

IBM Research Report

Load Balancing Using MIMO Linear Control

**Yixin Diao, Joseph Hellerstein, Adam Storm*, Maheswaran Surendra,
Sam Lightstone*, Sujay S. Parekh, Christian Garcia-Arellano***

IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

*IBM Toronto Laboratory
8200 Warden Avenue
Markham, Ontario L6G 1C7



Research Division

Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Load Balancing Using MIMO Linear Control

Abstract

Load balancing is generally a resource allocation problem that can be solved using constraint optimization methods. However, system dynamics are usually not considered. In this paper we propose an optimization algorithm that manages system dynamics using multi-input multi-output dynamic state feedback. In particular, we study the optimization problem in the context of a database memory allocation problem where the cost function value is unknown but the gradient is known. Our studies of a DB2 Universal Database Server in an OLTP benchmarking environment indicate that our approach can be effective in practice.

1 Introduction

The advent of eCommerce has created a need for responsive and cost-effective information technology (IT) services. However, the increasing complexity of computing systems has resulted in a correspondingly larger human effort for system configuration, especially the optimization of configuration parameters. Enterprise level software, especially middleware, has tens to hundreds of configuration parameters. For example, IBM's DB2 Universal Database Server has approximately 100 to 200 configuration parameters (e.g., buffer pool sizes, time delay for writing commit records, maximum number of database applications). The challenges here are well recognized, as evidenced by efforts such as IBM's autonomic computing initiative to develop self-managing systems [1].

Stimulated by the complexity and importance of developing autonomic computing systems, significant research efforts have been attracted for both theoretical and practical reasons. In particular, control theory is emerging as a promising cornerstone to provide rigorous mathematics for designing and analyz-

ing an autonomic controller, which reduces the work of system administrators and provides guaranteed control performance. Research results of applying control theory to computing systems include flow and congestion control [2, 3, 4, 5], differentiated caching and web service [6, 7], multimedia streaming [8], web server performance [9], and email server control [10, 11].

Although many system resource management problems can be formulated as regulation or tracking problems where the policy objectives are known as reference values, some management concerns optimizing a service level metric (e.g., minimizing the system response time, maximizing server utilization) rather than regulate it, which typically means that an optimization routine is required. [12] describes a system that performs on-line optimization of a web server using hill climbing techniques. However, the approach taken requires a detailed knowledge of the system being optimized in order to construct the queueing models. A similar concern arises with the approach in [13] who consider how to maximize profits based on queueing-theoretic formulas. [14] proposes a fuzzy control approach to minimize response time using a combination of feedback control system and qualitative insights into the effect of tuning parameters on QoS. However, only one configuration parameter is considered (online optimization of multiple configuration parameters is considered in [15] using the direct search method). [16] presents case studies of using randomly generated configuration settings for application servers. This approach is simple and generates good results for off-line parameter optimization, but the absence of a guided search may turn out to be problematic for online optimization. A further concern with all of the foregoing is that none of the approaches consider the system dynamics that are essential for on-line optimization.

In this paper we propose an optimization algorithm that manages system dynamics using multi-input multi-output dynamic state feedback. In particular, we study the optimization problem in the context of a database memory allocation problem where the cost function value is unknown but the gradient is known. We apply our controller to IBM's DB2 Universal Database Server since automated configuration of databases is a pressing issue [17]. While there have been many efforts in this area (e.g., [18, 19]), our approach differs in that it requires no prior knowledge of the target system being optimized, although we do require access to the sensors (e.g., disk I/O time measurements) and effectors (e.g., buffer pool sizes).

The remainder of the paper is organized as follows. Section 2 formulates the optimization problem in

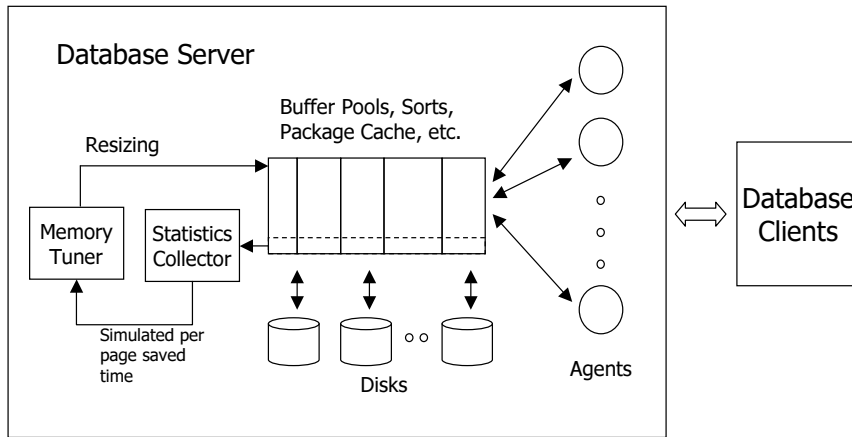


Figure 1: Schematic architecture of database memory management.

the context of database memory management. Section 3 details the control architecture and algorithms. The experimental results are shown in Section 4 to demonstrate the effectiveness of the proposed controller with the application to a DB2 Universal Database Server. Our conclusions are contained in Section 5.

2 Problem Formulation

Load balancing is a class of constraint optimization problems where the total resource is constrained and the resource requirements from different consumers need to be balanced in order to optimize the overall utility function.

In this paper we consider the load balancing problem in the context of database memory management. Figure 1 illustrates the schematic architecture of database memory management. Database clients interact with the database server through a number of agents for accessing data stored in the tables. The data and tables are stored in the disks but also cached in buffer pools, sort memory, package cache, and other components in the memory for fast access. Database performance is largely impacted by memory configuration. A proper configuration can reduce costly disk I/O time. Database memory can be allocated or reallocated to memory consumers through some configuration mechanism such as configuration parameters or registry variables. Latest database technology enables dynamic update for most important parameters. For example, the buffer pool size can be dynamically altered without stopping and restarting the database to make the change effective. However, optimizing configuration parameters is

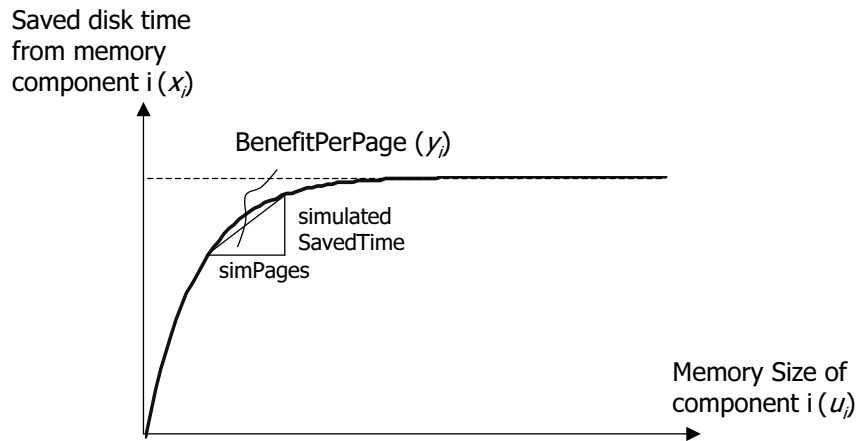


Figure 2: Illustrative relationship between saved disk I/O time and memory size for individual memory component.

time-consuming and skills-intensive, especially for small and medium sized businesses which lack dedicated DBA (database administrator) and extensive knowledge of database performance tuning, and for the customer workloads that are varying over time and difficult to predict. Therefore, on-line self-tuning memory is desired to achieve optimal performance.

As shown in Figure 1, the memory tuner operates based on the “benefit” data – simulated saved disk access time per simulated-increasing memory page per memory component. Figure 2 illustrates the nonlinear relationship between saved disk time and memory size for memory component i . The saved disk time is increasing when the the memory size is increasing, and becomes saturated when the memory is large enough to hold the whole table(s). We model this nonlinear relationship using an exponential function

$$x_i = a_i(1 - e^{-b_i u_i}) \quad (1)$$

where u_i denotes the size of memory component i , x_i denotes the saved disk read/write time for memory component i , and a_i, b_i are model parameters. We assume the additivity of the saved disk read/write time from multiple memory components. Therefore, for a database with N memory components, the load balancing problem is defined as to maximize the total saved disk time

$$\max z = \sum_{i=1}^N x_i = \sum_{i=1}^N a_i(1 - e^{-b_i u_i}) \quad (2)$$

subject to the constraint of the total available memory

$$\sum_{i=1}^N u_i = c \quad (3)$$

and the existence of minimum size requirement for each memory component

$$u_i \geq d_i. \quad (4)$$

Generally, the above optimization problem can be solved using a variety of techniques, which requires knowledge of the function value (e.g., direct search [20], genetic algorithms) or the knowledge of both function value and gradient (e.g., gradient methods [21]) where the function value is used in line search to determine the step size. However, for the database memory management problem the function value, i.e., the saved disk time, is difficult to evaluate because once the data is in the memory the corresponding disk access time does not exist. Instead, a statistics collector can be built to simulate increasing memory pages and to estimate the corresponding simulated saved disk time. This results in the approximated gradient – so called “benefit”

$$y_i = \frac{dx_i}{du_i} = a_i b_i e^{-b_i u_i} \quad (5)$$

where y_i denotes the measurable benefit value for memory component i . The memory tuning objective is to maximize the total saved disk time from all memory components where only the gradient is known but the function value is unknown.

3 Controller Design

In this section we first formulate the optimization problem as a regulation problem, and then design the state feedback controller with LQR design that considers database transaction throughput and memory resizing cost.

For the optimization problem defined above with total resource constraint, optimizing the objective function is equivalent to equalizing the gradient. For the database memory management problem, this indicates that having the benefits from all memory components equal leads to the maximum total saved

disk time. To see this, substitute the constraint condition (3) into the objective function (2) so that

$$\begin{aligned}\max z &= \sum_{i=1}^{N-1} x_i + x_N \\ &= \sum_{i=1}^{N-1} a_i(1 - e^{-b_i u_i}) + a_N(1 - e^{-b_N(c - \sum_{i=1}^{N-1} u_i)}).\end{aligned}$$

The above objective function is optimized if the gradient is equal to zero, that is,

$$\begin{aligned}\frac{\partial z}{\partial u_i} &= a_i b_i e^{-b_i u_i} + \frac{\partial a_N(1 - e^{-b_N(c - \sum_{i=1}^{N-1} u_i)})}{\partial u_i} \\ &= a_i b_i e^{-b_i u_i} - a_N b_N e^{-b_N u_N} \\ &= y_i - y_N = 0\end{aligned}$$

for $i = 1, 2, \dots, N - 1$. This gives the optimal memory setting at $y_i = y_j$ for $i, j = 1, 2, \dots, N$. Since z is a convex function, the optimal solution is unique, i.e., the local optimum is also the global optimum.

Note that for either regulation or tracking problems, the reference value is known in advance to the feedback controller. However, for the above optimization problem, the reference is defined in a relative sense, i.e., the benefits from all memory components are equal to each other, but the absolute values are unknown. To convert from the “relative reference” to “absolute reference,” we define the benefit error as

$$e_i = \frac{1}{N} \sum_{j=1}^N y_j - y_i \quad (6)$$

which is a linear transformation from the benefit data and defines the difference between average benefit and individual benefit. Hence, the control objective is to make $e_i = 0$.

Next, we formulate the control problem using the state space model and design the state feedback controller for maximizing the total saved disk time. Consider the following state space model for the database memory system

$$\mathbf{y}(k+1) = \mathbf{A}\mathbf{y}(k) + \mathbf{B}(\mathbf{u}(k) + \mathbf{d}_1(k)) \quad (7)$$

$$\mathbf{e}(k) = \left(\frac{1}{n}\mathbf{1}_{n,n} - \mathbf{I}\right)(\mathbf{y}(k) + \mathbf{d}_2(k)) \quad (8)$$

where $\mathbf{y}(k)$, $\mathbf{u}(k)$, $\mathbf{e}(k)$ are $n \times 1$ vectors for the system state (measured benefit), control input (memory size), and system output (benefit difference), and $\mathbf{d}_1(k)$, $\mathbf{d}_2(k)$ are $n \times 1$ vectors representing disturbance

and noise. The $n \times n$ matrices \mathbf{A} and \mathbf{B} contain state space model parameters, and $\mathbf{I} = \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 1 \end{bmatrix}$,

$\mathbf{1}_{n,n} = \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & & \vdots \\ 1 & \cdots & 1 \end{bmatrix}$ are $n \times n$ matrices used to compute the benefit difference. Note that we represent

the nonlinear system using a linear model to facilitate controller design. However, this linearization may affect the model accuracy and thus control performance. Also note that the system dynamics and interrelationship between the memory components are considered in the model. This provides a more close system representation than a static model that is typically used by most optimization methods.

The control objective is disturbance rejection with the reference input set to 0 for the benefit error. Load balancing is achieved by driving to 0 the difference between the measured benefits from all memory components and the mean of the measured benefits so that the total saved disk time is maximized. To design the dynamic state feedback controller, we augment the system state by considering the integral of error

$$\mathbf{e}_I(k+1) = \mathbf{e}_I(k) + \mathbf{e}(k). \quad (9)$$

The state feedback controller is in the form of

$$\mathbf{u}(k) = \mathbf{K}\mathbf{e}_I(k). \quad (10)$$

However, note that $(\frac{1}{n}\mathbf{1}_{n,n} - \mathbf{I})$ is singular. This makes the control objective cannot be achieved. To see this, derive the closed loop system

$$\begin{aligned} \mathbf{y}(k+1) &= \mathbf{A}\mathbf{y}(k) + \mathbf{B}\mathbf{K}\mathbf{e}_I(k) + \mathbf{B}\mathbf{d}_1(k) \\ \mathbf{e}_I(k+1) &= \mathbf{e}_I(k) + (\frac{1}{n}\mathbf{1}_{n,n} - \mathbf{I})(\mathbf{y}(k) + \mathbf{d}_2(k)) \end{aligned}$$

and the characteristic polynomial is

$$\det \left(z\mathbf{I} - \begin{bmatrix} \mathbf{A} & \mathbf{BK} \\ \frac{1}{n}\mathbf{1}_{n,n} - \mathbf{I} & \mathbf{I} \end{bmatrix} \right)$$

Hence, the controller can be designed to stabilize the system with the desired poles. In the steady state, $\mathbf{e}_I(k+1) = \mathbf{e}_I(k)$ so that $\mathbf{e}(k) = 0$. However, the target system is not fully controllable, i.e., not any desired state can be achieved. To see this, in the steady state we have

$$\begin{bmatrix} \mathbf{y}_{ss} \\ \mathbf{e}_{I,ss} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{BK} \\ \frac{1}{n}\mathbf{1}_{n,n} - \mathbf{I} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{y}_{ss} \\ \mathbf{e}_{I,ss} \end{bmatrix} + \begin{bmatrix} \mathbf{B}\mathbf{d}_1(k) \\ (\frac{1}{n}\mathbf{1}_{n,n} - \mathbf{I})\mathbf{d}_2(k) \end{bmatrix}$$

so that

$$\begin{bmatrix} \mathbf{I} - \mathbf{A} & -\mathbf{BK} \\ -\frac{1}{n}\mathbf{1}_{n,n} + \mathbf{I} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{y}_{ss} \\ \mathbf{e}_{I,ss} \end{bmatrix} = \begin{bmatrix} \mathbf{B}\mathbf{d}_1(k) \\ (\frac{1}{n}\mathbf{1}_{n,n} - \mathbf{I})\mathbf{d}_2(k) \end{bmatrix}$$

Note that $\begin{bmatrix} \mathbf{I} - \mathbf{A} & -\mathbf{BK} \\ -\frac{1}{n}\mathbf{1}_{n,n} + \mathbf{I} & \mathbf{0} \end{bmatrix}$ is singular. \mathbf{y}_{ss} and $\mathbf{e}_{I,ss}$ may not exist for any $\mathbf{d}_1(k)$ and $\mathbf{d}_2(k)$.

To make the target system controllable we consider the constraint on the control inputs and redefine the state space model to separate out the n^{th} node

$$\begin{aligned} \mathbf{y}(k+1) &= \mathbf{A}\mathbf{y}(k) + \mathbf{B}(\mathbf{u}(k) + \mathbf{d}_1(k)) \\ \mathbf{e}(k) &= \left(\frac{1}{n}\mathbf{1}_{n-1,n-1} - \mathbf{I}\right)(\mathbf{y}(k) + \mathbf{d}_2(k)) + \mathbf{1}_{n-1,1}y_n(k) \\ y_n(k+1) &= a_n y_n(k) + b_n(f - \mathbf{1}_{1,n-1}\mathbf{u}(k)) \end{aligned}$$

where $\mathbf{y}(k)$ and $\mathbf{u}(k)$ are $(n-1) \times 1$ vectors, and \mathbf{A} , \mathbf{B} , and \mathbf{I} are $(n-1) \times (n-1)$ matrices. We design the feedback controller for $\mathbf{u}(k)$, while $u_n(k)$ can be derived from $\mathbf{u}(k)$ using the total memory size constraint. Similarly, define the integral of error

$$\mathbf{e}_I(k+1) = \mathbf{e}_I(k) + \mathbf{e}(k)$$

The controller is in the form of

$$\mathbf{u}(k) = \mathbf{K}\mathbf{e}_I(k)$$

The closed loop system is

$$\begin{aligned} \mathbf{y}(k+1) &= \mathbf{A}\mathbf{y}(k) + \mathbf{BK}\mathbf{e}_I(k) + \mathbf{B}\mathbf{d}_1(k) \\ \mathbf{e}_I(k+1) &= \mathbf{e}_I(k) + \left(\frac{1}{n}\mathbf{1}_{n-1,n-1} - \mathbf{I}\right)(\mathbf{y}(k) + \mathbf{d}_2(k)) + \mathbf{1}_{n-1,1}y_n(k) \end{aligned}$$

The characteristic polynomial is

$$\det \left(z\mathbf{I} - \begin{bmatrix} \mathbf{A} & \mathbf{BK} \\ \frac{1}{n}\mathbf{1}_{n-1,n-1} - \mathbf{I} & \mathbf{I} \end{bmatrix} \right)$$

In the steady state, we have $\mathbf{e}(k) = 0$ and

$$\begin{bmatrix} \mathbf{y}_{ss} \\ \mathbf{e}_{I,ss} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{BK} \\ \frac{1}{n}\mathbf{1}_{n-1,n-1} - \mathbf{I} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{y}_{ss} \\ \mathbf{e}_{I,ss} \end{bmatrix} + \begin{bmatrix} \mathbf{B}\mathbf{d}_1(k) \\ (\frac{1}{n}\mathbf{1}_{n-1,n-1} - \mathbf{I})\mathbf{d}_2(k) + \mathbf{1}_{n-1,1}y_n(k) \end{bmatrix}$$

so that

$$\begin{bmatrix} \mathbf{I} - \mathbf{A} & -\mathbf{BK} \\ -\frac{1}{n}\mathbf{1}_{n-1,n-1} + \mathbf{I} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{y}_{ss} \\ \mathbf{e}_{I,ss} \end{bmatrix} = \begin{bmatrix} \mathbf{B}\mathbf{d}_1(k) \\ (\frac{1}{n}\mathbf{1}_{n-1,n-1} - \mathbf{I})\mathbf{d}_2(k) + \mathbf{1}_{n-1,1}y_n(k) \end{bmatrix}$$

Note that $\begin{bmatrix} \mathbf{I} - \mathbf{A} & -\mathbf{BK} \\ -\frac{1}{n}\mathbf{1}_{n-1,n-1} + \mathbf{I} & \mathbf{0} \end{bmatrix}$ will not become singular because of $-\frac{1}{n}\mathbf{1}_{n-1,n-1} + \mathbf{I}$. Arbitrary disturbance rejection can be achieved.

We design the state feedback controller by relating the control performance to quality of service requirement of database systems. The performance of a database system is measured by its throughput, the number of transactions that can be handled by the database server per unit of time. The database memory controller is designed to maximize the total saved disk time so that each transaction can spend less time on disk I/O. A shorter transaction time leads to larger throughput, given that the workload is sufficiently large to generate enough transaction requests. Since maximizing total saved disk time can be achieved through equalizing the benefits from all memory consumers, this leads to the control performance requirements, i.e., zero steady state error, short settling time, small overshoot. On the other hand, there is cost associated with memory resizing for a database system. CPU cycles are needed to move data around. For decreasing memory size, certain data need to be written back to the disk (victimization). While benefit data are quite oscillating due to the stochastic nature of the system, oscillation in the memory size should be avoided to reduce the resizing cost. This leads to the control performance requirement on minimizing control overheads.

We balance the trade-off between short settling times and overreacting to random fluctuations by choosing control gains based on a cost function. Specifically, we use the LQR, or *linear quadratic regulator*

[22], to find the control gains that minimize the following quadratic cost function:

$$J = \sum_{k=1}^{\infty} [e^{\top}(k)e_I^{\top}(k)]Q \begin{bmatrix} e(k) \\ e_I(k) \end{bmatrix} + u(k)^{\top}Ru(k) \quad (11)$$

The cost function includes the control errors $e(k)$, the accumulated errors $e_I(k)$, and the control inputs $u(k)$. The matrices Q and R determine the trade-off between control error and the control gain. The intuition is as follows. If Q is large compared to R , there will be larger control gains so that the controller will react quickly to deviations. On the other hand, if R is large compared to Q , the reverse happens and so the controller responds slowly to such deviations, thereby avoiding over-reactions to random fluctuations.

The control design problem has now shifted to choosing the weighting matrices Q and R for the LQR problem. For database memory management, we start by relating them to the throughput. The larger the throughput deterioration, the larger the weight. For example, if throughput drops 50 transactions due to 0.01 benefit difference, we set $Q = 5^2/0.01^2$, and if throughput drops 50 transactions due to a memory size oscillation with a standard deviation of 100, we set $R = 5^2/100^2$.

4 Simulation and Experimental Results

4.1 Testbed Setup

To assess the applicability of our approach to online optimization of database memory, we study it in the context of IBM’s DB2 version 8.1 which provides a plethora of tuning parameters that can be changed programmatically in an online environment. Among them are some memory related parameters such as buffer pool size, package cache size, and sort heap size, which have drastic impact on database performance. In this paper, we consider buffer pool tuning.

Our evaluations are done using TPC-C, an industry standard online transaction processing (OLTP) benchmark [23]. We used 20 buffer pools to contain data and index for the database tables. Different buffer pools are used to separate table data from index and temporary data. A temp buffer pool will ensure that any temp activity does not negatively hurt the buffer pool’s data caching characteristics.

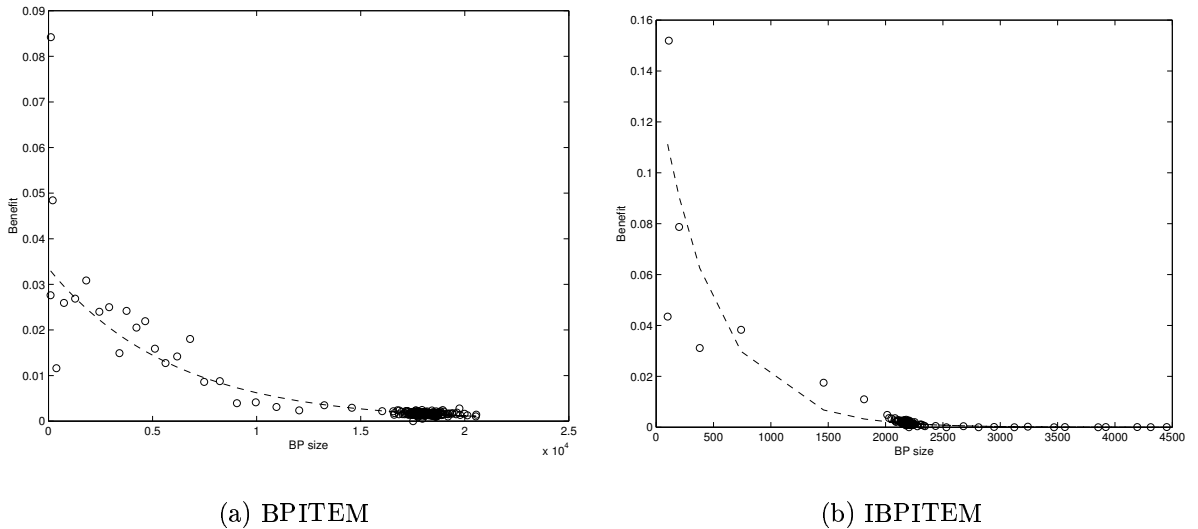


Figure 3: Modeling performance of buffer pools for a TPC-C workload.

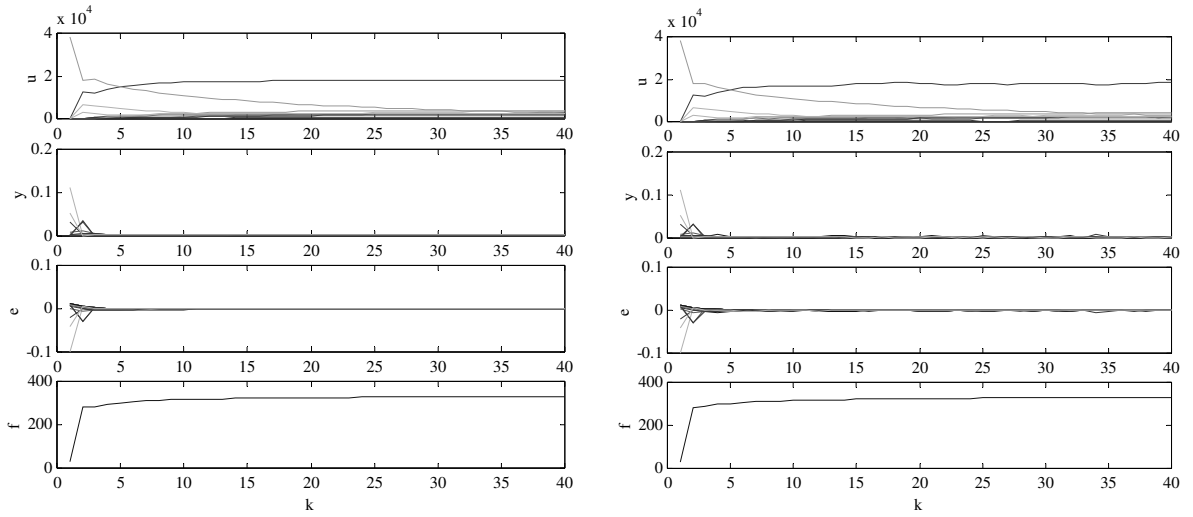
A separate index buffer pool is also often needed to have a high percentage of the index data remain resident in memory, which will accelerate data search. Generally, having larger buffer pool size results in smaller disk access time and thus higher throughput. However, the total buffer pool size cannot go to infinity due the limited memory space.

Our experiments proceed as follows. A set of 50 TPC-C clients were running which executed transactions against the database server according to the TPC-C benchmark specifications. The memory tuner is first used to collect buffer pool data for modeling. Afterwards, the state feedback control algorithm is used by the memory tuner for memory self-tuning.

4.2 Modeling and Simulation Studies

Figure 3 shows the modeling results of two buffer pools BPITEM for holding data in the ITEM table and IBPITEM for holding the index of the ITEM table. The circles indicate the measured data points, and the dashed line indicate the modeling results with exponential models (5). Note that the exponential models provide a reasonable estimate of the buffer pool behaviors.

Next, we conduct simulation studies. The nonlinear models obtained above are used as the truth model, and the linear models obtained from the nonlinear models are used as the design model for designing the controller. The control results are shown in Figure 4 with or without noise. From top to



(a) Without noise

(b) With noise

Figure 4: Simulated control performance of buffer pools for a TPC-C workload.

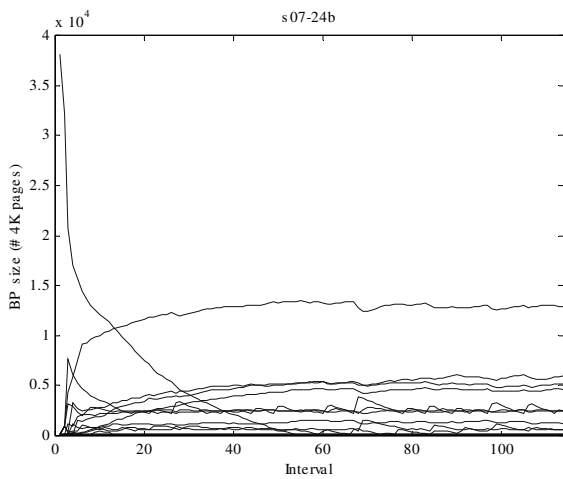
bottom, the buffer pool sizes, the measured benefit, and the benefit errors are shown for the 20 buffer pools to be managed. The plot at the bottom shows the function value of the total saved disk time computed from the model. Note that the buffer pool sizes and the objective function converge within 40 intervals (30-second interval).

4.3 Experimental Results

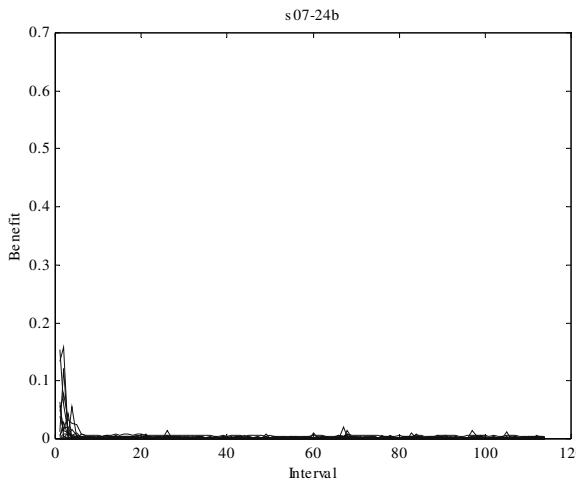
The experimental control results are shown in Figure 5 where buffer pool sizes and benefits are given where the x axis is in the unit of 30-second control intervals. Figure 6 shows the database throughput for the same experiment where the x axis is in the unit of 10-second throughput measurement intervals. Note that the throughput converges within about 10 minutes.

5 Conclusions

Load balancing is generally a resource allocation problem that can be solved using constraint optimization methods. However, system dynamics are usually not considered. In this paper we have proposed an optimization algorithm that manages system dynamics using multi-input multi-output dynamic state



(a) Buffer pool sizes



(b) Buffer pool benefits

Figure 5: Experimental control performance of buffer pools for a TPC-C workload.

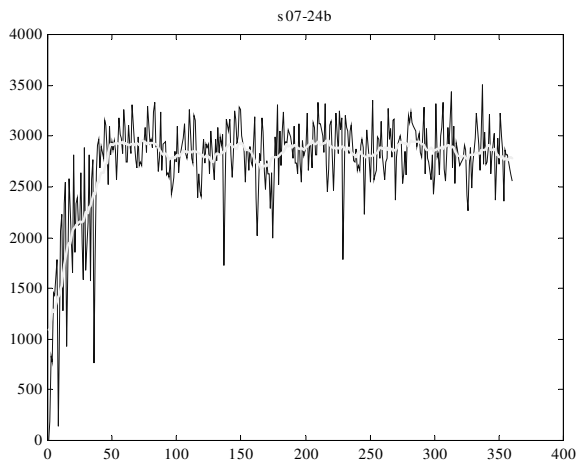


Figure 6: Database throughput for buffer pool experiment of a TPC-C workload.

feedback. In particular, we studied the optimization problem in the context of a database memory allocation problem where the cost function value is unknown but the gradient is known. Our studies of a DB2 Universal Database Server in an OLTP benchmarking environment indicated that our approach can be effective in practice.

References

- [1] IBM, “Autonomic computing: IBM’s perspective on the state of information technology,” *available at <http://www.research.ibm.com/autonomic/>*, 2001.
- [2] D.-M. Chiu and R. Jain, “Analysis of the increase and decrease algorithms for congestion avoidance in computer networks,” *Computer Networks and ISDN systems*, vol. 17, June 1989.
- [3] S. Keshav, “A control-theoretic approach to flow control,” in *Proceedings of ACM SIGCOMM ’91*, Sept. 1991.
- [4] S. Mascolo, “Classical control theory for congestion avoidance in high-speed internet,” in *Proceedings of the 38th Conference on Decision & Control*, Dec. 1999.
- [5] C. V. Hollot, V. Misra, D. Towsley, and W. B. Gong, “On designing improved controllers for AQM routers supporting TCP flows,” in *INFOCOM*, 2001.
- [6] Y. Lu, A. Saxena, and T. F. Abdelzaher, “Differentiated caching services: A control-theoretic approach,” in *International Conference on Distributed Computing Systems*, Apr. 2001.
- [7] C. Lu, T. F. Abdelzaher, J. A. Stankovic, and S. H. Son, “A feedback control architecture and design methodology for service delay guarantees in web servers,” Tech. Rep. CS-2001-06, University of Virginia, Department of Computer Science, 2001.
- [8] B. Li and K. Nahrstedt, “Control-based middleware framework for quality of service applications,” *IEEE Journal on Selected Areas in Communication*, 1999.

- [9] Y. Diao, N. Gandhi, J. L. Hellerstein, S. Parekh, and D. M. Tilbury, "Using MIMO feedback control to enforce policies for interrelated metrics with application to the Apache web server," in *Proceedings of Network Operations and Management*, 2002.
- [10] S. Parekh, N. Gandhi, J. L. Hellerstein, D. M. Tilbury, and J. P. Bigus, "Using control theory to achieve service level objectives in performance management," in *Proceedings of IEEE/IFIP Symposium on Integrated Network Management*, 2001.
- [11] Y. Diao, J. L. Hellerstein, and S. Parekh, "A business-oriented approach to the design of feedback loops for performance management," in *Distributed Systems Operations and Management*, 2001.
- [12] D. Menasce, D. Barbara, and R. Dodge, "Preserving QoS of e-commerce sites through self-tuning: A performance model approach," in *Proceedings of 2001 ACM Conference on E-commerce*, 2001.
- [13] Z. Liu, M. S. Squillante, and J. L. Wolf, "On maximizing service-level-agreement profits," in *Proceedings of the ACM Conference on Electronic Commerce*, 2001.
- [14] Y. Diao, J. L. Hellerstein, and S. Parekh, "Optimizing quality of service using fuzzy control," in *Proceedings of Distributed Systems Operations and Management*, 2002.
- [15] Y. Diao, F. Eskesen, S. Froehlich, J. L. Hellerstein, L. F. Spainhower, and M. Surendra, "Generic online optimization of multiple configuration parameters with application to a database server," in *Proceedings of Distributed Systems Operations and Management*, 2003.
- [16] M. Raghavachari, D. Reimer, and R. Johnson, "The deployer's problem: Configuring application servers for performance and reliability," in *Proceedings of the International Conference on Software Engineering, Portland, OR*, 2003.
- [17] G. Weikum, A. Moenkeberg, C. Hasse, and P. Zabback, "Self-tuning database technology and information services: from wishful thinking to viable engineering," in *International Conference on Very Large Data Bases*, 2002.
- [18] G. M. Lohman and S. S. Lightstone, "Smart: Making DB2 (more) autonomic," in *Proceedings of the 28th International Conference on Very Large Data Bases, Hong Kong, China*, 2002.

- [19] J. Rao, C. Zhang, G. M. Lohman, and N. Megiddo, "Automating physical database design in a parallel database," in *SIGMOD*, 2002.
- [20] J. A. Nelder and R. Mead, "A simplex method for function minimization," *Computer Journal*, 1965.
- [21] D. G. Luenberger, *Linear and nonlinear programming*. Addison-Wesley, Reading, MA, 1984.
- [22] G. F. Franklin, J. D. Powell, and M. L. Workman, *Digital Control of Dynamic Systems*. Reading, Massachusetts: Addison-Wesley, third ed., 1998.
- [23] TPC, "TPC-C: Benchmarking an OLTP solution," in <http://www.tpc.org/tpcc>.