A Taxonomy and Survey of Power Models and Power Modeling for Cloud Servers

WEIWEI LIN and FANG SHI, South China University of Technology
WENTAI WU, University of Warwick
KEQIN LI, State University of New York
GUANGXIN WU and AL-ALAS MOHAMMED, South China University of Technology

Due to the increasing demand of cloud resources, the ever-increasing number and scale of cloud data centers make their massive power consumption a prominent issue today. Evidence reveals that the behaviors of cloud servers make the major impact on data centers' power consumption. Although extensive research can be found in this context, a systematic review of the models and modeling methods for the entire hierarchy (from underlying hardware components to the upper-layer applications) of the cloud server is still missing, which is supposed to cover the relevant studies on physical and virtual cloud server instances, server components, and cloud applications. In this article, we summarize a broad range of relevant studies from three perspectives: power data acquisition, power models, and power modeling methods for cloud servers (including bare-metal, virtual machine (VM), and container instances). We present a comprehensive taxonomy on the collection methods of server-level power data, the existing mainstream power models at multiple levels from hardware to software and application, and commonly used methods for modeling power consumption including classical regression analysis and emerging methods like reinforcement learning. Throughout the work, we introduce a variety of models and methods, illustrating their implementation, usability, and applicability while discussing the limitations of existing approaches and possible ways of improvement. Apart from reviewing existing studies on server power models and modeling methods, we further figure out several open challenges and possible research directions, such as the study on modeling the power consumption of lightweight virtual units like unikernel and the necessity of further explorations toward empowering server power estimation/prediction with machine learning. As power monitoring is drawing increasing attention from cloud service providers (CSPs), this survey provides useful guidelines on server power modeling and can be inspiring for further research on energy-efficient data centers.

This work was supported by the National Natural Science Foundation of China (nos. 61872084 and 61772205); Key-Area Research and Development Program of Guangdong Province (no. 2020B010164003); Guangzhou Science and Technology key projects (nos. 202007040002, 201902010040, and 201907010001); Guangzhou Development Zone Science and Technology (no. 2018GH17); Guangdong Major Project of Basic and Applied Basic Research (2019B030302002); and Fundamental Research Funds for the Central Universities, SCUT (no. 2019ZD26).

Authors' addresses: W. Lin, F. Shi (corresponding author), G. Wu, and A.-A. Mohammed, South China University of Technology, Guangzhou, China, 510006; emails: linww@scut.edu.cn, {csshifang, cswgx1nfinite}@mail.scut.edu.cn, alalssoft@qq.com; W. Wu, University of Warwick, West Midlands, England, CV4 7AL; email: wentai.wu@warwick.ac.uk; K. Li, State University of New York, New Paltz, NY 12561; email: lik@newpaltz.edu.

0360-0300/2020/09-ART100 $15.00
https://doi.org/10.1145/3406208

CCS Concepts: • **General and reference → Surveys and overviews**; • **Computer systems organization → Cloud computing**;

Additional Key Words and Phrases: Cloud server, data center, power model, power modeling, power consumption

ACM Reference format:
Weiwei Lin, Fang Shi, Wentai Wu, Keqin Li, Guangxin Wu, and Al-Alas Mohammed. 2020. A Taxonomy and Survey of Power Models and Power Modeling for Cloud Servers. *ACM Comput. Surv.* 53, 5, Article 100 (September 2020), 41 pages.
https://doi.org/10.1145/3406208

1 INTRODUCTION

With the onset of big data and the emerging application of Internet of Things (IoT), data centers, as the mission-critical computing infrastructure, have been operating around the clock to propel the fast growth of IT industry and economy [1]. However, these large-scale infrastructures are also monsters that keep swallowing up energy (and the owners' budgets as well), which raises various energy efficiency issues. According to statistics, back in 2011, the number of data centers around the world already exceeded 500,000 [2], and their electricity consumption accounted for approximately 1.5% of the world's total consumption [3]. By 2014, the energy consumption of U.S. data centers had reached 70 billion kWh, which accounted for 1.8% of the total U.S. electricity consumption [4]. It is predicted that the electricity demand for global data centers will increase by 66% from 2011 to 2035 [5]. Excessive energy consumption imposes a heavy budget burden on cloud service providers (CSPs), and energy reservation has been reckoned as a crucial mission in data center management.

An important aspect of power and energy management is the implementation of a dedicated monitoring system [2]. Here, energy consumption (E) refers to the total amount of electricity used to perform some work by a system over a time period (T), whereas Power (P) consumption is the rate at which the system consumes electricity. The relationship can be formulated as $E = \int_0^T P(t)dt$. Accurate measurement can indeed be realized by traditional monitoring methods (e.g., using external metering devices and internal integrated circuits), but they are not practically feasible in many scenarios due to major drawbacks like expensiveness, poor scalability and compatibility, and coarse granularity. Thus, the software-based power/energy monitoring has emerged as a prominent solution, where models are used to estimate and predict power or energy consumption.

Energy is the integral of power over a period of time, whereas power reflects the instant status of a target system. The difference makes it more challenging for the estimation of energy consumption wherein the difficulties are not only attributed to the uncertainty and inaccuracy in the prediction of $P(t)$ but also in the forecast of execution time for a given task. In this regard, most energy reservation related studies start from modeling systems' power or use power reduction as a primary incentive for energy optimization. In this survey, we cover an extensive scope of relevant work on power and energy management (including measuring, modeling, and optimization). Therefore, without loss of generality and for brevity, in this work we refer to all models, modeling methods, and techniques concerning power and energy consumption as power models, power modeling methods, and power optimization techniques, and additional statements will be made if they are energy specific.

Mathematically, a power model can be defined as a function that maps one or several variables related to the system states to the system's power consumption (or cumulative energy), which takes one or more system state indicators (CPU, memory utilization, etc.) as input and is essentially a function of these input variables (although it could be in more complicated forms like neural nets). As a result, the model produces an estimate of the instantaneous power (or the cumulative energy within a period of time) as its output.

The power consumption of a data center is mainly attributed to three parts: IT equipment (servers, network equipment, storage equipment, etc.), the air-conditioning systems, and supporting infrastructures (lighting and power conditioning systems, etc.). However, in this article, we do not intend to cover the entire system but only focus on cloud servers, and our primary purpose is to present a holistic and fine-grained overview on cloud servers' power acquisition, power models, and modeling methods. The reasons we adopt a relatively narrow scope are twofold. First, we believe that the power consumption by cloud servers is worth conducting a comprehensive study, as it typically reaches 70% to 80% [6], and challenges in server power management arise as features like heterogeneity, virtualization, and workload complexity have been commonly introduced to cloud server clusters. We also cover the power models for different forms of server instances and the hosted applications. Second, power/energy consumption by other IT equipment and cooling devices is admittedly important, and we have seen several surveys (e.g., [2, 7]) that cast their light on the data center level. However, despite the broad scope they cover, a thorough and comprehensive review on servers' power consumption is still missing, which is exactly the motivation of our work.

Currently, server power models have shown broad utility in research work related to data center energy efficiency [7]. Although there have been a variety of existing models available for cloud servers and extensive research on the power breakdown of servers and cloud applications, a systematic review of power models regarding the entire "component-server-application" hierarchy and a comprehensive comparison between existing models are still missing. Power models at multiple granularity (from the digital circuit level to data center level) are summarized in the review by Dayarathna et al. [2], but so far there is a lack of a comprehensive survey that covers the entire hierarchy of server power monitoring from the underlying hardware level to the application level. To this end, we present a comprehensive survey and apply a taxonomy on the power collection methods, power consumption models, and power modeling methods for cloud servers in a bottom-up manner, covering the collection methods of power-related data, component-level power models, server instance-level power models, power models for VMs and containers, and the state-of-the-art modeling methods. We also discuss the limitations of existing techniques, models, and modeling methods, as well as some possible ways of improvement, based on which we provide some insight into future research. The organization of this work is shown in Figure 1.

Our primary focus is on the cloud server in this work. Cloud servers are distinct from ordinary physical servers in many features. First, they are typically set up in numbers while being heterogeneous in a cloud data center. This requires the collection methods to have decent scalability, have compatibility to different machine models, and be easy to deploy with low deployment cost. Second, a cloud server can be bare-metal or a virtualized instance like a VM. The feature of virtualization makes it challenging to acquire the server's power when it is running on top of a hypervisor potentially with several other instances. This also causes difficulties in building power models for virtualized server instances since the sharing and contention of resources make an impact on their power behavior. Moreover, most cloud servers are not dedicated to a monopolized workload, and the uncertainty in the load pattern deteriorates a power model's accuracy if it has not been trained/fitted using a suitable model in a proper way. In view of these properties of cloud servers, we start with power data collection in this survey and then introduce a wide range of power models as the second part. The third part of this article focuses on the methods of power modeling with which one can establish a purpose-built server power model and conduct training

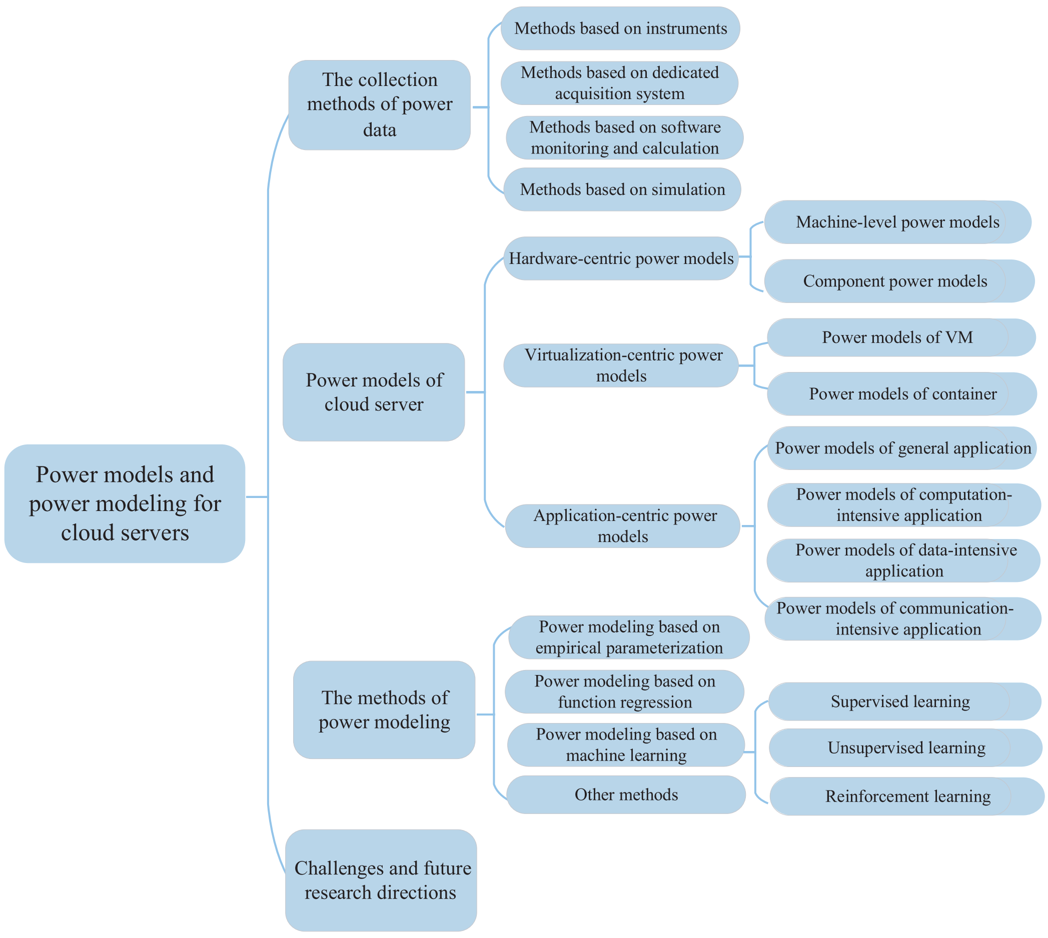

Power models and power modeling for cloud servers
The collection methods of power data
Methods based on instruments
Methods based on dedicated acquisition system
Methods based on software monitoring and calculation
Methods based on simulation
Power models of cloud server
Hardware-centric power models
Machine-level power models
Component power models
Virtualization-centric power models
Power models of VM
Power models of container
Application-centric power models
Power models of general application
Power models of computation-intensive application
Power models of data-intensive application
Power models of communication-intensive application
The methods of power modeling
Power modeling based on empirical parameterization
Power modeling based on function regression
Power modeling based on machine learning
Supervised learning
Unsupervised learning
Reinforcement learning
Other methods
Challenges and future research directions

Fig. 1. Overview of the taxonomy on power data collection methods, server power models, and power modeling methods.

with a right method. Particularly, we cast more light on emerging methods like the artificial neural network (ANN) and reinforcement learning (RL).

The main contributions of this survey are as follows:

- We present comprehensive guidelines on power data collection methods and review a variety of existing power models based on the hierarchy of cloud server instances. A broad scope of studies in the area are discussed with necessary analysis on their usability, applicability, and limitations.
- Considering the expertise required in modeling, we summarize a wide range of both traditional methods and emerging methods like ANN and RL, introduce their mathematical foundations, and analyze their limitations and suitable scenarios in a detailed manner.
- We figure out several open challenges covering critical issues including the application of power models to modern production environments with new virtualization techniques (e.g., containers and unikernel) and novel power modeling methods using complicated architectures like deep neural nets. These insights are provided as guidance for future

research directions especially on realizing energy-aware cloud data center management at the server level.

The main purpose of this survey is to provide guidance to the monitoring of power/energy consumption by cloud server instances, so the topics we cover can be of particular interest to CSPs. We opt not to delve further into the hardware design of bare-metal servers at the circuit level because there is already extensive work on summarizing this scope of research (e.g., [2]), and we believe that virtual cloud server instances, such as VMs and containers, deserve equal or even more attention. In addition, we review several lightweight, application-centric power models that are suitable for monitoring the power/energy of applications running on the cloud. Li et al. [8] investigated the full life cycle of applications involving all resources from the device side to the cloud side and present a variety of application energy consumption models. The part of our survey regarding application-centric models (Section 3.3) is distinct from their work in several aspects. First, we set our perspective of study on CSPs and model the power and energy impact of applications on cloud servers only. Second, the target problem of our work is the estimation of instant power consumption, whereas most of the models presented by Li et al. [8] are energy consumption models established on the assumption that the power cost $P(R)$ of the relevant resources R is known. By contrast, we organize our survey by introducing power data collection methods as the first part, followed by a review of lightweight, hardware-centric power consumption models in Section 3.1. These two leading parts of content provide ways to obtain power values (i.e., $P(R)$) using practical metering methods (Section 2) or theoretical estimation (Section 3.1). In this way, we not only outline different granularities of energy/power modeling but also associate the application-centric energy/power modeling with hardware-centric models (especially component power models) logically. This sets our work apart from other relevant surveys in terms of integrity.

2 THE COLLECTION METHODS OF POWER DATA

The collection of power data is the first step when it comes to establish and train/fit a power model for a cloud server. In this section, we investigate the techniques related to data collection from cloud servers. Existing methods are presented in four categories: methods based on instruments, methods based on dedicated acquisition system, methods based on software monitoring and calculation, and methods based on simulation.

2.1 Methods Based on Instruments

Traditionally, the main idea of instrument-based direct measurement methods is to obtain the power data through an external power metering device. It is often required to connect the corresponding power metering device to each server under test to collect instant power (or accumulated energy for some meters) data.

Implementation. Regarding the way how the instrument is plugged into the system, methods of data acquisition can be categorized into two types: through external devices and internal devices. The difference is shown in Figure 2. The typical implementation of power metering with external devices (e.g., power meter, the combination of galvanometer and voltmeter, and specialized power modules) is to connect them to the upstream of the power distribution unit (PDU). The typical implementation of power data sampling with internal devices is installing a specialized circuit or module between the PDU and the motherboard. Currently, a great number of power/energy metering devices are available, including internal modules like PowerMon2 and external metering devices like OmegaWatt and WattsUp Pro. Table 1 summarizes the general features of external and internal power acquisition devices.

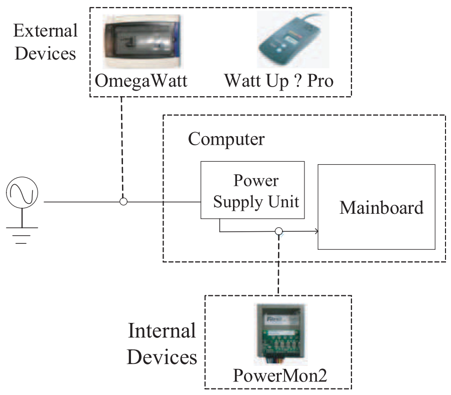


Fig. 2. The installation of external devices and internal devices.

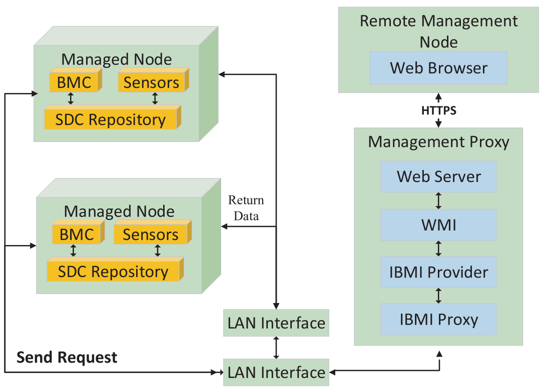


Fig. 3. The dataflow in remote power data collection methods.

Table 1. General Features of External and Internal Power Acquisition Devices

	Type of Data Measured	**Installation**	**Accuracy**	**Sampling Rate**	**Data Granularity**
External Meters	Average power	Plug and play	High	Medium	Machine
Internal Modules	Instantaneous power	Interventional	High	High	Machine/component

Advantages and applicable scenarios.

- *Good compatibility*: External devices are easy to install and set up for basically any type of bare-metal machines. Many servers are compatible with the ATX (Advanced Technology Extended) specification, and thereby metering with internal devices is applicable in these cases.
- *High accuracy*: Instrument-based data collection methods can obtain very accurate power data, no matter if it is directly measured power or derived from voltage and current values.
- *Necessity as the baseline*: It is necessary to compare the real power consumption data obtained by direct measurement with the predictions and estimates by some model during experiments.

Drawbacks.

- *Deployment and measurement difficulties*: Easy deployment and fine-grained measurement are difficult to achieve at the same time. In the actual test system, since the test object can be an independent working node or a working node cluster, coarse-grained power monitoring at the server level is easy and low cost. But if we want a finer granularity of power information (e.g., component level), we have to resort to more hardware-specific (i.e., less generic) instrument-based measurement to obtain the power data from the circuit level, which means that the deployment difficulty will be relatively high as servers are typically heterogeneous in the cloud.
- *Poor scalability*: Installing power metering devices on each and every server in a cloud data center is prohibitive and difficult for management.

2.2 Methods Based on a Dedicated Acquisition System

Some manufacturers have developed specialized power data acquisition systems for their own server products. Such systems usually run on a server cluster and can directly obtain or estimate the entire cluster's information including the status of each work node and power consumption. But they are generally customized by the developers and may require specific hardware compatibility.

Implementation. The baseboard management controller (BMC), shown in Figure 3, is a typical dedicated acquisition system usually integrated with the motherboard as a part of the intelligent platform management interface (IPMI). It can be connected to the system bus, sensors, and a number of components to provide power and temperature information about the CPU, memory, LAN port, fan, and BMC itself. The remote power data collection host first sends a data acquisition request service to the LAN port, then it is received by the BMC. The BMC updates the specific information of the sensor and stores it into the SDR data warehouse. The latest information in the SDR data warehouse is transmitted to the remote management host via BMC and LAN connections in turn.

Some comprehensive management systems have been further developed based on BMC. Such systems are generally customized by the manufacturer, and the BMC is used to obtain the operation status of each device and perform visualization. For example, the Huawei iBMC for Huawei servers is able to collect the CPU temperature, power consumption, and fan speed through the IPMI [9]. Similar systems include Dell iDRAC and HPE Power Advisor.

Advantages and applicable scenarios.

- *High accuracy*: Dedicated acquisition systems are system softwares operating closely with the underlying hardware. Thus, they can typically provide high quality, fine-grained power data on compatible hardware.
- *Ideal for homogeneous servers*: These data acquisition solutions are tightly bound to a series of server models and thus can be equipped for a cluster of homogeneous servers easily.

Drawbacks.

- *Poor compatibility*: Most production-level monitoring systems are designed for a limited number of server models with poor compatibility across different vendors and manufacturers. Sometimes it could also be difficult in deployment on servers in the same model but with different specifications. More importantly, the inter-connection between several monitoring systems (which is very practical in large-scale data centers) could cause a lot of trouble.

2.3 Methods Based on Software Monitoring

The main idea of the software-based monitoring calculation method is to estimate the power consumption by modeling the relationship between the system's power and some critical system state indicators at the level of software. The common software monitoring tools include Joulemeter, CloudMonitor, and DEM [10].

Implementation. The basic steps of methods based on software monitoring generally include the following. First, obtain the critical system state–indicating information (e.g., resource utilization or event counters) through the system APIs. Next, establish a model to express the system's power consumption using the indicators. In general, the most important components (e.g., CPU, cache, disk, and network interface) are first considered for modeling and a lot of empirical analysis is required. Then, estimate the power consumption of the system with the built model.

A series of power monitoring software tools have been developed, such as Joulemeter, which is based on the work by Kansal et al. [3] in 2010 and originally designed for modeling VMs' power consumption. Joulemeter works completely at the software level based on constantly collecting

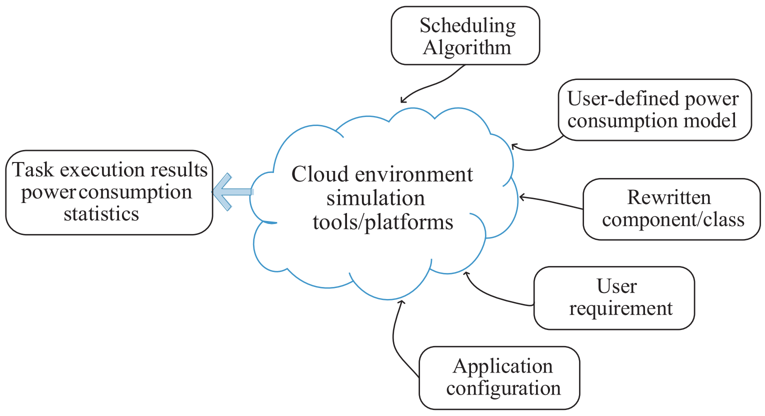

Scheduling Algorithm
User-defined power consumption model
Rewritten component/class
User requirement
Application configuration
Cloud environment simulation tools/platforms
Task execution results power consumption statistics

Fig. 4. General modules incorporated in a cloud simulation tool/platform.

the utilization of various components like the CPU and memory. It has been successfully applied to various operating systems (OS). Similar to Joulemeter, CloudMonitor [11] is a power monitoring software developed by Bohra and Chaudhary [12] and designed for predicting bare-metal machines' power consumption.

In our previous work [10], we introduced the implementation of DEM, a distributed power monitoring system as a prototype for the power measurement in a cloud server cluster. The implementation is based on multiple component-level power models that are adaptive to heterogeneous hardware and compatible with cross-platform cloud environments (e.g., Linux and Windows NT).

Advantages and applicable scenarios.

- *Fine-grained measurement*: Some power measurement software can provide component-level power estimation as they typically decompose a server's power into several parts, each of which accounts for the power contributed by a certain component.
- *Suitable for virtual instances*: Products like Joulemeter do rely on hardware-specific information and only work on the OS-level resource counters, which are generally available in virtual instances like VMs and containers.

Drawbacks.

- *No guarantee in accuracy*: The essence of software-based monitoring methods is the underlying power models that are usually specified by their developers. This makes them very generic but meanwhile could cause high error in case users lack necessary expertise of how to set up the software on their servers. For instance, the monitoring software may fail to self-calibrate in some circumstances and requires manual parameter setup.

2.4 Methods Based on Simulation

Different from the preceding methods, the simulation-based method obtains the power data by simulating the environment of cloud data centers. Many tools support the monitoring of task, job, or machine-level power and energy consumption throughout the emulation, but its credibility depends largely on the simulation program design. Figure 4 abstractly describes in-flow and outflow of such tools or platforms.

Implementation. Currently, most cloud simulation frameworks are based on the CPU to establish the power model and estimate the power consumption of large-scale data centers. For example, the linear power model, square power model, cubic power model, and square root power model have been integrated in CloudSim [13] to help establish a user-defined power model. These models are used for describing the power curve of servers using different mathematical forms for different situations and demands. For example, the linear models require a relatively small amount of data for fitting, whereas the square and cubic power models can be more accurate but also susceptible to over-fitting with the increase in the number of parameters [14]. CloudLighting [15] adopts a cubic polynomial power model, which formulates the average server power consumption as a third-degree polynomial function of the average CPU utilization. Hsu and Poole [14] have pointed out that the trend of the server power curve is clearly deviating from linearity. Considering the phenomenon, introducing non-linear terms to the power model can better fit servers' power curves and in turn helps reduce error. The details about the definitions and comparisons of these power models are provided in Section 3.

Advantages and applicable scenarios.

- *Ideal for early experiments*: Studies and projects on energy-efficient data centers must go through extensive evaluations, but there will be a huge time and monetary cost if we repeat them on real servers with real meters. By contrast, a simulated environment greatly shortens the experiment and development cycle given that the emulated results (e.g., energy consumption) are basically reliable.
- *Not limited by hardware resources*: Using simulation tools is not limited by hardware resources, which means that simulating a large-scale cluster (especially regarding its power behavior) can be very useful before one decides to really deploy more servers. The results of simulation are not real but could be informative for decisions like scaling out.

Drawbacks.

- *Low credibility*: The simulation-based environment only considers very limited factors compared to a real cluster. Sometimes the obtained data by simulation probably has a large deviation from the real cases.

2.5 Comparison of Power Collection Methods

There is no one-size-fits-all method in power data collection from cloud servers. The best choice depends on the application scenario and the specific requirements by a cloud server provider. For example, when it comes to quantitatively evaluating a power model to check its accuracy, the collection method based on the instrument is necessary. But if we want to evaluate the power of a VM, since the measurement methods based on the instrument cannot provide any VM-wise power data, we need to resort to the collection method based on software monitoring and calculation. To provide clear guidance, in Table 2 we compare the four power consumption collection methods from multiple perspectives, including how to implement them, suitable targets, the difficulty of deployment, data granularity, and data credibility.

It should be noted that multiple power collection methods can be combined in practice. For instance, when designing a resource scheduling algorithm of a VM, the simulation technology can be used to efficiently verify and refine the algorithm. When the algorithm enters the stage of practical test, it is necessary to directly obtain the power using some real-world instruments as the baseline.

3 POWER MODELS OF THE CLOUD SERVER

The power (consumption) model is essentially a function that maps the variables related to the system's state to the system's power consumption. A power model usually takes one or multiple

Table 2. Comparison of Four Power Collection Methods for Cloud Servers

Methods	Implementation	Suitable Targets	Deployment Difficulty	Data Granularity	Data Credibility
Based on instrument	Installation of extra devices	Bare-metal machines	Easy	Machine level	Very high
Based on dedicated acquisition system	Specialized systems	Specific models of machines	Difficult	Machine or component level	High
Based on software monitoring	Built-in power models	Bare-metal and virtual servers	Moderate	Machine, component, or VM level	Fair
Based on simulation	System simulation	Machine, component, or VM level	Easy	Machine, component, or VM level	Low

system state indicators (CPU, memory utilization, etc.) as the function's independent variables and takes the instantaneous power (or the cumulative energy within a period of time) as its output. Power models work as the foundation of software-based and simulation-based power monitoring mechanisms, which have gradually become prevalent in today's data center management. Using power models enables flexible, scalable, and low-cost power estimation and prediction in large-scale, heterogeneous cloud server clusters.

In this section, we present a comprehensive review on the existing power models for the cloud server. Based on the target and the applicable type of cloud server, we summarize these models into three categories: hardware-centric power models, virtualization-centric power models, and application-centric power models.

3.1 Hardware-Centric Power Models

3.1.1 Machine-Level Power Models. This type of model is used for estimating the power consumption of a physical machine and is applicable to the scenarios where our target is a bare-metal server. Beloglazov et al. [7] consider the power consumption of a entire server as the sum of two terms called *static power consumption* and *dynamic power consumption*, which can be formulated as

$$P_{server} = P_{static} + P_{dynamic}. \tag{1}$$

Similar to CMOS circuits, the static power consumption is mainly incurred by the current leakage, whereas the dynamic power consumption is mainly caused by the charging and discharging of capacitors [2].

The more common way to model the power of a physical server is considering it as an assembly of functioning components. For example, Song et al. [16] proposed to divide the total power consumption of a server into multiple major components including the CPU, memory, disk, and network interface card (NIC):

$$P_{server} = P_{cpu} + P_{memory} + P_{disk} + P_{NIC}. \tag{2}$$

Similar models include those of Kansal et al. [3] and Tudor and Teo [17] where more or less components are taken into account.

Inspired by the power characteristics of CMOS circuits, some previous research (e.g., [18]) adopted frequency-based power models to estimate a server's power:

$$P_{server} = c_0 + c_1 f^3, \tag{3}$$

where c_0 is a constant representing the base power of a physical server, c_1 is a constant related to the capacitance and voltage of the CPU, and f denotes the operating frequency of the CPU. It is worth noting that this power model is actually derived from the basic CMOS circuit power model $P = ACV^2 f$, where A is an activity factor accounting for the frequency of gate switching, C is the total capacitance at the gate outputs, and V is the voltage of the processor.

It is intuitive that a server's power is basically proportional to the workload it takes, which leads to a straightforward way to formulate power consumption as a function of resource utilization of the server. Empirical studies have shown that CPU usage/utilization has a strong correlation with the server's power [14]. Considering this, a great number of studies use CPU utilization as the only indicator of server power and propose diversity forms of univariate models. Fan et al. [19] proposed a power model that is essentially a linear interpolation between the idle power and full (max) power of a server:

$$P_{server} = P_{idle} + (P_{max} - P_{idle})u_{cpu}, \tag{4}$$

where P_{idle} and P_{max} denote the power consumption when the server is idle and fully utilized, respectively, and u_{cpu} represents CPU utilization. The model implies that the server power consumption grows linearly as CPU utilization increases from 0% to 100%. Considering that this model may oversimplify the power curve of a server, they further proposed another model in a non-linear form:

$$P_{server} = P_{idle} + (P_{max} - P_{idle})(2u_{cpu} - u_{cpu}^{\gamma}), \tag{5}$$

where γ is a parameter fitted by minimizing the square error of the model on training data. Through experiment, Fan et al. [19] pointed out that the error of Equation (5) is reduced from 5% to 1% by introducing the non-linear term. Compared to Equation (4), this non-linear model performs better in tracking the dynamic power consumption of a server. However, the determination of parameter α at the exponent may be difficult, to which a potential solution could be empirical analysis on a large power dataset.

More non-linear power models with similar ideas have been studied. Rivoire et al. [20] refined the preceding model by adding more trainable parameters to improve accuracy:

$$P_{server} = \alpha_0 + \alpha_1 u_{cpu} + \alpha_2 u_{cpu}^{\gamma}, \tag{6}$$

where α_i and γ are model parameters that need to be determined during fitting/training. Hsu and Poole [14] proposed another complex model and proved its accuracy based on their empirical study on a large public server power dataset—SPECpower_ssj2008. Based on their experiment observations covering server power data released from December 2007 to August 2010, they suggested adopting two non-linear terms in the power model:

$$P_{server} = \alpha_0 + \alpha_1 u_{cpu} + \alpha_2 u_{cpu}^{\gamma_0} + \alpha_3 (1 - u_{cpu})^{\gamma_1}. \tag{7}$$

3.1.2 Component Power Models. The main limitation of machine-level power models is that we can only evaluate the overall power consumption of a cloud server, and it is hardly useful when we need to know how much power is consumed by each component inside the server. Figure 5 displays the proportion of power consumed by major server components for a server in a data center owned by Google.

In this article, the components of physical server refer to the power-consuming parts in a server that are not independent or not working independently, such as the CPU, memory, disk, and NIC. From the breakdown of a typical server's power consumption (Figure 5), it is clear that the CPU, memory, and disk are the main contributors to the power consumption of a server. With little change to the computer architecture since the prevalence of commodity machines, the distribution of power is expected to remain basically the same in the near future. In the following contents, we cast the focus on these major components and summarize a wide range of component power

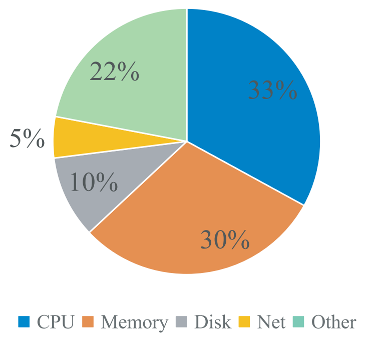


Fig. 5. Typical breakdown of the physical server's power consumption [21].

models of the CPU, memory, disk, and NIC, with which the overall power consumption of a server can be formulated as

$$P_{server} = P_{cpu} + P_{mem} + P_{disk} + P_{NIC} + C, \tag{8}$$

where P_{cpu}, P_{mem}, P_{disk}, and P_{NIC} represent power consumption of the CPU, memory, disk, and NIC, respectively. C denotes the server's base power, which includes the power consumption of other components regarded as a static part.

CPU power models. Generally, the CPU is the primary power consumer in a server, especially under computation-intensive workloads. In many cases, the power behavior of a server mainly depends on the power characteristics of its CPU. CPU utilization is the most commonly used measure of the proportion of non-idle CPU time slices, which effectively reflects the workload intensity on the CPU. CPU utilization is easy to obtain at the OS level, so most studies adopt CPU utilization for estimating the CPU's power consumption.

A very prototypical form of the CPU power model is the linear model. For example, Fan et al. [19] parameterized a simple CPU power model as

$$P_{cpu} = \alpha_0 u_{cpu} + \alpha_1, \tag{9}$$

where both α_0 and α_1 are parameters that need to be obtained through training or fitting, and u_{cpu} denotes CPU utilization. When the maximum power and idle power of the CPU are known, the model can be further simplified:

$$P_{cpu} = P_{cpu}^{idle} + \left(P_{cpu}^{max} - P_{cpu}^{idle}\right) u_{cpu}, \tag{10}$$

where P_{cpu}^{idle} and P_{cpu}^{max} represent the idle power consumption and maximum power consumption, respectively. A problem that stands out for the model is that its usability counts on the fixed values of the CPU's idle and max power, which, however, can change over time if frequency scaling technologies (e.g., DVFS) are applied. Basmadjian et al. [22] observed that the power consumed by each core on a multi-core processor is the same as that by a single-core processor. Inspired by this observation, they proposed an accumulative form to calculate the power consumed by the CPU:

$$P_{cpu} = P_{cpu}^{idle} + \sum_{i=1}^{n} P_{core}^{i}, \tag{11}$$

where n represents the number of CPU cores and P_{core}^{i} represents the power consumed by the i-th core.

Non-linear models are also widely adopted to estimate the power consumption of CPU use based on its utilization. For example, Luo et al. [23] proposed a polynomial model as a univariate function of CPU utilization:

$$P_{cpu} = \alpha_0 + \alpha_1 u_{cpu} + \alpha_2 u_{cpu}^{\gamma} + \ldots, \tag{12}$$

where α_i is the coefficient of the polynomial expression. In case $\gamma = 2$, the model is a quadratic polynomial model. In theory, the power curve of any complexity can be fitted by this model if we add as many terms (with higher order) as needed. But at the same time, the more terms and parameters the model has, the more prone to over-fitting it becomes.

In addition, some research models the power consumption through investigating the internal structure of the CPU. For instance, Basmadjian and De Meer [24] proposed an additive processor power model:

$$P_{cpu} = P_{mc} + P_{dies} + P_{intd}, \tag{13}$$

where P_{mc} is the power of chip-level mandatory components, P_{dies} represents the power of constituent dies, and P_{intd} stands for the power of inter-die communication. Sarood et al. [25] considered the impact of cache and memory access on the power consumption of each CPU core and formulated a CPU power model as follows:

$$P_{cpu} = P_{core} + \sum_{i=1}^{3} g_i L_i + g_m M + P_{base}, \tag{14}$$

where P_{core} is the idle power of CPU, L_i represents the number of cache accesses, g_i represents the unit cost of cache access, $g_m M$ represents the cost of memory access, and P_{base} denotes the static package power.

Although various forms of models are available for estimating the CPU's power consumption, we find two potential issues. On the one hand, CPU utilization, as the most popular indicator of CPU workload, in some cases cannot truly reflect how "busy" the processor is because it is typically computed based on the number of non-idle time slices, which could remain high when it is actually working on moving data around without much computation. On the other hand, existing models rarely consider the correlation between multiple cores, which could bring a significant impact on the CPU's power. A possible direction of improving existing models is to take the intra-core load sharing policy into account.

Memory power models. The role of memory is critical in both computation and power consumption. It has been figured out that in peta-scale systems, the main memory consumes about 30% of the total power [26]. The activities of memory closely correlate to the multi-level caches in modern server architecture. Since cache misses will trigger memory operations, some studies (e.g., [3]) calculate the power consumed by memory based on the number of the last-level-cache misses (LLCMs) in a unit of time, which is called the *LLCM-based memory model*:

$$P_{mem} = P_{mem}^{idle} + \alpha \cdot N_{LLCM}, \tag{15}$$

where α is a training parameter and N_{LLCM} denotes the number of cache misses. The principle behind this is the strong correlation between LLCMs and the intensity of memory page swapping activities. From a different perspective, Basmadjian et al. [22] proposed to describe the power model of SDRAM memory by mainly considering the state of memory:

$$P_{mem} = P_{mem}^{idle} + P_{mem}^{dynamic} = \sum_{i}^{n} s_i \cdot p + 7.347 \cdot C \cdot \gamma, \tag{16}$$

where the idle power of memory is defined as the sum of each memory module's size s_i times a constant p that depends on the type and vendor of the memory module (a list of example p values are provided in the work of Basmadjian et al. [22]). C is a constant whose value also depends on the specific memory type (e.g., $C = 2.3$ for buffered DDR2), and $\gamma \in [0, 1]$ reflects the memory state and can be obtained with a probabilistic approach.

The preceding memory power models are designed from the angle concerning how the memory works, but using these models requires access to some low-level counters in the first place. For example, the counter of LLC (e.g., LLC-load-misses and LLC-store-misses) is a kind of hardware event counter gathering information from the processor, but not all types of OS kernels provide (direct) access to it. Considering the parameters N_{LLCM} and γ of Equation (15) and Equation (16) that are both lower-level counters, they can more accurately reflect the memory activity and be more helpful for constructing highly accurate models. But this, however, results in relatively poor applicability to heterogeneous server architectures, as only a limited portion of them provide relevant interfaces. In addition, frequently acquiring LLCM events can cause considerable overhead to the system.

Lightweight approaches to the estimation of memory power is required in many situations where accuracy is not the primary demand. Lin et al. [10] proposed using the memory footprint to characterize the memory activity and estimated the power consumption of memory using a simple linear model:

$$P_{mem} = P_{mem}^{idle} + \alpha \cdot u_{mem}, \tag{17}$$

where α is a training parameter and u_{mem} is the memory footprint, namely the memory usage. Although there is no direct relationship between the memory footprint and memory activity, it is very likely to experience frequent memory access and physical page swapping in case a large portion of memory space is occupied.

Another intuitive idea is to directly use the frequency of memory access to characterize the current power consumption of memory. For example, the works of Bohra and Chaudhary [12] and Arroba et al. [27] modeled the power consumption of memory by using the following linear formula:

$$P_{mem} = P_{mem}^{idle} + \alpha \cdot s, \tag{18}$$

where α represents the training parameter and s is the number of memory accesses in a unit of time. The number of memory reads and writes requested by the CPU is recorded by specialized performance counters invisible to ordinary OS users, so the model needs the support from some special performance monitoring tool (which has to be installed as an extra module into the kernel) to read the counter's value.

Arroba et al. [27] divided the power consumption of memory into idle power and dynamic power. They suggested that the dynamic power consumed by memory is proportional to the frequency of memory accesses, whereas the idle power is related to the working temperature. Considering the impact of DVFS technology, the proposed memory power model is formulated as

$$P_{mem}(k) = \alpha_1 \cdot T_{mem} + \alpha_2 \cdot T_{mem}^2 + \alpha_3 \cdot f_{mem}(k), \tag{19}$$

where T_{mem} is the operating temperature of memory and $f_{mem}(k)$ represents the frequency at which memory is accessed when the server is in the DVFS state denoted as mode k. Model parameter α_i is obtained via training.

Disk power models. A great portion of power consumed by a hard disk drive (HDD) comes from its mechanical operations (e.g., the spinning plates and the moving read-write heads). Due to the lack of visibility into the hard drive's power state and the complicated impact of disk hardware caching techniques, the power behavior of the disk is difficult to catch [3]. Zhang et al. [28] suggested that the power consumed by the disk can be divided into a idle part and a dynamic part, whereas the dynamic power of the disk can be modeled based on its work state. Basmadjian et al. [22] introduced a disk state-based power modeling method with a simple disk power model:

$$P_{disk} = a \times P_{access} + b \times P_{idle} + c \times P_{startup}, \tag{20}$$

where they consider three states of a disk: accessing, idle, and startup. Accordingly, a, b, and c represent the probability that the disk is in one of these three states ($a + b + c = 1$), respectively. The idle and start-up power of the disk (i.e., P_{idle} and $P_{startup}$) are reference values from the manufacturer's data sheet.

Allalouf et al. [29] pointed out the idle power of disk accounts for about two-thirds of the maximum power, and the dynamic power of disk is closely related to its I/O request rate. They used a quadratic polynomial function to describe the power characteristics of an HDD:

$$P_{disk} = P_{disk}^{idle} + \alpha_1 q + \alpha_2 q^2, \tag{21}$$

where both α_1 and α_2 are training parameters and q is the request rate for disk I/O. Disk throughput or I/O speed (often measured in megabytes per second) is another indicator that reflects the intensity of disk operations in real time. In view of this, Kansal et al. [3] proposed a throughput-based power model where they consider the read and write speed of disk separately:

$$P_{disk} = P_{disk}^{idle} + C_r m_{read} + C_w m_{write}, \tag{22}$$

where m_{read} and m_{write} represent the read and write speed of disk, respectively. C_r and C_w depend on disk specifications as the coefficients for read and write operations, respectively. At the same time, they also observed that the difference in power consumption between read and write on an HDD is basically negligible, and thus the disk model based on throughput can be simplified to a linear model with only one parameter:

$$P_{disk} = P_{disk}^{idle} + \alpha_0 m, \tag{23}$$

where α_0 is the model parameter and m denotes the current throughput of disk, namely the sum of its read speed and the write speed.

In our previous work [10], we made use of a benchmarking toolkit—IOmeter—to investigate the power behavior of an HDD under different data transfer rates. We found surprisingly different power behaviors of the disk in different work modes. Specifically, in random I/O mode, disk power consumption is basically proportional to its throughput (in megabytes per second), and meanwhile a greater difference between the read and write operation ratio leads to higher power consumption by the disk. When the disk is working in sequential I/O mode, its power is almost independent on the read/write operation ratio. Based on the observations, we presented a solution based on differentiating the work mode (i.e., sequential I/O or random I/O) of a disk. The proposed disk model can be formulated as follows:

$$P_{disk} = \begin{cases} \alpha_{seq} \cdot S \ \text{ if } S > H_S \text{ or } O > H_O, \\ \alpha_{rnd} \cdot S \ \text{ otherwise}, \end{cases} \tag{24}$$

where

$$S = S_{read} + S_{write}, \tag{25}$$

$$O = O_{read} + O_{write}, \tag{26}$$

where α_{seq} and α_{rnd} represent the average disk power at the throughput of 1 MB/s in sequential and random I/O mode, respectively. Variable S is the I/O speed (i.e., throughput combining read and write), O is the number of disk operations per second, S_{read} is the I/O speed of read (in megabytes per second), S_{write} represents the I/O rate of write, O_{read} represents reading operations per second (in times per second), O_{write} represents the operations writing per second, and the thresholds H_O and H_S are used to determine the work state of the disk.

Network interface power models. Gandhi et al. [30] pointed out that the power consumption of peripheral devices can reach 23% of the total power consumed by a server. In communication-intensive applications, the power consumed by the NIC is not negligible. Gupta et al. [31–33] figured out the importance of network energy consumption. It has been found that most Ethernet interfaces can stay in a low-power mode, but most of the them are not really in an energy-saving state when they are not working. We have also seen protocol-level (e.g., TCP) studies on the energy saving mechanism. For example, Gunaratne et al. [34] proposed a G-TCP connection scheme, which adds a shim layer between the data transport layer and the application layer as a probe to control the devices at both ends of the TCP protocol connection. Blanquicet and Christensen [35] proposed a G-SNMP energy-saving management protocol, which collects network device energy consumption information into a library by adding additional information to the SNMP protocol for real-time management and estimation of power consumption.

Focusing on the NIC itself, Basmadjian et al. [36] proposed that the interface can either be idle or in the dynamic mode in any given time slice, and they used P_{NIC}^{idle} and $P_{NIC}^{dynamic}$ to represent its idle power and dynamic power, respectively:

$$E_{NIC} = P_{NIC}^{idle} T_{idle} + P_{NIC}^{dynamic} T_{dynamic}, \tag{27}$$

where T_{idle} and $T_{dynamic}$ represent the time duration that the network interface is in idle mode and dynamic mode, respectively. Given a period of time $T = T_{idle} + T_{dynamic}$, the average power consumption of the NIC in time T is formulated as

$$\begin{aligned} P_{NIC} &= \frac{\left(T - T_{dynamic}\right) P_{idle} + P_{dynamic} T_{dynamic}}{T} \\ &= P_{idle} + \left(P_{dynamic} - P_{idle}\right) \frac{T_{dynamic}}{T}. \end{aligned} \tag{28}$$

In the model, the impact on the NIC's power by the work mode of the underlying network specifications is taken into account. In addition, the mode and the communication pattern in which the network component is working also affect the utilization of other components like the CPU. For example, in case of serial P2P communication, a remarkable portion of CPU time slices will be used for communication-related instructions, which could significantly increase the CPU's utilization and power. By contrast, embedded network solutions (e.g., Infiniband) are likely to transfer a heavy communication load to the embedded architecture and thereby put less pressure on the CPU.

We have seen extensive studies on hardware-centric power models. However, it can be observed that the interactions between the components are not considered in most research at either the machine level or the component level. Considering the complexity of the workload on a cloud server, we believe that it is necessary to investigate the correlation between the power consumption of different components and take that into account. For example, the power characteristic of the CPU could be entirely different in memory-intensive (while low disk usage) jobs and in IO-intensive jobs.

3.2 Virtualization-Centric Power Models

3.2.1 *Power Models of the VM.*

Virtualization technology has been prevailing in modern data centers where VMs, as the maturest implementation of virtualization, are usually the actual "servers" that directly run user applications. On this point, the VM is a sort of cloud server—it

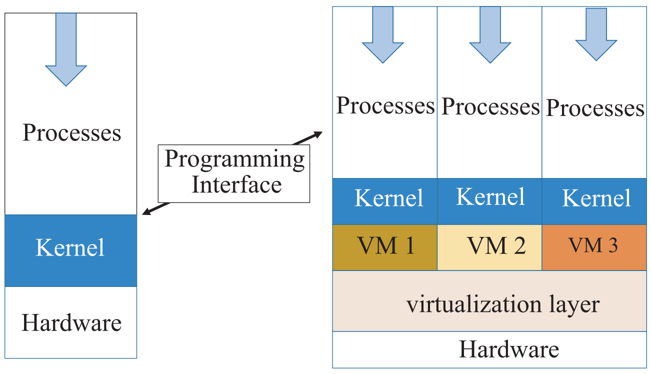


Fig. 6. Comparison of traditional processes running on top of the OS and the architecture of VMs hosted by the server through a virtualization layer.

provides the computation, storage, and network to users independently. When it comes to modeling VMs' power consumption (which could be very useful for service pricing), some studies consider the VM as an independent power-consuming entity (although it is not), whereas others attempt to associate its power behavior to its host machine.

The virtualization layer (on the right-hand side of Figure 6) allows applications to run in a VM just like they do in a real bare-metal machine. Meanwhile, it also provides several standard interfaces (which may differ between hypervisors) from which we can monitor the resource usage (e.g., CPU utilization) of each VM.

However, power metering at the hardware layer cannot be applied directly to sampling VMs' power data. In current research on VM power consumption, there are basically two methods to obtain VM power data: a white-box method and a black-box method. The white-box method is to implant a probe into the VM and obtain information from inside the VM, whereas with the black-box approach, we regard a VM as a process and gather its power information from the host.

Most existing studies adopt the white-box method and generally need to set up a monitoring probe into the VM to gather information from inside the VM instance. For example, Li et al. [37] proposed a method for modeling the power consumption of each VM the in case where there are n VMs on the server (each VM denoted by VM_i):

$$P_{server} = P_{static} + \alpha_0 \sum_i^n U_{VM_i}^{cpu} + \alpha_1 \sum_i^n U_{VM_i}^{mem} + \alpha_2 \sum_i^n U_{VM_i}^{IO} + ne, \tag{29}$$

from which one can derive the power consumption of a single VM on this server:

$$P_{VM_i} = P_{static} + \alpha_0 U_{VM_i}^{cpu} + \alpha_1 U_{VM_i}^{mem} + \alpha_2 U_{VM_i}^{IO} + e, \tag{30}$$

where P_{static} represents the baseline power, and $U_{VM_i}^{cpu}$, $U_{VM_i}^{mem}$, and $U_{VM_i}^{IO}$ represent the CPU utilization, memory usage, and disk throughput of VM_i, respectively. Parameters α_0, α_1, and α_2 are weights that need to be trained offline, n is the number of VMs, and e is the adjustable residual term.

With a similar approach, Kansal et al. [3] proposed another component-based VM power model:

$$P_{VM_i} = \alpha_0 U_{VM_i}^{cpu} + \alpha_1 N_{VM_i}^{LLCM} + \alpha_2 b_{VM_i}^{IO}, \tag{31}$$

where $U_{VM_i}^{cpu}$ represents the CPU utilization of VM i, $N_{VM_i}^{LLCM}$ represents the LLCMs caused by VM_i, and $b_{VM_i}^{IO}$ represents rate of I/O requested by the VM. Parameter $\alpha_i (i = 0, 1, 2)$ needs to be trained offline. Bohra and Chaudhary [12] used inside-the-VM monitoring daemons and base their model on the pairwise relations CPU, cache and disk, DRAM according to their empirical study. They first presented an additive model including the power consumption by all co-hosted VMs:

$$P_{sum} = C_1\left(\alpha_1 + \alpha_2 P_{cpu} + \alpha_3 P_{cache}\right) + C_2(\alpha_4 + \alpha_5 P_{DRAM} + \alpha_6 P_{disk}), \tag{32}$$

where C_1 and C_2 are constants determined by observing the average increase in total power consumption when an idle machine runs CPU- or I/O-intensive jobs, respectively. Parameters α_1 and α_2 represent the base power when the VM is idle, whereas α_2, α_3, α_5, and α_6 represent the weights of corresponding power-consuming sources. Then the authors derived the power model for an individual VM:

$$P_{VM} = \alpha P_{VM}^{cpu} + \beta P_{VM}^{Cache} + \gamma P_{VM}^{DRAM} + \omega P_{VM}^{Disk}, \tag{33}$$

where P_{VM}^{cpu}, P_{VM}^{Cache}, P_{VM}^{DRAM}, and P_{VM}^{Disk} denote the corresponding resource usage by the target VM. Coefficients α, β, γ, and ω are training parameters.

Different from the linear models introduced previously, Versick et al. [38] and Wabmann et al. [39] proposed adopting a polynomial model to estimate the dynamic power consumption of a VM via the information of the CPU, HDD, and NIC. The comprehensive model is formulated as

$$\begin{aligned} P_{VM} &= P_{CPU} + P_{HDD} + P_{NIC} + P_{Static} \\ &= \alpha_1 x_C + \cdots + \alpha_m x_C^m + b_C \\ &\quad + \beta_1 x_H + \cdots + \beta_m x_H^m + b_H \\ &\quad + \gamma_1 x_N + \cdots + \gamma_m x_N^m + b_N + P_{static}, \end{aligned} \tag{34}$$

where x_C, x_H, and x_N denote the features that indicate the utilization of the CPU, HDD, and NIC, respectively. For each component, a series of polynomial terms (from order 1 to m) are used to model its power, and the corresponding parameters are α_k and b_C (for P_{CPU}), β_k and b_H (for P_{HDD}), and γ_k and b_N (for P_{NIC}). With the model, the authors claim that the error can be reduced to 3.1% when the order m is set to 6.

Peng et al. [40, 41] also proposed a power model by considering a series of lower-level power features such as the uOps (micro-operations), halted cycles, LLCMs, translation look-aside buffer (TLB), front-side bus (FSB), interrupts, and DMAs. In the proposed model, all information acquired in a fixed time period and the overall power consumption of the VM is the sum of the power consumption of each component:

$$\begin{aligned} P_{VM} &= P_{cpu} + P_{mem} + P_{disk} + P_{IO}, \\ P_{cpu} &= uOps - Halt^2, \\ P_{mem} &= LLC + TLB + FSB, \\ P_{disk} &= Interrupt + DMA^3, \\ P_{IO} &= Interrupt + DMA. \end{aligned} \tag{35}$$

Compared with utilization-based modeling methods, this model requires reading low-level counters from the kernel, and the availability of these counters in a VM is dependent on the hypervisor.

In our previous research [42], we found that from the perspective of the host, the power consumption curves of VMs with different core numbers are very similar. According to this observation, we adopted a lightweight approach and proposed a VM power model (CAM) by considering the vCPU configuration and the current vCPU utilization of the VM. The core specification-aware VM power model can be formulated as follows:

$$P_{vm}(u_v, n) = \alpha \cdot \left(\frac{n}{N}\right)^{\gamma} \cdot u_v^{\gamma}, \tag{36}$$

where both α and γ are model parameters tunable during training, u_v represents the current vCPU utilization of the VM, and n and N represent the number of cores assigned to the VM and the total number of physical cores in the host machine, respectively. Similar to the machine-level power model, the v-core specification-aware VM power model adopts a non-linear form so that it can characterize the non-linear VM power curve. In addition, it is adaptive to different vCPU configurations on the same host machine through a unified expression with no need for retraining.

Phung et al. [45] proposed a lightweight, combinatorial linear power model to evaluate the power consumption of a VM. They divided the power into three terms: idle power, additional power due to workloads, and temperature-dependent additional power use led by increased fan speeds:

$$P_{total} = P_{idle} + P_{work} + P_{heat}(T), \tag{37}$$

where $P_{heat}(T)$ denotes the extra power caused by excessive heat dispersion as a function CPU temperature T. The other two terms are formulated as

$$P_{idle} = aF + bF^2 + c,$$

$$P_{work} = \sum_{k=0}^{n} P_k = \sum_{k=0}^{n} (fF + eF^{3/2} + fF^2)\mu + g\alpha + h\beta,$$

where F is the effective frequency of the CPU, and a, b, c, d, e, f, g, and h are coefficients to be determined. P_k is the power consumed by running workload k (with a total number of n workloads), μ is the number of micro-instructions issued, α is the number of LLC read misses, and β is the number of TLB misses.

Despite the similarity between a VM and a real machine, virtualization overhead has been the major limitation of VMs and is an important factor that existing VM power models hardly take into account. The functioning of a VM hypervisor does consume power, and it is worthwhile to delve into how and to what degree it accounts for each VM's power consumption among all co-hosted instances.

3.2.2 Power Models of the Container. Being regarded as a more lightweight approach to resource isolation than the VM, containers and server containerization (Figure 7) have gained increasing popularity driven by the uprising trend of micro-services in web applications.

Being more process like and less machine like, containers make it more difficult to model their power consumption than VMs. One of the reasons is the life cycle of each container being quite uncertain, which means that each container can be running long-term applications (e.g., background processes or management processes) or short-term jobs (e.g., temporarily online services). The second reason relates to the nature of containers in that they are frequently scheduled by the orchestration system and typically are not bound to servers. This somehow makes it difficult to identify and associate a particular container to any host to determine its power behavior. In addition, many orchestration systems (e.g., Docker Swarm and Kubernetes) manage containers in groups (called *Pods* in Kubernetes), and the monitoring interfaces they provide may not be fine grained enough for power monitoring at single-container level.

So far, there are only a few studies related to modeling the power behavior of containers. Considering the similarity between multiple containers, it is important to study the characteristics of a group of containers via machine learning (ML) or statistical methods. For instance, Kang et al. [43] proposed a container power model based on k-medoids clustering. The proposed model first needs to calculate the power consumed by n containers under different workload types in a standard test pool. Then it uses the server specification and container characteristics as attributes and

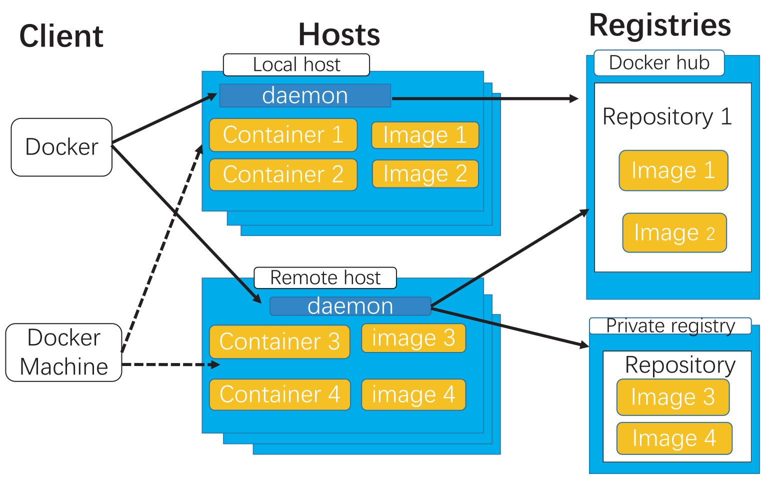


Fig. 7. Diagram illustrating the placement of containers on bare-metal hosts and the creation of containers based on remote images. Docker containers are used here as an example.

applies the k-medoids to cluster the containers in the test pool. Finally, when a new container is brought online, the similarity (e.g., Euclidean distance) between the container and each of the resulting medoids is calculated, and the power consumption of each medoid will represent all containers' power in its corresponding cluster. The model effectively solves the problem caused by a large number containers in heterogeneous environments through clustering. However, two aspects may significantly influence its effectiveness in accuracy: (1) the size of the test pool used for generating the reference clusters and (2) the coverage of each cluster and potential bias in each medoid.

Targeting at fully containerized servers, Piraghaj et al. [44] considered multi-level virtualization wherein containers are consolidated into VMs, which are hosted by bare-metal servers. To serve their power optimization objective, they used the following formula to estimate the power consumption of a containerized server:

$$P_{sum} = \begin{cases} P_{idle} + (P_{max} - P_{idle}) \cdot U & if N_{VM} > 0, \\ 0 & if N_{VM} = 0, \end{cases} \tag{38}$$

where

$$U = \sum_{j=1}^{N_{VM}} \sum_{k=1}^{N_c} U_{c(j,k)}, \tag{39}$$

where $U_{c(j,k)}$ is the CPU utilization of container k in VM j hosted by the server. In the first equation, we can reckon $P_{max} - P_{idle}$ as the dynamic part of the host's power consumption, whereas in the second equation, the authors assume that the CPU utilization is completely attributed to the containers running on the server. Combining the two equations and given the utilization of container k (running in VM j), we can derive the underlying power model for this container as

$$P_{j,k}^{container} = P_{idle} + (P_{max} - P_{idle}) \cdot U_{c(j,k)}. \tag{40}$$

Phung et al. [45] proposed a RAPL model, which is a linear function of the RAPL CPU package energy counter:

$$P_{container} = (aE_{CPU} + b) \times \alpha, \tag{41}$$

where a and b are coefficients to be determined, E_{CPU} is the amount of energy consumed by the CPU package, and α is the CPU utilization of container c.

In addition, Fieni et al. [46] introduced a lightweight power monitoring system SMARTWATTS that adopts online calibration to automatically adjust the CPU and DRAM power models to maximize the accuracy of runtime power estimation of containers. They pointed out that given power model M^f_{res} from aggregated events, $E^f_{res} = \sum_{c \in C} E^f_{res}(c)$, the power consumption of any container c by applying the inferred power model M^f_{res} at the scale of the container's events $E^f_{res}(c)$ can be predicted:

$$\hat{p}^{dyn}_{res}(c) = M^f_{res} \cdot E^f_{res}(c),\ f \in F, \tag{42}$$

where $\hat{p}^{dyn}_{res}(c)$ denotes dynamic power consumption of the monitored resource of container c, f represents the given frequency, F represents the available frequencies, M^f_{res} is derived from a Ridge regression linear least squares regression with L2 regularization, and E^f_{res} is the hardware performance counter (HwPC) reading.

Tadese et al. [47] proposed a power model for Docker containers running benchmarking applications including CPU-intensive and network-intensive tasks. Linear regression and second-order polynomial interpolation are used to fit the model on the observed power values:

$$P^{cpu}_{container} = 0.1065u + 12.4411, \tag{43}$$

$$P^{net}_{container} = -17.7368r^2 + 37.2859r + 3.8210, \tag{44}$$

where P refers to the power consumed by Docker, u means the percentage of CPU usage (as an addition by cores), and r means utilization of the data rate. The authors came to the conclusion that the CPU is not the only contribution to container power consumption.

Considering the features of container as a form of lightweight virtualization and its potential of becoming prevalent, it is worthwhile to carry out deeper research on how to effectively model containers' power consumption. First, we believe that it is an indispensable effort to associate a container's power behavior to that of its host. In addition, we believe that potentially this can be done from two perspectives. For one, the resource contention and virtualization overhead should be taken into account, which also applies to the power modeling of VMs. For the other, it is worth further study as to whether it is possible to consider Pod (a basic unit consisting of multiple containers) as a whole in a large-scale scenario (e.g., Kubernetes) where containers are monitored and managed in groups.

3.3 Application-Centric Power Models

Application is another angle from which we can look into the power consumption of a cloud server. On the one hand, extensive research has shown the difference in a server's power characteristics when hosting different types of applications. On the other hand, application-level power monitoring is often of great use when it comes to service pricing and resource scheduling. Therefore, in this section, we summarize a broad range of application-centric power models by first categorizing them into the power models for general applications, computation-intensive applications, data-intensive applications, and communication-intensive applications.

3.3.1 Power Models of General Applications. A great portion of the applications running on cloud servers are not easy to be identified as computation-intensive, data-intensive, or communication-intensive applications. A common feature of general applications is that they produce a mixed workload instead of imposing an extreme single type of pressure like floating point operations or disk I/O. Smith et al. [48] developed CloudMonitor for cluster power monitoring, which uses a process that monitors system resource utilization to measure power consumption. In this work, they suggested using a single power model for different types of workloads:

$$P_{app} = P_{idle} + \alpha_1 P_{cpu} + \alpha_2 P_{mem} + \alpha_3 P_{hdd} + \alpha_4 P_{net}, \tag{45}$$

where P_{idle} is a constant base value, and P_{cpu}, P_{mem}, P_{hdd}, and P_{net} represent the power consumption by the application from utilizing the CPU, memory, disk, and network, respectively. Weight coefficient α_i $(i = 1, 2, 3, 4)$ is the parameter for training.

Wang et al. [49] specified more sources of power consumption in their application model as follows:

$$P_{app} = \alpha_1 u_{cpu} + \alpha_2 cmr + \alpha_3 cer + \alpha_4 nie + \alpha_5 f_{cpu} + P_{idle}, \tag{46}$$

where α_k $(k = 1, 2, 3, 4, 5)$ is training parameter, and u_{cpu} and f_{cpu} represent the CPU utilization and frequency, respectively. By cmr, cer, and nie, they denote the cache miss rate, context exchange ratio, and number of instructions executed in a unit cycle caused by the application, respectively.

Koller et al. [50] proposed an application throughput-based power model, in which they observed that there is a linear relationship between the dynamic power of any application and its throughput. The linear model has a simple form and can be formulated as

$$P_{app} = c_0 + c_1 \cdot tr, \tag{47}$$

where tr is the throughput rate of the application. Both c_0 and c_1 are constants depending on the application. The authors suggest calibrating c_0 and c_1 in separate runs for each application.

3.3.2 Power Models of Computation-Intensive Applications. The computation-intensive applications mainly refer to those running programs that feature a large amount of computation, such as floating point operations. Looking into the problem by regarding an application as a set of tasks, Chen et al. [51] suggested that the task-related factors and system configuration directly affect the total energy needed to finish the task. They proposed a task-level model for estimating the energy demand of a given computation-intensive task by mainly considering the data size of each task, system configuration, and some other factors:

$$E^i_{app} = f_{comp}\left(PT^i_{comp}, DS^i_{comp}, DT^i_{comp}, C^i_{comp}\right), \tag{48}$$

where E^i_{app} represents the energy consumption for executing the i-th computation-intensive task, and f_{comp} is a function of PT^i_{comp}, DS^i_{comp}, DT^i_{comp}, and C^i_{comp}, which are the number of processes, size of data to be processed, size of data for transmission, and system configuration, respectively.

With the focus on the entire workflow of applications, Gamell et al. [52] investigated the energy consumption caused by every operation in the workflow and proposed a model as follows:

$$E_{app} = \frac{P_{cpu}}{C} \cdot I_s \cdot V \cdot \left(t_{prod} + \frac{t_{cons}}{I_a}\right), \tag{49}$$

where P_{cpu} represents the dynamic power of CPU, I_s represents the number of simulation steps, I_a represents the number of simulation steps between two analyses, V represents the number of variables, and t_{prod} and t_{cons} represent the time to produce a variable and consume a variable, respectively.

Colmant et al. [53] investigated how to estimate power consumption at the process level. They proposed a CPU frequency and unhalted cycles based power model that is suitable for computation-intensive applications:

$$P^{app}_f(uc_{pid}) = \frac{C_1}{10^9} uc_{pid} - \frac{C_2}{10^{18}} uc^2_{pid}, \tag{50}$$

where f is the frequency of the CPU, and C_1, C_2 are frequency-dependent constant parameters. The variable uc_{pid} represents the number of unhalted cycles count by process (pid). They specify the parameters for a Xeon (f =2.90 GHz) processor: $C_1 = 8.64$ and $C_2 = 6.10$.

Leite et al. [54] studied the energy characteristic of computation-intensive programs and introduced an analytical prediction based on their experimental measurements of instruction-level energy cost. Their model can be easily converted to estimating an application's power consumption as a function of CPU frequency f given a short time interval Δt:

$$P_{app}(f) = \sum_{i \in INS} \frac{N_i \cdot e_i(f)}{\Delta t} + P_{cooling}(f, T_{max}), \tag{51}$$

where INS denotes the set of atomic instructions and N_i represents the number of executions of the i-th type of instruction. With $e_i(f)$, they record the per-instruction energy cost at CPU frequency f. T_{max} is the temperature threshold for the cooling system, the power of which is excluded because it cannot be attributed to a single application.

3.3.3 Power Models of Data-Intensive Applications. Data-intensive applications tend to process massive amounts of data and thereby incur very frequent I/O requests. According to the type of operation, a data-intensive workload can be divided into online data-intensive workloads and offline data-intensive workloads [55]. Currently, most of the existing models for data-intensive loads fall into the second type, within which a great number of these models are built for estimating the power consumption of MapReduce applications. Therefore, in the following, we first summarize power models for general data-intensive workloads and then present some special focus on MapReduce applications widely deployed as a very typical offline workload.

Poess and Nambiar [56] proposed to separately consider the power impact of I/O-intensive applications on the server and any external storage attached. They performed experiments using the TOC-H benchmark of database applications and first introduced a server-level model for any servers running database applications:

$$P_{app}^{server} = 1.3\,(N_c P_c + 9N_m + N_{di} P_d) + 100, \tag{52}$$

where P_{app}^{server} represents the power consumption of the server, N_c is the number of CPUs used, P_c is the value of thermal design power (TDP) consumption of the CPU, N_m denotes the number of memory DIMMs, N_{di} represents the number of internal disks, and P_d represents the power consumption of the disk. The authors further provided a power consumption model for estimating the power of the I/O subsystem as follows:

$$P_{app}^{io} = 1.2 N_e \cdot N_{de} \cdot P_{de}, \tag{53}$$

where N_e denotes the number of chassis, N_{de} represents the external disks in each chassis, and P_{de} stands for the power consumed by each external disk. Therefore, the power consumption of the whole system under a data-intensive workload can be formulated as

$$P_{app} = P_{app}^{server} + P_{app}^{io}. \tag{54}$$

At present, many data-intensive applications are comprised of multiple interrelated tasks, which in turn make up a workflow and typically run on distributed servers. For example, Gamell et al. [52] proposed an intricate model to estimate the energy required to finish a data-intensive workflow:

$$E_{app} = V \cdot I_s \left(\left(t_{mem}^{st} + \frac{t_{mem}^{ld}}{I_a} \right) \cdot P_{mem} + \left(t_{stg}^{st} + \frac{t_{stg}^{ld}}{I_a} \right) \cdot P_{stg} + \left(t_{net}^{st} + \frac{t_{net}^{ld}}{I_a} \right) \cdot P_{net} \right), \tag{55}$$

where V represents the number of (workflow) variables; I_s represents the number of simulation steps; I_a represents the number of simulation steps between two analyses; t_{mem}^{st}, t_{stg}^{st}, and t_{net}^{st}

represent the data storage time from the memory, staging, and network, respectively; t^{ld}_{mem}, t^{ld}_{stg}, and t^{ld}_{net} represent the data loading time from the memory, staging, and network, respectively; and P_{mem}, P_{stg}, and P_{net} represent the dynamic power of the memory, staging, and network, respectively.

One of the most popular models in batch data processing is the MapReduce paradigm. For a MapReduce processing task, Zhu et al. [57, 58] proposed an auto-regression-based power model, in which the power consumption of a single worker is estimated by the following function:

$$P_{app}(t) = \alpha_0 P'_{data}(t-1) + \alpha_1 \Delta x(t), \tag{56}$$

where both α_0 and α_1 are parameters that require training, $P'_{data}(t-1)$ is the measured power consumption at time $t-1$, and $\Delta x(t)$ denotes the arrival rate threshold. Specifically, in this work, the authors proposed obtaining α_0 and α_1 via the least squares estimation method along with a technique called *exponential forgetting training*. They also adopted an estimation module that continuously monitors each server under a dynamic workload.

Lang and Patel [59] introduced an approach to estimate the energy demand of a MapReduce application by quantifying the energy consumed by each computation phase:

$$E_{app} = P_i T_i + P_m T_m + P_s T_s + P_r T_r, \tag{57}$$

where $P_i T_i$, $P_m T_m$, $P_r T_r$, and $P_s T_s$ correspond to the initial phase, map phase, reduction phase, and shift phase of a MapReduce application, respectively.

3.3.4 Power Models of Communication-Intensive Applications. Communication-intensive applications typically need to exchange a large number of messages between processes. The message passing interface (MPI) has attracted extensive studies as a widely used API for developing high-performance computing (HPC) applications that require frequent messaging. Messages can be sent in multiple ways, among which broadcasting is a typical information transmission method and a great number of communication-intensive power model have been designed for it. For example, Diouri et al. [60] proposed a method for power consumption estimation for a particular broadcast algorithm under various execution configurations. They presented two models for estimating the power of a server node and a switch, respectively:

$$p^{server}_{app} = p^{server}_{idle} + \delta p^{server}, \tag{58}$$

$$p^{switch}_{app} = p^{switch}_{idle} + \delta p^{switch}, \tag{59}$$

where p^{server}_{idle} represents the idle power consumption of server, p^{switch}_{idle} represents the power consumption when the switch j is powered on without any workload, and δp^{server} and δp^{switch} are the mean extra power costs caused by the high-level operations. Further, they provided a sophisticated method for estimating the energy consumption of running Scatter & AllGather (SAG) algorithms with an MPI:

$$\begin{aligned} E^{MPI/SAG}_{app} &= \sum_{i=1}^{N} p^{node_i}_{sag} + \sum_{j=1}^{M} p^{switch_j}_{sag} \\ &= t_{scatter}(N,M,v)\left(\sum_{i=1}^{N} p^{node_i}_{scatter}(v) + \sum_{j=1}^{M} p^{switch_j}_{scatter}\right) + t_{allgather}(N,M,v)\left(\sum_{i=1}^{N} p^{node_i}_{allgather}(v) + \sum_{j=1}^{M} p^{switch_j}_{allgather}\right), \end{aligned} \tag{60}$$

where N, M, and v represent the number of compute nodes, switches, and number of processes per node, respectively. The work divides the energy consumption in the scattering and aggregation stages into two independent and symmetric terms, respectively.

Gamell et al. [61] investigated the energy consumption produced during the transmission process of a data-intensive application. They built a model for estimating the total energy required for communication:

$$E_{app}^{comm} = \begin{cases} \sum_{i=1}^{M} \frac{data_i}{BW_{net}} P_{transfer} & \text{if } smp(src_i) \neq smp(dest_i), \\ \sum_{i=1}^{M} \frac{data_i}{BW_{mem}} \left(P_{cpu}^{active} + P_{mem}^{active}\right) & \text{otherwise.} \end{cases} \tag{61}$$

where BW_{net} and BW_{mem} represent the bandwidth of the network and memory, respectively. The condition $smp(src_i) \neq smp(dest_i)$ indicates that the MPI ranks i and j are mapped to the cores that do not share memory. P_{cpu}^{active} and P_{mem}^{active} denote the dynamic power of the CPU and memory, respectively. $P_{transfer}$ is the power consumed in data transfer and depends on the network characteristics. An issue of the model is that the authors do not take network/memory contention into account.

Although the research on application-centric power models has somehow been extensive, many problems remain that need further consideration. For example, given so many application type-specific power models and generic models, there is a lack of empirical study to compare their effectiveness in different situations with different combinations of workload and load intensity. In addition, the power behavior of applications (of different types) probably differs on cloud servers wherein basically three cases of contention can be expected: monopolized servers (little contention), similar applications on one host (moderate contention), and disparate applications on one host (fierce contention). For more accurate power modeling, applications' power behavior in each case needs further investigation.

4 THE METHODS OF POWER MODELING

In many cases, existing power models may be too general to be sufficiently accurate on a specific type of machine that exhibits sophisticated power consumption behavior. Moreover, a limited number and type of variables are considered in most of the models we survey, resulting in poor flexibility in applying existing models to completely different specifications of servers with different hardware and software configurations. These problems motivate a lot of exploration in how to establish an effective power model based on the power data we possess. In this section, we summarize both classical and emerging power modeling methods and the corresponding implementation of model training and hyper-parameter tuning.

Most of the modeling methods we introduce in this section take the machine state-related information (e.g., the utilization of CPU, memory and I/O, and other various indicators) as the features. It is worth noting that the modeling methods vary in the number (and even types) of features they can take as input: some of them only accept one or two variables, whereas some other methods (e.g., neural nets and support vector regression (SVR)) can theoretically cover an arbitrary number of feature variables.

4.1 Power Modeling Based on Empirical Parameterization

In the case where collecting a sufficient amount of power data (for training/fitting) is not feasible and the accuracy of the resulting model is not extremely important, we can determine all of the model parameters based on the server's specification, a limited number of measurements, or even experience. The parameters determined in these ways are referred to as rated parameters, measured parameters, and empirical parameters, respectively. Specifically, the rated parameters refer to the official values labeled by the manufacturer. They are product specific and are usually results of calibration by the vendors. Typical rated parameters include the rated power consumption of servers, max frequency of memory modules, TDP of CPUs, and so forth. Measured parameters refer to the parameters whose values are obtained through measurement in the actual experiment. For example, the idle and full power of a blade server can be measured with a power meter. The empirical parameters mainly refer to the parameters determined on the basis of existing knowledge and experience we gained from empirical practice by others. Such knowledge can be a set of power data released online.

With the increasing availability of server power-related data (most of which have been verified by authorities and organizations), it has become possible to determine the power model parameters totally based on the existing knowledge and data obtained by a third party. The work of Hsu and Poole [14] is a good example, in which they comprehensively studied the power characteristics (also called *signature* by the authors) of servers completely based on a large public dataset named SPECpower_ssj2008. Verified and published seasonally by SPEC, the dataset contains power (and performance) data of a variety of servers submitted by mainstream manufacturers like Huawei, IBM, and Dell. Hsu and Poole [14] suggested that a non-linear, utilization-based server power model is the best fit for the data in SPECpower_ssj2008 ranging from years 2007 to 2010. With only measured and empirically determined parameters, the model they proposed is formulated as follows:

$$P(u) = P_{idle} + (P_{max} - P_{idle})u_{cpu}^{0.75}, \tag{62}$$

where 0.75 is the empirical parameter as a conclusion from their analysis. Clearly, the main advantage of the power modeling based on empirical analysis is that it is simple to use and easy to configure. Although it is not realistic to expect high accuracy from these models, they can be a considerable option in case we cannot perform in situ experiments or we are unable to collect sufficient training data for some reason.

4.2 Power Modeling Based on Function Regression

Regression analysis is one of the most conventional methods we use to quantify the relationship between the target variable (i.e., server power, in our case) and several feature (or explanatory) variables (CPU utilization, cache misses, etc.). It can be single-variable regression or multi-variable regression depending the dimension of input, whereas for multi-variable regression we usually resort to Lasso regression or ridge regression for data fitting. In the following, we introduce how to build and train regression-based server power models using statistical approaches.

The objective function (or the criterion) of regression models is generally defined as the mean absolute error (MAE) or some extended form based on MAE. Therefore, most existing studies apply the least squares estimation to obtain the optimal model parameters given a set of data. For instance, Kansal et al. [3] proposed constructing separate models for estimating the power of the CPU, memory, and disk using linear regression with the least squares estimation method. A similar way of training the regression method is adopted by Lin et al. [62], in which they fitted multiple component power models.

Model fitting is more intricate for the non-linear power models. Luo et al. [23] evaluated different kernel function forms in an attempt to estimate server power with a multivariate nonlinear power model. They adopted and compared different training methods (which are the objective functions essentially) such as the polynomial with Lasso:

$$\min_{\alpha_0, \alpha_1} \left[\frac{1}{2N} \sum_{i=1}^{N} \left(y_i - \alpha_0 - x_i^T \alpha_1 \right)^2 + \lambda ||\alpha_1|| \right], \tag{63}$$

where α_0 and α_1 represent the model parameters that need to be trained using Lasso regression and λ represents the regularization coefficient. The main purpose of the penalty term $\lambda||\alpha_1||$ is to constrain the parameter space to make it sparse, which effectively prevents over-fitting.

Arguing that least squares estimation cannot guarantee a smaller maximum error, Hsu and Poole [14] proposed using the maximum absolute percentage error as the error metric, which is defined as

$$\max_{u} \left| \frac{P_u}{P(u)} - 1 \right|, \tag{64}$$

where P_u represents the actual power of server at utilization u and $P(u)$ is the estimated value by the power model. The authors claimed that when a prediction error threshold is set, the trusted range of the model estimation should be gradually narrowing as the utilization u decreases (because the power consumption of a server is low when under-utilized), and thereby the advantage of this metric is that it ensures the model has a small prediction error over the entire range of utilization.

After collecting a relatively large number of server component-level features, Zhou et al. [63] proposed adopting principal component analysis (PCA) to reduce the dimension of the feature space, based on which they evaluated multiple regression models including a multivariable linear regression model, a power regression model, and an exponential regression model. We denote them by P_{lin}, P_{pow}, and P_{exp}, respectively. The expressions are shown as follows:

$$P_{lin} = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_m x_m + \varepsilon, \tag{65}$$

$$P_{pow} = \alpha_0 \cdot x_1^{\alpha_1} \cdot x_2^{\alpha_2} \cdot x_3^{\alpha_3} \cdot \cdots \cdot x_m^{\alpha_m} + \varepsilon, \tag{66}$$

$$P_{exp} = \alpha_0 \cdot e^{\alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_m x_m} + \varepsilon, \tag{67}$$

where α_i $(i = 1, 2, 3, \dots)$ and ε are parameters that need to be trained and x_i $(i = 1, 2, 3, \dots)$ are the input features extracted using PCA.

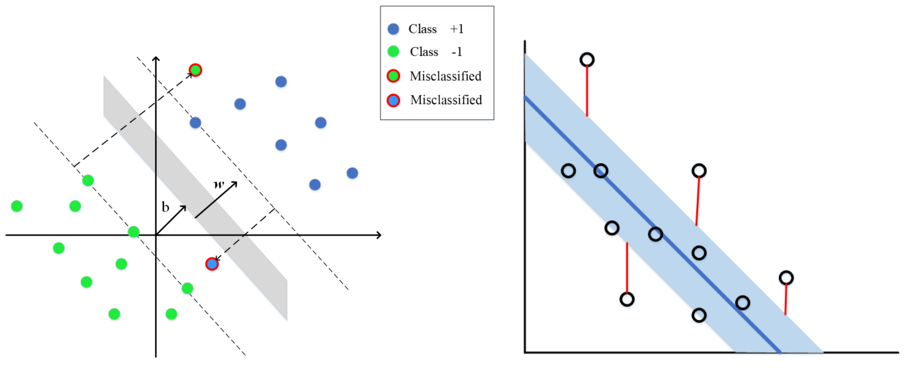
4.3 Power Modeling Based on ML

With the rapid advances in ML, it has been widely used in many research areas, one of which is the application of ML methods in power modeling and forecast [64–66]. In the following, based on our survey on relevant methods that have been successfully applied to cloud server modeling, we introduce the most representative approaches in the area and compare them with regard to usability and applicability.

In general, there are three classes of ML-based methods according to the statistical foundation based on which they learn from the data. With this in mind, we categorize existing ML-based power modeling approaches into the following types: supervised learning methods, unsupervised learning methods, and RL methods.

4.3.1 Power Modeling Based on Supervised Learning. The task of supervised learning is to learn a model from labeled data so that the model can predict the output from any given input. In the problem setting of power modeling, the input includes any features that can be collected from the system under test, whereas the output is the estimate or prediction of server power. Typical supervised modeling approaches include SVR, ensemble learning, and neural networks.

Support vector regression. The underlying theory of SVR is an extension to support vector machine (SVM), which is a classical learning method in ML. The idea of SVM is to map input from the original space to a high-dimensional feature space to produce a classifying hyper-plane that maximizes the interval between (the support vectors of) two classes (depicted in Figure 8(a)). The learning strategy of SVM is interval maximization, which can be formalized into a problem of convex quadratic programming. Power modeling is essentially a regression problem, but the way we construct the hyper-plane in SVM can be easily extended to performing predictive regression—this



(a) SVM (b) SVR

Fig. 8. Schematic diagrams showing how SVM builds the hyper-plane in a two-dimensional space and how SVM regression can be used to fit power data with a similar statistical idea to SVM.

makes an SVR model. The difference between SVR and SVM is that the training goal of SVM is to keep the points in each dataset away from the boundaries of classification, whereas SVR needs to keep each data point in the training set as close to the hyper-plane (represented by a vector of weights) as possible (Figure 8(b)). The expression of the hyper-plane to fit is formulated as follows:

$$y_i = w \cdot \varphi(x_i) + b, \tag{68}$$

where $\varphi(x_i)$ is the representation of x_i in the mapped space by the kernel. The training of SVR models requires a target function (or loss function) with tolerance to marginal error and robustness to over-fitting. Generally, the construction of the hyper-plane (i.e., the actual resulting power model we desire) is realized by imposing constraints on the optimization of the target function as a convex quadratic problem. For example, Liang et al. [67] proposed using the following target function to find the optimal parameters of their server power model based on SVR:

$$\begin{aligned} &\min \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{I}(\xi_i + \xi_i^*) \\ &\text{s.t. } y_i - \boldsymbol{w} \cdot \phi(\boldsymbol{x}_i) - b \le \varepsilon + \xi_i \\ &\quad \boldsymbol{w} \cdot \phi(\boldsymbol{x}_i) + b - y_i \le \varepsilon + \xi_i^* \\ &\quad \xi_i \ge 0, \xi_i^* \ge 0, \end{aligned} \tag{69}$$

where $\boldsymbol{x}_i$ denotes the input features (as a vector); y_i represents the labels (i.e., measured power of the server); $\boldsymbol{w}$ and b denote the parameters that construct the hyper-plane; ξ_i and ξ_i^* represent the lower relaxation boundary and the upper relaxation boundary, respectively; and C is a pre-set constant for the target function. Luo et al. [23] also built a server power prediction model based on SVR and compared its accuracy with Lasso linear regression and stepwise regression models using the benchmark workload of SPEC CPU 2006. They concluded that SVR attains the highest accuracy in power prediction. A similar approach was adopted by Veni and Bhanu [68] and Salam et al. [69], where the SVR method is applied to modeling VMs' power consumption. SVR is sensitive to the quality and the number of features provided. To further improve SVR-based power models, Yang et al. [70] boosted SVR with feature extraction using PCA.

Ensemble learning. The main idea of ensemble learning is training a number of individual learners and combining them into a more powerful one on a specified learning task. Figure 9 depicts the general framework of ensemble learning, in which we first need to build and train a set of

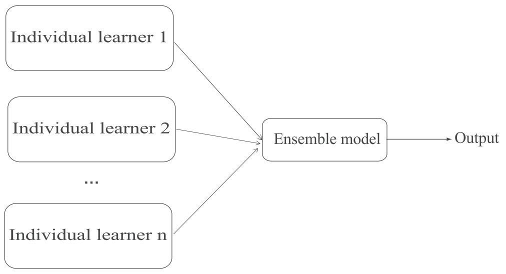


Fig. 9. Brief scheme of ensemble learning.

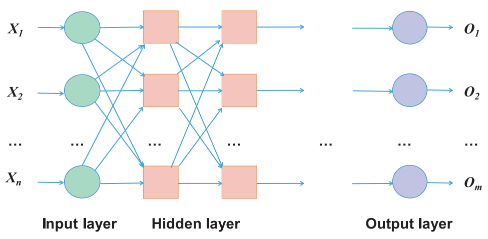


Fig. 10. General architecture of a (fully connected) neural network model.

relatively weak learners (i.e., models) and then combine them using a certain strategy. The advantage of ensemble learning is that integrating multiple learners can effectively mitigate the risk of over-fitting and under-fitting caused by using a single learner [71].

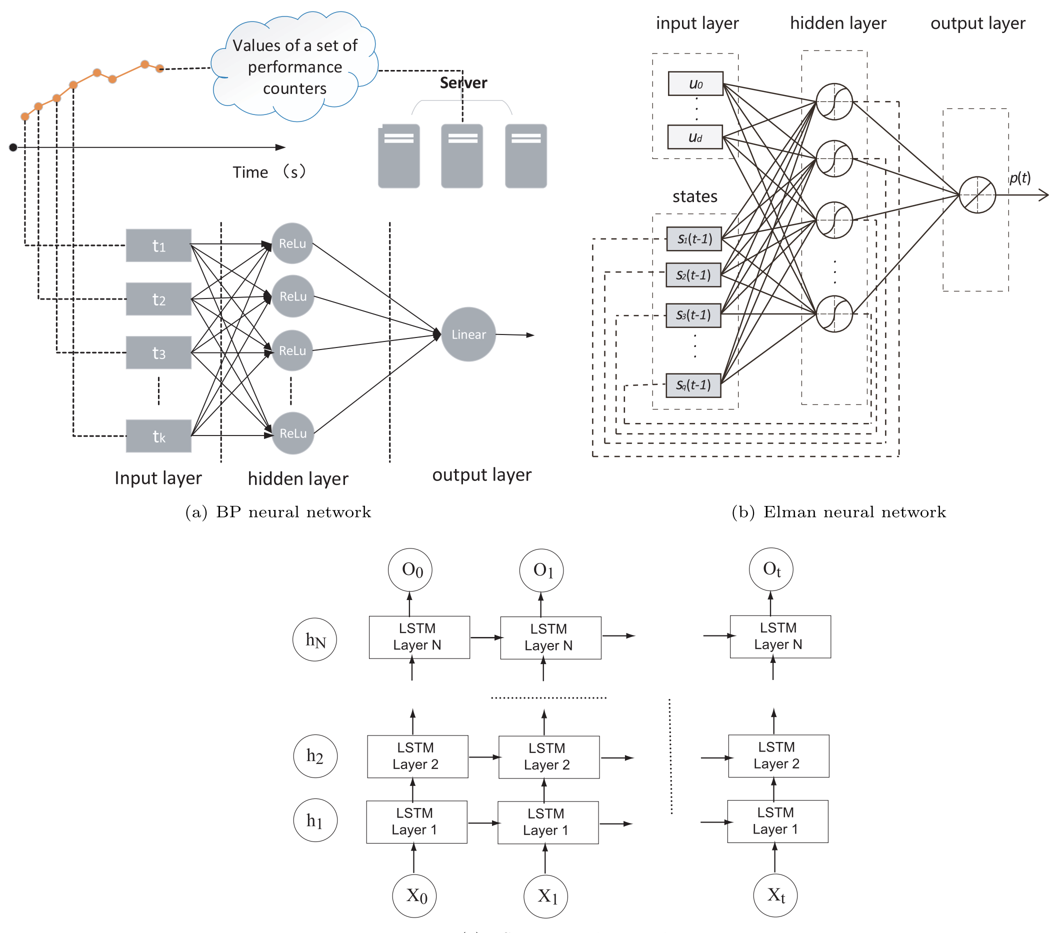
According to the way individual learners are constructed, the existing algorithms of ensemble learning can be roughly divided into two categories: serialized training of learners and parallel training of learners. In the first method, the individual learners are trained one after another so as to produce improving models step by step. This incurs a strong dependency between individual learners [72]. By contrast, parallel training generates usable learners in a parallel manner and integrates all of these weak learners as the final model [71]. Typical algorithms of serialized training and parallel training are the boosting method and the random forests (RF) algorithm, respectively. In the problem of server power estimation/prediction, parallel methods like RF and bagging are currently used more frequently. For example, Harton et al. [73] proposed an RF-based server power estimation framework, where a large number of base predictors are trained and contribute to the final result of prediction.

Lin et al. [10] exploited the utilization of multiple server components as the input features and used R-Square as the metric to investigate the performance of different RF models, which are composed of different numbers of CART trees. The results report that RF can be fairly accurate and robust as a way to build power models, but its major limitation is the long training time required.

Apart from RF, the GBDT, XGBoost [74], LightGBM [75], and many other boosting algorithms are widely used in prediction tasks. These new approaches can be more efficient and easily adapted to server power estimation and prediction.

Neural network. The ANN is an emerging computational connectionist model that is able to learn very complex representations (i.e., the underlying patterns of data) using a net consisting of a large number of neurons connected in form of layers. The most basic form of ANN is the fully connected net, which is also called *multi-level perceptron* (MLP). Despite that many variants have been proposed, the general structure of ANN is shown in Figure 10.

The ANN has been widely applied to complicated regression tasks, among which power estimation and prediction have drawn much attention. Liu et al. [76] built a back-propagation neural network (BPNN) and a long short-term memory neural network (LSTM) to predict the power consumption of a data center using the CPU utilization and memory usage data from Google workload traces. Li et al. [77] proposed a deep neural network based power model that takes into account a set of time granularity. Lin et al. [78] experimentally compared the performance of cloud server power models based on BPNN, Elman neural network, and LSTM (the corresponding power modeling frameworks are shown in Figure 11). Specifically, the power model based on BPNN takes features



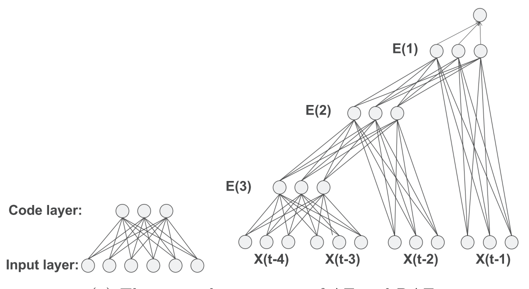
(a) BP neural network (b) Elman neural network

(c) LSTM neural network

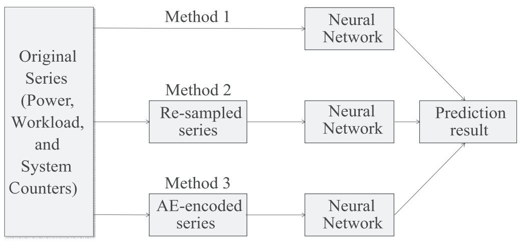
Fig. 11. General architectures of modeling a cloud server's power with the BPNN, Elman neural network, and LSTM neural network.

in a time window as input and predicted server power after being training with back-propagation methods. The Elman neural network (Figure 11(b)) is a type of recurrent neural network (RNN) that incorporates a special layer of memory cells that memorize the hidden state of every forward propagation in the network. As a more complex form of RNN, LSTM (Figure 11(c)) adopts more parameters (and thus more complicated functions) for each neuron to adaptively control the impact by neurons' memory and the instant input. LSTM can outperform non-recurrent forms of neural nets especially in case where there are long-term temporal patterns in the time series of server power.

As two powerful tools for learning the representations of data, the auto-encoder (AE) and recursive auto-encoder (RAE) are often used for both long-interval prediction and short-interval prediction. Their structures are shown in Figure 12(a). Considering the potential availability of a great number of system features in server power modeling and the dependency between one feature and another, both of them can be very useful in feature engineering. For example, if we use a large window of past power values as input, which makes the problem a long-interval autoregression, the better choice is applying RAE to produce highly representative features rather than directly using the raw vector of past power values as the features of our model.



(a) The network structures of AE and RAE



(b) Three implementations in coarse time granularity prediction

Fig. 12. Power consumption prediction model based on AE and RAE.

In addition to the preceding works, neural network based methods have been widely used in resource prediction and load prediction tasks [64, 65, 79–81]. Although the target is different, most of the models used in studies can be adapted to the problem of server power modeling as it is essentially a time series prediction task in the context where we are estimating a server's power given a window of the past measures.

4.3.2 Unsupervised Learning. Targeting ML scenarios without the presence of labels, unsupervised learning algorithms perform error correction and back propagation independently on labeled/annotated data. For example, clustering algorithms (e.g., *K*-means and hierarchical clustering) and the Gaussian mixture model (GMM) have been widely used in unsupervised learning tasks. In power modeling, the output of a model is essentially equivalent to drawing an sample from an estimated distribution of power data. On this point, many relevant studies choose GMM as an unsupervised approach to establishing power models [82, 83].

Gaussian mixture model. The GMM is an extended Gaussian model that uses a linear combination of multiple Gaussian distributions to characterize the data distribution based on a limited number of observations. Each individual Gaussian distribution represents a stochastic factor that makes impact on the observed output, say, the true power. A weight is assigned to each distribution and fitted on a set of observations so that the hybrid of these Gaussian distributions can approximate the real distribution of power. The expression of the GMM can be formulated as

$$P(y|\theta) = \sum_{k=1}^{K} \alpha_k \Phi(y|\theta_k), \tag{70}$$

where $\alpha_k \geq 0$ is the vector of weights (as training parameters) and subject to $\sum_{k=1}^{K} \alpha_k = 1$, $\Phi(y|\theta_k)$ is the density function of (single-variate) Gaussian distribution, which estimates the probability of observing y as follows:

$$\Phi(y|\theta_k) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{(y-\mu_k)^2}{2\sigma_k^2}\right), \tag{71}$$

where $\theta_k = (\mu_k, \sigma_k^2)$. In theory, if the GMM integrates a sufficient number of Gaussian models and the weights are trained properly, the hybrid model of the GMM can precisely fit the true distribution of power data. Dhiman et al. [82] proposed a GMM-based model for estimating the power consumption of VMs. They first partitioned the data into different groups according to the CPU utilization. Then they used a method called *Gaussian mixture vector quantization* (GMVQ) to generate a GMM (the weights, essentially) in each group. Finally, they calculated the distance between multiple Gaussian distributions and selected the closest ones to produce power predictions.

Expectation-maximization. The expectation-maximization (EM) algorithm is designed for determining the optimal set of parameters $(\alpha_k, \mu_k, \sigma_k^2)$ in the GMM to find the proximate distribution to the ground truth (i.e., the real distribution of data). For example, Hao et al. [83] proposed a GMM-based power model and applied the EM algorithm to find the best parameters. In general, the parameter estimation of the Gaussian distribution function only requires the process of maximum likelihood estimation. However, maximum likelihood estimation is challenging on data in the presence of latent variables, which, in the problem of power modeling, can be the unknown factors affecting power behaviors. As a solution, EM performs iterative steps of constructing an expectation (i.e., the likelihood) with estimated latent variables and optimizing the parameters to maximize the likelihood. EM can be used to search for the optimal parameter set of a GMM-based power model by executing the follow steps:

(1) An initial guess is generated for the each individual Gaussian distribution's parameters.
(2) *E-step*: Compute the expectation of the likelihood function of current parameters based on the observed power data.
(3) *M-step*: Perform maximum likelihood estimation to optimize the current parameters in the hybrid distribution.
(4) Repeat E-step and M-step until convergence.

The main strength of using EM to train unsupervised distribution models like GMM is its usability with the presence of missing data and robustness against noisy data.

4.3.3 Reinforcement Learning. Adopting an entirely different philosophy from supervised and unsupervised learning methods, RL does not intend to find a concrete model but targets at building a smart agent that can make "right" decisions. RL has been successfully applied to a wide range of scenarios, such as scheduling, gaming, industrial control, and robotics. The Google artificial intelligence (AI) team further proposed deep reinforcement learning (DRL) by combining RL with deep learning [84], which is considered to be an important way to move toward the general intelligence.

A great portion of existing research uses server power models as a tool but sets their target at cluster or data center-wise energy conservation. One of the applications of RL on a cloud platform is the research on automatic resource scheduling and power management. For example, Farahnakian et al. successfully used an RL algorithm to learn energy-efficient strategies of VM consolidation without prior knowledge about the environment and workload. Liu et al. [86] proposed a hierarchical cloud resource and power management method based on DRL using the framework shown in Figure 13. The framework mainly consists of a global layer responsible for VM allocation and a local layer responsible for local power management. Specifically, in the global layer, DRL is used to solve the resource optimization problems with supplemental techniques like automatic encoder and weight sharing. And in the local layer, LSTM-based workload prediction adopts some model-free RL methods to forecast the server power distribution to support proactive scheduling.

4.4 Other Methods

In this category, we outline some server power modeling approaches that are not commonly adopted in research but still could be inspiring. A very typical approach is the evolutionary algorithms. For instance, unlike the statistical methods, Arroba et al. [27] proposed to use the particle swarm optimization (PSO) algorithm to search for the best parameter settings during the training of a power model. By comparing with the traditional least squares method, they reported that the model trained with the PSO algorithm performs much better in various tests using different benchmark workloads. In addition, Hilburg et al. [87] developed a model for dynamically

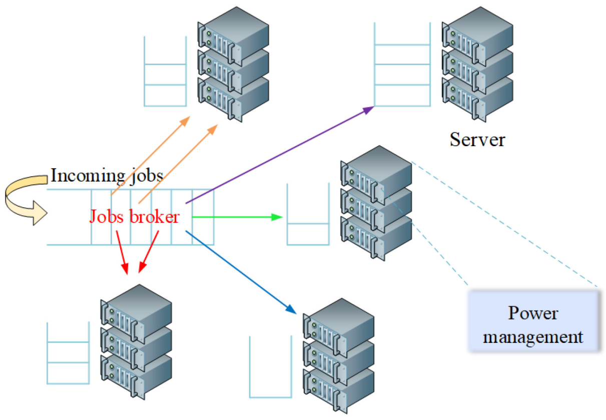


Fig. 13. Cloud computing resource allocation and server power management framework.

Table 3. Comparison of Power Modeling Methods Based on Empirical Parameterization, Function Regression, and ML

Modeling Methods	Advantages	Limitations	Suitable Scenarios
Empirical parameterization	Out of the box	Poor accuracy	Any scenarios with no power data available and no strict accuracy requirements
Function regression	Easy to build and fit	Model functions must well fit power curves for decent accuracy	Cloud servers with relatively clear power behavior patterns
Machine learning	Can learn from complex power behavior and typically high accuracy after being well trained	Requires large training dataset, long training time, and easing to over-fitting	Higher input dimensions and complex power behavior

predicting power consumption using the grammatical evolution (GE) algorithm, which is an optimization algorithm that simulates the generation and evolution of DNA sequences.

4.5 Comparison of Power Modeling Methods

In Table 3, we summarize the usability and applicability of each type of power modeling method. The truth is that there is no one-size-fits-all solution when it comes to cloud server power modeling, and acquiring a good power model often means putting more effort in data collection and model design. Using empirical parameterized models and fitting models in forms of regression functions are generally easy in practice, but the drawback is also obvious—the resulting model could be inaccurate if the target server shows a much more complicated power pattern than the model can approximate. Using more complex models with the help of ML can provide a decent solution in case there is a sufficient amount of data for training and reasonable knowledge of parameter tuning.

5 CHALLENGES AND FUTURE RESEARCH DIRECTIONS

Currently, a lot of data centers are still in the evolving stage from traditional architecture to the cloud architecture. The way of hybrid deployment allows the use of IT equipment to cover both traditional stand-alone patterns and cloud IT resource sharing patterns. Moreover, some leading cloud service giants have already introduced new virtualization technology to further improve the utilization of resources and reduce waste of energy in their server farms. In this trend, cloud server power management has become a critical aspect that service providers must pay attention to. Meanwhile, challenges also arise as the orchestration of resources and the applications on the cloud make the power behavior of servers more complicated while flexible, accurate, scalable, and fine-grained power monitoring is still in urgent demand.

5.1 Refinement of Models for Power-Intensive Components

Most of the existing research only considers the basic server components (the CPU, memory, disk, etc.) in their power models. However, peripheral devices (e.g., the GPU) connected via the PCI slots have become increasingly important in both functioning and power consumption. In view of this, it is necessary to develop fine-grained power models that cover a broader range of components in modern cloud servers.

General-purpose GPU (GPGPU) computing has emerged as a significant computing architecture thanks to the rapid advance of AI and HPC. However, it is also undoubtful that the GPUs are components that can swallow up energy, as their rated power can reach almost 1,000 in wattage. GPU power models have received increasing attention. For instance, a major direction is to investigate the power behavior of servers running both CPU- and GPU-intensive applications [88–91]. In addition, applying DVFS technology to reduce the power consumption of the GPU is also a research hotspot [92–95]. The GPU is designed to perform highly parallel computation like processing high-dimensional matrices, but there is still a significant difference in its power characteristics when running different workloads. For example, training fully connected neural nets and convolutional neural nets on a same size of data can result in completely different GPU utilization and power consumption. As well, power-saving methods for specific GPU models are also worth investigating [96–102].

The surge of GPUs also brings about challenges in monitoring the power of virtualized server instances like VMs. CSPs like Amazon have been offering powerful instances equipped with a GPU, which turns out to be extremely expensive in the hourly price. Is computation on a GPU in a virtualized environment as efficient as that on a bare-metal one? Is the price really proportional to the actual power cost? The answers require further exploration into the power behavior of GPUs. In addition, some HPC systems use shared component structures, which is another challenge for power management as it relates to how to manage power at the intra-node and inter-node levels in such a system with nodes sharing geographically distributed components such as the CPU, GPU, and FPGA.

5.2 Power Consumption Modeling for Containers

Containerization is believed to be the very path to realizing lightweight cloud services, or the so-called micro-services. Most of the global CSPs have begun to provide the container platform open for developers and end users. Despite the debate as to whether containers will take the place of traditional VMs sooner or later, this new technology certainly brings challenges to the way we used to monitor cloud servers' power, and we have to consider a container as a potential form of cloud server, which resembles a process from the prospective of the OS while functioning just like a VM for end users. The difficulties in estimating the power consumption of containers are multi-fold. On the one hand, the number of container instances that can be run on a single server is much bigger than that of the VM, which implies myriads of them need to be monitored when we look at an entire data center with thousands of physical servers. This probably results in prohibitive overhead in gathering, estimating, and predicting their power. On the other hand, the life cycle of containers is often uncertain while we may see frequent re-allocation, restart, and destroy of containers under the management of automatic orchestration software. To make container power monitoring practical, a possible solution is to monitor containers in groups just like the orchestrator does in the Kubernetes platform where containers are put in pods.

5.3 Power Consumption Modeling for Unikernel

With the development of virtual technology, there are increasing doubts that VMs may not be the actual solution for achieving smooth scaling in high-density applications. But there an alternative to containers. Developed to be even more lightweight than containers, the concept of unikernel is gradually attracting more attention. Song et al. [103] demonstrated that the unikernel can show promising advantages in delay, memory usage, image size, and power consumption over VMs and containers. Although the unikernel has been widely used in some special applications like cloud service security [104], topics such as how to model the power consumption of unikernel instances and how to maximize its advantage in power saving still need further exploration.

5.4 Power Consumption Modeling Based on the ANN

We have seen more and more studies build a variety of ANN models for predicting cloud servers' workload and power consumption. Although many decent models have been built and were tested to be very accurate, there remain two aspects that we can focus on in future work.

The first aspect is about how to effectively perform training in virtualized environments (e.g., VMs and containers) where the main difficulty is the acquisition of true power data. There is no external instrument that can be used for direct power measurement in such environments. However, VM- or container-level power information could be very important to realizing energy-aware resource allocation [76] and instance scheduling in a hierarchically virtualized data center. A possible theoretical approach is to leverage unsupervised learning methods, through which the power contribution by each virtualized instance can be regarded as latent variables that follow certain distributions.

Another aspect worth exploring is how to choose the optimal ANN structures (and the corresponding optimization algorithms) when building a power model. Models that are too complex are prone to over-fitting and difficulty in generalization and, more importantly, cause significant waste of resources during training. The suitability of different ANN structures depends on scenarios that need to be carefully considered. For example, given a series of server power data with successive timestamps, it is worth considering models able to learn temporal patterns like the auto-regressive model, RNN, and LSTM [105]. Another interesting approach is hybrid ANN [106], which is the combination of ANN and traditional technologies to establish a more efficient prediction model. Since searching for the optimal model structure in power modeling required lots of expertise, we suggest using automatic structure search methods like neural net architecture search (NAS) to shorten the parameterization process.

5.5 Power Consumption Modeling for the Edge Cloud in the IoT

With the rapid development and deployment of of IoT applications, the power consumption and its distribution over the complex hierarchical architecture of the IoT cloud is drawing more concerns. Li et al. [107] conducted experiments on real-world test beds and drew a conclusion that the edge cloud (incorporating the compute and storage resources) consumes three times more energy than the sum of all IoT devices plus the wireless access points. A similar phenomenon is found in fog computing. Bonomi et al. [108] argued that moving to fog computing is very important for future cloud services and applications, especially for IoT applications with geographical distribution, latency sensitivity, and high resiliency. From the perspective of the power efficiency, Jalali et al. [109] proposed two dataflow and time-based power models in the context of shared and non-shared network devices, respectively. In the work, they conclude that the power efficiency of fog computing is higher than cloud computing under the same service request rate. To achieve power saving, Wang et al. [110] proposed to cache various network resources into fog devices in the edge layer, from which terminal devices can get quick response through public network and device-to-device communication (D2D) technology. There is a trade-off between the service quality (e.g., metrics like delay) and the reservation of power/energy consumption in fog computing. Deng et al. [111] studied how to optimize overall power consumption with constraints on the service delay under different distributions of workload. Liang et al. [112] proposed a fog-cloud hybrid wireless access network architecture based on the load distribution in fog computing. They established a power model to estimate the power required and apply constraints to ensure the service quality by adaptively offloading the computing tasks between the cloud and fog devices.

IoT services may well be consuming even a larger amount of energy than the clouds do so far, which definitely deserves more attention especially from the perspective of reining in their power consumption. The main difficulty here is that one should not solely focus on the energy consumed by the cloud servers or that by end devices. Instead, it is necessary to pay attention to the interaction between the cloud, edge, and end layers and find out the potential correlation in power consumption.

5.6 Joint Optimization of Power by Coordinating Workload on Servers and the Cooling System

Although we do not reach the scope of modeling the cooling system's power in this survey, it generally accounts for a big part in a data center's electricity bills. Studies have found that both the power of cloud servers and cooling instruments are not only prominent but often are closely related to each other. It is intuitive if we consider the fact that the more utilized the servers are, the more heat they will be producing and, consequently, the harder the cooling system needs to work. In view of this, it is of great significance to explore how to realize joint optimization of their power via adaptive control over both cloud servers and cooling devices. This requires a thorough understanding on the power behaviors of both servers and the cooling system and the heat exchange design of data centers.

6 CONCLUSION

Monitoring the power consumption of cloud servers is the very foundation of any power and energy management strategies in modern cloud data centers. Although there are several methods to acquire real-time power and a variety of models available for estimating or predicting the power consumption of servers, how to select the best fit in practice requires much expertise. In view of this, we presented a comprehensive survey that covers a broad range of techniques and studies concerning how to collect power data from cloud servers, how to select a suitable power consumption model, and how to establish a specialized power model on a set of power data. For each aspect, we summarized existing techniques, models, and modeling methods in the form of a taxonomy. By comparing existing approaches and solutions, we analyzed their advantages, limitations, and suitable scenarios to provide useful guidance for researchers and engineers. In addition, as part of the survey, we also pointed out several open challenges along with possible research directions covering the research on power estimation in new virtualized circumstances and edge/fog environments and modeling methods using cutting-edge ML methods, aiming to provide useful guidance and inspiration for research on energy consumption management and the application of energy-aware cloud computing.

REFERENCES

[1] Rajkumar Buyya, Anton Beloglazov, and Jemal Abawajy. 2010. Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges. arXiv:1006.0308.

[2] Miyuru Dayarathna, Yonggang Wen, and Rui Fan. 2017. Data center energy consumption modeling: A survey. *IEEE Communications Surveys & Tutorials* 18, 1 (2017), 732–794.

[3] Aman Kansal, Zhao Feng, Liu Jie, Nupur Kothari, and Arka A. Bhattacharya. 2010. Virtual machine power metering and provisioning. In *Proceedings of the ACM Symposium on Cloud Computing*.

[4] Jayant Baliga, Robert W. A. Ayre, Kerry Hinton, and Rodney S. Tucker. 2010. Green cloud computing: Balancing energy in processing, storage, and transport. *Proceedings of the IEEE* 99, 1 (2010), 149–167.

[5] Amir Varasteh Hajipour and Maziar Goudarzi. 2017. Server consolidation techniques in virtualized data centers: A survey. *IEEE Systems Journal* 11, 2 (2017), 772–783.

[6] Howard Cheung, Shengwei Wang, Chaoqun Zhuang, and Jiefan Gu. 2018. A simplified power consumption model of information technology (IT) equipment in data centers for energy system real-time dynamic simulation. *Applied Energy* 222 (2018), 329–342.

[7] Anton Beloglazov, Rajkumar Buyya, Young Choon Lee, and Albert Zomaya. 2011. A taxonomy and survey of energy-efficient data centers and cloud computing systems. In *Advances in Computers*. Vol. 82. Elsevier, 47–111.

[8] Zheng Li, Selome Tesfatsion, Saeed Bastani, Ahmed Ali-Eldin, Erik Elmroth, Maria Kihl, and Rajiv Ranjan. 2017. A survey on modeling energy consumption of cloud applications: Deconstruction, state of the art, and trade-off debates. *IEEE Transactions on Sustainable Computing* 2, 3 (2017), 255–274.

[9] Huawei Technologies Co. n.d. FusionServer iBMC. Retrieved July 24, 2020 from https://e.huawei.com/en/material/onLineView?materialid=8a3ba90f16654be49cd91d8b526f6b44.

[10] Weiwei Lin, Haoyu Wang, Yufeng Zhang, Deyu Qi, James Z. Wang, and Victor Chang. 2018. A cloud server energy consumption measurement system for heterogeneous cloud environments. *Information Sciences* 468 (2018), 47–62.

[11] James W. Smith and Ian Sommerville. 2011. Workload classification & software energy measurement for efficient scheduling on private cloud platforms. arXiv:1105.2584.

[12] Ata E. Husain Bohra and Vipin Chaudhary. 2010. VMeter: Power modelling for virtualized clouds. In *Proceedings of the IEEE International Symposium on Parallel & Distributed Processing*.

[13] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, Cesar A. F. De Rose, and Rajkumar Buyya. 2011. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience* 41 (2011), 23–50.

[14] Chung Hsing Hsu and Stephen W. Poole. 2011. Power signature analysis of the SPECpower_ssj2008 benchmark. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems & Software*.

[15] Christos K. Filelis-Papadopoulos, Konstantinos M. Giannoutakis, George A. Gravvanis, and Dimitrios Tzovaras. 2017. Large-scale simulation of a self-organizing self-management cloud computing framework. *Journal of Supercomputing* 3 (2017), 1–21.

[16] Shuaiwen Leon Song, Kevin Barker, and Darren Kerbyson. 2013. Unified performance and power modeling of scientific workloads. In *Proceedings of the 1st International Workshop on Energy Efficient Supercomputing*. 1–8.

[17] Bogdan Marius Tudor and Yong Meng Teo. 2013. On understanding the energy consumption of ARM-based multicore servers. In *Proceedings of the ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems*. 267–278.

[18] E. N. Mootaz Elnozahy, Michael Kistler, and Ramakrishnan Rajamony. 2002. Energy-efficient server clusters. In *Proceedings of the International Workshop on Power-Aware Computer Systems*. , 179–197.

[19] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. 2007. Power provisioning for a warehouse-sized computer. *ACM Sigarch Computer Architecture News* 35, 2 (2007), 13–23.

[20] Suzanne Rivoire, Parthasarathy Ranganathan, and Christos Kozyrakis. 2008. A comparison of high-level full-system power models. In *Proceedings of the Workshop on Power Aware Computing & Systems*.

[21] Luiz André Barroso and Urs Hölzle. 2009. The datacenter as a computer: An introduction to the design of warehouse-scale machines. *Synthesis Lectures on Computer Architecture* 4, 1 (2009), 1–108.

[22] Robert Basmadjian, Nasir Ali, Florian Niedermeier, Hermann De Meer, and Giovanni Giuliani. 2011. A methodology to predict the power consumption of servers in data centres. In *Proceedings of the ACM SIGCOMM International Conference on Energy-Efficient Computing & Networking*.

[23] Liang Luo, W. U. Wen-Jun, and Fei Zhang. 2014. Energy modeling based on cloud data center. *Journal of Software* 7 (2014), 1371–1387.

[24] Robert Basmadjian and Hermann De Meer. 2012. Evaluating and modeling power consumption of multi-core processors. In *Proceedings of the 3rd International Conference on Future Energy Systems: Where Energy, Computing, and Communication Meet*. Article 12, 10 pages.

[25] Osman Sarood, Akhil Langer, Abhishek Gupta, and Laxmikant Kale. 2014. Maximizing throughput of overprovisioned HPC data centers under a strict power budget. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage & Analysis.*

[26] Bharan Giridhar, Michael Cieslak, Deepankar Duggal, Ronald Dreslinski, Hsing Min Chen, Robert Patti, Betina Hold, Chaitali Chakrabarti, Trevor Mudge, and David Blaauw. 2013. Exploring DRAM organizations for energy-efficient and resilient exascale memories. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage, and Analysis.* 1–12.

[27] Patricia Arroba, Jose L. Risco-Martin, Marina Zapater, Jose M. Moya, Jose L. Ayala, and Katzalin Olcoz. 2014. Server power modeling for run-time energy optimization of cloud computing facilities. *Energy Procedia* 62 (2014), 401–410.

[28] Yan Zhang, Sudhanva Gurumurthi, and Mircea R. Stan. 2007. SODA: Sensitivity based optimization of disk architecture. In *Proceedings of the 44th Annual Design Automation Conference.* 865–870.

[29] Miriam Allalouf, Yuriy Arbitman, Michael Factor, Ronen I. Kat, Kalman Z. Meth, and Dalit Naor. 2009. Storage modeling for power estimation. In *Proceedings of SYSTOR 2009: the Israeli Experimental Systems Conference.* Article 3, 10 pages.

[30] Anshul Gandhi, Mor Harchol-Balter, Rajarshi Das, and Charles Lefurgy. 2009. Optimal power allocation in server farms. In *Proceedings of the 11th International Joint Conference on Measurement & Modeling of Computer Systems.*

[31] Maruti Gupta and Suresh Singh. 2003. Greening of the Internet. *ACM SIGCOMM Computer Communication Review* 33, 4 (2003), 19–26.

[32] Suresh Gupta and Maruti Singh. 2007. Dynamic ethernet link shutdown for energy conservation on Ethernet links. In *Proceedings of the 2007 IEEE International Conference on Communications.* IEEE, Los Alamitos, CA, 6156–6161.

[33] Maruti Gupta and Suresh Singh. 2007. Using low-power modes for energy conservation in Ethernet LANs. In *Proceedings of the IEEE International Conference on Computer Communications (IEEE INFOCOM'07).*

[34] Chamara Gunaratne, Ken Christensen, and Bruce Nordman. 2010. Managing energy consumption costs in desktop PCs and LAN switches with proxying, split TCP connections, and scaling of link speed. *International Journal of Network Management* 15, 5 (2010), 297–310.

[35] F. Blanquicet and K. Christensen. 2008. Managing energy use in a network with a new SNMP Power State MIB. In *Proceedings of the IEEE Conference on Local Computer Networks.*

[36] Robert Basmadjian, Hermann De Meer, Ricardo Lent, and Giovanni Giuliani. 2012. Cloud computing and its interest in saving energy: The use case of a private cloud. *Journal of Cloud Computing Advances Systems & Applications* 1, 1 (2012), 5.

[37] Yanfei Li, Wang Ying, Yin Bo, and Guan Lu. 2012. An online power metering model for cloud environment. In *Proceedings of the IEEE International Symposium on Network Computing & Applications.*

[38] Daniel Versick, Ingolf Wabmann, and Djamshid Tavangarian. 2013. Power consumption estimation of CPU and peripheral components in virtual machines. *ACM SIGAPP Applied Computing Review* 13, 3 (2013), 17–25.

[39] Ingolf Wabmann, Daniel Versick, and Djamshid Tavangarian. 2013. Energy consumption estimation of virtual machines. In *Proceedings of the ACM Symposium on Applied Computing.*

[40] Xiao Peng, Zhigang Hu, Dongbo Liu, Guofeng Yan, and Xilong Qu. 2013. Virtual machine power measuring technique with bounded error in cloud environments. *Journal of Network & Computer Applications* 36, 2 (2013), 818–828.

[41] Xiao Peng and Zhao Sai. 2013. A low-cost power measuring technique for virtual machine in cloud environments. *International Journal of Grid & Distributed Computing* 6, 3 (2013), 69–80.

[42] Wentai Wu, Weiwei Lin, and Zhiping Peng. 2016. An intelligent power consumption model for virtual machines under CPU-intensive workload in cloud environment. *Soft Computing* 21 (2016), 5755–5764.

[43] Dong Ki Kang, Gyu Beom Choi, Seong Hwan Kim, Il Sun Hwang, and Chan Hyun Youn. 2017. Workload-aware resource management for energy efficient heterogeneous docker containers. In *Proceedings of the 2010 Region 10 Conference.*

[44] Sareh Fotuhi Piraghaj, Amir Vahid Dastjerdi, Rodrigo N. Calheiros, and Rajkumar Buyya. 2016. A framework and algorithm for energy efficient container consolidation in cloud data centers. In *Proceedings of the IEEE International Conference on Data Science & Data Intensive Systems.*

[45] Phung James, Young Choon Lee, and Albert Y. Zomaya. 2019. Lightweight power monitoring framework for virtualized computing environments. *IEEE Transactions on Computers* 69, 1 (2019), 14–25.

[46] Guillaume Fieni, Romain Rouvoy, and Lionel Seinturier. 2020. SmartWatts: Self-calibrating software-defined power meter for containers. arXiv:2001.02505.

[47] Senay Semu Tadesse, Francesco Malandrino, and Carla Fabiana Chiasserini. 2017. Energy consumption measurements in Docker. In *Proceedings of the IEEE Computer Software & Applications Conference*, Vol. 2.

[48] James William Smith, Ali Khajehhosseini, Jonathan Stuart Ward, and Ian Sommerville. 2012. CloudMonitor: Profiling power usage. In *Proceedings of the IEEE International Conference on Cloud Computing.*

[49] Shinan Wang, Youhuizi Li, Weisong Shi, Lingjun Fan, and Abhishek Agrawal. 2013. Safari: Function-level power analysis using automatic instrumentation. In *Proceedings of the International Conference on Energy Aware Computing*.

[50] Ricardo Koller, Akshat Verma, and Anindya Neogi. 2010. WattApp: An application aware power meter for shared data centers. In *Proceedings of the 7th International Conference on Autonomic Computing*. 31–40.

[51] Feifei Chen, John Grundy, and Yun Yang. 2013. Experimental analysis of task-based energy consumption in cloud computing systems. In *Proceedings of the ACM/SPEC International Conference on Performance Engineering*. 295–306.

[52] Marc Gamell, Ivan Rodero, Manish Parashar, and Stephen Poole. 2013. Exploring energy and performance behaviors of data-intensive scientific workflows on systems with deep memory hierarchies. In *Proceedings of the International Conference on High Performance Computing*.

[53] Maxime Colmant, Mascha Kurpicz, Pascal Felber, Loic Huertas, Romain Rouvoy, and Anita Sobe. 2015. Process-level power estimation in VM-based systems. In *Proceedings of the 10th European Conference on Computer Systems*.

[54] Alessandro Leite, Claude Tadonki, Christine Eisenbeis, and Alba de Melo. 2014. A fine-grained approach for power consumption analysis and prediction. *Procedia Computer Science* 29 (2014), 2260–2271.

[55] David Meisner, Christopher M. Sadler, Luiz Andre Barroso, Wolf Dietrich Weber, and Thomas F. Wenisch. 2011. Power management of online data-intensive services. In *Proceedings of the International Symposium on Computer Architecture*.

[56] Meikel Poess and Raghunath Othayoth Nambiar. 2010. A power consumption analysis of decision support systems. In *Proceedings of the 1st Joint WOSP/SIPEW International Conference on Performance Engineering*.

[57] Nan Zhu, Lei Rao, Xue Liu, Jie Liu, and Haibin Guan. 2011. Taming power peaks in MapReduce clusters. *ACM SIGCOMM Computer Communication Review* 41, 4 (2011), 416–417.

[58] Nan Zhu, Lei Rao, Xue Liu, and Jie Liu. 2012. Handling more data with less cost: Taming power peaks in MapReduce clusters. In *Proceedings of the Asia-Pacific Workshop on Systems*. 1–6.

[59] Willis Lang and Jignesh M. Patel. 2010. Energy management for MapReduce clusters. *Proceedings of the VLDB Endowment* 3, 1–2 (2010), 129–139.

[60] Mohammed El Mehdi Diouri, Olivier Gluck, Laurent Lefevre, and Jean-Christophe Mignot. 2013. Energy estimation for MPI broadcasting algorithms in large scale HPC systems. In *Proceedings of the European MPI Users Group Meeting*.

[61] Marc Gamell, Ivan Rodero, Manish Parashar, Janine C. Bennett, Hemanth Kolla, Jacqueline Chen, Peer-Timo Bremer, et al. 2013. Exploring power behaviors and trade-offs of in-situ data analytics. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage, and Analysis*. 1–12.

[62] Weiwei Lin, Wentai Wu, Haoyu Wang, James Z. Wang, and Ching Hsien Hsu. 2016. Experimental and quantitative analysis of server power model for cloud data centers. *Future Generation Computer Systems* 86 (2016), 940–950.

[63] Zhou Zhou, Jemal H. Abawajy, Fangmin Li, Zhigang Hu, and Keqin Li. 2017. Fine-grained energy consumption model of servers based on task characteristics in cloud data center. *IEEE Access* PP, 99 (2017), 1.

[64] John J. Prevost, Kranthi Manoj Nagothu, Brian Kelley, and Jamshidi Mo. 2011. Prediction of cloud data center networks loads using stochastic and neural models. In *Proceedings of the International Conference on System of Systems Engineering*.

[65] Yao Lu, John Panneerselvam, Lu Liu, and Yan Wu. 2016. RVLBPNN: A workload forecasting model for smart cloud computing. *Scientific Programming* 2016 (2016), 1–9.

[66] Jitendra Kumar and Ashutosh Kumar Singh. 2018. Workload prediction in cloud using artificial neural network and adaptive differential evolution. *Future Generation Computer Systems* 81 (2018), 41–52.

[67] Luo Liang, Wenjun Wu, W. T. Tsai, Dichen Di, and Zhang Fei. 2013. Simulation of power consumption of cloud data centers. *Simulation Modelling Practice & Theory* 39 (2013), 152–171.

[68] T. Veni and S. Mary Saira Bhanu. 2016. Prediction model for virtual machine power consumption in cloud environments. *Procedia Computer Science* 87 (2016), 122–127.

[69] Humaira Abdul Salam, Franco Davoli, Alessandro Carrega, and Andreas Timm-Giel. 2018. Towards prediction of power consumption of virtual machines for varying loads. In *Proceedings of the 2018 28th International Telecommunication Networks and Applications Conference (ITNAC'18)*. IEEE, Los Alamitos, CA, 1–6.

[70] Hailong Yang, Zhao Qi, Zhongzhi Luan, and Depei Qian. 2014. iMeter: An integrated VM power model based on performance profiling. *Future Generation Computer Systems* 36, 3 (2014), 267–286.

[71] Shuai Ye, Ruoyan Zhao, and Xinru Fang. 2019. An ensemble learning method for dialect classification. In *IOP Conference Series: Materials Science and Engineering*, Vol. 569. IOP Publishing, 052064.

[72] De-Shuang Huang, Vitoantonio Bevilacqua, and Prashan Premaratne. 2016. *Intelligent Computing Theories and Application: 12th International Conference, ICIC 2016, Lanzhou, China, August 2-5, 2016, Proceedings I*. Lecture Notes in Computer Science, Vol. 9771. Springer.

[73] Timothy Harton, Cameron G. Walker, and Michael O'Sullivan. 2016. Towards power consumption modeling for servers at scale. In *Proceedings of the IEEE/ACM International Conference on Utility & Cloud Computing*.

[74] Balaji Krishnapuram, Mohak Shah, Alex Smola, Charu Aggarwal, Dou Shen, and Rajeev Rastogi. 2016. *KDD'16: The 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY.

[75] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*. 3146–3154.

[76] Ning Liu, Lin Xue, and Yanzhi Wang. 2017. Data center power management for regulation service using neural network-based power prediction. In *Proceedings of the International Symposium on Quality Electronic Design*.

[77] Yuanlong Li, Hu Han, Yonggang Wen, and Jun Zhang. 2016. Learning-based power prediction for data centre operations via deep neural networks. In *Proceedings of the International Workshop on Energy Efficient Data Centres*.

[78] Weiwei Lin, Guangxin Wu, Xinyang Wang, and Keqin Li. 2019. An artificial neural network approach to power consumption model construction for servers in cloud data centers. *IEEE Transactions on Sustainable Computing*. Early Access.

[79] Nilabja Roy, Abhishek Dubey, and Aniruddha S. Gokhale. 2011. Efficient autoscaling in the cloud using predictive models for workload forecasting. In *Proceedings of the IEEE International Conference on Cloud Computing*.

[80] Jitendra Kumar, Rimsha Goomer, and Ashutosh Kumar Singh. 2018. Long short term memory recurrent neural network (LSTM-RNN) based workload forecasting model for cloud datacenters. *Procedia Computer Science* 125 (2018), 676–682.

[81] Z. Chen, Y. Zhu, Y. Di, and S. Feng. 2015. Self-adaptive prediction of cloud resource demands using ensemble model and subtractive-fuzzy clustering based fuzzy neural network. *Computational Intelligence and Neuroscience* 2015, 10a (2015), 17.

[82] Gaurav Dhiman, Kresimir Mihic, and Tajana Rosing. 2010. A system for online power prediction in virtualized environments using Gaussian mixture models. In *Proceedings of the Design Automation Conference*.

[83] Zhu Hao, Huadong Dai, Shazhou Yang, Yuejin Yan, and Bin Lin. 2017. Estimating power consumption of servers using Gaussian mixture model. In *Proceedings of the International Symposium on Computing & Networking*.

[84] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. arXiv:1312.5602.

[85] Fahimeh Farahnakian, Pasi Liljeberg, and Juha Plosila. 2014. Energy-efficient virtual machines consolidation in cloud data centers using reinforcement learning. In *Proceedings of the Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*.

[86] Ning Liu, Li Zhe, Zhiyuan Xu, Jielong Xu, and Yanzhi Wang. 2017. A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning. In *Proceedings of the IEEE 37th International Conference on Distributed Computing Systems (ICDCS'17)*. 372–382.

[87] Juan C. Salinas Hilburg, Marina Zapater, Jose L. Risco Martin, Jose M. Moya, and Jose L. Ayala. 2015. Using grammatical evolution techniques to model the dynamic power consumption of enterprise servers. In *Proceedings of the 9th International Conference on Complex, Intelligent, and Software Intensive Systems*.

[88] Sparsh Mittal and Jeffrey S. Vetter. 2014. A survey of methods for analyzing and improving GPU energy efficiency. *ACM Computing Surveys* 47, 2 (2014), 1–23.

[89] Kenneth O'Neal and Philip Brisk. 2018. Predictive modeling for CPU, GPU, and FPGA performance and power consumption: A survey. In *Proceedings of the 2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI'18)*. 763–768.

[90] Mark Gebhart, Daniel R. Johnson, David Tarjan, Stephen W. Keckler, William J. Dally, Erik Lindholm, and Kevin Skadron. 2011. Energy-efficient mechanisms for managing thread context in throughput processors. In *Proceedings of the 2011 38th Annual International Symposium on Computer Architecture (ISCA'11)*. IEEE, Los Alamitos, CA, 235–246.

[91] Daniel Nikolai Peroni. 2019. *Approximate Computing for GPGPU Acceleration*. Ph.D. Dissertation. University of California, San Diego.

[92] Wenjie Liu, Zhihui Du, Xiao Yu, David A. Bader, and Xu Chen. 2011. A waterfall model to achieve energy efficient tasks mapping for large scale GPU clusters. In *Proceedings of the IEEE International Symposium on Parallel & Distributed Processing Workshops & Ph.D. Forum*.

[93] Minsoo Rhu, Michael Sullivan, Jingwen Leng, and Mattan Erez. 2013. A locality-aware memory hierarchy for energy-efficient GPU architectures. In *Proceedings of the IEEE/ACM International Symposium on Microarchitecture*.

[94] Gong Fan, Ju Lei, Deshan Zhang, Mengying Zhao, and Zhiping Jia. 2017. Cooperative DVFS for energy-efficient HEVC decoding on embedded CPU-GPU architecture. In *Proceedings of the Design Automation Conference*.

[95] João Guerreiro, Aleksandar Ilic, Nuno Roma, and Pedro Tomás. 2019. DVFS-aware application classification to improve GPGPUs energy efficiency. *Parallel Computing* 83 (2019), 93–117.

[96] Andrew Kerr, Gregory F. Diamos, and Sudhakar Yalamanchili. 2010. Modeling GPU-CPU workloads and systems. In *Proceedings of the Workshop on General Purpose Processing on Graphics Processing Units*.

[97] John A. Stratton, Nasser Anssari, Christopher Rodrigues, I. Jui Sung, and W. M. Hwu. 2012. Optimization and architecture effects on GPU computing workload performance. In *Proceedings of the 2012 Conference on Innovative Parallel Computing (InPar'12)*.

[98] Mengchi Zhang, Roland Green, and Timothy G. Rogers. 2018. Characterizing the runtime effects of object-oriented workloads on GPUs. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems & Software*.

[99] Wang Yue, Soumyaroop Roy, and Nagarajan Ranganathan. 2012. Run-time power-gating in caches of GPUs for leakage energy savings. In *Proceedings of the Design, Automation, and Test in Europe Conference & Exhibition*.

[100] Junke Li, Guo Bing, Shen Yan, Deguang Li, Jihe Wang, Yanhui Huang, and Li Qiang. 2015. GPU-memory coordinated energy saving approach based on extreme learning machine. In *Proceedings of the IEEE International Symposium on High Performance Computing & Communications*.

[101] Bing Li, Mengjie Mao, Xiaoxiao Liu, Tao Liu, Zihao Liu, Wujie Wen, Yiran Chen, et al. 2019. Thread batching for high-performance energy-efficient GPU memory design. arXiv:1906.05922.

[102] Mohsen Imani and Tajana S. Rosing. 2019. *Approximate CPU and GPU Design Using Emerging Memory Technologies*. Springer.

[103] Wu Song, Mei Chao, Jin Hai, and Duoqiang Wang. 2018. Android Unikernel: Gearing mobile code offloading towards edge computing. *Future Generation Computer Systems* 86 (2018), 694–703.

[104] Zeyi Tao, Qi Xia, Zijiang Hao, Cheng Li, Lele Ma, Shanhe Yi, and Qun Li. 2019. A survey of virtual machine management in edge computing. *Proceedings of the IEEE* 107, 8 (2019), 1482–1499.

[105] Xian Yu, Guangxing Zhang, Zhenyu Li, Wei Liangs, and Gaogang Xie. 2019. Toward generalized neural model for VMs power consumption estimation in data centers. In *Proceedings of the 2019 IEEE International Conference on Communications (ICC'19)*.

[106] Muhammad Qamar Raza and Abbas Khosravi. 2015. A review on artificial intelligence based load demand forecasting techniques for smart grid and buildings. *Renewable & Sustainable Energy Reviews* 50 (2015), 1352–1372.

[107] Yunbo Li, Anne-Cecile Orgerie, Ivan Rodero, Betsegaw Lemma Amersho, Manish Parashar, and Jean Marc Menaud. 2017. End-to-end energy models for edge cloud-based IoT platforms: Application to data stream analysis in IoT. *Future Generation Computer Systems* 87 (2017), 667–678.

[108] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. 2012. Fog computing and its role in the Internet of Things. In *Proceedings of the 1st edition of the MCC Workshop on Mobile Cloud Computing*.

[109] Fatemeh Jalali, Kerry Hinton, Robert S. Ayre, Tansu Alpcan, and Rodney S. Tucker. 2016. Fog computing may help to save energy in cloud computing. *IEEE Journal on Selected Areas in Communications* 34, 5 (2016), 1728–1739.

[110] Siming Wang, Xumin Huang, Liu Yi, and Yu Rong. 2016. CachinMobile: An energy-efficient users caching scheme for fog computing. In *Proceedings of the IEEE/CIC International Conference on Communications in China*.

[111] Ruilong Deng, Rongxing Lu, Chengzhe Lai, Tom Hao Luan, and Liang Hao. 2017. Optimal workload allocation in fog-cloud computing towards balanced delay and power consumption. *IEEE Internet of Things Journal* 3, 6 (2017), 1171–1181.

[112] Kai Liang, Liqiang Zhao, Xiaohui Zhao, Yong Wang, and Shumao Ou. 2016. Joint resource allocation and coordinated computation offloading for fog radio access networks. *China Communications* 13, 2 (2016), 131–139.

Received October 2019; revised June 2020; accepted June 2020