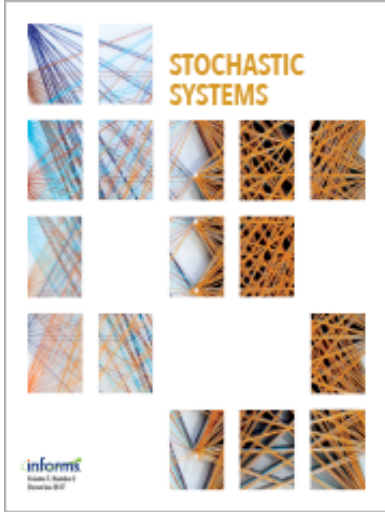




Stochastic Systems

Publication details, including instructions for authors and subscription information:
<http://pubsonline.informs.org>

Asymptotically Optimal Scheduling of Multiple Parallelizable Job Classes

Benjamin Berg, Benjamin Moseley, Weina Wang, Mor Harchol-Balter

To cite this article:

Benjamin Berg, Benjamin Moseley, Weina Wang, Mor Harchol-Balter (2026) Asymptotically Optimal Scheduling of Multiple Parallelizable Job Classes. *Stochastic Systems*

Published online in Articles in Advance 05 Oct 2026

. <https://doi.org/10.1287/stsy.2024.0084>

This work is licensed under a Creative Commons Attribution 4.0 International License. You are free to copy, distribute, transmit and adapt this work, but you must attribute this work as “*Stochastic Systems*. Copyright © 2026 The Author(s). <https://doi.org/10.1287/stsy.2024.0084>, used under a Creative Commons Attribution License: <https://creativecommons.org/licenses/by/4.0/>.”

Copyright © 2026 The Author(s)

Please scroll down for article—it is on subsequent pages



With 12,500 members from nearly 90 countries, INFORMS is the largest international association of operations research (O.R.) and analytics professionals and students. INFORMS provides unique networking and learning opportunities for individual professionals, and organizations of all types and sizes, to better understand and use O.R. and analytics tools and methods to transform strategic visions and achieve better outcomes. For more information on INFORMS, its publications, membership, or meetings visit <http://www.informs.org>

Asymptotically Optimal Scheduling of Multiple Parallelizable Job Classes

Benjamin Berg,^{a,*} Benjamin Moseley,^b Weina Wang,^c Mor Harchol-Balter^c

^aDepartment of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, North Carolina 27599; ^bTepper School of Business, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213; ^cDepartment of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213

*Corresponding author

✉ ben@cs.unc.edu (B. Berg), moseleyb@andrew.cmu.edu (B. Moseley), weinaw@cs.cmu.edu (W. Wang), harchol@cs.cmu.edu (M. Harchol-Balter)

ORCID: <https://orcid.org/0000-0002-4147-6860> (B. Berg), <https://orcid.org/0000-0001-6808-0156> (W. Wang)

Received: July 18, 2024

Revised: June 4, 2025; December 17, 2025;
June 4, 2026

Accepted: July 24, 2026

Published Online in Articles in Advance:
October 5, 2026

<https://doi.org/10.1287/stsy.2024.0084>

Copyright: © 2026 The Author(s)

 **Open Access Statement:** This work is licensed under a Creative Commons Attribution 4.0 International License. You are free to copy, distribute, transmit and adapt this work, but you must attribute this work as “Stochastic Systems. Copyright © 2026 The Author(s). <https://doi.org/10.1287/stsy.2024.0084>, used under a Creative Commons Attribution License: <https://creativecommons.org/licenses/by/4.0/>.”

Abstract. Modern computing workloads are often composed of parallelizable jobs. A parallelizable job can be completed more quickly when run on additional servers. However, each job can only use a limited number of servers, known as its parallelizability level, which is determined by the type of computation the job performs and how it is implemented. Workloads generally consist of multiple job classes, where jobs from different classes have different parallelizability levels and follow different job size (service requirement) distributions. This paper considers scheduling parallelizable jobs belonging to an arbitrary number of job classes. Given a limited number of servers, we must allocate servers across a stream of arriving jobs to minimize mean response time—the average time from when a job arrives to the system until it completes. We find that in lighter-load scaling regimes (i.e., sub-Halfin-Whitt), the optimal allocation policy is least-parallelizable-first, which prioritizes jobs from the least parallelizable job classes regardless of their size distributions. By contrast, we find that in the heavier-load regimes (i.e., super-nondegenerate slowdown), the optimal allocation policy prioritizes jobs with the shortest expected remaining processing time. We also develop policies that are asymptotically optimal when the scaling regime is not known a priori.

Funding: Open Access funding was provided by the University of North Carolina at Chapel Hill. This work was supported by the National Science Foundation (NSF) [Grants NSF-CIF-2403194, NSF-CCF-2403195, NSF-III-2322973, NSF-IIS-2322974, and NSF-CMMI-2307008]. W. Wang is supported in part by the NSF [Grants ECCS-2145713, CCF-2428569, and ECCS-2432545].

Supplemental Material: The online appendix is available at <https://doi.org/10.1287/stsy.2024.0084>.

Keywords: parallel computing • scheduling • speedup functions • scaling regimes

1. Introduction

The vast majority of modern computer systems are designed to exploit parallelism. Whether considering a single multicore server, a compute cluster of many servers, or even an entire data center, achieving good system performance requires carefully choosing how to parallelize jobs across the underlying cores or servers in the system. For example, database queries for an in-memory database must be parallelized across the cores of a multicore server to achieve low query latency (Leis et al. 2014, Wagner et al. 2021), machine learning training jobs must be parallelized across the many GPUs of a compute cluster to reduce job completion time (Qiao et al. 2021, Jayaram Subramanya et al. 2023), and datacenter computing workloads must be parallelized across the many servers of a data center (Tumanov et al. 2016, Delgado et al. 2018). In each of these cases, the system operator is presented with a fixed set of cores or servers and asked to devise a strategy to allocate these resources across a stream of parallelizable jobs. However, devising optimal scheduling policies for parallelizable jobs in multiserver systems remains an open problem.

This paper develops asymptotically optimal scheduling policies for parallelizable jobs. Specifically, given a limited number of servers, k , we ask how to allocate these servers to a stream of parallelizable jobs arriving over time to minimize the mean response time across jobs.

If a workload consists solely of perfectly parallelizable jobs that can each fully utilize all k cores, the problem is easy because each job effectively sees one fast server that runs k times faster than a single server. Unfortunately, in practice, parallelizable jobs typically have limits to their level of parallelizability. For example, a database

query is compiled to run with a certain number of threads, t , where $t \ll k$ (recall k is the total number of servers; Chen et al. 2016). Allocating more than t cores to this query will not reduce its processing time. Similarly, a job's level of parallelism may be limited by overhead from communication between its threads. For example, machine learning training jobs can be parallelized across many servers, but these servers must synchronize at the end of each training iteration (Lin et al. 2018, Qiao et al. 2021). Additionally, some jobs consist of a mixture of parallelizable work and nonparallelizable work. This leads to the well-known phenomenon described as *Amdahl's law*, which says that the speedup a job can receive from parallelism is bounded in terms of the fraction of the job that is parallelizable (Hill and Marty 2008).

Each of these factors will affect the parallelizability of different jobs to different degrees. Some jobs in a workload may be highly parallelizable, whereas others may benefit only slightly from running on additional servers. This mixture of jobs with different levels of parallelizability is commonly seen in databases (Leis et al. 2014), machine learning workloads (Qiao et al. 2021), and datacenter scheduling (Tumanov et al. 2016). This paper develops new theoretical analyses of systems with parallelizable jobs of varying parallelizability levels.

1.1. High-Level Problem Statement

To capture the limited parallelism in today's parallel computing workloads, we approximate the behavior of different parallelizable jobs by assuming that jobs are perfectly parallelizable up to some *parallelizability limit*. This limit represents aspects of how the job was compiled, the communication overhead the job incurs, and what fraction of the job is parallelizable. Formally, we characterize each job by a pair (S, c) where S is a random variable denoting the job's *inherent size* and c is its parallelizability level. The time required to complete a parallelizable job is a function of both its inherent size and the servers allocated to it over time. When a job runs on $m \leq c$ servers, its remaining size decreases at a rate of m units of work per second until the remaining size reaches zero and the job is considered complete. The job cannot utilize more than c servers.

To capture the varying levels of parallelizability seen within a single workload, we allow jobs to belong to one of ℓ *job classes*. Each job class, i , has an associated job size distribution, S_i , and parallelism limit, c_i , that applies to every class i job. We assume that a job's exact size is unknown to the system, but its class is known; that is, the system knows (S_i, c_i) , for all i . Furthermore, we assume that jobs are *malleable*, meaning that a job can change the number of servers it runs on over time (Gupta et al. 2014).

Given a stream of malleable jobs that arrive to the system over time, a *scheduling policy* must determine how many servers to allocate to each job at every moment in time. Given a system with k homogeneous servers, our goal is to develop scheduling policies that minimize the *mean response time* across jobs—the average time from when a job arrives to the system until it is completed.

1.2. Prior Work

Scheduling in multiserver systems is known to be a difficult problem. There is a variety of work from the systems community on scheduling parallelizable jobs, including work on scheduling in databases (Leis et al. 2014, Wagner et al. 2021), machine learning clusters (Qiao et al. 2021, Jayaram Subramanya et al. 2023), data analytics clusters (Vernica et al. 2010, Zhu et al. 2014), and data centers (Tumanov et al. 2016, Delgado et al. 2018). This work uses simple policies with no formal guarantees. Additionally, many systems optimize for metrics such as fairness in addition to trying to minimize response times.

In the theoretical scheduling community, even when jobs are not parallelizable and each job occupies a single server, the optimal scheduling policy is not known. Recent results (Grosz et al. 2018) derived the first bounds on the mean response time of shortest-remaining-processing-time (SRPT) in multiserver systems, showing that SRPT is optimal in conventional heavy traffic (see Section 2). These heavy-traffic results were extended to a Gittins index policy (Scully et al. 2020a) in the case where job sizes are not fully known. Unfortunately, none of these results deal with parallelizable jobs.

Another related area of active research is scheduling for *multiserver jobs* (Grosz et al. 2020, Harchol-Balter 2022, Hong and Wang 2022). Multiserver jobs each have a hard, fixed resource requirement on the number of servers they need. Different jobs have different fixed server requirements. In general, it is not known how to minimize the mean response time for multiserver jobs, even in heavy traffic. Under some constraints on the server requirements of each job, an SRPT variant called ServerFilling-SRPT was recently shown to be optimal in conventional heavy traffic (Grosz et al. 2022a, b). Because jobs have hard resource requirements in the multiserver job model, one of the main concerns of a scheduling policy is to ensure that jobs are efficiently packed onto the available servers so that the system utilization remains high. Failing to pack the jobs efficiently can cause the system to be unstable. By contrast, the malleable jobs we consider do not have this stability concern, because they can adapt their levels of parallelism to ensure that the servers remain fully utilized. Hence, the policies developed for

scheduling multiserver jobs tend to prioritize packing in a way that has no benefit when scheduling malleable jobs.

The work closest to our model considers scheduling parallelizable jobs with *speedup functions* (Edmonds 1999; Berg et al. 2018, 2020a, b, 2022). Here, each job has an associated speedup function that determines the job’s service rate on different numbers of servers. Of these papers, Berg et al. (2018, 2020a) focus on the case where there is only one class of jobs that all follow the *same* speedup function. The case of two classes of jobs is considered in Berg et al. (2020b, 2022), but jobs are assumed to be either nonparallelizable or fully parallelizable, and the results require highly restrictive assumptions about the job size distributions. Little is known in the case where jobs have more than two levels of parallelizability.

Some further-afield work on parallel scheduling has used different models and objectives to design scheduling policies. Work on directed acyclic graph (DAG) scheduling has considered the problem of scheduling parallelizable tasks with precedence constraints that are specified by the edges of a DAG (Blumofe and Leiserson 1999, Agrawal et al. 2008, 2016). Much of this work considers the problem of scheduling a *single* DAG (Blumofe and Leiserson 1999, Agrawal et al. 2008). For scheduling a stream of DAGs, only pessimistic worst-case bounds are known (Agrawal et al. 2016).

1.3. Tradeoff

As we saw in Section 1.2, SRPT-like policies that prioritize short jobs are optimal in conventional heavy traffic for a variety of multiserver scheduling models. However, when the system is not so heavily loaded, it is not obvious that favoring short jobs suffices for optimality. In particular, when scheduling malleable jobs, there is another consideration that is at least as important: *deferring parallelizable work*. Here, a scheduling policy can raise the long-run average utilization of the system by prioritizing less parallelizable jobs ahead of more parallelizable jobs. The benefits of deferring parallelizable work were first described in Berg et al. (2018), which considers a less complex model with just two classes of jobs.

To see the benefit of deferring parallelizable work, suppose we have a job, A, that can only use one server and a job, B, that can parallelize across all k servers. If we prioritize job B, there will be a period after B completes where job A runs and $k - 1$ servers sit idle. To minimize mean response time, it is better to give one server to job A and use the remaining $k - 1$ servers to run job B. This holds regardless of the sizes of the jobs. By prioritizing the less parallelizable job A, we *defer* working on the more parallelizable job B.

This raises the question of how to handle the *tradeoff* between favoring jobs that are shorter versus deferring parallelizable work. What does a policy do with a job that has a small expected remaining size, but is also highly parallelizable? Does the policy prioritize this job due to its small size, or defer working on this job because of its high parallelizability? To derive an optimal scheduling policy, we must compare the relative benefits of these two effects. We find that the optimal choice between prioritizing short jobs and deferring parallelizable work depends on the scaling behavior of the system. When the system scales such that queueing is rare, it is more important to defer parallelizable work. However, if the system scales such that the probability of queueing goes to one, it is more important to prioritize short jobs.

1.4. Contributions

This paper derives the first results on asymptotic optimality when scheduling many classes of parallelizable jobs. Specifically, this is the first work to develop asymptotically optimal policies when there are more than two classes of parallelizable jobs. We analyze the mean response time of ℓ classes of jobs, each with a different level of parallelizability. We prove our results by considering how the system behaves under a variety of *scaling regimes*, where the number of servers goes to infinity and the system load goes to one simultaneously (see Section 2). Although our results rely on asymptotic analysis, the results cover a wide range of scaling regimes, including sub-Halfin-Whitt regimes and mean field (constant load) scaling. We find that the choice of an optimal policy depends on the chosen scaling regime. Specifically, our contributions are as follows.

- When the system load is not too heavy (sub-Halfin-Whitt), we show that the benefit of deferring parallelizable work outweighs the benefit of prioritizing short jobs. In this case, it is beneficial to use a *least-parallelizable-first* (LPF) policy that assigns the highest priorities to the least parallelizable classes of jobs. We show in Theorem 1 that LPF is asymptotically optimal with respect to mean response time. Furthermore, we show in Theorem 2 that it can be sub-optimal to simply prioritize short jobs in these lighter-load scaling regimes.
- Conversely, when system load is very heavy (super-nondegenerate slowdown (NDS)), it is more important to prioritize short jobs by using the shortest-expected-remaining-processing-time (SERPT) policy as shown in Theorem 4. In these scaling regimes, there are almost always enough jobs to fully utilize all k servers. Hence, it is more

important to minimize queueing times by using SERPT than it is to defer parallelizable work. As a result, we also prove in Theorem 5 that LPF can be suboptimal in these heavier-load scaling regimes.

- Although the above results suggest that one may need to know the system scaling behavior a priori to select an asymptotically optimal policy, we show that this is not the case. Specifically, we define a policy called Threshold (THRESH) that switches between LPF and SERPT depending on how many jobs are in the system. We show in Theorem 6 that THRESH can perform well in both sub-Halfin-Whitt and super-NDS scaling regimes by deferring parallelizable work (using LPF) when there are few jobs in the system and prioritizing short jobs (using SERPT) when there are many jobs in the system.
- Finally, we examine extensions to our theoretical results via simulation in Section 6.

2. Model

We model a system with k identical servers managed by a centralized scheduler. Arriving jobs are placed in a central queue. We consider the case where jobs are preemptible and malleable, meaning that the scheduler can change the number of servers on which a job runs over time. Although more complex architectures may exist, this scenario is common in a wide range of real-world systems that process parallelizable jobs.

Without loss of generality (WLOG), we assume each server runs at speed 1. We assume that each job belongs to one of ℓ job classes. Each job class, i , has its own job size distribution, denoted by the random variable S_i , and parallelizability level, c_i .

A job's size denotes its running time (service time) on a single server. We generally assume that job sizes are unknown and exponentially distributed, that is,

$$S_i \sim \text{Exp}(\mu_i) \quad \forall 1 \leq i \leq \ell.$$

Our theoretical results assume exponential job sizes, but we discuss other size distributions in Section 6.2.

Class i jobs have a parallelizability level of c_i , meaning a class i job can run on up to c_i servers. When a class i job is run on $m \leq c_i$ servers, the job is served at rate m . As a result, this job's remaining service time is distributed as $\text{Exp}(m \cdot \mu_i)$.

We assume class i jobs arrive according to a Poisson process with rate $\lambda_i^{(k)}$, where the total arrival rate is $\lambda^{(k)} = \sum_{i=1}^{\ell} \lambda_i^{(k)}$. We define the system load as

$$\rho := \sum_{i=1}^{\ell} \rho_i \quad \text{where} \quad \rho_i := \frac{\lambda_i^{(k)}}{k\mu_i} \quad \forall 1 \leq i \leq \ell.$$

The system is stable under any work-conserving scheduling policy if $\rho < 1$ (Berg et al. 2020b).

2.1. Asymptotic Analysis

We analyze the system under several scaling regimes. In the *conventional heavy traffic* scaling regime, the number of servers, k , is held constant, and we take the limit as $\rho \rightarrow 1$. We will also consider several *scaling regimes* where $k \rightarrow \infty$ and the arrival rate, $\lambda^{(k)}$, scales with k , meaning that load, ρ , scales with k . See Appendix A for a summary of the standard asymptotic notation used in this paper.

We assume that the proportion of jobs from each class and the job size distributions remain fixed. That is, we define the constants $p_1, p_2, \dots, p_\ell$ such that

$$\lambda_i^{(k)} = p_i \lambda^{(k)} \quad \forall 1 \leq i \leq \ell.$$

We restrict ourselves to scaling regimes where $\lambda^{(k)} = \Theta(k)$ so that ρ does not go to zero as we scale k .

It is convenient to describe the scaling behavior of $\lambda^{(k)}$ in terms of α , the number of “spare servers” worth of capacity. Specifically, we define α to be

$$\alpha := k(1 - \rho).$$

Note that, although α and ρ depend on k , we omit this dependence from the notation for brevity. We will write expressions in terms of $\lambda^{(k)}$ when an explicit dependence on k requires emphasis.

The asymptotic behavior of many queueing systems is known to depend on the asymptotic behavior of α . For example, consider a simple $M/M/k$ system. When $\alpha = \Theta(k)$, the scaling regime is known as the *large-system limit* or *mean-field limit*. The probability of queueing in an $M/M/k$ goes to zero in the mean-field limit. When $\alpha = \Theta(\sqrt{k})$, the scaling regime is known as the Halfin-Whitt regime (Halfin and Whitt 1981). The probability of queueing in an $M/M/k$ converges to a constant in the Halfin-Whitt regime, but the mean queueing time goes to zero. When $\alpha = \Theta(1)$, the scaling regime is known as the NDS regime (Atar 2012, Gupta and Walton 2019).

The probability of queueing in an $M/M/k$ goes to one in the NDS regime, and the mean queueing time converges to a constant. We refer to all scaling regimes where $\alpha = \omega(\sqrt{k \log k})$ as *sub-Halfin-Whitt* scaling regimes.¹ In the sub-Halfin-Whitt regime, the probability of queueing in an $M/M/k$ goes to zero. We refer to all scaling regimes where $\alpha = o(1)$ as *super-NDS* regimes. The probability of queueing in an $M/M/k$ goes to one and the mean queueing time goes to ∞ in super-NDS regimes. Note that, under all super-NDS scaling regimes, $\rho \rightarrow 1$. Under the sub-Halfin-Whitt scaling regimes, ρ either goes to one or converges to a constant less than one, as in mean-field scaling. *It is not clear which, if any, of the results from the simple $M/M/k$ system hold when scheduling parallelizable jobs.*

Although ρ_i depends on k for each class i , ρ_i asymptotically approaches a subcritical load. Specifically, let

$$\rho_i^* := \lim_{k \rightarrow \infty} \rho_i = \lambda^{(k)} p_i \mathbb{E}[S_i] \quad \forall 1 \leq i \leq \ell,$$

where $0 < \rho_i^* < 1$ for any class i , and $\sum_{i=1}^{\ell} \rho_i^* \leq 1$. Further, $\sum_{i=1}^{\ell} \rho_i^* = 1$ in any heavy-traffic scaling regime.

2.2. Scaling Parallelizability Levels

We previously stated that each job class, i , has an associated parallelizability level c_i . For some job classes, this parallelizability level results from algorithmic limitations or limitations of a job's software implementation. For these job classes, the parallelizability level will not change as the system scales. However, other more-parallelizable classes of jobs can be easily reconfigured as the system scales to leverage newly available servers. We therefore allow the value c_i to scale with k for some classes of jobs. In general, we allow each c_i to be any non-decreasing function of k . In this way, we can represent a variety of real-world system constraints that might cause a job's parallelizability level to grow sublinearly with k . For example, in a system with a complex network topology, it is common that only a fraction of the servers (e.g., $\sqrt{k}$) can communicate directly with one another. Hence, for a class of jobs with high communication overhead, the parallelizability level of the jobs might scale as $\sqrt{k}$.

WLOG, let the classes be asymptotically ordered by parallelizability so that, for sufficiently large k ,

$c_1 \leq c_2 \leq \dots \leq c_{\ell}$. Furthermore, we assume that $c_1 = O(1)$, so at least one class of jobs requires a constant number of servers. This mild technical assumption holds in the vast majority of real-world systems. For simplicity, we assume that $c_1 = 1$, but setting c_1 to some other constant does not affect our results.

2.3. Scheduling Policies

A *scheduling policy* must decide, at every moment in time, how many servers to allocate to each job in the system. Let $\{N(t)\}$ be the stochastic process denoting the number of jobs in the system at time $t \geq 0$. Because we only consider scheduling policies that do not idle servers unnecessarily, $\rho < 1$ implies that $\{N(t)\}$ is a positive-recurrent, irreducible continuous-time Markov chain. By ergodicity, there exists a distribution, N , such that $N(t) \xrightarrow{d} N$ as $t \rightarrow \infty$. N is the unique stationary distribution of $N(t)$ and $\mathbb{E}[N] < \infty$. We refer to $\mathbb{E}[N]$ as the *mean number of jobs in system*. We define $\{N_i(t)\}$, N_i , and $\mathbb{E}[N_i]$ analogously, but to describe class i jobs in the system. This gives $\mathbb{E}[N] = \sum_{i=1}^{\ell} \mathbb{E}[N_i]$. Let $X = (N_1, N_2, \dots, N_{\ell})$ denote the full state of the system in stationarity.

Let T denote the *response time*—the time from when a job arrives until it completes—of a job that arrives to a system whose state is drawn according to X . We refer to $\mathbb{E}[T]$ as the *mean response time* across jobs. By Little's law, $\mathbb{E}[T] = \mathbb{E}[N] / \lambda^{(k)} < \infty$. We similarly define $\mathbb{E}[T_i]$ to be the mean response time of a class i job, where $\mathbb{E}[T] = \sum_{i=1}^{\ell} \mathbb{E}[T_i] p_i$ by conditioning. Our goal is to find scheduling policies that minimize $\mathbb{E}[T]$ (or, equivalently, minimize $\mathbb{E}[N]$). We assume the scheduling policy knows the class information (μ_i and c_i) for each job, as well as the system state.

Let $\mathbb{E}[T^{\pi}]$ denote the mean response time under any scheduling policy, π ; π is *asymptotically optimal* if

$$\lim_{k \rightarrow \infty} \frac{\mathbb{E}[T^{\pi'}]}{\mathbb{E}[T^{\pi}]} \geq 1$$

for all policies π' . Again, the asymptotic optimality of a policy generally depends on the asymptotic behavior of α . Asymptotic optimality is defined analogously in the conventional heavy traffic limit.

Our results primarily focus on analyzing two policies: one based solely on deferring parallelizable work and the other based solely on favoring short jobs. The policy that defers parallelizable work is called LPF. LPF preemptively prioritizes the jobs with the lowest parallelizability levels. For example, LPF will allocate as many servers as possible to jobs belonging to the first i classes before allocating any leftover servers to jobs in the $i + 1$ st class. The policy that favors short jobs is called the SERPT policy (Scully et al. 2020b), which at all times preemptively prioritizes the job with the smallest expected remaining size. Recall that a job's size is its running time on a

single server. The SERPT policy allocates as many servers as can be used to the job with the smallest expected remaining size before allocating any servers to the next larger size job.

We also introduce a new policy called THRESH, which is the first policy to dynamically combine the orthogonal goals of deferring parallelizable work and favoring short jobs. THRESH runs LPF when the number of jobs in the system is small and runs SERPT when the number of jobs is large. We will fully define THRESH in Section 5.

In analyzing these policies, we use a combination of coupling and drift arguments.

3. Scheduling Parallelizable Jobs in sub-Halfin-Whitt Regimes

In this section, we consider scheduling in the sub-Halfin-Whitt scaling regimes described in Section 2. We begin by showing that LPF, the policy that defers parallelizable work by prioritizing the least parallelizable jobs, is asymptotically optimal in this case. Intuitively, in the lighter-load sub-Halfin-Whitt scaling regimes where queueing is rare, the benefit of prioritizing short jobs ahead of long jobs is reduced. Furthermore, because there will frequently be fewer than k jobs in the system under these scaling regimes, it is important to “save up” the more parallelizable jobs, which are better at occupying many servers when there are few jobs. LPF performs well because it keeps more servers occupied than other policies.

Given that there is little queueing in sub-Halfin-Whitt regimes, one might expect a wide variety of policies to be asymptotically optimal. However, we also show the surprising result that SERPT can be asymptotically sub-optimal in sub-Halfin-Whitt regimes. If SERPT happens to prioritize very parallelizable jobs that fully utilize the system, all the other jobs in the system are forced to queue unnecessarily. The system is then underutilized when running less-parallelizable jobs. Hence, the choice of scheduling policy greatly affects the overall mean response time of the system, even in sub-Halfin-Whitt scaling regimes.

We begin by showing the asymptotic optimality of LPF in Theorem 1.

Theorem 1. *If $\alpha = \omega(\sqrt{k \log k})$, then for any scheduling policy π ,*

$$\lim_{k \rightarrow \infty} \frac{\mathbb{E}[T^\pi]}{\mathbb{E}[T^{\text{LPF}}]} \geq 1.$$

That is, LPF is asymptotically optimal with respect to mean response time in sub-Halfin-Whitt scaling regimes.

We will prove Theorem 1 via Lemmas 1, 2, and 3. We first outline our argument at a high level. Then, we prove the necessary lemmas before proving Theorem 1.

We will prove that, for each class of jobs i , the mean response time of class i jobs under LPF approaches some lower bound as $k \rightarrow \infty$, and thus the overall mean response time under LPF is asymptotically optimal. When analyzing LPF in this section, we will omit the LPF superscript for the sake of readability.

For any class, i , it is desirable to have more than $\rho_i k$ servers working on class i jobs. In this case, class i jobs are being completed at a rate greater than $\mu_i \rho_i k = \lambda_i^{(k)}$, and hence the number of class i jobs in the system will tend to decrease. Given a system with α spare servers, it is theoretically possible to guarantee that $\rho_i k + \alpha/\ell$ servers are available to class i jobs for all classes, i . Our goal is to prove that LPF guarantees this property with high probability, and therefore it is usually possible to complete class i jobs at a very fast rate. We then argue that this is sufficient to keep the average number of class i jobs in the system very low.

We divide the ℓ job classes into two cases. First, we consider all the job classes whose level of parallelism, $c_i = O(1)$. We call these *inelastic* (I) jobs, because their level of parallelism does not scale with k . Second, we consider all the job classes whose level of parallelism $c_i = \omega(1)$. We refer to these as *elastic* (E) jobs.

To handle any inelastic job class, i , we argue in Lemma 1 that the probability of having a large number of jobs belonging to the first $i - 1$ job classes goes to zero very quickly as $k \rightarrow \infty$. As a result, LPF almost always devotes a sufficient number of servers to the class i jobs. This allows us to show in Lemma 2 that the mean response time of class i jobs is asymptotically optimal.

We use a similar argument to prove the asymptotic optimality for elastic jobs in Lemma 3. However, we can prove this result for all elastic classes simultaneously instead of treating the classes one by one as we did for the inelastic classes. The proof of Theorem 1 follows easily from Lemmas 2 and 3.

The key to proving these lemmas is that we have α/ℓ spare servers to devote to each class. Roughly, this means the LPF policy can tolerate fluctuations in the number of class i jobs on the order of α without having these jobs interfere significantly with the service of lower priority jobs. In sub-Halfin-Whitt scaling regimes, we assume $\alpha = \omega(\sqrt{k \log k})$. Hence, our proofs use drift arguments to recover a well-known form of result—that fluctuations in the number of class i jobs larger than $O(\sqrt{k})$ are rare as $k \rightarrow \infty$.

We define β_i to be the smallest number of class i jobs that can fully utilize $\rho_i k + \alpha/\ell$ servers, giving

$$\beta_i = \left\lceil \frac{\rho_i k + \frac{\alpha}{\ell}}{c_i} \right\rceil.$$

We begin with the following lemma that the number of class i jobs rarely exceeds β_i for any inelastic class.

Lemma 1. *Let I be the highest number such that $c_I = O(1)$. For any class of jobs, $i \leq I$, let $\mathcal{E}_i$ be the set of states defined as*

$$\mathcal{E}_i = \{(n_1, n_2, \dots, n_\ell) : n_j < \beta_j, \forall 1 \leq j \leq i\}.$$

In sub-Halfin-Whitt scaling regimes,

$$\Pr((N_1, N_2, \dots, N_\ell) \notin \mathcal{E}_i) = o\left(\frac{1}{k^3}\right) \quad \forall i \leq I. \quad (1)$$

Proof. We will prove (1) by induction on the job class, i . Specifically, we will prove the following two properties by simultaneous induction. First, we will show there exist constants $d_j^{(i)}$ and $f_j^{(i)} > 0$ such that

$$\Pr(N_i \geq \beta_i) = \sum_{j=1}^i f_j^{(i)} \left(\frac{k}{\alpha}\right)^{j-1} e^{-d_j^{(i)} \alpha^2/k} \quad \forall i \leq I. \quad (2)$$

Second, for some constants $g_j^{(i)} > 0$ and $h_j^{(i)} > 0$, we will show that

$$\Pr((N_1, N_2, \dots, N_\ell) \notin \mathcal{E}_i) \leq \sum_{j=1}^i g_j^{(i)} \left(\frac{k}{\alpha}\right)^{j-1} e^{-h_j^{(i)} \alpha^2/k} \quad \forall i \leq I. \quad (3)$$

Once we have established (2) and (3), we note $\alpha = \omega(\sqrt{k \log k})$ in the sub-Halfin-Whitt scaling regimes and thus $\frac{\alpha^2}{k} = \omega(\log k)$. Note that, for any constant $x > 0$, $\alpha^2/k = \omega(x \log k) = \omega(\log k^x)$. Hence, by setting $x = (j+2)/h_j^{(i)}$, (3) implies

$$g_j^{(i)} \left(\frac{k}{\alpha}\right)^{j-1} \cdot e^{-h_j^{(i)} \alpha^2/k} = o\left(\frac{1}{k^3}\right)$$

for any $i, j \leq I$, as desired.² We require (2) as part of our inductive proof.

Our inductive proof of (2) and (3) relies on the drift bounds established in Wang et al. (2022). These bounds require one to define a Lyapunov function $V(x)$, that takes a system state $x = (n_1, n_2, \dots, n_\ell)$ and returns a real number. The goal is to choose a Lyapunov function such that, for a state x , the instantaneous rate of change of $\mathbb{E}[V(X)|X=x]$, known as the *Lyapunov drift*, is almost always negative. Specifically, if the instantaneous rate of change of $\mathbb{E}[V(X)|X=x]$ is negative whenever $V(x) > B$, Wang et al. (2022) provides bounds on $\Pr(V(X) > B)$ (see Appendix B for a precise statement). We set $V(x) = n_1$ to bound $\Pr(N_1 > B)$ and prove the base case for our induction ($i = 1$). To prove our claims inductively for class $i+1$ jobs, we set $V(x) = n_{i+1}$. Here, we must account for the fact that there may be many lower-class jobs in the system preventing the class $i+1$ jobs from running and causing positive drift, even when $V(x)$ is large. To handle this issue, we exploit a stronger drift bound (see Appendix C for a precise statement) that tolerates some positive drift as long as the states with positive drift occur with bounded probability. The problematic states in this scenario are exactly $\{x \notin \mathcal{E}_i\}$. Our inductive hypothesis holds that $\Pr(X \notin \mathcal{E}_i)$ is small, resulting in the desired bounds on $\Pr(N_{i+1} > B)$ and $\Pr(X \notin \mathcal{E}_{i+1})$.

To prove both statements when $i = 1$, we define the Lyapunov function $V(x) = n_1$. When $n_1 \geq \rho_1 k + \alpha/2\ell$, the drift of the Lyapunov function, ΔV , is

$$\Delta V(x) = \lambda_1^{(k)} - \mu_1 \cdot n_1 \leq \lambda_1^{(k)} - \mu_1 \left(\rho_1 k + \frac{\alpha}{2\ell}\right) = -\mu_1 \frac{\alpha}{2\ell}.$$

Hence, the drift bound from Appendix B yields

$$\Pr(N_1 \geq \beta_1) \leq \Pr\left(N_1 \geq \rho_1 k + \frac{\alpha}{\ell}\right) \leq f_1^{(1)} e^{-d_1^{(1)} \alpha^2/k}$$

for some constant $d_1^{(1)} > 0$ and $f_1^{(1)} = 1$. This proves both (2) and (3) when $i = 1$.

Now, we inductively assume that (2) and (3) hold for the first i job classes. If the Lyapunov function $V(x) = n_{i+1} > \rho_{i+1} k + \alpha/2\ell$, the drift is positive if and only if $x \notin \mathcal{E}_i$; otherwise, the drift is negative. We can

therefore use the modified drift bound from Appendix C to get the bound

$$\Pr(N_{i+1} \geq \beta_{i+1}) \leq f_{i+1}^{(i+1)} e^{-d_{i+1}^{(i+1)} \alpha^2 / k} + O\left(\frac{k}{\alpha}\right) \Pr(X \notin \mathcal{E}_i)$$

for some constants $d_{i+1}^{(i+1)} > 0$ and $f_{i+1}^{(i+1)} > 0$.

Using our inductive hypothesis to bound $\Pr(X \notin \mathcal{E}_i)$, we see that there exist some new constants $f_j^{(i+1)}$ and $d_j^{(i+1)}$ for all $1 \leq j \leq i+1$ such that (2) holds for the $i+1$ st class of jobs.

Finally, to prove (3) for the $i+1$ st class, we note that

$$\Pr(X \notin \mathcal{E}_{i+1}) = \Pr\left(\bigcup_{j=1}^{i+1} N_j > \beta_j\right) \leq \Pr(X \notin \mathcal{E}_i) + \Pr(N_{i+1} > \beta_{i+1}) \leq \sum_{j=1}^{i+1} g_j^{(i+1)} \left(\frac{k}{\alpha}\right)^{j-1} e^{-h_j^{(i+1)} \alpha^2 / k}$$

for some new set of constants $g_j^{(i+1)} > 0$ and $h_j^{(i+1)} > 0$. This completes the proof by induction. $\square$

We now use Lemma 1 to prove the optimality of the inelastic response times via a drift argument.

Lemma 2. For any class $i \leq I$, if $\alpha = \omega(\sqrt{k \log k})$, then

$$\lim_{k \rightarrow \infty} \mathbb{E}[T_i^{LPF}] = \frac{1}{c_i \mu_i}.$$

Proof. Let K_i be the number of servers that process class i jobs under LPF in a random state $X = (N_1, N_2, \dots, N_I)$ in the stationary regime. Let $\theta_i = \lceil \frac{\rho_i k + \log k}{c_i} \rceil$ be the number of class i jobs required to use slightly more than the average number of class i servers. The number of servers used by θ_i class i jobs is at most $\rho_i k + \log k + c_i$ due to rounding. We also define $\delta_i = (N_i - \theta_i)^+$, where $(\cdot)^+$ denotes $\max(\cdot, 0)$ (i.e., the *positive part*).

Intuitively, although class i jobs use $\rho_i k$ servers on average, Lemma 1 suggests that it should be rare for the class i jobs to use $\rho_i k + \log k$ servers. We will use Lemma 1 to show that $\mathbb{E}[\delta_i] \rightarrow 0$ and hence $\mathbb{E}[N_i]$ is at most θ_i as k becomes large. We then apply Little's law to this bound to finish our proof.

Formally, we will make a drift argument using the Lyapunov function $V(x) = ((n_i - \theta_i)^+)^2$ so that $V(X) = \delta_i^2$. Following standard techniques (Bertsimas et al. 1994, Eryilmaz and Srikant 2012, Wang et al. 2022, Xu 2023), we define $\Delta V(x) = \sum_{x' \in \mathcal{X}: x \neq x'} q_{xx'} (V(x') - V(x))$, note that $\mathbb{E}[\Delta V(X)] = 0$, and use this equation to obtain the desired bounds.³ See Section EC.1 of the Online Appendix for a full overview of this technique.

To obtain $\Delta V(X)$, note that δ_i does not change for small values of N_i . If an arrival occurs when there are fewer than θ_i jobs, nothing changes. Similarly, unless there are strictly more than θ_i jobs, nothing changes on a departure either. This gives

$$\Delta V(X) = \mathbb{1}_{(N_i \geq \theta_i)} \lambda_i^{(k)} (2\delta_i + 1) + \mathbb{1}_{(N_i > \theta_i)} \mu_i K_i (-2\delta_i + 1).$$

Setting $\mathbb{E}[\Delta V(X)] = 0$ then gives

$$\mathbb{E}[2\mathbb{1}_{(N_i \geq \theta_i)} \lambda_i^{(k)} \delta_i + \mathbb{1}_{(N_i \geq \theta_i)} \lambda_i^{(k)} + \mathbb{1}_{(N_i > \theta_i)} \mu_i K_i] = \mathbb{E}[2\mathbb{1}_{(N_i \geq \theta_i)} \mu_i K_i \delta_i]. \quad (4)$$

We now claim that $\mathbb{E}[\mathbb{1}_{(N_i > \theta_i)} \mu_i K_i] = \mathbb{E}[\mathbb{1}_{(N_i \geq \theta_i)} \lambda_i^{(k)}]$. Here, we use a similar drift argument (see Section EC.1 in the Online Appendix), this time using the Lyapunov function $G(x) = \delta_i$.⁴ This gives

$$\begin{aligned} \Delta G(X) &= \mathbb{1}_{(N_i \geq \theta_i)} \lambda_i^{(k)} - \mathbb{1}_{(N_i > \theta_i)} \mu_i K_i. \\ \mathbb{E}[\Delta G(X)] &= \mathbb{E}[\mathbb{1}_{(N_i > \theta_i)} \mu_i K_i] - \mathbb{E}[\mathbb{1}_{(N_i \geq \theta_i)} \lambda_i^{(k)}] = 0 \end{aligned} \quad (5)$$

as desired.

Applying (5) to (4) yields

$$\mathbb{E}[\mathbb{1}_{(N_i \geq \theta_i)} \lambda_i^{(k)} \delta_i + \mathbb{1}_{(N_i \geq \theta_i)} \lambda_i^{(k)}] = \mathbb{E}[\mathbb{1}_{(N_i \geq \theta_i)} \mu_i K_i \delta_i]. \quad (6)$$

Now, noting that $\mathbb{1}_{(N_i \geq \theta_i)} \delta_i = \delta_i$, we have

$$\lambda_i^{(k)} \Pr(N_i \geq \theta_i) + \lambda_i^{(k)} \mathbb{E}[\delta_i] = \mu_i \mathbb{E}[K_i \delta_i]. \quad (7)$$

If K_i and δ_i were independent, $\mu_i \mathbb{E}[K_i \delta_i]$ would equal $\lambda_i^{(k)} \mathbb{E}[\delta_i]$ because we know the long run completion rate of type i jobs equals the long run arrival rate of type i jobs in a stable system. Intuitively, however, δ_i and K_i should be positively correlated—we should expect to use more servers on type i jobs when more type i jobs are in the

system. Hence, to turn (7) into an upper bound on $\mathbb{E}[\delta_i]$, we will lower bound $\mathbb{E}[K_i\delta_i]$ to be significantly larger than $\lambda_i^{(k)}\mathbb{E}[\delta_i]$; K_i and δ_i only move inversely when many lower-class jobs enter the system and prevent class i jobs from running. Hence, we can use our upper bound on $\Pr(X \notin \mathcal{E}_{i-1})$ to argue that K_i and δ_i are positively correlated and thus

$$\mathbb{E}[K_i\delta_i] \geq (\mathbb{E}[K_i] + \log k)\mathbb{E}[\delta_i] - O(1) = (\rho_i k + \log k)\mathbb{E}[\delta_i] - O(1), \quad \forall i \leq I. \quad (8)$$

This is shown formally in Appendix D. Plugging (8) into (7) yields

$$\lambda_i^{(k)} + O(1) \geq (\mu_i(k\rho_i + \log k) - \lambda_i^{(k)})\mathbb{E}[\delta_i] = \mu_i \log k \mathbb{E}[\delta_i], \quad (9)$$

$$\frac{\lambda_i^{(k)} + O(1)}{\mu_i \log k} \geq \mathbb{E}[\delta_i], \quad (10)$$

for sufficiently large k . Hence,

$$\mathbb{E}[N_i] \leq \mathbb{E}[\delta_i] + \theta_i \leq o(k) + \theta_i, \quad (11)$$

$$\mathbb{E}[N_i] \leq o(k) + \frac{\rho_i k}{c_i}. \quad (12)$$

Applying Little's law gives

$$\mathbb{E}[T_i] = o(1) + \frac{1}{c_i \mu_i}.$$

Hence, for the first I classes, the mean response time of each class converges to a service time plus an additive term that goes to 0 as $k \rightarrow \infty$. Note that this mean service time represents the lowest possible mean service time for a class i job and is thus a lower bound on the mean response time for class i jobs. $\square$

At this point, we have shown that the I jobs have asymptotically optimal mean response time in sub-Halfin-Whitt regimes. Now, we handle the remaining classes whose level of parallelizability scales with k .

Lemma 3. For any class $i > I$, if $\alpha = \omega(\sqrt{k \log k})$, then

$$\lim_{k \rightarrow \infty} \mathbb{E}[T_i^{LPF}] = 0.$$

Proof. Recall that $c_i = \omega(1)$ for all classes of elastic (E) jobs. We will bound the response time of all elastic classes simultaneously. Let

$$\rho_E = \sum_{j=I+1}^{\ell} \rho_j \quad \text{and} \quad \lambda_E^{(k)} = \sum_{j=I+1}^{\ell} \lambda_j^{(k)}.$$

Let K_E be the combined number of servers used on any elastic job and let $\mu_{\min} = \min_{i \in [I+1, \ell]} \mu_i$. We will bound $\mathbb{E}[N_E]$, the expected number of total jobs of class $I+1$ or higher. Following our earlier argument, we want to exploit our bound on $\mathcal{E}_I$ to get an upper bound on $\mathbb{E}[N_E]$.

Rather than tracking the number of jobs directly, we will track the expected sum of the sizes of elastic jobs in the system. We refer to this quantity as the expected *work* from elastic jobs, W_E . This quantity can be written as

$$W_E = \sum_{j=I+1}^{\ell} \frac{N_j}{\mu_j}.$$

Let $\theta_E = \frac{\rho_E k + \log k}{c_{I+1} \mu_{\min}}$ and let $\delta_E = (W_E - \theta_E)^+$. Furthermore, let $\delta_E(x)$ be the value of δ_E given that the system is in state x .

Note that we divide by c_{I+1} to be pessimistic about how many servers each job uses. We will develop a bound on $\mathbb{E}[\delta_E]$ and use this bound to recover a bound on $\mathbb{E}[N_E]$. The key to this lemma is that $\theta_E = o(k)$ because $c_{I+1} = \omega(1)$. Hence, we can stand to pick up a θ_E term in our bound on $\mathbb{E}[N_E]$, and this term will be dominated by $\lambda_E^{(k)}$ when applying Little's law.

We will now use the drift argument of Section EC.1 in the Online Appendix with the Lyapunov function $V(x) = (\delta_E(x))^2$. This yields

$$\Delta V(X) = \mathbb{1}_{(W_E \geq \theta_E)} \left(\sum_{j=I+1}^{\ell} \frac{2\lambda_j^{(k)}}{\mu_j} \delta_E + \frac{\lambda_j^{(k)}}{\mu_j^2} \right) + \mathbb{1}_{(W_E < \theta_E)} \left(\sum_{j=I+1}^{\ell} -2k_j \delta_E + \frac{k_j}{\mu_j} \right).$$

Once again setting $\mathbb{E}[\Delta V(X)] = 0$,⁵ we get

$$\mathbb{E}[\mathbb{1}_{(W_E \geq \theta_E)} \delta_E k \rho_E] + O(k) = \mathbb{E}[\mathbb{1}_{(W_E \geq \theta_E)} \delta_E K_E], \quad (13)$$

$$k \rho_E \mathbb{E}[\delta_E] + O(k) = \mathbb{E}[\delta_E K_E], \quad (14)$$

where the indicators on both sides vanish because they are redundant, and the terms without δ_E are aggregated into the $O(k)$. Hence, lower bounding $\mathbb{E}[\delta_E K_E]$ will give an upper bound on $\mathbb{E}[\delta_E]$.

Our argument now follows the proof of (8) (see Appendix D) to show that, for sufficiently large k ,

$$\mathbb{E}[\delta_E K_E] \geq (\rho_E k + \log k) \mathbb{E}[\delta_E] - O(1).$$

Adapting Lemma D.2 to prove this claim is straightforward and makes use of our bound on $\Pr(X \notin \mathcal{E}_I)$.

Plugging this lower bound into (14) gives

$$O(k) + O(1) \geq (\rho_E k + \log k) \mathbb{E}[\delta_E] - \rho_E k \mathbb{E}[\delta_E], \quad (15)$$

$$\frac{O(k) + O(1)}{\log k} \geq \mathbb{E}[\delta_E], \quad (16)$$

$$o(k) = \mathbb{E}[\delta_E]. \quad (17)$$

To conclude, we note that

$$\mathbb{E}[W_E] \leq \mathbb{E}[\delta_E] + \mathbb{E}[\theta_E] = \mathbb{E}[\delta_E] + o(k).$$

Let $\mu_{\max} = \max_{i \in [I+1, \ell]} \mu_i$. We then have

$$\mathbb{E}[N_E] = \sum_{j=I+1}^{\ell} \mathbb{E}[N_j] \leq \sum_{j=I+1}^{\ell} \frac{\mathbb{E}[N_j] \mu_{\max}}{\mu_j} = \mu_{\max} \mathbb{E}[W_E].$$

Hence,

$$\mathbb{E}[N_E] \leq \mu_{\max} (\mathbb{E}[\delta_E] + o(k)) = o(k).$$

Applying Little's law, we have

$$\lim_{k \rightarrow \infty} \mathbb{E}[T_E] = \lim_{k \rightarrow \infty} \frac{\mathbb{E}[N_E]}{\lambda_E^{(k)}} = 0.$$

This implies our claim, that the mean response time of class i jobs goes to 0 for all $i > I$. $\square$

We now combine the above lemmas to complete our proof of Theorem 1.

Proof of Theorem 1. By Lemmas 2 and 3, for any class, i , $\mathbb{E}[T_i^{LPF}]$ converges to a lower bound as k becomes large, because the mean response time of a job is lower bounded by its mean service time. Hence, the overall mean response time across jobs, which is a linear combination of the mean response times for each class, also converges to a lower bound. For any policy, π , this gives

$$\lim_{k \rightarrow \infty} \frac{\mathbb{E}[T^\pi]}{\mathbb{E}[T^{LPF}]} \geq 1,$$

and thus LPF is optimal in sub-Halfin-Whitt scaling regimes. $\square$

We conclude this section with Theorem 2, which shows that SERPT is asymptotically suboptimal in sub-Halfin-Whitt scaling regimes. That is, LPF outperforms SERPT (sometimes significantly) in sub-Halfin-Whitt scaling regimes.

Theorem 2. *There exists a set of job classes such that, when $\alpha = \omega(\sqrt{k \log k})$,*

$$\lim_{k \rightarrow \infty} \frac{\mathbb{E}[T^{\text{SERPT}}]}{\mathbb{E}[T^{\text{LPF}}]} > 1.$$

That is, SERPT can be strictly suboptimal in the sub-Halfin-Whitt regime.

Proof. Consider the case where $\ell = 2$ with $c_1 = 1$, $c_2 = k$, and $\mu_2 > \mu_1$. We can easily see that the mean response time of class 2 jobs goes to zero under SERPT in the sub-Halfin-Whitt regime. Hence, to prove the suboptimality of SERPT, we must show that $\lim_{k \rightarrow \infty} \mathbb{E}[T_1^{\text{LPF}}] < \lim_{k \rightarrow \infty} \mathbb{E}[T_1^{\text{SERPT}}]$. Lemma 2 shows that, under LPF in the sub-

Halfin-Whitt regime, the mean response time of a class 1 job converges to $1/\mu_1$. We will show that SERPT does not attain this lower bound on the mean response time of class 1 jobs in the sub-Halfin-Whitt regime.

We consider the case where $\mu_2 > \mu_1$, so SERPT gives strict priority to class 2 jobs. Consider the behavior of one particular class 1 job. This class 1 job will be served for some random amount of time, S_1 . The queueing time of the job is therefore *at least* the amount of time spent waiting behind class 2 jobs that arrive during its S_1 time in service. We define $A_{S_1}^{(2)}$ to be the number of class 2 arrivals during a (random) class 1 service time. We can then write

$$\mathbb{E}[T_1^{\text{SERPT}}] \geq \mathbb{E}\left[\sum_{i=1}^{A_{S_1}^{(2)}} \frac{S_{2_i}}{k}\right] + \mathbb{E}[S_1].$$

Clearly, each S_{2_i} is independent and identically distributed (i.i.d.) and independent of $A_{S_1}^{(2)}$, so we have

$$\mathbb{E}[T_1^{\text{SERPT}}] \geq \mathbb{E}[A_{S_1}^{(2)}] \mathbb{E}\left[\frac{S_2}{k}\right] + \mathbb{E}[S_1].$$

We can then see that

$$\mathbb{E}[A_{S_1}^{(2)}] = \int_0^\infty \mu_1 e^{-\mu_1 t} \mathbb{E}[A_{S_1}^{(2)} | S_1 = t] dt = \int_0^\infty t \cdot \mu_1 e^{-\mu_1 t} \cdot \lambda_2^{(k)} dt = \lambda_2^{(k)} \mathbb{E}[S_1].$$

Thus,

$$\mathbb{E}[T_1^{\text{SERPT}}] \geq \frac{\lambda_2^{(k)}}{k} \mathbb{E}[S_2] \mathbb{E}[S_1] + \mathbb{E}[S_1] = (\rho_2 + 1) \mathbb{E}[S_1]$$

and

$$\lim_{k \rightarrow \infty} \mathbb{E}[T_1^{\text{SERPT}}] \geq \frac{(1 + \rho_2^*)}{\mu_1} > \lim_{k \rightarrow \infty} \mathbb{E}[T_1^{\text{LPF}}].$$

Because SERPT achieves suboptimal mean response time for class 1 jobs as $k \rightarrow \infty$, the overall mean response time under SERPT is suboptimal in sub-Halfin-Whitt scaling regimes. $\square$

4. Scheduling Parallelizable Jobs in Super-NDS Regimes

LPF performs optimally in sub-Halfin-Whitt scaling regimes because it defers parallelizable work in order to keep servers occupied. SERPT, on the other hand, performs suboptimally in these regimes. In super-NDS scaling regimes, this trend is reversed. Namely, in super-NDS scaling regimes, the mean queueing time becomes the dominant term in the mean response time. Because SERPT minimizes mean queueing time by prioritizing short jobs ahead of long jobs, we can show it is asymptotically optimal in this case. Not only does LPF fail to minimize mean queueing time, but the benefits of deferring parallelizable work are diminished in super-NDS regimes. Here, there are usually enough jobs in the system to keep all servers occupied, even without deferring parallelizable work. Hence, we show that LPF is suboptimal in super-NDS scaling regimes. Additionally, we will show that both results also apply in the conventional heavy traffic limit, where k is fixed and $\rho \rightarrow 1$.

The point of this section will be to prove Theorems 3, 4, and 5, which state that SERPT is optimal in conventional heavy traffic and super-NDS scaling regimes, whereas LPF can be suboptimal in these cases. We start by proving a preliminary lemma that will be used repeatedly in our proofs.

This preliminary lemma, Lemma 4, relates the mean number of jobs in a SERPT system to a lower bound on the optimal mean number of jobs in system. Specifically, we consider the lower bound of an optimal policy for a *single-server* preemptive system that runs at speed k . The optimal policy for this single-server system serves jobs one at a time in SERPT order (Van Mieghem 1995). We will refer to the optimal policy for the single-server system as SERPT-1 to emphasize that this is SERPT applied to a system with one fast server. The SERPT policy (without the 1) will continue to denote SERPT in our k -server system.

To describe the priority ordering of classes under SERPT and SERPT-1, let $\gamma = (\gamma_1, \gamma_2, \dots, \gamma_\ell)$ be a sequence such that

$$\mu_{\gamma_i} \geq \mu_{\gamma_{i+1}} \quad \forall 1 \leq i < \ell.$$

We now compare the expected number of jobs under SERPT in our k -server system to the expected number of jobs in the SERPT-1 system.

Lemma 4. *The expected number of jobs under SERPT in the k -server system can be bounded as $\mathbb{E}[N^{\text{SERPT}-1}] \leq \mathbb{E}[N^{\text{SERPT}}] \leq \mathbb{E}[N^{\text{SERPT}-1}] + k \cdot \ell$.*

Proof. The first part of our claim, that $\mathbb{E}[N^{\text{SERPT}-1}] \leq \mathbb{E}[N^{\text{SERPT}}]$, holds because SERPT-1 is optimal for the single-server system. Any policy from the k -server system can be mimicked by the single-server system, so SERPT-1 must provide a lower bound on the optimal policy for the k -server system.

The rest of this proof is devoted to proving the claim that $\mathbb{E}[N^{\text{SERPT}}] \leq \mathbb{E}[N^{\text{SERPT}-1}] + k \cdot \ell$. To prove this, we will track the total work in the system, W , for each policy. Specifically, let $W_i^\pi(t)$ be the total remaining size of all class i jobs in a system using the scheduling policy π at time t . Furthermore, let $W_{\leq \gamma_i}^\pi(t)$ be the total remaining size of job in classes γ_1 through γ_i at time t . Finally, let $\Delta_{\leq \gamma_i}(t)$ be defined as

$$\Delta_{\leq \gamma_i}(t) := W_{\leq \gamma_i}^{\text{SERPT}}(t) - W_{\leq \gamma_i}^{\text{SERPT}-1}(t).$$

We can take expectations to get

$$\mathbb{E}[W_{\leq \gamma_i}^{\text{SERPT}}] = \mathbb{E}[W_{\leq \gamma_i}^{\text{SERPT}-1}] + \mathbb{E}[\Delta_{\leq \gamma_i}].$$

Our goal is to get upper and lower bounds on $\mathbb{E}[\Delta_{\leq \gamma_i}]$, and then use these bounds on the work in system to get a bound on the expected number of jobs in system.

First, it is easy to see that $\mathbb{E}[\Delta_{\leq \gamma_i}] \geq 0$. Consider any fixed arrival sequence of jobs and their associated sizes. This sequence corresponds to one possible sample path for all the random variables in the system. Assume for contradiction that, at some time t ,

$$W_{\leq \gamma_i}^{\text{SERPT}}(t) < W_{\leq \gamma_i}^{\text{SERPT}-1}(t).$$

This implies that just before time t , the SERPT system completes work on classes γ_1 through γ_i at a faster rate than the SERPT-1 system. However, the SERPT-1 system always completes this work at the maximal rate unless $W_{\leq \gamma_i}^{\text{SERPT}-1}(t) = 0$. This implies that

$$W_{\leq \gamma_i}^{\text{SERPT}}(t) < 0,$$

a contradiction. Because $W_{\leq \gamma_i}^{\text{SERPT}}(t) \geq W_{\leq \gamma_i}^{\text{SERPT}-1}(t)$ at any time t , for any arrival sequence of jobs, we have

$$\mathbb{E}[W_{\leq \gamma_i}^{\text{SERPT}}] \geq \mathbb{E}[W_{\leq \gamma_i}^{\text{SERPT}-1}] \quad \forall 1 \leq i \leq \ell.$$

To derive an upper bound, we again consider any fixed arrival sequence of jobs. For any value of i , we partition this arrival sequence into “nonbusy” intervals where the SERPT system has fewer than k jobs in classes γ_1 through γ_i and “busy” intervals where SERPT has more than k jobs from these classes. For any time t_n in a non-busy interval, it is clear that $\Delta_{\leq \gamma_i}(t_n)$ is at most the total remaining size of the at most k jobs in the SERPT system.

We now take expectations over all possible sample paths, giving

$$\mathbb{E}[\Delta_{\leq \gamma_i}(t_n)] \leq \mathbb{E}\left[\sum_{j=1}^{N_{\leq \gamma_i}^{\text{SERPT}}(t_n)} S_{\gamma_i}^{(j)}\right] \leq k\mathbb{E}[S_{\gamma_i}],$$

where $\mathbb{E}[S_{\gamma_i}]$ is the expected size of a job from the class with the i th smallest expected job size. We are exploiting memorylessness here—the remaining size of each of the jobs in the SERPT system is exponentially distributed. Furthermore, because the classes are ordered by their expected job sizes, the remaining size of each job is stochastically less than or equal to S_{γ_i} .

We now use a similar argument to handle any time t_b in a busy interval. For any time t_b , there exists some $t^* \leq t_b$ that marks the start of the busy interval. At time t^* , $\Delta_{\leq \gamma_i}(t^*)$ is at most the total remaining size of the (exactly) k jobs in the SERPT system. In the interval $[t^*, t_b]$, the SERPT completes work at rate k , the maximal rate of work for both systems, so $\Delta_{\leq \gamma_i}(t_b) \leq \Delta_{\leq \gamma_i}(t^*)$. We again take expectations to find that, for any time t_b ,

$$\mathbb{E}[\Delta_{\leq \gamma_i}(t_b)] \leq k\mathbb{E}[S_{\gamma_i}].$$

Because our upper bound on $\Delta_{\leq \gamma_i}$ holds for any time $t \geq 0$, we have

$$\mathbb{E}[\Delta_{\leq \gamma_i}] \leq k\mathbb{E}[S_{\gamma_i}] \quad \forall 1 \leq i \leq \ell$$

and therefore

$$\mathbb{E}[W_{\leq \gamma_i}^{\text{SERPT}}] \leq \mathbb{E}[W_{\leq \gamma_i}^{\text{SERPT-1}}] + k\mathbb{E}[S_{\gamma_i}] \quad \forall 1 \leq i \leq \ell.$$

For class γ_1 jobs, we now immediately use our upper bound to get

$$\mathbb{E}[W_{\gamma_1}^{\text{SERPT}}] \leq \mathbb{E}[W_{\gamma_1}^{\text{SERPT-1}}] + \frac{k}{\mu_{\gamma_1}}.$$

Because job sizes are exponential, we can divide this expression by the mean size of a class γ_1 job to get

$$\mathbb{E}[N_{\gamma_1}^{\text{SERPT}}] \leq \mathbb{E}[N_{\gamma_1}^{\text{SERPT-1}}] + k.$$

For class γ_i jobs where $1 < i \leq \ell$, we have

$$\mathbb{E}[W_{\leq \gamma_i}^{\text{SERPT}}] \leq \mathbb{E}[W_{\leq \gamma_i}^{\text{SERPT-1}}] + \frac{k}{\mu_{\gamma_i}} \quad \text{and} \quad \mathbb{E}[W_{\leq \gamma_{i-1}}^{\text{SERPT}}] \geq \mathbb{E}[W_{\leq \gamma_{i-1}}^{\text{SERPT-1}}].$$

We subtract the latter inequality from the former to yield

$$\mathbb{E}[W_{\leq \gamma_i}^{\text{SERPT}}] - \mathbb{E}[W_{\leq \gamma_{i-1}}^{\text{SERPT}}] \leq \mathbb{E}[W_{\leq \gamma_i}^{\text{SERPT-1}}] + \frac{k}{\mu_{\gamma_i}} - \mathbb{E}[W_{\leq \gamma_{i-1}}^{\text{SERPT-1}}], \quad (18)$$

$$\mathbb{E}[W_{\gamma_i}^{\text{SERPT}}] \leq \mathbb{E}[W_{\gamma_i}^{\text{SERPT-1}}] + \frac{k}{\mu_{\gamma_i}}. \quad (19)$$

We again divide by the average size of a class γ_i job to get

$$\mathbb{E}[N_{\gamma_i}^{\text{SERPT}}] \leq \mathbb{E}[N_{\gamma_i}^{\text{SERPT-1}}] + k \quad \forall 1 \leq i \leq \ell \Rightarrow \mathbb{E}[N^{\text{SERPT}}] \leq \mathbb{E}[N^{\text{SERPT-1}}] + k \cdot \ell$$

as desired. $\square$

We are now ready to use Lemma 4 to prove Theorems 3 and 4.

Theorem 3. Consider a system in the conventional heavy traffic limit as $\rho \rightarrow 1$. For any policy, π ,

$$\lim_{\rho \rightarrow 1} \frac{\mathbb{E}[T^\pi]}{\mathbb{E}[T^{\text{SERPT}}]} \geq 1.$$

That is, SERPT is asymptotically optimal in the conventional heavy traffic regime.

Proof. To establish the optimality of SERPT in conventional heavy traffic, we note that for any policy π , $\mathbb{E}[T^\pi] \geq \mathbb{E}[T^{\text{SERPT-1}}]$. If we can show that

$$\lim_{\rho \rightarrow 1} \frac{\mathbb{E}[T^{\text{SERPT}}]}{\mathbb{E}[T^{\text{SERPT-1}}]} = 1,$$

then

$$\lim_{\rho \rightarrow 1} \frac{\mathbb{E}[T^\pi]}{\mathbb{E}[T^{\text{SERPT}}]} = \lim_{\rho \rightarrow 1} \frac{\mathbb{E}[T^\pi]}{\mathbb{E}[T^{\text{SERPT}}]} \cdot \lim_{\rho \rightarrow 1} \frac{\mathbb{E}[T^{\text{SERPT}}]}{\mathbb{E}[T^{\text{SERPT-1}}]} = \lim_{\rho \rightarrow 1} \frac{\mathbb{E}[T^\pi]}{\mathbb{E}[T^{\text{SERPT-1}}]} \geq 1.$$

Hence, we now show that the $\mathbb{E}[T^{\text{SERPT}}]$ converges to the $\mathbb{E}[T^{\text{SERPT-1}}]$ as $\rho \rightarrow 1$.

We apply Little's law to the result of Lemma 4, which yields

$$\mathbb{E}[T^{\text{SERPT-1}}] \leq \mathbb{E}[T^{\text{SERPT}}] \leq \mathbb{E}[T^{\text{SERPT-1}}] + O(1). \quad (20)$$

The mean response time of the SERPT-1 system in conventional heavy traffic can be derived from standard results on priority queueing (e.g., (32.1) in Harchol-Balter (2013)), yielding

$$\lim_{\rho \rightarrow 1} \mathbb{E}[T^{\text{SERPT-1}}] = \Theta\left(\frac{1}{1-\rho}\right).$$

Combining these two results gives

$$\lim_{\rho \rightarrow 1} \frac{\mathbb{E}[T^{\text{SERPT}}]}{\mathbb{E}[T^{\text{SERPT-1}}]} \leq \lim_{\rho \rightarrow 1} \frac{\mathbb{E}[T^{\text{SERPT-1}}] + O(1)}{\mathbb{E}[T^{\text{SERPT-1}}]} = \lim_{\rho \rightarrow 1} 1 + O(1-\rho) = 1.$$

Because the mean response time under SERPT converges to that of the SERPT-1 system, SERPT is optimal in conventional heavy traffic. $\square$

We now extend our proof from conventional heavy traffic to cover all super-NDS scaling regimes.

Theorem 4. *If $\alpha = o(1)$, then for any policy, π ,*

$$\lim_{k \rightarrow \infty} \frac{\mathbb{E}[T^\pi]}{\mathbb{E}[T^{\text{SERPT}}]} \geq 1.$$

That is, SERPT is asymptotically optimal in super-NDS scaling regimes.

Proof. This argument closely follows the proof of Theorem 3. We again apply Lemma 4 to get (20). In super-NDS scaling regimes, we can again use a standard analysis of priority queueing systems to find that $\mathbb{E}[T^{\text{SERPT}-1}] = \Theta(1/\alpha)$. This gives the analogous result that

$$\lim_{k \rightarrow \infty} \frac{\mathbb{E}[T^{\text{SERPT}}]}{\mathbb{E}[T^{\text{SERPT}-1}]} \leq \lim_{k \rightarrow \infty} \frac{\mathbb{E}[T^{\text{SERPT}-1}] + O(1)}{\mathbb{E}[T^{\text{SERPT}-1}]} = \lim_{k \rightarrow \infty} 1 + O(\alpha) = 1.$$

Hence, SERPT is asymptotically optimal in super-NDS scaling regimes. $\square$

To conclude this section, we show that LPF can fail to minimize mean queueing time and can therefore be asymptotically worse than SERPT in both conventional heavy traffic and super-NDS scaling regimes.

Theorem 5. *Consider a system with a constant number of servers, k_0 , in conventional heavy traffic. There exists a set of job classes such that*

$$\lim_{\rho \rightarrow 1} \frac{\mathbb{E}[T^{\text{LPF}}]}{\mathbb{E}[T^{\text{SERPT}}]} > 1.$$

Furthermore, in super-NDS scaling regimes where $\alpha = o(1)$, there exists a set of job classes such that

$$\lim_{k \rightarrow \infty} \frac{\mathbb{E}[T^{\text{LPF}}]}{\mathbb{E}[T^{\text{SERPT}}]} > 1.$$

That is, LPF can be strictly suboptimal in conventional heavy traffic and super-NDS scaling regimes.

Proof. Consider the case where $\ell = 2$ with $c_1 = 1$, $c_2 = k$, and $\mu_2 > \mu_1$. We established that SERPT is optimal in conventional heavy traffic by showing that

$$\lim_{\rho \rightarrow 1} \frac{\mathbb{E}[T^{\text{SERPT}}]}{\mathbb{E}[T^{\text{SERPT}-1}]} = 1.$$

Given this result, we can prove the first part of our claim by showing that

$$\lim_{\rho \rightarrow 1} \frac{\mathbb{E}[T^{\text{LPF}}]}{\mathbb{E}[T^{\text{SERPT}-1}]} > 1 \Rightarrow \lim_{\rho \rightarrow 1} \frac{\mathbb{E}[T^{\text{LPF}}]}{\mathbb{E}[T^{\text{SERPT}}]} = \lim_{\rho \rightarrow 1} \frac{\mathbb{E}[T^{\text{LPF}}]}{\mathbb{E}[T^{\text{SERPT}-1}]} \cdot \frac{\mathbb{E}[T^{\text{SERPT}-1}]}{\mathbb{E}[T^{\text{SERPT}}]} = \lim_{\rho \rightarrow 1} \frac{\mathbb{E}[T^{\text{LPF}}]}{\mathbb{E}[T^{\text{SERPT}-1}]} > 1.$$

We will also make an analogous argument for the case of super-NDS scaling. We begin by proving a lower bound on the performance of LPF.

We refer to a single-server system of speed k that serves jobs in LPF order as an LPF-1 system. We compare the performance of LPF-1 to the performance of LPF in our original k -server system to get $\lim_{\rho \rightarrow 1} \frac{\mathbb{E}[T^{\text{LPF}}]}{\mathbb{E}[T^{\text{LPF}-1}]} \geq 1$. Similarly, in a super-NDS scaling regime, $\lim_{k \rightarrow \infty} \frac{\mathbb{E}[T^{\text{LPF}}]}{\mathbb{E}[T^{\text{LPF}-1}]} \geq 1$. These claims follow from a coupling argument similar to the proof of Lemma 4. The main difference is that we are assuming a different priority ordering of the job classes in this case, but this does not affect our coupling argument.

To complete the first part of our claim, we will show that in conventional heavy traffic,

$$\lim_{\rho \rightarrow 1} \frac{\mathbb{E}[T^{\text{LPF}-1}]}{\mathbb{E}[T^{\text{SERPT}-1}]} > 1. \quad (21)$$

This will imply that

$$\lim_{\rho \rightarrow 1} \frac{\mathbb{E}[T^{\text{LPF}}]}{\mathbb{E}[T^{\text{SERPT}-1}]} = \lim_{\rho \rightarrow 1} \frac{\mathbb{E}[T^{\text{LPF}-1}]}{\mathbb{E}[T^{\text{SERPT}-1}]} \cdot \frac{\mathbb{E}[T^{\text{LPF}}]}{\mathbb{E}[T^{\text{LPF}-1}]} > 1. \quad (22)$$

To show (21), we rewrite this claim in terms of class 1 and class 2 response times to get

$$\begin{aligned} \frac{\mathbb{E}[T^{\text{LPF}-1}]}{\mathbb{E}[T^{\text{SERPT}-1}]} &= \frac{p_1 \mathbb{E}[T_1^{\text{LPF}-1}] + p_2 \mathbb{E}[T_2^{\text{LPF}-1}]}{p_1 \mathbb{E}[T_1^{\text{SERPT}-1}] + p_2 \mathbb{E}[T_2^{\text{SERPT}-1}]} = \frac{p_1 \frac{\mathbb{E}[S_1]}{k(1-\rho_1)} + p_2 \mathbb{E}[T_2^{\text{LPF}-1}]}{p_2 \frac{\mathbb{E}[S_2]}{k(1-\rho_2)} + p_1 \mathbb{E}[T_1^{\text{SERPT}-1}]} = \frac{O(1) + p_2 \mathbb{E}[T_2^{\text{LPF}-1}]}{O(1) + p_1 \mathbb{E}[T_1^{\text{SERPT}-1}]} \\ &= \frac{\frac{O(1)}{p_1 \mathbb{E}[T_1^{\text{SERPT}-1}]} + \frac{p_2 \mathbb{E}[T_2^{\text{LPF}-1}]}{p_1 \mathbb{E}[T_1^{\text{SERPT}-1}]}}{\frac{O(1)}{p_1 \mathbb{E}[T_1^{\text{SERPT}-1}]} + 1}. \end{aligned}$$

Hence, it suffices to show that

$$\mathbb{E}[T_1^{\text{SERPT}-1}] = \omega(1) \quad (23)$$

and

$$\lim_{\rho \rightarrow 1} \frac{p_2 \mathbb{E}[T_2^{\text{LPF}-1}]}{p_1 \mathbb{E}[T_1^{\text{SERPT}-1}]} > 1. \quad (24)$$

From standard techniques for analyzing single-server priority queues, we know that $\mathbb{E}[T_1^{\text{SERPT}-1}] = \Theta\left(\frac{\rho}{1-\rho}\right) = \omega(1)$ as $\rho \rightarrow 1$, and that (24) holds when $\mu_2 > \mu_1$. Combined, these two conditions prove (21) and (22), completing the first part of the theorem.

To prove our second claim, we make an analogous argument about super-NDS scaling, where all limits are taken as $k \rightarrow \infty$. Specifically, $\mathbb{E}[T_1^{\text{SERPT}-1}] = \Theta\left(\frac{1}{\alpha}\right)$ under super-NDS scaling, so (23) still holds in this case. Similarly, we know from standard single-server queueing analysis that

$$\lim_{k \rightarrow \infty} \frac{p_2 \mathbb{E}[T_2^{\text{LPF}-1}]}{p_1 \mathbb{E}[T_1^{\text{SERPT}-1}]} > 1. \quad (25)$$

Thus, we can conclude that

$$\lim_{k \rightarrow \infty} \frac{\mathbb{E}[T^{\text{LPF}}]}{\mathbb{E}[T^{\text{SERPT}-1}]} = \lim_{k \rightarrow \infty} \frac{\mathbb{E}[T^{\text{LPF}-1}]}{\mathbb{E}[T^{\text{SERPT}-1}]} \cdot \frac{\mathbb{E}[T^{\text{LPF}}]}{\mathbb{E}[T^{\text{LPF}-1}]} = \lim_{k \rightarrow \infty} \frac{\mathbb{E}[T^{\text{LPF}-1}]}{\mathbb{E}[T^{\text{SERPT}-1}]} > 1. \quad \square$$

5. THRESH Policy

One limitation of our results thus far is that one may not know their system's scaling regime. Theorems 1 and 4 suggest that one must know the system scaling behavior in order to select an asymptotically optimal policy. We therefore propose THRESH that dynamically chooses between LPF and SERPT without knowing the system scaling regime. THRESH toggles between LPF and SERPT depending on whether the number of jobs in the system exceeds some threshold, $\tau \approx k$. When the number of jobs in the system is less than τ , THRESH uses LPF to defer parallelizable work and avoid idling servers. When the number of jobs in the system is at least τ , THRESH uses SERPT to minimize queueing time.

Given two job classes with $c_1 = 1$ and $c_2 = k$, we show that THRESH is asymptotically optimal in both sub-Halfin-Whitt and super-NDS regimes. Although our theoretical results are limited to this two-class case, we show in Section 6 that THRESH also performs well with additional inelastic and elastic classes of jobs.

The exact threshold value, τ , affects the traffic regimes in which THRESH is asymptotically optimal. We show that when $\tau = \omega(k)$, THRESH is asymptotically optimal in sub-Halfin-Whitt and conventional heavy traffic regimes. Furthermore, THRESH is asymptotically optimal in any super-NDS regime where $\alpha = o(k/\tau)$. That is, the threshold value τ can be any function that is $\omega(k)$, but its exact order affects the optimality result in the super-NDS regime. For example, we can choose τ to be $\tau = k \log^* k$, where $\log^*$ is the iterated logarithm function. Then in the super-NDS regime, THRESH is asymptotically optimal when $\alpha = o(1/\log^* k)$. Hence, THRESH can be asymptotically optimal in an arbitrarily large portion of the super-NDS regimes, depending on the choice of τ . We state this formally in Theorem 6.

Theorem 6. Consider the problem of scheduling $\ell = 2$ classes of parallelizable jobs with $c_1 = 1$ and $c_2 = k$. Consider the THRESH policy with a threshold value $\tau = \omega(k)$. For any policy, π :

1. If $\alpha = \omega(\sqrt{k \log k})$, then $\lim_{k \rightarrow \infty} \frac{\mathbb{E}[T^\pi]}{\mathbb{E}[T^{\text{THRESH}}]} \geq 1$.
2. In a conventional heavy-traffic scaling regime, $\lim_{\rho \rightarrow 1} \frac{\mathbb{E}[T^\pi]}{\mathbb{E}[T^{\text{THRESH}}]} \geq 1$.
3. If $\alpha = o(k/\tau)$, then $\lim_{k \rightarrow \infty} \frac{\mathbb{E}[T^\pi]}{\mathbb{E}[T^{\text{THRESH}}]} \geq 1$.

Proof. When $\mu_1 \geq \mu_2$, LPF, SERPT, and THRESH are equivalent, and this result is immediate. Hence, we will assume that $\mu_1 < \mu_2$. We prove the three claims of Theorem 6 here.

5.1. Proof of Claim 1

We first split the mean response time into two parts by conditioning on job class:

$$\mathbb{E}[T^{\text{THRESH}}] = p_1 \mathbb{E}[T_1^{\text{THRESH}}] + p_2 \mathbb{E}[T_2^{\text{THRESH}}].$$

To prove the asymptotic optimality of THRESH, we begin by showing that $\lim_{k \rightarrow \infty} \mathbb{E}[T_1^{\text{THRESH}}] = \frac{1}{\mu_1}$.

Our proof strategy is to prove that $\mathbb{E}[\text{\#class 1 jobs in queue}] = o(k)$. Specifically, let N_1 and K_1 be the number of class 1 jobs in the system and in service, respectively, in steady state. Then we know that $\mathbb{E}[\text{\#class 1 jobs in queue}] = \mathbb{E}[N_1] - \mathbb{E}[K_1] = \mathbb{E}[N_1] - k\rho_1$, where we have used the fact that $\mathbb{E}[K_1] = k\rho_1$, which can be verified using Little's law. Then $\mathbb{E}[N_1] - k\rho_1$ can be further upper bounded as

$$\mathbb{E}[N_1] - k\rho_1 \leq \mathbb{E}[(N_1 - (k\rho_1 + \delta))^+ + \delta], \quad (26)$$

where δ is chosen such that $k\rho_1 + \delta$ is an integer and $\delta = \Theta(\sqrt{k})$, and the superscript $+$ means taking the positive part, that is, $x^+ = \max\{x, 0\}$ for any real number x . We assume that k is sufficiently large that $k\rho_1 + \delta < k$. Inequality (26) can be verified by considering the two cases that $N_1 - (k\rho_1 + \delta) \geq 0$ and $N_1 - (k\rho_1 + \delta) < 0$. We choose to bound $\mathbb{E}[N_1] - k\rho_1$ in this particular way because $(N_1 - (k\rho_1 + \delta))^+$ effectively removes the dynamics of N_1 when $N_1 < k\rho_1 + \delta$, enabling us to focus on the more tractable scenario where $N_1 \geq k\rho_1 + \delta$. Note that $\delta = \Theta(\sqrt{k}) = o(k)$. Therefore, it suffices to prove $\mathbb{E}[(N_1 - (k\rho_1 + \delta))^+] = o(k)$, which is the focus of the remainder of this proof.

We bound $\mathbb{E}[(N_1 - (k\rho_1 + \delta))^+]$ using Lyapunov drift analysis. We consider the Lyapunov function $V(n_1, n_2) = ((n_1 - (k\rho_1 + \delta))^+)^2$ for any state (n_1, n_2) . Let k_1 denote the number of class 1 jobs in service when the state is (n_1, n_2) . Then the drift of V can be written as

$$\Delta V(n_1, n_2) = \mathbb{1}_{(n_1 = k\rho_1 + \delta)} \cdot \lambda_1^{(k)} + \mathbb{1}_{(n_1 \geq k\rho_1 + \delta + 1)} \cdot (\lambda_1^{(k)} \cdot (2(n_1 - (k\rho_1 + \delta))^+ + 1) + \mu_1 k_1 (-2(n_1 - (k\rho_1 + \delta))^+ + 1)).$$

We know that in steady state, $\mathbb{E}[\Delta V(N_1, N_2)] = 0$. Therefore,

$$\begin{aligned} & \mathbb{E}[2(N_1 - (k\rho_1 + \delta))^+ (\mu_1 K_1 - \lambda_1^{(k)}) \cdot \mathbb{1}_{(N_1 \geq k\rho_1 + \delta + 1)}] \\ &= \mathbb{E}[(\lambda_1^{(k)} + \mu_1 K_1) \cdot \mathbb{1}_{(N_1 \geq k\rho_1 + \delta + 1)} + \lambda_1^{(k)} \cdot \mathbb{1}_{(N_1 = k\rho_1 + \delta)}] \leq \mathbb{E}[(\lambda_1^{(k)} + \mu_1 K_1) \cdot \mathbb{1}_{(N_1 \geq k\rho_1 + \delta)}] \\ &\leq \lambda_1^{(k)} + k\mu_1. \end{aligned} \quad (27)$$

We lower bound the left-hand-side term $\mathbb{E}[2(N_1 - (k\rho_1 + \delta))^+ (\mu_1 K_1 - \lambda_1^{(k)}) \cdot \mathbb{1}_{(N_1 \geq k\rho_1 + \delta + 1)}]$. We abuse notation slightly and reuse LPF to also denote the event that the system is running LPF, that is, the event that $N_1 + N_2 \leq \tau$. Then,

$$\begin{aligned} & \mathbb{E}[2(N_1 - (k\rho_1 + \delta))^+ (\mu_1 K_1 - \lambda_1^{(k)}) \cdot \mathbb{1}_{(N_1 \geq k\rho_1 + \delta + 1)}] \\ &= \mathbb{E}[2(N_1 - (k\rho_1 + \delta))^+ (\mu_1 K_1 - \lambda_1^{(k)}) \cdot \mathbb{1}_{(\text{LPF} \cap \{N_1 \geq k\rho_1 + \delta + 1\})}] \\ &\quad + \mathbb{E}[2(N_1 - (k\rho_1 + \delta))^+ (\mu_1 K_1 - \lambda_1^{(k)}) \cdot \mathbb{1}_{(\text{LPF}^c \cap \{N_1 \geq k\rho_1 + \delta + 1\})}] \\ &\geq \mathbb{E}[2(N_1 - (k\rho_1 + \delta))^+ \cdot \mu_1 (\delta + 1) \cdot \mathbb{1}_{(\text{LPF} \cap \{N_1 \geq k\rho_1 + \delta + 1\})}] \end{aligned} \quad (28)$$

$$\begin{aligned} &\quad + \mathbb{E}[2(N_1 - (k\rho_1 + \delta))^+ (-\lambda_1^{(k)}) \cdot \mathbb{1}_{(\text{LPF}^c \cap \{N_1 \geq k\rho_1 + \delta + 1\})}] \\ &= \mathbb{E}[2(N_1 - (k\rho_1 + \delta))^+ \cdot \mu_1 (\delta + 1) \cdot \mathbb{1}_{(\text{LPF})}] + \mathbb{E}[2(N_1 - (k\rho_1 + \delta))^+ (-\lambda_1^{(k)}) \cdot \mathbb{1}_{(\text{LPF}^c \cap \{N_1 \geq k\rho_1 + \delta + 1\})}] \end{aligned} \quad (29)$$

$$\begin{aligned} &= \mathbb{E}[2(N_1 - (k\rho_1 + \delta))^+ \cdot \mu_1 (\delta + 1)] - \mathbb{E}[2(N_1 - (k\rho_1 + \delta))^+ \cdot \mu_1 (\delta + 1) \cdot \mathbb{1}_{(\text{LPF}^c)}] \\ &\quad + \mathbb{E}[2(N_1 - (k\rho_1 + \delta))^+ (-\lambda_1^{(k)}) \cdot \mathbb{1}_{(\text{LPF}^c \cap \{N_1 \geq k\rho_1 + \delta + 1\})}] \\ &\geq \mathbb{E}[2(N_1 - (k\rho_1 + \delta))^+ \cdot \mu_1 (\delta + 1)] - \mathbb{E}[2(N_1 - (k\rho_1 + \delta))^+ \cdot (\mu_1 (\delta + 1) + \lambda_1^{(k)}) \cdot \mathbb{1}_{(\text{LPF}^c)}], \end{aligned} \quad (30)$$

where (28) is because $K_1 \geq k\rho_1 + \delta + 1$ when LPF is in use and $N_1 \geq k\rho_1 + \delta + 1$, and (29) is due to the equality $(N_1 - (k\rho_1 + \delta))^+ \cdot \mathbb{1}_{(N_1 \geq k\rho_1 + \delta + 1)} = (N_1 - (k\rho_1 + \delta))^+$.

We now plug the lower bound (30) back into (27), which gives

$$\begin{aligned}\mathbb{E}[2(N_1 - (k\rho_1 + \delta))^+ \cdot \mu_1(\delta + 1)] &\leq \lambda_1^{(k)} + k\mu_1 + \mathbb{E}[2(N_1 - (k\rho_1 + \delta))^+ \cdot (\mu_1(\delta + 1) + \lambda_1^{(k)}) \cdot \mathbb{1}_{(\text{LPF}^c)}] \\ \mathbb{E}[2(N_1 - (k\rho_1 + \delta))^+] &\leq \frac{\lambda_1^{(k)} + k\mu_1}{2\mu_1(\delta + 1)} + \frac{\mathbb{E}[(N_1 - (k\rho_1 + \delta))^+ \cdot (\mu_1(\delta + 1) + \lambda_1^{(k)}) \cdot \mathbb{1}_{(\text{LPF}^c)}]}{\mu_1(\delta + 1)}.\end{aligned}$$

Because the first term $\frac{\lambda_1^{(k)} + k\mu_1}{2\mu_1(\delta + 1)} = o(k)$, it suffices to show the second term is also $o(k)$. To this end, we have

$$\begin{aligned}\mathbb{E}[(N_1 - (k\rho_1 + \delta))^+ \cdot (\mu_1(\delta + 1) + \lambda_1^{(k)}) \cdot \mathbb{1}_{(\text{LPF}^c)}] &\leq (\mu_1(\delta + 1) + \lambda_1^{(k)}) \cdot \mathbb{E}[N_1 \cdot \mathbb{1}_{(\text{LPF}^c)}] \\ &\leq (\mu_1(\delta + 1) + \lambda_1^{(k)}) \cdot \mathbb{E}[N_1^2]^{1/2} \cdot (\Pr(\text{LPF}^c))^{1/2} \\ &\leq (\mu_1(\delta + 1) + \lambda_1^{(k)}) \cdot O(k^2) \cdot (\Pr(\text{LPF}^c))^{1/2}.\end{aligned}\tag{31}$$

$$\tag{32}$$

Here, (31) follows from the Cauchy-Schwarz inequality, and (32) follows from the same moment bound used in Appendix D. What remains is to bound $\Pr(\text{LPF}^c)$.

Recall that THRESH performs LPF if $N_1 + N_2 \leq \tau$. Thus, $\Pr(\text{LPF}^c) = \Pr(N_1 + N_2 > \tau)$. Given sufficiently large values of k ,

$$N_1 + N_2 > \tau \Rightarrow \frac{N_1}{\mu_1} + \frac{N_2}{\mu_2} > \frac{N_1 + N_2}{\mu_2} > \frac{\tau}{\mu_2} \Rightarrow \frac{N_1}{\mu_1} + \frac{N_2}{\mu_2} > \frac{2k}{\mu_1},\tag{33}$$

where (33) uses the assumption that $\mu_1 < \mu_2$ and $\tau = \omega(k)$. We consider the Lyapunov function $W(n_1, n_2) = n_1/\mu_1 + n_2/\mu_2$. When $W(n_1, n_2) \geq \frac{k}{\mu_1}$, we have either at least k class 1 jobs or at least 1 class 2 job in service. This means all servers are busy under THRESH. Let k_1 and k_2 be the numbers of servers working on class 1 and class 2 jobs, respectively, in state (n_1, n_2) . Therefore, because $k_1 + k_2 = k$, we have

$$\Delta W(n_1, n_2) = \lambda_1^{(k)} \frac{1}{\mu_1} + \lambda_2^{(k)} \frac{1}{\mu_2} - k_1 \cdot \mu_1 \frac{1}{\mu_1} - k_2 \cdot \mu_2 \frac{1}{\mu_2} = k\rho_1 + k\rho_2 - k = -\alpha.$$

Then, using the same tail bound from Appendix B, we have

$$\Pr\left(\frac{N_1}{\mu_1} + \frac{N_2}{\mu_2} > \frac{2k}{\mu_1}\right) \leq \left(\frac{(\lambda_1^{(k)} + \lambda_2^{(k)})/\mu_1}{(\lambda_1^{(k)} + \lambda_2^{(k)})/\mu_1 + \alpha}\right)^{k+1} = O(e^{-d\alpha}),\tag{34}$$

for some constant $d > 0$ independent of k . Therefore,

$$\Pr(\text{LPF}^c) = \Pr(N_1 + N_2 > \tau) = O(e^{-d\alpha}).$$

Plugging this back into (32) yields

$$\mathbb{E}[(N_1 - (k\rho_1 + \delta))^+ \cdot (\mu_1(\delta + 1) + \lambda_1^{(k)}) \cdot \mathbb{1}_{(\text{LPF}^c)}] = O(k^3 e^{-d\alpha}).\tag{35}$$

Putting everything together gives

$$\begin{aligned}\mathbb{E}[2(N_1 - (k\rho_1 + \delta))^+] &\leq \frac{\lambda_1^{(k)} + k\mu_1}{2\mu_1(\delta + 1)} + \frac{\mathbb{E}[(N_1 - (k\rho_1 + \delta))^+ \cdot (\mu_1(\delta + 1) + \lambda_1^{(k)}) \cdot \mathbb{1}_{(\text{LPF}^c)}]}{\mu_1(\delta + 1)} \\ &= o(k) + \frac{O(k^3 e^{-d\alpha})}{\mu_1(\delta + 1)} = o(k).\end{aligned}$$

Applying Little's law then gives

$$\lim_{k \rightarrow \infty} \mathbb{E}[T_1] \leq \frac{o(k) + k\rho_1 + \delta}{\lambda_1^{(k)}} = \frac{1}{\mu_1}.$$

To complete our proof, note that the response time of class 2 jobs under THRESH should intuitively be lower than their response time under LPF, because the number of servers devoted to class 2 jobs under THRESH is greater than or equal to the number of servers used for class 2 jobs by LPF in every state. Formally, we can prove this using a drift argument that is nearly identical to that of Lemma 3. Specifically, we pessimistically assume that class 2 jobs always get $(k - N_1)^+$ servers. Then, we bound the number of class 1 jobs using the modified tail bound from Appendix C. Here, we use our bound on $\Pr(\text{LPF}^c)$ to bound the probability of having positive drift in the number of class 1 jobs. From this point, the argument to bound $\mathbb{E}[N_2]$ is identical to the proof of Lemma 3, giving $\lim_{k \rightarrow \infty} \mathbb{E}[T_2^{\text{THRESH}}] = 0$.

Hence, the mean response time of both class 1 and class 2 jobs under THRESH is asymptotically optimal.

5.2. Proof of Claim 2

Our proof follows the general argument of Theorem 3 with some adjustments. Recall that in the proof of Theorem 3, we consider the SERPT-1 system, which uses a single server of speed k to run the optimal single-server policy that prioritizes jobs with the highest values of μ_i .

For any scheduling policy π , $\lim_{\rho \rightarrow 1} \frac{\mathbb{E}[T^\pi]}{\mathbb{E}[T^{\text{SERPT-1}}]} \geq 1$. Therefore, to prove that $\lim_{\rho \rightarrow 1} \frac{\mathbb{E}[T^\pi]}{\mathbb{E}[T^{\text{THRESH}}]} \geq 1$, it suffices to prove that $\lim_{\rho \rightarrow 1} \frac{\mathbb{E}[T^{\text{SERPT-1}}]}{\mathbb{E}[T^{\text{THRESH}}]} = 1$. Because $\lim_{\rho \rightarrow 1} \frac{\mathbb{E}[T^{\text{SERPT-1}}]}{\mathbb{E}[T^{\text{THRESH}}]} \leq 1$, it suffices to prove that $\lim_{\rho \rightarrow 1} \frac{\mathbb{E}[T^{\text{SERPT-1}}]}{\mathbb{E}[T^{\text{THRESH}}]} \geq 1$.

We first split the mean response time into two parts based on class 1 and class 2 jobs:

$$\frac{\mathbb{E}[T^{\text{SERPT-1}}]}{\mathbb{E}[T^{\text{THRESH}}]} = \frac{p_1 \mathbb{E}[T_1^{\text{SERPT-1}}] + p_2 \mathbb{E}[T_2^{\text{SERPT-1}}]}{p_1 \mathbb{E}[T_1^{\text{THRESH}}] + p_2 \mathbb{E}[T_2^{\text{THRESH}}]}.$$

Intuitively, class 1 jobs should perform well under THRESH compared with the SERPT-1 system, because class 1 jobs always have low priority in the SERPT-1 system. Specifically, we can show that

$$\mathbb{E}[N_1^{\text{THRESH}}] \leq \mathbb{E}[N_1^{\text{SERPT-1}}] + k. \quad (36)$$

To prove this, we compare THRESH to a single-server system of speed k that also runs THRESH. Using the same coupling argument from the proof of Theorem 3, we show that the single-server version of THRESH is within k jobs of the k server version of THRESH on average. Then, because the SERPT-1 system gives lower priority to class 1 jobs than THRESH, the SERPT-1 system must have more class 1 jobs than the single-server THRESH system on average. Taken together, these arguments yield (36). We then have

$$\frac{\mathbb{E}[T^{\text{SERPT-1}}]}{\mathbb{E}[T^{\text{THRESH}}]} \geq \frac{p_1 \mathbb{E}[T_1^{\text{SERPT-1}}] + p_2 \mathbb{E}[T_2^{\text{SERPT-1}}]}{p_1 \mathbb{E}[T_1^{\text{SERPT-1}}] + O(1) + p_2 \mathbb{E}[T_2^{\text{THRESH}}]} \geq \frac{p_1 \mathbb{E}[T_1^{\text{SERPT-1}}]}{p_1 \mathbb{E}[T_1^{\text{SERPT-1}}] + O(1) + p_2 \mathbb{E}[T_2^{\text{THRESH}}]}. \quad (37)$$

Next, we know from classical queueing theory that

$$\mathbb{E}[T_1^{\text{SERPT-1}}] \geq \frac{1}{1 - \rho_2} \left(\frac{1}{k\mu_1} + \frac{\rho_2}{k\mu_2(1 - \rho)} + \frac{\rho_1}{k\mu_1(1 - \rho)} \right). \quad (38)$$

We can also prove the following lemma, whose proof is given in Appendix E.

Lemma 5. *The mean response time of class 2 jobs under THRESH can be bounded as $\mathbb{E}[T_2^{\text{THRESH}}] \leq \frac{2\tau}{k\mu_2\rho_2} + \frac{4}{k\mu_2\rho_2(1-\rho_2)}$.*

To finish our proof, we can apply these bounds to (37), giving

$$\frac{\mathbb{E}[T^{\text{SERPT-1}}]}{\mathbb{E}[T^{\text{THRESH}}]} \geq \frac{p_1 \mathbb{E}[T_1^{\text{SERPT-1}}]}{p_1 \mathbb{E}[T_1^{\text{SERPT-1}}] + O(1) + O(1)} \quad \text{and} \quad \lim_{\rho \rightarrow 1} \frac{\mathbb{E}[T^{\text{SERPT-1}}]}{\mathbb{E}[T^{\text{THRESH}}]} \geq \lim_{\rho \rightarrow 1} \frac{1}{1 + O(1 - \rho)} = 1.$$

5.3. Proof of Claim 3

To prove Claim 3, we can slightly modify our proof of Claim 2. Recall that for THRESH in super-NDS regimes, we have assumed $\alpha = o(k/\tau)$. Furthermore, we have that $\lim_{k \rightarrow \infty} \rho_2 = \rho_2^*$ where ρ_2^* is a constant. Therefore, in super-NDS regimes, the bound in (38) becomes $\mathbb{E}[T_1^{\text{SERPT-1}}] = \omega(\tau/k)$, and the upper bound in Lemma 5 becomes $\mathbb{E}[T_2^{\text{THRESH}}] = O(\tau/k)$. Applying these bounds to (37) (which remains the same) gives

$$\frac{\mathbb{E}[T^{\text{SERPT-1}}]}{\mathbb{E}[T^{\text{THRESH}}]} \geq \frac{p_1 \mathbb{E}[T_1^{\text{SERPT-1}}]}{p_1 \mathbb{E}[T_1^{\text{SERPT-1}}] + O(1) + O(\frac{\tau}{k})} \quad \text{and} \quad \lim_{k \rightarrow \infty} \frac{\mathbb{E}[T^{\text{SERPT-1}}]}{\mathbb{E}[T^{\text{THRESH}}]} \geq \lim_{k \rightarrow \infty} \frac{1}{1 + o(\frac{k}{\tau}) + o(1)} = 1.$$

Taken together, these three claims give Theorem 6. $\square$

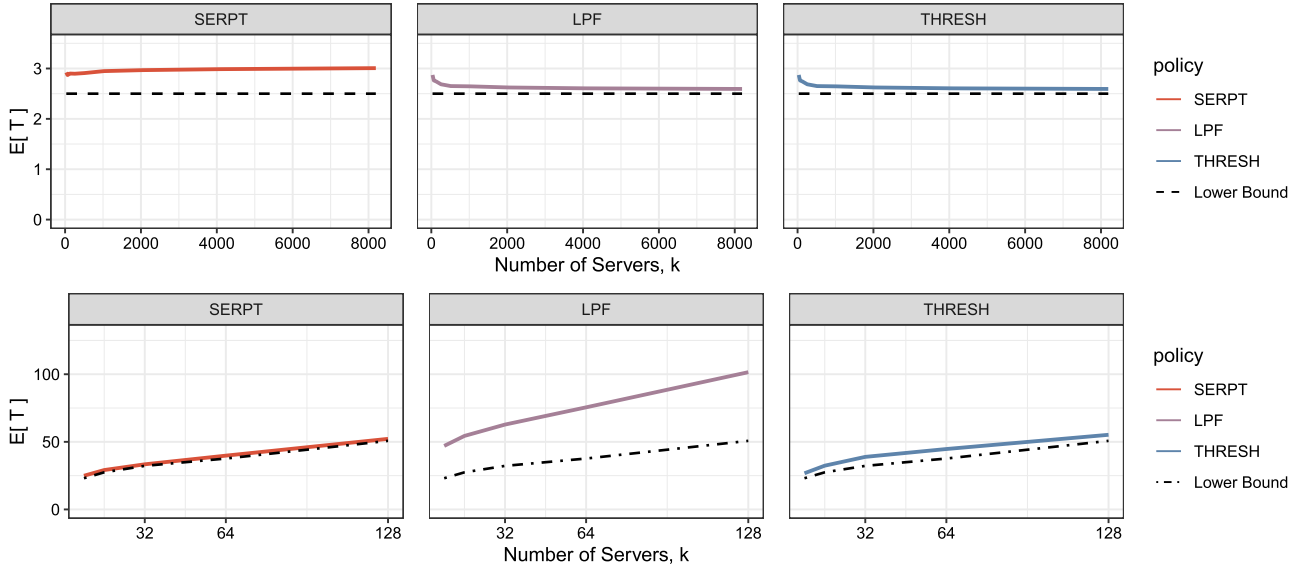
6. Evaluation

In this section, we evaluate our theoretical results via simulation. Although our theoretical results require some technical assumptions, we find that the high-level lessons from these theorems—deferring parallelizable work in lighter load and prioritizing short jobs in heavier load—apply more generally.

6.1. THRESH with Additional Classes

Although we prove the asymptotic optimality of THRESH in a special case of our model with $\ell = 2$ job classes, we conjecture that THRESH works well in general for any number of job classes with arbitrary levels of parallelizability. To validate this intuition, we simulate LPF, SERPT, and THRESH in various scaling regimes in Figure 1. Indeed, given $\ell = 4$ job classes with a variety of parallelizability levels, THRESH performs near-optimally in both sub-Halfin-Whitt and super-NDS scaling regimes.

Figure 1. Simulations of Various Scheduling Policies in Sub-Halfin-Whitt (Top) and Super-NDS (Bottom) Regimes with $\alpha = 2k^{3/4}$ and $\alpha = 2k^{-1/4}$, Respectively



Notes. Here, $\ell = 4$, and $c_1 = 1$, $c_2 = 4$, $c_3 = \log k$, $c_4 = k$ and $\mu_1 = .2$, $\mu_2 = .05$, $\mu_3 = .3$, $\mu_4 = .1$. In sub-Halfin-Whitt regimes, mean response time under LPF converges to mean service time (our “lower bound”). In super-NDS regimes, mean response time under SERPT is close to that of the SERPT-1 policy, a lower bound that assumes a single fast server. THRESH performs well in both regimes.

Although our theoretical results only consider the sub-Halfin-Whitt and super-NDS scaling regimes, we simulated the performance of these policies in a middle scaling regime that lies between these cases for the sake of completeness. These results are presented in Appendix F.

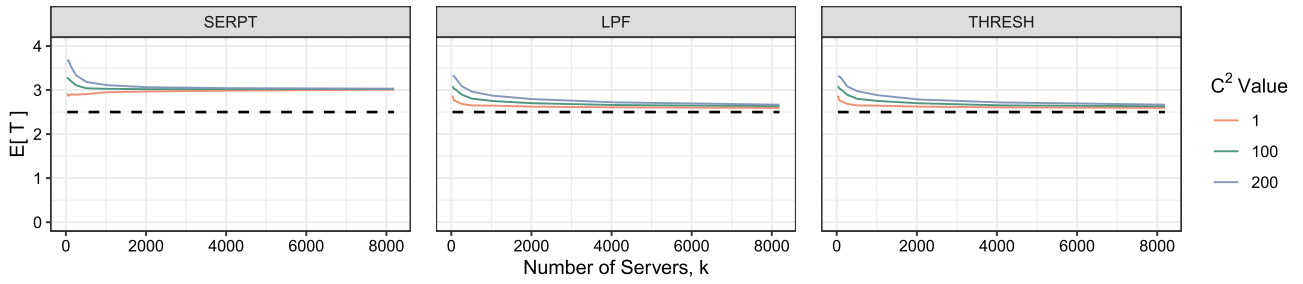
6.2. General Job Size Distributions

We have thus far assumed that each class of jobs, i , is associated with some exponential job size distribution. It is not clear how our results generalize when job sizes are generally distributed.

First, consider scheduling generally distributed parallelizable jobs in conventional heavy traffic. Recent results (Scully et al. 2020a) have demonstrated that a Gittins index policy is optimal for scheduling nonparallelizable jobs in conventional heavy traffic. We find that these results hold when jobs are parallelizable. Specifically, Scully et al. (2020a) define a quantity called $B(r)$ that tracks the number of servers used to run jobs with a Gittins index of r or less. When moving from nonparallelizable jobs to parallelizable jobs, $B(r)$ does not decrease for any given system state, implying that the bounds from this work continue to hold when scheduling parallelizable jobs. The key fact exploited in Scully et al. (2020a) is that, when $B(r)$ is small because fewer than k servers are working on a job with Gittins index at most r , there must be at most $k - 1$ rank r jobs in the system. When jobs are parallelizable, this same principle holds because we now assume that each job runs on *at least* one server. Following the arguments from Scully et al. (2020a), we find that prioritizing parallelizable jobs based on their Gittins indices is optimal in conventional heavy traffic.

Unfortunately, this argument does not generalize cleanly to other scaling regimes. The proof of heavy-traffic optimality in Scully et al. (2020a) relies on the analysis of SRPT in conventional heavy traffic from Lin et al. (2010). Currently, the behavior of SRPT in other scaling regimes is not well understood. We conjecture that a Gittins-based policy is asymptotically optimal in super-NDS scaling regimes, but a proof of this claim does not follow immediately from Scully et al. (2020a) and is beyond the scope of this paper.

Second, consider scheduling non-exponentially distributed jobs in sub-Halfin-Whitt scaling regimes. Given that our results hold for ℓ exponential job classes, we conjecture that our results hold for phase-type job size distributions. Specifically, our current proof of the optimality of LPF bounds $\mathbb{E}[T_i]$ by showing that the departure rate of class i jobs is almost always greater than $\lambda_i^{(k)}$. Given jobs with phase-type distributions, we can use a similar argument to bound the time required to complete each phase of each type of job. However, when considering some phase, j , of class i jobs, the rate at which jobs enter phase j is not stationary—It depends on the number of class i jobs in service whose next phase could be j . Hence, we must argue that the completion rate of jobs in phase j is almost always greater than the time-varying rate at which jobs enter phase j . We can easily compute the

Figure 2. Simulations Showing the Effect of Variability on Mean Response Time

Notes. Experiments use the same $\ell = 4$ job classes as Figure 1, but with different job size distributions. When $C^2 = 1$, the job size distribution is exponential. As C^2 increases, we examine hyperexponential job size distributions with two states such that the mean job size remains the same, but the squared coefficient of variation is increased.

long-run average rate at which jobs enter phase j , but we must show that the fluctuations around this average rate do not cause problems for a policy like LPF.

Although a proof for the case of phase-type job sizes is beyond the scope of this paper, we check via simulation to confirm that our conjecture is plausible. Figure 2 shows that our results appear to hold given jobs that follow different hyperexponential distributions with various levels of job size variability. Here, $C^2 = 1$ corresponds to the same exponential distributions as Figure 1. The larger values of C^2 denote that the job size distributions are balanced-mean hyperexponential distributions with the same mean job sizes but correspondingly higher squared coefficients of variation. We find that, although convergence appears slower as C^2 increases, LPF and THRESH appear to be asymptotically optimal in sub-Halfin-Whitt scaling regimes.

7. Conclusion

This paper presents the first theoretical results on balancing the tradeoff between prioritizing short jobs and deferring parallelizable work. Additionally, this paper provides the first optimality results on scheduling more than two classes of parallelizable jobs, where each class has its own job size distribution and parallelizability level. We show that the choice of an asymptotically optimal scheduling policy depends on the scaling regime chosen. LPF is asymptotically optimal in sub-Halfin-Whitt regimes, and SERPT is asymptotically optimal in super-NDS regimes. The THRESH policy, which combines LPF and SERPT, is asymptotically optimal in both sub-Halfin-Whitt and super-NDS scaling regimes.

We see two main avenues for extending our results. First, as discussed in Section 6.2, allowing jobs to follow arbitrary size distributions complicates our analysis. Existing techniques for analyzing $M/G/k$ systems with non-parallelizable jobs have only considered scheduling in conventional heavy traffic, and it remains unclear how to extend these results both to lighter-load scaling regimes and to scheduling parallelizable jobs. Second, we have assumed that our jobs are perfectly parallelizable up to some parallelizability level. In practice, however, jobs may receive a *sublinear* speedup from parallelization. Hence, although our results consider a work-conserving system, it is of great practical interest to analyze a non-work-conserving variant of our problem. We note that this generalization is related to the problem of state-dependent service rates (Ayesta et al. 2020) and the analysis of systems with Markov-Modulated service rates (Duran et al. 2022). We hope that the techniques and intuition developed in these papers can be applied to the problem of scheduling jobs with sublinear speedups.

Appendix A. Asymptotic Notation

We use standard asymptotic notation throughout the paper to describe scaling behavior. Specifically, consider two functions $f(k), g(k) > 0$, and let $c_1, c_2 > 0$ be arbitrary constants. We define the relevant notation in the table below.

Table A.1. Asymptotic Notation

Name	Notation	Definition	Description
Little-o	$f(k) = o(g(k))$	$\lim_{k \rightarrow \infty} \frac{f(k)}{g(k)} = 0$	$f(k)$ grows much slower than $g(k)$
Big-O	$f(k) = O(g(k))$	$\limsup_{k \rightarrow \infty} \frac{f(k)}{g(k)} \leq c_1$	$f(k)$ grows weakly slower than $g(k)$
Big- Θ	$f(k) = \Theta(g(k))$	$c_1 \leq \limsup_{k \rightarrow \infty} \frac{f(k)}{g(k)} \leq c_2$	$f(k)$ grows roughly as fast as $g(k)$
Little- ω	$f(k) = \omega(g(k))$	$\lim_{k \rightarrow \infty} \frac{f(k)}{g(k)} = \infty$	$f(k)$ grows much faster than $g(k)$

Appendix B. Drift-Based Tail Bound

We use lemma 10 from Wang et al. (2022) to bound the tail probability of N_1^{LPF} . This lemma defines a Lyapunov function, $V(x)$. If the drift of the Lyapunov function is negative when $V(x) > y$, the lemma bounds the probability that $V(X) \gg y$. A full statement of lemma 10 from Wang et al. (2022) can be found in the Online Appendix. Using this lemma, we prove the following tail bound.

Lemma B.1. *For some constant $d > 0$,*

$$\Pr\left(N_1^{\text{LPF}} \geq \rho_1 k + \frac{\alpha}{\ell}\right) = O(e^{-d\alpha^2/k}).$$

Proof. We begin by defining a Lyapunov function $V(x) = n_1$. Let $\mu_{\max} = \max_{1 \leq i \leq \ell} \mu_i$.

When $n_1 \geq \rho_1 k + \frac{\alpha}{2\ell}$, the drift of the Lyapunov function, ΔV , is

$$\Delta V(x) = \lambda_1^{(k)} - \mu_1 \cdot n_1 \leq \lambda_1^{(k)} - \mu_1 \left(\rho_1 k + \frac{\alpha}{2\ell}\right) = -\mu_1 \frac{\alpha}{2\ell}.$$

The other conditions needed to apply the lemma from Wang et al. (2022) are trivially satisfied. Furthermore, the maximum rate of departures from any state, $q_{\max}$, is $\mu_{\max} + \lambda^{(k)}$. We therefore obtain the following bound:

$$\Pr\left(N_1^{\text{LPF}} \geq \left(\rho_1 k + \frac{\alpha}{2\ell}\right) + 2 \cdot \frac{\alpha}{4\ell}\right) \leq \left(\frac{q_{\max}}{q_{\max} + \frac{\alpha}{2\ell}}\right)^{1 + \frac{\alpha}{4\ell}} \leq \left(\frac{q_{\max}}{q_{\max} + \frac{\alpha}{2\ell}}\right)^{\frac{\alpha}{4\ell}}.$$

We then simplify this expression as follows:

$$\begin{aligned} \Pr\left(N_1^{\text{LPF}} \geq \left(\rho_1 k + \frac{\alpha}{2\ell}\right) + 2 \cdot \frac{\alpha}{4\ell}\right) &\leq \left(\frac{q_{\max} + \frac{\alpha}{2\ell}}{q_{\max} + \frac{\alpha}{2\ell}} - \frac{\frac{\alpha}{2\ell}}{q_{\max} + \frac{\alpha}{2\ell}}\right)^{\frac{\alpha}{4\ell}} \\ &= \left(1 - \frac{1}{\frac{2\ell \cdot q_{\max}}{\mu_1 \alpha} + 1}\right)^{\frac{\alpha}{4\ell}} \\ &= \left(1 - \frac{1}{\frac{k}{\mu_1 \alpha} (2\ell \mu_{\max} + \frac{\lambda^{(k)}}{k} + \frac{\alpha}{k})}\right)^{\frac{\alpha}{4\ell}} \\ &\leq \left(1 - \frac{d}{\frac{k}{\alpha}}\right)^{\frac{\alpha}{4\ell}} \end{aligned}$$

for some constant $d > 0$. Note that this constant d exists because $\alpha < k$ and $\lambda^{(k)} = \Theta(k)$. To get the desired bound, we can then write

$$\Pr\left(N_1^{\text{LPF}} \geq \left(\rho_1 k + \frac{\alpha}{2\ell}\right) + 2 \cdot \frac{\alpha}{4\ell}\right) \leq \left(\left(1 - \frac{c_1}{\frac{k}{\alpha}}\right)^{\frac{k}{\alpha}}\right)^{\frac{\alpha}{4\ell k}}.$$

For any positive value of k , this yields

$$\Pr\left(N_1^{\text{LPF}} \geq \left(\rho_1 k + \frac{\alpha}{2\ell}\right) + 2 \cdot \frac{\alpha}{4\ell}\right) = \Pr\left(N_1^{\text{LPF}} \geq \rho_1 k + \frac{\alpha}{\ell}\right) \leq e^{-d\alpha^2/4\ell k}.$$

□

Appendix C. Modified Drift-Based Tail Bound

We now apply a slightly modified version of the drift bound from Appendix B to bound N_i^{LPF} . Whereas the original bound requires strictly negative drift in the Lyapunov function when its value is sufficiently large, the modified tail bound allows positive drift in the Lyapunov function as long as the stationary probability of being in the states with positive drift is bounded. This bound is originally proven as lemma A.1 of Weng and Wang (2020) (see the Online Appendix for details).

Lemma C.1. *For any class of jobs, $i + 1 \leq I$, there exist constants $f_{i+1}^{(i+1)} > 0$ and $d_{i+1}^{(i+1)} > 0$ such that*

$$\Pr(N_{i+1}^{\text{LPF}} \geq \beta_{i+1}) \leq f_{i+1}^{(i+1)} e^{-d_{i+1}^{(i+1)} \alpha^2/k} + O\left(\frac{k}{\alpha}\right) \Pr(X \notin \mathcal{E}_i),$$

where $\mathcal{E}_i$ is the set of states such that $n_j < \beta_j$ for all $j \leq i$.

Proof. We define a Lyapunov function $V(x) = n_{i+1}$.

For any state, x , in $\mathcal{E}_i$, the number of servers available to class $i + 1$ jobs is at least

$$k - \sum_{j=1}^i c_j \cdot \beta_j \geq k - \sum_{j=1}^i \rho_j k + \frac{\alpha}{\ell} + c_j \geq \rho_{i+1} k + \frac{\alpha}{\ell} - \sum_{j=1}^i c_j.$$

Hence, as long as a state is in $\mathcal{E}_i$, we can apply the same drift argument used in Lemma B.1. Here, we are relying on the assumption that, for any $i \leq I$,

$$\sum_{j=1}^i c_j = O(1) \quad \text{and} \quad \alpha = \omega(1),$$

so the extra servers used by the lower-class jobs do not affect our drift argument order-wise.

The issue is that when a state is not in $\mathcal{E}_i$, there may not be enough free servers to guarantee negative drift of the Lyapunov function. However, the positive drift in these states is upper bounded by $\lambda_{i+1}^{(k)} = O(k)$. Hence, according to the lemma from Weng and Wang (2020), we can bound the tail probability of N_{i+1} in terms of the stationary probability that a state lies outside $\mathcal{E}_i$ and causes positive drift in our Lyapunov function. Specifically, we have

$$\Pr(N_{i+1}^{\text{LPF}} > \beta_{i+1}) \leq f_{i+1}^{(i+1)} e^{-d_{i+1}^{(i+1)} \alpha^2 / k} + O\left(\frac{k}{\alpha}\right) \Pr(X \notin \mathcal{E}_i)$$

for some constants $f_{i+1}^{(i+1)} > 0$ and $d_{i+1}^{(i+1)} > 0$. $\square$

Appendix D. Proof of Equation (8)

We divide the proof of (8) into two parts: a technical lemma and a main lemma.

We begin by proving the following technical lemma using the drift-based moment bound from lemma 10 of Wang et al. (2022). See the Online Appendix for a full statement of the moment bound.

Lemma D.1. *For any class of jobs, i ,*

$$\mathbb{E}[(N_i^{\text{LPF}})^2] = O\left(k^2 + \left(\frac{k}{\alpha}\right)^2\right).$$

Thus, in the sub-Halfin-Whitt regime,

$$\mathbb{E}[(N_i^{\text{LPF}})^2] = O(k^2).$$

Proof. We begin by defining the Lyapunov function $V(x) = \sum_{j=1}^i n_j / \mu_j$ and let $\mu_{\min} = \min_{j \leq i} \mu_j$. Consider any state x where $V(x) \geq 2k / \mu_{\min}$. This implies that $\sum_{j=1}^i n_j (\mu_{\min} / \mu_j) \geq 2k$ and thus $\sum_{j=1}^i n_j \geq 2k$.

Because there are more than k jobs in the system, and each job can use at least one server, there are no idle servers when $V(x) \geq 2k / \mu_{\min}$. Let k_i be the number of servers allocated to class i jobs when the system is in state x . We have

$$\Delta V(x) = \sum_{j=1}^i \frac{\lambda_j^{(k)}}{\mu_j} - \sum_{j=1}^i \frac{k_j \mu_j}{\mu_j} \tag{D.1}$$

$$= -k \left(1 - \sum_{j=1}^i \rho_j \right) = -\alpha. \tag{D.2}$$

The total transition rate out of any state is upper bounded by some $O(k)$ term. Furthermore, the change in V between adjacent states is bounded by $\sum_{j=1}^i 1 / \mu_j$. Hence, the bound from Wang et al. (2022) yields

$$\mathbb{E}[V(X)^2] \leq d_1 k^2 + d_2 + \left(\frac{d_3 k + \alpha}{\alpha} \right)^2$$

for some constants $d_1, d_2, d_3 > 0$. This gives

$$\mathbb{E}[(N_i^{\text{LPF}})^2] \leq \mu_i \cdot \mathbb{E}[V(X)^2] = O\left(k^2 + \left(\frac{k}{\alpha}\right)^2\right).$$

In the sub-Halfin-Whitt regime, we have $\alpha = \omega(\sqrt{k \log k})$, and thus

$$\mathbb{E}[(N_i^{\text{LPF}})^2] = O(k^2). \quad \square$$

We now use the technical lemma to prove our main lemma, restated from (8).

Lemma D.2. *For sufficiently large k , when $\alpha = \omega(\sqrt{k \log k})$,*

$$\mathbb{E}[K_i \delta_i] \geq (\rho_i k + \log k) \mathbb{E}[\delta_i] - O(1), \quad \forall i \leq I.$$

Proof. To get this lower bound, we will use the bound on $Pr(X \notin \mathcal{E}_{i-1})$ from Lemma 1. Note that when $X \in \mathcal{E}_{i-1}$, we have

$$K_i \geq k - \sum_{j=1}^{i-1} k\rho_j + \frac{\alpha}{\ell} + c_j \geq k\rho_i + \frac{\alpha}{\ell} - \sum_{j=1}^{i-1} c_j,$$

and thus, for any $i \leq I$ and for sufficiently large k ,

$$\mathbb{E}[K_i \delta_i] \geq \mathbb{E}[\mathbb{1}_{(N_i \geq \theta_i)} \mathbb{1}_{(X \in \mathcal{E}_{i-1})} \delta_i K_i] \geq (k\rho_i + \log k) \mathbb{E}[\mathbb{1}_{(N_i \geq \theta_i)} \mathbb{1}_{(X \in \mathcal{E}_{i-1})} \delta_i] \quad (\text{D.3})$$

$$\geq (k\rho_i + \log k) \mathbb{E}[\mathbb{1}_{(N_i \geq \theta_i)} \delta_i - \mathbb{1}_{(\delta_i \geq \theta_i)} \mathbb{1}_{(X \notin \mathcal{E}_{i-1})} \delta_i] \quad (\text{D.4})$$

$$\geq (k\rho_i + \log k) \mathbb{E}[\mathbb{1}_{(N_i \geq \theta_i)} \delta_i - \mathbb{1}_{(X \notin \mathcal{E}_{i-1})} \delta_i] \quad (\text{D.5})$$

$$= (k\rho_i + \log k) \mathbb{E}[\delta_i - \mathbb{1}_{(X \notin \mathcal{E}_{i-1})} \delta_i]. \quad (\text{D.6})$$

Here, we exploit the fact that there are $\rho_i k + \alpha/\ell - O(1)$ free cores when $X \in \mathcal{E}_{i-1}$, and we are conditioning on having enough jobs to use $\rho_i k + \log k$ cores. Because we are in a sub-Halfin-Whitt scaling regime, $\alpha/\ell - O(1) > \log k$ for sufficiently large k , so (D.3) holds for sufficiently large k .

We now apply Cauchy-Schwarz to show that the second, negative term inside the expectation is $o(1/k)$. That is, we want to show that

$$\mathbb{E}[\mathbb{1}_{(X \notin \mathcal{E}_{i-1})} \delta_i] \leq \sqrt{\mathbb{E}[(\mathbb{1}_{(X \notin \mathcal{E}_{i-1})})^2] \mathbb{E}[\delta_i^2]} = \sqrt{\mathbb{E}[\mathbb{1}_{(X \notin \mathcal{E}_{i-1})}] \mathbb{E}[\delta_i^2]} = o\left(\frac{1}{k}\right).$$

For the indicator term, it suffices to apply the bound from Lemma 1. We then need to show that $\mathbb{E}[\delta_i^2]$ is not too large. We bound $\mathbb{E}[\delta_i^2]$ using our technical lemma, Lemma D.1, to get

$$\mathbb{E}[\delta_i^2] \leq \mathbb{E}[N_i^2] \leq \mu_i E[V(x)^2] = O(k^2).$$

Hence, the second term in (D.6) is $o(1/k)$. This gives the desired bound on $\mathbb{E}[K_i \delta_i]$. $\square$

Appendix E. Proof of Lemma 5

Lemma 5 (Restated). The mean response time of class 2 jobs under THRESH can be bounded as $\mathbb{E}[T_2^{\text{THRESH}}] \leq \frac{2\tau}{k\mu_2\rho_2} + \frac{4}{k\mu_2\rho_2(1-\rho_2)}$.

Proof. Consider the Lyapunov function $V(n_1, n_2) = n_2$ for any state (n_1, n_2) . Then, when $V(n_1, n_2) \geq \tau$, we know that THRESH must be prioritizing class 2 jobs, and thus the drift is given by

$$\Delta V(n_1, n_2) = \lambda_2^{(k)} - k\mu_2 = -k\mu_2(1 - \rho_2).$$

Therefore, by the moment bounds in Wang et al. (2022), we have

$$\begin{aligned} \mathbb{E}[N_2^{\text{THRESH}}] &\leq 2\tau + 4 \cdot \frac{\lambda_2^{(k)} + k\mu_2(1 - \rho_2)}{k\mu_2(1 - \rho_2)} \\ &= 2\tau + \frac{4}{1 - \rho_2}. \end{aligned}$$

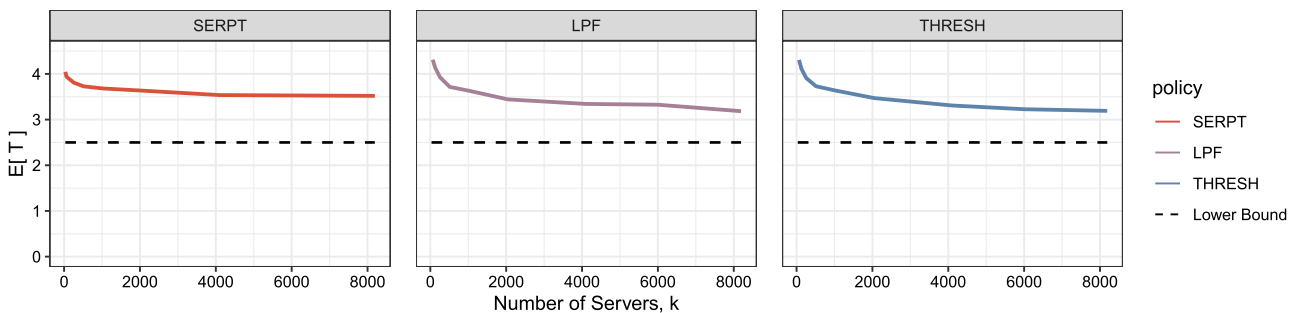
Then by Little's law, we have

$$\mathbb{E}[T_2^{\text{THRESH}}] = \frac{\mathbb{E}[N_2^{\text{THRESH}}]}{\lambda_2^{(k)}} \leq \frac{2\tau}{k\mu_2\rho_2} + \frac{4}{k\mu_2\rho_2(1 - \rho_2)}. \quad \square$$

Appendix F. Simulations in a Middle Scaling Regime

Although our theoretical results apply to both the sub-Halfin-Whitt and super-NDS scaling regimes, we can simulate the performance of various policies in the middle scaling regimes between these two cases. Figure F.1 shows that performance in

Figure F.1. Simulations of Various Scheduling Policies in a Middle Scaling Regime with $\alpha = 2k^{1/4}$



Notes. Here, $\ell = 4$, and $c_1 = 1, c_2 = 4, c_3 = \log k, c_4 = k$ and $\mu_1 = 0.2, \mu_2 = 0.05, \mu_3 = 0.3, \mu_4 = 0.1$. In this case, as in the sub-Halfin-Whitt regimes, both LPF and THRESH appear to perform well as k increases, whereas SERPT appears to converge to an asymptotically suboptimal mean response time.

the middle scaling regimes appears similar to performance in the sub-Halfin-Whitt regimes, with LPF and THRESH converging to the optimal mean response time as $k \rightarrow \infty$.

Endnotes

- ¹ The sub-Halfin-Whitt regime was first introduced in Liu and Ying (2020) and was subsequently discussed in Varma et al. (2023) and Storm and Borst (2023). The original definition requires $\alpha = \Theta(k^b)$ for any $.5 < b < 1$. Our definition is more general in that it requires just $\sqrt{\log k}$ separation from the Halfin-Whitt regime and includes mean-field scaling regimes.
- ² This is where we use the $\sqrt{\log k}$ separation from the Halfin-Whitt regime in our argument about the sub-Halfin-Whitt regime.
- ³ This argument requires that $\mathbb{E}[V(X)^2]$ is finite. It is easy to see that δ_i has finite moments using the drift bound in Section EC.2 of the Online Appendix.
- ⁴ Again, $\mathbb{E}[G(X)^2]$ is finite because δ_i has finite moments.
- ⁵ This argument requires $\mathbb{E}[V(X)^2]$ to be finite, which is easily shown using the drift bound in Section EC.2 of the Online Appendix.

References

- Agrawal K, Leiserson CE, He Y, Hsu WJ (2008) Adaptive work-stealing with parallelism feedback. *ACM Trans. Comput. Systems* 26(3):1–32.
- Agrawal K, Li J, Lu K, Moseley B (2016) Scheduling parallel DAG jobs online to minimize average flow time. Krauthgamer R, ed. *Proc. Twenty-Seventh Annual ACM-SIAM Sympos. Discrete Algorithms (SODA 2016)* (Society for Industrial and Applied Mathematics, Philadelphia), 176–189.
- Atar R (2012) A diffusion regime with nondegenerate slowdown. *Oper. Res.* 60(2):490–500.
- Ayesta U, Prabhu B, Righter R (2020) Scheduling in a single-server queue with state-dependent service rates. *Probab. Engrg. Inform. Sci.* 34(4):507–521.
- Berg B, Dorsman J, Harchol-Balter M (2018) Towards optimality in parallel scheduling. *Proc. ACM Measurement Analysis Comput. Systems* 1(2):1–30.
- Berg B, Vesilo R, Harchol-Balter M (2020a) heSRPT: Parallel scheduling to minimize mean slowdown. *Performance Evaluation* 144:102–147.
- Berg B, Harchol-Balter M, Moseley B, Wang W, Whitehouse J (2020b) Optimal resource allocation for elastic and inelastic jobs. *Proc. 32nd ACM Sympos. Parallelism Algorithms Architectures* (ACM, New York), 75–87.
- Berg B, Whitehouse J, Moseley B, Wang W, Harchol-Balter M (2022) The case for phase-aware scheduling of parallelizable jobs. *ACM SIGMETRICS Performance Evaluation Rev.* 49(3):65–66.
- Bertsimas D, Paschalidis IC, Tsitsiklis JN (1994) Optimization of multiclass queueing networks: Polyhedral and nonlinear characterizations of achievable performance. *Ann. Appl. Probab.* 4(1):43–75.
- Blumofe RD, Leiserson CE (1999) Scheduling multithreaded computations by work stealing. *J. ACM* 46(5):720–748.
- Chen J, Jindel S, Walzer R, Sen R, Jimshelishvili N, Andrews M (2016) The memsql query optimizer: A modern optimizer for real-time analytics in a distributed database. *Proc. VLDB Endowment* 9(13):1401–1412.
- Delgado P, Didona D, Dinu F, Zwaenepoel W (2018) Kairos: Preemptive data center scheduling without runtime estimates. *Proc. ACM Sympos. Cloud Comput.* (ACM, New York), 135–148.
- Duran S, Ayesta U, Verloop I (2022) On the Whittle index of Markov modulated restless bandits. *Queueing Systems* 102(3–4):373–430.
- Edmonds J (1999) Scheduling in the dark. *Proc. 31st Annual ACM Sympos. Theory Comput.* (ACM, New York), 179–188.
- Eryilmaz A, Srikant R (2012) Asymptotically tight steady-state queue length bounds implied by drift conditions. *Queueing Systems* 72(3):311–359.
- Grosf I, Harchol-Balter M, Scheller-Wolf A (2020) Stability for two-class multiserver-job systems. Preprint, submitted October 1, <https://arxiv.org/abs/2010.00631>.
- Grosf I, Harchol-Balter M, Scheller-Wolf A (2022a) WCFS: A new framework for analyzing multiserver systems. *Queueing Systems* 102(1–2):143–174.
- Grosf I, Scully Z, Harchol-Balter M (2018) SRPT for multiserver systems. *Performance Evaluation* 127:154–175.
- Grosf I, Scully Z, Harchol-Balter M, Scheller-Wolf A (2022b) Optimal scheduling in the multiserver-job model under heavy traffic. *Proc. ACM Measurement Analysis Comput. Systems* 6(3):1–32.
- Gupta V, Walton N (2019) Load balancing in the nondegenerate slowdown regime. *Oper. Res.* 67(1):281–294.
- Gupta A, Acun B, Sarood O, Kalé LV (2014) Towards realizing the potential of malleable jobs. *Proc. IEEE 21st Internat. Conf. High Performance Comput.* (IEEE Computer Society, Washington, DC), 1–10.
- Halfin S, Whitt W (1981) Heavy-traffic limits for queues with many exponential servers. *Oper. Res.* 29(3):567–588.
- Harchol-Balter M (2013) *Performance Modeling and Design of Computer Systems: Queueing Theory in Action* (Cambridge University Press, Cambridge, UK).
- Harchol-Balter M (2022) The multiserver job queueing model. *Queueing Systems* 100(3):201–203.
- Hill MD, Marty MR (2008) Amdahl’s law in the multicore era. *Computer* 41:33–38.
- Hong Y, Wang W (2022) Sharp waiting-time bounds for multiserver jobs. *Proc. Twenty-Third Internat. Sympos. Theory Algorithmic Foundations Protocol Design Mobile Networks Mobile Comput.* (ACM, New York), 161–170.
- Jayaram Subramanya S, Arfeen D, Lin S, Qiao A, Jia Z, Ganger GR (2023) Sia: Heterogeneity-aware, goodput-optimized ML-cluster scheduling. Flinn J, Seltzer MI, Druschel P, Kaufmann A, Mace J, eds. *Proc. 29th Sympos. Operating Systems Principles* (ACM, New York), 642–657.
- Leis V, Boncz P, Kemper A, Neumann T (2014) Morsel-driven parallelism: A NUMA-aware query evaluation framework for the many-core age. Dyreson C, ed. *Proc. 2014 ACM SIGMOD Internat. Conf. Management Data* (ACM, New York), 743–754.
- Lin M, Wierman A, Zwart B (2010) The average response time in a heavy-traffic SRPT queue. *ACM SIGMETRICS Performance Evaluation Rev.* 38(2):12–14.

- Lin S-H, Paolieri M, Chou C-F, Golubchik L (2018) A model-based approach to streamlining distributed training for asynchronous sgd. *2018 IEEE 26th Internat. Sympos. Modeling Anal. Simulation Comput. Telecomm. Systems (MASCOTS)* (IEEE Computer Society, Washington, DC), 306–318.
- Liu X, Ying L (2020) Steady-state analysis of load-balancing algorithms in the sub-Halfin-Whitt regime. *J. Appl. Probab.* 57(2):578–596.
- Qiao A, Choe SK, Subramanya SJ, Neiswanger W, Ho Q, Zhang H, Ganger GR, Xing EP (2021) Pollux: Co-adaptive cluster scheduling for goodput-optimized deep learning. *15th USENIX Sympos. Operating Systems Design Implementation (OSDI 2021)* (USENIX Association, Berkeley, CA).
- Scully Z, Grosf I, Harchol-Balter M (2020a) The Gittins policy is nearly optimal in the M/G/k under extremely general conditions. *Proc. ACM Measurement Analysis Comput. Systems* 4(3):1–29.
- Scully Z, Grosf I, Harchol-Balter M (2020b) Optimal multiserver scheduling with unknown job sizes in heavy traffic. *ACM SIGMETRICS Performance Evaluation Rev.* 48(2):33–35.
- Storm PJ, Borst S (2023) Diffusion-level universality of many-server systems with concurrent service. *Oper. Res. Lett.* 51(3):212–219.
- Tumanov A, Zhu T, Park JW, Kozuch MA, Harchol-Balter M, Ganger GR (2016) Tetrisched: Global rescheduling with adaptive plan-ahead in dynamic heterogeneous clusters. Cadar C, Pietzuch PR, Keeton K, Rodrigues R, eds. *Proc. Eleventh European Conf. Comput. Systems* (ACM, New York), 1–16.
- Van Mieghem JA (1995) Dynamic scheduling with convex delay costs: The generalized $c\mu$ rule. *Ann. Appl. Probab.* 809–833.
- Varma SM, Castro F, Maguluri ST (2023) Power-of-d choices load balancing in the sub-Halfin Whitt regime. *Stochastic Systems* 15(4):291–336.
- Vernica R, Carey MJ, Li C (2010) Efficient parallel set-similarity joins using mapreduce. *Proc. ACM SIGMOD Internat. Conf. Management Data* (ACM, New York), 495–506.
- Wagner B, Kohn A, Neumann T (2021) Self-tuning query scheduling for analytical workloads. *Proc. Internat. Conf. Management Data* (ACM, New York), 1879–1891.
- Wang W, Maguluri ST, Srikant R, Ying L (2022) Heavy-traffic insensitive bounds for weighted proportionally fair bandwidth sharing policies. *Math. Oper. Res.* 47(4):2691–2720.
- Weng W, Wang W (2020) Achieving zero asymptotic queueing delay for parallel jobs. *Proc. ACM Measurement Analysis Comput. Systems* 4(3):1–36.
- Xu K (2023) Drift method: From stochastic networks to machine learning. Accessed September 3, 2023, <https://web.stanford.edu/kuangxu/papers/driftmethod.pdf>.
- Zhu Y, Jiang Y, Wu W, Ding L, Teredesai A, Li D, Lee W (2014) Minimizing makespan and total completion time in MapReduce-like systems. *IEEE INFOCOM 2014-IEEE Conf. Comp. Comm.* (IEEE, Piscataway, NJ), 2166–2174.