Global Journals LaTeX JournalKaleidoscope™
Artificial Intelligence formulated this projection for compatibility purposes from the original article published at Global Journals. However, this technology is currently in beta. *Therefore, kindly ignore odd layouts, missed formulae, text, tables, or figures.*

Sim_Dsc: Simulator for Optimizing the Performance of Disk Scheduling Algorithms

Sumit Mittal[1]

[1] Kurukshetra University

Received: 10 August 2011 Accepted: 3 September 2011 Published: 16 September 2011

Abstract

Disk scheduling involves a careful examination of pending requests to determine the most efficient way to service these requests. A disk scheduler examines the positional relationship among waiting requests, then reorders the queue so that the requests will be serviced with minimum seek. The purpose of the study is to obtain the best scheduling algorithm based on the seek time, rotation time and transfer time for moveable head disks. Keeping in view an attempt has been made to design a simulator for optimizing the performance of disk scheduling algorithms using Box-Muller transformation. The input for the simulator has been derived by using an algorithm for generating pseudo random numbers which follows box-muller transformations. Simulator takes access time which is generated using seek time, rotation time and transfer time, as the request of cylinder numbers, current position of read/write head as inputs. On the basis of these inputs, total head movement of each disk scheduling algorithm is calculated under various loads.

Index terms— disk scheduling algorithms, seek time, rotational delay, transfer time, access time, head movement, box-muller transformation.

1 INTRODUCTION

mong major responsibilities of operating system disk scheduling is one of the important tasks to use disk efficiently. For meeting these objective disk drives should have fast access time and disk bandwidth. Access time is improved by scheduling the service of disk I/O in a good manner. Many processes make request for reading/writing data on disk simultaneously. As these requests sometimes makes requests faster than serviced by the disk. Therefore, a request queue has to hold disk requests. To reduce the time spent seeking records, the request queue is ordered in some manner. This process is called Disk scheduling.

A disk-scheduling algorithm decides that which request of cylinder is to be serviced when there are so many requests. Various disk-scheduling algorithms are used. However, there will be common criteria for evaluating the performance of all these algorithms that is total head movement. Each algorithm aims to minimise the total head movement. The algorithms can be evaluated by running them on a particular string of randomly generated requests and computing the access time of the moveable head disks. Access Time has two major components. First one is Seek time and another on is Rotational Latency Time. The Seek Time is the time taken by read/write head to reach at a requested Cylinder/Track number and later one the time taken by the disk to rotate the desired sector under the read/write head. The disk bandwidth is defined as the total number of bytes transferred, divided by the total time between first request and completion of last transfer. Both the access time and disk bandwidth can be improved by scheduling the service of disk I/O in a good manner [7]. The time taken by a disk to move the required data under the read/write head is called rotational latency time. A disk's average rotational latency is simply half the time it takes to complete one revolution.

2 a) FCFS algorithm

This algorithm treats the requests of cylinders as a FIFO queue. Besides simplicity, this policy is preferred because this ensures that no request can be postponed indefinitely. This policy suffers from global zigzag effect.

3 b) SSTF algorithm

This algorithm selects the request, which has shortest seek from the current position of R/W head. As this policy can leads to indefinite postponement of the requests, which are not closer to R/W head. This policy gives a substantial improvement in performance, but it leads to problem of starvation.

4 c) SCAN algorithm

In this algorithm request is chosen for service that requires the shortest seek in preferred direction & do not change the direction until it reaches at the end of the disk. After that head moves in reverse direction and services all the requests in the opposite direction. This policy is also called as elevator algorithm.

5 d) C-SCAN algorithm

In C-Scan head moves only in one direction to service the requests. When head moves in reverse direction it does not service the incoming requests. When head has completed its inward sweep, it jumps to outermost cylinder without servicing the requests and then it resumes its inward sweep. In this, head goes only as far as the final request in each direction. Then, it reverses direction immediately, without going all the way to end of disk. It is appropriate to call the elevator algorithm as it continuous in one direction until it reaches the last request in that direction, then reverse direction.

6 f) C-Look algorithm

This algorithm reduces the bias against request located at the extreme ends of platters. When there is no request on a current sweep in either direction (inward or outward) the read/write head moves to the request closest to the outer/inner cylinder and again begins the next sweep.

7 II.

8 RELATED WORK

David M. Jacobson and John Wilkes [1] have discussed the disk scheduling algorithm based on rotational position in their research paper. Disk scheduling based on rotational position as well as disk arm position is shown to provide improved performance. The access time based algorithms match or outperform all the seek-time ones. The best of them is Aged Shortest Access Time First, or ASATF, which forms a continuum between FCFS and SATF. It is equal or superior to the others in both mean response time and variance over the entire useful range. Daniel T. Joyce [3] in his article "An Investigation of Disk Scheduling Algorithms Laboratory" discussed the behaviour of disk scheduling algorithms by using a simulation program. The program is used to generate data that reflects the performance of the FCFS and SSTF algorithms under a variety of conditions. For each algorithm under each situation the program simulates how the algorithm would handle the situation and calculates the expected service time b/w requests, the expected waiting time for a request and the standard deviation of these waiting times.

Toby J. Teorey and Tad B. Pinkerton [4] has discussed five well-known scheduling policies for movable head disks. These policies are compared using the performance criteria of expected seek time and expected waiting time. The variance of waiting time is introduced as another meaningful measure of performance, showing possible discrimination against individual requests. Then the choice of a utility function to measure total performance including system oriented and individual request oriented measures is described.

Helen D. Karatza [5] has discussed scheduling in a distributed system. A simulation model is used to address performance issues associated with scheduling. Three policies which combine processor and I/O scheduling are used to schedule parallel jobs for a variety of workloads.

Hu Ming [6] has discussed disk-scheduling algorithms based on both disk arm and rotational positions. Their time-resolving powers are more precise in comparison with those for disk-scheduling algorithms based only on disk arm position. For modem disks, increase of disk rotation rate makes overhead of disk access to data transfer heavier. Therefore, it seems more important to improve both parallel processing capability of disk I/O and disk-scheduling performance at the same time.

9 III.

10 PROPOSED MODEL

In this research effort, the problem under study is to optimize the performance of various disk scheduling algorithms before these are actually followed in any operating system and to design the simulator to mimic the real behaviour of the system. Because the seek distance between the position of head and position of

requesting cylinder at the time of request is the basic need for evaluating the performance of the I/O system. Thus an efficient Disk Scheduling algorithm can enhance the performance of overall system whereas a poorly design scheme can degrade the performance. Thus to study the various algorithms, simulator is designed.

A simulation of any process in which there are inherently random components requires a method of generating random numbers. Thus whenever simulator is used, as a tool for research, there is need for generating random numbers that are conveniently and efficiently generated from a desired probability distribution. The present research work uses box-muller transformation for generation of cylinder numbers.

11 S = (-2 log e (R 1))

½ * cos(2? R 2)

Here S is independent random variables with a normal distribution of standard deviation 1. In present research work, the foremost criterion for the evaluation of disk scheduling algorithms is the access time calculated by seek time, rotational delay and transfer time that are produced by each policy under same set of conditions and same workload. The workload here is the cylinder numbers whose data is to be accessed to perform I/O operation. This calculated access time is Margo Seltzer, Peter Chen and John Ousterhout [2] have jointly written a research paper "Disk Scheduling Revisited". In this paper, the invention of the movable head disk has been discussed. These techniques have been applied to systems with large memories and potentially long disk queues. Disk bandwidth utilisation can be improved by applying some traditional disk scheduling techniques, which attempt to optimise head movement and guarantee fairness in response time.

T S (seek time): time for the disk arm to move the heads to the cylinder containing the desired sector.

T R (rotational delay): time waiting for the disk to rotate the desired sector to the disk head.

T T (transfer time): the time it takes to transfer a block of bits to and from the disk. Among these three, seek time has large significant effect on the total access time of the disk. As seek time is the time relating to cylinder number. Therefore cylinder number and number of seek movements are central point of consideration.

Simulator for Optimizing the Comparative Performance of Disk Scheduling Algorithms N : no. of cylinders NODE : current position of moveable read/write head R 1 /R 2 :

independent random variables in the interval [0, 1] T S (i) : seek time of N cylinders T A (i) : access time of N cylinders T R : rotational speed of the disk T T : transfer time between adjacent cylinders RUNS : no. of times the simulation process is repeated RAND : random number L_TIME : latency time to move the head from one to another cylinder CL[i] : left requests with respect to head position. CR[j] : right requests with respect to head position. Algorithm to compute the access time to read/write a disk Step 1. Read no. of cylinders for different workload.

Step 2. Generate random numbers using the random number generation method in the interval of [0, 1].

IV.

12 RESULTS

The best way to compare the result of different algorithms is to present them in form of table depicting the result in the form of rows and columns. Different test cases are simulated by varying the number of randomly generated cylinders and accordingly results are shown as in Table 1/Table **??** Table **??** : Total head movement for heavy load (No. of cylinders: 1200)

V.

13 DISCUSSION AND CONCLUSION

After analysing the results and findings of the simulator, it might be concluded no single policy is best in all situations. The performance do not depend upon only on the number of requests but it also depends on the position of read/write head & direction of the movement of head and it varies with the variation in number of requests even the current head position is same. It has been also observed that if there is only one outstanding request, then all the policies behave the same. [1] [2] [3] [4]

[1] Sim_Dsc: Simulator for Optimizing the Performance of Disk Scheduling Algorithms

[2] © 2011 Global Journals Inc. (US) Global Journal of Computer Science and Technology Volume XI Issue XVIII Version I 4 2011 October Sim_Dsc: Simulator for Optimizing the Performance of Disk Scheduling Algorithms

[3] © 2011 Global Journals Inc. (US) Global Journal of Computer Science and Technology Volume XI Issue XVIII Version I 6

[4] OctoberSim_Dsc: Simulator for Optimizing the Performance of Disk Scheduling Algorithms

2011

Figure 1: A © 2011

1

Test case 1: No. of cylinders (Low Laod) =200
Test case 2: No. of cylinders (Medium Laod) =700
Test case 3: No. of cylinders (Heavy Laod) =1200
Test Case 1: It is shown in the table 1 regarding
total head movement of different disk scheduling
algorithms in the case of low load on various simulation
runs.

Figure 2: Table 1 :

FCFS policy can be considered best for the system, which has fewer loads of Input-output requests, but in heavy load of requests, FCFS tends to saturate the device. SSTF produced least number of head movement in maximum runs as compared to FCFS. Therefore this policy is the optimal policy. But this policy can not be considered optimal as this policy has the starvation problem. LOOK has no starvation problem. But this policy has the overhead of decision variable, which is used to decide the direction (inward or outward) of read/write head. LOOK (Down) algorithm is always better than as compared to LOOK (UP) algorithm. C-Look disk scheduling algorithm performs better for those systems which puts medium and heavy load of requests on the disk. The graph 1 depicts the head movement for different number of simulation runs for FCFS algorithm under various loads.

.1 Graph No. 1

The graph 2 depicts the head movement for different number of simulation runs for SSTF algorithm under various loads.

.2 Graph No. 2

The graph 3 depicts the head movement for different number of simulation runs for SCAN algorithm under various loads.

[Teorey and Pinkerton (1972)] *A comparative analysis of disk scheduling policies*, Toby J Teorey , Tad B Pinkerton . March 1972. New York, NY, USA.

[Karatza ()] *A Comparative Analysis of Scheduling Policies in A Distributed System Using Simulation*, Helen D Karatza . 2000. Thessaloniki, Greece.

[Robbins (2004)] *A Disk Head Scheduling Simulator*, Steven Robbins . March, 2004. Norfolk, Virginia, USA.

[Dietal ()] *An Introduction to Operating Systems*, H M Dietal . 1984. Addition-Wesley. (Rev. 1 st Edition Reading)

[Joyce ()] *An Investigation of Disk Scheduling Algorithms Laboratory*, Daniel T Joyce . 2001.

[David et al. (1995)] *Disk scheduling algorithms based on rotational position*, M David , John Jacobson , Wilkes . May 1995. Hewlett Packard.

[Seltzer et al. (1990)] 'Disk Scheduling Revisited'. Margo Seltzer , Peter Chen , John Ousterhout . *Winter Usenix*, (Washington) January 1990.

[Hu (2005)] *Improved disk scheduling algorithms based on rotational position*, Ming Hu . October, 2005.

[Silberschatz and Galvin ()] *Operating System Concepts*, A Silberschatz , P B Galvin . 2001. (6 th Edition)

[Muhammad Younus Javed and Ihsan Llah Khan ()] *Simulation and performance comparison of four disk scheduling algorithms*, Muhammad Younus Javed , Ihsan Llah Khan . 2000. IEEE.

[Simulation Runs FCFS SSTF SCAN C -SCAN LOOK (UP) LOOK (DN) C -LOOK (1000 25629)] *Simulation Runs FCFS SSTF SCAN C -SCAN LOOK (UP) LOOK (DN) C -LOOK*, 1000 25629.

[Deo and Narsingh ()] *System Simulation with Digital Computer*, Deo , Narsingh . 2002. New Delhi, India. (15 th edition, PHI)

[Table 2 : Total head movement for medium load] *Table 2 : Total head movement for medium load*, p. 700.

[Test Case 3: It is shown in the table 3 regarding total head movement of different disk scheduling algorithms in the case of heavy
Test Case 3: It is shown in the table 3 regarding total head movement of different disk scheduling algorithms in the case of heavy load on various simulation runs,