

To Queue or Not to Queue:
When Queueing is Better Than Timesharing in a Distributed System

Mor Harchol-Balter*
Laboratory for Computer Science
MIT, NE43-340
Cambridge, MA 02139
harchol@theory.lcs.mit.edu

Mark E. Crovella[†] Cristina D. Murta[‡]
Department of Computer Science
Boston University
Boston, MA 02215
{crovella,murta}@bu.edu

October 31, 1997

BUCS-TR-1997-017

Abstract

We examine the question of whether to employ the first-come-first-served (FCFS) discipline or the processor-sharing (PS) discipline at the hosts in a distributed server system. We are interested in the case in which service times are drawn from a heavy-tailed distribution, and so have very high variability. Traditional wisdom when task sizes are highly variable would prefer the PS discipline, because it allows small tasks to avoid being delayed behind large tasks in a queue. However, we show that system performance can actually be significantly better under FCFS queueing, if each task is assigned to a host based on the task's size. By task assignment, we mean an algorithm that inspects incoming tasks and assigns them to hosts for service. The particular task assignment policy we propose is called SITA-E: Size Interval Task Assignment with Equal Load. Surprisingly, under SITA-E, FCFS queueing typically outperforms the PS discipline by a factor of about two, as measured by mean waiting time and mean slowdown (waiting time of task divided by its service time). We compare the FCFS/SITA-E policy to the processor-sharing case analytically; in addition we compare it to a number of other policies in simulation. We show that the benefits of SITA-E are present even in small-scale distributed systems (four or more hosts). Furthermore, SITA-E is a static policy that does not incorporate feedback knowledge of the state of the hosts, which allows for a simple and scalable implementation.

*Supported by the NSF Postdoctoral Fellowship in the Mathematical Sciences.

[†]Supported in part by NSF Grants CCR-9501822 and CCR-9706685.

[‡]Supported by a grant from CAPES, Brazil. Permanent address: Depto. de Informática, Universidade Federal do Paraná, Curitiba, PR 81531, Brazil. Email: cristina@dcc.ufmg.br.

1 Introduction

System designers are often faced with the question of how to share an important resource such as a processor or a communication channel. One of the simplest of such decisions concerns the question of whether to timeshare the resource, or to require customers to queue and wait for service. Conventional wisdom, well grounded in queueing theory, states that when task sizes are variable, user-perceived performance is better when resources are timeshared; intuition suggests that timesharing allows customers with small service demands to “get out from behind” customers with large service demands.

In this paper we are concerned with a distributed server whose task sizes (service demands) are drawn from a heavy-tailed distribution. A heavy-tailed distribution is one whose tail declines like a power-law, that is, $P[X > x] \sim x^{-\alpha}$ for $0 < \alpha \leq 2$. We are motivated to consider heavy-tailed distributions because they are increasingly being observed in a wide range of computer workloads, see Section 6. These distributions exhibit extremely high variability. Counter to intuition, we show that although service demands show high variability, user-perceived performance can be significantly better when the queueing discipline at individual hosts of the server is first-come-first-served (FCFS) rather than processor-sharing (PS).

In this paper we consider a distributed server model consisting of a collection of hosts and a central node, see Figure 1. Each incoming task to the server arrives at the central node and is immediately assigned to one of the hosts for service. The *task assignment policy* is the rule (algorithm) used for assigning tasks to hosts.

Our result is based on a new task assignment policy, called Size Interval Task Assignment with Equal Load: **SITA-E**. The SITA-E algorithm is based on the following observation: if task size variability were very small ($C^2 < 1$), FCFS would outperform PS for a single queue. Therefore SITA-E’s goal is to reduce the variability of tasks arriving at each host. It achieves this by partitioning tasks among the hosts, according to their sizes. Surprisingly, this method is even able to compensate for the high variability of a heavy-tailed distribution. Furthermore, SITA-E has the additional advantage that it is a static policy, and therefore has a simple implementation.

The metrics by which we judge system performance are user-oriented metrics: mean queue length, mean waiting time for tasks, and most importantly, mean slowdown. Slowdown is the ratio of a task’s waiting time to its service demand. Minimizing slowdown is especially important in a system with highly variable service demands, because minimizing waiting time alone can nonetheless force users to wait for extremely long periods for short tasks to execute.

We analyze the properties of our proposed policy and show that for heavy-tailed task size distributions with α greater than 1, FCFS queueing at the hosts with SITA-E task assignment can significantly outperform a system using PS at the hosts. Typically FCFS/SITA-E reduces mean slowdown and mean queue length by a factor of two over PS. We also show that given a sufficiently large number of hosts, FCFS/SITA-E also results in lower mean waiting time than PS.

Having shown analytically that SITA-E with FCFS queueing outperforms processor sharing, we next examine whether our results are due to the characteristics of SITA-E specifically, or whether they in fact apply to FCFS queueing in general. To do so we simulate a variety of task assignment

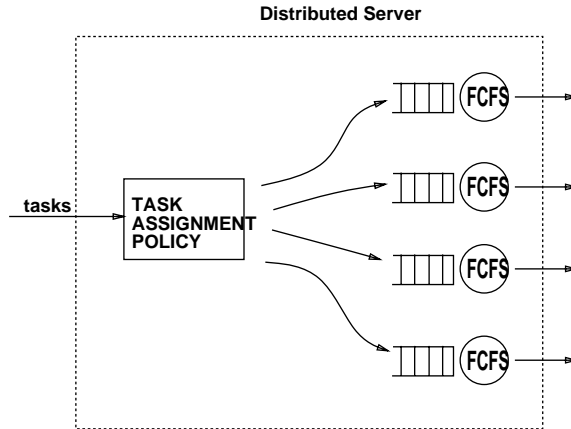


Figure 1: Model for distributed server with task assignment policy (a “server farm”).

policies at the central node, all of which employ FCFS at the hosts. We show that in fact FCFS performs quite poorly under naive task assignment policies such as round-robin or random task assignment; while FCFS with SITA-E performs quite well in the same simulated context. In fact FCFS/SITA-E outperforms FCFS/Random by many orders of magnitude. In addition since SITA-E is a static policy, we ask whether a dynamic policy that takes into account load at individual hosts might perform better than SITA-E. We show surprisingly that for lower α SITA-E is the better of the two.

In this work we have been loosely inspired by an abstraction of a distributed Web server. Recent characterizations of Web server workloads suggest that service times are strongly heavy-tailed [3, 4, 1], while at the same time distributed systems are increasingly being proposed as architectures for high performance Web servers [12, 8, 15, 17]. The two aspects of Web servers that we incorporate in our model are that, first, Web file sizes have been found to be heavy-tailed, and second, task sizes may often be inferred from the name of the file being requested. However, our results have general applicability to any distributed system with centralized task routing.

The paper is organized as follows. The next section describes the details of the distributed system model we assume. Section 3 then describes the SITA-E policy and its analysis. Section 4 presents analytic results comparing FCFS/SITA-E with the PS case, and Section 5 presents simulation results comparing SITA-E with other task assignment policies, including dynamic task assignment, assuming FCFS queuing at the nodes. Section 6 presents other related work; and Section 7 presents our conclusions.

2 Distributed Server Model

Our model of a distributed server system assumes h identical server hosts. Tasks arrive to the system according to a Poisson process with rate λ . A task corresponds to a request for service. A *task-assignment policy* is an algorithm which assigns each task to one of the hosts for service.

Heavy-Tailed distributions. One of the motivating factors in this work is the recent observation that file sizes on Web servers often exhibit *heavy-tailed* distributions [1, 4, 3]. We say here that a random variable X follows a heavy-tailed distribution (with tail index α) if

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

where $a \sim b$ means that $\lim_{x \rightarrow \infty} a/b = c$ for some constant c . The simplest heavy-tailed distribution is the *Pareto* distribution, with probability mass function

$$f(x) = \alpha k^\alpha x^{-\alpha-1}, \quad \alpha, k > 0, \quad x \geq k,$$

and cumulative distribution function

$$F(x) = P[X \leq x] = 1 - (k/x)^\alpha.$$

Heavy-tailed distributions are characterized by extremely high variability, which increases sharply as α decreases. Such a distribution has infinite variance; and if $\alpha \leq 1$ the distribution has infinite mean.

For Web file size measurements, typical values of the tail index α seem to be in range of 1.1 to 1.3 [4, 3]. These values of α are so low as to motivate particular focus on analyzing the effects of high variability in file size, and on developing resource management policies that specifically address high variability in file size, which we do in this paper.

Task sizes. We assume that task sizes show some maximum (but large) value. Note that this would be the expected case for a Web server, which would have some largest file. As a result, we model task sizes using a distribution that follows a power law, but has an upper bound. We refer to this distribution as a *Bounded Pareto*. It is characterized by three parameters: α , the exponent of the power law; k , the smallest possible observation; and p , the largest possible observation. The probability mass function for the Bounded Pareto $B(k, p, \alpha)$ is defined as:

$$f(x) = \frac{\alpha k^\alpha}{1 - (k/p)^\alpha} x^{-\alpha-1} \quad k \leq x \leq p. \quad (1)$$

The shape of $f(x)$ is shown in Figure 2 (left). If X is a random variable drawn from a $B(k, p, \alpha)$ distribution, then the moments of X are given by:²

$$\mathbf{E} \{ X^j \} = \frac{\alpha k^\alpha (k^{j-\alpha} - p^{j-\alpha})}{(\alpha - j) (1 - (k/p)^\alpha)} \quad (2)$$

Note that the Bounded Pareto distribution has all its moments finite. Thus, it is not a heavy-tailed distribution in the sense we have defined above. However, this distribution will still show high variability if $k \ll p$. For example, Figure 2 (right) shows the second moment $\mathbf{E} \{ X^2 \}$ of this distribution as a function of α for $p = 10^{10}$, where k is chosen to keep $\mathbf{E} \{ X \}$ constant at 3000 ($0 < k \leq 1500$). The figure shows that the second moment explodes exponentially as α declines.

Finally, we make the additional simplifying assumption that the service times of the tasks are independent; in particular, the sizes of the i th and $i + 1$ st arriving tasks are uncorrelated.

²This formula applies when $\alpha \neq j$; expressions for other cases are given in Theorem 5.

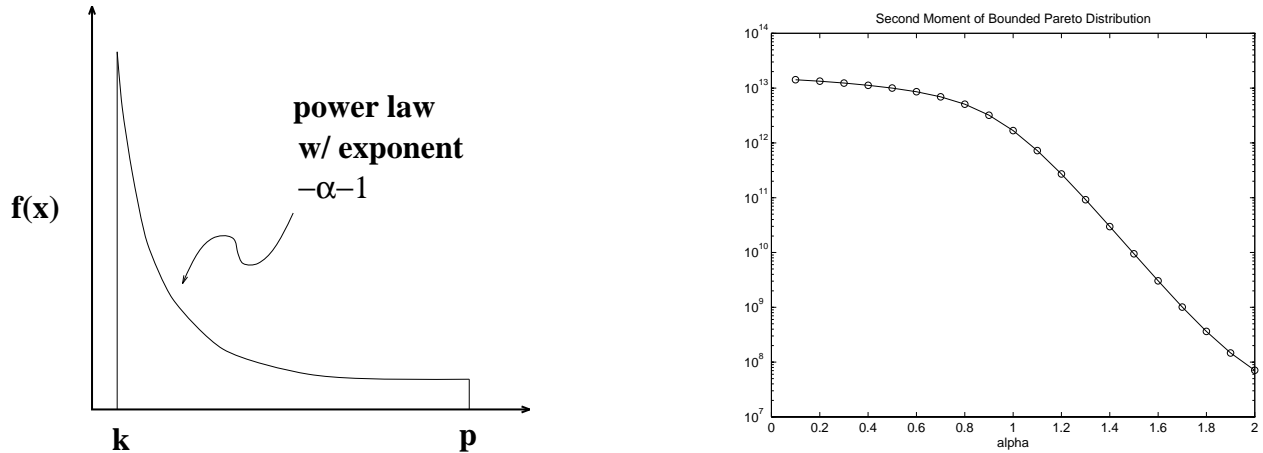


Figure 2: Parameters of the Bounded Pareto Distribution (left); Second Moment of $B(k, 10^{10}, \alpha)$ as a function of α , when $\mathbf{E}\{X\} = 3000$ (right).

Known task sizes. We assume that a centralized task router is able to inspect the service demands of each arriving task. The fact that in this model of a distributed server a task’s service time is known at the time of its arrival greatly differentiates this model from a typical network of workstations load balancing model where task sizes are *not* known in advance and must be estimated as the task runs (*e.g.*, see [9]). The design of a centralized router for distributed Web servers that can inspect task service demands is described in [10].

Performance metrics. The effectiveness of a task assignment scheme will be measured in terms of mean waiting time, mean queue length, and mean slowdown, where a task’s slowdown is its waiting time divided by its service demand. All means are per-task averages.

Of these three metrics, we consider slowdown to be most important, because it is desirable that a task’s delay be proportional to its size. That is, in a system in which task sizes are highly variable, users are likely to anticipate short delays for short tasks, and are likely to tolerate long delays for longer tasks. This may be especially true in the case of Web requests, where users may typically have a feeling for the rough order of magnitude of their request.

Notation. In analyzing the distributed system model presented here we use the notation shown in Table 1.

W	Waiting time for tasks arriving at the system	W_i	Waiting time for only those tasks at the i th host
S	Slowdown for tasks arriving at the system	S_i	Slowdown for only those tasks at the i th host
Q	Number of tasks an arriving task sees ahead of it	Q_i	Number of tasks at the i th host on task arrival
X	Service time (execution time) for tasks	X_i	Execution time for those tasks at the i th host
λ	Arrival rate into the system	λ_i	The arrival rate at the i th host
p_i	The probability that a task is sent to the i th host; that is $\lambda_i = \lambda \cdot p_i$, where p_i is defined differently for the different task assignment schemes		

Table 1: Notation Used in the Paper

3 SITA-E

In this section we introduce the SITA-E task assignment policy for a distributed system of FCFS hosts. In Section 3.1 we motivate the SITA-E idea. We discuss why a single FCFS host has poor performance under a heavy-tailed workload as compared with a PS host, and why a distributed system of FCFS hosts under commonly-used task assignment policies would also have poor performance as compared with a distributed system of PS hosts. This leads to the intuition behind the FCFS/SITA-E task assignment policy. Section 3.2 defines the SITA-E algorithm and provides analysis under all heavy-tailed workloads, including closed-form expressions for the first and second moments of several performance metrics.

3.1 SITA-E Motivation: Comparing queueing and non-queueing approaches

Traditionally, under a highly variable task size distribution, FCFS service order is thought to result in poor performance because small jobs which are queued behind larger jobs have to first wait for those large jobs to complete before they can receive service. In contrast, for PS service order all jobs are serviced simultaneously, so smaller jobs complete quickly, resulting in much lower mean slowdown.

To make the above argument more precise, consider Theorem 1 below which analyzes the M/G/1 FCFS queue using the Pollaczek-Kinchin formula:

Theorem 1 *Given an M/G/1 FCFS queue, where the arrival process has rate λ , X denotes the service time distribution, and ρ denotes the utilization ($\rho = \lambda \mathbf{E}\{X\}$). Let W be a task's waiting time in queue, S be its slowdown, and Q be the queue length on its arrival. Then,*

$$\begin{aligned} \mathbf{E}\{W\} &= \frac{\lambda \mathbf{E}\{X^2\}}{2(1-\rho)} & \text{var}(W) &= \mathbf{E}\{W\}^2 + \frac{\lambda \mathbf{E}\{X^3\}}{3(1-\rho)} \\ \mathbf{E}\{S\} &= \mathbf{E}\{W/X\} = \mathbf{E}\{W\} \cdot \mathbf{E}\{X^{-1}\} & \mathbf{E}\{S^2\} &= \mathbf{E}\{W^2\} \cdot \mathbf{E}\{X^{-2}\} \\ \mathbf{E}\{Q\} &= \lambda \mathbf{E}\{W\} \end{aligned}$$

Proof: The slowdown formulas follow from the fact that W and X are independent for a FCFS queue, and the queue size follows from Little's formula. ■

Observe that every metric for the FCFS queue is dependent on $\mathbf{E}\{X^2\}$, the second moment of the service time. If the workload is heavy-tailed, the second moment of the service time explodes, as shown in Figure 2. In contrast, Theorem 2 below show that the second moment of the service time plays no role for a PS queue.

Theorem 2 [13] *Consider an M/G/1 PS queue with arrival rate λ :*

$$\mathbf{E}\{W\} = \frac{\rho}{1-\rho} \mathbf{E}\{X\}$$

$$\mathbf{E}\{S\} = \frac{\rho}{1 - \rho}$$

Now consider a distributed server employing FCFS queueing at the hosts. To start, consider the natural task assignment policy, RANDOM, whereby each arriving task is assigned to host i with probability $1/h$, where h is the number of hosts. Then the arrival process into each host is still a Poisson process and the moments of the service time at each host are the same as that in the original workload. Consequently, RANDOM's performance on a distributed FCFS server is exactly that of a single FCFS host; under a heavy-tailed, highly variable task size distribution this server's performance would be poor.

By contrast, consider a distributed PS server using *any* task assignment policy which achieves balanced load and results in a Poisson arrival process at the hosts. Then the load at every server is ρ , and by Theorem 2 above, the mean slowdown at every server is just $\rho/(1 - \rho)$, which is a constant independent of the task size distribution.

Given these examples, it seems doubtful that a distributed FCFS server can outperform a distributed PS server in the case of a heavy-tailed workload. However, recall the following simple fact: if the variance of the workload were very low (close to zero), rather than very high, a FCFS server would have *only half* the waiting time of a PS server (*i.e.*, an M/D/1 FCFS queue has only half the waiting time of an M/G/1 PS queue). The SITA-E task assignment policy exploits the distributed server to take advantage of this fact. That is, although the original workload has a very high variance in task size, SITA-E allocates the tasks in such a way that most of the FCFS hosts receive a reduced-variance workload. More importantly, most of the tasks will be sent to these low-variance hosts.

In fact, even at these reduced-variance hosts, variance is not always close to zero. However there is a second effect working in favor of SITA-E: under SITA-E, small tasks are only queued with other small tasks. That is, a small task never waits in queue behind a large task. This has the positive consequence of low mean slowdown for the system.

3.2 SITA-E Definition and Analysis

Size Interval Task Assignment with Equal Load (SITA-E) relies on the assumption that the distribution of the size of incoming requests is known, and furthermore that this distribution has *finite* mean M . In SITA-E, each host only accepts tasks whose size falls within a specified size interval, where this size range is chosen such that each host receives equal work in expectation. Specifically, let $F(x) = \mathbf{P}\{X \leq x\}$ denote the cumulative distribution function of request sizes and $f(x) = \mathbf{P}\{X = x\}$ the corresponding density function. Let k denote the smallest possible request size, p (possibly equal to infinity) denote the largest possible request size, and h be the number of hosts. Then we determine "cutoff points" x_i , $i = 0 \dots h$ where $k = x_0 < x_1 < x_2 < \dots < x_{h-1} < x_h = p$, such that

$$\int_{x_0=k}^{x_1} x \cdot dF(x) = \int_{x_1}^{x_2} x \cdot dF(x) = \int_{x_2}^{x_3} x \cdot dF(x) = \dots = \int_{x_{h-1}}^{x_h=p} x \cdot dF(x) = \frac{M}{h} = \frac{\int_k^p x \cdot dF(x)}{h}$$

and assign to the i th host all files ranging in size from x_{i-1} to x_i .

SITA-E as defined can be applied to *any* task size distribution with finite mean. In the remainder of the paper we will always assume the task size distribution is the Bounded Pareto distribution, $B(k, p, \alpha)$, and we will demonstrate SITA-E's benefits in that case. The advantage of this particular heavy-tailed distribution is that it allows us to easily vary α and thereby study how the performance of SITA-E is affected by changes in the variance of task sizes. The performance benefits of SITA-E are however not limited to only those systems with workloads following the Bounded Pareto distribution.

The theorem below shows that there is a simple formula for x_i , $i = 0 \dots h$ in the case where the request distribution is $B(k, p, \alpha)$:

Theorem 3 *Let X be a random variable which follows the $B(k, p, \alpha)$ distribution where $0 < \alpha \leq 2$ and k and p are positive constants with $k < p$. Then, if $\alpha \neq 1$, defining*

$$x_i = \left(\frac{(h-i)}{h} k^{1-\alpha} + \frac{i}{h} p^{1-\alpha} \right)^{\frac{1}{1-\alpha}}, \text{ for } i = 0 \dots h,$$

we have $k = x_0 < x_1 < x_2 < \dots < x_{h-1} < x_h = p$, and

$$\int_{x_0=k}^{x_1} x \cdot f(x) = \int_{x_1}^{x_2} x \cdot f(x) = \int_{x_2}^{x_3} x \cdot f(x) = \dots = \int_{x_{h-1}}^{x_h=p} x \cdot f(x) = \frac{\mathbf{E}\{X\}}{h}$$

If $\alpha = 1$, the corresponding definition is:

$$x_i = k \left(\frac{p}{k} \right)^{\frac{i}{h}}, \text{ for } i = 0 \dots h$$

Proof: See Appendix. ■

To analyze the system of h hosts with a SITA-E load balancing policy, we make two observations:

1. The arrival process into each queue is in fact Poisson. To see this, define p_i to be the probability that an arriving task has size between x_{i-1} and x_i . Because the task sizes are uncorrelated, p_i is well-defined. Now, since the arrival process into the system is Poisson with rate λ , it follows that the arrival process into host i is Poisson with rate λp_i .
2. Each queue may be solved independently as an independent M/G/1 queue where the service time distribution at host i is $B(x_{i-1}, x_i, \alpha)$. That is, partitioning the Bounded Pareto distribution into contiguous regions and renormalizing each of the resulting regions to unit probability yields a new set of Bounded Pareto distributions.

The above discussion is summarized in the theorem below:

Theorem 4 *Assume task arrivals into a system of h hosts with task sizes drawn from the $B(k, p, \alpha)$ distribution. Let X be a random variable from this distribution. Assume the load balancing scheme*

is SITA-E, where the x_i 's are defined according to Theorem 3. Then each queue may be analyzed independently, where the i th queue is an M/G/1 queue with arrival rate λp_i and service times X_i , as defined below.

$$p_i = \int_{x_{i-1}}^{x_i} f(x) dx = \int_{x_{i-1}}^{x_i} \frac{\alpha k^\alpha x^{-\alpha-1}}{(1 - (k/p)^\alpha)} dx = \frac{k^\alpha}{1 - (k/p)^\alpha} (x_{i-1}^{-\alpha} - x_i^{-\alpha})$$

X_i has a $B(x_{i-1}, x_i, \alpha)$ distribution. The moments of X_i are given by Equation 2, with $k = x_{i-1}$ and $p = x_i$. Furthermore, the load at the i th queue, ρ_i , is equal to the overall load, ρ .

Proof: See Appendix. ■

The theorem below presents the full analysis of a distributed FCFS server with SITA-E task assignment. As defined in Section 2, the random variables W , S , Q , and X refer to the waiting time, slowdown, queue size and service time, respectively, for the h host system. The variables W_i , S_i , Q_i , and X_i are the corresponding random variables for just the i th queue (i th host) of the system.

Theorem 5 *Given a system of h hosts with a Poisson arrival process with rate λ , task sizes drawn from a Bounded Pareto distribution $B(k, p, \alpha)$, and employing the SITA-E task assignment policy, then:*

$$\begin{aligned} x_i &= \begin{cases} \left(\frac{(h-i)}{h} k^{1-\alpha} + \frac{i}{h} p^{1-\alpha} \right)^{\frac{1}{1-\alpha}} & \text{if } \alpha \neq 1 \\ k \left(\frac{p}{k} \right)^{\frac{i}{h}} & \text{if } \alpha = 1 \end{cases} \\ p_i &= \frac{k^\alpha}{1 - (k/p)^\alpha} (x_{i-1}^{-\alpha} - x_i^{-\alpha}) \\ \mathbf{E} \{ X_i^j \} &= \begin{cases} \frac{\alpha x_{i-1}^\alpha (x_{i-1}^{j-\alpha} - x_i^{j-\alpha})}{(\alpha-j)(1 - (x_{i-1}/x_i)^\alpha)} & \text{if } \alpha \neq j \\ \frac{x_{i-1} x_i}{x_i - x_{i-1}} (\ln x_i - \ln x_{i-1}) & \text{if } \alpha = j = 1 \end{cases} \\ \mathbf{E} \{ 1/X_i^j \} &= \mathbf{E} \{ X_i^{-j} \} \\ \mathbf{E} \{ W_i \} &= \frac{\lambda p_i \mathbf{E} \{ X_i^2 \}}{2(1 - \lambda p_i \mathbf{E} \{ X_i \})} \\ \mathbf{E} \{ W \} &= \sum_{i=1}^h p_i \mathbf{E} \{ W_i \} \\ \text{var}(W_i) &= \mathbf{E} \{ W_i \}^2 + \frac{\lambda p_i \mathbf{E} \{ X_i^3 \}}{3(1 - \lambda p_i \mathbf{E} \{ X_i \})} \\ \mathbf{E} \{ W_i^2 \} &= \text{var}(W_i) + (\mathbf{E} \{ W_i \})^2 \\ \mathbf{E} \{ W^2 \} &= \sum_{i=1}^h p_i \mathbf{E} \{ W_i^2 \} \end{aligned}$$

$$\begin{aligned}
\text{var}(W) &= \mathbf{E}\{W^2\} - \mathbf{E}\{W\}^2 \\
\mathbf{E}\{Q_i\} &= \lambda p_i \mathbf{E}\{W_i\} \\
\mathbf{E}\{Q\} &= \sum_{i=1}^h p_i \mathbf{E}\{Q_i\} \\
\mathbf{E}\{S_i\} &= \mathbf{E}\left\{\frac{W_i}{X_i}\right\} = \mathbf{E}\{W_i\} \cdot \mathbf{E}\{1/X_i\} \\
\mathbf{E}\{S\} &= \sum_{i=1}^h p_i \mathbf{E}\{S_i\} \\
\mathbf{E}\{S_i^2\} &= \mathbf{E}\{W_i^2\} \cdot \mathbf{E}\{1/X_i^2\} \\
\mathbf{E}\{S^2\} &= \sum_{i=1}^h p_i \mathbf{E}\{S_i^2\} \\
\text{var}(S) &= \mathbf{E}\{S^2\} - \mathbf{E}\{S\}^2
\end{aligned}$$

Proof: See Appendix. ■

4 Analytic comparison of PS and FCFS/SITA-E

In the previous section we derived performance metric formulas for a distributed FCFS server with SITA-E task assignment (FCFS/SITA-E). We also derived formulas for a distributed FCFS server with RANDOM task assignment (FCFS/RANDOM) and for a distributed PS server with any equal-load task assignment scheme (PS/EQ-LOAD, or PS for short). In this section we will depict the results of the previous section graphically and provide intuition. For each performance metric, we will compare the performance of the following distributed systems: FCFS/SITA-E, FCFS/RANDOM, and PS/EQ-LOAD.

4.1 Parameteric ranges used in graphs

In the graphs below, we choose specific values or ranges for the system parameters, as summarized in Table 3. These values are chosen with the aim of being realistic for a medium-scale distributed system with heavy-tailed workload, such as a distributed Web server.

We choose the number of hosts h to be 8, representative of a medium-scale distributed system; later we show the effects of varying h . We choose a load ρ of 0.8 representing a heavily loaded server. The mean task execution time, $\mathbf{E}\{X\}$, is chosen to be 3000 time units. This could be interpreted as the number of bytes in an average Web page [5]. The arrival rate λ is then determined by:

$$\lambda = \rho \cdot \frac{1}{\mathbf{E}\{X\}} \cdot h = .0021 \text{ tasks/unit time}$$

Number of hosts	$h = 8$
System load	$\rho = .8$
Mean service time	$\mathbf{E}\{X\} = 3000$ time units
Task arrival rate	$\lambda = .0021$ tasks/unit time
Maximum task service time	$p = 10^{10}$ time units
α parameter	$0 < \alpha \leq 2$
Minimum task service time	chosen so that mean task service time stays constant as α varies ($0 < k \leq 1500$)

Figure 3: Parameters used in evaluating task assignment policies

For completeness we consider a range of α from 0 to 2; as discussed in Section 2 this means that task variability spans a wide range from moderately variable ($\alpha = 2$) to highly variable ($\alpha \approx 0$). Recent measurements indicate that a Web server might experience α in the range of approximately 1.1 to 1.3 [4].

If k and p are held constant, an increase in α results in a decrease in the mean task size. Therefore, to hold $\mathbf{E}\{X\}$ constant, we decrease k , the lower bound on the $B(k, p, \alpha)$ distribution, as we increase α . Changing p has a relatively less pronounced effect on $\mathbf{E}\{X\}$, so we choose to hold p constant at 10^{10} . Again, this value corresponds to a reasonable value for a largest file on a Web server, measured in bytes. Thus α and k are throughout defined in pairs, where

$$3000 = \mathbf{E}\{X\} = \frac{\alpha k^\alpha (k^{1-\alpha} - p^{1-\alpha})}{(\alpha - 1)(1 - (k/p)^\alpha)}. \quad (3)$$

As α ranges from 0 to 2, k ranges from just slightly greater than 0 (time units) to 1500 (time units).

4.2 System performance under PS, FCFS/RANDOM and FCFS/SITA-E policies

From the formulas in Theorems 1, 2 and 5 we obtain the performance metric values for FCFS/RANDOM, PS/EQ-LOAD, and FCFS/SITA-E. SITA-E's performance is best in the the range $1 \leq \alpha \leq 2$; results for this range are shown in Figure 4. In this figure, results for an 8-host system are shown in the left-hand column, while results for a 32-host system are shown in the right-hand column; the top row shows mean waiting time, the middle row shows mean slowdown, and the bottom row shows mean queue length.

Figure 4 shows important features of SITA-E's performance. First, with respect to mean slowdown and mean queue length, FCFS/SITA-E outperforms PS by roughly a factor of two for most values of α in the range 1 to 2. Secondly, FCFS/SITA-E is surprisingly insensitive to the value of α in this range; this is despite the fact that varying α causes drastic changes in service time variability (as shown graphically in Figure 2). Finally, note that while waiting time under FCFS/SITA-E is not typically better than that of PS for an 8 host system, increasing the system size to 32 hosts causes waiting time under FCFS/SITA-E to consistently outperform that of the PS system. In fact, for some values of α , waiting time achieves its minimal value—half that of the PS case.

However, this relationship between FCFS/SITA-E and PS changes for values of α less than 1. The performance of both systems, along with that of the FCFS/RANDOM system, are shown for

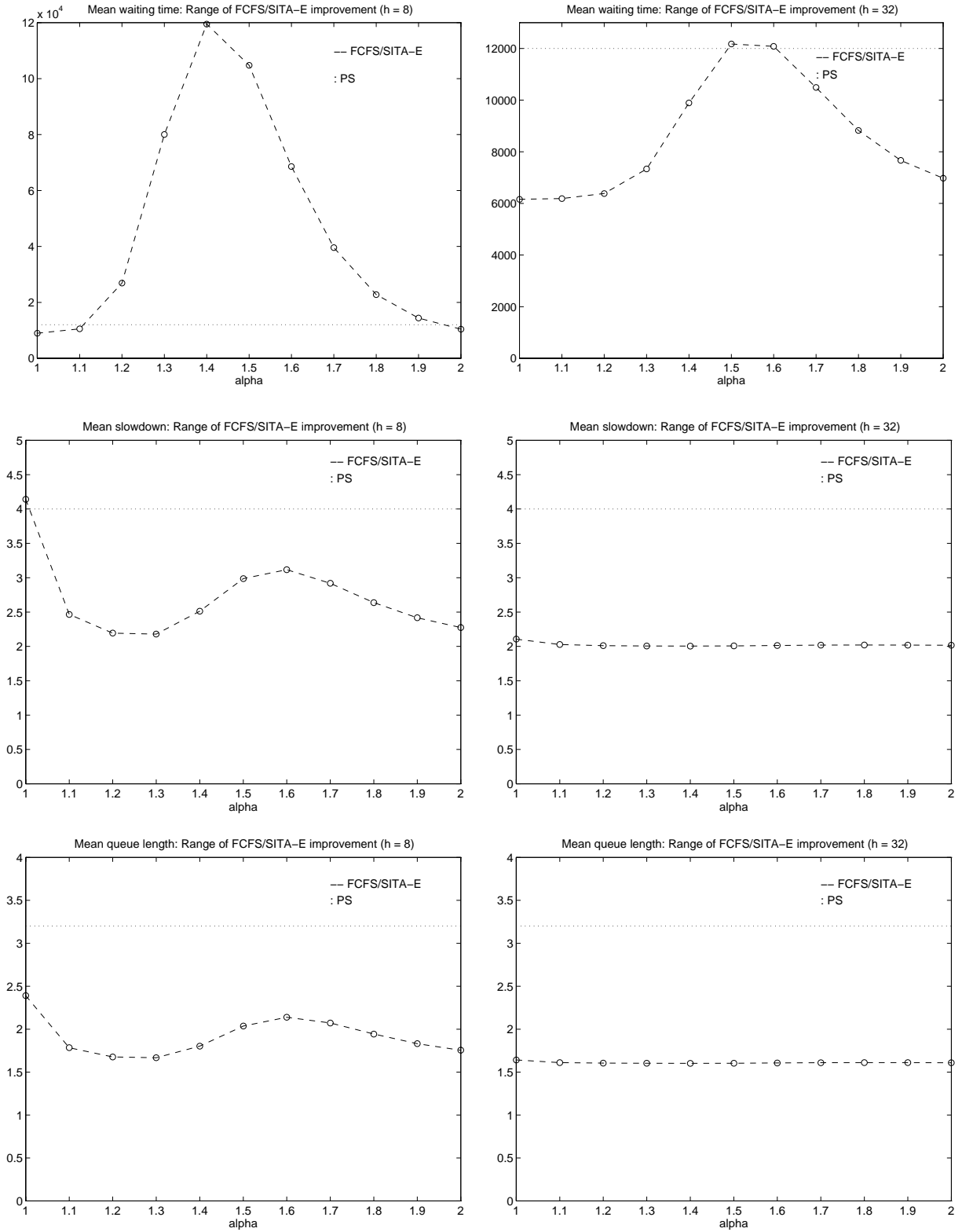


Figure 4: Predicted values of performance metrics for FCFS/SITA-E, FCFS/RANDOM, and PS/EQ-LOAD for $1 \leq \alpha \leq 2$ with 8 hosts (left) and 32 hosts (right).

values of α in the larger range of $0 < \alpha \leq 2$ in Figure 5, on a logarithmic scale. When α grows very small, SITA-E is no longer able to provide acceptable system performance, and its performance degrades rapidly in a fashion similar to the simple FCFS/RANDOM system.

To understand the comparative performance, first consider each system separately. The performance of PS/EQ-LOAD is straightforward. PS/EQ-LOAD is not at all affected by α ; the mean slowdown and queue length depend only on the load, and the mean waiting time depends only on the load and the mean task size.

The performance of FCFS/Random is also straightforward. FCFS/Random is directly dependent on $\mathbf{E}\{X^2\}$, the second moment of the task size distribution. $\mathbf{E}\{X^2\}$ increases as α drops, as shown in Figure 2, which explains why the performance of FCFS/Random gets steadily worse as α drops. Observe that at $\alpha = 2$ the performance of FCFS/RANDOM is about a factor of ten worse than that of PS/EQ-LOAD on all metrics. If we were to increase α beyond 2, the performance of FCFS/RANDOM would continue to improve and at some point (as service time variance approaches zero) the performance of FCFS/RANDOM would be superior to that of PS/EQ-LOAD.

The performance of FCFS/SITA-E is more complex. We separate the discussion of SITA-E's performance into two parts: mean waiting time under SITA-E, and mean slowdown under SITA-E.

Understanding SITA-E's Mean Waiting Time. In order to understand the difference in $\mathbf{E}\{W\}$ for FCFS/SITA-E compared to PS as shown in Figures 4 and 5, it is helpful to consider the squared coefficient of variation ($C^2 = \sigma^2/\mu^2$) of the service times at each host in the SITA-E system. C^2 is an informative metric in this case because when the service time $C^2 < 1$, the waiting time in an M/G/1 FCFS queue is less than that in the corresponding M/G/1 PS queue.

Figure 6 shows the values of C^2 at each host in an 8 host system, as a function of α . Lines are labeled with names of the hosts they correspond to; note that the order of lines from top to bottom inverts at the point $\alpha = 1$.

The overall mean waiting time of the system can be understood in terms of hosts numbered 1 and 8 (the extreme hosts). Host 1 has the smallest tasks, and so has a very large number of tasks. Host 8 has the largest tasks, and so has very few tasks. When $\alpha > 1$, only host 8 has $C^2 > 1$; thus nearly all tasks go to hosts which have performance better than that of a PS host. However host 8's C^2 value is quite high (over 100 when $\alpha = 1.4$), and so its behavior still strongly affects the overall system; this can be seen in the similar shape of the host 8 C^2 curve and the overall waiting time curve in Figure 4.

On the other hand, for $\alpha < 1$, all hosts except host 1 have C^2 values less than 1. As α declines, C^2 for host 1 explodes in a manner similar to the second moment of the entire distribution (Figure 2). Since host 1 still has more tasks than the other hosts, this explosion in C^2 has devastating consequences for the system as a whole.

Understanding SITA-E's Mean Slowdown. Turning to slowdown, we observe that SITA-E improves slowdown relative to PS to a greater degree than it improves waiting time. The reason

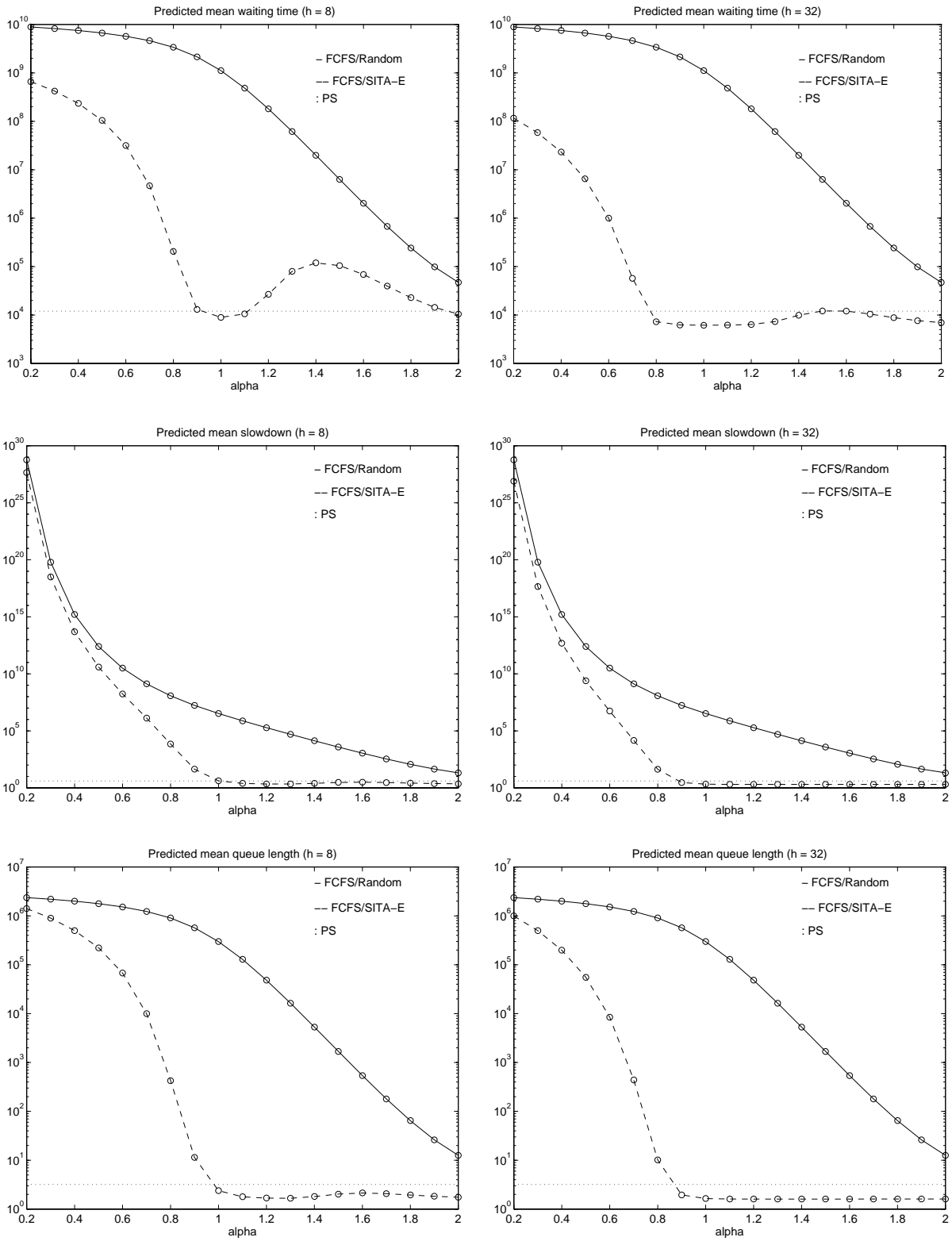


Figure 5: Predicted values of performance metrics for FCFS/SITA-E, FCFS/RANDOM, and PS/EQ-LOAD for $0 < \alpha \leq 2$ with 8 hosts (left) and 32 hosts (right).

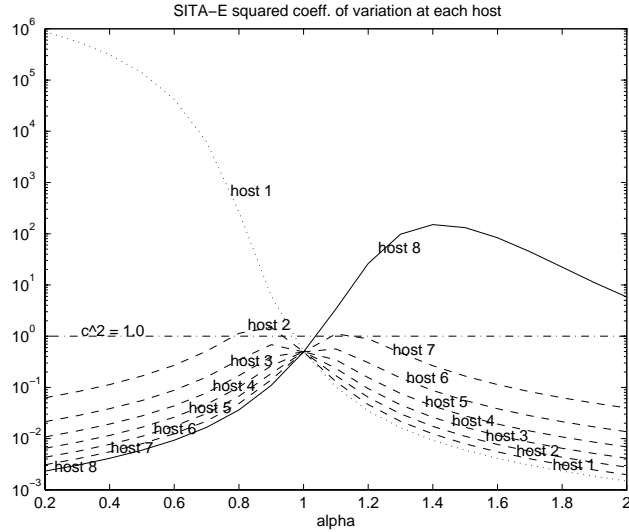


Figure 6: SITA-E values for C^2 on each host (1 through 8), as a function of α .

behind this difference is reflected in the fact that lines for lower numbered hosts in Figure 6 are toward the bottom of the figure when $\alpha > 1$. This is an indication that the mean waiting time on lower numbered hosts is less than that on higher numbered hosts. Now since SITA-E assigns smaller tasks to smaller numbered hosts, it improves slowdown considerably. Thus the second beneficial feature of SITA-E is that tasks tend to be queued with their peers (that is, other tasks of nearly equal service demand).

To summarize, in Figures 4 and 5 the plots for mean waiting time are essentially a measure of the effectiveness of SITA-E in reducing variability on the low numbered hosts, which receive the majority of the tasks. In contrast, the difference between the plots for mean waiting time and the plots of mean slowdown is an indication of the effectiveness of SITA-E in queueing tasks with their peers.

4.3 The effect of varying the number of hosts

In this section we ask the question: how many hosts are necessary for FCFS/SITA-E to show good performance? Figure 7 shows FCFS/SITA-E, compared with FCFS/RANDOM and PS, as a function of h . In this figure α has been held constant at 1.2, λ varies as a function of h to keep ρ constant at 0.8 (determined by Equation 3), and all other parameters are set as in Table 3.

As shown in Figure 7, for $h = 1$, FCFS/RANDOM and FCFS/SITA-E are equivalent. Also, as expected, FCFS/RANDOM and PS/EQ-LOAD do not change as h varies, but increasing h does improve mean waiting time for FCFS/SITA-E. For these parameter settings, FCFS/SITA-E outperforms PS on the mean slowdown and queue length metrics when $h \geq 4$, and outperforms PS on the mean waiting time metric when $h > 12$. Again, the fact that the threshold is so much smaller for mean slowdown as compared to mean waiting time reflects the additional benefit that accrues to slowdown from reducing variability at the hosts *as well as* queueing tasks with their

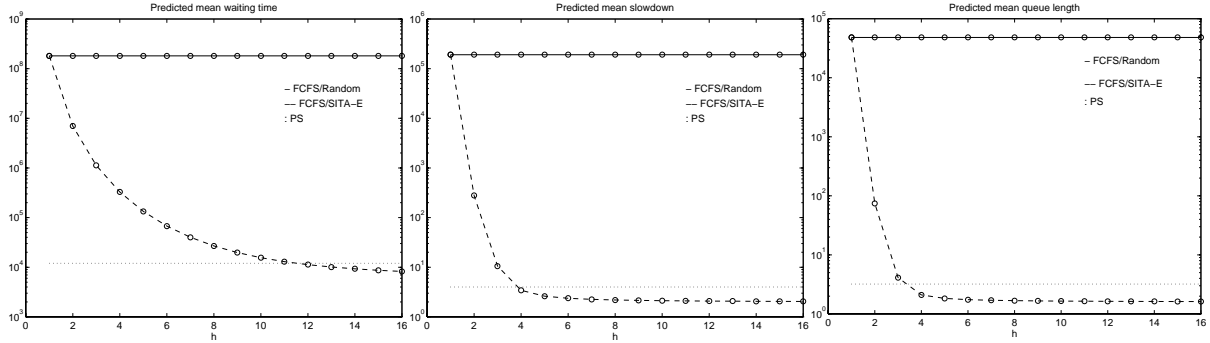


Figure 7: FCFS/RANDOM, FCFS/SITA-E, and PS/EQ-LOAD as a function of h , where $\alpha = 1.2$ and $\rho = .8$ and $\mathbf{E}\{X\} = 3000$ and λ varies as a function of h

peers.

The improvement in mean waiting time as a function of h is comparatively gradual. However the effectiveness of SITA-E on mean slowdown is demonstrated in the sharp decline in slowdown even when system sizes are small. For example, simply increasing h from 1 to 2 improves slowdown dramatically – by a factor of almost 1000.

5 Comparing SITA-E to other Queuing Policies

In the previous section we showed that a distributed FCFS server with the SITA-E task assignment policy (FCFS/SITA-E) can sometimes outperform a distributed PS server running any load balancing task assignment policy (PS/EQ-LOAD) even under the conditions of a heavy-tailed workload where FCFS service order is normally a poor choice. In this section we ask: is this improvement due to SITA-E, or would another task assignment policy combined with FCFS at the hosts do as well or better?

In particular, we ask the question of whether a DYNAMIC task assignment policy, which has the advantage over SITA-E of knowing the load at each host, might outperform SITA-E on a distributed FCFS server. Analysis of DYNAMIC policies is difficult, so we resort to simulation. In our simulation we assume that the router in the DYNAMIC policy has perfect knowledge of the current (instantaneous) load at each host.

5.1 Simulation Details

We simulate a distributed FCFS server with various task assignment policies. The task arrival process is again Poisson. Each arrival has a service time drawn from a Bounded Pareto distribution with parameters α , k , and p . All results presented are the averages of 400 independent executions of the simulator.

Parameters for the distributed web server are chosen in the same manner as for the analytic comparison (see Table 3) except that in these simulations α varies from $\alpha = 1.2$ to $\alpha = 1.9$. Simulating $\alpha < 1.2$ was computationally prohibitive. As before, the lower bound of the Bounded Pareto, k , is a function of α defined to keep $\mathbf{E}\{X\}$ constant.

We simulate four task-assignment policies: RANDOM, in which each incoming task is assigned a host chosen at random; RR, in which the i th incoming task is directed to host number $1 + (i \bmod h)$; DYNAMIC, in which each incoming task is directed to the currently least-loaded host; and SITA-E in which each incoming task is directed to a host according to the task’s size (see Section 3).

5.2 Simulation Results

Figure 8 shows the simulation results for the system as a whole as a function of α (again, note the logarithmic scale on the y axis in these plots). These results show that in simulation, most metrics are smaller than predicted by analysis; we believe this is evidence that these simulations are slow to converge steady state. However, the simulations show the same trends as the analytical results (Figure 5); in particular the performance of SITA-E is relatively insensitive to α , the performance of RANDOM grows worse approximately exponentially with decreasing α , and the difference between the two policies is typically many orders of magnitude.

Not surprisingly, Figure 8 shows that the RANDOM and RR policies have nearly indistinguishable performance behavior on all metrics, and in all cases they are much worse than either SITA-E or DYNAMIC. The difference between SITA-E and RANDOM is sometimes as great as 10^4 . This is evidence that RANDOM and RR are a poor choice for FCFS distributed systems with highly variable task sizes.

Interestingly, the simulation shows that the performance of SITA-E is relatively constant with changing α , whereas the performance of the DYNAMIC algorithm degrades as α becomes smaller. Observe that in all the performance graphs of Figure 8, the SITA-E and DYNAMIC curves cross for α between 1.4 and 1.6, below which SITA-E’s performance dominates that of DYNAMIC. At $\alpha = 1.2$, SITA-E dominates DYNAMIC by a factor of around 50. Thus we expect that in the range of α s relevant to Web file sizes, SITA-E should have superior performance to DYNAMIC.

Table 2 shows the simulation results for each server host for the case $\alpha = 1.2$. The tables show that SITA-E’s performance on slowdown and queue length at *every* host is superior to RANDOM, DYNAMIC, and RR. That is, even host h to which the largest tasks are assigned has better slowdown behavior than under the DYNAMIC policy. However the table also shows that SITA-E’s performance on waiting time falls short for the last hosts.

The results in this section confirm analytic indications that FCFS/SITA-E is remarkably effective and insensitive to α when $\alpha > 1$. SITA-E generally performs better than any other policy studied when used with FCFS queueing at the hosts. In particular, even DYNAMIC shows sensitivity to α in this range; as a result, for small enough α values, FCFS/SITA-E outperforms FCFS/DYNAMIC.

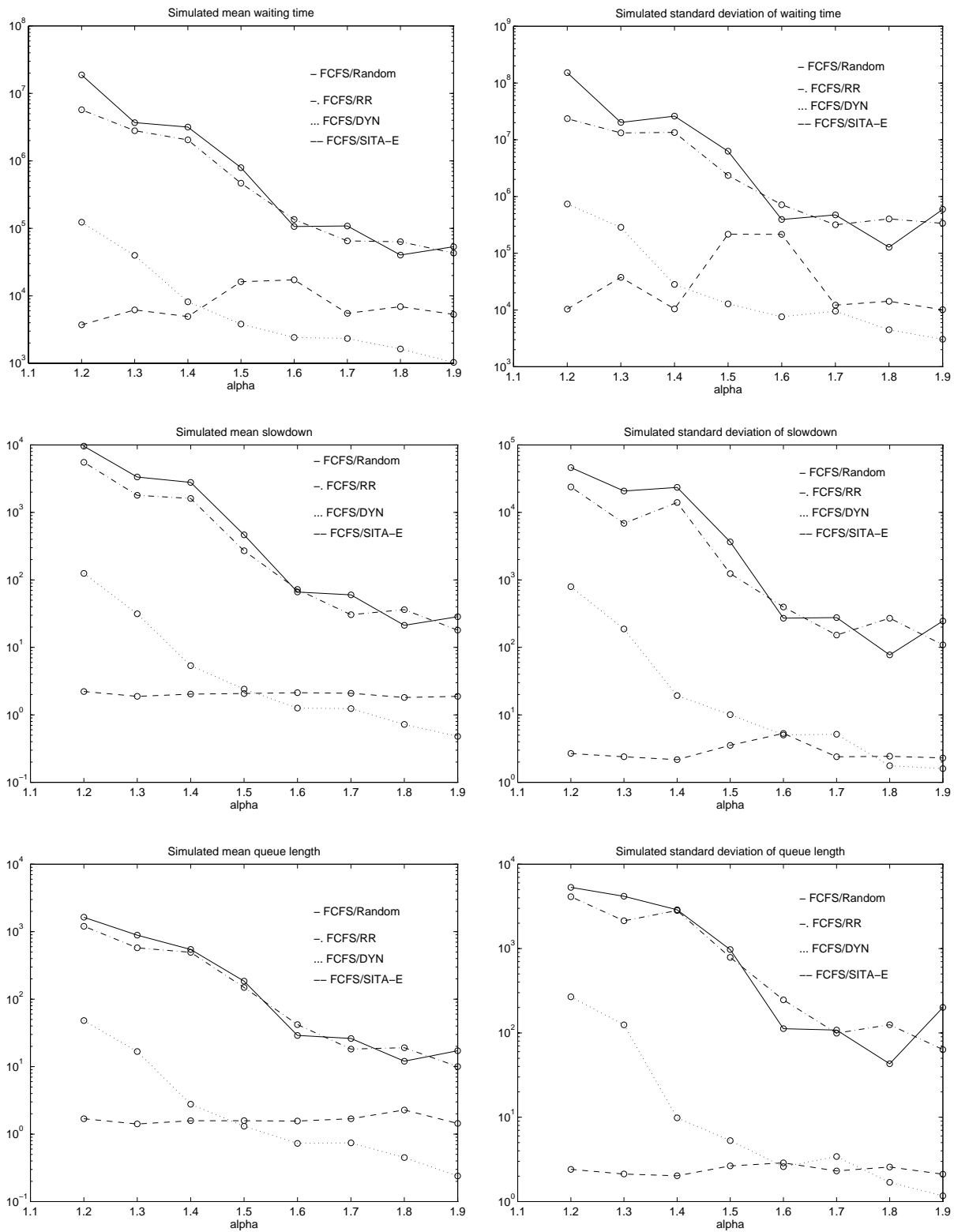


Figure 8: Simulated values of performance metrics for FCFS/RANDOM, FCFS/RR, FCFS/DYNAMIC, and FCFS/SITA-E as a function of α

	RAN	RR	DYN	SITA-E
1	14805	5540	222	2.05
2	4603	13742	150	1.74
3	7476	8771	144	1.75
4	8812	14199	188	1.42
5	23719	14746	151	1.67
6	6997	10904	176	2.65
7	5626	11816	175	5.93
8	15308	5806	118	2.99

	RAN	RR	DYN	SITA-E
1	1423	1205	32	1.64
2	1407	1463	29	1.35
3	1691	1468	31	1.23
4	1552	1301	36	1.08
5	815	1184	31	1.19
6	1426	1375	28	1.68
7	1204	1718	24	2.85
8	1501	1240	27	0.29

	RAN	RR	DYN	SITA-E
1	7.3E6	5.7E6	2.1E5	1.3E3
2	1.3E7	1.4E7	1.2E5	2.3E3
3	1.5E7	8.6E6	1.2E5	4.8E3
4	8.9E6	1.4E7	1.8E5	1.1E4
5	5.4E6	1.4E7	1.6E5	4.0E4
6	1.1E7	1.0E7	1.5E5	2.5E5
7	1.1E7	1.3E7	1.6E5	4.1E6
8	1.2E7	5.8E6	1.1E5	5.1E7

Table 2: Per-host mean slowdown (left), queue length (center) and waiting time (right) for $\alpha = 1.2$

6 Related work

Our work combines two areas of research: load balancing on distributed servers and heavy-tailed distributions.

There is a huge body of literature on load balancing in general distributed systems, including analytic, simulation, and implementation work (see [9] for a number of references). More recently, analytic work on load balancing has considered more general task size distributions than the traditional exponential distribution (this usually requires some decomposition approximation where the nodes of the network are assumed to behave independently of each other, see for example, [6]). However the specific properties of *heavy-tailed* task sizes are only just now beginning to be incorporated into the design of load balancing policies [9]. Performance analysis of load balancing policies under heavy-tailed task sizes is usually difficult. Heavy-tailed distributions have however been considered in some load balancing simulations [16, 14, 9].

Nonetheless, heavy-tailed distributions are important because they are beginning to appear in many measurements of computer systems. Work in [3, 4, 1] has shown that Web file sizes often exhibit heavy tails. The ranges of α reported in [3, 4] are approximately 1.1 to 1.3. There is evidence that file sizes in systems other than the Web may show heavy tails as well: Unix file size measurements are presented in [11], and I/O in a general computing environment is presented in [21]. Also, the authors in [20] found that the upper tail of the distribution of data bytes in FTP bursts (file transfers over the Internet) was well fit to a heavy-tailed distribution with $0.9 \leq \alpha \leq 1.1$.

In addition to files and I/O traces, other computer system attributes have also been shown to exhibit heavy tailed distributions. In particular, the lifetime of processes in some systems can show heavy tails: [16] found that Unix process lifetimes showed heavy tails with $1.05 \leq \alpha \leq 1.25$ and [9] found that Unix process lifetimes showed heavy tails with $\alpha \approx 1$.

Analysis of queues with infinite variance task sizes is difficult [22]. For this reason finite variance approximations to heavy-tailed distributions are increasingly being used [7].

7 Conclusion

The main contribution of this paper is the introduction and performance evaluation of the SITA-E task assignment policy. The SITA-E approach may be applied to any task size distribution so

long as that distribution is known in advance and has finite mean. The key idea of SITA-E is to limit the range of task sizes allocated to each host, while using the distribution of task sizes to ensure that the expected work assigned to each host is equal. The effects of SITA-E are (1) reduced variance at most hosts, and (2) tasks are assigned to queues that contain tasks of a similar size. As a result, SITA-E reduces both mean waiting time and mean slowdown. The SITA-E policy has a straightforward implementation and, unlike dynamic load balancing, does not require any knowledge of conditions at the hosts.

We have analyzed SITA-E and shown that for heavy-tailed task size distributions spanning the range of α values between 1 and 2, SITA-E task assignment with FCFS queueing at the hosts can significantly outperform a load-balanced server using the PS discipline at the hosts. In this range of α , FCFS/SITA-E reduces mean slowdown and mean queue length by a factor of two over PS, for servers with four or more hosts. For large enough systems, FCFS/SITA-E also results in lower mean waiting time as compared to PS. We also show in simulation that while FCFS performs quite poorly under naive task assignment policies such as Round-Robin or Random task assignment, FCFS with SITA-E performs quite well. Finally, we show that in many cases SITA-E is better than Dynamic task assignment, despite the additional information about instantaneous load at hosts used by the dynamic policy.

One additional implication of this work is that it strengthens the argument for admission control in distributed servers. Admission control has often been proposed in timeshared systems as a means to improve overall system throughput, for example, to avoid virtual memory overhead due to thrashing [2, 19]. In admission control, at most m tasks are allowed to timeshare the host where m is near the optimal multiprogramming level for the host. However, the use of admission control means that the system must either deny service to some requests, or queue them for service. We assume that it is undesirable to deny service; therefore admission control leads to a system in which if a host receives more than m concurrent tasks, the first m are serviced in parallel and once one of the m leaves the next successive arrival is allowed in from the queue. Under heavy loads such a system may be expected to experience significant queueing. Normally such queueing would introduce undesirable user-perceived delays, and consequently most Web servers *do not* directly exercise admission control. However, the results in this paper are suggestive that if SITA-E is used, such queueing may not introduce serious delays.

Thus the combination of SITA-E and heavy-tailed task size distributions refines the traditionally accepted wisdom about the benefits of queueing versus timesharing. Under a high variance task size distribution, timesharing is generally used to keep small tasks from waiting behind large tasks. However, we have shown that in the case of a distributed system, SITA-E can often accomplish this *more effectively* than timesharing.

References

- [1] Martin F. Arlitt and Carey L. Williamson. Web server workload characterization: The search for invariants. In *Proceedings of the 1996 SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 126–137, 1996.
- [2] Azer Bestavros and Sue Nagy. Concurrency Admission Control in ACCORD. In Azer Bestavros

- and Victor Fay-Wolfe, editors, *Advances in Real-Time Database and Information Retrieval Systems*, chapter 15. Kluwer Academic Publishers, Norwell, Massachusetts, 1997.
- [3] Mark E. Crovella and Azer Bestavros. Self-similarity in World Wide Web traffic: Evidence and possible causes. In *Proceedings of the 1996 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 160–169, May 1996.
 - [4] Mark E. Crovella, Murad S. Taqqu, and Azer Bestavros. Heavy-tailed probability distributions in the world wide web. In *A Practical Guide To Heavy Tails*, chapter 1, pages 1–23. Chapman & Hall, New York, 1997.
 - [5] Carlos A. Cunha, Azer Bestavros, and Mark E. Crovella. Characteristics of WWW client-based traces. Technical Report TR-95-010, Boston University Department of Computer Science, April 1995.
 - [6] E. Lazowska D. Eager and J. Zahorjan. Adaptive load sharing in homogeneous distributed systems. *IEEE Transactions on Software Engineering*, 12(5), 1986.
 - [7] Anja Feldmann and Ward Whitt. Fitting mixtures of exponentials to long-tail distributions to analyze network performance models. In *Proceedings of IEEE INFOCOM'97*, pages 1098–1116, April 1997.
 - [8] Simson L. Garfinkel. The wizard of Netscape. *WebServer Magazine*, 1(2):59–63, 1996.
 - [9] Mor Harchol-Balter and Allen Downey. Exploiting process lifetime distributions for dynamic load balancing. In *Proceedings of SIGMETRICS '96*, pages 13–24, 1996.
 - [10] Guerny Hunt, Erich Nahum, and John Tracey. Enabling content-based load distribution for scalable services. Preprint, 1997.
 - [11] Gordon Irlam. Unix file size survey - 1993. Available at <http://www.base.com/gordoni-ufs93.html>, September 1994.
 - [12] E. D. Katz, M. Butler, and R. E. McGrath. A scalable web server: The NCSA prototype. In *Proceedings of the First International WWW Conference*, 1994.
 - [13] Leonard Kleinrock. *Queueing Systems*, volume II. Computer Applications. John Wiley & Sons, 1976.
 - [14] Phillip Krueger and Miron Livny. A comparison of preemptive and non-preemptive load distributing. In *8th International Conference on Distributed Computing Systems*, pages 123–130, June 1988.
 - [15] K.L.E. Law, B. Nandy, and A. Chapman. A scalable and distributed WWW proxy system. In *Proceedings of ACM Multimedia '97*, 1997.
 - [16] W. E. Leland and T. J. Ott. Load-balancing heuristics and process behavior. In *Proceedings of Performance and ACM Sigmetrics*, pages 54–69, 1986.
 - [17] Antoine Morad and Huiqun Liu. Redirection-based scalable web server architecture. preprint, 1997.
 - [18] [Author Omitted]. To queue or not to queue: When queueing is better than timesharing in a distributed system (extended version). Technical Report 10/97.

- [19] H. Pang, M. J. Carey, and M. Livny. Managing memory for real-time queries. In *Proceedings of the 1994 ACM SIGMOD Conference on Management of Data*, pages 221–232, 1994.
- [20] Vern Paxson and Sally Floyd. Wide-area traffic: The failure of poisson modeling. *IEEE/ACM Transactions on Networking*, pages 226–244, June 1995.
- [21] David L. Peterson. Data center I/O patterns and power laws. In *CMG Proceedings*, December 1996.
- [22] E. Willekens and J.L. Teugels. Asymptotic expansions for waiting time probabilities in an M/G/1 queue with long-tailed service times. *Queueing Systems*, 10:295–312, 1992.

8 Appendix

8.1 Proof of Theorem 2

Proof:

In a PS queue, all jobs experience the same slowdown, namely $\frac{\rho}{1-\rho}$, [13]. That is:

$$\mathbf{E}\{W|tasksize = x\} = \frac{\rho}{1-\rho} \cdot x$$

(Observe W and X are not independent for a PS queue, although they are for a FCFS queue).

The above equation is used to derive the metrics below.

$$\begin{aligned}
\mathbf{E}\{W\} &= \sum_x \mathbf{P}\{Tasksize = x\} \cdot \mathbf{E}\{W|tasksize = x\} \\
&= \sum_x \frac{\rho}{1-\rho} x \mathbf{P}\{Tasksize = x\} \\
&= \frac{\rho}{1-\rho} \sum_x \mathbf{P}\{Tasksize = x\} x \\
&= \frac{\rho}{1-\rho} \mathbf{E}\{X\} \\
\mathbf{E}\{Q\} &= \lambda \mathbf{E}\{W\} \\
&= \lambda \frac{\rho}{1-\rho} \cdot \mathbf{E}\{X\} \\
&= \frac{\rho^2}{1-\rho} \\
\mathbf{E}\{S\} &= \sum_x \mathbf{P}\{Tasksize = x\} \cdot \mathbf{E}\{S|tasksize = x\} \\
&= \sum_x \mathbf{P}\{Tasksize = x\} \cdot \mathbf{E}\{W|tasksize = x\} \cdot \frac{1}{x} \\
&= \sum_x \frac{\rho}{1-\rho} \mathbf{P}\{Tasksize = x\} \cdot x \cdot \frac{1}{x} \\
&= \frac{\rho}{1-\rho}
\end{aligned}$$

■

8.2 Proof of Theorem 3

Proof: We demonstrate that when

$$x_i = \left(\frac{(h-i)}{h} k^{1-\alpha} + \frac{i}{h} p^{1-\alpha} \right)^{\frac{1}{1-\alpha}}, \text{ for } i = 0 \dots h,$$

then the work allocated to the i th host is exactly $\mathbf{E}\{X\}/h$, for all i , assuming $\alpha \neq 1$.

$$\begin{aligned} \int_{x_{i-1}}^{x_i} x \cdot f(x) dx &= \int_{x_{i-1}}^{x_i} \frac{\alpha k^\alpha}{1 - (k/p)^\alpha} x^{-\alpha} dx \\ &= \frac{\alpha k^\alpha}{1 - (k/p)^\alpha} \int_{x_{i-1}}^{x_i} x^{-\alpha} dx \\ &= \frac{\alpha k^\alpha}{(1 - (k/p)^\alpha)(1 - \alpha)} \cdot (x_i^{1-\alpha} - x_{i-1}^{1-\alpha}) \\ &= \frac{\alpha k^\alpha}{(1 - (k/p)^\alpha)(1 - \alpha)} \cdot \frac{(-k^{1-\alpha} + p^{1-\alpha})}{h} \\ &= \frac{\mathbf{E}\{X\}}{h} \end{aligned}$$

If $\alpha = 1$, then defining

$$x_i = k(p/k)^{(i/h)}, \text{ for } i = 1 \dots h,$$

we have:

$$\begin{aligned} \int_{x_{i-1}}^{x_i} x \cdot f(x) dx &= \int_{x_{i-1}}^{x_i} \frac{\alpha k^\alpha}{1 - (k/p)^\alpha} x^{-\alpha} dx \\ &= \frac{\alpha k^\alpha}{1 - (k/p)^\alpha} \int_{x_{i-1}}^{x_i} x^{-\alpha} dx \\ &= \frac{\alpha k^\alpha}{(1 - (k/p)^\alpha)(1 - \alpha)} \cdot (\ln(x_i) - \ln(x_{i-1})) \\ &= \frac{\alpha k^\alpha}{(1 - (k/p)^\alpha)(1 - \alpha)} \cdot \left(\frac{1}{h} \cdot (\ln(p) - \ln(k)) \right) \\ &= \frac{\mathbf{E}\{X\}}{h} \end{aligned}$$

■

8.3 Proof of Theorem 4

Proof:

To see that X_i has a $B(x_{i-1}, x_i, \alpha)$ distribution, observe that X_i is simply the random variable X , where only values in the range (x_{i-1}, x_i) are permitted. Since X has a Bounded Pareto distribution, X_i does as well.

To prove that $\rho_i = \rho$ for all i , first observe the following hand-waving argument: In SITA-E the “work” is being split up evenly between the h hosts. Since all the hosts process work at the same rate ($\mu = 1$), this says intuitively that each host is getting the same “load,” so all hosts have load ρ . More formally,

$$\begin{aligned}
 \rho_i &= \lambda_i \mathbf{E}\{X_i\} \\
 &= \lambda_i \int_{x_{i-1}}^{x_i} x \frac{f(x)}{p_i} dx \\
 &= \lambda_i \cdot \frac{1}{p_i} \int_{x_{i-1}}^{x_i} x f(x) dx \\
 &= \lambda \frac{\mathbf{E}\{X\}}{h} \\
 &= \rho
 \end{aligned}$$

where the second equality from the end follows from the definition of SITA-E.

■

8.4 Proof of Theorem 5

Proof: The h queues are independent and i th queue may be viewed as an M/G/1 queue with arrival rate λp_i and service distribution $B(x_{i-1}, x_i, \alpha)$. Since the queues are not identical, evaluating the system performance metrics usually requires a weighted sum over the queues. The important observation for SITA-E is that although W and X are *not* independent for a task entering the system in SITA-E (unlike in the RANDOM task assignment policy), within a single host W and X are *independent*. Thus in the analysis of slowdown performance metrics for the i th host, we can assume W_i and X_i to be independent.

Each formula follows from those above it: The equation for mean and variance of waiting time at the i th queue are given by the Pollaczek-Kinchin formulae. The mean waiting time for the system is the weighted average of the waiting time at the individual hosts. To derive the variance in the system waiting time, we first derive the second moment of the system waiting time by again using a weighted average over the individual hosts. The mean queue size at a host again follows from Little’s formula. Since Q is defined as the queue size seen by an arrival into the system, the expected value of Q is the weighted average over the queue size at each queue (note, Little’s law does not apply to our definition of Q). The slowdown derivations are first done for an individual

host, where we know that waiting time, W_i , and service time, X_i , are independent. Then the mean and second moment of the system slowdown are derived by taking a weighted average over those values at the individual hosts.

■