This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2025.3640771

Analyzing Request Volatility in Cloud-based Machine Learning: Insights from Alibaba's Machine Learning as a Service Platform

Qiang Zou, Yuhui Deng, Yifeng Zhu, Yi Zhou, Jianghe Cai, Shuibing He, and Lina Ge

***Abstract*—With advancements in machine learning (ML) technology and the deployment of large ML-as-a-Service (MLaaS) clouds, accurately understanding request behaviors in an MLaaS cloud platform is paramount for resource scheduling and optimization. This paper sheds light on the correlation of request arrivals in a representative and dynamic MLaaS workload – Alibaba PAI (an ML platform for artificial intelligence). For requests in the PAI workloads at the job, task, instance, and machine levels, our burstiness diagnosis reveals that the request arrival processes at all levels are significantly bursty. Additionally, our Gaussianity test indicates that the bursty activities in PAI consistently appear to be non-Gaussian. Our findings show that there exists a certain degree of correlation between request arrivals at each level over long-term time scales. Moreover, we reveal the self-similar nature of request activities in the various-level wild MLaaS workloads on Alibaba PAI through visual evidence, the auto-correlation structure of the aggregated process of request sequences, and Hurst parameter estimates. Furthermore, we implement a versatile workload synthetic model to synthesize request series based on the inputs measured from the PAI trace. Experimental results demonstrate that our model outperforms typical self-similar workload models, and can improve accuracy by up to 99% compared to them.**

***Index Terms*—Machine learning cloud platform, workload analysis, burst, heavy-tailed, self-similarity, synthetic model.**

I. Introduction

Qiang Zou is with the School of Artificial Intelligence, Guangxi Minzu University, Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis, Nanning 530006, P. R. China.
E-mail: qzou@gxmzu.edu.cn
Yuhui Deng is with the Department of Computer Science, Jinan University, Guangzhou 510632, P. R. China.
E-mail: tyhdeng@jnu.edu.cn
Yifeng Zhu is with the Department of Electrical and Computer Engineering, University of Maine, Orono, ME 04469, USA.
E-mail: yifeng.zhu@maine.edu
Yi Zhou is with the TSYS School of Computer Science, Columbus State University, Columbus, GA.
E-mail: zhou_yi@columbusstate.edu
Jianghe Cai is with the Department of Computer Science, Jinan University, Guangzhou 510632, P. R. China.
E-mail: 761571151@qq.com
Shuibing He is with the College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, P. R. China.
E-mail: heshuibing@zju.edu.cn
Lina Ge is with the School of Artificial Intelligence, Guangxi Minzu University, Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis, Nanning 530006, P. R. China.
E-mail: gelina@gxmzu.edu.cn

This paper was produced by the IEEE Publication Technology Group. They are in Piscataway, NJ.
Manuscript received July XX, 2024; revised January XX, 2025.

WITH the sustained advances in ML technology, deep learning (DL) methods are being widely used by a growing number of users in various scientific fields to drive scientific discovery and innovation, such as auto-driving cars, drug development, and image processing. To accommodate the computing demands of rapidly growing ML workloads, large tech firms are channeling investments in ML-as-a-service (MLaaS) clouds outfitted with costly hardware including GPUs to execute various types of ML workloads, such as training and inference [1]. Optimizing resource scheduling and management of GPU clusters requires insight into job characteristics and user behaviors in ML application workloads [2]. A useful and judicious first step in understanding arrival characteristics is to analyze real workloads and employ corresponding stochastic models to effectively characterize the workload [3].

Recently, the research community adopted traditional distributions, such as uniform [4] and exponential [5], to approximate the description of request features in computation-intensive ML workloads. Meanwhile, ML researchers found that burst phenomena are prevalent in both ML and DL workloads [1], [2], [6], which even show a heavy-tailed feature [1], [7]. However, the traditional distributions mentioned above fail to specialize in capturing the burst behavior, which brings many challenges to accurately understanding request behaviors in ML application workloads.

Motivated by these observations, we analyze the workload traces [1] gleaned from the ML Platform for Artificial Intelligence (PAI) – an integrative MLaaS platform offered by Alibaba cloud. The PAI workloads consist of both training and inference tasks executing cutting-edge ML algorithms. For the PAI workloads at the levels of job, task, instance, and machine, we characterize request burstiness, conduct the Gaussianity tests [8] for request sequences, and diagnose the correlation of arrival patterns. For the PAI workloads with certain degrees of correlations, this paper presents visual and theoretical evidence for the existence of self-similarity and estimates the self-similarity parameters using statistical tools. Based on the inputs measured from the PAI traces, we further employ a versatile synthetic model to faithfully synthesize request sequences in the PAI workloads at various levels, respectively. To the best of our knowledge, few research works on this topic have been reported in the literature.

In short, this paper makes the following four contributions:

- For the requests in the PAI workloads at the levels of job, task, instance, and machine, we characterize the

Authorized licensed use limited to: Zhejiang Lab. Downloaded on January 04,2026 at 01:40:45 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2025.3640771

burstiness and reveal that request arrival processes are significantly bursty. Furthermore, we study the correlation of request arrivals. Our findings uncover that there is a certain degree of correlation between request arrivals. Moreover, for the requests in the job-level and task-level workloads, both the correlations among request arrivals and the measurements for the burstiness of requests are relatively close, respectively, which may be caused by the fact that most jobs in PAI have only one task [1].

- We perform the Gaussianity tests for the request sequences in the PAI workloads at various levels. The testing results show that the burst activities in the PAI workloads appear to be non-Gaussian. This observation is surprising because it differs significantly from previous studies using Gaussian methods to analyze Alibaba production cluster data [5]. Therefore, our findings help to faithfully describe the burst behaviors in PAI.
- We unearth the self-similar nature of request activities in the PAI workloads at various levels through visual evidence, the auto-correlation structure of an aggregated process of request sequences, and Hurst parameter [8] estimates.
- We employ a versatile request generator based on the inputs measured from the Alibaba PAI traces to generate synthetic request sequences for PAI at the levels of job, task, instance, and machine. Experimental results demonstrate that our model outperforms existing models in terms of accuracy in emulating burstiness and heavy-tailed property.

The rest of this paper is organized as follows. Section II gives an overview of the Alibaba PAI traces investigated throughout this work and summarizes related research studies. Section III characterizes the burstiness of the request arrival process in the PAI workload at each level, performs the Gaussianity tests for the request sequences in the PAI workloads at various levels, and elaborates on the correlation of request arrivals in the four-level PAI workloads. Section IV presents both visual and statistical evidence for the existence of self-similarity in the PAI workloads. Section V articulates the implementation of a request series generator to synthesize request series for PAI. Section VI discusses the importance of gaining insights into the workload characteristics of ML applications. Section VII concludes this paper.

II. Background and Motivation

A. Alibaba PAI

In machine learning cloud platforms, jobs, tasks, instances, and machines are the core concepts for building and managing machine learning workflows. A job is a complete computing process performed on a machine learning cloud platform, usually corresponding to an end-to-end machine learning workflow, such as model training, inference, or data processing. Each job is decomposed into one or more tasks that are executed sequentially or in parallel. A task is a specific execution unit within a job, corresponding to a stage in the machine learning workflow, such as data cleaning, feature extraction, model training or evaluation. An instance is a unit of computing resources allocated by the cloud platform, usually referring to a virtual machine or containerized environment. Tasks are typically bound to specific computing resources (instances) for execution. Machines are the physical or virtualized hardware resources that provide computing power at the underlying layer of the cloud platform, such as GPU clusters, or CPU servers. Each task may consist of one or multiple instances and can run on multiple machines.

Taking the TensorFlow framework [9] as an example, users define computational graphs (jobs) on the client side, which include variables, operators, and data flow relationships. These jobs are submitted to the distributed master node via a Session. The master node optimizes the job (e.g., operator fusion, constant folding) and splits the logical computational graph into multiple subgraphs based on device topology and parallelization strategies (data parallelism/model parallelism). Subsequently, the master node assigns the optimized subgraphs to different worker nodes, with each subgraph corresponding to a task. The task can then be mapped to devices, such as CPU or GPU, forming a physical execution plan.

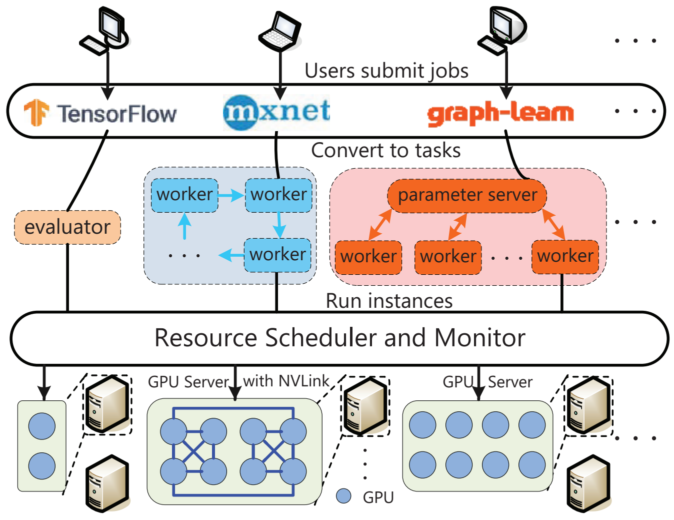


Fig. 1. PAI architecture overview.

MLaaS packages the entire machine learning workflow, including data preprocessing, model training, inference, and more, into standardized cloud services, lowering technical barriers through elastic computing power, automated tools, and pre-built algorithm libraries. To enable developers to use ML technology flexibly and efficiently, Alibaba Cloud launched the ML Platform for Artificial Intelligence (PAI) – a large production cluster comprising over 6,500 GPUs across 1,800 machines – to provide a variety of services spanning the whole ML pipeline (see Figure 1 for an architectural overview). In PAI, users send ML jobs, including training and inference, which are developed in various frameworks, including TensorFlow, MXNet [10], and Graph-Learn [11], and provide application code. Meanwhile, users specify the computational resources required, such as GPUs, CPUs, and memory. Each job is then split into multiple tasks with different compute roles: parameter server (PS), workers for training jobs, and estimators for inference jobs. To simplify

Authorized licensed use limited to: Zhejiang Lab. Downloaded on January 04,2026 at 01:40:45 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2025.3640771

scheduling and execution on heterogeneous hardware, Docker containers are employed to instantiate tasks in PAI. Each task consists of many instances that run in Docker containers. Due to cost considerations, only some subclusters in PAI are fitted with NVLink in multi-GPU servers.

As the core platform of Alibaba's MLaaS, PAI empowers enterprises to rapidly build and optimize predictive models, covering multiple application scenarios, such as e-commerce, product recommendation, and financial. For instance, in product recommendation, PAI leverages collaborative filtering and deep interest network models to deliver millisecond-level response recommendations, dynamically generating personalized product suggestions (e.g., Taobao homepage Feed stream) based on users' real-time behaviors. Additionally, PAI's pre-built algorithms reduce the model deployment cycle from the traditional 2-3 weeks to within 3 days. Since its deployment, PAI has attracted tens of thousands of enterprises and individual developers, becoming one of the leading MLaaS platforms in China.

To address the challenges of load imbalance and long queuing delays across heterogeneous machines caused by cluster scheduling, Weng *et al.* [1] ran training and inference jobs using state-of-the-art ML algorithms on PAI in the second half of 2020, and collected application workload traces for two months. The trace files analyzed in this work, include *pai_job_table* (job launch information), *pai_task_table* (task launch information), *pai_instance_table* (instance launch information), and *pai_machine_metric* (machine resource metrics with respect to the instance). Taking â*pai_job_table*â?as an example, each row represents the information corresponding to one request, including timestamp, user name, name of users' submit jobs, job status, etc. A detailed description of the PAI traces can be found in the referenced literature [1].

Table I briefly summarizes the PAI trace, in which the V100M32 and V100 GPUs are equipped with NVLink. The traces record timestamps, resource requests, and usage at the levels of job, task, instance, and machine. The traces at the job, task, and instance levels include the corresponding launch information, while the machine-level trace contains machine resource metrics related to the instance. Most jobs in the traces have only one task, but some may launch multiple tasks with different names. However, since our study aims at studying the arrival process of requests from a temporal perspective, we only use timestamp information (in seconds for about two months), without considering other information.

TABLE I
SUMMARY OF PAI TRACE AND MACHINE SPECS OF GPU CLUSTERS [1].

#Machines	1800		**Duration**	2 months	
Memory (GiB)	512	512	512	384	512/384
GPU type	P100	T4	Misc.	V100M32	V100
#GPUs	2	2	8	8	8
#Nodes	798	497	280	135	104

Previous studies on the PAI traces have shown that request behaviors in ML application workloads are bursty and exhibit heavy-tailed features [1]. In genereal, request bursts often exist on different time scales, and traditional methods (e.g., exponential) will gradually become smooth at long time scales, making them difficult to accurately describe the burst behaviors in request activities [12], posing significant challenges to the accurate understanding of request activities in typical ML workloads, as well as to their scheduling and optimization.

These observations have inspired us to examine the feasibility and effectiveness of using traditional methods such as uniform [4] and exponential [5] distributions to describe request behaviors in representative MLaaS workloads. In other words, we must consider the following critical issues: Is it appropriate to use independently and identically distributed (IID) methods to describe the bursts and heavy-tailed behaviors in MLaaS workloads? Whether do the request activities in MLaaS workloads also present the self-similarity observed in other workloads [13], [14]? To find the answer, we revisit the PAI workloads at various levels, and diagnose the temporal behaviors in Section III.

B. Related Work

At present, the research community is making significant efforts to guide cluster design based on characterizing ML application workloads. In this section, we focus on previous works most relevant to this study.

1) Workload Characterization: Currently, the research community often employs traditional methods to approximately characterize request behaviors in ML workloads. For instance, Li *et al.* [15] assumed that requests for each model follow independent Poisson processes, thereby modeling the service of individual GPUs as an M/D/1 queue. Based on this, they conducted queuing theory analyses to derive analytical expressions for critical metrics such as the average number of requests and average latency. Additionally, Zhang *et al.* [16] hypothesized that job arrivals adhere to a Poisson distribution and generated corresponding job arrival processes by parameterizing them using publicly available GPU trace data from Helios, aiming to simulate request behaviors in production clusters.

Recently, researchers have discovered that request behaviors in ML workloads exhibit bursty or heavy-tailed characteristics. For instance, while evaluating their proposed GPU cluster task scheduling optimization solution, *IADeep*, Chen *et al.* [17] noted that the completion times of small DL tasks in the baseline demonstrated heavy-tailed properties. Additionally, Saxena *et al.* [18] generated a long-duration (up to 12 hours) workload featuring bursty arrivals of DL jobs based on a Poisson distribution parameterized by the average job arrival rate, claiming it reflects real-world DL job arrival scenarios. However, the Poisson distribution is not well-suited for capturing burstiness [19].

2) Self-similarity: In nature, many phenomena exhibit self-similarity, such as a fern plant, where a part resembles the whole or other parts. This characteristic is referred to as self-similarity. For time series, self-similarity implies that the attributes of a given process remain consistent across different time scales. Recently, researchers have been examining self-similarity in various domains, such as network traffic [20], cloud workloads [13], social network dynamics [21], and internet traffic [14], to accurately characterize their respective workloads. For instance, Li *et al.* [20] observed that normal

Authorized licensed use limited to: Zhejiang Lab. Downloaded on January 04,2026 at 01:40:45 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2025.3640771

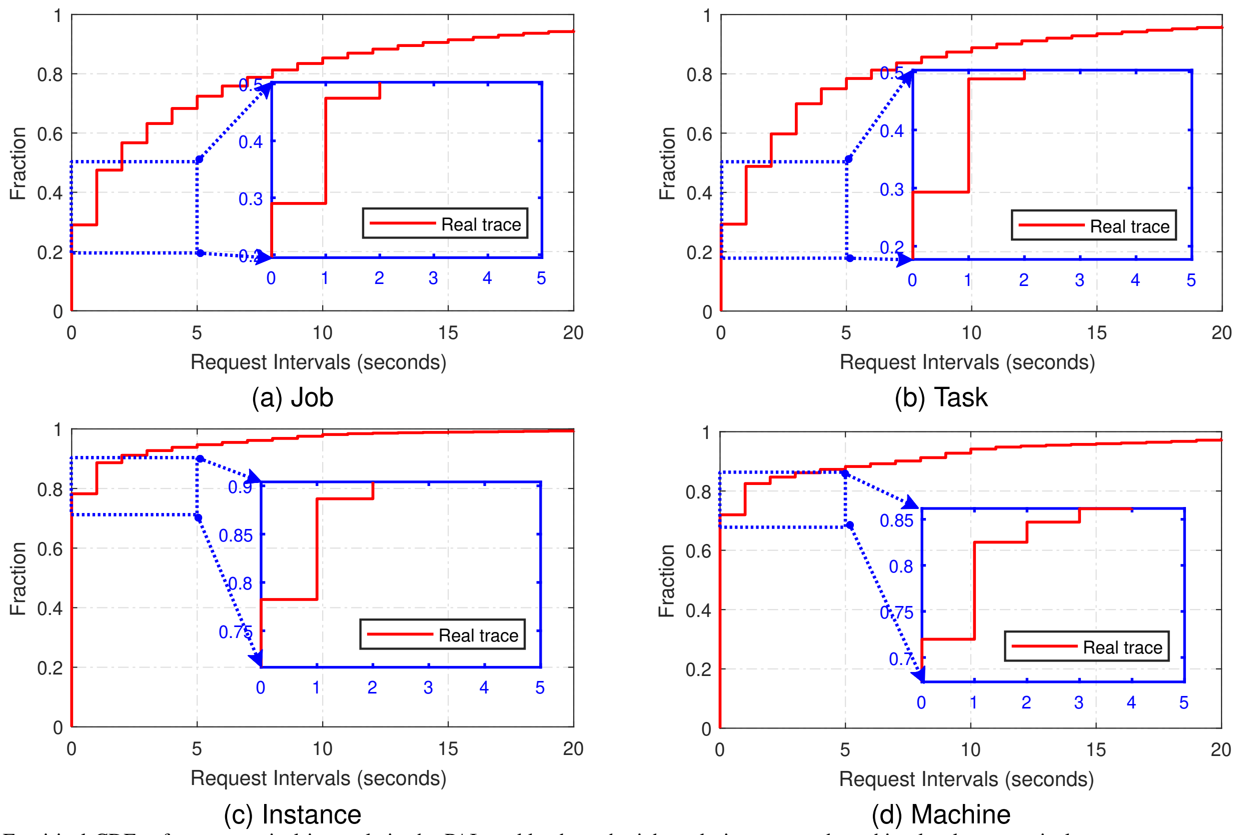


Fig. 2. Empirical CDFs of request arrival intervals in the PAI workloads at the job, task, instance and machine levels, respectively.

OpenFlow traffic has a low self-similarity degree whereas the occurrences of saturation attacks typically imply a higher degree of self-similarity measured by Hurst exponents. Then they exploited the self-similarity degrees of OpenFlow traffic for anomaly detection. Gupta *et al.* [13] studied Google cluster traces and found self-similarity and heavy-tailed behaviors in cloud workloads using auto-correlation analysis and rescaled adjusted range (R/S) analysis. Liu *et al.* [21] analyzed traces from Renren social networks and Facebook, and found self-similarity in the edge creation process of networks. Similarly, Li *et al.* [14] demonstrated self-similarity in industrial internet traffic. Furthermore, Talluri *et al.* [22] conducted a statistical analysis of file popularity, read size, arrival interval, reuse time, etc. using Spark data collected over six months. Their findings showed that read operations exhibit heavy tails, bursts, and negative long-range dependence.

C. Motivation

Although Weng *et al.* [1] observed that the request behaviors in the PAI workloads are bursty and even exhibit heavy-tailed properties, they did not further investigate or characterize the burstiness of request behaviors in the PAI workloads at various levels. However, accurately characterizing burstiness, as an important workload characteristic, is crucial for both resource scheduling and system tuning. This paper aims to fill this gap by providing an in-depth analysis. Furthermore, given the distinctive access mode of machine learning platforms [6], these observations prompt us to investigate whether self-similarity exists in the PAI workloads.

Anchored on the real-world wild PAI workloads from Alibaba, timestamps are extracted from the four trace files (*i.e.*, job, task, instance, and machine). Due to the fact that the original trace files are not strictly recorded in chronological sequence during the collection or storage of trace information, the extracted timestamp are reordered chronologically, then converted to the time intervals between two adjacent requests (*i.e.*, inter-arrival times), and the number of requests per second. No outliers or noise were removed, ensuring that trace faithfully reflects real-world variability. This work focuses on the characterization of request burstiness, and the appropriateness of describing request burst behaviors with traditional distributions. A second focus of this study is the presence of self-similarity. The third aspect of this work is the synthesis of the request sequence.

III. PAI Workload Diagnosis

In this section, we provide an in-depth understanding of the request activities in typical DL applications by diagnosing four real levels of Alibaba PAI workloads, namely, job, task, instance, and machine.

A. Burstiness

To deeply understand the access features in system workloads, a useful perspective is to dissect the arrival mode. For a series of access requests in system workloads, the workload can be considered a burst workload if the corresponding request arrival process X is *non-stationary* and has a large

Authorized licensed use limited to: Zhejiang Lab. Downloaded on January 04,2026 at 01:40:45 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2025.3640771

variance. Previous works have observed that "burstiness" exists widely in the workloads of various systems, such as network [23]–[25], mobile storage [26], and cloud block storage [27].

To characterize request burstiness in the PAI workloads at the levels of job, task, instance and machine, this work extracts the request arrival intervals (RAI) from the corresponding PAI traces and explores the corresponding empirical distribution rules. To visually show the features that exist in the arrival pattern, we present the *cumulative distribution function* (CDF) of the request arrival intervals in Figure 2. In Figure 2, the horizontal axis denotes request arrival intervals in seconds, and the vertical axis represents the proportion of the corresponding value in the whole. A point (x, y) in the distribution curve, indicates that the proportion of request arrival intervals less than or equal to x in the corresponding time series is y.

For each subplot within Figure 2, we have magnified the local part involved in the distribution curve, to clearly display the important observations in the empirical distribution curve, as shown in the blue box within each subplot. As can be seen from Figure 2, for the PAI workload at the levels of job, task, instance, and machine, about 48%, 49%, 88% and 83% of requests arrive within an interval of no more than 1 second, respectively, and even up to 29%, 30%, 78% and 72% of request arrival intervals are 0 second, respectively. The aforementioned gauges are summarized in Table II.

Non-stationary. These aforementioned observations reveal that most requests in the PAI workloads at the levels of job, task, instance, and machine, arrive intensively at some hotspot moments, thus causing the aggregation effect of requests, which makes the entire request arrival process non-stationary.

Variance. For the PAI workloads at various levels, as shown in Table II, the statistical findings show that the variances of the corresponding request arrival intervals are as high as 4896.4, 4090.5, 1312.2, and 8892.0, respectively, satisfying another necessary condition to become a burst workload. Therefore, the concept of "burstiness" can be used to describe the high variability of the request arrival process in the PAI workloads at all levels.

TABLE II
SUMMARY OF MEASURES FOR THE REQUEST BURSTINESS IN THE PAI WORKLOADS AT THE LEVELS OF JOB, TASK, INSTANCE AND MACHINE.

Trace level	Empirical study		Variance	CV	IDI
	RAI ≤ 1	RAI $= 0$			
Job	$\approx 48\%$	$\approx 29\%$	4896.4	12.356	152.67
Task	$\approx 49\%$	$\approx 30\%$	4090.5	13.500	182.25
Instance	$\approx 88\%$	$\approx 78\%$	1312.2	34.958	1222.1
Machine	$\approx 83\%$	$\approx 72\%$	8892.0	38.973	1518.8

To strengthen the reliability of observations regarding burstiness, we utilize the coefficient of variation (CV) [28] to assess the variability of arrival intervals. For a time seires, the CV is defined as the quotient of standard deviation and mean. A CV greater than 1 indicates high variability. For the request arrival intervals in the PAI workloads at the job, task, instance, and machine levels, we calculated the CVs as 12.356, 13.500, 34.958, and 38.973, respectively, as shown in Table II. All CV values are much greater than 1, thereby confirming the burst nature of requests within the PAI workloads.

Previous works have suggested that the strength of burstiness can be quantified by the *index of dispersion* [29]. The larger the value of the index of dispersion, the higher the strength of burstiness. For the PAI workload at each level, this section utilizes the *index of dispersion for intervals* (IDI) proposed in the literature [29], to measure the corresponding strength of burstiness. For a time seires of arrival intervals, X, the IDI is defined as:

$$IDI = \frac{Var[X]}{E^2[X]}, \tag{1}$$

where the numerator represents the variance of X, and the denominator is the square of the mean of X. By calculating Equantion (1), we obtain the indices of dispersion for request arrival intervals in the PAI workload at various levels as 152.67, 182.25, 1222.1, and 1518.8, respectively, as shown in Table II. This finding indicates that request arrivals in the PAI workloads at various levels consistently present a significant bursty nature, especially for the instance and machine-level PAI workloads.

Upon examining Table II, the indices of dispersion and the empirical studies of the request intervals suggest that the burstiness of requests at the instance and machine levels seems to be stronger than that at the job and task levels. Moreover, we observe that the measurements for the burstiness of requests in the job and task-level workloads are relatively close, which may be due to the fact that most jobs in PAI have only one task [1].

The burstiness of job-level requests leads to resource contention and queuing delays under traditional static allocation strategies. To mitigate these issues, priority scheduling and preemption mechanisms should be adopted to enable dynamic resource release. For instance, deep learning algorithms can be utilized to evaluate task priorities and dynamically adjust resource allocation strategies in real-time. When the number of tasks within a single job increases significantly, it becomes essential to dynamically partition the computing unit and optimize parallel scheduling. Systems like Flink reduce redundant computation overhead through intermediate state caching for computed results, while distributed architectures enhance throughput by coordinating node-level collaborative computations. For bursty instance-level requests, load rebalancing can be achieved by integrating hot migration technology with energy consumption monitoring. Furthermore, research on the burstiness of machine-level requests supports dynamic cache management and capacity planning for burst buffers, as well as the design of adaptive load-balancing algorithms to improve overall system performance and response efficiency.

Despite its practical value, the index of dispersion has limitations. A notable limitation is its sensitivity to outliers; extreme values can disproportionately affect variance, causing the index of dispersion to be artificially inflated and leading to misleading results. Moreover, the index of dispersion default that the mean is a valid measure of central tendency, but this assumption may not hold in skewed distributions.

Now, let us recap this major observation as

Authorized licensed use limited to: Zhejiang Lab. Downloaded on January 04,2026 at 01:40:45 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2025.3640771

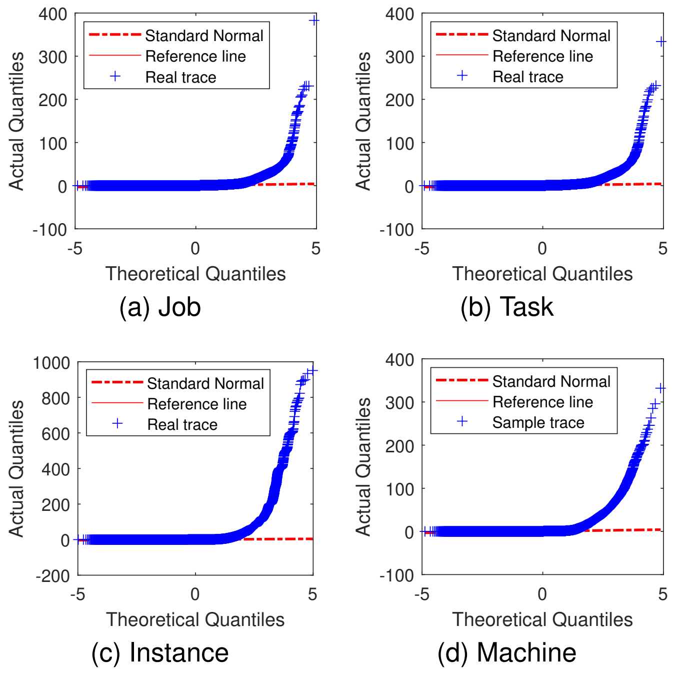


Fig. 3. Examine the Gaussianity of request activities in the PAI workloads at the job, task, instance, and machine levels through QQ plots of PAI trace data versus standard normal, respectively.

Finding (1): In the PAI workloads at each level, approximately more than half of the requests arrive no more than 1 second apart, and more than 30% of the request intervals are 0 second.
Finding (2): For PAI, the request arrival process at each level has a large variance.
Finding (3): For request arrival intervals in the PAI workloads at various levels, all CV values are significantly exceed 1.
Finding (4): The index of dispersion for request arrival intervals in the PAI workloads at the four levels are particularly large, especially for the instance and machine levels.
Implications: Request activities in the PAI workloads at all levels consistently exhibit burstiness, especially at the instance and machine levels. Scaling storage subsystems or allocating appropriate storage resources in response to increased computing demands caused by burstiness can enhance the performance of Alibaba's MLaaS platform.

B. Gaussianity Study

Previous research [5], [30] suggests that Gaussianity or non-Gaussianity in system workloads is a concern when building performance evaluation models. Conducting a Gaussianity test facilitates the accurate description of tail trends in the distribution of access characteristics and the construction of convincing models. Therefore, we deploy the Gaussianity test to study the request sequences in the PAI workloads.

A Gaussianity test can be performed through quantile-quantile (QQ) plots. For a random variable $X = \{X_t : t = 1, 2, \ldots\}$, the quantile refers to the real number x that satisfies the condition $P(X_t \le x) = c$, where c is a constant. The quantiles x, y for two random variables X and Y constitute a coordinate (x, y), and a series of coordinates form the trajectory a QQ plot. If the two data sets X and Y follow the same distribution, then the coordinates will fall on a straight line at a 45-degree angle, and vice versa.

For request activities in the PAI workloads at the job, task, instance, and machine levels, Figure 3 depicts the corresponding QQ plots of PAI trace data versus the corresponding standard normal distributions. As shown in Figures 3(a)-(d), the scatters corresponding to the PAI trace data at the aforementioned four levels clearly do not fall on a straight line. All the curves are concavely upward and display a heavy-tailed trend. Furthermore, we conducted the Kolmogorov-Smirnov test [31] on each of the four levels of the PAI workloads previously mentioned. The test results consistently yielded a statistic of 1, leading to the rejection of the null hypothesis of standard normal distribution at the 5% significance level. These observations indicate that the request behaviors appear to be non-Gaussian at the job, task, instance, and machine levels.

This finding is surprising because it diverges significantly from the Gaussian distribution previously used in analyzing Alibaba's production cluster data [5]. This phenomenon may be associated with the greater number of bursts present in the PAI workload when running state-of-the-art ML algorithms.

We summarize this key observation as

Finding (5): The scatter plots corresponding to the PAI workload at each level clearly do not fall on a straight line, and exhibit a heavy-tailed feature.
Implications: Request behaviors in the PAI workloads at the job, task, instance, and machine levels appear to consistently be non-Gaussian. During busy periods of heavy-tailed behaviors, Alibaba's MLaaS platform can allocate additional resources to handle increased computing demands, thereby reducing task latency.

C. Correlation Analysis

In the following, we employ a widely used statistical tool, the *auto-correlation function* (ACF) [13], [32], [33], to diagnose the correlation of arrival intervals of requests in the PAI workloads.

Let a time series be denoted as $Y = \{Y_t : t = 1, 2, \cdots, n\}$, with the expectation $\theta = E[Y_t]$; let k denote each time interval (or *lag*). Each lag corresponds to an auto-correlation coefficient independent of the time itself and is denoted as

Authorized licensed use limited to: Zhejiang Lab. Downloaded on January 04,2026 at 01:40:45 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2025.3640771

$R(k)$. Note $y_t = Y_t - \theta$, and we have $\{y_t : t = 1,\ 2, \cdots\}$. Then, for $k \geq 0$, the auto-correlation function with a lag of k is defined as:

$$R(k) = \frac{E[y_t \cdot y_{t+k}]}{E[y_t^2]} \tag{2}$$

where $R(k)$ denotes the correlation coefficient at the independent variable k. Equation (2) aims to measure the interrelationship between two adjacent elements in a time series. A comprehensive description of ACF can be found in [32].

The trend of the auto-correlation function curve is closely related to the patterns of request activities in the PAI workloads. If, as the lag increases, the correlation coefficient of the inter-arrival interval decreases rapidly and approaches zero, the correlation of request behaviors in PAI will become trivial. This suggests that the burstiness in the PAI workloads will be gradually smoothed out over time, making it reasonable to adopt an independent identically distributed method to characterize request activities. If the correlation coefficient does not rapidly approach zero, there exists a certain degree of correlation between the request arrivals in the PAI workloads. In this case, to accurately describe the request behaviors in PAI, it is necessary to consider other approaches, such as long-range dependence (also known as self-similarity).

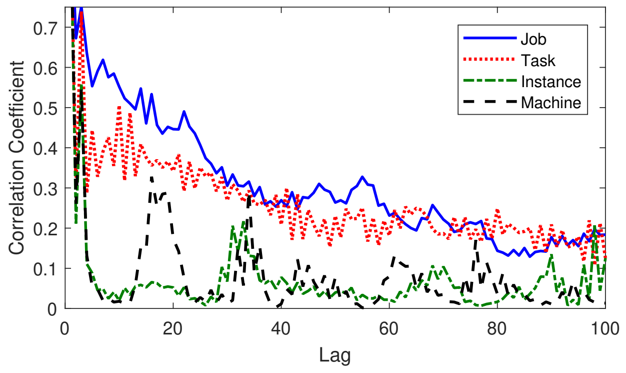


Fig. 4. Auto-correlation functions of request arrival intervals in the PAI workloads at the job, task, instance and machine levels.

Figure 4 illustrates the auto-correlation functions of arrival intervals for requests in the PAI workloads at the job, task, instance, and machine levels, respectively. As the lag increases from 0 to 500, the correlation coefficients of requests at the aforementioned four levels, do not sharply approach zero but show a gradual trend; instead, they exhibit a gradual decline. At certain lags, the coefficients increase, suggesting that the correlation persists over longer intervals. Interestingly, as shown in Figure 4, the trends of curves for the job-level and task-level workloads are relatively synchronized. This phenomenon may be related to the fact that most jobs in PAI have only one task [1].

These observations indicate a significant degree of correlation between request arrivals in the PAI workloads at the job, task, instance, and machine levels, rendering the use of an independent identically distributed method to describe request activities unsuitable. Consequently, request activities in PAI demonstrate not only bursts and heavy-tailed distributions but also correlations across extended time scales.

Autocorrelation causes job requests to exhibit periodic or trend-like characteristics, allowing the scheduling algorithm to predict future resource demand by analyzing historical data. For example, GPU resources are pre-allocated based on time series analysis to reduce delays caused by resource contention. The autocorrelation of task-level requests can lead to abrupt surges or drops in resource demands, exacerbating response delays during elastic scaling. In containerized environments, integrating predictive maintenance techniques, such as load prediction through real-time data analysis, is essential for optimizing elastic resource scheduling. For instance-level request streams with strong autocorrelation, sharding combined with algorithms like consistent hashing can direct correlated requests to the same instance. This approach reduces cross-instance communication overhead and prevents hotspot issues. Based on the autocorrelation of machine-level requests, elastic resource quotas can be reserved for auto-correlated tasks, and a preemption mechanism for interruptible low-priority tasks can be designed. When highly auto-correlated workloads are detected, reserved resources can be rapidly released to smooth load fluctuations.

The aforementioned observations demonstrate that when executing state-of-the-art ML algorithms on PAI, there are new and distinctive characteristics of request behaviors at the job, task, instance, and machine levels – not merely burstiness but also a discernible degree of correlation. The observed correlation of arrival intervals in the PAI workloads compels us to investigate the presence of self-similarity in the subsequent section.

We encapsulate this crucial insight as follows:

Finding (6): There exists a certain degree of correlation between requests in the PAI workload at each level.

Implications: Exploring self-similarity in the PAI workload at all levels becomes essential to accurately understand request behaviors and improve performance in Alibaba's MLaaS platform.

IV. Self-similarity Study

In this section, we explore the self-similarity in the PAI workloads at the job, task, instance and machine levels from the following three aspects: (1) showing the visualization, (2) providing theoretical evidence through the auto-correlation structures of the aggregation processes of request sequences, and (3) estimating the self-similarity parameter through classical tools and making the corresponding judgments.

A. Self-similar Process

First, let us outline the basic knowledge of the self-similar process involved in this work. A detailed introduction can be found in the literature [34].

Authorized licensed use limited to: Zhejiang Lab. Downloaded on January 04,2026 at 01:40:45 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2025.3640771

Consider a stochastic process $X = \{X_t : t = 1,\ 2,\ 3, \cdots\}$, then $X^{(m)} = \{X_t^{(m)} : t = 1,\ 2,\ 3, \cdots\}$ is referred to as the m-order aggregated process corresponding to X, if

$$X_t^{(m)} = \frac{1}{m} \sum\nolimits_{i=0}^{m-1} X_{tm-i}. \tag{3}$$

Hence, we have the ACF corresponding to $X^{(m)}$ as $R^{(m)}(k)$. For the traditional stochastic process such as Poisson, the ACF for $X^{(m)}$ *degenerates* with an increasing m, and converges to 0, i.e.,

$$R^{(m)}(k) \to 0,\ as\ m \to \infty. \tag{4}$$

If the structure of $R^{(m)}(k)$ does not *degenerate* with the increase of m and tends to be the same function structure (i.e., as $m \to \infty$), we have

$$R^{(m)}(k) \to \frac{1}{2}\left[(k+1)^{2-\lambda} - 2k^{2-\lambda} + (k-1)^{2-\lambda}\right], \tag{5}$$

and the process X is said to be self-similar with the *Hurst parameter* H ($H = 1 - \frac{\lambda}{2},\ 0 < \lambda < 1$). The Hurst parameter is the sole parameter describing the degree of self-similarity, and a value in the range of $(0.5, 1)$ indicates the presence of self-similarity, which is also called *long-range dependence* (LRD).

Lemma 1 [35]. For a stochastic process X with covariance stationarity, X is self-similar and the two propositions below are equivalent if the process X matches any of the following two conditions:

(1) X has an auto-correlation function in the form

$$R(k) = \frac{1}{2}\left[(k+1)^{2-\lambda} - 2k^{2-\lambda} + (k-1)^{2-\lambda}\right]. \tag{6}$$

(2) the m-order aggregated process for X matches

$$Var(X^{(m)}) = \sigma^2 m^{-\lambda},\ for\ 0 < \lambda < 1. \tag{7}$$

B. Visualization

A self-similar workload is predominantly characterized by the persistence of bursts and burst aggregations at diverse timescales, with the persistent pattern itself exhibiting notably similar traits.

Therefore, at different timescales, we examine request behaviors in the PAI workload at the levels of job, task, instance, and machine, observing analogous patterns of activity. To intuitively demonstrate the findings, we depict the task-level request sequences at three disparate timescales in the left column of Figure 5, and in the right column, we draw the machine-level request sequences on the same time scale as in the left column, respectively.

In Figure 5, the left column includes subplots (a)-(c), and the right column encompasses subplots (a')-(c'), with each subsequent timescale being an order of magnitude larger than the one preceding it. On the horizontal axis, we plot the timescale, and the vertical axis indicates the number of requests per time unit.

In Figure 5, for three subplots within each column, each subplot originates from a subinterval randomly selected from the time range represented in the succeeding subplot and amplifies the temporal resolution by a factor of 10. For instance, subplot (a) in the left column delineates a brief span (100,000 seconds) sampled randomly from subplot (b), which in turn portrays a concise interval (1,000,000 seconds) randomly chosen from subplot (c), and similarly for the subplots (a')-(c') in the right column.

Evident in each column of Figure 5, each subplot exhibits numerous "spikes" (i.e., bursts), which are sequences of intervals marked by pronounced fluctuations – larger bursts intermingling with smaller ones. Across the three subplots in both the left and right columns, these "spikes" within the PAI workload at the task (or machine) level, manifest a consistent pattern, unvarying across different timescales.

The aforementioned patterns indicate that the temporal span of bursty request activities in PAI is composed of nested subintervals, each characterized by bursty behavior. In turn, these subintervals are comprised of further subintervals reflecting similar burst-like dynamics. This implies that sequences of requests in PAI exhibit the characteristics of a self-similar process over extended timescales.

Comparing the left and right columns in Figure 5, it is readily apparent that the requests in the machine-level PAI workload are more intensive than those at the task level – a trend that aligns fully with the test results given in Section III-A and supports the burstiness diagnosis results.

The aforementioned critical observations are encapsulated as

Finding (7): The time range in which requests are bursty consists of nested subintervals that are made up of even smaller subintervals with similar burst behaviors.

Finding (8): Requests in the instance and machine-level PAI workloads are more intensive than those at the job and task levels.

Implications: Request activities in the PAI workload at each level exhibit characteristics of self-similarity. This offers an opportunity to enhance load balancing by incorporating the factor of self-similarity when dynamically and reasonably distributing computing tasks, thereby avoiding bottlenecks during bursty periods.

C. Theoretical Evidence

The discussion about the structure of $R^{(m)}(k)$ in Section IV-A provides a theoretical basis for detecting the existence of self-similarity. Specifically, self-similar workloads exhibit burstiness at different time scales due to similar scale-invariant properties.

For requests in the PAI workloads at the levels of job, task, instance, and machine, we examine the auto-correlation functions of the aggregated time series of request sequences at multiple aggregation levels. Firstly, taking the request series at the machine level as an example, we draw the auto-correlation functions of the time series at three aggregation levels in

Authorized licensed use limited to: Zhejiang Lab. Downloaded on January 04,2026 at 01:40:45 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2025.3640771

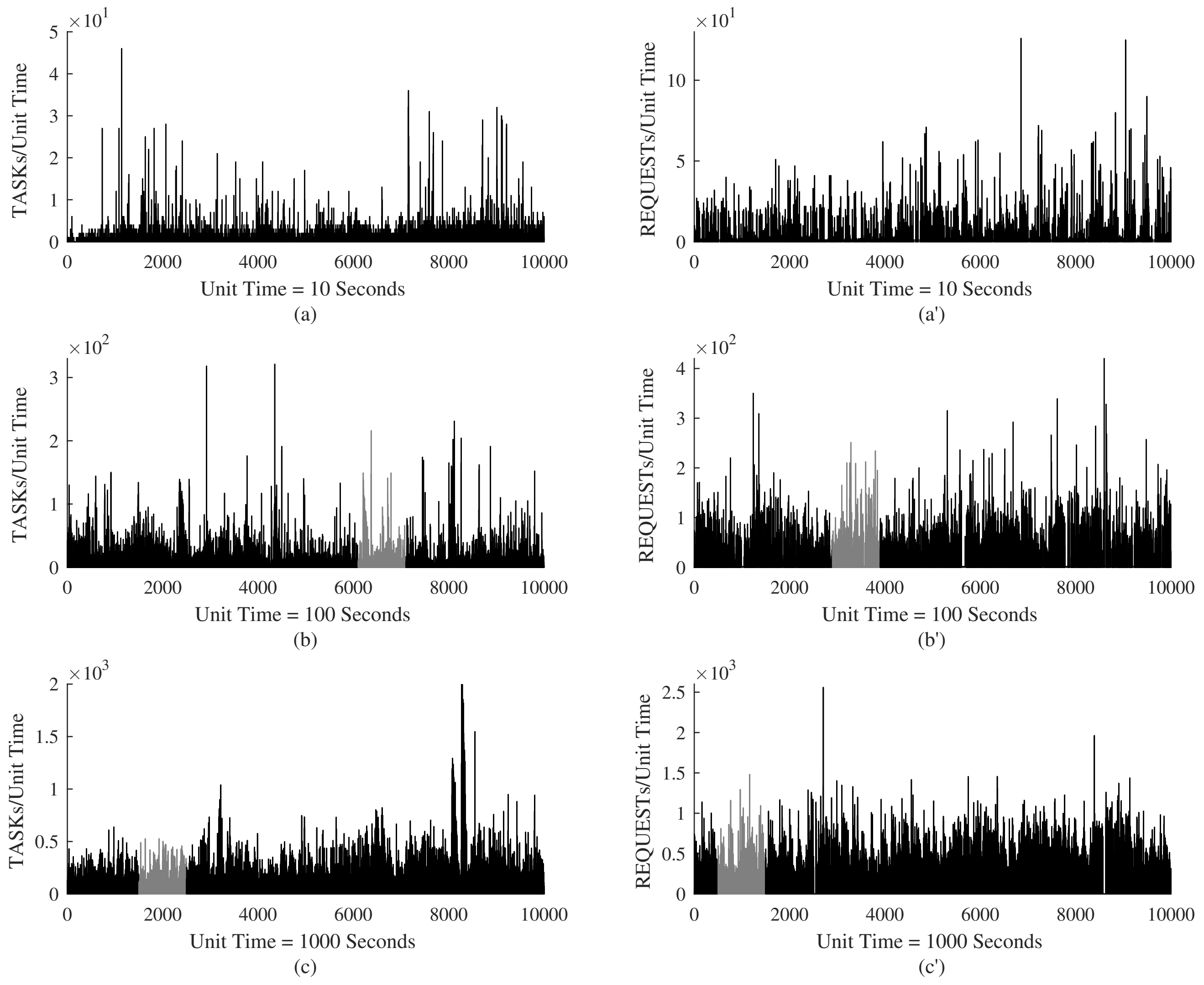


Fig. 5. Visualization of request sequences in the PAI workloads at the levels of task and machine: Left plots depict the number of requests at the task level (TASKs) per unit time for three different time scales (a)-(c), and right plots illustrate the number of requests at the machine level (REQUESTs) per unit time for the same time scales (a′)-(c′). Each plot has ten thousand buckets.

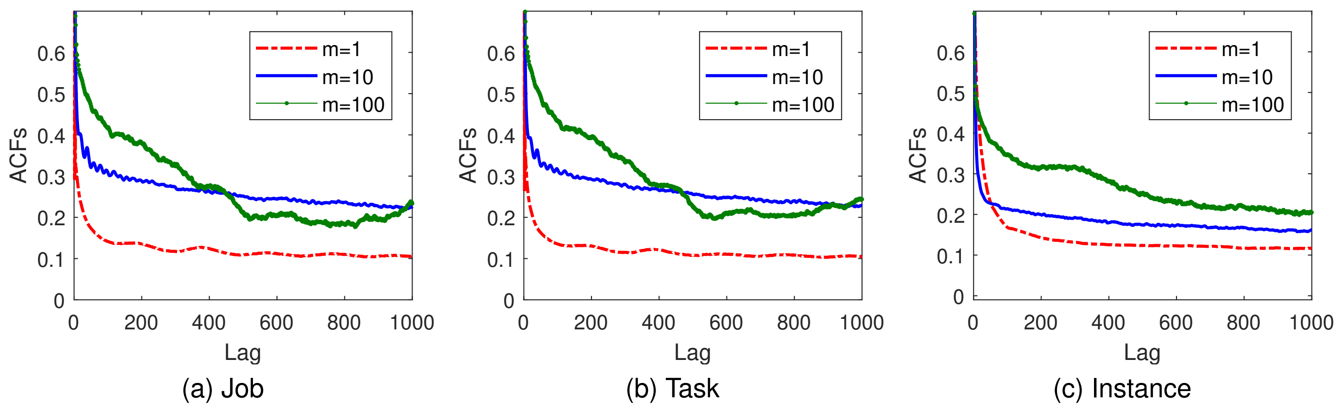


Fig. 6. Auto-correlation functions of the aggregated time series for request sequences in the PAI workloads at the levels of: (a) job, (b) task, and (c) instance, respectively.

Figure 7(a) and plot the auto-correlation functions of the corresponding artificial Poisson workloads in Figure 7(b). The aggregation levels (m) are 1, 10, and 100, respectively.

As observed in Figure 7(a), with the increase of lag from 1 to 1000, the correlation coefficients of the machine-level request sequence fluctuate and do not approach zero. This

Authorized licensed use limited to: Zhejiang Lab. Downloaded on January 04,2026 at 01:40:45 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2025.3640771

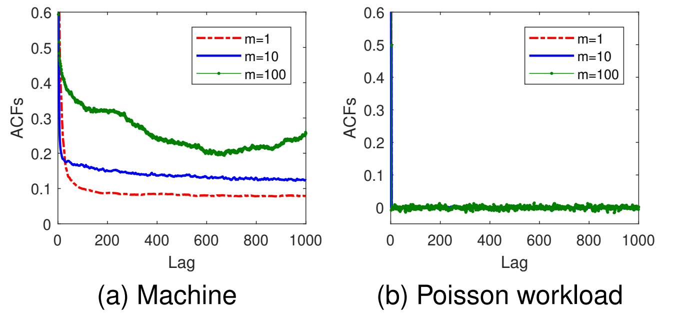


Fig. 7. Auto-correlation functions of the aggregated time series for (a) the request sequence in the machine-level PAI workload, and (b) an artificial Poisson workload.

suggests that the corresponding request behaviors exhibit long-range dependence. Additionally, the auto-correlation curves at various aggregation levels appear to converge to a similar function structure. In contrast, as seen in Figure 7(b), the auto-correlation coefficients of artificial Poisson workloads at each aggregation level are generally very small and almost equal to zero. These observations indicate that the auto-correlation structure of the request sequence in the machine-level PAI workload behaves in a manner akin to the self-similar process, distinctly different from the Poisson series.

Similarly, we examine the request sequences in PAI workloads at the job, task, and instance levels. Figures 6(a)-(c) depict the auto-correlation curves at three aggregation levels, respectively. As seen in Figures 6(a)-(c), as the *lag* increases from 1 to 1000, the auto-correlation coefficients of the aggregated time series for request sequences at the job, task, and instance levels do not converge to zero either, indicating long-range dependence. Furthermore, the trends of the auto-correlation curves at various aggregation levels in Figure 6 also seem to approach a similar function structure. This feature is similar to a self-similar process.

In summary, all request sequences in the PAI workloads at the job, task, instance, and machine levels, appear to be self-similar. We summarize this key observation as

Finding (9): For the PAI workload at each level, the auto-correlation curves of the request sequence at different aggregation levels appear to converge to a similar function structure.

Implications: Request activities in the PAI workload at each level exhibit characteristics of a self-similar process. Incorporating the knowledge of self-similarity into workload modeling can better forecast resource requirements for ML tasks.

D. Estimating the Hurst Parameter

In the open interval (0.5, 1), the greater the Hurst parameter, the higher the degree of self-similarity. In this study, for the request sequences in the PAI workloads at the levels of job, task, instance, and machine, we employ two well-known analytical tools – the variance-time plot [12] and R/S analysis (also called Pox plot) [36] – to estimate the Hurst parameters, as shown in Figure 8. Both the variance-time plots and the Pox plots are widely used to estimate the Hurst parameter of data samples and provide faithful test results.

Using the variance-time plot as an example, we can calculate the variance of the corresponding m-order aggregated process $X^{(m)}$ for a given set of sample trace data X using Equation (7). By taking the logarithm of both sides of Equation (7), we get:

$$log_{10}(Var(X^{(m)})) = log_{10}(\alpha^2) - \lambda log_{10}(m), \quad (8)$$

where the constant $log_{10}(\alpha^2)$ is independent of m. If $log_{10}(Var(X^{(m)}))$ is considered as a function of $log_{10}(m)$, we draw the curve of $log_{10}(Var(X^{(m)}))$ versus $log_{10}(m)$, and obtain a series of scattered points which can usually fit a linear regression straight line with a slope of $-\lambda = 2(H-1)$. Then we can calculate the Hurst parameter H.

Figure 8(a) illustrates the variance-time plots for request activities in the PAI workloads at the levels of job, task, instance, and machine. Figure 8(a) where $log_{10}(Var(X^{(m)}))$ is abbreviated as $log_{10}(variances)$, yields a line of slope $-\lambda = 2(H-1)$ by depicting the scatter plots of $log_{10}(variances)$ versus $log_{10}(m)$. The corresponding Hurst parameter is estimated to be 0.636, 0.629, 0.548, and 0.563, respectively. Furthermore, the Pox plots generated from the R/S analysis of the PAI traces are depicted in Figure 8(b). As shown in Figure 8(b), through a least squares linear fitting, the Hurst parameter is estimated to be 0.529, 0.558, 0.567, and 0.504, respectively.

According to the above observations, for requests in the PAI workloads at the levels of job, task, instance, and machine, the Hurst parameter estimates are all greater than 0.5, which further proves the existence of self-similarity quantitatively. We summarize this key observation as

Finding (10): For the PAI workloads at various levels, all Hurst parameter estimates are greater than 0.5.

Implications: These Hurst parameter estimates for PAI quantitatively confirm the existence of self-similarity. This makes it possible to take into account and apply self-similarity as evaluating and optimizing the performance of Alibaba's MLaaS platform.

V. PAI Workload Synthesis

Generally, in the research community, synthetic request workloads can be applied to actual systems to simulate request sequences and evaluate system performance. By performing burstiness diagnosis, correlation study, Gaussianity test, and self-similarity analysis for the PAI workloads at the job, task, instance, and machine levels, we have obtained the following

Authorized licensed use limited to: Zhejiang Lab. Downloaded on January 04,2026 at 01:40:45 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2025.3640771

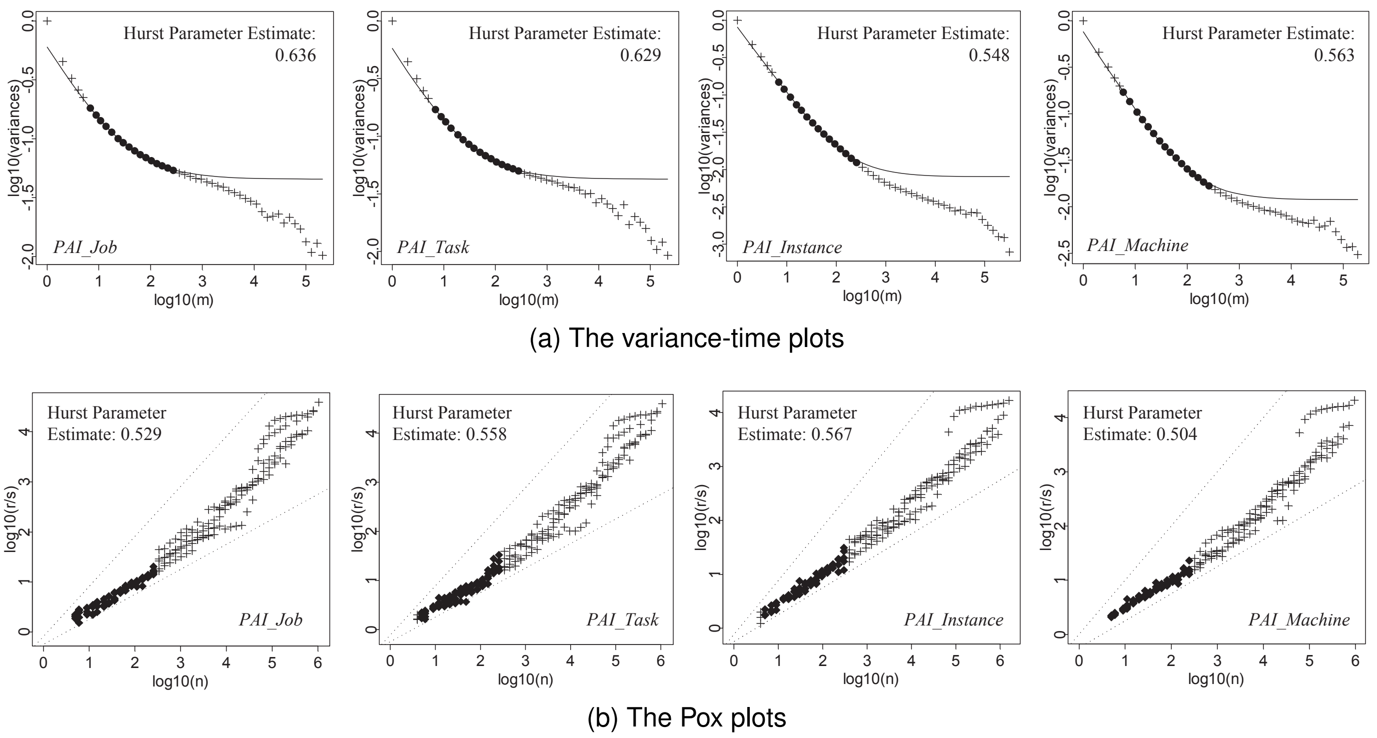


Fig. 8. Estimating Hurst parameters for request activities in the PAI workloads at the job, task, instance, and machine levels using (a) variance-time plots, and (b) Pox plots, respectively.

observations: (1) All the request arrival processes in the four-level PAI workloads are significantly bursty. (2) Traditional methods may be inadequate to accurately describe the PAI workloads in which request arrivals show a certain degree of correlation. (3) Self-similarity seems to exist in the PAI workloads at various levels. (4) Request activities in the PAI workload at each level appear to be non-Gaussian.

Inspired by these findings, this section proposes a workload model to faithfully synthesize request series in the self-similar PAI workloads at the aforementioned four levels.

A. Workload Modeling

To faithfully synthesize request sequences in PAI, the generator not only needs to efficiently describe self-similarity, but also accurately capture the bursts and heavy-tailed properties under non-Gaussian conditions. Since the alpha-stable process perfectly matches the generator's requirements, we extend the alpha-stable model proposed in the referenced literature [8] by redefining model parameters to generate request sequences at the job, task, instance, and machine levels, respectively.

A random variable X that follows an alpha-stable distribution is denoted by $X \sim S^{\alpha}_{\sigma,\beta,\mu}$ [8]. For PAI, we can formalize the alpha-stable workload model as:

$$REQUEST(i) = v \cdot N_{\alpha,\beta,H}(i) + \delta \tag{9}$$

where $REQUEST(i)$ denotes the number of requests in the i^{th} unit time; α is the degree of heavy tail ($0 < \alpha \leq 2$); β denotes the skewness parameter ($-1 \leq \beta \leq 1$); viariables σ (> 0) and μ ($\in R$) measure the scale and location parameters, respectively.

The distribution morphology and tail behavior in synthetic workloads are driven by parameters α and β. As α approaches 2, the distribution gradually converges to a Gaussian form with thinner tails. This underestimates the probability of extreme events, resulting in reduced accuracy in reproducing spike states (such as request bursts) within synthetic workloads. When α decreases (e.g., $\alpha < 1.5$), the tails thicken significantly. This enables more accurate simulation of heavy-tailed characteristics, thereby enhancing fidelity in peak generation. Meanwhile, parameter β modulates the asymmetry of the distribution. When β approaches 1, the distribution becomes right-skewed, amplifying the weight of the right tail – a configuration particularly useful for synthesizing positively shifted workloads (e.g., sudden surges in service requests). When β approaches -1, the distribution exhibits left-skewed characteristics, accentuating the left tail weight to simulate anomalies with low values in workload. The deviation of β distorts the temporal distribution of event occurrence probabilities, reducing temporal alignment accuracy between synthetic sequences and actual workloads.

In addition, variable v is referred to as the mean number of requests per unit time in a request series, H denotes the degree of self-similarity, δ measures the standard deviation of the number of requests per unit time relative to the mean, and detailed description of $N_{\alpha,\beta,H}(i)$ can be found in the literature [8]. All the above parameters can be measured and obtained by the maximum likelihood estimation from real PAI trace datasets at the job, task, instance, and machine levels, respectively. More specifically, for a sample series $X = \{x_i\}$ of the PAI trace at each level, we simplify the alpha-stable parameters to be represented by a vector $\boldsymbol{\theta} = (\alpha, \beta, \sigma, \mu)$, with

Authorized licensed use limited to: Zhejiang Lab. Downloaded on January 04,2026 at 01:40:45 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2025.3640771

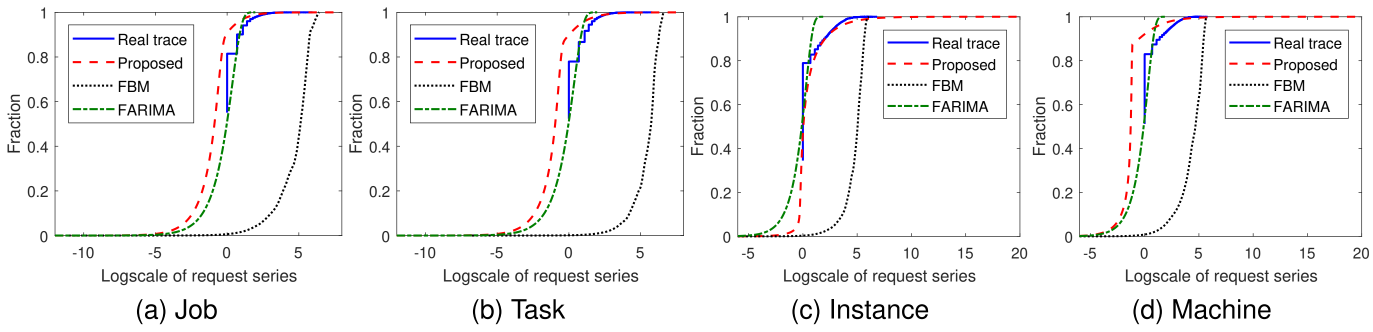


Fig. 9. Comparison of CDFs between synthetic sequence and real one for the PAI workloads at the job, task, instance and machine levels, respectively.

the corresponding density denoted as $f(x_i|\boldsymbol{\theta})$. The parameter space is $\boldsymbol{\Theta} = (0, 2] \times [-1, 1] \times (0, \infty) \times (-\infty, +\infty)$. The log-likelihood function for X can be expressed as $L(\boldsymbol{\Theta}) = \sum_{i=1}^{n} log(f(x_i|\boldsymbol{\theta}))$. Then, using the pre-computed spline approximation method [37], the maximum likelihood estimation of stable parameters is computed, and the stability is verified. Table III summarizes the estimated results for each parameter.

TABLE III
ESTIMATES OF THE PARAMETER OF ALPHA-STABLE MODEL BASED ON MAXIMUM-LIKELIHOOD ESTIMATE.

Trace level	**Alpha-stable parameter**				v	δ
	α	β	σ	μ		
Job	1.17084	1.00000	0.24957	–0.36245	1.01	2.73
Task	0.99094	1.00000	0.18884	–0.35424	1.16	2.91
Instance	0.60134	0.64731	0.23627	0.89549	3.63	14.6
Machine	0.61770	1.00000	0.03200	–0.28233	2.12	47.1

Each row of Table III includes the trace level, the estimates of v, δ, and four alpha-stable parameters. As shown in Table III, the estimates of the parameter α are all less than 2, indicating that the request behaviors in the PAI workloads at the job, task, instance, and machine levels are not Gaussian. This observation aligns completely with the test results in Section III-B, effectively verifying the accuracy of the test results in Section III-B.

Additionally, for the aforementioned four alpha-stable parameters, i.e., α, β, σ and μ, the sensitivity levels are high, medium, medium, and low, respectively. Among them, parameter α has a significant impact on the model's robustness – when the value of α approaches 1, the model exhibits higher tolerance to noise and outliers.

B. Trimmed Mean of Errors

For the PAI workloads, based on the parameters estimated in Table III, we employ the alpha-stable workload model to synthesize request sequences at the job, task, instance, and machine levels, respectively. To evaluate the error between the synthetic request sequence and the actual one, we leverage the trimmed mean of errors [8].

The trimmed mean is the arithmetic average calculated after truncating a smaller proportion of data at both ends of a sample. It is more robust to outliers than the usual sample mean, such as the arithmetic average. The smaller the trimmed mean of the errors, the higher the matching degree between the synthetic sequence and the real one.

Given the existence of self-similarity in PAI, we also employ two typical self-similar workload models – fractional Brownian motion (FBM) [38] and fractional auto-regressive integrated moving average (FARIMA) [39] – to synthesize the request sequences at the job, task, instance, and machine levels. Furthermore, we use FBM and FARIMA to compare the fit of alpha-stable request sequences. The FARIMA model is capable of describing both long-range and short-range dependencies, while the FBM model is suitable for characterizing self-similarity under Gaussian conditions. A detailed comparison of the advantages and disadvantages of the proposed model versus representative self-similar models, including FBM and FARIMA, is summarized in Table IV.

Table V summarizes the trimmed mean of errors for the three workload models, namely, FBM, FARIMA, and our proposed model. According to Table V, for the PAI workloads at the job, task, instance, and machine levels, the trimmed means of errors between the actual request sequences and the corresponding alpha-stable request sequences are minimal.

Moreover, our alpha-stable request sequences improve the matching degrees of FBM by 99%. In Table V, our scheme also shows improvements in the matching degree of FARIMA by varying degrees. However, the matching degrees of our scheme (i.e., 0.84, 0.88, 1.28, and 0.78) are very close to those of FARIMA (i.e., 1.28, 1.34, 1.39, and 1.26), respectively. This indicates that our proposed approach is at least one of the best scenarios.

C. An Empirical Study

For the four levels of PAI workloads – namely, job, task, instance, and machine – to intuitively compare the matching degree of the various synthetic request sequences with the actual ones, we depict the cumulative distribution functions (CDFs) of both the actual and the synthetic sequences in Figure 9.

In Figure 9, the horizontal axis denotes the log scale of the number of requests per unit time, while the vertical axis represents the proportion of the corresponding values in the whole request sequence. Each coordinate (x, y) indicates that the proportion of logarithms less than or equal to x in the corresponding time series is y.

Authorized licensed use limited to: Zhejiang Lab. Downloaded on January 04,2026 at 01:40:45 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2025.3640771

TABLE IV
COMPARISON OF ADVANTAGES AND DISADVANTAGES BETWEEN THE PROPOSED MODEL AND THE REPRESENTATIVE SELF-SIMILAR MODELS, INCLUDING FBM AND FARIMA.

Model	Advantage	Disadvantage	Applicable scenario
FBM	Simple parameters make the model tractable, and it can describe self-similarity under Gaussian conditions.	Cannot simultaneously describe both long-range and short-range dependence.	Suitable for real-time system modeling, simulation, and performance analysis.
FARIMA	Highly flexible, capable of characterizing both long-range and short-range dependence simultaneously.	Overly complex, computationally intensive, and does not account for tail characteristics in workload.	Applicable to complex system modeling and non-real-time performance analysis.
Proposed	Parsimonious and capable of characterizing burstiness and tail characteristics in workloads under both Gaussian and non-Gaussian conditions, capturing both long-range and short-range dependence simultaneously.	Lacks a closed-form expression for the probability density function, hindering further performance analysis.	Investigating self-similarity, burstiness, and variations of tail characteristics in workloads under Gaussian and non-Gaussian conditions.

TABLE V
THE TRIMMED MEANS OF ERRORS FOR SYNTHESIZING REQUEST SEQUENCES IN THE PAI WORKLOADS AT THE JOB, TASK, INSTANCE AND MACHINE LEVELS.

Trace level	FBM ①	FARIMA ②	Proposed	Improvement	
				vs ①	vs ②
Job	166.72	1.28	0.84	99.5%	34.4%
Task	293.79	1.34	0.88	99.7%	34.3%
Instance	156.13	1.39	1.28	99.2%	7.9%
Machine	109.46	1.26	0.78	99.3%	38.1%

As shown in Figure 9, for each level of PAI workload, the alpha-stable synthetic sequence matches well with the actual sequence, particularly at the instance level, echoing the corresponding trimmed mean of errors, 0.84, 0.88, 1.28 and 0.78, as shown in Table V.

Furthermore, Figure 9 demonstrates that the FARIMA synthetic sequences also seem to match well with the actual sequences, even being non-inferior to the alpha-stable synthetic sequences at the job, task, and machine levels. This observation is consistent with the findings presented in Table V.

However, the alpha-stable synthetic sequences have one advantage over the FARIMA synthetic ones, as shown in Figures 9(a)-(d), the alpha-stable synthetic sequences better characterize the heavy-tailed features of the actual sequences, especially for the instance-level request series. This phenomenon may be associated with the fact that, while FARIMA is capable of describing both long-range and short-range dependence, it struggles to accurately represent the tail characteristics in self-similar workloads, as illustrated in Table IV.

Furthermore, Figure 9 reveals that the matching degree of the alpha-stable synthetic sequences is much better than that of the FBM ones. The aforementioned observation may be caused by the following reasons: although FBM can characterize self-similarity under Gaussian conditions, it fails to faithfully describe self-similarity under non-Gaussian conditions, as shown in Table IV. This suggests that the request behaviors in the PAI workloads at the job, task, instance, and machine levels, do not tend to be Gaussian. This observation is fully consistent with the test results presented in Section III-B and supports the Gaussianity test results.

To summarize, for the PAI workloads at the job, task, instance, and machine levels, the alpha-stable synthetic sequences yield convincing matching degrees.

VI. DISCUSSION

To facilitate the design of next-generation hardware architectures, it is crucial to gain insights into ML application workloads and assess access patterns in the execution stack [40]. In this section, we discuss the significance of characterizing ML application workloads for building an MLaaS platform.

A. Performance Evaluation

Characterizing the access patterns of ML workloads is an effective means of evaluating the performance of ML clusters. For example, Paul *et al.* [6] analyzed I/O behaviors of ML I/O workloads based on a Darshan log of 23,000 HPC ML I/O jobs. Paul's analysis provided insights into potential performance optimization efforts and offered support for assessing I/O trends across the entire HPC cluster. Li *et al.* [4] studied system operations, job characteristics, and user behaviors of contemporary GPUs in HPC systems. Li observed that when HPC systems encounter GPU-accelerated AI and ML workloads, a significant number of users perform low-utilization development and idle work. Based on an operation-level empirical analysis of various CNNs, Hafeez *et al.* [41] proposed a model-driven method called *Ceer* to determine the optimal GPU instance(s) for any given CNN.

Moreover, Wang *et al.* [42] studied DL training workloads from Alibaba's AI platform and evaluated the achievable performance of the workload on various potential software/hardware mappings. They revealed that weight/gradient communication during training accounts for almost 62% of the total execution time.

B. Resource Management

Resource management in data centers often relies on studying job features and user behaviors. Yeung *et al.* [43] proposed a predictive resource manager based on the GPU utilization of heterogeneous DL jobs inferred from the computational graph features of DL models. Weng *et al.* [1] characterized a two-month-long workload trace collected in an Alibaba production model cluster containing more than 6,000 GPUs, revealing challenges of heterogeneous GPU clusters, such as low GPU utilization, long queuing delay, and load imbalance across

Authorized licensed use limited to: Zhejiang Lab. Downloaded on January 04,2026 at 01:40:45 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2025.3640771

heterogeneous machines. Hu *et al.* [2] analyzed a real job trace from SenseTime and designed a quasi-shortest service priority scheduling service that minimized the average job completion time of clusters.

Additionally, Jeon *et al.* [7] analyzed the workload characteristics of a two-month-long trace from Microsoft's multi-tenant GPU clusters to study the factors influencing cluster utilization of DNN-trained workload on multi-tenant clusters and provided a design guide for next-generation cluster scheduler. Jiang *et al.* [5] investigated a cluster-trace-v2018 trace of an Alibaba production cluster. Jiang's study observed daily periodic fluctuations of workload characteristics in the production clusters and suggested that performance bottlenecks in collocated clusters are caused by the memory system, which data centers can use to establish efficient scheduling strategies.

C. Self-similarity

The utilization of self-similarity in request behavior can significantly enhance system optimization across multiple dimensions [22]. For machine learning cloud platforms, this manifests in the following specific aspects:

First, accelerated model training. Understanding and modeling the self-similar characteristics in training tasks enables the implementation of dynamic caching strategies. For example, prioritizing the caching of frequently accessed training dataset segments can improve iteration speeds for typical tasks.

Second, elastic resource scheduling. Constructed prediction models based on the self-similarity of workload, help cloud platforms anticipate computational resource demands in advance, automatically pre-warm GPU instances, and pre-allocate storage bandwidth.

Finally, enhanced simulation testing. A virtual workload generator designed with self-similar properties can simulate real-world stress scenarios, ranging from small-scale image classification tasks to training trillion-parameter large-scale models.

D. Synthetic Model

Trace-driven simulators, often used to evaluate the GPU cluster performance in deep learning systems [43], benefit from synthetic models in several ways. Synthetic models offer three main advantages over trace-driven simulators. First, synthetic models provide a deeper understanding of offline workloads, such as bursty activities and heavy-tailed properties. Second, they contribute to evaluating specific subsystems rather than the entire system by providing workloads for various isolated subsystems. Third, synthetic models enable hypothetical performance analysis by modifying input parameters, helping to identify key factors influencing outcomes.

The modeling method directly affects performance evaluation and system design verification. In this study, the proposed synthetic models, based on inputs measured from the PAI traces, can faithfully capture the characteristics of the PAI workloads at the job, task, instance, and machine levels. This is vital for system design and performance optimization in MLaaS.

Due to the fact that PAI, a typical cloud-based ML platform for Artificial Intelligence, adheres to generic cloud construction principles similar to other ML platforms such as Amazon SageMaker, Google Vertex AI, and Microsoft Azure Machine Learning, these cloud ML platforms share identical characteristics. For instance, they all provide hybrid CPU/GPU computing resources, support on-demand auto-scaling, and integrate batch processing with real-time inference capabilities. Consequently, our proposed method is applicable for studying the burstiness of request activities in other MLaaS platforms.

VII. Conclusion

With the continued advances and widespread use of machine learning technologies, major tech companies are deploying MLaaS clouds equipped with GPU clusters to run various types of ML application workloads. To effectively schedule and manage GPU clusters, it is imperative to deeply understand request behaviors in MLaaS workloads.

For requests in the PAI workloads – a representative and real-word MLaaS workload – at the job, task, instance, and machine levels, we gauged the burstiness and unveiled that the request arrival process is highly bursty. We then performed the Gaussianity tests and observed that bursty request activities in the PAI workload at each level appear to be non-Gaussian.

We also investigated the correlation of request arrivals and discovered that correlations exist among the requests in the PAI workloads at various levels. This finding led us to reveal the self-similar nature of the PAI workloads through visual evidence, the auto-correlation structure of an aggregated process of request sequences, and Hurst parameter estimates.

Furthermore, based on the inputs measured from the PAI traces, we have implemented an alpha-stable workload model to generate synthetic request sequences for the PAI workloads at all levels. Experimental results demonstrate that our model outperforms typical self-similar workload models in faithfully simulating burstiness and heavy-tailed characteristics. Moreover, we are designing a burst-aware scheduler embedded with this proposed model, which enables predicting when bursts are likely to occur in advance. This allows finding an appropriate time to proactively activate backup servers and smoothly handle burst requests. This approach both reduces system latency and saves energy consumption.

When the research community studies the characteristics of computer system workloads, the specific application scenarios may lead to diverse timeframes for datasets, including months [22], days [27], hours or even minutes [44]. We utilize the PAI trace data primarily for the following reasons: (1) The trace data we use was collected by Wengâs group [1], with a maximum length of two months, and this two-month timeframe has already captured sufficient characteristics. (2) The job types in the PAI workloads are representative and have been recognized by the community [1]. Furthermore, should longer traces of machine learning cloud platform covering more job types become available in the future, we are willing to conduct further analysis over longer time periods to investigate whether any previously unseen characteristics emerge.

Authorized licensed use limited to: Zhejiang Lab. Downloaded on January 04,2026 at 01:40:45 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2025.3640771

In the MLaaS architecture, the burst buffer mechanism, a core component for handling high-concurrency I/O requests, has made intelligent optimization of resource allocation essential for building high-performance distributed training systems. Therefore, we plan to dynamically adjust read/write cache allocation in the burst buffer based on quantitative analysis of ML I/O burst intensity in the near future. This approach aims to enhance the storage subsystemâs throughput efficiency for intermittent data surges through elastic resource scheduling.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their helpful comments in reviewing this paper. This research was supported by Guangxi Natural Science Foundation under Grant No. 2025GXNSFAA069426. This work was supported in part by the National Key Research and Development Program of China under Grant 2021ZD0110700, in part by the National Natural Science Foundation of China under Grants 62072214 and 62172361, in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2021B1515120048, in part by the specific research fund of The Innovation Platform for Academicians of Hainan Province under Grant YSPTZX202410, and in part by the Program of Zhejiang Province Science and Technology under Grant 2022C01044.

REFERENCES

[1] Q. Weng, W. Xiao, Y. Yu, and et al, "MLaaS in the Wild: Workload Analysis and Scheduling in Large-Scale Heterogeneous GPU Clusters," in *Proceedings of the 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI'22)*, Renton, WA, April 2022.

[2] Q. Hu, P. Sun, S. Yan and et al, "Characterization and prediction of deep learning workloads in large-scale GPU datacenters," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC'21)*, St. Louis, MO, USA, November 2021.

[3] M. Wajahat, A. Yele, T. Estro and et al, "Distribution fitting and performance modeling for storage traces," in *Proceedings of the 27th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, October 2019, pp. 138–151.

[4] B. Li, R. Arora, S. Samsi, and et al, "AI-Enabling Workloads on Large-Scale GPU-Accelerated System: Characterization, Opportunities, and Implications," in *Proceedings of the 28th Annual IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, April 2022, pp. 1224–1237.

[5] C. Jiang, Y. Qiu, W. Shi, Z. Ge, J. Wang, S. Chen, C. Cerin, Z. Ren, G. Xu and J. Lin, "Characterizing Co-located Workloads in Alibaba Cloud Datacenters," *IEEE Transactions on Cloud Computing*, October 2020, DOI: 10.1109/TCC.2020.3034500

[6] A. K. Paul, A. M. Karimi, and F. Wang, "Characterizing Machine Learning I/O Workloads on Leadership Scale HPC Systems," in *Proceedings of the 29th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Houston, TX, USA, November 2021.

[7] M. Jeon, S. Venkataraman, A. Phanishayee, and et al, "Analysis of Large-Scale Multi-Tenant GPU Clusters for DNN Training Workloads," in *Proceedings of the 2019 USENIX Annual Technical Conference (ATC)*, Renton, WA, USA, July 2019, pp. 947–960.

[8] D. Feng, Q. Zou, H. Jiang, and Y. Zhu, "A novel model for synthesizing parallel I/O workloads in scientific applications," in *Proceedings of the 2008 IEEE International Conference on Cluster Computing (CLUSTER)*, 2008.

[9] M. Abadi, P. Barham, J. Chen, and et al, "TensorFlow: A System for Large-Scale Machine Learning," in *Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2016, pp. 265–283.

[10] T. Chen, M. Li, Y. Li, and et al, "MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems," 2015, arXiv preprint arXiv:1512.01274.

[11] R. Zhu, K. Zhao, H. Yang, and et al, "AliGraph: a comprehensive graph neural network platform," in *Proceedings of the VLDB Endowment*, 2019.

[12] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, "On the self-similar nature of ethernet traffic (extended version)," IEEE/ACM Transactions on Networking, vol. 2, pp. 1–15, February 1994.

[13] S. Gupta, and A. D. Dileep, "Long range dependence in cloud servers: a statistical analysis based on google workload trace," *Computing*, vol. 102, pp. 1031–1049, 2020.

[14] Q. Li, S. Wang, Y. Liu, and et al, "Traffic self-similarity analysis and application of industrial internet," *Wireless Networks*, 2020, DOI: 10.1007/s11276â?20â?2420â?.

[15] Z. Li, L. Zheng, Y. Zhong, and et al, "AlpaServe: Statistical Multiplexing with Model Parallelism for Deep Learning Serving," in *Proceedings of the 17th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, Boston, MA, USA, July 2023: 663â?79.

[16] B. Zhang, S. Li, and Z. Li, "MIGER: Integrating Multi-Instance GPU and Multi-Process Service for Deep Learning Clusters," in *Proceedings of the 53rd International Conference on Parallel Processing (ICPP)*, Gotland, Sweden, August 2024: 504â?13.

[17] W. Chen, Z. Mo, H. Xu, and et al, "Interference-aware Multiplexing for Deep Learning in GPU Clusters: A Middleware Approach," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SCâ?3)*, Denver, CO, USA, Nov. 2023.

[18] V. Saxena, K. R. Jayaram, S. Basu, and et al, "Effective Elastic Scaling of Deep Learning Workloads," in *Proceedings of the 28th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Nice, France, Nov. 2020.

[19] Q. Zou, Y. Deng, Y. Zhu, and et al, "Dissecting I/O Burstiness in Machine Learning Cloud Platform: A Case Study on Alibaba's MLaaS," in *Proceedings of the 38th International Conference on Massive Storage Systems and Technology (MSST'24)*, Santa Clara, CA, June 2024.

[20] Z. Li, W. Xing, S. Khamaiseh, and D. Xu, "Detecting saturation attacks based on self-similarity of OpenFlow traffic," IEEE Transactions on Network and Service Management, vol. 17, no. 1, pp. 607–621, Mar. 2020.

[21] Q. Liu, X. Zhao, W. Willinger, and et al, "Self-similarity in social network dynamics," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, vol. 2, 2016.

[22] S. Talluri, A. Luszczak, C. Abad, and A. Iosup, "Characterization of a Big Data Storage Workload in the Cloud," in *Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering (ICPE)*, April 2019, pp. 33–44.

[23] A. Yu, H. Yang, K. K. Nguyen, J. Zhang, and M. Cheriet, "Burst traffic scheduling for hybrid E/O switching DCN: An error feedback spiking neural network approach," IEEE Transactions on Network and Service Management, vol. 18, no. 1, pp. 882–893, Mar. 2021.

[24] T. Yu, H. Yang, Q. Yao, and et al, "Multi-visual-GRU-based survivable computing power scheduling in metro optical networks," IEEE Transactions on Network and Service Management, vol. 21, no. 1, pp. 1302–1315, February 2024.

[25] Q. Shi, F. Wang, and D. Feng, "IntFlow: Integrating per-packet and per-flowlet switching strategy for load balancing in datacenter networks," IEEE Transactions on Network and Service Management, vol. 17, no. 3, pp. 1377–1388, Sept. 2020.

[26] C. Ji, R. Pan, L. Chang, and et al, "Inspection and Characterization of App File Usage in Mobile Devices," ACM Transactions on Storage, vol. 16, no. 4, September 2020.

[27] J. Li, Q. Wang, P. Lee, and C. Shi, "An In-Depth Comparative Analysis of Cloud Block Storage Workloads: Findings and Implications," ACM Transactions on Storage, vol. 19, no. 2, 2023.

[28] M. Shahrad, R. Fonseca, I. Goiri, and et al, "Serverless in the Wild: Characterizing and Optimizing the Serverless Workload at a Large Cloud Provider," in *Proceedings of the 2020 USENIX Annual Technical Conference (ATC)*, July 2020, pp. 205–218.

[29] R. Gusella, "Characterizing the variability of arrival processes with indexes of dispersion," IEEE Journal on Selected Areas in Communications, vol. 9, no. 2, pp. 203–211, February 1991.

[30] Z. Li, C. Liang, W. He, and et al, "Metis: Robustly Optimizing Tail Latencies of Cloud Systems," in *Proceedings of the 2018 USENIX Annual Technical Conference (ATC)*, Boston, MA, USA, July 2018, pp. 981–992.

Authorized licensed use limited to: Zhejiang Lab. Downloaded on January 04,2026 at 01:40:45 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2025.3640771

[31] D. Tiwari, S. Gupta, J. Rogers, and et al, "Understanding GPU errors on large-scale HPC systems and the implications for system design and operation," in *Proceedings of the 21st IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, Burlingame, CA, USA, February 2015.

[32] J. Zhang, A. Sivasubramaniam, H. Franke, and et al, "Synthesizing representative I/O workloads for tpc-h," in *Proceedings of the 10th International Symposium on High Performance Computer Architecture (HPCA)*, Madrid, Spain, 2004.

[33] B. Schroeder, and G. A. Gibson, "Disk failures in the real world: what does an MTTF of 1,000,000 hours mean to you?" Proceedings of the 5th USENIX Conference on File and USENIX Association Storage Technologies (FAST'07), Berkeley, CA, 2007, pp. 1–16.

[34] J. Beran, R. Sherman, M. S. Taqqu, and W. Willinger, "Long-range dependence in variable-bit-rate video traffic," IEEE Transactions on Communications, vol. 43, pp. 1566–1579, March 1995.

[35] P. Embrechts and M. Maejima, "Self-similar processes," Princeton University Press, 2002.

[36] S. Gribble, G. Manku, and E. Brewer, "Self-similarity in high-level file systems: Measurement and applications," in *Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS/Performance)*, Madison, Wisconsin, 1998.

[37] J. P. Nolan, "Maximum Likelihood Estimation and Diagnostics for Stable Distributions," *LÃ©vy Processes*, pp. 379–400, 2001.

[38] Norros, "On the use of fractional brownian motion in the theory of connectionless networks," *IEEE Journal of Selected Areas in Communications (JSAC)*, vol. 15, pp. 200–208, 1997.

[39] M. W. Garrett, and W. Willinger, "Analysis, modeling and generation of self-similar VBR video traffic," in *Proceedings of SIGCOMMâ?4*.

[40] M. Jain, S. Ghosh, and S. P. Nandanoori, "Workload Characterization of a Time-Series Prediction System for Spatio-Temporal Data," in *Proceedings of the 19th ACM International Conference on Computing Frontiers (CF)*, Torino, Italy, May 2022.

[41] U. U. Hafeez, and A. Gandhi, "Empirical Analysis and Modeling of Compute Times of CNN Operations on AWS Cloud," in *Proceedings of the 2020 IEEE International Symposium on Workload Characterization (IISWC)*, October 2020, pp. 181–192.

[42] M. Wang, C. Meng, G. Long, and et al, "Characterizing Deep Learning Training Workloads on Alibaba-PAI," in *Proceedings of the 2019 IEEE International Symposium on Workload Characterization (IISWC)*, November 2019, pp. 189–202.

[43] G. Yeung, D. Borowiec, R. Yang, and et al, "Horus: Interference-Aware and Prediction-Based Scheduling in Deep Learning Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 1, pp. 88–100, 2022.

[44] D. Zhou, W. Pan, W. Wang, and T. Xie, "I/O characteristics of smartphone applications and their implications for eMMC design," in *Proceedings of the 2015 IEEE International Symposium on Workload Characterization (IISWC)*, October 2015, pp 12–21.

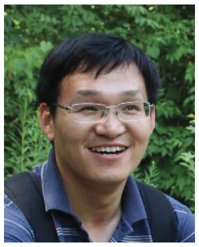
Qiang Zou received his Ph.D. in Computer Architecture and M.S. in Applied Mathematics from Huazhong University of Science and Technology (HUST) in 2009 and 2006 respectively. He is Associate Professor with the School of Artificial Intelligence, Guangxi Minzu University, Nanning, China. Before that, he worked as Associate Professor in the Department of Computer Science of the School of Computer and Information Science at Southwest University, Chongqing, China. His current research interests include workload characterization, modeling, performance evaluation, and applications of deep learning.

Yuhui Deng received the Ph.D. degree in computer science from the Huazhong University of Science and Technology, in 2004. He is currently a Professor with the Department of Computer Science, Jinan University. Before joining Jinan University, he worked with the EMC Corporation as a Senior Research Scientist from 2008 to 2009. He worked as a Research Ofï¬cer with Cranï¬eld University, U.K., from 2005 to 2008. His research interests cover green computing, cloud computing, information storage, computer architecture, and performance evaluation.

Yifeng Zhu is Professor in the Department of Electrical and Computer Engineering of the College of Engineering at the University of Maine. He received his Ph.D. and M.S. in Computer Science and Engineering from the University of Nebraska in 2005 and 2002 respectively. Before that, he received B.S. in Electrical Engineering from Huazhong University of Science and Technology in China. His current research interests include computer architecture and systems, data storage systems, energy-efficient memory systems, parallel/distributed computing, embedded systems, and applications of deep learning.

Yi Zhou received the BS and MS degrees in electronic engineering all from Beijing University of Technology, Beijing, in 2006 and 2010, respectively. He received the PhD degree in computer science from Auburn University, in 2018. He is currently an assistant professor with the TSYS School of Computer Science at Columbus State University. Prior to joining Columbus State University in 2018, he has been a software engineer in Alcatel-Lucent Technologies (China) Co., Ltd. for four years from 2010 to 2014. His research interests include energy-saving techniques, database systems, big data techniques and parallel computing.

Jianghe Cai received his BS degree in software engineering from the College of Computer Science and Technology at Huaqiao University, Xiamen, China, in 2021. He is currently pursuing the MS degree in Computer Architecture from the College of Information Science and Technology at Jinan University, Guangzhou, China. His research interests include Granular computing, data mining, and fuzzy computing.



Shuibing He received the PhD degree in computer science and technology from Huazhong University of Science and Technology, in 2009. He is now a ZJU100 Young Professor with the College of Computer Science and Technology, Zhejiang University, China. His research areas include intelligent computing, memory and storage systems, processing-in-memory. He is a member of the IEEE and ACM.

Lina Ge received the Ph.D. degree in computer science and technology from the South China University of Technology, Guangzhou, China, in 2009. Before that, she received the M.S. degree in computer science and technology from Guangxi University, Nanning, in 2004. She is currently a Professor with the School of Artificial Intelligence, Guangxi Minzu University, Nanning, China. Her research interests include intelligent computing and network security.

Authorized licensed use limited to: Zhejiang Lab. Downloaded on January 04,2026 at 01:40:45 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.