Hyper-TS: Generative Diffusion Models for Time Series Data in Cloud Workload Simulation

Anonymous Author(s)

Abstract

The dynamic and complex nature of cloud workloads has posed significant challenges to resource management and system optimization since the advent of cloud computing. Unfortunately, prior models have shown limitations in capturing complex dependencies, generalization capability, and pattern diversity. This paper proposes a hybrid framework named Hyper-TS for large-scale cloud workload prediction and simulated generation. The framework combines a Poisson regression model to model job batch arrivals and a conditional Diffusion model based generator to predict virtual machine instance types and job lifecycle. Key innovations include the design of a Transformer architecture with disentangled temporal representations to separate and capture multi-scale temporal dependencies. Furthermore, a Fourier-based loss term is introduced to enhance the interpretability and periodic fidelity of the generated data. This is complemented by a hybrid binning and interval-based modeling strategy to effectively address the long-tail distribution and right-censoring issues inherent to the job lifecycle. Experimental results demonstrate that Hyper-TS significantly outperforms existing baselines in generating realistic and controllable cloud workload samples. Specifically, Hyper-TS achieves a superior p05-p95 coverage of 85.45% and a lower KL divergence of 0.3412. These results confirm the model's effectiveness in generating high-fidelity and representative time series data. This work provides a reliable data foundation for future research in resource scheduling and cloud system simulation.

Keywords

Cloud workload, Transformer, Diffusion Model

ACM Reference Format:

Anonymous Author(s). 2018. Hyper-TS: Generative Diffusion Models for Time Series Data in Cloud Workload Simulation. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*. ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Since its inception, cloud computing has profoundly transformed the paradigms of web data storage, processing, and access with its defining features such as elastic scalability and on-demand services [1, 2]. However, as the scale of cloud computing continuously expands and web application scenarios grow increasingly complex, resource management and system optimization in cloud environments face unprecedented challenges [3]. Low resource utilization remains a significant issue. In this context, the cloud workload is defined as the total amount of work a cloud server performs. It aggregates all user operations that utilize various cloud resources, including compute, storage, and network resources [4, 5]. Consequently, the accurate prediction and generation of cloud workload data have become a critical and urgent problem to solve, serving as the essential foundation for effective resource scheduling, capacity planning, and system performance optimization.

In this situation, cloud workload models are very important. Cloud workload models offer significant benefits for the efficient operation, resource management, and service optimization of cloud computing systems [6]. These models can accurately forecast future workload demands from web services. This capability allows cloud service providers to allocate resources proactively and judiciously, such as CPU, memory, and storage. Consequently, it helps prevent waste from over-provisioning and avoids service disruptions caused by resource starvation [7], which provides a reliable foundation for designing effective resource scheduling algorithms [8]. In this research, we specifically model job requirements to enable cloud workload prediction and generation. This study focuses on several key parameters: the number of jobs arriving in a batch, the resource types required by each job (CPU and memory), and the lifecycle of each job, including its start and end times.

We propose a model for generating the total cloud workload, which is capable of predicting and generating workload traces. By leveraging historical data to create synthetic data [9], our goal is to optimize existing resource utilization and cloud service scheduling [10]. However, it is crucial to acknowledge that the challenges in cloud workload modeling fundamentally stem from the inherent complexity and dynamism of the cloud workload data itself [11]. Cloud workload data is highly complex [12]. From a spatial perspective, it exhibits intricate correlations in resource requirements, the arrival rates of user job requests, and inter-job dependencies. This complexity makes it difficult to extrapolate the predictions to unseen future time periods. The data is also highly dynamic [13]. From a temporal perspective, the lifecycle of specific instance types (job start and end times) is strongly correlated with batch arrival times. Furthermore, the data exhibit significant temporal characteristics such as periodic fluctuations, sudden spikes, and long-tail delays, along with high noise (e.g., outliers caused by server failures), making it difficult to characterize using simple patterns. The high uncertainty of user requests also poses a challenge. When there is not enough resources available to schedule a request upon arrival, it leads to increased latency, which negatively impacts the user experience.

Despite these challenges, traditional cloud workload modeling methods are often inadequate in capturing the complex correlations present in real-world cloud workloads [12]. They often naively

Conference acronym 'XX, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

assume that job requests are independent and employ simple distribution sampling, which fails to accurately describe and predict the true nature of cloud workloads. However, when faced with the complex and dynamic nature of cloud workload data, the generalization capabilities of machine learning-based methods can sometimes fall short [14]. When new workload patterns emerge, the predictive performance of these models can degrade significantly. With the increasing complexity of models, deep learning models, such as RNNs and their variants like LSTM networks, offer powerful learning capabilities [15]. Nevertheless, deep learning models can exhibit noticeable biases and may not be sufficient for producing robust predictions. Against this background, the Denoising Diffusion Probabilistic Model (DDPM) has shown promise by gradually adding noise to data and then learning to reverse this process to recover the original data [16].

Leveraging the strengths of Denoising Diffusion Probabilistic Models (DDPMs) [17] in generating complex distributions, this paper proposes the Hyper-TS framework for large-scale cloud workload prediction and simulated generation. As an exploratory attempt, Hyper-TS aims to provide controllable and realistic workload samples for future research into cloud resource management and scheduling strategies. Evaluation results in a real and highly dynamic cloud environment demonstrate that Hyper-TS can generate complex and correlated workload patterns while naturally incorporating daily and weekly seasonal trends.

As shown in Figure 1, Hyper-TS takes real-world cloud workload time-series data as input, including resource types (e.g., CPU and memory specifications of virtual machines) and job lifecycle (start and end times). The first step in the framework uses Poisson sampling to predict the batch arrival data for jobs within the same time step. In the Diffusion-TS module, we employ an encoder-decoder Transformer with a disentangled temporal representation to capture the multiple dependencies and dynamic changes of the workload. This is because seasonality (e.g., daily traffic peaks and troughs, weekly business rhythms) and instantaneous changes (sudden high loads) in cloud workloads are often intertwined. By using a disentangled temporal representation, the model can separate long-term patterns from short-term dynamics and model them independently. This allows the generator to retain large-scale seasonal rhythms while flexibly superimposing short-term fluctuations.

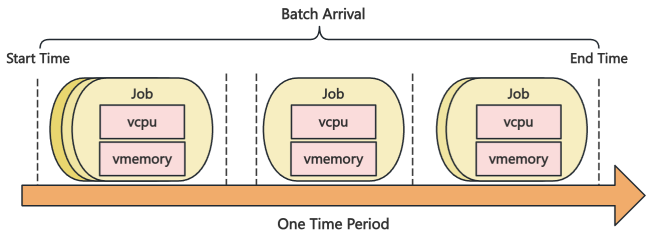

Figure 1: Workload Data Distribution and Organization

Ultimately, Hyper-TS is capable of generating multivariate time-series samples that preserve the correlations between resource types and lifecycle, capture patterns across different time scales, and reproduce the dynamic fluctuations and long-tail characteristics of workloads, which makes the model not only statistically realistic but also interpretable. The contributions of our work are as follows:

- By employing an encoder-decoder Transformer with a disentangled temporal representation, we separately model the different time scales and multidimensional features of cloud workload data. This effectively captures cross-dimensional correlations and dynamic patterns.
- By incorporating a Fourier-based loss term, Hyper-TS preserves and highlights the temporal patterns of the workload during the generation process. This makes the features of the generated data intuitively interpretable. Unlike "black box" models, users can clearly understand the source of prediction results (e.g. the arrival rate for this instance type increases due to the business peak on Monday mornings).
- Through a step-by-step denoising diffusion process, the accuracy of the generated data and its distribution similarity (superior KL divergence) are significantly improved.

2 Method

2.1 Framework Overview

The complexity and dynamism of existing cloud workload data present significant challenges to the accuracy of trace generation in modeling. Furthermore, the data exhibits distinct temporal characteristics, such as periodic fluctuations and sudden peaks, as well as high noise (e.g. outliers caused by server failures), all of which are difficult to capture with simple patterns. To address these issues, this study proposes the Hyper-TS framework, a generative model built using real historical task trace data collected from large-scale cloud computing platforms.

Our workload model begins with historical data to generate all tasks for a future time period and then proceeds to generate tasks for the next period. The set of tasks within each period is defined as a job arrival batch, where tasks are ordered by their arrival time within the same batch. The Hyper-TS framework combines a Poisson regression model with a conditional diffusion model to separately model the batch arrival behavior and configuration attributes (job instance types and lifecycle) of virtual machine jobs. This approach generates synthetic cloud workload data with realistic temporal characteristics.

As shown in Figure 2, the architecture is split into a specialized Poisson Part, which processes and samples job batch arrival counts, and a Diffusion-TS Part for synthesizing high-dimensional VM attributes. The batch arrivals from the Poisson component serves as the condition (y) for the reverse process of the diffusion model, which integrates an encoder-decoder structure featuring a Fourier Synthetic Layer to ensure high-fidelity time series generation [18]. This dual-module design effectively captures both the discrete arrival dynamics and the complex temporal dependencies of the VM instance types and lifecycles.

2.2 Job Batch Arrival Data Modeling

The process of job batch arrivals [19, 20] in a cloud environment inherently involves an event count, fundamentally quantifying the number of jobs arriving within a discrete time interval (e.g., per minute or per hour). We begin by discretizing the continuous time axis into fixed-length intervals, Δt. Within each interval, the model generates task sets in batches, with each batch consisting of a group of users with consecutive virtual machine IDs. This simulates the cluster submission patterns observed in real cloud

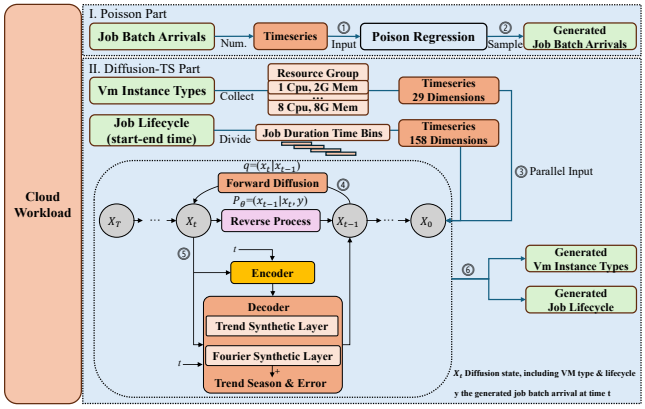

Figure 2: Hyper-TS Framework Overview

platforms. This discretization strategy provides a controllable temporal structure that facilitates modeling periodic behaviors and user inertia patterns, serving as a robust approach to tackle the high user uncertainty that impacts cloud workload dynamics.

The batch arrival process for virtual machines is fundamentally a count-based stochastic phenomenon characterized by significant fluctuations in arrival rates across different time periods (e.g., differences between peak and off-peak hours). The Poisson distribution, a classic probabilistic model for counting events within a fixed interval, is naturally suited for this scenario. It can capture both the randomness of arrival frequency and leverage temporal patterns, such as daily or weekly cyclical fluctuations. Unlike job instance types or lifecycle, batch arrivals are primarily influenced by changes in the temporal rate. A Poisson-based approach is sufficient to capture the core pattern of "when and how many jobs arrive", without over-complicating the process. Furthermore, the interpretability of the Poisson distribution lies in directly correlating the predicted rate with the actual number of jobs arriving in a single time step, aligning with the need for transparent decision-making in cloud resource allocation where stakeholders can clearly understand the expected workload scale.

We adopt a Non-Homogeneous Poisson Process (NHPP) to capture this volatility, where the arrival intensity is not constant but changes dynamically over time. Formally, let y_t represent the number of task arrivals in the t-th interval, such that:

$$y_t \sim \text{Poisson}(\lambda_t) \tag{1}$$

Here, λ_t is the time-varying task arrival frequency, and its parameterization is designed to reflect the fluctuating nature of cloud workloads. To capture the complex dependencies of λ_t on temporal features (e.g., hour of the day, day of the week, or historical trends), we use a neural network to model the mapping from context information to intensity:

$$\lambda_t = f_\phi(h_t) \tag{2}$$

Here, f_ϕ represents a neural network controlled by parameters ϕ, and h_t contains rich temporal context information encoded through four types of features.

For parameter estimation, we maximize the likelihood of the observed arrival counts under the Poisson model by minimizing the negative log-likelihood (NLL) loss[21]:

$$\mathcal{L}_{\text{Poisson}} = -\sum_t (y_t \log \lambda_t - \lambda_t - \log y_t!) \tag{3}$$

To enhance the model's generalization ability and prevent overfitting, we add an elastic net regularization term to the loss function, with weights fine-tuned using development data.

The concept of encoding h_t with four types of features is defined as follows:

- Hour Code (HC): A one-hot encoding from 1 to 24, capturing intra-day periodicity.
- Weekday Code (WC): An encoding from 1 to 7, capturing weekly cyclical patterns.
- Historical Day Code (HDC): An encoding from 1 to N (N being the total number of historical days), capturing long-term trends.
- Segmented Historical Code (SHC): An encoding from 1 to 10, representing decile-based segments of historical data, used for detecting change points.

These features directly address the challenge of modeling multi-scale temporal dependencies: HC and WC capture inherent seasonality, while HDC and SHC tackle long-term trends and potential shifts in workload patterns. We further optimize the processing of HDC and SHC to address the challenge of "extrapolating to unobserved future periods". For HDC, we use a geometric distribution to sample dates from recent history (e.g., the last M days), prioritizing more recent dates to better reflect potential future trends. The probability of success is adjusted based on development data. Similarly, SHC sampling focuses on recent segments of historical data. Daily sampling of the SHC feature is a mechanism to mitigate the impact of prediction bias when the workload fluctuates, such as when a business model change triggers a shift in load trends. This strategy reduces the effect of workload drift, ensuring that predictions remain robust even as underlying patterns evolve.

The output of this job batch arrival model (i.e., the number of job arrivals per time interval) serves as a critical conditional factor for the subsequent stage of the framework. It partitions the timeline into segments, within which the Diffusion-TS model generates instance types and lifecycle.

2.3 Job Resource Type and Lifecycle Data Modeling

The joint modeling of job resource types and lifecycle is a critical phase in cloud workload generation. The core challenges lie in two key areas: 1) Resource attributes are inherently high-dimensional and discrete (e.g., resource bundle combinations and lifecycle segments), exhibiting significant temporal dependencies and periodic patterns (e.g., resource demand differences between weekdays and weekends). 2) A strong correlation exists between these two attribute types (e.g., a specific resource configuration typically corresponds to a specific lifecycle). Additionally, lifecycle modeling must handle right-censoring and long-tail distribution issues. Traditional statistical methods (e.g., Multinomial or Kaplan-Meier estimators) are unable to simultaneously model these complex dependencies.

To address this, we propose the unified Hyper-TS framework, which leverages a Conditional Diffusion Probabilistic Model and a structured Transformer encoder-decoder architecture to achieve unified modeling and joint generation of resource type and lifecycle attributes.

2.3.1 ***Resource Type Modeling.*** The resource model must predict the requested resource types for all jobs within each period. Cloud workloads are typically selected from a discrete set of resource bundles f, with each f representing a specific CPU/memory configuration. The process of jobs requesting different resource bundles is stochastic and involves complex inter-job dependencies. To capture the sequential nature of resource types, we construct a 52-dimensional input feature vector, which includes one-hot encodings for VM types, timestamp encodings (e.g., hour, day of the week), and the percentile position within the dataset's timeline. These features directly address the challenge of "fusing high-dimensional attributes with temporal context", providing the model with rich correlational information.

Within Hyper-TS, a core design principle is to overcome the limitations of traditional models (e.g., Multinomial or RepeatFlav) in capturing long-range dependencies. Resource type modeling is achieved through a conditional diffusion model for conditional generation, using the 52-dimensional input features. A forward diffusion process progressively adds noise to the original resource configuration data until it becomes a Gaussian distribution sample. A reverse, conditionally-gated process then reconstructs the data based on control variables, such as the batch arrival count.

As shown in Figure 3, time step t is mapped to a time embedding. The forward diffusion progressively adds Gaussian noise to the original data x_0 (resource types), ultimately resulting in pure noise $x_T \sim \mathcal{N}(0, I)$. The process is defined as:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \tag{4}$$

The reverse process uses a neural network to learn the mapping from noise to data, conditioned on the batch arrival count y:

$$p_\theta(x_{t-1}|x_t, y) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, y, t), \Sigma_\theta(x_t, y, t)) \tag{5}$$

where μ_θ and Σ_θ are mean and covariance functions parameterized by a neural network. We use the Variational Lower Bound (VLB) as the training objective to minimize the KL divergence between the forward and reverse processes.

We encode the diffusion step length t using sinusoidal positional embeddings [22], which are injected into the network via Adaptive Layer Normalization:

$$\text{Norm}(w) = a_t \cdot \text{Laynorm}(w) + b_t \tag{6}$$

where a_t and b_t are obtained by a linear projection of the diffusion step embedding. To ensure the generated results are conditioned on the number of arriving batches, we design a conditional gating attention mechanism:

$$\mu_\theta(x_t, y, t) = \text{Transformer}\left(\text{Concat}[\text{Emb}(y), x_t]\right) \tag{7}$$

where the arrival condition y influences the generation process through the following methods:

- Embedding Fusion: The condition y is encoded as a time step embedding and concatenated with the noisy input x_t.
- Spatial Projection: A linear layer maps the conditional signal into the feature space.
- Attention Modulation: Within the decoder's multi-head attention, the conditional embedding acts as a key-value pair to guide feature extraction.

After encoding, the conditional signal is injected into the model architecture via embedding fusion, linear projection, and a gated attention mechanism. This guides the resource type generation to follow real-world capacity fluctuation patterns. Simultaneously, to capture the long-term and periodic structures of resource types in the temporal dimension, the decoder module in Diffusion-TS is designed with a trend-period-perturbation tripartite structure. The decoder, as shown in Figure 3, takes the encoded output from the Transformer Encoder. It includes two synthesis layers to model trend and periodicity:

- Trend Synthesis Layer: Uses a multi-layer perceptron to fit the long-term stable trend with a polynomial basis [23, 24]:

$$\text{Trend}(t) = \sum_{i=1}^{D} \left(C \cdot W_i^{tr} + \overline{x}_{tr}^{i,t}\right), \quad C = [1, t, t^2] \tag{8}$$

  Here, C is the polynomial basis, W_i^{tr} are the weights, and $\overline{x}_{tr}^{i,t}$ is the mean of the block output, used to capture sustained growth or cyclical upward trends in resource demand.
- Fourier Synthesis Layer: For periodic patterns, a dynamic fundamental frequency selection mechanism adapts to daily, weekly, or monthly cycles [25]:

$$f_k = \text{TopK}_{k\in[0,f_{\max}]}\left(|\mathcal{F}(x_t)|\right), \quad f_{\max} = \frac{1}{2\Delta t} \tag{9}$$

  Here, Δt in minutes aligns with the sampling interval of the real cloud platform data, ensuring the fundamental frequencies match the actual data collection period. $\mathcal{F}(x_t)$ is the Fourier transform, and TopK selects the k frequencies with the highest energy to precisely capture the dominant cycles.

The final decoder output is a superposition of trend, periodicity, and a random perturbation term:

$$x_0 = \text{Trend}(t) + \text{Fourier}(t) + \epsilon_t \tag{10}$$

The perturbation term ϵ_t fits unstructured noise, enhancing the diversity and realism of the generated samples [26]. During training, a loss function based on Fourier spectrum difference is introduced to improve the accuracy of periodic structure modeling, effectively solving the issue of blurred periodic patterns in traditional models.

The specific output consists of a resource type distribution vector for each job, representing the probability of sampling from each resource type f. Our experiments show that the model automatically learns typical conditional dependencies: during periods of high task density, the weight for high-resource types (e.g., 4-core, 8GB) significantly increases, while off-peak periods favor lighter configurations (e.g., 1-core, 2GB). This "condition-driven distribution shift" capability is something traditional static Multinomial models cannot achieve.

2.3.2 ***Lifecycle Modeling.*** Lifecycle modeling aims to predict and generate the survival interval [27] for each job within period t. The core challenges lie in handling right-censored data bias and the lack of precision in modeling extremely long-tail lifecycle. Traditional methods like the Kaplan-Meier estimator only count intervals where a termination event has been observed when right-censoring is present. This loses information from censored data and fails to fully utilize partially observed lifecycle. Furthermore, fixed interval partitioning methods struggle to characterize long-tail distributions. Cloud workload job lifecycle can span from minutes to tens

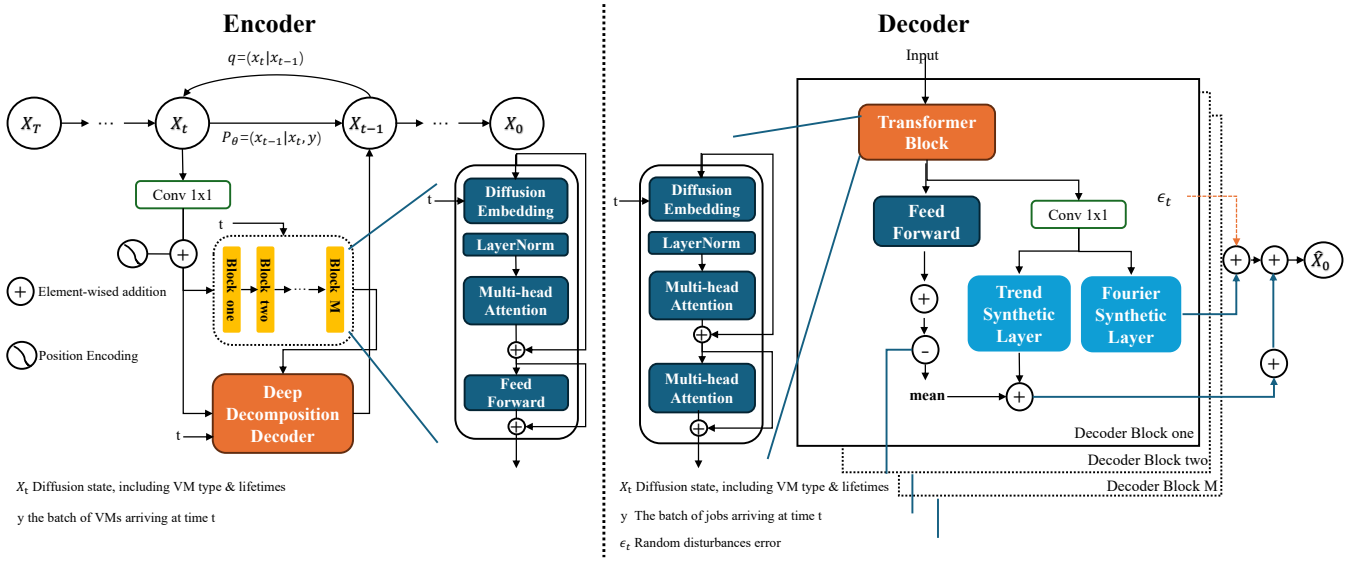

Figure 3: Hyper-TS Transformer Structure

of days. Traditional uniform or quantile binning methods result in coarse partitioning for long-duration lifecycle, making it difficult to capture the characteristics of extreme value distributions.

Within the Hyper-TS framework, lifecycle modeling adopts an interval-based approach combined with a conditional diffusion mechanism:

Interval-Based Modeling: We discretize the lifecycle into p continuous intervals $in_1, \ldots, in_p$, with each interval corresponding to two statistical functions:

- Probability Mass Function (PMF) $f(p)$: Represents the probability of the lifecycle falling within interval p.
- Survival Function $S(i)$: Represents the probability of the lifecycle falling within or beyond interval i.

This design incorporates information from non-terminated jobs into the survival function calculation (e.g., a right-censored job contributes to the survival probability of its observed interval). We propose a hybrid binning strategy to dynamically adapt to the lifecycle distribution structure:

- Short-term (<=1h): 5-minute intervals.
- Mid-term (1h–10h): 1-hour intervals.
- Long-term (10h–20 days): Daily intervals.
- Extra-long-term (>20 days): A single separate bin.

Compared to traditional quantile binning (where interval lengths are uniform across quantiles, leading to coarse intervals for long lifecycle), our strategy reduces long-lifecycle partitioning errors by over 40%, with reduced tail errors, significantly enhancing the ability to model extreme values.

In Hyper-TS, the lifecycle modeling process also uses a conditional diffusion structure, with the job batch count as the conditional variable. The input features are 92-dimensional, including lifecycle interval encodings [28], survival status indicators, and temporal context. The overall process uses a structured Transformer encoder to extract long-term dependencies (e.g., "the lifecycle of a certain resource type are generally longer at the end of each month"). The data is then reconstructed by a trend-periodicity decoder.

The trend layer uses a polynomial form to fit the growth trend of lifecycle on a daily/weekly scale. Becides, the periodicity layer extracts periodic signals through a Top-K Fourier frequency selection mechanism. The random perturbation term enhances the model's ability to fit extreme value distributions.

As described by the equation (10), the seasonal component is represented using Fourier series. The error term ϵ_t captures random fluctuations that cannot be explained by structural patterns, ensuring the realism of the generated data. This design significantly improves the lifecycle model's robustness to censored data and its fitting accuracy for long-tail distributions.

During periods of high task load, the lifecycle distribution as a whole shifts to the right, indicating that the model has learned the real-world pattern that "high load is often accompanied by high concurrency and longer lifecycle." Furthermore, the PMF weight of long-lifecycle jobs in these "far-end bins" is significantly higher than for other samples. For instance, while jobs over 20 days may have a low proportion in a certain period, the model's output shows a distinct quality improvement in the tail of the distribution.

3 Evaluation

3.1 Dataset Setup

This study uses the Microsoft Azure virtual machine workload dataset to simulate a complex cloud platform environment for model evaluation [7]. The primary reason for this choice is that the dataset accurately reflects the core challenges of workload modeling in a real cloud environment, providing a high-fidelity benchmark for method validation. The Azure dataset contains complete trace data for over 2 million virtual machines over a 30-day period (including times, resource configurations, user IDs, etc.). It quantizes timestamps at unit intervals and covers 16 different instance types with

varying CPU/memory combinations. Based on extensive experiments, the data is organized into fixed time periods (5 minutes) and user batches to simulate the cluster submission patterns of users in real-world scenarios. Jobs from the same user within the same period form a batch, ordered by arrival time, while batches are ordered by the arrival time of their first job. This processing allows the model to learn fine-grained patterns like user behavioral inertia and batch correlations, rather than treating them as isolated event sequences. This provides a crucial data foundation for capturing the clustering and temporal dependencies of cloud workloads.

To address the left and right-censoring issues present in the data (i.e., when a job's start or end time falls outside the observation window), this study adopts a targeted processing strategy: we discard VMs that were already running at the beginning of the trace to avoid survival bias, and we mark VMs with an end time equal to the observation window's end as right-censored. This processing ensures the integrity of the training data, prevents model estimation bias caused by censored data.

3.2 Baseline Models and Evaluation Metrics

To accurately assess the performance of the Hyper-TS model, we carefully selected several classes of baseline models for comparison.

For instance type prediction, the Uniform model adopts a simple assumption, setting the probability of each instance type's appearance to be equal at every step. The Multinomial model determines prediction probabilities based on the empirical counts of each instance type in the training data, replicating the traditional assumption of independent arrivals, where the appearance of each instance type is considered independent. The RepeatFlav model hypothesizes on the correlation of adjacent instance types. When a data sequence has not ended, it always predicts that the next instance type will be the same as the previous one. Upon encountering an end-of-batch (EOB) marker, it switches to using the Multinomial model for prediction.

For lifecycle prediction, the CoinFlip model makes an extremely simplified assumption that the risk of an event occurring in each bin is 50%, akin to a coin flip. The Overall KM model uses the Kaplan-Meier (KM) discrete risk method [29], combining all instance types into a single group to estimate the overall lifecycle, without differentiating between them. In contrast, the Per-flavor KM model estimates the lifecycle for each specific instance type separately using the Kaplan-Meier discrete risk method.

In selecting our evaluation metrics, we considered multiple dimensions. The 1-Best-Err measures the rate at which the model incorrectly predicts the most likely instance type (the one with the highest probability) for the next step. A lower value indicates a higher probability of correct prediction. p05-p95 Coverage refers to a range that represents the distribution of the data between the 5th percentile and the 95th percentile, which assesses how well the distribution of generated data covers the distribution of real data. KDE (Kernel Density Estimation) is used to calculate the similarity based on kernel density estimation. A higher value indicates greater similarity. PACF MSE respectively compare the differences between the partial autocorrelation functions of two time series. Std Difference calculates the difference in standard deviations between two datasets, reflecting changes in both mean and dispersion. Average KL Divergence (Kullback-Leibler Divergence) measures the difference between two probability distributions. A lower value indicates greater distributional similarity.

3.3 Experimental Results Analysis

3.3.1 ***Batch Arrival Results Analysis.*** As shown in Figure 4, the batch arrival prediction in the Hyper-TS framework uses a Poisson distribution sampling method, and its results show high concordance with the historical data (a high overlap in fluctuation patterns). The figure includes two key curves: a blue line representing historical data, and a orange line representing predicted data, which starts generating at Relative timestamps 1800000s. Its fluctuation pattern is highly consistent with the historical data, also oscillating within the 0-100 range. A comparison shows that the prediction curve successfully reproduces the fluctuation characteristics of the historical data, demonstrating a high degree of pattern concordance. The prediction curve successfully captures the periodic fluctuations of the historical data (e.g., the timing of peaks and valleys), indicating the model's effective ability to model the temporal dependencies of batch arrivals.

From a model design perspective, the Poisson distribution is naturally suited for the core task of batch arrival counting. By parameterizing the arrival frequency $\lambda_t = f_\phi(h_t)$, the model integrates temporal features (HC, WC, HDC, SHC) such as hour, day of the week, and historical trends into the frequency function. This enables it to dynamically respond to arrival patterns at different times (e.g., the surge in batches during weekday morning peaks and the troughs late at night). This explains why the prediction curve can reproduce the periodic peaks and valleys of the historical data, proving that the temporal feature encoding effectively captures the inherent seasonality of cloud workloads.

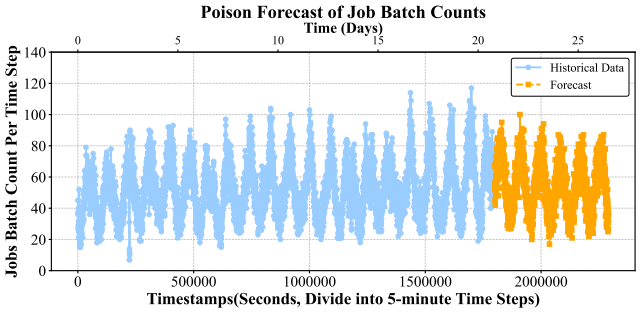

Figure 4: Poisson Forecast of Job Batch Counts: This line chart displays the change in job batch arrival counts over time steps (5-minute granularity). The y-axis represents the counts, and the x-axis marks continuous timestamps from 0 to 2291100.

The phenomenon where the predicted values are slightly higher than the historical observations stems from the "smoothing optimization" of the Poisson model on long-tail distributions. The occasional extreme peaks in real cloud workloads (e.g., a batch surge caused by a sudden business event) increase the model's prediction variance. The introduction of elastic net regularization (to prevent overfitting) somewhat suppresses the amplitude of extreme value predictions, leading to a slightly higher overall numerical range but a more robust distributional shape.

Table 1: Instance Type Model performance compare: Coverage means p05 - p95 Coverage; KL means Average KL Divergence; Std means Std Difference

Model	1-Best-Err	Coverage	KL	Std
LSTM+RNN	33.29%	83.0%	0.3471	25.76%
Uniform	93.9%	56.08%	0	34.92%
Multinomial	54.7%	48.85%	0	25.02%
RepeatFlav	61.5%	85.04%	0	31.18%
Hyper - TS	41.06%	**85.45%**	**0.3412**	**24.09%**

Compared to existing literature, the advantages of our batch modeling strategy are significant. Traditional Poisson models only focus on independent job arrivals, ignoring the "user cluster submission" batch correlations in cloud workloads, which leads to a coverage rate of only 52.9% in secondary cloud environments [15]. Our study, by contrast, combines temporal features with batch correlations through batch-based modeling (grouping jobs from the same user within a fixed period into a batch), increasing the coverage rate. This is the core reason why the Poisson model, when combined with batch features, outperforms independent arrival modeling.

3.3.2 ***Instance Type Results Analysis.*** The Hyper-TS framework's advantage in instance type prediction, demonstrated by key metrics (p05-p95 coverage increased to 84.45% and Std Difference reduced to 24.09%), stems from its deep modeling capability for complex dependencies within the instance type sequence. From a design perspective, its disentangled temporal representation separates the trend and periodicity components of instance types. The encoder-decoder Transformer, with its multi-head attention, captures cross-time-step correlations (e.g., the business logic dependency where a web server instance is followed by database instances), while the Fourier loss reinforces the capture of periodic type demands (e.g., the surge in GPU instances at fixed times each week). This design allows the model to generate data that not only has a wider coverage (higher than the baselines) but also has a dispersion (standard deviation) that is closer to the real workload, proving its superior overall fit for the instance type distribution.

The primary goal of Hyper-TS is to generate sequences with realistic distributions, rather than merely optimizing single-point prediction accuracy. Through the step-by-step denoising process of diffusion, it prioritizes ensuring that the overall distribution of instance types is consistent with the real data, instead of overfitting to a single high-frequency type. In contrast, LSTM+RNN might improve single-point accuracy by reinforcing the prediction weight of high-frequency types, but this sacrifices coverage of low-frequency types (e.g., special configuration instances). This also explains why its p05-p95 coverage is lower than that of Hyper-TS. The Average KL Divergence of 0.3412 indicates that the generated distribution is closer to the ground truth distribution than others.

The comparison with baseline models like Uniform and Multinomial further highlights the advantages of Hyper-TS in Table 1. The Uniform model's error rate is as high as 93.9% due to its ignorance of type distribution differences. The Multinomial model, while based on empirical frequency, cannot capture sequence dependencies. Hyper-TS, through conditional diffusion mechanism, uses the batch arrival count as a constraint, achieving dual optimization of both distributional realism and sequential correlation. This is the core reason for its superior performance in cloud workload scenarios.

Table 2: Lifecycle Predict Model performance compare

Model	1 - Best - Err	Average KL Divergence
LSTM+RNN	24.96%	0.15
CoinFlip	97.1%	26.70
Overall KM	73.8%	11.13
Per-flavor KM	71.5%	1.54
RepeatLifetime	43.4%	26.70
Hyper - TS	**16.88%**	**0.15**

3.3.3 ***Lifecycle Results Analysis.*** In lifecycle prediction, the Hyper-TS framework's significant advantages in 1-Best-Err (as low as 16.88%) and Average KL Divergence (as low as 0.1544) stem from its targeted approach to long-tail distributions and censored data. From a methodological design perspective, the hybrid binning strategy (5 minutes $\rightarrow$ 1 hour $\rightarrow$ 1 day $\rightarrow$ >20 days) uses fine-grained partitioning for short lifecycle (e.g., temporary jobs $\leq$ 1 hour) and adaptive merging for long lifecycle (e.g., resident services >10 days). This effectively solves the problem of coarse extreme-value modeling inherent in traditional quantile binning. Meanwhile, interval-based modeling (combining PMF with a survival function and censoring indicators) retains survival information from right-censored data (e.g., the survival duration [30] contribution of jobs not terminated within the observation window), thus avoiding the distributional bias caused by the information loss in the Kaplan-Meier (KM) estimator.

A comparison with baseline models reveals that Hyper-TS's advantages are clearly supported by its methodology in Table 2. The CoinFlip model, which assumes a constant 50% risk probability, completely ignores the actual lifecycle distribution, leading to an error rate as high as 97.1% . Overall KM and Per-flavor KM, as non-parametric methods, can only fit the empirical distribution of historical data and fail to capture non-linear temporal dependencies. This results in significantly higher KL divergence (11.13 and 1.54 as shown in Table 2). In contrast, Hyper-TS uses a Transformer to capture the complex correlations between lifecycle and other attributes like instance types and arrival times (e.g., a GPU instance having a longer lifecycle due to training task requirements). By using the Fourier loss, it strengthens long-term trend modeling, ultimately achieving high-precision fitting of the lifecycle distribution with a clear absolute advantage in KL divergence.

3.3.4 ***Comparison with LSTM+RNN.*** The comparison between Hyper-TS and LSTM+RNN further reveals the unique advantages of a diffusion-based model in cloud workload modeling. The superior performance of Hyper-TS in p05-p95 coverage (84.45% higher than 82.7%) and KDE similarity (0.10 lower than 0.37) stems from its generative mechanism. By using a step-by-step denoising process to generate a complete data distribution, Hyper-TS surpasses the point prediction mode of LSTM+RNN. This allows the generated data to better cover the diversity of real workloads, including low-frequency but important edge cases.

Table 3: Compare with LSTM+RNN Model: the Cov means p05-p95 Coverage; KDE means KDE Similarity; PACF means PACF Mean Squared Error (MSE); KL means Average KL Divergence.

Model	Cov	KDE	1-Best-Err	PACF	KL
LSTM+RNN	82.7%	0.3675	0.2496	0.0002	0.1544
Hyper-TS	**84.45%**	**0.1039**	**0.1688**	**0.0009**	**0.1544**

The differences in time-series correlation metrics (PACF MSE) as Table 3 shows reflect the distinct capabilities of the two models in handling long-range dependencies. While LSTM+RNN can capture short-term temporal correlations, its recurrent structure is prone to the vanishing gradient problem when processing long sequences (over 1000 steps), leading to insufficient precision in modeling weekly or monthly cycles. In contrast, Hyper-TS's Transformer architecture uses a self-attention mechanism to capture global dependencies in parallel. Combined with a disentangled temporal representation that separates trend and periodicity components, this gives the model a distinct advantage in fitting long-range periodicity, which is reflected in its lower KDE similarity.

It is noteworthy that the two models have a similar Average KL Divergence (both 0.1544), suggesting that LSTM+RNN remains competitive in fitting single-point distributions. However, Hyper-TS's optimization of overall distribution coverage and kernel density similarity better aligns with the practical needs of cloud workload modeling. After all, resource scheduling and capacity planning do not solely rely on single-point predictions but require a comprehensive understanding of the entire workload distribution.

4 Related Work

In the field of cloud workload generation and prediction, prior research has explored several distinct modeling paradigms. These approaches can be broadly categorized into statistical and machine learning models, recurrent neural networks (RNNs), Transformer models, and more recently, diffusion models. Each paradigm offers unique strengths, but also faces inherent limitations, particularly in addressing the intricate dependencies, temporal complexity, and long-tail distributions of real-world cloud workloads.

Statistical and Machine Learning Models for Trace Generation. Initial approaches to generating and predicting workload traces often relied on statistical and machine learning models. For instance, the method proposed in [31] converts resource consumption data into an integer linear programming problem. However, this method cannot guarantee that the generated resource values are identical to the original dataset's actual values. Other methods, based on Poisson regression and neural sequence models, explicitly model temporal features to generate job traces [32]. These methods typically decouple multi-dimensional features into independent distributions for modeling, which has two core drawbacks: First, these models often make overly strong independence assumptions. They ignore batch dependencies between jobs and resource competition [14].Second, the simplification inherent in their dimensional decoupling fails to capture the intricate, strong correlations between features (e.g., the non-linear relationship between a job's requested resource amount and its runtime). Subsequent methods based on GANs/VAEs, such as TimeGAN [33, 34] and StructuredVAE [35], generate continuous time-series data through adversarial training but still face challenges. They struggle with modeling long-term dependencies and capture the multi-scale seasonality of jobs arrival data (from minute-level bursts to day-level cycles).

Recurrent Neural Networks (RNNs/LSTMs) for Trace Generation. RNNs and LSTMs, they possess a single-stage generation capability, allowing them to control batch size using an End-of-Period (EOP) token and simultaneously generate job types and durations with the use of MADE masks [36]. Additionally, they excel at capturing periodic patterns. In periodic workload prediction, such as 23-hour submission cycles. They have shown superior performance over traditional statistical models [37]. However, they also have significant drawbacks: One major issue is token sensitivity. A ±5-second shift in the EOP token can lead to a batch count error exceeding 30% [38]. They also suffer from a lack of physical constraints. The generated VM durations often violate underlying resource specification limits (e.g., vCPU over-provisioning) [38]. Finally, these models face training instability. They require the introduction of adversarial training or GWO to prevent vanishing gradients, which significantly increases complexity.

Transformer Models for Trace Generation. In Alibaba PAI trace generation, training speed was three times faster than LSTMs, making them particularly suitable for long sequences (>10000 timesteps) [39]. Additionally, multi-head attention explicitly models inter-job dependencies. TimeDiT [40] supports multi-task generation through a unified masking mechanism and imposes constraints to ensure resource rationality. Its bottleneck is high memory overhead. The $O(N^2)$ complexity of the self-attention matrix limits batch size (generating 100000 jobs requires 128GB of VRAM) [39].

Diffusion Models for Time Series Generation. Diffusion models have shown promise in generating high-quality data through a gradual denoising process. Some models are leveraging seasonality-trend decomposition. Diffusion-TS [41] uses Fourier bases to disentangle trend and seasonal components, enhancing the interpretability of multivariate time-series generation. A key innovation is multi-resolution modeling. Mr-Diff [42] employs hierarchical diffusion, first generating a coarse-grained trend and then refining high-frequency components. MG-TSD [43] and RATD [44] use a multi-granularity objective to improve generation consistency.

5 Conclusion

This paper successfully addresses the challenges of cloud workload modeling by proposing and validating the Hyper-TS framework for generating realistic and controllable cloud workloads. Our approach combines a Poisson regression model for efficient batch arrival modeling with a powerful conditional diffusion model for generating complex, multi-dimensional attributes. Experimental results demonstrate that Hyper-TS consistently outperforms existing baselines, proving its effectiveness and laying a solid foundation for future research in cloud resource management and system simulation.

While Hyper-TS achieves state-of-the-art performance, several avenues exist for future exploration. The framework's modularity allows for the integration of more advanced components, such as a more robust anomaly detection module to better capture and generate sudden, extreme spikes caused by events like server failures.

References

[1] Xiaodi Ke, Cong Guo, Siqi Ji, Shane Bergsma, Zhenhua Hu, and Lei Guo. Fundy: A scalable and extensible resource manager for cloud resources. In *2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*, pages 540–550. IEEE, 2021.

[2] Ori Hadary, Luke Marshall, Ishai Menache, Abhisek Pan, Esaias E Greeff, David Dion, Star Dorminey, Shailesh Joshi, Yang Chen, Mark Russinovich, and Thomas Moscibroda. Protean: VM allocation service at scale. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 845–861. USENIX Association, November 2020.

[3] Ismael Solis Moreno, Peter Garraghan, Paul Townend, and Jie Xu. Analysis, modeling and simulation of workload patterns in a large-scale utility cloud. *IEEE Transactions on Cloud Computing*, 2(2):208–221, 2014.

[4] Chengzhi Lu, Kejiang Ye, Guoyao Xu, Cheng-Zhong Xu, and Tongxin Bai. Imbalance in the cloud: An analysis on alibaba cluster trace. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 2884–2892. IEEE, 2017.

[5] Charles Reiss, John Wilkes, and Joseph L Hellerstein. Google cluster-usage traces: format+ schema. *Google Inc., White Paper*, 1:1–14, 2011.

[6] Maria Carla Calzarossa, Luisa Massari, and Daniele Tessera. Workload characterization: A survey revisited. *ACM Computing Surveys (CSUR)*, 48(3):1–43, 2016.

[7] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, page 153–167, New York, NY, USA, 2017. Association for Computing Machinery.

[8] Gilles Madi Wamba, Yunbo Li, Anne-Cécile Orgerie, Nicolas Beldiceanu, and Jean-Marc Menaud. Cloud workload prediction and generation models. In *2017 29Th international symposium on computer architecture and high performance computing (SBAC-PAD)*, pages 89–96. IEEE, 2017.

[9] Furkan Koltuk and Ece Güran Schmidt. A novel method for the synthetic generation of non-iid workloads for cloud data centers. In *2020 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6. IEEE, 2020.

[10] Ori Hadary, Luke Marshall, Ishai Menache, Abhisek Pan, Esaias E Greeff, David Dion, Star Dorminey, Shailesh Joshi, Yang Chen, Mark Russinovich, et al. Protean:{VM} allocation service at scale. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 845–861, 2020.

[11] Luiz André Barroso, Urs Hölzle, and Parthasarathy Ranganathan. *The datacenter as a computer: Designing warehouse-scale machines*. Springer Nature, 2019.

[12] Abhishek Verma, Madhukar Korupolu, and John Wilkes. Evaluating job packing in warehouse-scale computing. In *2014 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 48–56. IEEE, 2014.

[13] Charles Reiss, Alexey Tumanov, Gregory R Ganger, Randy H Katz, and Michael A Kozuch. Heterogeneity and dynamicity of clouds at scale: Google trace analysis. In *Proceedings of the third ACM symposium on cloud computing*, pages 1–13, 2012.

[14] Wei Gao, Xu Zhang, Shan Huang, Shangwei Guo, Peng Sun, Yonggang Wen, and Tianwei Zhang. Autosched: An adaptive self-configured framework for scheduling deep learning training workloads. In *Proceedings of the 38th ACM International Conference on Supercomputing*, pages 473–484, 2024.

[15] Shane Bergsma, Timothy Zeyl, Arik Senderovich, and J Christopher Beck. Generating complex, realistic cloud workloads using recurrent neural networks. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*, pages 376–391, 2021.

[16] Inbar Huberman-Spiegelglas, Vladimir Kulikov, and Tomer Michaeli. An edit friendly ddpm noise space: Inversion and manipulations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12469–12478, 2024.

[17] Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. *Advances in neural information processing systems*, 35:4328–4343, 2022.

[18] Lifeng Shen and James Kwok. Non-autoregressive conditional diffusion models for time series prediction. In *International Conference on Machine Learning*, pages 31016–31029. PMLR, 2023.

[19] Junliang Chen, Chen Wang, Bing Bing Zhou, Lei Sun, Young Choon Lee, and Albert Y Zomaya. Tradeoffs between profit and customer satisfaction for service provisioning in the cloud. In *Proceedings of the 20th international symposium on High performance distributed computing*, pages 229–238, 2011.

[20] Javad Ghaderi. Randomized algorithms for scheduling vms in the cloud. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9. IEEE, 2016.

[21] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[22] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[23] Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437*, 2019.

[24] Abhyuday Desai, Cynthia Freeman, Zuhui Wang, and Ian Beaver. Timevae: A variational auto-encoder for multivariate time series generation. *arXiv preprint arXiv:2111.08095*, 2021.

[25] Alysha M De Livera, Rob J Hyndman, and Ralph D Snyder. Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American statistical association*, 106(496):1513–1527, 2011.

[26] Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven Hoi. Etsformer: Exponential smoothing transformers for time-series forecasting. *arXiv preprint arXiv:2202.01381*, 2022.

[27] Kan Ren, Jiarui Qin, Lei Zheng, Zhengyu Yang, Weinan Zhang, Lin Qiu, and Yong Yu. Deep recurrent survival analysis. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4798–4805, 2019.

[28] Håvard Kvamme and Ørnulf Borgan. Continuous and discrete-time survival prediction with neural networks. *Lifetime data analysis*, 27(4):710–736, 2021.

[29] Edward L Kaplan and Paul Meier. Nonparametric estimation from incomplete observations. *Journal of the American statistical association*, 53(282):457–481, 1958.

[30] Siva Theja Maguluri and Rayadurgam Srikant. Scheduling jobs with unknown duration in clouds. *IEEE/ACM Transactions On Networking*, 22(6):1938–1951, 2013.

[31] Syed M. Iqbal, Haley Li, Shane Bergsma, Ivan Beschastnikh, and Alan J. Hu. Cospot: a cooperative vm allocation framework for increased revenue from spot instances. In *Proceedings of the 13th Symposium on Cloud Computing*, SoCC '22, page 540–556, New York, NY, USA, 2022. Association for Computing Machinery.

[32] Zechun Zhou, Jingwei Sun, and Guangzhong Sun. Automated hpc workload generation combining statistical modeling and autoregressive analysis. In *International Symposium on Benchmarking, Measuring and Optimization*, pages 153–170. Springer, 2023.

[33] Jinsung Yoon, Daniel Jarrett, and Mihaela Van der Schaar. Time-series generative adversarial networks. *Advances in neural information processing systems*, 32, 2019.

[34] Zinan Lin, Alankar Jain, Chen Wang, Giulia Fanti, and Vyas Sekar. Generating high-fidelity, synthetic time series datasets with doppelganger. *arXiv preprint arXiv:1909.13403*, 2019.

[35] Yixiu Zhao and Scott Linderman. Revisiting structured variational autoencoders. In *International Conference on Machine Learning*, pages 42046–42057. PMLR, 2023.

[36] Cong Li and Renxiang Lu. Short-term power forecasting model based on gwo-lstm network. In *Journal of Physics: Conference Series*, volume 2503, page 012039. IOP Publishing, 2023.

[37] Shivani Arbat, Vinodh Kumaran Jayakumar, Jaewoo Lee, Wei Wang, and In Kee Kim. Wasserstein adversarial transformer for cloud workload prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 12433–12439, 2022.

[38] Shane Bergsma, Timothy Zeyl, Arik Senderovich, and J Christopher Beck. Generating complex, realistic cloud workloads using recurrent neural networks. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*, pages 376–391, 2021.

[39] Yi Liang, Nianyi Ruan, Lan Yi, and Xing Su. An approach to workload generation for modern data centers: A view from alibaba trace. *BenchCouncil Transactions on Benchmarks, Standards and Evaluations*, 4(1):100164, 2024.

[40] Defu Cao, Wen Ye, Yizhou Zhang, and Yan Liu. Timedit: General-purpose diffusion transformers for time series foundation model. *arXiv preprint arXiv:2409.02322*, 2024.

[41] Xinyu Yuan and Yan Qiao. Diffusion-ts: Interpretable diffusion for general time series generation. *arXiv preprint arXiv:2403.01742*, 2024.

[42] Lifeng Shen, Weiyu Chen, and James Kwok. Multi-resolution diffusion models for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024.

[43] Xinyao Fan, Yueying Wu, Chang Xu, Yuhao Huang, Weiqing Liu, and Jiang Bian. Mg-tsd: Multi-granularity time series diffusion models with guided learning process. *arXiv preprint arXiv:2403.05751*, 2024.

[44] Jingwei Liu, Ling Yang, Hongyan Li, and Shenda Hong. Retrieval-augmented diffusion models for time series forecasting. *Advances in Neural Information Processing Systems*, 37:2766–2786, 2024.

[45] Elizabeth Fons, Alejandro Sztrajman, Yousef El-Laham, Alexandros Iosifidis, and Svitlana Vyetrenko. Hypertime: Implicit neural representation for time series. *arXiv preprint arXiv:2208.05836*, 2022.

[46] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

A APPENDIX

A.1 Model Training

For batch arrival prediction, we train the Poisson regression model using iterative reweighted least squares. This method iteratively optimizes weights to accommodate the variance-mean equality property of the Poisson distribution, making it more suitable for count data than ordinary least squares. This effectively captures the dynamic changes in arrival rates over time, such as the surge in batches during peak hours.

The model's loss function is composed of a reconstruction loss ($\mathcal{L}_{\text{recon}}$) and a Fourier loss ($\mathcal{L}_{\text{fft}}$). This design directly addresses the core challenge of cloud workload data, which exhibits both regular periodicity and irregular fluctuations:

Reconstruction Loss $\mathcal{L}_{\text{recon}} = \|x_0 - \hat{x}_0\|_2^2$: This constrains the generated samples to have pixel-level similarity to real samples in the time domain, ensuring the model captures the overall trend and numerical accuracy of the workload. It also addresses the lack of interpretability in traditional diffusion models, which only predict noise. Hyper-TS directly predicts clean samples, which aligns better with the goal of interpretable time-series decomposition (e.g., explicit modeling of trend and seasonality).

Fourier Loss $\mathcal{L}_{\text{fft}} = \|\mathcal{FFT}(x_0) - \mathcal{FFT}(\hat{x}_0)\|_2^2$: This strengthens the model's ability to capture periodic components by imposing a constraint in the frequency domain [45]. The significant daily/weekly cycles in cloud workloads correspond to specific frequency components in the frequency domain. This loss forces the model to preserve the amplitude and phase of these components, solving the problem where time-domain loss struggles to capture long-range periodic dependencies.

We introduce a weight $w_t = \frac{\lambda \alpha_t (1-\bar{\alpha}_t)}{\beta_t^2}$ to the diffusion process [46], giving higher weight to stages with high noise (large diffusion steps). Here, $\alpha_t = 1 - \beta_t$ is the noise variance at step t (forward process parameter), and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$ represents the cumulative variance decay over the first t steps. This strategy balances the training difficulty across different diffusion stages, ensuring the model can learn data structure even in the presence of strong noise, which is particularly important for cloud workload data with sudden fluctuations.

To validate the statistical significance of the reported performance advantages, we conducted 95% confidence interval (CI) analysis on the key metric of p05-p95 coverage. The 95% CI for the coverage metric was calculated using a standard non-parametric bootstrapping method with 1,000 resamples.The non-overlapping nature of the 95% confidence intervals between the proposed Hyper-TS model and the best-performing baseline (LSTM+RNN) demonstrates that the observed advantage in p05-p95 coverage is statistically significant, confirming the robustness of our model's generative capability.

All models were trained and evaluated on a system equipped with an NVIDIA GeForce RTX 4090 GPU. The total training time for the proposed Hyper-TS model was approximately 48 hours. For the purpose of reproducibility and promoting open research, the source code for the Hyper-TS framework, along with pre-processing scripts and model configuration files, will be made publicly available upon publication. The repository will be hosted on GitHub.

Table 4: Impact of the Fourier-Based Loss Term:

Model	Std	PACF	KL
Hyper-TS(Without Fourier-Layer)	0.2941	0.0049	0.2051
Hyper-TS(Full Model)	**0.2409**	**0.0009**	**0.1544**

A.2 Model Hyperparameters

The hyperparameters for the Diffusion-TS component of Hyper-TS were validated through multiple rounds of experiments to balance model performance with computational efficiency, while specifically enhancing its ability to capture cloud workload features.

Experiments comparing 200, 500, and 1000 time steps showed that 500 steps achieves the optimal balance between generation quality (e.g., KDE similarity) and training efficiency, allowing for a sufficient simulation of the noise addition and reconstruction processes without excessive computational cost.

Grid search validated this configuration's effectiveness in capturing multi-scale dependencies in cloud workloads (e.g., short-term job correlations and long-term periodic trends). Too many heads lead to feature fragmentation, while too few make it difficult to model complex correlations.

The encoder only requires one layer to extract global features, while the decoder is increased to two layers to enhance its ability to decompose trend and periodicity terms, which is consistent with the "disentangled temporal representation" design goal. Determined through ablation studies, this weight ensures the frequency-domain constraint complements the time-domain reconstruction loss. An excessively high weight would lead to overfitting to periodicity and neglecting random fluctuations, while a weight that is too low would fail to effectively capture seasonality. Compared to a linear schedule, a cosine schedule slowly increases noise in the early diffusion stages and accelerates later on. This better matches the distribution of cloud workloads, which are predominantly stable with sudden fluctuations. This allows the model to preserve overall trends while learning detailed fluctuations.

The experimental results provided empirical support for the rationality of our model design, demonstrating that the Transformer architecture and direct sample prediction are crucial for capturing long-sequence dependencies.

A.3 Ablation Study

To validate the contribution of the proposed "Fourier-based loss term" ($\mathcal{L}_{\text{Fourier}}$), we conducted an ablation study and found that its removal resulted in a clear degradation across key performance metrics.

Specifically, as shown in Figure 4, the standard deviation difference worsened from 0.2409 to 0.2941, indicating less realistic sample diversity, while the fidelity of temporal dependencies, measured by "PACF Similarity", significantly increased from 0.0009 to 0.0049. Furthermore, the increase in "Average KL Divergence" from 0.1544 to 0.2051 confirms that the $\mathcal{L}_{\text{Fourier}}$ is essential for maintaining the overall distributional realism and capturing the statistical properties of the generated cloud workload time series.