Contents lists available at ScienceDirect

Performance Evaluation

journal homepage: www.elsevier.com/locate/peva

Analyzing the distribution fit for storage workload and Internet traffic traces

Muhammad Wajahat[*], Aditya Yele, Tyler Estro, Anshul Gandhi, Erez Zadok

Department of Computer Science, Stony Brook University, NY, USA

A R T I C L E I N F O

Article history:
Available online 11 June 2020

Keywords:
Distribution fitting
Storage traces
Hyper-exponential
Performance modeling

A B S T R A C T

Understanding workloads and modeling their performance is important for optimizing systems and services. A useful first step towards understanding the characteristics of workloads is to analyze their inter-arrival times and service requirements. If these characteristics are found to follow certain probability distributions, then corresponding stochastic models can be employed to efficiently estimate the performance of workloads. Such approaches have been explored in specific domains using an assortment of distributions, including the Normal, Weibull, and Exponential. Our primary goal in this work is to understand and model storage workload performance. However, our analysis and others' past attempts revealed that none of the commonly-employed distributions provided a good fit for storage workloads. We analyzed over 250 traces across 5 different workload families using 20 widely used distributions, including ones seldom used for storage modeling. We found that the *Hyper-exponential* distribution with just two phases (H_2) was superior in modeling the storage traces compared to other distributions under five diverse metrics of accuracy, including metrics that assess the risk of over-fitting. Based on these results, we developed a Markov-chain-based stochastic model that accurately estimates the storage system performance across several workload traces. To assess the applicability of the Hyper-exponential for distribution fitting beyond storage traces, we evaluated distribution fitting for Internet traffic traces using over 1,600 traces from 3 different sources. We again found that the Hyper-exponential distribution provided a superior fit compared to other probability distributions. To highlight the applicability of our model, we conducted what-if analyses to investigate (i) the storage performance impact of workload variability and garbage collection under various scenarios and (ii) the impact on service response time of Internet flash crowds.

1. Introduction

Analyzing workload traces can provide useful insights into the characteristics of a system, helping to design better scheduling, caching, or service policies. Trace analysis can also help in the development of performance models that can enable useful what-if analysis, providing answers to questions such as "how will response time be affected if the arrival rate doubles?" or "how does workload variability impact performance?" Request-level traces, such as *inter-arrival times* (IATs) or *service times* (STs), are especially useful as they lend themselves to such performance modeling efforts and to the identification of system bottlenecks.

[*] Corresponding author.
E-mail addresses: mwajahat@cs.stonybrook.edu (M. Wajahat), ayele@cs.stonybrook.edu (A. Yele), testro@cs.stonybrook.edu (T. Estro), anshul@cs.stonybrook.edu (A. Gandhi), ezk@cs.stonybrook.edu (E. Zadok).

https://doi.org/10.1016/j.peva.2020.102121
0166-5316/© 2020 Elsevier B.V. All rights reserved.

A popular approach to analyzing request-level traces is to *infer the distribution* of events (e.g., distribution fitting), such as the distribution of IATs [1,2]. By fitting the empirical IATs to known distributions, such as the Normal distribution, one can leverage the various properties of the distribution to assess the traffic characteristics, such as burstiness and skew. Some of these distributions enable stochastic modeling of the performance of the system or device. For example, if the empirical IAT and/or ST traces can be shown to follow an Exponential distribution, then Markov Chain analysis or suitable queueing models can be developed to estimate the system performance [3–5]. The benefits offered by such distributions have encouraged many attempts to fit empirical data to these distributions or to simply assume that empirical data follows such distributions [6–8].

Unfortunately, storage workload characteristics are often too complex or skewed to be accurately modeled by simple Normal or Exponential distributions. Prior work has shown that storage workloads often exhibit long-tail latencies [9–13]; specifically, prior studies [2,14] have found, via parametric fitting, that storage traffic IATs and access patterns are well modeled by the heavy-tailed Generalized Pareto distribution. These observations also extend to other workloads; for example, the IATs of web requests, grid computing workloads, and supercomputing workloads were found to be well approximated by the 2-parameter Weibull distribution [15–17]. Such complex distributions often have atypical properties that make them infeasible for practical analysis. For instance, the Pareto distribution can have an infinite variance [18]. Likewise, the generalized extreme value (GEV) distribution [19] and the log-logistic distribution [20], both of which we found can accurately model storage IAT traces (see Section 5), can have infinite or undefined mean and/or variance. Thus, distributions that accurately model empirical request-level traces may not be helpful for analyzing storage workloads, say, for performance modeling, as we demonstrate in Section 6.

The goal of this article is to analyze various request-level traces across multiple workloads and find distributions that (i) provide high distribution fitting accuracy *and* (ii) provide practical analytical properties across all traces. To the best of our knowledge, such practical and large-scale distribution fitting study has not been carried out for request-level storage traces. While request-level traces for various historical and modern storage systems already exist in the public domain (e.g., SNIA's trace repository [21]), prior studies that analyze such traces either did not focus on distribution fitting [22–24] or did not leverage the distribution fit to enable performance modeling [1,2]; see Section 9 for a detailed discussion of related work.

The key takeaway of our analysis is that the flexible *Hyper-exponential* distribution is ideally suited for fitting and modeling request-level storage traffic. We further validate the superiority of the Hyper-exponential distribution by considering the fitting of Internet traffic traces; the performance of Internet services is known to be critically dependent on the variability in their IAT process [25–27]. The Hyper-exponential distribution is a probabilistic mixture of several exponential distributions, or phases. In general, the Hyper-exponential distribution can model most heavy-tailed distributions by selecting the appropriate parameters [28]. Importantly, since it is a mixture of Exponential distributions, it is amenable to stochastic analysis, including queueing-theoretic modeling. Further, because the Hyper-exponential is expressed in terms of simpler Exponential distributions, it has finite and well-defined (closed-form) mean and variance, which are easy to compute (see Section 3). The Hyper-exponential thus provides an opportunity to accurately model request-level traces while retaining the benefits offered by simpler distributions.

For our analysis, we use over 250 publicly available block-layer traces from five different workload families (see Section 4), and over 1600 Internet traffic traces from three different sources (see Section 7). We used three different metrics and associated techniques to assess the accuracy of the distribution fit: (i) R^2 [29], which indicates the goodness of fit, (ii) the Jensen–Shannon divergence [30], which measures the similarity between two distributions, and (iii) the likelihood [31], which describes the plausibility of observing the empirical data given the fitted distribution. We also employed the Akaike and Bayesian information criteria (AIC and BIC) [32,33], that assess over-fitting by evaluating the quality and simplicity of the fit.

We find that, compared to 19 widely used distributions (including Exponential, Generalized Pareto, Beta, Normal, Gamma, and Weibull), the Hyper-exponential distribution, with just two phases, provides better accuracy and lower risk of over-fitting across all traces we considered. For individual trace families, we found that Hyper-exponential is almost always among the top 3 distributions, and often the top distribution, for any metric of accuracy. While some distributions, like Burr and Pareto, do provide the best distribution fit in a few cases, their fit was poor in other cases (see Section 5).

To highlight the importance of distribution fitting for workload traces, we developed a stochastic model based on using the Hyper-exponential distribution fit for IATs and STs that can estimate the performance of the storage system. Our model relies on the fact that the Hyper-exponential is a mixture of Exponentials, and is thus amenable to Markov chain modeling. Our resulting model accurately predicts the mean response time for workload traces. The median response time modeling error for our Hyper-exponential–based model was 17.5%; by contrast, the median error for other distributions was at least $2.7\times$ larger than our error (see Section 6).

Finally, to illustrate the applications of our distribution fitting based performance model, we conducted three what-if analyses (Section 8). First, we investigated the impact on response time of an increase in workload traffic and/or increase in workload variability. We found that, at high arrival rates, doubling the workload variability can increase response time by as much as 66%. Second, we explored the performance degradation caused by garbage collection (GC), common in SSDs, as a function of various parameters, including the percentage of time spent in GC and its service rate slowdown. We found that GC can degrade average performance by as much as $2.8\times$ even if it runs only 1% of the time. Third, we investigated the impact of flash crowds on the response time of Internet services. We found that, even if the flash crowd persists for only one minute, the mean response time within a one hour window can increase by as much as $100\times$. Without our performance model, the above analyses would require extensive experimentation, and might even be infeasible.

2. Background and prior work

To motivate the contributions of this article and provide some context for our work, we next provide a brief overview of distribution fitting and then discuss related prior works that specifically focus on distribution fitting for storage traces. We discuss other related works later in Section 9.

2.1. Significance of distribution fitting

Distribution fitting is the process of selecting a statistical distribution that best fits the target empirical data set. Distribution fitting is a popular tool for analyzing empirical data, with books and journals dedicated to the topic [18,34]. We are specifically interested in Parametric fitting (or inference), where the empirical data is fit to a distribution with a known structure, but variable parameter values [35].

The key advantage of distribution fitting is that the many properties of the fitted distribution can now be directly applied to study the empirical data and possibly make predictions of future events. Further, appropriate statistical tests or hypothesis testing can be used to analyze the characteristics of the data. For example, if the service time (ST) of a storage workload is shown to follow a Pareto distribution, then the many moments of the distribution, as well as the tail probability (probability that a request takes longer than x seconds to complete), can be obtained in closed-form without any significant computational effort [18]. Likewise, if the inter-arrival time (IAT) of requests is shown to follow a Normal distribution, confidence intervals can be easily obtained for various measures of the data [5].

A more subtle but practical advantage of distribution fitting is the *performance models* that it enables. For example, if the request-level characteristics of a storage workload are shown to follow an Exponential distribution, its Markovian property can be used to track the evolution of the number of requests in the storage system as a continuous time Markov chain [4]. Likewise, based on the fitted distribution, various queueing-theoretic results can be applied to estimate the performance (e.g., response time) of the storage system.

Of course, the above advantages can only be realized if an *accurate* enough distribution fit is found. There are several techniques that exist in the literature for distribution fitting [18,34]. Typically, there is an associated metric of accuracy that each technique aims to optimize for when deriving the parameters of a fitted distribution. Rather than using a single technique or metric, we employ the suggested practice [36,37] of using *multiple techniques and metrics* to evaluate the distribution fitting; this avoids bias in results as a distribution may exhibit high accuracy for only one metric. We discuss our techniques and metrics in Section 5.1.

2.2. Prior work on fitting storage traces

Prior work on distribution fitting for storage workloads is restricted to analyzing traces from a specific source. Gomez et al. [2] analyzed disk access patterns for HP-UX servers [38] and found that the spatial access pattern is well modeled by a Pareto distribution. The authors employed parametric fitting to find the Pareto parameters but did not evaluate the accuracy of the fit. Gracia-Tinedo et al. [14] analyzed network traffic for the UbuntuOne cloud storage service and found that the IATs of some of the operations have long tails and are thus not well approximated by the Exponential distribution. Instead, the authors visually inspected the IATs and used a Pareto distribution fit. Birke et al. [39] analyzed storage workloads in an enterprise cloud and found that the VM-level storage capacity is well approximated by an Exponential distribution.

In general, heavy-tailed distributions have been used to model storage workload characteristics. However, we note that the above works rarely employ (one or more) statistical tests for evaluating the accuracy of the distribution fit. Instead, these studies employ simple accuracy fit metrics, such as mean and percentiles of modeling error, to evaluate the distribution fit. Further, prior work has not explored the performance models enabled by the fitted distribution.

3. Hyper-exponential distribution and related performance models

We now describe the Hyper-exponential distribution, which we find to be an accurate fit for the storage traces we analyze in Section 5. We then discuss the performance models enabled by the Hyper-exponential distribution, and other distributions.

The k-phase Hyper-exponential distribution, denoted as H_k, is a probabilistic mixture of k Exponential distributions. The k-phase Hyper-exponential has $(2k-1)$ parameters, and can be expressed as:

$$H_k = \begin{cases} Exp(\lambda_1) & \text{with probability } p_1 \\ Exp(\lambda_2) & \text{with probability } p_2 \\ \quad\vdots & \\ Exp(\lambda_k) & \text{with probability } p_k, \end{cases} \tag{1}$$

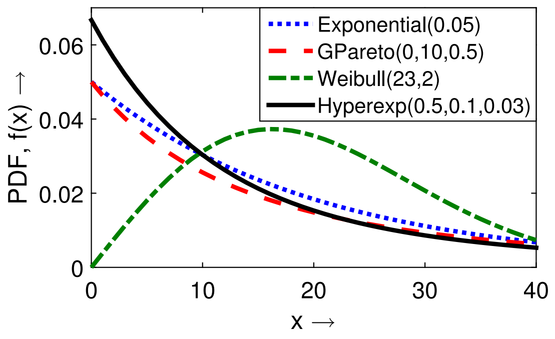


Fig. 1. Illustration of the probability distribution function (PDF) of various distributions, all with mean 20.

where $p_1 + p_2 + \cdots + p_k = 1$. Since the Hyper-exponential is simply a mixture of Exponentials, its moments are finite and can be easily expressed in closed form. For example, the mean (first moment) of a k-phase Hyper-exponential is $\sum_{i=1}^{k} p_i/\lambda_i$.

In its simplest form, a 2-phase Hyper-exponential, or H_2, is a mixture of two Exponential distributions, say $Exp(\lambda_1)$ with probability p and $Exp(\lambda_2)$ with probability $(1-p)$. Since the number of parameters to be estimated for the k-phase Hyper-exponential distribution scales linearly with k, it is beneficial to use a small value of k for efficient distribution fitting. In Section 5 we show that the H_2 with $k = 2$ is already powerful enough to model the inter-arrival times (IATs) and service times (STs) of storage traces.

The Hyper-exponential distribution has been used for modeling metrics in other communities, such as modeling the amount of rainfall in a region [40], modeling the completion time in manufacturing systems [41], modeling the reliability of software [42], and even the modeling of network traffic [43]. However, to the best of our knowledge, the Hyper-exponential is not a commonly employed distribution in the storage community and hence has not been applied to model storage workload characteristics.

3.1. *The need for a flexible and heavy-tailed distribution*

As discussed in Section 2.2, storage workload characteristics often exhibit heavy-tailed behavior [18]; this is further evidenced by prior work that focuses on the long-tail latencies of storage workloads [9–13]. The Exponential distribution has a single parameter and is not heavy-tailed. In fact, a heavy-tailed distribution is often defined as one whose tail probability is heavier than that of an Exponential [44]. The Pareto and Weibull are heavy-tailed distributions that are often employed for distribution fitting of empirical data that exhibits long tails. There are several other heavy-tailed distributions that exist, such as the Lognormal, Burr, Loglogistic, etc.; we evaluated distribution fitting with many of these in our trace analysis in Section 5. We find that, despite several statistical fitting techniques, the above distributions are *not* flexible enough to accurately fit the IATs and STs of storage workload traces obtained from different sources. That is, while a given heavy-tailed distribution fits a specific trace accurately, it does not fit other traces well. The 2-phase Hyper-exponential distribution, H_2, is flexible (3 parameters) and heavy-tailed. It has been shown that the k-phase Hyper-exponential can model most heavy-tailed distributions by selecting the appropriate parameters [28]. Like the Exponential distribution, the Hyper-exponential does have a decaying probability distribution function. We illustrate the probability density function (PDF) of the Hyper-exponential and other common distributions in Fig. 1; here, we show the PDF for a 2-phase Hyper-exponential, or H_2.

3.2. *Performance models enabled by the hyper-exponential*

Since the Hyper-exponential has more parameters than the Exponential, it is more flexible than the Exponential distribution. However, the Hyper-exponential retains many of the analytical advantages of the Exponential distribution. Specifically, the memoryless or Markovian property of the Exponential allows us to model the evolution of events as a continuous time Markov chain [4]. If the IAT and ST are modeled as Exponentials, then we can model the storage system using an M/M/1 Markov chain, as shown in Fig. 2. The Markov chain tracks the number of requests in the system as they dynamically increase due to arrivals and decrease due to service events. By solving for the steady-state probability of the different states of the chain, we can derive the mean number of requests in the system [5]; the mean number in system can then be converted to the mean response time via Little's Law [45]. Fortunately, the Hyper-exponential is also amenable to Markov chain analysis due to its mixture of Exponential nature. Specifically, each phase of a Hyper-exponential can be modeled as a state in the Markov chain. However, the resulting chain is quite complex. Consider a workload whose IAT and ST are modeled as 2-phase Hyper-exponentials:

$$IAT = \begin{cases} Exp(\lambda_1) \text{ w.p. p} \\ Exp(\lambda_2) \text{ w.p. (1-p)} \end{cases} \quad ST = \begin{cases} Exp(\mu_1) \text{ w.p. q} \\ Exp(\mu_2) \text{ w.p. (1-q)} \end{cases}$$

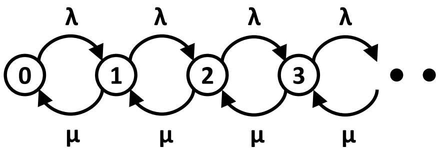


Fig. 2. Illustration of an M/M/1 Markov chain performance model with mean IAT $= 1/\lambda$ and mean ST $= 1/\mu$. The Markov chain tracks the number of requests in the storage system.

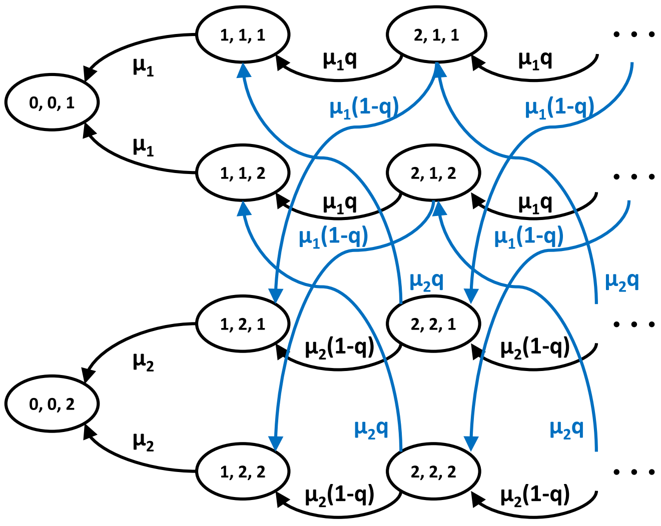


Fig. 3. Illustration of an $H_2/H_2/1$ Markov chain with the IAT modeled as an $H_2(p, \lambda_1, \lambda_2)$ and the ST modeled as an $H_2(q, \mu_1, \mu_2)$. We color-code some of the transitions and only show ST events for simplicity. The state space (i, j, k) refers to the number of requests in system, phase of the ST, and phase of the IAT, respectively.

Fig. 3 shows the Markov chain for the $H_2/H_2/1$ system with the above IAT and ST distributions. Such chains have a repeating structure and can be solved, either numerically (e.g., using matrix analytic methods [46]) or analytically. Thus, the Hyper-exponential is a flexible and heavy-tailed distribution that also allows for exact performance models.

For other distributions, such as Pareto or Weibull, exact performance models are not known. However, approximations are available [47]; we leverage these models and approximations in Section 6 to model the performance of a storage system with different distribution fits.

4. Description of traces and workloads

For the distribution fitting analysis of storage traces, we consider more than 250 different block-level traces from 5 different sources, including those from Flash-based devices and hard disks. We also consider more than 1600 Internet traffic traces from 3 different sources, including web access traces and network packet traces. We focus on the following request-level information in the traces, when available:

- *Inter-arrival time (IAT):* The IAT is defined as the time between successive requests. When analyzing IAT, we distinguish between reads and writes to better understand their individual characteristics.
- *Service time (ST):* The ST is defined as the time taken by the request for processing at the device, and does not include the queueing/waiting time at the device or at the upper layers. ST is often difficult to obtain as there is some queueing that happens within the device which cannot be easily tracked due to vendor-specific (proprietary) firmware [48].
- *Response time (RT):* The RT is the performance metric defined as the time taken by the request to complete service from when it first arrives at the block layer.

4.1. Florida international university traces

These are an assortment of 3–4 week-long block traces obtained from various HDD-based production systems in the Department of Computer Science at Florida International University (FIU) by Verma et al. [49]. The *home 1-4* workloads are 4 separate traces of the home directories of 4 different users in FIU's research group. The *mail* workload served the department's e-mail inboxes. The *online* workload is a web server hosting the department's course management system.

The *webmail* workload is a web interface to the department's mail server. The *webusers* workload served the department members' websites. Lastly, the *webresearch* workload is an Apache server managing around 10 research projects. We analyzed the read and write IATs separately for each trace to better understand access type specific traffic, resulting in 18 total traces. ST information is not available for these traces.

4.2. Virtual desktop infrastructure (VDI) traces

These are a collection of storage traffic traces from an enterprise virtual desktop infrastructure (VDI), obtained from Lee et al. [24]. The month-long traces contain I/O information for six different block storage devices (LUNs), with each device corresponding to one VDI server, which itself hosts about 50 VMs. We analyzed the read and write IATs separately for each of the 6 devices, resulting in 12 traces. ST information is not provided for these traces.

4.3. Mobile storage subsystem traces

These 31 application-specific block-level I/O traces were collected on a Nexus 5 smartphone when running different mobile applications, as detailed in Zhou et al. [23]. The storage subsystem is a Flash-based (SanDisk iNAND) eMMC. I/O information is collected at the block layer and the eMMC driver layer, so we have both IAT and ST traces, and response times (RT). We analyzed the read and writes IATs and STs separately, resulting in 62 traces each. We leveraged the response times to validate our performance models in Section 6.

4.4. Microsoft production server storage traces

These historical traces were collected on real production storage servers of Microsoft services, as described in Kavalanekar et al. [22]. We use the block-level IAT information for the storage metadata servers, separated by access type (reads and writes), resulting in 72 traces. ST information is not provided for these traces.

4.5. Cloudphysics traces

These week-long block I/O traces were collected by CloudPhysics [50] from production VMware virtual disks in customer data centers running the VMware ESXi hypervisor [51]. A user-mode application, deployed on each ESXi host, coordinated with the standard VMware vscsiStats utility [52] to collect complete block I/O traces for the virtual disks. Although CloudPhysics did not record customers' specific application usage, they estimated that approximately 10%–15% were running database applications. We obtain arrival time information from 42 traces and use them for IAT fitting; the traces did not have ST information.

4.6. Wikipedia access traces

These traces consist of sampled user requests to Wikipedia in all languages, as detailed in Guido et al. [53] and obtained from WikiBench [54]. These traces contain request arrival times, allowing us to obtain IAT information. We choose the most recently available subset of traces (for October 2007) in our analysis. This subset spans 21 days, and is divided into 500 subtraces of one hour each. We filter the requests to include only topic page views and exclude edit requests and image load requests that are indirectly generated from the original requests.

4.7. Stack exchange: Stackoverflow traces

These traces are published by Stack Exchange for all user-contributed content across different websites in their network, available at the Internet Archive [55]. We use *Posts* data from *Stackoverflow* site, the biggest site in terms of daily traffic. This dataset contains questions and answers posted by site users; we use all available data for the last decade (January 2010 to November 2019), resulting in 119 monthly subtraces. We consider only question type posts, and obtain IAT information for each subtrace.

4.8. Waikato internet traffic storage (WITS) traces

These publicly available network traces are published by the WAND network research group [56]; they contain packet level information captured at the University or ISP level. We use the ISPDSL-II [57] trace family, which contains packet header information captured from a New Zealand ISP. We focus on the one billion most recent packets, resulting in 1077 subtraces, each containing roughly one minute's worth of packets. These traces contain only IAT information of the network packets.

Table 1
List of distributions evaluated for fitting, organized by the number of parameters in the distribution.

1 parameter	2 parameter	3 parameter
Exponential	Birnbaum–Saunders	Burr
Rayleigh	Beta	Generalized Extreme Value
Rician	Extreme Value	Generalized Pareto
	Gamma	Hyper-exponential (H_2)
	Half Normal	t Location-Scale
	Inverse Gaussian	
	Logistic	
	Loglogistic	
	Lognormal	
	Nakagami	
	Normal	
	Weibull	

5. Request-level distribution fitting for storage traces

We now present our first contribution, analyzing the distribution fit for storage workload traces. We start with a description of the distribution fitting methods we employ and then present our results for IAT analysis and ST analysis. Later, in Section 7, we extend our distribution fitting analysis to Internet traffic traces.

Note that the focus of our study is analyzing the distribution fitting of the traces, and not the analysis of the traces themselves. The traces we analyze for distribution fitting have been studied before [22–24,58], in other contexts, such as for deduplication and energy management.

5.1. Methods and metrics for distribution fitting

We consider several widely used distributions for our distribution fitting analysis. The full list of distributions we consider is listed categorically by the number of parameters in Table 1. To find the parameters of a given distribution that result in the best fit to the empirical trace, we use three different techniques. Each of these techniques has its own designated metric of accuracy. We also evaluate the risk of over-fitting by reporting metrics that estimate the model quality. Using multiple techniques and metrics avoids any bias that a fitted distribution may have to a single metric [36,37]. Our code for distribution fitting using the following techniques is publicly available at GitHub [59].

5.1.1. Least squares optimization to maximize R^2

The coefficient of determination, R^2 [29], indicates the goodness of fit—that is, the closeness of the empirical data to the fitted distribution. It is often used as the first step in evaluating a fit. R^2 typically lies between 0 and 1, with higher values indicating a better fit, though negative values are possible when the fit is poor. We use the *least squares* approach to minimize the sum of the squares of the residuals between the empirical Cumulative Distribution Function (CDF) and the CDF of the target distribution. Our global optimization heuristically chooses initialization points and then applies the interior-point method to find the best parameter values [60,61]. At a high-level, the optimization tests the value of the objective function in the neighborhood of the initialization points, and moves in the direction that best improves the objective value [62]; eventually, the algorithm converges to the parameter values that result in the best value of the objective function [63].

It has been shown in prior statistical studies that the R^2 metric alone may be insufficient to evaluate the fit [64]; we thus consider additional metrics as well.

5.1.2. Global search algorithm to minimize divergence

JSD, the Jensen–Shannon divergence [30], is a symmetric and smoothed version of the Kullback–Leibler divergence, and measures the similarity between two probability distributions. JSD is popular in information theory and coding theory to measure the relative entropy, or distance, between two distributions [65]. JSD typically lies in the (0, 1) range, with lower values indicating higher accuracy. We use the global search algorithm [60] to find parameter values that minimize the JSD between the empirical PDF and the PDF of the target distribution. The algorithm is similar to the one described above for maximizing R^2, and uses a similar framework for initialization and convergence.

5.1.3. Expectation–maximization to maximize likelihood

The likelihood objective function [31], the higher the better, is often used in Bayesian statistics to describe the probability of observing the empirical data given the target (fitted) distribution [66]. We use the popular expectation–maximization (EM) algorithm [31] to find the distribution parameters that maximize the expected log likelihood. EM is an iterative algorithm; we follow the suggested practice of using normally distributed values, with the same mean and variance as that of the empirical data, to generate our initial guesses [67,68]. The best performing initialization is then chosen.

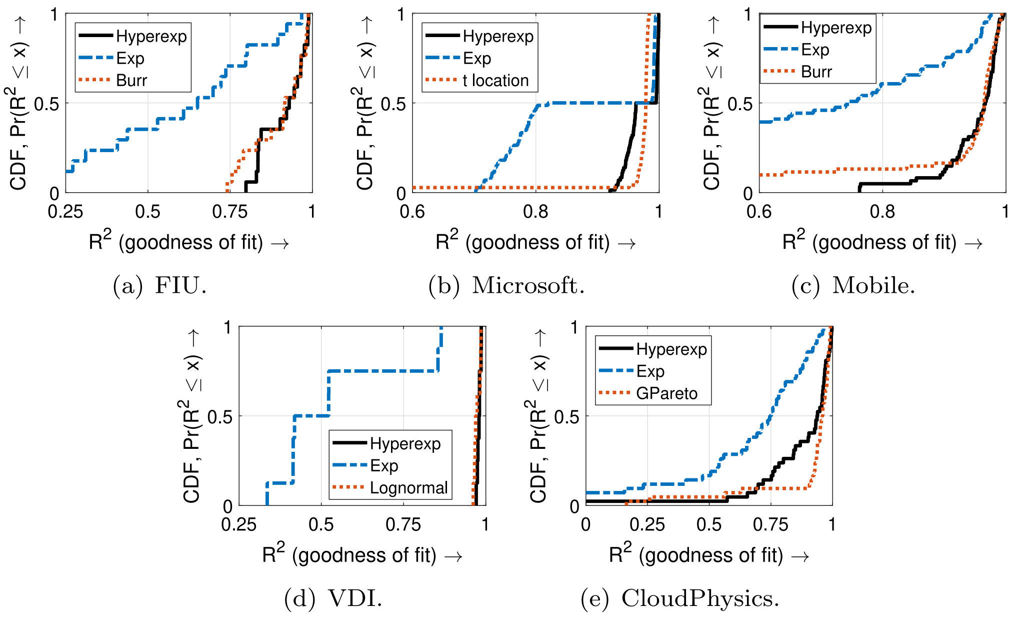


Fig. 4. CDF of the R^2 metric (higher is better) for Hyper-exponential, Exponential, and the best alternative distribution fit for all five storage workload traces.

5.1.4. *Akaike information criterion*

AIC [32], the lower the better, is an estimator of the relative quality of statistical models for a given dataset, and is often used for model selection. AIC estimates the amount of information lost by the model, and deals with the trade-off between *goodness of fit* and *simplicity* of the model by penalizing log likelihood proportional to the number of model parameters. AIC is reported for the distribution fit obtained via the EM algorithm that maximizes likelihood.

5.1.5. *Bayesian information criterion*

BIC [33], the lower the better, is similar to AIC but imposes a larger penalty for the number of parameters. BIC can select the *true model* with probability close to 1 when the number of data points is high. As both AIC and BIC deal with the trade-off between goodness of fit and simplicity of the model, they allow us to assess both over-fitting and under-fitting of distributions to data, and help select the best model.

5.2. *Inter-arrival time (IAT) trace analysis*

All five storage trace families that we studied – FIU, Microsoft, Mobile, VDI, and CloudPhysics (see Sections 4.1–4.5) – have IAT information. Figs. 4, 5, and 6, respectively, show the R^2, JSD, and log-likelihood metrics for the different trace families under three distribution fits: (i) Hyper-exponential with two phases (H_2), (ii) Exponential, and (iii) the best alternative distribution (apart from Hyper-exponential and Exponential). We include the Exponential as a baseline as it enables useful performance modeling of systems and has often been used as the default IAT distribution in performance studies [6–8].

We see that the Hyper-exponential is always at least as good as the best alternative distribution; the median R^2 for Hyper-exponential ranges from 0.93–0.98 for all trace families. By contrast, the Exponential is typically inaccurate, with the median R^2 for Exponential ranging from 0.4–0.8 for the different trace families. Note the stark contrast around the median in the Cumulative Distribution Function (CDF) plots for the Microsoft trace; this is because of the difference in behavior of reads and writes. We separately analyzed reads and writes and found that the fit accuracy for all distributions was better for reads than for writes, indicating bursty IAT behavior of write requests; this is expected as most operating systems batch writes in their page cache and flush them periodically in groups.

Finally, note that the best alternative distribution often changes based on the trace family. Similarly, we observed that the best alternative distribution *for a given trace family* changed with the metric of accuracy. We note that the likelihood value depends on the number of data points, and so likelihood values should not be compared across trace families.

To assess the risk of over-fitting, we now present the AIC and BIC values for the various distribution fits. Figs. 7 and 8 show the AIC and BIC metrics, respectively, for the different trace families; we show results for the H_2, Exponential, and the best alternative distribution. We see that H_2 provides a superior fit, and typically has the lowest median AIC and BIC (lower is better). Note that the AIC and BIC results look similar, as they are both based on the log-likelihood metric (see Sections 5.1.4 and 5.1.5), with slightly different penalty functions for the number of model parameters.

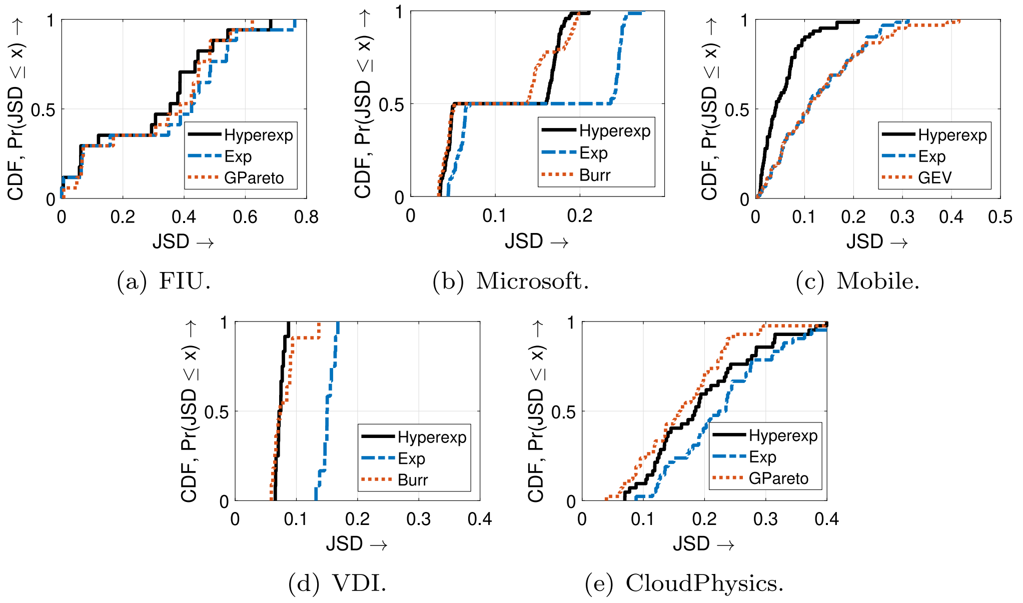


Fig. 5. CDF of the JSD metric (lower is better) for Hyper-exponential, Exponential, and the best alternative distribution fit for all five storage workload traces.

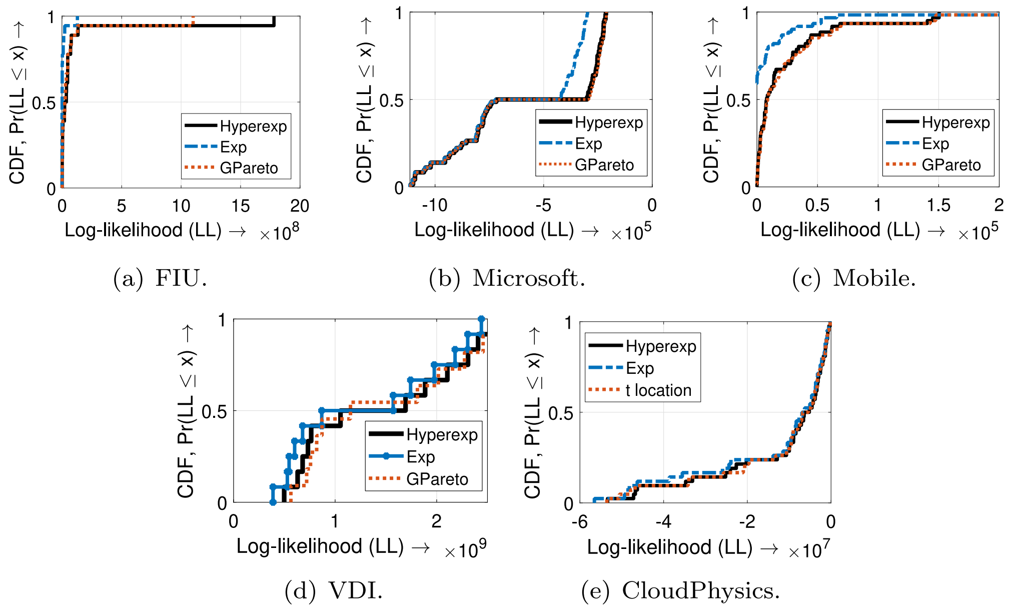
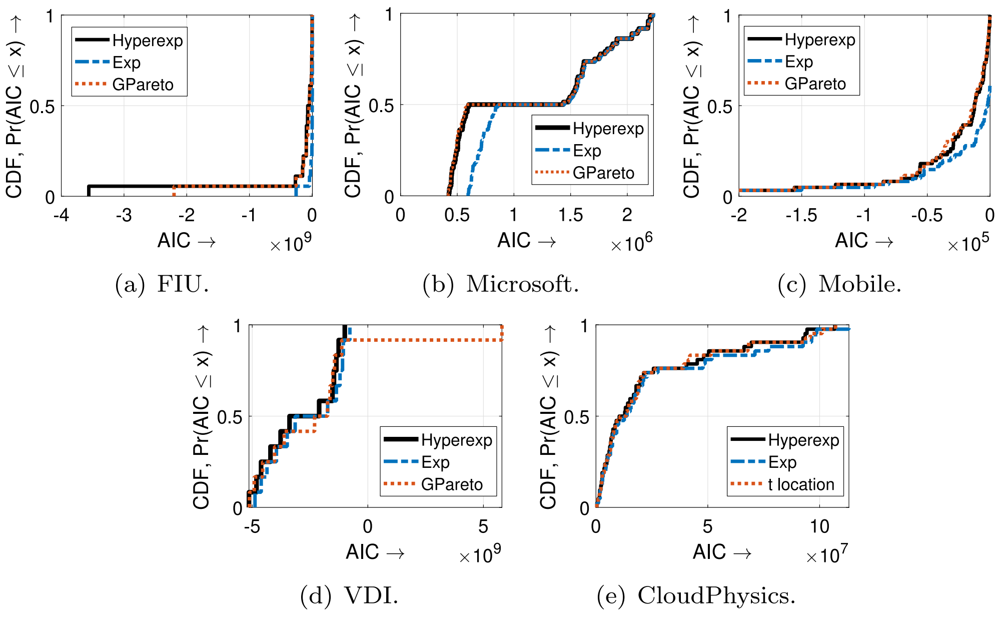


Fig. 6. CDF of Log-likelihood (LL, higher is better) for Hyper-exponential, Exponential, and the best alternative distribution fit for all five storage workload traces.

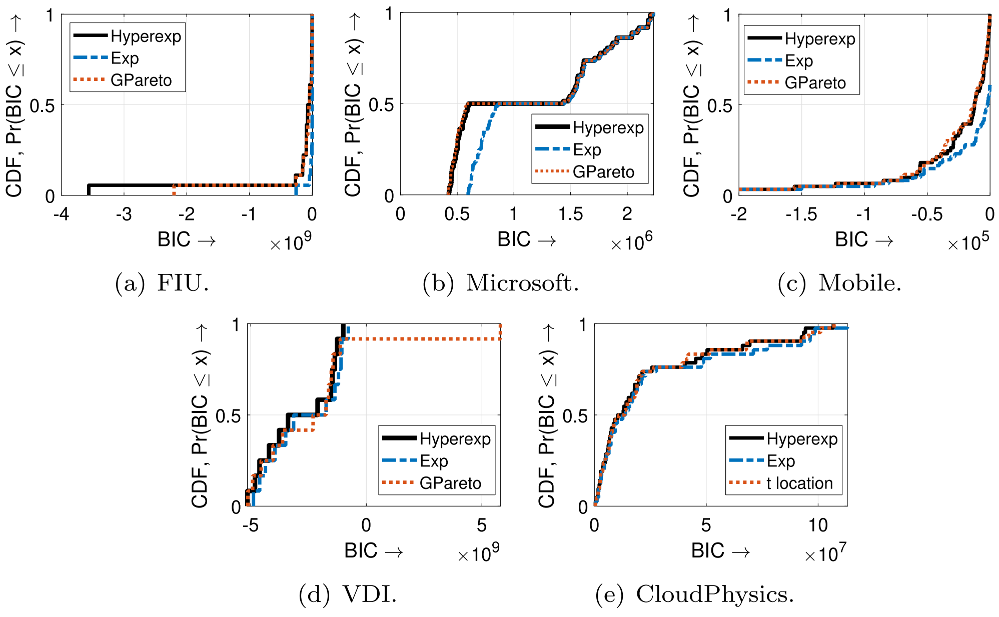
To better assess the fitting capabilities of the Hyper-exponential, we show the top 3 distribution fits, using the median accuracy, across *each trace family* and for *all* traces in Tables 2 and 4, respectively. We also show the top 3 distribution fits, using the median AIC and BIC values, across *each trace family* and for *all traces* in Tables 3 and 5, respectively. We see that the Hyper-exponential almost always ranks in the top 3 for any trace family under all 5 accuracy metrics. The only exception is the R^2 and JSD metrics under the CloudPhysics traces, for which the Hyper-exponential ranks as the fifth best distribution fit. For these traces, the median R^2 and JSD values for Hyper-exponential are 0.938 and 0.184, respectively, whereas the corresponding median values for the top-ranked distribution are 0.959 (2.2% better) and 0.159 (13.6% better), respectively. One possible reason for the slightly lower R^2 and JSD values under the CloudPhysics traces for the Hyper-exponential is that these traces contain virtual machine level IAT information; VMs are naturally more prone to performance fluctuations due to their virtual nature, and this hurts the distribution fit accuracy [69,70].



(a) FIU. (b) Microsoft. (c) Mobile.

(d) VDI. (e) CloudPhysics.

Fig. 7. CDF of Akaike information criterion (AIC, lower is better) for Hyper-exponential, Exponential, and the best alternative distribution fit for all five storage workload traces.



(a) FIU. (b) Microsoft. (c) Mobile.

(d) VDI. (e) CloudPhysics.

Fig. 8. CDF of Bayesian information criterion (BIC, lower is better) for Hyper-exponential, Exponential, and the best alternative distribution fit for all five storage workload traces.

Across all storage traces, the Hyper-exponential resulted in the best fit under JSD and log-likelihood (Table 4), with the median being 8.7% and 0.8% more accurate, respectively, than the top alternative distribution. For R^2, the Hyper-exponential is a close second, next only to t-location, and only by 1.4%. Finally, the Hyper-exponential also provides the highest quality fit, with the lowest median AIC and BIC values across all traces we used (Table 5). This shows that the Hyper-exponential's superior distribution fit is not a result of over-fitting.

In summary, the Hyper-exponential consistently provides superior distribution fit under diverse accuracy metrics for all trace families we consider.

Table 2
Ranking of the top 3 fitted distributions (top-to-bottom) for *each* storage workload trace family. Here, LogN, LogL, BSaunders, GPareto, and GEV refer to Lognormal, Loglogistic, Birnbaum–Saunders, Generalized Pareto, and Generalized Extreme Value, respectively.

FIU			Microsoft			Mobile		
R^2	JSD	LL	R^2	JSD	LL	R^2	JSD	LL
t-location	H_2	H_2	H_2	Burr	GPareto	H_2	H_2	H_2
H_2	GPareto	GPareto	t-location	LogL	GEV	Burr	GEV	GPareto
Burr	Gamma	t-location	LogN	H_2	H_2	LogN	Exp	GEV

VDI			CloudPhysics		
R^2	JSD	LL	R^2	JSD	LL
H_2	H_2	H_2	GPareto	GPareto	H_2
LogN	Burr	Exp	GEV	LogN	GPareto
Burr	GPareto	GPareto	t-location	BSaunders	LogL

Table 3
Ranking of the top 3 fitted distributions (top-to-bottom), according to AIC and BIC, for *each* storage workload trace family. Here, LogL, GPareto and GEV refer to Log Logistic, Generalized Pareto, and Generalized Extreme Value, respectively.

FIU		Microsoft		Mobile		VDI		CloudPhysics	
AIC	BIC	AIC	BIC	AIC	BIC	AIC	BIC	AIC	BIC
H_2	H_2	GPareto	GPareto	H_2	H_2	H_2	H_2	H_2	H_2
GPareto	GPareto	GEV	GEV	GPareto	GPareto	Exp	Exp	GPareto	GPareto
t-location	t-location	H_2	H_2	GEV	GEV	GPareto	GPareto	LogL	LogL

Table 4
Median R^2, JSD, and log-likelihood across *all* storage workload traces for the top 3 distributions in each case, sorted by accuracy.

(a) R^2 (higher is better)

Distribution	R^2
t-location	0.967
Hyper-exponential	0.953
Burr	0.951

(b) JSD (lower is better)

Distribution	JSD
Hyper-exponential	0.126
Birnbaum–Saunders	0.138
Generalized Pareto	0.144

(c) Log-likelihood, LL (higher is better)

Distribution	LL
Hyper-exponential	−740558
Generalized Pareto	−746798
Generalized Extreme Value	−784925

Table 5
Median AIC and BIC values across *all* storage workload traces for the top 3 distributions in each case, sorted by accuracy.

(a) AIC (lower is better)

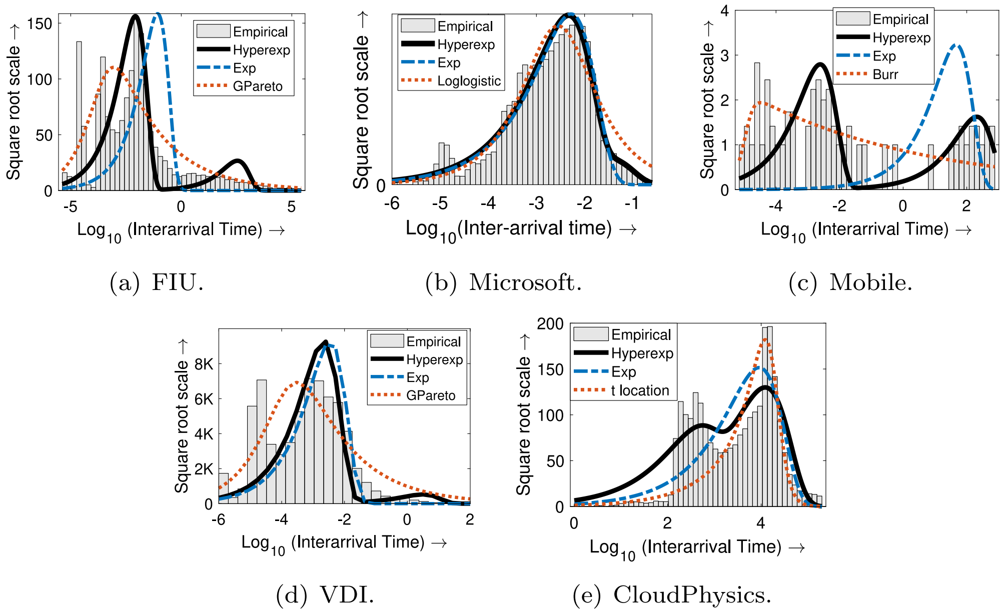
Distribution	AIC
Hyper-exponential	−3816
Generalized Pareto	−3226
t-location	−2299

(b) BIC (lower is better)

Distribution	BIC
Hyper-exponential	−3799
Generalized Pareto	−3213
t-location	−2283

5.3. *Analyzing the distribution fit*

Fig. 9 shows examples of distribution fits from each family of storage workload traces. The x-axis is on a log scale, and the y-axis uses square root scale, a measure that preserves the relative y-axis values per x-axis point and allows us to visually compare tails of empirical data [71]. We show the histogram for the empirical trace data and overlay it with



(a) FIU. (b) Microsoft. (c) Mobile. (d) VDI. (e) CloudPhysics.

Fig. 9. Results of distribution fit for a sample trace from each storage workload trace family for Hyper-exponential, Exponential, and the best alternative distribution (per log-likelihood).

the probability density function (PDF) of the Hyper-exponential (H_2), Exponential, and the top alternative distribution for that trace.

Fig. 9(a) represents the case where the H_2 captures the high PDF region (around 10^{-2}) well whereas the other distributions, including the best alternative distribution for this trace, Generalized Pareto, fail to accurately fit around this region. Fig. 9(b) represents the case where all distributions perform similarly, but there is a difference at the tail distribution (right of the graph). On close inspection, we see that the Exponential under-fits and the Loglogistic over-fits the tail probability; by contrast, the H_2 accurately fits the tail. Fig. 9(c) represents a worst-case fitting example where no distribution performs well. However, we clearly see that the H_2 has two distinct centers of high PDF (around 10^{-3} and 10^{2}) that provide good coverage of the empirical data. By contrast, the other distributions concentrate around a single IAT range; Fig. 9(e) shows a similar example for a CloudPhysics trace. Finally, Fig. 9(d) shows another non-trivial example where the H_2 is able to fit the center of the PDF (around 10^{-3}) as well as the tail (around 10^{0}), whereas the other distributions only fit the center.

These examples illustrate the flexible and heavy-tailed nature of H_2, which is important for accurately fitting the different types of storage traces, as we alluded to in Section 3.1.

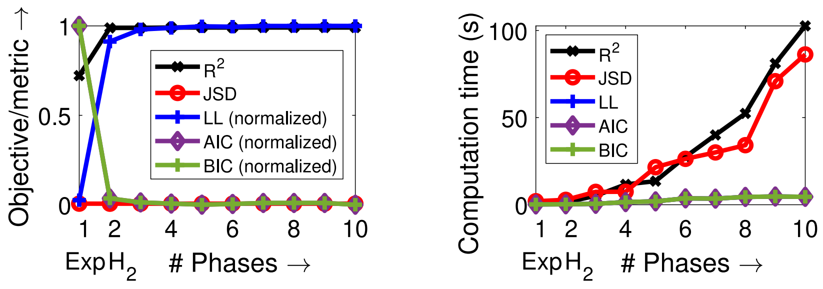
5.4. Sensitivity analysis for number of phases of the hyper-exponential

While we only make use of the 2-phase Hyper-exponential in the above analysis, the Hyper-exponential can be extended to include more phases (more Exponentials within the mixture distribution), though at the expense of increased computational complexity. Fig. 10(a) shows the accuracy for all five metrics as a function of the number of phases, k, of the k-phase Hyper-exponential. These results are for the *home3* subtrace from the FIU trace family; results are similar for other traces. Note that $k = 1$ refers to the Exponential distribution. We see that accuracy increases as we go from the Exponential to the 2-phase Hyper-exponential, but then largely stabilizes beyond $k = 2$; note that for AIC and BIC, lower values are better. For JSD, whose range of values is small and thus not distinguishable in the figure, the value drops from 0.0061 for $k = 1$ to 0.0058 for $k = 2$ (smaller JSD is better), and then largely remains unchanged.

Fig. 10(b) shows the time taken for the distribution fitting for different k values. We see that the computation time scales with the number of phases, as expected. Note that the LL, AIC, and BIC data points overlap as the same (EM) algorithm is employed for their fits (see Section 5.1). We did not specifically optimize the code for computation time as that is not the focus of this work; however, we used the same code for all phases to enable a fair comparison. In summary, $k = 2$ provides a good trade-off between high accuracy and low computation time, thus representing a good choice for the Hyper-exponential distribution fitting.

5.5. Service time (ST) trace analysis

We performed a similar distribution fitting for service times (ST). Only one of the trace families we studied, mobile storage traces, had ST information. Our ST analysis results are similar to IAT analysis, so we briefly highlight the results.



(a) Improvement in accuracy. (b) Increase in computation time.

Fig. 10. Impact of number of phases on (a) improvement in accuracy, and (b) computation time, for different techniques/objectives. Note that *phase* = 1 refers to Exponential and *phase* = 2 refers to 2-phase Hyper-exponential (H_2).

We again find that H_2 consistently provides a superior fit for service time under all accuracy metrics. We also find that the top alternative distribution changes with the accuracy metric. For R^2, JSD, and log-likelihood, the best alternative distribution was Lognormal, Birnbaum–Saunders, and Burr, respectively. The Exponential and Generalized Pareto distributions did not provide a good fit for ST.

6. Performance modeling evaluation

We now present our performance modeling study that demonstrates the applicability of the Hyper-exponential distribution fit to predict the mean response time for the modeled storage workload. We first describe the performance models we use, and then present our modeling results.

6.1. Methodology

As discussed in Section 3.2, both the Exponential and Hyper-exponential distributions enable Markov chain models. These, in turn, can be solved analytically or numerically to find the mean response time; for other distributions, only approximate results are available.

6.1.1. Hyper-exponential–based model ($H_2/H_2/1$)

When the IAT and ST are distributed as 2-phase Hyper-exponentials, the resulting queueing model is referred to as a $H_2/H_2/1$ queue [5]. For this queue, closed-form analytical expressions for mean response time can be obtained [72]. The analysis involves tracking the queue length in the Markov chain (see Fig. 3) given the input (IAT) and output (ST) processes, resulting in a degree 3 polynomial that can be solved to derive the mean queueing time, $E[W]$; here, $E[X]$ denotes the expectation or mean of the random variable X. Adding the mean ST to the mean queueing time gives the mean response time of the system, $[T] = E[W] + E[ST]$. Using the above approach, the mean response time for the $H_2/H_2/1$ model can be obtained in less than one millisecond with negligible CPU and memory overhead.

6.1.2. Exponential-based model ($M/M/1$)

When the IAT and ST are distributed as Exponentials, we can model the resulting $M/M/1$ system [4] as a simple Markov chain (see Fig. 2) that can be easily solved to obtain the mean response time as $E[T] = 1/(E[ST]^{-1} - E[IAT]^{-1})$.

6.1.3. Models for other distributions ($G/G/1$)

For general IAT and ST distributions, the queueing model is referred to as the $G/G/1$ queue [4]. For $G/G/1$, exact results are not known, and Markov chain modeling is not applicable for distributions other than the Exponential and Hyper-exponential. However, the Kingman's approximation [47] is widely used to estimate the mean waiting time of $G/G/1$ as

$$E[T] \approx E[ST] + \frac{E[ST]^2}{2(E[IAT] - E[ST])} \cdot \left(\frac{Var[IAT]}{E[IAT]^2} + \frac{Var[ST]}{E[ST]^2} \right)$$

where $Var[X]$ denotes the variance of the random variable X. Note that given the parameters of the IAT and ST distribution fit, their mean and variance can be easily computed.

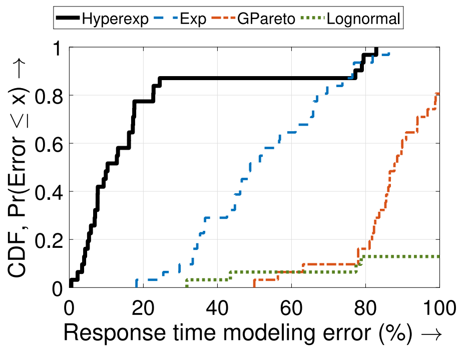


Fig. 11. CDF of mean response time modeling error using the Hyper-exponential and Exponential distribution fits, along with the other top alternative distribution fits for the mobile storage traces.

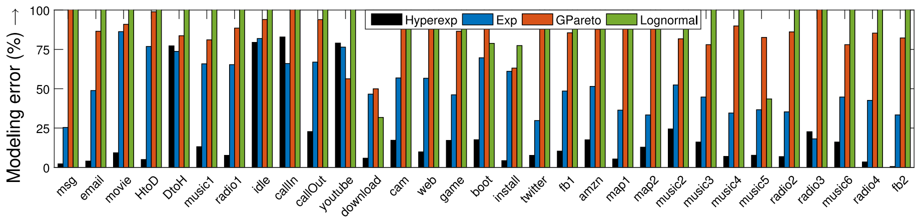


Fig. 12. Response time modeling error for the Hyper-exponential, Exponential, and other top alternative distribution fits for all mobile storage traces.

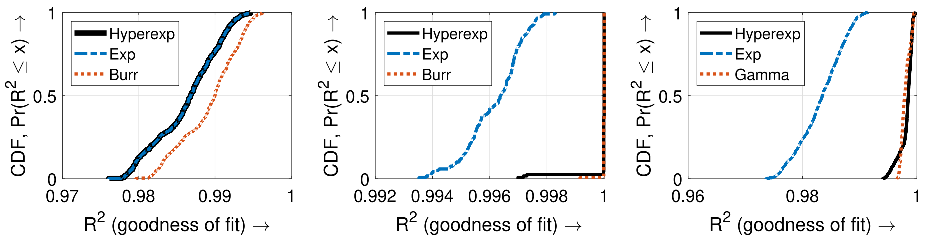
6.2. Response time modeling results

We consider the mobile storage subsystem traces (see Section 4.3), which contain IAT, ST, and response time (RT) information for 31 traces. Note that the other trace families we consider in this article do not contain service time (ST) or response time (RT) information, so we do not evaluate the performance model on those traces. Zhou et al. [23] reported that the observed response time for their mobile traces is typically $2\times$ the service time, suggesting that there is significant delay in the system. This motivated our response time modeling efforts for these traces. Since the scheduler used for the Flash-based mobile storage subsystem does not maintain different service queues for reads and writes [23], we model the IATs and STs for both request access types together.

Fig. 11 shows the CDF of the mean response time modeling error for the 31 mobile storage traces. We show modeling error results for the case of Hyper-exponential and Exponential based distribution fits of IATs and STs, using the $H_2/H_2/1$ and $M/M/1$ queueing models, respectively. Additionally, we show results for the two other top alternative distribution fits (ordered by median accuracy), whose response time is modeled by the $G/G/1$ approximation. We note that while the General Extreme Value, t-location, Burr, and the Loglogistic distribution also resulted in high median accuracy for the IAT and ST distribution fits, their fitted parameters resulted in infinite mean and/or variance. Clearly, this would result in a poor approximation and so we omit these distributions.

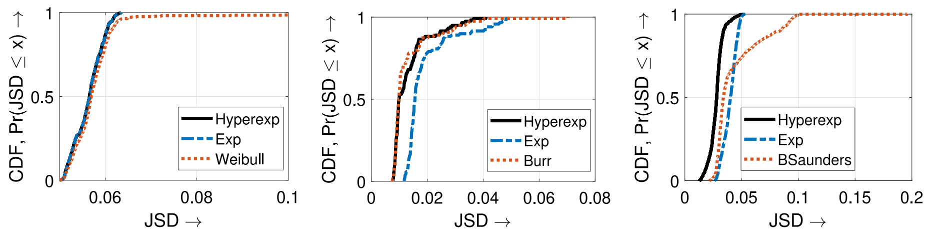
We see that the Hyper-exponential-based modeling error for mean response time is significantly lower than the other distributions in Fig. 11. The median error for the Hyper-exponential, Exponential, Generalized Pareto, and Lognormal based response time modeling is 17.5%, 48.8%, 87.8%, and 361.2%, respectively. The corresponding mean error numbers are 19.8%, 52.1%, 96.9%, and 672.2%, respectively; the mean error is higher than the median error due to the much higher error values for a few traces. The high error numbers for the Generalized Pareto and Lognormal based modeling should be expected since the response time model is only an approximation for these cases. Note that an error $>$ 100% indicates that the predicted response time is at least twice the actual response time. Across all traces, the $H_2/H_2/1$ model reduces the relative modeling error by about 64% when compared to the $M/M/1$ model.

Fig. 12 shows the per-trace response time modeling errors for all 31 mobile storage traces. In most cases, the Hyper-exponential–based $H_2/H_2/1$ model results in low error; the modeling error is less than 20% for 24 of the 31 traces and less than 25% for 27 of the 31 traces. We inspected the remaining 4 traces (DtoH, idle, callIn, and youtube) and found that for most of these, the number of entries in the trace was small. It is likely that the smaller sample size in these traces resulted in poor accuracy for our modeling approach. It should be noted that the modeling error for these 4 traces using the other distributions in Fig. 12 is also high. Although our approach does not have a minimum sample size requirement for the trace, in general, the more data points we have, the better is our modeling accuracy. In our evaluation, a minimum trace length of 3000 provided good modeling accuracy. Of the four traces that had high modeling error, two of them, namely callIn and youtube, had fewer than 3000 data points; specifically, they had 1490 and 2079 data points, respectively.



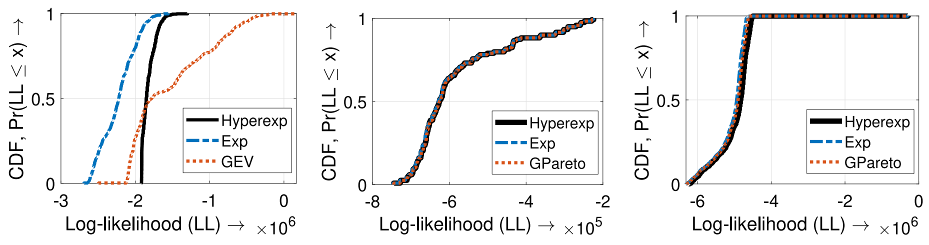
(a) Wikipedia. (b) Stackoverflow. (c) WITS.

Fig. 13. CDF of the R^2 metric (higher is better) for Hyper-exponential, Exponential, and the best alternative distribution fit for all three Internet traffic traces.



(a) Wikipedia. (b) Stackoverflow. (c) WITS.

Fig. 14. CDF of the JSD metric (lower is better) for Hyper-exponential, Exponential, and the best alternative distribution fit for all three Internet traffic traces.



(a) Wikipedia. (b) Stackoverflow. (c) WITS.

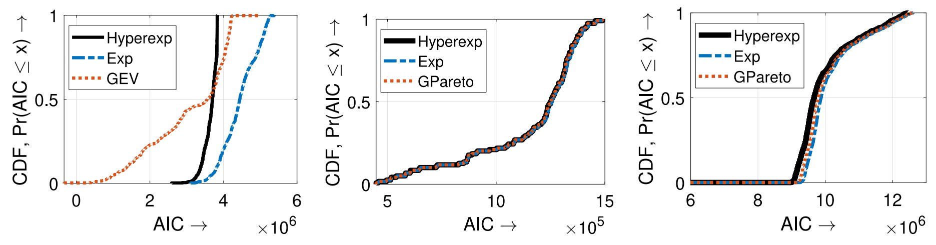
Fig. 15. CDF of Log-likelihood (LL, higher is better) for Hyper-exponential, Exponential, and the best alternative distribution fit for all three Internet traffic traces.

Compared to the $M/M/1$ model, our $H_2/H_2/1$ model provides better modeling accuracy for 27 of the 31 traces, lowering the modeling error by about 76% for these traces. For the remaining 4 traces, the $M/M/1$ results in about 7% lower relative error. Note that the $H_2/H_2/1$ is significantly better than the Generalized Pareto and Lognormal based models for almost all traces.

7. Distribution fitting for internet traces

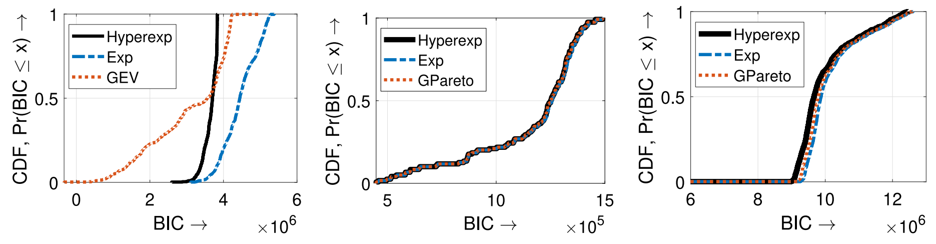
We now explore the applicability of the Hyper-exponential for distribution fitting beyond storage traces. Specifically, we evaluate distribution fitting for the three Internet traffic trace families – Wikipedia, Stackoverflow, and WITS – described in Sections 4.6–4.8. We use the same methods and metrics for fitting as described in Section 5.

All Internet trace families that we studied have IAT information. Figs. 13, 14, 15, 16, and 17, respectively, show the R^2, JSD, log-likelihood, AIC, and BIC metrics for the different Internet trace families under three distribution fits: (i) Hyper-exponential with two phases (H_2), (ii) Exponential, and (iii) the best alternative distribution (apart from Hyper-exponential and Exponential). We see that the Hyper-exponential almost always provides a superior fit. The median metric value achieved for the H_2, as observed in each figure, is always superior when compared to the other distributions, except for the R^2 metric under the Wikipedia trace (Fig. 13(a)) where the median value is slightly lower than that of Burr, but is still quite competitive (median R^2 greater than 0.985). Note that the R^2 value for Hyper-exponential under the Stackoverflow and WITS traces (Figs. 13(b) and 13(c)) is almost always close to 1 for all subtraces. As with the storage workload traces, the best alternative distribution often changes based on the trace family and based on the metric of accuracy.



(a) Wikipedia. (b) Stackoverflow. (c) WITS.

Fig. 16. CDF of Akaike information criterion (AIC, lower is better) for Hyper-exponential, Exponential, and the best alternative distribution fit for all three Internet traffic traces.



(a) Wikipedia. (b) Stackoverflow. (c) WITS.

Fig. 17. CDF of Bayesian information criterion (BIC, lower is better) for Hyper-exponential, Exponential, and the best alternative distribution fit for all three Internet traffic traces.

Table 6
Ranking of the top 3 fitted distributions (top-to-bottom) for *each* Internet traffic trace family. Here, LogL, BSaunders, GPareto, and GEV refer to Loglogistic, Birnbaum–Saunders, Generalized Pareto, and Generalized Extreme Value, respectively.

Wikipedia			Stackoverflow			WITS		
R^2	JSD	LL	R^2	JSD	LL	R^2	JSD	LL
Burr	H_2	GEV	H_2	Burr	GPareto	H_2	H_2	H_2
Weibull	Exponential	H_2	Burr	H_2	H_2	Gamma	BSaunders	GPareto
Gamma	Weibull	GPareto	GPareto	LogL	Burr	Nakagami	Exponential	Weibull

Table 7
Ranking of the top 3 fitted distributions (top-to-bottom), according to AIC and BIC, for *each* Internet traffic trace family. Here, GPareto and GEV refer to Generalized Pareto, and Generalized Extreme Value, respectively.

Wikipedia		Stackoverflow		WITS	
AIC	BIC	AIC	BIC	AIC	BIC
GEV	GEV	GPareto	GPareto	H_2	H_2
H_2	H_2	H_2	H_2	GPareto	GPareto
GPareto	GPareto	Burr	Burr	Weibull	Weibull

To better assess the fitting capabilities of the Hyper-exponential, we show the top 3 distribution fits, using the median accuracy, across *each trace family* and for *all* traces in Tables 6 and 8, respectively. We also show the top 3 distribution fits, using the median AIC and BIC values, across *each trace family* and for *all traces* in Tables 7 and 9, respectively. We see that the Hyper-exponential always ranks in the top 3 for any trace family under all 5 accuracy metrics, except for the R^2 metric under Wikipedia. Importantly, when considering the median metric values for the distribution fit across all traces, the Hyper-exponential always ranks at the top for *all five metrics* (Tables 8 and 9).

In summary, the Hyper-exponential consistently provides superior distribution fit under diverse accuracy metrics for all Internet traffic trace families we consider. The results in this section also show that the Hyper-exponential's superior distribution fit properties (illustrated by some example fits in Fig. 18) extend beyond the storage workload traces.

Table 8
Median R^2, JSD, and log-likelihood across *all* Internet traffic traces for the top 3 distributions in each case, sorted by accuracy.

(a) R^2 (higher is better)

Distribution	R^2
Hyper-exponential	0.9982
Gamma	0.9975
Nakagami	0.9968

(b) JSD (lower is better)

Distribution	JSD
Hyper-exponential	0.0299
Exponential	0.0432
Birnbaum–Saunders	0.0434

(c) Log-likelihood, LL (higher is better)

Distribution	LL
Hyper-exponential	−4687689
Generalized Pareto	−4759002
Weibull	−4775405

Table 9
Median AIC and BIC values across *all* Internet traffic traces for the top 3 distributions in each case, sorted by accuracy.

(a) AIC (lower is better)

Distribution	AIC
Hyper-exponential	9375384
Generalized Pareto	9518010
Weibull	9550813

(b) BIC (lower is better)

Distribution	BIC
Hyper-exponential	9375420
Generalized Pareto	9518046
Weibull	9550837

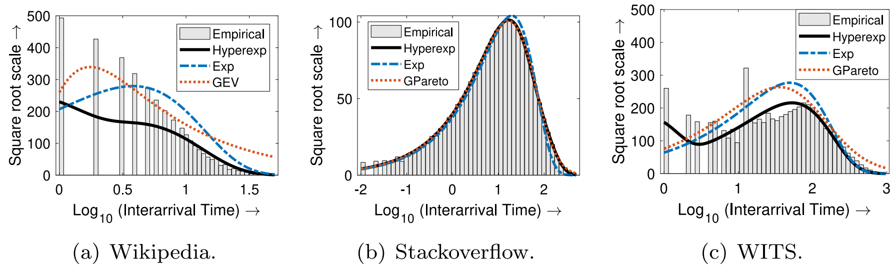


Fig. 18. Results of distribution fit for a sample trace from each Internet traffic trace family for Hyper-exponential, Exponential, and the best alternative distribution (per log-likelihood).

8. Modeling use case: What-if analysis

An immediate and interesting use case for any system modeling approach is what-if analysis. We now present three such what-if analyses enabled by the $H_2/H_2/1$ performance models we presented in Section 6.1.1, covering both storage systems and Internet services.

8.1. Impact of request arrivals on response time

Our first use case analyzes the impact of change in arrival rate of requests (inverse of inter-arrival time, $1/E[IAT]$) on response time. Without loss of generality, we consider the first subtrace shown in Fig. 12, namely, the *msg* subtrace from the mobile storage traces. We use the $H_2/H_2/1$ model, whose Markov chain is shown in Fig. 3, to obtain the mean response time estimates. Note, from Fig. 12, that the $H_2/H_2/1$ model provides an accurate response time estimation for the *msg* subtrace. As discussed in Section 6.1.1, the input to this model is the IAT and ST parameters of the subtrace. We use the IAT and ST Hyper-exponential distribution fit parameters for the *msg* subtrace from Section 5 as inputs for our modeling and what-if analyses in this section.

The black line in Fig. 20(a) shows the results of our analysis; the cross marker on the line corresponds to the arrival rate observed in the *msg* subtrace (about 330 req/s). As the arrival rate increases, the mean response time increases, as expected. When the request rate doubles to 660 req/s, the mean response time increases from about 1.9 ms to 2.7 ms.

We also analyzed the impact of variability in inter-arrival time (IAT). A less variable IAT indicates that the request arrivals are more evenly spaced whereas a more variable IAT indicates that the request arrivals are more unevenly spaced (e.g., more temporally batched or bursty requests). We used the squared coefficient of variation of IAT, C^2_{IAT}, to parameterize variability. C^2_{IAT} is the normalized variability of IAT, and is mathematically defined as the variability in IAT

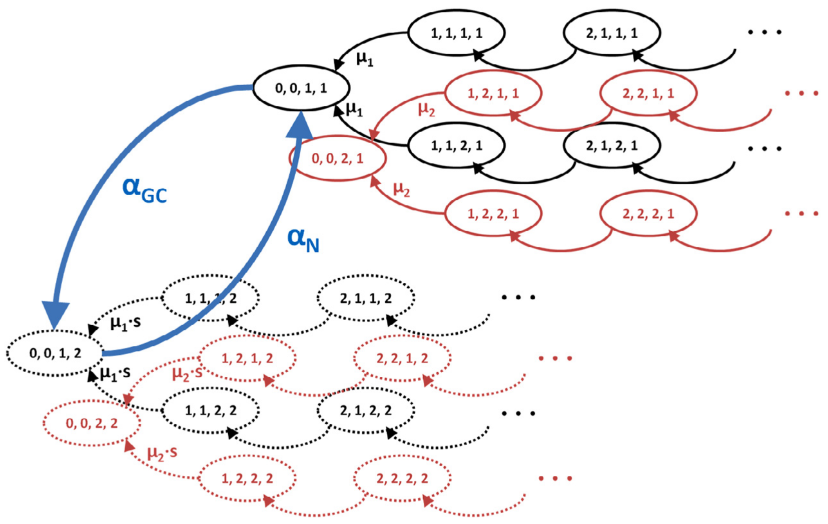


Fig. 19. Illustration of our modified $H_2/H_2/1$ Markov chain that models the impact of garbage collection (GC) on storage system performance. For ease of presentation, we highlight specific only transitions and transition rates in the figure.

divided by the square of the mean IAT. For the *msg* subtrace, $C^2_{IAT} \approx 4$. Note that C^2_{IAT} has no units: larger C^2_{IAT} values imply higher IAT variability. For a fixed IAT or request rate, a doubling of C^2_{IAT} implies a doubling of the IAT variability.

The red, black, and blue lines in Fig. 20(a) show the impact on mean response time under $C^2_{IAT} = 8$, $C^2_{IAT} = 4$, and $C^2_{IAT} = 2$, respectively. We modeled the different IAT variabilities by changing the probability parameter (p) of the IAT hyper-exponential distribution (see Eq. 3.2). We see that IAT variability significantly impacts response time, especially at high arrival rate. When C^2_{IAT} doubles from 2 to 4, mean response time increases by 29% on average, and by up to 53%, for the arrival rate range considered in Fig. 20(a). Likewise, as C^2_{IAT} doubles from 4 to 8, mean response time increases by 33% on average, and by up to 66% at high arrival rates.

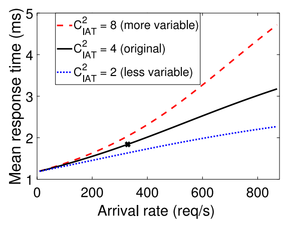
8.2. *Impact of garbage collection's service rate slowdown on response time*

We analyzed a more complex use case using our performance models: the impact of performance degradation events, such as garbage collection (GC) on response time. To analyze this use case, we extended our $H_2/H_2/1$ Markov chain model for the *msg* subtrace to include an additional regime where the service rate of the storage device (inverse of mean service time, $1/E[ST]$) is reduced to represent the degraded service rate under GC; the service rate slowdown under GC is a model parameter that we vary. We also vary the percentage of time spent in GC, with the frequency of GC set to once every minute (configurable).

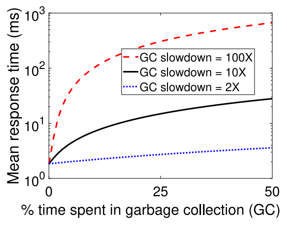
Fig. 19 shows our modified $H_2/H_2/1$ Markov chain; for ease of presentation, we show only specific transitions in the figure. Essentially, this modified Markov chain additionally includes a replica of the chain in Fig. 3 with lower service rates ($\mu_1 \times slowdown$ and $\mu_2 \times slowdown$), representing the state space under GC. In Fig. 19, the states with a dotted boundary represent the replica states when the system is under GC, with service rates lowered by a multiplicative slowdown factor of $s < 1$. The state space is now a quadruple, (i, j, k, l), with the last element denoting whether the system is under GC ($l = 2$) or not ($l = 1$). Transitions are added from every state in the original chain to the corresponding state in the replica chain, with the transition rate determined by the GC frequency. In Fig. 19, we denote the rate of transition of the system to GC by α_{GC}, and the rate of return to "normal" conditions (no GC) by α_N. The fraction of time spent in GC is then given by $\frac{\alpha_N^{-1}}{\alpha_N^{-1}+\alpha_{GC}^{-1}}$.

This extended model is more complex than the $H_2/H_2/1$ model in Fig. 3, and has not been analyzed before, to the best of our knowledge. Although this extended model has an infinite state space, the repeating structure of the underlying Markov chain still allows us to obtain the mean response time numerically as a function of the various parameters, using matrix analytic methods [46]—and in less than one second with a small memory footprint (a couple MBs). Note that our model is only an approximation, because real-world SSDs typically have proprietary firmware whose behavior and parameters are largely unknown [48]; nevertheless, to provide useful results, we explored all model parameters over a range of possible values. (More precise models can be constructed using parameters obtained from the GC bounds of a specific storage device under a given workload.)

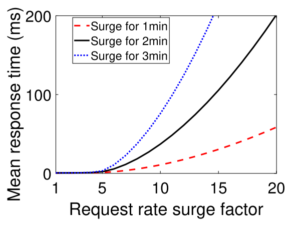
Fig. 20(b) shows the results of our analysis; note the log scale on the y-axis. Here, the arrival rate is the same as that observed in the *msg* subtrace (330 req/s). The range of parameters (percentage time spent in GC and GC slowdown) in the figure was chosen based on numbers reported in prior studies on GC [73,74]. We see that the service rate slowdown



(a) Impact of request arrivals.



(b) Impact of garbage collection.



(c) Impact of flash crowds.

Fig. 20. Results of our what-if analysis. Figure (a) shows the impact of arrival rate on response time under different inter-arrival time (IAT) variabilities. Figure (b) shows the impact of time spent in garbage collection (GC) on response time for different GC service rate slowdowns. Figure (c) shows the impact of intensity of flash crowds on response time for different flash crowd durations.

significantly impacts response time, even when the fraction of time spent in GC is quite small. For example, even if we spend only 5% of the time in GC, the mean response time increases to 2 ms, 4.4 ms, and 40.9 ms, under 2×, 10×, and 100× service rate slowdown, respectively. By contrast, without GC, the response time is about 1.8 ms. We also tried other GC frequencies, once every 6s and once every 600 s, and found the results (and trends) to be qualitatively similar. In general, our above models can also be employed for analyzing other use cases, such as the impact of device aging (by considering multiple service rate slowdown regimes) or the impact of newer hardware (faster service rate) on response time.

8.3. Impact of flash crowds on response time

Our last use case analyzes the impact of flash crowds, a common occurrence for Internet services [25,75], on the response time of requests. We model flash crowds by extending our $H_2/H_2/1$ Markov chain model to include an additional regime where the arrival rate of requests is increased (by a multiplicative factor, *surge*) to represent the surge in traffic; the time spent in this regime and the increased request rate are configurable model parameters that we vary. The resulting Markov chain is similar to the one illustrated in Fig. 19, but this time there is no slowdown in service rate in the replica chain (dotted states); instead, there is an increase in request rate for this regime (from λ_1 to $\lambda_1 \times surge$ and from λ_2 to $\lambda_2 \times surge$).

Again, this extended model is more complex than the $H_2/H_2/1$ model in Fig. 3, and has not been analyzed before, to the best of our knowledge. We employ matrix analytic methods [46] to efficiently obtain the mean response time numerically for this model. Although only an approximation, we believe that the results obtained from this model provide us with expected trends in performance as the intensity and duration of the flash crowd changes.

We consider a Wikipedia access trace (specifically, trace 1191201596, for which the Hyper-exponential distribution resulted in the best fit) and employ its H_2 fit model parameters for the $H_2/H_2/1$ Markov chain IAT distribution. For this trace, $C^2_{IAT} = 2.8$ with mean request rate of 263 req/s. For modeling the H_2 ST, we scale the IAT parameters to obtain a load (or system utilization) of 20%, consistent with the low utilization values experienced by Internet host servers [76–78]. To model a flash crowd, we scale the λ_1 and λ_2 values for the IAT, as explained above, while keeping all other parameters constant.

Fig. 20(c) shows the results of our analysis. We vary the flash crowd surge factor from 1–20, based on flash crowd peak sizes reported by prior work [75,79]. We consider three different flash crowd durations, with flash crowd frequency of once per hour: 1, 2, and 3 min; these values are based on flash crowd durations reported by prior empirical work [80,81]. When there is no flash crowd (corresponding to an x-value of 1), the mean response time is about 0.6 ms. For a 5× larger flash crowd, the response time increases to about 3 ms. For a 20× larger flash crowd, the response time substantially increases to about 59 ms, 201 ms, and 411 ms, for flash crowd durations of 1, 2, and 3 min, respectively; this corresponds to a response time increase (relative to no flash crowds) of almost 100×, 335×, and 685×, respectively. Clearly, the response time suffers significantly as the flash crowd duration increases.

The above three use cases highlight the benefits of our distribution fitting based performance modeling approach. Without such models, the above what-if analysis would require extensive experimentation and might even be infeasible.

9. Related work

9.1. Analyzing disk access patterns

Storage workloads have been the focus of analysis since at least as far back as the 1980s when disk access patterns were studied. Ruemmler and Wilkes [38] presented an analysis of disk access patterns on three HP-UX systems collected in 1992. Their analysis focused on the volume of read versus write traffic, and the nature of the traffic, such as sequential, synchronous, swap traffic, etc. A similar study was later conducted by Keeton et al. [82] for disk block traces from a mail server and a database server in 2000. While both papers analyze skew in I/O load across devices, distribution fitting of the I/O traffic is not considered. Gomez and Santonja [2] later analyzed and specifically modeled the disk access patterns for the traces provided by Ruemmler and Wilkes. They found that the spatial access pattern is well modeled as a heavy-tailed Pareto distribution; however, the fitted parameters do result in infinite variance.

Verma et al. [49] analyzed block-level I/O traces from various servers at FIU's Computer Science department. The analysis focused on the usage of working sets, variability in workload intensity, and read-idle time distribution. The authors observed that data usage was highly skewed, but did not consider distribution fitting.

9.2. Analyzing request-level storage traces

Kavalanekar et al. [22] analyzed the characteristics of storage workload traces from production servers at Microsoft. The analysis focused on block-level statistics, file access frequencies, and temporal and spatial self-similarity. IAT (inter-arrival time) analysis was also conducted, focusing on the visualization of the IAT histograms and the overall rate of requests, but not on distribution fitting.

Gracia-Tinedo et al. [14] analyzed user-level storage workloads for the personal cloud service, UbuntuOne. The analysis shows that some of the inter-operation times, such as those for Upload, are not well approximated by the Exponential distribution and are better approximated by the Pareto distribution.

Zhou et al. [23] analyzed block-level I/O traces from common applications, such as email and YouTube, on a Nexus 5 smartphone equipped with a Flash-based eMMC (embedded multimedia card) device. The analysis focused on the size of the requests (or service times, STs) received by the eMMc device and their access type (read versus write). IATs of the traces are also considered, but the analysis is restricted to mean IAT and the extent of batching among requests.

In our analysis here we considered several of the above traces (that are publicly available on SNIA's repository [21]), but focused on the *distribution fitting* of their IAT and ST traces, separated by reads and writes. By finding an accurate distribution fit, we enabled queueing-based performance models that can accurately predict the response time for requests.

9.3. Analyzing the aggregate storage volume of workloads

Leung et al. [83] analyzed the traffic for two enterprise file servers deployed at NetApp. The paper analyzes various characteristics of file I/O traffic such as volume, lifetimes, access frequency, etc. The authors do find that several of the characteristics are heavy-tailed, but do not identify a specific distribution fit. Birke et al. [39] analyzed storage workloads on VMs in an enterprise private cloud and found that the VM-level storage capacity and used storage volume can be well approximated by an Exponential distribution. Mei et al. [84] analyzed and modeled a few traces from MSR and found that the spatio-temporal behavior is well modeled as a Gaussian.

Seo et al. [85] presented a data-mining and clustering approach to classify and thus characterize I/O workloads using previously published deduplication traces [21,58]. While classification is a useful characterization of workloads, the classified workloads and clusters are not intuitive and require further analysis. For instance, while the authors do use the mean IAT as a feature, the clustering results do not provide any information about the IAT distribution within the cluster.

9.4. Analyzing the network traffic of storage workloads

Lee et al. [24] analyzed storage traffic on servers that host commercial virtual desktop infrastructure (VDI) VMs. The analysis is focused on the traffic volume and burstiness of specific applications on these VDI servers. A similar analysis was conducted on a smaller set of traces by Shamma et al. [86]. Drago et al. [87] analyzed the network traffic to Dropbox from home networks and found that a small number of users are responsible for most of the traffic.

9.5. Analyzing file system characteristics

Prior work has also focused on analyzing file system characteristics, such as file size, file count, directory count, etc. For example, the file size distribution was found to be well modeled by a Lognormal [88], Lambda [89] (similar to Normal and Logistic), or Bimodal [90,91] distribution, depending on the source of the trace. A recent work, Impressions [92], focused on generating realistic file system images with associated metadata to facilitate performance benchmarking of file systems. Our focus in this article is on *request-level* modeling for IAT and ST (service time). ST is the *time* required to service a request on a device, and is thus different from file size.

9.6. Other storage analysis works

There are several other storage analysis studies that focus on other metrics such as device failures [93,94], data corruption [95], data deduplication, etc. These are orthogonal to our article: we focus specifically on request-level distribution fitting and performance modeling.

10. Conclusions

Storage workload modeling is critical to optimizing storage systems—often the slowest component of any system. Good models depend on an accurate characterization of inter-arrival times and service requirements. Many distributions exist that have been found to accurately model behavior in other domains, but not for storage systems—in part because storage systems exhibit multi-modalities and long tails [48,96]. And some distributions that do fit storage systems, however, do not provide analytical properties that can be used to, say, build an accurate and efficient performance model for storage systems. This article makes the following contributions:

1. We undertook a detailed study of distribution fitting for storage workloads, using over 250 traces from five different sources, and evaluated their fitness using 20 different probability distributions under 5 diverse accuracy metrics.
2. We discovered that the seldom used *Hyper-exponential* distribution provided the best fit based on all five metrics of accuracy. Moreover, we found that only two phases were needed to make this distribution fit well: more phases did not improve accuracy by much, and took longer to fit compared to other distributions.
3. This Hyper-exponential distribution with two terms (H_2) is amenable to performance modeling. We built such a model and evaluated it in predicting storage performance. Whereas the few other distributions (e.g., Exponential) that do enable modeling resulted in at least 48% median modeling error, and as high as 361% error, H_2's median error was under 18%.
4. To assess the applicability of the Hyper-exponential for distribution fitting beyond storage traces, we evaluated distribution fitting for Internet traffic traces using over 1600 traces from 3 different sources. We again found that the Hyper-exponential distribution provided a superior fit compared to other probability distributions.
5. We employed and extended our Hyper-exponential–based performance model to conduct three different what-if analyses. First, we highlighted the severe impact of workload variability on response time. Second, we investigated the performance impact of different parameters of garbage collection; we found that even if garbage collection is only active for a fraction of time, performance can degrade by as much $20\times$. Third, we investigated the impact of flash crowds on the response time of internet services; we found that flash crowds lasting only a couple of minutes can increase response time by as much as $335\times$.

In the near term we plan to collect a larger and longer-term set of traces to analyze. For example, We are interested in how H_2's parameters may change over time as, say, the workloads change. In general, we expect to redo the distribution fitting only in response to a significant change in the workload, and not just a change in the mean arrival rate; such significant changes in the workload can be detected by existing techniques, such as change point detection [97,98].

There is a shortage of traces with timing information from *inside* devices. We plan to purchase, instrument, and capture traces with internal device service times (e.g., using Open-Channel SSDs from CNEX Labs). Such data sets will be invaluable when released and for our future performance modeling efforts. Our long-term goal is to develop an I/O scheduler for the Linux kernel that leverages our H_2-based performance model, prove its efficacy, and release it publicly. Scheduling I/Os and predicting performance for just a single device is challenging. Our longer term plans therefore include evaluating the Hyper-exponential distribution for more complex environments such as multi-tier and hybrid storage, virtualized storage, RAID, LVM—especially where devices of very different properties may be used (e.g., NVMe+SSD+SMR). We will start by developing H_2 models for each device because there is no known theory currently for developing a single unified model for a composite system. However, queueing theory suggests that, with some limitations, it may be possible to mathematically *compose* certain individual models (including Hyper-exponential–based models) into a single unified performance model [99–101].

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was made possible in part thanks to Microsoft Azure, Dell-EMC, NetApp, and IBM support; NSF awards 1251137, 1302246, 1305360, 1622832, 1650499, and 1730726; ONR award N00014-16-1-2264; and a Stony Brook OVPR Seed Grant.

References

[1] Berk Atikoglu, Yuehai Xu, Eitan Frachtenberg, Song Jiang, Mike Paleczny, Workload analysis of a large-scale key-value store, in: Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE joint international conference on Measurement and Modeling of Computer Systems, 2012, pp. 53–64.

[2] María Engracia Gomez, Vicente Santonja, A new approach in the analysis and modeling of disk access patterns, in: 2000 IEEE International Symposium on Performance Analysis of Systems and Software. ISPASS (Cat. No. 00EX422), IEEE, 2000, pp. 172–177.

[3] Leonard Kleinrock, Queueing Systems, Volume 2, Wiley-Interscience, New York, 1976.

[4] Leonard Kleinrock, Queueing Systems, Volume I: Theory, Wiley-Interscience, New York, USA, 1975.
[5] M. Harchol-Balter, Performance Modeling and Design of Computer Systems: Queueing Theory in Action, Cambridge University Press, Cambridge, UK, 2013.
[6] Anshul Gandhi, Yuan Chen, Daniel Gmach, Martin Arlitt, Manish Marwah, Minimizing data center sla violations and power consumption via hybrid resource provisioning, in: 2011 International Green Computing Conference and Workshops, IEEE, 2011, pp. 1–8.
[7] Xi Chen, Lukas Rupprecht, Rasha Osman, Peter Pietzuch, Felipe Franciosi, William Knottenbelt, Cloudscope: diagnosing and managing performance interference in multi-tenant clouds, in: 2015 IEEE 23rd international symposium on modeling, analysis, and simulation of computer and telecommunication systems, IEEE, 2015, pp. 164–173.
[8] Christopher Stewart, Aniket Chakrabarti, Rean Griffith, Zoolander: efficiently meeting very strict, low-latency slos, in: 10th International Conference on Autonomic Computing ({ICAC} 13), 2013, pp. 265–277.
[9] Zhao Lucis Li, Chieh-Jan Mike Liang, Wenjia He, Lianjie Zhu, Wenjun Dai, Jin Jiang, Guangzhong Sun, Metis: robustly tuning tail latencies of cloud systems, in: 2018 {USENIX} Annual Technical Conference ({USENIX}{ATC} 18), 2018, pp. 981–992.
[10] Călin Iorgulescu, Reza Azimi, Youngjin Kwon, Sameh Elnikety, Manoj Syamala, Vivek Narasayya, Herodotos Herodotou, Paulo Tomita, Alex Chen, Jack Zhang, et al., Perfiso: performance isolation for commercial latency-sensitive services, in: 2018 {USENIX} Annual Technical Conference ({USENIX}{ATC} 18), 2018, pp. 519–532.
[11] Shiqin Yan, Huaicheng Li, Mingzhe Hao, Michael Hao Tong, Swaminathan Sundararaman, Andrew A Chien, Haryadi S Gunawi, Tiny-tail flash: near-perfect elimination of garbage collection tail latencies in nand ssds, ACM Transactions on Storage (TOS) 13 (3) (2017) 1–26.
[12] Jun He, Duy Nguyen, Andrea Arpaci-Dusseau, Remzi Arpaci-Dusseau, Reducing file system tail latencies with chopper, in: 13th {USENIX} Conference on File and Storage Technologies ({FAST} 15), 2015, pp. 119–133.
[13] Mingzhe Hao, Gokul Soundararajan, Deepak Kenchammana-Hosekote, Andrew A Chien, Haryadi S Gunawi, The tail at store: a revelation from millions of hours of disk and {SSD} deployments, in: 14th {USENIX} Conference on File and Storage Technologies ({FAST} 16), 2016, pp. 263–276.
[14] Raúl Gracia-Tinedo, Yongchao Tian, Josep Sampé, Hamza Harkous, John Lenton, Pedro García-López, Marc Sánchez-Artigas, Marko Vukolic, Dissecting ubuntuone: autopsy of a global-scale personal cloud back-end, in: Proceedings of the 2015 Internet Measurement Conference, 2015, pp. 155–168.
[15] Hui Li, David Groep, Lex Wolters, Workload characteristics of a multi-cluster supercomputer, in: Workshop on Job Scheduling Strategies for Parallel Processing, Springer, 2004, pp. 176–193.
[16] Paul Barford, Mark Crovella, Generating representative web workloads for network and server performance evaluation, in: Proceedings of the 1998 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems, 1998, pp. 151–160.
[17] A. Iosup, D. Epema, Grid computing workloads, IEEE Internet Comput. 15 (2) (2011) 19–26.
[18] H. Cramer, Mathematical Methods of Statistics, Princeton University Press, Princeton, NJ, USA, 1946.
[19] Daniel L. McFadden, Modelling the choice of residential location, in: Spatial Interaction Theory and Residential Location, New Haven, CT, USA, 1978, pp. 75–96.
[20] Pandu R. Tadikamalla, A look at the Burr and related distributions, Internat. Statist. Rev. 48 (3) (1980) 337–344.
[21] SNIA IOTTA Repository, http://iotta.snia.org/traces, Storage Networking Industry Association.
[22] Swaroop Kavalanekar, Bruce Worthington, Qi Zhang, Vishal Sharda, Characterization of storage workload traces from production windows servers, in: 2008 IEEE International Symposium on Workload Characterization, IEEE, 2008, pp. 119–128.
[23] Deng Zhou, Wen Pan, Wei Wang, Tao Xie, I/o characteristics of smartphone applications and their implications for emmc design, in: 2015 IEEE International Symposium on Workload Characterization, IEEE, 2015, pp. 12–21.
[24] Chunghan Lee, Tatsuo Kumano, Tatsuma Matsuki, Hiroshi Endo, Naoto Fukumoto, Mariko Sugawara, Understanding storage traffic characteristics on enterprise virtual desktop infrastructure, in: Proceedings of the 10th ACM International Systems and Storage Conference, 2017, pp. 1–11.
[25] Martin Arlitt, Tai Jin, Workload characterization of the 1998 World Cup web site, IEEE Netw. 14 (2000) 30–37.
[26] Bhuvan Urgaonkar, Giovanni Pacifici, Prashant Shenoy, Mike Spreitzer, Asser Tantawi, An analytical model for multi-tier internet services and its applications, ACM SIGMETRICS Performance Evaluation Review 33 (1) (2005) 291–302.
[27] Yiyu Chen, Amitayu Das, Wubi Qin, Anand Sivasubramaniam, Qian Wang, Natarajan Gautam, Managing server energy and operational costs in hosting centers, in: Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, 2005, pp. 303–314.
[28] D.R. Cox, A use of complex probabilities in the theory of stochastic processes, Math. Proc. Camb. Phil. Soc. 51 (2) (1955) 313–319.
[29] Norman R Draper, Harry Smith, Applied regression analysis, 326, John Wiley & Sons, Hoboken, NJ, USA, 1998.
[30] J. Lin, Divergence measures based on the Shannon entropy, IEEE Trans. Inform. Theory 37 (1) (1991) 145–151.
[31] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, J. R. Stat. Soc. Ser. B 39 (1) (1977) 1–38.
[32] Hirotogu Akaike, Information theory and an extension of the maximum likelihood principle, in: Selected Papers of Hirotugu Akaike, Springer, New York, NY, USA, 1998, pp. 199–213.
[33] Gideon Schwarz, et al., Estimating the dimension of a model, Ann. Statist. 6 (2) (1978) 461–464.
[34] David Freedman, Statistical Models: Theory and Practice, Cambridge University Press, 2005.
[35] S. Geisser, W.O. Johnson, Modes of Parametric Statistical Inference, in: Wiley Series in Probability and Statistics, Wiley, 2006.
[36] D.N. Moriasi, J.G. Arnold, M.W. Van Liew, R.L. Bingner, R.D. Harmel, T.L. Veith, Model evaluation guidelines for systematic quantification of accuracy in watershed simulations, Trans. Am. Soc. Agric. Biol. Eng. 50 (3) (2007) 885–900.
[37] A.J. Jakeman, R.A. Letcher, J.P. Norton, Ten iterative steps in development and evaluation of environmental models, Environ. Model. Softw. 21 (5) (2006) 602–614.
[38] Chris Ruemmler, John Wilkes, Unix disk access patterns, in: USENIX Winter, USENIX Association, San Diego, CA, USA, 1993, pp. 405–420.
[39] Robert Birke, Mathias Bjoerkqvist, Lydia Y. Chen, Evgenia Smirni, Ton Engbersen, (Big)data in a virtualized world: volume, velocity, and variety in cloud datacenters, in: 12th USENIX Conference on File and Storage Technologies (FAST 14), USENIX Association, Santa Clara, CA, 2014, pp. 177–189, https://www.usenix.org/conference/fast14/technical-sessions/presentation/birke.
[40] Larry N. Singh, G.R. Dattatreya, Estimation of the hyperexponential density with applications in sensor networks, Int. J. Distrib. Sens. Netw. 3 (3) (2007) 311–330.
[41] H.T. Papadopoulos, Cathal Heavey, Queueing theory in manufacturing systems analysis and design: A classification of models for production and transfer lines, European J. Oper. Res. 92 (1) (1996) 1–27.
[42] Mitsuru Ohba, Software reliability analysis models, IBM J. Res. Dev. 28 (4) (1984) 428–443.
[43] Anja Feldmann, Ward Whitt, Fitting mixtures of exponentials to long-tail distributions to analyze network performance models, Perform. Eval. 31 (3–4) (1998) 245–279.

[44] Jayakrishnan Nair, Adam Wierman, Bert Zwart, The fundamentals of heavy-tails: properties, emergence, and identification, in: Proceedings of the ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems, in: SIGMETRICS âĂŹ13, vol. 2, Association for Computing Machinery, Pittsburgh, PA, USA, 2013, p. 387âĂŞ388, New York, NY, USA.
[45] J.D.C. Little, A proof of the queueing formula $L = \lambda W$, Oper. Res. 9 (1961) 383–387.
[46] G. Latouche, V. Ramaswami, Introduction to Matrix Analytic Methods in Stochastic Modeling, ASA-SIAM, Philadelphia, PA, USA, 1999.
[47] J.F.C. Kingman, The single server queue in heavy traffic, Math. Proc. Camb. Phil. Soc. 57 (4) (1961) 902–904.
[48] Abutalib Aghayev, Mansour Shafaei, Peter Desnoyers, Skylight—a window on shingled disk operation, ACM Transactions on Storage (TOS) 11 (4) (2015) 1–28.
[49] Akshat Verma, Ricardo Koller, Luis Useche, Raju Rangaswami, Srcmap: energy proportional storage using dynamic consolidation, in: Proceedings of the 8th USENIX Conference on File and Storage Technologies, in: FAST'10, USENIX Association, San Jose, California, USA, 2010, p. 20.
[50] Carl A. Waldspurger, Nohhyun Park, Alexander Garthwaite, Irfan Ahmad, Efficient MRC construction with SHARDS, in: 13th USENIX Conference on File and Storage Technologies, FAST 15, USENIX Association, Santa Clara, CA, 2015, pp. 95–110.
[51] Consolidate applications with less hardware with vSphere hypervisor, 2020, VMware, https://www.vmware.com/products/vsphere-hypervisor.html.
[52] Irfan Ahmad, Easy and efficient disk i/o workload characterization in vmware esx server, in: 2007 IEEE 10th International Symposium on Workload Characterization, IEEE, Boston, MA, USA, 2007, pp. 149–158.
[53] Guido Urdaneta, Guillaume Pierre, Maarten van Steen, Wikipedia workload analysis for decentralized hosting, Comput. Netw. 53 (11) (2009) 1830–1845.
[54] Wikipedia access traces, WikiBench, http://www.wikibench.eu/?page_id=60.
[55] Stack exchange data dump: Stack exchange, inc., 2019, Internet Archive, https://archive.org/details/stackexchange.
[56] Richard Nelson, Daniel Lawson, Perry Lorier, Analysis of long duration traces, ACM SIGCOMM Comput. Commun. Rev. 35 (1) (2005) 45–52.
[57] WITS: ISPDSL-II, 2013, WAND Group, https://wand.net.nz/wits/ispdsl/2/.
[58] Ricardo Koller, Raju Rangaswami, I/O deduplication: Utilizing content similarity to improve I/O performance, ACM Transactions on Storage (TOS) 6 (3) (2010) 13:1–13:26.
[59] M. Wajahat, A. Yele, Distribution fitting to traces using MATLAB, 2020, https://github.com/PACELab/h2fitting.
[60] Zsolt Ugray, Leon Lasdon, John Plummer, Fred Glover, James Kelly, Rafael Martí, Scatter search and local NLP solvers: A multistart framework for global optimization, INFORMS J. Comput. 19 (3) (2007) 328–340.
[61] Fred Glover, A template for scatter search and path relinking, Lecture notes in computer science 1363 (1998) 13–54.
[62] R.A. Waltz, J.L. Morales, J. Nocedal, D. Orban, An interior algorithm for nonlinear optimization that combines line search and trust region steps, Math. Program. 107 (3) (2006) 391–408.
[63] Richard H. Byrd, Mary E. Hribar, Jorge Nocedal, An interior point algorithm for large-scale nonlinear programming, SIAM J. Optim. 9 (4) (1999) 877–900.
[64] D.L.J. Alexander, A. Tropsha, David A. Winkler, Beware of R2: Simple, unambiguous assessment of the prediction accuracy of QSAR and QSPR models, J. Chem. Inf. Model. 55 (7) (2015) 1316–1322.
[65] Bent Fuglede, Flemming Topsoe, Jensen-shannon divergence and hilbert space embedding, in: International Symposium onInformation Theory, 2004. ISIT 2004. Proceedings., IEEE, 2004, p. 31.
[66] Larry Wasserman, All of Statistics: A Concise Course in Statistical Inference, Springer Publishing Company, Incorporated, New York, NY, USA, 2010.
[67] Christian P. Robert, George Casella, Introducing Monte Carlo Methods with R (Use R), first ed., Springer-Verlag, New York, NY, USA, 2009.
[68] Christophe Biernacki, Gilles Celeux, Gerard Govaert, Choosing starting values for the EM algorithm for getting the highest likelihood in multivariate Gaussian mixture models, Comput. Statist. Data Anal. 41 (3) (2003) 561–575.
[69] Jianzong Wang, Peter Varman, Changsheng Xie, Avoiding performance fluctuation in cloud storage, in: 2010 International Conference on High Performance Computing, IEEE, Dona Paula, India, 2010, pp. 1–9.
[70] Giuseppe Lettieri, Vincenzo Maffione, Luigi Rizzo, A study of I/O performance of virtual machines, Comput. J. 61 (6) (2018) 808–831.
[71] Organisation for Economic Co-operation and Development (OECD), What are Equivalence Scales? http://www.oecd.org/eco/growth/OECD-Note-EquivalenceScales.pdf.
[72] V.N. Tarasov, Analysis of queues with hyperexponential arrival distributions, Probl. Inf. Transm. 52 (1) (2016) 14–23.
[73] Li -Pin Chang, Tei -Wei Kuo, Shi -Wu Lo, Real-time garbage collection for flash-memory storage systems of real-time embedded systems, ACM Trans. Embed. Comput. Syst. 3 (4) (2004) 837–863.
[74] Myoungsoo Jung, Ramya Prabhakar, Mahmut Taylan Kandemir, Taking garbage collection overheads off the critical path in ssds, in: ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing, Springer, 2012, pp. 164–186.
[75] Anshul Gandhi, Timothy Zhu, Mor Harchol-Balter, Michael A Kozuch, Softscale: stealing opportunistically for transient scaling, in: ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing, Springer, 2012, pp. 142–163.
[76] Christina Delimitrou, Christos Kozyrakis, Quasar: resource-efficient and qos-aware cluster management, in: Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems, in: ASPLOS âĂŹ14, Association for Computing Machinery, New York, NY, USA, 2014, p. 127âĂŞ144, Salt Lake City, Utah, USA.
[77] Arunchandar Vasan, Anand Sivasubramaniam, Vikrant Shimpi, T Sivabalan, Rajesh Subbiah, Worth their watts?-an empirical study of datacenter servers, in: HPCA-16 2010 The Sixteenth International Symposium on High-Performance Computer Architecture, IEEE, 2010, pp. 1–10.
[78] Hailong Yang, Alex Breslow, Jason Mars, Lingjia Tang, Bubble-flux: precise online qos management for increased utilization in warehouse scale computers, in: Proceedings of the 40th Annual International Symposium on Computer Architecture, in: ISCA âĂŹ13, Association for Computing Machinery, New York, NY, USA, 2013, pp. 607–618, https://doi.org/10.1145/2485922.2485974, Tel-Aviv, Israel.
[79] Timothy Zhu, Anshul Gandhi, Mor Harchol-Balter, Michael A. Kozuch, Saving Cash by Using Less Cache, in: Presented as part of the, USENIX, Submitted, https://www.usenix.org/conference/hotcloud12/saving-cash-using-less-cache.
[80] Bo Li, Gabriel Y Keung, Susu Xie, Fangming Liu, Ye Sun, Hao Yin, An empirical study of flash crowd dynamics in a p2p-based live video streaming system, in: IEEE GLOBECOM 2008-2008 IEEE Global Telecommunications Conference, IEEE, 2008, pp. 1–5.
[81] Patrick Wendell, Michael J. Freedman, Going viral: flash crowds in an open cdn, in: Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference, in: IMC '11, Association for Computing Machinery, New York, NY, USA, 2011, p. 549âĂŞ558, https://doi.org/10.1145/2068816.2068867.
[82] Kimberly Keeton, Alistair Veitch, Doug Obal, John Wilkes, I/o characterization of commercial workloads, in: Proceedings of the 3rd Workshop on Computer Architecture Evaluation using Commercial Workloads, IEEE, Toulouse, France, 2000.
[83] Andrew W Leung, Shankar Pasupathy, Garth R Goodson, Ethan L Miller, Measurement and analysis of large-scale network file system workloads., in: USENIX annual technical conference, 1, (2) Boston, MA, USA, 2008, 5–2.
[84] Li Mei, Gao Xu, Wu Yanjun, Zhao Chen, Li Mingshu, Characterizing the spatio-temporal burstiness of storage workloads, in: Proceedings of the 5th International Workshop on Cloud Data and Platforms, in: CloudDP '15, Association for Computing Machinery, New York, NY, USA, 2015, https://doi.org/10.1145/2744210.2744211, Bordeaux, France.

[85] B. Seo, S. Kang, J. Choi, J. Cha, Y. Won, S. Yoon, IO workload characterization revisited: A data-mining approach, IEEE Trans. Comput. 63 (12) (2014) 3026–3038.

[86] Mohammad Shamma, Dutch T Meyer, Jake Wires, Maria Ivanova, Norman C Hutchinson, Andrew Warfield, Capo: recapitulating storage for virtual desktops., in: FAST, 11, USENIX Association, 2011, pp. 31–45.

[87] Idilio Drago, Marco Mellia, Maurizio M. Munafo, Anna Sperotto, Ramin Sadre, Aiko Pras, Inside dropbox: understanding personal cloud storage services, in: Proceedings of the 2012 Internet Measurement Conference, in: IMC âĂŹ12, Association for Computing Machinery, New York, NY, USA, 2012, p. 481âĂŞ494, https://doi.org/10.1145/2398776.2398827, Boston, Massachusetts, USA.

[88] John R Douceur, William J Bolosky, A large-scale study of file-system contents, ACM SIGMETRICS Performance Evaluation Review 27 (1) (1999) 59–70.

[89] Kylie Evans, Geoffrey Kuenning, A study of irregularities in file-size distributions, International Symposium on Performance Evaluation of Computer and Telecommunication Systems (2002).

[90] Songbin Liu, Xiaomeng Huang, Haohuan Fu, Guangwen Yang, Understanding data characteristics and access patterns in a cloud storage system, in: 2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, IEEE, Delft, Netherlands, 2013, pp. 327–334.

[91] Nitin Agrawal, William J. Bolosky, John R. Douceur, Jacob R. Lorch, A five-year study of file-system metadata, ACM Transactions on Storage (TOS) 3 (3) (2007).

[92] Nitin Agrawal, Andrea C Arpaci-Dusseau, Remzi H Arpaci-Dusseau, Generating realistic impressions for file-system benchmarking, ACM Transactions on Storage (TOS) 5 (4) (2009) 1–30.

[93] Iyswarya Narayanan, Di Wang, Myeongjae Jeon, Bikash Sharma, Laura Caulfield, Anand Sivasubramaniam, Ben Cutler, Jie Liu, Badriddine Khessib, Kushagra Vaid, Ssd failures in datacenters: what? when? and why?, in: Proceedings of the 9th ACM International on Systems and Storage Conference, in: SYSTOR âĂŹ16, Association for Computing Machinery, New York, NY, USA, 2016, https://doi.org/10.1145/2928275.2928278, Haifa, Israel.

[94] Bianca Schroeder Garth A Gibson, Disk failures in the real world: what does an mttf of 1,000,000 hours mean to you?, in: Proceedings of the 5th USENIX Conference on File and Storage Technologies, FAST, San Jose, CA, USA, 2007, USENIX.

[95] Lakshmi N. Bairavasundaram, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau, Garth R. Goodson, Bianca Schroeder, An analysis of data corruption in the storage stack, ACM Transactions on Storage (TOS) 4 (3) (2008) 8:1–8:28.

[96] N. Joukov, A. Traeger, R. Iyer, C.P. Wright, E. Zadok, Operating system profiling via latency analysis, in: Proceedings of the 7th Symposium on Operating Systems Design and Implementation, OSDI 2006, ACM SIGOPS, Seattle, WA, 2006, pp. 89–102.

[97] Kawahara Yoshinobu, Sugiyama Masashi, Change-Point Detection in Time-Series Data by Direct Density-Ratio Estimation, in: Proceedings of the 2009 SIAM International Conference on Data Mining, 389-400, https://epubs.siam.org/doi/abs/10.1137/1.9781611972795.34.

[98] A.G. Tartakovsky, B.L. Rozovskii, R.B. Blazek, Hongjoong Kim, A novel approach to detection of intrusions in computer networks via adaptive sequential and batch-sequential change-point detection methods, IEEE Trans. Signal Process. 54 (9) (2006) 3372–3382.

[99] James R. Jackson, Networks of waiting lines, Oper. Res. 5 (4) (1957) 518–521.

[100] James R. Jackson, Jobshop-like queueing systems, Manage. Sci. 10 (1) (1963) 131–142.

[101] J. Walrand, An Introduction to Queueing Networks, Prentice Hall, New Jersey, USA, 1988.