Enhancing Reliability and Response Times via Replication in Computing Clusters

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

***Abstract*—Computing clusters have been widely deployed for scientific and engineering applications to support intensive computation and massive data operations. As applications and resources in a cluster are subject to failures, fault-tolerance strategies are commonly adopted, sometimes at the expense of additional delays in job response times, or unnecessarily increasing resource usage. In this paper, we explore *concurrent replication with canceling*, a fault-tolerance approach where jobs and their replicas are processed concurrently, and the successful completion of either triggers the removals of its replica. We propose a stochastic model to study how this approach affects the cluster service level objectives (SLOs), particularly the offered response time *percentiles*. In addition to the expected gains in reliability, the proposed model allows us to determine the regions of the utilization where introducing replication with canceling effectively reduces the response times. Moreover, we show how this model can support resource provisioning decisions with reliability and response time guarantees.**

I. INTRODUCTION

Fueled by the ever-growing demand of computation-intensive and massive-data operations in scientific and engineering applications, computing clusters (CC) have been widely deployed, providing a cost-effective, high-performance environment [1], [2]. A CC is usually composed of tens or hundreds of processing elements, interconnected by a high-speed communication network, which provides computing services either to a single or multiple users. CCs, as any computing system, are subject to failures, and their overall reliability deteriorates with increasing scale and complexity [1], degrading the offered quality of service (QoS). Failures can be at the application level, also referred to as request failures, caused for instance by deadlocks, errors in the input data, communication errors [3], timeouts of resources with constrained availability, or application outputs exceeding latency requirements. On the other hand, server failures can be caused by hardware failures, or, in the case of virtualized CCs, by errors in the management of virtual machines.

To address this problem, various fault-tolerance mechanisms have been developed, among which, checkpoint/restart has been the most prevailing [4]. In this approach, a failed request is re-executed, either from the beginning or from the last checkpoint, on the same server in the case of a request failure. In the case of a server failure, the clustering software can switch the request to a standby resource, without administrative intervention [5]. Although effective to handle failures, a major concern with this approach is the additional delay it introduces, degrading the application QoS, which is a major issue for deadline-driven applications [6], [7]. Moreover, this approach has been found to impose large overheads and storage requirements to read and write checkpoints [4], [8].

An alternative approach is to process requests' replicas concurrently, improving the reliability as it is sufficient if one of the replicas succeeds. Concurrent replication can thus handle unpredictable failures, unless they occur to all replicas simultaneously. This approach has been recently considered for deadline-driven and mission-critical systems [6], [9]–[11] to improve both latency and reliability. For example, [10] proposes an outlier mitigation strategy by processing requests clones simultaneously to mitigate the effect of latency. This approach has become appealing with the observation that most clusters are highly underutilized. For instance, the overall utilization of data center servers has been found to be around 18% [12], while traces from the Facebook cluster show that resources remain idle, with median slot, CPU, and memory utilizations under 20% [10]. However, concurrent replication may introduce unacceptable additional load, leading to excessive resource and energy usage, to the point of degrading the offered response times (RTs). An approach to limit the additional load is to let servers share updates on the status of their replicas, and cancel any outstanding replica immediately after the first successful result is returned [9]. As a result, this approach, referred to as *replication with canceling*, has the potential to better meet service-level objectives (SLOs) on both reliability and RTs. This approach has been recently evaluated for the case of two servers with independent queues [13], and for the case of synchronous parallel processors [14]. In this paper we focus on the case of a CC, where jobs are processed asynchronously, posing additional modeling challenges as this requires keeping track of the state of all the servers.

To study the effect of replication with canceling we introduce a stochastic model that estimates the RT *distribution*. The ability to determine the RT distribution, instead of simply the mean [13], [14], allows us to evaluate percentile SLOs, where the interest lies on the RT faced by the majority, e.g., the 95%, of the requests. As described in Section V, the proposed model is able to determine the regions of the utilization where replication with canceling is able to effectively reduce the RT percentiles. A second advantage of the model is its ability to capture very general request arrival processes, by relying on Markovian arrival processes (MAPs). The relevance of this generality is illustrated in Section VI, where we conduct a trace-driven case study based on a real workload from an integrated cluster [15]. The results show that the MAP process is much better at capturing the arrival patterns observed in the trace than the traditional Poisson process. Further, the model illustrates the important effect that the variability and

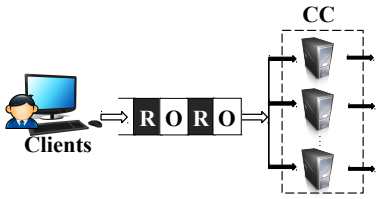


Fig. 1: Reference model

correlation of the arrival process have on the offered RT percentiles. In fact, ignoring such effects may lead to large SLO violations. We further illustrate the potential of the model in Section VI, where it forms the basis of an SLO-driven resource provisioning strategy. The results highlight how provisioning decisions can be improved by explicitly considering the replication strategy, the RT percentiles and the arrival process variability. The next section presents the reference model, and Sections III and IV introduce the stochastic model.

II. BACKGROUND

A. Reference model

We consider a system consisting of a central queue and c distributed, homogeneous and independent servers. Requests form a single queue in the order of arrival and join the next server that becomes available with first-come first-served (FCFS) scheduling. In a system with replication, one extra replica is adopted to increase the reliability. As shown in Figure 1, for each request O submitted to the CC, a single replica R is adopted and sent right after the original request to the central queue, and the CC replies with the result from whichever replica completes first. We denote by *request/replica* the original request or its replica, and by *job* a request and its replica. To reduce unnecessary workload, servers communicate updates on the status of their replicas to each other. When a replica completes execution, it immediately cancels any outstanding replica of this job.

This model may represent either request or server failures, where the time-to-failure is exponentially distributed with rate α, a common assumption in reliability engineering [16], [17]. In the case of a request failure, we assume that the server is not affected by the failure, and continues serving the next request in front of the queue. In the case of a server failure, the request in service is lost, and the failed server is immediately replaced with a standby server. This can be achieved through active redundancy, where all redundant servers are in operation at all times, and in case of a primary server failure, the clustering software automatically switches all the requests to a standby server [18]. For warm and hot standby systems, when the restoration time is too short to impact the system performance, the switching time can be modeled as zero [5], [18].

In the next sections we propose a model to evaluate the performance of a CC adopting replication with canceling approach, and to compare it with the case without replication. While the *request* processing times are exponentially distributed with rate μ, we will show that the *job* processing times follow a Phase-type (PH) distribution, whose parameters depend on the overall system state. Further, motivated by the high variability observed in CC [15], the model assumes that requests are submitted according to a Markovian Arrival Process (MAP). Compared to traditional Poisson arrivals, a MAP can represent more general inter-arrival times (IATs), including their high variability and auto-correlation. We now introduce PH and MAP for later reference.

B. Phase-type distributions

PH distributions have gained popularity thanks to their ability to capture very general behaviors, while maintaining some of the analytical tractability of the exponential distribution. A PH random variable X [19] represents the absorption time in a Markov Chain (MC) with $n+1$ states, where the states $\{1, 2, \ldots, n\}$ are transient and state 0 is absorbing. We denote it as PH($\boldsymbol{\tau}, T$), where $\boldsymbol{\tau}$ is the $1 \times n$ vector of the initial probability distribution, T is the $n \times n$ sub-generator matrix, and the vector $\boldsymbol{t} = -T\boldsymbol{e}$ holds the absorption rates, with $\boldsymbol{e}$ a column vector with unit elements. Its cumulative distribution function (CDF) is $F(x) = 1 - \boldsymbol{\tau} exp(Tx)\boldsymbol{e}$, for $x \geq 0$, and its expected value is $E[X] = -\boldsymbol{\tau} T^{-1}\boldsymbol{e}$.

C. Markovian Arrival Processes

MAPs are widely used to represent arrival streams with general and correlated IATs [19]. The continuous-time MAP [19] is a marked MC with generator $D = D_0 + D_1$, where the elements of D_0, resp. D_1, hold the rates associated to transitions without, resp. with, arrivals. D_1 is a non-negative matrix, and D_0 has non-negative off-diagonal entries and strictly negative diagonals, with $(D_0 + D_1)\boldsymbol{e} = \boldsymbol{0}$. The arrival rate is given by $\lambda = \boldsymbol{\gamma} D_1 \boldsymbol{e}$, where $\boldsymbol{\gamma}$ is the stationary distribution of the underlying MC, i.e. $\boldsymbol{\gamma} D = 0$ and $\boldsymbol{\gamma}\boldsymbol{e} = 1$. A MAP can model a renewal process with PH($\boldsymbol{\sigma}, S$) IATs by setting $D_0 = S$ and $D_1 = \boldsymbol{s}\boldsymbol{\sigma}$, where $\boldsymbol{s} = -S\boldsymbol{e}$.

III. THE QUEUE LENGTH DISTRIBUTION

In this section we focus on the computation of the queue-length distribution for the reference model introduced above. While relevant on its own, we will make use of the queue-length distribution to obtain the job response time distribution in Section IV. The system without replication can be cast as a MAP/M/c queue with service rate $\mu + \alpha$ and its queue length distribution can be obtained using the method in [19]. However, for the system with replication, since each job is composed of two replicas, a more detailed analysis is required.

For the system with replication, we set up a two-dimensional MC $\{(L(t), J(t)) | t \geq 0\}$, where $L(t)$, the *level* variable, holds the number of jobs in the system, and $J(t) = (A(t), S(t))$, the *phase* variable, holds both the phase of the arrival process $A(t)$ and of the service process $S(t)$. The arrival process is a MAP with m_a phases and parameters D_0 and D_1. To model the state of the service process $S(t)$ we need to introduce the following definitions. Let an *all-busy* period refer to the time period when all the servers are busy, and a *not-all-busy* period to the period when at least one server is idle [20]. During a not-all-busy period, we define the service state to be $S(t) = (L_2(t), L_1(t))$, where $L_i(t)$ is the number of jobs with i replicas in service. A job has only one replica in service if the other one already failed. For instance, for a system with 3 servers and 1 job in process, either one or both replicas of this job are in service, leading the possible service phases in level 1 to be $\{(1, 0), (0, 1)\}$. On the other

hand, during an all-busy period we define the service state to be $S(t) = (L_2(t), Y(t))$, where $Y(t)$ represents the state of the youngest job in service. $Y(t)$ takes values from $\{2, O, R\}$, where: 2 means both replicas of the youngest job are in service; O means only the first replica is in service while the second one is waiting in front of the queue; and R means one of the youngest job's replicas failed while the other one is still in service. In this case there is no need to record the number of jobs with one replica in service, since this can be calculated as $c - 2L_2(t)$ when all the servers are busy. In the example with 3 servers, but 3 jobs in the system, the possible service phases are $\{(1, O), (1, 2), (1, R), (0, O), (0, R)\}$, where service phase $(1, O)$ represents the case where one job has both replicas in service, the youngest job in service has its first replica in service and its second replica waiting, and one job is waiting in the queue. The number of service phases for level k is given by

$$m_s(k) = \begin{cases} k+1, & k \leq \lfloor c/2 \rfloor, \\ 2k - \lfloor c/2 \rfloor + 1, & \lfloor c/2 \rfloor < k < c. \end{cases} \quad (1)$$

From level c onwards, all the levels have the same number of service phases $m_s = 2c - \lfloor c/2 \rfloor$, where $\lfloor a \rfloor$ is the largest integer smaller than or equal to a. We order the state-space lexicographically and define level k as the set of states where k jobs are present, i.e., the set $\{(k, j), 0 \leq j \leq m_k\}$, where $m_k = m_a m_s(k)$, for $0 \leq k < c$, and $m_k = m_a m_s$ for $k \geq c$.

Notice that levels 0 to $\lfloor c/2 \rfloor - 1$ correspond to not-all-busy periods, while levels greater and equal to c correspond to all-busy periods. However, levels $\lfloor c/2 \rfloor$ to $c - 1$ have both not-all-busy and all-busy states since a job may have either one of or both replicas in service. The MC therefore has a Quasi-Birth-and-Death (QBD) structure [19], with level-dependent transition rates between levels 0 and c, and level-independent rates from level $c + 1$. As a result, the infinitesimal generator of the MC can be written as

$$P = \begin{bmatrix} A_1^{(0)} & A_2^{(0)} & 0 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ A_0^{(1)} & A_1^{(1)} & A_2^{(1)} & 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & 0 & A_0^{(c)} & A_1^{(c)} & A_2^{(c)} & 0 & \cdots \\ 0 & \ddots & \ddots & \ddots & 0 & A_0 & A_1 & A_2 & \cdots \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 & A_0 & A_1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}.$$

The matrices $A_0^{(k)}$, $A_1^{(k)}$, and $A_2^{(k)}$ hold the transition rates from level k to levels $k - 1$, k, and $k + 1$, respectively. From level $c+1$ onwards, these matrices do not depend on the level and are simply labeled A_0, A_1, and A_2. While a successful service completion of any replica in service leads to a level decrease, a failure may not necessarily lead to the change of the level variable. For example, if one job has both replicas in service, and one of them fails, the number of jobs in the system is not altered as its other replica remains in service. We therefore model the evolution of the service phase in level k as a MAP with parameters $(m_s(k), C_0^{(k)}, C_1^{(k)})$, removing the superindex k for levels greater than or equal to $c + 1$ as these transitions become independent of the level. The matrices $C_0^{(k)}$, resp. $C_1^{(k)}$ hold the service transition rates without, resp. with, a change of the level. Letting I_n denote the $n \times n$ identity matrix, the QBD blocks for levels $k \geq c + 1$ are given by

$$A_0 = C_1 \otimes I_{m_a}, A_1 = C_0 \otimes I_{m_a} + I_{m_s} \otimes D_0, A_2 = I_{m_s} \otimes D_1,$$

and similarly for levels $0 \leq k \leq c$, adding the super-index k.

Let the MC stationary probability vector $\boldsymbol{\pi}$ be partitioned as $\boldsymbol{\pi} = [\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \dots]$, where the $1 \times m_k$ vector $\boldsymbol{\pi}_k$ corresponds to level k, and its j-th element $\pi_{k,j}$ holds the probability that k jobs are in the system, while the phase is j. The QBD structure implies that $\boldsymbol{\pi}_{c+i} = \boldsymbol{\pi}_c R^i$, for $i \geq 1$, thus the stationary distribution for levels $k \geq c + 1$ can be obtained from $\boldsymbol{\pi}_c$ and R. The matrix R is the minimal non-negative solution to the matrix equation $R = A_2 + RA_1 + R^2 A_0$, and can be obtained with, for instance, the Cyclic Reduction algorithm [21]. To determine the sub-vector $[\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \dots, \boldsymbol{\pi}_c]$ we modify the transition rates within level c to $A_1^{(c)} + RA_0$, thus censoring the transitions beyond this level. The result is a finite level-dependent QBD, the stationary probability vector of which can be found with the linear level-reduction algorithm in [19]. Once the sub-vector $[\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \dots, \boldsymbol{\pi}_c]$ is obtained, it must be normalized with the condition $\sum_{k=0}^{c-1} \boldsymbol{\pi}_k \boldsymbol{e} + \boldsymbol{\pi}_c (I - R)^{-1} \boldsymbol{e} = 1$. Notice that the vector $\boldsymbol{\pi}$ exists if the QBD MC is positive recurrent, which occurs under the condition

$$\varrho = \frac{\boldsymbol{\alpha}_A A_2 \boldsymbol{e}}{\boldsymbol{\alpha}_A A_0 \boldsymbol{e}} < 1. \quad (2)$$

Here ϱ is the drift of the MC, and $\boldsymbol{\alpha}_A$ is the unique solution of the system $\boldsymbol{\alpha}_A A = \boldsymbol{0}, \boldsymbol{\alpha}_A \boldsymbol{e} = 1$, with $A = A_0 + A_1 + A_2$. This condition will be used as a constraint in the resource-provisioning strategy introduced later in Section VI.

We now describe how to determine the matrices $(C_0^{(k)}, C_1^{(k)})$ that govern the evolution of the service process.

1) Levels $0 \leq k \leq c$: For levels 0 to c, each level has a different number of service phases, as in Equation (1), and transitions may lead to a switch between not-all-busy and all-busy periods states. Service transition rates depend on different conditions as shown in Table I, where $\beta = \mu + \alpha$. Three different conditions are possible: when $k_2 > c - k$, there is at least one job waiting in the queue; when $k_2 = c - k$, either the queue is empty or the second replica of the youngest job is in the queue; and when $k_2 < c - k$, at least one server is idle. As an example, assume $c = 3$, then the service MAP matrices $C_1^{(2)}$ and $C_0^{(2)}$ for level 2 are given by

$$C_1^{(2)} = \begin{array}{c} \\ (1,O) \\ (1,2) \\ (1,R) \\ (0,2) \end{array} \begin{array}{c} \begin{array}{cc} (1,0) & (0,1) \end{array} \\ \begin{pmatrix} 3\mu & 0 \\ \beta & 2\mu \\ \beta & 2\mu \\ 0 & 2\beta \end{pmatrix} \end{array}, C_0^{(2)} = \begin{array}{c} \\ (1,O) \\ (1,2) \\ (1,R) \\ (0,2) \end{array} \begin{array}{c} \begin{array}{cccc} (1,O) & (1,2) & (1,R) & (0,2) \end{array} \\ \begin{pmatrix} -3\beta & 2\alpha & \alpha & 0 \\ 0 & -3\beta & 0 & 2\alpha \\ 0 & 0 & -3\beta & 2\alpha \\ 0 & 0 & 0 & -2\beta \end{pmatrix} \end{array},$$

respectively. For instance, in service phase $(1, O)$, the successful completion of any replica in service leads to the start of a not-all-busy period with rate 3μ. Instead, the failure of any replica of the job with both replicas in service leads to service phase $(1, 2)$ with rate 2α, continuing with the all-busy period.

2) Levels $k \geq c + 1$: For levels greater than or equal to $c+1$, the transitions are independent of the level. The transition rates of matrices C_0 and C_1 are summarized in Table II. As an example, consider service phase $(k_2, 2)$ ($k_2 \geq 1$) in level k, k_2 of the k jobs present have both replicas in service. From this state, either replica of the k_2 jobs may complete service and cancel its partner immediately, leading to service phase $(k_2, 2)$ in level $k - 1$, with rate $2k_2\mu$.

TABLE I: Transition rates for levels $0 \leq k \leq c$

Matrix	Condition	From	To	Rate
$C_0^{(k)}$	$k_2 > c-k$	$(k_2, 2)$	(k_2-1, O)	$2k_2\alpha$
		(k_2, O)	(k_2, R)	α
		(k_2, O)	$(k_2, 2)$	$2k_2\alpha$
		(k_2, R)	(k_2-1, O)	$2k_2\alpha$
	$k_2 = c-k$	$(k_2, 2)$	$(k_2-1, c-2k_2+1)$	$2k_2\alpha$
		(k_2, O)	(k_2, R)	α
		(k_2, O)	$(k_2, 2)$	$2k_2\alpha$
		(k_2, R)	$(k_2-1, c-2k_2+1)$	$2k_2\alpha$
	$k_2 < c-k$	(k_2, k_1)	(k_2-1, k_1+1)	$2k_2\alpha$
$C_1^{(k)}$	$k_2 > c-k$	$(k_2, 2)$	$(k_2, 2)$	$2k_2\mu$
		$(k_2, 2)$	(k_2, O)	$(c-2k_2)\beta$
		(k_2, O)	(k_2, O)	$(2k_2+1)\mu$
		(k_2, O)	$(k_2+1, 2)$	$(c-2k_2-1)\beta$
		(k_2, R)	$(k_2, 2)$	$2k_2\mu$
		(k_2, R)	(k_2, O)	$(c-2k_2)\beta$
	$k_2 = c-k$	$(k_2, 2)$	$(k_2-1, c-2k_2)$	$2k_2\mu$
		$(k_2, 2)$	$(k_2, c-2k_2-1)$	$(c-2k_2)\beta$
		(k_2, O)	$(k_2, c-2k_2-1)$	$(2k_2+1)\mu$
		(k_2, O)	$(k_2+1, 2)$	$(c-2k_2-1)\beta$
		(k_2, R)	$(k_2-1, c-2k_2)$	$2k_2\mu$
		(k_2, R)	$(k_2, c-2k_2-1)$	$(c-2k_2)\beta$
	$k_2 < c-k$	(k_2, k_1)	(k_2-1, k_1)	$2k_2\mu$
		(k_2, k_1)	(k_2, k_1-1)	$k_1\beta$

TABLE II: Transition rates for levels $k \geq c+1$

Matrix	From	To	Rate
C_0	$(k_2, 2)$	(k_2-1, O)	$2k_2\alpha$
	(k_2, O)	$(k_2, 2)$	$2k_2\alpha$
	(k_2, O)	(k_2, R)	α
	(k_2, R)	(k_2-1, O)	$2k_2\alpha$
C_1	$(k_2, 2)$	$(k_2, 2)$	$2k_2\mu$
	$(k_2, 2)$	(k_2, O)	$(c-2k_2)\beta$
	(k_2, O)	(k_2, O)	$(2k_2+1)\mu$
	(k_2, O)	$(k_2+1, 2)$	$(c-2k_2-1)\beta$
	(k_2, R)	$(k_2, 2)$	$2k_2\mu$
	(k_2, R)	(k_2, O)	$(c-2k_2)\beta$

TABLE III: Transition rates of $Q_S(k)$, $k \geq \lfloor c/2 \rfloor + 1$

Condition	Current state	Initial states	Transition rates
$k_2 > c-k$	$(k_2, 2)$	2	$2k_2\mu$
	$(k_2, 2)$	O	$(c-2k_2)\beta + 2k_2\alpha$
	(k_2, O)	O	$(2k_2+1)\mu$
	(k_2, R)	2	$2k_2\mu$
	(k_2, R)	O	$(c-2k_2)\beta + 2k_2\alpha$

IV. The Response Time Distribution

In this section, we obtain the response time (RT) distribution for the reference system with replication. Specifically, we show that the RT distribution has a PH representation that is obtained by considering the service time and waiting time separately. The system *without* replication, where the service times follow an exponential distribution, can be modeled as a MAP/M/c queue, which can be analyzed as in [20] to obtain a PH representation for the waiting time distribution. However, the case *with* replication requires a more detailed analysis due to the particular dynamics introduced by both replication and canceling. Further, we make extensive use of the $\boldsymbol{\pi}$ vector found in the previous section for the queue-length process.

A. The service time distribution

In the system with replication, the *job* service time does not follow an exponential distribution since it is composed of two replicas, whose execution depend on the system state. The service time however has a PH representation with parameters $\{\boldsymbol{\alpha}_{\text{service}}, S_{\text{service}}\}$. Let $Y(t)$ be the service state of a tagged job at time t, which takes values from $\{2, O, R\}$, as defined in Section III. We add two absorbing states S and F to represent the cases where the job completes service successfully or encounters a failure, respectively. Further, the holding times in each state are exponentially distributed, and we can describe the transition among the states as an MC with generator $S_{\text{service}} = \tilde{S}_{\text{service}} \otimes I_{m_a}$, where $\tilde{S}_{\text{service}}$ is given by

$$[\tilde{S}_{\text{service}}|\boldsymbol{t}_S|\boldsymbol{t}_F] = \begin{array}{c} \\ \text{(2)} \\ \text{(O)} \\ \text{(R)} \end{array} \begin{array}{c} \begin{array}{ccccccc} \text{(2)} & \text{(O)} & \text{(R)} & & \text{(S)} & & \text{(F)} \end{array} \\ \begin{pmatrix} -2\beta & 0 & 2\alpha & \vdots & 2\mu & \vdots & 0 \\ (c-1)\beta & -c\beta & \alpha & \vdots & \mu & \vdots & 0 \\ 0 & 0 & -\beta & \vdots & \mu & \vdots & \alpha \end{pmatrix} \end{array}.$$

We have labeled $\boldsymbol{t}_S$ and $\boldsymbol{t}_F$ the absorption vector to states S and F, respectively, while S_{service} is precisely the sub-generator matrix we need for the PH representation of the service time distribution. Notice that a job can only start service in phase 2 or O, since in R one of the replicas has already failed. Further, for phases 2 and R we only need to consider the state of the tagged job's replicas in process. Instead, in phase O, which only occurs when all the servers are busy, the first replica of the tagged job is in service while its second replica is waiting in front of the queue. Thus, if *any* of the other $c-1$ requests in service completes service or fails, the second replica of the tagged job starts service. This occurs with rate $(c-1)\beta$ and leads the tagged job to service phase 2.

Having obtained S_{service}, it remains to determine the initial probability vector $\boldsymbol{\alpha}_{\text{service}}$, which is the stationary probability that a job starts service in each of the phases $\{2, O, R\}$. In other words, $\boldsymbol{\alpha}_{\text{service}}$ is the distribution of the service phase of the youngest job in service, immediately after the initiation of a job service. This can be obtained from the queue-length process defined in Section III by multiplying the stationary distribution $\boldsymbol{\pi}$ by a matrix that holds the rates at which a new job starts service in each state, and the phase in which the service starts. We can therefore obtain $\boldsymbol{\alpha}_{\text{service}}$ as

$$\frac{\sum_{k=0}^{\lfloor c/2 \rfloor} \boldsymbol{\pi}_k Q_A(k) + \sum_{k=\lfloor c/2 \rfloor+1}^{c-1} \boldsymbol{\pi}_k (Q_A(k) + Q_S(k)) + \sum_{k=c}^{\infty} \boldsymbol{\pi}_k Q_S(k)}{\left(\sum_{k=0}^{\lfloor c/2 \rfloor} \boldsymbol{\pi}_k Q_A(k) + \sum_{k=\lfloor c/2 \rfloor+1}^{c-1} \boldsymbol{\pi}_k (Q_A(k) + Q_S(k)) + \sum_{k=c}^{\infty} \boldsymbol{\pi}_k Q_S(k) \right) \boldsymbol{e}},$$

where the denominator is simply the sum of the numerator entries. The matrices $Q_A(k)$ and $Q_S(k)$ hold the rates at which a new job starts service due to an arrival or to a service completion in level k, respectively, as described next.

The matrices $Q_A(k)$ depend on the level k and are defined up to level $c-1$, as beyond this level an arrival does not trigger a job service start. These matrices can be written as $Q_A(k) = \tilde{Q}_A(k) \otimes D_1$, where the (i,j)-th entry of $\tilde{Q}_A(k)$ is equal to 1 if the arrival of a job to a system with k jobs and service phase i causes the start of a new job in service state j; otherwise it is equal to 0. For example, in the case with $c = 3$ servers and $k = 2$ jobs present we have

$$\tilde{Q}_A(2) = \begin{array}{c} \\ \text{(1,O)} \\ \text{(1,2)} \\ \text{(0,2)} \\ \text{(1,R)} \end{array} \begin{array}{c} \begin{array}{ccc} \text{(2)} & \text{(O)} & \text{(R)} \end{array} \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{array}.$$

In service phase $(0, 2)$ one server is idle, enabling the start of a new job in phase O in case of an arrival. In the other phases, an arrival does not have this effect as all servers are busy.

The matrices $Q_S(k)$ are only relevant for states where all the servers are busy and at least one job is waiting in the queue, i.e., for states in levels $k \geq \lfloor c/2 \rfloor + 1$ satisfying the

TABLE IV: Transition rates of Q and Π

matrix	From	To	Rate
Q	(k_2, O)	(k_2, R)	α
	(k_2, O)	$(k_2+1, 2)$	$(c-2k_2-1)\beta$
	(k_2, O)	$(k_2, 2)$	$2k_2\alpha$
Π	(k_2, O)	(k_2, O)	$(2k_2+1)\mu$
	$(k_2, 2)$	$(k_2, 2)$	$2k_2\mu$
	$(k_2, 2)$	(k_2-1, O)	$2k_2\alpha$
	$(k_2, 2)$	(k_2, O)	$(c-2k_2)\beta$
	(k_2, R)	$(k_2, 2)$	$2k_2\mu$
	(k_2, R)	(k_2, O)	$(c-2k_2)\beta$
	(k_2, R)	(k_2-1, O)	$2k_2\alpha$

condition $k_2 > c-k$. Only in this case a service completion or failure may trigger the first job in the queue to start service. Further, for levels $c+1$ or greater we remove the index k as these matrices no longer depend on the level k. We let $Q_S(k) = \tilde{Q}_S(k) \otimes I_{m_a}$, where the (i,j)-th entry of $\tilde{Q}_S(k)$ holds the service and failure rate in service phase i, for $1 \leq i \leq m_s(k)$, that triggers the start of a job in service phase j. The transition rates of $Q_S(k)$ are shown in Table III. As an example, for $c=3$ and level $k=3$, the matrix $Q_S(3)$ is

$$Q_S(3) = \begin{array}{c} \\ \text{(1,O)} \\ \text{(1,2)} \\ \text{(0,O)} \\ \text{(1,R)} \\ \text{(0,R)} \end{array} \begin{array}{c} \begin{array}{ccc} \text{(2)} & \text{(O)} & \text{(R)} \end{array} \\ \begin{pmatrix} 0 & 3\mu & 0 \\ 2\mu & 2\alpha+\beta & 0 \\ 0 & 0 & 0 \\ 2\mu & 2\alpha+\beta & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{array}.$$

In service phase $(1,2)$, one job in service has one replica left, another, the youngest in service, has both replicas executing, and a third remains in queue. Thus, the completion of either replica of the youngest job causes the job in the queue to start service in state 2, as both replicas start at the same time. Instead, the completion or failure of the job with a single replica left, lets the job in queue to start service with its one replica only, thus starting in service state O.

B. The waiting time distribution

To determine the waiting time distribution, we observe the queue *only during the all-busy periods* [20], and define a bivariate Markov process $\{X(t), J(t) | t \geq 0\}$ [22]. Here the *age process* $X(t)$ is the total time-in-system of the youngest job in service, and the *phase process* $J(t) = (A(t), L_2(t), Y(t))$ holds the state of the arrival and service processes, as in the queue-length process defined in Section III. The phase process $J(t)$ thus takes $m = m_a m_s$ different values. The age process $\{X(t) | t \geq 0\}$ is right-continuous and takes values in $[0, \infty)$. $X(t)$ may either increase linearly with rate 1, if no *job* service completions occur, or may have downward jumps. 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. Different from the standard case treated in [20], where the service completion of any job in service triggers the start of a new job service, we need to keep track of the service state of the youngest job in service $Y(t)$ as part of the phase process. This is necessary because, with the introduction of replication and canceling, a service completion or failure does not necessarily triggers the start of a new job service, as a replica of the youngest job may be the next to start service. For instance, if $Y(t) = O$, a service completion or failure leads the second replica of the youngest job to start service, *without a new job* stating service.

To determine the PH representation $(\boldsymbol{\alpha}_{\text{wait}}, S_{\text{wait}})$ of the waiting time distribution, we rely on the stationary distribution $\boldsymbol{\rho}(x)$ of the $(X(t), J(t))$ process, which has a matrix exponential representation [22] $\boldsymbol{\rho}(x) = \boldsymbol{\rho}(0) exp\{Tx\}$, for $x > 0$. The $m \times m$ matrix T satisfies the non-linear integral equation

$$T = Q \otimes I_{m_a} + \int_0^\infty \exp\{Tt\} \Pi^{MAP}(t) dt,$$

where $\Pi^{MAP}(t) = \Pi \otimes exp\{D_0 t\} D_1$. Q and Π are $m_s \times m_s$ matrices that hold the transition rates of the service process associated to transitions without and with the start of a new job service, respectively [20], [22]. $Q + \Pi$ is the generator of the marginal service phase process, the transition rates of which are shown in Table IV. Letting

$$L = \int_0^\infty \exp\{Tt\} (I_{m_s} \otimes \exp\{D_0 t\}) dt,$$

we have $T = Q \otimes I_{m_a} + L\Pi \otimes D_1$. Integrating L by parts we obtain

$$TL + L(I_{m_s} \otimes D_0) = -I, \tag{3}$$

which can be used to compute T iteratively as in [22]. Notice that (3) is a Sylvester matrix equation that can be solved in $O(m^3)$ time with the Hessenberg-Schur algorithm [23].

The vector $\boldsymbol{\rho}(0)$ can be obtained by observing that its i-th element equals the probability that the phase $J(t)$ is i immediately after an arrival that finds $c-2$ or $c-1$ busy servers, thus starting an all-busy period. We thus define the $1 \times m_k$ vectors $\boldsymbol{u}_k$ that hold the number of busy servers in each state of level k. Further, let the i-th entry of the $1 \times m_k$ vector $\boldsymbol{v}_k$ be $[\boldsymbol{v}_k]_i = \mathbb{1}\{u_{k,i} = c-1 | c-2\}$, where the indicator function $\mathbb{1}\{x\}$ equals 1 if x is true and 0 otherwise. Therefore, $\boldsymbol{\rho}(0)$ can be computed as

$$\boldsymbol{\rho}(0) = \frac{\left(\sum_{k=0}^{c-1} (\boldsymbol{\pi}_k \circ \boldsymbol{v}_k)(K_k \otimes I_{m_a}) \right) (I_{m_s} \otimes D_1)}{\left(\sum_{k=0}^{c-1} (\boldsymbol{\pi}_k \circ \boldsymbol{v}_k)(K_k \otimes I_{m_a}) \right) (I_{m_s} \otimes D_1) \boldsymbol{e}},$$

where $\circ$ stands for the Hadamard product [24], and the $m_s(k) \times m_s$ matrix K_k holds the first $m_s(k)$ rows of I_{m_s}.

Notice that the vector $\boldsymbol{\alpha}_{\text{wait}}$ can be interpreted as the stationary distribution of the phase process immediately after a downward jump of the age process. Let $\boldsymbol{\sigma}$ be the steady-state marginal distribution of the phase process $J(t)$, and $\boldsymbol{\varphi}$ the probability that when the current state is $(t+dt, i)$, the next return to a state with age in the interval $[t, t+dt]$ is by way of a downward jump. $\boldsymbol{\alpha}_{\text{wait}}$ can thus be obtained as

$$\boldsymbol{\alpha}_{\text{wait}} = \upsilon \boldsymbol{\sigma} \circ \boldsymbol{\varphi} / ((\boldsymbol{\sigma} \circ \boldsymbol{\varphi}) \boldsymbol{e}).$$

Here the denominator serves to normalize the distribution, while υ is the probability that a job must wait. As υ is the probability that all servers are busy at a job's arrival instant, we can rely on the queue-length process to obtain it as

$$\upsilon = 1 - \frac{\sum_{k=0}^{c-1} (\boldsymbol{\pi}_k \circ \boldsymbol{w}_k)(I_{m_s(k)} \otimes D_1) \boldsymbol{e}_{m_s(k)}}{\sum_{k=0}^{\infty} (\boldsymbol{\pi}_k (I_{m_s(k)} \otimes D_1) \boldsymbol{e}_{m_s(k)})},$$

where the entries of $\boldsymbol{w}_k$ are given by $[\boldsymbol{w}_k]_i = \mathbb{1}\{u_{k,i} \leq c-1\}$, identifying the states with at least one idle server.

Following [22], we can obtain $\boldsymbol{\sigma}$ as

$$\boldsymbol{\sigma} = \int_0^\infty \boldsymbol{\rho}(t)dt = \boldsymbol{\rho}(0)\int_0^\infty exp\{Tx\}dt = -\boldsymbol{\rho}(0)T^{-1},$$

while the vector $\boldsymbol{\varphi}$ is given by [20]

$$\boldsymbol{\varphi} = \int_0^\infty exp\{Tt\}\Pi^{\text{MAP}}(t)dt\boldsymbol{e} = (T-Q)\boldsymbol{e}.$$

This determines $\boldsymbol{\alpha}_{\text{wait}}$, and the S_{wait} matrix is given by [20]

$$S_{\text{wait}} = \Delta^{-1}T'\Delta,$$

where $\Delta = \text{diag}(\boldsymbol{\alpha}_{\text{wait}})$.

C. The response time distribution

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

$$\boldsymbol{\alpha}_{\text{res}} = [\boldsymbol{\alpha}_{\text{wait}} \quad (1-\boldsymbol{\alpha}_{\text{wait}}\boldsymbol{e})\boldsymbol{\alpha}_{\text{service}}],$$

$$S_{\text{res}} = \begin{bmatrix} S_{\text{wait}} & (-S_{\text{wait}}\boldsymbol{e})\boldsymbol{\alpha}_{\text{service}} \\ \mathbf{0} & S_{\text{service}} \end{bmatrix}.$$

From this, we readily have the RT PDF, CDF, and moments. The average number of busy servers U can be computed as

$$U = \left(\sum_{k=1}^{c-1} k\boldsymbol{\pi}_k \circ \boldsymbol{u}_k + c\left(1 - \sum_{k=1}^{c-1}\boldsymbol{\pi}_k\boldsymbol{e}\right)\right)/c.$$

V. EXPERIMENTAL ASSESSMENT

In this section, we make use of the proposed model to compare systems with and without replication in terms of their reliability and offered RTs. To illustrate the impact of the arrival processes, we consider renewal processes with exponential (Exp) 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 $C_X^2 = \text{Var}[X]/E[X]^2$. While for Exp arrivals the SCV is one, for the HE_2 and MAP cases we set the SCV to 10. For MAP arrivals we also set the decay rate of the auto-correlation function to 0.9. We use the methods in [25], [26] to obtain the HE_2 and MAP representations. The service rate μ is set to 1, and the arrival and failure rates are in proportion to μ. In the figures, the legends N and R refer to systems without and with replication, respectively.

A. Reliability

The reliability, measured as the probability that a job completes service successfully, only depends on the service and failure rates, being given by $\mu/(\mu+\alpha)$ and $1-(\alpha/(\mu+\alpha))^2$ for the systems without and with replication, respectively. Figure 2 compares the system reliability under failure rates between 0.1 and 0.5, showing a significant improvement in reliability when introducing replication. For instance, even under a large failure rate of 0.5, the reliability improves from 66.67% to 88.89% thanks to the added replica, confirming the effectiveness of replication to improve reliability.

B. The effect of canceling

To evaluate the effectiveness of canceling, we compare the RTs obtained with the replication approach with canceling against those observed without canceling. The results without canceling are obtained with a simulation model, as this is not the focus of the model introduced in this paper. We consider an instance with HE_2 arrivals, 40 servers, failure rate $\alpha = 0.1$, and two different utilization levels for the system without replication: 0.20 and 0.45. Figure 3 depicts the RT complementary CDF (CCDF) of the system with canceling, labeled C, and without canceling, labeled NC. We observe that in the low-utilization case (0.2), the RT CCDFs of both replication approaches are very similar as the waiting time is negligible in both cases. In this case, the 0.2 utilization without replication, becomes 0.22 and 0.40 when introducing replication with and without canceling, respectively. However, when the utilization without replication is 0.45, the introduction of replication *without canceling* doubles the utilization to 0.90, leading to large long-tailed RTs. Instead, the utilization in the case *with canceling* only increases to 0.49, while the RT CCDF remains very close to the one under 0.2 utilization. This is caused by the high reliability (0.9), which enables the selection of the first replica that completes service. Clearly, replication with canceling is very effective in limiting the additional load introduced by the replicas, allowing it to provide better RTs, in addition to the expected gains in reliability.

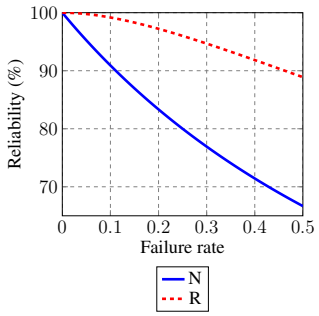


Fig. 2: Reliability

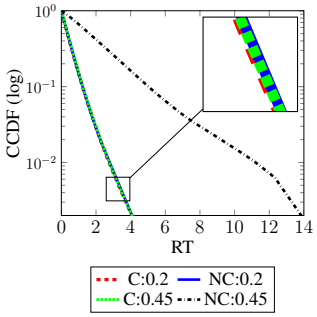


Fig. 3: The effect of canceling

C. Performance of replication with canceling

Another benefit of replication with canceling is its ability to exploit the concurrent execution of the replicas to select the first available result. We have observed that this effect can effectively reduce the RTs, as long as the utilization of the system without replication stays below a certain threshold. To study this behavior, we focus on the RT p-th percentile RT_p, which is the maximum RT faced by $p\%$ of the successful jobs. Figure 4 depicts RT_{95} as a function of the utilization in the no-replica case, under HE_2 arrivals, 40 servers, and failure rates 0.1 and 0.3. Focusing on the case with failure rate 0.1, we observe how RT_{95} is 40% smaller under replication, when the utilization is up to 40%. For larger utilization levels, the difference reduces and becomes zero at a *crossing point* at 0.78 utilization. Thus, below this value, the introduction of replication with canceling effectively reduces the RTs. Figure 4 also captures the effect of the failure rate on the RT distribution. Without replication, the system with *higher* failure rate shows a shorter RT_{95}, as with a lower reliability, only short jobs can complete service before a failure. However, the introduction of replication makes the RT_{95} under failure rate 0.1 to be shorter than under failure rate 0.3. This is caused by the ability of the replication strategy to obtain the result of the first replica to finish, but this effect weakens with a higher failure rate, as this reduces the probability that both

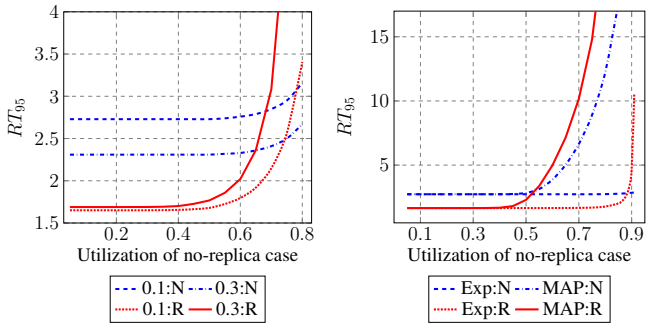


Fig. 4: The failure rate effect

Fig. 5: The IAT process effect

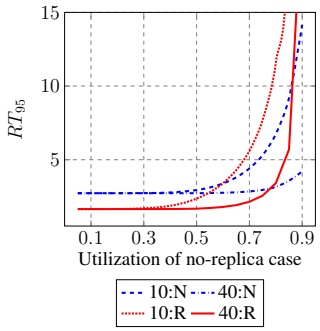


Fig. 6: The number of servers

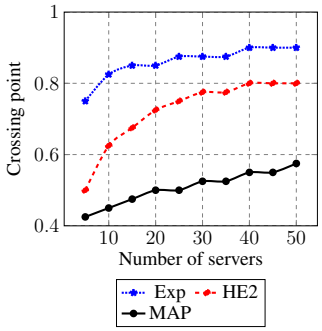


Fig. 7: Crossing point

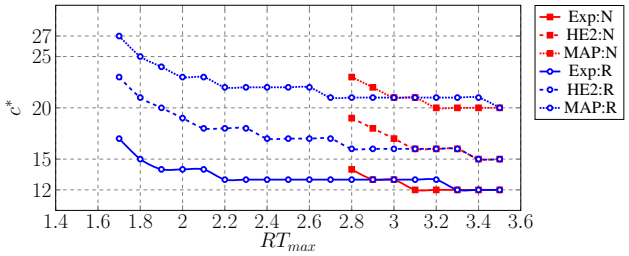


Fig. 8: Mininum number of servers

replicas are running until the first one completes. Further, the crossing point diminishes under the higher failure rate, limiting the conditions under which replication reduces the RTs.

Figure 5 compares the RT_{95} under Exp and MAP arrivals, keeping the same mean arrival rate. First, we observe how the bursty workload modeled by the MAP arrivals has a large effect on the RTs, although this effect only takes shape after a certain utilization threshold, in this case between 40% and 50%. Second, the system under MAP arrivals has a *smaller crossing point* than under Exp arrivals, and we observe how under MAP arrivals and replication, the RTs start to grow earlier than without replication. Recall that MAP arrivals show high variability and positive auto-correlation, with the high variability leading to a higher probability of having small IATs, and the auto-correlation leading to a high probability that short IATs come in bursts. Considering these characteristics is therefore key when deciding whether to replicate or not, as under MAP arrivals the room for replication is limited to scenarios with lower utilizations.

Figure 6 compares the RT_{95} under 10 and 40 servers, with HE_2 arrivals. The crossing point of the 10-server system is smaller than that of the 40-servers case, as more servers provide more flexibility to handle the additional load introduced with the replicas. We also observe how the RT_{95} is the same for the 10 and 40 server cases up to a certain utilization level, after which they start to diverge. This is because under low utilizations, RTs mostly correspond to service times, and as the utilization increases, queueing times become more important. We conclude this section with Figure 7, which shows the crossing points for Exp, HE_2, and MAP arrivals, with the number of servers between 5 to 50. This figure summarizes how the variability and auto-correlation of the arrival process affect the decisions on whether to replicate or not, and how additional servers provide more flexibility to deal with the load introduced by the replication scheme.

VI. SLO-driven Resource Provisioning

A key challenge in CC management is to adequately provision resources to meet SLOs on reliability, latency and throughput [27]. In this section, we present a model-based resource provisioning strategy, aimed at finding the minimum number of servers needed to meet an RT percentile SLO, and explore the maximum throughput achievable with a given configuration. We set the service rate μ and failure rate α to be 1.0 and 0.1, respectively, and consider the Exp, HE_2, and MAP arrival processes, as defined in Section V.

A. Resource provisioning

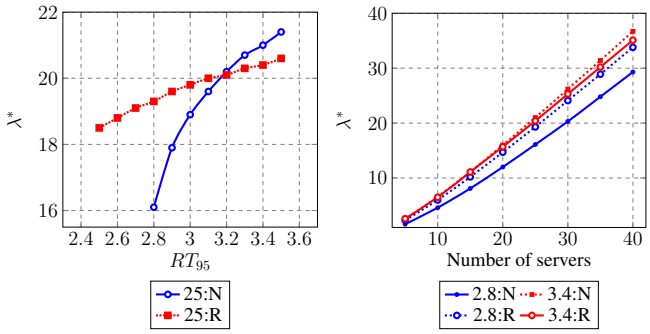
Under the precondition of meeting a reliability SLO, we aim at determining the minimum number of servers c needed to satisfy an RT SLO defined as RT_{max}. Using the RT 95-th, this problem can be formulated as:

$$\begin{aligned} c^* = \text{Min.} \quad & c, \\ \text{s.t.} \quad & c \in \{1, 2, \cdots, c_{max}\}, \\ & P(RT(c) < RT_{max}) = 0.95, \\ & \varrho(c) < 1, \end{aligned}$$

where c_{max} is the maximum number of servers available. $RT(c)$ is the RT obtained with c servers, and the constraint on $\varrho(c)$, defined in Equation (2), ensures stability. Both $RT(c)$ and $\varrho(c)$ are derived from the model introduced in Sections III and IV, and the optimization problem is solved with a binary search algorithm [28].

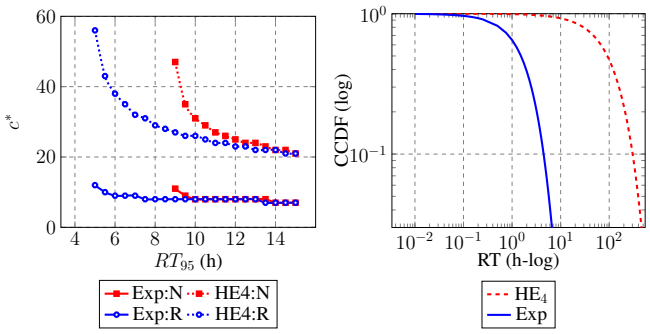
Figure 8 shows the minimum number of servers needed for systems with and without replication, given $c_{max} = 100$, for different RT_{max} objectives: $[1.5, 1.6, \cdots, 3.5]$. The missing results correspond to cases where the required RT_{max} cannot be achieved with the 100 servers available. The results show that tight RT_{max} objectives can only be achieved with replication. For instance, the system with replication can achieve an RT_{max} as low as 1.70, while the system without replication cannot achieve an RT_{max} below 2.80. Furthermore, the system with replication requires less servers than the system without replication to achieve tighter RT_{max} objectives. However, a looser RT_{max}, e.g., 3.3, under MAP arrivals, can be achieved with less servers without replication. It is worth noting that, for the same RT_{max} objective, the system under non-exponential arrivals requires many more servers to cope with the variability and correlation in the workload.

Relying on this model-based resource provisioning strategy, one can decide whether or not to replicate by considering both reliability and RT SLOs. For instance, the systems without



(a) The effect of RT_{95} (b) The effect of server number

Fig. 9: Maximum throughput



(a) Minimum number of servers (b) RT CCDF

Fig. 11: The effect of the arrival process

TABLE V: KS test

IAT	p-value	KS
Exp	$8.05E{-}30$	0.8235
HE_2	0.4134	0.1253
HE_3	0.4506	0.1217
HE_4	0.4531	0.1215
HE_5	0.4532	0.1215
HE_6	0.4532	0.1215

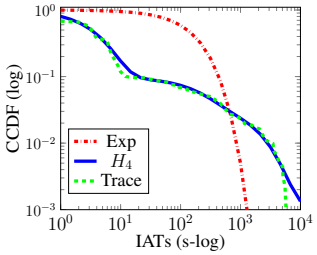


Fig. 10: IATs CCDF

and with replication considered in this section have a reliability of 90.91% and 99.17%, respectively. If the reliability SLO is 95%, then replication should always be adopted. Instead, if this SLO is only 90% and the workload is modeled as a MAP, replication should be adopted if the RT SLO is up to 3.1. Otherwise, no replication should be adopted.

B. Maximum achievable throughput

The system throughput is the average number of requests served per time unit, and mainly depends on the resources available. However, we want to explore how the throughput is affected by the introduction of replication with canceling. To do so, we setup the following optimization program, where we find the maximum throughput achievable (λ^*) while maintaining stability and complying with the RT SLO. The problem can be formulated as:

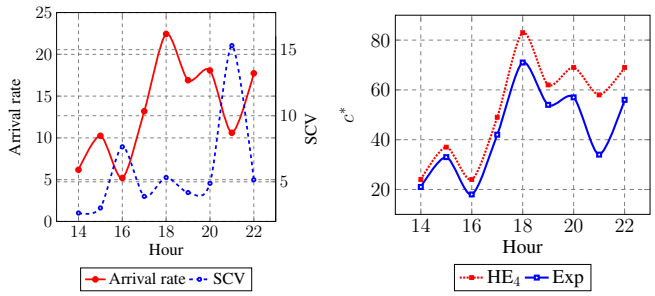
$$\begin{aligned} \lambda^* = \text{Max.} \quad & \lambda, \\ \text{s.t.} \quad & P(RT(\lambda) < RT_{max}) = 0.95, \\ & \varrho(\lambda) < 1. \end{aligned}$$

Figure 9(a) shows λ^* for a system with HE_2 arrivals, 25 servers and RT_{max} between 2.5 and 3.5. Clearly, the system with replication can achieve a much higher throughout under tight RT_{max}, in this case below 3.2. However, under looser RT_{max} objectives, the system without replication performs better. This trend can also be observed in Figure 9(b), where we consider only two values for RT_{max}, 2.8 and 3.4. Here we also observe that the difference in λ^*, between the systems with and without replication, amplifies as the number of servers increases.

C. Case Study: RICC Log

We now evaluate the proposed fault-tolerance approach and the model-based resource provisioning strategy on realistic traffic patterns. To this end, we make use of the RICC (RIKEN Integrated Cluster of Clusters) log, available on the Parallel Workloads Archive [15]. This log contains a total of 447,794 records of jobs submitted to the RICC installation in Japan from May to September 2010. We parameterize our model with jobs comprised of a single task, estimating a request service rate μ of 0.3245 per hour and a failure probability of 3.63%, based on which we estimate a failure rate α of 0.0122 per hour. As job arrivals show a strong daily and hourly cycle, we divide the arrival data into sets according to the day of the week and the hour of the day. Focusing on single-task jobs, we find that the IATs statistical characteristics vary significantly with time and day, with the highest SCV being 14.06 on Sunday between 12:00 and 13:00, and the lowest being 1.88 on Tuesday from 11:00 to 12:00. In the following we use the arrival data with the highest SCV as an example. Given the high variability, we use the method in [29], as implemented in jPhase [30], to find a hyper-exponential distribution with r phases (HE_r). We test the goodness-of-fit of the fitted CDFs with the Kolmogorov-Smirnov test [31]. Table V shows the p-value and maximum absolute difference between the CDFs (KS) for Exp and various HE_r distributions. Clearly, the Exp assumption fails the KS test, while any of the HE_r passes it under typical significance levels. Further, the p-value increases and the KS statistic decreases as the number of phases increases, although the improvement is limited for 5 or more phases. We thus choose the HE_4 representation, and compare the CCDF of the real trace, and the fitted Exp and HE_4 in Figure 10. Here we observe how the HE_4 distribution captures the characteristics of the trace much better than the Exp, especially its long tail along a wide range of values.

To illustrate the impact of the arrival process variability on the system performance, we compute the minimum number of servers needed with Exp and HE_4 arrivals, given different RT_{95} objectives and a maximum of 100 servers, as in Section VI-A. Figure 11(a) shows how, in contrast with the results in Section V, the case with replication always requires less servers than the case without replication. This is caused by the higher reliability observed here, which increases the likelihood of a replica completing service before the failure of either, enabling the selection of the first available result. In addition, the system with HE_4 arrivals requires many more servers than with Exp arrivals to achieve the RT SLO, either with or without replication. If the Exp assumption was used to size the cluster, we may expect it to have poor performance. Figure 11(b) depicts the RT CCDFs obtained under HE_4 arrivals and Exp arrivals, when the cluster is provisioned assuming Exp arrivals,



(a) Arrival rate and SCV (b) Minimum number of servers

Fig. 12: Time dependent resource provisioning

and setting RT_{max} to 6 hours. Clearly, the number of servers computed assuming Exp IATs is far from sufficient under HE$_4$ arrivals, with the RT_{95} under HE$_4$ arrivals being 395 hours, which is two orders of magnitude larger than the RT_{max}. This highlights the importance of considering the IAT variability observed in the real data-trace when dimensioning the cluster.

To further emphasize the importance of the SCV in provisioning CC resources, we look at the RICC historical data on Tuesdays from 14:00 to 22:00, fitting both HE$_r$ and Exp arrivals for each hour. While Figure 12(a) depicts the observed rate and SCV, Figure 12(b) shows the required number of servers to achieve an RT_{max} of 10 hours. The effect of the arrival rate appears dominant as the trend of the number of servers is similar to the trend of the arrival rate for both IAT distributions. However, we also observe non-trivial interactions between the arrival rate and the SCV. For instance, from hour 20 to hour 21 the arrival rate decreases by 41% but the number of servers under HE$_r$ arrivals decreases only by around 16% while this number decreases by 40% under Exp arrivals. This is caused by the large increase in SCV, as it almost doubles. Another example is at hours 14 and 16, which have very different arrival rates and SCVs, but require the same number of servers, assuming HE$_r$ arrivals, to achieve the RT SLO.

VII. CONCLUSION

In this paper, we evaluate the potential benefits of concurrent replication with canceling, considering its effect on both reliability and RTs. This is done with an analytic model that handles general arrival processes, and obtains the distribution of the response times instead of just its mean. The model enables us to determine when replication should be implemented, considering the job failure rate, the number of CC servers, the utilization of the system without replication, and the variability and auto-correlation of the arrival process. The model also serves as a basis for an SLO-aware resource provisioning strategy, which determines the minimum number of CC servers necessary to achieve a certain RT SLO. This strategy can be generalized to consider the costs of running the CC servers, or renting them in case of a cloud-based deployment. Further, based on measured and forecast arrival patterns, the model can serve as the basis of an online system that determines whether to adopt replication and adapts the CC size considering SLOs.

REFERENCES

[1] G. Kandaswamy, A. Mandal, and D. A. Reed, "Fault tolerance and recovery of scientific workflows on computational grids," in *IEEE CCGRID*, 2008, pp. 777–782.

[2] 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, no. 1, pp. 39–50, 2003.

[3] M. T. Özsu and P. Valduriez, "Distributed and parallel database systems," *ACM CSUR*, vol. 28, no. 1, pp. 125–128, 1996.

[4] K. Ferreira, J. Stearley, J. H. Laros III, R. Oldfield, K. Pedretti, R. Brightwell, R. Riesen, P. G. Bridges, and D. Arnold, "Evaluating the viability of process replication reliability for exascale systems," in *ACM SC*, 2011, p. 44.

[5] G. Levitin, L. Xing, and Y. Dai, "Optimal sequencing of warm standby elements," *CIE*, vol. 65, no. 4, pp. 570–576, 2013.

[6] A. Vulimiri, O. Michel, P. Godfrey, and S. Shenker, "More is less: reducing latency via redundancy," in *ACM HotNets*, 2012, pp. 13–18.

[7] J. Brutlag, "Speed matters for Google web search," *Google. June*, 2009.

[8] R. A. Oldfield, S. Arunagiri, P. J. Teller, S. Seelam, M. R. Varela, R. Riesen, and P. C. Roth, "Modeling the impact of checkpoints on next-generation systems," in *IEEE MSST*, 2007, pp. 30–46.

[9] J. Dean and L. A. Barroso, "The tail at scale," *CACM*, vol. 56, no. 2, pp. 74–80, 2013.

[10] G. Ananthanarayanan, A. Ghodsi, S. Shenker, and I. Stoica, "Why let resources idle? aggressive cloning of jobs with Dolly," *Memory*, vol. 40, no. 60, p. 80, 2012.

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

[12] B. Snyder, "Server virtualization has stalled, despite the hype," *InfoWorld*, 2010.

[13] P. G. Harrison and Z. Qiu, "Performance enhancement by means of task replication," in *EPEW*, 2013, pp. 191–205.

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

[15] "Parallel workloads archive." [Online]. Available: http://www.cs.huji.ac.il/labs/parallel/workload/

[16] Z. Zheng and Z. Lan, "Reliability-aware scalability models for high performance computing," in *IEEE CLUSTER*, 2009, pp. 1–9.

[17] M. Wu, X.-H. Sun, and H. Jin, "Performance under failures of high-end computing," in *ACM/IEEE SC*, 2007, p. 48.

[18] S. Amari and G. Dill, "Redundancy optimization problem with warm-standby redundancy," in *IEEE RAMS*, 2010, pp. 1–6.

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

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

[21] D. Bini, G. Latouche, and B. Meini, *Numerical Methods for Structured Markov Chains*. Oxford University Press, 2005.

[22] B. Sengupta, "Markov processes whose steady state distribution is matrix-exponential with an application to the GI/PH/1 queue," *AAP*, vol. 21, no. 1, pp. 159–180, 1989.

[23] G. Golub, S. Nash, and C. Van Loan, "A Hessenberg-Schur method for the problem AX+ XB= C," *IEEE TACON*, vol. 24, no. 6, 1979.

[24] R. A. Horn, "The Hadamard product," in *Proc. Symp. Appl. Math*, vol. 40, 1990, pp. 87–169.

[25] W. Whitt, "Approximating a point process by a renewal process, I: Two basic methods," *Operations Research*, vol. 30, no. 1, pp. 125–147, 1982.

[26] A. Heindl, G. Horváth, and K. Gross, "Explicit inverse characterizations of acyclic MAPs of second order," in *EPEW*, 2006, pp. 108–122.

[27] Y. Chen, S. Iyer, X. Liu, D. Milojicic, and A. Sahai, "SLA decomposition: Translating service level objectives to system level thresholds," in *IEEE ICAC*, 2007, pp. 3–3.

[28] C. E. Leiserson, R. L. Rivest, C. Stein, and T. H. Cormen, *Introduction to algorithms*. MIT press, 2001.

[29] R. El Abdouni Khayari, R. Sadre, and B. R. Haverkort, "Fitting world-wide web request traces with the EM-algorithm," *Performance Evaluation*, vol. 52, no. 2, pp. 175–191, 2003.

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

[31] F. J. Massey Jr, "The Kolmogorov-Smirnov test for goodness of fit," *JASA*, vol. 46, no. 253, pp. 68–78, 1951.