Analysis of Traditional Hard Disk Scheduling Algorithms: A Review

Mustafa Ali Bamboat¹, Mansoor Ahmed Khuhro^{1,*}, Kamlesh Kumar², Imtiaz Ali Halepoto³, Naadiya Mirbahar^{1,4}, Ghulam Sarfaraz Khan¹

¹Department of Computer Science, Sindh Madressatul Islam University, Karachi.

²Department of Software Engineering, Sindh Madressatul Islam University, Karachi.

³Department of Software Engineering, QUEST Nawabshah, Sindh Pakistan.

⁴Research Lab of Information Security and AI, Benazir Bhutto Shaheed University, Karachi, Pakistan.

*Corresponding author: makhuhro@smiu.edu.pk

Abstract

This review paper compares the various disk scheduling algorithms used to schedule processes in a queue, such as FCFS, SSTF, SCAN, C-SCAN, LOOK, C-LOOK, OTHDSA and Zone Base Disk Scheduling. It then applies all of these techniques using two data sets of I/O requests to assess the performance of each algorithm. The comparison includes total seek time to locate the requested I/O, the overall throughput of jobs done per unit, and the fairness of each request to be serviced. The comparison results also supported the statement that all disk scheduling techniques offer a good performance. The comparison shows that among all disk scheduling algorithms, OTHDSA has excellent performance and takes 175 seek times and 139 seek times to complete all I/O requests of both example lists, respectively.

Keywords—Disk Scheduling, Modification Over C-LOOK, OTHDSA, Zone-Based, FCFS, SSTF, SCAN, C-SCAN, C-LOOK.

1 Introduction

HARD disk is a non-volatile storage device that retains power-free data [1]. It stores data that consists of tracks, sectors, and clusters on circular plates. A disk head is connected to the disk arm for reading/writing data on the tracks. The disk head is taken to travel back and forth to read/write, known as “seek time” [2-3] on the desired bit storage index. Disk scheduling algorithms can effectively perform such tasks. The various disk scheduling algorithms implemented on traditional mechanical hard disks comprise the modified versions, and new algorithms proposed by researchers were compared and reviewed. The two illustration lists of I/O requests [11-12] are used to reflect each disk scheduling algorithm’s implementation and performance.

The solid state drives are the successor of traditional hard disk technology and are made up of integrated circuit assemblies that save data permanently. They were first introduced to the market in 1991.

Storage Technology Corporation floated the original idea in 1978, which Oracle Cooperation now owns. These drives are more stable and faster than magnetic drives due to the lack of a physical spinning disc and movable read-write mechanical arms and the storage made in the semiconductor cells containing between 1 bit to 4 bits data [13]. Furthermore, since the SSD has no moving parts, it uses the FCFS policy to serve I/O requests. As a result, the output is unaffected by seeking time and rotational latency. Therefore, the main goal of this paper is to examine disc scheduling algorithms on conventional drives (Magnetic) which is a cost-effective alternative for small and medium-sized businesses and consumers searching for a less expensive option; whereas, SSDs are a much more expensive solution for small and medium-sized businesses.

An overview is given in section I. Literature review is in section II, accompanied by an explanation of current disk scheduling algorithms such as FCFS, SSTF, SCAN, C-SCAN, LOOK, and C-LOOK [2-3] along with their pros and cons in Section III. Section IV focuses on the recent algorithms such as Updated C-LOOK, OTHDSA, ZONE Focused [4-7]. Finally, the results of all algorithms are discussed in section V.

ISSN: 2523-0379 (Online), ISSN: 1605-8607 (Print)

DOI: <https://doi.org/10.52584/QRJ.1901.07>

This is an open access article published by Quaid-e-Awam University of Engineering Science & Technology, Nawabshah, Pakistan under CC BY 4.0 International License.

Section VI concludes the paper.

2 Literature Review

The authors of [22] examine the performance of disk scheduling algorithms such as FCFS, SSTF, Scan, Look, C-Scan, and C-Look and the characteristics of each technique. The comparison was conducted by fetching data against the input list of I/O requests (work queue) and calculate the Total Head Movement (THM) that occurred to serve the input list. In [22], the analysis result reflected that the FCFS technique prevents starvation but fails to serve faster. The SSTF technique reduced the THM time compared to FCFS but led I/O requests to starvation. The SCAN technique shows much better results in contrast to SSTF. In contrast, C-SCAN is slightly better performed as it provides a uniform waiting time than SCAN. LOOK technique moves head until the end of the requested I/O and immediately moves in reserve direction without going to the end of the track as SCAN does. C-LOOK differs with C-SCAN in that it does not move head till the end of the track.

In [23], the authors introduced a technique named New-Heuristic technique to decrease the disk head movements, producing a high throughput rate. The requested I/O block is arranged in non-decreasing order, then moves the disk head from the smaller I/O block to the higher block. A queue of ten I/O request blocks is used to compare the results of a new approach with FCFS, SSTF, SCAN, C-SCAN, LOOK, and C-LOOK techniques. The comparison results show that the New-Heuristic approach takes 236 head movements to serve all I/O requests; whereas, other techniques take FCFS(582), C-SCAN(388), SSTF(256), C-LOOK(346), LOOK (298), and SCAN(318) total head movements respectively.

In [24], the authors proposed a modified version of the SSTF technique, named MHSF (Major Half Served First), claiming that it is faster than SSTF. First, the requested I/O blocks are distributed into two zones: lower and upper from the current head position. Then, the algorithm checks the number of I/O requests in both zones and selects the zone having more I/O requests than the other for serving. In contrast, if both zones have an equal number of I/O requests, then the algorithm checks the current position of the disk head and starts serving I/O requests where the current disk head was lying either in the lower zone or upper zone. The analysis shows that MHSF outperforms SSTF and FCFS in total head movements average time and fetching efficiency in a real-time system.

List	I/O Request	Disk head position
List-1 (LS-1)	{95,180,34,119,11,123,62,64}	50
List-2 (LS-2)	{55,58,39,18,90,160,150,38,184}	100

TABLE 1: Lists of I/O requests

The authors of [25] evaluated the performance of the disk cache and parallelism I/O scheduling technique. The experiment was conducted on the RAID consisting of five disks of different vendors having 16MB disk cache for pre-fetching two types of workload used in the experiment: production and HPC. The authors collected a trace log of seven days of each workload for analysis. An experiment shows that the traditional DCAP model has a disadvantage of I/O request starvation; whereas, the proposed DCAP-G reduced the rate of starvation.

3 Disk Scheduling Algorithms

This study compares all disk scheduling algorithms described in Section I using lists of I/O requests, taken from [11-12], as shown in Table-1. These I/O requests read a block from 0 to 199 and calculate the seek-time of a disk head. There are various simulation softwares available to calculate the seek-time of a real example list of I/O requests, such as Nicomedes [19], Disk Scheduler Simulator developed in Java [20], and C-Sharp [21]. This work focuses on how these algorithms work. Two simple/basic lists of I/O requests are used.

3.1 First Come First Serve (FCFS)

First Come First Serve, as the names imply, schedules tasks based on their arrival order, regardless of the processing time needed to run a task or any other parameters. This scheme is very popular for scheduling transmissions [18]. FCFS provides an equal opportunity for all the indices requested but does not provide an optimum solution in terms of search time and does not provide the best scheduling services [8]. FCFS operates sequentially, meaning all I/O requests will be served as provided in the example list. In accordance with the LS-1 requests, the head location is initially on track number 50 and the first I/O requested in the set is 95. The head moves from 50 to 95, while 180 is the second requested track in a set, such that the head moves from 95 to 180, and this process continues until all the I/O requests are served as stated in LS-1. During the disk head movement, the total number of tracks crossing a path covered from one position to reach another location is the seek time of the requested I/O. The total seek time is the aggregate of all seeks

Lists	Total Seek Time
LS-1	$(50-95)+(95-180)+(180-34)+(34-119)$ $+(119-11)+(11-123)+(123-62)+(62-64)=644$
LS-2	$(100-55)+(55-58)+(58-39)+(39-18)$ $+(18-90)+(90-160)+(160-150)+$ $(150-38)+(38-184)=498$

TABLE 2: FCFS — total seek time

List	Head Movement	Total Seek Time
LS-1	50, 62, 64, 34, 11, 95, 119, 123, 180	236
LS-2	100, 90, 58, 55, 39, 38, 18, 150, 160, 184	258

TABLE 3: SSTF - total seek time

times occurred while serving all requests defined in LS-1. By using the FCFS algorithm on LS-1, the total seek time is 644. In LS-2, the head's initial position is on track number 100, and the first I/O request in a set is 55. The head goes from 100 to 55, then according to the subsequent request in a queue, the head shifts from 55 to 58 and keeps going till all requests are served as sequentially appeared in LS-2. The total seek time FCFS taken on LS-2 is 498. Calculation of total seek time is shown in Table 2.

3.2 Shortest Seek Time First (SSTF)

First, this algorithm arranges all I/O requests in increasing order, calculates the disk head time seek for each I/O request, and selects a job with the shortest seek-time. The SSTF decreases the average response time, which increases the throughput rate [8]. The drawback of the SSTF is that the constant collection of the indices of least search time contributes to the hunger of those who require more seek time [9]. LS-1 begins from track number 50, then moves to 62 because of the shortest seek time of 12 tracks of disk head travel. Subsequently, it moves from 62 to 64 as only two tracks of disk head travel to reach 64. It then continues to select the shortest search time until all requests have been served. A similar procedure implies on LS-2. Table 3 shows the movement order of I/O requests, given in the example lists and the total seek time of each list.

3.3 SCAN (Elevator Algorithm)

An elevator in a building with the restriction that it moves upwards will not move downwards until it reaches the building's end floor and vice versa. SCAN algorithm serves the requested indices in the same way that it serves all the requests coming in the direction until it reaches the end track of the same direction.

List	Disk Head Direction	Total Seek Time
LS-1	Left Direction (Right to Left)	$(50-34)+(34-11)+(11-0)$ $+(0-62)+(62-64)+(64-95)$ $+(95-119)+(119-123)$ $+(123-180)=230$
LS-2	Right Direction (Left to Right)	$(100-150)+(150-160)$ $+(160-184)+(184-199)$ $+(199-90)+(90-58)$ $+(58-55)+(55-39)$ $+(39-38)+(38-18)=280$

TABLE 4: SCAN - total seek time

Then, the disk head will move in the opposite direction. SCAN gives the moderate throughput and low variance of response time, but the long waiting time for the new coming requests of the locations just visited by disk arm [8]. For example, track number 50 is requested, and the head is on track number 80, which is moving towards 0 track, and then such request (50) will be served without any delay. Whereas, another request of track number 150 needs to wait for the head to complete the trip until the end track zero (0) is reached. Then, re-trip in the reverse direction takes place to serve those requests. Such scenario may lead to a long waiting time [9]. Suppose the disk is head moving in the left direction (from 199 to 0) and LS-1 requests are needed to be served. The initial disk head is on 50. The algorithm sorts the I/O requests in ascending order, then starts serving all the requests between 0 and 50, as it is moving in the left direction. After that, disk head re-trips in the reverse direction to serve the rest of the requests in the direction of the movement. The drawback of the SCAN algorithm is that the disk head moves till the end of each direction whether the track number is requested or not. Table 4 shows the total seek time of both lists of I/O requests using the SCAN algorithm.

3.4 C-SCAN (Circular Elevator Algorithm)

Imagine the same elevator scenario described in the SCAN scheduling algorithm, but this time such an elevator only serves in one direction. For example, if it is moving up, then it serves all requests coming in that way till the top end of the building. From there, it will come back to the bottom end without serving any request. Then, from the bottom, it will again move up and will serve the rest of the requests in this direction. C-SCAN gives uniform waiting time as compared to the SCAN and provides a better response time. However, it forces the disk head to move in the reverse direction without serving any request and starts serving requests from only one end which causes

List	Disk Head Direction	Total Seek Time
LS-1	Left Direction (Right to Left)	$(50-34)+(34-11)+(11-0)$ $+(0-62)+(62-64)+(64-95)$ $+(95-119)+(119-123)$ $+(123-180)=230$
LS-2	Right Direction (Left to Right)	$(100-150)+(150-160)$ $+(160-184)+(184-199)$ $+(199-90)+(90-58)$ $+(58-55)+(55-39)$ $+(39-38)+(38-18)=280$

TABLE 5: C-SCAN - total seek time

more seek movements than SCAN [9]. In LS-1, we apply the C-SCAN algorithm. The disk head goes in the left direction, the initial position of the disk head is 50, and LS-1's I/O requests are arranged in ascending order. As explained earlier, C-SCAN served all requests coming in the direction of head movement, such as 34 and 11 will be served. As track number zero hits, the disk head moves towards the last track of opposite direction without serving, and then start serving again in the left direction.

Similarly, for LS-2, the disk head goes in the right direction, where the disk head's initial location is 100. C-SCAN begins to serve the requests in the right direction such as 150, 160, and 184. The disk head travels in the opposite direction without serving any requests. It reaches tracking number 199 and begins to serve the rest of the I/O requests as it reaches track zero in the right direction. The downside of C-SCAN is that as the disk head travels till the end of the moving path, it includes additional search time such as in LS-1 disk head moving in the left direction and the I/O request queue ends on track number 11. However, as the disk head moves from 11 to 0 according to the algorithm, then re-trips in the reverse direction. Such search time is considered to be an overhead. Similarly, in LS-2 overhead, search time is from 184 to 199 before re-trip in the opposite direction. Such movement produces a circular effect, but the downside of overhead is to sum additional seek time. Table 5 shows the total seek time of C-SCAN on both lists of I/O requests.

3.5 LOOK

LOOK is an advanced version of the SCAN algorithm with a slightly better seek time in which head disk does not move beyond the requested I/O. This algorithm reduces starvation, waiting time, and provides low diversity in response time [8]. There is an overhead of finding the end requests because it looks for new requests before moving in a given direction [9]. We

List	Disk Head Direction	Total Seek Time
LS-1	Left Direction (Right to Left)	$(50-34)+(34-11)+(11-62)$ $+(62-64)+(64-95)+(95-119)$ $+(119-123)+(123-180)=208$
LS-2	Right Direction (Left to Right)	$(100-150)+(150-160)$ $+(160-184)+(184-90)$ $+(90-58)+(58-55)$ $+(55-39)+(39-38)+(38-18)=250$

TABLE 6: LOOK - total seek time

List	Disk Head Direction	Total Seek Time
LS-1	Left Direction (Right to Left)	$(50-34)+(34-11)+(11-180)$ $+(180-123)+(123-119)$ $+(119-95)+(95-64)=326$
LS-2	Right Direction (Left to Right)	$(100-150)+(150-160)$ $+(160-184)+(184-18)$ $+(18-38)+(38-39)+(39-55)$ $+(55-58)+(58-90)=322$

TABLE 7: C-LOOK - total seek time

found the seeks times of this algorithm to be 208 and 250 on LS-1 and LS-2, respectively. Table 6 shows the total seek time of both illustration lists using the LOOK algorithm.

3.6 C-LOOK

C-LOOK is an enhanced version of the C-SCAN algorithm with a slightly better seek time. CLOOK algorithm applies the same procedure as the C-SCAN that does not serve while coming back from one end. However, C-LOOK stops till the queue's end request and then starts serving a request from that point moving in the reverse direction. We found the seeks times of this algorithm to be 326 and 322 on LS-1 and LS-2, respectively. Table 7 shows the total seek time of both illustration lists using the C-LOOK algorithm.

4 Modified C-LOOK, Optimized Two Head Disk Scheduling (OTHDSA), and Zone-Based Disk Scheduling Algorithm

4.1 Modified C-Look Algorithm

It is a slightly modified version of C-LOOK which does not serve any request while coming back in the reverse direction. This behavior of the C-LOOK algorithm increases the seek time because the disk head moves till the end of the request of the queue without serving requests coming in its direction. Modified C-LOOK serves requests while coming back in the reverse direction. This technique reduces the seek time as compared to the C-LOOK algorithm [5]. Table 8 shows the total seek time of both lists. We observed that on

List	Disk Head Direction	Total Seek Time
LS-1	Left Direction (Right to Left)	$(50-34) + (34-11) + (11-62) + (62-64) + (64-95) + (95-119) + (119-123) = 208$
LS-2	Right Direction (Left to Right)	$(100-150) + (150-160) + (160-184) + (184-90) + (90-58) + (58-55) + (55-39) + (39-38) + (38-18) = 250$

TABLE 8: Modified C-LOOK - total seek time

LS-1, the disk head serves requests while coming back in other direction from track 11 till the end request of queue, i.e., 180. Whereas, on LS-2, the disk head serves requests while coming back in other direction from track 184 till the end request of queue, i.e., 18.

4.2 Optimized Two Head Disk Scheduling Algorithm (OTHDSA)

A new disk model MHPR (Multi-head Parallelism and Redundancy), represented by Yang [14], reduced the access time of a single I/O request and increased the response time. In [6], the authors inspired by [14], proposed optimized disk scheduling based on two heads disk. As the name suggests, the first disk head is at the position of track number zero (0), named as “HL,” which moves from left to right. The second head is at track number 199, named “HR,” which moves from right to left with the restriction that only one (either HL or HR) can be moved at a time. The pseudocode of OTHDSA is given as follows.

- 1) Sort requests of a queue in non-decreasing order.
- 2) Find the minimum and maximum value from the sorted requested queue.
- 3) Make a decision which disk head (either HL or HR) should first start serving the requests of the queue. For that, calculate the seek time from both points as shown in Equation 1-2.

$$HL \text{ Seek Time} = \min \text{ value requested} - 0 \quad (1)$$

$$HR \text{ Seek Time} = 199 - \max \text{ value requested} \quad (2)$$

- 4) Pick a disk head (either HL or HR) to start serving which has minimum seek time by comparing the result of the Equation 1-2.
- 5) If the disk head HL has chosen to make the first move, then it will serve all the requests which come in this direction from left to right. However, only those requests will be served which are less than or equal to the track number 100. Whereas, if the HR has chosen the first head, it will reserve all

Disk Head	Initial Track Position	Request Served	Total Seek Time
HL	0	{11,34,62,64,95}	$(0-11) + (11-34) + (34-62) + (62-64) + (64-95) = 95$
HR	199	{180, 123, 119}	$(199-180) + (180-123) + (123-119) = 80$

TABLE 9: Seek Time of each Head on LS-1

requests that are greater than the track number 100, as described by Equation 3-4.

$$HL \text{ serves request} \leq 100 \quad (3)$$

$$HR \text{ serves request} > 100 \quad (4)$$

- 6) When the disk head is either HL or HR reaches 100, it has two options: whether to keep moving the same disk head to complete the rest of the requests or to stop and move the second disk head in the respected direction. To decide, we need to calculate which disk head would travel a lesser distance to complete it. If the moving disk head takes a lesser traveling distance, it will keep on moving. Otherwise, it will stop, and the second starts moving.

The implementation of the OTHDSA technique has been explained to calculate the total seek time. We assume that lists of I/O requests are sorted in ascending order, and initially, HL and HR are on track zero (0) and 199, respectively.

4.2.1 LS-1 Total Seek Time using OTHDSA

We decide which disk head may take the first move. HL seek time is 11, and HR seek time is 19, according to Equation 1-2. HL will take the first move as it has a smaller seek time compared to HR, and serve all the requests coming in the way 11,34,62,64,95 till it reaches equal or less than track number 100 and make a decision to either keep on moving to serve all requests or to give a chance to the second head HR to start serving all the requests. This decision is based on the following calculations.

$$HL \text{ Travel distance till end request} = 95 - 180 = 85$$

$$HR \text{ Travel distance till end request} = 199 - 119 = 80$$

We observed that HR travel distance is lesser than HL travel distance; therefore, HL stops, and HR starts serving the remaining request from right to left such as 180, 123, 119. Table 9 shows HL seek time and HR seek time. The Total Seek Time of LS-1 would be the sum

Disk Head	Initial track position	Request Served	Total Seek Time
HL	0	{18,38,39,55,58,90}	$(0-18)+(18-38)$ $+(38-39)+(39-55)$ $+(55-58)+(58-90)$ $= 90$
HR	199	{184,160,150}	$(199-184)$ $+(184-160)$ $+(160-150) = 49$

of HL seek time and HR seek time as shown below:

$$TotalSeekTime = HLSeektime + HRSeektime$$

$$TotalSeekTime = 95 + 80 = 175$$

4.2.2 LS-2 Total Seek Time Using OTHDSA

Similar to the above steps, we decide which disk head may take the first move. HL seek time is 18, and HR seek time is 15, according to Equation 1-2. HR will take the first move as it has a smaller seek time compared to HL, and serve all the requests coming in the way (184,160,150) till it reaches track number 100 and makes a decision to either keep on moving to serve all requests or to give a chance to the second head HL to start serving all the requests. This decision is based on the following calculations.

$$HL \text{ Travel distance till end request} = 90 - 0 = 90$$

$$HR \text{ Travel distance till end request} = 150 - 18 = 132$$

We observed that HL travel distance is smaller than HR travel distance; therefore, HR will stop, and HL start serving the remaining request from left to right such as 18,38,39,55,58,90. Table 10 shows HL seek time and HR seek time. The Total Seek Time of LS-2 would be the sum of HL seek time and HR seek time as shown below:

$$Total \text{ Seek Time} = HL \text{ Seek Time} + HR \text{ Seek Time}$$

$$Total \text{ Seek Time} = 90 + 45 = 139$$

We found that Optimize Two Head Disk Scheduling Algorithm (OTHDSA) has shown superior performance compared to the other disk scheduling algorithms.

4.3 Zone-Based Disk Scheduling Algorithm

In this scheduling algorithm, the requested I/O queue consists of a lower zone and an upper zone. The lower zone contains all requests that are less than or equal to the initial head disk position. The upper zone consists of all higher requests than the initial head disk position. The distribution into zones is based on the LCFS (Last

Come First Serve) requests. The following steps are performed.

- 1) Divide the requested queue into two parts according to the initial head disk position in such order that all requests are served on LCFS (Last Come First Serve) basis in the requested queue. If there are requests between the LCFS requests, they are served first before the last arrived requests.
- 2) Make a decision which zone will be selected to be served first. Compare the length of the requests of both zones and start from the zone which has smaller number of requests. However, if both zones have an equal number of requests, then pick that zone which has smaller seek time of initial request from current disk head position.

The implementation of the zone-based technique has been explained to calculate the total seek time.

4.3.1 LS-1 Total Seek Time using Zone-Based Algorithm

The request queue is divided into two parts. All requests less than or equal to the current head position 50 are considered to be in lower zone, and those higher than the current head position are considered to be in upper zone. Both zones contain requests on LCFS basis. A decision on which zone disk arm starts serving is made such that the disk arm serves from lower zone, and after completing the lower zone, the disk arm starts to serve upper zone. Therefore, we first start serving from lower zone. Track number 11 of lower zone will be served first according to the LCFS order rule, which will be served first if there are no requests between the current head position (50) and track number (11). If so, the disk arm will serve them first before moving on to track number (11). After completing lower zone, the disk arm starts serving a request from upper zone. Currently, the disk head position is on track number 11 in the Requested Queue (RQ). Track number is 64, and it has come first according to LCFS order which will be served first. But track number 62 is in between the current head position and track number 64, therefore, disk head will serve 62 before 64. Now that the current head position is on track 64, the algorithm again checks whether the request from upper zone 123 is in order of LCFS in RQ. But track number 95,119 is in between current head position 64 to 123. Therefore, it will serve all requests between 64 and 123. After that, only one request left in upper zone is 180, therefore, the disk head will move from 123 to 180 to complete the request queue.

List	Total Seek Time
LS-1	$(50-34) + (34-11) + (11-62) + (62-64) + (64-95) + (95-119) + (119-123) + (123-180) = 208$
LS-2	$(100-150) + (150-160) + (160-184) + (184-90) + (90-58) + (58-55) + (55-39) + (39-38) + (38-18) = 250$

TABLE 10: Zone based algorithm— total seek time

4.3.2 LS-2 Total Seek Time using Zone-Based Algorithm

The request queue is divided into two parts. All requests less than or equal to the current head position 100 are considered to be in lower zone, and those higher than the current head position are considered to be in upper zone. Both zones will contain requests on LCFS basis. It is decided which zone disk arm starts serving such that the disk arm starts serving from upper zone. After completing the upper zone, the disk arm starts serving lower zone, therefore, upper zone is served first. Track number 184 of upper zone arrives first according to LCFS order which will be served first. But track number 150,160 is between the current head position and LCFS request 184, therefore, the disk head serves 150 and 160 before 184. After completing upper zone, the disk arm starts serving a request from lower zone. Currently, the disk head position is on track number 184. Now, RQ track number 38 arrives first according to the LCFS order which will be served first. But track number 90,58,55,39 are in between the current head position and track number 38, therefore, disk head will serve 90,58,55,39 before 38. After that, a single request left in lower zone is 18, therefore, the disk head will move from 38 to 18. Table 11 shows the total seek time of LS-1 and LS-2 using zone-based technique.

5 Results and Analysis of all Disk Scheduling Techniques

This paper discussed different disk scheduling algorithms and compared their performances on two lists of I/O requests to determine each algorithm's time to execute the queue processes. Disk head scheduling takes into account two key concerns: throughput and fairness. Throughput refers to the number of jobs completed per unit time. Whereas, fairness refers to each job being treated equally and is ultimately served. An effective scheduling algorithm seeks a balance between these two issues. The best algorithm has the highest level of fairness and throughput. Table 12 shows the comparison between total seek time of all disk scheduling techniques and their overall throughput and fairness for each track in the requested queue.

Fig. 1: Comparison of total seek time

Table 12 shows the comparison of total seek time of all disk scheduling techniques along with their overall throughput and fairness for each track in the request queue. The measures set for throughput and fairness include better, good, bad, moderate, and poor, as shown in Table 12. Figure-1 shows that the Optimize Two Head Disk Scheduling Algorithm (OTHDSA) has outstanding performance among all Disk Scheduling Techniques.

6 Conclusion

FCFS shows good fairness among the requested track turns, however, its throughput is less than other algorithms. The throughput of SSTF and SCAN is moderate, but does not have good fairness in serving the requested tracks. Circular SCAN (C-SCAN) performs better in both aspects. LOOK and C-LOOK provide much better throughput than SCAN and C-SCAN and are also better in terms of fairness.

The Optimized Two Head Disk Scheduling Algorithm (OTHDSA) shows promising performance in terms of both aspects (throughput and fairness). Modified over C-Look and Zone Based Disk Scheduling have same performance in terms of total seek time. In 2017, Seagate Technology announced that they are working on Multi-Actuator Hard Disk Drives, which will help to improve the data rate of transfer performance [15]. According to Seagate's official blog site [16], MACH.2 drives will be available shortly. In another blog [17], Seagate competitor Western Digital introduces a 16TB drive with eight platters and sixteen heads instead of Seagate's 16TB drive with nine platters and eighteen heads. Both companies have not disclosed the prices of such disk drives yet, but it is assumed that fewer platters and heads would result in lower costs.

Disk Scheduling Algorithm	Total Seek Time		Throughput	Fairness in terms of I/O request severed
	I/O Request List-1	I/O Request List-2		
FCFS	644	498	Bad	Good
SSTF	236	258	Moderate	Poor
SCAN	230	280	Moderate	Moderate
C-SCAN	386	388	Good	Good
LOOK	208	250	Better than SCAN	Moderate
C-LOOK	326	322	Better than C-SCAN	Good
Modified C-LOOK	208	250	Better than C-LOOK	Good
OTHDSA	175	139	Better than all	Best
ZONE-BASED	208	250	Same as Modified C-Look	Good

TABLE 11: Result and analysis

References

- [1] Hong, Seungbum, Orlando Auciello, and Dirk Wouters, eds. Emerging non-volatile memories. Springer, 2014.
- [2] Silberschatz, A., Peter Baer Galvin and Gagne, G, “Operating system concepts essentials”, John Wiley & Sons, Cop. 2014.
- [3] Geist, R. and Daniel, S. “A continuum of disk scheduling algorithms.” ACM Transactions on Computer Systems, vol. 5, no. 1, pp. 77–92, 1987.
- [4] Naik, S., and Lalchandani, J. T. “An Optimal Disk Scheduling Technique.” In 3rd International Conference for Convergence in Technology (I2CT), pp.1-4, 2018.
- [5] Bhuguna, R. “Modification over C-Look Disk scheduling.” In IEEE 2nd International Conference on Computing for Sustainable Global Development (INDIACom), pp. 83-85, 2015.
- [6] Singh, K., Nidhi, Rastogi, D., and Singh, D. “Optimized Two Head Disk Scheduling Algorithm.” In Fifth International Conference on Advanced Computing & Communication Technologies, pp. 234-240, 2015.
- [7] Pandey, P., Joshi, G., and Gola, K. K. “A zone-based improved disk scheduling algorithm.” In International Conference on Advances in Computing, Communication, & Automation (ICACCA) (Spring), pp.1-5, 2016.
- [8] Pokharel, B.P. “A Comparative Analysis of Disk Scheduling Algorithms.” International Journal for Research in Applied Sciences and Biotechnology, 8(2), pp.184-195, 2021.
- [9] Stallings, W. “Operating systems: Internals and design principles” Boston, Mass.: Pearson Prentice Hall, 2015.
- [10] Kerwad, Z.A. “Scheduling Algorithms for CPU.” International Journal of Progressive Sciences and Technologies, vol. 27, no. 1, pp.458-463, 2021.
- [11] Available at: <https://www.ques10.com/p/35381/compare-the-following-disk-scheduling-algorithms-u/>. Accessed on Jan 2021.
- [12] Mohamed, M. “Performance Evaluation Movable Head Disk Schedulers.(Dept. E)” MEJ. Mansoura Engineering Journal, vol. 15, no. 2, pp.16-23, 2021.
- [13] Kasavajhala, V. “Solid state drive vs. hard disk drive price and performance study.” Proc. Dell Tech. White Paper, pp.8-9, 2011.
- [14] Hu, Y., Feng, D., Zhang, S. P., and Liu, J. N. “MHPR: Multi-head Parallelism and Redundancy Disk Model.” In IEEE International Conference on Networking, Architecture, and Storage, 2009.
- [15] Shah, P., and Horowitz, R. “Data-Driven Feedforward Control Design and Input Shaping Techniques for Multi Actuator Drives.” ASME Letters in Dynamic Systems and Control, vol. 1, no. 3, 2021.
- [16] Available at: <https://www.seagate.com/in/en/innovation/multi-actuator-hard-drives/>. Accessed on April 2021.
- [17] Mellor, C. “Double-headed Seagate disk drives? Yes, on their way.” Blocks and Files. 2019.
- [18] Halepoto, I.A. “Scheduling and flow control in CMT-SCTP.” HKU Theses Online (HKUTO), 2014.
- [19] Nicomedes “Disk Scheduling Online simulator.” Available at <https://nicomedes.assistedcoding.eu/#/dashboard>. Accessed on Jan 2021.
- [20] Mohamed, M. “Performance Evaluation Movable Head Disk Schedulers.” MEJ. Mansoura Engineering Journal, vol. 15, no. 2, pp.16-23, 2021.
- [21] Suranauwarat, S. “A Disk scheduling algorithm simulator.” Computers in Education Journal, vol. 8, no. 3, 2017.
- [22] Celis, J. R., Gonzales, D., Lagda, E., and Rutaquio Jr., L. “A Comprehensive Review for Disk Scheduling Algorithms”. IJCSI International Journal of Computer Science, vol. 11, no. 1, pp. 74, 2014.
- [23] Saha, S., Akhter, Md. N., and Kashem, M. A. “A New Heuristic Disk Scheduling Algorithm”. International journal of scientific & technology research, vol. 2, no. 1, pp. 49-53, 2013.
- [24] Mishra, M. K. “Major Half Served First (MHSF) Disk Scheduling Algorithm.” International Journal of Computer Applications & Information Technology, vol. 2, no. 1, 2013.
- [25] Prabhakar, R., Kandemir, M., and Jung, M. “Disk-Cache and Parallelism Aware I/O Scheduling to Improve Storage System Performance.” IEEE 27th International Symposium on Parallel and Distributed Processing, pp. 357-368, 2013.