A New Heuristic Disk Scheduling Algorithm

*Sandipon Saha, **Md. Nasim Akhter, ***Mohammod Abul Kashem

Abstract:- Since the invention of the movable head disk, people have improved I/O performance by intelligent scheduling of disk accesses. Processor speed and memory capacity are increasing several times faster than disk speed. This disparity suggests that disk I/O performance will become an important bottleneck .Methods are needed for using disks more efficiently. Past analysis of disk scheduling algorithms has largely been experimental and little attempt has been made to develop algorithms with provable performance guarantees. Disk performance management is an increasingly important aspect of operating system research and development. In this paper a new disk scheduling algorithm has been proposed to reduce the number of movement of head. It is observed that in existing scheduling algorithms the number of head movement is high. But we proposed a new real-time disk scheduling algorithm that reduces the head movement therefore it maximizes throughput for modern storage devices.

Key words: Disk scheduling, SSTF, C-LOOK, SCAN, FCFS, C-SCAN, LOOK, HEAD MOVEMENT

1. Introduction

Scheduling is a fundamental operating system function, since almost all computer resources are scheduled before use. The disk is of course, one of the computer resources. For the disk drivers, meeting this responsibility entails having fast access time and large disk bandwidth. Processor speed and disk and memory capacity are increasing by over 40% 40%per year. In contrast, disk speed is increasing more gradually, growing by only 7%per year [13]. Since this rate is unlikely to change substantially in the near future, I/O performance may become the system bottleneck. However, despite the difficulty of improving mechanical components, we can still aim to *use* the disks more efficiently. The access time has two major components. For example, disks generally operate at a small fraction of their maximum bandwidth. Researchers have demonstrated experimentally that sophisticated disk head scheduling algorithms can deliver higher throughput [20, 12, 23]. This past research has focused almost exclusively on two types of work loads : synthetic work loads , where disk requests are randomly and uniformly distributed across the disk, and more recently, traces, where the requests to an actual disk are recorded and used as a testing ground for algorithms. However, for these or for general work loads, researchers have made little attempt to develop algorithms with provable performance *guarantees*. In addition, no one has determined the computational complexity of the disk scheduling problem. There is a risk that synthetic workloads and traces from a few environments may not represent all possible situations. And the **seek time** is the time for the disk arm to move the heads to the cylinder containing the desired sector. 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.

- Sandipon Saha, Md. Nasim Akhter, Mohammod Abul Kashem
- Department of CSE, Dhaka University of Engineering & Technology.
- Department of CSE, Dhaka University of Engineering & Technology.
- Department of CSE, Dhaka University of Engineering & Technology.

We can improve both the access time and bandwidth by scheduling the servicing of disk I/O requests in a good order. Whenever a process needs I/O to or from the disk, it issues a system call to the operating system. The request specifies several pieces of information:

- Whether this operation is input or output.
- Whether the disk address for the transfer is
- What the memory address for the transfer is
- What the number of sectors to be transferred is

If the disk driver and controller are available, the request can be serviced immediately. If the driver or controller is busy, any new, request for service will be placed in the queue of pending requests for the drive. For a multiprogramming system with many processes, the disk queue may often have several pending requests. Thus, when one request is complete, the operating system chooses which pending request to service next.

2. Problem Statement and motivation:

Several algorithms exist to schedule the servicing of disk I/O requests. We illustrate them with a requests queue(10-199) :36,180,120,10,15,40,188,150 , 120 , 168.Head starts at 130.

1. FCFS: The simplest form of disk scheduling is, of course, the first-come,first-served algorithm.But it generally does not provide the faster service.Concider , for example a disk queue with requests for I/O to blocks on cylinders: Queue(10-199) : 36,180,120, 10 , 15,40,188,150 , 168 .And head current position is :130. This schedule is diagrammed in figur 1.1

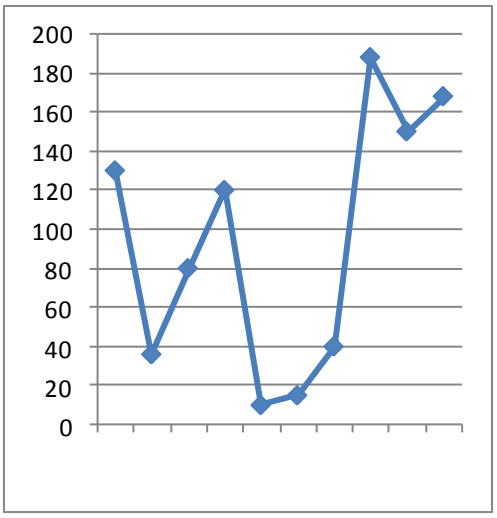

Figure:1.1

Head movement: (130-36)+ (80-36)+(120-80) +(120-10) +(15-10) +(40-15) +(188-40)+ (188-150) +(150-120) +(168-120)=582

In FCFS total head movement: 582

2. SSTF: Shortest Seek Time First-Selects the request with the minimum seek time from the current head osition.Since seek time increases with the number of cylinders traversed by the head, SSTF chooses the pending request closest to the current head position.SSTF scheduling is a form of SJF scheduling:may cause starvation of some requests. Concider , for example a disk queue with requests for I/O to blocks on cylinders:

Queue(10-199): 36,180,120,10,15,40,188,150 , 168. And head current position is :130. This schedule is diagrammed in figur 1.2

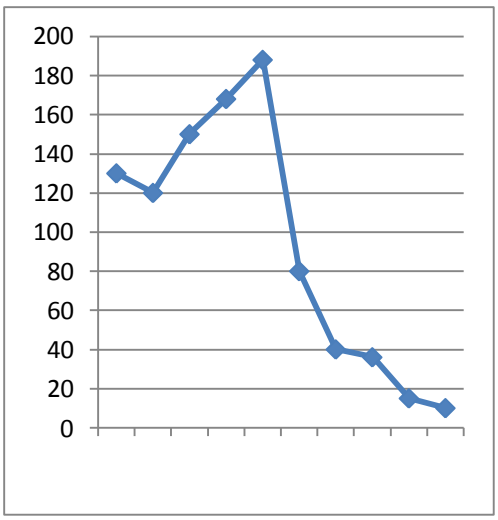

Figure :1.2

Head Movement:(130-120) +(150-120) +(168-150) +(188-168) +(188-80) +(80-40) +(40-36) +(36-15) +(15-10)=256

In SSTF total head movement: 256.

3. SCAN: The disk arm starts at one end of the disk and moves toward the other end, servicing requests until will get to the other end of the disk, where the head movement is reversed and the servicing continues.Some time called the elevator algorithm.Concider , for example a disk queue with requests for I/O to blocks on cylinders:Queue(10-199) :36,180,120,10,15,40,188,150 , 168. And head current position is :130. This schedule is diagrammed in figur 1.3

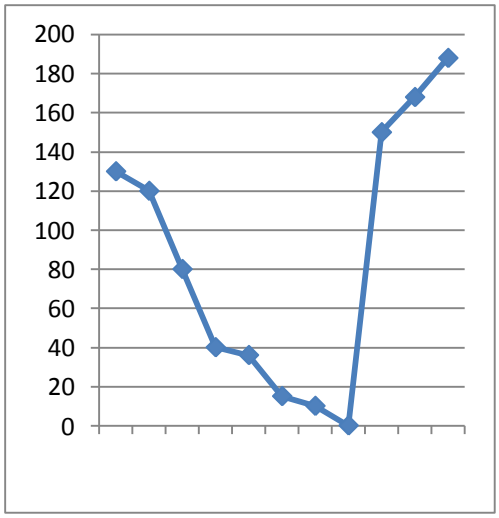

Figure:1.3

Head movement:(130-120) +(120-80) +(80-40) +(40-36) +(36-15) +(15-10) +(10-0) +(150-0) +(168-150) +(188-168)=318

In SCAN total head movement: 318.

4. C-SCAN: Provides a more uniform wait time than SCAN. The head moves from one end of the disk to the other. Servicing requests as it goes. However, when it reaches of the other end, it immediately will return to the beginning of the disk, without servicing any requests on the return trip. Treats the cylinders as a wraparound circular list from the first cylinder to the last one. Concider , for example a disk queue with requests for I/O to blocks on cylinders:Queue(10-199):36,180,120,10,15,40,188,150, 168. And head current position is :130. This schedule is diagrammed in figur 1.4

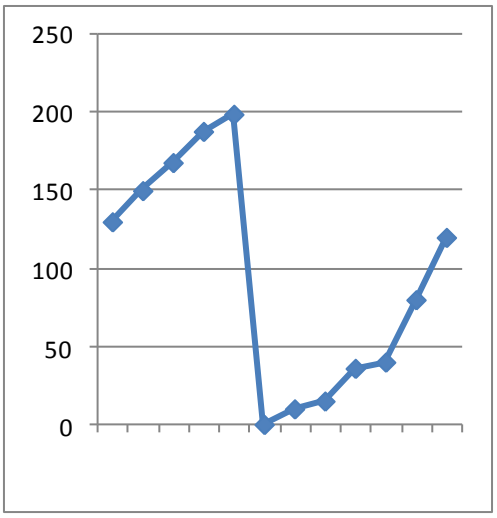

Figure:1.4

Head movement: (150-130) +(168-150) +(188-168)+(199-188) +(199-0) +(10-0) +(15-10) +(36-15) +(40-36) +(80-40) +(120-80)=388.

In C-SCAN total head movement: 388

5. C-LOOK: A version of C-SCAN. Arm goes only as far as the last request in each direction, the reverses direction immediately, without first going all the way to the end of the disk. Concider , for example a disk queue with requests for I/O to blocks on cylinders: Queue(10-199): 36,180,120,10,15, 40,188,150 ,168. And head current position is :130. This schedule is diagrammed in figur 1.5

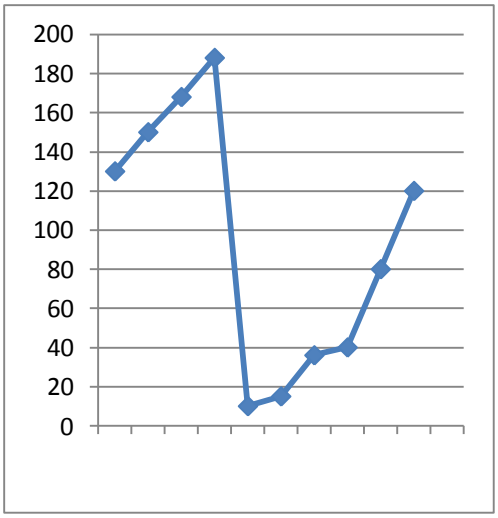

Figure 1.5

Head movement: (150-130)+(168-150) +(188-168) +(188-10) +(15-10) +(36-15) +(40-36) +(80-40) +(120-80)=346

In C-LOOK total head movement: 346

6. Look: LOOK is similar to SCAN in that the heads sweep across the disk surface in both directions performing reads and writes. However, unlike SCAN, which visits the innermost and outermost cylinders each sweep, LOOK will change directions when it has reached the last request in the current direction. Concider , for example a disk queue with requests for I/O to blocks on cylinders : Queue(10-199): 36,180,120,10,15, 40,188,150 ,168. And head current position is :130. This schedule is diagrammed in figur 1.6

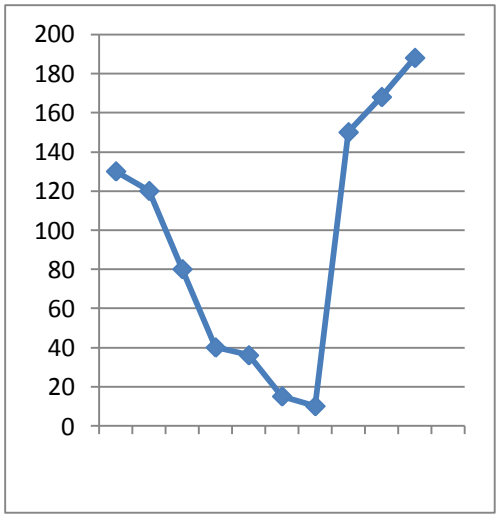

Figure:1.6

Head movement: (130-120) +(120-80) +(80-40)+(40-36) +(36-15) +(15-10) +(150-10) +(168-150) +(188-168)=298

In LOOK total head movement: 298

3. Proposed Disk Scheduling Algorithm:

At first sorting in ascending order of all cylinders input blocks by using any sorting method. Find the distance between the smallest block number and current head position. Let it is P and again find the distance between the largest block number and current head position. Let it is Q. Sequentially move and reached head from these block to the highest block number. Else head moves sequentially from its current position to the highest block number in forward and again in backward which block is not visited. Then sequentially move and reached head from these block to the largest block number.

New Heuristic Disk Scheduling Algorithm (a, n, count, h)

1. // a [] is an array that contains cylinders number. N is the number of cylinder. Count is use
2. // for counting head movement's. h denote the present head position.
3. Sorting input blocks of cylinder number in ascending order by any sorting method.
4. Input present head position h.
5. Temp:=h;
6. For i :=1 to n do
   If(a[i]>=h) { Position=i; break;}
7. Left_distance:=head-a [1];
   Right_distance:=a[n]-head;
8. Count:=0;
9. If (Left_distance< Right_distance)
   {

```
                For i:=position-1 to 1 step -1 do {
                count:=count+Temp-a [i]; Temp:=a[i];
                }
                 count :=count+[position]-a[1];
                For I: =position+1 to n do
                  count: =count + a [i]-a [i-1];
          }
    Else
      {
         For i:=position to n do
         {count:=count+a[i]-head;
            head:=a[i];}
          Count:=count+a[n]-a[position-1];
        For i:=position-1 to  2 do
            Count:=count+a[i]-a[i-1];
       }
10.  Return count;  // total head movement
```

Graphical representation of proposed algorithm: Concider , for example a disk queue with requests for I/O to blocks on cylinders : Queue(10-199) :36,180,120,10,15, 40,188,150 ,168. And head current position is :130. This schedule is diagrammed in figur 2

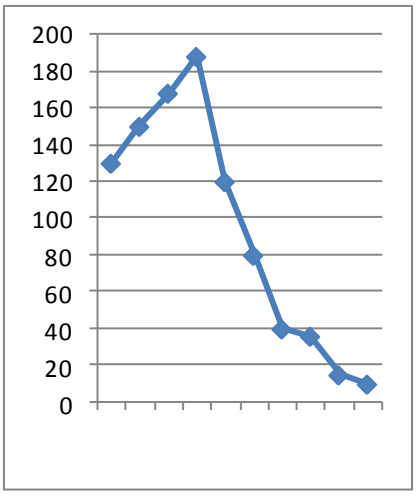

Figure: 2

Head movement: (150-130)+(168-150) +(188-168) +(188-120) +(120-80) +(80-40) +(40-36) +(36-15) +(15-10)=236

In new heuristic algorithm total head movement: 236

General Equeation :
Let ,
Left distance=Ld
Right distance =Rd
Head position=Hp
Request queue=a[]
Max queue position=n
Total Head Movement=Hm
Ld=Hp-a[1]
Rd=a[n]-Hp

$$Hm = \begin{cases} Rd + a[n] - a[1]; Ld > Rd \\ Ld + a[n] - a[1]; Ld < Rd \end{cases}$$

Calculation: Let ,
Queue(10-199) :36,180,120,10,15,40,188,150 , 168.
Head starts at: 130.
Ld=130-10=120
Rd=188-130=58
Here, Ld>Rd so,
Hm=58+188-10=236
Total head movement=236(For new algorithm)

Comparisons table among proposed and existing algorithms:

SL.NO	Name of Algorithm	Number of head movement
1.	FCFS	582
2.	C-SCAN	388
3.	SSTF	256
4.	C-LOOK	346
5.	LOOK	298
6.	SCAN	318
7.	**NEW Heuristic**	**236**

Comparisons Graph among proposed and existing algorithms. That show performances:

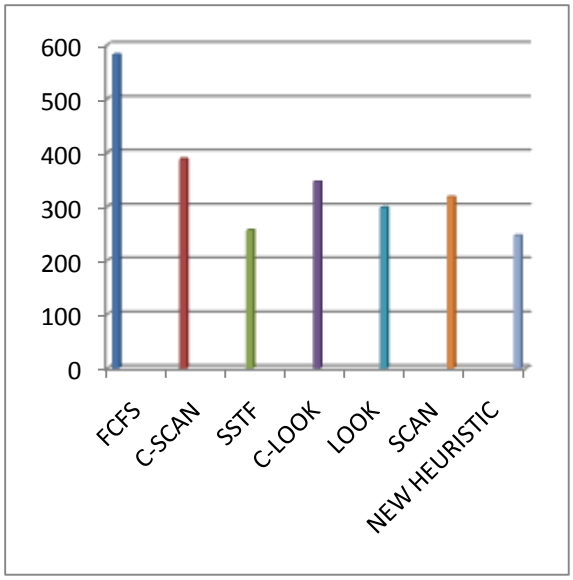

Figure:3

Limitations:

1. Sometime number of head movement is equal to SSTF or LOOK scheduling.
2. When input blocks are stay in ascending order without sorting then FCFS is best. But in dynamic allocation of cylinder it is almost impossible.

Conclusion:

In conclusion, we have presented a new real-time disk scheduler that imposes almost no performance penalty over non-real-time optimal schedulers when given sufficient slack time. We showed how to characterize a device's performance. From the above experiment and comparison of proposed algorithm with existing algorithm it is clear to us that the existing algorithm reduces head movement. Who wants to work with disk scheduling this algorithm will open new era for them. This would help the new generation to be go ahead.

References

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

[2] Mordern operating system (2nd edition) Andrew S. Tanenbaum .

[3] Operating Systems: A Concept-based Approach(2E) D.M. Dhamdhere

[4] Intel, And Seagate. Serial Ata native command queuing july 2003.

[5] Kumar, R. Fairness in disk scheduling. Master thesis, Indian Institute of science, Bangladesh, India, Jan. 1993.

[6] SCSI architecture model-3(SAM3).Tech. Rep. T10 project 1561-D ,revision 14,International for information Committee Technology standards (INCITS),T10 Technical Committee. Sept. 2004.

[7] Chang, R., Shih, W., And Chnag, R. Real-time disk scheduling for multimedia application with deadline-modification-scan scheme. International Journal of Time-critical Computing System 19(2000),149-168.

[8] Dees, B. Native Command queuing-advanced performance in desktop storage. Potentials, IEEE 24, 4(2005),4-7.

[9] Frank, H .Analysis and optimization of disk storage devices for Time-sharing system. J ACM 16, 4(1969),602-620.

[10] Huang, Y., And Huang, J. Disk scheduling on multimedia storage servers. Computers, IEEE Transactions on 53, 1(2004), 77-82.

[11] Reddy,A.L.N.,And Wyllie,J. Disk Scheduling in a multimedia I/O system. ACM, pp.225-223.

[12] Ruemmler, C., And wiklkes,J. An introduction to disk drive modeling. Computer 27, 3(1994),17-28.

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

[14] Seaman, P.H., Lind, R. A., And Wilson, T. L. An analysis of auxiliary-storage activity. IBM System Journal 5, 3(1996), 158-170.

[15] Seltzer, M., Chen, P., And Ousterhout, J. Disk scheduling revisited, Jan . 1990.

[16] Yu, Y . J.,Shin, D. I., Eom, H., And Yeom , H. y. NCQ vs.I/O scheduler: Preventing unexpected misbehaviors. Trans. Storage 6, (2010), 1-37.

[17] Zhu, Y. Evolution of scheduling algorithms for real-time disk I/O. Tech.rep., Department of computer science and Engineering, University of Nebraska, May 2002.

[18] Reddy, A.L.N., Wyllie, J., and Wijyaratne, K.B.R. disk scheduling in a multimedia I/O system. ACM Trans. Multimedia Comput. Commun. Appl. 1, 1(2005), 37-59.

[19] Mesnier, M. P.,Wachs, M., Sambasivan, R.R. , Zheng, A.X., And Ganger, G .R . Modeling the relative fitness of storage. Sigmetrics Perform. Eval. Rev. 35, 1(2007),37