Design and Performance Evaluation of Median Range Scheduling Algorithm

Jainil Vachhani
School of Engineering and Applied Sciences,
Ahmedabad University
Ahmedabad, India 380009

Yash Turakhia
School of Engineering and Applied Sciences,
Ahmedabad University
Ahmedabad, India 380009

ABSTRACT
Disk system performance can be dramatically improved by dynamically scheduling, ordering and pending requests. Past analysis of disk scheduling algorithms has to largely experimental with little effort to develop algorithms with measurable performance guarantees. In this paper, the authors propose an algorithm that reduces average seek time. Then the proposed algorithm is compared with conventional scheduling algorithm and measurable evidence is provided for the same. Our results and calculations show that the proposed algorithm will improve the performance of the disk by reducing average seek time and thereby providing a faster disk subsystem

Keywords
Operating System, Scheduling Algorithm, Optimization.

1. INTRODUCTION
In operating systems, many processes simultaneously generate read/write request for disk records and sometimes the processes make requests faster than they are serviced by moving head which results in queues being build up for devices. Disk scheduling technique is the process of allocating these requests such that it minimizes head movement and provides least seek time. According to, [1] incorporating information into scheduler provides less than 2% delay while algorithms that reduce overall position delays can provide significant performance improvement by exploiting a perfecting cache.

A set of evaluation criteria for any disk scheduling algorithm is established as follows: [2]

- Seek Time: Total time taken by disk arm to move to head of cylinder.
- Rotational Delay: Additional delay for the disk to reach the desired section of the head.
- Disk Bandwidth: Total information, measured in bytes, transferred between first request and completion of the last request.
- Transfer Time: Time required for transfer, depending on rotational speed.

Section 2 describes few of the most conventional disk scheduling algorithms. Section 3 provides a detailed description of the Median range algorithm. In section 4, the proposed algorithm is compared with traditional disk scheduling algorithm on randomized data sets. Section 5 summarizes this work and suggests avenues for future research. Related work in this field can be found at [3] and [4].

2. CONVENTIONAL SCHEDULING ALGORITHM
The following are the few well known conventional scheduling algorithms: [5].

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 this algorithm, the read write head moves to the track nearest to the head position. The request requiring minimum seek time is served first and so on.

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.

2.4 LOOK
In this algorithm, the head moves in a particular direction, serving the request, reaches the last request and then reverses its direction.

2.5 CSCAN
In this scheduling algorithm, the head moves in a particular direction, serving the requests and after reaching the end, jumps in opposite direction, without serving any requests and then reverses its direction until all the requests are served.

3. MEDIAN RANGE SCHEDULING ALGORITHM
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 then 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 median range scheduling (MRSA) is defined as follows

3.1 Algorithm
1. Declaration and Initialization

A[ ]: A list containing all the requests to be served.
HP: Head Position.

n: Number of requests to be served.
MR[ ]: List containing MR(Median Range)
lowMR: First element of MR[ ]
highMR: Last element of MR[ ]
LST: The element having least difference from
HP: Head Pointer
LST(i): The array index corresponding to
LST: LowestSeek Time value

2. Sort(A)

3. if(n is odd)

$$MR[\,] = \left[\left(\frac{n}{2} - 1\right)\right], \left[\,\left(\frac{n}{2} + 1\right)\right]$$

else

$$MR[\,] = \left(\frac{n}{2} - 1\,, n/2\right)$$

4. if($HP \geq lowMR\ and\ HP \leq highMAR$)
serve LST
j = LST(i)

while(j is not 0)
serve A[j]
j = j - 1
j = LST(i+1)

while(j is not n)
serve A[j]
j = j + 1

5. else if($HP \leq lowMR$)
for(j=0 to n-1)
serve(A[j])

else if (HP _ highMR)
for(j=n-1 to 0)
serve(A[j])

4. RESULT AND PERFORMANCE ANALYSIS

4.1 Performance Assumptions and Parameters

All requests are independent of each other and have the same priority. The requests are initially stored in a request queue and. all cases are considered ideal in nature. The performance based on minimum seek time i.e. average seek time should be less for better performance.

4.2 Performance Evaluation

Suppose a disk drive has 200 cylinders, numbered as 0 to 199. Consider a disk queue with requests: 190, 75, 155, 25, 85, 130, and 120 for I/O to blocks on cylinder. Assume the R/W head is at 100.

Disk Drive: 200 cylinders
Sequence: 190, 75, 155, 25, 85, 130, and 120
R/W arm: 100

Total head movement is given as follows:

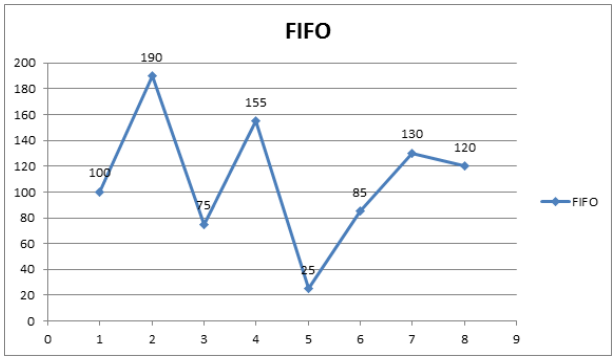

Fig 1: FIFO

FIFO = 190 - 100) + (190 – 75) + (155 – 75) + (155 - 25) + (85 - 25) + (130 – 85) + (130 – 120) = 530

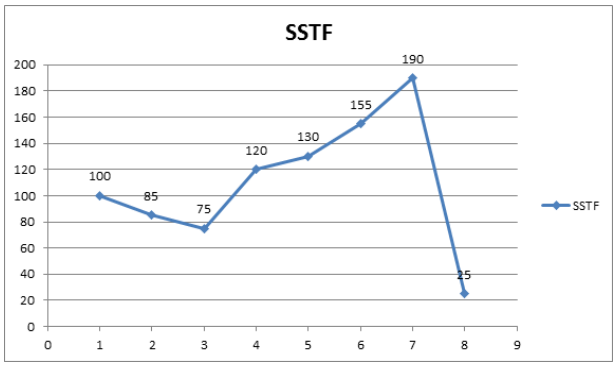

Fig 2: SSTF

SSTF = (100 – 85) + (85 – 75) + (120 – 75) + (130 - 120) + (155 – 130) + (190 – 155) + (190 – 25) = 310

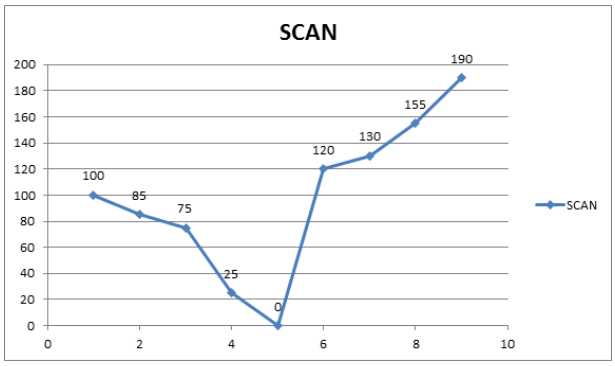

Fig 3: SCAN

SCAN = (100 – 85) + (85 – 75) + (75 – 25) + (25 - 0) + (120 - 0) + (130 – 120) + (155 – 130) + (190 – 155) = 305

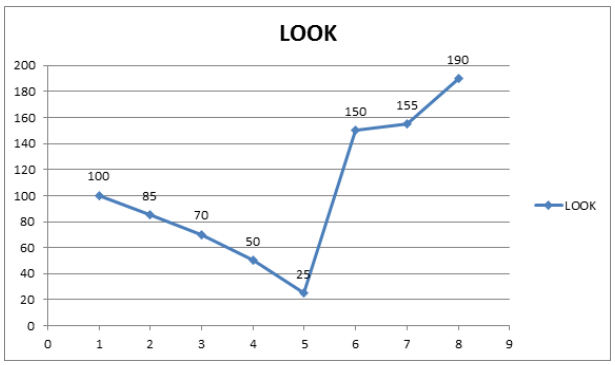

Fig 4: LOOK

LOOK = (100 – 85) + (85 – 75) + (75 – 25) + (190 - 25) + (190 – 155) + (155 – 130) + (130 – 120) = 315

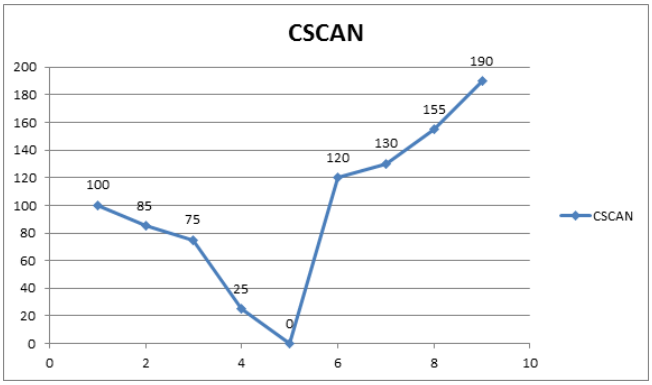

Fig 5: CSCAN

CSCAN = (100 – 85) + (85 – 75) + (75 – 25) + (25 - 0) + (120 - 0) + (130 – 120) + (155 – 130) + (190 – 155) = 290

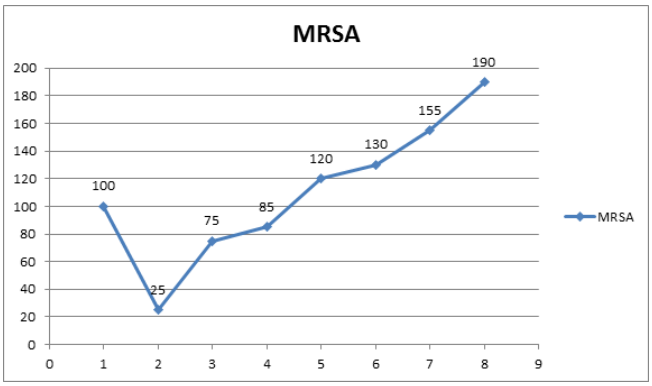

Fig 6: MRSA

MRSA = (100 – 25) + (75 – 25) + (85 – 75) + (120 - 85) + (130 – 120) + (155 – 130) + (190 – 155) = 240

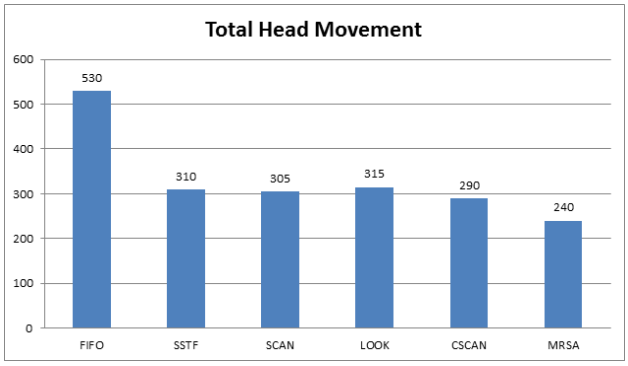

Fig 7: Total Head Movement

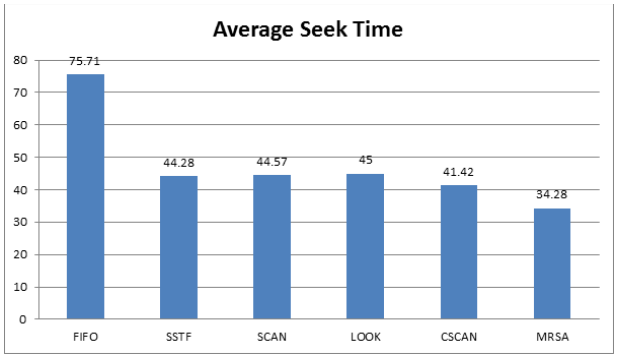

Fig 8: Average Seek Time

It is observed from above experiment that MRSA is better than traditional scheduling algorithms. There are following advantages of using MRSA:

1. The total disk movement is always less than 2N where N is the position of last track.
2. If all requests are concentrated near median, then it provides best results.
3. The algorithm is simple and easy to follow.

5. CONCLUSION

In this paper, the authors proposed and implemented a new disk scheduling algorithm that works better than conventional scheduling algorithms and imposes almost no performance penalty when provided with sufficient slack time. The average seek time and transfer time has been improved which in turn improves the performance of disk. 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.

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. 241–251.

[2] M. M. Kumar and B. R. Rajendra, "An improved approach to maximize the performance of disk scheduling algorithm by minimizing the head movement and seek time using sort mid current comparison (smcc) algorithm," Procedia Computer Science, vol. 57, pp. 222–231, 2015.

[3] W. Basu and S. Chaudhuri, "Missed deadlines should be consideredproposals for modifying existing real-time disk scheduling algorithms," International Journal of Advanced Research in Computer Science, vol. 7, no. 3, 2016.

[4] M. Lee, "A disk scheduling algorithms based on the insertion and two-way scan techniques," International Information Institute (Tokyo). Information, vol. 19, no. 5, p. 1565, 2016.

[5] M. Y. Javed and I. Khan, "Simulation and performance comparison of four disk scheduling algorithms," in TENCON 2000. Proceedings, vol. 2. IEEE, 2000, pp. 10–15.