C

Control-Based Database Tuning Under Dynamic Workloads

Yi-Cheng Tu
University of South Florida, USA

Gang Ding
Olympus Communication Technology of America, Inc., USA

INTRODUCTION

Database administration (tuning) is the process of adjusting database configurations in order to accomplish desirable performance goals. This job is performed by human operators called database administrators (DBAs) who are generally well-paid, and are becoming more and more expensive with the increasing complexity and scale of modern databases. There has been considerable effort dedicated to reducing such cost (which often dominates the total ownership cost of mission-critical databases) by making database tuning more automated and transparent to users (Chaudhuri *et al*, 2004; Chaudhuri and Weikum, 2006). Research in this area seeks ways to automate the hardware deployment, physical database design, parameter configuration, and resource management in such systems. The goal is to achieve acceptable performance on the whole system level without (or with limited) human intervention.

According to Weikum *et al*. (2002), problems in this category can be stated as:

workload × configuration (?) → performance

which means that, given the features of the incoming workload to the database, we are to find the right settings for all system knobs such that the performance goals are satisfied The following two are representatives of a series of such tuning problems in different databases:

- **Problem 1: Maintenance of multi-class service-level agreements (SLA) in relational databases.** Database service providers usually offer various levels of performance guarantees to requests from different groups of customers. Fulfillment of such guarantees (SLAs) is accomplished by allocating different amounts of system resources to different queries. For example, query response time is negatively related to the amount of memory buffer assigned to that query. We need to dynamically allocate memory to individual queries such that the absolute or relative response times of queries from different users are satisfied.
- **Problem 2: Load shedding in stream databases.** Stream databases are used for processing data generated continuously from sources such as a sensor network. In streaming databases, data processing delay, i.e., the time consumed to process a data point, is the most critical performance metric (Tatbul *et al*., 2003). The ability to remain within a desired level of delay is significantly hampered under situations of overloading (caused by bursty data arrivals and time-varying unit data processing cost). When overloaded, some data is discarded (i.e., load shedding) in order to keep pace with the incoming load. The system needs to continuously adjust the amount of data to be discarded such that 1) delay is maintained under a desirable level; 2) data is not discarded unnecessarily.

Such problems can hardly be solved by using rules of thumbs and simply throwing in more hardware. In the following section, we shall also see that the traditional approach of treating tuning problems as static optimization problems does not work well for dynamic workloads such as those with OLAP queries. In this chapter, we introduce an emerging new approach to attack self-tuning database problems that is based on well-established results in feedback control theory. Specifically, we address the core issues of the approach and identify critical challenges of applying control theory in the area of self-tuning databases.

BACKGROUND

Current research in automatic tuning (or self-tuning) of databases tend to treat the problem as an optimization problem with the performance metrics and workload characteristics as inputs. The main drawback for this strategy is: real-world workloads, especially OLAP workloads, are highly unpredictable in that their parameters and behaviors can change very frequently (Tu *et al*., 2005). Such uncertainties in workloads can bring dramatic variations to system performance and cause the database to run in suboptimal status. In order to maintain consistently good performance, we need to develop means for the database to quickly adapt to the changes in workload.

One way to address the above challenge is to treat the problem as an online optimization (Chaudhuri and Weikum, 2006) and solve it by incremental algorithms. However, there is generally no guarantee on the accuracy and convergence of such algorithms, and some problems have no incremental solutions. Another important question, which is either ignored or answered empirically in current studies, is *how often do we need to rerun the optimization*? Our observation is that people tend to follow *ad hoc* strategies for individual problems in this field. It would be desirable to have a common theoretical framework under which a series of problems can be approached.

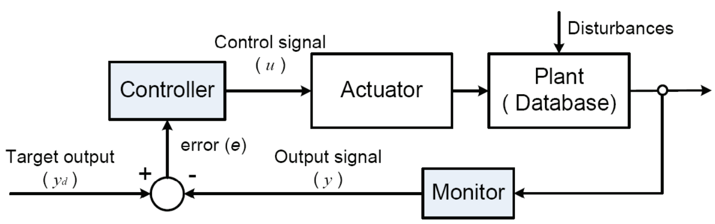
In this chapter, we argue that control theory provides such a foundation to approach the aforementioned problems in self-tuning databases. The reason for this is: designing systems with resistance to internal/external uncertainties is one of the main goals of control theory (Hellerstein *et al*., 2004). Note that control theory is not a single technique. Instead, it is the collection of a rich set of mathematical tools for analyzing system dynamics and designing mechanisms with guaranteed performance. We discuss some of the core issues of using control techniques in self-tuning databases. Currently, we have seen control-theoretical methods in solving database tuning problems and the effectiveness of the method is supported by both analytical and experimental results (Lightstone *et al*., 2007; Storm *et al*., 2006; Tu *et al*., 2005, 2006, 2007; Tu and Prabhakar 2006). For example, a load shedding framework based on feedback control designed by Tu and coworkers (2006, 2007) achieves processing delay violations that are 2 to 3 orders of magnitude lower (with the same amount of data loss), as compared to optimization-based methods. Storm *et al.*, (2006) reported the implementation of the Self-Tuning Memory Manager (STMM) with a control-based design and up to 254% increase in performance (e.g., decrease in query response time) in DB2 Version 9.1.

MAIN FOCUS

In this chapter, the term *control* refers to the manipulation of particular feature(s) (i.e., output signal) of a system (called *plant* in control terminology) by adjusting inputs injected into it (Hellerstein *et al*., 2004). We focus on *feedback control* where output signals are taken into account in making control decisions. The main components of a feedback control system form a *feedback control loop* (Figure 1): a *monitor* measures the output signal y of the plant; the measurements are compared with a desirable output value y_d and the difference between them is called *control error*; a *controller* maps control error e to a control signal u; an *actuator* adjusts the behavior of the plant according to signal u. The goal of the control operations is to overcome the effects of system and environmental uncertainties (named *disturbances*) such that the output signal tracks the target value.

The above conceptual design can be mapped into a concrete model for our problem: the plant is the database system; the actuator is the existing database mechanisms

Figure 1. Components of a feedback control loop



to implement a tuning decision; output signal is the performance metric(s) we consider; and the control signal is the database configurations we need to tune to. For example, in Problem 1, we have the query response time as output and resource allocation to multiples queries as input. Clearly, the most critical part of the control loop is the controller. Control theory is the mathematical foundation to the design of controllers based on the dynamic features of the plant. Carefully designed controllers can overcome unpredictable disturbances with guaranteed runtime performance such as *stability* and *convergence time*. However, due to the inherent differences between database systems and traditional control systems (i.e., mechanical, electrical, chemical systems), the application of control theory in database systems is by no means straightforward.

Modeling Database Systems

Rigorous control theory is built upon the understanding of the dynamics of the system to be controlled. Derivation of models that describe such dynamics is thus a critical step in control engineering. Therefore, the first challenge of control-based tuning is to *generate dynamic models of various database systems*. Linear systems have been well-studied in the control community. Generally, the dynamic behavior of a single-input-single-output (SISO) linear time-invariant (LTI) system can be modeled by a transfer function between the input u and output y in the frequency domain:

$$G(s) = \frac{a_n s^{n-1} + a_{n-1} s^{n-2} + \ldots + a_1}{s^n + b_n s^{n-1} + b_{n-1} s^{n-2} + \ldots + b_1}$$

where n is called the *order* of the model. The above transfer function only gives the relationship between the input and output. All the underlying n system dynamics x can be represented by a state-space model in time domain:

$$\begin{cases} x = Ax + Bu \\ y = Cx + Du \end{cases}$$

where A, B, C, and D are model parameters. Given an accurate model, all dynamics of the object can be analytically obtained, based on which we can analyze important characteristics of the output and states such as stability of the output, the observability, and controllability of each state.

Depending on the available information about the system, there are different ways to obtain the model. When the physical mechanism that drives the system is clearly known, the structure and all parameters of the model can be accurately derived. However, for complex systems such as a database, the analytical model may be difficult to derive. Fortunately, various *system identification* techniques (Franklin *et al*., 2002; Tu *et al*., 2005) can be used to generate approximate models for databases. The idea is to feed the database system with inputs with various patterns. By comparing the actual outputs measured with the derivation of the output as a function of unknown parameters and input frequency, the model order n and parameters a_i, $b_i (1 \le i \le n)$ can be calculated.

Control System Design

The dynamic model of a system only shows the relationship between system input and output. The second challenge of control-based tuning is to *design feedback control loops that guide us how to change the input* (*database configuration*) *in order to achieve desirable output* (*performance*). We need to test various design techniques and identify the ones that are appropriate for solving specific database tuning problems. When the linear system model is known, the proportional-integral-derivative (PID) controller is commonly used. It has three components and we can use mathematical tools such as *root locus* and *bode plot* to find controller parameters that meet our runtime performance requirements (Franklin *et al*., 2002). When model is partly known, more complicated design methods should be involved. For example, when some parameters of the model are unknown or time-variant, the controller can only be designed for an estimated model by online system identification (i.e., adaptive control). For systems with unstructured noises which vary fast, robust control has been actively studied (Ioannou and Datta, 1991).

Database-Specific Issues

Although control theory provides a sound theoretical background for the aforementioned problems, its application in self-tuning databases is non-trivial. The inherent differences between database/information systems and traditional control systems (i.e., mechanical, electrical, chemical systems) bring additional challenges to the design of the control loop. In this section, we discuss some of the challenges we identified:

1. **Lack of real-time output measurement:** Response time and processing delay are important performance metrics in database systems. Accurate measurements of system output in real-time is essential in traditional control system design. Unfortunately, this requirement is not met in self-tuning database problems where response time and processing delays are the output signals. Under this situation, the output measurement is not only delayed, but also delayed by an unknown amount (the amount is the output itself!). A solution to this is to estimate the output from measurable signals such as queue length.
2. **Actuator design:** In traditional control systems, the actuator can precisely apply the control signal to the plant. For example, in the cruise control system of automobiles, the amount of gasoline injected into the engine can be made very close to the value given by the controller. However, in database systems, we are not always sure that the control signal given by the controller can be correctly generated by our actuator. Two scenarios that cause the above difficulties in actuator design are: 1) Sometimes the control signal is implemented as a modification of the original (uncontrolled) system input signal that is unpredictable beforehand; 2) Another source of errors for the control signal is caused by the fact that the actuator is implemented as a combination of multiple knobs.
3. **Determination of the control period:** The control (sampling) period is an important parameter in digital control systems. An improperly selected sampling period can deteriorate the performance of the feedback control loop. As we mentioned earlier, current work in self-tuning databases consider the choice of control period as an empirical practice. Although the right choice of control period should always be reasoned case by case, there are certain theoretical rules we can follow. For controlling database systems, we consider the following issues in selecting the control period: 1) *Nature of disturbances*. In order to deal with disturbances, the control loop should be able to capture the moving trends of these disturbances. The basic guiding rule for this is the Nyquist-Shannon sampling theorem; 2) *Uncertainties in system signals*. Computing systems are inherently discrete. In database problems, we often use some statistical measurements of continuous events occurring within a control period as output signal and/or system parameter. This requires special consideration in choosing the control period; and 3) *Processing (CPU) overhead*. High sampling frequency would interfere with the normal processing of the database and causes extra delays that are not included in the model. In practice, the final choice of control period is the result of a tradeoff among all the above factors.
4. **Nonlinear system control:** Non-linear characteristics are common in database systems. When the model is nonlinear, there is no generic approach to analyze or identify the model. The most common approach is to linearize the model part by part, and analyze or identify each linear part separately. For the worst case when no internal information about the system is available, there are still several techniques to model the object. For example, the input-output relationship can be approximated by a set of rules provided by people familiar with the system, and a rule is represented by a mapping from a fuzzy input variable to a fuzzy output variable. An artificial neural network model can also be employed to approximate a nonlinear function. It has been proven that a well-tuned fuzzy or neural network model can approximate any smooth nonlinear function within any error bound (Wang *et al*., 1994).

FUTURE TRENDS

Modeling database systems is a non-trivial task and may bear different challenges in dealing with different systems. Although system identification had provided useful models, a deeper understanding of the underlying dynamics that are common in all database systems would greatly help the modeling process. We expect to see research towards a fundamental principle to describe the dynamical behavior of database systems, as Newton's Law to physical systems.

Another trend is to address the database-specific challenges in applying control theory to database tuning problems from a control theoretical viewpoint. In our work (Tu *et al*., 2006), we have proposed solutions to such challenges that are specific to the context in which we define the tuning problems. Systematic solutions that extend current control theory are being developed (Lightstone *et al*., 2007). More specifically, tuning strategies based on fuzzy control and neural network models are just over the horizon.

CONCLUSION

In this chapter, we introduce the new research direction of using feedback control theory for problems in the field of self-tuning databases under a dynamic workload. Via concrete examples and accomplished work, we show that various control techniques can be used to model and solve different problems in this field. However, there are some major challenges in applying control theory to self-tuning database problems and addressing such challenges is expected to become the main objectives in future research efforts in this direction. Thus, explorations in this direction can not only provide a better solution to the problem of database tuning, but also many opportunities to conduct synergistic research between the database and control engineering communities to extend our knowledge in both fields.

REFERENCES

Chaudhuri, S., Dageville, P., & Lohman, G. M. (2004). Self-managing technology in database management systems. *Proceedings of 30th Very Large Data Bases Conference* (pp.1243-1243).

Chaudhuri, S., & Weikum, G. (2006). Foundations of automated database tuning. *Proceedings of International Conference of Date Engineering* (pp. 104).

Franklin, G. F., Powell, J. D. & Emami-Naeini, A. (2002). *Feedback control of dynamic systems*. Massachusetts: Prentice Hall.

Hellerstein, J. L., Diao, Y., Parekh, S., & Tilbury, D. (2004). *Feedback control of computing systems*. Wiley-Interscience.

Ioannou, P. A. & Datta, A. (1991). Robust adaptive control: A unified approach. *Proceedings of IEEE*, *79*(12), 1736-1768.

Lightstone, S., Surendra, M., Diao, Y., Parekh, S., Hellerstein, J., Rose, K., Storm, A., & Garcia-Arellano, C. (2007). Control theory: A foundational technique for self managing databases. *Proceedings of 2nd International Workshop on Self-Managing Database Systems* (pp.395-403).

Paxson, V., & Floyd, S. (1995). Wide-area traffic: The failure of poisson modeling. *IEEE/ACM Transactions on Networking*, *3*(3), 226-244.

Storm, A., Garcia-Arellano, C., Lightstone, S., Diao, Y., & Surendra, M. (2006). Adaptive self-tuning memory in DB2. *Proceedings of 32nd Very Large Data Bases Conference* (pp. 1081-1092).

Tatbul, N., Cetintemel, U., Zdonik, S., Cherniack, M., & Stonebraker, M. (2003). Load shedding in a data stream manager. *Proceedings of 29th Very Large Data Bases Conference* (pp.309-320).

Tu, Y., Liu, S., Prabhakar, S., & Yao, B. (2006). Load shedding in stream databases: A control-based approach. *Proceedings of 32nd Very Large Data Bases Conference* (pp. 787-798).

Tu, Y., Liu, S., Prabhakar, S., Yao, B., & Schroeder, W. (2007). Using control theory for load shedding in data stream management. *Proceedings of International Conference of Data Engineering* (pp.1491-1492).

Tu, Y., Hefeeda, M., Xia, Y., Prabhakar, S., & Liu, S. (2005). Control-based quality adaptation in data stream management systems. *Proceedings of International Conference of Database and Expert Systems Applications* (pp.746-755).

Tu, Y., & Prabhakar, S. (2006). Control-based load shedding in data stream management systems. *Proceedings of International Conference on Data Engineering* (pp.144-144), PhD workshop/symposium in conjunction with ICDE 2006.

Wang, L. X. (1994). *Adaptive fuzzy systems and control: Design and stability analysis*. New Jersey, Prentice Hall.

Weikum, G., Moenkeberg, A., Hasse, C. & Zabback, P. (2002). Self-tuning database technology and information services: From wishful thinking to viable engineering. *Proceedings of 28th Very Large Data Bases Conference* (pp. 20-31).

KEY TERMS

Control Theory: The mathematical theory and engineering practice of dealing with the behavior of dynamical systems by manipulating the inputs to a system such that the outputs of the system follow a desirable reference value over time.

Controllability and Observability: Controllability of a state means that the state can be controlled from any initial value to any final value within finite time by only the inputs. Observability of a state means that, for any possible sequence of state and control inputs, the current state can be determined in finite time using only the outputs.

Convergence Rate: Another evaluation metric for closed-loop performance. It is defined as the time needed for system to bring the system output back to the desirable value in response to disturbances.

Feedback Control: the control method that takes output signals into account in making control decisions. Feedback control is also called closed-loop control due to the existence of the feedback control loop. On the contrary, the control that does not use output in generating control signal is called open-loop control.

Nyquist-Shannon Sampling Theorem: A fundamental principle in the field of information theory, the theorem states that: when sampling a signal, the sampling frequency must be greater than twice the signal frequency in order to reconstruct the original signal perfectly from the sampled version. In practice, a sampling frequency that is one order of magnitude larger than the input signal frequency is often used. The Nyquist-Shannon sampling theorem is the basis for determining the control period in discrete control systems.

System Identification: The process of generating mathematical models that describe the dynamic behavior of a system from measured data. System identification has evolved into an active branch of control theory due to the difficulty of modeling complex systems using analytical methods (so-called white-box modeling).

System Stability: One of the main features to consider in designing control systems. Specifically, stability means that bounded input can only give rise to bounded output.

Workload: A collection of tasks with different features such as resource requirements and frequencies. In the database field, a workload generally consists of various groups of queries that hold different patterns of data accessing, arrival rates (popularity), and resource consumption. Representative workloads are encapsulated into well-known database benchmarks.