in Tables 1, 2, 3 and 4 respectively. Typically, since the goal is to minimize the average waiting time, therefore, base on the minimum mean value of the average waiting time criterion, the algorithms are ranked as SJF, FC and PS for all the problems sizes (5 to 50 jobs) considered (Table 1). The same ranking was obtained for the average turnaround time (Table 2).

Typically, the goal is to maximize the average CPU utilization. Therefore, base on the maximum mean value of the average CPU utilization criterion, the algorithms are ranked as FC, PS and SJF for all the problems sizes considered (Table 3). The same ranking was also obtained for the average throughput (Table 4).

When the results shown in Tables 1, 2, 3 and 4 were subjected to statistical test to determine whether or not the differences observed in the mean values of average waiting time, average turnaround time, average CPU utilization and average throughput obtained from the various algorithms are significant, the results obtained (for 5x1 problem size) are shown in Tables 5, 6, 7 and 8 respectively.

Table 5 shows that the performance of SJF with respect to the average waiting time is significantly different (at 5%) from the performances of both FC and PS algorithms for the 5x1 problem size. Also, the performance of SJF with respect to the average turnaround time is significantly different (at 5%) from the performances of both FC and PS algorithms for the 5x1 problem size (Table 6). The summary of the test of means of SJF, FC and PS carried out with respect to the average waiting time and average turnaround time for all the problem sizes considered is shown in Table 9. It is evident that the performance of SJF with respect to both the average waiting time and average turnaround time is significantly different (at 5%) from the performances of

Table 1: Mean of Average Waiting time by algorithms

Problem	Mean of Average Waiting time		
Size	FC	PS	SJF
5x1	12.59	14.69	10.98
10x1	35.76	38.89	28.91
15x1	50.73	53.59	37.10
20x1	71.17	74.13	51.40
25x1	87.34	92.50	64.04
30x1	114.10	119.40	82.70
35x1	130.50	135.80	95.00
40x1	153.72	156.97	105.99
45x1	170.50	182.00	127.00
50x1	201.00	201.30	141.30

Sample size = 50

Table 2: Mean of Average Turnaround time by algorithms

Problem	Mean of Average Turnaround time		
Size	FC	PS	SJF
5x1	20.30	22.40	18.69
10x1	44.28	47.41	37.44
15x1	58.37	61.23	44.74
20x1	78.96	81.92	59.19
25x1	95.04	100.20	71.75
30x1	122.20	127.60	90.80
35x1	138.60	143.90	103.00
40x1	161.60	164.90	113.90
45x1	178.70	190.20	135.20
50x1	209.10	209.40	149.40

Sample size = 50

Table 3: Mean of Average CPU Utilization by algorithms

Problem	Mean of Average CPU Utilization		
Size	FC	PS	SJF
5x1	94.41	88.20	85.54
10x1	97.70	93.21	92.02
15x1	98.89	95.64	94.44
20x1	99.29	96.61	95.47
25x1	99.42	97.17	96.82
30x1	99.56	97.75	97.15
35x1	99.64	98.26	97.50
40x1	99.67	98.11	97.66
45x1	99.72	98.36	98.46
50x1	99.75	98.49	98.14

Sample size = 50

Table 4: Mean of Average Throughput by algorithms

Problem	Mean of Average Throughput		
Size	FC	PS	SJF
5x1	0.1285	0.1190	0.1161
10x1	0.1182	0.1127	0.1111
15x1	0.1327	0.1285	0.1264
20x1	0.1294	0.1258	0.1243
25x1	0.1305	0.1275	0.1271
30x1	0.1225	0.1203	0.1195
35x1	0.1252	0.1234	0.1224
40x1	0.1263	0.1243	0.1237
45x1	0.1223	0.1206	0.1207
50x1	0.1241	0.1225	0.1221

Sample size = 50

Table 5: Test of means of Average Waiting time for 5x1 problem (probability values)

	Algorithms		
Algorithms	FC	PS	SJF
FC	-	0.014*	0.058*
PS	0.014*	-	<.001*
SJF	0.058*	<.001*	-

Note: * indicate significant result at 5% level; Sample size = 50
-indicate not necessary

Table 6: Test of means of Average Turnaround time for 5x1 problem (probability values)

	Algorithms		
Algorithms	FC	PS	SJF
FC	-	0.059*	0.148
PS	0.059*	-	0.001*
SJF	0.148	0.001*	-

Note: * indicate significant result at 5% level;Sample size = 50
-indicate not necessary

Table 7: Test of means of Average CPU Utilization for 5x1 problem (probability values)

	Algorithms		
Algorithms	FC	PS	SJF
FC	-	<.001*	<.001*
PS	<.001*	-	0.035*
SJF	<.001*	0.035*	-

Note: * indicate significant result at 5% level;Sample size = 50
-indicate not necessary

Table 8: Test of means of Average Throughput time for 5x1 problem (probability values)

	Algorithms		
Algorithms	FC	PS	SJF
FC	-	0.082	0.024*
PS	0.082	-	0.595
SJF	0.024*	0.595	-

Note: * indicate significant result at 5% level;Sample size = 50
-indicate not necessary

both FC and PS algorithms for all the problem sizes considered (Table 9). This means that for the CPU scheduling problems where it is desired to minimize either the average waiting time or average turnaround time, the best algorithm to use is the shortest job first (SJF).

The performance of FC algorithm with respect to the average CPU utilization and average throughput is significantly different (at 5%) from the performances of both PS and SJF algorithms for the 5x1 problem size (Tables 7 and 8). The summary of the test of means of SJF, FC and PS carried out with respect to the average CPU utilization and average throughput for all the problem sizes considered is shown in Table 10. For all the problem sizes considered, the performance of SJF with respect to both the average waiting time and average turnaround time is significantly different (at 5%) from the performances of both FC and PS algorithms (Table 10). This shows that for the CPU scheduling problems where it is desired to maximize either the average CPU utilization or average throughput, the best algorithm to use is the first come first serve (FC).

Tables 9 and 10 should be interpreted as follows: Both Tables consists of three major columns namely problem size, average waiting time and average turnaround time (for Table 9) and problem size, average CPU utilization and average throughput (for Table 10). The second and third major columns in each table were further divided into three columns each. Two of such columns (second and third columns) are labeled 1 and 2.

Table 9: Summary of test of means of Average Waiting time and Average Turnaround time by problem sizes

Problem	Average Waiting time			Average Turnaround time		
Size		1	2		1	2
5x1	SJF^{*1,2},	FC^{*2},	PS	SJF^{*2},	FC^{*2},	PS
10x1	SJF^{*1,2},	FC^{*2},	PS	SJF^{*1,2},	FC^{*2},	PS
15x1	SJF^{*1,2},	FC^{*2},	PS	SJF^{*1,2},	FC,	PS
20x1	SJF^{*1,2},	FC,	PS	SJF^{*1,2},	FC,	PS
25x1	SJF^{*1,2},	FC^{*2},	PS	SJF^{*1,2},	FC^{*2},	PS
30x1	SJF^{*1,2},	FC^{*2},	PS	SJF^{*1,2},	FC^{*2},	PS
35x1	SJF^{*1,2},	FC,	PS	SJF^{*1,2},	FC,	PS
40x1	SJF^{*1,2},	FC^{*2},	PS	SJF^{*1,2},	FC,	PS
45x1	SJF^{*1,2},	FC^{*2},	PS	SJF^{*1,2},	FC^{*2},	PS
50x1	SJF^{*1,2},	FC,	PS	SJF^{*1,2},	FC,	PS

Sample size = 50

Note: *1,2 indicates significant result at 5% level from solution methods in columns 1, 2
*1 indicates significant result at 5% level from solution methods in column 1
*2 indicates significant result at 5% level from solution methods in column 2

Table 10: Summary of test of means of Average CPU Utilization and Average Throughput by problem sizes

Problem	Average CPU Utilization			Average Throughput		
Size		1	2		1	2
5x1	FC^{*1,2},	PS^{*2},	SJF	FC^{*1,2},	PS,	SJF
10x1	FC^{*1,2},	PS^{*2},	SJF	FC,	PS,	SJF
15x1	FC^{*1,2},	PS^{*2},	SJF	FC,	PS,	SJF
20x1	FC^{*1,2},	PS^{*2},	SJF	FC,	PS,	SJF
25x1	FC^{*1,2},	PS,	SJF	FC,	PS,	SJF
30x1	FC^{*1,2},	PS^{*2},	SJF	FC,	PS,	SJF
35x1	FC^{*1,2},	PS^{*2},	SJF	FC,	PS,	SJF
40x1	FC^{*1,2},	PS^{*2},	SJF	FC,	PS,	SJF
45x1	FC^{*1,2},	PS,	SJF	FC,	PS,	SJF
50x1	FC^{*1,2},	PS^{*2},	SJF	FC,	PS,	SJF

Sample size = 50

Note: *1,2 indicates significant result at 5% level from solution methods in columns 1, 2
*1 indicates significant result at 5% level from solution methods in column 1
*2 indicates significant result at 5% level from solution methods in column 2

So, the algorithm that appeared in the first column of the major columns is the one that gives the best value of the CPU scheduling goal being reported in the major column. The algorithms that appeared in the columns labeled 1 and 2 are ranked next and second next to the algorithm in the first column. Also, the combination of star and number superscripts are used to indicate algorithms whose performance is significantly different from others in the specified columns. For example in Table 9, under the major column for the average waiting time and for problem size=50x1, the appearance of SJF^{*1,2} indicates that the performance of the SJF algorithm with respect to the average waiting time is significantly different from FC (in column labeled 1) and also from PS (in column labeled 2).

In terms of execution speed, the FC algorithm was consistently faster than both the SJF and PS algorithms for all the problem sizes (Table 11). The differences in executions speed of FC compared with PS and SJF is only significant for 40x1 and 50x1 problem sizes (Table 12).

Table 11: Mean of Execution time by algorithms

Problem	Mean of Execution time		
Size	FC	PS	SJF
5x1	0.00141	0.00047	0.00047
10x1	0.00094	0.00094	0.00094
15x1	0.00125	0.00125	0.00156
20x1	0.00187	0.00218	0.00187
25x1	0.00234	0.00374	0.00281
30x1	0.00265	0.00343	0.00359
35x1	0.00312	0.00452	0.00468
40x1	0.00359	0.00468	0.00593
45x1	0.00437	0.00577	0.00515
50x1	0.00468	0.00655	0.00499

Sample size = 50

Table 12: Summary of test of means of Execution time by problem sizes

	Execution time		
Problem Size		1	2
5x1	FC,	PS,	SJF
10x1	FC,	PS,	SJF
15x1	FC,	PS,	SJF
20x1	FC,	SJF ,	PS,
25x1	FC,	SJF ,	PS,
30x1	FC,	PS,	SJF
35x1	FC,	PS,	SJF
40x1	FC^{*2},	PS,	SJF
45x1	FC,	SJF ,	PS,
50x1	FC^{*2},	SJF ,	PS,

Sample size = 50

Note: *2 indicates significant result at 5% level from solution methods in column 2

CONCLUSION

In this paper, the problem of scheduling jobs/processes on the central processing unit (CPU) of the computer system has been discussed. Five goals that are often desired were discussed while three algorithms commonly in use were also discussed. In order to know which algorithm to use for which CPU scheduling goal, a number of randomly generated problems were solved.

Therefore, based on performance, the shortest job first (SJF) algorithm is recommended for the CPU scheduling problems of minimizing either the average waiting time or average turnaround time. Also, the first come first serve (FC) algorithm is recommended for the CPU scheduling problems of minimizing either the average CPU utilization or average throughput.

REFERENCES

Englander, I., 2003. The Architecture of Computer Hardware and Systems Software; An Information Technology Approach, 3rd Edition, John Wiley & Sons, Inc.,

H.H.S. Lee; Lecture: CPU Scheduling, School of Electrical and Computer Engineering, Georgia Institute of Technology.

Silberschatz,A. and P.B. Galvin,1997. Operating System concepts, Fifth Edition, John Wiley & Sons, Inc.,

Stallings,W., 1996. Computer Organization and Architecture; Designing for Performance, Fourth Edition, Prentice Hall

Yavatkar, R. and K. Lakshman,1995. A CPU Scheduling Algorithm for Continuous Media Applications, In Proceedings of the 5th International Workshop on Network and Operating System Support for Digital Audio and Video, pp: 210-213