Tackling Latency via Replication in Distributed Systems

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

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

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

ABSTRACT

Consistently high reliability and low latency are twin requirements common to many forms of distributed processing; for example, server farms and mirrored storage access. To address them, we consider replication of requests with canceling – i.e. initiate multiple concurrent replicas of a request and use the first successful result returned, canceling all outstanding replicas. This scheme has been studied recently, but mostly for systems with a single central queue, while server farms exploit distributed resources for scalability and robustness. We develop an approximate stochastic model to determine the response-time distribution in a system with distributed queues, and compare its performance against its centralized counterpart. Validation against simulation indicates that our model is accurate for not only the mean response time but also its percentiles, which are particularly relevant for deadline-driven applications. Further, we show that in the distributed set-up, replication with canceling has the potential to reduce response times, even at relatively high utilization. We also find that it offers response times close to those of the centralized system, especially at medium-to-high request reliability. These findings support the use of replication with canceling as an effective mechanism for both fault- and delay-tolerance.

CCS Concepts

•Mathematics of computing → Markov processes; •Computer systems organization → Reliability; Redundancy;

Keywords

Latency-tolerance; Fault-tolerance; Matrix-analytic methods; Response time distribution; Distributed system

1. INTRODUCTION

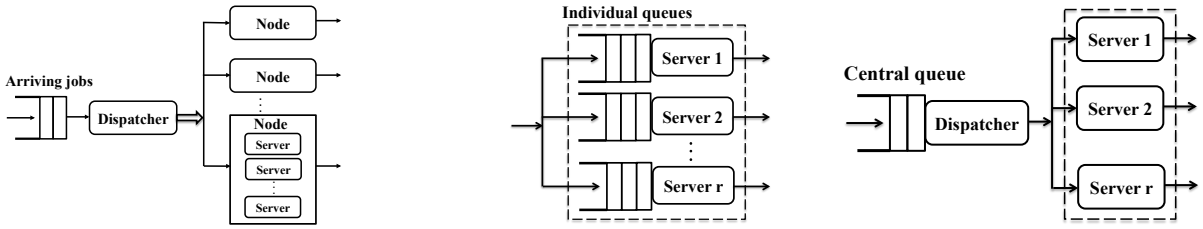
Server farms have been widely deployed, fueled by the ever-growing demand for computation-intensive and massive-data operations, to provide cost-effective and high-performance services for organizations such as Amazon, Google, IBM, Microsoft or Yahoo, by exploiting large collections of inexpensive resources [12]. Although much effort has been spent on optimizing the performance of server farms [4, 6, 12], requests unavoidably experience failures or delays, degrading the offered quality of service. Application-level failures, which are the focus of this paper, can arise for reasons such as communication errors [13], timeouts of resources with constrained availability, or outputs exceeding latency requirements. Further, low latency, especially keeping the tail of the latency distribution short, can be difficult to achieve in the face of contention for shared resources, queueing, or hardware problems [3]. For instance, experiments at Google show that a system where each request typically responds in 1ms, has a 99^{th} percentile latency of 10ms [3].

In this context, request replication with canceling has been proposed as a powerful mechanism to improve reliability, and to limit the response time, by initiating multiple copies of a request on separate servers and using the result from the copy that completes first [3, 16, 15]. To limit the additional load introduced by replicas, upon the successful completion of any replica, all other outstanding replicas are canceled immediately. This is achieved by allowing servers to share updates on the status of their replicas. Three key points make this approach viable. First, most clusters today are highly underutilized, with the average utilization of major data center servers being around 18% [21]; Second, much of the energy consumption is wasted at low utilization, e.g., even an idle server consumes about 65% of the power of its peak consumption [5]. Thus it is cost-effective to use these idling resources for running extra replicas of requests. Third, concurrent replication can handle unpredictable failures, as it is sufficient that one of the replicas succeeds. Further, replication has the potential to reduce both the mean and the tail of the response-time distribution, since the overall latency becomes the minimum of the delays across all the replicas [16, 15].

Although replication has been studied recently, most works focus on systems with a single central queue, while server farms exploit distributed resources for scalability and robustness. Clearly, the centralized set-up offers better performance, but the distributed set-up provides more flexibility, and in some systems holding a central queue is not possible, such as when accessing mirrored storage systems in parallel. To better understand the performance of these two settings, in this paper we develop an approximate stochastic model to determine the response-time distribution in a system with

ICPE'16, March 12-18, 2016, Delft, Netherlands

DOI: http://dx.doi.org/10.1145/2851553.2851562



(a) Server farm (b) The distributed set-up (c) The centralized set-up

Figure 1: Reference models. A server farm is split in nodes, as in (a). The set-up of each node can be as in (b) or in (c).

distributed queues, and compare its performance against its centralized counterpart. Although approximate, validation results indicate that our model is accurate for not only the mean response time but also its percentiles, a significant advantage since mean response time guarantees are not sufficient in many situations, in particular for deadline-driven applications. Further, we show that in the distributed set-up replication with canceling has the potential to reduce the response times, even under relatively high utilization. The effect of replication actually depends on the specific response time percentile evaluated, especially in low-reliability scenarios. This effect is more uniform across the response-time distribution when the request reliability is high or when more than one extra replica is adopted. We also find that the distributed set-up offers response times close to those of the centralized system, especially under a medium to high request reliability. These findings support the use of replication with canceling as an effective mechanism for both fault- and delay-tolerance in server farms.

2. RELATED WORK

Concurrent replication has been considered recently as a means to reduce latency in interactive and deadline-driven applications [1, 3, 7, 16, 15, 10, 17, 20, 22]. For instance, [1] observes in trace-driven experiments that processing replicas concurrently is effective in mitigating the effect of latency. [17] explores concurrent replication with canceling as a tail-tolerant approach, and shows that this approach can be powerful in keeping the response time tail short. In particular, [7, 15, 16] evaluate the effectiveness of this approach in improving the reliability for systems with request failures. These works focus on the case with *one extra replica* for systems made up of two distributed queues [7], synchronous parallel processors [15], and for computing clusters with a central queue [16]. Among them, [16] determines the response-time distribution, while [7, 15] focus on the mean response time only. In this paper, we obtain the response-time *distribution* for systems with *both* distributed and centralized queues, and implementing *multiple* extra replicas.

3. BACKGROUND

3.1 Reference model

We consider a server farm consisting of a number of distributed, homogeneous, and independent servers, which process incoming requests with rate μ. Requests in service are subject to failures, with failure rate α. Both service and failure times are exponentially distributed, a common assumption in reliability engineering [24, 23]. In case of a failure, the request currently in service is lost, but the server itself is not affected, and continues to serve the next request that enters. To improve the reliability, $r-1$ extra replicas are adopted for each arriving request, i.e., a total of r replicas of a request are submitted to the system. In particular, $r=1$ represents the case where no replication is adopted. We denote by *request/replica* the original request or any of its replicas, and by *job* the set of replicas for a single request, where the number of replicas r is also referred to as the *replication level*. To reduce unnecessary workload, the system replies with the result from whichever replica completes successfully first, and immediately cancels all the other outstanding replicas in the same job. The canceling overhead, which is the time to remove all the replicas in a job, is assumed to be negligible.

To take advantage of replication, we consider a server farm consisting of n distributed computing nodes, and a central scheduler that assigns jobs to the processing nodes in a round-robin or random fashion, as shown in Figure 1(a). Each node is composed of r independent and statistically identical processing servers, thus serving a job of replication level r, by processing each of the r replicas in each server. The node can have distributed queues in front of each of the servers, as shown in Figure 1(b), where each of the r replicas joins the end of the queue at one of the r servers, and is processed with first-come first-served (FCFS) scheduling. An alternative setup is to have a centralized queue in front of all the servers, as shown in Figure 1(c), where all the replicas of the incoming requests form a single queue in the order of arrival and join the next server that becomes available with FCFS scheduling. For both models, when one of the replicas in a job completes service successfully, it immediately cancels all its outstanding siblings, *either waiting or in service*. However, if a replica fails during service, it leaves the system without influencing its siblings. Analyzing the model with distributed queues is far more challenging than its counterpart with a single centralized queue, as the synchronized arrivals of replicas to all queues correlates their dynamics, while individual replicas fail asynchronously. We therefore develop an approximated model in Section 4 to cope with this setup, capturing the dynamics introduced by failures, replication and canceling. To analyze the performance of a system with a central queue, we extend our previous work [16], where we considered the $r=2$ case, to handle any number of replicas, i.e., $r\geq 2$, as described in Section 6.

3.2 Preliminaries

Motivated by the high variability and auto-correlation observed in inter-arrival times in computer systems, we utilize Markovian arrival processes (MAP) to represent inter-arrival times [14]. The continuous-time MAP [11] is a marked Markov chain (MC) with generator matrix $D = D_0 + D_1$,

where matrices D_0 and D_1 hold the rates associated to transitions without and with arrivals. 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), where m_a is the number of states in the underlying MC, or arrival phases. The mean arrival rate is $\lambda=\boldsymbol{d}D_1\mathbf{1}$, where $\mathbf{1}$ is a column vector of ones, and $\boldsymbol{d}$ is the stationary distribution of the underlying MC, i.e. $\boldsymbol{d}D=0$ and $\boldsymbol{d}\mathbf{1}=1$. In the special case of Poisson arrivals, $D_0=-\lambda$ and $D_1=\lambda$.

In the next sections we show that the *job* processing time follows a Phase-type (PH) distribution, whose parameters depend on the overall system state. A PH random variable X represents the absorption time in an MC with $n+1$ states, where the states $\{1,\ldots,n\}$ are transient and state 0 is absorbing [11]. This random variable or its distribution are denoted as PH$(\boldsymbol{\tau}, S)$, where the $1\times n$ vector $\boldsymbol{\tau}$ is the MC initial probability distribution for the transient states, and matrix S is the $n\times n$ sub-generator matrix holding the transition rates among the transient states. The vector $\boldsymbol{S}^*=-S\mathbf{1}$ holds the absorption rates from the transient states. Its cumulative distribution function (CDF) is $F(x)=1-\boldsymbol{\tau}\exp(Sx)\mathbf{1}$, for $x\geq 0$, and its expected value is $E[X]=-\boldsymbol{\tau}S^{-1}\mathbf{1}$.

4. THE DISTRIBUTED SETUP

In this section we introduce a stochastic model to determine the job response-time distribution offered by a single computing node that implements replication with canceling and operates with r individual queues, as in Figure 1(b). To this end we start by obtaining the waiting-time and service-time distributions separately, both of which have PH representations. For each job, composed of r replicas, the waiting time is the period between its arrival and the time the first of its replicas starts service, while the service time starts when the waiting period ends and concludes when one of the replicas completes service successfully, or when all replicas fail. In the following we refer to a period during which the server of the shortest queue is busy as an *all-busy* period, which terminates when the one server becomes idle and the queue is empty. This marks the start of a *not-all-busy* period, where at least one server is idle and terminates when a job arrives, submitting one replica to each server, initiating a not-all-busy period.

4.1 The waiting-time distribution

To determine the waiting-time distribution, we observe the queues only during the *all-busy* periods and define an *age process*, following [2, 19]. Different from [2, 19], which consider queues with service times independent of the system state, the replication and canceling mechanism introduces dependencies among the servers that cannot be analyzed by existing models. We thus define a bivariate Markov process $\{X(t), J(t)|t\geq 0\}$, where the *age* $X(t)$ is the total time-in-system of the youngest job in service at time t. The age $X(t)$ thus takes values in $[0,\infty)$, increasing linearly with rate 1 as long as no new jobs start service. Note that, during the all-busy period, a new job starts service only if the replica in service in the *shortest queue* completes service or fails. This is because for the replicas waiting in other queues, one or more of their siblings have already failed. Thus in case of a service completion or a failure in the shortest queue, a new job starts service and its age will be equal to its waiting time, thus triggering a downward jump in $X(t)$. The *phase* $J(t)=(A(t), D(t))$ holds the joint state of the arrival process $A(t)$ and the service process $D(t)$. The arrival process is a MAP with m_a phases and parameters (D_0, D_1) as defined in Section 3.

To model the service process, we first order the queue lengths in ascending order, i.e., $(q_1, q_2, \ldots, q_r)$ with $q_i\leq q_j$ for $i<j$ and $1\leq i, j\leq r$, where the queue length includes jobs waiting and in service. During the all-busy period, the shortest queue length must be positive, i.e., $q_1>0$, while during the not-all-busy period the shortest queue must be empty, i.e., $q_1=0$. Further, we focus on the differences between two consecutive queues after ordering, defined as $(d_1,\ldots,d_{r-1})$, where $d_i=q_{i+1}-q_i$ for $1\leq i\leq r-1$. Notice that this model is closely related to the fork-join model in [18], where the difference in queue-lengths are also used to model the evolution of a set of queues. However, here we consider the replication with canceling mechanism, which displays different dynamics from the fork-join queue, and the replicas are allowed to fail, a feature not considered for the fork-join queue. Also, [18] relies on the queue-length differences with respect to the shortest queue, while here we focus on the differences between two consecutive queues after ordering. The queue-length difference is unbounded in principle, but, to keep the phase space finite, we introduce an upper bound $C<\infty$, such that the difference is at most C. As a result, the service process $D(t)$ takes values in the set $S_D = \{(d_1,\ldots,d_{r-1})|d_i \in \{1,\ldots,C\}$ for $1\leq i\leq r-1\}$, the cardinality of which is $m_s=(C+1)^{r-1}$ for a system with r queues and an upper bound of C. The phase process $J(t)$ thus takes $m=m_a m_s$ different values, where m_a is the number of arrival phases. The limit C introduces an approximation, the goodness of which depends on the system parameters. For instance, when the failure rate α is small compared to the service rate μ, the probability of a large difference d_i is small, since successful service completions are more likely to occur, activating the canceling mechanism, which keeps the queues more synchronized. With a larger failure rate, we may expect a larger difference. However, due to the canceling mechanism, the probability of large differences between queue lengths stays small, even at high loads. Section 5 evaluates the accuracy of this approximation and explores the selection of the limit C.

To determine the PH representation $(\boldsymbol{s}_{\text{wait}}, S_{\text{wait}})$ of the waiting-time distribution, we rely on the stationary distribution $\boldsymbol{\pi}(x)$ of the $(X(t), J(t))$ process, which has a matrix exponential representation [19] $\boldsymbol{\pi}(x)=\boldsymbol{\pi}(0)\exp(Tx)$, for $x>0$. The $m\times m$ matrix T satisfies the non-linear integral equation

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

where $S^{(\text{MAP})}=S\otimes I_{m_a}$, $A^{(\text{MAP})}(u)=A^{(\text{jump})}\otimes\exp(D_0 u)D_1$, while I_n is the identity matrix of size n, and $\otimes$ denotes the Kronecker product. Here $S+A^{(\text{jump})}$ is the generator of the marginal service phase process, where S and $A^{(\text{jump})}$ are $m_s\times m_s$ matrices that hold the transition rates of the service process associated to transitions *without* and *with* the start of a new job service, respectively. As mentioned above, only the service completion or failure of the replica in service in the shortest queue triggers the start of a new *job* service, thus transitions in the shortest queue correspond to matrix $A^{(\text{jump})}$, while transitions in other queues correspond to matrix S.

Table 1 shows the transition rates in S and $A^{(\text{jump})}$. The first row considers the case where the replica in service in the

Table 1: Transition rates for matrices S and $A^{(\text{jump})}$

Matrix	From	To	Rate	Range
S	$(d_1,\dots,d_i,d_{i+1},\dots,d_{r-1})$	$(d_1,\dots,d_i{-}1,d_{i+1},\dots,d_{r-1})$	μ	$d_i>0,\ d_{i+1}>0, i\geq 1$
	$(d_1,\dots,d_i,d_{i+1},\dots,d_{r-1})$	$(d_1,\dots,d_i{-}1,\max\{C,d_{i+1}{+}1\},\dots,d_{r-1})$	α	$d_i>0,\ d_{i+1}>0, i\geq 1$
	$(d_1,\dots,d_i,0,\dots,0,d_k,\dots,d_{r-1})$	$(d_1,\dots,d_i{-}1,0,\dots,0,d_k,\dots,d_{r-1})$	$(k{-}i)\mu$	$d_i>0,\ d_j=0\quad \forall i<j<k$
	$(d_1,\dots,d_i,0,\dots,0,d_k,\dots,d_{r-1})$	$(d_1,\dots,d_i{-}1,1,0,\dots,0,d_k,\dots,d_{r-1})$	$(k{-}i)\alpha$	$d_i>0,\ d_j=0\quad \forall i<j<k$
$A^{(\text{jump})}$	$(d_1,d_2,\dots,d_{r-1})$	$(d_1,d_2,\dots,d_{r-1})$	μ	$d_1>0$
	$(d_1,d_2,\dots,d_{r-1})$	$(\max\{C,d_1{+}1\},d_2,\dots,d_{r-1})$	α	$d_1>0$
	$(0,\dots,0,d_i,\dots,d_{r-1})$	$(0,\dots,0,d_i,\dots,d_{r-1})$	$i\mu$	$d_i>0,\ d_j=0\quad \forall j<i$
	$(0,0,\dots,0,d_i,\dots,d_{r-1})$	$(1,0,\dots,0,d_i,\dots,d_{r-1})$	$i\alpha$	$d_i>0,\ d_j=0\quad \forall j<i$

$(i{+}1)^{\text{th}}$ shortest queue *completes service*, canceling its partners in all the queues longer than queue $(i{+}1)^{\text{th}}$, resulting in the difference between the $(i{+}1)^{\text{th}}$ and i^{th} shortest queues decreasing by 1, while other differences remain unaffected. The second row considers the case where the replica in the $(i{+}1)^{\text{th}}$ shortest queue *fails*, decreasing by 1 the difference between the $(i{+}1)^{\text{th}}$ and i^{th} shortest queues, while the difference between the $(i{+}2)^{\text{th}}$ and $(i{+}1)^{\text{th}}$ shortest queues increases by 1, but bounded by C. In the previous two cases we assumed that $d_{i+1}{>}0$, thus ensuring that the $(i{+}1)^{\text{th}}$ shortest queue is actually a single queue. The third and fourth rows consider the case where $d_{i+1}{=}\cdots{=}d_{k-1}=0$, such that from the $(i{+}1)^{\text{th}}$ to the k^{th} shortest queues have the same length. Thus, the transition rates in the third row reflect that a service completion in any of these queues triggers the same transition. The fourth row considers the same condition for the case of replica failures. The second block in Table 1 considers similar conditions but for the matrix $A^{(\text{jump})}$. The first and second rows consider a service completion or failure of the replica in the shortest queue, thus initiating a new job service. In case of a service completion, the replica that completes service cancels all its partners, thus the differences between queue-lengths remain unchanged. On the other hand, a failure in the shortest queue leads d_1 to increase by 1, but bounded by the limit C. The last two rows consider the case where there are multiple shortest queues.

The matrix T can be found by iteratively solving Eq. (1), where each iteration involves the solution of a Sylvester matrix equation [8]. Once T has been found, we need to determine the steady state distribution $\boldsymbol{\pi}(0)$ of the phases at the beginning of an all-busy period. To find $\boldsymbol{\pi}(0)$, we need to connect the not-all-busy and the all-busy periods [2]. Compared to the all-busy period, in the not-all-busy period the shortest queue is empty, $q_1{=}0$, but the differences between queue-lengths can be modeled just as in the all-busy period. Thus, during the not-all-busy period we keep track of the arrival and services phases $J(t){=}(A(t),D(t))$, with the service phase $D(t)$ taking values in the set S_D. We can thus follow [18] to find the stationary distribution $\boldsymbol{\pi}(0)$ that solves

$$\boldsymbol{\pi}(0){=}\boldsymbol{\pi}(0)\int_0^\infty \exp(Tu)(A^{(\text{jump})}\otimes\exp(D_0u))du \\ (S_{\text{not-all}}\oplus D_0)^{-1}(I_{m_s}\otimes D_1), \tag{2}$$

where the matrix $S_{\text{not-all}}$ holds all the service transition rates *between arrivals* during a not-all-busy period. Since no arrivals are allowed and the queue is empty, there are no new jobs starting service during this period and the $S_{\text{not-all}}$ matrix holds the same transition rates as the matrix S. The only difference is in the diagonal of $S_{\text{not-all}}$, which needs to be such that $S_{\text{not-all}}\mathbf{1}=\mathbf{0}$.

Let the steady state distribution of the phase during the busy period be $\boldsymbol{\pi}_{\text{busy}}{=}{-}\boldsymbol{\pi}(0)T^{-1}$, and define $\boldsymbol{\varphi}{=}(T{-}S^{(\text{MAP})})\mathbf{1}$ [2]. The PH representation of the waiting time is given by

$$\boldsymbol{s}_{\text{wait}}=\gamma\boldsymbol{\pi}_{\text{busy}}\circ\boldsymbol{\varphi}/((\boldsymbol{\pi}_{\text{busy}}\circ\boldsymbol{\varphi})\mathbf{1}),\quad S_{\text{wait}}=\Delta^{-1}T'\Delta, \tag{3}$$

where $\Delta{=}\text{diag}(\boldsymbol{\pi}_{\text{busy}})$, and $\circ$ stands for the Hadamard product. The parameter γ is the probability that a job has to wait, and is given by $\gamma{=}(E[\eta_0]{-}1)/(E[\eta_0]{-}1{+}E[\eta_1])$, where $E[\eta_0]$ and $E[\eta_1]$ are the expected number of arrivals during an all-busy period and a not-all-busy period, respectively. Since the job that initiates the not-all-busy period does not have to wait, $E[\eta_0]{-}1$ is the expected number of arrivals that have to wait in a cycle of an all-busy period followed by a not-all-busy period. Further, $E[\eta_1]{=}1$ as an arrival during the not-all-busy period sends replicas to all queues in the node, initiating an all-busy period. Thus $\gamma{=}1{-}1/E[\eta_0]$, where $E[\eta_0]$ can be obtained as in [2, Section 6].

4.2 The service-time distribution

We now determine the job service-time distribution, which we show to be PH with parameters $(\boldsymbol{s}_{\text{ser}},S_{\text{ser}})$ that depend on the overall system state. Let $Y(t)$ be the service state of a *tagged job* in service at time t. We define $Y(t){=}(R(t),D(t))$, where $R(t)$ records the number of alive replicas of the tagged job at time t, thus $R(t)\in\{1,\dots,r\}$. The variable $D(t)$ is again the difference between queue lengths, but it focuses on the queue lengths *in front of the tagged replicas only*, ignoring any jobs that arrive after the tagged job. Thus if a tagged replica has already failed, the corresponding queue length is kept at 0. Notice that since we are interested in the service time of a tagged job, we keep track of its service phase from the moment the first tagged replica starts service, and we order the service states according to $R(t)$ in descending order. Further, when $R(t){=}1$, only one tagged replica remains alive, thus it is enough to keep track of the length of the queue where this replica is located. To build the PH representation of the service-time distribution we consider the sub-generator $\bar{S}_{\text{ser}}$, and two absorbing states, S and F, representing the cases where the job completes service successfully or encounters a failure, respectively. We can then write the generator of the service-time process (ignoring the zero rows corresponding to the absorbing states) as

$$\begin{bmatrix}\bar{S}_{\text{ser}} & \bar{\boldsymbol{S}}_S^* & \bar{\boldsymbol{S}}_F^*\end{bmatrix},$$

where the absorption vectors $\bar{\boldsymbol{S}}_S^*$ and $\bar{\boldsymbol{S}}_F^*$ hold the absorption rates into states S and F, respectively, and $\bar{\boldsymbol{S}}^*{=}\bar{\boldsymbol{S}}_S^*{+}\bar{\boldsymbol{S}}_F^*$.

The transitions among service phases depend not only on whether there is a successful service completion or a failure, but also on which queue this happens. The transition rates of the service process are shown in Table 2, which we

Table 2: Transition rates for $\bar{S}_{\rm ser}$, $\bar{\boldsymbol{S}}^*_S$ and $\bar{\boldsymbol{S}}^*_F$

From	To	Rate	Range
$m,(d_1,\ldots,d_{r-1})$	$m{-}1,(\max\{C,d_1{+}1\},\ldots,d_{r-1})$	α	$m=r$
$m,(d_1,\ldots,d_{r-1})$	S	μ	$m=r$
$m,(0,\ldots,0,d_i,\ldots,d_{r-1})$	$m{-}1,(1,0,\ldots,0,d_i,\ldots,d_{r-1})$	$i\alpha$	$m{=}r, d_i{>}0, d_j{=}0 \quad \forall j{<}i$
$m,(0,\ldots,0,d_i,\ldots,d_{r-1})$	S	$i\mu$	$m{=}r, d_i{>}0, d_j{=}0 \quad \forall j{<}i$
$m,(\ldots,d_i,d_{i+1},\ldots,d_{r-1})$	$m,(\ldots,d_i{-}1,d_{i+1},\ldots,d_{r-1})$	μ	$d_i{>}0,\ d_{i+1}{>}0,\ m{\leq}r$
$m,(\ldots,d_i,d_{i+1},\ldots,d_{r-1})$	$m,(\ldots,d_i{-}1,\max\{C,d_{i+1}{+}1\},\ldots,d_{r-1})$	α	$d_i{>}0,\ d_{i+1}{>}0,\ m{\leq}r$
$m,(\ldots,d_i,0,\ldots,0,d_k,\ldots,d_{r-1})$	$m,(\ldots,d_i{-}1,0,\ldots,0,d_k,\ldots,d_{r-1})$	$(k{-}i)\mu$	$d_i{>}0,\ d_j{=}0\ \forall i{<}j{<}k,\ m{\leq}r$
$m,(\ldots,d_i,0,\ldots,0,d_k,\ldots,d_{r-1})$	$m,(\ldots,d_i{-}1,1,0,\ldots,0,d_k,\ldots,d_{r-1})$	$(k{-}i)\alpha$	$d_i{>}0,\ d_j{=}0\ \forall i{<}j{<}k,\ m{\leq}r$
$m,(\mathbf{0}_{r-m-1},1,d_{r-m+1},\ldots,d_{r-1})$	$m{-}1,(\mathbf{0}_{r-m-1},0,\max\{C,d_{r-m+1}{+}1\}\ldots,d_{r-1})$	α	$d_{r-m+1}{>}0,\ 1<m<r$
$m,(\mathbf{0}_{r-m-1},1,d_{r-m+1},\ldots,d_{r-1})$	S	μ	$d_{r-m+1}{>}0,\ 1<m<r$
$m,(\mathbf{0}_{r-m-1},1,0,\ldots,0,d_k,\ldots,d_{r-1})$	$m{-}1,(\mathbf{0}_{r-m-1},0,1,0,\ldots,0,d_k,\ldots,d_{r-1})$	$(k{-}r{+}m{+}1)\alpha$	$1<m<r$
$m,(\mathbf{0}_{r-m-1},1,0,\ldots,0,d_k,\ldots,d_{r-1})$	S	$(k{-}r{+}m{+}1)\mu$	$1<m<r$
$1,(\mathbf{0}_{r-2},1)$	S	μ	$m=1$
$1,(\mathbf{0}_{r-2},1)$	F	α	$m=1$

split in four sets for clarity. The first set of rows considers the case where $R(t){=}r$, i.e., none of the tagged replicas has completed service or failed, thus there is at least one tagged replica in service in one of the shortest queues. In the first two rows there is only one tagged replica in service, thus its failure, with rate α, leads d_1 to increase by 1 but bounded to C, while its service completion, with rate μ, marks the service completion of the whole job, thus entering the S state. Rows 3-4 cover a similar case, but with i tagged replicas concurrently in service. In the second set, we consider transitions in queues other than the shortest ones, which do not hold tagged replicas. The first and second rows consider the case of a service completion or a failure in the $(i{+}1)^{\rm th}$ queue, respectively, assuming there is a single queue with this queue length ($d_{i+1}{>}0$). The third and fourth rows consider a similar transition, but in this case the $(i{+}1)^{\rm th}$ to the $k^{\rm th}$ queues have the same length, thus the transition rates are $(k{-}i)\mu$ and $(k{-}i)\alpha$, respectively. In the third set we assume that the number of tagged replicas alive is $1{<}m{<}r$, thus at least one tagged replica has already failed. In fact, the zero vector $\mathbf{0}_{r-m+1}$ in front of the state description corresponds to the queue-length differences among the $r{-}m$ servers where the tagged replicas have already failed. Here the first two rows correspond to the failure or service completion, respectively, of a tagged replica in the (only) shortest non-zero queue, thus assuming $d_{r-m+1} > 0$. The third and fourth rows cover the same case, but considering multiple (i.e., $k{-}r{+}m{-}1$) shortest queues. Finally, the last set considers the case where only one tagged replica remains in service, thus its service completion marks a successful completion of the whole job, while its failure leads to the job failure.

Having obtained $\bar{S}_{\rm ser}$, we define $S_{\rm ser} = \bar{S}_{\rm ser} \otimes I_{m_a}$, which is a matrix of size $m_{\rm ser}$. Since the service phase space is ordered in decreasing order according to $R(t)$, we let m_r be the number of phases where $R(t){=}r$ and m_0 the remaining phases, such that $m_{\rm ser}{=}m_r{+}m_0$. We can now determine the initial probability vector $\boldsymbol{s}_{\rm ser}$, which is the stationary probability with which a job starts service in each of the phases, and following [18, Proposition 1], it is given by

$$\boldsymbol{s}_{\rm ser} = [(1{-}\gamma)\boldsymbol{\pi}(0) + \gamma c \boldsymbol{\pi}_{\rm busy}(T{-}S^{(\rm MAP)}) \quad \mathbf{0}_{m_0}], \qquad (4)$$

where $c^{-1}{=}\boldsymbol{\pi}_{\rm busy}(T{-}S^{(\rm MAP)})\mathbf{1}$, and the zero vector correspond to phases with $R(t){<}r$ as a job can only start service in phases with $R(t){=}r$, since it starts with all its replicas alive. As a result, we have that $(\boldsymbol{s}_{\rm ser}, S_{\rm ser})$ is a PH representation of the service time for all jobs. For further reference we also define the corresponding exit vector $\boldsymbol{S}^*_S = \bar{\boldsymbol{S}}^*_S \otimes \mathbf{1}_{m_a}$. However, we want to focus on the successful jobs only, for which we obtain a PH representation of the service time in the following proposition, the proof of which is given in the Appendix.

Proposition 1. *The service time of* successful jobs *follows a PH distribution with parameters $(\boldsymbol{\beta}_{ser}, B_{ser})$, where*

$$\boldsymbol{\beta}_{ser} = \boldsymbol{S}^{*\prime}_S \Pi / p_S, \quad B_{ser} = \Pi^{-1} S'_{ser} \Pi.$$

Π is a diagonal matrix such that $\Pi\mathbf{1}{=}\eta'$, $\eta{=}{-}\boldsymbol{s}_{ser}S^{-1}_{ser}$ is the stationary distribution of the service phase, and $p_S = -\boldsymbol{s}_{ser}S^{-1}_{ser}\boldsymbol{S}^*_S$ is the probability that a job is successful.

Finally, we are in a position to obtain the response-time distribution in the following theorem, the proof of which is given in the Appendix.

Theorem 1. *The response time of successful jobs follows a PH distribution with parameters $(\boldsymbol{s}_{res}, S_{res})$, where*

$$\boldsymbol{s}_{res}{=}\begin{bmatrix}\boldsymbol{\beta}^{idle}_{ser} & \boldsymbol{\beta}^{busy}_{ser} & \mathbf{0}\end{bmatrix},\ S_{res}{=}\begin{bmatrix}B^{idle}_{ser} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & B^{busy}_{ser} & (-B^{busy}_{ser}\mathbf{1})P_{s,w}\\ \mathbf{0} & \mathbf{0} & S_{wait}\end{bmatrix}, \qquad (5)$$

where $(\boldsymbol{\beta}^{idle}_{ser}, B^{idle}_{ser})$ and $(\boldsymbol{\beta}^{busy}_{ser}, B^{busy}_{ser})$ are the PH representations of the service times of successful jobs that start service immediately upon arrival and that must wait, respectively. Letting Δ_{idle} be a diagonal matrix such that $\Delta_{idle}\mathbf{1}{=}{-}(1-\gamma)(\boldsymbol{\pi}(0)S^{-1}_{ser})'$, $(\boldsymbol{\beta}^{idle}_{ser}, B^{idle}_{ser})$ are given by

$$\boldsymbol{\beta}^{idle}_{ser}{=}\boldsymbol{S}^*_S\Delta_{idle}/p_S, \quad B^{idle}_{ser}{=}\Delta^{-1}_{idle}S'_{ser}\Delta_{idle}.$$

Similarly, letting Δ_{busy} be a diagonal matrix such that $\Delta_{busy}\mathbf{1}$ $={-}\gamma(\boldsymbol{\alpha}_{busy}S^{-1}_{ser})'$, $(\boldsymbol{\beta}^{busy}_{ser}, B^{busy}_{ser})$ are given by

$$\boldsymbol{\beta}^{busy}_{ser}{=}\boldsymbol{S}^*_S\Delta_{busy}/p_S, \quad B^{busy}_{ser}{=}\Delta^{-1}_{busy}S'_{ser}\Delta_{busy}.$$

Finally, $P_{s,w}$ is an $m_{ser} \times m$ matrix given by

$$P_{s,w} = \begin{bmatrix}\tilde{P}_{s,w}\\ 0_{m_0\times m}\end{bmatrix},$$

where $\tilde{P}_{s,w} = \Gamma^{-1}(T - S^{(MAP)})'\Lambda$, and Γ and Λ are diagonal matrices such that $\Gamma\mathbf{1} = (T - S^{(MAP)})'\Lambda\mathbf{1}$ and $\Lambda\mathbf{1} = \boldsymbol{\alpha}'_{busy}$. Here $\boldsymbol{\alpha}_{busy} = c\boldsymbol{\pi}_{busy}(T{-}S^{(MAP)})$ is the initial service phase of jobs that wait, and c is a normalizing constant such that $c^{-1}{=}\boldsymbol{\pi}_{busy}(T{-}S^{(MAP)})\mathbf{1}$.

Table 3: Approximation errors compared with simulation results

r	Arr	NR-load	Measure	Err(%) - NR-reliability:10%			Err(%) - NR-reliability:50%			Err(%) - NR-reliability:90%		
				C = 5	C = 10	C = 50	C = 5	C = 10	C = 50	C = 5	C = 10	C = 50
3	Poisson	0.1	mean	<1	<1	<1	<1	<1	<1	<1	<1	<1
			R_{95}	<1	<1	<1	<1	<1	<1	<1	<1	<1
		0.5	mean	<1	<1	<1	<1	<1	<1	<1	<1	<1
			R_{95}	<1	<1	<1	<1	<1	<1	<1	<1	<1
		0.9	mean	15.16	2.62	<1	<1	<1	<1	<1	<1	<1
			R_{95}	19.55	3.53	<1	<1	<1	<1	<1	<1	<1
	MAP	0.1	mean	<1	<1	<1	<1	<1	<1	<1	<1	<1
			R_{95}	<1	<1	<1	<1	<1	<1	<1	<1	<1
		0.5	mean	31.63	11.28	<1	<1	<1	<1	<1	<1	<1
			R_{95}	28.60	9.97	<1	<1	<1	<1	<1	<1	<1
		0.9	mean	34.48	11.95	<1	1.99	<1	<1	<1	<1	<1
			R_{95}	35.08	12.78	<1	1.32	<1	<1	<1	<1	<1
2	Poisson	0.1	mean	<1	<1	<1	<1	<1	<1	<1	<1	<1
			R_{95}	<1	<1	<1	<1	<1	<1	<1	<1	<1
		0.5	mean	1.05	<1	<1	<1	<1	<1	<1	<1	<1
			R_{95}	1.01	<1	<1	<1	<1	<1	<1	<1	<1
		0.9	mean	23.89	8.17	<1	<1	<1	<1	<1	<1	<1
			R_{95}	28.28	11.32	<1	<1	<1	<1	<1	<1	<1
	MAP	0.1	mean	<1	<1	<1	<1	<1	<1	<1	<1	<1
			R_{95}	<1	<1	<1	<1	<1	<1	<1	<1	<1
		0.5	mean	39.31	22.86	<1	<1	<1	<1	<1	<1	<1
			R_{95}	35.38	18.59	<1	<1	<1	<1	<1	<1	<1
		0.9	mean	36.22	16.76	<1	3.54	<1	<1	<1	<1	<1
			R_{95}	36.51	17.16	<1	3.43	<1	<1	<1	<1	<1

4.3 The multi-node system

We conclude this section by noticing that the analysis of a single computing node can be extended to the multi-node case, and therefore to evaluate the overall performance of the server farm. This result follows by observing that, under either round-robin or random allocation, if the overall arrival process is a MAP(D_0,D_1), the arrival process to each node is also a MAP. Under round-robin allocation, the parameters are

$$C_0 = \begin{bmatrix} D_0 & D_1 & . & \cdots \\ . & D_0 & D_1 & \cdots \\ \vdots & \vdots & \ddots & \ddots \\ . & . & \cdots & D_0 \end{bmatrix}, \quad C_1 = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ D_1 & 0 & \cdots & 0 \end{bmatrix}.$$

These matrices are of size nm_a, where n is the number of computing nodes in this group. Under the random allocation, the arrival process has parameters

$$C_0 = D_0 + (1-p)D_1, \quad C_1 = pD_1,$$

where p=$1/n$. As the arrival process to each computing node is a MAP, its performance measures are readily obtained with the analytical model proposed.

5. EXPERIMENTAL VALIDATION

We now demonstrate that the proposed model is able to provide accurate results with quite moderate values of the limit C. To study its behavior, we focus on the mean response time, and the p^{th} response time percentile, denoted as R_p, which is the maximum response time faced by p% of the successful jobs. With these two metrics, Table 3 summarizes the relative errors of the values obtained from the approximated model (finite C), against simulation results. We show the errors achieved when r=2 and 3, and consider different system settings. Thanks to the flexibility offered by the proposed model, we consider Poisson arrivals, and MAPs of second order. While the Poisson case is a standard assumption, MAPs allow us to explore the effect of the variability and auto-correlation of the inter-arrival times on the accuracy of the approximation. We use the method in [9] to obtain the MAP representations, for which we set the squared coefficient of variation (SCV) to be 10 and the decay rate of the auto-correlation function to be 0.9. We denote the reliability of the system without replication by *NR-reliability*, which is given by $\mu/(\mu+\alpha)$%. The service rate μ is set to be 1.0, and the failure rate α is set to achieve different NR-reliability levels, namely 10%, 50% and 90%. Further, the arrival rate is set to achieve different load levels for the system without replication, denoted as *NR-load*. For each configuration we consider different values of the limit C: 5, 10 and 50. For each setting, the simulations were run for 5,000 times with 500,000 samples each time, from which we obtain the mean response time and R_{95}, and their 95% confidence intervals.

Focusing first on the cases with r=3 and the NR-reliability of 50% and 90%, we observe that the relative error is below 1% for most test cases with C as small as 5, with the exception of the case with MAP arrivals and 0.9 load, which requires a larger C of 10 to achieve an error below 1%. Here both the load and the more variable and auto-correlated arrival process increase the likelihood of larger queues and larger differences among queue lengths, thus requiring a larger limit C. However, when the NR-reliability is just 10%, we observe significant errors for several cases with C=5 and 10. For instance, with MAP arrivals and 0.9 load, the error rate with C=5 and 10 are 34.48% and 11.95% for the mean, and 35.08% and 12.78% for R_{95}, respectively. The low NR-reliability case is more challenging as the higher likelihood of failures increases the differences among the queue lengths. Increasing the limit C to 50 allows us to obtain errors below 1%. With medium and high NR-reliability levels, failures are less frequent and the queues stay more synchronized, such that a small C is sufficient to cover most of the

Table 4: Transition rates of matrix S

From		To		Rate	Condition
$(l_r(t),\ldots,l_1(t))$	$Y(t)$	$(l_r(t),\ldots,l_1(t))$	$Y(t)$		
$(l_r,\ldots,l_i,\ldots,l_1)$	(i,j)	$(l_r,\ldots,l_{i+1}{+}1,l_i{-}1,\ldots,l_1{-}1)$	$(i{+}1,j{-}1)$	$l_1\beta$	$j{\geq}1$
$(l_r,\ldots,l_b,\ldots,l_1)$	(i,j)	$(l_r,\ldots,l_b{-}1,\ldots,l_i{-}1,\ldots,l_{i+b}{+}1,\ldots,l_1)$	$(i{+}b,j{-}b)$	$bl_b\mu$	$1{<}b{\leq}r,j{\geq}b$
$(l_r,\ldots,l_b,\ldots,l_1)$	(i,j)	$(l_r,\ldots,l_b{-}1,l_{b-1}{+}1,\ldots,l_{i+1}{+}1,l_i{-}1,\ldots,l_1)$	$(i{+}1,j{-}1)$	$bl_b\alpha$	$1{<}b{\leq}r,j{\geq}1$
$(l_r,\ldots,l_i,\ldots,l_1)$	(i,j)	$(l_r,\ldots,l_i,\ldots,l_1)$	$(i,j{-}1)$	$i\alpha$	$1{\leq}i{\leq}r,j{\geq}1$

Table 5: Transition rates of matrix $A^{(\text{jump})}$

From		To		Rate	Condition
$(l_r(t),\ldots,l_1(t))$	$Y(t)$	$(l_r(t),\ldots,l_1(t))$	$Y(t)$		
$(l_r,\ldots,l_i,\ldots,l_1)$	(i,j)	$(l_r,\ldots,l_i,\ldots,l_1)$	$(i,r{-}i)$	$i\mu$	$1{\leq}i{\leq}r$
$(l_r,\ldots,l_1)$	$(1,0)$	$(l_r,\ldots,l_1)$	$(1,r{-}1)$	β	$i{=}1,j{=}0$
$(l_r,\ldots,l_1)$	$(i,0)$	$(l_r,\ldots,l_1)$	$(1,r{-}1)$	$l_1\beta$	$1{\leq}i{\leq}r,j{=}0$
$(l_r,\ldots,l_i,\ldots,l_1)$	$(i,0)$	$(l_r,\ldots,l_i{-}1,l_{i-1}{+}1,\ldots,l_1{+}1)$	$(1,r{-}1)$	$i\alpha$	$1{\leq}i{\leq}r,j{=}0$
$(l_r,\ldots,l_b,\ldots,l_1)$	$(i,0)$	$(l_r,\ldots,l_b{-}1,l_{b-1}{+}1,\ldots,l_1{+}1)$	$(1,r{-}1)$	$bl_b\alpha$	$1{<}b{\leq}r$
$(l_r,\ldots,l_b,\ldots,l_1)$	(i,j)	$(l_r,\ldots,l_b{-}1,\ldots,l_{b-j}{+}1,\ldots,l_{i+j}{+}1,\ldots,l_i{-}1,\ldots,l_1)$	$(b{-}j,r{-}b{+}j)$	$bl_b\mu$	$1{<}b{\leq}r,j{<}b$

queue-length differences observed.

Considering the cases with $r{=}2$ we observe similar trends, with errors increasing with the load and the arrival process variability, and decreasing with the NR-reliability. However, we observe larger errors than with $r{=}3$ thus requiring a larger C to achieve the same level of accuracy. This can be explained by noticing that with r replicas and limit C the maximum difference allowed by the model between the shortest and the largest queues is rC. Thus a larger number of replicas allows the model to consider larger queue-length differences, given the same limit C.

6. THE CENTRALIZED SET-UP

We now consider the set-up with a centralized queue as depicted in Figure 1(c). In this case, a request is replicated such that its r copies join the central queue in a computing node, and are submitted to the next server that becomes available with FCFS scheduling. Similar to the distributed set-up, jobs are distributed to nodes with either round-robin or random scheduling, but the analysis can focus on a single computing node by appropriately modifying the arrival process. The analysis extends [16], which considered the case with $r{=}2$, to any number of replicas.

6.1 The waiting-time distribution

Similar to the distributed case, we define a bivariate Markov process $\{X(t),J(t)|t{\geq}0\}$, where the *age* $X(t)$ is the total time-in-system of the youngest job in service at time t. Different from the distributed case, the phase $J(t)$ is defined as $J(t)=(l_r(t),\ldots,l_1(t),Y(t))$, where $l_i(t)$ is the number of jobs with i replicas in service, and $Y(t)$ holds the state of the youngest job in service. Thus, $Y(t){=}(i,j)$ means that i replicas of the youngest job are in service, j replicas are waiting in the queue, and $r{-}i{-}j$ replicas already failed. $Y(t)$ therefore takes value in the set $S_Y{=}\{(i,j)|1{\leq}i{+}j{\leq}r,i{\geq}1,j\geq 0\}$.

To determine the PH representation $(\boldsymbol{s}_{\text{wait}},S_{\text{wait}})$ of the waiting-time distribution, we follow similar steps as in Section 4, by solving Eq. (1) to find the matrix T that defines the matrix exponential representation $\boldsymbol{\pi}(x){=}\boldsymbol{\pi}(0)\exp(Tx)$ of the stationary version of the process $(X(t),J(t))$. We therefore need to define the matrices S and $A^{(\text{jump})}$, which hold the transition rates of the service process associated to transitions without and with the start of a new job service, respectively, as summarized in Tables 4 and 5. For matrix S the first row considers the case where one of the l_1 jobs with a single replica in service either completes service or fails, with rate $\beta{=}\mu{+}\alpha$, allowing one of the j replicas of the youngest job in the queue to start service. In the second row, one of the l_b jobs with b replicas in service terminates successfully, canceling its siblings, and letting b new replicas to start service. Notice that the number of replicas of the youngest job waiting (j) must be at least b to ensure that no new job starts service. Similarly, if one of the b replicas of either of these l_b jobs fails, with rate α, one replica in the queue starts service. The last row for this matrix covers the case where one of the i replicas of the youngest job fails, allowing one of its siblings waiting to start service. For the $A^{(\text{jump})}$ matrix we consider similar scenarios, the main difference being that the number of replicas of the youngest job waiting j is assumed to be zero in most of the cases, as this implies that the next replica to join service will be part of a new job, and the transition thus corresponds to $A^{(\text{jump})}$. The only exception is in the last row, where a job with b replicas finishes successfully, and since $j{<}b$, this allows the j replicas of the youngest job in the queue, and $b{-}j$ replicas of a new job, to start service.

Using these matrices we can solve Eq. (1) to find T, and then define a similar system as that in Eq. (2) to find the $\boldsymbol{\pi}(0)$, the distribution of the phase at the beginning of an all-busy period. The main difference here is that the generator of the service process during the not-all-busy period, $S_{\text{not-all}}$ in Eq. (2), requires a more detailed analysis, similar to the one developed in [16]. The key idea is that this generator has a block structure amenable for analysis, namely a level-dependent quasi-birth-and-death process [11]. Further details can be found in [16]. After finding $\boldsymbol{\pi}(0)$ we can use Eq. (3) to determine the PH representation of the waiting-time distribution $(\boldsymbol{s}_{\text{wait}},S_{\text{wait}})$.

6.2 The service-time distribution

The next step is to show that the service-time distribution has a PH distribution with parameters $(\boldsymbol{s}_{\text{ser}},S_{\text{ser}})$. For the service time, we focus on a tagged job in service, and let $Y(t)$ to be the service phase of this job at time t. Here $Y(t)$ takes values from $S_Y{=}\{(i,j)|1{\leq}i{+}j{\leq}r,i{\geq}1,j\geq 0\}$, as defined be-

Table 6: Transition rates of S_{ser}

Condition	From		To		Rate	Condition
	$(l_{r-1}(t),\ldots,l_1(t))$	$Y(t)$	$(l_{r-1}(t),\ldots,l_1(t))$	$Y(t)$		
$j=0$	/	$(i,0)$	/	S	$i\mu$	$1\leq i\leq r$
	/	$(i,0)$	/	$(i-1,0)$	$i\alpha$	$1<i\leq r$
	/	$(1,0)$	/	F	α	$i=1$
$r-i=1$	/	$(r-1,1)$	/	S	$(r-1)\mu$	/
	/	$(r-1,1)$	/	$(r,0)$	β	/
	/	$(r-1,1)$	/	$(r-1,0)$	$(r-1)\alpha$	/
$r-i>1, j\geq 1$	$(l_{r-1},\ldots,l_i,\ldots,l_1)$	(i,j)	/	S	$i\mu$	$1\leq i\leq r$
	$(l_{r-1},\ldots,l_1)$	(i,j)	$(l_{r-1},\ldots,l_1-1)$	$(i+1,j-1)$	$l_1\beta$	$j\geq 1$
	$(l_{r-1},\ldots,l_b,\ldots,l_1)$	(i,j)	$(l_{r-1},\ldots,l_b-1,l_{b-1}+1,\ldots,l_1)$	$(i+b,j-b)$	$bl_b\mu$	$1<b\leq r, j\geq b$
	$(l_r,\ldots,l_b,\ldots,l_1)$	(i,j)	/	$(i+b,0)$	$bl_b\mu$	$1<b\leq r, j<b$
	$(l_r,\ldots,l_b,\ldots,l_1)$	(i,j)	$(l_r,\ldots,l_b-1,l_{b-1}+1,\ldots,l_1)$	$(i+1,j-1)$	$bl_b\alpha$	$1<b\leq r, 1\leq j<b$

fore. Clearly, the service time of a tagged job is affected by other jobs in service. Specifically, when the tagged job has replicas waiting in the queue, these replicas can only start service if one of the replicas in service frees a server. Thus it is essential to keep track of the states of all jobs in service, which can be done by means of $S(t)=(l_{r-1}(t),\ldots,l_1(t))$, where $l_i(t)$ is again the number of jobs with i replicas in service at time t, *but excluding the tagged job*.

Similarly to the case with individual queues, we define two absorbing states S and F that represent the cases where the job completes service successfully or encounters a failure, and describe the evolution of the service process as an MC with generator S_{ser}, given in Table 6. We split the transitions in three sets, and the first case in any of these sets considers the successful completion of the tagged job caused by a successful tagged replica. The first set considers the case where the tagged job has zero replicas waiting in the queue. Thus, if any of its i replicas in service fails (second row) the number of replicas in service decreases by one. In case there is only one tagged replica in service, $i=1$, a failure triggers the failure of the whole job (third row). The second set covers the special case where 1 tagged replica is waiting in the queue while the other $r-1$ are in service. In the second row we consider the successful completion or failure, with rate $\beta=\mu+\alpha$, of the only non-tagged replica in service, which allows the tagged replica in the queue to start service. The third row instead considers the failure of one of the tagged replicas, which also allows the tagged replica in the queue to start service. Notice that in the first two sets the evolution is independent of the state of the other jobs in service, either because there are zero tagged replicas in the queue, or because there are $r-1$ tagged replicas in execution. The other cases, which depend on the state of the non-tagged jobs in service, are described in the third set, starting with the successful completion of any of the i tagged replicas in service. In the second row, one of the l_1 jobs with a single replica in service either completes service successfully or fails, allowing one more tagged replica to start service. In the third and fourth rows, one of the l_b jobs with $b>1$ replicas in service completes successfully, allowing b new replicas to start service. If there are at least as many tagged replicas as free servers, i.e. $j\geq b$, as in the third row, we are left with $j-b$ tagged replicas waiting in queue. If not, $j<b$ as in the fourth row, all tagged replicas waiting in the queue start service and we can just focus on the tagged replicas in service, ignoring any other jobs. In the last row, one of the b replicas of the l_b jobs in service fails, with $b>1$, allowing a new tagged replica to start service.

Having obtained S_{ser}, $\boldsymbol{\pi}(0)$ and T, we can determine the initial probability vector $\boldsymbol{s}_{\text{ser}}$ as in Eq. (4). Actually, using the block structure of the service process generator during the not-all-busy-period $S_{\text{not-all}}$, described in the previous section, we can improve this computation in a manner similar to [16]. Further, with this representation of the service time for all jobs, we can obtain the PH representation of the service-time distribution for successful jobs by directly applying Proposition 1. Finally, the PH representation of the response-time distribution $(\boldsymbol{s}_{\text{res}}, S_{\text{res}})$ is obtained by using Theorem 1.

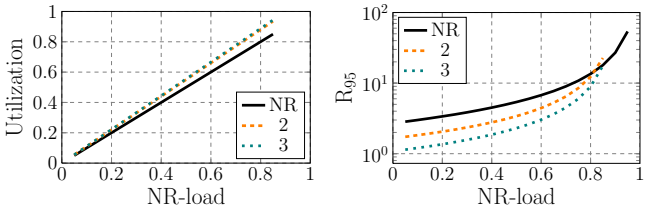
7. EXPERIMENTAL RESULTS

In this section, we make use of the proposed model for the distributed set-up to evaluate its performance in the terms of its reliability and offered response times. We also compare against the centralized set-up and determine the performance gains obtained by keeping a central queue. Through the whole section, the service rate μ is set to 1, and the mean arrival and failure rates are set in proportion to μ to obtain different load levels and NR-reliability.

7.1 Performance of the distributed set-up

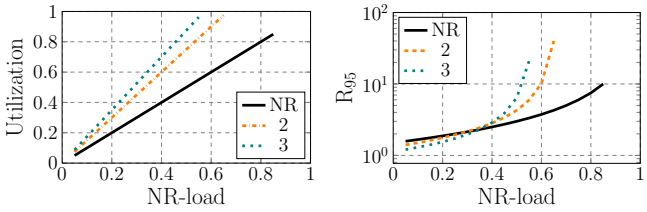
In the distributed set-up, as well as in the centralized one, the reliability, i.e. the probability that a job completes service successfully, increases with the deployment of extra replicas, and the improvement is actually independent of the queueing model implemented. For both the distributed and the centralized set-ups, with replication level $r \geq 1$ the reliability achieved is $1-(\alpha/(\mu+\alpha))^r$, which simplifies to $\mu/(\mu+\alpha)$ if no replication is adopted. Beyond the evident improvement in reliability, another benefit of replication with canceling is its potential to reduce the response time by allowing the selection of the first successful result. Focusing first on the case with individual queues, we compare the response times achieved under different replication levels. We consider three different systems with r servers, mean arrival rate $r\lambda$, and replication level r, for $r=1$, 2 and 3, thus offering the same load to all systems. Figures 2, 3 and 4 show how the utilization and R_{95} change under different replication levels, under Poisson arrivals, and different NR-reliability levels (10%, 50% and 90%).

Figure 2(a) depicts how the system utilization increases as the replication level increases, although the increase is relatively minor. In this case we assume an NR-reliability of 90%. Decreasing this reliability to 50% and 10%, as de-



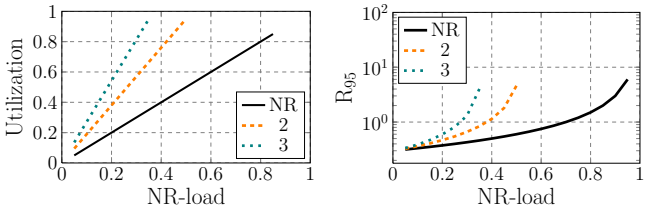
(a) Utilization (b) R_{95}

Figure 2: Poisson arrival, NR-reliability: 90%



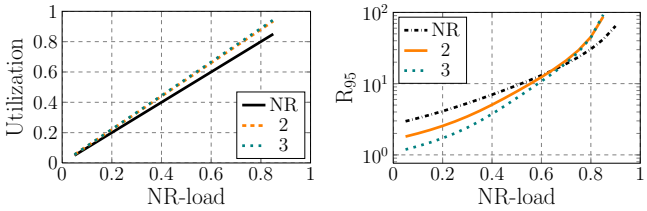
(a) Utilization (b) R_{95}

Figure 3: Poisson arrivals, NR-reliability: 50%



(a) Utilization (b) R_{95}

Figure 4: Poisson arrivals, NR-reliability: 10%



(a) Utilization (b) R_{95}

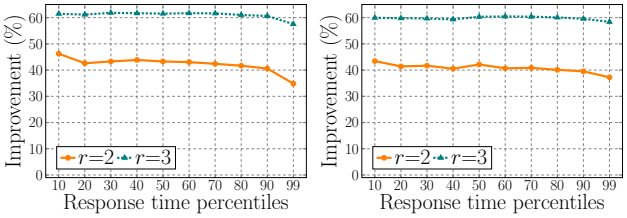
Figure 5: MAP arrivals, NR-reliability: 90%

picted in Figures 3(a) and 4(a), the relative increase in utilization when replication is implemented is larger. For instance, when the NR-reliability is just 10%, a utilization of 0.35 without replication becomes 0.67 when r=2, and further increases to 0.95 when r=3. However, the increase in utilization does not necessarily lead to a higher delay. On the contrary, replication leads to lower response times, as long as the baseline NR-load stays below a certain threshold. For instance, Figure 2(b) shows that both replication levels achieve a lower R_{95} than without replication, when the NR-load is below 0.85. In particular, the system with 3 replicas achieves the lowest response times among the three. Although the introduction of replicas introduces extra load, which leads to an increase in the response times, at the same time it allows the selection of the first replica that finishes, potentially reducing the response times. However, the second effect weakens with a higher failure rate, as we observe in Figures 2(b), 3(b) and 4(b) that the NR-load threshold decreases with decreasing NR-reliability. In particular, when the NR-reliability is 10%, the introduction of replication increases the response times at any NR-load considered. The reason is two-fold: first, the system without replication and a low NR-reliability shows short response times since only short jobs can complete service before a failure; second, the probability that all replicas are running until the first one completes reduces with a higher failure rate, weakening the benefit of selecting the first replica that completes service.

Figure 5 shows the utilization and R_{95} obtained under MAP arrivals and an NR-reliability of 90%, while keeping the same mean arrival rate as for Poisson arrivals. We observe how the bursty workload modeled by the MAP arrivals leads to much larger response times and a lower NR-load threshold, below which replication reduces response times, compared to the case with Poisson arrivals in Figure 2. This is caused by the more variable and auto-correlated workload represented by the MAP arrivals, which limit the load range where replication is beneficial.

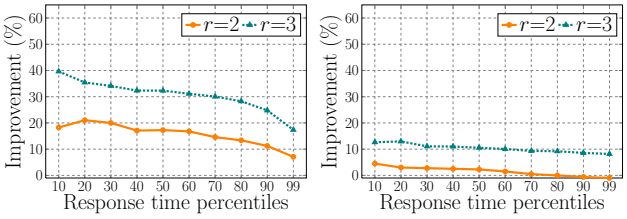
7.2 The Effect across the Distribution

We now look further into the effect that replication has



(a) NR-load:0.1 (b) NR-load:0.3

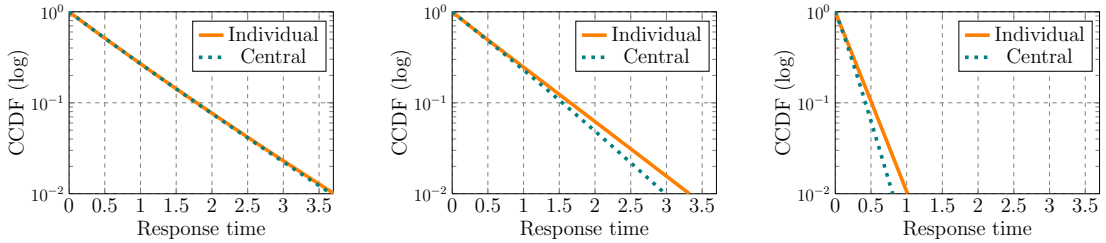
Figure 6: Improvement on R_p (Poisson, 90% NR-reliability)



(a) NR-load:0.1 (b) NR-load:0.3

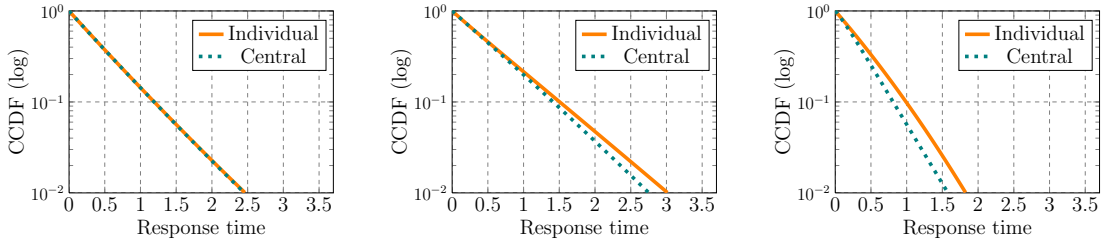
Figure 7: Improvement on R_p (Poisson, 50% NR-reliability)

across the whole response-time distribution. To this end, for each percentile p in the range {10, 20, ..., 90, 99} we obtain the relative improvement $(R_p^1 - R_p^r)/R_p^1$%, comparing the response time percentiles obtained with replication (r=2, 3) against those without (R_p^1). Figure 6(a)-(b) depicts the improvements for the case under Poisson arrivals with NR-reliability of 90% and NR-load levels of 0.1 and 0.3, respectively. We observe fairly stable improvements across the whole percentile range, with the exception of the improvement on the tail. For instance, when the NR-load is 0.1, and r=2, the improvement on the 99$^{\text{th}}$ percentile is 34.83%, while the improvement experienced by most percentiles is around 43%. In fact, we observe that the improvement generally decreases as the percentile considered increases. The reason for this is that a larger percentile covers longer re-



(a) NR-reliability: 90% (b) NR-reliability: 50% (c) NR-reliability: 10%

Figure 8: CCDFs of response times with individual queues and with a central queue (Poisson arrivals, r=2, NR-load:0.3)



(a) NR-reliability: 90% (b) NR-reliability: 50% (c) NR-reliability: 10%

Figure 9: CCDFs of response times with individual queues and with a central queue (Poisson arrivals, r=3, NR-load:0.3)

sponse times, which are partially associated to longer service times. In a long service time there is more room for a failure to occur, in which case the system looses its ability to use the minimum of r execution times. Notice that this effect decreases if r is larger, as shown in Figure 6(a)-(b), as even if one replica fails, there are $r-1$ replicas that could execute concurrently and among which the system selects the minimum execution time. Further, this effect is stronger if the utilization is low, as confirmed when comparing Figure 6(a) and (b), since under low loads the contribution of the service time on the response time is larger as the queueing times are very short.

In the previous set-up we assumed an NR-reliability of 90%. If we reduce it to 50%, Figure 7(a)-(b) shows that the gains across the percentiles are not uniform, where we observe peaks on small percentiles, and a stronger decrease in the improvement for larger percentiles. For instance, when r=3 and NR-load is 0.1, the improvement of the 10^{th} percentile is 39.64%, while it is 17.36% for the 99^{th} percentile. This more pronounced effect is in agreement with the observation above as in this case failures are more likely and higher percentiles evaluate conditions where service times can be larger and provide more chances for failures to occur. This clearly highlights the importance of explicitly considering the response-time distribution, and not just its mean, or one percentile, when evaluating a replication strategy.

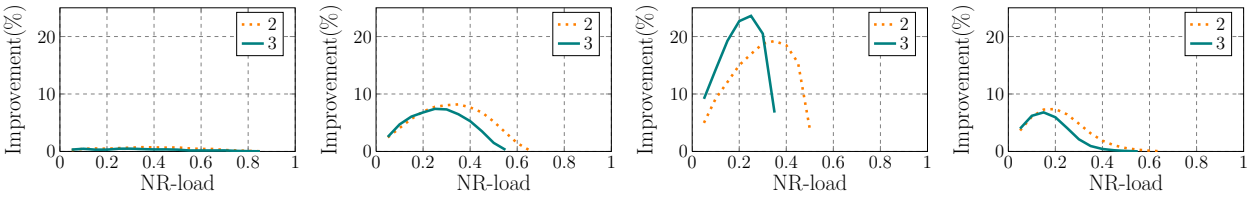
7.3 Distributed vs Centralized

Clearly, the distributed and centralized set-ups perform identically in terms of reliability, as this only depends on the replication level r, given our independence assumptions. In terms of response times, the centralized set-up must achieve lower response times than its distributed counterpart, as in the latter case it is possible for some servers to have non-zero waiting lines, while other servers remain idle. This is not possible in the centralized set-up, as all servers must be busy if there is any job waiting to start service. To better understand this difference in performance we show in Figure 8(a) the complementary CDFs (CCDFs) of the response times achieved with 2 replicas by these two models, assuming Poisson arrivals, NR-reliability of 90%, and NR-load of 0.3 as an example. Clearly, it is hard to tell the difference between these two CCDFs. Reducing the NR-reliability to 50% and 10%, Figure 8(b)-(c) shows a more significant improvement of the centralized over the distributed set-up along the whole distribution. Increasing the replication level to r=3, Figure 9 shows that when the NR-reliability is high, the distributed set-up performs almost as well as the centralized one, while this difference increases when the NR-reliability decreases. Figures 8 and 9 also show that the improvement is not uniform along the distribution, and that it is more significant in the tail than in the body (e.g. second and third quartiles).

Considering different NR-load levels, Figure 10(a) examines an NR-reliability of 90%, where the centralized set-up shows a relative improvement on the R_{95} of less than 1% over the distributed case. As discussed above, Figure 10(b)-(c) shows that reducing the NR-reliability increases the advantage of the centralized set-up, and this can be over 20% for NR-load around 0.2 and r=3 when the NR-reliability is just 10%. If we modify the arrival process considered in Figure 10(b) from a standard Poisson to a correlated MAP, Figure 10(d) shows that the gains obtained with the centralized operation are similar in magnitude but are restricted to a smaller subset of values for the NR-load, as with MAP arrivals replication is beneficial for a more limited load range. Similar results, both in trend and magnitude, can be observed if we compare the *mean* response time or other percentiles instead of the R_{95}.

8. DISCUSSION

In the previous section we observed, as expected, that the



(a) Poisson, 90% NR-reliability (b) Poisson, 50% NR-reliability (c) Poisson, 10% NR-reliability (d) MAP, 50% NR-reliability

Figure 10: Improvement on R_{95} of the central model over the individual queues mode

centralized set-up achieves lower response times than the distributed case, especially at low NR-reliability. However, most large server farms dispatch incoming request to servers immediately, without holding a central queue. Replication with distributed queues is also required when accessing mirrored disks and searching distributed databases in parallel. This set-up provides more flexibility since there is no need to set up and manage a central dispatcher that keeps track of the servers state, making it easier to increase or decrease the number of servers as the load rises or falls.

A major point of our investigation is to assess the magnitude of the difference in response time between the two set-ups, to see if and when the distributed set-up suffers an excessive performance penalty. From our results, we observe that, although the centralized system performs better than the distributed set-up, this difference is quite small, under 1%, when the NR-reliability is high (90%). Even if this reliability is 50%, which is already low as it assumes that one out of two requests fail, the improvement obtained with the centralized queue is at most 8%. We therefore observe that replication not only is effective to reduce the response times in the distributed set-up, as shown in Sections 7.1 and 7.2, but it also provides a performance close to that of the centralized set-up as long as the NR-reliability is not too low. In addition, the proposed models focus on request failures, but not on server failures. While in the distributed system the replicas in a job receive service from distinct servers, in the centralized set-up several replicas in a job may be submitted to the same server as the failure of one replica allows its sibling in the queue to start at the same server, increasing the risk of failure. The distributed set-up thus offers many benefits over the centralized case, while its performance disadvantage is limited when the NR-reliability is high.

In the future, we intend to investigate other parameters that may affect the performance of the proposed approach, including the overhead of the replication and cancellation mechanisms, and the correlation between request replicas.

9. ACKNOWLEDGMENTS

The research presented in this paper was supported by the EPSRC grant EP/L00738X/1 (iBids). The research of Juan F. Pérez is supported by the ARC Centre of Excellence for Mathematical and Statistical Frontiers (ACEMS).

10. REFERENCES

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

[2] 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.*, 37:9–29, 2001.

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

[4] A. Gandhi, M. Harchol-Balter, R. Das, and C. Lefurgy. Optimal power allocation in server farms. In *ACM SIGMETRICS PER*, pages 157–168, 2009.

[5] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel. The cost of a cloud: research problems in data center networks. *ACM SIGCOMM CCR*, 39:68–73, 2008.

[6] B. Guenter, N. Jain, and C. Williams. Managing cost, performance, and reliability tradeoffs for energy-aware server provisioning. In *IEEE INFOCOM*, 2011.

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

[8] Q. He. Analysis of a continuous time SM[K]/PH[K]/1/FCFS queue: Age process, sojourn times, and queue lengths. *JSSC*, 25:133–155, 2012.

[9] A. Heindl, G. Horváth, and K. Gross. Explicit inverse characterizations of acyclic MAPs of second order. In *EPEW*, 2007.

[10] G. Joshi, E. Soljanin, and G. Wornell. Efficient redundancy techniques for latency reduction in cloud systems. *arXiv preprint arXiv:1508.03599*, 2015.

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

[12] I. Mitrani. Managing performance and power consumption in a server farm. *Ann. Oper. Res.*, 202:121–134, 2013.

[13] M. T. Özsu and P. Valduriez. Distributed and parallel database systems. *ACM CSUR*, 28:125–128, 1996.

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

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

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

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

[18] Z. Qiu, J. F. Pérez, and P. G. Harrison. Beyond the mean in fork-join queues: Efficient approximation for response-time tails. *Perform. Eval.*, 2015.

[19] B. Sengupta. Markov processes whose steady state distribution is matrix-exponential with an application

to the GI/PH/1 queue. *AAP*, 21:159–180, 1989.
[20] N. B. Shah, K. Lee, and K. Ramchandran. When do redundant requests reduce latency? In *Allerton*, 2013.
[21] B. Snyder. Server virtualization has stalled, despite the hype. http://www.infoworld.com/print/146901, 2010.
[22] A. Vulimiri, P. Godfrey, R. Mittal, J. Sherry, S. Ratnasamy, and S. Shenker. Low latency via redundancy. In *CoNEXT*, 2013.
[23] M. Wu, X.-H. Sun, and H. Jin. Performance under failures of high-end computing. In *ACM/IEEE SC*, page 48, 2007.
[24] Z. Zheng and Z. Lan. Reliability-aware scalability models for high performance computing. In *IEEE CLUSTER*, 2009.

APPENDIX

Proof of Proposition 1

Since jobs only fail during service, the failure probability is given by $p_S = \int_0^\infty \boldsymbol{s}_{\text{ser}} \exp(S_{\text{ser}}x)\boldsymbol{S}_S^* dx = -\boldsymbol{s}_{\text{ser}} S_{\text{ser}}^{-1}\boldsymbol{S}_S^*$. Thus, the probability that a job service time lasts for at most x time units and succeeds is

$$\begin{aligned} F_s(x) &= \frac{1}{p_S}\int_0^x \boldsymbol{s}_{\text{ser}} \exp(S_{\text{ser}}y)\boldsymbol{S}_S^* dy \\ &= 1 + \frac{1}{p_S}\boldsymbol{s}_{\text{ser}} S_{\text{ser}}^{-1} \exp(S_{\text{ser}}x)\boldsymbol{S}_S^*. \end{aligned}$$

This matrix-exponential representation of the service times for successful jobs can be turned into a PH representation by defining a diagonal matrix Π such that $\Pi\mathbf{1}=\eta'$, where $\eta=-\boldsymbol{s}_{\text{ser}}S_{\text{ser}}^{-1}$ is the stationary distribution of the service phase. Defining $B_{\text{ser}}=\Pi^{-1}S'_{\text{ser}}\Pi$, we have

$$\begin{aligned} 1 - F_s(x) &= -\frac{1}{p_S}\boldsymbol{s}_{\text{ser}} S_{\text{ser}}^{-1}\Pi^{-1}\Pi \exp(S_{\text{ser}}x)\Pi^{-1}\Pi\boldsymbol{S}_S^* \\ &= \frac{1}{p_S}\eta\Pi^{-1} \exp(B'_{\text{ser}}x)\Pi\boldsymbol{S}_S^* \\ &= \frac{1}{p_S}\boldsymbol{S}_S^{*\prime}\Pi \exp(B_{\text{ser}}x)\mathbf{1}. \end{aligned}$$

This defines a proper PH distribution as the vector $\boldsymbol{S}_S^{*\prime}\Pi/p_S$ is stochastic. This results from $\boldsymbol{S}_S^*$ and Π being non-negative, and $\boldsymbol{S}_S^{*\prime}\Pi\mathbf{1} = -\boldsymbol{S}_S^{*\prime}(\boldsymbol{s}_{\text{ser}}S_{\text{ser}}^{-1})' = p_S$.

Proof of Theorem 1

The proof of this result follows similar arguments as that of [18, Theorem 1], so we focus on the main differences. As in [18, Theorem 1], we rely on the fact that the PH representation of the waiting time distribution is obtained from a time-reversal argument [19], thus we also use this argument to build the PH representation of the response times. Since we focus on the *successful* jobs only, we follow Proposition 1 to obtain the PH representation of the service time distribution of successful jobs. Further, we split this representation for jobs that wait and jobs that do not. The initial service phase of jobs that wait is given by $\boldsymbol{\alpha}_{\text{busy}} = c\boldsymbol{\pi}_{\text{busy}}(T-S^{(\text{MAP})})$, as this is the distribution of the phase *just after a downward jump in* $X(t)$, and c is a normalizing constant such that $c^{-1}=\boldsymbol{\pi}_{\text{busy}}(T-S^{(\text{MAP})})\mathbf{1}$. Thus we apply a time-reversal by defining the diagonal matrix Δ_{busy} such that $\Delta_{\text{busy}}\mathbf{1}=-\gamma(\boldsymbol{\alpha}_{\text{busy}}S_{\text{ser}}^{-1})'$. We then follow similar steps as in the proof of Proposition1 to obtain the PH representation for jobs that wait $(\boldsymbol{\beta}_{\text{ser}}^{\text{busy}}, B_{\text{ser}}^{\text{busy}})$ as

$$\boldsymbol{\beta}_{\text{ser}}^{\text{busy}}=\boldsymbol{S}_S^*\Delta_{\text{busy}}/p_S, \quad B_{\text{ser}}^{\text{busy}}=\Delta_{\text{busy}}^{-1}S'_{\text{ser}}\Delta_{\text{busy}}.$$

A similar result is obtained for jobs that do not wait, considering that in this case jobs start service according to $(1-\gamma)\boldsymbol{\pi}(0)$. We thus define Δ_{idle} as a diagonal matrix such that $\Delta_{\text{idle}}\mathbf{1}=-(1-\gamma)(\boldsymbol{\pi}(0)S_{\text{ser}}^{-1})'$, and the corresponding PH representation $(\boldsymbol{\beta}_{\text{ser}}^{\text{idle}}, B_{\text{ser}}^{\text{idle}})$ is given by

$$\boldsymbol{\beta}_{\text{ser}}^{\text{idle}}=\boldsymbol{S}_S^*\Delta_{\text{idle}}/p_S, \quad B_{\text{ser}}^{\text{idle}}=\Delta_{\text{idle}}^{-1}S'_{\text{ser}}\Delta_{\text{idle}}.$$

With these PH representations for the service time, we obtain Eq. (5) by putting together the paths of jobs that do not wait, which start service with $\boldsymbol{\beta}_{\text{ser}}^{\text{idle}}$, with those of jobs that wait, which start service with $\boldsymbol{\beta}_{\text{ser}}^{\text{busy}}$. Given the time-reversal, the response time of jobs that wait is composed of a first stage of service followed by a second stage of waiting. Further, the phase in which the service stage ends determines the stage in which the waiting stage begins. The remaining of the proof follows the same steps as that of [18, Theorem 1], such that the matrix $\tilde{P}_{s,w}=\Gamma^{-1}(T-S^{(\text{MAP})})'\Lambda$ is a stochastic matrix that determines how the phase at the end of the service phase determines the phase at the beginning of the waiting phase. Once a job starts the waiting phase, it evolves according to S_{wait} until absorption. Further details can be found in [18].