IJCSE International Journal of Computer Sciences and Engineering Open Access
Review Paper Volume-04, Issue-05 E-ISSN: 2347-2693

Performance Analysis of Disk Scheduling Algorithms

C.Mallikarjuna[1*], P.Chitti Babu[2]

[1,2]Department of MCA, Annamacharya PG College of Computer Studies, JNTUAnantapur, Andhrapradesh, India.

Available online at: www.ijcseonline.org

Received: Apr/21/2016 Revised: May/04/2016 Accepted: May/18/2016 Published: May/31/2016

Abstract— This paper aims to make performance analysis of various disk scheduling algorithms based on various factors. In these disks scheduling algorithms we consider First Come First Serve (FCFS), Shortest Seek Time First (SSTF), Scan, Look, C-Scan, and C-Look Scheduling Algorithms. Based on head movements of various disk scheduling algorithms we made performance analysis. These head movements are calculated for different disk scheduling algorithms while serving disk requests. Here we consider the performance factors such as access time, Disk throughput, Disk Utilization etc.

Keywords—Disk scheduling,Seek Time,Disk throughput,Disk utilization.

I. INTRODUCTION

From the starting development of computers, computer hardware can developed with tremendous speed. The hard drive of a computer system utilizes multiple discs coated with a magnetic material to record and read data[2]. These magnetized "platters" spin rapidly within the device and are read by magnetic heads attached to mechanized arms, similar to the design of a record player. In hard drives that use more than one platter, multiple arms are used to record and read data from each individual platter [3]. Data can be stored on both sides of the discs, and arms can move to the inner and outer portions of each one. When your computer stores data on its hard drive, it doesn't just throw magnetized nails into a box, all jumbled up together. The data is stored in a very orderly pattern on each platter. Bits of data are arranged in concentric, circular paths called tracks. Each track is broken up into smaller areas called sectors [5]. Part of the hard drive stores a map of sectors that have already been used up and others that are still free. (In Windows, this map is called the File Allocation Table or FAT.) When the computer wants to store new information, it takes a look at the map to find some free sectors[2]. Then it instructs the read-write head to move across the platter to exactly the right location and store the data there. To read information, the same process runs in reverse[3].

II. ANATOMY OF DISK

In the following figure physical structure of the hard disk is represented. In figure we describe the structure of a disk and how information storage on it. A disk is composed with several platters. Every platter consists of several rings or tracks. The rings are divided into various sectors where actually the information is stored [5].

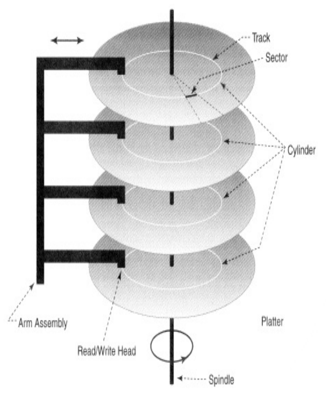

Fig 1:Physical structure of Hard Disk.

The rings with similar positions on different platters are said to form a cylinder. As the disk spins around a spindle, the heads transfer the information from the sectors along the rings. Note that information can be read from the cylinder surface without any additional lateral head movement. So it is always a good idea to organize all information sequentially along the cylinder. This is done by first putting information along a ring and then carrying on with it across to a different platter on the cylinder [6].

III. DISK PERFORMANCE FACTORS

The performance of a hard disk is very important to the overall speed of the system. A slow hard disk having the potential to hinder a fast processor like no other system component and the effective speed of a hard disk is determined by a number of factors [1].

- **Seek Time**

Seek time is the time taken for a hard disk controller to locate a specific piece of stored data. Other delays include transfer time (data rate) and rotational delay (latency) [1]. When anything is read or written to a disc drive, the read/write head of the disc needs to move to the right position. The actual physical positioning of the read/write head of the disc is called seeking [6]. The amount of time that it takes the read/write head of the disc to to move from on part of the disk to another is called the seek time. The seek time can differ for a given disc due to the varying distance from the start point to where the read/write head has been instructed to go. Because of these variables, seek time is generally measured as an average seek time[1].

- **Latency Time**

Disk Latency refers to the time delay between a request for data and the return of the data. It is a simple one, but this time factor is very important to the performance of the. It depends on the rotational speed of the disk. Sometimes it is also referred to as Rotational Latency or Rotational Delay. It should be measured in RPM (Revolutions per Second) [1]. The disk has plates that rotate with a speed expressed in **Revolutions per Minute** or "**RPM**". That is the number of times the plates will do a full rotate in one minutes time. Since the disk arm and the head (who does to actual read or write) is fixed in one position it will often have to wait for the plate to spin to the right position[1]. Common RPMs is **5400** or **7200** for consumer SATA disks and **10000** or **15000** RPM for high performance server / SAN disks. This implies following results.

RPM	5400	7200	10000	15000
Rotations Per sec	90	20	166	250
AvgRotational Latency	11.1	8.3	6	4

Table 1:Rotational Latency for Various Disk Capacities.

So for the disk to spin the plate one full rotation takes from 4 to 11 milliseconds depending on the RPM. This is called the Rotational Delay and is important since the disk can at any moment be given an instruction to read at any sector of any track.

- **Disk Transfer Time**

The transfer time to r from the disk depends on the rotational speed of the disk. This is calculated as follows [1].

$$T = \frac{b}{rN} \qquad (1)$$

Where T=Transfer Time, b=Number of bytes to be transferred, N=Number of bytes on a track and r=Rotational speed in Revolutions per second.

- **Access Time**

Disk access time is the total time it takes the computer to process a data request from the processor and then retrieves the required data from a storage device. To get the Access Time we are adding the Seek Time, Rotational Delay and Disk Transfer time[1]. Thus the average access time can be expressed with following equation.

$$T_a = TS + \frac{1}{2r} + \frac{b}{rN} \qquad (2)$$

- **Utilization**

Because transfer rates vary among disks, most operating systems do not report disk utilization directly. Instead, they report the number of data transfers per second.

- **Disk Throughput**

It is defined as the amount of data transferred from one place to another or processed in specified amount of time. Data transfer rates for hard disks and networks are measured in terms of throughput. Typically throughputs are measured in Kbps, Mbps and Gbps. Throughput is usually used to measure device performance for large block random or sequential operations such file transfer and audio/video streaming. The formula for calculating Disk throughput is as follows [1].

$$Throughput = IOPS * Block\_Size \qquad (3)$$

Where IOPS=Input and output operations per second and Block_size is the amount of data transferred per second**.**

- **IOPS**

IOPS are used to define the performance of a given disk of disk array. To calculate the actual IOPS of a given disk we require information about Average Latency and Average Seek Time. By using following formula we calculate IOPS[1].

$$IOPS = \frac{1}{Avg\ Latency + Avg\ Seek\ Time} \qquad (4)$$

In the case of sequential IO operations require significantly shorter head movements in comparison to random IO operations. During sequential IO operations head assembly moves only between adjacent tracks and stays in the position until all the blocks on the track are accessed. In simplified form Drive throughput for large block sequential IO can be evaluated using the formula below.

$$Throughput = R_s * \frac{T_t}{(T_a + T_l + T_t)} \quad (5)$$

Where R_s=Sustained disk Transfer Rate, T_a=Single Track Access time, T_l=Rotational Latency, T_t=Transfer Time.

IV. DISK SCHEDULING ALGORITHMS

- **FCFS**

The modest form of disk scheduling is First Come First Serve (FCFS) or First in First out (FIFO) [1]. Here tracks are processed in sequential order. This algorithm is quite simple, but it does provide fastest service. Consider for example, a disk queue when request for input and output operations to different blocks on cylinders in the following order as follows.

100, 180, 40, 120, 20, 130, 60, 70.

If the disk head is initially at cylinder 59, it will first move from 60 to 100, then to 180, 40, 120, 20, 130, 60 and finally to 70. This schedule is shown in following figure.

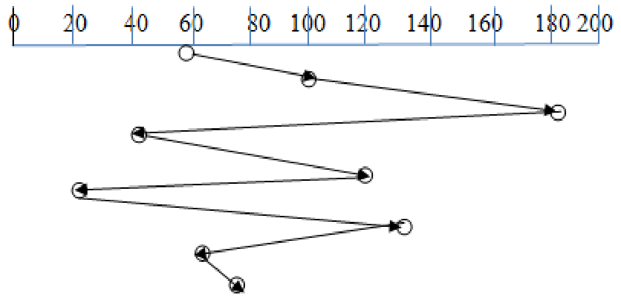

Fig 2: FCFS Disk Scheduling.

Total Head Movements will become 631 Cylinders. Assume that the seek time is 4 ms then Total Seek Time =631*4=2524 and Average seek time will become 315.5 milliseconds.

- **Shortest Seek Time First(SSTF)**

Next, we will look at an algorithm called Shortest Seek Time First or SSTF[1]. It is also sometimes called as shortest positioning time first or SPFT. This algorithm is working based on the idea that the read or write head moves towards a track which is vey nearer to it. This process will continue until all the requests in the disk queue are completed. Due to this the Read or Write head has least movement from current head position. However it would provide better performance than FIFO. Consider for example, a disk queue when request for input and output operations to different blocks on cylinders in the following order as follows.

100, 180, 40,120,20,130,60,70.

If the disk head is initially at cylinder 59, then it will first move towards 60, after that 70, 40, 20, 100, 120, 130 and finally 180.

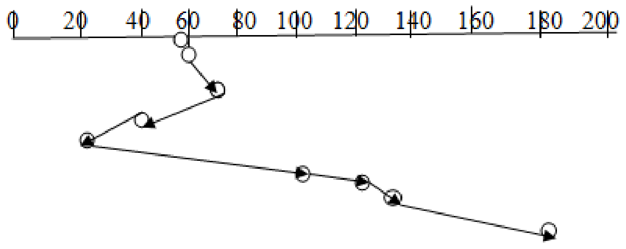

Fig 3: SSTF DISK Scheduling Algorithm

Total Head Movements will become 230 head movements. Assume that the seek time is 4 ms then total Seek Time 920 milliseconds and Average seek time will become 115 milliseconds.

- **SCAN**

Within the scan , the arm is required to move in one direction only by satisfying all the requests in that route until it reaches the last track in that direction or until there are no more requests in that direction[1]. The service direction is then reversed and scan the proceeds in the opposite direction, again picking up all the requests in order. The scan policy behaves identically with the SSTF policy. Indeed if we had assumed that the arm was moving in the direction of lower track numbers at the beginning of the example, then the scheduling pattern would have been identical for SSTF and SCAN. This algorithm has also called as "Elevator algorithm". Consider for example, a disk queue when request for input and output operations to different blocks on cylinders in the following order as follows.

100,180, 40,120,20,130,60,70.

If the disk head is initially at cylinder 59 and assuming that the head is moving towards increasing order of tracks. So the disk request 60 is get serviced first, then after 70,100,120,130,180 and disk head traverse towards end of the disk and get reverse and provide services to the disk requests 40 and 20. It is represented graphically as shown in figure.

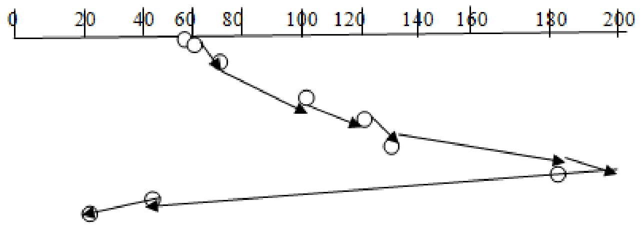

Fig 4: SCAN DISK Scheduling Algorithm

Total Head Movements will be 321and assume that the seek time is 4 ms then Total Seek Time is 1284 and Average seek time will become 165.5 milliseconds.

- **LOOK**

This algorithm is similar to SCAN Disk scheduling Algorithm. But At each side of the disk it touches the end point. And very similar to SCAN, It provides services in only one direction at a time. After get reverse by touching end of the disk it provide services to the reaming disk requests. Consider for example, a disk queue when request for input and output operations to different blocks on cylinders in the following order as 100,180, 40,120,20,130,60,70.

Let us assume that, the head move in the direction of higher numbered tracks and it is initially at 59. Then the disk request 60 is get serviced first, after that 70,100,120,130 and 180. After servicing the disk request 180, there are no more requests in that direction to serve. Then head is reversed towards other end of the disk and provide services to the disk requests 40 and 20. It is graphically represented as follows.

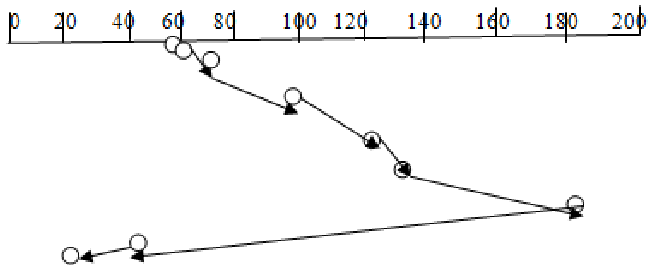

Fig 5: LOOK DISK Scheduling Algorithm

For this algorithm the total Head Movements will become281 and Assume that the seek time is 4 ms then Total Seek Time is 1124 and Average seek time will become 140.5 milliseconds.

- **C-SCAN**

The C-SCAN is an acronym for “Circular SCAN”. This algorithm restricts providing services to one direction only. Thus last track has been visited in one direction, the arm is returned to the opposite end of the disk and scan begins again. This reduces the maximum delay experienced by the new requests [3]. Consider for example, a disk queue when request for input and output operations to different blocks on cylinders in the following order as follows.

100,180, 40,120,20,130,60,70.

Let us assume that, the head move in the direction of higher numbered tracks and it is initially at 59. Then the disk request 60 is get serviced first, after that 70,100,120,130 and 180. After provide the service to the disk request there are no more requests in that direction to serve. But the head is moved towards end of the disk and get reverse and provide services to the disk requests 40 and 20. It is graphically represented as follows.

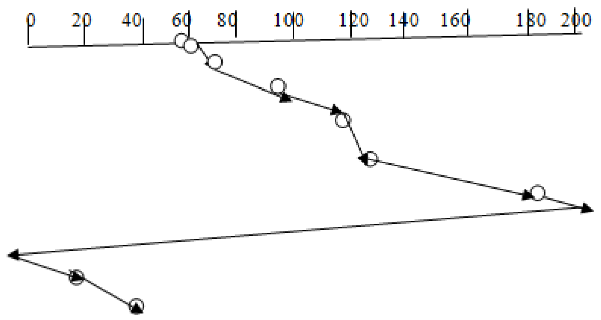

Fig 6: C-SCAN DISK Scheduling Algorithm

Total Head Movements will be 381 and assume that the seek time is 4 ms then Total Seek Time is 1524 and Average seek time will become 190.5 milliseconds.

- **C-LOOK**

SCAN And C-SCAN algorithms move the disk arm across the full width of the disk. But In practice neither of these algorithms is implemented this way. In C-LOOK disk scheduling algorithm the arm goes only as far as final request in each direction. Then it reverses the direction immediately without going all the way to the end of the disk [3]. Consider for example, a disk queue when request for input and output operations to different blocks on cylinders in the following order as follows.

100,180, 40,120,20,130,60,70

Let us assume that, the head move in the direction of higher numbered tracks and it is initially at 59. Then the disk request 60 is get serviced first, after that 70,100,120,130 and 180. After provide the service to the disk request there are no more requests in that direction to serve. Then the head is immediately reversed and move towards end of the disk and get and provide services to the disk requests 20 and 40. It is graphically represented as follows.

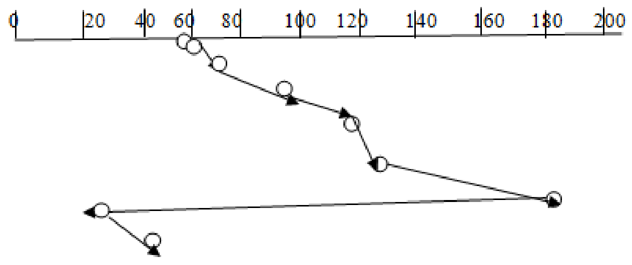

Fig 7: C-LOOK DISK Scheduling Algorithm

Total Head Movements for C-SCAN is 301 and assume that the seek time is 4 ms then Total Seek Time is 1204 and Average seek time will become 150.5 milliseconds.

V. Comparative Analysis

Suppose we have a 10000 RPM disk has 8 heads and 480 cylinders. It is divided into 120- cylinder zones with the cylinders in different zones containing 200, 240, 280, and 320 sectors. Assume each sector contains 4K bytes and a seek time between adjacent cylinders of 2 msec. So average Rotational latency for this disk is 8.3 milli seconds [5].

Algorithm	TMH	Avg Seek Time	IOPS	Throughput
FCFS	631	315.5	3.08	12.32
SSTF	230	115	8.11	32.44
SCAN	321	165.5	5.75	23
LOOK	281	140.5	6.72	26.88
C-SCAN	381	190.5	5.03	20.12
C-LOOK	301	150.5	6.29	25.16

Table 2: Comparison of throughput for different Disk Scheduling algorithms.

The performance analysis of the Disk Scheduling algorithms is represented graphically as follows.

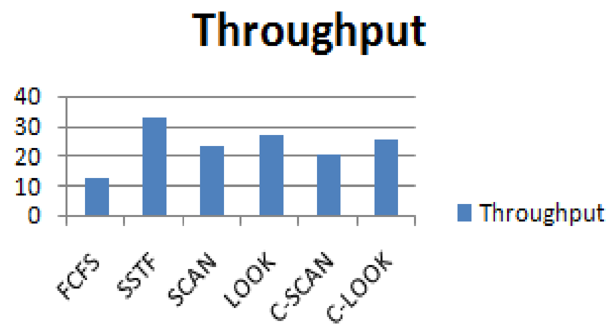

Fig 8: Graphical Representation of Comparative Analysis.

VI. CONCLUSION

In this paper we discuss about various disk scheduling algorithms and their principles. In addition to these, disk performance related measurements are also discussed here. And we can evaluate all disk scheduling algorithm based on these performance factors. From the comparative analysis, it is determined that Smallest Seek Time First(SSTF)Disk scheduling algorithm produce better throughput among from remaining disk scheduling algorithms.

REFERENCES

[1]. William Stallings, “Operating Systems Internals and Design Principles”, Pearson Education, Sixth (6th) Edition , ISBN: 97881317 2528-3.

[2]. Elmasri, Carrick, “Operating System –A Spiral Approach”, Tata McGraw-Hill Education, First Edition, ISBN NO: 9780071070942, Page No: 302-314.

[3]. Silberchatz, Peter B.Galvin, Greg Gange,”Operating systems Principles”, Willey Edition, Eighth(8th) Edition, ISBN:978812650962-1, Page No:440-444.

[4] C.Mallikarjuna, P.Chitti Babu,” Priority Based Disk Scheduling Algorithm”, International Journal of Innovative Research in Science, Engineering and Technology, Vol.3, Issue 9, Page No:15954 -15959, Sept2014.

[5]. Andrew S Tanenbaum, “Modern Operating System”, Tata McGraw-Hill Education, Second(2nd) edition, ISBN:97880074635513,Page No:318-321.

[6]. Charles Crowley,” Operating System-A Design Oriented Approach”, Tata McGraw-Hill Education, Fifth (5Th) Edition, ISBN: 978074635513, Page No: 622-634.

Author’s Profile

C. Mallikarjuna[1] holds two master degrees, first one in Computer Applications, and the other in Computer Science Engineering, currently working as Assistant Professor in the Department of M.C.A, Annamacharya P.G college of Computer Studies, Rajampeta, Kadapa Dist., A.P.

Dr. P. Chitti Babu is currently working as Professor & Principal, Annamacharya PG College of Computer Studies, Rajampet, AndhraPradesh. He received his Master's Degree in Computer Applications (MCA) from SV University, Tirupathi. He has completed M.Phil in Computer Science from Alagappa University and M.Tech in Computer Science & Engineering from Acharya NagarjunaUniveristy, Guntur, India. He has successfully received his Ph.D (Computer Science) from S.V. University. He has a total of 15 years ofexperience. He has published 25 papers in International and National Journals. He is a life member of CSI, IACSIT, IAENG and ISTE.Presently he is guiding 2 students for research under Ph.D Programme