Optimal Scheduling in the Multiserver-job Model under Heavy Traffic

ISAAC GROSOF, ZIV SCULLY, MOR HARCHOL-BALTER, ALAN SCHELLER-WOLF

Multiserver-job systems, where jobs require concurrent service at many servers, occur widely in practice. Essentially all of the theoretical work on multiserver-job systems focuses on maximizing utilization, with almost nothing known about mean response time. In simpler settings, such as many known-size single-server-job settings, minimizing mean response time is merely a matter of prioritizing small jobs. However, for the multiserver-job system, prioritizing small jobs is not enough, because we must also ensure servers are not unnecessarily left idle. Thus, minimizing mean response time requires prioritizing small jobs while simultaneously maximizing throughput. Our question is how to achieve these joint objectives.

We devise the ServerFilling-SRPT scheduling policy, which is the first policy to minimize mean response time in the multiserver-job model in the heavy traffic limit. In addition to proving this heavy-traffic result, we present empirical evidence that ServerFilling-SRPT outperforms all existing scheduling policies for all loads, with improvements by orders of magnitude at higher loads.

Because ServerFilling-SRPT requires knowing job sizes, we also define the ServerFilling-Gittins policy, which is optimal when sizes are unknown or partially known.

ACM Reference Format:
Isaac Grosof, Ziv Scully, Mor Harchol-Balter, Alan Scheller-Wolf. 2022. Optimal Scheduling in the Multiserver-job Model under Heavy Traffic. 1, 1 (August 2022), 29 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

1.1 The multiserver-job model

Traditional multiserver queueing theory focuses on models, such as the M/G/k, where every job occupies exactly one server. For decades, these models remained popular because they captured the behavior of computing systems, while being amenable to theoretical analysis. However, such one-server-per-job models are no longer representative of many modern computing systems.

Consider today's large-scale computing centers, such as the those of Google, Amazon and Microsoft. While the *servers* in these data centers still resemble the *servers* in traditional models such as the M/G/k, the *jobs* have changed: each job now requires many servers, which it holds simultaneously. While some jobs require few servers, other jobs require many more servers. For instance, in Fig. 1, we show the distribution of the number of CPUs requested by jobs in Google's recently published trace of its "Borg" computation cluster [12, 34]. The distribution is highly variable, with jobs requesting anywhere from 1 to 100,000 normalized CPUs. Throughout this paper, we will focus on this "multiserver-job model" (MSJ), by which we refer to the common situation in modern systems where each job concurrently occupies a fixed number of servers (typically more than one), throughout its time in service.

The multiserver-job model is fundamentally different from the one-server-per-job model. In the one-server-per-job model, any work-conserving scheduling policy such as FCFS can achieve full

Author's address: Isaac Grosof, Ziv Scully, Mor Harchol-Balter, Alan Scheller-Wolf.

XXXX-XXXX/2022/8-ART $15.00
https://doi.org/XXXXXXX.XXXXXXX

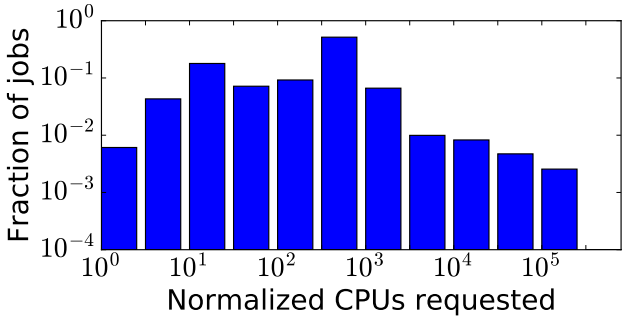

Fig. 1. The distribution of number of CPUs requested by jobs in Google's recently published Borg trace [34]. Number of CPUs is normalized so that the smallest job in the trace uses one normalized CPU.

server utilization. In the multiserver-job model, a naïve scheduling policy such as FCFS will waste more servers than necessary. As a result, server utilization and system stability are dependent on the scheduling policy in the multiserver-job model. While finding throughput-optimal scheduling policies is a challenge, several such policies are known, including MaxWeight [20], Randomized Timers [9, 21], and ServerFilling [12]. However, none of these policies give consideration to optimizing mean response time; each policy solely focuses on optimizing throughput. In fact, the empirical mean response time of such policies can be very poor [9], motivating our goal of finding throughput-optimal policies which moreover minimize mean response time.

1.2 The challenges of minimizing MSJ mean response time

In the M/G/k setting, where each job requires a single server, it was recently proven that the SRPT-k (Shortest Remaining Processing Time-k) scheduling policy minimizes mean response time in the heavy-traffic limit [13]. SRPT-k is a very simple policy: serve the k jobs of least remaining duration.

Unfortunately, in the multiserver-job system, trying to simply adapt the SRPT-k policy does not result in an optimal policy for two reasons:

- Prioritizing by remaining job duration is not the right way to minimize mean response time. We will show that an optimal policy must prioritize by remaining *size*, which we define to be the product of a job's *duration* (service time) and its *server need*, the number of servers the job requires, divided by the total number of servers k. We define these terms in more detail in Section 3.
- Even with this concept of size, a prerequisite for minimizing mean response time in the heavy-traffic limit is throughput-optimality, which requires a policy to efficiently utilize all of the servers whenever possible: Our policy must be throughput-optimal, while *also* prioritizing small jobs.

Our goal is to prove a result comparable to the SRPT-k result in [13], but for the MSJ setting. We therefore ask:

> What scheduling policy for the multiserver-job model should we use to *minimize mean response time* in the heavy-traffic limit?

By "heavy-traffic" we mean as load $\rho \to 1$, while the number of servers, k, stays fixed. The precise definition of load ρ and the heavy-traffic limit will be explained in detail in Section 3.

	Maximize throughput		Minimize mean response time	
Policies	Attempted	Proven	Attempted	Proven
MaxWeight [20]	✓	✓		
Randomized Timers [9, 21]	✓	✓		
ServerFilling [12]	✓	✓		
FCFS [6, 18, 23]				
Simple backfilling heuristics: First-Fit, BestFit, etc. [18, 37]	✓			
Size-aided backfilling: EASY, conservative, dynamic, etc. [4, 18]	✓			
Size-based heuristics: GreedySRPT, FirstFitSRPT, etc. [4]			✓	
Size & learning heuristics [14]	✓		✓	
ServerFilling-SRPT (Section 4)	✓	✓	✓	✓

Table 1. Comparison of multiserver-job scheduling policies

1.3 ServerFilling-SRPT and ServerFilling-Gittins

To answer this question, we introduce the ServerFilling-SRPT scheduling policy, the first scheduling policy to minimize mean response time in the multiserver-job model in the heavy traffic limit.

ServerFilling-SRPT is defined in the setting where k is a power of 2, and all server needs are powers of 2. This setting is commonly seen in practice in supercomputing and other highly-parallel computing settings.

To define ServerFilling-SRPT, imagine all jobs are ordered by their remaining size. Select the smallest initial subset M of this sequence such that the jobs in M collectively require at least k servers. Finally, place jobs from M into service in order of largest server need. This procedure is performed preemptively, whenever a job arrives or completes. As we show in Section 3.2, this procedure will fill all k servers whenever jobs with server need totaling at least k are present in the system. We use this property to prove in Section 4 that ServerFilling-SRPT minimizes mean response time in the heavy-traffic limit.

ServerFilling-SRPT requires the scheduler to know job durations, and hence sizes, in advance. Sometimes the scheduler does not have duration information. In the M/G/1 setting, when job sizes are unknown, the Gittins policy [10] is known to achieve optimal mean response time. We therefore introduce the ServerFilling-Gittins policy in Section 5. We prove similar heavy-traffic optimality results for ServerFilling-Gittins.

Beyond the setting where all server needs are powers of 2, we also consider the setting where all server needs perfectly divide k. For this setting, we define the DivisorFilling-SRPT and DivisorFilling-Gittins policies in Appendix C, and prove similar results for each policy.

1.4 Comparison with other policies

In Table 1, we compare our ServerFilling-SRPT policy and our asymptotic optimality results with prior work in the multiserver-job setting. Prior work broadly falls into two categories: theoretical results focusing on throughput-optimality, and good heuristic policies, which may or may not use duration information. Our result is the first to theoretically study the problem of minimizing mean response time.

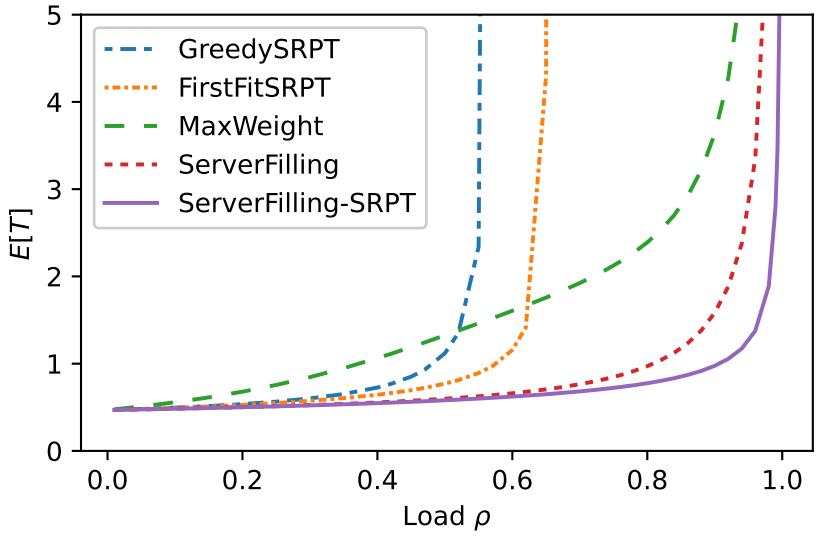

Fig. 2. Simulated mean response time $E[T]$ as a function of load ρ in a multiserver-job setting. $k = 8$ total servers. Server need is sampled uniformly from $\{1, 2, 4, 8\}$. Size is exponentially distributed, independent of the number of servers required. Policies defined in Section 6. Simulations use 10^7 arrivals. Loads $\rho \in [0, 0.999]$ simulated.

Fig. 2 compares the mean response time of ServerFilling-SRPT to that of prior throughput-optimal policies, as well as naïve size-based heuristic policies. These selected policies are representative of the empirical behavior of a wide variety of prior policies: Some of the policies shown have SRPT-like behavior, some policies are throughput-optimal, but only our ServerFilling-SRPT policy achieves both. Correspondingly, ServerFilling-SRPT has the best mean response time at all loads ρ, often by huge margins.

1.5 Summary of our contributions and outline

- In Section 3, we introduce the ServerFilling-SRPT scheduling policy.
- In Section 4, we bound the mean response time of ServerFilling-SRPT. Our analysis introduces a new way of bounding the total "relevant" work in the system. Using that bound, we prove that ServerFilling-SRPT has asymptotically optimal mean response time as load $\rho \to 1$.
- In Section 5, we introduce the ServerFilling-Gittins scheduling policy, in the setting of unknown or partially-known job sizes and durations. We prove a similar bound and asymptotic optimality result for ServerFilling-Gittins.
- In Section 6, we empirically evaluate ServerFilling-SRPT using simulation, showing that it outperforms prior policies on realistic distributions over a variety of loads, not just the $\rho \to 1$ limit.

2 PRIOR WORK

There are no prior optimality or asymptotic optimality results for mean response time in the multiserver-job system. The most similar system where such results have been proven is the M/G/k, a multiserver system with single-server jobs, and those results build off of classical results in the M/G/1.

2.1 Single-server-job models (one server per job)

In the single-server setting, the Shortest Remaining Processing Time policy (SRPT), which prioritizes the job of least remaining size, has been proven to minimize mean response time in the known-size M/G/1, as well as the worst-case single-server system [24, 25]. Note that in the single-server setting, a job's size is simply its duration. In the unknown- and partially-known-size settings, the Gittins policy is known to minimize mean response time in the M/G/1 [10, 29].

In the M/G/k, where jobs require a single server, [13] proves that the SRPT-k scheduling policy, the natural analogue of SRPT in the M/G/k, asymptotically minimizes mean response time in the known-size M/G/k in the heavy-traffic limit. There, as in this paper, load ρ is defined as the long-term average fraction of busy servers; $\rho \to 1$ is the heavy-traffic limit. Specifically, the paper shows that

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

This is proven despite the fact that the optimal policy OPT-k is unknown.

In the unknown job size setting, similar asymptotic optimality results for mean response time have been proven for the Gittins-k policy [27] and a monotonic variant thereof [28]. Moreover, for the Gittins-k policy, these results generalize to the partially-known job size setting, such as a setting with imperfect job size estimates.

2.2 Multiserver-job model (many servers per job)

Theoretical results in the multiserver-job model are limited. The *blocking* model, where arriving jobs either immediately receive service or are dropped, has received significant attention, with many strong results [2, 33, 35, 39] such as the exact steady state distribution. However, without any queue these models don't fit most real computing systems well. In the *queueing* MSJ model, which we focus on, results are much more limited. The best-studied scheduling policy is the first-come first-served (FCFS) policy. Stability region results for FCFS are known in several limited settings [22, 23], and steady state results are only known in the case of two servers [3, 8, 19].

Recently, the Work Conserving Finite Skip (WCFS) framework has been used to analytically characterize response time under the ServerFilling and DivisorFilling scheduling policies [12], both of which serve jobs in near-FCFS order. We modify the ServerFilling and DivisorFilling policies to prioritize jobs of shortest remaining size (SRPT). We then use a novel proof procedure to demonstrate that ServerFilling-SRPT and DivisorFilling-SRPT achieve asymptotically similar mean response time to SRPT in an analogous M/G/1 setting.

There has also been work in the *scaling* multiserver-job model, where one analyzes a sequence of multiserver-job systems with jointly increasing arrival rate, number of servers, and server needs [16, 38]. The regimes investigated include multiserver-job analogues of the Halfin-Whitt regime. Our results complement these, as we study a system with a fixed number of servers k in the heavy-traffic limit.

2.3 Supercomputing

Supercomputing centers are one of the originators of the multiserver-job model: Supercomputing jobs closely resemble the jobs in the multiserver-job model. Jobs commonly demand anywhere from one core to thousands of concurrent cores [17, 36]. Unfortunately, all of the papers in this area focus on simulation or empirical results, rather than analytical results [1, 5–7, 18, 31, 32]. These papers study a variety of scheduling policies, such as FCFS, various backfilling policies, and other more novel policies. Backfilling policies considered include simpler, no-duration-information policies such as FirstFit and BestFit [18, 37], as well as more complex, duration-information-based

policies such as EASY backfilling [30], conservative backfilling [30], Smallest Area[1] First-backfilling [4], Dynamic Backfilling [18], and many more.

Often the primary goal of these papers is achieving high utilization, with secondary goals including minimizing mean response time and fairness between different types of jobs. However, their settings are typically more restrictive than our setting: preemption is either limited or impossible. As a result, maximum utilization is lower, often around $\rho = 70\%$, and mean response times are often high near the utilization threshold. We leverage preemption to achieve much stronger results.

2.4 Virtual Machine Scheduling

In the field of cloud computing, the Virtual Machine (VM) scheduling problem is essentially a multi-resource generalization of the multiserver-job model, which can include preemption. In this model, rather than a single requirement like server need, each job requires concurrent utilization of several different limited resources, such as RAM, CPU, GPU, network bandwidth, etc. Of course, any results in this more general setting also apply to the multiserver-job setting. In the VM scheduling literature, papers typically focus on finding a throughput-optimal policy. Two major categories of such policies are the preemptive MaxWeight [20] and non-preemptive Randomized Timers [9, 21] scheduling frameworks.

These papers focus entirely on achieving throughput optimality, and the mean response time of the resulting policies can be poor, as several of the above papers note. Work on optimal mean response time in the VM scheduling literature has been limited to heuristic policies and empirical evaluation [14].

3 SETTING

3.1 Multiserver-job Model

The Multiserver-job (MSJ) model is a multiserver queueing model where each job requires a fixed number of servers concurrently over its entire time in service. The jobs are therefore called "multiserver jobs."

A job j has two requirements: A server need k_j and a service duration d_j. These requirements are sampled i.i.d. from some joint distribution with random variables (K, D). Note that K and D can be correlated. A job's server need k_j is at most the total number of servers, k. The total server need of the jobs in service at any time must sum to at most k. The job j will complete after d_j time in service.

We assume Poisson arrivals with rate λ, and we assume preemption is allowed with no loss of progress.

Let a job j's *size* s_j be defined as $k_j d_j / k$, and likewise define the job size distribution $S = KD/k$. Job j's size can be viewed as the area of a rectangle with height equal to the job's duration d_j and width equal to k_j/k, the fraction of the total service capacity occupied by job j. Likewise, a job's *remaining size* r_j is its remaining duration multiplied by k_j/k. We define a job j's *service rate* to be k_j/k, the rate at which the job's remaining size decreases during service. We define a job's age a_j to be $s_j - r_j$, which increases at rate k_j/k whenever the job is in service.

A *resource-pooled* M/G/1 is defined to be a system with a single server with the same capacity as all k original servers pooled together, and the same arrival rate λ and job size distribution S as the original MSJ system. We allow the resource-pooled M/G/1 to divide its capacity arbitrarily among the jobs in the system. In particular, while jobs in the MSJ system have fixed service rates depending on their server needs, in the resource-pooled system any combination of service rates is allowed, decreasing remaining sizes accordingly. Note that the resource-pooled system is strictly

[1]The term "area" used in [4] is equivalent to our "size".

more flexible than the MSJ system, so the optimal policy in the resource-pooled system is superior to the optimal policy in the MSJ system.

Let $W(t)$ be the total work in the system at time t: The sum of the remaining sizes r_j of all job's in the system at time t. Let $B(t)$ be the *"busyness"* of the system at time t: The fraction of servers that are occupied at time t. Note that $B(t)$ is also the total service rate of all jobs in service at time t, and so $B(t) = -\frac{d}{dt}W(t)$, outside of arrival moments. We also define W and B to be the corresponding stationary random variables.

Let *load* $\rho = \lambda E[S]$ be the long-run average rate at which work arrives to the system. We assume $\rho < 1$ as a necessary condition for stability. We will focus on settings where $\rho < 1$ is also sufficient for stability for some feasible scheduling policy. Note that ρ is a constant and that $\rho = E[B]$, under any scheduling policy for which the system is stable.

Next, let us define an *r-relevant* job, where r is a remaining size threshold. A job j is r-relevant if $r_j \leq r$. This terminology is in reference to the tagged job analysis used in studying SRPT in the M/G/1 and M/G/k settings [13, 25]; in those settings, the service of a job with remaining size r is only affected by the presence of r-relevant jobs in the system. The multiserver-job system is not as simple, so we do not employ a tagged-job approach, but we reuse the terminology.

Correspondingly, let the r-relevant work $W_r(t)$ be the total remaining size of all r-relevant jobs in the system at time t, and let $B_r(t)$ be the fraction of servers which are serving r-relevant jobs at time t. Define B_r and W_r correspondingly. The core of our proof lies in bounding expectations of random variables involving B_r and W_r, and combining these with a characterization of mean response time $E[T]$ in terms of B_r and W_r.

Next, let us define the r-relevant load ρ_r to be the long-run average r-relevant busyness of the system. A job with size s_j receives $\min(s_j, r)$ service while having remaining size $\leq r$. As a result, $\rho_r = \lambda E[\min(S, r)] = E[B_r]$. We further divide the r-relevant load based on whether the job in question has initial size $\leq r$. Let the *arrival load* $\rho_r^A = \lambda E[S\mathbb{1}\{S < r\}]$, and let the *recycled load* $\rho_r^R = \lambda r P(S > r)$. Note that $\rho_r = \rho_r^A + \rho_r^R$. Note also that ρ_r, ρ_r^A, and ρ_r^R are all not dependent on the policy π.

Finally, let us define an *r-recycling moment* to be a moment when a job j with initial size $s_j > r$ reaches remaining size $r_j = r$. Let $E_r[\cdot]$ be an expectation taken over r-recycling moments, just prior to the job recycling.

3.2 ServerFilling-SRPT

This paper considers two settings of server needs:

- The "power of two" setting: k is power of two, and all server needs k_j are powers of two.
- The "divisible" setting: k is general, and all server needs k_j are divisors of k.

Corresponding to these two settings, we have two policies of interest: ServerFilling-SRPT for the power of two setting, and DivisorFilling-SRPT for the divisible setting. We define ServerFilling-SRPT here, and DivisorFilling-SRPT in Appendix C. When writing equations throughout the paper, we abbreviate ServerFilling-SRPT as SFS-k.

To implement SFS-k, start by ordering jobs in increasing order of remaining size r_j, breaking ties arbitrarily. Define $j_1, j_2, \ldots$ such that

$$r_{j_1} \leq r_{j_2} \leq \ldots$$

Next, consider initial subsets of this ordering:

$$\{j_1\}, \{j_1, j_2\}, \{j_1, j_2, j_3\} \ldots$$

We are interested in the smallest initial subset M in which the total server need is at least k. In other words, let i^* be the smallest index such that

$$\sum_{i=1}^{i^*} k_{j_i} \geq k.$$

If there is no such index, then ServerFilling-SRPT serves all jobs in the system simultaneously.

Otherwise, ServerFilling-SRPT will serve a subset of $M = \{j_1, j_2, \ldots, j_{i^*}\}$. Among this subset, ServerFilling-SRPT prioritizes jobs of largest server need, placing jobs into service in descending order of server need, until no servers remain or the next job cannot fit, breaking ties by smallest remaining size, and further ties arbitrarily.

In the power-of-two setting, ServerFilling-SRPT guarantees the following strong property: At all times, either all servers are busy, or all jobs are in service. This was proven for the ServerFilling policy [12, Lemma 1], which is identical to ServerFilling-SRPT, except that jobs are ordered in arrival order, rather than SRPT order. For completeness, we reprove this result here:

Lemma 3.1. *Under the ServerFilling-SRPT policy, in the power-of-two setting, if the total server need of jobs in the system is at least k servers, all k servers are busy.*

Proof. Recall that M is a set of jobs, each with server need a power of two, which have a total server need of at least k. Label the jobs $m_1, m_2, \ldots$ in decreasing order of server need, tiebroken by least remaining size.

$$k_{m_1} \geq k_{m_2} \geq \ldots$$

Let $\textsc{need}(z)$ represent the total server need of the first z jobs in this ordering:

$$\textsc{need}(z) = \sum_{i=1}^{z} k_{m_i}$$

The set of jobs served by ServerFilling-SRPT is an initial sequence of this server need ordering: $\{m_i \mid i \leq \ell\}$ for some ℓ. Specifically, the index ℓ up to which ServerFilling-SRPT serves jobs is the largest index z such that $\textsc{need}(z) \leq k$. To prove Lemma 3.1, it suffices to show that $\textsc{need}(\ell) = k$.

Note that $\textsc{need}(0) = 0$ and $\textsc{need}(|M|) \geq k$. As a result, $\textsc{need}(z)$ must cross k at some point. To prove that $\textsc{need}(\ell) = k$, it suffices to prove that:

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

To prove (1), let us define $\textsc{remain}(z)$, the number of servers remaining after z jobs have been placed into service:

$$\textsc{remain}(z) = k - \textsc{need}(z)$$

Because all server needs k_j are powers of two, we will show that $\textsc{remain}(z)$ carries an important property:

$$\textsc{remain}(z) \text{ is divisible by } k_{m_{z+1}} \text{ for all } z. \tag{2}$$

We will use (2) to prove (1). We write $a|b$ to indicate that a divides b.

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

Now, we are ready to prove (1). Assume for contradiction that such an ℓ' exists. Then $\textsc{remain}(\ell') > 0$, and $\textsc{remain}(\ell'+1) < 0$. Because $\textsc{remain}(\ell'+1) = \textsc{remain}(\ell') - k_{m_{\ell'+1}}$, we therefore know that

$k_{m_{\ell'+1}} > \text{REMAIN}(\ell')$. But from (2), we know that $k_{m_{\ell'+1}}$ divides $\text{REMAIN}(\ell')$, which is a contradiction. □

Note that Lemma 3.1 remains true if the power-of-two setting is replaced by the power-of-x setting, for any integer x. In fact, the only condition on the server needs necessary to prove Lemma 3.1 is that all server needs divide k, and all server needs divide all larger server needs.

An important corollary of Lemma 3.1 is a property which we call "relevant work efficiency":

Corollary 3.1 (Relevant work efficiency). *Under the ServerFilling-SRPT policy, in the power-of-two setting, if there are k or more r-relevant jobs in the system, all servers are occupied by r-relevant jobs, meaning that $B_r = 1$.*

Proof. Note that $|M| \leq k$, because M is the smallest initial subset of the SRPT ordering with total server need at least k, and all jobs have server need at least 1. Therefore, if there are k or more r-relevant jobs in the system, then all jobs in M are r-relevant, so ServerFilling-SRPT fills all k servers with r-relevant jobs, meaning that $B_r = 1$. □

Corollary 3.1 is the sole property of ServerFilling-SRPT that we will use to prove our main theorems, Theorems 4.1 and 4.2.

DivisorFilling-SRPT in the divisible setting also satisfies the relevant work efficiency property: If there are k or more r-relevant jobs in the system, then $B_r = 1$, as we discuss in Appendix C. As a result, our main theorems, Theorems 4.1 and 4.2, also hold for DivisorFilling-SRPT.

4 SERVERFILLING-SRPT: ASYMPTOTICALLY OPTIMAL MEAN RESPONSE TIME

4.1 Summary of Results and Proofs

To prove the optimality of ServerFilling-SRPT, we will compare ServerFilling-SRPT's mean response time against a resource-pooled M/G/1/SRPT system with the same size distribution S. Recall that the resource-pooled M/G/1/SRPT system combines the power of all k servers into a single server, which can work on any job or any mixture of jobs. This resource-pooled system is strictly more flexible than the multiserver-job system, so the optimal policy in the resource-pooled system forms a lower bound on the optimal policy in the MSJ system. Because SRPT minimizes mean response time in the M/G/1, M/G/1/SRPT forms a lower bound on optimal in the MSJ system.

We will upper bound the gap in mean response time between ServerFilling-SRPT and resource-pooled M/G/1/SRPT for all loads ρ, and prove that the gap asymptotically grows slower than $E[T^{SRPT\text{-}1}]$. By doing so, we will show that ServerFilling-SRPT is asymptotically optimal in the multiserver-job system.

First, we prove a bound on the gap in mean response time between ServerFilling-SRPT and resource-pooled M/G/1/SRPT, which we call "SRPT-1":

Theorem 4.1. *For all loads ρ, in the power-of-two setting, the mean response time gap between ServerFilling-SRPT and SRPT-1 is at most*

$$E[T^{SFS\text{-}k}] - E[T^{SRPT\text{-}1}] \leq \frac{(e+1)(k-1)}{\lambda} \ln \frac{1}{1-\rho} + \frac{e}{\lambda}$$

The same is true of DivisorFilling-SRPT in the divisible setting.

Proof deferred to Section 4.3. □

We use this bound to prove that ServerFilling-SRPT yields optimal mean response time in the heavy-traffic limit:

Theorem 4.2. *If $E[S^2(\log S)^+] < \infty$, then ServerFilling-SRPT is asymptotically optimal in the multiserver-job system:*

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

The same is true of DivisorFilling-SRPT in the divisible setting.

Proof deferred to Section 4.3. □

The condition $E[S^2(\log S)^+] < \infty$ is very slightly stronger than finite variance.

In Section 5, we generalize both results to the settings of unknown- and partially-known job duration.

4.2 A Novel Proof Technique

4.2.1 Challenges of multiserver-job analysis. As mentioned in Section 1, mean response time analysis in the multiserver-job system is a difficult problem, with no size- or age-based scheduling policies having been analyzed. The difficulty arises from two sources: First, analyzing the mean response time of any system with multiple servers under a size- or age-based scheduling policy is already very difficult, even in a single-server-job setting such as the M/G/k. New techniques based on relevant work have recently been developed to handle this challenge. The first such analysis is as recent as 2018, when the SRPT-k policy was analyzed in the M/G/k [13], followed by the analysis of the monotonic-Gittins-k and Gittins-k policies in the M/G/k in 2020 and 2021 [27, 28].

Unfortunately, the multiserver-job system presents a major additional challenge. We will show in Sections 4.2.3 and 4.2.4 that these recent techniques for multiserver systems break when dealing with our multiserver-job system. As a result, we need a new technique to analyze the multiserver-job systems, which we introduce in Section 4.2.5.

4.2.2 Key idea of previous approaches: Relevant work similarity. The first step in applying relevant-work-based techniques [13, 27, 28] is to prove a property which we call "relevant work similarity":

Definition 4.1. *A policy π achieves* relevant work similarity (RWS) *if, for all remaining sizes r (or ranks[2] r), the policy π system and the optimal resource-pooled system OPT-1 (e.g. SRPT-1 or Gittins-1) have similar expected r-relevant work:*

$$E[W_r^{\pi}] - E[W_r^{OPT\text{-}1}] \le O(r).$$

The RWS property holds for all three policies and systems analyzed previously [13, 27, 28], as well as for ServerFilling-SRPT. Unfortunately, the RWS property is not sufficient on its own to tightly bound mean response time, or to prove asymptotically optimal mean response time.

4.2.3 First attempt: Tagged job approach. One way to build on the RWS property to prove asymptotic optimality is to use the tagged job approach, employed by the SRPT-k [13] and monotonic-Gittins-k [28] results. The tagged job approach combines the RWS property with an additional property, which we call "relevant work implies response time":

Definition 4.2. *A policy π achieves* relevant work implies response time (RW→RT) *if the following holds: If a generic tagged job of size r sees some amount x of r-relevant work in each of the policy π system and the optimal resource-pooled system OPT-1, then its expected response must be similar (within $O(r)$) in the two systems.*

[2]Rank is the analogue of remaining size under the Gittins policy.

If the RWS and RW→RT properties can both be proven for some policy π, it is relatively straightforward to tightly bound mean response time and prove that the policy π has asymptotically optimal mean response time. Unfortunately, for our ServerFilling-SRPT policy, the RW→RT property fails, meaning that the tagged-job approach cannot be used.

For a counterexample to the RW→RT property for ServerFilling-SRPT, consider a scenario where the tagged job requires 1 server and has the smallest size of any job in the system, and where it sees many jobs on arrival, all of which require an even number of servers and have larger remaining sizes. Furthermore, assume that arriving jobs rarely require 1 server. The resource-pooled SRPT-1 system will quickly complete the tagged job, as it has the smallest remaining size of any job in the system.

In contrast, the ServerFilling-SRPT system will not quickly complete the tagged job, because ServerFilling-SRPT prioritizes the jobs of largest server need among the initial subset M, as defined in Section 3.2. The tagged job will need to wait until the system empties or additional 1-server jobs arrive to be served. Clearly, similar relevant work does not imply similar response time.

This is an inherent difficulty of the multiserver-job system: Serving the tagged job any earlier would require leaving at least one server empty, as the tagged job is the only job requiring an odd number of servers, given the power-of-two setting. This could endanger throughput-optimality. As a result, the tagged-job approach cannot be used to effectively analyze the multiserver-job system.

4.2.4 Second Attempt: Gittins-k. The analysis of the Gittins-k policy for the M/G/k [27] also relies on the RWS property, which again is insufficient alone to prove asymptotically optimal mean response time in their setting. As in our setting, for the Gittins-k system, the RW→RT property fails, so the tagged-job approach cannot be employed.

The authors take a different approach: They introduce WINE [27, Theorem 6.3], our Lemma 4.3, a new identity that relates response time and relevant work in all systems.[3] WINE implies

$$E[T^{\pi\text{-}k}] - E[T^{OPT\text{-}1}] = \frac{1}{\lambda}\int_0^\infty \frac{E[W_r^{\pi\text{-}k}] - E[W_r^{OPT\text{-}1}]}{r^2}. \tag{3}$$

WINE is more general than the RW→RT property, because RW→RT only holds in certain systems.

We can see from (3) that the RWS property is almost enough to bound mean response time, but the $O(r)$ bound is too loose to show that the integral converges. The authors therefore prove a stronger version of the RWS property at sufficiently low and high ranks r. Combining their strengthened bounds with WINE, they prove that Gittins-k achieves asymptotically optimal mean response time in the M/G/k.

However, their proof of a stronger version of RWS at low ranks r relies on the fact that under Gittins-k in the M/G/k, the job of least rank is guaranteed to be served. This fails when applied to ServerFilling-SRPT, because in our multiserver-job system the job of least rank is not guaranteed to receive service. See the counterexample given in Section 4.2.3.

4.2.5 Our approach. Our key idea is to directly focus on the integrated relevant work difference given in (3). This circumvents the need to strengthen the RWS property (like in Section 4.2.4) or prove an RW→RT property (like in Section 4.2.3).

We start with a key property of the ServerFilling-SRPT system, which we call "relevant work efficiency" (RWE). RWE states that if there are k or more r-relevant jobs in the system, then all servers are occupied by r-relevant jobs. We prove in Section 3.2, specifically in Corollary 3.1, that ServerFilling-SRPT satisfies the RWE property.

[3]The name "WINE", short for "work integral number equality" [26], is more recent than [27], but refers to their Theorem 6.3.

While one can show that RWE implies RWS, RWS alone is not enough, as discussed in Section 4.2.4. Instead, we use the RWE property to directly bound the integrated relevant work difference given in (3), thereby directly bounding the mean response time difference. We prove this result in Theorem 4.6. This forms the core of our proof that ServerFilling-SRPT achieves asymptotically optimal mean response time.

We note that our new technique is stronger than the techniques used in all three previous results [13, 27, 28]. In particular, one could use our technique to reprove all of the asymptotic optimality results in those papers. This follows from the fact that the multiserver-job model is a generalization of the M/G/k: A multiserver-job setting where all server needs are 1 is simply an M/G/k.

4.3 Proof of Main Results

Our goal is to bound the mean response time of the ServerFilling-SRPT policy, relative to the resource-pooled SRPT-1 policy.

To bound mean response time, we start by applying the "work integral number equality" (WINE) technique [26, 27] to write mean response time $E[T^\pi]$ for a general policy π in terms of expected relevant work $E[W_r^\pi]$. This technique was introduced in [27, Theorem 6.3], but we reprove it here for completeness.

Lemma 4.3 (WINE Identity [27]). *For an arbitrary scheduling policy π, in an arbitrary system,*

$$E[T^\pi] = \frac{1}{\lambda} E[N^\pi] = \frac{1}{\lambda} \int_{r=0}^{\infty} \frac{E[W_r^\pi]}{r^2} dr$$

Proof. We will prove that at every moment in time,

$$N^\pi(t) = \int_{r=0}^{\infty} \frac{W_r^\pi(t)}{r^2} dr. \tag{4}$$

Recall that r-relevant work $W_r^\pi(t)$ is simply a sum over the r-relevant jobs in the system. As a result, we can consider the integral in (4) as a sum over the jobs in the system.

Consider a general job j, with remaining size r_j. The contribution of j to the r-relevant work $W_r^\pi(t)$ is r_j, for thresholds r such that $r_j \le r$, and 0 otherwise.

Therefore, the contribution of job j to the integral in (4) is

$$\int_{r=0}^{\infty} \frac{r_j \mathbb{1}\{r_j \le r\}}{r^2} dr = \int_{r=r_j}^{\infty} \frac{r_j}{r^2} dr = r_j \int_{r=r_j}^{\infty} \frac{1}{r^2} dr = r_j \frac{1}{r_j} = 1$$

Because the contribution of an arbitrary job is 1, the integral in (4) simply counts the number of jobs in the system at time t, giving $N^\pi(t)$ as desired.

Note that $E[T^\pi] = \frac{1}{\lambda} E[N^\pi]$, by Little's Law [15]. □

Now that we have written mean response time in terms of relevant work, we need to understand $E[W_r^\pi] - E[W_r^{SRPT\text{-}1}]$, the difference in r-relevant work between a general policy π and the resource pooled SRPT-1 system. To do so, we employ the work-decomposition law. This technique was introduced in [27], and we specialize it here to the SRPT setting. For completeness, we give the proof in Appendix A.

Lemma 4.4. *[27, Theorem 7.2] For an arbitrary scheduling policy π, in an arbitrary known-size system,*

$$E[W_r^\pi] - E[W_r^{SRPT\text{-}1}] = \frac{E[(1 - B_r^\pi) W_r^\pi] + \rho_r^R E_r[W_r^\pi]}{1 - \rho_r^A}$$

Proved in Appendix A. □

Combining Lemma 4.3, and specifically its implication (3), with Lemma 4.4, we arrive at the following characterization of the mean response time difference between a general policy π and SRPT-1:

Lemma 4.5. *For any scheduling policy π, in any system,*

$$E[T^{\pi}] - E[T^{SRPT\text{-}1}] = \frac{1}{\lambda}\int_0^{\infty} \frac{E[(1-B_r^{\pi})W_r^{\pi}]}{r^2(1-\rho_r^A)}dr \tag{5}$$

$$+ \frac{1}{\lambda}\int_0^{\infty} \frac{\rho_r^R E_r[W_r^{\pi}]}{r^2(1-\rho_r^A)}dr \tag{6}$$

Intuitively, (5) and (6) measure the inefficiency of the policy π relative to the ideal SRPT-1 system, through the lens of W_{π}^r, the r-relevant work under policy π.

The first term (5) measures the extent to which r-relevant work is present, but not being worked on. In the multiserver-job system, not all of the system can be devoted to a single job, so $E[(1-B_r^{\pi})W_r^{\pi}]$ will typically be nonzero.

The second term (6) measures the extent to which jobs r-recycle while r-relevant work is present in the system. In the multiserver-job system, not all of the system can be devoted to a single job, so jobs with remaining size above r will be worked on, and will r-recycle, while r-relevant work is present, so $E_r[W_r^{\pi}]$ will also typically be nonzero.

Our goal is to bound the magnitude of (5) and (6) under the ServerFilling-SRPT policy, in the power-of-two setting. We do so by making use of the key property of ServerFilling-SRPT, relevant work efficiency (Corollary 3.1): If there are k or more r-relevant jobs in the system, then $B_r = 1$.

We bound (5) in Theorem 4.6, and we bound (6) in Theorem 4.7.

Theorem 4.6 (Bound idle work). *Under the ServerFilling-SRPT policy, in the power-of-two setting,*

$$\int_{r=0}^{\infty} \frac{E[(1-B_r)W_r]}{r^2(1-\rho_r^A)}dr \le e(k-1)\left\lceil \ln\frac{1}{1-\rho}\right\rceil$$

The same is true of DivisorFilling-SRPT in the divisible setting.

Proof. First, we make use of the key fact about ServerFilling-SRPT (and DivisorFilling-SRPT), relevant work efficiency: If there are at least k jobs with rank $\le r$ in the system, then $B_r = 1$. This is proven in Corollary 3.1 for ServerFilling-SRPT, and in Appendix C for DivisorFilling-SRPT.

Let us define W_r^* to be the r-relevant work of the $k-1$ jobs of least remaining size in the system. Note that if $B_r < 1$, then $W_r = W_r^*$, for ServerFilling-SRPT and DivisorFilling-SRPT. As a result,

$$\int_{r=0}^{\infty} \frac{E[(1-B_r)W_r]}{r^2(1-\rho_r^A)}dr = \int_{r=0}^{\infty} \frac{E[(1-B_r)W_r^*]}{r^2(1-\rho_r^A)}dr$$

Next, we will break up the range of remaining sizes $r \in [0,\infty)$ into a finite set of buckets. Let $\{r_0, r_1, \ldots, r_m\}$ be a list of m different remaining sizes, where $r_0 = 0$. We will specify the list $\{r_i\}$ later. Implicitly, we will say that $r_{m+1} = \infty$. We can rewrite the above integral as:

$$\int_{r=0}^{\infty} \frac{E[(1-B_r)W_r^*]}{r^2(1-\rho_r^A)}dr = \sum_{i=0}^{m}\int_{r=r_i}^{r_{i+1}} \frac{E[(1-B_r)W_r^*]}{r^2(1-\rho_r^A)}dr \tag{7}$$

Next, we replace r with either r_i or r_{i+1}, selectively, to simplify things. Note that B_r is increasing as a function of r, because as we increase the rank r, more servers are busy with r-relevant jobs. Likewise, ρ_r^A is increasing as a function of r. Thus, for any $r \in [r_i, r_{i+1}]$,

$$B_{r_i} \le B_r \qquad \rho_r^A \le \rho_{r_{i+1}}^A$$

Substituting into the integral from (7), we find that

$$\int_{r=r_i}^{r_{i+1}} \frac{E[(1-B_r)W_r^*]}{r^2(1-\rho_r^A)}dr \le \int_{r=r_i}^{r_{i+1}} \frac{E[(1-B_{r_i})W_r^*]}{r^2(1-\rho_{r_{i+1}}^A)}dr$$

Next, let us perform some algebraic manipulation:

$$\int_{r=r_i}^{r_{i+1}} \frac{E[(1-B_{r_i})W_r^*]}{r^2(1-\rho_{r_{i+1}}^A)}dr = \int_{r=r_i}^{r_{i+1}} E\left[\frac{(1-B_{r_i})W_r^*}{r^2(1-\rho_{r_{i+1}}^A)}dr\right]$$
$$= E\left[\int_{r=r_i}^{r_{i+1}} \frac{(1-B_{r_i})W_r^*}{r^2(1-\rho_{r_{i+1}}^A)}dr\right] = E\left[\frac{1-B_{r_i}}{1-\rho_{r_{i+1}}^A}\int_{r=r_i}^{r_{i+1}} \frac{W_r^*}{r^2}dr\right] \tag{8}$$

Now, let us make use of the definition of W_r^*. Recall that W_r^* is the total remaining size of the $k-1$ jobs of least remaining size in the system.

$$W_r^* = \sum_{j=1}^{k-1} r_j \mathbb{1}\{r_j \le r\}$$

Substituting this into (8), we find it is equal to

$$= E\left[\frac{1-B_{r_i}}{1-\rho_{r_{i+1}}^A}\sum_{j=1}^{k-1}\int_{r=r_i}^{r_{i+1}} \frac{r_j\mathbb{1}\{r_j \le r\}}{r^2}dr\right] \tag{9}$$

Now, we will bound the integral in (9). As noted in Lemma 4.3, for an arbitrary remaining size r_j,

$$\int_{r=0}^{\infty} \frac{r_j\mathbb{1}\{r_j \le r\}}{r^2}dr = r_j\int_{r=r_j}^{\infty}\frac{1}{r^2}dr = r_j\frac{1}{r_j} = 1$$

As a result,

$$\int_{r=r_i}^{r_{i+1}} \frac{r_j\mathbb{1}\{r_j \le r\}}{r^2}dr \le 1$$

Substituting in this bound into (9), we find that

$$E\left[\frac{1-B_{r_i}}{1-\rho_{r_{i+1}}^A}\sum_{j=1}^{k-1}\int_{r=r_i}^{r_{i+1}} \frac{r_j\mathbb{1}\{r_j \le r\}}{r^2}dr\right] \le E\left[\frac{1-B_{r_i}}{1-\rho_{r_{i+1}}^A}(k-1)\right]$$
$$= (k-1)E\left[\frac{1-B_{r_i}}{1-\rho_{r_{i+1}}^A}\right] = (k-1)\frac{1-\rho_{r_i}}{1-\rho_{r_{i+1}}^A} \le (k-1)\frac{1-\rho_{r_i}^A}{1-\rho_{r_{i+1}}^A}$$

Returning all the way back to the beginning, we find that

$$\int_{r=0}^{\infty} \frac{E[(1-B_r)W_r]}{r^2(1-\rho_r^A)}dr \le (k-1)\sum_{i=0}^{m}\frac{1-\rho_{r_i}^A}{1-\rho_{r_{i+1}}^A} \tag{10}$$

We are now ready to construct the list $\{r_i\}$. Our goal in doing so is to minimize the sum

$$\sum_{i=0}^{m}\frac{1-\rho_{r_i}^A}{1-\rho_{r_{i+1}}^A}$$

Our only constrains are that $r_0 = 0$ and $r_{m+1} = \infty$. In particular,

$$1-\rho_{r_0}^A = 1-\rho_0^A = 1, \qquad 1-\rho_{r_{m+1}}^A = 1-\rho_\infty^A = 1-\rho$$

All other r_i thresholds are ours to choose.

We will set r_i such that the values $1-\rho^A_{r_i}$ form a geometric progression. In particular, define $r_1, r_2, \ldots$ to satisfy the following:

$$1-\rho^A_{r_1} = \frac{1}{e}, \qquad 1-\rho^A_{r_2} = \frac{1}{e^2}, \qquad \ldots \qquad 1-\rho^A_{r_i} = \frac{1}{e^i} \quad \forall i \le m \tag{11}$$

If the size distribution S is continuous, we choose r_i to exactly satisfy (11). If S is discontinuous, then ρ^A_r is discontinuous, so exact equality is not necessarily possible. However, it suffices to choose r_i such that

$$\frac{1}{e^i} \in [1-\rho^A_{r_i^+}, 1-\rho^A_{r_i^-}] \quad \forall i \le m,$$

which is always possible. By $^+$ and $^-$, we refer to the one-sided limits.

We then set $m = \lceil \ln \frac{1}{1-\rho} \rceil - 1$, so that $1-\rho^A_{r_{m+1}} = 1-\rho$ fits into the progression in (11). This choice of $\{r_i\}$ ensures that

$$\frac{1-\rho^A_{r_i}}{1-\rho^A_{r_{i+1}}} \le e \quad \forall i \le m$$

$$\sum_{i=0}^{m} \frac{1-\rho^A_{r_i}}{1-\rho^A_{r_{i+1}}} \le e(m+1) = e\left\lceil \ln \frac{1}{1-\rho} \right\rceil$$

Applying (10), we find that

$$\int_{r=0}^{\infty} \frac{E[(1-B_r)W_r]}{r^2(1-\rho^A_r)} dr \le e(k-1)\left\lceil \ln \frac{1}{1-\rho} \right\rceil. \qquad \square$$

Now, it remains to bound (6):

Theorem 4.7 (Bound recycled work). *Under the ServerFilling-SRPT policy, in the power-of-two setting,*

$$\int_{r=0}^{\infty} \frac{\rho^R_r E_r[W_r]}{r^2(1-\rho^A_r)} dr \le (k-1)\ln \frac{1}{1-\rho}$$

The same is true of DivisorFilling-SRPT in the divisible setting.

Proof. First, recall the key property of ServerFilling-SRPT and DivisorFilling-SRPT, relevant work efficiency: If there are at least k jobs with remaining size $\le r$ in the system, then $B_r = 1$. This is proven in Corollary 3.1 for ServerFilling-SRPT, and in Appendix C for DivisorFilling-SRPT.

When a job r-recycles, it must have been in service despite having remaining size $> r$. As a result, there are at most $k-1$ other jobs with remaining size $\le r$ present in the system at an r-recycling moment. Each such job contributes at most r work to W_r. As a result, $E_r[W_r] \le (k-1)r$.

$$\int_{r=0}^{\infty} \frac{\rho^R_r E_r[W_r]}{r^2(1-\rho^A_r)} dr \le \int_{r=0}^{\infty} \frac{(k-1)r\rho^R_r}{r^2(1-\rho^A_r)} dr = (k-1)\int_{r=0}^{\infty} \frac{\rho^R_r}{1-\rho^A_r}\frac{1}{r} dr$$

To bound the integrand, we will expand the definitions of ρ^R_r and ρ^A_r in the SRPT setting.

$$\rho^R_r = \lambda r P(S > r)$$
$$\rho^A_r = \lambda E[S\mathbb{1}\{S \le r\}]$$

We therefore bound as follows:

$$\frac{\rho^R_r}{1-\rho^A_r}\frac{1}{r} = \frac{\lambda r P(S>r)}{1-\lambda E[S\mathbb{1}\{S\le r\}]}\frac{1}{r} = \frac{\lambda P(S>r)}{1-\lambda E[S\mathbb{1}\{S\le r\}]}$$

Now, note that $P(S > r) = \frac{d}{dr}E[\min(S, r)]$, and that $E[\min(S, r)] \geq E[S\mathbb{1}\{S \leq r\}]$. As a result,

$$\frac{\lambda P(S > r)}{1 - \lambda E[S\mathbb{1}\{S \leq r\}]} \leq \frac{\lambda P(S > r)}{1 - \lambda E[\min(S, r)]} = \frac{\lambda \frac{d}{dr}E[\min(S, r)]}{1 - \lambda E[\min(S, r)]} = -\frac{d}{dr} \ln \frac{1}{1 - \lambda E[\min(S, r)]}$$

Integrating over all $r \in [0, \infty)$, we find that

$$\int_{r=0}^{\infty} \frac{\rho_r^R}{1 - \rho_r^A} \frac{1}{r} dr \leq \left[-\ln \frac{1}{1 - \lambda E[\min(S, r)]} \right]_{r=0}^{\infty} = \ln \frac{1}{1 - \rho}. \qquad \square$$

Now, we're ready to put it all together. We derive a bound on mean response time:

Theorem 4.1. *In any multiserver-job system, the difference in mean response time between ServerFilling-SRPT and SRPT-1 is at most*

$$E[T^{SFS\text{-}k}] - E[T^{SRPT\text{-}1}] \leq \frac{(e+1)(k-1)}{\lambda} \ln\left(\frac{1}{1-\rho}\right) + \frac{e}{\lambda}$$

The same is true of DivisorFilling-SRPT in the divisible setting.

Proof. From Lemma 4.5, we know that

$$\begin{aligned} &E[T^{SFS\text{-}k}] - E[T^{SRPT\text{-}1}] \\ &= \frac{1}{\lambda} \int_0^{\infty} \frac{E[(1 - B_r^{SFS\text{-}k}) W_r^{SFS\text{-}k}]}{r^2(1 - \rho_r^A)} dr + \frac{1}{\lambda} \int_0^{\infty} \frac{\rho_r^R E_r[W_r^{SFS\text{-}k}]}{r^2(1 - \rho_r^A)} dr \end{aligned}$$

We apply Theorem 4.6 and Theorem 4.7 to bound the two terms:

$$E[T^{SFS\text{-}k}] - E[T^{SRPT\text{-}1}] \leq \frac{1}{\lambda} e(k-1) \left\lceil \ln \frac{1}{1-\rho} \right\rceil + \frac{1}{\lambda}(k-1) \ln \frac{1}{1-\rho}.$$

We use the bound $\lceil x \rceil \leq x + 1$ to simplify the resulting expression. $\square$

Now, we use this bound to prove asymptotic optimality:

Theorem 4.2. *If* $E[S^2(\log S)^+] < \infty$,

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

The same is true of DivisorFilling-SRPT in the divisible setting.

Proof. From Theorem 4.1, we know that the gap $E[T^{SFS\text{-}k}] - E[T^{SRPT\text{-}1}]$ grows as $O(\log \frac{1}{1-\rho})$ in the $\rho \to 1$ limit. It is known that if $E[S^2(\log S)^+] < \infty$, then $E[T^{SRPT\text{-}1}] = \omega(\log \frac{1}{1-\rho})$ in the $\rho \to 1$ limit. This is proven in [27, Appendix B.2], and specifically in the proof of [27, Theorem 1.3]. $\square$

5 SERVERFILLING-GITTINS: ASYMPTOTIC OPTIMALITY WITH UNKNOWN SIZES

We generalize our results to the setting of unknown sizes or of partially known sizes (e.g. size estimates). To do so, we replace the SRPT job ordering with the Gittins job ordering, thus creating the ServerFilling-Gittins (SFG-k) and DivisorFilling-Gittins policies.

5.1 Background

The Gittins policy is the optimal scheduling policy for minimizing mean response time in the M/G/1 in the unknown and partially-known size settings [10, 29], filling the same role as SRPT in the known-size setting.

The Gittins policy is an age-based index policy, meaning that it assigns each job a rank according to the job's age and static characteristics (e.g. server need), as well as any other information the scheduler may have, and serves the job of least rank. In the blind MSJ setting, the Gittins rank function can be defined as follows: Let S_i be the job size distribution of jobs with server need i. Then a job with server need i and age a has rank:

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

The definition of the Gittins rank in settings where the server has more information is similar, but more complicated. For more details, see [27, 29].

We define the ServerFilling-Gittins policy by ordering jobs in increasing order of Gittins rank, and then applying the same ServerFilling procedure as described in Section 3.2. We define DivisorFilling-Gittins similarly, based on the DivisorFilling procedure given in Appendix C.

5.2 Notation

Our notation follows [27]. We start by defining a job state space X of all possible job states x. For instance, in the unknown size setting, a job's state is simply its age a. In the known-size setting, a job's state was its remaining size. Every state x is mapped to $\text{rank}(x)$. We call a job in state x r-relevant if $\text{rank}(x) < r$.

Next, we need to adjust the concept of "remaining size" slightly. We define $S_r(x)$, the *r-relevant remaining size* of a job in state x, to be the random variable denoting the amount of service the job needs in order to reach an r-irrelevant state or complete. In the known-size case, this amount of service was deterministic, but here it is a random variable.

We can now define W_r, the *r-relevant work* in the system, to be the total of all jobs' r-relevant remaining size in steady state. Likewise, B_r is the fraction of servers occupied by r-relevant jobs.

We also define two state distributions: X^A, the state of arriving jobs, and X_r^R, the state of jobs recycling relative to rank r. In the known-size case, X_r^R is deterministic, and in the unknown size case, X^A is deterministic, but in general both are random variables. We also define λ_r^R to be the rate at which jobs recycle relative to rank r. This is equal to λ times the expected number of r-recyclings per job.

We can now define the two constituents of r-relevant load, ρ_r^A and ρ_r^R.

$$\rho_r^A := \lambda E[S_r(X^A)]$$
$$\rho_r^R := \lambda_r^R E_r[S_r(X_r^R)]$$

Now, we are ready to state our main result for ServerFilling-Gittins.

5.3 Asymptotic Optimality for ServerFilling-Gittins

Our main result for ServerFilling-Gittins is an analogous bound on mean response time to Theorem 4.1, our bound on mean response time for ServerFilling-SRPT:

Theorem 5.1. *For all loads ρ, in the power-of-two setting, the mean response time gap between ServerFilling-Gittins and Gittins-1 is at most*

$$E[T^{SFG\text{-}k}] - E[T^{Gittins\text{-}1}] \leq \frac{(e+1)(k-1)}{\lambda} \ln \frac{1}{1-\rho} + \frac{e}{\lambda}$$

The same is true of DivisorFilling-Gittins in the divisible setting.

Note that this bound is in some ways stronger than the bound on Gittins-k given in [27]. Our bound is the first uniform bound on multiserver Gittins, meaning that our bound doesn't depend on S except via $E[S]$, unlike the bound on Gittins-k in [27]. Note also that the M/G/k is a special case of the multiserver-job system when server needs are all 1, and that in this special case, ServerFilling-Gittins specializes to Gittins-k. As a result, Theorem 5.1 is a strict improvement upon the bound given in [27].

We use this bound to prove that ServerFilling-Gittins yields optimal mean response time in the heavy-traffic limit:

Theorem 5.2. *If $E[S^2(\log S)^+] < \infty$, then ServerFilling-Gittins is asymptotically optimal in the multiserver-job system:*

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

The same is true of DivisorFilling-Gittins in the divisible setting.

Theorem 5.2 follows from Theorem 5.1 just as Theorem 4.2 follows from Theorem 4.1.

To prove Theorem 5.1, an analogous proof to the proof of Theorem 4.1 given in Section 4.3 suffices. We simply must replace certain quantities used in Section 4.3 with the equivalent quantities for the Gittins policy. Specifically, rather than thinking of a job as r-relevant if it has remaining size $\le r$, we instead think of a job as r-relevant if it has rank $\le r$ under the Gittins policy. We redefine W_r^π, B_r^π, and ρ_r, ρ_r^A, and ρ_r^R accordingly, as described in Section 5.2. For full details, see Appendix B.

The recycling term of our key background lemma Lemma 5.3 is likewise slightly different:

Lemma 5.3. *For any scheduling policy π,*

$$E[T^\pi] - E[T^{Gittins\text{-}1}] = \frac{1}{\lambda}\int_0^\infty \frac{E[(1-B_r^\pi)W_r^\pi]}{r^2(1-\rho_r^A)}dr \tag{12}$$

$$+ \frac{1}{\lambda}\int_0^\infty \frac{\lambda_r^R E_r[S_r(X_r^R)W_r^\pi]}{r^2(1-\rho_r^A)}dr \tag{13}$$

Here $\rho_R^r E_r[W_r^\pi]$ from Lemma 4.5 becomes $\lambda_r^R E_r[S_r(X_r^R)W_r^\pi]$. Lemma 5.3 follows from [27, Theorem 7.2].

Bounding the server-idleness term involving $E[(1-B_r^\pi)W_r^\pi]$ proceeds completely analogously to Theorem 4.6. Bounding the recycled work term involving $\lambda_r^R E_r[S_r(X_r^R)W_r^\pi]$ is likewise completely analogous to Theorem 4.7. For the full details, see Appendix B.

6 EMPIRICAL RESULTS

We have proven that ServerFilling-SRPT yields asymptotically optimal mean response time in the heavy-traffic limit (as $\rho \to 1$). To empirically validate our theoretical results and broaden our comparison to general ρ, we use simulation to compare the mean response time of ServerFilling-SRPT to that of several previously proposed policies:

MaxWeight: A throughput optimal policy which considers all possible sets of jobs that can be served at a time. Each job is given a weight equal the number of jobs in the system with the same server need. The set of jobs with the maximum total weight is served [20]. Note that this policy requires solving a NP-hard Bin Packing problem for each service.

ServerFilling: A policy which orders jobs in arrival order, then uses the same procedure to place jobs onto servers as our ServerFilling-SRPT policy specified in Section 3.2. ServerFilling is throughput-optimal in the power-of-two setting [12].

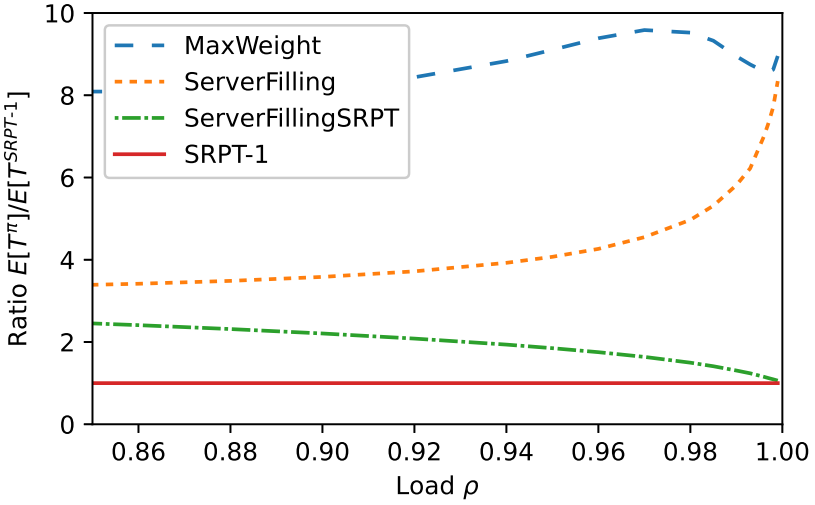

Fig. 3. Ratio of mean response time between several multiserver-job policies and SRPT-1. K uniformly sampled from $\{1, 2, 4, 8\}$. S exponentially distributed, independent of K. Each simulation consists of 10^7 arrivals. Loads up to $\rho = 0.999$ simulated.

We also compare against resource-pooled SRPT-1, our lower bound on the optimal policy.

In Fig. 3, we show the ratio of mean response time between the multiserver-job policies and SRPT-1. As proven in Theorem 4.2, for ServerFilling-SRPT, this ratio converges to 1, implying that ServerFilling-SRPT yields asymptotically optimal mean response time. In contrast, for MaxWeight and ServerFilling, the ratio is far from one, and appears to diverge. ServerFilling-SRPT has superior mean response time at all ρ.

In Fig. 4, we show a setting with higher variance job sizes, where $C^2 = 10$. In high-variance settings, making effective use of job size information is at its most important. Here, the ratio for ServerFilling-SRPT again converges smoothly to 1, while the ratios for MaxWeight and ServerFilling diverge rapidly.

In Section 1, Fig. 2, we also compared ServerFilling-SRPT against two size-based heuristic policies:

GreedySRPT: Order jobs in increasing order of remaining size. As long as sufficient servers are available, place jobs into service. When a job has higher server need than the remaining number of servers available, stop.

FirstFitSRPT: Order jobs in increasing order of remaining size. As long as sufficient servers are available, place jobs into service. If a job has higher server need than the remaining number of servers available, skip that job. Continue through the list of jobs, placing jobs into service if sufficient servers are available, until all servers are full, or all jobs are exhausted. This policy was studied under the name "Smallest Area First" [4].

GreedySRPT makes no effort to pack jobs efficiently onto servers, while FirstFitSRPT is unreliable at doing so. For both of these policies, the stability region is significantly smaller than the optimal stability region. This is why neither policy is depicted in Fig. 3 or Fig. 4, as both are unstable for all loads $\rho \geq 0.85$, and hence have infinite mean response time on this domain.

We summarize our experiments as follows: In all experiments, at all ρ, ServerFilling-SRPT has minimal mean response time.

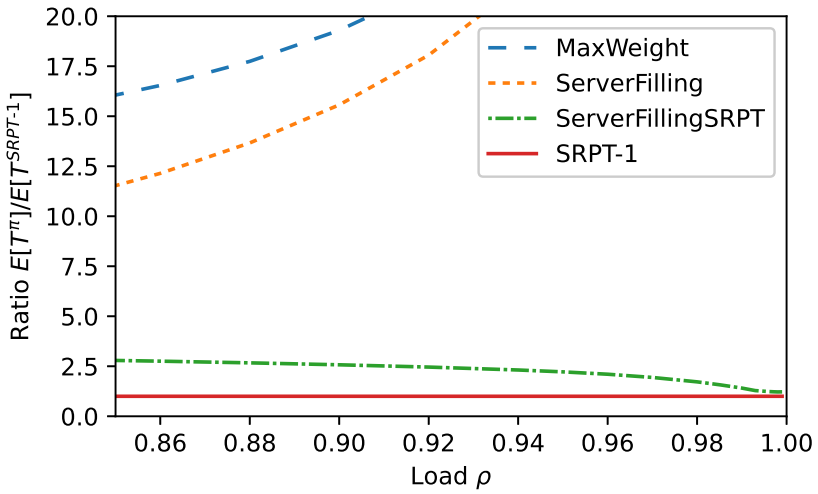

Fig. 4. Ratio of mean response time between several multiserver-job policies and SRPT-1 under high variance. K uniformly sampled from $\{1, 2, 4, 8\}$. S hyperexponentially distributed, $C^2 = 10$, independent of K. Each simulation consists of 10^7 arrivals. Loads up to $\rho = 0.999$ simulated.

7 CONCLUSION

We introduce the ServerFilling-SRPT scheduling policy for the multiserver-job system. We prove a tight bound on the mean response time of ServerFilling-SRPT in the power-of-two setting, which applies for all loads ρ. We use that bound to prove that ServerFilling-SRPT achieves asymptotically optimal mean response time in heavy traffic. We also show that ServerFilling-SRPT empirically achieves the best mean response time of any policy simulated, across all loads ρ. We also introduce the DivisorFilling-SRPT policy, in the more general divisible setting, and the ServerFilling- and DivisorFilling-Gittins policies, in the settings of unknown- and partially-known job sizes, proving similar asymptotic optimality results for each.

One of the major insights of this paper is that achieving asymptotically optimal mean response time requires prioritizing jobs of small remaining size without sacrificing the throughput of the system. ServerFilling-SRPT is the first policy to achieve both goals simultaneously.

The analysis technique introduced in this paper extends beyond ServerFilling-SRPT and the multiserver-job setting. In fact, it allows the analysis of any system and any policy in which the relevant work efficiency property (Corollary 3.1) can be proven.

One direction of future work is to study multiserver-job scheduling policies outside of the divisible setting. No mean response time analysis is currently known for any scheduling policy in this more general setting, much less any optimality results, so new techniques will likely be needed. In particular, no policy with the remaining work efficiency property can exist in this setting.

REFERENCES

[1] T. G. Armstrong, Z. Zhang, D. S. Katz, M. Wilde, and I. T. Foster. 2010. Scheduling many-task workloads on supercomputers: Dealing with trailing tasks. In *2010 3rd Workshop on Many-Task Computing on Grids and Supercomputers*. 1–10.

[2] E. Arthurs and J. S. Kaufman. 1979. Sizing a message store subject to blocking criteria. In *Proceedings of the third international symposium on modelling and performance evaluation of computer systems: Performance of computer systems*. 547–564.

[3] Percy H. Brill and Linda Green. 1984. Queues in Which Customers Receive Simultaneous Service from a Random Number of Servers: A System Point Approach. *Management Science* 30, 1 (1984), 51–68. https://doi.org/10.1287/mnsc.30.1.51 arXiv:https://doi.org/10.1287/mnsc.30.1.51

[4] Danilo Carastan-Santos, Raphael Y. De Camargo, Denis Trystram, and Salah Zrigui. 2019. 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)*. 1–10. https://doi.org/10.1109/CCGRID.2019.00010

[5] Yoav Etsion and Dan Tsafrir. 2005. A short survey of commercial cluster batch schedulers. *School of Computer Science and Engineering, The Hebrew University of Jerusalem* 44221 (2005), 2005–13.

[6] Dror G. Feitelson and Larry Rudolph. 1996. Toward convergence in job schedulers for parallel supercomputers. In *Job Scheduling Strategies for Parallel Processing*, Dror G. Feitelson and Larry Rudolph (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–26.

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

[8] D. Filippopoulos and H. Karatza. 2007. An M/M/2 parallel system model with pure space sharing among rigid jobs. *Mathematical and Computer Modelling* 45, 5 (2007), 491 – 530. https://doi.org/10.1016/j.mcm.2006.06.007

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

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

[11] Isaac Grosof, Mor Harchol-Balter, and Alan Scheller-Wolf. 2021. WCFS: A new framework for analyzing multiserver systems. *arXiv preprint arXiv:2109.12663* (2021).

[12] Isaac Grosof, Mor Harchol-Balter, and Alan Scheller-Wolf. 2022. WCFS: A new framework for analyzing multiserver systems. *Queueing Systems* (2022).

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

[14] M. Guo, Q. Guan, and W. Ke. 2018. Optimal Scheduling of VMs in Queueing Cloud Computing Systems With a Heterogeneous Workload. *IEEE Access* 6 (2018), 15178–15191.

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

[16] Yige Hong. 2022. Sharp Zero-Queueing Bounds for Multi-Server Jobs. *SIGMETRICS Perform. Eval. Rev.* 49, 2 (jan 2022), 66–68. https://doi.org/10.1145/3512798.3512822

[17] Minnesota Supercomputing Institute. 2020. Queues. https://www.msi.umn.edu/queues

[18] James Patton Jones and Bill Nitzberg. 1999. Scheduling for Parallel Supercomputing: A Historical Perspective of Achievable Utilization. In *Job Scheduling Strategies for Parallel Processing*, Dror G. Feitelson and Larry Rudolph (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–16.

[19] Sung Shick Kim. 1979. *M/M/s queueing system where customers demand multiple server use*. Ph. D. Dissertation. Southern Methodist University.

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

[21] Konstantinos Psychas and Javad Ghaderi. 2018. Randomized Algorithms for Scheduling Multi-Resource Jobs in the Cloud. *IEEE/ACM Transactions on Networking* 26, 5 (2018), 2202–2215. https://doi.org/10.1109/TNET.2018.2863647

[22] Alexander Rumyantsev, Robert Basmadjian, Sergey Astafiev, and Alexander Golovin. 2022. Three-level modeling of a speed-scaling supercomputer. *Annals of Operations Research* (2022), 1–29.

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

[24] Linus Schrage. 1968. A Proof of the Optimality of the Shortest Remaining Processing Time Discipline. *Operations Research* 16, 3 (1968), 687–690. https://doi.org/10.1287/opre.16.3.687 arXiv:https://doi.org/10.1287/opre.16.3.687

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

[26] Ziv Scully. 2021. WINE: A New Queueing Identity for Analyzing Scheduling Policies in Multiserver Systems. https://ziv.codes/pdf/wine-talk.pdf INFORMS Annual Meeting.

[27] Ziv Scully, Isaac Grosof, and Mor Harchol-Balter. 2020. The Gittins Policy is Nearly Optimal in the M/G/k under Extremely General Conditions. *Proc. ACM Meas. Anal. Comput. Syst.* 4, 3, Article 43 (Nov. 2020), 29 pages. https://doi.org/10.1145/3428328

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

[29] Ziv Scully and Mor Harchol-Balter. 2021. The Gittins Policy in the M/G/1 Queue. In *2021 19th International Symposium on Modeling and Optimization in Mobile, Ad hoc, and Wireless Networks (WiOpt)*. 1–8. https://doi.org/10.23919/

WiOpt52861.2021.9589051

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

[31] W. Tang, Z. Lan, N. Desai, D. Buettner, and Y. Yu. 2011. Reducing Fragmentation on Torus-Connected Supercomputers. In *2011 IEEE International Parallel Distributed Processing Symposium*. 828–839.

[32] W. Tang, D. Ren, Z. Lan, and N. Desai. 2012. Adaptive Metric-Aware Job Scheduling for Production Supercomputers. In *2012 41st International Conference on Parallel Processing Workshops*. 107–115.

[33] Oleg M Tikhonenko. 2005. Generalized Erlang problem for service systems with finite total capacity. *Problems of Information Transmission* 41, 3 (2005), 243–253.

[34] Muhammad Tirmazi, Adam Barker, Nan Deng, Md E. Haque, Zhijing Gene Qin, Steven Hand, Mor Harchol-Balter, and John Wilkes. 2020. Borg: The next Generation. In *Proceedings of the Fifteenth European Conference on Computer Systems* (Heraklion, Greece) *(EuroSys '20)*. Association for Computing Machinery, New York, NY, USA, Article 30, 14 pages. https://doi.org/10.1145/3342195.3387517

[35] Nico M. van Dijk. 1989. Blocking of finite source inputs which require simultaneous servers with general think and holding times. *Operations Research Letters* 8, 1 (1989), 45 – 52. https://doi.org/10.1016/0167-6377(89)90033-3

[36] Chad Vizino, J Kochmar, N Stone, and R Scott. 2005. Batch Scheduling on the Cray XT3. *CUG 2005* (2005).

[37] Juan Wang and Wenming Guo. 2009. The Application of Backfilling in Cluster Systems. In *2009 WRI International Conference on Communications and Mobile Computing*, Vol. 3. 55–59. https://doi.org/10.1109/CMC.2009.252

[38] Weina Wang, Qiaomin Xie, and Mor Harchol-Balter. 2021. Zero Queueing for Multi-Server Jobs. In *Abstract Proceedings of the 2021 ACM SIGMETRICS / International Conference on Measurement and Modeling of Computer Systems* (Virtual Event, China) *(SIGMETRICS '21)*. Association for Computing Machinery, New York, NY, USA, 13–14. https://doi.org/10.1145/3410220.3453924

[39] Ward Whitt. 1985. Blocking when service is required from several facilities simultaneously. *AT&T technical journal* 64, 8 (1985), 1807–1856.

A PROOF OF LEMMA 4.4 (WORK DECOMPOSITION)

Lemma 4.4. *[27, Theorem 7.2] For an arbitrary scheduling policy π, in an arbitrary system,*

$$E[W_r^\pi] - E[W_r^{SRPT\text{-}1}] = \frac{E[(1 - B_r^\pi)W_r^\pi] + \rho_r^R E_r[W_r^\pi]}{1 - \rho_r^A}$$

Proof. We will employ the rate conservation law, applied to the random variable $(W_r^\pi)^2$, the square of the stationary distribution of r-relevant work in the system. The rate conservation law states that, because $(W_r^\pi)^2$ is a stationary random variable, its expected rate of increase and decrease must be equal. This argument can be formalized further using Palm Calculus.

To find these rates of increase and decrease, let us first examine W_r^π. W_r^π decreases continuously as work completes, and increases by jumps whenever jobs arrive. W_r^π decreases at rate B_r^π, the fraction of servers that are occupied by r-relevant jobs. When a job arrives with size S, it contributes $[S\mathbb{1}\{S \le r\}]$ relevant work, increasing W_r^π by that amount. Such arrivals occur at rate λ. Finally, whenever a job recycles, by being served until its remaining size falls to r, it adds r relevant work to W_r^π.

Using these rates, we can calculate the expected rates of increase and decrease of $(W_r^\pi)^2$.

$$\begin{aligned}
\text{Increase due to arrivals:} &\quad \lambda E[(S\mathbb{1}\{S \le r\})^2] + 2\rho_r^A E[W_r^\pi] \\
\text{Increase due to recycling:} &\quad \lambda_r^R r^2 + 2\rho_r^R E_r[W_r^\pi] \\
\text{Decrease due to service:} &\quad 2E[B_r^\pi W_r^\pi]
\end{aligned}$$

Equating these rates, we find that

$$2E[B_r^\pi W_r^\pi] = \lambda E[(S\mathbb{1}\{S \le r\})^2] + 2\rho_r^A E[W_r^\pi] + \lambda_r^R r^2 + 2\rho_r^R E_r[W_r^\pi]$$

$$E[B_r^\pi W_r^\pi] = \frac{\lambda}{2}E[(S\mathbb{1}\{S \le r\})^2] + \rho_r^A E[W_r^\pi] + \frac{\lambda_r^R}{2}r^2 + \rho_r^R E_r[W_r^\pi]$$

$$E[W_r^\pi] - E[(1-B_r^\pi)W_r^\pi] = \frac{\lambda}{2}E[(S\mathbb{1}\{S \le r\})^2] + \rho_r^A E[W_r^\pi] + \frac{\lambda_r^R}{2}r^2 + \rho_r^R E_r[W_r^\pi]$$

$$E[W_r^\pi] = E[(1-B_r^\pi)W_r^\pi] + \frac{\lambda}{2}E[(S\mathbb{1}\{S \le r\})^2] + \rho_r^A E[W_r^\pi] + \frac{\lambda_r^R}{2}r^2 + \rho_r^R E_r[W_r^\pi]$$

$$E[W_r^\pi](1-\rho_r^A) = E[(1-B_r^\pi)W_r^\pi] + \frac{\lambda}{2}E[(S\mathbb{1}\{S \le r\})^2] + \frac{\lambda_r^R}{2}r^2 + \rho_r^R E_r[W_r^\pi]$$

$$E[W_r^\pi](1-\rho_r^A) = E[(1-B_r^\pi)W_r^\pi] + \rho_r^R E_r[W_r^\pi] + \frac{\lambda}{2}E[(S\mathbb{1}\{S \le r\})^2] + \frac{\lambda_r^R}{2}r^2 \tag{14}$$

Let us evaluate (14) in the case where the policy π is SRPT-1. The first two terms of the right-hand side are nonnegative terms depending on the policy π, while the second two terms are the same for all policies.

Let us start with the first term on the right-hand side, $E[(1-B_r^\pi)W_r^\pi]$. Note that under SRPT-1, if W_r^π is nonzero, i.e. if a r-relevant job is present, then SRPT-1 will serve a r-relevant job on its single server, and so $B_r^{SRPT\text{-}1} = 1$. As a result, either $W_r^{SRPT\text{-}1}$ or $1 - B_r^{SRPT\text{-}1}$ must always be zero, so this term is equal to 0.

Next, consider the second term, $\rho_r^R E_r[W_r^\pi]$. Recall that $E_r[\cdot]$ is an expectation over system states at times when r-relevant jobs recycle. In the SRPT-1 system, if a job is recycling by falling down to remaining size r, there must be no jobs in the system with remaining size less than r. As a result, $E_r[W_r^\pi] = 0$.

We therefore conclude that

$$E[W_r^{SRPT\text{-}1}](1-\rho_r^A) = \frac{\lambda}{2}E[(S\mathbb{1}\{S \le r\})^2] + \frac{\lambda_r^R}{2}r^2 \tag{15}$$

As an aside, note that this argument shows that SRPT-1 has the least value of $E[W_r^\pi]$ for any policy π. This fact, combined with Lemma 4.3, provides an alternative proof that SRPT-1 is the optimal scheduling policy in the M/G/1.

Subtracting (15) from (14), we find that

$$E[W_r^\pi](1-\rho_r^A) - E[W_r^{SRPT\text{-}1}](1-\rho_r^A) = E[(1-B_r^\pi)W_r^\pi] + \rho_r^R E_r[W_r^\pi]$$

$$E[W_r^\pi] - E[W_r^{SRPT\text{-}1}] = \frac{E[(1-B_r^\pi)W_r^\pi] + \rho_r^R E_r[W_r^\pi]}{1-\rho_r^A}$$

□

B SERVERFILLING-GITTINS PROOFS

Our results for ServerFilling-Gittins follow near-identical proofs as given in Section 4.3 for ServerFilling-SRPT. We give the proofs here for completeness.

Our starting point is the "work integral number equality" (WINE) identity [26, 27].

Theorem B.1 (Theorem 6.3, [27]). *The mean number of jobs and mean response time in an arbitrary system, under an arbitrary scheduling policy, is*

$$E[N] = \lambda E[T] = \int_0^\infty \frac{E[W_r]}{r^2} dr$$

Now, we can state the work-decomposition law in a Gittins system.

Theorem B.2 (Theorem 7.2, [27]). *For all $r \geq 0$, the mean r-relevant work gap between an arbitrary policy π and M/G/1/Gittins is*

$$E[W_r^\pi] - E[W_r^{Gittins\text{-}1}] = \frac{E[(1-B_r^\pi)W_r^\pi] + \lambda_r^R E_r[S_r(X_r^R)W_r^\pi]}{1-\rho_r^A} \tag{16}$$

We will handle the two numerator terms of (16) separately. Let us start by combining Theorem B.1 with Theorem B.2, and try to bound the resulting integral.

We must bound

$$E[T^\pi] - E[T^{Gittins\text{-}1}] = \frac{1}{\lambda}\int_0^\infty \frac{E[(1-B_r^\pi)W_r^\pi]}{r^2(1-\rho_r^A)} + \frac{1}{\lambda}\int_0^\infty \frac{\lambda_r^R E_r[S_r(X_r^R)W_r^\pi]}{r^2(1-\rho_r^A)}$$

We bound the first term in Lemma B.3 and the second term in Lemma B.5.

Lemma B.3.

$$\int_{r=0}^\infty \frac{E[(1-B_r)W_r]}{r^2(1-\rho_r^A)}dr \leq e(k-1)\lceil \ln \frac{1}{1-\rho} \rceil$$

Proof. First, we make use of the key fact about ServerFilling-Gittins (and DivisorFilling-Gittins): If there are at least k jobs with rank $\leq r$ in the system, then $B_r = 1$. Thus, we can replace W_r by W_r', the work of the $k-1$ jobs of least rank in the system:

$$\int_{r=0}^\infty \frac{E[(1-B_r)W_r]}{r^2(1-\rho_r^A)}dr = \int_{r=0}^\infty \frac{E[(1-B_r)W_r']}{r^2(1-\rho_r^A)}dr$$

Next, we will break up the ranks $r \in [0,\infty)$ into a finite set of buckets. Let $R = [r_1, r_2, \ldots]$ be a list of ranks, where $r_1 = 0$. We will specify the list R later. Implicitly, we will say that $r_{|R|+1} = \infty$. We can rewrite the above integral as:

$$\int_{r=0}^\infty \frac{E[(1-B_r)W_r']}{r^2(1-\rho_r^A)}dr = \sum_{i=1}^{|R|}\int_{r=r_i}^{r_{i+1}} \frac{E[(1-B_r)W_r']}{r^2(1-\rho_r^A)}dr \tag{17}$$

Next, we replace r with either r_i or r_{i+1}, selectively, to simplify things. Note that B_r is increasing as a function of r - as we increase the rank r, more servers are busy with r-relevant jobs. Likewise, ρ_r^A is increasing as a function of r. Thus,

$$B_{r_i} \leq B_r$$
$$\rho_r^A \leq \rho_{r_{i+1}}^A$$

Substituting into the integral from (7), we find that

$$\int_{r=r_i}^{r_{i+1}} \frac{E[(1-B_r)W_r']}{r^2(1-\rho_r^A)}dr \leq \int_{r=r_i}^{r_{i+1}} \frac{E[(1-B_{r_i})W_r']}{r^2(1-\rho_{r_{i+1}}^A)}dr$$

Next, let us perform some algebraic manipulation:

$$\begin{aligned}
&\int_{r=r_i}^{r_{i+1}} \frac{E[(1-B_{r_i})W_r']}{r^2(1-\rho^A_{r_{i+1}})}dr \\
&= \int_{r=r_i}^{r_{i+1}} E\left[\frac{(1-B_{r_i})W_r'}{r^2(1-\rho^A_{r_{i+1}})}dr\right] \\
&= E\left[\int_{r=r_i}^{r_{i+1}} \frac{(1-B_{r_i})W_r'}{r^2(1-\rho^A_{r_{i+1}})}dr\right] \\
&= E\left[\frac{1-B_{r_i}}{1-\rho^A_{r_{i+1}}}\int_{r=r_i}^{r_{i+1}} \frac{W_r'}{r^2}dr\right]
\end{aligned}$$

Note that B_{r_i} and W_r' are conditionally independent because given $\vec{X}$, the current states of the jobs in the system, the busyness B_{r_i} is deterministic. We can make this explicit:

$$E\left[\frac{1-B_{r_i}}{1-\rho^A_{r_{i+1}}}\int_{r=r_i}^{r_{i+1}} \frac{W_r'}{r^2}\right]dr = E\left[\frac{1-B_{r_i}}{1-\rho^A_{r_{i+1}}}\int_{r=r_i}^{r_{i+1}} \frac{E[W_r' \mid \vec{X}]}{r^2}dr\right] \tag{18}$$

Next, let us recall the definition of W_r':

$$\begin{aligned}
W_r' &= \sum_{j=1}^{k-1} S_r(X_j) \\
E[W_r' \mid \vec{X}] &= \sum_{j=1}^{k-1} E[S_r(X_j)|X_j]
\end{aligned}$$

Following [27], let us define $\textsc{service}(X_j, r)$ to be $E[S_r(X_j)|X_j]$, the expected r-relevant work of a job X_j.

Substituting this into (18), we find that

$$\begin{aligned}
&E\left[\frac{1-B_{r_i}}{1-\rho^A_{r_{i+1}}}\int_{r=r_i}^{r_{i+1}} \frac{E[W_r' \mid \vec{X}]}{r^2}dr\right] \\
=&E\left[\frac{1-B_{r_i}}{1-\rho^A_{r_{i+1}}}\sum_{j=1}^{k-1}\int_{r=r_i}^{r_{i+1}} \frac{\textsc{service}(X_j, r)}{r^2}dr\right]
\end{aligned} \tag{19}$$

Now, let us make use of the basic fact about $\textsc{service}(X_j, r)$ from [27] which underlies Theorem B.1:

For any job state X_j which is not the empty job,

$$\int_{r=0}^{\infty} \frac{\textsc{service}(X_j, r)}{r^2}dr = 1$$

For the empty job, service is 0.

This provides a loose bound on the integral in (9), which integrates over a smaller interval of ranks. Substituting in this bound, we find that

$$\begin{aligned}
&E\left[\frac{1-B_{r_i}}{1-\rho^A_{r_{i+1}}}\sum_{j=1}^{k-1}\int_{r=r_i}^{r_{i+1}}\frac{\textsc{service}(X_j,r)}{r^2}dr\right]\\
&\le E\left[\frac{1-B_{r_i}}{1-\rho^A_{r_{i+1}}}\min\{N,k-1\}\right]\\
&\le (k-1)E\left[\frac{1-B_{r_i}}{1-\rho^A_{r_{i+1}}}\right]\\
&= (k-1)\frac{1-\rho^A_{r_i}-\rho^R_r}{1-\rho^A_{r_{i+1}}}\\
&\le (k-1)\frac{1-\rho^A_{r_i}}{1-\rho^A_{r_{i+1}}}
\end{aligned}$$

Returning all the way back to the beginning, we find that

$$\int_{r=0}^{\infty}\frac{E[(1-B_r)W_r]}{r^2(1-\rho^A_r)}dr \le (k-1)\sum_{r=0}^{|R|}\frac{1-\rho^A_{r_i}}{1-\rho^A_{r_{i+1}}}$$

To optimize this bound, we need to choose R to minimize this sum. To do so, we set $|R| = \lceil \ln \frac{1}{1-\rho} \rceil$, and choose r_i such that

$$\frac{1-\rho^A_{r_i^+}}{1-\rho^A_{r_{i+1}^-}} \le e$$

for all $i < |R|$. By $^+$ and $^-$, we refer to the left and right limits, thereby handling the possibility that ρ^A_r is discontinuous as a function of r. We therefore find that

$$\int_{r=0}^{\infty}\frac{E[(1-B_r)W_r]}{r^2(1-\rho^A_r)}dr \le e(k-1)\left\lceil \ln\frac{1}{1-\rho}\right\rceil$$

□

Now, it remains to bound the recyclings term in (16). Note that this term is identical to the one in [27], so we can use essentially the same approach - we just disentangle it from the other term. First, we use a basic theorem from [27]:

Lemma B.4 (Lemma 8.2, [27]).

$$\lambda^R_r E_r[S_r(X^R_r)W_r] \le (k-1)r\rho^R_r$$

Now, it remains to bound the recyclings-dependent term, plugged into Theorem B.1.

Lemma B.5.

$$\int_{r=0}^{\infty}\frac{(k-1)r\rho^R_r}{r^2(1-\rho^A_r)}dr \le (k-1)\ln\frac{1}{1-\rho}$$

Proof. First, let us simplify:

$$\int_{r=0}^{\infty}\frac{(k-1)r\rho^R_r}{r^2(1-\rho^A_r)}dr = (k-1)\int_{r=0}^{\infty}\frac{\rho^R_r}{1-\rho^A_r}\frac{1}{r}dr$$

To bound the integrand, we will explicitly consider the Gittins game. Using the definitions of $\text{UNDONE}_A(r)$, and $\text{GAME}_A(r)$ given in Appendix B.2 of [27], we bound as follows:

$$\begin{aligned}\frac{\rho_r^R}{1-\rho_r^A}\frac{1}{r} &\le \frac{\lambda r \text{UNDONE}_A(r)}{1-\lambda(\text{GAME}_A(r)-r\text{UNDONE}_A(r))}\frac{1}{r}\\ &\le \frac{\lambda \text{UNDONE}_A(r)}{1-\lambda\text{GAME}_A(r)}\\ &= \frac{\lambda\frac{d}{dr}\text{GAME}_A(r)}{1-\lambda\text{GAME}_A(r)}\\ &= \frac{d}{dr}\ln\frac{1}{1-\lambda\text{GAME}_A(r)}\end{aligned}$$

Above, we make use of [27, Lemma 5.3].

Integrating over all $r \in [0,\infty)$, we find that

$$\begin{aligned}\int_{r=0}^{\infty}\frac{\rho_r^R}{1-\rho_r^A}\frac{1}{r}dr &\le \left[\ln\frac{1}{1-\lambda\text{GAME}_A(r)}\right]_{r=0}^{\infty}\\ &= \ln\frac{1}{1-\lambda\text{GAME}_A(\infty)} - \ln\frac{1}{1-\lambda\text{GAME}_A(0)}\end{aligned}$$

From the definition of the Gittins game, it is straightforward to prove that $\text{GAME}_A(0)=0$, and that $\text{GAME}_A(\infty)=E[S]$.

As a result,

$$\int_{r=0}^{\infty}\frac{\rho_r^R}{1-\rho_r^A}\frac{1}{r}dr \le \ln\frac{1}{1-\rho}$$

□

Now, we're ready to put it all together. We derive a bound on mean response time:

Theorem 5.1. *In any multiserver-job system in the power-of-two setting the difference in mean response time between ServerFilling-Gittins and Gittins-1 (resource pooled) is at most*

$$E[T^{SFG\text{-}k}] - E[T^{Gittins\text{-}1}] \le \frac{(e+1)(k-1)}{\lambda}\ln\left(\frac{1}{1-\rho}\right)+\frac{e}{\lambda}$$

The same is true of DivisorFilling-Gittins in the divisible setting.

Proof. Combine Theorem B.1 with Theorem B.2, using Lemma B.3 and Lemma B.5 to bound the two terms. □

Note that this bound is in some ways stronger than the bound on Gittins-k given in [27]. Our bound is the first uniform bound on multiserver Gittins, meaning that our bound doesn't depend on S except via $E[S]$, unlike the bound on Gittins-k in [27]. Note also that the M/G/k is a special case of the multiserver-job system when server needs are all 1, and that in this special case, ServerFilling-Gittins specializes to Gittins-k. As a result, Theorem 5.1 is a strict improvement upon the bound given in [27].

Analogous to Theorem 4.2, we use our bound to prove that ServerFilling-Gittins (and DivisorFilling-Gittins) achieve asymptotically optimal mean response time.

Theorem 5.2. *If* $E[S^2(\log S)^+] < \infty$,

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

Note that $E[T^{SRPT\text{-}1}] \leq E[T^{Gittins\text{-}1}]$ by the optimality of SRPT, so $E[T^{Gittins\text{-}1}] = \omega(\frac{1}{1-\rho})$ whenever $E[S^2(\log S)^+] < \infty$, just as $E[T^{SRPT\text{-}1}] = \omega(\frac{1}{1-\rho})$ in this case.

C DIVISORFILLING-SRPT

The DivisorFilling-SRPT policy is a scheduling policy for the divisible server needs setting of the multiserver-job system, where all server needs k_j perfectly divide the total number of servers k.

To implement DivisorFilling-SRPT, we order jobs in increasing order of remaining size r_j, and then apply a recursive procedure to select the jobs to serve, which we will specify in Appendix C.1. DivisorFilling-Gittins is defined identically, replacing increasing remaining size order with increasing rank order.

DivisorFilling-SRPT achieves two key guarantees:

(1) DivisorFilling-SRPT always serves a subset of the k jobs of least remaining size in the system.

(2) If at least k jobs are present, DivisorFilling-SRPT serves jobs with total server need exactly k.

The proof of Item 1 is immediate from the definition of DivisorFilling-SRPT in Appendix C.1. As for Item 2, this result was proven for the DivisorFilling policy [11, Appendix A], which is identical to DivisorFilling-SRPT, except that the jobs are ordered in arrival order, rather than SRPT order. As the proof is significantly involved, we do not reprove it here. An identical proof applies to DivisorFilling-SRPT. See [11, Appendix A] for details.

As a corollary of Items 1 and 2, we can prove the "relevant work efficiency" property for DivisorFilling-SRPT:

Corollary C.1 (Relevant work efficiency). *Under the DivisorFilling-SRPT policy, in the divisible setting, if there are k or more r-relevant jobs in the system, all servers are occupied by r-relevant jobs.*

From Corollary C.1, we can use the same techniques as were used for ServerFilling-SRPT to prove Theorem 4.1 and Theorem 4.2.

C.1 DivisorFilling-SRPT Definition

Order all jobs in the system in order of least remaining size. Let M be the set of k jobs with least remaining size, labeled such that $r_{m_1} \leq r_{m_2} \leq \ldots$, breaking ties arbitrarily.

We now split into three cases:

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

C.1.1 Case 1. If M contains at least $k/6$ jobs with server need 1, we initially parallel the ServerFilling-SRPT policy: we order jobs in M by server need (tiebroken by least remaining size), and place jobs into service in that order. However, because server needs are not powers of two, we may reach a point where no more jobs fit into service, but servers are still unoccupied. In this case, we place jobs from M with server need 1 into service, again tiebroken by least remaining size. We continue doing so until all k servers are full or no more server need 1 jobs remain.

In [11, Appendix A], it is proven that if $|M| \geq k$, at most $k/6$ servers are open after the "ordered by server need" phase is over, so this procedure always fills all k servers.

C.1.2 Case 2. Suppose that k is of the form $2^a 3^b$, and Case C.1.1 does not apply.

We will recurse on one of two subsets of M: the set of jobs with even server need, or the set of jobs of odd server need greater than 1. Note that all jobs in the latter subset have server needs

divisible by 3. We call the former subset M_2 and the latter subset M_3. To decide which subset to recurse on, we compare the values $2|M_2|$ and $3|M_3|$, and recurse on subset whose value is larger. In the case of a tie, we arbitrarily select M_2.

If $2|M_2|$ is larger, we will only serve jobs from among M_2. To decide which jobs to serve, imagine that we combine pairs of servers. Doing so reduces k by a factor of 2, and reduces the server requirement of each job in M_2 by a factor of 2. We now recursively compute which jobs from M_2 the DivisorFilling-SRPT policy would serve in this subproblem, and serve those same jobs. If $3|M_3|$ is larger, we combine triples of servers, and then perform the same recursion.

In [11, Appendix A], it is proven that if $|M| \geq k$, then $\max(2|M_2|, 3|M_3|) \geq k$, so this procedure always fills all k servers.

C.1.3 Case 3. Suppose that k has largest prime factor $p \geq 5$, and that Case C.1.1 does not apply.

Let M_p be the set of jobs in M with server need divisible by p. If $p|M_p| \geq k$, we recurse as in Case C.1.2 by combining groups of p servers.

Otherwise, we will only serve jobs from M whose server need is *not* divisible by p, and also greater than 1. Let M_r be this subset of M. Note that all jobs in M_r have server requirements which are divisors of k/p. We therefore construct a set M' consisting of the k/p elements of M_r with least remaining size. We then apply the DivisorFilling-SRPT procedure to M', setting the total number of servers $k' = k/p$ in the subproblem. We extract the subset of jobs that DivisorFilling-SRPT serves in the subproblem from M_r. We repeat this process by extracting subsets from the remaining jobs in M_r, repeating until we have extracted p subsets from M_r. DivisorFilling-SRPT serves all jobs that were served in any of the p subproblems.

Note that this service is valid, with total server need at most k, because each of the p subproblems have total server need at most k/p.

In [11, Appendix A], it is proven that if $|M| \geq k$, there are at least k/p remaining jobs in M_r in of the p steps, implying that each of the p extracted subsets has server need exactly k/p. As a result, this procedure always fills all k servers.