NIPD: Non-Intrusive Power Disaggregation in Legacy Datacenters

Guoming Tang, *Student Member, IEEE,*
Weixiang Jiang, Zhifeng Xu, Fangming Liu, *Senior Member, IEEE,* and Kui Wu, *Senior Member, IEEE*

Abstract—Fine-grained power monitoring, which refers to power monitoring at the server level, is critical to the efficient operation and energy saving of datacenters. Fined-grained power monitoring, however, is extremely challenging in legacy datacenters that host server systems not equipped with power monitoring sensors. Installing power monitoring hardware at the server level not only incurs high costs but also complicates the maintenance of high-density server clusters and enclosures. In this paper, we present a zero-cost, purely software-based solution to this challenging problem. We use a novel technique of non-intrusive power disaggregation (NIPD) that establishes power mapping functions (PMFs) between the states of servers and their power consumption, and infer the power consumption of each server with the aggregated power of the entire datacenter. The PMFs that we have developed can support both linear and nonlinear power models via the state feature transformation. To reduce the training overhead, we further develop adaptive PMFs update strategies and ensure that the training data and state features are appropriately selected. We implement and evaluate NIPD over a real-world datacenter with 326 nodes. The results show that our solution can provide high precision power estimation at both rack level and server level. In specific, with PMFs including only two nonlinear terms, our power estimation i) at rack level has mean relative error of 2.18%, and ii) at server level has mean relative errors of 9.61% and 7.53% corresponding to the idle and peak power, respectively.

Index Terms—Datacenter, servers, power monitoring, non-intrusive power disaggregation.

1 Introduction

Deployed all over the world to host computing services and data storage, datacenters have become indispensable in the modern information technology (IT) landscape. With rapid expansion of datacenters in both number and scale, their energy consumption is increasing dramatically. The energy expense has become one of the most significant operating costs in today's datacenters. Companies like Amazon, Google, IBM, Microsoft, and Facebook, pay tens of millions of dollars every year for electricity [1]. A recent report [2] states that in US alone "*datacenter electricity consumption is projected to increase to roughly* 140 *billion kilowatt-hours annually by 2020, the equivalent annual output of 50 power plants, costing American businesses* $13 *billion annually in electricity bills and emitting nearly* 1004 *million metric tons of carbon pollution per year.*" To tackle the problem, more attention than ever has been paid to power management (PM) in today's datacenters [3], [4].

Power monitoring is the foundation of power management. Fine-grained power monitoring, which refers to power monitoring at the server level in this paper, is of particular importance. Fine-grained power monitoring facilitates the implementation of various power management strategies, such as power capping and accounting [5], [6], idle power eliminating [7], and even cooling control and load balancing [8]. A fine-grained power monitoring platform not only helps audit the total energy use of the datacenter but also continuously shows the real-time server-level power consumption. Such a platform can greatly help the datacenter operators to adjust their power management policies and explore potential benefits. Taking the cooling control as an example, to optimize the air flow and locate the thermal "hot spot"(which refers to server input air conditions that are either too hot or too dry and may hamper the efficiency of the datacenter.) in a datacenter, the real-time feedback of server-level power distribution can provide important information to identify hot spot "suspects".

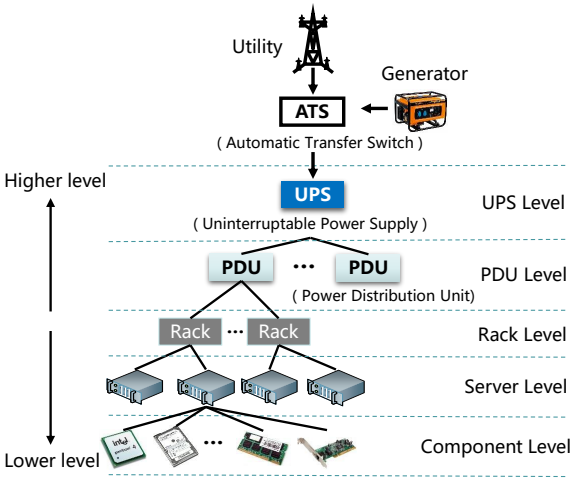


Fig. 1: Power distribution hierarchy of the IT facilities in a typical datacenter.

In addition to the aforementioned significance, fine-

- *G. Tang and K. Wu are with the Department of Computer Science, University of Victoria, Victoria, B.C, Canada. E-mail: {guoming, wkui}@uvic.ca.*
- *W. Jiang, Z. Xu and F. Liu are with the Services Computing Technology and System Lab, Cluster and Grid Computing Lab in the School of Computer Science and Technology, Huazhong University of Science and Technology, 1037 Luoyu Road, Wuhan 430074, China. The corresponding author is F. Liu. E-mail:* fmliu@hust.edu.cn.

grained power monitoring is also critical to the safe operations of datacenters. With continuous scaling-out (i.e., adding computing resources) and scaling-up (i.e., upgrading IT facilities), the maximum power capacity of a datacenter may be quickly reached. According to [9], 30% of the enterprise datacenters are expected to run out of the power capacity within 12 months. The datacenter operators are facing the dilemma of limited power capacity and increased power demand. Meanwhile, it is also realized that the power capacity from servers' nameplate power is normally over provisioned. For example, it is shown in [10] that the peak power of a server (145 *Watts*) was less than 60% of its nameplate value (251 *Watts*) in normal cases. As a result, more and more operators tend to overbook the power infrastructure for a high percentile of their needs [11]. Overbooking, however, may cause power deficit at some level of IT facilities (illustrated in Fig. 1). Even worse, once the power usage at a lower level exceeds its power capacity, if no actions were taken immediately, the impact cascades and results in the overrun or system crash at a higher level [12]. Hence, fine-grained power monitoring of the IT facilities is in urgent need of to ensure the safe operation of datacenters.

Nevertheless, fine-grained power monitoring is extremely challenging in datacenters that house diverse legacy servers as well as high-density blade servers and enclosures. While high-density computer systems greatly reduce the space of IT infrastructure and simplify the cabling process, many widely-used blade servers, such as DELL PowerEdge M100e, HPE ProLiant DL380 and some of IBM BladeCenter H series, are not equipped with power sensors, and power meters are typically installed at power distribution units (PDU) or at the rack level (refer to Fig. 1). In a legacy datacenter consisting of tens of racks, each with hundreds of servers not equipped with power sensors, how can we precisely capture the real-time power consumption of each server?

Naturally, one solution is to use power measurement hardware. For instance, SynapSense [13] has developed power monitoring solutions using power clamps or intelligent power strips. The IBM PowerExecutive solution [14] installs dedicated power sensors on servers during manufacturing to provide real-time power information of individual servers. Despite the above solutions, many legacy or even most recent server systems used in datacenters, such as DELL PowerEdge M100e and IBM BladeCenter H series, are not equipped with power measuring units. In this case, it is inconvenient for datacenter operators to install extra power meters on racks and it is extremely hard and costly to assemble power meters to individual blade servers as they are highly compacted in racks. This difficult task is typically contracted out to companies specialized in datacenter power monitoring, such as NobleVision [15] and ServerTechnology [16], that combine special hardware and intelligent software for fine-grained power monitoring.

Due to the above difficulties and also for cost saving, software-based solutions are more desirable and have been adopted in power monitoring platforms, such as PowerPack [17], [18] and Mantis [19]. They estimate the power consumption of individual servers by using power models, which are learned from the dependence between the server power consumption and the utilization of resources (e.g., CPU, memory, disk and NIC). During the learning process of power models, however, power measurements at the server or component level (refer to Fig. 1) are usually needed. We call this type of software-based solutions *intrusive*, because they need power metering at the server or lower levels for initial model training. Although it is possible to train a power model offline using model machines and then apply it to estimate online power consumption, the offline training has to be performed for each category of servers. This makes offline power model training a tedious and even impractical task for a large-scale datacenter hosting heterogeneous servers.

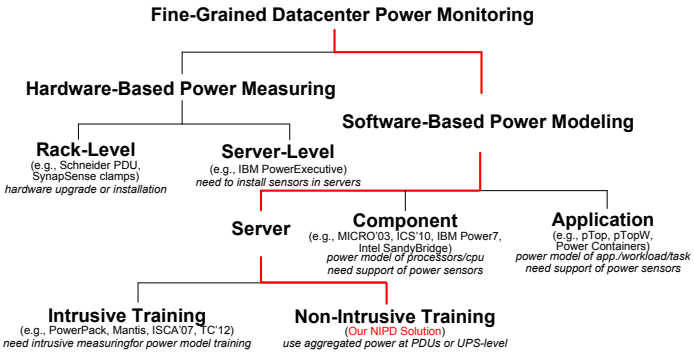


Fig. 2: Classification of fine-grained power monitoring for datacenters.

We are thus motivated to develop a non-intrusive, purely software-based, fine-grained power monitoring solution for legacy datacenters. Our method is non-intrusive in the sense that it does not need power monitoring at the server or lower levels for initial model training. We use a novel technique, called non-intrusive power disaggregation (NIPD), to achieve fine-grained power monitoring in datacenters. NIPD establishes power mapping functions (PMFs) between the states of a server and its power consumption and uses PMFs to infer the power consumption of each server with the aggregated power of the entire datacenter. Compared with existing methods, our solution is unique, as illustrated in Fig. 2. Specifically, the main contributions of our work include:

- We are the first to formally define the problem of non-intrusive power disaggregation (NIPD) in datacenters and develop NIPD models using power mapping functions (PMFs) trained with power data from running datacenters. Our solution can extract server-level power consumption from aggregated power consumption of the whole datacenter, and can be easily implemented in modern datacenters.
- We extend the servers' original state feature space to a dilated one using the technique of state feature transformation (SFT). With linear or nonlinear SFT in the dilated feature space, we can train both linear and nonlinear PMFs and offer much more choices for server power modeling.
- We further improve the NIPD solution to support adaptive PMFs update, with strategies of selective training data collection and iterative state feature elimination. The adaptive PMFs update can not only

TABLE 1: Comparison of current fine-grained power monitoring solutions vs. our NIPD solution

specification / *solution*	*software/ hardware*	*intrusive/ non-intrusive*	*monitoring levels*	*deployed scale (#server)*	*homogeneous/ heterogeneous*	*error rate*	*hardware cost ($/server)*
Schneider Solution [20] SynapSense Solution [13]	hardware	intrusive	rack	any scale	both	($< 1\%$)	55 - 68
PowerExecutive [14]	hardware	intrusive	rack server	any scale	both	($< 1\%$)	2K - 2.8K
PowerPack [17], [18]	software	intrusive	rack server component	9	homogeneous	($< 1\%$)	15 - 80
Mantis [19]	software	intrusive	rack server	1	–	$< 15\%$	30 - 200
ISCA'07 [10]	software	intrusive	rack server	≥ 100	homogeneous	$< 18\%$	30 - 200
TC'12 [21]	software	intrusive	rack server component	1	–	$< 9\%$	129 - 340
Our NIPD Solution	software	non-intrusive	rack server	326	both	$< 10\%$	0

1. The *error rates* shown in brackets are the measurement errors of corresponding power meters or sensors.
2. The *hardware cost* of each solution is estimated by the current prices of measuring devices (or prices of similar products) on the market.

effectively improve the precision of power disaggregation, but also significantly reduce the overhead of PMFs training process.

- We implement our NIPD solution over a 326-node datacenter without introducing any extra meters (i.e., zero-cost). The results show that our solution can provide high precision power estimation. The linear PMFs provide rack level estimation with mean relative error (MRE) of 2.63%, and server level estimation with MREs of 10.27% and 8.17% for idle power and peak power, respectively. The PMFs with two nonlinear terms provide rack level estimation with MRE of 2.18%, and server level estimation with MREs of 9.61% and 7.53% for idle power and peak power, respectively.

The rest of the paper is organized as follows. In Sec. 2, we review related work for fine-grained power monitoring in the datacenter. The overview and rationale of our non-intrusive power monitoring (NIPD) idea are explained in Sec. 3. Then, we formally define the problem of NIPD, develop models, and provide solutions in Sec. 4. We further implement NIPD over a real-world datacenter in Sec. 5 and evaluate its performance in Sec. 6.. The paper is concluded in Sec. 7.

2 Related Work

In this section, we first review existing solutions for fine-grained power monitoring in the datacenter, including hardware-based power measuring and software-based power modeling [22] [23]. Then, focusing on server-level power monitoring, we briefly summarize the key points that need to be considered and the major improvements of this paper upon our previous work [24].

2.1 Fine-Grained Power Monitoring

2.1.1 Hardware-Based Power Measuring

Power monitoring with dedicated hardware is regarded as the most accurate yet expensive approach to obtain the fine-grained power information. For rack-level power monitoring, Schneider Electric [20] provided the metered Rack Power Distribution Units (RPDU). In addition, SynapSense [13] developed power monitoring solutions using devices like power clamps and intelligent power strips. Regarding the power monitoring at the server level, IBM developed its own power management system, PowerExecutive [14], which utilized the embedded sensors to measure the power usage and allowed users to monitor power consumption at the server level. For a large-scale datacenter not adopting the above equipments, the upgrade of power infrastructures and IT facilities would be prohibitively expensive.

2.1.2 Software-Based Power Modeling

This type of solution establishes power models to estimate the power consumption of a server, using information collected from the level of servers, components, or applications. Power models could be built for server, hardware components, or applications. Here, we only review the power models built for the servers, since they are mostly related to our work.

Server-level power models are usually trained based on the correlation between the state (or resource utilization) of individual hardware component and power consumption of corresponding component. In [25], using hardware performance monitoring counters (PMCs), a surrogate linear regression model was applied to build the power model and predict the power consumption of computer system. The notion of "ensemble-level" power management was proposed in [26], which leveraged usage patterns of concurrent resource across the individual server blades for power monitoring and saving. Power modeling with microprocessor PMCs was proposed in [27]. Five different sever-level power models, which correlate AC power measurements with software utilization metrics, were investigated in [28], in which the accuracy and portability of the power models were also compared over different workloads and servers.

Various power monitoring platforms were developed utilizing software-based power modeling. PowerPack was initially established in [17] and further improved in [18]. It was implemented on a small-scale (9 nodes) cluster and

supports power measuring of individual servers as well as power estimation of parallel applications. In [19], a hybrid hardware-software infrastructure, Mantis, was introduced. It first collected the individual server power consumption using an AC power meter. By correlating the power consumption data with system utilization metrics, it trained a linear regression model to predict server-level power consumption based on system utilization data. In [10], a power model was learned from the aggregated power of a few hundreds of homogeneous servers along with their corresponding CPU utilization.

2.2 Key Points of Server-Level Power Monitoring

2.2.1 Linear/Non-Linear Power Models

Among the developed power models for the server-level power estimation, the linear regression model is the most widely applied. It is simple and has been shown to yield accurate results [19], [25]. Nevertheless, there is evidence showing that nonlinear power models can also be good alternatives in some cases [28], [29]. Therefore, both linear and nonlinear power models may need to be considered before we choose one of them for the power estimation at server level.

2.2.2 Intrusive/Non-Intrusive Power Model Training

To obtain the regression coefficients in the power model, a model training process (e.g., least square estimation) is needed. In existing power model training, either server-level or component-level power information is required. As such, we call these methods intrusive since power measuring at the server or lower levels is needed during the initial model training phase, even if afterwards no hardware-based power measuring is needed. As the difficulty of training data acquisition because of the integration of servers in enclosures, non-intrusive training can be a more practical approach.

2.2.3 Improvements Upon Previous Work

In our previous work [24], we proposed a non-intrusive power model training process, in which the power model was defined in a linear form. In this paper, we extend the power modeling to support both linear and nonlinear forms, and this improvement provides more choices for server power modeling and increases the accuracy of power estimation. Moreover, in [24], we manually specified state features to build up power model. As an improvement in this paper, adaptive power model update strategies are adopted to *automatically* capture the major features related to power consumption. Hence, the human workload for power model training is reduced significantly.

As a summary, the features of current power monitoring solutions and our NIPD method are listed and compared in Table 1. To the best of our knowledge, our solution is unique since we do not find any existing solution falling in the same class, as shown in Fig. 2.

3 NIPD: Overview and Rationale

In this section, we clarify why we need non-intrusive power disaggregation (NIPD) in datacenters. Then, inspired by non-intrusive load monitoring for residential houses, we propose a new way to achieve NIPD in datacenters. We also overview the steps of NIPD.

3.1 Why NIPD?

To reduce the metering cost to zero, a software-based solution has to be adopted. As we have introduced in Sec. 2.1.2, a power model training process is needed for the software-based solution. The traditional intrusive model training is undesirable for legacy datacenters, since it is hard and tedious to obtain node-level power consumption data from highly integrated rack that is needed for model training. Therefore, a non-intrusive power model training scheme needs to be developed.

As shown in Fig. 1, electric power for datacenters is usually supplied via the uninterruptible power supply (UPS) and power distribution units (PDUs). With the help of UPS/PDUs, we can easily get access to the aggregated power consumption of the datacenter[1], from either the embedded meter [30] or certain interfaces (e.g., RS485 or RS232 serial interface) provided by the vendors. Such readily-available power readings, however, are aggregated power consumption from multiple racks of servers. To extract the fine-grained power information, we need to infer the power consumption of servers from the aggregated power readings, which is termed as power *disaggregation*.

In summary, we need non-intrusive training for building power model and power disaggregation to obtain the fine-grained power information. Hence NIPD comes naturally.

3.2 How to Develop NIPD?

The idea of separating aggregated power into individual units can be found in non-intrusive load monitoring (NILM) in residential houses [31], [32]. The NILM technology was initially proposed to separate the aggregated electricity consumption of a household into that of individual appliances. As it does not need any intrusive measuring in the house but only refers to the measurements from one meter outside the house, this technology has drawn much attention in energy conservation and demand response programs.

When applying existing NILM approaches in the datacenter environment, however, most assumptions in these approaches do not hold anymore. For example, servers do not turn on/off so often like household appliances. In addition, household appliances typically have their own power features, so-called appliances' signatures, which are used in many NILM solutions. This property is not obvious in datacenters for multiple servers can have exactly the same power ratings. To the best of our knowledge, no NILM solution developed for household power monitoring can be applied directly in datacenters.

Is there any way to revamp the NILM technology for datacenter environment? With embedded firmwares in the

1. In the rest of the paper, the power consumption of the datacenter refers to that of the IT facilities in particular. The power supply of others (e.g. cooling facilities) is out of the focus of this paper.

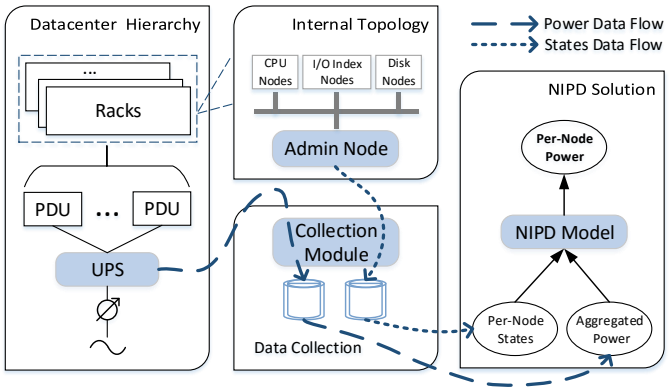


Fig. 3: Framework of non-intrusive power disaggregation over a datacenter.

server and easy-to-access interfaces, we can get the **component states**[2] information of a server. At a particular instant, the component states from different servers are most likely different, even when they are fed with the same type of workloads. This observation motivates us to utilize the distinguishability of component states from individual servers to achieve power disaggregation.

Our procedure of NIPD for a datacenter is shown in Fig. 3. As illustrated in the figure, we first collect aggregated power information and component states information of all servers through the collection module. Then, the two types of information are processed in non-intrusive power model training (the NIPD model in the figure). Finally, with the trained power model, we estimate the power consumption of each individual server.

While the above procedure is clear, the details on modeling, training and using the power model need further explanation. To start with, we formally formulate the problem and present solutions accordingly in Section 4.

4 Model Design for NIPD

In this section, we formally define the problem of NIPD for fine-grained power monitoring in datacenters and develop solutions for training and updating power models used in NIPD.

4.1 Formal Definition

Without loss of generality, we consider a datacenter consisting of m servers. We denote the aggregated power consumption of the m servers sampled in time interval $[1, t]$ by an *aggregated power vector* as:

$$y := [y_1, y_2, \cdots, y_t]^\intercal, \tag{1}$$

and we denote the power consumption of the i-th ($1 \leq i \leq m$) server in the same time interval, which is unknown, by an *individual power vector* as:

$$y^{(i)} := \left[y_1^{(i)}, y_2^{(i)}, \cdots, y_t^{(i)}\right]^\intercal. \tag{2}$$

2. A component state of a server in this paper refers to the instant index/value of one component's utilization or working speed, such as the CPU or memory utilization, I/O speeds, or hardware performance monitoring counters (PMCs).

For the purpose of NIPD, we use the state information of components collected from each server, which is recorded in a *state vector* containing the n scalars (n is the number of components whose information is available):

$$s := [\mu_1, \mu_2, \cdots, \mu_n]^\intercal. \tag{3}$$

Accordingly, the state vector of the i-th server at time j ($1 \leq j \leq t$) can be represented as:

$$s_j^{(i)} := \left[\mu_{1,j}^{(i)}, \mu_{2,j}^{(i)}, \cdots, \mu_{n,j}^{(i)}\right]^\intercal \tag{4}$$

in which $\mu_{\kappa,j}^{(i)}$ represents the value of the κ-th ($1 \leq \kappa \leq n$) component state in the i-th server at time instant j.

Definition 1. *During a time interval* [1, t]*, given the aggregated power vector y of m servers and each server's state vector $s_j^{(i)} \in \mathbb{R}^n, 1 \leq i \leq m, 1 \leq j \leq t$, the problem of* ***non-intrusive power disaggregation (NIPD)*** *is to estimate the power consumption of each individual server at each time instant, i.e., $y_j^{(i)}, 1 \leq i \leq m, 1 \leq j \leq t$.*

4.2 PMFs Modeling

To solve the NIPD problem, we first *logically* divide the servers in the datacenter into multiple **virtual homogeneous clusters (VHCs)**, so that in each VHC the major hardware components (e.g., CPU, memory, disk and NIC) of servers are the same or similar (i.e., the same brand and similar capacity). Thus, if a datacenter is composed by $r (r \geq 1)$ types of servers, we can divide the servers into r VHCs.

Definition 2. *We define a* ***state feature transformation (SFT)*** *$\phi : \mathbb{R}^n \to \mathbb{R}^{\tilde{n}}$, such that the original state vector $s \in \mathbb{R}^n$ can be transformed to a* ***dilated state vector*** *$x \in \mathbb{R}^{\tilde{n}}$, i.e.,*

$$x = \phi(s) \tag{5}$$

and the elements in x (named state features) can be nonlinear-form of individual component states and/or their combinations.

Example 1. *Given a state vector consisting of two elements, $s = [\mu_1, \mu_2]^\intercal$, one of the possible dilated state vectors can be constructed as $x = [1, \mu_1, \mu_2, \mu_1^2, \mu_1\mu_2]^\intercal$, where the SFT is defined as $\phi : \mathbb{R}^2 \to \mathbb{R}^5$.*

With SFT, the original feature space of component states will be extended. Accordingly, the dilated state vector of the i-th server at time j ($1 \leq j \leq t$) can be represented as:

$$x_j^{(i)} := \left[x_{1,j}^{(i)}, x_{2,j}^{(i)}, \cdots, x_{\tilde{n},j}^{(i)}\right]^\intercal \tag{6}$$

in which $x_{\kappa,j}^{(i)}$ represents the value of the κ-th ($1 \leq \kappa \leq \tilde{n}$) state feature for the i-th server at time instant j.

We set the first state feature as a unit constant 1, i.e., $x_1 = 1$ or $x_{1,j}^{(i)} = 1$, which is convenient for the following power model representation. Note that using a different constant leads to different coefficient values but has no impact on the final NIPD results.

Definition 3. *For servers in the same VHC, we define a* ***power mapping function (PMF)*** *$f : \mathbb{R}^{\tilde{n}} \to \mathbb{R}$, such that the input of a server's dilated state vector x at any time instant can yield its power consumption at corresponding time instant, i.e., for the i-th server's dilated state vector at time j, x_j, $f(x_j^{(i)})$ approximates $y_j^{(i)}$.*

According to the related works, both linear and nonlinear models have been explored to depict the relation between energy consumption of a server and its component states. As we have mentioned in Sec. 2.1.2, the linear power models are widely used and have shown good power estimations. Nevertheless, there are also evidences showing that nonlinear power models may perform better in some situations.

As illustrated in Fig. 4, a quadratic relationship between the CPU frequency and system power consumption can be observed under condition of high frequency or overclocking. Furthermore, the correlation among the server components' power consumption is usually ignored, as taking it into consideration can bring in nonlinear terms in the power model (e.g., the term of $\mu_1\mu_2$ in Example 1).

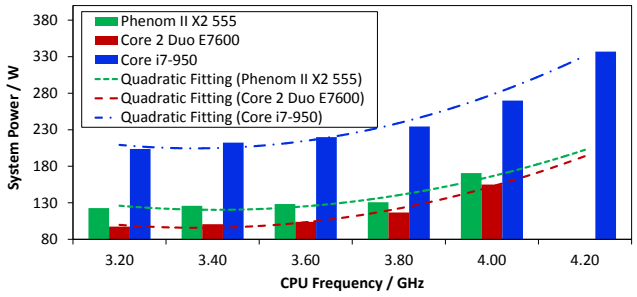


Fig. 4: Total system power consumption with respect to three categories of CPU at high-end working frequency [33]. As a power model to depict the relationship between CPU frequency and system power consumption, the quadratic function is fitting more tightly than the linear one.

Therefore, in this paper, we explore and compare the potentials of both linear and nonlinear power models by applying SFT in the PMFs modeling.

For servers in the same VHC, with the $\tilde{n}$-dimension dilated state vector x from a user-defined SFT in (5), we formulate their PMF as follows:

$$f(x) = w^{\intercal}x \tag{7}$$

where w is the *coefficient vector* denoted by:

$$w = [w_1, w_2, w_3, \cdots, w_{\tilde{n}}]^{\intercal}. \tag{8}$$

Example 2. *Given the state vector and SFT shown in Example 1, i.e., $x = [1, \mu_1, \mu_2, \mu_1^2, \mu_1\mu_2]^{\intercal}$, the PMF of the server can be formulated as:*

$$f(x) = w_1 + w_2 \cdot \mu_1 + w_3 \cdot \mu_2 + w_4 \cdot \mu_1^2 + w_5 \cdot \mu_1\mu_2 \tag{9}$$

where the coefficient vector $w = [w_1, w_2, \cdots, w_5]^{\intercal}$.

Note that as SFT can be either linear or nonlinear transformation of the state vector, the PMF in (7) thus can in an either linear or nonlinear form w.r.t. component states of each server in a VHC.

Remark 1. *Conventional methods try to build a power model for each major component in a server, which is used to estimate the power consumption of each component. The server's power consumption is approximated by the aggregate of the estimated power consumption of its major components. Our PMF can be regarded as a special type of power model, but it is different from conventional ones in that our PMF just indicates a way of mapping the dilated state features to the server's overall power consumption. The power of uncovered components, e.g., fans within the server enclosure, will be properly "absorbed" (in the sense that $f(x_j^{(i)})$ best approximates $y_j^{(i)}$) by the modeled terms in PMF. Hence, the value of each term in PMF is not necessarily the true power value.*

4.3 PMFs Training

For the overall power consumption of a server $f(x)$, it can be broken down into two parts: idle power (or static power) and dynamic power [34]. The former is considered as the baseline power supplied to maintain the server system in an idle state, and the latter is the additional power consumption for running specific workloads.

As we have set the first state feature in the dilated state vector as a unit constant (i.e., $x_1 = 1$), the first term in the PMF model (e.g., w_1 in Example 2) represents a constant and reflects the idle power. The left coefficients $w_2, w_3, \cdots, w_{\tilde{n}}$ are thus associated with the dynamic power of a server system.

4.3.1 Estimation of Coefficients

We first estimate the coefficients of a server's PMF. Assume that a datacenter consists of r VHCs, and m_κ servers are in the κ-th ($1 \le \kappa \le r$) VHC. Moreover, each server of the κ-th VHC reports a n_κ-dimensional state vector, and with a user-defined SFT, the state vector is transformed to a $\tilde{n}_\kappa$-dimensional dilated state vector x. Then, with the dilated state vector, the PMF for the VHC can be expressed as:

$$f_\kappa(x) = (w^{(\kappa)})^{\intercal}x \tag{10}$$

where $w^{(\kappa)}$ is the coefficient vector of PMF for the κ-th VHC denoted as:

$$w^{(\kappa)} = \left[w_1^{(\kappa)}, w_2^{(\kappa)}, \cdots, w_{\tilde{n}_\kappa}^{(\kappa)}\right]^{\intercal}. \tag{11}$$

At an arbitrary time instant j, the aggregated power consumption of the κ-th VHC can be expressed as: $\hat{y}_j = (w^{(\kappa)})^{\intercal}\hat{x}_j^{(\kappa)}$, where:

$$\hat{x}_j^{(\kappa)} = \left[\sum_{i=1}^{m_\kappa} x_{1,j}^{(i)}, \sum_{i=1}^{m_\kappa} x_{2,j}^{(i)}, \sum_{i=1}^{m_\kappa} x_{3,j}^{(i)}, \cdots, \sum_{i=1}^{m_\kappa} x_{\tilde{n},j}^{(i)}\right]^{\intercal}. \tag{12}$$

Meanwhile, the aggregated power consumption of the whole datacenter (or r VHCs) can be expressed as: $y_j = \tilde{w}^{\intercal}\tilde{x}_j$, where:

$$\tilde{x}_j = \left[\hat{x}_j^{(1)}, \hat{x}_j^{(2)}, \cdots, \hat{x}_j^{(r)}\right]^{\intercal}, \tag{13}$$

and

$$\tilde{w} = \left[w^{(1)}, w^{(2)}, \cdots, w^{(r)}\right]^{\intercal}, \tag{14}$$

in which $\hat{x}_j^{(\kappa)}$ and $w^{(\kappa)}$ are defined by (12) and (11), respectively. (Refer to Sec. 1 of supplementary materials for detailed transformations of the above equations.)

With the measured aggregated power vector of the whole datacenter y (in form of (1)), the following least square estimation (LSE) problem is formulated as the training model for the r PMFs of the datacenter:

$$\min_{\tilde{w}} \quad \sum_{j=1}^{t} \left(\tilde{w}^{\intercal}\tilde{x}_j - y_j\right)^2. \tag{15}$$

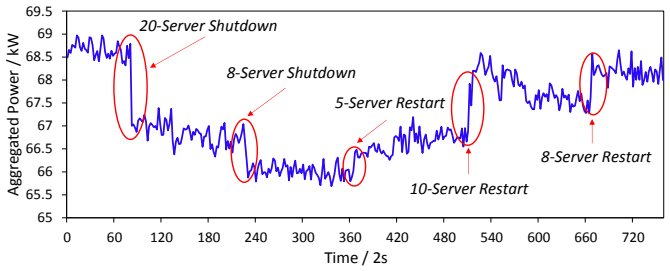


Fig. 5: On/off events captured during turning on/off servers in our datacenter consisting of 326 servers.

By solving the above problem, we can obtain the optimal coefficients for the r PMFs appearing in $\tilde{w}$, with which we can estimate the power consumption of individual servers in different VHCs by providing corresponding dilated state vectors.

Remark 2. *The LSE problem represented in (15) belongs to linear regression, and to solve the problem is trivial (with closed form given in Sec. 3 of the Supplementary Materials). Therefore, rather than the way to solve problem (15), it is the way to represent and formulate the NIPD into a solvable form that is the major innovation of this work.*

Nevertheless, the above LSE training model cannot capture multiple but only one constant term appearing in the coefficient vector [28]. Consequently, if there are more than one VHC in the datacenter ($r > 1$), the resulted constant terms (i.e., $w_1^{(1)}, w_1^{(2)}, \cdots, w_1^{(r)}$) from (15) are not accurate. In other words, the idle power of servers in each VHC cannot be estimated by this model. Therefore, further approaches need to be developed to estimate the constant terms in PMFs.

4.3.2 Estimation of Constant Terms

A widely used energy saving strategy in many datacenters is to shutdown idle servers. They will be turned on again when the working servers cannot satisfy the workload [35], [36], [37]. Such a strategy provides us with an opportunity to estimate the constant terms in PMFs.

Definition 4. *For a datacenter with r VHCs, at an arbitrary time instant j, if h servers are turned off (or on), and meanwhile a power decrease (or increase) in the aggregated power consumption of the whole datacenter, $\Delta y (\Delta y > 0)$, is detected, we call that an **off/on event** is captured. We use $\Delta y > 0$ to indicate that only the absolute value is considered in our late problem formulation.*

According to our real-world experiments illustrated in Fig. 5, although the aggregated power of the whole datacenter always fluctuates over time, we are still able to capture the off/on events without turning on/off a large proportion of servers.

Assume that t off/on events have been captured in the datacenter consisting of r VHCs. For the j-th ($1 \leq j \leq t$) off/on event, a *counting vector* can be defined as:

$$d_j := \left[d_j^{(1)}, d_j^{(2)}, \cdots, d_j^{(r)}\right]^\intercal, \tag{16}$$

where $d_j^{(\kappa)}$ stands for the number of turned-off (or turned-on) servers in the κ-th VHC at time j, and the detected (mean) power decrease (or increase) is Δy_j. Then the following optimization problem can be formulated to find the optimal estimation of the constant terms, i.e., $w_1 = [w_1^{(1)}, w_1^{(2)}, \cdots, w_1^{(r)}]^\intercal$:

$$\min_{w_1} \quad \sum_{j=1}^{t} \left(w_1^\intercal d_j - \Delta y_j\right)^2. \tag{17}$$

Remark 3. *In the estimation of constant terms of PMFs, we can combine the optimization strategy using (17) and the manual setup with information from technical specification of servers. For servers that can be shut down, e.g., the computing nodes, it is easy to gather off/on events and estimate the idle power via the optimization method. For other IT units that cannot be shut down during the operation of datacenter, e.g., the admin nodes, the best way is to refer to the server's technical specification or approximate their idle power using the information from other servers equipped with similar hardware components.*

So far, we have introduced the details of building PMFs. The datacenter operators can then use PMFs to estimate the real-time power consumption of individual servers by referring to real-time component states from corresponding servers.

4.4 Adaptive PMFs Update

According to the analysis in Sec. 3 of supplementary materials, the complexity of PMFs training relies on two metrics: the number of training data and the number of state features. Specifically, the training complexity has a linear growth with the increase of training data and a quadratic growth with the increase of state features. Therefore, it is critical to choose an appropriate number of training data as well as state features, so that the training process is lightweight while the resulted PMFs is accurate enough.

4.4.1 Selective Training Data Collection

To make PMFs as accurate as possible, we need a training dataset that contains complete states in the feature space, i.e., all possible state features of the servers in each VHC should be included in the training dataset. Nevertheless, in real-world datacenter operations, it is hard to stress each of the components in a server to work through all possible states. The training dataset collected in a time interval of several hours or even several days may be incomplete. In other words, there is no guarantee that the training dataset covers all possible state features.

Simply collecting training data as much as possible, however, is not a good solution to the above problem due to two reasons: (1) the larger the training dataset, the higher the overhead in PMF training, and (2) more redundant data entries will be collected while they do not contribute to the improvement of PMFs. Therefore, we develop a selective data collection strategy as follows.

Preprocessing: we first set an update time interval for the training dataset, denoted as Δt_1. At an arbitrary time instant j, the dilated state features from r VHCs can be expressed as $\tilde{x}_j$ in form of (13). Along with the measured aggregated power consumption of the datacenter at the same moment y_j, a data entry can be represented as $(\tilde{x}_j, y_j)$.

With data entry of $(\tilde{x}_j, y_j)$, the process of selective training data collection is as below:

- *Step 1.* normalize each element in $\tilde{x}_j$ with the corresponding maximum value[3], i.e., rescale the values of each element to $[0, 1]$.
- *Step 2.* compare the normalized data entry with those in the training dataset. if it already exists, go to *Step 4.*; otherwise, go to *Step 3*.
- *Step 3.* insert $(\tilde{x}_j, y_j)$ into the training dataset as a new entry.
- *Step 4.* backup the power value y_j for the existed entry with the same component states.

Note that in the fourth step, we do not simply discard the redundant entry, but keep record of its power value. Thus, one data entry in the training dataset may have multiple power values, and we calculate their median as the final value used for PMFs training. Using the median can alleviate the affect of outliers [38] and make the PMFs' training more robust. In addition to the gathering of state features, the same strategy can also be applied to collect the off/on events for constant terms estimation introduced in Sec. 4.3.2.

Remark 4. *For our selective data collection strategy, the resolution of the normalized state features determines the maximum number of data entries in the training dataset. Assuming that a datacenter consists of r $(r \geq 1)$ VHCs, each with $\tilde{n}_\kappa$ $(1 \leq \kappa \leq r)$ state features, and the preset resolution of normalized state features is p $(0 < p \ll 1)$, then the number of data entries in the training dataset is upper-bounded by $\sum_{\kappa=1}^{r} \left\lceil \frac{1}{p^{n_\kappa}} \right\rceil$. Refer to Sec. 2 of supplementary materials for the proof.*

We can update training dataset at a regular basis, e.g., every Δt_1 interval time. Theoretically, with the above data collection strategy, the training dataset will become complete as time goes on. Meanwhile, we use the most updated dataset to perform PMFs training introduced in Sec. 4.3 at a regular basis, e.g., every Δt_2 interval time.

4.4.2 Iterative State Feature Elimination

There are multiple tools that can be used to collect system component states of a server, e.g., the hardware performance monitoring counters (PMC) and `dstat` tool [39]. According to [40], as many as 86 PMC states were selected from all available ones. As to the `dstat` tool that will be used in our experiments, it can provide up to 16 different states. Simply gathering all component states provided by these tools can result in a large state vector. Furthermore, with our state feature transformation introduced in Sec. 4.2, the dilated state vector for PMFs training will be even larger. To control the overhead in model training, we need to limit the number of state features, especially when the number of training data entries is already huge.

So far, it is still challenging to choose the most relevant and effective state features to construct the power model. Current approaches (like principal component analysis (PCA) [41]) usually falls into the supervised learning and thus need intrusive ground-truth measuring. Since intrusive measuring is not feasible in our context, a new state feature selection approach is needed.

To deal with this problem, we propose and perform an iterative state feature elimination process, which does not need supervised learning and can significantly reduce the irrelevant state features in PMFs. The process is shown by following steps:

- *Step 1.* Initially choose potential state features from the the raw data, and discard the irrelevant or redundant state features, e.g, the CPU idle and CPU utilization are actually equivalent and one of them can be discarded.
- *Step 2.* Apply the left state features to formulate the preliminary PMFs, and then conduct PMFs training following the routine introduced in Sec. 4.3 when selected training dataset is small.
- *Step 3.* When the number of training data entries exceeds a pre-defined threshold value n_δ, we check the coefficient of each state feature: if its absolute value is approximate to zero, e.g., less than 1×10^{-3} in our implementation, eliminate this term in the corresponding PMF.

The pseudocodes of selective training data collection and iterative state feature elimination are illustrated in Algorithm 1 and Algorithm 2, respectively. The two processes that cooperate together can achieve adaptive PMFs updating, with which we can significantly reduce the PMFs training overhead. As explained in Remark 4, the number of the training data entries using selective data collection is not large (less than $10K$ in our latest experiment). Furthermore, by applying iterative state feature elimination, we will show that a small number of state features can be resulted and sufficient to provide accurate PMFs in Sec. 6.

Overall, to recap our NIPD solution introduced in this section: at background the PMFs are adaptively updated by selective training dataset collection and iterative state features elimination; at foreground the real-time component

Algorithm 1 Selective Training Data Collection

Require: Sampling interval Δt_0, training dataset update interval Δt_1
Ensure: Training dataset $\mathcal{D}$

```
D ← ∅
while True do
    t ← gettime()
    if t mod Δt1 == 0 then
        for j ← 1 : Δt0 : Δt1 do
            compose the data entry (x̃j, yj)
            normalize each element in x̃j
            if x̃j ∉ D then
                D ← D ⋃ (x̃j, yj)
            else
                backup power value yj
            end if
        end for
    end if
end while
```

3. The maximum value could be found from technical specification, e.g., maximum I/O speed, or if unknown, it could be set as a value higher than any possible values of the state feature.

Algorithm 2 Iterative State Feature Elimination

Require: Dilated state vector x, PMFs update interval Δt_2, training dataset $\mathcal{D}$, elimination threshold n_δ
Ensure: Updated PMFs

```
initialize preliminary PMFs with x
while True do
    t ← gettime()
    if t mod Δt2 == 0 then
        if |D| < nδ then
            update PMFs coefficients with D
        else
            reformulate PMFs: eliminate state features with coefficients approximate to zero
        end if
    end if
end while
```

state information is fed into the most updated PMFs to obtain the server-level power estimations.

5 Implementation

We implement our NIPD solution over a real-world 326-node server cluster. It consists of 12 (blade) server racks that house 306 CPU nodes, 16 disk array nodes, 2 I/O index nodes, and 2 admin nodes, each running a Linux kernel. Table 2 shows the detailed configuration of each type of server. Fig. 6 illustrates the power architecture, network topology, and data collection modules of the experimental environment.

TABLE 2: Configuration of Server Nodes

Type	*Configurations*	*Num.*
CPU Node	2×Intel Xeon E5-2670 8-core CPU(2.6G) 8×8GB DDR3 1600MHz SDRAM 1×300G 10000rpm SAS HDD	306
Disk Array Node	1×Intel Xeon E5-2603 4-core CPU(1.8G) 4×4GB DDR3 ECC SDRAM 1×300G 10000rpm SAS HDD 36×900G SAS HDD Networking Switches	16
I/O Index Node	2×Intel Xeon E5-2603 4-core CPU(1.8G) 8×4GB DDR3 ECC SDRAM 1×300G 10000rpm SAS HDD	2
Admin Node	2×Intel Xeon E5-2670 8-core CPU(2.6G) 8×16GB DDR3 1600MHz SDRAM 1×300G 10000rpm SAS HDD	2

5.1 Data Collection and Feature Elimination

As shown in Fig. 6, we collect aggregated power consumption of the IT infrastructure via the UPS interface and a power monitoring proxy (P-1 in the figure). The sampling interval is 2 seconds. Besides the UPS, our datacenter is further equipped with 6 Power Data Management Modules (PDMMs) as part of the PDUs, each of which can provide power measuring at the rack-level, also at the sampling interval of 2 seconds. To verify our power estimation at the rack-level in Sec. 6.2, we also collect the power consumption of each rack via corresponding PDMM using the rack proxies (P-2 in Fig. 6).

In addition to the collection of power consumption data, the admin node is used to collect the component state information from each node (with sampling interval of 1 second).

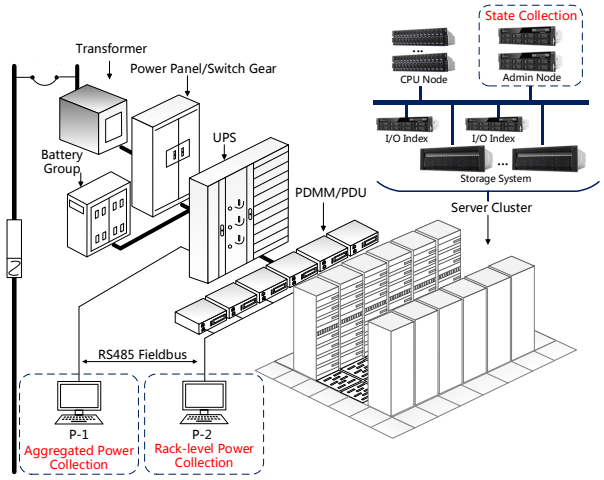


Fig. 6: Power architecture, network topology and data collection modules in our experimental environment. Note that the rack-level power measurements provided by PDMMs are only used for validation purpose in this paper.

TABLE 3: State metrics collected using `dstat` tool

Component	*Label*	*Description*
processor	`usr`	CPU utilization for user processes
	`sys`	CPU utilization for system processes
	`idle`	CPU in idle
	`wai`	CPU utilization for I/O waiting
memory	`used`	memory usage for processes
	`buff`	buffer memory
	`cach`	cache memory
	`free`	free memory
disk	`read`	disk reading amount
	`write`	disk writing amount
network	`recv`	traffic amount that the system received
	`send`	traffic amount that the system sent
paging	`in`	# pages changed from disk to memory
	`page`	# pages changed from memory to disk
system	`int`	system interruption time
	`csw`	content switch times

Particularly, we use `dstat` tool [39], a widely-used resource statistic tool that can gather various component states of a server, as shown in Table 3. Note that other tools can also be used here, such as `vmstat`, `iostat`, `mpstat` and `netstat`. To ensure the time synchronization between the component state collection and aggregated power collection, we set their clock source the same as Time Stamp Counter (TSC).

We first applied all the states terms listed in Table 3 to establish the preliminary PMFs. Then, we adopted the process of state feature elimination introduced in Sec. 4.4.2 and iteratively reduced the number of state features provided by `dstat` to six: total CPU utilization, total memory utilization, disk reading/writing and network traffic receiving/sending. With the six component states, according to (3), the state vector can be represented as:

$$s = [\mu_1, \mu_2, \ldots, \mu_6]^\intercal . \tag{18}$$

When constructing nonlinear PMFs, we first included all second-order items with states in (18). Then, the state feature elimination was applied to minimize the dilated feature space. At last, only the nonlinear terms of μ_1^2 and $\mu_1\mu_2$

left, which capture the quadratic trend shown in Fig. 4 and the correlation between CPU utilization and memory usage, respectively.

In real-world datacenter environment, there are moments of component failures, e.g., some servers or ToR switch may be down. Under this situation, the component states of some servers may not be available. To alleviate the impact of component failures and increase the reliability, we can take the following two measurements when implementing NIPD:

- During the PMFs training, special caution should be taken to make sure that only the data entries containing the state information of all servers are inserted into the training dataset. In this way, we guarantee that the trained PMFs are not affected by component failures.
- Once PMFs are obtained, we only estimate the power of servers whose component states are available. For servers whose component states are not accessible, we should not assign them zero power. Instead, we do not estimate their power and correspondingly raise a warning message to the administrator.

5.2 Estimation of Idle Power

For the estimation of idle power (or constant terms in PMFs) of CPU nodes in our experiments, we identify the idle nodes and remotely turn them off and on. For purpose of remote operation, the industry-standard IPMI [42] is used to turn on/off servers. During the on/off time period, multiple off/on events and corresponding power changes are captured from the event logs and data logs, respectively (as illustrated in Fig. 5), which are fed into the optimization model (17) to estimate the constant terms (idle power) of CPU nodes. As for the estimation of idle power of I/O and admin nodes, they are not allowed to be shut down for the normal operation of a running datacenter. Since the number of these two server types is quite small (only 2 for each type) and their hardware configurations are similar with that of CPU nodes, we set their idle power as the same as that of CPU nodes. The disk array nodes also need to be kept on all the time. Thus we infer their idle power from their working power range by making use of rack power: in our experimental datacenter, some racks only contain CPU nodes and disk arrays, so we can shut down all the CPU nodes and only leave the disk arrays running to obtain the idle power.

6 Evaluation

In our experimental environment introduced in Sec. 5, we evaluate the performance of different (linear and nonlinear) PMFs for power monitoring at the rack level and the server level, respectively.

6.1 Experiment Configuration

Table 4 summarizes the setting of parameters in our experiments. We setup the parameters based on the following considerations:

- Number of VHCs (r): According to Table 2, we can logically divide the whole datacenter into 4 VHCs, and the number of servers in each VHC is 306, 16, 2, 2, respectively.
- Number of component states (n_κ): As introduced in Sec. 5.1, we select 6 component states for each individual server based on the information provided by Table 3.
- Dilated state vectors (x): Based on the original state vector of (18), we establish one linear PMFs and two nonlinear PMFs via the SFT showing in the table. Specifically, for PMFs with one nonlinear term (i.e., μ_1^2), we consider the quadratic trend shown in Fig. 4; for the PMFs with two nonlinear terms (i.e., μ_1^2 and $\mu_1\mu_2$), besides the quadratic trend, we also consider the correlation between CPU utilization and memory usage.
- Normalizing resolution (p): In the update of training dataset, we set the resolution of normalized data in each entry as 0.01, which is proved to be precise enough for accurate PMFs training, as shown in Sec. 6.2. According to Remark 4, a higher resolution will increase the size of training dataset as well as PMFs training complexity.
- Interval for updating training dataset (Δt_1): As the sampling interval for aggregated power consumption in our case is 2 seconds, we set the update interval of training dataset to the same value in order to collect training data quickly.
- PMFs update interval (Δt_2): We first set this value as 5 minutes, which is based on the estimation of PMFs training time needed under the theoretical maximum size of training dataset. As time goes, having observed that the training dataset size tends to be constant, we then change the update interval to 0.5 hour to reduce the overhead of PMFs update.
- Elimination threshold (n_δ): When the number of training data entries exceeds $8,000$, the resulted PMFs coefficients tend to be accurate (with mean relative error less than 15%), which can be verified by Fig. 7a.

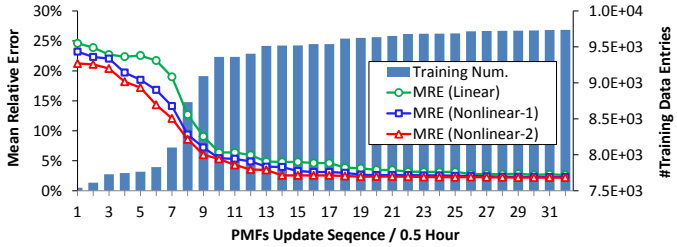
TABLE 4: Parameter settings of our experiments

Parameter		*Setting*
Number of VHCs (r)		4
Number of component states (n_κ)		[6, 6, 6, 6]
Dilated state vectors (x)	Linear	$[1, s]^\intercal$
	Nonlinear-1	$[1, \mu_1^2, s]^\intercal$
	Nonlinear-2	$[1, \mu_1^2, \mu_1\mu_2, s]^\intercal$
Normalizing resolution (p)		0.01
Training dataset update interval (Δt_1)		2 seconds
PMFs update interval (Δt_2)		5 minutes, 0.5 hour
Elimination threshold (n_δ)		$8,000$

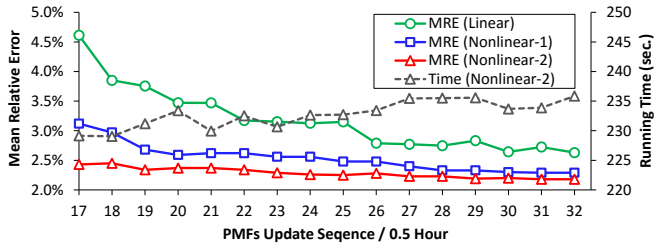
*The s in dilated state vectors refers to the state vector defined by (18).

6.2 Power Monitoring at Rack Level

Putting the real-time component state information of the servers into the corresponding PMFs, we can get the estimated power consumption of each server. The estimated power consumption of servers in the same rack are aggregated as the estimated power consumption of the rack. To measure the error rate of our rack-level estimation, we apply

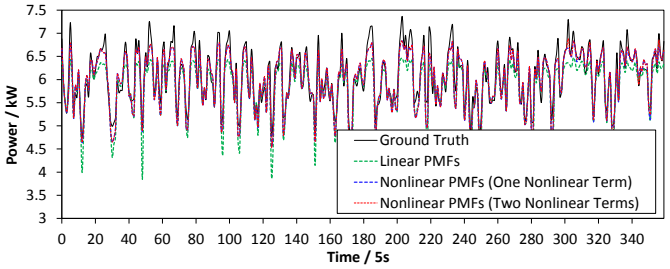


(a) MRE vs. number of training data entries.

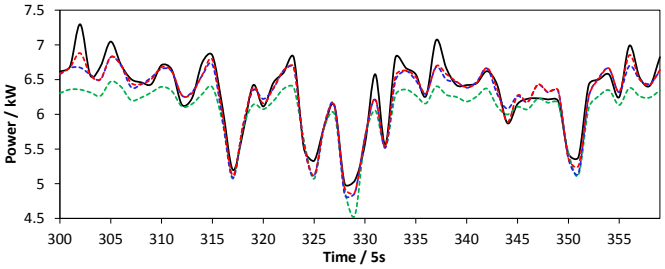


(b) MRE ($\leq 5.0\%$) and running time of solving Nonlinear-2.

Fig. 7: The overview of MRE from three PMFs along with the training data size (left) and that of a zoomed-in view when MRE $\leq 5.0\%$ (right).



(a) A Global View



(b) A Local View

Fig. 8: The estimated power of one rack with corresponding ground truth values: a global view (left) and a local view (right).

TABLE 5: Workloads/benchmarks for NIPD evaluations

Workload		*Description*	*Purpose*
Idle		Background OS processes	Server-level validation
Peak		Stress CPU usage to 100% malloc memory till 100%	
SPECint	`gcc`	Compiler	Training data collection and PMFs update
	`gobmk`	Artificial Intelligence: go	
	`sjeng`	Artificial Intelligence: chess	
	`omnetpp`	Discrete Event Simulation	
SPECfp	`namd`	Biology/Molecular Dynamics	
	`wrf`	Weather Prediction	
	`tonto`	Quantum, Chemistry	
IOZone		File system benchmark tool	
Our Synthetic		Occupy CPU randomly Read/write memory randomly	Rack-level validation

the widely used metric of *mean relative error (MRE)* defined by:

$$\text{MRE} := \frac{1}{t}\sum_{j=1}^{t}\left|\frac{y'_j - y_j}{y_j}\right| \tag{19}$$

where t is the number of data entries in the dataset, and y_j and y'_j are the ground truth and estimated rack power for the j-th data entry, respectively.

By running different benchmarks shown in Table 5, we collect training data and duly update PMFs following the strategies introduced in Sec. 4.4. Furthermore, after each PMFs update, we run our synthetic workloads, collect rack power consumption and server component states, and calculate MRE of the power estimation with updated PMFs.

The results are summarized in Table 6 (column 2-4) and illustrated in Fig. 7, respectively. From the rack level results in Table 6, we can see that the two nonlinear PMFs slightly outperform the linear PMF. According to Fig. 7a, the MRE of power estimation from the three PMFs monotonically decreases with the increase of training dataset, and tends to be stable at the value strictly smaller than 5%. As a zoomed-in view, Fig. 7b illustrates the MRE of three PMFs when MRE $\leq 5.0\%$, in which the MRE from nonlinear PMFs decreases faster than that from the linear PMFs.

To illustrate the performance of three PMFs more clearly, their power estimations for a random rack along with the ground truth values (in 0.5 hour) are shown in Fig. 8. A global view in Fig. 8a and a local view in Fig. 8b are shown, from which we can see that the nonlinear PMFs provide more tight fittings for the ground truth than the linear PMFs.

Note that the power estimation shown in Fig. 8 is slightly lower than the ground truth values. This may cause a risk for peak power monitoring applications like rack-level power capping. To reduce the risk of the power underestimation, we suggest that the datacenter operators set the redline threshold, i.e., pre-defined peak power that triggers management actions, slightly lower than the expected one, when applying NIPD for rack-level power monitoring. As the power estimation at rack level is relatively accurate (with MRE $< 3\%$ in our case), a small magnitude (e.g., 5%) drop of the redline threshold can much reduce the risk of power overflow.

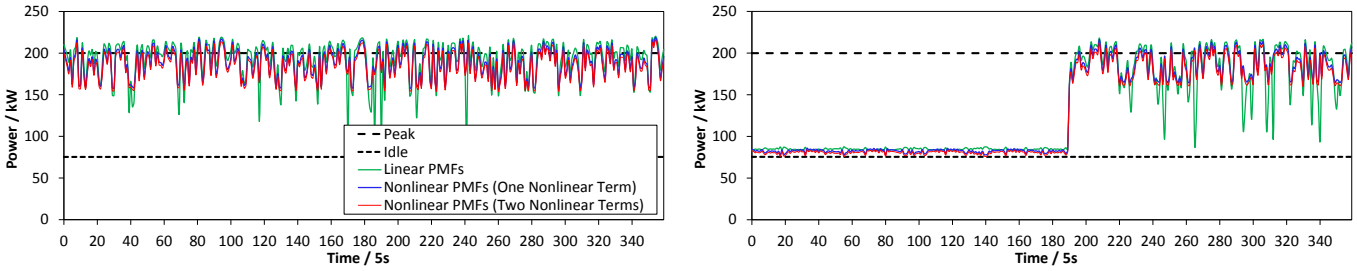
6.3 Power Monitoring at Server Level

It is hard to fully validate the accuracy of our estimation at server level, because the power consumption of individual servers in our experimental environment is hard to be obtained. As the (blade) servers are highly integrated in the rack, e.g., fourteen 4U CPU nodes are tightly packaged in one row, it is difficult to assemble sensors/meters *inside* individual servers. In addition, multiple servers may share the same power supply, e.g., the fourteen 4U CPU nodes share only four power suppliers, so it is also hard to obtain server-level power *outside* the servers.

TABLE 6: Performance of linear and nonlinear PMFs for power estimations at rack level and server level

Monitoring Level	**Rack Level**			**Server Level**		
PMFs Type	Linear	Nonlinear-1	Nonlinear-2	Linear	Nonlinear-1	Nonlinear-2
Mean Relative Error (*Power Disaggregation of Datacenter*)	2.63%	2.29%	2.18%	10.27% / 8.17%	9.43% / 7.70%	9.61% / 7.53%
Mean Relative Error (*Power Disaggregation of Racks*)	–	–	–	6.92% / 6.30%	5.97% / 5.95%	6.03% / 5.91%

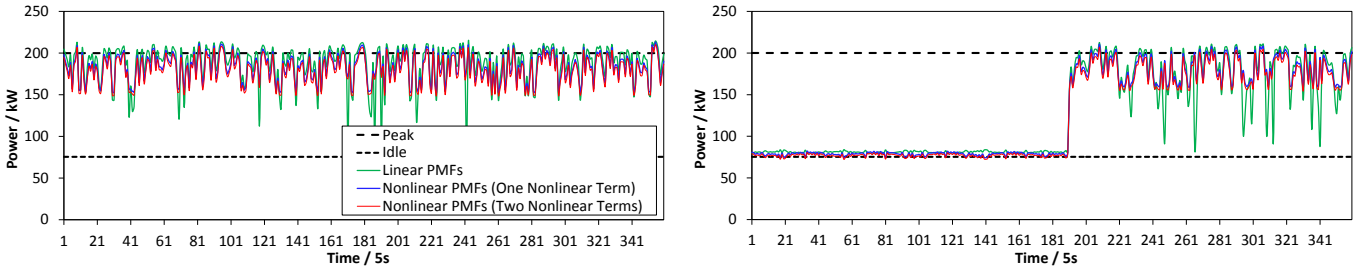
*The two values of MRE at the server level are corresponding to the idle power and peak power, respectively.



(a) Peak Power Estimation

(b) Idle/Peak Power Estimation

Fig. 9: Disaggregating datacenter power: estimated power consumption vs. referred idle/peak power.



(a) Peak Power Estimation

(b) Idle/Peak Power Estimation

Fig. 10: Disaggregating rack power: estimated power consumption vs. referred idle/peak power.

Even though we cannot record ground truth power consumption for individual nodes, we do have a knowledge about the *idle power*, *peak power* or *working power range* of each server type. We focus on the power estimation of the CPU nodes, which are dominated in our datacenter (306 out of 326 servers). Their idle power and peak power can be either estimated via the process introduced in Sec. 4.3.2 or learned by referring to the nameplate power provided by the server vendor. Then, we use the measured idle and peak power values as references to evaluate our server-level power estimation.

6.3.1 Power Disaggregation of Datacenter

Using the PMFs trained from the aggregated power readings of IT facilities, we estimate the real-time power consumption of individual servers. To illustrate the performance, we choose four CPU nodes (Node-25 to Node-28) among our datacenter as test nodes. Moreover, we make two of them run our peak workload (listed in Table 5), and the other two firstly keep idle for 15 minutes and then run peak workload for another 15 minutes.

Along with corresponding referred power bounds, the resulted power estimations for two of the CPU test nodes are demonstrated in Fig. 9 (the situations of the other two nodes are similar). From the results we can see that, both the estimated idle/peak power[4] from the three PMFs are close to the referred power bounds, with the nonlinear PMFs slightly outperform the linear PMFs. The overall performance is summarized in Table 6 (the last three columns). By checking the performance under these two extreme cases, we can validate the effectiveness of our solution.

We have also observed that the estimated power values are slightly larger than the referred ones, as shown in Fig. 9. This is because when disaggregating the datacenter power, the power loss during the transmission (e.g., by wire and PDUs) as well as power consumed by interconnection network (e.g., network switches, line cards, and datacenter accessories) are assigned to individual servers, as discussed in Remark 1.

6.3.2 Power Disaggregation of Racks

When a datacenter is capable of monitoring power consumption of each rack, our NIPD can be used to disaggregate the rack-level power consumption into server-level power consumption. As the servers in a rack are usually homogeneous, we can set the number of VHCs as one, and in this case the computational complexity for training PMFs will be much lower than that in a heterogeneous environment (refer to Sec. 3 of supplementary materials).

In our datacenter, we choose a test rack which contains 28 CPU nodes (Node-1 to Node-28) and 2 I/O index nodes. Since the number of CPU nodes is much larger than that of the I/O index nodes and the CPU nodes' working power ranges are very similar, this rack is approximately homoge-

4. Peak power values refer to the power readings when the CPU utilization is 100%.

neous. The collected historical data from this rack are used for PMF training, and the updated PMF is used to make estimation under idle/peak workloads for individual servers in this rack. The resulted idle/peak power estimation of the same test nodes in Sec. 6.3.1 is illustrated in Fig. 10.

According to the server level results shown in Table 6 and the comparison between Fig. 10 and Fig. 9, we can find that the server-level power estimation by disaggregating the rack power is better than that from disaggregating the entire datacenter power. This is because the impact of hardware components not modeled in PMFs is smaller at the rack level than at the whole datacenter level, as per the discussion in Remark 1.

As a concluding suggestion, if the datacenter operators can obtain rack-level power information, it would improve the accuracy of NIPD estimation to directly disaggregate the rack-level power than to disaggregate the power of the entire datacenter.

7 Conclusions

In this paper, we defined the problem of non-intrusive power disaggregation (NIPD) for legacy datacenters, and developed a software-based approach for power estimation of individual servers. A non-intrusive training procedure was proposed to find the power mapping functions (PMFs) between the states of servers and their power consumption. With linear or nonlinear state feature transformation, the PMFs can represent linear or nonlinear power models. To effectively improve the precision of PMFs as well as lower the training overhead, we adopted adaptive PMFs update strategies of selective training data collection and iterative state feature elimination. Based on the updated PMFs with servers' running state information, the power consumption of individual servers can be estimated in real-time by only referring to the aggregated power of the entire datacenter. Our solution introduced no hardware cost and incurred no interruption to the running servers. The experimental results over a 326-node datacenter showed that our solution can provide precise power estimation at both the rack level and the server level. For example, with the nonlinear PMFs including two nonlinear terms, the mean relative error of our power estimations can reach 2.18% at rack level, and 9.61% and 7.53% at server level with respect to the idle power and peak power.

Acknowledgements

This work was partially supported by the Mitacs Globalink Research Award, the Natural Sciences and Engineering Research Council of Canada (NSERC) under grant No.195819339, the National Natural Science Foundation of China (NSFC) under grants No.61373152 and No.61520106005, and the National 973 Basic Research Program under grant No.2014CB347800. The corresponding author is Fangming Liu.

References

[1] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs, "Cutting the electric bill for internet-scale systems," in *Proceedings of SIGCOMM*. ACM, 2009.

[2] NRDC, "Natural resource defense council report: America's data centers are wasting huge amounts of energy," http://www.nrdc.org/energy/files/data-center-efficiency-assessment-IB.pdf, [Online; accessed in 12-May-2015].

[3] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam, "Managing server energy and operational costs in hosting centers," in *Proceedings of SIGMETRICS*. ACM, 2005.

[4] V. Petrucci, O. Loques, and D. Mossé, "A dynamic optimization model for power and performance management of virtualized clusters," in *Proceedings of International Conference on Energy-Efficient Computing and Networking (e-Energy)*. ACM, 2010.

[5] C. Lefurgy, X. Wang, and M. Ware, "Server-level power control," in *Proceedings of International Conference on Autonomic Computing (ICAC)*. IEEE, 2007.

[6] ——, "Power capping: a prelude to power shifting," *Cluster Computing*, 2008.

[7] D. Meisner, B. T. Gold, and T. F. Wenisch, "Powernap: eliminating server idle power," in *Proceedings of ASPOLOS*. ACM, 2009.

[8] C. Bash and G. Forman, "Cool job allocation: Measuring the power savings of placing jobs at cooling-efficient locations in the data center." in *Proceedings of USENIX Annual Technical Conference (ATC)*. USENIX, 2007.

[9] UptimeInstitute, "Data center industry survey," 2012.

[10] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *Proceedings of ISCA*. ACM, 2007.

[11] D. Wang, C. Ren, and A. Sivasubramaniam, "Virtualizing power distribution in datacenters," in *Proceedings of ISCA*. ACM, 2013.

[12] Ponemon Institute, "2013 study on data center outages," http://www.emersonnetworkpower.com/documentation/en-us/brands/liebert/documents/white%20papers/2013_emerson_data_center_outages_sl-24679.pdf, 2014, [Online; accessed in 15-May-2015].

[13] SynapSense, "Power monitoring," http://www.synapsense.com/?page_id=22, 2015, [Online; accessed in 13-March-2015].

[14] P. Popa, "Managing server energy consumption using IBM PowerExecutive," *IBM Systems and Technology Group, Tech. Rep*, 2006.

[15] Noble Vision Group, "Energy and eco friendly data center solutions," http://www.noblevisiongrp.com/, 2015, [Online; accessed in 21-April-2015].

[16] ServerTechnology, "Server technology products," https://www.servertech.com/products, 2015, [Online; accessed in 21-April-2015].

[17] X. Feng, R. Ge, and K. W. Cameron, "Power and energy profiling of scientific applications on distributed systems," in *Proceedings of International Parallel & Distributed Processing Symposium (IPDPS)*. IEEE, 2005.

[18] R. Ge, X. Feng, S. Song, H.-C. Chang, D. Li, and K. W. Cameron, "Powerpack: Energy profiling and analysis of high-performance systems and applications," *Parallel and Distributed Systems, IEEE Transactions on*, 2010.

[19] D. Economou, S. Rivoire, C. Kozyrakis, and P. Ranganathan, "Full-system power analysis and modeling for server environments," in *Proceedings of Workshop on Modeling Benchmarking and Simulation (MOBS)*. Boston USA, 2006.

[20] Schneider Electric, "Metered rack PDU," http://www.apc.com/products/family/?id=136, 2015, [Online; accessed in 11-April-2015].

[21] W. L. Bircher and L. K. John, "Complete system power estimation using processor performance events," *Computers, IEEE Transactions on*, 2012.

[22] R. Bertran, M. Gonzalez, X. Martorell, N. Navarro, and E. Ayguade, "Decomposable and responsive power models for multicore processors using performance counters," in *Proceedings of ICS*. ACM, 2010.

[23] C. Isci and M. Martonosi, "Runtime power monitoring in high-end processors: Methodology and empirical data," in *Proceedings of International Symposium on Microarchitecture (MICRO)*. IEEE Computer Society, 2003.

[24] T. Guoming, J. Weixiang, X. Zhifeng, L. Fangming, and W. Kui, "Zero-cost, fine-grained power monitoring of datacenters using non-intrusive power disaggregation," in *Proceedings of the 16th International Middleware Conference (Middleware)*. ACM/IFIP/USENIX, 2015.

[25] M. Y. Lim, A. Porterfield, and R. Fowler, "Softpower: fine-grain power estimations using performance counters," in *Proceedings of International Symposium on High-Performance Parallel and Distributed Computing (HPDC)*. ACM, 2010.

[26] P. Ranganathan, P. Leech, D. Irwin, and J. Chase, "Ensemble-level power management for dense blade servers," in *Proceedings of ISCA*. IEEE Computer Society, 2006.

[27] W. L. Bircher and L. K. John, "Complete system power estimation: A trickle-down approach based on performance events," in *Proceedings of International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 2007.

[28] S. Rivoire, P. Ranganathan, and C. Kozyrakis, "A comparison of high-level full-system power models." *HotPower*, 2008.

[29] J. C. McCullough, Y. Agarwal, J. Chandrashekar, S. Kuppuswamy, A. C. Snoeren, and R. K. Gupta, "Evaluating the effectiveness of model-based power characterization," in *Proceedings of USENIX Annual Technical Conference (ATC)*. USENIX, 2011.

[30] K. David and T. Wendy, "Types of electrical meters in data centers (Schneider Electric white paper)," http://www.apcmedia.com/salestools/VAVR-8NALSE/VAVR-8NALSE_R1_EN.pdf, 2015, [Online; accessed in 21-February-2015].

[31] M. Zeifman and K. Roth, "Nonintrusive appliance load monitoring: Review and outlook," *Consumer Electronics, IEEE Transactions on*, 2011.

[32] A. Zoha, A. Gluhak, M. A. Imran, and S. Rajasegarar, "Non-intrusive load monitoring approaches for disaggregated energy sensing: A survey," *Sensors*, 2012.

[33] I. Gavrichenkov, "Cpu overclocking vs. power consumption," http://www.xbitlabs.com/articles/cpu/display/power-consumption-overclocking.html, 2010, [Online; accessed in 30-August-2015].

[34] S. Wang, H. Chen, and W. Shi, "Span: A software power analyzer for multicore computer systems," *Sustainable Computing: Informatics and Systems*, 2011.

[35] R. Das, J. O. Kephart, C. Lefurgy, G. Tesauro, D. W. Levine, and H. Chan, "Autonomic multi-agent management of power and performance in data centers," in *Proceedings of International Conference on Autonomous Agents & Multiagent Systems (AAMAS)*. IFAAMAS, 2008.

[36] C.-H. Hwang and A. C.-H. Wu, "A predictive system shutdown method for energy saving of event-driven computation," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 2000.

[37] C. Rusu, A. Ferreira, C. Scordino, and A. Watson, "Energy-efficient real-time heterogeneous server clusters," in *Proceedings of Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2006.

[38] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel, *Robust statistics: the approach based on influence functions*. John Wiley & Sons, 2011.

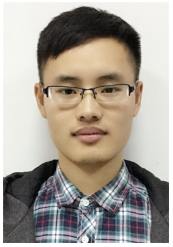
[39] die.net, "Linux man page: dstat," http://linux.die.net/man/1/dstat, 2014, [Online; accessed in 11-November-2014].

[40] R. Zamani and A. Afsahi, "A study of hardware performance monitoring counter selection in power modeling of computing systems," in *Proceedings of International Green Computing Conference (IGCC)*. IEEE, 2012.

[41] C. Lively, X. Wu, V. Taylor, S. Moore, H.-C. Chang, C.-Y. Su, and K. Cameron, "Power-aware predictive models of hybrid (mpi/openmp) scientific applications on multicore systems," *Computer Science-Research and Development*, 2012.

[42] Intel Corp., "Intelligent platform management interface specification v2.0," http://www.intel.com/design/servers/ipmi/pdf/, 2014, [Online; accessed in 11-October-2014].

Guoming Tang (S'12) received the bachelor's and master's degrees from the National University of Defense Technology, China in 2010 and 2012, respectively. He is currently working toward the PhD degree in the Department of Computer Science, University of Victoria, BC, Canada. His research interests include green computing, smart energy systems, and machine learning.



Weixiang Jiang received his B.S. degree from School of Computer Science and Technology, Huazhong University of Science and Technology, China in 2014. He is currently a Ph.D. student in School of Computer Science and Technology, Huazhong University of Science and Technology. His research interests include green computing and datacenter energy.

Zhifeng Xu is currently a Master student in the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China. His current research interests focus on cloud computing and networking.

Fangming Liu (S'08-M'11-SM'16) received his B.Engr. degree in 2005 from Department of Computer Science and Technology, Tsinghua University, Beijing; and his Ph.D. degree in computer science and engineering from Hong Kong University of Science and Technology, Hong Kong, in 2011. He is a full professor in Huazhong University of Science and Technology, Wuhan, China. His research interests include cloud computing and datacenter, mobile cloud, green computing, SDN and virtualization. He is selected into National Program for Support of Top-Notch Young Professionals of National Program for Special Support of Eminent Professionals. He is a Youth Scientist of National 973 Basic Research Program Project of SDN-based Cloud Datacenter Networks. He was a StarTrack Visiting Faculty in Microsoft Research Asia in 2012-2013. He has been the Editor-in-Chief of EAI Endorsed Transactions on Collaborative Computing, a guest editor for IEEE Network Magazine, and served as TPC for ACM Multimedia 2014 and 2016, e-Energy 2016, IEEE INFOCOM 2013-2017, ICNP 2014 TPC and 2016 Poster/Demo Co-Chair, IWQoS 2016 and ICDCS 2015-2016. He is a Senior Member of IEEE.

Kui Wu (S'98-M'02-SM'07) received the B.Sc. and the M.Sc. degrees in Computer Science from Wuhan University, China in 1990 and 1993, respectively, and the Ph.D. degree in Computing Science from the University of Alberta, Canada, in 2002. He joined the Department of Computer Science at the University of Victoria, Canada in 2002 and is currently a Professor there. His research interests include smart grid, mobile and wireless networks, and network performance evaluation.

NIPD: Non-Intrusive Power Disaggregation in Legacy Datacenters (Supplementary Material)

Guoming Tang, *Student Member, IEEE,*
Weixiang Jiang, Zhifeng Xu, Fangming Liu, *Senior Member, IEEE,* and Kui Wu, *Senior Member, IEEE*

1 Equation Transformation

1.1 Transformation of Equation (12)

For a VHC consisting of m servers, each with n component states and $\tilde{n}$ dilated state features, given its PMF in form of (7) and state vector in form of (4), the aggregated power consumption at time j can be expressed as:

$$\hat{y}_j = \sum_{i=1}^{m} f(x_j^{(i)}) \tag{1a}$$
$$= f(x_j^{(1)}) + f(x_j^{(2)}) + \cdots + f(x_j^{(m)}) \tag{1b}$$
$$= w^\intercal \left(x_j^{(1)} + x_j^{(2)} + \cdots + x_j^{(m)}\right) \tag{1c}$$
$$= w^\intercal \hat{x}_j \tag{1d}$$

where

$$\hat{x}_j = x_j^{(1)} + x_j^{(2)} + \cdots + x_j^{(m)} \tag{2a}$$
$$= \left[\sum_{i=1}^{m} x_{1,j}^{(i)}, \sum_{i=1}^{m} x_{2,j}^{(i)}, \cdots, \sum_{i=1}^{m} x_{\tilde{n},j}^{(i)}\right]^\intercal. \tag{2b}$$

1.2 Transformation of Equation (13)

Assume that a datacenter consists of r VHCs and the PMF of the κ-th ($1 \leq \kappa \leq r$) VHC is denoted in form of (10). Then, at an arbitrary time instant j, the aggregated power consumption generated by r VHCs can be expressed as:

$$y_j = \sum_{\kappa=1}^{r} \sum_{i=1}^{m_\kappa} f_\kappa(x_j^{(i)}) \tag{3a}$$
$$= \sum_{\kappa=1}^{r} \left\{ f_\kappa(x_j^{(1)}) + f_\kappa(x_j^{(2)}) + \cdots + f_\kappa(x_j^{(m_\kappa)}) \right\} \tag{3b}$$
$$= \sum_{\kappa=1}^{r} \left\{ (w^{(\kappa)})^\intercal \left(x_j^{(1)} + x_j^{(2)} + \cdots + x_j^{(m_\kappa)}\right) \right\} \tag{3c}$$
$$= \tilde{w}^\intercal \tilde{x}_j \tag{3d}$$

where

$$\tilde{x}_j = \left[\hat{x}_j^{(1)}, \hat{x}_j^{(2)}, \cdots, \hat{x}_j^{(r)}\right]^\intercal \tag{4}$$

and

$$\tilde{w} = \left[w^{(1)}, w^{(2)}, \cdots, w^{(r)}\right]^\intercal, \tag{5}$$

in which $\hat{x}_j^{(\kappa)}$ and $w^{(\kappa)}$ are defined by (12) and (11), respectively.

2 Proof of Remark 3

Given that a datacenter is consist of r ($r \geq 1$) VHCs, each with n_κ component states and $\tilde{n}_\kappa$ ($1 \leq \kappa \leq r$) dilated state features, for each data entry in the training dataset in form of $(\tilde{x}, y)$, the number of non-constant elements of $\tilde{x}$ is $\sum_{\kappa=1}^{r} \tilde{n}_\kappa$ (referring to (12)). Then, for each of the elements, as the normalizing resolution is set as p and the normalized range is $[0, 1]$, the number of its possible values is $\left\lceil \frac{1}{p} \right\rceil$. Therefore, the total number of possible combinations, i.e., the values of $\tilde{x}$, is $\left\lceil \frac{1}{p^{\tilde{n}_1}} \right\rceil + \left\lceil \frac{1}{p^{\tilde{n}_2}} \right\rceil + \cdots + \left\lceil \frac{1}{p^{\tilde{n}_r}} \right\rceil$, i.e., $\sum_{\kappa=1}^{r} \left\lceil \frac{1}{p^{\tilde{n}_\kappa}} \right\rceil$.

3 PMFs Training Complexity

For PMFs training, the optimization model established in (15) is used to find the optimal PMFs coefficients, which essentially falls into the form of lease square linear regression. With t data entries in the training dataset, the closed-form solution to the least square regression problem (15), i.e., the PMFs coefficients $\tilde{w}$, can be expressed as:

$$\tilde{w} = (X^\intercal X)^{-1} X^\intercal \hat{y}, \tag{6}$$

where $X = [\tilde{x}_1, \tilde{x}_2 \cdots, \tilde{x}_t]^\intercal$ and $\hat{y} = [y_1, y_2 \cdots, y_t]^\intercal$.

Assuming that the total number of dilated state features for all VHCs is $\tilde{n}$, $\tilde{n} = \sum_{\kappa=1}^{r} m_\kappa$ where m_κ denotes the number of component states for the κ-th VHC, the time complexity to get $\tilde{w}$ from formulation (6) is $O(\tilde{n}^2 \cdot t)$.

For the datacenter hosting extremely large number of servers, the complexity of PMFs training can be much reduced under the following two cases:

- If the number of servers in a virtual homogeneous cluster (VHC) is small, the PMF modeling and training process developed for the whole datacenter can be directly applied within the VHC, as long as the aggregated power of the VHC is accessible.
- If the number of servers in a VHC is large, the (dilated) state features from part of the servers (e.g., a few hundreds as in our case) should be enough to yield a relatively accurate PMF, as long as the aggregated power of these servers is accessible.