A First-Principles Based LPV Modeling and Design for Performance Management of Internet Web Servers

Wubi Qin, Qian Wang, Yiyu Chen, and Natarajan Gautam

Abstract— **This paper presents a control-theoretic approach to performance management of Internet Web servers to meet Service Level Agreements (SLA). In particular, a CPU frequency management problem is studied to provide response time guarantees with minimal energy cost. A linear uncertain model and a Linear-Parameter-Varying (LPV) system are derived based on first-principles analysis of transient and steady-state queueing dynamics from the allocated CPU resource to request response time. The LPV modeling utilizes the workload arrival and service parameters as scheduling variables, which allows the Web server to meet the response time SLA in the presence of dynamically changing load conditions. Using real Web server workloads, the performance of an LPV- H_∞ controller is compared to that of a linear controller designed at the high-percentile load parameters and a G/G/1-queueing based nonlinear optimization. The proposed LPV modeling and control framework can be generalized to incorporate more sophisticated workload models and more complicated server environments. In addition, due to the LPV nature of Web systems with respect to load conditions, the proposed approach can be applied to a variety of resource management problems and used for middleware design.**

I. Introduction

High-performance computer systems are widely used in today's commercial (Web, database, e-commerce) and scientific server environments. The high demand on automatically managing server resources, under dynamically changing load and operating conditions, strongly motivates feedback-based control schemes. Compared to using trial-and-error methods for tuning system parameters, control-theoretic approaches provide a rigorous mathematical foundation and systematic mechanisms in the design of feedback-control systems. There has been increasing research effort in applying control theoretic approaches to the performance management for computer systems, though mostly in the area of network systems and congestion control. The research on application of control theory to software systems such as Web, email, database, and storage servers has just started [3], [5-9], [11], [13] (see the survey paper [2], [10] and references therein). There exists substantial difference between performance management of a communication network system and of an Internet server system, e.g., the difference in modeling request service demand. Requests in a network system are fixed-size packets; the service time of a request at a link (leaving a router) corresponds to the transmission time of the packets. Thus service time for each request is more or less constant (depending on the link bandwidth). In contrast, service demand in a Web/storage server system could be highly varied or even depend on locality and access pattern.

Most of the existing work for performance management of Internet server systems adopts system-identification based linear-time-invariant (LTI) modeling and classical PID control ([2-3], [8], to name a few). Due to the inherent nonlinear dependence of request response time on resource allocation variables as well as the demand on adaptation to time-varying load conditions, nonlinear modeling and design are motivated to improve system performance. An adaptive controller was designed for the performance control of differentiated caching services [11], where an online identified linear model was built using a recursive least-squares algorithm. A fuzzy-logic control is used to optimize performance of an Apache Web server in [7]. For the admission control of an Internet Web server, our previous paper [12] presented an LPV design based on direct LPV system identification using empirical data, where workload intensity was utilized as the scheduling variable. The present paper studies the performance/power management of computer server systems under the context of Internet hosting centers; the objective is to dynamically manage server CPU in meeting response time SLA with minimal energy cost. Starting with a first-principles analysis of transient and steady-state queueing dynamics, this paper approximates the dynamics from CPU frequency to response time by a Linear-Parameter-Varying system with request arrival and service demand parameters as scheduling variables. LPV control has the advantage that it does not require a priori information on scheduling variables but their online measurements, thus no prediction is needed for workload arrival and service demand. Compared to the adaptive control in [11], the LPV modeling provides analytical functional expression of system dynamics on load conditions, which allows the corresponding design to explicitly utilize load information and existing work on workload characterization.

Major contributions of this paper are: 1) A Linear-Parameter-Varying model is derived based on first-principles analysis of system dynamics and then an LPV- H_∞ robust control is designed; 2) using real Web server workloads, we show that the LPV- H_∞ control design outperforms linear control designs and a G/G/1-queueing based nonlinear optimization. The general framework of this LPV modeling and control can be applied to many resource

This work is supported in part by NSF under Grants 0325056 and 0409184. W. Qin (wubi@psu.edu) and Q. Wang (quw6@psu.edu) are with the Mechanical Engineering Department; Y. Chen (yzc107@psu.edu) is with the Industrial Engineering Department, Penn State University, University Park, PA 16802. N. Gautam (gautam@tamu.edu) is with the Industrial Engineering , Texas A&M University, College Station, TX 77843.

allocation problems for managing hosting centers due to the LPV-nature of system dynamics with respect to load.

II. Problem Description and Server System Model

A. Performance/Power Management of a Hosting Center

This paper investigates the performance management of computer server systems under the context of Internet hosting centers. A hosting center operates thousands of servers to house multiple applications. A guaranteed level of performance, which is referred to as Quality of Service (QoS) delivered to end customers, is often part of a Service-Level Agreement between the service provider and application owners. On one hand, a hosting center needs to make revenue by provisioning sufficient resources (CPU, memory, and I/O bandwidth) for each application to meet performance SLA in the presence of dynamically varying load conditions. On the other hand, a hosting center needs to reduce operational costs to maximize profit.

Particularly, we are interested in the performance/power management problem – to meet performance SLA with minimal power consumption. A hosting center can use different mechanisms to manage power cost: e.g., allocating appropriate number of servers to each application and turning off unused servers; dynamical control of server CPU frequency using the dynamic voltage/frequency scaling (DVS) mechanism, which is allowed by most processors today. The latter scheme may not be able to save as much energy as turning off a server completely, but it has the advantage of allowing requests to be served while at a lower CPU speed (with no or less performance degradation); further, it takes less time for the server to recover to its normal CPU frequency than rebooting a machine.

Traditional performance management for server systems relies on worst-case estimates of load and resource availability thus provisions resources to meet peak demands. Since the worst-case resource demand is likely to be significantly higher than its normal usage, static optimization based on peak loads could cause server resources to be heavily underutilized most of time, while over-provisioning service capacity for worst-case load could incur economically unfavorable operational costs. Therefore, it is desirable to design control algorithms for the server to adapt to workload behavior.

Consequently, the objective of the performance/power management for a hosting center is to dynamically allocate an appropriate number of servers to each application and to control each server's CPU frequency such that the target response time of each application is met with minimal energy consumption / operational cost.

B. Server System Model and Problem Formulation

Consider a hosting center that operates identical servers to support multiple applications at any time. Each server is equally capable of running any application and it is devoted to a particular application while each application may span on multiple servers. By the DVS scheme, each server can operate at one (may change over time) of a set of discrete CPU frequency levels between a maximum frequency and a minimum frequency.

We decompose the above-described performance/power management problem into two subproblems: 1) dynamically determine an aggregate frequency for each application that can meet response time guarantee when there is a single server per application running at this frequency, and 2) solve a server allocation problem, which determines the number of servers for each application and the operating CPU frequency of each individual server to provide the aggregate CPU frequency obtained from the first subproblem.

Since this paper is aimed to develop a nonlinear control-theoretic modeling and design for server performance management, we focus on the first subproblem, i.e., dynamical control of the aggregate CPU frequency to meet target response time with minimal CPU usage (noting that the CPU power consumption of a server is proportional to the cubic power of the CPU frequency). In the rest of the paper, the aggregate CPU frequency is often referred to as CPU frequency. In our previous work [5], a linear ARX model was built using system identification to characterize the dynamics from server's CPU frequency to request response time; then a linear optimal control combined with an on-line optimization algorithm was applied to this performance/power management problem. In this paper, we develop a first-principles based LPV modeling and control design.

III. Derivation of First-Principles Models

Consider a Web server that serves http requests in a single queue with first-come first-serve manner, as depicted by Fig. 1. Let Δt be the sampling interval, and let $N_s(k)$ denote the number of jobs in the system at time $k\Delta t$. Since the number of jobs in the system at $(k+1)\Delta t$ is the sum of the initial number of jobs in system at $k\Delta t$ and the number of arrival requests (n^k_{arriv}) minus the number of requests serviced (n^k_{srved}) in $[k\Delta t, (k+1)\Delta t]$, we have

$$N_s(k+1) = \{N_s(k) + n^k_{arriv} - n^k_{srved}\}^+ \quad (1)$$

where $\{\cdot\}^+ = \max(\cdot, 0)$ since N_s is nonnegative.

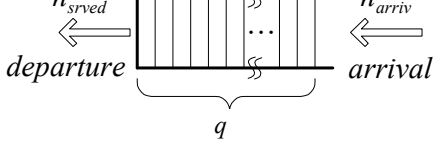


Fig.1. A queueing system structure

The mean arrival rate, $\lambda(k)$, and mean service rate, $\mu(k)$, in $[k\Delta t, (k+1)\Delta t]$ are computed as

$$\lambda(k) = n^k_{arriv} / \Delta t, \quad \mu(k) = n^k_{srved} / \Delta t \quad (2)$$

We approximate (1) by removing the projection to the positive plane $\{\cdot\}^+$ in the modeling for control design (while we keep the projection in simulation), which leads to

$$N_s(k+1) = N_s(k) + (\lambda(k) - \mu(k))\Delta t \quad (3)$$

The average response time $T(k)$ can be approximated by

$$T(k) = \frac{N_s(k)}{\mu(k)} \quad (4)$$

Note that the mean service rate is defined as the reciprocal of the mean service time, which conventionally denotes the mean time it takes a server operated at certain CPU frequency to process a request (here for CPU demand). In this paper, the DVS scheme allows the CPU frequency of a server to be a tunable parameter; further, the request service time on a server is assumed to be inversely proportional to this server's CPU frequency. Let $u(k)$ denote the CPU frequency (GHz), and let $s(k)$ denote the mean service time on a server operating at the unit CPU frequency. Then the service rate $\mu(k)$ satisfies

$$\mu(k) = \frac{u(k)}{s(k)} \tag{5}$$

Plug (5) into (3) and (4), we have

$$N_s(k+1) = N_s(k) - \frac{\Delta t}{s(k)} u(k) + \lambda(k)\Delta t \tag{6}$$

$$T(k) = \frac{N_s(k)s(k)}{u(k)} \tag{7}$$

which is a nonlinear time-varying system with output variable $T(k)$, state variable $N_s(k)$, and control variable $u(k)$.

A. A Linearized Model

We first derive a linearization of (6) and (7) around an equilibrium point. By observing (6) and (7), the equilibrium condition satisfies

$$\overline{\lambda} = \overline{\mu} = \frac{\overline{u}}{\overline{s}}, \quad \overline{T} = \frac{\overline{N}_s \overline{s}}{\overline{u}} \tag{8}$$

Define

$$N_s(k) = \overline{N}_s + \hat{N}_s(k),\ T(k) = \overline{T} + \hat{T}(k),\ u(k) = \overline{u} + \hat{u}(k) \tag{9}$$

Then the linearized model around $(\overline{N}_s, \overline{T}, \overline{u})$ and $(\overline{\lambda}, \overline{s})$ can be written as,

$$\hat{N}_s(k+1) = \hat{N}_s(k) - \frac{\Delta t}{\overline{s}} \hat{u}(k) \tag{10}$$

$$\hat{T}(k) = \frac{1}{\overline{\lambda}} \hat{N}_s(k) - \frac{\overline{T}}{\overline{s}\,\overline{\lambda}} \hat{u}(k) \tag{11}$$

This can be treated as a linear time-invariant model if constant $(\overline{\lambda}, \overline{s})$ and $\overline{T}$ are considered (which can be computed as the average over the entire workload duration).

B. A Linear-Parameter-Varying Model Derived Using Jacobian Linearization

Based on Jacobian linearization, an LPV model can be derived by linearizing (6-7) around the equilibrium trajectory $\overline{u}(k) = \overline{\lambda}(k)\overline{s}(k)$, $\overline{N}_s(k) = \overline{T}\overline{u}(k) / \overline{s}(k) = \overline{T}\overline{\lambda}(k)$, where the "-" and index *k* are used to denote "steady-state" values in the *kth* sampling interval, while the target response time $\overline{T}$ is the same across all intervals. Following a similar procedure as in Section III.A, we derive the LPV model as follows:

$$\hat{N}_s(k+1) = \hat{N}_s(k) - \frac{\Delta t}{\overline{s}(k)} \hat{u}(k) \tag{12}$$

$$\hat{T}(k) = \frac{1}{\overline{\lambda}(k)} \hat{N}_s(k) - \frac{\overline{T}}{\overline{s}(k)\overline{\lambda}(k)} \hat{u}(k) \tag{13}$$

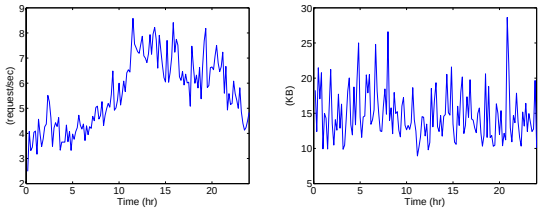
The scheduling parameters $\overline{\lambda}(k)$ and $\overline{s}(k)$ used in Section IV-VI are approximated by the mean arrival rate $\lambda(k)$ and service time $s(k)$ in the *kth* sampling interval. We can also update the scheduling parameters using average arrival rate and service time computed over a longer time period, e.g., multiple sampling periods, since the load is often considered to vary in a much slower time scale than system dynamics.

IV. Model Validation

A. Workload Description and Simulation Setup

The workload used in this paper is a real http trace from the Web Caching Project group [1]. Fig. 2 plots the request arrival rate and file size (KB) for this one-day trace. Modeling and control designs have been evaluated using a simulator built on top of the CSIM simulation package. In implementing the simulator, it has been assumed that the static http requests hit in the cache. Then microbenchmarks were run for requests (which hit in the cache) with different file size on a server machine to obtain request service times at the highest operating frequency. The relationship between file sizes and service times is verified to be more or less linear – the service time is proportional to the file size. Similar experiments have also been conducted on a laptop with DVS capabilities to confirm that the service time is inversely proportional to the CPU operating frequency, in terms of which the service times obtained from Laptop DVS frequencies are then scaled for the server-class CPU in the simulation model.

The target response time $\overline{T}$ is set to be *20 sec*. We choose a 2-minute sampling period for implementing the models and feedback control design. However, for the purpose of better visibility in a limited space, some of the figures in the rest of the paper are plotted using a 10-minute sampling periods.



(a) (b)

Fig.2. Workload (a) arrival rates; (b) request file size (KB).

We first study how analytical models derived in Section III capture system dynamics. In validating the derived control-oriented models, different trajectories of the CPU frequency $u(k)$ can be specified to process the workload, then the model-predicted response time $T(k)$ will be compared with that obtained from the simulation corresponding to the same $u(k)$. Fig. 3 plots the simulated versus the LPV model (12-13) predicted response time $T(k)$ resulting from the $u(k)$ which is set to be proportional to the product of arrival rate $\lambda(k)$ and service demand $s(k)$. From Fig. 3, we can see that the LPV model is able to capture the real system dynamics, especially when the response time is adequately large. It is noted that the prediction loses accuracy when response time is relatively small after *Hour 20*; this deterioration in prediction corresponds to situations where the server is experiencing extreme light load which degrades

the approximation of (1) by (3).

V. Control Design

Based on the first-principles models, (8-11) or (12-13), we dynamically control the CPU frequency $u(k)$ so that the response time $T(k)$ will meet the target value $\overline{T}$ in the presence of time-varying load conditions with minimal CPU usage. We present a linear controller that is designed at a high percentile value of load conditions, and an LPV- H_∞ robust controller with scheduling parameters specified in terms of the mean arrival rate $\lambda(k)$ and service demand $s(k)$.

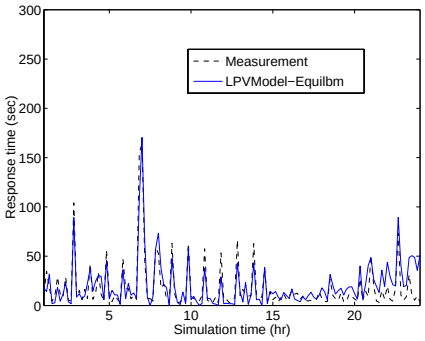


Fig.3. Validation of the LPV model.

A. A Linear Controller Designed at High-Percentile Load Conditions

Consider the linearized model (10-11), which represents a linear time-invariant model with coefficients defined in terms of a nominal (equilibrium) condition. In order to meet response time SLA for varying load conditions, a linear uncertain model is studied by identifying bounds for the uncertain parameters $\overline{\lambda}$ and $\overline{s}$. Essentially we consider a linear robust control problem. Rather than design a linear robust controller using worst-case values of arrival rate λ and service demand s for $\overline{\lambda}$ and $\overline{s}$, we design the linear controller using the high-percentile values of λ and s. The latter is motivated by the cumulative distribution of request service demand; it is noted that the worst-case value of file size is more than ten times the 95-percentile value, which is more than ten times the mean value. Thus designing at a high-percentile value of the load condition may save significantly in resource provisioning.

We formulate a Linear Quadratic (LQ) control problem optimizing the cost function as follows:

$$J = \sum_{k=1}^{\infty} (r_T \cdot \hat{T}^2(k) + r_u \cdot \hat{u}^2(k)) \tag{14}$$

where r_T and r_u are weights for meeting the response time SLA and minimizing control (CPU) energy. By (10-11), the optimization is subject to the following dynamic equation,

$$\hat{N}_s(k+1) = \hat{N}_s(k) - \frac{\Delta t}{s_{\alpha\%}} \hat{u}(k) \tag{15}$$

$$\hat{T}(k) = \frac{1}{\lambda_{\alpha\%}} \hat{N}_s(k) - \frac{\overline{T}}{s_{\alpha\%}\lambda_{\alpha\%}} \hat{u}(k) \tag{16}$$

where the subscript "$\alpha\%$" represents the α-percentile value. In order to reduce steady-error in meeting target response time, we augment the system (15-16) by adding an integrator. Define a new state variable, $\hat{N}_s^I(k) = \sum_{i=1}^{k-1} \hat{N}_s^I(i)$, then we have

$$\hat{N}_s^I(k+1) = \hat{N}_s^I(k) + \hat{N}_s(k) \tag{17}$$

The state feedback control law for CPU frequency $u(k)$ is then computed as,

$$\begin{aligned} u(k) &= \overline{u} + k_1 \hat{N}_s^I(k) + k_2 \hat{N}_s(k) \\ &= \lambda_{\alpha\%} s_{\alpha\%} + k_1 \hat{N}_s^I(k) + k_2 \hat{N}_s(k) \end{aligned} \tag{18}$$

where the feedback gain k_1 and k_2 are calculated by solving the LQ problem defined by (14-17).

B. LPV- H_∞ Control Design

By utilizing the time-varying load parameters as scheduling parameters, the Linear-Parameter-Varying control design is expected to improve performance with efficient control usage. The LPV control can be classified as a generalized gain-scheduling control. It designs a parameter-dependent controller $K(\delta)$ to stabilize an augmented Linear-Parameter-Varying plant $P_{aug}(\delta)$ for all admissible parameter trajectories δ, minimizing the effect of the exogenous inputs on the controlled variables in certain norm. The augmented plant $P_{aug}(\delta)$ includes the actual LPV system $P(\delta)$ to be controlled as well as auxiliary weighting functions representing closed-loop performance criteria. For an affine parameter-dependent plant $P(\delta)$, the design of an affine parameter-dependent controller $K(\delta)$ is often reduced to solving a set of parameter-dependent Linear Matrix Inequalities (LMIs) [4].

Define scheduling parameters $\delta_1(k) := 1/\lambda(k)$, $\delta_2(k) := 1/s(k)$, and $\delta_3(k) := 1/(\lambda(k)s(k))$, the model (12-13) then becomes an affine parameter-dependent plant. Note that the scheduling parameters defined in this way are not independent, which may cause the resulting design more conservative, but it allows the direct application of existing affine parameter-dependent LPV control designs without resorting to the polynomial parameter-dependent LPV.

We formulate an LPV- H_∞ control design problem, where the performance specifications on minimizing tracking error of meeting target response time and reducing control action are addressed through the design of weighting functions W_e and W_u, respectively. In order to apply the LPV- H_∞ control design from [4], low-pass filters are appended to both input and output channels of the original LPV plant. The cutoff bandwidth of the low-pass filters should be much higher than the feedback sampling frequency so that the system performance would not be affected. With a bit abuse of notion, we let $P(\delta)$ denote the LPV model that includes the original plant (12-13) as well as the input/output low-pass filters. The controller $K(\delta)$ is designed such that the closed-loop system is stabilized and the H_∞ norm of the transfer function from the exogenous input (the reference

response time $\overline{T}$) to the controlled variables (the weighted error signal $\tilde{e}$ and the weighted control signal $\tilde{u}$) is minimized, i.e., $\left\| \begin{matrix} W_e S \\ W_u KS \end{matrix} \right\|_\infty \le \gamma$ with performance level γ , where S denotes the sensitivity transfer function. We then apply the LPV- H_∞ control design from [4].

VI. Simulation results & Performance Analysis

A. Control Design Results

Both the LQ and LPV design have been implemented on the simulator. Their performance is evaluated using the real http trace plotted in Fig. 2, and is compared to that of a G/G/1-queueing based nonlinear optimization given in [5].

LQ Design: The weights in (14) are chosen to favor meeting target response time, which is specified as $\overline{T} = 20$ *sec* for the trace in Fig. 2. We have designed feedback control gains for model (15-16) with $\alpha\% =$ 50-, 85-, and 95-percentile load conditions. The design at $\alpha\% =$ 85-percentile gives the best tradeoff between meeting target response time and using minimal CPU frequency; the corresponding control gains are $k_1 = -0.016$, $k_2 = -0.032$.

LPV Design: In the LPV- H_∞ formulation, the weighting function W_e and W_u are chosen to reduce tracking error and peak control action. They are designed as,

$$W_e = \frac{0.239z^2 - 0.2446z + 0.133}{z^2 - 1.769z + 0.787},\ W_u = \frac{3.109z^2 - 5.505z + 2.458}{z^2 - 0.693z + 0.473}$$

Then the LPV controller is derived to satisfy $\gamma < 1$.

A G/G/1-Queueing Based Nonlinear Optimization: The result from a G/G/1-queueing based nonlinear optimization [5] is also presented here in comparison with results from control-theoretic approaches. The algorithm, denoted as *Queueing* in the rest of the paper, consists of three components: a multiplicative S-ARMA (seasonal autoregressive moving average) for workload characteristics estimation, an G/G/1 model for predicting response time, and a heuristic solution to a nonlinear optimization problem. The sampling period for modeling and design in this G/G/1-based nonlinear optimization is chosen to be 1hr, which produces the best result. Details of the *Queueing* algorithm can be found in the co-authored work [5].

B. Simulation Results & Analysis

In this section, simulation results from the approaches in the previous section are presented and their performance is compared. Table 1 lists the average response time and average aggregate CPU frequency for LQ, LPV- H_∞ and the G/G/1-based nonlinear optimization (denoted by *Queueing*). The average values are calculated in terms of the whole 24 hr duration for the trace.

The results for LQ regulator in Table 1 are obtained from the design conducted at 85-percentile of λ and s . From Table 1, we can see that the LQ design uses as much as control action but has higher average response time than the Queueing. The LPV control satisfies the target response time as the Queueing does, but using 20% less CPU, which would reduce the operational cost significantly. The LPV design has lower values in both average response time and control effort than that of LQ designed at high-percentile load conditions. This demonstrates that the utilization of detailed time-varying information on workload improves the performance of control design efficiently.

Table 1. Simulation results of three designs.

	LQ	LPV	Queueing
Mean response time (sec)	21.48	19.89	17.22
Mean Aggregate CPU (GHz)	17.50	14.03	17.18

Fig. 4 compares time histories of response time for LPV control to the LQ designed at 85-percentile of the load condition. We can see that the LPV design outperforms linear designs significantly. In particular, during the transient overload situation (~ *hour 8* & *hour 21*), the LPV controller is able to allocate the right amount CPU thus to maintain a reasonable response time. In contrast, the LQ controller fails to do so, which causes considerable oscillations in CPU allocation, substantial spikes in response time, and system getting stuck in overload for significant longer time.

Fig. 5 shows the response time history of LPV control versus that of G/G/1-based nonlinear optimization. It is observed that the LPV control is more responsive with respect to workload changes. In particular, the LPV control has reduced the spikes around *hour 8, 12*, and *hour 21*, and the LPV's time response is less oscillated than that of the Queueing approach. In terms of CPU allocation (Fig. 5(b)), we can see that the *Queueing* is over-allocating CPU during most of the time compared to LPV. At *hour 8, 12*, and *21,* but it fails to adapt to the sudden increase of workload and response time is much higher than that of LPV. LPV control on the other hand is more responsive and has allocated sufficient resource to maintain a reasonable response time.

Though the LPV control has lower mean response time than the LQ and the *Queueing* approaches, it still shows spikes periodically in response time, as plotted in Figs. 4 & 5. These spikes are mostly due to the dramatic variation of request-level service demand (file size). For the studied trace, a sudden increase of file size from less than 1KB to over 1MB is frequently observed. Note that the average CPU speed in LPV control is 14KB/sec, which implies that it takes 71 seconds for the server to process a single request of 1MB. All requests after this request inevitably have longer waiting time. Indeed the spikes of response time correspond to bursts in service demand and arrival rate (see Fig. 2(b)). Though the dramatic changes of request file size often occur, these jumps are not persistent, but very transient instead. In addition, change of sampling interval is barely helpful for the modeling to capture or predict such burstiness due to the self-similarity behavior in real traces.

C. Easiness for Real-Time Control

As far as implementation is concerned, since the presented LPV design is based on an analytical model (12-13), it only requires on-line measurement of load parameters; while the *Queueing* requires significant amount of data for training in order to achieve reasonably good prediction. The LQ designs presented in Sections V-VI are not on-line implementable since the entire workload is not available a priori to compute

the required average (percentile) values for arrival rate and service demand. For real-time implementation, predicted workload parameters would be used instead, which could lead to additional modeling uncertainty thus degrading control performance. In this paper, we mainly use the LQ designs as baseline to compare with the LPV approach, thus we neglect this implementation issues for the LQ.

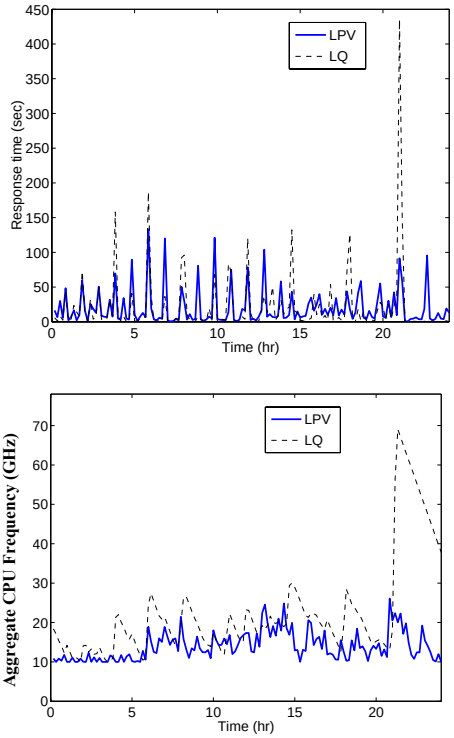


Fig. 4. LPV vs. LQ: response time and CPU frequency.

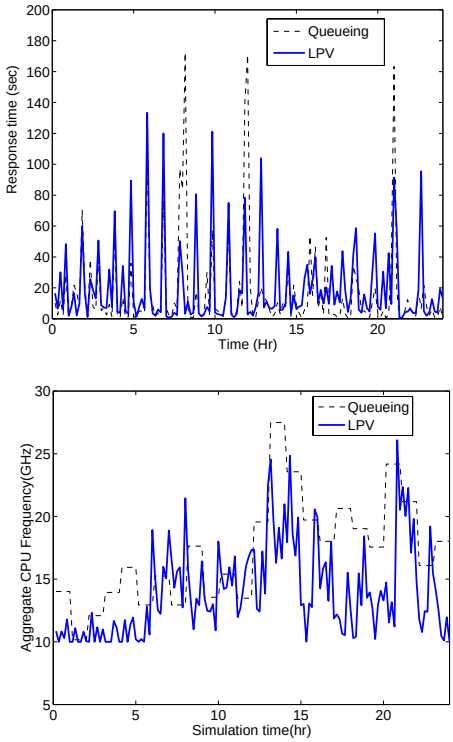


Fig. 5. LPV vs. Queueing: response time and CPU frequency.

VII. Conclusion

This paper has presented a first-principles based LPV modeling and control for performance management of Web servers in achieving response time SLA. By analyzing transient queueing dynamics for a Web server, we have derived analytical models that characterize dynamics from the CPU frequency to response time in the presence of time varying load conditions. We have derived a linear uncertain model and an LPV model which uses time-varying request arrival rate and service demand as scheduling parameters. Through simulations using real http traces, the LPV design is demonstrated to outperform both the LQ design and the G/G/1-queueing based optimization. The advantage of this first-principles based LPV modeling and control is that it only requires on-line measurement of workload statistics, thus avoiding extensive model training using historical traffic data. This framework provides the versatility in dealing with different types of workload and operating environment without modifying the implementation of control algorithms.

Acknowledgment

The authors would like to thank Dr. A. Sivasubramiam and Mr. A. Das for helpful discussions and part of system setup.

Reference

[1] Web Caching project, http://www.ircache.net.

[2] T. F. Abdelzaher, J. A. Stankovic, C. Lu, R. Zhang, and Y. Lu, "Feedback performance control in software services," *IEEE Control Systems Magazine*, Vol. 23, No. 3, 2003, pp. 74-90.

[3] T. F. Abdelzaher, Y. Lu, R. Zhang, D. Henriksson, "Practical application of control theory to web service," *Proceedings of American Control Conference,* 2004, pp. 1992-1997.

[4] P. Apkarian and R. J. Adams, "Advanced gain-scheduling techniques for uncertain systems," *IEEE Transactions on Control System Technology*, Vol. 6, 1998, pp. 21-32.

[5] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam, "Managing server energy and operational costs in hosting centers," *SIGMETRICS* , Banff, Canada, 2005, pp. 303-314.

[6] Y. Diao, N. Gandhi, J. L. Hellerstein, S. Parekh, and D. M. Tilbury, "Using MIMO feedback control to enforce policies for interrelated metrics with applications to the Apache web servers," *Proceedings of Network Operations and Management*, 2002, pp. 219-234.

[7] Y. Diao, J. L. Hellerstein, and S. Parakh, "Optimizing quality of service using fuzzy control, *Proceedings of the 13th IFIP/IEEE International Workshop on Distributed Systems Operations and Management*, 2002, pp. 42-53.

[8] Y. Diao, C. W. Wu, J. L. Hellerstein, A. J. Storm, M. Surendra, S. Lightstone, S. Parekh, C. Garcia-Arellano, M. Carroll, L. Chu, and J. Colaco, "Comparative studies of load balancing with control and optimization techniques," *Proceedings of American Control Conference,* Portland, OR, 2005, pp. 1484-1490.

[9] J. Hellerstein, Y. Diao, and S. Parekh, "A first-principles approach to constructing transfer functions for admission control in computing systems," *Proceedings of IEEE Conference on Decision and Control*, LasVegas, 2002.

[10] J. Hellerstein, "Challenges in control engineering of computer systems," *Proceedings of American Control Conference,* 2004, pp. 1970-1979.

[11] Y. Lu, T. F. Abdelzaher, C. Lu, and G. Tao, "An adaptive control framework for QoS guarantees and its application to differentiated caching services," *Proceedings of Tenth International Workshop on Quality of Service*, 2002.

[12] W. Qin and Q. Wang, "Feedback performance control for computer systems: an LPV approach," *Proceedings of the American Control Conference*, Portland, OR, 2005.

[13] A. Robertsson, B. Wittenmark, M. Kihl, and M. Andersson, "Design and evaluation of load control in web server systems," *Proceedings of American Control Conference,* 2004, pp. 1980-1985.