Zero-Cost, Fine-Grained Power Monitoring of Datacenters Using Non-Intrusive Power Disaggregation

Guoming Tang[1], Weixiang Jiang[2], Zhifeng Xu[2], Fangming Liu[2]*, and Kui Wu[1]
[1]University of Victoria, Victoria, BC, Canada
[2]Huazhong University of Science and Technology, Wuhan, Hubei, China
[1]{guoming, wkui}@uvic.ca, [2]{wxjiang, xzf, fmliu}@hust.edu.cn

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. 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 the rack level, with mean relative error of 2.63%, and the server level, with mean relative error of 10.27% and 8.17% for the estimation of idle power and peak power, respectively.

Categories and Subject Descriptors

C.5.5 [**Computer System Implementation**]: Servers

General Terms

Design; Measurement; Experimentation

Keywords

datacenter, power monitoring, 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 [27]. A recent report [23] 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 [5, 24].

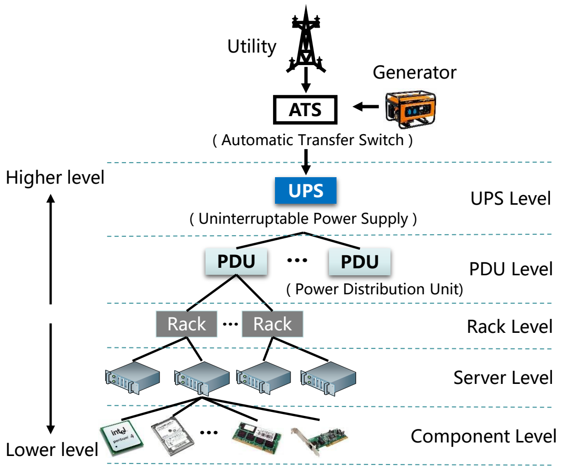


Figure 1: Power distribution hierarchy of the IT facilities in a typical datacenter.

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 [17, 18], idle power eliminating [21], and even cooling control and load balancing [1]. 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

*Part of this work was done at the School of Computer Science and Technology, Huazhong University of Science and Technology. The corresponding author is Fangming Liu (fmliu@hust.edu.cn).

Middleware '15, December 07-11, 2015, Vancouver, BC, Canada.
© 2015 ACM. ISBN 978-1-4503-3618-5/15/12 ...$15.00.
DOI: http://dx.doi.org/10.1145/2814576.2814728.

server-level power distribution can provide important information to identify hot spot “suspects”.

In addition to the aforementioned significance, fine-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 [34], 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 [35]. 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 [25]. 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 and 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 [33] has developed power monitoring solutions using power clamps or intelligent power strips. The IBM PowerExecutive solution [26] 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 [22] and ServerTechnology [32], 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 [11, 12] and Mantis [9]. 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

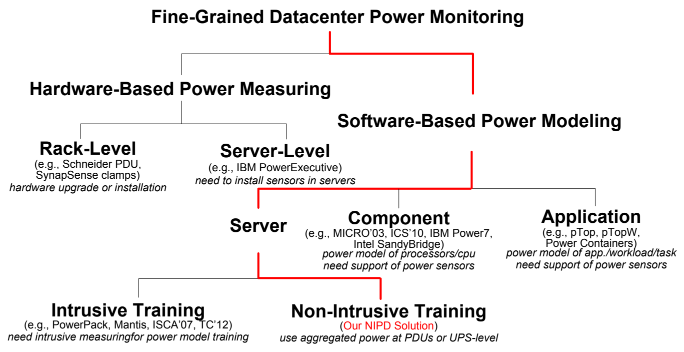

Fine-Grained Datacenter Power Monitoring
Hardware-Based Power Measuring
Software-Based Power Modeling
Rack-Level
(e.g., Schneider PDU, SynapSense clamps)
hardware upgrade or installation
Server-Level
(e.g., IBM PowerExecutive)
need to install sensors in servers
Server
Component
(e.g., MICRO'03, ICS'10, IBM Power7, Intel SandyBridge)
power model of processors/cpu
need support of power sensors
Application
(e.g., pTop, pTopW, Power Containers)
power model of app./workload/task
need support of power sensors
Intrusive Training
(e.g., PowerPack, Mantis, ISCA'07, TC'12)
need intrusive measuringfor power model training
Non-Intrusive Training
(Our NIPD Solution)
use aggregated power at PDUs or UPS-level

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

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.

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 further improve the NIPD solution to support online update of PMFs with selective training data collection. The online PMFs update and training data collection can not only improve the precision of power disaggregation, but also support the NIPD solution running in real-time.

- 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 at the rack level, with mean relative error of 2.63%, and the server level, with mean relative error of 10.27% and 8.17% for the estimation of 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. We discuss the use of NIPD as a middleware for power management in Sec. 7. The paper is concluded in Sec. 8.

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

solution \ *specification*	*software/ hardware*	*intrusive/ non-intrusive*	*monitoring level*	*deployed scale (#server)*	*homogeneous/ heterogeneous*	*error rate*	*hardware cost ($/server)*
Schneider Solution [31] SynapSense Solution [33]	hardware	intrusive	rack	any scale	both	$(< 1\%)$	$55 \sim 68$
PowerExecutive [26]	hardware	intrusive	rack, server	any scale	both	$(< 1\%)$	$2000 \sim 2800$
PowerPack [11, 12]	software	intrusive	rack, server, component	9	homogeneous	$(< 1\%)$	$15 \sim 80$
Mantis [9]	software	intrusive	rack, server	1	–	$< 15\%$	$30 \sim 200$
ISCA'07 [10]	software	intrusive	rack, server	a few hundred	homogeneous	$< 18\%$	$30 \sim 200$
TC'12 [4]	software	intrusive	rack, server, component	1	–	$< 9\%$	$129 \sim 340$
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.

2. RELATED WORK

Existing solutions for fine-grained power monitoring in the datacenter include two categories: hardware-based power measuring and software-based power modeling [2] [16], as shown in Fig. 2.

2.1 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 [31] provided the metered Rack Power Distribution Units (RPDU). In addition, SynapSense [33] 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 [26], 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.

2.2 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 [19], 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 [28], 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 [3]. Five different sever-level power models, which correlate AC power measurements with software utilization metrics, were investigated in [29], in which the accuracy and portability of the power models were also compared over different workloads and servers. More power models including linear and non-linear ones were discussed in [20]. Among the developed power models, linear regression models are the most widely applied. It is simple and has been shown to yield accurate results in the server-level power estimation.

Various power monitoring platforms were developed utilizing software-based power modeling. PowerPack was initially established in [11] and further improved in [12]. 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 [9], 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.

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 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. 2.2, 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 [7] or certain interfaces (e.g.,

[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.

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 [38, 39]. 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 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 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. For ease of reference, we list the notations in Table 2.

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:

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

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

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

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):

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

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

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

in which $\mu_{\kappa,j}^{(i)}$ represents the value of the κ-th ($1 \le \kappa \le n$) 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 $s_j^{(i)}, 1 \le i \le m, 1 \le j \le t$, the problem of **non-intrusive power disaggregation (NIPD)** is to estimate the power consumption of each individual server at each time instant, i.e., $y_j^{(i)}, 1 \le i \le m, 1 \le j \le t$.*

[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).

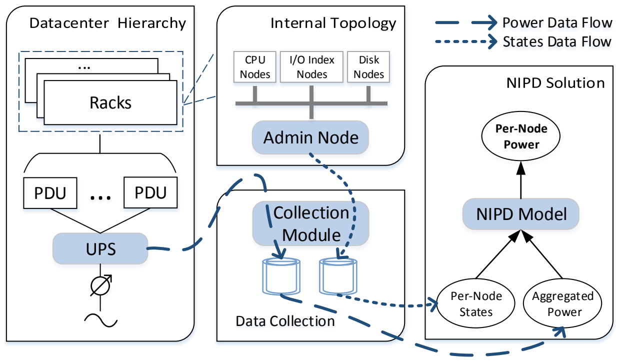


Figure 3: Framework of non-intrusive power disaggregation over a datacenter.

Table 2: Summary of notations

Notation	*Description*
m	number of servers
n	number of component states
r	number of virtual homogeneous clusters (VHCs)
y	aggregated power vector of datacenter
$y_j^{(i)}$	power consumption of the i-th server at time j
$s_j^{(i)}$	state vector of the i-th server at time j
$\mu_{n,j}^{(i)}$	the n-th component state of server i at time j
$d_j^{(r)}$	on/off number of servers in the r-th VHC at time j
$w^{(\kappa)}$	coefficient vector of PMF of the κ-th VHC
$\tilde{w}$	coefficient vector of PMFs of all VHCs

4.2 Training PMFs

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

As mentioned in Sec. 2.2, a linear model can capture the relation between the energy consumption of a server and its component states, and its complexity is much lower than that of non-linear ones. Therefore, as the first choice, we model the PMF as a linear function, i.e., a server's power consumption can be modeled by the linear combination of its component states. The accuracy of linear PMF will be validated later in Sec. 6.

For servers in the same VHC, with the state vector in (3), we define their PMF as follows:

$$f(s) = [1, s]\, w \tag{5}$$

where w is a *coefficient vector*[3] denoted as:

$$w = [w_0, w_1, w_2, \cdots, w_n]^\intercal . \tag{6}$$

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 major components' states 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* $f(s_j^{(i)})$ *best approximates* $y_j^{(i)}$*) by the components modeled in PMF. Hence, the power consumption of each component modeled in PMF is not necessarily the true value.*

For the overall power consumption of a server $f(s)$, it can be broken down into two parts: idle power (or static power) and dynamic power [36]. 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. In the PMF coefficient vector (6), w_0 is the constant term that models the idle power, and $w_1, w_2, \cdots, w_n$ are coefficients associated with the dynamic power of different components.

4.2.1 Estimation of Coefficients

We first estimate the coefficients of a server's PMF. Consider a datacenter that consists of r VHCs. Assume that there are m_κ servers in the κ-th ($1 \leq \kappa \leq r$) VHC, and each server of the κ-th VHC reports the states of n_κ components. Thus, with the state vector in (3), the PMF for the VHC can be expressed as:

$$f_\kappa(s) = [1, s]\, (w^{(\kappa)})^\intercal \tag{7}$$

[3]Note that the coefficient vector also includes the constant term w_0.

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

$$w^{(\kappa)} = \left[w_0^{(\kappa)}, w_1^{(\kappa)}, w_2^{(\kappa)}, \cdots, w_{n_\kappa}^{(\kappa)}\right] . \tag{8}$$

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

$$\hat{s}_j^{(\kappa)} = \left[m_\kappa, \sum_{i=1}^{m_\kappa} \mu_{1,j}^{(i)}, \sum_{i=1}^{m_\kappa} \mu_{2,j}^{(i)}, \cdots, \sum_{i=1}^{m_\kappa} \mu_{n,j}^{(i)}\right] . \tag{9}$$

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

$$\tilde{s}_j = \left[\hat{s}_j^{(1)}, \hat{s}_j^{(2)}, \cdots, \hat{s}_j^{(r)}\right] , \tag{10}$$

and

$$\tilde{w} = \left[w^{(1)}, w^{(2)}, \cdots, w^{(r)}\right]^\intercal , \tag{11}$$

in which $\hat{s}_j^{(\kappa)}$ and $w^{(\kappa)}$ are defined by (9) and (8), respectively. (Refer to Appx. A.1 and Appx. A.2 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 can be formulated as the training model for the r PMFs of the datacenter:

$$\min_{\tilde{w}} \quad \sum_{j=1}^{t} (\tilde{s}_j \tilde{w} - y_j)^2 . \tag{12}$$

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 state vectors.

Nevertheless, the above LSE training model cannot capture multiple but only one constant term appearing in the coefficient vector [29]. Consequently, if there are more than one VHC in the datacenter ($r > 1$), the resulted constant terms (i.e., $w_0^{(1)}, w_0^{(2)}, \cdots, w_0^{(r)}$) from (12) 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.

4.2.2 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 [6, 14, 30]. Such a strategy provides us with an opportunity to estimate the constant terms in PMFs.

DEFINITION 3. *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. 4, 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:

$$d_j := \left[d_j^{(1)}, d_j^{(2)}, \cdots, d_j^{(r)}\right] , \tag{13}$$

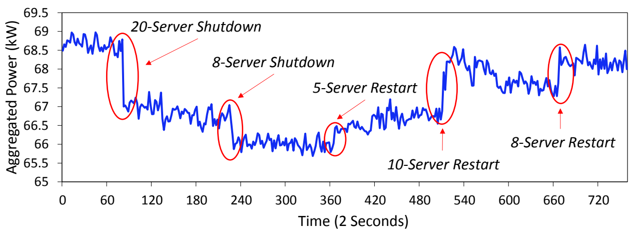


Figure 4: Off/on events captured during turning on/off servers in our datacenter consisting of 326 servers.

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., $w_0 = [w_0^{(1)}, w_0^{(2)}, \cdots, w_0^{(r)}]^\intercal$:

$$\min_{w_0} \sum_{j=1}^{t} (d_j w_0 - \Delta y_j)^2 . \tag{14}$$

REMARK 2. *In the estimation of constant terms of PMFs, we can combine the optimization strategy using (14) 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.

4.3 PMFs Update

To make PMFs accurate, we need a training dataset that contains complete component states, i.e., all possible component states 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 information. This phenomenon may result in inaccurate PMFs.

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.

4.3.1 Selective Training Data Collection

We first set an update time interval for the training dataset, denoted as Δt_1. At an arbitrary time instant j, the components states collected from r VHCs can be expressed as $\tilde{s}_j$ in form of (10). Along with the measured aggregated power consumption of the datacenter at the same moment y_j, a data entry can be represented as $(\tilde{s}_j, y_j)$. With data entry of $(\tilde{s}_j, y_j)$, the process of selective training data collection is as below:

- *Step 1.* normalize each element in $\tilde{s}_j$ with the corresponding maximum value[4], 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 $(\tilde{s}_j, y_j)$ 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 alleviates the affect of outliers [13] and makes the PMFs' training more robust. In addition to the collection of component states, the same strategy can also be applied to collect the off/on events for constant terms estimation introduced in Sec. 4.2.2.

REMARK 3. *For our selective data collection, the resolution of the normalized component states determines the maximum number of data entries in the training dataset. Assuming that a datacenter consists of $r(r \geq 1)$ VHCs, each with $n_\kappa (1 \leq \kappa \leq r)$ component states, and the preset resolution of normalized component states 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 Appx. B for the proof.*

Theoretically, with the above data collection strategy, the training dataset will eventually become complete as time goes on. In practice, however, datacenter scaling-out (i.e., adding computing resources) or scaling-up (i.e., upgrading IT facilities) may lead to changes of PMFs. In this case, a new training dataset needs to be collected with the same procedure, and PMFs need to update accordingly.

4.3.2 Complexity of PMFs Update

We can update PMFs at a regular basis, e.g. every Δt_2 interval time, using the most updated training dataset. Note that the PMFs update is carried out during the normal running of the datacenter and has very small overhead.

According to the analysis in Appx. C, the complexity of PMFs training has a linear growth with increase of data entries and a quadratic growth with increase of component states. Fortunately, as explained in Remark 3, the number of the training data entries using selective data collection is not large (less than $10K$ in our latest experiment), We will also show that a small number of component states (e.g., 6 in our experiment) are sufficient to provide accurate PMFs in Sec. 6.

Finally, to recap our NIPD solution, at background the training dataset is selectively updated and duly applied to update PMFs; at foreground the real-time component state information is used to obtain server-level power estimation.

[4]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.

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 3 shows the detailed configuration of each type of server. Fig. 5 illustrates the power architecture, network topology, and data collection modules of the experimental environment.

Table 3: Configuration of Server Nodes

Type	*Configurations*	*Number*
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

As shown in Fig. 5, 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. 5).

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 rate of 1 second). Particularly, we use `dstat` tool [8], a widely-used resource statistic tool that can gather various component states of a server, as shown in Table 4. Note that other tools can also be used here, such as `vmstat`, `iostat`, `mpstat` and `netstat`.

Table 4: State metrics that can be collected using `dstat` tool

Component	*State label*	*Description*

processor	`usr`
	`sys`
	`idle`
	`wai`
		CPU utilization for user processes
		CPU utilization for system processes
		CPU in idle
		CPU utilization for I/O waiting

memory	`used`
	`buff`
	`cach`
	`free`
		memory usage for processes
		buffer memory
		cache memory
		free memory

disk	`read` `write`	disk reading amount disk writing amount
network	`recv` `send`	traffic amount that the system received traffic amount that the system sent
paging	`in` `page`	# pages changed from disk to memory # pages changed from memory to disk
system	`int` `csw`	system interruption time content switch times

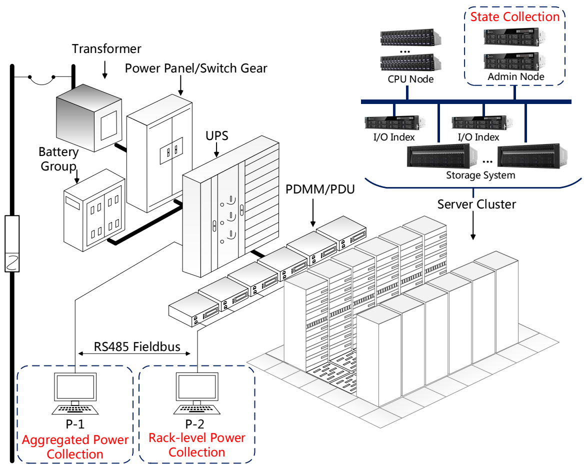


Figure 5: Power architecture, network topology and data collection modules in our experimental environment. Note that the rack-level power measurements provided by PDMMs are only used for validation purpose in this paper.

Rather than using all states information provided by `dstat` for PMFs training, we perform comprehensive tests and select the following 6 state metrics from the collected states in Table 4: total CPU utilization (1−`idle`), total memory utilization (1−`free`), disk reading/writing (`read`/`write`) and network traffic receiving/sending (`recv`/`send`). The main reasons to use only the above 6 state metrics are: (1) the selected ones cover the major hardware components of the server, and (2) including other metrics increases the overhead of training PMFs but do not improve the accuracy of PMFs clearly in our test.

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 [15] 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. 4), which are fed into the optimization model (14) 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, which will be introduced in Sec. 6.3.

6. EVALUATION

In our experimental environment introduced in Sec. 5, we evaluate the precision and complexity of our NIPD solution for power monitoring at the rack level and the server level, respectively.

6.1 Experiment Configuration

Table 5 summarizes the values of parameters set in our experiments. We setup the parameters based on the following considerations:

Table 5: Parameter settings of our experiments

Parameter	*Setting*
number of VHCs (r)	4
number of component states (n_κ)	[6, 6, 6, 6]
normalizing resolution (p)	0.01
training dataset update interval (Δt_1)	2 seconds
PMFs update interval (Δt_2)	5 minutes, 0.5 hour

- Number of VHCs (r): According to Table 3, 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 choose 6 component states for PMFs training as well as power estimation of individual servers.
- 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 3, 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.

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

$$\text{MRE} := \frac{1}{t}\sum_{j=1}^{t}\left|\frac{y'_j - y_j}{y_j}\right| \tag{15}$$

Table 6: Workloads/benchmarks used for NIPD evaluations

Workload		*Description*	*Purpose*
Idle		Only 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		Filesystem benchmark tool	
Our Synthetic		Occupy CPU randomly Read/write memory randomly	Rack-level validation

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 6, we collect training data and duly update PMFs following the strategies introduced in Sec. 4.3. 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. Part of the results are summarized and shown in Fig. 6. We can see that MRE monotonically decreases with the increase of training dataset, and tends to be stable at the value strictly smaller than 5%.

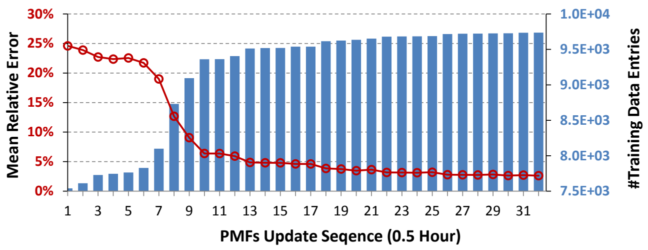


Figure 6: With increase of training dataset, the MRE of power estimation from the updated PMFs decreases.

To illustrate the performance results more clearly, the power estimation results for the first two racks (in 0.5 hour) are shown in Fig. 7, along with the ground truth power values. The MRE for the power estimation of the two racks is 1.29% and 4.55%, respectively. To have a view of the overall performance in the datacenter, we test and illustrate the MRE values over all 12 racks in Fig. 8, and the average MRE over the entire datacenter is only 2.63%. Note that rack-12 is dedicated to an InfiniBand(IB) switch and its power consumption is quite stable around $2.5 \pm 0.1kW$ (provided by the vendor and validated by our observations). Thus, we only use the constant term for power estimation of rack-12 and can obtain quite accurate result (with MRE of 0.85%).

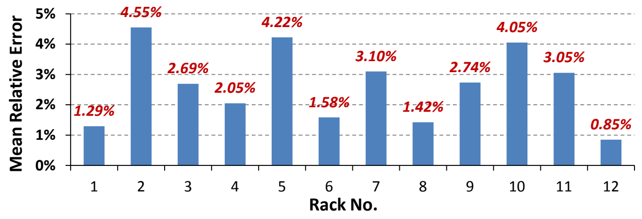


Figure 8: The overall NIPD performance (MRE errors) over the datacenter using updated PMFs.

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.

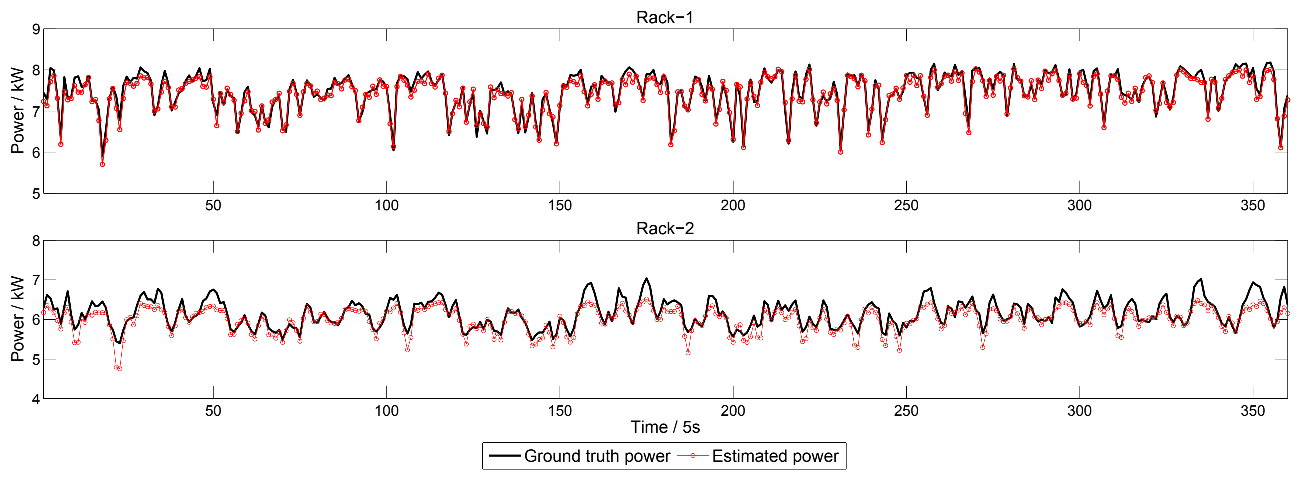


Figure 7: Estimated power consumption for rack-1 and rack-2 with corresponding ground truth values.

Table 7: Referred idle/peak power or power range of CPU nodes and disk array nodes

Node type	*Referred metric*	*Power (range)*
CPU node	idle power	75.4 *Watts*
	peak power	200 *Watts*
Disk array node	power range	1020 ∼ 1170 *Watts*

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. The idle power of CPU nodes in our datacenter can be estimated via the process introduced in Sec. 4.2.2, and their peak power (or name plate power) can be learned by referring to nameplate power provided by the server vendor. For a disk array node, its power consumption is usually larger but relatively stable compared with the CPU node. We estimate its working power range rather than the idle/peak power by making use of rack-level power[5]. The measured or estimated idle/peak power and working power range of the servers in our datacenter are illustrated in Table 7, and we use these 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) and two disk array nodes (Node-309 and Node-310) among our datacenter as test nodes. For the four CPU test nodes, we make two of them run our peak workload (listed in Table 6), and the other two firstly keep idle for 15 minutes and then run peak workload for another 15 minutes. As to the two disk array test nodes, we just leave them running and available to other users.

Along with corresponding referred power bounds, the resulting power estimation for the four CPU test nodes and two disk array nodes is demonstrated in Fig. 9 and Fig. 10, respectively. From the results we can see that, both the estimated idle/peak power for CPU nodes and power consumption for disk array nodes are close to the referred power (range). With respect to the referred idle/peak power values of CPU nodes[6], the MRE is 10.27% and 8.17% for the estimation of the idle power and peak power, respectively. By checking the performance under these two extreme cases, we can validate the effectiveness of our solution.

We have observed that the estimated power values are slightly larger than the referred ones. 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 some shared facilities (e.g., network switches 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 Appx. C).

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 homogeneous. 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 test nodes in Sec. 6.3.1 (Node-25 to Node-28) is illustrated in Fig. 9.

We can observe that the server-level power estimation by disaggregating the rack power is slightly better than that from disaggregating the entire datacenter power. The MRE of CPU nodes is 6.92% and 6.30% for the estimation of idle power and peak power, respectively. This is because that the impact of hardware components not modeled in PMFs is smaller at the rack level than at the whole datacenter, as per the discussion in Remark 1.

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

[5]In our experimental datacenter, some racks only contain CPU nodes and disk arrays. Thus we can shut down all the CPU nodes and only leave the disk arrays running to obtain the working power range of disk arrays.

[6]Peak power values refer to the power readings when the CPU utilization is 100%.

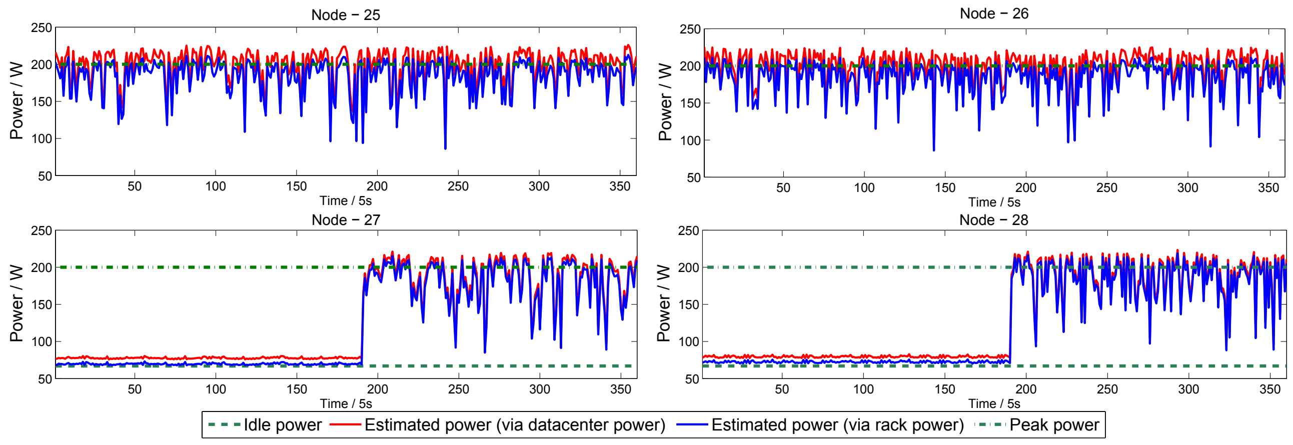


Figure 9: Estimated power consumption for four CPU test nodes and their referred power bounds using updated PMFs.

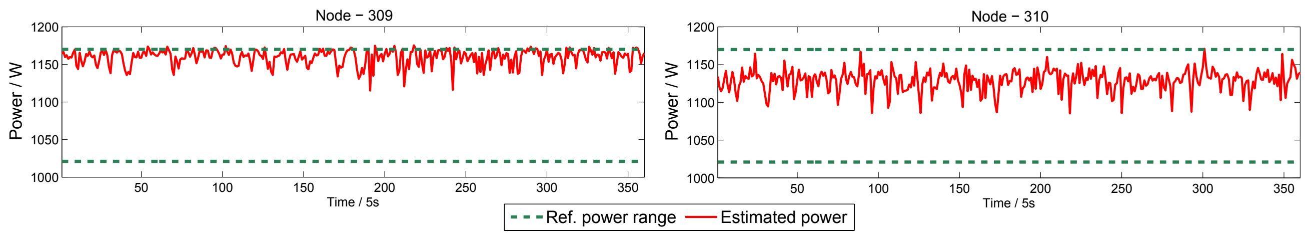
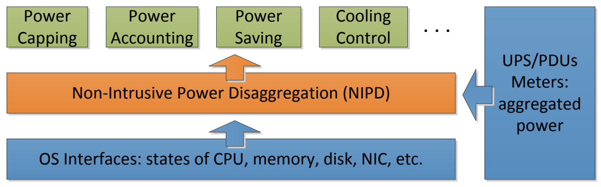


Figure 10: Estimated power consumption for two disk array test nodes and their referred power range using updated PMFs.

7. FURTHER DISCUSSION: NIPD AS MIDDLEWARE

As our NIPD solution can provide fine-grained power information at the server level, it can be used as an important middleware to support different power management applications, as illustrated in Fig. 11.



Power Capping
Power Accounting
Power Saving
Cooling Control
. . .
Non-Intrusive Power Disaggregation (NIPD)
OS Interfaces: states of CPU, memory, disk, NIC, etc.
UPS/PDUs Meters: aggregated power

Figure 11: NIPD as an important middleware to support various datacenter PM applications.

Support for Power Capping: As mentioned in the introduction, the power capacity of the IT facilities (such as a rack) estimated by servers' nameplate rating is much higher than their actual consumption. Taking the datacenter in our experimental environment as an example, according to our observations in Fig. 12, the average power consumption of a rack is no more than 60% of the designed power capacity. Under this situation, with server-level power information provided by NIPD, the datacenter operators can confidently allocate unused power to support new system requirements. Meanwhile, integrated with power capping technologies (e.g., DVFS, workload migration), our solution can help avoid the overrun of power infrastructure.

Support for Power Accounting: The fine-grained power information from NIPD can also be used for power accounting from

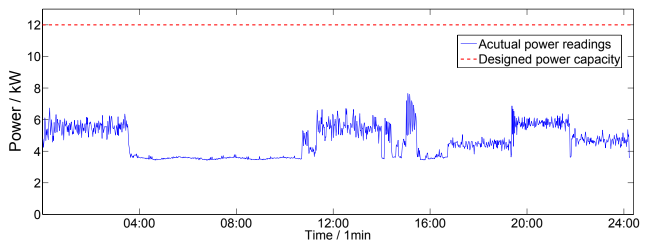


Figure 12: The power readings from the first rack in our datacenter, whose designed power capacity is 12 kW.

different perspectives. For example, as shown in Fig. 13 (left), the energy contributions from different server types in our datacenter can be easily derived, which can be used for better analysis of power efficiency. Furthermore, if the servers in the datacenter are dedicated to certain users from different departments, our NIPD solution can be applied for departmental power accounting, as the example shown in Fig. 13 (right). Such departmental power accounting can be used by the datacenter operator to gain insights into users' behavior from perspective of power consumption and to accurately charge different users in colocation dataceners [37].

Others: Based on results from NIPD, we can analyse the power consumption characteristics of different servers, workloads or users, and then adopt corresponding policies to save energy. For example, the power efficiency of different server types under the same workloads can be measured and used by the operators to choose the most appropriate servers for energy saving. In addition, the server-level power information can be used to draw the power distribution map of the datacenter, which provides clues to identify or predict "hot spots" for more intelligent cooling systems.

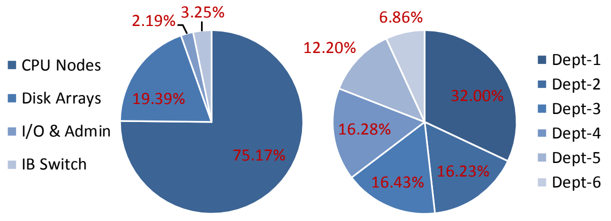


Figure 13: Power accounting examples using server-level power estimation.

8. CONCLUSIONS

In this paper, we defined the problem of non-intrusive power disaggregation (NIPD) for modern 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. Based on PMFs, which were updated online 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 the rack level with the mean relative error of 2.63% and at the server level with the mean relative error of 10.27% and 8.17% for the estimation of idle power and peak power, respectively.

9. ACKNOWLEDGEMENTS

This work was partially supported by the Mitacs Globalink Research Award, the Natural Sciences and Engineering Research Council of Canada under grant No.195819339, the National Natural Science Foundation of China (NSFC) under grant No.61370232, and the National Basic Research Program (973 Program) under grant No.2014CB347800.

10. REFERENCES

[1] Bash, C., and Forman, G. 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)* (2007), USENIX.

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

[3] Bircher, W. L., and John, L. K. 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)* (2007), IEEE.

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

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

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

[7] David, K., and Wendy, T. 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].

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

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

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

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

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

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

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

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

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

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

[18] Lefurgy, C., Wang, X., and Ware, M. Power capping: a prelude to power shifting. *Cluster Computing* (2008).

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

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

[21] Meisner, D., Gold, B. T., and Wenisch, T. F. Powernap: eliminating server idle power. In *Proceedings of ASPOLOS* (2009), ACM.

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

[23] 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].

[24] PETRUCCI, V., LOQUES, O., AND MOSSÉ, D. 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)* (2010), ACM.

[25] 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].

[26] POPA, P. Managing server energy consumption using IBM PowerExecutive. *IBM Systems and Technology Group, Tech. Rep* (2006).

[27] QURESHI, A., WEBER, R., BALAKRISHNAN, H., GUTTAG, J., AND MAGGS, B. Cutting the electric bill for internet-scale systems. In *Proceedings of SIGCOMM* (2009), ACM.

[28] RANGANATHAN, P., LEECH, P., IRWIN, D., AND CHASE, J. Ensemble-level power management for dense blade servers. In *Proceedings of ISCA* (2006), IEEE Computer Society.

[29] RIVOIRE, S., RANGANATHAN, P., AND KOZYRAKIS, C. A comparison of high-level full-system power models. *HotPower* (2008).

[30] RUSU, C., FERREIRA, A., SCORDINO, C., AND WATSON, A. Energy-efficient real-time heterogeneous server clusters. In *Proceedings of Real-Time and Embedded Technology and Applications Symposium (RTAS)* (2006), IEEE.

[31] SCHNEIDER ELECTRIC. Metered rack PDU. `http://www.apc.com/products/family/?id=136`, 2015. [Online; accessed in 11-April-2015].

[32] SERVERTECHNOLOGY. Server technology products. `https://www.servertech.com/products`, 2015. [Online; accessed in 21-April-2015].

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

[34] UPTIMEINSTITUTE. Data center industry survey, 2012.

[35] WANG, D., REN, C., AND SIVASUBRAMANIAM, A. Virtualizing power distribution in datacenters. In *Proceedings of ISCA* (2013), ACM.

[36] WANG, S., CHEN, H., AND SHI, W. Span: A software power analyzer for multicore computer systems. *Sustainable Computing: Informatics and Systems* (2011).

[37] WIKIPEDIA. Colocation center. `http://en.wikipedia.org/wiki/Colocation_centre`, 2015. [Online; accessed in 15-March-2015].

[38] ZEIFMAN, M., AND ROTH, K. Nonintrusive appliance load monitoring: Review and outlook. *Consumer Electronics, IEEE Transactions on* (2011).

[39] ZOHA, A., GLUHAK, A., IMRAN, M. A., AND RAJASEGARAR, S. Non-intrusive load monitoring approaches for disaggregated energy sensing: A survey. *Sensors* (2012).

APPENDIX

A. EQUATION TRANSFORMATION

A.1 Transformation of Equation (9)

For a VHC consisting of m servers, each with n component states, given its PMF in form of (5) and state vector in form of (4), the aggregated power consumption at time j can be expressed as:

$$\begin{aligned}\hat{y}_j &= \sum_{i=1}^{m} f(s_j^{(i)}) && (16a)\\ &= f(s_j^{(1)}) + f(s_j^{(2)}) + \cdots + f(s_j^{(m)}) && (16b)\\ &= \left[m, s_j^{(1)} + s_j^{(2)} + \cdots + s_j^{(m)}\right] w && (16c)\\ &= \hat{s}_j w && (16d)\end{aligned}$$

where

$$\begin{aligned}\hat{s}_j &= \left[m, s_j^{(1)} + s_j^{(2)} + \cdots + s_j^{(m)}\right] && (17a)\\ &= \left[m, \sum_{i=1}^{m} \mu_{1,j}^{(i)}, \sum_{i=1}^{m} \mu_{2,j}^{(i)}, \cdots, \sum_{i=1}^{m} \mu_{n,j}^{(i)}\right]. && (17b)\end{aligned}$$

A.2 Transformation of Equation (10)

Assume that a datacenter consists of r VHCs and the PMF of the κ-th ($1 \le \kappa \le r$) VHC is denoted in form of (7). Then, at an arbitrary time instant j, the aggregated power consumption generated by r VHCs can be expressed as:

$$\begin{aligned}y_j &= \sum_{\kappa=1}^{r} \sum_{i=1}^{m_\kappa} f_\kappa(s_j^{(i)}) && (18a)\\ &= \sum_{\kappa=1}^{r} \left\{ f_\kappa(s_j^{(1)}) + f_\kappa(s_j^{(2)}) + \cdots + f_\kappa(s_j^{(m_\kappa)}) \right\} && (18b)\\ &= \sum_{\kappa=1}^{r} \left\{ \left[m_\kappa, s_j^{(1)} + s_j^{(2)} + \cdots + s_j^{(m_\kappa)}\right] (w^{(\kappa)})^\mathsf{T} \right\} && (18c)\\ &= \tilde{s}_j \tilde{w} && (18d)\end{aligned}$$

where

$$\tilde{s}_j = \left[\hat{s}_j^{(1)}, \hat{s}_j^{(2)}, \cdots, \hat{s}_j^{(r)}\right] \qquad (19)$$

and

$$\tilde{w} = \left[w^{(1)}, w^{(2)}, \cdots, w^{(r)}\right]^\mathsf{T}, \qquad (20)$$

in which $\hat{s}_j^{(\kappa)}$ and $w^{(\kappa)}$ are defined by (9) and (8), respectively.

B. PROOF OF REMARK 3

Given that a datacenter is consist of $r (r \ge 1)$ VHCs, each with $n_\kappa (1 \le \kappa \le r)$ component states, for each data entry in the training dataset in form of $(\tilde{s}, y)$, the number of non-constant elements of $\tilde{s}$ is $\sum_{\kappa=1}^{r} n_\kappa$ (referring to (9)). 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 $\tilde{s}$, is $\left\lceil \frac{1}{p^{n_1}} \right\rceil + \left\lceil \frac{1}{p^{n_2}} \right\rceil + \cdots + \left\lceil \frac{1}{p^{n_r}} \right\rceil$, i.e., $\sum_{\kappa=1}^{r} \left\lceil \frac{1}{p^{n_\kappa}} \right\rceil$.

C. PMFS TRAINING COMPLEXITY

For PMFs training, the optimization model established in (12) 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 (12), i.e., the PMFs coefficients $\tilde{w}$, can be expressed as:

$$\tilde{w} = (S^\mathsf{T} S)^{-1} S^\mathsf{T} \hat{y}, \qquad (21)$$

where $S = [\tilde{s}_1, \tilde{s}_2 \cdots, \tilde{s}_t]^\mathsf{T}$ and $\hat{y} = [y_1, y_2 \cdots, y_t]^\mathsf{T}$.

Assuming that the total number of component states for all VHCs is n, $n = \sum_{\kappa=1}^{r} m_\kappa$ where m_κ denotes the number of component states for the κ-th VHC, the time complexity to get $\tilde{w}$ from formulation (21) is $O(n^2 \cdot t)$.