Volume 3, Issue 5, May 2013 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Analysis of CPU Scheduling Policies through Simulation

Ashok Kumar *
M.Tech. (CSE),
Ch. Devi Lal University,
Sirsa, India.

Dr. Harish Rohil
Assistant Professor
Dept. Computer Sci. & Applications,
CDLU Sirsa , India.

Suraj Arya
HOD & Assistant Professor
Deptt.Computer Sci. & Engg.
BIET Sardulgarh, India

Abstract - CPU Scheduling is an area of research where computer scientists used to design efficient algorithms for scheduling the processes in order to get output in the form of optimum turn around time and average waiting time. There are many CPU scheduling schemes available like FCFS, SJF, RRS, PBS, Multilevel queue scheduling and so on. This paper analysis the four scheduling policies FCFS, SJF, RRS, PBS and gives the result which policy is the best among them. A comparative analysis is made on the basis of data generated through the simulation using exponentially generated random numbers.

Keyword: Simulation, CPU Scheduling Policy, Operating System,

1. INTRODUCTION

In the past, most computers ran standalone, and most operating system was designed to run on a single processor. This situation is rapidly changing into one in which computers are networked together, making distributed operating system more important, Computer software can be roughly divided into two kinds, the system program, which manage the operation of the computer itself, and the application program, which solve problems for their user. The most fundamental of the system program is the Operating System, which controls all the computer's resources and provides the base upon which the application program can be written. The 1960's definition of an operating system is "the software that controls the hardware". But due to a better definition, an operating system is an important part of almost every computer system. A major goal of an operating system is to hide all the complexities and give the programmer a more convenient set of instructions to work with [1][3].

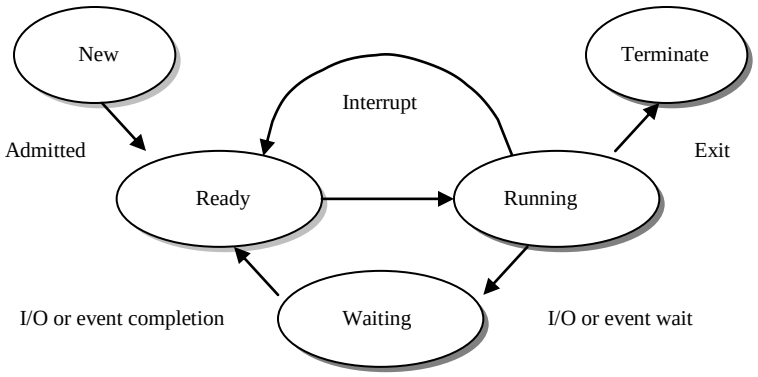


Figure 1 Diagram of process state.

Informally, a process is a program in execution. A process is more than the program code, which is sometime known as the text section. As a process executes, it changes state. The state of a process is defined in part by the current activity of that process. Each process may be in the one of the states as shown in the figure 1.

An operating system is an important part of every computer system. The main purpose of a computer system is to execute programs. These programs must be in main memory during execution. To improve both the utilisation of CPU and the speed of its response to users, the computer must keep several processes in memory. Many memory schemes exist and each algorithm depends on situation. In present research work, simulator has been designed for analysing the performance of

© 2013, IJARCSSE All Rights Reserved

different memory CPU scheduling policies. Simulator takes burst time of process, compare with in each-other. CPU scheduling deals with the problem of deciding which of the process in the ready queue is to be allocated the CPU.

(a) First come First Serve (FCFS) Scheduling

By far the simplest CPU scheduling algorithm is the first-come, first served scheduling (FCFS) algorithm .with this scheme; the process that requests the CPU first is allocated the CPU first. The implementation of the FCFS policy is easily managed with a FIFO queue. When a process enters the ready queue, its PCB is linked onto tail of the queue. When the CPU is free, it is allocated to the process at the head of the queue. The running process is then removed from the queue [1][3].

(b) Shortest Job First (SJF) scheduling

This algorithm associates with each process the length of the latter's next CPU burst. When the CPU is available, it is assigned to the process that has the smallest next CPU burst. If two processes have the same length next CPU burst. Note that a one appropriate term would be the shortest next CPU burst, because the scheduling is done by examining the length of the next CPU burst of a process, rather than its total length[1][3].

(c) Priority Scheduling Algorithm

When more than one process is runnable, the Operating system must decide which one to run first. The part of the Operating system concerned with this decision is called the Scheduler, and the algorithm is called the scheduling algorithm. In the Priority Scheduling each process is assigned a priority, and the runnable process with the highest priority is allowed to run first [1].

(d) Round Robin Scheduling

It is one of the oldest, simplest, and fairest and most widely used scheduling algorithms, designed especially for time-sharing systems. Small units of time, called time slice or quantum is defined. All runnable processes are kept in a circular queue. The CPU scheduler goes around this queue, allocating the CPU to each process for a time interval of one quantum. New processes are added to the tail of the queue. The CPU scheduler picks the first process from the queue, sets a timer to interrupt after one quantum, and dispatches the process. If the process is still running at the end of the quantum, the CPU is preempted and the process is added to the tail of the queue. If the process finishes before the end of the quantum, the process itself releases the CPU voluntarily. In either case, the CPU scheduler assigns the CPU to the next process in the ready queue. Every time a process is granted the CPU, a context switch occurs, which adds overhead to the process execution time. [1][3].

2. RELATED WORK

Terry Regner & Craig Lacey [5] has introduced the concepts and fundamentals of the structure and functionality of operating systems. The purpose of this article was to analyze different scheduling algorithms in a simulated system. This article has the implementation of three different scheduling algorithms: shortest process first, round robin, and priority sequence. Comparing the three algorithms they find that the CPU utilization values indicate that the shortest process first has the highest throughput values with CPU utilization times comparable to those of the round robin. Nazleeni Samiha Haron et.al. [6] has analyzed distributed systems, process scheduling plays a vital role in determining the efficiency of the system. Process scheduling algorithms are used to ensure that the components of the system would be able to maximize its utilization and able to complete all the processes assigned in a specified period of time. Ajit Singh [7] has developed a new approach for round robin scheduling which help to improve the CPU efficiency in real time and time sharing operating system. Kadhim [8] addresses different problems of task scheduling in computer-based systems. In some applications, SJF scheduling algorithm is more suitable than PrS algorithm since it provides less waiting time and less turnaround time. In real-time applications, PrS algorithm must be used to deal with different priorities, since each task has a priority order. Alexander [9] has stated that Multimedia applications have unique requirements that must be met by network and operating system components. In any multimedia application, we may have several processes running dependently on one another. Multimedia is a real-time application. In context of multimedia applications, the CPU scheduler determines quality of service rendered. The more CPU cycles scheduled to a process, the more data can be produced faster, which results in a better quality, more reliable output. Lalit [4] discussed about various types of scheduling. A comparison of various types of algorithms is also shown with practical implementation using MATLAB. By this experimental setup he has been able to do statistical analysis of the performance of all the four basic scheduling algorithms.

3. DESIGN OF SIMULATOR

The simulator is designed using VB 6.0. The simulator analysis four CPU Scheduling policies. CPU scheduling policies is most important function and also critical part of an operating system. There are several policies of process allocation such as FCFS, SJF RRS and PBS. Simulator is designed to evaluate the process schedule strategies by considering randomly generated reference Process. Burst time of processor is generated by random procedure. Assumption is that burst time of process has been randomly generated. It is also assumed that burst time of process will be generated between 1 and 50 numbers. Number of process will be 5. The different algorithms are then compared on the basis of the burst time and average waiting time. The present research work uses an efficient random number generator that produces a random references string of size of process and size of memory. A random number string can be generated that follows a particular distribution such as exponential, poison, Beta, Gamma, Normal etc. In the present research work exponential distribution has been used to generate random numbers. Reference string of process size are generated randomly and distributed exponentially. Here

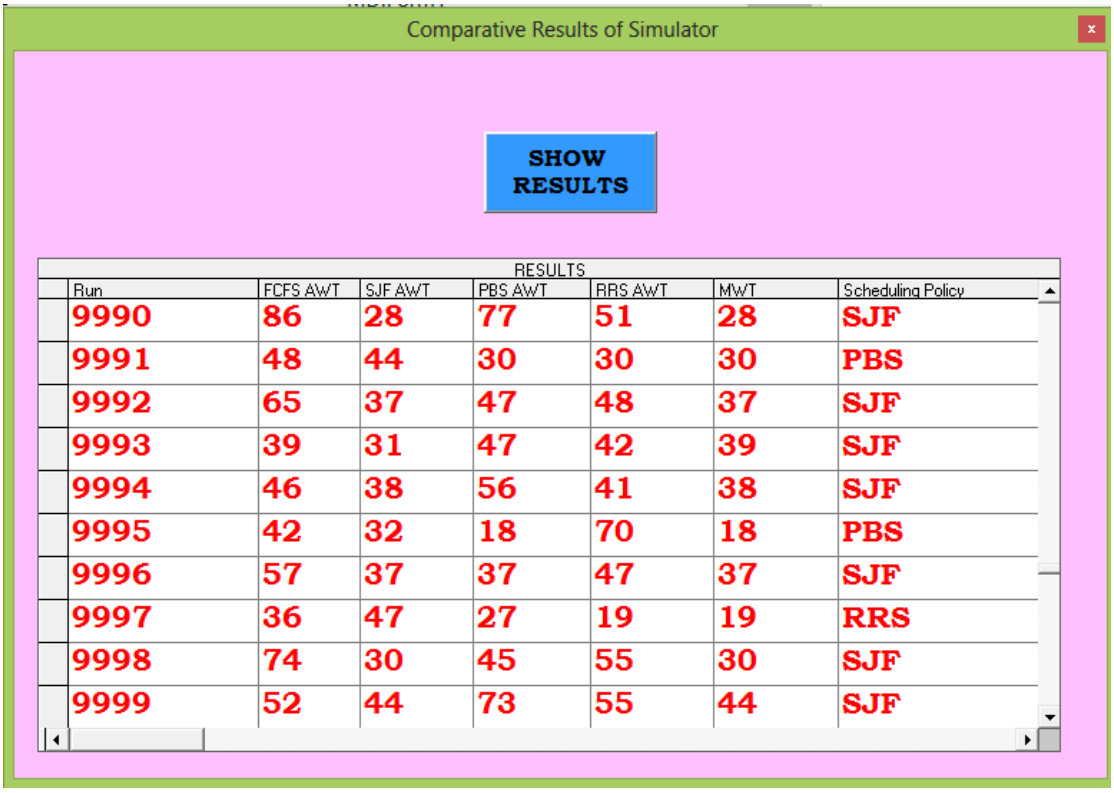
© 2013, IJARCSSE All Rights Reserved

exponential distribution has been used to model inter arrival times when arrivals are completely random and to modal service times that are highly variable.

4. EXPERIMENT

As discussed earlier the main objective of the present research work is to analyze various policies of CPU scheduling. The foremost criterion for the evaluation of CPU scheduling is the waiting time and burst time of the processes that are produced by each policy under same set of conditions and workload. The workload here is the size of memory whose allocate to coming process. Than simulator has been designed to study the behavior pattern of different policies under similar conditions and for a particular set of requested process, which are generated randomly. In present research work here waiting time are considered to evaluate the performance of policies. Therefore number of comparisons is considered here to calculate the waiting time. Therefore process size and burst times are central point of consideration. Burst time takes in millisecond. In present research work following assumptions are made.

- **Sizes of processes (CPU Burst time in millisecond) have been generated between 1 to 50.**
- **During a simulation run size of processes is generated randomly using random number generator that generates exponentially distributed CPU Burst time.**
- **5 Number of Processes have been generated in each simulation run.**
- **Size of process can not be a zero.**
- **Priorities of 5 processes lie between 1 and 10 depending on their CPU times.**
- **No two processes can have same priority.**



Run	FCFS AWT	SJF AWT	PBS AWT	RRS AWT	MWT	Scheduling Policy
9990	86	28	77	51	28	SJF
9991	48	44	30	30	30	PBS
9992	65	37	47	48	37	SJF
9993	39	31	47	42	39	SJF
9994	46	38	56	41	38	SJF
9995	42	32	18	70	18	PBS
9996	57	37	37	47	37	SJF
9997	36	47	27	19	19	RRS
9998	74	30	45	55	30	SJF
9999	52	44	73	55	44	SJF

Figure 2 Snapshot of Simulator showing Comparative Result of CPU Scheduling Policies.

Figure 2 show the data generated using the exponentially random method. In visual basic 6.0 we use the Rnd() function for generate the data. 10,000 runs of four policies are show in figure 2 using the grid.

5. RESULTS AND DISCUSSION

The simulator is run 10000 times. Each times a burst time is generated. The number of average waiting time under different policies has been produced in the grid. During each run, the size of process is generated randomly. The module has been executed; average waiting time is calculated by each policies are produced. We are comparing these four policies according to the average waiting time. Firstly, we compare the average waiting time of four policies. Than we gives these results and show that which process gives the minimum average waiting time out of 10000 runs.

The figure 3 shows the result of four processes in 10 runs using the data given in table 1

© 2013, IJARCSSE All Rights Reserved

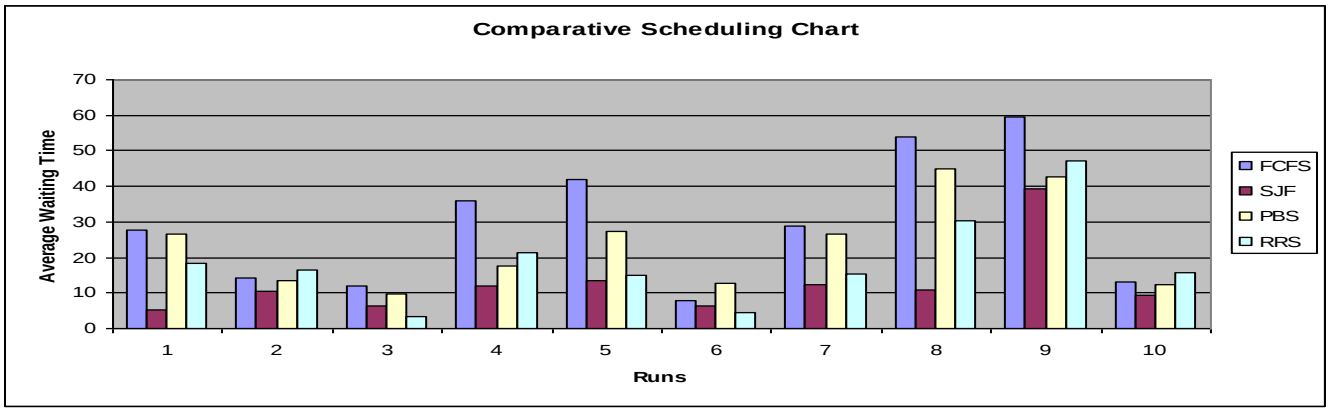


Figure 3 Comparison between Scheduling Policies in Graphical Representation

Table 1 contains the data of 4 scheduling policy in 10 runs. These data also show the minimum average waiting time of the process as shown in table 1, SJF produce the minimum average waiting time 5.4 in first run, in second run it is 10.4 and so on. A graphical representation is shown in the figure 3 with the help of data given in the table 1.

Table 1 Average Waiting Time of First Ten Processes

Runs	FCFS	S J F	P B S	RRS
1	27.8	5.4	26.4	18.4
2	14.2	10.4	13.6	16.6
3	11.8	6.4	9.8	3.2
4	36	11.8	17.6	21.4
5	42	13.4	27.2	14.8
6	7.8	6.4	12.8	4.4
7	28.8	12.4	26.4	15.4
8	54	11	45	30.4
9	59.4	39.4	42.8	47.2
10	13	9.2	12.2	15.8

In First Come first scheduling, which process come in first in the ready queue that takes the CPU first and firstly execute. But in the Shortest Job First which process has take less time that will execute firstly. In priority based which has given highest priority is firstly execute. In round robin scheduling a time quantum is defined and process execute under this time quantum if process is big then the time quantum than it is again implemented in the ready queue for execution. If the average waiting time is less in any scheduling, is proved to be best scheduling policy .The simulator is run 10000 times. The reason for 10000 runs is the need to overcome the problem of chance factor.

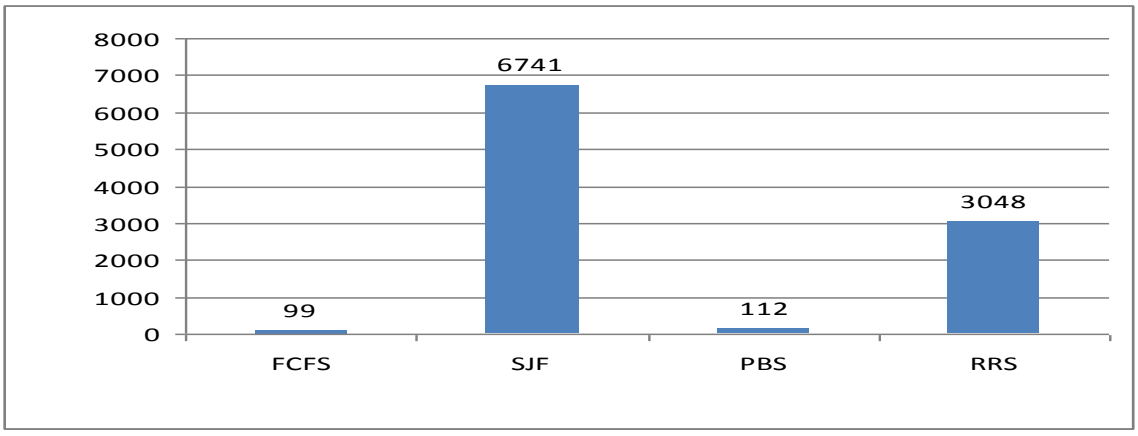


Figure 4 Graphical Representation of the Best Scheduling Policy.

© 2013, IJARCSSE All Rights Reserved

Figure 4 show the best scheduling policy among the four scheduling policy. Out of 10,000 simulation runs, FCFS scheduling was reported the best for only 99 times, SJF scheduling policy was reported to be the best for 6741 times, PBS scheduling was reported the best for the 112 times and round robin scheduling was reported the best for the 3048 times. This is clearly shows that SJF was reported the best policy for the maximum times out of 10000 run which is best for 6741 times.

6. CONCLUSION

The present paper analysis the four CPU scheduling policies viz FCFS, SJF, RRS, PBS. The comparative analysis was made using simulation. Simulation was designed and implemented in Visual Basic 6.0. In the simulator 10,000 simulation runs are performed and it was observed that average waiting time under Non-Pre-emptive SJF is least most of the times and hence it was the best CPU Scheduling policy.

REFERENCES

[1] Silberschatz,Galvin,Gagne.A. P.B; Operating System Concept 6th edition Prentice Hall; 2001. India, New Delhi.

[2] MiroSlav Ruda; Grid Simulator with production scheduling algorithms; 2007. University Cracow (Available at www.fi.muni.cz/~hanka/publ/cgw07.pdf.)

[3] Andrew S.Tanenbaum; Modern Operating System; 2008. Prentice Hall.

[4] Lalit Kishor, Dinesh Goyal; Comparative Analysis of Various Scheduling Algorithms; April 2013.(Available at www. http://ijarcet.org/index.php/ijarcet/article/view/904/PDF).

[5] Terry Regner, craig lacy; An Introductory Study of Scheduling Algorithms; feb 2005. (A project Assignment available at www.regner.ca/documents/An%20Introductory%20Study%20of%20Scheduling%20Algorithms.pdf).

[6] Nazleeni Samiha Haron, Anang Hudaya Muhamad Amin; Time Comparative Simulator for Distributed Process Scheduling Algorithms;2006. (Available at https://www.waset.org/journals/waset/v19/v19-162.pdf).

[7] Ajit Singh, Priyanka Goyal ; An Optimized Round Robin Scheduling Algorithm for CPU Scheduling; 2010. (Available at www.enggjournals.com/ijcse/doc/IJCSE10-02-07-83.pdf.).

[8] Shatha J. Kadhim , Kasim M. Al-Aubidy; Design and Evaluation of a Fuzzy-Based CPU Scheduling Algorithm; 2010. (Available at http://www.philadelphia.edu.jo/research_papers/Design_and_Evaluation_of_a_Fuzzy_Based_CPU.pdf).

[9] Alexander Taranovsky ; CPU Scheduling in Multimedia Operating Systems; 1999. **(Available at www.dgp.utoronto.ca/~dannyt/os_res5.pdf).**

© 2013, IJARCSSE All Rights Reserved