RESEARCH ARTICLE OPEN ACCESS

A Comparative Study of Disk Scheduling Algorithms

Tarsem Singh
Post Graduate
Department of Computer Science and Applications
GHG Khalsa College, Gurusar Sadhar
Ludhiana - India

ABSTRACT
In the recent years, the speed of processor and main memory has increased more rapidly than those of secondary storage devices, such as hard disk. As a result, processes requesting data on secondary storage tend to experience relatively long service delays. In operating system, seek time is most important to get best access time. So, scheduling of disk tracks is one of the main responsibilities of the operating system. In this paper, various basic disk scheduling techniques like FCFS, SSTF, SCAN, LOOK, C-SCAN and C-LOOK along with some additional SCAN techniques like FSCAN, N-Step SCAN etc. are discussed.
Keywords:- FCFS, SSTF, SCAN, LOOK, C-SCAN, C-LOOK, FSCAN, N-Step SCAN

I. INTRODUCTION

The management of disk performance is an important aspect of an operating system. Since the speed of processor and main memory have been increased several times than the speed of the disk, the difference in the speed of processor and the disk, I/O performance of disk has become an important bottleneck. The performance of disk storage subsystem is of vital concern, and much research has gone into schemes for improving that performance. In any disk system with a moving read/write head, the seek time between cylinders takes a significant amount of time. This seek time should be minimized to get better access time. The main responsibility of the operating system is to use the hardware efficiently.

II. DISK PERFORMANCE PARAMETERS

Seek Time: The seek time is the time for the disk arm to move the heads to the cylinder containing the desired track.

Rotational Latency: The Rotational Latency is the additional time for the disk to rotate the desired sector to the disk head.

Access Time: Access time is the sum of the seek time and the rotational latency.

Transfer Time: The transfer time to or from the disk depends on the rotation speed of the disk.

$$T = {}^{b}/_{rN}$$

Where T = Transfer Time
b = Number of bytes to be transferred
N = Number of bytes on a track
r = Rotation speed, in revolutions per second

Thus the total average access time can be expressed as:

$$T_a = T_s + {}^{1}/_{2r} + {}^{b}/_{rN}$$

where T_s is the average seek time.

Mean Response Time: The average time spent waiting for a request to be serviced.

Disk Bandwidth: The disk bandwidth is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer.

Throughput: The number of requests serviced per unit time.

A scheduling policy should attempt to maximize throughput and minimize the mean response time. The smaller the variance, the more likely it is that most disk requests are serviced after waiting for a similar amount of time. Therefore, variance can be seen as a measure of fairness and of predictability. We desire a scheduling policy that minimizes variance to avoid erratic service times. Disk scheduling algorithms are used to choose one of the disk requests available to execute.

III. DISK SCHEDULING ALGORITHMS

First Come First Serve (FCFS) Scheduling:

It is the simplest form of disk scheduling algorithm. This policy uses a FIFO queue so that requests are serviced in

the order in which they arrive. In this algorithm the I/O requests are processed from the queue in sequential order. This algorithm is intrinsically fair because every request is honoured. The disk assesses are in the same order as the requests were originally received. The first request is accessed and processed first and other requests are processed with their order of arrival. This algorithm is generally does not provide the fastest service.

Example: Consider a disk with 200 tracks (0-199) and that the disk request queue has random requests in it. The following track requests for I/O to blocks on cylinders:

60, 143, 15, 185, 85, 120, 33, 28, 146

Consider that the read/write head is initially at cylinder 70. Compute the total head movements for a 200 track disk (0-199) and the total seek time needed to traverse all the requests if the seek rate of 5 milliseconds is given.

Solution:

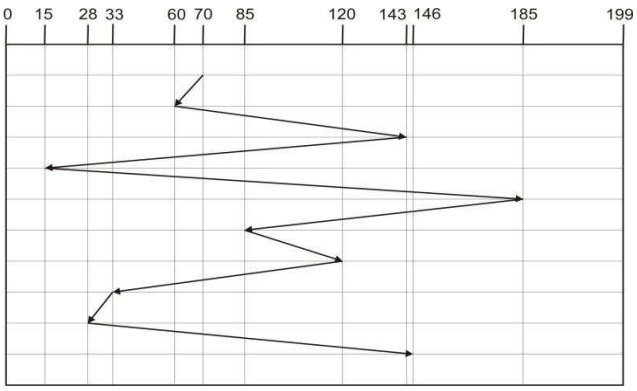


Fig. 1 FCFS Scheduling

Total Head Movements = (70-60) + (143-60) + (143-15) + (185-15) + (185-85) + (120-85) + (120-33) + (33-28) + (146-28) = 736 cylinders
Total Seek Time = 736 x 5 = 3680ms

Shortest Seek Time First (SSTF) Scheduling:

This algorithm selects the disk I/O request that requires the minimum seek time from the current head position. For the same example as used in FCFS, in SSTF, the first track accessed is 60, because this is the closest requested track to the starting position. The next track accessed is 85 because this is the closest of the remaining requested tracks to the current position of 60. Since seek time increases with the number of cylinders traversed by the head, SSTF chooses the pending request closest to the current head position

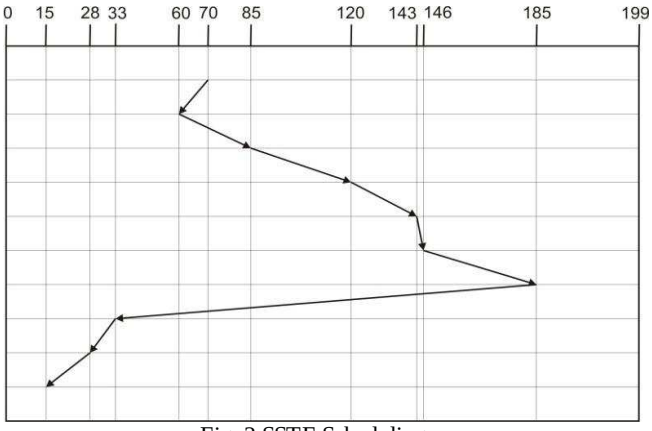


Fig. 2 SSTF Scheduling

Total Head Movements = (70-60) + (85-60) + (120-85) + (143-120) + (146-143) + (185-146) + (185-33) + (33-28) + (28-15) = 305 cylinders
Total Seek Time = 305 x 5 = 1525 ms

Since SSTF Scheduling has very less total head movements from FCFS, it is not optimal. In the above example, if head moves from 70 to 85, even though the cylinder is not closest, and then to 60, 33, 28, 15, 120, 143, 146, 185. This technique reduces the total head movement to 255 cylinders.

SCAN Scheduling:

In SCAN scheduling algorithm, the disk arm required to move in one direction only, servicing requests until it reaches the last track in that direction. At the other end, the direction of head movement is reversed, and services the pending requests. This algorithm is also known as *Elevator Algorithm* since it works like an elevator. It first processes all the requests in one direction and after that all the remaining processes are processed in reverse direction.

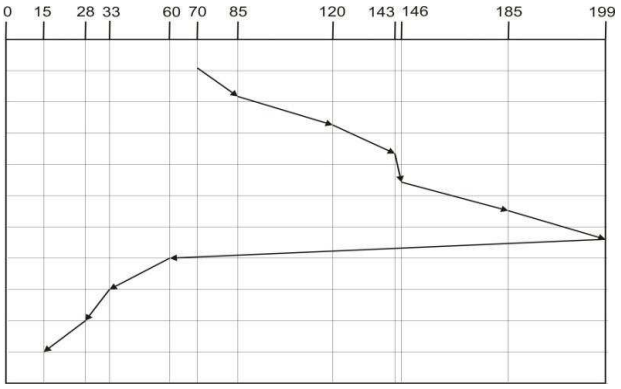


Fig. 3 SCAN Scheduling

Total Head Movements = (199-70) + (199-15)= 313 cylinders.
Total Seek Time = 313x5 = 1565ms.

***C-SCAN Scheduling*:**

C-SCAN (Circular SCAN) Scheduling policy is a variant of SCAN scheduling which restricts scanning to one direction only. In this scheduling when I/O head moves from one end to the other, servicing requests as it goes. When it reaches the other end, it immediately returns to the other end of the disk without servicing any request and scan remaining processes after moving other direction. It further reduce variance of response times as the expense of throughput and mean response time.

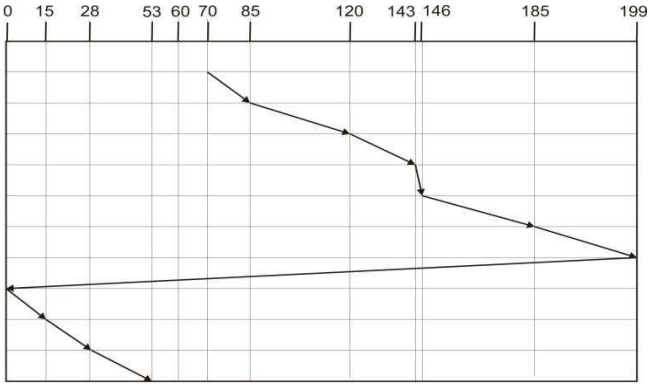


Fig. 4 C-SCAN Scheduling

Total Head Movements = (199-70) + (199-0) + (60-0)
= 388 cylinders
Total Seek Time = 388 x 5 = 1940ms

LOOK Scheduling:

This algorithm is similar to SCAN algorithm except the sweep of head movement restricts to the first and last serviced track instead of end to end sweep.

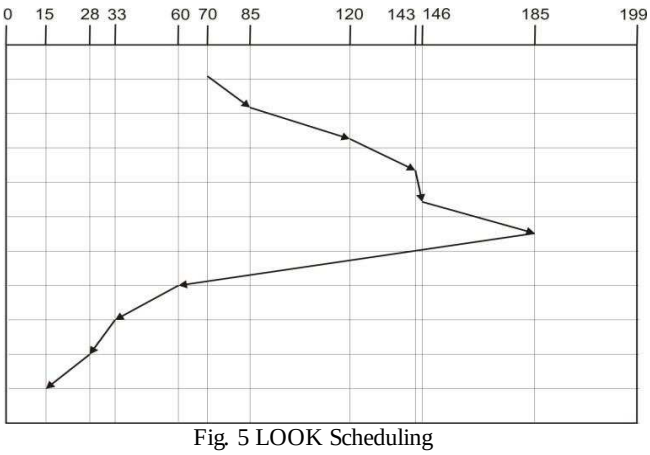


Fig. 5 LOOK Scheduling

Total Head Movements = (185-70) + (185-15)= 285 cylinders
Total Seek Time = 285 x 5 = 1425 ms

C-LOOK Scheduling:

This algorithm is a modified version LOOK Scheduling. Here the disk arm only travels as far as the last request in each direction and reverses direction immediately without going to the end of the disk.

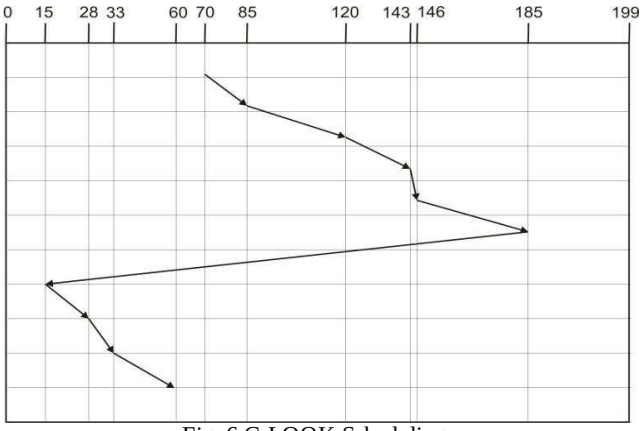


Fig. 6 C-LOOK Scheduling

Total Head Movements=(185-70)+(185-15)+(60-15)
=330 cylinders
Total Seek Time = 330 x 5 = 1650ms

Modifications to the SCAN Scheduling:

FSCAN Scheduling: In FSCAN Scheduling is modification of SCAN scheduling which eliminate the possibility of indefinitely postponing requests. This strategy uses two sub-queues. When a scan begins, all of the requests are in one of the queues, with the other empty. During the scan, all new requests are put into the other queue. Thus, service of new requests is deferred until all of the old requests have been processed. FSCAN uses the SCAN strategy to service only those requests waiting when a particular sweep begins (the "F" stands for "freezing" the request queue at a certain time).

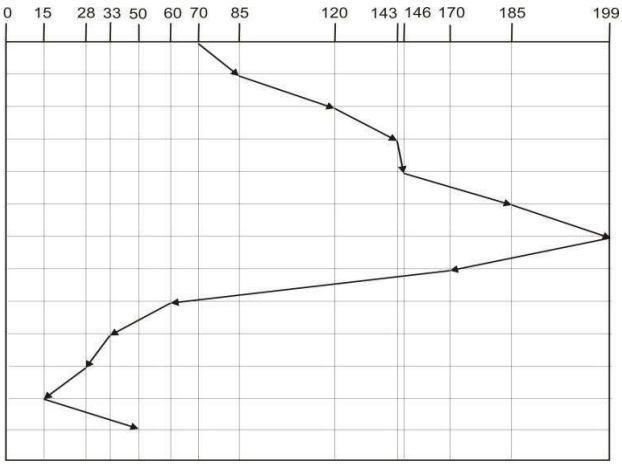
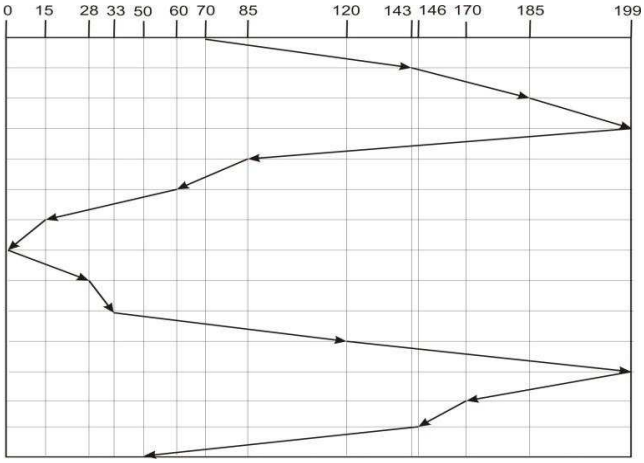


Fig. 7 FSCAN Scheduling

Suppose request 170 arrives after 146 is processed and request 50 arrives after 60 is processed, the seek pattern under FSCAN strategy is shown in figure .

N-Step SCAN Scheduling: N-Step SCAN scheduling is another modification of SCAN strategy which also prevents starvation. It segments the disk request queue into sub-queues of length *n*. These sub-queues are

processed one at a time using SCAN. When the sweep is complete, the next *n* requests are serviced. New requests are placed at the end of the request queue. N-Step SCAN can be tuned by varying the value for *n*. When *n*=1, N-Step SCAN degenerates to FCFS. As *n* approaches infinite, N-Step degenerates SCAN.



This policy also offers good performance due to high throughput and low mean response times, because they prevent indefinite postponement.

IV. COMPARISON OF VARIOUS DISK SCHEDULING ALGORITHMS

FCFS Scheduling:

It is a fair scheduling policy. It prevents starvation and gives low overhead. This is acceptable when the load on a disk is light. It has extremely low throughput due to lengthy seeks.

SSTF Scheduling:

SSTF Scheduling has higher throughput and lower response time than FCFS Policy. It is reasonable solution for batch processing system. Sometimes, it does not ensure fairness because with this scheduling starvation is possible. This policy is generally not acceptable for interactive systems. It leads to higher variances of response times.

SCAN Scheduling:

It offers an improved variance of response time. The drawback of this scheduling policy is that it does not change the direction until edge of disk is reached. Starvation is still possible in this scheduling. Under a light load, SCAN policy is best.

C-SCAN Scheduling:

It maintains high level of throughput while further limiting variance of response times by avoiding discrimination against the innermost and outermost cylinders.

LOOK Scheduling:

The main advantage of this scheduling is that it only performs sweeps large enough to service all requests. It improves efficiency by avoiding unnecessary seek operation. It gives high throughput.

C-LOOK Scheduling:

It gives lower variance of response time than LOOK, at the expense of throughput.

REFERENCES

[1]. H. M. Deitel, *Operating Systems*, 3rd Edition, Pearson Education, Inc. and Dorling Kindersley Publishing, Inc. , 2004

[2]. William Stallings, *Operating Systems, Internal and Design Principles*, 6th Edition, Dorling Kindersley Pvt. Ltd., 2009

[3]. Andrew S. Tanenbaum, *Modern Operating Systems*, 3rd Edition, Pearson Education Inc. , 2008

[4]. A. Silberschatz, P. B. Galvin, and G. Gagne, "Operating System Concepts", 7th Edn. , John Wiley and Sons Inc, 2005

[5]. Sourav Kumar Bhoi, Sanjaya Kumar Panda, and Imran Hossain Faruk, "Design and Performance Evaluation of an Optimized Disk Scheduling Algorithm (ODSA)", International Journal of Computer Applications, Vol. 40, No. 11, Feb 2012, pp. 28-35.

[6]. John Ryan Celis, Dennis Gonzales,Erwinaldgeriko Lagda and Larry Rutaquio Jr. "A Comprehensive Review for Disk Scheduling Algorithms"

[7]. Manish, K. M., "An Improved First Come First Serve(IFCFS) Disk Scheduling Algorithm", Volume 47–No.13,June 2012, pp. 20-24

[8]. Sandipon Saha, *et al,* "A New Heuristic Disk Scheduling Algorithm', International Journal of Science and Technology Researh, Vol. 2, Issue 1, Jan 2013, pp. 49-53.