International Journal Of Computer Science And Applications Vol. 1, No. 3, December 2008
ISSN 0974-1003

A Novel Disk Scheduling Approach for Real-time Database Systems

G.R.Bamnote
PRM Institute of Technology & Research, Badnera-Amravati, Maharashtra , India
+917212510215
grbamnote@rediffmail.com

Dr.M.S.Ali
PRM Institute of Technology & Research, Badnera-Amravati, Maharashtra , India
+917212578385
softalis@hotmail.com

S.Y.Amdani
Babasaheb Naik College of Engineering, Pusad, Dist Yeotmal, Maharashtra , India
+919764996786
salimamdani@yahoo.com

ABSTRACT
It has been identified that for the improvement of the performance of Real-time Database Systems all the resources should be utilized efficiently and disk is an important resource which should be concentrated. In this paper a novel disk scheduling algorithm for real-time database system is proposed. A simulator is developed which simulated the algorithms EDF, FD-SCAN, P-SCAN, SSEDO, SSEDV and the proposed algorithm and their performance is compared for Disk scheduling in Real-time Database Systems. After evaluation it was found that our proposed approach gives better performance than the existing algorithms.

Categories and Subject Descriptors
H.2.4 [Database Management]: Systems, Real-time, I.6.6
[Simulation Output Analysis]

General Terms
Algorithms, Performance, Experimentation

Keywords
Real-Time Database Systems, Disk Scheduling Algorithms.

1 INTRODUCTION
Database Management Systems are said to be complex, mission-critical software systems. Traditionally application-dependent designs were used in Real-Time Systems for data management but as the applications become more complex and amount of data increases, the code, which deals with data management, become very difficult to develop and maintain. Real-time Database Systems (RTDBS) have immerged as an alternative to manage the data with a structured and systematic approach [1,2]. In RTDBS the transactions are associated with explicit timing constraints, such as deadlines and the maximum temporal distance requirements between the accessed data objects. The correctness of a RTDBS depends upon the logical results and also upon the time at which the results are produced. Transactions in the system must be scheduled in such a way that they can be completed before their corresponding deadlines expire as well as satisfy database consistency constraints [3,4]. RTDBS have different performance goals, correctness criteria, and assumptions about the applications. The conventional database system's main objective is to provide fast response time, whereas a RTDBS may be evaluated based on how often transactions miss their deadlines, the average "tardiness" of late transactions, the cost incurred in transactions missing their deadlines, data external consistency and data temporal consistency. Data in a real-time system is managed on individual basis by every task within the system. Therefore, in various application domains, data can no longer be treated and managed on individual basis; rather it is becoming a vital resource requiring an efficient data management mechanism. [5,6]. It has been identified that for the improvement of the performance of RTDBS all the resources should be utilized efficiently. There are three types of major physical resources: the processors, the disks and main memory buffers, that should be managed effectively and efficiently even at the database level with support from underlying operating systems [7]. The main criteria in assessing the success of any scheduling policy is the success ratio i.e. the number of transactions completed successfully before their deadline. In this paper we have discussed the disk scheduling algorithms. We have simulated some of the algorithms and compared them. The organization of this paper is as follows: Section 2 gives an overview of the disk-scheduling problem. In section 3 Disk Scheduling Algorithms are discussed. In section 4, the proposed approach is described. In section 5 the experimental results and performance evaluation is presented. Finally section 7 concludes with a summary.

2 DISK SCHEDULING PROBLEM
In a disk-based database system, disk I/O occupies a major portion of transaction execution time. As with CPU scheduling, disk-scheduling algorithms that take into account timing constraints can significantly improve the real-time performance. CPU scheduling algorithms, like Earliest Deadline First and Highest Priority First, are really suitable but have to be modified before they can be applied to I/O scheduling. The main reason is that disk seeks time, which accounts for a very significant fraction of disk access latency, depends on the disk head movement. The order in which I/O requests are serviced, therefore, has a great impact on the response time and throughput of the I/O subsystem [15,17].

To service a disk request, several operations take place. First, the disk head must be moved to the appropriate cylinder (seek time). Then, the portion of the disk on which the disk page is stored

must be rotated until it is immediately under the disk head (latency time). Then, the disk page must be made to spin by the disk head (transmission time). Disk scheduling involves a careful examination of the pending disk requests to determine the most efficient way to service the disk requests. The disk scheduling problem involves reordering the disk requests in the disk queue so that the disk requests will be serviced with the minimum mechanical motion by employing seek optimization and latency optimization [8].

For a given set of jobs, the general scheduling problem asks for an order according to which the jobs are to be executed such that various constraints are satisfied. Typically, a job is characterized by its execution time, ready time, deadline, and resource requirement. The execution of the job may or may not be interrupted (preemptive or non-preemptive scheduling). Over the set of jobs, there is a precedence relation which constraints the order of execution. Specially, the execution of a job cannot begin until the execution of all its predecessors is completed. The following goal should be considered in scheduling a real-time system [9].

- Meeting the timing constraints of the system.
- Preventing simultaneous access to shared resources and devices.
- Attaining a high degree of utilization while satisfying the timing constraints of the system.
- Reducing the communication cost in real-time systems.

Basically, the scheduling problem is to determine a schedule for the execution of the jobs so that they all are completed before the overall deadline.

3 DISK SCHEDULING ALGORITHMS

3.1 Classical Scheduling Algorithms

The following classical scheduling algorithms are well accepted [10].

FCFS : This is the simplest strategy in which each request is served in first-come-first-serve basis.

SCAN : This is also known as the elevator algorithm in which the arm moves in one direction and serves all the request in that direction until there are no further request in that direction.

C-SCAN : The circular SCAN algorithm works in the same way as SCAN except that it always scans in one direction. After serving the last request in the scan direction, the arm return to the start position.

SSTF : The SSTF, for shortest seek time first, algorithm simply selects the request closest to the current arm position for service.

A common feature of all these classical scheduling algorithms is that none of them takes the time constraint of request into account. This results in poor performance of classical algorithms in real-time systems.

3.2 Real-Time Disk Scheduling Algorithms

The real-time disk scheduling algorithms like Earliest Deadline First (EDF), Priority Scan (P-Scan), Feasible Deadline Scan (FD-Scan), Shortest Seek and Earliest Deadline by Ordering (SSEDO) and Shortest Seek and Earliest Deadline by Value (SSEDV) are discussed here.

EDF Algorithm : The Earliest Deadline First (EDF) algorithm is an analog of FCFS. Requests are ordered according to deadline and the request with the earliest deadline is serviced first. Assigning priorities to transactions an Earliest Deadline policy minimizes the number of late transactions in systems operating under low or moderate levels of resource and data contention. This is due to the highest priority given to the transactions that have the least remaining time in which to complete. However, the performance of Earliest Deadline steeply degrades in an overloaded system [11]. This is because, under heavy loading, transactions gain high priority only when they are close to their deadlines. Gaining high priority at this late stage may not leave sufficient time for transactions to complete before their deadlines. Under heavy loads, then, a fundamental weakness of the Earliest Deadline priority policy is that it assigns the highest priority to transactions that are close to missing their deadlines, thus delaying other transactions that might still be able to meet their deadlines [12,16].

P-SCAN Algorithm: In Priority Scan (P-Scan) all request in the I/O queue are divided into multiple priority levels. The Scan algorithm is used within each level, which means that the disk serves any requests that is passes in the current served priority level until there are no more requests in that direction. On the completion of each disk service, the scheduler checks to see whether a disk request of a higher priority is waiting for service [13]. If found the scheduler switches to that higher level. In this case, the request with shortest seek distance from the current arm position is used to determine the scan direction. All the I/O requests are mapped into three priority levels according to their deadline information. [10].

FD-SCAN Algorithm: In Feasible Deadline-Scan (FD-SCAN), the track location of the request with earliest feasible deadline is used to determine the scan direction. A deadline is feasible if we estimate that it can be met. Each time that a scheduling decision is made, the read requests are examined to determine which have feasible deadlines given the current head position. The request with the earliest feasible deadline is the target and determines the scanning direction. The head scans toward the target servicing read requests along the way. These requests either have deadlines later than the target request or have unfeasible deadlines, ones that cannot be met. If there is no read request with a feasible deadline, then FD-SCAN simply services the closest read request. Since all request deadlines have been missed, the order of service is no longer important for meeting deadlines [13].

SSEDO Algorithm: The idea behind Shortest Seek and Earliest Deadline by Ordering (SSEDO) algorithm is that requests with smaller deadlines are given higher priorities so that they can receive service earlier. This can be accomplished by assigning smaller values to their weights. On the other hand, when a request with large deadline is “very” close to the current arm position (which means less service time), it should get higher priority. This is especially true when a request is to access the cylinder where the arm is currently positioned. Since there is no seek time in this case and it is assumed the seek time dominates the service time ,

the service time can be ignored. Therefore these requests should be given the highest priority. [10].

SSEDV Algorithm: The Shortest Seek and Earliest Deadline by Value (SSEDV) use the deadline value information for decision-making**.** In the SSEDO algorithm the scheduler uses only the ordering information of request deadlines and does not use the differences between deadlines of successive requests in the window. A common characteristic of SSEDV and SSEDO algorithm is that both consider time constraints and disk service times. Which part plays the greater role in decision-making can be adjusted by tuning the scheduling parameters depending on the algorithm [10][14].

4 THE PROPOSED APPROACH

In our proposed approach all the request in the I/O queue are divided into multiple priority levels. Transactions are assigned the priorities depending on the deadlines. All the I/O requests are mapped into three priority levels according to their deadline information. Specially, transactions relative deadlines are uniformly distributed between LOW_DL and UP_DL, where LOW_DL and UP_DL are lower and upper bounds for transaction deadline settings. If a transactions relative deadline is greater than (LOW_DL + UP_DL)/2, then it is assigned the lowest priority. If the relative deadline is less than (LOW_DL + UP_DL)/4, then the transaction receives the highest priority. Otherwise the transaction is assigned a middle priority.

The algorithm selects the transaction with minimum deadline from the high priority level and also servers the transactions that are close to the current head position and then serve the transaction with next minimum deadline. Thus the requests with smaller deadlines can receive service earlier and also when a request with large deadline is very close to the current arm position are also served which will reduce the arm moment. Same procedure is repeated for all the transactions in the three priority levels. The important steps in scheduling algorithm are :.

- Construct three queues to store the transaction with minimum, middle or maximum priorities.
- Store the transaction in the corresponding queue.
- Set start_time, end_time, seek_time, current_head_position, total_transaction_time, turn_around_time for the transactions with minimum deadline in the three queues.
- Find transactions with seek time within threshold
- Check transaction is miss or hit in all the queues.

5 THE EXPERIMENTAL RESULTS

We have investigated and implemented various real-time disk scheduling algorithms namely EDF, P-SCAN, FD-SCAN, SSEDO SSEDV and our proposed novel algorithm with non-preemptive policy for soft deadline transaction. In these algorithms, preferential treatment is given to transactions, which are very critical, and with stringent timing constraints. Hence deadline is calculated on the basis of transaction execution time and slack time. We have also compared the performance of these algorithms under same workload condition. For the implementation of above-mentioned algorithms first we have formulated the disk-scheduling problem for real-time database systems and then implemented the mathematical model for all the algorithms. To get the evaluation parameters values, we have simulated the mathematical model for number of times. The experimental results show that the performance of SSEDO and SSEDV is better than EDF, FD-SCAN and P-SCAN in heavy workload. When the proposed algorithm was implemented it gave still better results as compared to SSEDO and SSEDV.

5.1 Performance of Various Disk Scheduling Algorithms

We explored the transaction loss probability of all the five algorithms stated in section 3 plus the proposed algorithm explained in section 4 under different workloads. For random *arrival fashion* of the transactions, with arrival rate 0.15, number of transactions 200 and disk size in blocks 100, the comparison given here is based on the properties like total transactions, successful transaction, time spend on all transactions, time spend on successful transaction, utilization of system and success ratio.

Table 1. Performance of Algorithms for random arrival

Properties	EDF	FD-SCAN	P-SCAN	SSEDO	SSEDV	Proposed
Total Transaction	200	200	200	200	200	200
Successful Transaction	70	93	98	132	151	154
Time spent On All Transactions	4147	1590	1598	1079	1563	1580
Time spent On Successful Transactions	1149	779	815	715	1252	1220
Utilization Of System	0.28	0.49	0.51	0.66	0.80	0.77
Success Ratio	0.35	0.47	0.49	0.66	0.76	0.77

As shown in table 1, performance of SSEDV is better than SSEDO, since the SSEDV uses more timing information than the SSEDO for decision-making. P-SCAN and FD-SCAN perform essentially at the same level. The EDF algorithm is good when the system is lightly loaded, but it degenerates as soon as load increases, as shown in figure 2. Our proposed approach shows still better performance.

In figure 1 the graph shows the performance of all the six algorithms in terms of utilization of systems and success ratio. Figure 2 is the result of nine runs with different transactions load ranging from 10 to 250 transactions with random arrival pattern. As discussed earlier it is clear that the performance of EDF degrades as the load increases and performance of our proposed algorithm is better when the load is increased.

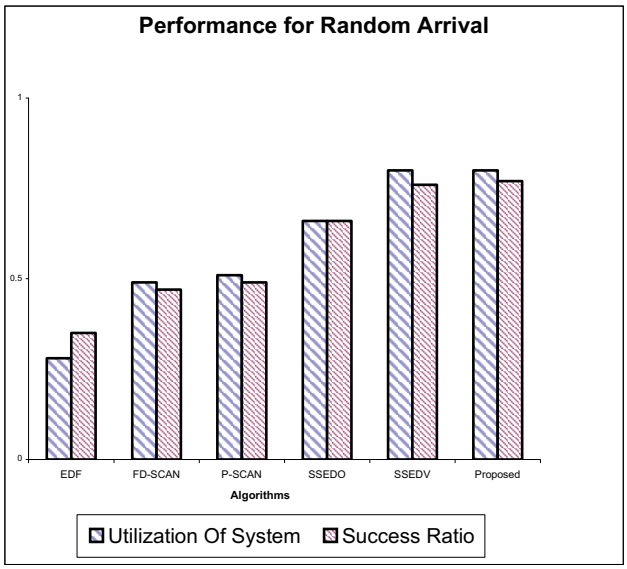

Figure 1 Combined Comparison for Random Arrival

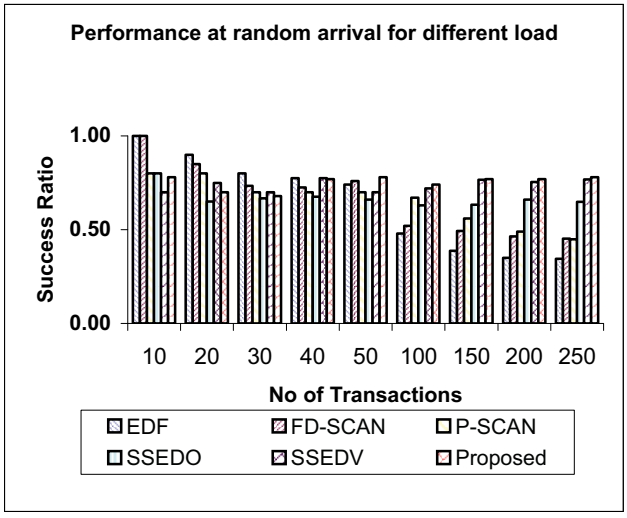

Figure 2 Performance at random arrival for different load

6 CONCLUSION

We investigated various real-time disk-scheduling algorithms namely EDF, P-SCAN, FD-SCAN, SSEDO and SSEDV. In EDF transactions are ordered according to deadline and the request with earliest deadline is serviced first. Priority-Scan divides all the request in the I/O queue the scan algorithm then serves any request that is passes in the current served priority level until there are no more requests in that direction. In FD-SCAN, the track location of the request with earliest feasible deadline is used to determine the scan direction. In the SSEDO algorithm, the scheduler uses the ordering information of request deadlines, whereas SSEDV use the difference between deadlines of successive requests in the window.

The results of the comparison shows that, performance of SSEDV is better than SSEDO, since the SSEDV uses more timing information than the SSEDO for decision making. P-SCAN and FD-SCAN perform essentially at the same level. The EDF algorithm is good when the system is lightly loaded, but it degrades as load increases. When we implemented our proposed algorithm, which divides the transactions into three classes, then schedules the transactions considering the priority and the head position, it gave better results. As different algorithms show different results at various transaction load the further modification to this disk scheduling problem in Real-time Database System can be monitoring the I/O load dynamically, focusing on using analyses of disk accesses to determine the best disk scheduling algorithm for the current workload, and switching algorithms as necessary to improve performance.

7 ACKNOWLEDGMENTS

The authors would like to thank the Principal, PRMITR Badnera-Amravati (India) for encouragement and support.

8 REFERENCES

[1] Krithi Ramamritham, “Real-Time Databases,” Journal of Distributed and Parallel Databases, Vol. 1, No. 2, 1993, pp.199-226., Kluwer Academic Publishers Hingham, MA, USA, April 1993.

[2] Barbosa Raul, “An Essay on Real-Time Databases”, Department of Computer Science and Engineering, Chalmers University of Technology, Goteborg, Sweden, 2007.

[3] Lund K, Goebel V., “Adaptive Disk Scheduling In A Multimedia DBMS”, Proceedings of the eleventh ACM international conference on Multimedia, Berkeley, CA, USA , 65 – 74, 2003.

[4] Ben Kao and Reynold Cheng,” Disk I/O Scheduling. In *Real-Time Database Systems: Architecture and Issues”*, edited by Kam-Yiu Lam and Tei-Wei Kuo, Kluwer Academic Publishers, pp. 97-107, Boston, December 2001.

[5] Ben Kao and Hector Garcia-Molina., ”An Overview of Real-Time Database Systems”, Proceedings of NATO Advanced Study Institute on Real-Time Computing. St. Maarten, Netherlands Antilles, Oct 1992.

[6] Aldarmi, S.,”*Real-Time Database Systems: Concepts and Design*”, Department of Computer Science, The University of York., April 1998.

[7] Ackovska N, Bozinovski S, Jovancevski G, “Real-Time Systems- Biologically Inspired Future”, Journal of Computers, Vol 3, No.3,No 3, pp 56-63, March 2008.

[8] Tsai, C., et al., “An Efficient Real-Time Disk-Scheduling Framework With Adaptive Quality Guarantee”, IEEE Transactions On Computers, Vol. 57, pp 634-647.No.5, May 2008.

[9] Z.Dimitrijevic, R.Rangaswami, and E. Chang.,”Design, Analysis and Implementation of Virtual IO”, International Multimedia Conference Proceedings of the Tenth ACM International Conference on Multimedia, pp 231-234, Juan-les-Pins, France, September 2002.

[10] Shenze Chen, John A. Stankovic, James Kurose and Don Towsley “Performance Evaluation of Two New Disk

Scheduling Algorithms", The Journal of Real-Time Systems, Vol 3, number 3, 307-336,1991.

[11] Krithi Ramamritham, Sang Hyuk Son, Lisa Cingiser DiPippo, "Real-Time Databases and Data Services" Journal of Real-Time Systems 28(2-3) pp:179-215, 2004.

[12] R. Abbott and H. Garcia-Molina, "Scheduling Real-Time Transactions: A Performance Evaluation", Proceedings of the 14th International Conference on Very Large Data Bases, Los Angeles, California, pp: 1 – 12, March 1988.

[13] Robert K. Abbott, Hector Garcia-Molina," Scheduling I/O Requests with Deadlines: a Performance Evaluation" Real-Time Systems Symposium, Proceedings, 11th Volume , Issue , 5-7 Dec 1990 pp: 113 -124 ,1990

[14] Jayant R. Haritsa, Michael J. Carey, and Miron Livny "Value-Based Scheduling in Real-Time Database. Systems", The VLDB Journal — The International Journal on Very Large Data Bases, Volume 2, Issue 2 , pp: 117 – 152, April 1993.

[15] Reuther, L.,Pohlack, M., Rotational-Position-Aware Real-Time Disk Scheduling Using A Dynamic Active Subset (DAS), Proceedings of the 24th IEEE International Real-Time Systems Symposium, Cancun, Mexico, pp: 374- 385, 3-5 December 2003.

[16] Shih, S.; Young-Kuk Kim; Son, S.H., " Performance Evaluation Of A Firm Real-Time Database System", Proceedings of Second International Workshop on Real-Time Computing Systems and Applications, pp:116 – 124, 25-27 Oct. 1995 .

[17] Walter A. Burkhard, John D. Palmer, "Rotational Position Optimization (RPO) Disk Scheduling", First Conference on File and Storage Technologies, Monterey, California, January 28-29,2002.