Bounding Mean Slowdown in Multiserver Systems

Ziv Scully*
Carnegie Mellon University
zscully@cs.cmu.edu

ABSTRACT

Recent progress in queueing theory has made it possible to analyze the mean response time of multiserver queueing systems under advanced scheduling policies. However, this progress has so far been limited to the metric of mean response time. In practice, there are a wide variety of other metrics that can be more important. One such metric is mean *slowdown*, which is the average ratio between a job's response time and its size. While it is known that the "RS" policy minimizes mean slowdown in the single-server M/G/1, the problem is open for multiserver systems, including the M/G/k and load-balancing systems.

In this work, we analyze RS scheduling in the M/G/k and an immediate dispatch model with "M/G" arrivals, pairing RS with a carefully designed class of dispatching policies in the latter case. We prove a *universal additive bound* on the gap between multiserver mean slowdown and RS's mean slowdown in an M/G/1 with the same total service capacity. The bound depends on the number of servers but not the load or job size distribution. The bound implies that RS is a constant-factor approximation for mean slowdown.

Our mean slowdown bound also implies heavy-traffic optimality if the job size distribution has (roughly speaking) finite third moment. Proving this result requires analyzing the heavy-traffic behavior of RS's mean slowdown in the M/G/1, a result of independent interest.

1. INTRODUCTION

There is a vast literature in queueing theory on designing scheduling and dispatching policies to lower mean response time (a.k.a. sojourn time, latency) in queueing systems. In particular, recent years have seen a flurry of results bounding the performance of policies like SRPT (Shortest Remaining Processing Time) in multiserver systems. For instance, Grosof et al. [2] show that if the job size distribution has (roughly speaking) finite variance, then SRPT minimizes mean response time in the M/G/k in heavy traffic (Section 2). Similarly, in load-balancing systems with "M/G" arrivals, Grosof et al. [3] again obtain a heavy-traffic optimality result by pairing SRPT with a carefully designed dispatching policy that obeys a "guardrail" constraint (Section 2.3). Other results apply to systems with unknown job sizes [6, 7]. The overall theme of these results is that *policies that minimize mean response time in the M/G/1 also perform well in multiserver systems.*

Unfortunately, this recent progress in bounding multiserver performance of scheduling policies has so far been limited to the metric of mean response time. A metric that can be more important in practice is mean *slowdown*, namely the mean ratio between a job's response time and its size (a.k.a service requirement). The motivation for slowdown is that, for example, a user would be more upset if a 1-second job were delayed for 10 seconds than if a 100-second job were delayed for 10 seconds.

The policy that minimizes mean slowdown in the M/G/1 is the "RS" policy [4] (Section 2.2). We ask: *does RS have low mean slowdown in multiserver systems?* This question is largely open. To the best of our knowledge, the most related prior work is that of Hyytiä et al. [4]. They design a new heuristic dispatching policy for minimizing mean slowdown in an immediate-dispatch model that uses RS scheduling at each server. However, the only theoretical guarantee on their heuristic is that it outperforms random dispatching.

1.1 Contributions

We give the first theoretical mean slowdown bounds for RS in multiserver queueing systems. We study the M/G/k and an immediate-dispatch model we call the *M/G/k/dispatch*, pairing RS with a carefully designed dispatching policy in the latter case. We compare the performance of each multiserver system to an M/G/1 with the same total service capacity (Section 2), proving the following results for both systems.

- (Theorem 3.1) Under RS in both multiserver systems, the gap between multiserver and single-server mean slowdown is upper bounded by a *constant* which depends only on the number of servers.
- (Corollary 3.2) RS is a constant-factor approximation for mean slowdown in both multiserver systems.
- (Theorem 3.3) If the job size distribution S satisfies $\mathbf{E}[S^3] < \infty$, then RS is heavy-traffic optimal in both multiserver systems.

2. PRELIMINARIES

We consider two different multiserver systems in this work. Both systems have Poisson arrivals at rate λ, a general job size distribution S, and k servers.

- The first system is the M/G/k queue. This is a central-queue model: there is only one queue, and jobs can be sent between queue and any of the servers at any time.
- The second system is a load-balancing system with k servers and an "M/G" arrival process, which we call the *M/G/k/dispatch*. This is an immediate-dispatch

*Research supported by NSF grant nos. CMMI-1938909 and CSR-1763701.

model: each server has its own queue, and every job must be irrevocably assigned to a queue upon arrival.

In both settings, we use the convention that each server runs at speed $1/k$, meaning a job of size s takes s/k time to serve. This means the total service capacity of the system is 1 for all values of k.

We define load as $\rho = \lambda \mathbf{E}[S]$ and assume $\rho < 1$ for stability. We can interpret ρ as the average fraction of servers that are busy. The *heavy-traffic limit*, written as $\rho \to 1$, is the limit as the arrival rate λ approaches $1/\mathbf{E}[S]$ while the size distribution S remains fixed.

We assume a preempt-resume model in which jobs may be preempted at any time with no delay or loss of work. Under this assumption, an M/G/1 is strictly more powerful than the M/G/k or M/G/k/dispatch for any $k \geq 2$. This is because the M/G/1 could simulate either multiserver system by sharing the processor between up to k jobs. Because the M/G/1 is more powerful than the multiserver systems, our results analyze multiserver performance relative to the M/G/1. In particular, optimal multiserver performance cannot be better than optimal M/G/1 performance.

2.1 Objective: Mean Slowdown

In this work, we study the problem of minimizing *mean slowdown*. A single job's *slowdown* is the ratio between its response time and its size, where *response time* is the amount of time between the job's arrival and departure. A job with response time t and size s has slowdown $z = t/s$. We write Z for the equilibrium distribution of jobs' slowdowns. Our goal is to minimize $\mathbf{E}[Z]$.

Our results compare mean slowdown in multiserver systems to mean slowdown in an M/G/1, with both systems using versions of RS scheduling (Sections 2.2 and 2.3). When discussing slowdown in a generic system, we write simply Z We write Z_1 to refer to the mean slowdown in the M/G/1, and we similarly write Z_k for the multiserver systems. It will always be clear from context whether we are discussing the M/G/k or the M/G/k/dispatch.

2.2 Scheduling Policy: RS

We assume that the scheduler has perfect knowledge of job sizes and can perfectly keep track of how long each job has been served. We therefore describe the state of each job as a tuple (s, x), where

- s is the job's original size; and
- x is the job's remaining size, namely s minus the amount of service it has received so far.

A job of size s arrives in state (s, s) and departs once it reaches state $(s, 0)$.

All systems we consider use a version of the "RS" scheduling policy. The name is a mnemonic for "remaining size times (original) size". RS works as follows: it assigns each job a numeric *rank* based on the job's state then prioritizes jobs by rank, where *lower rank means better priority*. Specifically, the rank of a job in state (s, x), meaning size s and remaining size x, is

$$\text{rank}(s, x) = sx.$$

We may break ties between jobs of the same rank arbitrarily.

Here is how RS works in the systems we study.

- In single-server systems, such as the M/G/1, RS always serves the job of lowest rank.
- In the M/G/k, RS serves the k jobs with the k lowest ranks. If there are fewer than k jobs, then RS serves all of them, leaving some servers idle.
- In the M/G/k/dispatch, we use RS at each server, meaning the system serves the job of lowest rank at each server. It may be that one server is left idle while another server has multiple jobs.

It is known that RS minimizes mean slowdown in the M/G/1 queue [4, Corollary 3]. Intuitively, one can think of RS as a version of the famous "$c\mu$ rule": a job's holding cost is $c = 1/s$, and its completion rate is $\mu = 1/x$, so prioritizing by highest $c\mu$ corresponds to prioritizing by lowest rank sx.

2.3 Dispatching Policy: Obey Guardrails

The only aspect of our system model that we have yet to discuss is the *dispatching policy*, which determines how we assign jobs to servers in the M/G/k/dispatch. Our results hold not for a single dispatching policy but rather for all dispatching policies satisfying a certain property, which we describe below. This property was originally introduced by Grosof et al. [3] when studying dispatching to SRPT servers.

To achieve good performance in a load-balancing system, we want to avoid situations where one server is idle while another has many jobs. More generally, given any rank u, it turns out that we want to avoid situations where one server has only jobs of rank more than u while another has many jobs of rank u or less. The dispatching policies we consider are designed to avoid this situation.

At any moment in time t, let $v_i^{(t)}(s_1, s_2)$ be the total size of jobs dispatched so far to server i, counting only jobs whose size is in the interval $[s_1, s_2)$. We call $v_i^{(t)}(s_1, s_2)$ a *work counter* We say that a dispatching policy *obeys guardrails* if for all times t, servers $1 \leq i < j \leq k$, and integers $m \in \mathbb{Z}$, the work counters obey the following constraint:[1]

$$\left| v_i^{(t)}(2^m, 2^{m+1}) - v_j^{(t)}(2^m, 2^{m+1}) \right| \leq 2^{m+1}.$$

Instead of specifying a specific dispatching policy, our results about the M/G/k/dispatch assume only that the dispatching policy obeys guardrails.

It is simple to construct a dispatching policy that obeys guardrails. An example: if a job of size s arrives at time t, dispatch the job to server $\arg\min_i v_i^{(t)}(2^{\lfloor \log_2 s \rfloor}, 2^{\lfloor \log_2 s \rfloor + 1})$, breaking ties arbitrarily. However, this is far from the only policy that obeys guardrails. For example, Grosof et al. [3] show how to modify a generic dispatching policy to make it obey guardrails.

3. MAIN RESULTS

All systems discussed use "RS (with guardrails)", meaning they use RS scheduling (Section 2.2) and, in the case of the M/G/k/dispatch, a dispatching policy that obeys guardrails (Section 2.3). Unless otherwise specified, all of our results apply to any job size distribution S, any load ρ, and any number of servers k.

Our main theorem bounds the mean slowdown gap between the multiserver systems and the M/G/1.

Theorem 3.1.

(i) The mean slowdown gap between RS in the M/G/k and RS in the M/G/1 is bounded by $\mathbf{E}[Z_k] - \mathbf{E}[Z_1] \leq 6k$.

[1] The original definition of guardrails given by Grosof et al. [3] is both more general and more complicated than ours. The definition we give suffices for our purposes, but our results can be generalized to handle the more general definition.

(ii) The mean slowdown gap between RS with guardrails in the M/G/k/dispatch and RS in the M/G/1 is bounded by $\mathbf{E}[Z_k] - \mathbf{E}[Z_1] \leq 54k$.

These bounds imply that RS with guardrails is a constant-factor approximation for mean slowdown.

Corollary 3.2.

(i) In the M/G/k, RS has mean slowdown within a factor of 7 *of optimal.*

(ii) In the M/G/k/dispatch, RS with guardrails has mean slowdown within a factor of 55 *of optimal.*

Proof. The optimal mean slowdown in either multiserver system is at least k, because the servers run at speed $1/k$. It is also at least $\mathbf{E}[Z_1]$, because the M/G/1 can simulate either multiserver system. RS minimizes mean slowdown in the M/G/1 [4], so the result follows from Theorem 3.1. □

The bound in Theorem 3.1 implies that RS (with guardrails) is heavy-traffic optimal provided that $\lim_{\rho\to 1} \mathbf{E}[Z_1] = \infty$. However, analyzing $\mathbf{E}[Z_1]$ in heavy traffic is an open problem. We therefore characterize the heavy-traffic scaling of $\mathbf{E}[Z_1]$. For brevity, we omit the full theorem statement, instead focusing on the result's implications for multiserver systems.

Perhaps surprisingly, for job size distributions with infinite third moment, RS sometimes achieves *constant* mean slowdown in the heavy-traffic M/G/1, where the constant depends on the job size distribution. However, with finite third moment, RS's mean slowdown does diverge in the heavy-traffic limit, implying the following result.

Theorem 3.3. *In either the M/G/k or M/G/k/dispatch, if the job size distribution* S *satisfies* $\mathbf{E}[S^3] < \infty$*, then we have* $\lim_{\rho\to 1} \mathbf{E}[Z_k]/\mathbf{E}[Z_1] = 1$*, so RS (with guardrails) is heavy-traffic optimal for mean slowdown.*

4. PROOF OVERVIEW

We now sketch the proof of Theorem 3.1. The key concept that makes the proof possible is the following.

Definition 4.1. The *r-work* in the system is the total remaining size of all jobs whose rank is currently at most r. We write $W(r)$ for the equilibrium distribution of r-work.

To prove Theorem 3.1, rather than directly comparing mean slowdowns, our argument goes via r-work, as the following diagram illustrates.

$$\begin{array}{ccc} \mathbf{E}[Z_k] & \xrightarrow{\text{Theorem 3.1}} & \mathbf{E}[Z_1] \\ \text{Lemma 4.2}\updownarrow & & \updownarrow\text{Lemma 4.2} \\ \mathbf{E}[W_k(r)] & \xrightarrow[\text{Lemma 4.3}]{} & \mathbf{E}[W_1(r)] \end{array}$$

The rest of this section covers the two lemmas in the diagram. We prove Lemma 4.2 and sketch the proof of Lemma 4.3.

Lemma 4.2. *In the M/G/1, M/G/k, and M/G/k/dispatch,*

$$\mathbf{E}[Z] = \frac{1}{\lambda}\int_0^\infty \frac{\mathbf{E}[W(r)]}{r^2}\,\mathrm{d}r.$$

Proof. By a generalization of Little's law [1], it suffices to show that the r-work contributed by a single job in arbitrary state (s, x) integrates to the job's holding cost, namely $1/s$. The r-work contributed by such a job is $x\mathbb{1}(r \geq sx)$: its remaining size is x, but it is only counted if its rank sx is at most r. From this, we compute

$$\int_0^\infty \frac{x\mathbb{1}(r \geq sx)}{r^2}\,\mathrm{d}r = \int_{sx}^\infty \frac{x}{r^2}\,\mathrm{d}r = \frac{1}{s}.$$ □

Lemma 4.3. *Let*

$$\varphi_S(r) = \mathbf{E}\left[S\mathbb{1}(S \leq \sqrt{r}) + \frac{2r}{S}\mathbb{1}(S > \sqrt{r})\right].$$

(i) The mean r-work gap between the M/G/k and the M/G/1 is bounded by

$$\mathbf{E}[W_k(r)] - \mathbf{E}[W_1(r)] \leq \lambda k\sqrt{r}\varphi_S(r).$$

(ii) The mean r-work gap between the M/G/k/dispatch and the M/G/1 is bounded by

$$\mathbf{E}[W_k(r)] - \mathbf{E}[W_1(r)] \leq 9\lambda k\sqrt{r}\varphi_S(r).$$

Proof sketch. We use a Palm calculus argument based on Miyazawa's rate conservation law [5] (see also Scully et al. [6, Section 7]) to write an expression for $\mathbf{E}[W_k(r)] - \mathbf{E}[W_1(r)]$. We obtain a result that looks like

$$\mathbf{E}[W_k(r)] - \mathbf{E}[W_1(r)] \leq \lambda\varphi_S(r) \cdot w_{\max}(r).$$

Above, $w_{\max}(r)$ is, roughly speaking, the maximum amount of r-work that a single server can have while another server has no r-work.

It remains only to determine $w_{\max}(r)$, which we do differently for each multiserver system.

- In the M/G/k, $w_{\max}(r)$ refers to only a job in service. A single job's r-work is maximized if the job is in state $(\sqrt{r}, \sqrt{r})$, so $w_{\max}(r) = \sqrt{r}$.
- In the M/G/k/dispatch, $w_{\max}(r)$ refers to all jobs assigned to a server, including those waiting in its queue. A known property of guardrails [3, Lemma 1] suggests that this quantity cannot be too large, but the known result is not exactly the right for our setting. We prove a new variant of the property, showing $w_{\max}(r) = 9\sqrt{r}$. □

References

[1] Shelby L. Brumelle. 1971. On the Relation between Customer and Time Averages in Queues. *Journal of Applied Probability* 8, 3 (1971), 508–520.

[2] Isaac Grosof, Ziv Scully, and Mor Harchol-Balter. 2018. SRPT for Multiserver Systems. *Performance Evaluation* 127-128 (Nov. 2018), 154–175.

[3] Isaac Grosof, Ziv Scully, and Mor Harchol-Balter. 2019. Load Balancing Guardrails: Keeping Your Heavy Traffic on the Road to Low Response Times. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 3, 2, Article 42 (June 2019), 31 pages.

[4] Esa Hyytiä, Samuli Aalto, and Aleksi Penttinen. 2012. Minimizing Slowdown in Heterogeneous Size-Aware Dispatching Systems. *ACM SIGMETRICS Performance Evaluation Review* 40, 1 (June 2012), 29–40.

[5] Masakiyo Miyazawa. 1994. Rate Conservation Laws: A Survey. *Queueing Systems* 15, 1 (March 1994), 1–58.

[6] Ziv Scully, Isaac Grosof, and Mor Harchol-Balter. 2020. The Gittins Policy Is Nearly Optimal in the M/G/k under Extremely General Conditions. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 4, 3, Article 43 (Nov. 2020), 29 pages.

[7] Ziv Scully, Isaac Grosof, and Mor Harchol-Balter. 2021. Optimal Multiserver Scheduling with Unknown Job Sizes in Heavy Traffic. *Performance Evaluation* 145, Article 102150 (Jan. 2021), 31 pages.