Coordinated Power and Performance Guarantee with Fuzzy MIMO Control in Virtualized Server Clusters

Palden Lama and Xiaobo Zhou

Abstract—It is important but challenging to assure the performance of multi-tier Internet applications with the power consumption cap of virtualized server clusters mainly due to system complexity of shared infrastructure and dynamic and bursty nature of workloads. This paper presents PERFUME, a system that simultaneously guarantees power and performance targets with flexible tradeoffs and service differentiation among co-hosted applications while assuring control accuracy and system stability. Based on the proposed fuzzy MIMO control technique, it effectively controls both the throughput and percentile-based response time of multi-tier applications due to its novel self-adaptive fuzzy modeling that integrates the strengths of fuzzy logic, MIMO control and artificial neural network. Furthermore, we address an important challenge of pro-actively avoiding violations of power and performance targets in anticipation of future workload changes. We implement PERFUME in a testbed of virtualized blade servers hosting multi-tier RUBiS applications. Performance evaluation based on synthetic and real-world Web workloads demonstrates its control accuracy, flexibility in selecting tradeoffs between conflicting targets, service differentiation capability and robustness against highly dynamic and bursty workloads. It outperforms a representative utility based approach in providing guarantee of the system throughput, percentile-based response time and power budget.

Index Terms—Power Budget, Performance Guarantee, Multi-tier Internet Services, Fuzzy MIMO control, Proactive Control, Self Adaptation, Server Virtualization.

1 Introduction

Modern datacenters apply virtualization technology to host multiple Internet applications that share underlying high density server resources for server consolidation and system manageability. The widely used high density blade servers impose stringent power and cooling requirements. Furthermore, datacenters may adopt over-subscription practice for power savings [27], by allowing the sum of the possible peak power consumption of all the servers combined to be greater than the provisioned capacity. Hence, it is essential to precisely control power consumption to ensure that actual total power use stays below capacity.

A common technique for enforcing power budget is to dynamically transition the hardware components from high power states to low-power states [7]. However, it has significant influence on the performance of hosted applications as it may result in violation of service level agreements (SLAs) in terms of response time and throughput required by customers. Furthermore, such an approach is not easily applicable to virtualized environments where physical processors are shared by multiple virtual machines (VMs). Changing the power state of a processor will affect the performance of multiple VMs belonging to different applications. Thus, power management may threaten the performance isolation of hosted applications. It is important to consider a coordinated approach in controlling power and performance in virtualized datacenters.

Recent studies such as [33] found highly dynamic workloads of Internet services that fluctuate over multiple time scales, which can have a significant impact on the processing demands imposed on datacenter servers. Furthermore, burstiness of Internet workloads has deleterious impact on client-perceived performance [29]. It is challenging to design autonomic resource provisioning techniques that are robust to dynamic variation and burstiness in workloads.

Many research studies focused on treating either power or performance as the primary control target in a datacenter while satisfying the other objective in a best-effort manner. Power oriented approaches [25], [32], [37] disregard the SLAs of hosted applications while performance oriented approaches do not have explicit control on power consumption [3]. Recently, vPnP [8] was proposed for explicit coordination of power and performance in virtualized datacenters using utility function optimization. The approach can achieve tradeoffs between power and performance in a flexible way. However, it lacks the guarantee on stability and performance of the server system especially in the face of highly dynamic and bursty workloads.

Multiple-input-multiple-output (MIMO) control technique has been applied for performance management of multi-tier applications and power control of high density servers in an enclosure [37]. However, those MIMO control solutions do not provide explicit coordination between

- *P. Lama is with the Department of Computer Science, University of Texas at San Antonio, TX, 78249. E-mail: palden.lama@utsa.edu*
- *X. Zhou is with the Department of Computer Science, University of Colorado, Colorado Springs, CO, 80918. E-mail: xzhou@uccs.edu*
- *X. Zhou is the corresponding author.*

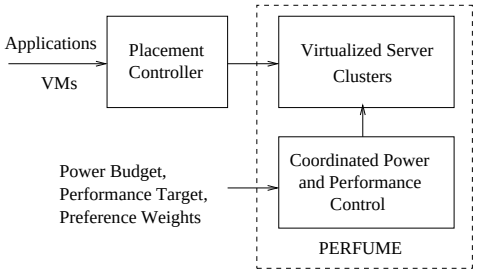


Fig. 1: PERFUME system overview.

power and performance. They are designed based on offline system identification for specific workloads [19], [37]. Thus, they are not adaptive to situations with abrupt workload changes though they can achieve control accuracy and system stability within a range theoretically.

An important goal in datacenters is to meet the SLAs with customers. Many studies focused on the average end-to-end response time within a multi-tier system [2], [8], [16], [31], [34]. However, the average response time guarantee is not sufficient for many applications, in particular for interactive ones as it is unable to represent the shape of a delay curve. Instead, providers of such services prefer percentile-based performance guarantee [26], [34], [39]. But it is challenging to control the percentile-based performance, even without a power consumption cap, due to its strong non-linear relation with resource allocation and workload dynamics [20].

In this paper, we propose and develop PERFUME, a system that simultaneously provides explicit guarantee on the power consumption of underlying server clusters and the performance of multi-tier applications hosted on them. As shown in Figure 1, various applications are hosted on a virtualized server cluster according to an interference-aware application placement policy. Such a placement policy decides which applications should be co-located so that the performance interference between applications can be minimized [28]. PERFUME provides the flexibility to the system administrator in setting the power and performance targets, and specifying the trade-offs between the cost of power consumption and the business value of providing performance assurance to various applications.

PERFUME's core is a fuzzy MIMO (FUMI) controller that minimizes the deviation of power and performance from their respective targets while assuring control accuracy and system stability. FUMI control is capable of dealing with the complexity of multi-tier applications in a shared virtualized infrastructure and the inherent non-linearities that exist in a real Web system. It is due to its integration of fuzzy modeling logic, MIMO control and artificial neural network. The control action is taken by adjusting the CPU usage limits among individual tiers of multiple applications in a coordinated manner. Furthermore, it provides service differentiation by prioritizing CPU allocations among different applications while avoiding power budget violations.

PERFUME provides performance guarantee for throughput and percentile-based response time in the face of highly dynamic and bursty workloads. Its novel FUMI control accurately captures the strong nonlinearity of percentile-based performance metric such as the 95_{th}-percentile response time by applying fuzzy models. It predicts the power consumption of the server clusters for various CPU usage limits in hosted applications. PERFUME is self-adaptive to highly dynamic workloads due to its online learning capability. It adjusts the fuzzy model parameters at run time using a weighted recursive least-squares (wRLS) method.

Furthermore, PERFUME addresses an important challenge of pro-actively avoiding violations of power and performance targets in anticipation of future workload changes. In spite of FUMI control's ability to adapt itself in the face of workload variations, target violations may occur to some extent. Such violations are even more significant in case of continuously changing workloads as seen in real-world Web traces. It is due to the purely reactive approach of updating the power and performance models in response to the measured modeling errors. Since it takes a few control intervals to accurately update the system model, the control actions may lag behind continuous variations in the workload. We enhance the proposed FUMI control by integrating a workload prediction component. The integration involves workload-aware MIMO fuzzy modeling of the virtualized server system and the design of proactive FUMI control based on this model. Our proactive FUMI control is able to make effective control decisions in anticipation of future workload changes. We apply a standard Kalman filtering technique to predict workload variations.

PERFUME is suitable for joint power and performance control at the server cabinet level. We further enhance its scalability by decomposing the global control problem into local sub-problems for each application hosted in the cluster. Then, we construct decentralized FUMI controllers for managing individual applications.

We implement PERFUME on a testbed of virtualized server clusters hosting RUBiS benchmark applications. The testbed consists of a cluster of HP ProLiant BL460C G6 blade servers using VMware VMs. We apply highly dynamic and bursty synthetic workloads as well as workloads based on real Web traces from the 1998 Soccer World Cup site [1]. Experimental results demonstrate that PERFUME's FUMI model significantly outperforms a recently applied modeling technique for Web systems, ARMA (Auto Regressive Moving Average) [8], [31] in terms of prediction accuracy for application performance and power usage.

Compared to vPnP [8], PERFUME delivers significantly improved performance, in terms of power, throughput and response time assurance with respect to the given targets in the face of highly dynamic and bursty workloads. This is due to its modeling accuracy, self-adaptiveness and control theoretical foundation. Note that vPnP was originally applied to a single tier, which was identified as the bottleneck of a Web application. However, in practice, the bottleneck tier can switch between multiple tiers depending on workload patterns. For fair comparison, we extend the vPnP implementation to a multi-tier application consisting of a front-end Web tier that is responsible for HTTP request

processing, a middle application tier that implements core application functionality, and a backend database tier.

Experimental results demonstrate that PERFUME delivers consistent performance for various control options that tradeoff between power and performance guarantee. PERFUME provides service differentiation according to the priorities of individual applications during power overload conditions. We also examine the impact of tuning important FUMI control parameters on the stability and responsiveness of the controller. We demonstrate the improvement in power and performance assurance due to the integration of workload prediction in PERFUME's architecture for proactive control. Finally, we evaluate the effectiveness of the decentralized FUMI controllers in providing application performance assurance and power control.

In the following, Section 2 discusses related work. The PERFUME system architecture is presented in Section 3. Section 4 describes the modeling of power and performance control. Section 5 discusses the design of FUMI control. Section 6 provides the testbed implementation details. Section 7 presents the experimental results and analysis. We conclude the paper with future work in Section 8.

2 Related Work

Power management in computing systems is an important and challenging research area. There were many studies in power management using the Dynamic Voltage Scaling (DVS) in embedded mobile devices and Web servers [5], [7]. Today, popular Internet applications have a multi-tier architecture forming server pipelines. Applying independent DVS algorithms in a pipeline will lead to inefficient usage of power for assuring an end-to-end delay guarantee due to the inter-tier dependency [13]. Wang *et al.* [37] proposed a MIMO controller to regulate the power consumption by conducting processor frequency scaling for each server while optimizing multi-tier application performance. Such controllers are designed based on offline system identification for specific workloads. They are not adaptive to situations with abrupt workload changes.

Modern datacenters apply virtualization technology to consolidate workloads on fewer powerful servers for improving server utilization.Traditional power management techniques are not easily applicable to virtualized environments where physical processors are shared by multiple VMs. For instance, changing the power state of a processor by DVS will inadvertently affect the performance of multiple VMs belonging to different applications.

Recent studies have dealt with the limitation of DVS technique in virtualized environments by applying 'soft' techniques which exploit a hypervisor's ability to limit hardware usage by guest VMs [8], [30]. Other studies have tackled the challenges of enforcing power budgets on virtualized environments by coordinating multiple power control knobs. Lim et al. [27] proposed a combination of hardware-based (e.g. DVS) and software-based (e.g. VM CPU time allocation) power control techniques to coordinate the power distribution among a large number of VMs within given peak power capacity. Verma *et al.* [36] combined automatic VM resizing and live migration techniques to ensure that datacenters can deal both with temporary power outages and with surges in workload.

It is a trend that power and performance management of virtualized multi-tier servers are jointly tackled. However, it is challenging due to the inherently conflicting objectives.

Power-oriented approaches aim to ensure that a server system does not violate a given power budget while maximizing the performance of hosted applications [6], [25], [32], [37], [38] or increasing the number of services that can be deployed [9]. pMapper [35] tackles power-cost tradeoffs under a fixed performance constraint. vManage [18] performs VM placement to save power without degrading performance. Co-Con [38] is a two-level control architecture for power control in virtualized server clusters. It gives a higher priority to power budget tracking and performance is a secondary goal.

Performance-oriented approaches aim to guarantee a performance target while minimizing the power consumption [15], [19], [24], [26]. They do not control the power consumption explicitly to meet the power budget.

Coordinated power and performance management with explicit trade-offs is recently studied in virtualized servers [8], [12], [16]. Mistral [16] optimizes power consumption, performance benefit, and the transient costs incurred by adaptations in virtualized server clusters. vPnP [8] coordinates power and performance in virtualized servers using utility function optimization. It provides the flexibility to choose the tradeoff between power and performance, but lacks the guarantee on system stability and performance, especially under highly dynamic workloads.

There are important studies in dynamic resource provisioning for delay guarantee in multi-tier Internet services [11], [20], [22], [26], [31], [34], [39]. For instance, Urgaonkar *et al.* [34] proposed a dynamic server provisioning approach based on queueing models, which requires extensive application profiling for each workload. An approach proposed in [39] can model the probability distributions of response time based on CPU allocations on VMs. However, the performance model is not adaptive to dynamic workloads. Furthermore, there was no power control and power-performance tradeoff capability.

Recent works have analyzed the performance interference between co-located applications in virtualized environments [10], [17]. There are performance isolation techniques based on interference-aware application placement [28] as well as dynamic resource management [23] in virtualized servers. In this paper, we consider that the first approach is in place to decide the application placement that mitigates or avoids performance interference.

The tradeoff flexibility and system stability requirements in the face of highly dynamic and bursty workloads demand novel techniques for autonomic performance and power control. In this paper, we propose a Fuzzy MIMO control to coordinate power and performance of virtualized server clusters, and provide service differentiation and proactive power control under dynamic workload and power budget.

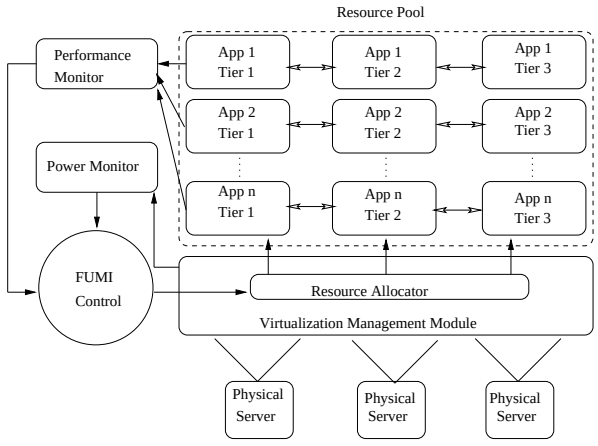


Fig. 2: The system architecture of PERFUME.

3 PERFUME System Architecture

Figure 2 illustrates the system architecture of PERFUME. The system under control is a virtualized server cluster hosting multi-tier applications. Each tier of an application is deployed at a VM created from a resource pool. The power monitor periodically measures the average power consumption of the server cluster. The performance monitor periodically measures the performance of each multi-tier application. FUMI control determines the CPU usage limits on the Web, application and database tiers of multiple applications to regulate per-application performance and the total power consumption of the server cluster. The resource allocator limits the CPU usage of each VM by using the virtualization management module. Applying CPU usage limits on VMs constrains the utilization of underlying processors, and thereby regulates power consumption. It is feasible due the idle power management of modern processors, which incorporate sleep states to achieve substantive power savings when a processor is idle [27].

4 Modeling of Coordinated Power and Performance Control

We apply fuzzy modeling to predict the performance of multi-tier applications for various CPU usage limits imposed on the VMs. Both throughput and the percentile-based response time are used as performance metrics. Fuzzy modeling also estimates the complex relationship between the power consumption of the resource pool and the CPU usage limits imposed on various tiers of the applications. A key strength of fuzzy model is its ability to represent highly complex nonlinear systems by a combination of inter-linked subsystems with simple functional dependencies. A simple linear model is not sufficient in this case due to the complex inter-tier dependencies and the underlying complexity of virtualized server infrastructure.

4.1 The Fuzzy Model

We consider a number of multi-tier applications hosted in a virtual resource pool as a MIMO system. The inputs to the system are CPU usage limits set at various tiers of the applications. The outputs of the system are the measured performance of each application and the average power consumption of the shared resource pool. We obtain two separate models for power and performance of the system, respectively. The system is approximated by a collection of MIMO fuzzy models as follows:

$$\mathrm{y}(k+1) = \mathrm{R}(\xi(k), \mathrm{u}(k)). \tag{1}$$

Let $\mathrm{y}(k)$ be the output variable and $\mathrm{u}(k) = [u_1(k), .., u_m(k)]^T$ be the vector of current inputs at sampling interval k. The regression vector $\xi(k)$ includes current and lagged outputs:

$$\xi(k) = [\mathrm{y}(k), .., \mathrm{y}(k - n_y))]^T \tag{2}$$

where n_y specifies the number of lagged values of the output variable. R is a fuzzy model consisting of a set of fuzzy rules. Each fuzzy rule is described as follows:

- *R_i: If $\xi_1(k)$ is $\Omega_{i,1}$ and .. $\xi_\varrho(k)$ is $\Omega_{i,\varrho}$ and $u_1(k)$ is $\Omega_{i,\varrho+1}$ and .. $u_m(k)$ is $\Omega_{i,\varrho+m}$ then*

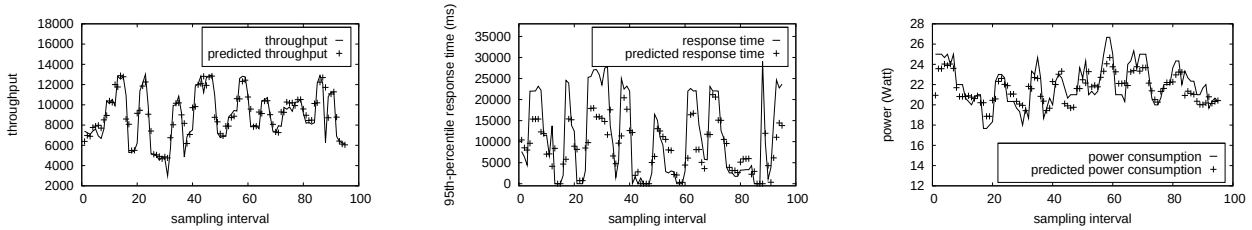
$$\mathrm{y}_i(k+1) = \zeta_i \xi(k) + \eta_i \mathrm{u}(k) + \phi_i. \tag{3}$$

Here, Ω_i is a set of fuzzy values, which describe the elements of regression vector $\xi(k)$ and the current input vector $\mathrm{u}(k)$ for the fuzzy rule, R_i. The numeric values of $\xi(k)$ and $\mathrm{u}(k)$ are mapped to fuzzy values by using the corresponding fuzzy membership functions. For example, the fuzzy membership function of $\Omega_{i,1}$ determines the degree to which it can accurately describe $\xi_1(k)$. ϱ denotes the number of elements in the regression vector $\xi(k)$. $\mathrm{y}_i(k+1)$ is the estimated model output according to the fuzzy rule R_i. ζ_i and η_i are vectors containing the consequent parameters and ϕ_i is the offset vector.

Note that the fuzzy membership functions may overlap with each other. As a result, multiple fuzzy rules can be triggered by a given set of input values. Each fuzzy rule describes a region of the complex non-linear system model using a simple functional relation given by the rule's consequent part. The contribution of each rule to the model output is determined by its firing strength, β_i. It is the product of the membership degrees of the antecedent variables in that rule. The final model output is calculated as the weighted average of the linear consequents in the K individual rules as follows.

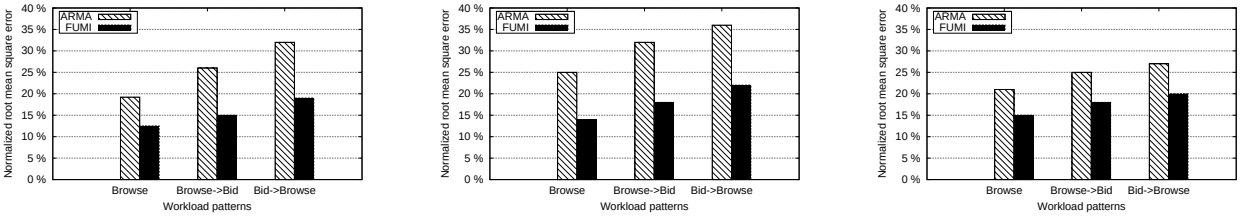
$$\mathrm{y}(k+1) = \frac{\sum_{i=1}^{K} \beta_i (\zeta_i \xi(k) + \eta_i \mathrm{u}(k) + \phi_i)}{\sum_{i=1}^{K} \beta_i} \tag{4}$$

We conduct a case study to demonstrate the accuracy of our MIMO fuzzy models in predicting the performance of a RUBiS application and the power consumption of the underlying virtualized server cluster. The data for modeling is collected by randomly allocating various CPU usage limits on the Web, application and database tiers of the RUBiS application, which faces a workload 1000 concurrent users. We construct an initial fuzzy model by applying subtractive clustering technique [4] on data collected from the system. Each obtained cluster represents a certain operating region of the system, where input-output data values are highly concentrated. Clustering partitions the input-output space to determine the number of fuzzy rules and the shape of membership functions.



(a) Throughput. (b) The 95_{th}-percentile response time. The power consumption.

Fig. 3: The prediction accuracy of performance and power by fuzzy models.



(a) Throughput. (b) The 95_{th}-percentile response time. The power consumption.

Fig. 4: The prediction accuracy comparison between fuzzy and ARMA models under a dynamic workload.

In this case study, we obtain four clusters in a five dimensional space. The first four dimensions correspond to the three input variables, $[u_1(k), u_2(k), u_3(k)]$, and one regression vector element $\xi_1(k)$. The fifth dimension corresponds to the output variable, $\mathrm{y}(k)$, which is expressed as a linear function of the input variables. Each cluster center describes a fuzzy rule in which the fuzzy values Ω_i corresponding to $\mathrm{u}(k)$ and $\xi(k)$ are represented by gaussian membership functions. The first four dimensions of the cluster center determine the mean of the gaussian functions. The function variance is determined by a tunable parameter in the subtractive clustering technique. For example, the four cluster centers in our performance model are [0.46, 0.61, 0.53, 0.6], [0.97, 0.4, 0.63, 0.81], [0.91, 0.87, 0.49, 0.94], and [0.55, 0.65, 0.93, 0.96]. These are normalized values with respect to the maximum CPU usage limits imposed at the three tiers of the RUBiS application and the average throughput in the previous sampling interval. We apply an adaptive network based fuzzy inference system [14] to further tune the fuzzy model parameters.

The consequent parameters of the fuzzy rules in our performance model corresponding to browsing and bidding workload mixes are shown in Table 1. Here, η_1, η_2, and η_3 represent the impact of CPU allocation at the Web, application, and database tiers respectively on the predicted application performance according to the four fuzzy rules. These values are significantly different for the two workload mixes since they exhibit different inter-tier dependencies. The number of fuzzy rules is the same in both cases, since the range of input-output space does not vary a lot for the two workload mixes with the same workload intensity.

Figures 3(a), 3(b) and 3(c) show the accuracy of our fuzzy models in predicting the application throughput, the 95_{th}-percentile response time and the average power consumption for various CPU allocations in the face of a browsing workload mix. The accuracy is measured by the normalized root mean square error (NRMSE), a standard metric for deviation. The case study show that the checking and the predicted data are very close, with the NRMSE 12.5%, 17.6% and 15.2% in the three scenarios respectively. We use different data sets for training and validating the system models. We observe similar prediction accuracy of fuzzy models in case of the bidding workload mix.

TABLE 1: Consequent parameters of fuzzy rules in the performance model for browsing and bidding workload mixes.

Rule	workload	η_1	η_2	η_3	ζ_1	ϕ
1	browse	0.183	0.144	0.106	0.021	0.108
	bid	0.108	0.243	0.162	0.02	0.107
2	browse	0.291	0.229	0.168	0.025	0.08
	bid	0.173	0.39	0.26	0.021	0.06
3	browse	0.317	0.25	0.183	0.03	0.09
	bid	0.182	0.409	0.273	0.027	0.1
4	browse	0.27	0.213	0.156	0.024	0.107
	bid	0.156	0.352	0.234	0.026	0.11

4.2 On-line Adaptation of the Fuzzy Model

Internet workloads to a datacenter vary dynamically in arrival rates as well as characteristics [33]. This results in significantly varying resource demands at multiple tiers of Internet applications. A static system model can not provide sufficient prediction accuracy of power and performance for all possible variations in the workload. Hence, the system models need to adapt on-line in the face of dynamic workloads. We apply a wRLS method to adapt the consequent parameters of the fuzzy model obtained. The technique continuously samples new measurements from the runtime system. It updates the model parameters in response to the errors made by the existing fuzzy models in predicting the performance and power consumption. The recursive nature of the wRLS method makes the time taken for this computation negligible for a control interval that is more

than 10 seconds. It applies exponentially decaying weights on the sampled data so that higher weights are assigned to more recent observations.

We express the fuzzy model output in Eq. (4) as follow:

$$\mathrm{y}(k+1) = X\theta(k) + e(k) \tag{5}$$

where e(k) is the error value between actual output of the system (i.e., measured performance or power) and predicted output of the model. $\theta = [\theta_1^T \theta_1^T .. \theta_p^T]$ is a vector composed of the model parameters. $X = [w_1 X(k), w_2 X(k), .., w_p X(k)]$ where w_i is the normalized degree of fulfillment or firing strength of i^{th} rule and $X(k) = [\xi^T(k), \mathrm{u}(k)]$ is a vector containing current and previous outputs and inputs of the system. The parameter vector $\theta(k)$ is estimated so that the following cost function is minimized. That is,

$$Cost = \sum_{j=1}^{k} \gamma^{k-j} e^2(j). \tag{6}$$

Here γ is a positive number less than one. It determines in what manner the current prediction error and old errors affect the update of parameter estimation. The parameters of fuzzy model are updated according to the wRLS method as follows:

$$\theta(k) = \theta(k-1) + Q(k)X(k-1)[y(k) - X(k-1)\theta(k-1)]. \tag{7}$$

$$Q(k) = \frac{1}{\gamma}[Q(k-1) - \frac{Q(k-1)X(k-1)X^T(k-1)Q(k-1)}{\gamma + X^T(k-1)Q(k-1)X(k-1)}]. \tag{8}$$

Here $Q(k)$ is the updating matrix. The initial value of $\theta(0)$ is equal to the value obtained in the off-line identification. And, the initial value of $Q(0)$ is equal to $(X^T X)^{-1}$.

To evaluate the self-adaptiveness of our fuzzy model, we measure its power and performance prediction accuracy when a RUBiS workload is changed from browsing mix of 1000 concurrent users to bidding mix of 500 concurrent users and vice versa. Our results are compared with a popular and recently used technique for modeling Internet systems, ARMA [8], [31]. As shown in Figures 4(a) and 4(b), our fuzzy model outperforms ARMA model in predicting performance of a multi-tier application. On average, the improvement in performance prediction accuracy for the throughput and 95_{th}-percentile end-to-end response time are 35% and 43%, respectively. The improvement in power prediction accuracy is shown in Figure 4(c). Compared to ARMA, our fuzzy models are more accurate in capturing the non-linear relationship between resource allocation and performance or power in a virtualized server system.

4.3 Integration of workload-aware fuzzy modeling for proactive FUMI Control

Although effective, the online adaptation of fuzzy model takes a few control intervals to capture the changing system behavior. As a result, the control performance may be degraded if the workload is continuously changing. We address this challenge by the integration of workload-aware fuzzy modeling to achieve proactive control in anticipation of future workload changes. The novelty of proactive FUMI control lies in its ability to consider the impact of future workload changes as well as current control actions while solving the MIMO control problem.

We incorporate the time varying workload intensity as a measured disturbance in the system model. The system is now approximated by a collection of MIMO fuzzy models as follows:

$$\mathrm{y}(k+1) = \mathrm{R}(\xi(k), \lambda(k), \mathrm{u}(k)). \tag{9}$$

where the workload intensity at the sampling interval k is denoted by $\lambda(k)$. Each fuzzy rule in the model is described as follows:

- *R_i: If $\xi_1(k)$ is $\Omega_{i,1}$ and .. $\xi_\varrho(k)$ is $\Omega_{i,\varrho}$ and $\lambda(k)$ is $\Omega_{i,\varrho+1}$ and $u_1(k)$ is $\Omega_{i,\varrho+2}$ and .. $u_m(k)$ is $\Omega_{i,\varrho+m+1}$ then*

$$\mathrm{y}_i(k+1) = \zeta_i \xi(k) + \theta_i \lambda(k) + \eta_i \mathrm{u}(k) + \phi_i. \tag{10}$$

Here, a set of fuzzy values denoted by Ω_i describes $\lambda(k)$ in addition to other model parameters. The model output is calculated as:

$$\mathrm{y}(k+1) = \frac{\sum_{i=1}^{K} \beta_i (\zeta_i \xi(k) + \theta_i \lambda(k) + \eta_i \mathrm{u}(k) + \phi_i)}{\sum_{i=1}^{K} \beta_i} \tag{11}$$

It can also be expressed in the form of

$$\mathrm{y}(k+1) = \zeta^* \xi(k) + \theta^* \lambda(k) + \eta^* \mathrm{u}(k) + \phi^*. \tag{12}$$

The aggregated parameters ζ^*, θ^*, η^* and ϕ^* are the weighted sum of vectors ζ_i, θ_i, η_i and ϕ_i respectively. They are applied to obtain a state-space system model and transform the complex control problem into a computationally efficient quadratic programming problem, which is described in Section 5.2.

We construct a workload-aware fuzzy MIMO model by applying subtractive clustering technique and adaptive network based fuzzy inference system on the data collected from the virtualized server system hosting multi-tier applications. For data collection, we measure the power consumption and performance achieved by the system for various combinations of CPU allocations and workload intensities. The power and performance models, which are learnt from the training data, have the ability to generalize the expected system behavior for previously unseen workloads and CPU allocations. Our workload-aware fuzzy model consists of six fuzzy rules.

At run time, we apply the wRLS method to update the model parameters if the system behaves differently than expected by the fuzzy model. Such a situation could arise due to change in the workload characteristics.

5 FUMI Control Design

We apply the model predictive control principle and fuzzy modeling to design the FUMI control. FUMI control is well suited for power and performance control in virtualized server clusters due to its capability to solve constrained MIMO control problems of complex non-linear systems. It determines control actions by optimizing a cost function, which expresses the control objectives and constraints over a time interval. We formulate the power and performance assurance of virtualized multi-tier applications as a predictive control problem. Then, we present detailed steps to transform the control formulation to a standard quadratic programming problem.

5.1 FUMI Control Problem Formulation

FUMI control aims to minimize the deviation of power consumption and performance of multi-tier applications from their respective targets. It decides the control actions at every control period k by minimizing the cost function:

$$\begin{aligned} V(k) &= \sum_{i=1}^{H_p} ||r1 - y_1(k+i)||_P^2 + \sum_{i=1}^{H_p} ||r2 - y_2(k+i)||_Q^2 \\ &+ \sum_{j=0}^{H_c-1} ||\Delta \mathrm{u}(k+j)||_R^2. \end{aligned} \quad (13)$$

Here, $y_1(k)$ is the power consumption of the resource pool. $y_2(k)$ is a vector containing the percentile-based end-to-end response time or the throughput of each application. The controller predicts both power and performance over H_p control periods, called the *prediction horizon*. It computes a sequence of control actions $\Delta \mathrm{u}(k), \Delta \mathrm{u}(k+1), .., \Delta \mathrm{u}(k+H_c-1)$ over H_c control periods, called the *control horizon*, to keep the predicted power and performance close to their pre-defined targets $r1$ and $r2$ respectively. The control action is the change in CPU usage limits imposed on various tiers of the multi-tier applications. P and Q are the tracking error weights that determine the trade-off between power and performance. The tracking error weights also facilitate service differentiation between different applications sharing the resource pool. The third term represents the control penalty, weighted by R. It penalizes big changes in control action and contributes towards system stability.

A system administrator can select the power and performance targets and their respective weighting parameters by using an independent optimization framework, or by following best practices, based on the business value of providing various levels of performance and the cost of power consumption. A higher preference weight can be set for power consumption control, when the power budget needs to be enforced. However, temporary violations of power budgets are allowable, as long as they are bounded. Thermal failover happens only when the power budget is violated long enough to create enough heat that increases the temperature beyond normal operational ranges. Hence, we do not treat power budget as a hard constraint in our problem formulation.

The control problem is subject to the constraint that the sum of CPU usage limits assigned to all multi-tier applications must be bounded by the total CPU capacity of the resource pool. The constraint is formulated as:

$$\sum_{m=1}^{M} (\Delta u_m(k) + u_m(k)) \leq U_{max} \quad (14)$$

where M is the number of applications hosted and U_{max} is the total CPU capacity of the resource pool. The use of resource pool enables resource management at the server cluster level, independently of the actual hosts that contribute to the resources. Hence, we do not consider the CPU capacity constraint of individual physical server.

5.2 Transformation to Quadratic Programming

We transform the non-convex, time-consuming optimization involved in the MIMO control problem into a standard quadratic programming problem, which can be solved efficiently at run time. We express the objective of FUMI control, defined by Eq. (13), as a quadratic program:

$$\text{Minimize } \frac{1}{2}\Delta \mathrm{u}(k)^T H \Delta \mathrm{u}(k) + c^T \Delta \mathrm{u}(k) \quad (15)$$

$$\text{subject to constraint } \Omega \Delta \mathrm{u}(k) \leq \omega.$$

The matrices Ω and ω are chosen to formulate the constraints on CPU resource usage as described in Eq. (14). Here, $\Delta \mathrm{u}(k)$ is a matrix containing the CPU usage limits on each VM over the entire control horizon H_c

For this transformation, first we linearize the fuzzy model at the current operating point and represent it as a state-space linear time variant model in the following form:

$$\begin{aligned} x_{lin}(k+1) &= A(k)x_{lin}(k) + B_{\mathrm{u}}(k)\mathrm{u}(k) + B_{\lambda}(k)\lambda(k). \\ \mathrm{y}_{lin}(k) &= C(k)x_{lin}(k). \end{aligned} \quad (16)$$

The vector for the state-space description is defined as

$$x_{lin}(k+1) = [\xi^T(k), 1]^T. \quad (17)$$

The matrices $A(k)$,$B_{\mathrm{u}}(k)$,$B_{\lambda}(k)$ and $C(k)$ are constructed by freezing the parameters of the fuzzy model at a certain operating point $\mathrm{y}(k)$ and u(k). Comparing Eq. (12) and Eq. (16), the state matrices are computed as follows:

$$A = \begin{bmatrix} \zeta^*_{1,1} & \zeta^*_{1,2} & .. & .. & .. & \zeta^*_{1,\varrho} & \phi^*_1 \\ 1 & 0 & .. & & & 0 & 0 \\ & & \vdots & & & & \\ 0 & 1 & & & & 0 & 0 \\ \vdots & \vdots & & & & \vdots & \vdots \\ \zeta^*_{2,1} & \zeta^*_{2,2} & .. & .. & .. & \zeta^*_{2,\varrho} & \phi^*_2 \\ 0 & \vdots & & & & \vdots & \vdots \\ \zeta^*_{p,1} & \zeta^*_{p,2} & .. & .. & .. & \zeta^*_{p,\varrho} & \phi^*_p \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & .. & 0 & .. & 0 & 0 & 1 \end{bmatrix}$$

$$B_{\mathrm{u}} = \begin{bmatrix} \eta^*_{1,1} & \eta^*_{1,2} & .. & \eta^*_{1,m} \\ 0 & .. & .. & 0 \\ \vdots & & & \vdots \\ \eta^*_{2,1} & \eta^*_{2,2} & .. & \eta^*_{2,m} \\ 0 & .. & .. & 0 \\ \vdots & & & \vdots \\ \eta^*_{p,1} & \eta^*_{p,2} & .. & \eta^*_{p,m} \\ 0 & .. & .. & 0 \\ 0 & .. & .. & 0 \end{bmatrix} \quad B_{\lambda} = \begin{bmatrix} \theta^*_1 \\ 0 \\ \vdots \\ \theta^*_2 \\ 0 \\ \vdots \\ \theta^*_p \\ 0 \\ 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & .. & .. & .. & .. & 0 \\ \vdots & & & & & & \vdots \\ 0 & .. & .. & .. & .. & 1 & 0 \end{bmatrix}$$

where ζ^*_{ij} is the j^{th} element of aggregate parameter vectors ζ^* for application i. η^*_{ij} is the j^{th} element of aggregate parameter vectors η^* for application i. θ^*_i is the aggregate parameter θ^* for application i.

To ensure offset-free reference tracking, the optimization problem is defined with respect to the increment in the control signal, $\Delta \mathrm{u}(k)$, rather than the control signal $\mathrm{u}(k)$. The state-space description is extended correspondingly as follows:

$$\begin{bmatrix} x_{lin}(k+1) \\ \mathrm{u}(k) \end{bmatrix} = \begin{bmatrix} A(k) & B_{\mathrm{u}}(k) \\ 0 & I \end{bmatrix} \begin{bmatrix} x_{lin}(k) \\ \mathrm{u}(k-1) \end{bmatrix}$$

$$+ \begin{bmatrix} B_{\mathrm{u}}(k) \\ I \end{bmatrix} \Delta \mathrm{u}(k) + \begin{bmatrix} B_{\lambda}(k) \\ 0 \end{bmatrix} \lambda(k)$$

$$\mathrm{y}_{lin} = \begin{bmatrix} C(k) & 0 \end{bmatrix} \begin{bmatrix} x_{lin}(k) \\ \mathrm{u}(k-1) \end{bmatrix}$$

$$\Updownarrow$$

$$\begin{aligned} X(k+1) &= \bar{A}X(k) + \bar{B}_{\mathrm{u}}\Delta\mathrm{u}(k) + \bar{B}_{\lambda}\lambda(k). \\ \mathrm{Y}(k) &= \bar{C}X(k). \end{aligned} \tag{18}$$

We obtain the state-space description, shown in Eq. (18), corresponding to both power and performance models. Henceforth, we use the notations $\bar{A}_1$, $\bar{B}_{1\mathrm{u}}$, $\bar{B}_{1\lambda}$, $\bar{C}_1$, $X_1(k)$, $Y_1(k)$ and $\bar{A}_2$, $\bar{B}_{2\mathrm{u}}$, $\bar{B}_{2\lambda}$, $\bar{C}_2$, $X_2(k)$, $Y_2(k)$ for the state matrices that describe the performance model and the power model respectively.

Assuming that at time instant k, the state vector, the future control sequence and the future workload are known, the future process outputs can be predicted through successive substitution. The complete output sequence over the prediction horizon H_p is given by the following:

$$\begin{bmatrix} Y_i(k+1) \\ Y_i(k+2) \\ \vdots \\ Y_i(k+H_p) \end{bmatrix} = R_{ix}\bar{A}_i X_i(k) + R_{iu}\begin{bmatrix} \Delta\mathrm{u}(k) \\ \Delta\mathrm{u}(k+1) \\ \vdots \\ \Delta\mathrm{u}(k+H_c-1) \end{bmatrix} + R_{i\lambda}\begin{bmatrix} \lambda(k) \\ \lambda(k+1) \\ \vdots \\ \lambda(k+H_p) \end{bmatrix}$$

where

$$R_{ix} = \begin{bmatrix} \bar{C}_i \\ \bar{C}_i\bar{A}_i \\ \vdots \\ \bar{C}_i\bar{A}_i^{H_p-1} \end{bmatrix}$$

$$R_{iu} = \begin{bmatrix} \bar{C}_i\bar{B}_i & 0 & .. & 0 \\ \bar{C}_i\bar{A}_i\bar{B}_{iu} & \bar{C}_i\bar{B}_{iu} & .. & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \bar{C}_i\bar{A}_i^{H_p-1}\bar{B}_{iu} & \bar{C}_i\bar{A}_i^{H_p-2}\bar{B}_{iu} & .. & \bar{C}_i\bar{A}_i^{H_p-H_c}\bar{B}_{iu} \end{bmatrix}$$

$$R_{i\lambda} = \begin{bmatrix} \bar{C}_i\bar{B}_{i\lambda} & 0 & .. & 0 \\ \bar{C}_i\bar{A}_i\bar{B}_{i\lambda} & \bar{C}_i\bar{B}_{i\lambda} & .. & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \bar{C}_i\bar{A}_i^{H_p-1}\bar{B}_{i\lambda} & \bar{C}_i\bar{A}_i^{H_p-2}\bar{B}_{i\lambda} & .. & \bar{C}_i\bar{A}_i^{H_p-H_c}\bar{B}_{i\lambda} \end{bmatrix}$$

Combining the output sequence over the prediction horizon with the FUMI control objective, we get a quadratic programming problem defined in Eq. (15) where,

$$\begin{aligned} H &= 2(R_{1u}^T P R_{1u} + R_{2u}^T Q R_{2u} + R). \\ c &= 2[R_{1u}^T P^T(R_{1x}\bar{A}_1 X_1(k) - r1 + R_{1\lambda}\lambda) \\ &\quad + R_{2u}^T Q^T(R_{2x}\bar{A}_2 X_2(k) - r2 + R_{2\lambda}\lambda)]^T. \end{aligned} \tag{19}$$

where

$$\lambda = \begin{bmatrix} \lambda(k) \\ \lambda(k+1) \\ \vdots \\ \lambda(k+H_p) \end{bmatrix}$$

The integration of workload intensity λ in the FUMI control problem enables proactive control in the face of dynamic workload variations. The future values of workload intensity $\lambda(k)$ over the prediction horizon H_p are predicted by applying Kalman filtering technique.

Note that we use the workload-aware fuzzy model for deriving a proactive FUMI control solution, assuming that workload predictions are sufficiently accurate. When the workload variations are abrupt and unpredictable, FUMI control uses the fuzzy model described in Section 4.1. In this case, we obtain the following:

$$\begin{aligned} H &= 2(R_{1u}^T P R_{1u} + R_{2u}^T Q R_{2u} + R). \\ c &= 2[R_{1u}^T P^T(R_{1x}\bar{A}_1 X_1(k) - r1) + R_{2u}^T Q^T(R_{2x}\bar{A}_2 X_2(k) - r2)]^T. \end{aligned} \tag{20}$$

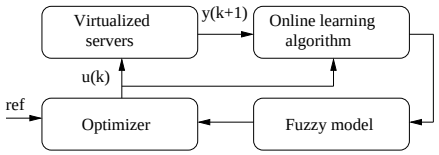


Fig. 5: Interface between FUMI control and learning components.

The matrices R_{1x}, R_{1u} are associated with the performance models of hosted applications and matrices R_{2x}, R_{2u} are associated with the power model of the resource pool.

$$R_{ix} = \begin{bmatrix} \bar{C}_i \\ \bar{C}_i\bar{A}_i \\ \vdots \\ \bar{C}_i\bar{A}_i^{H_p-1} \end{bmatrix}$$

$$R_{iu} = \begin{bmatrix} \bar{C}_i\bar{B}_i & 0 & .. & 0 \\ \bar{C}_i\bar{A}_i\bar{B}_i & \bar{C}_i\bar{B}_i & .. & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \bar{C}_i\bar{A}_i^{H_p-1}\bar{B}_i & \bar{C}_i\bar{A}_i^{H_p-2}\bar{B}_i & .. & \bar{C}_i\bar{A}_i^{H_p-H_c}\bar{B}_i \end{bmatrix}$$

5.3 FUMI Control Interface

Figure 5 shows the interface between the FUMI online learning and MIMO control components. The continuous chain of interactions between various components of the FUMI control system is described by the Algorithm 1.

Algorithm 1 The Control Algorithm.

1: Start with MIMO fuzzy models that are constructed from offline data collected from the system.
2: **loop**
3: Linearize the fuzzy models at the current operating state and obtain the state-space matrices: $\bar{A}_1$, $\bar{B}_{1\mathrm{u}}$, $\bar{B}_{1\lambda}$, $\bar{C}_1$, $\bar{A}_2$, $\bar{B}_{2\mathrm{u}}$, $\bar{B}_{2\lambda}$, and $\bar{C}_2$.
4: Predict the workload intensity of the hosted applications over the prediction horizon, H_p, if workload prediction capability is available.
5: Solve the quadratic programming problem described in Section 5.2. Obtain a sequence of CPU resource adjustments, $u(k)$, over the control horizon, H_c.
6: Apply the first control action from the computed sequence of actions, in terms of CPU resource adjustments on the hosted applications.
7: Adapt the fuzzy models using the wRLS method, in response to any modeling errors observed.
8: **end loop**

5.4 FUMI Control Scalability Enhancement

We now present a methodology for decentralizing FUMI control, with the aim to further enhance its scalability. Given a total power budget of $r1$ and a total CPU capacity of U_{max} for a virtualized server cluster, we assign the power budget and resource constraint for each application according to its priority as follows:

$$r1_a = r1 \cdot Q(a) \tag{21}$$

$$U_{max,a} = U_{max} \cdot Q(a) \tag{22}$$

Here, $r1_a$ and $U_{max,a}$ are the power budget and the maximum amount of CPU resources that can be allocated to application a, respectively. $Q(a)$ is the priority of application a as determined by the datacenter administrator based on the business value of providing performance assurance to application a. The performance target of each application is determined by its SLA. Thus, we can decompose the global control problem given by Eq. (13) into local subproblems for individual applications. Each local control problem is solved by a decentralized FUMI controller. We follow the same FUMI design principles to construct such decentralized controllers. The control solutions are obtained by solving the quadratic programming problem given by Eq. (15). The CPU allocations set by these individual controllers are feasible as long as the total amount of CPU allocated across all the applications do not exceed the server cluster capacity. It is mainly due to the use of VMware resource pool, which enables resource management and load balancing at the server cluster level. The decentralized control approach is scalable to the number of applications hosted in the cluster since it reduces the problem size drastically by allowing each controller to manage one application only.

6 PERFUME IMPLEMENTATION

6.1 The Testbed

We have implemented PERFUME on a testbed consisting of two HP ProLiant BL460C G6 blade server modules and a HP EVA storage area network with 10 Gbps Ethernet and 8 Gbps Fibre/iSCSI dual channels. Each blade server is equipped with Intel Xeon E5530 2.4 GHz quad-core processor and 32 GB PC3 memory. We use VMware ESX 4.1 for server virtualization, and create a resource pool from the virtualized server cluster to host multi-tier applications. An important feature of VMware resource pool is that the VMs do not have a static mapping with the physical servers. VMware's Distributed Resource Scheduling mechanism dynamically changes the mapping for load balancing. Each tier of an application is hosted inside a VM with 2 VCPUs, 4GB RAM and 15GB disk space. The guest operating system used is Ubuntu Linux 10.04.

As many related studies [8], [31], [34], [39], our work uses a multi-tier application benchmark RUBiS in the testbed. RUBiS implements the core functionality of an auction site. It has nine tables in the database and defines 26 interactions that can be accessed from the clients' Web browsers. The application contains a Java-based client that generates a session-oriented workload. RUBiS sessions have an average duration of 15 minutes and the average think time is five seconds. It defines two workload mixes: a browsing mix made up of only read-only interactions and a bidding mix that includes 15% read-write interactions.

6.2 PERFUME Components

1) Power Monitor: The average power consumption can be measured at the resource pool level or at the VM level by using a feature of VMware ESX 4.1. VMware gathers such data through its Intelligent Power Management Interface sensors. The power monitor program uses vSphere API to collect the power measurement data.
2) Performance Monitor: It uses a sensor program provided by RUBiS client for performance monitoring. We modify the sensor to measure the client-perceived percentile-based response time and throughput over a period of time. The number of requests finished during a control interval is the throughput.
3) FUMI Controller: It determines the control actions at every interval of 30 seconds. This control interval is chosen by considering the trade-off between noise in the sensor measurements and faster response of the controller. It invokes a quadratic programming solver, *quadprog*, in MATLAB to execute the control algorithm described in Section 5.
4) Resource Allocator: It uses vSphere API to impose CPU usage limits on the VMs. The vSphere module provides an interface to execute a method $ReconfigVM\_Task$ for this purpose.

In our testbed, the overhead of applying the wRLS technique on the fuzzy models and executing the control algorithm is less than half second. It is negligible compared to the control interval of 30 seconds.

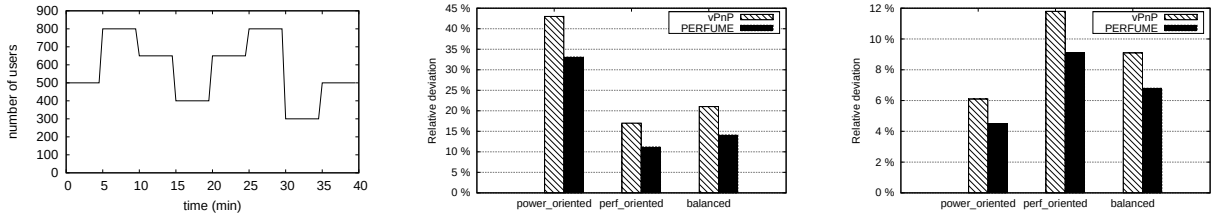
7 PERFORMANCE EVALUATION

7.1 Power and Performance Assurance

7.1.1 Flexible Tradeoffs

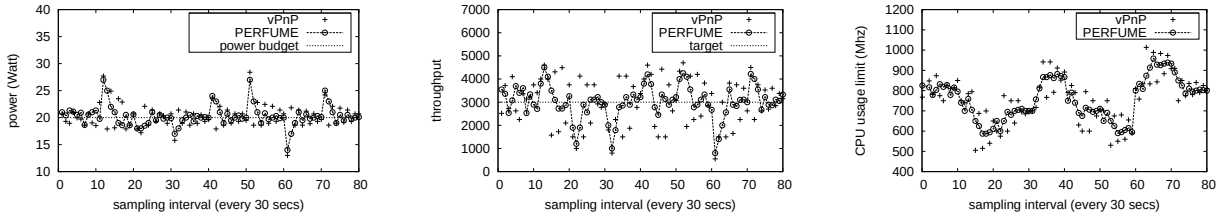
A key feature of PERFUME is its ability to assure joint power and performance guarantee with flexible tradeoffs while assuring control accuracy and system stability. The tradeoffs between inherently conflicting power and performance objectives can be specified by a datacenter administrator. The system stability is measured in terms of relative deviation of power and performance from their respective targets, as defined in vPnP [8]. The relative deviation is $|y(k) - r|/r$, where $y(k)$ is the measured application performance or the power consumption of the resource pool at time interval k and r is the corresponding target value. We experiment with power-preferred, performance-preferred and balanced control options under a highly dynamic workload [20]. Figure 6(a) shows the dynamic changes in the number of concurrent users. The workload prediction component is not used in this case, due to the difficulty in achieving sufficient prediction accuracy for randomly varying step change workload.

PERFUME achieves the specified tradeoffs by tuning the tracking error weights, P and Q, in the MIMO control objective defined by Eq. (13). Figure 6(b) compares the control accuracy of vPnP with PERFUME in assuring the throughput target for various tradeoffs between power and performance. Our results demonstrate that, compared to vPnP, PERFUME delivers average improvement of 30% in performance assurance in terms of relative deviation for various control options. We obtained similar results with the



(a) A highly dynamic workload. (b) Throughput. (c) Power consumption.

Fig. 6: Comparison of PERFUME and vPnP for control accuracy under a dynamic workload.



(a) Power consumption. (b) Throughput. (c) CPU allocation.

Fig. 7: Comparison of PERFUME and vPnP for power and performance assurance under a dynamic workload.

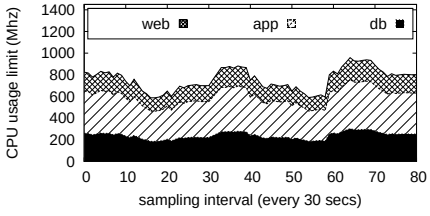


Fig. 8: Per-tier CPU allocation for the bidding workload mix .

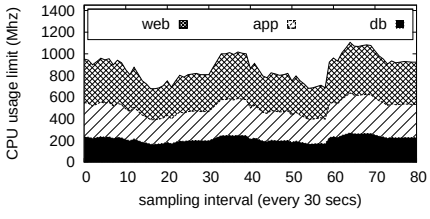


Fig. 9: Per-tier CPU allocation for the browsing workload mix.

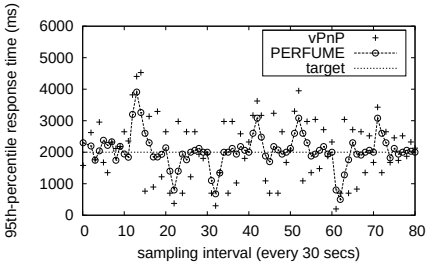


Fig. 10: The 95_{th}-percentile response time.

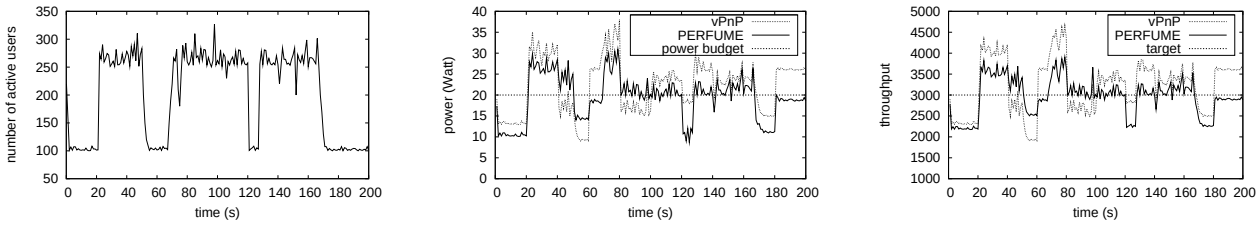
average improvement of 25% for relative deviation in power consumption with respect to its power budget, as shown in Figure 6(c). The control accuracy of the power-preferred option is the highest for power assurance but the lowest for throughput assurance. Whereas, the control accuracy of the performance-preferred option is the highest for throughput assurance and the lowest for power assurance. The balanced control option shows good control accuracy for both power and performance assurance.

7.1.2 System Stability

We now take a closer look at the system stability of PERFUME under the highly dynamic workload. We experiment with the power-performance balanced control option. Figures 7(a) and 7(b) illustrate that PERFUME offers more accurate assurance of power and performance targets compared to vPnP [8]. We show the results for only one of the hosted RUBiS applications. Similar results were obtained for the other. PERFUME is able to adapt itself and control both power consumption and throughput so that they eventually converge to the steady state. On the other hand, results show there are more significant oscillations in power and performance assurance due to the lack of control accuracy and system stability guarantee in vPnP. There is an improvement of 25% and 32% in relative deviation of power consumption and throughput respectively.

Figure 7(c) compares the total CPU usage limits allocated by vPnP and PERFUME at various sampling intervals. The total CPU usage limits is the sum of the CPU usage limits at the Web, application and database tiers. On average, PERFUME uses similar amount of CPU resources as vPnP. However, there is significantly less fluctuations in resource allocation. The CPU allocations are adjusted to track the power and performance targets. For instance, when an increase in the workload intensity causes the power consumption to exceed the power budget, CPU allocation is reduced to bring down the power consumption.

We investigate PERFUME's ability to capture multi-tier resource dependency for making control decisions by applying different workload mixes in this experiment. Figures 8 and 9 show the distribution of CPU allocation among the three tiers of the RUBiS application when



(a) A bursty workload. (b) Power consumption. (c) Throughput.

Fig. 11: Power and Performance assurance under a bursty workload generated by 1000 users.

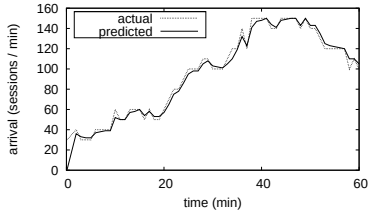


Fig. 12: Workload traffic trace and FUMI prediction.

bidding and browsing workload mixes are used respectively. We observe that the proportion of CPU allocations at the three tiers vary with different workload mixes. The relative deviation of power and performance from their respective targets are similar in both cases. It is due to the fact that PERFUME is able to satisfy the various resource demands at the three tiers of RUBiS application depending on the type of workload mix used.

7.1.3 Percentile-Based Response Time Guarantee

We now demonstrate the capability of PERFUME in accurately achieving the 95_{th}-percentile response time guarantee in the face of highly dynamic workload. Note that PERFUME is able to provide any percentile based response time guarantee. Compared with the mean-based performance metric, a percentile-based response time introduces much stronger nonlinearity in the system. Figure 10 shows that compared to vPnP [8], PERFUME delivers significantly improved control accuracy and performance assurance for highly non-linear percentile-based response times. For this experiment, we set the 95_{th}-percentile response time target of a RUBiS application as two seconds. We observe that compared with vPnP, there is the improvement of 40% in terms of relative deviation by PERFUME. This is mainly due to two reasons. First, PERFUME's FUMI model has better accuracy, even in case of highly non-linear percentile-based performance. Second, it provides more accurate control and system stability due to the FUMI control design.

7.1.4 Sytem Robustness under A Bursty Workload

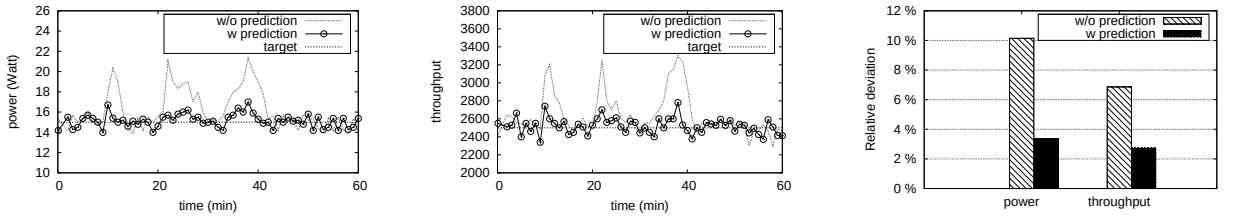
We evaluate the robustness of PERFUME under a bursty workload. We use an approach proposed in [29] to inject burstiness into the arrival process of RUBiS clients according to the index of dispersion. The dispersion index modulates the think times of users between submission of consecutive requests. We set the index of dispersion to 4000 and the maximum number of concurrent users to 1000. Figure 11 (a) shows the bursty workload in which the number of active users in a RUBiS application fluctuates over a period of 200 seconds.

Figures 11(b) and 11(c) illustrate that, compared to vPnP, PERUME is able to provide better assurance of average power consumption and throughput targets in the face of the bursty workload. We choose a sampling interval of 20 seconds for both approaches. A smaller sampling interval provides better responsiveness to workload fluctuations, but increases the sensitivity towards random noise. The variations in the average power consumption and throughput are mainly due to burstiness in the workload and the control actions (CPU allocations) taken at each sampling interval. The robustness of PERFUME under bursty workloads is attributed to the fact that its control actions are based on a more accurate model of the system and a sound control theoretic foundation. Moreover, PERFUME is more adaptive to variations in workload due to its fast online learning algorithm. We observe that compared with vPnP, there is the improvement of 32% and 44% in terms of relative deviation of power and throughput by PERFUME.

7.2 Impact of Proactive FUMI Control on Power and Performance Assurance

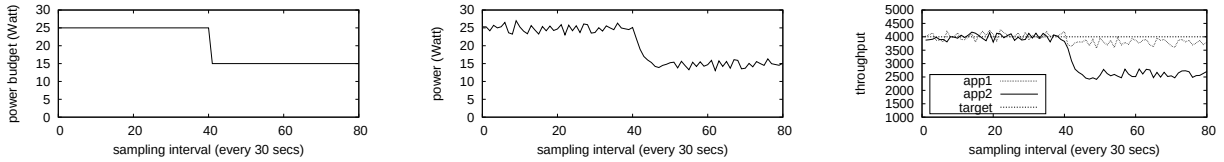
We demonstrate the benefit of integrating workload prediction with FUMI control. We modify the RUBiS client to generate workload based on the Web traces from the 1998 Soccer World Cup site [1]. These traces contain the number of arrivals per minute to this website over an 8-day period. As a case study, we choose the workload trace of a moderately busy day and compress the original 24-hour long trace to 1 hour, similar to the related work in [34]. Figure 12 shows the actual workload and the predictions made by Kalman filtering technique. Our proactive FUMI controller acquires workload predictions over a horizon of four steps and computes the CPU resource adjustments required to meet the power and performance goals. The power budget and the performance target are set to be 15 Watts and 2500 requests per sampling interval respectively.

Figures 13(a) and 13(b) compare the average power consumption and system throughput achieved by PERFUME with and without the integration of workload prediction. In case of a purely reactive FUMI control, we observe significant violations of the power and performance targets during the time intervals 10-13 min, 21-29 min and 34-41 min. It



(a) Power consumption. (b) System Throughput. (c) Improvement in relative deviation.

Fig. 13: Comparison of PERFUME with and without workload prediction of proactive FUMI control.



(a) Variation in power budget. (b) Power consumption. (c) Throughput.

Fig. 14: Service differentiation between two RUBiS applications for varying power budgets.

is due to the continuous increase in the workload intensity during those intervals as shown in Figure 12. Without the integration of workload prediction, FUMI control incorrectly estimates the future states of the system assuming that the workload intensity will not change. Furthermore, it requires a few control intervals to update the system model in response to the workload variations. As a result, the control actions in terms of CPU resource adjustments lag behind the continuously increasing workload.

On the other hand, the integration of workload prediction allows FUMI control to make proactive control decisions in anticipation of future changes in the workload intensity. Hence, it is able to avoid significant violations of power and performance in the face of dynamic workload variations. For instance, Figure 13(a) shows that the average power consumption starts decreasing at time 18 min due to proactive control action in anticipation of future workload change starting at time 21 min. As a result of this trend, the controller is able to keep the power consumption close to its target during the time interval 21-29 min in spite of the continuous increase in the workload. Figure 13(c) shows the improvement achieved by proactive FUMI control in terms of the relative deviation of power and performance.

7.3 Service Differentiation Provisioning

We evaluate the service differentiation capability of PERFUME in the face of a dynamic power budget as shown in Figure 14(a). In practice, the power budget might change due to thermal condition or temporary reductions in cooling or power delivery capacity.

In this experiment, two RUBiS applications, app1 and app2, are hosted on a shared resource pool. For service differentiation, the performance tracking error weight Q for app 1 is set to be larger than that of app 2. Hence, PERFUME gives a higher priority to app 1. As a case study, a power-preferred control policy is applied. Each application faces a workload of 1000 users. The throughput target is set to be 4000 requests per sampling interval.

Figures 14(b) and 14(c) show the average power consumption of the resource pool and the achieved throughput of the RUBiS applications. For the first 39 intervals, both applications are able to meet their performance goals while keeping the average power consumption below the specified power budget of 25 Watts. At interval 40, the power budget is changed to 15 Watts. This constrains the system in such a way that both applications can not maintain their current performance level without violating the power budget. PERFUME responds to this situation automatically and correctly re-distributes CPU resources so that the higher priority application, app1, still achieves an average throughput that is close to the target. On the other hand, the performance of lower priority application is degraded. As a result of this service differentiation, PERFUME is able to avoid power budget violations.

7.4 Effect of Control Parameter Tuning

TABLE 2: Impact of control penalty weight R on rise time (t_r) and percent overshoot (PO).

	R=0.02	R=0.04	R=0.06
t_r	90 sec	120 sec	300 sec
PO	13.7%	2.8%	1.1%

We now present the effect of tuning FUMI control parameters on the control performance. Figure 15 shows the performance of one RUBiS application under a workload of 1000 concurrent users, for different values of control penalty weight R. For R = 0.02, the controller adjusts the CPU resources more quickly and aggressively to meet the performance target. However, it results in large oscillations in performance. On the other hand, for R = 0.06 the controller avoids significant oscillations in performance. However, it becomes more sluggish and takes ten control intervals to meet the performance target. We quantify these observations by measuring two important properties of our

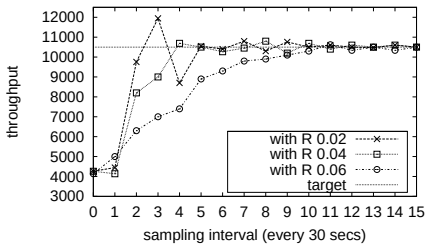


Fig. 15: Performance results of PERFUME with different values of control penalty weight R.

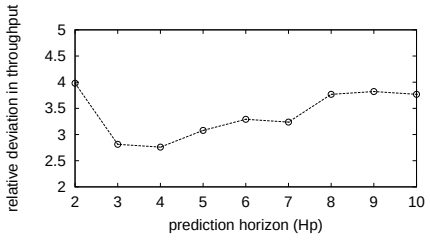


Fig. 16: FUMI control performance as a function of H_p.

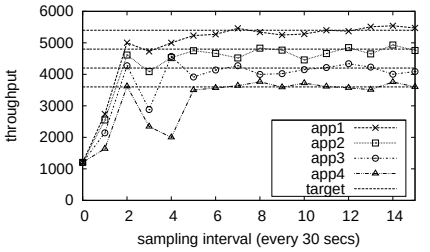


Fig. 17: Performance assurance by decentralized FUMI control.

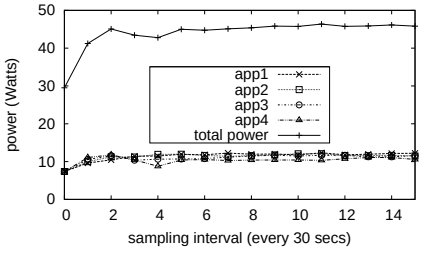


Fig. 18: Power control by decentralized FUMI control.

control system. First, we measure rise time (t_r), which is the time required for the throughput to reach its target value. Second, we measure percent overshoot (PO), which represents the maximum amount by which the throughput overshoots its target. It is desirable to minimize t_r for control responsiveness, and minimize PO for control accuracy and stability. Hypothetically, an ideal control system will have a rise time equivalent to one control interval, and zero percent overshoot. However in practice, these two properties often conflict with each other. Table 2 shows that the control penalty weight R = 0.04 balances this tradeoff.

Next, we study the impact of tuning the prediction horizon H_p on the control performance. In this experiment, the RUBiS application faces a dynamic workload shown in Figure 12. The control performance is measured in terms of the relative deviation of average throughput with respect to the target of 2500 requests per control interval. Figure 16 shows that the control performance initially improves with the increase in H_p. Intuitively, as the controller looks further ahead, it anticipates future workload demands and takes control actions accordingly at the current time step itself. However, control performance degrades when H_p is larger than four. It is due to the fact that the prediction accuracy decreases with the increase in H_p. Therefore, H_p must be chosen carefully, considering the trade-off between look-ahead performance and estimation errors.

7.5 Evaluation of Decentralized FUMI Control

We evaluate the effectiveness of the decentralized FUMI control in assuring application performance and controlling the power consumption of the virtualized server cluster. In this experiment, four RUBiS applications are hosted on a resource pool of three virtualized blade servers. Each application is controlled by a decentralized FUMI controller. Each application faces a workload of 700 concurrent users. The throughput targets are set to 5400, 4800, 4200, and 3600 requests per sampling interval for app1, app2, app3, and app4 respectively. The priority, $Q(a)$, for each application is set to be 0.25. According to the problem decomposition strategy, each application is assigned an equal fraction of the total power budget, which is 47 Watts. Figure 17 shows that each decentralized FUMI controller is able to meet the application performance target within six sampling intervals. As shown in Figure 18, the power consumption of each application increases in a controlled manner due to the CPU allocations made by each controller. As a result, the total power consumption of the virtualized server cluster stays close to the peak power budget.

8 Conclusion

Datacenters face significant multi-facet challenges in power and performance management for meeting SLAs, resource utilization efficiency and power savings. PERFUME provides a coordinated and self-adaptive power and performance control in a virtualized server cluster. As demonstrated by experimental results based on a testbed implementation, its main contributions are the precise and proactive control of power consumption of virtualized servers for power budgeting, average throughput and percentile-based response time guarantee of multi-tier applications, tradeoff flexibility between power and performance targets and service differentiation among co-located applications while assuring control accuracy and system stability in the face of highly dynamic and bursty workloads.

The main technical novelty of PERFUME system is due to the proposed fuzzy MIMO (FUMI) control technique, which integrate the strengths of fuzzy logic, MIMO control and artificial neural network. It is self-adaptive to dynamic workloads due to online learning of fuzzy model parameters using a computationally efficient wRLS method. This is complemented by the integration of future workload prediction for proactive control.

Our future work will integrate power-aware consolidation techniques with PERFUME and explore autonomous performance and power control for building energy-efficient datacenters.

Acknowledgement

This research was supported in part by NSF CAREER award CNS-0844983 and research grant CNS-1217979. A preliminary version of the paper appeared in [21]. The authors thank the anonymous reviewers for their valuable suggestions for revising the manuscript.

REFERENCES

[1] M. Arlitt and T. Jin. Workload characterization of the 1998 world cup web site. In *Technical report, HPL-99-35R, HP Labs.*, 1999.

[2] S. Chen, K. R. Joshi, M. A. Hiltunen, R. D. Schlichting, and W. H. Sanders. Cpu gradients: Performance-aware energy conservation in multitier systems. In *Proc. GREENCOMP*, 2010.

[3] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam. Managing server energy and operational costs in hosting centers. In *Proc. ACM SIGMETRICS*, pages 303–314, 2005.

[4] S. Chiu. Fuzzy model identification based on cluster estimation. *Journal of Intelligent & Fuzzy Systems*, 2(3), 1994.

[5] M. Elnozahy, M. Kistler, and R. Rajamony. Energy conservation policies for Web servers. In *Proc. USITS*, 2003.

[6] X. Fan, W. D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. *ACM SIGARCH*, 35(2):13–23, 2007.

[7] A. Gandhi, M. Harchol-Balter, R. Das, and C. Lefurgy. Optimal power allocation in server farms. In *Proc. ACM SIGMETRICS*, 2009.

[8] J. Gong and C.-Z. Xu. vpnp: Automated coordination of power and performance in virtualized datacenters. In *Proc. IEEE IWQoS*, 2010.

[9] S. Govindan, J. Choi, B. Urgaonkar, A. Sivasubramaniam, and A. Baldini. Statistical profiling-based techniques for effective power provisioning in data centers. In *Proc. EuroSys*, 2009.

[10] S. Govindan, J. Liu, A. Kansal, and A. Sivasubramaniam. Cuanta: quantifying effects of shared on-chip resource interference for consolidated virtual machines. In *Proc. ACM SOCC*, 2011.

[11] Y. Guo, P. Lama, and X. Zhou. Automated and agile server parameter tuning with learning and control. In *Proc. IEEE IPDPS*, 2012.

[12] Y. Guo and X. Zhou. Coordinated vm resizing and server tuning: Throughput, power efficiency and scalability. In *Proc. IEEE MASCOTS*, 2012.

[13] T. Horvath, T. Abdelzaher, K. Skadron, and X. Liu. Dynamic voltage scaling in multitier Web servers with end-to-end delay control. *IEEE Trans. on Computers*, 56(4):444–458, 2007.

[14] J.-S. Jang. Anfis: adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man and Cybernetics*, 23(3):665 – 685, 1993.

[15] C. Jiang, X. Xu, J. Wan, J. Zhang, X. You, and R. Yu. Power aware job scheduling with qos guarantees based on feedback control. In *Proc. IEEE IWQoS*, 2010.

[16] G. Jung, M. A. Hiltunen, K. R. Joshi, R. D. Schlichting, and C. Pu. Mistral: Dynamically managing power, performance, and adaptation cost in cloud infrastructures. In *Proc. IEEE ICDCS*, 2010.

[17] Y. Koh, R. Knauerhase, P. Brett, M. Bowman, W. Zhihua, and C. Pu. An analysis of performance interference effects in virtual environments. In *Proc. IEEE ISPASS*, 2007.

[18] S. Kumar, V. Talwar, V. Kumar, P. Ranganathan, and K. Schwan. vmanage: Loosely coupled platform and virtualization management in data centers. In *Proc. IEEE ICAC*, 2009.

[19] D. Kusic, J. Kephart, J. Hanson, N. Kandasamy, and G. Jiang. Power and performance management of virtualized computing environments via lookahead control. In *Proc. IEEE ICAC*, 2008.

[20] P. Lama and X. Zhou. Autonomic provisioning with self-adaptive neural fuzzy control for end-to-end delay guarantee. In *Proc. IEEE/ACM MASCOTS*, pages 151–160, 2010.

[21] P. Lama and X. Zhou. PERFUME: Power and performance guarantee with fuzzy mimo control in virtualized servers. In *Proc. IEEE IWQoS*, pages 345–354, 2011.

[22] P. Lama and X. Zhou. Efficient server provisioning with control for end-to-end delay guarantee on multi-tier clusters. *IEEE Transactions on Parallel and Distributed Systems, 19 pages*, 23(1), 2012.

[23] P. Lama and X. Zhou. Ninepin: Non-invasive and energy efficient performance isolation in virtualized servers. In *Proc. IEEE/IFIP DSN*, 2012.

[24] K. Le, R. Bianchiniy, M. Martonosiz, and T. D. Nguyeny. Cost- and energy-aware load distribution across data centers. In *Proc. HotPower*, 2009.

[25] C. Lefurgy, X. Wang, and M. Ware. Server-level power control. In *Proc. IEEE ICAC*, 2007.

[26] J. C. B. Leite, D. M. Kusic, D. Mossé, and L. Bertini. Stochastic approximation control of power and tardiness in a three-tier Web-hosting cluster. In *Proc. IEEE ICAC*, 2010.

[27] H. Lim, A. Kansal, and J. Liu. Power budgeting for virtualized data centers. In *Proc. USENIX Annual Technical Conference*, 2011.

[28] J. Mars, L. Tang, R. Hundt, K. Skadron, and M. L. Soffa. Bubble-up: increasing utilization in modern warehouse scale computers via sensible co-locations. In *Proc. IEEE/ACM MICRO*, 2011.

[29] N. Mi, G. Casale, L. Cherkasova, and E. Smirni. Injecting realistic burstiness to a traditional client-server benchmark. In *Proc. IEEE ICAC*, 2009.

[30] R. Nathuji, K. Schwan, A. Somani, and Y. Joshi. Vpm tokens: virtual machine-aware power budgeting in datacenters. *Cluster Computing*, 12(2):189–203, 2009.

[31] P. Padala, K.-Y. Hou, K. G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, and A. Merchant. Automated control of multiple virtualized resources. In *Proc. EuroSys*, pages 13–26, 2009.

[32] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu. No power struggles: Coordinated multi-level power management for the data center. In *Proc. ASPLOS*, 2008.

[33] R. Singh, U. Sharma, E. Cecchet, and P. Shenoy. Autonomic mix-aware provisioning for non-stationary data center workloads. In *Proc. IEEE ICAC*, pages 21–30, 2010.

[34] B. Urgaonkar, P. Shenoy, A. Chandra, P. Goyal, and T. Wood. Agile dynamic provisioning of multi-tier Internet applications. *ACM Trans. on Autonomous and Adaptive Systems*, 3(1):1–39, 2008.

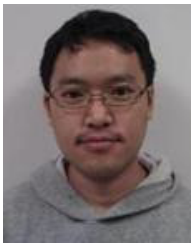
[35] A. Verma, P. Ahuja, and A. Neogi. pmapper: power and migration cost aware application placement in virtualized systems. In *Proc. ACM/IFIP/USENIX Middleware*, 2008.

[36] A. Verma, P. De, V. Mann, T. Nayak, A. Purohit, G. Dasgupta, and R. Kothari. Brownmap: enforcing power budget in shared data centers. In *Proc. ACM/IFIP/USENIX Middleware*, 2010.

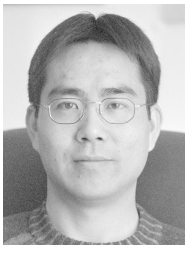
[37] X. Wang, M. Chen, and X. Fu. MIMO power control for high-density servers in an enclosure. *IEEE Trans. on Parallel and Distributed Systems*, 21(10):1412–1426, 2010.

[38] X. Wang and Y. Wang. Co-con: Coordinated control of power and application performance for virtualized server clusters. In *Proc. IEEE IWQoS*, 2009.

[39] B. J. Watson, M. Marwah, D. Gmach, Y. Chen, M. Arlitt, and Z. Wang. Probabilistic performance modeling of virtualized resource allocation. In *Proc. IEEE ICAC*, 2010.



Palden Lama received the BTech degree in electronics and communication engineering from the Indian Institute of Technology, in 2003. He received his PhD degree in Computer Science from the University of Colorado, Colorado Springs in 2013. Currently, he is an Assistant Professor in the Department of Computer Science at the University of Texas, San Antonio. His research interests include the areas of Cloud computing, sustainable computing, autonomic resource and power management, and big data processing in the Cloud.



Xiaobo Zhou received the BS, MS, and PhD degrees in Computer Science from Nanjing University, in 1994, 1997, and 2000, respectively. Currently he is a Professor and the Chairperson of the Department of Computer Science, University of Colorado, Colorado Springs. His research lies broadly in computer network systems, more specifically, autonomic and sustainable computing in datacenters, Cloud computing, server virtualization, scalable Internet services and architectures, and computer network security. His research was supported in part by the US NSF and Air Force Research Lab. He was a recipient of the NSF CAREER AWARD in 2009. He is a senior member of the IEEE.