ISSN 0974 - 9373

Vol. 14 No.1 (2010) **Journal of International Academy of Physical Sciences** pp. 101-112

Weighted Approach to Better Job First Scheduling*

D. Pandey
Department of Mathematics
C. C. S University Meerut(India)
Email: pandey_diwakar2k1@rediffmail.com

Vandana
Department of Computer Applications
BIT Meerut (India)
Email: ranavd@rediffmail.com

(Received January 19, 2010)

Abstract: Order of arrival, CPU time requirement and priority are three important factors that are usually attached with a process arriving for execution. Several algorithms have been developed to order the processes in ready queue in an attempt to minimize average waiting time, response time or increasing the throughput. Existing algorithms have some positive and negative effects by way of assigning individual importance to any of the three factors. Present work is an effort to analyze the collective effect of time of arrival, size of CPU burst and priority of the process, through a logical combination of all the three. Proposed approach is based on a suitable classification of the attributes and ranking the processes according to logically assigned weights. The policy thus enjoys some advantages of all the criteria.
Keywords: Process, CPU scheduling, Ready queue

1. Introduction

Modern operating systems have become more complex than ever before. They have evolved from a single task, single user architecture to a multitasking environment in which processes run in a concurrent manner and share the resources. Scheduling is one of the important functions of any operating system that contributes substantially on overall performance of the system. One of the major tasks of traditional general-purpose operating system is to provide ordered and controlled allocation of processor in a fair and efficient manner among various executing programs. Therefore CPU allocation to a process requires careful attention to ensure fairness and to

*Presented at CONIAPS XI, University of Allahabad, Feb. 20-22, 2010.

avoid process starvation for CPU. A good scheduling algorithm is one that is able to optimize some of the performance measures like waiting time, turnaround time or throughput. Several algorithms for CPU scheduling exist in the literature having advantages and disadvantages of different kinds. Thus choice of an algorithm is based on certain desired qualities. Some commonly used important qualities are listed below.

CPU utilization- CPU utilization means to keep CPU busy. It is desirable to have a higher percent of CPU utilization.

Throughput- The number of processes completed per unit of time indicates the throughput. A good CPU scheduling algorithm tries to maximize the throughput of a system.

Turnaround time- It is the measure of time taken between the submission of a process and the completion its execution. A good CPU scheduling algorithm minimizes this time.

Waiting time- This measures the time a process spends in ready queue.

Response Time- It is the amount of time it takes to start responding to a request. This criterion is useful for interactive systems.

Fairness- This amounts to giving a fair share of CPU to each process such that no process suffers indefinite postponement.

A good scheduling algorithm is one that is able to optimize these performance measures.

When a process is submitted for the purpose of execution its time of arrival, size of its CPU burst and the assigned priority are the major considerations that are generally attached with it. Each of these factors has their own importance. There are some basic algorithms that possess considerations for one or the other. FCFS looks for arrival time, SJF bothers about the size of burst and priority scheduling gives preference in accordance with the assigned priorities. In FCFS, processes are executed in the order of arrival, without any regard to its size and priority. It is thus particularly troublesome for time-sharing systems, where it is important that each user gets a share of CPU at regular intervals. It would be disastrous to allow one process to keep CPU busy for an extended period of time. Round robin is a straightforward way to reduce the penalty suffered by short jobs in FCFS policy through preemption based on clock. RR policy is basically based on the order of arrival of the process, yet it is effective in general purpose time-sharing system or transaction processing system due to its preemptive nature after a fixed quantum. One drawback to round robin is its relative treatment of processor-bound and I/O bound processes. Some variants of this algorithm combine the features of two or more policies.

Shortest job first approach reduces the biasness in favor of long processes inherent in FCFS. But this algorithm focuses only on smallest CPU burst, without any concern to the priority or its arrival order. Conventional operating systems employ numerical priorities for scheduling processes[1].

A priority scheduler simply grants the processor to the process with the highest priority. Thus, priorities represent absolute resource rights, since a process with higher priority is given absolute precedence over a process with lower priority. Unfortunately, several significant problems are associated with priority scheduling. One of the most severe problems with a pure priority scheduling scheme is that lower-priority processes may suffer starvation, if there is always a steady supply of higher-priority ready processes. Priority aging is a common technique that gradually increases the priorities of processes that have been waiting for execution for a long time[2].

Having observed several positive and negative implications of assigning individual importance to arrival order, burst length or priorities in different scheduling algorithms, it is but proper to study and analyze their collective effect through a logical combination of all the three. This will help the scheduler in determining next most worthy job to be executed. We, therefore, propose an algorithm that combines arrival order, CPU burst length, and priority of the process together to select a better job to be executed. Al-Husainy[3] has done an effort to mix the properties of some basic scheduling algorithms. He introduced a new factor *f* that represented the mixed effect produced from the combination of three factors. His approach relies on the term percentage ratio that involves imprecision in its calculation and judgment for appropriateness. This paper suggests a scheduler that uses weighted technique to combine the importance attached to the arrival order; CPU burst length, and priority of the process to select a better job to be executed next. The proposed approach logically assigns weights to each process according to some specified classifications of the three attributes and then ranks the processes according to their combined outcome. Average waiting times have been computed for different randomly generated data-sets and the results have been compared with FCFS, SJF and round robin policies to justify this work. Results obtained through our technique have also been compared in three possible cases.

The contents of this paper are organized as follows: In Section 2, weighted technique for better job first has been introduced. Section 3 defines the algorithm and its method of implementation. Results on different data-sets have been discussed in Section 4, while the last section highlights the conclusions.

2. Weighted Technique for BJF

Weighting technique is used to order processes for the purpose of execution by considering the combined effect of the three attributes. Selection of better job for execution is made through rule base. Output values to the rule base are assigned on the basis of logical reasoning. First of all, time of arrival, size of CPU burst and priority are classified into linguistic categories. Numeral weights, starting from 1 onwards, are assigned to each category of an attribute in the decreasing order of suitability of the category in scheduling. For example- for short, medium and long CPU bursts numeral weights 1, 2 and 3 may be assigned respectively, whereas in case of priority lower weight is assigned to higher priority process. The output value of the logical rule combining some category of every attribute is obtained as the sum of the numeral weights of all categories participating in the rule. Processes can then be ordered in ascending order of their output values. Thus a process with lowest output value will be picked first. Processes having same output value are ordered within themselves in accordance with the weights assigned to any of the three attributes. This leads to three invariants of this algorithm, that is, BJF(Arrival)/ BJF(Burst)/ BJF(Priority) according as the processes having same output values are ordered in the order of their time of arrival/ CPU burst/ priority respectively.

We present an example to explain the weighting technique and formation of the rule base. Let the arrival time of a process be categorized into four categories (Very Early, Early, Late, Very Late); size of CPU burst into three (Short, Medium, Long) and the priority of a process into five categories(Very High, High, Normal, Low, Very Low). The categories along with assigned weights are listed below.

Arrival	Weight	Bursts	Weight	Priority	Weight
Very Early	1	Short	1	Very High	1
Early	2	Medium	2	High	2
Late	3	Long	3	Normal	3
Very Late	4			Low	4
				Very Low	5

The rule base can be formed for all possible combinations of the three attributes in the following manner.

If the arrival is Early, CPU burst is Long and the priority is Low then the output value is 9.

There will be 60 (4×3×5) such rules in the rule base which are presented in Table 1.

It may be pointed out here that to ignore the contribution of any attribute, we must define the weight of that attribute to be zero for the entire range. For example- to reduce this policy to FCFS, the weights for all the categories of CPU burst and priority can be set to zero. Likewise this policy can also be reduced to SJF and Priority.

Table 1

Weights			OUTPUT VALUE	Weights			OUTPUT VALUE
Arrival	Burst	Priority		Arrival	Burst	Priority	
1	1	1	3	3	1	1	5
1	1	2	4	3	1	2	6
1	1	3	5	3	1	3	7
1	1	4	6	3	1	4	8
1	1	5	7	3	1	5	9
1	2	1	4	3	2	1	6
1	2	2	5	3	2	2	7
1	2	3	6	3	2	3	8
1	2	4	7	3	2	4	9
1	2	5	8	3	2	5	10
1	3	1	5	3	3	1	7
1	3	2	6	3	3	2	8
1	3	3	7	3	3	3	9
1	3	4	8	3	3	4	10
1	3	5	9	3	3	5	11
2	1	1	4	4	1	1	6
2	1	2	5	4	1	2	7
2	1	3	6	4	1	3	8
2	1	4	7	4	1	4	9
2	1	5	8	4	1	5	10
2	2	1	5	4	2	1	7

2	2	2	6	4	2	2	8
2	2	3	7	4	2	3	9
2	2	4	8	4	2	4	10
2	2	5	9	4	2	5	11
2	3	1	6	4	3	1	8
2	3	2	7	4	3	2	9
2	3	3	8	4	3	3	10
2	3	4	9	4	3	4	11
2	3	5	10	4	3	5	12

3. Algorithm and implementation of BJF method

A general sequence of Better Job First (BJF) algorithm is listed below:

Step 1: Define linguistic categories such as *low, medium, high etc.* to be used for the time of arrival, size of CPU burst and priority of the processes. Linguistic category of each attribute is attached with every admitted process.

Step 2: Assign weights to all linguistic categories for each attribute.

Step 3: Set-up a logical rule base on the basis of the number of categories defined for each attribute in Step 1. If *l, m, n* categories are defined for time of arrival, size of burst, priority respectively then the fuzzy rule base will have ($l \times m \times n$) rules.

Step 4: Obtain the output values for every process in accordance with the logical rule base.

Step 5: Order the processes in ascending order of output values. This serves as the order of execution on the processor for the processes. Go to Step 6, if some processes have equal output values.

Step 6: Processes having same output values shall have same order. To sort the processes having same output value within them, any of the following procedure may be used as per the suitability of requirement:

(a) Order them in the order of their time of arrival. This scheduling algorithm is called BJF (Arrival).

(b) Order them in the order of their size of CPU burst. This scheduling algorithm is called BJF (Burst).

(c) Order them in the order of their priorities. This scheduling algorithm is called BJF (Priority).

To understand the functioning of BJF algorithm, we observe its implementation through an example of 15 processes with their CPU estimates, Arrival time and priority as given below in Table 2:

Table 2

P-ID	CPU Burst	Arrival Time	Priority
1	1.48	0.19	2
2	5.16	1.12	5
3	9.71	1.84	2
4	2.28	3.28	3
5	7.11	3.45	4
6	1.10	4.45	5
7	0.41	5.64	1
8	3.74	6.49	3
9	2.07	7.62	2
10	1.78	9.18	4
11	2.15	10.02	4
12	0.81	11.65	3
13	4.24	12.50	2
14	9.84	13.17	4
15	8.78	14.45	1

Processes with their time of arrival, size of CPU bursts and assigned priorities are listed in Table 2. The output values and the execution orders of the processes under all variants of BJF are presented in Table 3.

The example of 15 processes given in Table 1 has been used to evaluate average waiting time of different policies including all variants of BJF. These results are listed in Table 4. It can be observed that except for SJF, results of all variants of BJF are better than FCFS and Priority scheduling.

Table 3

P- ID	Output Value	BJF (Arrival)	BJF (Burst)	BJF (Priority)
1	3	1	1	1
2	5	6	8	8
3	5	7	6	6
4	4	2	5	4
5	6	9	13	13
6	4	3	3	5
7	4	4	2	2
8	6	10	10	11
9	4	5	4	3
10	5	8	7	7
11	6	11	11	12
12	6	12	9	10
13	6	13	12	9
14	8	15	15	15
15	7	14	14	14

Table 4

Policy	Average Waiting Time
FCFS	26.04
SJF	15.22
Priority	26.74
BJF (Arrival)	21.00
BJF (Burst)	18.43
BJF (Priority)	20.434

4. Results and Discussions

Several randomly generated data sets have been tested in an attempt to compare performance of better job first policy with FCFS, SJF and RR in respect of average waiting time. Performances of all variants of BJF such as BJF (Arrival), BJF (Burst) and BJF (Priority) have also been compared.

Program codes for BJF have been developed in C++. Same data sets were used to evaluate the average waiting time for different scheduling policies, in order to compare the performance of the proposed scheduling policy against them.

Data set 1

No of processes: 50;

Following linguistic categories have been used for size of burst, time of arrival and priority in this data set:

Size of Burst Short burst : $0 \le x_i \le 5$;

Medium burst: $6 \le x_i \le 10$;

Long burst: $11 \le x_i \le 15$.

Time of Arrival Early arrival : $0 \le x_i \le 10$;

Intermediate arrival: $11 \le x_i \le 20$;

Late arrival: $21 \le x_i \le 30$.

Priority High priority : $1 \le x_i \le 5$;

Normal priority: $6 \le x_i \le 10$;

Low priority: $11 \le x_i \le 15$.

Values for average waiting time computed through different policies are presented below. It can be observed that the results of the proposed Better Job First policy is superior to FCFS, Priority and Round Robin policies and is comparatively closer to SJF. The burst variant of BJF produces a close approximation of SJF. We present in Figure 1 a bar chart of average waiting time results in order to facilitate a visual comparison of the performances of different scheduling policies on this data-set.

Policy	Average waiting time
FCFS	144.3994
SJF	97.00
PRIORITY	156.945
ROUND-ROBIN (Q=5)	174.90
BJF (Arrival)	123.8496
BJF (Burst)	112.8702
BJF (Priority)	129.6476

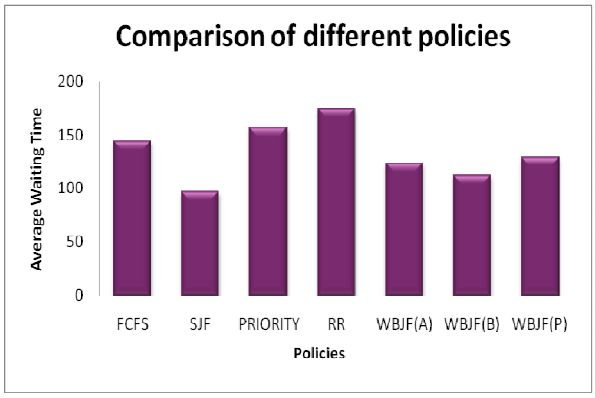

Figure 1

Data Set 2

No of processes: 50;

Following linguistic categories have been used for size of burst, time of arrival and priority in this data set:

Size of Burst	Short burst :	$0 \le x_i \le 10$;
	Medium burst:	$11 \le x_i \le 20$;
	Long burst:	$21 \le x_i \le 30$.
Time of Arrival	Early arrival :	$0 \le x_i \le 20$;
	Intermediate arrival:	$21 \le x_i \le 40$;
	Late arrival:	$41 \le x_i \le 60$.
Priority	High priority :	$1 \le x_i \le 3$;
	Normal priority:	$4 \le x_i \le 6$;
	Low priority:	$7 \le x_i \le 10$.

Values for average waiting time computed through different policies are presented below. It can be observed that the results of the proposed Better Job First policy are superior to FCFS, Priority and Round Robin policies and are comparatively closer to SJF.

Policy	**Average waiting time**
FCFS	391.1116
SJF	264.61
PRIORITY	355.6988
ROUND-ROBIN(Q=5)	512.40
BJF (Arrival)	325.2534
BJF (Burst)	297.8324
BJF (Priority)	318.420

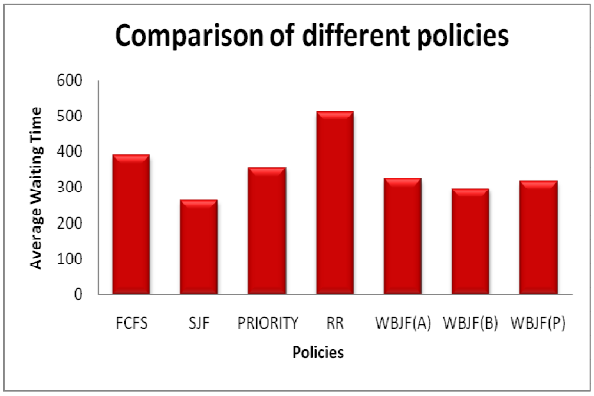

Figure 2

Data Set 3

No of processes: 50;

Following linguistic categories have been used for size of burst, time of arrival and priority in this data set:

Size of Burst	Short burst :	$0 \le x_i \le 3;$
	Medium burst:	$4 \le x_i \le 6;$
	Long burst:	$7 \le x_i \le 10.$
Time of Arrival	Early arrival :	$0 \le x_i \le 10;$
	Intermediate arrival:	$11 \le x_i \le 20;$
	Late arrival:	$21 \le x_i \le 30.$
Priority	High priority :	$1 \le x_i \le 5;$
	Normal priority:	$6 \le x_i \le 10;$
	Low priority:	$11 \le x_i \le 15.$

Policy	**Average waiting time**
FCFS	98.3754
SJF	54.9468
PRIORITY	89.9838
ROUND-ROBIN(Q=5)	108.06438
BJF (Arrival)	74.708
BJF (CPU)	65.0184
BJF(Priority)	75.058

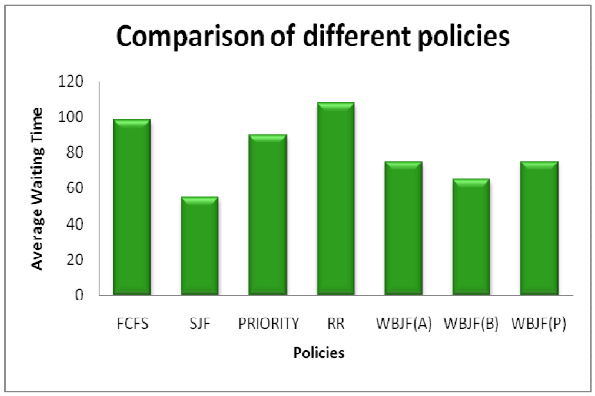

Figure 3

5. Conclusions

Present work introduces a new CPU scheduling algorithm. Better Job First (BJF) policy that has been proposed in this paper determines the order of a process in ready queue taking into consideration the integrated effect of time of arrival, size of CPU burst and priority. The proposed policy thus enjoys the advantages all the criteria. The philosophy of BJF policy gives rise to three invariants and each of its invariant BJF (Arrival), BJF (Burst) and BJF (Priority) performs better than FCFS, Priority and Round Robin scheduling on average waiting time. Proposed algorithm not only helps the scheduler in determining next most worthy job to be executed but can easily be reduced to FCFS, Priority or SJF with a simple adjustment in the weighting technique.

References

1. H. M. Deital, *Operating Systems*, Pearson, 2006.
2. William Stallings, *Operating Systems- Internals and Design Principles*, Pearson, 2006.
3. A. F. Al-Husainy Mohammed, Best-Job first CPU Scheduling Algorithm, *Information Technology Journal*, **6(2**) (2007) 288-293.
4. D. Pandey, Vandana and M. K. Sharma, CPU Scheduling: FCFS with Shorter Processes First, MR, *International Journal of Engineering and Technology*, **1 (2)** (2008)11-17.