Research Article Impact Factor: 0.621 ISSN: 2319-507X
SY Amdani, IJPRET, 2014; Volume 2 (8): 307-314 IJPRET

INTERNATIONAL JOURNAL OF PURE AND APPLIED RESEARCH IN ENGINEERING AND TECHNOLOGY

A PATH FOR HORIZING YOUR INNOVATIVE WORK

PERFORMANCE EVALUATION OF GLOBAL SEEK OPTIMIZING REAL-TIME DATABASE ALGORITHMS

DR. S. Y. AMDANI[1], MISS. ANUPAMA CHANDRASEN GIRAM[2]

1. Associate Professor, Department of Computer Sc. & Engg., Babasaheb Naik College of Engg., Pusad.
2. ME Student, Department of Computer Sc. & Engg., Babasaheb Naik College of Engg., Pusad.

Accepted Date: 15/02/2014 ; Published Date: 01/04/2014

Abstract: *The* 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. The gap between the performance of processors and the performance of disks is enlarged and the performance bottleneck in a computer system is shifted to the storage subsystem. When multimedia applications become more important, the gap problem becomes more serious. This motivates the need for more efficient use of the transaction scheduler in order to maximize disk throughput and minimize seek time. Earliest-deadline-first (EDF), is good for scheduling real-time tasks in order to meet timing constraint. However, it is not good enough for scheduling real-time disk tasks to achieve high disk throughput. In contrast, although SCAN can maximize disk throughput, its schedule results may violate real-time requirements. Thus, during the past few years, various approaches were proposed to combine EDF and SCAN (e.g., SCAN-EDF and RG-SCAN) to resolve the real-time disk-scheduling problem. However, in previous schemes, real-time tasks can only be rescheduled by SCAN within a local group. Therefore, to improve the performance a new algorithm is proposed based on globally seek optimizing scheduling approach: GSR (Globally Seek-optimizing Rescheduling) algorithm. TGGSR achieves higher disk throughput as compare to EDF and GSR. This approach first generates the initial schedule considering the track locations of the transactions and then this input schedule is decomposed into the SCAN groups and final rescheduled result is found.

Keywords: Real-time, DBMS, Deadline, Scheduling, Transaction, EDF, GSR, TGGSR

PAPER-QR CODE

Corresponding Author: DR. S. Y. AMDANI

Access Online On:

www.ijpret.com

How to Cite This Article:

SY Amdani, IJPRET, 2014; Volume 2 (8): 307-314

INTRODUCTION

A disk scheduling algorithm is a technique of setting the order of disk I/O request aiming at improving the performance of disk system. To improve the disk performance of disk system, we must consider three aspects. First, throughput per unit time that measures the how many disk I/O requests are served during the specific unit time. Second, average response time that measures how fast each disk I/O request is processed. And third predictability of response time used in variance of response time as an element to measure predictability. Considering these aspects, various disk scheduling algorithms have been proposed. However, these algorithms are focused only on disk efficiency but do not consider the ending time of real-time disk I/O requests, so that they are not adequate for real time systems. In response to the problem, a number of real-time disk scheduling algorithms have been proposed. The real-time disk scheduling algorithms completes there transactions within an ending time limit, is the main advantage of a real-time disk scheduling algorithms.

Earliest-deadline-first (EDF), is good for scheduling real-time tasks in order to meet timing constraint. However, it is not good enough for scheduling real-time disk tasks to achieve high disk throughput. The real-time disk scheduling algorithms lowers the disk efficiency as they process disk I/O request in order of deadline. To solve this problem, disk efficiency was improved with scheduling algorithms using the SEEK OPTIMIZATION technique. For this group of consecutive transactions are made on the basis of certain condition and then SCAN algorithm is applied within these groups locally or globally[1][2].

In *locally seek-optimizing schemes* the transactions can only be rescheduled by SCAN within a local group. Note that, each group is a set of consecutive transactions that can be rescheduled by SCAN without missing their respective timing constraints. To resolve the drawback of previous approaches a *globally seek-optimizing* scheduling approach: GSR (globally seek-optimizing rescheduling) scheme is proposed.

ORGANIZATION

Rest of the paper is organizing as follows. Section 2 contains brief discussion of the related work in the previous seek optimization real time database algorithm. In section 3 we compare the performance of global seek optimization algorithms. Finally, we conclude this paper in section 4.

PROBLEM DESCRIPTION AND RELATED WORK

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 attractive candidates 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 an immense impact on the response time and throughput of the I/O subsystem. Classical disk scheduling schemes attempt to minimize the average seek distance. For example, in the *elevator* algorithm, the disk head is in either an inward-seeking phase or an outward-seeking phase. While seeking inward, it services any requests it passes until there are no more requests ahead. The disk head then changes direction, seeking outward and servicing all requests in that direction as it reaches their tracks.

EDF: In 1973 Liu and Layland, suggested the most popular real time disk scheduling algorithm Earliest Deadline First EDF. The Earliest Deadline First 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. The EDF only consider the order of deadlines and introduces huge amount of seek-time costs with poor disk throughput[3].

GSR: The SCAN can maximize data throughput, its schedule result does not meet the timing constraints of real-time transactions. In contrast, the EDF schedule is good for real-time requirements. However, its disk throughput is low. The SCAN schedule derives a shorter schedule fulfill-time but is not feasible. In contrast, the EDF schedule owns a feasible result but results in a longer schedule-fulfill time. There is a tradeoff relation between these two extreme schedule cases. It motivates us to construct a graph of EDF-to-SCAN mapping (ESM) to develop a new scheme for real-time disk scheduling. EDF-to-SCAN mapping is a bipartite mapping obtained by connecting each node in EDF schedule to corresponding node in SCAN schedule with an edge. To decompose the input schedule into a sequence of scan-groups where, each scan-group contains the maximum number of contiguous transactions with the same SCAN direction. Therefore, the input schedule can be represented by a piecewise-SCAN schedule. GSR try to minimize the number of scan-groups or to maximize the sizes of scan-groups under real-time requirements of such a piecewise-SCAN schedule. Since the more the transactions can be seek-optimized, the more the disk throughput is obtained[5].

TGGSR: The Track Group GSR (TGGSR) algorithm based on GSR algorithm. The TGGSR algorithm finds a rescheduled result where GSR may fail. In addition, it is more efficient than GSR in terms of response time. Experimental results indicate that the data throughput of TGGSR is 1.2 times of GSR, 1.25 times of RG-SCAN, and 1.27 times of DM-SCAN. TGGSR generates the initial schedule considering the track locations of the transactions and then this input schedule is decomposed into the SCAN groups and final rescheduled result is found. This new approach consists of three steps. The priority Pi is assigned to each transaction Ti such that **Pi= α*Di + (1-α)* Ai** where, (0< **α** < 1) is the percent of the deadline Di for Ti, Di is the deadline of Ti, and Ai is the track location(Start Block) of Ti. Then initial schedule will contain the transactions in an ascending order of their priority values. The mapping is formed considering priority based schedule as an input schedule unlike GSR where EDF is considered as an input schedule. Once this mapping is done, GSR is applied[6].

I. PERFORMANCE EVALUATION

With the help of following example the performance evaluation of global seek optimizing algorithms GSR and TGGSR will be done as follows.

Initial disk hesd ptions= 7

Tid	At	Tp	Bs	Sb	Eb	AET	Dl	Tt
T0	2	R	3	12	14	4.5	11	1.8
T1	0	W	2	10	11	3	6	1.2
T2	0	W	5	5	9	7.5	15	3
T3	4	R	6	19	24	9	22	3.6
T4	5	R	4	15	18	6	17	2.4
T5	4	W	5	11	15	7.5	19	3

Table 1.1 Parameter Calculations

The following table shows the service time of each transaction and the formula for calculating service time is as follows:

Cij=(End index of i – Start index of j)*0.3+ Transfer Time of j

Cji	T0	T1	T2	T3	T4	T5
T0	0	2.4	5.7	5.1	2.7	3.9
T1	2.1	0	4.8	6.0	3.6	3.0
T2	2.7	1.5	0	6.6	4.2	3.6
T3	5.4	5.4	8.7	0	5.1	6.9
T4	3.6	3.6	6.9	3.9	0	5.1
T5	2.7	2.7	6.0	4.8	2.4	0

Table 1.2 Service Table

After applying the EDF algorithm on above example the timing diagram is as follows, in EDF the transactions are arrange in ascending order of their deadline.

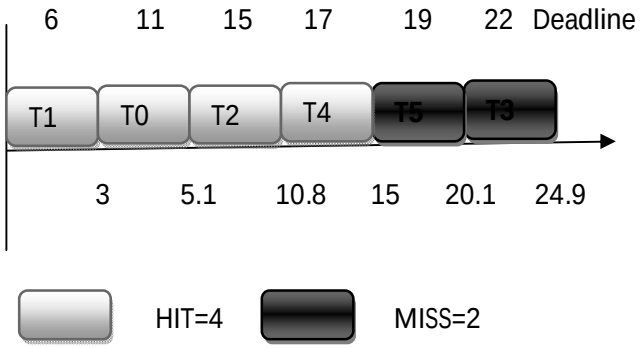

Figure 1: Timing diagram for EDF

After applying the GSR algorithm the timing diagram is as follows. GSR has main three steps:

A. EDF to SCAN mapping: EDF-to-SCAN mapping is a bipartite mapping obtained by connecting each node in EDF schedule to corresponding node in SCAN schedule with an edge.

B. Scan Group Identification(SGI): To decompose the input schedule into a sequence of scan-groups where, each scan-group contains the maximum number of contiguous transactions with the same SCAN direction

C. **GSR algorithm:** GSR try to minimize the number of scan-groups or to maximize the sizes of scan-groups under real-time requirements of such a piecewise-SCAN schedule. Since the more the transactions can be seek-optimized, the more the disk throughput is obtained.

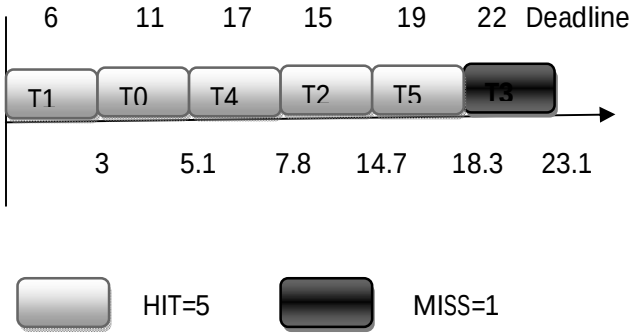

Figure2: Timing diagram for GSR

In case of TGGSR instead of taking EDF to SCAN mapping the initial schedule will form by using priority function: **Pi= α*Di + (1- α)* Ai** and then form SGI groups and GSR algorithm is applied on formed groups**.** After applying TGGSR algorithm the timing diagram is as follows.

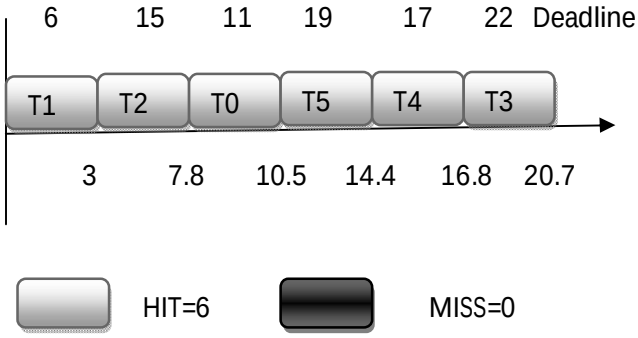

Figure 3: Timing diagram for TGGSR

From the above timing diagrams we can compare the performance of EDF, GSR and TGGSR[7].

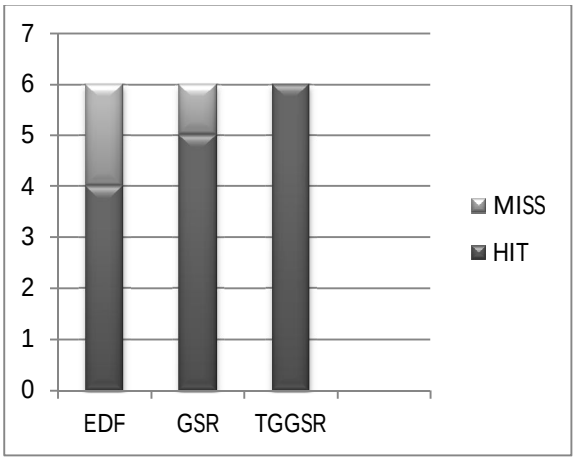

Figure 4: Performance Evaluation

II. CONCLUSION

In order to improve data throughput, the seek-optimizing SCAN scheme should be employed to reschedule the input tasks as much as possible. The goal of transaction and query processing in real-time databases is to maximize the number of successful transactions in the system and to minimize the response time. This motivates the need for more efficient use of the transaction scheduler in order to maximize disk throughput and minimize seek time. In this paper we have shown the performance evaluation of global seek optimization real time algorithms. GSR initially forms the scan groups on the basis of EDF to SCAN mapping and then reschedule the transactions globally i.e. from one group to another. GSR considers only Scan directions to form initial schedule. In case of TGGSR the initial schedule will form by using priority function. TGGSR considers Track locations as well as deadlines to form initial schedule.

REFERENCES:

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

2. Abbott, Robert and Hector Garcia-Molina, "Scheduling Real-Time Transactions: a Performance Evaluation," *Proceedings of the 14th VLDB Conference*, pp. 1-12, 1988.

3. D. L. Liu and James W. Layland “Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment” *Journal of the ACM*, volume 20, issue 1, pp. 46-61, January 1973.

4. Jayant R. Haritsa Miron Livny Michael J. Carey "Earliest Deadline Scheduling for Real-Time Database Systems" *Computer Sciences Department University of Wisconsin Madison*, WI 53706

5. Hsung-Pin Chang , Ray-I Chang , Wei-Kuan Shih , Ruei-Chuan Chang, "GSR: A global seek-optimizing real-time disk-scheduling algorithm." *The Journal of Systems and Software* 80 198–215, in 2007.

6. Nianmin Yao, Jinzhong Chen and Ang Li, "An Inter-group Seek-optimizing Disk Scheduling Algorithm for Real-time System", *Journal of Computational Information Systems*, volume 8, number 18, pp. 7579-7586, September 2012.

7. S. Y. Amdani, Dr. M. S. Ali "Performance Evaluation Of Seek Optimizing Real-Time Database Algorithms" *IJCIS, Vol. 2, No. 2*, 2011