In *Proceedings of Performance Tools '98*
Lecture Notes in Computer Science, Vol. 1468, pp. 231-242, September 1998

On Choosing a Task Assignment Policy for a Distributed Server System

Mor Harchol-Balter⋆,1, Mark E. Crovella⋆⋆,2, and Cristina D. Murta⋆⋆⋆,2

[1] Laboratory for Computer Science, MIT
harchol@theory.lcs.mit.edu
[2] Department of Computer Science, Boston University
{crovella,murta}@bu.edu

Abstract. We consider a distributed server system model and ask which policy should be used for assigning tasks to hosts. In our model each host processes tasks in First-Come-First-Serve order and the task's service demand is known in advance. We consider four task assignment policies commonly proposed for such distributed server systems: Round-Robin, Random, Size-Based, in which all tasks within a give size range are assigned to a particular host, and Dynamic-Least-Work-Remaining, in which a task is assigned to the host with the least outstanding work. Our goal is to understand the influence of task size variability on the decision of which task assignment policy is best. We find that no *one* of the above task assignment policies is best and that the answer depends critically on the variability in the task size distribution. In particular we find that when the task sizes are not highly variable, the Dynamic policy is preferable. However when task sizes show the degree of variability more characteristic of empirically measured computer workloads, the Size-Based policy is the best choice. We use the resulting observations to argue in favor of a specific size-based policy, SITA-E, that can outperform the Dynamic policy by almost 2 orders of magnitude and can outperform other task assignment policies by many orders of magnitude, under a realistic task size distribution.

1 Introduction

To build high-capacity server systems, developers are increasingly turning to distributed designs because of their scalability and cost-effectiveness. Examples of this trend include distributed Web servers, distributed database servers, and high performance computing clusters. In such a system, requests for service arrive and must be assigned to one of the host machines for processing. The rule for assigning tasks to host machines is known as the *task assignment policy.*

⋆ 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.

In this paper we concentrate on the particular model of a distributed server system in which each incoming task is immediately assigned to a host machine, and each host machine processes its assigned tasks in first-come-first-served (FCFS) order. We also assume that the task's service demand is known in advance. Our motivation for considering this model is that it is an abstraction of some existing distributed servers, described in Section 3.

We consider four task assignment policies commonly proposed for such distributed server systems: Round-Robin, in which tasks are assigned to hosts in a cyclical fashion; Random, in which each task is assigned to each host with equal probability; Size-Based, in which all tasks within a certain size range are sent to an individual host; and Dynamic (also known as Least-Work-Remaining) in which an incoming task is assigned to the host with the least amount of outstanding work left to do (based on the sum of the sizes of those tasks in the queue).

Our goal is to study the influence of task size variability on the decision of which task assignment policy is best. We are motivated in this respect by the increasing evidence for high variability in task size distributions, witnessed in many measurements of computer workloads. In particular, measurements of many computer workloads have been shown to fit a heavy-tailed distributions with very high variance, as described in Section 2.2.

In comparing task assignment policies, we make use of simulations and also analysis or analytic approximations. We show that the variability of the task size distribution makes a crucial difference in choosing a task assignment policy, and we use the resulting observations to argue for a specific task assignment policy that works well under conditions of high task size variance.

2 Background and Previous Work

2.1 Fundamental Results in Task Assignment

The problem of task assignment in a model like ours has been extensively studied, but many basic questions remain open. In the case where task sizes are unknown, the following results exist: Under an exponential task size distribution, the optimality of Shortest-Line task assignment policy (send the task to the host with the shortest queue) was proven by Winston [14] and extended by Weber [12] to include task size distributions with nondecreasing failure rate. The actual performance of the Shortest-Line policy is not known exactly, but is approximated by Nelson and Phillips [9]. In fact as the variability of the task size distribution grows, the Shortest-Line policy is no longer optimal, Whitt [13].

In the case where the individual task sizes are known, as in our model, equivalent optimality and performance results have not been developed for the task assignment problem, to the best of our knowledge. For the scenario in which the ages of the tasks currently serving are known, Weber [12] has shown that the Shortest-Expected-Delay rule is optimal for task size distributions with increasing failure rate, and Whitt [13] has shown that there exist task size distributions for which the Shortest-Expected-Delay rule is not optimal.

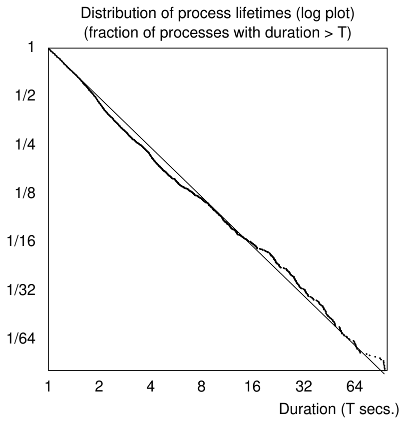


Fig. 1. Measured distribution of UNIX process CPU lifetimes, from [5]. Data indicates fraction of jobs whose CPU sevice demands exceed T seconds, as a function of T.

2.2 Measurements of task size distributions in computer applications

Many application environments show a mixture of task sizes spanning many orders of magnitude. In such environments there are typically many small tasks, and fewer large tasks. Much previous work has used the exponential distribution to capture this variability, as described in Section 2.1. However, recent measurements indicate that for many applications the exponential distribution is a poor model and that a heavy-tailed distribution is more accurate. In general a heavy-tailed distribution is one for which $\Pr\{X > x\} \sim x^{-\alpha}$, where $0 < \alpha < 2$.

Task sizes following a heavy-tailed distribution show the following properties:

1. Decreasing failure rate: In particular, the longer a task has run, the longer it is expected to continue running.
2. Infinite variance (and if $\alpha \leq 1$, infinite mean).
3. The property that a very small fraction ($< 1\%$) of the very largest tasks make up a large fraction (half) of the load. We will refer to this important property throughout the paper as the *heavy-tailed property*.

The lower the parameter α, the more variable the distribution, and the more pronounced is the heavy-tailed property, *i.e.* the smaller the faction of large tasks that comprise half the load.

As a concrete example, Figure 1 depicts graphically on a log-log plot the measured distribution of CPU requirements of over a million UNIX processes, taken from paper [5]. This distribution closely fits the curve

$$\Pr\{\text{Process Lifetime } > T\} = 1/T.$$

In [5] it is shown that this distribution is present in a variety of computing environments, including instructional, reasearch, and administrative environments.

In fact, heavy-tailed distributions appear to fit many recent measurements of computing systems. These include, for example:

- Unix process CPU requirements measured at Bellcore: $1 \leq \alpha \leq 1.25$ [8].
- Unix process CPU requirements, measured at UC Berkeley: $\alpha \approx 1$ [5].
- Sizes of files transferred through the Web: $1.1 \leq \alpha \leq 1.3$ [1, 3].
- Sizes of files stored in Unix filesystems: [7].
- I/O times: [11].
- Sizes of FTP transfers in the Internet: $.9 \leq \alpha \leq 1.1$ [10].

In most of these cases where estimates of α were made, $1 \leq \alpha \leq 2$. In fact, typically α tends to be close to 1, which represents very high variability in task service requirements.

3 Model and Problem Formulation

We are concerned with the following model of a distributed server. The server is composed of h hosts, each with equal processing power. Tasks arrive to the system according to a Poisson process with rate λ. When a task arrives to the system, it is inspected by a dispatcher facility which assigns it to one of the hosts for service. We assume the dispatcher facility knows the size of the task. The tasks assigned to each host are served in FCFS order, and tasks are not preemptible. We assume that processing power is the only resource used by tasks.

The above model for a distributed server was initially inspired by the `xolas` batch distributed computing facility at MIT's Laboratory for Computer Science. `Xolas` consists of 4 identical multiprocessor hosts. Users specify an upper bound on their job's processing demand. If the job exceeds that demand, it is killed. The `xolas` facility has a dispatcher front end which assigns each job to one of the hosts for service. The user is given an upper bound on the time their job will have to wait in the queue, based on the sum of the sizes of the jobs in that queue. The jobs queued at each host are each run to completion in FCFS order.

We assume that task sizes show some maximum (but large) value. 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:

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

Throughout this paper we model task sizes using a $B(k, p, \alpha)$ distribution, and vary α over the range 0 to 2 in order to observe the effect of changing variability of the distribution. To focus on the effect of changing variance, we keep the distributional mean fixed (at 3000) and the maximum value fixed (at

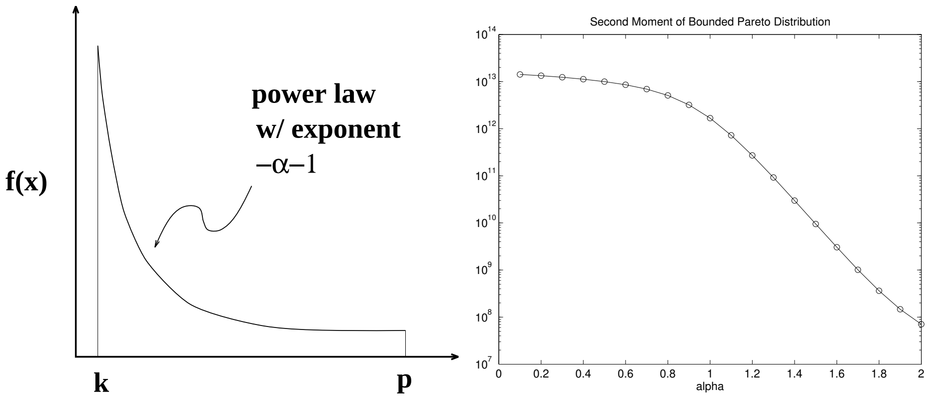


Fig. 2. Parameters of the Bounded Pareto Distribution (left); Second Moment of $B(k, 10^{10}, \alpha)$ as a function of α, when $\mathbf{E}\{X\} = 3000$ (right).

Number of hosts	$h = 8$.
System load	$\rho = .8$.
Mean service time	$\mathbf{E}\{X\} = 3000$ time units
Task arrival process	Poisson with rate $\lambda = \rho \cdot 1/\mathbf{E}\{X\} \cdot h = .0021$ tasks/unit time
Maximum task service time	$p = 10^{10}$ time units
α parameter	$0 < \alpha \le 2$
Minimum task service time	chosen so that mean task service time stays constant as α varies ($0 < k \le 1500$)

Table 1. Parameters used in evaluating task assignment policies

$p = 10^{10}$). In order to keep the mean constant, we adjust k slightly as α changes ($0 < k \le 1500$). The above parameters are summarized in Table 1.

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 very high variability if $k \ll p$. For example, Figure 2 (right) shows the second moment $\mathbf{E}\left\{X^2\right\}$ of this distribution as a function of α for $p = 10^{10}$, where k is chosen to keep $\mathbf{E}\{X\}$ constant at 3000, ($0 < k \le 1500$). The figure shows that the second moment explodes exponentially as α declines. Furthermore, the Bounded Pareto distribution also still exhibits the heavy-tailed property and (to some extent) the decreasing failure rate property of the unbounded Pareto distribution.

Given the above model of a distributed server system, we ask how to select the best task assignment policy. The following four are common choices:

Random : an incoming task is sent to host i with probability $1/h$. This policy equalizes the expected number of tasks at each host.

Round-Robin : tasks are assigned to hosts in cyclical fashion with the ith task being assigned to host i mod h. This policy also equalizes the expected number of tasks at each host, and typically has less variability in interarrival times than Random.

Size-Based : Each host serves tasks whose service demand falls in a designated range. This policy attempts to keep small tasks from getting "stuck" behind large tasks.

Dynamic : Each incoming task is assigned to the host with the smallest amount of outstanding work, which is the sum of the sizes of the tasks in the host's queue plus the work remaining on that task currently being served. This policy is optimal from the standpoint of an individual task, and from a system standpoint attempts to achieve instantaneous load balance.

In this paper we compare these policies as a function of the variability of task sizes. The effectiveness of these task assignment schemes will be measured in terms of mean waiting time and mean slowdown, where a task's slowdown is its waiting time divided by its service demand. All means are per-task averages.

3.1 A New Size-Based Task Assignment Policy: SITA-E

Before delving into simulation and analytic results, we need to specify a few more parameters of the size-based policy.

In size-based task assignment, a size range is associated with each host and a task is sent to the appropriate host based on its size. In practice the size ranges associated with the hosts are often chosen somewhat arbitrarily. There might be a 15-minute queue for tasks of size between 0 and 15 minutes, a 3-hour queue for tasks of size between 15 minutes and 3 hours, a 6-hour queue, a 12-hour queue and an 18-hour queue, for example. (This example is used in practice at the Cornell Theory Center IBM SP2 job scheduler [6].)

In this paper we choose a more formal algorithm for size-based task assignment, which we refer to as SITA-E — Size Interval Task Assignment with Equal Load. The idea is simple: define the size range associated with each host such that the total work (load) directed to each host is the same. The motivation for doing this is that balancing the load minimizes mean waiting time.

The mechanism for achieving balanced expected load at the hosts is to use the *task size distribution* to define the cutoff points (defining the ranges) so that the expected work directed to each host is the same. The task size distribution is easy to obtain by maintaining a histogram (in the dispatcher unit) of all task sizes witnessed over a period of time.

More precisely, let $F(x) = \Pr\{X \leq x\}$ denote the cumulative distribution function of task sizes with finite mean M. Let k denote the smallest task size, p (possibly equal to infinity) denote the largest task size, and h be the number of hosts. Then we determine "cutoff points" x_i, $i = 0 \ldots h$ where $k = x_0 < x_1 < x_2 < \ldots < x_{h-1} < x_h = p$, such that

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

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

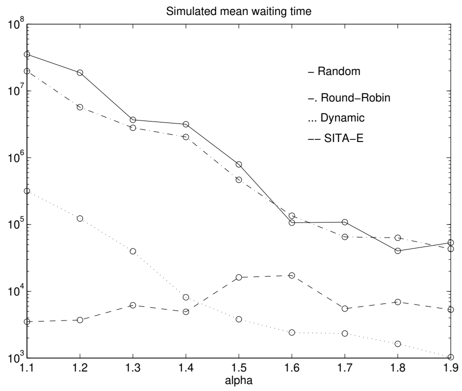
SITA-E as defined can be applied to *any* task size distribution with finite mean. In the remainder of the paper we will always assume the task size distribution is the Bounded Pareto distribution, $B(k, p, \alpha)$.

4 Simulation Results

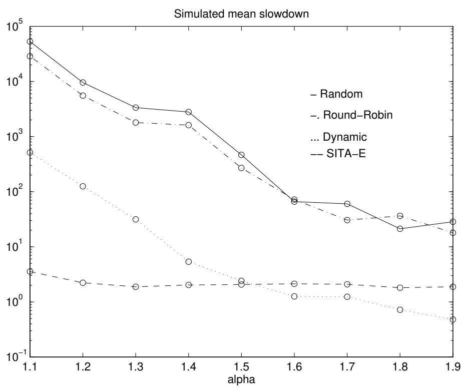
In this section we compare the Random, Round-Robin, SITA-E, and Dynamic policies via simulation. Simulation parameters are as shown in Table 1.

Simulating a server system with heavy-tailed, highly variable service times is difficult because the system approaches steady state very slowly and usually from below [2]. This occurs because the running average of task sizes is typically at the outset well below the true mean; the true mean isn't achieved until enough large tasks arrive. The consequence for a system like our own is that simulation outputs appear more optimistic than they would in steady-state. To make our simulation measurements less sensitive to the startup transient, we run our simulation for 4×10^5 arrivals and then capture data from the next single arrival to the system only. Each data point shown in our plots is the average of 400 independent runs, each of which started from an empty system.

We consider α values in the range 1.1 (high variability) to 1.9 (lower variability). As described in Section 2.2, α values in the range 1.0 to 1.3 tend to be common in empirical measurements of computing systems.



(a)



(b)

Fig. 3. Mean Waiting Time (a) and Mean Slowdown (b) under Simulation of Four Task Assignment Policies as a Function of α.

Figure 3 shows the performance of the system for all four policies, as a function of α (note the logarithmic scale on the y axis). Figure 3(a) shows mean waiting time and 3(b) shows mean slowdown. Below we simply summarize these results; in the next section, we will use analysis to explain these results.

First of all, observe that the performance of the system under the Random and Round Robin policies is similar, and that both cases perform much more poorly than the other two (SITA-E and Dynamic). As α declines, both of the performance metrics under the Random and Round-Robin policies explode approximately exponentially. This gives an indication of the severe impacts that heavy-tailed workloads can have in systems with naive task assignment policies.

The Dynamic policy shows the benefits of instantaneous load balancing. Dynamic is on the order of 100 times better for both metrics when compared to Random and Round Robin. For large α, this means that Dynamic performs quite well—with mean slowdown less than 1. However as the variability in task size increases (as $\alpha \to 1$), Dynamic is unable to maintain good performance. It too suffers from roughly exponential explosion in performance metrics as α declines.

In contrast, the behavior of SITA-E is quite different from that of the other three. Over the entire range of α values studied, the performance of the system under SITA-E is relatively unchanged, with mean slowdown always between 2 and 3. This is the most striking aspect of our data: in a range of α in which performance metrics for Random, Round Robin, and Dynamic all explode, SITA-E's performance remains remarkably insensitive to increase in task size variability.

As a result we find that when task size is less variable, Dynamic task assignment exhibits better performance; but when task sizes show the variability that is more characteristic of empirical measurements ($\alpha \approx 1.1$), SITA-E's performance can be on the order of 100 times better than that of Dynamic.

In [4] we simulate a range of loads (ρ) and show that as load increases, SITA-E becomes preferable to Dynamic over a larger range of α.

The remarkable consistency of system performance under the SITA-E policy across the range of α from 1.1 to 1.9 is difficult to understand using the tools of simulation alone. For that reason the next section develops analysis of SITA-E and the other policies, and uses that analysis to explain SITA-E's performance.

5 Analysis of Task Assignment Policies

To understand the differences between the performance of the four task assignment policies, we provide a full analysis of the Round-Robin, Random, and SITA-E policies, and an approximation of the Dynamic policy.

In the analysis below we will repeatedly make use of the Pollaczek-Kinchin formula below which analyzes the M/G/1 FCFS queue:

$$\mathbf{E}\{\text{Waiting Time}\} = \lambda \mathbf{E}\left\{X^2\right\}/2(1-\rho) \qquad \text{[Pollaczek-Kinchin formula]}$$
$$\mathbf{E}\{\text{Slowdown}\} = \mathbf{E}\{W/X\} = \mathbf{E}\{W\} \cdot \mathbf{E}\left\{X^{-1}\right\}$$

where λ denotes the rate of the arrival process, X denotes the service time distribution, and ρ denotes the utilization ($\rho = \lambda \mathbf{E}\{X\}$). The slowdown formulas follow from the fact that W and X are independent for a FCFS queue.

Observe that every metric for the simple FCFS queue is dependent on $\mathbf{E}\left\{X^2\right\}$, the second moment of the service time. Recall that if the workload is heavy-tailed, the second moment of the service time explodes, as shown in Figure 2.

Random Task Assignment. The Random policy simply performs Bernoulli splitting on the input stream, with the result that each host becomes an independent $M/B(k,p,\alpha)/1$ queue. The load at the ith host, is equal to the system load, that is, $\rho_i = \rho$. So the Pollaczek-Kinchin formula applies directly, and all performance metrics are proportional to the second moment of $B(k,p,\alpha)$. Performance is generally poor because the second moment of the $B(k,p,\alpha)$ is high.

Round Robin. The Round Robin policy splits the incoming stream so each host sees an $E_h/B(k,p,\alpha)/1$ queue, with utilization $\rho_i = \rho$. This system has performance close to the Random case since it still sees high variability in service times, which dominates performance.

SITA-E. The SITA-E policy also performs Bernoulli splitting on the arrival stream (which follows from our assumption that task sizes are independent). By the definition of SITA-E, $\rho_i = \rho$. However the task sizes at each queue are determined by the particular values of the interval cutoffs, $\{x_i\}, i = 0, ..., h$. In fact, host i sees a $M/B(x_{i-1}, x_i, \alpha)/1$ queue. The reason for this is that 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. In [4] we show how to calculate the set of x_is for the $B(k,p,\alpha)$ distribution, and we present the resulting formulas that provide full analysis of the system under the SITA-E policy for all the performance metrics.

Dynamic. The Dynamic policy is not analytically tractable, which is why we performed the simulation study. However, in [4] we prove that a distributed system of the type in this paper with h hosts which performs Dynamic task assignment is actually equivalent to an M/G/h queue. Fortunately, there exist known approximations for the performance metrics of the M/G/h queue [15]:

$$\mathbf{E}\left\{Q_{M/G/h}\right\} = \mathbf{E}\left\{Q_{M/M/h}\right\} \cdot \mathbf{E}\left\{X^2\right\} / \mathbf{E}\left\{X\right\}^2,$$

where X denotes the service time distribution and Q denotes the number in queue. What's important to observe here is that the mean queue length, and therefore the mean waiting time and mean slowdown, are all proportional to the second moment of the service time distribution, as was the case for the Random and Round-Robin task assignment policies.

Using the above analysis we can compute the performance of the above task assignment policies over a range of α values. Figure 4 shows the analytically-derived mean waiting time and mean slowdown of the system under each policy over the whole range of α. Figure 5 again shows these analytically-derived metrics, but only over the range of $1 \leq \alpha \leq 2$, which is the range of α corresponding to most empirical measurements of process lifetimes and file sizes (see Section 2.2). (Note that, because of slow simulation convergence as described at the beginning of Section 4, simulation values are generally lower than analytic predictions; however all simulation trends agree with analysis).

First observe that the performance of the Random and Dynamic policies in both these figures grows worse as α decreases, where the performance curves follow the same shape as the second moment of the Bounded Pareto distribution, shown in Figure 2. This is expected since the performance of Random and Dynamic is directly proportional to the second moment of the service time distribution. By contast, looking at Figure 5 we see that in the range $1 < \alpha < 2$, the mean waiting time and especially mean slowdown under the SITA-E policy is remarkably constant, with mean slowdowns around 3, whereas Random and Dynamic explode in this range. The insensitivity of SITA-E's performance to α in this range is the most striking property of our simulations and analysis.

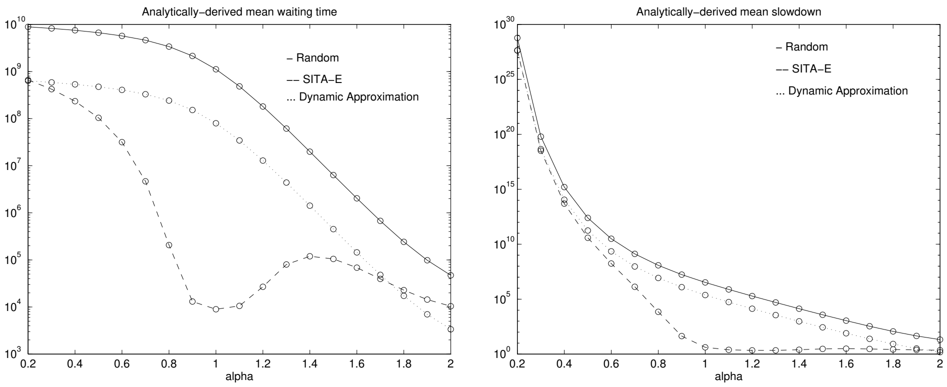


Fig. 4. Analysis of mean waiting time and mean slowdown over whole range of α, $0 < \alpha \leq 2$.

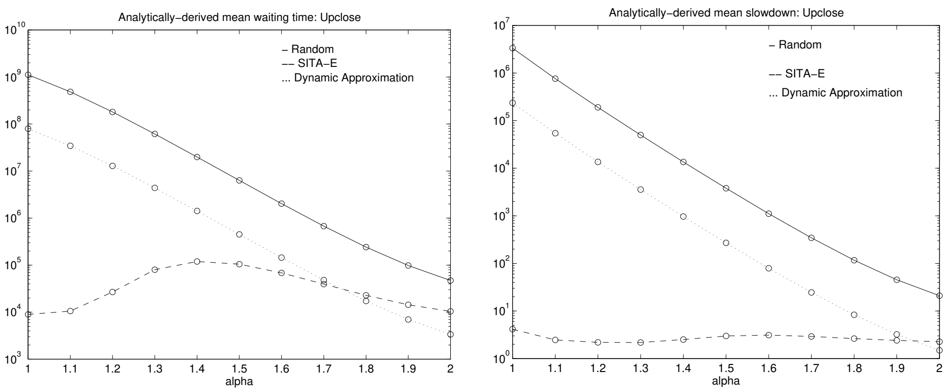


Fig. 5. Analysis of mean waiting time and mean slowdown over empirically relevant range of α, $1 \leq \alpha \leq 2$.

Why does SITA-E perform so well in a region of task size variability wherein a Dynamic policy explodes? A careful analysis of the performance of SITA-E at each queue of the system (see [4]) leads us to the following answers:

1. By limiting the range of task sizes at each host, SITA-E greatly reduces the variance of the task size distribution witnessed by the lowered-numbered hosts, thereby improving performance at these hosts. In fact the performance at most hosts is superior to that of an M/M/1 queue with utilization ρ.
2. When load is balanced, the majority of tasks are assigned to the low-numbered hosts, which are the hosts with the best performance. This is intensified by the heavy-tailed property which implies that *very* few tasks are assigned to high numbered hosts.
3. Furthermore, mean slowdown is improved because small tasks observe proportionately lower waiting times.

For the case of $\alpha \leq 1$, shown in Figure 4, even under the SITA-E policy, system performance eventually deteriorates badly. The reason is that as overall variability in task sizes increases, eventually even host 1 will witness high variability. Further analysis [4] indicates that adding hosts can extend the range over which SITA-E shows good performance. For example, when the number of hosts is 32, SITA-E's performance does not deteriorate until $\alpha \leq .8$.

6 Conclusion

In this paper we have studied how the variability of the task size distribution influences which task assignment policy is best in a distributed system. We consider four policies: Random, Round-Robin, SITA-E (a size-based policy), and Dynamic (sending the task to the host with the least remaining work).

We find that the best choice of task assignment policy depends critically on the variability of task size distribution. When the task sizes are not highly variable, the Dynamic policy is preferable. However, when task sizes show the degree of variability more characteristic of empirical measurements ($\alpha \approx 1$), SITA-E is best.

The magnitude of the difference in performance of these policies can be quite large: Random and Round-Robin are inferior to both SITA-E and Dynamic by several orders of magnitude. And in the range of task size variability characteristic of empirical measurements, SITA-E outperforms Dynamic by close to 2 orders of magnitude.

More important than the above results, though, is the insights about these four policies gleaned from our analysis:

Our analysis of the Random, Round-Robin and Dynamic policies shows that their performance is directly proportional to the second moment of the task size distribution, which explains why their performance deteriorates as the task size variability increases. Thus, even the Dynamic policy, which comes closes to achieving instantaneous load balance and directs each task to the host where it waits the least, is not capable of compensating for the effect of increasing variance in the task size distribution.

To understand why size-based policies are so powerful, we introduce the SITA-E policy which is a simple formalization of size-based policies, defined to equalize the expected load at each host. This formalization allows us to obtain a full analysis of the SITA-E policy, leading to a 3-fold characterization of its power: (i) By limiting the range of task sizes at each host, SITA-E greatly reduces the variability of the task size distribution witnessed by each host – thereby improving the performance at the host. (ii) When load is balanced, most tasks are sent to the subset the hosts having the best performance. (iii) Mean slowdown is improved because small tasks observe proportionately lower waiting times. These 3 properties allow SITA-E to perform very well in a region of task size variability in which the Dynamic policy breaks down.

References

1. M. E. Crovella and A. Bestavros. Self-similarity in World Wide Web traffic: Evidence and possible causes. *IEEE/ACM Transactions on Networking*, 5(6):835–846, December 1997.
2. M. E. Crovella and L. Lipsky. Long-lasting transient conditions in simulations with heavy-tailed workloads. In *1997 Winter Simulation Conference*, 1997.
3. M. E. Crovella, M. S. Taqqu, and A. Bestavros. Heavy-tailed probability distributions in the world wide web. In *A Practical Guide To Heavy Tails*, pages 1–23. Chapman & Hall, New York, 1998.
4. M. Harchol-Balter, M. E. Crovella, and C. D. Murta. On choosing a task assignment policy for a distributed server system. Technical Report MIT-LCS-TR-757, MIT Laboratory for Computer Science, 1998.
5. M. Harchol-Balter and A. Downey. Exploiting process lifetime distributions for dynamic load balancing. *ACM Transactions on Computer Systems*, 15(3), 1997.
6. S. Hotovy, D. Schneider, and T. O'Donnell. Analysis of the early workload on the Cornell Theory Center IBM SP2. Technical Report 96TR234, CTC, Jan. 1996.
7. G. Irlam. Unix file size survey. `http://www.base.com/gordoni/ufs93.html`, 1994.
8. W. E. Leland and T. J. Ott. Load-balancing heuristics and process behavior. In *Proceedings of Performance and ACM Sigmetrics*, pages 54–69, 1986.
9. R. D. Nelson and T. K. Philips. An approximation for the mean response time for shortest queue routing with general interarrival and service times. *Performance Evaluation*, 17:123–139, 1998.
10. V. Paxson and S. Floyd. Wide-area traffic: The failure of Poisson modeling. *IEEE/ACM Transactions on Networking*, pages 226–244, June 1995.
11. D. L. Peterson and D. B. Adams. Fractal patterns in DASD I/O traffic. In *CMG Proceedings*, December 1996.
12. R. W. Weber. On the optimal assignment of customers to parallel servers. *Journal of Applied Probability*, 15:406–413, 1978.
13. Ward Whitt. Deciding which queue to join: Some counterexamples. *Operations Research*, 34(1):226–244, January 1986.
14. W. Winston. Optimality of the shortest line discipline. *Journal of Applied Probability*, 14:181–189, 1977.
15. R. W. Wolff. *Stochastic Modeling and the Theory of Queues*. Prentice Hall, 1989.