Optimizing for Tail Sojourn Times of Cloud Clusters

Mathias Björkqvist, Natarajan Gautam, Robert Birke, Lydia Y. Chen, and Walter Binder

Abstract—A common pitfall when hosting applications on today's cloud environments is that virtual servers often experience varying execution speeds due to the interference from co-located virtual servers degrading the tail sojourn times specified in service level agreements. Motivated by the significance of tail sojourn times for cloud clusters, we develop a model of N parallel virtual server queues, each of which processes jobs in a processor sharing fashion under varying execution speeds governed by Markov-modulated processes. We derive the tail distribution of the workloads for each server and the approximation for the tail sojourn times based on large deviation analysis. Furthermore, we optimize the cluster sizes that fulfill the requirements of target tail sojourn times. Extensive simulation experiments show very good matches to the derived analysis in a variety of scenarios, i.e., large numbers of servers experiencing a high number of different execution speeds, under various traffic intensities, workload variations and cluster sizes. Finally, we apply our proposed analysis on estimating the tail sojourn times of a Wikipedia system hosted in a private cloud, and the testbed results strongly confirm the applicability and accuracy of our analysis.

Index Terms—Tail response times, cloud system, large deviation analysis, capacity provisioning

1 Introduction

Several empirical studies [1], [2], [3], [4] point out a common pitfall in clouds: the execution speed of an application within a virtual machine (VM) fluctuates significantly due to the heterogeneity of the underlying hardware and the workloads co-located on the same physical host. Although virtualization enables efficient multiplexing of workloads across ample hardware resources, performance isolation is limited [5], [6]. The resulting exogenous variability not only hampers the satisfaction of the users, but also results in non-negligible business losses associated with the violation of service level agreements (SLAs) often specified in terms of tail sojourn times.

The degradation of tail sojourn times in the cloud is further exacerbated when deploying cluster-based applications [1], i.e., relying on a large number of VMs. Web [7] and big data services [8] are typical examples requiring such cluster deployments. In addition to the modulated execution speed and cluster size, the distribution of sojourn times, particularly the tail, is also affected by the load balancing algorithm distributing the load across VMs and the processor scheduling mechanism at each VM. Typically, a simple round robin algorithm is widely adopted, such as the one used in the Amazon EC2 cloud [9]. Requests are executed in a Processor Sharing (PS) fashion on individual VMs, which are typically hosted on separate physical servers. Overall, when deploying application clusters on today's cloud, three aspects are crucial for capturing the distribution of workloads and sojourn times: the modulated execution speeds of VMs, the load balancing algorithm, and the processor scheduling. Our research here aims to address the challenging question of how to best dimension the size of a cluster deployed in a cloud, in terms of number of VMs experiencing varying execution speeds, so that the target value of tail sojourn times can be met. We particularly take an analytical perspective and focus on deriving the tail response times for any given cloud cluster size using various key system parameters.

Related work in the areas of queueing networks and operations research sheds light on analyzing either single or multiple aspects of deploying clusters in the cloud. Obtaining the distribution of sojourn times is always challenging, especially with complex arrival and service processes, i.e., Markov modulated speed, and non- First Come First Serve (FCFS) scheduling. The only easy expression relates to the first moment of sojourn times. A large body of related work concentrates on deriving the

- *M. Björkqvist, R. Birke and L. Y. Chen are with IBM Research Zurich Laboratory, e-mail: {mbj,bir,yic}@zurich.ibm.com*
- *N. Gautan is with Texas A&M University, e-mail: gautam@tamu.edu*
- *W. Binder is with University of Lugano (USI), e-mail: walter.binder@usi.ch*

Manuscript received March 5, 2015.

distribution of the $M/G/1/PS$ queue [10], [11], where the service rate is fixed. Considering varying service rates, the related work centers around two directions: (1) service rates depending on the state of the system [12], [13] and (2) service rates changing due to external environmental processes [14], [15], [16], [17]. Motivated by the fact that the "speed" of a system increases with the number and variability of jobs, Gupta et al. [13] developed the approximation for mean sojourn times for $M/G/PS-MPL$ and $GI/G/PS-MPL$ queues with state-dependent service rate, where MPL denotes the multi-programming limit. While most of the aforementioned work centers around single server/queue, Dorsman et al. [17] and Casale et al. [3] study parallel queueing networks with Markov-modulated execution speeds using a functional central limit theorem and ordinary differential equations, respectively, with a strong assumption on Markovian arrivals. Therefore, little is known on the challenging question of how to predict the tail sojourn times for parallel queueing systems with renewal arrivals, high job size variability, Markov-modulated execution speeds, and processor sharing discipline under various traffic intensities.

Various strategies have been proposed to overcome the degradation of tail sojourn times due to exogenous variability. On one hand, various opportunistic VM selection algorithms [18], [19] reactively obtain the VMs with better performance in terms of execution speed. On the other hand, cluster sizing algorithms aim to proactively provision VMs according to their predicted performance, such as the tail throughput [6], and SLA targets. Though a pro-active optimization scheme is able to provide a reliable statistical guarantee on the performance metrics of interest, one needs to first obtain the prediction of those metrics, such as tail sojourn times. Overall, optimizing for tail sojourn times is achieved as best effort, without any guarantees.

In this paper, we develop an abstract parallel queueing system, where each queue is a $G/G/1/PS$ with Markov modulated execution speeds, to represent the application cluster hosted in a cloud. We obtain the distribution of workloads accumulated in the system, with special focus on their tail, using large deviation analysis. To derive the approximation of the tail sojourn times, we leverage the workload distribution and a mean-based analysis of the $M/G/1/PS$ queue with an average execution speed. We first derive the conditional distribution of the number of jobs for a given workload distribution for the $M/G/1/PS$ queue. Then we develop an approximation scheme for the tail sojourn times – a kind of worst case analysis. As a special case to model highly varying job sizes, we present closed form results for a degenerate hyperexponential distribution. We compare the proposed analysis against simulation results under various parameter settings, i.e., number of servers, different levels of exogenous variability (execution speeds), traffic intensities, and job size variability. Moreover, we construct a small case study on estimating the tail sojourn times of a Wikipedia system running in a VM subject to speed changes deployed on an IBM private cloud. Overall, our analysis shows a good match with experimental results for the tail distribution of workloads, and sojourn times, especially in cases with high job variability, number of speeds, and number of servers.

Our contribution can be summarized as follows. First, we derive the workload distribution for hard-to-analyze systems that capture the key characteristics of today's cloud systems, i.e., renewal arrivals, highly varying job sizes, Markov-modulated execution speeds, processor sharing, and round-robin load balancing. Second, we develop an approximation scheme for the tail sojourn times, which are one of the critical SLA parameters, and further optimize the cluster size based on that. Last but not least, we evaluate our proposed analysis via extensive simulation and a small Wikipedia prototype system in an IBM private cloud.

The outline of this work is as follows. Section 2 provides an overview of the system model. In Section 3 we obtain the workload distribution based on the large deviation analysis for the $G/G/1/PS$ queue with Markov-modulated execution speeds and illustrate an approximation scheme to obtain the tail sojourn times. We develop a mean-based approximation in Section 4, which captures the conditional probability of the number of jobs and tail sojourn times for the $M/G/1/PS$ queue with a special case on degenerated hyperexponential distribution of job sizes. Extensive experiments comparing analysis and simulation are given in Section 5, followed by the testbed study of a Wikipedia system in Section 6. Section 7 presents related work, followed by conclusions in Section 8.

2 Problem Statement And System Model

2.1 A Case Study of Wikipedia in the Cloud

We first motivate this work by a case study of deploying a Wikipedia service in the cloud. User page requests are first routed to a load balancer, which forwards each request to one of the Wikipedia VMs

according to a certain policy. We define such requests as *jobs*, and page sizes as *job sizes*. The default load balancing policies available on the Apache web server are "round robin", and "bybusiness", which is essentially known as the join-the-shortest-queue [20] policy in the queueing literature. The number of load balancers can be scaled with the cluster size, i.e., a larger number of load balancers for bigger clusters. An example statistics taken from Wikipedia says that on average one load balancers is for 25-30 application servers [21].

To build the Wikipedia system, we first downloaded a partial *pages-articles.xml* dump, consisting of 6284 entries amounting to 263 MB. The average requested page size is roughly 74 kB and the standard deviation is around 63 kB. Each Wikipedia VM is equipped with Ubuntu 14, Apache as web server (v2.4.7), PHP (v5.5.9) as server-side script engine [22], MediaWiki 1.23.6 [23] as web application, and a MySQL (v5.5.40) as database [24]. Moreoever, each Wikipedia VM is configured with one virtual core and 4 GB of virtual memory. The effective core capacity of a VM depends on how many VMs are consolidated on the hosting physical server. E.g., consolidating 4 VMs equipped with 1 core each on a physical server with a capacity of 4 cores, results in an effective VM capacity of exactly 1 core. Whereas, consolidating 5 such VMs on the same server decreases the effective capacity to the level of only 0.8 core. In contrast to CPU cores, the state-of-the-practice in the cloud is to not overcommit memory resources [25], i.e., the allocation of virtual memory of consolidated VMs is very similar to actual physical memory. The dynamics of fluctuation in effective core capacity can be modeled as Markov modulated processes [3], i.e., staying at different levels of effective core capacity for exponentially distributed times. In terms of scheduling policies, processor sharing is employed to process requests in modern web servers [13]. Consequently, VMs execute jobs at different execution speeds, corresponding to different levels of effective core capacity governed by a Markov modulated process.

When deploying such a cluster in a cloud, an essential and challenging question is how to size the cluster such that the target tail response time can be met? For example, can a cluster of three Wikipedia VMs sustain a performance level such that the 99^{th} percentile of sojourn times is lower than target $\bar{W}$?

2.2 System Model

In the following, we present an abstract system model that serves the proxy for aforementioned

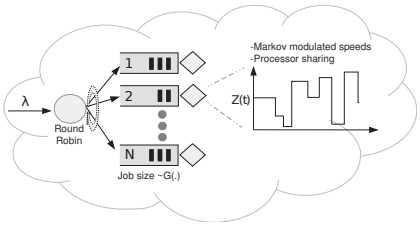


Fig. 1: Queueing schematics of a cloud cluster

real system and the base for our analysis of tails response times. We consider a system of N parallel servers and a dispatcher in front of them, as depicted in Figure 1. Each server is functionally equivalent. Jobs arrive to the dispatcher according to a renewal process with a rate of λ per second, and an inter-arrival time squared coefficient of variation (SCOV) c_a^2. Each job requires a random amount of page (say, in KBytes) to be processed by one of the servers. We assume that the amount of work, denoted by H, for various arrivals is IID with a common CDF $G(\cdot)$, mean m and SCOV c^2. The dispatcher cannot observe the state of the servers, and hence routes jobs to the servers in a round robin fashion. The servers use complete processor sharing. However, the speed of the server changes according to a finite-state continuous time Markov chain (CTMC) $\{Z(t), t \geq 0\}$ with state-space $\mathcal{S}$ and infinitesimal generator matrix $\mathbf{Q}$. At any time t, if $Z(t) = i$, then the server uses execution speed ϕ_i, for any $i \in \mathcal{S}$. We define the unit of ϕ_i as KBytes per second. If there are no jobs at the server at time t then the processor is idle, otherwise it serves jobs at speed ϕ_i. Note that the speed can change during any time of the job execution process, i.e., a job can be executed over multiple speeds. In summary, using Kendall's notation from queueing theory, each queue of our system is a $G/G/1/PS(\boldsymbol{\phi})$ queue, where $\boldsymbol{\phi}$ denotes the vector of all possible execution speeds.

This is a somewhat non-traditional description of a queueing system. Hence before proceeding ahead, we explain the service process in some detail. Say, at time t an arrival occurs to a server which already has $n-1$ jobs that it is processing in parallel. Let x_1, x_2, ..., x_n be the remaining amount of work (in KBytes) for jobs 1, 2, ..., n (where the n^{th} job corresponds to the one that just arrived at time t). Since the server's speed is ϕ_i, the amount of work for each of the n jobs would reduce at rate ϕ_i/n because of the processor sharing discipline. For instance, the k^{th} job has the smallest remaining work, i.e., $x_k = \min\{x_1, \ldots, x_n\}$. Then, the k^{th} job would

be completed at time $t+nx_k/\phi_i$, provided that there are no arrivals as well as no state changes for execution speeds during the interval $(t, t+nx_k/\phi_i)$. If there are arrivals during the interval $(t, t+nx_k/\phi_i)$ but no state change, then as soon as the first arrival occurs, each job would now be processed at rate $\phi_i/(n+1)$. Conversely, if a state change but no arrival occurs during the interval $(t, t+nx_k/\phi_i)$, then immediately after the state changes (to say $j \in \mathcal{S}$) each of the n jobs would now get processed at rate ϕ_j/n.

Let $\mathbf{p}$ be the steady-state probability row-vector of the CTMC $\{Z(t), t \geq 0\}$ (assuming it is irreducible), i.e., $\mathbf{p}$ satisfies $\mathbf{pQ} = \overline{0}$ and $\mathbf{p}\overline{1}' = 1$ (where $\overline{0}$ and $\overline{1}$ are row-vectors of zeros and ones respectively). The condition for system stability is

$$\frac{\lambda m}{N} < \mathbf{p}\boldsymbol{\phi} \tag{1}$$

where $\boldsymbol{\phi}$ is a column vector of the ϕ_i values. Thus from inequality (1), we can immediately identify the minimum number of servers N that would result in a stable system.

Prior to formally introducing our problem, we first describe the definition for the so-called $(1-\epsilon)^{th}$ percentile of workloads and sojourn times. For a random variable X, e.g., workloads and sojourn times, the value $X_{1-\epsilon}$ is such that $P\{X \leq X_{1-\epsilon}\} = 1-\epsilon$. Thus $X_{1-\epsilon}$ is the $(1-\epsilon)^{th}$ percentile of X.

Problem statement: *What we look for is the minimum number of servers that would guarantee that the sojourn time for an arbitrary arrival in steady state would be less that ζ with a probability greater than $1-\epsilon$, for some given values of ζ and ϵ, i.e., so-called $(1-\epsilon)^{th}$ sojourn times.*

For that, we assume there are N servers that would result in a stable system, i.e., the inequality (1) is satisfied. Specifically, let S be the sojourn time experienced by an arbitrary job that arrives in steady state to one of the N servers. We obtain an expression for $P\{S > \zeta\}$ and subsequently check if it is less than ϵ. If not, since our expression for $P\{S > \zeta\}$ is a function of N, it can be easily changed and we can find the smallest N so that $P\{S > \zeta\}$ is less than ϵ.

3 Tail Sojourn Times of $G/G/1/PS(\boldsymbol{\phi})$ Queues

In this section we consider any one of the N servers, i.e., $G/G/1/PS(\boldsymbol{\phi})$ queues, and aim to obtain the (tail) sojourn times. To such an end, we first derive the tail workload distribution based on large deviation analysis, and then derive approximation schemes of sojourn times based on a mean-based approximation of the $M/G/1/PS$ queue with an average execution speed. It is crucial to point out that it is extremely difficult to obtain the sojourn time distribution for the $M/G/1/PS$ queue. However, in this study we add some features beyond the $M/G/1/PS$ queue, such as Markov-modulated execution speed and round robin dispatching (resulting in general inter-arrival times). Undoubtedly, their analysis gets even more intractable, and hence we resort to an approximation scheme.

3.1 Workload Distribution

To obtain the tail distribution of the workload under round-robin load balancing, we use a method based on large deviations. Here we consider any one of the N servers. Arrivals occur at any server according to a delayed renewal process, with inter-renewal times according to a general distribution with mean N/λ and variance Nc_a^2/λ^2. Let W be the workload to be processed in the server at any arbitrary time in steady state. To obtain the limiting tail distribution of the workload in the queue, $W(t)$, i.e., $P\{W(t) > x\}$ as $t \to \infty$ for some large x, we consider a fictitious server with constant processing speed $\phi_{\max} = \max_{i\in\mathcal{S}} \phi_i$. In addition to the regular stream of arrivals, the fictitious server also gets fluid workload at rate $\phi_{\max} - \phi_j$ at time t (if $Z(t) = j$ for some $j \in \mathcal{S}$) from a compensating source. Note that the workload at time t at this fictitious server is stochastically identical to $W(t)$ for all $t \geq 0$. Thus we analyze the tail distribution of this fictitious server and represent it as $W(t)$.

Let $A(t)$ be the total amount of workload that arrived from time 0 to t from the regular renewal source. We can compute for some $v \geq 0$,

$$\begin{aligned} E[e^{vA(t)}] &= E[\{\tilde{G}(-v)\}^{\mathcal{N}(t)}] \\ &= e^{\frac{\lambda t}{N}\log\{\tilde{G}(-v)\} + \frac{\lambda t c_a^2}{2N^2}\{\log\{\tilde{G}(-v)\}\}^2} \end{aligned}$$

where $\mathcal{N}(t)$ is the number of arrivals in time t, assuming a non-delayed renewal process (this would not affect us as we will let $t \to \infty$), and for the latter term we use a Normal approximation to $\mathcal{N}(t)$ which is reasonable for large t. Now, we can write down $h(v)$ the asymptotic logarithmic moment generating function (ALMGF) defined as [11]

$$h(v) = \lim_{t\to\infty} \frac{1}{t} \log E[e^{vA(t)}]$$

and for our $A(t)$, $h(v)$ can be computed as

$$h(v) = \frac{\lambda}{N}\log\{\tilde{G}(-v)\} + \frac{\lambda c_a^2}{2N^2}\{\log\{\tilde{G}(-v)\}\}^2 \tag{2}$$

for any $v \geq 0$. Likewise, since the compensating source is a CTMC fluid source, we can compute its ALMGF $h_c(v)$ as

$$h_c(v) = e(\mathbf{Q} + v\phi_{\max}\mathbf{I} - v\mathbf{\Phi}) \quad (3)$$

where $e(A)$ denotes the largest real eigenvalue of a square matrix A, $\mathbf{I}$ is the identity matrix, and $\mathbf{\Phi}$ is the diagonal rate matrix, i.e., $\mathbf{\Phi} = [diag(\phi)]$,

Let η be the unique solution to

$$h(\eta) + h_c(\eta) = \eta\phi_{\max}$$

where $h(v)$ and $h_c(v)$ can be computed from Equations (2) and (3), respectively. Then, using results from large deviations,

$$\lim_{t\to\infty} P\{W(t) > x\} \approx e^{-\eta x} \quad (4)$$

for large values of x (in particular as $x \to \infty$). Thus we have an approximation for the tail distribution of the steady-state workload.

3.2 Approximating Tail Sojourn Times

Here, we seek to obtain an expression for the sojourn time tail, based on the workload tail derived in Section 3.1. Notice that in Section 3.1 we do not make any assumptions about the service discipline except that it is work conserving. Here, we focus on the tail probability of sojourn time under a *processor sharing* discipline.

Assumptions. In particular, we assume fast dynamics of execution speeds, i.e., jobs can experience many different execution speeds during their sojourn. The motivation behind is two-fold. First, co-located neighboring VMs can execute complex applications that exhibit volatile run-time behavior and cause frequent changes in execution speeds. Second, under a special case, it is possible to analytically derive the tail probabilities of sojourn times under very fast dynamics: $\lambda/N \ll -\min_{j\in\mathcal{S}} q_{jj}$, where q_{jj} is the j^{th} element on the diagonal of $\mathbf{Q}$ matrix of CTMC. Essentially, the dynamics of the number of jobs in the system is much faster than the dynamics of execution speeds, i.e., a server changes state many times during a job's sojourn. Especially for the scenario in [8], this is a reasonable assumption.

3.2.1 Approximation Scheme

As described earlier, obtaining the tail sojourn time for an $M/G/1/PS$ queueing system with fixed speed is itself not straightforward, and to the best of our knowledge not available in the open literature. Therefore, as a first step, we obtain an approximate expression for $S_{1-\epsilon}(M, \overline{\phi})$, the $(1-\epsilon)^{th}$ percentile of the sojourn time under the $M/G/1/PS$ queue with an average speed $\overline{\phi}$ (where $\overline{\phi} = \mathbf{p}\boldsymbol{\phi}$). To obtain an expression for $S_{1-\epsilon}(M, \overline{\phi})$, we need a closed-form formula for $W_{1-\epsilon}(M, \overline{\phi})$, the $(1-\epsilon)^{th}$ percentile of the workload under the $M/G/1/PS$ queue with constant speed $\overline{\phi}$. However, what we ultimately need is an approximation for $S_{1-\epsilon}(G, \boldsymbol{\phi})$, the $(1-\epsilon)^{th}$ percentile of the sojourn time under the $G/G/1/PS$ queue with varying speed $\boldsymbol{\phi}$. We already have an expression for the corresponding workload $W_{1-\epsilon}(G, \boldsymbol{\phi})$ from Equation (4), which can be written as

$$W_{1-\epsilon}(G, \boldsymbol{\phi}) = \frac{-1}{\eta}\log(1-\epsilon). \quad (5)$$

Then, based on the expressions for $S_{1-\epsilon}(M, \overline{\phi})$, $W_{1-\epsilon}(M, \overline{\phi})$ and $W_{1-\epsilon}(G, \boldsymbol{\phi})$, we can obtain an approximation for $S_{1-\epsilon}(G, \boldsymbol{\phi})$ using

$$\frac{S_{1-\epsilon}(M, \overline{\phi})}{W_{1-\epsilon}(M, \overline{\phi})} \approx \frac{S_{1-\epsilon}(G, \boldsymbol{\phi})}{W_{1-\epsilon}(G, \boldsymbol{\phi})}$$

conjecturing that the ratio of the almost-worst-case sojourn time to the almost-worst-case workload would be approximately equal for two queues that have the same mean arrival rate, same mean processing speed and same job size distribution. Similar ideas have been considered in other areas of queueing theory connecting Markovian queues to general queues (see Buzacott and Shanthikumar [5]). Essentially, we propose to estimate

$$S_{1-\epsilon}(G, \boldsymbol{\phi}) \approx S_{1-\epsilon}(M, \overline{\phi})\frac{W_{1-\epsilon}(G, \boldsymbol{\phi})}{W_{1-\epsilon}(M, \overline{\phi})}, \quad (6)$$

where the derivation of $W_{1-\epsilon}(G, \boldsymbol{\phi})$ is given in Equation (5) and $S_{1-\epsilon}(M, \overline{\phi})$ and $W_{1-\epsilon}(M, \overline{\phi})$ are given in the next section based on a mean-based approximation.

To achieve our objective of this study, i.e., searching for a minimal number of servers that guarantees the $1-\epsilon^{th}$ sojourn times, we need to iterate through $W_{1-\epsilon}(G, \boldsymbol{\phi})$ with different number of servers, i.e., N, using our proposed approximation.

4 Mean-based Approximation

To obtain values of $S_{1-\epsilon}(M, \overline{\phi})$ in Equation (6), we propose a mean-based approximation, i.e., considering the average arrival rate, average execution speed ($\overline{\phi}$), and average number of jobs, for an $M/G/1/PS(\overline{\phi})$ queueing system. We first obtain the distribution of the number of jobs in an $M/G/1/PS(\overline{\phi})$ queueing system, conditioned upon the workload. To accommodate a high variability of job size, we approximate the job size to be degenerate hyperexponential and derive the LST expression

of the number of jobs as well as sojourn times, i.e., $W_{1-\epsilon}(M, \overline{\phi})$. Finally, combining the conditional probability of the number of jobs and $W_{1-\epsilon}$, we develop an approximation scheme for $S_{1-\epsilon}(M, \overline{\phi})$.

4.1 Conditional Distribution of Number of Jobs

For such a server that adopts processor sharing, let $X(t)$ be the number of jobs in the system at time t and $R_i(t)$ be the remaining amount of work to be processed for the i^{th} job in the system. The multi-dimensional stochastic process $\{(X(t), R_1(t), R_2(t), \ldots, R_{X(t)}(t)), t \geq 0\}$ is a Markov process. We denote $F_n(t, y_1, y_2, \ldots, y_n)$ as the joint probability

$$F_n(t, y_1, y_2, \ldots, y_n) = P\{X(t) = n, R_1(t) \leq y_1, R_2(t) \leq y_2, \ldots, R_n \leq y_n\}.$$

Thereby the density function $f_n(t, y_1, y_2, \ldots, y_n)$ is defined as

$$f_n(t, y_1, y_2, \ldots, y_n) = \frac{\partial^n F_n(t, y_1, y_2, \ldots, y_n)}{\partial y_1 \, \partial y_2 \, \ldots \, \partial y_n}.$$

It is known that as $t \to \infty$, $f_n(t, y_1, y_2, \ldots, y_n)$ converges to the stationary distribution $f_n(y_1, y_2, \ldots, y_n)$ which is given by

$$f_n(y_1, y_2, \ldots, y_n) = (1-\rho)\frac{\lambda^n}{(N\overline{\phi})^n}\prod_{i=1}^{n}[1 - G(y_i)]$$

where $\rho = \lambda m/(N\overline{\phi})$.

Using the above we next obtain the joint probability that there are n jobs in the system and the total workload is not more than y which we denote as $\hat{F}_n(y)$ and define as

$$\hat{F}_n(y) = \lim_{t\to\infty} P\{X(t) = n, R_1(t) + R_2(t) + \ldots + R_n(t) \leq y\}$$

for all $n \geq 0$ and $y \geq 0$. Using the expression for $f_n(y_1, y_2, \ldots, y_n)$, we have

$$\begin{aligned}\hat{F}_n(y) &= \int_0^y \int_0^{y-y_1} \int_0^{y-y_1-y_2} \cdots \int_0^{y-y_1-y_2-\ldots-y_{n-1}} f_n(y_1, y_2, \ldots, y_n) dy_n dy_{n-1} \ldots dy_1 \\ &= (1-\rho)\frac{\lambda^n}{(N\overline{\phi})^n}\int_0^y (1-G(y_1)) \int_0^{y-y_1}(1-G(y_2)) \int_0^{y-y_1-y_2}(1-G(y_3)) \\ &\quad \cdots \int_0^{y-y_1-y_2-\ldots-y_{n-1}} (1-G(y_n)) \, dy_n dy_{n-1} \ldots dy_1.\end{aligned}$$

Since there is an n-folded convolution, it is only natural that we write down the LST of $\hat{F}_n(y)$, namely $\tilde{\hat{F}}_n(s)$, which after some algebra and calculus is

$$\tilde{\hat{F}}_n(s) = (1-\rho)\frac{\lambda^n}{(sN\overline{\phi})^n}[1 - \tilde{G}(s)]^n \qquad (7)$$

for all $s \geq 0$.

It would typically be difficult to invert the above LST and obtain $\hat{F}_n(y)$ except for some special cases that we would investigate in the next subsection. Nonetheless, we go ahead and use the density $\hat{f}_n(y)$ defined as the derivative of $\hat{F}_n(y)$ with respect to y for all $y > 0$. Using that, we can write down an expression for the steady-state probability that there are n jobs in the system, given the workload, $W(t)$, is y as

$$\lim_{t\to\infty} P\{X(t) = n | W(t) = y\} = \frac{\hat{f}_n(y)}{\sum_{k=1}^{\infty} \hat{f}_k(y)}. \qquad (8)$$

4.2 Special Case: Degenerate Hyperexponential Work

Here, we consider a special case of amount of work requested by jobs, i.e., degenerate hyperexponential distribution, to represent a highly varying job size as well as to demonstrate how to compute conditional probability in Equation (8) and tail workload.

Let H be a non-negative random variable that has a degenerate hyperexponential distribution with parameters q and θ if its CDF is $P\{H \leq h\} = 1 - qe^{-\theta h}$ for some $h \geq 0$ where q satisfies $0 \leq q \leq 1$ and $\theta > 0$. Notice that when $q = 1$, H reduces to the standard exponential distribution, while $q = 0$ corresponds to $H = 0$, a deterministic value. Also, when $0 \leq q < 1$, H has a mass at 0 with probability $1 - q$. One can compute the LST of the CDF as $E[e^{-sh}] = (1 - q) + q\theta/(s + \theta)$. Further, the mean and variance of H are $E[H] = q/\theta$ and $Var[H] = (2q - q^2)/\theta^2$.

We model the amount of work a job brings to the following degenerate hyperexponential distribution. We particularly consider the workload distribution that has SCOV at least 1. From a practical standpoint this would be a reasonable assumption when there are many extremely tiny jobs and the SCOV is greater than 1 (both of which are typical in many server environments). Also, from a mathematical point of view, there are just two parameters to estimate. Recall from Section 2 that the amount of work has a mean m, SCOV c^2 and distribution $G(\cdot)$. We can estimate q and θ by fitting the mean and SCOV. Thus we have

$$q = 2/(1 + c^2) \quad \text{and} \quad \theta = 2/(m + mc^2).$$

4.2.1 Distribution of Number of Jobs

Now, reverting back to Equation (7), we approximate the LST $\hat{F}_n(s)$ as

$$\hat{\tilde{F}}_n(s) = (1-\rho)\frac{\lambda^n}{(sN\overline{\phi})^n}[sq/(s+\theta)]^n$$

which can be inverted to get

$$\hat{f}_n(y) = (1-\rho)\left(\frac{\lambda q}{N\overline{\phi}}\right)^n \frac{y^{n-1}e^{-\theta y}}{(n-1)!}$$

for all $y > 0$. Thereby, we can write down the steady state probability of having n jobs at the server, given the workload is y using Equation (8) for all $n \geq 1$ as

$$\lim_{t\to\infty} P\{X(t) = n | W(t) = y\} = \left(\frac{\lambda q y}{N\overline{\phi}}\right)^{n-1} \frac{e^{-\lambda q y/(N\overline{\phi})}}{(n-1)!}$$

which is a Poisson distribution with parameter $\lambda qy/(N\overline{\phi})$ with the adjustment made from n to $n-1$ since n cannot be zero if $y > 0$. Notice that the number of jobs at a server, given a non-zero workload does not depend on the mean job size. However, it does depend on the SCOV of job sizes through q.

4.2.2 Workload Distribution

Recall that we would like to analyze an $M/G/1$ queue with PP(λ/N) arrivals, and each arrival brings a random amount of work that needs to be processed by a single server. We let $\lambda' = \lambda/N$. The amount of work for various arrivals are IID degenerate-hyperexponential with common CDF, as described earlier. The server uses a single processing rate $\overline{\phi}$. The jobs can be served according to any work-conserving discipline, i.e., FCFS or processor sharing. If the $M/G/1$ queue is observed at an arbitrary time in steady state, then let V be the time to empty all the workload in the system. The LST of V (assuming stable) can be computed as:

$$E[e^{-sV}] = \frac{(1-\rho)s}{s - \lambda(1-\tilde{G}(-s/\overline{\phi}))}$$

where $\rho = \lambda' q/(\theta\overline{\phi})$ and $\tilde{G}(-s/\overline{\phi}) = (1-q) + q\theta/(s/\overline{\phi}+\theta)$. Rearranging the terms we get

$$E[e^{-sV}] = (1-\rho) + \rho\frac{\theta\overline{\phi} - \lambda' q}{\theta\overline{\phi} - \lambda' q + s}$$

which upon inverting we get the CDF of V as

$$P\{V \leq t\} = 1 - \rho e^{-(\theta\overline{\phi} - \lambda' q)t}$$

for all $t \geq 0$. Now, the amount of workload in steady-state W_∞ can be written as $W_\infty = V\overline{\phi}$. Hence the CDF of the amount of workload is

$$P\{W_\infty \leq x\} = 1 - \rho e^{-(\theta\overline{\phi} - \lambda' q)x/\overline{\phi}}.$$

Thus, we have $W_{1-\epsilon}(M, \overline{\phi}) = \{x : P\{W_\infty \leq x = 1-\epsilon\}\}$

4.3 Tail Sojourn Time Approximation

In the remainder of this subsection, we present an approximation scheme, based on the so-called $(1-\epsilon)$–worst case analysis. Our approximation scheme for $(1-\epsilon)^{th}$ percentile sojourn time is motivated by Little's law and based on the idea that the tail sojourn time depends on the tail job size, the tail number of jobs, and the average execution speed.

Approximation algorithm for $(1-\epsilon)$–worst case analysis

1) Obtain the average speed, $\overline{\phi} = \mathbf{p}\boldsymbol{\phi}$, where the terms are defined near Equation (1)
2) Obtain the $1-\epsilon^{th}$ percentile of the steady-state workload, $W_{1-\epsilon} = -\log(\epsilon)/\eta$, according to Equation (4).
3) Obtain the number of jobs in the system. Using the analysis in Section 4.2, we obtain the number of jobs to be a Poisson random variable with parameter $\lambda qy/(N\overline{\phi})$ where for y we use $W_{1-\epsilon}$ in step 2 above and $\overline{\phi}$ is from step 1. Then, we obtain 50^{th} percentile of number of jobs in the system, termed $J_{1-\epsilon}$, corresponding to having $W_{1-\epsilon}$ amount of workload.
4) Obtain the $(1-\epsilon)^{th}$ percentile of the amount of work brought by an arriving job in steady state, $Y_{1-\epsilon} = qG^{-1}(1-\epsilon)$, where q is the parameter in the degenerate hyperexponential distribution defined in Section 4.2.
5) Approximate the $(1-\epsilon)^{th}$ sojourn time as $S_{1-\epsilon} = Y_{1-\epsilon}(1 + J_{1-\epsilon})/\overline{\phi}$.

While steps 1, 2 and 4 of the above algorithm are straightforward, it is worthwhile explaining the rationale behind steps 3 and 5. As a conservative approximation we essentially assume that the tail sojourn time corresponds to the tail job size and tail number of jobs in the system. A crucial thing to observe is that small changes to the number of other jobs during an arriving job's sojourn (especially when that number of other jobs is high) would not adversely affect the sojourn time as the large job size would overwhelm, leading to the expression in step 5 and the use of the median number as a surrogate for the number of jobs in step 3.

5 Experimental Results

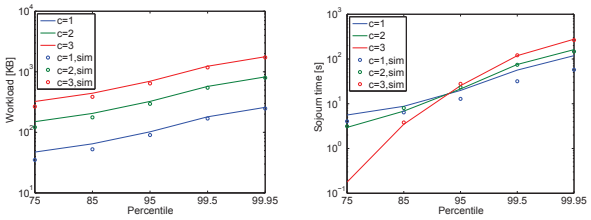
In this section, we present an extensive set of experimental results, comparing the proposed analysis against simulation results. Our objective is to show: (1) the accuracy of the proposed workload analysis; (2) the accuracy of the sojourn time approximations; (3) the robustness of our proposed scheme for different system choices, and (4) the optimality of the dimensioned cluster size. To such an end, we consider a large number of system and workload parameters, in particular, the number of servers, different execution speeds, job size variability, and varying number of load balancers. The metrics of interest are the tail workload distributions and the tail sojourn time distributions, i.e., the 50^{th}, 75^{th}, 85^{th}, 95^{th}, 99.5^{th}, 99.95^{th}, 99.995^{th}, and 99.9995^{th} percentiles. Due to lack of space we present only partial results. In the following, we first explain the experiment settings, followed by the accuracy and scalability of the proposed analysis, and finally, the analysis of optimizing the cluster for a target tail sojourn time.

5.1 Experiment Setup

We develop a simulator based on OMNet++ [26] for the system shown in Figure 1, consisting of N queues, i.e., $N = \{1, 4, 10, 20, 30\}$, each of which execute requests in a processor sharing fashion. Requests following Poisson distribution with rates $\lambda = \{0.8, 3.2, 8, 16, 24\}$, corresponding to $N = \{1, 4, 10, 20, 30\}$ respectively, are dispatched to each queue by a round-robin load balancer. Each server experiences different execution speeds, governed by the $\mathbf{Q}$ matrix of CTMC. In particular, we consider two scenarios: (1) servers alternate between 2 speeds, $\phi = [8, 10]$ according to $\mathbf{Q_2}$, (2) servers alternate between 5 speeds, $\phi = [8, 9, 10, 11, 12]$ according to $\mathbf{Q_5}$, where $\mathbf{Q_2}$ has all non-diagonal entries $q_{ij} = 0.05$ and $\mathbf{Q_5}$ has all non-diagonal entries $q_{ij} = 0.1$.

Each job brings a workload amount with mean $m = 10$ KBytes and different SCOV, i.e., $c^2 = \{1, 4, 9\}$. The job sizes follow the degenerate hyper-exponential distributions described in Section 4.2. Note that we purposely make the values of $\mathbf{Q_2}$ and $\mathbf{Q_5}$ comparable so as to study the effect of increasing the number of speed choices $|\phi|$.

The exact parameter combinations and the resulting prediction errors between the proposed analysis and the simulation results are listed in Table 1. The prediction error is defined as the absolute difference between the prediction and simulation result, divided by the simulation result. Sojourn times of lower percentiles, such as the 50^{th} and 75^{th} percentile, are measured as low as 0 in the simulation. In these cases the prediction error is listed as n/a in Table 1. In most cases, our prediction overestimates the simulation results. Note that for each combination, our simulation results are averaged across 10 runs, where each queue roughly receives 6.4 million requests. The simulation time grows exponentially with the number of speeds, servers, traffic intensities, as well as, the tail statistics. The longer simulations are in the order of several hours. In contrast, the computational time of the proposed analysis is negligible.



(a) Workload (b) Sojourn time
Varying c with $N = 4$, $|\phi| = 5$ (exp. 13, 14, 15)

Fig. 2: Sensitivity analysis of tail workloads and sojourn times.

5.2 Accuracy and Sensitivity Analysis

In this subsection, we first report on the accuracy of our proposed tail workload and sojourn time analysis, using Table 1, and then discuss how these two metrics vary across different parameter settings, and what their implications are on system design.

Prediction Errors

The average prediction error is around 5% for both the workload and sojourn time in most considered cases. For workload predictions, our model works very well particularly for higher percentiles and for highly varying workloads, i.e., for higher values of c, at a given number of servers. One can observe that in the workload part of the table, i.e., its middle section, prediction errors in the lower left corner are higher than in the upper right corner. This indicates that the workload prediction suffers slightly, in particular of overestimation, when increasing the number of servers and number of execution speeds.

As for sojourn time predictions, our proposed approximation results in slightly higher errors compared to the workload part. In particular, our estimations are highly conservative for job size distributions with low variance, i.e., $c = 1$, and for higher

Exp	Parameters					Workload error [%]					Sojourn Times error [%]				
	N	λ	$\|\phi\|$	c	ρ	p75	p85	p95	p99.5	p99.95	p75	p85	p95	p99.5	p99.95
1	1	0.8	2	1	0.89	9.1	6.0	2.9	0.6	0.1	88.2	101.6	138.6	181.1	213.3
2	1	0.8	2	2	0.89	8.8	6.1	3.5	1.6	1.1	7.1	13.4	14.7	24.5	32.0
3	1	0.8	2	3	0.89	8.9	6.3	3.6	1.6	3.6	n/a	4.8	6.2	4.0	0.1
4	4	3.2	2	1	0.89	18.1	12.6	7.4	3.8	2.3	34.5	46.1	62.4	94.5	113.6
5	4	3.2	2	2	0.89	11.5	8.2	4.9	2.2	0.9	0.9	5.8	2.3	8.3	13.9
6	4	3.2	2	3	0.89	10.2	7.2	4.3	1.9	1.6	n/a	5.0	11.5	9.9	5.3
7	10	8.0	2	1	0.89	29.2	21.9	15.2	10.4	8.6	27.3	42.3	62.4	86.1	105.4
8	10	8.0	2	2	0.89	13.9	10.3	6.9	4.6	3.6	8.4	4.2	5.2	4.5	11.9
9	10	8.0	2	3	0.89	11.3	8.2	5.4	3.2	2.1	n/a	4.3	13.6	10.7	5.4
10	20	16.0	2	1	0.89	17.1	10.9	5.2	1.5	0.7	16.5	30.3	40.2	65.2	84.8
11	20	16.0	2	2	0.89	10.6	7.2	4.0	1.8	1.2	10.9	6.8	7.8	1.8	9.1
12	20	16.0	2	3	0.89	10.0	7.0	4.0	1.9	1.7	n/a	5.6	14.7	11.8	6.6
13	30	24.0	2	1	0.89	20.9	14.2	8.1	3.5	2.1	19.5	21.5	43.5	68.7	83.4
14	30	24.0	2	2	0.89	11.5	8.0	4.6	2.6	2.2	10.2	6.1	7.1	2.4	7.6
15	30	24.0	2	3	0.89	10.3	7.3	4.4	2.2	2.3	n/a	5.3	14.4	11.5	7.3
16	1	0.8	5	1	0.80	15.5	9.8	4.8	1.1	1.0	60.2	76.7	116.3	153.0	183.6
17	1	0.8	5	2	0.80	18.2	12.4	7.1	3.0	3.0	6.3	3.1	7.5	16.9	25.2
18	1	0.8	5	3	0.80	18.9	12.8	7.2	3.1	0.6	n/a	9.0	4.2	3.3	10.1
19	4	3.2	5	1	0.80	34.5	22.9	13.1	6.7	4.5	38.1	36.8	54.1	77.3	107.0
20	4	3.2	5	2	0.80	23.5	16.2	9.6	5.2	3.3	5.9	14.7	8.5	1.9	10.2
21	4	3.2	5	3	0.80	21.2	14.7	8.8	4.7	2.6	n/a	8.8	8.9	1.9	4.7
22	10	8.0	5	1	0.80	53.9	36.9	23.3	14.8	11.8	17.2	20.8	45.1	75.2	99.4
23	10	8.0	5	2	0.80	27.3	19.2	12.1	7.3	5.5	20.3	13.3	7.0	3.6	7.9
24	10	8.0	5	3	0.80	22.8	16.0	9.9	5.6	3.7	n/a	8.2	8.3	1.2	5.6
25	20	16.0	5	1	0.80	36.9	22.9	11.5	4.3	1.8	7.1	10.4	32.6	48.6	73.1
26	20	16.0	5	2	0.80	23.2	15.5	8.7	4.2	2.4	22.4	15.7	9.6	4.7	4.9
27	20	16.0	5	3	0.80	20.9	14.2	8.2	4.2	3.1	n/a	9.3	9.5	2.6	4.1
28	30	24.0	5	1	0.80	43.3	27.8	15.4	7.4	4.4	9.9	13.3	36.0	52.6	77.6
29	30	24.0	5	2	0.80	24.8	16.8	9.9	5.2	4.0	21.7	15.0	8.7	3.8	6.0
30	30	24.0	5	3	0.80	21.6	14.8	8.8	4.6	2.9	n/a	9.1	9.2	5.2	4.6

TABLE 1: Prediction errors, comparing analytical results with simulation: λ,$|\phi|$, c, and ρ denotes the arrival rates, execution speed vector, coefficient of variation, and system load, respectively

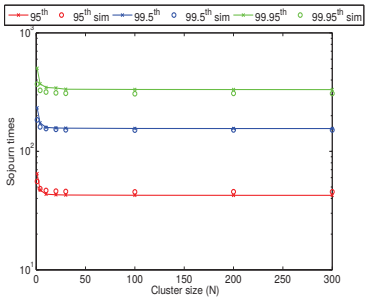
tail sojourn times. The good news is that the overestimation effect is mitigated with increasing variability of job sizes, increasing numbers of servers, and increasing number of execution speeds. Overall, our proposed analysis is very accurate in predicting the 95^{th} and 99.5^{th} percentile of sojourn times for systems with high numbers of servers experiencing high job size variability, and frequent execution speed changes due to the external environment.

Sensitivity Analysis

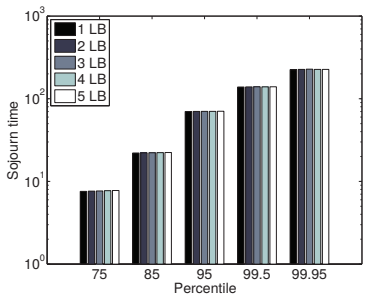
To see how the distribution tail of workload and sojourns time changes with the job variability, we select three sets of experiments and summarize them in Figure 2, corresponding to the cases having $N = 4$, $|\phi| = 5$ while varying $c = \{1, 2, 3\}$. We plot both the analytical and simulation results.

In Figures 2 (a) and (b), one can clearly observe that the workload increases with the degree of job variability for any given percentile. The tail workload grows exponentially, shown as almost linear curves in Figure 2 (a), due to the log scale. As stated earlier, our workload predictions are overestimated for lower percentiles, but very accurate for higher percentiles. In terms of tail sojourn time distributions, a different pattern with respect to the job variability can be observed. In Figure 2 (b), the tail distribution of sojourn times increases differently for different job variability values. Comparing $c = 1$ and $c = 3$, the 75^{th} and 85^{th} percentile of $c = 3$ are lower than $c = 1$, whereas for the 95^{th}, 99.5^{th} and 99.95^{th} percentile, the opposite holds true. Such an observation is not too surprising, since extensive related work has pointed out that the average sojourn times is much worse in systems with highly varying job sizes. The additional merits of our analysis is to pinpoint at which part of the tail the sojourn times degrade drastically and impact the average performance. This further indicates that dimensioning a cluster based on (high) tail sojourn times is very different from using the average sojourn time. Moreover, we again stress our sojourn time prediction as being particularly conservative for $c = 1$, i.e., overestimating, and very accurate for $c = 2, 3$.

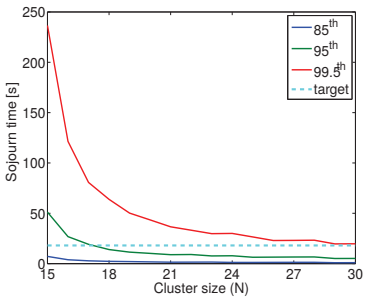
We also compare the above results with the cases having $N = 4$, $|\phi| = 2$. Due to lack of space, we skip the graphical presentation and only report the observations. All observations made in the first case still hold, except the exact values are slightly higher due to the lower traffic intensity, ρ. As such, the cross point among sojourn times for different jobs size variabilities happens at slightly higher percentiles. This highlights another important factor in dimensioning cloud clusters, i.e., the traffic intensity loading the system, but this is currently



(a) Scalability: 95^{th}, 99.5^{th}, 99.95^{th} percentiles for different cluster sizes.



(b) The effect of multiple load balancers in a cluster of 100 Wikipedia VMs.



(c) Optimal cluster size.

Fig. 3: Sensitivity analysis of scalability and optimal size.

out of the scope of this study.

5.3 Scalability

To see how our approximation scheme scales to larger systems, we expand the cluster size to up to 300 VMs. Figure 3a presents tail sojourn times of VMs that experience two execution speeds and a job size variability of $c = 2$. The simulation results are averaged over 5 runs. We can clearly see that the analytical predictions are more accurate for bigger clusters, for all three tail percentiles presented. For the 99.5^{th} and 99.95^{th} percentiles, we also see that the analytical predictions are slightly higher than the simulation results. Such conservative predictions can result in better guarantees for service level agreements, at the cost of slightly larger clusters. Similar observations can be made for the case of VMs experiencing 5 execution speeds, but due to space limitations we skip the presentation of this set of results. Note that the required simulation time is much longer for larger systems, especially for higher percentiles, whereas the analytical results have negligible computational overhead.

5.4 The Impact of Multiple Load Balancers

To accommodate increasing workloads, we need to scale not only the number of VMs in the cluster, but also the number of load balancers. Here, we investigate the impact on tail sojourn times from having multiple load balancers. In particular, we choose a cluster size of 100 VMs experiencing 5 execution speeds and job size variability of $c = 2$, and vary the number of load balancers from one to five. All arriving requests are first randomly distributed among the load balancers, which in turn apply the round-robin policy to distribute independently of each other the requests among the VMs. We depict the simulated sojourn times of different percentiles in Figure 3b. One can see that the differences between different numbers of load balancers are negligible for all presented percentiles. It can therefore be said that the performance of cloud clusters is very robust with respect to such a design choice. This observation also holds for the case where VMs experience 2 execution speeds, which further implies that the accuracy of our approximation scheme applies to clusters using different numbers of load balancers.

5.5 Optimizing for Cloud Clusters

The objective of this subsection is to demonstrate how our prediction can be applied in optimizing a cloud cluster so that the SLAs specified by the tail sojourn times can be achieved with the minimum number of servers, i.e., lowest cost. To such an end, we assume that the system has a job arrival rate of $\lambda = 14$ jobs per second, and the job size follows a degenerate hyperexponential distribution with $c = 3$. The server experiences 5 different execution speeds $|\phi| = 5$ governed by a CTMC with Q_5, described at the beginning of this section. The system also considers 15 seconds as a target value, which can be used for the SLA specified by a tail percentile. In Figure 3c, we show how the optimal cluster size changes across different percentiles, using 15 seconds as a target value.

In particular, we depict how the 85^{th}, 95^{th}, and 99.5^{th} percentile of sojourn times evolve with different cluster sizes, i.e., 15-30 servers, using our analysis. On one hand, a cluster size greater than 15 achieves that the 85^{th} sojourn time percentile is less than 15 seconds. On the other hand, 18 servers and 30 servers are the minimal cluster size to fulfill the same target at the 95^{th} and 99.5^{th} percentile,

respectively. To provide statistical guarantees on the long tail of sojourn times, a substantial service provisioning cost is unavoidable. Overall, the proposed analysis enables an efficient methodology to evaluate such emerging challenges, i.e., capturing the tail distribution of sojourn times in highly distributed and volatile systems, i.e., in terms of workloads and server execution speeds.

6 A Case Study of Wikipedia in the Cloud

In this section, we apply the proposed approximation scheme on the case study of the Wikipedia system described in Section 2.1. The challenges immediately arising are: how to generate the request workload, how to emulate the dynamics of changing execution speed in the cloud, and how to parameterize the models by profiling systems of interest. In the following, we first describe our Wikipedia testbed setup, detail the procedure of parameterizations, and finally discuss the fitting results. As our derivation centers on a single server, i.e., $G/G/1/PS$, we focus on presenting the results for a single Wikipedia VM.

6.1 System Setup

To generate the user requests, we use Apache Jmeter [27], which allows to configure the number of parallel users, as well as, their request frequency via the option of "think time". The longer the think time between requests is, the lower the request frequency is. We set the number of users to 20 and vary the frequency of their requests, depending on the phase. In the profiling phase the load is set high to obtain the maximum sustainable execution speeds, whereas in the prediction phase the average arrival rate is 4 pages per second by setting the think times to be exponentially distributed with a mean of 5 seconds.

In our system, we assume that the effective core capacity of each Wikipedia VM varies between 1 and 0.8 cores due to underlying workload dynamics at the physical system. The transition time between each configuration is exponentially distributed with mean of 160 seconds. We emulate such a system by hosting a Wikipedia VM on an IBM private cloud, whose workload dynamics are (un)fortunately fairly stable, i.e., conservative and stable VM consolidation. Consequently, we proactively alternate the core capacity of the Wikipedia VM between 1 and 0.8 cores, following the aforementioned assumptions regarding workload dynamics. The overhead of switching core capacity of a VM is in the order of a couple of milliseconds. In a way, our system dynamics are well "controlled" without experiencing the wild dynamics of a real public cloud. Each run of our experiment lasts for a total of 10000 seconds.

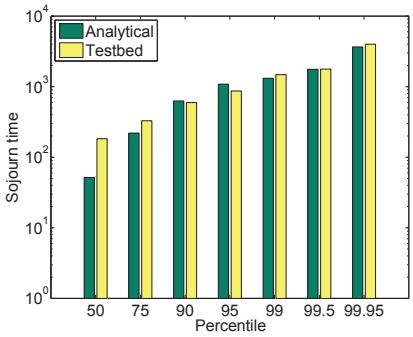


Fig. 4: Prediction of page request sojourn times of a Wikipedia VM hosted in the cloud.

6.2 Profiling and Parameterization

To apply the proposed approximation scheme, we first need to obtain the following key parameters: (1) the changing execution speeds in terms of number of pages/bytes (ϕ), (2) the average and standard deviation of page (job) sizes, (3) the average job (request) arrivals, and (4) the CTMC process of VM execution speeds.

To obtain the execution speeds, corresponding to different core capacities, we generate a high volume of requests by setting to zero the think time between requests and measure the resulting throughput. In steady state, the average number of pages executed are $\phi_1 = 4$ and $\phi_2 = 5$, relative to 0.8 and 1 cores, respectively. In terms of pages, the average size is $m = 74072$ bytes and the standard deviation is 62343. The resulting squared coefficient of variation, c^2, is around 0.71. The average measured arrival rates measure are $\lambda = 4$ pages per second, as how we generate the requests. Finally, we plug the controlled execution speeds dynamics into the generating metrics of CTMC, i.e., $\mathbf{Q} = [-1/160, 1/160; 1/160, -1/160]$. With the aforementioned parameters, one can see that the average execution speed is $\overline{\phi} = 4.5$ and the system load intensity is $\rho = 0.89$.

6.3 Prediction

We apply the proposed approximating scheme on the deployed Wikipedia VM, using the parameters listed above. For numerical stability, we normalize

the page sizes by a factor of 30. We summarize the prediction of sojourn times in Figure 4, listing the comparisons between the analysis and the measurement from the testbed for 50^{th}, 75^{th}, 90^{th}, 95^{th}, 99^{the}, 99.5^{th}, and 99.95^{th} percentiles of sojourn times. The fitting results are exceptionally accurate for the higher percentiles of sojourn times, though the lower percentiles, e.g., 50^{th} and 75^{th}, show a significant gap between the analytical and testbed results. This observation is indeed expected, as the proposed scheme particularly targets estimation of the higher percentiles.

We note that VMs hosted on other cloud systems can be subject to very different CTMC generating metrics and the parametrization of such non-controlled systems is even more challenging than our well "controlled" system. Nonetheless, our proposed profiling and approximation scheme can well capture the underlying complexity of real systems, i.e., multiple layers of software stacks executed on a virtual machines with oscillating execution speeds.

7 Related Work

In this section, we discuss the related work in two directions: the modeling work on predicting the tail response times for complex systems that show a great resemblance to real systems, and the optimization work that tries to mitigate the response time degradation due to exogenous variability in the cloud.

Modeling Tail Sojourn Times Given a vast amount of research on estimating the average sojourn times, obtaining the entire distribution of sojourn times is no mean feat for non-Markovian systems, in terms of arrival and service processes. An example of this are $M/G/1/PS$ queueing systems [10], [11], which are widely adopted to model various computing systems executing highly varying job sizes with a fixed speed in a processor sharing discipline. The introduction of cloud computing pinpoints another dimensionality of the modeling challenges, i.e., varying execution speed. The state-of-the-art deals with two causes of varying execution speeds: state-dependent and exogenous/environmental variability. The former [13], [12], [28] models the execution speed based on the current state of the system, i.e., the combination of the number of jobs and the multiprogramming levels. The latter [3], [14], [16], [17] models the transition of execution speeds as Markov-modulated speed for single queue and multiple queues.

In terms of the impact of arrival process, prior studies efficiently model the average response times of bursty workloads using a Markovian Arrival process [29], [30], particularly for multi-tier applications [31]. The Markov-modulated execution speed for single queues is discussed in [11], [15], [32]. While both Zhou et al. [33] and Boxma et al. [15] study two execution speeds, they employ different assumptions on when the changes of execution speed take place, i.e., only after the completion of job execution and during the job execution. Moreover, Boxma et al. consider an $M/G/1$ queue where the speed of the server alternates between two values with high speed periods having exponential distribution and low speed periods having a general distribution. Another related article is one by Massey [34], where the author focuses on deriving the queue process and the time-varying aspect, in particular an $M_t/M_t/1$ queueing system. In [14], the authors generalize the execution speed to any Markov process, and also the tail distribution of the workload in steady state. However, all the aforementioned analysis requires the arrival process to be Markovian.

There are a few studies that consider multiple queues with Markovian-modulated speeds. Dorsman et al. [17] obtain the heavy-traffic approximation for a steady distribution of workloads, queue lengths, and sojourn times for parallel queueing networks with Markov-modulated service speeds, by combining a functional central limit theorem approach and matrix-analytic methods. However, the impact of different traffic intensities and renewal arrivals are not considered there. To efficiently approximate the mean performance indexes, i.e., throughput and response time, Casale et al. [3] propose a generalize blending algorithm to model any number of execution speeds experienced by servers in the cloud. They are based on solving the ordinary differential [35] that define approximate transient analysis method for queueing network models. Their approach is limited to queueing networks with Markovian arrivals and Coxian jobs size distribution.

To the best of our knowledge, the tail distribution of sojourn times of many queue scenarios with execution speed modulated according to exogenous environmental processes and renewal arrivals are yet to be explored.

Optimizing for Tail Sojourn Times in Cloud Recognizing the importance and challenges of mitigating the performance variability in cloud computing, various reactive and proactive optimization strategies have been proposed. The reactive strategies center on obtaining a set of VMs that are of the same configurations but provide better

performance, i.e., faster execution speed. To such an end, Farley et al. [19] use testbed experiments whereas Björkqvist et al. [18] leverage simulation. Krebs et al. [36] quantify the performance isolation of cloud-based systems using different metrics. Kraft et al. [37] model the impact of workload consolidation on VM disk IO response times. Schad et al. [2] focus on the exogenous variability resulting from the underlying heterogeneous hardware and develop strategies to select VMs that are hosted on the same platform as to reduce the variability of the execution speeds. As for proactively mitigating the performance degradation, Björkqvist et al. [6] leverage a continuous Markovian model to capture the distribution of tail throughput and further optimize the cloud cluster that fulfills the target tail throughput at minimal cost. Sansottera et al. [38] provide a consolidated model that considers power, performance and reliability aspects when estimating the impact of hardware virtualization on the operational cost and performance of data centers. The model developed by Casale et al. [3] is meant to enable efficient exploration of the decision space for cloud deployments, but with focus on the average performance index.

In summary, the optimization strategies for tail response times fall short in providing SLA guarantees, i.e., they only promise best effort. Our study adopts the proactive approach to model various important aspects of cloud clusters and further optimize the cluster size so that the tail response times specified in SLAs are statistically guaranteed.

8 Conclusion

Motivated by the emerging challenge in capturing the tail sojourn times in cloud systems where workload and server execution speeds are highly varying, we develop an approximation of tail sojourn times for the $G/G/1/PS$ queueing system with Markov modulated execution speeds. We first derive the tail workload distribution and then develop the approximation algorithms based on $M/G/1/PS$ queueing systems. Using extensive simulation results, we show that our proposed analysis is particularly accurate for systems with highly varying job sizes, and large number of servers experiencing frequent execution speed changes. To make this study relevant to real applications, we further validate our analysis using a small Wikipedia testbed running on an IBM private cloud, and show promising results in capturing the tail sojourn times of a real system.

Acknowledgments

The research presented in this paper has been supported by the Swiss National Science Foundation (project 200021_141002), and partly by the EU Commission under the FP7 GENiC project (Grant Agreement No 608826).

References

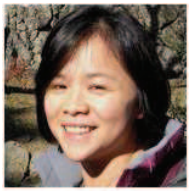
[1] Y. Xu, Z. Musgrave, B. Noble, and M. Bailey, "Bobtail: Avoiding Long Tails in the Cloud," in *USENIX NSDI*, 2013, pp. 329–341.
[2] J. Schad, J. Dittrich, and J.-A. Quiané-Ruiz, "Runtime measurements in the cloud: observing, analyzing, and reducing variance," *Proceedings of VLDB Endowment*, vol. 3, no. 1-2, pp. 460–471, 2010.
[3] G. Casale and M. Tribastone, "Modelling exogenous variability in cloud deployments," *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 4, pp. 73–82, 2013.
[4] Z. Ou, H. Zhuang, A. Lukyanenko, J. K. Nurminen, P. Hui, V. Mazalov, and A. Ylä-Jääski, "Is the same instance type created equal? exploiting heterogeneity of public clouds," *IEEE T. Cloud Computing*, vol. 1, no. 2, pp. 201–214, 2013.
[5] L. Y. Chen, D. Ansaloni, E. Smirni, A. Yokokawa, and W. Binder, "Achieving application-centric performance targets via consolidation on multicores: myth or reality?" in *ACM HPDC*, 2012, pp. 37–48.
[6] M. Björkqvist, S. Spicuglia, L. Y. Chen, and W. Binder, "QoS-Aware Service VM Provisioning in Clouds: Experiences, Models, and Cost Analysis," in *Service-Oriented Computing*. Springer, 2013, pp. 69–83.
[7] J. Dean and L. A. Barroso, "The tail at scale," *Communications of the ACM*, vol. 56, no. 2, pp. 74–80, Feb. 2013.
[8] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *ACM SoCC*, 2012, pp. 7:1–7:13.
[9] http://aws.amazon.com/ec2/.
[10] L. Kleinrock, *Queueing Systems*. Wiley Interscience, 1975, vol. I: Theory.
[11] N. Gautam, *Analysis of Queues: Methods and Applications*, 1st ed. CRC Press, Inc., 2012.
[12] K. M. Rege and B. Sengupta, "Sojourn time distribution in a multiprogrammed computer system," *AT&T technical journal*, vol. 64, no. 5, pp. 1077–1090, 1985.
[13] V. Gupta and M. Harchol-Balter, "Self-adaptive admission control policies for resource-sharing systems," *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 1, pp. 311–322, 2009.
[14] S. R. Mahabhashyam and N. Gautam, "On queues with Markov modulated service rates," *Queueing Systems*, vol. 51, no. 1-2, pp. 89–113, 2005.
[15] O. J. Boxma and I. A. Kurkova, "The M/G/1 queue with two service speeds," *Advances in Applied Probability*, vol. 33, pp. 520–540, 2001.
[16] B. Zhang and B. Zwart, "Fluid models for many-server Markovian queues in a changing environment," *Operations Research Letters*, vol. 40, no. 6, pp. 573–577, 2012.
[17] J.-P. Dorsman, M. Vlasiou, and B. Zwart, "Parallel queueing networks with Markov-modulated service speeds in heavy traffic," *ACM SIGMETRICS Performance Evaluation Review*, vol. 41, no. 2, pp. 47–49, 2013.
[18] M. Björkqvist, L. Y. Chen, and W. Binder, "Opportunistic service provisioning in the cloud," in *IEEE CLOUD*, 2012, pp. 237–244.
[19] B. Farley, A. Juels, V. Varadarajan, T. Ristenpart, K. D. Bowers, and M. M. Swift, "More for your money: exploiting performance heterogeneity in public clouds," in *ACM SoCC*, 2012, pp. 20:1–20:14.

[20] S. Spicuglia, L. Y. Chen, and W. Binder, "Join the best queue: Reducing performance variability in heterogeneous systems," in *IEEE Cloud*, 2013, pp. 139–146.
[21] Wikimedia project, "Wikimedia Grid Report," http://ganglia.wikimedia.org/latest.
[22] PHP, "http://php.net/."
[23] MediaWiki, "http://www.mediawiki.org/."
[24] MySQL, "http://www.mysql.com/."
[25] R. Birke, A. Podzimek, L. Y. Chen, and E. Smirni, "State-of-the-practice in data center virtualization: Toward a better understanding of VM usage," in *IEEE/IFIP DSN*, 2013, pp. 1–12.
[26] http://www.omnetpp.org/.
[27] A. Jmeter, "http://jmeter.apache.org/."
[28] J. Zhang and B. Zwart, "Steady state approximations of limited processor sharing queues in heavy traffic," *Queueing Systems*, vol. 60, no. 3-4, pp. 227–246, 2008.
[29] G. Casale, N. Mi, and E. Smirni, "Bound analysis of closed queueing networks with workload burstiness," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 36, no. 1, 2008, pp. 13–24.
[30] A. Sansottera, G. Casale, and P. Cremonesi, "Fitting second-order acyclic marked markovian arrival processes," in *IEEE/IFIP DSN*, 2013, pp. 1–12.
[31] N. Mi, G. Casale, L. Cherkasova, and E. Smirni, "Burstiness in multi-tier applications: Symptoms, causes, and new models," in *ACM/IFIP/USENIX Middleware*, 2008, pp. 265–286.
[32] M. Baykal-Gursoy and Z. Duan, "M/M/C queues with Markov modulated service processes," in *ACM ValueTools*, 2006, p. 38.
[33] Y.-P. Zhou and N. Gans, "A single-server queue with Markov modulated service times," 1999, http://fic.wharton.upenn.edu/fic/papers/99/9940.pdf.
[34] W. A. Massey, "The analysis of queues with time-varying rates for telecommunication models," *Telecommunication Systems*, vol. 21, pp. 173–204, 2002.
[35] T. G. Kurtz, "Solutions of ordinary differential equations as limits of pure Markov processes," *Journal of Applied Probability*, vol. 7, no. 1, pp. 49–58, 1970.
[36] R. Krebs, C. Momm, and S. Kounev, "Metrics and techniques for quantifying performance isolation in cloud environments," *Science of Computer Programming*, vol. 90, pp. 116–134, 2014.
[37] S. Kraft, G. Casale, D. Krishnamurthy, D. Greer, and P. Kilpatrick, "Io performance prediction in consolidated virtualized environments," in *ACM SIGSOFT Software Engineering Notes*, vol. 36, no. 5, 2011, pp. 295–306.
[38] A. Sansottera, D. Zoni, P. Cremonesi, and W. Fornaciari, "Consolidation of multi-tier workloads with performance and reliability constraints," in *HPCS*, 2012, pp. 74–83.

Mathias Björkqvist is a software engineer at IBM Zurich Research Lab, Zurich, Switzerland. He received a MSc. in Computer Science from the Helsinki University of Technology, Finland, in Dec 2007. His research interests include storage, security, and resource and service provisioning in clouds.

Natarajan Gautam is a Professor in the Department of Industrial and Systems Engineering at Texas A&M University with a courtesy appointment in the Department of Electrical and Computer Engineering. Prior to joining Texas A&M University in 2005, he was on the Industrial Engineering faculty at Penn State University for eight years. He received his M.S. and Ph.D. in Operations Research from the University of North Carolina at Chapel Hill, and his B.Tech. from Indian Institute of Technology, Madras. His research interests are in the areas of modeling, analysis and performance evaluation of stochastic systems with special emphasis on optimization and control in computer, communication and information systems. He is an Associate Editor for the INFORMS Journal on Computing, IIE Transactions, and OMEGA.

Robert Birke received his PhD degree from the Politecnico di Torino in 2009 with the telecommunications group under the supervision of professor Fabio Neri. Currently he works as a postdoc in the cloud server technologies group at IBM Zurich Research Lab, Zurich, Switzerland. He is a co-author of more than 40 scientific papers. His main research interests are high performance computing, cloud computing and datacenter networks with special focus on performance, quality of service, and virtualization. Dr. Birke is a IEEE and USENIX member.



Lydia Y. Chen is a research staff member at the IBM Zurich Research Lab, Zurich, Switzerland. She received a Ph.D. in Operations Research and Industrial Engineering from the Pennsylvania State University in Dec 2006. Her research interests include performance modeling in multi-core systems and power-workload management in data centers. She has served on several technical program committees in various performance and network conferences, including DSN, INFOCOM, Globecom, ICC, ICNC, and ICCCN, and has served as an organizer of the Data Center Performance (DCPerf) workshop in 2011-2013.

Walter Binder is an associate professor in the Faculty of Informatics, University of Lugano, Switzerland. He holds an MSc, a PhD, and a Venia Docendi from the Vienna University of Technology, Austria. Before joining the University of Lugano, he was a post-doctoral researcher at the Artificial Intelligence Laboratory, École Polytechnique Fédérale de Lausanne, Switzerland. His main research interests are in the areas of program analysis, virtual machines, parallel programming, and Cloud computing.