Algorithms for Managing QoS for Real-Time Data Services Using Imprecise Computation

Mehdi Amirijoo, meham@ida.liu.se
Dept. of Computer and Information Science (IDA)
Linköping University (LiU)

1. Background

Lately the demand for real-time data services has increased. Applications used in manufacturing, web-servers, e-commerce etc. are becoming very sophisticated in their data needs. In these applications it is desirable to process user requests within their deadlines using fresh data [4]. Since the external environment is constantly changing, it is imperative to maintain consistency between the environment and the database. In dynamic systems, such as web servers and sensor networks with non-uniform access patterns, the workload of the databases cannot be precisely predicted and, hence, the databases can become overloaded. As a result, many deadline misses and freshness violations may occur. To address this problem we propose a quality of service (QoS) sensitive approach, to guarantee a set of requirements on the behavior of the database, even in the presence of unpredictable workloads. Our scheme is important to applications where timely execution of transactions is emphasized, but where it is not possible to have exact analysis of the worst case execution times.

2. Approach

In our database model, data objects in a RTDB are updated by update transactions (simply referred to as updates), e.g. sensor values, while user transactions (simply referred to as transactions) represent user requests, e.g. complex read-write operations.

Below, we give an overview of how imprecise computation can be introduced in real-time databases (RTDB), followed by the definition of QoS and how this can be specified. Finally, a short description of the system architecture is given.

2.1 Imprecise Computation

Our approach is based on imprecise computation [2], where it is possible to trade off resource needs for quality of requested service. Imprecise computation techniques have been successfully applied to applications where timeliness is important. In our work, imprecise computation is employed on both data and transactions. For a data object representing a real-world variable, we can allow a certain degree of deviation compared to the real-world value. Hence, we relax the temporal consistency requirement. If such a deviation can be tolerated, arriving updates may be discarded and, hence, the saved CPU power can be allocated to other transactions. Discarding update transactions results in imprecision in data, which we refer to as data error.

Introducing impreciseness in transactions gives us another dimension by which we can trade off execution time for quality. For transactions, there are a variety of imprecise

computation models one can consider. These include milestone, use of sieve functions and multiple versions [2]. In this work we have used the milestone technique, since it has been shown that it can successfully be used in the context of RTDBs [5]. The main idea is to logically divide a transaction into a mandatory and one or more optional subtransaction. The mandatory subtransaction is necessary for an acceptable result and must be computed to completion before the transaction deadline. By executing more optional subtransactions, the overall quality of the result produced by the transaction is enhanced. During overloads, the optional parts can be discarded and consequently decreasing the execution time and required resources for the transaction. Discarding one or more optional parts give rise to a certain error which we call the transaction error.

2.2 QoS Specification

In our approach, the database administrator (DBA) can explicitly specify the required database QoS, which defines the desired behavior of the database. The QoS specification is given in terms of data error, transaction error, and system utilization and allows the DBA to specify the desired steady-state and transient-state performance. The performance metrics used in the QoS specification are:

- deadline miss percentage of mandatory user subtransactions, M^M
- deadline miss percentage of optional user subtransactions, M^O
- maximum data error, *MDE*
- utilization, U
- overshoot, M_p
- settling time, T_s

The steady-state performance is given using references, whereas the transient-state performance is given using overshoot and settling time [3]. Overshoot is the worst-case system performance in the transient system state and settling time is the time for the transient overshoot to decay and reach the steady state performance. Hence, the settling time determines the system adaptability and how fast the performance converges towards the desired performance given by the references.

2.3 System Architecture and Algorithms

During transient overloads the miss percentage of optional subtransactions may increase, increasing the transaction error. Here, a lower transaction error can be achieved by allocating more resources to transactions. This is achieved by lowering the workload of the data updates, resulting in an increase in data error. This forms the basic concept of our QoS management scheme.

We apply feedback control scheduling policy [3] to provide robustness against unpredictable workload variations. In a feedback control system, the reference performance can be achieved by dynamically adjusting the system behavior based on the difference of the current performance and the reference performance. Feedback control is very effective to support the specified performance when the dynamics of the controlled system includes uncertainties. By adapting the robustness of feedback control, we can provide the guaranteed real-time data services in terms of transaction error and data error.

The general outline of the feedback control scheduling architecture is given in Figure 1.

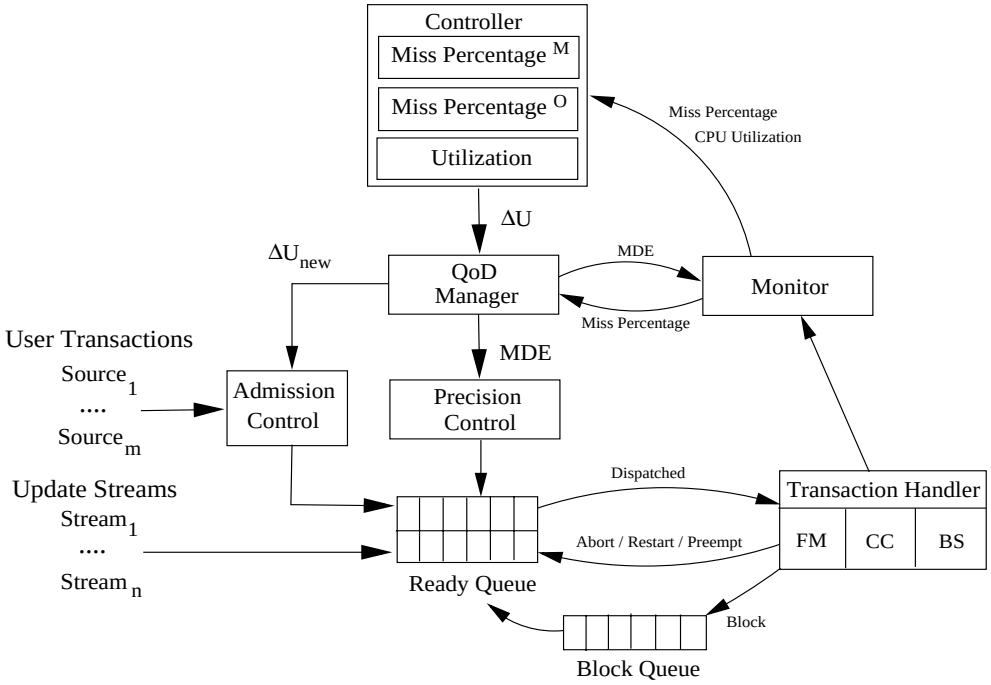

Figure 1. Feedback control scheduling architecture

Admitted transactions (or subtransactions) are placed in the ready queue. The transaction handler manages the execution of the transactions. At each sampling instant, the controlled variables, miss percentages and utilization, are monitored and fed into the miss percentage and utilization controllers, which compare the performance references with the corresponding controlled variables to get the current performance errors and compute a change, denoted ΔU, to the total estimated requested utilization. We refer to ΔU as the manipulated variable. Based on ΔU, the quality of data (QoD) manager changes the total estimated requested utilization by adapting the QoD level (i.e. adjusting *MDE*). The precision control then schedules the update transactions based on *MDE*. The portion of ΔU not accommodated by the QoD manager, denoted ΔU_{new}, is returned to the admission controller, which enforces the remaining utilization adjustment.

For managing the workload allocation between user and update transactions, we propose two dynamic balancing algorithms, FCS-IC-1 and FCS-IC-2, to balance the workload and hence the quality of the data and transactions. Main challenges include unpredictability of workload and effective workload balancing between user transaction and data quality. The suggested algorithms, FCS-IC-1 and FCS-IC-2, are designed such that the behavior of a real-time database can be controlled, even in the presence of load variation and inaccurate run-time estimates. For more detailed description we refer to [1].

3. Performance Evaluation

We have carried out a set of experiments to evaluate the performance of our algorithms. The main objective of the experiments is to show whether the presented algorithms can provide guarantees based on a QoS specification. We have for this reason studied and evaluated the behavior of the algorithms according to a set of performance metrics. The performance evaluation is undertaken by a set of simulation experiments, where a set of parameters have been varied. These are:

- Load. Computational systems may show different behaviors for different loads, especially when the system is overloaded. For this reason, we measure the performance when applying different loads to the system.
- Execution Time Estimation Error, EstErr. Often exact execution time estimates of transactions are not known. To study how runtime error affects the algorithms, we measure the performance considering different execution time estimation errors.
- QoS specifications. It is important that an algorithm can manage different QoS specifications. Here we compare the results of the presented algorithms with regards to different QoS specifications.

In our simulation studies we have applied a wide range of workload and run-time estimates to model potential unpredictabilities. Further, to the best of our knowledge, there has been no earlier work on techniques for managing data and transaction impreciseness, satisfying QoS and QoD requirements. For this reason, we have developed two baselines, Baseline-1 and Baseline-2, that can manage data impreciseness based on miss percentage of optional subtransactions. We use the baselines to study the impact of the workload on the system.

FCS-IC-1 and FCS-IC-2 give robust and controlled behavior of transaction and data quality, even during transient overloads and when we have inaccurate run-time estimates of the transactions. Comparing the performance against selected baseline algorithms has showed this.

More specifically, FCS-IC-1 can manage to provide near zero miss percentage for optional subtransactions. We have also seen that FCS-IC-1 can efficiently suppress miss percentage overshoots. However, the average performance of FCS-IC-1 does not fully comply with a given QoS specification. Miss percentages and *MDE* are kept significantly lower than the references, violating the QoS specification. This is due to policy used by FCS-IC-1, where the utilization is exponentially decreased every time M^O overshoots its reference. Although this policy is an efficient way of reducing number of potential overshoots, it often lowers the utilization unnecessarily, and as a result, data error is increased even though transaction error is low.

The performance of FCS-IC-2 has shown to satisfy a given QoS specification. Miss percentage of optional subtransactions, M^O, and *MDE* are consistent with the specified references. In addition, we have seen that the data error and transaction error increase and decrease together. FCS-IC-2, however, produces overshoots above the maximum allowed overshoot, as given by the QoS specifications. Overshoots occur from various disturbances, such as restart and blocking of transactions, and cannot be suppressed when M^O is kept near the reference. This is the disadvantage of FCS-IC-2.

We conclude that FCS-IC-1 performs better than FCS-IC-2 for suppressing overshoots. FCS-IC-1 should be applied to RTDBs where overshoots cannot be tolerated, but where consistency between the controlled variables and their references is relaxed, i.e. we do not require the system to produce the desired miss percentages and *MDE*. The experiments show that FCS-IC-2 is particularly useful when consistency between the controlled variables and their references is emphasized, but where some overshoots higher than the maximum allowed can be accepted.

4. Conclusion

The need for real-time data services has increased during the last years, e.g. sensor fusion support, web based applications, and telecommunications. Here timely processing of user transactions using fresh data is important. As the run-time environment of such applications tends to be dynamic, it is imperative to handle transient overloads effectively. It has been shown that feedback control scheduling is quite adaptive to errors in run-time estimates (e.g. changes in workload and estimated execution time). Further, imprecise computation techniques have shown to be useful in many areas where timely processing of tasks or services is emphasized. In this work, we combine the advantages from feedback control scheduling and imprecise computation techniques, forming a framework where a database administrator can specify a set of requirements on the database performance and service quality. We introduce two algorithms, FCS-IC-1 and FCS-IC-2, for managing steady state and transient state performance in terms of data and transaction error. FCS-IC-1 and FCS-IC-2 give a robust and controlled behavior of RTDBs, in terms of transaction and data quality, even during transient overloads and when we have inaccurate run-time estimates of the transactions.

References

[1] M. Amirijoo, "Algorithms for Managing QoS for Real-Time Data Services Using Imprecise Computation", Master's Thesis Report LiTH-IDA-Ex-02/90, www.ida.liu.se/~rtslab/master/past

[2] J. W.S. Liu, K. Lin, W. Shin, A. Chuang-shi Yu, "Algorithms for Scheduling Imprecise Computations", IEEE Computer, 1991.

[3] C. Lu, J. A. Stankovic, G. Tao, S. H. Son, "Feedback control real-time scheduling: Framework, Modeling and Algorithms", Journal of Real-time Systems, Special Issue on Control-Theoretical Approaches to Real-Time Computing, 2002.

[4] K. Ramamritham, "Real-time Databases", International Journal of Distributed and Parallel Databases, 1993.

[5] S. V. Vrbsky, W. S. Liu, " APPROXIMATE - A Query Processor That Produces Monotonically Improving Approximate Answers ", IEEE Transactions on Knowledge and Data Engineering, 1993.

Acknowledgement

This project was supervised by Jörgen Hansson at Linköping University and Sang H. Son at University of Virginia, Charlottesville. The author wishes to thank the supervisors for their support throughout the project.