Supply Framework of Physical Machine Demand in Elastic Computing Service

Zhanyu Liu[1†], Xudong Zhang[2], Zhidong Hu[2], Xiejing Li[2], Fei Peng[2], Jian Zhou[2], Siyu Deng[2], and Guanjie Zheng[1](✉)

[1] Shanghai Jiao Tong University, China
{zhyliu00,gjzheng}@sjtu.edu.cn
[2] Alibaba Cloud Computing, China
{yulan.zxd,huzhidong.hzd,xiejing.lxj,xinyou.pf,
xuming,desy.dsy}@alibaba-inc.com

Abstract. In the context of Elastic Computing Service (ECS), ensuring an adequate supply of physical machines to meet the varying computing demands is crucial for sustaining high performance and low cost. In industrial practices, different from the typical resource allocation problem that allocates the computing demand into servers, provision of physical servers is a supply chain problem that predicts the future demand for physical machines based on forecasts derived from historical vCPU usage and potential future customer needs, particularly for those customers with high demand. This provision process encompasses three main stages: customer text demand analysis, future demand forecasting, and the allocation of physical servers. However, each stage presents specific challenges. Firstly, large demands from customers are often ambiguously expressed. Secondly, the forecasting process is complicated to model due to the scarce, spiky, and ambiguous nature of the data. Thirdly, the conversion of forecasted vCPU demand into actual physical server quantities is inefficient and ineffective. To address these issues, we propose a novel framework for physical server provisioning. Initially, client requests are aggregated and processed using Large Language Model to extract Potential Future Demand (PFD). Subsequently, future vCPU demand is predicted based on PFD data through a specialized forecasting model tailored with PFD-specific optimizations. Finally, physical machine allocation is executed employing a hierarchical bin-packing algorithm enhanced by heuristic selection and integer programming. Extensive experiments demonstrate the effectiveness and efficiency of the proposed framework with over 60% accuracy improvement and 90% fragment reduction on average compared with the baselines. This framework has been applied to the real industrial scenario of Alibaba Cloud.

Keywords: time series forecasting, elastic computing service, cloud computing

†: Work was done during Alibaba Cloud Computing intership
✉: Corresponding Author

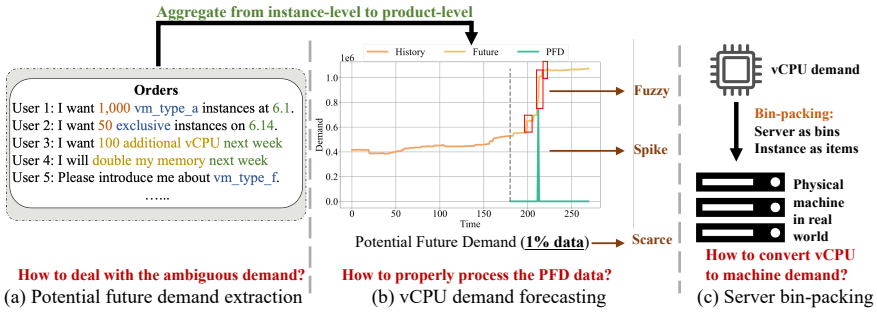

Fig. 1. The challenges in physical machine demand supplement task in elastic computing service.

1 Introduction

In Elastic Computing Service (ECS), provisioning physical machines to meet client computing demands is a critical supply-chain challenge [2, 8, 42, 19] distinct from traditional resource allocation. Unlike static server distribution, this requires forecasting future infrastructure needs using historical vCPU usage data and proactive customer demand signals, including support tickets where clients explicitly request supply solutions for anticipated large-scale requirements or report immediate shortages. These client-initiated communications serve as critical inputs for procurement planning, enabling timely equipment orders and deployment scheduling to preempt gaps between supply and demand.

The process of supplying the physical machine in the cloud service consists of three stages in the industrial practice, as shown in Fig. 1. Firstly, the clients, especially those with big demand, would request resources ahead of time in the Orders. The requests will be aggregated and processed to potential future demand (PFD) focusing on virtual CPU (vCPU) demand. Furthermore, the future vCPU demand will be predicted based on the historical vCPU demand and PFD data. Finally, the physical machine supply allocation is executed which aims to use physical machines with minimum cost to satisfy the predicted vCPU demand. The generated physical machine could offer a reasonable reference for the downstream procurement department.

However, the task of supplementing physical machines in real-world industrial applications is highly complex. As illustrated in Fig. 1, this challenge involves several pivotal factors: (1) managing ambiguous large demand requests from the users, (2) effectively processing PFD data to accurately forecast the future vCPU demand, and (3) optimizing physical machine allocation strategies to reduce overall costs. In particular, during the first stage, client-provided requests are often vague and unstructured, making it challenging to directly extract precise data points, such as dates, exact vCPU requirements, and the quantity of instances needed. Moreover, the second stage involves difficulties in integrating PFD data with predictions concerning future vCPU demand. This complexity arises due to the characteristics of the PFD data since it is non-zero on only 1% of data samples (indicating scarcity), the non-zero values tend to be large (indi-

cating sparsity), and there is often a discrepancy between real demand and the PFD data (indicating fuzziness). Finally, a gap exists between vCPU demand and actual physical machine provisioning that translating the demand into a concrete number of servers while minimizing costs is a complex task.

In this paper, we develop a deployed comprehensive framework aimed at systematically forecasting and processing future vCPU and physical machine demand while addressing the identified challenges. Our framework initially employs LLM-enhanced techniques to accurately interpret and structure requests provided by clients. Specifically, we utilize the in-context learning ability of LLM to enhance the understanding of client text inputs. Furthermore, we introduce a specialized time series forecasting model tailored for PFD-based vCPU prediction. This model includes multiple modules to manage the challenges of scarcity, sparsity, and fuzziness in vCPU demand prediction respectively. Finally, physical machine demand is determined using a hierarchical bin-packing algorithm to meet the forecasted vCPU requirements. Our contributions could be summarized as follows:

- To the best of our knowledge, we are the first to systematically address the physical machine demand supplement task in industrial cloud computing supply-chain applications.
- We propose a supply framework of physical machine demand that seamlessly integrates algorithms from LLM, time series forecasting, and bin-packing. This framework effectively addresses challenges related to ambiguous client requests, complex PFD data, and efficient server provisioning.
- We conduct extensive experiments on real-world industrial datasets of Alibaba Cloud. The results validate the effectiveness of our framework and significant performance boost in demand prediction and resource allocation.

2 Related Work

2.1 Demand Forecasting

Demand forecasting is essential not only in cloud computing but also in a variety of other domains [1, 24, 32]. In industrial applications, effective forecasting could enhance resource management and operational efficiency across various scenarios. For instance, [43] employs an LSTM-based method [12] to forecast the number of tourists in different countries. Similarly, [9] utilizes ARIMA and linear models to predict company sales. Moreover, [33] compares statistical and machine learning methods for daily demand forecasting. However, there is a noticeable gap in methods tailored for demand forecasting in cloud computing, where PFD data significantly impacts the demand curve. In the academic realm, time series forecasting has been extensively studied due to its pivotal role in various downstream applications such as traffic management [22, 26, 27] and energy management [13, 18], where its implementation is often facilitated through the application of graph modeling [16, 17, 25]. Numerous methods have been developed to enhance time series forecasting, each focusing on different aspects of data

modeling [23, 7]. Some approaches leverage transformer-based architectures [34] to capture cross-channel and temporal dependencies. Notable examples include Autoformer [38], Pyraformer [20], Informer [45], FEDFormer [46], and iTransformer [21]. Recently, [40] evaluated the performance of these transformer-based methods and concluded that linear methods could achieve greater effectiveness and efficiency. Consequently, linear-based models and convolution-based models have demonstrated superior performance, such as PatchTST [29], TimesNet [37], Mamba [10], CMamba [41] and TimeMixer [36]. However, these academic methods are typically evaluated on datasets related to energy, weather, and traffic, which exhibit high periodicity, stable trends, and similar data distributions across different nodes. This contrasts sharply with real industrial data related to vCPU and server demand, which are characterized by high volatility and are significantly influenced by PFD data.

2.2 Resource Allocation

Resource allocation is a fundamental component of cloud computing, primarily focused on the efficient distribution of computational resources to satisfy user demands while minimizing operational expenses [19, 28]. Traditional mechanisms for resource allocation often rely on heuristic methods such as First-Fit, Best-Fit, and Worst-Fit algorithms [4]. In addition to heuristic methods, some approaches employ price-based greedy algorithms, which are applicable in scenarios like cloud gaming, where resource allocation decisions are influenced by pricing strategies [5]. Other methods involve prediction-based algorithms that dynamically adjust resource allocation based on predicted future demands [39]. Optimization techniques such as integer programming have also been extensively utilized for resource allocation and scheduling to achieve optimal results [11, 14, 31]. In recent years, deep reinforcement learning (DRL) methods have emerged as promising solutions for the resource allocation problem [3, 15, 44]. Nevertheless, the integration of accurate demand forecasting with resource allocation remains an underexplored area. While deep learning and optimization methods have made significant strides in addressing resource allocation challenges, their effectiveness can be further enhanced by incorporating precise demand forecasts.

3 Preliminary

Definition 1 (Physical Machine). *A **Physical Machine** is a hardware machine in a data center that provides computational resources, including CPU, memory, storage, and network bandwidth.*

Definition 2 (Virtual CPU). *A **Virtual CPU** (vCPU) represents a portion of physical CPU resources allocated to a virtual machine (VM). Each vCPU is scheduled and managed by the hypervisor or virtualization engine. In the industrial application, we focus on demand forecasting for product-level vCPU to guarantee the robustness of the prediction result.*

Definition 3 (Potential Future Demand). ***Potential Future Demand*** *(PFD) describes the requirement for computational resources based on the mined order data. PFD is critical for effective demand forecasting, helping to ensure that supply meets demand while minimizing costs and optimizing performance.*

Definition 4 (Instance, Product, Order). *An* ***Instance*** *is a single deployment of a virtual machine configured according to specific customer requirements. A* ***Product*** *includes a range of cloud service instances with varying computational ability, as well as consistent configurations of other services such as networking and physical machines. An* ***Order*** *refers to a formalized communication, typically text-based, between the customer and customer support, aimed at obtaining additional information or making requests regarding specific products. In this paper, we aim to forecast vCPU demand at the product level to ensure the robustness of the results and subsequently divide this forecast to the instance level for optimized downstream physical machine demand allocation.*

Problem 1 (Demand Forecasting). Given T-day historical vCPU demand data $\mathbf{X}^{t-T+1:t}$ and auxiliary information of the PFD data $\mathbf{P}^{t+1:t+T'}$ extracted from the orders, the aim of demand forecasting is to learn a model $f(\cdot)$ to forecast future T' days vCPU demand data $\mathbf{X}^{t+1:t+T'}$. Formally, it could be formulated:

$$[\mathbf{X}^{t-T+1}, \cdots, \mathbf{X}^{t}, \mathbf{P}^{t+1:t+T'}] \xrightarrow{f(\cdot)} [\mathbf{X}^{t+1}, \cdots, \mathbf{X}^{t+T'}].$$

Problem 2 (Physical Machine Allocation). Basically, this problem is a vector bin-packing problem (VBP). Given the forecasted vCPU demand data $\mathbf{X}^{t+1:t+T'}$ obtained from the demand forecasting model $f(\cdot)$, the objective of physical machine allocation is to optimally assign these CPU demands of product-level of certain day $\mathbf{X}^{T'}$ to the instances of different type $[\mathbf{I}_0, \mathbf{I}_1, \cdots, \mathbf{I}_n]$, where each instance has its size of CPU c_i and memory m_i. The physical machine allocation problem is converted to a bin-packing problem that instances $[\mathbf{I}_0, \mathbf{I}_1, \cdots, \mathbf{I}_n]$ serve as items and physical machines $[\mathbf{S}_0, \mathbf{S}_1, \cdots, \mathbf{S}_m]$ serve as bins. Each physical machine $\mathcal{S}_i$ has its cpu capacity $\bar{c}_i$, memory capacity $\bar{m}_i$, and cost $\bar{o}_i$. The bin-packing problem can be formulated as $\min \sum_{j=1}^{m} \bar{o}_j y_j$, which is subject to

$$\sum_{i=1}^{n} c_i a_{ij} \leq \bar{c}_j y_j, \sum_{i=1}^{n} m_i a_{ij} \leq \bar{m}_j y_j, \sum_{j=1}^{m} a_{ij} = 1, \forall i \in \{1, \ldots, n\}, \forall j \in \{1, \ldots, m\},$$

where a_{ij} is a binary variable indicating whether instance i is assigned to server j ($a_{ij} = 1$) or not ($a_{ij} = 0$) and y_j is a binary variable indicating whether server j is used ($y_j = 1$) or not ($y_j = 0$).

4 Method

In this section, we detail our comprehensive framework for provisioning physical machines within the elastic computing service, as shown in Fig. 2. Firstly, the Potential Future Demand (PFD) Extraction Module leverages LLM to extract

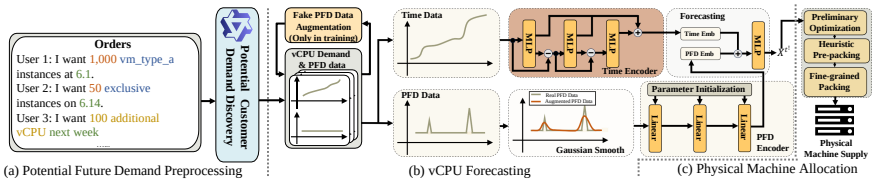

Fig. 2. Illustration of our framework for physical machine supplement.

potential vCPU future demand derived from incoming orders of large demand. Subsequently, the vCPU Forecasting Module utilizes separate modeling techniques for historical time series data and PFD data to accurately forecast future vCPU demand. Finally, the Physical Machine Allocation Module employs a multi-stage approach to ensure efficient resource allocation.

4.1 PFD Preprocessing Module

Goal: The primary challenge in extracting future demand lies in the ambiguous nature of the textual input provided by the users with large demand, as depicted in Fig. 2(a). To address this, the Potential Future Demand (PFD) Extraction Module employs the ability of in-context learning (ICL) of LLM to convert unstructured text into structured entity tuples. This structured PFD data can then be seamlessly integrated into the downstream vCPU forecasting module, facilitating accurate and reliable demand prediction.

Input & Output The PFD Preprocessing Module processes the textual reports derived from orders. Example texts of these inputs are provided in Fig. 2(a). This module translates these ambiguous requests by Qwen2.5-7B [3] LLM into clear and structured data such as (UserID, Quantity, InstanceType, Date) to enhance the accuracy of the forecasting process.

Potential Customer Demand Discovery To interpret the requests from the users, we propose the ICL-enhanced Potential Customer Demand Discovery module, as shown in Tab 1. The module integrates structured system configurations with contextual learning to extract structured demand tuples (UserID, Quantity, InstanceType, Date) from heterogeneous inputs, including user information and requests. First, it integrates **system configurations**, such as the current date and ECS knowledge base defining VM specifications, to ensure temporal relevance and technical grounding. Then, it formalizes a **structured extraction task** targeting (UserID, Quantity, InstanceType, Date) tuples, explicitly bridging user context (e.g., existing instances) and natural language requests. Subsequently, **in-context examples** demonstrate schema alignment that colloquial expressions like "6.1" are mapped to standardized formats by leveraging

[3] https://github.com/QwenLM/Qwen2.5

Table 1. ICL-enhanced prompt of Potential Customer Demand Discovery.

Potential Customer Demand Discovery
[System Config] Current Date: {Weekday}, {YYYY-MM-DD} ECS Knowledge Base: {vm_type_a: this VM contains 4 vcpu and ...} **[Task]** Extract (UserID, Quantity, InstanceType, Date) tuples from: 1. User Context → [Existing Resources/Contracts] 2. Current Request → [Natural Language] **[In-Context Examples]** UserID: User1. Context: {User_Instances: None, ...} Request: "I want 1,000 vm_type_a instances at 6.1" Output: (User1, 1000, vm_type_a, 2024.6.1) ..... **[New Request]** UserID: {UserX}, Context: {User_Summary}, Request: {Natural Language Text}

the system configuration, while VM type definitions from the knowledge base disambiguate technical terms. Finally, the **new request** combines summarized user profiles with raw textual queries, enabling joint reasoning to infer the coming user request. This hierarchical design supports scalable adaptation to updated VM types or policies through the knowledge base, ensuring industrial-grade robustness in demand extraction.

4.2 vCPU Forecasting Module

Goal: As shown in Fig. 1 (b), the main challenges in forecasting the vCPU demand encompass several critical aspects: (1) the PFD data is **scarce**, with only 1% of the dates containing non-zero PFD data; (2) most of the non-zero PFD data consists of **spikes**, characterized by sudden, large values; (3) the PFD demand data is not consistently aligned with the actual demand, as customers often provide **fuzzy** future demand requests and fail to purchase instances punctually. Consequently, in this subsection, we propose three modules to address these challenges separately.

Input & Output The vCPU Forecasting Module takes two inputs: the T-day historical vCPU demand data $\mathbf{X}^{t-T+1:t}$ and the future T'-day PFD data $\mathbf{P}^{t+1:t+T'}$. The output generated by this module is the forecasted vCPU demand for the future T' days, denoted as $\mathbf{X}^{t+1:t+T'}$.

Fake PFD Augmentation To mitigate the **scarcity** of the PFD data in training, we propose the Fake PFD Augmentation Module, as shown in Fig. 3. This module aims to enhance the volume and variability of the training data, improving the robustness of the forecasting models. The intuition behind this module

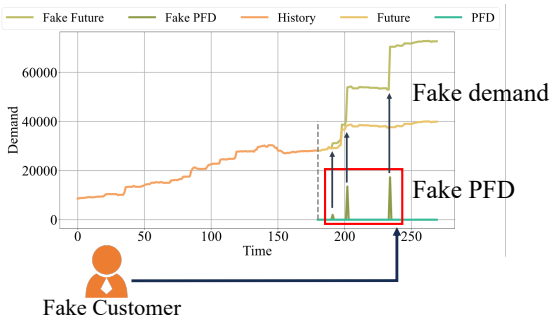

Fig. 3. Illustration of Fake PFD Data Augmentation

is based on the observation that if the customers request a specific amount of vCPU for a future date, they are likely to purchase a similar amount around that time. Initially, we randomly select a subset of the existing training samples. For each selected sample, we generate a fake PFD dataset, denoted as $\mathbf{P}'$, which includes a small, random number of randomly distributed non-zero entries. These non-zero values are crafted to replicate the characteristics observed in real PFD data, such as spikes and variability in magnitude. Subsequently, we incorporate this synthetic PFD data $\mathbf{P}'$ with the corresponding future vCPU demand data $\mathbf{X}$ of the selected samples by adding the synthetic PFD data with a random scaling factor. This integration process aims to simulate realistic demand patterns, thereby allowing the model to generalize more effectively and adapt to the sparsity and irregularities that are typical of real-world PFD data.

Gaussian Smoothing Augmentation The **spiked** PFD data could harm the performance of the downstream prediction model since the spike values may distort the learning process and lead to overfitting or inaccurate predictions. To alleviate the impact of spiked data in the forecasting process, we introduce the Gaussian Smoothing Augmentation module. This method aims to smooth the spike in the PFD data and keep the information of the vCPU demand, thus helping the models to generate stable output and learn robust knowledge, as shown in Fig. 2(b). Specifically, each spiked value in $\mathbf{P}^k$ is converted into a Gaussian distribution and aggregated to form $\hat{\mathbf{P}}$, as represented in Eq. 1.

$$\hat{\mathbf{P}}^t = \sum_{k=1}^{T'} \mathbf{P}^k \frac{1}{\sigma_1\sqrt{2\pi}} e^{-\frac{(t-k)^2}{2\sigma_1^2}} \tag{1}$$

Gaussian Center Parameter Initialization In practice, the PFD data is **fuzzy** and often does not precisely align with the actual demand, as customers may purchase instances approximately around the projected date rather than on the exact date, as depicted in Fig. 1(b). Consequently, the PFD data is highly correlated with the actual demand, exhibiting temporal locality where the PFD

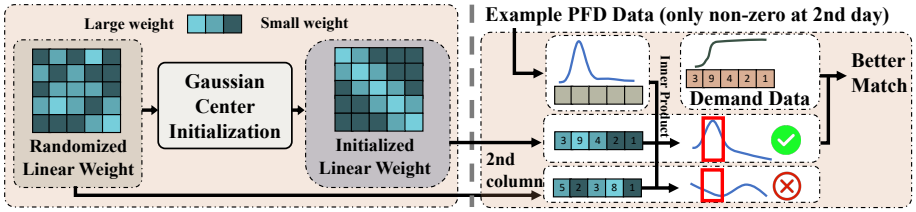

Fig. 4. Gaussian Center Initialization. (a) This panel depicts the initialized linear parameter matrix. (b) This panel provides a practical example on a PFD data with only potential demand on the 2nd day. It shows that the Gaussian Center Initialized parameter of the 2nd column has highest weight on the 2nd slot, thereby enhancing the demand of the 2nd day and generate accurate embedding.

at day t ($\hat{\mathbf{P}}^t$) is significantly related to the actual demand on nearby dates. To effectively utilize the PFD data for forecasting demand, we incorporate this temporal locality into the initialization of the model, thereby leveraging prior knowledge to enhance the training process, as shown in Fig.4. For a linear model W that encodes the PFD data, we can express this relationship:

$$\hat{\mathbf{P}}^t_{out} = \sum_{i=1}^{T'} \hat{\mathbf{P}}^i W_{it}, \tag{2}$$

where $\hat{\mathbf{P}}^t_{out}$ is the output data of position t. Given the temporal locality of the PFD data, the parameter matrix should exhibit the following characteristic:

$$W^{tt} \geq W^{it} \quad \forall i \in \{1, 2, \ldots, T'\}. \tag{3}$$

Accordingly, we initialize the parameter matrix using a column-wise Gaussian distribution. By denoting the t-th column of the parameter matrix as w^t, this initialization can be described as:

$$w^t \sim \mathcal{N}(t, \sigma_2). \tag{4}$$

Forecasting To model the temporal data, we utilize a forecast-backcast model based on N-BEATS [30] to generate time embeddings. In each layer of this model, a Multi-Layer Perceptron (MLP) is used to process the input data H_i (where $H_0 = \mathbf{X}^{t-T+1:t}$), producing two types of embeddings: forecast embedding fo_i and backcast embedding ba_i at the i-th layer.

$$fo_i, ba_i = MLP(H_i) \tag{5}$$

The input to the subsequent layer is then computed as the difference between H_i and ba_i and the final time embedding is obtained by summing the forecast embeddings across all layers:

$$H_{i+1} = H_i - ba_i, \quad H_{time} = \sum_i fo_i. \tag{6}$$

Finally, the time embedding H_{time} is combined with the PFD embedding H_{PFD}, and a final MLP layer is used to forecast the future vCPU demand:

$$\hat{\mathbf{X}}^{t+1:t+T'} = MLP(H_{time} + H_{PFD}). \tag{7}$$

For the training of the forecasting model, we use the Mean Square Error (MSE) as the loss function.

$$\mathcal{L} = \sum_{i=t+1}^{t+T'} (\hat{\mathbf{X}}^i - \mathbf{X}^i)^2 \tag{8}$$

4.3 Physical Server Allocation Module

Goal: In the previous subsection, we get the predicted future vCPU demand. Nevertheless, to supply this demand, the downstream purchasing department needs the actual physical machine demand and there is a gap between the vCPU demand and the physical machine demand. Consequently, the Physical Server Allocation Module is designed to bridge this gap by mapping the predicted vCPU demand to tangible physical server demand.

Input & Output The input of this module is the predicted future vCPU demand $\mathbf{X}^{t+1:t+T'}$. In practice, we use the vCPU demand of the final day $\mathbf{X}^{t+T'}$ as the target demand and convert it into the instances of different type $[\mathbf{I}_0, \mathbf{I}_1, \cdots, \mathbf{I}_n]$ by the selling ratio of different instances to get a robust estimation. The output of this module is the usage of physical machines, including the machine list $[\mathbf{S}_0, \mathbf{S}_1, \cdots, \mathbf{S}_m]$ and indicator $y_{1:m}$.

Preliminary Optimization In this subsection, we first utilize the instance demand $[\mathbf{I}_0, \cdots, \mathbf{I}_n]$ and the physical server types $[\mathbf{R}_0, \cdots, \mathbf{R}_k]$ to get a initial physical server lists $[\mathbf{S}_0, \cdots, \mathbf{S}_m]$. The primary constraints ensure that the CPU and memory capacities of the physical servers meet the requirements of the instances. This is formulated as:

$$\sum_{i=1}^{n} c_i \leq \sum_{i=1}^{k} \hat{c}_i N_i \quad \sum_{i=1}^{n} m_i \leq \sum_{i=1}^{k} \hat{m}_i N_i, \tag{9}$$

where $\hat{c}_i$, $\hat{m}_i$, and N_i represent the CPU, memory, and the number of physical server types $\mathbf{R}_i$. Additionally, we incorporate a constraint on the fragment space ratio to ensure that the available fragmented space can ideally accommodate an instance. This is expressed as:

$$\frac{\sum_{i=1}^{k} \hat{m}_i N_i - \sum_{i=1}^{n} m_i}{\sum_{i=1}^{k} \hat{c}_i N_i - \sum_{i=1}^{n} c_i} \geq r, \tag{10}$$

where r is the mem/CPU ratio, which is set to 2. By solving this optimization problem using cvxpy [6], we could get the initial quantities of each physical machine N_i and subsequently convert it to the physical machine list $[\mathbf{S}_0, \cdots, \mathbf{S}_m]$.

Heuristic Pre-packing To alleviate the redundancy of the intermediate result, we formalize the problem to a two-dimensional bin-packing problem. However, directly optimize the problem with integer programming is highly costly since the quantity of instance is too large. Consequently, we propose to first use a Heuristic Pre-packing based on a bin-centric best-fit algorithm to reduce the scale of the problem. We sort the physical server and pack the server with a larger capacity first. To pack the server, we compute the cosine similarity between its remaining capacity and the demand of each instance in both memory and CPU dimensions and repeatedly. Then, we pack the instance with the highest cosine similarity to the server. By iteratively applying this heuristic packing method, we can significantly reduce the scale of the problem and achieve an effective, scalable solution to the bin-packing problem.

Fine-grained Packing Following the Heuristic Pre-packing phase, the set of physical servers $[\mathbf{S}_0, \cdots, \mathbf{S}_m]$ may contain some residual fragmentation and redundancy (about 5% of the total servers on average). Consequently, we process the remaining servers and instances by the integer-programming-based Fine-grained Packing module to enhance the efficiency. Firstly, **Index-Independent Optimization** eliminates instance and server indices, focusing on their quantities. Let n and m be the number of instances and servers, and p and q the number of instance and server types, with $p \ll n$ and $m \ll q$. Directly packing instances into servers has a complexity of $O(n^m)$, which is impractical. The Index-Independent Optimization reduces this to $O(p^q)$, simplifying the problem. Then, the **Instance Type Merging** technique merges instances with similar CPU and memory capacities, thus reducing the number of instance types from p to p', where $p' < p$. This further lowers the time complexity of the packing process from $O(n^m)$ to $O(p'^q)$. Finally, we utilize cvxpy [6] to optimize this problem to get the vector $y_{1:m}$ that indicates whether each physical server is used.

5 Experiment

5.1 Experiment Setting

The framework is evaluated in the real vCPU demand data for 31 different products from Alibaba Cloud, spanning the period from January 2018 to April 2023. The experiment comprises two parts. All of the experiments were conducted on an NVIDIA A10 GPU.

– In vCPU Forecasting, we follow the setting of previous time series forecasting research on benchmark datasets [35, 46] that segment the dataset with continuous sliding windows of 270 days and utilize historical vCPU data of 180 days to forecast the future vCPU data of subsequent 90 days. We search the hyper-parameters σ_1 and σ_2 within the set $\{1, 3, 7\}$, and synthesize fake PFD data using 10% of the training set of the original dataset. Additionally, within the Fake PFD Augmentation module, we randomly allocate up to three days

Table 2. Overall performance of the vCPU () demand forecasting task on the real data of Alibaba Cloud from January 1, 2018, to April 1, 2023. The mean and standard deviation of the results in five runs are shown. In each column, the best result is highlighted in bold and grey. All numbers are in units of 1k.

	Day 7			Day 30			Day 90		
	RMSE	MAE	MAPE(%)	RMSE	MAE	MAPE(%)	RMSE	MAE	MAPE(%)
Autoformer	147.53±14.98	104.25±19.93	81.82±24.64	171.34±8.52	114.70±14.62	82.36±24.84	245.91±20.69	155.71±17.30	89.29±16.12
Pyraformer	546.63±60.09	154.84±17.64	21.54±7.09	504.12±33.92	159.87±27.98	23.93±8.35	483.47±26.35	179.57±10.78	40.87±2.04
Informer	403.77±38.04	135.09±11.50	36.78±8.65	382.19±24.68	131.01±7.86	39.16±8.30	383.61±23.33	151.86±12.45	48.80±4.39
FEDformer	69.50±12.26	46.83±7.56	31.28±4.15	97.09±7.32	58.53±7.93	34.26±6.96	181.78±19.52	83.41±11.56	32.46±7.70
MICN	57.53±5.52	25.46±2.33	9.22±2.47	88.66±2.84	37.31±2.55	12.56±2.74	166.54±6.32	73.83±3.86	28.74±6.68
iTransformer	47.37±3.18	20.48±1.53	6.95±0.33	89.12±4.17	36.24±1.49	10.46±0.38	169.69±2.30	68.54±1.63	20.46±0.75
PatchTST	62.11±19.65	31.30±10.88	9.87±3.67	95.52±8.65	45.39±2.95	13.75±0.35	209.67±23.55	93.44±13.21	27.37±4.91
Mamba	105.80±4.47	51.60±2.65	16.82±1.10	149.29±7.21	73.79±4.11	25.43±1.35	367.35±132.28	134.68±39.77	46.92±27.06
NLinear	26.71±4.04	15.24±6.28	11.27±9.58	64.32±3.10	29.50±6.56	13.28±10.22	158.84±2.00	70.07±3.31	24.04±7.14
Ours	**21.79±0.53**	**8.09±0.39**	**2.62±0.43**	**58.16±0.93**	**22.35±0.33**	**6.66±0.25**	**148.10±1.48**	**57.30±0.63**	**17.02±0.29**

of fake customer demand to each sample. We use Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE) as the metrics.

$$RMSE = \sqrt{\frac{1}{T'}\sum_{i=1}^{T'}(\mathbf{X}^i - \hat{\mathbf{X}}^i)^2}, MAE = \frac{1}{T'}\sum_{i=1}^{T'}|\mathbf{X}^i - \hat{\mathbf{X}}^i|, MAPE = \frac{1}{T'}\sum_{i=1}^{T'}|\frac{\mathbf{X}^i - \hat{\mathbf{X}}^i}{\mathbf{X}^i}| \tag{11}$$

– In Physical Server Allocation, we utilize the predicted vCPU as the input. In practical industrial scenarios, the demands for different products necessitate allocation to distinct types of physical servers. To assess the efficacy and efficiency of the proposed allocation method, we use the predicted vCPU requirements for the one with largest demand in the 31 products, assigning its demands to three specific types of physical servers accordingly.

5.2 vCPU Forecasting Overall Performance

Baseline We select nine well-acknowledged baselines to evaluate the performance of our proposed module on the vCPU forecasting task. We utilize the hyperparameter of the original papers for the baseline models. Moreover, these baselines are fed with the PFD embedding processed by a three-layer MLP.

– Transformer-based methods: These methods employ the Transformer [34] as the backbone model, incorporating specific optimizations tailored for time series analysis, including adjustments for trend and seasonal patterns. The comparison encompasses models such as **Autoformer [38]**, **Pyraformer [20]**, **Informer [45]**, **FEDformer [46]**.
– Linear-based methods: These methods utilize the linear network or convolutional neural network (CNN), including **MICN [35]**, **PatchTST [29]**, **NLinear [40]**, **Mamba [10]**. Note that when the input is single-channel, the **iTransformer [21]** is reduced to the linear model.

Table 3. Ablation Study of vCPU demand forecasting. The mean and standard deviation of the results in five runs are shown. In each column, the best result is highlighted in bold and grey. All numbers are in units of 1k.

	Day 7			Day 30			Day 90		
	RMSE	MAE	MAPE(%)	RMSE	MAE	MAPE(%)	RMSE	MAE	MAPE(%)
Base	24.56±1.01	9.83±0.25	3.44±0.14	77.44±4.66	31.01±1.31	7.76±0.50	214.75±11.72	84.72±4.87	21.29±1.39
Base+P	23.54±0.37	8.81±0.18	3.09±0.10	66.36±3.05	25.52±1.61	6.91±0.68	177.24±3.02	66.35±1.56	18.97±1.30
Base+P+F	22.79±0.39	8.51±0.13	2.83±0.05	61.15±3.47	23.64±1.37	6.75±0.48	161.42±8.47	62.04±3.84	18.29±2.45
Base+P+F+A	**21.79±0.53**	**8.09±0.39**	**2.62±0.43**	**58.16±0.93**	**22.35±0.33**	**6.66±0.25**	**148.10±1.48**	**57.30±0.63**	**17.02±0.29**

Overall Performance We report the forecasting accuracy of the vCPU demand on days 7, 30, and 90 in Table 2. From the analysis of the table, several key observations emerge: (1) The proposed framework consistently outperforms state-of-the-art (SOTA) baselines across all experimental settings. (2) In comparison to transformer-based methods, our framework shows a remarkable improvement: 67.74% in RMSE, 74.99% in MAE, and 79.72% in MAPE. This suggests that the transformer-based approaches do not align well with the characteristics of vCPU demand data, likely due to their assumption of a pronounced trend and seasonal patterns. The observed superior performance signifies that our straightforward model effectively captures the underlying patterns in vCPU demand data, learning robust knowledge from them. (3) When compared with linear-based methods, our framework exhibits performance enhancements of 38.48% in RMSE, 47.35% in MAE, and 54.35% in MAPE. Notably, linear-based methods outperform transformer-based methods, consistent with the findings of [40], where linear models are noted for their robustness in time series forecasting. Moreover, the substantial performance gains underscore the efficacy of the specialized PFD data modeling approach of our framework.

5.3 vCPU Forecasting ablation study

In this section, we will verify the effectiveness of the proposed modules in the vCPU Forecasting framework. We define **Base** as a straightforward MLP model that utilizes raw vCPU time series and PFD data. The enhanced model, **Base+P**, incorporates Gaussian Center Parameter Initialization. Further optimizations are defined as **Base+P+F**, which includes Fake PFD Augmentation, and **Base+P+F+A**, which integrates Gaussian Smoothing Augmentation. The result is shown in Table 3. From the table, we can observe that (1) The integration of Gaussian Center Parameter Initialization (in the **Base+P** model) results in a substantial improvement in performance. This enhancement underscores the significance of embedding prior knowledge, which facilitates the model in acquiring more effective and robust representations. (2) The incremental additions of Fake PFD Augmentation and Gaussian Smoothing Augmentation (in the **Base+P+F** and **Base+P+F+A** models, respectively) contribute further to the accuracy and reliability of the model. These results collectively demonstrate that each augmentation not only enhances the predictive capability of

Table 4. The baseline comparison of Physical Server Allocation module. The mean and standard deviation are reported. In each column, the best result is highlighted in bold and grey.

Methods	1000 Instances (n = 1000)			5000 Instances (n = 5000)			10000 Instances (n = 10000)		
	Packing Density(%)	Fragment (%)	Solving Time (s)	Packing Density(%)	Fragment (%)	Solving Time (s)	Packing Density(%)	Fragment (%)	Solving Time (s)
First-Fit-B	66.74±7.37	32.73±7.04	1.86±1.90	65.35±5.65	34.54±5.61	6.59±1.79	73.58±4.82	26.36±4.77	**12.49±0.87**
First-Fit-I	96.56±1.45	2.52±1.30	**1.05±0.52**	96.52±1.40	3.39±1.44	**6.24±1.22**	97.84±0.83	2.03±0.74	14.52±4.24
Best-Fit-B	97.31±1.30	1.68±0.98	2.63±1.09	98.06±0.26	1.23±0.60	19.51±2.86	97.42±1.54	2.22±1.98	46.78±9.84
Best-Fit-I	84.89±10.61	4.30±6.67	12.51±12.64	87.36±14.73	2.30±4.22	191.75±37.13	91.66±6.73	5.05±4.46	387.85±33.40
IP	98.71±1.48	0.49±0.57	120.00±0.00	88.96±8.36	5.29±4.22	120.00±0.00	89.09±3.02	5.13±1.54	120.00±0.00
Ours	**99.40±0.74**	**0.05±0.12**	11.31±3.09	**99.63±0.70**	**0.06±0.11**	23.66±3.18	**99.54±0.73**	**0.05±0.08**	43.59±10.37

the MLP but also reinforces the stability and generalization of the forecasting framework.

5.4 Physical Server Allocation Result

In this section, we will verify the efficacy and efficiency of the proposed Physical Server Allocation module in different problem scales by employing several performance metrics:

- Packing Density: Calculated as $\frac{1}{2}(\frac{\text{Demand\_CPU}}{\text{Supplied\_CPU}} + \frac{\text{Demand\_Mem}}{\text{Supplied\_Mem}})$.
- Fragment: The remaining space ratio that cannot accommodate any instance.
- Solving Time: The average time required to solve each forecasted result.

We compare our module with several baseline methods including both instance-centric and server-centric algorithms. Specifically, we consider the First-Fit algorithm, which assigns the current instance to the first available server (-I) or places the first available instance onto the current server (-S); the Best-Fit algorithm, which allocates the instance to a server (-I) or the server to an instance (-S) based on the highest cosine similarity in resource capacity; and an Integer Programming (IP) approach that directly formulates the instance-server assignment as a linear programming problem, with a 120-second time constraint imposed to mitigate computational complexity while maintaining solution feasibility. The results summarized in Table 4 lead to two key observations: (1) our proposed module outperforms the baseline methods significantly while maintaining an acceptable computational cost, and (2) the Packing Density and Fragment metrics remain robust and stable across various problem scales, thereby substantially reducing costs.

6 Conclusion

In this work, we introduce a novel framework for provisioning physical machines in elastic computing services. The framework incorporates a PFD Preprocessing module to structure PFD data from text inputs, a vCPU Forecasting Module to precisely predict future vCPU requirements by modeling the PFD data, and a Physical Server Allocation Module to efficiently translate these vCPU demands

into actual physical server allocations. In the future, we will aim at further optimize the efficiency and efficacy of the framework.

Acknowledgment

This work was sponsored by National Key Research and Development Program of China under Grant No.2022YFB3904204, National Natural Science Foundation of China under Grant No. 62102246, 62272301, and Provincial Key Research and Development Program of Zhejiang under Grant No. 2021C01034. This work was supported by Alibaba Research Intern Program.

References

1. Abolghasemi, M., Beh, E., Tarr, G., Gerlach, R.: Demand forecasting in supply chain: The impact of demand volatility in the presence of promotion. Computers & Industrial Engineering **142**, 106380 (2020)
2. Al-Dhuraibi, Y., Paraiso, F., Djarallah, N., Merle, P.: Elasticity in cloud computing: state of the art and research challenges. IEEE Transactions on services computing **11**(2), 430–447 (2017)
3. Chen, Z., Hu, J., Min, G., Luo, C., El-Ghazawi, T.: Adaptive and efficient resource allocation in cloud datacenters using actor-critic deep reinforcement learning. IEEE Transactions on Parallel and Distributed Systems **33**(8), 1911–1923 (2021)
4. Coffman, E.G., Csirik, J., Galambos, G., Martello, S., Vigo, D., et al.: Bin packing approximation algorithms: Survey and classification. In: Handbook of combinatorial optimization, pp. 455–531. Springer (2013)
5. Deng, Y., Li, Y., Tang, X., Cai, W.: Server allocation for multiplayer cloud gaming. In: Proceedings of the 24th ACM international conference on Multimedia. pp. 918–927 (2016)
6. Diamond, S., Boyd, S.: CVXPY: A Python-embedded modeling language for convex optimization. Journal of Machine Learning Research **17**(83), 1–5 (2016)
7. Ding, J., Liu, Z., Zheng, G., Jin, H., Kong, L.: Condtsf: one-line plugin of dataset condensation for time series forecasting. arXiv preprint arXiv:2406.02131 (2024)
8. Dong, H., Wang, B., Qiao, B., Xing, W., Luo, C., Qin, S., Lin, Q., Zhang, D., Virdi, G., Moscibroda, T.: Predictive job scheduling under uncertain constraints in cloud computing. In: IJCAI. pp. 3627–3634 (2021)
9. Feizabadi, J.: Machine learning demand forecasting and supply chain performance. International Journal of Logistics Research and Applications **25**(2), 119–142 (2022)
10. Gu, A., Dao, T.: Mamba: Linear-time sequence modeling with selective state spaces. arXiv preprint arXiv:2312.00752 (2023)
11. Heinz, S., Ku, W.Y., Beck, J.C.: Recent improvements using constraint integer programming for resource allocation and scheduling. In: Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems: 10th International Conference, CPAIOR 2013, Yorktown Heights, NY, USA, May 18-22, 2013. Proceedings 10. pp. 12–27. Springer (2013)
12. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)

13. Hong, T., Pinson, P., Wang, Y., Weron, R., Yang, D., Zareipour, H.: Energy forecasting: A review and outlook. IEEE Open Access Journal of Power and Energy **7**, 376–388 (2020)
14. Ibrahim, H., Aburukba, R.O., El-Fakih, K.: An integer linear programming model and adaptive genetic algorithm approach to minimize energy consumption of cloud computing data centers. Computers & Electrical Engineering **67**, 551–565 (2018)
15. Islam, M.T., Karunasekera, S., Buyya, R.: Performance and cost-efficient spark job scheduling based on deep reinforcement learning in cloud computing environments. IEEE Transactions on Parallel and Distributed Systems **33**(7), 1695–1710 (2021)
16. Kang, Z., Liu, Z., Pan, S., Tian, L.: Fine-grained attributed graph clustering. In: Proceedings of the 2022 SIAM International Conference on Data Mining (SDM). pp. 370–378. SIAM (2022)
17. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
18. Le Cam, M., Daoud, A., Zmeureanu, R.: Forecasting electric demand of supply fan using data mining techniques. Energy **101**, 541–557 (2016)
19. Li, F.: Cloud-native database systems at alibaba: Opportunities and challenges. Proceedings of the VLDB Endowment **12**(12), 2263–2272 (2019)
20. Liu, S., Yu, H., Liao, C., Li, J., Lin, W., Liu, A.X., Dustdar, S.: Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In: International conference on learning representations (2021)
21. Liu, Y., Hu, T., Zhang, H., Wu, H., Wang, S., Ma, L., Long, M.: itransformer: Inverted transformers are effective for time series forecasting. arXiv preprint arXiv:2310.06625 (2023)
22. Liu, Z., Ding, J., Zheng, G.: Frequency enhanced pre-training for cross-city few-shot traffic forecasting. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 35–52. Springer (2024)
23. Liu, Z., Hao, K., Zheng, G., Yu, Y.: Dataset condensation for time series classification via dual domain matching. In: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. pp. 1980–1991 (2024)
24. Liu, Z., Liang, C., Zheng, G., Wei, H.: Fdti: fine-grained deep traffic inference with roadnet-enriched graph. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 174–191. Springer (2023)
25. Liu, Z., Zeng, C., Zheng, G.: Graph data condensation via self-expressive graph structure reconstruction. In: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. pp. 1992–2002 (2024)
26. Liu, Z., Zheng, G., Yu, Y.: Cross-city few-shot traffic forecasting via traffic pattern bank. In: Proceedings of the 32nd ACM International Conference on Information and Knowledge Management. pp. 1451–1460 (2023)
27. Liu, Z., Zheng, G., Yu, Y.: Multi-scale traffic pattern bank for cross-city few-shot traffic forecasting. ACM Transactions on Knowledge Discovery from Data **19**(4), 1–24 (2025)
28. Nahir, A., Orda, A., Raz, D.: Resource allocation and management in cloud computing. In: 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM). pp. 1078–1084. IEEE (2015)
29. Nie, Y., Nguyen, N.H., Sinthong, P., Kalagnanam, J.: A time series is worth 64 words: Long-term forecasting with transformers. arXiv preprint arXiv:2211.14730 (2022)
30. Oreshkin, B.N., Carpov, D., Chapados, N., Bengio, Y.: N-beats: Neural basis expansion analysis for interpretable time series forecasting. arXiv preprint arXiv:1905.10437 (2019)

31. Rezvani, M., Akbari, M.K., Javadi, B.: Resource allocation in cloud computing environments based on integer linear programming. The Computer Journal **58**(2), 300–314 (2015)
32. Seyedan, M., Mafakheri, F.: Predictive big data analytics for supply chain demand forecasting: methods, applications, and research opportunities. Journal of Big Data **7**(1), 53 (2020)
33. Spiliotis, E., Makridakis, S., Semenoglou, A.A., Assimakopoulos, V.: Comparison of statistical and machine learning methods for daily sku demand forecasting. Operational Research **22**(3), 3037–3061 (2022)
34. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
35. Wang, H., Peng, J., Huang, F., Wang, J., Chen, J., Xiao, Y.: Micn: Multi-scale local and global context modeling for long-term series forecasting. In: The eleventh international conference on learning representations (2023)
36. Wang, S., Wu, H., Shi, X., Hu, T., Luo, H., Ma, L., Zhang, J.Y., Zhou, J.: Timemixer: Decomposable multiscale mixing for time series forecasting. arXiv preprint arXiv:2405.14616 (2024)
37. Wu, H., Hu, T., Liu, Y., Zhou, H., Wang, J., Long, M.: Timesnet: Temporal 2d-variation modeling for general time series analysis. In: The eleventh international conference on learning representations (2022)
38. Wu, H., Xu, J., Wang, J., Long, M.: Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. Advances in neural information processing systems **34**, 22419–22430 (2021)
39. Xiao, Z., Song, W., Chen, Q.: Dynamic resource allocation using virtual machines for cloud computing environment. IEEE transactions on parallel and distributed systems **24**(6), 1107–1117 (2012)
40. Zeng, A., Chen, M., Zhang, L., Xu, Q.: Are transformers effective for time series forecasting? In: Proceedings of the AAAI conference on artificial intelligence. vol. 37, pp. 11121–11128 (2023)
41. Zeng, C., Liu, Z., Zheng, G., Kong, L.: Cmamba: Channel correlation enhanced state space models for multivariate time series forecasting. arXiv preprint arXiv:2406.05316 (2024)
42. Zhang, G., Ravishankar, M.: Exploring vendor capabilities in the cloud environment: A case study of alibaba cloud computing. Information & Management **56**(3), 343–355 (2019)
43. Zhang, Y., Li, G., Muskat, B., Law, R.: Tourism demand forecasting: A decomposed deep learning approach. Journal of Travel Research **60**(5), 981–997 (2021)
44. Zhang, Y., Yao, J., Guan, H.: Intelligent cloud resource management with deep reinforcement learning. IEEE Cloud Computing **4**(6), 60–69 (2017)
45. Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., Zhang, W.: Informer: Beyond efficient transformer for long sequence time-series forecasting. In: Proceedings of the AAAI conference on artificial intelligence. vol. 35, pp. 11106–11115 (2021)
46. Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., Jin, R.: Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In: International conference on machine learning. pp. 27268–27286. PMLR (2022)