Optimal Scheduling in Multiserver Queues

Isaac Grosof
PhD Affiliation: Carnegie Mellon University, Computer Science Department
Current Affiliation: Georgia Institute of Technology, School of Industrial and Systems Engineering
Long-term Affiliation: Northwestern University, Dept. of Industrial Engineering & Management Sciences

ABSTRACT

Scheduling theory is a key tool for reducing latency (i.e. response time) in queueing systems. Scheduling, i.e. choosing the order in which to serve jobs, can reduce response time by an order of magnitude with no additional resources. Scheduling theory is well-developed in single-server systems, where one job is processed at a time. However, little is known about scheduling in multiserver systems, where many jobs are processed at once. Results are especially limited in stochastic multiserver scheduling theory. Today's datacenters have thousands of servers, and scheduling theory is unable to analyze such systems.

My thesis proves the first optimality results and first closed-form bounds on mean response time for scheduling policies in stochastic multiserver models which reflect the behavior of modern computing systems. My thesis proves novel optimality results in each of three areas:

I start by studying one-server-per-job multiserver models, and prove the first results on optimal scheduling in that setting. Next, I study the multiserver-job (MSJ) model, where different jobs require different amounts of resources to be served. I prove the first characterization of mean response time for any scheduling policy in the MSJ model, as well as the first optimality results. Finally, I study the effects of scheduling on the tail of response time, rather than mean response time. I invent a novel scheduling policy, Nudge, which I prove to be the first policy to outperform FCFS's asymptotic tail of response time.

This extended abstract briefly outlines the main contributions of my thesis, describing my novel performance bounds and optimality results in each of these three areas.

1 Introduction

In queueing systems, careful scheduling decisions can be very beneficial for improving system performance. In single-server models, where only one job is served at a time, scheduling is well understood. Queueing theorists have analyzed the mean response time of a wide variety of scheduling policies [17, 21], and have discovered the optimal scheduling policies for minimizing mean response time [16, 2].

However, modern queueing systems are increasingly often *multiserver* queues, where many jobs can be served at once, especially in modern computing systems. Despite their importance, little is known about scheduling in multiserver queueing models. Few analytical bounds on performance are known, and optimal scheduling is an open problem.

My thesis [3] proves the first closed-form analytical bounds on the performance of scheduling policies on multiserver queueing models. Moreover, I prove the first optimality results for multiserver scheduling. In particular, I prove novel performance bounds and optimality results in each of three areas:

- Scheduling in multiserver models where each job being served occupies one server.
- Scheduling in models where a given job may require many servers in order to run.
- Scheduling for performance metrics which capture the tail of response time.

2 One-server-per-job Models

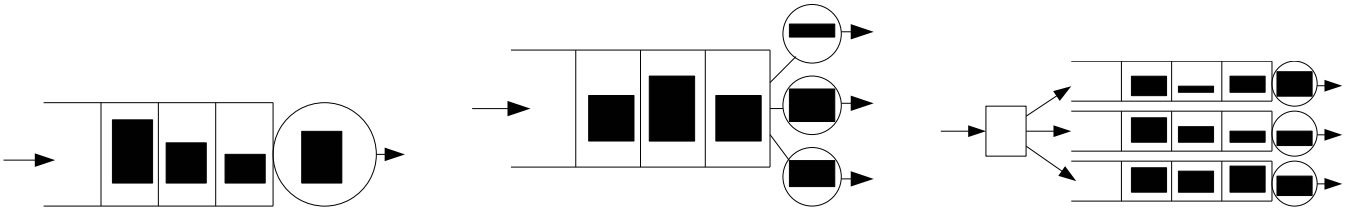
We want to understand how scheduling decisions affect the performance of multiserver models. Based on our experience with single-server models, we have the potential to dramatically improve performance if we can find the right scheduling policies.

In particular, consider the Shortest Remaining Processing Time (SRPT) policy. In the single-server setting, such as the M/G/1 model shown in Fig. 1a, SRPT is known to minimize mean response time [16], and its mean response time has been exactly analyzed [17].

As a result, it is natural to consider the SRPT scheduling policy in multiserver settings, such as the central-queue M/G/k depicted in Fig. 1b, and the dispatching M/G/k depicted in Fig. 1c.

Unfortunately, in each setting, SRPT's mean response time has not been analyzed, and it is unknown whether or not SRPT achieves optimal mean response time. Even beyond exact analysis, where the exact mean response time is derived, no closed-form bounds or asymptotic limits are known. Similarly, on the optimality question, even beyond exact optimality, *asymptotically* optimal scheduling policies are unknown.

A natural asymptotic limit is the "heavy traffic" limit, the limit as the load of arriving work approaches the capacity of the system. In this limit, the number of jobs present in the system at a time increases and scheduling becomes more and more important. My goal in this section of my thesis is to prove asymptotically optimality results in heavy traffic, and to analyze and bound the mean response time of these policies, with bounds that are tight in heavy traffic.



(a) M/G/1 (b) M/G/k (c) Dispatching M/G/k

Figure 1: Standard queueing models: In each case, jobs arrive according to a Poisson process, and have i.i.d. sizes. In the M/G/1, the scheduling policy can choose any single job to serve. In the M/G/k, the scheduling policy can choose any k jobs to serve at a time, at equal rates. In the dispatching M/G/k, the dispatching policy assigns each job to a server upon arrival, and the scheduling policy at each server can choose any single job to serve at that server.

2.1 Optimal Scheduling for the M/G/k

In [3, Chapter 2], which corresponds to my paper [8], I study the SRPT scheduling policy for the central-queue M/G/k depicted in Fig. 1b. In particular, I study the SRPT-k scheduling policy, which preemptively serves the k jobs of least remaining size, where k is the number of servers.

Given SRPT's optimality in the single-server setting [16], it is natural to ask whether SRPT-k is likewise optimal.

My key technique in studying SRPT-k is to compare SRPT-k against a resource-pooled system where all k servers are combined into one gigantic server. This gigantic server can serve any job at k times the speed of any of the original k servers. The resource-pooled system is a huge upgrade over the original M/G/k: One could split the gigantic server's efforts k ways, to emulate the original system, but one also has far more options. As a result, the resource-pooled system forms a lower bound that I can compare SRPT-k against. Specifically, I compare SRPT-k against the SRPT policy for the resource-pooled system, which we refer to as SRPT-1.

In particular, I prove that in the heavy traffic limit, the mean response time of SRPT-k and SRPT-1 converge. As a result, in that same limit, SRPT-k's mean response time converges to that of the optimal M/G/k scheduling policy:

THEOREM 2.1 (THEOREM 2.7.1, COROLLARY 2.7.1 OF [3]). *In an M/G/k with job size S such that $\mathbb{E}[S^2(\log S)^+]$ is finite,*

$$\lim_{\rho\to 1}\frac{\mathbb{E}[T^{SRPT\text{-}k}]}{\mathbb{E}[T^{SRPT\text{-}1}]}=1, \text{ and hence } \lim_{\rho\to 1}\frac{\mathbb{E}[T^{SRPT\text{-}k}]}{\mathbb{E}[T^{OPT\text{-}k}]}=1,$$

where T denotes a job's response time, and OPT-k is the M/G/k scheduling policy which minimizes mean response time.

In addition, I prove closed-form analytic bounds on SRPT-k's mean response time [3, Theorem 2.6.2], as well as stochastic bounds on SRPT-k's response time distribution [3, Theorem 2.6.1].

Beyond SRPT-k, I also prove the first analysis and optimality results for multiserver Preemptive Shortest Job First (PSJF-k), Remaining Size Times Original Size (RS-k), and Foreground-Background (FB-k).

All of these results, except for the FB-k result, focus on settings where a job's exact service duration (its *size*) is known to the scheduling policy before the job is served.

In follow-up work, we prove the first analysis and optimality results in multiserver scheduling settings where a job's size is unknown to the scheduling policy, or is partially known to the scheduling policy, but the job size distribution is known [18, 19]. If the size distribution is unknown, optimal scheduling is open even in the single-server setting, and we prove an initial optimality result in that setting [20].

Further work by others on this topic has also proven analysis and optimality results in settings where the arrival process is non-Poisson [13].

2.2 Optimal Scheduling and Load Balancing

In [3, Chapter 3], which corresponds to my paper [9], I study the SRPT scheduling policy for the dispatching M/G/k depicted in Fig. 1c.

Modern computing systems typically contain huge numbers of servers, allowing them to process many jobs at once. At this large scaling, holding jobs in a central queue can become a bottleneck. Instead, these multiserver systems often *dispatch* an arriving job to a specific server immediately upon the job's arrival. In such a system, both the dispatching policy (where to send each job) and the scheduling policy (which job at each server to schedule) are important to the performance of the system.

Prior work on dispatching systems has focused on combining nontrivial dispatching policies with First-Come First-Served (FCFS) scheduling at each server. Optimality results are known for Least-Work-Left (LWL) dispatching in this setting [12], but only under the assumption that FCFS scheduling is used at the servers, and that the dispatcher does not know the size of the arriving job.

It is straightforward to show that SRPT scheduling is the optimal scheduling policy for minimizing mean response time, in combination with any dispatching policy. Surprisingly, policies with great performance under FCFS scheduling, such as LWL, can have terrible performance under SRPT scheduling, performing worse than random dispatching in some settings.

I invent a novel class of dispatching polices, "guardrails" polices. The key idea behind guardrails policies is to balance not just the total amount of work at each server as in LWL, but also the amount of work of *each kind of job* at each server. Imagine labeling every job as small, medium, or large, based on its size. Guardrails policies ensure that each server has the same amount of work of small jobs, of medium jobs, and of large jobs. As a result, if any small jobs are present in the system, all servers will work on small jobs.

Using this key idea, I compare the guardrails/SRPT combination against the same resource-pooled SRPT-1 system as in Section 2.1. In [3, Theorem 3.4.2 and Corollary 3.4.1], I prove that the mean response time of any guardrails/SRPT

combination converges to that of SRPT-1 in heavy traffic, and hence that any guardrails/SRPT achieves optimal mean response time in the heavy traffic limit.

3 Multiserver-job Models

In the real world, jobs in large-scale computing systems have a huge spread of resource requirements. Moreover, some jobs require a large fraction of the entire system's resources, such as large machine-learning training jobs. As a result, it is vitally important to pack the jobs onto the servers in an efficient manner. Poor packing can lead to poor utilization, namely using only a small fraction of system resources. In practice, utilization is often poor in these large-scale systems. Scheduling has the opportunity to improve utilization, and by extension overall performance.

To model systems in which different jobs require different amounts of resources, we consider the *multiserver-job* (MSJ) model, in which each job requires some number of servers, its *server need*. Each job is specified by the pair (server need, duration).

Prior work on the MSJ model focused on characterizing its stability region [15, 6], the range of arrival rates for which the system remains stable, and using scheduling to optimize that stability region [14, 1]. In contrast, the effect of the scheduling policy on mean response time was completely open, and no optimality results for mean response time were known.

In this section, I start by devising the first MSJ scheduling policy for which I can analyze mean response time, the *ServerFilling* policy. Next, I build upon that policy to design a new scheduling policy for which I prove the first result on optimal mean response time in the MSJ system. Finally, I analyze the mean response time of the MSJ FCFS scheduling policy, which is practically important to understand, as it is used as a component in many scheduling policies which are popular in practice.

3.1 First MSJ Mean Response Time Analysis

In [3, Chapter 4], which corresponds to my paper [5], my goal is to find any MSJ scheduling policy for which I can analyze mean response time.

Prior to this work, mean response time had not been theoretically characterized in closed form for any MSJ scheduling policy, except for FCFS service in a system with $k = 2$ servers.

My key technique was to identity and focus on the class of *work-conserving finite skip* scheduling polices. "Work conservation" refers to keeping all servers occupied, whenever a sufficient number of jobs are present. "Finite skip" refers to serving jobs in near-FCFS order, only choosing between a finite number of the oldest jobs in the system. I develop a clean, unified mean response time analysis for all WCFS scheduling policies, proving that all WCFS polices achieve mean response time nearly identical to that of resource-pooled FCFS.

Focusing on the WCFS class of policies, I invent two MSJ policies which lie within the WCFS class, ServerFilling and DivisorFilling. ServerFilling operates in the setting where all jobs' server needs are powers of two, and the total number of servers k is a power of two. This setting is of relevance in supercomputing and other large-scale computing environments, as it allows jobs to pack better and improve utilization. DivisorFilling operates in the setting where all jobs' server needs perfectly divide the total number of servers k, which generalizes the power-of-two setting. In each case, my WCFS analysis proves tight bounds on the mean response times of the MSJ policy, thereby giving that the first closed-form bounds on mean response time for any MSJ scheduling policies.

In related work, we contrast the pros and cons of ServerFilling with other MSJ scheduling policies that are popular in theory and practice [4].

3.2 Optimal MSJ Scheduling

In [3, Chapter 5], which corresponds to my paper [10], my goal is to find a MSJ scheduling policy which achieves optimal mean response time in heavy traffic.

This chapter focuses on combining the ServerFilling policy introduced in [3, Chapter 4] with the SRPT scheduling policy, which is known to achieve optimal mean response time in the single-server setting [16], and which I proved achieves optimal mean response time in heavy traffic in [3, Chapter 2]. I invent the ServerFilling-SRPT policy, which combines the advantages of both polices: It uses the full capacity of the MSJ system, like ServerFilling, while prioritizing jobs with small remaining size, like SRPT. One key idea is using the appropriate concept of "size" for this prioritization. I define a job's size to be the product of its server need and its service duration, and I define SRPT correspondingly.

With this definition of job size, I show that the mean response time of ServerFilling-SRPT converges to that of resource-pooled SRPT-1 in heavy traffic, where the size distribution in the resource-pooled system matches the size distribution in the MSJ system. As a result, ServerFilling-SRPT achieves optimal mean response time in heavy traffic for the MSJ system.

I also prove optimality results for DivisorFilling-SRPT and the Gittins variants of both policies, in their corresponding settings.

3.3 Analyzing MSJ FCFS

In [3, Chapter 6], which corresponds to my paper [7], I study the FCFS scheduling policy for the MSJ system.

In the MSJ setting, the FCFS policy is not work conserving: If the job at the head of the queue requires more servers than are available, that job must wait for servers to become available, and all jobs behind it must wait as well. MSJ FCFS is important as a component of algorithms which are popular in practice, as system operates often want to guarantee that a job is served no later than it would be served under FCFS.

In this chapter, I analyze the mean response time the MSJ FCFS policy. The key challenge obstructing this analysis is that the MSJ FCFS does not use the full capacity of the system, and hence does not resemble a resource-pooled system, unlike each of the previous chapters of this thesis.

Instead, my key idea is to relate the MSJ FCFS system to a single-server queue with a Markov-modulated service rate. I first prove that MSJ FCFS's mean response time is nearly identical to such a system with an appropriately chosen service-rate modulation process. Then, I give the first closed-form mean response time analysis for the Markov-modulated service rate system in heavy traffic. Combining these results yields the first mean response time analysis of MSJ FCFS in heavy traffic.

4 Tail scheduling: Nudge

In [3, Chapter 7], which corresponds to my paper [11], I turn my focus to optimizing objectives focusing on the tail of response time.

Standard queueing theory analysis focusing on mean response time, but in the real world, system operator typically focus on tail metrics such as the 99th percentile of response time. Even in the single-server setting, the effects of scheduling on the tail of response time are poorly understood. In this chapter, I focus on the single-server setting, to serve as a stepping stone to analyzing the tail of response time in more realistic multiserver models.

One of the few results in this setting demonstrates that FCFS achieves *weakly optimal* asymptotic tail probability, meaning that the probability that response time T exceeds a threshold decays almost as quickly as possible, within a constant multiplicative factor of the optimal decay rate. It has been conjectured that FCFS is *strongly optimal*, meaning that the multiplicative factor converges to 1. These results hold in the setting of light-tailed job size distributions, which is the focus of this chapter.

Contrary to that conjecture, I devise a simple scheduling policy, Nudge, which provably improves upon FCFS's asymptotic tail of response time by a multiplicative factor. Moreover, I prove that Nudge's response time distribution stochastically dominates that of FCFS, achieving a strictly better response time distribution:

THEOREM 4.1 (THEOREM 7.5.2 AND COR. 7.5.1 OF [3]).

$$\forall t > 0, P(T^{Nudge} > t) < P(T^{FCFS} > t)$$
$$\lim_{t\to\infty} \frac{P(T^{Nudge} > t)}{P(T^{FCFS} > t)} < 1.$$

Moreover, Nudge is simple to implement and only requires a small amount of job size information, giving it a strong potential for practical application.

5 References

[1] J. Ghaderi. Randomized algorithms for scheduling VMs in the cloud. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9, Apr. 2016.

[2] J. C. Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society: Series B (Methodological)*, 41(2):148–164, 1979.

[3] I. Grosof. *Optimal Scheduling in Multiserver Queues*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 2023.

[4] I. Grosof and M. Harchol-Balter. Invited paper: ServerFilling: A better approach to packing multiserver jobs. In *Proceedings of the 5th Workshop on Advanced Tools, Programming Languages, and PLatforms for Implementing and Evaluating Algorithms for Distributed Systems*, ApPLIED 2023, New York, NY, USA, 2023. Association for Computing Machinery.

[5] I. Grosof, M. Harchol-Balter, and A. Scheller-Wolf. WCFS: A new framework for analyzing multiserver systems. *Queueing Systems*, 2022.

[6] I. Grosof, M. Harchol-Balter, and A. Scheller-Wolf. New stability results for multiserver-job models via product-form saturated systems. *MAthematical performance Modeling and Analysis (MAMA)*, 2023.

[7] I. Grosof, Y. Hong, M. Harchol-Balter, and A. Scheller-Wolf. The RESET and MARC techniques, with application to multiserver-job analysis. *Performance Evaluation*, 162:102378, 2023.

[8] I. Grosof, Z. Scully, and M. Harchol-Balter. SRPT for multiserver systems. *Performance Evaluation*, 127-128:154–175, 2018.

[9] I. Grosof, Z. Scully, and M. Harchol-Balter. Load balancing guardrails: Keeping your heavy traffic on the road to low response times. *Proc. ACM Meas. Anal. Comput. Syst.*, 3(2), jun 2019.

[10] I. Grosof, Z. Scully, M. Harchol-Balter, and A. Scheller-Wolf. Optimal scheduling in the multiserver-job model under heavy traffic. *Proc. ACM Meas. Anal. Comput. Syst.*, 6(3), dec 2022.

[11] I. Grosof, K. Yang, Z. Scully, and M. Harchol-Balter. Nudge: Stochastically improving upon FCFS. In *Abstract Proceedings of the 2021 ACM SIGMETRICS / International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '21, page 11–12, New York, NY, USA, 2021. Association for Computing Machinery.

[12] M. Harchol-Balter. Job placement with unknown duration and no preemption. *ACM SIGMETRICS Performance Evaluation Review*, 28(4):3–5, 2001.

[13] Y. Hong and Z. Scully. Performance of the Gittins policy in the G/G/1 and G/G/k, with and without setup times. *Performance Evaluation*, 163:102377, 2024.

[14] S. T. Maguluri, R. Srikant, and L. Ying. Stochastic models of load balancing and scheduling in cloud computing clusters. In *2012 Proceedings IEEE Infocom*, pages 702–710. IEEE, 2012.

[15] A. Rumyantsev and E. Morozov. Stability criterion of a multiserver model with simultaneous service. *Annals of Operations Research*, 252(1):29–39, 2017.

[16] L. Schrage. A proof of the optimality of the shortest remaining processing time discipline. *Operations Research*, 16(3):687–690, 1968.

[17] L. E. Schrage and L. W. Miller. The queue M/G/1 with the Shortest Remaining Processing Time discipline. *Operations Research*, 14(4):670–684, 1966.

[18] Z. Scully, I. Grosof, and M. Harchol-Balter. The Gittins policy is nearly optimal in the M/G/k under extremely general conditions. *Proc. ACM Meas. Anal. Comput. Syst.*, 4(3), Nov. 2020.

[19] Z. Scully, I. Grosof, and M. Harchol-Balter. Optimal multiserver scheduling with unknown job sizes in heavy traffic. *Performance Evaluation*, 145:102150, 2021.

[20] Z. Scully, I. Grosof, and M. Mitzenmacher. Uniform bounds for scheduling with job size estimates. In *13th Innovations in Theoretical Computer Science Conference, ITCS*, 2022.

[21] Z. Scully, M. Harchol-Balter, and A. Scheller-Wolf. SOAP: one clean analysis of all age-based scheduling policies. *Proc. ACM Meas. Anal. Comput. Syst.*, 2(1):16:1–16:30, Apr. 2018.

6 Author Biography

Isaac Grosof completed their PhD in Computer Science at Carnegie Mellon University in 2023, where they were advised by Prof. Mor Harchol-Balter. They will join Northwestern University's Department of Industrial Engineering and Management Science as an assistant professor in Fall 2024. They are conducting postdoctoral research at Georgia Tech in Fall 2023 and at University of Illinois, Urbana-Champaign in Spring 2024.

Isaac studies queueing theory and performance modeling and evaluation. They specifically focus on analyzing and optimizing stochastic queueing models, such as models of job scheduling, load balancing, and fluctuating load.

Isaac's work has been recognized by Best Student Paper and Best Paper awards from ACM SIGMETRICS, IFIP Performance, and INFORMS, including winning the 2022 George Nicholson Student Paper Competition.