International Journal on Emerging Technologies **2**(1): 126-130(2011) **ISSN : 0975-8364**

An Overview of Real-Time Disk Scheduling Algorithms

S.Y. Amdani* and M.S. Ali**

**Department of CSE, Babasaheb Naik Collge of Engineering, Pusad, (MS)*
***Prof. Ram Meghe College of Engineering and Management, Badnera, Amravati, (MS)*

(Received 14 Feburary 2011, 13 March 2011 Accepted)

ABSTRACT : Real-time disk scheduling plays an important role in time-constraints applications. The real time database system depends not only on the strict data consistency requirements but also on the time at which the results are produced. Due to rigorous timing requirements for error free output, data must be accessed under real-time constraints. Therefore how to maximize data throughput under real-time constraints poses a big challenge in the design of real-time disk scheduling algorithms. Numbers of algorithms are proposed to schedule real time transactions in order to increase the overall performance. In this paper we have presented overview of various Real-Time disk scheduling algorithms.

Keywords : Real-time, DBMS, Deadline, Scheduling, Transaction.

I. INTRODUCTION

In the information age, information spreading worldwide through Internet, and other medium, is bulk and changing constantly and dynamic in nature. As our society becomes more integrated with computer technology, information processed for human activities necessitates computing that responds to requests in real-time rather than just with best effort. In fact, Database Management systems have entered the Internet Age. If too many users approach for information, then this degrades the system performance. The degradation may cause delay and trouble for particular end user in accessing the information. Accessing information in easy way and within certain time limit, by keeping its freshness, assessing user's requirements and then providing them information in time is important aspect.

Conventional databases are mainly characterized by their strict data consistency requirements. Database systems for real-time applications must satisfy timing constraints associated with transactions.

Real time data base systems combine the concepts from real time systems and conventional database systems. Real time systems are mainly characterized by their strict timing constraints. Conventional databases are mainly characterized by their strict data consistency requirements. Thus, real time database systems should satisfy both the timing constraints with data integrity and consistency constraints.

Typically, a timing constraint is expressed in the form of a deadline, a certain time in the future by which a transaction needs to be completed. In real-time database systems, the correctness of transaction processing depends not only on maintaining consistency constraints and producing correct results but also on the time at which a transaction is completed. Transactions must be scheduled in such a way that they can be completed before their corresponding deadlines expire.

Example applications that handle large amounts of data and have stringent timing requirements include telephone switching radar tracking and others. Arbitrage trading, for example, involves trading commodities in different markets at different prices. Since price discrepancies are usually short-lived, automated searching and processing of large amounts of trading information are very desirable. In order to capitalize on the opportunities, buy-sell decisions have to be made promptly, often with a time constraint so that the financial overheads in performing the trade actions are well compensated by the benefit resulting from the trade

The goal of transaction and query processing in real-time databases is to maximize the number of successful transactions in the system. [1-5]

A. Disk Scheduling Problem

In a disk-based database system, disk I/O occupies a major portion of transaction execution time. 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).

Queues build up for each disk because the inter-arrival time of the disk requests can be smaller than the time required by the disk to service a disk request. 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. [6-7]

II. OVERVIEW OF EXITING ALGORITHMS

There are some ordinary I/O scheduling algorithms as following: FCFS (First Come First Serve) is the simplest strategy, for which the I/O request is served in first come first serve

sequence. SCAN is also called elevator algorithm, in which the disk arm moves in one direction and serves all the requests in that direction until there is no farther request. The disk arm then changes its scan direction and serves that direction request. C-SCAN is the circular SCAN algorithm, which is the same as SCAN except that after serving the last request in the scan direction, the disk arm returns to the start position without serving any request and then begins another scanning. SSTF (shortest seek time first) simply selects the request closest to the current disk arm position for service. [3]

In 1973 Liu and Layland [8][9] suggested the most popular real time disk scheduling algorithm Earliest Deadline First EDF. In EDF transactions are ordered according to deadline and the request with earliest deadline is serviced first. The EDF algorithm is good when the system is lightly loaded, but it degenerates as soon as load increases. Critical task may not get priority over non-critical tasks because the closeness of deadline is only deciding factor.

Pierre G. Jansen in 2003 [10] suggested a algorithm EDF with inheritance EDFI combination of EDF and deadline inheritance over shared resources. It can manage scheduling and dispatching very efficiently. The scheduler manages the set of admitted tasks using two queues and a stack. The Wait Queue holds tasks awaiting their release. When a task gives up the processor or reaches its deadline, it is put on this queue, from which it will be transferred to the next queue when it is released. The Released Queue holds processes that have been released but have not yet run. This queue is maintained in deadline order, earliest deadline first. The Run Stack holds the tasks that have already run; the currently running task is at the top of the stack and the tasks below it were preempted by the tasks immediately above them. But it is appropriate for those systems that have to work with limited resources.

In 2006 Wenming Li [11] proposed algorithm group-EDF, or gEDF, where the tasks with "similar" deadlines are grouped together (i.e., deadlines that are very close to one another), and the Shortest Job First (SJF) algorithm is used for scheduling tasks within a group. The algorithm tends to favor smaller jobs and thus it does not always guarantee fairness. Also the algorithm needs to sort the jobs in each group, which could incur more overhead during execution.

Carey, M. J. , Jauhari, R. and Livny, M. in 1989[12][9] suggested P-SCAN Priority-Scan, in which all the requests in the I/O queue are divided into multiple priority levels. The SCAN algorithm is used within each level, which means that the disk will serve any requests in the current priority level until there is no more request in this level. When each disk service is completed, the scheduler will check to see whether a disk request with higher priority is waiting for service. If there is, the scheduler will switch to that higher level. In this algorithm, the request with shortest seek time from the current disk arm position is used to determine the scan direction. It will have better I/O performance when there is only a little priority level. This algorithm has a small think time and makes good use of the disk bandwidth. Nevertheless, it can lead to unbounded starvation.

Abbott suggested a real-time disk scheduling algorithm in 1990[13][9], called Feasible Deadline SCAN (FD-SCAN). In this algorithm, the track location of the request with earliest feasible deadline is used to determine the scan direction. The deadline is feasible if it is estimated to be met. That is, the deadline of the request is greater than the current time plus its service time, which can be determined by the current disk arm position and the request's track location. So in the scheduling point, all requests are examined to determine which has the earliest feasible deadline. After the scan direction is selected, the disk arm will move toward that direction and serve all requests along the direction. This algorithm has been found to do well in terms of response and disk utilization but has high overhead. Potentially, it will have to reevaluate the feasibility criterion at the end of each block access.

To improve the disk-seek time, SCAN-EDF algorithm was proposed by Reddy and Wylie in 1993[14]. The SCAN-EDF disk scheduling algorithm combines seek optimization techniques and EDF in the following way. Requests with earliest deadline are served first. But, if several requests have the same deadline, these requests are served by their track locations on the disk or by using a seek optimization scheduling algorithm for these requests. But the efficiency of this algorithm relies on the number of requests that have the same deadline.

To overcome this problem, in 1998 [15] R. 1.Chang, W K. Shih, and R. C. Chang proposed Deadline-Modification-SCAN (DM-SCAN) that suggests the use of maximum-scannable-groups compute the suitable request group for seek-optimizing with guaranteed real-time requirements for rescheduling. The DM-SCAN identifies and reschedule the maximum-scanable-group repeatedly. In this algorithm, request deadlines are reduced several times during the process of rescheduling to preserve EDF schedule.

Unlike DM-SCAN, Reschedulable-group-SCAN (RG-SCAN) suggested by H. P. Chang, R. I. Chang, W. K. Shih, and R. C. Chang in 2002[16], does not require its input disk requests to be sorted by their deadlines. It also forms larger groups without any deadline modification.

In SCAN-EDF, DM-SCAN and RG-SCAN algorithms rescheduling is only possible within a local group of requests. H.-P. Chang in 2007 [17] suggests Global Seek-optimizing Real-time (GSR) disk scheduling algorithm that groups the EDF input tasks based on their scan direction. These tasks are moved to their suitable groups to improve the system performance in terms of increased disk throughput and decreased number of missed deadlines. GSR schedules are always feasible if the input real-time disk requests are EDF feasible sequence. But with an infeasible input, it is very unlikely to have a feasible output. This is due to the fact that after each regrouping of input tasks, GSR checks the feasibility of the new schedule. If the new schedule is infeasible, GSR algorithm ignores the movement and selects another request to regroup

and this continues until it reaches the last request.

In 2009, Myung Sub Lee [18] proposed a new real-time disk scheduling algorithm based on the insertion technique and a two-way scan technique. The research is composed of insertion technique that can insert inconsecutive request into proper SCAN groups when deciding SCAN groups and SCAN merge technique that can merge consecutive SCAN groups, and two-way SCAN technique that can decide the direction of SCAN in an effective way. But in this technique there are problems in deciding SCAN groups and service direction of SCAN.

S. Chen, J. A. Stankovic, J. F. Kurose, and D. Towsley [19] in suggested two better I/O scheduling algorithms in real-time system, which are SSEDO (shortest seek and earliest deadline by ordering) and SSDEV (shortest seek and earliest deadline by value). They maintain a queue sorted according to the absolute deadline of each request. There is a window for the queue, if window size is m, the first m requests in the queue will have the smallest deadline.

In SSEDO algorithm, the scheduler selects one of the requests from the window for service. In the scheduling rule, each request in window is given a weight that is increment sequentially. The request with minimum value by weight multiplying the distance between the current disk arm position and the request position is scheduled. The value is also called the priority of request. If these are more than one request with the same priority value, the request with earliest deadline is selected. Clearly the priority of each request will be changed as the disk arm moves.

In SSEDV algorithm, the scheduling rule is mainly the same as the SSEDO, but the way of calculating priority is different. In SSEDO algorithm, the scheduler uses only the ordering information of request deadlines. The SSEDV uses the differences between deadlines of successive requests in the window i.e. choose the request with minimum value for service (remaining lifetime of request i.e. length of time between current time and request deadline)

The SSEDO and SSEDV use the time constraint and the disk service time to make decision, so they can have better performance than the other variants of SCAN. They also have been proved that they significantly improve the performance of real-time system and are easily to implement. This algorithm has been found to do well in terms of response time and disk bandwidth utilization but has high overhead.

Cheng peng in 2000 [20] suggested an improvement on SSEDV, M-SSEDV (Multiple queues SSEDV). There are many disk I/O task queues, each of which is used for a disk. In every queue, the SSEDV algorithm is used, by choosing a perfect scheduling parameter to get the best performance. If one task is scheduled, when it waits for the arrival of the data, it can be put into sleep status. It will be waked up when the data arrives. The scheduler will choose a task from other queues by SSEDV. So many I/O tasks can be scheduled in parallelism. If a task has been scheduled, the other task in the same queue must not be scheduled until the original task is finished. This algorithm has an overhead of choosing the task from different queues and putting the task in different status.

Zhao Yuehua in 2010 [21] suggested MDTS (Multi-Dynamic Trans Scheduling). MDTS consider three types of transactions hard real time, soft real time and non real time. At the arrival of transaction, the system checks the type of transaction, if it is a hard real-time transaction, the module will suspend lowest priorities of non-real time or soft real-time transactions to meet the requirements; else if it is the soft real-time transaction then suspending non-real-time transaction to the requirements. For the different types of transactions, when a new one gets into the corresponding ready queue, It needs to dynamically adjust the priorities of the transactions in the ready queue. Since different types of transactions have different ways to calculate the priority, it dynamically adjusts priorities only in the corresponding type of ready queue. The problem with MDTS is it considers non real time transaction, since there are three types of transaction overhead of calculating priorities by three ways. Also for MDTS Access Control is required to determine the space in the ready queue.

In 2010 [22], Zhongcheng Yu, Qianzhu Shi and Chao Liu proposed Buffer-Controller-EDF (BC-EDF). Scheduler is decomposed into two components feedback controller and traditional EDF scheduler. The feedback controller is used to predict the share of the buffer cache overflow/underflow. An EDF scheduler is used to select the most suitable request to send to avoid buffer overflow/underflow. But it gives average performance in predictable workloads.

III. CONCLUSION

In this paper we have discussed overview of various real time disk scheduling algorithms. The above discussions illustrate lack of performance isolation provided by a disk scheduling algorithm. Research on disk scheduler is required so that it can properly account for multiple outstanding I/O requests and guarantee real-time constraints for both outstanding and pending real-time requests.

REFERENCES

[1] Ben Kao and Hector Garcia-Molina "An Overview of Real-Time Database Systems", in proceedings of NATO Advanced Study Institute on Real-Time Computing. St. Maarten, Netherlands Antilles, Springer-Verlag, 1993.

[2] Gyanendra Kumar Gupta, Shubha Jain, Vishnu Swaroop and A K Sharma "Resource Scheduling in Mobile Distributed Real Time Database Systems: A New Perception For Operating Systems", Proceedings of the 4th National Conference; INDIACom-2010.

[3] Stuart Shih, Young-Kuk Kim, and Sang H. Son "Performance Evaluation of a Firm Real-Time Data Base System" , IEEE, 1995.

[4] Richard S. Gray "A Study of Disk Performance Optimization" Thesis, East Tennessee State University, 2000.

[5] Lars Reuther Martin Pohlack "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, December 2003.

[6] Sameh Mohamed Ibrahim Elnikety "A Real-Time Disk Scheduling Algorithm for Multimedia Storage Servers" Thesis, Alexandria University, 1999.

[7] Arezou Mohammadi and Selim G. Akl "Scheduling Algorithms for Real-Time Systems" Technical Report No. 2005-499,, This work was supported by the Natural Sciences and Engineering Research Council of Canada, 2005.

[8] C. L. Liu and James W. Layland "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment" Journal of the Association for Computing Machinery, Vol 20, No. 1, pp. 46-61, 1973.

[9] Yifeng Zhu "Evaluation of Scheduling Algorithms for Real-Time Disk I/O", 2002.

[10] Pierre G. Jansen, Sape J. Mullender, Paul J.M. Havinga, Hans Scholten "Lightweight EDF Scheduling with Deadline Inheritance", Technical Report TR-CTIT-03-23, Centre for Telematics and Information Technology, University of Twente, Enschede. ISSN 1381-3625, 2003.

[11] Wenming Li, Krishna Kavi , Robert Akl, " A non-preemptive scheduling algorithm for soft real-time systems", 2006 .

[12] M. J. Carey, R. Jauhari, and M. Livny. "Priority in DBMS resource scheduling." Proceedings of the Fifteenth International Conference on Very Large Data Bases, pages 397-410, 1989.

[13] R. Abbott and H. Garcia-Molina. "Scheduling I/O requests with deadlines: a performance evaluation". Proceedings of the Real-time Systems Symposium, pages 113-124, 12, 1990.

[14] A. N. Reddy and J. Wyllie. "Disk scheduling in multimedia I/O system". Proceedings of ACM Multimedia'93, Anaheim, CA, pages 225-234, 8, 1993.

[15] R. 1. Chang, W K. Shih, and R. C. Chang, "Deadline-modification-scan with maximum scannable-groups for multimedia real-time disk scheduling," in Proceedings of the 19th IEEE Real-Time Systems Symposium, pp. 40-49, 1998.

[16] H. P. Chang, R. I. Chang, W. K. Shih, and R. C. Chang, "Reschedulable-Group-Scan Scheme for Mixed Real-Time/ Non-Real-Time Disk Scheduling in a Multimedia System," The Journal of Systems and Software vol. 59, no. 2, pp. 143-152, 2002.

[17] H.-P. Chang, R.-I. Chang, W.-K. Shih, and R.-C. Chang, "GSR: A global seek-optimizing real-time disk-scheduling algorithm," The Journal of Systems and Software, vol. 80, no. 2, pp. 198-215, 2007.

[18] Myung Sub Lee, Kwang-Jung-Kim and Chang-Hyeon-Park, "Real Time Disk Scheduling Algorithms based on two-way scan Techniques", Eight International Conference on Scalable Computing and Communications, 2009.

[19] S. Chen, J. A. Stankovic, J. F. Kurose, and D. Towsley. "Performance evaluation of two new disk scheduling algorithms for real-time systems" Technical Report UM-CS-1990-077, University of Massachusetts, Amherst, 1990.

[20] Cheng peng Zhou xinrong Zhang jiangling, "The Design of High Performance Control System in RAID", IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, 2000.

[21] Zhao Yuehua and Qiu Jing, "A new multi-dynamic priority real-time database scheduling algorithm", IEEE 2nd International Conference on Computer Engineering and Technology, 2010.

[22] Zhongcheng Yu, Qianzhu Shi and Chao Liu, "Research on Multimedia Transmission Controlled Scheduling Algorithm on Internet", Journal of Communication and Computer, ISSN 1548-7709, USA, Volume 7, No.2 (Serial No.63), 2010.