Abstract

While resource oversubscription proves efficient for reusing data center infrastructure, it still faces power under-utilization problem. Previous studies propose market-based auction algorithms or DVFS-based schemes to utilize unused power resources under oversubscription. However, these methods ignore the power fluctuations of jobs, resulting in the exacerbated power overload. Meanwhile, the resource usage in the preceding time window is often used as the auction target for the next time window. This also introduces the lagged resource reallocation, which brings more power overload. Faced with the above problems, we propose FLAPS to reduce the power overload probability while maximizing the power utilization. FLAPS comprises a resource prediction module and a fluctuation-aware auction module. The prediction model provides accurate power usage prediction for the next time window by leveraging decile features of job power usage. The fluctuation-aware auction strategy takes power usage variance into account, reducing the power overload caused by power-fluctuating jobs. Our experiments show that FLAPS can reduce the power overload by 9%-10% on average while maintaining a resource utilization rate difference of less than 2.8%, compared to the state-of-the-art power management algorithms.

Keywords power oversubscription, market algorithm, overload probability, workload prediction

1 Introduction

Expenditures related to power resources constitute a significant portion of data center maintenance costs [1, 2]. To better utilize the power resources, data center usually employs capacity oversubscription [3–5]. By deploying more servers than the power capacity allows, the overall power utilization could be improved. Meantime, if the actual overall power demand exceeds the physical power capacity, the data center needs to use power capping to enforce limitations on the power usage of running jobs [6, 7].

Figure 1 shows the cluster-wide power resource utilization using the Google Cluster Trace 2019 [8, 9] for one day. Even with capacity oversubscription applied (area A), the power under-utilization problem still remains (area C). This is because the aggregate server power demand fluctuates and does not always stay at high levels. At the same time, capacity oversubscription also inevitably brings power overload (area B), and power capping brings an unpleasant experience to users.

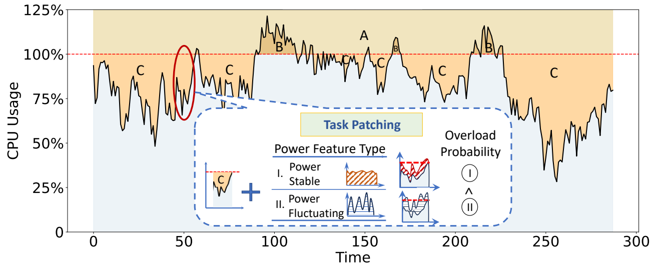

Fig. 1: The problems with existing resource oversubscription techniques.

Many previous studies use market-based auction strategies to utilize unused power capacity [10–13]. These methods encourage more users to use power resources by lowering the price of unused power [12] or avoid resource preemption by malicious users by raising the price of unused power [13]. As for the power overload cases, these methods release existing power usage by auctioning lower power prices in the future to reduce the impact of power overload [14].

Dynamic Voltage and Frequency Scaling (DVFS)-based algorithms, originally designed for power capping, are now widely used to enhance infrastructure resource utilization. A mainstream solution entails squeezing more servers into data centers by regulating the servers' frequencies to manage increased workloads [15, 16], thus improving energy efficiency while meeting rising demand.

While these methods may appear effective, *they ignore the potential abrupt fluctuations in power usage by jobs*. Existing methods typically choose a 5-minute time window, and use the average actual power usage in the previous time window for resource management of the next time window. Within a time window, a job's power usage may surge or drop abruptly, existing methods will be ineffective. In aggressively provisioned data centers, discrepancies between predicted average idle resources and runtime usage of newly added jobs can lead to frequent overload issues. For example, when the actual idle power is less than the anticipated average value and the newly introduced jobs consume more than the average, it can lead to cluster-wide power overload and necessitate power capping. If these jobs entail power fluctuations, the overall overload in the data center will increase significantly. Moreover, these methods utilize resource usage data from the preceding window to inform resource management for the subsequent window. Lagged resource data exacerbate the potential for power under-utilization and overload.

Faced with the above two problems, we find that jobs in the data center could be divided into power-stable jobs and power-fluctuating jobs. These jobs can be further distinguished by the power usage variance during job execution. In the job execution process, power-stable jobs will not cause large fluctuations in power usage. Compared with power-fluctuating jobs, it is more suitable to be used to fill unused power capacity without triggering power overload.

Meanwhile, we find that the lag problem in resource usage data can be solved by resource usage prediction methods. For power-stable jobs, accurate predictions can be made directly using the Long Short-Term Memory (LSTM) model. However, in the case of power-fluctuating jobs, direct attainment of accurate power usage results using the LSTM model is unfeasible. Hence, it becomes imperative to devise a dedicated power prediction model tailored for power-fluctuating jobs.

In this paper, we propose a Fluctuation-Aware Power Auction Strategy (FLAPS), which includes a power usage prediction model and a fluctuation-aware auction strategy. Leveraging the power prediction model, precise computation of available power resources within the next time window is achievable. Upon acquiring the accurate power resources, we employ a market-based auction strategy that capitalizes on power usage fluctuations to facilitate multiple jobs utilizing unused power allocations without power overload. Below, we provide a detailed exposition of our contributions.

- We add the variance of job power usage to the auction strategy to support jobs with smaller variances to use unused power capacity. This strategy can be seamlessly integrated into various existing auction strategies to ensure that newly scheduled jobs will not cause power overload and power capping.
- We build an enhanced LSTM model for power-fluctuating jobs. The power usage prediction module improves the vanilla LSTM model by incorporating a feature selection layer. This addition highlights the decile features of CPU utilization that are most relevant to the prediction results, thereby enhancing the training convergence speed and inference accuracy of the prediction model.

We evaluate our method based on real Google job traces [8, 9]. Experimental results show that we reduce the power overload by 9%-10% on average while maintaining a resource utilization rate difference of less than 2.8%.

2 Related Work

In recent years, several schemes have been developed for improving power utilization [4, 12, 13, 17] and predicting the workload [18–21]. However, these schemes all could not solve the power fluctuation problem.

Power Utilization. Many researches [4, 17, 22–24] keep an eye on oversubscription to enhance power utilization. One research direction [10–13] is to use market-based auction strategies to exploit unused resources in an attempt to maximize the benefits of the datacenter operator. Specifically, spotDC [12] incorporates a demand function based on pricing dynamics, whereby an escalation in prices precipitates a proportional reduction in the requisite power capacity. PowerGrab [13] strategically employs a demand function contingent on service-level considerations, intricately accounting for fixed expenses. This approach systematically evaluates whether the existing power capacity, given the prevailing system prices, is sufficient to sustain the mandated service level. At the same time, some efforts attempt to design methods, which could achieve peak shaving and valley filling for the resource usage. ELIS [25] builds a reward-based migrator designed to efficiently transfer excessive workloads to remote data centers, which can ensure the QoS of user-facing applications. Nodens [26] recognizes and focuses on the 'to-be-processed' workload, achieving fast QoS recovery through proactive management with just-enough resource allocation. None of these works consider the fluctuation of task power usage, which directly causes power overload.

Workload Prediction. Many research works use workload prediction to predict the resource usage of a task at the next time window, which further supports fine-grained task scheduling. ARIMA-based time series models [18, 19, 27] attain stationarity through differencing, subsequently fitting the model using a combination of autoregressive and moving average components. These models demonstrate limited accuracy, particularly in handling bursty cloud workloads. [28–30] use a machine learning-based model, such as Bayesian or KNN, to improve prediction accuracy and reduce manual intervention. PRACTISE [31] identifies relevant workload features with autocorrelation to predict target objects. Deep Belief Network [32] is also proposed to make both long-term and short-term predictions. However, these works did not take into account the cpu usage distribution feature, which resulted in their inability to obtain accurate results with long training times and complex network models.

Workload Analysis. Many studies focus on application characteristics and resource usage analysis on data centers. One paper from Google [21] delves into the intricacies of memory access analysis concerning specific applications within conventional CPU clusters. Another paper from Google [33] focuses on jobs distribution characteristics and

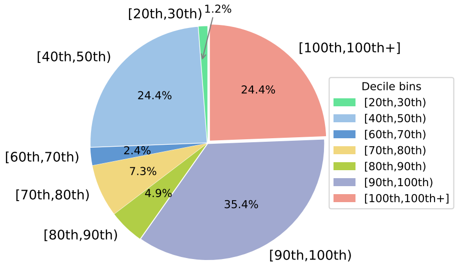

Fig. 2: The relation between jobs' power requests and their actual power consumption. The power consumption of jobs can be categorized into different intervals based on their deciles. The pie chart illustrates the distribution of requested values across these intervals. E.g., if the request value falls within the [100th, 100th+] range, it implies that the job never consumes more power than what is requested. If the value falls in the [90th, 100th] range, it suggests that the job's actual usage exceeds its requested power for 10% of the time at most.

the relationship between job executions and microstructures through comprehensive cluster analyses. In recent times, scholarly attention has been directed towards the examination of job dynamics within GPU clusters. A notable study by Facebook [20] systematically investigates the distributional patterns and inherent characteristics of deep learning tasks within existing clusters, concurrently optimizing the management of job processes. These researches are instructive for the power management. However, these papers are orthogonal to our paper. They will help the bidding function construction, which could be integrated with our paper.

3 Background and Motivation

In this section, we first demonstrate the power usage fluctuations in existing job traces. Secondly, we present the experimental results of directly using the LSTM model for power usage prediction. Third, we present the background of the market-based auction strategy in power management. Finally, we summarize the challenges for solving the above problems.

3.1 Power usage fluctuations

While existing power management strategies focus on 24.4improving power efficiency, they regrettably overlook the power usage fluctuations inherent in both to-be-scheduled jobs and running jobs. Significant fluctuations in many to-be-scheduled jobs can make the cluster highly prone to power overload and capping. Similarly, great fluctuations in running jobs result in inaccurate assessments of unused power capacity, consequently increasing the risk of power overload or under-utilization.

We demonstrate that power fluctuation is a prevalent issue in data centers by using data from the Google Cluster Trace [8,9] and the Alibaba cluster trace (AMTrace) [34,35]. Utilizing the records provided by the trace, we are able to infer the job's power usage [36,37] based on its cpu utilization.

The Google Trace models each job's cumulative distribution function (CDF) using sampling points and delivers cpu usage deciles every 5 minutes. We analyze the top-80 jobs that consume the most power within 48h, comparing their requested power with their actual power usage, as illustrated in Figure 2. Only 24.4% of jobs adher to their specified power usage limits throughout their entire execution. Additionally, 25.6% of the jobs exceed the requested power usage within 50% of the time. Furthermore, we collect the average power usage of all these jobs. Experimental results show that 81.7% of jobs have an average power usage less than 60th percentile. This means that these jobs request more power than average power usage within 40% of the time. These findings indicate that a significant portion of jobs exhibit substantial power usage fluctuations.

AMTrace is designed to analyze the microarchitectural resource characteristics of datacenter jobs. Leveraging the cycles, frequency and sampling time from the trace, we can calculate each container's resource utilization in the job. Furthermore, with the mapping between containers and applications in the *'container_meta'* table, we can summarize how each job utilizes resources. Due to the absence of job request values in AMTrace, comparing them with the actual usage values is not feasible. We examine the coefficient of variation (CV, CV = standard deviation / mean) of power usage in jobs to demonstrate the variability of resource usage over time. In Figure 3, we model the probability distribution of CV for jobs' power usage. Additionally, we plot the CV probability distribution for jobs from the Google Trace for comparison. In AMTrace, jobs are small-scale yet numerous, whereas in the Google Trace, jobs are larger-scale but fewer in number. Despite differing load patterns, both data centers show coefficient of variation peaks exceeding 30% (0.3), indicating high variability in power usage [38,39]. Our analysis indicates that power fluctuation during job execution is a common issue in data centers.

In the appendix (see 9), we model the issue of data center overload and present evidence that power fluctuations are more likely to trigger these incidents.

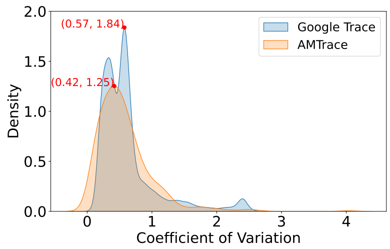

Fig. 3: CV probability distribution for jobs' power usage in both cluster traces.

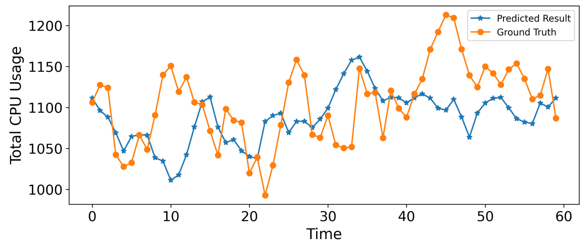

Fig. 4: Comparing LSTM prediction results with actual resource usage.

3.2 Power usage prediction

As stated in the above section, severe fluctuations in power usage may cause power overload. At the same time, fluctuations in power usage of running jobs may lead to imprecise assessments of unused power capacity. Present research predominantly relies on the average unused power capacity from the previous pricing interval, a method that may yield inaccurate estimations.

In addressing the aforementioned challenge, an intuitive approach is to use a sequence-based model such as LSTM to predict the average power usage of all jobs within the current pricing interval and subsequently derive the power usage for said interval. We train the LSTM model using the Google Trace dataset and conduct corresponding inference tests. The inputs comprise the job's conventional power usage-related features (excluding deciles) from the previous 24 time windows, while the output is the job's average power usage in the next time window. Figure 4 illustrates the prediction results with LSTM. As depicted in Figure 4, employing the LSTM model directly yields a Mean Squared Error (MSE) of 17.2. This suboptimal performance inadequately addresses the issue of lagged power usage problem.

3.3 Auction strategy in power management

The market mechanism serves as a negotiation platform between various users and data center operators, optimizing the interests of all involved parties. Operators can collect user feedback through pricing without accessing the private information of user jobs. The mechanism finds widespread application in real-life scenarios such as smart grids [40, 41], cloud/edge computing [42, 43], and beyond.

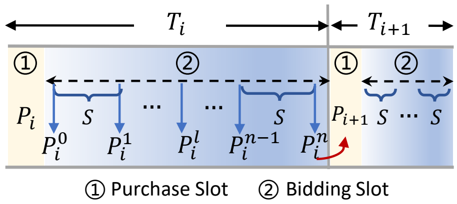

Fig. 5: The general bidding process.

In power management scenario, users first bid to express their runtime power demand, usually in the form of a demand function. Then the operator adjusts the market prices for different users or jobs. This bidding process needs multiple iterations to achieve a Nash equilibrium. The operator's objective is to maximize profitability, while users seek to optimize power utilization at minimal cost.

Figure 5 shows a general bidding process. The auction strategy first divides time into multiple pricing intervals T. Each pricing interval encompasses several bidding slots, represented by S. At the beginning of each pricing interval T_i, a constant system price P_i is calculated to guide each user to make their purchase decision. Subsequently, a sequence of bidding processes during T_i is employed to calculate the next system price, denoted as P_{i+1}. Within each bidding slot, the system computes an intermediate price P_i^l. The last P_i^l in the pricing interval T_i is the system price for the whole pricing interval T_{i+1}.

Each user employs a demand function to dynamically respond to the fluctuating system prices. Various demand functions are utilized in the bidding process. For instance, spotDC [12] incorporates the price-based demand function, wherein an increase in prices results in a corresponding decrease in the required power capacity. On the other hand, PowerGrab [13] leverages the service-level-based demand function, which factors in fixed expenses and assesses whether the available power capacity under the prevailing system prices is adequate to sustain the required service level. If the service level cannot be met, the bidding process ceases. The auction algorithm, conducted in multiple rounds of iteration, further enhances interactivity and user-friendliness. If the demand function exhibits varying functional expressions under different conditions, users can adjust their auction decisions in each round.

Market-based auction strategies have the capacity to accommodate multiple user bidding for available unused power capacity. Prior studies commonly rely on the utilization of the average unused power capacity from the preceding pricing interval as a basis for bidding.

3.4 Challenges

Based on the above analysis, we can summarize two challenges in solving power usage fluctuations.

- The prediction model needs to accurately predict power usage in the next time window, which supports accurate power bidding.
- Market-based auction strategies necessitate addressing diverse power usage patterns. This auction strategy entails filtering out jobs with stable power usage through the bidding process to mitigate the likelihood of power overload.

4 The Overview of FLAPS

FLAPS is a market-based power auction method that can support accurate power usage prediction and reduce the power overload probability. In market terms, datacenter users are consumers, intelligent power distribution units managed by datacenter operators are producers, and agents are used to assist in power redistribution in the market. Figure 6 shows the design overview of FLAPS, more details are as follows.

The prediction module forecasts the power usage of different jobs based on historical records. To ensure fine-grained power management, the data center divides time into multiple windows. The infrastructure module is responsible for collecting the power usage of all jobs since current time window T_i starts. Before the near end of T_i, the infrastructure module passes the collected power usage data to the power usage prediction module. Then the prediction module predicts the task power usage in the next time window T_{i+1}.

The pricing layer is mainly responsible for setting prices based on user bidding prices and system power supply conditions. The pricing algorithm takes into account both supply and demand, as well as the jobs' load fluctuation characteristics. Between the pricing layer and the application layer is a coordination layer. It consists of many agents. Agents are responsible for participating in bidding and local power management. It also supports several power controller drivers to translate the successful bidding decision into actual power management operations.

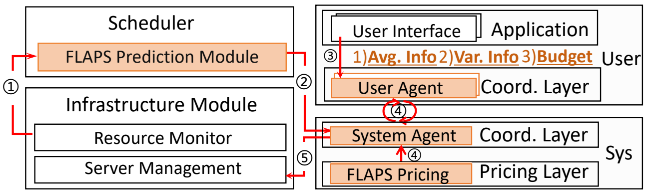

Fig. 6: FLAPS Framework Overview.

At the near end of T_i, multiple tenants can bid for the unused power capacity. This bidding process takes into account the power usage fluctuation of ready-to-submit jobs. If there is historical power usage data for one job, the historical power usage characteristic is used. If there is no historical data, the requested power usage is used. Before T_{i+1} starts, we can get the results of the bidding process, for example, task A obtain 2% of unused power capacity with price P.

Due to the lightweight coupling of components, FLAPS can be seamlessly deployed into existing data center management systems. The prediction module can be integrated into the scheduler and invoked to predict the power usage of jobs. It's convenient to deploy the pricing layer on existing cluster management systems, such as Google's cluster [13]. Current kernel control systems already monitor and maintain usage information for the underlying power infrastructure. Agents can be hosted on load servers or hypervisors.

5 Power Prediction Model

Data center's idle power can be calculated by subtracting the power usage of active jobs from its total capacity. Suppose N denotes the total quantity of resources in the data center, R^m_{i+1} denotes the predicted average power usage of job m within time window t_{i+1}, $Idle_{i+1} = N - \sum_{m=1}^{n} R^m_{i+1}$ denotes the predicted idle power resource during t_{i+1}. After reallocating $Idle_{i+1}$, the resources used for capping in the data center decrease. To avoid severe overload situations, ensuring the precision of the prediction R^m_{i+1} becomes imperative.

The analysis of the Google Trace reveals significant variations in the characteristics of different jobs. Some workloads exhibit periodic patterns over time, such as daily fluctuations, while others, potentially interactive, demonstrate completely irregular variations in their workloads. Given the presence of power fluctuations, both statistical models (e.g., ARIMA) and deep learning models (e.g., LSTM) exhibit limited accuracy in predicting irregular workloads. Therefore, we delve deeper into the trace to improve prediction accuracy.

The problem of power usage prediction can be formalized

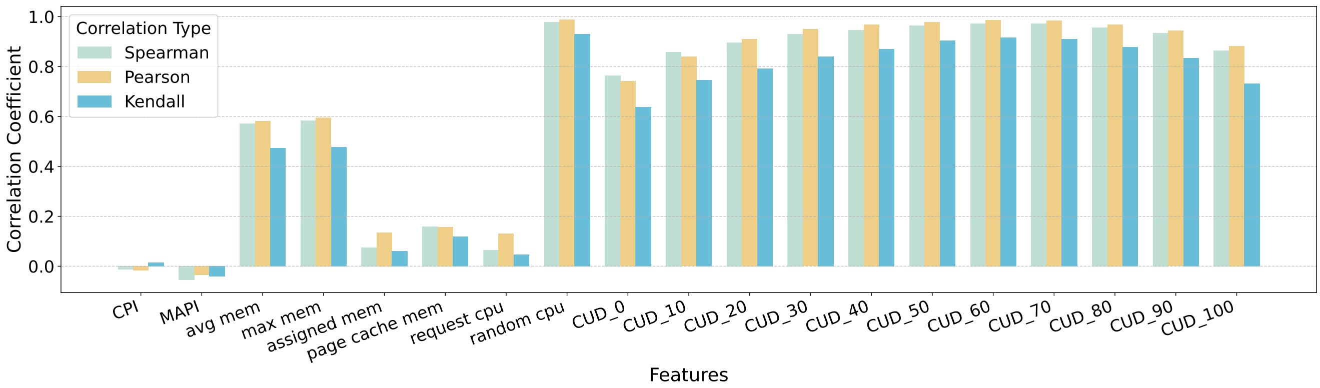

Fig. 7: Correlation of cpu usage distribution with avg cpu usage. CPI stands for cycles per instruction, while MAPI stands for memory accesses per instruction.

as follows: given the power records in the past t time windows $\{X^m_{i-t}, X^m_{i-t+1}, ..., X^m_i\}$ of job m, where X^m_i is a vector consisting of a set of features $\{x^m_{i1}, x^m_{i2}, ..., x^m_{in}\}$, we need to predict the average power usage for the next time window $\{x_{i+1}\}$.

We employ Pearson [44], Spearman [45] and Kendall [46] correlation coefficients to assess the relationship between average cpu usage and other features, as illustrated in Figure 7. Conventional features such as request values, cycles per instruction, and average memory are correlated with mean values. Still, there is a notable difference when compared to the correlation at each decile (field *'CPU usage distribution'* in the trace or *'CUD'*). Furthermore, among all the decile features, the highest correlation is observed from the 40th to 70th percentiles. Incorporating all deciles directly may introduce more noise rather than the accuracy. We build the following prediction model, as shown in Figure 8, which introduces decile feature in two ways to maximize its impact.

The power ranges exhibit notable variations among distinct jobs. Therefore, during data preprocessing, normalization is first performed separately for each job. Then the model performs feature selection after normalization, where one can filter out decile features with high correlation based on the statistical results shown in Figure 7. The selected features are analyzed by LSTM layers to obtain overall statistical patterns. Incorporating layer normalization after LSTM layers speeds up network convergence, improves training stability, and enhances model generalization. To address overfitting, a dropout layer is added after layer normalization.

In the neighboring time window, the distribution of resources exhibits greater similarity. We extract the decile feature at time t_i with fully connected layer and activation elu. Then the model concatenates the result with the output from the LSTM module to obtain the final result. The above design, which emphasizes the last decile, effectively enhances prediction accuracy. Meanwhile, as shown in Equation 1, Huber loss function [47] is chosen to train the model. The key advantage of Huber loss lies in its ability to strike a balance between the robustness of MAE and the sensitivity of MSE.

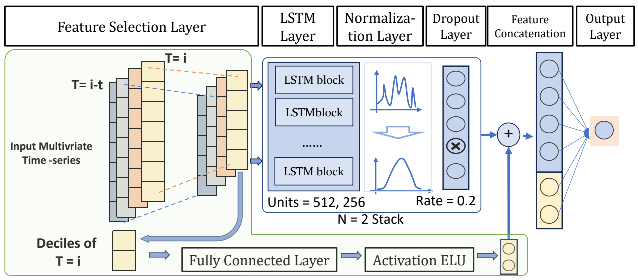

Fig. 8: Structure of the prediction model.

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{if } |y - f(x)| \le \delta \\ \delta(|y - f(x)| - \frac{1}{2}\delta) & \text{otherwise} \end{cases} \quad (1)$$

6 Fluctuation-Aware Auction Strategy

FLAPS uses a market-based auction algorithm to allocate idle resources. The auction process could reflect market supply and demand changes, and ensure a fair competitive environment for all users. In this section, we first introduce the entire auction process and then explain how to incorporate job variations into the auction algorithm.

6.1 The Auction Process

Figure 9 illustrates the workflow of the auction module, along with its interaction with other modules. The system can be di-

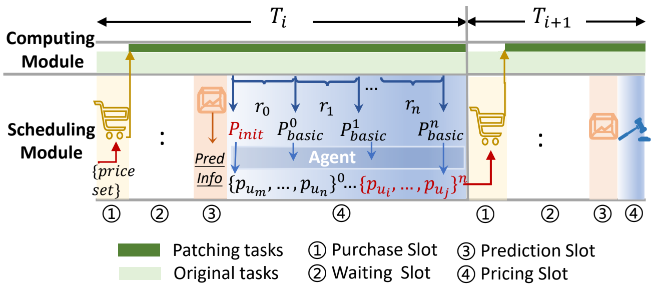

Fig. 9: A tatonnement bidding process.

vided into two modules, namely the computing module and the scheduling module. The computing module displays the jobs' execution status within the datacenter, including existing jobs and to-be-scheduled jobs. The scheduling module demonstrates the process of how to-be-scheduled jobs are selected. This process consists of four stages: the purchase slot, waiting slot, prediction slot, and pricing slot.

Suppose the data center allocates idle resources at intervals of T. At the beginning of T_i, there exists a purchase slot where users make resource purchases and submit jobs, relying on the auction outcomes from T_{i-1}. The scheduling module enters the waiting slot until the end of T_i. Then it enters the prediction slot to predict the idle power for T_{i+1}. Upon acquiring information about available idle resources and job execution details, the module finally enters a pricing slot.

The pricing process comprises multiple bidding rounds, $r_0, ..., r_n$. During the initial phase, the system calculates the initial price P_{init} based on the predicted quantity of idle resources and the variance of running jobs. Based on P_{init}, the agent module first updates bidding jobs' bid-price according to the job information submitted by the user. Then it compares the above bid-prices with jobs' budget, and collects the willing jobs, along with the corresponding price set $\{P_{u_0}, ..., P_{u_n}\}^0$. After each round, the system adjusts the basic price P^r_{basic} for the next bidding round according to the new supply and demand. The auction round repeats until it reaches the termination condition. After the final round, the winners acquire resources in the purchase slot at T_{i+1} and submit patching jobs.

6.2 The Auction Algorithm

This section specifically elaborates the details of the bidding algorithm, as shown in Algorithm 1. The system agent determines the initial price P_{init} based on the forecasted idle power and the current status of running jobs (Line 1). The initial price P_{init} is given by $P_{\text{init}} = \text{basic}_p \cdot \left(1 + \alpha \cdot CV_{\text{sys}}\right)$, where

Algorithm 1: Bidding Algorithm

Input: SysIdle, SysVar, jobs
Output: bidList
Data: *diff*, *ratio* /* constants for termination conditions */
1 $P_{init} \leftarrow$ calSysP($sysIdle, sysVar$);
2 $(P_{u_0}, ..., P_{u_n}) \leftarrow$ calJobP($P_{init}, job.avg, job.var$);
3 updateBid(); /* Acquire winning jobs */
4 $demandNew, varNew \leftarrow$ updateBidInfo();/* Calculate the updated demand and variance */
5 **while** $demandNew \geq sysIdle$ *or* $diff \geq ratio \times sysIdle$ **do**
6   $demandOld \leftarrow demandNew$;
7   $P^{r+1}_{basic} \leftarrow$ updateBaiscP($P^r_{basic}, sysIdle, sysVar,$
8   $demandNew, varNew$) /* Calculate the basic price P^{r+1}_{basic} for round $r+1$ */
9   updatejobP();
10   updateBid();
11   $demandNew, varNew \leftarrow$ updateBidInfo();
12   $diff \leftarrow$ |newDemand − sysIdle|;
13 **end**
14 **return** bidList;

basic_p is determined based on the unit rental price of the data center. When the data center experiences high volatility, overloads are more likely to occur. In these situations, idle power becomes scarce and must be allocated to more urgent and higher-value jobs. Therefore, α can be set to a larger value. If the running workload is stable, setting a lower α can encourage users to submit more jobs, increasing participation in bidding.

After obtaining the initial price P_{init}, each user agent calculates prices based on the job information in the bidding (Line 2). $P_{u_i} = P_{init} \cdot \beta^{1+\theta \cdot CV_{job}}$ or $P_{u_i} = P^{r+1}_{basic} \cdot \beta^{1+\theta \cdot CV_{job}}$, where the effect of β is similar to that of α. When confronted with numerous candidate jobs or when the data center cannot tolerate fluctuations, increasing β can promptly identify jobs with higher volatility. Unlike the linear changes in P_{init} or P^{r+1}_{basic}, job prices change exponentially based on their coefficient of variation. This approach can also be viewed as a penalty for jobs with high fluctuation. The varying importance of each job determines its maximum acceptable price. If this price falls below $P^r_{u_i}$, the auction for that job terminates (Line 3). The system agent updates the jobs participating in the next round of auctions (Line 4). When the total demand is less than the supply of idle power and the allocated quantity is close to the amount of idle power, the auction terminates. Otherwise, the system continuously adjusts prices to ensure a

Algorithm 2: P_{basic}^{r+1} Calculation

```
Input: P^r_basic, sysIdle, sysVar, demandNew, varNew
Output: P^{r+1}_basic
Data: β, δ /* constants */
flag ← if demandNew > sysIdle then 1 else -1;
P^{r+1}_basic ← P^r_basic + flag · ((varNew/sysVar) + 1) · β_2 ·
    ((demandNew/sysIdle) + δ);
return P^{r+1}_basic
```

successful auction (Lines 5-13).

The procedure to compute P_{basic}^{r+1} is outlined in Algorithm 2. When updating prices, the parameter $flag$ directs how the system price adjusts based on supply and demand. The extent of the change in P_{basic}^{r+1} depends on the supply-demand gap, which in turn leads to higher prices for scarce power. δ prevents the program from entering an infinite loop when *demandNew* is 0. The effect of β_2 is similar to β.

Table 1: Raw Data From Google Cluster.

Trace Characteristics	Value
Time span of trace	31 days
Cells (clusters)	8
Machines	96.4k
Machines per cell	12.0k
Amount of jobs	5M
Numbers of users	1951
Submitted tasks	80M
Scheduler events	1.5G
Resource usage records	7.0G

Table 2: Testbed Server Specification.

Name	Value
Model	Intel(R) Xeon(R) Gold 5218 2.30GHz
Cores	64 (32*2)
OS	Ubuntu 18.04 (kernel 4.15)
Memory	252GB
Disk	29TB

7 Evaluation

In this section, we first present the experimental setup of FLAPS. Secondly, we present the experimental results of the two sub-modules in FLAPS separately, including the accuracy of the prediction model and the effectiveness of the fluctuation-aware auction algorithm. Finally, we will demonstrate the performance improvement facilitated by FLAPS.

7.1 Experiment Setup

7.1.1 Dataset and Testbed

We generate the evaluation workload based on the Google 2019 Trace [8, 9]. The dataset is constructed from the *"instance_usage"* and *"instance_event"* table. We first aggregate the total CPU resources used by different jobs(collection_id), select the top 150 jobs after sorting, and extract the records for these jobs from the trace within the first two days. The trace records instance resource usage in 5-minute (300s) intervals, but few records might be shorter if the instance starts, stops, or undergoes an update within that window. To ensure data consistency and reduce noise data, we retain records that have a duration of 5 minutes. When constructing the model, records of the *top-30* jobs will be employed as the test set, and the remaining records will serve as the training set.

We also build a trace-based simulation testbed of FLAPS on the physical server, whose specifications are shown in Table 2. The *top-30* jobs are referenced as *"running jobs"*, and other jobs are referenced as *"patching jobs"*.

7.1.2 Baselines

Baseline for Prediction Model. We compare FLAPS with three other solutions to demonstrate the effectiveness of the predictive model. Specifically, 'History' refers to the scheme that substituting the idle resources of the current window for those of the subsequent window. The second baseline is 'ARIMA', with which p,q,r chooses the best parameter. 'FLAPS-N' refers to the model doesn't concatenate the cud feature of the last time window. We use three evaluation metrics, including MSE $= \frac{1}{n}\sum_{i=1}^{n}(\mathbf{y}-\hat{\mathbf{y}})^2$, MAE $= \frac{1}{n}\sum_{i=1}^{n}|\mathbf{y}-\hat{\mathbf{y}}|$ on each prediction window and MAPE $= \frac{1}{n}\sum_{i=1}^{n}|(\mathbf{y}-\hat{\mathbf{y}})/\mathbf{y}|$

Baseline for Overall Solution. To emphasize the enhancement in overall performance, we compare our design with other 6 kinds of the present resource allocation schemes as summarized in Table 3. **Bid-R** and **Bid-H** are both dynamic strategies that continue to allocate idle resources during job execution using a competitive market mechanism [12, 13]. Both strategies establish resource prices in auctions solely by considering the total demand. **Bid-H** doesn't predict idle resources but substitutes the future idle value with the current window's. **Bid-R** conducts the auction with the actual amount of idle resource. Tailored to specific scenarios, different DVFS-based strategies may vary slightly in the scale and adjustment ratios [15, 16], however their operational results consistently exhibit regular patterns. In **DVFS-80**, we

Table 3: Evaluated Resource Allocation Schemes.

Scheme	Acquisition of Idle Power	Auction or Allocate Metric
Bid-R	Real	Demand
Bid-H	History	Demand
DVFS-80	Real	Priority
DVFS-90	Real	Priority
FLAPS	Predict	Demand & Deviation
FLAPS-H	History	Demand & Deviation
FLAPS-R	Real	Demand & Deviation

assume the data center servers have a 20% frequency reduction, reducing job power consumption to 80%. The remaining power can be allocated to additional jobs based on their priority, which also run on servers operating at a 20% reduced frequency. Similarly, **DVFS-90** denotes servers with their frequency reduced to 90%. The **FLAPS** series represents a dynamic strategy that ensures safety by introducing standard deviation during auctions. The distinction among the three lies in the source from which the idle resource is acquired. **FLAPS-R** conducts auctions based on real idle resources from the trace, **FLAPS** uses the predictive results for auctions, while **FLAPS-H** employs the current value for the next window.

7.2 Prediction Model Accuracy Comparison

We initially analyze the resource usage patterns of jobs in the real-world trace. The temporal correlation of job's `cpu_average_usage` is demonstrated with Autocorrelation Function (ACF), as shown in Figure 10. Jobs can be roughly categorized into two types based on the strength of their correlation: regular and irregular. For jobs in (a) and (b), they exhibit a strong regularity in resource usage, with a lag order of approximately 288 (5min each point), indicating an approximate daily repeating pattern. Also, their correlation coefficient exceeds the confidence interval, indicating that the coefficient is statistically significant. For jobs in (c) and (d), they show less apparent regularity in resource usage, with correlation coefficients only covering the range of [-0.2, 0.2]. Especially the job in (d), the autocorrelation is poor when resource usage changes intensely.

Table 4 summarizes the evaluation results of all the four prediction methods on the Google trace, the best results are highlighted in boldface.

In **FLAPS** and **FLAPS-N**, the window lengths are set 12, 24, 48, 60 (1h, 2h, 4h, 5h). **ARIMA** predicts the target with all historical data, hence there exists no parameter indicating the window size. For **FLAPS** and **FLAPS-N**, improved prediction performance is observed when the window size is set to 24. For this size, **FLAPS** outperforms **FLAPS-N** on MAE, MSE, and MAPE, by decreasing 29.58%, 56.06%, and 38.98%. Our model excels in extracting valuable information from 'cud' features and filtering out noise from irrelevant 'cud' data, thereby minimizing interference with the model. **FLAPS-N** converge in approximately 30 epochs with a loss on the order of 1e-4, while **FLAPS** achieves a loss on the order of 1e-6 typically only requires 3 epochs. **FLAPS** accelerates the convergence speed of the vanilla LSTM model. Relative to **ARIMA**, **FLAPS** demonstrates a significant improvement, achieving reductions of 54.8%, 86.5%, and 28.4% in MAE, MSE, and MAPE. Our model showcases superior adaptability to scenarios characterized by substantial variations in workload. In comparison to **History**, our model shows a reduction of 39.56% in MAE, 79.49% in MSE, and 21.44% in MAPE. This demonstrates the model's effectiveness in addressing lagging issues.

Table 4: Comparison of predict results across four schemes.

Method	Size	MAE	MSE	MAPE
FLAPS	12	1.020	2.943	**10.527**
	24	**0.976**	**2.766**	11.907
	36	1.086	3.440	11.292
	48	1.161	4.044	10.626
	60	1.484	10.383	20.164
FLAPS-N	12	1.487	7.741	16.566
	24	1.386	6.295	19.515
	36	1.405	6.645	17.311
	48	1.498	7.295	23.52
	60	2.248	16.573	27.805
ARIMA	-	2.162	20.551	16.632
History	-	1.615	13.491	15.157

We compile the total resource predictions for 30 running jobs and compare them with their actual values across 4 schemes, as depicted in Figure 11. Figure 11 compares the total predicted value and the total actual value across the four schemes. Due to space constraints, we provide an example with 50 predicted points. The figure reveals that, upon aggregating all the predicted values of diverse jobs, the **FLAPS** predictions closely align with the actual total resources in the data center. **FLAPS-N** accurately forecasts the evolving

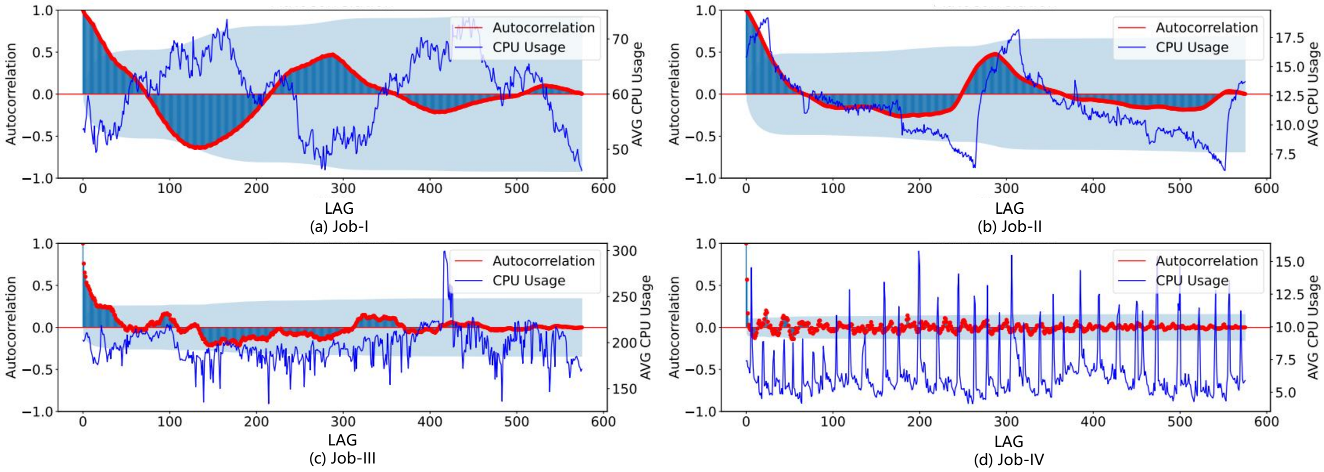

Fig. 10: The Autocorrelation Function (ACF) figure of jobs on the 'cpu_average_value'. The autocorrelation curve calculates the correlation between the average CPU value at the current time and the values at subsequent time points. The light blue shaded area in the graph indicates the range of the confidence interval. When a calculated autocorrelation coefficient falls outside this interval, it indicates that the coefficient is statistically significant.

trend of total resources in the data center, yet the predicted values exhibit a noteworthy margin of error. The **ARIMA** model's prediction for the total resource amount shows a notable margin of error when compared to the actual values. This discrepancy arises from the aggregation of subpar predictions for several irregular jobs.

7.3 Flucuation-aware Auction Results

7.3.1 The Variation in Demand and Standard Deviation

Figure 12 depicts the fluctuations in both the overall resource demand and total standard deviation throughout the auction process. As shown in Figure 12 (a), When the initial price P_{init} is low, the total demand exceeds the target idle resource. As the auction rounds increase, the overall demand and standard deviation gradually decrease. When the total demand falls below the target value and the difference is within a specific threshold, the auction concludes. In contrast, when the initial price P_{init} is high, both the overall demand and standard deviation increase as the number of auction rounds progresses, converging towards the target value, as shown in Figure 12 (b).

7.3.2 Price

Figure 13 illustrates the variation in prices during the auction process. As the total demand surpasses the idle resources (Figure 13 (a)), P^r_{basic} gradually rises. The price varies significantly among jobs with different features. For the losing job, a higher standard deviation per unit resource results in a higher auction price and higher growth rate, reaching their price budget more rapidly. In this manner, the system promptly filters out power-flucation jobs. If opposite, P^r_{basic} gradually decreases, facilitating the procurement of resources for jobs with lower per-unit standard deviation, as shown in Figure 13 (b).

7.4 Impact on System Performance

7.4.1 Difference in Patching Jobs

Figure 14 compares the characteristics of winning jobs across various schemes, including **FLAPS** and **Bid-H**. Since **Bid-H** barely distinguish between power-stable and power-fluctuating jobs, the unit price remains uniform across jobs with varying characteristics. As the points encircled in Figure 14 (b), although the jobs exhibiting a higher standard deviation in unit resources, they can still achieve successful bids owing to the higher maximum price. **FLAPS** impose pricing penalties on power-fluctuating jobs, resulting in the failure of these jobs in the auction. They comprehensively consider the characteristics of jobs along with their maximum price, prioritizing jobs with smaller standard deviations. In the **DVFS** series, jobs are selected based on their priority, without considering price.

7.4.2 System Performance Comparison

With the job lists obtained from different schemes, we simulate the usage curve and calculate the system performance, the details are outlined as follows. Since time is discretized

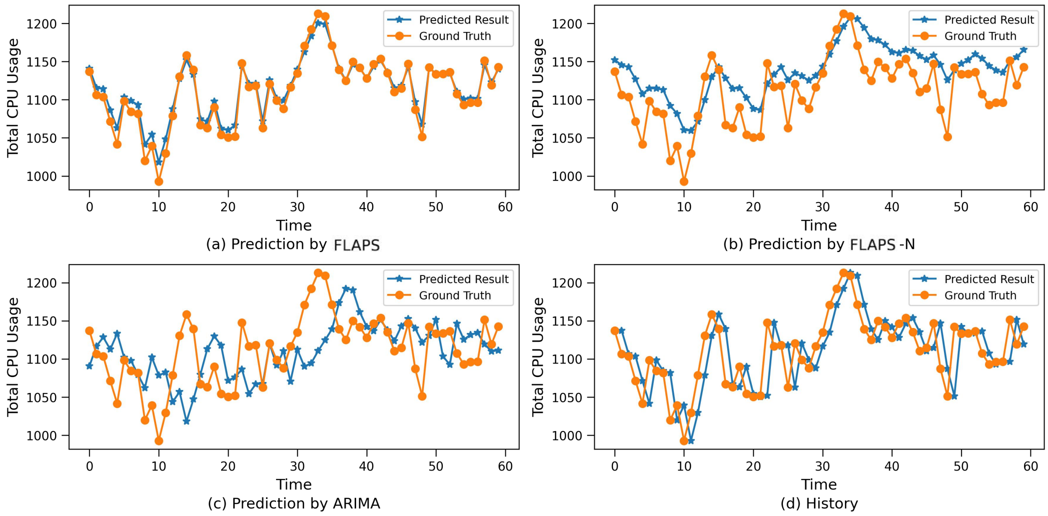

Fig. 11: Comparison of different predictive schemes on total resources.

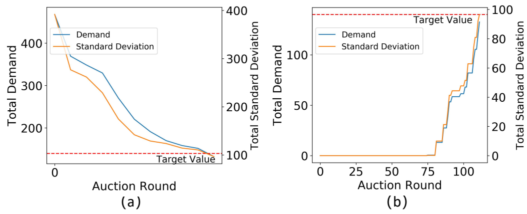

Fig. 12: The variation curve of total demand and total standard deviation during auction.

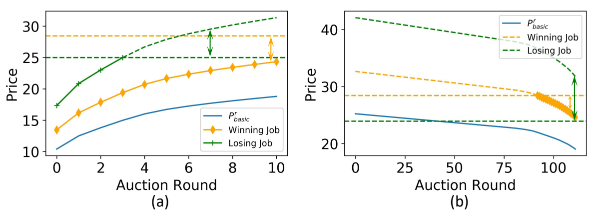

Fig. 13: The variation curve of basic price and job price. The horizontal dashed line represents the maximum price acceptable for the job. In the price curves of two jobs, solid lines indicate prices accepted by the jobs, while dashed lines represent prices exceeding the maximum price. The arrows illustrate the difference between the auction price and the maximum price of the jobs.

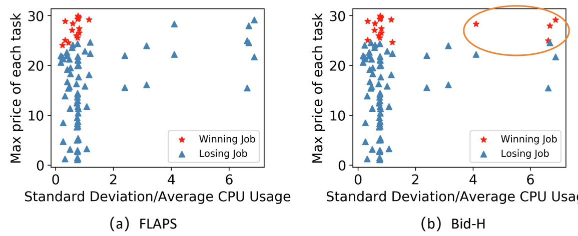

Fig. 14: The comparison of winning jobs between **FLAPS** and **Bid**. The x and y axes represent the job's two attributes: The x-axis represents the standard deviation introduced per unit resource, while the y-axis signifies the maximum price per unit resource.

into 5-minute slots, the two-day trace we construct consists of a total of 288 slots, where a subscript i signifies the i-th 5-minute interval. According to section 2, the prediction model performs best with a window size of 24. Hence, the comparison of the schemes starts from $i = 25$. In addition, certain slots have few idle power (less than 5% of the total rated power), and these are beyond the scope of our discussion. Utilizing the fields 'cpu_usage_distribution' and 'average_cpu' of each job, we first simulate the resource usage curve of each job for each slot i in seconds, totaling 300 points. Then we aggregate the overall resource curves within each slot based on the job lists from different schemes. Based on the curve, we determine the overload ratio and resource utilization for each schemes at time slot i.

The results of different slots are illustrated in the CDF graph, as shown in Figure 15. Contrasted with **FLAPS-H**, **FLAPS** demonstrates superior prediction accuracy, leading to a more substantial enhancement in resource utiliza-

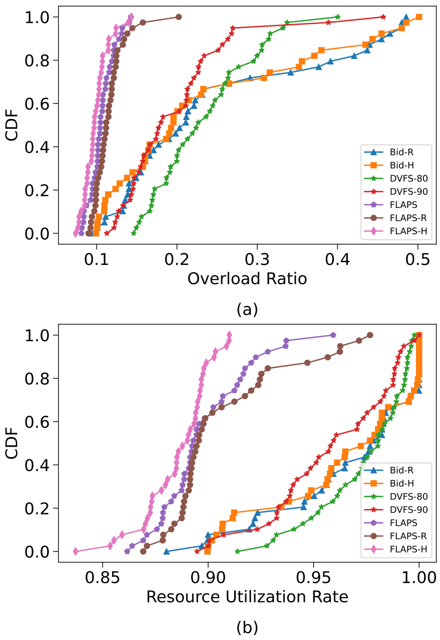

Fig. 15: Comparison of four schemes on system performance.

tion. For **FLAPS** series, the overall overload capacity is relatively small. This suggests that even in the presence of overload, the situation remains manageable and does not pose a threat to the normal operation of the data center. The overload capacity ratio is concentrated between 8% and 13% for **FLAPS** series. In contrast, the **Bid** series displays the highest overload ratio, peaking at 50%. Furthermore, in over 60% slots, its overload ratio exceeds 15%. On average, to attain a 5% enhancement in resource utilization, **Bid** series introduces an overload capacity approximately 30%, potentially resulting in significant downtime and disruptions. In the **DVFS** series, lowering server frequency extends jobs' execution time, boosts CPU utilization, and improves infrastructure efficiency. Lower server frequencies lead to higher overall resource utilization in the data center. For example, **DVFS-80** achieves higher utilization than **DVFS-90**. On the other hand, while reducing server frequency dampens jobs' power fluctuations, adding more jobs can still combine to overload the data center. Lower server frequencies result in higher overload rates in data centers. **DVFS-80** experiences overloads of around 15%-40%, while **DVFS-90** sees around 10%-45%.

FLAPS can reduce the power overload by 10.89% and 9.12% on average compared to **Bid-H** and **DVFS-90**. With an overload ratio of no more than 15%, the discrepancy in resource utilization between **FLAPS** and **Bid-H**, as well as **DVFS-90**, is 2.4% and 2.8%, respectively. Therefore, utilizing **FLAPS** not only facilitates a well-rounded enhancement in resource utilization but also ensures the security of the data center.

8 Overhead of FLAPS

We evaluate the overhead introduced by the FLAPS framework, accounting for the costs of both the prediction model and the auction algorithm. The prediction model's overhead arises from training and inference. Our model employs around 5 million parameters, and the training dataset in the prototype system comprises 42K records. During the training phase, We preset 40 epochs with an average early stop at 30 epochs. Each epoch takes 100 seconds on average, making the total time approximately 3000 seconds. After the initial model construction, frequent updates are unnecessary. In cases of significant load characteristic changes, incremental training with the new trace is feasible. Given the model's good initial weight, parameter adjustments are minor. Early stopping typically occurs around 20 epochs, and adding 1k data takes about 50 seconds. During the inference phase, the average time cost for predicting a single job is 40ms. In production scenarios involving the prediction of multiple jobs, increasing parallelism can ensure that the prediction time remains controllable. In the prototype system, the auction algorithm typically runs for about 50ms to 80ms. With a large number of jobs, adjusting the price coefficient can reduce the algorithm's execution time.

9 Conclusion

In this paper, we propose a Fluctuation-Aware Power Auction Strategy (FLAPS), which aims to reduce power usage overload. FLAPS identifies the previously unnoticed power fluctuation and further emphasizes that it can be recognized by the variance in jobs. Firstly, FLAPS enhances the LSTM model by adding a feature selection layer. Second, FLAPS employs a fluctuation-aware auction mechanism designed to prioritize power-stable jobs. We have implemented FLAPS and the experimental results show that, compared to the state-of-the-art power management algorithms, FLAPS reduces the power overload by 9%-10% on average while maintaining a resource utilization rate difference of less than 2.8%.

Appendixes

In this Section, we model the issue of data center overload and provide theoretical analysis that power fluctuations are more likely to trigger these incidents.

Appendix A Problem Definition

We list the notations of major variables in Table 5. Define E be the event of whether the data center is overload or not. $E = 1$ iff the data center is overload, otherwise $E = 0$. Define Q as the event that the data center retains idle resources during the execution of $\mathcal{M}$, and allocates them to $\tilde{\mathcal{M}}$. The problem is to determine how to select job set $\tilde{\mathcal{M}}$ to minimize the probability of event E occurring, given the occurrence of event Q, i.e. $\min_{\tilde{\mathcal{M}}}\{\Pr(E = 1|Q)\}$.

Appendix B Probability Calculation

We derive $\Pr(E = 1|Q)$ according to its definition, demonstrated in Eq. 2, the explanation of the key steps is as follows. $\Pr(E = 1 \cap Q)$ denotes that upon incorporating the successfully auctioned tasks into the running job queue, the actual resource usage of the data center surpasses the total rated resource. Hence, $\Pr(E = 1 \cap Q) = \Pr(Y \geq N)$. Since we have supposed there exists idle resources in the data center, and these idle resources are allocated to $\tilde{\mathcal{M}}$, we can obtain $\Pr(Q) = 1$. Since the tasks execute independently within containers [48], there is no correlation among their resource usage. Hence, X_m and $X_{\hat{m}}$ can be treated as independent variables. According to the principle of independence, the joint probability is the dot product of individual probabilities.

$$\begin{aligned}
&\Pr(E = 1|Q) \\
&= \frac{\Pr(E = 1 \cap Q)}{\Pr(Q)} = \frac{\Pr(Y \geq N)}{1} = \sum_{y \geq N} \Pr(Y = y) \\
&= \sum_{x_1,\ldots,x_{\hat{m}}: x_1+\cdots+x_{\hat{m}} \geq N} \Pr(X_1 = x_1, \cdots, X_{\hat{m}} = x_{\hat{m}}) \\
&= \sum_{x_1,\ldots,x_{\hat{m}}: x_1+\cdots+x_{\hat{m}} \geq N} \Pr(X_1 = x_1) \cdot \ldots \cdot \Pr(X_{\hat{m}} = x_{\hat{m}})
\end{aligned} \tag{2}$$

Appendix C Simplification of the Formula

According to Equ. 2, the computation of $\Pr(E = 1|Q)$ entails calculating all $\Pr(X_i = x_i)$ where $X_i \in \mathcal{M} \cup \tilde{\mathcal{M}}$ based on jobs' history records and the corresponding $\Pr(X_i = x_i)$ are multiplied and then accumulated according to different combinations. The equation exhibits high time and space complexities, rendering it inappropriate for implementation within auction algorithms. Instead of direct computation, we found that there exists an upper bound for the overloading probability formula, as proven in Theorem 9.

Table 5: Notations in the Problem Formulation

Notation	Description
m	Variable, the running job.
$\hat{m}$	Variable, the auction job candidates.
$\mathcal{M}$	Set, consisting of all running jobs.
$\tilde{\mathcal{M}}$	Set, consisting of all jobs participated in the auction.
X_m	Discrete random variable, the resource usage of job m over time.
$X_{\hat{m}}$	Discrete random variable, the resource usage of job $\hat{m}$ over time.
N	Constant, the total quantity of resource in the data center.
Y	Discrete random variable, the total resource usage of the data center over time.

Theorem. *Suppose $M = X_1 + \ldots + X_n$ and $\hat{M} = \hat{X}_1 + \ldots + \hat{X}_n$, then $Y = M + \hat{M}$. $E(M)$ and $E(\hat{M})$ denote expected value of M and $\hat{M}$, $\sigma(M)$ and $\sigma(\hat{M})$ denote standard deviation of M and $\hat{M}$. Then it follows:*

$$\Pr(E = 1|Q) \leq \frac{\sigma(M)^2 + \sigma(\hat{M})^2}{(N - E(M) - E(\hat{M}))^2}$$

Proof.

$$\begin{aligned}
&\Pr(E = 1|Q) \\
&= \Pr(Y \geq N) = \Pr(M + \hat{M} \geq N) \\
&= \Pr(M + \hat{M} - E(M + \hat{M}) \geq N - E(M + \hat{M})) \\
&\leq \Pr(|M + \hat{M} - E(M + \hat{M})| \geq N - E(M + \hat{M})) \\
&\leq \frac{\sigma(M + \hat{M})^2}{(N - E(M + \hat{M}))^2} \text{(According to Chebyshev's inequality)} \\
&= \frac{\sigma(M)^2 + \sigma(\hat{M})^2}{(N - E(M) - E(\hat{M}))^2}
\end{aligned} \tag{3}$$

The explanation for the final step in Equ.3 is as follows. Since each task executes in an isolated container environment, the variables $X_1, \ldots, X_m$ in M and $X_{\hat{1}},\ldots,X_{\hat{m}}$ in $\hat{M}$ are mutually independent. According to the linearity of expectation, $E(Y) = E(M + \hat{M}) = E(M) + E(\hat{M})$. As the variables are uncorrelated, the covariance between variables is 0, then $\sigma(Y)^2 = \sigma(M + \hat{M})^2 = \sigma(M)^2 + \sigma(\hat{M})^2$. □

Appendix D Analysis

When allocating the same idle resources to different sets of jobs $\hat{M}_1$ and $\hat{M}_2$, then $E(\hat{M}_1) = E(\hat{M}_2)$. We apply the Equ. 3 with the variable $\hat{M}_1$ and $\hat{M}_2$, we can obtain that

$$\mathrm{Pr}_{\hat{\mathrm{M}}_1}(E = 1|Q) \leq \frac{\sigma(M)^2 + \sigma(\hat{M}_1)^2}{(N - E(M) - E(\hat{M}_1))^2} \quad (4)$$

$$\mathrm{Pr}_{\hat{\mathrm{M}}_2}(E = 1|Q) \leq \frac{\sigma(M)^2 + \sigma(\hat{M}_2)^2}{(N - E(M) - E(\hat{M}_2))^2} \quad (5)$$

If $\sigma(\hat{M}_1)^2 > \sigma(\hat{M}_2)^2$, $\sup\{\mathrm{Pr}_{\hat{\mathrm{M}}_1}(E = 1|Q)\} > \sup\{\mathrm{Pr}_{\hat{\mathrm{M}}_2}(E = 1|Q)\}$. For different auction job set, if it has a smaller upper bound, all possible probability values are lower, under which relative safety can be assured. *Hence, when distributing identical resources, a smaller standard deviation (or variance) within the chosen set implies a relatively safer data center.*

References

1. Synergy Research Group. Hyperscale operator capex returns to growth mode in q3. `http://bit.ly/q3-2019-hyperscale-capex`, 2020.
2. Inc. Gartner. Gartner says global it spending to grow 1.1 percent in 2019. `http://bit.ly/gartner-2019-04-07.`, 2020.
3. Alok Gautam Kumbhare, Reza Azimi, Ioannis Manousakis, Anand Bonde, Felipe Frujeri, Nithish Mahalingam, Pulkit A Misra, Seyyed Ahmad Javadi, Bianca Schroeder, Marcus Fontoura, et al. {Prediction-Based} power oversubscription in cloud platforms. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 473–487, 2021.
4. Varun Sakalkar, Vasileios Kontorinis, David Landhuis, Shaohong Li, Darren De Ronde, Thomas Blooming, Anand Ramesh, James Kennedy, Christopher Malone, Jimmy Clidaras, et al. Data center power oversubscription with a medium voltage power plane and priority-aware capping. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 497–511, 2020.
5. Sriram Govindan, Jeonghwan Choi, Bhuvan Urgaonkar, Anand Sivasubramaniam, and Andrea Baldini. Statistical profiling-based techniques for effective power provisioning in data centers. In *Proceedings of the 4th ACM European conference on Computer systems*, pages 317–330, 2009.
6. Vasileios Kontorinis, Liuyi Eric Zhang, Baris Aksanli, Jack Sampson, Houman Homayoun, Eddie Pettis, Dean M Tullsen, and Tajana Simunic Rosing. Managing distributed ups energy for effective power capping in data centers. *ACM SIGARCH Computer Architecture News*, 40(3):488–499, 2012.
7. Miyuru Dayarathna, Yonggang Wen, and Rui Fan. Data center energy consumption modeling: A survey. *IEEE Communications surveys & tutorials*, 18(1):732–794, 2015.
8. John Wilkes. Yet more Google compute cluster trace data. Google research blog, April 2020. Posted at `https://ai.googleblog.com/2020/04/yet-more-google-compute-cluster-trace.html`.
9. John Wilkes. Google cluster-usage traces v3. Technical report, Google Inc., Mountain View, CA, USA, April 2020. Posted at `https://github.com/google/cluster-data/blob/master/ClusterData2019.md`.
10. Magnus Perninge and Robert Eriksson. Frequency control in power systems based on a regulating market. *IEEE Transactions on Control Systems Technology*, 26(1):27–37, 2017.
11. Xiaodong Wang and José F Martínez. Rebudget: Trading off efficiency vs. fairness in market-based multicore resource allocation via runtime budget reassignment. *ACM SIGPLAN Notices*, 51(4):19–32, 2016.
12. Mohammad A Islam, Xiaoqi Ren, Shaolei Ren, and Adam Wierman. A spot capacity market to increase power infrastructure utilization in multi-tenant data centers. In *Proceedings of the 2017 ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems*, pages 19–20, 2017.
13. Xiaofeng Hou, Luoyao Hao, Chao Li, Quan Chen, Wenli Zheng, and Minyi Guo. Power grab in aggressively provisioned data centers: What is the risk and what can be done about it. In *2018 IEEE 36th International Conference on Computer Design (ICCD)*, pages 26–34. IEEE, 2018.
14. Mohammad A Islam, Xiaoqi Ren, Shaolei Ren, Adam Wierman, and Xiaorui Wang. A market approach for handling power emergencies in multi-tenant data center. In *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 432–443. IEEE, 2016.
15. Ali Jahanshahi, Nanpeng Yu, and Daniel Wong. Powermorph: Qos-aware server power reshaping for data center regulation service. *ACM Transactions on Architecture and Code Optimization (TACO)*, 19(3):1–27, 2022.
16. Leonardo Piga, Iyswarya Narayanan, Aditya Sundarrajan, Matt Skach, Qingyuan Deng, Biswadip Maity, Manoj Chakkaravarthy, Alison Huang, Abhishek Dhanotia, and Parth Malani. Expanding datacenter capacity with dvfs boosting: A safe and scalable deployment experience. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, pages 150–165, 2024.
17. Xiaofeng Hou, Mingyu Liang, Chao Li, Wenli Zheng, Quan Chen, and Minyi Guo. When power oversubscription meets traffic flood attack: Re-thinking data center peak load management. In *Proceedings of the 48th International Conference on Parallel Processing*, pages 1–10, 2019.
18. Parminder Singh, Pooja Gupta, and Kiran Jyoti. Tasm: technocrat arima and svr model for workload prediction of web applications in cloud. *Cluster Computing*, 22(2):619–633, 2019.
19. Jing Bi, Libo Zhang, Haitao Yuan, and MengChu Zhou. Hybrid task prediction based on wavelet decomposition and arima model in cloud data center. In *2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC)*, pages 1–6. IEEE, 2018.

20. Kim Hazelwood, Sarah Bird, David Brooks, Soumith Chintala, Utku Diril, Dmytro Dzhulgakov, Mohamed Fawzy, Bill Jia, Yangqing Jia, Aditya Kalro, et al. Applied machine learning at facebook: A datacenter infrastructure perspective. In *HPCA*, pages 620–629. IEEE, 2018.
21. Grant Ayers, Jung Ho Ahn, Christos Kozyrakis, and Parthasarathy Ranganathan. Memory hierarchy for web search. In *HPCA*, pages 643–656. IEEE, 2018.
22. Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. Power provisioning for a warehouse-sized computer. *ACM SIGARCH computer architecture news*, 35(2):13–23, 2007.
23. Chang-Hong Hsu, Qingyuan Deng, Jason Mars, and Lingjia Tang. Smoothoperator: Reducing power fragmentation and improving power utilization in large-scale datacenters. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 535–548, 2018.
24. Guosai Wang, Shuhao Wang, Bing Luo, Weisong Shi, Yinghang Zhu, Wenjun Yang, Dianming Hu, Longbo Huang, Xin Jin, and Wei Xu. Increasing large-scale data center capacity by statistical power control. In *Proceedings of the Eleventh European Conference on Computer Systems*, pages 1–15, 2016.
25. Jiuchen Shi, Jiawen Wang, Kaihua Fu, Quan Chen, Deze Zeng, and Minyi Guo. Qos-awareness of microservices with excessive loads via inter-datacenter scheduling. In *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 324–334. IEEE, 2022.
26. Jiuchen Shi, Hang Zhang, Zhixin Tong, Quan Chen, Kaihua Fu, and Minyi Guo. Nodens: Enabling resource efficient and fast {QoS} recovery of dynamic microservice applications in datacenters. In *2023 USENIX Annual Technical Conference (USENIX ATC 23)*, pages 403–417, 2023.
27. Rodrigo N Calheiros, Enayat Masoumi, Rajiv Ranjan, and Rajkumar Buyya. Workload prediction using arima model and its impact on cloud applications' qos. *IEEE transactions on cloud computing*, 3(4):449–458, 2014.
28. Sheng Di, Derrick Kondo, and Walfredo Cirne. Host load prediction in a google compute cloud with a bayesian model. In *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pages 1–11. IEEE, 2012.
29. Farid Benhammadi, Zahia Gessoum, Aicha Mokhtari, et al. Cpu load prediction using neuro-fuzzy and bayesian inferences. *Neurocomputing*, 74(10):1606–1616, 2011.
30. Jianhuang Liang, Jian Cao, Jungang Wang, and Yuxia Xu. Long-term cpu load prediction. In *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, pages 23–26. IEEE, 2011.
31. Ji Xue, Feng Yan, Robert Birke, Lydia Y Chen, Thomas Scherer, and Evgenia Smirni. Practise: Robust prediction of data center time series. In *2015 11th International Conference on Network and Service Management (CNSM)*, pages 126–134. IEEE, 2015.
32. Weishan Zhang, Pengcheng Duan, Laurence T Yang, Feng Xia, Zhongwei Li, Qinghua Lu, Wenjuan Gong, and Su Yang. Resource requests prediction in the cloud computing environment with a deep belief network. *Software: Practice and Experience*, 47(3):473–488, 2017.
33. Svilen Kanev, Juan Pablo Darago, Kim Hazelwood, Parthasarathy Ranganathan, Tipp Moseley, Gu-Yeon Wei, and David Brooks. Profiling a warehouse-scale computer. *ACM SIGARCH Computer Architecture News*, 2016.
34. Alibaba. Alibaba open trace. Technical report, Alibaba Inc., 2022. Posted at `https://github.com/alibaba/clusterdata/blob/master/cluster-trace-microarchitecture-v2022/README.md`.
35. Shutian Luo, Huanle Xu, Chengzhi Lu, Kejiang Ye, Guoyao Xu, Liping Zhang, Yu Ding, Jian He, and Chengzhong Xu. Characterizing microservice dependency and performance: Alibaba trace analysis. In *Proceedings of the ACM Symposium on Cloud Computing*, pages 412–426, 2021.
36. Suzanne Rivoire, Parthasarathy Ranganathan, and Christos Kozyrakis. A comparison of high-level full-system power models. *HotPower*, 8(2):32–39, 2008.
37. Weiwei Lin, Tianhao Yu, Chongzhi Gao, Fagui Liu, Tengyue Li, Simon Fong, and Yongxiang Wang. A hardware-aware cpu power measurement based on the power-exponent function model for cloud servers. *Information Sciences*, 547:1045–1065, 2021.
38. Shimon Aronhime, Claudia Calcagno, Guido H Jajamovich, Hadrien Arezki Dyvorne, Philip Robson, Douglas Dieterich, M Isabel Fiel, Valérie Martel-Laferriere, Manjil Chatterji, Henry Rusinek, et al. Dce-mri of the liver: effect of linear and nonlinear conversions on hepatic perfusion quantification and reproducibility. *Journal of Magnetic Resonance Imaging*, 40(1):90–98, 2014.
39. George F Reed, Freyja Lynn, and Bruce D Meade. Use of coefficient of variation in assessing variability of quantitative assays. *Clinical and Vaccine Immunology*, 9(6):1235–1239, 2002.
40. Rumpa Dasgupta, Amin Sakzad, and Carsten Rudolph. Cyber attacks in transactive energy market-based microgrid systems. *Energies*, 14(4):1137, 2021.
41. Fayiz Alfaverh, Mouloud Denai, and Yichuang Sun. A dynamic peer-to-peer electricity market model for a community microgrid with price-based demand response. *IEEE Transactions on Smart Grid*, 2023.
42. Xiaowen Huang, Shimin Gong, Jingmin Yang, Wenjie Zhang, Liwei Yang, and Chai Kiat Yeo. Hybrid market-based resources allocation in mobile edge computing systems under stochastic information. *Future Generation Computer Systems*, 127:80–91, 2022.
43. Che Chen, Shimin Gong, Wenjie Zhang, Yifeng Zheng, and Yeo Chai Kiat. Drl-based contract incentive for wireless-powered and uav-assisted backscattering mec system. *IEEE Transactions on Cloud Computing*, 2024.
44. Philip Sedgwick. Pearson's correlation coefficient. *Bmj*, 345, 2012.
45. Leann Myers and Maria J Sirois. Spearman correlation coefficients, differences between. *Wiley StatsRef: Statistics Reference Online*, 2014.
46. Hervé Abdi. The kendall rank correlation coefficient. *Encyclopedia of measurement and statistics*, 2:508–510, 2007.
47. Gregory P Meyer. An alternative probabilistic interpretation of the huber loss. In *Proceedings of the ieee/cvf conference on computer vision*

and pattern recognition, pages 5261–5269, 2021.

48. Muhammad Tirmazi, Adam Barker, Nan Deng, Md E Haque, Zhijing Gene Qin, Steven Hand, Mor Harchol-Balter, and John Wilkes. Borg: the next generation. In *Proceedings of the fifteenth European conference on computer systems*, pages 1–14, 2020.