Evaluating the Effectiveness of Replication for Tail-tolerance

Zhan Qiu, Juan F. Pérez
Department of Computing
Imperial College London
London, UK
{zhan.qiu11, j.perez-bernal}@imperial.ac.uk

***Abstract*—Computing clusters (CC) are a cost-effective high-performance platform for computation-intensive scientific and engineering applications. A key challenge in managing CCs is to consistently achieve low response times. In particular, tail-tolerant methods aim to keep the *tail* of the response-time distribution short. In this paper we explore *concurrent replication with canceling*, a tail-tolerant approach that involves processing requests and their replicas concurrently, retrieving the result from the first replica that completes, and canceling all other replicas. We propose a stochastic model that considers any number of replicas, general processing and inter-arrival times, and computes the response time distribution. We show that replication can be very effective in keeping the response-time tail short, but these benefits highly depend on the processing-time distribution, as well as on the CC utilization and the statistical characteristics of the arrival process. We also exploit the model to support the selection of the optimal number of replicas, and a resource provisioning strategy that meets service-level objectives on the response-time percentiles.**

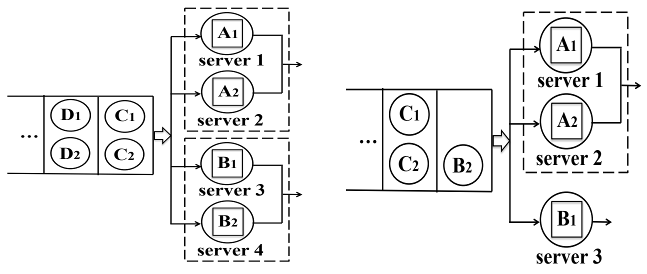
I. Introduction

Computing clusters (CC) have become a cost-effective high-performance platform for computation-intensive scientific and engineering applications [1], [2]. A CC consists of a set of processing elements interconnected by a high-speed communication network, providing computing services to multiple users. A key challenge in managing CCs is to consistently achieve low response times, or latency, especially to keep the tail of the latency distribution short. For instance, experiments of Google show that a system where each request typically responds in 1ms, has a 99-th percentile latency of 10ms [3]. High latency can be caused by a number of reasons, including contention for shared resources, queueing, or hardware problems [3]. Keeping latency low is critical, since even a small amount of delay may significantly degrade the system perceived quality of service. For instance, the traffic and revenue of Google searches has been observed to decrease by 20% when introducing a 500ms delay [4].

To address this problem we consider *concurrent replication with canceling*: initiate multiple replicas of a request, use the result from the replica that responds first, and cancel all the other replicas. As the overall latency becomes the minimum of the delays across all the replicas, it can potentially reduce both the mean and the tail of the latency distribution, especially since the source of latency is often due to external interference, rather than inherent to the requests [3]. Further, concurrent replication can also handle unpredictable delays due to exceptional conditions, unless they occur to all replicas simultaneously [5]. Heavy concurrent replication is appealing given the common observation that most clusters are highly underutilized, with the average utilization of major data center servers being around 18% [6]. However, concurrent replication may introduce unacceptable additional load, leading to excessive resource usage, degrading the offered response times (RTs). This additional load can be limited by the canceling mechanism [3], which requires the servers to share updates on the status of their replicas. In this paper, we evaluate the effectiveness of *concurrent replication with canceling* as a tail-tolerant mechanism, that is, it ability to reduce latency, especially the *tail* of the latency distribution.

A. Contributions

The main contribution of this paper is the introduction of an analytical model to compute the RT distribution offered by a CC implementing concurrent replication with canceling. The key aspects of the model are: i) it considers any number of replicas, enabling the evaluation of different replication levels; ii) it accepts fairly general request processing times, which is necessary to capture the variability observed in CCs [7]; iii) it considers general inter-arrival times (IATs), capturing both high variability and auto-correlation; iv) it computes the RT distribution, which is necessary to analyze the latency tail, as well as to evaluate service-level objectives (SLOs) defined as RT percentiles. The model, introduced in Sections IV and V, goes beyond existing proposals [5], [8], [9], which focus on a single additional replica, and assume exponential processing times that rarely fit real data. We exploit the model to show that replication can be very effective in keeping the latency tail short, but these benefits highly depend on the processing-time distribution, as well as on the CC utilization and the statistical characteristics of the arrival process. In Sections VII and VI, we use the proposed model to determine optimal replication levels, and for an SLO-driven resource provisioning strategy. The results highlight that ignoring either the variability and correlation in the arrival process, or the variability in the processing times, may lead to large SLO violations, or wasteful over-provisioning.



(a) Basic case (b) General case

Fig. 1: The basic and the general cases

II. Related work

Concurrent replication has been considered in [5], [6], [10] to reduce latency for interactive and latency-sensitive applications. Trace-driven simulations in [10] show that processing replicas simultaneously is effective in keeping the tail of the latency distribution short. [5] derives the utilization level below which replication helps to improve the mean RT, for a CC with Poisson arrivals and exponential service times, and provide approximate upper bounds for more general services. The canceling mechanism has been considered in [3], [8], [9], [11] to limit the additional load introduced by replicas. [3] observes that adopting replication with canceling in a benchmark experiment reduces the latency substantially, while introducing only a slight additional load. Simulation results for the average latency are presented in [11] for a CC with exponential, light-everywhere, and heavy-everywhere service-time distributions. [8], [9] propose stochastic models to derive the mean RT obtained with one additional replica, for the case of two servers with independent queues in [8], and for the case of synchronous parallel processors in [9].

III. Background

A. Reference model

We consider a system consisting of a central dispatcher and c distributed, homogeneous, and independent servers. Requests arrive at the dispatcher and join the next server that becomes available with first-come first-served (FCFS) scheduling. For each arriving request, $r \geq 1$ replicas are initiated and submitted to the system, where $r = 1$ represents the case without replication. The set of replicas for a single request is referred to as a *job*, and the number of replicas r is also called the job size. To reduce unnecessary workload, servers communicate updates on the status of their replicas to each other, such that, when a replica completes execution, it immediately cancels all the other replicas in the same job. The removal of all the replicas in a job may require a non-negligible time, which we refer to as the canceling overhead. In the model introduced in the next sections we assume zero canceling overhead, but in Section VI we modify the model to explicitly consider overheads of different magnitudes, compared to the job processing times.

For the sake of clarity, we will introduce the model in two steps. First, we consider the *basic case*, where we assume that the number of servers c is a multiple of the job size r, i.e., $c = nr$ for $n \in \mathbb{N}$. In this case the analysis simplifies because all the requests in a job start and complete service at the same time. Figure 1(a) illustrates this for the case $c = 4$ and $r = 2$. For instance, the completion of a replica in either job A or B triggers the completion of the corresponding job, and all the replicas in job C start service at the same time. Thereafter, we consider the *general case*, where no restrictions are posed on the number of servers. In this case, a job may start service in either of two states: with all its replicas at the same time (denoted as state A, which is short for *all*); or with only part of its replicas, while the remaining ones wait in the queue (denoted as state P, which is short for *part*). An example of this is depicted in Figure 1(b), for the case $c = 3$ and $r = 2$. Here jobs A and B are in service, but job B has one replica waiting in the queue. This replica will only start service once job A completes, and it may not start at all if the replica of job B already in service finishes before job A. The more complex dynamics present in this case will be considered in Section V.

TABLE I: Notation

Parameter	Definition
c	Number of servers
r	Job size: number of replicas
c_A	Number of super-servers, $c_A = c/r$
MAP(D_0, D_1)	Arrival process with m_a phases
λ	Mean arrival rate
PH$(\boldsymbol{\alpha}_R, S_R)$	Replica processing time PH dist.
PH$(\boldsymbol{\alpha}_i, S_i)$	Job service-time PH dist. with d_i service phases $i = J$ for the basic case, $i = F$ for the general case
PH$(\boldsymbol{\alpha}_w, S_w)$	Job waiting-time PH dist.
PH$(\boldsymbol{\alpha}_{\text{res}}, S_{\text{res}})$	Job response-time PH dist.
Q_i, Π_i	Transition rates with or without a service completion, $i = A$ for the basic case, $i = P$ for the general case
γ	Probability that a job has to wait upon arrival
$\boldsymbol{e}_s$	A zero vector with a 1 in its s-th element
N_i	Set of service phases (cardinality d_i). $i = S$ for all jobs in service, $i = Y$ for the youngest job, $i = N$, initial phases of replicas remaining in the queue
I_n	Identity matrix of size n
SCV	Squared coefficient of variation

In the next sections we propose a model to evaluate the performance of a CC adopting replication with canceling. Table I summarizes the notation used in the model. The model proposed assumes that the replicas of a request are independent. Although the replicas may be correlated in general, recent experiments [5] show that the independent approximation is reasonable when the number of servers is sufficiently large. For instance, the mean RT obtained under the independent assumption is within 0.1% of the value observed in the experiments with 20 servers.

B. Phase-type distributions

To be able to analyze a broad range of behaviors for the replica processing times, we will rely on Phase-type (PH) distributions. In fact, PH distributions are dense in the set of all positive-valued distributions, thus being able to approximate general distribution [12]. PH distributions are also popular for maintaining some of the analytical tractability of the exponential distribution. 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 [12]. 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 PH$(\boldsymbol{\tau},S)$. The vector $\boldsymbol{S}^*=-S\mathbf{1}$ holds the absorption rates from the transient states, where $\mathbf{1}$ denotes 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 $E[X]=-\boldsymbol{\tau}S^{-1}\mathbf{1}$.

C. Markovian Arrival Processes

Motivated by the high variability in the inter-arrival times (IATs) observed in CCs [7], we model the request arrival process as a Markovian Arrival Process (MAP). The continuous-time MAP [12] is a marked MC with m_a states, or arrival phases, and 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 MAP(m_a,D_0,D_1). The mean arrival rate is

$$\lambda=\boldsymbol{d}D_1\mathbf{1}, \tag{1}$$

where $\boldsymbol{d}$ is the stationary distribution of the underlying MC, i.e. $\boldsymbol{d}D=0$ and $\boldsymbol{d}\mathbf{1}=1$. A renewal process with PH$(\boldsymbol{\tau},S)$ IATs can be modeled as a MAP, by setting $D_0=S$ and $D_1=\boldsymbol{S}^*\boldsymbol{\tau}$. Compared to traditional Poisson arrivals, a MAP is able to represent more general IATs, including features such as high variability and auto-correlation.

IV. The basic case

In this section we focus on the basic case, where the number of servers c is a multiple of the number of replicas r, as in Figure 1(a). We obtain the RT distribution for this case by relying on [13], where the PH representation of the waiting-time distribution is obtained for the MAP/PH/c queue. The case without replication, i.e., when $r=1$, is in fact a MAP/PH/c queue, so we can readily use [13] to obtain the waiting-time distribution, and combine it with the service-time distribution to obtain the RT distribution, as in Section IV-C. The case with replication requires a more detailed analysis to capture the dynamics introduced by the replication and canceling. In the following we refer to a period during which all the servers are busy as an *all-busy* period, and to a period where at least one server is idle as a *not-all-busy* period. We assume that the distribution of the *replica* processing times is PH$(\boldsymbol{\alpha}_R,S_R)$, and a key step in the following analysis is showing that the *job* processing times also follow a PH distribution, with parameters $(\boldsymbol{\alpha}_J,S_J)$ that depend on the overall system state.

A. The service-time distribution

As mentioned before, in the basic case all the replicas in a job start service at the same time, and terminate when either of the replicas completes service. As a result we can aggregate the servers into $c_A=c/r$ "super-servers", letting each job be processed by a single super-server. We thus need to describe the job service-time as a PH distribution, which is possible since the job service time is in this case the minimum of the replicas service time, and PH distributions are closed under the minimum operation [12]. Since the replicas are homogeneous, instead of recording the service phase of each replica, it suffices to count the number of replicas in each service phase. The service time of a job then follows a PH distribution with representation $(\boldsymbol{\alpha}_J,S_J)$, with service phases $N_J=\{\boldsymbol{x}_J=(x_J(1),\ldots,x_J(m))|x_J(i)\in\{0,\ldots,r\},\sum_{i=1}^m x_J(i)=r\}$, where $x_J(i)$ is the number of replicas in the i-th service phase, for $1\leq i\leq m$. The number of job service phases is thus

$$d_J=\binom{m+r-1}{r}.$$

The initial distribution of the job service phase $\boldsymbol{\alpha}_J$ follows a multinomial distribution, as each replica picks its initial phase independently. The probability that a job starts service in phase $\boldsymbol{x}_J$ is thus

$$\alpha_J(\boldsymbol{x}_J)=r!\prod_{i=1}^{m}\frac{\alpha_R(i)^{x_J(i)}}{x_J(i)!},\quad \boldsymbol{x}_J\in N_J. \tag{2}$$

Let $\boldsymbol{e}_s$ be a zero vector with a 1 in its s-th element, such that a job goes from service phase $\boldsymbol{x}_J$ to phase $\boldsymbol{x}_J+\boldsymbol{e}_i-\boldsymbol{e}_j$ when a replica jumps from the j-th phase to the i-th. The elements of the matrix S_J, are thus given by $S_J(\boldsymbol{x}_J,\boldsymbol{x}_J+\boldsymbol{e}_i-\boldsymbol{e}_j)=x_J(j)S_R(j,i)$, as in this case one of the $x_J(j)$ replicas in the j-th service phase jumps to the i-th with rate $S_R(j,i)$.

As an example, consider the case where the replica service-time distribution has $m=2$ phases and sub-generator matrix

$$[S_R|\boldsymbol{S}_R^*]=\begin{array}{c|cc|c} & 1 & 2 & C\\ \hline 1 & -2 & 2 & 0\\ 2 & 0 & -2 & 2\end{array}, \tag{3}$$

where C is the absorbing phase, and the vector $\boldsymbol{S}_R^*=-S_R\mathbf{1}$ holds the absorbing rates into phase C. Assuming $r=2$ replicas, the job service phases are $N_J=\{(2,0),(1,1),(0,2)\}$, and its sub-generator matrix S_J is

$$[S_J|\boldsymbol{S}_J^*]=\begin{array}{c|ccc|c} & (2,0) & (1,1) & (0,2) & C\\ \hline (2,0) & -4 & 4 & 0 & 0\\ (1,1) & 0 & -4 & 2 & 2\\ (0,2) & 0 & 0 & -4 & 4\end{array},$$

where $\boldsymbol{S}_J^*=-S_J\mathbf{1}$. For instance, in job phase $(2,0)$ either of the 2 replicas in the first phase could jump to the second phase with rate $S_R(1,2)=2$, thus $S_J((2,0),(1,1))=4$.

B. The waiting-time distribution

The waiting time of a job is the time period between its arrival and the time its service starts. By aggregating the c servers into $c_A=c/r$ super-servers and looking at the job level, we can compute the waiting-time distribution for the basic case using the approach in [13], as we have a MAP/PH/c_A queue with PH$(\boldsymbol{\alpha}_J,S_J)$ service times. Here we detail some of the steps in [13] as we will need to treat them in more detail for the general case.

The waiting-time distribution is obtained by observing the queue *only during the all-busy periods*, and defining a bivariate Markov process $\{X(t), J(t)|t \geq 0\}$. The *age process* $X(t)$ is the total time-in-system of the *youngest* job in service, and the *phase process* $J(t) = (M(t), S(t))$, holds the phases of both the MAP arrival process $M(t)$ and the service process $S(t)$. The age process $\{X(t)|t \geq 0\}$ takes values in $[0, \infty)$, increasing linearly with rate 1 if no *job* service completions occur. A downward jump occurs when a service completion triggers the start of a new *job* service, and the value of the age process after the jump is equal to the waiting time of the job starting service.

As we aggregate the servers in c_A super-servers, the variable $S(t)$ records the phases in which the c_A jobs in service are. The set of possible service phases of all the jobs in service is $N_S = \{\boldsymbol{x}_S \in \mathbb{N}^{1\times d_J} | x_S(\boldsymbol{i}) \in \{0, \dots, c_A\}, \sum_{\boldsymbol{i}\in N_J} x_S(\boldsymbol{i}) = c_A\}$, where $\mathbb{N}^{1\times d_J}$ is a $1 \times d_J$ vector of natural numbers, and $x_S(\boldsymbol{i})$ is the number of jobs in service phase $\boldsymbol{i}$, for $\boldsymbol{i} \in N_J$. The bold index $\boldsymbol{i}$ here represent the *job* service phase $\boldsymbol{i}$, which is a vector, and is different from the scalar indexes used before for the *replica* service phase. The cardinality of N_S is

$$d_S = \binom{d_J + c_A - 1}{c_A}.$$

The dimension of the full phase process $J(t) = (M(t), S(t))$ is thus $m_a d_S$.

Let the $1 \times m_a d_S$ vector $\boldsymbol{\pi}(x)$ hold the steady-state density of $\{X(t), J(t)\}$. The entry $(i, \boldsymbol{j})$ of $\boldsymbol{\pi}(x)$ is the probability measure at age $x \geq 0$, phase $i \in \{1, \dots, m_a\}$ and $\boldsymbol{j} \in N_S$. This measure has been shown [13] to have a matrix-exponential form such that

$$\boldsymbol{\pi}(x) = \boldsymbol{\pi}(0) exp(Tx).$$

Further, the matrix T can be found by solving the non-linear integral equation [13], [14]

$$T = Q_A^{(MAP)} + \int_0^\infty \exp\{Tt\}(\exp\{D_0 t\} \otimes I_{d_S}) dt (D_1 \otimes \Pi_A), \quad (4)$$

where I_n is the identity matrix of size n, and $\otimes$ denotes the Kronecker product. Here $Q_A^{(MAP)} = I_{m_a} \otimes Q_A$, with Q_A a $d_S \times d_S$ matrix whose off-diagonal elements hold the transition rates between the service phases N_S not associated with a service completion. Also, Π_A is a $d_S \times d_S$ matrix that holds the transition rates between the service phases N_S accompanied by a service completion, during an all-busy period.

The elements of Q_A and Π_A can be obtained by considering the transitions from service phase $\boldsymbol{x}_S$ to phase $\boldsymbol{x}_S + \boldsymbol{e_i} - \boldsymbol{e_j}$, for each phase $\boldsymbol{x}_S \in N_S$, and $\boldsymbol{i}, \boldsymbol{j} \in N_J$. Here the vector $\boldsymbol{e_i}$ is a zero vector of size d_J, indexed by N_J, with a 1 in the entry that corresponds to service phase $\boldsymbol{i}$. Recall that in this case $\boldsymbol{x}_S$ describes the status of *all* the jobs in service. Thus the entry $Q_A(\boldsymbol{x}_S, \boldsymbol{x}_S + \boldsymbol{e_i} - \boldsymbol{e_j})$ is given by $x_S(\boldsymbol{j}) S_J(\boldsymbol{j}, \boldsymbol{i})$, where one of the $x_S(\boldsymbol{j})$ jobs in service phase $\boldsymbol{j}$ jumps to phase $\boldsymbol{i}$ with rate $S_J(\boldsymbol{j}, \boldsymbol{i})$. Also, the element $\Pi_A(\boldsymbol{x}_S, \boldsymbol{x}_S + \boldsymbol{e_i} - \boldsymbol{e_j})$ is given by $x_S(\boldsymbol{j}) S_J^*(\boldsymbol{j}) \alpha_J(\boldsymbol{i})$, where one of the $x_S(\boldsymbol{j})$ jobs in service phase $\boldsymbol{j}$ terminates with rate $S_J^*(\boldsymbol{j})$, and a new job starts service in phase $\boldsymbol{i}$ with probability $\alpha_J(\boldsymbol{i})$.

As in [13], [14], Eq. (4) can be solved iteratively. However, different from the numerical integration used in [13], we follow the approach as described in [15]. Once T has been found, a few more steps are needed, such as finding the vector $\boldsymbol{\pi}(0)$, as described in [13]. Here we recall the computation of the probability γ that a job finds all servers busy and needs to wait. Let η_0 be the number of service completions in an all-busy period, and η_1 the number of arrivals in a not-all-busy period, where their expected values $E[\eta_0]$ and $E[\eta_1]$ can be obtained as in [13, Section 6]. Thus the probability that a job has to wait is $\gamma = (E[\eta_0] - 1)/(E[\eta_0] - 1 + E[\eta_1])$ since $E[\eta_0] - 1$ is the expected number of jobs that have to wait.

We can now determine the waiting-time distribution PH($\boldsymbol{\alpha}_w$, S_w). Letting $\boldsymbol{\varphi} = (T - Q_A^{(MAP)})\mathbf{1}$ and $\boldsymbol{\alpha} = -\boldsymbol{\pi}(0) T^{-1}$, the initial probability vector $\boldsymbol{\alpha}_w$ is given by

$$\alpha_w(i) = \gamma \frac{\alpha(i)\varphi(i)}{\boldsymbol{\alpha}\boldsymbol{\varphi}}, \quad 1 \leq i \leq m_a d_S.$$

And the elements of the matrix S_w are given by

$$S_w(i,j) = \frac{\alpha_w(j) T(j,i)}{\alpha_w(i)}, \quad 1 \leq i, j \leq m_a d_S.$$

C. The response-time distribution

Given the PH representations of waiting and service times, the PH representation $(\boldsymbol{\alpha}_{\text{res}}, S_{\text{res}})$ of the RT distribution is

$$\boldsymbol{\alpha}_{\text{res}} = [\boldsymbol{\alpha}_w \quad (1 - \boldsymbol{\alpha}_w \mathbf{1})\boldsymbol{\alpha}_J],$$

$$S_{\text{res}} = \begin{bmatrix} S_w & (-S_w \mathbf{1})\boldsymbol{\alpha}_J \\ 0_{m_a d_S \times d_J} & S_J \end{bmatrix},$$

where $0_{m\times n}$ denotes an $m \times n$ zero matrix. With this representation we can compute the CDF, percentiles, and moments.

V. The general case

In this section, we obtain the RT distribution for the general case, where the number of servers c is not necessarily a multiple of the number of replicas r. In this case, during the all-busy period there are $c_A = \lfloor c/r \rfloor$ jobs in service with all the replicas in service, i.e., in state A, where $\lfloor a \rfloor$ is the largest integer smaller than or equal to a. In addition, the youngest job in service is always in state P, with $p = (c \bmod r)$ replicas in service and the remaining $r - p$ replicas waiting in front of the queue. Instead, during a not-all-busy period, all the jobs in service are in state A. This therefore affects the service-time and waiting-time distributions, as described next.

A. The service-time distribution

When a job starts service during an all-busy period, it does so in state P as the youngest job in service. We thus introduce $Y(t)$ to keep track of the service phase of the youngest job in service at time t. $Y(t)$ takes values in the set $N_Y = \{\boldsymbol{x}_Y = (x_Y(1), \dots, x_Y(m)) | x_Y(i) \in \{0, \dots, p\}, \sum_{i=1}^m x_Y(i) = p\}$, where $x_Y(i)$ is the number of replicas in the i-th service phase, for $1 \leq i \leq m$. The cardinality of N_Y is

$$d_Y = \binom{m + p - 1}{p}.$$

The youngest job can complete service if any of its p replicas in service completes service. Otherwise, when one of the c_A jobs in state A completes service, the $r - p$ replicas of the youngest job waiting in the queue start service, and the job jumps to state A. Thus, for the youngest job in state P, whether it completes service or jumps to state A, depends not only on the service phase of the job itself, but also on the phase of all the jobs in service in state A. Therefore, to describe the job service-time distribution it is essential to keep track of the service phase of all the jobs in service, while the job is in state P. Once it switches from state P to state A, its service time depends solely on the state of its own replicas.

With the previous considerations, we define PH($\boldsymbol{\alpha}_F, S_F$), the job service-time distribution. As a job may be in either state A or P, we partition the sub-generator S_F as

$$S_F = \begin{bmatrix} S_{A,A} & 0_{d_J \times d_F} \\ S_{P,A} & S_{P,P} \end{bmatrix}. \tag{5}$$

The $S_{A,A}$ matrix holds the transition rates without a service completion for a job in state A. The service-time distribution of a job in state A was defined in Section IV, thus $S_{A,A} = S_J$, which is a $d_J \times d_J$ matrix. Similarly, the matrix $S_{P,P}$ holds the transition rates without a service completion when the job is in state P. As stated above, when the job is in state P we need to keep track of $(S(t), Y(t))$, where $Y(t)$ holds the service phase of the job in state P, while $S(t)$ holds the service phase of all the jobs in state A. The set of service phases in this case is thus the product of the two sets, $N_F = N_S \times N_Y$, and its cardinality is $d_F = d_S d_Y$. As as example, consider the case $r = 2$, $m = 2$, and $c = 3$, depicted in Figure 1(b). During an all-busy period, one of the two jobs in service is in state A, while the youngest job is in state P with 1 replica in service. The combined phases of jobs in service are $N_S \times N_Y = \{((1,0,0), (1,0)), ((1,0,0),(0,1)), ((0,1,0),(1,0)),((0,1,0),(0,1)), ((0,0,1),(1,0)), ((0,0,1),(0,1))\}$.

The elements of the $d_F \times d_F$ matrix $S_{P,P}$ cover two cases. In phase $(\boldsymbol{x}_S, \boldsymbol{x}_Y)$, one of the $\boldsymbol{x}_S(\boldsymbol{j})$ jobs in state A, service phase $\boldsymbol{j}$, may jump to service phase $\boldsymbol{i}$ without a service completion with rate $S_J(\boldsymbol{j}, \boldsymbol{i})$, thus $S_{P,P}((\boldsymbol{x}_S, \boldsymbol{x}_Y), (\boldsymbol{x}_S + \boldsymbol{e_i} - \boldsymbol{e_j}, \boldsymbol{x}_Y)) = \boldsymbol{x}_S(\boldsymbol{j}) S_J(\boldsymbol{j}, \boldsymbol{i})$, for $\boldsymbol{i}, \boldsymbol{j} \in N_J$. Also, in phase $(\boldsymbol{x}_S, \boldsymbol{x}_Y)$, one of the $\boldsymbol{x}_Y(j)$ replicas of the youngest job in the j-th service phase may jump to the i-th without completing service with rate $S_R(j,i)$, making $S_{P,P}((\boldsymbol{x}_S, \boldsymbol{x}_Y), (\boldsymbol{x}_S, \boldsymbol{x}_Y + \boldsymbol{e}_i - \boldsymbol{e}_j)) = \boldsymbol{x}_Y(j) S_R(j,i)$, for $1 \le i,j \le m$.

Finally, the matrix $S_{P,A}$ holds the transition rates with which the youngest job in state P enters state A. When the job in state P jumps to state A, its initial service phase in state A is the combination of the current phase of its p replicas in service, and the starting phase of its remaining replicas in the queue. Let $\boldsymbol{x}_N$ be a $1 \times m$ vector where the entry $x_N(i)$ is the number of replicas starting service in phase i, out of the $r-p$ replicas waiting in the queue. All possible vectors of this type are in the set $N_N = \{\boldsymbol{x}_N = (x_N(1), \dots, x_N(m)) | x_N(i) \in \{0, \dots, r-p\}, \sum_{i=1}^{m} x_N(i) = r-p\}$, and the probability $\alpha_N(\boldsymbol{x}_N)$ of the $r-p$ replicas starting in phases $\boldsymbol{x}_N$ can be computed with Eq. (2), replacing r by $r-p$, and $\boldsymbol{x}_J$ by $\boldsymbol{x}_N$. Then, in phase $(\boldsymbol{x}_S, \boldsymbol{x}_Y)$, where the youngest job is in phase $\boldsymbol{x}_Y \in N_Y$, if its waiting replicas start service in phase $\boldsymbol{x}_N \in N_N$, the job jumps to service phase $(\boldsymbol{x}_Y + \boldsymbol{x}_N) \in N_J$. Thus the elements of the $d_F \times d_J$ matrix $S_{P,A}$ are given by $S_{P,A}((\boldsymbol{x}_S, \boldsymbol{x}_Y), \boldsymbol{e_j}) = x_S(\boldsymbol{i}) S_J^*(\boldsymbol{i}) \alpha_N(\boldsymbol{x}_N)$, as in this case one of the $x_S(\boldsymbol{i})$ jobs in state A, phase $\boldsymbol{i} \in N_J$ completes service, and the job in state P jumps to state A, phase $\boldsymbol{j} = (\boldsymbol{x}_Y + \boldsymbol{x}_N)$.

To determine the initial probability vector $\boldsymbol{\alpha}_F$ of the job service process, we consider jobs starting service in state A and P separately. During the all-busy period, every job starts service in state P, while in the not-all-busy period, all jobs start service in state A except for the arrival that initiates an all-busy period. As described in Section IV-C, $E[\eta_0]$ is the expected number of job service completions during an all busy period, so it is equal to the expected number of arrivals during such a period, all of which start service in phase P. Also, $E[\eta_1]$ is the expected number of arrivals during a not-all-busy, all of which, but one, start service in phase A. As a result, the probability p_A that a job starts service in state A is given by $(E[\eta_1]-1)/(E[\eta_1]-1+E[\eta_0])$. The initial probability vector $\boldsymbol{\alpha}_F$ is thus

$$\boldsymbol{\alpha}_F = [\boldsymbol{\alpha}_A \quad \boldsymbol{\alpha}_P] = [p_A \boldsymbol{\alpha}_J \quad (1-p_A)\boldsymbol{\alpha}_Y]. \tag{6}$$

B. The waiting-time distribution

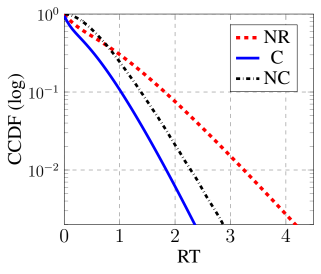
As in Section IV, to find the PH($\boldsymbol{\alpha}_w, S_w$) waiting-time distribution we need to find the matrix T, solving Eq. (4). Apart from changing the identity matrix I_{d_S} to I_{d_F}, we must specify the $d_F \times d_F$ matrices Q_P and Π_P, which replace Q_A and Π_A, respectively, in the general case. As the matrix Q_P holds the transition rates among the service phases N_F without triggering a service completion during an all-busy period, this is simply equal to $S_{P,P}$. The matrix Π_P, instead, holds the transition rates among the phases N_F associated with a service completion during the all-busy period, which can derive from two cases. In the case the job in state P completes service, all its replicas, including the ones waiting in the queue, are canceled immediately, and the next job in the queue starts service in state P. Thus the element $\Pi_P((\boldsymbol{x}_S, \boldsymbol{x}_Y), (\boldsymbol{x}_S, \boldsymbol{x}_K))$ is given by $x_Y(i) S_R^*(i) \alpha_Y(\boldsymbol{x}_K)$, for $\boldsymbol{x}_S \in N_S$ and $\boldsymbol{x}_Y, \boldsymbol{x}_K \in N_Y$, as any of the $x_Y(i)$ replicas of the youngest job in the i-th service phase completes service with rate $S_R^*(i)$, and a new job starts service in state P and phase $\boldsymbol{x}_K$ with probability $\alpha_Y(\boldsymbol{x}_K)$. Similarly, when a job in state A completes service, the associated elements of the matrix Π_P are given by $\Pi_P((\boldsymbol{x}_S, \boldsymbol{x}_Y), (\boldsymbol{x}_S - \boldsymbol{e_i} + \boldsymbol{e_j}, \boldsymbol{x}_K)) = x_S(\boldsymbol{i}) S_J^*(\boldsymbol{i}) \alpha_N(\boldsymbol{x}_N) \alpha_Y(\boldsymbol{x}_K)$, as any of the $x_S(\boldsymbol{i})$ jobs in state A, phase $\boldsymbol{i} \in N_J$ completes service, and the job that was in state P, jumps to state A, phase $\boldsymbol{j} = (\boldsymbol{x}_Y + \boldsymbol{x}_N)$. In addition, a new job starts service in phase $\boldsymbol{x}_K \in N_Y$. We can now follow the same steps as in Section IV to find the waiting-time distribution PH($\boldsymbol{\alpha}_w, S_w$).

C. The response-time distribution

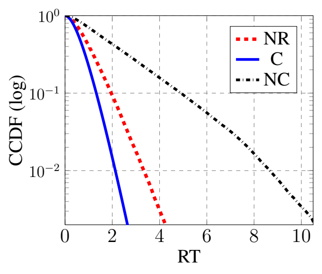
As in Section IV, the PH representation of the RT distribution under the general case is given by

$$\boldsymbol{\alpha}_{\text{res}} = [\boldsymbol{\alpha}_w \quad (1 - \boldsymbol{\alpha}_w \mathbf{1})\boldsymbol{\alpha}_F],$$

$$S_{\text{res}} = \begin{bmatrix} S_w & (-S_w \mathbf{1})\boldsymbol{\alpha}_F \\ 0_{m_a d_S \times d_F} & S_F \end{bmatrix}.$$



(a) NR-utilization: 0.25



(b) NR-utilization: 0.4

Fig. 2: The effect of cancelling

D. *Utilization*

The utilization U is the expected fraction of the time that a server is busy, which can be computed as

$$U = (E[r]\lambda)/(c\mu), \tag{7}$$

where $E[r]$ is the expected number of servers used by a job, the mean arrival rate λ is defined in Eq. (1), and the job mean service rate μ is given by $\mu = \boldsymbol{\alpha}_J(-S_J)^{-1}\mathbf{1}$. In the basic case, all jobs start and complete service in state A, thus $E[r] = r$. Instead, in the general case, jobs may finish service using either r or p replicas depending on whether they complete service in state A or P. This can be determined by relying on the service-time distribution PH($\boldsymbol{\alpha}_F, S_F$), as defined in (5) and (6). Let $\boldsymbol{S}^*_{P,C}$ be the vector holding the absorption, or service completion, rates for a job in state P, which is given by $\boldsymbol{S}^*_{P,C} = -(S_{P,A} + S_{P,P})\mathbf{1}$. Thus the probability $p_{P,C}$ that a job completes service in state P, is given by

$$p_{P,C} = \boldsymbol{\alpha}_P(-S_{P,P})^{-1}\boldsymbol{S}^*_{P,C},$$

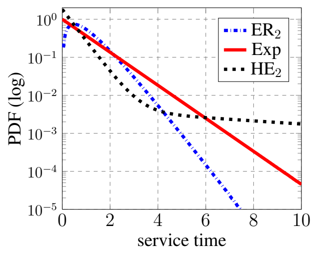
which captures the path of a job that starts service in state P with initial probability vector $\boldsymbol{\alpha}_P$, and then spends some time in state P before eventually completing service, being absorbed in phase C with probability vector $(-S_{P,P})^{-1}\boldsymbol{S}^*_{P,C}$. Thus the expected number of servers a job uses is given by

$$E[r] = r - p_{P,C}(r - p),$$

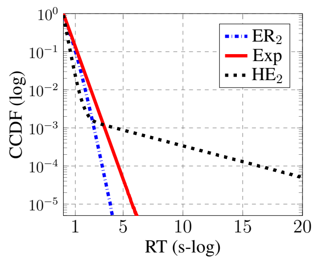
since every job uses r servers, except those that, with probability $p_{P,C}$, use p servers only.

VI. Experimental Assessment

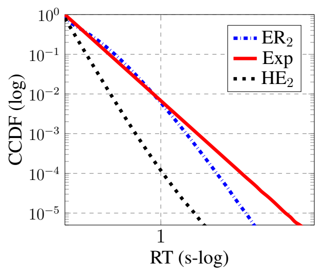
We now make use of the proposed model to evaluate the effect of different parameters on the offered RTs. Thanks to the flexibility offered by the model, we consider arrival processes with exponential (Exp), 2-phase Erlang (ER$_2$), and 2-phase hyper-exponential (HE$_2$) inter-arrival times (IATs), as well as 2-phase MAPs (MAP). We thus cover a broad range of behaviors in terms of variability, measured by the squared coefficient of variation (SCV), defined for a random variable X as $C^2_X = Var[X]/E[X]^2$. The SCV for ER$_2$ and Exp is 0.5 and 1, respectively, while the HE$_2$ covers the cases with SCV greater than 1. We use the methods in [16], [17] to obtain the ER$_2$, HE$_2$ and MAP representations. For the service-time distribution, we also consider Exp, ER$_2$ and HE$_2$ distributions. The mean service rate μ is set to 1, while the arrival rate is set to achieve a target utilization $U = \lambda/c\mu$ for the system without replication (NR-utilization).



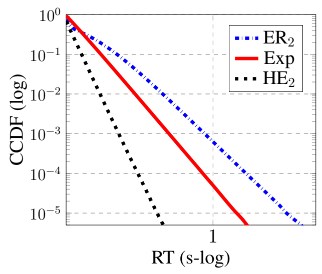
(a) PDF of service dist.



(b) CCDF with 2 replicas



(c) CCDF with 5 replicas

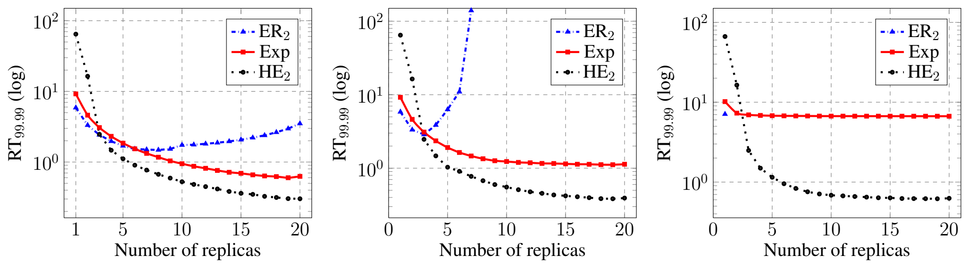


(d) CCDF with 10 replicas

Fig. 3: Service time PDF and RT CCDF

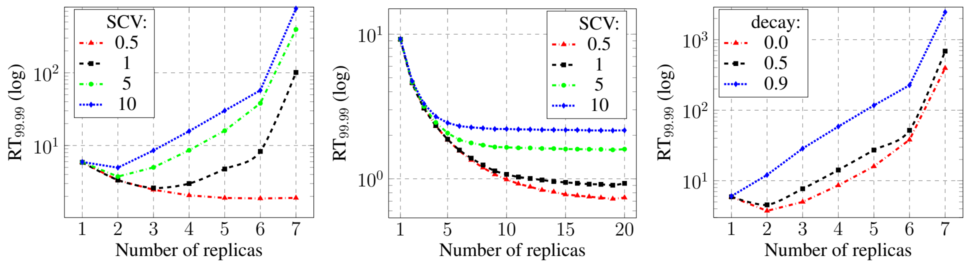
A. *The effect of canceling*

To evaluate the effectiveness of canceling, we compare the RTs obtained with and without canceling, using 2 replicas as an example. We consider an instance with HE$_2$ arrivals (SCV = 10), ER$_2$ services, and 20 servers. We first consider a relatively low NR-utilization of 0.25, and then increase it to 0.4, which ensures that the case with replication and canceling is stable. The results without canceling are obtained with a simulation model, as this is not the focus of the model introduced in this paper. Figure 2 depicts the RT complementary CDF (CCDF) offered by the systems without replication (NR), with canceling (C), and without canceling (NC). We observe that under the relatively low NR-utilization case (0.25), the RT CCDFs of both replication approaches are very similar, with the one of the case with canceling being slightly smaller. In both replication cases, the tails are much shorter than without replication. The reduction in RTs is caused by the introduction of replicas as it allows the selection of the first replica that completes service. In this case, the 0.25 NR-utilization becomes 0.31 and 0.50 after introducing one extra replica, with and without canceling, respectively. However, when the NR-utilization increases to 0.4, as shown in Figure 2(b), the utilization in the case without canceling increases to 0.81, leading to a much longer tail and a RT that is even larger than the one obtained without replication. Instead, the RT obtained with canceling only increases the utilization to 0.5, causing the RT to be much smaller than the one without canceling. The canceling mechanism thus limits the additional load introduced by the replicas,improving the benefits of replication for latency-tolerance.



(a) NR-utilization: 0.2 (b) NR-utilization: 0.5 (c) NR-utilization: 0.9

Fig. 4: RTs under different NR-utilizations and number of replicas



(a) ER$_2$ services (b) Exp services (c) ER$_2$ services

Fig. 5: Effect of the arrival process

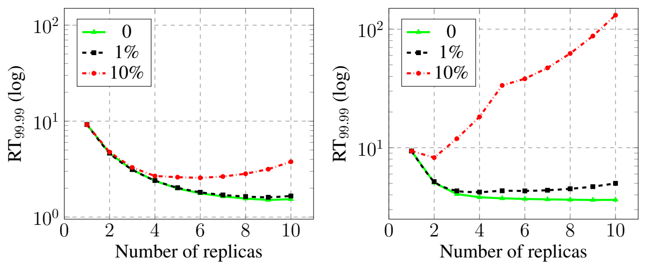
B. The effect of the service-time distribution

We now illustrate the impact of the service-time distribution on the system performance, as measured by the RT. We focus on the value x such that $P(RT \leq x) = 0.9999$, labeled RT99.99, but similar results hold for other values of the RT distribution. As a reference, Figure 3(a) compares the probability density functions (PDFs) of the service-time distributions considered, where we see that the HE$_2$ distribution (SCV = 10) has the longest tail while the ER$_2$ has the shortest. Also, HE$_2$ has the highest probability of having short service times while ER$_2$ has the lowest. Assuming 20 servers, Exp arrivals, an NR-utilization of 0.1, and 2 replicas, Figure 3(b) shows how the RT CCDFs have a similar trend as the service times, with ER$_2$ and HE$_2$ having the shortest and longest tails, respectively. When the number of replicas increases to 5, as in Figure 3(c), the RT tail shrinks for all service-time distributions, but this change is largest for HE$_2$ services, which offers the shortest tail. This is further reinforced when the number of replicas increases to 10, as in Figure 3(d), where the tail under ER$_2$ services becomes the longest, and the order is reversed compared to the 2-replica case. The advantage of the HE$_2$ distribution lies on its larger proportion of small jobs. Since the replication mechanism allows the selection of the first replica that completes service, a larger number of replicas increases the probability that a job benefits from at least one of its replicas having a short service time. Replication with canceling is thus more effective if the probability of a short service time is non-negligible. Highly concentrated processing time distributions, like the ER$_2$, offer less chances to gain from replication.

C. The effect of the number of replicas and the utilization

We now consider a scenario with 20 servers, starting with a relatively low NR-utilization of 0.2. Figure 4(a) depicts the RT99.99 obtained under three service-time distributions, varying the number of replicas between 1 and 20. We observe that, under Exp and HE$_2$ (SCV = 10) services, the RT99.99 decreases consistently as the number of replicas increases, achieving the lowest RT99.99 when the number of replicas equals the number of servers. Instead, for ER$_2$ services, the RT99.99 decreases until the number of replicas reaches 8, and increases afterward. These different behaviors arise from the limited advantage that can be gained from replication under ER$_2$, as service times are highly concentrated. Instead, for Exp and HE$_2$ services, the larger number of replicas increases the probability of having a short job service time. Notice that having more replicas than servers offers no benefit, as at most c replicas can start service at the same time.

When the NR-utilization increases to 0.5, as shown in Figure 4(b), the RT99.99 under ER$_2$ services increases dramatically, achieving a minimum RT99.99 with 3 replicas, and increasing thereafter until the system becomes unstable with 7 or more replicas. Further, under ER$_2$ services, increasing the NR-utilization to 0.9, as in Figure 4(c), makes the system become unstable when introducing only one extra replica. However, under the Exp and HE$_2$ service-time distributions, the RT99.99 keeps decreasing although the main gain is obtained with the first few replicas. In particular, when the NR-utilization is 0.9, the main reduction in RTs is achieved with the first and second replicas, with marginal improvements beyond this number. Replication with canceling can thus be beneficial even under *high loads*, as long as the *replica* service-



(a) Exp, NR-utilization:0.4 (b) MAP, NR-utilization:0.4

Fig. 6: RT99.99s under different overhead with Exp services

time distribution offers enough variability, as it increases the chances of a short *job* service time.

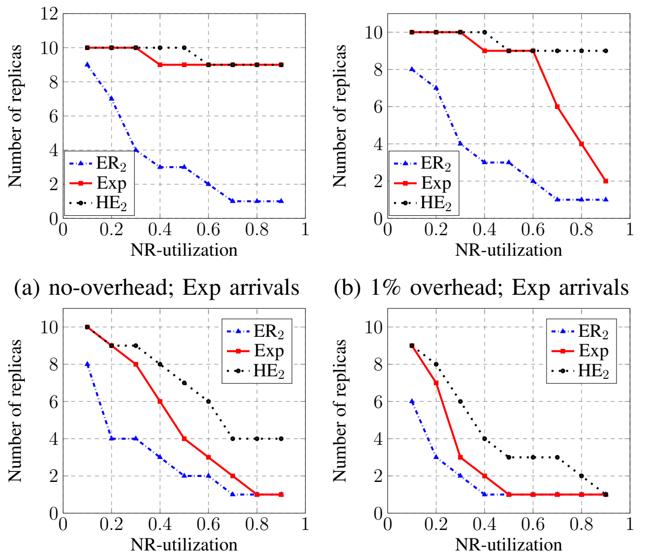
D. The effect of the arrival process

We now look into the effect that the statistical characteristics of the job arrival process have on the RTs experienced under different replication levels. In addition to considering different values for the IATs' SCV, we also modify the decay rate of the auto-correlation function (ACF) of the IAT sequence. Here we assume 20 servers and a 0.5 NR-utilization, as an example. Figure 5(a) shows how the RT99.99 increases as the IAT SCV increases, under ER$_2$ services, reflecting how the variability of the arrival process affects the system performance. In particular, when the SCV is 0.5, the RT99.99 decreases consistently with the an increasing number of replicas. Instead, for a larger IAT SCV, the RT99.99 decreases only with the first few replicas. On the other hand, if Exp services are assumed, as in Figure 5(b), the RT99.99 also decreases with the number of replicas when the SCV is low. However, when the SCV increases to 5 and 10, the reduction in the RTs is limited, as the main reduction is achieved with the first 3 replicas.

In Figure 5(c), we fix the SCV to 5 and consider, for ER$_2$ services, various values for the ACF decay rates of the IAT sequence. For an ACF decay rate of 0.5, the RT99.99 decreases with the first replicas but increases thereafter. For a larger ACF decay rate of 0.9, it is optimal not to adopt replication, as the RT99.99 increases with the first replica. As a high auto-correlation leads to a high probability that short IATs come in bursts, replication during bursty periods increases the probability of causing a high utilization, and thus longer delays. The IATs' variability and auto-correlation must therefore be considered when deciding whether to replicate or not, as these features limit the room for replication.

E. Introducing the canceling overhead

So far we have assumed that the cost of removing replicas is negligible. However, the canceling operation may require a significant amount of time, compared to the replica processing time. To model the canceling overhead, we introduce an additional phase $\triangle$ in the replica service-time distribution such that, when the replica completes service, it has to spend some time in phase $\triangle$ canceling the remaining replicas of this job before entering the absorbing phase C. For example, assuming



(a) no-overhead; Exp arrivals (b) 1% overhead; Exp arrivals

(c) 10% overhead; Exp arrivals (d) 10% overhead; MAP arrivals

Fig. 7: Optimal number of replicas

a mean canceling overhead of 0.1 in Example (3), we have

$$[S_R|\boldsymbol{S}_R^*] = \begin{array}{c} \\ 1 \\ 2 \\ \triangle \end{array} \begin{array}{cccc} 1 & 2 & \triangle & C \\ \end{array} \left(\begin{array}{ccc|c} -2 & 2 & 0 & 0 \\ 0 & -2 & 2 & 0 \\ 0 & 0 & -10 & 10 \end{array}\right),$$

where the replica that completes service enters the phase $\triangle$ before entering the absorbing phase C.

Figure 6(a) shows how the RT99.99 under 0% and 1% overhead, as a proportion of the replica mean service time, are very similar, decreasing with an increasing number of replicas, assuming Exp arrivals and services, 10 servers and a 0.4 NR-utilization. However, when the overhead increases to 10%, the RT99.99 reaches a minimum at 6 replicas, and addition replicas degrade the RT. In Figure 6(b), we consider the same cases but under MAP arrivals, where we immediately observe larger RTs caused by the bursty traffic. The difference is such that, for an overhead of 10%, the RT99.99 decreases only with the first replica, and even a 1% overhead can make a difference in the number of replicas that achieves the minimum RT.

VII. Optimal replication and resource provisioning

As we have seen, the potential benefits of replication depend on a number of factors, including the arrival and service processes, the overhead, the NR-utilization, and the number of replicas. In this section, we first explore the optimal number of replicas to adopt to minimize the offered RT. Further, we show how the model can support resource provisioning decisions with SLO guarantees on RT percentiles.

A. Optimal replication

We are interested in finding the number of replicas r^* that minimizes the p-th RT percentile (RT$_p$). Based on the proposed

Algorithm 1 Computing the optimal number of servers c^*

```
Require: $c_{\max}$, $r_{\max}$, $RT_{\max}$, $p$
1: for $r = 1 : r_{\max}$ do
2:     $c(r) \leftarrow \min c \in \{1, \dots, c_{\max}\} : RT_p(c, r) \leq RT_{\max}$.
3:     if $c(r) \leq r$ then break end if
4: end for
5: $c^* \leftarrow \min c(r)$, $R \leftarrow \{r : c(r) = c^*\}$.
6: if $|R|$ = 1 then return $c^*$, $r^*$.
7: else return $c^*$, and $r^* = \text{argmin}_{r \in R} RT(c^*, r)$.
8: end if
```

analytical model, we formulate this problem as

$$\begin{aligned} r^* = \text{argmin} \quad & RT_p(r), \\ \text{s.t.} \quad & r \in \{1, 2, \dots, c\}, \\ & U(r) < 1, \end{aligned}$$

where $\text{RT}_p(r)$ is the RT_p obtained when r replicas are implemented, c is the number of servers, and the constraint on $U(r)$, as defined in Eq. (7), ensures stability. This optimization problem is solved using a line-search method [18], taking advantage of the observed quasi-convexity of the RT percentiles as a function of the number of replicas.

Figure 7(a) shows the optimal number of replicas for a system with 10 servers and Exp arrivals, under different NR-utilization levels, assuming a negligible overhead. Under HE_2 services (SCV = 10), it is optimal to adopt as many replicas as the number of servers as long as the NR-utilization is under 0.6. Beyond this point the optimal number of replicas decreases to 9. Under ER_2 services instead, the optimal number of replicas consistently decreases with a larger NR-utilization, until it is optimal to not replicate when the NR-utilization reaches 0.7. It can be seen from Figures 7(b) and 7(c) that with the increase of the overhead, the optimal number of replicas decreases. For instance, for Exp services and an NR-utilization of 0.7, it is optimal to adopt 9, 6 and 2 replicas, for an overhead of 0%, 1%, and 10%, respectively.

Figure 7(d) considers the case of MAP arrivals and 10% overhead, showing how the optimal numbers of replicas in this case is less than or equal to the same number under Exp arrivals, given the same NR-utilization and service process. For instance, under 10% overhead and Exp services, replication is optimal if the NR-utilization is at most 0.8 for Exp arrivals, but this is only valid for NR-utilizations up to 0.5 for MAP arrivals. The optimal number of replicas is therefore affected by the variability and auto-correlation of the arrival process, as well as the overhead and the NR-utilization.

B. SLO-driven resource provisioning

We now rely on the analytical model proposed to determine the minimum number of servers c^* needed to meet an SLO on the RT_p, i.e., the SLO mandates that $p\%$ of the requests are served in at most $\text{RT}_{\max}$ seconds. Assuming at most $c_{\max}$ servers are available, and the number of replicas is upper bounded by $r_{\max}$, the optimization strategy is described in Algorithm 1. To find the optimal number of servers c^*, for each possible number of replicas r between 1 and $r_{\max}$, we rely on a line-search method to find the minimum number of servers $c(r)$ required to satisfy the RT SLO. The search terminates when the $c(r)$ found is equal to or less than the current r, since having more replicas than servers offers no additional benefit. As it is possible that the optimal number of servers c^* is achieved by various values for the number of replicas, collected in set R, we select the number of replicas that achieves the lowest RT. The selected number of replicas is labeled r^*. In the next section we illustrate this resource provisioning strategy, considering realistic traffic patterns.

TABLE II: Relative error (%) between the moments

Log	Dist.	first	second	third
RICC	HE_2	1.25E-11	3.18E-11	5.04E-11
(SCV=10.36)	Exp	0	82.39	97.14
Intel	HE_2	3.92E-12	3.04E-12	4.97E-12
(SCV=29.99)	Exp	0	93.55	99.84

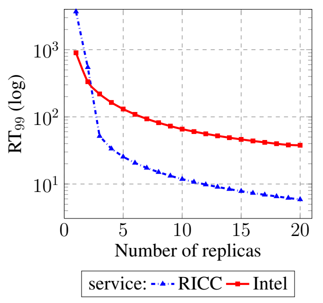


Fig. 8: Effect of services

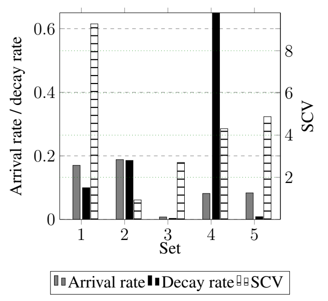


Fig. 9: Arrival statistics

VIII. Case-study: RICC and Intel logs

In this section, we consider realistic traffic patterns in CCs by making use of the RIKEN Integrated Cluster of Clusters (RICC) and the Intel NetBatch Grid logs, available on the Parallel Workloads Archive [7], parameterizing the analytical model with single-task jobs that completed service successfully. The IATs show a strong daily and hourly cycle, thus we divide the arrival data into sets according to the day of the week and the hour of the day. We also observe high variability and auto-correlation in the IATs. Thus, for each hour we fit the first three moments, and the ACF decay rate of the IATs, into a MAP of second order, with the method in [19]. In addition, given the variability observed for the service times, we use the method in [20] to match the empirical first three moments with an acyclic PH distribution. For comparison, we also fit the service times to an Exp distribution, which matches the first moment only. Table II shows the relative errors between the observed and the fitted moments. Clearly, the exponential fitting fails to capture higher moments, displaying large errors for both the RICC and the Intel traces.

We compare the two service-time distributions obtained from the logs, scaling them to have the same mean service time, but keeping their higher moments. Figure 8 depicts the RT99 for these two cases, with an increasing number of replicas, Exp arrivals, NR-utilization of 0.2, and 20 servers. Although in both cases the RT99 decreases with the increasing number of replicas, it decreases much faster for RICC service times. In fact, for $r = 1, 2$, the RT99 is larger for RICC than

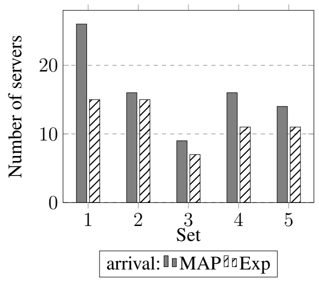


Fig. 10: Number of servers

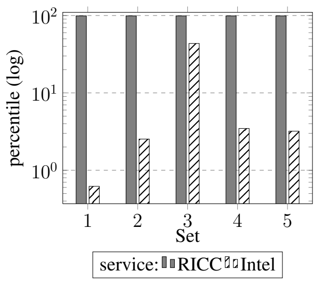


Fig. 11: RT percentiles

for Intel service times, but this order is inverted for larger values of r. Considering the fitted service-time distributions, the initial probability vectors for RICC and Intel service times are $[0.15, 0.85]$ and $[0.01, 0.99]$, respectively, with each phase offering different service times. This implies that for Intel service times most of the requests start in the second phase, and only around 1% start in the first phase, while for RICC service times the options are more varied. The result is that the introduction of replicas has more potential for RICC service times, as it is more likely that the replicas follow different paths, experiencing different service times, and potentially reducing the *job* service times.

From the daily and hourly partition of the data, we select 5 data-sets for the RICC case, which have the mean arrival rates, SCVs, and ACF decay rates shown in Figure 9. For each of these 5 conditions, we solve the resource provisioning problem introduced in Section VII, with $c_{\max} = 30$, $r_{\max} = 5$, and $RT_{\max} = 26$ for the 99-th percentile (RT99). The results, depicted in Figure 10, show how the effect of the arrival rate appears dominant as the trend of the number of servers is similar to that of the arrival rate. For example, the set 3 has the lowest arrival rate and requires the least number of servers to satisfy the RT99 SLO. However, both the SCV and ACF play a significant role. For instance, the sets 1 and 2 have a similar arrival rate, but the SCV of set 1 is much larger, leading to the minimum number of servers needed to be 26, while only 16 servers are needed for set 2. Also, comparing sets 4 and 5, set 5 has a larger arrival rate and SCV than set 4; however, set 4 requires more servers to achieve the RT99 SLO. This is caused by the large ACF decay rate of set 4, which is 0.649, while that of set 5 is almost 0. If in this dimensioning problem we replace the MAP by Exp arrivals, we obtain that the system requires less servers to satisfy the SLO on RT99, as shown in Figure 10. For instance, the number of servers needed for sets 1 and 2 are very similar under Exp arrivals, as now the variability and auto-correlation of the IATs are ignored. As a result, using the dimensioning proposed for set 1 under Exp arrivals will cause a violation of the RT SLO, as the resources are not enough to cope with the IATs variability.

We conclude this section by taking the dimensioning obtained above under MAP arrivals for the 5 different RICC cases, but using the scaled Intel service times. Figure 11 depicts the percentile that corresponds to $\mathrm{RT}_{\max} = 26$, achieved under RICC and Intel service times, using the RICC dimensioning. Clearly, the percentiles obtained using the Intel service times are much smaller than the 99% objective. For instance, for data-set 1, the $\mathrm{RT}_{\max}$ objective is only achieved by 0.62% of the requests under Intel service times, and by 99.02% under RICC service times. This illustrates how important the service-time higher moments actually are, as ignoring them may lead to SLO violations or over-provisioning.

IX. Conclusion

The results presented in this paper illustrate that replication with canceling has a significant potential to keep latency low, especially the tail of the RT distribution. This potential, however, depends on a number of factors, including the the arrival and service processes, the utilization, and the number of replicas. Notably, the service times play a key role, as a highly concentrated distribution offers limited chances to benefit from replication. As the workload and utilization change dynamically, the introduction of replication with canceling would require an online decision system to determine the adequate replication level, given the current conditions.

References

[1] A. Keren and A. Barak, "Opportunity cost algorithms for reduction of I/O and interprocess communication overhead in a computing cluster," *IEEE TPDS*, vol. 14, pp. 39–50, 2003.

[2] G. Kandaswamy, A. Mandal, and D. A. Reed, "Fault tolerance and recovery of scientific workflows on computational grids," in *IEEE CCGRID*, 2008.

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

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

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

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

[7] "Parallel workloads archive," http://www.cs.huji.ac.il/labs/parallel/workload/.

[8] P. G. Harrison and Z. Qiu, "Performance enhancement by means of task replication," in *EPEW*, 2013.

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

[10] 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.

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

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

[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] 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.

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

[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, G. Horváth, and K. Gross, "Explicit inverse characterizations of acyclic MAPs of second order," in *EPEW*, 2007.

[18] J. Kiefer, "Sequential minimax search for a maximum," *Proc. Amer. Math. Soc.*, vol. 4, pp. 502–506, 1953.

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

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