A Feedback-Based Approach for Data Contention Control

Kyoung-Don Kang, Phil H. Sin, and Dong Kook Shin[1]

Summary

Databases have been successful to serve as the backbone of the information system. However, performance management in modern database systems is hard due to the inherent complexity. One of the key problems is managing data contention, since a database system may thrash under severe data conflicts. In this paper, we present a control theoretic approach for a database to autonomously−without substantial human interventions for (re)tuning the database−control data conflicts by adjusting the multiprogramming level even when the workload varies dynamically. To apply control theoretic approaches, we mathematically model the relation between the multiprogramming level and data conflicts. Based on the model, a feedback control framework is developed to maintain data conflicts, if any, below the threshold. Admission control and transaction cancellation are applied to enforce the required multiprogramming level adjustment computed in the feedback loop, if necessary, to control data conflicts. In this way, data conflicts can be controlled to be below the threshold even when the system is under a transient state subject to substantial data conflicts. Hence, the availability of database service can be significantly improved.

Introduction

Databases have been successful to serve as the backbone of information systems. As a result, databases have become a core of value-added information systems such as e-commerce systems. However, managing database performance is becoming harder, involving a lot of costly human interventions, due to the increasing complexity of databases. Hence, a self-tuning database that can dynamically adjust its own behavior to support reasonable performance even given dynamic workloads is desired [6, 13].

Managing data conflicts is one of the key problems towards self-adaptive database systems that can provide reasonable performance against possibly time-varying data contention, which is prevalent in databases.[2] As the multiprogramming level (MPL) increases, database throughput may increase; however, too high a MPL may cause a lot of data conflicts and resulting aborts, rollbacks, and restarts. In an extreme case, severe data contention can lead to data contention thrashing in which most resources are wasted due to excessive aborts, rollbacks, and restarts without making much progress in transaction processing.

To address this problem, we present a novel control theoretic approach to let a database system, if necessary, *autonomously* control the degree of data conflicts, measured in terms of data conflict ratio [8, 9, 12] (DCR), to be below a certain threshold. We apply control theoretic approaches for

[1]Department of Computer Science, T. J. Watson School of Engineering, State University of New York at Binghamton.

[2]We assume that the database system uses a pessimistic concurrency control scheme, since a large number of commercial databases employ two phase locking and its variants.

autonomous DCR control, because they are known to be very effective to support the desired performance when the system model involves uncertainties [11]. As a result, in our approach, the DCR can be managed to be below the threshold even when the system is in a transient state subject to a high degree of data contention.

Our approach is dynamically adaptable unlike a number of existing approaches in which the MPL is manually tuned and fixed. In the static MPL approaches, a database is not adaptive against the variable arrival rate and transaction mix incurring different degrees of data conflicts [8, 9]. Consequently, data contention thrashing may happen when the workload varies dynamically. Our approach can avoid the thrashing, while admitting more incoming transactions when the DCR is lower than the threshold specified by the database administrator (DBA). In our approach, the desired DCR can be supported even given dynamic workloads without requiring substantial human interventions for (re)tuning the database system. Thus, the availability of database service can be improved.

In our approach, we apply control theoretic methods to systematically manage both the average and transient DCR. Note that controlling the transient DCR in addition to managing the (long-term) average DCR is important, because a database may suffer severe transient data conflicts even when the average DCR is low. For example, the transient DCR reached 10 when the the threshold was set to 1.3 and the average was only 1.127 given online transaction processing workloads [9]. To apply control theoretic approaches, we first model the relation between the MPL and DCR. Based on the model, a feedback control framework is developed to adapt the MPL, if necessary, to control the DCR. When the DCR exceeds the threshold, the control signal computed in the feedback loop becomes negative requiring the reduction of the MPL. In this case, the MPL will be reduced by applying admission control to incoming transactions according to the control signal. For the further reduction of the MPL under severe data contention, we also apply a transaction cancellation scheme to abort a subset of transactions that are already in the system but blocked, while blocking (an)other transaction(s), similar to [9]. The aborted transactions will be restarted when the DCR drops below the threshold, i.e., when the control signal becomes positive.

We will compare the performance of our work to existing static MPL approaches and feedback-based approaches [7, 9] that rely on ad hoc feedback without applying mathematically well established control theoretic methods. Our performance evaluation to be reported in a separate paper will consist of simulation studies and actual implementations on top of an open source database system such as MySQL [1] or Postgre SQL [2]. For simulation purposes, we will vary the transaction arrival rate using both exponential and self-similar [5] arrival patterns in which the transaction arrival rate affecting the MPL can widely vary, similar to the web traffic. For performance evaluation, we will also vary transaction sizes in terms of the number of locks requested and data access patterns in terms of the hot spot size and read/write probability. Further, the TPC-W benchmark [4], which is designed for database performance evaluation in e-commerce applications, will be used in the real implementation-based experiments.

The remainder of this paper will give an overview of the feedback control framework, the definition of the average/transient DCR, the discussion of a mathematical model describing the relation between MPL and DCR, the development of a feedback controller for DCR control, and a discussion of the related work followed by the conclusions and future work.

An Overview of the Feedback Control Framework

As shown in Figure 1, the data contention controller *DCC* based on the MPL vs. data contention model (described in the next section) computes the required MPL adjustment ΔM according to the data contention error $E = \zeta_\theta - \zeta_C$, which is the difference between the data contention threshold and the data contention measured at the current sampling instant.

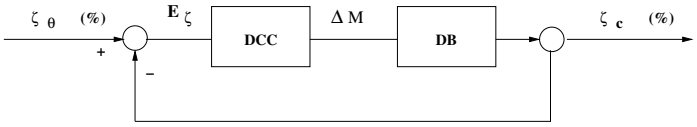

Figure 1: Feedback-Based Data Contention Control

When $\Delta M < 0$, the database will not immediately admit incoming transactions, but add them to the wait queue. Further, as many as $\Delta M (< 0)$ transactions, which are blocked while blocking others, can be canceled, i.e., aborted. When $\Delta M > 0$, aborted transactions, if any, will be restarted and waiting transactions will be admitted according to ΔM indicating the degree of allowed MPL increase. More details of admission control and transaction cancellation schemes are omitted due to space limitations. In this paper, we focus on the MPL vs. data contention modeling and the derivation of the feedback control framework as follows.

Modeling and Controlling Data Contention

To apply control theoretic approaches to performance management, one first needs to identify the controlled system such as a database in differential or difference equations [11]. We express the database system using the difference equations in the discrete time domain, in which most computational systems operate, to express the relations between the MPL and data contention. To this end, we identify the system model using the least square method [11], which is a powerful mathematical tool used to model a database [10] and other computational systems [3]. Based on the model, we derive the transfer function showing the relation between the input to the database, i.e., the MPL, and the output, i.e., data contention, in a control theoretic manner. Using the transfer function, we develop a feedback controller that can indicate how much to adjust the MPL, if necessary, to control the DCR to be below the threshold.

The DCR measured at a sampling period, e.g., 5 sec, is defined as follows, similar to [8, 12]:[3]

$$\zeta = \frac{\text{\#locks held by all transactions}}{\text{\#locks held by active (unblocked) transactions}} \tag{1}$$

Determining the threshold is important, because too high a threshold may cause data contention thrashing. On the other hand, too low a threshold may increase the response time, since some transactions can be needlessly forced to wait although the system is not thrashing. Moenkeberg et al. [9] experimentally found that a database is highly likely to thrash when $\zeta > 1.3$ using online transaction processing workloads. Later, Thomasian [12] analytically confirmed the existence of the

[3]Our work, if necessary, can also be extended to control data contention when an optimistic concurrency control is used. In this case, for example, the abort/rollback ratio $\omega = \frac{C+A+R}{C}$, in which C, A, and R represent the number of transaction commits, aborts, and restarts, respectively, can be used in lieu of the DCR.

critical DCR threshold of 1.3. We will initially use the critical threshold 1.3 and verify (or disprove) the validity of the critical threshold given TPC-W workloads [4].

In addition to the average DCR, we define the *overshoot* and *settling time* to control the transient DCR as follows.

- An overshoot is the DCR higher than the threshold, e.g., 1.3 [9], in a transient system state.
- The settling time T, e.g., 60 sec, is the time for a DCR overshoot, if any, to decay. After T, the database should enter a steady state in which the DCR is below the specified threshold such as 1.3.

Therefore, using our approach, the DBA can simply specify the desired average/transient DCR threshold and settling time instead of struggling to find the best possible MPL. For example, she can require that the average DCR ≤ 1.3, overshoot ≤ 1.5, and settling time ≤ 60 sec. Using our approach, the system can support the desired average and transient DCR by dynamically adjusting the MPL via admission control/transaction cancellation according to the control signal computed in the feedback loop as discussed before.

We model the relation between the MPL and DCR via an n^{th} ($n \geq 1$) order difference equation with unknown parameters, i.e., $\{a_i, b_i | 1 \leq i \leq n\}$ initialized to zero, as follows.[4]

$$\zeta(k) = \sum_{i=1}^{n} a_i \zeta(k-i) + \sum_{i=1}^{n} b_i M(k-i) \tag{2}$$

where $M(k-i)$ and $\zeta(k-i)$ are the MPL and DCR at the $(k-i)^{th}$ sampling instant, respectively. Eq 2 denotes that the current DCR is dependent on the inputs to the system, i.e., MPL's, and outputs from the system, i.e, DCR's, measured at the previous sampling instants. Hence, Eq 2 can model the database dynamics in terms of the MPL and DCR relation over the sampling history.

To estimate the parameters in Eq 2, the least square method is periodically invoked. By substituting the current estimates of the parameters to Eq 2, the DCR $\zeta(k)^p$ can be *predicted* at the k^{th} sampling instant. Given the DCR $\zeta(k)$ *measured* at the k^{th} sampling instant, the estimation error is $\zeta(k) - \zeta(k)^p$. It has been proved that the least square estimator can iteratively estimate the parameters in Eq 2 at each sampling instant such that the sum $\sum_{i=1}^{n} [\zeta(i) - \zeta(i)^p]^2$ is minimized. Let a vector $q(k) = (\zeta(k-1), ..., \zeta(k-n), M(k-1), ..., M(k-n))^T$. Define a vector $\theta(k) = (a_1(k), ..., a_n(k), b_1(k), ..., b_n(k))^T$, and let $P(k)$ be a square matrix that is initially a unit matrix. The least square estimator is described by the following equations:

$$\gamma(k) = \left[q(k)^T P(k-1) q(k) + 1\right]^{-1} \tag{3}$$

$$\theta(k) = \theta(k-1) + P(k-1) q(k) \gamma(k) \left[\zeta(k) - q(k)^T \theta(k-1)\right] \tag{4}$$

$$P(k) = P(k-1) \left[I - q(k) \gamma(k) q(k)^T P(k-1)\right] \tag{5}$$

Once the parameters $\{a_i, b_i | 1 \leq i \leq n\}$ are derived by iteratively solving Eq 3 $-$ Eq 5, the transfer function describing the relation between the MPL and DCR can be derived by taking the z-transform [11] of Eq 2. The z-transform is a commonly used technique in digital control developed

[4]The higher is the order, the more accurate is the model at the increased cost of modeling. We will experimentally select the specific order, e.g., $n = 3$ or 4, which can properly model the relation between the MPL and lock conflicts at a reasonable cost.

in the discrete time domain. It transforms difference equations into equivalent algebraic equations that are easier to manipulate. Although a complete review of z-transform is beyond the scope of this paper, we present one key property useful to manipulate the difference equations for our database model. Consider a sampled variable x whose samples are represented by $x[1], x[2], x[3], ...$; that is, $x[k]$ denotes the k^{th} sample. Let the z-transform of $x[k]$ be $X(z)$, then the z-transform of $x[k-n]$ is $z^{-n}X(z)$. We can apply the z-transform property to Eq 2 to get the following open-loop transfer function, which shows the relation between the MPL and DCR, after some algebraic manipulation:

$$T_{open}(z) = \frac{b_1 z^{n-1} + b_2 z^{n-2} + ... + b_n}{z^n - a_1 z^{n-1} - ... - a_n} \tag{6}$$

When the open-loop transfer function is T_{open}, the closed loop transfer function for feedback control as shown in Figure 1 is:

$$T_{closed} = \frac{T_{open}}{1 + T_{open}} \tag{7}$$

Given the closed-loop transfer functions and sampling period, e.g., 5sec, one (with basic knowledge in control theory) can apply the Root Locus method [11] to graphically tune the controller, i.e., *DCC* in Figure 1, to control the average and transient DCR to be below the thresholds specified by the DBA.

Related Work

Database self-tuning [13, 6] is considered a key problem in the next generation database research. Data contention control is one of the challenges towards autonomous databases [13]. Fixed MPL approaches, in which the DBA manually sets the MPL, may perform poorly when workloads involve varying degrees of data contention [8, 9]. Feedback-based approaches [7, 8, 9] have substantially improved the database throughput/response time by applying high level notions of feedback control, i.e., performance observation and dynamic adaptation of the system behavior in a conceptual feedback loop. Our work is novel in that it is based on formal control theoretic approaches to maintain the conflict ratio below not only the average but also transient thresholds even when the system is in a transient status. Therefore, our approach can further reduce the risk of thrashing, improving the database service availability in dynamic environments. In this way, our approach can partly handle the database self-tuning issues in terms of data conflict control [6, 13].

Conclusions and Future Work

Database tuning is a complex task. One of the key challenges is setting the multiprogramming level that can support good performance when the workloads may vary dynamically. Static approaches cannot effectively address this issue. In this paper, we present a control theoretic approach to autonomously control data conflicts by dynamically adjusting the multiprogramming level. Using our approach the data conflict ratio can be controlled to be below the threshold even when the system is under a transient state possibly incurring severe data contention. In the future, we will evaluate the performance of our approach and baselines by simulations and real implementations. We will also investigate how to manage other critical resources, e.g., main memory, for efficient transaction processing even in the presence of dynamic workloads.

References

[1] MySQL. http://www.mysql.com.

[2] PostgreSQL. http://www.pgsql.com.

[3] T. F. Abdelzaher. An Automated Profiling Subsystem for QoS-Aware Services. In *Real-Time Technology and Applications Symposium*. IEEE, 2000.

[4] Transaction Processing Performance Council. TPC-W. http://www.tpc.org/tpcw/default.asp.

[5] M. E. Crovella and A. Bestavros. Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes. *IEEE/ACM Transactions on Networking*, 5(6), Dec 1997.

[6] J. Gray. The Next Database Revolution. In *ACM SIGMOD*, June 2004.

[7] H.-U. Heiss. Overload Effects and Their Prevention. *Performance Evaluation*, 1991.

[8] A. Moenkeberg and G. Weikum. Conflict-Driven Load Control for the Avoidance of Data-Contention Thrashing. In *ICDE*, 1991.

[9] A. Moenkeberg and G. Weikum. Performance Evaluation of an Adaptive and Robust Load Control Method for the Avoidance of Data-Contention Thrashing. In *the 18th VLDB Conference*, 1992.

[10] H. Pang, M. Carey, and M. Livny. Multiclass Query Scheduling in Real-Time Database Systems. *IEEE Transactions on Knowledge and Data Engineering*, 7(4):533–551, August 1995.

[11] C. L. Phillips and H. T. Nagle. *Digital Control System Analysis and Design (3rd edition)*. Prentice Hall, 1995.

[12] A. Thomasian. Two-Phase Locking Performance and Its Thrashing Behavior. *ACM Transactions on Database Systems*, 18(4), 1993.

[13] G. Weikum, A. Moenkeberg, C. Hasse, and P. Zabback. Self-tuning Database Technology and Information Services: From Wishful Thinking To Viable Engineering. In *VLDB Conference*, 2002.