Simple Sequence Oriented Disk (SSOD) Scheduling Algorithm

Monu Kumar
DIT University, Dehradun

Mukul Panwar
DIT University, Dehradun

Sanjay Bhargava, Ph.D
DIT University, Dehradun

ABSTRACT
As the hard-disk technology has been improved considerably. A significant amount of work also has been done to reduce the seek time of the disk. With the increased speed of processor, faster RAM compatible disk scheduling algorithms had been proposed and some of them are really implemented. The main focus of most of the proposed algorithms is to reduce head movement. In this paper a new disk scheduling algorithm has been proposed ie. Simple Sequence Oriented Disk Scheduling Algorithm (SSOD) which significantly reduces the head movement when compared to some famous already existing disk scheduling algorithms.

Keywords
Disk scheduling, SSTF, C-LOOK, SCAN, FCFS, C-SCAN, LOOK, HEAD MOVEMENT, RQ (Request Queue)

1. INTRODUCTION
The basic characteristics of disk is the seek time and the rotational latency that increases the performance and the efficiency of the system. To increase the performance of disk there is a requirement to schedule the requests in a proper manner. So the disk scheduling algorithms is the major problem for the system that can improve the throughput of the system. However, the performances are increasing day by day but scheduling is still remains a problem because the size of the disk is also increases day by day. In the modern scenario regarding the computer application there is a huge amount of data that we have to access quickly such as audio files and video files but if there is no scheduling algorithm that gives the quick access of that kind of file we have to proposed new algorithm which provides the better services. [1, 2]

Dick Scheduling is the procedure by which the request can access in a quickly accessed manner. There are several disk scheduling algorithm are exit in nature that gives better solution of the disk scheduling [3].

One of the most time consuming operations disk system is head movement since it is a complex task to design better physical devices to move the arm efficiently that will reduce the disk movement. If the speed is increased after a certain extent then it will slip from the desired sectors and will be hard to manage. Most inexpensive disk drive move the head only one track at a time. If the driver has read block from track 10 and desire to read another block which can at track 90. If we follow the simple scheme ie one track at a time than the head will move 80 tracks before reaching to actual destination. Some expensive disk drives equipped with more hardware support can directly jump to desired track[4].

2. PROBLEM DESCRIPTION
The major problem of operating system is disk scheduling because it provides the way that the huge amount of data is stored in the disk. And disk scheduling algorithms gives the methods to access that data on demand in short time.

Primarily disk scheduling problem can be classified in two ways. In the first type more hardware support is added for example RAID (Redundant Array of Independent Disk). It is the combination of many disks that increases the throughput of the system. The another approach to increase the efficiency of the system is by better software system ie by better disk scheduling algorithms which efficiently uses the available resource[3].

The mechanisms that are used by a disk scheduler are: recording of disk request, merging of adjacent request into single large request and delaying a request to the disk drive [3].

Rotational positional sensing (RPS) is a technique that is mainly used in mainframe systems. When seek request is made channel is released for that request. Once seek process is completed i/o device informs when data will rotate under the head. When the appropriate sector is reached under the head the i/o device reestablish the communication to the host if the control unit is busy, or channel is busy then the reconnection will fail and the device will wait for the whole revolution before it can reconnect, this process is called RPS miss[5]

The time taken by the disk driver to complete the request is called as **completion time**.

Completion time=seek time + rotational latency + transmit time.

The time taken by the disk driver to reach the desired request or cylinder number is called as **seek time.**

Seek time=start time + n*c;

Where n=Number of tracks traversed,

c=constant that depends on the disk drive

The waiting time by the disk driver (disk head) to reach the desired sector is called as **Rotational Latency.**

The **Transfer time** to or from the disk depends on the rotational speed of the disk in the following formula [5].

$$\mathbf{T_t=B_t/R_s*N_b}$$

Where $\mathbf{T_t}$=transfer time, $\mathbf{B_t}$=no. of byte to be transfer

$\mathbf{R_s}$=rotation speed in rps, $\mathbf{N_b}$=no. bytes on a track.[5]

3. RELATED WORK
Most of the disk scheduling algorithms like FCFS, SSTF, SCAN, C-SCAN, LOOK and C-LOOK are proposed to reduced the head movement of the disk and the system throughput[8,9,10].

FCFS algorithm performs the operations in sequential order in which order the requests are arrived. However the performance of FCFS is not good because of zig-zag problem (ie. Movement of head from one side to another side). [11, 12]

SSTF performs the operations according to least seek time requests. It gives the better performance in comparison of FCFS but it has a major drawback that is starvation [12,13,14].

SCAN algorithm, serves the requests from the start of the disk arms and fetches the requests towards the spindle[15, 13]

C-SCAN algorithm is the improvement of SCAN scheduling algorithm. It serves the requests towards the spindle and then moves to the start of the disk without reading any request and then again serves the requests toward the spindle.[15, 16]

LOOK algorithm is just similar to SCAN but in Look the disk head can move inward or outward according to the request. The performance of LOOK is better than SCAN [7,17]

C- **LOOK** is the improvement of LOOK It moves the head in one direction from its current position, and read all the requests in same direction. After reading all the request in current direction the head moves at the start in opposite direction without reading any request and then read again the same direction [6, 3,17].

3.1 Existing Disk scheduling algorithms

Several algorithms are exit for disk scheduling. It has been explain them with an example, suppose the request queue (having 200 cylinders from 0 to 199) is 180,50,20,10,15,25,80,100,150 and starting head position is 100.

3.1.1 FCFS

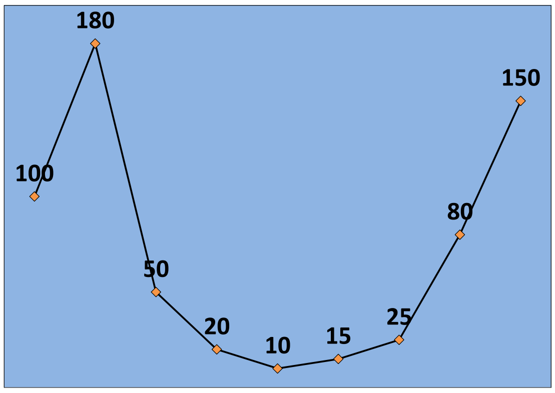

Fig. 1: Head movement in FCFS

Head movement= (180-100)+ (180-50)+ (50-20)+ (20-10)+ (15-10)+ (25- 15)+ (80-25)+ (150-80)=290. *(As shown in fig.1 above).*

3.1.2 SSTF

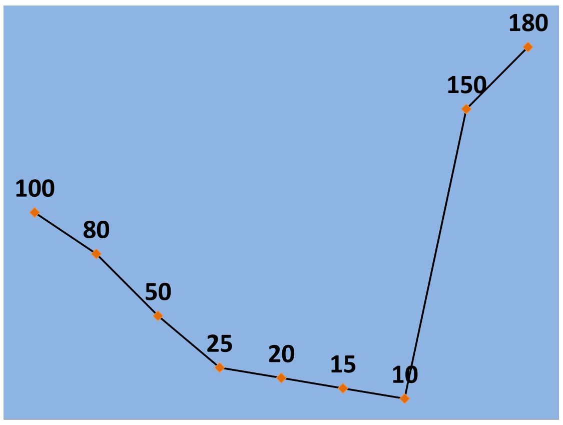

Fig. 2: Head movement in SSTF

Head movement = (100-80) + (80-50) + (50-25) + (25-20)+ (20-15)+ (15-10)+ (150-10)+ (180-150)=260. *(As shown in fig.2 above)*

3.1.3 SCAN

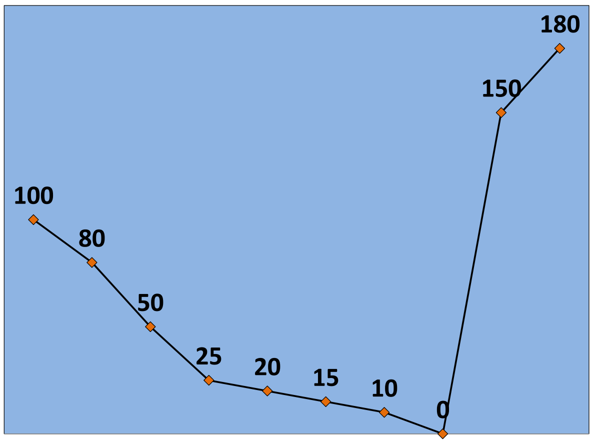

Fig. 3: Head movement in SCAN

Head movement=(100-80)+ (80-50)+ (50-25)+ (25-20)+ (20-15)+ (15-10)+ (10-0)+ (150-0)+ (180-150)=280. *(As shown in fig.3 above)*

3.1.4 C-SCAN

Head movement=(150-100)+ (180-150)+ (199-180)+ (199-0)+ (10-0)+ (15-10)+ (20-15)+ (25-20)+ (50-25)+ (80-50)=378 *(As shown in fig. 4 above).*

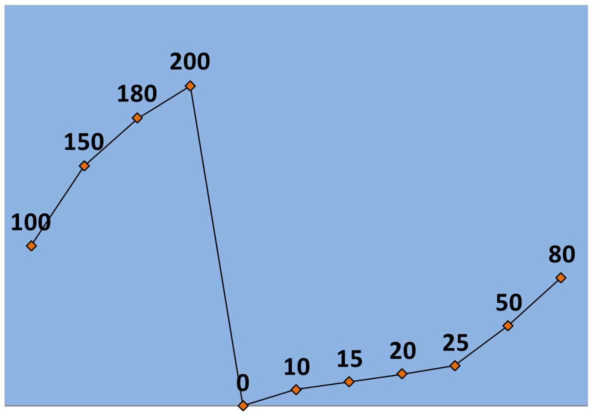

Fig. 4: Head movement in C-SCAN

3.1.5 LOOK

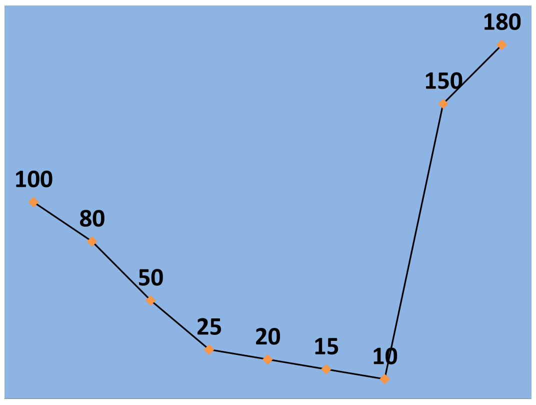

Fig. 5: Head movement in LOOK

Head movement=(100-80)+(80-50)+ (50-25)+ (25-20)+ (20-15)+ (15-10)+ (150-10)+ (180-150)=260. *(As shown in fig.5 above).*

3.1.6 C-LOOK

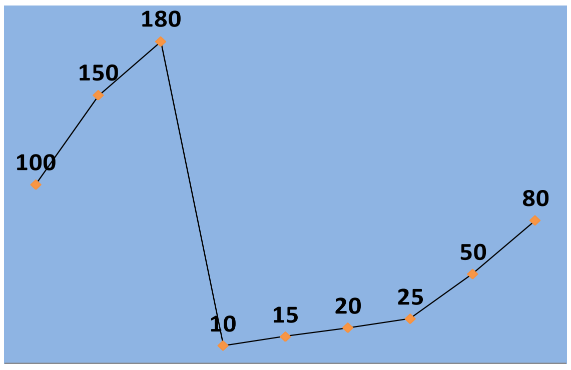

Fig. 6: Head movement in C-LOOK

Head movement=(150-100)+ (180-150)+ (180-10)+ (15-10)+ (20-15)+ (25-20)+ (50-25)+ (80-50)=320. *(As shown in fig.6 above).*

4. PROPOSED ALGORITHM

In this algorithm there two variables are used that is MAX and MIN and they contains the maximum and minimum cylinder number respectively. HEAD holds the current head position. According to this algorithm, the head moves at one end to the last request where either the left or right distance is minimum.

This **algorithm** works as follows:

1. RQ[ ] is the request queue which contains the requests of n cylinders and HEAD is the initial head position.

2. Calculate MAX, the maximum cylinder request in RQ.

3. Calculate MIN, the minimum cylinder request in RQ.

4. if (MAX - HEAD) < (HEAD - MIN), then

Sort the RQ in **Descending order**, and move the head to the starting position of RQ and read the requests from i=0 to n

Otherwise

Sort the RQ in **Ascending order**, and move the head to the starting position of RQ and read the requests from i=0 to n.

Implementation of the proposed algorithm:

Let RQ = [180,50,20,10,15,25,80,100,150]

HEAD = 100;

MAX = 180;

MIN = 10;

Now check MAX-HEAD = 180-100 = 80;

HEAD-MIN = 100-10 = 90;

Here 80<90, so sort RQ in descending order ie.

RQ = [180,150,100,80,50,25,20,15,10]

Head reads the request from 180 to 10

Total head movement = (180-100) + (180-10) = 250

Head movement is illustrated in fig. 7 below.

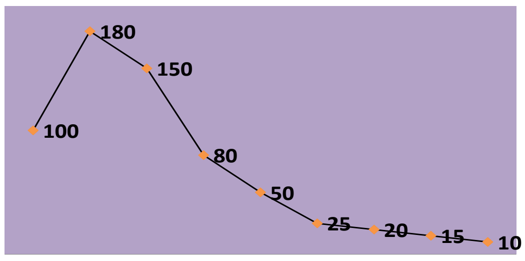

Fig. 7: Head movement in PROPOSED ALGORITHM

5. COMPARISONS WITH EXISTING ALGORITHMS

It has been compared the head movement in various existing algorithms with the head movement in the proposed algorithm. The comparison is summarized in the following Table 1.

Table 1: Comparison in Head Movement

S.No.	Algorithms	Head Movement
1	FCFS	290
2	SSTF	260
3	SCAN	280
4	C-SCAN	378
5	LOOK	260
6	C-LOOK	320
7	New SSOD	250

The results are also shown by the following fig. 8.

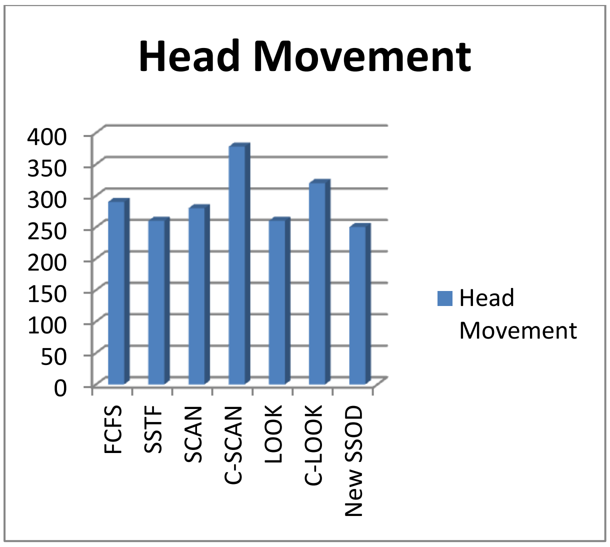

Fig. 8: Head movement comparison as shown in Table 1

Let us take **another example** to compare the proposed scheduling algorithms with existing scheduling algorithms.

Suppose the request queue (having 200 cylinders from 0 to 199) is 180, 100, 190, 10, 90, 30, 50 and starting head position is 100. The comparison of the head movement is as shown in Table 2 below.

Table 2: Comparison in Head Movement

S.No.	Algorithms	Head Movement
1	FCFS	580
2	SSTF	280
3	SCAN	300
4	C-SCAN	390
5	LOOK	280
6	C-LOOK	350
7	**New SSOD**	**260**

The results are also shown by the following fig. 9.

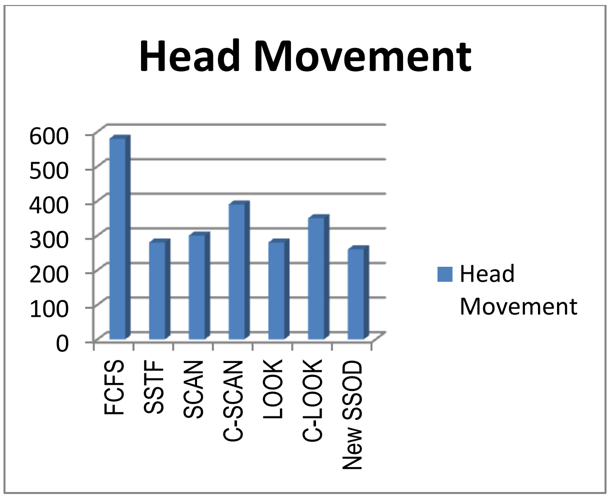

Fig. 9: Head movement comparison as shown in Table 2

Based on the head movement criteria, It has been compared the performance of all the above algorithms for 500 cases and then obtained the following results as shown in Table 3 below.

Table 3: Comparison of proposed algorithm performance with existing algorithms

S.No.	Algorithms	Number of times Head movement was found to be least	% of times Head movement was found to be least
1	FCFS	5	1
2	SSTF	8	1.6
3	SCAN	3	0.6
4	C-SCAN	3	0.6
5	LOOK	11	2.2
6	C-LOOK	3	0.6
7	**New SSOD**	**467**	**93.4**

Clearly the results, as shown in Table 3, show the effectiveness of proposed algorithm. The same has been shown in Fig. 10 below.

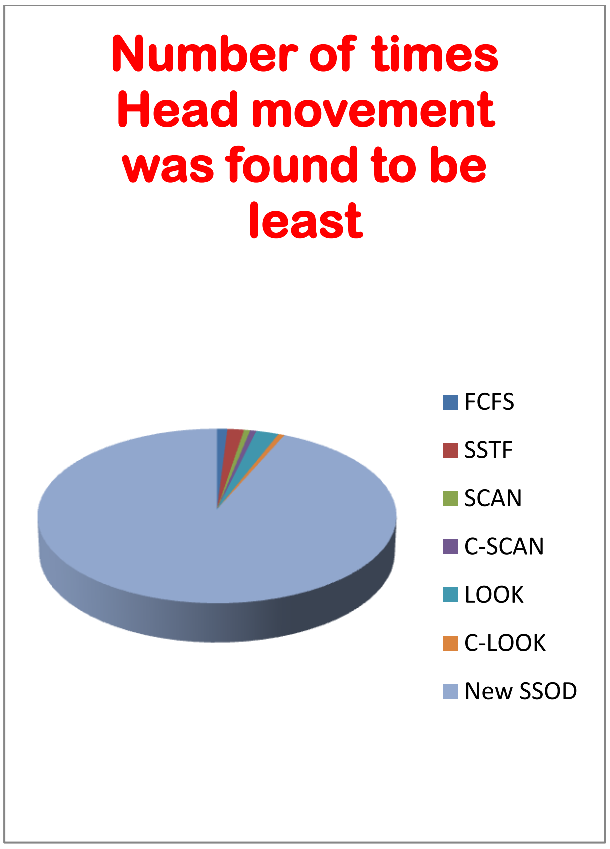

Fig. 10: Number of times the Head movement was found least (Out of 500)

6. CONCLUSION

In this paper a new disk scheduling algorithm has been proposed with lesser head movement when compared to major disk scheduling algorithms. Some random inputs are taken and head movement is calculated with that input. Same set of input has been applied to FCFS, SSTF, SCAN, C-SCAN, LOOK and C-LOOK. The head movement of new SSOD algorithm is minimum among all mentioned algorithms in most of the cases. If this newly proposed algorithm is implemented the overall disk performance and system throughput will increase.

7. REFERENCES

[1] C. Ruemmler and J. Wilkes. An introduction to disk drive modeling. IEEE computer, 27(3):17–29, March 1994.

[2] Lee, S.hyun. & Kim Mi Na, (2008) “This is my paper”, ABC Transactions on ECE, Vol. 10, No. 5, pp120-122

[3] R.Muthu Selvi and R.Rjaram, a Genetic Based Approach for Multi- Objective Optimization of Disk Scheduling to reduce completion time and Missed Task (Vol,1 No.4, August 2011).

[4] Gary J.Nutt, Operating System-A modern perspective(second edition).

[5] Willian Stallings, Operating system(Second Edition, fifth edition).

[6] Operating System Principles (6th edition) Abraham Silberschatz, Peter Bare Galvin, Greg Gagne.

[7] A New Heuristic Disk Scheduling Algorithm Sandipon Saha, Md. Nasim Akhter, Mohammod AbulKashem IJSTR RESARCH VOLUME 2 ISSUE1,JANUARY 2013 ISSN 2277-8616

[8] [8].Denning, P.J. (1967) "Effects of scheduling on file memory operations", AFIPS Joint Computer Conferences, pp9-21.

[9] Geist, R & Daniel, S, (1987) "A continuum of disk scheduling algorithms", ACM Transactions on Computing Systems, Vol.5, No.1, pp77-92.

[10] Chen, S., Stankovic, J.A., Kurose, J.F. and Towsley, D, (1991) "Performance evaluation of two new disk scheduling algorithms for real-time systems", Journal of Real-Time System, Vol.3, No.3, pp307-336.

[11] Seltzer, M., Chen, P. and Ousterhout, J, (1990) "Disk scheduling revisited", USENIX Tech Conference, pp313-324.

[12] Hofri, M, (1980) "Disk scheduling: FCFS vs. SSTF revisited", Communications of the ACM, Vol.23, No. 11, pp645-653.

[13] Chen, T.S., Yang, W.P. and Lee, R.C.T, (1992) "Amortized analysis of some disk scheduling algorithms: SSTF, SCAN and N-Step SCAN", BIT 32, pp546-558.

[14] Worthington, B.L., Ganger, G.R. and Patt, Y.N, (1994) "Scheduling algorithms for modern disk drives", Proceedings of ACM SIGMETRICS Conference, pp241-251.

[15] Coffman, E.G. and Hofri, M, (1982) "On the expected performance of scanning disks", SIAM Journal on Computing, Vol.11, No.1, pp60-70.

[16] Coffman, E.G, (1973) "A note on the relative performance of two disk scanning policies", Information Processing Letters, Vol.2, No. 1, pp15 17.

[17] Sohn, J.M. and Kim, G.Y, (1997) "Earliest-deadline-first scheduling on non-preemptive real-time threads for continuous media server", Proceedings of HPCN, Vol. 1225, pp950-956.

[18] http://airccse.org/journal/ijitcs/papers/0811ijitcs07.pdf

[19] http://wenku.baidu.com/view/e9bebee90975f46527d3e13 4.

[20] http://www.ijstr/final-print/jan2013/A-New-Heuristic-Disk-Scheduling-Algorithm.pdf.