

Simplified Control for Complex IT Systems Using Dimension Reduction

Xue Liu¹, Xiaoyun Zhu², Jianguo Yao¹, Zhikui Wang², Sharad Singhal²

¹School of Computer Science
McGill University, Montreal, Canada, H3A2A7
{xueliu, jianguo}@cs.mcgill.ca

²Hewlett-Packard Labs
Palo Alto, CA 94304
{xiaoyun.zhu, zhikui.wang, sharad.singhal}@hp.com

ABSTRACT

Automated management of complex information technology (IT) applications and systems require dynamic configuration of both application-level and system-level parameters. The existence of large number of tunable parameters makes it difficult to design a feedback controller that adjusts these parameters effectively in order to achieve application-level quality of service (QoS) targets. In this paper, we introduce a new approach for simplified control of complex IT systems based on dimension reduction techniques. It combines online selection of critical control knobs through Lasso — a powerful L1-constrained fitting methods, and adaptive control of the identified knobs. The latter relies on the online estimation of the input-output model with the selected control knobs using the recursive least square (RLS) method and a self-tuning linear quadratic (LQ) optimal controller for output regulation. The results of a simulation study in Matlab are presented to demonstrate the effectiveness of our approach.

Keywords

Information technology, quality of service, dimension reduction, Lasso, adaptive control

1. INTRODUCTION AND RELATED WORK

Today's information technology (IT) systems are becoming larger in scale and more abundant in features, resulting in increasing complexity in their operation. Configuring application-level and system-level parameters for these complex IT systems is a challenging task for IT operators. Current management products typically set these parameters statically via offline analysis and expose interfaces that allow the experienced operators to change the parameter values if needed. However, statically set values are rarely ideal for individual workloads, and it is usually expensive and error-prone for human operators to decide appropriate values for these parameters to meet the quality of service (QoS) targets of individual applications.

For example, the workloads of Internet servers and enterprise applications fluctuate considerably, and statically allocated resources suffer the same fate as dedicated resources: they are either over-provisioned or under-provisioned (i.e. overloaded). Thus modern IT systems and applications expose interfaces to allow dynamic resource allocation or application configuration. But in practice, it is difficult to configure and adjust the relevant parameters properly for a number of reasons. First, there are often tens or hundreds of tunable parameters (knobs) that can be adjusted online. Normally, only some of these parameters are critical to the application QoS for a given workload or under a certain operating condition. To identify in real-time which knobs to tune is a nontrivial task. Second, as the workload

characteristics or system conditions vary over time, the key knobs that affect the performance most may change accordingly. Third, the relationship between an application's performance (e.g., transaction response time or throughput) and its application-level (e.g., number of concurrent threads) and system-level (e.g., CPU allocation, I/O bandwidth, and cache size) configurations is complex.

As a result, it is challenging to properly identify and tune the most critical system-level or application-level parameters in response to changes in workloads or system conditions in order to meet application-level QoS targets. We need automated and adaptive computing solutions to dynamically manage these knobs in the complex IT systems.

In recent years, there has been significant interest in both industry and academia in using feedback control to regulate performance in IT systems. Both [1] and [2] provide an overview of how control theory can be used to guide the design of these feedback loops. However, in most of the prior applications, the control knobs for a particular system are determined offline. The authors in [3] focused on the automatic diagnosis problem where key metrics that are most correlated with a service level violation are identified among all the observable metrics using statistical machine learning. However, these identified metrics are often observables instead of tunables. This means they can not be used as control knobs to correct performance problems that occurred. Similarly, online discovery of critical metrics for a database system was presented in [4], where the selected metrics are then used in the automatic construction of a quantitative model for service level management. However, the authors did not discuss how the metrics or the model can be adapted online or used in controller design in real time. The standard balanced truncation approach for model reduction of dynamical systems aims at reducing the dimensionality of the state space instead of the input space [5]. Therefore, it cannot be used here directly.

To the best of our knowledge, there has not been a single solution where the discovery of key knobs and the control of these knobs are integrated and automated for management of complex IT systems. In this paper, we present such a solution that relies on the synthesis of the following three key techniques:

- a) Online selection of critical control knobs by using a dimension reduction method. This technique automatically reduces the number of control inputs for the target system;
- b) Online estimation of the input-output model with the selected subset of control knobs, using the recursive least squares (RLS) method;
- c) An adaptive linear quadratic (LQ) optimal controller for output regulation.

A prototype of our approach has been implemented in Matlab. We present the results of a simulation study that shows that our approach achieves the following design objectives: 1) It can judiciously select a subset of control knobs for the target system in a dynamic environment. These selected knobs have the most significant impact on the output of the system being controlled; 2) It can dynamically tune the selected control knobs effectively to maintain the system outputs at desired values; 3) As the most critical control knobs change, our system automatically responds by switching to use the new set of knobs to regulate the outputs.

The remainder of the paper is organized as follows. Section 2 presents a motivating example and Section 3 discusses the proposed architecture for simplified control based on dimension reduction. Section 4 describes the design and implementation details of our simplified control solution. We present an evaluation study for our approach in Section 5. Finally we conclude the paper in Section 6.

2. A MOTIVATING EXAMPLE

One motivating example for the dimension reduction problem is a shared hosting platform as depicted in Fig. 1. On such a platform, multiple distributed applications share a common pool of virtualized servers. Each component of each application is hosted inside a virtual container on a shared physical server. A virtual container can be a virtual machine (VM) provided by hypervisor technologies including Xen [6] and VMWare [7] or OS-level virtualization such as OpenVZ [8] and Linux VServer [9]. This type of shared computing paradigm has gained interest in many enterprises data centers as well as in emerging cloud computing environments due to its potential to reduce infrastructure and operational costs. The grouping of application tiers on each physical server can be arbitrary in principle. As an example, Fig. 1 shows a specific scenario where the same tiers from different applications are hosted on the same physical server.

It is critical to offer quality of service (QoS) guarantees for applications hosted on such a shared platform. This is challenging because the resource demand of each application varies depending on the number of concurrent users and the workload mix. As a result, a shared server can become saturated when the aggregate demand from all the application components sharing the server exceeds its total capacity, causing degraded applications QoS. It is possible to design a feedback control system to dynamically allocate the shared resources (CPU, memory, I/O) to each virtual container in order to meet application-level QoS targets, by taking advantage of the resource scheduler interfaces exposed by the virtualization layer. An example of such a control system was presented in [10] where two 2-tier applications were sharing two virtualized nodes and CPU was the only bottleneck resource. In the more general case, both the number of physical nodes (N) and the number of virtual containers per node (M) can be much larger, and multiple resources (R) may become a bottleneck. That means, the large number of potential knobs for the control system ($N*M*R$) can make controller design extremely difficult.

The reduced dimension closed-loop control method introduced in this paper can help address this challenge by allowing the resource control system to automatically identify in real time the subset of resource allocation knobs that are most correlated to

application QoS and then adjust their settings accordingly. It is worth noting that the proposed approach is general and can be used in other scenarios where reduction in the number of control knobs is desirable.

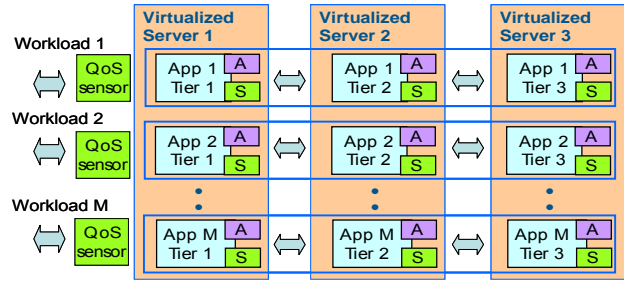


Fig. 1. A virtualized shared hosting platform for multiple multi-tier applications

3. PROPOSED ARCHITECTURE

In this section, we present the system architecture for simplified control using dimension reduction, as shown in Fig. 2. On the right side of this figure is a complex IT system (or our *target system*). It has multiple tuning knobs (*inputs*) that can affect multiple performance metrics of the system (*outputs*). As we have discussed, our design goal is to select online a subset of these inputs that have the most impact on the outputs, and to adjust them dynamically such that the outputs meet the QoS targets (*reference*) of the applications running in the system, in spite of changes in the workloads or system condition.

To achieve this goal, we design three modules for our control system: a *dimension reduction* module, a *model identification* module, and a *controller* module. More specifically, the dimension reduction module employs the Lasso [11][12] method to fulfill the knob selection functionality; the model identification module estimates a linear input-output model for the selected inputs using the recursive-least-squares (RLS) method; finally, the controller module dynamically adjusts the values of the selected knobs based on the estimated model and a quadratic cost function for tracking the output reference. All these three tasks are performed periodically in an online fashion.

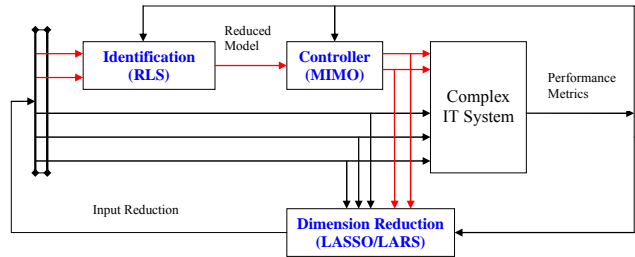


Fig. 2. System architecture for simplified control using dimension reduction

In the example shown in Fig. 2, the target system has 5 potential control knobs. Based on the statistical regression analysis, the dimension reduction module selects 2 knobs that affect the system performance most; the identification module estimates a linear model that correlates these two knobs with the system output; then the controller module computes the optimal values for these two knobs, and provides them to the IT system.

4. DETAILED MODULAR DESIGN

In this section, we elaborate on the specific implementation of the three modules in our control system architecture.

4.1 Dimension Reduction using Lasso

Lasso [12] is a shrinkage and selection method for linear regression. It is a least squares method for choosing a linear model to predict a response variable. Let u_1, u_2, \dots, u_m represent the covariates (system inputs), and y represent the response (system output). Typically we have available a large collection of possible covariates (i.e. m is large) from which we hope to select a parsimonious set for the good prediction of the output. ‘‘Goodness’’ is often defined in terms of prediction accuracy; but parsimony is another important criterion [12] because simpler models provide better scientific insight into the input–output relationship and later on allow more robust controller designs.

Lasso minimizes the usual sum of squared errors between the measured output and the predicted output, with a bound on the sum of the absolute values of the coefficients. Below we briefly describe the idea of Lasso. The description does not intend to be complete. It only serves to help the readers understand how model reduction in our architecture is achieved based on Lasso.

Given a set of input measurements u_1, u_2, \dots, u_m , and output measurement y , Lasso fits the following linear model to the data:

$$\hat{y} = \sum_{j=1}^m \hat{\beta}_j u_j. \quad (1)$$

Lasso assumes that the inputs (u) have zero mean and unit length, and that the output (y) has zero mean. This step is called ‘‘standardization’’, and it can be done through location and scale transformations on the raw data. This is an important step, since for real systems, different inputs (covariates) may have different meanings and units. Using a different unit will usually magnify/reduce the coefficients values. Hence in order to compare the effects of different inputs (‘‘knobs’’) on the output, they need to be standardized.

The criterion Lasso uses is the following constraint optimization:

$$\min \|y - \hat{y}\|_2 = \sum_{i=1}^n (y_i - \sum_{j=1}^m \hat{\beta}_j u_j)^2, \quad (2)$$

$$\text{s.t. } \sum_{j=1}^m |\hat{\beta}_j| \leq t. \quad (3)$$

The sum in the objective function is taken over all observations in the dataset. The bound ‘‘ t ’’ in the constraint is a tuning parameter. When ‘‘ t ’’ is large enough, the constraint has no effect and the solution is just the usual linear least squares regression of y on u_1, u_2, \dots, u_m . However, when smaller values of t ($t > 0$) are used, the solutions are shrunken versions of the least squares estimates. Often, some of the coefficients $\hat{\beta}_j$ are zeros. Choosing ‘‘ t ’’ has the effect on choosing the number of predictors to use in a regression model, and cross-validation is a good tool for estimating the best value for ‘‘ t ’’.

The following example illustrates the effect of ‘‘ t ’’ on the resulting estimates $\hat{\beta}_j$. This example will be used throughout this paper.

Example: Consider an input-output linear system with 7 inputs and 1 output. The inputs are denoted as u_1, \dots, u_7 , and the output is denoted as y . The system is governed by the equation below.

$$\begin{aligned} y(k+1) = & 1.0u_1(k) + 0.1u_2(k) + 0.1u_3(k) \\ & + 2.0u_4(k) + 0.1u_5(k) + 0.1u_6(k) \\ & + 5.0u_7(k) + e(k) \end{aligned} \quad (4)$$

where $e(k)$ represents a (white) noise disturbance.

First we collect system input-output data through simulation. To this end, we excite the system using random inputs. Each system input is selected as a random number time sequence in $[0,1]$. The input noise $e(k)$ is selected as a white noise with mean 0.005. We collect a set of training data with a total of 500 input-output tuples. At instance k , the tuple is: $\{u_1(k), u_2(k), u_3(k), u_4(k), u_5(k), u_6(k), u_7(k), y(k+1)\}$.

Now we apply the Lasso method on the data collected. We vary the tuning parameter ‘‘ t ’’ from 0 to 55 and plot the resulting regression coefficient corresponding to each input. Fig. 3 shows the Lasso estimates $\hat{\beta}_j$ as a function of t . As we can see, the covariates enter the regression equation sequentially as t increases, in the order of $j=7, 4, 1, \dots$, which corresponds to Equation (4), where we can clearly see that the effect of different inputs on the response y is in decreasing order of $u_7, u_4, u_1 \dots$.

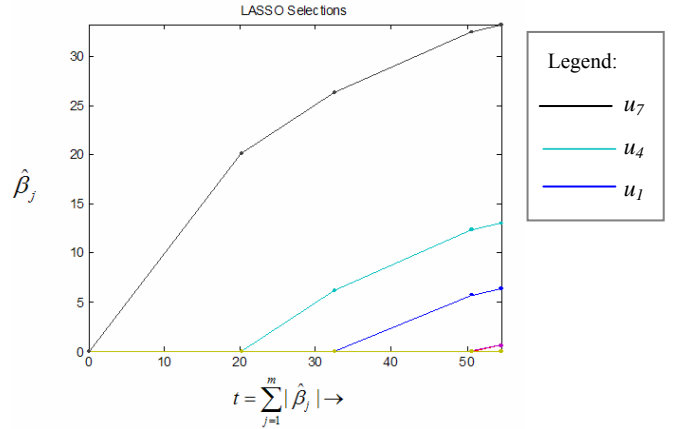


Fig. 3. Estimates of regression coefficients $\hat{\beta}_j$ v.s. t

In this paper, we apply a variant of Lasso called LARS (Least Angel RegreSsion) [11]. LARS is a computationally simpler method than Lasso. LARS uses a simple mathematical formula to accelerate the computation of coefficients. Unlike Lasso, only m steps are required for the full set of solutions, where m is the number of covariates. For detailed discussions of LARS, please refer to reference [11].

In the following, we evaluate the performance in terms of prediction accuracy for the above open-loop system (i.e. without control). We pose the requirement that among the $m=7$ inputs, only n ($n < m$) inputs should be selected to predict the response y . We vary the number of n from 1 to 3, and compute the mean squared errors and the r^2 measure. The higher the r^2 value, the more accurate the prediction is. Table 1 summarizes when n is chosen from 1 to 3, the corresponding reduced input(s) selected by LARS. Table 1 also shows the mean square error between the measured system output y and model-predicted (with reduced input) output and the corresponding r^2 measure.

Table 1: Selected input subsets v.s. n and their prediction performance

n	LARS selected inputs	Mean square error	r^2 measure
1	$\{u_7\}$	360.01	0.73
2	$\{u_7, u_4\}$	131.76	0.90
3	$\{u_7, u_4, u_1\}$	2.09	0.998

From the results, we observe:

- 1) Lasso/LARS-based knob selection method can select the most appropriate (i.e. best possible) subset of knobs that affect the system output the most;
- 2) When more knobs are selected, the prediction of the system response becomes more accurate.

4.2 RLS-Based Model Identification

Once the subset of control inputs has been selected, we need an input-output model that represents the dynamics of the target system with reduced inputs. We assume such dynamics can be characterized by the following linear auto-regressive-moving-average (ARMA) model [14][15]:

$$A(q^{-1})y(k+1) = B(q^{-1})u(k) + e(k), \quad (5)$$

where $A(q^{-1})$ and $B(q^{-1})$ are matrix polynomials in the backward-shift operator:

$$\begin{aligned} A(q^{-1}) &= I - A_1q^{-1} - \dots - A_nq^{-n}, \\ B(q^{-1}) &= B_0q^{-1} + \dots + B_{n-1}q^{-n}, \end{aligned} \quad (6)$$

and n is the order of the system. $\{e(k)\}$ is a sequence of independent, identically distributed random vectors with zero means. It is further assumed that $e(k)$ is independent of $y(k-j)$ and $u(k-j)$ for $j > 0$. We use $e(k)$ to represent disturbances or noises in the system that are not accounted for by the model.

It is worth noting that for computing systems, the system order n is usually fixed and low [1] and can be estimated offline. However, the values of the coefficient matrices A_i and B_j , where $0 < i \leq n$ and $0 < j \leq n$, are likely to vary in a typical computing service due to changes in system operating conditions and workload dynamics. We deal with this issue by periodically re-estimates the model using the Recursive Least Squares (RLS) method [15] with exponential forgetting [16].

For notational convenience, we rewrite the system model in the following RLS-friendly form.

$$y(k+1) = X(k)\phi(k) + e(k+1), \quad (7)$$

$$\phi(k) = [u^T(k), \dots, u^T(k-n+1), y^T(k), \dots, y^T(k-n+1)]^T, \quad (8)$$

$$X = [B_0, \dots, B_{n-1}, A_1, \dots, A_n]. \quad (9)$$

For every control interval k , the value of $X(k)$ is re-estimated, which will be used in the controller design discussed below.

4.3 Linear Quadratic Controller Design

For the controller design, we aim at minimizing the following quadratic cost function:

$$J = \|W(y(k+1) - y_{ref}(k+1))\|^2 + (\|Q(u(k) - u(k-1))\|^2) \quad (10)$$

where W is a weighting matrix on the tracking errors and Q is a weighting matrix on the control values.

The goal of the controller is to steer the system into a state of optimum reference tracking with minimum variance, while penalizing large changes in the control values. For computing systems, a large change in the actuator setting is not desirable, since it may cause large oscillations in these systems. The W and Q weighting matrices are commonly chosen as diagonal matrices. Their relative magnitude provides a way to trade-off tracking accuracy for system stability.

The minimization should be over the set of all admissible controllers, where a controller is admissible if each control action $u(k)$ is a function of vector $\phi(k-1)$ and of the new measurement $y(k)$. Similar to the derivation shown in [13], we can derive the optimal controller by explicitly capturing the dependency of the cost function J on $u(k)$. We define

$$\tilde{\phi}(k) = [0, u^T(k-1), \dots, u^T(k-n+1), y^T(k), \dots, y^T(k-n+1)]^T, \quad (11)$$

The simplicity of the cost function allows us to have the following closed-form expression for the optimal control law, which can be easily implemented in our controller module:

$$\begin{aligned} u^*(k) &= ((W \hat{B}_0)^T W \hat{B}_0 + Q^T Q)^{-1} [(W \hat{B}_0)^T W (y_{ref}(k+1) \\ &\quad - \hat{X}(k) \tilde{\phi}(k)) + Q^T Q u(k-1)] \end{aligned} \quad (12)$$

5. SIMULATION AND EVALUATION

In this section, we present simulation results to demonstrate the effectiveness of the proposed control scheme. In particular, we show that our controller can adapt to workload and system dynamics as well as changes in performance goals.

5.1 Simulator and Experimental Methodology

We built a simulator using Matlab. The simulator is composed of four major modules. The *Plant Module* simulates a target system using a ‘‘underlying model’’, which will be unknown in real systems. This module represents the system’s output using the full set of inputs. In the evaluation below, we use the model described in Equation (4) for simulation. The *Dimension Reduction Module* implements the LASSO/LARS algorithm. Its functionality is to find a reduced set of inputs of the system such that they can affect the system’s output the most. The *Model Identification Module* implements the recursive least square algorithm. Its functionality is that given the selected control inputs, identify the input-output dynamics for the reduced dimension system. The fourth module is the *Controller Module*. Its functionality is that given the control reference, system output feedback, the module calculates the control values over the reduced inputs set to achieve the desired output reference. In our simulator, we implemented the Linear Quadratic Controller described in Section 4.3.

5.2 Evaluation Results

We performed extensive evaluation of our approach on the simulator we built. In the following, we report the results from two set of experiments. The first set of experiments evaluate the performance of our reduced dimension control design when system workload changes. The second set of experiments evaluate the performance of our design when the system internal dynamics (model) change.

In all these experiments, the controller’s Q and W matrices are selected as identity matrix. The total simulation time is 500 steps.

For the control using the reduced input set based on the Lasso/LARS algorithm, the system in the simulation is open-loop during the first 20 sampling intervals. This is a warm-up period to allow the Model Reduction Module to collect enough samples for input-reduction. After collecting the first 20 samples of input-output dataset, Lasso/LARS algorithm is used to select the optimal inputs.

In the first set of experiments, the reference target in is 1 in the first 250 steps, and the target changes to 2 in the remaining 250 steps. Fig. 5-8 show the experiment results when the system model is governed by Equation (4). In each of the figures, the top graph shows the control-computed values for the selected inputs, and the bottom graph demonstrates the closed-loop performance by showing the simulated output value.

Fig. 4 shows the result when all the 7 inputs are used in the controller. **Fig. 5** shows the result when the 3 most significant control inputs (u_7, u_4, u_1) are selected by the Lasso/LARS algorithm to control the system output. **Fig. 6** shows the result when the 2 most significant control inputs (u_7, u_4) are selected to control the output. **Fig. 7** shows the result when only the single most significant control input (u_7) is selected to control the output.

From these figures, we observe:

1. When only a subset of control inputs is allowed to be selected, our dimension reduction module can correctly select the right control inputs to control the system output;
2. As more control inputs are allowed to be selected, the performance of the closed-loop control system is improved, where both the steady-state error and the variance of the controlled output become smaller;
3. The closed-loop performance using the selected 3 most significant control inputs is comparable to the performance using the full set of 7 control inputs. This demonstrates that using a (small) subset of control inputs is suffice to achieve the performance goal in this example.

In the second set of experiment, we test the adaptive property of our approach. In the first 250 steps, the system model is governed by Equation (4); whereas in the remaining 250 steps, the system model is governed by the following equation:

$$\begin{aligned}
 y(k+1) = & 0.1u_1(k) + 3.1u_2(k) + 14.1u_3(k) \\
 & + 0.1u_4(k) + 2.1u_5(k) + 0.1u_6(k) \\
 & + 0.1u_7(k) + e(k)
 \end{aligned} \quad (13)$$

In all 500 simulation steps, the output reference is set to 1. **Fig. 8** shows the control input values and the simulated output under the varying model. After 20 steps of open-loop execution, the Lasso/LARS algorithm selects u_7, u_4, u_1 , which are the top 3 most significant control inputs under the system model in Equation (4). At the 250th step sample, the system model changes, and the new system is governed by Equation (13). As we can see from **Fig. 8**, the closed-loop performance deteriorates where the output deviates significantly from the reference value. This is because the formerly selected control inputs are no longer the optimal subset. Since our controller is constantly monitoring the system's performance, when performance degradation is detected, our controller starts to collect a new set of 20 input-output pairs and feeds the data into the dimension reduction module (Lasso) to re-select the optimal subset of control inputs. After time step 270,

the new optimal control inputs u_2, u_3 , and u_6 are selected. The performance of the closed-loop system becomes better and finally converges to the output target.

As we can see in this experiment, our scheme can dynamically detect changes in the system model, and adjust the selection of the correct subset of control inputs accordingly. Hence our controller can continue to ensure good tracking performance.

6. CONCLUSION AND FUTURE WORK

In this paper, we present a novel approach for simplified control of complex IT systems. Evaluation results have shown that our three-module controller design achieves the following design objectives: 1) It can judiciously select a subset of control knobs for the target system in a dynamic environment. These selected knobs have the most significant impact on the outputs of the system being controlled; 2) It can dynamically tune the selected control knobs effectively to maintain the system output at the desired value under a dynamic environment; 3) It can automatically detect the change in the most critical knobs and use the new knobs to regulate the system outputs accordingly.

In our future work, we would like to apply our simplified control approach to dynamic resource allocation and application configuration problems in IT systems management with possible multiple inputs and outputs, and validate the approach on a real system testbed we are building.

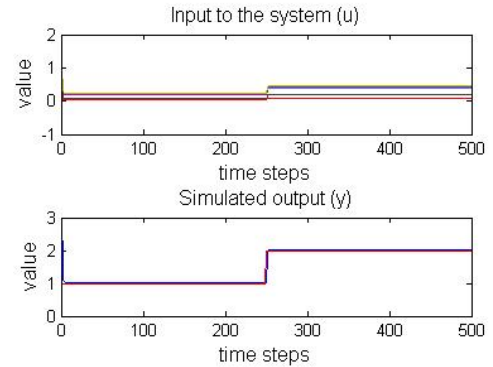


Fig. 4. Performance of Experiment 1.a: Control inputs and simulated output using 7 (full) inputs

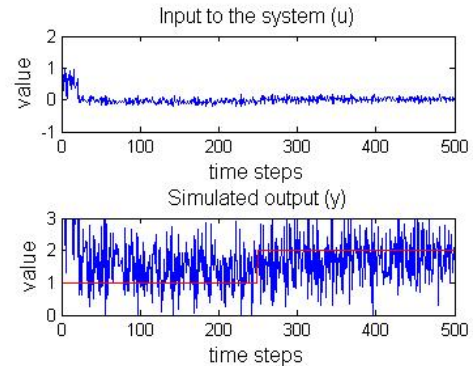


Fig. 5. Performance of Experiment 1.b: Control inputs and simulated output using 3 inputs (u_7, u_4, u_1)

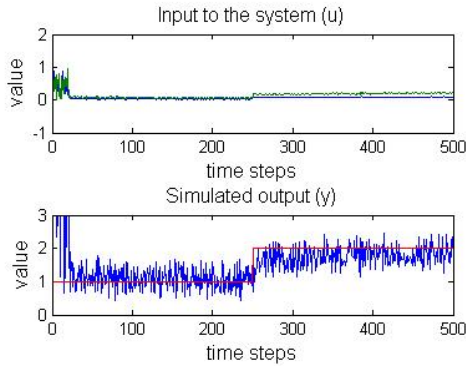


Fig. 6. Performance of Experiment 1.c: Control inputs and simulated output using 2 control inputs (u_7, u_4)

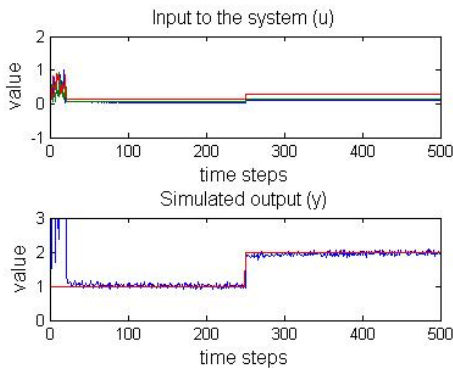


Fig. 7. Performance of Experiment 1.c: Control inputs and simulated output using 1 control input (u_7)

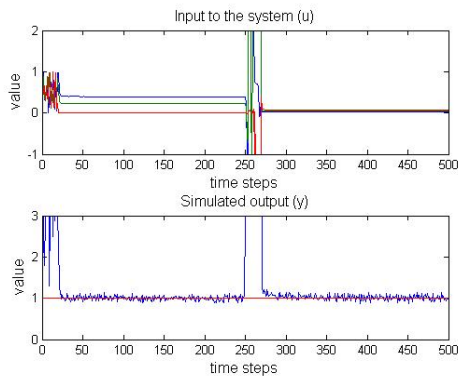


Fig. 8. Performance of Experiment 2: Control inputs and simulated output when the system behavior changes

7. REFERENCES

- [1] J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury, *Feedback Control of Computing Systems*: Wiley-IEEE Press, 2004.
- [2] T. Abdelzher, J.A. Stankovic, C. Lu, R. Zhang, and Y. Lu, "Feedback performance control is software services," *IEEE*

Control System Magazine, vol. 23, no. 3, pp.74-90, June 2003.

- [3] Ira Cohen, J. S. Chase, M. Goldszmidt, T. Kelly, and J. Symons, "Correlating Instrumentation Data to System States: A Building Block for Automated Diagnosis and Control," in *Proceedings of the 6th Symposium of Operating Systems Design and Implementation*, December 2004.
- [4] Y. Diao, F. Eskesen, S. Froehlich, J.L. Hellerstein, A. Keller, L. Spainhower, and M. Surendra, "Generic on-line discovery of quantitative models for service level management," in *IFIP Symposium on Integrated Management (IM'03)*, 2003.
- [5] K. Zhou, J.C. Doyle, and K. Glover, *Robust and Optimal Control*, Prentice Hall, August 1995.
- [6] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *19th ACM Symposium on Operating Systems Principles (SOSP)*, 2003, pp. 164-177.
- [7] M. Rosenblum, "VMware's Virtual Platform: A virtual machine monitor for commodity PCs," in *Hot Chips 11 1999*, Stanford University, Stanford, CA, 1999.
- [8] Wikipedia, "OpenVZ --- Wikipedia, The Free Encyclopedia," 2007.
- [9] Wikipedia, "Linux-VServer --- Wikipedia, The Free Encyclopedia," 2007.
- [10] P. Padala, X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, K. Salem, and K. Shin, "Adaptive control of virtualized resources in utility computing environments," in *EuroSys'07*, March 2007.
- [11] B. Efron, I. J. Trevor Hastie, and R. Tibshirani, "Least angle regression," *Annal of Statistics*, vol. 32, pp. 407-499, 2004.
- [12] R. Tibshirani, "Regression Shrinkage and Selection via the Lasso," *Journal of Royal Statistics Sociey*, vol. B, pp. 267-288, 1996.
- [13] X. Liu, X. Zhu, P. Padala, Z. Wang, S. Singhal, "Optimal Multivariate Control for Differentiated Services on a Shared Hosting Platform," in *Proceedings of the 46th IEEE Conference on Decision and Control (CDC'07)*, New Orleans, LA, 2007.
- [14] X. Liu, X. Zhu, S. Singhal, and M. Arlitt, "Adaptive Entitlement Control to resource containers on shared servers," in *9th IEEE/IFIP International Symposium on Integrated Network Management (IM'05)*, 2005.
- [15] K. J. Astrom and B. Wittenmark, *Adaptive Control*, 2nd ed.: Prentice Hall, 1994.
- [16] R. Kulhav'y, "Restricted exponential forgetting in real-time identification," in *Automatica*, vol. 23(5), September 1987, pp. 589-600.