Journal of Computer and Communications, **2014, 2, 1-5**
Published Online January 2014 (http://www.scirp.org/journal/jcc)
http://dx.doi.org/10.4236/jcc.2014.21001

Scientific Research

A New Approach to Disk Scheduling Using Fuzzy Logic

Priya Hooda*, Supriya Raheja

Department of Computer Science and Engineering, ITM University, Gurgaon, India.
Email: *priya.26hooda@gmail.com

Received October 31st, 2013; revised November 25th, 2013; accepted December 3rd, 2013

ABSTRACT

Disk scheduling is one of the main responsibilities of Operating System. OS manages hard disk to provide best access time. All major Disk scheduling algorithms incorporate seek time as the only factor for disk scheduling. The second factor rotational delay is ignored by the existing algorithms. This research paper considers both factors, Seek Time and Rotational Delay to schedule the disk. Our algorithm Fuzzy Disk Scheduling (FDS) looks at the uncertainty associated with scheduling incorporating the two factors. Keeping in view a Fuzzy inference system using If-Then rules is designed to optimize the overall performance of disk drives. Finally we compared the FDS with the other scheduling algorithms.

KEYWORDS

Operating System; Hard Disk; Disk Scheduling; Fuzzy Logic; Fuzzy Inference System (FIS)

1. Introduction

Operating System provides an interface between user and hardware. One of the major responsibilities of OS is to use the disk drive efficiently and to provide the fast access time. Multiprogramming environment for controlling and providing memory to process the concept of Disk Scheduling is used.

When the disk starts operating, the disk operates at a constant speed. To read or write, the head must be positioned at the desired track and at the beginning of the desired sector on that track [1].

The seek time is the time for the disk arm to move the heads to the cylinder containing the desired sector [2]. Seek time is a very important factor for any operating system. Different disk scheduling algorithms are used to reduce the seek time, namely, First Come First Serve (FCFS), Shortest Seek Time First (SSTF), Scan, C-Scan, Look, and C-Look.

Another main component of Access Time is Rotational Delay. After the read/write head seeks to the desired track, it has to rotate to the desired sector from which data read or write operations are performed. This time consumed in rotation after seeking is called as Rotational Delay.

Various work is available related to reduce the seek time using different disk scheduling algorithms. But they are not providing any solution to handle the uncertainty and starvation problem associated with disk scheduling.

For example, supposing that there are two requests with the same distance, one request is near the spindle and one towards the outer track and there is a need to decide which request to process first. The reason to use fuzzy logic is to handle such kind of uncertainty, so that we can choose the right request to serve. In this paper, we are implementing a new algorithm, Fuzzy Disk Scheduling (FDS) for disk scheduling algorithm using fuzzy logic. In our work we gave emphasis on both the factors discussed above to improve access time. Moreover, our algorithm will further reduce the seek time.

This paper is divided into 8 sections. Section 2 describes the related work done so far. Section 3 includes a brief discussion on disk scheduling. Fuzzy set theory is discussed in Section 4. Section 5 comprises problem definition. Section 6 contains the designed FIS. Section 7

*Corresponding author.

comprises proposed algorithm. Finally Section 8 is Conclusion.

2. Related Work

In the recent years many researchers have came forward with their work and ideas of improving the disk performance. Margo Seltzer, 1990, wrote a research paper "Disk Scheduling Revisited". In this paper they discussed the disk scheduling technique based on sectors traversed and rotational delay experienced by movable head system disk drives.

David M. Jacobson, 1995, discussed disk scheduling algorithm based on rotational position [3]. Disk scheduling based on rotational position as well as disk arm position is shown to provide improved performance [4]. They discovered Aged Shortest Access Time First algorithm which is a continuum between FCFS and SSTF.

Another approach was proposed by Hu Ming, 2005, in his research paper based on the idea of disk arm and rotational position. Modern disk drives emphasise on seek time reduction, but increase in disk rotation leads to higher data transfer time [5].

Quality of Service Aware disk scheduling algorithm was proposed by Walid G. Aref, 2002. It is applicable in environments where data requests arrive with different QoS requirements such as real-time deadline, and user priority [6].

Mohammad Sofian Abu Talip, 2009, in his revolutionary research paper discussed the implementation of fuzzy disk scheduling algorithm using Fuzzy Inference System and IF-THEN Rules. This paper uses seek time and arrival time as two factors as input to the fuzzy inference engine [1].

3. Disk Scheduling

Since in operating system all device requests are linked in queues, if seek time increases the system becomes slow. A disk scheduling algorithm is used to reduce total access time of disk requests. In this section we are briefly discussing different types of disk scheduling algorithms.

First Come First Serve Disk Scheduling Algorithm (FCFS) is easy and simple disk scheduling algorithm to implement. This algorithm is implemented using FIFO queue [7]. The requests are executed in the same sequence as they enter in the queue. However, FCFS disk scheduling is simplest one but doesn't provide good results. It's easy to calculate the number of head movements in FCFS but with increased seek time.

Shortest Seek Time First (SSTF), the requests are executed in a rearranged order based on the shortest next cylinder request distance. SSTF services the request that is closest to the current read/write head location. This algorithm minimizes the seek time, but may cause starvation. Sometimes the high priority request with more head movements may get starved with the lower priority request that is close to the head. SSTF dominates disk activity and increases the latency for other processes tracks. It is unfair type of disk scheduling algorithm because of higher variance of service time which is not acceptable.

SCAN approach was introduced to overcome the problem of high latency caused by SSTF. This approach is also called as ELIVATOR disk scheduling. This algorithm services a request that results in shortest next distance request but in a preferred direction. The read/write head scans down in that direction till end and when it reaches the bottom of that direction and then the head is reversed and it scans up servicing in the opposite direction till the second end of the disk. SCAN is more optimal but not the best one.

Cyclic or Circular SCAN, to some extent works same as SCAN works. The only difference, it is one directional scan. It services all disk requests from innermost to outermost cylinders or vice-versa. It starts its scan toward the nearest end and services the requests all the way to the end. Once it reaches the end the read/write head jumps to the other end. The jump made by the head is not counted as head movement. C-SCAN has same utilization as of SSTF [8].

In LOOK disk scheduling, head moves back and forth servicing requests just like SCAN algorithm. But unlike SCAN, the head doesn't reaches to last cylinder at the end of the disk; the head will change its direction after servicing the last request in the current direction. Recognition of the dynamic request of the queue leads to the look algorithm [9]. As compared on the basis of seek time LOOK has better seek time average than SCAN.

CIRCULAR-LOOK disk scheduling algorithm is best disk scheduling algorithm so far. C-LOOK treats the disk in manner as the last track is adjacent to first track. In C-LOOK the read/write head services requests in one direction till last request in that direction and changes its direction and jumps to the last request without servicing any requests, and then starts servicing requests again.

4. Background on Fuzzy Sets and FIS

Fuzzy logic is a way to map input space to output space. It maps the human thinking in programming. A fuzzy set is a class of objects with a continuum of grades of membership [10]. It extends conventional Boolean Logic to deal with ambiguity and uncertainty. A fuzzy set provides a notion to represent crisp values to fuzzy values by assigning them between 0 and 1.

The basic elements of fuzzy logic related to fuzzy sets are linguistic variables, membership functions, and fuzzy rules. The linguistic variables' values are words-specifically, adjectives such as small, little, medium and so on [11]. The membership function of a fuzzy set works as an

indicator function of crisp set. Membership function assigns values from 0 to infinity, to decimal values between 0 and 1.

Fuzzy rules are the basis of fuzzy reasoning consisting of IF-THEN rules, expressed in the form: "IF input IS from set THEN output" and are evaluated without ELSE. Using the fuzzy rules the human or expert knowledge is represented in natural language. A Fuzzy Inference System is composed of four elements namely, a Rule Base, Inference Engine Module, a Fuzzification Interface, and a Defuzzification Interface [12].

The crisp input is mapped to fuzzy values using membership functions in fuzzification module and passed to Inference Module. Where based on the fuzzy IF-THEN rules fuzzy input is converted into fuzzy output. Fuzzy output values are again transformed into crisp values. The Defuzzification process involves some methods for the purpose namely, Centroid method, Mean of Maximum, Smallest of Maximum, Largest of maximum etc. The different types of FIS available are MAMDANI, TAKAGI-SUGENO-KANG and TSUKAMOTO FIS. In our work we have used MAMDANI type FIS, because of its simple structure of "min-max" operations. It is intuitive in nature, has widespread acceptance, and is perfect to human input [13].

5. Problem Definition

In order to provide an efficient disk scheduling algorithm we are using Fuzzy Logic to map two inputs to a single output. The FIS used is Mamdani based system. Given a set of disk requests, this FIS takes Sectors Traversed and Rotational Delay as inputs and based on If-Then Rules defined the output is Optimized Priority. We use this Priority to reorder the given read/write requests in order to provide optimum results for overall Access Time.

6. Designed FIS

In our work we have designed the Mamdani type fuzzy inference system named "Disk Scheduling" using the Mat Lab Fuzzy Tool Box as shown in **Figure 1**. The designed system consisting of two inputs namely, Sectors Traversed and Rotational Delay, and one output named Priority.

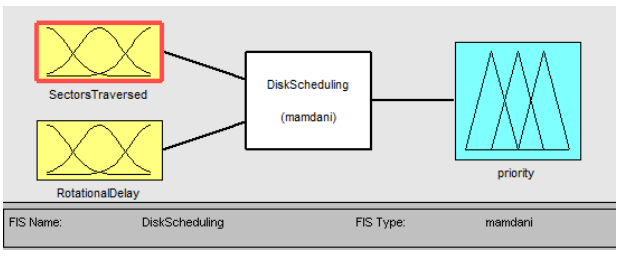

Figure 1. FIS for disk scheduling.

Sectors Traversed defines the difference from current read/write head position to next request position and Rotational Delay defines the time for the disk to rotate to the desired sector to start reading the first byte.

Based on the set of rule the values in the input vector are mapped to output vector. In our designed FIS, the output is optimized priority that depends on the above defined inputs. Here, the triangular type membership functions for all three linguistic variables are used. The Defuzzification method used is Centroid of Area.

For the input "Sectors Traversed", three membership functions are defined: low, medium, and high, ranging from 0 to 199. The input membership functions are shown in **Figure 2**.

In similar way, three membership functions are defined for "Rotational Delay": small, average, and large ranging from 0 to 8.33. The membership functions for "Rotational Delay" are shown in **Figure 3**.

Rule Base of FIS Disk Scheduling

Rule Base is the set of rules that maps input vector to output vector. In designed FIS "Disk Scheduling" we have defined the 9 fuzzy if then rules as shown in the **Figure 4**. Based on these rules our system generates the optimized priority based on the two inputs Sectors Traversed and Rotational Delay. Based on the calculated priority allotted, the disk request will be served.

With the help of these rules we can apply different input values by moving the horizontal bar in the rule viewer window to get the output. The system itself will calculate the priority after selecting the input values.

Figure 5 represents the Surface View of the FIS. It displays the range of two inputs and allows us to examine the output surface of the FIS. Because it does not

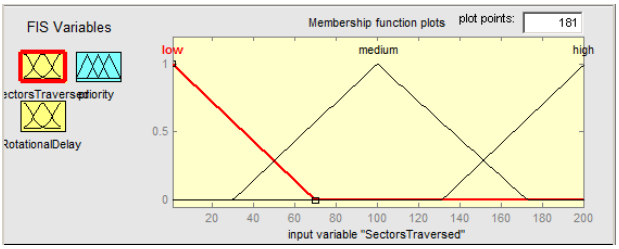

Figure 2. Input sectors traversed membership functions.

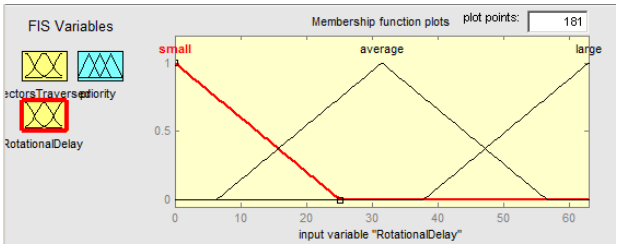

Figure 3. Input rotational delay membership functions.

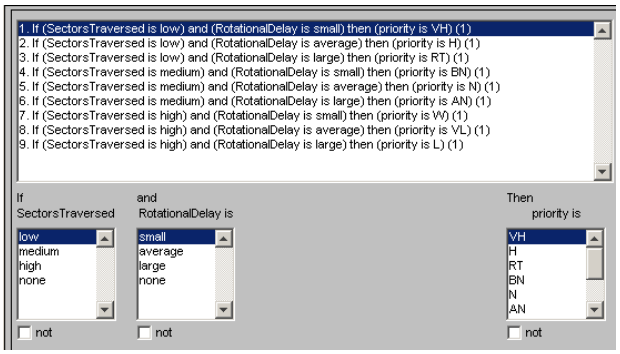

Figure 4. **Set of If-Then rules.**

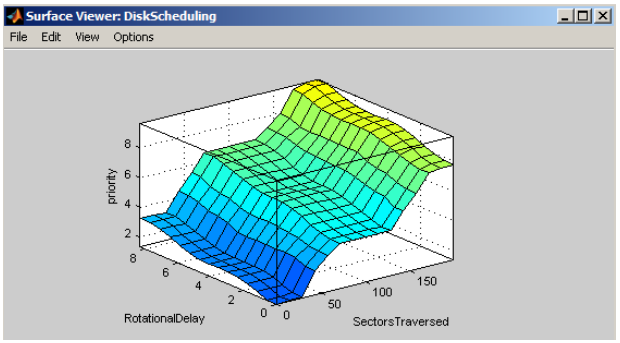

Figure 5. **Surface view of the designed FIS.**

alter the fuzzy system or its associated FIS structure in any way, Surface Viewer is a read-only editor [14].

7. Proposed Algorithm

In this section we are discussing the proposed fuzzy based Disk Scheduling algorithm.

Inputs: FDS [Sectors Traversed (0,199), Rotational Delay (0,8.33)].

Output: Priority [(1,10) (1 is highest)].

1) Calculate the estimated Seek Distance (SD) from the current head position for request in the queue using the given relation:

$$SD = |\text{Current head position-next request position}|$$

2) Calculate the Rotational Delay (RD) from the current head position for each request forming in queue using the given relation:

$$RD = 8.33 * \text{No. of Sectors causing Rotation}/63$$

3) Apply the calculated seek distance and the rotational delay to the designed FIS "Diskscheduling".

4) Find the optimized priority as "Priority" for each request.

5) Sort the queue in the descending order on the basis of priority.

6) Serve the request with highest priority in the queue.

7) After each request updates the head position and repeat steps 1 to 7 until the queue is empty.

Results

The implemented algorithm FDS is compared with other conventional algorithms and the fuzzy algorithm given by Mohamad Sofian Abu Talip [1]. A simple example is taken from William Stallings Book [15]. The order of requested sectors received by the scheduler, are 55, 58, 39, 18, 90, 160, 150, 38, and 184. The read/write head is currently located at sector 100.

Based on this set of requests sectors traversed and rotational delay is calculated. These two inputs are applied to the FIS developed, and come out with optimized priority. The requests are then rescheduled from highest priority to lowest priority. Then these requests are serviced by the operating system.

As we complete all the computations, bar graph is plotted for both the factors, *i.e*. Sectors Traversed and Rotational Delay as shown in **Figures 6** and **7**.

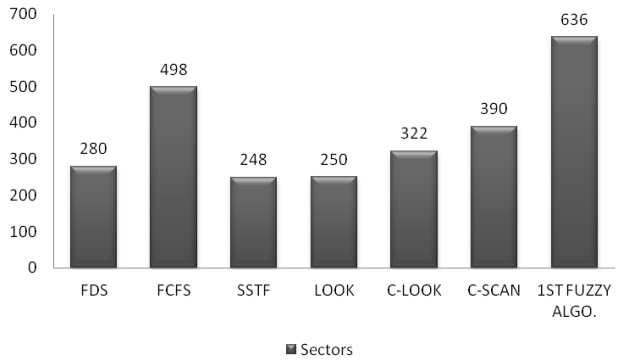

Figure 6. **Comparison of sectors traversed.**

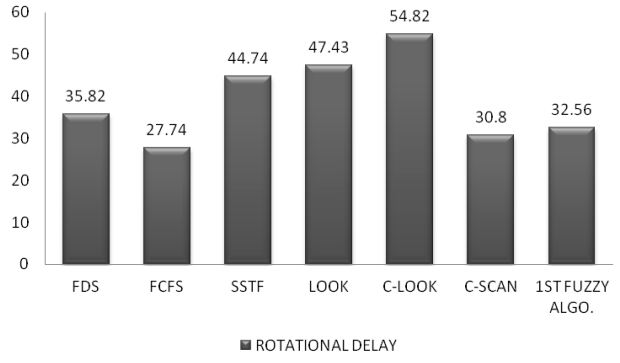

Figure 7. **Comparison of rotational delay.**

From the graphs we can show that in case of sectors traversed the FDS is better than FCFS, C-LOOK, C-SCAN and the 1st fuzzy algorithm. However FDS lacks behind SSTF & LOOK Scheduling. Moreover, in terms of Rotational Delay, this algorithm provides better results than SSTF & LOOK Scheduling.

One main advantage of this FDS algorithm is that it services all the requests fairly by taking in consideration both the factors and services all the requests without

causing starvation. Hence, it solves the starvation problem associated with SSTF Scheduling.

8. Conclusions

Fuzzy logic uses the human way to handle inexact information in machines and computers. In this paper, a new disk scheduling algorithm is proposed. To handle the inexact information we designed a fuzzy inference system "Disk scheduling" using MATLAB-Fuzzy Tool Box. The designed FIS has generated the optimum priority based on two factors, sectors traversed and rotational delay. This generated priority is used by our proposed fuzzy disk scheduling (FDS) algorithm to improve the disk access time.

It is shown using the bar graphs that there are slight deviations of FDS algorithm in terms of sectors traversed when compared with SSTF and LOOK scheduling. But in terms of rotational delay, the FDS algorithm provides much better results. The pattern of the requests after applying the algorithm shows that the read/write head doesn't stick to one area of disk. All the disk requests are serviced with the same perspective.

REFERENCES

[1] M. S. A. Talip, A. H. Abdalla, A. Asif and A. A. Aburas, "Fuzzy Logic Based Algorithm for Disk Scheduling Policy," *International Conference of Soft Computing and Pattern Recognition*, 2009, pp. 746-749. doi.ieeecomputersociety.org/10.1109/SoCPaR.2009.151

[2] S. Saha, N. Akhter and M. A. Kashem, "A New Heuristic Disk Scheduling Algorithm," *International Journal of Scientific & Technology and Research*, Vol. 2, 2013, pp. 49-53.

[3] D. M. Jacobson and J. Wilkes, "Disk Scheduling Algorithms Based on Rotational Position," *Technical Report*, 1991.

[4] P. K. Suri and S. Mittal, "Sim_Dsc: Simulator for Optimizing the Performance of Disk Scheduling Algorithms," *Global Journal of Computer Science and Technology*, Vol. 11, No. 18, 2011, pp. 1-6.

[5] M. Hu, "Improved Disk Scheduling Algorithms Based on Rotational Position," *Journal of Shanghai University*, Vol. 9, No. 5, 2005, pp. 411-414. http://dx.doi.org/10.1007/s11741-005-0024-z

[6] W. G. Aref, K. El-Bassyouni, I. Kamel and M. F. Mokbel, "Scalable QoS-Aware Disk-Scheduling," *Proceedings of the International Database Engineering and Applications Symposium*, Alberta, 17-19 July 2002, pp. 256-265.

[7] M. Y. Javed and I. U. Khan, "Simulation and Performance Comparison of Four Disk Scheduling Algorithms," *Proceedings of TENCON* 2000, Kuala Lumpur, 24-27 September 2000, pp. 10-15.

[8] M. Seltzer, P. Chen and J. Ousterhout, "Disk Scheduling Revisited," *Proceedings of the* 1990 *Winter Usenix*, Washington DC, 1990.

[9] A. Thomasian and C. Liu, "Disk Scheduling Policies with Lookahead," *ACM Sigmetrics Performance Evaluation Review*, Vol. 30, No. 2, 2002, p. 33. http://dx.doi.org/10.1145/588160.588165

[10] L. A. Zadeh, "Fuzzy Sets," *Information and Control*, Vol. 8, No. 3, 1965, pp. 338-353. http://dx.doi.org/10.1016/S0019-9958(65)90241-X

[11] S. C. Nadig, "Intelligent Scheduling Using Fuzzy Logic in Applications," *Open VMS Technical Journal*, Vol. 12, No. 12, 2009, pp. 3-7.

[12] K. M. Passino and S. Yurkovich, "Fuzzy Control: A Tutorial Introduction," *Fuzzy Control*, pp. 24-40.

[13] "Fuzzy Logic Toolbox User's Guide," 2013, p. 108. http://www.mathworks.com/help/pdf_doc/fuzzy/fuzzy.pdf

[14] "Fuzzy Logic Toolbox, Fuzzy Inference System." http://www.mathworks.in/help/fuzzy/surfview.html

[15] W. Stallings, "Operating Systems: Internal and Design Principles," 6 Edition, Prentice Hall, 2009.