Analytic Modeling of Load Balancing Policies for Tasks with Heavy-tailed Distributions *

Alma Riska Evgenia Smirni Gianfranco Ciardo
Department of Computer Science
College of William and Mary
Williamsburg, VA 23187-8795, USA
e-mail {riska,esmirni,ciardo}@cs.wm.edu

ABSTRACT
We present an analytic technique for modeling load balancing policies on a cluster of servers conditioned on the fact that the service times of arriving tasks are drawn from heavy tail distributions. We propose a new modeling methodology for the exact solution of an $M/H_k/1$ server and illustrate its use for modeling two distinct load balancing policies in a distributed multi-server system. Our analytic results provide exact information regarding the distribution of task sizes that compose the waiting queue on each server and suggest an easy and inexpensive way to provide load balancing based on the sizes of the incoming tasks.

1. INTRODUCTION
We consider the resource allocation problem in a distributed multi-server system. We assume that tasks arrive to a front-end system, which is responsible for dispatching them to the back-end nodes. This happens according to a task scheduling policy that aims to route the request to the "best" back-end server, since a task can potentially be served by *any* server. Such a system can be considered as an abstraction of a distributed web server [7, 10, 19].

Balancing the load across the back-end servers is critical for performance [7]. In the past two decades, there has been a significant research effort in task scheduling and load balancing (see [11] and references therein). The basic assumption in much of this work is that the service demands of the various tasks are governed by an exponential distribution. In contrast to the above assumption, there is very strong evidence that the size of web documents, and accordingly their service demands, are governed by heavy-tailed distributions [3, 4, 1, 2]. As a consequence, load balancing in distributed servers must be re-examined.

A problem that plagues the analysis of load balancing policies via simulation is the heavy-tailed distribution of the task service times. Simulation becomes either very time-consuming, or at times even intractable. Consequently, analytic models become an attractive alternative.

In this paper, we illustrate the use of analytic models for the analysis of load balancing policies in distributed servers whose abstraction is illustrated in Figure 1. We consider two scheduling policies that do not use information about the load of each individual server for their scheduling decisions and are consequently easy to implement. We compare a random policy (i.e., a policy where the dispatcher allocates each new task to a randomly selected server), and a size-based policy (i.e., a policy that where the dispatcher allocates each new task to the servers based solely on the task's size). Size based policies have been shown to outperform even dynamic scheduling policies where incoming tasks are dispatched towards the least loaded host if the workload is governed by a bounded Pareto distribution [10]. Our modeling results further reinforces the belief that the heavier the tail of the distribution, the better the size-based policy performs.

The analysis of our model uses an extended version of the ETAQA approach that was recently developed for the study of quasi-birth-death (QBD) processes and a more complex case of M/G/1-type processes [5, 6]. ETAQA provides a solution that is very efficient both computation-wise and storage-wise in comparison to the classic solutions for QBD and M/G/1-type processes but its major drawback is the fact that it applies to

*This work has been supported by National Science Foundation under grands EIA-9974992 and EIA-9977030, and by the National Aeronautics and Space Administration under NASA Grant NAG-1-2168.

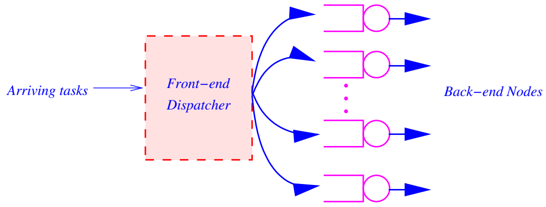

Figure 1: Model of a distributed server.

a very restricted family of processes for which "returns" from a higher level of states to the immediate lower level are always directed toward a single state only. Effectively, this means that ETAQA can apply when the matrix **B** (i.e., the matrix that represents the returns from a higher level of states to the immediate lower level) is a single column matrix.

Here, we extend ETAQA by allowing the matrix **B** to be a full matrix consisting of non-zero columns that are multiples of one (any) column. This allows us to solve M/H$_k$/1-type queues exactly (i.e., queues with Markovian arrivals and k-stage hyper-exponential servers) using ETAQA. This particular result is of great importance in the area of modeling of queues where the customer demands are drawn by heavy-tailed distributions, since such distributions can be closely approximated by hyper-exponential distributions [9].

Using real trace data that show the file distribution of the web server for all requests for the 1998 World Soccer Cup, we apply fitting techniques [9] to extract a hyper-exponential distribution that best approximates the actual data distribution. Then, using ETAQA, we study the behavior of the random and the size-based policy. ETAQA allows for the individual calculation of the contribution to the queue length (or to any of its higher moments) due to each individual "phase" of the server. Thus, we can perform a detailed study that quantifies the effect of the task sizes on queue build-up for the various servers. This allows us to argue about the effectiveness of a size-based policy.

This paper is organized as follows. In Section 2 we present the workload and the various steps we follow in order to obtain a hyper-exponential distribution that closely approximates the workload behavior. In Section 3 we extend ETAQA to obtain the exact solution of M/H$_k$/1-type queues. Section 4 presents the performance comparisons of the two load balancing policies. Section 5 concludes the paper and outlines future work.

2. THE WORKLOAD

The workload used in our analysis is obtained from web server traces. Since we evaluate different load balancing policies using analytic techniques, we characterize the data with a hyper-exponential distribution. In this section we describe the workload used and the fitting steps that allow us to derive a hyper-exponential distribution that closely matches the web server trace data.

We obtained workload traces of the 1998 World Soccer Cup Web site[1]. The World Cup site server was composed of 30 low-latency platforms distributed across four physical locations. Client requests were dispatched to a location via a Cisco Distributed Director, and each location was responsible for load balancing incoming requests among its available servers.

The traces provide information about each request received by each server. For each request the following information is recorded: the IP address of the client issuing the request, the date and time of the request, the URL requested, the HTTP response status code, and the content length (in bytes) of the transferred document. Trace data were collected for each day during the total period of time that the web server was operational. Since the focus of this work is on load balancing, irrespective of possible caching policies at the server, we only extracted the content length of the transfered document from each trace record assuming that the service time of each request is a function of the size of the requested document. For a detailed analysis of the World Cup workload see [2].

2.1 Fitting a day's data into a distribution

It is a well-known fact that the sizes of web server requests are highly variable and are best described by heavy-tail distributions. To check for the heavy-tail property, we used Boston University's `aest` tool that verifies and estimates the heavy-tail portion of a distribution [8][2]. Using the scaling estimator methodology, the tool helps identify the portion of the data set that exhibits power-law behavior by demonstrating graphically the tail of the distribution where the heavy-tailed behavior is present. The selection of the point where the power-law behavior starts is significant because it affects the computation of the parameters of the distribution. Figure 2 shows the results of the scaling analysis for a representative day of the dataset, day 80. Considering the tail portion of the plots, for requests larger than 1 MByte, we see that they are close to linear, suggesting that the heavy-tailed portion of the dataset begins at around 1 MByte. Based on this observation, we conclude that the original distribution is best approximated by a hybrid model that combines a lognormal distribution for the body of the data and a power-law distribution for its tail [2].

After identifying the two portions of the workload, we

[1] http://researchsmp2.cc.vt.edu/cgi-bin/reposit/search.pl?details=YES&detailsoffset=135.
[2] http://www.cs.bu.edu/faculty/crovella/aest.html.

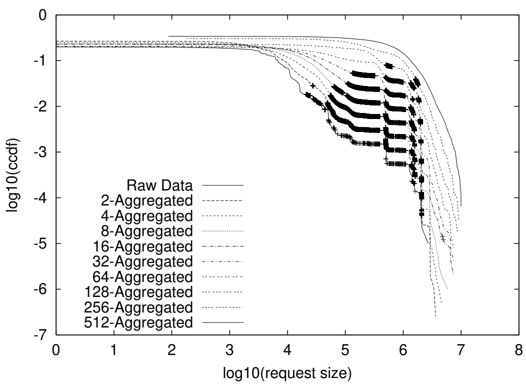

Figure 2: Tail characterization for day 80. The various curves in the figure show the complementary cumulative distribution function (ccdf) of the dataset on a log-log scale for successive aggregations of the dataset by factors of two. The figure illustrates that the shape of the tail (i.e., for size $> 10^6$) is close to linear and suggests the parameter for its power-law distribution. The '+' signs on the plot indicate the points used to compute the α in the distribution. The elimination of points for each successive aggregation indicates the presence of a heavy tail.

need to compute the parameters of each of its portions. The body of the distribution is considered lognormal with probability distribution function (p.d.f):

$$f(x) = \frac{1}{bx\sqrt{2\pi}} \exp\left(\frac{-(\ln x - a)^2}{2b^2}\right)$$

We compute $b > 0$ (i.e., the shape parameter), and $a \in (-\infty, \infty)$ (i.e., the scale parameter) using the maximum likelihood estimators [14]:

$$\hat{a} = \frac{\sum_{i=1}^{n} \ln X_i}{n}, \quad \hat{b} = \left[\frac{\sum_{i=1}^{n} (\ln X_i - \hat{a})^2}{n}\right]^{\frac{1}{2}}$$

where X_i for $1 \le i \le n$ are the sample data.

The workload is heavy-tailed with tail index α if its complementary distribution function is:

$$P[X > x] \sim x^{-\alpha}, \quad x \to \infty, \quad 0 < \alpha < 2$$

where X is the random variable describing the request size. In our study, we compute α via the `aest` tool.

2.2 Fitting the distribution into a mix of exponentials

Our second step is to apply Feldmann and Whitt's algorithm [9] for approximating a heavy-tailed distribution with a hyper-exponential distribution. Since both the body and the tail of our distributions have a heavy-tail component, we apply the algorithm to each component

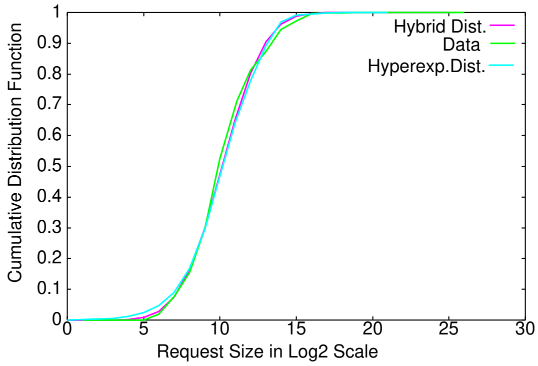

Figure 3: Fitted data of day 57.

separately and finally combine both distributions into a hyper-exponential one using a weighted sum. The weights for combining the two hyper-exponential distributions corresponding to the lognormal and the power-law portions of the original data are given by the probability that a request is for a file for size less or equal to, or greater than, 1 Mbyte, respectively, as computed from the empirical data.

The Feldmann-Whitt algorithm attempts to fit various regions of the distribution with exponential components in a recursive manner. At each step, the fitted exponential component is subtracted from the distribution, such that each component focuses on a specific portion of the random variable values, increasingly closer to 0. If there are enough exponential components, the algorithm manages to closely approximate a heavy-tail distribution in the area of primary interest. The algorithm output is then a hyper-exponential distribution H_k with probability distribution function

$$h(x) = \sum_{i=1}^{k} \beta_i \mu_i e^{-\mu_i x}, \quad x \ge 0.$$

We point the interested reader to [9] for a detailed description of the algorithm and its fitting accuracy.

2.3 Fitting examples

We use the Feldmann-Whitt algorithm to fit the data from two representative days, day 57 and day 80 into hyper-exponential distributions. Figures 3, 4 illustrate the cumulative distribution of the actual data, their fitting into a hybrid distribution (lognormal for the body and power-law for the tail), and the fitting of the hybrid distribution into a hyper-exponential one. We observe that the resulting hyper-exponential distribution closely matches the behavior of the original data.

Tables 1 and 2 illustrate the parameters of the lognormal and the power-law portions of the distribution for each day. For both days, we see that the bulk of the data

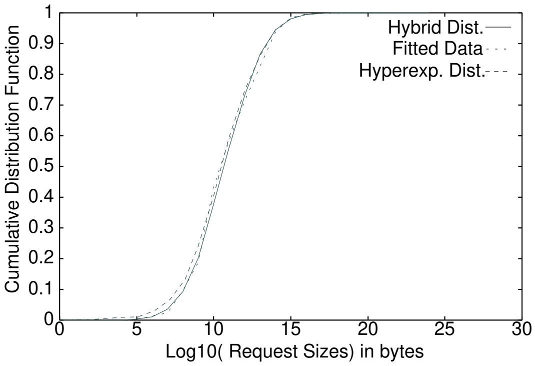

Figure 4: Fitted data of day 80.

lies in the lognormal portion of the workload, while only a very small part (albeit with very large file sizes) lies in the power-law part. Tables 1 and 2 also show the parameters that the Feldmann-Whitt algorithm suggests for the fitting of the above distributions into hyper-exponentials. In both cases, a total of seven exponential phases, four for the lognormal portion and three for the power-law portion, are sufficient to achieve an excellent approximation of the original data.

Data from Day 57		
Lognormal(a,b)	Power(α)	Weight for
$a = 7.033358$ $b = 1.509296$	$\alpha = 0.82$	Lognormal 0.99935
Parameters of the $H_7(\mu_i, \beta_i : 1 \le i \le 7)$ fitting		
μ_i	β_i	
0.0000000084695	0.0000000004383	
0.0000001060318	0.0000000023312	
0.0000113172652	0.0006499972304	
0.0000015100478	0.0000147002669	
0.0000189146863	0.0144507953832	
0.0001900075398	0.4437013666789	
0.0012356156678	0.5411831376710	

Table 1: Workload parameters for day 57.

As our purpose is to analyze load balancing policies according to workload variability, we also considered a synthetic workload that exhibits different variability characteristics with respect to one of the selected days, namely day 57. Since the power-tail portion of each of the selected days is very small, we turn our attention to the lognormal portion which also exhibits a heavy-tail behavior. By changing simultaneously both the a and b parameters of a lognormal distribution, we change both the scale and shape of the distribution so as to vary the variance of the distribution and at the same type keep the mean of the distribution constant (and equal to the mean of day 57, i.e., 3629 Bytes. This way, we can examine the sensitivity of our load balancing policies to

Data from Day 80		
Lognormal(a,b)	Power(α)	Weight for
$a = 7.43343$ $b = 1.42824$	$\alpha = 0.89$	Lognormal 0.999977
Parameters of the $H_7(\mu_i, \beta_i : 1 \le i \le 7)$ fitting		
μ_i	β_i	
0.0000000137089	0.0000000001905	
0.0000002156677	0.0000000015697	
0.0000432933235	0.0005699982398	
0.0000053128803	0.0007440620149	
0.0000296466848	0.0393217234927	
0.0001509998638	0.3679160877122	
0.0008702852306	0.5914481267802	

Table 2: Workload parameters for day 80.

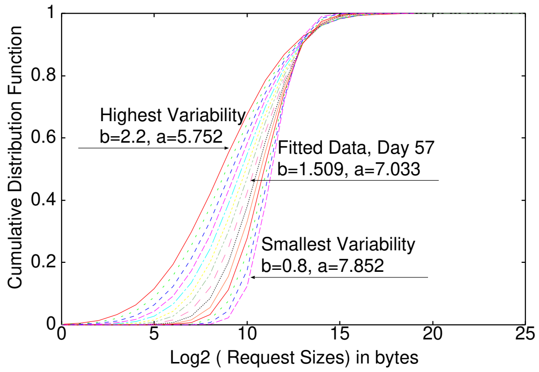

Figure 5: Shape of the cdf when changing the variability of the lognormal portion of the workload.

the workload variability. Figure 5 illustrates how the cdf of the distribution changes when we change the workload variability (the fitting technique we use resulted in three, four, or five stages for the lognormal fitting, thus a total of six, seven, or eight stages were used to fit the mixture of the lognormal and power-law distribution). We will return to the issue of policy sensitivity analysis as a function of workload variability in Section 4.

3. ANALYSIS OF M/H$_K$/1 QUEUES

After performing the fitting steps outlined in the previous section, we obtain a hyper-exponential distribution that closely describes the workload. Therefore, we can model each server as an M/H$_k$/1 queue. In this section, we describe a new methodology for the *exact* solution of such queues.

The continuous time Markov chain (CTMC) modeling the behavior of an M/H$_k$/1 queue [12, page 143] has a matrix geometric form [15, 16]. This implies that its state space can be partitioned into the "boundary" set of states $\mathcal{S}^{(0)} = \{s_1^{(0)}, \ldots, s_m^{(0)}\}$, of size m, and the "repetitive" sets of states $\mathcal{S}^{(j)} = \{s_1^{(j)}, \ldots, s_n^{(j)}\}$, for $j \geq 1$, each of size n. The infinitesimal generator can accordingly be block partitioned as:

$$\mathbf{Q} = \begin{bmatrix} \hat{\mathbf{L}} & \hat{\mathbf{F}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \hat{\mathbf{B}} & \mathbf{L} & \mathbf{F} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{B} & \mathbf{L} & \mathbf{F} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{B} & \mathbf{L} & \mathbf{F} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \quad (1)$$

(we use the letter "L", "F", and "B" according to whether the matrices describe "local", "forward", and "backward" transition rates, respectively, and we use a "ˆ" for matrices related to $\mathcal{S}^{(0)}$). The repetitive structure of the infinitesimal generator of quasi-birth-death (QBD) processes, i.e., processes with an infinitesimal generator illustrated in (1) allows for a recursive formulation of the stationary probabilities for the CTMC states that facilitates their computation. A significant body of research concentrates on the solution of QBD processes with matrix geometrix form, with most prominent the works of Neuts [16] and Latouche and Ramaswami [13]. Neuts [16] proposed an algorithm that takes advantage of the repetitive structure of a QBD process. If $\boldsymbol{\pi}^{(j)}$ is the stationary probability vector for the states in $\mathcal{S}^{(j)}$, the values of $\boldsymbol{\pi}^{(j)}$, $j \geq 2$ have a geometric form:

$$\boldsymbol{\pi}^{(j)} = \boldsymbol{\pi}^{(1)} \cdot \mathbf{R}^{j-1}, \quad j \geq 2$$

where $\mathbf{R}$ is the *rate matrix*, solution of the quadratic equation

$$\mathbf{F} + \mathbf{R} \cdot \mathbf{L} + \mathbf{R}^2 \cdot \mathbf{B} = \mathbf{0},$$

and can be obtained numerically using an iterative procedure. Another matrix that can greatly facilitate the computation of the probability distribution of all states in QBDs is $\mathbf{G}$ [13], solution of:

$$\mathbf{B} + \mathbf{L} \cdot \mathbf{G} + \mathbf{F} \cdot \mathbf{G}^2 = \mathbf{0}$$

which implies

$$\mathbf{G} = -(\mathbf{L} + \mathbf{F} \cdot \mathbf{G})^{-1} \cdot \mathbf{B}.$$

Both $\mathbf{R}$ and $\mathbf{G}$ have important probabilistic interpretations. $\mathbf{G}$ expresses the probability of first entering $\mathcal{S}^{(j-1)}$ through each of its states, starting from each state $\mathcal{S}^{(j)}$. $\mathbf{R}$ records the expected number of visits to each state in $\mathcal{S}^{(j)}$, starting from each state in $\mathcal{S}^{(j-1)}$, before reentering $\mathcal{S}^{(j-1)}$. The following relation between matrices $\mathbf{R}$ and $\mathbf{G}$ holds [13, pages 137-8]:

$$\mathbf{R} = -\mathbf{F} \cdot (\mathbf{L} + \mathbf{F}\mathbf{G})^{-1}$$

from which we derive the fundamental relation

$$\mathbf{R} \cdot \mathbf{B} = \mathbf{F} \cdot \mathbf{G}. \quad (2)$$

Obtaining $\mathbf{R}$ is usually computationally intensive in general (see [13] for an overview of alternative methods to compute $\mathbf{R}$), but knowing $\mathbf{G}$ can greatly facilitate the computation of $\mathbf{R}$ [18].

In [5], ETAQA, an alternative methodology for the solution of QBD processes that avoids using the matrices $\mathbf{R}$ and $\mathbf{G}$ was introduced. ETAQA does not operate on the steady state distribution of *all* states in the chain, not even in implicit recursive form, but rather on the exact *aggregate* probabilities of classes of states defined by partitioning the state space. Even if it only computes aggregate probabilities, ETAQA allows to easily obtain measures of interest such as queue length or any of its higher moments.

The major drawback of ETAQA as presented in [5] is that it applies to a restricted family of QBDs, for which transition from states in $\mathcal{S}^{(j)}$ to the immediately preceding set $\mathcal{S}^{(j-1)}$ can only be directed toward a single special "return" state in $\mathcal{S}^{(j-1)}$. This effectively imposes a special structure on the matrix $\mathbf{B}$: all of its columns must be zero except for the one corresponding to the special return state.

Given ETAQA's original restriction, it is immediately obvious that it cannot be used as-is for the solution of M/H$_k$/1 queues since, in this case, the matrix $\mathbf{B}$ is full, albeit with a special structure: its columns are multiples of each other. This implies that $\mathbf{B} = \boldsymbol{\mu} \cdot \boldsymbol{\beta}$, where $\boldsymbol{\mu}$ is a column vector and $\boldsymbol{\beta}$ is a row vector of k elements each. The elements of $\boldsymbol{\mu}$ and $\boldsymbol{\beta}$ are the parameters of the hyper-exponential service time distribution H$_k$, i.e., the rate of the exponential stages and their respective probabilities. This implies that $\boldsymbol{\beta} \cdot \mathbf{1}^T = 1$ and, coupled with the fact that $\mathbf{B}$ can be expressed as the product of two vectors, also implies that $\mathbf{G}$ can be explicitly obtained and is equal to [18]:

$$\mathbf{G} = \mathbf{1} \cdot \boldsymbol{\beta}. \quad (3)$$

Using (3) we can extend ETAQA to the case of M/H$_k$/1 queues, as shown in the next section.

3.1 ETAQA for M/H$_k$/1 queues

Partitioning the stationary probability vector according to the set of states $\mathcal{S}^{(0)}$, and $\mathcal{S}^{(i)}$ for $i \geq 1$ defined in section 3 we get:

$$\boldsymbol{\pi} = \left[\boldsymbol{\pi}^{(0)}, \boldsymbol{\pi}^{(1)}, \boldsymbol{\pi}^{(2)}, \cdots\right]$$

with $\boldsymbol{\pi}^{(0)} \in \mathbb{R}^m$ and $\boldsymbol{\pi}^{(1)}, \boldsymbol{\pi}^{(2)}, \cdots \in \mathbb{R}^n$. Furthermore we modify the infinitesimal generator into $\mathbf{Q}'$ such that $\mathbf{Q} = \mathbf{Q}'$ and $\mathbf{Q}'$ is defined as:

$$\mathbf{Q}' = \begin{bmatrix} \hat{\mathbf{L}} & \hat{\mathbf{F}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \hat{\mathbf{B}} & \mathbf{L} & \mathbf{F} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{B}+\mathbf{S} & \mathbf{L} & \mathbf{F}+\mathbf{S} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{S} & \mathbf{B} & \mathbf{L} & \mathbf{F}+\mathbf{S} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \quad (4)$$

where $\mathbf{S} = \mathbf{F}\cdot\mathbf{G} - \mathbf{F}\cdot\mathbf{G} = \mathbf{0}$. We can write $\boldsymbol{\pi}\cdot\mathbf{Q}' = \mathbf{0}$ as:

$$\begin{cases} \boldsymbol{\pi}^{(0)}\cdot\hat{\mathbf{L}} + \boldsymbol{\pi}^{(1)}\cdot\hat{\mathbf{B}} = \mathbf{0} \\ \sum_{i=2}^{\infty}\boldsymbol{\pi}^{(i)}\cdot\mathbf{S} + \boldsymbol{\pi}^{(0)}\cdot\hat{\mathbf{F}} + \boldsymbol{\pi}^{(1)}\cdot\mathbf{L} + \boldsymbol{\pi}^{(2)}\cdot\mathbf{B} = \mathbf{0} \\ \boldsymbol{\pi}^{(1)}\cdot\mathbf{F} + \boldsymbol{\pi}^{(2)}\cdot\mathbf{L} + \boldsymbol{\pi}^{(3)}\cdot\mathbf{B} = \mathbf{0} \\ \boldsymbol{\pi}^{(2)}\cdot\mathbf{S} + \boldsymbol{\pi}^{(2)}\cdot\mathbf{F} + \boldsymbol{\pi}^{(3)}\cdot\mathbf{L} + \boldsymbol{\pi}^{(4)}\cdot\mathbf{B} = \mathbf{0} \\ \boldsymbol{\pi}^{(3)}\cdot\mathbf{S} + \boldsymbol{\pi}^{(3)}\cdot\mathbf{F} + \boldsymbol{\pi}^{(4)}\cdot\mathbf{L} + \boldsymbol{\pi}^{(5)}\cdot\mathbf{B} = \mathbf{0} \\ \vdots \end{cases} . \quad (5)$$

We are going to show how to derive $m+2n$ equations in $\boldsymbol{\pi}^{(0)}$, $\boldsymbol{\pi}^{(1)}$, and a new vector of n unknowns, $\boldsymbol{\pi}^{(*)} = \sum_{i=2}^{\infty}\boldsymbol{\pi}^{(i)}$, representing the stationary probability of being in the macro-state $\{s_j^{(i)} : i \geq 2\}$, for $1 \leq j \leq n$. First, however, observe that, since the matrix-geometric property holds, $\boldsymbol{\pi}^{(j+1)} = \boldsymbol{\pi}^{(j)}\cdot\mathbf{R}$ for $j \geq 1$. Using (2) we obtain

$$\boldsymbol{\pi}^{(j+1)}\cdot\mathbf{B} = \boldsymbol{\pi}^{(j)}\cdot\mathbf{R}\cdot\mathbf{B} = \boldsymbol{\pi}^{(j)}\cdot\mathbf{F}\cdot\mathbf{G}. \;\; j \geq 1 \quad (6)$$

Summing (6) over all $j \geq 2$ we obtain

$$\sum_{j=3}^{\infty}\boldsymbol{\pi}^{(j)}\cdot\mathbf{B} = \sum_{j=2}^{\infty}\boldsymbol{\pi}^{(j)}\cdot\mathbf{F}\cdot\mathbf{G}$$

which implies

$$\boldsymbol{\pi}^{(*)}\cdot\mathbf{F}\cdot\mathbf{G} = \sum_{j=3}^{\infty}\boldsymbol{\pi}^{(j)}\cdot\mathbf{B}.$$

or alternatively that

$$\boldsymbol{\pi}^{(*)}\cdot\mathbf{F}\cdot\mathbf{G} + \boldsymbol{\pi}^{(2)}\cdot\mathbf{B} = \boldsymbol{\pi}^{(*)}\cdot\mathbf{B}.$$

- The first row in (5) provides m equations:

$$\boldsymbol{\pi}^{(0)}\cdot\hat{\mathbf{L}} + \boldsymbol{\pi}^{(1)}\cdot\hat{\mathbf{B}} = \mathbf{0}. \quad (7)$$

- The second row in (5) provides n equations:

$$\boldsymbol{\pi}^{(0)}\cdot\hat{\mathbf{F}} + \boldsymbol{\pi}^{(1)}\cdot\mathbf{L} - \boldsymbol{\pi}^{(*)}\cdot\mathbf{F}\cdot\mathbf{G} + \boldsymbol{\pi}^{(*)}\cdot\mathbf{F}\cdot\mathbf{G} + \boldsymbol{\pi}^{(2)}\cdot\mathbf{B} =$$
$$\boldsymbol{\pi}^{(0)}\cdot\hat{\mathbf{F}} + \boldsymbol{\pi}^{(1)}\cdot\mathbf{L} + \boldsymbol{\pi}^{(*)}\cdot(\mathbf{B} - \mathbf{F}\cdot\mathbf{G}) = \mathbf{0}. \quad (8)$$

- If we sum all the remaining equations in (5), we obtain:

$$\begin{aligned} &\boldsymbol{\pi}^{(1)}\cdot\mathbf{F} + \boldsymbol{\pi}^{(*)}\cdot(\mathbf{L} + \mathbf{F} + \mathbf{S}) + \sum_{j=3}^{\infty}\boldsymbol{\pi}^{(j)}\cdot\mathbf{B} = \\ &\boldsymbol{\pi}^{(1)}\cdot\mathbf{F} + \boldsymbol{\pi}^{(*)}\cdot(\mathbf{L} + \mathbf{F} + \mathbf{F}\cdot\mathbf{G}) + \\ &\sum_{j=3}^{\infty}\boldsymbol{\pi}^{(j)}\cdot\mathbf{B} - \boldsymbol{\pi}^{(*)}\mathbf{F}\cdot\mathbf{G} = \\ &\boldsymbol{\pi}^{(1)}\cdot\mathbf{F} + \boldsymbol{\pi}^{(*)}\cdot(\mathbf{L} + \mathbf{F} + \mathbf{F}\cdot\mathbf{G}) = \mathbf{0} \end{aligned} \quad (9)$$

Equations (7), (8), and (9) can be collectively written in matrix form as:

$$[\boldsymbol{\pi}^{(0)}, \boldsymbol{\pi}^{(1)}, \boldsymbol{\pi}^{(*)}]\cdot\overline{\mathbf{Q}} = [\mathbf{0}], \quad (10)$$

where

$$\overline{\mathbf{Q}} = \left[\begin{array}{c|c|c} \hat{\mathbf{L}} & \hat{\mathbf{F}} & \mathbf{0} \\ \hline \hat{\mathbf{B}} & \mathbf{L} & \mathbf{F} \\ \hline \mathbf{0} & \mathbf{B} - \mathbf{F}\cdot\mathbf{G} & \mathbf{L} + \mathbf{F} + \mathbf{F}\cdot\mathbf{G} \end{array}\right]. \quad (11)$$

The following theorem states that the rank of $\overline{\mathbf{Q}}$ is $m+2n-1$, implying that (10) is sufficient to compute $\boldsymbol{\pi}^{(0)}$, $\boldsymbol{\pi}^{(1)}$, and $\boldsymbol{\pi}^{(*)}$, if we also take into account the normalization constraint

$$\boldsymbol{\pi}^{(0)}\cdot\mathbf{1}^T + \boldsymbol{\pi}^{(1)}\cdot\mathbf{1}^T + \boldsymbol{\pi}^{(*)}\cdot\mathbf{1}^T = 1.$$

THEOREM 3.1. *Given an ergodic CTMC with infinitesimal generator* $\mathbf{Q}$ *having the structure shown in (1), the rank of matrix* $\overline{\mathbf{Q}}$ *defined in (11) is* $m+2n-1$.

Proof. *The proof is based on the fact that* $\mathbf{Q}'$ *is an infinitesimal generator, and its colums are linearly independent, except one of them. We use all the columns of* $\mathbf{Q}'$ *in our proof, and obtain in this way* $\overline{\mathbf{Q}}$. *The first* $m+n$ *columns of* $\overline{\mathbf{Q}}$ *are the first* $m+n$ *columns of* $\mathbf{Q}'$. *The last* n *columns of* $\overline{\mathbf{Q}}$ *are linear combination of the rest of the columns in* $\mathbf{Q}'$. *So, the rank of matrix* $\overline{\mathbf{Q}}$ *is* $m+2n-1$. *We need to drop any of its columns and substitute it with a column of* 1s *in order to obtain a unique solution for the our stationary probability vector* $[\boldsymbol{\pi}^{(0)}, \boldsymbol{\pi}^{(1)}, \boldsymbol{\pi}^{(*)}]$

We stress that, in the special case of M/H$_k$/1 queues, we can explicitly obtain $\mathbf{G}$ using (3). Indeed, we only need to store vector $\boldsymbol{\beta}$ to fully capture $\mathbf{G}$.

3.2 Measures of interest

Knowing the aggregated steady state probability vectors allows us to compute a rich collection of measures of interest. A detailed description on how to derive them can be found in [5]. Here, we only summarize these results. By writing the measure of interest r as

$$r = \boldsymbol{\pi}^{(0)}\cdot\boldsymbol{\rho}^{(0)T} + \boldsymbol{\pi}^{(1)}\cdot\boldsymbol{\rho}^{(1)T} + \sum_{j=2}^{\infty}\boldsymbol{\pi}^{(j)}\cdot\boldsymbol{\rho}^{(j)T}$$

(where $\boldsymbol{\rho} = [\boldsymbol{\rho}^{(0)}, \boldsymbol{\rho}^{(1)}, \cdots]$ are the *reward rates* for the states in $\mathcal{S}^{(0)}, \mathcal{S}^{(1)}, \cdots$), the definition of $\boldsymbol{\rho}$ is only restricted by our need to compute the above summation. Assuming the reward rate of state $s_i^{(j)}$, for $j \geq 2$ and $i = 1, \ldots, n$, is a polynomial of degree k in j with arbitrary coefficients $a_i^{[0]}, a_i^{[1]}, \ldots, a_i^{[k]}$:

$$\forall j \geq 2,\ \forall i \in \{1, 2, \ldots, n\}, \quad \rho_i^{(j)} = a_i^{[0]} + a_i^{[1]}j + \cdots + a_i^{[k]}j^k.$$

we obtain

$$\sum_{j=2}^{\infty} \boldsymbol{\pi}^{(j)} \cdot \boldsymbol{\rho}^{(j)T} = \sum_{j=2}^{\infty} \boldsymbol{\pi}^{(j)} \cdot \left(\mathbf{a}^{[0]} + \mathbf{a}^{[1]} j + \cdots + \mathbf{a}^{[k]} j^k\right)^T$$

$$= \sum_{j=2}^{\infty} \boldsymbol{\pi}^{(j)} \cdot \mathbf{a}^{[0]T} + \cdots + \sum_{j=2}^{\infty} j^k \boldsymbol{\pi}^{(j)} \cdot \mathbf{a}^{[k]T}$$

$$= \mathbf{r}^{[0]} \cdot \mathbf{a}^{[0]T} + \cdots + \mathbf{r}^{[k]} \cdot \mathbf{a}^{[k]T},$$

where $\mathbf{r}^{[l]} = \sum_{j=2}^{\infty} j^l \boldsymbol{\pi}^{(j)}$ for $l = 0, \ldots, k$, and its computation can be illustrated by strong induction. For the base case, $\mathbf{r}^{[0]}$ is simply $\boldsymbol{\pi}^{(*)}$. For $k > 0$, $\mathbf{r}^{[k]}$ can be computed by solving the system of n linear equations:

$$\begin{cases} \mathbf{r}^{[k]} \cdot (\mathbf{L} + \mathbf{F} + \mathbf{B})_{1:n,1:n-1} & = & \mathbf{b}_{1:n-1} \\ \mathbf{r}^{[k]} \cdot (\mathbf{F} - \mathbf{B}) \cdot \mathbf{1}^T & = & c \end{cases} \tag{12}$$

where

$$\mathbf{b} = -\left(2^k \boldsymbol{\pi}^{(0)} \cdot \hat{\mathbf{F}} + 2^k \boldsymbol{\pi}^{(1)} \cdot \mathbf{L} + 3^k \boldsymbol{\pi}^{(1)} \cdot \mathbf{F} + \sum_{l=1}^{k} \binom{k}{l} \left(2^l \mathbf{r}^{[k-l]} \cdot \mathbf{F} + \mathbf{r}^{[k-l]} \cdot \mathbf{L}\right)\right)$$

$$c = -2^k \boldsymbol{\pi}^{(1)} \cdot \mathbf{F} \cdot \mathbf{1}^T - \sum_{l=1}^{k} \binom{k}{l} \mathbf{r}^{[k-l]} \cdot \mathbf{F} \cdot \mathbf{1}^T$$

In practice, to obtain the system queue length we need to solve a linear system in n unknowns, and to compute its k^{th} moment we must solve k linear systems in n unknowns each.

4. ANALYSIS OF LOAD BALANCING POLICIES

We consider the following model of a distributed server environment. We assume a fixed number c of hosts with the same processing power, each serving tasks in first-come-first-serve order. We further assume that each host has an unbounded queue. Tasks arrive to the dispatcher from the outside world according to a Poisson process. The dispatcher is responsible for distributing the jobs among the various back-end servers according to a scheduling policy. We also assume that the dispatcher can derive the request duration (the size of the file) from the name of the file requested. We consider the following two load balancing policies (in neither case the dispatcher uses feedback from the individual hosts to better balance the load among them):

Random: The dispatcher assigns the incoming job to a randomly selected host, with probability $1/c$. The performance of this policy has been shown to be very similar that of round-robin, i.e., a policy that assigns jobs to hosts in cyclical fashion [10].

Size-based: The dispatcher assigns tasks to hosts according to the tasks' size. This policy is motivated by the desire to separate large from small tasks, to avoid the significant slowdowns that small tasks would experience when queued behind large tasks. A policy based on the same principle has been examined in [10] and compared very favorably to a dynamic policy where the dispatcher assigns tasks to hosts according to the hosts' load at the time of task arrival.

We first consider the performance of the random policy under the synthetic workload described in Subsection 2.3, recalling that one of the curves in Figure 5 corresponds to the actual fitted data from day 57 from the World Cup 98 trace data. In all cases, we assume that the overall arrival rate to the dispatcher is λ, and that there are eight hosts. Thus, the arrival rate to each individual host with the random policy is $\lambda/8$.

The average task slowdown for the random policy is illustrated in Figure 6(a). Although the system saturates at the same value of λ regardless of the workload choice (recall that all workloads have the same mean task size), the average task slowdown differs dramatically from workload to workload, especially in the range of medium-to-high system utilization. Figure 6(b) illustrates the average queue length at each host as a function of the workload variability for various arrival rates[3]. The figure further confirms that the higher the workload variability, the more dramatic the average queue build-up is.

To further understand the behavior of the system, we look closer at the range of task sizes that contribute to the queue build-up and consequently to the performance degradation. Our analytical model allows us to further explore the system behavior by analyzing how the queue length builds up. Figure 7 depicts the CTMC that models a host, an $M/H_7/1$ server (for presentation clarity, not all arcs are illustrated in the picture, but the reader can visualize the shape of the Markov chain and most importantly identify the parts of the CTMC that correspond to the power-law portion of the workload and the lognormal portion of the workload).

As described in Section 3, our analytic model allows the *exact* computation of the system queue length that corresponds to the different portions of the distribution. Figure 8 illustrates the contribution to the overall queue length from the power-law portion of the workload and from all tasks (files) greater than 100 KBytes (i.e., the phase of the lognormal distribution with large

[3]For presentation clarity, the x-axis of Figure 6(b) shows only the value of the b parameter of the lognormal distribution for the workload but we remind the reader that a different value of b implies also a different value of a, to keep the same mean task size across all workloads.

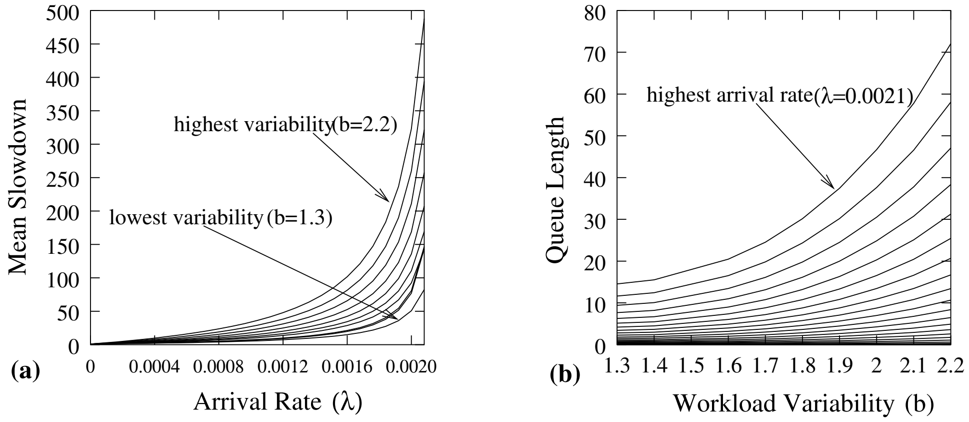

Figure 6: Average task slowdown as a function of the overall task arrival rate λ for workloads with high-to-low variance in their task service time (a), and average queue length as a function of the workload variability b for various arrival rates of the workload (b).

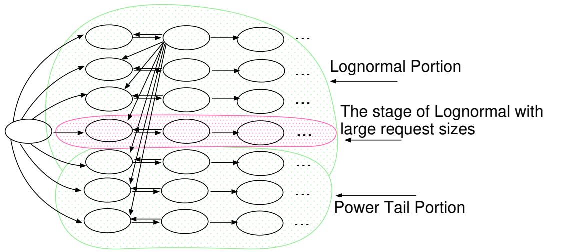

Figure 7: The CTMC that models one host.

request sizes and all phases of the hyper-exponential corresponding to the power-law part of the distribution). Since the Feldmann-Whitt algorithm "splits" the workload (each portion corresponding to one phase of the hyper-exponential distribution), it is possible to calculate approximately the contribution of specific task sizes to the queue length. Figure 8(a) illustrates that, at medium-to-high load, the queue occupation due to tasks with power-law distribution is about 20% of the overall queue length (even if the frequency of these tasks is almost negligible). This percentage is much larger at smaller arrival rates. We also note that if the lognormal portion has a small variability, the power tail queue dominates the queue build up. This appears at first counterintuitive, but it can be explained by examining the contribution to the queue build up by the tail of the lognormal distribution. Figure 8(b) shows that the tail of the lognormal distribution is very important for performance (requests for files larger than 100 KBytes dominate the queue across the whole range of arrival rates) and illustrates that for higher workload variabilities, the system queue length due to large yet rare tasks is significant.

These last observations suggest that it may be appropriate to assign tasks to specific hosts according to their sizes. We conjecture that by reserving hosts for scheduling tasks of similar sizes, we ensure that no severe imbalances in the utilization of each of the hosts occur. The workload fitting provided by the Feldmann-Whitt algorithm provides a hyper-exponential distribution with a special property: each exponential phase corresponds to a certain range of task (file) sizes. Thus, we use the hyper-exponential distribution to make an educated guess about a how to distribute the workload across the back-end servers, ensuring that the variance of the service time distribution of the tasks served by each server is kept as low as possible. Figure 9 illustrates this size-based policy.

By applying the size-based policy to our workload, we notice that a single server suffices to serve the power-law and the tail of the lognormal portions of the requests. The body of the lognormal portion must instead by served by the remaining seven servers. Figure 10 illustrates the average queue length of the hosts using either the size-based or the random policy for two fixed arrival rates, $\lambda = 0.0012$ and $\lambda = 0.0016$, representing modest and high load, respectively. In contrast to the random policy, the average queue length with the size-based policy does not increase as a function of the workload variability. This indicates that the size-based policy achieves a good utilization of all back-end hosts. Figure 10 shows similar behavior with respect to the expected task slowdown. We conclude this section by stressing that, while the size-based balancing algorithm does not provide an optimal solution to the problem (thus, there are "bumps" in the size-based graphs in Figure 10), it offers a simple and inexpensive solution that is significantly better than the random policy.

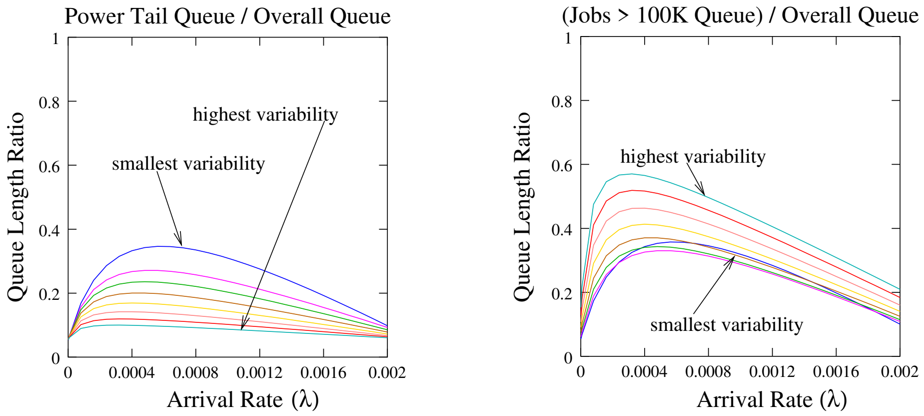

Figure 8: Contribution to the overall queue from the power-law portion of the workload (a), and from all files greater than 100 KBytes (b).

1. Compute the expected service time S_i for each phase of the hyper-exponential distribution, weighted by its probability β_i:

$$S_i = \frac{\beta_i}{\mu_i}, \quad 1 \le i \le k$$

2. Normalize S_i to compute each stage's contribution to the overall expected mean service time of the distribution:

$$\hat{S}_i = \frac{S_i}{\sum_{i=1}^{k} S_i}, \quad 1 \le i \le k$$

3. If c servers are available, then phase i should be served by

$$c_i = \hat{S}_i \cdot c \quad \text{servers}$$

(the specific server for the task is chosen randomly among the c_i servers)

4. Treat heavy-tail differently from the body of the distribution:

- **a.** $\forall c_i < 1, \quad 1 \le i \le k$, (i.e., for stages corresponding to the heavy tail) such that $\sum c_i < 1.5$, are to be served the *same* single server.
- **b.** $\forall c_i \ge 1, \quad 1 \le i \le k$, (i.e., for stages corresponding to the body), assign $\lfloor c_i + 0.5 \rfloor$ servers and schedule jobs within these servers using the random policy. Attention should be paid so as to ensure that the total server assignment across all stages of the hyper-exponential does not exceed c.

Figure 9: Our size-based scheduling policy.

5. CONCLUSIONS

In this paper we presented an analytic methodology for the exact analysis of load balancing policies in distributed multi-server system conditioned on the fact that the duration of arriving tasks is best described by a heavy tail distribution. The contributions are two-fold:

- the development of a new analytic methodology for the exact solution of $M/H_k/1$ queues, and
- the application of the analytic methodology for the analysis of the performance benefits of a size-based scheduling policy.

We emphasize that our methodology allows for the exact *quantification* of the effect of the task sizes on queue build-up for the various servers, and consequently allows for the detection of the cause of load imbalances. Our analysis indicates that a size-based policy that reserves specific servers for specific ranges of incoming task sizes greatly outperforms a policy that simply assigns an incoming task to a randomly selected host.

In the future, we intend to explore the effects of different inter-arrival distributions to the performance of the two policies analyzed in this paper and to consider the effects of caching at each back-end server. Further, we

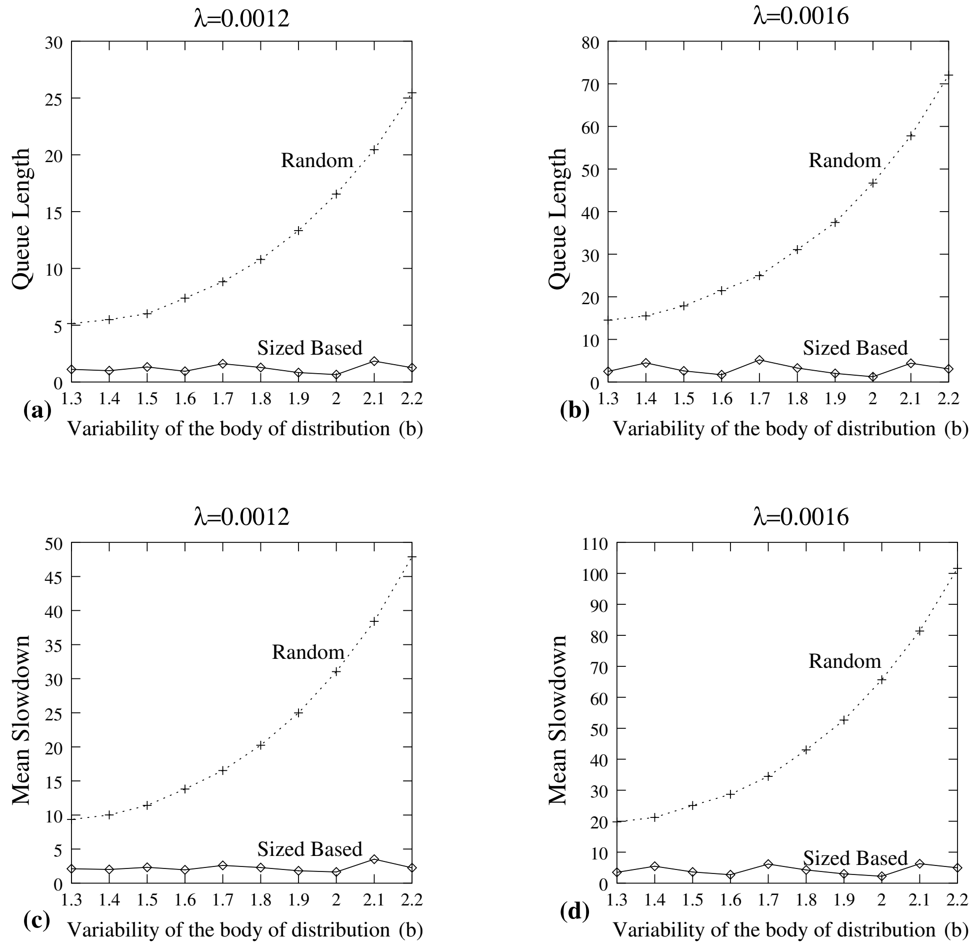

Figure 10: Policy comparisons as a function of the workload variability.

will explore the applicability of our model to analyze the performance of more complex scheduling policies, and in particular of policies that provide feedback to the dispatcher. To this end, we intend to combine simulation with analysis in the form of hybrid modeling.

Acknowledgments

We would like to thank Stephen K. Park for providing us insightful suggestions on fitting real data into distributions.

6. REFERENCES

[1] M. Arlitt and C.L. Williamson. Web Serwer Workload Characterization, the Search for Invariants. *In Proceedings of ACM SIGMETRICS Conference*, pp. 126-138, Philadelphia, PA, May 1996.

[2] M. Arlitt and T. Jin. Workload Characterization of the 1998 World Cup Web Site. *Hewlett-Packard Laboratories Technical Report*, September 1999.

[3] P. Barford and M.E. Crovella, Generating Representative Web Workloads for Network and Server Performance Evaluation. In Proceedings of Performance '98/ACM SIGMETRICS '98, pp. 151-160, Madison

[4] P. Barford, A. Bestavros, A. Bradley and M.E. Crovella, Changes in Web Client Access Patterns: Characteristics and Caching Implications. In World Wide Web, Special Issue on Characterization and Performance Evaluation, Vol. 2, pp. 15-28, 1999.

[5] G. Ciardo and E. Smirni. ETAQA: An Efficient Technique for the Analysis of QBD-processes by Aggregation. *Performance Evaluation 36-37 1999*, pp. 71-93.

[6] G. Ciardo, A. Riska and E. Smirni. An Aggregation-based Solution method for M/G/1-type processes. *In Proceedings of*

Numerical Solution of Markov chains '99, pp. 21-40, Zaragoza, Spain, September 1999.

[7] M. Colajanni, P.S. Yu and D.M. Dias, Analysis of Task Assignment Policies in Scalable Distributed Web-Servers Systems. *IEEE Transactions on Parallel and Distributed Systems*, Vol. (, N). 6, June 1998.

[8] M.E. Crovella and M.S. Taqqu. Estimating the Heavy Tail Index from Scaling Properties. In *Methodology and Computing in Applied Probability*, Vol 1, No. 1, pp. 55-79, 1999.

[9] A. Feldmann and W. Whitt. Fitting Mixtures of Exponentials to Long-Tail Distributions to Analyze Network Performance Models. *Performance Evaluation*, 31(8), pp. 963–976, Aug. 1998.

[10] M. Harchol-Balter, M.E. Crovella and C.D. Murta. On Choosing a Task Assignment Policy for a Distributed Server System. *In Proceedings of Performance Tools '98, Lecture Notes in Computer Science* Vol 1469, pp. 231–242, 1998.

[11] M. Harchol-Balter, and A. Downey. Exploiting Process Lifetime Distributions for Dynamic Load Balancing. *ACM Transactions on Computer Systems*, 15(3), pp. 253–285, Aug. 1997.

[12] L. Kleinrock. *Queueing Systems Volume I: Theory.* Wiley, 1975.

[13] G. Latouche and V. Ramaswami. *Introduction to Matrix Analytic Methods in Stochastic Modeling.* ASA-SIAM, 1999.

[14] A.M. Law and W.D. Kelton. *Simulation Modeling and Analysis.* McGraw-Hill Inc., 1982.

[15] R. Nelson. Matrix geometric solutions in Markov models: a mathematical tutorial. Research Report RC 16777 (#742931), IBM T.J. Watson Res. Center, Yorktown Heights, NY, Apr. 1991.

[16] M.F. Neuts. *Matrix-geometric solutions in stochastic models.* Johns Hopkins University Press, Baltimore, MD, 1981.

[17] M.F. Neuts. *Structured stochastic matrices of M/G/1 type and their applications.* Marcel Dekker, New York, NY, 1989.

[18] V. Ramaswami and G. Latouche. A general class of Markov processes with explicit matrix-geometric solutions. *Operation Research Spectrum 8*, pp. 209–218, Aug. 1986.

[19] V. Pai, M. Aron, G. Banga, M. Svendsen, P. Druschel, W. Zwaenepoel and E. Nahum. Locality-aware Request Distribution in Cluster-based Network Servers. *In Proceedings of the Eighth International Conference on Architectural Support for Programming Languages and Operating Systems* (ASPLOS-VIII), San Jose, California, October 1998.