WCFS: A new framework for analyzing multiserver systems

Isaac Grosof[1*], Mor Harchol-Balter[1] and Alan Scheller-Wolf[2]

[1] Computer Science Department, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, 15213, PA, USA.

[2] Tepper School of Business, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, 15213, PA, USA.

*Corresponding author(s). E-mail(s): igrosof@cs.cmu.edu;
Contributing authors: harchol@cs.cmu.edu;
awolf@andrew.cmu.edu;

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. We do so by introducing the work-conserving finite-skip (WCFS) framework, which encompasses a broad class of important models. This class includes 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. We prove that for all WCFS models, scaled mean response time $\boldsymbol{E[T](1-\rho)}$ converges to the same value, $\boldsymbol{E[S^2]/(2E[S])}$, in the heavy-traffic limit, which is also the heavy traffic limit for the M/G/1/FCFS. Moreover, we prove additively tight bounds on mean response time for the WCFS class, which hold for all load $\boldsymbol{\rho}$. For each of the four models mentioned above, our bounds are the first known bounds on mean response time.

Keywords: queueing, response time, bounds, heavy traffic, multiserver, M/G/k, scheduling

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 servers run at different speeds. 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, 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 service. If fewer than k jobs are present, the server is split equally among all jobs.
- **Threshold parallelism:** A multiserver system where jobs can run on any number of servers up to some threshold, with perfect speedup. We consider FCFS service, where each job is allocated a number of servers equal to its threshold, as long as servers are available. The final job served may be allocated fewer servers than its 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, and in fact 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 convergence in heavy traffic.

The similarity of these models is illustrated by Fig. 1, which shows mean response time, $E[T]$, scaled by a factor of $1-\rho$, 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.

This similarity is striking – to see just how notable it 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.

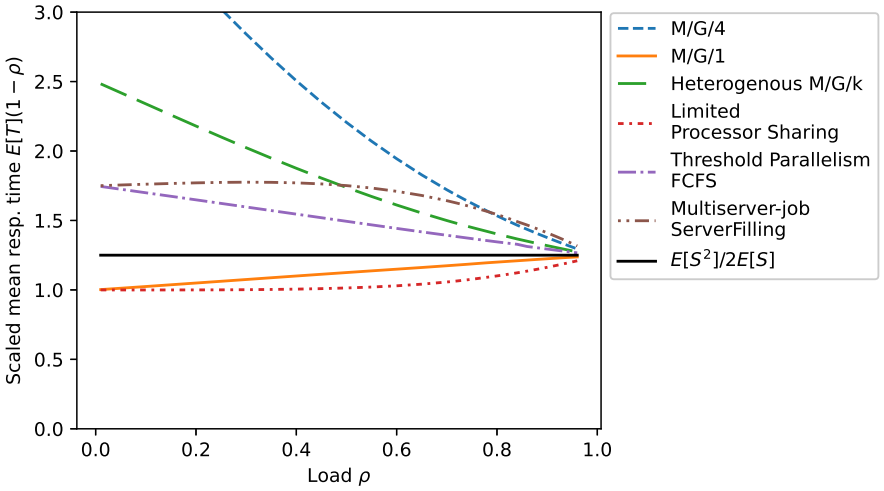


Fig. 1: Scaled mean response time of our four motivating models, as well as the related M/G/k and M/G/1 models. Our four motivating models will be further defined in Section 3. In each case, the job size distribution S is distributed as $Hyperexp(\mu_1 = 2, \mu_2 = \frac{2}{3}, p_1 = \frac{1}{2})$. The 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.

To answer this question, we define the "work-conserving finite-skip" framework (WCFS), which applies to a broad class of models. The WCFS class contains our four motivating queueing models, as well many others. We demonstrate that for any model in the WCFS class (which we call a "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]*,*

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

Theorem 1 follows from an even stronger result: 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 on the specific WCFS model.

[1]This assumption is defined in Section 2.3.

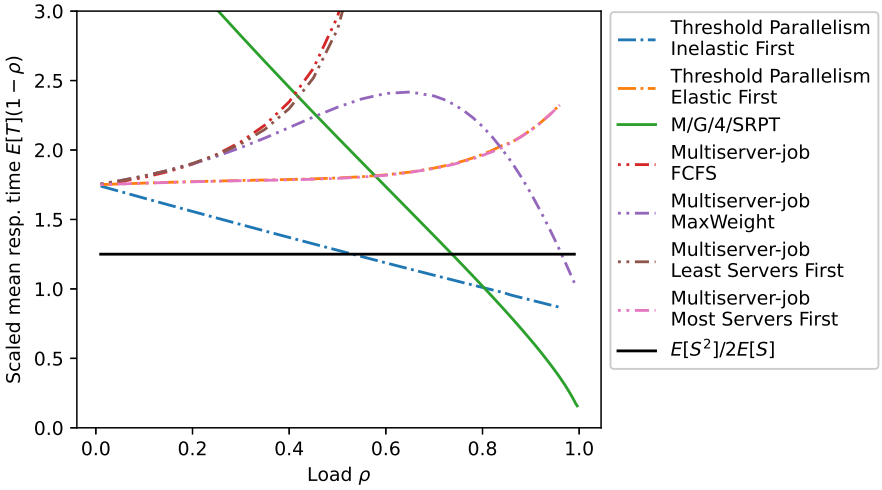


Fig. 2: Scaled mean response time of alternative models and policies. All of these models and policies will be explained in Section 6. $S \sim Hyperexp(\mu_1 = 2, \mu_2 = \frac{2}{3}, p_1 = \frac{1}{2})$. Black line is $E[T](1-\rho) = \frac{E[S^2]}{2E[S]}$. 10^9 arrivals simulated, $\rho \in [0, 0.96]$ to ensure accurate results, except MaxWeight and M/G/4/SRPT: 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 not only implies Theorem 1, it also guarantees 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 framework 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 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)

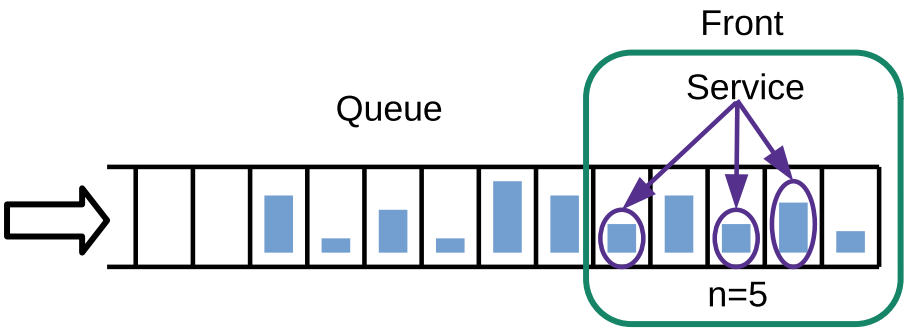


Fig. 3: Diagram of a Finite-Skip Model

2 The WCFS Framework and WCFS Models

In Sections 2.1 and 2.2, we define the WCFS framework and resulting 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.

Job sizes are sampled *i.i.d.* from a job size distribution. Once sampled, job sizes are fixed: we assume preempt-resume service if a job is preempted while in service. 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 Framework and WCFS Models

The WCFS framework applies to the class of models with Poisson arrivals at rate λ, which satisfy the following properties:

1. Finite skip (Section 2.1.1),
2. Work conserving (Section 2.1.2)
3. Non-idling (Section 2.1.3).

2.1.1 Finite skip

We first define the finite-skip property, which defines the class of 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*. The *finite-skip property* specifies that, 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.

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 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 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 $B(t) \leq 1$ denote the total service rate of all jobs in service at time t, and let B be the stationary total service rate, assuming for now such a quantity exists.

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 size and class pair. 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.

Definition 2. *We call the system* maximally busy *if the entire 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 time-average service rate, or equivalently the time-average fraction of capacity in use. Specifically, $\rho = E[B]$. We assume $\rho < 1$ to ensure stability.

2.1.3 Non-idling

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 $b_{\text{inf}} > 0$.

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 fit within the WCFS framework, we give several examples, both positive and negative.

- **M/G/k/FCFS:** This is a WCFS model with $n = k$.
- **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 because the jobs with smallest remaining size can be arbitrarily far back in the arrival ordering.
- **Multiserver-job model:** Consider a multiserver system with $k = 2$ servers, and where each job requires either 1 or 2 servers. Let the front size $n = 2$. If jobs are served in FCFS order, with head-of-the-line (HOLB) blocking, this policy is finite-skip, but not work-conserving. If the front consists of a job requiring 1 server followed by a job requiring 2 servers, under HOLB the system will only utilize one server. In this case, the front is full, because $n = 2$ jobs are present in the system, 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.
- **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 is not non-idling. To see why the non-idling property 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. To rule out systems where $E[T]$ diverges in the $\lambda \to 0$ limit, we assume the non-idling property.
- **Red and Blue M/G/k:** Imagine an M/G/k with red and blue jobs. Only one color of jobs is allowed to be in service at a time. To determine which jobs to serve, the scheduler counts off jobs in arrival order until it finds k red jobs or k blue jobs and serves all k of the appropriate color (if fewer than k jobs are found for both colors, the system serves the more populous color). This scheduling policy is WCFS with $n = 2k - 1$.

2.3 Bounded expected remaining size: Finite rem_{sup}

At a given point in time, let the *state* of a job j consist of its class c_j and its age a_j. Within our WCFS framework, we 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:

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

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

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

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:

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

More generally, an arbitrary phase type job size distribution S' must have finite $\text{rem}_{\sup}$.

On the other hand, Pareto job size distributions do not have finite $\text{rem}_{\sup}$. Let $S'' \sim Pareto(\alpha = 3, x_{\min} = 1)$, which has finite first and second moments.

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

In general, finite $\text{rem}_{\sup}$ 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 $\text{rem}_{\sup}$, while for $k \geq 1$, $\text{rem}_{\sup}$ is finite.

As a final example, suppose the WCFS scheduling policy is a known-size policy, such as a policy which serves the job with least remaining size among the n jobs in the front, at rate 1. Because we require that service is based only on the age and class of a job, we model this situation by saying that a job's class is its original size. In this case, $S = C$, and the distribution S_x is simply the constant x. As a result, $\text{rem}_{\sup}(S, C) = \sup(S)$. Therefore, in a known-size setting, $\text{rem}_{\sup}$ is finite only if S is bounded.

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. 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/FCFS models multiserver systems where servers have different speeds. This scenario commonly arises in datacenters, which are often composed of servers with a wide variety of different types of hardware [1, 2]. In the mobile device setting, the big.LITTLE architecture employs heterogeneous processors to improve battery life [3].

Let each server i have speed $v_i > 0$, scaled 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.

If there are multiple servers idle when a job arrives, a server is chosen according to an arbitrary server assignment policy. Jobs may also be migrated between servers when a job completes. 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 fit within the WCFS framework.

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 $b_{\text{inf}} \geq v_{\min}$, 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 [4]. However, in real systems, running too many jobs at once causes a significant overhead. A natural remedy is to utilize a policy is known as Limited Processor Sharing (LPS) [5–8].

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 $1/k$. LPS is a WCFS model with $n = k$.

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

3.2.1 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 $b_{\text{inf}} \geq v_{\min}$, where $v_{\min} = \min_i v_i$.

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 [9, 10]. Under Threshold Parallelism, 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 receives service rate $\frac{q}{k}$ and will complete after

$\frac{ks_j}{q}$ time in service. The number of servers a job is allocated 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, or all jobs are in service. Each job j other than the final job in service is served by ℓ_j servers. The final job in service is served by the remaining servers. Under FCFS service, Threshold Parallelism fits the WCFS framework with $n = k$.

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 policies do not fit within the WCFS framework, because a job may skip over an arbitrary number of jobs.

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.

3.4.1 Multiserver-Job Setting

When we look at jobs in cloud computing systems [11] and in supercomputing systems [12–14], jobs commonly require 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 [15, 16]. The distribution is highly variable, with jobs requesting anywhere from 1 to 100,000 normalized CPUs[2].

[2]The data was published in a scaled form [15]. We rescale the data so the smallest job in the trace uses one normalized CPU.

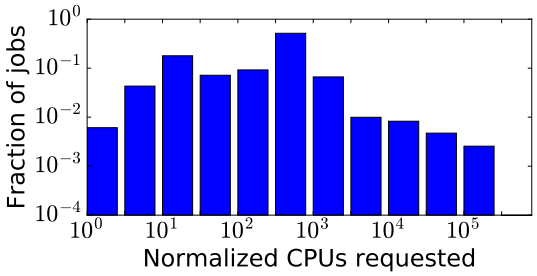


Fig. 4: The distribution of number of CPUs requested in Google's recently published Borg trace [15]. Number of CPUs is normalized to the size of the smallest request observed, not an absolute value.

The Multiserver-Job (MSJ) model is a natural model for these computing systems. In an MSJ model, 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 (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

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

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 formally 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 $\frac{1}{2}$.

Unfortunately, no existing policies fit within the WCFS framework – 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. The scaled mean response time of 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.

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.

Next, the jobs in M are ordered by their server requirements v_j, from largest to smallest, tiebroken by arrival order. Jobs are placed into service in that order until no more servers are available. In our example, jobs requiring $4, 2, 1$, and 1 server(s) would be placed into service.

To show that ServerFilling fits within WCFS with $n = k$, we must show that ServerFilling always utilizes all k servers if at least k jobs are in the system.

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 $\text{REQ}(z)$ count the number of servers required by the first z jobs in this ordering:

$$\text{REQ}(z) = \sum_{j=1}^{z} v_{m_j}.$$

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

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

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

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

$$k - \text{REQ}(z) \text{ is divisible by } v_{m_{z+1}} \text{ for all } z. \quad (2)$$

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

We will prove (2) inductively. For $z = 0$, $k - \text{REQ}(0) = k$. 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 $v_{m_{z+1}} | (k - \text{REQ}(z))$. Note that $\text{REQ}(z+1) = \text{REQ}(z) + v_{m_{z+1}}$. As a result, $v_{m_{z+1}} | (k - \text{REQ}(z+1))$. 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, $v_{m_{z+2}} | (k - \text{REQ}(z+1))$, completing the proof of (2).

Now, we are ready to prove (1). Assume for contradiction that there does exist such an ℓ'. Then $k - \text{REQ}(\ell') > 0$, and $k - \text{REQ}(\ell' + 1) < 0$. Because $\text{REQ}(\ell' + 1) =$

$\text{REQ}(\ell') + v_{m_{\ell'+1}}$, we therefore know that $v_{m_{\ell'+1}} > k - \text{REQ}(\ell')$. But from (2), we know that $v_{m_{\ell'+1}}$ divides $k - \text{REQ}(\ell')$, which is a contradiction. □

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 $b_{\text{inf}} \geq 1/k$.

3.4.4 Generalizations of ServerFilling

The ServerFilling policy can be generalized, as long as all server requirements divide k. We describe the corresponding scheduling policy, which we call DivisorFilling, in Section A.

DivisorFilling is the most general possible WCFS policy for the MSJ setting. If some server requirement does not divide k, then no policy fits within the WCFS framework, because the system is not work conserving if all jobs present require that non-divisible number of servers and more than n jobs are present.

4 Prior Work

4.1 M/G/k

4.1.1 Fixed k

In this regime, the best known bounds on response time either require much stronger assumptions on the job size distribution S than we assume [17], or prove much weaker bounds on mean response time [18, 19].

A paper by Loulou [17] 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. 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.

These bounds are comparable to those in our Lemma 3 when S is bounded, but our bounds require a much weaker assumption on the job size distribution S.

Köllerström [18] 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 [19] improves upon [18] by characterizing the rate of convergence and thereby derives explicit moment bounds. However, unlike prior single-server results [20], these bounds are quite weak. Specialized to the M/G/k, Köllerström [19]'s bounds 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 $\frac{E[S^2]}{2E[S]}$.

Our bounds are tighter in that they are constants not depending on ρ, but we assume S has finite rem_{sup}, while Köllerström [19] merely assumes that S has finite second moment.

4.1.2 Scaling k

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; they indicate that the limiting behavior in the Halfin-Whitt regime depends in a complex way on the job size distribution S [21–23].

Turning to the more general case of scaling k, in work currently under submission, Goldberg and Li [24] prove the first bounds on $E[T_Q]$ that scale as $\frac{c}{1-\rho}$ for an explicit constant c and arbitrary joint scaling of k and ρ. Unfortunately, the constant c is enormous, scaling as $10^{450}E[S^3]$. In contrast, we focus

on the regime of fixed k, and prove tight and explicit bounds on mean response time. Goldberg and Li [24] 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 [25–28]. 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 [26]. 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/$\mathbf{H_m}$/k

Van Harten and Sleptchenko [29] 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. [30]. 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. [30] derive 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, under both FCFS service and related service disciplines [31–33].

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 in solving even the simplest heterogeneous multiserver models. In contrast, our WCFS framework 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 [5–8, 34–36], but none bound mean response time for all loads ρ.

4.3.1 Asymptotic Regimes

A series of papers by Zhang, Dai and Zwart [7, 34, 35] derive the strongest known results on Limited Processor Sharing in a variety of asymptotic regimes. These authors derive the measure-valued fluid limit [34], the diffusion limit [35] and a steady-state approximation [7]. The most comparable of their results to our work is their steady-state approximation. When specialized to mean response time in the M/G/1/LPS, their approximation states that

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

They prove that this approximation is accurate in the heavy-traffic limit; they do not provide specific error bounds, but empirically show the approximation performs well at all loads ρ [7]. Our results therefore complement their results by proving concrete error bounds.

4.3.2 State-dependent Server Speed

To model the behavior of databases, Gupta and Harchol-Balter [8] introduce a variant of the Limited Processor Sharing model, where the total server speed is a function of the number of jobs in service. In their setting 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). While this two-moment approximation is not known to be tight, 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, because the system only partially fulfills our definition of work conservation.

Subsequently, Telek and Van Houdt [6] 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, making it difficult to derive general insights across general job size distributions. Instead, the authors numerically invert the Laplace transform for a handful of specific distributions to derive empirical insights.

4.4 Threshold Parallelism

Jobs with "speedup functions" are common in Machine Learning and other highly parallel computing settings. A job's speedup function specifies the degree to which it can be parallelized. In [37–39], the authors study optimal allocation policies of servers to jobs when the arriving jobs have different speedup function. In many cases, a job's speedup function takes the form of a "threshold" function: here the job receives perfect (linear) speedup up to some threshold number of servers and receives no additional speedup beyond that number of servers. We refer to this as the Threshold Parallelism model.

While understanding the response time in systems where jobs have speedup functions is generally intractable, Berg et al. [39] were able to approximately analyze response time in the case where every job is 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). In this setting, they approximate the mean response time of EF and IF within 1% error by using a combination of the Busy-Period Transitions technique and Matrix-Analytic methods to evaluate their multidimensional Markov chain.

The Threshold Parallelism model in our paper is far broader than that in the prior literature, and our bounds are tighter in the heavy-traffic limit.

4.5 Multiserver Jobs

The Multiserver-Job model has been extensively studied, in both practical [12–14] and theoretical settings [11, 40–46]. 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 scheduling policy, prior to our bound on ServerFilling.

4.5.1 FCFS Scheduling

The most natural policy is FCFS, where the oldest jobs are placed into service until a job requires more servers than remain, at which point the queue is blocked. Therefore, the FCFS policy can leave a large number of servers idle even when many jobs are present. As a result, FCFS does not in general achieve an optimal stability region. Even worse, deriving the stability region of FCFS is an open problem, and has only been solved in a few special cases [40, 41].

One technique that may be useful for characterizing this stability region is the *saturated system* approach [47, 48]. 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, analyzing the general Multiserver-Job FCFS saturated system seems intractable.

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

4.5.2 MaxWeight Scheduling

One natural throughput-optimal policy is the MaxWeight policy [11]. Here jobs are divided into classes based on their server requirements, with $N_i(t)$ denoting the number of jobs requiring i servers in the system at time t. Let the set $Z(t)$ denote all possible packings of jobs at time t onto servers. Let $z \in Z(t)$ be a particular packing, where z_i denotes the number of jobs requiring i servers that are served by packing z.

The MaxWeight service policy picks the packing z which maximizes

$$\max_z \sum_i N_i(t) z_i.$$

For example, if there are many jobs requiring 3 servers, we want to pick a packing that serves many 3-server jobs. While MaxWeight is throughput optimal, 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 our 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 for 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 [12–14]. These start by serving jobs in FCFS order, until a job is reached that requires more servers than remain. At this point, jobs further back in the queue that require fewer servers are scheduled, but only if they 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 a challenging problem. Several such policies have been designed [43, 44, 46], 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, which 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, Our bounds

on the mean response time of ServerFilling can serve as such a baseline, albeit in the more permissive setting of preemptive scheduling.

5 Theorems and Proofs

We perform a heavy traffic analysis within our WCFS framework, assuming finite $\text{rem}_{\text{sup}}(S, C)$. Specifically, we 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 $\text{rem}_{\text{sup}}(S, C)$ 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, for any $\pi \in$ WCFS.

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

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

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 $\text{rem}_{\text{sup}}(S, C)$ 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, leaving that to future work. Specifically, for three of our four motivating models, the $\frac{n}{b_{\inf}}$ term scales as $O(n^2)$. For these models this term is unnecessarily loose, and could easily be lowered to an $O(n)$ bound by using a more detailed view.

Our approach is to 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 come from the intuition that a WCFS model behaves like a FCFS M/G/1 system. In Lemma 2, we prove that $E[T_Q] = E[W] + c$, for some constant c; in a WCFS model, jobs progress through the system in essentially FCFS order, and as $\rho \to 1$ work is completed essentially at rate 1.

In Lemma 3, we prove that $E[W] = E[W^{M/G/1}] + c$, for some constant c. The key idea here 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. Likewise, if the front is not full, then W cannot be large.

In Lemma 4, we combine Lemmas 2 and 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 indeed stationary with 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 Lemmas 2 and 3.

Finally, in Lemma 6, we bound $E[T_F]$, utilizing Little's law.

Combining Lemmas 4 and 6 proves 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 the state descriptor consists of the remaining size and class of all jobs in the system; we sample jobs' sizes and classes when the jobs enter the system. For a given system state, work is a deterministic quantity.

Limited view: In the limited view, the state descriptor consists of the age and class of the jobs in the front, and the number of jobs in the queue. 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. 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 useful when analyzing total work in the system, and the limited view is useful when analyzing work at the front.

5.3 Lemma 2: $E[T_Q]$ and $E[W]$

First, we prove that mean queueing time and mean work 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 Start by writing time in queue T_Q in terms of work in system. Let us consider the omniscient view of the system, so work W is a deterministic quantity given the system state. Consider an arbitrary tagged job j. When j arrives, let $W^A(j)$ be the amount of work j sees in the system. Let $W_F^F(j)$ 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, the subscript F indicates that we are looking at the amount of work at the front, and the superscript 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. 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; 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

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

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, we 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 have two simple bounds on W_F^F: First, $W_F^F \geq 0$. Next, because $W_F^F(j)$ is the work of at most $n-1$ jobs, the jobs at the front when a given job enters the front, we know that

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

Combining these bounds with (6), we can bound $E[T_Q]$ in terms of $E[W]$:

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

□

5.4 Lemma 3: Bounding $E[W]$

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

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

Proof Consider the stationary random variable W^2 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, due to the rate conservation law [49] 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, where $B(t)$ is the total service rate of the system at time t. As a result, $\frac{d}{dt}W(t)^2 = -2W(t)B(t)$, ignoring arrival epochs. This expected rate of decrease is a well-defined random variable, because the system converges to stationarity. Thus 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, at any time t for which $B(t) < 1$,

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

Therefore,

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

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

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

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

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

□

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. [50] 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.

Instead, by using a rate-conservation approach, formalized by Palm Calculus, 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 $E[T_Q]$ by combining Lemmas 2 and 3:

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

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

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 Recall that $W = W_F + W_Q$; we 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 $E[W_F] \leq n\text{rem}_{\text{sup}}(S, C)$.

As for the stationarity of the state of the front, this follows from two assumptions we made in Section 2.3. First, we assumed that the service policy is dependent only on the state of the front. Second, the front must empty and thereby undergo renewals, because the service rate $B(t)$ is at least b_{inf} whenever the system is nonempty. As a result, W_F is stationary.

We now turn to 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 [51], and Scheller-Wolf [52]'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. Because $W_Q = (\mathcal{W} - W_F)^+$, the stationarity of $\mathcal{W}$ also implies the stationarity of W_Q, given the stationarity of W_F.

To write $\mathcal{W}$ as an inventory process as in [51], we must define a process $X(t)$ with stationary and ergodic increments, such that

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

where

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

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

We will apply [52, Theorem 2.2.1], for the special case of the first moment. Note by Remarks 1 and 3, for the first moment of an inventory process, it suffices to show:

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

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

where $\mathcal{F}_t$ is the filtration defined by the behavior of the system up to time t.

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

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

Now, we 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 an 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 $W_F(t)$. These increments are stationary and ergodic because the state of the front, and W_F in particular, are stationary. Thus, $X(t)$ has stationary and ergodic increments.

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 a $W_Q > 0$ 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, we will 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 [36, Chapter 23]. By applying this construction with $Y = R(a, c)$, we find that

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

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

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

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

$$E[R(a, c)_e] = \frac{E[R(a, c)^2)]}{2E[R(a, c)]}$$

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

Thus, the requirements of [52, 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 non-idling assumption from Section 2.1.3, in all states of the front s where $N_F(s) \geq 1$, the service rate $B(s) \geq b_{\text{inf}}$. Because $N_F(s) \leq n$, we can bound the ratio $B(s)/N_F(s)$ in all $N_F(s) \geq 1$ states:

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

Therefore, in all states,

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

In expectation, the same must hold:

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

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

$$\rho \geq \frac{b_{\text{inf}}}{n} \lambda E[T_F]$$

$$\frac{nE[S]}{b_{\text{inf}}} \geq E[T_F].$$

□

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.

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

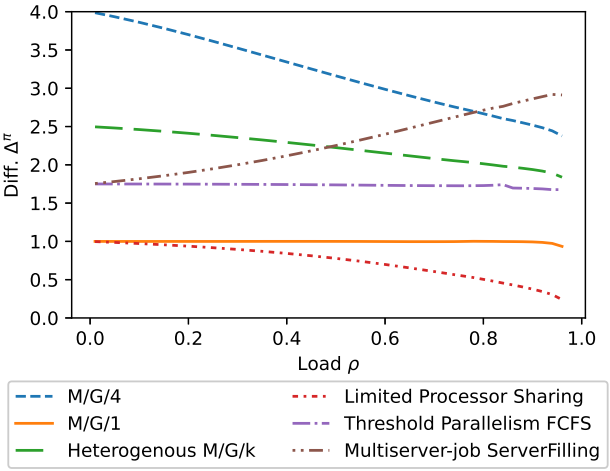


Fig. 5: Δ^π for WCFS models. Job size distribution S is hyperexponential: $Exp(2)$ w.p. $1/2$, $Exp(2/3)$ otherwise. 10^9 arrivals simulated. $\rho > 0.96$ omitted due to the large amount of random noise under high load. Specific settings: Heterogeneous M/G/k with speeds $[0.4, 0.3, 0.2, 0.1]$. Limited Processor Sharing with Multi-programming 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.

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

This definition is useful because 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.

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

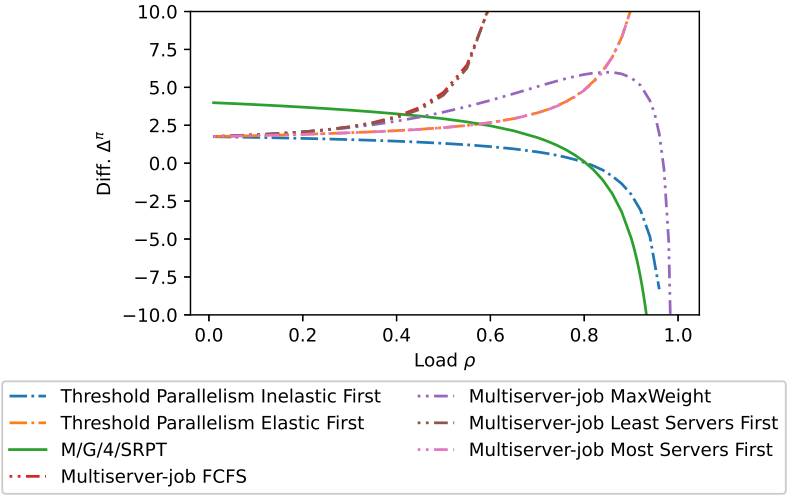


Fig. 6: Δ^π for non-WCFS models. Same job sizes and specific settings as in Fig. 5. Same number of arrivals and range of ρ except MaxWeight: 10^{10} arrivals, $\rho \in [0, 0.99]$.

find that Δ^π remains bounded quite close to 0, meaning that Theorem 2 holds with constants 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, this depends on which jobs tend to be prioritized as $\rho \to 1$. 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.

In contrast, 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 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, 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 [37].

- **Threshold Parallelism Elastic First:** This is the Threshold Parallelism model from Section 3.3, 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, idling sufficient servers to allow the job to 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 which we describe in Section 4.5.2.

In all cases, prioritization is preemptive.

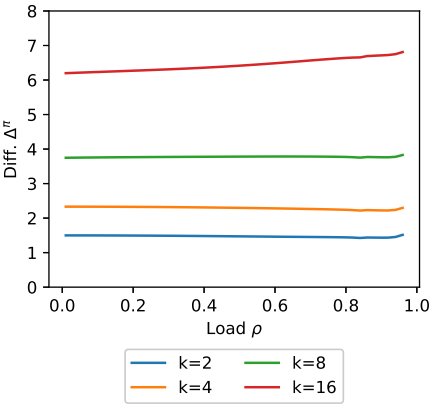
Our empirical results in Fig. 6 indicate that for these non-WCFS policies, Δ^π diverges as $\rho \to 1$. Specifically, for Threshold Parallelism Elastic First, Multiserver-job FCFS, Multiserver-job Least Servers First, and Multiserver-job Most Servers First, Δ^π appears to diverge in the positive direction. 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 $\rho \to 1$.

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

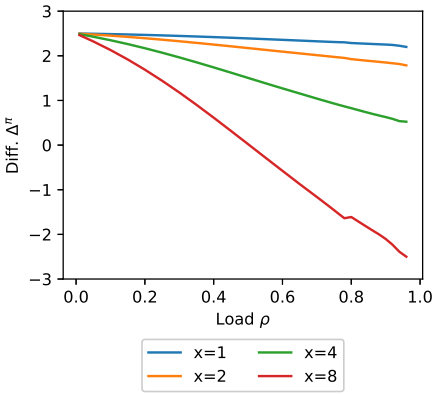
First, in Fig. 7a, we investigate the effects of varying front size n on Δ^π for the Multiserver-job model with our ServerFilling policy; under this model, the front size n is equal to the number of servers k. In this setting, the difference Δ^π empirically grows approximately linearly with the number of servers k, and is nearly constant 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.

In Fig. 7b we investigate the effects of varying job size distribution S on Δ^π in 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 $\text{rem}_{\sup}(S) = x$.

In Fig. 7b, we see that as x increases, Δ^π at loads near 1 falls linearly, with more negative slope for larger x. However, for each specific x, it does



(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 $x \in [1, 2, 4, 8]$. $E[S] = 1, C^2 \approx [1, 1.67, 3.57, 7.53]$.

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

not appear that Δ^π is diverging to positive or negative infinity. For instance, consider the red curve, $x = 8$: 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, and we believe that similar behavior will occur for other common job size distributions.

7 Conclusion

We introduce the *work-conserving finite-skip* (WCFS) framework, and use it to analyze 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, but seems quite doable.

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 base 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. One possible approach to doing so would be to generalize the rate-conservation technique used in Lemma 3.

References

[1] Nathuji, R., Isci, C., Gorbatov, E.: Exploiting platform heterogeneity for power efficient data centers. In: Fourth International Conference on Autonomic Computing (ICAC'07), pp. 5–5 (2007)

[2] Mars, J., Tang, L., Hundt, R.: Heterogeneity in "homogeneous" warehouse-scale computers: A performance opportunity. IEEE Computer Architecture Letters **10**(2), 29–32 (2011)

[3] Cho, H.-D., Engineer, P.D.P., Chung, K., Kim, T.: Benefits of the big.LITTLE architecture. EETimes, Feb (2012)

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

[5] Nuyens, M., Van Der Weij, W.: Monotonicity in the limited processor sharing queue. resource **4**, 7 (2008)

[6] Telek, M., Van Houdt, B.: Response time distribution of a class of limited processor sharing queues. SIGMETRICS Perform. Eval. Rev. **45**(3), 143–155 (2018)

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

[8] Gupta, V., Harchol-Balter, M.: Self-adaptive admission control policies for resource-sharing systems. SIGMETRICS Perform. Eval. Rev. **37**(1), 311–322 (2009)

[9] Delimitrou, C., Kozyrakis, C.: 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, pp. 127–144 (2014)

[10] Peng, Y., Bao, Y., Chen, Y., Wu, C., Guo, C.: Optimus: An efficient dynamic resource scheduler for deep learning clusters. In: Proceedings of

the Thirteenth EuroSys Conference. EuroSys '18 (2018)

[11] Maguluri, S.T., Srikant, R., Ying, L.: Stochastic models of load balancing and scheduling in cloud computing clusters. In: 2012 Proceedings IEEE Infocom, pp. 702–710. IEEE, Orlando, FL, USA (2012)

[12] Feitelson, D.G., Rudolph, L., Schwiegelshohn, U.: Parallel job scheduling—a status report. In: Workshop on Job Scheduling Strategies for Parallel Processing, pp. 1–16. Springer, New York, NY, USA (2004)

[13] Srinivasan, S., Kettimuthu, R., Subramani, V., Sadayappan, P.: Characterization of backfilling strategies for parallel job scheduling. In: Proceedings. International Conference on Parallel Processing Workshop, pp. 514–519 (2002)

[14] Carastan-Santos, D., De Camargo, R.Y., Trystram, D., Zrigui, S.: 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), pp. 1–10 (2019)

[15] Tirmazi, M., Barker, A., Deng, N., Haque, M.E., Qin, Z.G., Hand, S., Harchol-Balter, M., Wilkes, J.: Borg: The next generation. In: Proceedings of the Fifteenth European Conference on Computer Systems. EuroSys '20 (2020)

[16] Grosof, I., Harchol-Balter, M., Scheller-Wolf, A.: Stability for two-class multiserver-job systems. arXiv preprint arXiv:2010.00631 (2020)

[17] Loulou, R.: Multi-channel queues in heavy traffic. Journal of Applied Probability **10**(4), 769–777 (1973)

[18] Köllerström, J.: Heavy traffic theory for queues with several servers. I. Journal of Applied Probability **11**(3), 544–552 (1974)

[19] Köllerström, J.: Heavy traffic theory for queues with several servers. II. Journal of Applied Probability **16**(2), 393–401 (1979)

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

[21] Gamarnik, D., Momčilović, P.: Steady-state analysis of a multiserver queue in the Halfin-Whitt regime. Advances in Applied Probability **40**(2), 548–577 (2008)

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

[23] Dai, J., Dieker, A., Gao, X.: Validity of heavy-traffic steady-state approximations in many-server queues with abandonment. Queueing Systems **78**(1), 1–29 (2014)

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

[25] Efrosinin, D.V., Rykov, V.V.: On performance characteristics for queueing systems with heterogeneous servers. Automation and Remote Control **69**(1), 61–75 (2008)

[26] Alves, F., Yehia, H., Pedrosa, L., Cruz, F., Kerbache, L.: Upper bounds on performance measures of heterogeneous M/M/c queues. Mathematical Problems in Engineering **2011** (2011)

[27] Efrosinin, D., Stepanova, N., Sztrik, J., Plank, A.: Approximations in performance analysis of a controllable queueing system with heterogeneous servers. Mathematics **8**(10) (2020)

[28] Lin, W., Kumar, P.: Optimal control of a queueing system with two heterogeneous servers. IEEE Transactions on Automatic Control **29**(8), 696–703 (1984)

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

[30] Boxma, O.J., Deng, Q., Zwart, A.P.: Waiting-time asymptotics for the M/G/2 queue with heterogeneous servers. Queueing Systems **40**(1), 5–31 (2002)

[31] Keaogile, T., Fatai Adewole, A., Ramasamy, S.: 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)

[32] Sani, S., Daman, O.A.: 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)

[33] Ramasamy, S., Daman, O.A., Sani, S.: 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

[34] Zhang, J., Dai, J.G., Zwart, B.: Law of large number limits of limited processor-sharing queues. Mathematics of Operations Research **34**(4), 937–970 (2009)

[35] Zhang, J., Dai, J.G., Zwart, B.: Diffusion limits of limited processor sharing queues. The Annals of Applied Probability **21**(2), 745–799 (2011)

[36] Harchol-Balter, M.: Performance Modeling and Design of Computer Systems: Queueing Theory in Action. Cambridge University Press, Cambridge, England (2013)

[37] Berg, B., Dorsman, J.-P., Harchol-Balter, M.: Towards optimality in parallel scheduling. Proc. ACM Meas. Anal. Comput. Syst. **1**(2) (2017)

[38] Berg, B., Harchol-Balter, M.: Optimal scheduling of parallel jobs with unknown service requirements. In: Handbook of Research on Methodologies and Applications of Supercomputing, pp. 18–40. IGI Global, Hershey, PA, USA (2021)

[39] Berg, B., Harchol-Balter, M., Moseley, B., Wang, W., Whitehouse, J.: Optimal resource allocation for elastic and inelastic jobs. In: Proceedings of the 32nd ACM Symposium on Parallelism in Algorithms and Architectures. SPAA '20, pp. 75–87 (2020)

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

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

[42] Hong, Y., Wang, W.: Sharp zero-queueing bounds for multi-server jobs (2021)

[43] Ghaderi, J.: Randomized algorithms for scheduling VMs in the cloud. In: IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications, pp. 1–9 (2016)

[44] Psychas, K., Ghaderi, J.: Randomized algorithms for scheduling multi-resource jobs in the cloud. IEEE/ACM Transactions on Networking **26**(5), 2202–2215 (2018)

[45] Psychas, K., Ghaderi, J.: On Non-Preemptive VM Scheduling in the Cloud. Proceedings of the ACM on Measurement and Analysis of Computing Systems **1**(2), 35–13529 (2017)

[46] Maguluri, S.T., Srikant, R.: Scheduling Jobs With Unknown Duration in Clouds. IEEE/ACM Transactions on Networking **22**(6), 1938–1951 (2014). Conference Name: IEEE/ACM Transactions on Networking

[47] Baccelli, F., Foss, S.: On the saturation rule for the stability of queues. Journal of Applied Probability **32**(2), 494–507 (1995)

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

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

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

[51] Sigman, K., Yao, D.D.: Finite moments for inventory processes. The Annals of Applied Probability, 765–778 (1994)

[52] Scheller-Wolf, A.: Finite moment conditions for stationary content processes with applications to fluid models and queues. PhD thesis, Columbia University (1996)

Appendix 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, in that when a job completes, the set of jobs in service may change, removing partially-complete jobs from service. 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 M be the set of jobs at the front.

To define DivisorFilling, we split into three cases:

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

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

First, assume that M 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 $f_1, \dots f_{i^*}$, 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 Work conservation

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

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

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

Therefore, to prove that $k - \text{SUM}_{i^*} \leq k/6$, we need only consider sequences of the i^* largest server requirements in M in which all such requirements are 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. Note that if k is not divisible by all of $\{2, 3, 4, 5\}$, some entries will not apply. This only tightens the resulting bound on $k - \text{SUM}_{i^*}$ for such k.

We list i^* requirements if $\text{SUM}_{i^*} = k$, and $i^* + 1$ otherwise. We write g_{i^*} as a shorthand for $k - \text{SUM}_{i^*}$.

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, $k - \text{SUM}_{i^*} \leq k/6$. 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 M that require 1 server is less than $k/6$.

Let M_2 be the set of jobs requiring an even number of servers in M, and let M_r be the remaining jobs:

$$M_2 := \{j \mid j \in M, v_j \text{ is even}\}$$
$$M_r := \{j \mid j \in M, v_j \text{ is odd}, v_j > 1\}$$

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

How we now schedule is based on which is larger: $2|M_2|$, or $3|M_r|$. In this case of a tie, either would be fine, so we arbitrarily select M_2.

If $2|M_2|$ is larger, we will only serve jobs from among M_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 M_2 by a factor of 2. We now compute which jobs from M_2 DivisorFilling would serve, in this simplified subproblem. DivisorFilling serves the corresponding jobs.

If $3|M_r|$ 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, $|M_2| + |M_3| \geq 5k/6$. As a result, either $2|M_2| \geq k$ or $3|M_r| \geq k$. Consider the case where $2|M_2| \geq k$. The constructed subproblem has $k/2$ servers and $|M_2| \geq k/2$ 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 $3|M_r| \geq k$ 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 M contains $< k/6$ jobs requiring 1 server. Specifically, let p be k's largest prime factor.

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

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

If $|M_r| \geq k/p$, let us apply the DivisorFilling policy on an arbitrary subset of M_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 M_r, creating M_r^1. We repeat this process until we have extracted p subsets, or $|M_r^i| < k/p$ for some i. DivisorFilling serves the extracted subsets.

A.3.1 Work conservation

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

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

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

Note also that every job in M_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.

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

To prove that $|M_r^{p-1}| \geq k/p$, 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 M_r. Combining these subsets fills all k servers, as desired.