Value Driven Load Balancing

Sherwin Doroudi[a], Esa Hyytiä[b,1], Mor Harchol-Balter[c,2]

[a]*Tepper School of Business, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213*
[b]*Department of Communications and Networking, Aalto University, 00076 Aalto, Finland*
[c]*School of Computer Science, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213*

Abstract

To date, the study of dispatching or load balancing in server farms has primarily focused on the minimization of response time. Server farms are typically modeled by a front-end router that employs a dispatching policy to route jobs to one of several servers, with each server scheduling all the jobs in its queue via Processor-Sharing. However, the common assumption has been that all jobs are equally important or valuable, in that they are equally sensitive to delay. Our work departs from this assumption: we model each arrival as having a randomly distributed value parameter, independent of the arrival's service requirement (job size). Given such value heterogeneity, the correct metric is no longer the minimization or response time, but rather, the minimization of value-weighted response time. In this context, we ask "what is a good dispatching policy to minimize the value-weighted response time metric?" We propose a number of new dispatching policies that are motivated by the goal of minimizing the value-weighted response time. Via a combination of exact analysis, asymptotic analysis, and simulation, we are able to deduce many unexpected results regarding dispatching.

Keywords: task assignment, dispatching, server farms, processor-sharing, heterogenous values, holding cost, valuations, c-mu rule

1. Introduction

Server farms are commonplace today in web servers, data centers, and in compute clusters. Such architectures are inexpensive (compared to a single fast server) and afford flexibility and scalability in computational power. However, their efficiency relies on having a good algorithm for routing incoming jobs to servers.

A typical server farm consists of a front-end router, which receives all the incoming jobs and dispatches each job to one of a collection of servers which do the actual processing, as depicted in Figure 1. The servers themselves are "off-the-shelf" commodity servers which typically schedule all jobs in their queue via Processor-Sharing (PS); this cannot easily be changed to some other scheduling policy. All the decision-making is done at the central dispatcher. The dispatcher (also called a load balancer) employs a *dispatching* policy (often called a *load balancing* policy or a *task assignment policy*), which specifies to which server an incoming request should be routed. Each incoming job is *immediately* dispatched by the dispatcher to one of the servers (this immediate dispatching is important because it allows the server to quickly set up a connection with the client, before the connection request is dropped). Typical dispatchers used include Cisco's Local Director [1], IBM's Network Dispatcher [2], F5's Big IP [3], Microsoft Sharepoint [4], etc. Since scheduling at the servers is not under our control, it is extremely important that the right dispatching policy is used.

Prior work has studied dispatching policies with the goal of minimizing mean *response time*, $\mathbb{E}[T]$; a job's response time is the time from when the job arrives until it completes. Several papers have specifically studied the case where

Email addresses: sdoroudi@andrew.cmu.edu (Sherwin Doroudi), esa.hyytia@aalto.fi (Esa Hyytiä), harchol@cs.cmu.edu (Mor Harchol-Balter)

[1]This work has been supported by the Academy of Finland in TOP-Energy project (grant no. 268992).
[2]This work was funded by NSF-CMMI-1334194 as well as a Computational Thinking grant from Microsoft Research.

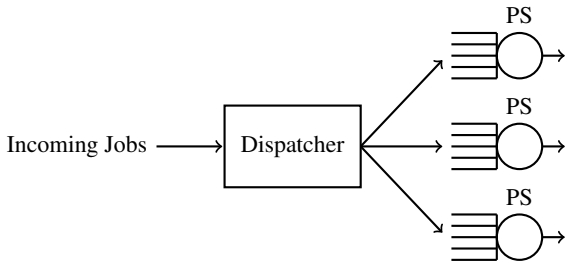


Figure 1: Dispatching in server farms with Processor-Sharing (PS) servers.

the servers schedule their jobs via PS (see [5–12]). Here, it has been show that the Join-the-Shortest-Queue (JSQ) policy performs very well, for general job size distributions. Even picking the shortest of a small subset of the queues, or simply trying to pick an idle queue if it exists, works very well. Interestingly, such simple policies like JSQ are superior even to policies like Least-Work-Left, which route a job to the server with the least remaining total work (sum of remaining sizes of all jobs at the queue), rather than simply looking at the number of jobs [13]. In addition, there have been many more papers studying dispatching policies where the servers schedule jobs in First-Come-First-Served (FCFS) order (see e.g., [9, 14–25]). Here high job size variability can play a large role, and policies like Size-Interval-Task-Assignment (SITA) [14], which segregates jobs based on job size, or Least-Work-Left [26], which routes job to the queue with the least total remaining work (rather than the smallest number of jobs), are far superior to JSQ.

However, all of this prior work has assumed that jobs have equal importance (value), in that they are equally sensitive to delay. This is not at all the case. Some jobs might be background jobs, which are largely insensitive to delay, while others have a live user waiting for the result of the computation. There may be other jobs that are even more important in that *many* users depend on their results, or other jobs depend on their completion. We assume that every job has a value, V, independent of its size (service requirement). Given jobs with heterogeneous values, the right metric to minimize is not the mean response time, $\mathbb{E}[T]$, but rather the mean *value-weighed response time*, $\mathbb{E}[VT]$, where jobs of higher value (importance) are given lower response times.

The problem of minimizing $\mathbb{E}[VT]$, where V and T are independent, is also not new, although it has almost exclusively been considered in the case of server scheduling, *not in the case of dispatching* (see Prior Work section). Specifically, there is a large body of work in the operations research community where jobs have a *holding cost*, c, independent of the job size, and the goal is to minimizing $\mathbb{E}[c \cdot T]$ over all jobs. Here it is well-known that the $c\mu$ rule is optimal [27]. In the $c\mu$ rule, c refers to a job's holding cost and μ is the reciprocal of a job's size. The $c\mu$ rule always runs the job with the highest product c times μ; thus, jobs with high holding cost and/or small size are favored. However, there has been no $c\mu$-like dispatching policy proposed for server farms.

In this paper, we assume a server farm with a dispatcher and PS servers. Jobs arrive according to a Poisson process and are immediately dispatched to a server. The value, V, of an arrival is known, but its size, S, is not known. Furthermore, we assume that value and size are independent, so that knowing the job's value does not give us information about the job's size. We assume that we know the distribution job values. Furthermore, job sizes are exponentially-distributed with unit mean. By requiring that jobs are exponentially distributed, we are consistent with the assumption that there is no way to estimate a job's size; otherwise, we could use "age" information to update predictions on the remaining size of each job, and some of the policies of interest would become much more complex.[3] Nothing else is known about future arrivals. In making dispatching decisions, we assume that we know the *queue length* at each server (this is the number of jobs at the PS server) as well as the values of the jobs at each server. In this context, we ask:

"What is a good dispatching policy to minimize $\mathbb{E}[VT]$?"

[3]We do in fact carry out a set of simulations assuming an alternative job size distribution, with policies that ignore "age" information. The qualitative results remain the same as those under exponentially distributed job sizes; see Section 5.

Even in this simple setting, it is not at all obvious what makes a good dispatching policy. We consider several policies (see Section 4 for more detail):

- The **Random (RND)** dispatching policy ignores job values and queue lengths. Arrivals are dispatched randomly.
- The **Join-Shortest-Queue (JSQ)** dispatching policy ignores values and routes each job to the server with the fewest number of jobs. This policy is known to be optimal in the case where all values are equal [5].
- The **Value-Interval-Task-Assignment (VITA)** dispatching policy is reminiscent of the SITA policy, where this time jobs are segregated by *value*, with low-value jobs going to one server, medium value jobs going to the next server, higher-value jobs going to the next server, and so on. The goal of this policy is to isolate high value jobs from other jobs, so that the high value jobs can experience low delay. The distribution of V and system load ρ are used to determine the optimal threshold(s) for minimizing $\mathbb{E}[VT]$.
- The **C-MU** dispatching policy is motivated by the $c\mu$ rule for scheduling in servers. Each arrival is dispatched so as to maximize the average instantaneous value of the jobs completing, assuming no future arrivals, where the average is taken over the servers. This policy makes use of the value of the arrival and the values of all the jobs at each server.
- The **Length-And-Value-Aware (LAVA)** dispatching policy is very similar to the C-MU policy. Both policies incorporate queue length and job values in their decision. However, whereas C-MU places jobs so as to maximize the expected instantaneous value of jobs completed, LAVA places jobs so as to explicitly minimize $\mathbb{E}[VT]$ over jobs. Both policies make their decisions solely based on jobs already in the system.

This paper is the first to introduce the VITA, C-MU, and LAVA policies.

Via a combination of asymptotic analysis, exact analysis, and simulation we show the following in Sections 5 and 6. We find that generally RND is worse than VITA, which is worse than JSQ, which is worse than LAVA. In fact, under an asymptotic regime we prove that as system load $\rho \to 1$, the ratio $\mathbb{E}[VT]^{\text{RND}} : \mathbb{E}[VT]^{\text{VITA}} : \mathbb{E}[VT]^{\text{JSQ}} : \mathbb{E}[VT]^{\text{LAVA}}$ approaches $4 : 2 : 2 : 1$. The C-MU policy, on the other hand, avoids neat classification. There are value distributions and loads for which C-MU is the best policy of those we study, and others for which C-MU is the worst. In fact, C-MU can become unstable even when system load $\rho < 1$. Finally, while VITA is generally not a great policy, we find that there are certain regimes under which VITA approaches optimality under light load ($\rho < 1/2$), performing far better than the other policies we study.

But is it possible to do even better than the above dispatching policies? We find that under a particularly skewed value distribution, there is a policy, "Gated VITA," which can outperform all of the aforementioned policies by an *arbitrary* factor. The idea behind this policy is to split high and low value jobs, while using a "gate" to place a limit on the number of low-value jobs that can interfere with high-value jobs (see Section 7 for details). If one is willing to forego simplicity in the dispatching policies, one can further use first policy iteration to significantly improve upon simple policies (see Section 8 for details).

2. Prior work on value-driven dispatching

The problem of finding dispatching policies with the aim of minimizing value-weighted response time has received very little attention in the literature. Below we discuss the few papers in this setting, which are (only tangentially) related to our own.

One paper concerned with the minimization of an $\mathbb{E}[VT]$-like metric is [7], where a constant value parameter is associated with each *server*. In this setting, job values are not treated as exogenous random variables determined at the time of arrival; instead, the value of a job is set to the value associated with the server serving the job, and hence, a job's value is determined by where the dispatcher sends it.

Another research stream that considers heterogeneity in the delay sensitivity of jobs is the dispatching literature concerned with minimizing *slowdown*, $\mathbb{E}[\frac{1}{X} \cdot T]$, where X is a job's service requirement (size) [28–30]. This body of literature differs from our work in two key ways. First, unlike our work, the "value" of each job is deterministically

related to (in particular, is the multiplicative inverse of), rather than independent of, the job's size. Second, the slowdown metric necessitates the examination of dispatching policies that can observe job sizes.

Finally, settings similar to ours are considered in [31, 32]. Unlike our paper, however, these papers do not provide a comprehensive comparison of dispatching policies: [31] is concerned with deriving one specific policy (the lookahead policy), while [32] only considers the simple random and round robin policies, together with FPI improvements on these policies, which make use of job sizes.

3. Model for PS server system

The basic system, illustrated in Fig. 1, is as follows:

- We have m servers with Processor-Sharing (PS) scheduling discipline and service rate μ_i. Throughout the simulation and analytic portion of the paper, we give particular attention to the case where $m = 2$ and $\mu_1 = \mu_2$.
- Jobs arrive according to the Poisson process with rate λ and are immediately dispatched to one of the m servers.
- Job j is defined by a pair $(X^{(j)}, V^{(j)})$, where $X^{(j)}$ denotes the size of the job and $V^{(j)}$ is its value.
- Job sizes obey exponential distribution with unit mean, $\mathbb{E}[X] = 1$, preventing us from using the "age" of a job to learn about its remaining size.
- The system load is given by $\rho \equiv \lambda/(\sum_{i=1}^{m} \mu_i)$. When $m = 2$ and $\mu_1 = \mu_2$, we have $\rho = \lambda/(2\mu)$.
- We can observe the number of jobs in each server (queue length), but not their service times.
- The values $\{V^{(j)}\}$ are drawn from a known distribution with finite mean and nonzero variance. A job's value becomes known upon arrival. We can also observe the values of jobs at each server.
- Jobs are i.i.d., i.e., $(X^{(j)}, V^{(j)}) \sim (X, V)$, where $X^{(j)}$ and $V^{(j)}$ are independent. In particular, it is not possible to deduce anything about a job's size based on its value.
- The objective is to minimize the mean (or time-average) value-weighted response time, given by $\mathbb{E}[VT] \equiv \lim_{n\to\infty} \left(\frac{1}{n} \sum_{j=1}^{n} V^{(j)} T^{(j)}\right)$, where $T^{(j)}$ is the response time experienced by job j.

Notation: Throughout, it will be convenient to use n_i to denote the number of jobs that an arrival sees at server i; $v_{i,j}$ to be the value of the jth job at server i; $v_i^{\text{sum}} \equiv \sum_{j=1}^{n_i} v_{i,j}$ to denote the total values of jobs that an arrival sees at server i; and $\bar{v}_i \equiv v_i^{\text{sum}}/n_i$ to denote the average value of jobs at server i.

4. Description of simple dispatching policies

In describing our dispatching policies, it will be convenient to use the following terms.

Definition 1. *The* **state** *of a queue consists of its queue length and the specific values of jobs at the queue.*

Definition 2. *A dispatching policy is called* **static** *if its decision is independent of the queue states and independent of all past placement of jobs.*[4]

Definition 3. *A dispatching policy is called* **value-aware** *if the policy requires knowing the value of a new arrival.*

4.1. Random dispatching (RND)

The RND policy dispatches each incoming job to server i with probability $1/m$, where m is the number of servers.

[4]Note that a policy such as Round-Robin is not considered **static** in this paper. The placement of a job in the Round-Robin policy is determined by the placement of the previous job: if the Round-Robin policy sends an arrival to server j, then it sends the next arrival to server $j + 1 \mod m$. In particular, a static policy ensures that the arrival process to each server is a Poisson process.

4.2. Join-the-Shortest-Queue dispatching (JSQ)

The JSQ policy dispatches each incoming job to the server with the shortest queue length. If multiple queues have the same shortest length, JSQ picks among them at random. Like RND, JSQ does not make use of the value a job. JSQ is typically superior to RND in that it balances the *instantaneous* queue lengths. It is known to be either optimal or very good for minimizing $\mathbb{E}[T]$ in a variety of settings [5]. Observe that $\mathbb{E}[T] = \mathbb{E}[VT]$ in the case where all values are equal.

4.3. Value-Interval-Task-Assignment (VITA)

The VITA policy is our first value-aware policy. The idea is that each server is assigned a "value interval" (e.g., "small," "medium," or "large" values), and an incoming job is dispatched to that server that is appropriate for its value. Specifically, assume that the value distribution has a continuous support without atomic probabilities, ranging from 0 to ∞. In this case, we can imagine specifying value "thresholds," $\xi_0, \xi_1, \ldots, \xi_m$, where $0 = \xi_0 < \xi_1 < \ldots < \xi_{m-1} < \xi_m = \infty$. Then VITA assigns jobs with value $V \in (\xi_{i-1}, \xi_i)$ to server i.[5] In the case where there is a nonzero probability mass associated with a particular value, v, it may be the case that jobs with value v are routed to a subset $n > 1$ of the m servers. In this case, we also must specify additional "thresholds" in the form of probabilities, $p_1, p_2, \cdots, p_n$, where $\sum_i p_i = 1$, and jobs of value v are routed to the ith server of the n with probability p_i.

Thus we can see that VITA may depend on various threshold parameters. *Throughout we define VITA to use those thresholds which minimize* $\mathbb{E}[VT]$. VITA is a static policy, and thus practical for distributed operation with any number of parallel dispatchers.

The intuition behind VITA is that it allows high-value jobs to have isolation from low-value jobs. Given that our goal is to provide high-value jobs with low response times, it makes sense to have some servers which serve exclusively higher-value jobs, so that these jobs are not slowed down by other jobs. Of course the optimal choice of thresholds depends on the value distribution and the load.

It turns out that VITA is the optimal static policy for minimizing $\mathbb{E}[VT]$. For clarity, we will prove that VITA is the optimal static value-aware policy in the case of $m = 2$ servers with identical service rates; however this result easily extends to $m > 2$ servers.

Proposition 1. *VITA is the optimal (i.e., $\mathbb{E}[VT]$-minimizing) static policy for any two-server system with identical service rates. Furthermore, VITA unbalances the load, whereas all load balancing static policies achieve the same performance as RND.*

Proof. Deferred to Appendix. □

4.4. C-MU

The classic $c\mu$ rule for scheduling in servers prioritizes jobs with the highest product of value (c) and inverse expected remaining service requirement (μ). This policy for server scheduling is known to be optimal in many scheduling contexts [27].

Our C-MU dispatching policy is inspired by the $c\mu$ scheduling rule, in that it aims to maximize the value-weighted departure rate.

As always, we use n_i to denote the number of jobs that an arrival sees at server i; $v_i^{\text{sum}} = \sum_{j=1}^{n_i} v_{i,j}$ for the sum of job values at server i; and $\bar{v}_i = v_i^{\text{sum}}/n_i$ to denote the average value of jobs at server i. Since PS scheduling provides all jobs equal service rate, $\bar{v}_i\mu_i$ denotes the current "rate of value departing" from server i. The *total rate of value departing* is of course $\sum_i \bar{v}_i\mu_i$.

The C-MU dispatching rule greedily routes each incoming job so as to maximize the instantaneous total rate of value departing. This policy is myopic in that it makes its routing decision solely based on jobs already in the system, not taking into account future arrivals or departures. Specifically, an incoming job of value v is routed to that server, i, whose rate of value departing will increase the most (or decrease least) by having the job. That is, i satisfying

$$\underset{i}{\text{argmax}} \ \frac{\mu_i}{n_i + 1}(v_i^{\text{sum}} + v) - \frac{\mu_i}{n_i} \cdot v_i^{\text{sum}} = \underset{i}{\text{argmax}} \ \frac{v - \bar{v}_i}{n_i + 1}\mu_i = \underset{i}{\text{argmin}} \ \frac{\bar{v}_i - v}{n_i + 1}\mu_i.$$

[5] Since it is preferable to send high-value jobs to wherever they can be served fastest, we assume without loss of generality that $\mu_1 \leq \cdots \leq \mu_m$.

Let $\alpha_{\text{C-MU}}(v)$ denote the server to which a job of value v is routed under C-MU dispatching. Then

$$\boxed{\alpha_{\text{C-MU}}(v) = \underset{i}{\operatorname{argmin}} \frac{\bar{v}_i - v}{n_i + 1}\mu_i.} \tag{1}$$

Note that if some value jobs are common, then $\bar{v}_i = v$ may occur frequently and the numerator in (1) becomes zero. In such cases C-MU is "clueless". For example, it cannot decide between a server with a billion jobs with value v and a server with a single job with value v. We use *random splitting* to resolve ties, but note that in some specific (asymptotic) cases significant improvements can be achieved if ties are resolved in some other manner, e.g., via JSQ.

As an example of C-MU, suppose that there are 2 jobs of value 10 at server 1 and 4 jobs of value 1 at server 2. Suppose also that $\mu_1 = \mu_2 = 1$. The current value-weighted departure rate at server 1 is 10, and that at server 2 is 1. Consider an incoming arrival of value v. If the arrival is routed to server 1, then it will increase the mean value-weighted departure rate at server 1 to $(20 + v)/3$. The total rate of value departing from the system will increase from 11 to $(23 + v)/3$. If, on the other hand, the arrival is routed to server 2, then it will increase the mean value-weighted departure rate at server 2 to $(4 + v)/5$. The total rate of value departing from the system will then increase from 11 to $(54 + v)/5$. Thus the new arrival should be routed to server 1 if $(23 + v)/3 > (54 + v)/5$; to server 2 if $(23 + v)/3 < (54 + v)/5$; and to each server with probability $1/2$ if $(23 + v)/3 = (54 + v)/5$. These cases correspond to v greater than, less than, and equal to $47/2$, respectively. Note that the value of the incoming job has to be very high (i.e., $47/2$ or greater) before C-MU is willing to send it to server 1.

4.5. Length-and-Value-Aware (LAVA)

Like the C-MU policy, LAVA is state-aware, using both the queue lengths and the values of all jobs at each server. Also like C-MU, LAVA is myopic with respect to further arrivals. The key difference is that LAVA attempts to directly minimize the metric of interest, $\mathbb{E}[VT]$, whereas C-MU aims to maximize the total rate of value departing, which is related to minimizing $\mathbb{E}[VT]$ but not the same.

LAVA routes an arriving job so as to minimize $\mathbb{E}[VT]$, averaged over all jobs currently in the system, including the current arrival, assuming that no future jobs arrive. Similar myopic policies have been considered in [5, 6] where values are homogeneous, and the goal is minimizing $\mathbb{E}[VT]$.

Before formally defining LAVA, it is useful to compute $\mathbb{E}[VT]$ for an arbitrary system state under the assumption that there will be no further arrivals. The jobs at a server are equally likely to leave in any order. Hence the sum of the response times of jobs currently at server i is $\frac{n_i(n_i+1)}{2} \cdot \frac{1}{\mu_i}$, and the expected response time for each job at server i is $\frac{n_i+1}{2\mu_i}$. Thus the mean value-weighted response time of the jth job at server i is $\frac{n_i+1}{2\mu_i} \cdot v_{i,j}$. Recall that $\mathbb{E}[VT]$ is a per-job average. Thus, if n is the total number of jobs in the system, (i.e., $n = \sum_{i=1}^{m} n_i$), then

$$\mathbb{E}[VT] = \frac{1}{n}\sum_{i=1}^{m}\sum_{j=1}^{n_i}\left\{\frac{n_i+1}{2\mu_i} \cdot v_{i,j}\right\} = \frac{1}{n}\sum_{i=1}^{m}\frac{n_i+1}{2\mu_i} \cdot v_i^{\text{sum}} = \sum_{i=1}^{m} S_i \qquad \text{where} \qquad S_i = \frac{n_i+1}{2\mu_i} \cdot v_i^{\text{sum}}.$$

We proceed to formally define LAVA. Given an arriving job with value v, we want to dispatch this job so as to minimize the resulting $\mathbb{E}[VT]$, as expressed above. That is, we want to send the arriving job to whichever server i will experience the least increase in S_i. If we send a job of value v to server i, then the expected response time of jobs at server i will increase from $(n_i + 1)/(2\mu_i)$ to $(n_i + 2)/(2\mu_i)$, resulting in a new value of S_i: $S_i^{new} = \frac{n_i+2}{2\mu_i}\left(v + v_i^{\text{sum}}\right)$. This means that

$$S_i^{new} - S_i = \frac{1}{2\mu_i}\left((n_i + 2)v + v_i^{\text{sum}}\right).$$

Let $\alpha_{\text{LAVA}}(v)$ denote the server to which a job of value v is routed under LAVA dispatching.

$$\boxed{\alpha_{\text{LAVA}}(v) = \underset{i}{\operatorname{argmin}} \frac{1}{2\mu_i}\left((n_i + 2)v + v_i^{\text{sum}}\right).} \tag{2}$$

With identical servers $\mu_i = \mu_j$, we can extract common constants and simplify the description of the policy to

$$\boxed{\alpha_{\text{LAVA}}(v) = \underset{i}{\operatorname{argmin}}\, n_i v + v_i^{\text{sum}}.} \tag{3}$$

As usual, ties are broken via random assignment.

Using this final formula (3), we return again to the example that we considered for C-MU, where there are 2 jobs of value 10 at server 1 and 4 jobs of value 1 at server 2. Under LAVA, a new arrival of value v prefers to go to server 1 if $2v + 2 \cdot 10 < 4v + 4 \cdot 1$, i.e., if $v > 8$. So a new arrival of value greater than 8 will join the value 10 jobs, while a new arrival of value less than 8 will join the value 1 jobs (a new arrival of value exactly 8 will pick randomly). Contrast this with C-MU, where the cutoff was $47/2$ rather than 8 for the same example.

5. Simulation results and intuitions

In this section, we report our findings from simulation in a two-server system (with identical service rates) for the value distributions given in Table 1. We also provide intuition for the results. Formal proofs will be given in the next section. In all cases, $\mathbb{E}[V] = 1$, and the continuous distributions (a)–(c) are presented in increasing order of variability, as are the discrete distributions (d)–(f). The variance is particularly high for distributions (e) and (f). Note that while the fraction of high-value jobs is the same in distributions (e) and (f), the *low-value* jobs contribute about 100 times as much to $\mathbb{E}[V]$ under (e) than under (f). Distribution (f) has a "sharply bimodal" form (to be defined in Section 6.3), whereby the high value jobs are extremely rare, yet comprise almost all of the value.

(a)	Uniform,	$V \sim \text{Uniform}(0, 2)$
(b)	Exponential,	$V \sim \text{Exp}(1)$
(c)	Bounded Pareto,	$V \sim \text{Pareto}(\text{min} = 0.188, \text{max} = 10\,000, \alpha = 1.2)$
(d)	Bimodal,	$V \sim \text{Bimodal}(99\%, 0.1; 1\%, 9.01)$
(e)	Bimodal,	$V \sim \text{Bimodal}(99.9\%, 0.1; 0.1\%, 900.1)$
(f)	Bimodal,	$V \sim \text{Bimodal}(99.9\%, 1/999; 0.1\%, 999)$

Table 1: Value distributions considered in the examples. $\mathbb{E}[V] = 1$ in all cases. Bimodal$(x\%, v_1; y\%, v_2)$ indicates that $x\%$ of jobs have value v_1 and $y\%$ of jobs have value v_2.

The simulation results are presented in Fig. 2, where for each of the distributions (a)–(f), we have plotted the performance of each policy, P, normalized by the performance of JSQ (i.e., $\mathbb{E}[VT]^{\text{P}}/\mathbb{E}[VT]^{\text{JSQ}}$), as a function of ρ. In all of these experiments, job sizes are exponentially distributed with mean 1. We ran additional simulations where (i) job sizes follow a Weibull distribution and (ii) job sizes follow a deterministic distribution. For these additional simulations, we continued to use the same implementation of LAVA and C-MU which do not attempt to learn a job's remaining service time. The results under these additional simulations were qualitatively similar to those under with exponentially distributed job sizes, suggesting that our findings are largely insensitive to the job-size distribution, as might be expected under Processor-Sharing service.

It is immediately apparent that RND is far worse than JSQ in all figures, and that $\mathbb{E}[VT]$ under RND converges to twice that under JSQ as load approaches 1. This factor 2 result will be proven in Proposition 2, however it is understandable since under high load the two servers under JSQ function similarly to a single server with twice the speed.

Given our result in Proposition 1 showing that VITA is the optimal static policy, it may seem surprising that VITA offers only a modest improvement over RND in Fig. 2 (a)–(d) and is so inferior to JSQ, despite JSQ not even using value information. To see what is going on, notice that under low load, there are only a few jobs in the system. Here VITA can mess up by putting these jobs onto the same server (if they have the same value), whereas JSQ never will. In fact dynamic policies like JSQ, CMU and LAVA are all "idle-eager," in that they will always route a job to an idle server if one exists. This gives them an advantage over VITA for low load. Under high load, VITA does not have much flexibility over protecting high-value jobs, since it is forced to balance load. Here, the performance under VITA is close to that under RND, even though VITA is the optimal static load balancing policy (cf. Proposition 1).

Fig. 2(f) is the exception. Here, under moderate load VITA outperforms all the other policies examined. The reason is that when nearly all of the value in the system is made up by a very small fraction of the jobs, VITA can maintain a much shorter queue for the most valuable jobs, without paying a big penalty for the resulting additional delays faced by the other jobs. This effect is particularly potent when $\rho < 1/2$, because the valuable jobs do not need to share the server "reserved" for them, since all low value jobs can be directed to the other server without violating

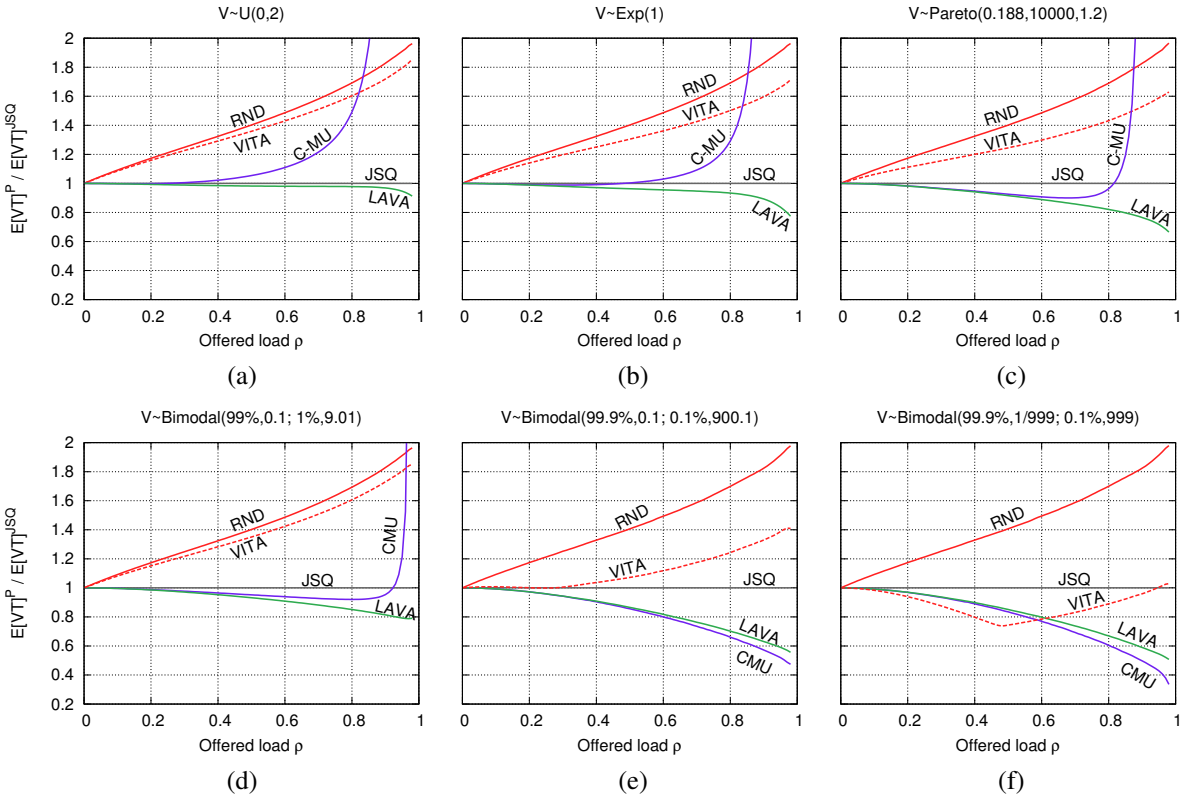


Figure 2: Performance relative to JSQ in a two-server system, with value distributions taken from Table 1 (a)–(f). Each point corresponds to the mean performance averaged over 100 million jobs.

stability constraints. Even under high load, we see that VITA performs similarly to JSQ, rather than RND. Although VITA has very limited flexibility for protecting high-value jobs under high load, the high value jobs under distribution (f) are *so extremely valuable* that even giving them a slight reduction in load (say 0.97 for the high value jobs and 0.99 for the low value jobs), will buy a factor of 2 improvement over RND. We formalize these notions in Theorems 2 and 3 of Section 6.

LAVA often outperforms all of the other policies. It is also the only policy that outperforms JSQ in all cases examined. Its consistent improvement over JSQ can be attributed to the fact that LAVA behaves like JSQ when all values are the same (or very close), but LAVA also uses value information to dispatch in favor of the most valuable jobs. Specifically, when an extremely valuable job enters the system, LAVA places the job somewhere and then essentially ceases sending jobs to that server (because it is crucial to the $\mathbb{E}[VT]$ metric that this job not be slowed down). Thus, this particularly valuable job ends up timesharing with an average of $n/2$ jobs over its lifetime under LAVA, where n denotes the number of jobs the high value job saw when it arrived. This "stopper" effect under LAVA is particularly important for sharply bimodal distributions like (f), where, as $\rho \to 1$, LAVA approximately obtains a 50% reduction in the $\mathbb{E}[VT]$ over JSQ. We formalize this result in Corollary 1.

Finally, turning to the C-MU policy, we notice that $\mathbb{E}[VT]$ diverges, suggesting that the system is unstable, for some values of $\rho < 1$ under distributions (a)–(d), while C-MU is the best performing policy under distributions (e) and (f). C-MU's good and bad performance can be attributed to the same protectiveness of high value jobs that we saw in LAVA. Specifically, the presence of a single high-value job at a server can prohibit the entry of any low-value jobs to that server. On the one hand, this is beneficial because the high-value jobs are protected; on the other hand, the server with the high-value job may be underutilized, giving rise to instability at the other server. While LAVA also exhibits such "stoppering" behavior, under C-MU this prohibition is more *severe* as it occurs *regardless of the length of the queue at the other server*. For value distributions (a)–(d), C-MU's extreme protectiveness of high-value jobs

leads to instability at the other server. For distributions (e) and (f), the high-value jobs are much rarer; so much so that the "stoppering" periods are a lot less frequent and instability does not occur. Observation 1 and Theorem 4 of Section 6 shed more light on C-MU's behavior.

6. Analytic results

Motivated by the observations in the previous section, we proceed to present and prove analytic results.

6.1. RND and JSQ under high load

We start by noting that V and T are independent under both RND and JSQ, yielding $\mathbb{E}[VT] = \mathbb{E}[V]\mathbb{E}[T]$. Using a result from [21] that $\mathbb{E}[T]^{\text{RND}} \geq \mathbb{E}[T]^{\text{JSQ}}$ for all ρ, it follows that $\mathbb{E}[VT]^{\text{RND}} \geq \mathbb{E}[VT]^{\text{JSQ}}$ for all ρ. The proposition below provides an asymptotic comparison of RND and JSQ as $\rho \to 1$.

Proposition 2. *As $\rho \to 1$, $\mathbb{E}[VT]^{\text{RND}}/\mathbb{E}[VT]^{\text{JSQ}} \to 2$.*

Proof. We start by recalling a result by Foschini and Salz [33], stating that as $\rho \to 1$, the response time for a two-server system under JSQ with servers operating at rate μ approaches that of a single-server operating at rate 2μ, i.e.,

$$\lim_{\rho \to 1} \frac{\mathbb{E}[T]^{\text{JSQ}}}{1/(2\mu - \lambda)} = 1.$$

Next, we use the fact that V and T are independent under both RND and JSQ, yielding

$$\mathbb{E}[VT]^{\text{RND}} = \mathbb{E}[V]\mathbb{E}[T]^{\text{RND}} = \frac{\mathbb{E}[V]}{\mu - \lambda/2} \quad \text{and} \quad \mathbb{E}[VT]^{\text{JSQ}} = \mathbb{E}[V]\mathbb{E}[T]^{\text{JSQ}},$$

and complete the proof by applying the result from [33]:

$$\lim_{\rho \to 1} \frac{\mathbb{E}[VT]^{\text{RND}}}{\mathbb{E}[VT]^{\text{JSQ}}} = \lim_{\rho \to 1} \left(\left(\frac{\mathbb{E}[V]}{\mu - \lambda/2} \right) \middle/ \left(\frac{\mathbb{E}[V]}{2\mu - \lambda} \right) \right) = \lim_{\rho \to 1} \left(\frac{2\mu - \lambda}{\mu - \lambda/2} \right) = 2. \qquad \square$$

6.2. Stability and instability

A dispatching policy is called **stable** if the resulting system has a finite response time (i.e., $\mathbb{E}[T] < \infty$), otherwise we say that it is unstable. A lack of stability can lead to infinitely poor performance with respect to $\mathbb{E}[VT]$. In fact, if the value distribution has positive lower and upper bounds, then $\mathbb{E}[VT]$ is finite if and only if $\mathbb{E}[T]$ is finite (i.e., if and only if the dispatching policy is stable). Hence, it is important to avoid implementing unstable policies. The following result shows that if the value distribution has positive lower and upper bounds, then all of the policies we have studied, with the exception of C-MU, are stable if and only if $\rho < 1$:

Theorem 1. *Let the value distribution have lower bound $a > 0$ and upper bound $b < \infty$. Then for a two-server system with identical service rates,* RND*,* JSQ*,* VITA*, and* LAVA *are stable if and only if $\rho < 1$.*

Proof. First we note that all dispatching policies are unstable when $\rho \geq 1$.

Clearly, $\mathbb{E}[T]^{\text{RND}} = 1/(\mu - \lambda/2)$, so this policy is stable whenever $\mu > \lambda/2$, or equivalently, whenever $\rho < 1$, as required. Next, recall that $\mathbb{E}[T]^{\text{JSQ}} \leq \mathbb{E}[T]^{\text{RND}} < \infty$ by [21], establishing that JSQ is also stable. Since VITA is the optimal static policy by Proposition 1, $\mathbb{E}[VT]^{\text{VITA}} \leq \mathbb{E}[VT]^{\text{RND}} = \mathbb{E}[V]\mathbb{E}[T]^{\text{RND}} < \infty$ for all $\rho < 1$. Moreover, since the value distribution has positive upper and lower bounds, $\mathbb{E}[VT]^{\text{VITA}} < \infty$ implies $\mathbb{E}[T]^{\text{VITA}} < \infty$, establishing that VITA is stable for all $\rho < 1$.

Next, we consider stability of LAVA: let N_{short} (N_{long}) be the time-varying random variable giving the number of jobs at whichever queue happens to be shorter (longer) at a given point in time. Clearly, $N_{\text{short}} \leq N_{\text{long}}$ at all times. Moreover, LAVA is stable if and only if $\mathbb{E}[N_{\text{long}}] < \infty$. Assume by way of contradiction that LAVA is actually unstable and $\mathbb{E}[N_{\text{long}}] = \infty$. In this case, $\mathbb{E}[N_{\text{short}}] < \infty$, because at least one of the servers must have an incoming time-average arrival rate of no more than $\lambda/2 < \mu$. Now observe that whenever $bN_{\text{short}} < aN_{\text{long}}$, LAVA sends all jobs to the shorter queue, setting the arrival rate to the longer queue to 0. However, since $\mathbb{E}[N_{\text{short}}] < \infty$, while $\mathbb{E}[N_{\text{long}}] = \infty$, this condition will hold almost always, contradicting the fact that $\mathbb{E}[N_{\text{long}}] = \infty$. $\square$

Unlike the other policies, C-MU can give rise to unstable systems even when $\rho < 1$.

Observation 1. *There exists a value distribution with positive upper and lower bounds such that* C-MU *is unstable for some load* $\rho < 1$.

We now provide some theoretical justification for Observation 1. Our theoretical justifications are motivated by what we have witnessed in simulation. Consider a value distribution which is bimodal with values (for example) 1 and 10, where the value 1 jobs are extremely rare. Imagine a two-server system under C-MU dispatching with $\rho \in (3/4, 1)$. For a long time, all arrivals have value 10, and these are randomly split between the two queues by the C-MU policy. Eventually, a value 1 job arrives and is dispatched to the server with the longer queue. Without loss of generality, we assume that this is server 2. At this point the C-MU policy enters a particular *overload regime*, in which all subsequent arrivals are sent to server 2.[6] The reason for this is that the average value of jobs at server 1 is 10, while the average value of jobs at server 2 is less than 10; hence, the average departing value across servers will be maximized by routing to server 2. During the overload regime, server 2's queue grows rapidly, as it receives all jobs.

Our simulations suggest that for sufficiently high ρ, we enter this overload regime frequently. Furthermore, although there are times when we are not in this overload regime, during which server 2's queue will shrink, we enter the overload regime so frequently that eventually server 2's queue grows beyond the point of no return, and server 2 heads off to instability.

To understand why this happens, consider what causes the overload regime to end. The *most common cause* is that server 1, which is receiving no jobs, becomes idle, dropping its average value to 0. In this case, the very next arrival will be dispatched to server 1. With very high probability, that arrival will be a value 10 job, which will again send us back to the overload regime, where all arrivals are sent to server 2. In fact, server 1 will likely repeatedly alternate between having no jobs and a single value 10 job, setting the effective arrival rate at server 2 to $2\lambda/3$, which is greater than the departure rate μ, as $\rho > 3/4$. The only catch is that with extremely low probability, when server 1 is idle, a value 1 job might be the next arrival, rather than a value 10 job. This creates a new regime where all value 10 jobs start going to server 1; however, with high probability this regime is (relatively) short-lived because it only lasts until that value 1 job leaves the system, which will happen quickly since the value 1 job was the first to join the queue at server 1. Once server 2 has built up enough jobs, this rare event barely affects its queue length.

A *less common cause* for the overload regime to end is that server 2 loses its value 1 job. This turns out to be unlikely for two reasons: (i) The value 1 job at server 2 arrives into an already busy queue, so if ρ is sufficiently high, it will take a while until it departs (given PS scheduling). (ii) While server 2 has a value 1 job and server 1 does not, new value 1 arrivals are twice as likely to come to server 2 (remember that server 2's effective arrival rate is $2\lambda/3$), thus server 2 is more likely to get any new value 1 job that arrives, hence "replenishing" its supply of value 1 jobs.

Again, the point is that once the queue length at server 2 builds sufficiently, any departure from the overload regime is quickly reversed, returning us to the overload regime.

6.3. Results under sharply bimodal distributions

In this section we explore the efficacy of policies under a class of value distributions exhibiting a high degree of variability. These distributions are of interest for two reasons. First, some of the policies are amenable to asymptotic analysis under these distributions, allowing us to formally establish trends seen in Section 5, albeit in a more limited setting. Second, the VITA and C-MU behave quite differently in this regime, as typified by Fig. 2 (f) and it is insightful to formally explore these observations.

We write $V \sim \text{SBD}(p)$, indicating that V obeys a *sharply bimodal distribution* with a "sharpness" parameter $p > 1/2$, such that

$$V \sim \begin{cases} \dfrac{1-p}{p} = \text{"low-value"} & \text{w.p. } p \\ \dfrac{p}{1-p} = \text{"high-value"} & \text{w.p. } 1-p. \end{cases}$$

[6] We are ignoring the rare case where server 2 has many completions of value 10 jobs, making its queue length sufficiently less than that of server 1, causing value 1 jobs to be sent to server 1, while the queue at server 2 is shorter.

This distribution satisfies the convention $\mathbb{E}[V] = 1$, where the low-value jobs (i.e., those with value $(1-p)/p$) constitute a $1-p$ fraction of the total value, and the high-value jobs (i.e., those with value $p/(1-p)$), constitute a p fraction of the total value. We are typically interested in this distribution in the asymptotic regime where $p \to 1$; here high-value jobs are extremely rare yet comprise essentially all of the value. Note that as $p \to 1/2$, SBD(p) converges to a constant; we name this family of distributions for their "sharp" behavior that emerges only when $p \approx 1$.

Our first result for sharply bimodal distributions concerns the asymptotic optimality of VITA when $V \sim \text{SBD}(p)$ as $p \to 1$ and $\rho < 1/2$.

Theorem 2. *Let $0 < \rho < 1/2$ and $V \sim \text{SBD}(p)$ in a system with two identical servers. As $p \to 1$,* VITA *is asymptotically optimal in the sense that for any policy* P*, we have* $\lim_{p\to 1} \mathbb{E}[VT]^{\text{VITA}}/\mathbb{E}[VT]^{\text{P}} \le 1$.

Proof. Consider a static dispatching policy, P′, that reserves server 1 for low-value jobs and server 2 for the high-value jobs. The system will be stable since, when $\rho < 1/2$, either server can process all arrivals even if operating alone. Since VITA is the optimal static policy, the performance of P′ gives an upper bound on the performance of VITA. Taking the limit as $p \to 1$, we have the bound

$$\lim_{p\to 1} \mathbb{E}[VT]^{\text{VITA}} \le \lim_{p\to 1} \mathbb{E}[VT]^{\text{P}'} = \lim_{p\to 1} \left(\frac{1-p}{\mu - p\lambda} + \frac{p}{\mu - (1-p)\lambda} \right) = \frac{1}{\mu}.$$

Finally, under *any* policy P, we have $\mathbb{E}[VT]^{\text{P}} \ge \mathbb{E}[V]/\mu = 1/\mu$, so $\lim_{p\to 1} \mathbb{E}[VT]^{\text{VITA}}/\mathbb{E}[VT]^{\text{P}} \le 1$. □

Next, we prove that although VITA may not be an asymptotically optimal dispatching policy under higher loads, it continues to dominate the performance of JSQ for all $\rho < 1$, as long as the value distribution is sufficiently "sharp."

Theorem 3. *Let $V \sim \text{SBD}(p)$ in a system with two identical servers. As $p \to 1$,* VITA *asymptotically performs as well as* JSQ*, if not better. That is,* $\lim_{p\to 1} \mathbb{E}[VT]^{\text{VITA}}/\mathbb{E}[VT]^{\text{JSQ}} \le 1$. *Moreover, this inequality is tight as $\rho \to 1$.*

Proof. The case where $\rho < 1/2$ is a direct consequence of Theorem 2.

Now consider the case where $\rho > 1/2$. We will upper bound the performance of VITA by a family of static policies, P(r), parametrized by $r \in (0, 1)$. The VITA-like policy P(r) sends as many high-value jobs to server 2 as possible, while setting the arrival rate of jobs to server 1 at $\lambda_1(r) \equiv \lambda/2 + (\mu - \lambda/2)r$ and the arrival rate of jobs to server 2 at $\lambda_2(r) \equiv \lambda/2 - (\mu - \lambda/2)r$. Note that P($r$) is a stable policy. Since $\rho > 1/2$, for p sufficiently close to 1, we must have $p > \mu/\lambda = 1/(2\rho)$, which ensures that the arrival rate of high-value jobs, $\lambda(1-p)$, is less than the total arrival rate of jobs to server 2, $\lambda_2(r) = \lambda/2 - (\mu - \lambda/2)r$, for all r. Consequently, whenever p is sufficiently large (i.e., $p > 1/(2\rho)$), P(r) sends *all* high-value jobs to server 2 for all $r \in (0, 1)$.

Consider an arbitrary arrival that enters a system and is dispatched via P(r). There are three mutually exclusive possibilities when $p > 1/(2\rho)$:

- with probability $\lambda_1(r)/\lambda$ we have a low-value arrival that is sent to server 1,
- with probability $\lambda_2(r)/\lambda - (1-p)$, we have a low-value arrival that is sent to server 2,
- and with probability $1-p$, we have a high-value arrival that is sent to 2.

Using this information, we find that

$$\mathbb{E}[VT]^{\text{P}(r)} = \left(\frac{\lambda_1(r)}{\lambda}\right)\left(\frac{1-p}{p}\right)\left(\frac{1}{\mu - \lambda_1(r)}\right) + \left[\left(\frac{\lambda_2(r)}{\lambda} - (1-p)\right)\left(\frac{1-p}{p}\right) + (1-p)\left(\frac{p}{1-p}\right)\right]\left(\frac{1}{\mu - \lambda_2(r)}\right).$$

Now recall that $\mathbb{E}[VT]^{\text{JSQ}} \ge 1/(2\mu - \lambda)$, while P($r$) upper bounds the performance of VITA, yielding

$$\lim_{p\to 1}\left(\frac{\mathbb{E}[VT]^{\text{VITA}}}{\mathbb{E}[VT]^{\text{JSQ}}}\right) \le \lim_{p\to 1}\left(\frac{\mathbb{E}[VT]^{\text{P}(r)}}{1/(2\mu - \lambda)}\right) = \frac{2}{1+r}.$$

Taking an infimum over $r \in (0, 1)$, we have the desired result.

To see that the inequality is tight as $\rho \to 1$, first observe that the family of static policies P(r) subsumes VITA, as this family of policies includes all static policies that isolate the high-value jobs. Next, observe that $\lim_{\rho\to 1}\left(\mathbb{E}[VT]^{\text{JSQ}} \cdot (2\mu - \lambda)\right) = 1$. Consequently, both the upper bound on the performance of VITA (after optimizing for r) and the lower bound on the performance of JSQ are tight as $\rho \to 1$, completing the proof. □

Remark: Note that although JSQ outperforms VITA in Figure 2 (f) as $\rho \to 1$ (i.e., for the highest values of ρ), this does not contradict Theorem 3. Under distribution (f), we only have $p \approx 1$, while Theorem 3 holds in the limit as $p \to 1$.

We proceed to state a result comparing the asymptotic performance of LAVA with that of JSQ under the $\mathbb{E}[VT]$ metric, but we first state a Lemma that will be helpful in proving this result.

Lemma 1. *Let $V \sim \text{SBD}(p)$ in a system with two identical servers. As $p \to 1$:*

- *The limiting distribution of the number of low value jobs, N_ℓ, under LAVA converges weakly to the limiting distribution of the total number of jobs, N, under JSQ, and $\mathbb{E}[N_\ell]^{\text{LAVA}} \to \mathbb{E}[N]^{\text{JSQ}}$.*
- *The limiting distribution of the number of high-value jobs, N_h, under LAVA converges weakly to the zero distribution, and $\mathbb{E}[N_h]^{\text{LAVA}} \to 0$.*
- *The limiting distribution of the length of the shorter queue (i.e., the instantaneous minimum length of the two queues), N_{short}, under LAVA converges to the limiting distribution of N_{short} under JSQ, and $\mathbb{E}[N_{\text{short}}]^{\text{LAVA}} \to \mathbb{E}[N_{\text{short}}]^{\text{JSQ}}$.*

Proof. Deferred to appendix. □

Theorem 4. *Let $V \sim \text{SBD}(p)$ in a system with two identical servers. As $p \to 1$,* LAVA *asymptotically performs at least as well as* JSQ. *That is, $\lim_{p\to 1} \mathbb{E}[VT]^{\text{LAVA}}/\mathbb{E}[VT]^{\text{JSQ}} \leq 1$. Moreover, there exists an asymptotic regime where $\rho \to 1$ and $p \to 1$, such that $\mathbb{E}[VT]^{\text{LAVA}}/\mathbb{E}[VT]^{\text{JSQ}} \to 1/2$.*

Proof. First, fix $\rho \in (0, 1)$, and consider a two-server system with values $V \sim \text{SBD}(p)$. Let T_ℓ and T_h be the response time of a low-value and high-value job respectively. As $p \to 1$, from Lemma 1, we have that $\mathbb{E}[N_\ell]^{\text{LAVA}} \to \mathbb{E}[N]^{\text{JSQ}}$, from which Little's Law yields

$$\lim_{p\to 1}\left(\mathbb{E}[T_\ell]^{\text{LAVA}}\right) = \lim_{p\to 1}\left(\frac{\mathbb{E}[N_\ell]^{\text{LAVA}}}{\lambda p}\right) = \frac{\mathbb{E}[N]^{\text{JSQ}}}{\lambda} = \mathbb{E}[T]^{\text{JSQ}} = \mathbb{E}[T_\ell]^{\text{JSQ}}.$$

That is, a low-value job under LAVA experiences a mean response time *asymptotically* equal to that experienced under JSQ as $p \to 1$. We proceed to show that high-value jobs experience even lower response times, which is sufficient to show that LAVA asymptotically does no worse than JSQ with respect to the $\mathbb{E}[VT]$ metric as $p \to 1$.

Observe that under *all* value distributions, $\mathbb{E}[T|V = v]^{\text{LAVA}}$ (i.e., the mean response time of a job with value v under LAVA) is a nonincreasing function of v: when a job arrives to a system, if its value, v, were any higher, the job would either be routed to the same queue or a queue of equal or shorter length. Once a job is in the system, if its value, v, were any higher, it would reduce (or keep fixed) the arrival rate of new jobs coming into its queue. Hence, jobs with higher values are no worse off than jobs with lower values with respect to mean response times, so $\mathbb{E}[T_h]^{\text{LAVA}} \leq \mathbb{E}[T_\ell]^{\text{LAVA}} \to \mathbb{E}[T]^{\text{JSQ}} = \mathbb{E}[T_h]^{\text{JSQ}}$ as $p \to 1$. Since both types of jobs are no worse off under LAVA than under JSQ, it follows that $\lim_{p\to 1} \mathbb{E}[VT]^{\text{LAVA}}/\mathbb{E}[VT]^{\text{JSQ}} \leq 1$ as claimed.

In order to prove the remaining claim that LAVA can outperform JSQ by a factor of two under heavy traffic, we must more accurately quantify $\mathbb{E}[T_\ell]^{\text{LAVA}}$ as $p \to 1$, rather than simply providing a bound as we have done above. We proceed by considering the state of the LAVA system seen by a high-value arrival. By PASTA, this high-value arrival sees a system in steady state, and as $p \to 1$, N_ℓ will be distributed like the number of jobs, N, in a JSQ system, and $\mathbb{E}[N_h]^{\text{LAVA}} \to 0$. Since this job sees only low-value jobs, it will be routed to the shorter queue, say server 2's queue, which will cease to accept further low-value jobs, in accordance with the LAVA policy. Server 2 will only accept high-value jobs, which arrive at a rate of $(1-p)\lambda \to 0$.[7] Hence, the high-value job must share server 2 only with those low-value jobs *already present* in the system when it arrives. The shorter queue contains N_{short} jobs (of independent exponentially distributed sizes S_i, each with mean $1/\mu$). By Lemma 1, $\mathbb{E}[N_{\text{short}}]^{\text{LAVA}} \to \mathbb{E}[N_{\text{short}}]^{\text{JSQ}}$, yielding

$$\lim_{p\to 1}\left(\mathbb{E}[T_\ell]^{\text{LAVA}}\right) = \mathbb{E}\left[\frac{\sum_{i=1}^{N_{\text{short}}+1} S_i}{N_{\text{short}}+1}\right]^{\text{JSQ}} = \frac{(\mathbb{E}[N_{\text{short}}]^{\text{JSQ}}+2)\mathbb{E}[S_i]}{2} = \frac{\mathbb{E}[N_{\text{short}}]^{\text{JSQ}}+2}{2\mu}.$$

[7] Since $p \to 1$, the high-value jobs are arbitrarily more valuable than the low-value jobs by a factor of $\left(\frac{p}{1-p}\right)^2 \to \infty$, so server 1's queue will never grow so long as to resume the arrival process of low-value jobs into server 2's queue during the high-value job's residence.

Consequently, we have

$$\lim_{p\to 1}\mathbb{E}[VT]^{\text{LAVA}} = \lim_{p\to 1}\left(p\left(\frac{1-p}{p}\right)\mathbb{E}[T_\ell]^{\text{LAVA}} + (1-p)\left(\frac{p}{1-p}\right)\left(\frac{\mathbb{E}[N_{\text{short}}]^{\text{JSQ}}+2}{2\mu}\right)\right) = \frac{\mathbb{E}[N_{\text{short}}]^{\text{JSQ}}+2}{2\mu},$$

where we use the fact that $\lim_{p\to 1}\left((1-p)\mathbb{E}[T_\ell]^{\text{LAVA}}\right) = 0\cdot\mathbb{E}[T]^{\text{JSQ}} = 0$.

Now consider the case where ρ is no longer fixed, and we in fact have $\rho\to 1$. Then we can evaluate the following iterated limit:

$$\lim_{\rho\to 1}\left(\lim_{p\to 1}\left(\frac{\mathbb{E}[VT]^{\text{LAVA}}}{\mathbb{E}[VT]^{\text{JSQ}}}\right)\right) = \lim_{\rho\to 1}\left(\frac{\mathbb{E}[N_{\text{short}}]^{\text{JSQ}}+2}{2\mu\cdot\mathbb{E}[T]^{\text{JSQ}}}\right) = \lim_{\rho\to 1}\left(\rho\cdot\frac{\mathbb{E}[N_{\text{short}}]^{\text{JSQ}}+2}{\mathbb{E}[N]^{\text{JSQ}}}\right) = \frac{1}{2},$$

where we have used the facts that $\mathbb{E}[N]^{\text{JSQ}}\to\infty$ and $\mathbb{E}[N_{\text{short}}]^{\text{JSQ}}/\mathbb{E}[N]^{\text{JSQ}}\to 1/2$ as $\rho\to 1$. The first fact is clear, and the latter fact follows form Foschini and Salz [33]: as $\rho\to 1$ the JSQ instantaneous queue lengths are asymptotically balanced, so the length of the shorter queue is on the order of half the total number of jobs in the system. The convergence of the iterated limit implies the existence of a sequence of pairs $\{(p_n,\rho_n)\}_{n=1}^{\infty}\to(1,1)$ (i.e., an "asymptotic regime") under which $\mathbb{E}[VT]^{\text{LAVA}}/\mathbb{E}[VT]^{\text{JSQ}}\to 1/2$, as claimed.

□

Corollary 1. *Let $V\sim$ SBD(p) in a system with two identical servers. There exists an asymptotic regime where $\rho\to 1$ and $p\to 1$, such that we have the following ratio between the performance of various policies:*

$$\mathbb{E}[VT]^{\text{RND}}:\mathbb{E}[VT]^{\text{VITA}}:\mathbb{E}[VT]^{\text{JSQ}}:\mathbb{E}[VT]^{\text{LAVA}}\to 4:2:2:1.$$

Proof. The result follows immediately from Proposition 2 and Theorems 3 and 4. □

We explain why C-MU also performs well in this regime. Just as LAVA essentially employs JSQ until the arrival of a rare high-value job, C-MU essentially employs a variant of RND (where a job is sent to an idle server whenever possible, but dispatching is otherwise random) until the arrival of such a job. Under both LAVA and C-MU, a high-value job is subsequently sent to the server with the shorter queue. The server receiving the high value job will essentially cease to receive arrivals until the completion of that job. It may appear that despite all this LAVA should outperform C-MU because queue lengths under JSQ are shorter than those under RND. We note, however, that under RND, the time-average length of the *shorter* queue is half that of an *arbitrary* queue.

Finally, we ask what would happen if the two-server system was replaced by a single server with twice the service rate, resulting in an M/M/1/PS system. At first it might seem that the single server is superior, but we need to remember that our metric is $\mathbb{E}[VT]$. Corollary 2 shows that there are regimes for which a two-server system is superior.

Corollary 2. *Let $V\sim$ SBD(p) in a system with two identical servers operating at rate μ. Then there exist $\rho^*<1$ and $p^*<1$, such that $\mathbb{E}[VT]^{\text{LAVA}}<\mathbb{E}[VT]^{\text{SINGLE}}$, where* SINGLE *refers to a single* PS *server operating at rate 2μ.*

Proof. From Theorem 4, there exists a regime with $\rho\to 1$, where the performance of LAVA is *strictly* better than JSQ. Moreover, we know from [33] that when $\rho\to 1$, the mean response times under JSQ and SINGLE are arbitrarily close, and since V and T are independent under both JSQ and SINGLE, their performance under $\mathbb{E}[VT]$ is also arbitrarily close. Hence, by continuity there exist $\rho^*<1$ and $p^*<1$ such that $\mathbb{E}[VT]^{\text{LAVA}}<\mathbb{E}[VT]^{\text{SINGLE}}$. □

7. A (sometimes) far better policy: Gated VITA (G-VITA)

In Sections 5 and 6, we saw that the VITA policy often performs poorly relative to most of the other policies, except under sharp bimodal value distributions such as distribution (f) (cf. the definition of SBD(p) in Section 6.3). For such distributions, VITA is asymptotically optimal (as the value distribution grows increasingly sharp) for $\rho<1/2$. Although VITA continues to perform modestly well for these distributions when $\rho>1/2$, it does not perform nearly as well as LAVA when $\rho\to 1$.

To understand why VITA does not perform as well under high loads, observe that the VITA policy's strength lies in isolating the highest value jobs. However, when load is particularly high, the relative efficacy of this isolating

effect is limited because high-value jobs must share their "dedicated" server with too many low-value jobs, in order to ensure system stability. In this section, we explore how adding a "non-static component" to VITA can greatly reduce the number of low-value jobs utilizing this "dedicated" server, without sacrificing system stability at higher loads. We present a policy, Gated VITA (G-VITA), that outperforms LAVA by *an arbitrary factor* under a particular high load regime.

7.1. G-VITA

We define the G-VITA policy with parameter g for two-server systems with bimodal value distributions.

- G-VITA sends low-value jobs to server 2 if and only if the number of low-value jobs present at server 2 is at most g.
- G-VITA always sends high-value jobs to server 2.

We can interpret server 2 as having a limited number of slots reserved for use by low-value jobs. When all of these slots are occupied, a "gate" will close and bar the entry of any further low-value arrivals (sending them to server 1). The gate remains closed until a low-value jobs departs. Note that the gate never bars the entry of high-value jobs. Moreover, while the queue at server 2 can hold up to g low-value jobs and any number of high-value jobs, the queue at server 1 only holds low-value jobs.

While we have defined G-VITA only for bimodal value distributions, the G-VITA policy can be extended to general value distributions by classifying jobs as "low-value" and "high-value" jobs in an appropriate manner.

There exist values of $\rho < 1$ for which the G-VITA policy with fixed parameter g is unstable; however, we will show that for any $\rho < 1$, stability can be guaranteed by requiring g to be sufficiently high.

Lemma 2. *Let $V \sim \mathrm{SBD}(p)$ in a system with two identical servers. As $p \to 1$, the* G-VITA *policy with parameter g under load ρ is stable whenever $\rho < 1$ and either*

$$g > -\frac{\log(2-2\rho)}{\log(2\rho)} - 1, \qquad \textit{or alternatively,} \qquad g > \log_2\left(\frac{\rho}{1-\rho}\right).$$

Proof. Deferred to appendix. □

Recall that when $V \sim \mathrm{SBD}(p)$ in a regime where $\rho \to 1$ and $p \to 1$, high-value jobs under RND, JSQ, VITA, or LAVA share their server with an average number of low-value jobs on the order of $\frac{\rho}{1-\rho}$. Meanwhile, we have just shown that there exist stable G-VITA policies in the same regime such that high-value jobs need only share their server with about $\log_2\left(\frac{\rho}{1-\rho}\right)$ low-value jobs. That is, in this regime, G-VITA "protects" high-value jobs by having them share their server with *exponentially fewer* low-value jobs. This phenomenon lies at the heart of the following result.

Theorem 5. *Let $V \sim \mathrm{SBD}(p)$ in a system with two identical servers and let* $\mathrm{P} \in \{\mathrm{RND}, \mathrm{JSQ}, \mathrm{VITA}, \mathrm{LAVA}\}$. *There exists an asymptotic regime with $\rho \to 1$, $p \to 1$, and (*G-VITA parameter*) $g \to \infty$, s.t. $\mathbb{E}[VT]^{\text{G-VITA}}/\mathbb{E}[VT]^{\mathrm{P}} \to 0$.*

Proof. We prove the result in the case where P is RND by giving an upper bound on $\mathbb{E}[VT]^{\text{G-VITA}}$, then dividing by $\mathbb{E}[VT]^{\mathrm{RND}} = 1/(\mu - \lambda/2)$ and taking a limit as $\rho \to 1$. We will express p and g as parametric functions of ρ. The result for the remaining policies follows from the fact that the performance of RND is within a bounded factor of that of the other policies in this regime (cf. Corollary 1).

Let T_ℓ and T_h be the response times of low-value and high-value jobs, respectively. In order to give an upper bound on $\mathbb{E}[VT]^{\text{G-VITA}}$, we first consider the mean response times of high-value jobs under G-VITA, $\mathbb{E}[T_h]^{\text{G-VITA}}$. High-value jobs arrive to server 2 according to a Poisson process with rate $(1-\lambda)p$, and when one or more such jobs are present, they depart server 2 with rate $h_2 \cdot \mu/(\ell_2 + h_2)$, where ℓ_2 and h_2 are the number of low-value and and high-value jobs present at server 2, respectively. Since $\ell_2 \le g$ and $h_2 \ge 1$ (when departures of high-value jobs from server 2 are possible) we can lower bound the departure rate of high-value jobs by $\mu/(g+1)$. Consequently,

$$\mathbb{E}[T_h]^{\text{G-VITA}} \le \frac{1}{\mu/(g(\rho)+1) - (1-p(\rho))\lambda}.$$

Now consider the mean response time of low-value jobs under G-VITA, $\mathbb{E}[T_\ell]^{\text{G-VITA}}$. From Lemma 2, we know that as $p \to 1$, this quantity is finite for all $\rho < 1$, as long as $g > \log_2(\rho/(1-\rho))$. Consequently, within this asymptotic regime we allow p and g to vary with ρ as $\rho \to 1$, subject to

$$p(\rho) \geq 1 - \frac{1}{\mathbb{E}[T_\ell]^{\text{G-VITA}}} \quad \text{and} \quad g(\rho) \geq \log_2\left(\frac{\rho}{1-\rho}\right) \quad (g(\rho) \in \mathbb{N}),$$

where the first constraint ensures that $(1 - p(\rho))\mathbb{E}[T_\ell]^{\text{G-VITA}} \leq 1$ and the second constraint guarantees stability. Note that these constraints imply that $p(\rho) \to 1$ and $g(\rho) \to \infty$ when $\rho \to 1$ as claimed.[8] We proceed to complete the proof by taking the required limit:

$$\begin{aligned}\lim_{\rho\to 1} \frac{\mathbb{E}[VT]^{\text{G-VITA}}}{\mathbb{E}[VT]^{\text{RND}}} &\leq \lim_{\rho\to 1}\left(\frac{p(\rho)\left(\frac{1-p(\rho)}{p(\rho)}\right)\mathbb{E}[T_\ell]^{\text{G-VITA}} + (1-p(\rho))\left(\frac{p(\rho)}{1-p(\rho)}\right)\left(\frac{\mu}{g(\rho)+1} - (1-p(\rho))\lambda\right)^{-1}}{1/(\mu - \lambda/2)}\right) \\ &\leq \lim_{\rho\to 1}\left(\frac{1 + \log_2\left(\frac{\rho}{1-\rho}\right)\Big/\mu}{(\mu(1-\rho))^{-1}}\right) = 0.\end{aligned}$$

□

7.2. G-VITA simulations

In this section we use simulations to numerically compare the performance of G-VITA with JSQ and LAVA under the value distribution (f). We have held the G-VITA parameter fixed at $g = 5$ for all values of ρ plotted. As shown in Fig. 3, G-VITA performs very well as $\rho \to 1$, strongly outperforming both JSQ and LAVA, as expected. However, the performance of G-VITA is very poor at low loads. This is because the relatively high G-VITA parameter, g, forces high-value jobs to share their server with unnecessarily many low-value jobs under low loads. Even if we vary g with ρ, consistently strong performance is still unattainable because $g \in \mathbb{N}$.

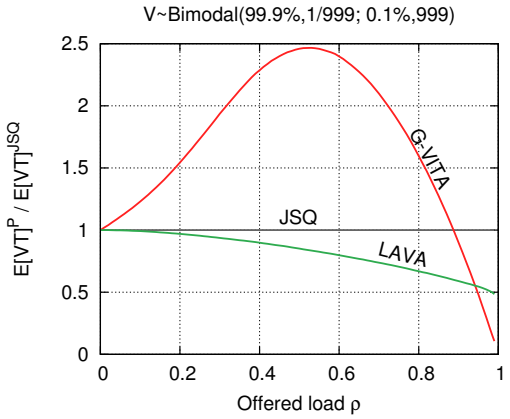


Figure 3: G-VITA dominates JSQ and LAVA under value distribution (f) as $\rho \to 1$ in accordance with Theorem 5.

One can, however, come up with "G-VITA inspired policies" that do very well across all loads under sharply variable (if not all) distributions. For, example, we could redefine G-VITA to be less "eager" in occupying all g available slots at server 2, e.g., as long as there are fewer than g low value jobs at server 2, dispatch low-value jobs using JSQ, rather than always sending them to server 2. Alternatively, consider a policy that routes low value jobs

[8] Note that the right-hand side of the formula bounding $p(\rho)$ actually depends on $p(\rho)$ (i.e., "computing" the bound on $p(\rho)$ involves solving a fixed point problem), but the effect of $p(\rho)$ on this bound vanishes when $p(\rho) \to 1$, as argued in the proof of Lemma 2, so a $p(\rho)$ satisfying this constraint may be found without any problems.

so as to maintain a fixed ratio between the queue lengths at servers 1 and 2 (e.g., attempt to keep server 1 four times as long as server 2), while sending all high-value jobs to the server with the shorter expected queue length. Such a policy would help isolate high-value jobs from low-value jobs to a greater extent than VITA or LAVA (although not as much as G-VITA), while not hurting high-value jobs too much at lower loads. We have verified via simulation that such policies perform well for all ρ (not shown due to lack of space).

8. More complex policies via the First Policy Iteration (FPI)

Thus far, we have considered only simple, intuitive dispatching policies. In this section, we analyze the value-aware dispatching problem in the framework of Markov decision processes (MDP) [34–36]. This will lead us to policies that often perform better than our existing policies, but are more complex and less intuitive.

We start with a tutorial example to explain the FPI approach. Consider a two-server system. If this system were to use the RND dispatching policy, arrivals would be randomly split between the two servers. Instead of using RND, we propose a "first policy iteration" on RND, which we shall call FPI-RND, whereby, an arrival is dispatched so as to minimize the *overall* $\mathbb{E}[VT]$ (given the current state of system), *under the assumption that all future arrivals will be dispatched via* RND.[9] Note that the assumption on how future jobs are routed is actually inaccurate, as future arrivals will continue to be routed via FPI-RND. Here, RND is referred to as the **basic policy** that is *improved upon* by FPI.

The central notion of FPI is the **value function**,[10] denoted by $\eta_{\mathbf{z}}(\alpha)$, where $\mathbf{z}$ is the system state (i.e., the number of jobs at each server and their values) and α is the basic policy being improved (e.g., RND, VITA, etc.). In defining the value function we view the system as incurring a "penalty" of magnitude v for each unit of time a job of value v spends in the system, so that minimizing $\mathbb{E}[VT]$ is equivalent to minimizing the expected rate at which penalty is incurred. Let $C_{\mathbf{z}}(\alpha, t)$ denote the cumulative penalty incurred under policy α during the time interval $(0, t)$ when the initial state is $\mathbf{z}$, and let $r(\alpha)$ denote the mean (equilibrium) rate at which penalty is incurred under α. More formally,

$$C_{\mathbf{z}}(\alpha, t) \equiv \mathbb{E}\left[\int_0^t \left(\sum_{i=1}^m v_i^{\text{sum}}(\tau)\right) d\tau \,\middle|\, \text{the state at time } \tau = 0 \text{ is } \mathbf{z}\right]^{\alpha},$$

$$r(\alpha) \equiv \mathbb{E}\left[\lim_{t\to\infty}\left(\frac{1}{t}\int_0^t \left(\sum_{i=1}^m v_i^{\text{sum}}(\tau)\right) d\tau\right)\right]^{\alpha} = \lim_{t\to\infty}\left(\frac{\lambda t \cdot \mathbb{E}[VT]^{\alpha}}{t}\right) = \lambda \cdot \mathbb{E}[VT]^{\alpha},$$

where $v_i^{\text{sum}}(\tau)$ denotes the sum of the values of the jobs at server i at time τ. With this framework, we can define the **value function**, $\eta_{\mathbf{z}}(\alpha)$, as the expected difference in cumulative (infinite time-horizon) penalties incurred between a system initially in state $\mathbf{z}$ and a system in equilibrium,

$$\eta_{\mathbf{z}}(\alpha) \equiv \lim_{t\to\infty} \left(C_{\mathbf{z}}(\alpha, t) - r(\alpha) \cdot t\right).$$

Hence, $\eta_{\mathbf{z}_2}(\alpha) - \eta_{\mathbf{z}_1}(\alpha)$ quantifies the benefit of starting in $\mathbf{z}_2$ rather than $\mathbf{z}_1$. In general, value functions enable *policy iteration*, a procedure that, under certain conditions, converges to the optimal policy. Here, due to the complexity of the system, we are limited to only the **first policy iteration** (FPI) step.

In our case, the value function depends on the dispatching policy, α, and the system state, $\mathbf{z} \equiv (z_1, \ldots, z_m)$, where each $z_i \equiv (v_{i,1}, \ldots, v_{i,n_i})$, gives the state of server i. The key idea with policy iteration is to consider the optimal deviation from the *basic policy* α for *one decision*, that is, dispatching one new arrival with value v, and then returning to α so that the expected future costs are given by the value function. The optimal decision corresponds to a new improved policy α',

$$\alpha'(\mathbf{z}, v) = \underset{\mathbf{z}' \in \mathcal{A}(\mathbf{z}, v)}{\operatorname{argmin}} \eta_{\mathbf{z}'}(\alpha) - \eta_{\mathbf{z}}(\alpha), \tag{4}$$

where $\mathcal{A}(\mathbf{z}, v)$ denotes the states that can result from dispatching a value v arrival to one of the m servers. Note that the resulting policy, $\alpha'(\mathbf{z}, v)$, also depends on the value of the new arrival, v, and although $\alpha'(\mathbf{z}, v)$ always assumes further

[9] Actually implementing such a policy requires some calculations, which will be shown later in this section.

[10] The word "value" in "value function" is not directly related to the use of the word "value," as used elsewhere in the paper.

arrivals will be routed according to α, the actual policy will continue to dispatch according to the one-step optimal deviations described above.

Note that LAVA, discussed in Section 4.5, is based on the assumption that no jobs arrive afterwards. In contrast, FPI assumes that after the current decision, each server continues to receives a stream of arrivals, based on how they would be dispatched by the dispatching policy α.

8.1. FPI policies

In this section we first determine the value function of a basic policy α and subsequently use it to derive the FPI policy. Due to the complex state-space, it is difficult to determine the value function of an arbitrary basic policy (e.g., LAVA). Therefore, as in [37], we use a static basic policy (e.g., RND or VITA). In this case, the arrival process decomposes to m independent Poisson processes, and it is sufficient to derive the value function for each M/M/1-PS queue separately as $\eta_{\mathbf{z}}(\alpha) = \sum_{i=1}^{m} \eta_{\mathbf{z}_i}^{(i)}(\alpha)$, where $\eta_{z_i}^{(i)}(\alpha)$ denotes the value function of server i in state z_i. Letting v^{sum} be the sum of the values of all jobs in an M/M/1-PS queue in state z, the corresponding value function is given by Proposition 3.

Proposition 3. *The value function for the M/M/1-PS queue with arrival rate λ, service rate μ, and values V is given by*

$$\eta_z = \frac{\lambda n(n+1)}{2(\mu-\lambda)(2\mu-\lambda)}\,\mathbb{E}[V] + \frac{n+1}{2\mu-\lambda}\,v^{\text{sum}} + c, \tag{5}$$

where c is a constant.

Proof. Deferred to appendix. □

As previously mentioned, we assume a static basic policy α. Since α is static, it necessarily defines an independent server-specific Poisson arrival process at each server with arrival rate λ_i and value distribution V_i. Consequently, these server-specific arrival processes (λ_i, V_i), allow us to use the value function from (5) to derive the FPI policy using (4).

Proposition 4. *For a static basic policy α, yielding server-specific arrival processes (λ_i, V_i), the corresponding FPI policy routes a job of value v to the server given by*

$$\alpha'(v) = \operatorname*{argmin}_i \quad \frac{1}{2\mu_i-\lambda_i}\left(\frac{\lambda_i\,\mathbb{E}[V_i](n_i+1)}{\mu_i-\lambda_i} + v_i^{\text{sum}} + (n_i+2)v\right). \tag{6}$$

Proof. Deferred to appendix. □

Remark: We note that letting $\lambda_i \to 0$ in (6) reduces to LAVA given in (2), and letting $\lambda_i \to \mu_i$ in (6) reduces to JSQ.

8.2. Enhancing FPI policies using discounting

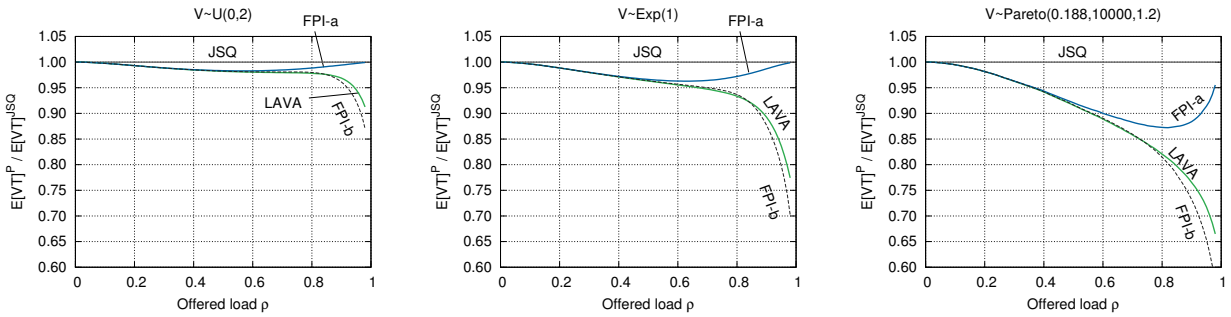
In this section we provide a novel idea for enhancing FPI policies. Inspired by LAVA, which ignores future arrivals completely, we consider modifications of FPI policies where we discount the impact of future arrivals on dispatching. We identify the terms corresponding to the harm caused to future jobs (the term with $\mathbb{E}[V_i]$) the present jobs (the terms with v_i^{sum} and v) in (6), and introduce an additional *weight parameter* γ to discount the former. This yields

$$\boxed{\alpha'_\gamma(v) = \operatorname*{argmin}_i \quad \frac{1}{2\mu_i-\lambda_i}\left(\gamma\cdot\frac{\lambda_i\,\mathbb{E}[V_i](n_i+1)}{\mu_i-\lambda_i} + v_i^{\text{sum}} + (n_i+2)v\right).} \tag{7}$$

Suppose that the servers are identical, $\mu_i = \mu$, and the basic policy α balances the load ($\lambda_i = \lambda/m$). Then,

$$\boxed{\alpha'_\gamma(v) = \operatorname*{argmin}_i \quad \left(\gamma\cdot\frac{\lambda\,\mathbb{E}[V_i](n_i+1)}{\mu-\lambda/m} + v_i^{\text{sum}} + n_i v\right).} \tag{8}$$

We observe that when $\gamma = 1$, this results in the original FPI policy presented in Proposition 4, while when $\gamma = 0$, this results in the LAVA policy (regardless of the basic policy used).



(a) (b) (c)

Figure 4: Performance of FPI-a, FPI-b, and LAVA relative to JSQ in a two-server system, with value distributions (a) Uniform, (b) Exponential, (c) Pareto from Table 1. Each point corresponds to the mean performance with about 100 million jobs.

8.3. FPI simulations

In this section we use simulations to numerically compare the performance of two FPI policies with JSQ and LAVA:

1. **FPI-a** uses a weight parameter of $\gamma = 1$ (i.e., the FPI policy presented in Proposition 4) and is based on RND.
2. **FPI-b** uses a weight parameter of $\gamma = 1/20$ (i.e., future arrivals are heavily discounted). Rather than being based on RND, this policy is based on a variation of VITA where load is equalized between the two servers.

The numerical results relative to JSQ are shown in Fig. 4. When ρ is low, all FPI policies perform like LAVA. Meanwhile, the performance of FPI-a converges to that of JSQ as $\rho \to 1$, in accordance with the remark made after Proposition 4. Consequently, FPI-a performs worse than LAVA. The FPI-b policy, however, outperforms LAVA under high ρ.

9. Conclusion

This paper presents the first comprehensive study of dispatching policies that aim to minimize value-weighted response times under Process-Sharing scheduling. We propose a large number of novel dispatching policies and compare these under a range of workloads, showcasing the fact that the value distribution and load can greatly impact the ranking of the policies. We also prove several intriguing results on the asymptotic behavior of these policies. Note that while we have assumed that job values are known exactly, most of our results generalize easily where jobs belong to classes and only the mean value of each class is known.

As value-driven dispatching is a very new problem, there remains ample room for future work on analyzing the policies in this paper and proposing new ones. Other directions for extending the results in this paper include considering more complicated arrival processes and job size distributions, possibly with correlations between consecutive arrivals. Additionally, one could consider alternative value-weighted response time metrics, including higher moments and distribution tails.

Acknowledgements

We would like to thank the reviewers for their helpful comments. Special thanks to Bruno Gaujal and Gautam Iyer for their assistance in refining some of the paper's technical details.

References

[1] Cisco systems localdirector, http://www.cisco.com/c/en/us/products/routers/localdirector-400-series/.
[2] M. Pistoia, C. Letilley, IBM websphere performance pack: Load balancing with IBM secureway network dispatcher, IBM Redbooks (Oct. 1999).

[3] F5 Products, Big-IP, http://www.f5.com/products/big-ip.
[4] Microsoft sharepoint 2010 load balancer, http://www.loadbalancer.org/sharepoint.php.
[5] V. Gupta, M. Harchol-Balter, K. Sigman, W. Whitt, Analysis of join-the-shortest-queue routing for web server farms, Performance Evaluation 64 (9-12) (2007) 1062–1081.
[6] F. Bonomi, On job assignment for a parallel system of processor sharing queues, IEEE Trans. Comput. 39 (7) (1990) 858–869.
[7] E. Altman, U. Ayesta, B. Prabhu, Load balancing in processor sharing systems, Telecommunication Systems 47 (1) (2011) 35–48.
[8] E. Hyytiä, J. Virtamo, S. Aalto, A. Penttinen, M/M/1-PS queue and size-aware task assignment, Performance Evaluation 68 (11) (2011) 1136–1148, (IFIP Performance'11).
[9] H. Feng, V. Misra, Mixed scheduling disciplines for network flows, SIGMETRICS Perform. Eval. Rev. 31 (2003) 36–39.
[10] M. Bramson, Y. Lu, B. Prabhakar, Randomized load balancing with general service time distributions, in: Proceedings of the ACM Special Interest Group on Computer Systems Performance, SIGMETRICS 2010, 2010.
[11] A. Mukhopadhyay, R. R. Mazumdar, Analysis of load balancing in large heterogeneous processor sharing systems, http://arxiv.org/abs/1311.5806 (Nov. 2013).
[12] Y. Lu, Q. Xie, G. Kliot, A. Geller, J. R. Larus, A. Greenberg, Join-idle-queue: A novel load balancing algorithm for dynamically scalable web services, Perform. Eval. 68 (11) (2011) 1056–1071.
[13] M. Harchol-Balter, Performance Modeling and Design of Computer Systems: Queueing Theory in Action, Cambridge University Press, 2013.
[14] M. Harchol-Balter, M. Crovella, C. Murta, On choosing a task assignment policy for a distributed server system, IEEE Journal of Parallel and Distributed Computing 59 (1999) 204–228.
[15] M. Harchol-Balter, Task assignment with unknown duration, Journal of the ACM 49 (2).
[16] M. El-Taha, B. Maddah, Allocation of service time in a multiserver system, Management Science 52 (4) (2006) 623–637.
[17] G. Ciardo, A. Riska, E. Smirni, Equiload: a load balancing policy for clustered web servers, Performance Evaluation 46 (2001) 101–124.
[18] E. Bachmat, H. Sarfati, Analysis of size interval task assigment policies, Performance Evaluation Review 36 (2) (2008) 107–109.
[19] V. Cardellini, E. Casalicchio, M. Colajanni, P. Yu, The state of the art in locally distributed web-server systems, ACM Computing Surveys 34 (2) (2002) 1–49.
[20] R. Weber, On optimal assignment of customers to parallel servers, Journal of Applied Probability 15 (1978) 406–413.
[21] W. Winston, Optimality of the shortest line discipline, Journal of Applied Probability 14 (1977) 181–189.
[22] A. Ephremides, P. Varaiya, J. Walrand, A simple dynamic routing problem, IEEE Transacactions on Autonomic Control AC-25 (4) (1980) 690–693.
[23] J. Kingman, Two similar queues in parallel, Biometrika 48 (1961) 1316–1323.
[24] I. Adan, G. van Houtum, J. van der Wal, Upper and lower bounds for the waiting time in the symmetric shortest queue system, Annals of Operations Research 48 (1994) 197–217.
[25] J. Lui, R. Muntz, D. Towsley, Bounding the mean response time of the minimum expected delay routing policy: an algorithmic approach, IEEE Transactions on Computers 44 (12) (1995) 1371–1382.
[26] M. Harchol-Balter, A. Scheller-Wolf, A. Young, Surprising results on task assignment in server farms with high-variability workloads, in: ACM Sigmetrics 2009 Conference on Measurement and Modeling of Computer Systems, 2009, pp. 287–298.
[27] D. W. Fife, Scheduling with random arrivals and linear loss functions, Management Science 11 (3) (1065) 429–437.
[28] S. Yang, G. de Veciana, Size-based adaptive bandwidth allocation: optimizing the average QoS for elastic flows, in: IEEE INFOCOM, Vol. 2, 2002, pp. 657–666.
[29] M. Harchol-Balter, K. Sigman, A. Wierman, Asymptotic convergence of scheduling policies with respect to slowdown, Perform. Eval. 49 (1-4) (2002) 241–256.
[30] E. Hyytiä, S. Aalto, A. Penttinen, Minimizing slowdown in heterogeneous size-aware dispatching systems, ACM SIGMETRICS Performance Evaluation Review 40 (2012) 29–40, (ACM SIGMETRICS/Performance conference).
[31] E. Hyytiä, Lookahead actions in dispatching to parallel queues, Performance Evaluation 70 (10) (2013) 859–872, (IFIP Performance'13).
[32] E. Hyytiä, S. Aalto, Round-robin routing policy: Value functions and mean performance with job- and server-specific costs, in: 7th International Conference on Performance Evaluation Methodologies and Tools (ValueTools), Torino, Italy, 2013.
[33] G. J. Foschini, J. Salz, A basic dynamic routing problem and diffusion, IEEE Transactions on Communications 26 (3) (1978) 320–327.
[34] S. M. Ross, Applied Probability Models with Optimization Applications, Holden-Day Inc., 1970.
[35] R. A. Howard, Dynamic Probabilistic Systems, Volume II: Semi-Markov and Decision Processes, Wiley Interscience, 1971.
[36] M. L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, Wiley, 2005.
[37] K. R. Krishnan, Joining the right queue: a state-dependent decision rule, IEEE Transactions on Automatic Control 35 (1) (1990) 104–108.
[38] A. F. Karr, Weak convergence of a sequence of markov chains, Probability Theory and Related Fields 33 (1) (1975) 41–48.
[39] M. M. Rao, R. J. Swift, Probability theory with applications, Vol. 582, Springer, 2006.
[40] D. Gross, J. F. Shortle, J. M. Thompson, C. M. Harris, Fundamentals of queueing theory, John Wiley & Sons, 2013.
[41] B. Sengupta, D. Jagerman, A conditional response time of M/M/1 processor-sharing queue, AT&T Bell Lab. Techn. J. 64 (2) (1985) 409–421.
[42] S. Aalto, J. Virtamo, Basic packet routing problem, in: The thirteenth Nordic teletraffic seminar NTS-13, Trondheim, Norway, 1996, pp. 85–97.

Appendix A. Proofs of results

Appendix A.1. Proof of Proposition 1

Proposition 1. *VITA is the optimal (i.e., $\mathbb{E}[VT]$-minimizing) static policy for any two-server system with identical service rates. Furthermore, VITA unbalances the load, whereas all load balancing static policies achieve the same performance as RND.*

Proof. First observe that any static policy can be described by a measurable function $\varphi(\cdot)$, such that $\varphi(v)$ is the probability that a job with value v is routed to server 1, and consequently, $1-\varphi(v)$ gives the probability of routing a job to server 2. For example, RND is given by $\varphi(v) = 1/2$ for all v, while VITA is given by $\varphi(v) = 1$ for all values v above some threshold and $\varphi(v) = 0$ for all values v below that threshold.

Describing static policies by such functions, the $\mathbb{E}[VT]$-minimizing policy is given by $\varphi(\cdot)$ in the solution of the following minimization problem:

$$\begin{aligned}
\min_{p_i;m_i;\varphi(\cdot)} \quad & \frac{m_1}{\mu - p_1\lambda} + \frac{m_2}{\mu - p_2\lambda} \\
\text{s.t.} \quad & \varphi\colon \mathbb{R}^+ \to [0,1] \text{ is measurable} \\
& m_1 = \int_0^\infty v\varphi(v)\,dF(v), \quad m_2 = \int_0^\infty v(1-\varphi(v))\,dF(v) \\
& p_1 = \int_0^\infty \varphi(v)\,dF(v), \quad p_2 = \int_0^\infty (1-\varphi(v))\,dF(v) \\
& p_1\lambda < \mu_1, \quad p_2\lambda < \mu_2 \\
& p_1 \geq p_2
\end{aligned}$$

where F is the c.d.f. of the value distribution. Here, we can interpret p_i as the fraction of jobs sent to server i and m_i as the average value of the jobs sent to server i weighted by the fraction of jobs sent to server i (i.e., the average value of the jobs sent to server i multiplied by p_i). We note that $p_1 + p_2 = 1$ and $m_1 + m_2 = \mathbb{E}[V]$. Moreover, we note that although the constraint $p_1 \geq p_2$ need not hold for all feasible static policies, this restriction is without loss of generality,[11] and simplifies the feasible region.

Now fix p_1 and p_2 at their optimal values, which simplifies the optimization problem: we must now minimize a weighted sum of m_1 and m_2 subject to $m_1+m_2 = \mathbb{E}[V]$ and bounds on m_1 and m_2. Since $p_1 \geq p_2$, we have $1/(\mu-p_1\lambda) \geq 1/(\mu - p_2\lambda)$, and hence, the coefficient of m_1 in this weighted sum is greater than that of m_2. Consequently, since $m_1, m_2 \geq 0$, the objective function is minimized by making m_1 as small as possible, subject to the lower bound on m_1 imposed by the fixed value of p_1. This means that we want to send as many higher value jobs to server 2 as possible, so we must have an optimal φ function given by

$$\varphi(v) = \begin{cases} 1 & v < \xi \\ \varphi(\xi) & v = \xi \\ 0 & v > \xi \end{cases},$$

where ξ and $\varphi(\xi)$ satisfy

$$\int_0^\xi dF(v) + \varphi(\xi)\mathrm{P}\{V = \xi\} = p_1,$$

with the integral is evaluated on an open interval. Since p_1 was chosen optimally, by assumption, we can conclude that ξ is the *optimal* dispatching threshold. Therefore, the optimal φ function describes the VITA policy exactly; so we may conclude that VITA is the optimal static policy.

Since the load at server i is $p_i\lambda/\mu$, in order to prove that VITA unbalances the load, we need only prove that $p_1 > p_2$ (i.e., $p_1 > 1/2$) in the optimal solution. Assume, by way of contradiction, that $p_1 = p_2$. Under VITA (and

[11] If $p_1 < p_2$, one can interchange p_1 and p_2, interchange m_1 and m_2, and replace φ with $1-\varphi$ to obtain the same objective value with $p_1 > p_2$, as required)

hence, in the optimal solution), m_1 (m_2) corresponds to the portion of $\mathbb{E}[V]$ made up of jobs lying below (above) the median of V, and hence $m_1 < m_2$. Now consider increasing p_1 (and consequently, decreasing p_2) by some small $\delta > 0$, while preserving a VITA-like threshold policy (i.e., $\varphi(v)$ is monotonically decreasing and $\{0, 1\}$-valued for all v, except at perhaps one threshold point).

Consequently, m_1 will increase and m_2 will decrease by some value $\epsilon(\delta)$, since the $\delta/2$ least valuable fraction of jobs that were being sent to server 2 will be rerouted to server 1. We argue that for sufficiently small $\delta > 0$, we must have $\epsilon(\delta) \leq c\delta$ for some constant c. For example, if $\delta < 1/4$, we must have $\epsilon(\delta) \leq c\delta$, where c is the upper quartile of the value distribution, as all rerouted values will have value at most c.

Finally, let Δ be the change in the objective function due to increasing p_1 from $1/2$ to $1/2 + \delta$. For $\delta > 0$ small enough to ensure that $\mu > \lambda(1/2 + \delta)$, we must have

$$\frac{1}{\mu - \lambda(1/2 + \delta)} - \frac{1}{\mu - \lambda(1/2 - \delta)} \geq 0,$$

which allows us to establish that

$$\begin{aligned}
\Delta &\equiv \frac{m_1 + \epsilon(\delta)}{\mu - \lambda(1/2 + \delta)} + \frac{m_2 - \epsilon(\delta)}{\mu - \lambda(1/2 - \delta)} - \frac{m_1 + m_2}{\mu - \lambda/2} \\
&\leq \frac{m_1 + c\delta}{\mu - \lambda(1/2 + \delta)} + \frac{m_2 - c\delta}{\mu - \lambda(1/2 - \delta)} - \frac{m_1 + m_2}{\mu - \lambda/2}, \\
\lim_{\delta \to 0}\left(\frac{\Delta}{\delta}\right) &\leq \lim_{\delta \to 0}\left(\frac{1}{\delta}\right)\left(\frac{m_1 + c\delta}{\mu - \lambda(1/2 + \delta)} + \frac{m_2 - c\delta}{\mu - \lambda(1/2 - \delta)} - \frac{m_1 + m_2}{\mu - \lambda/2}\right) \\
&= \frac{4\lambda(m_1 - m_2)}{(\mu - \lambda/2)^2} < 0,
\end{aligned}$$

as $m_1 < m_2$. Hence, the objective function can be decreased by a slight increase in p_1, which provides the desired contradiction: a load balancing policy is suboptimal, and since we have shown VITA to be optimal, it unbalances load.

Finally, any load balancing static policy, including RND, that dispatches according to some function φ obtains

$$\mathbb{E}[VT] = \frac{\int_0^\infty v\varphi(v)\, dF(v) + \int_0^\infty v(1 - \varphi(v))\, dF(v)}{\mu - \lambda/2} = \frac{\int_0^\infty v\, dF(v)}{\mu - \lambda/2} = \frac{\mathbb{E}[V]}{\mu - \lambda/2},$$

which does not depend on φ, completing the proof. □

Appendix A.2. Proof of Lemma 1

Lemma 1. *Let $V \sim \mathrm{SBD}(p)$ in a system with two identical servers. As $p \to 1$:*

- The limiting distribution of the number of low value jobs, N_ℓ, under LAVA converges weakly to the limiting distribution of the total number of jobs, N, under JSQ, and $\mathbb{E}[N_\ell]^{\mathrm{LAVA}} \to \mathbb{E}[N]^{\mathrm{JSQ}}$.
- The limiting distribution of the number of high-value jobs, N_h, under LAVA converges weakly to the zero distribution, and $\mathbb{E}[N_h]^{\mathrm{LAVA}} \to 0$.
- The limiting distribution of the length of the shorter queue (i.e., the instantaneous minimum length of the two queues), N_{short}, under LAVA converges to the limiting distribution of N_{short} under JSQ, and $\mathbb{E}[N_{\mathrm{short}}]^{\mathrm{LAVA}} \to \mathbb{E}[N_{\mathrm{short}}]^{\mathrm{JSQ}}$.

In proving Lemma 1, we make use of Sublemma 1 below. This result is a special case of a result due to Karr [38].

Sublemma 1. *Let $M_1, M_2, \ldots, M_n, \ldots$ be a sequence of ergodic Markov chains defined on the same countable space A, each with its own transition rate matrix Q_n and unique nowhere-zero limiting distribution π_n, uniquely solving $\pi_n Q_n = \mathbf{0}$ and $\pi_n \cdot \mathbf{1} = 1$. Furthermore, let M be a (possibly non-ergodic) Markov chain with transition rate matrix Q and unique limiting distribution π, uniquely solving $\pi Q = \mathbf{0}$ and $\pi \cdot \mathbf{1} = 1$ such that $Q_n \to Q$ uniformly. Then $\pi_n \to \pi$ in the sense of weak convergence.*

Proof of Sublemma 1. If π_n uniquely solves the linear system $\pi_n(\mathbf{1}, Q_n) = (1, \mathbf{0})$ while π uniquely solves the linear system $\pi(\mathbf{1}, Q) = (1, \mathbf{0})$, and $Q_n \to Q$ uniformly (and thus, $(\mathbf{1}, Q_n) \to (\mathbf{1}, Q)$ uniformly), we must have $\pi_n \to \pi$ uniformly. It follows that $\pi_n \to \pi$ in the sense of weak convergence. □

Note that since M may not be ergodic, π may be zero somewhere. This occurs when all transitions (in Q_n) into some nonempty proper subset of states $B \subsetneq A$ converge to 0. Note that since π is unique, it is guaranteed that there are paths made up of transitions with nonzero rates (in Q) from all states in B to the non-transient portion of the state space of M.

Proof of Lemma 1. The three individual results follow in a straightforward way after the proper application of Sublemma 1 to the Markov chains of interest.

For $p \in (1/2, 1)$, let the Markov chain $M^{(p)}$ (with transition rate matrix $Q^{(p)}$) denote the underlying Markov chain of the two-server system under LAVA when $V \sim \text{SBD}(p)$, with state space $A \equiv \{(\ell_1, h_1, \ell_2, h_2)\colon \ell_1, h_1, \ell_2, h_2 \in \mathbb{N}_{\geq 0}\}$, where ℓ_i and h_i track the number of low-value and high-value jobs at server i, respectively. By Theorem 1, the LAVA policy is stable (so long as the value distribution has a nonzero lower bound and a finite upper bound, as is the case here). Moreover, all states communicate with one another, so these Markov chains are ergodic. Consequently, each Markov chain, $M^{(p)}$, has a unique nowhere-zero limiting distribution, $\pi^{(p)}$, which uniquely solves $\pi^{(p)}Q^{(p)} = \mathbf{0}$ and $\pi^{(p)} \cdot \mathbf{1} = 1$.

Next, define the Markov chain $M^{(1)}$ over the state space A by letting its transition rate matrix, $Q^{(1)}$, be given by replacing every instance of p in $Q^{(p)}$ with 1.[12] Observe that the states $(\ell_1, h_1, \ell_2, h_2)$ with $h_1 = h_2 = 0$ all communicate with one another, but states with $h_1 > 0$ or $h_2 > 0$ are inaccessible from the aforementioned states (transition rates entering these states from the other states in the chain are zero), making this a non-ergodic Markov chain. The non-transient states of $M^{(1)}$ are exactly those states where $h_1 = h_2 = 0$. Since all transition rates of $M^{(p)}$ are either constant in p or equal to λp, $\lambda p/2$, $\lambda(1-p)$, or $\lambda(1-p)/2$ (in accordance with the LAVA policy), we see that as $p \to 1$, the transition rate matrices $Q^{(p)} \to Q^{(1)}$ uniformly.

Now let M^{JSQ} (with transition rate matrix Q^{JSQ}) denote the underlying Markov chain of the two-server system under JSQ, with state space $\{(j_1, j_2)\colon j_1, j_2 \in \mathbb{N}_{\geq 0}\}$, where j_i tracks the number of jobs at server i. Note that the composition of jobs is unimportant to the evolution of the system under JSQ, and hence, this Markov chain does not track this composition, and nor does not depend on p. Observe that M^{JSQ} is exactly the same Markov chain as $M^{(1)}$ with the transient (inaccessible) portion removed: we identify states (j_1, j_2) with states $(j_1, 0, j_2, 0)$, and remove the other states. Now observe that by Theorem 1, we know that the JSQ system is stable, and moreover all states communicate with one another, so M^{JSQ} has a unique nowhere-zero limiting distribution π^{JSQ}, which uniquely solves $\pi^{\text{JSQ}}Q^{\text{JSQ}} = \mathbf{0}$ and $\pi^{\text{JSQ}} \cdot \mathbf{1} = 1$. It follows that $M^{(1)}$ also has a unique limiting distribution, $\pi^{(1)}$, uniquely solving $\pi^{(1)}Q^{(1)} = \mathbf{0}$ and $\pi^{(1)} \cdot \mathbf{1} = 0$, where $\pi^{(1)}$ coincides with π^{JSQ} on the states $(\ell_1, h_1, \ell_2, h_2)$ where $h_1 = h_2 = 0$, and is equal to zero on all other states.

Now consider an arbitrary increasing sequence $p_1, p_2, \ldots p_n, \ldots \in (0, 1)$, such that $p_n \to 1$. Define a sequence of Markov chains $M_1, M_2, \ldots, M_n, \ldots$ (with transition rate matrices $Q_1, Q_2, \ldots, Q_n, \ldots$), where $M_n = M^{(p_n)}$, and let $M = M^{(1)}$, $Q = Q^{(1)}$ and $\pi = \pi^{(1)}$. We may now apply Sublemma 1, obtaining $\pi_n \to \pi$ in the sense of weak convergence, from which it follows that as $p \to 1$, $\pi^{(p)} \to \pi^{\text{JSQ}}$ on the states $(\ell_1, h_1, \ell_2, h_2)$, where $h_1 = h_2 = 0$ (using the identification previously explained), and $\pi^{(p)}$ converges to the zero distribution, elsewhere. It follows from this convergence that as $p \to 1$:

- The limiting distributions of $N_\ell \equiv \ell_1 + \ell_2$ under LAVA converges to the limiting distribution of $N \equiv j_1 + j_2$ under JSQ.
- The limiting distribution of $N_h \equiv h_1 + h_2$ converges to the zero distribution.
- The limiting distribution of $N_{\text{short}} \equiv \min\{\ell_1 + h_1, \ell_2 + h_2\}$ converges to limiting distribution of $N_{\text{short}} \equiv \min\{j_1, j_2\}$ under JSQ.

[12] We cannot associate $M^{(1)}$ directly with the two-server system under LAVA when $V \sim \text{SBD}(1)$, as SBD(1) is not a well-defined distribution.

To complete the proof, we must show that the expectations also converge. This does not follow immediately from the convergence of the limiting distributions, as N_ℓ, N_h, and N_{short} are *unbounded* functions on the state space $A \equiv \{(\ell_1, h_1, \ell_2, h_2)\colon \ell_1, h_1, \ell_2, h_2 \in \mathbb{N}_{\geq 0}\}$. Now observe that for any $p \in (1/2, 1)$, at any given time we have $N_\ell, N_h, N_{\text{short}} \leq N \equiv \ell_1 + h_1 + \ell_2 + h_2$ under LAVA. Using the convergence of limiting probabilities established above, together with the fact that N bounds the random variables of interest, we may apply the de la Vallée-Poussin Lemma (cf. [39]) to obtain a sufficient condition for $\mathbb{E}[N_\ell]^{\text{LAVA}} \to \mathbb{E}[N]^{\text{JSQ}}$, $\mathbb{E}[N_h]^{\text{LAVA}} \to 0$, and $\mathbb{E}[N_{\text{short}}]^{\text{LAVA}} \to \mathbb{E}[N_{\text{short}}]^{\text{JSQ}}$. In order for the expectations to converge as claimed, it is sufficient to show that for any fixed $\rho \in (0, 1)$, there exists some $\delta > 0$ and $K < \infty$, such that for all $p \in (1 - \delta, 1)$, we have $\mathbb{E}[N^2]^{\text{LAVA}} < K$. That is, we want to show that the second moment of the number of jobs in the LAVA system is bounded for all p in a neighborhood of 1.

In proving that this sufficient condition holds, it will be useful to introduce a policy, which we call LAVA′, defined form $V \sim \text{SBD}(p)$, as follows:

- if there are no high-value jobs in the system, low-value jobs are dispatched randomly (i.e., according to RND);
- if there are one or more high-value jobs in the system, low-value jobs are routed to server 1;
- high-value jobs are always routed to server 2.

We argue that N under LAVA with load $\rho < 1$, is stochastically no greater than N under LAVA′ with load $\rho' \equiv 2\rho/(1 + \rho) < 1$. In the absence of a high-value job, LAVA dispatches according to JSQ, while LAVA′ dispatches according to RND. Moreover, observe that each queue in a system under RND with load ρ' has as many jobs (in expectation) as an entire two-server system under RND with load ρ, and hence, more jobs than either queue of a system under JSQ with load ρ. Therefore, ignoring the effect of high-value jobs, each queue of the LAVA′ system with load ρ' is stochastically longer than either queue of the LAVA system under load ρ.

Meanwhile, LAVA′ sends high-value jobs to the same server, and allows them to always create a "stopper" effect at that server (i.e., all low-value jobs are sent to the other server). This "stoppering" behavior causes the queue at the other server to be much longer than it otherwise would be. Although a similar phenomenon occurs under LAVA, this effect is more pronounced under LAVA′, as the latter policy sends *all* high-value jobs to server 2.[13] Hence, N under LAVA′ with load ρ' (with associated arrival rate $\lambda' \equiv 2\mu \cdot \rho'$ and service rate μ) stochastically dominates N under LAVA with load ρ.

We proceed to show that $\mathbb{E}[N^2]^{\text{LAVA}'}$ is finite for any given $\rho' < 1$ and p sufficiently close to 1. Clearly, the contribution to $\mathbb{E}[N^2]^{\text{LAVA}'}$ from server 2 is bounded, because (for all p sufficiently close to 1) server 2 receives jobs with mean interarrival that are always less than $1/\mu$.

Turning our attention to server 1, we may view the arrival process to server 1 as alternating between a Poisson process with rate $\lambda' p/2$ (when there are no high-value jobs at server 2), and a Poisson process with rate $\lambda' p$ (when there are one or more high-value job at server 2). The duration during which server 1 receives jobs at the higher arrival rate of $\lambda' p$ corresponds to a "high-value busy period" started by the arrival of the first high-value job to server 2, and concluded when server 2 is no longer serving any high-value jobs. We write B_h to denote the length of this busy period. We upper bound N by assuming that all *additional* arrivals which would arrive during this high-value busy period arrive at the same time as a "batch arrival."[14] That is, we can view server 1 as receiving jobs according to a Poisson process with a fixed rate, except whenever server 2 receives a high-value job (when it previously had none), server 1 will receive many jobs at once. The number of jobs, A_{B_h}, making up this batch of "many jobs," will be distributed like the number of *additional* arrivals server 1 would have received in B_h time. Hence, we can upper bound the queue length at server 1 with the number of jobs, N, in an M^Y/M/1 system (cf. [40]),[15] with arrival rate

[13] The higher load under LAVA′ (i.e., $\rho' > \rho$) also contributes to longer queue lengths under LAVA′.

[14] We note that receiving these additional arrivals at once may (rarely) allow for server 1 to work on jobs before they would arrive in the original system without batching. Even with this possibility, sending future arrivals earlier can only cause N to (stochastically) increase, rather than decrease.

[15] We use the notation M^Y/M/1 to refer to the system more commonly denoted by M^X/M/1 in order to prevent ambiguity in the use of the random variable X in this paper, as X has previously denoted service requirements.

$\lambda' p/2 + \lambda'(1-p) = \lambda'(1-p/2)$ and "batch size," Y, distributed as follows:

$$Y \sim \begin{cases} 1 & \text{w.p. } \dfrac{p}{2-p} \\ \\ A_{B_h} & \text{w.p. } \dfrac{2-2p}{2-p}. \end{cases}$$

Here, we are again overestimating N by assuming that *each* high-value arrival sent to server 2 (rather than only the first high-value arrival to start each high-value busy period) causes A_{B_h} additional jobs to be sent to server 1.

Next, we show that for all p sufficiently close to 1, B_h has all finite moments, and hence A_{B_h} and Y have finite moments, and finally, N has finite moments. We may overestimate B_h with B_h^*, the duration of a high-value busy period under the assumption server 2 serves low-value jobs ahead of high-value jobs, rather than employing Processor-Sharing. Note that this alternative scheduling policy can only lengthen the busy period, so B_h^* is indeed stochastically greater than B_h. Under this alternative scheduling policy, when a high-value job arrives to server 2, it starts a busy period of length B_h^*, with Laplace transform

$$\widetilde{B_h^*}(s) = \widetilde{W}\left(s + \lambda'(1-p)\left(1 - \widetilde{B}_h(s)\right)\right),$$

where W is the random variable giving the amount of work at server 2 seen by the first high-value arrival [13]. We know that for all p sufficiently close to 1, W has all finite moments and its Laplace transform, $\widetilde{W}(s)$, exists. Hence, $\widetilde{B_h^*}(s)$ is well-defined, and B_h^* has all finite moments. Since B_h^* stochastically dominates B_h, it follows that B_h must also have all finite moments, and its Laplace transform, $\widetilde{B}_h(s)$, exists. Consequently, the z-transforms of both A_{B_h} and Y exist (establishing that both have all finite moments) and are given by

$$\widehat{A_{B_h}}(z) = \widetilde{B}_h(\lambda' p/2 \cdot (1-z)),$$
$$\widehat{Y}(z) = \frac{pz}{2-p} + \frac{2-2p}{2-p} \cdot \widehat{A_{B_h}}(z).$$

Moreover, for all p sufficiently close to 1, the aggregate arrival rate to server 1, $\lambda'(1-p/2) \cdot \mathbb{E}[Y]$, is less than the departure rate, μ. This fact, combined with the existence of $\widehat{Y}(z)$, guarantees that the number of jobs, N, in the $M^Y/M/1$ system of interest has a well-defined z-transform, and hence, N has all finite moments (see [40] for details). Therefore, the number of jobs at server 1 under LAVA′ has all finite moments for all p sufficiently close to 1, and hence, $\mathbb{E}[N^2]^{\text{LAVA}'}$ is finite.

Finally, for any $\rho' < 1$, $\mathbb{E}[N^2]^{\text{LAVA}'}$ must be bounded for all p sufficiently close to 1, because an increase in p leads to shorter high-value busy period durations, B_h, smaller batch sizes, Y, and less frequent batch arrivals of size A_{B_h}, in exchange for a vanishingly higher "low-traffic" arrival rate to server 1. Hence, $\mathbb{E}[N^2]^{\text{LAVA}'}$ is eventually decreasing in p as $p \to 1$. It follows that for each ρ', there exists some $K < \infty$ such that $\mathbb{E}[N^2]^{\text{LAVA}'} < K$ for all p in a neighborhood of 1. Consequently, $\mathbb{E}[N^2]^{\text{LAVA}}$ is bounded for any $\rho < 1$, which establishes that the expectations of interest converge as claimed. □

Appendix A.3. Proof of Lemma 2.

Lemma 2. *Let $V \sim$ SBD(p) in a system with two identical servers. As $p \to 1$, the* G-VITA *policy with parameter g under load ρ is stable whenever*

$$g > -\frac{\log(2-2\rho)}{\log(2\rho)} - 1, \qquad \textit{or alternatively,} \qquad g > \log_2\left(\frac{\rho}{1-\rho}\right).$$

Proof. As $p \to 1$, server 2 is clearly stable, as there are at most g low-value jobs at this server by definition, and high-value jobs arrive according to a Poisson process with rate $(1-p)\lambda \to 0$. Hence, the question of stability primarily concerns server 1. The arrival process to server 1 is non-Poisson, but we can still measure the time-average arrival rate to this server. Observe that server 1 receives jobs whenever there are exactly g low-value jobs at server

2. Moreover, as $p \to 1$, the impact of the high-value jobs on the number of low-value jobs at server 2 becomes negligible. Consequently, we may view the number of low-value jobs at server 2 as being distributed like the total number of jobs in an M/M/1/g system, with arrival rate $p\lambda \to \lambda$ and departure rate, μ. Hence, we may treat the arrival process to server 1 as the "loss process" of this M/M/1/g system. The time-average arrival rate associated with this loss-process is known to be (cf. PASTA)

$$\lambda\left(\frac{(\lambda/\mu)^g(1-\lambda/\mu)}{1-(\lambda/\mu)^{g+1}}\right) = \lambda\left(\frac{(2\rho)^g(1-2\rho)}{1-(2\rho)^{g+1}}\right).$$

The system is stable if and only if the time-average arrival rate is less than the service rate, μ, which corresponds to the condition

$$\frac{(2\rho)^{g+1}(1-2\rho)}{1-(2\rho)^{g+1}} < 1.$$

Simplifying, we have the stability condition $g > -\log(2-2\rho)/\log(2\rho) - 1$. The alternative condition is stronger, but still valid because it can be shown that $-\log(2-2\rho)/\log(2\rho) - 1 < \log_2(\rho/(1-\rho))$ for all $\rho \in (1/2, 1)$ (algebra omitted), while when $\rho \in (0, 1/2)$, any $g \geq 0$ would suffice as both bounding quantities are negative. □

Appendix A.4. Proof of Proposition 3

Proposition 3. *The value function for the M/M/1-PS queue with arrival rate λ, service rate μ, and values V is given by*

$$\eta_z = \frac{\lambda n(n+1)}{2(\mu-\lambda)(2\mu-\lambda)}\,\mathbb{E}[V] + \frac{n+1}{2\mu-\lambda}\,v^{\text{sum}} + c,$$

where c is a constant.

Proof. In proving this result, we invoke two earlier results. First, in an M/M/1-PS queue with n jobs, the mean response time of each job is given by (Sengupta and Jagerman, [41]),

$$\mathbb{E}[T|n] = \frac{n+1}{2\mu-\lambda}, \tag{A.1}$$

which interestingly is finite even if the system is somewhat overloaded (i.e., $\mu < \lambda < 2\mu$). Second, the value function *with respect to mean response time* (rather than value-weighted mean response time) in an M/M/1 system under any work-conserving scheduling discipline (e.g., FCFS or PS) is given by (see [37, 42])

$$\frac{n(n+1)}{2(\mu-\lambda)} + c', \tag{A.2}$$

where c' is some constant. The value function (A.2) can be broken into the sum of (i) the *total* mean response time of the n jobs, and (ii) the expected *total additional* response time experienced by all future arrivals due to the n jobs currently in the system. Here, (i) is given by

$$n \cdot \mathbb{E}[T|n] = \frac{n(n+1)}{2\mu-\lambda}, \tag{A.3}$$

and hence, (ii) is obtained by subtracting (A.3), from (A.2), yielding

$$\left(\frac{n(n+1)}{2(\mu-\lambda)} + c\right) - \frac{n(n+1)}{2\mu-\lambda} = \frac{\lambda n(n+1)}{2(\mu-\lambda)(2\mu-\lambda)} + c'. \tag{A.4}$$

Next observe that since V and T are independent in an M/M/1-PS, we obtain the expected total additional *penalty* incurred by all future arrivals due to the n jobs currently in the system by multiplying (A.4) by $\mathbb{E}[V]$:

$$\frac{\lambda n(n+1)}{2(\mu-\lambda)(2\mu-\lambda)} \cdot \mathbb{E}[V] + c' \cdot \mathbb{E}[V]. \tag{A.5}$$

Moreover, the mean penalty incurred by the n jobs currently in the system is obtained by multiplying (A.1) by the total value of those jobs, v^{sum}:

$$\frac{n+1}{2\mu-\lambda} \cdot v^{\text{sum}}. \tag{A.6}$$

Finally to obtain the value function of interest (the one with respect to $\mathbb{E}[VT]$), we add (A.5) and (A.6), yielding the desired result (with $c = c' \cdot \mathbb{E}[V]$), completing the proof. □

Note that Proposition 3 is a new result. In contrast to our setting, [37, 42] give a value function for the M/M/1/-PS queue with respect to $\mathbb{E}[T]$ rather than $\mathbb{E}[VT]$. Meanwhile [8] gives a value function where the state-information includes the (remaining) service times.

Appendix A.5. Proof of Proposition 4

Proposition 4. *For a static basic policy α, yielding server-specific arrival processes (λ_i, V_i), the corresponding FPI policy, routes a job of value v to the server given by*

$$\alpha'(v) = \underset{i}{\text{argmin}} \quad \frac{1}{2\mu_i-\lambda_i}\left(\frac{\lambda_i\,\mathbb{E}[V_i](n_i+1)}{\mu_i-\lambda_i} + v_i^{\text{sum}} + (n_i+2)v\right).$$

Proof. First, recall that with a static basic policy, the system decomposes to m independent M/M/1-PS queues, and the value function of the whole system is therefore the sum of the queue-specific value functions,

$$\eta_{\mathbf{z}}(\alpha) = \eta^{(1)}_{z_1}(\alpha) + \ldots + \eta^{(m)}_{z_m}(\alpha). \tag{A.7}$$

Each $\eta^{(i)}_{z_i}(\alpha)$ is given by (5) with the queue-specific $(\lambda_i, \mathbb{E}[V_i])$ defined by α, and the queue-specific service rate μ_i.

Given the value function is available, we can carry out the FPI step (4). For clarity, we omit the basic policy α from the notation. The admission penalty of a job with value v is equal to the change in the value function,[16]

$$a(v,i) = \eta_{\mathbf{z}\oplus(v,i)} - \eta_{\mathbf{z}},$$

where $\mathbf{z} \oplus (v,i)$ denotes the resulting state when a job with value v is added to server i. Given η_z is the sum of queue-specific terms (A.7), the change in $\eta_z(\alpha)$ is local to server i, and the admission penalty becomes

$$a(v,i) = \left(\eta^{(1)}_{z_1} + \ldots + \eta^{(i)}_{z_i\oplus v} + \ldots + \eta^{(m)}_{z_m}\right) - \left(\eta^{(1)}_{z_1} + \ldots + \eta^{(m)}_{z_m}\right) = \eta^{(i)}_{z_i\oplus v} - \eta^{(i)}_{z_i}. \tag{A.8}$$

From Proposition 3 and (A.8) we have

$$a(v,i) = \frac{\lambda_i\,\mathbb{E}[V_i](n_i+1)}{(\mu_i-\lambda_i)(2\mu_i-\lambda_i)} + \frac{v_i^{\text{sum}}}{2\mu_i-\lambda_i} + \frac{(n_i+2)v}{2\mu_i-\lambda_i}, \tag{A.9}$$

where the first and second terms corresponds to the expected total additional penalty incurred by the future arrivals and the n_i jobs currently at server i, respectively. The last term is the expected penalty incurred by the new arrival of value v. FPI chooses the queue with the smallest expected penalty, $\alpha'(v) = \underset{i}{\text{argmin}}\, a(v,i)$, yielding (6). □

[16] There are no immediate penalties associated with any state changes.