On the Relative Value of Local Scheduling versus Routing in Parallel Server Systems

Rong Wu
ask.com
343 Thornall Street
Edison, NJ, USA 08837

Douglas G. Down
Department of Computing and Software
McMaster University
Hamilton, Ontario, Canada L8S 4K1

Abstract

We consider a system with a dispatcher and several identical servers in parallel. Task processing times are known upon arrival. We first study the impact of the local scheduling policy at a server. To this end, we study random routing followed by a priority scheme at each server. Our numerical results show that the performance (mean waiting time) of such a policy could be significantly better than the best known suggested policies that use FCFS at each server. We then propose to use multi-layered round robin routing, which is shown to further improve system performance. Our analysis involves a combination of comparing analytic models, heavy traffic asymptotic and numerical work.

1. Introduction

A system with a front-end dispatcher and several back-end parallel servers can be used to model systems such as clustered web servers. We study models in which task processing times are known upon arrival. Many scheduling policies have been proposed and studied for such a system (we list several references below).

The scheduling policies that we consider have two components. The first is the *routing policy*, which is used to determine how tasks are assigned to servers. Examples are random routing, where an arrival is assigned to a server according to a fixed set of probabilities, round robin (RR) (the ith arriving task is assigned to the $(i \mod c)$th server, where c is the number of servers), and Join the Shortest Queue (JSQ). Notice that different information about the system state may be required by a routing rule: round robin does not require any information from the servers, while JSQ requires the queue length information at each server. The second part is the *local scheduling policy*, which schedules the order of processing at a server. Examples are First Come First Served (FCFS) and Shortest Remaining Processing Time (SRPT).

Recent research shows that high task processing time variance appears to be a feature in many aspects of computing systems, such as sizes of files transferred through the Web [2], I/O times [9], and sizes of FTP transfers in the Internet [8]. For a system with high task size variability, Harchol-Balter and her co-authors in [6] proposed a size based policy called SITA-E. There is a size range associated with each server and a task is routed to the appropriate server by its size. Each server uses FCFS local scheduling to process its assigned tasks. They also showed that under highly variable task processing time distributions, SITA-E routing may significantly outperform a routing policy that assigns arrivals to the queue with the smallest workload (with FCFS local scheduling). The advantage of SITA-E is that it reduces the variance of task processing times at each server and thus improves the overall system performance. Similar policies such as EquiLoad and AdaptLoad are suggested in the work of Riska et al. [1, 11, 15]. In Zhang et al. [14], the authors considered a size based policy for the situation where tasks are autocorrelated. They proposed a policy which distributes tasks such that the load to each server is proportional to the correlation structure of the arrival process. Their results showed that for the autocorrelated case, not all servers are equally utilized, but this imbalance brings significant performance improvement. Currently, this group of policies appears to demonstrate the best performance in the literature. They are all also relatively easy to implement.

For a single server system, Schrage in [12] showed that for an arbitrary arrival process and arbitrary processing time distribution, SRPT minimizes the number of tasks in system. Therefore if one wishes to look at such performance measures as mean waiting time, SRPT is a good choice for the local scheduling policy. In [3], we did simulations for SITA-E and round robin routing followed by shortest-remaining-processing-time local scheduling (RR-SRPT) for processing times that follow a bounded Pareto distribution. The results showed that the expected queue length for SITA-

E can be significantly larger than that for RR-SRPT. In [4], we proposed the use of multi-layered round robin routing followed by SRPT local scheduling (RRK-SRPT) for a discrete task processing time distribution (with K types of tasks). By multi-layered round robin, we mean that the dispatcher uses independent round robin policies for each type of task. We showed that RRK-SRPT performed better than RR-SRPT in heavy traffic (RRK-SRPT is asymptotically optimal in heavy traffic). This provides some evidence that similar policies may be good in general, but is not completely convincing. The main drawbacks are that [3] is based on simulation only, while in [4], continuous distributions are not considered, which in turn prevents a fair comparison with policies considered elsewhere in the literature.

There are two barriers to a further study of SRPT-based policies. On the theoretical side, it is difficult to analyze SRPT. For example, the expressions in Schrage and Miller [13] for an M/G/1 queue are quite complicated, so when we move to the multiple server environment, where in particular the arrival stream to an individual server may no longer be Poisson, exact analysis becomes quite difficult. One possibility would be to look at a diffusion approximation as in [4], but unfortunately no results exist for the diffusion approximation of a single queue under SRPT when the processing time distribution is continuous. The other barrier is on the implementation side. In most applications, SRPT is simply too difficult to implement. However, in [5], Harchol-Balter et al. show that SRPT may be approximated by a priority scheme in a manner that is straightforward to implement. We provide more details of this scheme in Section 3 below. It turns out that examining such an approximate policy allows us to set aside the difficulties in the theoretical analysis of SRPT. In [7], Jaiswal provided a formula to calculate the mean number of each type of tasks for an M/G/1 preemptive priority queue, which gives us a way to analyze the performance directly. In addition, diffusion approximations for priority systems are well studied, see Reiman [10], amongst others. We are concerned in this paper with showing that there exists a routing policy that, combined with the SRPT approximation, yields better performance than if FCFS were used as the local scheduling policy (in particular, we make comparisons with SITA-E). Thus, our contribution is not so much methodological (we apply existing results), but shows that a different viewpoint on scheduling systems of parallel servers can yield better performance. Rather than using routing to compensate for inefficiencies in the local scheduling policy (when FCFS is used in the face of high variance), we suggest using routing to play to the strengths of a more efficient (yet still implementable) local scheduling policy. If one has control over the local scheduling policy (which is not always the case), then it may be worthwhile to consider such policies. It is worth noting that many operating systems have built in support for such priority schemes.

To emphasize this viewpoint we first study a system consisting of naive (i.e. random) routing of arrivals followed by priority scheduling at each server. We give expressions for the performance of such a system that are valid over all system loads and perform a numerical study. We then give asymptotic results that show superior performance even with naive routing. We then give a simple adjustment to the routing scheme that in turn gives a performance improvement. We hope that our results will in general persuade designers to first tackle local scheduling in such systems (if at all possible) and if time/resources permit. then examine the routing issue.

The organization of this paper is as follows. In Section 2, we study the performance of preemptive priority scheduling (KLP) on a single server. We provide an algorithm to calculate the cutoff points for task types. In Section 3, we introduce the multiserver model and describe SITA-E and our proposed policy (Random-KLP). In Section 4, we compare the performance of SITA-E and Random-KLP by both numerical and heavy traffic analysis. In Section 5, we discuss the performance of RRK-KLP by simulation and heavy traffic analysis, demonstrating additional performance gains. Section 6 provides some concluding remarks.

2. K Level Preemptive Priority on a Single Server

In this section, we describe the proposed policy for a single server. We assume that arrivals follow a Poisson process with rate λ_0. The processing times are assumed to be independent and identically distributed (i.i.d.) and follow a density $f(x)$. We also assume that the processing times are known upon arrival. We have mentioned that the authors in [5] suggested to use preemptive priority scheduling to approximate SRPT. For the proposed policy, we partition the support of the processing time distribution into K intervals. A task whose processing time lies in interval k given by $[y_{k-1}, y_k)$ is called a type k task. Type i tasks have preemptive priority over type j tasks if $i < j$. The proportion of type k tasks is $\alpha_k = \int_{y_{k-1}}^{y_k} f(y)dy$, and the conditional density function for the size of a task given that it is type k is $f_k(y) = f(y)/\alpha_k$, $y \in [y_{k-1}, y_k)$. A server then uses FCFS within tasks of the same type. There are K priority levels. Note that this differs slightly from the policy in [5], as they appear to implement a mechanism that dynamically changes a task's priority. The performance of their policy should actually be better than that described here but is difficult to analyze. On the other hand, the proposed policy is very straightforward to implement, if a priority scheduling policy is available. We call such a policy K level preemptive priority (KLP).

Jaiswal in [7] derived the formula for the mean queue length of type k tasks ($\bar{Q}_k$) for an M/G/1 preemptive priority queue, where type k' tasks have preemptive (resume) priority over type k tasks if $k' < k$. Let $\lambda_k = \alpha_k \lambda_0$, $\mu_k = 1/(\int_{y_{k-1}}^{y_k} y f_k(y)dy)$, and $\rho_k = \lambda_k/\mu_k$. For KLP, if $\sum_{i=1}^{k} \rho_i < 1$, from (7.42) of [7], we have

$$\bar{Q}_k = \frac{\rho_k}{1-\sum_{i=1}^{k-1}\rho_i} + \frac{\lambda_k \sum_{i=1}^{k} \lambda_i E[Y_i^2]}{2(1-\sum_{i=1}^{k-1}\rho_i)(1-\sum_{i=1}^{k}\rho_i)}, \tag{1}$$

where $E[Y_i^2] = \int_{y_{i-1}}^{y_i} y^2 f_i(y)dy$, and $\rho_k = \lambda_k/\mu_k$ is the load due to type k tasks. The total mean queue length over all types, $\bar{Q}_0$ is

$$\begin{aligned}\bar{Q}_0 &= \sum_{k=1}^{K}\left(\frac{\rho_k}{1-\sum_{i=1}^{k-1}\rho_i}\right. \\ &\qquad \left. + \frac{\lambda_k \sum_{i=1}^{k} \lambda_i E[Y_i^2]}{2(1-\sum_{i=1}^{k-1}\rho_i)(1-\sum_{i=1}^{k}\rho_i)}\right). \end{aligned} \tag{2}$$

The expression for the mean total queue length is quite complicated. One would like to compute the cutoffs for task type i using (2), however this is quite difficult. In general, one should choose as many intervals as possible. If, for implementation reasons, one wishes to decrease the number of task types, Procedure 1 below gives a means to use (2) to find a reasonable number of task types.

Procedure 1 *For an M/G/1 system operated under KLP with arrival rate λ_0, if the task processing times follow a continuous distribution with density function $f(x)$, the cut-off points can be calculated by the following steps.*

Step i: *Partition the support of the processing time distribution into N intervals (we also call them N types) such that $\rho_i = \rho_j$ for all $1 \leq i,\ j \leq N$ and $i \neq j$, where ρ_i is the load for type i tasks ($\rho_i = \lambda_i/\mu_i$). Calculate the total mean queue length $\bar{Q}_N$ with the N intervals by using (2). Let $\bar{Q}_{min} = \bar{Q}_N$.*

Step ii: ***If*** *$N = 1$*
then *stop;*
else *decrease the number of intervals by 1 by merging two adjacent intervals. Note that this involves $N-1$ cases. Calculate the mean queue length for each of the $N-1$ cases with $N-1$ intervals and take the minimum mean queue length $\bar{Q}_{N-1}$. When merging intervals k and $k+1$ ($1 \leq k \leq N-1$), the load for the merged interval is $\rho_k + \rho_{k+1}$.*

Step iii: ***If*** *$\frac{\bar{Q}_{N-1}-\bar{Q}_{min}}{\bar{Q}_{min}} > \delta$ ($\delta \geq 0$, is chosen by the user);*
then *stop, return the N intervals.*
else *go to the next step.*

Step iv: ***If*** *$\bar{Q}_{N-1} < \bar{Q}_{min}$*
then *$\bar{Q}_{min} = \bar{Q}_{N-1}$.*
Let $N = N-1$, go back to Step ii.

We now study the performance numerically. Tables 1 and 2 provide the mean queue length for different cutoff points (K is the number of intervals) with different system loads using processing times following a bounded Pareto distribution, varying the parameters to achieve different workloads (ρ). The bounded Pareto distribution has density f given by:

$$f(x) = \begin{cases} \frac{\alpha p_1^{\alpha}}{1-(p_1/p_2)^{\alpha}} x^{-\alpha-1} & 0 < p_1 \leq x \leq p_2 \\ 0 & \text{otherwise} \end{cases}$$

where $0 < \alpha < 2$ is the exponent of the power law, p_1 is the smallest possible observation, and p_2 is the largest possible observation. We let $p_1 = 512$ and $p_2 = 10^{10}$. The reason we choose these parameters is to keep consistent with the study in [6]. Tables 1 and 2 show how, for various α, expression (2) varies when one starts with 20 intervals and uses Procedure 1 to reduce the number of intervals. It is not hard to see that varying δ will yield different results. Further, note that if we choose the tolerance as small as $\delta = 0.1$, for the bounded Pareto distribution, we could achieve better mean queue length using 4-6 types. If we allow a higher tolerance, for moderate and light system loads, good performance can be achieved using just two types. In the next section, we apply KLP to multiple servers.

We now move to showing how well parallel server systems (each implementing KLP) perform, under naive (random) routing.

Table 1. Mean Queue Length with $\rho = 0.9$ for a Single Server

K	$\alpha = 1.2$	$\alpha = 1.5$	$\alpha = 1.9$
3	27.030	8.869	4.412
4	4.800	8.234	3.838
6	2.935	7.801	3.736
7	2.808	7.790	3.735
9	2.700	7.774	3.742
13	2.670	7.779	3.758
20	2.675	7.795	3.782

3. Parallel Server Model

In this section, we introduce the system model for the proposed policy, SITA-E, and related policies. We assume

Table 2. Mean Queue Length with $\rho = 0.5$ for a Single Server

K	$\alpha = 1.2$	$\alpha = 1.5$	$\alpha = 1.9$
2	5.788	2.113	0.939
3	2.573	1.643	0.766
4	0.862	0.838	0.752
5	0.808	0.813	0.749
6	0.760	0.811	0.750
8	0.717	0.810	0.752
13	0.711	0.812	0.757
20	0.713	0.816	0.762

that arrivals follow a Poisson process with rate λ. Upon arrival, a task must be immediately assigned to one of c identical servers. The processing times are assumed to be i.i.d. and follow a density $f(x)$ which satisfies $\int_0^\infty x^2 f(x)dx < \infty$. Note that this discounts some heavy-tailed distributions, but certainly allows for processing times with high variance (we consider bounded Pareto distributions later). We let $\{u_i, i \geq 1\}$ and $\{v_i, i \geq 1\}$ be the interarrival and processing time sequences, μ be the processing rate, and s the variance of the processing times.

3.1. SITA-E and related policies

For SITA-E [6], an arrival is routed to server k if its processing time lies in the range $[x_{k-1}, x_k)$, where $x_0 = 0$ and $x_c = \infty$. The intervals are chosen such that

$$\int_0^{x_1} xf(x)dx = \int_{x_1}^{x_2} xf(x)dx = \cdots = \int_{x_{c-1}}^{\infty} xf(x)dx.$$

The local scheduling policy at each server is FCFS. Note that the choice of intervals balances the load at each of the servers, while at the same time yielding a processing time variance at each server that is typically much less than the overall processing time variance. Simulation results in [6] show that this policy appears to significantly outperform other policies (in particular a routing policy that assigns arrivals to the queue with the smallest workload, followed by FCFS local scheduling).

Other size based policies in [1, 11, 15] are similar to SITA-E. If the task processing time distribution is fixed, the performance of these policies is the same as SITA-E. Therefore, in this paper, we only compare the proposed policy with SITA-E. Notice that we do not consider the situation when the task processing time distribution is not fixed. We refer the reader to [1, 11, 15] for policies that react to time varying workloads.

3.2. Random-KLP

For the proposed policy, we would like to use KLP as the local scheduling policy. To focus on the relative importance of the local scheduling policy, we first consider a naive routing policy (random routing).

4. SITA-E and Random-KLP - Performance Analysis

In this section, we compare SITA-E and Random-KLP both numerically and asymptotically (in a heavy traffic sense). Section 4.1 compares the performance by using exact analytic expressions. In Section 4.4, we compare the performance in heavy traffic.

4.1. Analytic models

In this section, we compare the performance of SITA-E and Random-KLP by providing expressions for both the mean waiting time and slowdown. The formulas are of sufficient complexity that a direct comparison is not possible, so we provide numerical results to support our findings. Section 4.2 provides the numerical results for mean queue length. Section 4.3 discusses the slowdown of each task type.

4.2. Mean queue length

We are interested in minimizing the mean waiting time, but due to Little's Law, we can equivalently look at the mean number of tasks in the system.

Let $\bar{Q}_k$ denote the mean queue length at server k and $\bar{Q}$ represent the total mean queue length for SITA-E. Also, let $\beta_k = \int_{x_{k-1}}^{x_k} f(x)dx$ be the proportion of arrivals that are assigned to server k. Define $\lambda_k = \beta_k \lambda$ and $\mu_k = 1/(\int_{x_{k-1}}^{x_k} xf(x)/\beta_k dx)$.

We then have, using the *Pollaczek-Kinchin* formula for each server,

$$\bar{Q} = \sum_{k=1}^{c} \left(\rho_k + \frac{\rho_k^2 + \lambda_k^2 s_k}{2(1-\rho_k)} \right), \qquad (3)$$

where $\rho_k = \lambda_k/\mu_k$, and $s_k = \int_{x_{k-1}}^{x_k} x^2 f(x)/\beta_k dx - \left(\int_{x_{k-1}}^{x_k} xf(x)/\beta_k dx\right)^2$.

In Section 2, we provided the formula for total mean queue length of KLP at a single server. Let $\lambda'_k = \alpha_k \lambda / c$ and $\mu'_k = 1/(\int_{y_{k-1}}^{y_k} yf_k(y)dy)$. For Random-$K$LP, the total mean queue length ($\bar{Q}'$) is thus

$$\bar{Q}' = c\sum_{k=1}^{K}\left(\frac{\rho'_k}{1-\sum_{i=1}^{k-1}\rho'_i} + \frac{\lambda'_k\sum_{i=1}^{k}\lambda'_i E[Y_i^2]}{2(1-\sum_{i=1}^{k-1}\rho'_i)(1-\sum_{i=1}^{k}\rho'_i)}\right), \quad (4)$$

where $E[Y_i^2] = \int_{y_{i-1}}^{y_i} z^2 f_i(z)dz$, and $\rho'_k = \lambda'_k/\mu'_k$, which is the load due to type k tasks at a server.

It is hard to compare $\bar{Q}$ and $\bar{Q}'$ since the expressions (3) and (4) are quite complicated. Thus, we use the expressions (3) and (4) to gain insight through numerical comparisons. Tables 3 and 4 provide some numerical results for SITA-E and Random-KLP using processing times following a bounded Pareto distribution, varying the parameters to achieve different workloads (ρ).

In the "Policy" column in Tables 3 and 4 we add a symbol '*' to represent the case when the cutoff points are chosen to balance the loads for each type of tasks at each server. This means we choose a cutoff point such that $\lambda_i/\mu_i = \lambda_j/\mu_j$ (λ_i represents the arrival rate for type i tasks to a single server), where $1 \le i, j \le K$. For the policies without '*', we use the cutoff points calculated according to Procedure 1 (with a starting point of 20 intervals). Note that for Random-KLP, we get the results directly from the single server formula.

Table 3. Mean Queue Length with $\rho = 0.9$

Policy	c	$\alpha = 1.2$	$\alpha = 1.5$	$\alpha = 1.9$
SITA-E	4	896.018	1510.558	78.115
Random-3LP	4	108.120	35.476	22.060
Random-4LP	4	19.200	32.936	15.352
Random-4LP*	4	680.337	1158.060	57.778
Random-7LP	4	11.232	31.160	14.940
Random-9LP	4	10.800	31.096	14.968
Random-13LP	4	10.680	31.180	15.032
SITA-E	8	153.127	785.025	92.038
Random-3LP	8	216.240	70.952	35.296
Random-7LP	8	22.464	62.320	29.880
Random-8LP*	8	83.739	459.658	55.144
Random-13LP	8	21.360	62.232	30.064
SITA-E	16	103.411	451.920	126.281
Random-4LP	16	76.800	131.744	61.408
Random-13LP	16	42.720	124.464	60.512
Random-16LP*	16	47.386	191.441	66.797

The numerical results for the mean queue length under both policies show that, under bounded Pareto distributions, if we choose the same task intervals for Random-KLP and SITA-E, Random-KLP consistently outperforms SITA-E for different system loads and task processing time variance. A further significant improvement can be made by using Procedure 1. While these numerical results are quite suggestive, they are not conclusive. We provide additional evidence in the form of heavy traffic asymptotic results in Section 4.4.

Table 4. Mean Queue Length with $\rho = 0.5$

Policy	c	$\alpha = 1.2$	$\alpha = 1.5$	$\alpha = 1.9$
SITA-E	4	57.079	95.027	6.599
Random-2LP	4	23.152	8.452	3.756
Random-4LP	4	3.448	3.352	3.008
Random-4LP*	4	24.380	39.684	4.383
Random-5LP	4	3.232	3.252	2.996
SITA-E	8	13.005	52.014	9.239
Random-4LP	8	6.896	6.704	6.016
Random-5LP	8	6.464	6.504	5.992
Random-8LP	8	5.736	6.480	6.016
Random-8LP*	8	7.170	16.005	6.658
SITA-E	16	13.495	35.007	14.905
Random-5LP	16	12.928	13.008	11.984
Random-8LP	16	11.472	12.960	12.032
Random-13LP	16	11.376	12.992	12.112
Random-16LP*	16	11.489	14.335	12.297

4.3. Slowdown

Slowdown is a performance metric that incorporates the notion of fairness. For any policy, if $E[W(x)]$ is the mean waiting time for a task of size x, the *slowdown* for a task of size x, $S(x)$, is given by

$$S(x) = \frac{E[W(x)]}{x}. \quad (5)$$

For SITA-E, the mean waiting time for type k ($1 \le k \le c$) tasks is

$$E[W(x)] = 1/\mu_k + \frac{\rho_k^2 + \lambda_k^2 s_k}{2\lambda_k(1-\rho_k)}, \quad x_{k-1} \le x < x_k. \quad (6)$$

For Random-KLP, the mean waiting time for type k ($1 \le k \le K$) tasks is

$$E[W'(x)] = \frac{\rho'_k}{\lambda'_k(1-\sum_{i=1}^{k-1}\rho'_i)} + \frac{\sum_{i=1}^{k}\lambda'_i E[X_i^2]}{2(1-\sum_{i=1}^{k-1}\rho'_i)(1-\sum_{i=1}^{k}\rho'_i)}, \quad (7)$$

where $x_{k-1} \le x < x_k$. Combining (5), (6), and (7), we can compare the slowdown for each type of tasks under SITA-E and Random-KLP.

Figure 1 gives the slowdown for tasks under SITA-E and Random-4LP with 4 servers, where $\alpha = 1.2$ and $\rho = 0.9$. The cutoff points for Random-4LP are calculated according to Procedure 1.

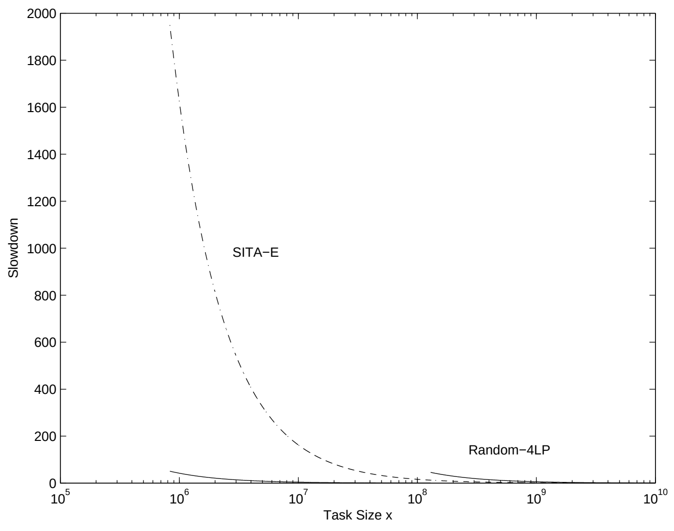

Figure 1. Slowdown for Random-4LP (unbalanced loads) and SITA-E

The results show that for a bounded Pareto distribution, if we use the cutoff points calculated by Procedure 1, the slowdown of Random-KLP for some tasks may not be better, although the mean queue length is better. However, if we use the same types for SITA-E and Random-KLP, the slowdown for each type of tasks for Random-KLP appears to be better than that of SITA-E. Figure 2 illustrates such a situation, where we choose four types with balanced work loads for $\alpha = 1.2$ and $\rho = 0.9$.

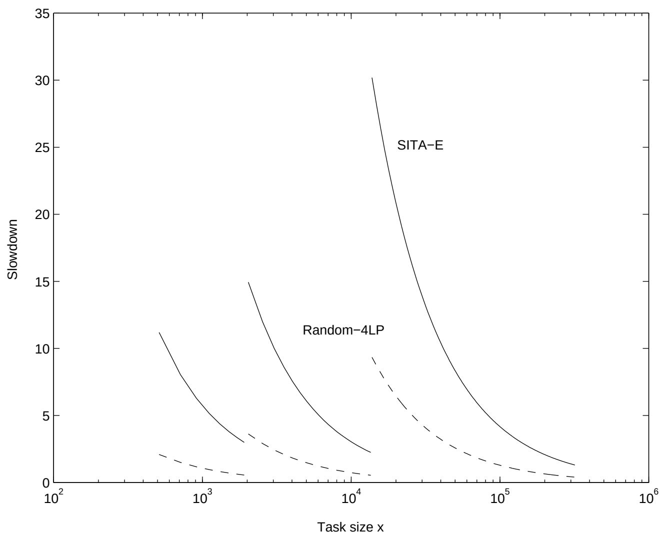

Figure 2. The Slowdown for SITA-E and Random-4LP (balanced loads)

Our numerical studies suggest that the performance for Random-KLP is better than that of SITA-E for bounded Pareto distributions if we use the same task type intervals. To achieve lower mean queue length, we could use Procedure 1 with lower tolerance (*e.g.* 0.1) and a larger number of starting points (20 in our examples) to partition tasks. However, the tradeoff for this is that the slowdown for certain types of task for Random-KLP could be worse than for SITA-E. The expressions we use here allow one to experiment to choose a good performance tradeoff, according to the designer's goals (there is no need to resort to simulation).

4.4. Performance analysis in heavy traffic

In the previous section, we provided evidence of the relative benefits of local scheduling versus routing. In this section, while we are unable to provide a proof that Random-KLP outperforms SITA-E over all loads, we are able to show better performance in an asymptotic sense, when the load on the system approaches one. We use the tool of heavy traffic limits to approximate the mean queue length for SITA-E and Random-KLP. To do this, assume that there is a sequence of systems indexed by (n) such that $\lambda(n) \to \lambda$, $\mu(n) \to \mu$, and $s(n) \to s$, as $n \to \infty$. Further assumptions on the sequence of systems will be made as needed below.

4.5. SITA-E

We first consider the performance of SITA-E. For each server k, let $\{u_i^k, i \geq 1\}$, $\{v_i^k, i \geq 1\}$ represent the interarrival and processing time sequences, respectively. For the sequence of systems indexed by (n), we have for $1 \leq k \leq c$,

$$\begin{aligned}
\lambda_k(n) &= (E[u_1^k(n)])^{-1} = \beta_k \lambda(n) \longrightarrow \lambda_k = \beta_k \lambda, \\
a_k(n) &= Var(u_1^k(n)) = \frac{1}{\beta_k^2 \lambda^2(n)} \longrightarrow a_k = \frac{1}{\beta_k^2 \lambda^2}, \\
\mu_k(n) &= (E[v_1^k(n)])^{-1} = \frac{1}{\int_{x_{k-1}}^{x_k} x f_k^{(n)}(x) dx} \\
&\longrightarrow \mu_k = \frac{1}{\int_{x_{k-1}}^{x_k} x f_k(x) dx}, \\
s_k(n) &= Var(v_1^k(n)) \\
&= \int_{x_{k-1}}^{x_k} x^2 f_k^{(n)}(x) dx - \left(\int_{x_{k-1}}^{x_k} x f_k^{(n)}(x) dx \right)^2 \\
&\longrightarrow s_k = \int_{x_{k-1}}^{x_k} x^2 f_k(x) dx - \left(\int_{x_{k-1}}^{x_k} x f_k(x) dx \right)^2 .
\end{aligned}$$

We also require $\sup_{n \geq 1} E[(u_1^k(n))^{2+\varepsilon}] < \infty$ and $\sup_{n \geq 1} E[(v_1^k(n))^{2+\varepsilon}] < \infty$, for some $\varepsilon > 0$.

The load of type k tasks in the nth system is defined as $\rho_k(n) = \lambda(n)\beta_k/\mu_k(n)$, $1 \le k \le K$. We have

$$\rho_k(n) \longrightarrow \rho_k = \beta_k\lambda/\mu_k \tag{8}$$

for all k. We assume that $\rho_k(n) < 1$ for all k and n, and $\rho_k(n) \to 1$ as $n \to \infty$ for all k.

We define $Q_k(t)$ to be the number of tasks at server k at time t. We are interested in the process $\hat{Q}_k^{(n)}(t) = n^{-1/2}Q_k^{(n)}(nt)$. We assume that $Q_k^{(n)}(0) = 0$, $1 \le k \le c$. For each server k, we also need to define $d_k(n) = \sqrt{n}\,(\rho_k(n) - 1)$. We assume

$$d_k(n) \quad\longrightarrow\quad d \quad \text{as } n \to \infty, -\infty < d < 0. \tag{9}$$

We call (9) the heavy traffic condition (note that (9) implies our earlier assumption that $\rho_k(n) \to 1$ as $n \to \infty$). We now have the following result, where RBM(a, b) denotes a reflected Brownian motion with drift a and variance b. Also, $\Longrightarrow$ denotes weak convergence in the metric space D consisting of all right continuous functions with left limits.

Theorem 1 *For queue k ($1 \le k \le c$) in a system operating under SITA-E, under the heavy traffic conditions,*

$$\hat{Q}_k^{(n)} \quad\Longrightarrow\quad \mu_k RBM\left(d, \beta_k\lambda s_k + \frac{1}{\beta_k\lambda}\right). \tag{10}$$

Proof. This follows directly from Theorem 3.4 of [10]. ■

4.6. Random-KLP

We focus on one server and separate the K different types into different arrival streams for that server. For each type k, we use $\{\tilde{u}_i^k,\ i \ge 1\}$, $\{\tilde{v}_i^k,\ i \ge 1\}$ to represent the interarrival and processing time sequences (at each server), respectively. Without loss of generality, we have $\tilde{u}_1^k = \sum_{j=1}^c u_j^k$ and $\tilde{v}_1^k = v_1^k$, where $\{u_i^k, i \ge 1\}$ and $\{v_i^k, i \ge 1\}$ are the interarrival and processing time sequences for type k tasks to the system. If λ_k', a_k', μ_k', and s_k' are, respectively, the arrival rate, interarrival time variance, service rate, and processing time variance for type k tasks at a single server, we then have

$$\begin{aligned}
\lambda_k'(n) &\longrightarrow \lambda_k' = \left(E[\sum_{j=1}^{c} u_j^k]\right)^{-1} = \alpha_k\lambda/c, \\
a_k'(n) &\longrightarrow a_k' = Var(\sum_{j=1}^{c} u_j^k) = \frac{c^2}{\alpha_k^2\lambda^2}, \\
\mu_k'(n) &\longrightarrow \mu_k = (E[v_1^k])^{-1} = \frac{1}{\int_{y_{k-1}}^{y_k} y f_k(y)dy}, \\
s_k'(n) &\longrightarrow s_k = Var(v_1^k) \\
&= \int_{y_{k-1}}^{y_k} y^2 f_k(y)dy - \left(\int_{y_{k-1}}^{y_k} y f_k(y)dy\right)^2.
\end{aligned} \tag{11}$$

We also require $\sup_{n\ge1} E\left[(\sum_{j=1}^c u_j^k(n))^{2+\varepsilon}\right] < \infty$ and $\sup_{n\ge1} E[(\tilde{v}_1^k(n))^{2+\varepsilon}] < \infty$, for some $\varepsilon > 0$.

We define $Q_k'(t)$ to be the number of type k tasks at a single server at time t. As in the previous section, we are interested in the process $\hat{Q'}_k^{(n)}(t) = n^{-1/2}Q'^{(n)}_k(nt)$. We assume that $Q'^{(n)}_k(0) = 0$, $1 \le k \le K$. We also define $d_k'(n) = \sqrt{n}\left(\sum_{i=1}^k \rho_i'(n)/c - 1\right)$, where $\rho_i'(n) = \lambda(n)\alpha_i/\mu_i'(n)$ denotes the load of type i tasks on the system.

We make the following assumptions

$$d_k'(n) \quad\longrightarrow\quad -\infty \quad \text{as } n \to \infty, 1 \le k \le K-1, \tag{12}$$
$$d_K'(n) \quad\longrightarrow\quad d' \quad \text{as } n \to \infty, -\infty < d' < 0. \tag{13}$$

From the definition of $\rho_k'(n)$, we have the load of type k tasks on a single server

$$\rho_k'(n)/c \longrightarrow \rho_k'/c \tag{14}$$

as $n \to \infty$, $0 \le \rho_k'/c < 1$, $1 \le k \le K-1$. We call (12), (13), and (14) the *heavy traffic* conditions. We can now give a result for the heavy traffic limit under Random-KLP.

Theorem 2 *For a single queue operating under Random-KLP, under the heavy traffic conditions,*

$$\hat{Q'}_k^{(n)} \quad\Longrightarrow\quad 0, \quad 1 \le k \le K-1, \tag{15}$$

$$\hat{Q'}_K^{(n)} \quad\Longrightarrow\quad \mu_K' RBM\left(d', \sum_{k=1}^{K}\left(\frac{\alpha_k\lambda s_k'}{c} + \frac{\alpha_k\lambda(\int_{z_{k-1}}^{z_k} y f_k(y)dy)^2}{c}\right)\right). \tag{16}$$

Proof. This follows directly from Theorem 4.1 and Theorem 4.3 of [10]. ■

4.7. Comparison

Now we can compare SITA-E and Random-KLP in heavy traffic. We first need to compute the mean of the heavy traffic limit for the total number of tasks in the system. Let $\bar{Q}_{\text{Random}} = c\bar{Q'}_{\text{Random}}$, where $\bar{Q'}_{\text{Random}}$ is the mean of the RBM in (16). Also let $\bar{Q}_{\text{SITA-E}} = \sum_{k=1}^c \bar{Q}^k_{\text{SITA-E}}$, where $\bar{Q}^k_{\text{SITA-E}}$ is the mean of the RBM in (10). The following theorem suggests that RRK-KLP should outperform SITA-E under high loads, for all processing time distributions with finite variance.

Theorem 3 *For a continuous processing time distribution with density function $f(x)$, where $\int_{-\infty}^{\infty} x^2 f(x)dx < \infty$,* *in heavy traffic, if the processing time distribution is partitioned for Random-KLP in an identical manner as for SITA-E (which in turn implies $K = c$ for Random-KLP), $\bar{Q}_{\text{Random}} < \bar{Q}_{\text{SITA-E}}$.*

Proof. First, we note that the fact that $K = c$ implies that the assumptions we made for both systems are in fact equivalent, so the comparison is fair. It also implies that $x_k = y_k$, $d = d'$, $\mu_k = \mu'_k$, $a_k = a'_k$, $s_k = s'_k$, and $\alpha_k = \beta_k$, so in the remainder of the proof we will use the quantities on the left hand side of each of the above equalities. First, for SITA-E,

$$\begin{aligned}\bar{Q}_{\text{SITA-E}} &= \sum_{k=1}^{K} \mu_k \frac{1}{2|d|}\left(\frac{1}{\alpha_k \lambda} + \alpha_k \lambda s_k\right) \\ &= \frac{K}{2|d|} + \frac{1}{2|d|}\sum_{k=1}^{K} \alpha_k \lambda s_k \mu_k.\end{aligned}$$

The last step is due to the heavy traffic conditions (the load on server k approaches 1). Now, for Random-KLP,

$$\begin{aligned}\bar{Q}_{\text{Random}} &= \frac{c\mu_K}{2|d|}\sum_{k=1}^{K}\left(\frac{\alpha_k \lambda}{c\mu_k^2} + \frac{\alpha_k \lambda s_k}{c}\right) \\ &< \frac{1}{2|d|}\sum_{k=1}^{K}\frac{\alpha_k \lambda \mu_K}{\mu_k \mu_K} + \frac{1}{2|d|}\sum_{k=1}^{K}\alpha_k \lambda s_k \mu_k \\ &= \frac{K}{2|d|} + \frac{1}{2|d|}\sum_{k=1}^{K}\alpha_k \lambda s_k \mu_k \\ &= \bar{Q}_{\text{SITA-E}}.\end{aligned}$$

So, we have a result that suggests that Random-KLP outperforms SITA-E for heavily loaded systems. Note that this is for a particular choice of K, so if we were to find the optimal value of K (and the associated intervals $[y_{k-1}, y_k)$), one would expect the performance to further improve. However, we will see that if we were to use the limiting process to do such an optimization, the problem is not well-posed. We emphasize that the main result of this section provides a *guarantee* of better performance with respect to the limiting processes. Of course, one suggestion would be to use the techniques described in Section 2.

Using the asymptotic expressions, it is not possible to calculate the optimal cutoff points, even for the case of $K = 2$. Let T be the sole cutoff point for $K = 2$. We cannot use (16) to calculate the cutoff point. The reason is that the expression for $\bar{Q}_{\text{Random}}$ is decreasing as T is increasing. As T goes to the maximum value of the task processing time, α_1 goes to 1 and μ'_1 goes to μ. Hence the load for type 1 tasks is $\lambda/(c\mu)$, which equals 1. This contradicts the assumption that $\rho'_1 < 1$. In other words, (15) will not hold for this case. The fact that the optimal value of T in (16) leads to a situation which violates the assumptions required for Theorem 2 leads to a logical inconsistency that prevents us from using the limiting processes to find the optimal cutoff point. Such problems also arise for larger values of K.

5. Multi-layered Round Robin Routing

We have provided some guidelines for parameter choices such that Random-KLP performs better than SITA-E. Our results in [4] suggest that if we use multi-layered round robin routing (tasks of type k follow a round robin policy that is independent of arrivals of all other task types), we may achieve much better performance. We use RRK to represent multi-layered round robin routing. In this section, we discuss the performance of RRK-KLP. Section 5.1 provides some simulation results for RRK-KLP. Section 5.2 provides asymptotic results for RRK-KLP.

5.1. Simulation results

Our simulation is based on a bounded Pareto distribution for the processing times. Table 5 gives the 90% confidence interval for the mean queue length with $\rho = 0.9$ and $c = 4$ for RRK-KLP. We ran 30 replications for each case, and each replication consists of 1×10^8 arrivals. The cutoff points are chosen according to Procedure 1, where we use the best cutoff points (the one providing the minimum mean queue length for Random-KLP). We also put the corresponding numerical results for Random-KLP in the table.

Table 5. Mean Queue Length of RRK-KLP and Random-KLP

Policy	α	$1/\lambda$	Mean Queue Length
RR13-13LP	1.2	823.61	(8.185, 9.348)
Random-13LP	1.2	823.61	10.680
RR9-9LP	1.5	426.57	(10.104, 28.396)
Random-9LP	1.5	426.57	31.096
RR7-7LP	1.9	33.25	(8.920, 12.497)
Random-7LP	1.9	33.25	14.940

5.2. Heavy traffic

As we did for Random-KLP in Section 4.6, we focus on one server and separate the K different types into different arrival streams for that server.

Define $Q''_k(t)$ to be the number of type k tasks at a single server at time t. We are interested in the process $\hat{Q}''^{(n)}_k(t) = n^{-1/2} Q''^{(n)}_k(nt)$. We assume that $Q''^{(n)}_k(0) = 0$, $k = 1, \ldots, K$. Under the same conditions as Theorem 2 (note that the interarrival time variance converges to $c/(\alpha_k^2 \lambda^2)$ (see (11))), we have the following result.

Theorem 4 *For a single queue operating under RRK-KLP, under the heavy traffic conditions, $\hat{Q''}_k^{(n)} \Rightarrow 0$, $k = 1, \ldots, K-1$ and*

$$\hat{Q''}_K^{(n)} \implies \mu'_K \textit{RBM}\left(d', \sum_{k=1}^{K}\left(\frac{\alpha_k \lambda s'_k}{c} + \frac{\alpha_k \lambda (\int_{z_{k-1}}^{z_k} y f_k(y) dy)^2}{c^2}\right)\right). \quad (17)$$

Proof. This follows directly from Theorem 4.1 and Theorem 4.3 of [10]. ∎

Let $\bar{Q}_{\text{RRK}} = c\bar{Q''}_{\text{RRK}}$, where $\bar{Q''}_{\text{RRK}}$ is the mean of the RBM in (17). The following theorem suggests that RRK-KLP outperforms both Random-KLP and SITA-E under high loads, for all processing time distributions with finite variance.

Theorem 5 *For a continuous processing time distribution with density function $f(x)$, where $\int_{-\infty}^{\infty} x^2 f(x) dx < \infty$, in heavy traffic, if the processing time distribution is partitioned in an identical manner for RRK-KLP, Random-KLP, and SITA-E (which in turn implies $K = c$ for RRK-KLP and Random-KLP), $\bar{Q}_{\text{RRK}} < \bar{Q}_{\text{Random}} < \bar{Q}_{\text{SITA-E}}$.*

Proof. For RRK-KLP,

$$\begin{aligned}\bar{Q}_{\text{RRK}} &= \frac{c\mu_K}{2|d|}\sum_{k=1}^{K}\left(\frac{\alpha_k\lambda}{c^2\mu_k^2} + \frac{\alpha_k\lambda s_k}{c}\right)\\ &< \frac{c\mu_K}{2|d|}\sum_{k=1}^{K}\left(\frac{\alpha_k\lambda}{c\mu_k^2} + \frac{\alpha_k\lambda s_k}{c}\right)\\ &= \bar{Q}_{\text{Random}}.\end{aligned}$$

Combining with Theorem 3, we immediately get that $\bar{Q}_{\text{RRK}} < \bar{Q}_{\text{Random}} < \bar{Q}_{\text{SITA-E}}$. ∎

6. Conclusion

We have shown that policies that are simple to implement and focus on local server scheduling yield better heavy traffic performance than polices which use FCFS for local scheduling. We have provided a detailed algorithm and guidelines for parameter choices that guarantee a performance improvement, with formulas that allow easily computed comparisons should a designer wish to choose other parameter values. Considering fairness issues, our study also shows that good mean waiting time may not lead to good mean slowdown for all type of tasks at a server. However, if we balance the load for each type of tasks, the proposed policy performs (mean waiting time and slowdown) better than policies based on FCFS local scheduling. Finally, it would be worthwhile to explore whether there is a means to directly calculate the optimal cutoff points (number and location).

References

[1] G. Ciardo, A. Riska, and E. Smirni. “equiload: a load balancing policy for clustered web servers”. *Performance Evaluation*, 46:101–124, 2001.

[2] M. E. Crovella and A. Bestavros. “self-similarity in world wide web traffic: evidence and possible causes”. *IEEE/ACM Transactions on Networking*, 5:835–846, 1997.

[3] D. G. Down and R. Wu. “scheduling distributed server systems with highly variable processing times”. In *Proceedings of the 2003 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS’03)*, Montreal, 2003.

[4] D. G. Down and R. Wu. “multi-layered round robin routing for parallel servers”. *Queueing Systems*, 53:177–188, 2006.

[5] M. Harchol-Balter, N. Bansal, B. Schroeder, and M. Agrawal. "implementation of SRPT scheduling in web servers". In *Proc. 7th Annual Workshop on Job Scheduling Strategies for Parallel Processing*, pages 11–35, 2001.

[6] M. Harchol-Balter, M. Crovella, and C. Murta. “on choosing a task assignment policy for a distributed server system”. *Journal of Parallel and Distributed Computing*, 59:204–228, 1999.

[7] N. Jaiswal. *Priority Queues*. Academic Press, 1968.

[8] V. Paxson and S. Floyd. “wide-area traffic: the failure of poisson modeling”. *IEEE/ACM Transactions on Networking*, pages 226–244, 1995.

[9] D. L. Peterson and D. B. Adams. “fractal patterns in dasd i/o traffic”. In *CMG Proceedings*, 1996.

[10] M. Reiman. “some diffusion approximations with state space collapse”. In *Lecture Notes in Controls and Information Sciences*, volume 60, pages 209–240, 1984.

[11] A. Riska, W. Sun, E. Smirni, and G. Ciardo. “adaptload: effective balancing in clustered web servers under transient load conditions”. pages 104–112, 2002.

[12] L. Schrage. “a proof of the optimality of the shortest remaining processing time discipline”. *Operations Research*, 16:687–690, 1968.

[13] L. Schrage and L. W. Miller. "the queue M/G/1 with the shortest remaining processing time discipline". *Operations Research*, 14:672–684, 1966.

[14] Q. Zhang, N. F. Mi, A. Riska, and E. Smirni. “load unbalancing to improve performance under autocorrelated traffic”. In *Proceedings of the 26th International Conference on Distributed Computing Systems (ICDCS’06)*, 2006.

[15] Q. Zhang, A. Riska, W. Sun, E. Smirni, and G. Ciardo. “load balancing for clustered web servers”. *IEEE Transactions on Parallel and Distributed Systems*, 16:219–233, 2005.