Using Control Theory for Self-Tuning Database Systems

A Position Paper

Yi-Cheng Tu† Gang Ding† Song Liu‡ Sunil Prabhakar† Bin Yao‡

†Department of Computer Sciences
Purdue University
250 N. University St.
West Lafayette, Indiana, USA
{tuyc, dingg, sunil}@cs.purdue.edu

‡School of Mechanical Engineering
Purdue University
140 S. Intramural Drive
West Lafayette, Indiana, USA
{liu1, byao}@purdue.edu

ABSTRACT

Current studies on self-tuning databases tend to adapt a static view of problems by focusing on long-term performance. We argue that a more important issue is to maintain transient performance in such systems due to the inevitable fluctuations in workloads and environmental factors. We propose the idea of using feedback control theory to deal with such dynamics and maintain performance in self-tuning databases. We discuss how relevant control-theoretical techniques can be used to solve typical problems in this field. One of the major difficulties in using formal methods to study such problems is the lack of accurate analytical models for complex database systems. This can be partially solved by system identification techniques. Concrete examples and preliminary results support our proposal. We also identify the major challenges in applying control theory to self-tuning database study.

1. INTRODUCTION

In the past decade, we have witnessed enthusiastic research interests in self-managing database and information systems that can automatically adjust their configuration and behavior in order to meet performance requirements. This is evident with the recent establishment of the IEEE Computer Society Workshop on Self-Managing Database Systems[1]. Motivated by the increasing human costs in the maintenance of database and information services, 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.

Challenges have to be met such that the self-managing systems [3] to be built are: 1) *self-configuring*: system components can be installed, configured, and interconnected with little human supervision; 2) *self-optimizing*, automatic tuning of database knobs towards performance goals; 3) *self-healing*: detection and correction of abnormal situations; 4) *self-protecting*: defense against security attacks. In this paper, we focus on self-optimizing (i.e., auto-tuning) problems. According to Weikum *et al.* [17], problems in this category can be described by the following statement:

$$\text{workload} \times \text{configuration } (?) \rightarrow \text{performance} ,$$

which means that we need to find the right settings for (multiple) system knobs given the features of incoming workload such that performance metrics are satisfied. The following are some sample problems in this category:

- EXAMPLE 1. *Maintenance of Multi-Class workload service level agreements (SLA).* Service providers usually offer various levels of Quality-of-Service (QoS) guarantees (e.g., response time) to requests from different groups of customers. Such guarantees are called *service level agreements*. Fulfillment of the SLAs is accomplished by assigning different amount of system resources to different request groups. For example, query response time is negatively related to the amount of memory buffer assigned to that query [18]. We need to find a way to dynamically allocate memory to queries such that the absolute or relative query response time of different service groups are satisfied. The problem can involve multiple classes of resources and multiple QoS parameters [1];
- EXAMPLE 2. *Load shedding in data stream management systems (DSMSs)* [14]. In DSMSs, query processing has to meet various QoS requirements. Among them, processing *delay*, which is the duration of tuples' staying in the DSMS, is the most critical one since the value of query results decreases dramatically with increased freshness of input data. The ability to remain within a desired level of delay is significantly hampered under situations of overloading, which are common in data stream systems. When overloaded, DSMSs employ load shedding (i.e., discarding some data tuples) in order to keep pace with the high rate of data arrivals. In this problem, answers to the questions of *when*, *how much*, and *where* to discard load are to be

[1] http://db.uwaterloo.ca/tcde-smdb/

provided. The problem is difficult because data stream applications are extremely dynamic due to bursty data arrivals and time-varying data processing costs;

- EXAMPLE 3. *Dynamic Data Placement.* Data items are placed on multiple hard disks. The main goal of this problem is to find a data placement solution in which the access load on each disk is approximately the same [17]. We assume that the popularity (i.e., access rate) of each data item changes with time.

Such problems can hardly be solved by using rules of thumbs and simply throwing in more hardware [17]. Current studies tend to view these problems as static optimizations with the workload parameters and performance metrics as inputs [2]. This strategy works perfectly fine if we only need to maintain a static view of the workload and performance metrics. For example, if we only consider long-term performance such as average query response time over a long time, steady-state parameters (e.g., average query arrival rate) of the workload would probably be sufficient to solve the problem. However, we are rarely interested in long-term performance in real-world information services. Instead, it is generally required that the performance goals are always met. Unfortunately, fluctuations do exist in workloads and they can bring variations in system performance if we adapt a static solution for the system configuration. Furthermore, other internal/environmental factors may also cause variations of performance. Under this situation, we need to develop means for the system to quickly adapt to the dynamics in the workload, the database system, and the environment where the system runs. In other words, we need to focus on transient-state performance.

One way to address the above challenge is to treat the problem as an online optimization [2] and solve it by incremental algorithms. However, there is generally no guarantees 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 paper, we argue that control theory provides such a foundation to approach the aforementioned problems in self-tuning databases. 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 used control-theoretical methods to solve the problem in Example 2 and the effectiveness of the method is supported by both analytical and experimental results [14]. Therefore, we utilize Example 2 to explain how control techniques can be applied in self-tuning databases. Furthermore, we also identify critical challenges of applying control theory in this area.

Related Work. Control-theoretic approaches have been used to solve various problems in the areas of networking [10], real-time systems [12], and multimedia [11]. To the best of our knowledge, our work on load shedding in DSMSs [14] is the only such work in the area of database systems. Weikum *et al.* [17, 2] propose the concept of *feedback control loop* as a general principle for solving problems in self-tuning databases. The loop consists of three phases: 1) *observation:* monitoring performance metrics and workload parameters; 2) *prediction:* assessing the possible adjustments of system knobs based on a mathematical model such that performance requirements are satisfied; 3) *reaction:* implementing the decision made by step 2). As we shall see, this strategy, although with a similar name, is fundamentally different from the control-theoretical solution we propose. Our solution uses formal synthetic methods to derive the dynamic controller which can guarantee the system performance over a wide range of systems and external disturbances.

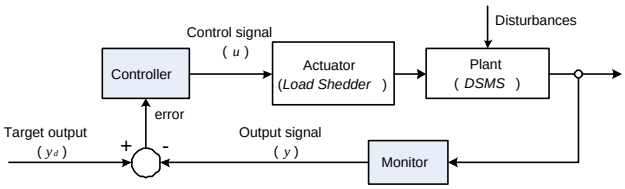

Figure 1: The feedback control loop (from [14]).

2. CONTROL SYSTEM OVERVIEW

The term *control* generally refers to the manipulation of particular feature(s) (i.e., output signal) of a *plant* by adjusting inputs into it. In this paper, we focus on feedback control where output signals are taken into account in making control decisions[2]. The main components of a feedback control system form a *feedback control loop* (Fig. 1). The runtime operations of this loop are performed periodically [3] as follows: a *monitor* measures the output signal of the *plant*, which is the system to be controlled; The measurements are sent to a *controller*, which compares the output signal with a *target* value and maps the difference between them (i.e., *control error*) to a *control signal*; An *actuator* adjusts the behavior of the plant according to the control signal. 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.

For Example 2, components in the feedback loop can be mapped to the following concrete components in DSMS. The plant to be controlled is the query engine of the DSMS and the actuator is the existing load shedding algorithm that adjusts load injected into the plant. In addition, we have a monitor that measures the output signal and a controller to generate the control signal. The unpredictable arrival patterns and processing costs of tuples are all treated as disturbances. In this loop, the output signal y is the average processing delay of all tuples arrived in a control period, and the control signal u is the desirable incoming data rate.

We can easily see that the most critical part of the control loop is the controller, which determines the quantity of the control signal based on the control error. Control theory is the mathematical foundation on how to design controllers based on the dynamic features of the plant.

[2]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. For problems of interest in this paper, we show in [14] that open-loop control does not work.

[3]This period is called *control period*. We discuss how to choose control period in Section 5.3.

3. SYSTEM MODELING

Rigorous control theory is built upon our 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. Linear systems have been well-studied in the control community. Generally, the dynamic behavior of a single-input-single-output (SISO)[4] linear time-invariant (LTI) system can be modeled by a transfer function between the input u and output y in frequency domain:

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

where the quantity 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:

$$\begin{cases} x &= Ax + Bu \\ y &= Cx + Du \end{cases} \quad (2)$$

where A, B, C, and D are model parameters. We can easily find the corresponding transfer function from Eq.(2): $G(s) = C(sI - A)^{-1}B + D$. 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[5]. But all the analysis of the system depends on model accuracy. Depending on the available information about the system, there are different ways to obtain the model.

When the physical meaning of the system is clearly known, the model structure and all the parameters in it can be accurately derived. However, for complex systems such as a DBMS, the analytical model may be difficult, if not impossible, to generate. Actually, lack of mathematical model has been listed as the No.1 obstacle in building self-tuning databases [17]. Fortunately, various *system identification* techniques have been established to generate approximate models that are useful in designing effective control loops.

For many systems such as databases, some of the parameters in $a = [a_1, a_2, \cdots, a_n]^T$ and $b = [b_1, b_2, \cdots, b_n]^T$ are unknown, although they are fixed or only vary slowly. These unknown parameters can be estimated based on the measured input u and output y. For example, we estimate a and b by $\hat{a}$ and $\lambda - \hat{b}$, respectively, where $\lambda = [\lambda_1, \lambda_2, \cdots, \lambda_n]^T$ is an auxiliary parameter vector. Quantities $\hat{a}$ and $\hat{b}$ are dynamically updated by an adaptation law such as

$$\hat{a} = -\frac{(\hat{y} - y)w}{1 + w^T w},$$

where $\hat{y} = [a^T, b^T]w$ and w is the auxiliary states generated by λ, u, and y. It is proven that the estimation will converge to the actual parameters a and b [13]. For low-order systems, the constant parameters can often be identified by experiments. For example, we can feed the real system with a constant or sinusoidal input of different frequencies. By comparing the measurements of the actual output with the theoretical derivation of the output as a function of unknown parameters and input frequency, the parameters can be derived by some calculations. Even for a high-order system, we can approximate it by a low-order model because the impact of higher order dynamics on system performance is often minor so that it can be ignored.

With careful choice of input data, system identification methods can give surprisingly accurate models of complex systems. In our work on load shedding in DSMSs, we derive a model for the Borealis DSMS we use for experiments by system identification. The runtime architecture of Borealis consists of a number of query operators that form a data flow network [14]. Each operator in the network i is characterized by its CPU cost c_i, and selectivity s_i. We define quantity c as the average processing (CPU) of all tuples: it is easy to see that c is a function of $c_i, s_i, \forall i$. For now, we treat c as a constant. We discuss how to deal with time-varying c in Section 4. Due to the complex structure of the operator network, it is hard to generate an analytical model that describes the relationship between system input (tuple arrival rate f) and output (average tuple delay y). To study the dynamics of the Borealis system, we feed it with synthetic data whose arrival rate follow step and sinusoidal functions of time. By studying the responses of the Borealis system to such inputs, we conclude that it can be described by the following first-order linear model:

$$G(s) = \frac{\alpha c}{s - 1} \quad (3)$$

where α is a constant. We also verified this model with a rich set of other inputs and the experimental results converge to a fixed α value very fast. In our problem, controller design becomes straightforward with this linear model.

4. CONTROLLER DESIGN

The model of a system only shows an open relationship between the input u and output y. In order to have the output follow a desired reference y_d, we need to design a controller that tells us how to change the input u. How to change u depends on how good the output y is approaching to y_d, so the controlled input u should be a function of y and y_d. Depending on how much information about the system is available, a wide variety of controller design methods exist.

When the dynamical linear model of the system is known, there are several well-known methods to design the controller with guaranteed performance. In addition to stability, other important performance parameters related to our problems include: 1) *Steady-state error*: the difference between output and the control target when system is at steady state; 2) *Convergence rate*: shows how fast the system goes back to steady state in response to disturbances; and 3) *System damping*: shows how smoothly the system converges to steady state in response to disturbances. Smaller damping means more oscillations during this process.

For example, given the transfer function $G(s)$ in frequency domain (Eq.(1)), the system dynamics all depend on the roots of the denominator (poles) and the roots of the numerator (zeros) of $G(s)$. We can add more zeros or poles to the controller (with transfer function $C(s)$). Then the transfer function of the overall close-loop system is changed to $\frac{G(s)C(s)}{1+G(s)C(s)}$, whose zeros and poles can be adjusted by the controller $C(s)$ in order to have the output meet performance requirements. The most commonly used controller

[4]Both input and output can also be of multiple dimensions.

[5]Stability means that bounded input can only gives rise to bounded output. Controllability of a state means that the sate 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.

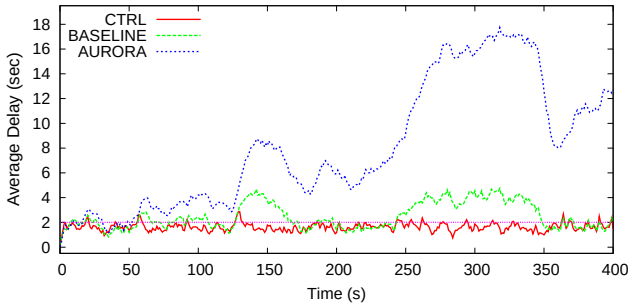

Figure 2: Performance of different load shedding methods.

only has one zero and one pole: $C(s) = a\frac{s+c}{s+b}$, which includes three simple components: proportional, integral, and derivative (PID). To achieve the desired performance goals, we only need to adjust the three parameters a, b, and c using mathematical tools such as *root locus* and *bode plot* [6].

With the model in Eq.(3), we designed a controller to guide load shedding in the Borealis system based on pole placement. In this design, we evaluated controller performance by the speed and smoothness, or *convergence rate* and *damping*, of control system's responses to disturbances. For example, we set convergence rate to three control periods and damping to zero in our design. This means that the system, in response to dynamics, would converge to $1 - \frac{1}{e} \approx 63\%$ of the desired value in 3 control periods and to 98% in 12 periods, and no oscillations will occur. The resulting controller is as follows:

$$u(k) = \frac{1}{c\alpha}[b_0 e(k) + b_1 e(k-1)] - au(k-1). \qquad (4)$$

where b_0, b_1, a are controller parameters that can be analytically obtained, $e(k), e(k-1)$ are the control errors in the present and previous control period, respectively.

Experimental results show effectiveness of the controller (see [14] for more details). In Fig. 2, we compare the performance of our control approach based on Eq.(4) (CTRL), the current rule-of-thumb solution in Borealis (AURORA), and a simple control solution without rigorous controller design (BASELINE). We feed the system with bursty data streams with variable unit processing costs. We can easily see that, the recorded system outputs (average tuple delay within control period of one second) in CTRL are very close to the target value of two seconds for most of the time. The AURORA solution leads to a unstable system: tuples delays increase unboundedly. Performance of the BASELINE solution goes between the above two methods.

When the system model is only partly known, the unknown part of the model should be dealt with during the controller design. For example, when the unknown part is additive Gaussian noise, the Linear Quadratic Gaussian (LQG) controller can be calculated by solving an algebraic equation and it can optimize the problem of minimizing the error between output y and reference y_d. When some parameters in the object model are unknown, the controller can only be designed for the estimated model by identification. Since the adaptive identification of the model introduces extra dynamics, the controller design is more complicated and belongs to adaptive control theory. In Example 2, we model the system by treating the average tuple processing cost c as a constant. However, c is a time-varying factor in practice, although we can safely assume it changes slower than the data arrival rate. To solve this problem, we design an adaptive controller with two feedback control loops: the original inner loop (Eq.(4)) and an outer loop that captures the changes of c (thus denoted as $c(k)$ for period k).

Adaptive control only considers the case when unknown parameters are constant or are varying very slowly. When there are unstructured noises which are varying fast, they might make the whole adaptive system unstable even if such noises are bounded. To solve this problem, robust control theory has been actively studied. We briefly introduce several typical robust control methods. The robust adaptive control theory [7] is an extension of the existing adaptive control theory. It takes the impact of unstructured uncertainties into consideration by modifying the parameter update law so that the stability of the overall system is still guaranteed. The H-infinity control [5] is an optimal control methods that optimizes the H-infinity norm of the transfer function between the system performance index and the unstructured noise. It reduces the impact of bounded noise on the system performance to be arbitrarily small. The μ-analysis theory [4] has been developed to further investigate the detailed structure of the noise and its impact on the system performance. The variable structure control theory [9] takes another approach to increase the robustness of a system. Instead of directly working on the uncertainties in the system model, it forces the controller to switch between some different structures. Which structure to select dependents on the current system performance.

For Example 3, we are currently developing a state-space model by taking the access rates of all disks as the states. The output of the model is the same as states. We then design a controller based on the current measurement of states so that the the access rate of every disk is approaching to the same constant. The control input directs us to reallocate data blocks. When the model is not accurate, adaptive control or robust control methods can be used.

5. ISSUES

Although control theory provides a sound theoretical background for the aforementioned problems, its application in self-tuning databases is by no means straightforward. 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. This requires careful mathematical study from a control theoretical viewpoint. In this section, we discuss some of the challenges we identified.

5.1 Lack of real-time output measurement

Response time and processing delay are important performance metrics in database systems (recall Examples 1 and 2 in Section 1). 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!). For instance, in Example 2, the output signal of our controller should be the delay of tuples that have just entered the system when we calculate $u(k)$. However, at time k, we can only measure the delay of those that entered the sys-

tem some time ago. This is a very interesting challenge to control theory as it does not exist in conventional control systems where the controlled signal can always be measured when we need it.

One of the solutions for this problem is to use other metrics that are measurable and derive the output signal from these metrics. Again, a model that maps the metric to the real output signal is needed. This model is called a *transducer* in [6]. Due to modeling errors, it is inevitable that the adaption of the transducer adds estimation errors to the output signal. We need to make sure that it does not change the characteristics of the controller such as its stability.

In the case of load shedding in DSMSs, we can easily modify the Borealis system to accurately record the number of outstanding data tuples q. This can be done by just counting all the inflow/outflow tuples. We know that at any time, $c(k)$ values can be accurately estimated. Therefore, instead of using a measurement of delay y as the feedback signal, we use an estimation of y as follows:

$$\hat{y}(k) = qc(k) \tag{5}$$

The intuition of the above equation is: because Borealis utilizes a round robin policy to schedule the execution of operators, the incoming tuples are expected to be processed after the outstanding tuples. Naturally, Eq.(5) adds estimation errors to the closed-loop. Fortunately, our controller is still found to be robust [14].

The usage of transducers is almost always *ad hoc*. It is interesting to see how this can be dealt with in a more systematic way. Integrating the delays into the model seems to be a viable way to approach this problem and it is a significant challenge to the control theory community.

5.2 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 desirable value. However, in database systems, we are not always sure that the control signal given by the controller can be correctly generated by our actuator. In this section, we discuss some scenarios with the above difficulty.

Sometimes the control signal is implemented as a modification of the original system input signal that is unpredictable beforehand. In Example 2, given the desired data flow rate $u(k)$ obtained from the controller (i.e., control signal), the task of the load adaptor (actuator) is to cut the incoming data stream (with rate f) such that the the actual number of tuples accepted into the system is close to $u(k)$. A straightforward way to implement the load shedder is to cut load before the tuples enter the DSMS and treat all input streams equally. For this purpose, we set a shedding/filtering factor β ($0 \leq \beta \leq 1$) to all the data streams. When Borealis receives a tuple, it flips an unfair coin with head probability $1 - \beta$. A tuple is admitted only when the coin shows head. At the end of period k, β should be determined as follows:

$$\beta = 1 - [u(k)/f(k+1)]. \tag{6}$$

However, $f(k+1)$ is unknown when we calculate β. We use its value in the current period $f(k)$ as an estimation. This, again, brings errors to the control signal. Fortunately, when the convergence rate is higher than the changing frequency of input f, the current controller can compensate for this estimation error (Fig. 2). Note that we add one more zero to the plant transfer function by replacing $f(k+1)$ with $f(k)$, we can design a slightly different controller with provable stability (details skipped here).

Another source of errors for the control signal is caused by the fact that the actuator is implemented as a combination of multiple knobs. For example, a more advanced load shedder in Borealis can discard tuples from any operators in the operator network. Given a control signal $u(k)$, the load shedder will choose the best places to discard load such that the tuples lost are of lower importance to the query results and the real load put on the system is close to $u(k)c$. The problem here is that we are not sure how close the real load is to the desirable value. From control theory point of view, we can identify the relationship between a control input and the load by a mathematical model, called *observer*. If the output of the observer does not match the real load, the difference between them is used to adjust the observer so that the estimation error is reduced. The system identification techniques mentioned in Section 3 can be employed to adjust the observer.

In problems such as in Example 1 and 3, the system model is continuous but the input is a discrete knob switching between multiple values. This controller design problem can be handled by a variable structure control method, called sliding mode control. Basically, we first define a sliding space s, which is typically the difference between the actual output and the desired output (e.g., the response time). Based on an approximated model of the system, we can design a discrete control input that force the sliding space $s = 0$. An advantage of sliding mode control is that it is pretty robust to noise due to inaccurate model or measurement.

5.3 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 closed-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 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, our 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 [16]. 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 Example 2, this means the control period should be at most half of the width of the spikes in input rate. In practice, a sampling frequency that is one order of magnitude larger than the input signal frequency is often used. Therefore, a high sampling frequency is preferred to capture the time-varying properties of the system and input data.

2. Uncertainties in system signals. Computing systems are inherently discrete. In our 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. In our solution to Example 2, the output signal $y(k)$ and processing cost $c(k)$ are defined as the corresponding mean values of a series of tuples. Taking such expectations can eliminate uncertainties brought by the heterogeneity of individual tuples. A larger sampling period (low sampling frequency) is preferred as more smoothing effects can be expected. For example, when tuple processing cost is in the order of milliseconds, setting control period to a fraction of one second level would give us tens to a few hundreds of samples to approximate the real values of $y(k)$ and $c(k)$. For higher sampling frequencies, we get fewer samples and may encounter more significant measurement errors.

3. CPU processing capability. Another factor that inhibits the use of very high sampling rate is the cost of controller itself. Unlike conventional control systems, the database system does not have independent sensing mechanisms and control processor, i.e., all the feedback signals as well as the control algorithms are conducted by the same CPU as the database itself. Too 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.

5.4 Non-linear systems

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 [15].

There is no generic method to control non-linear systems, either. Some nonlinear models can be completely linearized [8] so that any linear control design methods can be applied. But in general, the nonlinear controller design has to be done case by case because each practical nonlinear model can often be linearized in one way or another according to its physical meanings. For the worst case when the object model is regarded as a black box, intelligent control theory can be used. The typical intelligent control method is based on the fuzzy model of the object. Even when the fuzzy model is not available, we can still design a rough fuzzy controller based on the experience gained while manually manipulating the system. Although fuzzy controller may work as well as the human being, it is not able to achieve the best performance by itself. It is expected to work together with other classic control methods when at least part of the system can be theoretically modeled.

6. CONCLUSIONS

In this paper, we propose the idea of using feedback control theory for problems in the field of self-tuning databases. Via concrete examples and partially accomplished work, we see that various control techniques can be used to model and solve different problems in this field. We also noticed that there are some major challenges in applying control theory to problems in this field. Thus, we believe our explorations can give rise to many opportunities to conduct synergistic research between the database and control engineering communities to extend our knowledge in both fields.

7. REFERENCES

[1] Kurt P. Brown, Manish Mehta, Michael J. Carey, and Miron Livny. Towards automated performance tuning for complex workloads. In *VLDB*, pages 72–84, 1994.

[2] S. Chadhuri and G. Weikum. Foundations of Automated Database Tuning. In *Procs. of ICDE*, pages 104–104, April 2006.

[3] Surajit Chaudhuri, Benoît Dageville, and Guy M. Lohman. Self-managing technology in database management systems. In *VLDB*, page 1243, 2004.

[4] J. Doyle, A. Packard, and K. Zhou. Review of LFTs, LMIs, and μ. In *30th IEEE Conference on Decision and Control*, pages 1227–1232, 1991.

[5] J. Doyle *et al.* State-Space Solution to Standard H-∞ and H-2 Control Problem. *IEEE Trans. Automatic Control*, 34(8):831–847, 1989.

[6] J. L. Hellerstein, Y. Diao, S. Parekh, and D. Tilbury. *Feedback Control of Computing Systems*. Wiley-Interscience, 2004.

[7] P. A. Ioannou and A. Datta. Robust Adaptive Control: A Unified Approach. *Procs. of IEEE*, 79(12):1736–1768, 1991.

[8] A. Isidor. *Nonlinear Control Systems*. Springer-Verlag, Berlin, 1989.

[9] Y. Itkis. *Control Systems of Variable Structure*. Wiley, New York, 1976.

[10] S. Keshav. A Control-Theoretic Approach to Flow Control. In *Procs. of SIGCOMM*, September 1991.

[11] B. Li and K. Nahrstedt. A Control-Based Middleware Framework for Quality of Service Adaptations. *IEEE Journal of Selected Areas in Communications*, 17(9):1632–1650, September 1999.

[12] C. Lu, J. Stankovic, G. Tao, and S. Han. Feedback Control Real-Time Scheduling: Framework, Modeling, and Algorithms. *Journal of Real-Time Systems*, 23(1/2):85–126, September 2002.

[13] S. Sastry and M. Bodson. *Adaptive Control: Stability, Convergence, and Robustness*. Prentice Hall, 1989.

[14] Y.-C. Tu, S. Liu, S. Prabhakar, and B. Yao. Load Shedding in Stream Databases: A Control-Based Approach. In *Procs. of VLDB*, September 2006.

[15] L. X. Wang. *Adaptive Fuzzy Systems and Control: Design and Stability Analysis*. Prentice Hall, NJ, 1994.

[16] W.D.Stanley. *Digital Signal Processing*. Reston Publishing Co., 1975.

[17] G. Weikum, A. Moenkeberg, C. Hasse, and P. Zabback. Self-Tuning Database Technology and Information Services: from Wishful Thinking to Viable Engineering. In *Procs. of VLDB*, pages 20–31, August 2002.

[18] Philip S. Yu and Douglas W. Cornell. Buffer management based on return on consumption in a multi-query environment. *VLDB J.*, 2(1):1–37, 1993.