The Finite-Skip Method for Multiserver Analysis

Isaac Grosof, Mor Harchol-Balter, Alan Scheller-Wolf

March 14, 2022

Abstract

Multiserver queueing systems are found at the core of a wide variety of practical systems. Many important multiserver models have a previously-unexplained similarity: identical mean response time behavior is empirically observed in the heavy traffic limit. We explain this similarity for the first time.

In this paper, we introduce the work-conserving finite-skip (WCFS) class of models. This class includes many important models, including the heterogeneous M/G/k, the limited processor sharing policy for the M/G/1, the threshold parallelism model, and the multiserver-job model under a novel scheduling algorithm which we introduce.

We prove that for all WCFS models, scaled mean response time $E[T](1-\rho)$ converges to the same value, $E[S^2]/(2E[S])$, in the heavy-traffic limit. Moreover, we prove additively tight bounds on mean response time for all WCFS models, which hold for all load ρ. For each of the four models mentioned above, our bounds are the first known bounds on mean response time.

1 Introduction

Consider the following four queueing models, which are each important, practical models, but which seem very different. We will refer to these models throughout the paper as our *four motivating models*:

- **Heterogeneous M/G/k:** A k-server system where each server runs at a different speed. Jobs are held at a central queue and served in First-Come-First-Served (FCFS) order when servers become available. If multiple servers are vacant when a job arrives, a server assignment policy such as Fastest Server First is applied.
- **Limited processor sharing:** A single-server system where if at least k jobs are present, the k earliest arrivals each receive an equal fraction of the server. If fewer than k jobs are present, the server is split equally among the jobs.
- **Threshold parallelism:** A multiserver system where jobs are moldable, meaning they can run on any number of servers, up to some threshold, with perfect speedup. We consider FCFS service, where jobs are allocated a number of servers equal to their threshold, as long as servers are available. The next job in FCFS order is then allocated the remaining servers, which may be less than the job's threshold.
- **Multiserver-jobs under the ServerFilling policy:** A multiserver system where the jobs are called "multiserver jobs," because each job requires a fixed number of servers, which it holds concurrently throughout its service. We examine a service policy called *ServerFilling*, which always fills all of the servers if enough jobs are available.

We define these models in more detail in Section 3.

We will show that, while our four motivating models appear quite different, their mean response times, $E[T]$, are very similar, especially in the heavy-traffic limit. Specifically, we will show that their behavior in the heavy traffic limit is identical to that of the M/G/1/FCFS model. Moreover, we will show that the mean response time of each of these disparate models only differs by an *additive* constant from that of M/G/1/FCFS for all loads, a much stronger result than merely proving that the ratio of their mean response times converges to 1 in the heavy traffic traffic limit.

The similarity of these models is illustrated by Fig. 1. In this figure, mean response time $E[T]$ has been scaled by a factor of $1-\rho$; this is done to help illustrate the asymptotic behavior in the $\rho \to 1$ limit. Observe that in each of our models of interest, as well as in the M/G/1 and the M/G/4, $E[T](1-\rho)$ converges to $E[S^2]/2E[S]$, the mean of the equilibrium (excess) distribution, where S denotes the job size distribution and $\rho = \lambda E[S] < 1$ is the system load.

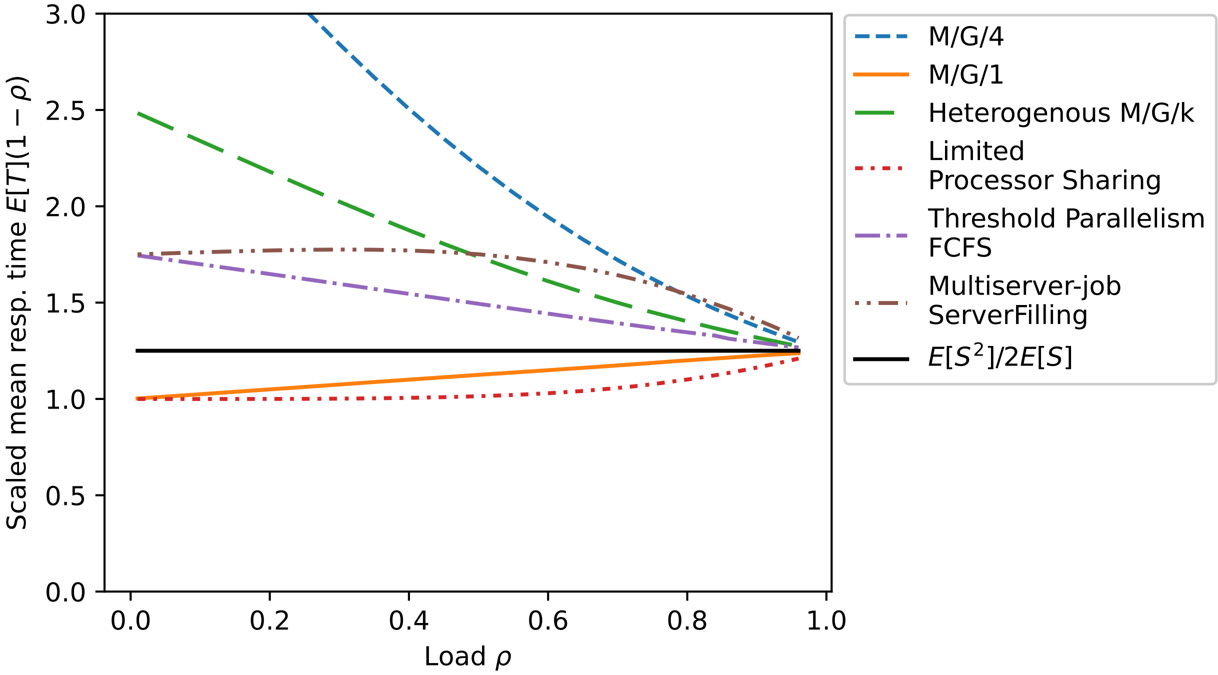


Figure 1: Scaled mean response time of our four motivating models, as well as the related M/G/k and M/G/1 models. Our four motivating models will be further defined in Section 3. In each case, the job size distribution S is distributed as $Hyperexp(\mu_1 = 2, \mu_2 = \frac{2}{3}, p_1 = \frac{1}{2})$. The black line is $E[T](1-\rho) = \frac{E[S^2]}{2E[S]}$, the heavy traffic behavior of M/G/1/FCFS and each of our models of interest. 10^9 arrivals simulated. $\rho \in [0, 0.96]$ to ensure accurate results.

This similarity between the wide variety of models in Fig. 1 is striking. To see just how notable this similarity is, consider a variety of alternative models and policies shown in Fig. 2. For these alternative models, scaled mean response time either does not converge at all, or converges to a different limit entirely.

This contrast poses an intriguing question:

Why do our four motivating models converge to M/G/1/FCFS in heavy traffic?

To put it another way, we ask what crucial property our four motivating models share, that is not shared by the alternative models in Fig. 2.

To answer this question, we define the "work-conserving finite-skip" (WCFS) class of models. The WCFS class contains our four motivating queueing models, as well many others. We demonstrate that for any WCFS model, if the job size distribution S has bounded expected remaining size, then its scaled mean response time converges to the same heavy traffic limit as the M/G/1/FCFS. Specifically, we prove that

Theorem 1. *For any model $\pi \in$ WCFS with bounded expected remaining size*[1]*,*

$$\lim_{\rho \to 1} E[T^\pi](1-\rho) = \frac{E[S^2]}{2E[S]}.$$

Moreover, we prove that the difference in mean response time between any WCFS model and M/G/1/FCFS is bounded by an explicit additive constant, that may depend the specific WCFS model:

[1] This assumption is defined in Section 2.3.

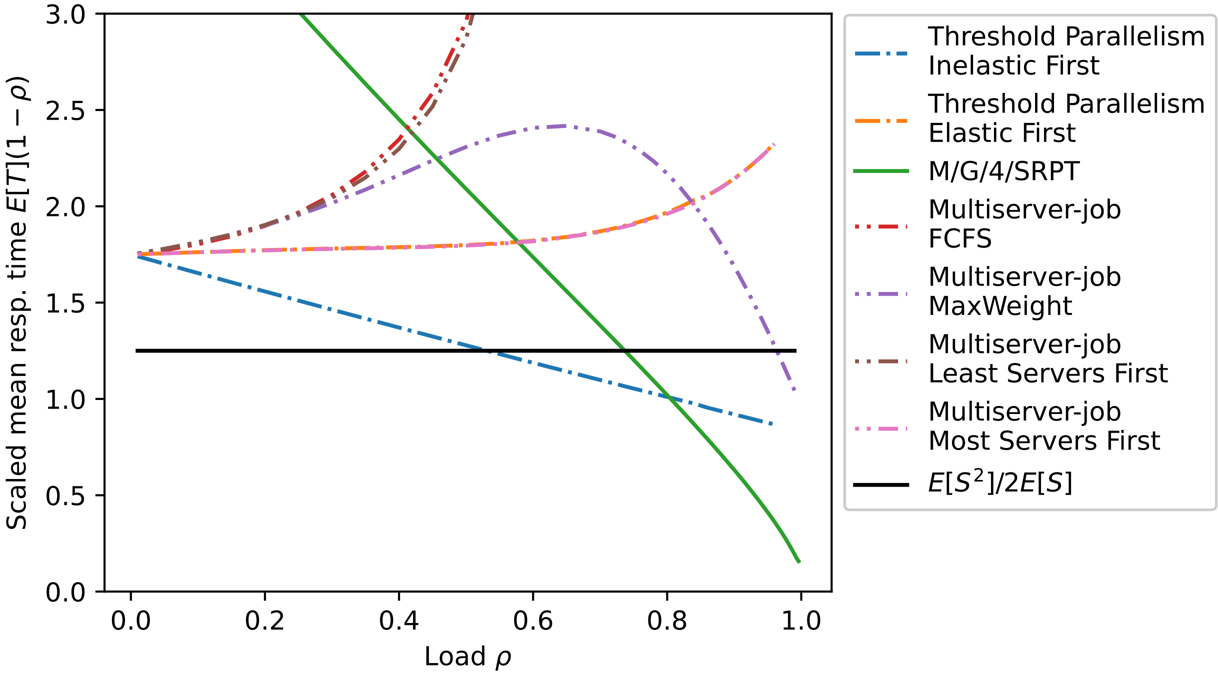


Figure 2: Scaled mean response time of alternative models and policies. All of these models and policies will be explained in Section 6. $S \sim Hyperexp(\mu_1 = 2, \mu_2 = \frac{2}{3}, p_1 = \frac{1}{2})$. Black line is $E[T](1-\rho) = \frac{E[S^2]}{2E[S]}$. 10^9 arrivals simulated, $\rho \in [0, 0.96]$ to ensure accurate results, except MaxWeight: 10^{10} arrivals, $\rho \in [0, 0.99]$.

Theorem 2. *For any model $\pi \in$ WCFS with bounded expected remaining size,*

$$E[T^\pi] \leq \frac{\rho}{1-\rho}\frac{E[S^2]}{2E[S]} + c^\pi_{upper}$$

$$E[T^\pi] \geq \frac{\rho}{1-\rho}\frac{E[S^2]}{2E[S]} + c^\pi_{lower}$$

for explicit constants c^π_{upper} and c^π_{lower} not dependent on load ρ.

Theorem 2 is a stronger version of Theorem 1, implying rapid convergence of scaled mean response time to the heavy traffic limit specified in Theorem 1.

In summary, this paper makes the following contributions:

- We define the WCFS class of models and our bounded expected remaining size assumption. (Section 2)
- We prove that each of the four motivating models is a WCFS model. (Section 3)
- We discuss prior work on WCFS models. (Section 4)
- We prove that all WCFS models with bounded expected remaining size have the same scaled mean response time as M/G/1/FCFS, and specifically have mean response time within an additive constant of M/G/1/FCFS. (Section 5)
- We empirically validate our results, contrasting heavy traffic behavior of WCFS models and non-WCFS models. (Section 6)

2 WCFS Models

In Sections 2.1 and 2.2, we define the WCFS class of models. In Section 2.3, we define our "bounded expected remaining size" assumption. In Section 2.4, we define a few more concepts that will be used in the paper.

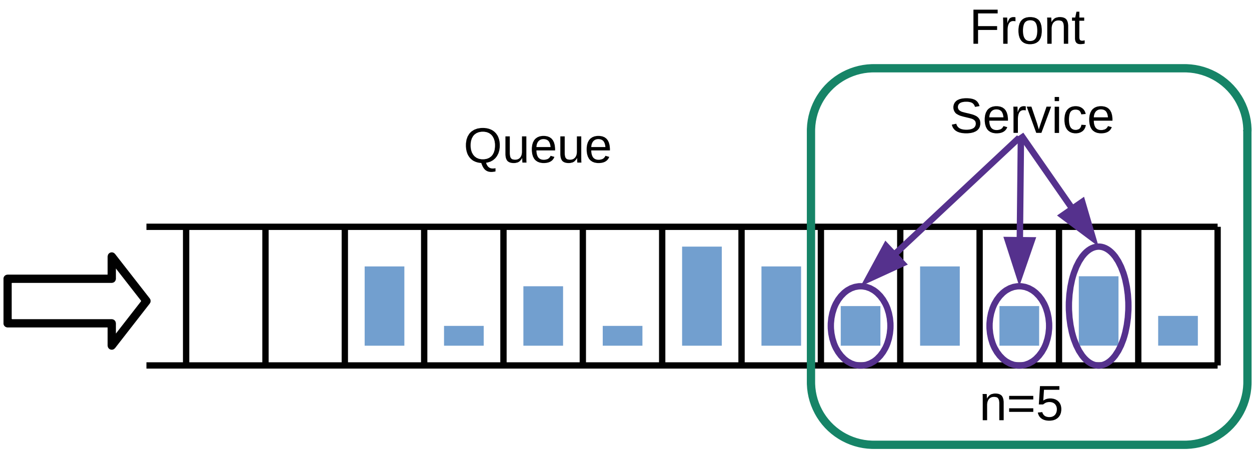


Figure 3: Diagram of a Finite-Skip Model

Throughout this paper, we assume that jobs arrive according to a Poisson process with rate λ. When we consider preemptive service policies, we assume preempt-resume service with no loss of work.

Job sizes are sampled *i.i.d.* from a job size distribution. Let S be a random variable denoting the size of a job. Intuitively, the size of a job represents the amount of work associated with the job. Size will be defined in more detail in Section 2.1.2.

2.1 WCFS Models

We define the WCFS class of models to be models with the following properties:

1. Finite skip (Section 2.1.1),
2. Work conserving (Section 2.1.2),
3. Positive service rate when nonempty (Section 2.1.3).

2.1.1 Finite-skip models

We first define finite-skip models. Consider the jobs in the system in arrival order. Associated with each finite-skip model, there is a finite parameter n. We partition the jobs in the system into two sets: the (up to) n jobs which arrived longest ago, which we call the *front*, and all other jobs, which we call the *queue*. In a *finite-skip model*, among all of the jobs in the system, the server(s) only serve jobs in the front. In particular, no jobs beyond the first n jobs in arrival order receive any service. Fig. 3 shows a generic finite-skip model. In a multiserver system with k servers, it will typically be the case that n is equal to the number of servers k, but this is not required for our results to hold.

The concept of a *full front* is important for working with finite-skip models:

Definition 1. *We call the front* full *if at least n jobs are present in the system, and therefore exactly n jobs are at the front.*

The parameter n denotes the "front size."

The intuition behind the term "finite skip" comes from imagining moving through the jobs in the system in arrival order, skipping over some jobs and serving others. In a finite-skip model, only the first n jobs can be served, so only finitely many jobs can be skipped.

2.1.2 Work conserving

Now, we will specify what we mean by "work conserving," which is a different concept here than in previous work.

First, we normalize the total system capacity to 1, regardless of the number of servers in the system. For instance, in a homogeneous k-server system, we would think of each server as serving jobs at rate $1/k$.

Whenever a job is in service, it receives some fraction of the system's total service capacity, which we call the job's *service rate*. Let $B(t)$ denote the total service rate of all jobs in service at time t. Because the total system capacity is 1, $B(t) \leq 1$ for all t. We also use the random variable B to denote the stationary total service rate.

We define a job's *age* at time t to be the total amount of service the job has received up to time t: a job's age increases at a rate equal to the job's service rate whenever the job is in service. Each job has a property called its *size*. When the job's age reaches its size, the job completes.

In particular, we assume that every job j has a size s_j and a class c_j drawn i.i.d. from some general joint distribution. Let (S, C) be the random variables denoting a job's pair of size and class. A job's class is static information known to the scheduler, while a job's size is unknown to the scheduler. For instance, in the threshold parallelism model defined in Section 3.3, a job's parallelism threshold is its class. In the Heterogeneous M/G/k defined in Section 3.1, all jobs are in the same class.

Definition 2. *We call the system* maximally busy *if the total capacity of the system is in use, namely if the total service rate of jobs in service is 1.*

We define a finite-skip model to be work conserving *if whenever the front is full, the system is also maximally busy.*

In other words, a finite-skip model is work-conserving if whenever there are at least n jobs in the system, the total service rate is 1.

Now that we have defined a job's size, we can also define the load of the system: $\rho = \lambda E[S]$. Load ρ is the long-term average service rate, or equivalently the long-term fraction of capacity in use. Specifically, $\rho = E[B]$. We assume $\rho < 1$ to ensure stability.

2.1.3 Positive service rate when nonempty

We also assume that the total service rate $B(t)$ is bounded away from zero, whenever a job is present. Specifically, whenever a job is present, we assume that $B(t) \geq b_{\text{inf}}$, for some constant $b_{\text{inf}} > 0$.

In contrast to the finite-skip and work-conserving assumptions, this assumption is only present to cover pathological cases. Reasonable queueing models may or may not be finite-skip, and reasonable finite-skip models may or may not be work-conserving, but essentially all reasonable models have positive service rate when nonempty.

That being said, this assumption is key to bounding mean response time under low load. For an example, see the batch-processing system in Section 2.2.

2.2 Examples and non-examples

To clarify which models are WCFS models, we give several examples, both positive and negative.

- **M/G/k/FCFS:** This is a work-conserving finite-skip model. Here we set the parameter n equal to k. The front consists of the jobs in service. Each job in the front is served at service rate $1/k$. The system is maximally busy (total service rate 1) exactly when it is full, namely when at least k jobs are present in the system.

  It also has service rate at least $1/k$ whenever a job is present.

- **M/G/∞:** This model is not finite skip. All jobs are in service, regardless of the number of jobs in the system: there is no finite bound on the number of jobs in service.

- **M/G/k/SRPT:** In this model, the k jobs with smallest remaining size are served at rate $1/k$. This model is not finite skip. This is because the jobs with smallest remaining size can be arbitrarily far back in the arrival ordering of jobs in the system.

- **Multiserver-job model:** Consider a multiserver system with $k = 2$ servers, and where each job requires either 1 or 2 servers.

  We now consider two finite skip service policies for this system, each with front size $n = 2$. First, consider serving jobs in FCFS order, with head-of-the-line blocking. This policy is finite-skip, but it is not work-conserving: if the front consists of a job requiring 1 server followed by a job requiring 2 servers, the system will only utilize one server. In this case, the system is full, because $n = 2$ jobs are present in the system, and hence in the front, but the system is not maximally busy.

  In contrast, consider a service policy which serves a 2 server job if either of the jobs in the front are 2 server jobs, or else serves each of the 1 server jobs at the front. This policy is a special case of the ServerFilling policy, depicted in Fig. 1 and defined in general in Section 3.4.2. This policy is finite-skip and work-conserving. If the front is full, then for any pair of jobs in the front, either one 2-server job or two 1-server jobs will be served, making the system maximally busy.

- **Batch-processing M/G/k:** If there are at least k jobs present, the oldest k jobs in the system are each served at rate $\frac{1}{k}$. Otherwise, no service occurs. This model is finite-skip and work-conserving, but does not satisfy the nonzero service rate assumption. To see why this assumption is necessary for our main results, specifically Theorem 2, one can show that in the $\lambda \to 0$ limit, response times will grow arbitrarily large in the batch-processing M/G/k. In particular, one can show that $E[N]$ approaches a constant in the $\lambda \to 0$ limit, so $E[T]$ diverges by Little's law. To rule out systems where $E[T]$ diverges in the $\lambda \to 0$ limit, we make the nonzero service rate assumption.

2.3 Bounded expected remaining size: Finite rem_{sup}

Recall that a job j has i.i.d. size s_j and class c_j, which are realizations of the random variables (S, C). We assume that the service policy is based only on jobs' classes and ages.[2] At a given point in time, the *state* of a job j consists of its class c_j and its age a_j. We also allow service to be based on the states of the jobs in the front, but not on the number or states of jobs in the queue.

A key assumption we make is that jobs have bounded expected remaining size from an arbitrary state. Let S_c be the job size distribution for jobs of class $c \in C$. We define $\text{rem}_{\text{sup}}(S, C)$ to be the supremum over the expected remaining sizes of jobs, taken over all states:

$$\text{rem}_{\text{sup}}(S, C) := \sup_{c \in C, a \in \mathbb{R}^+} E[S_c - a \mid S_c > a].$$

When size S is independent from class C, or when a model has no class information, we will simply write $\text{rem}_{\text{sup}}(S)$.

In this paper, we focus on job size distributions for which $\text{rem}_{\text{sup}}(S, C)$ is finite.

To better understand the finite $\text{rem}_{\text{sup}}(S, C)$ assumption, let's walk through a couple of examples. In all of these examples, let's suppose that the class information is independent of the job size distribution S, so we can simply write $\text{rem}_{\text{sup}}(S)$.

Consider a job size distribution S that is hyperexponential:

$$S = \begin{cases} Exp(\mu_1) & \text{w.p. } p_1 \\ Exp(\mu_2) & \text{w.p. } p_2 \\ Exp(\mu_3) & \text{w.p. } p_3 \end{cases}$$

For all ages a, the expected remaining size is bounded:

$$E[S - a \mid S > a] \leq \frac{1}{\min(\mu_1, \mu_2, \mu_3)}.$$

In fact, $\text{rem}_{\text{sup}}(S) = \frac{1}{\min(\mu_1, \mu_2, \mu_3)}$.

More generally, an arbitrary phase type job size distribution S' must have finite rem_{sup}. To see why, note that starting from an arbitrary phase, the expected remaining size is finite. Let r be the maximum expected remaining size from any phase. It is straightforward to show that

$$E[S' - a \mid S' > a] \leq r, \forall a$$
$$\text{rem}_{\text{sup}}(S') \leq r.$$

On the other hand, Pareto job size distributions do not have finite rem_{sup}. Let $S'' \sim Pareto(\alpha = 3, x_{\min} = 1)$.

$$E[S'' - a \mid S'' > a] = \frac{a}{2}, \forall a \geq 1$$
$$\lim_{a \to \infty} E[S'' - a \mid S'' > a] = \infty$$
$$\text{rem}_{\text{sup}} = \sup_a E[S'' - a \mid S'' > a] = \infty$$

Note that $\text{rem}_{\text{sup}}(S'')$ is infinite, despite the fact that $E[S'']$ and $E[(S'')^2]$ are both finite.

In general, finite rem_{sup} roughly corresponds to service time having an exponential or sub-exponential tail, though there are some subtleties. For instance, a Weibull distribution with $P(S \geq a) = a^{-k}$ for some $k < 1$ has infinite rem_{sup}, while for $k \geq 1$, rem_{sup} is finite.

[2] The WCFS model can also be generalized to known-size service policies, but for simplicity we do not do so in this paper.

2.4 Work, Number, Response Time

Let the *work* in the system be defined as the sum of the remaining sizes of all jobs in the system. A job's remaining size is its size minus its age. We defined size and age in Section 2.1.2.

Let $W(t)$ be the total work in the system at time t. Let $W_Q(t)$ and $W_F(t)$ be the work in the queue and the work at the front, respectively, at time t. We will generally use the subscripts $_Q$ and $_F$ to denote the queue and the front. Let W, W_Q, and W_F denote the corresponding time-stationary random variables.

Recall from Section 2.1.2 that $B(t)$ is the total service rate at time t. Note that $\frac{d}{dt}W(t) = -B(t)$, except at arrival instants.

Let $N(t)$ be the number of jobs in the system at time t. Note that $N_F(t) = n$ whenever $N(t) \geq n$, because the front is full, and $N_F(t) = N(t)$ otherwise.

Let T be a random variable denoting a job's time-stationary response time: the time from when a job arrives to when it completes.

3 Important WCFS Models

Here we define in more detail the four motivating models mentioned in the introduction and depicted in Fig. 1, and show that each is a WCFS model.

3.1 Heterogeneous M/G/k

The heterogeneous M/G/k models multiserver systems where servers may have different speeds. This scenario commonly arises in datacenters, which are often composed of servers with a wide variety of different types of hardware, leading to heterogeneous performance [33, 35]. In the mobile device setting, the big.LITTLE architecture employs heterogeneous processors to improve battery life [10].

To define the heterogeneous M/G/k, let each server i have speed $v_i > 0$. We normalize the server speeds so that $\sum_i v_i = 1$. While a job is being served by server i, the job's age increases at a rate of v_i, completing when its age reaches its size: If job j has size s_j, and runs at server i, the job will complete after $\frac{s_j}{v_i}$ time in service.

In the heterogeneous M/G/k, jobs enter service in FCFS order. If there are multiple servers open when a job arrives, a service is chosen according to a server assignment policy. There are many such policies, including Fastest Server First, Random Server Assignment, and Slowest Server First [2, 13].

We do not focus on any particular assignment policy. We only assume that jobs are served in FCFS order, and that no job is left waiting while a server is idle. Under these assumptions, all assignment policies give rise to WCFS models, so our results apply to all of them.

As an example, in Fig. 1 we show the scaled mean response time of a heterogeneous M/G/4 with server speeds $0.4, 0.3, 0.2, 0.1$, and the Preemptive Fastest Server First assignment policy.

3.1.1 Heterogeneous M/G/k is a WCFS model

To show that the heterogeneous M/G/k is a WCFS model, we must verify the three properties from Sections 2.1.1 to 2.1.3.

Finite skip: Jobs enter service in FCFS order. As a result, the jobs in service are exactly the (up to) k oldest jobs in the system. The model is finite skip with parameter $n = k$.

Work conserving: The system has total capacity $\sum_i v_i = 1$. Whenever at least k jobs are present in the system, all servers are occupied, and the total service rate is 1. In other words, whenever the front is full, the system is maximally busy.

Positive service rate when nonempty: If a job is present, the job will be in service on some server. The system will therefore have minimum service rate $b_{\inf} \geq v_{\min}$, where $v_{\min} = \min_i v_i$.

3.2 Limited Processor Sharing

The Processor Sharing policy for the M/G/1 is of great theoretical interest, and has been extensively studied [49]. However, in real systems, running too many jobs at once causes a significant overhead. A natural remedy is to set a Multi-Programming Level k, and only processor-share up to k jobs at a time, in arrival order. This policy is known as Limited Processor Sharing (LPS).

The LPS policy is parameterized by some Multi-Programming Level k. If at most k jobs are present in the system, then the policy is equivalent to Processor sharing, serving all jobs at an equal rate, with total service rate 1. When more than k jobs are present, the k oldest jobs in FCFS order are each served at rate $1/k$.

As an example, in Fig. 1 we show the scaled mean response time of a LPS system with MPL 4.

3.2.1 Limited Processor Sharing is a WCFS model

Finite skip: The jobs in service are exactly the (up to) k oldest jobs in the system. As a result, the model is finite skip with parameter $n = k$.

Work conserving: Whenever at least one job is present in the system, the total service rate is 1, and the system is maximally busy. As a special case, whenever at least k jobs are present, making the front full, the system is maximally busy.

Positive service rate when nonempty: If a job is present, the system is maximally busy. As a result, $b_{\text{inf}} = 1$.

3.3 Threshold Parallelism

In modern datacenters, it is increasingly common for jobs to be parallelizable across a variety of different numbers of servers, where the level of parallelism is chosen by the scheduler [12, 37]. The Threshold Parallelism setting models this scenario by assuming that for each job, the user gives the ideal, maximum number of servers that the job can utilize.

The Threshold Parallelism model is a multiserver queueing model in which different jobs can be parallelized across different numbers of servers. A job j has two characteristics: Its size s_j and its parallelism threshold ℓ_j, where ℓ_j is some number of servers. Job j may be parallelized across up to ℓ_j servers, with linear speedup. The pair (s_j, ℓ_j) is sampled i.i.d. from some joint distribution (S, L). Note that ℓ_j is the class of the job j.

Let k be the total number of servers. Note that $\ell_j \in [1, k]$. If a job j is served on $q \leq \ell_j$ servers, then it receive service rate $\frac{q}{k}$ and would complete after $\frac{ks_j}{q}$ time in service. In this model, the number of servers a job receives can change over time, correspondingly changing its service rate.

We focus on the FCFS service policy. Under this policy, jobs are placed into service in arrival order until their total parallelism thresholds sum to at least k. Any job j which is not the newest job in service is served by ℓ_j servers. The newest job in service is served by the remaining servers.

As an example, in Fig. 1 we show the scaled mean response time of a Threshold Parallelism model where the joint distribution (S, L) is $(Exp(2), 1)$ with probability $\frac{1}{2}$, and $(Exp(\frac{2}{3}), 4)$ with probability $\frac{1}{2}$, and with FCFS service.

As a comparison, in Fig. 2, we show Threshold Parallelism models with the same joint distribution (S, L), but with different service policies: "Elastic First," prioritizing jobs with $L = 1$, and "Inelastic First," prioritizing jobs with $L = 4$. These are not WCFS policies.

3.3.1 Threshold Parallelism with FCFS service is a WCFS model

Finite skip: The jobs in service are the initial set of jobs in arrival order whose parallelism thresholds sum to at least k. This initial set can contain at most k jobs, because every job has parallelism threshold at least 1. As a result, the model is finite skip with parameter $n = k$.

Work conserving: Whenever jobs are present in the system whose parallelism thresholds sum to at least k, all servers are occupied, and the system is maximally busy. Whenever k jobs are present, the system must be maximally busy.

Positive service rate when nonempty: If a job is present in the system, at least one server must be occupied, and so the service rate is at least $1/k$. Hence $b_{\text{inf}} \geq 1/k$.

3.4 Multiserver-jobs under the ServerFilling policy

First, we will describe the multiserver-job setting. Then we will specify the ServerFilling policy.

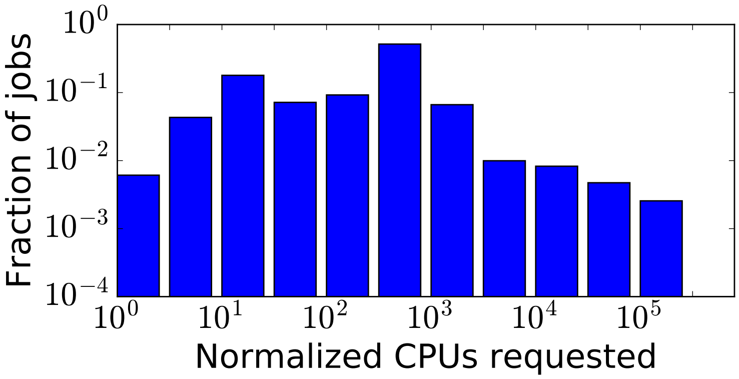


Figure 4: The distribution of number of CPUs requested in Google's recently published Borg trace [47]. Number of CPUs is normalized to the size of the smallest request observed, not an absolute value.

3.4.1 Multiserver-Job Setting

When we look at jobs in cloud computing systems [32] and in supercomputing systems [9, 15, 45], each job typically requires an exact number of servers for the entire time the job is in service. To illustrate, in Fig. 4 we show the distribution of the number of CPUs requested by the jobs in Google's recently published trace of its "Borg" computation cluster [21, 47]. The distribution is highly variable, with jobs requesting anywhere from 1 to 100,000 normalized CPUs[3].

The Multiserver-Job (MSJ) model is a natural model for these computing systems. The MSJ model is a multiserver queueing model where each job requires a fixed number of servers. As a result, the jobs are called "multiserver jobs."

A job j has two requirements: A number of servers v_j and an amount of time x_j, which are sampled i.i.d. from some joint distribution with random variables (V, X). If job j requires v_j servers, then it can only be served when exactly v_j servers are allocated to it. The job will complete after x_j time in service.

Let a job j's size be defined as

$$s_j = \frac{v_j x_j}{k} \qquad S = \frac{VX}{k}.$$

Whenever job j is in service, its service rate is $\frac{v_j}{k}$, ensuring that it completes in time x_j.

There are a wide variety of possible service policies for placing jobs at open servers, including FCFS, MaxWeight, Most Servers First and many others. We define these policies in Section 6.

As examples, in Fig. 2, we show the scaled mean response time of Multiserver-Job models under a variety of service policies, where the joint distribution (V, X) is $(1, Exp(\frac{1}{2}))$ with probability $\frac{1}{2}$, and $(4, Exp(\frac{2}{3}))$ with probability $\frac{1}{2}$.

However, no existing policies result in a WCFS model – all existing policies, including those shown in Fig. 2, are either non-finite-skip, such as Most Servers First, or non-work-conserving, such as FCFS. Correspondingly, in Fig. 2, we see that no existing policy has its scaled mean response time converge to the same limit as $M/G/1/FCFS$.

We therefore define a novel service policy called ServerFilling which yields a WCFS model. This service policy is depicted in Fig. 1, with the same joint distribution (V, X) as the policies shown in Fig. 2.

3.4.2 ServerFilling

For simplicity, we initially define the Server Filling policy for the common situation in computer systems where all jobs require a number of servers which is a power of 2 (V is always a power of 2), and where k is also a power of 2. We discuss generalizations in Section 3.4.4.

First, ServerFilling designates a candidate set M, consisting of the minimal prefix (i.e. initial subset) of the jobs in the system in arrival order which collectively require at least k servers. If all jobs in the system collectively require fewer than k servers, then all are served. Note that $|M| \leq k$ because all jobs require at least 1 server.

[3] The data was published in a scaled form [47]. We rescale the data so the smallest job in the trace uses one normalized CPU.

For instance, if $k = 8$ and the jobs in the system require $[1, 2, 1, 1, 4, 2, 2, 1]$ servers, in arrival order (reading from left to right), then M would consist of the first 5 jobs: $[1, 2, 1, 1, 4]$, which collectively require 9 servers. A shorter prefix would require at most 5 servers.

Next, the jobs in M are ordered by their server requirements v_j, from largest to smallest, tiebroken by arrival order. In that order, jobs are placed into service until no more servers are available.

For instance, if $k = 8$ and M contains jobs requiring $[1, 2, 1, 1, 4]$ servers, then jobs requiring $4, 2, 1$, and 1 server(s) would be placed into service.

To show that ServerFilling is WCFS, we must show that this procedure always utilizes all k servers.

Lemma 1. *Let M be a set of jobs such that $\sum_{j \in M} v_j \geq k$, where each $v_j = 2^i$ for some i and $k = 2^{i'}$ for some i'. Label the jobs $m_1, m_2, \ldots$ in decreasing order of server requirement: $v_{m_1} \geq v_{m_2} \geq \ldots$. Then there exists some index $\ell \leq |M|$ such that*

$$\sum_{j=1}^{\ell} v_{m_j} = k.$$

Proof. Let $\textsc{req}(z)$ count the number of servers required by the first z jobs in this ordering:

$$\textsc{req}(z) = \sum_{j=1}^{z} v_{m_j}.$$

We want to show that $\textsc{req}(\ell) = k$ for some ℓ. To do so, it suffices to prove that:

$$\text{There exists no index } \ell' \text{ such that both } \textsc{req}(\ell') < k \text{ and } \textsc{req}(\ell' + 1) > k. \tag{1}$$

Equation (1) states that $\textsc{req}(z)$ cannot cross from below k to above k without exactly equalling k. Because $\textsc{req}(0) = 0$ and $\textsc{req}(|M|) \geq k$, $\textsc{req}(\ell)$ must exactly equal k for some ℓ.

To prove (1), let us examine the quantity $k - \textsc{req}(z)$, the number of remaining servers after z jobs have been placed in service. Because all v_js are powers of 2, $k - \textsc{req}(z)$ carries an important property:

$$\text{For all } z, k - \textsc{req}(z) \text{ is divisible by } v_{m_{z+1}}. \tag{2}$$

We write $a|b$ to indicate that a divides b.

We will prove (2) inductively. For $z = 0$, $k - \textsc{req}(0) = k$. Because k is a power of 2, and v_{m_1} is a power of 2 no greater than k, the base case holds. Next, assume that (2) holds for some index z, meaning that $v_{m_{z+1}} | (k - \textsc{req}(z))$. Note that $\textsc{req}(z+1) = \textsc{req}(z) + v_{m_{z+1}}$. As a result, $v_{m_{z+1}} | (k - \textsc{req}(z+1))$. Now, note that $v_{m_{z+2}} | v_{m_{z+1}}$, because both are powers of 2, and $v_{m_{z+2}} \leq v_{m_{z+1}}$. As a result, $v_{m_{z+2}} | (k - \textsc{req}(z+1))$, completing the proof of (2).

Now, we are ready to prove (1). Assume for contradiction that there does exist such an ℓ'. Then $k - \textsc{req}(\ell') > 0$, and $k - \textsc{req}(\ell' + 1) < 0$. Because $\textsc{req}(\ell' + 1) = \textsc{req}(\ell') + v_{m_{\ell'+1}}$, we therefore know that $v_{m_{\ell'+1}} > k - \textsc{req}(\ell')$. But from (2), we know that $v_{m_{\ell'+1}}$ divides $k - \textsc{req}(\ell')$. A larger positive integer cannot divide a smaller positive integer, so this is a contradiction, as desired. □

3.4.3 ServerFilling for the Multiserver-Job system is a WCFS policy

Finite skip: The jobs in service are a subset of the candidate set M, the initial set of jobs in arrival order whose server requirements v_j sum to at least k. This initial set must contain at most k jobs, because every job requires at least 1 server. As a result, the model is finite skip with parameter $n = k$.

Work conserving: By Lemma 1, whenever jobs are present in the system whose server requirements v_j sum to at least k, all servers are occupied, and the system is maximally busy. Thus, whenever k jobs are present, the system must be maximally busy.

Positive service rate when nonempty: If a job is present in the system, at least one server must be occupied, and so the service rate is at least $1/k$. Hence $b_{\inf} \geq 1/k$.

3.4.4 Generalizations of ServerFilling

The ServerFilling policy can be generalized beyond the situation where all job server requirements are powers of 2, as is k. If all job server requirements are powers of some integer a, as is k, then Lemma 1 still holds, with an identical proof. More generally, if each server requirement divides all larger server requirements, as well as k, then Lemma 1 still holds, again with an identical proof.

Beyond ServerFilling, a WCFS policy can also be defined whenever all server requirements divide k. Specifically, we give an algorithm which selects, from a set of k jobs whose server requirements divide k, a subset whose server requirements sum to exactly k. We describe this algorithm and the resulting WCFS policy, which we call DivisorFilling, in Appendix A.

DivisorFilling is the most general possible WCFS policy for the MSJ setting. If some server requirement does not divide k, then no WCFS policy exists, because the system cannot be work conserving if all jobs require that non-divisible number of servers.

4 Prior Work

4.1 M/G/k

The simplest WCFS model is the M/G/k, one of the most heavily studied models in all of queueing theory. We are interested in bounds on mean response time in the M/G/k, particularly bounds that are tight in the heavy traffic regime, where $\rho \to 1$ and k is constant.

4.1.1 Fixed k

In this regime, the best known bounds either require much stronger assumptions on the job size distribution S than we assume [30], or prove much weaker bounds on mean response time [27, 28].

A paper by Loulou [30] bounds mean work in system in the M/G/k to within an additive gap, under the strong assumption that the job size distribution S is bounded. This result is comparable to our Lemma 3, but in a simpler setting and under stronger assumptions.

While the paper mostly focuses on the overload regime ($\rho > 1$), their equations (9) and (10) apply in our setting ($\rho < 1$) as well. They couple the multiserver system with a single-server system on the same arrival sequence. They show that

$$0 \leq W^{M/G/k}(t) - W^{M/G/1}(t) \leq k \max_{1 \leq i \leq A(t)} S_i,$$

where $A(t)$ is the number of jobs that have arrived by time t. In the case of a bounded job size distribution S, one can therefore show that

$$0 \leq W^{M/G/k}(t) - W^{M/G/1}(t) \leq k \sup(S). \tag{3}$$

One could then use this workload bound to prove a bound on mean response time in the M/G/k.

Our workload bounds in Lemma 3 are comparable in tightness to (3), but follow from a much weaker assumption on the job size distribution S. We merely assume that the job size distribution S has bounded rem_{sup}, in contrast to Loulou's assumption that S itself is bounded.

Köllerström [27] proves convergence of queueing time to an exponential distribution in the GI/GI/k. Specialized to the M/G/k, the result states that in the $\rho \to 1$ limit, $T_Q^{M/G/k}$ converges to an exponential distribution with mean

$$\frac{\rho}{1-\rho}\frac{E[S^2]}{2E[S]} - \frac{1}{\lambda} = E[T_Q^{M/G/1}] - \frac{1}{\lambda}.$$

Köllerström [28] improves upon [27] by characterizing the rate of convergence to the limiting exponential distribution, and thereby derives explicit moment bounds. However, unlike prior single-server results [26], these moment bounds are quite weak. Specialized to the M/G/k, Köllerström [28]'s bounds on $E[T_Q]$ state that

$$E[T_Q^{M/G/k}] - E[T_Q^{M/G/1}] \geq \frac{c_{lower}}{(1-\rho)^{1/2}} \tag{4}$$

$$E[T_Q^{M/G/k}] - E[T_Q^{M/G/1}] \leq \frac{c_{higher}}{1-\rho} \tag{5}$$

for constants c_{lower}, c_{higher} not dependent on ρ.

The $\Theta(\frac{1}{1-\rho})$ scaling in (5) is especially poor: this bound is too weak to give any explicit bound on the convergence rate of $E[T_Q^{M/G/k}](1-\rho)$ to the previously established limit of $\frac{E[S^2]}{2E[S]}$.

We prove much tighter bounds on $E[T_Q^{M/G/k}]$ in Lemma 4. Our bounds replace the right hand sides of (4) and (5) with explicit constants, not dependent on ρ. To achieve our stronger result, we make a stronger assumption on the job size distribution S than Köllerström [28]. We assume S has finite rem_{sup}, while Köllerström [28] merely assumes that S has finite second moment. With our stronger assumption, we not only prove vastly tighter bounds on mean response time, but also prove such bounds far beyond the M/G/k.

4.1.2 Scaling k

More recent work has focused on regimes where both ρ and k scale asymptotically, such as the Halfin-Whitt regime. These results are not directly comparable to ours. Recent results in the Halfin-Whitt regime indicate that the limiting behavior depends in a complex way on the job size distribution S [1, 11, 17].

Turning to the more general case of scaling k, in work currently under submission, Goldberg and Li [19] prove the first bounds on $E[T_Q]$ that scale as $\frac{c}{1-\rho}$ for an explicit constant c and arbitrary k as a function of ρ. Unfortunately, the constant c is enormous, scaling as $10^{450}E[S^3]$. These results indicate how difficult the problem becomes in the regime where k scales with ρ. In contrast, we focus on the regime of fixed k, and prove tight and explicit bounds on mean response time. Goldberg and Li [19] also provide a highly detailed literature review on bounds on $E[T_Q]$ and related measures in the M/G/k and related models.

4.2 Heterogeneous M/G/k

4.2.1 Heterogeneous M/M/k

Much of the previous work on multiserver models with heterogeneous service rates has focused on the much simpler M/M/k setting, where jobs are memoryless, so the only variation is between the server speeds [2, 13, 14, 29]. In this model, one can analyze the preemptive Fastest-Server-First policy to derive a natural lower bound on the mean response time of any server assignment policy. One can similarly analyze the preemptive Slowest-Server-First policy to derive an upper bound. These two policies each lead to a single-dimensional birth-death Markov chain, allowing for straightforward analysis [2]. One can think of our bounds as essentially extending these bounds for the M/M/k to the much more complex setting of the M/G/k.

4.2.2 Heterogeneous M/H$_m$/k

Van Harten and Sleptchenko [48] primarily study a *homogeneous* multiserver setting with hyperexponential job sizes. However, in their conclusion, they mention that their methods could be extended to a setting with heterogeneous servers, but at the cost of making their Markov chain grow exponentially. This exponential blowup seems inevitable when applying exact Markovian methods to a heterogeneous setting with differentiated jobs.

4.2.3 M/(M+G)/2 Model

Another intermediate model is the M/(M+G)/2 model of Boxma et al. [7]. In this model, jobs are not differentiated. Instead, the service time distribution is entirely dependent on the server. Server 1, the first server to be used, has an exponential service time distribution, while server 2 has a general service time distribution. Boxma et al. [7] make partial progress by deriving an implicit expression for the Laplace-Stieltjes transform of response time in this setting, which they are only able to make explicit when the general service time distribution has rational transform. Subsequent work has fully solved the M/(M+G)/2 model, giving exact stationary distributions under both FCFS service and related service disciplines [25, 40, 42].

Our results are not directly applicable to the M/(M+G)/2 setting, because the servers have different distributions of service time, not just different speeds. However, the slow progress on this two-server model illustrates the immense difficulty of applying prior methods to any but the simplest heterogeneous multiserver models. In contrast, our finite-skip technique handles both differentiated jobs and an arbitrary number of servers with no additional effort.

4.3 Limited Processor Sharing

The Limited Processor Sharing policy has been studied by a wide variety of authors [22, 36, 46, 50, 51, 52], but none have proven results that bound mean response time for all loads ρ.

4.3.1 Asymptotic Regimes

A series of papers by Zhang, Dai and Zwart [50, 51, 52] derive the strongest known results on Limited Processor Sharing in a variety of asymptotic regimes. In their three papers, the authors derive the measure-valued fluid limit [51], the diffusion limit [52] and a steady-state approximation [50], which they prove is accurate in the heavy traffic limit ($\rho \to 1$).

Of their results, the most comparable to our work is their steady-state approximation. When specialized to mean response time in the M/G/1, their approximation states that

$$E[T] \approx \frac{E[S]}{1-\rho}(1-\rho^k) + \frac{E[S^2]}{2E[S]}\frac{\rho^k}{1-\rho}$$

While they prove that this approximation is accurate in the heavy-traffic limit, they give no specific error bounds. However, it is empirically an excellent approximation at all load ρ [50]. Our results therefore complement the results of Zhang et al., by proving concrete error bounds, in contrast to their approximation.

4.3.2 State-dependent Server Speed

In order to model the behavior of databases, Gupta and Harchol-Balter [22] introduce a variant of the Limited Processor Sharing model, where the total server speed is a function of the number of jobs in service. They focus on a setting where server speed increases to a peak, and then slowly declines as more jobs enter service. They derive a two-moment approximation for mean response time, and use it to derive a heuristic policy for choosing the Multi-Programming Level (MPL). However, this two-moment approximation is not known to be tight. Nonetheless, it indicates that the optimal MPL for minimizing mean response time may be significantly larger than the service-rate-maximizing MPL, if job size variability is large and load is not too high.

Using our WCFS framework it is possible to derive bounds on mean response time for the state-dependent server speeds setting. For MPL parameters less than or equal to the service-rate-maximizing MPL, both our upper and lower bounds apply, while if the MPL parameter is greater than the service-rate-maximizing MPL, only our upper bounds apply.

Subsequently, Telek and Van Houdt [46] derive the Laplace-Stieltjes transform of response time in the LPS model with state-dependent server speed, under phase-type job sizes. Unfortunately, the transform takes the form of a complicated matrix equation. As a result, it is difficult to derive general insights from this result across general job size distributions. Instead, the authors numerically invert the Laplace transform for a handful of specific distributions to derive empirical insights. This is in contrast to our simple, explicit and tight bounds on mean response time in Theorem 2.

4.4 Threshold Parallelism

Berg et al. [5] [4] introduce the concept of "speedup functions" to capture the common situation in Machine Learning and other highly parallel computing settings where different jobs can be parallelized to different degrees. One important kind of speedup function is the Threshold Parallelism model, where the service rate of a job is a concave sublinear function of the number of servers it receives. However, results are only known in the multiple speedup function model in the setting where the job size distribution is exponential, and there are exactly two speedup functions, which corresponds to exactly two parallelism thresholds. In this setting, the optimal policy is shown to be one called "GREEDY*," which corresponds to the policy which preemptively prioritizes the jobs with smaller parallelism threshold. Even with these restrictions, no analytic bounds on response time are known. Our bounds apply to general job size distributions, and arbitrary parallelism thresholds, under FCFS service.

4.4.1 Elastic and Inelastic Jobs

A special case of the Threshold Parallelism policy is the Elastic/Inelastic model of Berg et al. [6]. This model assumes that all jobs are either "inelastic," with parallelism threshold 1, or "elastic," with parallelism threshold k. They also assume that inelastic jobs have size distributed as $Exp(\mu_I)$, and elastic jobs have size distributed as $Exp(\mu_E)$, with sizes unknown to the scheduler. They focus on two preemptive-priority service policies for this setting: Inelastic First (IF) and Elastic First (EF). They prove that if $\mu_I \geq \mu_E$, then IF is the optimal service policy for minimizing mean response time, over all possible policies. They empirically show that if $\mu_I < \mu_E$, then EF often has lower mean response time than IF. They also perform an approximate response time analysis of EF and IF with a combination of the Busy-Period Transitions technique and Matrix-Analytic methods, to overcome the difficulties of a multidimensional Markov chain. This gives a numerical approximation that is empirically within 1% of simulation.

Our bounds on Threshold Parallelism with FCFS service are the first analytic bounds for any service policy and any parallelism thresholds, subsuming the Elastic/Inelastic setting. Our bounds thus form a baseline for judging the performance of policies like IF and EF. Moreover, we handle arbitrary parallelism thresholds, not just 1 and k.

4.5 Multiserver Jobs

The Multiserver-Job model has been extensively studied, in both practical [9, 15, 45] and theoretical settings [8, 18, 24, 31, 32, 38, 39, 41]. It captures the common scenario in datacenters and supercomputing where each job requires a fixed number of servers in order to run. Characterizing the stability region of policies in this model is already a challenging problem, and there were no bounds on mean response time for any policy in this model, prior to our bound on ServerFilling.

4.5.1 FCFS Scheduling

The most natural policy is FCFS scheduling, where the oldest jobs are placed into service until a job requires more servers than remain, at which point the queue is blocked until the job at the head of the queue can enter service. Therefore, the FCFS policy can leave a large number of servers idle even when many jobs are present. As a result, one can show that FCFS does not in general achieve an optimal stability region. Even worse, deriving the stability region of FCFS is major open problem, and has only been achieved in a few special cases [8, 41].

One technique that may be useful towards characterizing this stability region is the *saturated system* approach [3, 16]. The saturated system is a system in which additional jobs are always available, so the front is always full. Only the composition of jobs in the front varies. The completion rate of the saturated system exactly matches the stability region of the equivalent open system, under a wide variety of arrival processes. Unfortunately, solving the general Multiserver-job FCFS saturated system seems intractable.

Given the difficulty of proving results under the FCFS scheduling policy, policies with better theoretical guarantees, such as ServerFilling, are desirable.

4.5.2 MaxWeight Scheduling

One natural throughput-optimal policy is the MaxWeight service policy [32]. To express this policy, divide all jobs into classes based on server requirements. Let $N_i(t)$ be the number of jobs requiring i servers in the system at time t. Next, consider the set M of all packings of jobs onto servers. Let m be one particular packing, and let m_i be the number of jobs requiring i servers served by packing m.

The MaxWeight service policy picks the packing m which maximizes

$$\max_m \sum_i N_i(t) m_i.$$

While MaxWeight is throughput optimal, it has a major drawback: it is very computationally intensive to implement, requiring the scheduler to solve an NP-hard optimization problem whenever a job arrives or departs. For comparison, ServerFilling is also throughput-optimal given its assumptions on the server requirements V, but it is far computationally simpler, requiring approximately linear time as a function of k. Moreover, no bounds on mean response time are known on MaxWeight, due in part to its high complexity.

4.5.3 Nonpreemptive Scheduling

In certain practical settings such as supercomputing, a nonpreemptive service policy is preferred. In such settings, a backfilling policy such as EASY backfilling or conservative backfilling is often used [9, 15, 45]. Backfilling policies start by serving jobs in FCFS order, until a job is reached that requires more servers than remain. At this point, newer jobs that require fewer servers are scheduled, but only insofar as this will not delay older jobs, based on user-provided service time upper bounds. While these policies are popular in practice, little is known about them theoretically, including their response time characteristics.

Finding any nonpreemptive throughput-optimal policy is already a challenging problem. Several such policies have been designed [18, 31, 39], typically by slowly shifting between different server configurations to alleviate overhead. Because such policies can have very large renewal times, many jobs can back up while the system is in a low-efficiency configuration. This can empirically lead to very high mean response times. However, no theoretical mean response time analysis exists for any policy in the Multiserver-Job setting. As a result, there is no good baseline policy to compare against novel policies, and it is thus impossible to tell whether a policy has low mean response time in a comparative or absolute sense. Our bounds on the mean response time of ServerFilling can therefore serve as such a baseline, albeit in the more permissive setting of preemptive scheduling.

5 Theorems and Proofs

Our goal is to derive a heavy-traffic analysis all WCFS models, under the assumption of finite $\text{rem}_{\sup}(S, C)$. Specifically, we want to prove that the scaled mean response time of any WCFS model converges to the same constant as an M/G/1/FCFS:

Theorem 1 (Heavy Traffic response time)**.** *For any model $\pi \in$ WCFS, if $\text{rem}_{\sup}(S, C)$ is finite,*

$$\lim_{\rho \to 1} E[T^\pi](1-\rho) = \frac{E[S^2]}{2E[S]}.$$

To prove Theorem 1, we prove a stronger theorem, tightly and explicitly bounding $E[T^\pi]$ up to an additive constant.

Theorem 2 (Explicit response time bounds)**.** *For any model $\pi \in$ WCFS, if $\text{rem}_{\sup}(S, C)$ is finite,*

$$E[T^\pi] \leq \frac{\rho}{1-\rho}\frac{E[S^2]}{2E[S]} + c^\pi_{upper}$$
$$E[T^\pi] \geq \frac{\rho}{1-\rho}\frac{E[S^2]}{2E[S]} + c^\pi_{lower}$$

for explicit constants c^π_{upper} and c^π_{lower} not dependent on load ρ.

Proof deferred to Section 5.1. □

From Theorem 2, Theorem 1 follows via a simple rearrangement:

$$\frac{\rho}{1-\rho}\frac{E[S^2]}{2E[S]} = \frac{\frac{E[S^2]}{2E[S]}}{1-\rho} - \frac{E[S^2]}{2E[S]}.$$

Moreover, Theorem 2 also implies rapid convergence of scaled mean response time to its limiting constant for any WCFS policy:

Corollary 1. *For any model $\pi \in$ WCFS, if $\text{rem}_{\sup}(S, C)$ is finite,*

$$E[T^\pi](1-\rho) = \frac{E[S^2]}{2E[S]} + O(1-\rho).$$

5.1 Outline of Proof of Theorem 2

We will prove Theorem 2 where

$$c^\pi_{upper} = (n-1)\text{rem}_{\sup}(S, C) + \frac{nE[S]}{b_{\inf}},$$
$$c^\pi_{lower} = -(n-1)\text{rem}_{\sup}(S, C) + E[S].$$

where n denotes the size of the front, and where $b_{\inf}$ is defined in Section 2.1.3.

Our goal is simply to prove the bounds in Theorem 2 for some constants $c^\pi_{upper}, c^\pi_{lower}$ independent of ρ; we have made no effort to optimize these constants. We leave that to future work. However, note that for our motivating models, the $\frac{n}{b_{\inf}}$ term scales as $O(n^2)$ as the front size n grows. For these models, this term is unnecessarily loose, and could easily be lowered to an $O(n)$ bound by using a more detailed view of these models.

We start our proof of Theorem 2 in Section 5.2 by discussing two different views of a WCFS system: The *omniscient view* and the *limited view*. We will make use of these two views throughout our proof. Next, we split response time T into two pieces, queueing time T_Q and front time T_F, and bound the expectation of each separately.

We first bound $E[T_Q]$, which forms the bulk of our proof. The two key ideas behind our bounds on $E[T_Q]$ come from the intuition that a WCFS model behaves like a FCFS M/G/1 system. First, in Lemma 4, we prove that $E[T_Q] = E[W] + c$, for some constant c. The key idea is that in a WCFS model, jobs progress through the system in essentially FCFS order, and work is completed essentially at rate 1. As a result, mean queueing time $E[T_Q]$ is within an additive constant of mean work in system $E[W]$.

Second, in Lemma 3, we prove that $E[W] = E[W^{M/G/1}] + c$, for some constant c. The key idea is that in a WCFS model, if W is large, work arrives and completes in exactly the same way as in an M/G/1. In particular, in a WCFS model, if the front is not full, then W cannot be large.

In Lemma 4, we combine Lemma 4 and Lemma 3 to prove that $E[T_Q] = E[T^{M/G/1}] + c$ for some constant c.

In Lemma 5, we prove that work W is stationary and has finite mean. This is a technical lemma that rules out pathological scenarios, which is necessary because our WCFS class of models is very general. Lemma 5 is used by both Lemma 4 and Lemma 3.

Finally, in Lemma 6, we bound $E[T_F]$, using an argument based on Little's law.

In Section 5.8, we combine these lemmas to prove Theorem 2.

5.2 Two Views

At several steps in our proof of Theorem 2, we will make use of two different views of the queueing system, corresponding to two different state descriptors:

Omniscient view: In the omniscient view, we consider the state descriptor consisting of the remaining size and class of all jobs in the system. Here we sample jobs' sizes and classes when the jobs enter the system. In this view, for a given system state, work is a deterministic quantity.

Limited view: In the limited view, we consider the state descriptor consisting of the age and class of the jobs in the front, and the number of jobs in the queue. Here we sample jobs' classes when they enter the front, and determine whether jobs complete according to the hazard rate of the job size distribution, as the job ages. In this view, for a given system state, work is a random variable.

We will make it clear which view of the system we are using in each step of the proof. Generally, the omniscient view is more useful when analyzing total work in the system, while the limited view is more useful when analyzing work at the front.

5.3 Lemma 4: $E[T_Q]$ and $E[W]$

First, we prove that mean queueing time $E[T_Q]$ and mean work $E[W]$ are similar:

Lemma 2 (Queueing time and work)**.** *For any model $\pi \in$ WCFS, if* $\text{rem}_{\text{sup}}(S, C)$ *is finite,*

$$E[W] - (n-1)\text{rem}_{\text{sup}}(S, C) \leq E[T_Q] \leq E[W].$$

Proof. Let us start by writing time in queue T_Q in terms of work in the system. Let us consider the omniscient view of the system, so work W is a deterministic quantity in a given state of the system. To write T_Q in terms of W, consider an arbitrary tagged job j. When j arrives, let $W^A(j)$ be the amount of work j sees in the system. Let $W^F_F(t)$ be the amount of work j sees in the front other than j itself, when j leaves the queue and enters the front. In W^F_F, one F indicates that we are looking at the amount of work at the front, and the other F indicates that we are looking at the moment when j enters the front.

Because the model is finite-skip, jobs move from the queue to the front in arrival order, so all of the $W^A(j)$ work that was in the system when j arrived is either complete or in the front when j enters the front. Only jobs which arrive before j receive service before j enters the front. As a result, the amount of work which is completed while j is in the queue is exactly $W^A(j) - W^F_F(j)$. Note that if j enters the front upon arrival to the system, $W^A(j) = W^F_F(j)$, and no work is completed while j is in the queue.

While j is in the queue, the front must be full. Because the model is work-conserving, the system must be maximally busy during this time, completing work at rate 1. Job j is in the queue for $T_Q(j)$

time, so the system must complete $T_Q(j)$ work during that time. We can therefore conclude that

$$W^A(j) - W_F^F(j) = T_Q(j).$$

Because j is an arbitrary job, we can write $W_F^F(j)$ as W_F^F, a random variable over all jobs that pass through the system. Likewise, $T_Q(j)$ is simply T_Q. Because Poisson arrivals see time averages, $W^A(j) \sim W$, the time-stationary amount of work in the system. Combining these equivalencies, we find that

$$W - W_F^F = T_Q, \tag{6}$$

where W_F^F represents the work seen at the front when a job enters the front. Note that W is time-stationary, while W_F^F and T_Q are event-stationary.

To rigorously demonstrate (6), we need to prove that the system converges to a stationary distribution, which we prove in Lemma 5.

To give bounds on W_F^F, let us switch to the limited view of the system, where the state of the front consists of the classes and ages of the jobs at the front. We can give two simple bounds on W_F^F: First, $W_F^F \geq 0$. Next, note that $W_F^F(j)$ is the work of at most $n-1$ jobs, the jobs at the front when a given job enters the front. The expected remaining size of a job is at most $\text{rem}_{\text{sup}}(S, C)$, for any arbitrary age and class. Therefore,

$$E[W_F^F] \leq (n-1)\text{rem}_{\text{sup}}(S, C).$$

We can therefore bound $E[T_Q]$ in terms of $E[W]$:

$$E[W] - (n-1)\text{rem}_{\text{sup}}(S, C) \leq E[T_Q] \leq E[W].$$

□

5.4 Lemma 3: Bounding $E[W]$

Lemma 3. *(Work bounds) For any model $\pi \in$ WCFS, if* $\text{rem}_{\text{sup}}(S, C)$ *is finite,*

$$\frac{\rho}{1-\rho}\frac{E[S^2]}{2E[S]} \leq E[W] \leq \frac{\rho}{1-\rho}\frac{E[S^2]}{2E[S]} + (n-1)\text{rem}_{\text{sup}}(S, C).$$

Proof. To bound $E[W]$, consider the stationary random variable W^2. We do so in the omniscient view, so work is a deterministic quantity at a given time on a given sample path. W^2 evolves in two ways: continuous decrease as work is completed, and stochastic jumps as jobs arrive. Because W^2 is a stationary random variable, the expected rate of decrease and increase must be equal. Formally, we make use of the rate conservation law [34] with respect to W^2.

To calculate the expected rate of decrease, note that, ignoring moments where jobs arrive, $\frac{d}{dt}W(t) = -B(t)$, by definition. Recall that $B(t)$ is the total service rate of the system at time t. As a result, $\frac{d}{dt}W(t)^2 = -2W(t)B(t)$. This expected rate of decrease is a well-defined random variable, because the system converges to stationarity. The expected rate of decrease of W^2 is $2E[WB]$.

To calculate the expected rate of increase, let t^- be the time just before a job arrives to the system. When the job arrives, W^2 increases from $W(t^-)^2$ to $(W(t^-)+S)^2$, a change of $2W(t^-)S + S^2$. Note that $W(t^-)$ is distributed as W, by PASTA. Note also that W and S are independent, because S is sampled i.i.d.. As a result, the expected increase per arrival is $2E[W]E[S] + E[S^2]$. Arrivals occur at rate λ. As a result, the expected rate of increase is $2\lambda E[W]E[S] + \lambda E[S^2]$.

To show that these rates are equal, we must show that the rates are finite. This follows from the fact that $E[W]$ is finite, which we prove in Lemma 5.

As a result, the rates of increase and decrease of W^2 are equal:

$$\begin{aligned}
2E[WB] &= 2\lambda E[W]E[S] + \lambda E[S^2] \\
E[WB] &= \lambda E[W]E[S] + \frac{\lambda}{2}E[S^2] \\
E[WB] &= \rho E[W] + \frac{\lambda}{2}E[S^2] \\
E[W] - E[W(1-B)] &= \rho E[W] + \frac{\lambda}{2}E[S^2] \\
E[W](1-\rho) &= E[W(1-B)] + \frac{\lambda}{2}E[S^2] \\
E[W] &= \frac{E[W(1-B)]}{1-\rho} + \frac{\lambda E[S^2]}{2(1-\rho)} \qquad (7)
\end{aligned}$$

Now, we merely need to bound $E[W(1-B)]$. We do so by switching to the limited view. Note that

$$\begin{aligned} E[W(1-B)] &= E[W(1-B)\mathbb{1}\{B=1\}] + E[W(1-B)\mathbb{1}\{B<1\}] \\ &= E[W(1-B)\mathbb{1}\{B<1\}] \end{aligned}$$

Because the model is work-conserving, if $B<1$, the front is not full, and there are at most $n-1$ jobs in the system. Taking expectations over the future randomness of these jobs, we find that at any time t for which $B(t)<1$,

$$E[W(t)] \le (n-1)\text{rem}_{\text{sup}}(S,C)$$

Therefore,

$$\begin{aligned} E[W(1-B)\mathbb{1}\{B<1\}] &\le (n-1)\text{rem}_{\text{sup}}(S,C)E[(1-B)\mathbb{1}\{B<1\}] \\ &= (n-1)\text{rem}_{\text{sup}}(S,C)E[1-B] \\ &= (n-1)\text{rem}_{\text{sup}}(S,C)(1-\rho) \\ E[W(1-B)] &\le (n-1)\text{rem}_{\text{sup}}(S,C)(1-\rho) \end{aligned}$$

Substituting this into (7), our equation for $E[W]$, we find that

$$E[W] \le \frac{\lambda E[S^2]}{2(1-\rho)} + (n-1)\text{rem}_{\text{sup}}(S,C).$$

Dropping the first term of (7), we also get a lower bound:

$$E[W] \ge \frac{\lambda E[S^2]}{2(1-\rho)}.$$

□

One might alternatively try to prove Lemma 3 via a coupling argument, by coupling the WCFS system to an M/G/1 with the same arrival process. Unfortunately, this proof strategy does not succeed, for a subtle reason.

One can show that the difference in work between the two systems during an interval when the WCFS system has a full front is bounded by the amount of work in the WCFS system at the beginning of the interval. This is analogous to the many-jobs interval argument used by Grosof et al. [20] to analyze relevant work in the M/G/k/SRPT. The key difference is that in the WCFS setting, we consider total work, not relevant work, meaning that job sizes are not bounded. As a result, while the expected work at the beginning of a full-front interval is bounded, the realization of that work may be arbitrarily large.

A coupling argument would therefore need to bound the relative length of full-front intervals started by different amounts of work, to prove a time-average bound on the gap between $E[W]$ and $E[W^{M/G/1}]$. This seems intractable, given the generality of WCFS policies.

By using a Palm Calculus approach, we directly connect the small expected amount of work in a WCFS system with non-full front to a small expected difference in work between the two systems. We therefore prove Lemma 3, while avoiding all of the complications of a coupling-based argument.

5.5 Lemma 4: Bounding $E[T_Q]$

Now, we can bound $E[T_Q]$:

Lemma 4 (Queueing time bounds)**.** *For any model $\pi \in$ WCFS, if* $\text{rem}_{\text{sup}}(S,C)$ *is finite,*

$$\begin{aligned} E[T_Q^\pi] &\le \frac{\rho}{1-\rho}\frac{E[S^2]}{2E[S]} + (n-1)\text{rem}_{\text{sup}}(S,C) \\ E[T_Q^\pi] &\ge \frac{\rho}{1-\rho}\frac{E[S^2]}{2E[S]} - (n-1)\text{rem}_{\text{sup}}(S,C) \end{aligned}$$

Proof. In Lemma 2, we prove that

$$E[W] - (n-1)\text{rem}_{\text{sup}}(S,C) \le E[T_Q] \le E[W].$$

In Lemma 3, we prove that

$$\frac{\rho}{1-\rho}\frac{E[S^2]}{2E[S]} \le E[W] \le \frac{\rho}{1-\rho}\frac{E[S^2]}{2E[S]} + (n-1)\text{rem}_{\text{sup}}(S,C).$$

Combining these results, we find that

$$E[T_Q] \le \frac{\rho}{1-\rho}\frac{E[S^2]}{2E[S]} + (n-1)\text{rem}_{\text{sup}}(S,C)$$
$$E[T_Q] \ge \frac{\rho}{1-\rho}\frac{E[S^2]}{2E[S]} - (n-1)\text{rem}_{\text{sup}}(S,C).$$

□

5.6 Lemma 5: Finite $E[W]$

Lemma 5 (Finite mean work)**.** *For any model $\pi \in$ WCFS, if* $\text{rem}_{\text{sup}}(S,C)$ *is finite, for any load $\rho < 1$, W is a well-defined stationary random variable and $E[W]$ is finite.*

Proof. First, note that $W = W_F + W_Q$. Let's first focus on W_F. There are at most n jobs in the front at any time. In the limited view, each job has expected remaining size at most $\text{rem}_{\text{sup}}(S,C)$, so $E[W_F] \le n\text{rem}_{\text{sup}}(S,C)$. We can therefore focus on W_Q.

To prove that W_Q is stationary and well-defined with finite mean, we will apply the "inventory process" results of Sigman and Yao [44], and Scheller-Wolf [43]'s refinement of those results.

We upper bound W_Q by $\mathcal{W}$, which we will write as an inventory process.

$$\mathcal{W} := W\mathbb{1}\{W_Q > 0\}.$$

Here we will use the omniscient view, so $\mathcal{W}(t)$ is a specific value. By proving $\mathcal{W}$ is stationary and well-defined with finite mean, we also show the same is true of W_Q.

To see why, recall from Section 2.3 our assumption that the service policy is dependent only on the class and age of jobs at the front. The front therefore evolves in a self-contained, Markovian fashion, except for the process by which jobs move from the queue to the front. To show that W_F is stationary, it suffices to show that the indicator $\mathbb{1}\{W_Q > 0\}$ is stationary. Only the indicator of whether the queue is occupied influences the state of the front, not the specific number of jobs in the queue. As a result, the stationarity of $\mathcal{W}$ also implies the stationarity of W_F. Because $W_Q = (\mathcal{W} - W_F)^+$, the stationarity of $\mathcal{W}$ also implies the stationarity of W_Q.

To write $\mathcal{W}$ as an inventory process as in [44], we must define a process $X(t)$ with stationary and ergodic increments, such that

$$\mathcal{W}(t) = X(t) + L(t),$$

where

$$L(t) := \sup_{0 \le s \le t}(-\min\{0, X(s)\})$$

Here $X(t)$ represents the potential workload process, and $L(t)$ corrects for the fact that the queue can empty.

We will apply [43, Theorem 2.2.1], for the special case of the first moment. Note by Remarks 1 and 3, in the case of the first moment of an inventory process, it suffices to show that

- Negative drift: There exists an amount of work $w < \infty$ and a drift rate $\delta > 0$ such that conditioned on $\mathcal{W}(t) \ge w$,

$$\lim_{\epsilon \to 0} \frac{E_{\mathcal{F}_t}[X(t+\epsilon) - X(t)]}{\epsilon} \ge -\delta$$

- Finite second moment of positive jumps: There exists a constant $k_1 < \infty$ such that

$$\lim_{\epsilon \to 0} E_{\mathcal{F}_t}[((X(t+\epsilon) - X(t))^+)^2] \le k_1$$

Now, we can define the potential workload process $X(t)$ based on $W(t)$ and $W_Q(t)$.

During intervals when $W_Q(t) = 0$, $X(t)$ is constant. If t_0 is the beginning of an interval where $W_Q(t) > 0$, $X(t)$ jumps up by $W(t_0^+)$ at time t_0. During an interval where $W_Q(t) > 0$, $X(t)$ mimics $W(t)$: $X(t)$ rises by S when a job arrives, and decreases at rate 1. If t_1 is the end of an interval where $W_Q(t) > 0$, $X(t)$ jumps down by $W(t_1^-)$ at time t_1.

By construction, $X(t)$ generates $\mathcal{W}(t)$ as an inventory process. For example, let t_1 be the end of a interval where $W_Q(t) > 0$. Assume that the desired relationship between $X(t)$ and $\mathcal{W}(t)$ holds up to time t_1^-. In particular, $\mathcal{W}(t_1^-) = W(t_1^-)$. Then $\mathcal{W}(t_1^+) = 0$, as desired.

Next, we show that $X(t)$ has stationary and ergodic increments. $X(t)$ has two types of increments: First, Poisson arrivals cause increments sampled i.i.d. from S, which are clearly stationary and ergodic. Second, the beginning and end of intervals where $W_Q(t) = 0$ cause increments equal to $W_F(t)$. To see that these increments are stationary and ergodic, it suffices to show that the state of the front is stationary and ergodic. This follows from two assumptions we made in Section 2.3. First, recall that we assumed that the service policy is dependent only on the state of the front. Second, note that the front must empty and thereby undergo renewals, because the service rate $B(t)$ is at least b_{inf} whenever the system is nonempty. Thus, $X(t)$ has stationary and ergodic increments, as desired.

To demonstrate negative drift, let w be an arbitrary nonzero amount of work. Whenever $\mathcal{W}(t) \geq w$, $X(t)$ has two types of increments: jumps of size S occurring at rate λ, and continuous decrease at rate 1. As a result, the drift of $X(t)$ is $-\rho - 1 < 0$.

To demonstrate finite second moment of positive jumps, note that $X(t)$ has two kinds of positive jumps: Jumps of size S, when $W_Q(t) > 0$, and jumps of size $W(t)$, at the beginning of such an interval.

Switching back to the limited view, note that the latter kind of jump consists of the remaining size of at most n jobs. These remaining sizes are distributed as

$$R(a, c) \sim [S_c - a \mid S_c > a]$$

for some age a and class c.

It therefore suffices to show that there exists a constant r such that for all a, c,

$$E[R(a, c)^2] \leq r < \infty$$

To do so, note that we can write $R(a, c)_e$, the excess of the remaining size distribution, as a mixture of remaining size distributions for different ages. Note that for any distribution Y, the excess Y_e is equivalent to

$$Y_e \sim [Y - Y_e \mid Y > Y_e].$$

This holds because the forward and backwards renewal times are distributed identically [23, Chapter 23]. This construction can likewise be applied where $Y = R(a, c)$. By doing so, we find that

$$\begin{aligned} R(a, c)_e &\sim [R(a, c) - R(a, c)_e \mid R(a, c) > R(a, c)_e] \\ &= [S - (a + R(a, c)_e) \mid S > a + R(a, c)_e]. \end{aligned}$$

As a result, $a + R(a, c)_e$ is the desired age distribution.

For any age a', $E[R(a', c)] \leq \text{rem}_{\text{sup}}(S, C)$. Because $R(a, c)_e$ can be written as a mixture of such distributions, $E[R(a, c)_e] \leq \text{rem}_{\text{sup}}(S, C)$, which is finite by assumption.

We can now bound $E[R(a, c)^2]$:

$$E[R(a, c)^2] \leq 2E[R(a, c)]E[R(a, c)_e] \leq 2\text{rem}_{\text{sup}}(S, C)^2$$

Thus, the requirements of [43, Theorem 2.2.1] are satisfied, so both $\mathcal{W}$ and W_Q are stationary and well-defined, and have finite mean.

□

5.7 Lemma 6: Bounding $E[T_F]$

Lemma 6 (Front time bounds)**.** *For any model $\pi \in$ WCFS,*

$$E[S] \leq E[T^F] \leq \frac{nE[S]}{b_{\text{inf}}}$$

Proof. First, to prove that $E[T^F] \geq E[S]$, note that if a job receives service at the maximum possible rate of 1 for the entire time it is in the front, then the job will complete in time S. As a result, $E[T^F] \geq E[S]$.

To prove the upper bound, recall that by the positive completion rate assumption from Section 2.1.3, in all states of the front s where $N_F(s) \geq 1$, the service rate $B(s) \geq b_{\inf}$. Because $N_F(s) \leq n$, we can bound the ratio $B(s)/N_F(s)$ in all states s where $N_F(s) \geq 1$:

$$\frac{B(s)}{N_F(s)} \geq \frac{b_{\inf}}{n}.$$

Therefore, in all states,

$$B(s) \geq \frac{b_{\inf}}{n} N_F(s).$$

In expectation, the same must hold:

$$E[B] \geq \frac{b_{\inf}}{n} E[N_F]$$

Note that $E[B] = \rho$ and $E[N_F] = \lambda E[T_F]$ by Little's Law. Thus,

$$\begin{aligned} \rho &\geq \frac{b_{\inf}}{n} \lambda E[T_F] \\ \frac{nE[S]}{b_{\inf}} &\geq E[T_F] \end{aligned}$$

□

Note that Lemma 6 proves a relatively weak bound on $E[T^F]$, because we have only made the weak assumption that $b_{\inf}$ is positive. In many models, one can prove a stronger bound on $E[T^F]$ by using more information about the model's dynamics when the front is not full.

5.8 Theorem 2: Combining Everything

From Lemma 4 and Lemma 6, Theorem 2 follows immediately, with explicit formulas for c^{π}_{upper} and c^{π}_{lower}:

Theorem 2 (Explicit response time bounds)**.** *For any model $\pi \in$ WCFS, if* $\text{rem}_{\sup}(S, C)$ *is finite,*

$$\begin{aligned} E[T^{\pi}] &\leq \frac{\rho}{1-\rho} \frac{E[S^2]}{2E[S]} + c^{\pi}_{upper} \\ E[T^{\pi}] &\geq \frac{\rho}{1-\rho} \frac{E[S^2]}{2E[S]} + c^{\pi}_{lower} \end{aligned}$$

where

$$\begin{aligned} c^{\pi}_{upper} &= (n-1)\text{rem}_{\sup}(S, C) + \frac{nE[S]}{b_{\inf}}, \\ c^{\pi}_{lower} &= -(n-1)\text{rem}_{\sup}(S, C) + E[S]. \end{aligned}$$

6 Empirical Comparison: WCFS and non-WCFS

We have proven tight bounds on mean response time for all WCFS policies. To quantify the tightness of our bounds, we define the *mean response time difference* Δ^{π} for a given policy π:

$$\Delta^{\pi} = E[T^{\pi}] - \frac{\rho}{1-\rho} \frac{E[S^2]}{2E[S]} = E[T^{\pi}] - E[T_Q^{M/G/1}].$$

For instance, $\Delta^{M/G/1} = E[S]$.

We have shown in Theorem 2 that for any load ρ, $\Delta^{\pi} \in [c^{\pi}_{lower}, c^{\pi}_{upper}]$, for constants $c^{\pi}_{lower}, c^{\pi}_{upper}$ not dependent on ρ, but potentially depending on the model π. However, we have made no effort to achieve tight values for $c^{\pi}_{lower}, c^{\pi}_{upper}$.

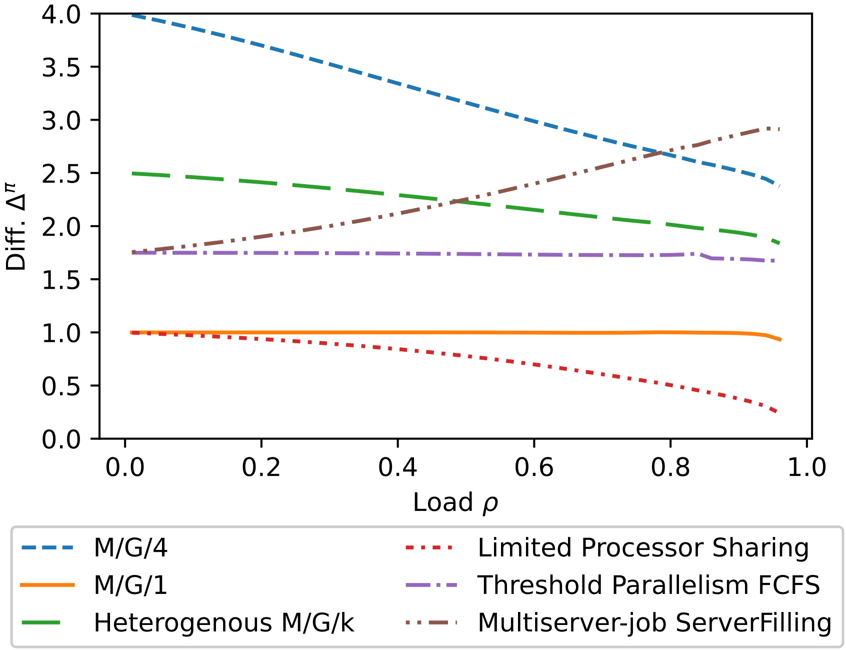


Figure 5: Δ^π for WCFS models. Job size distribution S is hyperexponential: $Exp(2)$ w.p. $1/2$, $Exp(2/3)$ otherwise. 10^9 arrivals simulated. $\rho > 0.96$ omitted due to the large amount of random noise under high load. Specific settings: Heterogeneous M/G/k with speeds $[0.4, 0.3, 0.2, 0.1]$. Limited Processor Sharing with Multiprogramming Level 4. Threshold Parallelism FCFS with joint random variable (S, L) of $(Exp(2), 1)$ w.p. $1/2$, $(Exp(2/3), 4)$ otherwise. Multiserver-job ServerFilling with joint random variable (V, X) of $(1, Exp(1/2))$ w.p. $1/2$, $(4, Exp(2/3))$ otherwise.

To investigate the behavior of Δ^π, we turn to simulation. We simulate both WCFS models, to confirm our results, as well as non-WCFS models, to show that non-WCFS models typically do not have constant Δ^π in the $\rho \to 1$ limit.

In Fig. 5, we simulate WCFS models: our four motivating models from Section 3, as well as the simpler M/G/k and M/G/1 models. In each case, we find that Δ^π does not diverge as $\rho \to 1$, and instead remains bounded quite close to 0. In particular, we find that Theorem 2 holds with constants very close to 0.

In Fig. 5, we see that for some models, Δ^π increases with ρ, while for others, Δ^π decreases with ρ. Intuitively, there are two competing behaviors: Policies which serve many jobs at once, such as the $M/G/4$ and Limited Processor Sharing systems, typically have Δ^π decrease as $\rho \to 1$, because they allow small and large jobs to share service. As a result, small jobs can complete faster than in an M/G/1, lowering Δ^π if ρ is large enough that many jobs are typically in the system.

Policies which reorder large jobs ahead of small jobs typically have Δ^π increase as $\rho \to 1$, by the same principle. For example, Multiserver-Job ServerFilling performs this reordering, because it prioritizes jobs in the front which require 4 servers. In the setting depicted in Fig. 5, such jobs have mean size $3/2$ in this system, compared to the overall mean size $E[S] = 1$.

In all of the settings simulated in Fig. 5, $\Delta^\pi > 0$. This is merely a coincidence for this particular setting, not a general rule, as can be seen in Fig. 7b.

Regardless of the different reordering behavior of these different WCFS policies, Δ^π does not diverge as $\rho \to 1$, as predicted by Theorem 2.

In contrast, in Fig. 6, we simulate several non-WCFS models, which we depicted earlier in Fig. 2. These models are:

- **Threshold Parallelism Inelastic First:** This is the Threshold Parallelism model from Section 3.3, but rather than serving jobs in FCFS order, we prioritize jobs j with smaller parallelism threshold p_j [5].
- **Threshold Parallelism Elastic First:** This is the Threshold Parallelism model from Section 3.3,

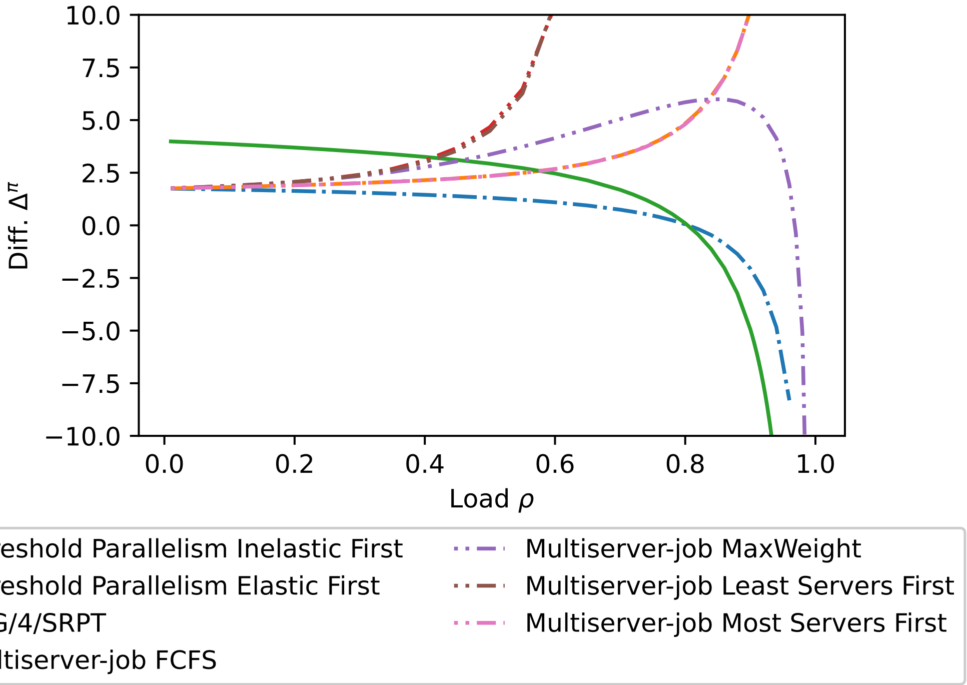


Figure 6: Δ^π for non-WCFS models. Same job sizes and specific settings as in Fig. 5. Same number of arrivals and range of ρ except MaxWeight: 10^{10} arrivals, $\rho \in [0, 0.99]$.

but we prioritize jobs j with larger parallelism threshold p_j.

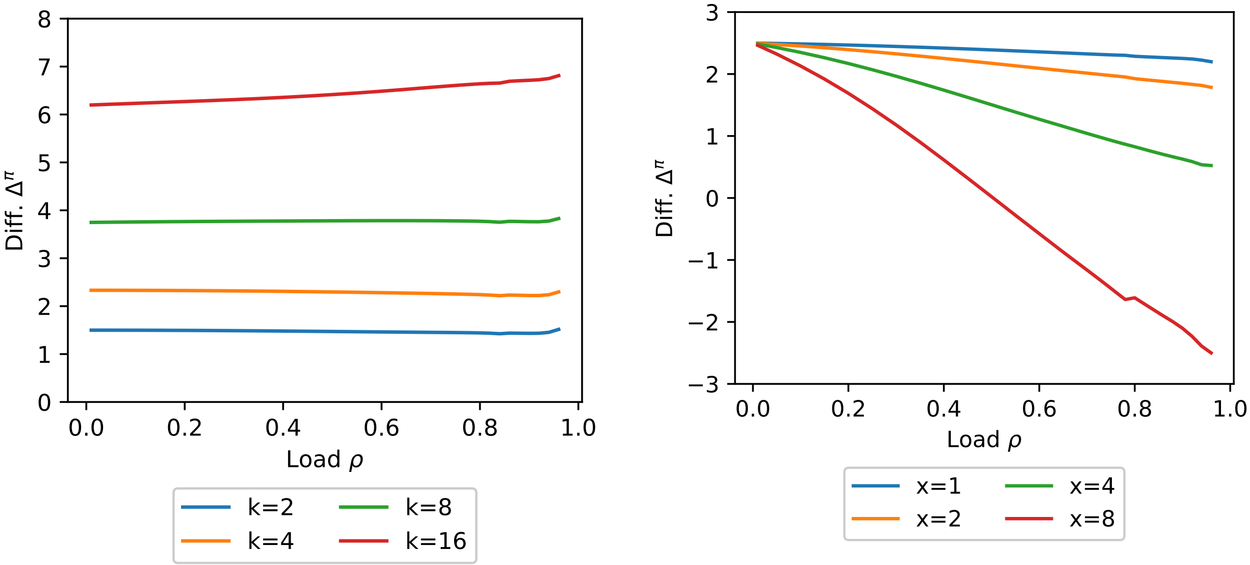
- **M/G/k/SRPT:** This is an M/G/k, where each of the k servers runs at speed $1/k$, and we prioritize jobs of least remaining size.
- **Multiserver-job FCFS:** This is the Multiserver-job model from Section 3.4, but we serve jobs in FCFS order. If the next job to be served doesn't "fit" in the remaining servers, those servers remain idle until other jobs complete, other servers become idle, and the job can now fit.
- **Multiserver-job Least Servers First:** This is the Multiserver-job model from Section 3.4, but we prioritize jobs j with smaller server requirements v_j. Again, if the next job doesn't fit, the remaining servers remain idle until the job can fit.
- **Multiserver-job Most Servers First:** This is the Multiserver-job model from Section 3.4, but we prioritize jobs j with larger server requirements v_j.
- **Multiserver-job MaxWeight:** This is the Multiserver-job model from Section 3.4, but we serve jobs according to the "MaxWeight" policy, an important service policy from the prior literature which we describe in Section 4.5.2.

In all cases, prioritization is preemptive.

Our empirical results in Fig. 6 indicate that for these non-WCFS policies, Δ^π diverges as $\rho \to 1$. Specifically, for Threshold Parallelism Elastic First, Multiserver-job FCFS, Multiserver-job Least Servers First, and Multiserver-job Most Servers First, Δ^π appears to diverge in the positive direction. For Threshold Parallelism Inelastic First, M/G/k/SRPT, and Multiserver-job ServerFilling, Δ^π appears to diverge in the negative direction. Note the expanded scale of Fig. 6 as compared to Fig. 5. For Multiserver-job MaxWeight, we performed additional simulation, which indicated that Δ^π diverged in the negative direction as $\rho \to 1$.

This demonstrates that the bounded Δ^π property observed for WCFS models in Fig. 5 and proven in Theorem 2 is highly non-trivial.

Next, we explore the behavior of Δ^π for WCFS models, as we vary the front size n and the job size distribution S.



(a) Varying front size n. Multiserver-job ServerFilling with $k = [2, 4, 8, 16]$. S distributed $Exp(1)$. Server requirement V distributed uniformly over all integer powers of $2 \leq k$.

(b) Varying job size distributions. Heterogeneous M/G/4 with speeds $[0.4, 0.3, 0.2, 0.1]$. S distributed hyperexponential: $Exp(1/x)$ with probability $1/2x$, else $Exp((2x-1)/x)$, for $x \in [1, 2, 4, 8]$. $E[S] = 1, C^2 \approxeq [1, 1.67, 3.57, 7.53]$.

Figure 7: Δ^π under WCFS models with varying conditions. Up to 10^9 arrivals simulated.

First, in Fig. 7a, we investigate the effects of varying front size n on Δ^π. Our WCFS model is the Multiserver-job model with our ServerFilling policy; under this model, the front size n is equal to the number of servers k.

In this setting, the difference Δ^π empirically grows approximately linearly with the number of servers k, and does not diverge as $\rho \to 1$. This matches the behavior of our bounds proven in Theorem 2, which expand linearly with n. Our simulations indicate that other WCFS policies similarly experience linear relationships between n and Δ^π.

Next, in Fig. 7b we investigate the effects of varying job size distribution S on Δ^π. Our WCFS model is the Heterogeneous M/G/k where the job size distribution S is parameterized by a real value x. Each S is a hyperexponential distribution with $E[S] = 1$. At large ages a, the remaining size distributions $[S - a \mid S > a]$ of these job size distributions converge to $Exp(1/x)$, the larger exponential branch. From this, it is straightforward to show that $\text{rem}_{\text{sup}}(S) = x$.

In Fig. 7b, we see that as x increases, Δ^π at loads ρ near 1 falls linearly, with more negative slope for larger x. However, for each specific x, it does not appear that Δ^π is diverging to positive or negative infinity. For instance, consider the red curve, $x = 8$. Our simulation indicates that as $\rho \to 1$, Δ^π converges to a value near -3, rather than diverging.

Broadly, Fig. 7b matches the behavior of our bounds proven in Theorem 2, which expand linearly with $\text{rem}_{\text{sup}}(S)$, which here is x. We have empirically found that other WCFS policies similarly experience linear relations between $\text{rem}_{\text{sup}}(S)$ and Δ^π, for hyperexponential job size distributions S.

7 Conclusion

We introduce *work-conserving finite-skip* (WCFS) queueing models, which includes many important queueing models which have eluded analysis thus far. We prove that the scaled mean response time $E[T^\pi](1-\rho)$ of any WCFS model π converges in heavy traffic to the same limit as M/G/1/FCFS. Moreover, we prove that the additive gap $\Delta^\pi = E[T^\pi] - E[T_Q^{M/G/1}]$ remains bounded by explicit constants at all loads ρ, proving rapid convergence to the heavy traffic limit.

A possible direction for future work would be to to tighten the explicit constants on Δ^π. Doing so will likely require use of more detailed properties of the WCFS models being analyzed.

This paper considers models which are finite skip and work conserving relative to the FCFS service ordering. Another interesting direction would be to investigate policies which are "finite-skip" relative to other service orderings. Hopefully, one could prove bounds on mean response time of models in this new class relative to an M/G/1 operating under the base service ordering.

Finally, one could try to characterize other metrics of response time for WCFS policies, such as tail metrics of response time. One approach to doing so would be to generalize the rate-conservation technique used in Lemma 3.

References

[1] Reza Aghajani and Kavita Ramanan. The limit of stationary distributions of many-server queues in the Halfin–Whitt regime. *Mathematics of Operations Research*, 45(3):1016–1055, 2020.

[2] FSQ Alves, HC Yehia, LAC Pedrosa, FRB Cruz, and Laoucine Kerbache. Upper bounds on performance measures of heterogeneous M/M/c queues. *Mathematical Problems in Engineering*, 2011, 2011.

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

[4] Benjamin Berg and Mor Harchol-Balter. Optimal scheduling of parallel jobs with unknown service requirements. In *Handbook of Research on Methodologies and Applications of Supercomputing*, pages 18–40. IGI Global, Hershey, PA, USA, 2021.

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

[6] Benjamin Berg, Mor Harchol-Balter, Benjamin Moseley, Weina Wang, and Justin Whitehouse. Optimal resource allocation for elastic and inelastic jobs. In *Proceedings of the 32nd ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '20, page 75–87, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450369350. doi: 10.1145/3350755.3400265.

[7] Onno J. Boxma, Qing Deng, and Albertus Petrus Zwart. Waiting-time asymptotics for the M/G/2 queue with heterogeneous servers. *Queueing Systems*, 40(1):5–31, 2002.

[8] Percy H. Brill and Linda Green. Queues in which customers receive simultaneous service from a random number of servers: A system point approach. *Management Science*, 30(1):51–68, 1984. doi: 10.1287/mnsc.30.1.51.

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

[10] Hyun-Duk Cho, Ph D Principal Engineer, Kisuk Chung, and Taehoon Kim. Benefits of the big.LITTLE architecture. *EETimes, Feb*, 2012.

[11] JG Dai, AB Dieker, and Xuefeng Gao. Validity of heavy-traffic steady-state approximations in many-server queues with abandonment. *Queueing Systems*, 78(1):1–29, 2014.

[12] Christina Delimitrou and Christos Kozyrakis. Quasar: Resource-efficient and QoS-aware cluster management. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '14, page 127–144, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450323055. doi: 10.1145/2541940.2541941.

[13] D. V. Efrosinin and V. V. Rykov. On performance characteristics for queueing systems with heterogeneous servers. *Automation and Remote Control*, 69(1):61–75, January 2008. ISSN 1608-3032. doi: 10.1134/S0005117908010074.

[14] Dmitry Efrosinin, Natalia Stepanova, Janos Sztrik, and Andreas Plank. Approximations in performance analysis of a controllable queueing system with heterogeneous servers. *Mathematics*, 8(10), 2020. ISSN 2227-7390. doi: 10.3390/math8101803.

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

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

[17] David Gamarnik and Petar Momčilović. Steady-state analysis of a multiserver queue in the Halfin-Whitt regime. *Advances in Applied Probability*, 40(2):548–577, 2008. doi: 10.1239/aap/1214950216.

[18] Javad Ghaderi. Randomized algorithms for scheduling VMs in the cloud. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9, April 2016. doi: 10.1109/INFOCOM.2016.7524536.

[19] David A Goldberg and Yuan Li. Simple and explicit bounds for multi-server queues with universal 1/(1-rho) scaling. *arXiv preprint arXiv:1706.04628*, 2017.

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

[21] Isaac Grosof, Mor Harchol-Balter, and Alan Scheller-Wolf. Stability for two-class multiserver-job systems. *arXiv preprint arXiv:2010.00631*, 2020.

[22] Varun Gupta and Mor Harchol-Balter. Self-adaptive admission control policies for resource-sharing systems. *SIGMETRICS Perform. Eval. Rev.*, 37(1):311–322, June 2009. ISSN 0163-5999. doi: 10.1145/2492101.1555385.

[23] Mor Harchol-Balter. *Performance modeling and design of computer systems: queueing theory in action*. Cambridge University Press, 2013.

[24] Yige Hong and Weina Wang. Sharp zero-queueing bounds for multi-server jobs. 2021.

[25] Thaga Keaogile, A. Fatai Adewole, and Sivasamy Ramasamy. Geo (λ)/Geo (μ)+ G/2 queues with heterogeneous servers operating under FCFS queue discipline. *Am. J. Appl. Math. Stat*, 3(2):54–58, 2015.

[26] JFC Kingman. Some inequalities for the queue GI/G/1. *Biometrika*, 49(3/4):315–324, 1962.

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

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

[29] Woei Lin and P. Kumar. Optimal control of a queueing system with two heterogeneous servers. *IEEE Transactions on Automatic Control*, 29(8):696–703, 1984. doi: 10.1109/TAC.1984.1103637.

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

[31] Siva Theja Maguluri and R. Srikant. Scheduling Jobs With Unknown Duration in Clouds. *IEEE/ACM Transactions on Networking*, 22(6):1938–1951, December 2014. ISSN 1558-2566. doi: 10.1109/TNET.2013.2288973. Conference Name: IEEE/ACM Transactions on Networking.

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

[33] Jason Mars, Lingjia Tang, and Robert Hundt. Heterogeneity in “homogeneous” warehouse-scale computers: A performance opportunity. *IEEE Computer Architecture Letters*, 10(2):29–32, 2011. doi: 10.1109/L-CA.2011.14.

[34] Masakiyo Miyazawa. Rate conservation laws: a survey. *Queueing Systems*, 15(1):1–58, 1994.

[35] Ripal Nathuji, Canturk Isci, and Eugene Gorbatov. Exploiting platform heterogeneity for power efficient data centers. In *Fourth International Conference on Autonomic Computing (ICAC'07)*, pages 5–5, 2007. doi: 10.1109/ICAC.2007.16.

[36] Misja Nuyens and Wemke Van Der Weij. Monotonicity in the limited processor sharing queue. *resource*, 4:7, 2008.

[37] Yanghua Peng, Yixin Bao, Yangrui Chen, Chuan Wu, and Chuanxiong Guo. Optimus: An efficient dynamic resource scheduler for deep learning clusters. In *Proceedings of the Thirteenth EuroSys Conference*, EuroSys '18, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450355841. doi: 10.1145/3190508.3190517.

[38] Konstantinos Psychas and Javad Ghaderi. On Non-Preemptive VM Scheduling in the Cloud. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(2):35:1–35:29, December 2017. doi: 10.1145/3154493.

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

[40] Sivasamy Ramasamy, Onkabetse A. Daman, and Sulaiman Sani. An M/G/2 queue where customers are served subject to a minimum violation of FCFS queue discipline. *European Journal of Operational Research*, 240(1):140–146, 2015. Publisher: Elsevier.

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

[42] Sulaiman Sani and Onkabetse A. Daman. The M/G/2 Queue with Heterogeneous Servers Under a Controlled Service Discipline: Stationary Performance Analysis. *IAENG International Journal of Applied Mathematics*, 45(1), 2015.

[43] Alan Scheller-Wolf. *Finite moment conditions for stationary content processes with applications to fluid models and queues*. PhD thesis, Columbia University, 1996.

[44] Karl Sigman and David D Yao. Finite moments for inventory processes. *The Annals of Applied Probability*, pages 765–778, 1994.

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

[46] Miklos Telek and Benny Van Houdt. Response time distribution of a class of limited processor sharing queues. *SIGMETRICS Perform. Eval. Rev.*, 45(3):143–155, March 2018. ISSN 0163-5999. doi: 10.1145/3199524.3199548.

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

[48] Aart Van Harten and Andrei Sleptchenko. On Markovian multi-class, multi-server queueing. *Queueing systems*, 43(4):307–328, 2003.

[49] SF Yashkov and AS Yashkova. Processor sharing: A survey of the mathematical theory. *Automation and Remote Control*, 68(9):1662–1731, 2007.

[50] Jiheng Zhang and Bert Zwart. Steady state approximations of limited processor sharing queues in heavy traffic. *Queueing Systems*, 60(3):227–246, 2008.

[51] Jiheng Zhang, J. G. Dai, and Bert Zwart. Law of large number limits of limited processor-sharing queues. *Mathematics of Operations Research*, 34(4):937–970, 2009. doi: 10.1287/moor.1090.0412.

[52] Jiheng Zhang, J. G. Dai, and Bert Zwart. Diffusion limits of limited processor sharing queues. *The Annals of Applied Probability*, 21(2):745 – 799, 2011. doi: 10.1214/10-AAP709.

A DivisorFilling

The DivisorFilling policy is a Multiserver-job service policy which assumes that all server requirements v_j divide the total number of servers k. The DivisorFilling policy is a WCFS policy with front size $n = k$, as we will show. Finite-skip will be straightforward, the main difficulty is showing work-conservation.

We first define the DivisorFilling policy. DivisorFilling is a preemptive policy, not maintaining any history. The DivisorFilling policy is defined recursively. The policy's behavior with respect to larger k is defined based on its behavior for smaller k. In particular, we will prove work conservation inductively.

Let F be the set of jobs at the front.

To define DivisorFilling, we split into three cases:

- F contains at least $k/6$ jobs with server requirement $v_j = 1$.
- $k = 2^a 3^b$ for some integers a, b, and F contains $< k/6$ jobs with $v_j = 1$.
- k has a prime factor $p \geq 5$ and F contains $< k/6$ jobs with $v_j = 1$.

A.1 At least $k/6$ jobs requiring 1 server

First, assume that F contain at least $k/6$ jobs requiring 1 server.

Just as in the ServerFilling policy, label the jobs $f_1, f_2, \ldots$ in decreasing order of server requirement. Let i^* be defined as

$$i^* = \arg\max_i \sum_{\ell=1}^{i} v_{f_\ell} \leq k.$$

In this case, the DivisorFilling policy serves jobs $f_1, \ldots f_{i^*}$, as well as any jobs requiring 1 server that fit in the remaining servers. Specifically, DivisorFilling serves

$$k - \sum_{\ell=1}^{i^*} v_{f_\ell}.$$

additional jobs that require 1 servers, or all jobs requiring 1 server if fewer are available.

A.1.1 Proof of work conservation

We want to show that if F contains k jobs, DivisorFilling serves jobs requiring k servers in this case.

Let us write $\textsc{sum}_{i^*} := \sum_{\ell=1}^{i^*} v_{f_\ell}$. Because we have at least $k/6$ jobs requiring 1 server, it suffices to show that $\textsc{sum}_{i^*} \geq 5k/6$. The remaining servers are filled be the jobs requiring 1 server.

First, note that $\textsc{sum}_k \geq k$, because there are k jobs, each requiring at least 1 server. Next, note that $k - \textsc{sum}_{i^*} < f_{i^*+1}$, because the $i^* + 1$ job does not fit in service. Because the labels $f_1, f_2, \ldots$ are in decreasing order of server requirement, $k - \textsc{sum}_{i^*} < f_{i^*}$.

Therefore, to prove that $k - \textsc{sum}_{i^*} \leq k/6$, we need only consider sequences of the i^* largest server requirements in F in which all such requirements at greater than $k/6$. We need only consider requirements equal to $k, k/2, k/3, k/4, k/5$.

We enumerate all such sequences. We list i^* requirements if $\textsc{sum}_{i^*} = k$, and $i^* + 1$ otherwise. We write g_{i^*} as a shorthand for $k - \textsc{sum}_{i^*}$.

Sequence	g_{i^*}	Sequence	g_{i^*}
k	0	$k/2, k/2$	0
$k/2, k/3, k/3$	$k/6$	$k/2, k/4, k/4$	0
$k/2, k/4, k/5, k/5$	$k/20$	$k/2, k/5, k/5, k/5$	$k/10$
$k/3, k/3, k/3$	0	$k/3, k/3, k/4, k/4$	$k/12$
$k/3, k/3, k/5, k/5$	$2k/15$	$k/3, k/4, k/4, k/4$	$k/6$
$k/3, k/4, k/5, k/5, k/5$	$k/60$	$k/3, k/5, k/5, k/5, k/5$	$k/15$
$k/4, k/4, k/4, k/4$	0	$k/4, k/4, k/4, k/5, k/5$	$k/20$
$k/4, k/4, k/5, k/5, k/5$	$k/10$	$k/4, k/5, k/5, k/5, k/5$	$3k/20$
$k/5, k/5, k/5, k/5, k/5$	0		

In all cases, $k - \textsc{sum}_{i^*} \geq k/6$. As a result, DivisorFilling is work conserving in this case.

A.2 $k = 2^a 3^b$

Suppose that k is of the form $2^a 3^b$, for some integers a and b, and that the number of jobs in F that require 1 server is less than $k/6$.

Let F_2 be the set of jobs requiring an even number of servers in F, and let F_r be the remaining jobs:

$$F_2 := \{j \mid j \in F, v_j \text{ is even}\}$$
$$F_r := \{j \mid j \in F, v_j \text{ is odd}, v_j > 1\}$$

Note that because 2 and 3 are the only prime factors of k, all jobs in F_r have server requirements divisible by 3.

How we now schedule is based on which is larger: $2|F_2|$, or $3|F_r|$. In this case of a tie, either would be fine, so we arbitrarily select F_2.

If $2|F_2|$ is larger, we will only serve jobs from among F_2. To do so, imagine that we combine pairs of servers, reducing k by a factor of 2, and reducing the server requirement of every job in F_2 by a factor of 2. We now compute which jobs from F_2 DivisorFilling would serve, in this simplified subproblem. DivisorFilling serves the corresponding jobs.

If $3|F_r|$ is larger, we do the same, except that we combine triples of jobs.

A.2.1 Work conservation

If at least k jobs are present, we will show that this process fills all of the servers.

Because there are $< n/6$ jobs requiring 1 server, $|F_2| + |F_3| \geq 5k/6$. As a result, either $2|F_2| \geq k$ or $3|F_r| \geq k$. Consider the case where $2|F_2| \geq k$. The constructed subproblem has $k/2$ servers and $|F_2| \geq k/2$ jobs, so by induction DivisorFilling fills all of the servers in the subproblem. That property is carried over in the main problem. The case where $3|F_r| \geq k$ is equivalent.

A.3 k has a prime factor $k \geq 5$

Finally, suppose that k has a prime factor $p \geq 5$, and that F contains $< k/6$ jobs requiring 1 server. Specifically, let p be k's largest prime factor.

Let us form the set F_p consisting of the jobs in F whose server requirements are multiples of p, and F_r consisting of jobs which require more than 1 server, but not a multiple of p. As in Appendix A.2, if $|F_p| \geq k/p$, we can recurse by combining groups of p servers to fill all of F.

Otherwise, we turn to F_r. Note that all jobs in F_r have server requirements which are divisors of k/p, because their requirements are divisors of k which are not multiples of p.

If $|F_r| \geq k/p$, let us apply the DivisorFilling policy on an arbitrary subset of F_r of size k/p. By induction, DivisorFilling finds a subset of these jobs requiring exactly k/p servers. Let us extract this subset from F_r, creating F_r^1. We repeat this process until we have extracted p subsets, or $|F_r^i| < k/p$ for some i. DivisorFilling serves the extracted subsets.

A.3.1 Work conservation

We must show that extraction procedure always successfully extracts p subsets, if $|F| = k$.

In the extraction case, note that $|F_p| < k/p \leq k/5$, and that there are $\leq k/6$ jobs requiring 1 server. F_r consists of the remaining jobs. As a result,

$$|F_r| \geq k - k/6 - k/5 = 19k/30.$$

Note also that every job in F_r requires at least 2 servers, so at most $k/2p$ jobs are extracted at each step. To prove that p subsets can be extracted, we must show that at least k/p jobs remain after $p-1$ subsets have been extracted.

$$|F_r^{p-1}| \geq \frac{19k}{30} - \frac{(p-1)k}{2p} = \frac{19k}{30} - \frac{k}{2} + \frac{k}{2p} = \frac{2k}{15} + \frac{k}{2p}$$

To prove that $|F_r^{p-1}| \geq k/p$, we just need to show that $2k/15 \geq k/2p$. But $p \geq 5$, so $2k/15 > k/10 \geq k/2p$.

Thus, we can always extract p disjoint subsets of jobs, each requiring a total of k/p servers, from F_r. Combining these subsets fills all k servers, as desired.