УДК 004.451.37

DISC SCHEDULER ALGORITHMS PERFORMANCE ANALYSIS USING OBJECT-ORIENTED PROGRAMMING

M.J. Tufegdžić

Academy of professional studies Šumadija, R. Krstića 19, 37240 Trstenik, Serbia
Corresponding author: mtufegdzic@asss.edu.rs

Disc scheduling, as one of the basic tasks of an operating system, requires applying various conventional techniques and policies, such as First Come First Serve (FCFS), Shortest Seek Time First (SSTF), SCAN, Circular SCAN (C-SCAN), LOOK, and Circular LOOK (C-LOOK). These algorithms will be presented and evaluated in terms of seek time calculated with results obtained for Total Head Movements (THMs) for all algorithms. Input data are randomly generated in the form of sequence of requests in executable files, obtained from scripts written in Python programming language. In order to facilitate running the executable files with a simple button click, a proper Graphical User Interface (GUI) in Python is also developed. The results of the study are compared for all proposed algorithms with the aim to estimate the algorithms' performances.

Keywords: FCFS; SSTF; SCAN; C-SCAN; LOOK; C-LOOK; seek time.

Introduction

Managing computer resources such as Central Processing Unit (CPU), memory space, storage space, and Input/Output (I/O) devices is one of the main responsibilities of an operating system. Due to the fact that the most modern computers use Hard Disk Drives (HDDs) and NonVolatile Memory (NVM) devices as an extension of main memory, the efficient and proper management of these devices is very important [1 p30].

HDDs have multiple spinning platters with the magnetic medium, which allows recording data on them. Platters are divided into concentric circles called tracks, further separated into sectors. The set of tracks at one arm position represents a cylinder. The read/write heads, located above the platters, provide reading and writing functions by accessing the proper sectors across the platters. These heads are attached to a disc arm [2 p12, 3 p053].

The time that is necessary to access data stored in HDDs affects computer system performances to a large degree. If this access time is too large, the performances greatly decrease [3 p053]. This is the reason why reducing the number of read/write head movements provides shorter access time and increases efficiency of disk drivers [4 p1]. In order to do this various disk scheduling techniques and policies are applied. The policies are categorized in two groups. The first group uses track's information while the other uses

additional information about requests' deadline or their priority [2 p13]. The policies from the first group use conventional disk scheduling algorithms to allocate the services to the requests, such as FCFS, SSTF, SCAN, C-SCAN, LOOK, and C-LOOK [2 p12].

Some studies include comparisons of basic disc scheduling algorithms according to the average head movement, with implementation of algorithms in Turbo C [5 p24], or in disc scheduling algorithm simulator developed in C# [6 p3]. Simulator in JAVA programming language with six basic modules for conventional disk scheduling algorithms has been developed [7 p111].

In order to evaluate various disc scheduling algorithms a set of criteria had to be established. Seek time, rotational latency, disk bandwidth, transfer time [3 p053, 5 p17] are some examples of these criteria. Some studies include access time as the combination of seek time, latency time and transfer time [3 p053, 5 p17].

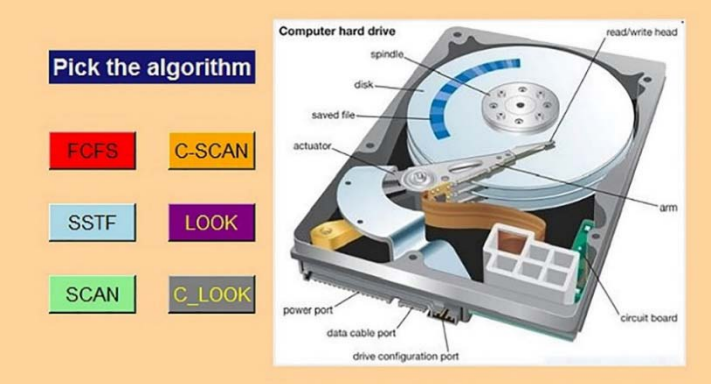
For the purpose of this study, seek time as a main parameter of disc scheduling algorithm is used. It represents the time required to move the head from current position to the right desired track [2 p12, 3 p053, 5 p17] and directly depends from the values of THMs [3 p054]. Values for THMs are taken from the executable files, which are running from proper GUI developed in Python programming language. These executable files are converted from scripts, also written in Python programming language for each algorithm. Proper plots are printed and the results are compared.

1. Methodology

The simplest algorithm from the group of conventional disk scheduling algorithms is FCFS algorithm. I/O requests are processed or served according to their order of arrival [2 p13, 3 p054, 4 p3, 5 p17, 6 p2]. Although it is simple to implement and the overhead is the smallest, it does not provide the fastest service [2 p13]. In SSTF algorithm, the selection of the next request is based on the criteria of the shortest distance from current head position, enabling minimum seek time [2 p14, 3 p054, 4 p4, 5 p18, 6 p2]. The performance is better than in the case of FCFS, but may cause starvation problems for some requests [2 p14, 3 p054]. SCAN algorithm acts as an "elevator" that alternately moves the heads from the beginning to the end of the disk and back again, serving the demands across the entire range of cylinders [2 p14, 3 p055, 4 p4, 5 p19, 6 p2]. This algorithm gives better performance than FCFS and SSTF, with less starvation then SSTF [2 p14]. C-SCAN algorithm, as a variant of SCAN algorithm, serve the requests only in one direction. When the head reaches the last cylinder, it moves to the beginning, without serving the requests along the way. After that, the service continues from the first to the last cylinder [2 p14, 3

p057, 4 p5, 5 p 21, 6 p3]. It provides more uniform waiting time than CSAN algorithm [3 p057]. LOOK and C-LOOK represents modifications of SCAN and C-SCAN algorithms. The heads move only to the last queued request in each direction. LOOK serves requests in both directions, while C-LOOK serves requests in ascending direction until it serves the last request in the queue. After that, it returns to the request that is closest to the beginning of the disk [2 p14, 3 p056-057, 5 p20-22, 6 p3]. LOOK algorithm eliminates unnecessary seek operations avoiding the problem of starvation [5 p20], while C-LOOK provides higher throughput, decreasing the variance of response time [2 p14].

Taking into account different ways to schedule disc requests, scripts for presented algorithms are written in Python programming language, according to their description presented above. These scripts are converted to executable files using PyInstaller. GUI with labels in Python programming language is designed (Picture 1) with the aid of tkinter module and its' methods.



Picture 1 – GUI for algorithms' selection

Simple click on the button named as appropriate algorithm (see Picture 1) allows for input data entry: number of discs, initial header position and sequence. The tested data are as it follows: number of discs 200, initial head position 50, and the sequence, randomly generated, is 82, 170, 43, 140, 24, 16, and 190, for each of the presented algorithms. THMs values are obtained as the results of programs' execution. The proper plots that graphically display the disk head movements are generated with the aid of sub-module pyplot from module matplotlib.

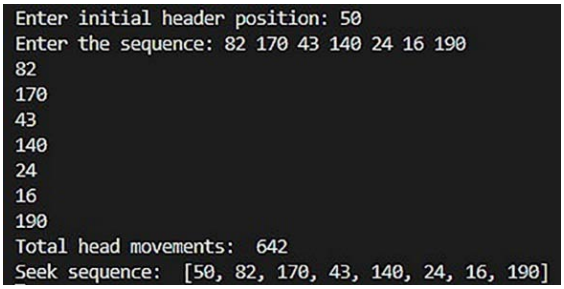
The values of seek time for six basic algorithms (FCFS, SSTF, SCAN, C-SCAN, LOOK, and C-LOOK) are calculated according to expression (1) given in [3 p054]:

$$\text{Seek Time} = \text{THMs} * \text{Seek rate}. \quad (1)$$

In equation (1) Seek rate is equal to 9 millisecond (ms) for modern hard drives [8].

2. Results and discussion

The scripts are written and tested in open source editor Visual Studio Code (Version 1.70.2), as well as the script written for GUI and plots printing. Results of code execution for the input data in case of FCFS algorithm, after clicking on proper button is presented in Picture 2, as an example.



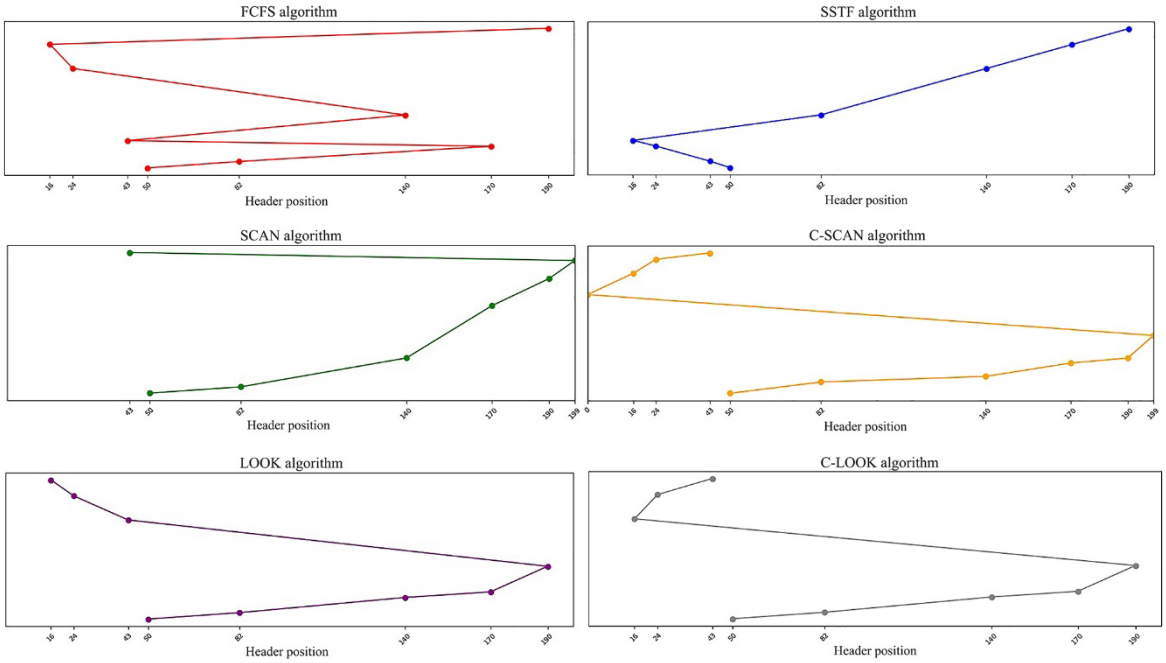
Picture 2 – Results of code execution for the input data for FCFS algorithm

The seek sequence in which requested tracks are serviced using FCFS algorithm is also presented in the form of an ordered array. The results of code executions in the form of THMs as well as the sequences in order of serviced requests are presented on Table 1. The Seek Times are also calculated.

Table 1 – Order of services requests, THMs and Seek Times

Algorithm	Order of serviced requests	THMs	Seek Time (ms)
FCFS	50, 82, 170, 43, 140, 24, 16, 19	642	5778
SSTF	50, 43, 24, 16, 82, 140, 170, 19	208	1872
SCAN	50, 82, 140, 170, 190, 199, 43	305	2745
C-SCAN	50, 82, 140, 170, 190, 199, 0, 16, 24, 43	391	3519
LOOK	50, 82, 140, 170, 190, 43, 24, 16	314	2826
C-LOOK	50, 82, 140, 170, 190, 16, 24, 43	341	3069

Plots in the form of graphs, that visualize the head movements are presented in Picture 3.



Picture 3 – Graphical visualization of disc scheduling algorithms

According to the the results of the study, based on seek time as the only easily measured performance, SSTF algorithm shows the best result, followed by SCAN algorithm. The worst performance is in the case of FCFS algorithm.

Conclusions

Proper management of HDDs implies a reduction in a number of read/write head movements and the time that is necessary to access data stored on them. Different issues of FCFS, SSTF, SCAN, C-SCAN, LOOK, and C-LOOK algorithms, as examples of conventional disc scheduling algorithms, are evaluated and compared in terms of seek time, as one of the main performance parameters. According to the results of the study, SSTF proved to be the algorithm that provides the best performance. The application of proposed methodology is facilitated to a large degree, due to GUI that allows easy and quick testing for different data sets in the form of disc requests. Further analysis should include some other policies for disc scheduling, as well as additional criteria for evaluation, such as access time.

References

1. Silberschatz A, Gagne G, Galvin PB. Operating System Concepts. N.-Y.: Tenth Edition Wiley, 2018. Chapter 1, Introduction. P. 4-39.

2. Ökdem S, Karaboğa D. Optimal disk scheduling based on ant colony optimization algorithm. Erciyes Üniversitesi Fen Bilimleri Enstitüsü Dergisi. 2006. № 22(1-2). P. 11-19, URL: https://dergipark.org.tr/tr/download/article-file/236632.
3. Jogamohan M, Partha PG. A comprehensive analysis of disk scheduling algorithms. International Journal of Latest Trends in Engineering and Technology [Internet]. 2018;11(1):053-058, URL: https://www.ijltet.org/journal/153334449210.%202610.pdf. DOI //dx.doi.org/10.21172/1.111.10.
4. Yashvir S, Prakash O. Selection of scheduling Algorithm. International Journal of Latest Trends in Engineering & Technology. 2012. № 1(2). P. 1–9. URL: https://arxiv.org/ftp/arxiv/papers/1210/1210.6447.pdf.
5. Shastri S, Sharma A, Mansotra V. A comparative analysis of disk scheduling algorithms. International Journal of Advanced Studies in Computer Science and Engineering. 2016. № 2(IV). P. 16–25. URL: https://arxiv.org/ftp/arxiv/papers/1210/1210.6447.pdf.
6. Suranauwarat S. A disk scheduling algorithm simulator. Computers in education journal. 2017. № 8(3). P. 1-9. URL: https://coed.asee.org/wp-content/uploads/2020/08/8-A-Disk-Scheduling-Algorithm-Simulator.pdf.
7. Bhade AW, Wankhade SR. File allocation methods performance over disk scheduling algorithms // International Journal of Application or Innovation in Engineering & Management. 2012. № 1(4). P. 109–120. URL: https://www.ijaiem.org/volume1Issue4/IJAIEM-2012-12-27-034.pdf.
8. Fisher T. What Does a Hard Drive's Seek Time Mean?. 2022. URL: https://www.lifewire.com/what-does-seek-time-mean-2626007.