Adaptive Earliest Deadline For Real Time Databases

Rajagopal Parthasarathi

Computer Science Department

Illinois Institute of Technology

Chicago,USA

rparthas@hawk.iit.edu

***Abstract*— we are living in the world of data explosion. The amount of data interactions has increased manifold. Some of them are critical transactions which need to be captured in a data structure and used for analysis and insight. Databases are a perfect solution to capture the humongous data today world is churning out. All data are not created equal and treated equal. Some data have temporal constraints and lose their value as soon as the time expires. Traditional databases are used to capture relational structured data and not time sensitive data. Real time databases are a perfect answer to the situation of time sensitive data. Earlier studies have observed that in lightly and moderately loaded systems, Earliest Deadline First algorithm performs best. But it performs poorly in case of heavily loaded situations. There have been various schemes proposed to improve this shortcoming of Earliest Deadline First. But all proposed schemes need priori knowledge of all the transactions. But this is nearly impossible in Real Time Database System. This paper proposes an new policy called Adaptive Earliest Deadline First (AED) which does not require priori knowledge of the transactions.**

Keywords—Realtime; Time Critical; Database; Deadlines; AED;Earliest Deadline First

I. INTRODUCTION

A Real Time Database System (RTDBS) is a transaction processing system with the intent of satisfying the time constraints associated with each transaction. The value of a transaction is nearly zero once the transaction executes after its deadline. For example, consider a stock market database. Each stock has value only till a specified time. After the time limit a new stock value replaces the old one. Thus the transaction is short lived and has to execute before its deadline

To address the problem of scheduling transactions with deadline, Earliest Deadline First was used in various schemes to address these transactions. EDF performs very well in case of lightly loaded and moderately loaded conditions. There are various works which prove this highly desirable property of EDF. But under heavy load, EDF performs very worse than even no priority situations. This is due to the fact that EDF schedules transactions only closer to its deadline. So every transaction starts missing its deadline instead of transactions which still can meet its deadline under heavy load.

To alleviate this shortcoming, this work discusses a new algorithm called Adaptive earliest deadline which is discussed in the later sections

II. RELATED WORK

Jayant R.Harshita et.al in their 1991 work considered Earliest Deadline Scheduling as the algorithm of choice for real time database systems. They proposed an Adaptive Earliest deadline algorithm which divides the workload into HIT group and MISS group. HIT tasks are scheduled on EDF while Random priority is followed for MISS group. This is based on the assumption that EDF performs poorly under overload situation while EDF is the best choice for normal load. Authors have verified the same by simulating a workload for a real time database. The work also involves Hierarchical AED to classify transactions based on value.

III. PROBLEM STATEMENT

Given this stringent transaction deadlines, under heavy load there exists no efficient algorithm which can schedule in manner so that maximum number of transactions meet their deadline. Also under heavy load conditions, Random priority performs much better than Earliest Deadline First policy. This automatically calls for a new scheduling policy which combines the best of both the situations.

In this work, only transactions with firm deadlines are considered i.e. there exists no value after the deadline is breached. There are no multiple priority mappings which makes it difficult to identify transaction behavior. Transactions are aborted only if they are found to be missing their deadline. Also preemption of transactions is considered to be resource intensive as it also involves additional overhead of synchronization. Hence transactions are not preempted to avoid this overhead. This work also does not consider batch transactions with data dependencies.

IV. ADAPTIVE EARLIEST DEADLINE

A. Algorithm

EDF is the best to schedule transactions to ensure it meets its deadline. But the flaw of EDF is it follows the same policy across all transactions even when the system is overloaded. EDF would perform better if we can classify transactions which would make its deadline during arrival time.

This is achieved in AED using a feedback control mechanism. In AED algorithm transactions are classified into

two groups HIT group and MISS group. The assignment is done when transaction arrives in queue the first time. Each transaction is assigned a unique identifier I_T and its position P_T in the list is noted. If P_T is less than HIT capacity then the transaction is assigned to HIT group, else it is assigned to MISS group. EDF is followed for all transactions belonging to the HIT group while MISS group follows Random Priority scheduling policy. Also all transactions in HIT group have higher priority than transactions in MISS group. This ensures that all transactions in HIT group reach its deadline. The goal is to make all transactions to be present in HIT group

HIT group is controlled by the HIT Capacity variable. The initial estimate of HIT capacity is set by the database administrator. HIT capacity is dynamically computed at each step of the algorithm with the following formula

1. HIT capacity = HIT Ratio (HIT) * HIT Capacity * 1.05
2. If HITRatio(ALL) < 0.95 then

   HIT Capacity = min(HIT Capacity, HIT Ratio(ALL) * Num of Trans * 1.25)

   HIT Ratio(HIT) is the number of transactions making their deadline in HIT group while HIT Ratio(ALL) contains transactions making their deadline all over the system.

   Step 1 ensures that whenever hit ratio reaches 0.95, HIT capacity becomes stabilized without any increase or decrease.

Step 2 is used to bring the system to best HIT capacity under heavy load conditions as soon as possible by using the 1.25 expansion factor. 1.25 was found by repeated experiments to be ideal value.

B. *Implementation*

Real-time Database is implemented using java. There is a transaction scheduler which interacts with the database. Transaction scheduler is initialized to interact with the particular database on startup. Transactions can be read or write transactions which the transaction manager executes by interacting with the database.

Data is stored in files and is operated by database object. Database object is exposed only to the scheduler. Transaction scheduler maintains a priority queue to hold the list of waiting transactions. Scheduler completes execution only if there are no waiting transactions and stop signal has been issued by database. This scheduler has various implementations which are discussed below

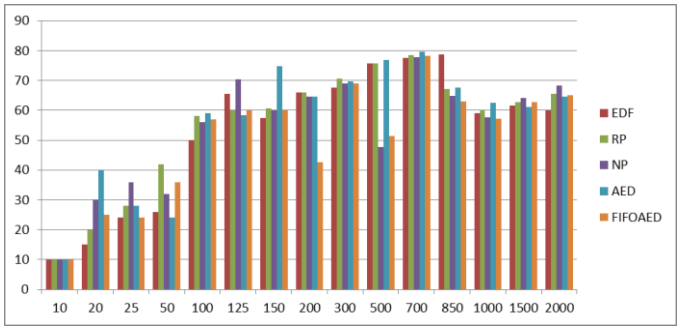
- No Priority – In this mechanism, transactions are executed in their natural order. They are executed as soon as scheduled.
- Random Priority – In this mechanism, transactions are executed in random order. When two transactions are compared, any one of the transaction may be executed first. A random generator is used to decide the priority.
- EDF – The absolute deadline of each transaction is computed before scheduling the transaction. The absolute deadline is then used to determine the transaction position in the list.
- AED – The algorithm is explained in detailed manner above. The same is implemented using a priority queue for hit group and a list structure for MISS group
- FIFO AED – This builds up on AED and uses natural ordering instead of random ordering. Natural ordering also leads to static priority assignment of the transactions

C. *Manual*

1. The entire source code is provided in the submitted zip file
2. The source code is also available in github : https://github.com/rparthas/CS552/tree/master/RTDBMS
3. Place the src and lib folders along with build.xml on the same folder
4. Ensure that ant and java_home properties are set and pointing to valid installations
5. Sample setcp.bat file provided for execution
6. Run the command ant in either shell(Linux) or command Prompt(Windows)
7. Ant generates the needed jar file
8. Use the file tasks.txt which serve as the input for the scheduler
9. Run the jar with the following command java -cp RTDBMS.jar edu.iit.cs552.test.TestDBOperations
10. The logs and result must be generated in log.txt file in the same working directory
11. Adjust dataset.properties to specify the workload size, arrival time and deadline.

V. EVALUATION

The database was simulated with various transactions and its corresponding miss rates were captured. The transaction load was simulated from 10, 20 to 2000 transactions. Miss rate in percentage were calculated. The below graph represents the summary of various transaction loads



In case of few transactions, EDF had a low miss percentage as it was ordering transactions based on deadline. Also AED and FIFO AED were adding extra overhead of calculating the various job parameters.

But as the transaction load started increasing, EDF performed poorly as the system was heavily loaded, lot of transactions started missing the deadline. AED was slightly better but in some case performed poorly than EDF. FIFOAED had a much better rate than EDF and AED having a very low miss rate. Since AED includes random priority, its exact performance cannot be predicted. Different runs produce different results. FIFO AED works on the principle of FCFS policy and also in most of the transactions earliest transactions seem to have early deadline. Hence FIFO AED performed well when the transaction count increased and system became heavily loaded. FIFO AED had an average miss rate of 50% while EDF had miss percentage of 56%. 6% becomes a critical factor when the number of transactions increases around 10,000. The other algorithms were randomly distributed between 50% and 56%

VI. FUTURE WORK

The batch transactions are important criteria for any dataset. They can be handled by Real Time Database using dependency graph. Multithreaded databases can be explored where a number of transactions can be executed simultaneously. Also in case of multithreaded transactions and batch transactions instead of random priority, least slack time can be used as the priority

VII. CONCLUSION

In this paper we addressed the issue of stabilizing the EDF under overload conditions. We also saw how AED can be combined with FCFS policy to exclude the random behavior and provide a guaranteed performance in case of real-time database systems

REFERENCES

[1] Jayant R. Harsita,Miron Linvy,Michael J. Carey, “Earliest Deadline Scheduling for Real Time Database Systems” , Computer Sciences Technical Report,May 1991

[2] Kao B, Garcia-Molina H, “ An overview of realtime database systems”, In: Stoyenko AD (ed) Real time computing.Springer, June 1993

[3] Jane W.S.Liu , “Real Time Systems”,April 2000