Journal of Computer Applications and Research, Volume 1, No 1, 2020

Practical Performance Comparison of Disk Scheduling Algorithms

Swe Swe Myint
University of Computer Studies, Monywa
sweswemyint.cu@gmail.com

Khin Nyein Myint
University of Computer Studies, Monywa
khinnyeinmyint.knm@gmail.com

Sein Kyaw Lwin
University of Computer Studies, Monywa
seinkyawlwin94@gmail.com

Abstract

The intention of this paper is to understand the student how to work disk scheduling policies in operating system. Nowadays, modern technology and high performance computing processors are rapidly evolved. Moreover, the speed of processor and main memory are also increased rapidly but secondary memory is still relatively slow. Thus the performances of disk I/O are also vital and many researchers are interested in schemes for improving that performance. Disk scheduling is crucial because multiple I/O requests may be by different operations and only one I/O request can be run at a time by the disk controller. It is hoped that this paper may provide an opportunity for all computer students to learn theories and pratical experience of allocating I/O device for various processes. In this research, the C programming language is used for practical interface and calculates total head movement performance of FIFO, SSTF, SCAN, and C-SCAN in disk scheduling algorithms.

Keywords: Disk scheduling, FIFO, SSTF, SCAN, C-SCAN, Seek time, Head movement, Operating system.

1. Introduction

The actual details of disk input/output operation depend on the computer system, the operating system and the nature of the input/output channel and disk controller hardware. In any disk system with a moving read/write head, the seek time between cylinders takes a significant amount of time. So seek time is most important to get best access time [1].

The file system can be viewed logically in three different i.e. user, programmer interface to the file system and secondary storage structure. The lowest level of the file system is secondary storage structure and disk is the main secondary storage device that is generally divided into tracks, cylinders and sectors and stores the data permanently [2].

Scheduling is a basic task of an operating system to schedule all computer resources before their use. To serve the read or write requests of multiple processes, a disk should be efficient enough to give services to all processes and this can be achieved by various traditional disk scheduling algorithms. Traditional disk scheduling algorithms are First-In-First-Out (FIFO), Shortest Seek Time First (SSTF), SCAN, and Circular SCAN (C-SCAN). Track and sectors are basic conceptual parts based on which the disk scheduling algorithms serves all memory requests. Tracks are a set of concentric circular path on the surface of a disk or platter on which the data is retrieved or written on. On a track the data bits are stored in the form of magnetized bit pattern. Each track is further sub-divided into a number of partitions known as sectors. A sector stores a fixed amount of user accessible data [3].

In multiprogramming systems, processes running concurrently may generate for requests to read and write disk records. The operating system handles these I/O requests from the queue and processes them one by one [4].

It is the simplest algorithm without any overload. But it requires more seek time for other disk scheduling algorithm. In disk scheduling, the main performance measures are seek time, rotational latency, data transfer time and access time [5].

The practical algorithm calculates the difference between the highest request value and the lowest request value and then compares it with the current head position. The main objectives for any disk scheduling algorithm are minimizing the average seek length [6].

This paper is organized with the following. Section 1 is the introduction, section 2 is related works. Section 3 is theoretical background, section 4 presents the experimental results, section 5 is the conclusion of the system and section 6 presents the future extension.

2. Related Works

The authors have proposed all computer resources need to be scheduled before use for faster utilization of them. In multi-processing environment many processes are generated which in turn generate many read or write request to store or retrieve data on a secondary storage device. But hard-disk can serve one memory request at a time. So all memory requests are stored in a queue. Two consecutive memory request may be far from each other based on tracks, which can result in greater disk arm movement. Hard drives are one of the slowest parts of computer system and thus need to be accessed in an efficient manner to match the speed of requests. To manage disk scheduling algorithm of operating system. The main goal of a disk scheduling algorithm is to reduce the starvation problem among the requests, to minimize the total amount of head movement and to minimize the average seek length. The performance of a hard disk can be enhanced by using various scheduling algorithm to give better seek time [4].

Comparative analysis of all traditional disk scheduling algorithms based on their performance measure. This algorithm develops a simulation tools to represent the working procedure of all disk scheduling algorithms with their statistics [7].

3. Theoretical Background

The fundamental disk scheduling algorithms are First-In-First-Out, Shortest Seek Time First, Scan, and Circular Scan. The procedures of these algorithms are described:

3.1. First-In -First-Out (FIFO)

This processes items from the queue in sequential order. This strategy is a process in which the first input is retrieved as first output and then the next ones are also processed with queuing.

3.2. Shortest Service Time First (SSTF)

The shortest-service-time-first policy is to select the disk I/O request that requires the least movement of the disk arm from its current position. In this process of initial state to read and write, the arm of SSTF system will choose the nearest input and continues to the next with sequential range because the arm may move in two directions.

3.3. SCAN

In SCAN disk scheduling algorithm, head starts from one end of the disk and moves towards the other end, servicing requests in between one by one and reach the other end. Then the direction of the head is reversed and the process continues as head scan continuously back and forth to access the disk. So, this algorithm works as an elevator and hence also known as the elevator algorithm.

3.4. Circular-SCAN(C-SCAN)

The C-SCAN policy restricts scanning to one direction only. Thus, when the last track has been visited in one direction, the arm is returned to the opposite end of the disk and the scan begins again.

3.5. Seek Time

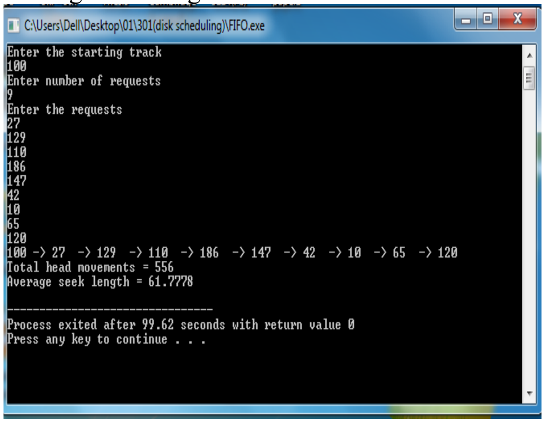
When the disk drive is running the disk rotates at fixed rate speed. To read and write, the arm has to be put at the longed track and at the launch of the longed part area on that track. The choice of track consists of the arm movement in a movable- system or electronic choosing on a fixed-system. It takes the time the arm to move at the track is called seek time on a movable-system. In this research, average seek length is calculated using the following equations:

$$\text{Average seek length} = \frac{\text{Total head movements}}{\text{number of tracks}}$$

4. Experimental Results

In this paper, assuming a disk with 200 tracks are used for compare the disk I/O performance and the disk request queue has random requests in it. The requested tracks, in the order received by the disk scheduler are 27, 129, 110, 186, 147, 42, 10, 65 and 120. And then calculate the above I/O request sequence using the FIFO, SSTF, SCAN, C-SCAN scheduling algorithms. The disk head is initially located at track 100 and is moving in the direction of decreasing track number.

Figure 1 shows the total head movements and average seek length of the FIFO disk schedule algorithms. If the disk head is initially at track 100, it will first move from 27 to 129, then to 110, 186, 147, 42, 10, 65 and finally to 120. FIFO disk schedule algorithm give:
Total Head Movements = (100-27)+(27-129)+(129-110)+(110-186)+(186-147)+(147-42)+(42-10)+(10-65)+(65-120) = 556 tracks
Average seek length = 61.77777



```
Enter the starting track
100
Enter number of requests
9
Enter the requests
27
129
110
186
147
42
10
65
120
100 -> 27  -> 129  -> 110  -> 186  -> 147  -> 42  -> 10  -> 65  -> 120
Total head movements = 556
Average seek length = 61.7778

--------------------------------
Process exited after 99.62 seconds with return value 0
Press any key to continue . . .
```

Figure 1. Calculations of FIFO Algorithm

Figure 2 describes the working of first in first out principle. Intial address is start at 100 and this algorithm works first job, second job and so on until last job is reached how job is short or long.They operate sequential of the sequence request and they can't interrupt the processes I/O disk.

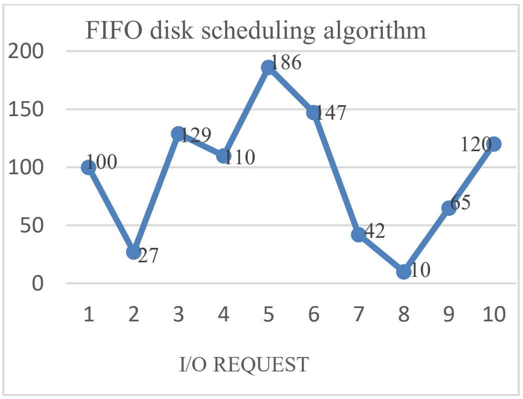
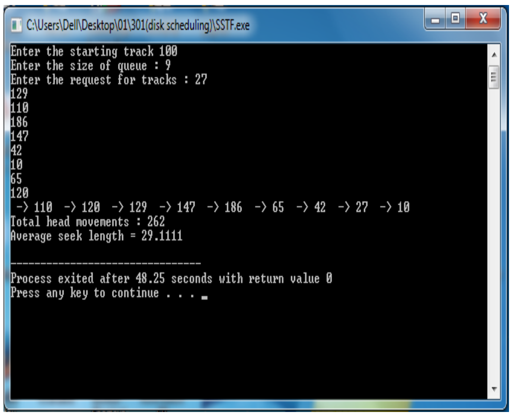


Figure 2. Describes the FIFO principle

Figure 3 shows the total head movements and average seek length of the SSTF disk schedule algorithms. If the disk head is initially at track 100, it will first move from 110 to 120, then to 129, 147, 186, 65, 42, 27 and finally to 10. Calculating for the total head movements of SSTF is described as follows:
Total Head Movements =(100-110)+(110-120)+(120-129)+(129-147)+(147-186)+(186-65)+(65-42)+(42-27)+(27-10) = 262 tracks
Average seek length = 29.1111



```
C:\Users\Dell\Desktop\01\301(disk scheduling)\SSTF.exe
Enter the starting track 100
Enter the size of queue : 9
Enter the request for tracks : 27
129
110
186
147
42
10
65
120
 -> 110  -> 120  -> 129  -> 147  -> 186  -> 65  -> 42  -> 27  -> 10
Total head movements : 262
Average seek length = 29.1111

--------------------------------
Process exited after 48.25 seconds with return value 0
Press any key to continue . . .
```

Figure 3. Calculations of SSTF Algorithm

Figure 4 describes the working of Shortest Seek Time First (SSTF) scheduling principle and this algorithm select the request closer to the initial disk head will serviced first. So the taken request for the job 110 is closer to the disk head track 100. And then this algorithm chooses to the next closest job and so on.

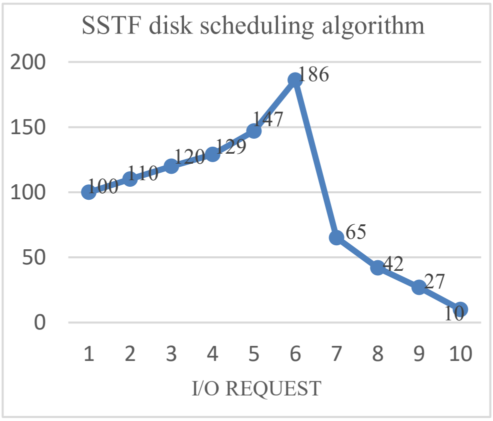
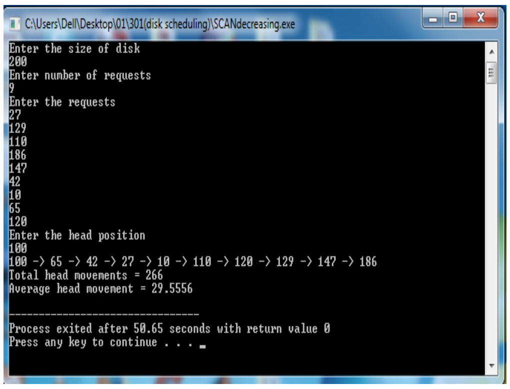


Figure 4. Describes the SSTF principle

Figure 5 shows the total head movements and average seek length of the SCAN disk scheduling algorithms. If the disk head is initially at track 100, it will first move from 65 to 42, then to 27, 10, 110, 120, 129, 147 and finally to 186.
Calculating for the total head movements of SCAN is described as follows:



```
C:\Users\Dell\Desktop\01\301(disk scheduling)\SCANdecreasing.exe
Enter the size of disk
200
Enter number of requests
9
Enter the requests
27
129
110
186
147
42
10
65
120
Enter the head position
100
100 -> 65 -> 42 -> 27 -> 10 -> 110 -> 120 -> 129 -> 147 -> 186
Total head movements = 266
Average head movement = 29.5556

--------------------------------
Process exited after 50.65 seconds with return value 0
Press any key to continue . . .
```

Total Head Movements =(100-65)+(65-42)+(42-27)+(27-10)+(10-110)+(110-120)+(120-129)+(129-147)+(147-186) = 266 tracks
Average seek length = 29.5556

Figure 5. Calculations of SCAN Algorithm

Figure 6 describes the processing of SCAN schedule principle and this algorithm is also known as elevator algorithm. The SCAN scheduling algorithm is the disk arm required to move in one direction and then the remaining requests are reverse direction.

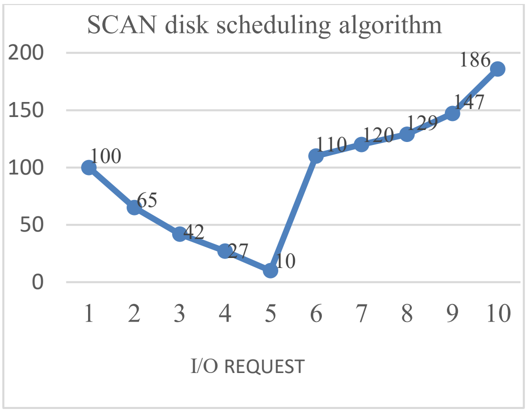


Figure 6. Describes the SCAN principle

Present the total head movements and average seek length of the C-SCAN disk schedule algorithm. If the disk head is initially at track 100, it will first move from 65 to 42, then to 27, 10, 186, 147,129, 120 and finally to 110.
Calculating for the total head movements of C-SCAN is described as follows:
Total Head Movements = (100-65)+(65-42)+(42-27)+(27-10)+(10-186)+(186-147)+(147-129)+(129-120)+(120-110) = 342 tracks
Average seek length = 38.0000

Figure 7 describes the processing of C-SCAN scheduling principle. The C-SCAN scheduling algorithm is a variant of SCAN scheduling. The C-SCAN scheduling algorithm is I/O head initially move from current location and then the last requests are reverse direction.

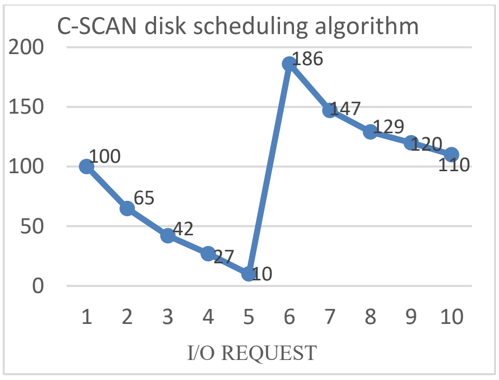


Figure 7. Describes the C-SCAN principle

Table 1 shows the comparison of the average head movement in all four kinds of disk scheduling algorithms. In this paper tested the five sequences of disk track requests:
First sequence = 27, 129, 110, 186, 147, 42, 10, 65, 120,
Second sequence = 55, 58, 39, 18, 90,160,150, 38, 184,
Third sequence = 10, 128, 110, 185, 150, 40, 27, 64, 120,
Fourth sequence = 53, 38, 110, 180, 146, 41, 10, 64, 120 and
Fifth sequence = 35, 65, 90, 170, 146, 40, 20, 64, 105 are run by C programming on four kinds of disk scheduling algorithms. Similarly, other requests for each individual can also be run. It is concluded that the average total head movements are 504.6, 248, 253.8 and 309.6. The average head movement of SSTF is 248.Thus, the SSTF principle is a good criterion for selecting the I/O requests from the disk queue.

Table 1. Comparative results of Four kinds of Disk Scheduling Algorithms

Sequence	FIFO	SSTF	SCAN	CSCAN
First	556	262	266	342
Second	498	248	248	282
Third	552	260	265	340
Fourth	482	250	260	330
Fifth	435	220	230	254
Average	504.6	248	253.8	309.6

Figure 8 gives comparative details of the four kinds of disk scheduling algorithms. The horizontal line represents five different operations and the average results and the vertical line expresses the calculation results of average total head movement in each algorithm.

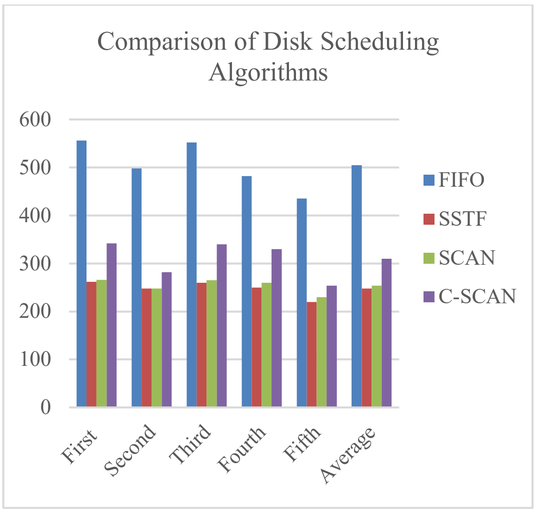


Figure 8. Comparative of Four kinds of Disk Scheduling Algorithms in graphical representation

5. Conclusions

In this paper, it is discussed the practical algorithms of four kinds of disk scheduling algorithms. Performance of disk scheduling is dependent on I/O requests, initial track and selected algorithm. In this research, Shortest Seek Time First gives minimum average seek length. So SSTF is the best for fast accessing I/O request.

6. Future Extension

In the future, random requests and some other parameters will be used to compare the various disk scheduling algorithms. There is a great deal of simulation tools in operating system. It is aimed to give knowledge how to run disk scheduling algorithms for all computer students after they have practiced and understood. Through self-access and practical, they can view significantly on both theory and practical of how to generate the disk scheduling algorithms in this paper. It is assumed that this system will be very helpful for both teachers and students who are learning simulation tools and operating system.

Acknowledgements

The authors would like to express her special thanks to Professor Dr. Thin Thu Naing, Rector of the University of Computer Studies, Taunggyi for an opportunity to carry out this paper.
The authors would like to special thanks the anonymous reviewers for their insight comment on this paper.

References

[1] William Stallings, “Operating Systems”, Internals and Design Principles (Seventh Edition).

[2] William Stallings, Operating Systems: Internals and Design Principles (India: Pearson, 2009).
[3] Abraham Silberschatz, Peter B. Galvin and Greg Gagne, (2014) "Operating System Concepts", Willey India 9th Edition, pp. 944.
[4] Amar Rajang Dash1, Sandipta Kumar Sahu2 and B Kewal316-8 "An Optimized Disk Scheduling Algorithm with Bad-Sector Management", International Journal of Computer - Science, Engineering and Applications (IJCSEA) 2019, Vol. 9, No.1/2/3
[5] C. Mallikarjuna and P. Chitti Babu 2016, "Performance Analysis of Disk Scheduling Algorithms", International Journal of Computer Sciences and Engineering, Vol. 4(5), pp. 180-184.
[6] SukAnya Suranauwarat, (2007),"A Disk Scheduling Algorithm Simulator" The ASEE Computers in Education (CoED) Journal, Vol. 8(3), pp. 1-9.
[7] Avneesh Shankar, Abhijeet Ravat, and Abhishek Kumar Pandey, "Comparative Study of Disk Scheduling Algorithms and Proposal of a New Algorithm for Better Efficiency". In Proceedings of and International Conference on Advanced Computing and Software Engineering (ICACSE) 2019. Available at SSRN: https://ssrn.com/abstract=3349013or http://dx.doi.org/10.2139/ssrn.3349013.