OPTIMIZED SOLUTION TO SHORTEST JOB FIRST BY ELIMINATING THE STARVATION

Conference Paper · November 2005

CITATIONS
8

READS
2,494

2 authors:

Basit Shahzad
King Saud University
78 PUBLICATIONS **445** CITATIONS
SEE PROFILE

Muhammad Tanvir Afzal
Capital University of Science and Technology, Islamabad, Pakistan
83 PUBLICATIONS **230** CITATIONS
SEE PROFILE

OPTIMIZED SOLUTION TO SHORTEST JOB FIRST BY ELIMINATING THE STARVATION

BASIT SHAHZAD
COMSATS Institute of Information Technology
Ph: +92 (0) 300 5270145
E-mail: basit_shahzad@comsats.edu.pk

MUHAMMAD TANVIR AFZAL
COMSATS Institute of Information Technology
Ph: +92 (0) 300 6013823
E-mail: muhammad_tanvir@comsats.edu.pk

Abstract

Many algorithms are available for CPU scheduling including First Come First Serve (FCFS)[3], shortest job first (SJF)[3] and Round Robin (RR)[3]. It has been observed that average waiting time in shortest job first is the minimum and it can not be reduced further while average waiting time in the FCFS is the maximum. Along with minimum average waiting time, SJF introduces starvation for bigger processes, which is a strong shortcoming of this smart technique. A new algorithm is suggested which is called "Enhanced SJF" and gives average waiting time which is convergent to average waiting time of SJF and also ensures that bigger processes are not going to starve and each process is executed in some definite time. Along with surety that no job will starve, the average waiting time is also brought closer to average waiting time of SJF in most cases and equal to average waiting time of SJF in some cases.

1. Introduction to CPU Scheduling Algorithms

The comparison of "Enhanced Shortest job First" with FCFS and SJF is made. In SJF average waiting time is minimum and in FCFS average waiting time is maximum.

1.1 FCFS

In FCFS CPU can be monopolized by a single bigger process if bigger Processes are run first, this can cause a large increase in the waiting time for other smaller Processes.

Average waiting time in FCFS depends on the order of arrival of Processes FCFS is a relatively simple algorithm but it has its problems.
For example the Processes p1 (35), p2 (3) and p3 (3) arrive in the order P1, P2, and P3. P2 will have to wait 35ms and p3 for 38ms to execute. The Average waiting time is (35+38)/3 = 24.3ms. If the same processes arrives in the order P2, P3 and P1. Then average waiting time is (3+6)/3=3ms.

So, if bigger processes arrive at the beginning of the queue in FCFS then these processes can monopolize the CPU and average waiting time for the processes may increase. FCFS deprives for the longer average waiting time while SJF with smaller average waiting time introduce starvation.
Enhanced SJF does not allow the bigger processes to monopolize the CPU and gives chance to bigger processes to execute in certain time, without delaying for indefinite period.

We can consider an example for making a comparison between FCFS, SJF and Enhanced SJF. All the following processes arriving at time zero.

Processes	CPU Burst	Waiting Time
P1	15	0
P2	20	15
P3	3	35
P4	8	38
P5	9	46

By using FCFS let's calculate average waiting time for these Processes as shown in figure 1.

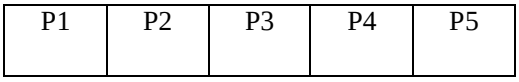


Figure No. 1: Execution of processes in FCFS.

The average waiting time for FCFS is (0+15+35+38+46) that is 134/5=26.8ms.

Same processes are run by using SJF in order to see the minimum average waiting time for the processes. Before executing the algorithm the shortest job first algorithm is seen in detail.

1.2 SJF

On the other hand Shortest Job First selects the process with shortest time of execution and executes the process.

When the CPU is available, the process with the shortest CPU burst is selected. If two or more processes have the same CPU burst time, then the FCFS algorithm is used to determine a tie.
SJF is provably optimal. By moving shorter processes to the front of the waiting queue, reduces the average waiting time up to maximum extent.

Major drawback of SJF is that shorter processes arrive in ready queue constantly and bigger processes continue growing towards tail of ready queues and their chance of getting CPU time even decreases.
SJF algorithm may cause a Process to block for indefinite time.

When the IBM 7094 at MIT was shut down in 1973, found that low priority Processes were found that were submitted in 1967 and were still pending and had not yet been run [1].Same the case may be with SJF if shorter processes arrive on regular basis and bigger processes will be waiting to get the chance and will not execute for a long time and may starve.

But Enhanced SJF algorithm ensures that bigger Processes will not starve and they will get the CPU after some specific time.

Let us take the same problem as discussed in FCFS and let's see the execution of SJF as shown in Figure No.2.

Processes	CPU Burst	Waiting Time
P1	15	20
P2	20	35
P3	3	0
P4	8	3
P5	9	11

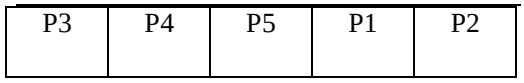


Figure No. 2: Execution of processes in SJF.

For SJF the average waiting time is (3+11+20+35)/5= 69/5=13.8ms which is the minimum average waiting time.

1.3 Enhanced SJF algorithm

In Enhanced SJF, Shorter processes are assigned the CPU as in SJF but this algorithm makes sure that after some interval bigger processes can get the CPU. The interval is selected dynamically.

In fact processes are sorted as soon as they arrive in the ready queue based upon their execution time. Shorter processes are arranged at the front of queue and bigger processes at the rear of the queue. Two pointers are used, one at the front of the queue and second at the rear of the queue, which is used to pick a process either from front or from rear of the queue.

Then a process is picked from front of queue that is shortest process and its estimated time is saved in a variable 'VarShorterJobs' and the estimated time of the biggest processes is saved in variable 'VarLargeJob' then next process is picked from front and its estimated time is added to the variable 'VarShorterJobs' and compared it with the variable 'VarLargeJob' when 'VarShortJobs' is equal to or greater than the 'VarLargeJob' then a process is picked from the rear and 'VarShorterJobs' is set to

zero and set the 'VarLargeJob' to the next bigger process in the queue. In this way after equal time of execution of shorter processes a bigger process will also get the CPU and again return the control to the shorter processes to be picked next for execution.

In this way bigger processes will not starve as may be in SJF and average waiting time for Enhanced SJF will also be approaching to average waiting time of the SJF in many cases. Some solutions might be added by introducing the time slices for each process. In that case additional computation is required for context switches. By giving CPU to bigger processes without introducing the time slice, gives strength to the Enhanced SJF algorithm for consideration.

The Enhanced SJF for the same set of Processes which was discussed in FCFS and in SJF.

Processes	**CPU Burst**	**Waiting Time**
P1	15	40
P2	20	20
P3	3	0
P4	8	3
P5	9	11

In Enhanced SJF these Processes will be inserted to the ready queue in the ascending order. After all processes are submitted they look like the figure 3.

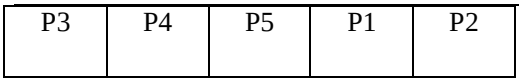


Figure No. 3: Arrangement of processes in ready queue for ESJF, P3 is on front and P2 is on rear.

Now there are two pointers, one pointing to front of the queue that is to the process P3 and second will be pointing to the process P2 at the rear of the queue. Value of 'VarShorterJobs' will be 3 and Process p3 gets the chance of execution. The value in variable 'VarLargejob' is 20. When process p3 has been executed then the value of 'VarShorterJobs' is compare with 'VarLargeJob' as VarShorterJobs value is not greater or equal to 'VarLargeJob' so next process from the front is chosen and now the value of 'VarShorterJobs' becomes 11. Again after execution of process P4. Process P5 is selected from the front of the queue and value of variable 'VarShorterJobs' becomes 20. after the execution of P5. When the values of variables are compared to pick the next process then value of 'VarShorterJob' becomes equal to value of the 'VarLargeJob'. Now the bigger process will be given the chance for execution. After the execution of P2, process P1 is selected for execution.

The difference between SJF and Enhanced SJF is that SJF will serve the process P1 then the process P2 because process P1 has shorter CPU Burst time than process P2, but Enhanced SJF execute Process P2 first because the sum of the estimated time for execution for shorter processes is greater than or equal to the estimated time for bigger process which is waiting. So, Enhanced SJF gives an equal chance of execution to bigger processes as well. Figure No.4 shows the details.

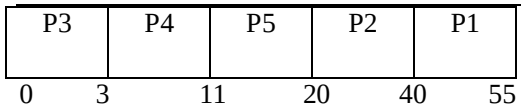


Figure No. 4: Execution of processes in ESJF.

Average waiting time for the Enhanced SJF is (20+40+3+11)/5 that is 74/5=14.8ms which is closer to minimum average waiting time. This is approximately equal to the Average waiting time calculated in SJF. Comparison for the average waiting time for the same problem in three Algorithms is given below.

FCFS	**SJF**	**Enhanced SJF**
25.6	13.8	14.8

By this example it is clear that average waiting time in Enhanced SJF is very close to SJF with the additional benefits that longer jobs now have lesser tendency to starve for CPU.

2. Detailed Discussion of Results

Enhanced SJF Algorithm is compared with FCFS and SJF for different set of processes. Suppose Processes arrive at the time zero in the order of P1, P2, P3, P4 and P5 and the waiting time for FCFS, SJF and ESJF(Enhanced Shortest Job First) are shown

a)

Processes	CPU Burst	FCFS	SJF	ESJF
P1	6	0	3	3
P2	8	6	16	9
P3	7	14	9	17
P4	3	21	0	0

Figure No. 5 result of first comparison of FCFS, SJF and ESJF

Average Waiting Time for FCFS=10.25ms
Average Waiting Time for SJF=7.0ms
Average Waiting Time for ESJF=7.2ms

b)

Processes	CPU Burst	FCFS	SJF	ESJF
P1	25	0	28	28
P2	8	25	4	4
P3	16	33	12	12
P4	4	49	0	0

Figure No. 6 result of second comparison of FCFS, SJF and ESJF

Average Waiting Time for FCFS=26.7ms
Average Waiting Time for SJF=11ms
Average Waiting Time for ESJF=11ms

c)

Processes	CPU Burst	FCFS	SJF	ESJF
P1	40	0	61	40
P2	8	40	0	0
P3	20	48	20	20
P4	12	68	8	8
P5	21	80	40	80

Figure No.7 result of Third comparison of FCFS, SJF and ESJF

Average Waiting Time for FCFS=47.2ms
Average Waiting Time for SJF=25.81ms
Average Waiting Time for ESJF=29.4ms

d)

Processes	CPU Burst	FCFS	SJF	ESJF
P1	30	0	37	37
P2	2	30	0	0
P3	20	32	17	17
P4	10	52	7	7
P5	5	62	2	2

Figure No. 8 result of fourth comparison of FCFS, SJF and ESJF

Average Waiting Time for FCFS=35.2ms
Average Waiting Time for SJF=12.0ms
Average Waiting Time for ESJF=12.0ms

e)

Processes	CPU Burst	FCFS	SJF	ESJF
P1	8	0	11	20
P2	5	8	0	0
P3	9	13	19	11
P4	6	22	5	5

Figure No. 9 result of fifth comparison of FCFS, SJF and ESJF

Average Waiting Time for FCFS=10.75
Average Waiting Time for SJF=8.75
Average Waiting Time for ESJF=9.0

f)

Processes	CPU Burst	FCFS	SJF	ESJF
P1	40	0	20	20
P2	100	40	275	205
P3	90	140	185	285
P4	45	230	60	105
P5	80	275	105	205
P6	20	355	0	0

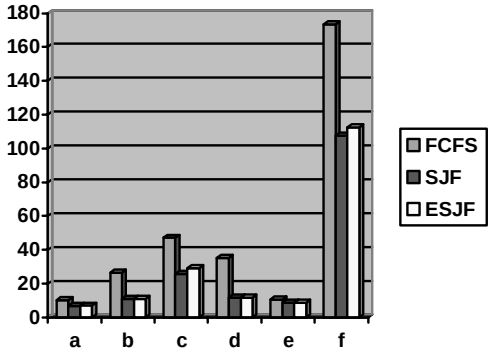
Figure No. 10 result of sixth comparison of FCFS, SJF and ESJF

Average Waiting Time for FCFS=173.3ms
Average Waiting Time for SJF=107.5ms
Average Waiting Time for ESJF=112.5ms

Comparisons of Enhanced SJF with FCFS and SJF in terms of average waiting time as shown in Figure No. 11.

Cases	FCFS	SJF	ESJF
Case #a	10.25	7.0	7.2
Case #b	26.7	11	11
Case #c	47.2	25.81	29.4
Case #d	35.2	12.0	12.0
Case #e	10.75	8.75	9.0
Case #f	17 3.3	107.5	112.5

Figure No. 11: Comparison of FCFS, SJF and ESJF in terms of average waiting time for different set of processes.



FigureNo.12: Graph showing the comparative results for FCFS, SJF and ESJF for different cases.

The Graph in Figure No.12 shows the comparison between FCFS, SJF and ESJF in terms of average waiting time. It is depicted from the graph that SJF and ESJF are very close in terms of average waiting time and the ESJF is giving CPU to the bigger jobs as well in the presence of shorter jobs in the ready queue.

3. Conclusions

This work focuses to provide an enhancement in existing algorithm. Shortest Job First algorithm gives minimum average waiting time but produce a serious problem i.e. starvation for the bigger processes. If shorter processes keep on coming on regular basis to the ready queue the bigger processes will remain in waiting state and the possibility that they do not get CPU for execution will increase.

Enhanced SJF eliminate the problem of starvation for bigger processes. Bigger processes will avail the chance to get the CPU on their turn even though the shorter processes are present in the Ready Queue. Main emphasis of this work is to ensure that both smaller processes and bigger processes get the CPU. Neither one process should monopolize the CPU as in FCFS nor should bigger processes starve. It has been ensured that the average waiting time of Enhanced SJF remains closer to the average waiting time of shortest job first and also that no job starve for CPU for longer time as does in shortest job first.

Many bigger processes are urgent to execute but submitted at the end of the queue. No conventional algorithm give them chance to complete their execution in a certain time, In SJF they will not get the CPU until all shorter processes have been executed and in FCFS as they are at the back of the queue they will not get the CPU before their turn. But Enhanced SJF will allocate the CPU to that process on its turn while ensuring that it is done in some definite amount of time.

As Enhanced SJF is non-preemptive, no context switch occurs and there is no need to save the current states of the processes which is seen as an overhead, in preemptive scheduling policies.

Enhanced SJF selects shorter processes first then give chance to bigger processes, so processes can not monopolize the CPU and CPU time is distributed among shorter processes and bigger processes equally.

Enhanced SJF eliminate starvation for bigger processes because it gives equal chance to bigger processes for execution. The dynamic policy of Enhanced SJF provides equal chance of execution to both smaller and bigger processes. The average wait time of Enhanced SJF is some time more than that of shortest job first. This can be overcomed by the benefit that the

Enhanced SJF will not let the bigger processes starve for longer time and every job submitted, does get the CPU with in some adequate amount of time.

References

[1] Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, "Operating System Concepts", John Wiley & Sons Inc, 6th ed, pp 160-pp 162

[2] Andrew S. Tanenbaum, "Modern Operating System", Prentice Hall, 2nd ed, Chapter No.2

[3] Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, "Operating System Concepts", John Wiley & Sons Inc, 6th ed, pp 160-200.

[4]William Stallings, "Operating System", Prentice Hall, 4th ed,

[5]http://williamstallings.com/Extras/OS-Notes/h6.html

[6]http://www.sci.brooklyn.cuny.edu/~philaris/cis25/cpusched.html