Modeling of Request Cloning in Cloud Server Systems using Processor Sharing

Tommi Nylander
tommi.nylander@control.lth.se
Lund University

Johan Ruuskanen
johan.ruuskanen@control.lth.se
Lund University

Karl-Erik Årzén
karlerik@control.lth.se
Lund University

Martina Maggio
martina.maggio@control.lth.se
Lund University

ABSTRACT

The interest for studying server systems subject to cloned requests has recently increased. In this paper we present a model that allows us to equivalently represent a system of servers with cloned requests, as a single server. The model is very general, and we show that *no* assumptions on either inter-arrival or service time distributions are required, allowing for, e.g., both heterogeneity and dependencies. Further, we show that the model holds for any queuing discipline. However, we focus our attention on Processor Sharing, as the discipline has not been studied before in this context.

The key requirement that enables us to use the single server G/G/1 model is that the request clones have to receive *synchronized service*. We show examples of server systems fulfilling this requirement. We also use our G/G/1 model to co-design traditional load-balancing algorithms together with cloning strategies, providing well-performing and provably stable designs.

Finally, we also relax the synchronized service requirement and study the effects of non-perfect synchronization. We derive bounds for how common imperfections that occur in practice, such as arrival and cancellation delays, affect the accuracy of our model. We empirically demonstrate that the bounds are tight for small imperfections, and that our co-design method for the popular Join-Shortest-Queue (JSQ) policy can be used even under relaxed synchronization assumptions with small loss in accuracy.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; • **Networks** → Network reliability.

KEYWORDS

Cloning, Cloud Computing, Datacenters

ACM Reference Format:
Tommi Nylander, Johan Ruuskanen, Karl-Erik Årzén, and Martina Maggio. 2020. Modeling of Request Cloning in Cloud Server Systems using Processor Sharing. In *Proceedings of ACM/SPEC ICPE (ICPE '20).* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

ICPE '20, April 20–24, 2020, Edmonton, Canada

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In cloud computing, *cloning* is used as a way of speeding up the generation of responses to requests. In this setting, the technique is also known as the generation of *redundant* requests. The basic idea is that, instead of sending requests to only one server, the requests are cloned and sent to multiple servers simultaneously. The response to the request is the result of the server that first completes the processing required to handle the request. When this happens, the pending requests (i.e., the clones that are being processed in the other servers) are cancelled.

Cloning can yield significant improvements to the performance of data centers, as shown in [1]. The motivation for cloning comes from the desire to reduce the mean and tail response times of applications running in the cloud. Hosted virtual machines or containers are allocated on shared resources. This means that their behavior is sometimes unpredictable, and the computation times of similar requests can vary among different instances [5]. Cloning can thus be viewed as an intuitive way to increase the predictability of cloud applications, by relying on multiple simultaneous copies of a user request. This is the reason why there has recently been an upsurge in the interest for modeling the behaviour of cloud applications subject to cloning.

Existing Results. Cloning is a particular case of the (n, k) fork-join model, where a request is split into n sub-tasks that are distributed to servers. The request completes when at least $k \leq n$ of those task are completed. Cloning implies that the n sub-tasks are identical and $k = 1$. Approximate analysis and latency bounds have been extensively studied for the general (n, k) fork-join systems [12, 20, 22], but unfortunately no exact analysis exists when $n \geq 3$. This is, however, not the case for cloning. The first exact analysis of cloning was performed by Gardner et al. [8]. They modeled servers using M/M/1 queues, i.e., queues where the arrivals follow a Poisson process and job service times have an exponential distribution. Other notable contributions concerning cloning with exponential distributions include [3, 9, 19]. Qiu et al. [19] compares the use of multiple queues (in a distributed servers setting) to a central queue. Gardner et al. [9] derived results on the largest marginal improvement that can be obtained using the *Redundancy-d* cloning policy, that clones each request to exactly d servers. Ayesta et al. [3] improved the analysis of *Redundancy-d*, including different alternatives for handling the request cancellation.

Subsequently, researchers started investigating cloning with specific probability distributions for inter-arrival times and service times, identifying the characteristics of the stochastic (inter-arrival and service time) processes that make cloning beneficial [21]. Joshi et al. [11, 13, 14] extended the results obtained with the M/M/1 model to an M/G/1 model, i.e., queues where the arrivals are still determined by a Poisson process, but job service times have a general distribution. However, an underlying assumption for the extension was that all service time distributions are independent and identically distributed (*i.i.d.*), which rules out heterogeneity. This showed that cloning is beneficial if the tail distribution of the service time is log-convex and disruptive if log-concave.

Contribution. In this paper we relax the assumptions made in earlier research contributions. We require *no assumptions* on either inter-arrival or service time distributions, effectively handling heterogeneity. We present a model that is valid with any queuing discipline, however, in this paper we focus on the Processor Sharing (PS) discipline [16]. In fact, to the best of our knowledge, PS has not been studied before in conjunction with request cloning.

The main contributions of this paper are the following:

- We show that the existing equivalent M/G/1 model for cloned systems under i.i.d assumptions can be generalized to allow for *any* inter-arrival or service time distributions, i.e., not requiring the i.i.d assumption. Our G/G/1 model thus allows for both heterogeneous and dependent service time distributions under any queuing discipline, as long as the server system guarantees synchronized service to all request clones. We explore the assumptions that the computing infrastructure needs to fulfill for this to be true.
- For such server systems, we analyze and compute the optimal cloning factor, with respect to the average response times of the server system, for any service time distribution under any load – i.e., the cloning factor that allows us to obtain the lowest possible average latency.
- We analyze more complex server systems, consisting of multiple clusters, and provide a co-design method for joint synthesis of cloning strategy and load-balancing technique. To the best of our knowledge, we present the first provably stable co-designed load-balancing and cloning strategy for the PS discipline.
- We relax the synchronized service assumption and derive bounds for how practical imperfections, such as arrival and cancellation delays, affect the accuracy of our model.

To validate our theoretical findings from a practical standpoint, we built a discrete event simulator with support for request cloning. Our experimental results show that we are able to accurately predict the behaviour of server systems subject to cloning. We empirically demonstrate the benefits of co-designing the cloning factor and load-balancing policy, and that the synchronized service assumption can be relaxed for the popular Join-Shortest-Queue (JSQ) policy with small loss in accuracy. Using simulations, we also show that our theoretical bounds can, especially for low arrival and cancellation delays, be used to predict the effect of practical imperfections on our model.

The remainder of the paper is organized as follows. Section 2 presents our model and Section 3 two examples of results that can be obtained. Section 4 shows applications of the model to capture commonly-used data center structures. Section 5 relaxes the synchronized service assumption and shows how this affects our model. Section 6 shows our experimental evaluation. Section 7 presents related research and Section 8 concludes the paper.

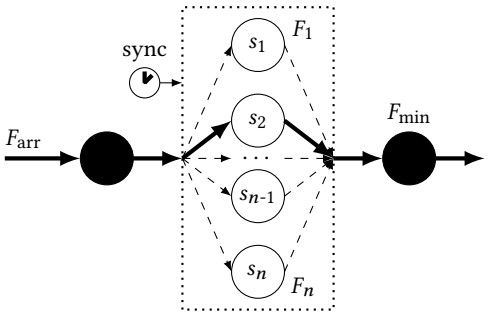


Figure 1: Synchronized service system.

2 SYNCHRONIZED MODEL

This section formally describes *cloning*, and presents the server system that is the subject of this study. Figure 1 shows a setup example, where n synchronized servers accept requests. An incoming stream of requests is received and each of them is cloned to the n servers. In the most common type of cloning, Cancel-on-Complete cloning, the response to the client is produced by the server that completes the request in the minimum amount of time (s_2 in Figure 1, denoted by thick arrows). The request processing in the other servers is then immediately canceled. In the rest of the paper, we describe the statistical distribution of a random variable X_i using its Cumulative Distribution Function (CDF) and denote the CDF with $F_i(x) = P(X_i \leq x)$. Coherent with this notation, in Figure 1 the CDFs of s_1 and s_n are respectively indicated with F_1 and F_n, the CDF of the request inter-arrival times is indicated with F_{arr}, and the CDF of the minimum service time is marked with $F_{\min}$.

We first define the Cancel-on-Complete cloning approach that we use throughout the paper. Then we discuss synchronized service and the assumptions needed for our theoretical analysis. Finally, we present the main results obtained with our model. Our model holds for any queuing discipline, but we focus our analysis and modeling on the PS discipline due to the lack of prior literature results that properly model this discipline and its closeness to real servers implementations.

Definition 1. ***(Cancel-on-Complete cloning – CoC cloning)*** *We define Cancel-on-Complete (CoC) cloning as the act of creating n copies of an original request r^o, denoted with $r^c_{1:n}$. More precisely, $r^c_{1:n}$ indicates the vector of n cloned requests. We refer to the i-th request in the vector using the notation r^c_i. We use a similar notation to indicate servers, where $s_{1:n}$ is the vector of servers and s_i indicates the i-th server. The n requests are simultaneously sent to n servers, $s_{1:n}$, on which they eventually enter service. In time, one of the n servers first terminates the computation needed to serve r^o. When this happens, the response that is produced is forwarded to the client and all remaining $n-1$ clones are immediately canceled.*

Another possible cloning approach is *Cancel-on-Start (CoS)*, where all remaining clones get canceled when the first request clone *starts* its service. However, CoS does not apply to the PS discipline as all clones $r^c_{1:n}$ always enter service *immediately*. As a result, we only consider the CoC approach. For the remainder of this paper, we simply use the word cloning to refer to CoC cloning.

We need perfect cancellation to describe the concept of *synchronized service* that forms the basis for enabling our G/G/1 model.

Assumption 1. ***(Perfect cancellation)*** *We assume perfect cancellation, i.e. that cancellation of requests takes zero time.*

Definition 2. ***(Synchronized service)*** *The $r^c_{1:n}$ request clones (sent to servers $s_{1:n}$) receive synchronized service if the clones $r^c_{1:n}$ both enter and leave service simultaneously, i.e., they are dispatched to the servers simultaneously and they are removed from the servers simultaneously at completion of the first clone, implying CoC cloning. This implies the following conditions for the cloning of all original requests r^o:*

1. *Clones $r^c_{1:n}$ have to be sent simultaneously to all servers $s_{1:n}$.*
2. *The service in the $n-1$ clones that did not produce a complete response has to be terminated using perfect cancellation, as soon as the fastest server completes the response generation for its clone.*

Note that synchronized service does not imply *immediate* service. Request clones $r^c_{1:n}$ do not have to enter service immediately, and can queue at the servers $s_{1:n}$. Synchronized service only requires that requests enter (and leave) service *simultaneously*. In other words, the synchronized service concept is compatible with any queuing discipline as long as the chosen queuing discipline is the same across all servers $s_{1:n}$. For PS, synchronized service implies that all clones $r^c_{1:n}$ of the same original request r^o experience *identical processor shares*.

The basic setup in Figure 1, with $s_{1:n}$ servers that receive clones, can be the basic block for more complex structures where – for example – a load balancer can be placed in front of multiple of these blocks, each containing a different number of servers, creating a possibly heterogeneous hierarchy. For the remainder of this section, we discuss the basic theoretical concepts using a single block with n servers $s_{1:n}$, as shown in Figure 1. The extension to more complex structures is described in Section 4.

To derive our results, we use the following Theorem, developed in the field of statistics.

Theorem 1. ***(Cumulative Distribution Function of the Minimum)*** *Given a set of n random variables $\{X_1, \ldots, X_n\}$ with **any** CDF, and denoting with $F_i(x)$ the CDF of X_i; the CDF of the random variable $X_{\min}$, where $X_{\min} = \min\{X_1, \ldots, X_n\}$ is given by*

$$\begin{aligned} F_{min}(x) \quad = \quad & (-1)^0 \sum_{i=1}^{n} F_i(x) + \\ & (-1)^1 \sum_{i<j} F_{i,j}(x,x) + \\ & (-1)^2 \sum_{i<j<k} F_{i,j,k}(x,x,x) \; + \; \ldots \; + \\ & (-1)^{n-1} F_{i,j,\ldots,n}(x, \cdots, x), \end{aligned} \tag{1}$$

where $F_{i,j}(x,x)$ is the joint CDF of random variables X_i and X_j. If X_i and X_j are independent, i.e. if $F_{i,j}(x,x) = F_i(x)F_j(x)$, Equation (1) reduces to

$$F_{min}(x) = 1 - \prod_{i=1}^{n} \{1 - F_i(x)\} \, . \tag{2}$$

Proof. This fact is well-known in statistics. The proof uses the inclusion-exclusion principle. A more detailed explanation can be found (for example) in [18, Proof of Corollary 2.70]. □

Theorem 1 is utilized in the following theorem, which is the main result presented in this section.

Theorem 2. ***(The Equivalent G/G/1 Model)*** *Assume cloning to a set of n servers using the same queuing discipline with service time distributions $F_{1:n}(x)$ with $x \geq 0$, that guarantee synchronized service. For all original requests r^o arriving with inter-arrival distribution $F_{arr}(y)$ with $y \geq 0$, the service time of the single request clone that completes service can be equivalently modeled using the distribution of the minimum value $F_{min}(x)$, determined according to Theorem 1. The server system with cloned requests then behaves equivalently to a G/G/1 server with inter-arrival distribution F_{arr} and service time distribution F_{min}.*

Proof. Each server s_i can be considered as a general and heterogeneous $G/G/1$ queue with inter-arrival distribution $F_{\text{arr}}(y)$ and service time distribution $F_i(x)$. Assume that there exists some $G/G/k$ server model with some inter-arrival distribution $F^{(s)}_{\text{arr}}(y)$ and service time distribution $F^{(s)}(x)$, that governs the response time of requests over the entire system. Synchronized service guarantees that all request clones $r^c_{1:n}$ of an original request r^o enter all servers simultaneously, and that the servers are kept in the same state. Thus the n servers can be seen as a single server of the same queuing discipline with $F^{(s)}_{\text{arr}}(y) = F_{\text{arr}}(y)$. This further implies that the shortest completion time for $r^c_{1:n}$ corresponds to the shortest service time for $r^c_{1:n}$, giving $F^{(s)}(x) = F_{\min}(x)$. Finally, the minimum of n draws from $F_{1:n}(x)$ distributions is equivalent to one draw from $F_{\min}(x)$, thus $k = 1$. □

Theorem 2 allows us to properly model and analyze the service time for server systems with cloned requests.

Remark 1. *Theorem 2 does **not** require any assumptions on properties of either the inter-arrival distribution F_{arr} or the service time distributions $F_{1:n}$. Furthermore, the theorem holds for any queuing discipline.*

Compared to previous research effort, the theorem extends the state of the art, in terms of the assumptions needed for its validity. In fact, previous research required to specify properties of either the inter-arrival distribution or the service time distributions. On the contrary, removing the need for these assumptions makes the theorem very general. Using Theorem 2, we define an equivalent G/G/1 model and by that can incorporate in our models both heterogeneity and dependencies across servers. Modeling dependencies across servers allows us to take into account things like the effect of database queries, that are the same no matter which machine is executing the query. We do, however, have to assume synchronized service which implies assumptions that might be unrealistic in practical implementations, such as perfect cancellations of clones. In Section 5, we study how our model is affected when the synchronized service assumption is relaxed.

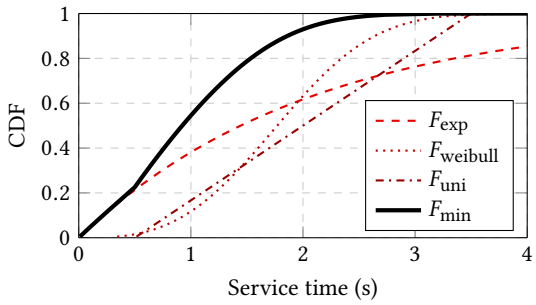


Figure 2: The service time CDFs for the Example with Heterogeneous Servers presented in Section 3.2.

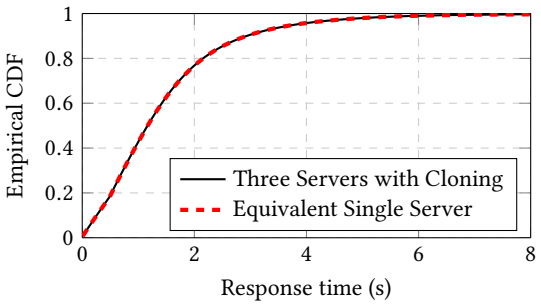


Figure 3: Empirical response time CDFs for the Example with Heterogeneous Servers presented in Section 3.2. Data retrieved through 20 repeated simulations of 10^6 requests each. The 95% confidence intervals lie within the lines.

3 EXAMPLES

In this section, we present two examples of how to use the model described in the previous section. In both examples, we assume synchronized service with n servers under PS.

3.1 Independent Exponential Distributions

The first example describes n heterogeneous servers $s_{1:n}$ whose service times behave according to the same distribution with different parameters. Specifically, we present results obtained with exponentially distributed inter-arrival times y with mean $1/\lambda$. Servers have exponentially distributed service times x with means $1/\mu_{1:n}$, where μ_i is the rate parameter used to describe the service time distribution of the i-th server. Here, we assume that the service time distributions are independent. This is the most common assumption made in all past research and the aim of this example is to verify that we can analytically obtain results covering the most commonly studied case. For example, using queuing theory, Gardner et al. [8] derived results about the distribution of service time of this setup and the FCFS queuing discipline. Here, we show that the same result also applies to any other queuing discipline.

The described setup implies following:

$$\begin{aligned} F_{\text{arr}}(y) &= 1 - e^{-\lambda y}, \\ F_i(x) &= 1 - e^{-\mu_i x}, \end{aligned} \tag{3}$$

with $x \geq 0, y \geq 0$. Using Theorem 2, we can model the synchronized service system composed of n servers as a single *equivalent* server having service time distribution $F_{\min}(x)$ as

$$F_{\min}(x) = 1 - \prod_{i=1}^{n} \{1 - F_i(x)\} = 1 - \prod_{i=1}^{n} e^{-\mu_i x} = 1 - e^{-\sum_{i=1}^{n} \mu_i x}. \tag{4}$$

The equivalent single server distribution $F_{\min}(x)$ is thus also exponential, with rate $\mu_{\text{tot}} = \sum_{i=1}^{n} \mu_i$. This means that the n server synchronized service system with cloned requests is *equivalent* to an M/M/1 server with arrival rate λ and service rate μ_{tot}. As anticipated, the expression derived in Equation (4) is the same presented in [8] for the FCFS queuing discipline. However, using Theorem 2 allows us to be more general and to show that the same result also holds for any queuing discipline, such as PS, assuming synchronized service.

3.2 Independent Heterogeneous Distributions

In the second example we want to show how to apply the results of Theorem 2 for the case of a synchronized service with n servers having independent and heterogeneous distributions, i.e., where the distribution type changes for each of the servers. We will present a practical example with $n = 3$, assuming that the inter-arrival times y are uniformly distributed between 0 s and 4 s,

$$F_{\text{arr}}(y) = \begin{cases} y/4 & \text{if } 0 \leq y \leq 4.0 \\ 1.0 & \text{if } y > 4.0 \end{cases}. \tag{5}$$

The service time distributions for $s_{1:3}$ are respectively an exponential, a Weibull, and a uniform distribution, with $x \geq 0$ and the following parameters.

$$\begin{aligned} F_1 &= F_{\text{exp}}(x) &&= 1 - e^{-0.480x} \\ F_2 &= F_{\text{weibull}}(x) &&= 1 - e^{-0.125x^3} \\ F_3 &= F_{\text{uni}}(x) &&= \begin{cases} 0 & \text{if } 0 \leq x < 0.5 \\ (x-0.5)/3.5 & \text{if } 0.5 \leq x \leq 3.5 \\ 1 & \text{if } x > 3.5 \end{cases} \end{aligned} \tag{6}$$

We choose these three distributions as they are typically used to model service times. Furthermore, the three distributions have different mean service times (relaxing the assumption of homogeneity, usually made in the literature).

Using Theorem 2, we can compute the equivalent single server service time distribution $F_{\min}(x)$ as follows.

$$\begin{aligned} F_{\min}(x) &= 1 - (1 - F_1)(1 - F_2)(1 - F_3) = \\ &1 - \{1 - F_{\text{exp}}(x)\} \{1 - F_{\text{weibull}}(x)\} \{1 - F_{\text{uni}}(x)\} \end{aligned}$$

The resulting equivalent model is a G/G/1 model with inter-arrival distribution F_{arr} and service time distribution $F_{\min}$. Figure 2 shows the service time distributions F_1, F_2, and F_3, together with $F_{\min}(x)$.

To demonstrate that the G/G/1 model is in fact equivalent to the cloned server system, we ran 20 simulations with 10^6 requests each, using the simulator described in Section 6. Figure 3 shows the empirical response time CDFs for this example when we simulate both the three servers with cloning case and the equivalent single server case, using the PS queuing discipline. The two response time CDFs are nearly identical, demonstrating the equivalence between the two models.

4 APPLICATIONS

Here, we use the equivalent G/G/1 model derived in Theorem 2 to analyze different systems under the PS discipline that fulfil the synchronized service criterion. The G/G/1 model is compliant with any inter-arrival process F_{arr} and service time distributions $F_{1:n}$, but in order to simplify the analysis, here we restrict ourselves to Poisson arrivals. For the service time distributions, we use the S&Z model, described in Section 4.1.

4.1 S&Z - A Service Time Model

Theorem 2 supports dependencies across service time distributions represented by joint CDFs. However, determining and analyzing these joint distributions is in general difficult. Gardner et al. [7] propose a model decoupling the task size of the original request r^o, denoted with Z^o, from the server slowdowns affecting clones $r^c_{1:n}$, which we indicate with $S^c_{1:n}$. In our paper, we use a multiplicative version, expressing the service time $X^c_{1:n}$ for clones $r^c_{1:n}$ as

$$X^c_{1:n} = Z^o \cdot S^c_{1:n}. \tag{7}$$

The idea behind this concept is to model the dependencies across servers $s_{1:n}$ that serve clones $r^c_{1:n}$ of the same original request r^o. As these cloned requests have identical task sizes, it is natural to include the shared task size Z^o in the service time model for the clones $r^c_{1:n}$.

The service time model in Equation (7) simplifies our analysis of dependent clones, as the server slowdowns $S_{1:n}$ can be viewed as *independent* across servers and original requests r^o. This allows us to use the simpler expression discussed in Theorem 1 – Equation (2) – when calculating the distribution of the minimum server slowdown $S_{\min}$ for each original request r^o. The complete minimum service time $X_{\min}$ for each r^o is defined as

$$X_{\min} = Z^o \cdot S_{\min}. \tag{8}$$

As shown in Equation (8), $X_{\min}$ belongs to a product distribution. Calculating this complete distribution is difficult, but its first moment can be determined according to:

$$\mathbf{E}[X_{\min}] = \mathbf{E}[Z^o] \cdot \mathbf{E}[S_{\min}]. \tag{9}$$

Exploiting the independence for $S_{1:n}$, we can use Equation (2) to determine the CDF of its minimum distribution $F^S_{\min}$ as

$$F^S_{\min} = 1 - \prod_{i=1}^{n} \left\{1 - F^S_i\right\}, \tag{10}$$

assuming F^S_i known for all S_i. Using $F^S_{\min}$, it is straightforward to determine the first moment of $S_{\min}$. We can then calculate the first moment of $X_{\min}$ using Equation (9), assuming that the distribution of the task size Z^o is known. This procedure is used in this section, where expressions for determining, e.g., average response time require this information.

The task sizes Z^o are modeled using a two-phase hyperexponential distribution with balanced means, using $\mathbf{E}[Z^o] = {}^1/_{4.7}$ and squared coefficient of variation $C^2_{Z^o} = 2$. For the server slowdowns S_i, we use the empirical Dolly distribution, with probability density function defined in Table 1 for the Dolly(1,12) case. First published in [1], and later on used in e.g. [7], the Dolly distribution is based on empirical data on server slowdowns from traces collected from Microsoft Bing's Dryad and Facebook's Hadoop clusters.

In Section 4.2, we derive the results for homogeneous servers, i.e. with server slowdowns S_i from the same Dolly(1,12) distribution.

4.2 Server Systems

Here, we present two server systems that both fulfill the synchronized service criterion. We investigate, using known expressions in queuing theory, how the cloning factor $c_f \in \mathbf{Z}^+$ (i.e., the number of clones for each request) affects performance and stability.

4.2.1 Clone-to-All. The simplest server system enabling synchronized service is where each request r^o is cloned and clones $r^c_{1:n}$ are sent to all n servers, i.e. with cloning factor $c_f = n$. This system is shown in Figure 1. Using the equivalent G/G/1 model presented in Section 2, we can represent the distribution $F_{\min}$ of service times $X_{\min}$ of this system according to the expression in Theorem 2. This enables us to calculate the mean response times $\mathbf{E}[T]$ of the cloned server system. We use the fact that the mean response time $\mathbf{E}[T^{M/G/1/PS}]$ for the PS queuing discipline only depends on the first moment of the service time distribution G. For our server system under PS, we can thus determine $\mathbf{E}[T^{M/G/1/PS}]$ as

$$\mathbf{E}[T^{M/G/1/PS}] = \frac{\mathbf{E}[X_{\min}]}{1 - \lambda\,\mathbf{E}[X_{\min}]}. \tag{11}$$

For stability, Equation (11) requires the utilization ρ of the cloned server system to be less than 1, thus we get

$$\rho = \lambda\,\mathbf{E}[X_{\min}] < 1. \tag{12}$$

Equations (11)–(12) allow us to exactly determine stability, utilization and mean response time of the Clone-to-All server system for any cloning factor c_f, any arrival rate λ and any service time distribution $F_{\min}$ from which we know the first moment.

Using service times X distributed according to the S&Z model with homogeneous server slowdowns, we can analytically retrieve the first moment of its equivalent service time distribution $X_{\min}$ as described in Section 4.1. By exhaustive search over λ and c_f, we can use Equation (11) to find the optimal cloning factors c_f^{opt} and the corresponding optimal mean response times $\mathbf{E}[T]^{\text{opt}}$, that minimize the mean response times of the server system.

Figure 4 shows an example with exact theoretical results for PS, assuming service times distributed according to the S&Z model described in Section 4.1. The dashed red lines show the optimal cloning factors c_f^{opt}, whereas the blue lines show the corresponding optimal mean response times $\mathbf{E}[T]^{\text{opt}}$. The comparison between cloning factors is performed such that the arrival rate per server is preserved, i.e. if a new server is added λ is scaled accordingly. As expected, higher cloning factors are more beneficial for lower

Table 1: The empirical Dolly(1,12) distribution from [1], used to model server slowdowns S.

S	1	2	3	4	5	6	7	8	9	10	11	12
Prob.	0.230	0.140	0.090	0.030	0.080	0.100	0.040	0.140	0.120	0.021	0.007	0.002

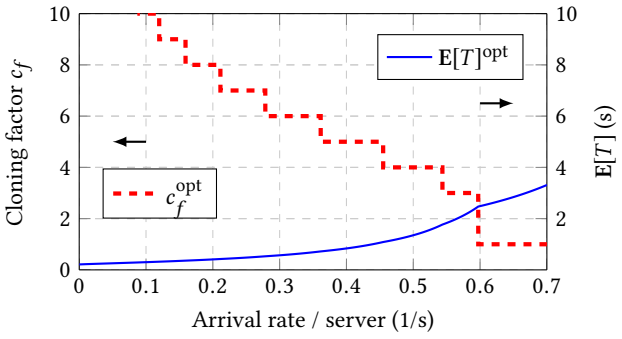


Figure 4: Clone-to-All: Optimal cloning factors together with the corresponding optimal mean response times. Data retrieved from theoretical analysis using Poisson arrivals and S&Z distributed service times.

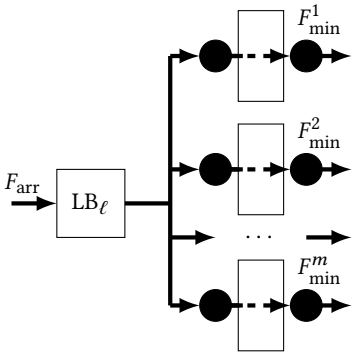


Figure 5: Clone-to-Clusters server system. Each rectangle represents a cluster of d servers that guarantees synchronized service to all clones $r^c_{1:d}$.

system loads since the clones utilize servers that otherwise would be in idle. A less expected result is that $c_f = 2$ is not optimal for *any* λ for this example. For high system loads, the service time dependencies introduced in the S&Z model, limit the use of cloning and for $\lambda > 0.6s^{-1}$ per server, no cloning ($c_f^{\text{opt}} = 1$) is optimal.

4.2.2 Clone-to-Clusters. The natural extension to only consider cloning to all n servers in the system is to allow for cloning to subsets of servers, as proposed in [14]. To simplify the analysis, we partition the set of n servers into m subsets of equal sizes, containing d servers each, i.e., such that $m \cdot d = n$. We denote these subsets as *clusters*. If an original request r^o is sent to a cluster it gets replicated to all d servers in the cluster. The clusters enable synchronized service for all request clones $r^c_{1:d}$, which means that each cluster can be equivalently represented as a G/G/1 model using Theorem 2.

This cloning strategy allows us to combine an arbitrary load-balancing strategy ℓ that decides to what cluster the original request r^o should be sent, together with the choice of cloning factor $c_f = d$. Figure 5 shows the complete system, that includes the load-balancing strategy ℓ and m clusters.

The choice of the cloning factor c_f depends on the load-balancing strategy ℓ, and we will now show that there is an advantage in conducting a joint design of the two. Given a load-balancing strategy ℓ, the cloning factor $c_f = d$ to be used in each cluster should be chosen such that the mean response time $\mathbf{E}[T]$ for the complete server system is minimized. Varying the value of the cloning factor c_f allows us to design and determine the statistical properties of the behavior of the m clusters (using n available servers), by using the equivalent G/G/1 modeling from Theorem 2 to each cluster separately. Using methods like exhaustive search, it is thus possible to determine the cloning factor c_f and the m clusters that minimize $\mathbf{E}[T]$ under ℓ. An important prerequisite for a successful co-design is that there exists a good enough (possibly approximate) expression for $\mathbf{E}[T]$ (or some other response time metric) when using the load-balancing strategy ℓ. If ℓ has a well-defined stability criterion, we can co-design a system with a provably stable cloning factor c_f. Here we restrict our analysis to two very common strategies, Random and Join-Shortest-Queue (JSQ).

The Random strategy distributes the original requests uniformly to the servers in the cluster, thus preserving the Poisson properties of the arrival rate λ towards the server system. Each cluster then receives the Poisson arrival rate λ/m. The exact analysis presented in Section 4.2.1 is directly applicable to the random load-balancing strategy, when deciding the optimal $c_f = d$. We denote this complete co-design as cluster-Redundancy-d (c-R-d), with d representing the cloning factor of each cluster.

The JSQ load balancer always selects the cluster with the shortest queue and sends the cloned requests to that cluster. To co-design the cloning factor with the JSQ strategy, we need to use approximations as no exact results exist for $\mathbf{E}[T]$. As we model our servers using the PS discipline, we utilize the approximation presented in [10]. Exploiting the near-insensitivity towards variability in service time distributions for JSQ under PS, it gives a very good approximation (error within 2-3%) for $\mathbf{E}[T]$, given Poisson arrivals, m clusters and the first moment of the service time distribution. This approximation is thus compatible with the S&Z model. Utilizing this approximation for $\mathbf{E}[T]$ thus allows us to find the optimal cloning factor c_f^{opt}, assuming that the approximations are accurate enough. We denote the complete co-design as cluster-Join-Shortest-Queue-d (c-JSQ-d), where d represents the cloning factor of each cluster.

4.2.3 Theoretical Co-design Example. To exemplify the co-design procedures of choosing the optimal c_f^{opt}, described in Section 4.2.2, we study the following example. Assume $n = 12$ servers, i.e. the cluster sizes $d = 1, 2, 3, 4, 6$ and 12 are available. We use Poisson arrivals and homogeneous service times distributed according to the S&Z model, and analyze the optimal cloning factors for each co-design using the exhaustive search method. We consider the queuing discipline PS, and investigate co-designs c-Redundancy-d and c-Join-Shortest-Queue-d. The theoretical results for five selected arrival rates are available in Table 2. In this particular example, the optimal cloning factors c_f^{opt} are equal for both co-designs for $\lambda = 0.30s^{-1}$ and $\lambda = 0.7s^{-1}$, whereas they are different for the other three arrival rates. As the co-design procedure implies optimization criteria that depend on ℓ, c_f^{opt} will in general also depend on ℓ. However, as the example shows, two different co-designs can of course also find the same c_f^{opt} under certain conditions.

Table 2: Theoretical analysis of the co-design example in Section 4.2.3, using Poisson arrivals and homogeneous service times distributions according to the S&Z model.

	$\lambda = 0.30$ / server (1/s)		$\lambda = 0.38$ / server (1/s)		$\lambda = 0.52$ / server (1/s)		$\lambda = 0.62$ / server (1/s)		$\lambda = 0.70$ / server (1/s)	
	c_f^{opt}	$\mathbf{E}[T]^{\text{opt}}$(s)	c_f^{opt}	$\mathbf{E}[T]^{\text{opt}}$(s)	c_f^{opt}	$\mathbf{E}[T]^{\text{opt}}$(s)	c_f^{opt}	$\mathbf{E}[T]^{\text{opt}}$(s)	c_f^{opt}	$\mathbf{E}[T]^{\text{opt}}$(s)
c-R-d	6	0.569	6	0.783	4	1.515	1	2.618	1	3.312
c-JSQ-d	6	0.404	4	0.482	3	0.692	2	0.955	1	1.112

For $\lambda = 0.7s^{-1}$ / server, c_f^{opt} is equal to 1 for both co-designs, implying that no cloning is optimal. The fact that we can compare no cloning ($c_f = 1$) to cloning ($c_f > 1$), using the same framework for both results, is powerful as it stops us from recommending cloning when it is not beneficial. Note that for c-JSQ-d the optimal cloning factors are based upon evaluating approximations for $\mathbf{E}[T]$, and their results are thus subject to the quality of the approximations.

5 NON-SYNCHRONIZED SERVICE

In this section we study the impact of relaxing the assumption of synchronized service, implying that the clones of an original request are no longer guaranteed to receive identical processor shares. First, we consider the effects of non-perfect arrivals and cancellations in the system. This is of high importance as it is most likely impossible to design a perfectly synchronized service in practice. Second, we relax the clone-to-cluster structure to allow for a more general cloning co-design approach, for which we cannot guarantee synchronized service.

In order to analyze our model in a non-synchronized context, the following definitions are needed. Consider a single, specific original request cloned over n servers. Let X_i, as before, be the stochastic variable (s.v.) associated with the service time distribution for server s_i. Denote with R_i the runtime of the cloned request on s_i, i.e., its time from service start to departure. Define θ as a vector of n inverse average processor shares for the clones during their runtime, i.e., $\theta_i \geq 1$ states how many requests are present at s_i in average during R_i. Further, let $N = \sum_{i=1}^{n} \theta_i$ be the total average amount of requests in the system during the runtimes of the clones of a specific original request.

The expected response time of the original request for a specific N can then be written as a function of θ as

$$\mathbf{E}[T|\theta] = \mathbf{E}[\min(\{\theta_j X_j\}_{j=1:n})]. \tag{13}$$

If one assumes that the service time distributions are homogeneous the following interesting result can be obtained.

Theorem 3. *The expected response time of an original request cloned to n servers at a specific N is maximized when all elements in θ are equal,*

$$\arg\max_{\theta} \mathbf{E}[T|\theta] = \theta^h, \quad \text{where } \theta^h = \{N/n\}_{j=1:n}.$$

Proof. Eq (13) can be rewritten using the Law of Total Expectation as

$$\sum_{k=1}^{n} \mathbf{E}[\min(\{\theta_j X_j\}_{j=1:n})|X_k \leq \forall X_i] \cdot \mathbf{P}(X_k \leq \forall X_i). \tag{14}$$

Since all X_i belong to the same distribution, $\mathbf{P}(X_k \leq \forall X_i) = 1/n$. Further, the minimum over a set is bounded by all of its members. In particular,

$$\begin{aligned} \mathbf{E}[\min(\{\theta_j X_j\}_{j=1:n})] &\leq \sum_{k=1}^{n} \theta_k \mathbf{E}[X_k|X_k \leq \forall X_i] \frac{1}{n} \\ &= \frac{N}{n} \mathbf{E}[\min(\{X_j\}_{j=1:n})] = \mathbf{E}\left[\min\left(\left\{\theta_j^h X_j\right\}_{j=1:n}\right)\right]. \end{aligned} \tag{15}$$

This proves the theorem. The homogeneous service time distributions yield that $\mathbf{E}[X_k|X_k \leq \forall X_i] = \mathbf{E}[\min(\{X_i\}_{i=1:n})]$ for each k. □

Theorem 3 holds under any value of N. For synchronized service, $\theta = \theta^h$ at all times, but in the non-synchronized case this is not true which makes Theorem 3 an important tool for comparison of the two cases.

5.1 Arrival and Cancellation Delays

In real settings, it is highly unlikely that perfect synchronization can be achieved. Instead, imperfections such as slightly different starting times for clones or latency differences between cancelling requests can occur. The imperfections can arise in two stages of the request handling, at arrival and at cancellation, which we model using the notion of *arrival delays* and *cancellation delays*.

Definition 3. *Let the arrival delay $a_i \geq 0$ be a s.v. representing the time difference between original request arrival and cloned request arrival on s_i. Further, let the cancellation delay $c_i \geq 0$ be a s.v. representing the time difference between the first completed cloned request on s_k and the departure (cancellation) on s_i.*

We will assume that the distributions of a_i and c_i are independent and homogeneous, we further assume that the service time distributions are homogeneous.

The presence of these imperfections becomes troublesome, as the clone-to-all system can no longer be guaranteed to be synchronized. No synchronization implies that the equivalent G/G/1 model can not be directly applied. It is, however, possible to derive a computable upper bound on the response time of the non-synchronized service. First, the following Lemma is stated.

Lemma 4. *Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two, possibly non-synchronized, clone-to-all systems with the same n and arrival rate. If for all N, $\mathbf{E}[T|N, \mathcal{S}_1] \leq \mathbf{E}[T|N, \mathcal{S}_2]$ and $\mathbf{E}[R_i|N, \mathcal{S}_1] \leq \mathbf{E}[R_i|N, \mathcal{S}_2]$, then*

$$\mathbf{E}[T|\mathcal{S}_1] \leq \mathbf{E}[T|\mathcal{S}_2]. \tag{16}$$

Proof. As $\mathbf{E}[R_i|N, \mathcal{S}_1] \leq \mathbf{E}[R_i|N, \mathcal{S}_2]$ is true for any N, then if $\mathcal{S}_1$ and $\mathcal{S}_2$ are subject to the same arrival rate, the expected number of requests present in the system must be smaller for $\mathcal{S}_1$ than $\mathcal{S}_2$, i.e.

$$\mathbf{E}[N|\mathcal{S}_1] \leq \mathbf{E}[N|\mathcal{S}_2] \tag{17}$$

Using the definition of the expected value, this inequality can be written as

$$\int Np(N|\mathcal{S}_1)dN \leq \int Np(N|\mathcal{S}_2)dN. \tag{18}$$

This inequality still holds if the function $g(N) = N$ is replaced by two functions that uphold the same inequality for all N, hence

$$\begin{aligned} \int \mathbf{E}[T|N,\mathcal{S}_1]p(N|\mathcal{S}_1)dN \leq \int \mathbf{E}[T|N,\mathcal{S}_2]p(N|\mathcal{S}_2)dN \\ \rightarrow \quad \mathbf{E}[T|\mathcal{S}_1] \leq \mathbf{E}[T|\mathcal{S}_2]. \end{aligned} \tag{19}$$

□

It is now possible to compute bounds on the effects of arrival and cancellation delays on the expected response time, by letting $\mathcal{S}_1$ be a clone-to-all system affected by delays and $\mathcal{S}_2$ a synchronized system such that $\mathcal{S}_1$ and $\mathcal{S}_2$ fulfills Lemma 4. Since $\mathcal{S}_2$ is synchronized, the equivalent G/G/1 model can be directly applied to explicitly computing a bound for $\mathcal{S}_1$. We proceed by first considering the two delays separately.

Theorem 5. ***(Arrival delays)*** *Let $\mathcal{S}_1$ be a clone-to-all system with arrival delay, and $\mathcal{S}_2$ an identical system but synchronized (without arrival delay). Let both systems be subjected to the same arrival rate. Then*

$$\mathbf{E}[T|\mathcal{S}_1] \leq \mathbf{E}[T|\mathcal{S}_2] + \mathbf{E}[a]. \tag{20}$$

Proof. Consider $\mathcal{S}_1$. For a specific N, the response time of an original request and the runtime of its clones becomes

$$\begin{aligned} T|\theta,\mathcal{S}_1 &= \min\{a_j + \theta_j X_j\}_{j=1:n}, \\ R_i|\theta,\mathcal{S}_1 &= \max(\min\{a_j + \theta_j X_j\}_{j=1:n} - a_i, 0). \end{aligned} \tag{21}$$

The maximum is introduced to prohibit negative R_i when the fastest clone completes before arrival at s_i. The maximum can be dealt with by introducing $b_i = \min(a_i, \min\{a_j + \theta_j X_j\}_{j=1:n})$. By definition, $b_i \leq a_i$ and further, $\min\{a_j + \theta_j X_j\}_{j=1:n} = \min\{b_j + \theta_j X_j\}_{j=1:n}$ as $\theta_j X_j \geq 0$. The following upper bound can then be created for the runtime

$$\begin{aligned} R_i|\theta,\mathcal{S}_1 &\leq \max(\min\{b_j + \theta_j X_j\}_{j=1:n} - b_i, 0) \\ &= \min\{b_j + \theta_j X_j\}_{j=1:n} - b_i. \end{aligned} \tag{22}$$

The expected runtime is bounded as

$$\mathbf{E}[R_i|\theta,\mathcal{S}_1] \leq \mathbf{E}\left[\min\{b_j + \theta_j X_j\}_{j=1:n}\right] - \mathbf{E}\left[b\right]. \tag{23}$$

Following the proof of Theorem 3, we can state that

$$\begin{aligned} &\mathbf{E}[\min(\{b_j + \theta_j X_j\}_{j=1:n})] \\ &\quad \leq \sum_{k=1}^{n} \left(\mathbf{E}[b_k + \theta_k X_k | X_k \leq \forall s_i]\right) \frac{1}{n} \\ &\quad = \mathbf{E}[b] + \mathbf{E}[\min\{\theta_j^h X_j\}_{j=1:n}], \end{aligned} \tag{24}$$

which gives

$$\begin{aligned} \mathbf{E}[R_i|\theta,\mathcal{S}_1] &\leq \mathbf{E}[\min\{\theta_j^h X_j\}_{j=1:n}] + \mathbf{E}[b] - \mathbf{E}[b] \\ &= \mathbf{E}[\min\{\theta_j^h X_j\}_{j=1:n}] = \mathbf{E}[R_i|\theta^h,\mathcal{S}_2]. \end{aligned} \tag{25}$$

Further, using Eq. (24) the response time for a request under a specific N can be bounded as

$$\begin{aligned} &\mathbf{E}[T|\theta,\mathcal{S}_1] = \mathbf{E}[\min\{a_j + \theta_j X_j\}_{j=1:n}] \\ &\leq \mathbf{E}[\min\{\theta_j^h X_j\}_{j=1:n}] + \mathbf{E}[a] = \mathbf{E}[T|\theta^h,\mathcal{S}_2] + \mathbf{E}[a]. \end{aligned} \tag{26}$$

As the two inequalities Eq. (25) and (26) hold for all $\theta \in \{\sum \theta_i = N,\ \theta_i \geq 1\ \forall i\}$, the statements can be conditioned on N instead without loosing the inequality property. Then using Lemma 4 the original statement is proven. □

Note that for Theorem 5, $\mathbf{E}[a]$ does not affect the stability of $\mathcal{S}_2$. Thus if $\mathcal{S}_2$ is stable, then so is $\mathcal{S}_1$ regardless of arrival delays.

Theorem 6. ***(Cancellation delays)*** *Let $\mathcal{S}_1$ be a clone-to-all system with cancellation delays, and $\mathcal{S}_2$ an identical system but without cancellation delays and thus synchronized and with service time $X_i|\mathcal{S}_2 = X_i|\mathcal{S}_1 + \mathbf{E}[c]$. Let both systems be subject to the same arrival rate. Then*

$$\mathbf{E}[T|\mathcal{S}_1] \leq \mathbf{E}[T|\mathcal{S}_2]. \tag{27}$$

Proof. Consider $\mathcal{S}_1$. For a specific N, the response time of an original request and the runtime of its clones become

$$\begin{aligned} T|\theta,\mathcal{S}_1 &= \min\{\theta_j X_j\}_{j=1:n}, \\ R_i|\theta,\mathcal{S}_1 &= \min\{\theta_j X_j\}_{j=1:n} + \min(c_i, \theta_i X_i - \min\{\theta_j X_j\}_{j=1:n}). \end{aligned} \tag{28}$$

The runtime incorporates the chance that a cloned request completes after $\min\{\theta_j X_j\}_{j=1:n}$ but before c_i has passed, thus the extra minimum. Further, the response time for a specific N is unaffected by the cancellation delay. As the response time is longer if the service time is longer, it can be trivially stated that

$$\mathbf{E}[T|\theta,\mathcal{S}_1] \leq \mathbf{E}[T|\theta^h,\mathcal{S}_2]. \tag{29}$$

The runtime for each server becomes bounded by the cancellation delay

$$R_i|\theta,\mathcal{S}_1 \leq \min\{\theta_j X_j\}_{j=1:n} + c_i. \tag{30}$$

The expected runtime for each server can thus be bounded as

$$\begin{aligned} \mathbf{E}[R_i|\theta,\mathcal{S}_1] &\leq \mathbf{E}[\min\{\theta_j X_j\}_{j=1:n} + c_i] \\ &\leq \mathbf{E}[\min\{\theta_j (X_j + \mathbf{E}[c])\}_{j=1:n}] \\ &\leq \mathbf{E}[R_i|\theta^h,\mathcal{S}_2], \end{aligned} \tag{31}$$

where Theorem 3 has been used in the last step. As the two inequalities Eq. (29) and (31) hold for all $\theta \in \{\sum \theta_i = N,\ \theta_i \geq 1\ \forall i\}$, the statements can be conditioned on N instead without loosing the inequality property. Then using Lemma 4 the original statement is proven. □

Note that for Theorem 6, for large $\mathbf{E}[c]$ the upper bound can become infinite despite potential stability of $\mathcal{S}_1$. Thus the arrival rate of the system has to be less than $1/(\mathbf{E}[X_i] + \mathbf{E}[c])$ to guarantee stability for both $\mathcal{S}_1$ and $\mathcal{S}_2$.

The following Theorem shows that the effect of both arrival and cancellation delays can be bounded by the sum of the individual bounds.

Theorem 7. *Let $\mathcal{S}_1$ be a clone-to-all system with both arrival and cancellation delays, and $\mathcal{S}_2$ an identical system but without delays and thus synchronized and $X_i|\mathcal{S}_2 = X_i|\mathcal{S}_1 + \mathbf{E}[c]$. Let both systems be subject to the same arrival rate. Then*

$$\mathbf{E}[T|\mathcal{S}_1] \leq \mathbf{E}[a] + \mathbf{E}[T|\mathcal{S}_2]. \tag{32}$$

Proof. Consider $\mathcal{S}_1$. For a specific N, the response time is still unaffected by the cancellation delay and the runtime for each clone becomes a clear combination of the two separate delay cases,

$$\begin{aligned} T|\theta,\mathcal{S}_1 &= \min\{a_j+\theta_j X_j\}_{j=1:n}, \\ R_i|\theta,\mathcal{S}_1 &= \max\Big(\min\{a_j+\theta_j X_j\}_{j=1:n}-a_i \\ &\quad +\min(c_i, a_i+\theta_i X_i-\min\{a_j+\theta_j X_j\}_{j=1:n}),0\Big). \end{aligned} \tag{33}$$

Following the proofs to Theorems 5 and 6 with the previous two equations, it is easy to see that the two bounds are additive. □

The benefit of Theorems 5-7 is two fold. First, they show that small imperfections are not detrimental when trying to implement synchronized service in practice. Further, the bounds are computable given that the expected response time of the equivalent G/G/1 model is computable, which gives a way of making informed decisions in capacity planning of such systems. However, the Theorems are only strictly valid if one assumes that a_i, c_i, X_i are homogeneous and known, which is not the case for all systems.

5.2 Clone-to-Any

The co-design procedure denoted c-ℓ-d in Section 4.2.2, assuming a clone-to-clusters structure, is of interest as it provides a way to compute and quantify the performance of such systems. The design itself is, however, not that intuitive as it is superfluous to pre-partition the servers into clusters. In practice, a more natural approach would instead be to allow the load-balancing strategy ℓ to, for each original request, choose c_f unique servers from $s_{1:n}$ to clone to. We define this as the *clone-to-any* cloning strategy, and denote co-designs under clone-to-any as a-ℓ-d for cloning factor $c_f = d$ and load-balancing strategy ℓ.

For a-ℓ-d co-designs, synchronized service is no longer guaranteed which implies that the equivalent G/G/1 model is not directly applicable. A measure that quantifies this non-perfect synchronization is the *clone error* ϵ, defined for clones $r^c_{1:d}$ to an original request r^o as $\epsilon = \mathbf{D}[p^c_{1:d}]/\mathbf{E}[p^c_{1:d}]$, with $\mathbf{D}$ the standard deviation and $p^c_{1:d}$ the processor shares for $r^c_{1:d}$. For $\mathbf{E}[\epsilon] > 0$, the system is non-synchronized, for $\mathbf{E}[\epsilon] = 0$ it is synchronized and for small $\mathbf{E}[\epsilon]$ we have *near-synchronization*.

It is intuitive to believe that c-ℓ-d could be used to form an adequate approximation of a-ℓ-d. In fact, the less utilization ρ a system under a-ℓ-d is subject to, the more similar to c-ℓ-d it becomes. This is formalized in the following theorem.

Theorem 8. *Consider the two server systems $\mathcal{S}_1$ and $\mathcal{S}_2$, where $\mathcal{S}_1$ uses a-ℓ-d and $\mathcal{S}_2$ c-ℓ-d but otherwise are identical.*

$$\text{Then as } \rho \to 0, \quad \mathbf{E}[T|\mathcal{S}_1] \to \mathbf{E}[T|\mathcal{S}_2]. \tag{34}$$

Proof. The smaller the utilization, the larger the probability that all clones $r^c_{1:d}$ to an original request r^o execute alone on their servers. As $\rho \to 0$ then processor shares $p^c_i \to 1$ for all r^c_i. If $p^c_i = 1$ for all r^c_i, then $\mathbf{E}[\epsilon] = 0$ and the clones are synchronized. □

As small $\mathbf{E}[\epsilon]$ implies near-synchronization, it is using c-ℓ-d possible to derive an accurate approximation for the mean response time of a-ℓ-d under low loads, for any ℓ and c_f. For more general system loads ρ, the similarity between a-ℓ-d and c-ℓ-d depends on the choice of ℓ. In particular, if ℓ is good at keeping p_i similar for all clones to the same original request, a-ℓ-d will behave close to synchronized as $\mathbf{E}[\epsilon]$ will be small. In Section 6, the load-balancing strategies random and JSQ, with clone-to-any co-designs denoted a-R-d and a-JSQ-d, are compared to their c-ℓ-d counterparts.

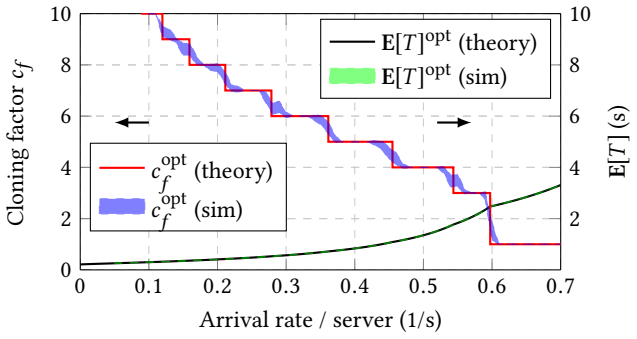


Figure 6: Clone-to-All: Comparison of theoretical values with 95% confidence intervals for the simulation results.

6 EVALUATION

In this section, we demonstrate and evaluate the examples and claims stated in the two previous sections, using our own discrete-event simulator[1]. We refrained from using existing simulators like CloudSim [4], because our evaluation requires us to simulate the cloud application-level behavior and not only the infrastructure behavior. For this reason, we took inspiration from the brownout [15] simulator[2], available online, and modified it to remove the adaptation layer and added cloning functionality. In the simulator we include the options to define: (i) the inter-arrival time distribution $F_{\text{arr}}(x)$, (ii) the service time distributions $F_{1:n}(x)$ for our n servers, (iii) the cloning factor c_f, (iv) the load balancing strategy and (v) arrival and cancellation delays. The arrival and service time distributions can be heterogeneous and we allow the user to set them based on empirical CDF data.

All simulations in this section are run using the PS discipline.

6.1 Server Systems

All experiments in this subsection are evaluated over 20 independent simulations per scenario with unique random seeds, each with 10^6 incoming requests from Poisson arrivals. The service time distribution is the S&Z model from Section 4.1.

6.1.1 Clone-to-All. The clone-to-all system in Section 4.2.1, for which the G/G/1 model yields an exact analysis for c_f^{opt} and $\mathbf{E}[T]^{\text{opt}}$, was simulated with a sweep over the arrival rates. The 95% confidence intervals for the results of the simulations are shown together with the theoretical values in Figure 6. As can clearly be seen, the simulated c_f^{opt} and $\mathbf{E}[T]^{\text{opt}}$ follow their theoretical values closely.

6.1.2 Theoretical Co-designs. We further evaluate the co-designs presented in Section 4.2.3 using the simulator. The results are shown as 95% confidence intervals in Figure 7, plotted together with the theoretical values for both co-designs c-R-d and c-JSQ-d. For the

[1] https://github.com/tomminylander/cloning-simulator
[2] https://github.com/cloud-control/brownout-simulator

optimal clone factor c_f^{opt}, the simulated and theoretical values match perfectly. The same applies for the matching of $\mathbf{E}[T]$, at least for c-R-d where the theoretical values are obtained via exact analysis. In fact, the simulated c-JSQ-d mean response times are slightly (1-3%) off compared to the theoretical values. The reason is that the JSQ values are based on (highly accurate) approximations. As our co-designs succeed in finding all optimal cloning factors c_f^{opt}, the simulations suggest that the co-designs perform well even when the accuracies of the involved approximations of $\mathbf{E}[T]$ are not perfect.

6.2 Non-Synchronized Service

In order to perform a general evaluation, all experiments in this subsection are evaluated over 1000 randomized scenarios with unique random seeds, each with 10^6 incoming requests from Poisson arrivals. We use the following service time distributions: (i) S&Z model from Section 4.1, (ii) Exponential ($\mu = 1$), (iii) Weibull (shape=0.5, scale=0.5), (iv) Pareto (Type 1, shape=2.5, scale=0.6) and (v) Uniform ($X_i \in [0, 2]$). The mean service time $\mathbf{E}[X]$ of all the above distributions at cloning factor $c_f = 1$ is 1s. The utilizations considered are $\rho^{sim} = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]$. For each scenario, we randomly select a service time distribution from (i)-(v), a utilization from ρ^{sim}, a number of servers from s_n^{sim} and a cloning factor from c_f^{sim}. The two latter are defined below.

6.2.1 Arrival and Cancellation Delays. First, we evaluate the theoretical bounds from Section 5.1 for clone-to-all server systems. We use $c_f^{sim} = s_n^{sim} = [2, 3, 4, 5, 6, 7, 9, 10]$. Additionally, from $delay^{sim} = [0.01, 0.02, 0.05, 0.1, 0.2, 0.5] \cdot E[X]$, we randomly select a normalized mean delay. We run three separate experiments: (a) arrival delays only, (b) cancellation delays only and (c) both delays present. In the latter experiment, the mean delays for each scenario are chosen such that, for $0 \leq \gamma \leq 1$ uniformly random, $\gamma\mathbf{E}[a] + (1-\gamma)\mathbf{E}[c]$ becomes the chosen delay from $delay^{sim}$. The results are available in Figure 8, and show that for low normalized delays (0.01-0.05) all normalized $\mathbf{E}[T]$ are close to 1. This implies that the synchronized model describes these cases accurately. Further, for the low delays the bounds are tight in all three experiments, which implies that the bounds are very useful for these delays. However, for the higher delays the normalized $\mathbf{E}[T]$ is larger. Also, the bounds for the cancellation delay become very large, and for some scenarios with delay 0.2 and 0.5 they even become infinite as the bound can not guarantee stability for these cases. As none of our simulated scenarios were unstable, it is obvious that the cancellation delay bound has limited usage for these higher delays.

6.2.2 Clone-to-Any. Second, we compare the clone-to-any co-designs with their synchronized counterparts. Here we set $s_n^{sim} = [4, 6, 9, 12, 15, 21, 27, 30, 45, 48]$ and randomly choose c_f, such that for each scenario the chosen number of servers is evenly divisible by c_f.

Figure 9 shows the results for both random and JSQ. The upper plot shows the mean clone errors $\mathbf{E}[\epsilon]$, and it can clearly be seen that for a-JSQ-d the values are much smaller than for a-R-d. This implies that a-JSQ-d is a much better approximation of its synchronized counterpart than a-R-d. The lower plot, showing normalized $\mathbf{E}[T]$, confirms this statement as the values for a-JSQ-d are much closer to 1. For low ρ, both co-designs approximate the synchronized behavior well in accordance with Theorem 8. For a-JSQ-d, the

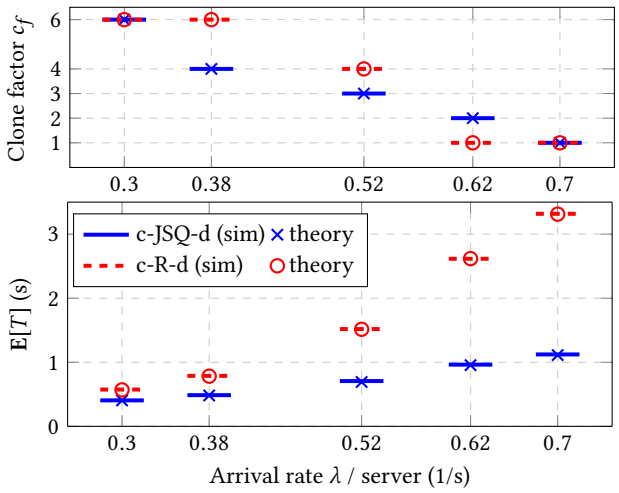


Figure 7: Co-designs: Comparison of theoretical values with 95% confidence intervals for the simulation results. The legend applies to both figures.

normalized $\mathbf{E}[T]$ are very close to 1 for all utilizations suggesting that we have near-synchronized service regardless of the arrival rate. This property can be intuitively explained by looking into the JSQ algorithm. As JSQ always sends the clones to the servers with the least amount of running requests, this will cause the servers to always have very similar amounts of running requests. The clones $r_{1:d}^c$ of the same original request r^o will then always receive very similar processor shares, leading to small mean clone errors $\mathbf{E}[\epsilon]$.

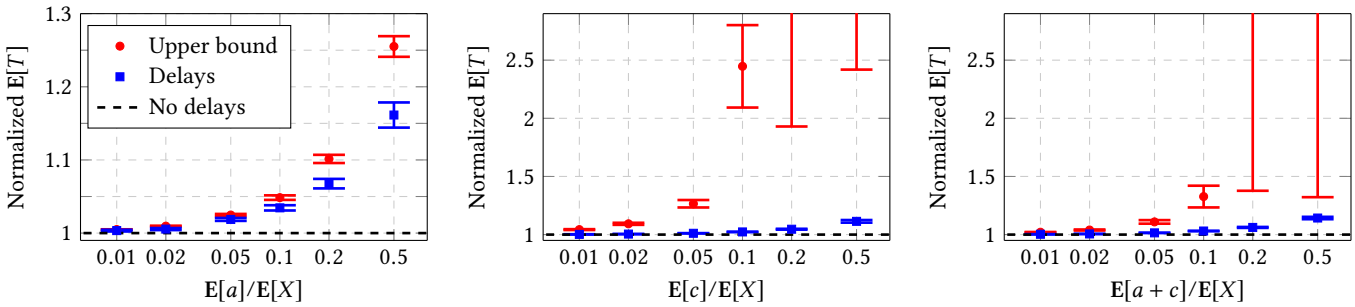
Looking more closely at the normalized $\mathbf{E}[T]$, it can be observed that the values for a-JSQ-d and a-R-d never exceed 1. As our simulation study is fairly general, considering many different parameters, this suggests that the mean response times for the synchronized c-ℓ-d co-design might actually form an upper bound for the a-ℓ-d counterparts. This claim is partially supported by Theorem 3, which can be read as that synchronized service in fact always is worse than non-synchronized. However, we have not been able to complete the proof to hold for complete co-designs, and it will thus have to be left for future work.

6.3 Summary

Our simulation campaign shows good compliance with our theoretical findings. For both the clone-to-all plots in Figure 6, and the co-design plot in Figure 7, our model predicts the optimal cloning factors c_f^{opt} with high accuracy. Figure 8 shows that, especially for low arrival and cancellation delays, our theoretical bounds can be used to predict the effect of practical imperfections on our model. Figure 9 shows the interesting near-synchronized service property of the JSQ policy, suggesting that our model could accurately describe setups involving the JSQ load-balancer, where synchronized service is not guaranteed.

7 RELATED WORK

Cloning has been studied in the research literature, although in most of the cases previous studies were limited to exponential



(a) Arrival delays only. **(b) Cancellation delays only.** **(c) Both delays present.**

Figure 8: Arrival and cancellation delay simulations. The normalization of E[T] is performed such that each value is divided by the theoretical value without delays. The intervals represent 95% confidence intervals. The legend applies to all figures.

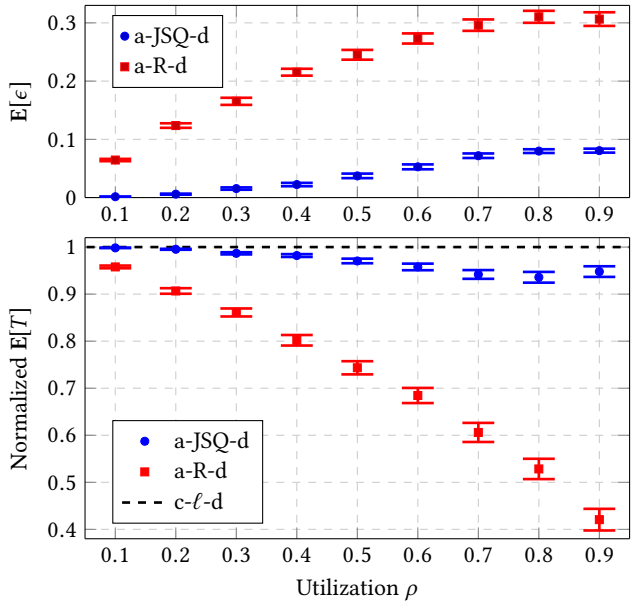


Figure 9: Simulations comparing a-ℓ-d to c-ℓ-d for random and JSQ. The normalization of E[T] is performed such that each value is divided by the value for the c-ℓ-d counterpart. The intervals represent 95% confidence intervals.

distributions for service times and the FCFS discipline?. We briefly presented an overview of the cloning literature in the introduction of this paper. Here we provide additional details and comparisons with the most related research contributions.

In contrast to pure cloning, speculative execution [2, 24] has previously been studied to remedy the effects of slow executing tasks in large data frameworks such as MapReduce [6] or Spark [23]. Using speculative execution, the infrastructure keeps track of request handling progress and launches copies of slow tasks to reduce the total execution time. As explained by Ganesh et al. [1], cloning can be viewed as an extreme case of speculative execution with no speculation time.

Restricting the arrival and service time distributions to being exponential and the queuing discipline to FCFS, the method presented in [8] is able to handle cases that are not covered by the synchronized service definition, e.g., to simultaneously handle both cloning and non-cloning request classes. However, as a result of Theorem 2, here we show that the result presented in [8] is valid also — in the case of server systems with synchronized service — for other queuing disciplines, including for example PS.

Gardner et al. [7] propose a service time model decoupling task sizes from server slowdowns, conveniently allowing for modeling of dependencies between request clones. We use this model throughout our paper (denoting it as the S&Z model), and show that using our G/G/1 modeling concept, its statistical properties under cloning can be analyzed even under PS. In the same paper [7], the authors propose the cloning strategy Redundant-to-Idle-Queue (RIQ), a policy that clones requests to all idle servers that it finds. The RIQ strategy is both provably stable and analytically tractable within the S&Z model, but cannot detect scenarios where cloning actually deteriorates performance. Our co-design procedure presented in Section 4 is, on the contrary, able to identify scenarios where cloning is not beneficial and should be avoided. In these scenarios, the optimal cloning factor is equal to 1.

It is possible to determine guidelines for cloning factors for service time distributions with specific properties [21]. The results presented in [21] are applicable to arbitrary arrival processes, and examples where cloning is beneficial include independent identically distributed memoryless service time distributions. In our paper, we go beyond this and utilize the G/G/1 modeling in Section 2 to determine optimal cloning factors for any service time distributions. However, we do require server systems to guarantee synchronized service.

In an attempt to make cloning models more realistic (closer to real implementations), Lee et al. [17] worked on modeling and analyzing the overhead of request cancellations, and how these can affect the optimal scheduling policy. We also consider practical imperfections in our paper, but using a different approach focused on studying the accuracy of our synchronized model when subjected to e.g. arrival and cancellation delays.

Joshi et al. [14] propose that an (n,1) fork-join system can be equivalently represented by an M/G/1 queue, under an i.i.d assumption for service time distributions. Utilizing Theorem 1 and 2, we show that, under synchronized service, a server system subject to cloned requests can equivalently be represented by a G/G/1 model, without any assumptions on either inter-arrival or service time distribution.

A group-based random cloning policy is presented in [14], that roughly corresponds to our cluster co-design with ℓ as the random load-balancing algorithm. Our G/G/1 modeling holds for any queuing discipline, allowing us to present and analyze a wider class of cloning co-designs. Specifically, we are able to co-design the JSQ policy together with the cloning factor for the PS discipline. Joshi et al. [14] show that for log-convex tail distributions, cloning to all n servers is optimal even in the heavy traffic regime. Additionally, the paper also takes the computing cost into account when deciding cloning factors, which we do not consider in our paper.

8 CONCLUSION

This paper presented a theoretical analysis that extends and generalizes known results about request cloning in data centers. We used the concept of synchronized service to denote a certain number of servers that simultaneously serve clones of a request. We demonstrated that request cloning in a server system under synchronized service can equivalently be modeled as a G/G/1 server. We showed that *no* assumptions on either inter-arrival or service time distributions are required, and that the G/G/1 model holds for *any* queuing discipline. In this paper, we focused on the PS discipline and show further results for it.

We further extended our theoretical results and discussed the optimal cloning factor. We also analyzed more complex server systems, consisting of multiple clusters. To demonstrate the possible applications of the equivalent G/G/1 modeling, we presented a co-design method that, under homogeneity assumptions, found the optimal cloning factor c_f^{opt} and the server system's corresponding mean response time $\mathbf{E}(T)$ under both random and JSQ load-balancing for clusters with synchronized service. To the best of our knowledge, this paper presents the first provably stable combined load-balancing and cloning strategy for the PS queuing discipline. Further, we relaxed the synchronized service assumption and derived bounds for how practical imperfections, such as arrival and cancellation delays, affect the accuracy of our model. We demonstrated using simulations that removing the synchronized service constraint for the JSQ co-design seems to only marginally reduce the accuracy of the model. We provided an intuitive explanation to this phenomenon, which implies that our theoretical model could be used to design the non-synchronized a-JSQ-d version as well.

REFERENCES

[1] Ganesh Ananthanarayanan, Ali Ghodsi, Scott Shenker, and Ion Stoica. 2013. Effective Straggler Mitigation: Attack of the Clones. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation (nsdi'13)*. USENIX Association, Berkeley, CA, USA, 185–198. http://dl.acm.org/citation.cfm?id=2482626.2482645

[2] Ganesh Ananthanarayanan, Srikanth Kandula, Albert Greenberg, Ion Stoica, Yi Lu, Bikas Saha, and Edward Harris. 2010. Reining in the Outliers in Map-reduce Clusters Using Mantri. In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation (OSDI'10)*. USENIX Association, Berkeley, CA, USA, 265–278. http://dl.acm.org/citation.cfm?id=1924943.1924962

[3] Urtzi Ayesta. 2019. On redundancy-d with cancel-on-start a.k.a Join-shortest-work (d). *ACM SIGMETRICS Performance Evaluation Review* 46, 2 (jan 2019), 24–26. https://doi.org/10.1145/3305218.3305228

[4] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose, and Rajkumar Buyya. 2011. CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. *Softw. Pract. Exper.* 41, 1 (2011), 23–50.

[5] Jeffrey Dean and Luiz André Barroso. 2013. The tail at scale. *Commun. ACM* 56, 2 (feb 2013), 74. https://doi.org/10.1145/2408776.2408794

[6] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: Simplified Data Processing on Large Clusters. *Commun. ACM* 51, 1 (Jan. 2008), 107–113. https://doi.org/10.1145/1327452.1327492

[7] Kristen Gardner, Mor Harchol-Balter, and Alan Scheller-Wolf. 2016. A Better Model for Job Redundancy: Decoupling Server Slowdown and Job Size. In *2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE. https://doi.org/10.1109/mascots.2016.43

[8] Kristen Gardner, Samuel Zbarsky, Sherwin Doroudi, Mor Harchol-Balter, and Esa Hyytia. 2015. Reducing Latency via Redundant Requests: Exact Analysis. *ACM SIGMETRICS Performance Evaluation Review* 43, 1 (jun 2015), 347–360. https://doi.org/10.1145/2796314.2745873

[9] Kristen Gardner, Samuel Zbarsky, Mor Harchol-Balter, and Alan Scheller-Wolf. 2016. The Power of d Choices for Redundancy. *ACM SIGMETRICS Performance Evaluation Review* 44, 1 (jun 2016), 409–410. https://doi.org/10.1145/2964791.2901497

[10] Varun Gupta, Mor Harchol Balter, Karl Sigman, and Ward Whitt. 2007. Analysis of join-the-shortest-queue routing for web server farms. *Performance Evaluation* 64, 9-12 (oct 2007), 1062–1081. https://doi.org/10.1016/j.peva.2007.06.012

[11] Gauri Joshi. 2018. Synergy via Redundancy: Boosting Service Capacity with Adaptive Replication. *ACM SIGMETRICS Performance Evaluation Review* 45, 2 (mar 2018), 21–28. https://doi.org/10.1145/3199524.3199530

[12] Gauri Joshi, Yanpei Liu, and Emina Soljanin. 2012. Coding for Fast Content Download. *CoRR* abs/1210.3012 (2012). arXiv:1210.3012 http://arxiv.org/abs/1210.3012

[13] Gauri Joshi, Emina Soljanin, and Gregory Wornell. 2015. Efficient replication of queued tasks for latency reduction in cloud systems. In *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE. https://doi.org/10.1109/allerton.2015.7446992

[14] Gauri Joshi, Emina Soljanin, and Gregory Wornell. 2017. Efficient Redundancy Techniques for Latency Reduction in Cloud Systems. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems* 2, 2 (apr 2017), 1–30. https://doi.org/10.1145/3055281

[15] Cristian Klein, Martina Maggio, Karl-Erik Årzén, and Francisco Hernández-Rodriguez. 2014. Brownout: Building More Robust Cloud Applications. In *Proceedings of the 36th International Conference on Software Engineering (ICSE 2014)*. 700–711.

[16] Leonard Kleinrock. 1975. *Queueing Systems. Vol. I: Theory.* Wiley Interscience.

[17] Kangwook Lee, Ramtin Pedarsani, and Kannan Ramchandran. 2017. On Scheduling Redundant Requests With Cancellation Overheads. *IEEE/ACM Transactions on Networking* 25, 2 (apr 2017), 1279–1290. https://doi.org/10.1109/tnet.2016.2622248

[18] Giuseppe Modica and Laura Poggiolini. 2012. *A First Course in Probability and Markov Chains.* John Wiley & Sons, Ltd. https://doi.org/10.1002/9781118477793

[19] Zhan Qiu, Juan F. Pérez, and Peter G. Harrison. 2016. Tackling Latency via Replication in Distributed Systems. In *Proceedings of the 7th ACM/SPEC on International Conference on Performance Engineering - ICPE '16*. ACM Press. https://doi.org/10.1145/2851553.2851562

[20] N. B. Shah, K. Lee, and K. Ramchandran. 2014. The MDS queue: Analysing the latency performance of erasure codes. In *2014 IEEE International Symposium on Information Theory*. 861–865. https://doi.org/10.1109/ISIT.2014.6874955

[21] Nihar B. Shah, Kangwook Lee, and Kannan Ramchandran. 2016. When Do Redundant Requests Reduce Latency? *IEEE Transactions on Communications* 64, 2 (feb 2016), 715–722. https://doi.org/10.1109/tcomm.2015.2506161

[22] Huajin Wang, Jianhui Li, Zhihong Shen, and Yuanchun Zhou. 2018. Approximations and Bounds for (n, k) Fork-Join Queues: A Linear Transformation Approach. In *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. IEEE. https://doi.org/10.1109/ccgrid.2018.00069

[23] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauly, Michael J. Franklin, Scott Shenker, and Ion Stoica. 2012. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing. In *Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*. USENIX, San Jose, CA, 15–28. https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/zaharia

[24] Matei Zaharia, Andy Konwinski, Anthony D. Joseph, Randy Katz, and Ion Stoica. 2008. Improving MapReduce Performance in Heterogeneous Environments. In *8th USENIX Conference on Operating Systems Design and Implementation (OSDI'08)*. 29–42.