ISSN 2319-2720

Volume 2, No.1, January – March 2013
International Journal of Computing, Communications and Networking
Available Online at http://warse.org/pdfs/2013/ijccn04212013.pdf

AN ANALYSIS OF SCHEDULING STRATEGIES BASED ON CRITICALITY OF JOBS

Raj Mohan Singh* and Harsh K Verma*
*Dr. B R Ambedkar National Institute of Technology Jalandhar

Abstract:
In this paper we will analyze the performance of job scheduler based on many parameters considering the criticality of job. Many job scheduling algorithms have been devised which affect the performance of the system in their own way. Improvement in job scheduling strategies will play a pivotal role in increasing the overall performance of the system. In this work we discuss some basic job scheduling strategies and also propose a new scheduling strategy which is based on the criticality i.e. how much important the job is for the user and priority of jobs with an effort towards improving the response time of the jobs. The idea is to motivate the users to submit more jobs and to minimize the chances of the users leaving the session. Interactive jobs usually require much less resources and are much more critical to the users than the batch jobs that execute over nights and weekends. The jobs are executed by first applying criticality to round robin scheduling and then applying priority to round robin scheduling. These scheduling strategies are then compared and their performance is evaluated on the basis of the three parameters viz. average waiting time, average turnaround time, and average response time. It is found that by applying criticality and priority on round robin scheduling there is significant improvement in the values of the three parameters especially the response time.

1. INTRODUCTION

Scheduling is the method by which threads, processes or data flows are given access to system resources. The need for a scheduling algorithm arises from the requirement for most modern systems to perform multitasking (execute more than one process at a time) and multiplexing (transmit multiple flows simultaneously). When a computer is multiprogrammed, it frequently has multiple processes competing for the CPU at the same time. This situation occurs whenever two or more processes are simultaneously in the ready state. If only one CPU is available, a choice has to be made which process to run next. The part of the operating system that makes the choice is called the scheduler and the algorithm it uses is called the scheduling algorithm. Because CPU time is a scarce resource on these machines, a good scheduler can make a big difference in perceived performance and user satisfaction. Consequently, a great deal of work has gone into devising clever and efficient scheduling algorithms [1]. Many criteria have been suggested for comparing CPU scheduling algorithms. Which characteristics are used for comparison can make a substantial difference in which algorithm is judged to be best. Most commonly used metrics are CPU utilization, throughput, turnaround time, waiting time and response time [2].

2. LITERATUE SURVEY

The problem of job scheduling is to determine how that sharing should be done in order to maximize the system's utility. How deployed scheduling policies can be improved to meet existing requirements needs to be discussed [3].
Fairness is an important issue for parallel job scheduling policies, but has been ignored in most of the previous studies. Commonly used summary statistics are applied to different job measures to evaluate the fairness under a wide range of non-preemptive parallel job scheduling policies. The impact of fairness on other scheduling performance needs to be studied [4].

User estimates of job runtimes have emerged as an important component of the workload on parallel machines, and can have a significant impact on how a scheduler treats different jobs, and thus on overall performance. It is therefore highly desirable to have a good model of the relationship between parallel jobs and their associated estimates [5].

Scheduling parallel jobs has been a popular research topic for many years. A couple of surveys have been written on this topic in the context of parallel supercomputers. The purpose is to update this material, and to extend it to include work concerning clusters and the grid [6]. A hierarchical multiprocessor scheduling (H-SMP), a novel hierarchical CPU scheduling algorithm designed for a symmetric multiprocessor (SMP) platform is also studied. The novelty of this algorithm lies in its combination of space and time multiplexing to achieve the desired bandwidth partition among the nodes of the hierarchical scheduling tree. This algorithm is also characterized by its ability to incorporate existing proportional-share algorithms as auxiliary schedulers to achieve efficient hierarchical CPU partitioning [7]. The schedulability problem of periodic and sporadic real-time task sets with constrained deadlines preemptively scheduled on a multiprocessor platform composed by identical processors. Two typical scheduling algorithms: Earliest Deadline First (EDF) and Fixed Priority (FP) are discussed [8].

The behaviour of the schedulers upon job arrival differs greatly. Most of the schedulers maintain a queue where the jobs wait for processors to become available and whenever the state of the system changes, either due to an arrival of a new job, or a termination of a running job, they scan the queue and select jobs for execution. It is difficult to determine which approach is the best, and in fact some studies have indicated that the relative performance of schedulers may actually depend on the workload. A scheduler known as CREASY (CR stands for criticality) that exploits knowledge on user behaviour in order to improve user satisfaction is discussed [9].

Balancing fairness, user performance, and system performance is also a critical concern when developing and installing parallel schedulers [10].The performance of scheduling algorithms is analyzed with respect to fairness. Existing works frequently consider fairness as a job related issue but in their work they analyze fairness with respect to different users of the system as this is a very important real-life problem. Notably, the fairness is considered as an important metric, which accompanies standard performance related metrics such as slowdown or wait time. The aim is to maintain fairness among different users of the system while keeping good performance regarding classical criteria such as slowdown or wait time [11].

3. OBJECTIVES

Users continually submit jobs to the system each having unique resource and service-level requirements as well as value to the user and the owner of the resource. The charge of job scheduling therefore, is to decide when and how each job should execute in order to maximize the system's utility to its owners. The field of job scheduling has been the subject of active research for well over a decade producing a sizeable body of literature.

In this paper we have proposed a scheduling strategy that can be used to decrease the response time of the job that is more critical to the user and is also senior as compared to the other jobs. We have defined three parameters viz. turnaround time, response time and waiting time on the basis of these parameters we have evaluated our strategy.

4. METHODOLOGY

In order to study and evaluate the performance of various scheduling strategies we have taken a data set of 10 jobs with four parameters as discussed below. The parameters are defined as follows:

1. Burst time, which is the time taken to execute a particular job.
2. Seniority i.e. which job arrives first in the queue for execution.
3. Criticality i.e. How much important the job is to the user.
4. Priority i.e. the combination of seniority and criticality.

In order to evaluate the performance of the job scheduling algorithms we have taken three parameters viz. average waiting time, average turnaround time and average response time. The data set that we have used to evaluate the scheduling strategies is shown below:

Table1: Depicting the Data Set

JOBS	BURST TIME	SENIORITY/ ARRIVAL TIME	CRITICALITY	PRIORITY
J1	4	2	7	9
J2	22	5	1	6
J3	3	1	5	6
J4	2	3	8	11
J5	6	7	2	9
J6	1	4	9	13
J7	8	8	3	11
J8	5	6	4	10
J9	12	9	6	15
J10	2	10	10	20

We have implemented the scheduling strategies using .NET Framework. .NET is an integral part of many applications running on Windows and provides common functionality for those applications to run. The .NET Framework consists of the common language runtime and the .NET Framework class library. The common language runtime is the foundation of the .NET Framework. You can think of the runtime as an agent that manages code at execution time, providing core services such as memory management, thread management, and remoting, while also enforcing strict type safety and other forms of code accuracy that promote security and robustness. The implementation has been done in C# using visual studio. Simulation has been done for the different job scheduling strategies like FCFS (First-Cum-first-Serve), SJF (Shortest Job First), Criticality, Priority, Criticality on Round Robin and lastly Priority on Round Robin for three parameters viz. average waiting time, average response time and average turnaround time. These parameters are the benchmark for any scheduling strategy in order to evaluate their performance.

5. RESULT AND DISCUSSION

Various scheduling strategies have been simulated using the interface as shown below:

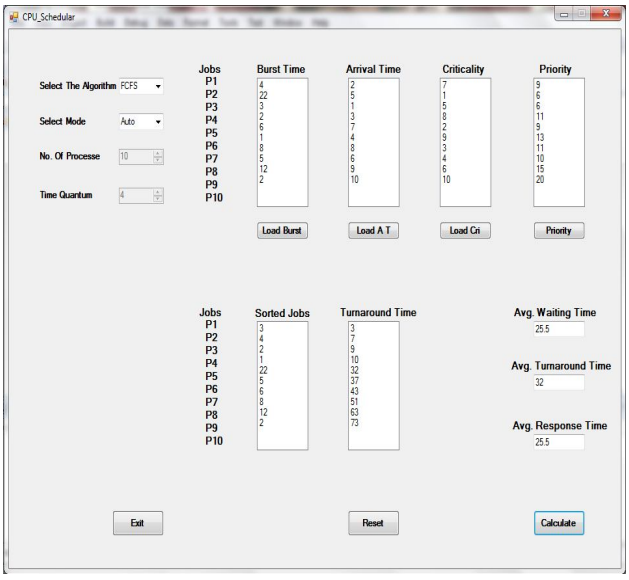

Figure 5.1: Simulation of jobs with FCFS

The figure 5.1 shows the simulation of jobs with FCFS strategy. The required scheduling strategy is selected along with the mode of operation which is set to auto in which a set of ten jobs are selected. The mode of operation of the interface is selected as auto. Next we load the burst time, arrival time, criticality and priority of jobs according to our data set by clicking at the appropriate buttons. The turnaround time for the individual jobs is calculated and is shown in the list box named turnaround time. The values of the three parameters viz. average waiting time, average turnaround time and average response time are calculated and the values are shown in the text boxes of the interface.

The figure 5.2 shows the simulation of jobs with SJF strategy for ten jobs. We load the burst time, arrival time, criticality and priority of jobs by clicking at the appropriate buttons. The turnaround time for the individual jobs is calculated and is shown in the list box named turnaround time. The values of the three parameters viz. average waiting time, average turnaround time and average response time are calculated.

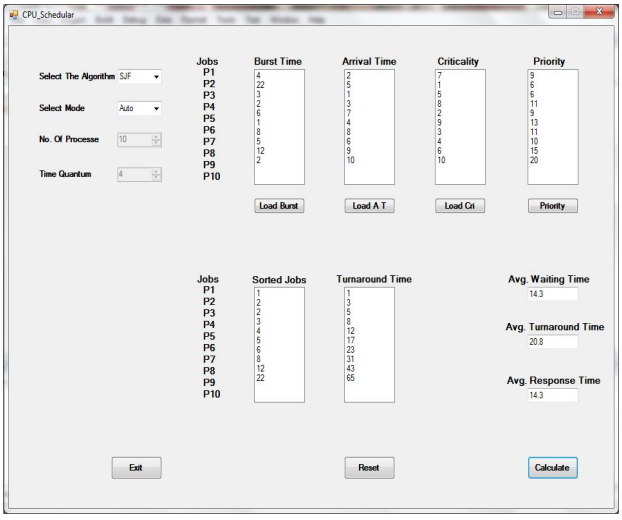

Figure 5.2: Simulation of jobs with SJF

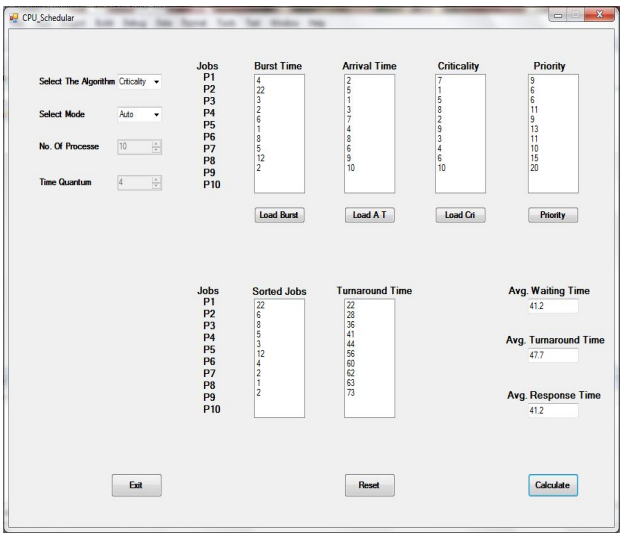

Figure 5.3: Simulation of jobs with Criticality

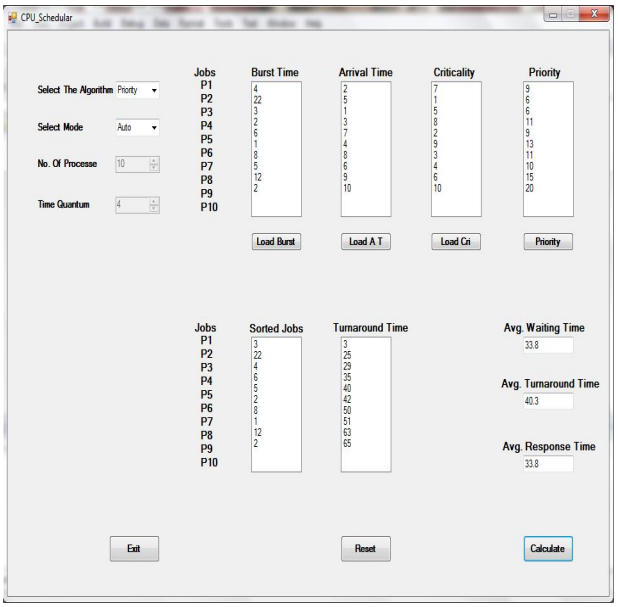

Figure 5.4: Simulation of jobs with Priority

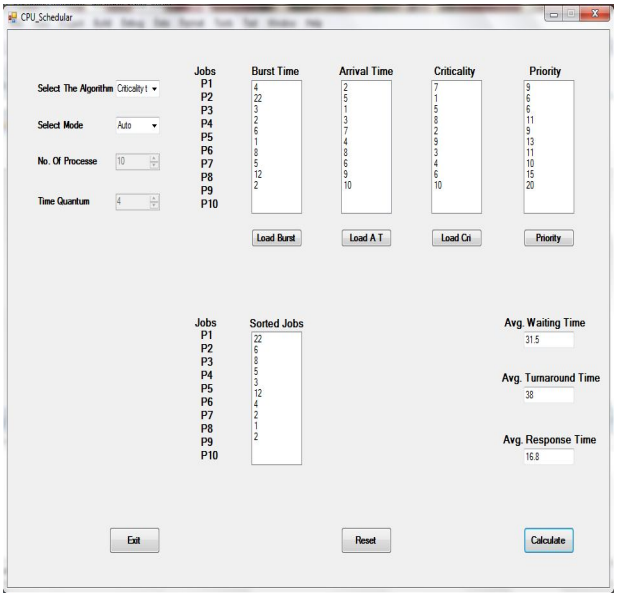

Figure 5.5: Simulation of jobs by applying Criticality to RR

The figures 5.3 to 5.6 show the simulation of jobs with Criticality, Priority, Criticality on Round Robin, and Priority on Round Robin respectively. The required scheduling strategy is selected along with the mode of operation which is set to auto in which a set of ten jobs are selected. We load the burst time, arrival time, criticality and priority of jobs according to our data set. The turnaround time for the individual jobs is calculated and shown in the list box named turnaround time. The values of the three parameters viz. average waiting time, average turnaround time and average response time are calculated and shown.

5.1 COMPARISON OF SCHEDULING STRATEGIES

We have discussed about the criticality of jobs and also applied the concept of criticality and priority to some of the scheduling schemes. We have calculated the values of the three parameters viz. average turnaround time, waiting time

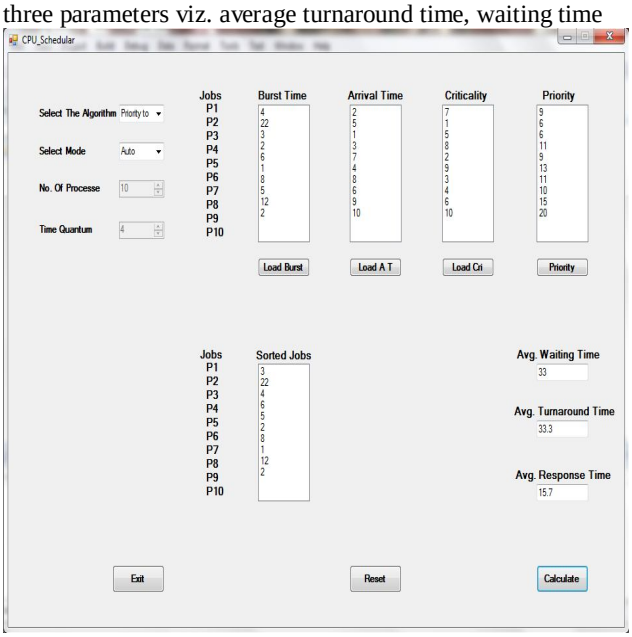

Figure 5.6: Simulation of jobs by applying Priority to RR

and response time for all the scheduling strategies discussed in the previous chapter and the results are shown in the following table as follows:

Table 5.1: Comparison of all Scheduling Strategies

SCHEDULING STRATEGY	AVERAGE WAITING TIME	AVERAGE TURNAROUND TIME	AVERAGE RESPONSE TIME
FCFS	25.5	32.0	25.5
SJF	14.3	20.8	14.3
CRITICALITY	41.2	47.7	41.2
PRIORITY	33.8	40.3	33.8
RR + CRITICALITY	31.5	38.0	16.8
RR + PRIORITY	26.8	33.3	15.7

5.2 DISCUSSION

5.2.1 Comparison of Average Waiting Time

(i) When the jobs are executed according to First-Come-First-Serve strategy the average waiting time is less if we compare it to other strategies, but it does not take into account the criticality of jobs.

(ii) When the jobs are executed according to SJF strategy, we get the lowest average waiting time, but SJF may lead to starvation and this strategy also does not take into account the criticality of jobs.

(iii) When we execute the jobs according to Criticality the average waiting time comes to be much higher, but at the same time the jobs which are critical to the user will have not have to wait more.

(iv) When we execute the jobs according to Priority the average waiting time comes to be lower than in the case of criticality.

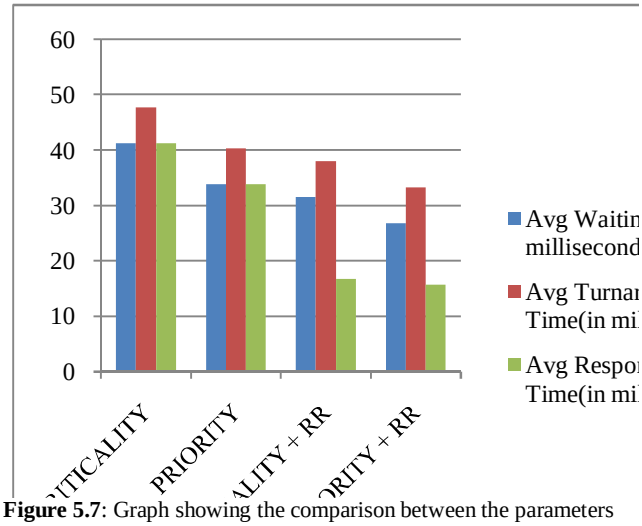

Figure 5.7: Graph showing the comparison between the parameters

(v) We now execute the jobs by applying Round robin on Criticality. We find that there is reduction in average waiting time.

(vi) To improve the average waiting time further we have considered the priority of jobs and applied Round Robin on it, which results in much lesser waiting time.

5.2.2 Comparison of Average Turnaround Time

(i) When the jobs are executed according to First-Come-First-Serve strategy the average turnaround time is less if we compare it to other strategies, but it does not take into account the criticality of jobs.

(ii) When the jobs are executed according to SJF strategy, we get the lowest turnaround time, but SJF may lead to starvation of large jobs. This strategy also does not rank the jobs according to their criticality.

(iii) When we execute the jobs according to Criticality the average turnaround time comes to be much higher, but at the same time the jobs which are critical to the user will execute early.

(iv) When we execute the jobs according to Priority the average turnaround time comes to be lower than in the case of criticality.

(v) We now execute the jobs by applying Round robin on Criticality. We find that there is reduction in average turnaround time.

(vi) To improve the average waiting time further we have considered the priority of jobs and applied

Round Robin on it, which results in much lesser turnaround time. The jobs that have the highest seniority and criticality will get the highest priority which will help in reducing the turnaround time for the jobs that are more critical to the user and moreover it does not lead to starvation.

5.2.3 Comparison of Average Response Time

(i) When the jobs are executed according to First-Come-First-Serve strategy the average response time is less if we compare it to other strategies, but it is not optimal for jobs which are critical to the user.

(ii) When the jobs are executed according to SJF strategy, we get the lowest response time, but SJF may lead to starvation. It does not take into account the issue of criticality of jobs.

(iii) When we execute the jobs according to Criticality the average response time comes to be higher, but at the same time the jobs which are critical to the user will have less response time.

(iv) When we execute the jobs according to Priority the average response time comes to be lower than in the case of criticality.

(v) We now execute the jobs by applying Round robin on Criticality. We find that there is reduction in average response time.

(vi) To improve the average response time further we have considered the priority of jobs and applied Round Robin on it, which results in much lesser response time. The jobs that have the highest seniority and criticality will get the highest priority which will help in reducing the response time for the jobs that are more critical to the user and moreover it does not lead to starvation.

6. CONCLUSION AND FUTURE WORK

In this paper we have proposed a scheduling strategy that can be used to decrease the response time of the job that is more critical to the user and is senior as compared to other jobs as well. Three parameters have been defined on the basis of which we have evaluated the strategy. After analyzing the three parameters on the basis of different strategies it is observed that there is need to make modifications in the scheduling strategy so that more and more users submit the jobs and the chances of the user leaving the session are minimized.

The performance of our scheduling strategy also needs to be evaluated with respect to parallel systems and the effects studied. The jobs which the users submit during the day are known to be different from those submitted during the night. Interactive jobs usually require much less resources and are much more critical to the users than the batch jobs that execute over nights and weekends. Such factors should also be taken into consideration.

Finally the scheduling strategy will need to be revised to consider the aggregate effect of all these factors on the users, and its performance will need to be evaluated again by extending the number of jobs to demonstrate that it can still significantly improve user productivity. This task alone is extremely challenging, but it is a necessary step towards improving the performance of job scheduling algorithms.

REFERENCES

[1] Anrew S Tanenbaum. **Modern Operating Systems,** 2nd ed. Prentice Hall Press, 2007, ch. 2, pp 181.

[2] Abraham Silberschatz, Peter Baer Galvin. **Operating System Concepts**, 5th ed. John Wiley & Sons, Inc, ch. 5, pp 127, 1999.

[3] Jonathan Weinberg. **Job Scheduling on Parallel Systems**, in Proc. *International Conference on Job Scheduling Strategies for Parallel Processing (IPPS)*, 2002.

[4] Sangsuree Vasupongayya, Su-Hui Chiang. **On Job Fairness in Non-Preemptive Job Scheduling**, in *Proc. International Conference on Parallel and Distributed Computing*, Phoenix, AZ, USA, Nov 14-16, 2005.

[5] Dan Tsafrir, Yoav Etsion, Dror G Feitelson. **Modeling User Runtime Estimates**, in *Proc. 11th International Conference on Job Scheduling Strategies for Parallel Processing*, Springer-Verlag Berlin, Heidelberg, pp 1-35, 2005.

[6] Dror G. Feitelson, Larry Rudolph, Uwe Schwiegelshohn. **Parallel Job Scheduling-A Status Report**, in *Proc. 10th International Conference on Job Scheduling Strategies for Parallel Processing*, Springer-Verlag Berlin, Heidelberg, pp 1-16, 2005.

[7] Abhishek Chandra, Prashant Shenoy. **Hierarchical Scheduling for Symmetric Multiprocessors**, *IEEE Transactions on Parallel and Distributed Systems*, Vol 19,Issue 3, pp 418-431, 2008.

[8] Marko Bertogna, Michele Cirinei, Giuseppe Lipari. **Schedulability Analysis of Global Scheduling Algorithms on Multiprocessor Platforms**, *IEEE Transactions on Parallel and Distributed Systems*, pp 553-566, 2009.

[9] Edi Shmueli, Dror G. Feitelson. **On Simulation and Design of Parallel-Systems Schedulers: Are We Doing the Right Thing?** *IEEE transaction on Parallel and Distributed System*, Vol. 20, No. 7, pp 983-996, July 2009.

[10] Vitus J Leung, Gerald Sabin, P Sadayappan. **Parallel Job Scheduling Policies to Improve Fairness: A Case Study**, in *Proc.39th International Conference on Parallel processing workshops*, pp 346-353, 2010.

[11] Dalibor Klusacek, Hana Rudova. **Performance and Fairness for Users in Parallel Job Scheduling**, in *Proc. 16th workshop on job scheduling strategies for parallel processing*, IPDPS, Shanghai, China 25 May, 2012.