Using Control Theory for Load Shedding in Data Stream Management

Yi-Cheng Tu† Song Liu‡ Sunil Prabhakar† Bin Yao‡ William Schroeder†

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

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

Abstract

Database performance can be greatly affected by environmental and internal dynamics such as workloads and system configurations. Existing strategies to maintain performance under such dynamics are often found to have poor robustness. To remedy this problem, we propose a systematic solution that takes advantages of formal feedback control techniques. In this demo, we show how the control-based solution derived from a dynamic DSMS model can be utilized to guide load shedding with the target of maintaining data processing delays.

1 Introduction

The ability to control the performance of database systems is a critical consideration in achieving a high level of user satisfaction. However, such an ability is greatly hampered by environmental and internal dynamics. For example, when large bursts of queries arrive such that the system is overloaded, the response time of all queries may increase dramatically. Leaving database configuration in response to dynamics to human operators is not an acceptable solution for obvious reasons. Therefore, the design of self-tuning DBMSs that automatically adapt to dynamics becomes an important research topic. In this demo, we show how formal feedback control techniques can be utilized to guide the adaptation of DBMS behavior for the purpose of maintaining performance at a desirable level. As compared to the *ad hoc* solutions currently used in existing systems, our strategy achieves more effective rejection of dynamics and a higher level of robustness.

Our demo system is built on top of a real-world Data Stream Management System (DSMS), a new type of DBMS that handles infinite streaming data and processes continuous queries against such data. The performance metric of interest in this demo is tuple processing delay, i.e., the time between the tuple's arrival to and departure from the DSMS. Processing delay is the most critical Quality-of-Service (QoS) parameter in many applications (e.g., tracking of stock prices and network monitoring). Processing delay can increase unboundedly in case of overloading. On the other hand, meaningful results can still be obtained with partial data loss. Therefore, in order to control the processing delay in case of overloading, we can discard some of the data. This strategy is called *load shedding* [2]. The real challenge is: *how to perform load shedding to meet delay targets with the smallest possible amount of data discarded*? The constraint of minimal data loss is important because it is reasonable to assume that data loss is positively related to degradation of query accuracy, which is another factor that affects user satisfaction.

State of the Art Load shedding in DSMSs has been addressed in a number of studies. Close in spirit to our work is the Aurora system [2] (now becoming part of the Borealis system [1]) where load shedding is performed for the purpose of maintaining QoS in query processing. Two critical questions about load shedding, as identified by [2] and confirmed by our experience, are: 1) *When* is the time to shed load? and 2) *How much* data to discard given the current load? A simple rule-of-thumb algorithm was developed in Aurora [2] to answer the above questions. However, in a dynamic environment where both factors keep changing, this approach may either under-react and thus make the DSMS unstable (i.e., QoS deviates unboundedly from the desired value) or over-react by discarding too much load.

Our contributions In this demo, we present a delay-driven load shedding framework to remedy the above problems. Developed based on our ideas presented in [3, 4], this framework differs from the aforementioned *ad hoc* solution in the following aspects: 1) the framework is based on a dynamic model that describes the quantitative relationship between system input (i.e., load flow into the DSMS) and output (i.e., tuple processing delay); 2) the load shedding decisions are made by a *Controller* module derived from formal feedback control theory. The controller is designed to be *robust*, meaning that desired system performance can

1-4244-0803-2/07/$20.00 ©2007 IEEE

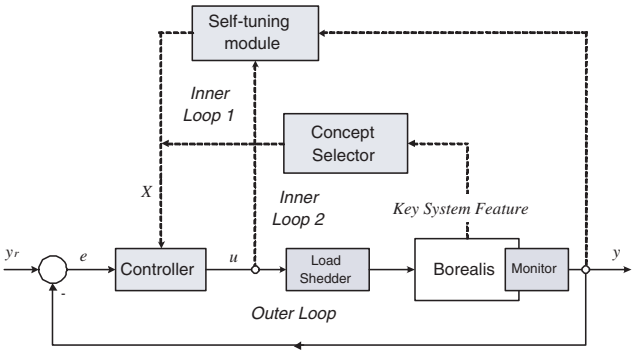

Figure 1. System architecture of C-DSM.

be guaranteed under a wide range of external disturbances (e.g., bursty arrivals); and 3) our framework can also automatically adapt to high-level dynamics such as concept-drifts inside DSMS by changing its own configuration.

2 System Architecture

We name our control-based load shedding framework C-DSM, short for Control-enhanced Data Stream Manager. The runtime architecture of C-DSM is illustrated in Fig. 1. Various components of this architecture are organized into three feedback loops: an *outer loop* and two *inner loops*. Although we can incorporate C-DSM to any DSMS, we use the Borealis data stream manager [1] as an example in the following discussions.

Outer loop components. The outer loop is the main entity of C-DSM. In this loop, the DSMS is the system to be controlled and we augment the DSMS with a *monitor* to keep track of the system output. In this study, we use average tuple processing delay y as the output signal. The output is compared with a target delay y_r and the difference between them is the *control error* (denoted as e). The target delay y_r is set by the database administrator according to user requests. The operative goal of C-DSM is to let the system output y track the target value y_r exactly under the dynamics of input load and system configuration. The focal point of the outer loop is the *controller*, which maps the control error e to a control action u. In our case, u is defined as the desirable load to enter the DSMS. Given the control action generated by the controller, the functionality of the *load shedder* is to implement the control action. Specifically, it determines how to discard data such that the load injected into the DSMS is u.

System modeling. Feedback controller design is performed on the basis of a thorough understanding of the system to be controlled. Specifically, a dynamic model that quantifies the effects of system inputs on measured outputs is needed. For complex systems such as a DSMS, we often use both analytical and experimental methods to develop the model. Although human intervention is necessary in system modeling, we have encapsulated all the procedures that can be automated into an offline component called *system modeler*. This module will test the raw system with various types of inputs and provide suggestions about potential system models to the database administrator.

Controller design. We design our controller via rigorous system modeling and analysis in accordance with feedback control theory. A great advantage of our control-based approach is that we can define parameters to formally describe the performance of load shedding algorithms using control metrics. Important parameters related to our problem include *Stability*, *Steady-state error*, *Convergence rate*, *System damping*, and so on. Control theory enables us to design controllers with desired performance target and these targets are guaranteed at runtime. Controller design is an offline process. For any given system model and set of performance parameters, we only need to go through the designing process once.

Inner loop components. Inner loop 1 of C-DSM is designed to achieve better control of the system when the system model is time-variant. We add a *self-tuning module* that performs online modifications of the controller based on real-time estimations of system parameters. The latter requires system output y and control signal u as inputs to the self-tuning module, which forms the first inner loop with X representing the updated controller design. More abrupt changes of system characteristics (e.g., scheduling policy) are handled by inner loop 2. At runtime, a *concept selector* senses the changes in the system running state, and selects the appropriate controller design accordingly.

3 The Demonstration

We implemented a C-DSM prototype on top of the open-source Borealis data stream management system. In our demo, we will run C-DSM on a single server, which accepts and processes streaming data generated from another machine. Our demo will illustrate the following key features of C-DSM: 1) Offline operations; 2) Performance of the system compared to ad-hoc solutions; and 3) Robustness of the control-based solution.

References

[1] D. J. Abadi *et al.* The Design of the Borealis Stream Processing Engine. In *Procs. of CIDR*, January 2005.

[2] N. Tatbul *et al.* Load Shedding in a Data Stream Manager. In *Procs. of VLDB*, pages 309–320, August 2003.

[3] Y.-C. Tu *et al.* Control-based Quality Adaptation in Data Stream Management Systems. In *Proceedings of DEXA*, pages 746–755, August 2005.

[4] Y.-C. Tu, S. Liu, S. Prabhakar, and B. Yao. Load Shedding in Stream Databases: A Control-Based Approach. In *Procs. of VLDB*, pages 787–798, September 2006.