Global Journal of Computer Science and Technology
Volume 11 Issue 18 Version 1.0 October 2011
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Sim_Dsc: Simulator for Optimizing the Performance of Disk Scheduling Algorithms

By P.K. Suri, Sumit Mittal
Kurukshetra University

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.

Keywords : disk scheduling algorithms, seek time, rotational delay, transfer time, access time, head movement, box-muller transformation.

GJCST-A Classification : F.2.1,G.1.6

SIMDSCSIMULATOR FOR OPTIMIZING THE PERFORMANCE OF DISK SCHEDULING ALGORITHMS

Strictly as per the compliance and regulations of:

Sim_Dsc: Simulator for Optimizing the Performance of Disk Scheduling Algorithms

P.K. Suri [α], Sumit Mittal [Ω]

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.

Keywords : disk scheduling algorithms, seek time, rotational delay, transfer time, access time, head movement, box-muller transformation.

I. Introduction

Among 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.

Author [α] : Department of Computer Science & Applications, Kurukshetra University, Kurukshetra, Haryana 136118, India.
Author [Ω] : M.M. Institute of Computer Technology & Business Management, M.M. University, Mullana, Ambala, Haryana, 133207, India. E-mail : sumit_amb@yahoo.com

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.

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.

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.

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.

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.

e) Look (Up/Down) algorithm

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.

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.

II. 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.

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.

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.

III. 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.

Suppose R_1 and R_2 are independent random variables that are uniformly distributed in the interval [0, 1].

$$S = (-2 \log_e(R_1))^{½} * \cos(2\pi R_2)$$

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 used to find out the total head movement for various disk scheduling algorithms.

$$T_A = Ts + T_R + T_T$$

Where

T_A (access time): sum of seek time, rotational latency time and transfer 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 : two 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].
Step 3. Compute the mean and standard deviation of m-pseudo random numbers.
Step 4. Apply Box-Muller transformation to calculate the value of S, using two random variates between [0, 1].
Step 5. Using the values of mean, standard deviation and S, calculate the value of x and store in an array x[i], which can use as the number of requests.
Step 6. Call modules for all seven policies named FCFS (), SSTF (), SCAN (), C-SCAN (), LOOK UP (), LOOK DOWN () and C-LOOK ().
Step 7. Compute access time based on seek time, rotational delay and transfer time produced by each policy is returned to the main module.
Step 8. Each algorithm is run for 20000 times and result of every 1000th run of each algorithm is displayed in a table.
Step 9. Stop

IV. 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 2/Table 3.

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.

Simulation Runs	FCFS	SSTF	SCAN	C-SCAN	LOOK (UP)	LOOK (DN)	C-LOOK
1000	4065	574	289	376	187	107	194
2000	4677	459	2€84	325	199	146	229
3000	4629	1077	293	410	199	119	211
4000	3867	479	281	361	182	121	201
5000	4328	415	299	396	226	155	252
6000	4253	536	285	369	184	113	197
7000	4133	586	282	310	187	128	208
8000	4095	530	290	378	194	118	206
9000	4372	612	282	456	180	114	193
10000	4604	448	293	385	208	137	229
11000	4260	426	302	402	218	130	230
12000	4492	558	278	355	184	134	211
13000	4438	450	281	379	183	123	203
14000	3837	403	278	355	171	108	185
15000	4713	517	290	402	203	136	225
16000	4130	539	290	379	198	126	215
17000	4690	444	298	395	204	114	211
18000	4139	481	293	326	200	121	212
19000	4580	548	298	393	222	150	245
20000	4518	482	292	382	199	122	212

Table 1 : Total head movement for low load (No. of cylinders: 200)

Global Journal of Computer Science and Technology Volume XI Issue XVIII Version I October 2011

Test Case 2: It is shown in the table 2 regarding total head movement of different disk scheduling algorithms in the case of medium load on various simulation runs.

Simulation Runs	FCFS	SSTF	SCAN	C-SCAN	LOOK (UP)	LOOK (DN)	C-LOOK
1000	15730	1057	287	1297	199	137	223
2000	15520	1069	290	1359	215	160	249
3000	14302	947	299	1415	213	129	227
4000	15615	976	285	1325	208	161	245
5000	15438	1210	292	1427	205	134	225
6000	15253	1026	296	1382	215	142	237
7000	15683	1106	294	1350	231	180	273
8000	14991	1117	297	1402	233	175	271
9000	15959	1132	304	1372	238	164	267
10000	15072	1043	289	1415	221	175	263
11000	14662	1098	293	1318	210	141	233
12000	14926	1128	300	1365	233	166	265
13000	14034	1057	288	1380	200	136	223
14000	16468	1026	297	1426	220	149	245
15000	15466	1100	289	1402	206	145	233
16000	15024	1178	290	1379	201	132	221
17000	15252	1065	284	1424	205	158	241
18000	14442	1106	286	1408	198	138	223
19000	15238	1352	291	1392	211	149	239
20000	15617	1094	289	1310	206	145	233

Table 2 : Total head movement for medium load(No. of cylinders: 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 load on various simulation runs.

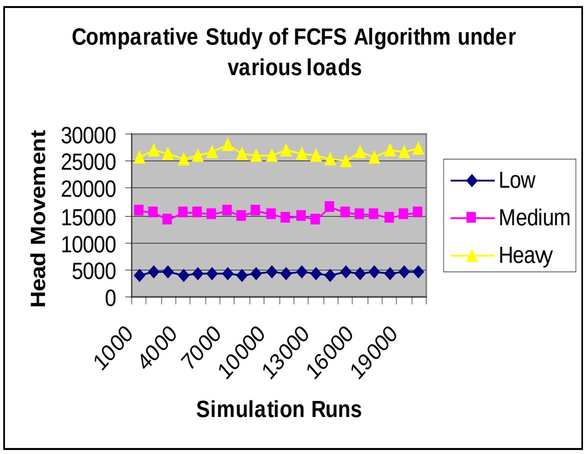
Simulation Runs	FCFS	SSTF	SCAN	C-SCAN	LOOK (UP)	LOOK (DN)	C-LOOK
1000	25629	1528	289	2397	209	151	239
2000	27118	1664	301	2382	240	177	277
3000	26256	1728	300	2356	223	146	245
4000	25234	1802	294	2326	228	174	267
5000	25969	1663	292	2415	215	154	245
6000	26546	1652	302	2340	233	160	261
7000	27861	1590	293	2502	224	169	261
8000	26404	1584	298	2448	228	162	259
9000	26019	1608	299	2397	229	161	259
10000	26055	1568	293	2415	215	151	243
11000	26978	1776	309	2345	242	157	265
12000	26299	1595	291	2300	210	147	237
13000	25891	1760	297	2417	222	153	249
14000	25360	1556	300	2397	233	166	265
15000	25035	1636	291	2396	226	179	268
16000	26601	1658	303	2318	248	187	289
17000	25792	1555	294	2368	217	152	245
18000	26916	1530	310	2420	250	170	279
19000	26671	1707	294	2382	213	144	237
20000	27463	1865	290	2392	212	154	243

Table 3 : Total head movement for heavy load (No. of cylinders: 1200)

V. Discussion and Conclusion

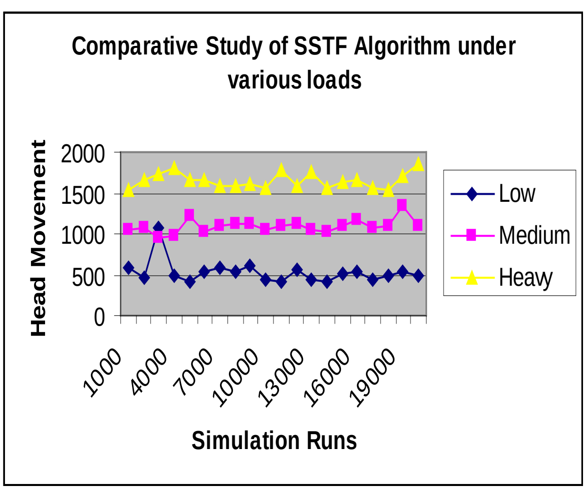
After analysing the results and findings of the simulator, it might be concluded no single policy is best in all situations. The performance do not depend upon only on the number of requests but it also depends on the position of read/write head & direction of the movement of head and it varies with 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.

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.



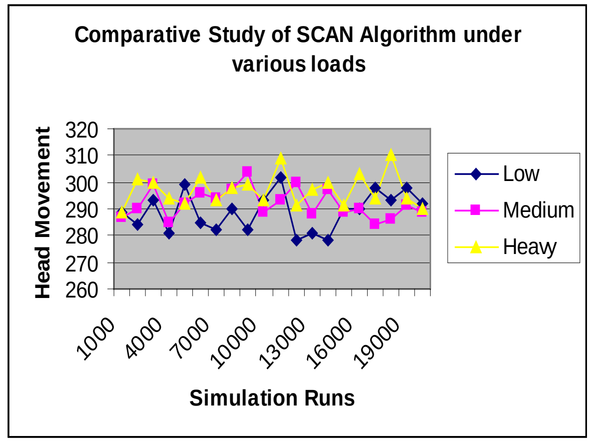
Graph No. 1

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



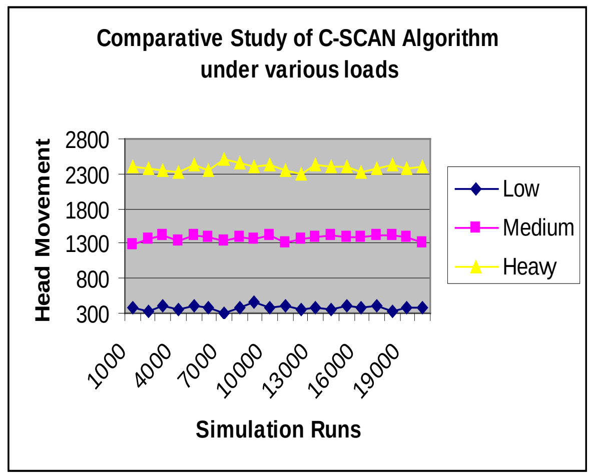
Graph No. 2

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



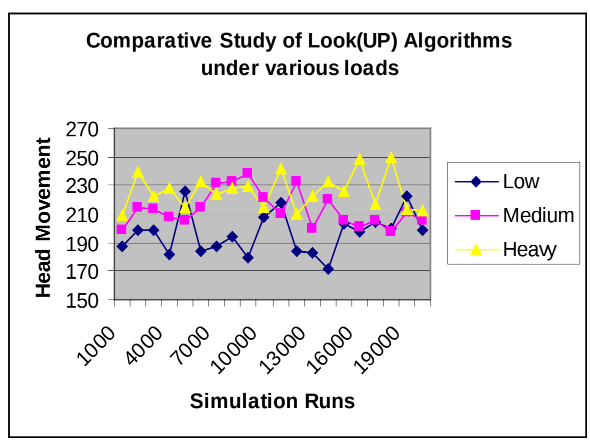
Graph No. 3

The graph 4 depicts the head movement for different number of simulation runs for C-SCAN algorithm under various loads.



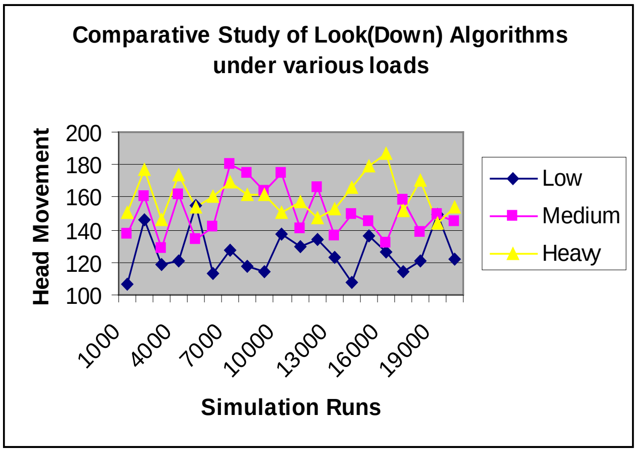
Graph No. 4

The graph 5 depicts the head movement for different number of simulation runs for Look (UP) algorithm under various loads.



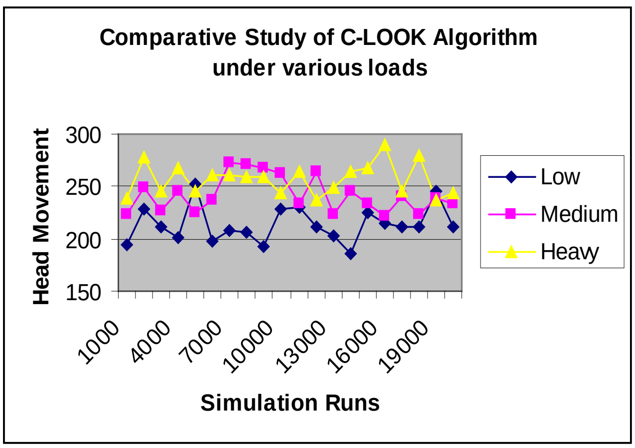
Graph No. 5

The graph 6 depicts the head movement for different number of simulation runs for Look (Down) algorithm under various loads.



Graph No. 6

The graph 7 depicts the head movement for different number of simulation runs for C-Look algorithm under various loads.



Graph No. 7

References References Referencias

1. David M. Jacobson and John Wilkes, "Disk scheduling algorithms based on rotational position" Hewlett Packard, May 1995.
2. Margo Seltzer, Peter Chen and John Ousterhout, "Disk Scheduling Revisited", Winter Usenix, Washington, January 1990.
3. Daniel T. Joyce, "An Investigation of Disk Scheduling Algorithms Laboratory", 2001.
4. Toby J. Teorey and Tad B. Pinkerton, "A comparative analysis of disk scheduling policies", March 1972, New York, NY, USA.
5. Helen D. Karatza," A Comparative Analysis of Scheduling Policies in A Distributed System Using Simulation", Thessaloniki, Greece, 2000.
6. Hu Ming, "Improved disk scheduling algorithms based on rotational position", October, 2005.
7. Silberschatz A.,P.B.Galvin et. al., "Operating System Concepts", 6th Edition, 2001.
8. Muhammad Younus Javed, Ihsan Ilah Khan, "Simulation and performance comparison of four disk scheduling algorithms", IEEE, 2000.
9. Deo. Narsingh, "System Simulation with Digital Computer", 15th edition, PHI, New Delhi, India, 2002.
10. Dietal H.M., "An Introduction to Operating Systems", Rev. 1st Edition Reading, Addition-Wesley, 1984.
11. Steven Robbins, "A Disk Head Scheduling Simulator", Norfolk, Virginia, USA, March, 2004.