Cluster Comput (2015) 18:847–863
DOI 10.1007/s10586-015-0436-x

A MISO model for power consumption in virtualized servers

Fawaz Al-Hazemi · Yuyang Peng · Chan-Hyun Youn

Received: 14 October 2014 / Revised: 8 December 2014 / Accepted: 22 January 2015 / Published online: 3 February 2015

Abstract Energy efficiency is always a concern in hosting servers. When any new development is added to a host server, the power consumption of the host server must be theoretically and empirically re-evaluated. Because of the ongoing development trends in computing systems at the hardware, software and middleware levels, deriving a direct mathematical model for quantifying the power consumption of a host server is difficult. Therefore, a system identification is used to construct the power consumption model for virtualized hosting servers. To date, three types of system identifications have been used in the literature for defining the power consumption model: the first-principles, the black-box and the gray-box identification approaches. To the best of our knowledge, the majority of these approaches are apparently used to model the power consumption in a single-input single-output (SISO) system model, in which a hardware component is reconfigured to meet the power budget target. In this paper, to accommodate the ongoing development trends in computing systems, we propose a multi-input single-output (MISO) model for modeling the power consumption of virtualized hosting servers. We use the black-box system identification method, and we utilize the Auto-Regressive eXogenous (ARX) mathematical model to construct the MISO power model. We compare our MISO power model with the SISO power model that is used in existing state-of-the-art works. Empirically, our MISO power model exhibits higher accuracy than the existing SISO power model in predicting power consumption. Using our model, we can achieve approximately 98 % accuracy in predicting the power consumption of virtualized hosting servers.

Keywords Control theory · Energy-aware system · Modeling techniques · Power management · System identification

F. Al-Hazemi (✉) · Y. Peng · C.-H. Youn
Department of Electrical Engineering, KAIST, Bldg. E3-2 , E6-3211 291 Daehak-ro, Yuseong-gu, Daejeon 305-701, South Korea
e-mail: fawaz@kaist.ac.kr; fawazhazemi@ieee.org

Y. Peng
e-mail: yuyangpeng@kaist.ac.kr

C.-H. Youn
e-mail: chyoun@kaist.ac.kr

1 Introduction

The variety of available tuning configuration settings in virtualized hosting servers has enabled a broad range of modeling and design aspects, such as power and performance modeling. In a virtualized hosting server (hereafter called a host server), the server has the capability to dynamically (on-the-fly) reconfigure its CPU frequency [26], memory frequency [15,33], network bandwidth [11], and storage disks [20,21]. These configurations are called (in this paper) a *physical* configuration capability. On the other hand, virtualization technology has enabled host servers to host virtual machines (hereafter called guest servers) and to share their physical hardware with these guest servers. Examples of hardware sharing between host and guest servers include a number of CPU cores being assigned to the guest server (called vCPU), the CPU time share (or the percentage of time that a guest server is allowed to use the physical hardware), the amount of memory, and network bandwidth, among others. These configurations are called (in this paper) *virtual* configurations.

These configuration capabilities (*physical* and *virtual*) are primarily used as controller inputs for different purposes in virtualized host servers. For example, they are used in performance tuning [1,14,26], thermal management [8,9,32,33] and revenue maximization [36,60]. Apparently,

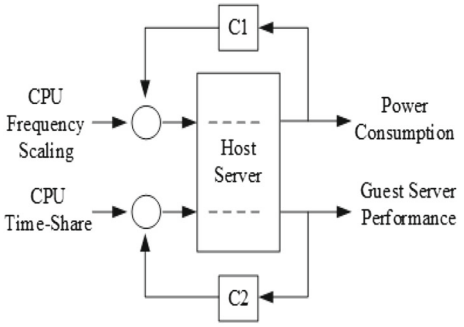


Fig. 1 A host server with two feedback controllers with two different purposes: controller C1 manages the power consumption of the host server, and controller C2 manages the performances of the guest servers

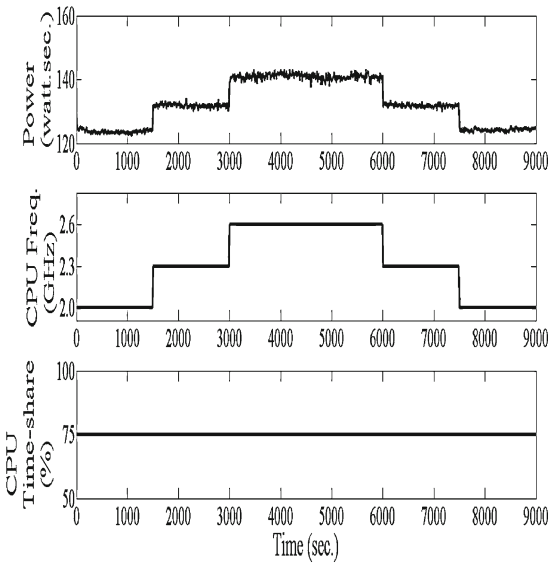
power consumption is implicitly considered within the scope of these purposes, and the *physical* configurations are used only to control the power consumption. For example, Fig. 1 illustrates how the power consumption in a host server is commonly modeled as a single-input single-output (SISO) system to control the power budget. Other configuration parameters are used for other controlling purposes. For example, CPU frequency scaling is used to control the power consumption of a cluster of host servers, as in [7,13,29,38,50]. However, we argue that there are other configurations in virtualized hosting servers that have an impact on the power consumed by these servers. For example, the performance model of a system is controlled using the server's CPU time share, and the power consumption is controlled using the CPU frequency scale in two SISO models. These two models are periodically checked in a hybrid hierarchy model [42,51,53]. The power model is either located in the inner [42] or outer [53] part of the overall control design of a system. The impact of the location of the SISO power model in the control design is shown in Fig. 2. For example, if the power model is located in the inner part of the control design of a system, then the power consumption of a host server would be as shown in Fig. 2a. Conversely, if the power model is located in the outer side of the control design, then the power consumption would be impacted by other controllers (e.g., the performance controller), as shown in Fig. 2b.

In the literature, the power consumption of host servers is investigated using the SISO model. In particular, the changes in the CPU frequency (with the help of dynamic voltage and frequency scaling) in a host server will reflect the changes in power consumption in that host server. Nevertheless, in virtualized environments, a host server will share its CPU timing among guest servers (VMs) that reside in it. This CPU time share is managed by a virtualized manager (such as xen [57] or kvm [28]). For example, in xen, the capability (cap) is the interface for controlling and managing the CPU time share. A cap of 100 for a guest server means that if the guest server is configured with one CPU core, then it has 100 % of the time of one CPU core; if the guest server is configured with two CPU cores, then it has 50 % of the time of the two CPU cores. Similarly, a cap of 150 for a guest server means that it has 75 % of the time of two CPU cores or 50 % of the time of three CPU cores.

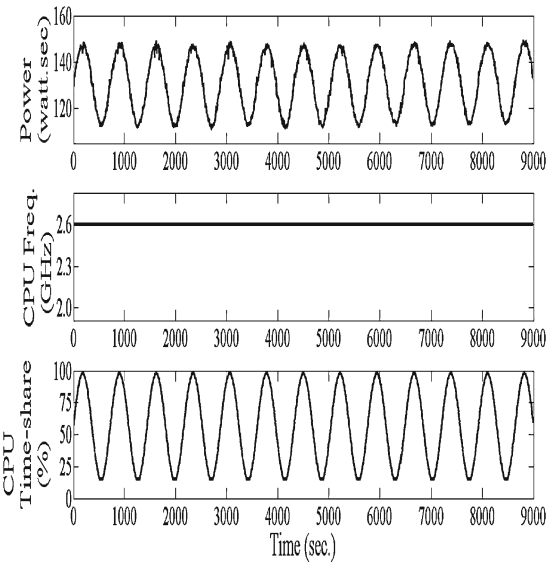
The primary objective in this paper is to estimate a multiple-input single-output (MISO) power consumption model for host servers that can reduce the power consumption settling time. Rather than using multiple SISO models running in a hybrid hierarchy, in which some of these SISO models will interact with power changes, a single MISO model will promptly respond to any required changes to the power settings. This power model will be vital if renewable energy sources (e.g., wind turbines or solar photovoltaic panels) are used to supply data centers, such as in a *parasol* data center [19]. The power supplied in this case comes from an intermittent supplier that has a high probability of power leakages or power outages. Therefore, any host server in a green data center must have a model that helps to promptly adapt to such changes in supplied power.

In this paper, we investigate the modeling of power consumption in host servers and the precise impact of multiple and concurrent configuration changes on power consumption. We model the power consumption in host servers using the MISO model. To the best of our knowledge, this study is a pioneering step that changes the modeling of power consumption from coordinating multiple SISO models to a single MISO power consumption model in virtualized hosting servers. The key contributions of this paper are as follows:

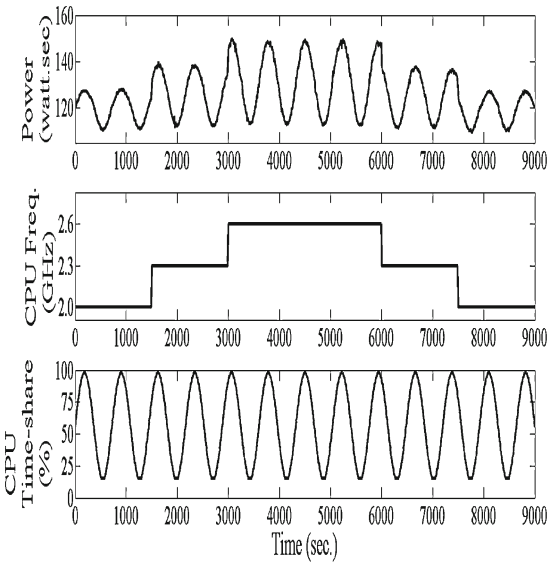
- We identify the factors that significantly contribute to the power consumption of a server. We define the contributors to power consumption in host servers as *physical* configurations, such as CPU frequency, and as *virtual* configurations, such as the CPU time share allocated for guest servers (virtual machine).
- We use the *black-box* system identification approach to estimate the power consumption model. We essentially estimate two SISO models (one model for the physical configuration and one model for the virtual configuration) and one MISO model that combines both physical and virtual configurations. We compare the estimated models (SISO and MISO models) in terms of how well they reproduce the actual measured power consumption of host servers.
- We further investigate the model complexity, in particular, the model orders and input delays. In addition, we determine the model sampling time that is adequate for system control.
- Empirically, we show that the MISO model (with its complexity) is superior to the SISO models used in existing works. We believe that the MISO model could simplify the hybrid hierarchy control design in some of the existing works, such as [27,51,53].



(a) CPU frequency changes only



(b) CPU time share changes only



(c) Both CPU frequency and time share changes

Fig. 2 Scenarios for system configurations and their effects on the power consumption of the host server. The *top* graph is the power consumed by the host server (W s), and the two *bottom* graphs are the CPU frequency (GHz) and CPU time share (%) varied over a period of time

– We review the characteristics of our MISO power model in terms of stability and oscillatory.

The remainder of this paper is organized as follows. In Sect. 2, we discuss the method that we use to model the power consumption in detail. In Sect. 4, we discuss the results of our work. The evaluations of our proposed model (including the limitations and future works) are presented in the discussion section, Sect. 5. Then, reviews on related works are presented in Sect. 6. Finally, conclusions are presented in Sect. 7.

2 Methods

Constructing a model to facilitate control over power consumption in host servers is one of the fundamental requirements for constructing an automated system. According to control theory, the construction of an automated system first requires studying and modeling the system, followed by investigating the possible control methods for automation. In this paper, we focus on studying the modeling of power consumption in virtualized hosting servers. Studying the system

involves (1) defining the inputs and outputs of the system and (2) identifying the system model. For the latter step (identifying the system), the methodology used to estimate the model is of considerable importance because it helps to identify the model complexity (model order) and to define the adequate sampling time for the proposed model (in this paper, we aim to model power consumption).

2.1 System inputs and outputs

Any system has inputs and outputs, which are the core elements that define the complexity of the system. For instance, a system with a single input and single output (SISO) is simply comparable to multi-input multi-output (MIMO) systems. Defining the relationship or the influence of the inputs on the behavior of the system outputs is essential for accurate modeling. In host servers, the inputs and outputs of a system are the system configurations (inputs) and system performances (outputs). System configurations (inputs) in host servers have different forms, such as physical and virtual configurations. The physical configuration of the hardware components inside host servers is an example of a physical configuration. CPU and memory frequency scaling (using dynamic voltage and frequency scaling DVFS) [26] are examples of this physical configuration form. On the other hand, the configuration of the guest servers (virtual machine), such as the allocated CPU cores and amount of memory for that guest server, are virtual configurations. The priority and the time sharing for using the physical components of the host servers are also virtual configurations.

In this paper, to identify the power model, we consider two inputs or system configurations (inputs): (1) the changes in CPU frequency (denoted Δf) of the host server (as a *physical* configuration) and (2) the changes in the guest server's time share (denoted Δc) to use the physical CPU of the host server (as a *virtual* configuration). However, these two configurations are not the only configurations that affect power consumption. The configuration and the allocation of memory, storage and network bandwidth for the guest server also affect power consumption, but we selected the CPU frequency scale and CPU time share because they are strongly correlated to each other with respect to both performance tuning and power consumption.

System performances (outputs) in host servers could include availability, responsiveness, efficiency and power budgeting. With recent calls to reduce the power consumption in information and communication technology (ICT), we consider the power consumption as the system output. The aforementioned system configurations (inputs) have an impact on the power consumption behavior. Figure 2a and b show the effects of changes in the CPU frequency and time share, respectively. In fact, the combination of the two system configuration changes has effects with different patterns on power consumption; see Fig. 2c. In this paper, we focused our modulation on the power consumption.

A virtualized CPU has predefined values for the frequency levels; these levels are defined by the manufacturer's vendors to operate the CPU processors according to different power budget plans. To the best of our knowledge, there is no way to operate the CPU processor on a frequency level that does not exist among these frequency levels. However, Lefurgy et al. [31] proposed a power-capping algorithm to modulate any middle CPU frequency value by supplying a series of the manufactured frequency levels at a tiny time scale (a fraction of the sampling time Ts), and the modulator was developed in [51,53]. In addition, Raghavendra et al. [42] suggested locating the CPU frequency controller to the most inner level to control the power consumed by promptly controlling the CPU frequency. In this paper, we did not follow this algorithm to produce a wide range of CPU frequencies; rather, we used the available manufactured frequency levels to identify the system. The reason for using the available manufactured frequency is that power capping adds an additional microcontroller to modulate the CPU frequency, and we must consider its performance impact on identifying the MISO model.

The configuration of CPU time share follows a ratio-style technique, which allocates a percentage of the CPU time to a guest server. For example, if a host server has a dual-core processor, then this host server has 200 % CPU time. Furthermore, if a guest server (residing on that host server) is going to work on half of a CPU core or on one full CPU core, then the CPU time share for that guest server would be 50 or 100 %, respectively.

2.2 System identification approach

The major contribution of this paper is the system identification; therefore, this subsection focuses on explaining the estimation of the mathematical model from the system identification. To construct a mathematical model for computing systems, we have to follow either a *first-principles* or a *black-box* identification approach. When the computing system is described through basic mathematical equations, the construction of the system model is called a *first-principles* identification approach. For example, starting from basic queuing theory equations to identify the performance model in a computing system is a *first-principles* identification approach. In contrast, supplying a series of control input signals to the computing system and collecting the outputs followed by analyzing and correlating the changes in the control inputs and outputs to estimate a system model is called a *black-box* identification approach. Both approaches have been used to model computing systems, and they have been demonstrated to be applicable for modeling and further controlling diverse aspects in computing systems.

For example, the authors in [44,47] suggested using the *first-principles* approach to identify the performance model. However, the authors in [6,27,42,53] suggested using the *black-box* approach to identify the power consumption model as a SISO model. Furthermore, there is another approach, called the *gray-box* approach, that combines the two aforementioned approaches. The *gray-box* approach depends on previously defined mathematical equations and uses the *black-box* approach to develop better estimated system models. Chen et al. [11–14] successfully used the *gray-box* approach to model the performance of host servers. Chen et al. relied on the queuing theory equations to define the response time in terms of processing demand and utilization. Then, they defined a basis function to correlate the server's CPU frequency with both the workload and utilization of the server. Subsequently, they used a training signal to practically estimate the impact of CPU frequency on the response time using the *black-box* approach. However, regarding the power consumed by a server, there are no previously defined and stable mathematical equations that describe a power consumption model for servers. Therefore, the *black-box* approach is preferable in this case.

In this paper, we used the *black-box* approach to identify the power consumption model for host servers as a (MISO model. The *physical* and *virtual* configurations, i.e., the CPU frequency scaling and CPU time share as discussed in the previous Sect. 2.1, are the control input signals used in the *black-box* system identification approach. Thus, we compare our proposed model with those that followed the *black-box* system identification approach [6,27,42,53]. Other proposed models that followed the *first-principles* or the *gray-box* system identification approach are not considered in this paper because there are no stable or accurate mathematical equations available for defining the power consumed in servers.

2.2.1 Model type

In system identification [35], there are several mathematical models available for defining a mathematical equation, such as Auto-Regressive (AR), Auto-Regressive Moving-Average (ARMAX), Output-Error (OE) and Box-Jenkins (BJ). For example, the OE model was used for thermal modeling in multi-core processors [10]. The ARX model was used to model the web server response time [53] and the CPU and memory utilization in a web server [16] [17]. We selected the ARX model as the most appropriate model because (1) evaluating its difference equations is simple and (2) this model is able to estimate MISO systems. The simplicity of the ARX model is primarily due to its difference equations, which are easy to interpret and control. The ARX model follows the difference equations as in Eq. 1,

$$y(t+1) = \sum_{i=0}^{n_a} a_i \cdot y(t-i) + \sum_{i=0}^{n_b} b_i \cdot u(t-i), \tag{1}$$

where $y(t+1)$ is the estimated next output, $y(t)$, $y(t-1), \ldots, y(t-n_a)$ are the past n_a outputs, and $u(t)$, $u(t-1), \ldots, u(t-n_b)$ are the past n_b control inputs. The coefficients or system parameters $a_0, \ldots, a_i$ and $b_0, \ldots, b_i$ are the model parameters that are estimated using the *Least Squares Regression* method [35] (a computer software such as MATLAB [34] can estimate these parameters). The details of how to estimate these parameters are available in control theory textbooks, such as [35]; however, we describe the estimation process to clarify the estimation of these parameters. To generate an ARX equation as in Eq. 1, we first collect real data regarding the system that is being modeled, which in this case is power modeling for a server. The data are sorted in a time series, which include the input configuration ($u(t)$) and the measured outputs ($y(t)$) at time t. Next, we attempt to predict the future output (e.g., $\hat{y}(t+1)$) from the previous known data, i.e., $y(t) \ldots y(0)$ and $u(t) \ldots u(0)$, as mentioned in Eq. 1. Then, we examine the prediction error or the residual $e(t+1)$ of the $(t+1)$ output between the measured output with the ARX prediction as follows:

$$e(t+1) = y(t+1) - \hat{y}(t+1). \tag{2}$$

Using the *Least Squares* method, we want to minimize the squared errors by tuning the parameters in the ARX Eq. 1, that is, $a_0, \ldots, a_i$ and $b_0, \ldots, b_i$. Mathematically, we want to minimize the following Eq. 3

$$f(a_0, \ldots, a_n, b_0, \ldots, b_m) = \sum_{t=0}^{K} e^2(t+1)$$
$$= \sum_{t=0}^{K} [y(t+1) - \hat{y}(t+1)]^2 \tag{3}$$

Precisely,

$$min \sum_{t=0}^{K} [y(t+1) - \sum_{i=0}^{n_a} a_i \cdot y(t-i) - \sum_{i=0}^{n_b} b_i \cdot u(t-i)]^2, \tag{4}$$

where K is the size of the time series in the experiment. The values of the original ARX model (Eq. 1) can be determined by taking the partial derivatives of the above Eq. 4 with respect to each corresponding parameter and setting them to zero, i.e.,

$$\frac{\partial}{\partial a_0}, \ldots, \frac{\partial}{\partial a_{n_a}}, \frac{\partial}{\partial b_0}, \ldots, \frac{\partial}{\partial b_{n_b}}. \tag{5}$$

The ARX model presented in the system model Eq. 1 is a SISO system model, that is, it contains a single input u and

a single output y. If the system requires a certain number (e.g., N) of previous output states, in particular, $y(t-i)$, and a certain number (e.g., M) of previously supplied control inputs, then the complexity of the ARX model would be represented as an Nth-order system model with an Mth-order input delay. The complexity of the order is proportional to the model order, i.e., the Nth order. In our proposed model (MISO), we empirically observed that the Mth-order input delay has a very clear impact on the system estimation and validation.

2.2.2 SISO model (physical configuration)

The power consumption is modeled using the SISO model, in which the CPU frequency is considered to be a tuning controller. The changes in power consumption ($\Delta p(t+1)$ and $\Delta p(t-i)$) are the system outputs ($y(t+1)$ and $y(t-i)$ as in Eq. 1), and the changes in CPU frequency ($\Delta f(t-i)$) are the control input ($u(t-i)$ as in Eq. 1). Because the controls of computing systems are dynamic, the changes in both power and CPU frequency were calculated by 1) setting operating points for both power ($\tilde{p}$) and CPU frequency ($\tilde{f}$) and by 2) evaluating the differences between the measured and operating values and using them in estimating the ARX model, i.e., $\Delta p = \tilde{p} - p$ and $\Delta f = \tilde{f} - f$. Consequently, the ARX difference equation used is as follows 6:

$$\Delta p(t+1) = \sum_{i=0}^{n_a} a_i \cdot \Delta p(t-i) + \sum_{i=0}^{n_b} b_i \cdot \Delta f(t-i) \qquad (6)$$

2.2.3 SISO model (virtual configuration)

In the literature, the CPU time share is commonly used to control the performance of the guest server (virtual machine) [6,44,47]. Additionally, because we have previously defined that CPU time share is a *virtual* configuration, we empirically demonstrate the impact of this type of configuration on power consumption, as shown in Fig. 2b. Although *virtual* configurations, namely, the CPU time share (*xen cap* or *kvm cpulimit*), are used to control the guest (virtual) server's performance, we treat this input control configuration as a power controller as the CPU frequency configuration. Thus, we applied the same methodology of the SISO power consumption model with CPU frequency control input ($\Delta f(t-i)$) to obtain another SISO model with CPU time share control input ($\Delta c(t-i)$). We set the operating point for CPU time share ($\tilde{c}$) and evaluated the differences between the measured (c) and operating value ($\tilde{c}$) and used them in estimating the ARX model as in previous SISO models, i.e., $\Delta c = \tilde{c} - c$. The obtained ARX model is represented as the following Eq. 7.

$$\Delta p(t+1) = \sum_{i=0}^{n_a} a_i \cdot \Delta p(t-i) + \sum_{i=0}^{n_b} b_i \cdot \Delta c(t-i) \qquad (7)$$

2.2.4 MISO model

Intuitively, each control input has an impact on the power consumption behavior; therefore, the power consumption could be controlled by setting appropriate gains to both control inputs (configurations). In this paper, we propose a power model with two concurrent control inputs, i.e., the *physical* and *virtual* configurations described in Sect. 2.1. The control inputs are the rate of change in CPU frequency and the rate of change in CPU time share, and they are denoted as (Δf) and (Δc), respectively. Then, we followed a similar procedure in *system identification* by setting an operating point for all model participants (power, CPU frequency and CPU time share), and then we evaluated the differences between the measured and operating values and used them in the ARX model estimation. The expected resulting ARX model is represented by the following Eq. 8.

$$\Delta p(t+1) = \sum_{i=0}^{n_a} a_i \cdot \Delta p(t-i) + \sum_{i=0}^{n_{bf}} b_{fi} \cdot \Delta f(t-i) + \sum_{i=0}^{n_{bc}} b_{ci} \cdot \Delta c(t-i), \qquad (8)$$

where we have altered the notation for the control input coefficients (b_i) as follows. In Eq. (8), b_{fi} are the coefficients for the CPU frequency configuration ($\Delta f(t-i)$), and b_{ci} are the coefficients for the CPU time share configuration ($\Delta c(t-i)$).

2.3 Model order

The complexity of the system model is vital in control design. The complexity of the model is determined based on various aspects, and one of those aspects is the model order. In this paper, we focus on identifying the model order that could best capture the real power consumption behavior in host servers. The model order is computed according to the order of the output (or the number n_a of past outputs) in the ARX model presented in Eq. 1.

In addition, during our experiment, we observed that the order of the input delay has an observable impact on the accuracy of the model estimation. Therefore, we include in our experiment the variation in input delay (or the number n_b of past control inputs) for both *physical* and *virtual* configuration types as they appeared in the ARX models (Eqs. 6, 7 and 8).

2.4 MISO model properties

Any estimated model should be capable of predicting the dynamic behavior of a system, and our MISO model should

have this prediction capability. Let us recall Eq. 8 and consider the 1st-order model and the 2nd-order input delay as the following Eq. 9

$$\begin{aligned}\Delta p(k) = a \cdot \Delta p(k-1) + b_{f1} \cdot \Delta f(k-1) \\ + b_{f2} \cdot \Delta f(k-2) + b_{c1} \cdot \Delta c(k-1) \\ + b_{c2} \cdot \Delta c(k-2). \end{aligned} \tag{9}$$

The solution for this system Eq. 9 is the following Eq. 10, and the details of the derivation are explained in Appendix 1.

$$\begin{aligned}\Delta p(k) = a^{k-1} \cdot \Delta p(1) + \sum_{i=1}^{k-1} a^{k-1-i} \cdot b_{f1} \cdot \Delta f(i) \\ + \sum_{i=1}^{k-1} a^{k-1-i} \cdot b_{f2} \cdot \Delta f(i-1) + \sum_{i=1}^{k-1} a^{k-1-i} \cdot b_{c1} \cdot \Delta c(i) \\ + \sum_{i=1}^{k-1} a^{k-1-i} \cdot b_{c2} \cdot \Delta c(i-1), \end{aligned} \tag{10}$$

where k is greater than 1 ($k > 1$), and a, b_{f1}, b_{f2}, b_{c1}, and b_{c2} are estimated parameters. From this solution, it is clear that parameter a has a considerable impact on determining the power consumption of a host server. This a parameter is the *pole* of the model, as explained earlier. From this *pole* a, we can determine several properties of our MISO power model, namely, the stability, responsiveness, and oscillatory of the system [24].

Stability: The system model is stable if $|a| < 1$. If the *pole* (a) of the MISO model is greater than one and k approaches infinity, then the MISO model output ($\Delta p(k)$) becomes larger without a limiting bound.

Oscillatory: The system model is oscillatory if a is negative, that is , if $a < 0$, then the MISO model output (Δp) will be oscillatory. This is due to the term a^k. If k is odd, then the output will be negative, and if k is even, then the output will be positive.

2.5 Sample time selection

Defining the sampling time is important for the control design because it affects the performance of the controller that is used to control the system. The sampling time determines the length of time of the averaged system outputs. A short sampling time enables the controller to quickly react to any changes in the system; however, it increases the measurement overhead of the controller. In contrast, a long sampling time will avoid random fluctuations in the system by averaging out the system's stochasticity; however, it will slow the response of the system's controller. This is a critical task when we need to monitor many variables, such as in the case of [18].

To select the best sampling time, we estimate the ARX model of power consumption over different lengths of sampling time and monitor the changes in the model's *pole*. The *pole* of the system, particularly a_i in the ARX model (Eq. 8), is changing due to the changes in the sampling time period. Because the ARX models estimated in this paper are *discrete-time* models, we have to convert the models to *continuous-time* models and observe the behavior of the *pole*. This conversion is important because the model's *pole* is more likely constant in *continuous-time* models [35]. In this paper, we focus our sampling time selection to our proposed MISO; therefore, we run our experiment over 9000 s, and we collect power consumption outputs over different sampling times (Ts): 2, 3, 5, 10, 15, 20, and 30 s. Then, we estimate seven ARX MISO models to represent different sampling time models. The *poles* of the ARX MISO models are important. Hence, the estimated ARX MISO models are *discrete-time* models, and we convert them into *continuous-time* models. Next, we plot the *pole* of each model and find the points where the *pole* becomes fairly constant.

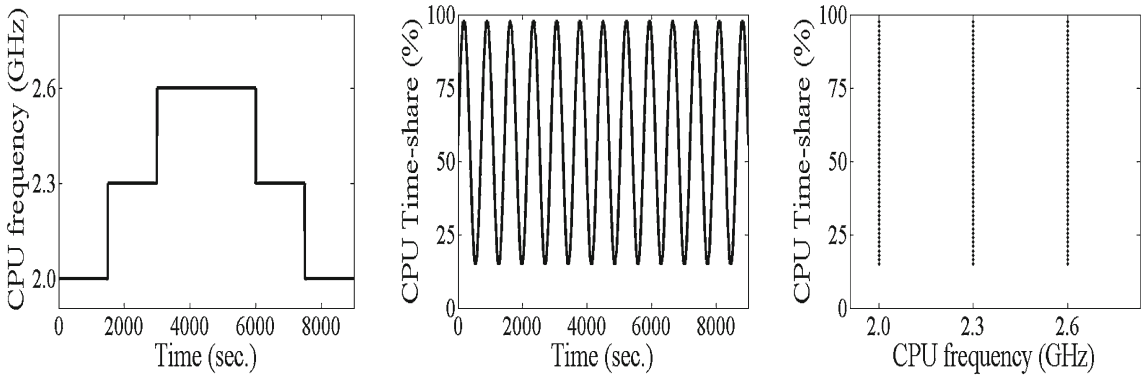
3 Experiment

3.1 Testbed

We installed xen version 4.1 [57] on a physical machine (host server) with the following hardware configuration: the processor is an Intel Core 2 Duo E6750@2.66 GHz processor with two cores; the processor supports three levels of CPU frequencies, namely, 2.66, 2.33 and 2 GHz; the host server has 4 GB of RAM; it has 80 GB of hard disk space; and it has a gigabyte Ethernet card. We configured three VMs on top of the server. Each VM has 2 cores (as the server has 2 physical cores), 256 MB of virtual RAM, and a virtual network connection. The CPU time share (or CPU credit) are divided among them such that the total CPU time share must be equal to the designated CPU time share for our experiment. For example, when a server has a CPU time share of 130, then two VMs will be configured with a CPU time share of 43 and one VM with a CPU time share 44.

3.2 Tools

We used the built-in xen application programmable interfaces (API) to control the host server. In particular, we controlled the server CPU frequency using the *xenpm* command [56] and controlled the server CPU time share using the *sched-credit* command [58]. Note that the middleware (virtual library) has an impact on scheduling the CPU time share among guests servers. In our work, we used the *fix credit* (or non-work conserving scheduler) as explained in [22], which guarantees that the guest server has its assigned CPU time share even when there are other running guest servers hosted by the same physical server. We used a power meter sensor



(a) CPU frequency input signal

(b) CPU time share (cap) input signal

(c) Coverage of input space

Fig. 3 The control input signals used for power modeling. We used a step input signal for CPU frequency and a sine wave input signal for CPU time share. Both input signals were used together to identify the MISO power model, and the coverage of the input space is the most right sub-figure. Notice the *vertical line* shape of the scattered input space, which is due to the step signal used for the CPU frequency

from Yoctopuce [59] attached to the power supply (power cable) connected to the server to monitor the power consumed by the server.

3.3 Workload

In our experiment, we focused on running all virtual machines (VMs) at their maximum CPU utilization while keeping other components, such as memory and network, at minimum utilization. In other words, all VMs are running a computation- or CPU-intensive workload application. There are several workload tools that can perform a computation-intensive workload; however, we created our own computation-intensive application. The application is a shell script that computes the value of (π), which is used to calculate the area of a circle, using the *Monte Carlo Algorithm* [37]. Similar to our previous works [2–4], the script is launched and continuously runs on all VMs during the experiment. In this way, we ensure that the utilization of all VMs' virtual CPUs (vCPUs) are at almost 100 % utilization, and consequently, the portion of the CPU time share designated according to the implementation is fully utilized.

3.4 Implementation

The control input signals are an important component of the system identification, and defining the shape of the signal is vital in the *black-box* system identification approach. On the one hand, we have two control input variables, namely, the CPU frequency and the CPU time share, and their rates of change are labeled in Eqs. 6, 7 and 8 as Δf and Δc, respectively. Both have a different range of values and steps. Therefore, we form two different series of control inputs that match the characteristics of each of them. Further explanation is provided in the following paragraphs. On the other hand, we have one output variable, which is the power consumption. The computing system is known to be a nonlinear system, and to generate a linear model for a nonlinear model, we should define an operating point in which the dynamic behavior of the system is linear. The power consumption of our server is operating linearly at the point of 138 watts second, so we set the operating point of the power consumption ($\tilde{p}$) to be 138 W s. It is not mandatory that 138 W s be the optimum operating point; different servers could have different operating points.

The available CPU frequencies of our host server are three levels, and they are converted to ratio levels with respect to the maximum frequency level (2.66 GHz). Consequently, we have three CPU frequency ratios: 1 (2.66 GHz), 0.87 (2.33 GHz) and 0.74 (2 GHz). We set the operating point of the CPU frequency ratio ($\tilde{f}$) to be 1, i.e., the max frequency 2.66 GHz available. We use a step signal to cover the CPU frequency range, as shown in Fig. 3a. We start from the minimum frequency ratio 0.74, then we step up the frequency ratio to the next levels, particularly 0.87 and 1, and then we step it down to the minimum level. The entire signal has a duration of 9,000 s, which is divided into five epochs as follows. The first epoch is from 0 to 1,500 s, the second epoch is from 1,500 to 3,000 s, the third epoch is from 3,000 to 6,000 s, the fourth epoch is 6,000–7,500 s, and the fifth and final epoch is from 7500 to 9000 seconds. At the first and last epochs, the CPU frequency ratio is set to 0.74 (or 2 GHz). At the second and fourth epochs, the CPU frequency ratio is set to 0.87 (or 2.33 GHz). At the middle epoch, the CPU frequency ratio is set to 1 (or 2.66 GHz). We designate each epoch to have a long duration because we would like to exploit the other control input range, the CPU time share, at each epoch of CPU frequency setting.

Because our host server's processor has two cores, the CPU time share has an input range from 1 to 200. However, we are interested in the rate of change in this configuration control input; therefore, we convert the CPU time share to a ratio between 1 to 100, and we set the operating point of the CPU time-share ratio ($\tilde{c}$) to be 0.75 (or 150 of 200) of the CPU time share. For the CPU time share, we use a sine wave signal to cover most of its range, as shown in Fig. 3b. The mean of the sine wave is 0.65 (or 130 of 200) of the CPU time share, with an amplitude of (197 of 200) of the CPU time share and a period of 720 seconds. The entire CPU time share input signal has a duration of 9000 seconds to match the duration of the other control signal (the CPU frequency signal).

The period of the sine wave signal is shorter than the duration of the epochs in the CPU frequency step signal, which helps us achieve a better coverage of the input space between the two control inputs; see Fig. 3c. In Fig. 3c, the input space is shaped as three vertical lines. These three lines are due to the step signals used for the CPU frequency control input. This is the best coverage we can obtain using the manufactured CPU frequency levels available. However, if we would like to have a scattered coverage, then we may use the proposed modulator in [31], but a special consideration has to be given to the impacts of the system identifications and estimation.

3.5 Model estimation

We applied the aforementioned control signals in three types of combinations, as shown in Fig. 2. The combinations are CPU frequency variation, CPU time share variation, and both variations. These combinations are used to estimate the three types of system models discussed in Sects. 2.2.2 (*physical* configuration SISO model), 2.2.3 (*virtual* configuration SISO model) and 2.2.4 (MISO model), respectively. After we performed these three combinations of experiments and collected the power consumption over different sampling times (Ts), we used a commercial software (a System Identification Toolbox available in MATLAB [34]) to estimate the parameters ($a, b_{f1}, b_{f2}, b_{c1}$ and b_{c2}) using the *Least Squares Regression* for each ARX power consumption model presented in the ARX model's Eqs. 6, 7 and 8.

4 Results

4.1 Power consumption modeling

The power consumption models are estimated as follows. Two 1st-order SISO models with a 1st-order input delay (CPU frequency and time share) and one 1st-order MISO model with a 1st-order input delay. Their ARX model's equations are as follows. The estimated 1st-order SISO power model with a 1st-order *physical* configuration input delay is presented in Eq. 11.

$$\Delta p(t+1) = 0.8014 \cdot \Delta p(t) + 10.58 \cdot \Delta f(t) \tag{11}$$

The estimated 1st-order SISO power model with a 1st-order *virtual* configuration input delay is presented in Eq. 12.

$$\Delta p(t+1) = 0.276 \cdot \Delta p(t) + 29.37 \cdot \Delta c(t) \tag{12}$$

Finally, the estimated 1st-order MISO power model with a 1st-order configuration input delay is presented in Eq. 13.

$$\Delta p(t+1) = 0.8334 \cdot \Delta p(t) + 8.031 \cdot \Delta f(t) + 5.478 \cdot \Delta c(t) \tag{13}$$

These models are used to predict the power consumption (output) of the host server and compare the predicted outputs with the actual measured data. Figure 4 (top row) shows the linear regression fits for each estimated model.

We also estimated the other higher-order models and input delays for all of the aforementioned model types. In this paper, we present only the 1st-order model with a 2nd-order input delay because it has the most appropriate linear regression fit (R^2). See the results of model orders in Sect. 4.2.

The estimated 1st-order SISO power model with a 2nd-order input delay (*physical* configuration) is presented in Eq. 14.

$$\Delta p(t+1) = 0.8682 \cdot \Delta p(t) + 61.86 \cdot \Delta f(t) - 54.86 \cdot \Delta f(t-1) \tag{14}$$

The estimated 1st-order SISO power model with a 2nd-order input delay (*virtual* configuration) is presented in Eq. 15.

$$\Delta p(t+1) = 0.263 \cdot \Delta p(t) + 28.57 \cdot \Delta c(t) + 1.332 \cdot \Delta c(t-1) \tag{15}$$

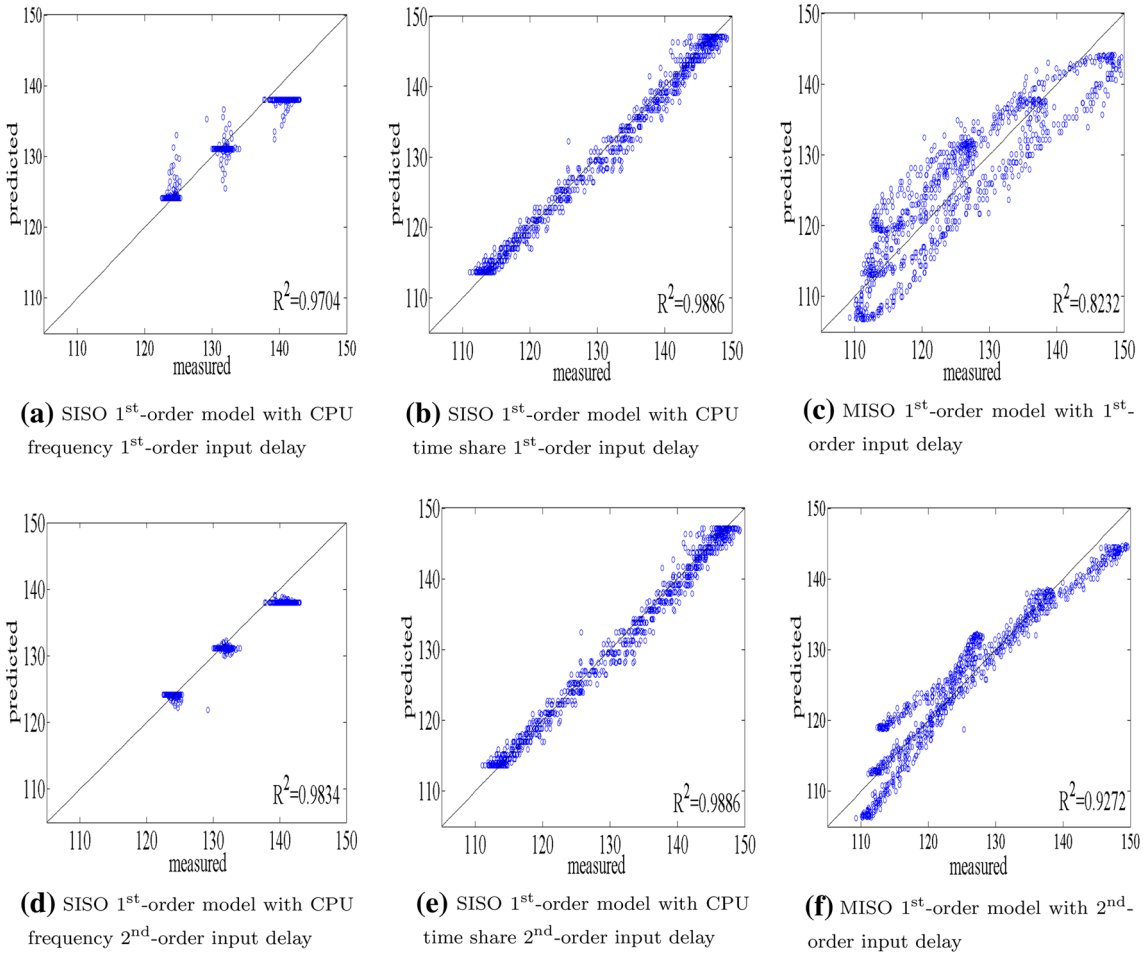
Finally, the estimated 1st-order MISO power model with a 2nd-order input delay is presented in Eq. 16.

$$\Delta p(t+1) = 0.9425 \cdot \Delta p(t) + 51.75 \cdot \Delta f(t) - 48.98 \cdot \Delta f(t-1) + 29.52 \cdot \Delta c(t) - 27.58 \cdot \Delta c(t-1) \tag{16}$$

These models are used to predict the power consumption (output) of the host server and compare the predicted outputs with the actual measured data. Figure 4 (bottom row) shows the fitting of each estimated model.

4.2 Model order selection

In the experiment, we investigated the ARX MISO model order. We fixed the sample time (Ts) and ran the experiment for 9000 seconds. After we collected the input and output



(a) SISO 1st-order model with CPU frequency 1st-order input delay

(b) SISO 1st-order model with CPU time share 1st-order input delay

(c) MISO 1st-order model with 1st-order input delay

(d) SISO 1st-order model with CPU frequency 2nd-order input delay

(e) SISO 1st-order model with CPU time share 2nd-order input delay

(f) MISO 1st-order model with 2nd-order input delay

Fig. 4 The linear regression fits (R^2) for all estimated power models. The *line* indicates the best fit between the actual and predicted values for the power consumed (in W s)

data, we estimated different model orders for our power consumption (MISO) model, and we also estimated different orders for the input delay. We reproduced the power consumption using all of the estimated MISO models, and we calculated their linear regression fits with the original data; Fig. 5 shows these linear regression fits. In the figure (Fig. 5), the x-axis represents the MISO model orders (from 1st to 4th), and the y-axis is the linear regression fits. If the regression fit is high, then the model is good. In the figure, there are four lines that represent the MISO input delay order, and they are varied (horizontally) with the MISO model order. From the figure, we can classify two groups of models: the improvable models and the non-improvable models.

The improvable models are the MISO models with a 1st-order input delay. This model has the worst fit among all models ($R^2 = 0.8232$); however, its regression fit improves when we consider higher-order models; that is, if we fix the input delay order to 1st-order and we increase the MISO model order to 2nd-order or higher, then the model fit improves. We increase the model order until we reach the 4th-order, and we achieve a 7 % improvement in the regression fit

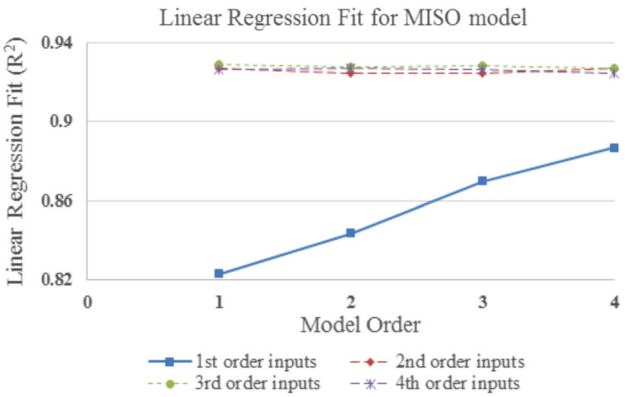
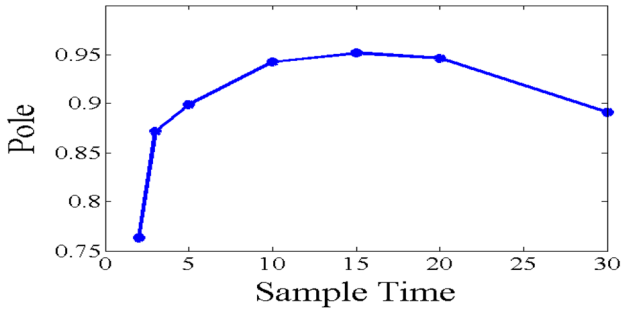
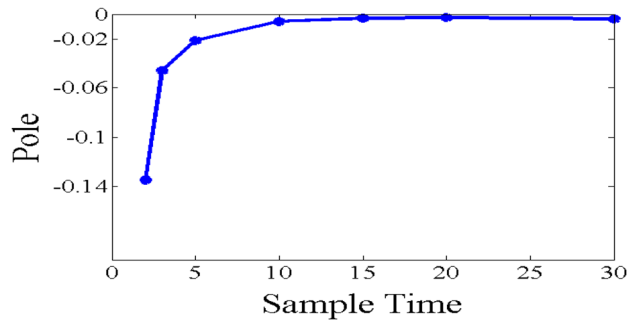


Fig. 5 The linear regression fit (R^2) for the MISO model. The *x-axis* shows the MISO order model, and the *lines* represent the order of the input delay. It is clear that a 1st-order MISO model with a 2nd-order input delay is sufficient for estimating the power model, and higher-order models have very low variation compared to the selected model

($R^2 = 0.8870$). On the other hand, the non-improvable models are the MISO models with a 2nd-order and higher input delay. All models in this group have almost similar



(a) Discrete-time System Model



(b) Continuous-time System Model

Fig. 6 The *pole* of the MISO system was plotted versus the sampling time to define the possible sampling time

linear regression fits ($R^2 = 0.9272$) across all MISO models' orders. As we discussed previously, if the model order is small, then the system model is easy to control. Therefore, the MISO model that is suitable for the power consumption in the host server is the 1st-order ARX MISO model with a 2nd-order input delay.

4.3 Pole and sampling time

In this subsection, we present the sampling time (Ts) study for the 1st MISO model with a 2nd-order input delay (as discussed in the previous Sect. 4.2). According to the experiment, Fig. 6 shows two sub-figures that illustrate the variation of the *pole* on both *discrete-time* (Fig. 6a) and *continuous-time* (Fig. 6b) models. We are interested in the *pole* behavior in the *continuous-time* model. From the point where the *pole* (in *continuous-time* models) starts to converge, we infer the best sampling time for the model. Therefore, in Fig. 6b, we can infer that the point where the *pole* start to converge is at 10 seconds, which reveals that the MISO model's sampling time (Ts) is 10 seconds.

5 Discussion

In this section, we evaluate our proposed MISO power model from two different perspectives. First, we compare our MISO power model with the SISO power models discussed in Sect. 2. Second, we evaluate the characteristics of our MISO power model in terms of stability and oscillatory behavior.

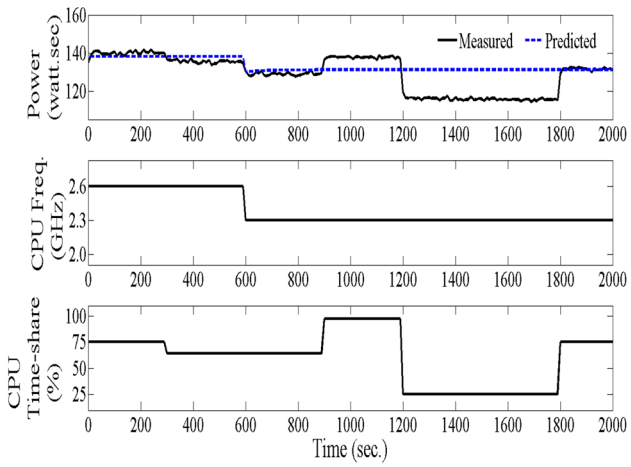


Fig. 7 The validation of a 1st-order SISO model with a 2nd-order input delay, where the control input is the *physical* configuration (CPU frequency scaling). This SISO model has not captured the behavior of the power consumption of virtualized hosting servers, and it did not accurately reproduce the output data

We defined a testing control input sequence to address the scenario of varying both the *physical* and *virtual* configurations during normal operation. The control input sequence is designed to act as a normal situation in which a host server could be configured at any time. The control sequence is shown in the two bottom sub-figures in Fig. 7. We applied the control input sequence to our host server, and we observed the power consumption during variations in the configuration. We supplied the same control input sequence to all three power consumption models estimated previously, namely, the SISO model with CPU frequency scaling as a single control input, the SISO model with varying CPU time share as a single control input, and the MISO model with both control inputs. All of these models have the same model orders, typically a 1st-order ARX model with a 2nd-order input delay. We compared the power consumption estimated by each model with the actual measured power consumption, as shown in Fig. 7 (SISO model with CPU frequency scaling only), Fig. 8 (SISO model with CPU time share only) and Fig. 9 (MISO model). Furthermore, to confirm the advantage of the 1st-order MISO model with a 2nd-order input delay, we supplied the same control input sequence to the estimated 1st-order MISO model with a 1st-order input delay. The power consumption is reported in Fig. 10.

For the SISO models, the power consumption was poorly estimated (see Figs. 7, 8). For example, in the SISO power model with the *physical* configuration (CPU frequency scaling control), the estimated power consumption was far from the true power measurements, and the linear regression fit

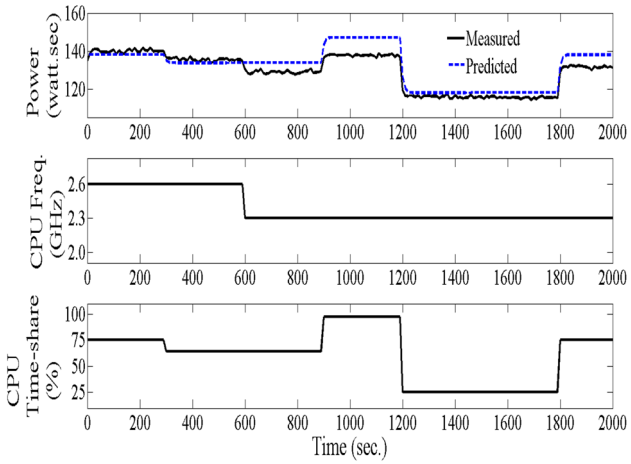


Fig. 8 The validation of a 1st-order SISO model with a 2nd-order input delay, where the control input is the *virtual* configuration (CPU time share). This SISO model has not captured the behavior of the power consumption of virtualized hosting servers. It did not accurately reproduce the output data

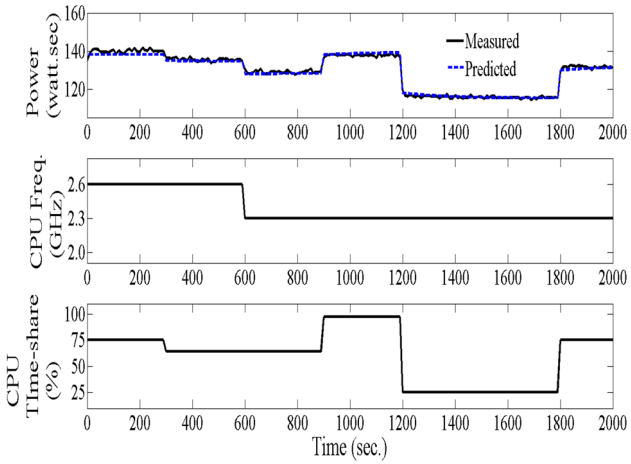


Fig. 9 The validation of a 1st-order MISO model with a 2nd-order input delay. This MISO model has exceptionally captured the behavior of the power consumption in virtualized hosting servers

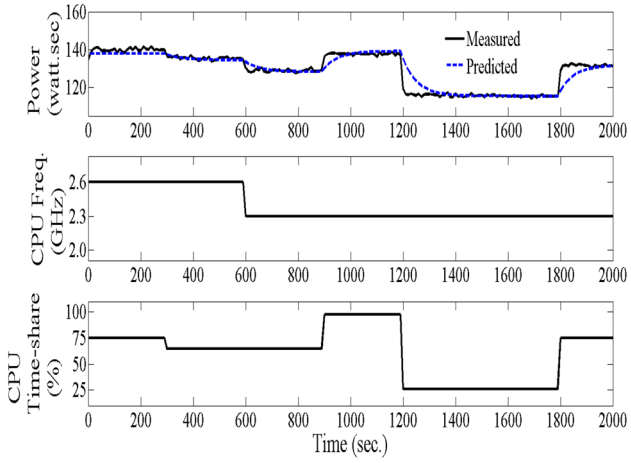


Fig. 10 The validation of a 1st-order MISO model with a 1st-order input delay. Although this MISO model has captured the behavior of the power consumption of virtualized hosting servers, it did not accurately reproduce the output data

(R^2) of the model was 0.3388. This result is due to the SISO model only considering the variation in one control input variable (CPU frequency scale) and ignoring the other potential variable, which is the CPU time share. Similarly, the other SISO power models with the *virtual* configuration (CPU time share control) estimated the power consumption with limited (but not acceptable) accuracy, and its fit (R^2) was 0.8505. For the same reason, this SISO model considers only the CPU time share variable and ignores the other critical power variable, which is the CPU frequency scale.

On the other hand, our proposed MISO power model (the 1st-order ARX MISO model with a 2nd-order input delay) was able to estimate and reproduce the power consumption with a very tiny variation from the measured data. Our model estimation was fit to the original measurements with a R^2 of 0.9886. This good estimation is a result of the benefits of having multiple inputs in the *black-box* system identification, which considers multiple variables in the estimated output. The high resolution ($R^2 = 0.9886$) in the estimation is a result of the 2nd-order input delay. To confirm the advantage of having a 2nd-order input delay in the model, we used the 1st-order ARX MISO power model with a 1st-order input delay, and we fitted the estimated power consumption with the original measurements. As expected, the estimated data were fairly consistent with the actual measured values; however, the model fit the measured data with a R^2 of 0.8888. This is a very low fitting ratio compared to our 1st-order MISO model with a 2nd-order input delay. Therefore, we favor the model with a 2nd-order input delay because of its high-resolution estimation. As a final observation, although the 1st-order MISO model with a 1st-order input delay provided a fair estimation of the power consumption for the host server, the model exhibited higher estimation fits compared to the other two SISO models. This result is due to the benefits of having multiple inputs in the *black-box* system identification.

According to the MISO model properties discussed in 2.4, and after reviewing our MISO model (Eq. 16), we can infer that our MISO power model is stable and not an oscillatory model. This result is due to its *pole*: it has a *pole* that resides between zero and one ($a = 0.9425$).

However, we do understand that limitations in our modeling exist. First, in our model estimations of the proposed MISO model, we did not consider the impacts of other physical and virtual configurations. For example, we did not consider the power consumption caused by memory, network, and storage, and we did not include the power consumed due

to the virtual configuration of these hardware components. Our MISO model could be extended to include all of these configuration changes, which could be achieved with a broad range and variety of input spaces used in the black-box system identification approach.

Because there are different types of machines with different capabilities, it is difficult to construct a single MISO model that can work with all types of machines. Therefore, we used a small-scale machine in this work as a proof-of-concept for the MISO power modeling, and we are going to apply the MISO power modeling methodology on higher and larger scale physical machines. However, we have not considered the recent advanced CPU frequency scaling capability in some virtualized multicore processors. For instance, some virtualized processors have their CPU cores divided into sets that share one supply voltage and operate on the same frequency [23].

6 Related works

System identification is important in computing system technology because it models many aspects of computing that are required to be controlled on demand. It is necessary to model the system performance, thermal dynamics and power consumption of a computing server. For example, the response time of a web server is modeled using system identification. In [1] [25], a *first-principles* system identification approach was used to construct a mathematical model for the web server response time. As a further extension of this approach (*first-principles* system identification), there is a move to construct a mathematical model in a *state space* model in which the system is considered as a Linear Parameter Varying (LPV). This direction appears in the works of Tanelli et al. [46–49], Lama et al. [29], Sun et al. [44], Wang et al. [55] and Qin et al. [39–41]. However, a *gray-box* system identification approach was used in [13,27,53]. Moreover, the admission control was included to support the system response time, and it was embedded into the system model as in Park et al. [38]. These contributions demonstrate the state-of-the-art in modeling the response time of web servers; however, they are generally *first-principles* approaches because it is possible to have initial mathematical equations from queuing theory. This is not the case with modeling of power consumption.

There are many diverse contributions for modeling the power consumption for computing servers and data centers. For example, there is an approach that uses basic power calculations to obtain an initial mathematical equation and that uses a training signal to predict the power consumption. This approach is a *gray-box* approach, in which the power calculation could be inherited from the microchip design (as illustrated by Zhuravlev et al. [61]) and the training signal could be analyzed, e.g., using a Fast Fourier Transform (FFT). Chen et al. [13,14] follow this approach; however, the authors in that work are focusing on improving the system performance.

Another approach is to construct a mathematical model for power consumption using the *black-box* system identification approach. This approach is widely used in server- and cluster-level power modeling, such as the contributions by Lefurgy et al. [30,31], Wang et al. [51–54] and Tanelli et al. [45,46]. In addition, Krishnan et al. [27] used this approach (the *black-box* approach) to study the host server's CPU and memory configuration impacts on virtual machine performance. The common implementation practice among these contributions is modeling the power consumption as a SISO system model. For example, Lefurgy et al. [30,31] modeled the power consumption as a SISO model, where the CPU frequency scale is the input of the system and the power consumption is the system output. Furthermore, Krishnan et al. [27] studied the server CPU frequency scale and memory modeling separately as SISO models. These contributions have escalated the design of the system controller to a multiple level and a hybrid design.

With the recent advances in computing technology that help improve system performance and power consumption, there are efforts to define a metering system for the power consumption in virtualized hosting servers. For example, Hagimont et al. [22] provided a good definition for the relationship between the CPU dynamic frequency scaling (DVFS) and the CPU time share (CPU credit). In addition, Krishnan et al. [27] investigated the feasibility and challenges for the server's CPU frequency scale and memory, where they examined the upper and lower bounds (mix. and min.) of the servers configurations. These meterings and studies support our observations on the escalation occurring in the control design for computing systems. Generally, these contributing works that identified computing systems in multiple SISO models to control power consumption and performance are suggesting a coordinated hybrid control design. For example, Deng et al. [15] constructed a coordinating system for both CPU and memory. Similar coordinating systems have been suggested by Ardagna et al. [5–7], Raghavendra et al. [42] and Luo et al. [36] [43].

To the best of our knowledge, our power consumption system modeling using the *black-box* system identification to construct a MISO model is a pioneering work in power consumption modeling. Our MISO model could reduce the degree of complexity in the power consumption control design.

7 Conclusion

Modeling and controlling the power consumption in host servers is taking its place in both development and industry. However, the rapid developments in computing resources

have made modeling the power consumption using a mathematical model a difficult task. There are efforts that consider only the physical configurations of a hosting server to identify the power consumption model as a single-input single-output system model. In this paper, we proposed a multi-input single-output power consumption model for virtualized host servers, in which it is possible to capture the impact of different configuration levels (i.e., physical and virtual configuration levels) in power consumption. We used the black-box system identification method to identify the MISO power model of virtualized host servers; we followed the auto-regressive exogenous (ARX) structure to construct the mathematical equations. Through empirical experiments, we evaluated both models (i.e., the SISO power model and our MISO power model) and showed that the MISO model captures the nature of power consumption in virtualized host servers better than the SISO model used in existing works. The prediction accuracy of power consumption using our model reaches 98 %, whereas the accuracy in SISO models is only 85 %.

In addition, in this paper, we studied the sampling time of the MISO model and the model order. Notably, we observed that there is a very noticeable impact of input delay in the power consumption modeling in both types (SISO and MISO) of power models. The 1st-order power model with a 2nd-order input delay has higher linear regression fits for predicting the power consumption in host servers than the 1st-order model with a 1st-order input delay.

There is no additional complexity when implementing our MISO model because it has a similar implementation level as other SISO models. However, we are working on integrating the performance of guest servers (virtual machines) into the MISO model.

Appendix: System dynamics of MISO power model

Any estimated model should be capable of predicting the dynamic behavior of a system, and our MISO model should have this prediction capability. Let us recall Eq. 8 and consider the 1st-order model and the 2nd-order input delay as the following Eq. 17

$$\Delta p(k) = a \cdot \Delta p(k-1) + b_{f1} \cdot \Delta f(k-1) + b_{f2} \cdot \Delta f(k-2) + b_{c1} \cdot \Delta c(k-1) + b_{c2} \cdot \Delta c(k-2). \quad (17)$$

Additionally, let us assume that we know the initial condition $\Delta p(1)$ and the inputs of the previous two steps $\Delta f(1)$, $\Delta f(0)$, $\Delta c(1)$, $\Delta c(0)$. Then, we could derive the solution to Eq. 17 as follows. The first prediction of $\Delta p(2)$ is

$$\Delta p(2) = a \cdot \Delta p(1) + b_{f1} \cdot \Delta f(1) + b_{f2} \cdot \Delta f(0) + b_{c1} \cdot \Delta c(1) + b_{c2} \cdot \Delta c(0), \quad (18)$$

and the second prediction $\Delta p(3)$ is

$$\Delta p(3) = a \cdot \Delta p(2) + b_{f1} \cdot \Delta f(2) + b_{f2} \cdot \Delta f(1) + b_{c1} \cdot \Delta c(2) + b_{c2} \cdot \Delta c(1). \quad (19)$$

We can re-write Eq. 19 after substituting Eq. 18, as follows

$$\Delta p(3) = a \cdot [a \cdot \Delta p(1) + b_{f1} \cdot \Delta f(1) + b_{f2} \cdot \Delta f(0) + b_{c1} \cdot \Delta c(1) + b_{c2} \cdot \Delta c(0)] + b_{f1} \cdot \Delta f(2) + b_{f2} \cdot \Delta f(1) + b_{c1} \cdot \Delta c(2) + b_{c2} \cdot \Delta c(1). \quad (20)$$

Or

$$\Delta p(3) = a^2 \cdot \Delta p(1) + a \cdot b_{f1} \cdot \Delta f(1) + a \cdot b_{f2} \cdot \Delta f(0) + a \cdot b_{c1} \cdot \Delta c(1) + a \cdot b_{c2} \cdot \Delta c(0) + b_{f1} \cdot \Delta f(2) + b_{f2} \cdot \Delta f(1) + b_{c1} \cdot \Delta c(2) + b_{c2} \cdot \Delta c(1). \quad (21)$$

The third prediction $\Delta p(4)$ is

$$\Delta p(4) = a \cdot \Delta p(3) + b_{f1} \cdot \Delta f(3) + b_{f2} \cdot \Delta f(2) + b_{c1} \cdot \Delta c(3) + b_{c2} \cdot \Delta c(2). \quad (22)$$

Similarly, we can re-write Eq. 22 after substituting Eq. 21, as follows

$$\Delta p(4) = a \cdot [a^2 \cdot \Delta p(1) + a \cdot b_{f1} \cdot \Delta f(1) + a \cdot b_{f2} \cdot \Delta f(0) + a \cdot b_{c1} \cdot \Delta c(1) + a \cdot b_{c2} \cdot \Delta c(0) + b_{f1} \cdot \Delta f(2) + b_{f2} \cdot \Delta f(1) + b_{c1} \cdot \Delta c(2) + b_{c2} \cdot \Delta c(1)] + b_{f1} \cdot \Delta f(3) + b_{f2} \cdot \Delta f(2) + b_{c1} \cdot \Delta c(3) + b_{c2} \cdot \Delta c(2) \quad (23)$$

Or

$$\Delta p(4) = a^3 \cdot \Delta p(1) + a^2 \cdot b_{f1} \cdot \Delta f(1) + a^2 \cdot b_{f2} \cdot \Delta f(0) + a^2 \cdot b_{c1} \cdot \Delta c(1) + a^2 \cdot b_{c2} \cdot \Delta c(0) + a \cdot b_{f1} \cdot \Delta f(2) + a \cdot b_{f2} \cdot \Delta f(1) + a \cdot b_{c1} \cdot \Delta c(2) + a \cdot b_{c2} \cdot \Delta c(1) + b_{f1} \cdot \Delta f(3) + b_{f2} \cdot \Delta f(2) + b_{c1} \cdot \Delta c(3) + b_{c2} \cdot \Delta c(2). \quad (24)$$

Then, we can obtain the following solution

$$\Delta p(k) = a^{k-1} \cdot \Delta p(1) + \sum_{i=1}^{k-1} a^{k-1-i} \cdot b_{f1} \cdot \Delta f(i) + \sum_{i=1}^{k-1} a^{k-1-i} \cdot b_{f2} \cdot \Delta f(i-1) + \sum_{i=1}^{k-1} a^{k-1-i} \cdot b_{c1} \cdot \Delta c(i) + \sum_{i=1}^{k-1} a^{k-1-i} \cdot b_{c2} \cdot \Delta c(i-1), \quad (25)$$

where k is greater than 1 ($k > 1$), and a, b_{f1}, b_{f2}, b_{c1}, and b_{c2} are estimated parameters.

References

1. Abdelzaher, T.F., Shin, K.G., Bhatti, N.: Performance guarantees for web server end-systems: a control-theoretical approach. IEEE Trans. Parallel Distrib. Syst. **13**(1), 80–96 (2002)
2. Al-Hazemi, F.: A hybrid green policy for admission control in web-based applications. In: 2013 21st International Conference on Software, Telecommunications and Computer Networks (SoftCOM), pp. 1–6. IEEE (2013)
3. Al-Hazemi, F.: A polymorphic green service approach for data center energy consumption management. In: Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing, pp. 110–117. IEEE (2013)
4. Al-Hazemi, F.: Feedback green control for data centers autonomy. In: 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing (UCC), pp. 333–338. IEEE (2013)
5. Ardagna, D., Cappiello, C., Lovera, M., Pernici, B., Tanelli, M.: Active energy-aware management of business-process based applications. In: Towards a Service-Based Internet, pp. 183–195. Springer (2008)
6. Ardagna, D., Tanelli, M., Lovera, M., Zhang, L.: Black-box performance models for virtualized web service applications. In: Proceedings of the First Joint WOSP/SIPEW International Conference on Performance Engineering, pp. 153–164. ACM (2010)
7. Ardagna, D., Panicucci, B., Trubian, M., Zhang, L.: Energy-aware autonomic resource allocation in multitier virtualized environments. IEEE Transactions on Services Computing **5**(1), 2–19 (2012)
8. Bartolini, A., Cacciari, M., Tilli, A., Benini, L.: Thermal and energy management of high-performance multicores: distributed and self-calibrating model-predictive controller. IEEE Trans. Parallel Distrib. Syst. **24**(1), 170–183 (2013)
9. Beneventi, F., Bartolini, A., Tilli, A., Benini, L.: An effective gray-box identification procedure for multicore thermal modelling (2013)
10. Benini, L., Tilli, A., Bartolini, A., Beneventi, F.: An effective gray-box identification procedure for multicore thermal modeling. IEEE Trans. Comput. **63**(5), 1097–1110 (2014)
11. Chen, S., Joshi, K.R., Hiltunen, M.A., Sanders, W.H., Schlichting, R.D.: Link gradients: predicting the impact of network latency on multitier applications. In: INFOCOM 2009, IEEE, pp. 2258–2266. IEEE (2009)
12. Chen, S., Joshi, K.R., Hiltunen, M.A., Schlichting, R.D., Sanders, W.H.: Cpu gradients: performance-aware energy conservation in multitier systems. In: Green Computing Conference, 2010 International, pp. 15–29. IEEE (2010)
13. Chen, S., Joshi, K.R., Hiltunen, M.A., Schlichting, R.D., Sanders, W.H.: Blackbox prediction of the impact of dvfs on end-to-end performance of multitier systems. ACM SIGMETRICS Perform. Eval. Rev. **37**(4), 59–63 (2010)
14. Chen, S., Joshi, K.R., Hiltunen, M.A., Schlichting, R.D., Sanders, W.H.: Using CPU gradients for performance-aware energy conservation in multitier systems. Sustain. Comput. Inf. Syst. **1**(2), 113–133 (2011)
15. Deng, Q., Meisner, D., Bhattacharjee, A., Wenisch, T.F., Bianchini, R.: Coscale: coordinating CPU and memory system DVFs in server systems. In: 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), pp. 143–154. IEEE (2012)
16. Diao, Y., Gandhi, N., Hellerstein, J.L., Parekh, S., Tilbury, D.M.: Using mimo feedback control to enforce policies for interrelated metrics with application to the apache web server. In: Network Operations and Management Symposium, 2002. NOMS 2002. 2002 IEEE/IFIP, pp. 219–234. IEEE (2002)
17. Gandhi, N., Tilbury, D., Diao, Y., Hellerstein, J., Parekh, S.: Mimo control of an apache web server: Modeling and controller design. In: Proceedings of the 2002 American Control Conference, 2002, vol. 6, pp. 4922–4927. IEEE (2002)
18. Ge, R., Feng, X., Song, S., Chang, H.C., Li, D., Cameron, K.W.: Powerpack: energy profiling and analysis of high-performance systems and applications. IEEE Trans. Parallel Distrib. Syst. **21**(5), 658–671 (2010)
19. Goiri, Í., Katsak, W., Le, K., Nguyen, T.D., Bianchini, R.: Designing and managing datacenters powered by renewable energy. IEEE Micro, p. 1 (2014)
20. Gurumurthi, S., Sivasubramaniam, A., Kandemir, M., Franke, H.: DRPM: dynamic speed control for power management in server class disks. In: Proceedings of the 30th Annual International Symposium on Computer Architecture, 2003, pp. 169–179. IEEE (2003)
21. Gurumurthi, S., Sivasubramaniam, A., Kandemir, M., Franke, H.: Reducing disk power consumption in servers with DRPM. Computer **36**(12), 59–66 (2003)
22. Hagimont, D., Kamga, C.M., Broto, L., Tchana, A., De Palma, N.: DVFs aware CPU credit enforcement in a virtualized system. In: Middleware 2013, pp. 123–142. Springer (2013)
23. Han, J.J., Wu, X., Zhu, D., Jin, H., Yang, L.T., Gaudiot, J.L.: Synchronization-aware energy management for vfi-based multicore real-time systems. IEEE Trans. Comput. **61**(12), 1682–1696 (2012)
24. Hellerstein, J.L., Diao, Y., Parekh, S., Tilbury, D.M.: Feedback Control of Computing Systems. Wiley (2004)
25. Hellerstein, J.L., Diao, Y., Parekh, S.: A first-principles approach to constructing transfer functions for admission control in computing systems. In: Proceedings of the 41st IEEE Conference on Decision and Control, 2002, vol. 3, pp. 2906–2912. IEEE (2002)
26. Horvath, T., Abdelzaher, T., Skadron, K., Liu, X.: Dynamic voltage scaling in multitier web servers with end-to-end delay control. IEEE Trans. Comput. **56**(4), 444–458 (2007)
27. Krishnan, B., Amur, H., Gavrilovska, A., Schwan, K.: VM power metering: feasibility and challenges. ACM SIGMETRICS Perform. Eval. Rev. **38**(3), 56–60 (2011)
28. KVM. http://www.linux-kvm.org/. Accessed June 5, 2014
29. Lama, P., Zhou, X.: Efficient server provisioning with control for end-to-end response time guarantee on multitier clusters. IEEE Trans. Parallel Distrib. Syst. **23**(1), 78–86 (2012)
30. Lefurgy, C., Wang, X., Ware, M.: Server-level power control. In: Fourth International Conference on Autonomic Computing, 2007. ICAC'07, pp. 4–4. IEEE (2007)
31. Lefurgy, C., Wang, X., Ware, M.: Power capping: a prelude to power shifting. Clust. Comput. **11**(2), 183–195 (2008)
32. Lin, J., Zheng, H., Zhu, Z., David, H., Zhang, Z.: Thermal Modeling and Management of DRAM Memory Systems, vol. 35. ACM (2007)
33. Lin, J., Zheng, H., Zhu, Z., Zhang, Z.: Thermal modeling and management of dram systems. IEEE Trans. Comput. **62**(10), 2069–2082 (2013)
34. Ljung, L.: System identification toolbox for use with {MATLAB} (2007)
35. Ljung, L.: System Identification—Theory for the User. Prentice-Hall (1999)
36. Luo, J., Rao, L., Liu, X.: Temporal load balancing with service delay guarantees for data center energy cost optimization. IEEE Trans. Parallel Distrib. Syst. **25**(3), 775–784 (2014)
37. Monte carlo method. http://en.wikipedia.org/wiki/Monte_Carlo_method/. Accessed Oct 19, 2014
38. Park, S.M., Humphrey, M.A.: Predictable high-performance computing using feedback control and admission control. IEEE Trans. Parallel Distrib. Syst. **22**(3), 396–411 (2011)

39. Qin, W., Wang, Q.: An lpv approximation for admission control of an internet web server: identification and control. Control Eng. Pract. **15**(12), 1457–1467 (2007)
40. Qin, W., Wang, Q.: Modeling and control design for performance management of web servers via an lpv approach. IEEE Trans. Control Syst. Technol. **15**(2), 259–275 (2007)
41. Qin, W., Wang, Q., Sivasubramiam, A.: An-stable model-based linear-parameter-varying control for managing server performance under self-similar workloads. IEEE Trans. Control Syst. Technol. **17**(1), 123–134 (2009)
42. Raghavendra, R., Ranganathan, P., Talwar, V., Wang, Z., Zhu, X.: No power struggles: Coordinated multi-level power management for the data center. In: ACM SIGARCH Computer Architecture News, vol. 36, pp. 48–59. ACM (2008)
43. Rao, L., Liu, X., Xie, L., Liu, W.: Coordinated energy cost management of distributed internet data centers in smart grid. IEEE Trans. Smart Grid **3**(1), 50–58 (2012)
44. Sun, Q., Dai, G., Pan, W.: Lpv model and its application in web server performance control. In: Proceedings of the 2008 International Conference on Computer Science and Software Engineering, vol. 03, pp. 486–489. IEEE Computer Society (2008)
45. Tanelli, M., Ardagna, D., Lovera, M., Zhang, L.: Model identification for energy-aware management of web service systems. In: Service-Oriented Computing-ICSOC 2008, pp. 599–606. Springer (2008)
46. Tanelli, M., Ardagna, D., Lovera, M.: Lpv model identification for power management of web service systems. In: IEEE International Conference on Control Applications, 2008. CCA 2008, pp. 1171–1176. IEEE (2008)
47. Tanelli, M., Schiavoni, N., Ardagna, D., Lovera, M.: Control-oriented multirate lpv modelling of virtualized service center environments. In: Proceedings of the 48th IEEE Conference on Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009, pp. 7412–7417. IEEE (2009)
48. Tanelli, M., Ardagna, D., Lovera, M.: LPV model identification in virtualized service center environments. Syst. Identif. **15**, 862–867 (2009)
49. Tanelli, M., Ardagna, D., Lovera, M.: Identification of lPV state space models for autonomic web service systems. IEEE Trans. Control Syst. Technol. **19**(1), 93–103 (2011)
50. Wang, X., Du, Z., Chen, Y., Li, S.: Virtualization-based autonomic resource management for multi-tier web applications in shared data center. J. Syst. Softw. **81**(9), 1591–1608 (2008)
51. Wang, X., Chen, M., Fu, X.: Mimo power control for high-density servers in an enclosure. IEEE Trans. Parallel Distrib. Syst. **21**(10), 1412–1426 (2010)
52. Wang, X., Ma, K., Wang, Y.: Adaptive power control with online model estimation for chip multiprocessors. IEEE Trans. Parallel Distrib. Syst. **22**(10), 1681–1696 (2011)
53. Wang, X., Wang, Y.: Coordinating power control and performance management for virtualized server clusters. IEEE Trans. Parallel Distrib. Syst. **22**(2), 245–259 (2011)
54. Wang, Y., Wang, X.: Virtual batching: request batching for server energy conservation in virtualized data centers. IEEE Trans. Parallel Distrib. Syst. **24**(8), 1695–1705 (2013)
55. Wang, Y., Wang, X.: Performance-controlled server consolidation for virtualized data centers with multi-tier applications. Sustain. Comput: Inf. Syst. **4**(1), 52–65 (2014)
56. Xen power management. http://wiki.xen.org/wiki/Xenpm_command/. Accessed June 5, 2014
57. Xen project. http://wiki.xen.org/. Accessed June 5, 2014
58. Xen sched-credit. http://wiki.xen.org/wiki/Credit_Scheduler/. Accessed June 5, 2014
59. Yoctopuce. http://www.yoctopuce.com/. Accessed June 5, 2014
60. Yu, L., Jiang, T., Cao, Y., Zhang, Q.: Risk-constrained operation for internet data centers in deregulated electricity markets. IEEE Trans. Parallel Distrib. Syst. **25**(5), 1306–1316 (2014)
61. Zhuravlev, S., Saez, J.C., Blagodurov, S., Fedorova, A., Prieto, M.: Survey of energy-cognizant scheduling techniques. IEEE Trans. Parallel Distrib. Syst. **24**(7), 1447–1464 (2013)

Fawaz Al-Hazemi received his B.S. degree in Computer Engineering from KFUPM, Saudi Arabia in 2003 and his M.S. degree in Information and Communications Engineering from KAIST, South Korea, in 2010. He is pursuing his PhD in Electrical Engineering at KAIST. His research interests are green datacentre and energy-aware computing. He is member of Technical Program Committee (TPC) for several international conferences as well as he servers as reviewer in IEEE Transactions on Computers and IEEE Computer Magazine. He published more than 19 technical papers, and he was the recipient of FTRA AIM 2014 "Best Paper Award". He is a member of IEEE (IEEE Computer & IEEE ComSoc) and ACM (SIGCOMM, & SIGMETRICS).

Yuyang Peng received M.S. and Ph.D. degrees in Electrical and Electronic Engineering from Chonbuk National University, Jeonju, Korea in 2011 and 2014, respectively. He is currently a post-doctoral research fellow in Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea. His current research interests include cooperative communication, wireless sensor network, energy optimization, cloud computing.

Chan-Hyun Youn received the B.S. and M.S. degrees in electronics engineering from Kyungpook National University, Korea, in 1981 and 1985, respectively, and the Ph.D. degree in electrical and communications engineering from Tohoku University, Japan, in 1994. From 1986 to 1997, he worked for KT Telecom Network Research Laboratories, where he was a Principal Investigator of high-speed networking projects. Since 2009, he has been a Professor in the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea. Where, he is a Dean of Office of Planning and Budgets and is also a Director of Grid Middleware Research Center, KAIST. His research areas are cloud, computing workflow system, distributed system, communication middleware, and e-Health application services. Dr. Youn was the Editor-in-Chief in the Korea Information Processing Society, an editor of the Journal of Healthcare Engineering, U.K., and was the head of Korea branch (computer section) of the IEICE, Japan, during 2009–2010. He is a member of the IEEE.