ISSN : 2229-4333(Print) | ISSN : 0976-8491(Online)

A Comparison between FCFS and Mixed Scheduling

Alka Pant
Dept. of Computer Science, College of Professional Education, Meerut, UP, India

Abstract
When more than one process is runable, the operating system must decide which one first. Scheduling refers to a set of policies and mechanisms built into the operating system that govern the order in which work to be done by a computer is completed. Many objectives must be considered in the design of a scheduling discipline. In particular, a scheduler should consider fairness, efficiency, response time, turnaround time, throughput, etc., Some of these goals depends on the system one is using for example batch system, interactive system or real-time system, etc. but there are also some goals that are desirable in all systems. A scheduling discipline is preemptive if, once a process has been given the CPU can taken away. Among the non preemptive processes the most commonly used scheduling policies are FCFS and SJF. A method which is discussed here shows much better performance than FCFS which works with the principle of mixing jobs. According to this method, from a list of N jobs, the job that needs minimum CPU time is executed first and then the highest form the list and so on till the Nth job. Through this method we can easily compare b/w FCFS & MIX job scheduling.

Keywords
FCFS, SJF, Scheduler, Waiting Time, Turnaround Time.

I. Introduction

"Scheduling is an activity that will be done by the operating system component called the Scheduler. The purpose of the scheduler is to choose processes from the list of ready processes". Scheduling is a key concept in computer multitasking, multiprocessing operating system and real-time operating system designs. Scheduling refers to the way processes are assigned to run on the available CPUs, since there are typically many more processes running than there are available CPUs. With time sharing system the scheduling algorithm becomes more complex because there are generally multiple users waiting for service. Some mainframes still combine batch and timesharing service, requiring the scheduler to decide whether a batch job or an interactive user at a terminal should go next. The scheduling algorithm differs from one another only on the choice of the processes given preferential treatment.

The performance improvement (example: Reduction of the response time) of one category of processes can be achieved only at the expense of performance degradation of processes in another category. There is a no single best scheduling algorithm that is good for all processes. Each operating system employs a different flavor of scheduling algorithm. The CPU scheduler occupies the tiny portion of the operating system but execute very frequently, especially in time sharing systems. Therefore the algorithm should reach a decision quickly, for which it must be simple and efficient. With the introduction of personal computers the situation changed in two ways. First most of the time there is only one active process. A user typing a document on a word processor is not likely to be simultaneously compiling a program in the background. when the user types a command to the word processor the scheduler does not have to do much work to choose which process to run as the only processes is running on the word processor.

II. Need And Purpose For Scheduling

The main purpose of scheduling algorithms is to minimise resource starvation and to ensure fairness amongst the parties utilizing the resources. Scheduling deals with the problem of deciding which of the outstanding requests is to be allocated resources. Scheduling disciplines are algorithms used for distributing resources among parties which simultaneously and asynchronously request them. Scheduling disciplines are used in routers (to handle packet traffic) as well as in operating systems (to share CPU time among both threads and processes), disk drives (I/O scheduling), printers (print spooler), most embedded systems etc.

A. CPU Utilization

We want to keep the CPU as busy as possible. The load on the system affects the level of utilization that can be achieved; high utilization is more easily achieved on more heavily loaded systems. The importance of this criterion typically varies depending on the degree the system is shared. On a single user system, CPU utilization is relatively unimportant.

B. Balanced Utilization

The percentage of time all resources are utilized. Instead of just evaluating CPU utilization, utilization of memory, I/O devices, and other system resources are also considered.

C. Throughput

If the CPU is busy executing processes, then work is being done. One measure of work is the number of processes that are completed per time unit, called throughput. For long processes, this rate may be one process per hour; for short transactions, it may be 10 processes per second.

D. Turnaround time

From the point of view of a particular process, the important criterion is how long it takes to execute that process. The interval from the time of submission of a process to the time of completion is the turnaround time. Turnaround time is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O.

E. Waiting time

The CPU scheduling algorithm does not affect the amount of the time during which a process executes or does I/O; it affects only the amount of time that a process spends waiting in the ready queue. Waiting time is the sum of periods spends waiting in the ready queue.

F. Response time

In an interactive system, turnaround time may not be the best criterion. Often, a process can produce some output fairly early and can continue computing new results while previous results are being output to the user. Thus, another measure is the time from the submission of a request until the first response is produced. This measure, called response time, is the time it takes to start responding, not the time it takes to output the response. The turnaround time is generally limited by the speed of the output device.

G. Predictability

Lack of variability in other measures of performance. Users prefer consistency. For example, an interactive system that routinely responds within a second, but on occasion takes 10s to respond, may be viewed more negatively than a system that consistently responds in 2s.

H. Fairness

Fairness is important under all circumstances. A scheduler makes sure that each process gets its fair share of the CPU and no process can suffer indefinite postponement. Note that giving equivalent or equal time is not fair. Think of safety control and payroll at a nuclear plant.

A key issue related to scheduling is when to make scheduling decisions. It turns out that there are a variety of situations in which scheduling is needed. First when a new process is created, a decision needs to be made whether to run the parent process or the child process. Since both processes are in ready state, it is a normal scheduling decision and it can go either way that is the scheduler can legitimately choose to run either the parent or the child next. An accurate illustration should involve many processes, each being a sequence of several hundred CPU bursts and I/O bursts. For simplicity of illustration, we consider only one CPU burst per process in our examples. Our measure of comparison is the average waiting time.

III. Non Preemptive and Preempive Scheduling

A scheduling discipline is nonpreemptive, if once a process has been given the CPU, the CPU cannot be taken away from that process.

Following are some characteristics of nonpreemptive scheduling

a) In nonpreemptive system, short jobs are made to wait by longer jobs but the overall treatment of all processes is fair.
b) In nonpreemptive system, response times are more predictable because incoming high priority jobs can not displace waiting jobs.
c) In nonpreemptive scheduling, a schedular executes jobs in the following two situations.

- When a process switches from running state to the waiting state.
- When a process terminates.

Following are some characteristics of preemptive scheduling-

a) Yank the CPU away from the currently executing process when a higher priority process is ready.
b) Can be applied to both Shortest Job First or to Priority scheduling.
c) Avoids "hogging" of the CPU
d) On time sharing machines, this type of scheme is required because the CPU must be protected from a run-away low priority process.
e) Give short jobs a higher priority – perceived response time is thus better.
f) What are average queueing and residence times? Compare with FCFS.

IV. Categories Of Scheduling Algorithms

In different environments different scheduling algorithms are needed. This situation arises because different application areas have different goals. There different environments are -

1. Batch
2. Interactive

Goals for batch and interactive systems

1. Provide fairness
2. Everyone makes some progress; no one starves
3. Maximize CPU utilization
   - Not including idle process
4. Maximize throughput
   - Operations/second (min overhead, max resource utilization)
5. Minimize turnaround time:
   - Batch jobs: time to execute (from submission to completion)
6. Shorten response time
   - Interactive jobs: time response (e.g. typing on a keyboard)
7. Proportionality
   - Meets user's expectations

In Batch System there are no users impatiently waiting at their terminal for a quick response. Consequently non preemptive algorithms or preemptive algorithms with long time periods reduces process are often acceptable. This approach reduces process switches and thus improves performance.

In Interactive systems, preemptive is essential to keep one process from hogging the CPU and denying service to the others.

V. Scheduling of Batch Systems

The two most commonly used algorithms for batch processing system are FCFS (First Come First Served) and SJF (Shortest Job First).

A. FCFS

First Come, First Served (FCFS), is the simplest scheduling algorithm, FIFO simply queues processes in the order that they arrive in the ready queue. Since context switches only occur upon process termination, and no reorganization of the process queue is required, scheduling overhead is minimal The FCFS scheduling is fair in the formal sense or human sense of fairness but it is unfair in the sense that long jobs make short jobs wait and unimportant jobs make important jobs wait.

FCFS is more predictable than most of other schemes since it offers time. FCFS scheme is not useful in scheduling interactive users because it cannot guarantee good response time. The code for FCFS scheduling is simple to write and understand.

1. Advantage

It is easy to understand and easy to program. It is also fair. A single linked list keeps track of all ready processes. Picking a process to run just requires removing one from the front of the queue. Adding a new job or unblocked process just requires attaching it to the end of the queue.

2. Disadvantage

- Throughput can be low, since long processes can hog the CPU
- Turnaround time, waiting time and response time can be low for the same reasons above.
- Processing time of each job must be known in advance. Suitable only for batch process.
- One of the major drawback of this scheme is that the average time is often quite long.

B. SJF

Shortest Job First (SJF), with this strategy the scheduler arranges processes with the least estimated processing time remaining to be next in the queue. This requires advance knowledge or estimations about the time required for a process to complete.

- If a shorter process arrives during another process' execution, the currently running process may be interrupted, dividing that process into two separate computing blocks. This creates excess overhead through additional context switching. The scheduler must also place each incoming process into a specific place in the queue, creating additional overhead.
- This algorithm is designed for maximum throughput in most scenarios.
- Waiting time and response time increase as the process' computational requirements increase. Since turnaround time is based on waiting time plus processing time, longer processes are significantly affected by this. Overall waiting time is smaller than FCFS, however since no process has to wait for the termination of the longest process.
- No particular attention is given to deadlines, the programmer can only attempt to make processes with deadlines as short as possible.
- Starvation is possible, especially in a busy system with many small processes being run.

1. Advantage

Best scheduling algorithm for shortest jobs. Waiting time and turn around time are less compared to FCFS.

2. Disadvantage

Long jobs may wait longer than in FCFS because it has to wait not only for jobs that are in the system at the time of its arrival, but also for all short jobs that are in the system at the time of its arrival, but also for all shorter jobs that arrive subsequently while it is waiting for service.

C. JOB MIX (JM)

n this method from the queue of jobs another queue is maintained which contain the shortest job first then the highest job, again shortest of the remaining and then highest of the remaining. First job in the queue is executed first and then the second and if suspended conditions arise they are blocked till the condition is satisfied and send back to the queue. The process will be executed again as its turn comes in the queue. Using this method we can eliminate the starvation of longer jobs compared to shortest job method and it provides better turn around and waiting time compared to the FCFS method.

1. Advantage

Better turn around and wait time compared to FCFS scheduling. Method is simple and easy to implement. Satisfies various criteria that constitutes a good scheduling algorithm such as -

Fairness: make sure that each process gets its fair share of the CPU.

Efficiency: Keep the CPU busy 100 of the time.

Turnaround time: minimize the time batch users must wait for output.

Wait time: minimize the wait time of the process compared to FCFS.

2. Disadvantage

The over head of maintaining a second queue is there which when compared to FCFS is little complex.

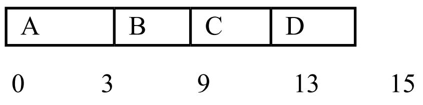
PROBLEM 1 (JOB ARRIVING AT THE SAME TIME)

FCFS

Job No	Process	CPU Time	Turnaround Time	Waiting Time	Waiting Turnaround Time
1	A	3.00	3.00	0.00	1.00
2	B	6.00	9.00	3.00	1.50
3	C	4.00	13.00	9.00	3.25
4	D	2.00	15.00	13.00	7.50

Average Turnaround Time = 10.00
Average Waiting Time = 6.25
Average Weighted Turnaround Time = 3.31

Gantt Chart

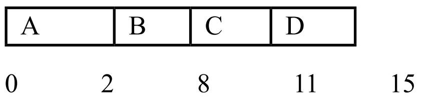


Job Mixing

Job No	Process	CPU Time	Job Order-ing	Turn-around Time	Waiting Time	Waiting Tur around Time
1	A	3.00	2.00	2.00	0.00	1.00
2	B	6.00	6.00	8.00	2.00	1.33
3	C	4.00	3.00	11.00	8.00	3.66
4	D	2.00	4.00	15.00	11.00	3.75

Average Turnaround Time = 9.00
Average Waiting Time = 5.25
Average Weighted Turnaround Time = 2.43

Gantt Chart



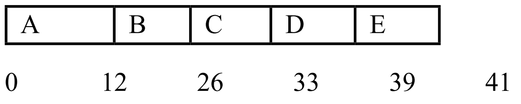
PROBLEM 2

FCFS

Job No	Process	CPU Time	Turnaround Time	Waiting Time	Waiting Turnaround Time
1	A	12.00	12.00	0.00	1.00
2	B	14.00	26.00	12.00	1.85
3	C	7.00	33.00	26.00	4.71
4	D	6.00	39.00	33.00	6.50
5	E	2.00	41.00	39.00	20.5

Average Turnaround Time = 30.20
Average Waiting Time = 22.00
Average Weighted Turnaround Time = 6.91

Gantt Chart



Job Mixing

Job No	Process	CPU Time	Job ordering	Turn-around Time	Waiting Time	Waiting Turn-around Time
1	A	12.00	2.00	2.00	0.00	1.00
2	B	14.00	14.00	16.00	2.00	1.14
3	C	7.00	6.00	22.00	16.00	3.66
4	D	6.00	12.00	34.00	22.00	2.83
5	E	2.00	7.00	41.00	34.00	5.85

Average Turnaround Time = 23.00
Average Waiting Time = 14.8
Average Weighted Turnaround Time = 2.89

Gantt Chart

A	B	C	D	E	
0	2	16	22	34	41

VI. CONCLUSION

Mix scheduling is better than FCFS by all means. We can solve the problems easily using the mix scheduling. It provides better turnaround time and waiting time for jobs whose execution time is known in advance.

VII. Acknowledgement

I am heartily thankful to Dr. (Mrs) Shilpi Gupta, Associate Professor, Amrapali Institute, Haldwani whose encouragement, guidance and support from the initial to the final level enabled me to develop an understanding of the subject.
Lastly, I offer my regards and blessings to all of those who supported me in any respect during the completion of my research.

References

[1] Abraham Silberschatz, Peter B. Galvin, Greg Gagne, “Operating System Concepts”, John Wiley & Sons, United States, 2005.
[2] Andrew S. Tanenbaum, Albert S. Woodhull, “Modern Operating Systems Second Edition”, Prentice Hall of India, New Delhi, 2001.
[3] Andrew S. Tanenbaum, Albert S. Woodhull, “Operating Systems Design and Implementation”, Prentice Hall of India, New Delhi, 2010.
[4] Charles Crowley, “Operaing Systems”, Tata McGraw-Hill Publishing Company Ltd, New Delhi, 1997.
[5] D.M. Dhamdhere, “Systems Programming And Operating Systems”, Tata McGraw-Hill Publishing Company Ltd, New Delhi, 1996.
[6] Ida M. Flynn, Ann McIver McHoes, “Understanding Operating Systems”‟, Thomson Press, Mumbai, 2005.
[7] Sibsankar Haldar, Alex A. Aravind, “Operating Systems‟, Pearson Education India, Delhi, 2010.
[8] Silberschatz, Galvin, “Operating System Concepts Seventh Edition”, John Wiley & Sons.Inc, United States, 2006.
[9] [Online] Available : www. Articles.Assyriancafe.Com/Documents/CPU_Scheduling.Pdf
[10] [Online] Available : www. emerald.com

Alka Pant received his M.C.A degree from U.P. Technical University, Lucknow in 2006, the Certification from Sun Micro-systems in Advanced Java in 2008 and the M.Phil degree in Computer Science from Vinayaka University, Tamil Nadu in 2008. His research interests include networking, algorithm, and computer graphics. She has around 5 years of teaching experience and authored 3 books. At present, she is engaged in teaching as an Assistant Professor in Computer Science Department of Swami Darshnanand Institute of Management and Technology (SDIMT), Haridwar, Uttarakhand. She has also as an Associate Editorship in the ‘Advanced Journal of Computer Science’ of Society of Educational and Applied Research and Technology, publication (SEART) Meerut.