Balanced Splitting: A Framework for Achieving Zero-wait in the Multiserver-job Model

Jonatha Anselmi and Josu Doncel

Abstract—We present a new framework for designing nonpreemptive and job-size oblivious scheduling policies in the multiserver-job queueing model. The main requirement is to identify a *static and balanced sub-partition* of the server set and ensure that the servers in each set of that sub-partition can only handle jobs of a given *class* and in a first-come first-served order. A job class is determined by the number of servers to which it has exclusive access during its entire execution and the probability distribution of its service time. This approach aims to reduce delays by preventing small jobs from being blocked by larger ones that arrived first, and it is particularly beneficial when the job size variability intra resp. inter classes is small resp. large. In this setting, we propose a new scheduling policy, Balanced-Splitting. In our main results, we provide a sufficient condition for the stability of Balanced-Splitting and show that the resulting queueing probability, i.e., the probability that an arriving job needs to wait for processing upon arrival, vanishes in both the subcritical (the load is kept fixed to a constant less than one) and critical (the load approaches one from below) many-server limiting regimes. Crucial to our analysis is a connection with the M/GI/s/s queue and Erlang's loss formula, which allows our analysis to rely on fundamental results from queueing theory. Numerical simulations show that the proposed policy performs better than several preemptive/nonpreemptive size-aware/oblivious policies in various practical scenarios. This is also confirmed by simulations running on real traces from High Performance Computing (HPC) workloads. The delays induced by Balanced-Splitting are also competitive with those induced by state-of-the-art policies such as First-Fit-SRPT and ServerFilling-SRPT, though our approach has the advantage of not requiring preemption, nor the knowledge of job sizes.

Index Terms—Multi-server jobs, online scheduling, queueing probability, zero-wait, Erlang's loss formula, asymptotic optimality

1 Introduction

The multiserver-job queueing model has recently gained traction in the performance analysis of parallel systems because it captures a key aspect of today's High Performance Computing (HPC) and cloud computing systems [1], [2], [3], [4]. Within this model, each job can occupy multiple servers simultaneously for the entire duration of the job itself. Servers are an abstraction for processing resources: in HPC systems, they refer to CPUs, cores, GPUs or nodes; in cloud systems, they refer to virtual machines or containers; and in serverless computing, they refer to cloud functions, instances or replicas. The number of servers required by each job is referred to as *server need* of that job, and logs of real parallel workloads from production systems reveal that they range from one to a few thousand, depending on the application and on the resource type [5], [6]. In turn, these workloads run on large systems that are composed of up to millions of resources. According to the TOP500 project [7], the United States' Frontier is currently the most powerful supercomputer, reaching 8,699,904 total cores.

Recently, the interest for multiserver jobs has profoundly increased due to the massive use of deep learning technologies and applications, which rely on highly-parallel machine learning jobs like TensorFlow [8] to handle several aspects of our daily life. Beyond the case of machine learning jobs, examples of jobs that typically require a specific number of servers are easily found in linear algebra applications that rely on singular value or matrix decompositions, in image processing scenarios such as neuroscience applications [9], where different servers operate on different sets of pixels, etc. Finally, in [10], [11] the authors show that tasks submitted to Google's Borg Scheduler (which represent jobs executing internet-facing services such as Docs, Sheets, and Gmail, for instance, or jobs from the internal tools and services) require a specific number of server, which can vary by five orders of magnitude across jobs (see also [12]).

In multiserver-job queueing models, jobs wait in a central queue before being processed, they then hold the simultaneous possession of multiple servers according to their server needs, and finally, they leave the system permanently upon service completion. These models differ from standard queueing models where each job can only occupy one server, and they present the technical challenge of designing scheduling policies that can efficiently pack jobs onto servers ensuring full capacity usage whenever the number of jobs exceeds the number of servers. In this case, the corresponding scheduling policy is said *throughput optimal*. However, it may be impossible to design a system that prevents a loss of capacity. For example, consider the case where the number of servers is odd but each job requires an even number of servers.

The fundamental problem is to design scheduling policies that yield high utilization/throughput and/or low delay/makespan. A scheduling policy for the multiserver-job model determines which jobs are processed by which servers at any given time. This problem is nontrivial and referred to in the literature as the "parallel on-line job scheduling problem" [13]. Since these policies may run on massively large computer systems, in practice HPC and cloud platform administrators opt to use simple heuristics,

- *J. Anselmi is with Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP, LIG, 38000 Grenoble, France. E-mail: jonatha.anselmi@inria.fr*
- *J. Doncel is with the University of the Basque Country, UPV/EHU, Barrio sarriena s/n, 48940 Leioa, Spain. E-mail: josu.doncel@ehu.eus*

with low computational complexity.

In the last decades, several works have addressed the problem of designing scheduling policies for multiserver jobs. Most existing studies focus their performance analysis on settings in finite time or with a finite number of jobs. In this paper, we consider an infinite stream of stochastically arriving jobs and investigate performance (stability and mean delays) in a *steady-state* regime. The scientific literature has received relatively little attention in this queueing theoretic setting, which is attributed to the increased mathematical complexity caused by the non-conventional structure of multiserver jobs. However, several new scheduling policies and interesting related results have recently been developed in the literature. Now, we provide an overview.

1.1 Common scheduling policies and positioning

The list of scheduling policies for multiserver jobs investigated in the literature is huge [3], [4], [13], [14] and our goal here is to provide the necessary high-level background highlighting the difference of our work. In general, it is difficult to establish whether a policy is better than another because the answer strongly depends on i) the assumptions made on the underlying architecture and ii) the information available to the scheduler. In this respect, existing scheduling policies may be classified according to whether they require preemption or job-size awareness:

- A scheduling policy is *preemptive* if jobs can be stopped during their execution and resumed afterward at any point in time. While preemption may improve performance theoretically, it has some important limitations. First, as discussed in [2], stopping and resuming jobs come with non-negligible switching costs, and job migration is also involved. Due to analytical intractability, existing queueing models do not usually take these costs into account, which leads to optimistic performance analyses. A further disadvantage of preemption is the starvation of jobs requiring a large number of servers.
- A scheduling policy is *size aware* if at any point in time the scheduler knows the remaining processing time of each job together with its server need. Here, the celebrated example is the Shortest Remaining Processing Time scheduling discipline, which in the single-server job model is known to minimize delays in a strong sense [15]. The downside of this requirement is that in practice such information is rarely available. While users of HPC systems usually submit jobs together with estimations of their running times, it is well known that these are largely imprecise; see [16, Section 1.1]

In turn, existing performance analyses of the multiserver-job queueing model may be classified according to whether they consider a *transient* or *steady-state* regime. The former considers the scheduling of a finite number of jobs, while the latter considers an infinite stream of jobs entering the system at the jump times of an exogenous stochastic point process. In this paper, we are interested in the steady-state regime. In this setting, a primary performance objective concerns throughput optimality, which roughly means that "the system is stable whenever the arrival rate is smaller than the service rate". Then, another important objective concerns the analysis of mean delay, or mean response time. Unfortunately, tractable analytical formulas for mean delays are usually out of reach, even assuming that jobs arrive according to a Poisson process. For this reason, existing approaches investigate dynamics in some limiting regime of practical interest as this often leads to the identification of simple asymptotic formulas that turn out to be extremely accurate in the pre-limit. In the literature, researchers have considered limiting regimes where i) the arrival rate approaches the boundary of the stability region from below while keeping constant the overall number of servers (*heavy traffic* limiting regime) [3], or ii) the traffic demand grows to infinity together with the total number of servers (*many servers* limiting regime) [17], [18]. In this paper, we are interested in the latter approach, as it fully captures the nature of real computer systems, which are composed of thousands, if not millions, of servers [7].

The requirements and features of the most common and newest scheduling policies proposed in the literature are summarized in Table 1; we point the reader to Section 2 for more details about these scheduling policies. In the list, we point out *Balanced Splitting*, a scheduling policy that we propose in this work and discuss below.

1.2 Balanced Splitting

We introduce a new framework, referred to as Balanced Splitting Framework (BSF), for the design of non-preemptive, job-size oblivious scheduling policies in the multiserver-job model. The main idea is to reduce interference between small and large jobs by keeping them separate to some extent. It works as follows. First, divide jobs into *classes*, where a job class is determined by its server need and its service time probability distribution. Then, identify a static *sub*-partition of the server set, say $\mathcal{A} := \cup_i \mathcal{A}_i$, in proportion to the relative demand of each job class i, where static means independently of the job arrival process. Finally, ensure that each subset of servers $\mathcal{A}_i$ can only process jobs of class i and in a first-come first-served order. The servers outside $\mathcal{A}$ are called *helpers* and their set is denoted by $\mathcal{H}$. Within BSF, we propose the following policy:

> **BalancedSplitting**-π: *Upon arrival of a class-i job, send it to servers in $\mathcal{A}_i$ if enough idle servers for immediate processing exist, otherwise send it to the helper set for* potential *processing by the servers in $\mathcal{H}$. In the helper set, process jobs according to π, an auxiliary scheduling policy for multiserver jobs. Upon completion of service of a class-i job in $\mathcal{A}_i$, assign to $\mathcal{A}_i$ the oldest class-i job that is waiting for service in the helper set, provided that it exists.*

While π can be any of the policies discussed above, to preserve a simple structure we assume that it is nonpreemptive, size-oblivious, and only acting on the helpers, i.e., it operates independently of the state of the servers in $\mathcal{A}$.

1.3 Main contribution

Our main objective is to develop a scheduling policy yielding the so-called *zero-wait property*, i.e., a condition where the mean job waiting time vanishes when the system size

	Requirements		Performance	
	Preemptive	Size-aware	Throughput optimality	Many-server delay asymptotics
First-Come First-Served	No	No	Almost* [18]	Yes [18]
Most Servers First [19], [20]	Yes	No	No [2]	N/A
First-Fit Back-Filling	No	No	?	N/A
First-Fit SRPT	Yes	Yes	?	N/A
Max-Weight [2]	Yes	No	Yes	N/A
GreedySRPT [4]	Yes	Yes	?	N/A
ServerFilling [21]	Yes	No	Yes**	N/A
ServerFilling-SRPT [3]	Yes	Yes	Yes**	N/A
ServerFilling-Gittins [3]	Yes	No	Yes**	N/A
BalancedSplitting-π (this paper)	No	No	Almost*	Yes

*: *No* in general and *yes* asymptotically in a many-server limiting regime, under mild conditions.

**: Provided that the overall number of servers and all the server needs are a power of two.

Table 1
Summary of the requirements and performance of the most common and latest scheduling policies.

grows large together with the traffic demand. In the queueing literature, this property has been considered in several works, e.g., [22], [23], [24], while for now it has received less attention in multiserver-job queueing models [18]; see Section 2 for more details. Within our approach, we address this objective by investigating the following question:

> *Within BalancedSplitting-π, under which conditions on the model parameters is it possible to neglect the probability, say $P_{\mathcal{H}}$, that a job is sent to the helper set $\mathcal{H}$ upon its arrival?*

Here, the idea is the following: if $P_{\mathcal{H}}$ is "small", then the impact of the auxiliary policy π is "neglibible" and the overall performance is captured by the dynamics that only occur in the $\mathcal{A}$ system. This implies the zero-wait property because jobs that join the $\mathcal{A}$ system are processed immediately upon arrival.

We answer the question above in the affirmative within two limiting regimes of practical interest that are motivated by the huge size of real systems. Specifically, in Theorems 1 and 2 we show that $P_{\mathcal{H}}$ vanishes in both a *subcritical* and a *critical* many-server limiting regime where also the server needs are allowed to scale to infinity. In the former, the demand and the overall number of servers grow to infinity in proportion while the load ρ is kept fixed to a constant less than one. In the latter, the difference is that at the same time the load ρ approaches one from below. Both regimes have been widely considered in the queueing literature [25], [26], [27], and in our case they are well justified because of the massive size of modern computer systems. The critical many-server regime considered in this paper corresponds to the Halfin-Whitt regime, which has been mainstream in the queueing literature since the seminal work in [26]. This regime aims to balance between efficiency and quality of offered service and is also known as Quality and Efficiency Driven (QED).

At the core of our proofs, there is connection a connection with the M/GI/s/s queue, which is a well-studied object in queueing theory since the pioneering works of Erlang. More specifically, we show that the dynamics induced by BalancedSplitting-π can be bounded by the dynamics of a modified version of BalancedSplitting-π ensuring that the $\mathcal{A}$ system can be decomposed into multiple independent M/GI/s/s queues. Within this decomposition, the $\mathcal{A}$ system becomes tractable from a mathematical point of view and the question above can be essentially rephrased as "*Under which conditions is it possible to make the blocking probability of an M/GI/s/s queue vanish?*". This question has a long history in queueing theory since the work of Erlang (see, e.g., [28]), and here we use existing theorems to establish our results.

Figure 1 plots the simulated mean response time induced by BalancedSplitting-π and other scheduling policies in the critical many-server limiting regime. Here, the server needs grow with rate $f_k = \lfloor (k/32)^{2/3} \rfloor$ where k is the overall number of servers, and the load $\rho = \rho(k)$ satisfies $(1-\rho)\sqrt{k/f_k} \to \theta = 0.7$ as $k \to \infty$. Job sizes are *highly variable* and follow the model of "several small and few large" jobs, which is typical in HPC and cloud systems. The relative demand of large resp. small jobs is $80 \cdot 0.05 f_k = 4 f_k$ resp. $0.95 f_k$; see the figure caption for details. For BalancedSplitting-π, we assume that the auxiliary policy π is First-Come First-Served (FCFS). First, we notice that BalancedSplitting-π outperforms FCFS. It also performs better than ServerFilling, which is preemptive, and provides average response times that are competitive with those achieved by ServerFilling-SRPT and First-Fit

This article has been accepted for publication in IEEE Transactions on Parallel and Distributed Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2024.3493631

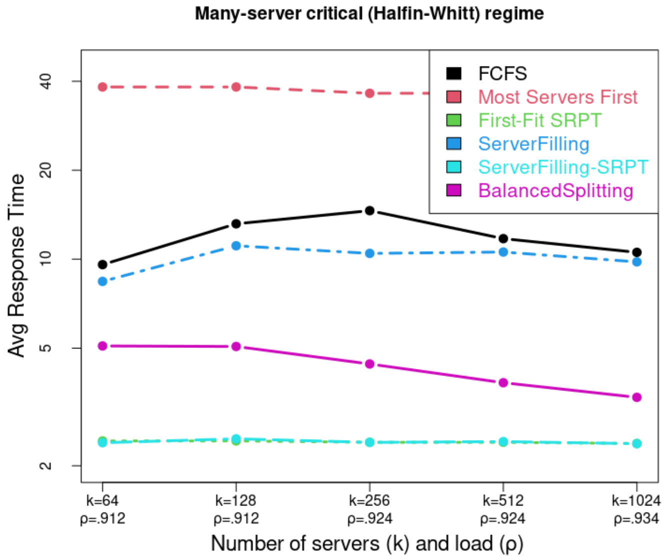


Figure 1. Simulated mean response time in the critically loaded limiting regime (Halfin–Whitt). Small resp. large jobs arrive with probability 0.95 resp. 0.05. Small jobs have server need and mean service time equal to $(f_k, 1)$, where $f_k = \lfloor (k/32)^{2/3} \rfloor$. Large jobs have server need and mean service time equal to ($2f_k$,40), ($4f_k$,20) or ($8f_k$,10), with equal probability. Service times are exponentially distributed and all independent. Arrivals are Poisson. Each simulation run uses 10^6 arrivals. The curves associated to ServerFilling-SRPT and First-Fit SRPT almost overlap.

SRPT, but without incurring the cost of preemption, nor requiring the knowledge of job sizes. In Section 6, we will complement our analysis by presenting an additional set of numerical simulations against both synthetic and real data traces. These will provide further evidence of the benefits of our approach and scheduling policy.

2 Scheduling policies review

This section complements Section 1.1 and reviews in more detail the most common and latest scheduling policies proposed in the literature (see also Table 1). Several of these are considered in this paper for benchmarking purposes.

- FCFS: First-Come First-Served. Jobs are processed in order of arrival if there are enough servers to meet their needs, otherwise they wait in the central queue. A queueing theoretic analysis of the resulting multiserver-job queueing model is challenging even assuming exponentially-distributed job durations and Poisson arrivals [1], [18].
- Most Servers First. At all times, jobs with the highest server need are processed first among all jobs that can be served, with preemption [19], [20]. This policy is also known as Best-Fit [2].
- Least Servers First. At all times, priority is given to jobs with the smallest server needs [21]. Within the same parameter settings used for Figures 1, 2 and 3, our simulations indicate that the response times induced by Least Servers First are huge. In fact, this policy does not seem stable within the considered range of parameters. For this reason, in the remainder we omit in our plots the corresponding numerical results.
- First-Fit Back-Filling. As FCFS, but if there are idle servers and jobs waiting in the queue that can be executed, then the first arrived job that fits the idle servers is processed. Several variants of this "Back-Filling" type of policy are available in the literature [4], [29].
- First-Fit Shortest Remaining Processing Time (SRPT). Serve the jobs with the least remaining processing time, regardless of their server needs. If a job has a higher server need than the remaining number of servers available, skip that job and continue through the list of jobs, placing jobs into service if sufficient servers are available, until all servers are full, or all jobs are assigned. This policy resembles policies like "Smallest Area First" and EASY BackFilling [4].
- MaxWeight. At all times, choose a packing x that maximizes $\sum_n Q_n x_n$ where Q_n is the number of jobs with server need n that are in the system and x_n is the number of jobs with server need n that are served by packing x [2]. While MaxWeight is throughput optimal, it is not practical as it requires the scheduler to solve an NP-hard optimization problem any time a job arrives or departs. There is a myopic version MaxWeight that is nonpreemptive and allocates a new job to a server using current queue length information at job departure times. Under some technical assumptions, it can achieve any arbitrary fraction of the stability region [2].
- ServerFilling. Identify the minimum set of jobs M that collectively require a number of servers at least equal to k, in order of arrival. If such a set is empty, all jobs present in the system are served simultaneously (if any exist), otherwise, the jobs in M are ordered by their server needs, from largest to smallest (with ties broken by arrival order), and placed into service in that order until no more servers are available. The overall number of servers k and server needs do not need to be a power of two. If k is not a power of two, a job j in M may not be packed in even when idle servers exist, and in this case the scheduling is completed with all the jobs arrived later than j blocked and some servers unused. ServerFilling has been proposed in [21].
- ServerFilling-SRPT [3]. Order jobs in increasing order of remaining size r_j, breaking ties arbitrarily, so that $r_{j_1} \leq r_{j_2}, \ldots$ for jobs j_1, j_2, etc. Here, the size of a job is its remaining service time times its service need. Let m^* be the smallest integer such that
$$\sum_{m=1}^{m^*} n_{j_m} \geq k,$$
where n_{j_m} is the server need of job j_m If there is no such index, then serve all jobs in the system simultaneously (if any exists). Otherwise, serve the jobs in $M := \{j_1, j_2, \ldots, j_{m^*}\}$ prioritizing jobs of largest server need until no servers remain or the next job cannot fit, breaking ties by smallest remaining size.
- ServerFilling-Gittins [3]. Identical to ServerFilling-SRPT except that jobs are ordered in increasing order of *Gittins rank*, which for a job with server need i, service time S_i and age (time spent in the system) a is defined as $\inf_{b>a} \mathbb{E}[\min(S_i, b) - a \mid S_i > a] / P(S_i \leq b \mid S_i > a)$. If the service times of jobs with a given server need

are deterministic or if job sizes are known, then ServerFilling-Gittins and ServerFilling-SRPT are equivalent.

Most of the existing policies require preemption. As discussed above, this is costly in practice.

Throughput optimality for the recent ServerFilling-* policies holds if the overall number of servers k and all the server needs are a power of two. Within this assumption, it is possible to perfectly pack jobs into the server set whenever the overall amount of server need exceeds the number of servers. The family of DivisorFilling-* policies discussed in [3] weakens this assumption requiring that all server needs are a divisor of k. Their applicability remains limited because in practice the overall number of servers is hardly a power of two [7]. One of our goals is to develop a scheduling policy that is more flexible in this respect.

2.1 Queueing probability and the zero-wait property

In the queueing literature, many-server limiting regimes have been extensively considered to evaluate the queueing probability, i.e., the probability that an arriving job needs to wait upon arrival. In several cases, this probability converges to zero in the limit, yielding the zero-wait property [25], [26], [27], [28]. In the multiserver-job queueing model however, only a few results are available. In [18], the authors analyze the FCFS scheduling policy under the assumption that service times follow an exponential distribution. While they prove that FCFS is not throughput optimal, the suboptimality gap vanishes in a many-server limiting regime and the zero-wait property is achieved. In this paper, we follow a similar approach to theirs, but with respect to our new policy BalancedSplitting-π.

3 Framework

This section contains our contribution from a modeling and algorithmic point of view. First, it describes the multiserver-job queueing model as in [18]. Then, it defines the proposed Balanced Splitting framework introducing two scheduling policies and related performance indicators.

3.1 Multiserver-job queueing model

We consider a queueing system where an infinite stream of jobs needs to be processed by k parallel servers working with unitary speed. Jobs join the system following a Poisson process with rate λ and can be partitioned into C *classes*. A job is of class i with probability α_i, for all $i = 1, \dots, C$. Class-i jobs require the simultaneous possession of $n_i \leq k$ servers for a random amount of time equal in distribution to the random variable D_i, for all $i = 1, \dots, C$. After processing, they leave the system. In the following, n_i (a constant) and D_i will be respectively referred to as "server need" and "service time" of class-i jobs. The stochastic sequences of job inter-arrivals, service times, and classes are assumed i.i.d. and independent.

Let $d_i := \mathbb{E}[D_i]$. We denote by $\varrho_i := \alpha_i d_i n_i$ the *relative demand* induced by class-i jobs, and let $\varrho := \sum_{i=1}^{C} \varrho_i$. We assume that the *load*

$$\rho := \frac{\lambda}{k} \varrho < 1, \tag{1}$$

which is necessary to ensure stability (positive Harris recurrence) of the underlying Markov process. Note that (1) corresponds to the usual stability condition "the overall arrival rate is less than the overall service rate".

Within the model described above, a *scheduling policy* is a rule that decides which jobs are processed by the servers at any point in time.

3.2 Partitioning the set of servers

For a set S, let $|S|$ denote its cardinality. We let sets $(\mathcal{A}_i)_i$ and $\mathcal{H}$ form a partition of the set of servers $\{1, \dots, k\}$ such that for all i

$$|\mathcal{A}_i| = \left\lfloor \psi \frac{k}{n_i} \frac{\varrho_i}{\varrho} \right\rfloor n_i =: a_i \tag{2a}$$

$$|\mathcal{H}| = k - \sum_{i=1}^{C} a_i \tag{2b}$$

where $\psi = 1$ if $\frac{k}{n_i} \frac{\varrho_i}{\varrho}$ is an integer for all i, and otherwise $\psi = \max\{x \in [0,1] : k - \sum_i \lfloor x \frac{k}{n_i} \frac{\varrho_i}{\varrho} \rfloor n_i \geq \max_i n_i\}$.

The servers in $\mathcal{A}_i$ will only process jobs of class i, while the servers in $\mathcal{H}$ will process jobs of potentially any class and are interpreted as *helpers*. We refer to $\cup_i \mathcal{A}_i$ as the $\mathcal{A}$ system.

The parameter ψ ensures that the helper set contains enough servers to serve jobs of any class, provided that it is non-empty. To minimize the interference between different job classes, and therefore to fully exploit the benefits of our approach, we want the helper set to be small compared to $\mathcal{A}$. To achieve this, it should be clear that ψ should be as close as possible to one. However, the choice $\psi = 1$ may not always be convenient because the resulting number of helpers may be less than some job's server need. Hence, the choice of ψ given above.

In the following, our analysis assumes that $a_i > 0$ for all i; if this is not the case, one can always redirect class-i jobs to the helpers.

Finally, we notice that the helper set is empty only when $\frac{k}{n_i} \frac{\varrho_i}{\varrho}$ is an integer for all i. This condition has "zero measure" because ϱ_i is a real number. To avoid unnecessary and minor complications, e.g., divisions by zero, in the remainder we assume that $\mathcal{H}$ is non-empty ($|\mathcal{H}| \geq \max_i n_i$). If $\mathcal{H}$ would be empty, all of our results remain true trivially.

3.3 Balanced Splitting framework

The Balanced Splitting Framework (BSF) specifies a principle for scheduling policies. Namely, a scheduling policy belongs to BSF if the subset of servers $\mathcal{A}_i$, defined in Section 3.2, process jobs of class i only and according to FCFS. In this manner, job class information is exploited to reduce the interference between jobs with different server need requirements, as their processing is forced to occur on different sets of servers.

Inside BSF, we propose the scheduling policy *Balanced Splitting-π*, denoted by BS-π, defined as follows.

Definition 1. BS-π *is the BSF policy that is based on the following rules:*

1) *Upon the arrival of a class-i job, send it to servers in $\mathcal{A}_i$ if enough idle servers for immediate processing exist, otherwise*

send it to the helper set for potential *processing by the servers in $\mathcal{H}$.*

2) *In the helper set, process jobs according to π, an auxiliary scheduling policy for multiserver jobs. The policy π is nonpreemptive, size-oblivious, and operates independently of the $\mathcal{A}$ system.*
3) *Upon completion of service of a class-i job in $\mathcal{A}_i$, assign to $\mathcal{A}_i$ the oldest class-i job that is waiting for service in the helper set, provided that it exists.*

Several other policies can comply with the BSF principle above. First, π may be preemptive and/or size-oblivious. Then, point (iii) in Definition 1 may be tweaked to assign to the servers in $\mathcal{A}_i$ the class-i job in the helper set with the highest remaining size or with the highest *expected* remaining size, as this would reduce the workload on the helper set the most. We do not consider these size-aware variants because we want to keep the overall structure as simple as possible.

The following remark summarizes the minimal requirements needed by BS-π.

Remark 1. BS-π *is nonpreemptive and size-oblivious. The scheduler only needs to know the class of each arriving job and, to build the $\mathcal{A}$ system, the overall number of servers together with the relative demands ϱ_i, which can be estimated a priori by user-profiling techniques. In particular, it does not need to know the arrival rate of jobs.*

3.4 Problem statement

Given a scheduling policy π, R_π denotes the mean response time (or delay) induced by π, i.e., the mean time spent in the system by a job. In this paper, we are mainly interested in analyzing the mean response time induced by BS-π. This comes with the technical difficulty of choosing the auxiliary policy π, which in principle is "arbitrary" and would require a separate analysis; for π generic, analytical formulas for R_π are not available. To overcome this difficulty, it is then convenient to introduce the steady-state probability that a job, upon arrival, needs to use the servers in the helper set. Let $P_{\mathcal{H}}$ denote such probability. If $P_{\mathcal{H}}$ is "small", then the contribution of π on the mean response time induced by BS-π is "small". This motivates us to investigate the following problem.

Problem 1. *We are interested in understanding under which conditions it is possible to drive $P_{\mathcal{H}}$ to zero.*

4 Main results

This section presents our main results about the performance induced by the proposed policy BS-π. The key observation, which will be used in our proofs, is that a class-i job is processed either by servers in $\mathcal{A}_i$ or by servers in $\mathcal{H}$, disjointedly. This holds true because the number of servers dedicated to class-i jobs is a multiple of n_i, by construction of the set of servers $\mathcal{A}_i$. This property will be exploited to show that the mean number of busy servers in the $\mathcal{A}_i$ system is lower bounded by the mean number of busy servers of an independent M/GI/s/s queue; see Section 5.1. In light of this connection, we will present our results using standard performance indicators that are relevant to this queue, for which we recall some of its main properties before stating our results.

4.1 Preliminaries

The M/GI/s/s queue is a crucial component of queueing theory and the investigation of the *blocking probability*, i.e., the probability that an incoming job finds all servers busy, is undoubtedly one of the most studied subjects in the literature. Assuming that the arrival process is Poisson with rate λ and that the mean service time of each server is d, it is well known that the blocking probability, which we denote by $E_s(\lambda\, d)$, is given by the Erlang's loss formula:

$$E_s(\lambda\, d) = \frac{(\lambda\, d)^s}{s!}\left(\sum_{\ell=0}^{s}\frac{(\lambda\, d)^\ell}{\ell!}\right)^{-1}. \tag{3}$$

This is connected to the mean response time, denoted by R_s, through the relation

$$R_s = d\,(1 - E_s(\lambda\, d)). \tag{4}$$

Also, we recall the following classical result due to Erlang [28].

Lemma 1. *Suppose that the arrival rate λ and the number of servers s in an M/GI/s/s queue grow to infinity while the load (or utilization) $\rho = \lambda d/s$ approaches one and $(1-\rho)\sqrt{s} \to \theta$ where θ is a fixed constant. Then,*

$$\lim_{s\to\infty}\sqrt{s}E_s(\lambda d) = \frac{\phi(\theta)}{\Phi(\theta)},$$

where $\Phi(x)$ and $\phi(x)$ denote the standard normal cumulative distribution function (CDF) and density, respectively.

The scaling in Lemma 1 satisfies the conditions that define the Halfin-Whitt limiting regime [26].

4.2 Stability

We say that the scheduling policy π is *throughput optimal* if $R_\pi < \infty$ whenever the load ρ is less than one.

Our first result provides a sufficient condition for stability; see Section 5 for a proof.

Proposition 1. *Assume that π is throughput optimal. If*

$$\frac{\lambda}{|\mathcal{H}|}\sum_{i=1}^{C}\varrho_i E_{s_i}(\lambda\alpha_i d_i) < 1, \tag{5}$$

then $R_{\text{BS-}\pi} < \infty$.

It is not clear whether BS-π is throughput optimal (when π is throughput optimal on the helper set). The problem is that service capacity can be lost as follows. Suppose that $(x_1,\dots,x_I,h_1,\dots,h_C)$ is a "state" of the system where x_i resp. h_i denotes the number of class-i jobs in $\mathcal{A}_i$ resp. in $\mathcal{H}$. For simplicity, suppose also that there are only two classes, i.e., $C = 2$, and that $n_1 < n_2$. Then, a loss of capacity occurs in state $(\frac{a_1}{n_1}, 0, h_1, 0)$ with $h_1 > \lfloor\frac{|\mathcal{H}|}{n_1}\rfloor$, as one class-1 job that is waiting for service in the helper set may be processed by the servers in $\mathcal{A}_2$. This does not occur due to the static server set partitioning of the proposed approach, which forces jobs of different classes not to interfere with each other.

In the following, we show that (5) is eventually true if the number of servers is large enough. In other words, this means that BS-π is throughput optimal *asymptotically*.

4.3 Subcritically loaded many-server regime

Now, we consider a limiting regime where the demand $\lambda\varrho$ and the overall number of servers k grow to infinity in proportion while the load ρ is kept fixed to a constant less than one. In the multiserver-job model, there are several ways to accomplish this as both the arrival rate λ and needs n_i, for all i, can grow with k. To identify a proper scaling of the system parameters, we consider a sequence of models indexed by k and add the superindex (k) to refer to the k-th system.

First, we introduce the sequence $(f_k)_k$ such that

$$f_k = o(k), \quad f_k \in \mathbb{N}. \tag{6}$$

Then, we let $k \to \infty$ with the following scaling:

$$\lambda^{(k)} = \lambda \frac{k}{f_k} \tag{7a}$$

$$n_i^{(k)} = n_i\, f_k \tag{7b}$$

$$\alpha_i^{(k)} = \alpha_i, \quad D_i^{(k)} = D_i, \tag{7c}$$

which ensures that the network load $\rho^{(k)}$ remains constant to $\rho = \lambda\varrho < 1$. This subcritical regime has been widely considered in the queueing literature; e.g., [25].

Within this regime, the next result shows that the probability that an arriving job is routed to the helper set, $P_{\mathcal{H}}^{(k)}$, converges to zero; see Section 5 for a proof.

Theorem 1. *Let* (6) *and* (7) *hold. As* $k \to \infty$, $P_{\mathcal{H}}^{(k)} \to 0$ *and* $R_{\text{BS-}\pi}^{(k)} \to \sum_i \alpha_i d_i$.

4.4 Critically loaded many-server regime

Proceeding in a similar way as above, we now consider a limiting regime where the demand $\lambda\varrho$ and the overall number of servers k grow to infinity in proportion but at the same time, the load ρ approaches one from below. Here, let the sequence $(f_k)_k$ satisfy again the conditions in (6) and consider the following scaling:

$$\lambda^{(k)} \to \infty \tag{8a}$$

$$\left(1 - \rho^{(k)}\right)\sqrt{\frac{k}{f_k}} \to \theta, \quad \theta > 0 \tag{8b}$$

$$n_i^{(k)} = n_i\, f_k \tag{8c}$$

$$\alpha_i^{(k)} = \alpha_i, \quad D_i^{(k)} = D_i. \tag{8d}$$

This scaling satisfies the conditions that define the Halfin-Whitt regime [26]. Within this scaling and since $f_k = o(k)$, the following result implies that the probability that a job is routed to the helper set $P_{\mathcal{H}}$ vanishes as $k \to \infty$, and it also provides a convergence speed; see Section 5 for a proof, which is based on Lemma 1.

Theorem 2. *Let* (6) *and* (8) *hold. As* $k \to \infty$,

$$\lim_{k\to\infty} \sqrt{\frac{k}{f_k}}\, P_{\mathcal{H}}^{(k)} \leq \theta \sum_{i=1}^{C} \frac{\alpha_i}{\theta_i} \frac{\phi(\theta_i)}{\Phi(\theta_i)}, \quad \theta_i := \theta \sqrt{\frac{\varrho_i}{n_i \varrho}} \tag{9}$$

where $\Phi(x)$ *and* $\phi(x)$ *denote the standard normal cumulative distribution function (CDF) and density, respectively.*

5 Proofs of our main results

In this section, we provide proofs for our main results along with the intuition behind our strategy. Roughly speaking, the general strategy for proving our results is as follows. Since the dynamics underlying BS-π are presumably intractable to investigate due to the complex interactions between the $\mathcal{A}$ and $\mathcal{H}$ systems, we first define a *modified* version of BS-π where the $\mathcal{A}$ and $\mathcal{H}$ systems are independent. Here, we show that the $\mathcal{A}$ system behaves exactly like a set of parallel M/GI/s/s queues. Then, we connect this modified version to BS-π by showing that it provides bounds on the performance metrics of interest. Finally, we rely on these bounds to develop our asymptotic results.

5.1 Modified policy and connection with the M/GI/s/s queue

Let us define the following modified version of BS-π, denoted by ModifiedBS-π.

Definition 2. ModifiedBS-π *is the BSF policy where an arriving job of class i is sent to $\mathcal{A}_i$ if $\mathcal{A}_i$ contains enough idle servers for immediate processing, otherwise it is* irrevocably *sent to the helper set, for all i. The helper set processes jobs according to an auxiliary scheduling policy π for multiserver jobs that operates independently of the state of the $\mathcal{A}$ system.*

Note that ModifiedBS-π is a scheduling policy that satisfies the BSF principle. The main difference with respect to BS-π is that jobs that upon arrival are sent to the helper set will be certainly processed by the helpers, while within BS-π they may be reassigned to the $\mathcal{A}$ system. For this reason, one expects that it performs worse than BS-π. From a mathematical point of view, the advantage of ModifiedBS-π over BS-π is that the dynamics of the $\mathcal{A}$ system are autonomous and therefore they can be analyzed independently of the $\mathcal{H}$ system – this is not true within BS-π because arrivals to the $\mathcal{A}$ system depend on the state of the helper set.

At this point, the key observation for the dynamics induced by ModifiedBS-π is that a class-i job is processed either by servers in $\mathcal{A}_i$ or by servers in $\mathcal{H}$. In other words, it is not possible that a job is processed by servers in $\mathcal{H}$, preempted and then processed by servers in $\mathcal{A}_i$. In addition, the decision whether a job is assigned to the $\mathcal{A}$ or $\mathcal{H}$ system is irrevocable and made upon job arrival. In view of this, the system enjoys the following property.

Property 1. *Within* ModifiedBS-π*, the process of the number of servers occupied in each $\mathcal{A}_i$ subsystem divided by n_i is the process of the number in an M/GI/s_i/s_i queue where $s_i = \lfloor \psi \frac{k\varrho_i}{n_i\varrho} \rfloor$, the arrival rate is $\lambda\alpha_i$ and service times are distributed as D_i.*

Now, it remains to establish the connection between the performance induced by BS-π and ModifiedBS-π. This is stated in the following proposition.

Proposition 2. *Let $\lambda_i^{\text{BS-}\pi}$ and $\lambda_i^{\text{ModifiedBS-}\pi}$ denote the arrival rates of class-i jobs at the $\mathcal{A}$ system induced by* BS-π *and* ModifiedBS-π*, respectively. Then,*

$$\lambda_i^{\text{BS-}\pi} \geq \lambda_i^{\text{ModifiedBS-}\pi}. \tag{10}$$

This article has been accepted for publication in IEEE Transactions on Parallel and Distributed Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2024.3493631

Proof. Within ModifiedBS-π, the arrival rate of class-i jobs at the $\mathcal{A}$ system is $\lambda\alpha_i$ minus the rate of blocked jobs of the M/GI/s_i/s_i queue defined in Property 1. Therefore,

$$\lambda_i^{\text{ModifiedBS-}\pi} = (1 - E_{s_i}(\lambda\alpha_i d_i))\lambda\alpha_i. \tag{11}$$

For $\lambda_i^{\text{BS-}\pi}$, we notice that $E_{s_i}(\lambda\alpha_i d_i)$ is higher than or equal to the probability that an incoming job will be processed by the helper set as BS-π may redirect that job to n_i servers in $\mathcal{A}_i$. □

The following corollary of Proposition 2 is then straightforward.

Corollary 1. *Let $P_{\mathcal{H}}^{\text{ModifiedBS-}\pi}$ denote the probability that a job, upon arrival and within* ModifiedBS-π*, needs to use the servers in the helper set. Then,*

$$P_{\mathcal{H}} \le P_{\mathcal{H}}^{\text{ModifiedBS-}\pi}. \tag{12}$$

5.2 Proof of Proposition 1

In view of Property 1, the arrival rate of class-i jobs at the $\mathcal{H}$ system induced by ModifiedBS-π corresponds to the rate of dropped jobs of the M/GI/s_i/s_i queue in Property 1. Therefore, the load induced by ModifiedBS-π on the $\mathcal{H}$ system $\rho_{\mathcal{H}}^{\text{ModifiedBS-}\pi}$ is given by

$$\rho_{\mathcal{H}}^{\text{ModifiedBS-}\pi} = \frac{\lambda}{|\mathcal{H}|}\sum_{i=1}^{C} \varrho_i E_{s_i}(\lambda\alpha_i d_i).$$

Finally, Proposition 2 ensures that the load at the helper set induced by BS-π is smaller than or equal to $\rho_{\mathcal{H}}^{\text{ModifiedBS-}\pi}$.

5.3 Proof of Theorem 1

For the M/GI/$s_i^{(k)}$/$s_i^{(k)}$ queue in Property 1, the offered load $\lambda\frac{k}{f_k}\alpha_i d_i$ and the number of servers $s_i^{(k)} = \lfloor\frac{k\varrho_i}{n_i f_k \varrho}\rfloor$ both grow to infinity with the same rate, as $k \to \infty$, and their ratio converges to $\lambda\varrho$. Within this regime, it is known that the loss probability converges to zero, i.e.,

$$E_{s_i^{(k)}}\left(\lambda\frac{k}{f_k}\alpha_i d_i\right) \to 0$$

provided that $\lambda\varrho < 1$, which is the case in our setting. Now, for the probability that in the k-th system an arriving job is routed to the helper set, $P_{\mathcal{H}}^{(k)}$, Property 1 and Corollary 1 imply that $P_{\mathcal{H}}^{(k)} \le P_{\mathcal{H}}^{(k),\text{ModifiedBS-}\pi} = \sum_{i=1}^{C}\alpha_i E_{s_i^{(k)}}(\lambda\alpha_i d_i k/f_k)$. Therefore, $P_{\mathcal{H}}^{(k)} \to 0$ and $R_{\text{BS-}\pi} \to \sum_i \alpha_i d_i$ follows by (4).

5.4 Proof of Theorem 2

Recall that the M/GI/$s_i^{(k)}$/$s_i^{(k)}$ queue in Property 1 has arrival rate $\lambda^{(k)}\alpha_i$, service times distributed as D_i and $s_i^{(k)} = \lfloor\psi^{(k)}\frac{k\varrho_i}{n_i f_k \varrho}\rfloor$. Thus, its load is

$$\rho_i^{(k)} := \lambda^{(k)}\alpha_i d_i / s_i^{(k)}.$$

Within the scaling (8), the following proposition states that also each M/GI/s_i/s_i queue in Property 1 scales in the Halfin-Whitt regime.

Proposition 3. *Let (6) and (8) hold. As $k \to \infty$, $s_i^{(k)} \to \infty$, $\psi^{(k)} \to 1$ and*

$$\left(1 - \rho_i^{(k)}\right)\sqrt{s_i^{(k)}} \to \theta_i. \tag{13}$$

Proof. Since $f_k = o(k)$ and $\psi^{(k)} \le 1$, $s_i^{(k)} = \left\lfloor\psi^{(k)}\frac{k}{n_i f_k}\frac{\varrho_i}{\varrho}\right\rfloor \to \infty$. For $x \in [0,1]$, $k - \sum_i\lfloor x\frac{k}{n_i^{(k)}}\frac{\varrho_{i,j}}{\varrho}\rfloor n_i^{(k)} \ge \max_i n_i^{(k)}$ if and only if

$$\frac{k}{f_k} - \sum_{i=1}^{C}\left\lfloor x\frac{k}{n_i f_k}\frac{\varrho_{i,j}}{\varrho}\right\rfloor n_i \ge \max_i n_i. \tag{14}$$

The RHS is constant in k while the LHS grows to infinity sublinearly in k. So, (14) holds true for all k large enough. Using the definition of ψ in Section 3.3, the floor function $\lfloor\cdot\rfloor$ gives

$$\psi^{(k)} = 1, \quad \forall k \text{ large enough.} \tag{15}$$

It remains to prove (13). We obtain

$$\begin{aligned}
1 \ge \lim_{k\to\infty}\frac{1-\rho_i^{(k)}}{1-\rho^{(k)}} &= \lim_{k\to\infty}\frac{1 - \lambda^{(k)}\alpha_i d_i\lfloor\psi^{(k)}\frac{k\varrho_i}{n_i f_k \varrho}\rfloor^{-1}}{1-\rho^{(k)}}\\
&\ge \lim_{k\to\infty}\left(1 - \frac{\lambda^{(k)}\alpha_i d_i}{\psi^{(k)}\frac{k\varrho_i}{n_i f_k \varrho} - 1}\right)\frac{1}{1-\rho^{(k)}}\\
&= \lim_{k\to\infty}\frac{\psi^{(k)} - \rho^{(k)}}{1-\rho^{(k)}} = 1
\end{aligned}$$

where the last equality follows by (15). Therefore,

$$\begin{aligned}
\lim_{k\to\infty}\left(1-\rho_i^{(k)}\right)\sqrt{s_i^{(k)}} &= \lim_{k\to\infty}\left(1-\rho^{(k)}\right)\sqrt{\left\lfloor\psi^{(k)}\frac{k\varrho_i}{n_i f_k \varrho}\right\rfloor}\\
&= \lim_{k\to\infty}\left(1-\rho^{(k)}\right)\sqrt{\frac{k}{f_k}}\sqrt{\frac{\varrho_i}{n_i\varrho}}\\
&= \theta\sqrt{\frac{\varrho_i}{n_i\varrho}} = \theta_i
\end{aligned}$$

which proves (13). □

Since each M/GI/$s_i^{(k)}$/$s_i^{(k)}$ queue scales in the Halfin-Whitt regime, we can apply Lemma 1 to our case to obtain that

$$\begin{aligned}
\frac{\phi(\theta_i)}{\Phi(\theta_i)} &= \lim_{k\to\infty}\sqrt{s_i^{(k)}}E_{s_i^{(k)}}(\lambda^{(k)}\alpha_i d_i)\\
&= \sqrt{\frac{\varrho_i}{n_i\varrho}}\lim_{k\to\infty}\sqrt{\frac{k}{f_k}}E_{s_i^{(k)}}(\lambda^{(k)}\alpha_i d_i).
\end{aligned}$$

In view of Property 1, the probability that an arriving job is routed to the helper set within ModifiedBS-π is given by

$$P_{\mathcal{H}}^{(k),\text{ModifiedBS-}\pi} = \sum_{i=1}^{C}\alpha_i E_{s_i}(\lambda\alpha_i d_i). \tag{16}$$

Substituting in (16), we obtain

$$\begin{aligned}
\lim_{k\to\infty}\sqrt{\frac{k}{f_k}}P_{\mathcal{H}}^{(k)} &= \sum_{i=1}^{C}\alpha_i\lim_{k\to\infty}\sqrt{\frac{k}{f_k}}E_{s_i}(\lambda^{(k)}\alpha_i d_i)\\
&= \theta\sum_{i=1}^{C}\frac{\alpha_i}{\theta_i}\frac{\phi(\theta_i)}{\Phi(\theta_i)}.
\end{aligned}$$

This article has been accepted for publication in IEEE Transactions on Parallel and Distributed Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2024.3493631

Finally, the result follows by Corollary 1.

6 Empirical comparison

We present the results of a numerical study that compares the simulated mean response times induced by our policy BalancedSplitting (BS-π, Definition 1) to that of other state-of-the-art scheduling policies discussed in Section 2. In our test bench, we test with respect to different traffic conditions and real data traces. We consider scenarios where, roughly speaking, jobs are either large or small, with respectively low and high probability of arriving, as this pattern is ubiquitous in real applications; see, e.g., [16, Section 2.1].

In our simulation setup, the auxiliary policy π used for BS-π is First-Come First-Served (FCFS), and the following consideration has been taken into account.

Remark 2. *We have always chosen the overall number of servers and the server needs to be all powers of two because the ServerFilling-* policies require this assumption for throughput optimality. This choice has been made to better stress the quality of the proposed policies as in this case ServerFilling-SRPT is also optimal for minimizing the mean response time in heavy traffic [3].*

6.1 Heavy-traffic and subcritical limiting regime

In Figure 1, we have presented simulation results related to the critically loaded limiting regime. Here, we complement those results by considering the subcritical and heavy-traffic regimes. In the latter, we recall that the load ρ approaches one from below but the overall number of servers is constant. The results are illustrated in Figure 2, where we have used the same job classes, service time distributions and server needs used for Figure 1. Also, each simulation run is based on 10^6 arrivals.

In the heavy traffic scenario (Figure 2.a), BS-π performs almost optimally up until moderate loads $\rho \leq 0.9$, that is up until the point where the helper set is slightly used. For higher loads, the helper set utilization becomes non-negligible and the intrinsic inefficiencies of FCFS, which is not throughput-optimal, start deteriorating performance, though the resulting response times are significantly smaller than the ones induced by FCFS, Most-Servers-First and ServerFilling.

In the subcritical scenario (Figure 2.b), BS-π provides again excellent performance, with average response times that are close to those achieved by ServerFilling-SRPT and First-Fit SRPT, which are preemptive and size-aware.

6.2 Real HPC workloads

We now assess the performance of the proposed policies against real data traces from the Parallel Workloads Archive [5], [6]. We consider the “cleaned version” of two datasets commonly used for HPC system benchmarks: namely, the SDSC (San Diego Supercomputer Center) SP2 log [30] and the KIT (Karlsruhe Institute of Technology) FH2 log [31]. For each job accounted for in this dataset, we follow the Standard Workload Format (SWF) [32] to extract arrival time (Submit Time), service time (Run Time) and server need (Number of Allocated Processors). Within these datasets, most jobs require a number of cores that is a power of two; for SDSC SP2, these jobs account for 84.4% of all jobs. Therefore, they are ideal to stress the performance of the proposed policy BS-π because within this setting, ServerFilling-SRPT minimizes the mean response time in heavy traffic [3]. To stress the performance of our algorithm even further, we remove from our analysis those jobs whose server needs is not a power of two. To limit the cost of simulations, we consider jobs with a server need no larger than 64. In our framework, this gives two models with $C = 7$ classes for both datasets, whose parameters are given in Tables 2 and 3. Then, Figure 3 plots the simulated response times within both datasets and for $k = 512$ and $k = 1024$ servers. As expected ServerFilling-SRPT and First-Fit SRPT always provide the best delays. Also, all plots show that

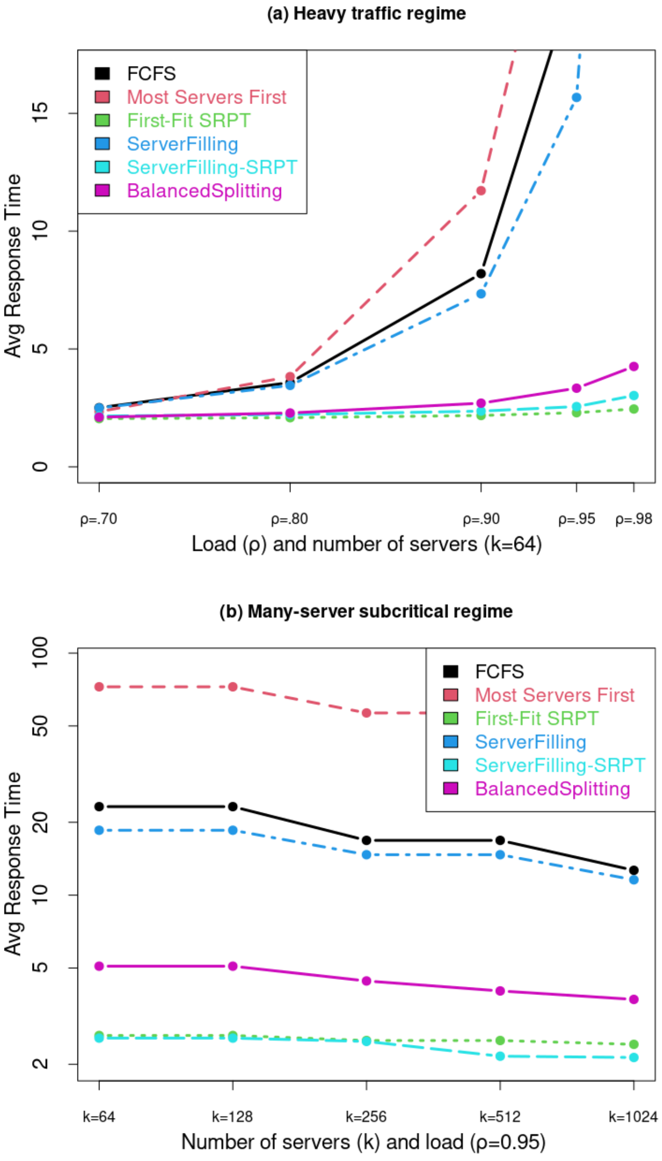


Figure 2. Simulated mean response time in the heavy-traffic (a) and subcritical (b) regimes. Small resp. large jobs arrive with probability 0.95 resp. 0.05. Server needs and mean service times are as in Figure 1. Arrivals are Poisson.

$\mathbb{E}[D_i]$	$\mathrm{std}(D_i)$	n_i	α_i
10519.71	18267.03	1	0.2321
1436.82	6250.19	2	0.1496
5643.69	18123.7	4	0.1624
9248.53	18468.51	8	0.1652
10601.46	17050.63	16	0.156
12139.59	22654.86	32	0.0807
8302.33	19074.81	64	0.054

Table 2
Model parameters extracted from the SDSC SP2 dataset.

$\mathbb{E}[D_i]$	$\mathrm{std}(D_i)$	n_i	α_i
1845.19	11440.31	1	0.7851
1470.13	5237.83	2	0.018
11169.87	38631.83	4	0.0406
3167.33	19727.29	8	0.0137
5706.45	17212.04	16	0.0539
60673.08	92531.56	32	0.0493
61343.42	106094.97	64	0.0393

Table 3
Model parameters extracted from the KIT FH2 dataset.

the proposed policy BalanceSplitting provides significantly better results than ServerFilling, which is preemptive, and FCFS, for almost all loads. We ought this gain to the large variability of service times across job classes and that our framework can reduce the interference between large and small jobs. In fact, especially in Table 3 we observe that the jobs that require 32 or 64 servers have huge service time requirements (relative to the other jobs), and within BS-π, their execution does not interfere with the execution of smaller jobs.

7 Conclusion

We have proposed Balanced-Splitting, a nonpreemptive and job-size oblivious scheduling policy for multiserver jobs that operate on a static and balanced partitioning of the server set. Our main results show that the proposed policy has the zero-wait property asymptotically. More precisely, the probability that a randomly arriving job needs to wait for service vanishes when the system size grows to infinity. Numerical simulations have shown that Balanced-Splitting provides much better delay performance than other common scheduling policies such as FCFS. Also, its average response time is competitive with the one induced by state-of-the-art preemptive and size-aware policies.

We have assumed that the static and balanced partitioning of the server set only depends on the class of each job. In our work, the class of a job is essentially determined by its server need, which is reasonable because this information is available precisely. To define the server set partitioning, we have assumed the per-class knowledge of the probability distribution of service times, which in practice can be obtained by user profiling techniques. We believe that a better partitioning of the server set may be obtained in a size-aware setting. Here, a class of a job would be determined by its server need *and* whether its service time belongs to a specified interval. Of course, the scheduler needs to know the size of each incoming job to properly assign to the proper subset of servers and, in contrast to the approach followed in this work, this would make any policy size-ware. On the other hand, we believe that a balanced size-aware partitioning yields better delay performance. We leave this question as future research.

Acknowledgment

The authors would like to thank Danilo Carastan-Santos and Isaac Grosof for their suggestions and remarks. Thanks are also given to the anonymous reviewers.

References

[1] M. Harchol-Balter, "The multiserver job queueing model," *Queueing Syst. Theory Appl.*, vol. 100, no. 3–4, p. 201–203, apr 2022. [Online]. Available: https://doi.org/10.1007/s11134-022-09762-x

[2] S. Maguluri, R. Srikant, and L. Ying, "Stochastic models of load balancing and scheduling in cloud computing clusters," 2012, pp. 702–710, iEEE Conference on Computer Communications, INFOCOM 2012.

[3] I. Grosof, Z. Scully, M. Harchol-Balter, and A. Scheller-Wolf, "Optimal scheduling in the multiserver-job model under heavy traffic," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 6, no. 3, dec 2022. [Online]. Available: https://doi.org/10.1145/3570612

[4] D. Carastan-Santos, R. Y. De Camargo, D. Trystram, and S. Zrigui, "One can only gain by replacing EASY Backfilling: A simple scheduling policies case study," in *2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, 2019, pp. 1–10.

[5] D. G. Feitelson, D. Tsafrir, and D. Krakov, "Experience with using the parallel workloads archive," *Journal of Parallel and Distributed Computing*, vol. 74, no. 10, pp. 2967–2982, 2014.

[6] "Logs of real parallel workloads from production systems," http://www.cs.huji.ac.il/labs/parallel/workload/logs.html, accessed: 2024-04-14.

[7] "The TOP500 project," https://www.top500.org, accessed: 2024-04-14.

[8] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. A. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016, Savannah, GA, USA, November 2-4, 2016*, K. Keeton and T. Roscoe, Eds. USENIX Association, 2016, pp. 265–283. [Online]. Available: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi

[9] J. Dongarra, B. Tourancheau, A. Gainaru, H. Sun, G. Aupy, Y. Huo, B. A. Landman, and P. Raghavan, "On-the-fly scheduling versus reservation-based scheduling for unpredictable workflows," *Int. J. High Perform. Comput. Appl.*, vol. 33, no. 6, p. 1140–1158, nov 2019. [Online]. Available: https://doi.org/10.1177/1094342019841681

[10] M. Tirmazi, A. Barker, N. Deng, M. E. Haque, Z. G. Qin, S. Hand, M. Harchol-Balter, and J. Wilkes, "Borg: the next generation," in *Proceedings of the fifteenth European conference on computer systems*, 2020, pp. 1–14.

[11] J. Wilkes, "Google cluster-usage traces v3." http://github.com/google/cluster-data, 2019.

[12] A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, "Large-scale cluster management at google with borg," in *Proceedings of the tenth european conference on computer systems*, 2015, pp. 1–17.

[13] P. Brucker, *Scheduling Algorithms*, 5th ed. Springer Publishing Company, Incorporated, 2010.

[14] J. P. Jones and B. Nitzberg, "Scheduling for parallel supercomputing: A historical perspective of achievable utilization," in *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 1–16.

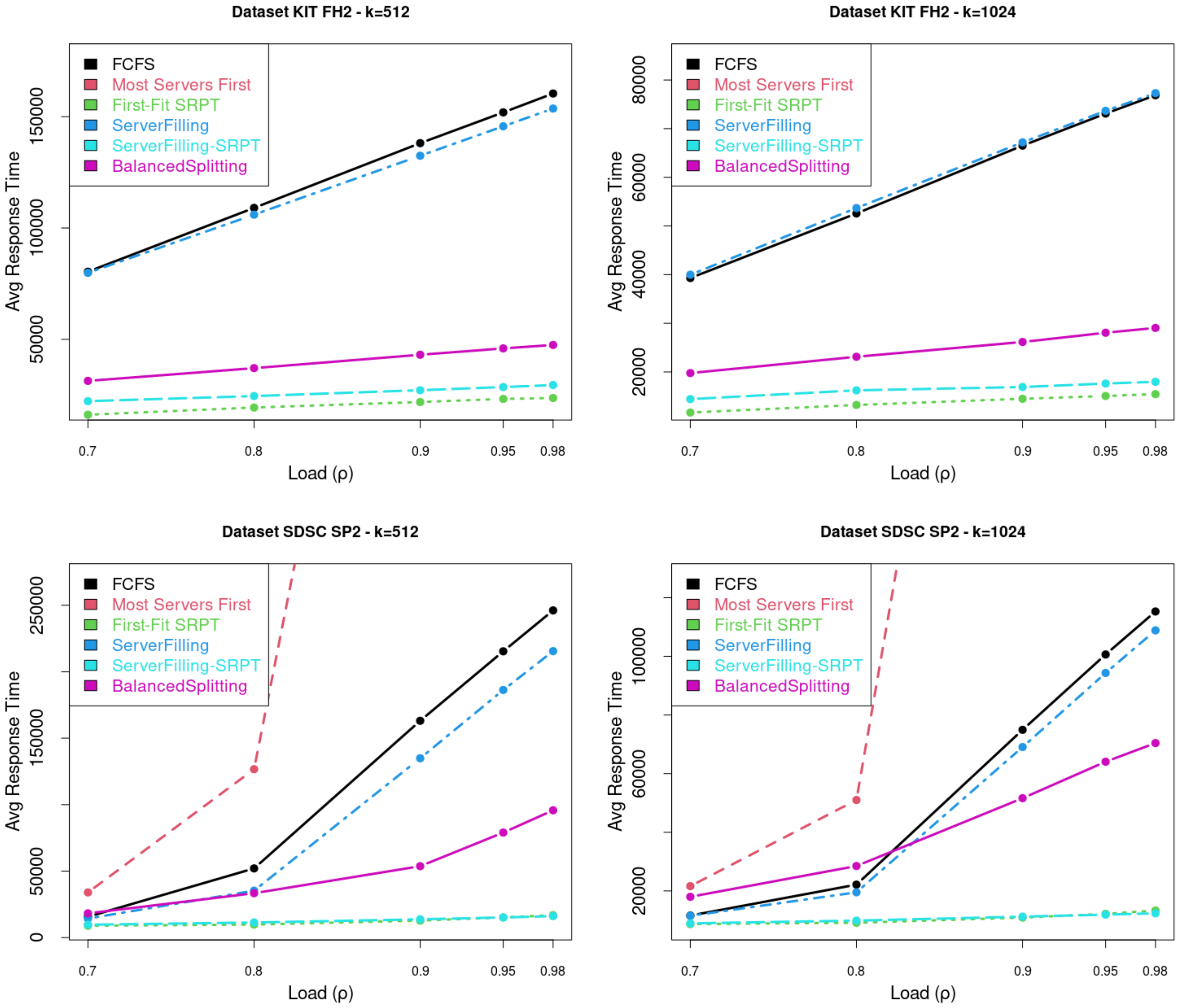


Figure 3. Simulated response times within the KIT FH2 and SDSC SP2 datasets by varying the network load and with respect to $k = 512$ (left) and $k = 1024$ (right) overall servers. The auxiliary policy π used within BalanceSplitting $(\text{BS-}\pi)$ is FCFS.

[15] L. Schrage, "A proof of the optimality of the shortest remaining processing time discipline," *Operations Research*, vol. 16, no. 3, pp. 687–690, 1968. [Online]. Available: http://www.jstor.org/stable/168596

[16] J. Anselmi and J. Doncel, "Load balancing with job-size testing: Performance improvement or degradation?" *ACM Trans. Model. Perform. Eval. Comput. Syst.*, mar 2024, just Accepted. [Online]. Available: https://doi.org/10.1145/3651154

[17] B. Zhang and B. Zwart, "Steady-state analysis for multiserver queues under size interval task assignment in the quality-driven regime," *Mathematics of Operations Research*, vol. 38, no. 3, pp. 504–525, 2013. [Online]. Available: https://doi.org/10.1287/moor.1120.0571

[18] W. Wang, Q. Xie, and M. Harchol-Balter, "Zero queueing for multiserver jobs," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 5, no. 1, feb 2021.

[19] B. Speitkamp and M. Bichler, "A mathematical programming approach for server consolidation problems in virtualized data centers," *IEEE Transactions on Services Computing*, vol. 3, no. 4, pp. 266–278, 2010.

[20] A. Beloglazov and R. Buyya, "Energy efficient allocation of virtual machines in cloud data centers," in *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, 2010, pp. 577–578.

[21] I. Grosof, M. Harchol-Balter, and A. Scheller-Wolf, "WCFS: a new framework for analyzing multiserver systems," *Queueing Syst. Theory Appl.*, vol. 102, no. 1–2, p. 143–174, oct 2022. [Online]. Available: https://doi.org/10.1007/s11134-022-09848-6

[22] A. L. Stolyar, "Pull-based load distribution in large-scale heterogeneous service systems," *Queueing Syst. Theory Appl.*, vol. 80, no. 4, pp. 341–361, Aug. 2015. [Online]. Available: http://dx.doi.org/10.1007/s11134-015-9448-8

[23] D. Gamarnik, J. N. Tsitsiklis, and M. Zubeldia, "Delay, memory, and messaging tradeoffs in distributed service systems," in *Proceedings of the 2016 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science*, ser. SIGMETRICS '16. New York, NY, USA: ACM, 2016, pp. 1–12.

[24] J. Anselmi, "Combining size-based load balancing with round-robin for scalable low latency," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 4, pp. 886–896, 2020.

[25] F. P. Kelly, "Loss Networks," *The Annals of Applied Probability*, vol. 1, no. 3, pp. 319 – 378, 1991. [Online]. Available:

https://doi.org/10.1214/aoap/1177005872

[26] S. Halfin and W. Whitt, "Heavy-traffic limits for queues with many exponential servers," *Operations Research*, vol. 29, no. 3, pp. 567–588, 1981.

[27] P. J. Hunt and F. P. Kelly, "On critically loaded loss networks," vol. 21, no. 4, pp. 831–841, Dec. 1989. [Online]. Available: http://www.jstor.org/stable/1427769

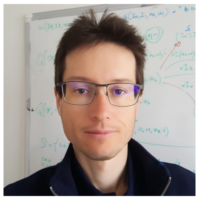
[28] A. J. E. M. Janssen, J. S. H. V. Leeuwaarden, and B. Zwart, "Gaussian expansions and bounds for the Poisson distribution applied to the erlang b formula," *Advances in Applied Probability*, vol. 40, no. 1, pp. 122–143, 2008. [Online]. Available: http://www.jstor.org/stable/20443573

[29] S. Srinivasan, R. Kettimuthu, V. Subramani, and P. Sadayappan, "Characterization of backfilling strategies for parallel job scheduling," in *Proceedings. International Conference on Parallel Processing Workshop*, 2002, pp. 514–519.

[30] "The San Diego Supercomputer Center (SDSC) SP2 log," https://www.cs.huji.ac.il/labs/parallel/workload/l_sdsc_sp2/index.html, accessed: 2024-04-14.

[31] "The KIT ForHLR II log," https://www.cs.huji.ac.il/labs/parallel/workload/l_kit_fh2/index.html, accessed: 2024-04-14.

[32] "The standard workload format," https://www.cs.huji.ac.il/labs/parallel/workload/swf.html, accessed: 2024-04-14.



Jonatha Anselmi is a tenured researcher at the French National Institute for Research in Digital Science and Technology (Inria), since 2014. Prior to this, he was a full-time researcher at the Basque Center for Applied Mathematics and a postdoctoral researcher at Inria. He received his PhD in computer engineering at Politecnico di Milano (Italy) in 2009. At the intersection of applied mathematics, computer science and engineering, his research interests focus on decision making under uncertainty, with particular emphasis on the development of highly-scalable algorithms that minimize congestion and operational costs of large-scale distributed systems.

Josu Doncel is an associate professor in the University of the Basque Country. He obtained from the same university the Industrial Engineering degree in 2007, the Mathematics degree in 2010 and, in 2011, the Master degree in Applied Maths and Statistics. He received in 2015 the PhD degree from Université de Toulouse (France). He has previously held research positions at LAAS-CNRS (France), INRIA Grenoble (France) and BCAM-Basque Center for Applied Mathematics (Spain), teaching positions at ENSIMAG (France), INSA-Toulouse (France) and IUT-Blagnac (France) and invited professor positions at CentraleSupelec (France), Inria Paris (France) and David laboratory (France).