Load Balancing Guardrails: Keeping Your Heavy Traffic on the Road to Low Response Times

Isaac Grosof
Carnegie Mellon University
Computer Science Department
Pittsburgh, PA, USA
igrosof@cs.cmu.edu

Ziv Scully
Carnegie Mellon University
Computer Science Department
Pittsburgh, PA, USA
zscully@cs.cmu.edu

Mor Harchol-Balter
Carnegie Mellon University
Computer Science Department
Pittsburgh, PA, USA
harchol@cs.cmu.edu

ABSTRACT

Load balancing systems, comprising a central dispatcher and a scheduling policy at each server, are widely used in practice, and their response time has been extensively studied in the theoretical literature. While much is known about the scenario where the scheduling at the servers is First-Come-First-Served (FCFS), to minimize mean response time we must use Shortest-Remaining-Processing-Time (SRPT) scheduling at the servers. Much less is known about dispatching polices when SRPT scheduling is used. Unfortunately, traditional dispatching policies that are used in practice in systems with FCFS servers often have poor performance in systems with SRPT servers. In this paper, we devise a simple fix that can be applied to any dispatching policy. This fix, called *guardrails*, ensures that the dispatching policy yields optimal mean response time under heavy traffic when used in a system with SRPT servers. *Any* dispatching policy, when augmented with guardrails, becomes heavy-traffic optimal. Our results yield the first analytical bounds on mean response time for load balancing systems with SRPT scheduling at the servers.

CCS CONCEPTS

• **General and reference** → **Performance**; • **Mathematics of computing** → **Queueing theory**; • **Networks** → **Network performance modeling**; • **Theory of computation** → *Routing and network design problems*; • **Computing methodologies** → *Model development and analysis*; • **Software and its engineering** → *Scheduling*;

KEYWORDS

load balancing; dispatching; scheduling; queueing; SRPT; response time; latency; sojourn time; heavy traffic

ACM Reference Format:
Isaac Grosof, Ziv Scully, and Mor Harchol-Balter. 2019. Load Balancing Guardrails: Keeping Your Heavy Traffic on the Road to Low Response Times. In *ACM SIGMETRICS / International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '19 Abstracts), June 24–28, 2019, Phoenix, AZ, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3309697.3331514

SIGMETRICS '19 Abstracts, June 24–28, 2019, Phoenix, AZ, USA

ACM ISBN 978-1-4503-6678-6/19/06.
https://doi.org/10.1145/3309697.3331514

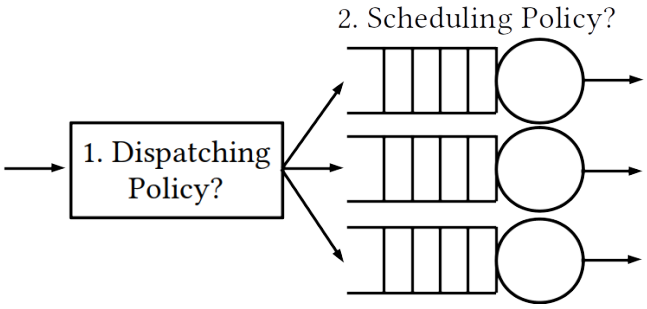

Figure 1: Two decision points within a load balancing system: (1) Pick the dispatching policy. (2) Pick the scheduling policy for the servers.

1 INTRODUCTION

Load balancers are ubiquitous throughout computer systems. They act as a front-end to web server farms, distributing HTTP requests to different servers. They likewise act as a front-end to data centers and cloud computing pools, where they distribute requests among servers and virtual machines.

In our paper [1], we consider the immediate dispatch load balancing model, where each arriving job is immediately dispatched to a server, as shown in Figure 1. Our goal is to find the *optimal* load balancing policy for this system. The system has two decision points:

(1) A *dispatching policy* decides how to distribute jobs across the servers.
(2) A *scheduling policy* at each server decides which job to serve among those at that server.

We ask:

> What (1) dispatching policy and (2) scheduling policy should we use to *minimize mean response time* of jobs?

We assume that the job arrival process is Poisson and that job sizes are i.i.d. from a general size distribution. We assume jobs are preemptible with no loss of work. Finally, we assume that job sizes are known at the time the job arrives in the system.

With these assumptions, the scheduling question turns out to be easy to answer: use Shortest-Remaining-Processing-Time (SRPT) at the servers. No matter what dispatching decisions are made, if we consider the sequence of jobs dispatched to a particular server, the policy which minimizes mean response time for that server must be to schedule those jobs in SRPT order. Thus, we assume SRPT is used at the servers.

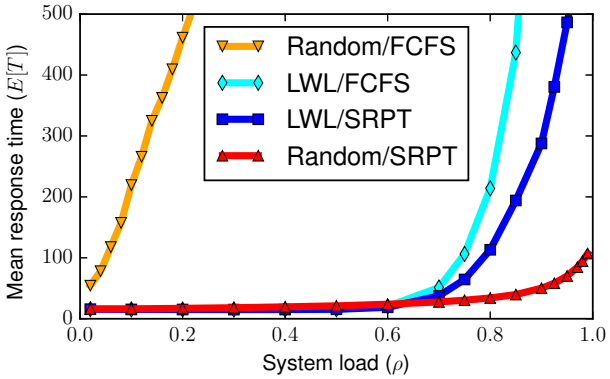

Figure 2: Two dispatching policies: Random and LWL. Two scheduling policies: FCFS and SRPT. FCFS scheduling at the servers yields higher mean response time as a function of load, compared with SRPT scheduling at the servers. Random dispatching is worse than LWL dispatching under FCFS scheduling at the servers, but Random dispatching is better than LWL dispatching under SRPT scheduling at the servers. Simulation uses $k = 10$ servers. Size distribution shown is Bimodal: 99.95% size 1 jobs and 0.05% size 1000 jobs.

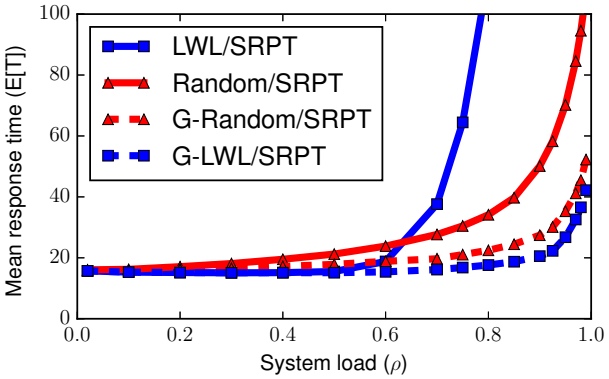

Figure 3: Adding guardrails to LWL yields much lower mean response time as a function of load. Guardrails yield a factor of 3 improvement even at $\rho = 0.8$, and a factor of 7 improvement at $\rho = 0.9$. Adding guardrails to Random also yields significantly lower mean response time as a function of load. Simulation uses $k = 10$ servers. Size distribution shown is Bimodal: 99.95% size 1 jobs and 0.05% size 1000 jobs.

The question remains: What dispatching policy minimizes mean response time given SRPT service at the servers? While many dispatching policies have been considered in the literature, they have mostly been considered in the context of First-Come-First-Served (FCFS) or Processor-Sharing (PS) scheduling at the servers. Popular dispatching policies include Random, Least-Work-Left (LWL), Join-Shortest-Queue (JSQ), JSQ-d, Size-Interval-Task-Assignment (SITA), Round-Robin (RR), and many more. However, only the simplest of these policies have been studied for SRPT servers.

One might hope that the same policies that yield low mean response time when servers use FCFS scheduling would also perform well when servers use SRPT scheduling. Unfortunately, this does not always hold. For example, when the servers use FCFS, LWL dispatching, which sends each job to the server with the least remaining work, outperforms Random dispatching. However, the opposite can happen when the servers use SRPT: as shown in Figure 2, Random/SRPT can outperform LWL/SRPT, and can do so by a factor of 5 or more under heavy load. This means that LWL is making serious mistakes. The heuristics that served us well for FCFS servers can steer us awry when we use SRPT servers.

We introduce *guardrails*, a new technique for creating dispatching policies. Guardrails view jobs as being classified into small jobs, medium-sized jobs, etc. Guardrails require the dispatcher to send a similar amount of work comprising small jobs to each server; likewise similar amounts of work of medium-sized jobs should be sent to each sever, and so forth. Our paper [1] precisely defines guardrails.

Given an arbitrary dispatching policy P, we can restrict P by disallowing any dispatch that would violate the guardrail requirement. Doing so results in an improved policy Guarded-P (G-P). We prove that G-P has asymptotically optimal mean response time in the heavy traffic limit, no matter what the initial policy P is. We also show empirically that adding guardrails to a policy almost always decreases its mean response time (and never significantly increases it), even outside the heavy-traffic regime.

As an example of the power of guardrails, Figure 3 shows the performance of guarded versions of LWL and Random. The guardrails stop LWL from making serious mistakes and dramatically improve its performance. Random dispatching also benefits from guardrails. Moreover, the guarded policies have a theoretical guarantee: In the limit as load $\rho \to 1$, G-Random/SRPT and G-LWL/SRPT converge to the optimal mean response time. In contrast, unguarded Random/SRPT is a factor of k worse than optimal in the $\rho \to 1$ limit, where k is the number of servers.

Our paper [1] makes the following contributions:

- We introduce guardrails, a technique for improving any dispatching policy.
- We bound the mean response time of any guarded dispatching policy when paired with SRPT scheduling at the servers. Using that bound, we prove that any guarded policy has asymptotically optimal mean response time as load $\rho \to 1$, subject to a technical condition on the job size distribution.
- We empirically show that guardrails improve a wide variety of common dispatching policies of these at all loads.
- We discuss practical considerations and extensions of guardrails:
  - guardrails for large systems, which may have multiple dispatchers and network delays;
  - guardrails for scheduling policies other than SRPT; and
  - guardrails for heterogeneous servers.

ACKNOWLEDGMENTS

This research was supported by NSF-XPS-1629444, NSF-CSR-180341, and a 2018 Faculty Award from Microsoft. Additionally, Ziv Scully was supported by an ARCS Foundation scholarship and the NSF GRFP under Grant Nos. DGE-1745016 and DGE-125222.

REFERENCES

[1] Isaac Grosof, Ziv Scully, and Mor Harchol-Balter. 2019. Load Balancing Guardrails: Keeping Your Heavy Traffic on the Road to Low Response Times. *Proc. ACM Meas. Anal. Comput. Syst.* 3, 2, Article 42 (June 2019), 31 pages. https://doi.org/10.1145/3326157