JAMM SCHEDULING: A MORE EFFICIENT APPROACH

Shahid Iqbal
Department of Computer Engineering
Faculty of Engineering & Technology
Jamia Millia Islamia Central University
New Delhi, India
shahidiqbal13@gmail.com

Abstract

Scheduling involves the allocation of resources and time to tasks in such a way that certain performance requirements are met. CPU scheduling is the mechanisms of selecting a process among various processes and allocate a processor in such a way so that other processes in a ready queue wait for minimum amount of time. This is usually done by short term scheduler. In a uniprocessor system where only one process may run at a time, any other process must wait until the CPU is free and rescheduled. This article discusses briefly about a more efficient way to scheduling which I termed as 'JAMM' scheduling. My experimental result has shown that it reduces the sufficient amount of average waiting time among processes.
Keywords: CPU scheduling, Ready-Queue, Short term scheduler, Average Waiting Time.

1. Introduction

In computer science, scheduling is the method by which threads, process or data flows are given access to system resources (e.g. processor time, communication bandwidth). But we will only talk about the CPU scheduling mechanism here in which we concern about the allocation of processor to the number of processes. CPU scheduling is the basis of multi-programmed operating systems. By switching the CPU among processes, the operating system can make the computer more productive. Scheduling is a fundamental operating system function, almost all computer resources are scheduled before use. Scheduling may be preemptive or non-preemptive approach. The previous research in this field helps us to understand the various factors.

2. Scheduler

A process migrates between the various scheduling queues throughout its lifetime. The operating system must select for scheduling purposes, processes from these queues in some fashion. The selection process is carried out by appropriate scheduler. It may be Long term scheduler or Short term scheduler. The primary distinction between these two schedulers is the frequency of their execution. The short term scheduler must select a new process for the CPU frequently. As our main focus would be on short term scheduler, in real situations the processes will arrive in ready queue in random fashion, which means we can't predict which process will have higher burst time and which one lower. Due to which this JAMM scheduling has been doing better.

2.1. JAMM Scheduling Algorithm

This scheduling is based on the fact that at time t=0 we have available suppose 'n' processes in the ready queue. This is also have assumed that each process has same priority i.e. (none process is higher or lower priority than the other) and we are executing on a uniprocessor system. The steps below are used to execute the algorithm.

- At time t=0 ready queue has 'n' processes (say P1, P2….Pn with their corresponding burst time as Pb1, Pb2…..Pbn).
- Allocate the processor to the very first process (P1 to processor).
- While P1 is executing sort the remaining processes in their ascending burst time order i.e. sort 'n-1' process.
- Keep allocating the processor remaining 'n-1' sorted processes until all the processes finished there task.

2.2. Mathematically

At time t=0 the following processes arrived in ready queue as shown in Fig.1.

Process	Burst time (in milli sec)
P1	3
P2	6
P3	7
P4	4
P5	2

Fig.1: Process arrived in this order

Process	Burst time (in milli sec)
P1	3
P5	2
P4	4
P2	6
P3	7

Fig.2: Sorted processes for JAMM scheduling

The Gantt chart for the JAMM Scheduling is as follows

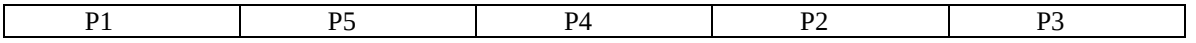

Fig.3: Gantt. Chart for JAMM scheduling

Waiting time of each process

.

P1=0 ms, P2=9 ms, P3=15 ms, P4=5 ms, P5=3 ms

Average Waiting Time (in JAMM case) = (0+9+15+5+3)/5

=6.4 ms

Where ms=milli second

2.3. Comparison of JAMM Scheduling with First Come First Serve (FCFS) Scheduling Algorithm

Now if we take same data from Fig.1 the Gantt. Chart for the FCFS scheduling will be as follows

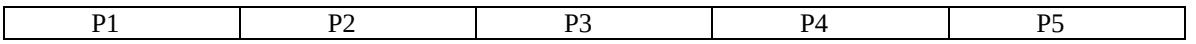

Fig.4: Gantt. Chart for FCFS scheduling

Waiting time of each process

P1=0 ms, P2=3 ms, P3=9 ms, P4=16 ms, P5=20 ms

Average Waiting Time (in FCFS case) = (0+3+9+16+20)/5

=9.6 ms

Average Waiting Time Reduced by (in percentage) = {((9.6-6.4)/9.6)*100}= 33.33 %

2.4. Characteristics of JAMM Scheduling

- This scheduling is better than the FCFS scheduling and this become much better when the number of processes arrived are more.
- Any sorting algorithm can be applied in this scheduling because the processes available at the short term scheduler level will be very few, so it doesn't matter much whether the sorting algorithm runs of order O(n^2) or O(nlogn).
- In worst case scenario when the processes available at short term scheduler level in already sorted form i.e (Pb1<=Pb2……. <=Pbn , where Pbn is the burst time of a nth process).In this scenario JAMM scheduling will behave exactly as FCFS scheduling. But this situation rarely happens.
- Switching the CPU to another process requires saving the state of the old process and loading the saved state for the new process, this task is known as Context switching. Context switch time is pure overhead, because the system does no useful work while switching. As JAMM scheduling is non-preemptive so there is no context switching in this scheduling.

3. Experimental Results

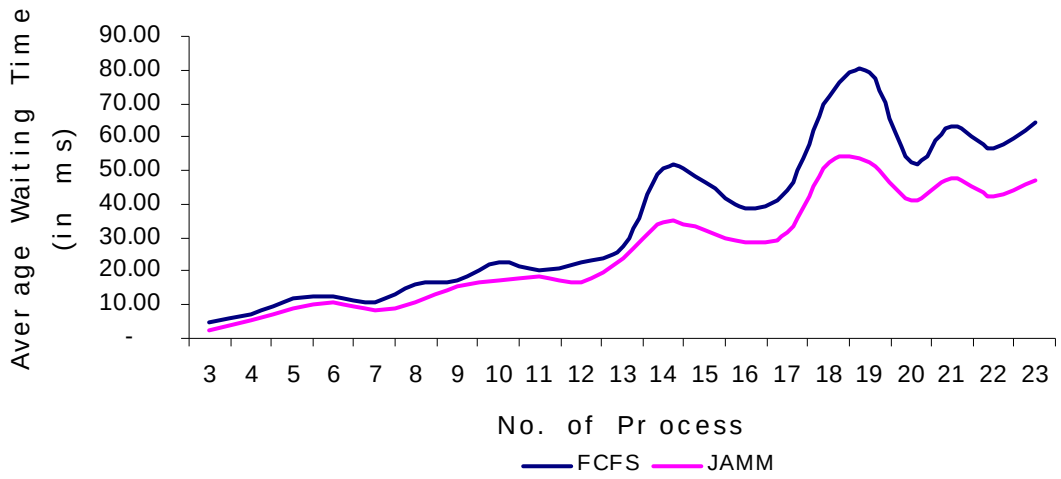

Fig .5: Comparison of JAMM with FCFS Scheduling Chart

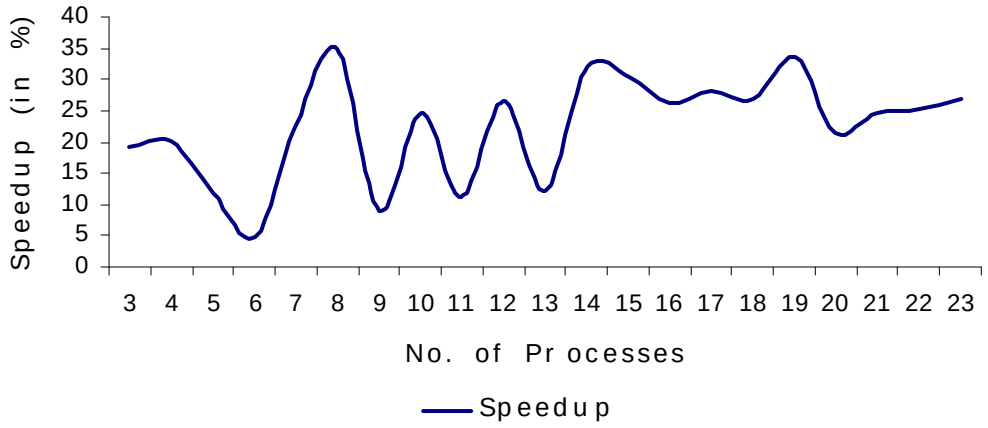

Fig.6: Speedup Chart

As shown in Fig.5 the chart shows the comparison between the JAMM versus FCFS Scheduling. This experiment implemented using C programming language. My experiment based on the fact that in real situations, processes will be available at random fashion, that means each time if 'n' processes gets available with there almost different burst time of each others and if there burst time varies between 1 to 20 ms then it can be seen that sufficient amount of average waiting time has been reduced and it becomes much more beneficial when number of processes available in large quantity. It has also been shown that less than 3% times JAMM Scheduling will behave exactly as FCFS Scheduling, even for very few processes, this rarely happens. In Fig.6 the speedup or we can say average reduction in waiting time(in %) chart which clearly shows that we can achieve average speedup ranging between 15 to 40 %. by using JAMM Scheduling as compared to FCFS.

4. Limitations

As this scheduling is non-preemptive in nature so context switching will not be done here therefore it concludes that this scheduling is not appropriate for the time sharing environment systems.

5. Implementation

For any scheduling algorithm it must be as much as simple to implement. This JAMM Scheduling can easily be implemented at the short term scheduler level by numerous programming languages so that efficient system design could be achieved. But programming languages which are more suitable for system programming would be efficient so 'C' language would be preferred in this.

6. Conclusion

When the average waiting time of a processes waiting to allocate a processor gets reduced in the ready queue, the turn around time will also get reduced by the same factor. Due to this the throughput of the system will also be increased. Hence the overall system will become more productive in nature. This JAMM Scheduling will be more suitable for heavily loaded system where more and more processes await to execute there task.

References

[1] Coffman E G.(1976)"Computer and Job/Shop Scheduling Theory"Wiley & Sons.
[2] Cormen T.H,Leiserson C.E,Rivest L and Stein C.(2009)."Introduction to Algorithms "2nd edn PHI.
[3] Dhamdhere D M.(1999)."Systems Programming and Operating Systems". 2nd edn,Tata McGraw Hill
[4] Li T.,Baumberger D.,Kaufaty D A.and Hahn S."Efiicient Operating System Scheduliing for Performance Assymmetric Multi- Core Architecture".Intel Corporation.
[5] Silberschatz A., Galvin P B and Gagne G.(2003)."Operating System Concepts".6th edn,John Wiley &Sons.
[6] Wierman A.(2007)"Scheduling for Today's Computer System:Bridging Theory and Practice",PhD Thesis ,Carnegie Mellon University.

Author

Shahid Iqbal received B. Tech in Computer Engineering from Jamia Millia Islamia University and pursuing M.E. from BITS, Pilani. My areas of interest are Operating System, Cryptography, Compilers, and Algorithm Design.