Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 8, August 2015, pg.364 – 368

RESEARCH ARTICLE

Demand Based Disk Scheduling Using Circular Look Algorithm

Sarita Negi[1], Kulvinder Singh[2], Shubham Sahu[3]
[1]Department of Computer Science, Uttrakhand Technical University, India
[2]Department of Computer Science, Uttrakhand Technical University, India
[3]Department of Computer Science, Uttrakhand Technical University, India
[1] sarita.sharma2612@yahoo.com, [2] kulvinder.taak@yahoo.com, [3] kumarshubhamsahu@gmail.com

ABSTRACT- This paper focus on the improved I/O performance by intelligent scheduling of disk accesses by the movable disk head. Since the beginning of the era of computers, the processor speed and memory capacity have increased continuously and drastically as compared to the disk speed. Although several experiments have been conducted on disk scheduling to improve seek time but they hardly proved. We have proposed and proved a new disk scheduling algorithm using look scan that reduces the number of head movements which result in reduced seek time. This can be used to maximize throughput for modern storage devices and processor speed versus disk access speed mismatch is reduced substantially.

Keywords- Look Scheduling, α-request, FCFS, SSTF, THM, C-Look scheduling

I. INTRODUCTION

DISK SCHEDULING- The technique that the operating system uses to determine which requests to satisfy first is called disk scheduling. The Scheduling is used to divide the CPU time between the processes so that the processes can execute concurrently at a single time. Disk Scheduling specifies that at which time which Process will be executed by the CPU. A disk scheduler's objective is to optimize average access time. There are three time factors involved in a disk access: the seek time, the rotational latency time and the data transfer time.

- Seek time is the time required to move disk arm to the track. It is defined as the time required to reach read/write head to desired track from its current position.
- Latency time is the time it takes for the beginning of the required sector to reach the head.
- Data transfer time is the time taken to data transfer

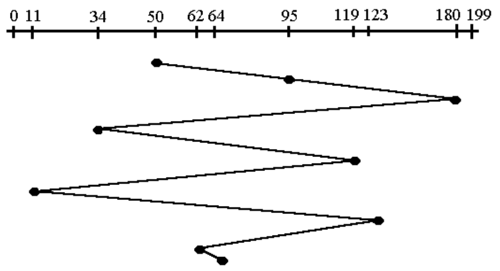
There are two objectives of any disk scheduling algorithm

1. Minimize the throughput.
2. Maximize the response time.

II. First Come First Serve (FCFS):

In this scheduling processes are executed in the same manner in which they are entered into the system. In this scheduling operating system creates a queue which contains the sequence order in which the CPU will execute the processes. In this the job which had requested first will be executed first by the CPU and the Jobs those are entered later will be executed according to the order in which they are entered.

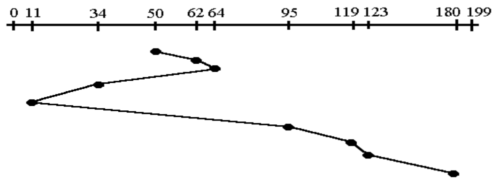
In FCFS scheduling, all request are treated equally i.e. average response time will be same for all request.. FCFS is a very simple algorithm and is easy to implement.



In this example, it had a total head movement of 640 tracks.

III. Shortest Seek Time First (SSTF)

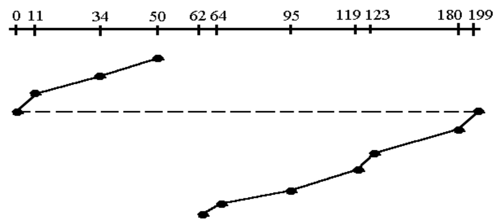
The shortest seek first algorithm determines which request is closest to the current position of the head, and then services that request next. SSTF picks the request from the queue closest to the current read/write head location. It scans down towards the nearest end and when it reached the bottom it scans up servicing the requests that it did not get going down. If a request comes in after it has been scanned, it will not be serviced until the process comes back down or moves back up. SSTF suffers from starvation problem: some request can wait much longer time than others.
Starvation occurs when requests continuously pour into the neighborhood of the current head position while some requests are waiting at other end.



The total head movement for this algorithm is only 236 tracks

IV. C-Scan Scheduling

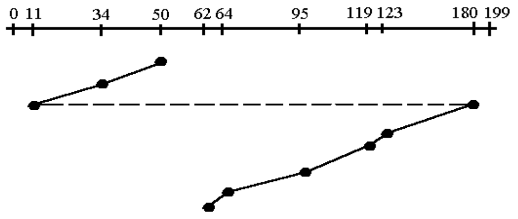
In C scan scheduling the head moves from one end of the disk to the other end, servicing request along the way.. Once the head reaches the other end, it immediately returns to the beginning of the disk without servicing any request on the return time.. Keep in mind that the huge jump doesn't count as a head movement. It is also called as elevator algorithm because of its works same as the elevator works. In C-scan, the service is more uniform. This scheduling does not suffer from starvations. This algorithm perform better for system that place heavy load on the disk .



The total head movement for this algorithm is only 187 tracks

V. Look Scheduling

LOOK behaves almost identically to SSTF, but avoids the starvation problem of SSTF. The **LOOK** algorithm is similar as the **SCAN** algorithm in that it also honors requests on both sweep direction of the disk head, however, this algorithm "Looks" ahead to see if there are any requests pending in the direction of head movement. If no requests are pending in the direction of head movement, then the disk head traversal will be reversed to the opposite direction and requests on the other direction can be served. In LOOK scheduling, the arm goes only as far as final requests in each direction and then reverses direction without going all the way to the end.



The total head movement for this algorithm is only 157 tracks

VI. Demand Based Disk Scheduling Using Circular Look Algorithm

This algorithm mainly based on the number of requests i.e. the total number of address requests on the either side of the current head position.
The address is arranged in ascending order from left to right direction. Left being the starting address and right being the ending address.

Lower Address $\xrightarrow{\hspace{6cm}}$ Higher Address

First we sort the requested address in ascending order using any type of sorting algorithm like quick sort or merge sort etc. and then we check the current head position. The concept of tracking the position of disk head is to find out the majority of request on either side of the current head position. We check both the sides of the head position.
If there are more request on the left side of the head than we start scanning to the left.
And if number of requests is more on the right side then we start scanning towards right of the head.
Particularly two types of data is collected and maintained

1. The lowest number of address requests
2. The highest number of address requests

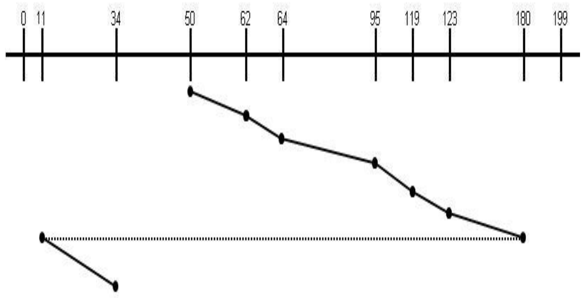
Once the direction of the sweep is decided by comparing the current head position, the following steps are followed

1. Leftmost and right most address of the platter is considered/used. It means the head is only moved as far as the last request in each direction is serviced.
2. Now the number of requests on left and right side of disk head are compared.
3. If number of request are higher on right side, we start sweeping towards right and fulfill all the address requests till the rightmost address request and at this position we trigger α-request.
   α-request- In this request we make use of data stored i.e. we request the CPU to move the head to the leftmost address from the rightmost address request or vice versa. The sweeping process is totally inactive during the α-request is performed.

The motive is to reach from the rightmost to the leftmost address request while the sweeping is inactive.

4. Now that we have our head on the leftmost address position, CPU again sweeps the remaining requests until all the address requests are fulfilled.
   *The sweeping process is paused during the α-request is processes.

In this example number of requesting locations on the right side of the current position of disk head is greater than the number of requesting locations on the left side.
i.e. 6>2 therefore the disk head starts sweeping towards right side until it service the last requesting location.
Head movement from head current location to rightmost data=130 tracks
At rightmost data α-request is enabled and in this duration CPU is inactive hence no head movement is counted.
When the disk head arrives leftmost data α-request is disabled and CPU becomes active. Now the disk head again moves towards right and service remaining data requests.
Head movement from leftmost data to last remaining request=23
Total head movement in this example =130+23=153, which is the smallest track coverage as compared to the earlier works in disk scheduling.



The total head movement for this algorithm is only 153 tracks

Algorithm

```
1. Merge sort[address[],req]
2. if(a<=b)
        • Starts scan towards right till largest value.
        • Make α-switch.
```

- Continue scan to the right and covers the rest.

3. else

- Start scan to the left till small value.
- Make α-switch.

Continue scan to the left and covers the rest.

Where:

- ✓ a is number of service requests on the left side of disk head.
- ✓ b is number of service requests on the right side of disk head.

VII. CONCLUSION

We have proposed and proved a highly optimized disk scheduling algorithm using demand based circular look scan method which has least number of head movements and least seek time, used in design of operating systems till date. By using this algorithm less process will suffer from starvation as this algorithm always choose the direction which has more number of requesting locations. This algorithm is supported with a real time experiment, of which different values/outputs are shown in the paper. Unlike previous research in disk scheduling optimization this innovation is easy to implement, supported with an example and operating system developed using this technique can give high throughput. This paper also shows a path that disk scheduling techniques are still required to be evolved.

REFERENCES

[1] "*ComputerNotes*" http://ecomputernotes.com/fundamental/disk-operating-system/what-is-disk-scheduling-type-of-disk-scheduling
[2] http://faculty.salina.kstate.edu/tim/ossg/File_sys/disk_scheduling.html
[3] John Ryan Celis, Dennis Gonzales, Erwinaldgeriko Lagda and Larry Rutaquio Jr. "*A Comprehensive Review for Disk Scheduling Algorithms"*
[4] Manish, K. M., *"An Improved First Come First Serve(IFCFS) Disk Scheduling Algorithm*", Volume 47– No.13,June 2012, pp. 20-24
[5] Gisela May A. Albano, and Angelito G. Pastrana, *Fundamentals of Operating System*, A & C Printers, 2009
[6] N. Evren Asan, *Offline and Online Disk Scheduling Problems*, December 2006
[7] Margo Seltzer, Peter Chen, John Ousterhout, *Disk Scheduling Revisited*, USENIX '90, pp. 313-324
[8] llinois Institute of Technology, College of Science, http://www.cs.iit.edu/~cs561/cs450/disksched/disksched.html
[9] W. Stallings, "*Operating Systems*", 4th Edn., Pearson Education Pte. Ltd., 2007, ISBN 81- 7808-503-8.
[10] C. Staelin, G. Amir, D. B. Ovadia, R. Dagan, M. Melamed and D. Staas, " *Real-time disk scheduling algorithm allowing concurrent I/O requests*", HP Laboratories, HPL-2009-344.
[11] A. L. N. Reddy, Jim Wyllie and K. B. R. Wijayaratne, "*Disk Scheduling in a Multimedia I/O System" ACM Transactions on Multimedia Computing, Communications and Applications*" Vol. 1, No. 1, Feb 2005, pp. 37-59
[12] A. Silberschatz, P. B. Galvin, and G. Gagne, "*Operating System Concepts*", 7th Edn., John Wiley and Sons Inc, 2005,ISBN 0-471-69466-5.
[13] http://www4.comp.polyu.edu.hk/~csajaykr/myhome/teaching/eel358/ds.pdf
[14] http://people.cs.umass.edu/~trekp/csc262/lectures/02b.pdf
[15] http://www.ecs.umass.edu/ece/koren/architecture/Disk/help/htm
[16] Sandipon Saha, Md. Nasim Akhter, Mohammod Abul Kashem *" A New Heuristic Disk Scheduling Algorithm*" international journal of scientific & technology research volume 2, issue 1, january 2013.