Dynamic Priority Assignment and Conflict Resolution in Real Time Databases

Alok Kumar
Indian Institute of Management

ABSTRACT
The conflict resolution technique normally used in Real Time Systems is EDF(Earliest Deadline First).However ,this technique is found to be biased towards the shorter transactions. No technique has been Reported in the literature to remove this biasing. This paper presents a new technique of dynamic priority assignment in real time transactions that is based on the ratio of time left to the time required to complete the transaction. The transaction which has the minimum fraction of time leftover as compared to the time taken for completion of that transaction is executed first. Thus it is free from any short of biasing.

Keywords
Real Time Systems, EDF, Dynamic Priority Assignment, Distributed Database

1. INTRODUCTION
Database systems serve as a backbone to thousands of systems and applications. Some of the systems have very high demands for availability and very fast real-time responses are required. Usually, such systems keep generating workload of very large transactions for the distributed real time database, and a substantial part of the workload consists of write, read and also update kind of transactions. Poor availability of real time systems and/or slow responses in processing such transactions used by real time business applications could infact be financially devastating and also in worst cases, cause deaths or damages. For example: tele-communication systems, trading applications, online gaming systems, sensor network applications etc. Typically, a sensor network system consists of a large number of sensors (both wireless and wired) which give reports on the status of some real-life situations. The situations include motion, sound, pressure, temperature, moisture and velocity etc. Such sensors send their data to a central application which makes such decisions based on both past and present inputs. For enabling such networks for making better quality outputs, both the number of sensors and the frequency of their updates need to be increased. Hence, such sensor networks systems should be able to withstand an increasing amount of load. For systems like health care in hospitals, auto car driving applications, space shuttle control systems etc., data is required in real-time scenario, and should be extremely reliable since any such poor availability or extra time taken by delays can lead to significant loss of human lives.

Most of the systems listed above using Distributed Real Time Databases require a distributed transaction to be executed at multiple sites. A commit protocol is made to ensure that either all the effects of the transaction should persist or none of them persist at all even in case of failure of a site or communication link and the resulting loss of messages. Hence, it is required that the Commit processing transactions should add as little overhead as possible to the transaction processing. Hence, it shows that the design of a much better commit protocol is very important for Distributed Real Time Databases.

2. DISTRIBUTED REAL TIME DATABASE SYSTEM MODEL
In the distributed database system model, the global or central database is sub-divided into a group of local databases stored at various different- different sites and locations. A communication network is required which interconnects the various sites. There is nothing as such global shared memory in the system, and all the sitesneed to communicate through message exchanges over the communication network. We assume that all the transactions are firm and real time. Each of these transactions in the present model exists in the form of a coordinator that executes at the site of origination of the transaction and a group of cohorts which execute at various other sites, where the required data items are located. If there are any local data which are in the access list of the transactions, then in those cases one of the cohort is executed locally. Before accessing any data item, the cohort needs to obtain lock on the data items. Sharing of such data items in conflicting modes creates dependencies among the group of conflicting local transactions and cohorts, which constraints their commit order. We also assume that:

- The processing of a transaction needs the use of CPU and the data items which are located at a local site or remote site.
- Arrival of any transactions at a site is independent of the arrivals at any other site and uses Poisson distribution.
- Each cohort can make read and update accesses.
- Each transaction has to pre-declare its read-set (set of data items that the transaction will read) and the update-set (data items that the transaction will update).
- S2PL-HP is used for locking the data items.
- Cohorts are executed in parallel order.
- Any lending transaction can not lend the same data item in read/update mode to more than a cohort.
- Any cohort already in the dependency set of any other cohort can not permit another incoming cohort to read or update.
- A distributed real time transaction is said to commit, if the coordinator has reached commit decision before the expiry of the deadline at its site location. This definition applies ir-respective of whether all

the cohorts have also received and recorded the commit decision within the deadlines or not.

- Studies have been made earlier for both main memory resident and disk resident database.
- Communication delays considered here is either 0 or 100 ms.
- For disk resident database, buffer space is considered sufficiently large to allow the retention of data updates until commit time.
- The updating of various data items is made in transactions own memory, not in place updating.

3. THE PROTOCOL

Let System time be Ts at the time of submission of the request for a Transaction to execute.

Let, Time required to execute the transaction(Ti) be= Tic

Deadline time for transaction completion(Time before which the transaction should be completed or aborted otherwise) = Tid .

We calculate the value of the ratio

Ri = (Tid - Ts)/ Tic

This ratio gives the fraction of time left for completion of the transaction as compared to the time required for the execution of the transaction. We calculate this ratio for all the transactions and execute them in increasing order of their Ri values. Thus the transaction with the least value of Ri value gets executed first.

It eliminates any biasing towards transactions having smaller completion time.

If two or more transactions have the same value of the ratio ,then we execute the transaction with lower execution time first to allow more number of transactions to complete within deadline time.

If some transactions have the same value of completion time ,then execute the transaction with lower number of write operations first.

It means transaction with higher number of read operations are performed first to allow more number of transactions to execute concurrently.

If the same data item is accessed for both Read and Write operations then we perform the Write operation first ,so that the Read operation reads the new(updated) value and not the older value. This is applicable to operations within a transaction.

Since,Read operations takes less time than Write operation we perform them earlier.

4. ALGORITHM

For each transaction compute the Ratio

If(Ri <1)

Then (/*abort the transaction since it cannot complete */)

Else /* execute the transactions in increasing value of Ri */

If(/* two or more transactions have the

same value of Ri */)

Then(/* execute the transaction with lower execution time first */)

If(/* some transactions have the same completion time */)

Then(/*execute the transaction with larger number of read operations first */)

If(/* same data is accessed for both read and write */)

Then(/*perform the write operation first*/)

Here, we note that we can execute the write operation first on a data if it is going to be read later so that the read operation reads the updated value only when it does not cause any consistency problems.

5. CONCLUSION

This paper presents a new technique of dynamic priority assignment in real time transactions that is based on the ratio of time left to the time required to complete the transaction. The transaction which has the minimum fraction of time leftover as compared to the time taken for completion of that transaction is executed first. Thus it is free from any short of biasing. This creates optimum schedule of transactions as per their possibility of completion and vastly improves the efficiency and success of the transactions.

6. REFERENCES

[1] Abbott Robert and Garcia - Molina H., "Scheduling Real - time Transactions with Disk Resident Data," Proceedings of the 15th International Conference on Very Large Databases**,** Amsterdam, The Netherlands, pp. 385 - 395, 1989.

[2] Abbott Robert and Garcia - Monila H., "Scheduling Real - Time Transaction: a Performance Evaluation," Proceedings of the 14th International Conference on Very Large Databases, pp. 1 - 12, August 1988.

[3] Abdallah Maha, Guerraoui R. and Pucheral P., "one - Phase Commit: Does It Make Sense," Proceedings of the International Conference on Parallel and Distributed Systems (ICPADS'98), Tainan, Taiwan, Dec. 14 - 16, 1998.

[4] Agrawal Divyakant, Abbadi A. El and Jeffers R., "Using Delayed Commitment in Locking Protocols for Real - Time Databases," Proceedings of the ACM International Conference on Management of Data (SIGMOD), San Diego, California, pp. 104 -113, June 2 - 5, 1992.

[5] Agrawal Divyakant, Abbadi A. El, Jeffers R. and Lin L., "Ordered Shared Locks for Real - time Databases," International Journals of Very Large Data Bases (VLDB Journal), Vol. 4, Issue 1, pp. 87 - 126, January 1995.

[6] Aldarmi Saud A. and Burns A., "Dynamic CPU Scheduling with Imprecise Knowledge of Computation Time," Technical Report YCS - 314, Department of Computer Science, University of York, U. K., 1999.

[7] Aldarmi Saud A., "Real - Time Database Systems: Concepts and Design," Department of Computer Science, University of York, April 1998.

[8] Al - Houmaily Yousef J. and Chrysanthis P. K., "Atomicity with Incompatible Presumptions," Proceedings of the 18th ACM Symposium on Principles of Database Systems (PODS), Philadelphia, June 1999.

[9] Al - Houmaily Yousef J. and Chrysanthis P. K., "In Search for An Efficient Real - Time Atomic Commit Protocol,"

[10] Url = citeseer.nj.nec.com/47189.html.

[11] Al - Houmaily Yousef J., Chrysanthis P. K. and Levitan Steven P., "Enhancing the Performance of Presumed Commit Protocol," Proceedings of the ACM Symposium on Applied Computing, San Jose, CA, USA, February 28 - March 1, 1997.

[12] Al - Houmaily Yousef J., Chrysanthis P. K. and Levitan Steven P., "An Argument in Favor of the Presumed Commit Protocol," Proceedings of the IEEE International Conference on Data Engineering, Birmingham, April 1997.

[13] Arahna Rohan F. M., Ganti Venkatesh, Narayanan Srinivasa, Muthukrishnan C. R., Prasad S. T. S. and Ramamritham K., "Implementation of a real - time database system," Information Systems, Vol. 21 , Issue 1, pp. 55 - 74, March 1996.

[14] Attaluri Gopi K. and Salem Kenneth, "The Presumed - Either Two - Phase Commit Protocol," IEEE Transactions on Knowledge and Data Engineering, Vol. 14, No. 5, pp. 1190 - 1196, Sept. - Oct. 2002.

[15] Audsley Neil C., Burns A., Richardson M. F. and Wellings A. J., "Data Consistency in Hard Real - Time Systems", YCS 203, Department of Computer Science, University of York, June 1993.

[16] Audsley Neil C., Burns A., Richardson M. F. and Wellings A. J., "Absolute and Relative Temporal Constraints in Hard Real Time Databases," Proceedings of the 4th Euromicro Workshop on Real - time Systems, IEEE Computer Society Press, Athens, pp. 148 – 153, June 1992.

[17] Bestavros Azer, "Advances in Real - Time Database Systems Research," ACM SIGMOD Record, Vol. 24, No. 1, pp. 3 - 8, 1996.

[18] Bestavros Azer, Lin K. J. and Son S. H., "Real - Time Database Systems: Issues and Applications," Kluwer Academic Publishers, 1997.

[19] Bowers David S., "Directions in Databases," Lecture Notes in Computer Science, 826, Springer - Verlag, pp. 23 - 54.