Load Balancing in Heterogeneous Server Clusters: Insights From a Product-Form Queueing Model

Mark van Der Boor, Céline Comte

Load Balancing in Heterogeneous Server Clusters: Insights From a Product-Form Queueing Model*

Mark van der Boor and Céline Comte

Eindhoven University of Technology

September 2, 2021

Abstract

Efficiently exploiting servers in data centers requires performance analysis methods that account not only for the stochastic nature of demand but also for server heterogeneity. Although several recent works proved optimality results for heterogeneity-aware variants of classical load-balancing algorithms in the many-server regime, we still lack a fundamental understanding of the impact of heterogeneity on performance in finite-size systems. In this paper, we consider a load-balancing algorithm that leads to a product-form queueing model and can therefore be analyzed exactly even when the number of servers is finite. We develop new analytical methods that exploit its product-form stationary distribution to understand the joint impact of the speeds and buffer lengths of servers on performance. These analytical results are supported and complemented by numerical evaluations that cover a large variety of scenarios.

Keywords— Load balancing, performance analysis, product-form queueing model, Jackson network, insensitivity

1 Introduction

Distributing a stochastic demand across a set of heterogeneous servers is not only a fundamental problem in queueing theory but also an essential building block of applications like parallel computing and production systems. Besides static approaches that do not require communication between the dispatcher and the servers, classical solutions include join-the-shortest-queue, power-of-d-choices [19], and join-idle-queue [18]. These solutions were originally designed for service systems that are *homogeneous* in the sense that all servers have the same speed [5]. Although these solutions successfully cope with the stochastic nature of demand, they are not always suitable when servers have unequal speeds [10].

Several heterogeneity-aware approaches were introduced to improve the performance of these classical solutions, for instance by using information on the job sizes and server speeds or by delaying the assignment decision. For instance, join-the-shortest-workload yields optimal performance if the service times of all jobs over all servers are known, an assumption that is rarely satisfied in practice. Redundancy scheduling achieves the same performance gain [2], this time by delaying the assignment decision, which may again be practically infeasible if the communication time between the dispatcher and servers is not negligible. To achieve good performance without these strong assumptions, more recent works introduced speed-aware variants of the above-mentioned well-known algorithms. More specifically, [21] introduced variants of join-the-shortest-queue and join-idle-queue where the server speeds are used as a tie-breaking rule, and proved that these variants minimize the mean response time in the many-server regime; [10] proposed variants of power-of-d-choices and join-idle-queue for service systems with two server types (fast and slow) by adapting the degree of diversity and assignment probabilities to the server speeds, and proved stability, again in the many-server regime. Despite these advances, we still lack a fundamental understanding of the impact of heterogeneity on performance in service systems with a finite number of servers.

In this paper, we make one step further into this direction by considering a load-balancing algorithm [3, 7, 8, 17] that leads to a product-form queueing model (assuming that jobs arrive according to a Poisson process), and can

*Author version of the paper available at https://doi.org/10.1109/IWQOS52092.2021.9521355.

therefore be analyzed exactly even when the number of servers is finite. Assuming that each server has a finite-length buffer, this algorithm assigns each incoming job to a server chosen at random, with a probability proportional to the number of available slots in the server's buffer; an incoming job that finds the buffers of all servers full is rejected and considered permanently lost. Although this algorithm does not account for the server speeds in the online assignment decision, these speeds can be used offline to adjust the buffer lengths. Besides its analytical tractability, this algorithm has the advantage of making performance insensitive to the job size distribution beyond its mean, provided that servers apply the processor-sharing policy. This insensitivity property, which contributed to the success of the Erlang-B formula for dimensioning circuit-switched networks, guarantees that the long-run performance metrics are not impacted by fine-grained traffic characteristics. In this paper, we use the product-form queueing model to better understand the impact of parameters on performance.

Related work on insensitive load balancing The works [1, 3, 16] analyze the performance of variants and generalizations of this load-balancing algorithm in heterogeneous service systems where jobs have constraints that restrict their assignment to resources. However, the objective of these works is to develop methods to *calculate* performance metrics, which is a different goal than *understanding* the impact of parameters on performance. Some of the formulas derived in these works are used in the numerical-evaluation section to assess our analytical results. The objectives of the related work [17] are closer to our work. This work analyzes the performance of the same algorithm, but it focuses on the many-server and heavy-traffic regimes and assumes, for the most part, that servers are homogeneous. The product-form stationary distribution of load-balancing models has also been studied in [4, 7, 8, 11] for systems with multiple dispatcher or arbitrary server-job compatibilities (also see references therein).

Additional related work on load balancing If all servers have unit-length buffers, our algorithm can be seen as a loss variant of join-idle-queue [18], whereby a job is rejected if all servers are busy upon its arrival. With arbitrary buffer lengths, our algorithm is related to idle-one-queue [13] and join-below-threshold [12, 15, 23, 24, 25], two generalizations of join-idle-queue introduced to improve performance in the heavy-traffic regime or when servers have unequal speeds. The idea is that servers notify the dispatcher when the number of jobs in their buffer falls below a threshold, so that the dispatcher assigns incoming jobs to lightly-loaded servers if possible. The threshold of a server, equal to one under join-idle-queue, offers a trade-off between performance improvement and communication overhead; in case of unequal server speeds, it can also be used to favor faster servers. In our algorithm, these thresholds correspond to the buffer lengths, and the trade-off between performance and communication overhead is materialized by the overall buffer length (that is, the sum of the lengths of the buffers at all servers). Assuming that this overall buffer length is fixed, we would like to understand how to optimally choose the buffer length of each server depending on the job arrival rate and the server speeds.

Contributions Our contributions can be summarized as follows. We first show the following results for a cluster of two servers in which the overall buffer size across the two servers is fixed. When the arrival rate is low, the optimal buffer lengths (to minimize the loss probability) are proportional to the server speeds, meaning that the buffer of a server is longer if this server is faster. On the contrary, when the arrival rate is large, the optimal buffer lengths are uniform, and the server speeds only intervene to break ties if the overall buffer length is odd. We also show that, between these two limiting regimes, the optimal buffer lengths evolve monotonically with the arrival rate. Besides the practical implications of these results in terms of system design, the analytical methods that we develop, based on an analogy with weighted paths in the two-dimensional lattice, are of independent interest. Afterwards, we explain how these results extend to clusters of more than two servers. We finally turn to numerical evaluations to assess the validity of these results in practice and understand the impact of parameters on the mean response time.

Organization of the paper The remainder of the paper is organized as follows. Section 2 introduces the cluster model and the equivalent closed queueing model. In Section 3, we use this model to derive closed-form expressions for several performance metrics, such as the loss probability. Sections 4 to 6 contain our main contributions for clusters of two servers. In Section 4, we prove that the optimal buffer lengths in terms of the loss probability are proportional to the server speeds when the arrival rate tends to zero, while, in Section 5, we prove that the optimal buffer lengths are uniform when the arrival rate tends to infinity. Section 6 fills the gap between these two limiting regimes by showing a monotonicity result. These results are generalized to clusters of more than two servers in Section 7. Section 8 gives numerical evaluations and Section 9 concludes the paper.

2 Heterogeneous server cluster

We consider a cluster that consists of a single dispatcher and two servers. Incoming jobs arrive at the dispatcher according to a Poisson process with rate λ. Each server has a finite buffer, of length ℓ_1 for server 1 and ℓ_2 for server 2, that contains all jobs assigned to this server (either waiting or in service), with the total buffer size being denoted by $L = \ell_1 + \ell_2$. An incoming job is permanently lost if the buffers of both servers are full upon its arrival, otherwise the dispatcher immediately assigns the job to one of the servers, as will be elaborated further. Two equivalent state descriptors are the vector $n = (n_1, n_2)$ that counts the jobs in the buffer of each server, and the vector $x = (x_1, x_2)$ that counts the available slots in the buffers of each server. The generalization of this model to clusters of more than two servers will be considered in Section 7.

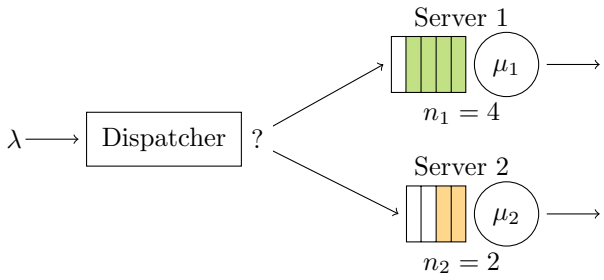

Figure 1: A heterogeneous cluster with two servers.

2.1 Load balancing and scheduling

The dispatcher applies the following randomized load-balancing algorithm, considered in [3, 7, 8, 17]. When a new job arrives, the dispatcher chooses a server at random, with a probability proportional to the number of available slots in the buffer of the server, and assigns the job to this server. In Figure 1 for instance, there are one and two available slots in the buffers of servers 1 and 2, respectively, so that an incoming job would be assigned to server 1 with probability $\frac{1}{3}$ and to server 2 with probability $\frac{2}{3}$. In general, if there are x_1 and x_2 available slots in the buffers of servers 1 and 2, respectively, then server 1 is chosen with probability $\frac{x_1}{x_1+x_2}$ and server 2 with probability $\frac{x_2}{x_1+x_2}$ if $x_1 + x_2 \geq 1$, otherwise the job is lost. This algorithm assumes that the dispatcher always knows the number of available slots in the buffer of each server; this happens, for instance, if jobs go through the dispatcher again when they leave the system, or if servers notify the dispatcher upon service completions. This load-balancing algorithm was considered in [3, 7, 8, 17] for clusters with an arbitrary number of servers.

We assume that each server processes the jobs in its buffer according to a non-anticipating work-conserving scheduling algorithm, such as processor-sharing or first-come-first-served. The service rates of the servers, assumed to be constant for simplicity, are denoted by μ_1 and μ_2. We assume without loss of generality that $1 > \mu_1 > \mu_2 > 0$ and $\mu_1 + \mu_2 = 1$, so that server 1 is the fastest. The job service requirements are independent and exponentially distributed with unit mean, so that the remaining service time of a job is exponentially distributed with mean $\frac{1}{\mu}$ if this job is currently served at rate μ. This memoryless assumption is actually not required to perform the subsequent analysis if each server applies processor-sharing or preemptive-resume last-come-first-served. Each job leaves the system immediately upon service completion.

Even if the dispatcher does not take the service rates μ_1 and μ_2 into account when making the assignment decision, we will see later that varying the buffer lengths ℓ_1 and ℓ_2 allows us to optimize the load distribution with respect to the service rates.

2.2 Queueing model

The dynamics can be described by a closed Jackson network [20] of three stations with two customer (or token) classes [4, 7]. In Section 3, we will use this observation to derive closed-form expressions for several performance metrics.

Instead of counting how many jobs are present in the cluster, we keep track to how tokens evolve in a network, where every token corresponds to a specific slot in a server's buffer. A token corresponding to a slot in the buffer of server i is said to be of class i, for $i \in \{1, 2\}$. When a slot in the buffer of server i is not occupied, the corresponding class-i token is located at the *dispatcher station*; when this slot becomes occupied by a job, the class-i token moves

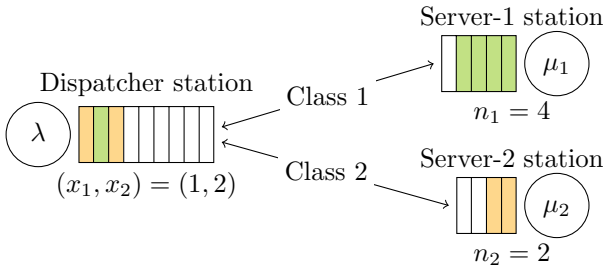

Figure 2: Jackson network associated with the cluster of Figure 1.

to the *server-i station*. The routing mechanism of tokens is deterministic, and each class-i token moves only between the dispatcher station and the server-i station. More specifically, whenever a job completes service at server i (this happens with rate μ_i), the corresponding slot becomes available and the corresponding token moves to the dispatcher station. Whenever a job is assigned to server i, a slot in the buffer of this server becomes occupied by this job, so that the corresponding class-i token moves to server-i station.

The corresponding closed Jackson network consists of the dispatcher station and the server-1 and 2 stations. There are ℓ_i tokens of class i, for $i \in \{1, 2\}$, and these tokens are either in the dispatcher station or in the server-i station. The service policy in the dispatcher station is processor-sharing, while the service policies in the server stations match those applied by the servers. The state of this network is described by the vector $x = (x_1, x_2)$ that counts the number of tokens in the dispatcher station, corresponding to available slots in the servers' buffers. In particular, $x = 0$ means that both buffers are full, while $x = \ell$, with $\ell = (\ell_1, \ell_2)$, means that the cluster is empty. We let $e_1 = (1, 0)$ and $e_2 = (0, 1)$ denote the states corresponding to a single (slot occupied by a) job at server i. Known results on closed Jackson networks [20, Chapter 1 and Section 3.1] show that the stationary distribution of the Markov process defined by the evolution of the network state is given by

$$\pi(x) = \frac{1}{G(\ell)} \binom{x_1 + x_2}{x_1} \left(\frac{\mu_1}{\lambda}\right)^{x_1} \left(\frac{\mu_2}{\lambda}\right)^{x_2}, \quad x \leq \ell, \tag{1}$$

where $G(\ell)$ is a normalization constant.

3 Performance analysis

As in Jackson networks with a single class of customers [6], the closed-form expressions of several metrics of interest stem directly from the normalization constant.

3.1 Normalization constant

The normalization constant follows from the normalization equation $\sum_{x \leq \ell} \pi(x) = 1$. Using (1), we obtain

$$G(\ell) = \sum_{x \leq \ell} \binom{x_1 + x_2}{x_1} \left(\frac{\mu_1}{\lambda}\right)^{x_1} \left(\frac{\mu_2}{\lambda}\right)^{x_2}, \quad \ell \in \mathbb{N}^2. \tag{2}$$

The following geometric interpretation guides the proofs of Theorems 1 and 2 in Sections 4 and 5. Consider the square lattice consisting of the 2-dimensional vectors $x = (x_1, x_2)$ with integer components. We are interested in the direct paths going from the origin 0 to some vector $x \leq \ell$, where by direct we mean a path that consists only of increasing unit steps in the horizontal or vertical direction. If each horizontal step has weight $\frac{\mu_1}{\lambda}$ and each vertical step weight $\frac{\mu_2}{\lambda}$, then $(\frac{\mu_1}{\lambda})^{x_1}(\frac{\mu_2}{\lambda})^{x_2}$ is the multiplicative weight of any direct path going from the origin 0 to the vector x. Since there are $\binom{x_1+x_2}{x_1}$ such paths, it follows that the normalization constant $G(\ell)$ is the sum of the weights of all direct paths going from the origin to a vector $x \leq \ell$. Intuitively, this suggests that:

- If $\lambda \ll \mu_1 + \mu_2 (= 1)$, the paths that have the most weight are the longest, meaning that the states with large values of $x_1 + x_2$ (that is, with many available slots) are the most likely. This intuition guides the proof of Theorem 1.

- If $\lambda \gg 1$, the paths that have the most weight are the shortest, meaning that the states with small values of $x_1 + x_2$ (that is, with few available slots) are the most likely. This intuition guides the proof of Theorem 2.

Although (2) unveils interesting properties of the constant $G(\ell)$, this expression is inconvenient when it comes to *compute* this constant because it leads to numerical instability. For the numerical results, we use instead the recursive expression

$$G(\ell) = 1 + \sum_{\substack{i=1 \\ \ell_i \geq 1}}^{2} \frac{\mu_i}{\lambda} G(\ell - e_i), \quad \ell \in \mathbb{N}^2 \setminus \{0\},$$

with the base case $G(0) = 1$. This expression, which is a special case of the formula derived in [3, Proposition 2], can be seen as a generalization of the Erlang-B formula [9].

It will often be convenient to consider the quantity

$$\delta G(\ell) = G(\ell + e_1 - e_2) - G(\ell), \quad \ell \in \mathbb{N}^2 : \ell_2 \geq 1, \tag{3}$$

that gives the variation of the normalization constant obtained by replacing a server-2 slot with a server-1 slot. By injecting (2) into this definition and making simplifications, we obtain

$$\delta G(\ell) = \sum_{n=\ell_1+1}^{\ell_1+\ell_2} \binom{n}{\ell_1+1} \left(\frac{\mu_1}{\lambda}\right)^{\ell_1+1} \left(\frac{\mu_2}{\lambda}\right)^{n-\ell_1-1} - \sum_{n=\ell_2}^{\ell_1+\ell_2} \binom{n}{\ell_2} \left(\frac{\mu_1}{\lambda}\right)^{n-\ell_2} \left(\frac{\mu_2}{\lambda}\right)^{\ell_2}. \tag{4}$$

3.2 Long-term performance metrics

We now consider three performance metrics called the loss probability, occupation rate, and mean response time. The formulas below are simple extensions of formulas derived for closed Jackson networks with a single class of customers [6]. However, to the best of our knowledge, the results regarding the occupation rate and mean response time have never been derived in the literature on insensitive load balancing.

Loss probability The loss probability $\beta(\ell)$ is defined as the probability that an incoming job is rejected, which happens when the buffers of all servers are full upon its arrival. According to the PASTA property [22], the loss probability is equal to the stationary probability $\pi(0)$ that the buffers of all servers are full, so that then by (1) we obtain

$$\beta(\ell) = \frac{1}{G(\ell)}. \tag{5}$$

By (2), the loss probability decreases when the number of slots in a given buffer increases (as intuition suggests).

Occupation rate For each $i \in \{1, 2\}$, the occupation rate of server i is defined as the fraction of time that this server is busy. According to (1), this quantity $\rho_i(\ell)$ is given by

$$\frac{1}{G(\ell)} \sum_{x \leq \ell - e_i} \binom{x_1 + x_2}{x_1} \left(\frac{\mu_1}{\lambda}\right)^{x_1} \left(\frac{\mu_2}{\lambda}\right)^{x_2} = \frac{G(\ell - e_i)}{G(\ell)}.$$

We have $\rho_1(\ell) > \rho_2(\ell)$ if and only if $G(\ell - e_1) > G(\ell - e_2)$, which by (5) also means that the loss probability increases less when we remove a slot from server 1 than when we remove a slot from server 2.

Mean response time The response time of a job is defined as the duration between its arrival in the cluster and its departure. We can show that the mean numbers of jobs in the buffers of servers 1 and 2 are given by

$$\alpha_1(\ell) = \frac{\sum_{x_1=0}^{\ell_1-1} G(x_1, \ell_2)}{G(\ell)}, \qquad \alpha_2(\ell) = \frac{\sum_{x_2=0}^{\ell_2-1} G(\ell_1, x_2)}{G(\ell)}.$$

According to Little's law, the mean response time of a job is given by $\Delta(\ell) = \frac{\alpha_1(\ell) + \alpha_2(\ell)}{\lambda(1 - \beta(\ell))}$.

3.3 Problem statement

We would like to understand the joint impact of the arrival rate λ, service rates μ_1 and μ_2, and buffer lengths ℓ_1 and ℓ_2 on these performance metrics. Despite the apparent simplicity of these expressions, using them to gain intuition on the impact of parameters on performance is a well-known difficult problem [14]. The results of Section 3.2 suggest however that the loss probability is the easiest of these metrics to analyze, as it is simply the inverse of the normalization constant.

We observed earlier that, although the load-balancing algorithm does not account for the service rates μ_1 and μ_2 to make the assignment decision, the buffer lengths ℓ_1 and ℓ_2 can be adjusted to optimize performance. Therefore, Sections 4 to 6, which contain our main contributions, will focus more specifically on the following question:

> Given the arrival rate λ, service rates μ_1 and μ_2, and overall buffer length $L = \ell_1 + \ell_2$, which value(s) of ℓ_1 and ℓ_2 minimize(s) the loss probability?

Since $\mu_1 > \mu_2$, a natural strategy consists of allocating (almost) all slots to server 1. However, the more jobs are in service on server 1, the smaller the fraction of the service rate of this server they receive, and the longer they stay in the system, thus preventing other jobs from entering. This simple observation suggests that it would be better to maximize the minimum service rate received by any job by allocating slots to servers proportionally to their service rates, that is, allocate a fraction μ_1 of the slots to server 1 and a fraction μ_2 of the slots to server 2. Theorem 1 shows that this allocation is indeed optimal when the arrival rate is small. Theorems 2 and 3 show that, as the arrival rate increases, the optimal allocation evolves monotonically towards a uniform allocation, in which both servers have approximately the same number of slots.

Extending these results to understand the impact of parameters on the mean response time is not straightforward. This impact will be assessed numerically in Section 8.

4 Low-traffic regime

Our first main contribution is a theorem showing that, when the arrival rate is small, the loss probability is minimized by allocating slots to servers in proportion to their service rate.

Theorem 1. *There is a $\lambda_* > 0$ such that, for each $\lambda \in (0, \lambda_*]$, the loss probability is minimized when $\ell = (\ell_1, L - \ell_1)$ with*

$$\ell_1 = \lceil \mu_1 L - \mu_2 \rceil \quad \text{or} \quad \lfloor \mu_1 L + \mu_1 \rfloor . \tag{6}$$

Proof. We first show that, as $\lambda \to 0$, the monotonicity of $G(\ell)$ as a function of ℓ is entirely dictated by that of the term corresponding to $x = \ell$ in (2). We will then study this term.

Let $\ell = (\ell_1, \ell_2) \in \mathbb{N}^2$ with $\ell_1 + \ell_2 = L$. First observe that

$$\sum_{x \le \ell : x \ne \ell} \binom{x_1 + x_2}{x_1} \left(\frac{\mu_1}{\lambda}\right)^{x_1} \left(\frac{\mu_2}{\lambda}\right)^{x_2} \le \frac{\left(\frac{1}{\lambda}\right)^L - 1}{\frac{1}{\lambda} - 1}.$$

By injecting this inequality into (2), we obtain

$$G(\ell) = \binom{L}{\ell_1} \left(\frac{\mu_1}{\lambda}\right)^{\ell_1} \left(\frac{\mu_2}{\lambda}\right)^{\ell_2} + O_{\lambda \to 0}\left(\left(\frac{1}{\lambda}\right)^{L-1}\right). \tag{7}$$

If $\ell_2 \ge 1$, we can apply this result to both ℓ and $\ell + e_1 - e_2$, so that, by (3), we obtain

$$\delta G(\ell) = \left(\frac{\ell_2}{\ell_1 + 1} \frac{\mu_1}{\mu_2} - 1\right) \cdot \binom{L}{\ell_1} \left(\frac{\mu_1}{\lambda}\right)^{\ell_1} \left(\frac{\mu_2}{\lambda}\right)^{\ell_2} + O_{\lambda \to 0}\left(\left(\frac{1}{\lambda}\right)^{L-1}\right).$$

As λ tends to zero, the first term tends to $+\infty$ like $(\frac{1}{\lambda})^L$, while the second term tends to $+\infty$ at most like $(\frac{1}{\lambda})^{L-1}$. Therefore, there is $\lambda_* > 0$ such that, for each $\lambda \in (0, \lambda_*]$ and each $\ell \in \mathbb{N}^2$ such that $\ell_1 + \ell_2 = L$ and $\ell_2 \ge 1$, $\delta G(\ell)$ is of the same sign as $(\frac{\ell_2}{\ell_1 + 1} \frac{\mu_1}{\mu_2} - 1)$, provided that this quantity is nonzero.

Now let $\lambda \in (0, \lambda_*]$ and assume that servers 1 and 2 have ℓ_1 and ℓ_2 slots, respectively, with $\ell_1 + \ell_2 = L$. According to the above equality, replacing a slot of server 2 (if any) with a slot of server 1 reduces the loss probability whenever $\frac{\ell_2}{\ell_1 + 1} \frac{\mu_1}{\mu_2} < 1$, that is, $\ell_1 < \mu_1 L - \mu_2$. By taking the symmetrical statement and rearranging the terms, we obtain that replacing a slot of server 1 (if any) with a slot of server 2 reduces the loss probability whenever $\ell_1 > \mu_1 L + \mu_1$. Therefore, the slot allocation can only be optimal when $\mu_1 L - \mu_2 \le \ell_1 \le \mu_1 L + \mu_1$, which is equivalent to (6). ∎

Remark 1. The ceil and floor values in (6) are different only in the pathological case where $\mu_1 L + \mu_1 = \mu_1 L - \mu_2 + 1$ is an integer, which means that μ_1 and μ_2 can be written as fractions with denominator $L+1$. In this case, the term corresponding to $x = \ell$ is zero in the expression of $\delta G(\ell)$, and which of $\ell_1 = \lceil \mu_1 L - \mu_2 \rceil$ or $\ell_1 = \lfloor \mu_1 L + \mu_1 \rfloor$ is optimal depends on the terms corresponding to $x \leq \ell$ and $x \neq \ell$.

5 Heavy-traffic regime

The following theorem shows that, when the arrival rate λ is large, the loss probability is minimized by taking $\ell_1 \simeq \ell_2$. The arrival rates μ_1 and μ_2 only serve to allocate the remaining slot when the overall buffer length L is odd.

Theorem 2. *There is $\lambda^* > 0$ such that, for each $\lambda \in [\lambda^*, \infty)$, the loss probability is minimized by choosing:*

- $\ell_1 = \ell_2 = \frac{L}{2}$ *if L is even;*
- $\ell_1 = \frac{L+1}{2}$ *and* $\ell_2 = \frac{L-1}{2}$ *if L is odd.*

Proof. Let $\ell \in \mathbb{N}^2$ such that $\ell_1 + \ell_2 = L$ and $\ell_1 \leq L - 1$. Using (4), we proceed by exhaustion, distinguishing several cases depending on the values of ℓ_1 and ℓ_2.

Case 1 ($\ell_1 \geq \ell_2$) We can split the second sum of (4) into two parts, corresponding to $n \in \{\ell_1 + 1, \ldots, \ell_1 + \ell_2\}$ and $n \in \{\ell_2, \ldots, \ell_1\}$, respectively. We obtain

$$\delta G(\ell) = \sum_{n=\ell_1+1}^{\ell_1+\ell_2} \left[\binom{n}{\ell_1+1} \left(\frac{\mu_1}{\lambda}\right)^{\ell_1+1} \left(\frac{\mu_2}{\lambda}\right)^{n-\ell_1-1} - \binom{n}{\ell_2} \left(\frac{\mu_1}{\lambda}\right)^{n-\ell_2} \left(\frac{\mu_2}{\lambda}\right)^{\ell_2} \right] - \sum_{n=\ell_2}^{\ell_1} \binom{n}{\ell_2} \left(\frac{\mu_1}{\lambda}\right)^{n-\ell_2} \left(\frac{\mu_2}{\lambda}\right)^{\ell_2}.$$

As λ tends to $+\infty$, each term in the first sum tends to zero at least as fast as $(\frac{1}{\lambda})^{\ell_1+1}$, while each term in the second sum tends to zero at most as fast as $(\frac{1}{\lambda})^{\ell_1}$. Therefore, when λ is sufficiently large, we have $\delta G(\ell) < 0$ whenever $\ell_1 \geq \ell_2$.

Case 2 ($\ell_1 \leq \ell_2 - 2$) We can split the first sum of (4) into two parts, corresponding to $n \in \{\ell_1 + 1, \ldots, \ell_2 - 1\}$ and to $n \in \{\ell_2, \ldots, \ell_1 + \ell_2\}$, respectively. We obtain

$$\delta G(\ell) = \sum_{n=\ell_1+1}^{\ell_2-1} \binom{n}{\ell_1+1} \left(\frac{\mu_1}{\lambda}\right)^{\ell_1+1} \left(\frac{\mu_2}{\lambda}\right)^{n-\ell_1-1} + \sum_{n=\ell_2}^{\ell_1+\ell_2} \left[\binom{n}{\ell_1+1} \left(\frac{\mu_1}{\lambda}\right)^{\ell_1+1} \left(\frac{\mu_2}{\lambda}\right)^{n-\ell_1-1} - \binom{n}{\ell_2} \left(\frac{\mu_1}{\lambda}\right)^{n-\ell_2} \left(\frac{\mu_2}{\lambda}\right)^{\ell_2} \right].$$

As λ tends to $+\infty$, each term in the first sum tends to zero at most as fast as $(\frac{1}{\lambda})^{\ell_2-1}$, while each term in the second sum tends to zero at least as fast as $(\frac{1}{\lambda})^{\ell_2}$. Therefore, when λ is sufficiently large, we have $\delta G(\ell) > 0$ whenever $\ell_1 \leq \ell_2 - 2$.

Case 3 ($\ell_1 = \ell_2 - 1$, assuming that L is odd) The two sums in (4) contain the same number of terms, and we obtain:

$$\delta G(\ell) = \sum_{n=\ell_1+1}^{\ell_1+\ell_2} \binom{n}{\ell_1+1} \left(\frac{\mu_1}{\lambda}\right)^{n-\ell_1-1} \left(\frac{\mu_2}{\lambda}\right)^{n-\ell_1-1} \times \left[\left(\frac{\mu_1}{\lambda}\right)^{\ell_1+\ell_2+1-n} - \left(\frac{\mu_2}{\lambda}\right)^{\ell_1+\ell_2+1-n} \right].$$

Since $\mu_1 > \mu_2$, it follows that $\delta G(\ell) > 0$.

Conclusion We now gather the three cases. If L is even, the sequence $\ell_1 \mapsto \beta(\ell_1, L - \ell_1)$ is decreasing on $\{0, 1, \ldots, \frac{L}{2}\}$ and increasing on $\{\frac{L}{2}, \ldots, L-1, L\}$. Therefore, the loss probability is minimal when $\ell_1 = \ell_2 = \frac{L}{2}$. If L is odd, the sequence $\ell_1 \mapsto \beta(\ell_1, L - \ell_1)$ is decreasing on $\{0, 1, \ldots, \frac{L+1}{2}\}$ and increasing on $\{\frac{L+1}{2}, \ldots, L-1, L\}$. Therefore, the loss probability is minimal when $\ell_1 = \frac{L+1}{2}$ and $\ell_2 = \frac{L-1}{2}$. ∎

6 Monotonicity

To make the connection between the results of the last two sections, we now show that the optimal slot allocation is monotonic with respect to the arrival rate λ. In the following theorem, with "optimal number of slots allocated to the fastest server", we mean the *smallest* number of slots allocated to the fastest server that minimizes the loss probability. This will be discussed again in Remark 2.

Theorem 3. *The optimal number of slots allocated to the fastest server, in terms of the loss probability, is decreasing with the arrival rate λ.*

Proof. It will be convenient to write the loss probability as a function $\beta(\lambda, \ell)$ of both the arrival rate λ and buffer lengths $\ell = (\ell_1, \ell_2)$. The proof relies on two monotonicity results that are presented in the following two propositions.

Proposition 4. *Let $\lambda > 0$ and $\ell \in \mathbb{N}^2$ such that $\ell_1 \geq 1$ and $\ell_2 \geq 1$. If $\beta(\lambda, \ell + e_1 - e_2) \leq \beta(\lambda, \ell)$, then $\beta(\lambda, \ell - xe_1 + xe_2) < \beta(\lambda, \ell - (x+1)e_1 + (x+1)e_2)$ for each $x \in \{0, 1, 2, \ldots, \ell_1 - 1\}$.*

Proposition 5. *Let $\lambda_* > 0$ and $\ell \in \mathbb{N}^2$ such that $\ell_2 \geq 1$. If $\beta(\lambda_*, \ell + e_1 - e_2) < \beta(\lambda_*, \ell)$, then $\beta(\lambda, \ell + e_1 - e_2) < \beta(\lambda, \ell)$ for each $\lambda \in (0, \lambda_*)$.*

Propositions 4 and 5 are proven in Appendices A and B, respectively. These propositions can be rephrased as follows. Assume that, for a given arrival rate λ_* and overall buffer length L, we know that allocating $\ell_1 + 1$ slots to the fastest server yields better performance than allocating ℓ_1 slots to this server. Then Proposition 4 shows that, with the same arrival rate λ_*, allocating even fewer slots than ℓ_1 to the fastest server is even worse in terms of performance. Proposition 5 shows that allocating $\ell_1 + 1$ slots to the fastest server remains better than ℓ_1 slots for any arrival rate $\lambda \in (0, \lambda_*)$.

Now consider two arrival rates $\lambda_* \in (0, \infty)$ and $\lambda \in (0, \lambda_*)$. Let ℓ^{λ_*} and ℓ^{λ} denote the optimal slot allocations under these arrival rates, with $L = \ell_1^{\lambda_*} + \ell_2^{\lambda_*} = \ell_1^{\lambda} + \ell_2^{\lambda}$. Our objective is to prove that $\ell_1^{\lambda_*} \leq \ell_1^{\lambda}$. The optimality of ℓ^{λ_*} implies that $\beta(\lambda_*, \ell^{\lambda_*}) < \beta(\lambda_*, \ell^{\lambda_*} - e_1 + e_2)$. Therefore, Proposition 4 gives $\beta(\lambda_*, \ell^{\lambda_*} - xe_1 + xe_2) < \beta(\lambda_*, \ell^{\lambda_*} - (x+1)e_1 + (x+1)e_2)$ for each $x \in \{0, 1, 2, \ldots, \ell_1^{\lambda_*} - 1\}$. By Proposition 5, each of these inequalities yields $\beta(\lambda, \ell^{\lambda_*} - xe_1 + xe_2) < \beta(\lambda, \ell^{\lambda_*} - (x+1)e_1 + (x+1)e_2)$ for each $x \in \{0, 1, 2, \ldots, \ell_1^{\lambda_*} - 1\}$. This in turn implies that $\ell_1^{\lambda_*} \leq \ell_1^{\lambda}$. □

Remark 2. Proposition 4 shows that, among all vectors $\ell \in \mathbb{N}^2$ such that $L = \ell_1 + \ell_2$, at most two can minimize the loss probability, and these are separated by only one slot. Therefore, in Theorem 3, the "optimal number of slots allocated to the fastest server" is defined up to plus or minus one, and we systematically choose the smallest value for convenience.

7 Extension to more than two servers

We now consider a cluster that consists of a dispatcher and a set $\mathcal{I} = \{1, 2, \ldots, N\}$ of servers. For each $i \in \mathcal{I}$, we let μ_i denote the service rate of server i, ℓ_i the length of its buffer, and x_i the number of available slots in this buffer. We assume without loss of generality that $1 > \mu_1 \geq \mu_2 \geq \ldots \geq \mu_N > 0$ and $\sum_{i \in \mathcal{I}} \mu_i = 1$, and we let $L = \sum_{i \in \mathcal{I}} \ell_i$ denote the overall buffer length. All definitions of Sections 2 and 3 are generalized in a natural way to this N-server cluster. In particular, the load-balancing algorithm is generalized as follows: an incoming job is assigned to server i with probability $\frac{x_i}{\sum_{j \in \mathcal{I}} x_j}$ for each $i \in \mathcal{I}$ if $\sum_{j \in \mathcal{I}} x_j \geq 1$, otherwise the job is rejected. The corresponding closed Jackson network consists of $N + 1$ stations that correspond to the dispatcher and the N servers, respectively. The set of token classes is $\mathcal{I}$ and there are ℓ_i class-i tokens, for each $i \in \mathcal{I}$.

Stationary distribution The stationary distribution of the Markov process defined by the evolution of the network state $x = (x_1, x_2, \ldots, x_N)$ over time is given by

$$\pi(x) = \frac{1}{G(\ell)} \binom{x_1 + x_2 + \ldots + x_N}{x_1, x_2, \ldots, x_N} \prod_{i=1}^{N} \left(\frac{\mu_i}{\lambda}\right)^{x_i}, \quad x \leq \ell,$$

where $\binom{x_1+x_2+\ldots+x_N}{x_1,x_2,\ldots,x_N} = \frac{(x_1+x_2+\ldots+x_N)!}{x_1!x_2!\cdots x_N!}$ is a multinomial coefficient, and the constant $G(\ell)$ is obtained by normalization. For each $i, j \in \mathcal{I}$ and $\ell \in \mathbb{N}^N$ with $\ell_j \geq 1$, the variation of the normalization constant obtained by replacing a server-j slot with a server-i slot is denoted by $\delta_{j \to i} G(\ell) = G(\ell + e_i - e_j) - G(\ell)$. We now explain how to generalize our results.

Theorem 1 Following the same approach as in the proof of Theorem 1, we obtain the following generalization of (7):

$$G(\ell) = \binom{L}{\ell_1, \ell_2, \ldots, \ell_N} \prod_{i=1}^{N} \left(\frac{\mu_i}{\lambda}\right)^{\ell_i} + O_{\lambda \to 0}\left(\left(\frac{1}{\lambda}\right)^{L-1}\right).$$

As λ tends to zero, the variations of the second term become negligible compared to those of the first. Therefore, an optimal slot allocation is a mode of the multinomial distribution with parameters L and $\mu_1, \mu_2, \ldots, \mu_N$.

Theorem 2 To generalize the proof of this theorem and the next, it is helpful to rewrite the stationary distribution as

$$\pi(x) = \frac{1}{G(\ell)} \varphi_1(x_1, x_2) \varphi_2(x_1 + x_2, x_{-1,2}), \quad x \leq \ell, \tag{8}$$

where $x_{-1,2} = (x_3, \ldots, x_N)$, and

$$\varphi_1(x_1, x_2) = \binom{x_1 + x_2}{x_1} \left(\frac{\mu_1}{\lambda}\right)^{x_1} \left(\frac{\mu_2}{\lambda}\right)^{x_2}, \qquad \varphi_2(x_1 + x_2, x_{-1,2}) = \binom{x_1 + x_2 + \ldots + x_N}{x_1 + x_2, x_3, \ldots, x_N} \prod_{i=3}^{N} \left(\frac{\mu_i}{\lambda}\right)^{x_i}.$$

The first factor is equal (up to a multiplicative constant) to the stationary distribution obtained in a two-server cluster, while the second factor depends on x_1 and x_2 only via their sum $x_1 + x_2$. Using this expression, we can rewrite $G(\ell)$, and then $\delta_{2\to1} G(\ell)$, as a nested sum over $x_1 \in \{0, 1, \ldots, \ell_1\}$ and $x_2 \in \{0, 1, \ldots, \ell_2\}$, where the term corresponding to x_1 and x_2 is the product of $\varphi_1(x_1, x_2)$ and a factor that depends on x_1 and x_2 only via their sum. Upon observing that this factor is $\Theta_{\lambda \to 0}((\frac{1}{\lambda})^{\ell_3 + \ldots + \ell_N})$, we conclude in a similar way as in the proof of Theorem 1 that, when λ is small enough, we have $\delta_{2\to1} G(\ell) < 0$ if $\ell_1 \geq \ell_2$ and $\delta_{2\to1} G(\ell) > 0$ if $\ell_1 \leq \ell_2 - 1$. A similar result holds for $G_{i\to j}(\ell)$ for each $i, j \in \mathcal{I}$ such that $\ell_j \geq 1$, so that the loss probability is again minimized by choosing the buffer lengths approximately equal to each other.

Theorem 3 The monotonicity result of this theorem can only be generalized to the fastest and slowest servers. More specifically, we can show that the optimal buffer length of the fastest server(s) decreases with the arrival rate, while the optimal buffer length of the slowest server(s) increases with the arrival rate. Indeed, all intermediary results in Appendices A and B can be generalized to two arbitrary servers $i, j \in \mathcal{I}$ using the same approach as in the previous paragraph. The only restriction is that $\mu_i \geq \mu_j$ (the assumption $\mu_i > \mu_j$ is used in Case 1 in the proof of Lemma 7 in Appendix B, and one can verify that it can be alleviated to $\mu_i \geq \mu_j$). Therefore, we can only conclude for a server $i \in \mathcal{I}$ such that either $\mu_i \geq \mu_j$ for each $j \in \mathcal{I} \setminus \{i\}$ or $\mu_i \leq \mu_j$ for each $j \in \mathcal{I} \setminus \{i\}$.

8 Numerical results

We now proceed to some case studies, in which we numerically evaluate the performance measures from Section 3.2 and show how the loss probability and mean response time are dependent on the arrival rate, service times and slot allocation.

8.1 Cluster of two servers

We first consider a cluster of two servers. The overall buffer length is kept constant equal to $L = 20$ in the interest of space, but we observed a similar behavior for other values of L. As before, we assume that $\mu_1 + \mu_2 = 1$ and $\mu_1 > \mu_2$.

Loss probability In Figure 3, the loss probability is shown as a function of the number of slots allocated to the fast server, for several values of the arrival rate λ, first in a linear plot and then in a log-linear plot. As intuition suggests, a larger arrival rate yields a larger loss probability. Since μ_1 is large, the loss probability tends to be lower when the first server has more than half of the slots. The log-linear plot reveals that, even in this range, the loss probability can still be reduced by several orders of magnitude by correctly choosing the buffer lengths.

To make more decisive statements, we consider, in Figure 4, the optimal buffer length of the fast server (called the *optimal buffer length* for brevity) as a function of the arrival rate, for several values of the service rates. The

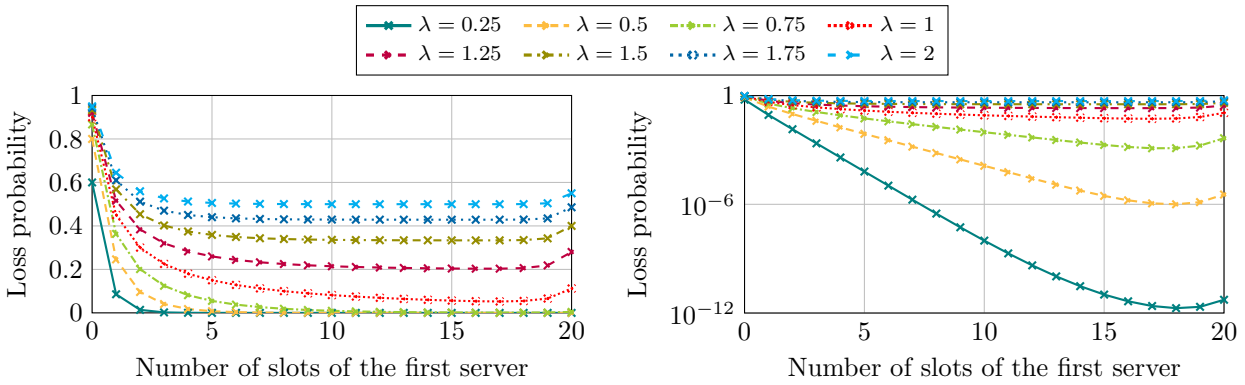

Figure 3: Loss probability in a two-server cluster, as a function of the buffer length of the first server and for several values of the arrival rate, with $L = 20$, $\mu_1 = 0.9$, and $\mu_2 = 0.1$.

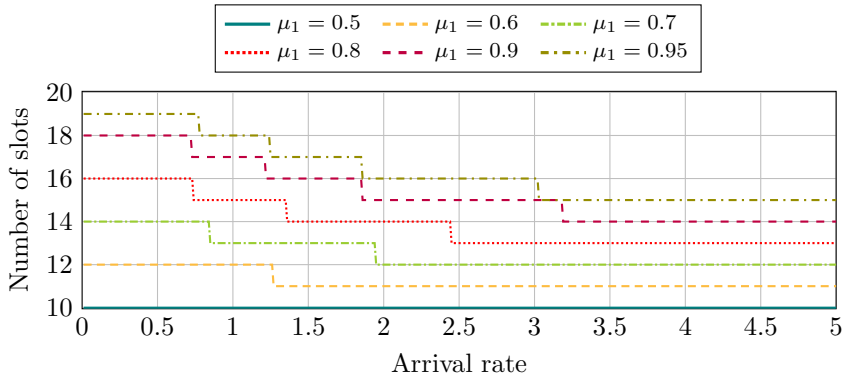

Figure 4: Optimal buffer length of the fast server (to minimize the loss probability) in a two-server cluster, as a function of the arrival rate and for several values of the service rates, with $L = 20$.

optimal buffer length is always at least $\lceil L/2 \rceil$, as it does not make sense to allocate more slots to a slower server, and it increases with μ_1.

When the arrival rate λ is small, the optimal buffer length is approximately $\mu_1 L$, which is consistent with Theorem 1. Although it may at first *seem* that the larger the difference in server speeds, the later the optimal number of slots changes when increasing λ, this is a coincidence, as can be seen by comparing the cases $\mu_1 = 0.9$ and $\mu_1 = 0.95$. As λ increases, the optimal buffer length converges to $L/2 = 10$, as predicted by Theorem 2, but the convergence is slower when μ_1 is larger. Figure 4 lastly shows that the optimal buffer length decreases as the arrival rate increases, as proven in Theorem 3.

Mean response time We now turn to the mean response time, for which we only show numerical results. The mean response time is shown in Figure 5 as a function of the arrival rate λ, for several values of the buffer length of the first server. We first turn to the paradoxical behavior of the system when the majority of slots is allocated to the slowest server. In this case, the mean response time is *not* necessarily monotonous in the arrival rate λ, as can be seen with $\ell_1 = 4$. Indeed, when the arrival rate is low, most jobs are served by the slowest server; as the arrival rate increases, a larger fraction of jobs is sent to the fastest server (even if the buffer of this server is shorter), so that the mean response time decreases. Note that this scenario is suboptimal anyway, as switching around the slots (i.e. giving the majority of slots to the fast server) leads to better performance. As a side remark, the mean response times always converges to the same value as λ increases, as long as at least one slot is allocated to each server.

Figure 6 shows the optimal buffer length of the fast server (to minimize the mean response time) as a function of the arrival rate λ. The same monotonicity property as for the loss probability seems to hold. Furthermore, when the arrival rate is large, the optimal buffer length also seems converge to $L/2$. The main difference with the loss

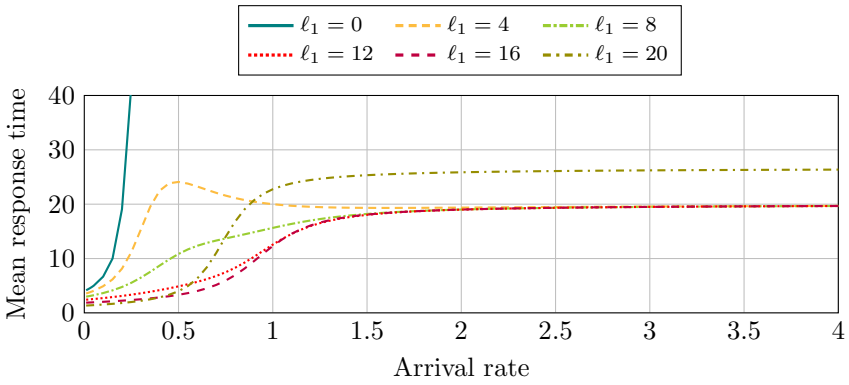

Figure 5: Mean response time in a two-server cluster, as a function of the arrival rate and for several values of the buffer length of the first server, with $L = 20$ and $\mu_1 = 0.75$.

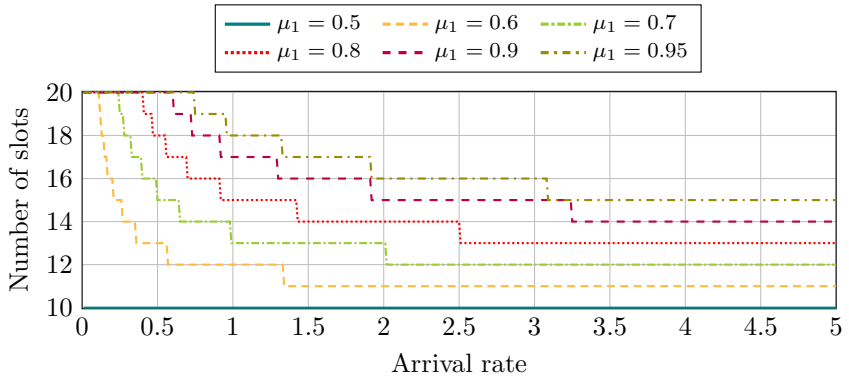

Figure 6: Optimal buffer length of the fast server in a two-server cluster (to minimize the mean response time), as a function of the arrival rate and for several values of the service rates, with $L = 20$.

probability appears in the low-traffic regime: instead of converging to $\lceil \mu_1 L \rceil$, the optimal buffer length of the fastest server seems to converge to L.

8.2 Cluster of four servers

We finally consider a cluster with four servers to illustrate the absence of monotonicity of the optimal buffer length in larger clusters. The server speeds are $\mu_1 = 0.45$, $\mu_2 = 0.3$, $\mu_3 = 0.2$, and $\mu_4 = 0.05$ and overall buffer size $L = 40$. The optimal buffer lengths for each server, in terms of either the loss probability or the mean response time, are shown in Figure 7 and 8 as functions of the arrival rate.

As mentioned in Section 7, all theorems for the loss probability can be generalized to more than two servers: the optimal buffer lengths are proportional to the server speeds when the arrival rate is low and uniform when the arrival rate is large, and the optimal buffer length of the fastest and slowest servers evolve monotonically with the arrival rate. The optimal buffer lengths of servers 2 and 3 are however not monotonic. From extensive numerical experiments conducted for clusters with four to ten servers, we conjecture that, for clusters with N servers, the total optimal buffer size of the $n = 1, 2, \ldots, N$ fastest servers is decreasing, e.g., with $N = 4$, ℓ_1, $\ell_1 + \ell_2$, and $\ell_1 + \ell_2 + \ell_3$ are decreasing in λ.

We observe in Figure 8 that when the arrival rate is low, the optimal buffer size of the fastest server (in terms of mean response time) is L, as in the two-server case. For a large arrival rate, the optimal buffer lengths are uniform. A similar sense of monotonicity is observed for the mean response time: the optimal buffer lengths of the fastest and slowest servers are monotonous. The total optimal buffer length of the $n = 1, 2, \ldots, L$ fastest servers also seems to be monotonous.

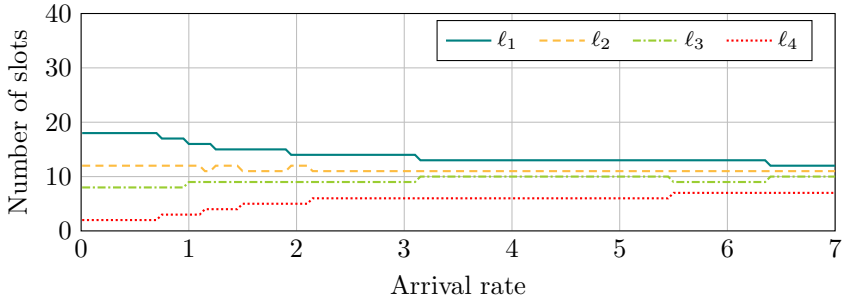

Figure 7: Optimal buffer length of each server (to minimize the loss probability) in a four-server cluster, as function of λ, with $L = 40$, $\mu_1 = 0.45$, $\mu_2 = 0.3$, $\mu_3 = 0.2$, $\mu_4 = 0.05$.

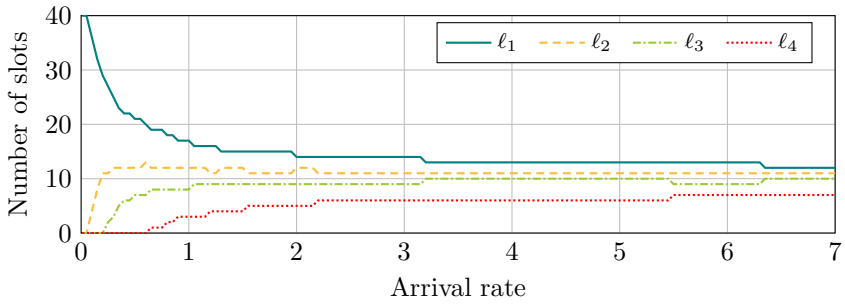

Figure 8: Optimal buffer length of each server (to minimize the mean response time) in the same setting as in Figure 7.

9 Conclusion

In this paper, we considered a load-balancing algorithm that leads to a product-form stationary distribution in clusters of servers with unequal service speeds. We developed analytical methods to understand the joint impact of the speeds and buffer lengths of the servers on performance. For a two-server cluster with arbitrary service speeds, we proved analytically that the optimal slot allocation in terms of the loss probability evolves monotonically from proportional to the server speeds to uniform as the arrival rate increases. We then generalized these results to clusters of more than two servers and assessed their validity on numerical examples.

For the future works, we would like to generalize these analytical results to the other performance metrics mentioned in the introduction. The main difficulty is that their expressions involve fractions of two sums that both span all states. We would also like to generalize these results in other directions, for instance by considering open variants of the algorithm or by accounting for assignment constraints [7, 21].

Acknowledgments

The research of Mark van der Boor is supported by the NWO Gravitation Networks grant 024.002.003. The authors thank Sem Borst for his valuable comments on an earlier draft of the paper, and in particular for suggesting the generalization of Theorem 1 to clusters with more than two servers. The authors are also grateful to Mor Harchol-Balter for an insightful discussion on Theorems 1 and 2.

References

[1] Ivo Adan and Gideon Weiss. A loss system with skill-based servers under assign to longest idle server policy. *Probability in the Engineering and Informational Sciences*, 26(3):307–321, 2012.

[2] Urtzi Ayesta, Tejas Bodas, and Ina Maria Verloop. On a unifying product form framework for redundancy models. *Perform. Evaluation*, 127-128:93–119, 2018.

[3] Thomas Bonald, Matthieu Jonckheere, and Alexandre Proutière. Insensitive load balancing. In *Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '04/Performance '04, pages 367–377, 2004.

[4] Mark van der Boor, Sem C. Borst, and Johan S. H. van Leeuwaarden. Load balancing in large-scale systems with multiple dispatchers. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pages 1–9, 2017.

[5] Mark van der Boor, Sem C. Borst, Johan S. H. van Leeuwaarden, and Debankur Mukherjee. Scalable load balancing in networked systems: A survey of recent advances. *arXiv:1806.05444 [cs, math]*, 2018.

[6] Jeffrey P. Buzen. Computational algorithms for closed queueing networks with exponential servers. *Commun. ACM*, 16(9):527–531, 1973.

[7] Céline Comte. Dynamic load balancing with tokens. *Computer Commun.*, 144:76–88, 2019.

[8] Céline Comte. Resource management in computer clusters: algorithm design and performance analysis, 2019. Ph.D. Thesis.

[9] Agner Krarup Erlang. Solution of some problems in the theory of probabilities of significance in automatic telephone exchanges. *Post Office Electrical Engineer's Journal*, 10:189–197, 1917.

[10] Kristen Gardner, Jazeem Abdul Jaleel, Alexander Wickeham, and Sherwin Doroudi. Scalable load balancing in the presence of heterogeneous servers. *Perform. Evaluation*, page 102151, 2020.

[11] Kristen Gardner and Ronda Righter. Product forms for FCFS queueing models with arbitrary server-job compatibilities: an overview. *Queueing Syst.*, 96:3–51, 2010.

[12] Diego Goldsztajn, Sem C. Borst, Johan S. H. van Leeuwaarden, Debankur Mukherjee, and Philip A. Whiting. Self-learning threshold-based load balancing. *arXiv:2010.15525 [cs, math]*, 2020.

[13] Varun Gupta and Neil Walton. Load balancing in the nondegenerate slowdown regime. *Operations Research*, 67(1):281–294, 2019.

[14] Arie Harel. Convexity results for the erlang delay and loss formulae when the server utilization is held constant. *Operations Research*, 59(6):1420–1426, 2011.

[15] Illés Antal Horváth, Ziv Scully, and Benny Van Houdt. Mean field analysis of join-below-threshold load balancing for resource sharing servers. *Proc. ACM Meas. Anal. Comput. Syst.*, 3(3):57:1–57:21, 2019.

[16] Matthieu Jonckheere and Jean Mairesse. Towards an erlang formula for multiclass networks. *Queueing Syst.*, 66(1):53–78, 2010.

[17] Matthieu Jonckheere and Balakrishna J. Prabhu. Asymptotics of insensitive load balancing and blocking phases. *Queueing Syst.*, 88(3):243–278, 2018.

[18] Yi Lu, Qiaomin Xie, Gabriel Kliot, Alan Geller, James R. Larus, and Albert Greenberg. Join-idle-queue: A novel load balancing algorithm for dynamically scalable web services. *Perform. Evaluation*, 68(11):1056–1071, 2011.

[19] Michael David Mitzenmacher. The power of two choices in randomized load balancing. *IEEE Trans. Parallel Distrib. Syst*, 12(10):1094–1104, 2001.

[20] Richard Serfozo. *Introduction to Stochastic Networks*. Stochastic Modelling and Applied Probability. Springer-Verlag, 1999.

[21] Wentao Weng, Xingyu Zhou, and R. Srikant. Optimal load balancing in bipartite graphs. *arXiv:2008.08830 [cs, math]*, 2020.

[22] Ronald W. Wolff. Poisson arrivals see time averages. *Operations Research*, 30(2):223–231, 1982.

[23] Xingyu Zhou, Jian Tan, and Ness Shroff. Flexible load balancing with multi-dimensional state-space collapse: Throughput and heavy-traffic delay optimality. *Perform. Evaluation*, 127-128:176–193, 2018.

[24] Xingyu Zhou, Jian Tan, and Ness Shroff. Heavy-traffic delay optimality in pull-based load balancing systems: Necessary and sufficient conditions. *Proc. ACM Meas. Anal. Comput. Syst.*, 2(3):41, 2018.

[25] Xingyu Zhou, Fei Wu, Jian Tan, Yin Sun, and Ness Shroff. Designing low-complexity heavy-traffic delay-optimal load balancing schemes: Theory to algorithms. *Proc. ACM Meas. Anal. Comput. Syst.*, 1(2):39:1–39:30, 2017.

Appendix

In the following two proofs, the arrival λ will be mentioned explicitly as a parameter of the quantities involved, so that in particular we will let $G(\lambda, \ell)$ denote the normalization constant under arrival rate λ and buffer length vector ℓ, and $\delta G(\lambda, \ell) = G(\lambda, \ell + e_1 - e_2) - G(\lambda, \ell)$. Also, it will be convenient to rewrite the normalization constant $G(\lambda, \ell)$ as

$$G(\lambda, \ell) = \sum_{x \leq \ell} \bar{\pi}(x), \quad \ell \in \mathbb{N}^2, \tag{9}$$

where $\bar{\pi}$ is a stationary measure such that $\bar{\pi}(0) = 1$, that is,

$$\bar{\pi}(x) = \binom{x_1 + x_2}{x_1} \left(\frac{\mu_1}{\lambda}\right)^{x_1} \left(\frac{\mu_2}{\lambda}\right)^{x_2}, \quad x \in \mathbb{N}^2. \tag{10}$$

Injecting this definition into the definition of $\delta G(\ell)$ yields

$$\delta G(\lambda, \ell) = \sum_{x_2=0}^{\ell_2 - 1} \bar{\pi}(\ell_1 + 1, x_2) - \sum_{x_1=0}^{\ell_1} \bar{\pi}(x_1, \ell_2). \tag{11}$$

Appendices A and B give the proofs of Propositions 4 and 5, respectively.

A Proof of Proposition 4

Proposition 4. *Let $\lambda > 0$ and $\ell \in \mathbb{N}^2$ such that $\ell_1 \geq 1$ and $\ell_2 \geq 1$. If $\beta(\lambda, \ell + e_1 - e_2) \leq \beta(\lambda, \ell)$, then $\beta(\lambda, \ell - xe_1 + xe_2) < \beta(\lambda, \ell - (x+1)e_1 + (x+1)e_2)$ for each $x \in \{0, 1, 2, \ldots, \ell_1 - 1\}$.*

Consider an arrival rate $\lambda > 0$. We will prove that, for each $\ell \in \mathbb{N}^2$ such that $\ell_1 \geq 1$ and $\ell_2 \geq 1$, $\beta(\lambda, \ell + e_1 - e_2) \leq \beta(\lambda, \ell)$ implies $\beta(\lambda, \ell) < \beta(\lambda, \ell - e_1 + e_2)$, that is, $\delta G(\lambda, \ell) > 0$ implies $\delta G(\lambda, \ell - e_1 + e_2) > 0$. Proposition 4 then follows from an induction argument that we omit. The following lemma, which follows directly from (10), will be useful.

Lemma 6. *For each $x \in \mathbb{N}^2$ such that $x_2 \geq 1$, we have*

$$\bar{\pi}(x) = \frac{x_1 + 1}{x_2} \frac{\mu_2}{\mu_1} \bar{\pi}(x + e_1 - e_2).$$

Let $\ell \in \mathbb{N}^2$ such that $\ell_1 \geq 1$, $\ell_2 \geq 1$, and $\delta G(\lambda, \ell) > 0$. By applying (11) to $k = \ell - e_1 + e_2$ and removing a (positive) term from the first sum, we obtain

$$\delta G(\lambda, k) > \sum_{x_2=1}^{\ell_2} \bar{\pi}(\ell_1, x_2) - \sum_{x_1=0}^{\ell_1 - 1} \bar{\pi}(x_1, \ell_2 + 1).$$

Applying Lemma 6 to each term, using the fact that $x_2 < \ell_2 + 1$ in the first sum and $x_1 < \ell_1$ in the second sum, and making a change of variable yields

$$\frac{\ell_2 + 1}{\ell_1 + 1} \frac{\mu_1}{\mu_2} \delta G(\lambda, k) > \sum_{y_2=0}^{\ell_2 - 1} \bar{\pi}(\ell_1 + 1, y_2) - \sum_{y_1=1}^{\ell_1} \bar{\pi}(y_1, \ell_2).$$

Lastly, we add a (positive) term in the last sum and apply (11), so as to obtain

$$\frac{\ell_2 + 1}{\ell_1 + 1} \frac{\mu_1}{\mu_2} \delta G(\lambda, k) > \delta G(\ell).$$

Therefore, $\delta G(\lambda, \ell) > 0$ implies that $\delta G(\lambda, k) > 0$.

B Proof of Proposition 5

Proposition 5. *Let $\lambda_* > 0$ and $\ell \in \mathbb{N}^2$ such that $\ell_2 \geq 1$. If $\beta(\lambda_*, \ell+e_1-e_2) < \beta(\lambda_*, \ell)$, then $\beta(\lambda, \ell+e_1-e_2) < \beta(\lambda, \ell)$ for each $\lambda \in (0, \lambda_*)$.*

We will prove equivalently that, for each $\lambda_* > 0$ and $\ell \in \mathbb{N}^2$ such that $\ell_2 \geq 1$, $\delta G(\lambda_*, \ell) > 0$ implies $\delta G(\lambda, \ell) > 0$ for each $\lambda \in (0, \lambda_*)$. The following lemma will be useful.

Lemma 7. *Let $\ell \in \mathbb{N}^2$ such that $\ell_2 \geq 1$. There is a sequence $\{c_n\}_{n \in \{\min(\ell_1+1, \ell_2), \ldots, \ell_1+\ell_2\}}$ such that*

$$\delta G(\lambda, \ell) = \sum_{n=\min(\ell_1+1, \ell_2)}^{\ell_1+\ell_2} c_n \frac{1}{\lambda^n}, \quad \lambda > 0. \tag{12}$$

The sequence $\{c_n\}_{n \in \{\min(\ell_1+1, \ell_2), \ldots, \ell_1+\ell_2\}}$ depends on the buffer lengths ℓ_1 and ℓ_2 and service rates μ_1 and μ_2 but not on the arrival rate λ, and satisfies one of the following conditions:

(1) $c_n > 0$ for each $n \in \{\min(\ell_1 + 1, \ell_2), \ldots, \ell_1 + \ell_2\}$,

(2) $c_n < 0$ for each $n \in \{\min(\ell_1 + 1, \ell_2), \ldots, \ell_1 + \ell_2\}$, or

(3) there is $n^* \in \{\min(\ell_1 + 1, \ell_2), \ldots, \ell_1 + \ell_2\}$ such that $c_n < 0$ for each $n \in \{\min(\ell_1 + 1, \ell_2), \ldots, n^* - 1\}$, $c_{n^*} \geq 0$, and $c_n > 0$ for each $n \in \{n^* + 1, \ldots, \ell_1 + \ell_2\}$.

Proof of the lemma. As in the proof of Theorem 2, we use (4) and distinguish two cases depending on the values of ℓ_1 and ℓ_2.

Case 1 ($\ell_1 + 1 \leq \ell_2$) We can rewrite (4) as (12), where the sequence $\{c_n\}_{n \in \{\min(\ell_1+1, \ell_2), \ldots, \ell_1+\ell_2\}}$ is given by (13a) in Figure 9. It follows immediately that $c_n > 0$ for each $n \in \{\ell_1 + 1, \ldots, \ell_2 - 1\}$. Additionally, we verify that $c_n > 0$ for each $n \in \{\ell_2, \ldots, \ell_1 + \ell_2\}$ by calculating the ratio between the first and second term of the subtraction and showing, thanks to the assumption $\mu_1 > \mu_2$, that this ratio is (strictly) larger than one. Therefore, the sequence satisfies Condition (1).

Case 2 ($\ell_1 \geq \ell_2$) We can rewrite (4) as (12), where the sequence $\{c_n\}_{n \in \{\min(\ell_1+1, \ell_2), \ldots, \ell_1+\ell_2\}}$ is given by (13b) in Figure 9. It follows directly that $c_n < 0$ for $n \in \{\ell_2, \ldots, \ell_1\}$. For $n \in \{\ell_1 + 1, \ldots, \ell_1 + \ell_2\}$, c_n is of the same sign as g_n, where

$$g_n = \log \left(\frac{\binom{n}{\ell_1+1} {\mu_1}^{\ell_1+1} {\mu_2}^{n-\ell_1-1}}{\binom{n}{\ell_2} {\mu_1}^{n-\ell_2} {\mu_2}^{\ell_2}} \right).$$

We conclude that the sequence $\{c_n\}_{n \in \{\min(\ell_1+1, \ell_2), \ldots, \ell_1+\ell_2\}}$ satisfies Conditions (2) or (3) by observing that $g_{\ell_1+\ell_2+1} = 0$, and then showing that the sequence $\{g_n\}_{n \in \{\ell_1+1, \ldots, \ell_1+\ell_2+1\}}$ is strictly concave in the sense that $g_n - g_{n+1} > g_{n+1} - g_{n+2}$ for each $n \in \{\ell_1 + 1, \ldots, \ell_1 + \ell_2 - 1\}$. The details of the calculation are omitted due to space constraints.

$$c_n = \begin{cases} \binom{n}{\ell_1 + 1} {\mu_1}^{\ell_1+1} {\mu_2}^{n-\ell_1-1}, & n \in \{\ell_1 + 1, \ldots, \ell_2 - 1\}, \\ \binom{n}{\ell_1 + 1} {\mu_1}^{\ell_1+1} {\mu_2}^{n-\ell_1-1} - \binom{n}{\ell_2} {\mu_1}^{n-\ell_2} {\mu_2}^{\ell_2}, & n \in \{\ell_2, \ldots, \ell_1 + \ell_2\}. \end{cases} \tag{13a}$$

$$c_n = \begin{cases} -\binom{n}{\ell_2} {\mu_1}^{n-\ell_2} {\mu_2}^{\ell_2}, & n \in \{\ell_2, \ldots, \ell_1\}, \\ \binom{n}{\ell_1 + 1} {\mu_1}^{\ell_1+1} {\mu_2}^{n-\ell_1-1} - \binom{n}{\ell_2} {\mu_1}^{n-\ell_2} {\mu_2}^{\ell_2}, & n \in \{\ell_1 + 1, \ldots, \ell_1 + \ell_2\}. \end{cases} \tag{13b}$$

Figure 9: Definitions of the sequence $\{c_n\}_{n \in \{\min(\ell_1+1, \ell_2), \ldots, \ell_1+\ell_2\}}$ depending on the value of $\ell = (\ell_1, \ell_2)$.

Conclusion The sequence $\{c_n\}_{n\in\{\min(\ell_1+1,\ell_2),\dots,\ell_1+\ell_2\}}$ satisfies Conditions (1), (2), or (3) of the proposition. □

We now prove Proposition 5. Let $\ell \in \mathbb{N}^2$ such that $\ell_2 \geq 1$. Using notation from Lemma 7, we distinguish two cases. If the sequence $\{c_n\}_{n\in\{\min(\ell_1+1,\ell_2),\dots,\ell_1+\ell_2\}}$ satisfies Conditions (1) or (2), the result is immediate because the sign of $\delta G(\lambda,\ell)$ does not depend on λ. Now assume that the sequence $\{c_n\}_{n\in\{\min(\ell_1+1,\ell_2),\dots,\ell_1+\ell_2\}}$ satisfies Condition (3). If $n^* = \ell_1+\ell_2$ and $c_{n^*} = 0$, the conclusion is the same as under Condition (2). Otherwise, we study the variations of the function $f : \lambda \mapsto \delta G(\lambda,\ell)$ on $(0,+\infty)$. This function is infinitely differentiable on this interval and, for each $\lambda > 0$, we have

$$\begin{aligned}
f'(\lambda) &= -\sum_{n=\min(\ell_1+1,\ell_2)}^{\ell_1+\ell_2} nc_n \frac{1}{\lambda^{n+1}},\\
&= -\sum_{n=\min(\ell_1+1,\ell_2)}^{n^*-1} nc_n \frac{1}{\lambda^{n+1}} - \sum_{n=n^*}^{\ell_1+\ell_2} (n^*-1)c_n \frac{1}{\lambda^{n+1}} - \sum_{n=n^*}^{\ell_1+\ell_2} (n-(n^*-1))c_n \frac{1}{\lambda^{n+1}},\\
&\leq -(n^*-1)\left(\sum_{n=\min(\ell_1+1,\ell_2)}^{\ell_1+\ell_2} c_n \frac{1}{\lambda^{n+1}}\right) - \sum_{n=n^*}^{\ell_1+\ell_2} (n-(n^*-1))c_n \frac{1}{\lambda^{n+1}},\\
&= -(n^*-1)f(\lambda) - \sum_{n=n^*}^{\ell_1+\ell_2} (n-(n^*-1))c_n \frac{1}{\lambda^{n+1}},\\
&< -(n^*-1)f(\lambda).
\end{aligned}$$

Hence, if there is $\lambda > 0$ such that $f(\lambda) \leq 0$, then $f'(\lambda) < 0$. Lemma 7 also implies $\lim_{\lambda\to 0^+} f(\lambda) = +\infty$. Therefore, either f is positive on $(0,+\infty)$, or there is $\lambda_0 > 0$ such that f is positive on $(0,\lambda_0)$ and negative on $(\lambda_0,+\infty)$, with $f(\lambda_0) = 0$. Both cases lead to the conclusion.