Optimal Scheduling in Modern Queueing Systems

Multiserver and Tail Scheduling for Computer Systems

Isaac Grosof

April 12, 2022

1 Introduction

1.1 Motivation

Computer systems contain a multitude of queues: Network switches serving packets, databases serving queries, processors serving processes, caches responding to loads and stores, and more. Each queue has a *scheduling policy* that it uses to determine which job(s) to serve at a given point in time. The choice of scheduling policy has a major impact on a system's performance, such as the distribution of response time[1] in the system. Given the importance of scheduling, *optimal* scheduling takes on particular importance: achieving the best possible performance with a given amount of resources. The question of optimal scheduling has therefore been studied extensively [17, 37, 40].

In sufficiently simple settings, exact optimal scheduling policies are known. For instance, given the setting of the preemptive M/G/1 queue, shown in Fig. 1a, and given the performance objective of minimizing mean response time, the optimal policy is known. When the scheduler knows each job's exact size[2], the *Shortest Remaining Processing Time* (SRPT) policy is optimal [40]. If the job's size is unknown to the scheduler, the *Gittins* policy is optimal [17].

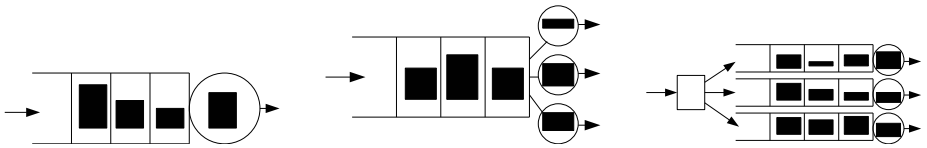
1.1.1 Multiserver queues

Real computer systems are rarely so simple. In modern computer systems, an increasing number of queues are *multiserver*, where many jobs from the queue are served at once: databases are multithreaded, datacenters aggregate hundreds of machines, and even a single processor has many instructions in flight at once. The modern frontier of performance lies in utilizing parallelism effectively.

The simplest multiserver system is the M/G/k queue, shown in Fig. 1b, where already very little is known. Mean response time has only been bounded for First-Come First-Served (FCFS) service, the simplest service policy [27, 28,

[1] A job's response time is the duration from its arrival to its completion. Also known as latency or sojourn time.

[2] A job's size is the amount of service necessary to complete the job.



(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.

30]. Analyzing mean response time under nontrivial scheduling policies is an open problem, addressed for the first time in this thesis. Optimal scheduling is entirely open, also first handled by this thesis.

In modern computer systems, the central queue of the M/G/k can become a bottleneck, so a *dispatching* model is typically preferred, as shown in Fig. 1c. In the dispatching model, each job must be irrevocably assigned to a server upon arrival. While this system has received much study, that study has focused on heuristic policies for the dispatching M/G/k or optimality in simpler settings [5, 10, 24, 49, 51] This thesis provides the first results on optimal scheduling in the dispatching M/G/k.

1.1.2 Concurrent Service

Another important feature of modern queueing systems is *concurrent service* of one job across many servers. In supercomputers, one job may demand anywhere from one core to thousands. In modern datacenters, the amount of computational resources demanded by different jobs can vary by a factor of 10^5 [21, 48]. A job might require a fixed number of servers to run, or might receive variable speedup [6] depending on the number of servers utilized. Little is known from a theoretical perspective about systems with concurrent service: results mostly focus on stability regions and heuristic policies, rather than bounds or optimality results [7, 11, 31, 38]. This thesis provides the first results on optimal scheduling in concurrent service models.

1.1.3 Single-server Tails

The most important performance metrics in modern computer systems are likewise shifting, towards *tail* performance, such as measures of the tail of response time. Websites often set Service Level Objectives based on the tail of response time, and database designers often focus on minimizing the 99th percentile of response time [23, 33]. Optimizing the tail of response time therefore takes on more importance than optimizing mean response time in modern systems.

From a theoretical perspective, the tail of response time is already not well understood even in simpler single-server systems, to say nothing of more complex multiserver systems. A simple measure of the tail of response time is the asymptotic tail of response time the behavior of $P(T > t)$ as $t \to \infty$. In the M/G/1, *weakly* optimal policies are known [9, 35, 43], where $P(T > t)$ as $t \to \infty$ is optimal up to a constant multiplicative factor, but strong optimality, where the constant must also be optimized, remained open until this thesis. Likewise, optimizing more complex tail metrics such as $E[T^2]$ or T^{99} (the 99th percentile of response time) is also an open problem.

1.2 Contributions

In the next three sections, I describe my thesis work in more detail. Each section focuses on a problem theme, and proposes solving several problems within that theme.

1.2.1 Multiserver Queues

Section 2 covers optimal scheduling in *standard multiserver models*, namely the M/G/k and the dispatching M/G/k, both shown in Fig. 1. I specifically focus on deriving the first policies that yield asymptotically optimal mean response time in the heavy traffic limit, as load $\rho \to 1$.

First, I[3] study how to optimally schedule in the M/G/k with *known sizes* to minimize mean response time. I prove that the SRPT policy has asymptotically optimal mean response time in the heavy traffic limit, and prove the first bounds on mean response time for the M/G/k/SRPT across all loads ρ. I also prove the first bounds on mean response time for several other important M/G/k scheduling policies. These bounds are the first known for any nontrivial scheduling policies for the M/G/k.

Next, I study how to optimally schedule in the M/G/k with *unknown sizes*, or in settings with *partial size information*. I prove that the Gittins policy has asymptotically optimal mean response time in the heavy traffic limit, and prove the first bounds on mean response time for the M/G/k/Gittins across all loads ρ.

Finally, I study how to optimally dispatch and schedule in the dispatching M/G/k. I introduce the *guardrails* class of dispatching policies, and prove that any dispatching policy from the guardrails class yields asymptotically optimal mean response time when combined with SRPT scheduling at each server.

1.2.2 Concurrent Service

Section 3 studies optimal scheduling in multiserver models with *concurrent service*. First, I introduce the work-conserving, finite-skip (WCFS) class of models,

[3] Throughout this document, I phrase all results with the singular first person - "I", "my", etc. I write this way for consistency and clarity. However, I want to make clear that most of these results were proven in collaboration with my many wonderful coauthors, not by me alone.

a very general class of queueing models which includes two important models with concurrent service: the *threshold parallelism model* under FCFS service and the *multiserver-job model* with a novel scheduling policy which I introduce. I prove additively-tight bounds on mean response time for any WCFS model, giving the first bounds on mean response time for my concurrent service models.

Next, I introduce the WCFS-SRPT class of models, which combines the WCFS class with the SRPT policy. By considering this class of models, I propose and analyze the SRPT-ServerFilling policy for the multiserver-job model, which I prove is the first multiserver-job scheduling policy with asymptotically optimal mean response time.

Finally, I discuss a more general finite-skip setting, with variable work completion rate. This setting captures additional important concurrent service models, including FCFS and backfilling policies for the multiserver-job system, and parallel models with variable speedup functions. I derive a subpolynomially-tight[4] characterization of the asymptotic behavior of mean response time in the heavy traffic limit for all models in this setting.

1.2.3 Single-server tails

Section 4 covers scheduling for the tail of response time, focusing on single-server systems for simplicity.

First, I focus on the asymptotic tail of response time, studying $P(T > t)$ in the limit as the threshold $t \to \infty$. We compare against FCFS scheduling in the setting of light-tailed[5] job sizes, which is the setting of interest because in the light-tailed setting, FCFS scheduling is known to have weakly optimal asymptotic tail [9], and has been conjectured to have strongly optimal asymptotic tail [50]. However, I introduce the Nudge policy, which I prove has strictly smaller asymptotic tail of response time than FCFS, disproving the above conjecture. Moreover, I show that Nudge's response time distribution stochastically dominates that of FCFS.

Next, I turn to the goal of optimizing non-asymptotic tail metrics, such as $E[T^2]$. An important candidate policy for optimizing such a metric is an Accumulating Priority policy, where a jobs priority increases linearly with time in system. Accumulating Priority policies have been studied, but only in the special case where there are a finite number of priority accumulation rate classes [47]. However, optimizing non-asymptotic tail metrics requires continuous, size-based priority accumulation rates, which is not covered by the existing analysis. I am currently working on generalizing the existing results to this setting.

[4] See Theorem 3.4.

[5] Specifically, Class I job size distributions. See Section 4.1.

2 First Scheduling Analysis and Optimality for Standard Multiserver Models

Multiserver queueing systems are increasingly ubiquitous throughout modern computer systems, common in multicore machines, network switches and entire datacenters. I study optimal scheduling in the M/G/k, one of the most fundamental multiserver queueing systems. Modern computer systems also often employ immediate dispatching upon arrival, to avoid the potential bottleneck of a central queue. To handle this setting, I study optimal scheduling in the dispatching M/G/k.

2.1 Setting and problem statements

The first queueing model I study is the M/G/k model. In the M/G/k, jobs arrive to a central queue according to a Poisson process with some rate λ. Each job has a size, which is sampled i.i.d. from some job size distribution with random variable S. At any given point in time, the scheduling policy selects up to k jobs, each of which is run at the k servers. I assume that jobs can be preempted and resumed with no loss of work.

When a job is served, it completes at rate $1/k$. Specifically, the job's age, which represents the amount of service that has been performed on the job, increases at rate $1/k$ whenever the job is in service. A job completes when its age reaches its size. Because each server runs at rate $1/k$, the total capacity of the system is 1.

An important aspect of the system is the load ρ, defined as $\lambda E[S]$. I assume $\rho < 1$, to ensure that the system is stable. Load ρ is equal to the average fraction of servers that are busy. I study the M/G/k in the heavy-traffic limit, meaning as load $\rho \to 1$.

I also study the M/G/k with *immediate dispatch*. In this model, when a job arrives, the job must immediately be placed at a server, where it will eventually be served.

I also study a variety of *job information* models, which model the information about the size of a job that the scheduler has access to before the job runs. In the *known size* setting, the scheduler has perfect information, knowing the size of each job upon arrival. In the *blind* setting, the scheduler has no information other than the job size distribution. I also consider a variety of *partial information* settings.

Two important scheduling policies are the Shortest Remaining Processing Time (SRPT) policy and the Gittins policy. SRPT serves the job of least remaining size, where a job's remaining size is defined as the job's size minus its age. SRPT can only be used in a perfect information setting.

Gittins has a more complicated definition, which is specific to the job information model in use. In the blind setting, the rank of job with age a is defined

as follows:

$$r(a) = \inf_{b>a} \frac{E[\min(S,b) - a \mid S > a]}{P(S \leq b \mid S > a)}$$

The Gittins policy serves the job of least rank at all times. In general information settings, the Gittins policy can be defined with reference to the Gittins game [42].

SRPT and Gittins are each the optimal scheduling policy in the M/G/1, the corresponding *single-server* model. SRPT is optimal in the perfect information setting [40], while Gittins is optimal in the blind and partial information settings [17, 41].

I study the response time distribution of a given scheduling policy in a given setting. A job's response time is the time from when the job arrives to when it completes. Let $T^{\pi\text{-}k}$ be a random variable representing the response time of a generic job under the scheduling policy π in the M/G/k. For a job of size x, let $T^{\pi\text{-}k}(x)$ be a random variable representing the response time of a job of size x specifically. For example, $T^{SRPT\text{-}k}$ is the response time of a job in an M/G/k with SRPT scheduling.

In the dispatching M/G/k, we must specify both the dispatching and scheduling policies. Under a dispatching policy δ and a scheduling policy π, let $T^{\delta/\pi\text{-}k}$ denote a job's response time in the dispatching M/G/k.

The problems I study are:

- What scheduling policy minimizes mean response time in the known size M/G/k, in the limit as load $\rho \to 1$?
- What scheduling policy minimizes mean response time in the blind and partial information M/G/k models, in the limit as load $\rho \to 1$?
- What combination of dispatching and scheduling policies minimizes mean response time in the dispatching M/G/k with known sizes, in the limit as load $\rho \to 1$?

2.2 Prior work

Prior study of the M/G/k has mostly been limited to study of the FCFS scheduling policy [27, 28, 30]. Study of more complex policies has been limited to simpler settings, such as simpler job size distributions. My work is the first to study more complex scheduling policies for the M/G/k.

2.2.1 M/G/k Perfect Information

In the known-size single-server setting, the SRPT policy has long been known to achieve optimal mean response time, even in the worst-case setting [40]. However, in the multiserver setting, worst-case counterexamples are known which separate mean response time under SRPT-k from that of the optimal policy by an arbitrarily large factor [29].

As for SRPT in the M/G/k, there are no prior results. The closest system that has been analyzed is the multiserver priority queue, in which there are finitely many classes of jobs, with phase-type service requirement distributions [25, 32, 45]. While formulas for exact solutions to the steady state of these systems are known, these exact solutions are computationally intensive to compute, and do not provide insight into whether the resulting scheduling policies are optimal or near-optimal. For more on the SRPT policy for the M/G/k, see the prior work section of [19], my paper on the topic.

2.2.2 M/G/k under Blind and Partial Information

Turning to the blind and partial information setting, first note that the Gittins policy is known to achieve optimal mean response time in the M/G/1 [17, 41]. In the M/G/k, much less is known about achieving optimality under imperfect information.

One important, albeit simpler, setting under which an optimality result is known is the Multiclass M/M/k with Bernoulli feedback and finitely many job classes [18]. This is a special-case of the M/G/k with partial information that I study. In this setting, Glazebrook and Niño-Mora prove that the Gittins policy achieves heavy-traffic optimality, as well as proving tight bounds.

Beyond the M/G/1 and the setting of Glazebrook and Niño-Mora, no optimality results are known in the blind or partial-information settings. For more, see the prior work sections of [42, 44], my two papers on this topic.

2.2.3 Dispatching M/G/k

On the dispatching M/G/k with known sizes, many dispatching policies, including size-based dispatching policies, have been studied in combination with simpler scheduling policies at the servers, most often FCFS and Processor Sharing (PS). Well-studied dispatching policies in these settings include Least Work Left, Join the Shortest Queue, Size Interval Task Assignment, and many more [10, 24, 49, 51]. However, to achieve optimality, a size-based scheduling policy at the servers, such as SRPT, is required. In this setting, little was known prior to my work. A worst-case impossibility result is known, akin to the result for SRPT-k in the central-queue multiserver system [3]. In the stochastic setting, nothing was known about optimality. For more, see the prior work section of [20], my paper on this topic.

2.3 Main Results

Because SRPT and Gittins are optimal in the single-server setting, it is natural to consider their mean response time in the multiserver setting as well. I prove bounds on mean response time for these policies in the M/G/k. These bounds are tight enough to imply that SRPT and Gittins are asymptotically optimal in the heavy traffic limit.

2.3.1 Perfect information M/G/k

I first focus on SRPT, corresponding to the perfect information setting:

Theorem 2.1 (SRPT-k mean response time bound [19]). *In an M/G/k with perfect information, the mean response time of a job of size x under the SRPT scheduling policy is bounded by*

$$E[T^{SRPT\text{-}k}(x)] \leq \frac{\lambda \int_0^x x^2 f_S(t)dt}{2(1-\rho_{\leq x})^2} + \frac{k\rho_{\leq x}x}{1-\rho_{\leq x}} + k\int_0^x \frac{1}{1-\rho_{\leq t}}dt$$

where $\rho_{\leq x} = \lambda \int_0^x x f_S(t)$.

Integrating Theorem 2.1 over all job sizes x, I get a bound on mean response for SRPT in the M/G/k, $E[T^{SRPT\text{-}k}]$. I compare this bound to the mean response time of SRPT in the M/G/1, $E[T^{SRPT\text{-}1}]$. I prove that, under a mild condition on S, the mean response time of SRPT in the M/G/k and SRPT in the M/G/1 converge in the heavy traffic limit:

Theorem 2.2 (SRPT-k converges to SRPT-1 [19]). *In an M/G/k with job size distribution S, if $E[S^2(\log S)^+] < \infty$*[6]*,*

$$\lim_{\rho\to 1} \frac{E[T^{SRPT\text{-}k}]}{E[T^{SRPT\text{-}1}]} = 1$$

I use Theorem 2.2 to prove that SRPT has asymptotically optimal mean response time in the M/G/k:

Theorem 2.3 (SRPT-k is asymptotically optimal [19]). *In an M/G/k with job size distribution S, if $E[S^2(\log S)^+] < \infty$,*

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

where OPT-k is the optimal scheduling policy for the M/G/k.

2.3.2 Blind and partial information M/G/k

Next, I prove analogous results for Gittins in the M/G/k. Doing so required the development of novel techniques for quantifying and bounding mean response time. Using these techniques, I first bound mean response time:

Theorem 2.4 (Gittins-k mean response time bound [42]). *In an M/G/k with job size distribution S, under any job information setting, for all $\alpha > 1$ such that $E[S^\alpha] < \infty$, the mean response time of the Gittins policy is bounded by*

$$E[T^{Gittins\text{-}k}] \leq E[T^{Gittins\text{-}1}] + (k-1)E[S]\left(\frac{1}{1-\alpha}\left(\log\frac{1}{1-\rho} + \log\frac{E[S^\alpha]}{\alpha E[S]^\alpha} + 1\right) + 4.547\right)$$

[6]Note that this condition on job size is slightly different than the one stated in [19], which focused on the Matusewska index. This condition is both less restrictive and more accurate. It was derived in my later paper [42], but also applies to this result and to Theorem 2.6.

where $E[T^{Gittins\text{-}1}]$ is the mean response time of Gittins in the M/G/1.

From this, I prove convergence of Gittins-k to Gittins-1, and hence to the optimal M/G/k policy in the heavy traffic limit:

Theorem 2.5 (Gittins-k is asymptotically optimal [42])**.** *In an M/G/k with job size distribution S, under any job information setting, if $E[S^2(\log S)^+] < \infty$,*

$$\lim_{\rho\to 1} \frac{E[T^{Gittins\text{-}k}]}{E[T^{Gittins\text{-}1}]} = \lim_{\rho\to 1} \frac{E[T^{Gittins\text{-}k}]}{E[T^{OPT\text{-}k}]} = 1$$

2.3.3 Dispatching M/G/k with perfect information

Finally, I study the dispatching M/G/k with perfect size information. In this setting, I must answer two questions: How to optimally dispatch the jobs to the servers, and how to optimally serve the jobs at a given server.

Using the worst-case optimality of SRPT, it is easy to show that for any dispatching policy, the optimal service policy is SRPT. My focus therefore turns to choosing a dispatching policy.

I define a class of dispatching policies called "guardrails policies", for which I prove an asymptotic optimality result, following my paper [20]. Intuitively, a policy in the guardrails class must always dispatch similar amounts of work to each server. Moreover, a guardrails policy must dispatch similar amounts of work of all sizes to all servers. Similar work of small jobs to all servers, of medium jobs to all servers, and so on.

To make this concrete, I first define the *rank* of a job of size x to be $r = \lfloor \log_c x \rfloor$, where $c > 1$ is a constant called the *guardrail rank width*. The concept of the guardrails class is to spread out the jobs within a rank r across all of the servers. To do so, for each rank r and each server s, define the *guardrail work counter* G_s^r. When the dispatcher dispatches a job of size x to server s, the counter G_s^r increases by x, where r is the rank of x.

A dispatching policy satisfies guardrails with width c and tightness g if at all times,

$$|G_s^r - G_{s'}^r| \leq gc^{r+1}$$

for all ranks r and all pairs of servers s and s', where the tightness is some constant $g \geq 1$.

Let GP be any guardrails policy with constant $g \geq 1$ and width

$$c = 1 + \frac{1}{1 + \ln \frac{1}{1-\rho}}$$

I prove that the dispatching policy GP, combined with SRPT scheduling at the servers, achieves asymptotically optimal mean response time:

Theorem 2.6 (Guardrails optimality [20])**.** *In a dispatching M/G/k with job size distribution S, if $E[S^2(\log S)^+] < \infty$,*

$$\lim_{\rho\to 1} \frac{E[T^{GP/SRPT}]}{E[T^{SRPT\text{-}1}]} = \lim_{\rho\to 1} \frac{E[T^{GP/SRPT}]}{E[T^{OPT}]} = 1$$

In order to prove Theorem 2.6, I also prove a bound on mean response time, which requires additional machinery to state.

2.4 Progress on results

Perfect information M/G/k: 100% complete. I prove Theorems 2.1 to 2.3 in my paper "SRPT for Multiserver Systems" [19].

Blind and partial information M/G/k: 100% complete. I prove Theorems 2.4 and 2.5 in my paper "The Gittins Policy is Nearly Optimal in the M/G/k under Extremely General Conditions" [42]. Previous to that paper, I also made progress on the blind setting in my paper "Optimal Multiserver Scheduling with Unknown Job Sizes in Heavy Traffic" [44].

Dispatching M/G/k with perfect information: 100% complete. I prove Theorem 2.6 and a corresponding bound on mean response time in my paper "Load Balancing Guardrails" [20].

3 First Analysis of Concurrent Service Models

In modern computer systems, one job, comprising one increment of useful work, is often served concurrently across many servers, cores, or other resources. This is in contrast to the most common multiserver queueing models, such as the M/G/k, which assume that each job requires a single server.

There are multiple important types of parallel workloads that occur commonly in modern systems. One type is *static parallelism*, where each job specifies the number of servers that it requires in order to run. Such jobs are referred to as "multiserver jobs", because a single job can span many servers. This type of parallelism is common in supercomputing and datacenter workloads.

Another important type of parallelism is *dynamic parallelism*, where the number of servers allocated to each job is chosen by the scheduler in response to the state of the system. This commonly occurs in databases and other query handling systems.

To better understand the performance of modern computer systems, I develop new models to capture concurrent workloads, and study optimal scheduling in those settings.

3.1 Setting

To capture the concepts of static and dynamic parallelism, I study two queueing models: The *threshold parallelism* model, and the *multiserver job* model.

In the threshold parallelism model, each job has two characteristics: a size S and a parallelism threshold P, sampled i.i.d. from some joint distribution (S, P). At any given point in time, the scheduling policy selects a set of jobs to serve, as well as how many servers to allocate to each job. If a job j has parallelism threshold p_j, then it can be allocated at most p_j servers. The total

number of servers allocated is at most k, the total number of servers. I assume that the number of servers allocated to a job can be increased or decreased with no loss of work.

When a job is served on $m \leq p_j$ servers, it completes at rate m/k, exhibiting perfect speedup from additional servers up to the threshold p_j. Specifically, the job's age, which represents the amount of service that has been performed on the job, increases at rate m/k whenever the job is in service on m servers. A job completes when its age reaches its size. Because each job runs at rate $1/k$ per server, the total capacity of the system is 1.

In the multiserver job model, a job again has two characteristics: a size S and a "number of servers" *requirement* V. At any given time, the scheduling policy selects a set of jobs to serve. The number of servers required by the jobs in service must total to at most k, the total number of servers. If a job j requires v_j servers, it completes at rate v_j/k whenever it is in service.

There are many important scheduling policies in each of these settings. In the threshold parallelism setting, natural policies include FCFS [21] and Inelastic/Elastic First [7]. In the multiserver-job setting, a natural baseline policy is FCFS, where jobs are placed into service in strict arrival order, blocking the queue if a job does not fit. Backfilling policies are often used in practice, where jobs are initially placed into service in arrival order, but when a job is reached that does not fit, further jobs from the queue are considered [14, 46]. An important characteristic of a multiserver-job policy is its stability region, the set of arrival rates that the policy can accommodate without the system become unstable and having jobs back up infinitely.

The best possible stability region, which I refer to as "full stability region", is achieved if the system is stable for all arrival rates λ such that $\lambda E[S] = \rho < 1$.

3.2 Problem statements

In the threshold parallelism and multiserver-job settings, there are no scheduling policies for which the response time behavior is understood. Given the state-of-the-art in the M/G/k setting in Section 2, exact characterization is out of reach, but bounds might be achievable. This leads to my first question:

- In the threshold parallelism and multiserver-job settings, what are scheduling policies for which I can derive tight analytical bounds on mean response time?
- Specifically in the multiserver-job setting, are there policies with full stability region for which I can prove tight bounds on mean response time?

Next, I turn to the problem of asymptotically optimal scheduling, in the heavy-traffic limit. In the threshold parallelism setting, results transfer essentially immediately from the M/G/k setting studied in Section 2. In contrast, in the multiserver-job setting, optimality is not so simple. Greedily prioritizing jobs by remaining size will lead to underutilization and poor response time or instability. Therefore, I ask:

- In the multiserver-job setting, what scheduling policy minimizes mean response time in the heavy-traffic limit?

The techniques that I employ to answer the above questions only apply to policies with full stability region, leaving open the behavior of many important scheduling policies in the multiserver-job setting such as FCFS and limited backfilling

- In the multiserver-job setting, what is the mean response time behavior of FCFS and Limited Backfilling?

3.3 Prior work

Here I present prior work on the threshold parallelism and multiserver-job models. For more, see the relevant section of my paper on the topic [21].

3.3.1 Threshold Parallelism

Berg et al. [6] introduce the concept of "speedup functions", in which different jobs achieve different benefits from running on increased numbers of servers. The threshold parallelism model can be seen as an important special case of that setting. The Elastic/Inelastic model [7] is a further special case of the threshold parallelism model, where each job either has a threshold of 1 or of k servers, and job sizes are exponential within each class. In the Elastic/Inelastic setting, optimality results and response time bounds are known for the Elastic First and Inelastic First policies [7]. In the more general threshold parallelism model, bounds on response time were unknown for any policy, and optimality was entirely open prior to this thesis.

3.3.2 Multiserver-job Model

The multiserver-job model has been a major focus of study, in both practical [12, 14, 46] and theoretical contexts [11, 16, 26, 31, 36, 38]. Even characterizing the stability region of scheduling policy is a challenging problem, and bounds on response time were entirely open.

A major focus of study has been FCFS scheduling, for which the stability region has only been characterized in a few special cases [11, 38]. One useful technique for characterizing the stability region is the saturated system approach, as explained in Section 3.1 [4, 15]. I further leverage the saturated system approach to present the first results on mean response time in this setting.

Beyond FCFS scheduling, most research has focused on throughput-optimal scheduling, which achieve the largest possible stability region [16, 31, 36]. No results on mean response time were known for any such policy, prior to my work.

3.4 Main Results

I start by proving bounds on mean response time for scheduling policies in the threshold parallelism and multiserver-job settings by introducing and analyzing

the "work-conserving finite-skip" (WCFS) class of models [21]. By introducing the WCFS class of models, which includes scheduling policies for the threshold parallelism and multiserver-job settings, I prove the first bounds on mean response time for any scheduling policies in these settings.

3.4.1 WCFS Class of Models

The work-conserving finite-skip (WCFS) class of models consists of queueing systems with the following two properties:

Finite skip: The system only serves jobs among the n oldest jobs in arrival order, where n is a finite parameter.

Work conserving: If at least n jobs are present in the system, the system's total service rate is equal to the system's maximum possible service rate, which I normalize to 1.

The WCFS class includes scheduling policies for both the threshold parallelism and multiserver-job models. In the threshold parallelism system, the FCFS policy results in a WCFS model. In the multiserver-job model, I introduce a novel scheduling policy called ServerFilling which results in a WCFS model in the special case where all server requirements v_j are powers of 2 and k is a power of 2, as well as several generalizations, including DivisorFilling [21], which merely assumes that all server requirements v_j divide the number of servers k.

I prove additively-tight bounds on mean response time for all WCFS models, showing that mean response time is within an additive margin of mean response time in the M/G/1. This bound holds as long as the job size distribution S has bounded expected remaining size, an assumption akin to sup-exponential tail [21].

Theorem 3.1 (WCFS mean response time bound [21]). *For any model $\pi \in$ WCFS, if S has bounded expected remaining size,*

$$E[T^{\pi}] - E[T^{M/G/1}] \in [c^{\pi}_{lower}, c^{\pi}_{upper}]$$

for explicit constants c^{π}_{lower}, c^{π}_{upper} not dependent on load ρ.

This gives the first tight bounds on mean response time for any scheduling policies for the threshold-parallelism or multiserver-job systems.

3.4.2 Optimal multiserver-job scheduling

Next, I turn to optimal scheduling in the multiserver-job model. In doing so, I seek to leverage my results in the M/G/k from Section 2. I therefore seek to define analogues of SRPT and Gittins for the multiserver-job model.

Recall that in the definition of WCFS, "finite-skip" is defined with reference to the arrival ordering. Let us define an analogue of finite-skip with reference

to a different job ordering, and correspondingly an analogue of WCFS with reference to that different job ordering.

Specifically, let us use the job ordering given by a general index policy P. An index policy assigns a priority to each job based on its age and other characteristics, and serves the job of best priority. Many common M/G/1 policies, including SRPT and Gittins, are index policies.

To define the WCFS-P class of models, I keep the definition of work conservation the same, but replace the "finite skip" requirement with a "finite skip P" requirement:

Finite skip P: The system only serves jobs among the n jobs of best priority, prioritizing according to policy P.

Given any WCFS policy, such as ServerFilling or DivisorFilling for the multiserver-job system, one can straightforwardly create the corresponding WCFS-P policy.

Now, I can state my optimality results. I show that WCFS-SRPT policies yield asymptotically optimal mean response time in known size settings, and WCFS-Gittins policies yield asymptotically optimal mean response time in blind or partially known size settings.

Theorem 3.2 (WCFS-SRPT optimality)**.** *For any model $\pi \in$ WCFS-SRPT, in a known-size setting, if $E[S^2(\log S)^+] < \infty$,*

$$\lim_{\rho\to 1} \frac{E[T^\pi]}{E[T^{SRPT\text{-}1}]} = \lim_{\rho\to 1} \frac{E[T^\pi]}{E[T^{OPT}]} = 1$$

Theorem 3.3 (WCFS-Gittins optimality)**.** *For any model $\pi \in$ WCFS-Gittins, in any job information setting, if $E[S^2(\log S)^+] < \infty$,*

$$\lim_{\rho\to 1} \frac{E[T^\pi]}{E[T^{Gittins\text{-}1}]} = \lim_{\rho\to 1} \frac{E[T^\pi]}{E[T^{OPT}]} = 1$$

In particular, ServerFilling-SRPT and related policies yield asymptotically optimal mean response time in the multiserver-job model.

3.4.3 Variable work rate finite-skip models

A key element of the WCFS approach from Section 3.4.1 is the idea of work conservation: If enough jobs are present, the system completes work at its maximum possible rate. However, there are many natural policies for which this is not the case, and models for which this is unachievable. For instance, consider the multiserver-job model in a setting where preemption is disallowed. No finite-skip policies are work conserving in this setting. Correspondingly, many common nonpreemptive multiserver-job scheduling policies, such as FCFS and backfilling-based policies, are not work-conserving.

I therefore seek to understand the response time of variable work rate finite-skip models, to better understand the performance of concurrent service systems.

To do so, I step back, and consider a general finite-skip system. A powerful framework for analyzing such a system is the saturated system [4, 15], a system in which an infinite number of jobs in the queue are always available, and only the state of the n jobs at the front of the queue are tracked.

An important characteristic of the saturated system is the intercompletion distribution Z. Note that adjacent intercompletion times are correlated: If a completion occurs from a low-completion-rate state, with few jobs in service, the next state is likely to also have low completion rate. To capture this correlation, let $Z_i(x)$ be a random variable measuring the ith intercompletion time starting from state x.

The stability region of the original open system is known to be closely related to Z: The system is stable if $\lambda E[Z] < 1$ [4, 15].

I seek to go one step further, and characterize the asymptotic behavior of mean response time in the limit as λ approaches $1/E[Z]$, the edge of stability. This limit is the heavy-traffic limit in this setting. Call $\rho^* = \lambda E[Z]$ the modified load of the system, and let us examine the $\rho^* \to 1$ limit.

To do so, I define $d(x)$, the *transient completion delay* starting from a state x, as follows:

$$d(x) = \lim_{i \to \infty} \sum_{j=1}^{i} (E[Z_i(x)] - E[Z])$$

Intuitively, the expression inside the limit measures how much faster or slower i completions occur after starting in state x, as compared to the long-term average completion rate. If the saturated system has a finite state-space, one can show that $d(x)$ is well-defined for all states x. This setting is general enough to model phase-type job-duration distributions, for instance.

I am now ready to state my result, which gives the first analysis of mean response time for the general finite skip system. Let X_{sat}^{time} be the time-average state of the saturated system.

Theorem 3.4 (General finite skip mean response time)**.** *For any finite skip system whose saturated system is described by a finite state Markov chain,*

$$E[T] = \frac{d(X_{sat}^{time}) + E[Z]}{1 - \rho^*} + o\left(\left(\frac{1}{1-\rho^*}\right)^{\epsilon}\right)$$

for all $\epsilon > 0$.

3.5 Progress on results

WCFS: 100% complete. I prove Theorem 3.1 in "The Finite-Skip Method for Multiserver Analysis", currently under submission [21].

Optimal multiserver-job scheduling: 50% complete. I have proof sketches of Theorem 3.2 and Theorem 3.3, combining my Gittins-k paper [42] with my WCFS paper [21], but I have not written up this proof in detail.

Variable work rate models: 50% complete. I have written up the proof of Theorem 3.4, but it is currently at first-draft quality.

4 First Tail Optimization in Single Server Models

Traditional queueing analysis often focuses on optimizing mean response time. But in modern computer systems, other metrics of response time typically take on more importance. The most important metrics tend to measure *tail* performance, the response time of the worst performing jobs. Such metrics are common when evaluating web servers, databases, and datacenters. As a result, I seek to optimize the tail of response time in queueing systems.

4.1 Setting

When studying the tail of response time, I focus on a single-server system, specifically the M/G/1. Optimizing the tail of response time is already an open problem in the simpler single-server system, to say nothing of multiserver systems. I choose to focus on the simpler system as a starting point.

There are a wide variety of important tail metrics of response time, such as $E[T^2]$, the second moment of response time, and T^{99}, the 99^{th} percentile of response time. While these response time metrics are very important in practice, they are not particularly amenable to theoretical analysis.

One tail metric that is often studied as a compromise between practical usefulness and theoretical tractability is the *asymptotic tail* of response time. In other words, this is the asymptotic behavior of the function $\bar{F}_T(t) = P(T > t)$, in the limit $t \to \infty$. The asymptotic tail of response time is well understood for many important scheduling policies, including FCFS, PS (Processor Sharing), SRPT, and more [9, 35, 43].

When it comes to optimizing the asymptotic tail of response time, there are two important kinds of optimality to consider: *weak optimality* and *strong optimality*.

For a scheduling policy π^* to have weakly optimal tail of response time, it most be the case that for all scheduling policies π,

$$\limsup_{t\to\infty} \frac{P(T^{\pi^*} > t)}{P(T^{\pi} > t)} \leq c$$

for some constant $c \in [1, \infty)$. If $c = 1$, then π^* is called strongly optimal.

Which policies are weakly optimal depends on the job size distribution. For light-tailed job size distributions, FCFS is known to be weakly optimal [9]. To be precise, FCFS is weakly optimal for class I job size distributions, which I define below. For heavy-tailed job size distributions, policies such as PS and SRPT are known to be weakly optimal [9, 35, 43]. In contrast, the problem of strongly optimizing the tail of response time remains open. In the setting of

light-tailed job size distributions, FCFS is conjectured to be strongly optimal [50], but no results beyond weak optimality are known.

On finite arrival sequences, FCFS minimizes the maximum response time, and more generally minimizes the sorted vector of response times under the lexicographic ordering. This property lends credence to the belief that FCFS may be strongly optimal, but it is not a proof.

Beyond the asymptotic tail of response time, little is known about optimality. Non-asymptotic tail metrics are typically too poorly behaved to prove clean theoretical results. However, one way to overcome this is via the concept of *Stochastic Dominance*.

For two scheduling policies π_1 and π_2, I say that π_1 stochastically dominates π_2 if, for any response time cutoff t,

$$P(T^{\pi_1} > t) < P(T^{\pi_2} > t) \quad \forall t$$

Intuitively, if π_1 stochastically dominates π_2, then π_1 outperforms π_2 for every reasonable metric of response time. For a policy to be optimal with respect to any metric of response time, it must not be stochastically dominated by any other policy. Such a policy is called "Pareto optimal". The only known Pareto optimal scheduling policies is SRPT, which cannot be stochastically dominated because SRPT has optimal mean response time.

Short of optimizing the response time distribution, an important intermediate step is to analytically quantify a scheduling policy's response time distribution. A common technique for doing so is to characterize the Laplace-Stieltjes transform of a scheduling policies' response time distribution. The transform of a random variable X is written $\widetilde{X}(s)$, and is defined as:

$$\widetilde{X}(s) = \int_0^\infty e^{-st} f_X(t)dt = E[e^{-sX}]$$

From the transform of response time, important tail metrics such as the asymptotic tail behavior and integer moments can be extracted.

Using the transform of the job size distribution, one can precisely define a Class I job size distribution [1, 2, 39]. For a job size distribution S, let $-s^*$ be the rightmost singularity of $\widetilde{S}(s)$, with $-s^* = \infty$ if $\widetilde{S}(s)$ is analytic everywhere. S is a class I distribution if $s^* > 0$ and $\widetilde{S}(s) = \infty$. Intuitively, class I distributions are well-behaved light-tailed distributions.

One important scheduling policy for achieving optimal or near-optimal non-asymptotic tail response time is the Accumulating Priority policy. Under this policy, each job has a priority accumulation rate b, and the priority of a job is equal to $b(t - t_A)$, where t_A is the job's arrival time and t is the current time. The job with the highest priority is served.

The transform of response time is known under the Accumulating Priority policy with a finite set of priority accumulation rates b and nonpreemptive service [47]. Generalizations handling preemptive service [13] and affine priority functions [8, 34] have also been studied. However, note that prior work has focused on the case of jobs with inherent priority, with b unrelated to size.

By instead choosing b based on the size of a job, very good tail performance may be possible, perhaps achieving better tail performance than any previously analyzed policy. Previous analysis frameworks, such as SOAP, require that a job's priority does not change when the job is not in service. To optimize tail performance, it is critical that a freshly arrived job not have the same priority as a job that arrived long ago. Accumulating Priority-based policies are among the few analyzable policies that prioritize based on time in system.

In this section, prior work has been referenced inline with the setting. For more, see [22], my paper on the topic.

4.2 Problem

As mentioned in Section 4.1, FCFS has weakly optimal asymptotic tail of response time under light-tailed job size distributions. This leads to two natural questions about optimality:

- Does FCFS have strongly optimal asymptotic tail of response time? If not, what policy provides a counterexample?
- Does FCFS have Pareto optimal mean response time? If not, what policy stochastically dominates it?

Accumulating Priority is an important scheduling policy for the tail of response time, because it prioritizes jobs that have been in the system for a long time. By choosing job j's accumulation rate b_j based on the job's size s_j, one can try to achieve near-optimal performance for the tail of response time. For instance, to optimize $E[T^2]$, a natural choice is to set $b_j = 1/s_j$. Such a policy is a Continuous Accumulating Priority policy, which has not yet been analyzed.

- What is Laplace-Stieltjes transform of response time under a Continuous Accumulating Priority policy?

4.3 Main Results

4.3.1 Nudge

I show that FCFS does not have strongly optimal asymptotic tail of response time under light-tailed job size distributions by exhibiting an explicit counterexample, a scheduling policy that I call Nudge. Moreover, I show that Nudge stochastically dominates FCFS, showing that FCFS is not even Pareto optimal.

To define the Nudge scheduling policy, I divide jobs into three types: Small, Large and Other. The nudge is parameterized by three size cutoffs: x_1, x_2, x_3. If a job has size smaller than x_1, the job is small. If a job has size in the interval $[x_2, x_3]$, the job is large. Otherwise, a job is "other".

The Nudge policy functions by maintaining an ordering over jobs which have not yet entered service, which I call the queue. Whenever a job completes, Nudge serves the job at the front of the queue. When a job arrives, it is by default placed at the back of the queue, resulting in default FCFS scheduling.

However, under certain conditions, the arriving job is placed second-to-last in the queue, swapping it with the last job in the queue. A swap is performed if:

- The arriving job is a small job,
- the job at the back of the queue is a large job, and
- the job at the back of the queue has never been swapped.

With Nudge defined, I can state my results, both on the asymptotic tail of response time, and on stochastic dominance.

Theorem 4.1 (Nudge asymptotic improvement [22]). *For any job size distribution S that is continuous, class I, and has support down to 0, for any load $\rho < 1$, there exist parameters x_1, x_2, x_3 such that Nudge has multiplicatively smaller asymptotic tail of response time than FCFS:*

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

As a result, FCFS is not strongly optimal.

Theorem 4.2 (Nudge stochastic dominance [22]). *For any job size distribution S that is continuous, class I, and has support down to 0, for any load $\rho < 1$, there exist parameters x_1, x_2, x_3 such that Nudge stochastically dominates FCFS:*

$$P(T^{Nudge} > t) < P(T^{FCFS} > t) \quad \forall t$$

As a result, FCFS is not Pareto optimal.

4.3.2 Continuous Accumulating Priority

The Continuous Accumulating Priority (CAP) policy is a natural generalization of the class-based Accumulating Priority (AP) model previously studied [47]. In particular, CAP is a natural choice for size-based AP policies.

I plan to generalize the nonpreemptive class-based AP analysis to the CAP setting. While several of the steps in the nonpreemptive analysis generalize cleanly, some parts of the analysis fundamentally rely on having a finite set of priority accumulation rates b. I believe that generalizing these steps to the CAP setting may result in the transform of response time being represented as the solution to a differential equation. Hopefully, for at least some job size distributions, this differential equation will be solvable, allowing the full response time behavior of the CAP policy to be analyzed.

4.4 Progress so far

Nudge: 100% complete. I prove Theorem 4.1 and Theorem 4.2 in my paper "Nudge: Stochastically Improving upon FCFS" [22].

Continuous Accumulating Priority: 20% complete. Progress on this analysis is just beginning, but early indications are that it seems doable.

5 Timeline

Here I present my proposed schedule for completing the research laid out in the proposal, submitting the work for publication, and writing up that work into my thesis document.

Before April 2022:

- Complete and publish SRPT-k, Gittins-k, and Guardrails results (Section 2).
- Complete and submit WCFS results (Section 3.4.1).
- Complete research on response time in variable work rate finite-skip models (Section 3.4.3).
- Complete and publish Nudge results (Section 4.3.1).

May-July 2022:

- Complete and submit research on optimal multiserver-job scheduling (Section 3.4.2), hopefully to the summer deadline for SIGMETRICS 2023.
- Write up and submit research on variable work rate finite-skip models (Sections 3.4.2 and 3.4.3) hopefully to the summer deadline for SIGMETRICS 2023.
- Progress research on size-based continuous accumulating priority scheduling (Section 4.3.2).
- Begin writing thesis chapters for SRPT-k, Gittins-k, Guardrails, Nudge (Sections 2 and 4.3.1)

August-October 2022:

- Complete and submit research on size-based continuous accumulating priority scheduling (Section 4.3.2).
- Complete thesis chapters on SRPT-k, Gittins-k, Guardrails, Nudge (Sections 2 and 4.3.1).
- Begin thesis chapters on concurrent job models (Section 3).

November 2022-January 2023:

- Complete thesis chapters on concurrent job models (Section 3).
- Write thesis chapter on size-based continuous accumulating priority scheduling (Section 4.3.2).

February-April 2023: Nothing important I have ever worked on has gone to schedule. Consider the schedule above a "best-case scenario", with the final months reserved for deadlines missed, rescheduling, and similar unplanned but inevitable contingencies. As the months go on, I will prepare revised schedules that move deadlines back when necessary, using these final months to make that possible without delaying my defense.

References

[1] Joseph Abate, Gagan L. Choudhury, and Ward Whitt. Asymptotics for steady-state tail probabilities in structured Markov queueing models. *Stochastic Models*, 10(1):99–143, 1994.

[2] Joseph Abate, Gagan L Choudhury, and Ward Whitt. Waiting-time tail probabilities in queues with long-tail service-time distributions. *Queueing systems*, 16(3-4):311–338, 1994.

[3] Nir Avrahami and Yossi Azar. Minimizing total flow time and total completion time with immediate dispatching. In *Proceedings of the Fifteenth Annual ACM Symposium on Parallel Algorithms and Architectures*, SPAA '03, pages 11–18, New York, NY, USA, 2003. ACM. ISBN 1-58113-661-7. doi: 10.1145/777412.777415.

[4] François Baccelli and Serguei Foss. On the saturation rule for the stability of queues. *Journal of Applied Probability*, 32(2):494–507, 1995. doi: 10.2307/3215303.

[5] Eitan Bachmat and Hagit Sarfati. Analysis of size interval task assignment policies. *SIGMETRICS Perform. Eval. Rev.*, 36(2):107–109, August 2008. ISSN 0163-5999. doi: 10.1145/1453175.1453199.

[6] Benjamin Berg, Jan-Pieter Dorsman, and Mor Harchol-Balter. Towards optimality in parallel scheduling. *Proc. ACM Meas. Anal. Comput. Syst.*, 1(2), December 2017. doi: 10.1145/3154499.

[7] Benjamin Berg, Mor Harchol-Balter, Benjamin Moseley, Weina Wang, and Justin Whitehouse. Optimal resource allocation for elastic and inelastic jobs. In *Proceedings of the 32nd ACM Symposium on Parallelism in Algorithms and Architectures*, pages 75–87, 2020.

[8] Blair Bilodeau and David A Stanford. High-priority expected waiting times in the delayed accumulating priority queue with applications to health care kpis. *arXiv preprint arXiv:2001.06054*, 2020.

[9] Onno J. Boxma and Bert Zwart. Tails in scheduling. *SIGMETRICS Performance Evaluation Review*, 34(4):13–20, 2007.

[10] Maury Bramson, Yi Lu, and Balaji Prabhakar. Asymptotic independence of queues under randomized load balancing. *Queueing Systems*, 71(3):247–292, Jul 2012. ISSN 1572-9443. doi: 10.1007/s11134-012-9311-0.

[11] Percy H Brill and Linda Green. Queues in which customers receive simultaneous service from a random number of servers: a system point approach. *Management Science*, 30(1):51–68, 1984.

[12] Danilo Carastan-Santos, Raphael Y. De Camargo, Denis Trystram, and Salah Zrigui. One can only gain by replacing EASY backfilling: A simple scheduling policies case study. In *2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 1–10, 2019. doi: 10.1109/CCGRID.2019.00010.

[13] Val Andrei Fajardo and Steve Drekic. Waiting time distributions in the preemptive accumulating priority queue. *Methodology and Computing in Applied Probability*, 19(1):255–284, 2017.

[14] Dror G. Feitelson, Larry Rudolph, and Uwe Schwiegelshohn. Parallel job scheduling—a status report. In *Workshop on Job Scheduling Strategies for Parallel Processing*, pages 1–16, New York, NY, USA, 2004. Springer.

[15] Serguei Foss and Takis Konstantopoulos. An overview of some stochastic stability methods. *Journal of the Operations Research Society of Japan*, 47 (4):275–303, 2004.

[16] Javad 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, April 2016. doi: 10.1109/INFOCOM.2016.7524536.

[17] John Gittins, Kevin Glazebrook, and Richard Weber. *Multi-armed bandit allocation indices*. John Wiley & Sons, 2011.

[18] Kevin D. Glazebrook and José Niño-Mora. Parallel scheduling of multiclass M/M/m queues: Approximate and heavy-traffic optimization of achievable performance. *Operations Research*, 49(4):609–623, 2001.

[19] Isaac Grosof, Ziv Scully, and Mor Harchol-Balter. SRPT for multiserver systems. *Performance Evaluation*, 127-128:154–175, 2018. ISSN 0166-5316. doi: https://doi.org/10.1016/j.peva.2018.10.001.

[20] Isaac Grosof, Ziv Scully, and Mor 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. doi: 10.1145/3341617.3326157.

[21] Isaac Grosof, Mor Harchol-Balter, and Alan Scheller-Wolf. The finite-skip method for multiserver analysis. *arXiv preprint arXiv:2109.12663*, 2021.

[22] Isaac Grosof, Kunhe Yang, Ziv Scully, and Mor 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. ISBN 9781450380720. doi: 10.1145/3410220.3460102.

[23] Mor Harchol-Balter. Open problems in queueing theory inspired by datacenter computing. *Queueing Systems: Theory and Applications*, 97(1): 3–37, 2021.

[24] Mor Harchol-Balter, Mark E. Crovella, and Cristina D. Murta. On choosing a task assignment policy for a distributed server system. *Journal of Parallel and Distributed Computing*, 59(2):204–228, 1999. ISSN 0743-7315. doi: https://doi.org/10.1006/jpdc.1999.1577.

[25] Mor Harchol-Balter, Takayuki Osogami, Alan Scheller-Wolf, and Adam Wierman. Multi-server queueing systems with multiple priority classes. *Queueing Systems*, 51(3):331–360, Dec 2005. ISSN 1572-9443. doi: 10.1007/s11134-005-2898-7.

[26] Yige Hong. Sharp zero-queueing bounds for multi-server jobs. *SIGMETRICS Perform. Eval. Rev.*, 49(2):66–68, jan 2022. ISSN 0163-5999. doi: 10.1145/3512798.3512822.

[27] Julian Köllerström. Heavy traffic theory for queues with several servers. I. *Journal of Applied Probability*, 11(3):544–552, 1974. doi: 10.2307/3212698.

[28] Julian Köllerström. Heavy traffic theory for queues with several servers. II. *Journal of Applied Probability*, 16(2):393–401, 1979. doi: 10.2307/3212906.

[29] Stefano Leonardi and Danny Raz. Approximating total flow time on parallel machines. *Journal of Computer and System Sciences*, 73(6):875 – 891, 2007. ISSN 0022-0000. doi: https://doi.org/10.1016/j.jcss.2006.10.018.

[30] Richard Loulou. Multi-channel queues in heavy traffic. *Journal of Applied Probability*, 10(4):769–777, 1973. doi: 10.2307/3212380.

[31] Siva Theja Maguluri, Rayadurgam Srikant, and Lei Ying. Stochastic models of load balancing and scheduling in cloud computing clusters. In *2012 Proceedings IEEE Infocom*, pages 702–710. IEEE, 2012.

[32] I. Mitrani and P.J.B. King. Multiprocessor systems with preemptive priorities. *Performance Evaluation*, 1(2):118 – 125, 1981. ISSN 0166-5316. doi: https://doi.org/10.1016/0166-5316(81)90014-6.

[33] Jeffrey C. Mogul and John Wilkes. Nines are not enough: Meaningful metrics for clouds. In *Proceedings of the Workshop on Hot Topics in Operating Systems (HotOS19)*, pages 136 – 141, USA, 2019.

[34] Maryam Mojalal, David A Stanford, and Richard J Caron. The lower-class waiting time distribution in the delayed accumulating priority queue. *INFOR: Information Systems and Operational Research*, 58(1):60–86, 2020.

[35] Misja Nuyens, Adam Wierman, and Bert Zwart. Preventing large sojourn times using smart scheduling. *Operations Research*, 56(1):88–101, 2008.

[36] Konstantinos Psychas and Javad Ghaderi. Randomized algorithms for scheduling multi-resource jobs in the cloud. *IEEE/ACM Transactions on Networking*, 26(5):2202–2215, 2018. doi: 10.1109/TNET.2018.2863647.

[37] Rhonda Righter and J. George Shanthikumar. Scheduling multiclass single server queueing systems to stochastically maximize the number of successful departures. *Probability in the Engineering and Informational Sciences*, 3 (3):323–333, 1989. doi: 10.1017/S0269964800001194.

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

[39] T. Sakurai. Approximating m/g/1 waiting time tail probabilities. *Stochastic Models*, 20(2):173–191, 2004.

[40] Linus E. Schrage and Louis W. Miller. The queue M/G/1 with the Shortest Remaining Processing Time discipline. *Operations Research*, 14(4):670–684, 1966. doi: 10.1287/opre.14.4.670. URL `https://doi.org/10.1287/opre.14.4.670`.

[41] Ziv Scully and Mor Harchol-Balter. The gittins policy in the M/G/1 queue. In *2021 19th International Symposium on Modeling and Optimization in Mobile, Ad hoc, and Wireless Networks (WiOpt)*, pages 1–8, 2021. doi: 10.23919/WiOpt52861.2021.9589051.

[42] Ziv Scully, Isaac Grosof, and Mor 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. doi: 10.1145/3428328.

[43] Ziv Scully, Lucas van Kreveld, Onno J. Boxma, Jan-Pieter Dorsman, and Adam Wierman. Characterizing policies with optimal response time tails under heavy-tailed job sizes. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 4(2), June 2020. ISSN 2476-1249, 2476-1249.

[44] Ziv Scully, Isaac Grosof, and Mor Harchol-Balter. Optimal multiserver scheduling with unknown job sizes in heavy traffic. *Performance Evaluation*, 145:102150, 2021. ISSN 0166-5316. doi: https://doi.org/10.1016/j.peva.2020.102150.

[45] Andrei Sleptchenko, Aart van Harten, and Matthieu van der Heijden. An exact solution for the state probabilities of the multi-class, multi-server queue with preemptive priorities. *Queueing Systems*, 50(1):81–107, May 2005. ISSN 1572-9443. doi: 10.1007/s11134-005-0359-y.

[46] S. Srinivasan, R. Kettimuthu, V. Subramani, and P. Sadayappan. Characterization of backfilling strategies for parallel job scheduling. In *Proceedings. International Conference on Parallel Processing Workshop*, pages 514–519, 2002. doi: 10.1109/ICPPW.2002.1039773.

[47] David A Stanford, Peter Taylor, and Ilze Ziedins. Waiting time distributions in the accumulating priority queue. *Queueing Systems*, 77(3):297–330, 2014.

[48] Muhammad Tirmazi, Adam Barker, Nan Deng, Md E. Haque, Zhijing Gene Qin, Steven Hand, Mor Harchol-Balter, and John Wilkes. Borg: The next generation. In *Proceedings of the Fifteenth European Conference on Computer Systems*, EuroSys '20, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450368827. doi: 10.1145/3342195.3387517.

[49] Richard R. Weber. On the optimal assignment of customers to parallel servers. *Journal of Applied Probability*, 15(2):406–413, 1978. doi: 10.2307/3213411.

[50] Adam Wierman and Bert Zwart. Is tail-optimal scheduling possible? *Operations Research*, 60(5):1249–1257, October 2012. ISSN 0030-364X, 1526-5463.

[51] Wayne Winston. Optimality of the shortest line discipline. *Journal of Applied Probability*, 14(1):181–189, 1977. doi: 10.2307/3213271.