Spatial-Temporal Heterogenous Graph Contrastive Learning for Microservice Workload Prediction

Mohan Gao[1], Kexin Xu[1], Xiaofeng Gao[1]*, Tengwei Cai[2], Haoyuan Ge[2]

[1]MoEKey Lab of Artificial Intelligence, Department of Computer Science and Engineering, Shanghai Jiao Tong University Shanghai, China

[2]Ant Group, Hangzhou, China

gao_mohan@sjtu.edu.cn, irene233@sjtu.edu.cn, gao-xf@cs.sjtu.edu.cn, tengwei.ctw@antgroup.com, gehaoyuan.ghy@antgroup.com

Abstract

With the widely adoption of microservice architecture in the cloud computing industry, accurate prediction of workloads, especially CPU cores, can support reasonable resource allocation, thereby optimizing the resource utilization of the system. However, workload prediction is challenging in two dimensions. In the temporal dimension, workload series 1) has non-stationary characteristics, leading to poor predictability; 2) has a multi-periodic nature with entangled temporal patterns; 3) may be influenced by dynamic system states like response time and number of requests. In the spatial dimension, when regarding microservices as nodes in a distributed system, there is no topology caused by physical connections, but exists complex similarity dependencies. Extracting robust spatial features from these dependencies presents difficulties.

To address these, we propose STEAM, a Spatio Temporal Heterogenous Graph Contrastive Learning for Microservice Workload Prediction. STEAM leverages non-stationary decomposition self-attention to extract temporal features from non-stationary and multi-periodic workload series, while the decoupled embedding is used to capture system state information of microservices. By treating microservices as nodes and constructing a similarity graph, STEAM effectively models the similarity relationships between microservices. To reduce the prior interference caused by the similarity threshold and improve the robustness, STEAM constructs two heterogeneous augmentation views and uses contrastive learning to extract the shared consistent spatial features. The multi-scale learning is adopted to model the long- and short-term temporal features, forming a spatio-temporal stacking structure.

Experiments on two datasets, including MS dataset obtained from Ant Group, which is one of the world's largest cloud service providers, demonstrate the superiority of STEAM.

Introduction

Cloud computing has become widespread across various industries to improve resource flexibility, cost efficiency, and fast deployment of applications (Gan et al. 2021).

Currently, cloud service providers widely adopt microservices architecture (Google 2021; Luo et al. 2022), consisting of many fine-grained, loosely coupled microservices, each with specific responsibilities that do not rely on other components. In such architecture, the main challenge is to improve resource utilization. Previous research has shown that resource waste is mainly caused by the mismatch between resource allocation and actual demand (Wang et al. 2023). Indeed, as a distributed system, the microservices architecture allocates resources in advance based on estimation of future workloads (the workload refers to CPU resources in most scenarios). Therefore, forecasting the workload of microservices has become the key to the efficient resource allocation (Krieger et al. 2017; Abdullah et al. 2022).

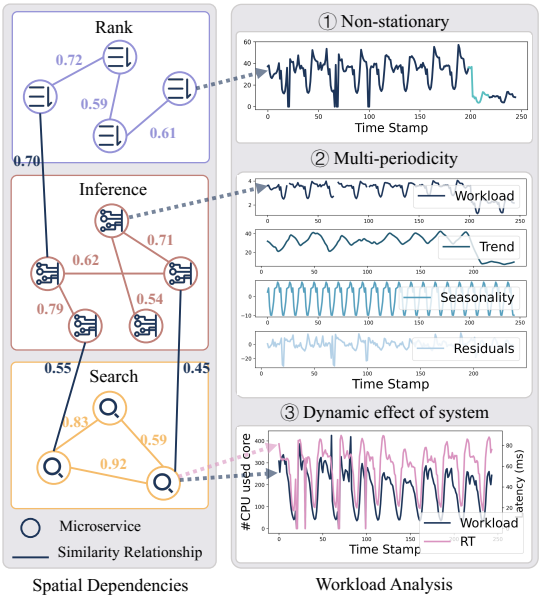


Figure 1: An Instance of Microservice Architecture.

Predicting the future workloads faces challenges. As shown in the right part of Fig. 1, microservice workloads are normally non-stationary with various period patterns. The latest time series forecasting methods leverage the non-stationarity characteristics of the time series (Liu et al. 2022c) or adopt series decomposition (Wu et al. 2021; Zhou et al. 2022) and multi-scale learning (Chen et al. 2024; Liu et al. 2022b) methods to enhance performance. However,

*Xiaofeng Gao is the corresponding author.

previous workload prediction methods (Chen et al. 2023; Hua et al. 2023) did not fully utilize these temporal characteristics. In addition, indicators like response time and CPU utilization rate reflect the current state of the system, and have dynamic and intricate effects on workload.

Moreover, the flexibility of the architecture results in the lack of explicit topological connection between microservices. Due to the frequent invocation of certain microservices by cloud applications, and the existence of inter-cluster communications, there exists similarity relationships between microservices. The previous spatio-temporal workload prediction method (Luo et al. 2024a) constructed similarity graphs of microservices, but did not consider the prior interference introduced by the similarity threshold (Liu et al. 2022a; Li et al. 2022a), resulting in the lack of robustness in the spatial features it extracted.

Therefore, we propose the Spatio Temporal Heterogenous Graph Contrastive Learning for Microservice Workload Prediction (STEAM). First, we adopt the de-stationary self-attention to improve the predictability of the non-stationarity workload series and use series decomposition to extract entangled periodic patterns. Second, we use the Convolutional Neural Network (CNN) to capture the dynamic effect of the system states, the decoupling embedding is adopted to avoid the feature entanglement caused by directly adding the auxiliary information to the value embedding. Third, to capture the spatial dependencies, we build similarity edges between microservices nodes. In order to avoid overfitting and reduce computational complexity, a similarity threshold needs to be adopted, that is, edges are built only when the similarity (like the Pearson coefficient) is larger than a certain threshold. However, a static similarity threshold cannot cope with the dynamic nature of microservices and will introduce prior interference. To this end, we use the workload similarity graph as the backbone and leverage the system states to construct two heterogeneous graphs as augmentation views. Then, augmentation-based contrastive learning is adopted to extract the shared consistent features between two views while alleviating the prior interference. Finally, we apply multi-scale Learning using self-attention to capture long-term temporal features and CNN for short-term ones, forming a spatio-temporal sandwich structure.

Collectively, STEAM encapsulates the insights of microservice workloads, including non-stationarity, multi periodicity, and multi-range temporal dependencies. It also leverages the similarity dependencies between microservices and the dynamic effect of system states to improve the performance. The contributions can be summarized as:

- We reveal the characteristics of microservice workloads through data analysis, thus using non-stationary decomposition self-attention to extract the temporal features of workloads and designing a system states encoder to model the dynamic influence of system states.
- For the first time, heterogeneous graph contrastive learning is adopted to extract spatial dependencies between microservices, improving the robustness of the spatial encoder. At the same time, multi-scale learning is used to capture long- and short-term dependencies, further improving the performance of workload prediction.
- Extensive experiments on real-world datasets show that the proposed STEAM outperforms several state-of-the-art competitors by a prominent margin.

Related Work

Time Series Methods. Modeling the workload as time series and using time series methods for prediction is intuitive and reasonable. Statistical methods like ARIMA (Calheiros et al. 2015) and Prophet (Taylor and Letham 2017) are used for workload forecasting due to their computational efficiency and interpretability. However, these methods are designed for univariate time series forecasting, which cannot capture the relationships between variables and cannot utilize additional information such as system states. Furthermore, these methods often fail to predict future changes in previously unknown patterns (Kim et al. 2016), leading to poor performance on non-stationary data. With the development of the deep learning techniques, recurrent neural networks-based algorithms have been used for workload forecasting (Tan, Klein, and Elmroth 2019; Jayakumar et al. 2020). In addition, some general multivariate time series forecasting methods based on convolutional neural networks (CNN) (Wu et al. 2023) and self-attention mechanisms (Nie et al. 2023; Liu et al. 2022c) can also be used for workload forecasting. These methods are able to utilize temporal dependencies and address multivariate scenarios by mapping variables into high-dimensional space. However, they are unable to leverage structured prior knowledge, such as the similarity relationship of response times between microservices, which limit their performance.

Spatial-Temporal Methods. Spatio-Temporal Graph Neural Networks (STGNN) can capture dependencies between variables by treating them as nodes in a graph and constructing edges based on the calling behaviors (Park et al. 2021; Tam et al. 2023) or similarity (Luo et al. 2023, 2024b). In the microservice workload forecasting scenario, STAMP (Luo et al. 2024a) adopts dynamic multi-graph attention to capture the relationships between workloads and employ cross-view contrastive learning to encode the system states. In other scenarios like traffic flow forecasting, there are also spatio-temporal graph neural network methods: STGCN (Yu, Yin, and Zhu 2018) uses CNN to extract temporal information and GCN to extract spatial information, forming a stacked spatio-temporal structure. ST-SSL (Li et al. 2022b) uses hypergraph cross-view contrastive learning (Wang et al. 2021; Chen et al. 2020) to alleviate the sparsity problem. These methods lack the ability to deal with non-stationary data in the temporal dimension, and the graphs constructed based on similarity threshold introduce noise, which can interfere with the extraction of spatial features.

Data Analysis

In this section, we analyze the 10-day microservice workload data MS, revealing the non-stationarity and multi-periodicity nature of workload series, as well as the dynamic impact of the system states.

Stationarity Analysis

The microservice architecture is highly flexible and scalable, allowing it to handle high concurrency requests, resulting in non-stationary workload series. Since most time series forecasting methods are based on weak stationarity assumptions, this property makes it difficult to accurately predict future workloads. We quantitatively analyzed the stationarity of the workload using the Autocorrelation Function (ACF), which describes the correlation between current and historical values of a series. In Fig. 2, the horizontal axis of each subplot indicates the lag order and the vertical axis indicates the correlation coefficient between the corresponding lagged series and the original series. The shaded area is the confidence interval, the points within it indicate that the correlation is not statistically significant. For stationary series, as the lag order increases, the correlation between the lag series and the original series rapidly decreases, entering the shaded region. The results illustrate the non-stationary characteristics of the workload series.

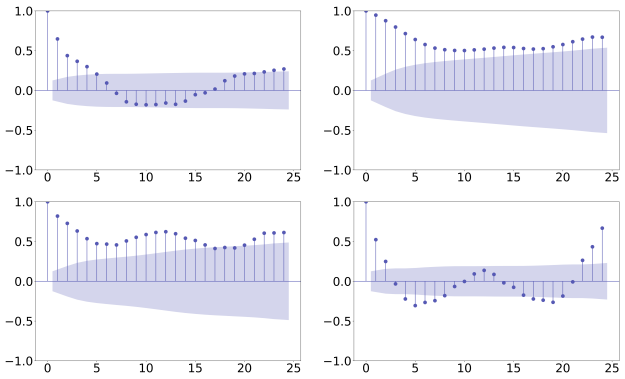


Figure 2: ACF of 4 different microservices.

Periodicity Analysis

In the actual industry scenario, there will be periodic data processing or computing tasks, and the workload is also affected by periodic changes in business requirements and user usage, which leads to multi-periodic characteristics in the workload series. The periodogram depicted in Fig. 3 provides a detailed view of the frequency components for microservices. The horizontal axis of each subplot represents different hourly periods, and the vertical axis represents the power spectral density of the signal at that period, indicating the significance of that period component. It shows that the workloads not only exhibit significant 12-hour and 24-hour cycles, but also displays different periodic characteristics due to the unique pattern of individual business demands. Therefore, Fully leveraging the unique periodic features within microservices is key to accurate forecasting.

Dynamic Effect of System States

The workload generated by microservices is closely related to the current states of the system. Due to the distributed nature of the microservicez architecture, communication between microservices will cause latency, which may become

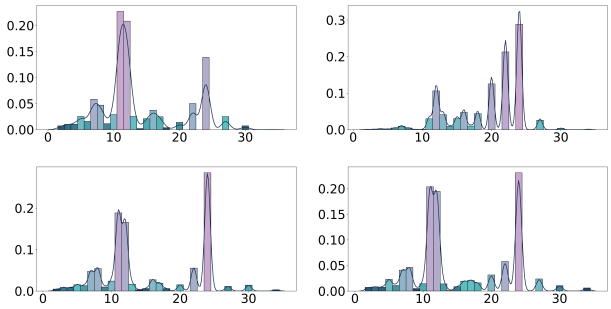


Figure 3: Periodogram of 4 different microservices.

the performance bottleneck for the system. In addition, metrics like the number of replicas and requests serve as indicators reflecting the current operational status of the system, which are also valuable for predictive analysis. As illustrated in the Fig. 4, the horizontal axis of each subgraph represents the time stamp, and the vertical one represents the Pearson C‘oefficient. Due to the flexibility of the microservices architecture, the impact of the system states on the workload is dynamic. Hence, we can utilize the dynamic impact of system states for accurate workload forecasting.

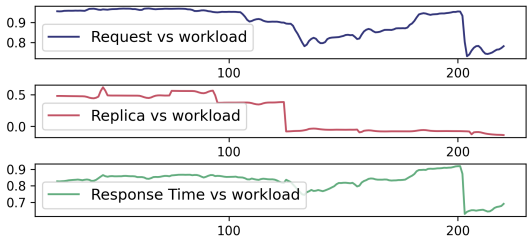


Figure 4: The dynamic Pearson Coefficients of system states and workload with a sliding window of 50 timestamps.

Problem Formulation

In this section, we present the definitions of time-series data in the microservice scenario, and further provide the problem definition of microservice workload prediction.

Definition 1 (Workload Series) *By taking the number of CPU cores $X_{m,t}$ used by microservice m at time t as the workload, we obtain the workload series $X \in \mathbb{R}^{M \times T}$, M is the number of microservices, T is the time stamps.*

Definition 2 (System States Series) *The response time RT, total number of requests Req, and number of replicas Rep are used to represent the system state, thus obtaining the system state series $RT, Req, Rep \in \mathbb{R}^{M \times T}$.*

By recording the historical workload of the microservices and using the system state as auxiliary series, we can make predictions about future workloads.

Definition 3 (Workload Prediction) *Given the workload series X and the system states series $\langle RT, Req, Rep \rangle$, the goal is to predict the workload $\tilde{X} \in \mathbb{R}^{M}$ at time $T+1$.*

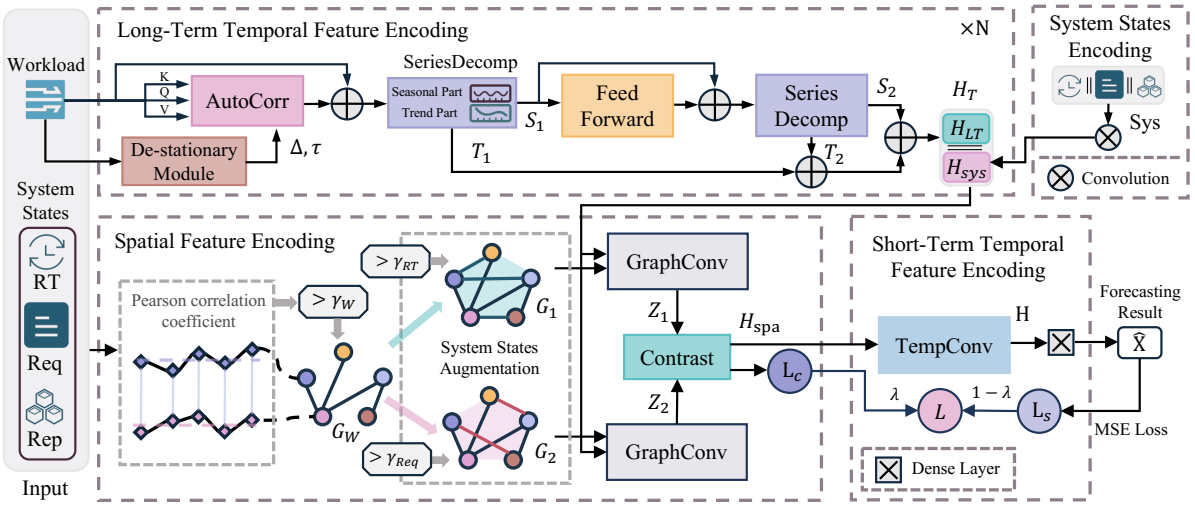


Figure 5: The overall architecture of STEAM.

The Proposed Method: STEAM

The structure of the proposed model is illustrated in Fig. 5. The model consists of four components: (1) Long-Term Temporal Feature Encoding, which extracts long-term dependencies in the workload series through non-stationary decomposition self-attention. (2) System States Encoding, which captures the dynamic impact of system states. (3) Spatial Feature Encoding, which leverages system states to extract robust dynamic spatial dependencies between microservices. (4) Short-Term Temporal Feature Encoding, extracting short-term dependencies and make predictions.

Long-Term Temporal Feature Encoding

The self-attention mechanism learns the correlation between all time stamps by calculating attention weight matrices, leading to attractive ability to capture long-term dependencies in a time series. We have revealed the non-stationary and multi-periodic nature of microservice workload through data analysis. As a result, the de-stationary module and series decomposition are used to combine with self-attention to extract long-term temporal feature in the workload series.

De-Stationary Module Normalization to time series with strong dynamics or weak stationarity can lead to over-stationarization problem (Liu et al. 2022c). To address this, the positive scaling scalar τ and shifting vector Δ are introduced to bring the vanished non-stationary information back to self-attention calculation. By utilizing multi-layer perceptrons to project the mean $\mu_X \in \mathbb{R}^M$ and standard deviation $\sigma_X \in \mathbb{R}^M$ of the workload sequence X, we obtain de-stationary factors $\tau \in \mathbb{R}^+$ and $\Delta \in \mathbb{R}^T$.

$$\log \tau = MLP(\sigma_X, X), \Delta = MLP(\mu_X, X) \quad (1)$$

After projecting the workload series X to obtain query Q, key K, and value V, the normalized autocorrelation coefficient $\hat{R}_{XX}(\Delta t)$ (Wu et al. 2021) are utilized to detect periodic patterns in the workload series, which reflects the time-delay similarity of sequence X at lag Δt. Then, we select the top k periodic patterns based on the coefficient and utilize the Auto-Correlation Mechanism, combined with the de-stationary factors to calculate self-attention.

$$\begin{aligned} \Delta t_1, \ldots, \Delta t_k &= argTopk(\hat{R}_{QK}(\Delta t)), \Delta t \in \{1, \ldots, T\} \\ AutoCorr\,(X) &= \sum_{i=1}^{k} (\tau \cdot \hat{R}_{QK}(\Delta t_i) + \Delta) \cdot Roll(V, \Delta t_i) \end{aligned} \quad (2)$$

where $Roll(V, \Delta t_i)$ represents the operation to X with time delay Δt_i, during which elements that are shifted beyond the first position are re-introduced at the last position.

Decomposition Self-Attention Since each workload has unique periodicities, the season-trend decomposition (Wu et al. 2021) is adopted to decompose the seasonal component and trend-cyclical component in the workload series. With N decomposition layers, the equations of the l-th layer can be formalized as follows:

$$\begin{aligned} S_l^1, T_l^1 &= SeriesDecomp(AutoCorr\left(X^{l-1}\right) + X^{l-1}) \\ S_l^2, T_l^2 &= SeriesDecomp(FeedForward\left(S_l^1\right) + S_l^1) \end{aligned} \quad (3)$$

where $X^l = S_l^2$ denotes the output of l-th decomposition layer, X^0 is the embedding of workload series X.

The Long-term Temporal Feature $H_{LT} \in \mathbb{R}^{M \times T \times d}$ is the sum of three decomposed components i.e., $H_{LT} = S_N^2 + W_N(T_N^1 + T_N^2)$, where $W_N \in \mathbb{R}^{d \times d}$ is the weight matrix of the projector, d is the embedding dimension.

System States Encoding

The system states indicators such as latency, total request, and number of replicas reflect the availability of resources which are closely related to workload. Modeling the dynamic impact of system states on the workload can enhance

insight into the causes of workload fluctuation, thus helping to improve predictive performance. First, we concatenate the system states to obtain the overall system states vector denoted as $sys = RT||Req||Rep \in \mathbb{R}^{M\times T\times 3}$. Next, Conv1d is used to get the system states representation $H_{sys} \in \mathbb{R}^{M\times T\times d}$. To avoid the feature entanglement problem, we adopt the decoupled embedding by concatenating H_{LT} and H_{sys} to obtain the temporal embedding of the workload $H_T \in \mathbb{R}^{M\times T\times 2d}$.

Spatial Feature Encoding

In large-scale cloud systems, dynamic dependencies exist among microservices due to shared business functionalities. However, the flexibility of microservice architecture leads to the absence of explicit topology caused by physical connections. Still, it does have similarity correlations, making it more suitable for building similarity-based graphs based on series similarity. Directly constructing a similarity graph based on historical workloads series faces several challenges: A fully connected graph may lead to overfitting problem, and using a similarity threshold will introduce prior interferences and cannot effectively capture the dynamic dependencies among microservices. To address these, we construct a Workload Similarity Graph based on the workload similarity as the backbone. Then, two dynamic heterogeneous graphs are created as augmentation views based on the similarity of system states. We employ augmentation-based contrastive learning to extract shared consistent spatial features between views. The key idea of contrastive learning is that the representation of the same sample in different views should be invariant. By preserving the common characteristics between views, we can exclude prior interference and obtain robust representations.

Workload Similarity Graph Construction To capture the similarity among microservices, we consider M microservices as nodes and construct a Workload Similarity Graph $G_W = \{V, E_W\}$, where E_W is a edge set built based on the series similarity between microservices on workload series and a similarity threshold γ_W. The edge weights are determined by the similarity coefficient.

System States Augmentation Based on the Workload Similarity Graph G_W, We build the edge set E_{RT} based on the series similarity of response time series and a similarity threshold γ_{RT}. Thus get a dynamic heterogeneous graph as an augmentation view $G_1 = \{V, E_W \cup E_{RT}\}$. Similarly, by considering the similarity of request series and a similarity threshold γ_{Req}, we construct the edge set E_{Req}, and get another augmentation view $G_2 = \{V, E_W \cup E_{Req}\}$.

Graph Contrastive Learning To obtain the spatial feature of the microservices, we apply graph contrastive learning to extract the shared consistent features in two views. Graph Convolutional Networks (GCN) are used to obtain the representations of microservice nodes in the two views $Z^1, Z^1 \in \mathbb{R}^{M\times T\times 2d}$.

$$\begin{aligned} Z^1 &= GraphConv(H_T, G_1) \\ Z^2 &= GraphConv(H_T, G_2) \end{aligned} \tag{4}$$

After obtaining the representations z_i^1 and z_i^2 of the microservice i under two views respectively, we adopt two MLPs to map them to the same space for calculating the contrastive loss:

$$\begin{aligned} q_i^1 &= MLP(z_i^1) \\ q_i^2 &= MLP(z_i^2) \end{aligned} \tag{5}$$

the contrastive loss is defined as:

$$\begin{aligned} L_c = \frac{-1}{|V|}\sum_{i\in V}[\log \frac{e^{sim(q_i^1, q_i^2)/\zeta}}{\sum_{j\epsilon V} e^{sim(q_i^1, q_j^2)/\zeta}} \\ + \log \frac{e^{sim(q_i^1, q_i^2)/\zeta}}{\sum_{j\epsilon V} e^{sim(q_j^1, q_i^2)/\zeta}}] \end{aligned} \tag{6}$$

where $sim(u, v)$ denotes the cosine similarity between two vectors u and v. ζ is the temperature parameter. The spatial features of microservice $H_{spa} \in \mathbb{R}^{M\times T\times 2d}$ is obtained by adding q^1 and q^2.

Short-Term Temporal Feature Encoding

Convolutional neural network excels at capturing short-term dependencies due to its characteristic of local response. After obtaining the representations H_{spa} that include the long-term dependencies and spatial features of the microservices, we use a temporal convolutional network to extract short-term dependencies, forming a multi-scale spatiotemporal stacked structure. (7) describes this process, where $H \in \mathbb{R}^{M\times 2d}$ is the final microservice representation, $TempConv$ denotes the convolution operation on the temporal dimension T. Leaky ReLU is used as the activation function in the temporal convolution.

$$H = TempConv(H_{spa}) \tag{7}$$

Finally, we utilize dense layers to transform H and obtain the final used prediction result $\tilde{\boldsymbol{X}} \in \mathbb{R}^M$.

We optimize the model parameters by combining the prediction error with the contrastive loss L_c , which forms the final loss function L:

$$\boldsymbol{L} = \lambda L_s + (1-\lambda)L_c. \tag{8}$$

where $L_s = \|\tilde{X} - X_{T+1}\|^2$ is the prediction errors. X_{T+1} is the real value of workload. λ is the weight coefficient to balance the contributions of different terms.

Experiments

Experiment Setup

Datasets Two real-world cloud computing workload datasets, including the exclusive microservice workload dataset, are used to validate the performance of STEAM. **MS** contains workload information for 46 microservices over a 10-day period. **Ali** [1] is sourced from the Alibaba Cluster Trace, it contains workload records for 148 machines over 8 days.

In the data preprocessing, any outlier values below 0 are set to 0. The datasets are divided into training and testing sets in a 7:1 ratio along the time dimension, with the last 24 hours of the training set used as the validation set.

[1] https://github.com/alibaba/clusterdata/tree/v2018

Datasets	Model	MAE ↓				RMSE ↓			
		$L=12$	$L=24$	$L=36$	$L=48$	$L=12$	$L=24$	$L=36$	$L=48$
MS	ARIMA	0.1028	0.1003	0.0954	0.0965	0.1549	0.1507	0.1452	0.1417
	Prophet	0.2868	0.2720	0.2203	0.0806	0.3715	0.3468	0.2847	0.1182
	STGCN	0.1460	0.1452	0.1459	0.1794	0.2132	0.2140	0.2123	0.2364
	MTGNN	**0.0819**	0.0802	0.0784	0.0810	0.1235	0.1152	0.1078	0.1175
	ST-SSL	0.2097	0.2113	0.2069	0.2083	0.2645	0.2503	0.2539	0.2536
	DLinear	0.1937	0.1587	0.1694	0.1667	0.2461	0.2054	0.2179	0.2131
	FEDformer	0.1289	0.1229	0.1315	0.1424	0.1736	0.1647	0.1762	0.1856
	N-Transformer	0.0980	0.1053	0.1147	0.1168	0.1385	0.1535	0.1616	0.1659
	PatchTST	0.0928	0.0829	0.0803	0.0788	0.1350	0.1066	0.1063	0.1037
	Pyraformer	0.1196	0.1208	0.1208	0.1207	0.1686	0.1665	0.1686	0.1683
	STEAM	0.0846	**0.0767**	**0.0770**	**0.0738**	**0.1154**	**0.1019**	**0.1037**	**0.1013**
Ali	ARIMA	0.1701	0.1659	0.1631	0.2610	0.2248	0.2157	0.2134	0.3509
	Prophet	0.2463	0.2470	0.2447	0.2437	0.3141	0.3121	0.3059	0.3034
	STGCN	0.2018	0.2026	0.2047	0.1985	0.2661	0.2631	0.2667	0.2620
	MTGNN	0.1598	0.1611	0.1614	0.1618	0.2021	0.2065	0.2051	0.2077
	ST-SSL	0.2123	0.1955	0.1900	0.1899	0.2517	0.2375	0.2373	0.2361
	DLinear	0.1840	0.2048	0.2085	0.1989	0.2387	0.2609	0.2646	0.2629
	FEDformer	0.1880	0.2070	0.2091	0.2040	0.2387	0.2600	0.2630	0.2625
	N-Transformer	0.1744	0.1838	0.1936	0.1885	0.2268	0.2381	0.2500	0.2439
	PatchTST	0.1563	0.1584	0.1582	0.1612	0.2025	0.2040	0.2025	0.2051
	Pyraformer	0.2077	0.2046	0.2018	0.1998	0.2639	0.2604	0.2583	0.2554
	STEAM	**0.1454**	**0.1357**	**0.1391**	**0.1348**	**0.1840**	**0.1746**	**0.1775**	**0.1747**

Table 1: Performance of STEAM and baselines. The best results are in boldface.

Baselines We use several state-of-the-art models to validate the performance of STEAM, including statistical methods: ARIMA, a classical time series forecasting method. Prophet (Taylor and Letham 2017), an additive model based on time series decomposition. Time series methods: FEDformer (Zhou et al. 2022), a frequency-enhanced decomposition transformer for long-term series forecasting. Dlinear (Zeng et al. 2023), a linear model with the trend-remainder decomposition. Pyraformer (Liu et al. 2022b), a multi-scale forecasting method based on pyramid attention. PatchTST (Nie et al. 2023), a channel-independence transformer with patching operation. N-Transformer (Liu et al. 2022c), a transformer framework that can gain predictability of non-stationary series. Spatio-temporal methods: STGCN (Yu, Yin, and Zhu 2018), a spatio-temporal stacked model that utilizes CNN to extract temporal features and GCN to extract spatial features separately. MTGNN (Wu et al. 2020) an adaptive graph learning method that captures spatio-temporal features using GCNs. ST-SSL (Ji et al. 2023), a self-supervised spatio-temporal forecasting model based on adaptive graph augmentation.

Evaluation Metrics We use two widely used metrics in time-series forecasting, Root Mean Square Error (RMSE) and Mean Absolute Error (MAE), to evaluate the forecasting accuracy of STEAM and other models. Lower MAE and RMSE scores indicate better forecasting performance.

Parameter Settings All models are implemented using PyTorch and trained using the Adam optimizer on a NVIDIA A10 GPU. The hyperparameters for STEAM are set as follows: the learning rate is 0.001, the batch size is 128. The dimension of representation d is 16. All similarity thresholds are set to the value of the top 10% Pearson coefficients. The kernel size for the temporal convolutions is set to 3, with two convolutional layers. The temperature parameter ζ is set to 0.5. The weight coefficient λ is set to 0.9.

Comparative Results

Table 1 presents the comparative results between STEAM and baselines. the following observations can be made:

STEAM outperforms statistical methods, time-series forecasting methods, and spatial-temporal forecasting methods in workloads forecasting. Statistical methods are based on strong mathematical assumptions, have higher requirements for the weak stationarity or predictability of time series, and cannot effectively deal with the non-stationary characteristics of microservice workloads. Time-series forecasting methods can improve the quality of extracted temporal features through non-stationarization and sequence decomposition, but these methods cannot utilize structured relationships between variables, nor can they utilize the dynamic impact of system states on workload. Although Patchtst achieved good performance by using patching mechanism, it still does not achieve the best results.

In spatial-temporal forecasting methods, STGCN only uses CNN to extract short-term temporal features, lacking the characterization of long-term ones. In the spatial dimension, it uses fully connected similarity graphs and similarity thresholds, introducing prior interference and reducing the quality of spatial feature extraction. MTGNN is an adaptive graph learning method designed for multivariate time series forecasting, the spatial features it extracted lack accuracy and dynamism. ST-SSL uses augmentation-based contrastive learning, but flow- and topology-level enhancement are more suitable for scenarios with clear topological structures, such as traffic flow prediction, which is unsuitable for

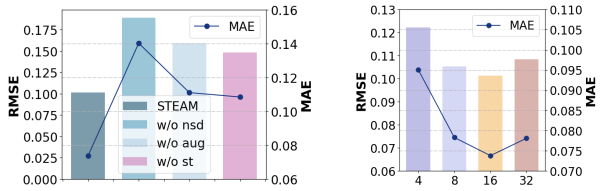
microservice workload forecasting.

What is more, we observe that STEAM has a performance advantage on the Ali dataset, which has weaker temporal regularity of the workloads, making it difficult for other methods to capture its temporal patterns, while STEAM can effectively handle this complex non-stationary data with multi-periodic patterns.

In summary, the performance advantage of STEAM can be attributed to: (1) Using non-stationary decomposition self-attention to improve the quality of extracting long-term temporal features, targeting the non-stationarity and multi-periodic nature of microservice workloads. (2) Adopting system state augmentation based on workload similarity graph and leveraging heterogeneous graph contrastive learning to extract robust spatial features. (3) Further enhancing model performance through multi-scale learning and decoupled embedding of system states.

Ablation Study

In this section, we conduct ablation studies to validate key designs in STEAM. The results are shown in Fig. 6a.



(a) Ablation results. (b) Performance w.r.t d.

Figure 6: Ablation and parameter sensitivity results of STEAM on MS dataset, with the input length $L = 48$.

Effect of the Non-Stationary Decomposition Mechanism. We designed a variant w/o nsd, which uses the transformer's encoder as the long-term temporal feature encoder. Compared to STEAM, the long-term temporal dependency extracting ablity is significantly reduced, which indicates the importance of the de-stationary module to improve the temporal predictability, and the rationality of using sequence decomposition to cope with multi-periodic time series.

Effect of the System States Augmentation. The variant w/o aug only uses GCN to extract information from the workload similarity graph with similarity threshold γ_W as spatial features, the corresponding model loss function being MSELoss. This variant reduces the model's ability to exclude the prior interference and extract the robust spatial features, validating the effectiveness of the system states augmentation and the contrative learning.

Effect of the Short-Term Temporal Feature Encoding. The variant w/o st removes the CNN-based short-term temporal feature encoder, and directly uses H_{spa} to predict the future workload. Adjacent time stamps are often more valuable for predicting the future series. However, without CNN to extract short-term dependencies, the overall performance of the model decreased, demonstrating the effectiveness of using multi-scale learning.

	$\lambda = 0.5$		$\lambda = 0.7$		$\lambda = 0.9$	
	MAE	RMSE	MAE	RMSE	MAE	RMSE
$L = 12$	**0.0789**	**0.1071**	0.0812	0.1101	0.0846	0.1154
$L = 24$	0.0846	0.1129	**0.0715**	**0.0959**	0.0767	0.1019
$L = 36$	0.0891	0.1204	0.0852	0.1225	**0.0770**	**0.1037**
$L = 48$	0.0990	0.1305	0.0979	0.1307	**0.0738**	**0.1013**

Table 2: Performance w.r.t λ.

Parameter Sensitivity

In this section, we investigate the sensitivity of STEAM to some hyper-parameters. We conduct the sensitivity results on MS dataset, with the input length $L = 48$.

Embedding Dimension. The embedding dimension d will affect the performance of STEAM. As depicted in Fig. 6b, the model with too small dimensions (d<8) lacks the spatio-temporal representation ability, increasing the embedding dimension size offers benefits to prediction performance. When the representation dimension is too large, the performance will decrease due to the feature sparsity issue. The optimal representation dimension is around 16.

Weight Coefficient Analysis. The weight coefficient λ adjusts the proportion of the supervised and self-supervised components in the loss function, having different optimal values for different input lengths. It can be observed in Tab. 2 that when the input length L is large, the MSE loss can effectively guide the model in predicting workload according to the abundant historical information, so a large lambda value with a greater proportion of L_s is more advantageous. Conversely, with a smaller input length, historical information is limited and the noise accounts for a larger proportion. In this case, contrastive learning can reduce noise by preserving the same information between two views and provides additional self-supervised information to assist in model training, so the small lambda value has better performance because the contrast loss function accounts for a larger proportion. Note that when $L = 12$, STEAM achieved the best results compared to other baselines in Table 1 when $\lambda = 0.5$.

Conclusion

In this paper, we propose STEAM for microservice workload prediction. To cope with the non-stationary and multi-periodic nature of the workload, STEAM uses a De-stationary Module to reintroduce non-stationary information that vanished in series normalization back into self-attention calculations, along with series decomposition to capture the long-term dependencies in the workload series. To capture dynamic dependencies between microservices, STEAM constructs two augmentation views of the workload similarity graph, and uses contrastive learning to extract shared consistency features between views as spatial features. A CNN-based short-term feature encoder is used to form a multi-scale spatio-temporal stacked structure. The experiments on two real-world datasets validate the effectiveness of STEAM.

Acknowledgements

This work was supported by the National Key R&D Program of China [2024YFF0617700], the National Natural Science Foundation of China [U23A20309, 62272302], Shanghai Municipal Science and Technology Major Project [2021SHZDZX0102], and CCF-Ant Research Fund [CCF-AFSG RF20220218, CCF-AFSG RF20230408].

References

Abdullah, M.; Iqbal, W.; Berral, J. L.; Polo, J.; and Carrera, D. 2022. Burst-Aware Predictive Autoscaling for Containerized Microservices. *IEEE Trans. Serv. Comput.*, 1448–1460.

Calheiros, R. N.; Masoumi, E.; Ranjan, R.; and Buyya, R. 2015. Workload Prediction Using ARIMA Model and Its Impact on Cloud Applications' QoS. *IEEE Trans. Cloud Comput.*, 449–458.

Chen, J.; Luo, Y.; Huang, X.; Jiang, F.; Shi, Y.; Zhang, T.; and Gao, X. 2023. IPOC: An Adaptive Interval Prediction Model based on Online Chasing and Conformal Inference for Large-Scale Systems. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 202–212.

Chen, P.; Zhang, Y.; Cheng, Y.; Shu, Y.; Wang, Y.; Wen, Q.; Yang, B.; and Guo, C. 2024. Pathformer: Multi-scale Transformers with Adaptive Pathways for Time Series Forecasting.

Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. E. 2020. A Simple Framework for Contrastive Learning of Visual Representations. In *International Conference on Machine Learning, ICML*, 1597–1607.

Gan, Y.; Liang, M.; Dev, S.; Lo, D.; and Delimitrou, C. 2021. Sage: practical and scalable ML-driven performance debugging in microservices. In *ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 135–151.

Google. 2021. Microservices in Google Cloud.

Hua, Q.; Yang, D.; Qian, S.; Hu, H.; Cao, J.; and Xue, G. 2023. KAE-Informer: A Knowledge Auto-Embedding Informer for Forecasting Long-Term Workloads of Microservices. In *ACM The Web Conference (WWW)*, 1551–1561.

Jayakumar, V. K.; Lee, J.; Kim, I. K.; and Wang, W. 2020. A Self-Optimized Generic Workload Prediction Framework for Cloud Computing. In *International Parallel and Distributed Processing Symposium (IPDPS)*, 779–788.

Ji, J.; Wang, J.; Huang, C.; Wu, J.; Xu, B.; Wu, Z.; Zhang, J.; and Zheng, Y. 2023. Spatio-Temporal Self-Supervised Learning for Traffic Flow Prediction. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI*, 4356–4364.

Kim, I. K.; Wang, W.; Qi, Y.; and Humphrey, M. 2016. Empirical Evaluation of Workload Forecasting Techniques for Predictive Cloud Resource Scaling. In *International Conference on Cloud Computing*, 1–10.

Krieger, M. T.; Tirado, Ó. T.; Trelles, O.; and Kranzlmüller, D. 2017. Building an open source cloud environment with auto-scaling resources for executing bioinformatics and biomedical workflows. *Future Gener. Comput. Syst.*, 329–340.

Li, F.; Yan, H.; Jin, G.; Liu, Y.; Li, Y.; and Jin, D. 2022a. Automated Spatio-Temporal Synchronous Modeling with Multiple Graphs for Traffic Prediction. In *ACM International Conference on Information & Knowledge Management*, 1084–1093.

Li, Z.; Huang, C.; Xia, L.; Xu, Y.; and Pei, J. 2022b. Spatial-Temporal Hypergraph Self-Supervised Learning for Crime Prediction. In *International Conference on Data Engineering, ICDE*, 2984–2996.

Liu, L.; Chen, J.; Wu, H.; Zhen, J.; Li, G.; and Lin, L. 2022a. Physical-Virtual Collaboration Modeling for Intra- and Inter-Station Metro Ridership Prediction. *IEEE Trans. Intell. Transp. Syst.*, 3377–3391.

Liu, S.; Yu, H.; Liao, C.; Li, J.; Lin, W.; Liu, A. X.; and Dustdar, S. 2022b. Pyraformer: Low-Complexity Pyramidal Attention for Long-Range Time Series Modeling and Forecasting. In *The Tenth International Conference on Learning Representations, (ICLR)*.

Liu, Y.; Wu, H.; Wang, J.; and Long, M. 2022c. Non-stationary Transformers: Exploring the Stationarity in Time Series Forecasting. In *Annual Conference on Neural Information Processing Systems, NeurIPS*.

Luo, S.; Xu, H.; Ye, K.; Xu, G.; Zhang, L.; Yang, G.; and Xu, C. 2022. The Power of Prediction: Microservice Auto Scaling via Workload Learning. In *ACM Symposium on Cloud Computing SoCC*.

Luo, Y.; Gao, M.; Yu, Z.; Ge, H.; Gao, X.; Cai, T.; and Chen, G. 2024a. Integrating System State into Spatio Temporal Graph Neural Network for Microservice Workload Prediction. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Luo, Y.; Gu, Z.; Zhou, S.; Xiong, Y.; and Gao, X. 2023. Meteorology-Assisted Spatio-Temporal Graph Network for Uncivilized Urban Event Prediction. In Chen, G.; Khan, L.; Gao, X.; Qiu, M.; Pedrycz, W.; and Wu, X., eds., *IEEE International Conference on Data Mining, ICDM*, 468–477.

Luo, Y.; Wang, S.; Yu, Z.; Lu, W.; Gao, X.; Ma, L.; and Chen, G. 2024b. Adaptive Two-Stage Cloud Resource Scaling via Hierarchical Multi-Indicator Forecasting and Bayesian Decision-Making. *CoRR*.

Nie, Y.; Nguyen, N. H.; Sinthong, P.; and Kalagnanam, J. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *International Conference on Learning Representations, ICLR*.

Park, J.; Choi, B.; Lee, C.; and Han, D. 2021. GRAF: a graph neural network based proactive resource allocation framework for SLO-oriented microservices. In *CoNEXT '21: The 17th International Conference on emerging Networking EXperiments and Technologies,*, 154–167.

Tam, D. S. H.; Liu, Y.; Xu, H.; Xie, S.; and Lau, W. C. 2023. PERT-GNN: Latency Prediction for Microservice-based Cloud-Native Applications via Graph Neural Networks. In *Conference on Knowledge Discovery and Data Mining, KDD*, 2155–2165.

Tan, C. N. L.; Klein, C.; and Elmroth, E. 2019. Multivariate LSTM-Based Location-Aware Workload Prediction for

Edge Data Centers. In *International Symposium on Cluster, Cloud and Grid Computing, CCGRID*, 341–350.

Taylor, S. J.; and Letham, B. 2017. Forecasting at Scale. *PeerJ Prepr.*, e3190.

Wang, S.; Sun, Y.; Shi, X.; Zhu, S.; Ma, L.; Zhang, J.; Zheng, Y.; and Jian, L. 2023. Full Scaling Automation for Sustainable Development of Green Data Centers. In *International Joint Conference on Artificial Intelligence IJCAI*, 6264–6271.

Wang, X.; Liu, N.; Han, H.; and Shi, C. 2021. Self-supervised Heterogeneous Graph Neural Network with Co-contrastive Learning. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 1726–1736.

Wu, H.; Hu, T.; Liu, Y.; Zhou, H.; Wang, J.; and Long, M. 2023. TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis. In *The Eleventh International Conference on Learning Representations, ICLR*.

Wu, H.; Xu, J.; Wang, J.; and Long, M. 2021. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. In *Annual Conference on Neural Information Processing Systems, NeurIPS*, 22419–22430.

Wu, Z.; Pan, S.; Long, G.; Jiang, J.; Chang, X.; and Zhang, C. 2020. Connecting the Dots: Multivariate Time Series Forecasting with Graph Neural Networks. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 753–763.

Yu, B.; Yin, H.; and Zhu, Z. 2018. Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. In *International Joint Conference on Artificial Intelligence IJCAI*, 3634–3640.

Zeng, A.; Chen, M.; Zhang, L.; and Xu, Q. 2023. Are Transformers Effective for Time Series Forecasting? In *Conference on Artificial Intelligence, (AAAI)*, 11121–11128.

Zhou, T.; Ma, Z.; Wen, Q.; Wang, X.; Sun, L.; and Jin, R. 2022. FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting. In *International Conference on Machine Learning, (ICML)*, 27268–27286.