Design and Simulation of Optimal Range Scheduling Algorithm

Jay Mohta
School of Engineering and Applied Sciences,
Ahmedabad University
Ahmedabad, India 380009

Suraj Patel
School of Engineering and Applied Sciences,
Ahmedabad University
Ahmedabad, India 380009

ABSTRACT

Disk Scheduling is performed by operating system to schedule IO requests arriving for disk. Disk system performance can be improved by dynamically scheduling and ordering the pending requests in the queue. Past analysis of the algorithms is experimental on certain datasets and not guaranteeing the optimal performance. In this paper, author proposes a disk scheduling algorithm which aims at reducing the seek time. Then the proposed algorithm is compared with conventional scheduling algorithms and simulated evidences are provided in the paper. Our results show that the proposed algorithm will improve the performance of disk by reducing the average seek time and thereby providing a faster disk subsystem.

Keywords

Operating System, Disk Scheduling, Optimization, Complexity Analysis

1. INTRODUCTION

In operating system, many processes generates read/write request for disk records and sometimes the process make requests faster than they are serviced by moving the head which results in queues being built up. Disk scheduling is also called IO scheduling. It is important because of the following reasons:

(1) Multiple IO request may arrive by different process and only one request could be served at a given point of time. Thus other request needs to wait in a queue.
(2) Two or more request may be far from each other so can result in greater disk arm movement.
(3) Hard drives are one of the slowest parts of computer system so needs to be accessed efficiently.

There are various types of delays associated with disk scheduling and they are listed as follows:[1]

(1) Seek Time: Seek time is the time taken to locate the disk arm to a specified track where the data is to be read or write.
(2) Rotational Latency: Rotational Latency is the time taken by the desired sector of disk to rotate into a position so that it can access the read/write heads.
(3) Transfer Time: Transfer time is the time to transfer the data.
(4) Disk Access Time = Seek Time + Rotational Latency + Transfer Time
(5) Disk Response Time: Response Time is the average of time spent by a request waiting to perform its I/O operation.

The goal is to minimize the disk response time so as to serve the request faster. The proposed algorithm provides the best disk response time compared to traditional scheduling algorithms.
Section 2 of the paper describes the traditional disk scheduling algorithms. Section 3 provides the Optimal Disk Scheduling algorithm proposed by the author. In Section 4 the author proposed disk scheduling algorithm is compared with traditional algorithms and there performance is analysed on random datasets. Section 5 summarizes the work done and suggests future research work.

2. CONVENTIONAL SCHEDULING ALGORITHM [2]

The following are the few well known conventional scheduling algorithms:

2.1 First In First Out (FIFO)

This algorithm serves the requests in the manner of their arrival. The first request is queued and served first and so on. The farther the location of the request, the higher the seek time will be.

2.2 Shortest Seek Time First (SSTF)

In SSTF, requests having shortest seek time are executed first. So, the seek time of every request is calculated in advance in queue and then they are scheduled according to their calculated seek time. As a result, the request near the disk arm will get executed first. SSTF is certainly an improvement over FCFS as it decreases the average response time and increases the throughput of system.

2.3 SCAN

In this algorithm, the disk head moves in a particular direction serving all the requests and after reaching the end of the disk reverses its direction serving all the requests in a reverse direction.

2.4 LOOK

It is similar to the SCAN disk scheduling algorithm except the difference that the disk arm in spite of going to the end of the disk goes only to the last request to be serviced in front of the head and then reverses its direction from there only.

2.5 CSCAN

In SCAN algorithm, the disk arm again scans the path that has been scanned, after reversing its direction. So, it may be possible that too many requests are waiting at the other end or there may be zero or few requests pending at the scanned area.

2.6 Median Range Scheduling Algorithm (MRSA) [3]

The main aim of this algorithm is to reduce the seek time by minimizing the number of head movement. In this algorithm, the requests are sorted and then if the head pointer is in the median range, the query having the least seek time of the median range is served first and then the algorithm proceeds to serve the next nearest requests. If the head pointer is not in median range then, the first or last request, whichever requires less seek time is served first and then the algorithm proceeds to serve the requests in ascending or descending order. The algorithm is described as follows:

```
        MR[] = List Containing Median Range
        lowMR = First Element of MR
        HighMR = Last Element of MR
        n = Number of Request to be served
        HP = Head Pointer
        A[] = Array containing the requests

(1) sort(A) in ascending order
(2) if( n is odd)
            MR[] = Elements in A indexed from ($\frac{n}{2}-1, \frac{n}{2}+1$)
    else
            MR[] = Elements in A indexed from ($\frac{n}{2}-1, \frac{n}{2}$)

(3) if(HP $\geq$ lowMR & HP $\leq$ HighMR )
            pos = Index Corresponding to lowest seek time value in MR[]
            if(pos$\leq \frac{n}{2}$)
                    serve(A) from left to right
            else
                    serve(A) from right to left
(4) else if(HP$\leq$lowMR)
            serve(A) from left to right
(5) else
            serve(A) from right to left
```

3. OPTIMAL RANGE DISK SCHEDULING

The main aim of all the algorithms described above is to reduce the seek time. The algorithm proposed here performs better than all the conventional disk scheduling algorithms mentioned above. The functioning of the algorithm is similar to median range scheduling. The difference here is, we look at the first and the last element of the sorted array and move to the side which minimizes the seek distance i.e. we move from left to right or right to left depending on which direction gives minimum seek distance from HP. Thus this algorithm is able to perform better than MRSA. The algorithm is described as follows:

```
        A[] = list containing all the request to be served
        n = number of requests
        HP = Head Pointer location
        midindex = $\frac{n}{2}$
        dir = Defining the final direction of the head pointer for the requests to be served. if(dir = 1) then move right to left else left to right.
        currptr = 0 (current pointer location)
        seekdist(a,b) = absolute difference between a and b

(1) sort(A)
(2) if(seekdist(HP,A[currptr])<seekdist(HP,A[n-currptr-1]))
            dir = 0         //serve from left to right
            break
(3) if(seekdist(HP,A[n-currptr-1] >seekdist(HP,A[currptr]))
            dir = 1         //serve from right to left
            break
(4) Increase Current Pointer by 1(currptr = currptr + 1).
(5) Repeat steps 2 to 4 until currptr $\leq$ midindex.
(6) Serve the request based on the value of dir.
(7) exit.
```

4. RESULT AND PERFORMANCE ANALYSIS

4.1 Assumptions and Parameters

All the request are independent of each other and have the same priority. The request are initially stored in a queue and all cases are considered ideal in nature. The goal of our algorithm is to reduce the head movements which means to minimize the average seek time.

4.2 Performance Evaluation

Suppose the disk has 200 tracks numbered from 1 to 200. Consider a disk queue which is array A in our algorithm as follows:

Disk Drives: 200 Cylinders
Sequence: 60,90,135,155,25,180,190,160,200
Head Pointer: 100

Total Head movements is given as follows:

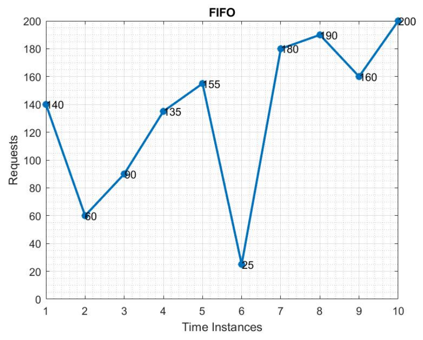

Fig. 1. **FIFO**

FIFO=(140-60)+(90-60)+(135-90)+(155-135)+(155-25)+(180-25)+ (190-180)+(190-160)+(200-160)=540

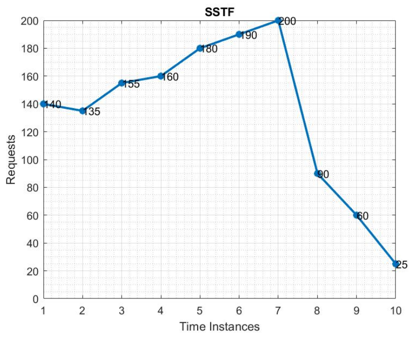

Fig. 2. **SSTF**

SSTF=(140-135)+(155-135)+(160-155)+(180-160)+(190-180)+(200-190)+(200-90)+(90-60)+(60-25)=245

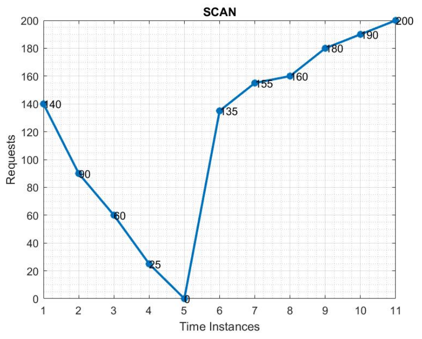

Fig. 3. **SCAN**

SCAN=(140-90)+(90-60)+(60-25)+(25-0)+(135-0)+(155-135)+(160-155)+(180-160)+(190-180)+(200-190)=340

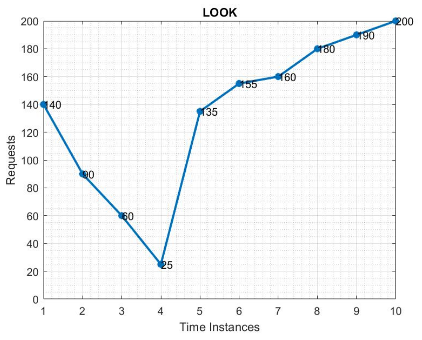

Fig. 4. **LOOK**

LOOK =(140-90)+(90-60)+(60-25)+(135-25)+(155-135)+ (160-155)+(180-160)+(190-180)+(200-190)=290

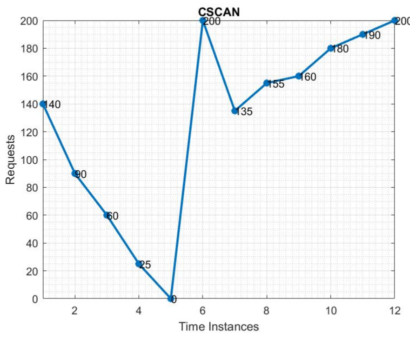

Fig. 5. **CSCAN**

CSCAN=(140-90)+(90-60)+(60-25)+(25-0)+(200-0)+(200-135)+(155-135)+(160-155)+(180-160)+(190-180)+(200-190)=540

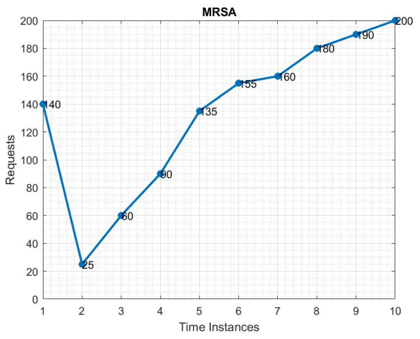

Fig. 6. **MRSA**

MRSA=(140-25)+(60-25)+(90-60)+(135-90)+(155-135)+(160-155) +(180-160)+(185-180)+(190-185)+(200-190)=290

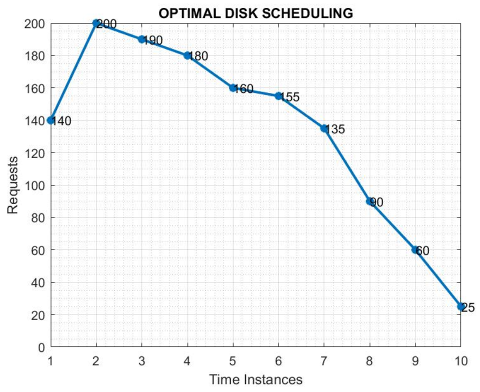

Fig. 7. **Optimal Disk Scheduling**

ODS=(200-140)+(200-190)+(190-180)+(180-160)+(160-155)+(155-135)+(135-90)+(90-60)+(60-25)=235

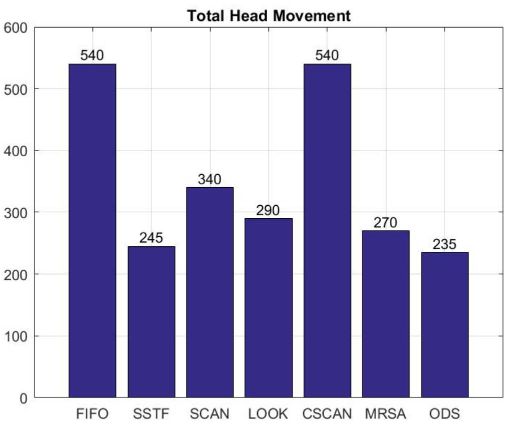

Fig. 8. **Total Head Movement**

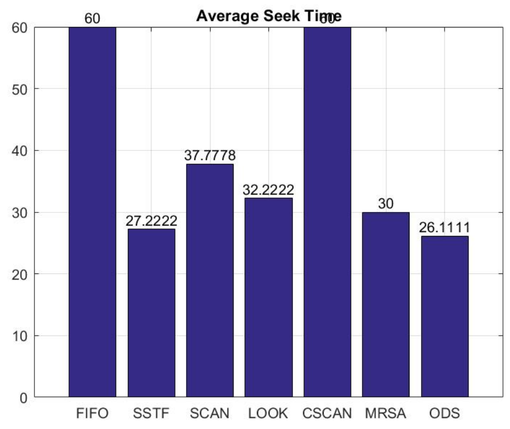

Fig. 9. **Average Seek Time**

It is observed from the empirical results shown above that ODS is better than all the traditional disk scheduling algortihms. These are the adavantages of ODS:

(1) It is very easy to implement. It provides a time complexity of O$(n * log(n))$
(2) The algorithm always leads to optimal scheduling providing the least seek time.

5. CONCLUSION

In this paper, the author proposes a new disk scheduling algorithm which performs better than all the traditional disk scheduling algorithm. The average seek time has been improved compared to traditional algorithms. This algorithm can be implemented on real time system and has applications, in the fields of operating systems, distributed computing, heterogeneous systems, cluster computing, computational models and multi criteria analysis. The algorithm described in the paper is off-line algorithm, to use this algorithm for real time disk scheduling we need to make it online so that it can process the query in online manner. However, if the request pattern is known apriori the proposed algorithm will produce best results.

6. REFERENCES

[1] B. L. Worthington, G. R. Ganger, and Y. N. Patt, *Scheduling algorithms for modern disk drives,*. in ACM SIGMETRICS Performance Evaluation Review, vol. 22, no. 1. ACM, 1994, pp. 241 to 251.
[2] M. Y. Javed and I. Khan, *Simulation and performance comparison of four disk scheduling algorithms,*. in TENCON 2000. Proceedings, vol. 2. IEEE, 2000, pp. 10 to 15.
[3] J Vachhani and Y. Turakhia, *Design and Performance Evaluation of Median Range Scheduling Algorithm,*. in International Journal of Computer Application Volume 172 No4 (August 2017).