Self-Optimization in Computer Systems via Online Control: Application to Power Management

Nagarajan Kandasamy*, Sherif Abdelwahed*, John P. Hayes**

**Institute for Software Integrated Systems*
Vanderbilt University
Nashville, Tennessee, U.S.A.
{nkandasa, sherif}@isis.vanderbilt.edu

***Advanced Computer Architecture Lab.*
EECS Department, University of Michigan
Ann Arbor, Michigan, U.S.A.
jhayes@eecs.umich.edu

Abstract

Computer systems hosting critical e-commerce applications must typically satisfy stringent quality-of-service (QoS) requirements under dynamic operating conditions and workloads. Also, as such systems increase in size and complexity, maintaining the desired QoS by manually tuning the numerous performance-related parameters will become very difficult. This paper addresses the design of self-optimizing computer systems using a generic online control framework in which the control actions governing the operation of the system are obtained by optimizing its behavior, as forecast by a mathematical model, over a limited time horizon. As a specific application of this control technique, we show how to minimize the power consumed by a single computer processing a time-varying workload. Assuming a processor capable of operating at multiple frequencies, we design an online controller to satisfy the QoS requirements of the workload while operating the processor at the lowest possible frequency. We describe the processor model, formulate the power management problem, and derive the online control algorithm. The performance of the controller is evaluated using representative e-commerce workloads. Finally, we discuss how the proposed technique can be applied to other resource management problems in computer systems.

1 Introduction

Computer systems hosting applications crucial to commerce and banking, transportation, military command and control, among others, have several distinguishing characteristics, which collectively pose some new and important research challenges: (1) Networks comprising hundreds of computers are now commonplace due to cheaper (and faster) processors and similar advances in data storage and communication bandwidth. However, increasing numbers of skilled personnel are required to design, operate, and maintain such complex systems; (2) since these systems can consume up to several megawatts of electricity, power management is a major design issue [20]; and (3) the computers must respond continuously to external events in real time and satisfy stringent QoS requirements; they must accommodate a dynamic workload while ensuring predictable response times to users.

In the forementioned computer systems, multiple performance-related parameters must be continuously tuned to achieve the desired QoS under dynamic operating conditions and workloads. The current state-of-the-art involves substantial human effort, and as the system complexity increases, achieving good performance via manual tuning will become very difficult [13]. Therefore, this paper develops an approach to designing self-optimizing computer systems using online control. The proposed method provides a systematic way to manage resources in a general setting; if the computer system is correctly modeled and its operating environment accurately estimated, the control actions required to maintain a certain QoS by optimizing a given cost function can be derived [3].

The problem of interest is to efficiently manage computer resources to meet specified performance objectives under dynamic operating conditions. Control theory has been successfully applied to such problems as task scheduling [15] [10], QoS adaptation in web servers [2], load management in e-mail and file servers [21] [16], network flow control [18], and power management [17]. The above methods all use classical feedback control to first observe the current system state and then take corrective action, if any, to achieve the specified QoS.

We propose an online control technique where the actions governing system operation are obtained by optimizing its forecast behavior, described by a mathematical model, for the specified QoS criteria over a limited look-ahead prediction horizon. The sequence of control actions resulting in the best system behavior over this horizon is obtained, and the first action within this sequence is applied as input during the current time instant while the rest are discarded. This process is repeated each time step.

Our method is conceptually similar to both model predictive control (MPC) which is widely used in the process control industry [9], and the limited look-ahead supervision of discrete-event systems [11]. The MPC approach allows the control objectives and operating constraints to be represented explicitly in the (multi-variable) optimization problem and solved at each control instant. It is used to control a variety of processes, from those with relatively simple dynamics to more complex ones, including systems with long delay or dead times, and those exhibiting non-linear behavior. Limited look-ahead supervision [11] considers the online control of a system with discrete events (inputs and outputs), where after each event occurrence, the next control action is determined using the projected behavior of the process over a limited look-ahead horizon represented as a search tree.

As a specific case study, we apply our control method to manage the power consumed by a computer processing a time-varying workload comprising HTTP and e-commerce related requests. Assuming a processor with multiple operating frequencies, an online controller is developed to achieve a specified response time for these requests while minimizing the operating frequency. Power consumption relates quadratically to the supply voltage which can be reduced at lower frequencies [20]. Therefore, energy savings can be quite significant. Furthermore, many processors such as the AMD-K-2 [1] and StrongARM [23] support multiple operating frequencies.

A control-theoretic approach to managing the power consumed by processors executing multimedia applications has been proposed in [17]. The authors assume a continuous frequency domain and use a feedback controller to scale the processor operating frequency appropriately to maintain the desired application QoS. This assumption, however, may not always hold in practice; for example, both the AMD-K-2 and StrongARM processors offer only a limited number of discrete frequency settings, eight and ten, respectively.

Unlike classical feedback control where a continuous input (output) domain is assumed, the controller proposed in this paper optimizes processor operation over a discrete state space comprising a small number of control inputs. We describe the processor model, formulate the power management problem, and develop the online controller to operate the processor within a limited and discrete frequency domain. Controller performance is evaluated using representative e-commerce workloads. We also discuss how online control can be applied to other resource management problems in computer systems [14].

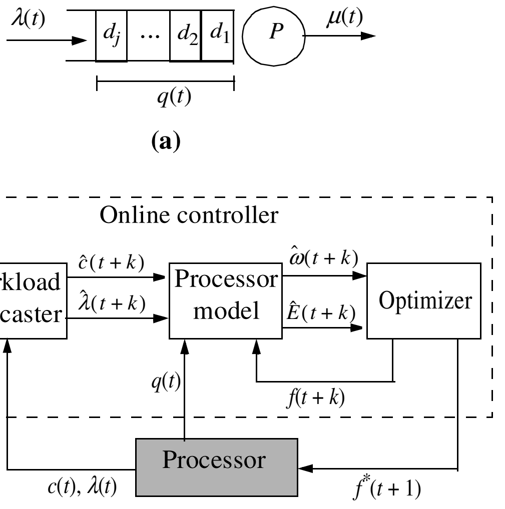


Figure 1. (a) A queuing model of the processor and (b) the overall structure of the online controller

The rest of this paper is organized as follows. Section 2 discusses system modelling assumptions and online control concepts while Section 3 develops the control algorithm. Section 4 evaluates controller performance and we conclude the paper with a discussion on future work as well as other applications of online control in Section 5.

2 Preliminaries

This section describes the assumed processor model and introduces key online control concepts.

Processor Model. Figure 1(a) shows the queuing model corresponding to processor P's operation where $\lambda(t)$ and $\mu(t)$ denote the arrival and processing rates, respectively, of requests $\{d_j\}$, and $q(t)$ denotes the queue size at time instant t. We do not assume an *a priori* arrival-rate distribution for $\{d_j\}$, and requests are processed by P in a first-come first-serve fashion.

We assume a processor capable of operating within a limited number of frequency settings $\{f_i\}$. The time required to process d_j while operating at the maximum operating frequency $f_{\max}$ is given by c_j. Then, the corresponding processing time while operating at some instantaneous frequency $f(t) \in \{f_i\}$ is $c_j/\alpha(t)$ where $\alpha(t) = f(t)/f_{\max}$ is the scaling factor. The average response time of requests arriving at P during a time t is denoted by $\omega(t)$ and includes both the waiting time in the queue and the processing overhead on P. We use the model proposed in [26] to estimate the average energy consumed by processor P while operating at $f(t)$ as

$E(t) = \alpha^2(t)$. This simple model provides reasonably accurate estimates of energy consumption.

Online Control Concepts. During any given time period, the controller aims to satisfy a designer-specified response time ω_{ref} for request arrivals while minimizing P's operating frequency $f(t)$. Figure 1(b) shows its overall structure where a mathematical model describes processor behavior in terms of the average response time $\omega(t)$ and energy consumption $E(t)$, and an optimizer minimizes the given cost function. The basic ideas behind the controller are as follows:

- Future processor outputs, in terms of average response time $\hat{\omega}(t+k)$ and energy consumption $\hat{E}(t+k)$, for a pre-determined prediction (look-ahead) horizon $k = 1, \ldots, n$ steps are estimated during each sampling instant t using the behavioral model. These predictions depend on known values (past inputs and outputs) up to the sampling instant t, and on the future control signals which are inputs to the processor that must be calculated.
- A sequence of control signals $\{f(t+k)\}$ resulting in the desired processor behavior is obtained for each step of the prediction horizon by optimizing a QoS-related cost function $J(\hat{\omega}(t+k), \hat{E}(t+k))$.
- The control signal $f^*(t+1)$ corresponding to the first frequency in the above sequence is applied as input to the processor during sampling instant t; the other inputs are rejected. During the next sampling instant, $\omega(t+1)$ and $E(t+1)$ are known and the above steps are repeated again.

3 Controller Design

This section develops the behavioral model of the processor and the online control algorithm.

Workload Forecasting. In order to estimate processor behavior over the prediction horizon, request arrival and processing rates must first be estimated. Various prediction models have been previously proposed for performance estimation of computer systems. In [25], an autoregressive model to predict trends in network traffic is developed, while [28] combines a Kalman filter with an autoregressive model to detect changes in web server workloads. The authors of [27] presents short- and long-term prediction algorithms to estimate various performance variables in a computer system including abnormal events such as QoS violations and system failures.

We now develop an appropriate forecasting model to predict request arrival rates using key characteristics of some representative e-commerce applications. Figure 2 shows HTTP requests made to a computer at ClarkNet, an Internet service provider in the Washington DC area over

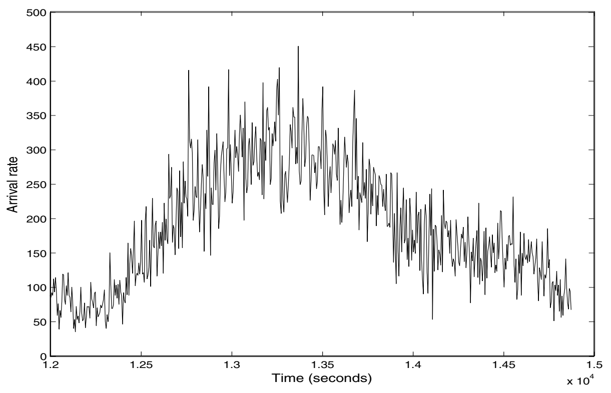


Figure 2. HTTP requests made to an Internet service provider showing cyclical variations in the arrival rate

a week [4]. This workload clearly shows cyclical trends, as do many other published ones [4] [5] [19]. Therefore, we conclude that such workloads may be predicted using a trend model, used when the increase (decrease) in a series of values persists for an extended time. The following equations describe this forecasting model [12]:

$$\hat{\lambda}(t) = \alpha \cdot \lambda(t) + (1-\alpha) \cdot (\hat{\lambda}(t-1) + \delta(t-1)) \qquad (1)$$

$$\delta(t) = \beta \cdot (\hat{\lambda}(t) - \hat{\lambda}(t-1)) + (1-\beta) \cdot \delta(t-1) \qquad (2)$$

$$\hat{\lambda}(t+k) = \hat{\lambda}(t) + k \cdot \delta(t) \qquad (3)$$

where $\lambda(t)$ and $\hat{\lambda}(t)$ denote the observed and estimated arrival rates for time t, respectively. The estimated arrival rate $\hat{\lambda}(t)$ is obtained by Equation 1 using a weighted moving average of recent past estimates and the current observation. The smoothing constant α determines the weight given to past observations and controls the rate of averaging. The trend $\delta(t)$ present in the arrival rate is detected by Equation 2 using a smoothed average of first differences, i.e., the change in arrivals from period $t+1$ to t. Finally, Equation 3 gives the forecast rate for time $t+k$ within the prediction horizon. This model is validated using a real-world workload in Section 4.

Model Dynamics. The following equations describe the dynamics of the processor model:

$$\hat{q}(t+1) = q(t) + \left(\hat{\lambda}(t+1) - \frac{\alpha(t+1)}{\hat{c}(t+1)}\right) \cdot t_S \qquad (4)$$

$$\hat{\omega}(t+1) = (1 + \hat{q}(t+1)) \cdot \hat{c}(t+1) \qquad (5)$$

$$\hat{E}(t+1) = \alpha^2(t+1) \qquad (6)$$

Given the observed queue length $q(t)$ at time t, Equation 4 estimates its length during the time interval $[t, t+1]$. The estimated processing time $\hat{c}(t+1)$ is obtained as $\hat{c}(t+1) = \gamma \cdot c(t) + (1-\gamma) \cdot \hat{c}(t)$ where $c(t)$ and $\hat{c}(t)$ denote the actual and estimated values at time t, respectively, and

```
Procedure PWMGR($\hat{x}(t)$, $\lambda(t)$) /* $\hat{x}(t)$ := initial estimate; $\lambda(t)$ := arrival rate */
    $s_t$ := { $\hat{x}(t)$ };              /* Initial state */
    for (each $t$ within a prediction horizon of depth $n$) begin/* $n$ is the depth of the prediction horizon */
    $\hat{\lambda}(t+1)$ := Forecast arrival rate using $\lambda(t)$;
    $s_{t+1} := \varnothing$;
    for (each $\hat{x}(t)$ in $s_t$) begin /* Expand the search tree one level */
        Generate all valid (reachable) states $\hat{x}(t+1)$;
        $s_{t+1} := s_{t+1} \cup \{ \hat{x}(t+1) \}$;
    end;
    for (each $\hat{x}(t+1)$ in $s_{t+1}$) begin /* Evaluate states at search level $t + 1$ */
        Estimate $\hat{\omega}(t+1)$ and $\hat{E}(t+1)$;
        Calculate the corresponding cost function $\hat{J}(t+1)$;
    end;
    $t := t + 1$;
    end;
    Obtain a sequence of reachable states $\hat{x}(t+1)$,..., $\hat{x}(t+n)$ having minimum cumulative cost;
    $f^*(t+1)$ := Operating frequency corresponding to $\hat{x}(t+1)$;
    return $f^*(t+1)$;   /* Return the control action for time $t + 1$ */
```

Figure 3. The online control algorithm

γ is the smoothing constant. The sampling period of the controller is denoted by t_s. The response times of requests arriving during the interval $[t, \ t+1]$ is estimated by Equation 5. The energy consumed by the processor while operating at frequency $f(t+1)$ is given by Equation 6 where $\alpha(t+1) = f(t+1)/f_{\max}$.

Control Algorithm. Given the average response time and energy consumption at sampling time t, the controller explores the prediction horizon comprising discrete states $\hat{x}(t+k) = (\hat{\omega}(t+k), \hat{E}(t+k))$, $k = 1, \ldots, n$ to determine the best frequency input to apply at time $t+1$. Each estimated state $\hat{x}(t+1)$ within the horizon is evaluated using the cost function $\hat{J}(t+1)$ as defined by:

$$\hat{J}(t+1) = w_1 \cdot \hat{G}(t+1) + w_2 \cdot \hat{E}(t+1) \qquad (7)$$

$$\hat{G}(t+1) = \begin{cases} 0 \text{ if } \hat{\omega}(t+1) - \omega_{\text{ref}} \le 0 \\ \dfrac{\hat{\omega}(t+1) - \omega_{\text{ref}}}{\omega_{\text{ref}}} \text{ if } \hat{\omega}(t+1) - \omega_{\text{ref}} > 0 \end{cases} \qquad (8)$$

Figure 3 shows the online control algorithm where, during each t, it accepts the initial state estimate $\hat{x}(t)$ and arrival rate $\lambda(t)$, and returns an operating frequency $f(t+1)$. The initial estimate $\hat{x}(t)$ comprises the observed $E(t)$ and $\omega(t)$ values. Starting from this state, the controller constructs, in breadth-first fashion, a tree comprising all possible future states up to the specified horizon as follows. Given an $\hat{x}(t)$ we first estimate the workload $\hat{\lambda}(t+1)$, and generate the next set of reachable processor states by applying all input frequencies from $\{f_i\}$. The cost function $\hat{J}(t+1)$ corresponding to each generated state $\hat{x}(t+1)$ is then computed. Once the prediction horizon is fully explored, a sequence of reachable states $\hat{x}(t+1)$,..., $\hat{x}(t+n)$ with the minimum cumulative cost

$$\sum_{k=1}^{n} \hat{J}(t+k)$$

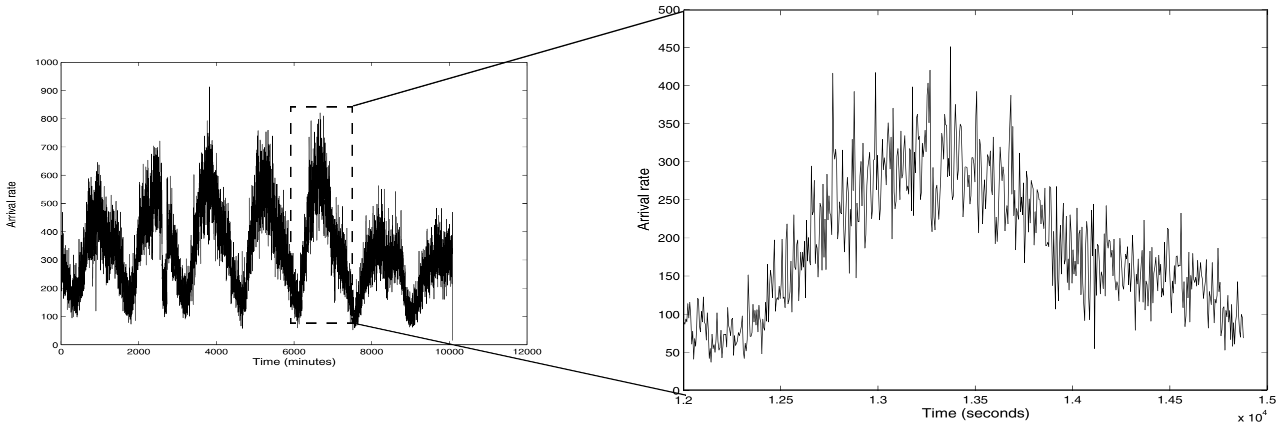
is obtained. The operating frequency $f^*(t+1)$ corresponding to $\hat{x}(t+1)$ (the first state in this sequence) is provided as input to the processor while the rest are discarded. The above control action is repeated each sampling step. In our experiments, the weights in Equation 7 were set to $w_1 = 100$ and $w_2 = 1$ to achieve a behavior where the controller is heavily penalized if a chosen operating frequency fails to satisfy ω_{ref}. However, if ω_{ref} is satisfied by multiple frequencies, the lowest frequency is chosen to minimize energy consumption.

Since the controller exhaustively evaluates all possible operating states within the prediction horizon to determine the best input to apply at time t, the overhead due to this approach must be analyzed. If $|\{f_i\}|$ denotes the size of the input set $\{f_i\}$, and n the prediction horizon depth, then the number of explored states is given by

$$\sum_{k=1}^{n} |\{f_i\}|^k$$

When both the look-ahead horizon and the number of control inputs is small, the computational overhead is negligible–as confirmed by the experiments in Section 4.

Since control actions are taken after exploring only a limited number of states, we must guarantee that the underlying physical system is online controllable [3]. Our system is, since given a state $\hat{x}(t)$, it is always possible to



(a) **(b)**

Figure 4. (a) Synthetic workload arrivals generated using the HTTP logs of an Internet service provider; (b) requests received by the computer during a day plotted in 30 second intervals

find a control input that forces the system into a different state. This implies that the controller can make continuous progress towards achieving the desired QoS objective without deadlocking.

We does not explicitly analyze the stability of the on-line controller here. However, when both the operating frequency and queue size are bounded, and if requests are simply dropped when the queue is full, then stability can be guaranteed in terms of worst-case queue size and request response times. A more detailed analysis of controller stability is left as future work.

4 Performance Evaluation

The performance of the controller is now evaluated using a representative e-commerce workload. We first describe how the workload is generated, and then discuss the obtained results.

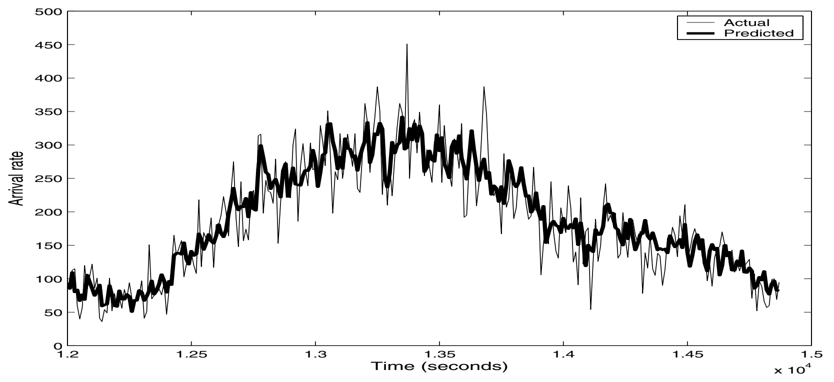
As noted in the introduction, [17] proposes a feedback controller to balance both energy consumption and QoS requirements on a processor executing multimedia applications. Our approach cannot be directly compared to [17] since the authors assume a processor capable of operating at arbitrary frequencies. By contrast, we assume a processor having a limited number of frequency settings–an AMD Athlon with possible operating frequencies of 532, 665, 798, 1197, and 1529 MHz [24].

Workload Generation. Our experiments simulated a busy server processing a synthetic yet realistic workload comprising HTTP requests. To generate the workload, we require a time-varying request arrival rate, execution times of the individual requests, and their distribution within the arrival stream. The workload arrival rate shown in Fig. 4(a) was obtained by combining real-world traces–HTTP requests made to a computer at Clarknet over a week [4]. (Unfortunately, the published log files do not have any execution time information for these requests).

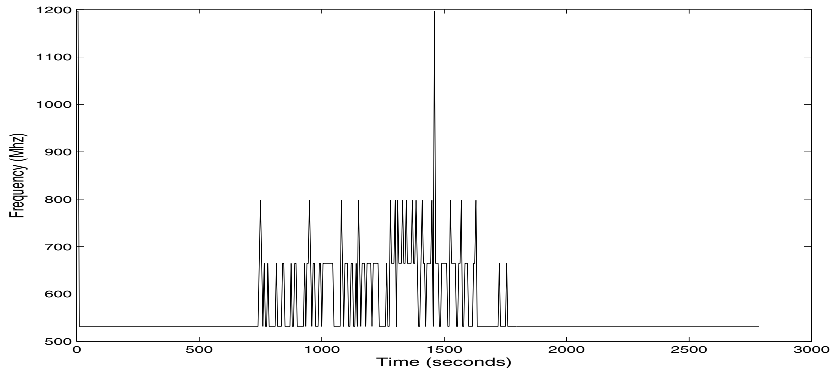
Using the rate information in Fig. 4(a), the distribution of individual requests within the arrival sequence was determined using two important characteristics of most web workloads: popularity and temporal locality. First, we generated a virtual store comprising 10,000 objects, and the time needed to process an object request was randomly chosen from a uniform distribution between $(25, 50)$ ms. Simulated requests to the store had the following characteristics:

- *Popularity*: It has been widely observed that some files are more popular than others, and that the popularity distribution commonly follows Zipf's law [4]. (A few files are extremely popular while many others are very rarely requested). Therefore, we partitioned the virtual store in two–a "popular" set with 1000 objects receiving 90% of all requests, and a "rare" set containing the remaining objects in the store receiving only 10% of requests.
- *Temporal locality*: This is the likelihood that once an object is requested, it will be requested again in the near future. In many web workloads, temporal locality follows a lognormal distribution [6].

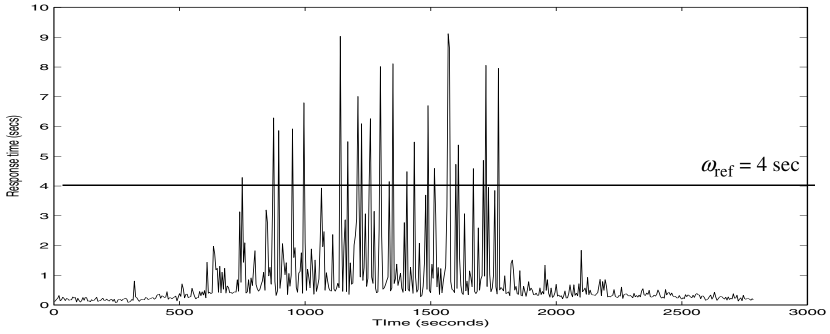
Analysis of Results. We first calibrated the trend model used to forecast arrival rates (described in Section 3). The best fit to the arrival pattern in Fig. 4(a) was obtained for smoothing constants of $\alpha = 0.17$ and $\beta = 0.1$; the goodness-of-fit measure was $\bar{R}^2 = 0.82$ and the predicted values had a mean absolute percentage error of 16% when compared to observed ones. Figure 5(a) shows the observed and predicted values overlaid on each other.



(a)



(b)



(c)

Figure 5. (a) An overlay of the workload arrival rates from Fig. 4(b) and the corresponding predictions made by the online controller; (b) the processor operating frequencies specified by the controller, and (c) the achieved response times

Though the trend model predicts the arrival rate well in most cases, it fails to track sudden surges (or spikes) in request arrivals. We compared the performance of the trend model with another widely used forecasting technique; a Box-Jenkins ARIMA model [7] was generated using Freefore [29] to best fit the observed data in Fig. 4(a). The generated model had $\bar{R}^2 = 0.83$ and an average error of 15% between predicted and observed values. Therefore, we conclude that the trend model provides an adequate fit to the data used in our experiments; the sudden spikes simply correspond to noise in the data values. Request processing times were estimated using $\gamma = 0.35$.

The performance of the controller was evaluated over a smaller portion of the overall workload, shown in Fig. 4(b), where the requests received by the computer during one day are plotted in 30 second observation intervals. We note that this interval is sufficient to smooth the variability in arrival rates and adequately track them using the prediction model. Therefore, the sampling period of the controller was set to $t_s = 30$–no smaller that the observation interval. Also, the overhead due to controller execution as well as the system dead time (the delay between changing the operating frequency and its completion) is negligible and therefore ignored in our experiments. The

Prediction horizon	% of requests satisfying QoS	Processor states explored
2	91.2%	31
3	90.1%	156
4	89.0%	781

Figure 6. Effect of prediction horizon on controller performance

response time to be achieved by the controller was set to $\omega_{\mathrm{ref}} = 4$ sec.

Figures 5(a)-(c) summarize the performance of the controller for a prediction horizon of two time steps. Figure 5(b) shows how the controller changes the operating frequency of the processor to accommodate the time-varying workload in Fig. 5(a). The achieved response times are shown in Fig. 5(c). The controller does not achieve the desired QoS during some time periods since it cannot predict sudden (and short-term) spikes in the arrival rate. The frequent switching activity in Fig. 5(b) occurs since the cost function in Equation 7 does not include a corresponding switching penalty. Though control actions in general systems typically incur some penalty, in the specific case of power management, this penalty is negligible; for example, a frequency change in the AMD-K-2 processor incurs a time overhead of only 41 µs [22]. Therefore, our cost function ignored this switching penalty.

The overall controller performance is promising; in our experiments, it achieved the desired response time $\omega_{\mathrm{ref}} = 4$ sec for about 91% of the received requests. Figure 6 shows the effect of different prediction horizons on controller performance in terms of the percentage of requests satisfying their QoS requirement. Increasing the horizon does not improve performance; in fact, performance suffers slightly. This may be due to the fact that for this specific workload, model forecasting errors accumulate with increasing horizon depth, thereby degrading controller performance.

To summarize this section, our experiments imply that optimal solutions for such online control problems do not exist, particularly when the arrival rates are unpredictable and potentially unbounded. The system designer must, therefore, decide upon an acceptable controller configuration after sufficient experimentation. In this specific case, a short-term forecasting horizon of depth two appears appropriate. Also, controller performance may be further enhanced by improving the behavioral model of the processor and/or the cost function–both are topics for future work. Finally, the controller overhead corresponding to the prediction horizons in Fig. 6 was found to be negligible, and hence not reported.

5 Discussion

This paper has addressed the design of self-optimizing computer systems using a generic online control framework. As a specific application of this technique, we showed how to minimize the power consumed by a computer by designing a controller to satisfy the QoS requirements of a time-varying workload while operating the processor at the lowest possible frequency. Its performance was evaluated using representative e-commerce workloads with encouraging results.

As future work, we plan to further improve controller performance. The QoS violations seen in Fig. 5(c) suggest that Equations 4, 5, and 6 describing the model dynamics are somewhat sensitive to noise in the observed data. We will enhance model accuracy and robustness by including the appropriate prediction error while forecasting the arrival and processing rates. Also, the controller can be designed to operate within a QoS region instead of the single reference point ω_{ref} considered in this paper.

The proposed control approach is very general and is applicable to other resource management problems in computer systems. In [14], we developed an online controller to operate a distributed computer system in energy-efficient fashion while satisfying the QoS requirements of a dynamic workload; computers are switched on (off) as needed to accommodate the time-varying workload. Online predictive control is especially useful when control actions have substantial dead times (such as switching on a computer). We have evaluated this approach on real-world e-commerce workloads with encouraging results. We also believe a similar control approach can help design self-healing distributed systems. Certain computer failures may be predicted shortly before their occurrence by analyzing the corresponding performance variables [27]. The controller can then initiate the appropriate reconfiguration action such as switching on a backup computer in anticipation of such failures to prevent service disruptions.

6 References

[1] Advanced Micro Devices Corp., Mobile AMD-K6-2+ Processor Data Sheet, Publication 23446, June 2000.

[2] T. F. Abdelzaher, K. G. Shin, and N. Bhatti, "Performance Guarantees for Web Server End-Systems: A Control Theoretic Approach," *IEEE Trans. Parallel & Distributed Syst.*, vol. 13, no. 1, pp. 80-96, January 2002.

[3] S. Abdelwahed et al., "Online Safety Control of a Class of Hybrid Systems," *Proc. Conf. Decision & Control*, pp. 1988-1990, 2002.

[4] M. F. Arlitt and C. L. Williamson, "Web Server Workload Characterization: The Search for Invariants," *Proc. ACM SIGMETRICS Conf.*, pp. 126-137, 1996.

[5] M. Arlitt and T. Jin, "Workload Characterization of the 1998 World Cup Web Site," *Tech. Report, HPL*-99-35R1, Hewlett-Packard Labs., September 1999.

[6] P. Barford and M. Crovella, "Generating Representative Web Workloads for Network and Server Performance Evaluation," *Proc. ACM SIGMETRICS Conf.*, pp. 151-160, 1998.

[7] G. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis: Forecasting and Control*, 3rd Edition, Prentice-Hall, Upper Saddle River, New Jersey, 1994.

[8] T. D. Burd and R. W. Brodersen, "Energy Efficient CMOS Microprocessor Design," *Proc. Hawaii Int'l Conf. Syst. Sciences*, pp. 288-297, 1995.

[9] E. F. Camacho and C. Bordons, *Model Predictive Control*, Springer-Verlag, London, 1999.

[10] A. Cervin, J. Eker, B. Bernhardsson, and K. Arzen, "Feedback-Feedforward Scheduling of Control Tasks," *J. Real-Time Syst.*, vol. 23, no. 1-2, July/September 2002.

[11] S. L. Chung, S. Lafortune, and F. Lin, "Limited Lookahead Policies in Supervisory Control of Discrete-Event Systems," *IEEE Trans. Automatic Control*, vol. 37, no. 12, pp. 1921-1935, December 1992.

[12] S. A. DeLurgio, *Forecasting Principles and Applications*, McGraw-Hill Intl., Singapore, 1998.

[13] A. G. Ganek and T. A. Corbi, "The Dawn of the Autonomic Computing Era," *IBM Systems Journal*, vol. 42, no. 1, pp. 5-18, 2003.

[14] N. Kandasamy and S. Abdelwahed, "Designing Self-Managing Distributed Systems via On-Line Predictive Control," *Tech. Report ISIS*-03-404, Vanderbilt University, 2003.

[15] C. Lu, J. A. Stankovic, S. H. Son, and G. Tao, "Feedback Control Real-Time Scheduling: Framework, Modeling, and Algorithms," *J. Real-Time Syst.*, vol. 23, no. 1-2, pp. 85-126, July/September 2002.

[16] C. Lu, G. A. Alvarez, and J. Wilkes, "Aqueduct: Online Data Migration with Performance Guarantees," *Proc. USENIX Conf. File Storage Tech.*, pp. 219-230, 2002.

[17] Z. Lu et al., "Control-Theoretic Dynamic Frequency and Voltage Scaling for Multimedia Workloads," *Proc. Int'l Conf. Compilers, Architectures, & Synthesis Embedded Syst.* (*CASES*), pp. 156-163, 2002.

[18] S. Mascolo, "Classical Control Theory for Congestion Avoidance in High-Speed Internet," *Proc. Conf. Decision & Control*, pp. 2709-2714, 1999.

[19] D. Menasce et al., "In Search of Invariants for E-Business Workloads," *ACM Conf. Electronic Commerce*, pp. 56-65, 2000.

[20] T. Mudge, "Power: A First-Class Architectural Design Constraint," *IEEE Computer*, vol. 34, no. 4, pp. 52-58, April 2001.

[21] S. Parekh et al., "Using Control Theory to Achieve Service level Objectives in Performance Management," *J. Real Time Systems*, vol. 23, no. 1-2, pp. 127-141, July/September 2002.

[22] P. Pillai and K. G. Shin, "Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems," *Proc. Symp. Operating Syst. Principles* (*SOSP*), pp. 89-102, 2001.

[23] J. Pouwelse, K. Langendoen, and H. Sips, "Dynamic Voltage Scaling on a Low-Power Microprocessor," *Proc. Conf. Mobile Computing & Networking* (*MOBICOM*), pp. 251-259, 2001.

[24] V. Sharma et al., "Power-aware QoS Management in Web Servers," *Proc. Real-Time Syst. Symp.*, pp. 63-72, 2003.

[25] D. Shen and J. L. Hellerstein, "Predictive Models for Proactive Network Management: Application to a Production Web Server," *Proc. Network Operations & Management Symp.*, pp. 833-846, 2000.

[26] A. Sinha and A. P. Chandrakasan, "Energy-Efficient Real-Time Scheduling," *Proc. Int'l Conf. Computer Aided Design,* pp. 458-463, March/April 2001.

[27] R. Vilalta et al., "Predictive Algorithms in the Management of Computer Systems," *IBM Systems Journal*, vol. 41, no. 3, pp. 461-474, 2002.

[28] F. Zhang and J. L. Hellerstein, "An Approach to Online Predictive Detection," *Proc. Modeling, Analysis & Simulation Computer & Telecom. Syst.*, pp. 549 - 556, 2000.

[29] Freefore, Automatic Forecasting Systems Inc., http://www.autobox.com