Using Parametric Models to Represent Private Cloud Workloads

UCSB Technical Report No. 2013-05

Rich Wolski and John Brevik

***Abstract*—Cloud computing has become a popular metaphor for dynamic and secure self-service access to computational and storage capabilities. In this study, we analyze and model workloads gathered from enterprise-operated commercial private clouds that implement "Infrastructure as a Service." Our results show that 3-phase hyperexponential distributions fit using the Estimation Maximization (E-M) algorithm capture workload attributes accurately. In addition, these models of individual attributes compose to produce estimates of overall cloud performance that our results verify to be accurate.**

As an early study of commercial enterprise private clouds, this work provides guidance to those researching, designing, or maintaining such installations. In particular, the cloud workloads under study do not exhibit "heavy-tailed" distributional properties in the same way that "bare metal" operating systems do, potentially leading to different design and engineering tradeoffs.

***Index Terms*—cloud workload, parametric modeling, performance evaluation**

I. Introduction

Cloud Computing [1] is a technological approach to Information Technology management that carries the promise of lower costs through greater efficiencies. Often associated with the phrase "Infrastructure as a Service" (IaaS) and originally conceived of as a utility computing model [2], the innovation stems from the use of scalable automation (developed to support e-commerce) to implement data center resource provisioning and access.

Since its inception, two categories of IaaS cloud computing deployments have emerged. *Public clouds* are computing utilities operated by service providers who provide access via the Internet. On the other hand, *private clouds* implement the cloud computing model as an infrastructure management platform within a private data center (e.g. one managed by a company's IT organization). While these deployment styles share a common technological approach, their usage patterns and technical rationale differ.

In particular, IaaS private clouds implement a self-service model so that users may automatically obtain resources that are managed by a separate IT organization. Users are free to customize and archive their application environments in the form of virtual machine images. When a user wishes to instantiate an application, she or he specifies the image or images to use to instantiate one or more virtual machines (VMs) that will host the application's components. Typically, a private cloud also conveys quality of service attributes that are associated with the underlying resources (users can know what types of resources they are accessing), and the cloud administers "charge-back" accounting and resource quotas rather than fee-based charging.

In this article, *we describe, analyze, and model workloads gathered from several production private cloud deployments*. Unlike approaches based strictly on empirical distributions and some form of clustering and/or regression [3], [4], [5], [6], [7] our study investigates the development of parametric statistical models similar in spirit to the parametric approaches used by those studying operating system workload migration [8], [9], service workload modeling [10], and data center workload characterization [11]. This analysis is useful for infrastructure capacity planning and cost estimation, since it allows procuring entities to predict the effects of adding or removing resources from a specific deployment. That is, by changing the parameters of a well-fit workload model, IT professionals can predict how many resources will be needed in a specific private cloud if and when workload changes in the future. Moreover, because the models take the form of statistical distributions, it is possible to predict load characteristics such as the degree to which load will vary, percentiles of the overall load distribution, *etc.* This research also informs, more broadly, future system software research, design, and engineering, since the observed workloads differ substantially in their statistical characterization from workloads reported previously [8], [10].

Public cloud providers have yet to publish quantitative traces of workload data, as the exact nature of customer usage and infrastructure capabilities are usually held to be trade secrets. Private clouds, however, are not typically revenue generating in a commercial context, making the companies that deploy them more willing to share (in anonymized form) their cloud workload instrumentation data.

We are fortunate to have been able to work with Eucalyptus [12], [13], a popular open-source private cloud platform maintained by the commercial entity Eucalyptus Systems, to study several production private cloud deployments implemented by commercial customers. As far as we know, this article constitutes the first analysis of instrumentation data captured from enterprise private clouds used for production (*i.e.*, not for evaluation purposes). To help foster additional

Rich Wolski is with the Department of Computer Science, University of California Santa Barbara, Santa Barbara, CA, 93106 USA e-mail: rich@cs.ucsb.edu

John Brevik is with the Department of Mathematics, California State University at Long Beach, Long Beach, CA, 90840 USA e-mail: jbevik@csulb.edu.

This work was supported in part by The National Science Foundation grant number 0751315 and by Eucalyptus Systems Inc.

research, we are making the data described in this article available to the wider community from the URL given in [14] [1].

The remainder of this article is organized as follows. Section II describes the data sets we have gathered and the methodology we have used to model them. In Section III we detail the accuracy of these new models and discuss their strengths and weaknesses. Section IV outlines the potential impact that these results have on private cloud usage and systems design. We discuss related work in Section V, and in Section VI we conclude and describe future work.

II. Private Cloud Workloads

Our investigation focuses on private cloud workloads from clouds that implement Infrastructure as a Service (IaaS). In this style of cloud computing, users request that the cloud provision collections of virtual machines (VMs) that are interconnected (via an isolated network) with each other, a common (and secure) storage infrastructure, and with a gateway or network transition point to at least one network outside of the cloud itself. Eucalyptus supports the Amazon AWS API and cloud abstractions [15] for IaaS.

Briefly, under the AWS API, each user request for VM provisioning specifies:

- **VM Type** – the core count, memory size, and disk partition size for each virtual machine to be provisioned;
- **Image** – the data set (captured as a root filesystem, kernel, and ramdisk) that will be used to initialize the VM when it starts;
- **Security Group** – an identifier of a virtual network interconnecting the VMs protected by a virtual firewall;
- **Availability Zone** – a logical partition of the cloud in which all resources share a common set of QoS attributes[2]; and
- **Instance Count** – the number of VMs that should be started from the Image, in the Security Group, in the Availability Zone as a result of the request.

Note that in this API specification, all VMs that result from a single request share the same Security Group, Availability Zone, and VM Type and all are started from the same Image. Users query the cloud for identifiers that name these components of a provisioning request and some, but not all, may instantiated from administrator-set defaults.

Eucalyptus, like AWS, either accepts an entire request or rejects it as a whole. In particular, if there are insufficient resources available to start all VMs in the same Availability Zone, the entire request is rejected and no VM will start. If, on the other hand, the request is accepted, Eucalyptus will attempt to start all VMs specified in a request simultaneously. Users must request the termination of the VMs individually, however, based on instance identifiers that are returned from the provisioning request. To effect the termination of a group of VMs, one may specify multiple instance identifiers within a single terminate request.

[1]We will make the data sets described herein available, in anonymized form, should this article be accepted for publication.

[2]In AWS, an availability zone corresponds to a region of fault isolation. Eucalyptus extends this concept to include other QoS attributes.

A Eucalyptus-internal VM scheduler (Eucalyptus can be configured to use either a round-robin or a high multi-tenancy scheduler) makes placement decisions and keeps track of utilization across physical machines. Both Eucalyptus schedulers place restrictions both on how VM types are defined and how VMs may be scheduled. In particular, VM types must be defined in a size order so that their core counts and memory sizes “nest.” That is, one or more VM types of a given size must “fit” within the size specification of the next larger size. Ideally, each VM type comprises a multiple of the next smaller VM type, although only the nesting criterion is enforced by Eucalyptus. Further, the resource requirements associated with a VM Type cannot be split across physical machines. Thus the scheduler will only place a VM on a machine where there are a sufficient number of free cores, sufficient free memory, and sufficient free disk space to accept its VM Type. Finally, the Eucalyptus administrator specifies a storage quota for each physical machine that is configured to accept VMs that limits the total disk space Eucalyptus is entitled to occupy. The schedulers respect these quotas when making placement decisions at run time.

In this study, the different cloud administrators had configured their respective clouds so that placement decisions depend only on core count. That is, each VM type and storage quota were defined so that core count (and not memory or disk storage) fit is the critical criterion for making placement decisions. If the scheduler determines that there are a sufficient number of free cores on a physical machine to run a VM, the configuration in each case guarantees that there is enough memory and disk space as well. This method of VM sizing is typical of Eucalyptus private clouds. Administrators typically ensure that each VM Type balances core, memory, and disk footprint to make it easier for users to reason about their resource usage.

A. VM Attributes

Thus, VM workload, as experienced by a Eucalyptus cloud, can be captured by four attributes (for each given request):

- **Interarrival Time** – the amount of time until the next request;
- **VM Lifetime** – the time duration over which a VM is provisioned to a physical machine;
- **Request Size** – the number of VMs in the request;
- **Core Count** – the CPU core count requested for each VM.

Our approach is to model workload for a specific cloud as a set of possibly interrelated statistical distributions of these attributes. In Subsection II-E we describe the specific models we have found to be effective for this purpose and a methodology for automatically determining their parameters from a Eucalyptus workload trace. Note that the data described in this study do not include instrumentation data gathered from within each VM. Each Eucalyptus administrator installed and maintained his or her own set of images without regard for this work, making it difficult or impossible for us to ensure that a common set of VM instrumentation tools could be employed. Thus our work focuses on occupancy workload as managed by the cloud (*i.e.*, the workload that the cloud unavoidably must manage without the cooperation of the VMs), complementing the work of others who have studied models that rely on data gathered from within the VMs [16].

B. Resource Attributes

According to this model, machines configured into a Eucalyptus system experience workload in terms of two types of utilization:

- **Core Utilization** – the fraction of the total core time available that CPU cores within the cloud are assigned to VMs; and
- **Node Utilization** – the fraction of the total machine time available within the cloud that nodes (physical machines) are assigned to *some* VM.

Henceforth we will use the term "node" as a synonym for "physical machine" for the sake of brevity. Core utilization captures the extent to which the cloud makes use of the CPU cores. Because cores, memory, and disk storage cannot be used independently, core utilization can also provide some insight into memory and disk usage, albeit indirectly. Clearly core utilization is dependent on VM lifetime and interarrival distributions while node utilization reflects the balance of workload across machines as well as VM lifetime and VM interarrival time. That is, if node utilization is low, it is either because the internal scheduler is not attempting to maximize balance (Eucalyptus includes a "greedy" scheduler that tries to maximize multi-tenancy and to keep nodes idle) or because there is insufficient work arriving to keep nodes busy given the lifetime distribution of the VMs.

C. Instrumentation

At the time of this study, Eucalyptus did not include instrumentation that directly measures both VM workload attributes and system-wide performance [3]. However, the default Eucalyptus configuration deployed at most sites specifies that the system capture maintenance logs in a "rolling" set of log files. To save disk space, only the latest 200 megabytes of log data, but not a long-term history, are maintained so that in the event of a problem, the recent state of the system can be interrogated.

Note also that the information stored in these logs is designed to illuminate internal system state rather than to track the trajectory of specific requests. In addition, the logs do not capture events such as node failure, the restart of an internal system component, or overall system restart. Moreover, Eucalyptus has been designed so that it can operate in a highly available mode. As a result, many of the system components are functional and stateless and thus do not log either session-oriented events (*e.g.*, VM lifetime) or system outage. For these reasons, both cloud workload and system utilization must be composed from different logging events rather than measured directly.

[3]Since the time of this writing, some instrumentation has been added to Eucalyptus for accounting purposes, but the accounting features can be configured optionally. Thus this facility in not in use at all installations.

Measuring VM Attributes: Each VM, when it is instantiated, is given a unique identifier (called an "instance id") that will not be reused for an acceptably long period of time after the VM terminates. Eucalyptus attempts to log both the arrival of a request to start a VM and any request to terminate it, but these logging events are "best effort." That is, the starting event and/or stopping event specific to a particular VM may not be present in the logs. In particular, while the logs fill and "roll over" (the log file names are kept ordered from most recent to least and each has a finite size so their names must be adjusted when the most recent log becomes full) logging events may be lost.

Logging events are also lost if and when the local site administrator decides to stop capturing consecutive logs (or, more likely, forgets to re-enable a capture script after a routine machine reboot). Because only a finite history of logging data is captured in the default configuration, events can be lost if the history is not archived periodically.

Finally, Eucalyptus is designed to continue to function in the presence of internal software failure (although in a degraded mode). If, for example, the software agent that runs on a node (which is responsible for starting and stopping VMs) crashes or is stopped, the VMs it has started continue to function. When the component, called a Eucalyptus Node Controller (NC), is restarted, it will "adopt" the VMs it finds running and continue as if the outage had not occurred. During the period that the NC is down, however, no logging information for that node is captured.

Fortunately, each node reports the VMs that it is hosting approximately every 15 seconds and these status updates are logged. In cases where the start event or stop event (or both) are missing, the time span between the first observed status update for a VM and the last are a close approximation to its user-controlled lifetime. Because only the first VM "heartbeat" and the last are necessary to approximate lifetime, logging outages that are due to NC stoppage that occur while the VM is running can be ignored when computing VM lifetime.

Longer-term outages that are due strictly to failure to archive the rolling logs are more problematic. If, for example, the local administrator fails to archive the logs over a period of time during which the system is running without outage, the lost logs will appear as a period of dropout in an overall trace of VM workload. VMs that start in the dropout period will appear to start immediately after the dropout ends (the logs will suddenly show a heartbeat for a previously unlogged VM). Similarly, VMs that end during a dropout period will appear to terminate before dropout, at the time of their last observed heartbeat.

To mitigate these effects, our methodology attempts to identify places in the logs where log dropout appears to occur. A large number of VM heartbeats (without start events) that appear simultaneously after a long period of inactivity marks the end of a dropout period. Similarly, a large number of final VM heartbeats (again, without stop events) at a single point in the log signals the beginning of a dropout period.

To correct for the artificial foreshortening of VM lifetime and elongation of VM interarrival time that log drop out introduces, we generate an "outage map" for each archive and use it to adjust affected VM start and stop times. Each entry in a map indicates the starting time stamp for a potential outage and its duration. For VMs that appear to start during an outage, we decrement their start times by a random fraction of the outage interval. Similarly, VMs that appear to have stopped during a dropout period have their stop times incremented by a random fraction of the outage interval.

We observe that this outage correction (the degree of which varies by installation) improves the model fit for VM interarrival time, VM lifetime, and CPU core utilization. Because it does not capture individual node outages, however, it does not completely correct node utilization predictions (*cf.* the next Subsection).

Resource Attributes: Measuring resource attributes over the long term is more problematic under the versions of Eucalyptus we studied because resource information is not aggregated into a single log (as are VM event data). Eucalyptus maintains several different rolling logs, each recording information that is specific to a subsystem or activity. Even with the aid of the cooperative and helpful cloud administrators who worked with us on this study, we observed log dropout on several occasions for an archive of only a single rolling log (the one that carries VM lifetime and scheduling information.) It was not possible to tax these administrators with the additional burden of archiving *all* of the different logs recorded by Eucalyptus in a given installation so that we could generate aggregate measurements of resource attributes.

Instead, we use a faster-than-real time simulator that implements the Eucalyptus VM scheduling algorithms and "replays" VM activity culled from the logs. The advantage of using a simulator is twofold. First, it is possible to instrument the simulator to provide information that explains a particular set of observations. Simply computing resource utilization, for example, from the logs provides little insight regarding the patterns of activity that leads to these measurements. Second, it is possible to predict the effect of hypothetical changes to the system using the simulator. For example, it is possible to change the scheduling algorithm and to observe the effect of the change on node utilization.

D. Data Sets

We present data from three separate commercial private clouds implemented using Eucalyptus. The commercial entities operating these private clouds have allowed us to monitor their respective installations over an extended period and have agreed to have these results data made publicly available in an anonymized form. All three clouds support the commercial activity of their operators (they are not operated for, *e.g.*, evaluation or investigative purposes).

The first data set (DS1) is taken from an organization with several large-scale software development efforts. While the private cloud is used for some company-wide service hosting, its primary use is to support software testing and development. DS1 captures private cloud VM activity that combines software development with service hosting, with an emphasis on development.

The second data set (DS2) is taken from an IT organization that "sells" time on a re-charge basis to other organizational units in its umbrella company. The accounting charges translate to operating budget for the following fiscal year, making the economic incentives similar to those driving a public cloud. Thus the usage of this cloud is not known (*i.e.*, the cloud does not have a specific purpose other than to host the workloads of its paying customers). The function of the umbrella company, however, makes it likely that much of the activity is generated by software development.

The third data set (DS3) is taken from a private cloud used to allow business partners to integrate their respective software products with the products made by the company operating the cloud. It also supports user and customer trials of the company's software products. Finally, these partners often use the cloud for demonstration or sales purposes. Thus the workload is a mixture of software development with on-demand hosting activities.

Table I provides summary descriptions of the cloud deployments from which we have gathered these data sets.

Each data set measures the workload from a single Eucalyptus Availability Zone. The cloud that generated DS1 was configured with two Availability Zones, only one of which was instrumented for monitoring. The other two clouds each implemented only a single Availability Zone. We show the time period that each dataset spans and the approximate employee counts for the organization operating each cloud. Our goal in using these data sets is to measure workload across a spectrum of commercial activity. However, note that each cloud is relatively small and is operated by a small IT staff. Also, we do not have measurements of the size of the user pool accessing each cloud. Thus the organizational scale does not necessarily reflect the size of the user community that generated the workload captured in each data set.

All three data sets span several months of continuous usage. During the monitoring periods, each of the hosting organizations upgraded their respective Eucalyptus clouds, in one case multiple times. Further details concerning the provenance of the data and the data sets themselves are available from the web site URL given in [14], hosted by the Computer Science Department at the University of California Santa Barbara.

E. Modeling Methodology

We represent the workload experienced by each cloud as a function of four distributions, one for each of the workload attributes: *Request Interarrival Time*, *VM Lifetime*, *Request Size*, and *Core Count*. Thus, the workload model for a given cloud is represented by a time series of requests, each composed of one or more VMs with each VM characterized by a core count and lifetime. All VMs in a request have the same core count (per AWS API semantics), but request sizes and core counts vary from request to request. Further, according to the API, while all VMs within a specific request are intended to start at the same time, it is possible for the user to terminate them either individually or as a collection. In this work we model VM lifetimes according to the latter usage pattern. That is, all VMs within a request are assumed to have the same lifetime. Anecdotally, we have experimented with generating workloads using lifetime models to determine the durations of

Data Set	Nodes	Cores/Node	Time Period	Description
DS1	13	24	Aug. 2012 to Oct. 2012	Large company with 50,000 to 100,000 employees
DS2	7	12	Aug. 2012 to Apr. 2013	Medium sized company with 2,000 to 5,000 employees
DS3	7	8	Aug. 2012 to May 2013	Small company with 50 to 100 employees

TABLE I
SUMMARY OF PRIVATE CLOUD DATASET CHARACTERISTICS

VMs within each request separately and found that it does not affect the results significantly.

We have found that a 3-*phase hyperexponential* distribution, together with some enhancements to be discussed below, provides a good model for both VM interarrival and lifetimes. Note that while hyperexponentials could be justified from an "explanatory" standpoint for VM lifetimes (*i.e.*, one could posit that there are three types of processes, namely short, medium, and long), there is no such simple explanatory justification in the case of interarrivals, and in any event we simply put this model forth as one able to capture the complexities of both distributions. The hyperexponential family of distributions is also convenient from the standpoint of parameter estimation via the E-M algorithm as implemented in the EMpht software package [17], [18].

The other two attributes are categorical, and we simply use their empirical proportions for parameters. Eucalyptus supports up to five VM types, making the number of request-size and core-count combinations relatively manageable. To generate a simulated workload trace, then, we extract these proportions for each size-count combination and sample randomly from the empirical distribution.

In addition, the workloads include modal behavior. Processes scheduled at regular intervals and large numbers of processes with identical lifetimes introduce jump-discontinuity-like behavior in the cumulative distribution functions of the interarrival and VM lifetime distributions respectively. We therefore implemented an algorithm for automatically identifying such modal behavior in the traces. We then generate our final model by separating these instances out as a fraction of the total, producing a hyperexponential fit for the remaining processes, and overlaying the distributions of the periodic and modal processes (*cf.* Section III).

III. RESULTS

The goal of our work is twofold. First, we wish to illustrate statistical distributions that are useful in modeling VM interarrival times and lifetimes. Second, we wish to demonstrate the use of these individual distributions to compose realistic models of overall cloud performance (*e.g.*, cloud utilization).

To investigate the question of representing VM attributes using statistical distributions, we compare a 3-phase hyperexponential (estimating parameters via the E-M algorithm discussed in the previous section), a lognormal distribution, and a Pareto distribution (for the last two there are closed forms for maximum likelihood parameter estimators) in terms of their ability to represent interarrival time and VM lifetime. During the course of this investigation we noticed that the distribution of VM lifetime data differed depending on whether the VMs being modeled use one core or more than one core; thus, we use two lifetime distributions for each data set.

In the following tables, we show the Kolmolgrov-Smirnov (KS) statistic [19] computed for the goodness-of-fit between the observed data and a distribution fit to it. Recall that the KS statistic measures the difference in quantile between a sample and a proposed model over all possible values and returns the maximum of these; for example, if 100 seconds is the 20^{th} percentile of true VM lifetimes and a proposed model has 100 seconds at the 35^{th} percentile, then the difference for the value of 100 is $.35 - .20 = .15$, and so the KS statistic for this model will be at least $.15$. Smaller values of this statistic thus reflect better fit, at least with respect to this test. We also include, simply to give a point of reference, the $\alpha = .05$ value for the statistic. Note that a given model might have a fairly small KS statistic and still have a large discrepancy in terms of population attributes such as mean or variance (which may be crucial depending on one's interest), because a small fraction of very large values can have a great influence on these statistics – graphically, the KS statistic detects *vertical* but not *horizontal* discrepancies in the CDFs. On the other hand, generally speaking, a large KS statistic implies a poor fit.

Table II compares KS statistics for the interarrival times. For this attribute, every model generates a KS statistic large enough to warrant a rejection of the null hypothesis at the $\alpha = .05$ level that the data have been drawn from the model distribution for the data in DS1 and DS2. However, both the hyperexponential and the lognormal are reasonably close to the critical value in each case. For DS3, both the hyperexponential and lognormal KS values are less than the critical value, which indicates that either or both might be a good choice for modeling interarrival times in this data set. Compared to the hyperexponential and lognormal models, the Pareto appears to be a poor choice, as evidenced by the large KS statistics it generates.

Data Set	Critical Value	3-P	Lognormal	Pareto
DS1	0.040227	0.061424	0.076046	0.235342
DS2	0.062933	0.076903	0.079632	0.196724
DS3	0.033230	0.020051	0.023726	0.365086

TABLE II
COMPARISON OF KS STATS FOR INTERARRIVAL DATA AND MODELS

Figure 1 shows the cumulative distribution functions (CDFs) for the DS3 measurement interarrival data and all three models (the units in the figure are seconds). Note that for the DS3 data set, the KS statistics show the hyperexponential and

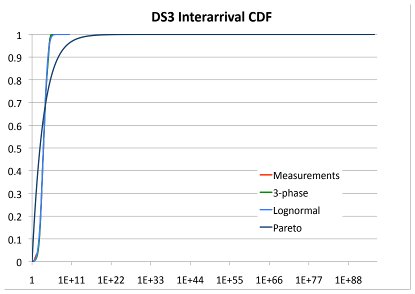

Fig. 1. DS3 interarrival distributions (units are seconds)

lognormal models to be much better fits than the Pareto and each comfortably below the rejection threshold. In the figure, the measurement data are completely obscured by both models on the scale necessary to show the upper tail of the Pareto. It is clear in this graphical comparison that the Pareto is a poorer fit than either of the other two alternatives.

Figure 2 shows the same comparison for the DS3 data set as in Figure 1 but with the Pareto model removed. Both the hy-

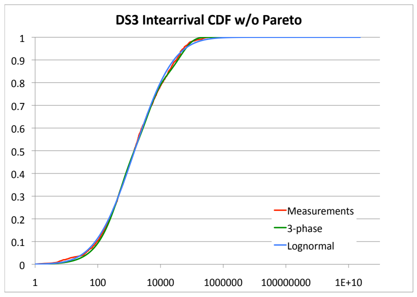

Fig. 2. DS3 interarrival data, hyperexponential and lognormal distributions (units are seconds)

perexponential and lognormal models appear to represent the shape of the measurement distribution correctly in the figure. However, the upper tail of the lognormal contains considerably greater probability mass than either the measurement or the hyperexponential distribution. Figure 3 "zooms in" on just the parts of the CDFs between 50000 and 5000000 seconds for the interarrival times in data set DS3. Note that the x-axis in each of Figure 2 and Figure 3 is defined using a log scale and that y-axis in Figure 3 shows only values between 0.93 and 1.0. As described in the previous section, the KS statistic fails to detect a poor fit in the extreme percentiles, which exert high leverage on population characteristics such as mean and variance.

To investigate further the modeling power of each approach, Table III compares, by data set, the sample mean and standard deviation for the interarrival observations to the mean and standard deviation from each fitted model. In each column of data, the units of time for the mean is seconds and the standard

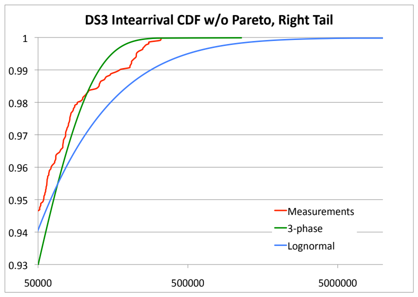

Fig. 3. DS3 interarrival data, hyperexponential and lognormal distributions between 50000 and 5000000 seconds

deviation is given in parentheses. Note that we omit the Pareto

Data Set	Sample	3-P	Lognormal
DS1	2202.1	2202.1	2278.5
	(2.2e+04)	(8.5e+03)	(5.0e+04)
DS2	41285.7	41285.9	265295.6
	(1.1e+05)	(1.0e+05)	(5.2e+07)
DS3	11238.8	11238.8	18792.7
	(3.0e+04)	(2.8e+04)	(2.4e+05)

TABLE III
COMPARISON OF MEAN AND STANDARD DEVIATION (IN PARENTHESES) FOR INTERARRIVAL DATA AND MODELS

results from this table because the the shape parameter for each fitted Pareto model is less than 1, so that the mean is infinite and the variance is undefined for these models. From a comparison of the remaining means and standard deviations, it is clear that 3-phase hyperexponential is a better choice than a lognormal as a model for the interarrival data across all data sets. In particular, as Figure 2 suggests, due to the additional probability mass in the upper tail of the lognormal its mean is larger than the mean of either the empirical data or the hyperexponential.

Table IV and Table V show KS statistics and mean and standard deviation comparisons for for multi-core VM lifetimes. In each column of Table V, the units of time for the mean is seconds and the standard deviation is shown in parentheses. Again, the KS comparisons do not show a clearly superior

Data Set	Critical Value	3-P	Lognormal	Pareto
DS1	0.055429	0.041428	0.096728	0.310270
DS2	0.071283	0.087782	0.050345	0.278906
DS3	0.053344	0.066853	0.026220	0.296969

TABLE IV
COMPARISON OF KS STATS FOR MULTI-CORE VM LIFETIMES AND MODELS

model, although they indicate that the Pareto distribution is likely to be a poor model. The lognormal fit fails to reject the null hypothesis for DS2 and DS3 where the hyperexponential statistic is just slightly larger than the critical value. The reverse is true (fail to reject for the hyperexponential, reject for the lognormal) for DS1.

However, comparing the means and standard deviations in Table V shows that the hyperexponential is the closest fit across all data sets and models, again due to a better match in the upper tails. The maximum-likelihood fits of the Pareto models generate distributions that have infinite means and undefined variances in each case, so they have been omitted from the table.

Data Set	Sample	3-P	Lognormal
DS1	257173.0 (4.6e+05)	257176.1 (5.7e+05)	1615134.1 (1.5e+08)
DS2	144669.0 (7.9e+05)	144669.0 (4.0e+05)	257314.2 (1.5e+07)
DS3	30739.2 (1.6e+05)	30739.3 (1.0e+05)	48756.6 (2.2e+06)

TABLE V
COMPARISON OF MEAN AND STANDARD DEVIATION (IN PARENTHESES) FOR MULTI-CORE VM LIFETIMES AND MODELS

Data Set	Critical Value	3-P	Lognormal	Pareto
DS1	0.035642	0.130662	0.206753	0.349283
DS2	0.089676	0.085053	0.108434	0.268853
DS3	0.053508	0.101547	0.071394	0.301746

TABLE VI
COMPARISON OF KS STATS FOR SINGLE-CORE LIFETIMES AND MODELS

Similarly, Table VI and Table VII show KS and mean/standard deviation statistics for single-core VM lifetimes. For this attribute, the KS statistics indicate slightly worse fits for the hyperexponential and lognormal models than for the other two attributes, and again there is little to choose between the two. However, the mean and standard deviation comparisons (with Pareto omitted for the same reason as before) show the hyperexponential to be the better choice, particularly with respect to the mean.

Data Set	Sample	3-P	Lognormal
DS1	28754.4 (1.6e+05)	28754.8 (1.3e+05)	4837.5 (2.4e+04)
DS2	599815.0 (1.7e+06)	599843.7 (1.6e+06)	2712876.3 (9.8e+08)
DS3	44447.8 (2.2e+05)	44448.5 (1.7e+05)	40081.3 (2.6e+06)

TABLE VII
COMPARISON OF MEAN AND STANDARD DEVIATION (IN PARENTHESES) FOR SINGLE-CORE VM LIFETIMES AND MODELS

Table VIII shows the parameters computed by EMpht for each hyperexponential model. In the table, the probability p_i is associated with the exponential distribution having parameter λ_i in each model; since p_3 can be calculated as $1-p_1-p_2$, we omit it. Note that the model of single-core VM lifetimes for data set DS1 is, in fact, a 2-phase hyperexponential (λ_2 is equal to λ_3). This case raises the question of whether additional parameters (*i.e.* more hyperexponential phases) would improve the fit. We further investigate this question in Subsection III-B.

A. Synthetic VM Traces

As described in Section II, accurate synthetic VM traces may need to combine fitted models with modal components. Figure 4 shows the CDF of the VM interarrival times taken from the measurement data. Note the presence of what appear

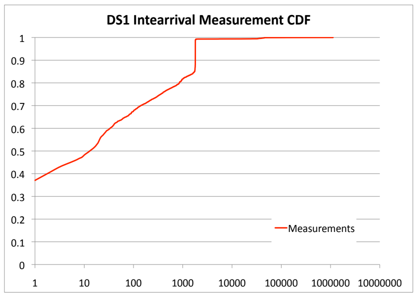

Fig. 4. DS1 interarrival measurement data (units are seconds)

to be two *modes*, one at zero seconds and the other at 1800 seconds. Less obviously from the graph, there are many sequences of VM launches at 1800-second intervals throughout the trace (the mode appears there because there is often no VM activity between two consecutive such launches). The mode at zero seconds results from two different phenomena. First, multiple VM launches specified in the same user request are logged by Eucalyptus as separate events with the same time stamp. Second, heavy periods of activity result in multiple requests that appear to occur simultaneously. Eucalyptus logs events to the nearest second and requests are conveyed to the place in the system where the logs are recored via a set of messages that are handled asynchronously. Thus requests are queue up and appear to be handled simultaneously when, in fact, they were issued by their respective users at slightly different times.

In order to build a realistic synthetic VM trace, one must take into account these modal components, which by their nature are not well modeled by a continuous distribution such as a hyperexponential, in the measurements. In this study, only the interarrival measurements for data set DS1 contain regularly spaced sequences. On the other hand, interarrival times from all datasets contain a mode at zero seconds (due to to the Eucalyptus logging mechanisms); additionally, the multi-core VM lifetime measurements for DS1 contain four further modes at 2493, 929, 992, and 1104 seconds respectively. As described previously, our methodology automatically detects these components and incorporates them into the synthetic traces it produces.

B. Modeling Cloud Performance

Both to compare the efficacy of the different attribute models and to test whether additional phases in the hyperexponential models would improve accuracy, Table IX compares the core utilization measured for each private cloud against synthetic workloads. In each case, if the measurement time series contains modes, the synthetic representation of that series includes similar modes. CPU core utilization, in this case, is computed as the ratio of total CPU core occupancy time to the total core time available. Note that each synthetic trace is composed of multiple samples from its constituent distributions. Thus the core utilization measured from the synthetic data varies trace to trace. In columns 3,4, and

Data Set	Interarrival Time	Multi-core Lifetime	Single-core Lifetime
DS1	λ_1=0.00080 λ_2=0.00005 λ_3=0.02894 p_1=0.34561 p_2=0.08648	λ_1=0.00004 λ_2=0.000002 λ_3=0.00059 p_1=0.24667 p_2=0.37948	λ_1=0.000003 λ_2=0.00109 λ_3=0.00109 p_1=0.09325 p_2=0.22251
DS2	λ_1=0.000006 λ_2=0.05228 λ_3=0.00081 p_1=0.38881 p_2=0.18227	λ_1=0.00186 λ_2=0.00008 λ_3=0.0000008 p_1=0.42093 p_2=0.43960	λ_1=0.00143 λ_2=0.00005 λ_3=0.0000004 p_1=0.44885 p_2=0.30675
DS3	λ_1=0.00030 λ_2=0.00003 λ_3=0.00257 p_1=0.39442 p_2=0.24644	λ_1=0.00498 λ_2=0.000005 λ_3=0.00022 p_1=0.37621 p_2=0.14838	λ_1=0.000297 λ_2=0.000003 λ_3=0.00410 p_1=0.34131 p_2=0.12544

TABLE VIII
HYPEREXPONENTIAL PARAMETERS COMPUTED FOR EACH MODEL MY EMPHT

Data Set	Measured	3-P	Lognormal	Pareto
DS1	0.41	0.39 (0.04)	0.46 (0.09)	0.03 (0.03)
DS2	0.22	0.22 (0.03)	0.07 (0.04)	0.06 (0.06)
DS3	0.13	0.14 (0.01)	0.08 (0.02)	0.01 (0.01)

TABLE IX
COMPARISON OF MEASURED AND MODELED CPU CORE UTILIZATION USING SYNTHETIC WORKLOADS

5, Table IX shows the average and standard deviation (in parentheses) of each synthetic utilization over 100 repeated experiments. In each case, the average from a synthetic trace generated using a 3-phase hyperexponential is well within two standard deviations of the true measurements (shown in column 2) for each data set and is the closest to the true measurement among the three models tested. The inaccuracy of the lognormal and Pareto models should not be surprising in light of the lognormal's failure to reflect the mean (and hence aggregate) process lifetime (*cf.* Tables V, VII) and the Pareto's failure even to have a finite mean. While it is possible that additional hyperexponential phases would improve accuracy, the improvements (in terms of modeling overall CPU utilization) would be slight and the extra complexity would be unwarranted. Table X shows measured and modeled node utilization. As in Table IX, the sample mean and standard

Data Set	Sample	3-P	Lognormal	Pareto
DS1	0.82	0.92 (0.02)	0.86 (0.08)	0.20 (0.20)
DS2	0.39	0.37 (0.05)	0.21 (0.10)	0.17 (0.17)
DS3	0.41	0.47 (0.03)	0.30 (0.11)	0.05 (0.05)

TABLE X
COMPARISON OF MEASURED AND MODELED NODE UTILIZATION USING SYNTHETIC WORKLOADS

deviation (shown in parentheses) for 100 repeated modeling experiments are shown in columns 3,4, and 5. To compute node utilization for the cloud we first record the occupancy duration for each node as total time during which it hosts at least 1 VM. The node utilization for the cloud is then the the ratio of the sum of the node occupancies for each node divided by total possible occupancy time.

$$\mathrm{NodeUtilization} = \frac{\sum_{i=1}^{N} \mathrm{NodeOccupancy}(i)}{N * \mathrm{TraceDuration}} \quad (1)$$

In Equation 1, $\mathrm{NodeOccupancy}(i)$ returns the total time that node i had work assigned to it by the Eucalyptus scheduler, $\mathrm{TraceDuration}$ is the total time period under study, and N is the number of nodes.

Table X shows that synthetic workloads generated from a 3-phase hyperexponential are generally more accurate than those generated from a lognormal or Pareto, but the accuracy is not as precise as in Table IX. These results illustrate Eucalyptus system behavior we chose not to model. In particular, we discovered two phenomena in the logs that are not represented in our models and which we believe contribute to the discrepancies between modeled and measured node utilization.

The first unmodeled characteristic is individual node dropout. Eucalyptus is engineered to avoid "fail-stop" in the event that one or more of its internal components fail. In particular, the temporary loss of a node manifests only in the temporary loss of capacity. Inspecting the logs, we noticed periods of time when individual nodes appeared inactive. The reasons for these inactivity periods (e.g. node failure, node maintenance, etc.) are not apparent in the logs nor were they frequent enough for us to attempt a statistical characterization.

We use the simulator to compute measured node utilization by replaying the actual measurement trace from the logs and observing the values of $\mathrm{NodeOccupancy}$ for each node. Thus the resulting values computed from Equation 1 include periods of individual node inactivity. In contrast, the modeled data comes from synthetic traces that are generated without individual node inactivity periods. Hence the 3-phase hyperexponential model is likely to be overstating the measured utilization in proportion to the duration of individual node inactivity periods.

The second complication arises from bug in the VM schedulers present in *some* versions of Eucalyptus. The faster-than-real-time simulator includes the Eucalyptus schedulers (both round-robin and "greedy"), but it invokes them as part of a single-threaded event-driven simulation. When Eucalyptus runs, however, the schedulers are invoked asynchronously in a way that appears occasionally to bring about a race condition. As a result, the logs record periods during which the schedulers temporarily assign more than the maximum specified occupancy to individual nodes. That is, a node is assigned VMs in such a way that the total number of cores in the VM exceeds the core capacity of the node. The hypervisors that were configured into each Eucalyptus cloud simply time-sliced these VMs so that the cloud appeared to function properly in the presence of this erroneous implementation of multi-tenancy. Indeed, neither the Eucalyptus engineering staff nor the individual cloud administrators were aware of this bug before our study. Moreover, it is not clear whether the bug is present or, if present, whether it is triggered with the same frequency, across versions of Eucalyptus. In this study, each of the clouds went through one or more Eucalyptus upgrades over the course of the logged time period.

As with individual node dropout, we chose not to model the manifestation of the scheduling race condition in the synthetic traces or in the simulator. We believe that these unmodeled effects contribute to the inaccuracies show in Table X for the 3-phase hyperexponential model. The other models, however, suffer from the additional probability mass in the tails discussed earlier. That is, they generate long-running VMs with greater frequency and longer lifetimes than what the logs report. As a result, near the end of each lognormal and Pareto simulation experiment there are a few of these extreme jobs, which keep a small number of VMs assigned while the others are idle. This explains the relatively low simulated node utilization for the lognormal and Pareto models.

IV. Discussion

From the data presented in Section III, it seems that private cloud workloads culled from enterprise settings (as they are being generated today) can be well modeled by a 3-phase hyperexponential distribution that is fit using the E-M algorithm. From a modeling perspective, this result is perhaps unsurprising in that the 5 parameters of this model should be able to represent the observations more realistically than the 2 parameters that characterize both the lognormal and Pareto models. Indeed, it is striking that the lognormal (which requires less computational effort to compute that the hyperexponential) is as effective as the results indicate. In a modeling setting where a quick approximation is needed, the lognormal could prove to be a useful alternative to the hyperexponential. As the results in the previous section show, however, the lognormal model is inaccurate with respect to mean interarrivals and process lifetimes, which are crucial to an accurate representation of workload and utilization.

From an explanatory perspective, these results indicate potentially greater long-term impact. Private clouds do not implement queuing disciplines in the same way that batch-scheduled systems do. The results seem to indicate that workload can be modeled as small, medium, and large-sized VMs arriving in small, medium, and large-sized intervals. Private clouds must schedule the offered load using multi-tenancy (*i.e.* by overcommitting resources) or by choosing to refuse admittance to certain requests when a resource shortfall is imminent. Accurate models of workload, such as the hyperexponentials detailed in this study, inform the design of cloud schedulers and capacity planners. Indeed, our implementation is fully automatic, making it possible to generate predictive models "on the fly" to be used by a run time cloud scheduler (this usage is the subject of our future work).

These models also allow IT professionals faced with medium- to long-term capacity planning decisions to answer "what if" questions. For example, the core utilization measurements shown in column 2 of Table IX indicate approximately 59% of the core occupancy time for DS1 is unused. The cloud administrators for this cloud, however, report that it is "full" with little extra capacity. The reason for this discrepancy is that the largest VM type configured for this cloud specifies half the number of cores as is available from any given node. Thus, using round-robin scheduling as this cloud does, when the core utilization is above 0.5 it is unlikely that a VM of the largest type can be scheduled. The overall utilization fraction of 0.41 shown in the table is near enough to the 0.5 threshold so that the system "feels" as though it it is running near capacity with respect to large VM sizes, but users are still able to run them when needed. Using the simulator to replay the original log data, we have verified this conjecture. For 99% of the log duration, at least 12 cores (half the capacity of a single node) are available.

By changing the parameters of the hyperexponential model, it is possible to estimate what the effect on core utilization if all VM lifetimes were to, say, double due to the need to complete additional work. To model this change, each λ in the lifetime models shown in Table VIII (columns 3, and 4) are multiplied by 0.5. In this example, such an increase in lifetime would result in an overall average utilization of 0.68 indicating that the system would appear overcommitted to its users periodically with respect to large VMs. Again, using the simulator to replay a synthetic workload generated from this altered model, we determine that users would not be able to run a 12 core VM for approximately 21% of the simulated time duration. Thus, without additional nodes added to the cloud, a doubling of VM duration would cause the cloud to refuse to run 12 core VMs approximately 21% of the time. Moreover, by changing the parameters of the simulator, we determine that the addition of 3 nodes, each having 24 cores (as do the others), will restore the user experience to the 99% availability level for large VMs if the workload were to double.

In addition to the capacity planning capability, we believe that these results provide early guidance to those designing applications, middleware, and operating systems for enterprise private clouds. Specifically, excellent earlier work [8], [10] has shown that various process process lifetimes are "heavy tailed" and in fact are well modeled by Pareto distributions. The virtual machines in this study all instantiated either an instance of the Linux operating system or the Windows operating system, which, presumably, has been designed specifically to support heavy-tailed process lifetime distributions. However, the operating system instances terminate with their associated (light-tailed) VM instances, and so the lifetimes of their subprocesses are not long enough to necessitate these design features. That is, in an enterprise private cloud, operating system support in the guest instances may only need to manage processes with relatively light-tailed lifetime distributions. The operating system *implementing* the private cloud probably does see heavy-tailed process lifetimes, but it seems that the guests do not. We have yet to investigate the ramifications of this observation fully, but it indicates that a simpler and less general operating system (still supporting the Linux or Windows system call interface) can simplify private cloud deployments and optimize performance. It also supports the notion that private cloud applications treat guest operating systems as software containers, the lifetime of which is determined by the lifetime of the application itself rather than as a software extension of the physical infrastructure.

Finally, we wish to acknowledge that this first study of *in vivo* private cloud workloads may only be capturing early usage patterns. Private clouds are new and even newer in production enterprise computing settings. As the approach matures, studies such as this one will certainly be necessary either to confirm its findings or to demonstrate the effect on workload that the maturation process has had. However, our results provide guidance today to the enterprise professional and to the computer systems researcher wishing to model private cloud workloads.

V. Related Work

Workload modeling has been studied extensively in the conjunction with parallel job scheduling and high-performance computing [20], [21], [22], [23], [24]. Supercomputers and grid systems [25] that aggregate them depend on efficient schedulers to keep expensive machines and storage systems (often shared by many scientists) highly utilized. To help design and understand the performance characteristics of these schedulers, a number of different workload modeling approaches have been explored.

Our work is particularly similar in methodology to the approaches described in [22] and [23]. We fit parametric models to workload traces culled from logging data *post facto*. We use both statistical goodness-of-fit metrics and instrumented simulations to determine the efficacy of each model.

Our work differs, primarily, in its motivation. While scheduler design is one possible use for our technique, it is also designed to be viable as a capacity planning tool. Thus we require a greater level of registration between measurement and modeled data than previous efforts, particularly with respect to mean and variance.

Workload models (in the form of traffic models) have also been developed for networks [26], [27], [28], network-facing services [29], [10], [30], and operating systems [8], [9]. We too seek reliable parametric distributional characterizations of the load presented to a network-facing set of services (*e.g.* a Eucalyptus private cloud). Unlike many of the traffic models, however, our approach generates distributions that can be sampled to produce synthetic sequences of work units. From these sequences, aggregate measures of cloud performance can then be derived. Traffic models, on the other hand, typically produce descriptions of aggregate behavior at some measurement point (*e.g.* the number of packets passing a specific locale in a network). The time series of these aggregates (at some suitable scale) captures the variability that will be experienced at the measurement point.

Finally, our work is part of a growing body of work that investigates cloud workloads specifically [4], [5], [16]. This research has focused primarily on the use of empirical distributions, clustering, and regression to generate predictive models. Our approach is distinct in that it combines empirical distributions with parametric models and simulation. As result, in addition to predictive capabilities, it is possible to answer certain "what if" questions by manipulating the parameters of the automatically fitted hyperexponentials.

VI. Conclusion and Future Work

IaaS-style Cloud computing is a new model for providing authenticated, automated, and self-service resource provisioning and private clouds implement this model to effect a new type of data center management platform. In this study, we analyze and model workload measurements from several commercial private cloud deployments. We compare the efficacy of three statistical approaches to modeling workload and find that the use of a 3-phase hyperexponential is most appropriate. While clearly early in the technology life cycle for cloud computing (indeed we believe this study to be the first to analyze production enterprise workloads) these results provide insight and guidance to those researchers and professionals who are working to improve the overall approach or who are concerned with making the best use of private clouds.

As part of our future work, we will enhance the predictive capabilities that these results engender. Specifically, we plan to study both on-line cloud schedulers and off-line capacity planning methods useful for data center design and management.

References

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," National Institute of Standards and Technology, U.S. Department of Commerce, Tech. Rep. SP800-145, September 2011.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010.

[3] A. K. Mishra, J. L. Hellerstein, W. Cirne, and C. R. Das, "Towards characterizing cloud backend workloads: insights from google compute clusters," *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 4, pp. 34–41, 2010.

[4] A. Ganapathi, Y. Chen, A. Fox, R. Katz, and D. Patterson, "Statistics-driven workload modeling for the cloud," in *Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on*. IEEE, 2010, pp. 87–92.

[5] Y. Chen, "Workload-driven design and evaluation of large-scale data-centric systems," DTIC Document, Tech. Rep., 2012.

[6] P. Shivam, S. Babu, and J. Chase, "Learning application models for utility resource planning," in *Third International Conference on Autonomic Computing (ICAC 2006)*. IEEE, June 2006, pp. 255–265.

[7] J. Rolia, L. Cherkasova, M. Arlitt, and A. Andrzejak, "A capacity management service for resource pools," in *Proceedings of the 5th international workshop on Software and performance*. ACM, 2005, pp. 229–237.

[8] M. Harchol-Balter and A. Downey, "Exploting process lifetime distributions for dynamic load balancing," *ACM Transactions on Computer Systems*, vol. 15, no. 3, pp. 253–285, August 1997.

[9] D. L. Eager, E. D. Lazowska, and J. Zahorjan, "The limited performance benefits of migrating active processes for load sharing," vol. 16, no. 1, pp. 63–72, May 1988.

[10] M. E. Crovella, "Performance evaluation with heavy tailed distributions," in *Job Scheduling Strategies for Parallel Processing*. Springer, 2001, pp. 1–10.

[11] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper, "Workload analysis and demand prediction of enterprise data center applications," in *Workload Characterization, 2007. IISWC 2007. IEEE 10th International Symposium on*. IEEE, 2007, pp. 171–180.

[12] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The eucalyptus open-source cloud-computing system," in *Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on*. IEEE, 2009, pp. 124–131.

[13] Eucalyptus Systems Inc. (2013) http://www.eucalyptus.com.

[14] R. Wolski and J. Brevik. (2013) http://www.cs.ucsb.edu/ rich/workload.

[15] J. Murty, *Programming Amazon Web Services: S3, EC2, SQS, FPS, and SimpleDB*. O'Reilly Media, Inc., 2009.

[16] A. Khan, X. Yan, S. Tao, and N. Anerousis, “Workload characterization and prediction in the cloud: A multiple time series approach,” in *Network Operations and Management Symposium (NOMS), 2012 IEEE*. IEEE, 2012, pp. 1287–1294.

[17] O. Häggström, O. Nerman, and S. Asmussen, *EMPHT: A Program for Fitting Phase-type Distributions*. Chalmers tekniska högskola, 1992.

[18] M. Olsson, “The empht-programme,” *Relatório técnico, Department of Mathematics, Chalmers University of Technology, and Göteborg University, Sweden*, 1998.

[19] F. J. Massey Jr, “The kolmogorov-smirnov test for goodness of fit,” *Journal of the American statistical Association*, vol. 46, no. 253, pp. 68–78, 1951.

[20] A. Iosup, H. Li, M. Jan, S. Anoep, C. Dumitrescu, L. Wolters, and D. H. Epema, “The grid workloads archive,” *Future Generation Computer Systems*, vol. 24, no. 7, pp. 672–686, 2008.

[21] B. Song, C. Ernemann, and R. Yahyapour, “Parallel computer workload modeling with markov chains,” in *Job Scheduling Strategies for Parallel Processing*. Springer, 2005, pp. 47–62.

[22] U. Lublin and D. G. Feitelson, “The workload on parallel supercomputers: modeling the characteristics of rigid jobs,” *Journal of Parallel and Distributed Computing*, vol. 63, no. 11, pp. 1105–1122, 2003.

[23] A. B. Downey, “A parallel workload model and its implications for processor allocation,” *Cluster Computing*, vol. 1, no. 1, pp. 133–145, 1998.

[24] W. Cirne and F. Berman, “A comprehensive model of the supercomputer workload,” in *Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop on*. IEEE, 2001, pp. 140–148.

[25] I. Foster and C. Kesselman, *The Grid 2: Blueprint for a new computing infrastructure*. Access Online via Elsevier, 2003.

[26] W. Willinger, V. Paxson, and M. S. Taqqu, “Self-similarity and heavy tails: Structural modeling of network traffic,” *A practical guide to heavy tails: statistical techniques and applications*, vol. 23, pp. 27–53, 1998.

[27] V. Paxson and S. Floyd, “Wide area traffic: the failure of poisson modeling,” *IEEE/ACM Transactions on Networking (ToN)*, vol. 3, no. 3, pp. 226–244, 1995.

[28] R. H. Riedi, M. S. Crouse, V. J. Ribeiro, and R. G. Baraniuk, “A multifractal wavelet model with application to network traffic,” *Information Theory, IEEE Transactions on*, vol. 45, no. 3, pp. 992–1018, 1999.

[29] P. Barford and M. Crovella, “Generating representative web workloads for network and server performance evaluation,” *ACM SIGMETRICS Performance Evaluation Review*, vol. 26, no. 1, pp. 151–160, 1998.

[30] S. D. Gribble, G. S. Manku, D. Roselli, E. A. Brewer, T. J. Gibson, and E. L. Miller, “Self-similarity in file systems,” *ACM SIGMETRICS Performance Evaluation Review*, vol. 26, no. 1, pp. 141–150, 1998.