Variability-aware Request Replication for Latency Curtailment

Zhan Qiu
Department of Computing
Imperial College London, UK
Email: zhan.qiu11@imperial.ac.uk

Juan F. Pérez
School of Mathematics and Statistics
University of Melbourne, Australia
Email: juan.perez@unimelb.edu.au

Peter G. Harrison
Department of Computing
Imperial College London, UK
Email: pgh@imperial.ac.uk

***Abstract*—Processing time variability is commonplace in distributed systems, where resources display disparate performance due to, e.g., different workload levels, background processes, and contention in virtualized environments. However, it is paramount for service providers to keep variability in response time under control in order to offer responsive services. We investigate how request replication can be used to exploit processing time variability to reduce response times, considering not only mean values but also the tail of the response time distribution. We focus on the distributed setup, where replication is achieved by running copies of requests on multiple servers that otherwise evolve independently, and waiting for the first replica to complete service. We construct models that capture the evolution of a system with replicated requests using approximate methods and observe that highly variable service times offer the best opportunities for replication – reducing the response time tail in particular. Further, the effect of replication is non-uniform over the response time distribution: gains in one metric, e.g., the mean, can be at the cost of another, e.g., the tail percentiles. This is demonstrated in wide range of numerical virtual experiments. It can be seen that capturing service time variability is key to the evaluation of latency tolerance strategies and in their design.**

I. Introduction

With the rise of cloud and utility computing, many services rely on distributed computing resources that display high variability in their offered performance. This variability has a number of sources, such as the underlying hardware, different workload levels, background processes, or contention in virtualized environments. The result is that requests' processing times, and the user-perceived latency, display very significant variability that can harm the offered service levels. This is particularly true for the tail of the response time distribution; it has been shown that the 99^{th} percentiles can be several times larger than the mean response time [1]. Thus, a non-negligible fraction of the processed requests face latencies that are very far from the average case. However, keeping a consistently low latency is critical for users to experience a responsive service, and even a small amount of additional delay may degrade the offered quality of service. For instance, Google's experiments show that the injection of a 500ms delay degrades the traffic and revenue of Google searches by 20% [2].

In this context, request replication has been proposed as a mechanism to reduce the response time tail [1], by initiating multiple copies of a request on separate servers and using the result from the copy that completes first. This approach is appealing as most clusters these days are highly underutilized, e.g., the average utilization of major data center servers remains under 18% [3]. Further, as delay is often due to external interference [1], replication is effective in handling exceptional delays unless they occur in all replicas simultaneously. For instance, replication of DNS queries can mask delays caused by cache misses, network congestion, or packet losses [4].

The main risk of replication is that it may negatively impact latency since it introduces extra load into the system, which may lead to excessive delays. Evaluating the effect of replication is, however, non-trivial, as a number of factors, such as the system load or the server pool size, have been shown to affect the effective response times [5]. In this paper we focus on the effect that the statistical characteristics of service time, and in particular their *variability*, have on the potential for replication as a latency-tolerance mechanism. It is natural to expect that the service time variability, as noted for instance in [4], [6], should have an important role in the effectiveness of replication as a means of reducing latency. However, a number of questions remain unanswered when considering replication for latency-tolerance: for a given service time pattern (distribution), what is the maximum load at which replication can improve response times? Moreover, is such an improvement uniform over the response time distribution? For example, can we expect a decrease in the average response time as well as in its higher percentiles? While variability is a second-order effect, do higher order statistics, such as the service time's skewness, have a significant effect on the performance of replication? In addition, in which conditions is it possible to gain from more than one additional replica?

A. Contributions

The present work addresses these questions by performing an in-depth analysis on how service times influence the effectiveness of request replication. To this end, we develop models that capture the evolution of a system with replicated requests. Different from previous results that consider a single queue at a centralized dispatcher [5], [7], we focus on the more common case where each server holds its own request queue and the role of the dispatcher is simply to assign requests and their replicas to these queues. This case is far more challenging as the servers work asynchronously but are not independent since for every new request several servers receive a copy at the same time. We therefore rely on approximate methods to cope with this setup, which we use to analyze several scenarios quantitatively. Our main findings can be summarized as follows:

- The effect of replication is *non-uniform* across the response time percentiles and mean. In fact, there are cases where

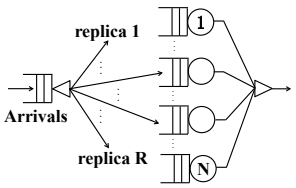


Figure 1: Replication model

replication offers improvements in the mean but actually hurts the response time tail.

- Highly variable service times offer the best opportunity to reduce response times, especially in the tail. In this case it is even possible to obtain reductions in the high percentiles of the response time distribution (90^{th}, 95^{th}, 99^{th}) even if the mean response time worsens.
- An evaluation based on the exponential assumption for service times hides important effects of replication. The characteristics of response time under highly-variable service times, as typically observed in real traces, are in fact much more complex than under the exponential assumption.
- Beyond variability, the service time skewness also has a significant effect on the potential benefits of replication.

B. Related work

Replication has been considered in [1], [4], [8], [3] as a way to reduce latency, and in [9], [5] to improve both reliability and latency. Trace-driven simulations in [8] show that processing replicas of requests simultaneously is effective in mitigating the high average response time of long-running outliers. [4] considers the case with distributed servers, where R copies of an arriving request are sent to R out of N available servers, chosen uniformly at random. The authors analytically derive an approximate threshold of the load below which replication reduces the *mean* latency, assuming Poisson arrivals and exponential service times. Approximate bounds on this threshold are provided for general service times. In Section IV we consider a similar approach to analyze the random allocation case, but our model approximates the response time *distribution* and considers general processing and inter-arrival times.

While [1], [5], [7], [6] deal with the case in which outstanding replicas of a request are cancelled upon completion of the first, in this work we focus on the more asynchronous case where no cancellation is performed. Replication without canceling is much easier to implement as it does not require any action from the central controller (which no longer needs to cancel replicas) once requests are scheduled. Further, most latency-sensitive requests are small and interactive, while the cost of canceling, including the communication time, and the canceling signal flight time, is usually non-negligible. For example, web services usually take milliseconds to respond, making canceling hard or even infeasible to implement.

II. Problem Setup

A. Reference model

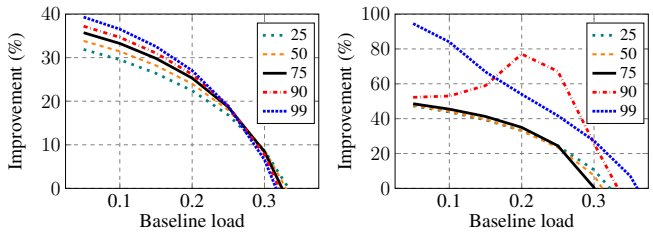
We consider a system consisting of N parallel, homogeneous and distributed servers, each with an associated buffer, as shown in Figure 1. For each arriving request, a total of R copies are sent to R servers for processing. Replicas in front of each server form a single queue in the order of arrival and receive service with first-come first-served (FCFS) scheduling. The result of the request is returned by the first replica to complete service. To analyze this system, Section III starts by introducing a model to determine the request response time distribution in the small-scale case where $N{=}R$, under fairly general inter-arrival and service times. This case can model, for instance, applications that maintain replicated data in a subset of resources, such that a request can be processed by any resource in this subset. Next, to analyze the general case where $N{>}R$, a different approach is proposed in Section IV, where we consider that the R replicas of each request are allocated to R out of N servers, chosen uniformly at random. This case, which we refer to as *random*, is more challenging to analyze and we therefore rely on the approximating assumption that replicas of a request are independent, to obtain the response time distribution under fairly general service times. This case captures medium- or large-scale systems where servers are not dedicated to serve a single arrival flow.

Notice that the basic case where $N{=}R$ can be easily extended to consider the system with $N{>}R$ servers by assuming that the servers are partitioned into N/R groups of size R, with N a multiple of R. Thus, upon the arrival of a request, all its R replicas are assigned to one of the groups, chosen at random. This allocation policy, which we refer to as *grouped*, can be found, for instance, in multi-core systems where a job is sent to one core utilizing all its threads and has the advantage of being easier to analyze as each group can be considered independently. However, we may expect this policy to offer worse performance than the random policy, as the latter benefits from greater multiplexing gains due to better load balance, with servers being idle less often. Thus for the grouped case we propose a different model in Section III.

B. Motivating example

Replication offers a trade-off between the additional load introduced and the potential to reduce response time as it utilizes the first replica that completes service. Thus it is unclear when replication reduces latency as this may depend on a number of system parameters. To illustrate this, let us consider a distributed system setup with 50 servers, Poisson arrivals, and a short-tailed service process with low variability. The variability is measured by the squared coefficient of variation (SCV), defined as $C_X^2{=}\text{Var}[X]/E^2[X]$, for a random variable X, where $\text{Var}[X]$ and $E[X]$ are the variance and expected value of X, respectively. In this low variability case, the SCV is set to 0.5. Figure 2(a) depicts the (positive) relative improvement (reduction) in the response times obtained by introducing replication ($R{=}2$). We show the 25^{th}, 50^{th} and 75^{th} percentiles, also known as the three quartiles, and the 90^{th}, 99^{th} percentiles, representative of the tail, for an increasing baseline load, which is the load of the system without replication. Clearly, replication provides significant gains under low loads, although the gain decreases with increasing load, and becomes negative when the load is larger than around 0.32.

Turning to service times with a long tail and high variability (SCV=20), Figure 2(b) shows that gains are much larger than in the low variability case, especially in the response time tail. Different from the low variability case, the gains across



(a) Variability:0.5 (b) Variability:20

Figure 2: Improvement on percentiles

the percentiles are not uniform, and even the range of loads under which replication provides gains is different depending on which metric is considered. For instance, under a 0.35 load, the 99^{th} percentile can be improved by close to 10% by introducing replication, but at this load replication is harmful to lower percentiles. As a result, if a decision is made to replicate based on possible gains on one percentile of the response time distribution only, other percentiles may actually suffer. This highlights the importance of explicitly considering the response time distribution, via the targeted percentiles, and not just its mean, when evaluating a replication strategy. This is the purpose of the models introduced in the forthcoming sections.

C. Preliminaries

Since we aim to study the impact of the service time *distribution* on the potential for replication to reduce latency, we use phase-type (PH) distributions to model fairly general behaviors. The motivation for this choice is three-fold. First, PH distributions can be employed to approximate any distribution, including long-tailed distributions [10]. Secondly, they are dense in the set of all positive-valued distributions but maintain some of the analytical tractability of the exponential distribution [11]. Further, PH distributions include the exponential, Erlang, and hyper-exponential distributions as special cases, and a number of fitting methods exist to obtain PH representations from data traces. A PH random variable X represents the absorption time in a Markov Chain (MC) with $n{+}1$ states, where the states $\{1,\ldots,n\}$ are transient and state 0 is absorbing [11]. Let $\boldsymbol{\tau}$ be the $1{\times}n$ vector of the MC initial probability distribution for the transient states, and let S be the $n{\times}n$ sub-generator matrix holding the transition rates among the transient states. We denote this random variable or its distribution as $\text{PH}(\boldsymbol{\tau}, S)$. The vector $\boldsymbol{S}^*{=}{-}S\mathbf{1}$ holds the absorption rates from the transient states, where $\mathbf{1}$ is a column vector of ones. Its cumulative distribution function (CDF) is given by $F(x){=}1{-}\boldsymbol{\tau}\exp(Sx)\mathbf{1}$, for $x \geq 0$, and its expected value is $E[X]{=}{-}\boldsymbol{\tau}S^{-1}\mathbf{1}$.

Further, motivated by the variable and correlated arrivals observed in workload traces [12], we model the request arrival process as a Markovian Arrival Process (MAP). A special case of a MAP is a renewal process with PH inter-arrival times, where there is no correlation. In general, however, the phase (or state of the underlying Markov chain) may be different at successive arrival instants, which leads to correlation in the inter-arrival times. The continuous-time MAP [11] is a marked MC with generator matrix $D{=}D_0{+}D_1$. The transition rates *not* associated with arrivals are held in D_0, while the rates that trigger new arrivals are kept in D_1. The diagonal entries of D_0 hold the total exit rate in each state, such that $(D_0{+}D_1)\mathbf{1}{=}\mathbf{0}$. We denote this process as $\text{MAP}(m_a, D_0, D_1)$, where m_a is the number of states, or arrival phases, in the MC. The mean arrival rate is $\lambda{=}\boldsymbol{d}D_1\mathbf{1}$, where $\boldsymbol{d}$ is the stationary distribution of the underlying MC, i.e., $\boldsymbol{d}D{=}0$ and $\boldsymbol{d}\mathbf{1}{=}1$.

In the reference model, we assume that requests arrive according to a $\text{MAP}(m_a, D_0, D_1)$, while the replica service times follow a $\text{PH}(\boldsymbol{\alpha}_{\text{rep}}, S_{\text{rep}})$ distribution. In the case of exponential service times, we assume a mean processing time $1/\mu$, such that $\boldsymbol{\alpha}_{\text{rep}}{=}1$, and $S_{\text{rep}}{=}{-}\mu$. When replication is adopted, the result of a request is obtained from the first replica that completes service. The request response time is thus the time interval between its arrival and the time its first replica completes service. The system load U is the expected fraction of the time that a server is busy, thus $U{=}R\lambda/\mu$, where λ is the mean arrival rate, and μ is the replica service rate.

III. Grouped allocation

To study the impact of replication under the grouped policy, we first introduce a stochastic model to determine the response time *distribution* focusing on a single group with R servers, and later we show how the analysis can be extended to cover systems with multiple groups. The request response time distribution is obtained by considering the service time and waiting time processes separately, and connecting them via a PH representation.

A. The waiting time distribution

To determine the waiting time distribution, we perform an age-based analysis [13], observing the R queues only during the *all-busy* periods, i.e, periods where all R servers are busy. We keep track of the *age*, or total time-in-system, $X(t)$ of the youngest *request* in service. The age takes values in $[0,\infty)$ and increases linearly with rate 1 if no service completions occur. Since all R replicas of a request are assigned to these R servers, a *replica* service completion in the *shortest* queue marks a *request* service completion, as all the siblings of the replica completing service are still in service or waiting. Thus, in case of a service completion in the shortest queue, $X(t)$ has a downward jump and its new value after the jump is equal to the waiting time of the request starting service. We also keep track of the phase of the arrival process $A(t)$, and the state of the R queues $D(t)$, which includes their service phase, described below. We put together these two variables in the phase $J(t){=}(A(t), D(t))$, which takes values in a finite set of size $m{=}m_a m_s$, where m_a is the number of arrival phases, and m_s the size of the set where $D(t)$ takes values. We put together the age and the phase to obtain the bivariate Markov process $\{X(t), J(t) | t \geq 0\}$.

To keep track of the state of the R queues, and taking advantage of their homogeneity, we focus on the queue-length differences with respect to the shortest one as in [14]. The length of a queue includes the number of requests waiting and in service. Although the queue-length difference is unbounded, we truncate it to be at most C. Recall that the replica service time distribution is $\text{PH}(\boldsymbol{\alpha}_{\text{rep}}, S_{\text{rep}})$ with m_{rep} phases. Thus, we define $D(t){=}(n_{i,j}(t), 0{\leq}\, i \,{\leq} C, 1{\leq}\, j \,{\leq} m_{\text{rep}})$, where $n_{i,j}(t)$ is the number of queues with queue-length difference i with respect to the shortest queue, and such that the replica in service is in phase j, at time t. Thus $D(t)$ takes values in

Table I: Transition rates for S, $A^{(\text{jump})}$ and $S_{\text{not-all-busy}}$

Matrix	From	To	Rate
S	$(\boldsymbol{n}_0,\ldots,\boldsymbol{n}_i,\ldots,\boldsymbol{n}_C)$	$(\boldsymbol{n}_0,\ldots,\boldsymbol{n}_i-\boldsymbol{e}_j+\boldsymbol{e}_k,\ldots,\boldsymbol{n}_C)$	$n_{i,j}S_{\text{rep}}(j,k),\ i\geq 0$
	$(\boldsymbol{n}_0,\ldots,\boldsymbol{n}_{i-1},\boldsymbol{n}_i,\ldots,\boldsymbol{n}_C)$	$(\boldsymbol{n}_0,\ldots,\boldsymbol{n}_{i-1}+\boldsymbol{e}_k,\boldsymbol{n}_i-\boldsymbol{e}_j,\ldots,\boldsymbol{n}_C)$	$n_{i,j}S^*_{\text{rep}}(j)\boldsymbol{\alpha}_{\text{rep}}(k),\ i\geq 1$
$A^{(\text{jump})}$	$(\boldsymbol{n}_0,\boldsymbol{n}_1,\ldots,\boldsymbol{n}_{C-1},\boldsymbol{n}_C)$	$(\boldsymbol{e}_k,\boldsymbol{n}_0-\boldsymbol{e}_j,\boldsymbol{n}_1,\ldots,\boldsymbol{n}_{C-2},\boldsymbol{n}_{C-1}+\boldsymbol{n}_C)$	$n_{0,j}S^*_{\text{rep}}(j)\boldsymbol{\alpha}_{\text{rep}}(k)$
$S_{\text{not-all-busy}}$	$(n_0,\ldots,\boldsymbol{n}_i,\ldots,\boldsymbol{n}_C)$	$(n_0,\ldots,\boldsymbol{n}_i-\boldsymbol{e}_j+\boldsymbol{e}_k,\ldots,\boldsymbol{n}_C)$	$n_{i,j}S_{\text{rep}}(j,k),\ i\geq 1$
	$(n_0,\ldots,\boldsymbol{n}_{i-1},\boldsymbol{n}_i,\ldots,\boldsymbol{n}_C)$	$(n_0,\ldots,\boldsymbol{n}_{i-1}+\boldsymbol{e}_k,\boldsymbol{n}_i-\boldsymbol{e}_j,\ldots,\boldsymbol{n}_C)$	$n_{i,j}S^*_{\text{rep}}(j)\boldsymbol{\alpha}_{\text{rep}}(k),\ i\geq 2$
	$(n_0,\boldsymbol{n}_1\ldots,\boldsymbol{n}_C)$	$(n_0+1,\boldsymbol{n}_1-\boldsymbol{e}_j,\ldots,\boldsymbol{n}_C)$	$n_{1,j}S^*_{\text{rep}}(j)$

the set $S_D{=}\{\boldsymbol{n}{=}(\boldsymbol{n}_1,\ldots,\boldsymbol{n}_C)|\boldsymbol{n}_i{=}(n_{i,1},\ldots,n_{i,m_{\text{rep}}}), n_{i,j}\in\{0,\ldots,R\}, \sum_{i=0}^{C}\sum_{j=1}^{m_{\text{rep}}} n_{i,j}=R\}$, the cardinality of which is m_s. Section III-D evaluates the accuracy of this approximation and explores the value of C needed to ensure accurate results.

The next step is to determine the stationary distribution $\boldsymbol{\pi}(x)$ of the $(X(t),J(t))$ process, which has a matrix-exponential representation $\boldsymbol{\pi}(x){=}\boldsymbol{\pi}(0)\exp(Tx)$, for $x{>}0$, as shown in [15]. The $m{\times}m$ matrix T satisfies the equation

$$T = S^{(\text{MAP})} + \int_0^\infty \exp(Tu)A^{(\text{MAP})}(u)du, \qquad (1)$$

where $S^{(\text{MAP})}{=}S\otimes I_{m_a}$, $A^{(\text{MAP})}(u){=}A^{(\text{jump})}\otimes\exp(D_0u)D_1$, I_n is the identity matrix of size n, and $\otimes$ denotes the Kronecker product. S and $A^{(\text{jump})}$ are $m_s{\times}m_s$ matrices that hold the transition rates of the service process associated to transitions without and with a new request starting service, respectively. Thus, the matrix $A^{(\text{jump})}$ holds the rates of service completion at any of the n_0 shortest queues, as a replica that completes service in these queues triggers the start of a new request service. Other transition rates are kept in the matrix S, as summarized in Table I. The first row in this table considers the case where the replica in service goes through a service-phase transition from phase j to phase k, without completing service. The second row covers the case of service completions at any queue but the shortest one, where the replica in service phase j completes, and a new replica starts service in phase k. In the last row, a replica in any of the shortest queues, and service phase j, completes service, resulting in a single shortest queue, which is in phase k. Also, as the queue-length differences are bounded by C, transitions leading these differences to exceed the limit C are re-assigned to C.

1) Computing $\boldsymbol{\pi}(0)$: To find $\boldsymbol{\pi}(0)$, which is the steady-state distribution of the phase at the beginning of an all-busy period, we need to model the system evolution during the not-all-busy period. During this period we only keep track of the arrival phase and the states of the queues, similar to the phase in the all-busy-period, but in this case the shortest queues are empty, and therefore n_0 holds the number of idle servers. We thus define the process $\bar{D}(t)$, which takes values in the set $\bar{S}_D{=}\{(n_0,\boldsymbol{n}_1,\ldots\boldsymbol{n}_C)|\boldsymbol{n}_i{=}(n_{i,1},\ldots,n_{i,m_{\text{rep}}}), 1{\leq}i{\leq}C, n_0{\in}\{1,\ldots,R\}, n_{i,j}{\in}\{0,\ldots,R\}, n_0{+}\sum_{i=1}^{C}\sum_{j=1}^{m_{\text{rep}}} n_{i,j}{=}R\}$. In this case, the first entry of the vector $(n_0,\boldsymbol{n}_1,\ldots\boldsymbol{n}_C)$ holds the number of idle servers, while the remaining entries are the same as for $D(t)$. The transition rates across service phases, between arrivals, during a not-all-busy period are held in the matrix $S_{\text{not-all-busy}}$, as summarized in Table I. Here the first case corresponds to phase transitions without a service completion, while the second and third cases correspond to service completions.

Since $\boldsymbol{\pi}(0)$ is the distribution of the phase $J(t)$ at the beginning of an all-busy period, it solves

$$\boldsymbol{\pi}(0) = \boldsymbol{\pi}(0)\int_0^\infty \exp(Tu)(A^{(\text{jump})}_{\text{not-all-busy}}\otimes\exp(D_0u))du \qquad (2)$$
$$(S_{\text{not-all-busy}}\oplus D_0)^{-1}(K^{(start)}\otimes D_1).$$

This equation shows that, during an all-busy period, the age reaches a value u with density $\boldsymbol{\pi}(0)\exp(Tu)$, and has a downward jump of size larger than u with density $A^{(\text{jump})}_{\text{not-all-busy}}\otimes\exp(D_0u)$. The matrix $A^{(\text{jump})}_{\text{not-all-busy}}$ is similar to $A^{(\text{jump})}$ in Table I, but from a state $(\boldsymbol{n}_0,\boldsymbol{n}_1,\ldots,\boldsymbol{n}_C){\in}S_D$, a transition to a state $(1,\boldsymbol{n}_1,\ldots,\boldsymbol{n}_C)\in\bar{S}_D$ occurs with rate $n_{0,j}S^*_{\text{rep}}(j)$. This transition corresponds to a service completion in the shortest queue, which starts a not-all-busy period with one empty queue. As the process enters a not-all-busy period, the phase evolves according to the sub-generator $S_{\text{not-all-busy}}\oplus D_0$, until an arrival occurs and triggers the process into a new all-busy period. When this occurs, the matrix $K^{(\text{start})}$ maps a state $(n_0,\boldsymbol{n}_1,\ldots,\boldsymbol{n}_C){\in}\bar{S}_D$ into a state $(\boldsymbol{n}_0,\boldsymbol{n}_1,\ldots,\boldsymbol{n}_C){\in}S_D$ with probability $p_{n_0,\boldsymbol{n}_0}$. As each of the n_0 idle servers selects its initial service phase according to $\boldsymbol{\alpha}_{\text{rep}}$, the probability that the shortest queues start the all-busy period in state $\boldsymbol{n}_0{=}(n_{0,1},\ldots n_{0,m_{\text{rep}}})$ follows a multinomial distribution, thus

$$p_{n_0,\boldsymbol{n}_0} = \frac{n_0!}{n_{0,1}!\cdots n_{0,m_{\text{rep}}}!}\boldsymbol{\alpha}_{\text{rep}}(1)^{n_{0,1}}\cdots\boldsymbol{\alpha}_{\text{rep}}(m_{\text{rep}})^{n_{0,m_{\text{rep}}}},$$

where $\boldsymbol{\alpha}_{\text{rep}}(i)$ denotes the i^{th} entry of vector $\boldsymbol{\alpha}_{\text{rep}}$. The integral in Eq. (2) can be found by integrating by parts and solving a Sylvester matrix equation in $O(m^3)$, which is then used to find $\boldsymbol{\pi}(0)$ by solving a linear system.

2) Obtaining the distribution: We are now ready to find the waiting time distribution. The steady state distribution of the phase $J(t)$ during the all-busy period is $\boldsymbol{\alpha}_{\text{busy}}{=}{-}\boldsymbol{\pi}(0)T^{-1}$, and let $\boldsymbol{\varphi}{=}(T-S^{(\text{MAP})})\mathbf{1}$. Thus, the PH representation of the waiting time distribution is given by [13]

$$\boldsymbol{\alpha}_{\text{wait}} = \gamma\boldsymbol{\alpha}_{\text{busy}}\circ\boldsymbol{\varphi}/((\boldsymbol{\alpha}_{\text{busy}}\circ\boldsymbol{\varphi})\mathbf{1}), \quad S_{\text{wait}} = \Lambda^{-1}T'\Lambda,$$

where $\circ$ stands for the Hadamard product, $'$ denotes the matrix transpose, Λ is a diagonal matrix such that $\Lambda\mathbf{1}{=}\boldsymbol{\alpha}'_{\text{busy}}$, and γ is the probability that a request has to wait. This probability is given by $\gamma{=}(E[\eta_0]{-}1)/E[\eta_0]{=}1{-}1/E[\eta_0]$, where $E[\eta_0]$ is the expected number of service completions in an all-busy period, such that in a cycle made of one all-busy and one not-all-busy period, $E[\eta_0]$ is the expected number of request arrivals, and $E[\eta_0]{-}1$ is the expected number of arrivals that have to wait. $E[\eta_0]$ can be obtained as [13, Section 7.2] $E[\eta_0]=-(\boldsymbol{\pi}(0)T^{-1}(A^{(\text{jump})}\otimes I_{m_a})\mathbf{1})/(\boldsymbol{\pi}(0)\mathbf{1})$.

B. The service time distribution

We now show that the request service times follow a PH distribution with parameters $(\boldsymbol{\alpha}_{\text{service}}, S_{\text{service}})$. Let $N(t)$ be the service state of a tagged request in service at time t, which we define as $N(t){=}(n_{i,j}(t), 0{\leq}i{\leq}C, 1{\leq}j{\leq}m_{\text{rep}})$, where $n_{0,j}(t)$ is

Table II: Approximation errors (Grouped)

Arr	Ser	Err(%) - 0.1 Load		Err(%) - 0.5 Load		Err(%) - 0.9 Load	
		mean	$W_2(99)$	mean	$W_2(99)$	mean	$W_2(99)$
Exp	Exp	<0.01	<0.01	<0.01	<0.01	<0.01	0.11
		<0.01	<0.01	<0.01	<0.01	<0.01	0.11
		<0.01	<0.01	<0.01	<0.01	<0.01	0.11
Exp	ER_2	<0.01	<0.01	<0.01	<0.01	0.04	0.03
		<0.01	<0.01	<0.01	<0.01	0.04	0.03
		<0.01	<0.01	<0.01	<0.01	0.04	0.03
MAP	HE_2	<0.01	0.05	0.39	1.88	23.92	26.06
		<0.01	0.05	<0.01	<0.05	0.83	0.14
		<0.01	<0.01	<0.1	<0.1	<0.1	<0.1

the number of queues with a replica of the tagged request in service phase j, and $n_{i,j}(t)$, for $i>0$, is the number of queues where the replica of the tagged request is in the i^{th} position in the queue. Notice that $N(t)$ is similar to the process $D(t)$, but it focuses on the queue-lengths *in front of the tagged request only*. Since the tagged request completes service when one of its replicas completes service, we are interested in the service phases in $N(t)$ where the R replicas of the tagged are still in service or waiting, thus $\sum_{i=0}^{C}\sum_{j=1}^{m_{\text{rep}}} n_{i,j}=R$. As a result, $N(t)$ takes values in the set S_D.

Next, to obtain the service time PH representation we describe the transitions among the states in S_D with a sub-generator S_{service}. Notice that S_{service} holds the service transition rates without a service completion in the shortest queue, thus it is equal to the matrix S in Table I. The final step is to obtain the initial probability vector $\boldsymbol{\alpha}_{\text{service}}$, which, following [14, Proposition 1], is given by

$$\boldsymbol{\alpha}_{\text{service}} = [(1-\gamma)\boldsymbol{\pi}(0) + c\gamma\boldsymbol{\alpha}_{\text{busy}}(T - S^{(\text{MAP})})],$$

where $c = \boldsymbol{\alpha}_{\text{busy}}(T - S^{(\text{MAP})})\mathbf{1}$.

C. The response time distribution and multiple groups

Based on the waiting and service time distributions, we adapt [14, Theorem 1] to obtain the response time distribution

$$\boldsymbol{\alpha}_{\text{res}} = [(1-\gamma)\boldsymbol{\pi}(0) \quad \tilde{\boldsymbol{\alpha}}_{\text{busy}} \quad \mathbf{0}_{1\times m}],$$

$$S_{\text{res}} = \begin{bmatrix} S_{\text{service}} & 0_{m\times m} & 0_{m\times m} \\ 0_{m\times m} & \tilde{S}_{\text{service}} & (-\tilde{S}_{\text{service}}\mathbf{1})P_{s,w} \\ 0_{m\times m} & 0_{m\times m} & S_{\text{wait}} \end{bmatrix},$$

where $\tilde{S}_{\text{service}}=\Delta^{-1}S'_{\text{service}}\Delta$, Δ is a diagonal matrix such that $\Delta\mathbf{1}=\boldsymbol{\delta}'$, and $\boldsymbol{\delta}=-\boldsymbol{\alpha}_{\text{service}}S^{-1}_{\text{service}}$ is the stationary distribution of the phase during service. Also, $\tilde{\boldsymbol{\alpha}}_{\text{busy}}=(-S_{\text{service}}\mathbf{1})'\Delta$, and $P_{s,w}=\Gamma^{-1}(T-S^{(\text{MAP})})'\Lambda$, where Γ and Λ are diagonal matrices such that $\Gamma\mathbf{1}=(T-S^{(\text{MAP})})'\boldsymbol{\Lambda}\mathbf{1}$ and $\Lambda\mathbf{1}=\boldsymbol{\alpha}'_{\text{busy}}$.

The above analysis of a single group can be extended to the multi-group case, where the group selection can be either deterministic or random. In the deterministic case, each group can be analyzed separately with its own arrival process. In the random case, given that the arrival process to the whole system is a MAP(m_a, D_0, D_1), the arrival process to a single group is also a MAP, with parameters [11] $D'_0=D_0+(1-p)D_1, D'_1=pD_1$, where $p=R/N$. We refer to the random selection among groups as *grouped-random*, and we compare its performance against the random policy in Section V.

D. Experimental validation

We evaluate the accuracy of the proposed model by comparing its results against simulation, focusing on the cases with R=2 as an example, considering low (0.1), medium (0.5), and high (0.9) loads (achieved with replication), as well as different service and arrival processes. For the service process, we consider exponential (Exp), Erlang-2 (ER_2), and 2-phase hyper-exponential (HE_2) distributions, thus covering a broad range of behaviors in terms of variability, measured by the SCV, as defined in Section II. The Exp and ER_2 distributions have SCV equal to 1 and 0.5, respectively, while for the HE_2 distribution we set it to 10. For the arrival process, we consider Poisson (Exp), and general order-2 MAP arrivals, where the SCV of the MAP inter-arrival times is set to 10 and the decay rate of the auto-correlation function is set to 0.5, thus modeling arrivals with high variability and auto-correlation. The HE_2 distribution is obtained with the moment-matching method in [16], while the MAP is obtained with the method in [17]. We compute the errors for different values of the limit C (100, 500 and 1000), and report the errors for some typical settings in Table II. For each setting, the simulations were run for 5,000 times with 200,000 samples each time, from which we obtain the response time mean and its 99^{th} percentile $W_2(99)$, and their 95% confidence intervals. Table II shows that, if C is large enough, the proposed model provides accurate results for both the mean and percentiles of the response times. Specifically, for the first two cases considered, Exp arrivals with Exp and ER_2 services, the errors are negligible with C=100. This is due to the relatively short tails of these service time distributions, which causes the queue-length differences to remain well within the C=100 limit. The more challenging case with auto-correlated (MAP) arrivals and HE_2 services requires a larger limit C to achieve the same level of accuracy. The high variability in both the service and arrival processes, and the auto-correlated arrival stream, increase the probability of large queue-length differences. A larger load further increase these differences. Still, errors under 1% are achievable by setting C=500 even under a 0.9 load.

IV. An Approximation for Random Allocation

We now consider the case where the request replicas are allocated to R out of the Nservers, selected at random without replacement. Analyzing this system is challenging because it requires keeping track of the state of the N queues, as a request's replicas can be allocated to any subset of queues. Also, the evolution of these queues is correlated as every time a request arrives, R queues receive one replica simultaneously. We thus opt for an approximation and show experimentally that it provides good results when the ratio N/R is large.

A. Approximation approach

To approximate the response time distribution of this system we introduce the simplifying assumption that each queue evolves independently, receiving one request replica with probability $p=R/N$ every time a request arrives. This assumption was also used in [4] to approximate the *mean* latency in a system with a single replica. While this is a simplification of the actual system dynamics, we may expect this assumption to be reasonable when the number of servers N is sufficiently large compared to the number of replicas R. More specifically, if the request arrivals are modeled as a MAP(m_a, D_0, D_1), and the replica service times are PH, each queue can be modeled as an independent MAP/PH/1 queue

Table III: Approximation errors (Random)

N/R	Arr	Ser	Load	Measure	R			
					2	3	4	5
2	Exp	ER_2	0.4	mean	3.14	4.91	5.87	6.54
				$W_R(99)$	7.73	12.74	15.82	17.93
25	Exp	ER_2	0.4	mean	0.27	0.48	0.39	0.38
				$W_R(99)$	0.61	1.09	1.15	1.19
25	Exp	ER_2	0.8	mean	0.54	0.88	1.32	1.37
				$W_R(99)$	0.97	1.35	2.85	2.90
25	Exp	Exp	0.8	mean	0.47	0.79	1.33	1.23
				$W_R(99)$	0.68	1.40	1.85	2.30
25	Exp	HE_2	0.8	mean	0.45	0.70	1.15	0.57
				$W_R(99)$	0.43	0.49	0.87	0.45
25	MAP	HE_2	0.8	mean	1.90	3.26	4.88	8.17
				$W_R(99)$	1.80	2.83	4.29	7.29
100	MAP	HE_2	0.8	mean	1.60	0.87	4.17	3.04
				$W_R(99)$	0.96	0.29	2.77	2.05
100	MAP	Exp	0.8	mean	1.84	2.98	4.39	5.84
				$W_R(99)$	2.54	4.99	7.43	9.91
100	MAP	ER_2	0.8	mean	1.66	4.10	5.09	6.08
				$W_R(99)$	3.20	7.70	9.73	12.18

with arrivals MAP(m_a, D_0', D_1'), where $D_0'{=}D_0 + (1-p)D_1$, $D_1'{=}pD_1$, and $p{=}R/N$, as in Section III.

Therefore, we can analyze each queue separately and obtain a PH representation $(\boldsymbol{\beta}_{\text{rep}}, B_{\text{rep}})$ of its response time distribution following [15]. Moreover, based on the independence assumption, the response time of a request can be obtained as the minimum of the independent response times of its R replicas. As the response time in each of the R servers is PH$(\boldsymbol{\beta}_{\text{rep}}, B_{\text{rep}})$, the request response time is PH$(\boldsymbol{\beta}_{\text{request}}, B_{\text{request}})$, given by the minimum of R independent PH variables as [11]

$$\boldsymbol{\beta}_{\text{request}} = \boldsymbol{\beta}_{\text{rep}} \otimes \cdots \otimes \boldsymbol{\beta}_{\text{rep}}, \quad B_{\text{request}} = B_{\text{rep}} \oplus \cdots \oplus B_{\text{rep}},$$

where $\oplus$ stands for the Kronecker sum.

B. Evaluating the Approximation

We now evaluate the accuracy of this approximation by comparing its results against simulation, considering different load levels and N/R ratios, as well as the service time distributions and arrival processes defined in Section III. Table III shows the relative error in the mean and 99$^{\text{th}}$ percentile $W_R(99)$ of the response time for different settings. The first row considers a 0.4 load, Poisson arrivals and ER_2 service times, with a small ratio $N/R{=}2$. We observe that the approximation produces errors lower than 8% for $R{=}2$. However, the error increases with a larger R as the request response time depends on a larger number of correlated variables. Increasing the N/R ratio to 25, as shown in the second row, the errors decrease, with the maximum error being as low as 1.19% for $W_5(99)$ with ER_2 services. Increasing the load to 0.8 with $N/R{=}25$, as shown in rows 3 to 5, for ER_2, Exp and HE_2 services, respectively, we observe that the errors increase with the load, although these remain below 3%. Clearly, for this and larger N/R ratios, the approximation provides high accuracy for both the mean and the tail of the response times. The next case in the table considers MAP arrivals and HE_2 services, which cause the errors to shoot up, compared to the case under Poisson arrivals. The reason is that MAP arrivals show high auto-correlation, while the approximation assumes independence among the response times of the replicas. To reach a higher accuracy, we increase the N/R ratio to 100 and the errors decrease to under 3% for HE_2 services. However, for Exp and especially ER_2 service times we observe larger errors. These results suggest that the approximation based on the independence assumption performs better with more variable service times as the variability decreases the dependence among the replicas' service times.

C. Threshold load & Optimal number of replicas

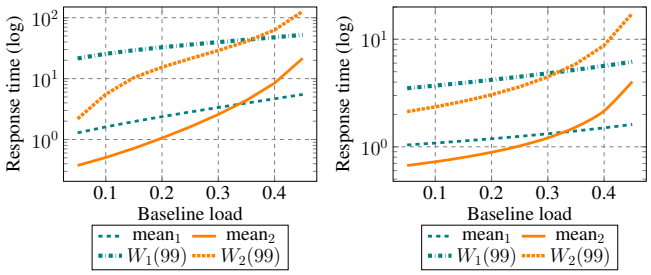
We now specialize the proposed approximation to the case of exponential services and Poisson arrivals, and try to answer two of the questions we posed in the introduction: Up to what load level can a reduction in response times be realized by implementing replication? What is the optimal number of replicas to adopt to minimize the response times? By focusing on exponential services and Poisson arrivals we can exploit closed-form expressions for the mean and the percentiles of the response times. As we shall see, however, the exponential assumption leads to an over-simplification of the effects of replication, and a better representation of the service times is necessary to fully assess its potential benefits.

1) Threshold load: Assuming R replicas are adopted for each request, we want to find the *threshold load*, which is the maximum baseline load under which replication is beneficial. The response time distribution of a request with R replicas is $F_R(t){=}1{-}\exp^{-R(\mu-\lambda^*)t}$, where $\lambda^*{=}R\lambda/N$ is the arrival rate to each queue, and the system arrival rate is λ. Thus the response time p^{th} percentile is $W_R(p){=}\ln(1/(1{-}p))/(R\mu{-}R^2\lambda/N)$, where ln is the natural logarithm. If the adoption of replication results in a lower latency, then we have $W_R(p){\leq}W_1(p)$, from which we obtain the threshold load as $\rho^* = 1/(R{+}1)$. For example, when $R{=}2$ the threshold load is $\rho^*{=}1/3$, i.e., only if the system load is below 1/3, the introduction of 2 replicas reduces the latency. Notice that the same threshold is found for the *mean* response times, as in [4] for $R{=}2$. As a result, under exponential services and Poisson arrivals, the threshold load is independent of the specific response time percentiles considered, and gains in the mean translates in gains in all the percentiles. As we show in Section V, this conclusion is not valid for more general service times.

2) Optimal number of replicas: The p^{th} percentile $W_R(p)$ of the response time distribution, can be shown to have a positive second derivative whenever the system is stable, i.e, $\mu{>}R\lambda/N$. Thus $W_R(p)$ is a convex function in R, and the optimal number of replicas to adopt is the smallest number R^* beyond which the latency increases, i.e., R^* is the smallest R that satisfies $W_R(p){\leq}W_{R+1}(p)$, thus

$$R^* = \min\left(\lfloor N\mu/\lambda \rfloor, \lfloor (N\mu - \lambda)/(2\lambda) \rfloor\right), \tag{3}$$

where the first condition ensures stability, and the second results from $W_R(p){\leq}W_{R+1}(p)$. In fact, the same result holds for the *mean* response time. Thus, under exponential services and arrivals, the number of replicas can be chosen to be optimal for both the mean and every percentile of the response time distribution. We will test this result for more general arrival processes and service times in the next section. To demonstrate this result, consider a system with 50 servers, mean arrival and service rates of 10 and 1, respectively. From Eq. (3) the optimal number of replicas to adopt is 2. Simulation results show that the mean response time when introducing 1, 2, 3 and 4 replicas is 1.251, 0.836, 0.839 and 1.275, respectively, while the 99$^{\text{th}}$ percentile is 5.762, 3.857, 3.889 and 5.957, respectively. These results verify that $R{=}2$ is the optimal number of replicas to adopt.



(a) HE_2 services (b) ER_2 services

Figure 3: Response time mean and percentiles under R=1, 2

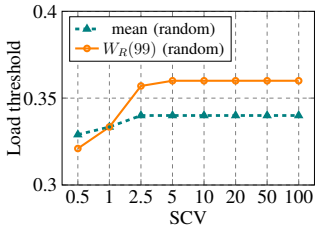


Figure 4: Load threshold

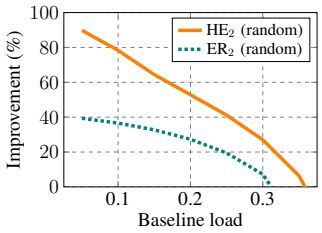


Figure 5: 99^{th} perc. imprv.

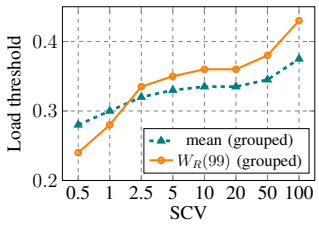


Figure 6: Load threshold

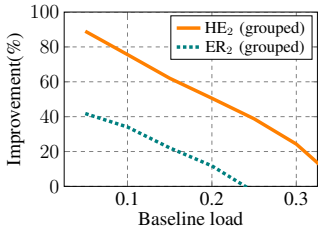


Figure 7: 99^{th} perc. imprv.

V. Experimental results

In this section we make use of the proposed models to evaluate the impact of replication on the offered response times. As we are interested in the effect of service time variability, we make use of the ER_2, Exp, and HE_2 distributions introduced in Section III. To look into the effect of the arrival process we consider both exponential (Exp) inter-arrival times (IATs), and correlated MAP arrival processes. Throughout, we assume N=50 servers, and set the service rate μ to 1, while the arrival rate is set to achieve different load levels.

A. The load threshold

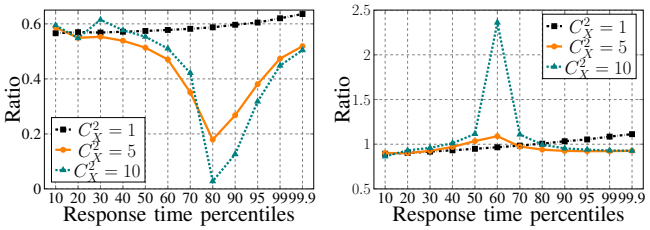
We start by investigating the load levels under which replication reduces latency, and whether this depends on the service time variability and the replica-allocation policy. Figure 3 shows the response time mean (mean_R) and 99^{th} percentile ($\text{W}_R(99)$) obtained for R=1 and 2, respectively, for baseline loads between 0.05 and 0.45, with random allocation. Notice that the load achieved after the introduction of replication doubles when R=2. While Figure 3(a) focuses on highly variable HE_2 services, Figure 3(b) considers lowly-variable ER_2 service times. In both cases, replication clearly reduces both the mean and the 99^{th} percentile of the response times, although the reduction decreases with increasing load, up to a load level where replication becomes harmful. Comparing these figures we observe that the load threshold is higher under HE_2 services than under ER_2. This is more explicit in Figure 4, which reports the load threshold for service times with an SCV that ranges from 0.5 to 100. Here we observe not only that the threshold increases with a larger variability, and stabilizes for SCVs larger than 5, but the threshold is actually different for the mean and the 99^{th} percentile. As a result, for service times with an SCV greater than one, there is a region of the load where replication does not improve the mean response time but it still improves the 99^{th} percentile.

In addition, Figure 3 also shows that the reduction in response times for HE_2 services is much larger than for ER_2 services. This is further highlighted in Figure 5, where we depict the relative improvement (reduction) in the response times 99^{th} percentile when introducing replication. While under HE_2 services the gains can reach up to 90% under very low loads, under ER_2 services these are limited to at most 50%. This difference arises from the more concentrated service times in the case with lower variability, which offers fewer chances to have at least one replica with a short response time.

Similarly, under grouped-random allocation the threshold load is also higher for HE_2 than for ER_2 services, but the differences are much larger than under the random policy. In fact, if we evaluate the threshold load under service times with different variability, as in Figure 6, we observe that this threshold can be between 20% and 45%, a much wider span than under the random policy (see Figure 4). This figure also highlights that for lowly-variable service times (SCV 1 or below), a decision to replicate based on a potential gain of the mean response time can actually worsen the 99^{th} percentile. Further, the improvements under highly variable (HE_2) service times are also more significant than under ER_2 services, as shown in Figure 7. The latter case provides gains of at most 40%, while the gains under HE_2 services can be as large as 90% and sustain to around 20% for loads as high as 0.3.

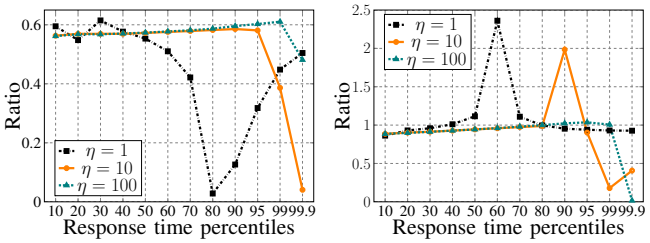
B. The effect of higher moments

We now look further into the impact of the service time variability on the effect of replication across the whole response time distribution. To this end we consider three values for the SCV (C_X^2=1, 5 and 10) and fix the first moment (mean) of the service time distribution M_1 to match the load. The second and third moments of the distribution are obtained as M_2=$(C_X^2+1)M_1^2$ and M_3=$3M_1^3(1+C_X^2)^2/2$. We rely on the method in [18] to match these first three moments with a PH distribution. For each percentile p in the range $\{10, 20, \ldots, 90, 95, 99, 99.9\}$ we obtain the ratio $W_2(p)/W_1(p)$, comparing the response times obtained with replication (R=2) against those without (R=1), with random allocation. Figure 8(a) depicts these ratios for a 0.1 load. The exponential case (C_X^2=1) shows a very smooth behavior, with ratios of about 60% across the whole percentile range. The more variable service times, however, show a strikingly different behavior, with ratios that vary wildly among the percentiles considered. For instance, for the 80^{th} percentile the ratio can be under 0.05 for the C_X^2=10 case, i.e., this percentile with replication is 20 times lower than without replication. The opposite case can be observed in Figure 8(b), where the load is 0.3. Here the exponential case is again very smooth, with a ratio of around 1 across the percentiles considered. However, with a larger service time variability we observe large peaks in the central percentiles. In this case replication can be beneficial



(a) Baseline load: 0.1 (b) Baseline load: 0.3

Figure 8: The impact of Variability



(a) Baseline load: 0.1 (b) Baseline load: 0.3

Figure 9: The impact of Skewness

for the response time tail, but it also increases the response times experienced by a significant proportion of requests.

Beyond the variability, we also consider the effect of the skewness, or third moment, which measures the asymmetry of the service time distribution. To this end we fix the first moment to match a given load, and the SCV to C_X^2=10, and define a baseline skewness coefficient as $\bar{\gamma}$=$3C_X^2/2+1/(2C_X^3)$, which is the minimum skewness coefficient representable by a PH distribution of order 2. We then let the skewness coefficient be γ=$\bar{\gamma}\eta$, and consider values of η in the set $\{1, 10, 100\}$. Figure 9 shows the ratio $W_2(p)/W_1(p)$ for percentiles across the response time distribution for the values of η given above. Of the three cases considered, the one closest to an exponential is η=100, as most of the mass of the initial probability vector is concentrated in one of the phases. In this case the percentile ratio is very stable across the distribution. However, as the skewness decreases we observe very significant differences in the percentile ratio. In both load scenarios a smaller skewness implies that the main effect of replication concentrates on lower percentiles. For instance, with load 0.1 and η=10 or 100, the main effect is on the tail, while for η=1 the main effect is around percentile 80. Thus, in addition to the mean service time, higher moments need to be considered when implementing replication as they have a very significant effect on the offered response times.

C. Comparing allocation policies

We also compare the performance of the two policies considered, random and grouped-random. The results, repeated over a wide range of scenarios, show that the grouped-random policy performs worse than the random policy. This is caused by the greater multiplexing gains offered by the random policy, as it allocates the replicas of new requests to a completely random subset of servers, while in the grouped-random case the random selection is done at the level of server groups, each of which receives all the replicas for a given request.

Table IV: Fitting error

Dis	Err%
Exp	5.004
HE_2	2.701
HE_3	0.812
HE_4	0.080
HE_5	0.049
HE_6	0.048

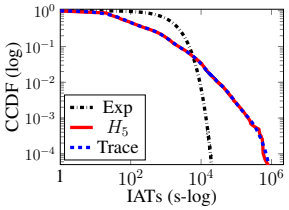
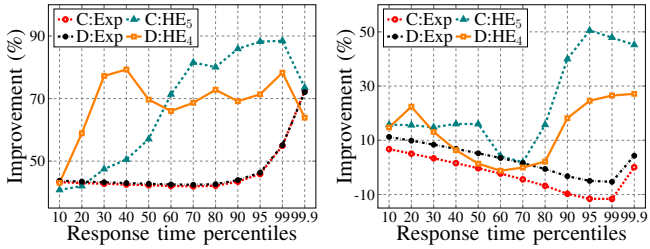


Figure 10: Log C services



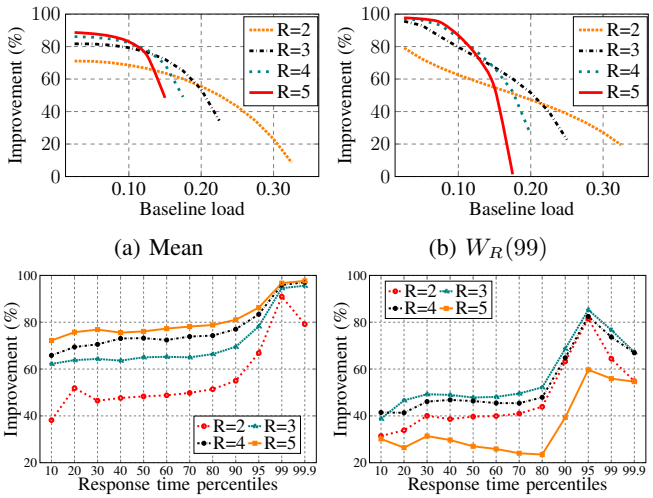
(a) Baseline load: 0.1 (b) Baseline load: 0.3

Figure 11: Percentile improvement in the case study

D. Case study

We now consider realistic traffic patterns by making use of the Intel-NetBatch C and D logs [12]. We focus on successful single-task requests that amount to over 99% of all successful requests. We observe heavy-tailed service times in both logs, with SCV of 76.07 and 12.20, respectively. Given the high variability, we fit the service times using hyper-exponential distributions with r phases (HE_r), relying on the maximum-likelihood method in [10], as implemented in jPhase [19]. Table IV shows the mean relative error between the empirical CDF and the fitted CDFs for log C, which clearly decreases as the number of phases increases. For instance, the error is 5.00% for Exp (i.e., HE_1), and 0.05% for HE_5. As the improvement is limited after five phases, we choose the HE_5 representation for log C. Figure 10 shows the empirical CCDF of the service times, and the fitted exponential and HE distributions. Clearly, the HE distribution captures the behavior of the real traces much better than the exponential distribution, both the body and tail. Similarly, we fit the service times of log D using an HE_4 distribution. For the arrival process, we observe that the SCV is 1.92 and the decay rate of the auto-correlation function (ACF) is 0.97 for log C, which we capture with a MAP using the method in [17]. For log D, the SCV is 8.43 and the ACF is just 0.09, thus we use an HE distribution to match the high variability and the low auto-correlation observed.

The fitted distributions allow us to explore the effect of replication on latency under realistic traffic patterns. As the number of servers is not disclosed, we set it to 50, but other values provide similar results. Figure 11(a) depicts the relative improvement on response time percentiles when introducing replication under a baseline load of 0.1, with grouped-random allocation. The first immediate observation is that the improvement is very different among the percentiles, and it also differs among traces. Log C shows a larger improvement in the tail, while log D shows a stronger improvement for small percentiles. Figure 11(a) also shows how the improvement would be evaluated if the empirical service times were modeled with a standard exponential distribution. Surprisingly, both traces show the same improvement across all percentiles, even



(a) Mean (b) $W_R(99)$

(c) Baseline load: 0.025 (d) Baseline load: 0.15

Figure 12: Improvement with $R>2$ replicas

though the arrival processes are very different. More importantly, the exponential assumption hides the major differences in the lower percentiles and underestimates the benefits of replication. In Figure 11(b) we increase the baseline load to 0.3 and observe that replication significantly improves the tail for both traces, although this improvement is more significant for log C. Also, the improvement of log C dominates that of log D for most of the distribution, while under a 0.1 load this only occurs for percentiles 60 and above. These trends are quite different from the exponential assumption, which leads to the conclusion that replication would be harmful for the tail. Explicitly considering the service time variability shows that it is actually the tail that can benefit the most from replication.

E. Beyond two replicas

We now consider the case with more than one additional replica. The introduction of replicas increases the load by a factor R compared to the baseline load, thus limiting the introduction of more than one replica to low-load cases. Focusing on random allocation as an example, Figures 12(a) and 12(b) show that the addition of more than one replica reduces the response time mean and 99$^{\text{th}}$ percentile, although under more limited baseline loads. In fact, the improvement in the 99$^{\text{th}}$ percentile with 5 replicas can be above 95% for a load under 0.1, but this improvement decreases very fast and becomes worse than the two-replica case for loads above 0.15. This is further illustrated in Figure 12(c), which shows the improvement across many percentiles under a very low load. Here each additional replica improves every percentile, with the main gains obtained in the tail. However, under a 0.15 load, Figure 12(d) shows that the improvement obtained with the third and fourth replica is very limited across all percentiles, compared to the two-replica scenario, while the addition of a fifth replica is actually harmful. Thus, except for scenarios with very low loads, the main benefits of replication can be obtained with a single additional replica. Additional replicas can provide further gains but only under very low loads.

VI. Conclusion

The approximate models introduced in this paper have enabled us to study the effectiveness of request replication to reduce the latency in systems with distributed servers. Results show that service time variability plays a key role when evaluating the effect of replication on latency. Highly variable service times provide larger gains, and offer more chances to benefit from replication. This also showcases the limitation of the exponential assumption for service times as it hides important effects of replication. Further, we observe that the improvement on latency is not uniform across different percentiles, and the load threshold under which replication helps to reduce latency largely depends on the metric targeted. Explicitly considering the response time distribution, via the targeted percentiles, is therefore fundamental to evaluate a replication strategy.

Acknowledgment

The research of Juan F. Pérez is supported by the ARC Centre of Excellence for Mathematical and Statistical Frontiers (ACEMS).

References

[1] J. Dean and L. A. Barroso, "The tail at scale," *CACM*, vol. 56, pp. 74–80, 2013.

[2] G. Linden, "Marissa Mayer at web 2.0," http://glinden.blogspot.com/2006/11/marissa-mayer-atweb-20.html, 2006.

[3] B. Snyder, "Server virtualization has stalled, despite the hype," http://www.infoworld.com/print/146901, 2010.

[4] A. Vulimiri, P. Godfrey, R. Mittal, J. Sherry, S. Ratnasamy, and S. Shenker, "Low latency via redundancy," in *CoNEXT*, 2013.

[5] Z. Qiu and J. F. Pérez, "Enhancing reliability and response times via replication in computing clusters," in *IEEE INFOCOM*, 2015.

[6] N. B. Shah, K. Lee, and K. Ramchandran, "When do redundant requests reduce latency?" in *Allerton*, 2013.

[7] Z. Qiu and J. F. Pérez, "Evaluating the effectiveness of replication for tail-tolerance," in *IEEE/ACM CCGRID*, 2015.

[8] G. Ananthanarayanan, A. Ghodsi, S. Shenker, and I. Stoica, "Why let resources idle? aggressive cloning of jobs with Dolly," *Memory*, vol. 40, p. 80, 2012.

[9] Z. Qiu and J. F. Pérez, "Assessing the impact of concurrent replication with canceling in parallel jobs," in *IEEE MASCOTS*, 2014.

[10] R. El Abdouni Khayari, R. Sadre, and B. R. Haverkort, "Fitting world-wide web request traces with the EM-algorithm," *Perform. Eval.*, vol. 52, pp. 175–191, 2003.

[11] G. Latouche and V. Ramaswami, *Introduction to matrix analytic methods in stochastic modeling*. SIAM, 1999.

[12] "Parallel Workloads Archive," http://www.cs.huji.ac.il/labs/parallel/workload/.

[13] S. Asmussen and J. R. Møller, "Calculation of the steady state waiting time distribution in GI/PH/c and MAP/PH/c queues," *Queueing Syst.*, vol. 37, pp. 9–29, 2001.

[14] Z. Qiu, J. F. Pérez, and P. G. Harrison, "Beyond the mean in fork-join queues: Efficient approximation for response-time tails," *Perform. Eval.*, vol. 91, pp. 99–116, 2015.

[15] B. Sengupta, "Markov processes whose steady state distribution is matrix-exponential with an application to the GI/PH/1 queue," *AAP*, vol. 21, pp. 159–180, 1989.

[16] W. Whitt, "Approximating a point process by a renewal process, I: Two basic methods," *Oper. Res.*, vol. 30, pp. 125–147, 1982.

[17] A. Heindl, "Inverse characterization of hyperexponential MAP(2)s," in *ASMTA*, 2004.

[18] A. Bobbio, A. Horváth, and M. Telek, "Matching three moments with minimal acyclic phase type distributions," *Stoch. Model.*, vol. 21, pp. 303–326, 2005.

[19] J. F. Pérez and G. Riano, "jPhase: an object-oriented tool for modeling phase-type distributions," in *ACM SMCtools*, 2006.