Access time based on initial head position in disk scheduling

Mr. Naresh Janwa
M.Tech. Scholar
Computer Science & Engineering
Mewar University , Gangrar
Chittorgarh - 312901
Naresh.janwa@gmail.com

Mr. B.L. Pal
Asst. Professor
Computer Science & Engineering
Mewar University , Gangrar
Chittorgarh - 312901
Contact2bl@rediffmail.com

Abstract –As we know that most of the job in computer is performed on the basis of job scheduling algorithm by disk access. The basic parameter of all these algorithm is head movement. The criteria for best algorithm is minimum head movement to perform the same job, whether algorithm used to perform job is LOOK or C-LOOK or C-SCAN or SCAN or SSTF or FIFO. Most jobs in computer depend heavily on disk access. But now, question arises "How to improve disk access time?" This can be done by scheduling the request for disk access for operating system. If the desired disk drive and controller are available the access can be immediately. in any system when the drive is operating, the disk is rotates at constant speed. To read or write, the head must be positioned at the desired track and at the beginning of the desired sector on that track. Track selection involves moving the head in a movable-head system.

The performance of disk schedulers is affected by many factors such as initial head position, workloads, file systems, and disk systems. Disk scheduling performance can be improved by positioning the Read/Write Heads at appropriate position. Scheduler performance tuning is mostly done manually. To automate this process, we propose initial head position in disk scheduling schemes. We conducted experiments to compare the performance and overhead of algorithms.

In this paper, we are going to study or analyze FIFO, SSTF or SCAN algorithm on the basis of increasing the range of head position from 0 - 200 to 0 - 500 because all these algorithms are analyzed previously on the basis of range from 0- to 200 whether it is done by using IBM's disk or HP disk.

Keywords or Index Terms: initial head position, access time, rotational delay, spindle

I. INTRODUCTION

In operating systems, seek time is very important. Since all device requests are linked in queues, the seek time is increased causing the system to slow down. Disk Scheduling Algorithms are used to reduce the total seek time of any request [1].
If the desired disk drive and controller are available, the request can be serviced immediately. If the drive or controller is busy, any new requests for service will be placed on the queue of pending requests for the drive, for a multi programming system with many process, the disk queue may often have several pending request. Thus, when one request is completed the operating system chooses which pending request to service next. This is called scheduling. When designing an operating system, a programmer must consider which scheduling algorithm will perform best for the use the system is going to see. There is no universal "best" scheduling algorithm, and many operating systems use extended or combinations of the scheduling algorithms above [2].

When some data has to be write into or read from the disk, it issues a system call to the operating system. The I/O devices are free the request is serviced, other wise such a request is placed on the separate queue for servicing later. Disk scheduling involves a careful examination of pending requests to determine the most efficient way to service these requests. A disk scheduler examines the positional relationship among waiting requests, then reorders the queue so that the requests will be serviced with minimum seek. The purpose of the study is to obtain the best scheduling algorithm based on the seek time, rotation time and transfer time for moveable head disks. Keeping in view an attempt has been made to design a technique for optimizing the performance of disk scheduling algorithms. it takes access time which is generated using seek time, rotation time and transfer time, as the request of cylinder numbers, current position of read/write head as inputs. On the basis of these inputs, total head movement of each disk scheduling algorithm is calculated under various loads. **Disk Performance Parameters**

•If the queue has only one outstanding request

–All scheduling algorithms behave as FCFS

•Requests for disk service can be affected by the file-allocation method

•Access Time= Seek time+ Rotational latency

Where

-Seek time is the time takes to position the head at the desired track

-Rotational latency is the time takes for the beginning of the sector to reach the head

-Transfer Time is the time taken to transfer the data

Disk bandwidth= total transferred bytes / total time

Where total time is the time between the first request for service and the completion of the last transfer

The OS is responsible for using h/w efficiently for the disk drives, this means having a fast access time and more disk bandwidth [3].

The simplest algorithm is to service requests in the order that they arrive, or First Come First Served (FCFS). This algorithm has poor performance for all but the lightest of loads, since it wastes a lot of time moving between areas on the disk relative to the time spent actually transferring data. Better is to scan the request queue for the request that is nearest to where the head is positioned and process that next. Traditionally, “nearest” is calculated from the difference in cylinder numbers, and this technique is known as Shortest Seek Time First (SSTF).During periods of very high load, and particularly when many requests are arriving for the same. area on the disk, the arm may “stick” in one region, leaving others requests waiting for a long time. This is known as the starvation problem. Even in the absence of complete starvation, this phenomenon increases the service time variance [4]. The SCAN algorithm sweeps back and forth across the disk stopping at each cylinder with pending requests. A variation of SCAN is to sweep in only one direction, and when the end of the disk is reached, to seek back to the beginning.

SCAN also suffers from starvation, but to a less than SSTF. Various authors have proposed adaptations to overcome this problem. The performance of FCFS, SSTF and SCAN has been extensively studied in recent years.

II. DISK STRUCTURE

•Disk drives are addressed as large 1-dimensional arrays of smallest unit of transfer.

•The 1-dimensional array is mapped into the sectors of the disk sequentially:•The first sector (Sector 0) of the first track on the outermost cylinder.

•Mapping proceeds in order through that track, then the rest of the tracks in that cylinder, and then through the rest of the cylinders from outermost to innermost.

Allocation methods are for effective disk space utilization with contiguous allocation the disk movement is minimal.

Modern disk drives have the ability to queue incoming requests and to service them in an out of order fashion. In the disk scheduling problem we are given at any given moment a set of queued requests and we wish to service them in an order which minimizes the access time or equivalently, in an order which maximizes the number of requests serviced per time unit. The disk scheduling problem involves reordering the disk requests in the disk queue so that the disk requests will be serviced with the minimum mechanical motion. If the OS sends a batch of requests to the controller

So why does accessing such a simple device take such a considerable amount of time? This problem arises because accessing a specific sector on the disk requires that the read/write head move to the correct track (seek time) and that the platter rotate until the desired sector is beneath the head (rotational latency). Only after this overhead has been paid can the actual transfer of data begin. The amount of time required to physically move the head and the platter is quite significant and can generally only be decreased to a certain point.

III. BACKGROUND AND RELATED PRIOR WORK

In this section, we will briefly introduce the access time based on initial head position in disk scheduling and discuss previous related work.

In this section we first survey information about disks. Next, we summarize the most effective approaches for disk-scheduling. We conclude the section Factors affecting Disk-Scheduling Algorithm which provide a detailed treatment of the algorithmic techniques which we used for optimization. Although there have been constant stream of proposed alternative massive storage technologies, magnetic disks have dominated secondary storage since the mid sixties [5,6]. Detailed description of magnetic disks can be found in many books. Relative comparison of disks with other massive storage alternatives is given in many books. The early disk-related research in operating systems has been focused on development of scheduling algorithms for efficient use of disks in time-shared manner . Later, operating systems researchers developed new disk scheduling algorithms for new platforms assuming increasingly more realistic and complex disk models [7]. There are surprisingly few detailed quantitative descriptions of mechanical, electrical, and magnetic components of a disk.

Several groups of authors proposed Optimization Systems approaches with relatively little impact of initial head position on algorithm performance. But this paper provides solid starting points to study key performance disk design. The resulting technique explores the optimization by allowing the head moves from appropriate initial state. Since this paper, has been a favorite approach for the optimization of access time due to its simple implementation and wide applicability, the proof of optimality and the extensive empirical performance evaluation studies. Most traditional disk scheduling algorithms are designed to reduce disk-seek time and increase its throughput. FCFS performs operations in order the job arrives. However, the performance of FCFS algorithm is poor. SSTF reduces the total seek time compared to FCFS. The disadvantage of SSTF is starvation that is the R/W head stays in one area of the disk if very busy. These algorithms are not suitable to be applied directly on a real-time system [8].

In the SCAN algorithm, the disk head starts at one end of the disk, and moves toward the centre of the disk, servicing requests as it reaches each cylinder, until it gets to the spindle. From this end, the direction of head movement is reversed and servicing continues. The head continuously scans back and forth across the disk. C-SCAN scheduling is a variant of SCAN designed to provide uniform wait time. Like SCAN, C-SCAN moves the head from one end of the disk to the spindle, servicing the requests along the way. When the head reaches the spindle, it immediately returns to the beginning of the disk, without servicing any requests on return trip. LOOK algorithm is also a variant of SCAN. In LOOK the disk head does not move inward or outward when there is no request in that direction. LOOK performs better than SCAN when load is low but it is equivalent to SCAN when the load is high. A variant of LOOK scheduling is C-LOOK. C-LOOK moves the head in one direction from its current head position, after serving all the requests in current direction, disk head starts to serve the first request in other end without serving the requests in return trip. It provides more uniform wait time for the requests.

Disk Access time = Seek time + Rotational Latency

This traveling head time should be minimized

Throughput - the number of disk requests that are completed in some period [9,10].

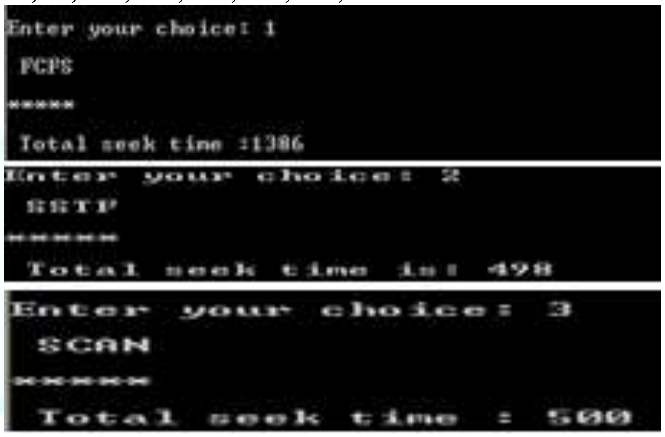
Fairness - some disk requests may have to wait a long time before being served

IV. OUR PROPOSED SCHEME

Let a disk head is initially located at different - different location to analyze scheduling algorithms.

- Assume a disk with 500 tracks and that the disk request queue has random requests in it.
- 8 Requests in work queue.
- Maximum numbers of cylinders are 5.

The requested tracks (received by the disk scheduler) are in the sequence of:
55, 99, 300, 105, 200, 400, 499, 1.



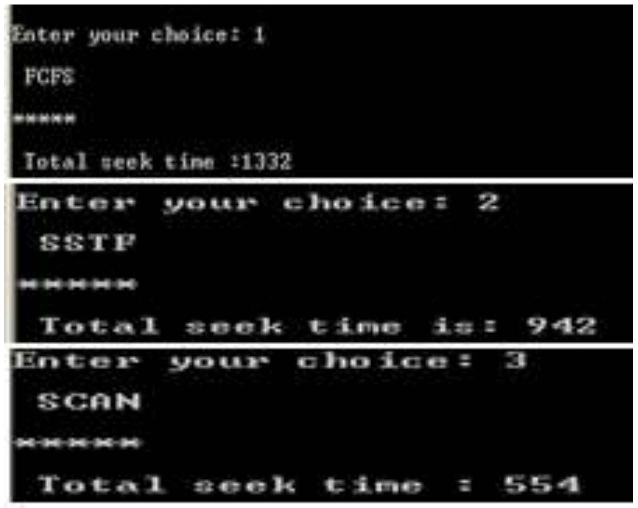
```
Enter your choice: 1
 FCFS
*****
 Total seek time :1386
```

```
Enter your choice: 2
 SSTF
******
 Total seek time is: 498
```

```
Enter your choice: 3
 SCAN
******
 Total seek time : 500
```

Figure 1 when initial head=1

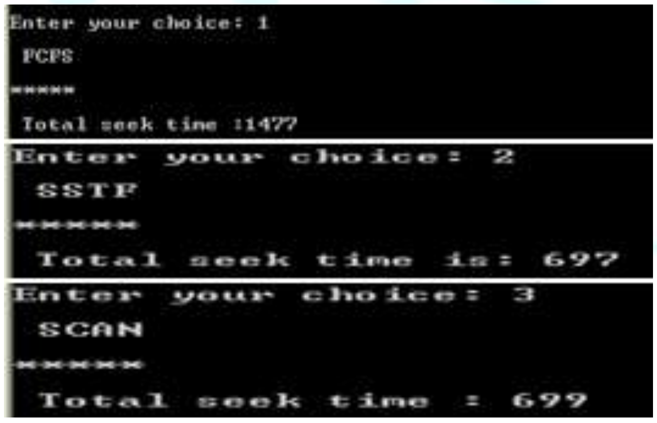
Figure 1 shows that when initial head at 1 then in FCFS total head movement will be 1386 and in SSTF total head movement will be 498 and in SCAN total head movement will be 500.



```
Enter your choice: 1
 FCFS
*****
 Total seek time :1332
Enter your choice: 2
 SSTF
******
 Total seek time is: 942
Enter your choice: 3
 SCAN
******
 Total seek time : 554
```

Figure 2 when initial head=55

Figure 2 shows that when initial head at 55 then in FCFS total head movement will be 1332 and in SSTF total head movement will be 942 and in SCAN total head movement will be 554.

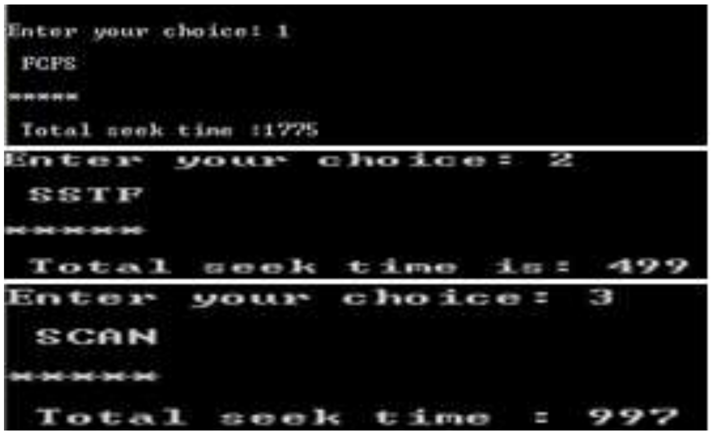


```
Enter your choice: 1
 FCFS
*****
 Total seek time :1477
Enter your choice: 2
 SSTF
******
 Total seek time is: 697
Enter your choice: 3
 SCAN
******
 Total seek time : 699
```

Figure 3 when initial head=200

Figure 3 shows that when initial head at 200 then in FCFS total head movement will be 1477 and in SSTF total head movement will be 697 and in SCAN total head movement will be 699.

Figure 4 shows that when initial head at 498 then in FCFS total head movement will be 1775 and in SSTF total head movement will be 499 and in SCAN total head movement will be 997.



```
Enter your choice: 1
 FCFS
*****
 Total seek time :1775
Enter your choice: 2
 SSTF
******
 Total seek time is: 499
Enter your choice: 3
 SCAN
******
 Total seek time : 997
```

Figure 4 when initial head=498

V. REDUCING ACCESS TIME THROUGH REDUCTION OF HEAD MOVEMENT

Modern disks spend almost twice less power in idle mode than in seek mode. In order to reduce seek time; system-level designer has to carefully arrange data assignment to tracks of disk as well as to properly order data access pattern. Therefore, task-level scheduling is most relevant synthesis task for exploring this trade-off. The advantage of this method is that no hardware alternations are required. However, as it is shown in this paper once the seek time is reduced it is usually more beneficial to reduce disk read/write so that the voltage and the spindle motor speed can be reduced than to exploit this trade-off [8].

VI. PERFORMANCE COMPARISON OF ALGORITHMS

Here Tested Parameter is initial head position.
•To compare various schemes, consider a disk head is initially located at different - different location. •
–Assume a disk with 500 tracks and that the disk request queue has random requests in it.
–8 Requests in work queue.
–Maximum number of cylinders are 5.
•The requested tracks, in the order received by the disk scheduler are
–55, 99, 300, 105, 200, 400, 499, 1.

Scheduling	THM when IHP=1	THM when IHP=55	THM when IHP=200	THM when IHP=498
FCFS	1386	1332	1477	1775
SSTF	498	942	697	499
SCAN	500	554	699	997

Table 1 TMH when IHP at different locations

Where **THM is Total Head Movement** and **IHP is Initial Head Position.**

Experimental Results for Performance based on Initial Head Position is that SSTF gives the optimized result when Initial Head Position is 1. So to Maximize Disk Throughput with Guaranteed Requirements of Initial Head Position is 1. it is the our aim for Proposed Scheme. Conventional Approaches gives the different - different result based on Initial Head Position

VII. CONCLUSION

As we know that our analysis is based on increasing the range of head position from 0-200 to 0- 500. So, our result is different. As per our analysis SSTF is best algorithm among FIFO, SSTF and SCAN because SSTF takes minimum head movement to do the same job while others are taking more head movement whether it is SCAN or FIFO, It does not matter and scheduling is affected by initial head position strategy.

ACKNOWLEDGMENTS

The authors wish to thanks Mr. shiv kumar, Mr. Rohit Maheshwari professors of CS department for the insightful comments and discussions.

Besides our advisor, we would like to thank the rest of my Friends and well wisher: Dr. D.B. Ojha and others, for their encouragement, insightful comments, and hard questions. Last but not the least; we would like to thank my friend gaurav kakhani supporting me spiritually to write this paper.

REFERENCES

[1]. A. Silberschatz, P. B. Galvin and G. Gagne, "Operating System Principles", 7th Edn., John Wiley and Sons, 2008, ISBN 978-81-265-0962-1.

[2]. Andrew S. Tanenbaum, Modern Operating system Concept 3rd Edition, PHI Learning.

[3]. Dhananjay M. Dhamdhere, Operating System: A Concept Based Approach 3rd Edition, Tata McGraw - Hill Education

[4]. H. M. Deitel, "Operating Systems", 2nd Edn., Pearson Education Pte. Ltd., 2002, ISBN 81-7808-035-4.

[5]. W. Stallings, "Operating Systems", 4th Edn., Pearson Education Pte. Ltd., 2007, ISBN 81-7808-503-8.

[6]. Z. Dimitrijevic, R. Rangaswami and E. Y. Chang, "Support for Preemptive Disk Scheduling", IEEE Transactions on computers, Vol. 54, No. 10, Oct 2005.

[7]. C. Tsai, T. Huang, E. Chu, C. Wei and Y. Tsai, "An Efficient Real-Time Disk-Scheduling Framework with Adaptive Quality Guarantee", IEEE Transactions on computers, Vol. 57, No. 5, May 2008.

[8]. John P Hayes "Computer Architecture" McGraw-Hill, 1998 third edition *ISBN* -0070273553, 9780070273559

[9]. H. Kopetz, Real-time systems: design principles for distributed embedded applications, 2nd ed. Springer

[10.] Sourav Kumar Bhoi, Department of CSE NIT, Rourkela Design and Performance Evaluation of an Optimized Disk Scheduling Algorithm (ODSA), International Journal of Computer Applications (0975 – 8887) Volume 40– No.11, February 2012