Dissecting I/O Burstiness in Machine Learning Cloud Platform: A Case Study on Alibaba's MLaaS

Qiang Zou[1,2], Yuhui Deng[3], Yifeng Zhu[4], Yi Zhou[5], Jianghe Cai[3], Shuibing He[6]
[1] *School of Artificial Intelligence, Guangxi Minzu University, Nanning, China*
[2] *Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis, Nanning, China*
[3] *Department of Computer Science, Jinan University, Guangzhou, China*
[4] *Department of Electrical and Computer Engineering, University of Maine, Orono, ME, USA*
[5] *TSYS School of Computer Science, Columbus State University, Columbus, GA, USA*
[6] *College of Computer Science and Technology, Zhejiang University, Hangzhou, China*

Abstract—**With advancements in machine learning (ML) technology and the availability of large ML-as-a-Service (MLaaS) clouds, accurately understanding the I/O behaviors in the storage subsystem of an MLaaS cloud platform is crucial for resource scheduling and optimization. This study provides valuable insights into the correlation of I/O request arrivals in a representative and dynamic MLaaS workload – Alibaba PAI (an ML platform for artificial intelligence). Regarding the I/O requests in the PAI workload at the machine level, our burstiness diagnosis reveals that the I/O arrival process exhibits significant bursts. Additionally, our Gaussianity test indicates that the bursty activities in PAI are non-Gaussian. Our findings highlight the existence of a certain level of correlation between I/O request arrivals on long-term time scales. Furthermore, we uncover the self-similar nature of I/O activities in the Alibaba PAI machine-level MLaaS workload through visual evidence, the auto-correlation structure of the aggregated I/O request sequence, and Hurst parameter estimates. Moreover, we create self-similar workload models to synthesize I/O request series based on the inputs measured from the PAI trace. Experimental results demonstrate that both FARIMA and alpha-stable models outperform existing models in accurately simulating burstiness.**

Index Terms—**MLaaS I/O workload, burst, correlation, self-similar, workload synthesis**

I. Introduction

With ongoing advancements in ML technology, deep learning (DL) methods are widely used in various scientific fields, driving scientific discovery and innovation in areas such as autonomous driving, drug development, and image processing. To meet the increasing computation demands of ML workloads, major tech companies have invested in expansive MLaaS clouds, equipped with high-end hardware, such as GPUs, to handle various types of ML tasks [1].

The storage subsystem plays a fundamental role in MLaaS platforms, including essential functionalities such as data ingestion, data access, and scalability. Understanding request behaviors in the storage subsystem is crucial for resource scheduling and optimization while accommodating growing storage needs and maintaining performance and reliability. To optimize resource scheduling and management of GPU clusters, it is essential to gain insights into job characteristics and user behaviors in ML application workloads [2]. Analyzing real workloads and employing stochastic models to accurately characterize the workload is a judicious first step in understanding arrival characteristics [3].

Recent researchers have started using traditional distributions, such as uniform [4] and exponential [5], to approximate the description of request features in computation-intensive ML workloads. However, ML researchers have found that burst phenomena are prevalent in both ML and DL workloads [1], [2], [6], which even show heavy-tailed features [1], [7]. Traditional distributions mentioned above fail to capture the burst behavior, presenting new challenges in understanding request behaviors in ML application workloads.

Motivated by these observations, we analyze workload traces [1] obtained from the ML Platform for Artificial Intelligence (PAI) – an integrated MLaaS platform provided by Alibaba Cloud. The PAI workload consist of a mixture of training and inference jobs using state-of-the-art ML algorithms. For the PAI workload at the machine level, we diagnose I/O burstiness by conducting a Gaussian test [8] for the I/O request sequence, and investigating the correlation of I/O request arrivals. For the PAI workload with certain degrees of correlation, we present visual and theoretical evidence for the existence of self-similarity and estimate the self-similarity parameter using statistical tools. Based on the inputs measured from the PAI trace, we further employ synthetic models to accurately generate the I/O request sequence in the machine-level PAI workload. To the best of our knowledge, there have been few research works reported in the literature on this particular topic.

This study makes the following four contributions:

(1) We diagnose the burstiness of I/O activities in the machine-level PAI workload and reveal that the arrival process of I/O requests is highly bursty. We investigate the correlation between I/O request arrivals in the bursty PAI workload. Our findings uncover that there is a certain degree of correlation between I/O arrivals, making the use of a traditional distribution unsuitable to describe request activities. Consequently, these findings assist researchers in accurately describing the burst behaviors in PAI.

(2) We perform a Gaussianity test for the I/O request

sequence in the PAI workload at the machine level. The test results show that the burst activities in the PAI workload appear to be non-Gaussian. This surprising observation differs significantly from previous studies that employed Gaussian methods to analyze the traced data obtained from Alibaba's production clusters [5]. These findings assist researchers in accurately describing the burst behaviors in PAI.

(3) We discover that I/O activities in the machine-level PAI workload exhibit a self-similar nature, as shown through visual evidence, the auto-correlation structure of an aggregated process of request sequence, and Hurst parameter [8] estimates.

(4) Based on the inputs measured from PAI, we employ self-similar models to generate synthetic I/O request sequences for PAI at the machine level. Experimental results demonstrate that both FARIMA and alpha-stable models [8] outperform existing models in terms of accuracy in emulating burstiness.

The rest of this paper is organized as follows. Section II provides an overview of the Alibaba PAI trace investigated throughout this work and summarizes related research studies. Section III presents the characterization of burstiness in the machine-level I/O request arrival process, the Gaussianity test for I/O requests, and an elaboration on the correlation of the I/O request arrivals in the machine-level PAI workload. Section IV presents both visual and statistical evidence of the existence of self-similarity in I/O activities on PAI. Section V articulates the implementation of several self-similar workload models to synthesize request series for PAI. Section VI discusses the importance of gaining insights into the workload characteristics of ML applications. Finally, Section VII concludes this paper.

II. Background and Motivation

A. Alibaba PAI

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 covering the entire ML pipeline (see Figure 1 for an architectural overview). In PAI, users submit ML jobs, including training and inference, which are developed in various frameworks, such as TensorFlow [9], MXNet [10], and Graph-Learn [11]. Meanwhile, users specify required computational resources, such as GPUs, CPUs, and memory. Each job is then split into multiple tasks with different computational roles: parameter server (PS), workers for training jobs, and estimators for inference jobs. Each task consists of multiple instances. Due to cost considerations, only some subclusters in PAI are equipped with NVLink in multi-GPU servers. Since PAI's deployment, it has attracted tens of thousands of enterprises and individual developers, becoming one of the leading MLaaS platforms in China.

Alibaba's PAI is composed of several Alibaba Cloud services, including Elastic Compute Service (ECS), etc. Together, these components provide elastic capacity that can be utilized during I/O burst scenarios. Specifically, the scalable storage solutions offered by Alibaba's PAI rely on four key components: Alibaba Cloud's OSS, Distributed File System

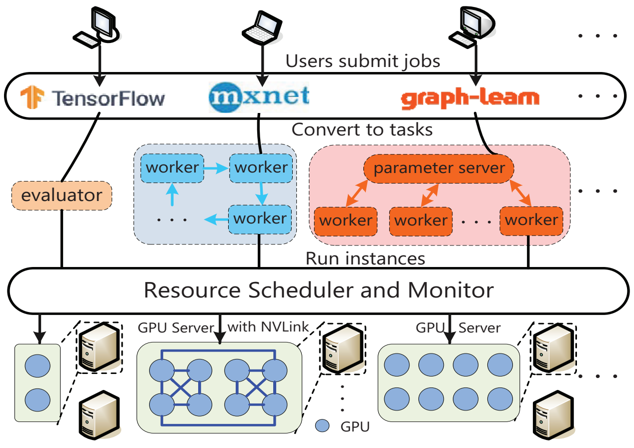


Fig. 1. PAI architecture overview.

(DFS), Alibaba Cloud's database solutions (ADBs), and Alibaba Cloud's Elastic Block Storage (EBS). Among these components, OSS is used to store training data and other machine learning assets, DFS (including Network Attached Storage and Alibaba Cloud File Storage) provides scalable and shared storage resources for machine learning workloads, ADBs (such as ApsaraDB, PolarDB, and AnalyticDB) offer scalable and managed database services, and EBS provides block-level storage volumes that can be attached to ECS instances. PAI trace contains requests from all the listed storage solutions.

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

To address the challenges of load balancing and long queuing delays caused by cluster scheduling on heterogeneous machines, it is crucial to gain insights into the impacts of characteristics of workloads on these heterogeneous machines. To achieve this, Weng *et al.* [1] conducted training and inference jobs using state-of-the-art ML algorithms on PAI in the second half of 2020. They collected application workload traces over a two-month period on machines with V100M32 and V100 GPUs with NVLink, which are summarized in Table I. These traces at the job, task, and instance levels provide launch information, while the machine-level trace contains information, such as timestamps of every request, I/O waiting times (iowait), execution times in user and kernel modes, etc. A more detailed description of the PAI trace can be found in the referenced literature [1]. 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.

Previous studies have shown that request behaviors in ML application workloads are bursty and exhibit heavy-tailed features [1]. In general, request bursts often exist on different time scales, and traditional methods (e.g., exponential) will gradually become smooth at long time scales, making it difficult to accurately describe the burst behaviors in request activities [12], posing significant challenges in accurately understanding request activities in typical ML workloads and establishing best practices for platform optimization.

These observations inspire us to examine the feasibility and effectiveness of using traditional methods, such as uniform [4] and exponential [5] distributions to describe request behaviors in typical ML workloads. Before doing so, we must address the following critical issues: Is it appropriate to use independently and identically distributed methods to describe the bursts and heavy-tailed behaviors in MLaaS I/O workloads? 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 machine-level PAI workload and diagnose the temporal behaviors in Section III.

B. Related Work

Recently, the research community has been working to improve the design of clusters based on the characteristics of ML application workloads. In this section, we will focus on previous works most relevant to this study.

Workload Characterization. Paul *et al.* [6] analyzed the workload characteristics of ML I/O jobs running on a large-scale supercomputer to understand how I/O behaviors vary across different scientific fields and workload scales. Their analysis provided insights into the usage of parallel file systems and burst buffers. Wang *et al.* [15] developed an analysis framework to study detailed execution time breakdowns to identify performance bottlenecks. Although they described training performance under various software frameworks and hardware configurations, their focus was primarily on training workloads, without considering general DL workloads.

Since deep neural networks (DNN) are typically trained on GPUs, Jeon *et al.* [7] analyzed the workload characteristics of a two-month-long trace from Microsoft's multi-tenant GPU clusters. Their aim was to improve cluster utilization for DNN training workloads in multi-tenant clusters. Li *et al.* [4] characterized system operations, job characteristics, user behaviors, and trends on contemporary GPU-accelerated production HPC systems. With this analysis, Li sought to aid in revamping the design of GPU-based large-scale systems for emerging application workloads, such as AI and machine learning. By analyzing the real job trace from SenseTime, Hu *et al.* [2] conducted a study on the characteristics of DL job and resource management. This led them to develop a priority scheduling scheme and a cluster energy-saving strategy.

Moreover, Weng *et al.* [1] proposed a solution by studying ML workload features on PAI to address cluster scheduling issues such as long queuing delays, and load imbalance across heterogeneous machines. *Although they mentioned that the request behaviors in the PAI workload are bursty and even exhibit heavy-tailed properties, Weng et al. did not further investigate and characterize the burstiness of I/O request behaviors in the machine-level PAI workload. Thus, this study aims to bridge this gap by providing an in-depth analysis.*

Self-similarity. Self-similarity means 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 cloud workloads [13], social network dynamics [16], and internet traffic [14]. For instance, Gupta *et al.* [13] studied Google cluster traces and found self-similarity and heavy-tailed behaviors in cloud workloads using auto-correlation analysis and R/S analysis. Liu *et al.* [16] 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.* [17] conducted a statistical analysis of file popularity, read size, arrival interval, etc. using Spark data collected over six months. Their findings showed that read operations exhibit heavy tails, bursts, and negative long-range dependence. *Given the distinctive I/O mode of machine learning platforms* [6], *these observations prompt us to investigate whether self-similarity exists in the PAI I/O workload, particularly in representative MLaaS workloads that exhibit bursty and heavy-tailed request behaviors* [1].

Based on the real-world machine-level PAI workload[1], this study focuses on the following three aspects. First, it examines the appropriateness of characterizing I/O behaviors with traditional distributions. Second, it explores the presence of self-similarity. Lastly, it delves into the synthesis of I/O request sequences.

III. PAI Workload Diagnosis

This section aims to provide a thorough understanding of the I/O request activities in Alibaba's MLaaS cloud platform by diagnosing the PAI workload.

A. Burstiness

To gain a deep understanding of the access features in system workloads, it is useful to analyze the arrival mode. A workload is considered bursty if its request arrival process X is *non-stationary* and has a large *variance*. Previous research has shown that "burstiness" is prevalent in various system workloads, such as Ethernet [12], mobile storage [18], and cloud block storage [19].

To characterize the I/O burstiness in the PAI workload, this study extracts the request arrival intervals from the PAI trace of about two months and analyzes the corresponding empirical distribution patterns. To visualize the features present in the arrival pattern, we present the *cumulative distribution function* (CDF) of the request arrival intervals in Figure 2. The horizontal axis in Figure 2 represents I/O request arrival intervals in seconds, while the vertical axis denotes the proportion of the corresponding value in the entire dataset. A point (x, y) on the distribution curve indicates that the proportion of I/O

[1]Unless specifically stated otherwise, the PAI workload (or trace) referred to in the subsequent sections pertains to the machine-level

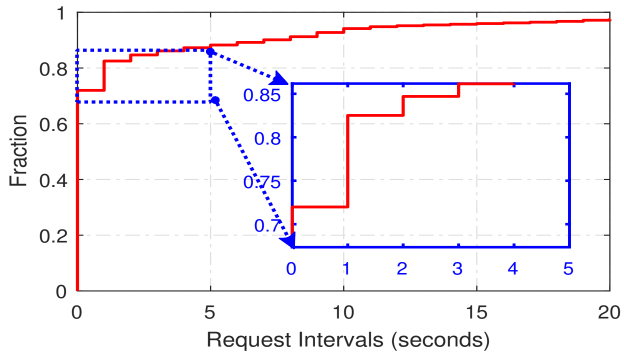


Fig. 2. Empirical CDF of request arrival intervals in the PAI workload.

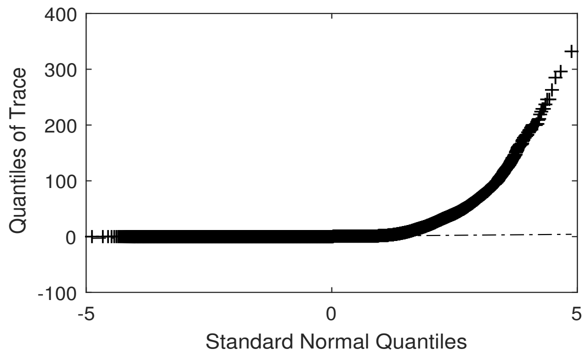


Fig. 3. Examine the Gaussianity of I/O request activities in the PAI workload through QQ plot of the PAI trace data versus standard normal, respectively.

arrival intervals less than or equal to x in the corresponding time series is y.

Non-stationary. In order to clearly illustrate our empirical observations, we have magnified a specific region outlined by the blue box in Figure 2. From this, it is apparent that approximately 83% of I/O requests arrive within an interval of no more than 1 second. Furthermore, up to 72% of I/O request arrival intervals are observed to be 0 seconds. These findings indicate that the majority of I/O requests in the PAI workload arrive at peak moments, resulting in the aggregation effect of I/O request arrivals and making the entire I/O arrival process non-stationary.

Variance. In addition, for I/O arrival intervals in the PAI workload, our statistical findings show that the variance of the corresponding time series is as high as 8892, which satisfies another necessary condition for categorizing the PAI workload as a burst workload. Based on the aforementioned observations and analyses, it can be affirmed that the concept of "burstiness" accurately describes the high variability of the I/O arrival process in the PAI workload.

Previous studies have demonstrated that the strength of burstiness can be quantified by the *index of dispersion for intervals* (IDI) [20]. Specifically, a larger value of the index of dispersion indicates stronger burstiness. Therefore, in this study, we employ the index of dispersion to measure the strength of burstiness. By applying the following formula,

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

we calculate the IDI for I/O arrivals in the PAI workload as 1519. This value suggests that the I/O arrivals in the PAI workload exhibit a significant bursty pattern.

Now, let us recap this major observation as

Finding (1): I/O request arrival process in the PAI workload exhibits significant burstiness.

Implications: 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 Test

Previous research [5], [21] suggests that adherence of a system workload to the Gaussian property is beneficial when building performance evaluation models. In particular, conducting a Gaussianity test helps accurately describe the tail trend in the distribution of access characteristics and construct a convincing model. Therefore, we perform a Gaussianity test to study the I/O request sequence in the PAI workload.

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

For the I/O request activities in the PAI workload, Figure 3 depicts the corresponding QQ plot of PAI trace data compared to a standard normal distribution. As shown in Figure 3, the scatter points corresponding to the PAI trace data mentioned above clearly do not fall on a straight line; instead, the curve is concave upward, indicating a heavy-tailed trend. This suggests that the I/O behaviors in the PAI workload are non-Gaussian.

This observation is surprising as it differs significantly from the Gaussian distribution previously used to analyze Alibaba's production cluster data [5]. The reason behind this phenomenon may be the presence of a greater number of request bursts in the PAI workload when running state-of-the-art ML algorithms.

Here, we summarize the key observations from our Gaussianity test as follows:

Finding (2): I/O request behaviors in the PAI workload appear be non-Gaussian and heavy-tailed.

Implications: During busy periods of heavy-tailed behaviors, Alibaba's MLaaS platform can allocate additional resources to handle increased computing demands, thereby reducing task latency.

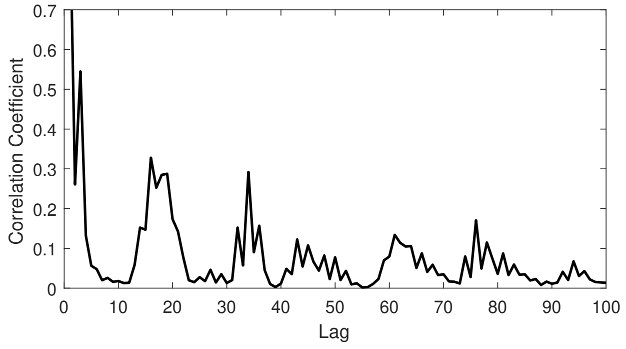


Fig. 4. Auto-correlation function of I/O request arrivals in the PAI workload.

C. Correlation Analysis

In this section, we use the *auto-correlation function* (ACF) [13], [22], [23] as a statistical tool to diagnose the correlation of I/O request arrivals in the PAI workload.

Let $Y = \{Y_t : t = 1, 2, \cdots, n\}$ to be a time series with an expectation $\theta = E[Y_t]$. Each time interval (or *lag*) is denoted as k. Each lag corresponds to an auto-correlation coefficient, denoted as $R(k)$, which is independent of the time itself. By defining $y_t = Y_t - \theta$, we obtain $\{y_t : t = 1,\ 2, \cdots\}$. Therefore, the auto-correlation function with a lag of k ($k \geq 0$) can be defined as:

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

Here, $R(k)$ represents the correlation coefficient at the independent variable k. This equation (Equation 2) measures the interrelationship between two adjacent elements in a time series. For a detailed description of ACF, please refer to [22].

The trend of the autocorrelation function curve is closely related to the patterns of I/O request activities in the PAI workload. Specifically, as the lag increases, the correlation coefficient of the inter-arrival interval rapidly decreases and approaches zero. This signifies that the correlation of I/O request behaviors in PAI becomes less significant over time. In other words, the burstiness in the PAI workload gradually smooths out over time, making it reasonable to use an independently identically distributed model to characterize I/O request activities. However, if the correlation coefficient does not rapidly approach zero, there will be a certain degree of correlation between I/O request arrivals in the PAI workload. In such cases, considering other approaches, such as long-range dependence (also known as self-similarity), becomes necessary to accurately describe the I/O behaviors in PAI.

Figure 4 illustrates the auto-correlation functions of arrival intervals for I/O requests in the PAI workload. As the lag increases from 0 to 100, the correlation coefficients of I/O requests do not approach zero sharply; instead, they exhibit a gradual declining trend. We can also observe that at some lags (e.g., 15, 35, etc.), the coefficients increase, suggesting that the correlation persists over longer intervals. These findings provide evidence for the existence of a correlation between I/O request arrivals in the PAI workload, making the use of an independently identically distributed method unsuitable to describe I/O request activities. In short, I/O request activities in PAI demonstrate not only bursts and a heavy-tailed distribution but also correlations across extended time scales.

The observations mentioned above show that when state-of-the-art ML algorithms are executed on PAI, there are distinct characteristics of I/O behaviors at the machine level. These characteristics include not only burstiness but also a noticeable degree of correlation. The correlation of request arrivals in the PAI workload motivates us to investigate self-similarity in the following section.

We summarize the important insight about the correlation between I/O requests as follows:

Finding (3): There is observable correlation between I/O requests in PAI.

Implications: Exploring self-similarity in the PAI workload becomes essential to accurately understand request behaviors and improve performance in PAI.

IV. Self-similarity Study

In this section, we will explore the self-similarity of the PAI workload in the following three aspects: (1) showing the visualization, (2) providing theoretical evidence through the auto-correlation structures of the aggregation processes of I/O request series, and (3) estimating the self-similarity parameter using classical tools.

A. Self-similar Process

First, let us outline the self-similar process involved in this study. A detailed introduction can be found in the literature [24].

Considering a stochastic process $X = \{X_t : t = 1,\ 2,\ 3, \cdots\}$, $X^{(m)} = \{X_t^{(m)} : t = 1,\ 2,\ 3, \cdots\}$ is then 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 auto-correlation function (ACF) corresponding to $X^{(m)}$ as $R^{(m)}(k)$. For traditional stochastic processes such as Poisson, the ACF for $X^{(m)}$ *degenerates* with an increasing m, and converges to 0, that is,

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

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 will 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 will be 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*.

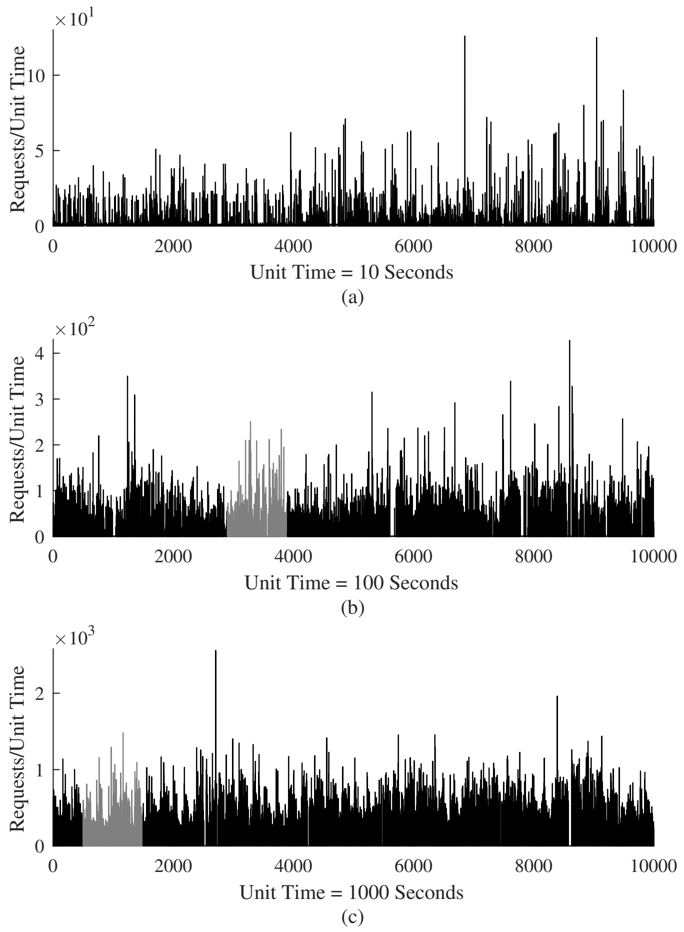


Fig. 5. Visualization of I/O request sequence in the PAI workload. The plots (a)-(c) illustrate the number of requests per unit time for three different time scales, respectively. Each plot has ten thousand buckets.

Lemma 1 [25]. For a stochastic process X with covariance stationarity, X is self-similar, and the two propositions below are equivalent if process X matches any of these two:

(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

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

B. Visualization

A self-similar workload is mainly characterized by the persistence of bursts and burst aggregations at various timescales, with the pattern itself showing similar traits.

Therefore, we examine I/O request behaviors in the PAI workload at different timescales and observe similar patterns of activity. To visually illustrate our findings, we present I/O request sequences at three different timescales in Figure 5, consisting of subplots (a)-(c), with each subsequent timescale being ten times larger than the previous one. The horizontal axis represents the timescale, and the vertical axis shows the number of requests per unit of time.

Each subplot within Figure 5 is derived from a subinterval randomly selected from the time range depicted in the following subplot and it enhances the temporal resolution by a factor of 10. For instance, subplot (a) delineates a brief span (100,000 seconds) randomly sampled from subplot (b), which, in turn, represents a concise interval (1,000,000 seconds) randomly chosen from subplot (c).

As shown in Figure 5, each subplot displays multiple "spikes" or bursts, which are request activities characterized by significant fluctuations – larger bursts mingling with smaller ones. These "spikes" within the PAI workload consistently follow the same pattern across the three subplots, regardless of the timescale. These patterns suggest that the bursty I/O request activity in PAI spans across nested subintervals, each exhibiting bursty behavior, and these subintervals, in turn, display similar dynamics. This indicates that the sequences of I/O requests in PAI exhibit self-similarity over extended periods of time. This important observation can be summarized as follows.

Finding (4): The time range characterized by bursty requests consists of nested subintervals, each is made of even smaller subintervals with similar burst behaviors.

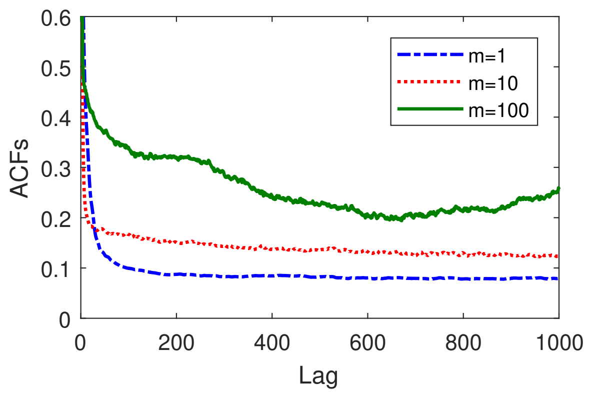
Implications: I/O requests in the PAI workload exhibit characteristics of self-similarity. This offers an opportunity to enhance load balancing by incorporating the factor of self-similarity when dynamically distributing computing tasks, thereby avoiding bottlenecks during bursty periods.

C. Theoretical Evidence

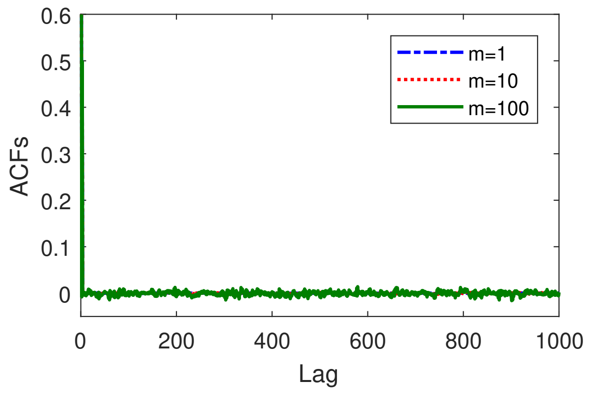
In this section, we focus on the theoretical evidence that supports the presence of self-similarity in the PAI workload. The statements regarding the structure of $R^{(m)}(k)$ in Section IV-A provide a theoretical basis for detecting the existence of self-similarity. In other words, self-similar workloads exhibit burstiness at different time scales due to their similar scale-invariant properties.

For I/O requests in the PAI workload, we examine the auto-correlation functions of the aggregated time series of the I/O request sequence at multiple aggregation levels. Specifically, we draw the auto-correlation functions of the time sequences at three aggregation levels in Figure 6(a) and plot the auto-correlation functions of the corresponding artificial Poisson workloads in Figure 6(b). The aggregation levels (m) are 1, 10, and 100, respectively.

As seen in Figure 6(a), as the lag increases from 1 to 1000, the correlation coefficients of the I/O request sequence fluctuate and do not approach 0. Consequently, the corresponding I/O request behaviors exhibit long-range dependence. We can also observe that the auto-correlation curves at various aggregation levels appear to converge to a similar function



(a) PAI workload



(b) Poisson workload

Fig. 6. Autocorrelation functions of the aggregated time series for (a) I/O request sequence in the PAI workload, and (b) artificial Poisson workload.

structure. In contrast, Figure 6(b) demonstrates that the auto-correlation coefficients of artificial Poisson workloads at each aggregation level are generally very small and almost equal to zero. These observations reveal that the auto-correlation structure of the I/O request sequence in the PAI workload behaves similarly to a self-similar process, which is quite different from a Poisson series.

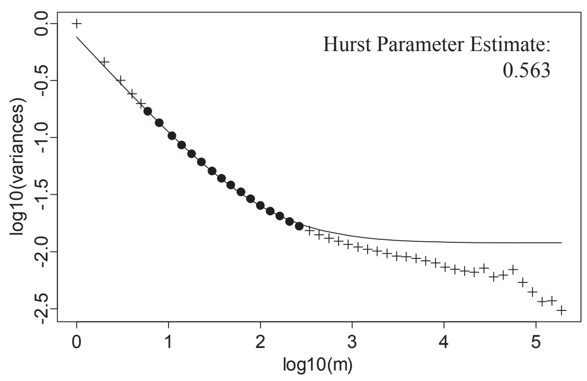
In summary, I/O request activities in the PAI workload behave like a self-similar process. We summarize this key observation as follows.

Finding (5): The auto-correlation curves of the I/O sequence at different aggregation levels seem to converge to a similar function structure.

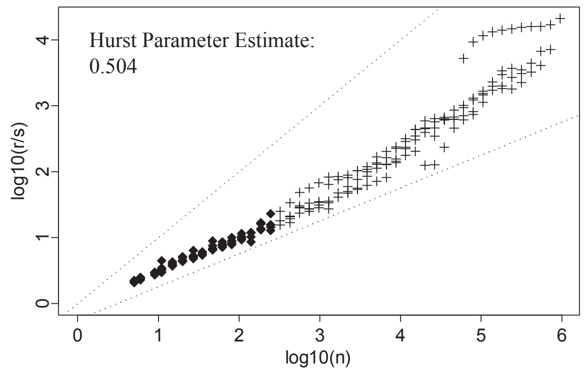
Implications: I/O request activities in the PAI workload behave like a self-similar process. Incorporating the knowledge of self-similarity into workload modeling can better forecast resource requirements for ML tasks.

D. Estimating Hurst Parameter

In this part of the study, we use the Hurst parameter to quantitatively estimate self-similarity. The Hurst parameter signifies the degree of self-similarity within an open interval



(a) The variance-time plot



(b) The Pox plot

Fig. 7. Estimating Hurst parameters for I/O requests in the PAI workload by (a) Variance-time plot, and (b) Pox plot, respectively.

from 0.5 to 1: the greater the Hurst parameter, the higher the degree of self-similarity. For the I/O request sequence in the PAI workload, we use two well-known analytical tools – the *variance-time plot* [12] and the *R/S analysis* (also called *Pox plot*) [26] – to estimate the Hurst parameters in Figure 7. Both the variance-time plot and the Pox plot are widely used to estimate the Hurst parameter for accurate 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:

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

where the constant $\log_{10}(\alpha^2)$ is independent of m. If we consider $\log_{10}(Var(X^{(m)}))$ as a function of $\log_{10}(m)$, we can plot a curve of $\log_{10}(Var(X^{(m)}))$ versus $\log_{10}(m)$ to obtain a series of scattered points, which can be approximately fit with a linear regression line having a slope of $-\lambda = 2(H-1)$. This allows us to calculate the Hurst parameter H.

We plot the variance-time plot for I/O request activities in the PAI sample workload in Figure 7(a), where $\log_{10}(Var(X^{(m)}))$ is abbreviated as $\log_{10}(variances)$. Figure 7(a) shows a line with a slope of $-\lambda = 2(H-1)$ by representing the scatter of $\log_{10}(variances)$ versus $\log_{10}(m)$. The estimated Hurst parameter for this case is 0.563. Meanwhile, Figure 7(b) illustrates the Pox plot generated from the R/S analysis of the PAI sample trace. As shown in Figure 7(b), the Hurst parameter is estimated to be 0.504.

The aforementioned observations indicate that the Hurst parameter estimates for I/O requests in the PAI workload are all greater than 0.5, which quantitatively confirms the presence of self-similarity. We summarize this key observation as follows.

Finding (6): For the PAI workload, 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 when evaluating and optimizing the performance of Alibaba's MLaaS platform.

V. Workload Synthesis

Synthetic I/O workloads are used to simulate request sequences and evaluate I/O performance in actual storage systems. By diagnosing burstiness, conducting correlation studies, Gaussian tests, and self-similarity analysis of the PAI workload, we have made the following findings: (1) The arrival process of I/O requests is highly bursty. (2) Traditional methods struggle to accurately characterize the PAI workload, as the I/O arrivals show a certain degree of correlation. (3) There seems to be self-similarity in the PAI workload. (4) The I/O request activities in PAI appear to be non-Gaussian.

These findings inspire us to use several methods to generate I/O request series for the self-similar PAI workload.

A. *Trimmed Mean of Errors*

To synthesize I/O request sequences in the PAI workload, we employ two typical self-similar workload models – fractional Brownian motion (FBM) [27] and fractional autoregressive integrated moving average (FARIMA) [28]. These models are chosen due to the existence of self-similarity in I/O activities. The FARIMA model is well-known for its ability to describe both long-range and short-range dependences, while the FBM model is adept at characterizing self-similarity under Gaussian conditions.

In order to faithfully describe the bursts and heavy-tailed properties under non-Gaussian conditions, we extend the versatile alpha-stable model [8] by redefining its model parameters to generate I/O request series for PAI. All the inputs for these models can be measured and obtained from the real PAI trace dataset. Specifically, for the PAI trace, we use the maximum likelihood estimation to measure the trace dataset. This enables us to derive parameter estimates for the aforementioned workload models, and generate synthetic I/O request sequences for PAI.

To evaluate the error between the synthetic I/O request sequence and the actual one, we use the trimmed mean of errors [8]. The trimmed mean is the arithmetic average after truncating a smaller proportion of data at both ends of a sample, which is more robust to outliers compared to the conventional sample mean (i.e., the arithmetic average). The smaller the trimmed mean of the errors, the higher the matching degree between synthetic sequence and real one.

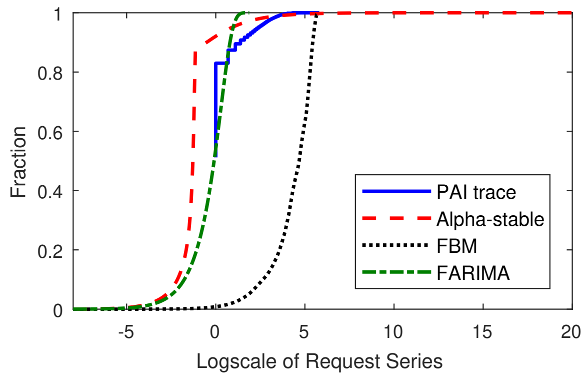


Fig. 8. Comparison of CDFs between synthetic I/O sequence and real one for PAI.

The trimmed mean of errors for FBM, FARIMA, and alpha-stable models is 109.46, 1.26, and 0.78, respectively. In other words, the trimmed mean of the errors between the alpha-stable synthetic sequence and the real one is minimal. This improves the matching degree of FBM and FARIMA by 99% and 38%, respectively. However, the trimmed mean of the errors for the alpha-stable synthetic sequence (i.e., 0.78) is very close to that for the FARIMA synthetic one (i.e., 1.26). This observation indicates that both the FARIMA and alpha-stable approaches can faithfully generate the synthetic sequence for the PAI workload.

B. *An Empirical Study*

To intuitively compare the matching degree of the various synthetic I/O request sequences with the actual one in the PAI workload, we depict the cumulative distribution functions (CDF) of the actual series and the synthetic ones in Figure 8.

In Figure 8, the horizontal axis represents the log scale of the number of requests per unit time, while the vertical axis represents the proportion of the corresponding values in the entire request sequence. Each coordinate (x, y) indicates that the proportion of logarithm values less than or equal to x in the corresponding time series is y.

As shown in Figure 8, the alpha-stable synthetic sequence matches the actual sequence well for the PAI workload, and this is corroborated by the corresponding trimmed mean of errors of 0.78.

Furthermore, Figure 8 demonstrates that the FARIMA synthetic I/O sequence also closely matches the actual sequence, displaying a comparable level of precision to the alpha-stable synthetic sequence. This supports the observations presented in Section V-A.

However, one advantage of the alpha-stable synthetic sequence over the FARIMA synthetic sequence is its ability to better capture the heavy-tailed feature of the actual sequence.

Moreover, Figure 8 reveals that both the alpha-stable and FARIMA synthetic sequences have significantly better matching degrees than the FBM series. This suggests that the I/O request behaviors in the PAI workload do not conform to the Gaussian condition. This observation aligns perfectly with the test results presented in Section III and lends support to our Gaussianity test results.

To summarize, for the PAI workload, both the FARIMA synthetic sequence and the alpha-stable synthetic sequence exhibit 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 patterns in the execution stack [29]. In this section, we will discuss the significance of characterizing ML application workloads when building an MLaaS platform.

A. Performance Evaluation

Characterizing the access patterns of ML workloads is an effective way to evaluate the performance of ML clusters. For instance, Paul *et al.* [6] analyzed the I/O behaviors of ML workloads by using a Darshan log of 23,000 HPC ML I/O jobs. Paul's analysis provided insights into potential performance optimization efforts and supported the assessment of I/O trends across an entire HPC cluster. Li *et al.* [4] studied the 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 engage in low-utilization development and idle work. Based on an operation-level empirical analysis of various CNNs, Hafeez *et al.* [30] proposed a model-driven method called *Ceer* to determine the optimal GPU instance(s) for any given CNN.

Furthermore, Wang *et al.* [15] 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.* [31] proposed a predictive resource manager based on GPU utilization of heterogeneous DL jobs inferred from computational graph features of DL models. Weng *et al.* [1] characterized a two-month-long workload trace collected from an Alibaba production model cluster comprising more than 6,000 GPUs. They revealed challenges of heterogeneous GPU clusters, such as low GPU utilization, long queuing delay, and load imbalance across heterogeneous machines. Hu *et al.* [2] designed a quasi-shortest service priority scheduling service that strives to minimize the average job completion time of clusters by analyzing a real job trace from SenseTime. 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. This analysis provided a design guide for next-generation cluster schedulers. Jiang *et al.* [5] investigated the cluster-trace-v2018 trace obtained from an Alibaba production cluster. Jiang's study observed daily periodic fluctuations in workload characteristics in the production clusters and suggested that performance bottlenecks in collocated clusters are caused by the memory system, which can be used by data centers to establish more efficient scheduling strategies.

C. Synthetic Model

Trace-driven simulators, often used to evaluate GPU cluster performance in deep learning systems [31], 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 characteristics. Second, by providing workloads for various isolated subsystems, synthetic models excel in evaluating specific subsystems rather than the entire system. Third, modifying input parameters enables synthetic models to perform hypothetical performance analysis, which facilitates the identification of key factors influencing outcomes.

The modeling method directly affects performance evaluation and system design verification. In this study, the FARIMA and alpha-stable synthetic models based on the inputs measured from the PAI trace, can faithfully capture the characteristics of the PAI workload. 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 like Amazon SageMaker, Google Cloud AI Platform, and Microsoft Azure Machine Learning, our proposed method is applicable for studying the burstiness of requests in other MLaaS platforms.

VII. Conclusion

With the continuous advancements and widespread usage of machine learning technologies, major tech companies are deploying MLaaS clouds equipped with GPU clusters to run various types of ML application workloads. It is crucial to have a deep understanding of the request behaviors in MLaaS workloads in order to effectively schedule and manage the I/O subsystem in GPU clusters.

To achieve this, we investigated the burstiness of the I/O requests in a representative and real-world MLaaS workload – the PAI workload. Our findings revealed that the arrival process of I/O requests is highly bursty. We conducted tests for Gaussianity and discovered that the bursty I/O activities in PAI appear to be non-Gaussian. Additionally, we analyzed the correlation of I/O request arrivals and found that there are correlations among the I/O requests in the PAI workload. These discoveries led us to uncover the self-similar nature of the PAI I/O workload through visual evidence, the autocorrelation structure of an aggregated process of I/O request sequences, and Hurst parameter estimates.

Furthermore, based on the inputs measured from real trace data, we implemented self-similar workload models to synthesize I/O request sequences for the PAI workload. The experimental results demonstrate that both the FARIMA and alpha-stable models can accurately capture the workload characteristics of PAI.

Acknowledgments

We would like to thank the anonymous reviewers and our shepherd, Dr. Vasily Tarasov, for their helpful comments in reviewing this paper. 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 Grant 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, W Wang, C Wang, J He, Y Li, L Zhang, W Lin, Y Ding, "MLaaS in the Wild: Workload Analysis and Scheduling in Large-Scale Heterogeneous GPU Clusters," Proceedings of the 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI), Renton, WA, 2022.

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

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

[4] B Li, R Arora, S Samsi, T Patel, W Arcand, D Bestor, C Byun, RB Roy, B Bergeron, and et al, "AI-Enabling Workloads on Large-Scale GPU-Accelerated System: Characterization, Opportunities, and Implications," Proceedings of the 28th Annual IEEE International Symposium on High-Performance Computer Architecture (HPCA), 2022, pp. 1224–1237.

[5] C Jiang, Y Qiu, W Shi, Z Ge, J Wang, S Chen, C Cérin, Z Ren, G Xu, J Lin, "Characterizing Co-located Workloads in Alibaba Cloud Datacenters," IEEE Transactions on Cloud Computing, vol. 10, pp. 2381–2397, 2020.

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

[7] M Jeon, S Venkataraman, A Phanishayee, J Qian, W Xiao, F Yang, "Analysis of Large-Scale Multi-Tenant GPU Clusters for DNN Training Workloads," Proceedings of the 2019 USENIX Annual Technical Conference (ATC), Renton, WA, USA, 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," Proceedings of the 2008 IEEE International Conference on Cluster Computing (CLUSTER), 2008.

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

[10] T Chen, M Li, Y Li, M Lin, N Wang, M Wang, T Xiao, B Xu, C Zhang, Z Zhang, "MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems," https://doi.org/10.48550/arXiv.1512.01274, 2015.

[11] R Zhu, K Zhao, H Yang, W Lin, C Zhou, B Ai, Y Li, J Zhou, "AliGraph: a comprehensive graph neural network platform," 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 AD Dileep, "Long range dependence in cloud servers: a statistical analysis based on Google workload trace," Computing, vol. 102, pp. 1031–1049, April 2020.

[14] Q Li, S Wang, Y Liu, H Long, J Jiang, "Traffic self-similarity analysis and application of industrial internet," Wireless Networks, July 2020.

[15] M Wang, C Meng, G Long, C Wu, J Yang, W Lin, Y Jia, "Characterizing Deep Learning Training Workloads on Alibaba-PAI," Proceedings of the 2019 IEEE International Symposium on Workload Characterization (IISWC), 2019, pp. 189–202.

[16] Q Liu, X Zhao, W Willinger, X Wang, BY Zhao, H Zheng, "Self-similarity in social network dynamics," ACM Trans. Model. Perform. Eval. Comput. Syst., vol. 2, October 2016.

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

[18] C Ji, R Pan, LP Chang, L Shi, Z Zhu, Y Liang, TW Kuo, CJ Xue, "Inspection and Characterization of App File Usage in Mobile Devices," ACM Transactions on Storage, vol. 16, no. 4, September 2020.

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

[20] 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.

[21] ZL Li, CJM Liang, W He, L Zhu, W Dai, J Jiang, G Sun, "Metis: Robustly Optimizing Tail Latencies of Cloud Systems," Proceedings of the 2018 USENIX Annual Technical Conference (ATC), Boston, MA, USA, 2018, pp. 981–992.

[22] J Zhang, A Sivasubramaniam, H Franke, N Gautam, Y Zhang, S Nagar, "Synthesizing representative I/O workloads for TPC-H," Proceedings of the 10th International Symposium on High Performance Computer Architecture (HPCA), Madrid, Spain, 2004.

[23] B Schroeder and GA 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 Storage Technologies (FAST'07), Berkeley, CA, USA, 2007, pp. 1–16.

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

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

[26] S Gribble, G Manku, and E Brewer, "Self-similarity in high-level file systems: Measurement and applications," Proceedings of the ACM SIGMETRICS'98, Madison, WI, 1998, pp. 141–150.

[27] Norros, "On the use of fractional Brownian motion in the theory of connectionless networks," IEEE Journal of Selected Areas in Communications, vol. 15, pp. 200–208, 1997.

[28] MW Garrett and W Willinger, "Analysis, modeling and generation of self-similar VBR video traffic," Proceedings of the ACM Conference on Special Interest Group on Data Communication (SIGCOMM), 1994.

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

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

[31] G Yeung, D Borowiec, R Yang, A Friday, R Harper, P Garraghan, "Horus: Interference-Aware and Prediction-Based Scheduling in Deep Learning Systems," IEEE Transactions on Parallel and Distributed Systems, vol. 33, pp. 88–100, 2022.