INTERNATIONAL JOURNAL OF PURE AND APPLIED RESEARCH IN ENGINEERING AND TECHNOLOGY

A PATH FOR HORIZING YOUR INNOVATIVE WORK

PERFORMANCE EVALUATION OF CONVENTIONAL DISK SCHEDULING ALGORITHMS

SALIM. Y. AMDANI, M. S. ALI, SUREKHA. B. CHAVHAN

1. Associate Professor, Department of CSE, BNCOE, Pusad, India.
2. Principal, P.R.M.C.E. & M, Bandera, India.
3. M.E. Student, Department of CSE, BNCOE, Pusad, India.

Accepted Date:

27/02/2013

Publish Date:

01/04/2013

Keywords

Disk Scheduling,

Seek Time,

Sorting,

Average Seek Time

Corresponding Author

Mr. Salim Y. Amdani

Abstract

In Disk Scheduling, seek time is very important. Since all device requests are linked in queues, the seek time is increased causing the system to slow down. There are different algorithms that reduce the seek time of all requests. FCFS algorithm is used to determine the number of head movements that would simply find the number of tracks it took to move from one request to the next. In SSTF, request is serviced according to next shortest distance. SCAN algorithm scans down towards the nearest end and then when it hits the bottom it scans up servicing the requests that it didn't get going down. Most of the disks are characterized by a linear seek time and their seek time is responsible for the most time of disk access. So, the existing disk scheduling algorithm have focused on the reduction of the average seek time. By reducing the average seeks time, we can improve the performance of disk I/O operation. But in FCFS if any process of maximum burst time is first come and after that a short burst time process come, then smaller processes have to wait for a long time until the maximum burst time process complete its execution. Also, in SSTF switching directions slows things down. Therefore, a new algorithm called ODSA (**Optimized Disk Scheduling Algorithm)** is proposed which takes less average seek time as compared to other disk scheduling algorithms (FIFO, SSTF, etc.) which enhances the scheduling of disk I/O requests in a better manner

INTRODUCTION:

The most basic algorithm a scheduler can implement is known as First In, First out (FIFO) scheduling or sometimes First Come, First Served (FCFS). FIFO has a number of positive properties: it is clearly very simple and thus easy to implement. SSTF works similar to SJF, which selects the disk I/O request that requires the least movement of the disk arm from its current position, regardless of direction reduces total seek time compared to FCFS. SCAN algorithm states that the head moves from its current position towards one direction and serves all requests in its way until it reaches the last cylinder in this direction, then it switches the direction to starts serving the requests along all cylinders till it reaches the other end. Circular SCAN(C-SCAN) moves inwards servicing requests until it reaches the innermost cylinder; then jumps to the outside cylinder of the disk without servicing any requests [1]. The disk scheduler is responsible for dynamically ordering the pending requests. By taking into account the various delays associated with disk accesses, a scheduler can minimize the total positioning overhead while providing reasonable response times for individual requests [2]. Management of disk scheduling is a very important aspect of operating system. Performance of the disk scheduling completely depends on how efficient is the scheduling algorithm to allocate services to the request in a better manner. The main aim of disk scheduling algorithms is to reduce or minimize the seek time for a set of requests. By reducing the average seeks time, we can improve the performance of disk I/O operation. In our proposed algorithm, Optimize Disk Scheduling Algorithm (ODSA) is taking less average seek time as compare to other disk scheduling algorithms like FCFS, SSTF [3].

DISK SCHEDULING PARAMETER:

The time taken to position the head at the desired track is called Seek Time. The time taken to reach the desired sector is called Latency Time or *Rotational Delay*. The sum of seek time and rotational delay is called Access Time [4].

RELATED WORK DONE:

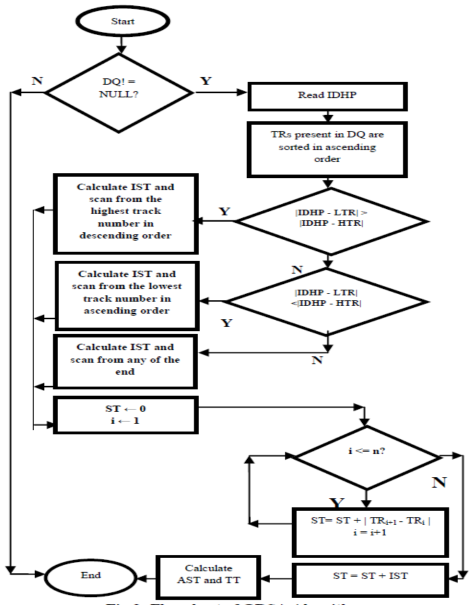
Most traditional disk scheduling algorithms, such as FCFS, SCAN, C-SCAN, LOOK, and SSTF are designed to reduce disk-seek time and increase its throughput

[5]. Daniel L. Martens and Michael J. Katchabaw developed a new disk scheduling algorithm focuses on dynamic scheduling algorithm selection and tuning [7].

ODSA ALGORITHM:

In ODSA algorithm, the requests in the disk queue are to be sorted according to the track number requested. Then we calculate the absolute difference between the initial disk head position (IDHP) and the lowest track request (LTR) of disk queue and absolute difference of the initial disk head position (IDHP) and the highest track request (HTR) of the disk queue. If (|IDHP – LTR|) is greater than (|IDHP – HTR|), then we scan the requests in ascending order starting from the initial position and if (|IDHP – LTR|) is less than (|IDHP – HTR|), then scanning starts in descending order (Highest track number to lowest track number). If (|IDHP – LTR|) is equal to (|IDHP – HTR|), then scanning can start from any of the side. Finally, we calculate the average seek time. The pseudo code of the algorithm is represented in figure 1 and figure 1 represents the flowchart of the algorithm.

Transaction ID	Track location	Arrival Time
T0	11	1
T1	1	2
T2	8	4
T3	4	5
T4	7	7
T5	14	9
T6	8	10
T7	22	12


Start
DQ! = NULL?
N
Y
Read IDHP
TRs present in DQ are sorted in ascending order
|IDHP - LTR| > |IDHP - HTR|
Y
Calculate IST and scan from the highest track number in descending order
N
|IDHP - LTR| <|IDHP - HTR|
Y
Calculate IST and scan from the lowest track number in ascending order
N
Calculate IST and scan from any of the end
ST ← 0
i ← 1
i <= n?
Y
N
ST= ST + | TRi+1 - TRi |
i = i+1
ST = ST + IST
Calculate AST and TT
End
Fig 2: Flowchart of ODSA Algorithm

EXAMPLE:

To illustrate the algorithms consider the following transactions having track locations and arrival times as follows:

Consider initial disk head position (IDHP) = 6

Now, each algorithm is illustrated as follows:

FIFO:

Also known as **F**irst **C**ome, **F**irst - **S**erved (FCFS), is the simplest scheduling algorithm, FIFO simply queue processes in the order that they arrive in the ready queue. To determine the number of head movements you would simply find the number of tracks it took to move from one request to the next

The FIFO schedule of above example is:

T0 T1 T2 T3 T4 T5 T6 T7

After servicing third transaction

Seek time (3) = ST = ST + | TRi+1 - TRi |

=15+| (track location of first transaction – track location of second)|

=15+|8-1| =22

Similarly after servicing other transaction

Seek time (4) =26,

Seek time (5) =29

Seek time (6) =36,

Seek time (7)=40

Seek time (8) =44

Now Average Seek Time (AST) for FCFS=seek time of last transaction/total number of transaction =44/8= 5.5

SSTF:

Like SJF, select the disk I/O request that requires the least movement of the disk arm from its current position, regardless of direction. It reduces total seek time compared to FCFS. In this case request is serviced according to next shortest distance.

The SSTF schedule of above example is:

T4 T2 T0 T5 T6 T7 T3 T1

initially Initially Seek time(IST)=0 and IDHP=6

For first transaction

Seek time(1) = ST + | TRi+1 - TRi |

=[seek time + (initial disk head position-track location of first transaction)] = |ST+(IDHP-track location of first transaction)|

= |0+(6-7)|=1

After servicing second transaction

Seek time (2) ST = ST + | TRi+1 - TRi |

= 1+|(track location of first transaction – track location of second)|

= 1+|7-8|

= 2

Now initially Seek time(IST)=0 and IDHP=6

For first transaction

Seek time(1)= ST = ST + |TRi+1 - TRi|

= [seek time + (initial disk head position-track location of first transaction)]

= |ST+ (IDHP-track location of first transaction)|

= |0+ (6-11)|=5

After servicing second transaction

Seek time (2) = ST = ST + |TRi+1 - TRi|

= 5+| (track location of first transaction – track location of second)|

= 5+|11-1|=15

After servicing third transaction

Seek time (3) ST = ST + | TRi+1 - TRi |

=2+| (track location of first transaction – track location of second)|

=2+|8-11|

=5

Similarly after servicing other trancsaction

Seek time (4) =8,

Seek time (5) =12

Seek time (6) =16,

Seek time (7) =34

Seek time (8) =37

Now Average Seek Time (AST) for

SSTF==37/8=4.6

ODSA:

Now we will illustrate ODSA for the above example.

1. initially DQ != NULL

2. IDHP(initial disk head position)=6

LTR = Lowest Track Request

HTR = Highest Track Request

IST = Initial Seek Time

TR = Track Request

n = number of TRs

ST=Seek time

3. All the transaction present in DQ sorted in ascending order is as follows:

T1 T3 T4 T2 T0 T5 T6 T7

If (| IDHP - LTR |) < (| IDHP - HTR |)

(|6-1|)< (|6-22|)

5<22 TRUE

Therefore IST = | IDHP - LTR |

=|6-1|=5

Scanning will start from the LTR

//Therefore the servicing schedule for above example using ODSA is as follows.

4. ST ← 0

// initializing ST to 0

5. For i=1

Seek time(1) ST = ST + | TRi+1 - TRi |

=[seek time + (initial disk head position-track location of first transaction)]

= |ST+ (IDHP-track location of first transaction)|

= |0+ (6-1)|

= 5

For i=2

Seek time (1) ST = ST + | TRi+1 - TRi |

=5+| (track location of first transaction – track location of second)|

=5+| (4-1)|

=8

For i=3

Seek time (1) ST = ST + | TRi+1 - TRi |

=8+| (track location of first transaction – track location of second)|

=8+ (|7-4|) =11

Similarly, after servicing other transaction

For i=4:- Seek time (4) =12

For i=5:- Seek time (5) =15

For i=6:- Seek time (6) =18

For i=7:- Seek time (7) =22

For i=8:- Seek time (8) =26

Now Average Seek Time (AST) for ODSA = seek time of last transaction/total number of transaction =26/8 =3.25

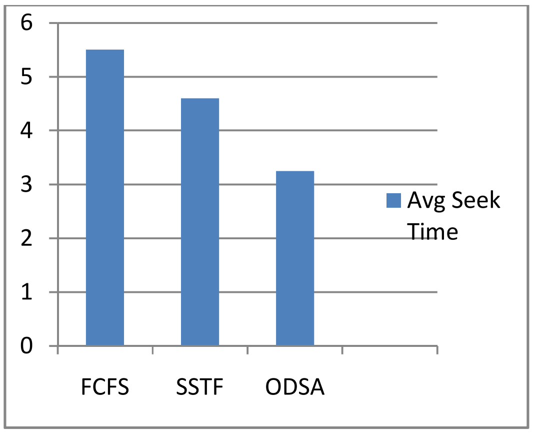
PERFORMANCE EVALUATION:

In optimized disk scheduling algorithm(ODSA) The less the head has to move the faster the seek time will be. Here in comparison the average seek time(ST) is less than the FIFO and SSTF. Relatively this algorithm is based on parameters like seek time and transfer time.

Algorithm	Average Seek Time Required
FIFO	5.5
SSTF	4.6
ODSA	3.25

Table: Comparison of Algorithm

PERFORMANCE EVALUATION GRAPH:



CONCLUSION:

Thus we have studied different types of algorithm FCFS, SSTF, ODSA. Performance evaluation shows that average seek time (AST) required for ODSA (Optimized disk scheduling algorithm) is less than the FCFS and SSTF algorithm. ODSA algorithm which increases the efficiency of the disk performance. In future we can implement this ODSA algorithm in real time systems.

REFERENCES:

1. Ammar Muqaddas, Hanady Abdulsalam, and Ayed Salman : “S-LOOK: A Preemptive Disk Scheduling Algorithm for Offline and Online Environments”. Computer Engineering Department, Kuwait University, Kuwait].

2. Bruce L. W orthington, Gregory R. Ganger, Y ale N. Patt: “Scheduling Algorithms for Modern Disk Drives” Appeared in the Proceedings of the ACM Sigmetrics Conference, May, 1994, pp. 241-251.]

3. Sourav Kumar Bhoi ,Sanjaya Kumar Panda ,Imran Hossain Faruk : “ Design and Performance Evaluation of an Optimized Disk Scheduling Algorithm (ODSA)” International Journal of Computer Applications (0975 – 8887) Volume 40– No.11

4. A. Silberschatz, P. B. Galvin and G. Gagne, “Operating System Principles”, 7th Edn., John Wiley and Sons, 2008, ISBN 978-81-265-0962-1.]

5. R.Muthu Selvi and R.Rajaram: “A Genetic Based Approach for Multiobjective Optimization of Disk Scheduling to reduce completion time and missed task”. International Journal of Information Technology Convergence and Services (IJITCS) Vol.1, No.4, August 2011].

6. Manish Kumar Mishra:”Major Half Served First (MHSF) Disk Scheduling Algorithm” International Journal of Computer Applications & Information Technology Vol. II, Issue I, January 2013 (ISSN: 2278-7720)

7. D. L. Martens and M. J. Katchabaw, “Optimizing System Performance Through Dynamic Disk Scheduling Algorithm Selection”, Department of Computer Science, The University of Western Ontario, London, Canada

8. A. Silberschatz, P. B. Galvin, G. Gagne, 2005. Operating System Concepts, seventh ed. John Wiley and Sons Inc, New Delhi.

9. W. A. Burkhard, J. D. Palmer, 2002. Rotational Position Optimization (RPO) Disk Scheduling. FAST, Monterey, California.