

# An LPV Approach to Performance Control of Web Servers Under Self-Similar Workloads

Wubi Qin, Qian Wang\*

Dept. of Mechanical Engineering  
Penn State University

Email: {wubi, quw6}@psu.edu

\* Corresponding author

Anand Sivasubramaniam

Dept. of Computer Science & Engineering  
Penn State University

Email: anand@cse.psu.edu

**Abstract**—Applying control-theoretic approaches to capacity provisioning of web servers is gaining popularity. This paper presents a novel approach that combines Linear Parameter Varying (LPV) modeling and control design, with workload characterization using stochastic  $\alpha$ -stable-model envelopes. We parameterize a control-oriented Web server model and the resulting controller using workload distribution parameters. By further incorporating the  $\alpha$ -stable modeling into the LPV control approach, the presented solutions not only allow system to adapt to workload changes, but also show great promise in handling *self-similar* workloads. The proposed method is applied to a CPU allocation problem for web servers, which performs Dynamic Voltage Scaling (DVS) to achieve response time guarantee. Simulations using real web-server traces are conducted to show the strength of the proposed approach.

## I. INTRODUCTION

Server capacity provisioning for Internet services is an extremely important and challenging problem. These servers may need to handle periods of high loads, while obeying responsiveness requirements. Several prior studies have shown that sustained periods of high loads are rare. With large variances in load over time, it may not be economically desirable to over-provision resources for occasional high load periods. Meanwhile the dynamically changing workload and the complex relationship between provisioned capacity and associated response times makes this capacity provisioning problem challenging.

This important problem has received considerable attention over the past decade. One body of work in this area (e.g. [30]) has focussed on understanding and characterizing the load imposed on these servers by studying real world traces from different environments. For example, an extensive bibliography of self-similar traffic and performance modeling can be found in [30], we refer to these techniques as *predictive-model-based* provisioning techniques. Recent work on workload characterization ([14], [19]) has proposed  $\alpha$ -stable models for self-similar workloads that widely exist in Internet services.  $\alpha$ -stable distributions are heavy tailed and have infinite variance, thus can model both the burstiness and long range dependence well. In addition  $\alpha$ -stable models are determined by a small set of parameters [14], allowing identification and reconstruction with manageable computation cost. Another body of work in this area (e.g. [4], [12]) uses instead feedback from the current system state and performance measurements for decision making. The intuitive rationale for this feedback strategy is that

even if the workload modeling is not accurate enough, periodic feedback from the system would give sufficient information to compensate for the inaccuracies. We refer to these techniques as *feedback-based* provisioning techniques. In this paper, we are specifically interested in control-theoretic based feedback techniques, since they provide a mathematical foundation for system design in order to meet performance guarantees.

Most existing work ([7], [9], [13], [10], [20], [25], [3], [2], [11], [21], [24], [26], [28]) on performance management of Internet servers have adopted linear system modeling and classical (PID) control design techniques, see the review papers ([4], [12]). There is a lack of rigorous robustness analysis for these linear designs with respect to coping with large load variations. In order to enhance adaptation to load variations, adaptive controllers have been designed for control of differentiated services [22] and storage systems [17], [16], [15], [18], where online linear models are identified using the least-squares methods. Still, it is not clear how well such models can handle large variations of load conditions that has not been observed in the workloads. A fuzzy-logic control has been used to optimize the performance of the Apache web server in [8]. Since fuzzy-logic control is an artificial-intelligent based approach, it is often difficult to rigorously prove system stability and quality of solutions. A model-predictive-control (MPC) was applied to the utilization control of distributed real-time systems in [29], where a linear time invariant (LTI) model was used as well. Though MPC is good at handling constraints on state and control variables and also provides certain robustness, a good modeling is still needed in order to deal with variations in load conditions.

With the goal of designing a controller that can adapt to the workload and operating conditions, this paper presents a Linear Parameter Varying (LPV) approach for the system modeling and control design. First we model the relationship between capacity provisioning and response time as an LPV system, with workload parameters as scheduling variables. We model the incoming traffic using alpha-stable processes, and periodically determine their parameters. These parameters are then used as scheduling variables to parameterize the LPV model and to parameterize the resulting controller, which is an LPV controller. The small parameter set of alpha-stable models makes the computation in identifying the LPV model and LPV control design manageable.

Compared to the linear-recursive models used in the adaptive control, whose coefficients though are updated from time to time but no explicit functional expression with respect to load parameters is ever explored, the proposed LPV approach is based on explicit analytical parameterization of the LPV model and controller using load parameters. Simulations using traces from the real world of a proxy server show that this LPV controller is able to (i) meet response time bounds without overprovisioning, and (ii) performs better than individual predictive (based on just workload modeling) and simple feedback-based models.

The rest of this paper is organized as follows. Section II formulates the problem, and the system model is derived in Section III. Section IV presents the controller design. Simulation results are given in Section V, and conclusions are drawn in the end.

## II. PROBLEM FORMULATION

In this paper, we consider managing CPU frequency to provide response time guarantees for requests in the context of Internet servers. As a first step, we consider a single queue served by a single server, whose CPU frequency can be tuned dynamically, e.g., using the dynamic voltage/frequency scaling (DVS) mechanism [25] which is allowed by most processors today. Noting that the CPU frequency is closely related to power consumption and electricity cost associated with each server, the DVS scheme allows energy saving while still serving requests at a lower CPU speed thus leading to no or less performance degradation. The design goal is to dynamically determine a minimal CPU frequency that can meet target response time. Then, by viewing the CPU here as an aggregate capacity which should be provided by multiple servers, this CPU control result can be combined with online optimization algorithms from our previous work [6] to provide solutions to multiple-server multiple-application CPU allocation problems. Nevertheless, this paper focuses on the first step - deriving advanced control-oriented model and controller design.

In the rest of the paper, we first derive a control-oriented model to characterize the dynamic relationship between the operating CPU frequency and the resulting mean response time. Then we design a controller that meets the target response time with minimal CPU frequency.

## III. SYSTEM MODEL

In this section, we will combine  $\alpha$ -stable self-similar workload characterization with LPV techniques to build a control-oriented LPV model. Essentially, we model the arrival and service demand of a self-similar workload using  $\alpha$ -stable models. The  $\alpha$ -stable models are periodically identified online to determine workload parameters, which are used as scheduling variables to parameterize the LPV model and controller. We will not introduce the basics of  $\alpha$ -stable distribution due to the space limit, but interested readers can find more details in [27].

### A. Self-similar workloads

Consider applying the  $\alpha$ -stable self-similar model to a self-similar arrival process, then the number of requests

arrived in the  $k$ th sampling period (with duration  $\Delta t$ ) can be computed as follows,

$$A(k) = (c \cdot N_{\alpha,H}(k) + \lambda) \cdot \Delta t \quad (1)$$

where the parameter  $\lambda$  denotes the mean arrival rate,  $N_{\alpha,H}$  is a linear fractional stable noise (LFSN) corresponding to an  $\alpha$ -stable measure with zero mean, scale parameter  $\sigma = 1$  and some fixed skewness  $\chi$  satisfying  $-1 \leq \chi \leq 1$  ( $\chi = 1$  is often used to represent a totally positively skewed distribution). The LFSN represents the increment during the  $k$ th unit length time interval of an linear fractional stable motion, which is often used to model the cumulative arrivals. The scaling factor  $c$  is defined as the ratio of the scale parameter of the traffic process to the scale parameter of the LFSN. Noting that the scale parameter of the LFSN  $N_{\alpha,H}$  is 1, the scaling factor  $c$  is actually the scale/dispersion around the mean of the traffic.

Next, we introduce the stochastic envelope for the alpha-stable distribution. Noting that requests' resource demand often exhibits heavy-tailed distribution, if the capacity allocation is targeted to the worst-case scenario, the resource may be over provisioned and the system could be underutilized most of the time. The notion of stochastic envelope provides flexibility for capacity allocation to satisfy different levels of service demand, so a good balance can be achieved between resource efficiency and reducing risk in not providing performance guarantees. A stochastic envelope for the arrivals in Eq.1 is computed as,

$$\hat{A} = (\lambda + \beta_A \cdot \sigma_A) \cdot \Delta t \quad (2)$$

The parameter  $\beta_A$  is uniquely determined by the probability that the number of arrivals  $A$  surpasses its approximation  $\hat{A}$ , where the probability is set to a pre-specified small risk  $\epsilon$ , i.e.,

$$P\{A > \hat{A}\} = P\left\{\frac{A - \lambda \cdot \Delta t}{\sigma_A \cdot \Delta t} > \beta_A\right\} = \epsilon \quad (3)$$

In Eq.2, the parameters  $\lambda$  and  $\sigma_A$  denote the mean and scale parameters of the arrival process. Comparing Eq.1 and Eq.2, and denoting the scale parameter of the LFSN  $N_{\alpha,H}$  by  $\sigma_N$ , then  $\sigma_A = c \cdot \sigma_N$ , where  $c$  is the scaling factor in Eq.1. Therefore, the arrival parameters  $\lambda$  and  $\sigma_A$  can be obtained from the  $\alpha$ -stable modeling of the request arrival process. Note that there is no closed-form mathematical formula to give the direct relation from the risk  $\epsilon$  to the parameter  $\beta_A$  in Eq.3, in terms of the  $\alpha$ -stable distribution. However, given a set of  $\epsilon$ , the corresponding  $\beta_{AS}$  can be numerically generated and tabulated using the  $\alpha$ -stable distribution model [19]. If we also model the service demand (or file size) of requests using the  $\alpha$ -stable self-similar model, similar results as Eq.1 and Eq.2 can be obtained as well, and the details are given in the next section.

### B. An LPV fluid model for Web servers

A general discrete-time LPV system, denoted by  $\Gamma(p)$ , can be represented by a difference equation as follows:

$$x(k+1) = F(p(k))x(k) + G(p(k))u(k) \quad (4)$$

where  $x(\cdot)$  and  $u(\cdot)$  are state variables and control variable respectively.  $p(\cdot)$  is an exogenous time-varying parameter, often referred to as scheduling variable of the LPV system. An LPV control designs a scheduling-variable-parameterized controller to stabilize the LPV system in Eq.4. The LPV control is often classified as a generalized gain-scheduling control, but it provides provable stability guarantee for the closed-loop system, which can not be provided by traditional gain-scheduling control. Another advantage of the LPV control is that it does not require a priori knowledge of the scheduling parameters but only their online measurements. In the sequel we derive a control-oriented LPV model for Web servers from CPU allocation to request response time. The scheduling parameters of the LPV model are the mean and scale parameters of the  $\alpha$ -stable self-similar models used to characterize the workload arrival and service demand.

Consider sampling intervals with sampling time  $\Delta t$ , and let  $N(t)$  denote the number of jobs in the system at the beginning of the  $k$ th sampling interval, then  $N(t+1)$  can be calculated as,

$$N(k+1) = \{N(k) + A(k) - D(k)\}^+ \quad (5)$$

where  $A(k)$  and  $D(k)$  denote the number of arrivals and the number of departures (fished requests) in the  $k$ th period, respectively. The notation  $\{\cdot\}^+ = \max(\cdot, 0)$ . The physical meaning of Eq.5 is: the number of jobs in the system at the end of  $\Delta t$  is the initial number of jobs plus the increment ( $A(k)-D(k)$ ).

Considering a self-similar arrival process, by Eq.2, the number of arrivals is approximated by its  $\epsilon$ -stochastic envelope,

$$A(k) \approx (\lambda(k) + \beta_A(k) \cdot \sigma_A(k)) \cdot \Delta t \quad (6)$$

with  $\beta_A$  defined in Eq.3. We further consider that the request file size (service demand), denoted by  $s(k)$ , is a random variable approximated by the  $\alpha$ -stable self-similar model,

$$s(k) \approx \nu(k) + \beta_s(k) \cdot \sigma_s(k) \quad (7)$$

where  $\nu(k)$  and  $\sigma_s(k)$  are the mean and scale parameters of  $s(k)$  respectively. The parameter  $\beta_s$  is determined similarly as in Eq.3,

$$\begin{aligned} P\{s(k) > \nu(k) + \beta_s(k) \cdot \sigma_s(k)\} \\ = P\left\{\frac{s(k) - \nu(k) \cdot \Delta t}{\sigma_A(k) \cdot t} > \beta_A(k)\right\} = \epsilon \end{aligned} \quad (8)$$

Note that different risk parameters  $\epsilon$  can be chosen for approximation of the arrival  $A(k)$  and file size  $s(k)$ .

Let  $u(k)$  denote the allocated capacity and assume that the service time is inversely proportional to the allocated capacity, then the number of requests being served is approximated as,

$$D(k) = \frac{\Delta t}{s(k)/u(k)} \approx \frac{u(k) \cdot \Delta t}{\nu(k) + \beta_s(k) \cdot \sigma_s(k)} \quad (9)$$

Consequently, by plugging Eq.6 and Eq.9 into Eq.5 and linearizing the fluid model (ignoring the nonlinearity  $\{\cdot\}^+$  in

the control design), we have the following control-oriented model,

$$N(k+1) = N(k) + \left\{ (\lambda(k) + \beta_A(k) \cdot \sigma_A(k)) - \frac{u(k)}{\nu(k) + \beta_s(k) \cdot \sigma_s(k)} \right\} \cdot \Delta t \quad (10)$$

By the Little's law, the average response time  $T(k)$  in the  $k$ th sampling time interval can be approximated by

$$T(k) = \frac{N(k)}{\lambda(k)} \quad (11)$$

Note that for a given pre-specified risk level  $\epsilon$  for stochastic envelope, Eqs. 10-11 are parameterized by workload characterizing parameters  $\lambda(k)$ ,  $\sigma_A(k)$ ,  $\nu(k)$ , and  $\sigma_s(k)$ , where the parameters  $\beta_A$  and  $\beta_s$  are determined once  $\epsilon$  is fixed. Define  $p = \{\lambda, \sigma_A, \nu, \sigma_s\}$ , and compare Eqs. 10-11 with Eq. 4, we see that the system model consists of Eqs. 10-11 is an LPV system with parameters of arrival and service demand defined as scheduling variables.

#### IV. CONTROL DESIGN

Corresponding to the LPV model in Eqs. 10-11, this section designs an LPV controller that generates time-varying  $u(k)$  such that the response time  $T(k)$  will meet its target value  $\bar{T}$  with minimal allocated capacity. Since the LPV controller itself is also parameterized by the scheduling variables (the load parameters), it is expected that the LPV controller can provide better adaptation to the workload changes than linear controllers with constant control gains.

An LPV model  $\Gamma_{aug}(p)$  can be viewed as an LTI model  $\Gamma_{aug}$  scheduled, in a specific way, by the parameter  $p$  (the subscript *aug* will be explained later). The right figure in Fig.1 shows that the scheduling parameter  $p$  enters the model  $\Gamma_{aug}(p)$  in a so-called linear-fractional-transformation (LFT) way [5]. For an LPV model  $\Gamma_{aug}(p)$ , an LPV control designs a scheduling-variable parameterized controller  $K(p)$  such that the closed-loop system is stabilized for all admissible parameter trajectories  $p(k)$  [5]. Next we explain how the augmented model  $\Gamma_{aug}$  relates to the Web server model  $\Gamma$  and what the input/output variables  $w, u, z, y$  in Fig.1 mean in English. The augmented model  $\Gamma_{aug}$  is a combination of  $\Gamma$  and a set of filters, which are often referred to as weighting functions. There are two input vectors and two output vectors of the augmented model  $\Gamma_{aug}$ :

- the input  $w$  denotes any external input signals to the system, e.g., the reference signal, disturbance and sensor noise.
- the input  $u$  denotes the control input to the system model, which is also the output of the controller  $K$ .
- the output  $y$  is the output variable of the system model, which is measured and fed back into the controller  $K$ .
- the controlled variable  $z$ , which consists of the set of variables of interest, e.g., the tracking error and control effort.

Given a set of external inputs  $w$  to the system, performance specifications of the controller are fulfilled by specifying appropriate controlled variables  $z$  and designing the weighting functions. For example, in Fig.2, given a target

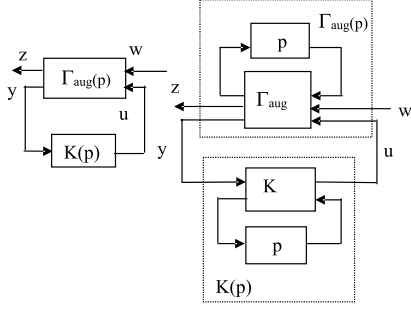


Fig. 1. Block diagram for a general LPV control structure

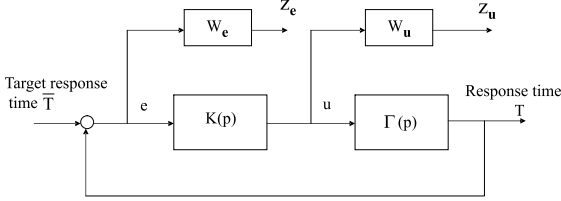


Fig. 2. LPV robust control block diagram

response time  $\bar{T}$  (which is an external input included in  $w$ ), reducing the tracking error can be achieved by 1) including the weighted tracking error signal  $Z_e$  in the controlled variable  $z$ , 2) designing a filter  $W_e$  that specifies the frequency range where reducing tracking error should be emphasized, and 3) minimizing (or restricting) the norm of the transfer function from the input  $w$  to the output  $z$ . In Fig.2, we also include the weighted control input signal  $Z_u = W_u \cdot u$  in the controlled output  $z$ , then limiting the control effort can be achieved by minimizing or reducing the norm of the transfer function from  $\bar{T}$  to  $Z_u$ . By comparing Fig.2 and the left figure of Fig.1, the augmented  $\Gamma_{aug}$  is formed by combining the original model  $\Gamma$  and weighting functions  $W_e$  and  $W_u$ , with appropriate block-diagram transformation.

For the LPV model in Eqs. 10-11, we formulate an LPV- $H_\infty$  control design problem as shown in Fig.2. Performance specifications on minimizing tracking error of meeting target response time and reducing control action are addressed by minimizing the  $H_\infty$  norm of the transfer function  $T_{zw}$  from the input signal  $w$ , which includes the target response time  $\bar{T}$ , to the controlled output variable  $z$ , which includes the frequency-weighted tracking error and the frequency-weighted control action. The algorithms for the LPV- $H_\infty$  can be found in [5], and off-the-shelf MATLAB LPV Robust Control toolbox is used in our controller design. The design parameters for applying the MATLAB LPV- $H_\infty$  control are the weighting functions  $W_e$  for the tracking error and  $W_u$  for the control input.

## V. EXPERIMENTAL EVALUATION

Our simulations use real http traces from the Web Caching project [1]. There are three traces in all and each corresponds to an individual web application for one-day duration. We denote the traces as Workload 1 to 3 (or WL 1-3) respec-

tively. The arrival rates and file sizes of the three workloads are plotted in Fig. 3 and Fig. 4 respectively. The modeling and control designs have been evaluated using a simulator built on top of the CSIM simulation package. It is assumed that the static http requests in WL 1-3 hit in the cache, and the service time of a request is proportional to the file size and inversely proportional to the operating CPU frequency [6].

### A. Alpha-Stable Modeling of Workload

In this paper, we implemented the quantile based algorithm [23] for identifying parameters of the  $\alpha$ -stable self-similar models. The whole trace of each workload is divided into segments, and the  $k$ th segment (with duration  $T$ ) is denoted by  $\Delta T(k)$ . Then we identify an  $\alpha$ -stable self-similar model (for the arrivals in each period segment  $\Delta T(k)$ ). Fig.3 shows the measured (including both raw data and mean value) vs.  $\alpha$ -stable-model predicted arrival rates. Similar characterization is done for service demand, as shown in Fig.4. In these figures, *Mean Value* is computed using the measurements, while *A-S* corresponds to  $\alpha$ -stable model predicted workload statistics for 85% envelope ( $\epsilon = 0.15$ ). Both the *Mean Value* and *A-S* are calculated using  $\Delta T = 1$ hr. It can be seen that the time varying  $\alpha$ -stable models capture the load variations very well.

### B. Control Design Results

To evaluate the LPV- $\alpha$ -stable control design which are described as follows, we compare them with those of three particular approaches. *Alloc-by-request (Req)*: a no-brainer scheme that allocates CPU demanded by the total requests, which is the product of the arrival rate and file size in a sampling period. *OpenLoop- $\alpha$ -stable (OpLp)*: a pure  $\alpha$ -stable model based CPU allocation scheme that does not use any feedback thus can be considered as an open-loop design [19]. *LQ*: a linear quadratic (LQ) controller which minimizes the weighted quadratic sum of response-time tracking error and CPU frequency. An identified ARX model is used for the LQ design. Our approach is denoted as *LPV*: the LPV controller which is scheduled by workload parameters of  $\alpha$ -stable self-similar models for arrivals and file sizes. Essentially, we compare among a design that allocates by latest demand with no modeling, a design that uses detail workload modeling but no feedback, a design that uses feedback but no sophisticated workload modeling, and the proposed LPV control which combines workload modeling and feedback control that adapts to workload changes.

The main metrics used for comparison here are average response time and average CPU frequency. The response time SLA is set to be 20s; with 10% slack, the design is considered to meet the response time SLA if the mean response time is less than 22s. The mean CPU frequency will be compared among feasible designs that meet the response time SLA. Besides the mean statistics, Table I also lists the variances of response time and CPU frequency to give an indication how these designs react to transient overloads.

Note that the  $\alpha$ -stable self-similar modeling and all design schemes depend on the time granularity of the sampling

periods (denoted by  $\Delta t$ ) during which the control is implemented and the response time and workload statistics are measured, and possibly the time intervals where the  $\alpha$ -stable models are updated (denoted by  $\Delta T$ ). In Table I, we include the results for different time granularities:  $\Delta t = 10s, 30s, 60s, 300s$ , and for each  $\Delta t$ , the corresponding  $\Delta T = 10min, 30min, 1hr, 4hr$ . All results shown in the Table are based on one-step prediction, which means these experiments are implemented online.

From the results one can see that the LPV controller consistently meets the response time SLA (except for WL2 at  $\Delta t = 300s$ ) and has lower mean CPU frequency than the other feasible designs. The LPV controller also has much lower variances of response time and CPU in most cases. In addition, the LPV controller is very robust to different time granularities.

*LPV vs. LQ*: In terms of the mean response time, the best LQ controllers can meet target response time for both WL1 and WL3, and have slightly higher response time than the target value for WL2. Though the mean CPUs of the LQ controllers are only slightly higher than the LPV controllers, the best LQ design has at least 50% higher (and up to ten-times) variance of response time than the best LPV controller for each workload. This indicates that the LPV controllers adapt to the load variations and handle transient overloads much better than the linear controllers.

*LPV vs. OpLp*: The best design of OpLp meets the response time SLA for WL1, but its response time is far from the target value for both WL2 and WL3, while it uses comparable mean CPU frequency as the LPV controller. It is also observed that OpLp has much higher variance of response time and is very sensitive to the sampling time. There is no consistent trend that smaller time granularities would provide better results or vice versa based on the experiments. The OpLp designs allocate CPU corresponding to certain stochastic envelope with no feedback, thus they ignore and do not react to the spikes in workloads, while the LPV designs smooth out the effect of workloads spikes on the response time.

*LPV vs. Req*: The best Alloc-by-request design is close to meeting target response time for WL3, but none of the Alloc-by-request designs can satisfy the response time SLA for WL1 and WL2. We can see that the Alloc-by-request designs are not able to allocate enough CPU by just utilizing mean statistics of workloads from the previous sampling interval.

In summary, by comparing the performance of designs w/ or w/o feedback, and w/ or w/o workload modeling, the  $\alpha$ -stable workload-parameter scheduled LPV modeling and control designs combine all the advantages and show the best results.

## VI. CONCLUSIONS

In this paper, we present an LPV modeling and control design framework for the performance management of Internet servers subject to time-varying self-similar workloads. We extract workload parameters using the  $\alpha$ -stable self-similar models, and use these parameters as scheduling variables to parameterize the control-oriented system model and the

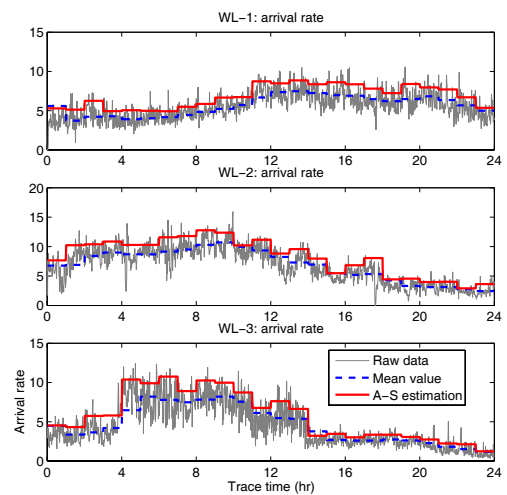


Fig. 3. Time-varying  $\alpha$ -stable model predicted arrivals

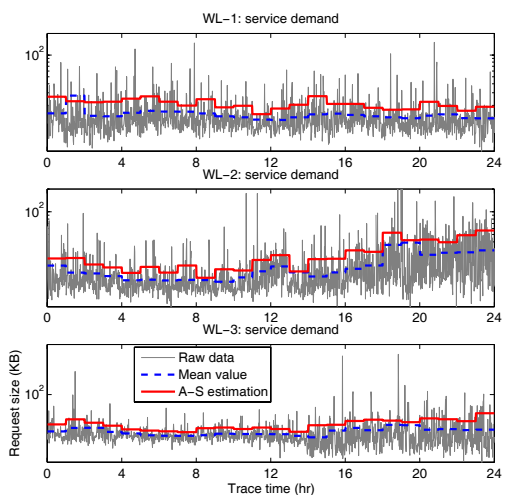


Fig. 4. Time-varying  $\alpha$ -stable model predicted service demand

resulting controller. Through simulations using real web traces, our proposed  $\alpha$ -stable model based LPV control design outperforms a naive demand based design, a *predictive* design which is based on workload modeling, and a *feedback* design that uses a system-identification model without detail knowledge of load conditions.

## REFERENCES

- [1] Web Caching project. <http://www.ircache.net>.
- [2] T. Abdelzaher, Y. Lu, R. Zhang, and D. Henriksson. Practical application of control theory to web service. In *Proceedings of American Control Conference*, pages 1992–1997, 2004.
- [3] T. Abdelzaher, K. G. Shin, and N. Bhatti. Performance guarantees for Web server end-systems: A control-theoretical approach. *IEEE Transactions on Parallel and Distributed Systems*, 13(1), 2002.
- [4] T. Abdelzaher, J. A. Stankovic, C. Lu, R. Zhang, and Y. Lu. Feedback performance control in software services. *IEEE Control Systems Magazine*, 23(3):74–90, 2003.
- [5] P. Apkarian and R. J. Adams. Advanced gain-scheduling techniques for uncertain systems. *IEEE Transactions on Control Systems Technology*, 6(1):21–32, 1998.
- [6] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam. Managing server energy and operational costs in hosting centers.

WL1																
$(\Delta T, \Delta t)$	mean resp. time (sec)				resp. time var.				mean CPU (MHz)				CPU var.			
(min, sec)	LPV	LQ	OpLp	Req	LPV	LQ	OpLp	Req	LPV	LQ	OpLp	Req	LPV	LQ	OpLp	Req
(10,10)	20.43	20.00	21.44	110.1	1610	1589	1587	3.6e4	14.14	16.10	15.68	12.93	88	94	13	87
(30,30)	20.08	19.79	34.84	74.62	1050	1532	3553	3.4e4	13.01	14.96	14.73	12.78	58	48	12	70
(60,60)	20.09	19.72	39.34	76.14	1011	1707	4818	2.5e4	13.22	16.01	14.58	12.55	42	38	11	44
(240,300)	20.25	22.62	61.04	78.61	1117	3730	9127	7907	14.23	20.62	14.43	12.07	17	89	12	13

WL2																
$(\Delta T, \Delta t)$	mean resp. time (sec)				resp. time var.				mean CPU (MHz)				CPU var.			
(min, sec)	LPV	LQ	OpLp	Req	LPV	LQ	OpLp	Req	LPV	LQ	OpLp	Req	LPV	LQ	OpLp	Req
(10,10)	21.39	22.80	418.3	1962	1734	4254	7.7e5	3.4e6	18.50	30.56	21.77	15.51	213	264	64	162
(30,30)	21.13	25.01	372.7	592.3	1563	7654	5.7e5	5.6e5	18.25	19.82	24.35	15.68	150	65	81	117
(60,60)	21.17	25.54	211.9	205.1	1990	9023	2.0e5	6.9e4	19.69	21.23	26.90	15.57	140	59	100	85
(240,300)	26.03	411.9	31.29	292.5	6884	4.5e5	2.1e4	4.1e4	25.04	23.61	29.47	15.43	235	298	45	54

WL3																
$(\Delta T, \Delta t)$	mean resp. time (sec)				resp. time var.				mean CPU (MHz)				CPU var.			
(min, sec)	LPV	LQ	OpLp	Req	LPV	LQ	OpLp	Req	LPV	LQ	OpLp	Req	LPV	LQ	OpLp	Req
(10,10)	18.40	18.17	210.8	24.21	4267	4303	1.1e5	7810	14.27	16.59	14.54	13.71	92	121	62	72
(30,30)	18.08	35.39	104.0	31.73	3482	1.0e4	5.1e4	1.1e4	15.70	16.03	15.61	13.27	101	119	68	47
(60,60)	18.13	104.3	120.2	40.33	4151	2.9e4	6.4e4	1.2e4	16.23	15.24	15.19	13.05	88	223	63	36
(240,300)	20.33	26.02	883.7	85.78	4462	1.2e4	2.1e6	1.9e4	16.02	19.99	14.71	12.81	27	74	33	23

TABLE I  
DESIGN RESULTS

- In *Proceedings of the International Conference on Measurement and Modeling of Computer Systems Sigmetrics05*, pages 303–314, 2005.
- [7] 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. In *Proceedings of Network Operations and Management*, pages 219–234, 2002.
- [8] Y. Diao, J. L. Hellerstein, and S. Parekh. Optimizing quality of service using fuzzy control. In *Proceedings of the 13th IFIP/IEEE International Workshop on Distributed Systems Operations and Management*, pages 42–53, 2002.
- [9] Y. Diao, J. L. Hellerstein, S. Parekh, R. Griffith, G. Kaiser, and D. Phung. Self-managing systems: A control theory foundation. In *Proceedings of the 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems*, 2002.
- [10] 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. In *Proceedings of American Control Conference*, pages 1484–1490, 2005.
- [11] N. Gandhi, S. Parekh, J. Hellerstein, and D. Tilbury. Feedback Control of a Lotus Notes Server: Modeling and Control Design. In *Proceedings of the American Control Conference*, 2001.
- [12] J. Hellerstein. Challenges in control engineering of computer systems. In *Proceedings of American Control Conference*, pages 1970–1979, 2004.
- [13] J. Hellerstein, Y. Diao, and S. Parekh. A first-principle approach to constructing transfer functions for admission control in computing systems. In *Proceedings of the IEEE Conference on Decision and Control*, pages 2906–2912, 2002.
- [14] A. Karasaridis and D. Hatzinakos. Network heavy traffic modeling using  $\alpha$ -stable self-similar processes. *IEEE Transactions on Communications*, 49:1203–1214, 2001.
- [15] M. Karlsson. Concurrency control in computer services using adaptive optimal control. In *Proceedings of the 25th IASTED International Conference on Modeling, Identification, and Control*, pages 155–160, 2006.
- [16] M. Karlsson, C. Karamanolis, and J. Chase. Controllable fair queuing for meeting performance goals. In *Proceedings of the IFIP International Symposium on Computer Performance Modeling, Measurement and Evaluation*, pages 278–294, 2005.
- [17] M. Karlsson, C. Karamanolis, and X. Zhu. Triage: Performance isolation and differentiation for storage systems. In *Proceedings of the 12th International Workshop on Quality of Service*, 2004.
- [18] M. Karlsson, X. Zhu, and C. Karamanolis. An adaptive optimal controller for non-intrusive performance differentiation in computing services. In *Proceedings of the IEEE Conference on Control and Automation*, 2005.
- [19] M. Lopez-Guerrero, L. Orozco-Barbosa, and D. Makrakis. Probabilistic envelope processes for  $\alpha$ -stable self-similar traffic models and their application to resource provisioning. *Performance Evaluation*, 61:257–279, 2005.
- [20] C. Lu, T. F. Abdelzaher, J. A. Stankovic, and S. H. Son. A Feedback Control Approach for Guaranteeing Relative Delays in Web Servers. In *Proceedings of the Seventh Real-Time Technology and Applications Symposium*, page 51, 2001.
- [21] C. Lu, J. A. Stankovic, G. Tao, and S. Son. Feedback control real-time scheduling: framework, modeling, and algorithms. *Journal of Real-Time Systems, Special Issues on Control-Theoretical Approaches to Real-Time Computing*, 23:85–126, 2002.
- [22] Y. Lu, T. F. Abdelzaher, C. Lu, and G. Tao. An adaptive control framework for QoS guarantees and its application to differentiated caching services. In *Proceedings of Tenth International Workshop on Quality of Service (IWQoS)*, pages 23–32, 2002.
- [23] J. H. McCulloch. Simple consistent estimators of stable distribution parameters. *Communications on Statistics-Simulation*, 15:1109–1136, 1986.
- [24] S. Parekh, N. Gandhi, J. Hellerstein, D. 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*, pages 841–854, 2001.
- [25] P. Pillai and K. Shin. Real-time dynamic voltage scaling for low-power embedded operating systems. In *Proceedings of the 18th ACM Symposium on Operating System Principles*, pages 89–102, 2001.
- [26] A. Robertsson, B. Wittenmark, M. Kihl, and M. Andersson. Design and evaluation of load control in web server systems. In *Proceedings of American Control Conference*, pages 1980–1985, 2004.
- [27] G. Samorodnitsky and M. S. Taqqu. *Stable Non-Gaussian Random Processes*. Chapman & Hall, 1994.
- [28] L. Sha, X. Liu, Y. Lu, and T. Abdelzaher. Queuing model based network server performance control. In *Proceedings of the 3rd IEEE Real-Time Symposium (RTSS'02)*, 2002.
- [29] X. Wang, C. Lu, and X. Koutsoukos. Feedback utilization control in distributed real-time systems with end-to-end tasks. *IEEE Transactions on Parallel and Distributed Systems*, 16(6):550–561, 2005.
- [30] W. Willinger, M. Taqqu, and A. Erramilli. *Stochastic Networks: Theory and Applications*, chapter A Bibliographical Guide to Self-Similar Traffic and Performance Modeling for Modern High-Speed Networks. Oxford University Press, 1996.