MRRK-SRPT: a Scheduling Policy for Parallel Servers

Rong Wu

Department of Computing and Software, McMaster University, Canada

Douglas G. Down

Department of Computing and Software, McMaster University, Canada

Abstract

For a system with a dispatcher and several identical servers in parallel, we propose a policy consisting of modified multi-layered round robin routing followed by shortest remaining processing time scheduling. We show that this policy has a heavy traffic limit that is identical to one in which there is a single queue (no routing) and optimal scheduling. Under light traffic, this policy is asymptotically optimal in the sense that it minimizes the first two terms of the Taylor series expansion for the waiting time. Additional simulation results show that this policy performs well under various conditions and that using policies other than shortest remaining processing time may yield reasonable performance.

Key words: heavy and light traffic approximations, parallel servers, round robin routing, shortest remaining processing time scheduling

Email addresses: wur2@mcmaster.ca (Rong Wu), downd@mcmaster.ca

(Douglas G. Down).

Preprint submitted to Elsevier Science
1 \hspace{1em} \textbf{Introduction}

With the explosive growth of the Internet and advances in computer networks, distributed systems with cluster-based architectures such as web server farms and distributed databases have become popular. Such systems typically include a front-end dispatcher and several back-end parallel servers, a configuration that we will call a loosely coupled server system. Effective and scalable scheduling of tasks for such loosely coupled server systems is crucial.

The scheduling policy for a loosely coupled server system can be divided into two components, the \textit{routing} policy, which assigns tasks to servers, and the particular \textit{local scheduling} policy, which is employed at each of the servers to which the tasks are assigned. Examples of routing policies are random routing (if \( c \) is the number of servers, a task is assigned to a particular server with probability \( 1/c \)), round robin (RR) routing (the \( i \)th arriving task is assigned to the \((i \mod c)\)th server), join-the-shortest-queue (JSQ), where each task is assigned to the server with the least number of waiting tasks, join-the-shortest-expected-workload (JSEW) routing (each task is assigned to the server with the least expected workload), and join-the-shortest-actual-workload (JSAW), where each task is assigned to the server with the least work load. Varying degrees of information about the system state are required by each routing rule: random and round robin require none, while JSQ requires the queue length information, JSEW requires the expected workload information at each server, and JSAW requires precise workload information at each server.

Examples of local scheduling policies are first-come-first-served (FCFS), in which tasks are served in the order that they arrive, shortest-remaining-processing-time (SRPT), where priority is given to the task with the least processing time remaining, and shortest-processing-time (SPT), where priority is given to the task with the
smallest processing time. We will use the format “A-B” to represent a scheduling policy, where “A” denotes the routing policy and “B” denotes the local scheduling policy. The performance measure in which we are interested is the mean waiting time in the system.

If the processing times are known on arrival, Schrage [14] showed that for a single server with an arbitrary arrival process and arbitrary processing time distribution, SRPT scheduling minimizes the completion time of the $n$th departing job. Therefore, for our model, no matter what the routing policy, it is optimal to use SRPT at each server. SRPT is rarely used in practice due to several reasons, such as the concern for pre-emption overhead. In [1], Bansal and Harchol-Balter give a proof that the pre-emption overhead under SRPT is actually lower than under practical implementations of scheduling policies such as Processor Sharing (PS). In [6], the authors implement SRPT scheduling in web servers and make a convincing case for SRPT being a viable policy in practice. The focus of this paper is to propose a routing policy followed by SRPT local scheduling to provide good system performance for various system loads and task size variabilities. We assume that the task size is known upon arrival. In [4], we identified the optimal policy ($c$-SRPT) for a tightly coupled server system, i.e. one in which there is a single queue for all of the servers. The $c$-SRPT policy gives preemptive (resume) priority to the first $c$ tasks with the least remaining processing times (if there are less than $c$ tasks present, every task receives service). We showed that this policy minimizes the completion time of the $n$th departing task from the system. The reason to explore $c$-SRPT is that it provides a lower bound for the performance of other policies, such as those policies for a loosely coupled server system. It also gives us insight into how to construct a policy for loosely coupled server systems. In [4], we also proposed multi-layered round robin routing followed by SRPT local scheduling and proved that the diffusion limit of the proposed policy converges to that of the lower bound ($c$-SRPT)
for a discrete task processing time distribution. Under light traffic, RR-SRPT was shown to be asymptotically optimal in the sense that it minimizes the first two terms of the Taylor series expansion for the waiting time. In this paper, we exploit those insights to design a policy which provides good performance under general system loads. Our work in this paper is thus a natural extension of that contained in [4].

The organization of this paper is as follows. Section 2 describes some related work. In Section 3, we describe the proposed policy. We show that the policy has a heavy traffic limit that is identical to c-SRPT for a discrete task size distribution. For general processing time distributions under light traffic conditions, the proposed policy is asymptotically optimal in the sense that it minimizes the first two terms of the Taylor series expansion for the waiting time. Section 4 gives some simulation results for discrete processing time distributions and Bounded Pareto processing time distributions. We compare our policy with JSAW-FCFS, SITA-E, and other SRPT related policies. In Section 5, we present possible alternatives to the proposed policy. Section 6 provides some concluding remarks.

2 Related Work

A number of policies have been proposed and studied for loosely coupled server systems. In this section, we give a brief review of this work. Until indicated otherwise, the references discussed assume that processing times are not known until a task is completed. If the server information (e.g. queue length) is not considered, Jean-Marie and Liu [8] showed that round robin routing yields smaller waiting times than the Bernoulli policy (Random) routing in the sense of increasing convex ordering. The local scheduling policy used here is FCFS. Ephremides, Varaiya,
and Walrand in [5] have shown that round robin routing followed by FCFS local scheduling minimizes the sum of the expected completion times for the first \( n \) tasks (for all \( n \)) if the task sizes follow an exponential distribution. Liu and Towsley in [9] showed that round robin routing followed by FCFS local scheduling minimizes (in the sense of increasing convex ordering) response times and queue lengths when processing times have increasing hazard rate.

If the dispatcher has global state information, and if all underlying distributions are exponential, Winston in [18] showed that JSQ routing followed by FCFS local scheduling minimizes the mean waiting time in the system. Ephremides, Varaiya, and Walrand in [5] showed that JSQ-FCFS also minimizes the expected total time for the completion of all tasks arriving by some fixed time. Weber in [16] proved that JSEW routing followed by FCFS local scheduling stochastically maximizes the number of completed tasks for task size distributions with non-decreasing hazard rate. As the variability of the task size distribution grows, Whitt in [17] argued that the JSEW-FCFS discipline does not minimize the mean waiting time by providing some counter examples.

We now turn to the situation when processing times are known on arrival. In recent years, Harchol-Balter, Crovella, and Murta [7] have proposed a routing policy called SITA-E (Size Interval Task Assignment with Equal Load) in which tasks within a particular size range are assigned to the same server, with the size ranges chosen to equalize the loads across servers. The task size variance at each server is significantly smaller than the variance for the processing times of tasks arriving to the system. They showed that SITA-E routing significantly outperforms JSAW routing (with FCFS local scheduling) under high task size variance. Similar policies such as EquiLoad and AdaptLoad are suggested in the work of Ciardo, Riska, and Smirni [2] and Riska, Sun, Smirni and Ciardo [13]. An important issue for SITA-E, EquiLoad, and AdaptLoad is that one must know the processing time distribution
to choose appropriate policy parameters. In [15], Ungureanu and her co-authors suggested the Class Dependent Assignment (CDA) routing policy, which divides arrivals into long and short tasks. Short tasks are assigned in a round robin manner and long tasks are kept at the dispatcher until a server becomes idle. Their simulation results showed that CDA-FCFS performs similarly to JSAW-FCFS (CDA is worse than JSAW if the communication overhead is ignored). In this paper, we also do simulations to show that the performance of the proposed policy is better than JSAW-FCFS.

In [4], we gave a discussion of RR$^K$-SRPT for a discrete distribution and RR-SRPT for general distributions. RR$^K$-SRPT uses multi-layered round robin as the routing policy and SRPT as the local scheduling policy. Suppose there are $K$ types of tasks. To implement a multi-layered round robin routing policy, the dispatcher employs an independent round robin policy for each type of task. If we approximate the system using a formal diffusion approximation (i.e. under heavy traffic conditions), we showed that the diffusion limits of the queue lengths of RR$^K$-SRPT converge to those of $c$-SRPT for a discrete task processing time distribution (which is what we mean by asymptotically optimal in heavy traffic). Under light traffic, the performance of RR$^K$-SRPT is worse than RR-SRPT. In this paper, we design a policy which takes advantage of the asymptotic performance results for RR$^K$-SRPT and RR-SRPT to provide good system performance under general system loads.

3 Proposed Policy

In [4], we have shown that RR$^K$-SRPT is asymptotically optimal under heavy traffic conditions for a discrete task processing distribution and RR-SRPT is asymptotically optimal under light traffic conditions. If we relax the assumption of the
system operating in either heavy or light traffic, this means that each server spends
some non-zero proportion of time idle and non-zero proportion of time busy, so at
moderate loads, it is not clear what a reasonable policy would be. This is because in
[4], we have shown that RR$K$-SRPT is not optimal in light traffic. The probability
of sending two consecutive arrivals to the same server is not zero, which has the
potential of creating longer task waiting times than RR-SRPT. In this section, we
design a policy which is based on RR$K$-SRPT but addresses its drawbacks under
light traffic. The policy allows the dispatcher to send new arrivals to an idle server
(if such an idle server exists). In light traffic, there are always idle servers, therefore
the policy does not send two consecutive tasks to the same server. Hence it has the
same light traffic behaviour as RR-SRPT. The following is a formal description for
the proposed policy, which uses modified multi-layered round robin routing fol-
lowed by SRPT local scheduling. We will call this policy MRR$K$-SRPT.
Let $\ell_{i,j}(t)$ represent the most recent time before $t$ that server $j$ has received a type
$i$ task, $c$ represent the number of servers in the system, $a_n$ represent the arrival
time of task $n$, and $I(t)$ represent the set of idle servers in the system at time $t$.
MRR$K$-SRPT sends task $n$ of type $i$ to server $k$, where

$$
k = \begin{cases} 
\arg\min_{j \in I(a_n)} \{\ell_{i,j}(t)\}, & \text{if } I(a_n) \neq \emptyset \\
\arg\min_{1 \leq j \leq c} \{\ell_{i,j}(t)\}, & \text{otherwise.}
\end{cases}
$$

At each server, SRPT is used as the local scheduling policy. Notice that compared
to either RR$K$-SRPT or RR-SRPT, MRR$K$-SRPT has some communication over-
head. However, such communication overhead is very low since each server only
needs to notify the dispatcher when it becomes idle.
Now we consider the performance of MRR$K$-SRPT. In Section 3.1, we show that
the diffusion limit of the queue lengths of MRR$K$-SRPT converges to that of $c$-
SRPT. In Section 3.2, we show that MRR\(K\)-SRPT is asymptotically optimal in light traffic, in the sense that it minimizes the first two terms of the Taylor series expansion for the waiting time.

3.1 Heavy Traffic

We use the same procedure as for RR\(K\)-SRPT in [4] to inspect the performance of MRR\(K\)-SRPT in heavy traffic. We will compare the performance of \(c\)-SRPT and MRR\(K\)-SRPT.

We assume that arrivals occur according to a Poisson process of rate \(\lambda\). A task has processing times that are chosen from a discrete distribution with finite support. Tasks with processing times \(x_k\) are labelled as type \(k\) tasks. The processing time \(X\) for a generic task is given by

\[
P[X = x_k] = \alpha_k, \quad k = 1, \ldots, K, \tag{1}
\]

where \(\alpha_k > 0\), \(\sum_{k=1}^{K} \alpha_k = 1\), and the possible processing times are ordered \(x_1 < x_2 < \cdots < x_K\). If \(K = 1\), the problem is trivial, so from this point we assume \(K > 1\).

For the entire network, the interarrival and processing time sequences for each type \(k\) are denoted by \(\{u_i^k, i \geq 1\}\) and \(\{v_i^k, i \geq 1\}\), respectively. The mean and variance of a generic element of each sequence are

\[
\begin{align*}
\lambda_k &= (E[u_i^k])^{-1} = \alpha_k \lambda, \\
\mu_k &= (E[v_i^k])^{-1} = 1/x_k, \\
s_k &= Var(v_i^k) = 0.
\end{align*}
\]

We assume that we have a sequence of systems indexed by \((n)\) where, as \(n \to \infty\),
\[ \lambda_k(n) \rightarrow \lambda_k, \]
\[ a_k(n) \rightarrow a_k, \]
\[ \mu_k(n) \rightarrow \mu_k, \]
\[ s_k(n) = 0. \]

If \( Q_k(t) \) is the number of type \( k \) tasks in the system at time \( t \), we form the scaled sequence
\[ \hat{Q}_k^{(n)}(t) = n^{-1/2} Q_k^{(n)}(nt), \]
where we assume that
\[ Q_k^{(n)}(0) = 0, \quad 1 \leq k \leq K - 1. \]

The heavy traffic conditions are as follows. Let \( \rho_i(n) \) be the load due to tasks of type \( i \), i.e. \( \rho_i(n) = \lambda_i(n)/c\mu_i(n), 1 \leq i \leq K \). We assume \( \rho_i(n) < 1, 1 \leq i \leq K \), and
\[ d_i(n) \rightarrow -\infty, \quad \text{as} \ n \rightarrow \infty, 1 \leq i \leq K - 1, \]
\[ d_K(n) \rightarrow d_K, \quad \text{as} \ n \rightarrow \infty, -\infty < d_k < 0, \]
\[ \rho_i(n) \rightarrow \rho_i, \quad \text{as} \ n \rightarrow \infty, 1 \leq i \leq K - 1. \]

Let \( \text{RBM}(a, b) \) denote a reflected Brownian motion with drift \( a \) and variance \( b \). Weak convergence in the metric space consisting of all right continuous functions with left limits will be denoted by \( \Rightarrow \). The following theorem gives the diffusion limit for a loosely coupled server system operating under \( \text{MRRK-SRPT} \).

**Theorem 1** For \( \text{MRRK-SRPT} \),
\[ \hat{Q}_k^{(n)} \Rightarrow 0, \quad 1 \leq k \leq K - 1, \]
\[ \hat{Q}_K^{(n)} \Rightarrow \frac{c}{x_K} \text{RBM} \left( d_K, \sum_{k=1}^{K} \frac{\alpha_k \lambda x_k^2}{c^2} \right). \]

**Proof.** If we focus attention on a single server, denoting by \( \lambda_k^o(n), a_k^o(n), \mu_k^o(n), \) and \( s_k^o(n) \) the associated arrival rate, interarrival time variance, processing rate, and
processing time variance for type $k$ tasks, we have $\mu^o_k(n) = \mu_k(n)$ and $s^o_k(n) = s_k(n)$. Now, if $\tilde{I}^o(t)$ is the cumulative idle time of a particular server at time $t$, and as $\tilde{I}^o(t) = n^{-1/2} I^o(n)(t) \Rightarrow 0$, we have

$$\lambda^o_k(n) \longrightarrow \lambda_k/c$$
$$a^o_k(n) \longrightarrow ca_k.$$ 

Now, these parameters are the same as for RR$K$-SRPT, so by Theorem 3.1 of [4] and the fact that there are $c$ identical servers, the result follows.

We say that a policy is asymptotically optimal in heavy traffic if it has the same diffusion limit as $c$-SRPT. (Note that this means that if we were to use the diffusion approximation to approximate the mean waiting time, the approximations for $c$-SRPT and MRR$K$-SRPT would be identical.)

**Theorem 2** MRR$K$-SRPT is asymptotically optimal in heavy traffic.

**Proof.** That RR$K$-SRPT has the same diffusion approximation as $c$-SRPT follows from Theorem 3.2 of [4] and Theorem 1.

3.2 Light Traffic

In [4], we showed that in light traffic, RR-SRPT is asymptotically optimal in the sense that it minimizes the first two terms of the Taylor series expansion for the waiting time. Also, RR$K$-SRPT is not optimal due to the non-zero probability that two consecutive tasks could be sent to the same server. As we did for RR-SRPT, we can use Reiman-Simon theory ([11,12]) to show that MRR$K$-SRPT is asymptotically optimal in light traffic.
First, we introduce Reiman-Simon light traffic theory. Once again, we assume an arrival process that is Poisson with rate \( \lambda \). Let \( \{ v_n, n = 0, \pm 1, \pm 2 \} \) and \( \{ w_n, n = 0, \pm 1, \pm 2 \} \) represent processing time and waiting time sequences, respectively. We are interested in \( P_\lambda[w_\infty > x] \ (x \geq 0) \), the complementary cumulative distribution function of the stationary waiting time. To force the system to reach steady state, we let the system operate from time \( t = -\infty \). Hence, \( w_0 =_{st} w_\infty \), where \( =_{st} \) means equality in distribution. Define two events, \( \emptyset \) and \( \{ t \} \), where \( \emptyset \) denotes the event that no task arrives in the interval \(( -\infty, \infty )\) except for a tagged task which arrives at time \( t = 0 \) (with associated processing time \( v_0 \)), and \( \{ t \} \) denotes the event where in addition to the tagged task, there is exactly one more arrival at time \( t \) (with processing time \( v \)). An expression for \( P_\lambda[w_\infty > x] \) is developed by conditioning on these two events. Define

\[
\psi = 1[w_0 > x], \\
\phi(\lambda) = \int \psi dP_\lambda, \\
\hat{\psi}(\emptyset) = E_\lambda[\psi|\emptyset], \\
\hat{\psi}(\{ t \}) = E_\lambda[\psi|\{ t \}], \ t \in \mathbb{R}.
\]

Proposition 3 provides a means to estimate \( P_\lambda[w_\infty > x] \) as \( \lambda \to 0 \). It can be found as Theorem 2 in [11].

**Proposition 3** If there exists \( \theta^* > 0 \) such that \( E[e^{\theta v}] < \infty \) for \( \theta < \theta^* \), then

\[
\lim_{\lambda \to 0^+} P_\lambda[w_\infty > x] = \hat{\phi}(\emptyset)
\]

and

\[
\frac{d}{d\lambda} \phi(0^+) = \lim_{\lambda \to 0^+} \frac{d}{d\lambda} \phi(\lambda) = \int_{\mathbb{R}} \left( \hat{\psi}(\{ t \}) - \hat{\psi}(\emptyset) \right) dt.
\]

Proposition 3 gives us a means to estimate the derivative of the waiting time distribution in light traffic. The waiting time distribution itself is then estimated by
using the Taylor series expansion around $\lambda = 0$. If $F(x)$ is a distribution, let $\bar{F}(x) = 1 - F(x)$ ($x \geq 0$) represent its complement. Theorem 4 provides results for the performance of MRRK-SRPT in light traffic.

**Theorem 4** For a loosely coupled server system with Poisson arrivals of rate $\lambda$ and finite exponential moment for the processing time distribution $F$, let $w_{\infty}^{\text{MRRK}}$ represent the waiting time in steady-state under MRRK-SRPT.

MRRK-SRPT is asymptotically optimal in light traffic in the sense that the first two terms of the Taylor expansion of $P_{\lambda}[w_{\infty} > x]$ ($x \geq 0$) are minimized.

**Proof.** On the event $\emptyset$,

$$\hat{\psi}(\emptyset) = P[v > x] = \bar{F}(x),$$

as $w_0 = v_0$. Under MRRK routing, on the event $\{t\}$, the tasks arriving at times 0 and $t$ cannot be assigned to the same queue. It immediately follows that

$$w_0 = v_0,$$
$$\hat{\psi}(\{t\}) = P[v_0 > x] = \bar{F}(x),$$
$$\frac{d}{d\lambda} \phi(0_+) = \int_{-\infty}^{\infty} \left( \hat{\psi}(\{t\}) - \hat{\psi}(\emptyset) \right) dt = 0.$$

Proposition 3 then implies that if we take a Taylor series expansion of $P_{\lambda}[w_{\infty} > x]$,

$$P_{\lambda}[w_{\infty} > x] = \bar{F}(x) + o(\lambda), x \geq 0. \tag{2}$$

A task’s waiting time cannot be less than its processing time. Therefore, the value of $P_{\lambda}[w_{\infty} > x]$ is minimized under MRRK routing, if we ignore the terms of order $\lambda^2$ and higher in (2).

We have shown that MRRK-SRPT is asymptotically optimal in both heavy and light traffic. In the next section, we use simulation to show that MRRK-SRPT
provides good performance under general system loads.

4 Simulation Results

We perform our simulations for both two-point and Bounded Pareto task processing time distributions. To get a better idea for the performance of $MRRK^*$-SRPT, we also do simulations for $RRK^*$-SRPT, RR-SRPT, and $c$-SRPT. For the Bounded Pareto distribution, we do simulations for SITA-E and JSAW-FCFS in addition to the aforementioned SRPT related policies. The number of servers in the system is chosen to be 8. Tasks arrive according to a Poisson process with rate $\lambda$, and this rate is adjusted to achieve the desired loads ($\rho$). Table 3 gives the 90% Confidence Interval for expected queue length with two-point processing time distributions under different system loads and task size variances. We ran 30 replications for each policy. Each replication includes $1.0 \times 10^6$ arrivals. Table 1 gives the parameters for the different two-point processing time distributions used. The processing times of type 1 and type 2 tasks are given by $x_1$ and $x_2$, respectively. The proportion of type 1 tasks is $\alpha_1$. The proportion of type 2 tasks is $1 - \alpha_1$. Table 2 gives the remaining parameters. There is a case number following each policy in Table 1. Each case number indicates the two-point processing time distribution used. For instance, RR2-SRPT (1) means we run RR2-SRPT on a two-point task processing time distribution with $x_1 = 1$ and $x_2 = 4$, and where the proportion of type 1 tasks is 0.333333.

Tables 5 and 6 give the 90% Confidence Interval for expected queue length with Bounded Pareto distributions under different system loads and task size variances. We ran 91 replications for JSAW-FCFS, and 30 replications for other policies. Each replication includes $1.0 \times 10^6$ arrivals. The reason we also use the Bounded Pareto
Table 1

Type of Tasks in Simulation

<table>
<thead>
<tr>
<th>Case</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$\alpha_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>1</td>
<td>4</td>
<td>0.333333</td>
</tr>
<tr>
<td>(2)</td>
<td>1</td>
<td>100</td>
<td>0.979798</td>
</tr>
<tr>
<td>(3)</td>
<td>1</td>
<td>10000</td>
<td>0.99979998</td>
</tr>
</tbody>
</table>

Table 2

Parameters for Two-Point Processing Time Distribution

<table>
<thead>
<tr>
<th>$\rho$</th>
<th>$E[X]$</th>
<th>$1/\lambda$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>3</td>
<td>0.750</td>
</tr>
<tr>
<td>0.7</td>
<td>3</td>
<td>0.536</td>
</tr>
<tr>
<td>0.85</td>
<td>3</td>
<td>0.441</td>
</tr>
<tr>
<td>0.95</td>
<td>3</td>
<td>0.395</td>
</tr>
</tbody>
</table>

distribution is that distributions of this form appear to be a feature in many aspects of computing systems [3,7]. As such, this distribution appears frequently in the literature in this area. We examined two particular instances of the Bounded Pareto distribution. The Bounded Pareto distribution with density $f$ is as follows:

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

where $0 < \alpha < 2$ is the exponent of the power law, $k$ is the smallest possible observation, and $p$ is the largest possible observation. We let $k = 512$ and $p = 10^{10}$. Table 4 gives the parameters used in simulation. In Tables 5 and 6, MRR2-SRPT(1)
Table 3

<table>
<thead>
<tr>
<th>Policy</th>
<th>(\rho = 0.5)</th>
<th>(\rho = 0.7)</th>
<th>(\rho = 0.85)</th>
<th>(\rho = 0.95)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MRR2-SRPT (1)</td>
<td>(4.059, 4.062)</td>
<td>(5.998, 6.004)</td>
<td>(8.549, 8.565)</td>
<td>(15.312, 15.481)</td>
</tr>
<tr>
<td>8-SRPT (1)</td>
<td>(4.046, 4.048)</td>
<td>(5.924, 5.929)</td>
<td>(8.361, 8.376)</td>
<td>(14.916, 15.097)</td>
</tr>
<tr>
<td>8-SRPT (3)</td>
<td>(3.957, 4.087)</td>
<td>(5.740, 5.931)</td>
<td>(7.6333, 8.119)</td>
<td>(9.635, 10.762)</td>
</tr>
</tbody>
</table>

and RR2-SRPT (1) mean that the policies divide the arrivals into two types of tasks, where the probability a task is of type 1 is 0.85. In other words, type 1 tasks have processing times in \([512, 2488]\) and type 2 tasks have processing times in \([2488, 10^{10}]\). MRR2-SRPT(2) and RR2-SRPT (2) also divide the arrivals into two types of tasks, but such that the probability a task is of type 1 is 0.95. In other words, type 1 tasks have processing times in \([512, 6220]\) and type 2 tasks have processing
times in $[6220, 10^{10}]$. MRR8-SRPT and RR8-SRPT divide arrivals into eight types of tasks, where the eight intervals are chosen according to the SITA-E algorithm.

Table 4

Parameters for Bounded Pareto Distributions

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$\rho$</th>
<th>$E[X]$</th>
<th>$1/\lambda$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2</td>
<td>0.5</td>
<td>2965</td>
<td>741.25</td>
</tr>
<tr>
<td>1.2</td>
<td>0.7</td>
<td>2965</td>
<td>390.13</td>
</tr>
<tr>
<td>1.2</td>
<td>0.85</td>
<td>2965</td>
<td>436.03</td>
</tr>
<tr>
<td>1.2</td>
<td>0.95</td>
<td>2965</td>
<td>529.46</td>
</tr>
<tr>
<td>1.5</td>
<td>0.5</td>
<td>1535.65</td>
<td>383.91</td>
</tr>
<tr>
<td>1.5</td>
<td>0.7</td>
<td>1535.65</td>
<td>274.29</td>
</tr>
<tr>
<td>1.5</td>
<td>0.85</td>
<td>1535.65</td>
<td>225.88</td>
</tr>
<tr>
<td>1.5</td>
<td>0.95</td>
<td>1535.65</td>
<td>202.11</td>
</tr>
</tbody>
</table>

Our simulation results show that MRR$\bar{K}$-SRPT performs well under various conditions, in particular when the processing time has high variance (i.e. $\alpha = 1.2$ for the Bounded Pareto distribution). Its performance is very close to $c$-SRPT for discrete distributions. SITA-E and JSAW-FCFS perform worse than the SRPT based policies. If the task variance and system load are high, JSAW-FCFS performs worst, which is not surprising. However, this does suggest that under heavy load, the role of the local scheduling policy is more important than that of the routing policy. Therefore, if the system load and task size variance are high, if we employ optimal local scheduling policies, we can use relatively naive, (near) state independent routing. Even in the face of arguments that SRPT is implementable [1,6], it may
Table 5

90% CI of Expected Queue Length with Bounded Pareto Distribution ($\alpha = 1.2$)

<table>
<thead>
<tr>
<th>Policy</th>
<th>$\rho = 0.5$</th>
<th>$\rho = 0.7$</th>
<th>$\rho = 0.85$</th>
<th>$\rho = 0.95$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MRR2-SRPT (1)</td>
<td>(3.978, 4.129)</td>
<td>(5.905, 6.104)</td>
<td>(8.419, 8.835)</td>
<td>(11.439, 12.135)</td>
</tr>
<tr>
<td>MRR2-SRPT (2)</td>
<td>(3.996, 4.175)</td>
<td>(5.822, 6.008)</td>
<td>(8.261, 8.602)</td>
<td>(10.773, 11.572)</td>
</tr>
<tr>
<td>RR2-SRPT (2)</td>
<td>(4.879, 5.107)</td>
<td>(7.964, 8.368)</td>
<td>(11.487, 11.952)</td>
<td>(15.171, 15.921)</td>
</tr>
<tr>
<td>MRR8-SRPT</td>
<td>(4.029, 4.195)</td>
<td>(5.991, 6.293)</td>
<td>(8.195, 8.590)</td>
<td>(10.954, 11.995)</td>
</tr>
<tr>
<td>JSW-FCFS</td>
<td>(4.249, 4.690)</td>
<td>(26.759, 44.188)</td>
<td>(418.054, 1196.224)</td>
<td>(2430.441, 4327.356)</td>
</tr>
<tr>
<td>8-SRPT</td>
<td>(3.893, 4.029)</td>
<td>(5.541, 5.736)</td>
<td>(7.051, 7.315)</td>
<td>(8.696, 9.193)</td>
</tr>
</tbody>
</table>

be useful to examine other local scheduling policies. In the next section we examine whether other options may be employed that rely on more common policies, in particular options that are based on FCFS.

5 FCFS-related Local Scheduling

SRPT can be approximated by a priority queue, where high priority tasks have pre-emptive priority over low priority tasks. For discrete distributions, tasks with small task processing times have higher priority. This means that type $i$ tasks have higher
Table 6

90% CI of Expected Queue Length with Bounded Pareto Distribution ($\alpha = 1.5$)

<table>
<thead>
<tr>
<th>Policy</th>
<th>$\rho = 0.5$</th>
<th>$\rho = 0.7$</th>
<th>$\rho = 0.85$</th>
<th>$\rho = 0.95$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SITA-E</td>
<td>(7.358, 9.962)</td>
<td>(15.726, 18.892)</td>
<td>(31.637, 84.369)</td>
<td>(104.434, 147.852)</td>
</tr>
<tr>
<td>JSAW-FCFS</td>
<td>(4.070, 4.092)</td>
<td>(7.225, 7.459)</td>
<td>(33.025, 44.772)</td>
<td>(708.748, 2498.472)</td>
</tr>
<tr>
<td>8-SRPT</td>
<td>(4.012, 4.036)</td>
<td>(5.833, 5.904)</td>
<td>(7.700, 7.871)</td>
<td>(10.152, 10.419)</td>
</tr>
</tbody>
</table>

priority than type $j$ tasks if $x_i < x_j$, which is very close to SRPT (if one looks at the priorities of all tasks in the system, these policies differ by at most one task). For a continuous task processing time distribution with density function $f(x)$, we could partition the support of $f$ into $K$ intervals such that a type $k$ task is one whose processing time lies in the $k$th interval. As in the discrete case, type $i$ tasks get higher priority than type $j$ tasks if $i < j$. A server then uses FCFS within tasks of the same type. We use MRR$K$-MultiPriorityFCFS to represent such a policy. Comparing to MRR$K$-SRPT, the main difference is that each server does not need to keep the remaining processing time of each task, rather only needs the total number of each
type of task that is waiting.

Notice that MRR$K$-MultiPriorityFCFS has the same light and heavy traffic properties as MRR$K$-SRPT. (In other words, for a discrete distribution, Theorems 2 and 4 hold if we replace MRR$K$-SRPT by MRR$K$-MultiPriorityFCFS.) This is additional justification that it is a reasonable policy to consider. One would expect that at intermediate values of load, the performance of MRR$K$-MultiPriorityFCFS would be somewhat worse than MRR$K$-SRPT. We suggest through simulation that this gap may not be significant. Tables 7 and 8 give the 90% Confidence Interval for the expected queue length for Bounded Pareto distributions with different system loads and task size variabilities. The parameters used in the simulation are the same as Table 4. We ran 30 replications for each policy. Each replication includes $1.0 \times 10^6$ arrivals. MRR2-MultiPriorityFCFS (1) and (2) mean we divide the tasks into two types, the intervals being the same as for MRR2-SRPT (1) and (2). MRR8-MultiPriorityFCFS means there are 8 types of tasks. The intervals are chosen according to the SITA-E algorithm.

Table 7
90% CI of Expected Queue Length with Bounded Pareto Distribution ($\alpha = 1.2$)

<table>
<thead>
<tr>
<th>Policy</th>
<th>$\rho = 0.5$</th>
<th>$\rho = 0.7$</th>
<th>$\rho = 0.85$</th>
<th>$\rho = 0.95$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MRR2-MultiPriorityFCFS (1)</td>
<td>(3.963, 4.086)</td>
<td>(5.930, 6.122)</td>
<td>(8.496, 8.994)</td>
<td>(11.078, 11.854)</td>
</tr>
<tr>
<td>MRR2-MultiPriorityFCFS (2)</td>
<td>(3.935, 4.057)</td>
<td>(5.914, 6.230)</td>
<td>(8.304, 8.728)</td>
<td>(11.325, 12.048)</td>
</tr>
</tbody>
</table>

The results show that the performance of MRR$K$-MultiPriorityFCFS is nearly the same as MRR$K$-SRPT (compare the values in Table 7 with the corresponding values in Table 5, for example). Our results suggest that partitioning tasks according
to size and using preemption in the local scheduling policy are the key factors. As long as something “reasonable” is done that satisfies these constraints, near optimal performance can be achieved.

6 Conclusion

We proposed the policy $\text{MRR}K\text{-SRPT}$ for loosely coupled server systems and showed that $\text{MRR}K\text{-SRPT}$ is asymptotically optimal for a discrete task size distribution in heavy traffic and for general task size distributions in light traffic. Our simulation results also show that $\text{MRR}K\text{-SRPT}$ performs well under various conditions, especially for high processing time variances and system loads. Although there is communication overhead for $\text{MRR}K\text{-SRPT}$, such communication overhead is not significant. The $\text{MRR}K\text{-SRPT}$ policy appears to outperform all other policies proposed in the literature. We feel that the key is preemption in the local scheduling policy, which if allowed, allows the routing decision to be made simpler. Of course, if preemption is not allowed, our observations are not valid. This conclusion is amplified by the observation that $\text{MRR}K\text{-MultiPriorityFCFS}$ has very similar performance to $\text{MRR}K\text{-SRPT}$.
References


