Dynamic Power Capping for Latency-Sensitive Cloud Applications: A Prediction-Based Power Efficiency Approach

Huikang Huang, Weiwei Lin, *Senior Member, IEEE*, Xiaoxuan Luo, James Z. Wang, *Senior Member, IEEE*, and Keqin Li, *Fellow, IEEE*

***Abstract*—Data centers often deploy servers that exceed the capacity of the power infrastructure to increase power utilization, i.e., power over-subscription. To address potential power overloads, power capping mechanisms are implemented for protection. However, the power efficiency disparities among heterogeneous servers, along with the adjustment intervals required for power capping at the cluster level, pose challenges to capping decisions, especially for latency-sensitive applications with higher quality of service requirements. To tackle this challenge, we propose a prediction-based, power efficiency-aware dynamic power capping framework (PPE-DPC), comprisingtwo stages. First, we design a lightweight online prediction to capture requests generated within the power adjustment time intervals. Then, leveraging prior knowledge of the power efficiency of heterogeneous servers, we design a greedy strategy to fine-tune the capping power for better capping decisions. Extensive simulations and conducted in Testbed using Alibaba traces demonstrate that PPE-DPC outperforms existing solutions, optimizing request latency, power utilization, and mitigating the negative impact of power adjustment intervals. Finally, we also explore the effect of prediction error on PPE-DPC and find that only 3x the true prediction error is weaker than the existing optimal capping algorithm.**

***Index Terms*—Cloud computing, cluster, dynamic power capping, latency-sensitive application, service level agreement.**

Received 28 November 2024; revised 10 December 2025; accepted 17 December 2025. Date of publication 22 December 2025; date of current version 13 July 2026. This work was supported in part by Guangdong Provincial Natural Science Foundation Project under Grant 2025A1515010113, in part by Shandong Provincial Natural Science Foundation Project under Grant ZR2024LZH012, in part by Guangxi Key Research and Development Project under Grant 2024AB02018, and in part by the Major Key Project of PCL, China under Grant PCL2025A08 and Grant PCL2025A11. Recommended for acceptance by J. Castrillon. *(Corresponding author: Weiwei Lin.)*

Huikang Huang and Xiaoxuan Luo are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China (e-mail: huikanghuang0321@gmail.com; 202210188550@mail.scut.edu.cn).

Weiwei Lin is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China, and also with Pengcheng Laboratory, Shenzhen 518066, China (e-mail: linww@scut.edu.cn).

James Z. Wang is with the School of Computing, Clemson University, SC 29634 USA (e-mail: jzwang@clemson.edu).

Keqin Li is with the Department of Computer Science, State University of New York, New Paltz, Clemson, NY 12561 USA (e-mail: lik@newpaltz.edu).

Digital Object Identifier 10.1109/TC.2025.3647020

I. Introduction

CLOUD service providers, such as Google [1], Facebook [2], and Microsoft [3], support the regular operation of cloud services by deploying a large number of servers in data centers. With business expansion, a conservative power strategy may not be able to meet the demand for further server deployment. However, upgrading the data center power infrastructure requires a longer time and capital expenditure (CAPEX) [4]. CAPEX estimated at $10–20 per watt [5], is proportional to the IT power deployed, as each watt requires supporting equipment. How to fully utilize the existing power infrastructure to deploy more servers becomes critical to reducing the total cost of ownership (TCO).

In fact, the power systems of modern data centers often involve over-estimation in the design of power capacity [6]. This is inevitable, as it is impossible to accurately predict the actual power requirements of the data center being constructed. This over-provision of power results in two distinct headroom spaces, denoted as H_1 and H_2, as illustrated in Fig. 1. H_1 represents the headroom resulting from a conservative estimate of the power budget, which is based on the nameplate power of the servers. H_2 arises from the difference between the peak power and the dynamic power observed in data center daily operations.

It is worth noting that there are hard constraints for the design of the power capacity in different power hierarchies, i.e., Circuit Breaker (CB), which protects the same hierarchy from interfering with each other [7]. If the current power hierarchy is not isolated using CBs, and the upper-level CBs are tripped, all equipment on the current hierarchy will go down, causing incalculable damage. In addition, the previous hierarchy of servers (i.e., rack) tends to have higher power volatility due to the peak power stacking, as shown in Fig. 2. Therefore, in modern data centers where power over-subscription is common, protecting the rack hierarchy from the risk of CB tripping becomes critical to improving power capacity utilization.

Power Capping is the most commonly used technique for directly limiting server power in data centers, and there are many related studies. For example, in balancing the power consumption of servers and optimizing the service level agreement (SLA) [8], [9], [10], or in the power security

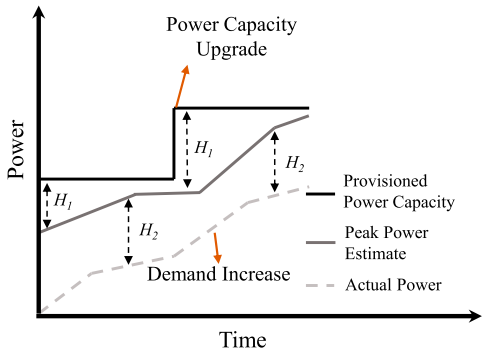


Fig. 1. Power over-provision in data centers.

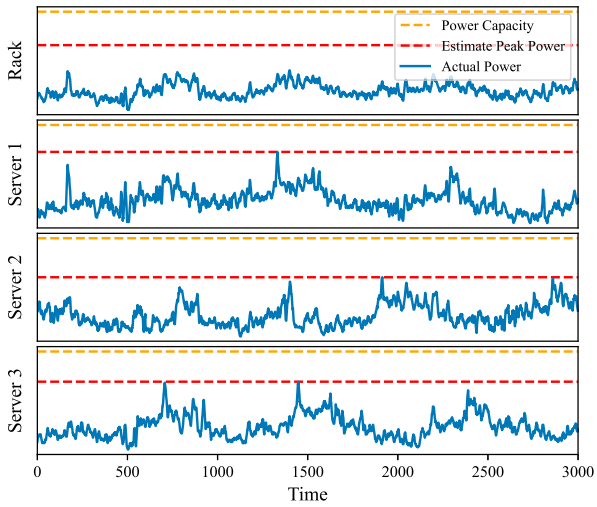


Fig. 2. Power over-estimation at the rack.

management at different hierarchies of clusters [11], [12], [13], etc. However, there are few in-depth discussions on the clustered power capping scenario with heterogeneous servers. If it is just a single heterogeneous server, server power can be quickly (even 1ms [14]) adjusted based on requests and the power efficiency of the server by employing power capping to balance energy and SLA. But in cluster/rack scenarios with multiple servers, the following key challenges are posed:

- In the case of cluster power budget constraints, how to set the capping power for heterogeneous servers with different power efficiencies to optimize total SLA?
- Compared to the single server, the joint capping decision of multiple servers is limited by the time granularity of the information collected by the cluster. For latency-sensitive applications, the negative impact of time intervals needs to be considered.

To overcome these challenges, we focus on small clusters, i.e., racks. We construct the power capping problem for latency-sensitive applications, which is essentially a stochastic optimization problem. In this paper, we propose a Prediction-based Power Efficiency-aware Dynamic Power Capping framework (PPE-DPC), a two-stage dynamic power capping to solve this problem. The first stage simply and quickly predicts requests online and converts them to the target capping threshold (thrld) for maintenance. Thrld is equal to server utilization. The second stage integrates the capping thrld values and converts them to capping power. Then, fine-tuning is based on server power efficiency and requests. PPE-DPC can effectively solve the rack-level power capping problem and reduce the negative impact of the capping adjustment interval. Overall, the contributions of this paper can be summarized as follows:

1) Leveraging the power efficiency model of heterogeneous servers and the FGD performance degradation model, a stochastic optimization problem is formulated to optimize SLA for latency-sensitive applications under power capping.
2) We propose PPE-DPC, a capping framework based on a lightweight request prediction model, enabling future information coverage within the power capping adjustment interval to enhance capping-decision performance.
3) We propose a power efficiency-aware fine-tuning algorithm that balances current and future states to optimize power capping adjustments.
4) Extensive simulations and evaluations on the Testbed using Alibaba traces and systematic comparisons with existing dynamic power capping algorithms. Experimental results show that PPE-DPC effectively reduces request SLA while improving power capacity utilization.

The subsequent sections of this paper are organized as follows. Section II presents related work. Section III gives the problem model definition and problem modeling approach. We focus on the PPE-DPC in Section IV. Section V validates the effectiveness of our approach through extensive experiments. Section VI summarizes our work.

II. Related Work

This paper focuses on improving power capacity utilization through power capping techniques while maintaining Quality of Service (QoS). Three primary approaches effectively enable power over-subscription: power adjustment, workload scheduling, and energy storage supplementation. Workload scheduling [2], [15] avoids peak stacking by routing workloads to other nodes or delaying their execution. Energy storage supplementation [16], [17] utilizes additional power capacity to mitigate peak power demands. However, the investigation of these two approaches is beyond the scope of this paper, which focuses on research related to power adjustment. Power capping is a widely used method for server power control in data centers, implemented through technologies such as DVFS [18], RAPL [19], and Thunderbolt [1]. It reduces power waste caused by redundant allocation, enables higher server deployment density, and lowers overall power consumption costs in data centers [20], [21]. This study focuses on power capping for latency-sensitive applications, excluding considerations related to batch applications, as discussed in [22], [23].

Power capping is applied to cross-layer power resource optimization in data centers. The Dynamo power management system [12], deployed by Facebook in a real-generation environment, interacts with power information between agents, Leaf

controllers, and Upper-Level controllers. It manages the power budget allocation of the power system. The Leaf controller uses the Three band algorithm to determine whether to turn on and off power capping, and penalizes servers that consume more power based on a high-bucket-first algorithm. IBM's CapMaestro [24], [25] power management system introduces a global, priority-aware power budget allocation strategy. Recognizing that power supply units (PSUs) within a server are often unevenly loaded, it employs a proportional-integral feedback controller to ensure each PSU adheres to its assigned power budget. When a surplus power budget is available, a stranded power optimization (SPO) mechanism is utilized to reallocate the excess, enhancing overall efficiency. The Ref. [26] considers different services and heterogeneous servers, and constructs three types of server sets with different priorities by using a performance-per-watt model (PPW), which employs a greedy policy to guarantee the power budget for servers with higher PPW. At a higher level of the power system, Google developed the Medium Voltage Power Plane (MVPP) [13] to enhance power over-subscription. Unlike the relatively small power-sharing domains at the rack and room levels, MVPP enables power sharing at the medium voltage distribution layer. This approach effectively reduces the frequency of security power capping triggers, thereby improving QoS.

In the area of fine-grained power capping research, Wu et al. [27] introduced Precise Power Capping (PPCapping), which employs Fine-Grained Differential (FGD) methods to accurately evaluate server performance degradation under power constraints. This approach enables precise power budget allocation, ensuring the SLAs of applications across all servers are met. DDPC [28] is a data-driven power controller designed for latency-sensitive web services. It uses system identification techniques to model the adaptive performance of web services. Based on performance data monitored by the cluster, the system integrates the performance model with a gain-scheduling proportional-integral controller. This ensures power allocation remains within the specified budget thrld. SmartOClock [29] is a workload-aware server overclocking management platform. It relies on power prediction to distribute power budgets proportionally among servers.

In scenarios where workloads have priority. Microsoft introduced criticality- and utilization-aware VM prediction methods to guide power over-subscription in VM scheduling within public cloud environments [30]. When the power budget is exceeded, RAPL is employed to prioritize power reduction for non-critical VMs. Conversely, power constraints are relaxed for these VMs when the power budget becomes sufficient. Additionally, Microsoft proposed the Flex power management system [3] for "zero reserved power" data centers. Leveraging a highly reliable and available telemetry system, Flex-online enables highly responsive capping of software-redundant workloads while minimizing the impact on latency-sensitive workloads. For Large Language Models (LLMs), which have gained popularity in data centers, Patel et al. [31] found that the average and peak power of LLM inference clusters is relatively low. Based on this, they proposed the POLCA framework for power over-subscription in LLM inference clouds. This framework employs a priority-based power recycling strategy, setting upper power thrlds for workloads of varying priority by analyzing historical power usage.

While many studies have focused on power capping for clusters, there is a lack of discussion on scenarios where servers with the same prioritized applications are deployed, as well as scenarios involving heterogeneous servers with varying efficiencies. These are typical real-world situations, and they are the primary focus of this paper.

III. Problem Formulation

A. System Model

In this paper, we address the small cluster scenario, i.e., rack, where the cloud service provider overplaces servers and deploys applications (e.g., Web search, E-commerce, etc.) in the rack based on the power budget. Generally, high-performance servers deploy more service copies to improve the server's computational resource utilization. Cloud services exhibit a distinct diurnal pattern [2], leading to power capping in racks with over-deployed servers during peak periods. To address this, operators establish power capping mechanisms paired with responsive and reliable communication pipelines. When a power violation is detected in a rack, the pipeline collects server data (e.g., power consumption, workload, CPU utilization) and calculates per-server power budgets based on the capping algorithm, ensuring strict enforcement. Servers enforcing power capping experience a certain degree of performance degradation. To better understand the problem studied in this paper, we show the system model in Fig. 3.

It can be noticed from the left side of Fig. 3, that in order to improve power resource utilization, a static rack capping power thrld (typically 99% of the CB thrld) is set at the rack level. Once the thrld is exceeded, the safety power capping mechanism is triggered. For example, at t_1, when power capping is activated, the dynamic power capping mechanism calculates the capping power based on the information collected by the monitor system, then sends and enforces it on downstream servers. The period from t_1 to t_2 represents the duration of the capping. The time interval granularity for adjusting server power capping between t_1 and t_2 depends on the system's data collection granularity (Facebook's Dynamo collects the information of the cluster at 9-second intervals [12], and IBM's CapMaestro [25] is 8-second). Power violations occur when server nodes are allocated an insufficient budget, causing some requests to be delayed and processed the next time (typically following an FCFS order), as shown in the yellow area on the right side of Fig. 3, which impacts SLAs of requests, particularly for latency-sensitive applications.

B. Problem Definition

To provide a clearer explanation of the system model, we present the mathematical equations, with the accompanying Table I defining the frequently used symbols.

The system model is mathematically defined in two main parts: the power model and the performance degradation model.

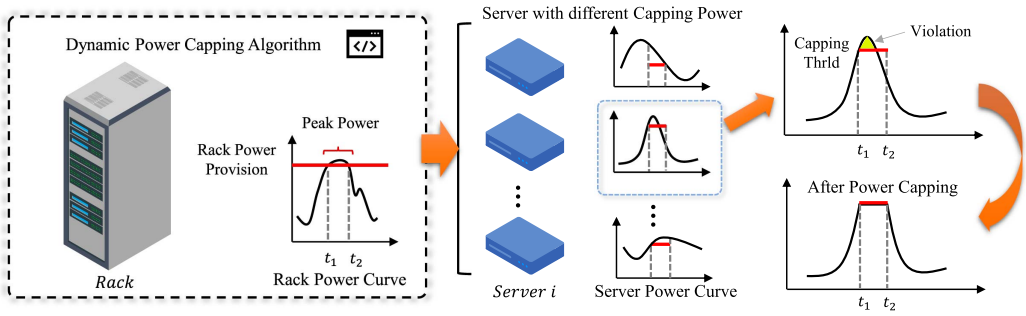


Fig. 3. Dynamic power capping in racks and its impact on server workload performance degradation.

TABLE I
LIST OF SYMBOLS

Name	Description
$thrld$	Server capping threshold ($0 \sim 1$)
PT_S	Server capping power
PT_R	Rack-level capping power
N	Number of servers in the rack
K	Time interval of dynamic power capping adjustments
M	Downsampling parameter
u	CPU utilization ($0 \sim 100\%$)
$proc(u)$	Server processing speed at utilization u (MIPS)
$power(u)$	Server power at utilization u
$peff(u)$	Server power efficiency at utilization u
w	Incoming workload ($0 \sim 1$)
W	Current cumulative workload
W^r	Remaining workload
Δw	Unit workload (100 MI)
Δt	Unit time interval
Δw^{endT}	Timestamp when the Δw is completed
Δw^{recT}	Timestamp when the Δw is received
$RT(\Delta w)$	Response time for the Δw
$\Delta power$	Unit power
$\Delta peff$	Power efficiency changing by $\Delta power$ adjustment
$IncSrv$	Set of servers that increase $thrld$ based on predictions
$DecSrv$	Set of servers that decrease $thrld$ based on predictions

To model the problem more accurately, we adopt the FGD performance degradation model [27] that integrates Cloudsim workload modeling [32], and the widely used power efficiency model from SPEC[1]. Additionally, we formulate this optimization problem by incorporating the delay optimization objective that is the focus of this paper.

1) FGD Performance Degradation Model: The FGD model is essentially a queuing model based on FCFS, where the total workload remains constant, and the latency of requests is calculated differentially based on the usage of computing resources [27]. Specifically, it assumes that the total workload, denoted as Ω, to be executed between time 0 and T is

$$\Omega = \int_0^T w(t)dt. \tag{1}$$

There are multiple timestamps t between 0 and T, with requests arriving at each timestamp denoted as $w(t)$ with receiving time $w^{recT}(t)$. The unit request is expressed as Δw, i.e., w is composed of several Δw. When power capping is triggered and the processing capacity of the server can't handle incoming requests, the unprocessed requests accumulate, leading to performance degradation. Let W_t denote the current cumulative requests at t, then the number of requests need to be processed at t is

$$W_t = W_{t-1}^r + w(t), \tag{2}$$

where W_{t-1}^r denotes the number of remaining requests from the previous time. The processing capacity of the server at time t varies depending on whether capping is triggered

$$thrld_t = \begin{cases} min(W_t, 1), & \text{No Capping} \\ thrld_{target}, & \text{Power Capping} \end{cases}. \tag{3}$$

If the server is not capped, then the maximum processing capacity of the server is 1 (i.e., the processing capacity of the server is normalized to 1). Otherwise, the processing capacity is equal to the $thrld_{target}$, which is set after the power capping algorithm is triggered. Then, the number of remaining requests W_t^r at t can be expressed as

$$W_t^r = max(W_t - thrld_t, 0). \tag{4}$$

This means that when $thrld_t$ equal to W_t, there is no request remaining at t, i.e., $W_t^r = 0$. And the completion timestamp for Δw in W_t is t. Otherwise, the completion of the remaining requests W_t^r will continue to be delayed until it is completed.

2) Power and Power Efficiency Model: To better capture the relationship between server power consumption, CPU utilization, and processing capacity, we use the server power efficiency model provided by SPEC. Specifically, the processing capacity is represented by processing speed, typically measured in MIPS [32]. Then, the server power efficiency model can be expressed as

$$peff(u) = \frac{proc(u)}{power(u)}, \tag{5}$$

[1]SPECssj2008: https://www.spec.org/power_ssj2008/

where u denotes CPU utilization, while $power(u)$ and $proc(u)$ represent power and processing speed at utilization u. Both of them at different u can be obtained through linear interpolation using the data provided by SPEC, which includes measurements at 0, 10%, 20%, ..., 90%, and 100%. In this case, the optimal power efficiency, denoted as $peak\_peff$, occurs at the load level opt_u, such that $peak\_peff = peff(opt\_u)$.

Due to the existence of the power capping mechanism, the FGD and power model must satisfy the following relationship

$$PT_S = power\,(thrld)\,, \tag{6}$$

$$PT_S \geq power(u), \tag{7}$$

$$thrld \geq u. \tag{8}$$

That is, neither the u nor the $power(u)$ of the server exceeds the capping $thrld$ with capping power PT_S.

3) Optimization Problem Definition: For the optimization objective, we use the response time RT as the metric. The response time per unit request Δw is calculated as shown below

$$RT(\Delta w) = \Delta w^{endT} - \Delta w^{recT}, \tag{9}$$

where Δw^{recT} represents the time when the server receives the request, and Δw^{endT} is the time when the server completes it. Since the optimization scenario targets delay-sensitive applications, SLA is typically evaluated using tail latency. So, we use p95 and p99 tail latency as the optimization objectives

$$Latency_{tail(x)} = p_x(RT(\Delta w)), \Delta w \in \Omega, \tag{10}$$

$$Latency_{avg} = avg(RT(\Delta w)), \Delta w \in \Omega, \tag{11}$$

where x is 95% or 99%. p_x represents the x percentile RT of all Ω. In addition, the average latency is included as an auxiliary metric to analyze the results. Then, in the scenario under power constraints at the rack level, it needs to obey

$$PT_R >= \sum_{i}^{N} PT_{S_i}(t), \forall t \in [0, T]. \tag{12}$$

It indicates that the sum of the capping power of all servers at any time $\sum_i^N PT_{S_i}(t)$ cannot exceed the capping power of the rack PT_R. Then, the model of the optimization problem studied in this paper can be expressed as

$$min(Latency_{tail(x)})$$
$$s.t.\ (7), (8)\ and\ (12).$$

4) Problem Analysis: After triggering power capping, due to the inherent time delay in collecting state information in clusters with many servers [12], [25], the capping power adjustment time interval $K * \Delta t$ must be considered. This is because for latency-sensitive applications, shorter adjustment intervals may also have a negative impact. Obviously, the problem is essentially an uncertain sequential decision-making problem stemming from unknown future requests. Decision making needs to consider not only the changes in the system during the adjustment time interval, but also the interactions of the system $state$ between each adjustment $thrld$, as shown in Fig. 4. Theoretically, this problem can be solved using deep reinforcement learning methods. However, this approach would need to consider safety constraints and the practical challenge of limited training samples.

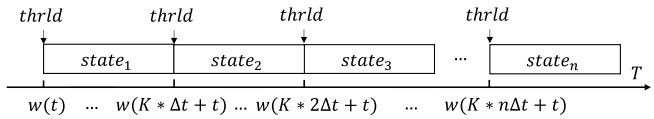


Fig. 4. Sequential decision-making problem for dynamic power capping.

IV. Prediction-Based Power Efficiency-Aware Dynamic Power Capping

To effectively address cluster power capping and optimize request SLAs, this paper introduces PPE-DPC, a prediction-based and power efficiency-aware dynamic power capping framework. PPE-DPC is a two-stage optimization scheme comprising Prediction-based Capping Power Presetting and Power Efficiency-aware Capping Power Tuning, as illustrated in Fig. 5. As shown on the left side of Fig. 5, upon triggering power capping, server power and workload information will be collected immediately, and PPE-DPC calculates and delivers the capping power for each server. The local manager then enforces the capping power using RAPL. The following subsections detail the two steps of PPE-DPC.

A. Prediction-Based Capping Power Presetting

To optimize the SLA of future requests during the capping power adjustment interval, a natural approach is to employ prediction to obtain the request information within the adjustment interval, so that it encompasses the state-space information of the adjustment interval $K * \Delta t$ for decision-making. Actually, we don't need to consider the exact value of each timestamp, we just need to know the average number of requests during the adjustment interval $K * \Delta t$. Therefore, we propose a lightweight and effective online prediction method incorporating the $downSampling()$ and $loadMovingAvg()$ methods, as illustrated in the upper right of Fig. 5.

downSampling(). Each server must store $M * K$ historical data points (representing the number of requests w) and continuously update the downsampled data during operation. Such as

$$downSampling \begin{pmatrix} w_k, & \ldots, & w_1 \\ w_{2\cdot k}, & \ldots, & w_{k+1} \\ w_{M\cdot k}, & \ldots, & w_{(M-1)\cdot k+1} \end{pmatrix}. \tag{13}$$

Then $dw_i = Avg(w_{i*k}, w_{i*k-1} \ldots w_{(i-1)*k+2}, w_{(i-1)*k+1})$ represents the downsampled value. Once requests for the next time are captured, the oldest historical data points are removed to maintain the most recent downsampling data (e.g., remove the oldest sample point w_{M*k}).

loadMovingAvg(). After obtaining the updated downsampling data dw_M,dw_{M-1},...,dw_1, the predicted values of $\widehat{dw_{pred}}$ at future times are updated. Specifically, we use a sliding window weighting-based approach to update and

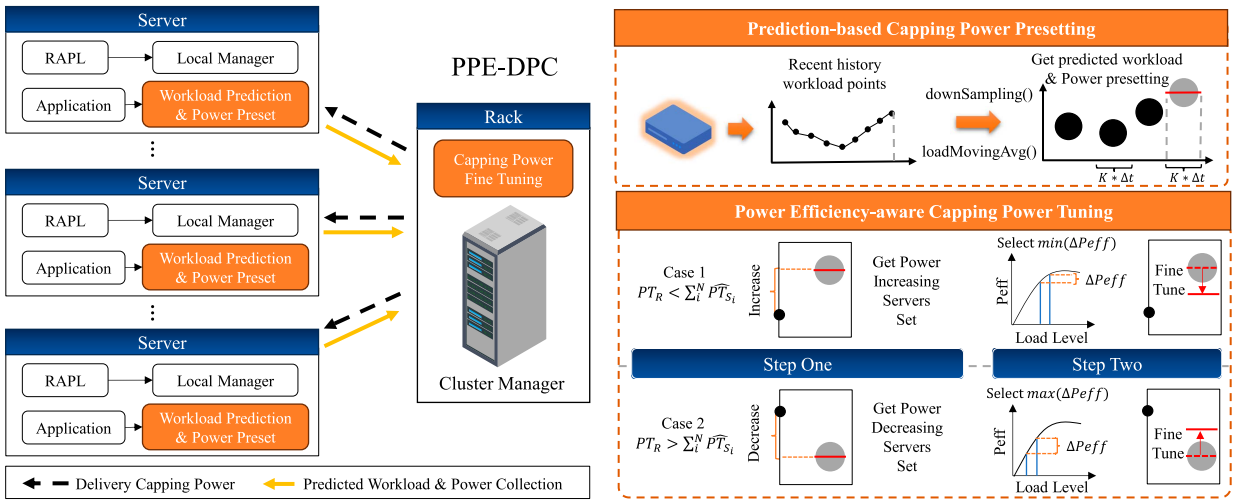


Fig. 5. Rack-level power capping framework PPE-DPC.

maintain $\widehat{dw_{pred}}$. The weights ω_t and predicted values $\widehat{dw_{pred}}$ are calculated as follows

$$\omega_t = \frac{e^{-t}\,(1-e^{t})}{1-e^{-M}}, t = 0, 1, 2 \ldots M-1, \tag{14}$$

$$\widehat{dw_{pred}} = \sum_{i=0}^{M-1} (dw_{i+1} * \omega_i). \tag{15}$$

That is, the newer the data point, the higher the weight it has, which reflects recent request trends. Then, all servers can maintain this prediction value $\widehat{dw_{pred}}$ online, to get a list of predicted values $\widehat{thrldList} = [\widehat{dw_{pred_1}}, \widehat{dw_{pred_2}} \ldots \widehat{dw_{pred_N}}]$. Now that, we get the presetting capping power of all servers $\widehat{PT_S List} = [\widehat{PT_{S_1}} \ldots \widehat{PT_{S_N}}]$, based on the predicted capping values $\widehat{thrldList}$ and power model. This means that the preset capping power can handle requests for the next time interval.

B. Power Efficiency-Aware Capping Power Tuning

The power capping fine-tuning procedure of PPE-DPC is shown in the bottom right of Fig. 5. This step assumes that predicted values $\widehat{thrldList}$ align with actual values, which means that the accuracy of predictions directly impacts the performance of fine-tuning. At this stage, three scenarios may occur:

- **Case1**: When $PT_R < \sum_i^N \widehat{PT_{S_i}}$, $\widehat{PT_{S_i}}$ needs to be reduced until it meets the rack capping power.
- **Case2**: When $PT_R > \sum_i^N \widehat{PT_{S_i}}$, $\widehat{PT_{S_i}}$ needs to be increased to enhance the performance of the throttled servers.
- **Case3**: $PT_R = \sum_i^N \widehat{PT_{S_i}}$, No adjustment to $\widehat{PT_{S_i}}$ is needed.

For Case 1 and Case 2, these are the situations that need to be addressed. The proposed power efficiency-aware power capping fine-tuning algorithm, a greedy strategy, is employed. The key idea is to consider the degree of change in $\Delta peff$ induced by power adjustment. For Case 1, where $\widehat{PT_{S_i}}$ must be reduced, priority is given to minimizing the change in $\Delta peff$, thereby reducing the performance impact on servers. For Case 2, where $\widehat{PT_{S_i}}$ needs to be increased, priority is given to maximizing the change in $\Delta peff$, enhancing server performance as much as possible. To achieve this, a quantitative relationship pair $< power, \Delta peff >$ is established. The $power$ is the server power, which is used as the key of the pair. And the value $\Delta peff$ of the pair, representing the change in $peff$ brought about by the change in $\Delta power$ of current power. A detailed explanation of the fine-tuning algorithm is provided in Example 1.

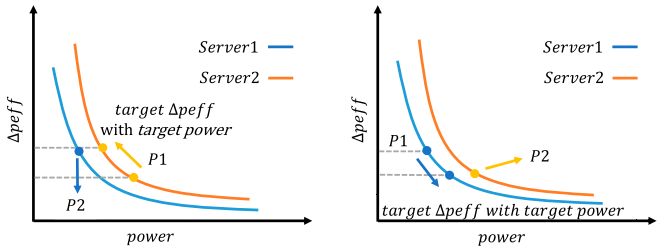


Fig. 6. Power fine-tuning based on $\Delta peff$.

Example 1: When power reduction is required (Case 1), as shown on the left side of Fig. 6, select the server with the smallest $\Delta peff(P1)$, i.e., Server 2. Reduce its capping power by $step$ until $\Delta peff(P1 - step * \Delta power) \geq \Delta peff(P2)$. At this point, Server 2 reaches $target\ \Delta peff$ with $target\ power$, and Server 1 becomes the server with the smallest $\Delta peff(P2)$. When an increase in capping power is required (Case 2), as shown on the right side of Fig. 6, select the server with the largest $\Delta peff(P1)$, i.e., Server 1. Increase its capping power by $step$ until $\Delta peff(P1 + step * \Delta power) \leq \Delta peff(P2)$. At this point, Server 1 reaches $target\ \Delta peff$, and $\Delta peff(P2)$ of Server 2 is the largest $\Delta peff$. In addition, if multiple servers have the same $\Delta peff$, traverse these servers and choose the server with the smallest/largest $\Delta peff$ under the same $step * \Delta power$ (where $step = 1$) to reduce/increase capping power. The core process of capping power adjustment is detailed in **Function 1** $cpower\_tuning()$.

Function 1: *cpower_tuning*()

1 **Input:** $srvList$, $\widehat{PT_S}List$, $const$
2 **Output:** update $\widehat{PT_S}List$
3 **while** $PT_R \neq \sum_i^N \widehat{PT_{S_i}}$ **do**
4 find s with $smallest/largest\ \Delta peff$ from $srvList$ with $const$
// If $const == true$, skip s that satisfy $s.\widehat{PT_S} \leq s.power / s.\widehat{PT_S} \geq s.power$
5 **if** s *is empty* **then**
6 break
7 **end**
8 find $s.target\ \Delta peff$ and $s.target\ power$ from $s. < power,\ \Delta peff >$ with $const$
// If $const == true$, the $target\ \Delta peff$ with $target\ power$ will be limitd by $s.power$
9 calculate $step = \frac{|s.\widehat{PT_S} - s.target\ power|}{\Delta power}$ by $< power,\ \Delta peff >$
10 update $s.\widehat{PT_S}$ with $step * \Delta power$
11 **if** $PT_R == \sum_i^N \widehat{PT_{S_i}}$ **then**
12 break
13 **end**
14 **end**

Algorithm 2: Power Efficiency-Aware Capping Power Tuning

1 **Input:** $srvList$, $\widehat{PT_S}List$
2 **Output:** $PT_S List$
3 $IncSrv = [\,]$, $DecSrv = [\,]$
4 **for** s *in srvList* **do**
5 **if** $s.power < s.\widehat{PT_S}$ **then**
6 $IncSrv$.append(s)
7 **end**
8 **else**
9 $DecSrv$.append(s)
10 **end**
11 **end**
12 **if** $PT_R \neq \sum_i^N \widehat{PT_{S_i}}$ **then**
13 $\widehat{PT_S}List =$ **Function 1**$(IncSrv/DecSrv, \widehat{PT_S}List, true)$
14 **end**
15 **if** $PT_R \neq \sum_i^N \widehat{PT_{S_i}}$ **then**
16 $\widehat{PT_S}List =$ **Function 1**$(IncSrv + DecSrv, \widehat{PT_S}List, false)$
17 **end**
18 $PT_S List$ = $\widehat{PT_S}List$
19 **return** $PT_S List$

Algorithm 2 presents the pseudo-code of capping power tuning, and the bottom right of Fig. 5 shows the overall process of this algorithm. In **Step One**, we establish the relationship between the original server power $s.power$ and the presetting capping power $s.\widehat{PT_S}$ to identify two server sets: $IncSrv$ and $DecSrv$ (Lines 4-11). Then, we evaluate the relationship between $\sum_i^N \widehat{PT_{S_i}}$ and PT_R, and get the $IncSrv$ and $DecSrv$ server sets. In **Step Two**, When $PT_R < \sum_i^N \widehat{PT_{S_i}}$, the first parameter of **Function 1** is set to $IncSrv$, else $DecSrv$, as shown in (Line 13). The third parameter, $const$, is set to $true$ because restoring $s.\widehat{PT_S}$ to its original power ($s.power$) implies that both $IncSrv$ and $DecSrv$ are in the same state, as illustrated in Fig. 7. For example, in Case 1, after tuning the servers of $IncSrv$, both $IncSrv$ and $DecSrv$ are under power provision to deal with current and future requests. Therefore, the design of constraints ensures fairness between $IncSrv$ and $DecSrv$. Similarly, in Case 2, $IncSrv$ and $DecSrv$ operate over power provision. Due to the constraints, it is possible that PT_R remains unequal to $\sum_i^N \widehat{PT_{S_i}}$. In such scenarios, both $IncSrv$ and $DecSrv$ are incorporated without constraints for further tuning, as shown in (Lines 15-17).

Time complexity. Once the power capping is triggered, the power capping values $PT_S List$ for each server are obtained by combining the online prediction results $\widehat{PT_S}List$ and **Algorithm 2**. The time complexity of (Lines 4-11) in **Algorithm 2** is $O(N)$. Assuming the worst-case scenario, the adjustment needs to be adjusted across the entire server power, donated by L, so the time complexity of (Lines 12-17) can be expressed as $O(N * L/\Delta power)$. Thus, the overall time complexity of **Algorithm 2** is $O(N + N * L/\Delta power)$.

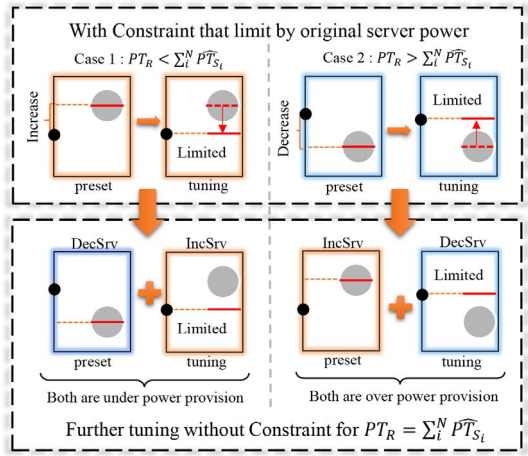
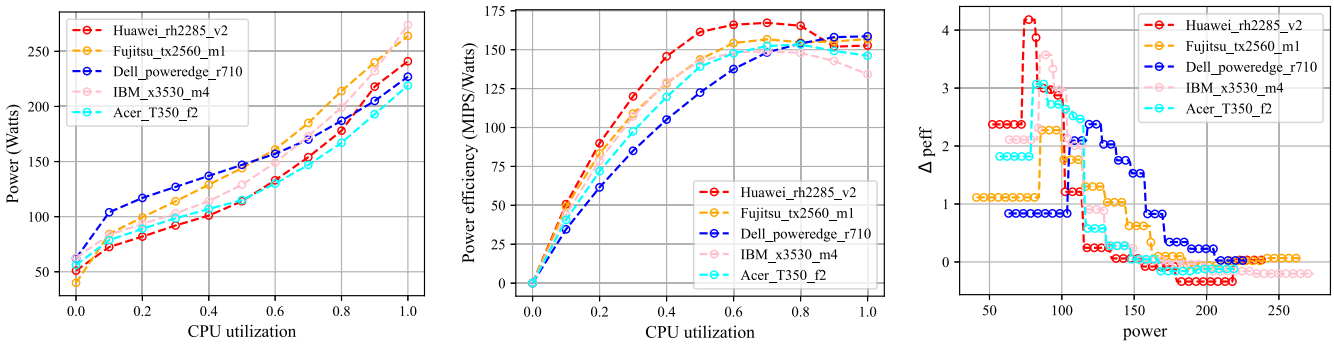


Fig. 7. Fairness optimization of power capping based on constraint mechanism.

V. Experimental Evaluation

A. Experimental Setup

Based on the modeling approach described above, we developed a simulation environment for a scenario with a single rack containing 20 servers. To investigate rack power capping with heterogeneous servers of varying power efficiencies, we randomly selected five different server types from SPEC, i.e.,



(a) Power curves of different servers (b) Power efficiency of different servers (c) $< power, \Delta peff >$ of different servers

Fig. 8. Performance differences in heterogeneous servers.

TABLE II
SERVER MODELS

Server Types	Cores	MIPS/Core	peak_peff	opt_u
Huawei_rh2285_v2	16	2300	167.3	0.7
Fujitsu_tx2560_m1	18	2300	156.8	1
Dell_poweredge_r710	12	3000	158.6	1
IBM_x3530_m4	16	2300	148.9	0.7
Acer_T350_f2	16	2000	153.3	0.8

TABLE III
DATASETS

Datasets	Average Load	Load Range	[Q1, Q2, Q3]
Low	Avg<0.35	[0.05, 0.98]	[0.23, 0.3, 0.4]
Medium	0.35<Avg<0.45	[0.05, 0.94]	[0.3, 0.38, 0.49]
Large	0.45<Avg	[0.15, 0.94]	[0.39, 0.47, 0.56]
Mix	Sampling from (Low, Medium, Large)	[0.05, 0.94]	[0.31, 0.4,0.5]

4 servers of each type in the rack. The detailed specifications are provided in Table II. Also, heterogeneous servers with different power, power efficiency and $< power, \Delta peff >$ are shown in Fig. 8. We can see from Fig. 8(b) and 8(c) that most servers will show lower $peff$ after the CPU utilization is less than 40%, and the $peff$ changes dramatically as the power decreases. The number of requests $w(t)$ received by the server at time t is calculated based on trace utilization $trace\_u$ and server processing speed as $w(t) = proc(trace\_u(t))$, and the unit request is considered as $\Delta w = 100MI$. This workload modeling method is the same as CloudSim [32]. In addition, consistent with Ref. [27], we only consider one type of request for rack power capping, i.e., cluster that provides a specific service. Different types of requests can be modeled as different Δw.

For the $trace\_u$, we used the $machine\_usage.csv$ dataset provided by Alibaba[2]. First, we divided the dataset into three groups: Low, Medium, and Large, based on average utilization. Then, we randomly selected 20 server utilization $trace\_u$ from each group (a total of 60 traces) to simulate the rack under different load pressures. Furthermore, we uniformly sample from these three selected datasets to obtain a fourth mixed dataset to simulate more complex scenarios. The total time $T = 5000$, the range, mean, and quartiles of the dataset used are summarized in Table III.

For comparison, we evaluate three advanced or well-known power capping methods:

- *Three band* [12]. An algorithm used in Facebook's Dynamo power management system, which applies high bucket priority for power capping.
- *PPCapping* [27]. A method that considers multiple server SLAs, evenly distributing the capping power across all servers.
- *PPW* [26]. A greedy strategy that prioritizes maximizing performance per watt for each server.
- *Ours*: The method proposed in this paper, a dynamic power capping approach driven by predictions and power efficiency, with $M = 4$ and $\Delta power = 2$.

The experiments in the following subsections aim to address the five key questions we are particularly interested in: ***Q1.*** How do power capping algorithms compare in performance and fairness at the same capping power? ***Q2.*** How does power utilization efficiency vary while meeting SLA requirements? ***Q3.*** What is the effect of different power adjustment/tuning intervals? ***Q4.*** How does prediction error impact the PPE-DPC? ***Q5.*** What is the performance of PPE-DPC in real application environment?

B. EXP1: Comparison of Performance and Fairness

To address ***Q1***, different capping powers are used for different datasets to compare the performance and fairness of different capping algorithms. We showcase the superiority of PPE-DPC by presenting various statistical metrics.

From the Table IV, it is evident that, except for the PPCapping, most algorithms are able to achieve p95 tail latency within $\Delta t = 1$. However, the Three band exhibits a significant gap in performance for the p99 tail latency. From an algorithmic perspective, these phenomena can be explained as follows. The PPCapping, based on the current workload information collected from all servers, employs a uniform capping strategy to achieve load balancing across servers. This approach can

[2]Alibaba Cluster-trace-v2018: https://github.com/alibaba/clusterdata/

TABLE IV
COMPARISON OF LATENCY (Δt) ACROSS DIFFERENT ALGORITHMS AND DATASETS

Group Name	Algorithms	p95 tail Latency	p99 tail Latency	Avg Latency
Small	Three band	1.128171	14.40384	1.474351
Capping	PPCapping	2.706031	3.982025	1.265876
Power	PPW	1	2.884117	1.066832
=2550	Ours	**1**	**1.596804**	**1.023881**
Medium	Three band	1	5.357111	1.304553
Capping	PPCapping	1.584444	3.394755	1.12567
Power	PPW	1	2.148799	1.054613
=2800	Ours	**1**	**1.429732**	**1.018581**
Large	Three band	1	8.443355	1.184328
Capping	PPCapping	2.132406	3.176417	1.217049
Power	PPW	1	3.41453	1.085126
=2950	Ours	**1**	**1.340611**	**1.015998**
Mix	Three band	1	14.52259	1.409845
Capping	PPCapping	1.849732	3.728974	1.165877
Power	PPW	1	2.334398	1.07706
=2710	Ours	**1**	**1.569749**	**1.022829**

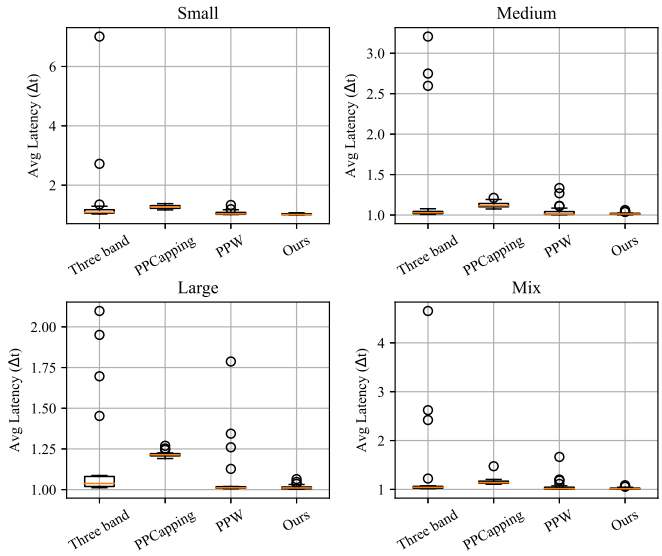


Fig. 9. Boxplots of experimental results comparing algorithms across different datasets.

reduce latency for all servers, but the latency ratio will be higher when the SLA is stricter. In contrast, the Three band uses a high-power-first capping strategy based on buckets, which sacrifices the performance of high-powerservers to ensure the performance of other servers. This leads to longer latency for requests on capping servers, but a smaller percentage of other servers are affected by power capping.

Additional support for these observations is provided by Fig. 9, which shows a boxplot of the average latency of the applications across the 20 servers. The Three band exhibit a greater number of outliers, indicating higher latency variability. In contrast, PPCapping shows tightly constrained upper and lower bounds, suggesting minimal disparity between maximum and minimum latencies, consistent with our analysis.

Additionally, PPW employs a greedy strategy that prioritizes the best performance per watt. While this approach enhances power efficiency, it overlooks future workload information and fairness among servers, potentially leading to extreme imbalances in performance. As shown in Table IV, PPW generally outperforms PPCapping. However, in the High dataset, PPW performs worse than PPCapping in terms of the p99 tail latency. The high boxplot reveals significant outliers for PPW, driven by its prioritization of power efficiency at the cost of other servers' performance. This tradeoff leads to performance degradation and makes PPW unsuitable forsome rare cases of heterogeneous servers with different power efficiencies.

Compared to the other algorithms, PPE-DPC consistently performs the best in terms of p95, p99, and average latency. This advantage arises from its consideration of future states, server power efficiency, and fairness between servers.

TABLE V
COMPARISON OF POWER UTILIZATION ACROSS DIFFERENT ALGORITHMS UNDER VARYING SLA REQUIREMENTS

Dataset	SLA	Capacity Increment Compared to Observed Peak Power			
	(p_x, Δt)	Three band	PPCapping	PPW	Ours
Small	(p95, 2)	4.8%	3.2%	6.1%	**6.8%**
	(p95, 5)	5.9%	8.1%	7.2%	**8.9%**
	(p99, 2)	2.0%	1.6%	3.4%	**4.6%**
	(p99, 5)	2.4%	5.7%	4.9%	**7.1%**
Medium	(p95, 2)	9.5%	8.2%	10.8%	**11.2%**
	(p95, 5)	10.4%	11.6%	11.5%	**13.3%**
	(p99, 2)	4.6%	3.4%	7.1%	**8.4%**
	(p99, 5)	6.2%	9.5%	8.5%	**10.1%**
Large	(p95, 2)	6.3%	5.1%	8.3%	**9.3%**
	(p95, 5)	7.7%	9.9%	9.3%	**10.7%**
	(p99, 2)	4.0%	1.9%	5.2%	**7.2%**
	(p99, 5)	4.3%	8.0%	6.4%	**9.1%**
Mix	(p95, 2)	5.7%	5.0%	6.6%	**7.8%**
	(p95, 5)	6.1%	8.0%	7.3%	**9.8%**
	(p99, 2)	4.1%	0.6%	4.6%	**6.0%**
	(p99, 5)	4.1%	5.5%	5.2%	**8.5%**

C. EXP2: Comparison of Power Utilization

To answer ***Q2***, we use the observed peak power as the baseline. Different Algorithms are compared across different datasets. Their power utilization is evaluated under varying SLA requirements. We set the tuning intervals $\Delta t = 5$.

Table V shows that with higher latency tolerance, such as (p95, 5) and (p99, 5), the PPCapping with load-balancing strategy generally achieves better power utilization than other algorithms, except for PPE-DPC. However, with lower latency tolerance, such as (p95, 2) and (p99, 2), the greedy strategies of Three Band and PPW outperform PPCapping, with PPW leading the Three band. This is because PPW accounts for the performance per watt of servers, whereas Three band focuses solely on high-power servers. In conclusion, it is clear that our algorithm consistently performs the best in improving power utilization under different SLA conditions. This is due to our algorithm achieving workload balancing, considering power efficiency and future information.

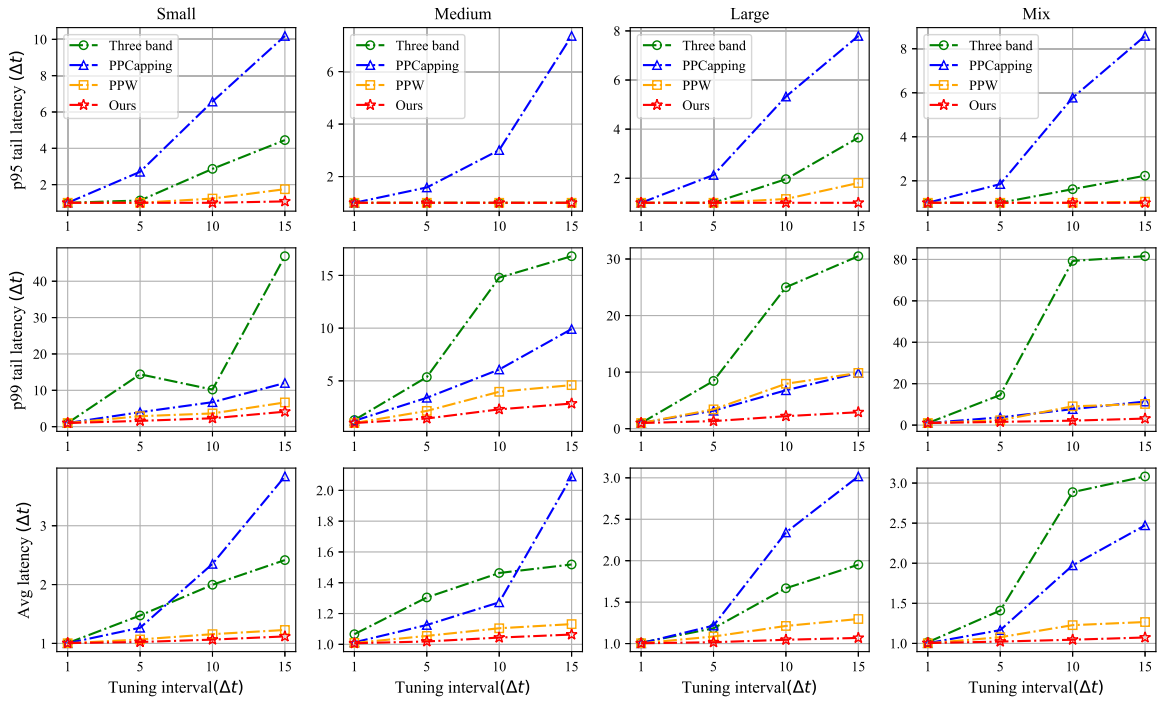


Fig. 10. Comparison of algorithm performance across different datasets and tuning intervals.

D. EXP3: Comparison of Different Tuning Intervals

For the ***Q3***, we compared the impact of four different adjustment intervals on dynamic power capping, specifically using intervals of $\Delta t = 1, 5, 10$, and 15. In cluster environments with high reliability and responsiveness, server states can be monitored more frequently, enabling power capping adjustments at shorter intervals.

First, an intuitive conclusion can be drawn from Fig. 10: both PPE-DPC and PPW demonstrate significant performance advantages across all four datasets and three metrics, with the proposed method clearly outperforming PPW. Additionally, the performance of the PPCapping and Three band in terms of the p95 and p99 tail latency aligns with the conclusions from the previous experiments. Specifically, the Three band sacrifices the performance of some servers to enhance that of others. This is evident in the p99 tail latency results, where, in the Mix dataset, $\Delta t = 10$ and 15 lead to latency increases of over 80.

From the adjustment time intervals, it can be observed that longer adjustment intervals generally result in more significant performance degradation. When the $\Delta t = 1$, all power capping algorithms exhibit very low latency, and there is almost no noticeable performance difference between them. This is a useful conclusion for data center operators, as it suggests that after triggering power capping, the frequency at which server status information is collected can be adjusted to optimize capping decisions. This approach allows operators to ensure the provision of QoS under power constraints. The proposed algorithm shows the smallest performance loss as the adjustment interval increases. This is due to the $downSampling()$ and $loadMovingAvg()$ strategies used in the prediction-driven capping power adjustment, which provide average requests over future adjustment intervals. This enables the capping algorithm to leverage changes in future states to guide adjustments in the capping power.

TABLE VI
PREDICTION ERROR OF *LOADMOVINGAVG*() ON DIFFERENT DATASETS

Dataset	Real MAE
Small	0.037607147919
Medium	0.03651096093
Large	0.03318339167
Mix	0.03569675626

E. EXP4: Comparison Under Different Prediction Errors

To address ***Q4***, which is a key focus of the research: How does the proposed prediction-driven power capping method perform under varying prediction errors? This question examines the impact of prediction errors on the effectiveness of the PPE-DPC in the cluster.

We performed full-volume forecasting using fourdatasets. Specifically, 20 historical data points were downsampled to $M = 4$ points, which were then used to predict a single future point. This predicted value was compared to the value obtained by downsampling 5 points from the true data. The prediction process was iterated continuously, and the Mean Absolute Error (MAE) was used to provide an intuitive measure of the difference between the predicted values and the actual values. The prediction error is shown in Table VI, where it does not exceed 0.04.

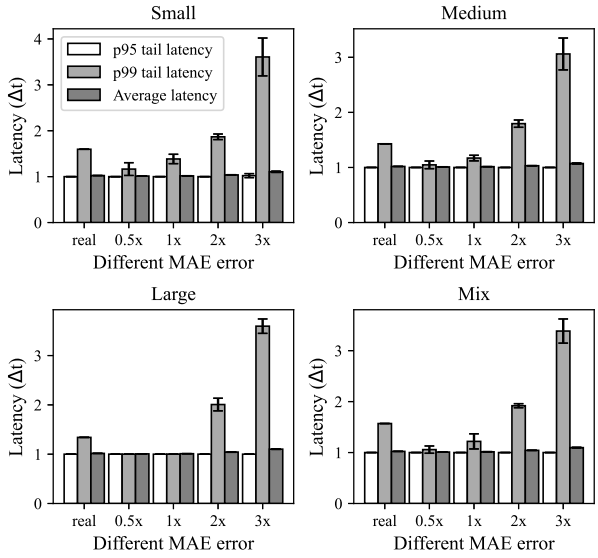


Fig. 11. Performance variations of PPE-DPC under different prediction errors.

TABLE VII
TESTBED IMPLEMENTATION

Testbed Hardware	Test Benchmark
Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz 32-Physical Cores, 220G-Memory	Application (Xapian)
Intel(R) Xeon(R) CPU E5-2660 @ 2.20GHz 16-Physical Cores, 126-Memory	Domain (Web Search)
Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz 12-Physical Cores, 110G-Memory	Dataset (English Wikipedia)

Therefore, using the real MAE error as a reference, we simulated different prediction errors by adding random noise. We designed prediction errors at 0.5x, 1x, 2x, and 3x of the real error to compare the performance impacts of varying levels of prediction error. We will repeat the experiment 5 times and calculate the mean and standard deviation for display. From the experimental results, as shown in Fig. 11, we can draw the conclusion that the more accurate the prediction (0.5x MAE), the smaller the performance impact caused by power capping. This demonstrates the effectiveness of **Algorithm 3**. When the prediction errors are 1x and 2x MAE, the performance is relatively close to the actual results. The standard deviation of the performance is still acceptable. As the prediction error increases, particularly when it reaches 3x MAE, the performance degradation reaches twice the original performance with more radical fluctuations. Compared with the **EXP1** results, where PPW performs second best, we observe that at 2x MAE, the proposed algorithm still outperforms PPW. However, at 3x MAE, its performance becomes worse than that of PPW.

F. EXP5: Comparison Based on Testbed

Finally, for ***Q5*** to demonstrate the effectiveness of PPE-DPC, we constructed a system prototype with heterogeneous servers based on the three different servers and web search benchmark [33] shown in Table VII. And random samples from the three types of traces (Small, Medium and Large) shown in Table III are converted to the Requests Per Second (*RPS*) of the corresponding servers based on the $trace\_u$, and integrated into the benchmark to build a load generator.

For the $proc$ of the server, it can be constructed based on $proc = (RPS/AvgLatency)$, which indicates how fast the server processes RPS requests per second. Correspondingly, the power, power efficiency and $< power, \Delta peff >$ curves of different servers can be constructed. In addition, we set the $T = 300s$, tuning interval $K * \Delta t = 5s$ and $M = 4$.

From Table VIII, it can be found that compared to different baselines, PPE-DPC is able to have the smallest latency, which is maintained across different datasets: *PPE-DPC<PPW≈PPCapping<Three band*. Compared to *w/o capping* (optimal latency), PPE-DPC also has smallest gap among the three latency metrics, with both the average and p95 latency across all datasets remaining within 4*ms* of *w/o capping*, illustrating the effectiveness of our approach.

In addition, we also test the actual decision overhead of PPE-DPC. As can be seen from Fig. 12, compared with other algorithms, although ours has a high overhead, the millisecond-level overhead (the maximum is 2.6 *ms*) is already much lower than the second-level overhead of the clustered collection of information from the servers, and it can adequately satisfy the decision latency required for the power capping of the clusters.

TABLE VIII
COMPARISON OF LATENCY (MS) ACROSS DIFFERENT ALGORITHMS AND DATASETS IN TESTBED

Group Name	Algorithm	p95 tail Latency	p99 tail Latency	Avg Latency
Small Capping Power =330w	*w/o Capping*	*8.510*	*10.693*	*3.373*
	Three band	19.973	494.183	18.207
	PPCapping	11.343	232.874	10.169
	PPW	12.137	385.438	15.052
	Ours	**10.215**	**148.522**	**6.489**
Medium Capping Power =340w	*w/o Capping*	*8.400*	*10.543*	*3.322*
	Three band	11.386	152.307	7.399
	PPCapping	9.846	55.573	4.845
	PPW	9.604	62.359	4.717
	Ours	**9.003**	**13.690**	**3.598**
Large Capping Power =330w	*w/o Capping*	*8.397*	*10.551*	*3.318*
	Three band	11.133	560.054	19.753
	PPCapping	10.259	281.676	9.991
	PPW	10.112	90.203	5.646
	Ours	**9.080**	**24.107**	**3.959**

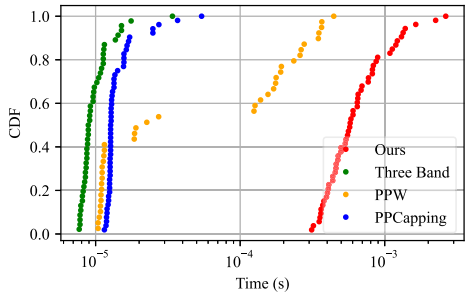


Fig. 12. Power capping decision latency.

VI. Discussion and Conclusion

A. Discussion and Future Work

Currently, the PPE-DPC method proposed in this paper still exists some optimization potential, which is a trade-off made for practical implementation in real-world scenarios. As discussed below:

The accuracy of long-term predictions needs to be improved. In **EXP4**, we showed that with precise prediction accuracy, PPE-DPC can further optimize SLAs. Therefore, future work could employ more advanced lightweight models to further optimize long-term prediction accuracy, such as those in [34], thereby improving QoS.

Performance can be enhanced through more granular modeling and decision-making. Considering both ease of use and feasibility, we adopted a greedy strategy to prioritize processing lists of performance-constrained servers and select the most power-efficient ones for enhancement. This approach may introduce some degree of error, which represents both the limitation of our approach and a necessary trade-off. In real cluster environments, obtaining fine-grained task information from servers online (e.g., remaining workload, waiting time, execution time) for precise modeling and solving incurs significant time overhead and latency errors. In contrast, our method only requires recording the arrival of tasks on each server to enable online solving, resulting in greater ease of use. How to construct more fine-grained mathematical problems and propose user-friendly and effective methods is the direction for further optimization.

Optimization for large-scale multi-rack clusters. In large multi-rack scenarios, power allocation at the rack level must also be considered [35], i.e., the optimization of capping power PT_{R_i} across multiple racks. Essentially, this problem involves cross-rack power cap decisions among server lists, which can be further modified and solved base on our methods. Furthermore, for large clusters, some nodes may have mixed workloads, and there remains a lack of in-depth research on how to construct power efficiency models of these nodes for power capping decisions.

B. Conclusion

In this paper, we focus on the study of cluster power capping, specifically considering how to reasonably allocate power budgets among heterogeneous servers in a multi-server environment to optimize the SLA of latency-sensitive applications. There are two key challenges in solving this problem: how to further optimize SLA in heterogeneous server scenarios and how to optimize SLA affected by adjustment time intervals. Based on the PPE-DPC proposed in this paper, we conducted extensive experiments using the Alibaba trace and performed a horizontal comparison with existing advanced solutions. The results validate that PPE-DPC is effective in optimizing SLAs, improving power utilization, and handling different capping power adjustment time intervals. Additionally, we further investigated the tolerance of PPE-DPC to prediction errors and found that it only underperforms relative to other optimal solutions that do not rely on prediction when the error is three times the actual prediction value.

References

[1] S. Li et al., "Thunderbolt: Throughput-optimized, quality-of-service-aware power capping at scale," in *Proc. 14th USENIX Symp. Oper. Syst. Des. Implementation (OSDI)*, USENIX Assoc., Nov. 2020, pp. 1241–1255.

[2] C.-H. Hsu, Q. Deng, J. Mars, and L. Tang, "SmoothOperator: Reducing power fragmentation and improving power utilization in large-scale datacenters," in *Proc. 23rd Int. Conf. Archit. Support Program. Lang. Oper. Syst. (ASPLOS)*, New York, NY, USA: ACM, 2018, pp. 535–548.

[3] C. Zhang et al., "Flex: High-availability datacenters with zero reserved power," in *Proc. ACM/IEEE 48th Annu. Int. Symp. Comput. Archit. (ISCA)*, 2021, pp. 319–332.

[4] L. A. Barroso, U. Hölzle, and P. Ranganathan, *The Datacenter as a Computer: Designing Warehouse-Scale Machines*. New York, NY, USA: Springer Nature, 2019.

[5] V. Kontorinis, et al., "Managing distributed ups energy for effective power capping in data centers," in *Proc. 39th Annu. Int. Symp. Comput. Archit. (ISCA)*, Washington, DC, USA: IEEE Computer Soc., 2012, pp. 488–499.

[6] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *Proc. 34th Annu. Int. Symp. Comput. Archit. (ISCA)*, New York, NY, USA: ACM, 2007, pp. 13–23.

[7] X. Fu, X. Wang, and C. Lefurgy, "How much power oversubscription is safe and allowed in data centers," in *Proc. 8th ACM Int. Conf. Autonomic Comput. (ICAC)*, New York, NY, USA: ACM, 2011, pp. 21–30.

[8] M. Savasci, A. Souza, L. Wu, D. Irwin, A. Ali-Eldin, and P. Shenoy, "SLO-power: SLO and power-aware elastic scaling for web services," in *Proc. IEEE 24th Int. Symp. Cluster, Cloud Internet Comput. (CCGrid)*, 2024, pp. 136–147.

[9] W. Lin, X. Luo, C. Li, J. Liang, G. Wu, and K. Li, "An energy-efficient tuning method for cloud servers combining DVFS and parameter optimization," *IEEE Trans. Cloud Comput.*, vol. 11, no. 4, pp. 3643–3655, Oct.–Dec. 2023.

[10] S. Chen, A. Jin, C. Delimitrou, and J. F. Martínez, "Retail: Opting for learning simplicity to enable QoS-aware power management in the cloud," in *Proc. IEEE Int. Symp. High-Perform. Computer Archit. (HPCA)*, 2022, pp. 155–168.

[11] S. Malla, and, and K. Christensen, "A survey on power management techniques for oversubscription of multi-tenant data centers," *ACM Comput. Surv*, vol. 52, no. 1, pp. 1–31, Feb. 2019.

[12] Q. Wu et al., "Dynamo: Facebook's data center-wide power management system," in *Proc. 43rd Int. Symp. Comput. Archit. (ISCA)*, Piscataway, NJ, USA: IEEE Press, 2016, pp. 469–480.

[13] V. Sakalkar et al., "Data center power oversubscription with a medium voltage power plane and priority-aware capping," in *Proc. 25th Int. Conf. Archit. Support Program. Lang. Oper. Syst. (ASPLOS)*, New York, NY, USA: ACM, 2020, pp. 497–511.

[14] J. Zhang, G. Yu, Z. He, L. Ai, and P. Chen, "DeepPower: Deep reinforcement learning based power management for latency critical applications in multi-core systems," in *Proc. 52nd Int. Conf. Parallel Process. (ICPP)*, New York, NY, USA: ACM, 2023, pp. 327–336.

[15] J. Sheng et al., "Learning cooperative oversubscription for cloud by chance-constrained multi-agent reinforcement learning," in *Proc. ACM Web Conf. (WWW)*, New York, NY, USA: ACM, 2023, pp. 2927–2936.

[16] X. Hou, C. Li, J. Yang, W. Zheng, X. Liang, and M. Guo, "Integrated power anomaly defense: Towards oversubscription-safe data centers," *IEEE Trans. Cloud Comput.*, vol. 10, no. 3, pp. 1875–1887, Jul.–Sep. 2022.

[17] S. Malla et al., "Coordinated priority-aware charging of distributed batteries in oversubscribed data centers," in *Proc. 53rd Annu. IEEE/ACM Int. Symp. Microarchit. (MICRO)*, 2020, pp. 839–851.

[18] T. Kidd, "Power management states: P-states, c-states, and package c-states," *Intel Xeon PhiTM Processor Documentation*, Apr. 2014. Available: https://software.intel.com/enus/articles/power-management-states-p-states-c-states-and-package-c-states

[19] H. David, E. Gorbatov, U. R. Hanebutte, R. Khanna, and C. Le, "RAPL: Memory power estimation and capping," in *Proc. 16th ACM/IEEE Int.*

Symp. Low Power Electron. Des. (ISLPED), New York, NY, USA: ACM, 2010, pp. 189–194.
[20] S. Malla and K. Christensen, "The effect of server energy proportionality on data center power oversubscription," *Future Gener. Comput. Syst.*, vol. 104, pp. 119–130, Mar. 2020.
[21] M. Dabbagh, B. Hamdaoui, and A. Rayes, "Peak power shaving for reduced electricity costs in cloud data centers: Opportunities and challenges," *IEEE Netw.*, vol. 34, no. 3, pp. 148–153, May/Jun. 2020.
[22] J. Ding and H. Hoffmann, "DPS: Adaptive power management for overprovisioned systems," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal. (SC),* New York, NY, USA: ACM, 2023, pp. 1–14.
[23] M. R. Hossen, K. Ahmed, and M. A. Islam, "Market mechanism-based user-in-the-loop scalable power oversubscription for HPC systems," in *Proc. IEEE Int. Symp. High-Perform. Computer Archit. (HPCA)*, 2023, pp. 485–498.
[24] Y. Li et al., "A scalable priority-aware approach to managing data center server power," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, 2019, pp. 701–714.
[25] Y. Li et al., "CapMaestro: Exploiting power redundancy, data center-wide priorities, and stranded power for boosting data center performance," IBM Research Rep. RC25680, Mar. 2018.
[26] L. Piga et al., "Expanding datacenter capacity with DVFS boosting: A safe and scalable deployment experience," in *Proc. 29th ACM Int. Conf. Archit. Support Program. Lang. Oper. Syst. (ASPLOS)*, New York, NY, USA: ACM, 2024, vol. 1, pp. 150–165.
[27] S. Wu et al., "Precise power capping for latency-sensitive applications in datacenter," *IEEE Trans. Sustain. Comput.*, vol. 6, no. 3, pp. 469–480, Jul.-Sep. 2021.
[28] M. Savasci, A. Ali-Eldin, J. Eker, A. Robertsson, and P. Shenoy, "DDPC: Automated data-driven power-performance controller design on-the-fly for latency-sensitive web services," in *Proc. ACM Web Conf. (WWW)*, New York, NY, USA: ACM, 2023, pp. 3067–3076.
[29] J. Stojkovic et al., "SmartOClock: Workload- and risk-aware overclocking in the cloud," in *Proc. ACM/IEEE 51st Annu. Int. Symp. Comput. Archit. (ISCA)*, 2024, pp. 437–451.
[30] A. G. Kumbhare et al., "Prediction-Based power oversubscription in cloud platforms," in *Proc. USENIX Annu. Tech. Conf. (USENIX ATC),* USENIX Assoc., Jul. 2021, pp. 473–487.
[31] P. Patel et al., "Characterizing power management opportunities for LLMs in the cloud," in *Proc. 29th ACM Int. Conf. Archit. Support Program. Lang. Oper. Syst. (ASPLOS)*, New York, NY, USA: ACM, 2024, vol. 3, pp. 207–222.
[32] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw., Pract. Exp.*, vol. 41, no. 1, pp. 23–50, 2011.
[33] H. Kasture and D. Sanchez, "TailBench: A benchmark suite and evaluation methodology for latency-critical applications," in *Proc. IEEE Int. Symp. Workload Characterization (IISWC)*, 2016, pp. 1–10.
[34] S. Lin, W. Lin, W. Wu, H. Chen, and J. Yang, "SparseTSF: Modeling long-term time series forecasting with 1k parameters," in *Proc. 41st Int. Conf. Mach. Learn. (ICML),* 2024, pp. 30211–30226.
[35] H. Huang, W. Lin, J. Lin, and K. Li, "Power management optimization for data centers: A power supply perspective," *IEEE Trans. Sustain. Comput.*, vol. 10, no. 4, pp. 784–803, Jul.-Aug. 2025.

Huikang Huang received the M.S. degree in computer science and theory from the South China Agricultural University, Guangzhou, China, in 2021. He is currently working toward the Ph.D. degree with the School of Computer Science and Engineering, South China University of Technology. His research interests include cloud computing, energy-efficiency optimization and reinforcement learning.

Weiwei Lin (Senior Member, IEEE) received the B.S. and M.S. degrees from Nanchang University in 2001 and 2004, respectively, and the Ph.D. degree in computer application from the South China University of Technology, in 2007. He is currently a Professor with the School of Computer Science and Engineering, South China University of Technology. His research interests include distributed systems, cloud computing, and AI application technologies. He has published more than 200 papers in refereed journals and conference proceedings. He has been a Reviewer for many international journals, including ICML, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON SERVICES COMPUTING, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON CYBERNETICS, etc. He is a distinguished member of CCF.

Xiaoxuan Luo is currently working toward the Ph.D. degree with the School of Computer Science and Engineering, South China University of Technology. His research interests include cloud computing and energy-efficiency optimization.

James Z. Wang (Senior Member, IEEE) received the B.S. and M.S. degrees in computer science from the University of Science and Technology of China. He received the Ph.D. degree in computer science from the University of Central Florida. He is currently a Professor with the School of Computing, Clemson University, South Carolina. His research interests include storage network, database system, distributed system, cloud computing and multimedia technologies. He is a senior member of ACM.

Keqin Li (Fellow, IEEE) received the B.S. degree in computer science from Tsinghua University, in 1985 and a Ph.D. degree in computer science from the University of Houston in 1990. He is a SUNY Distinguished Professor with the State University of New York and a National Distinguished Professor with Hunan University (China). He has authored or co-authored more than 1200 journal articles, book chapters, and refereed conference papers. He holds nearly 80 patents announced or authorized by Chinese National Intellectual Property Administration. He is among the world's top few most influential scientists in parallel and distributed computing, regarding single-year impact (ranked #2) and career-long impact (ranked #3) based on a composite indicator of the Scopus citation database. He is listed in Scilit Top Cited Scholars (2023–2025) and is among the top 0.02% out of over 20 million scholars worldwide based on top-cited publications in the last ten years. He is listed in ScholarGPS Highly Ranked Scholars (2022–2024) and is among the top 0.002% out of over 30 million scholars worldwide based on a composite score of three ranking metrics for research productivity, impact, and quality in the recent five years. He received the IEEE TCCLD Research Impact Award from the IEEE CS Technical Committee on Cloud Computing in 2022 and the IEEE TCSVC Research Innovation Award from the IEEE CS Technical Community on Services Computing in 2023. He won the IEEE Region 1 Technological Innovation Award (Academic) in 2023. He was a recipient of the 2022–2023 International Science and Technology Cooperation Award and the 2023 Xiaoxiang Friendship Award of Hunan Province, China. He is a Member of the SUNY Distinguished Academy. He is an AAAS Fellow, an AAIA Fellow, an ACIS Fellow, and an AIIA Fellow. He is a member of European Academy of Sciences and Arts. He is a member of Academia Europaea (Academician of the Academy of Europe).