EquiLoad: a load balancing policy for clustered web servers*

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

Abstract

We present a new strategy for the allocation of requests in clustered web servers, based on the size distribution of the requested documents. This strategy, EquiLoad, manages to achieve a balanced load to each of the back-end servers, and its parameters are obtained from the analysis of a trace's past data. To study its performance, we use phase-type distribution fittings and solve the resulting models using a new solution method for M/PH/1 queues that only requires solution of linear systems. The results show that EquiLoad greatly outperforms random allocation, performs comparably or better than the Shortest Remaining Processing Time and Join Shortest Queue policies and maximizes cache hits at the back-end servers, therefore behaving similarly to a "locality-aware" allocation policy, but at a very low implementation cost.

1 Introduction

The ubiquity and explosive growth of the World Wide Web presents many challenges to both researchers and practitioners, as information and its location increasingly determine the performance perceived by each individual user. The type of information (i.e., static data accessed with a simple file fetch vs. dynamic data that must be reconstructed at each access) as well as rapid fluctuations in the user demands imposed on a web server (e.g., special events such as the Olympic Games), make system administration and capacity planning a remarkably difficult task.

To improve the throughput of a web site and alleviate the server bottleneck, researchers have proposed clustering with a single-system image, that is, having multiple hosts behave like a single host. Indeed, a distributed web server that appears as a single host to the user is currently the most popular architecture and can be implemented at various levels: DNS-based, dispatcher-based (at the network level), and server-based. These trends point towards the need for robust routing algorithms that provide scalability, effective load balancing, and high availability in a constantly changing environment. Yet, there is no consensus on how to achieve high performance and support the ever increasing and diverse workloads without manual tuning. Furthermore, there is increasing evidence that the size of web documents, and accordingly their service demands, are governed by

*This work was partially supported by National Science Foundation under grant CCR-0098278.

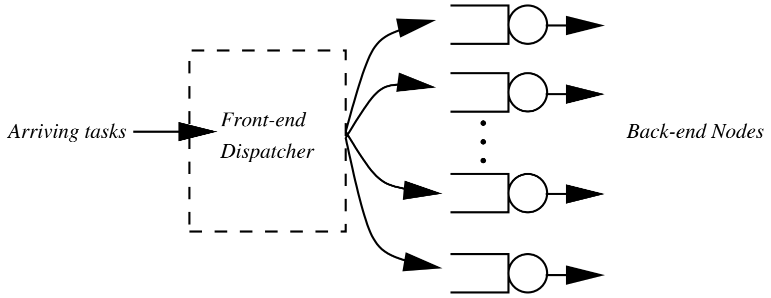

Figure 1: Model of a clustered web server.

heavy-tail distributions [2, 3, 4, 5]. As a consequence, load balancing strategies that perform well under light-tail workloads become inadequate [15].

In this paper, we propose a new, easy-to-implement scheduling policy for a clustered web server, whose abstraction is illustrated in Figure 1. With our new scheduling policy, called EquiLoad, the size of each incoming request determines the identity of the back-end server that will eventually satisfy it. The front-end dispatcher does not directly consider any information regarding the load of the individual back-end servers. Instead, an integral part of EquiLoad is a distributed workload characterization activity performed in the background by each back-end server. By continuously monitoring all incoming workload, these servers can then periodically renegotiate their agreement about the size of requests that will be allocated to them. To study the effectiveness of the renegotiation activity performed by EquiLoad, we fit the empirical distribution of the requests sizes experienced so far by the back-end servers as a whole, using a phase-type probability distribution. The result is one Markovian model of the M/G/1-type for each back-end server, which we can solve extending the efficient analytic models we have previously devised [8, 9, 26], to provide a fast analytical/numerical solution.

Our policy is inspired by previous work on size-based policies that advocate allocating incoming tasks to the back-end servers based solely on the task's size [14, 26], however, it is unique in that it is parametrized in such a way to present the same load to each back-end server. We evaluate the proposed policy via both analysis and trace-driven simulation, using real trace data showing the file size distribution for all requests to the 1998 World Soccer Cup Web Site.

Our contributions are two-fold. First, we provide a methodology for detailed characterization of web workloads, concentrating on fitting them with phase-type distributions that closely resemble the original distribution, and we show how the resulting models have efficient numerical solutions. We stress that this characterization can be done both off-line (i.e., on complete workload traces) and on-line (i.e., using censoring that periodically examines the workload seen so far and accordingly adjusting its fitting). Second, we propose a new size-based load balancing strategy for clustered web servers, compare its performance with other policies and measure its effectiveness on a real trace.

This paper is organized as follows. Section 2 summarizes related work. Section 3 introduces our load balancing strategy. In Section 4 we present the workload and the various steps we follow to obtain a phase-type distribution that closely approximates it. In Section 5 we give an quick overview of ETAQA, the fast numerical solution method we use for our M/G/1-type models, and we extend it to general M/PH/1 queues. Section 6 presents detailed comparisons of the performance of our proposed load balancing policy with other policies from the literature. Section 7 concludes the paper and outlines future work.

2 Related work

In a typical clustered web server system, URL requests arrive to a front-end system responsible for dispatching them to the back-end nodes [10, 14, 23]. This happens according to a task scheduling policy that aims at optimizing some measure quality-of-service, such as minimizing the expected response time. Identifying the "best" server to which assign each request requires detailed information about the distribution of the service demands.

In the past two decades, there has been a significant research effort in task scheduling and load balancing (see [15] 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, however, there is very strong evidence that the size of a web document, and accordingly its service demand, is governed instead by heavy-tail distributions [2, 3, 4, 5]. This implies that, to minimize response time in such a multi-server system, "short" and "long" jobs should be assigned to different queues [14, 15]. Under highly variable workloads, system performance improves considerably if detailed information about the activity of each back-end server is available to the front-end dispatcher [13]. In general-purpose computing systems, this ideal workload partition is hard to achieve because the length of a job is known only *a posteriori*. In web server systems, however, the size of the file corresponding to a given URL request is a good characterization of the length of the job, so approaching this ideal is not out of the question: this is the reason for the popularity of size-based policies for clustered web server systems [14].

However, a critical piece of information is normally missing: the contents of the RAM of each server, before the request is routed to the particular server and at the time the request is actually going to be served. This is fundamental, since the service time for a request can differ by over an order of magnitude depending on whether the request can be fulfilled from the RAM's contents, or it requires disk access. For this reason, much work has been directed towards two related directions. First, researchers have sought to characterize [1, 4] and then reproduce in synthetic traces [5, 7] the locality of web references. Then, "locality-aware" allocation policies that exploit this knowledge in the attempt to maximize cache hits have been proposed [6, 23].

3 EQUILOAD: a new load balancing policy

Previous work on load balancing in a distributed server environment where task service times are exponentially distributed highlights the need to direct tasks of "similar" size to the same server to avoid the significant slowdowns that small tasks would experience when queued behind large tasks [26, 15]. Knowing *a priori* the distribution of the incoming task sizes can be of great help.

Assuming that the number of back-end servers is N, our EQUILOAD policy requires partitioning the possible request sizes into N intervals, $[s_0 \equiv 0, s_1)$, $[s_1, s_2)$, $\ldots$, $[s_{N-1}, s_N \equiv \infty)$, so that server i will be responsible for satisfying requests of size between s_{i-1} and s_i. In practice, the size corresponding to an incoming URL request might not be available to the front-end dispatcher, but this problem can be solved using a two-stage allocation policy. First, the dispatcher assigns each incoming request very quickly to one of the N back-end servers using a simple policy such as uniform random (or round-robin, which is even easier to implement in practice). Then, when server i receives a request from the dispatcher, it looks up its size s and, if $s_{i-1} \le s < s_i$, it puts the request in its queue, otherwise it reallocates it to the server j satisfying $s_{j-1} \le s < s_j$ (of course any request server i receives from another server is instead enqueued immediately, since it is guaranteed to be in the correct size range).

Letting the back-end servers reallocate requests among themselves is very sensible, since the size information is certainly available to them. The potential advantages of such a policy are clear:

- Partitioning the workload into N class sizes and assigning each to a different server maximizes the homogeneity of request sizes within each queue, thus improves the overall response time.
- The dispatcher does not need to be aware of the sizes of the incoming requests, nor of the load of each back-end server.
- Requests must be reallocated at most once; indeed, if the dispatcher uses a simple random or round-robin allocation, the probability that a request must be reallocated is exactly $\frac{N-1}{N}$.
- Except for the small reallocation overhead, no server's work is wasted.
- The cache behavior is guaranteed to be close to optimal, since requests for the same file are assigned to the same server.

but so are the challenges:

- The value of the $N-1$ size boundaries $s_1, s_2, \ldots, s_{N-1}$ is critical, since it affects the load seen by each server; the mechanism used to choose these boundaries must achieve the desired performance goals, in our case, maintaining a similar expected queue length at each server.
- As the workload changes over time, we must dynamically readjust these boundaries, but the cost for doing so must be small.

For the first challenge, the objective is to provide each back-end server with (approximately) the same "load" by choosing the intervals $[s_{i-1}, s_i)$ so that the requests routed to server i contribute a fraction $1/N$ of the mean $\overline{S}$ of the distribution size. In other words, we seek to determine $s_1, s_2, \ldots, s_{N-1}$ such that, for $1 \le i \le N$,

$$\int_{s_{i-1}}^{s_i} x \cdot dF(x) \approx \frac{1}{N} \int_0^\infty x \cdot dF(x) = \frac{\overline{S}}{N},$$

where $F(x)$ is the CDF of the request sizes. Given an empirical request size distribution, that is, a trace of R requests and their sizes, this can be accomplished in three passes:

1. first, we sort the request sizes in increasing order,
2. then, we compute the expected request size $\overline{S}$ (using sorted data improves numerical stability),

3. finally, we scan the sorted data and accumulate their sizes, recording the points $s_1, s_2, \ldots, s_{N-1}$ at which we reach a fraction $\frac{1}{N}$, $\frac{2}{N}$, ..., $\frac{N-1}{N}$ of $R \cdot \overline{S}$.

The approach just described corresponds essentially to an off-line algorithm where the request size distribution is a fixed input parameter. The second challenge can be instead seen as an on-line problem, where the stream of requests is at the same time to be allocated according to the current values of the $N-1$ boundaries, but also used to determine their new values. We then need to devise a strategy where, these boundaries are periodically adjusted according to the overall request size distribution seen so far. In this work we simply hint at a few possibilities, and report data on the cost incurred when the boundaries are not properly adjusted.

4 The workload

The workload used in our analysis is obtained from real web server traces. Since we evaluate different load balancing policies using analytic techniques, we fit the data using phase-type distributions. In this section, we describe the workload used and the fitting steps.

We obtained workload traces of the 1998 World Soccer Cup Web site[1]. The server for this site 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 extract 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.

4.1 Fitting

Recall that our policy must identify the boundaries of the request sizes assigned to each server, and it does this by partitioning the set of possible sizes, from 0 to ∞, into N disjoint intervals, so that the same load is seen by all servers. Since we intend to study our policy using analytical models, we then need to fit our real trace data using phase-type distributions. In particular, the data falling into each interval is fitted into a phase-type distribution, then we probabilistically merge the N fittings, resulting in a distribution that is still phase-type, since it is a mixture of hypo-exponential (possibly Erlang) and hyper-exponential distributions [16]. The idea is illustrated in Fig. 2, where the vertical bars correspond to the empirical data (appropriately discretized into bins) which, since we assume $N = 4$, is fitted with four phase-type distributions over the four different ranges of

[1] Available at `http://researchsmp2.cc.vt.edu/cgi-bin/reposit/search.pl?details=YES&detailsoffset=135`.

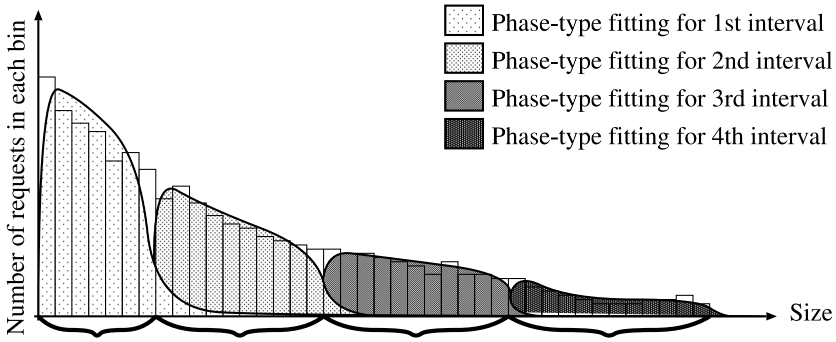

Figure 2: Fitting $N = 4$ intervals of the size distribution with individual phase-type distributions.

data[2].

Given a choice for an interval's boundaries, we consider the contribution of each discrete bin falling in the interval, and compute the mean and variance of the distribution, hence the the coefficient of variation. Then, according to whether this coefficient of variation is greater or smaller than one, we fit the data in the interval with a hyper-exponential or a hypo-exponential distribution using the method of moments (see the Newton-Raphson algorithm in [27]). To illustrate the fitting methodology, we selected a random date for the 1998 World Soccer Cup (day 57). Figure 3 shows the phase-type distribution resulting form the probabilistic merging of the individual four fittings; of those, the first one is a two-stage hyper-exponential, while the other three are hypo-exponential, with the last one very close to an Erlang (the numbers written inside each stage are the rates of the corresponding exponential distributions, while those on the arcs describe probabilistic splittings). To assess the quality of the overall phase-type fitting, we can consider Figure 4, which compares the probability density function (pdf) and the cumulative distribution function (CDF) for the empirical data and for our fitting. Even more importantly, we can consider Table 1, which shows how close important measures, the expected queue length and slowdown (defined as the ratio of the response time by the service time), are when computed using either the real vs. the fitted traffic (we assume a single server system with exponential arrivals where the unit of time is the time required to service a 1-byte request).

5 Analytic solution of M/PH/1 queues

Once we have characterized the load offered to each server using a phase-type distribution, we can use analytic models to study the resulting performance. This is especially efficient because of the potentially high variability present in the workload: although we do use trace-driven simulations to verify our analytic models, we observe that the use of simulation can be prohibitively time-consuming, especially if we want to obtain high-confidence measures related to the *rare events* responsible for the heavy-tail aspects of the distribution. In the following, we present an overview of classic techniques, and we introduce a new technique, for solving the class of queuing systems

[2]This is different from [11], where data is also broken into subsets, but then each section is fitted to a simple exponential distribution, and a hyper-exponential distribution is obtained when probabilistically merging back these distributions.

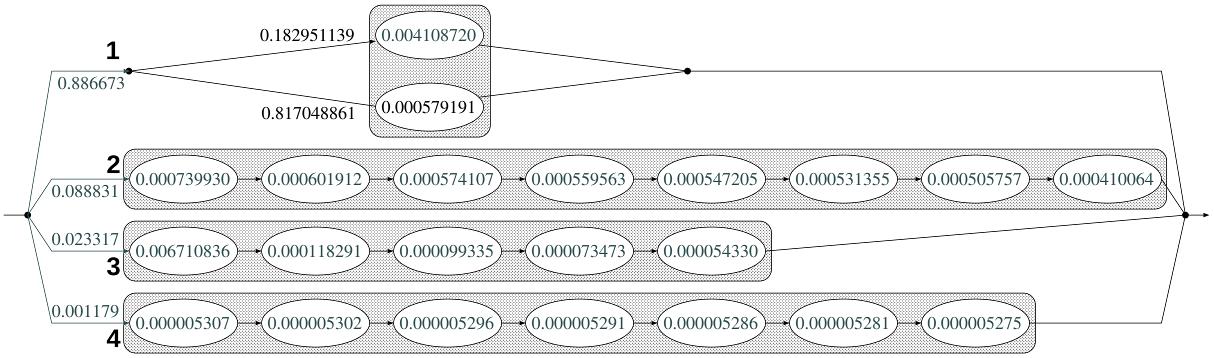

Figure 3: The resulting overall phase-type distribution.

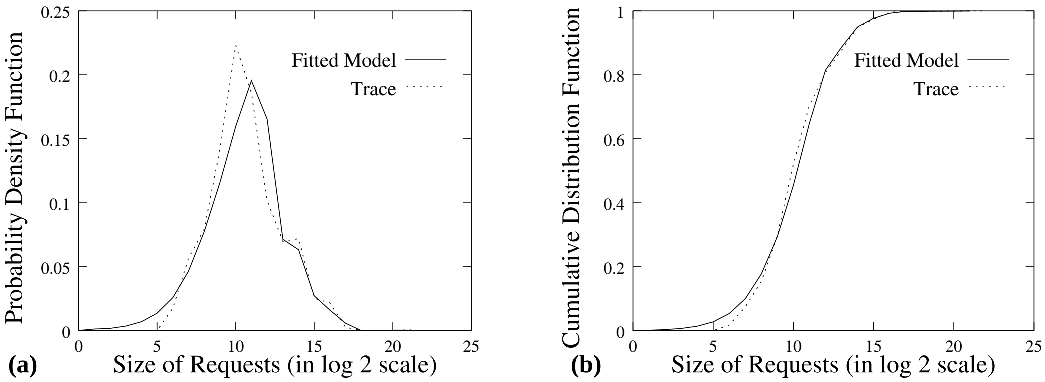

Figure 4: Comparing the empirical and fitted data: pdf (left) and CDF (right).

Arrival rate	**Real data (trace-driven simulation)**		**Fitted data (analytical solution)**	
	E[queue length]	**E[slowdown]**	**E[queue length]**	**E[slowdown]**
0.0000250	0.84	6.51	0.83	6.41
0.0000375	1.92	9.96	1.92	9.81
0.0000500	3.60	14.01	3.60	13.84
0.0000625	6.05	18.82	6.06	18.60
0.0000750	9.51	24.65	9.52	24.38
0.0000875	14.37	31.92	14.40	31.60
0.0001000	21.22	41.24	21.26	40.82
0.0001125	30.99	53.54	31.10	53.09
0.0001250	45.34	70.49	45.44	69.81
0.0001375	67.42	95.29	67.41	94.14
0.0001500	104.22	135.02	103.95	133.07

Table 1: Comparing the empirical and fitted data: performance results.

that models the systems we consider.

5.1 Background: matrix analytic methods

Over the last two decades, considerable effort has been put into the development of *matrix analytic techniques* [21, 22] for the exact analysis of a general and frequently encountered class of queuing models. In these models, the embedded Markov chains are two-dimensional generalizations of elementary GI/M/1 or M/G/1 queues [17]. In either case, the state space is partitioned into the "boundary" states $\mathcal{S}^{(0)} = \{s_1^{(0)}, \ldots, s_m^{(0)}\}$ and the "repetitive" sets of states $\mathcal{S}^{(j)} = \{s_1^{(j)}, \ldots, s_n^{(j)}\}$, for $j \geq 1$.

For GI/M/1-type and M/G/1-type Markov chains, the infinitesimal generator $\mathbf{Q}$ can be block-partitioned, respectively, as:

$$\begin{bmatrix} \hat{\mathbf{L}} & \hat{\mathbf{F}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \hat{\mathbf{B}}^{(1)} & \mathbf{L} & \mathbf{F} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \hat{\mathbf{B}}^{(2)} & \mathbf{B}^{(1)} & \mathbf{L} & \mathbf{F} & \mathbf{0} & \mathbf{0} & \cdots \\ \hat{\mathbf{B}}^{(3)} & \mathbf{B}^{(2)} & \mathbf{B}^{(1)} & \mathbf{L} & \mathbf{F} & \mathbf{0} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \hat{\mathbf{L}} & \hat{\mathbf{F}}^{(1)} & \hat{\mathbf{F}}^{(2)} & \hat{\mathbf{F}}^{(3)} & \hat{\mathbf{F}}^{(4)} & \hat{\mathbf{F}}^{(5)} & \cdots \\ \hat{\mathbf{B}} & \mathbf{L} & \mathbf{F}^{(1)} & \mathbf{F}^{(2)} & \mathbf{F}^{(3)} & \mathbf{F}^{(4)} & \cdots \\ \mathbf{0} & \mathbf{B} & \mathbf{L} & \mathbf{F}^{(1)} & \mathbf{F}^{(2)} & \mathbf{F}^{(3)} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{B} & \mathbf{L} & \mathbf{F}^{(1)} & \mathbf{F}^{(2)} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix},$$

where 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 "^" for matrices related to $\mathcal{S}^{(0)}$.

The *matrix geometric* approach [21] can be used to study GI/M/1-type processes. If $\boldsymbol{\pi}^{(j)}$ is the stationary probability vector for states in $\mathcal{S}^{(j)}$, we have:

$$\forall j \geq 1,\ \boldsymbol{\pi}^{(j+1)} = \boldsymbol{\pi}^{(j)}\mathbf{R} = \cdots = \boldsymbol{\pi}^{(1)}\mathbf{R}^j, \tag{1}$$

where $\mathbf{R}$ is the solution of the matrix equation $\mathbf{F} + \mathbf{R}\mathbf{L} + \sum_{k=1}^{\infty} \mathbf{R}^{k+1}\mathbf{B}^{(k)} = \mathbf{0}$. Iterative numerical algorithms can be used to compute $\mathbf{R}$. Then, we can write

$$\left[\boldsymbol{\pi}^{(0)} \;\middle|\; \boldsymbol{\pi}^{(1)}\right] \cdot \left[\begin{array}{c|c} \hat{\mathbf{L}} & \hat{\mathbf{F}} \\ \hline \sum_{k=1}^{\infty} \mathbf{R}^{k-1}\hat{\mathbf{B}}^{(k)} & \mathbf{L} + \sum_{k=1}^{\infty} \mathbf{R}^{k}\mathbf{B}^{(k)} \end{array}\right] = \mathbf{0}$$

which, together with the normalization condition $\boldsymbol{\pi}^{(0)}\mathbf{1} + \boldsymbol{\pi}^{(1)} \sum_{j=1}^{\infty} \mathbf{R}^{j-1}\mathbf{1} = 1$, or, equivalently, $\boldsymbol{\pi}^{(0)}\mathbf{1} + \boldsymbol{\pi}^{(1)}(\mathbf{I}-\mathbf{R})^{-1}\mathbf{1} = 1$, yields a unique solution for $\boldsymbol{\pi}^{(0)}$ and $\boldsymbol{\pi}^{(1)}$ (we let $\mathbf{0}$ and $\mathbf{1}$ be a row vector or a matrix of 0's, and a column vector of 1's of the appropriate dimensions, respectively). For $j \geq 2$, $\boldsymbol{\pi}^{(j)}$ can be obtained numerically from (1). More importantly, though, many useful performance metrics such as expected system utilization, throughput, or queue length can be computed exactly in explicit form from $\boldsymbol{\pi}^{(0)}$, $\boldsymbol{\pi}^{(1)}$, and $\mathbf{R}$ alone.

For the solution of M/G/1-type processes, several algorithms exist [12, 20, 22]. Here, we outline one that provides a stable calculation for the values of $\boldsymbol{\pi}^{(j)}$, the stationary probability vector for states in $\mathcal{S}^{(j)}$. Using Ramaswami's recursive formula [24] we have:

$$\forall j \geq 1, \quad \boldsymbol{\pi}^{(j)} = -\left(\boldsymbol{\pi}^{(0)}\hat{\mathbf{S}}^{(j)} + \sum_{l=1}^{j-1} \boldsymbol{\pi}^{(l)}\mathbf{S}^{(j-l)}\right) \mathbf{S}^{(0)^{-1}},$$

where $\hat{\mathbf{S}}^{(j)} = \sum_{l=j}^{\infty} \hat{\mathbf{F}}^{(l)}\mathbf{G}^{l-j}, \;\; j \geq 1$, and $\mathbf{S}^{(j)} = \sum_{l=j}^{\infty} \mathbf{F}^{(l)}\mathbf{G}^{l-j}, \;\; j \geq 0$ (letting $\mathbf{F}^{(0)} = \mathbf{L}$), and $\mathbf{G}$ is the solution of the matrix equation $\mathbf{B} + \mathbf{LG} + \sum_{j=1}^{\infty} \mathbf{F}^{(j)}\mathbf{G}^{j+1} = \mathbf{0}$.

Several iterative algorithms for the computation of $\mathbf{G}$ exist [12, 22], the most efficient being the one based on Toeplitz matrices [20]. See [25] for cases where $\mathbf{G}$ can be explicitly defined and does not require any calculation.

Given the above definition of $\boldsymbol{\pi}^{(j)}$ and the normalization condition, a unique vector $\boldsymbol{\pi}^{(0)}$ can be obtained by solving the following system of m equations

$$\boldsymbol{\pi}^{(0)} \left[\left(\hat{\mathbf{L}} - \hat{\mathbf{S}}^{(1)} \mathbf{S}^{(0)^{-1}} \hat{\mathbf{B}} \right)^{\diamond} \;\middle|\; \mathbf{1} - \left(\sum_{j=1}^{\infty} \hat{\mathbf{S}}^{(j)} \right) \left(\sum_{j=0}^{\infty} \mathbf{S}^{(j)} \right)^{-1} \mathbf{1} \right] = [\mathbf{0} \mid 1],$$

where the symbol "$\diamond$" indicates that we discard one (any) column of the corresponding matrix, since we added a column representing the normalization condition. Once $\boldsymbol{\pi}^{(0)}$ is known, we can then iteratively compute $\boldsymbol{\pi}^{(j)}$ for $j = 1, 2, \ldots$, stopping when the accumulated probability mass is close to one.

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 $\mathcal{S}^{(j-1)}$.

We are particularly interested in quasi-birth-death (QBD) Markov chains, which are the intersection of the two previous cases, and whose infinitesimal generator $\mathbf{Q}$ can be block-partitioned as:

$$\begin{bmatrix} \hat{\mathbf{L}} & \hat{\mathbf{F}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \hat{\mathbf{B}} & \mathbf{L} & \mathbf{F} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{B} & \mathbf{L} & \mathbf{F} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{B} & \mathbf{L} & \mathbf{F} & \mathbf{0} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \tag{2}$$

We can solve QBD processes using the matrix-geometric method. In this case, $\mathbf{R}$ is solution of the quadratic equation $\mathbf{F} + \mathbf{RL} + \mathbf{R}^2\mathbf{B} = \mathbf{0}$ and can be obtained numerically using an iterative procedure. Since QBD processes are also a special case of M/G/1-type processes, we can also consider the matrix $\mathbf{G}$ [18], solution of $\mathbf{B} + \mathbf{LG} + \mathbf{FG}^2 = \mathbf{0}$. Then, the relation $\mathbf{R} = -\mathbf{F}(\mathbf{L} + \mathbf{FG})^{-1}$ holds [18, pages 137-8], from which we can derive the fundamental equality

$$\mathbf{RB} = \mathbf{FG}. \tag{3}$$

5.2 ETAQA

We introduced the ETAQA approach first for the solution of QBD models [9], then extended for the more general M/G/1-type case [8]. The advantages of ETAQA are its simplicity and efficiency, as it is derived from first principles using the classic Chapman-Kolmogorov [17] equations that relate the flow into and out of each state in equilibrium. Other explicit solution techniques in the literature [28] for the type of chains we consider calculate the stationary probabilities using recursion (i.e., once we know the stationary probability of the states in $\mathcal{S}^{(j)}$, we can obtain the stationary

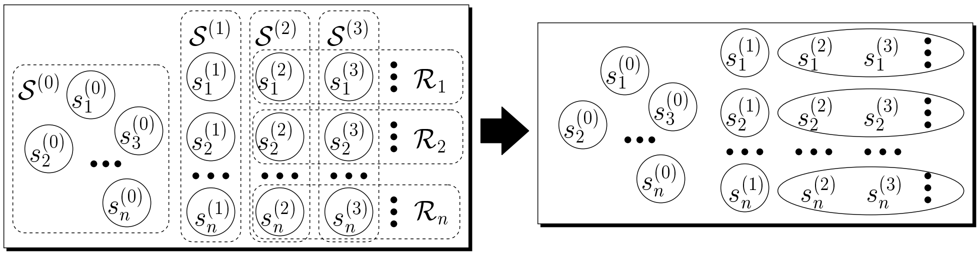

Figure 5: Aggregation of an infinite $\mathcal{S}$ into a finite number of states.

probability of the states in $\mathcal{S}^{(j+1)}$, and so on). Our method differs in that we do not evaluate the probability of *all* states in the chain, rather we compute the aggregate probability distribution for the repetitive portions of the chain. If this repetitive portion is composed of multiple subchains, we compute the aggregate probabilities of each infinite subchain in the system.

As introduced in [8, 9], ETAQA imposes no restriction on the structure of $\mathbf{L}$ and $\mathbf{F}$, they can be completely general, but $\mathbf{B}$ must instead have a special structure: only one of its columns (which by convention we number last, n), can contain nonzero entries. This effectively means that all transitions from $\mathcal{S}^{(j)}$ to $\mathcal{S}^{(j-1)}$ are restricted to go to $s_n^{(j-1)}$. When this condition holds (and we demonstrated several practical applications where it does in [8, 9]), it is possible to derive a system of $m+2n$ independent linear equations in $\boldsymbol{\pi}^{(0)}$, $\boldsymbol{\pi}^{(1)}$, and $\boldsymbol{\pi}^{(*)}$, a new vector of n unknowns representing the stationary probability of being in the sets of states $\mathcal{R}_i = \{s_i^{(j)} : j \geq 2\}$, for $i = 1, \ldots, n$, i.e., $\boldsymbol{\pi}^{(*)} = \sum_{j=2}^{\infty} \boldsymbol{\pi}^{(j)}$. Fig. 5 illustrates how this approach can be thought of as aggregating the states of $\mathcal{R}_i$ into a single macro-state.

The following theorem formalizes the above ideas and shows how to apply them to specific classes of CTMCs.

Theorem 5.1 Consider an M/G/1-type process with with infinitesimal generator $\mathbf{Q}$ and such that the first $n-1$ columns of $\mathbf{B}$ are null, $\mathbf{B}_{1:n,1:n-1} = \mathbf{0}$. Assume that the infinite sets of matrices $\{\hat{\mathbf{F}}^{(j)} : j \geq 1\}$ and $\{\mathbf{F}^{j} : j \geq 1\}$ are summable and that, in particular, they can be expressed as $\hat{\mathbf{F}}^{(j)} = \hat{\mathbf{A}}^{j-1}\hat{\mathbf{F}}$ and $\mathbf{F}^{(j)} = \mathbf{A}^{j-1}\mathbf{F}$, for $j \geq 1$, where $\hat{\mathbf{A}}$ and $\mathbf{A}$ are nonnegative diagonal matrices with elements strictly less than one. If $\boldsymbol{\pi} = \left[\boldsymbol{\pi}^{(0)}, \boldsymbol{\pi}^{(1)}, \boldsymbol{\pi}^{(2)}, \ldots\right]$ is the stationary probability vector of the process, the system of linear equations $\mathbf{xM} = [1, \mathbf{0}]$, where $\mathbf{M} \in \mathbb{R}^{(m+2n)\times(m+2n)}$ is defined as follows:

$$\mathbf{M} = \left[\begin{array}{c|c|c|c|c} \mathbf{1} & \hat{\mathbf{L}} & \hat{\mathbf{F}}^{(1)}_{1:m,1:n-1} & (\hat{\mathbf{A}}(\mathbf{I}-\hat{\mathbf{A}})^{-1}\hat{\mathbf{F}})_{1:n,1:n-1} & \hat{\mathbf{A}}(\mathbf{I}-\hat{\mathbf{A}})^{-2}\hat{\mathbf{F}}\mathbf{1} \\ \mathbf{1} & \hat{\mathbf{B}} & \mathbf{L}_{1:n,1:n-1} & ((\mathbf{I}-\mathbf{A})^{-1}\mathbf{F})_{1:n,1:n-1} & (\mathbf{I}-\mathbf{A})^{-2}\mathbf{F}\mathbf{1} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & ((\mathbf{I}-\mathbf{A})^{-1}\mathbf{F}+\mathbf{L})_{1:n,1:n-1} & ((\mathbf{I}-\mathbf{A})^{-2}\mathbf{F}-\mathbf{B})\mathbf{1} \end{array}\right]$$

admits a unique solution $\mathbf{x} = [\boldsymbol{\pi}^{(0)}, \boldsymbol{\pi}^{(1)}, \boldsymbol{\pi}^{(*)}]$ where $\boldsymbol{\pi}^{(*)} = \sum_{i=2}^{\infty} \boldsymbol{\pi}^{(i)}$.

Proof: See [8], or [9] for a version of this theorem restricted to QBD processes.

ETAQA provides excellent computational and storage savings in comparison to the classic approaches; we point the interested reader to [8] and [9] for a detailed discussion.

5.3 Extending ETAQA to M/PH/1 queues

In the special case of M/PH/1 queues, the infinitesimal generator has a peculiar structure, and we now show how our ETAQA solution approach can be extended to solve them, even if there are multiple return states from $\mathcal{S}^{(j)}$ to $\mathcal{S}^{(j-1)}$, that is, $\mathbf{B}$ contains multiple nonzero columns[3].

The service time of an M/PH/1 queue can be characterized as the time to absorption in a CTMC with state space $\mathcal{T} \cup \{a\}$, where $\mathcal{T}$ is a set of n transient states and $\{a\}$ is the unique absorbing state. The infinitesimal generator of this absorbing CTMC is

$$\mathbf{P} = \left[\begin{array}{c|c} \mathbf{P}^{\mathcal{T}} & \boldsymbol{\mu} \\ \hline \mathbf{0} & 0 \end{array}\right]$$

and its initial probability vector is $[\boldsymbol{\beta} | \, 0]$, with $\boldsymbol{\beta} \in \mathbb{R}^n$.

If the arrival rate to the M/PH/1 queue is λ, the CTMC modeling this queue has then a QBD form with $\mathbf{L} = \mathbf{P}^{\mathcal{T}}$, $\mathbf{F} = \lambda \mathbf{I}$, and $\mathbf{B} = \boldsymbol{\mu}\boldsymbol{\beta}$. In other words, unless the phase-type distribution of the service time is such that the absorbing CTMC starts in a given state of $\mathcal{T}$ with probability one (i.e., exactly one entry in $\boldsymbol{\beta}$ equals one, while all others are zero, as is the case in an Erlang distribution), multiple, and possibly all, columns in $\mathbf{B}$ are nonzero, thus ETAQA cannot be applied as stated in [8, 9]. However, $\mathbf{B}$ still has a very special form: its columns are all proportional to the vector $\boldsymbol{\mu}$, where the proportionality constants are given by the entries in $\boldsymbol{\beta}$, which of course sum to one.

Since $\boldsymbol{\beta}\mathbf{1} = 1$ and $\mathbf{B} = \boldsymbol{\mu}\boldsymbol{\beta}$, Ramaswami and Latouche showed that $\mathbf{G}$ can be explicitly obtained as $\mathbf{G} = \mathbf{1}\boldsymbol{\beta}$ [25]. This is a very important result which greatly facilitates the computation of the exact stationary probabilities of the M/PH/1 queue.

Consider the equation $\boldsymbol{\pi}\mathbf{Q} = \mathbf{0}$ for a QBD process, which can be written as

$$\left\{ \begin{array}{ccccccc} & & \boldsymbol{\pi}^{(0)}\,\hat{\mathbf{L}} & + & \boldsymbol{\pi}^{(1)}\,\hat{\mathbf{B}} & = & \mathbf{0} \\ \boldsymbol{\pi}^{(0)}\,\hat{\mathbf{F}} & + & \boldsymbol{\pi}^{(1)}\,\mathbf{L} & + & \boldsymbol{\pi}^{(2)}\,\mathbf{B} & = & \mathbf{0} \\ \boldsymbol{\pi}^{(1)}\,\mathbf{F} & + & \boldsymbol{\pi}^{(2)}\,\mathbf{L} & + & \boldsymbol{\pi}^{(3)}\,\mathbf{B} & = & \mathbf{0} \\ \boldsymbol{\pi}^{(2)}\,\mathbf{F} & + & \boldsymbol{\pi}^{(3)}\,\mathbf{L} & + & \boldsymbol{\pi}^{(4)}\,\mathbf{B} & = & \mathbf{0} \\ & & \vdots & & & & \end{array} \right. . \tag{4}$$

If we sum all row blocks in Eq. (4) from the third one on, we obtain

$$\sum_{j=1}^{\infty} \boldsymbol{\pi}^{(j)}\mathbf{F} + \sum_{j=2}^{\infty} \boldsymbol{\pi}^{(j)}\mathbf{L} + \sum_{j=3}^{\infty} \boldsymbol{\pi}^{(j)}\mathbf{B} = \mathbf{0},$$

which we can rewrite as

$$\boldsymbol{\pi}^{(1)}\mathbf{F} - \boldsymbol{\pi}^{(2)}\mathbf{B} + \boldsymbol{\pi}^{(*)}(\mathbf{B} + \mathbf{L} + \mathbf{F}) = \mathbf{0}, \tag{5}$$

where $\boldsymbol{\pi}^{(*)} = \sum_{j=2}^{\infty} \boldsymbol{\pi}^{(j)}$. Then, using now Eq. (1) and (3), we obtain

$$\forall j \geq 1, \;\; \boldsymbol{\pi}^{(j+1)}\mathbf{B} = \boldsymbol{\pi}^{(j)}\mathbf{R}\mathbf{B} = \boldsymbol{\pi}^{(j)}\mathbf{F}\mathbf{G},$$

[3]An earlier version of this result appeared in [26], where we were only interested in the special case of hyper-exponential service times, which can arise when fitting a heavy-tail distribution.

which, summed over all $j \geq 1$ gives

$$\boldsymbol{\pi}^{(*)}\mathbf{B} = \boldsymbol{\pi}^{(1)}\mathbf{FG} + \boldsymbol{\pi}^{(*)} \cdot \mathbf{FG},$$

or

$$\boldsymbol{\pi}^{(*)}(\mathbf{B} - \mathbf{FG}) = \boldsymbol{\pi}^{(1)}\mathbf{FG} = \boldsymbol{\pi}^{(1)}\mathbf{RB} = \boldsymbol{\pi}^{(2)}\mathbf{B}.$$

Using this last equality, we can then substitute $\boldsymbol{\pi}^{(*)}(\mathbf{B} - \mathbf{FG})$ for $\boldsymbol{\pi}^{(2)}\mathbf{B}$ in both the second row block of Eq. (4) and in Eq. (5), and obtain, respectively,

$$\boldsymbol{\pi}^{(0)}\hat{\mathbf{F}} + \boldsymbol{\pi}^{(1)}\mathbf{L} + \boldsymbol{\pi}^{(*)}(\mathbf{B} - \mathbf{FG}) = \mathbf{0}$$

and

$$\boldsymbol{\pi}^{(1)}\mathbf{F} + \boldsymbol{\pi}^{(*)}(\mathbf{L} + \mathbf{F} + \mathbf{FG}) = \mathbf{0},$$

which, together with the first row block of Eq. (4), provides a new system of $m + 2n$ equations in $\boldsymbol{\pi}^{(0)}$, $\boldsymbol{\pi}^{(1)}$, and $\boldsymbol{\pi}^{(*)}$. Written in matrix form, this system is:

$$\left[\boldsymbol{\pi}^{(0)} \middle| \boldsymbol{\pi}^{(1)} \middle| \boldsymbol{\pi}^{(*)}\right] \cdot \underbrace{\left[\begin{array}{c|c|c} \hat{\mathbf{L}} & \hat{\mathbf{F}} & \mathbf{0} \\ \hline \hat{\mathbf{B}} & \mathbf{L} & \mathbf{F} \\ \hline \mathbf{0} & \mathbf{B} - \mathbf{FG} & \mathbf{L} + \mathbf{F} + \mathbf{FG} \end{array}\right]}_{\mathbf{X}} = [\, \mathbf{0} \mid \mathbf{0} \mid \mathbf{0} \,]. \tag{6}$$

The following theorem states that the rank of $\mathbf{X}$ is $m + 2n - 1$, implying that (6) contains enough information to compute $\boldsymbol{\pi}^{(0)}$, $\boldsymbol{\pi}^{(1)}$, and $\boldsymbol{\pi}^{(*)}$, once we also take into account the normalization constraint

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

Theorem 5.2 Given an ergodic CTMC with infinitesimal generator $\mathbf{Q}$ having the structure shown in (2), the rank of matrix $\mathbf{X}$ defined in (6) is $m + 2n - 1$.

Proof. To show that the rank of $\mathbf{X}$ is $m + 2n - 1$, we use the (infinite) column vectors shown in Figure 6. The blocks labeled $\mathbf{V}^{(0)}$, $\mathbf{V}^{(1)}$, ... are simply the original blocks of (2), and we know the corresponding infinite set of column vectors has a defect of one, that is, if we discard any one column, the rest is a infinite set of linearly independent vectors. The blocks labeled $\mathbf{W}^{(3)}$, $\mathbf{W}^{(4)}$, ... are obtained as $\mathbf{W}^{(j)} = \sum_{l=j}^{\infty} \mathbf{V}^{(l)}$, for $j \geq 3$. The blocks labeled $\mathbf{Y}^{(3)}$, $\mathbf{Y}^{(4)}$, ... are obtained as $\mathbf{Y}^{(j)} = \mathbf{W}^{(j)}G$, for $j \geq 3$, this can be seen by observing that (i) $(\mathbf{L} + \mathbf{F})\mathbf{G} = \mathbf{LG} + \mathbf{FG} = \mathbf{LG} + \mathbf{FG}^2 = -\mathbf{B}$, where the second to the last step is due to the fact that $\mathbf{G}^2 = \mathbf{G}(\mathbf{1}\boldsymbol{\beta}) = (\mathbf{G1})\boldsymbol{\beta} = \mathbf{1}\boldsymbol{\beta} = \mathbf{G}$ and the last step is due to the fact that $\mathbf{B} + \mathbf{LG} + \mathbf{FG}^2 = \mathbf{0}$, and that (ii) $(\mathbf{B} + \mathbf{L} + \mathbf{F})\mathbf{G} = (\mathbf{B} + \mathbf{L} + \mathbf{F})\mathbf{1}\boldsymbol{\beta} = \mathbf{0}\boldsymbol{\beta} = \mathbf{0}$, since the rows of $\mathbf{B} + \mathbf{L} + \mathbf{F}$ sum to zero. Finally, $\mathbf{Z}^{(1)} = \mathbf{V}^{(1)} - \sum_{j=3}^{\infty} \mathbf{Y}^{(j)}$, and $\mathbf{Z}^{(2)} = \mathbf{V}^{(2)} + \mathbf{W}^{(3)} + \sum_{j=3}^{\infty} \mathbf{Y}^{(j)}$.

The row sums of the $m+2n$ column vectors described by the matrix $\mathbf{X}' = \left[\mathbf{V}^{(0)} \middle| \mathbf{Z}^{(1)} \middle| \mathbf{Z}^{(2)}\right]$ are zero, hence the rank of $\mathbf{X}'$ *at most* equals $m + 2n - 1$. To show that the rank is *exactly* equal to $m + 2n - 1$, we simply need to recall that the defect of the $\mathbf{Q}$ is one, and observe that $\mathbf{X}'$ can be expressed as the sum of $\left[\mathbf{V}^{(0)} \middle| \mathbf{V}^{(1)} \middle| \mathbf{V}^{(2)}\right]$ with a matrix

$\mathbf{V}^{(0)}$	$\mathbf{V}^{(1)}$	$\mathbf{V}^{(2)}$	$\cdots$	$\mathbf{W}^{(3)}$	$\mathbf{W}^{(4)}$	$\cdots$	$\mathbf{Y}^{(3)}$	$\mathbf{Y}^{(4)}$	$\cdots$	$\mathbf{Z}^{(1)}$	$\mathbf{Z}^{(2)}$
$\hat{\mathbf{L}}$	$\hat{\mathbf{F}}$	$\mathbf{0}$	$\cdots$	$\mathbf{0}$	$\mathbf{0}$	$\cdots$	$\mathbf{0}$	$\mathbf{0}$	$\cdots$	$\hat{\mathbf{F}}$	$\mathbf{0}$
$\hat{\mathbf{B}}$	$\mathbf{L}$	$\mathbf{F}$	$\cdots$	$\mathbf{0}$	$\mathbf{0}$	$\cdots$	$\mathbf{0}$	$\mathbf{0}$	$\cdots$	$\mathbf{L}$	$\mathbf{F}$
$\mathbf{0}$	$\mathbf{B}$	$\mathbf{L}$	$\cdots$	$\mathbf{F}$	$\mathbf{0}$	$\cdots$	$\mathbf{FG}$	$\mathbf{0}$	$\cdots$	$\mathbf{B}-\mathbf{FG}$	$\mathbf{L}+\mathbf{F}+\mathbf{FG}$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{B}$	$\cdots$	$\mathbf{L}+\mathbf{F}$	$\mathbf{F}$	$\cdots$	$-\mathbf{B}$	$\mathbf{FG}$	$\cdots$	$\mathbf{B}-\mathbf{FG}$	$\mathbf{L}+\mathbf{F}+\mathbf{FG}$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\cdots$	$\mathbf{B}+\mathbf{L}+\mathbf{F}$	$\mathbf{L}+\mathbf{F}$	$\cdots$	$\mathbf{0}$	$-\mathbf{B}$	$\cdots$	$\mathbf{B}-\mathbf{FG}$	$\mathbf{L}+\mathbf{F}+\mathbf{FG}$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\cdots$	$\mathbf{B}+\mathbf{L}+\mathbf{F}$	$\mathbf{B}+\mathbf{L}+\mathbf{F}$	$\cdots$	$\mathbf{0}$	$\mathbf{0}$	$\cdots$	$\mathbf{B}-\mathbf{FG}$	$\mathbf{L}+\mathbf{F}+\mathbf{FG}$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\cdots$	$\mathbf{B}+\mathbf{L}+\mathbf{F}$	$\mathbf{B}+\mathbf{L}+\mathbf{F}$	$\cdots$	$\mathbf{0}$	$\mathbf{0}$	$\cdots$	$\mathbf{B}-\mathbf{FG}$	$\mathbf{L}+\mathbf{F}+\mathbf{FG}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$

Figure 6: The blocks of column vectors used in our proof.

whose columns are a linear combination of vectors in $\mathbf{V}^{(j)}$ for $j \geq 3$, but not in $\mathbf{V}^{(0)}$, $\mathbf{V}^{(1)}$, $\mathbf{V}^{(2)}$.

Since all row blocks of $\mathbf{X}'$ except the first two are the same, the rank of the first three row blocks of $\mathbf{X}'$, that is, the rank of $\mathbf{X}$, is clearly equal to the rank of $\mathbf{X}'$, thus the theorem is proven. $\square$

We observe that, even if matrix $\mathbf{X}$ has rows that sum to zero, we cannot conclude that it is an infinitesimal generator, because the bottom block of its middle column, $\mathbf{B} - \mathbf{FG}$, is not guaranteed to be non-negative in general. We also stress that, in the special case of M/PH/1 queues, we can explicitly obtain $\mathbf{G}$ as $\mathbf{1}\boldsymbol{\beta}$. Indeed, we only need to store the vector $\boldsymbol{\beta}$ to fully capture $\mathbf{G}$.

5.4 Measures of interest

From just the three vectors $\boldsymbol{\pi}^{(0)}$, $\boldsymbol{\pi}^{(1)}$, and $\boldsymbol{\pi}^{(*)}$ we can then compute any measure of interest r that can be written as

$$r = \boldsymbol{\pi}^{(0)}\boldsymbol{\rho}^{(0)} + \boldsymbol{\pi}^{(1)}\boldsymbol{\rho}^{(1)} + \sum_{j=2}^{\infty} \boldsymbol{\pi}^{(j)}\boldsymbol{\rho}^{(j)} \tag{7}$$

(where $\boldsymbol{\rho}^{(j)}$ is a column vector expressing the *reward rates* for the states in $\mathcal{S}^{(j)}$), provided that 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 $a_i^{[0]}, a_i^{[1]}, \ldots, a_i^{[k]}$:

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

This includes a very rich class of measures which includes, for example any moment of the system queue length.

Since $\boldsymbol{\pi}^{(0)}\boldsymbol{\rho}^{(0)}$ and $\boldsymbol{\pi}^{(1)}\boldsymbol{\rho}^{(1)}$ are trivially computable, we only need to focus on the computation of the summation in Eq. (7):

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

$$
\begin{aligned}
&= \sum_{j=2}^{\infty} \boldsymbol{\pi}^{(j)}\mathbf{a}^{[0]} + \sum_{j=2}^{\infty} j\boldsymbol{\pi}^{(j)}\mathbf{a}^{[1]} + \cdots + \sum_{j=2}^{\infty} j^k\boldsymbol{\pi}^{(j)}\mathbf{a}^{[k]} \\
&= \mathbf{r}^{[0]}\mathbf{a}^{[0]} + \mathbf{r}^{[1]}\mathbf{a}^{[1]} + \cdots + \mathbf{r}^{[k]}\mathbf{a}^{[k]},
\end{aligned}
$$

where $\mathbf{r}^{[l]} = \sum_{j=2}^{\infty} j^l \boldsymbol{\pi}^{(j)}$ for $l = 0, \ldots, k$.

To compute these vectors, observe that $\mathbf{r}^{[0]}$ is simply $\boldsymbol{\pi}^{(*)}$ while, for $k > 0$, $\mathbf{r}^{[k]}$ can be computed after having obtained $\mathbf{r}^{[l]}$ for $0 \le l < k$, by solving the system of n linear equations:

$$
\begin{cases}
\mathbf{r}^{[k]}(\mathbf{B}+\mathbf{L}+\mathbf{F})^{\diamond} & = \mathbf{b}^{\diamond} \\
\mathbf{r}^{[k]}(\mathbf{F}-\mathbf{B})\mathbf{1} & = c
\end{cases}
\tag{8}
$$

where

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

This was shown in [9] for the case where $\mathbf{B}$ contains a single nonzero column, but the derivation is valid also in the more general case we are considering, so we only outline the idea here. We obtain equation $\mathbf{r}^{[k]}(\mathbf{B}+\mathbf{L}+\mathbf{F}) = \mathbf{b}$ by multiplying the second row block in Eq. 4 by 2^k, the third row block by 3^k, and so on, and then summing them. We obtain equation $\mathbf{r}^{[k]}(\mathbf{F}-\mathbf{B})\mathbf{1} = c$ by summing the equations $\boldsymbol{\pi}^{(j-1)}\mathbf{F}\mathbf{1} = \boldsymbol{\pi}^{(j)}\mathbf{B}\mathbf{1}$, which express the flow balance between $\mathcal{S}^{(j-1)}$ and $\mathcal{S}^{(j)}$, for $j \ge 2$, after having multiplied them by j^k. The only difference from [9] is that, to ensure that the linear system of Eq. 8 has a unique solution, we now need to show that $[(\mathbf{B}+\mathbf{L}+\mathbf{F})^{\diamond}|\,(\mathbf{F}-\mathbf{B})\mathbf{1}]$ has full rank. This is true because $(\mathbf{B}+\mathbf{L}+\mathbf{F})$ is an infinitesimal generator with rank $n-1$, thus has a unique stationary probability vector $\boldsymbol{\gamma}$ satisfying $\boldsymbol{\gamma}(\mathbf{B}+\mathbf{L}+\mathbf{F}) = \mathbf{0}$. However, this same vector must satisfy $\boldsymbol{\gamma}\mathbf{B}\mathbf{1} > \boldsymbol{\gamma}\mathbf{F}\mathbf{1}$ to ensure that the QBD process has a positive drift toward $\mathcal{S}^{(0)}$, thus is ergodic, hence $\boldsymbol{\gamma}(\mathbf{F}-\mathbf{B})\mathbf{1} < 0$, which shows that $(\mathbf{F}-\mathbf{B})\mathbf{1}$ cannot be possibly obtained as a linear combinations of columns of $(\mathbf{B}+\mathbf{L}+\mathbf{F})$. In other words, if we discard any column of $(\mathbf{B}+\mathbf{L}+\mathbf{F})$ and replace it with $(\mathbf{F}-\mathbf{B})\mathbf{1}$, the rank of the resulting matrix must be n.

To conclude this section, we stress that, to compute the k^{th} moment of the queue length we must then solve k linear systems in n unknowns each and, in particular, the expected queue length is obtained by solving just one linear system in n unknowns.

6 Results

We consider the following model of a distributed server environment. We assume a fixed number N of servers with the same processing power, each serving tasks in first-come-first-serve order. We further assume that each server has an unbounded queue. Tasks arrive to the dispatcher from the outside world according to a Poisson process. To evaluate the behavior of the EQUILOAD policy, we first compare its performance with a Random policy where the dispatcher has no knowledge about the load of the back-end server and assigns each incoming job to a randomly selected server with probability $1/N$. We also compare EquiLoad with the Shortest Remaining Processing Time (SRPT) policy, which has been shown to handle well both highly variable workload and heavy overall system load, and the Join Shortest Queue (JSQ) policy, which handles the load of the system reasonably well [29, 30]. We demonstrate that EquiLoad is, by its nature, a good choice to maximize cache hits at the back-end servers and compare its performance with LARD [23], a "locality-aware" load balancing policy in clustered web-servers. We conclude this section by commenting on the sensitivity of EquiLoad to the dynamic workload behavior.

6.1 Workload variability

As our purpose is to analyze load balancing policies according to workload variability, we use a synthetic workload that resembles significantly the one observed in the World Cup workload traces. We note that in previous works [2, 26] it has been demonstrated that each day of the trace data can be fit into a lognormal distribution using the method of maximum likelihood estimators. Here we selected a random day of the workload, namely day 57, and first fitted the data into a lognormal distribution. Then, the workload's probability distribution function (p.d.f) is given by:

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

We compute $b > 0$ (i.e., the shape parameter), and $a \in (-\infty, \infty)$ (i.e., the scale parameter) using the maximum likelihood estimators [19]:

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

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

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 change the variance of the distribution while at the same type keeping its mean constant (and equal to the mean of day 57, i.e., 3,629 bytes. This way, we can change the impact of the workload variability to the performance of load balancing policies. Figure 7 illustrates how the CDF of the distribution changes when we change the workload variability.

6.2 Comparison of EquiLoad and the Random policy

We first consider the performance of the Random policy, under the synthetic workload described in the previous subsection, recalling that one of the curves in Figure 7 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 four back-end servers. Thus, the arrival rate to each individual server with the random policy is $\lambda/4$. All graphs presented here are obtained analytically. We note that whenever possible, i.e., for the case of day 57, the analytic performance measures were also validated with trace driven simulation and shown to be in excellent agreement.

The expected system queue length for the Random policy as a function of the workload arrival intensity is illustrated in Figure 8(a). Although the system saturates at the same value of λ regardless

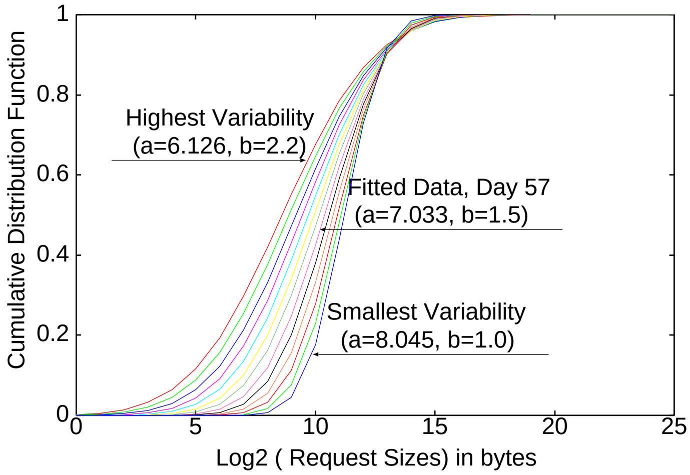

Figure 7: Shape of the CDF when changing the variability of the lognormal portion of the workload.

of the workload choice (recall that all workloads have the same mean task size), the expected queue length differs dramatically from workload to workload, especially in the range of medium-to-high system utilization. Figure 8(b) reports the same results as a function of the workload variability for various arrival rates[4]. The figure further confirms that the higher the workload variability, the more dramatic the expected queue build-up is. The expected task slowdown (not shown) closely follows the observed behavior for system queue length.

We continue by comparing the expected queue build-up of the Random and EquiLoad policies as a function of the workload variability. We concentrate on performance figures for medium arrival intensity into the system (Figure 9). In contrast to the Random policy, the expected queue length with EquiLoad does not increase as a function of the workload variability. EquiLoad exhibits a remarkable ability to select the size range boundaries for each server so as to keep the expected queue build-up to a minimum, thus offers a simple and inexpensive solution that is significantly better than the Random policy.

6.3 Comparison of EquiLoad, SRPT, and JSQ policies

To assess the performance of EquiLoad with respect to workload variability and system load, we compare it with the Shortest Remaining Processing Time (SRPT) and Join the Shortest Queue (JSQ) load balancing policies. SRPT assigns each incoming request to the server that is expected to finish first the requests already assigned to it. The dispatcher makes this assignment decision based on detailed information about the completion time for each request in all servers of the cluster. JSQ assigns each incoming request to the server with the shortest queue. In this policy, the front-end dispatcher must know only the size of the waiting queue for each server in the cluster. In a distributed environment, this policy handles system load well [29, 30].

[4]For presentation clarity, the x-axis of Figure 8(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.

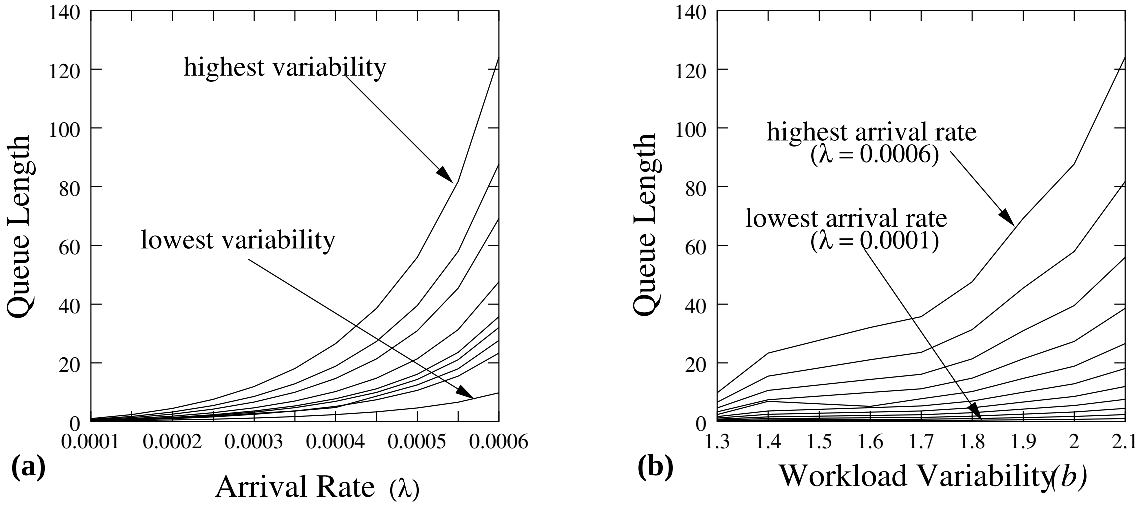

Figure 8: Average queue length of the Random policy as a function of the overall task arrival rate λ for workloads with high-to-low variance in their task service time **(a)**, and expected queue length as a function of the workload variability b for various arrival rates of the workload **(b)**.

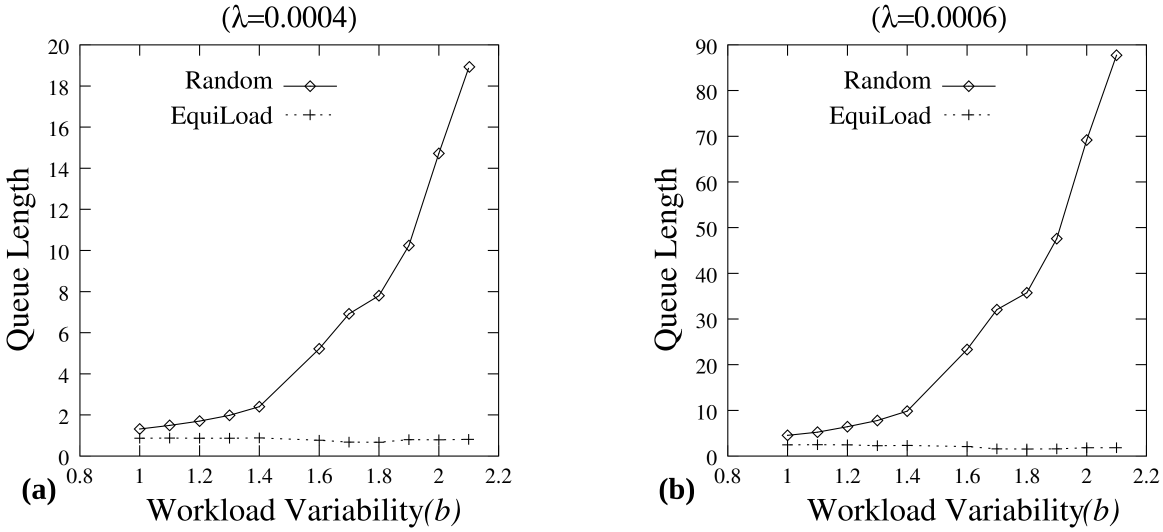

Figure 9: Average queue length of Random and EquiLoad policies as a function of the workload variability for medium arrival rate ($\lambda = 0.0004$) and high arrival rate ($\lambda = 0.0006$).

We compare EquiLoad with SRPT and JSQ using simulation. Figure 10 illustrates the expected queue length in each server of the cluster as function of workload variability, for a medium ($\lambda = 0.0004$) or heavy ($\lambda = 0.0006$) system load. The expected task slowdown (not shown) closely follows the observed behavior for system queue length. SRPT handles well both the variability of the workload and the load of the cluster. JSQ performs slightly worse than SRPT but it does

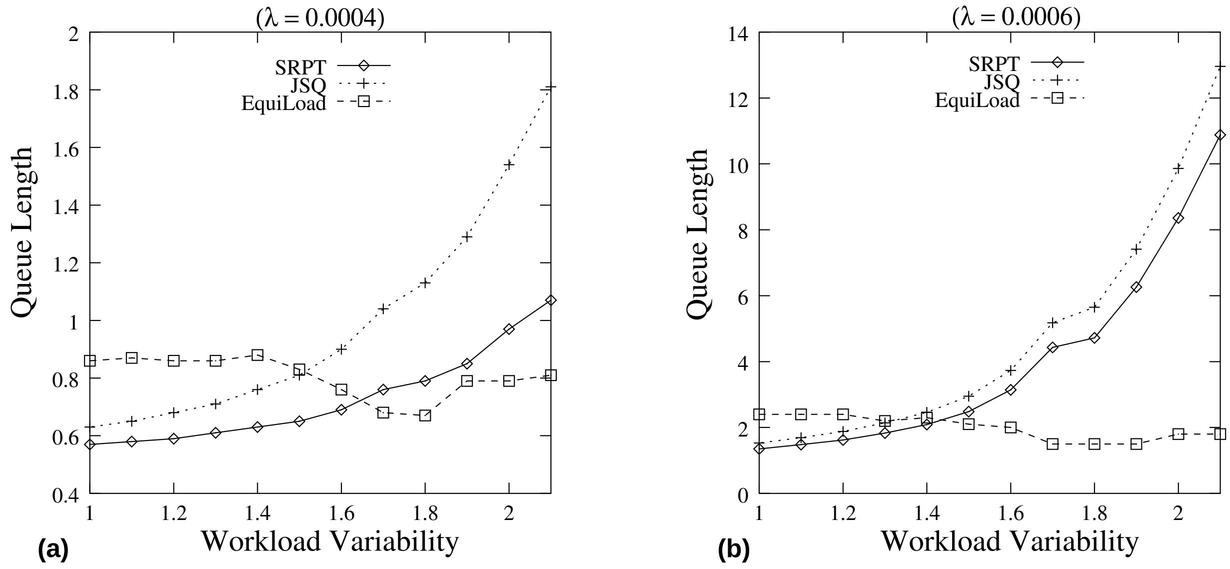

Figure 10: Average queue length of SRPT, JSQ and EQUILOAD policies as a function of the workload variability for medium arrival rate ($\lambda = 0.0004$) and high arrival rate ($\lambda = 0.0006$).

not require as much information to be available to the front-end dispatcher. When the workload variability or the system load are not very high, both SRPT and JSQ perform slightly better than EQUILOAD (the "jagged" EQUILOAD curve is an artifact of the workload fitting). In more critical cases for the operation of the cluster, such as higher workload variability or higher system load, EQUILOAD performs better than SRPT and JSQ (note the scale difference of the y-axis between Figures 10(a) and 10(b)).

We conclude by observing that the performance of EQUILOAD is comparable to that of SRPT and JSQ in non-critical cases, but substantially better in critical ones. Furthermore, EQUILOAD does not require the front-end dispatcher to have any additional knowledge about the processing and the load of each server in the cluster, while both SRPT and JSQ require the front-end dispatcher to know in detail (SRPT even in more detail than JSQ) the status of operation for each server in the cluster.

We conclude that EQUILOAD does not perform as well as SRPT or JSQ for lightly loaded system, or low workload variability, because in these situations the servers assigned to "short" jobs are more loaded than those assigned to "long" jobs: with low workload variability or system load, "long" requests are infrequent.

6.4 EQUILOAD and its locality awareness

A fundamental property of EQUILOAD is that files within a specific size range are sent to the same server. This increases the chances that a given request can be satisfied from the local cache, resulting in a service time that is considerably less than when the requested file is not currently in the cache but only in the disk. In the previous sections, we assumed that files are always fetched from the disk, i.e., that the service time of each request is linearly related to the actual file size. However, the effect of local caching in load balancing of clustered web servers significantly affects performance. LARD is a "locality-aware" scheduling policy that takes into consideration the cache contents of the back-end servers [23] and offers significant performance advantages versus non locality-aware policies.

By its nature, we expect the cache behavior of the back-end servers with EquiLoad to be close to optimal, since requests for the same file are assigned to the same server. To examine EquiLoad's performance with respect to caching we adjust our simulation implementation and ensure that the service time for each request is computed according to the storage location (cache or disk) from where the requested file is fetched. We assume that the service time is two orders of magnitude smaller if the file is fetched from cache instead of local disk. We stress that even in this version of EquiLoad the dispatcher does not require any information about the status of the back-end servers.

For comparison, we simulate a LARD-like load balancing policy where the dispatcher knows the cache contents of each server in the cluster and assigns each incoming request to the least-loaded server that has already the requested file in its local cache, if there is one, otherwise to the least-loaded server. To avoid the possibility of extreme load unbalances, if all servers that have the requested file in their local cache are heavily loaded compared to those that don't have it, the dispatcher assigns the request to the server with the lightest load (more precisely, this happens if the difference between the loads of the least-loaded server among those with the request in the cache and the least-loaded server among those without the request in the cache exceeds a given threshold T). In our experiments, we use different values of the threshold T, to investigate the values yielding the best performance.

The simulations run on a real trace, the day 57 of the World Soccer Cup. Figures 11(a), and Figure 11(b), illustrate the average system queue length and the average system cache hit ratio respectively, as function of arrival rate. [5] We note that EquiLoad handles the increasing load in the system very well and has the highest and constant cache hit ratio. EquiLoad outperforms the LARD-like policy for all threshold values T, and all system loads. The LARD-like policy, although not as good as EquiLoad, also performs well. The higher the values of threshold T, the better the performance of the system for high system loads, while for low system loads the smaller the threshold values the better the performance. However, the LARD policy does not sustain a high cache-hit ratio as the system load increases.

6.5 Behavior of EquiLoad policy in a dynamic environment

We have established the fact that EquiLoad is an effective policy for clustered web servers. It is important to note that the effectiveness of EquiLoad is related to the selection of the intervals $[s_{i-1}, s_i)$, i.e., the range of request sizes allocated to server i. Our policy determines these intervals so that the requests routed to each of the N servers contribute a fraction $1/N$ of the mean $\overline{S}$ of the distribution size. The policy effectiveness is ultimately tied to determining appropriately these

[5] Note that the system can sustain higher arrival intensities comparing to the previous subsections. This is a direct outcome of the fact that the service rates are much higher (by two orders of magnitude) if the file resides in cache

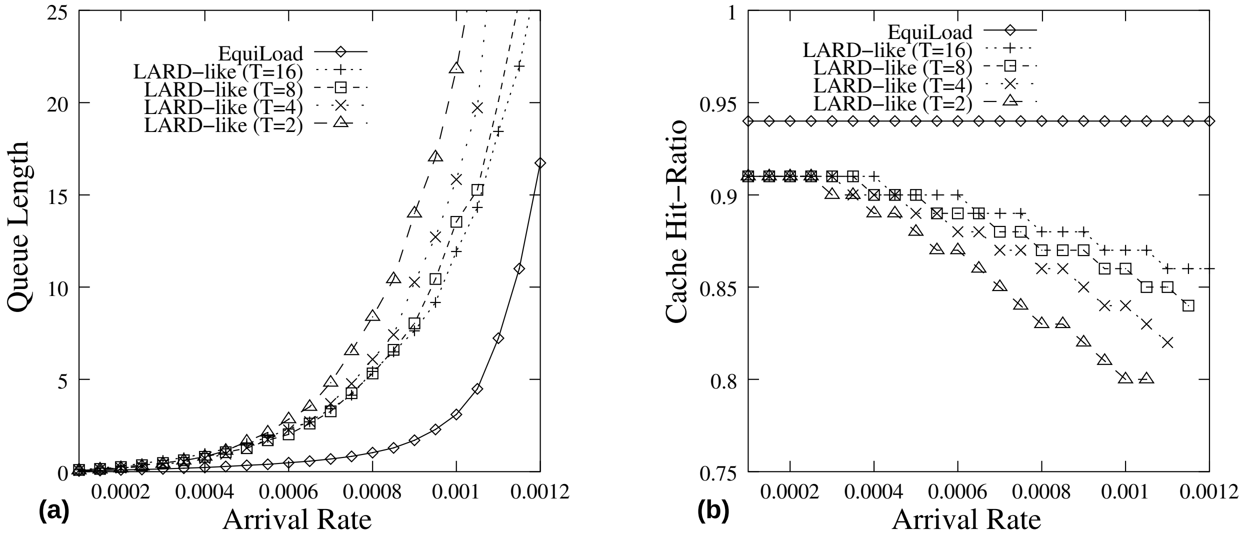

Figure 11: Average queue length and cache hit-ratio of LARD-like and EQUILOAD policies as a function of the arrival rate.

intervals. Given that there is clear evidence that the workload changes not just from day to day but could also change from hour to hour, it becomes increasingly important to be able to dynamically adjust the size intervals accepted by each server.

To assess the importance of selecting the appropriate endpoints for each of the s_i intervals, we selected six traces from the World Cup 1998 web site, each trace representing a distinct day. The means of the distribution of each trace are equal to 2,995, 3,989, 5,145, 5,960, 7,027, and 8,126 bytes for days a, b, c, d, e, and f, respectively. Figure 12 illustrates the values of the s_1, s_2, and s_3 size boundaries of the four servers for the six days.

Figure 13 shows the system expected queue length as a function of the workload arrival intensity for days b, c, d, and f. The solid curve in each graph (labeled "optimal") corresponds to the queue length curve when the optimal partitioning for the particular trace data is used as defined by the EQUILOAD algorithm. Figures 13(a) and (b) show that if the computed intervals correspond to a past portion of the trace that is close enough to the current portion then the expected behavior is close to the optimal. If instead the computed intervals correspond to a trace portion with very different behavior than the current portion (e.g., day f in comparison to days b and c), then the system reaches saturation much faster. Since day f has a much higher mean than both days b and c, its intervals are shifted towards the right and favor load unbalancing in each host's queue.

Figures 13(c) concentrates on the queue build-up when the workload is day d and the policy parameters are computed using the portion of the trace that corresponds to days a, c, e, f, the trace that results when merging the trace portions of days a, b, and c, as well as the trace the that results when merging the trace portions of days b and c. Under the assumption that day a strictly precedes days b, that day b strictly precedes day c, and that day c strictly precedes day d, we see that computing the policy parameters from the very immediate past (i.e., only day's c portion) may be more beneficial than using a more extended past history. The same behavior is further

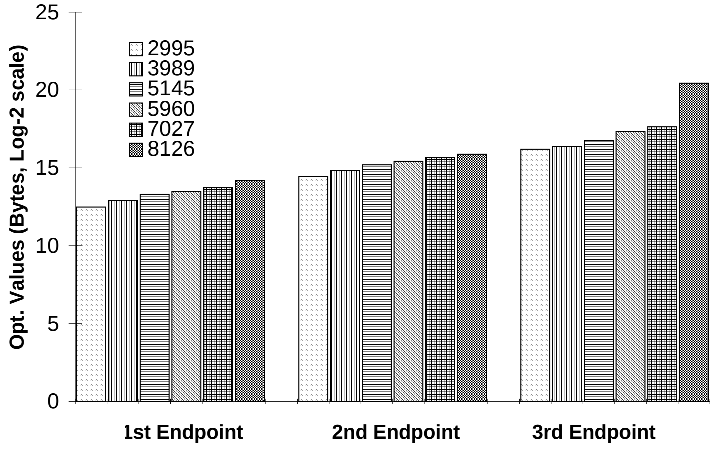

Figure 12: Optimal boundaries for the various workloads.

confirmed in Figure 13(d). Overall, Figure 13 highlights the need for a dynamic on-line algorithm that can intelligently adjust its parameters as a function of the current workload.

7 Conclusions

One of the main challenges in clustered web servers is devising an effective policy to allocate requests to the back-end servers. Size-based policies have been proposed for this purpose, and the one we presented, EquiLoad, falls into this category. In its off-line version, our policy ensures that the expected load experienced by each server, measured in “bytes to be transferred”, is the same for each server, while, at the same time, it achieves the desirable goal of maintaining a high-degree of homogeneity in the requests allocated to each distinct server. The performance gains of EquiLoad are clear especially when the cluster operates under high load or high variability (two characteristics common to clustered web server). Furthermore, the front-end dispatcher of the cluster is not required to maintain any information about the load, the operation, or the cache status of the back-end servers. EquiLoad performs equally to or better than some of the scheduling policies shown to handle well the characteristics of web-related workloads.

However, web loads normally vary over time. Using actual traces, we show how, if the parameters of our policy are computed based on a (past) portion of the trace, they might result in inferior performance for another (future) portion of the same trace. This suggests the need to develop an on-line version of our policy, which dynamically monitors the size distribution of the incoming request load and adjusts itself to achieve optimal performance. We plan to work on this idea in the near future.

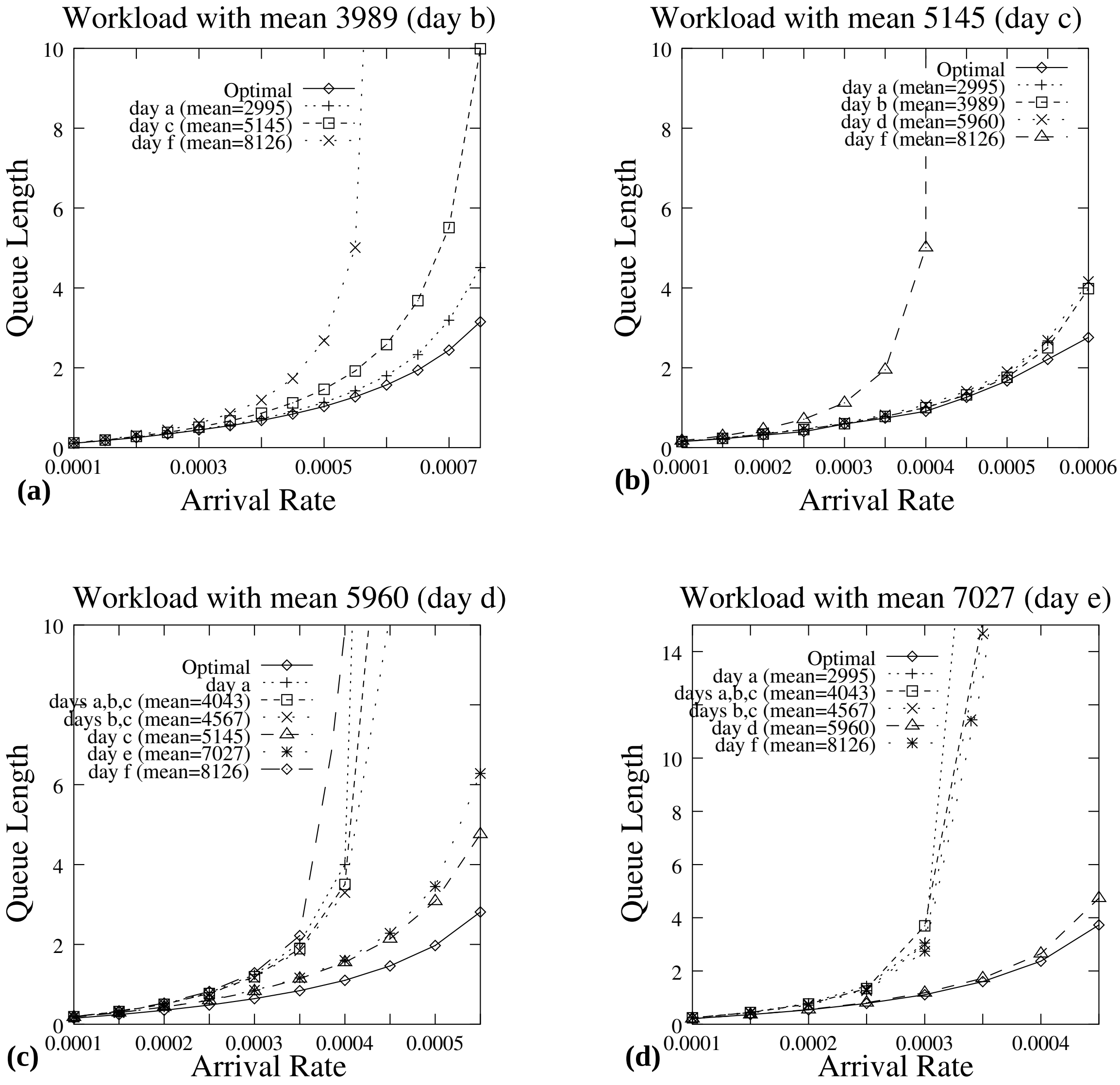

Figure 13: Average queue lengths for various size boundaries for days b, c, d, and e .

Our paper makes also a more theoretical contribution. Since the actual trace is partitioned into disjoint size intervals by our policy, the data in each interval can be fitted to a phase-type distribution using the method of moments so that the resulting model for each back-end server is an M/PH/1 queue. While such queues have well-known solution algorithms such as the matrix-geometric method, we presented a new solution approach that is both simpler and more efficient, where the computation of measures such as the expected queue length requires only the solution of two linear systems.

References

[1] V. A. F. Almeida, A. Bestavros, M. Crovella, and A. Oliviera. Characterizing reference locality in the WWW. In *Proc. 4th Int. Conf. Parallel and Distributed Information Systems (PFIS)*, pages 92–106. IEEE Comp. Soc. Press, Dec. 1996.

[2] M. Arlitt and T. Jin. Workload characterization of the 1998 World Cup Web Site. HP Labs Technical Report, Hewlett Packard, Palo Alto, CA, Sept. 1999.

[3] M. Arlitt and C. L. Williamson. Web serwer workload characterization, the search for invariants. In *Proc. 1996 ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, pages 126–138, Philadelphia, PA, May 1996.

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

[5] P. Barford and M. Crovella. Generating representative Web workloads for network and server performance evaluation. In *Proc. 1998 ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, pages 151–160, Madison, WI, June 1998. ACM Press.

[6] P. Cao and S. Irani. Cost-aware WWW proxy caching algorithms. In *Proc. USENIX Symposium on Internet Technologies and Systems (USITS)*, Monterey, CA, Dec. 1997.

[7] L. Cherkasova and G. Ciardo. Characterizing temporal locality and its impact on Web server performance. In T. Engbersen and E. K. Park, editors, *Proc. IEEE International Conference on Computer Communication and Networks (ICCCN 2000)*, pages 434–441, Las Vegas, NV, USA, Oct. 2000. IEEE Comp. Soc. Press.

[8] G. Ciardo, A. Riska, and E. Smirni. An aggregation-based solution method for M/G/1-type processes. In B. Plateau, W. J. Stewart, and M. Silva, editors, *Numerical Solution of Markov Chains*, pages 21–40. Prensas Universitarias de Zaragoza, Zaragoza, Spain, Sept. 1999.

[9] G. Ciardo and E. Smirni. ETAQA: an efficient technique for the analysis of QBD-processes by aggregation. *Performance Evaluation*, (36-37):71–93, 1999.

[10] M. Colajanni, P. S. Yu, and D. M. Dias. Analysis of task assignment policies in scalable distributed Web servers. *IEEE Trans. Par. and Distr. Syst.*, 9(6):585–600, June 1998.

[11] A. Feldmann and W. Whitt. Fitting mixtures of exponentials to long-tail distributions to analyze network performance models. *Perf. Eval.*, 31(8):963–976, Aug. 1998.

[12] W. K. Grassmann and D. A. Stanford. Matrix analytic methods. In W. K. Grassmann, editor, *Computational Probability*, pages 153–204. Kluwer Academic Publishers, Boston, MA, 2000.

[13] M. Harchol-Balter, N. Bansal, and B. Schroeder. Implementation of SRPT scheduling in web servers. Technical Report CMU-CS-00-170, Carnegie Mellon University, Oct. 2000.

[14] M. Harchol-Balter, M. E. Crovella, and C. D. Murta. On choosing a task assignment policy for a distributed server system. In *Proc. 10th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation*, Lecture Notes in Computer Science 1469, pages 231–242. Springer-Verlag, 1998.

[15] M. Harchol-Balter and A. Downey. Exploiting process lifetime distributions for dynamic load balancing. *ACM Transactions on Computer Systems*, 15(3):253–285, Aug. 1997.

[16] M. A. Johnson and M. R. Taffe. Matching moments to phase distributions: mixtures of Erlang distribution of common order. *Stochastic Models*, 5:711–743, 1989.

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

[18] G. Latouche and V. Ramaswami. *Introduction to Matrix Geometric Methods in Stochastic Modeling.* ASA-SIAM Series on Statistics and Applied Probability. SIAM, Philadelphia PA, 1999.

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

[20] B. Meini. Solving M/G/1 type Markov chains: Recent advances and applications. *Stochastic Models*, 14(1 & 2):479–496, 1998.

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

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

[23] V. S. Pai, M. Aron, G. Banga, M. Svendsen, P. Druschel, W. Zwaenepoel, and E. Nahum. Locality-aware request distribution in cluster-based network servers. In *Proc. ACM 8th Int. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS-VIII)*, pages 205–216, San Jose, CA, Oct. 1998.

[24] V. Ramaswami. A stable recursion for the steady state vector in Markov chains of M/G/1 type. *Commun. Statist.– Stochastic Models*, 4:183–263, 1988.

[25] V. Ramaswami and G. Latouche. A general class of Markov processes with explicit matrix-geometric solutions. *OR Spektrum*, 8:209–218, Aug. 1986.

[26] A. Riska, E. Smirni, and G. Ciardo. Analytic modeling of load balancing policies for tasks with heavy-tailed distributions. In *Proc. Workshop on Software Performance Analysis (WOSP)*, pages 147–157, Ottawa, Canada, Sept. 2000. ACM Press.

[27] C. Singh and R. Billinton. *System Reliability Modelling and Evaluation.* Hutchinson, 1977.

[28] P. M. Snyder and W. J. Stewart. Explicit and iterative numerical approaches to solving queueing models. *Operations Research*, 33(1):183–202, Jan-Feb. 1985.

[29] S. Zhou. A trace driven simulation study of load balancing, *IEEE Transactions on Software Engineering*, 14(9):1327-1341, 1988.

[30] S. Zhou, J. Wang, X. Zheng and P. Delisle. UTOPIA: a load-sharing facility for large heterogenous distributed computing systems. *Software-Practive and Experience*, 23(2):1305–1336, 1993.