# On Resource Pooling in SITA-like Parallel Server Systems

Yinghui Wang
Department of Computing & Software
McMaster University
Email: wang382@mcmaster.ca

Douglas Down
Department of Computing & Software
McMaster University
Email: downd@mcmaster.ca

***Abstract*—The routing policy Size Interval Task Assignment (SITA) isolates small arrivals from large arrivals, while choosing intervals to balance the workload of each server. It works well for highly variable arrivals, but the isolation can cause server idleness. To improve this, we suggest a scheme to add pooling to these SITA-like systems, which can result in better performance. We propose a routing policy, SITA-JSQ, which chooses a proportion of the dedicated arrivals, originally allocated by the SITA policy with equal loads, as flexible arrivals allocated by a JSQ policy between adjacent servers. Under heavy traffic and Complete Resource Pooling conditions, the asymptotic Brownian Motion limit for the unfinished processing times processes is obtained. Using these limits, we show that SITA-JSQ gives asymptotically better performance with respect to unfinished processing times than SITA. Through simulation, we also demonstrate significant reductions in mean waiting times. Finally, we compare our approach to cycle stealing from idle servers.**

## I. Introduction

In distributed service system design, the job assignment policy (the policy to route arrivals when they arrive) is one of the most significant performance factors. For many of today's distributed web service systems and content delivery systems, arrivals have highly variable processing times, for example when http requests and large file requests are served together. As Crovella et al. [1] have shown, a large number of processing time distributions in the Web exhibit such high variability. The traditional Join the Shortest Queue (JSQ) job assignment policy performs poorly for this kind of arrivals [2], [3] because it may lead to arrivals with small processing times waiting behind arrivals with very large processing times. The simulation results provided in [3] show that the Size Interval Task Assignment policy (SITA), which overcomes the variability of job sizes simply through routing the arrivals based on their sizes, will provide better performance. Harchol-Balter et al. [2] carry out extensive experiments on routing policies including JSQ, Random, Size Interval Task Assignment (SITA) and Dynamic-Least-Work-Remaining, and show that the SITA policy is the best choice among them for heterogeneous arrivals. SITA first defines $N-1$ cutoff points $(C_1,\ C_2,\ \ldots, C_{N-1})$ of the job size distribution for the system with $N$ servers and routes arrivals according to Figure 1.

While the static isolation of the SITA policy provides better performance than non-isolating ones like JSQ, it also causes server idleness. To improve this, dynamic pooling could be added. In this paper we propose a new routing policy SITA-JSQ which improves the utilization as well as mean waiting time.

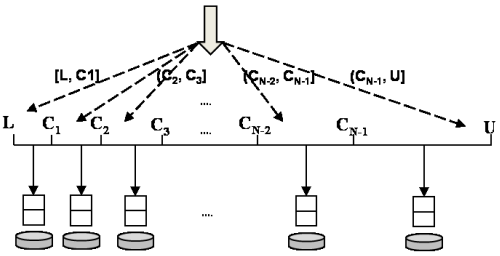



Fig. 1: SITA assignment Policy

### A. Related Work and Motivations

Basically, there are two trends in choosing the cutoff points for SITA policies:

1. SITA-E (Size Interval Task Assignment with Equal loads), where cutoff points are chosen according to (1). This is proposed by Harchol-Balter et al. [2].

$$\int_L^{C_1} xf(x)dx = \cdots = \int_{C_{N-1}}^{U} xf(x)dx, \qquad (1)$$

where $f(x)$ is the density function of the processing times, and $L$ and $U$ are the minimum and maximum processing times, respectively (see Figure 1).

Analysis of SITA-E systems can be found in both [2] and [4]. Bachmat and Sarfat [5] give asymptotic formulas for the mean waiting time and the slowdown of SITA systems with a Bounded Pareto processing time distribution, where the cutoff points are determined asymptotically to be optimal. Vesilo [6] also gives a similar asymptotic analysis for both low load and high load situations based on the Pollaczek-Khinchine formula for SITA systems with Bounded Pareto processing time distribution.

2. SITA-V (Size Interval Task assignment with Variable Load). Although the SITA-E policy has received lots of attention, Crovella et al. [7] and Schroeder and Harchol-Balter [8] suggest it might be better if cutoff points are chosen such that the loads on the servers are unbalanced. In [7] and [8] systems with two servers with unbalanced SITA routing policies are

studied. Suggestions are provided as to how to choose the optimal cutoff point for the two servers to minimize the slowdown. Harchol-Balter and Vesilo [9] discuss the criteria for determining cutoff points for systems with two servers. When the system becomes large, it is hard to define a set of optimal cutoff points for the system. Bachmat and Sarfati [5] derive asymptotic expressions for optimal cutoff points which are $N$ competitive with respect to minimizing average waiting time for a system with $N$ servers. However, the authors point out that the performance for a given system may not match the asymptotics.

The SITA-E policy balances the long-run workload at each server. In the short term, the highly variable job sizes can cause the situation where most of the arrivals are routed to a subset of the servers while servers for the largest jobs are idle for a long period of time. Even though SITA-V tends to improve this problem, it is still a static assignment policy. There is also very little analytic work on conditions when it will be optimal or to what degree it is suboptimal. In section III, we will compare the simulation performance of SITAE-JSQ with SITAV-JSQ, where the latter chooses cutoff points $C_i = (U/L)^{i/N}$, where $N$ is the number of servers. According to [5], such cutoff points can provide $\mathcal{O}(U^{1/N})$ normalized average waiting time $E(W)/E(X)$, where $X$ and $W$ are the processing time and waiting time, respectively. Based on the simulation results, we will see that for a bounded Pareto processing time distribution with $0 < \alpha < 1$, this SITA-V will outperform SITA-E and the SITAV-JSQ policy will slightly outperform SITA-V. However, for bounded Pareto distribution with $1 < \alpha < 2$, our SITA-V will give very poor performance and SITAV-JSQ can improve this dramatically. For all $0 < \alpha < 2$, SITAE-JSQ can provide considerable improvement over SITA-E.

There is another way to improve SITA-E: steal cycles from idle servers, in which arrivals initially assigned to a busy server are instead routed to an appropriate idle server. Harchol-Balter et al. [10] analyze cycle stealing and conclude that beneficiary arrivals may benefit unboundedly from cycle stealing, regardless of arrival variability, while the impact to donor arrivals (the arrivals whose cycles can be stolen) is comparatively small. Harchol-Balter et al. [11] show that there are situations (when the variance of processing times is very large) when SITA-E and the cycle stealing policy are detrimental to performance. In section III, we also compare the performance of SITAE-JSQ systems with cycle stealing approaches.

One important assumption for our analysis is complete resource pooling (CRP), which is first referred to by Harrison and López in [12]. They study a system with $N$ parallel servers with $m$ classes of arrivals, where each class has an associated holding cost. To minimize the cost, they assume the total overlap of processing capabilities, which leads to a limiting one dimensional reflected Brownian Motion in heavy traffic. It gives the intuition that under CRP conditions, the $N$ servers work like a super server, and there exists an optimal routing policy to minimize the cumulative cost incurred up to any time $t$. In short, the CRP conditions guarantee a one dimensional Brownian motion limit through assuming the server capacities can be exchanged or overlap.

### *B. SITA-JSQ routing policy*

First, a set of cutoff points is determined for SITA according to the processing time distribution, then a sub-interval is defined around every cutoff point. When a job arrives, if its size belongs to a flexible routing interval, the queue lengths of the server it should be allocated to by SITA and a set of adjacent servers are compared. The shortest queue is then chosen and the job is assigned to that queue. The detailed policy is shown in Figure 2. As shown in Figure 2a, SITA-JSQ2 divides the job size into $N$ intervals for $N$ servers: $[L, C_1]$, $(C_1, C_2]$, $(C_2, C_3]$, $\ldots$, $(C_{N-1}, U]$. The flexible routing interval $[C_i^1,\ C_i^2)$ around cutoff point $C_i,\ (i \in [1, N-1])$ is determined by a constant proportion $p$. A proportion $\frac{p}{2}$ of the arrivals around cutoff point $i$ are routed to the shortest queue between queue $i$ and queue $i+1$. The rest of the arrivals in the same size interval $i$ are routed to queue $i$. SITA-JSQ3 means the flexible arrivals around cutoff point $i$ can compare the queue length among servers $i$ to $i+2$. (The last group will be routed to the shortest queue among servers $[N-2, N]$, as shown in Figure 2b).In a similar manner, a set of policies SITA-JSQ$h$ ($h \leq N$) can be defined.

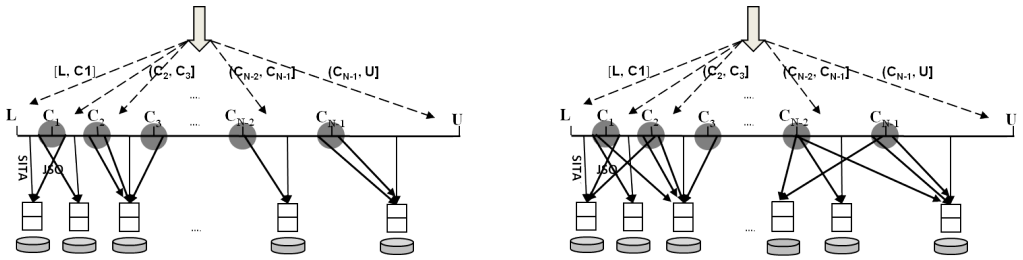
The pooling should be restricted so that the variance of each queue will not be increased to an unacceptable degree by the flexible arrivals. In Section III we study the tradeoff between adding flexibility and improving waiting times. Furthermore, a larger range of choices for flexible arrivals means more overhead in gathering queue lengths.

In this paper three problems are studied: 1. **Can SITA-JSQ outperform SITA?** This question is with respect to both unfinished workload and mean waiting time. Using diffusion approximations, we prove that SITA-JSQ satisfying a CRP condition asymptotically minimizes unfinished workload. Unfortunately, due to the dependency between queues, one cannot easily obtain explicit expressions for the mean waiting times or mean queue length [13]. In this paper we only consider the waiting time through simulations. 2. **What is the appropriate number of flexible servers?** 3. **What is the appropriate proportion of flexible arrivals $p$?**. Is the system performance sensitive to this choice? We will see below that, somewhat surprisingly, the performance is not particularly sensitive to this choice.

## II. Asymptotic limits for SITA-JSQ routing Policy

**Model**: The system has $N$ servers. Arrivals follow a Poisson process with rate $\lambda$. An arrival is immediately assigned to a server, and all the servers have the same processing capacity and process their jobs in first-come-first-served (FCFS) order. Processing times are known upon arrival. We would first like to analyze the simplest policy SITA-JSQ2.

As a reminder, the SITA-JSQ2 policy routes arrivals as follows: divide the job size into $N$ intervals: $[L,\ C_1)$, $[C_1,\ C_2)$, $\ldots$, $[C_{N-1},\ U]$. Define sets $I_1 = \{1,\ \ldots,\ N\}$, $I_2 = \{N+1,\ \ldots,\ 2N-1\}$ and $I = I_1 \bigcup I_2$. The dedicated




(a) SITA-JSQ2 routing policy.

(b) SITA-JSQ3 routing policy.

Fig. 2: SITA-JSQ routing policies

arrivals are labeled by $I_1$, while the flexible arrivals are labelled by $I_2$. So there are $|I|$ types of arrivals. Define the set $J = \{1, \ldots, N\}$ as the set of servers.

Given the processing time density function $f(x)$, and a constant proportion $p$, $0 \leq p \leq 1$ for the flexible routing intervals, then along with cutoff points calculated according to equation (1) in SITAE-JSQ2 and according to $C_i = (U/L)^{i/N}$ in SITAV-JSQ2, flexible routing intervals $[C_i^1,\ C_i^2)$ (as defined in Section I-B) are chosen for the JSQ routing policy as follows:

$$\frac{\int_{C_i^1}^{C_i} f(x)dx}{\int_{C_{i-1}}^{C_i} f(x)dx} = \frac{\int_{C_{i-1}}^{C_i^2} f(x)dx}{\int_{C_{i-1}}^{C_i} f(x)dx} = p/2, \tag{2}$$

where $C_N = U$ and $C_0 = L$. The arrival and processing rates of each arrival type $i$ are given by, respectively: $\lambda_i = \lambda p_i$ and $\mu_i = \frac{p_i}{\int_{C_i}^{C_{i+1}} xf(x)dx}$, where

$$p_i = \begin{cases} \int_{C_{i-1}^2}^{C_i^1} f(x)dx & i \in I_1, i \neq 1, N \\ \int_{C_{i-N}^1}^{C_{i-N}^2} f(x)dx, & i \in I_2. \end{cases} \tag{3}$$

Based on previous definitions, we use a matrix $\Phi = (\phi_i^j)_{|I|\times|J|}$ to represent the routing policy, where $\phi_i^j$ is the average rate at which server $j$'s time is allocated to type $i$ arrivals. Here,

$$\Phi = \begin{bmatrix} \Phi_1 \\ \hline \Phi_2 \end{bmatrix} \tag{4}$$

where $\Phi_1$ is a diagonal matrix that represents the routing of arrivals of type $I_1$ and $\Phi_2$ is an upper bidiagonal matrix that represents the corresponding routing of arrivals of type $I_2$. The entries in $\Phi$ should satisfy:

$$\begin{cases} \sum_{j\in J} \mu_i \phi_i^j = \lambda_i, & i \in I \\ \sum_{i\in I} \phi_i^j = 1, & j \in J. \end{cases} \tag{5}$$

From (3) and (5), one can obtain the entries in $\Phi$:

$$\phi_i^j = \begin{cases} \frac{\lambda_i}{\mu_i}, & i \in I_1,\ i = j \\ \sum_{k=0}^{N-j-1} \frac{\lambda_{i+k}}{\mu_{i+k}} + \\ \sum_{l=1}^{N-j} \frac{\lambda_{j+l}}{\mu_{j+l}} - (N-j), & i \in I_2,\ j = i - N \\ \sum_{k=0}^{j-2} \frac{\lambda_{i-k}}{\mu_{i-k}} + \\ \sum_{l=1}^{j-1} \frac{\lambda_{j-l}}{\mu_{j-l}} - (j-1), & i \in I_2,\ j = i - N + 1. \end{cases} \tag{6}$$

Every $\phi_i^j$ in (6) is positive and the system (5) has a unique solution, for which, according to [14], [15], the Complete Resource Pooling condition is satisfied.

We will use what is by now standard heavy traffic analysis for this system. Consider a sequence of systems indexed by $n$. According to Reiman [16] and Whitt [17] for a GI/GI/1-FCFS model, assume the heavy traffic condition,

$$\lim_{n\to\infty} \sqrt{n}(\lambda(n) - \mu(n)) = c, \quad -\infty < c < +\infty. \tag{7}$$

Here, the arrival rate and processing rate of the $n$th system are given by $\lambda(n)$ and $\mu(n)$ respectively. As $n \to \infty$, assume that

$$\lambda(n) \to \lambda, \qquad \sigma_a^2(n) \to \sigma_a^2, \tag{8}$$

$$\mu(n) \to \mu, \qquad \sigma_s^2(n) \to \sigma_s^2. \tag{9}$$

Here, $\sigma_a^2(n)$ and $\sigma_s^2(n)$ are the variances of the interarrival times and processing times, respectively. Let the unfinished processing times (virtual waiting times) of all arrivals waiting at time $t$ be $U(t)$, and let the scaled $n$th process be $\hat{U}^n(t) = U^n(nt)/\sqrt{n}$. Then $\hat{U}^n(t)$ converges to a reflected Brownian motion as $n \to \infty$,

$$\hat{U}^n(t) \xrightarrow{d} \hat{U} = RBM(\frac{c}{\mu}, \lambda(\sigma_a^2 + \sigma_s^2)), \tag{10}$$

where $\xrightarrow{d}$ denotes weak convergence, or convergence in distribution.

We are considering a sequence of queueing systems where only the arrival rate is changing. So we can assume $\lambda = \mu$, and

$$\lim_{n\to\infty} \lambda(n) = \lambda, \quad \lim_{n\to\infty} \frac{\lambda(n) - \lambda}{\sqrt{n}} = c. \tag{11}$$

Define $b_i = c\lambda_i/\lambda$ so that $\sum_{i\in I} b_i = c$, then from (11):

$$\lim_{n\to\infty} \sqrt{n}(\lambda_i(n) - \lambda_i) = c\lambda_i/\lambda = b_i.$$

Define $U^n_{i,j}(t)$ as the unfinished processing times of type $i$ arrivals at server $j$ at time $t$. Define the scaled process to be $\hat{U}^n_{i,j}(t) = \frac{U^n_{i,j}(nt)}{\sqrt{n}}$. According to [15], under the heavy traffic and CRP conditions:

$$\sum_{j\in J} \xi^*_j \sum_{i\in I} \hat{U}^n_{i,j}(t) \xrightarrow{d} RBM(\theta, \sigma^2),$$

where

$$\begin{aligned} \theta &= \sum_{i\in I} v^*_i b_i, \\ \sigma^2 &= \sum_{i\in I} (v^*_i)^2[\lambda^3_i \sigma^2_{ai} + \sum_{j\in J} \phi^j_i \mu^3_i \sigma^2_{si}], \end{aligned}$$

where $v^*_i$ is the workload contribution of type $i$ arrivals and $\xi^*_j$ is the workload contribution of server $j$, and $\sigma^2_{ai}$ and $\sigma^2_{si}$ are the variances of the interarrival time distribution and the processing time distribution of type $i$ arrivals, respectively.

According to [16], when the CRP condition is satisfied, $v^*_i$ and $\xi^*_j$ will be related as follows:

$$\xi^*_j = \max_i \mu_i v^*_i, \quad j \in J, \tag{12}$$

$$v^*_i = \min_j \xi^*_j/\mu_i, \quad i \in I. \tag{13}$$

*Proposition 2.1:* The asymptotic limit of the total unfinished processing times process in SITA-JSQ2 is optimal (with respect to the mean) and independent of $p$, the proportion of flexible arrivals.

*Proof:* From (12) and (13), $\xi^*_j$ is a constant. One can easily get $v^*_i = \frac{1}{N\mu_i}$ and $\xi^*_j = \frac{1}{N}$. Substituting these expressions, we see:

$$\theta = \sum_{i\in I} v^*_i b_i = \frac{c}{N} \int_L^U x f(x) dx. \tag{14}$$

As we have discussed, the arrival rate $\lambda_i$ and processing rate $\mu_i$ for arrival type $i$ satisfy equation (5). According to it, we can find the variance to be:

$$\begin{aligned} \sigma^2 &= \sum_{i\in I} (v^*_i)^2[\lambda^3_i \sigma^2_{ai} + \sum_{j\in J} \phi^j_i \mu^3_i \sigma^2_{si}] \\ &= \sum_{i\in I} (\frac{\lambda_i}{N^2\mu^2_i} + \lambda_i \sigma^2_{si}) \\ &= \frac{\lambda}{N^2} \int_L^U x^2 f(x) dx. \end{aligned} \tag{15}$$

Define $\hat{U}^n = \sum_{j\in J} \xi^*_j \sum_{i\in I} \hat{U}^n_{i,j}(t)$. From equations (14) and (15), the limiting process as $n \to \infty$ is independent of $p$. Also, we let $E[\hat{U}_{\text{SITA-JSQ2}}]$ denote the mean of the stationary distribution of $\hat{U}$. (The stationary distribution exists only if $c < 0$.) Then

$$E[\hat{U}_{\text{SITA-JSQ2}}] = \frac{\frac{\lambda}{N^2} \int_L^U x^2 f(x) dx}{2N|c| \int_L^U x f(x) dx}.$$

Obviously $E[\hat{U}_{\text{SITA-JSQ2}}]$ is also independent of $p$.

A lower bound on the unfinished processing times is obtained by considering an M/G/1 server working at rate $N\mu$. Given the heavy traffic condition $\lim_{n\to\infty} \frac{\lambda(n) - N\mu(n)}{\sqrt{(n)}} = c$, one can easily obtain that the limiting unfinished processing time process of this M/G/1 system weakly converges to

$$RBM(\frac{c}{N\mu}, \frac{\lambda}{N^2} \int_L^U x^2 f(x) dx),$$

which is the same as for SITA-JSQ2. ■

Somewhat surprisingly, according to Proposition 2.1, no matter how many flexible arrivals we add, the limiting process will not change. Furthermore, it shows that SITA-JSQ2 provides optimal unfinished workload under heavy traffic. However, we still need to verify whether SITA-JSQ2 can outperform SITA with respect to unfinished processing times.

*Proposition 2.2:* For all processing time distributions with finite mean and variance, under heavy traffic conditions, $E[\hat{U}_{\text{SITA-JSQ2}}] < E[\hat{U}_{\text{SITA}}]$.

*Proof:* The SITA system with $N$ servers works as $N$ $M/G/1$ systems. The arrival rate at server $i$ is given by: $\lambda_i = \lambda \int_{C_{i-1}}^{C_i} f(x) dx$, and the heavy traffic condition (11) reduces to a heavy traffic condition for each server $i$:

$$\lim_{n\to\infty} \frac{\lambda_i(n) - \lambda_i}{\sqrt{n}} = c \int_{C_{i-1}}^{C_i} f(x) dx.$$

Then $\hat{U}^n(t)$ converges to a reflected Brownian motion given by (10) as $n \to \infty$.

In our model, we first define $U^n_i(t)$ as the unfinished processing times at server $i$ at time $t$. Under the heavy traffic condition, according to (10) we have the scaled process $\hat{U}^n_i(t)$ weakly converges to

$$\hat{U}^n_i(t) \xrightarrow{d} \hat{U}_i = RBM(c \int_{C_{i-1}}^{C_i} x f(x) dx, (\frac{1}{\lambda_i} + \lambda_i \sigma^2_{si})).$$

Then the total unfinished workload of the system $\hat{U}_{SITA}$ is equal to $\sum_{i=1}^{N} \hat{U}_i$. It weakly converges to

$$\begin{aligned} &RBM(\sum_{i=1}^{N} \frac{c_i}{\mu_i}, \sum_{i=1}^{N} (\frac{1}{\lambda_i} + \lambda_i \sigma^2_{si})) \\ &= RBM(c \int_L^U x f(x) dx, \lambda \int_L^U x^2 f(x) dx). \end{aligned} \tag{16}$$

Denote the mean and variance of the limiting unfinished processing times process for SITA-JSQ2 as $\theta_{\text{SITA-JSQ2}}$ and $\sigma^2_{\text{SITA-JSQ2}}$, respectively, while the corresponding values for SITA are $\theta_{\text{SITA}}$ and $\sigma^2_{\text{SITA}}$.

From (14) and (16), obviously $\theta_{\text{SITA-JSQ2}} = \theta_{\text{SITA}}/N$ and $\sigma^2_{\text{SITA-JSQ2}} = \sigma^2_{\text{SITA}}/N^2$. Then $E[\hat{U}_{\text{SITA}}]$ will be

$$\frac{\lambda \int_L^U x^2 f(x) dx}{2|c| \int_L^U x f(x) dx} = N E[\hat{U}_{\text{SITA-JSQ2}}],$$

for $N > 1$, obviously $E[\hat{U}_{\text{SITA}}] > E[\hat{U}_{\text{SITA-JSQ2}}]$. ■

Under heavy traffic conditions, we have shown that SITA-JSQ2 performs better with respect to unfinished processing times than SITA, without any assumption on the processing time distribution (other than it having finite mean and variance).

We should also verify whether SITA-JSQ2 will perform better with respect to mean waiting time. However, because of the dependency between queues, it is hard to find an explicit approximation for the mean waiting time from the unfinished processing time process. This issue is still under investigation. A conjecture is that SITA-JSQ2 still performs well with respect to mean waiting time because it still isolates small arrivals from large arrivals. In particular, the mean waiting times may be sensitive to $p$, whereas the unfinished processing times are not. There may be processing time distributions where any SITA-like policy may not perform well (i.e. when processing time distributions have low variance). In this case, both SITA and SITA-JSQ2 would perform poorly against a policy such as JSQ. In order to investigate this in more detail, we carry out a set of simulations in the next section.

From the proof of Proposition 2.1, one can see that any policy adding complete resource pooling to SITA will achieve the asymptotic optimization of unfinished processing times. Our next step is to check whether SITA-JSQ$h$ for any $h < N$ will satisfy the CRP condition. Unfortunately, this is not the case. We consider SITA-JSQ3, but the insight below holds for larger values of $h$.

As for SITA-JSQ2, we can define the routing matrix $\Phi'(\phi_i^j) = \left[\begin{array}{c}\Phi_1' \\ \hline \Phi_2'\end{array}\right]$ where $\Phi_1' = \Phi_1$ as in (4), and $\Phi_2'$ is a matrix with entries on the diagonal and next two upper diagonals. Then the average service rate $\phi_i^j$ server $j$ allocates to type $i$ arrivals should satisfy the following system of equations:

$$\begin{cases} \phi_{N+1}^1 + \phi_{N+1}^2 + \phi_{N+1}^3 = \frac{\lambda_{N+1}}{\mu_{N+1}}, \\ \vdots \\ \phi_{2N-2}^{N-2} + \phi_{2N-2}^{N-1} + \phi_{2N-2}^{N} = \frac{\lambda_{2N-2}}{\mu_{2N-2}}, \\ \phi_{2N-1}^{N-2} + \phi_{2N-1}^{N-1} + \phi_{2N-1}^{N} = \frac{\lambda_{2N-1}}{\mu_{2N-1}}, \\ \phi_1^1 + \phi_{N+1}^1 = 1, \\ \phi_2^2 + \phi_{N+1}^2 + \phi_{N+2}^2 + \phi_{N+3}^2 = 1, \\ \vdots \\ \phi_{N-1}^{N-1} + \phi_{2N-3}^{N-1} + \phi_{2N-2}^{N-1} + \phi_{2N-1}^{N-1} = 1, \\ \phi_N^N + \phi_{2N-2}^N + \phi_{2N-1}^N = 1. \end{cases}$$

In total there are $3N-3$ variables and $2N-1$ equations. (The proportions of dedicated arrivals of type $j$ are known to be $\frac{\lambda_j}{\mu_j}$, $j \in [1, N]$). This means the solution is not unique. One can also deduce this from the routing map: the map is connected but has cycles, whereas the CRP condition requires a connected tree. Thus, SITA-JSQ3 with this routing configuration could not be analyzed through the same method. But obviously any modification to SITA-JSQ$h$ satisfying the CRP condition will produce a unique solution for the system of routing equations thus optimizing the unfinished workload.

Note that it is not obvious that moving to JSQ3 from JSQ2 will necessarily improve system performance. While JSQ3 should reduce short-term server idleness, it does create situations where a greater range of job sizes are combined at a particular queue.

In this section, we have provided a partial analysis of our proposed policies. In the next section, we continue our studies using simulation.

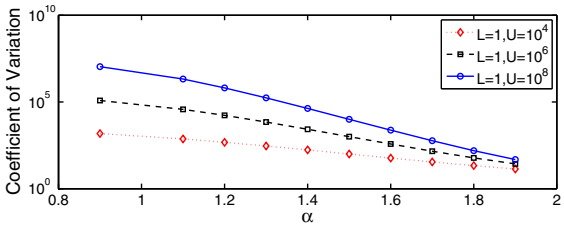



Fig. 3: CVs for different Bounded Pareto distributions

## III. Simulation Results

For all results in this section, event-driven simulations are written in C++ with approximately $10^6$ arrivals. Each result has a 95% confidence interval.

### A. Effect of Proportion of Flexible Arrivals

In the previous section, we proved that SITA-JSQ2 provides lower unfinished processing times than SITA in heavy traffic. Moreover, the asymptotic limits are independent of $p$, the proportion of flexible arrivals. A set of simulations are done to check whether this insight will still hold for mean waiting time performance. To get highly variable arrivals, we choose a number of processing time distributions that follow a Bounded Pareto distribution with density function

$$f(x) = \frac{\alpha L^\alpha x^{-\alpha-1}}{1-(\frac{L}{U})^\alpha}, \ 0 < \alpha < 2,$$

where $L$ and $U$ are the lower and upper bounds of the processing times, respectively. Figure 3 gives some rough ideas about how $\alpha$ affects the variability of Bounded Pareto distributions. According to [3], in most application cases, if the task size can be fit to a Bounded Pareto distribution, $\alpha$ is close to 1 (and no larger than 2). Here we consider three different $\alpha$: 1.22 and 0.9 (high variability), and 1.9 (low variability). The effect of changing $\alpha$ will be discussed in section III-D.

First, dedicated arrivals are chosen according to SITA-E, and the workload of all 20 servers is set to 0.96, 0.9 or 0.5 to represent different load levels. Through changing $p$, the proportion of flexible arrivals is changed. When $p = 0$, the routing policy reduces to SITA-E.

As shown in Figure 4a, when $\rho = 0.96$ (heavy traffic), the average waiting time of the system improves approximately 30% to 40% for all of the Bounded Pareto distributions

by adding only 10 percent flexible arrivals. The subsequent improvement of average waiting time drops to around 20% for 20 percent flexible arrivals. As $p$ increases, the rate of improvement decreases. This is consistent with the insight of the previous section: 1. The SITAE-JSQ2 policy will provide a large improvement over SITA-E in mean waiting time independent of the parameters of the Bounded Pareto distribution. 2. The SITAE-JSQ2 policy is not sensitive to the proportion of flexible arrivals. In Figure 4b when $\rho = 0.9$, making twenty percent of arrivals flexible will provide 30% to 40% relative improvement, and then the rate of improvement decreases as $p$ increases. Figure 4c shows that SITAE-JSQ2 does not give much improvement for light traffic, $\rho = 0.5$. This is reasonable because under light traffic flexible arrivals will often see an idle server. Thus the proportion of flexible arrivals will not affect the performance. This is also demonstrated in Figure 4c.

When adding pooling to SITA-V, the problem becomes complicated. The choice of cutoff points will significantly affect the performance. As discussed in the Introduction, several papers ( [2], [18]–[20]) consider finding cutoff points heuristically. However this is not what we focus on in this paper. A simple choice of cutoff points ($C_i = U^{i/h}$) is used and according to [5], it is asymptotically $N$-optimal for normalized waiting time. Based on this version of SITA-V, we will study whether and how adding pooling will improve it.

Figure 5 shows several sets of simulation results for SITAV-JSQ2. In Figure 5a, SITA-V based on our choice of cutoff points does not give reasonable performance for arrivals with $\alpha > 1$, while it performs very well when $\alpha < 1$. In all cases, adding pooling to it will improve the performance: when $\alpha = 1.22$, the improvement is around 20% to 30% when $p = 1$. However, when $\alpha = 1.22$, when the load is lower, adding pooling will dramatically improve the performance (see Figure 5b). When $\rho = 0.7$, the improvement is around 50% by changing all arrivals to flexible arrivals. When $\rho$ is around 0.4 through 0.6, the curve drops dramatically for some $p$. For example, the average waiting time drops from $10^4$ to 14.79 when $\rho = 0.55$. A similar effect can be seen in Figure 5c. This shows a very important idea: adding pooling can compensate for a poor choice of cutoff points. In practice, it can be very difficult to estimate the processing time distribution. As a result, for example SITA-E or SITA-V can perform far from expected when a Bounded Pareto distribution is used with an error in the estimate of $\alpha$. SITA-JSQ will increase the robustness of routing polices. On the other hand, we can see in Figure 5c that when SITA-V performs well by itself, adding pooling will improve it but the degree of improvement is not large.

### B. Effect of Increasing the Range of Flexible Servers

Even though we were unable to determine the asymptotic limits for SITA-JSQ$h$ for $h > 2$, we still would like to check whether involving more servers into the JSQ group results in significant benefits. One conjecture is there will be some point at which adding flexibility actually leads to worse performance. If arrivals are allowed to join a queue where the processing times of dedicated arrivals are very different, the variance at the queue will increase and this might offset the benefits brought by pooling. Figure 6 and Figure 7 give evidence for the conjecture. For a processing time distribution with heavier tail ($\alpha = 1.1$), this point happens later. For lighter tailed processing times ($\alpha = 1.5$), increasing the range of flexible arrivals to more than 2 does not show too much improvement. In general, from the simulation results, going beyond SITAE-JSQ5 is not advisable, while the most improvement is made simply by using SITAE-JSQ2.

### C. Effect of System Size

For the same processing time distribution, more servers means the size intervals of SITA-E will become narrower. We perform a set of simulations to check whether different server numbers require different configurations. Figure 8 shows the corresponding average waiting times. We keep the average load of each server fixed, so the arrival rate increases proportionally with the number of servers.

Both the average waiting time of SITAE-JSQ2 and SITAE-JSQ3 drop dramatically as the number of servers increases from 3 to 15, and remain relatively steady beyond that. This is reasonable because when the number of servers is small, the variance of sizes in each queue is very high, thus adding pooling will not perform very well.

### D. Effect of variability

In this section, we study the impact of processing time variance on performance. The value of $\alpha$ determines the shape of the tail of the distribution. The smaller $\alpha$ is the heavier the tail. We use CV (Coefficient of Variation) to present the variability of distributions. In terms of CV, when $\alpha$ varies from 1.1 to 1.9, CV of the Bounded Pareto distribution changes from 734.7 to 13.61 with $U = 10^4$ and from $3.727 \times 10^4$ to 26.83 with $U = 10^6$ (As shown in Figure 3). The simulation results for a system with 20 servers are shown in Figure 9. As in the previous discussion, SITAE-JSQ still outperforms SITA-E in every case, especially for arrivals with $\alpha$ close to 1 (heavy tail). For $\alpha$ around 2 (very light tail), the improvement stops. However, for arrivals with $\alpha$ close to 2, one should think about whether isolation is a good idea. Typically, SITA-like routing policies are designed for highly variable arrivals. As we can see from Figure 9, for $\alpha$ from 0.8 to 1.5, the curve drops significantly.

### E. Cycle Stealing vs. SITAE-JSQ2

A common way to realize pooling for SITA-E systems is to steal cycles from idle servers. In [2], Harchol-Balter et al. analyze the performance of cycle stealing in models where arrivals to servers processing small jobs will steal cycles from donor servers processing large jobs. One version of cycle stealing has an arrival randomly choose an idle server. If there is no idle server, follow SITA-E. However, in heavy traffic, it is rare that a server is idle. Thus there is not too much pooling. However, adding a flexible arrival to an empty queue

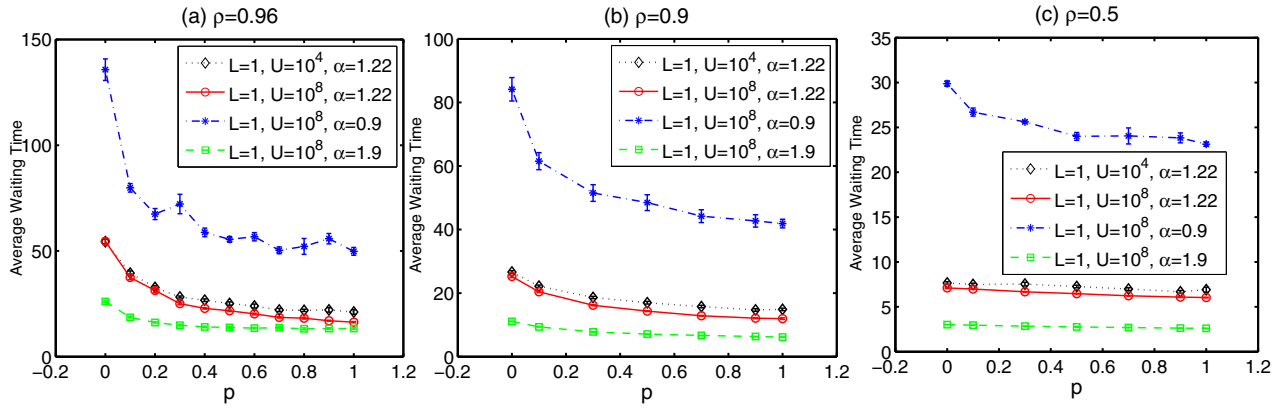



Fig. 4: Performance of SITAE-JSQ2 for different $p$

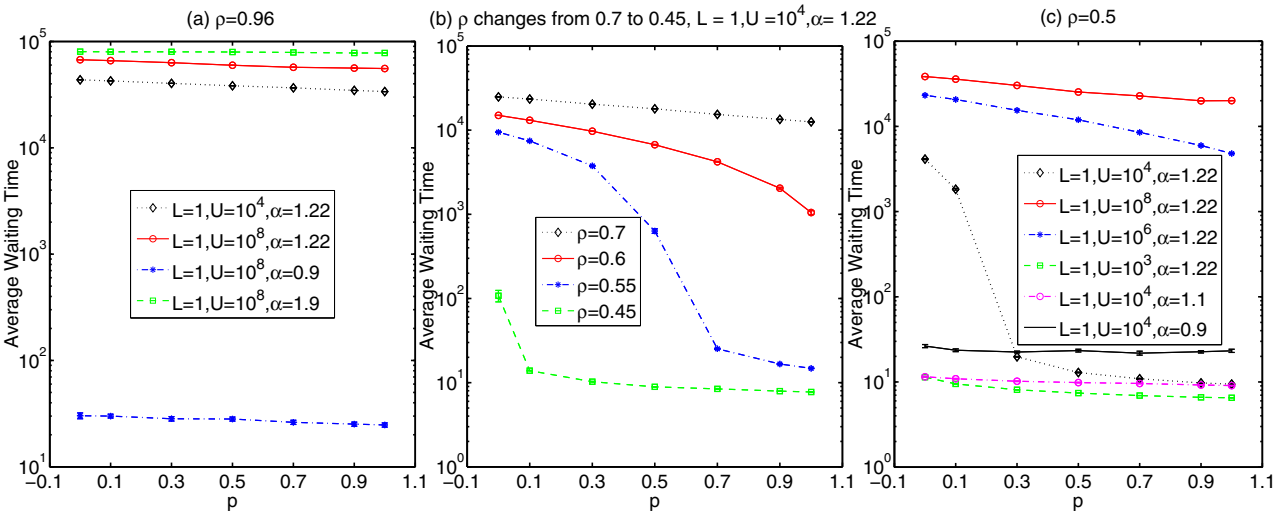
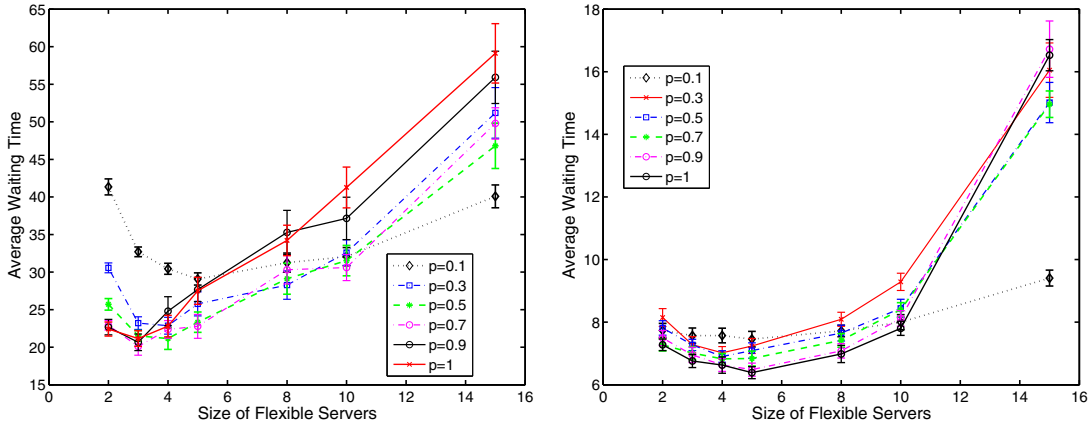


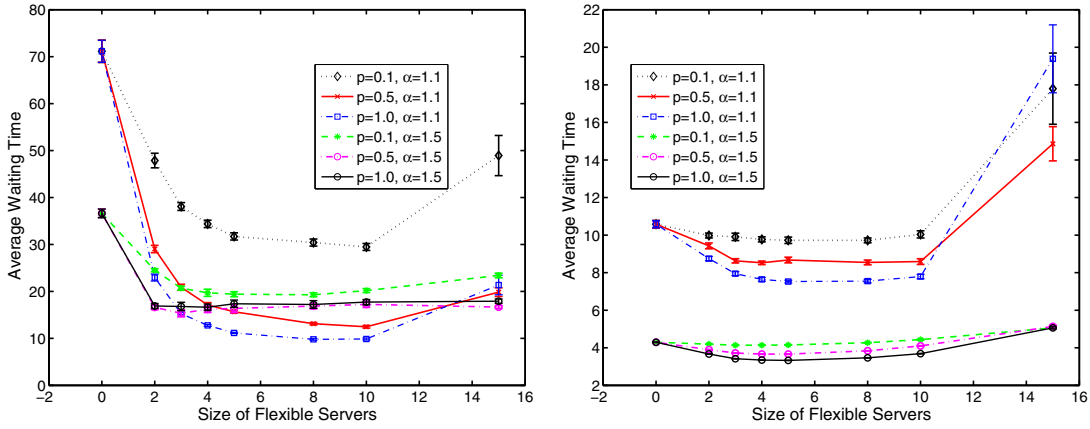

Fig. 5: Performance of SITAV-JSQ2 for different $p$




(a) $\rho = 0.96$ (b) $\rho = 0.5$

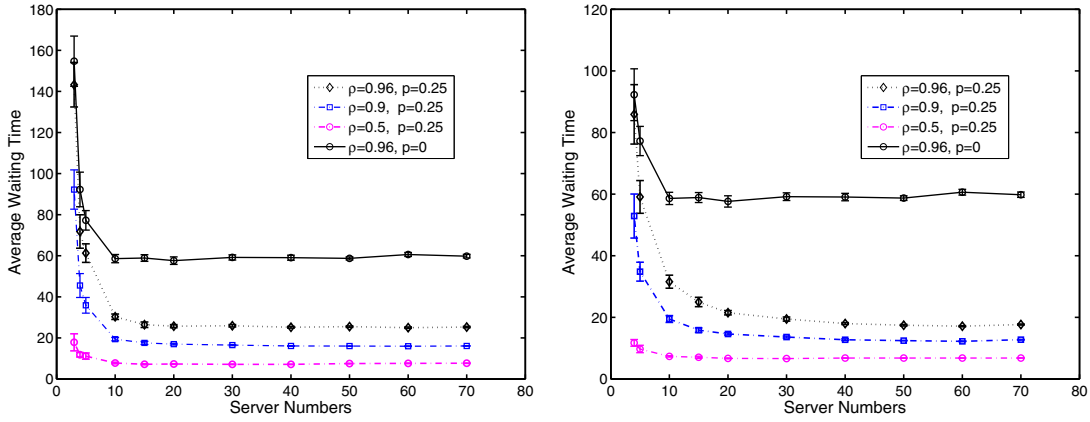
Fig. 6: Performance of SITAE-JSQ$h$ for different $p$, $L = 1$, $U = 10^4$, $\alpha = 1.22$

guarantees immediate service, while in SITAE-JSQ arrivals might still have to wait in a long queue. Another concern is that pooling arrivals with very different sizes will unacceptably increase the variability at each queue. So we also consider a




(a) $\rho = 0.96$, $L = 1, U = 10^8$ (b) $\rho = 0.5$, $L = 1, U = 10^8$

Fig. 7: Performance of SITAE-JSQ$h$ for different Bounded Pareto Processing times




(a) SITAE-JSQ2, $L = 1, U = 10^4, \alpha = 1.22$ (b) SITAE-JSQ3, $L = 1, U = 10^4, \alpha = 1.22$

Fig. 8: Performance of SITAE-JSQ$2, 3$ for different server numbers

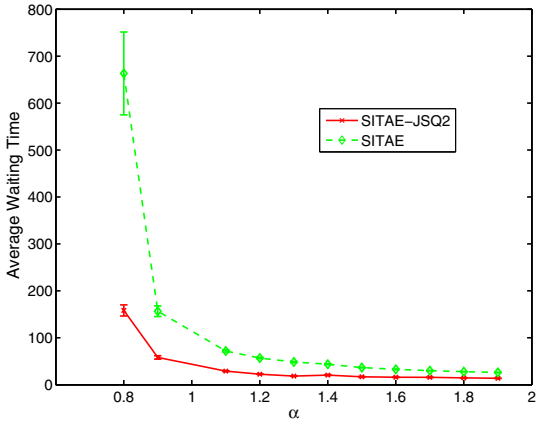
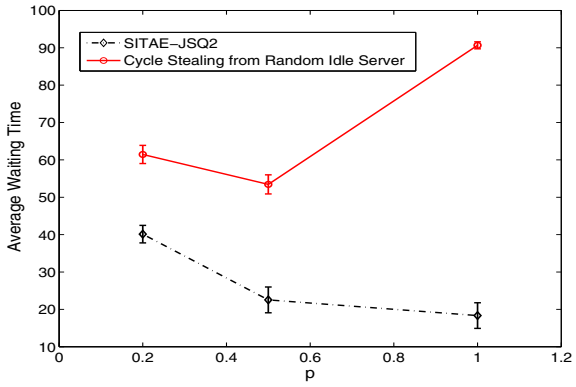
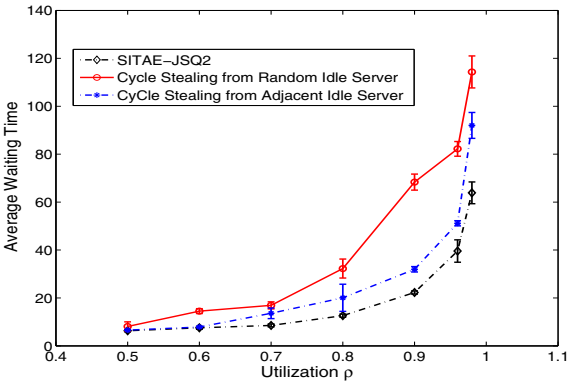



Fig. 9: Average Waiting Time versus different variances, $L = 1, U = 10^8, \rho = 0.96$

version of cycle stealing from only adjacent idle servers. To compare performance of these systems, we carry out a set of simulations for a 20 server system having the same arrival rate and processing time distribution as the bounded Pareto distribution with $L = 1, U = 10^4, \alpha = 1.22$. As shown in Figure 10, under the setting $L = 1, U = 10^4$ and $\alpha = 1.22$, cycle stealing from a random server cannot beat SITAE-JSQ for any value of utilization $\rho$. This could be due to the fact that either only idle servers are considered, or that arrivals are randomly assigned (or both). The impact of the latter choice is lessened when cycles can only be stolen from adjacent servers - when the utilization is relatively small (0.5, 0.6), the performance of SITAE-JSQ and cycle stealing from adjacent idle servers is quite close. This is reasonable because when the servers are less busy, the queue with shorter length may actually be idle. However, when the utilization becomes high, SITAE-JSQ outperforms it because the probability of idleness is low. Figure 10a compares the performance of SITAE-JSQ and cycle stealing from adjacent idle servers with varying proportion of flexible arrivals. The latter still cannot beat SITAE-JSQ no matter how many flexible arrivals are chosen.




(a) SITAE-JSQ2 vs. Cycle stealing for different $p$




(b) SITAE-JSQ2 vs.Cycle stealing for different utilizations

Fig. 10: SITAE-JSQ2 vs. Cycle stealing

This might be because when $\rho = 0.96$, there would be very few adjacent servers that are idle. So cycle stealing works almost like SITA-E.

It is hard to analyze the system with multiple donor servers. We can not guarantee that $\lambda_i$ for every $i \in I_2$ is positive, thus complete resource pooling is not guaranteed for these scenarios. A deeper analytic comparison of SITAE-JSQ2 and cycle stealing is left for future work.

## IV. Conclusions

In order to add pooling to SITA-like queueing systems to achieve simultaneously low mean waiting time and high server utilization, we propose a new routing method: SITA-JSQ. Under heavy traffic, we proved that the unfinished processing times process for the SITA-JSQ2 routing policy is minimized and does not depend on the proportion of flexible arrivals. Simulation results showed that the insights developed for unfinished processing times carry over to waiting times. SITA-JSQ can provide significant improvement in mean waiting time by adding a small proportion of flexible arrivals as well as compensating for errors in estimating the processing time distribution.

## References


[1] M. E. Crovella, M. S. Taqqu, and A. Bestavros, *Heavy-tailed probability distributions in the World Wide Web*. Cambridge, MA, USA: Birkhauser Boston Inc., 1998, pp. 3–25.

[2] M. Harchol-Balter, M. Crovella, and C. D. Murta, "On choosing a task assignment policy for a distributed server system," in *Proceedings of the 10th International Conference on Computer Performance Evaluation: Modelling Techniques and Tools*. London, UK: Springer-Verlag, 1998, pp. 231–242.

[3] M. Harchol-Balter, "Task assignment with unknown duration," *Journal of the ACM*, vol. 49, pp. 260–288, 2000.

[4] A. Riska, E. Smirni, and G. Ciardo, "Analytic modeling of load balancing policies for tasks with heavy-tailed distributions," in *Proceedings of the 2nd international workshop on Software and performance*, ser. WOSP '00. ACM, 2000, pp. 147–157.

[5] E. Bachmat and H. Sarfati, "Analysis of size interval task assignment policies," *SIGMETRICS Performance Evaluation Review*, vol. 36, no. 2, pp. 107–109, 2008.

[6] R.Vesilo, "Asymptotic analysis of load distribution for size-interval task allocation with Bounded Pareto job sizes," in *Proceedings of the 2008 14th IEEE International Conference on Parallel and Distributed Systems*, Washington, DC, USA, 2008, pp. 129–137.

[7] M. E. Crovella, M. Harchol-Balter, and C. D. Murta, "Task assignment in a distributed system : improving performance by unbalancing load," *SIGMETRICS Performence Evaluation Reveiw*, vol. 26, no. 1, pp. 268–269, 1998.

[8] B. Schroeder and M. Harchol-Balter, "Evaluation of task assignment policies for supercomputing servers: The case for load unbalancing and fairness," *Cluster Computing*, vol. 7, no. 2, pp. 151–161, 2004.

[9] M. Harchol-balter and R. Vesilo, "To balance or unbalance load in size-interval task allocation," *Probability in Engeering and Informational Science*, vol. 24, no. 2, pp. 219–244, 2010.

[10] M. Harchol-Balter, C. Li, T. Osogami, A. Scheller-Wolf, and M. S. Squillante, "Cycle stealing under immediate dispatch task assignment," in *In Proceedings of the Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*. University Press, 2003, pp. 274–285.

[11] M. Harchol-Balter, A. Scheller-Wolf, and A. Young, "Why segregating short jobs from long jobs under high variability is not always a win," in *Proceedings of the 47th annual Allerton Conference on Communication, Control, and Computing*. Piscataway, NJ, USA: IEEE Press, 2009, pp. 121–127.

[12] J. M. Harrison and J. López, "Heavy traffic resource pooling in parallel-server systems," *Queueing System: Theory and Applications*, vol. 33, pp. 339–368, 1999.

[13] D. Bertsimas, "An exact FCFS waiting time analysis for a general class of G/G/s queueing systems," *Queueing Systems: Theory and Applications*, vol. 3, pp. 305–320, 1988.

[14] Y. He and D. Down, "Limited choice and locality considerations for load balancing," *Performance Evaluation*, vol. 65, pp. 670–687, 2008.

[15] A. L. Stolyar, "Optimal routing in output-queued flexible server systems," *Probability in the Engineering and Informational Sciences*, vol. 19, pp. 141–189, 2005.

[16] M. Reiman, "Some diffusion approximations with state space collapse," in *Modelling and Performance Evaluation Methodology*, ser. Lecture Notes in Control and Information Sciences, F. Baccelli and G. Fayolle, Eds. Springer, 1984, vol. 60, pp. 207–240.

[17] W. Whitt, *Stochastic Process Limits*. Springer, 2002.

[18] M. E. Crovella., M. Harchol-Balter, and C. D. Cristina, "Task assignment in a distributed system (extended abstract): Improving performance by unbalancing load," *SIGMETRICS Performance Evalation Reveiw*, vol. 26, no. 1, pp. 268–269, 1998.

[19] N. Mi, Q. Zhang, A. Riska, and E. Smirni, "Load balancing for performance differentiation in dual-priority clustered servers," in *Quantitative Evaluation of Systems, 2006. QEST 2006. Third International Conference on*, 2006, pp. 385–394.

[20] Q. Zhang, N. Mi, A. Riska, and E. Smirni, "Performance-guided load (un)balancing under autocorrelated flows," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 5, pp. 652–665, 2008.