Download Time Analysis for Distributed Storage Codes With Locality and Availability

Mehmet Fatih Aktaş, Swanand Kadhe, *Member, IEEE*, Emina Soljanin, *Fellow, IEEE*,
and Alex Sprintson, *Senior Member, IEEE*

***Abstract*— The paper presents techniques for analyzing the expected download time in distributed storage systems that employ systematic availability codes. These codes provide access to hot data through the systematic server containing the object and multiple recovery groups. When a request for an object is received, it can be replicated (forked) to the systematic server and all recovery groups. We first consider the low-traffic regime and present the close-form expression for the download time. By comparison across systems with availability, maximum distance separable (MDS), and replication codes, we demonstrate that availability codes can reduce download time in some settings but are not always optimal. In the high-traffic regime, the system contains multiple inter-dependent Fork-Join queues, making exact analysis intractable. Accordingly, we present upper and lower bounds on the download time, and an M/G/1 queue approximation for several cases of interest. Via extensive numerical simulations, we evaluate our bounds and demonstrate that the M/G/1 queue approximation has a high degree of accuracy.**

***Index Terms*— Distributed coded storage, availability, download with redundancy.**

I. Introduction

DISTRIBUTED systems implement reliable storage despite failures by storing data objects redundantly across multiple servers. Replication is traditionally preferred for its simplicity. Maximum Distance Separable (MDS) codes are used when replication is costly. Even though MDS codes maximize storage efficiency, they incur large communication overhead during object recovery. This has motivated coding theorists to look for novel erasure codes that are *recovery-efficient*, see, e.g., [3]–[5]. An important class of such codes is Locally Recoverable Codes (LRCs) [4], [5]. LRCs enable recovery of an object by accessing only a small group of servers, referred to as a *recovery group*. The number of servers in the largest recovery group is referred to as the code *locality*.

A special class of LRCs, known as *availability codes*, have an additional property that each object has multiple, disjoint recovery groups [6]–[9]. Separate recovery groups allow simultaneous download of the same object by multiple users. For instance, consider the binary Simplex code that encodes $\{f_1, f_2, f_3\}$ into $[f_1,\ f_2,\ f_3,\ f_1+f_2,\ f_1+f_3,\ f_2+f_3,\ f_1+f_2+f_3]$. This code is said to have *availability* three as each of f_1, f_2 and f_3 has three disjoint recovery groups. For example, f_1 can be recovered by reading both f_2 and f_1+f_2, or by reading both f_3 and f_1+f_3, or by reading f_2+f_3 and $f_1+f_2+f_3$. This code furthermore has *locality* two as each recovery group consists of at most two servers.

The notion of code availability was proposed with the goal of making the stored data more accessible (see, e.g., [7]). Having multiple disjoint recovery groups for an object enables high data availability since each recovery group provides an additional way to retrieve the object. Thus, availability codes make it possible to assign multiple requests for the same object to different servers without blocking any request. Consequently, availability codes have a significant potential to provide low-latency access for *hot data*, i.e., objects that are frequently and simultaneously accessed by multiple users [7].

It is important to quantify to what extent availability codes can reduce the download latency as compared to MDS and replication codes. This is because accessing an object through one of its recovery groups requires downloading one object from each of the recovery servers. A retrieval from a recovery group is therefore complete once the objects from *all* the servers in the recovery group are fetched. This means that the retrieval is slow even if the service is slow at only one of the servers. In fact, service times in modern large-scale systems are known to exhibit significant variability [10], [11], and thus, the download time from a recovery group can be significantly slower than that from a single server. As an example, if service times at the servers are independent and exponentially distributed, mean time to download from a recovery group of size r will scale (approximately) by $\ln r$ for large r. Motivated by this practical challenge of service time variability, this paper presents techniques for analyzing the download latency of availability codes by leveraging tools from queuing theory.

Manuscript received September 20, 2020; revised December 21, 2020 and March 2, 2021; accepted March 3, 2021. Date of publication March 18, 2021; date of current version June 16, 2021. This material is based upon work supported by the NSF under Grant No. CIF-1717314 and CIF-1718658. The work of Alex Sprintson (while serving at NSF) was supported by the NSF. This article was presented in part at the ISIT'15 and in part at the Sigmetrics'17. The associate editor coordinating the review of this article and approving it for publication was V. Stankovic. *(Corresponding author: Mehmet Fatih Aktaş.)*

Mehmet Fatih Aktaş and Emina Soljanin are with Department of ECE, Rutgers University, NJ 08854 USA (e-mail: mfatihaktas@gmail.com; emina.soljanin@rutgers.edu).

Swanand Kadhe is with Department of EECS, UC Berkeley, Berkeley, CA 94720 USA (e-mail: swnanand.kadhe@berkeley.edu).

Alex Sprintson is with Department of ECE, Texas A&M University, College Station, TX 77843 USA.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCOMM.2021.3067054.

Digital Object Identifier 10.1109/TCOMM.2021.3067054

A. Contributions and Organization of the Paper

We present techniques for analyzing the download time of individual data objects that are jointly encoded with a systematic availability code. We assume the Fork-Join (FJ) access strategy, under which requests are *forked* upon arrival to the *systematic* server containing the requested object and all its recovery groups. While other access schemes are possible, we focus our attention to the FJ strategy for the following reasons. First, request service time at a systematic server is typically smaller than that at a recovery group. Thus, a user whose request is assigned to a recovery group would experience larger latency. FJ strategy treats all the requests uniformly, resulting in a form of fairness. Second, FJ strategy is widely adopted for download from coded storage systems (see, e.g., [12]–[15]).

We consider two arrival regimes: (i) *low-traffic regime* where each request is served before the new request arrives; (ii) *queuing regime* where requests can possibly overlap at the storage servers, which serve them through a first-in first-out queue. First, for the low-traffic regime, we derive closed-form expressions for the distribution and expected value of download time for availability codes (Theorem 1), along with replication (Lemma 1) and MDS codes (Lemma 2). This enables us to compare availability codes with state-of-the-art erasure codes used in production systems, in particular the MDS code used in the Google file system and the locally recoverable code used in Windows Azure storage. Next, for the queuing regime, we observe that the system consists of multiple inter-dependent Fork-Join queues. This results in the infamous state space explosion problem, which makes it intractable to perform an exact analysis. Therefore, we establish upper and lower bounds on the download time.

Our key idea is to consider *special cases* of our system by carefully imposing restrictions, and use download times of these restricted models to find the bounds. In particular, we consider the following two restricted models. (1) Fork-Join Fixed Arrival (FJ-FA) model, in which every request arrival asks for the same object. This will hold for instance if the object popularities exhibit extreme skew, that is when the probability that a request asks for a particular object is one while it is zero for the rest of the objects. (2) Fork-Join Split-Merge (FJ-SM): in which requests are buffered upon arrival in a centralized First-come First-serve (FCFS) queue, and are fed to the system one by one only when all the servers are idle. Leveraging these two models, we find lower and upper bounds on the performance of our Fork-Join system (Theorem 2 and Theorem 3). In addition, via numerical simulations, we compare the expected download time for availability codes with with various practical coding schemes.

In addition to the bounds, we also propose an M/G/1 queue approximation for FJ-FA systems with locality two. We then proceed by refining our approximation for systems with availability one. In our analysis, we build on techniques from Markov processes and Renewal theory. Combination of the techniques we present is useful to study another type of Fork-Join system that is defined by an $(n, 2)$ MDS code [16].

B. Related Work

Literature on download from coded storage has focused on downloading the *complete set of data objects* that are jointly encoded with an MDS code. This is an important research question and has been studied extensively, see *e.g.* [12], [15], [17]–[21] and references therein. This paper differs from this literature in two important aspects. First, we are concerned with downloading individual objects. In practice, users are typically interested in only a subset of the stored data, namely *hot data*. This leads to skews in object popularities as shown by traces collected from production systems [22]. Our model for data access incorporates the skewed object popularities and the notion of hot data. Second, we focus on storage systems with availability codes. LRCs with availability have recently replaced MDS codes in production, *e.g.* [23], [24]. It is important to understand the download performance when these new codes are at use.

The paper is organized as follows: Section II explains the storage and data access model. Section III analyzes the download time under low-traffic regime. Section IV presents bounds on the download time under queuing regime. Section V analyzes the FJ-FA and FJ-SM models to set up the background for the proofs of our bounds, and the proofs are presented in Section VI. Section VII presents an M/G/1 approximation for the FJ-FA system. In Section VII-B and VII-C, we consider the special cases with locality two and availability one, and obtain close approximations for the average download time for the FJ-FA model. We skip some of the technical details in several proofs in Section VII due to the space limitations. For the omitted details, we refer the reader to [25].

II. System Model

A. Data Storage Model

We consider a system that encodes k equal sized objects $\mathbb{F} = \{f_1, \dots, f_k\}$ into n objects by an (n, k) linear systematic erasure code, and then stores the n encoded objects across n servers. Each object f_i is a symbol in some finite field $\mathbb{F}$. The *storage overhead* of an (n, k) linear code is defined as the inverse of the code rate, i.e., n/k.

We are interested in a special class of erasure codes for which each systematic server has (r, t)-*availability* [6]–[9]. A code is said to have (r, t)-availability if it ensures that failure of any systematic server can be recovered using one of the t disjoint *recovery groups* of size r, where typically $r \ll k$. We denote such an LRC as an (n, k, r, t)-LRC or an (n, k, r, t) availability code. We denote the lth recovery group for the ith server as $\mathbb{R}_i^l$, where $|\mathbb{R}_i^l| = r$. The recovery group size r is referred to as the code *locality*. The (r, t)-availability allows retrieving an object f_i in $t+1$ ways: either by downloading it from the systematic server or by downloading all the symbols in one of its t recovery groups. In the example given in Section I, we used the $(n = 7, k = 3)$ binary Simplex code that encodes $\{f_1, f_2, f_3\}$ into $[f_1,\ f_2,\ f_3,\ f_1 + f_2,\ f_1 + f_3,\ f_2 + f_3,\ f_1 + f_2 + f_3]$. This code has $(r = 2, t = 3)$-availability. Object f_1 can be accessed either at the systematic server or by recovering it from any of the three pairs of coded symbols

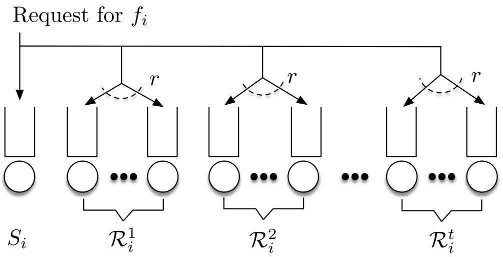

Fig. 1. Fork-Join access model for (n,k,r,t)-LRC. Each request is split upon arrival into its systematic server and each of its t recovery groups.

$\{f_2, f_1+f_2\}$, $\{f_3, f_1+f_3\}$, $\{f_2+f_3, f_1+f_2+f_3\}$. Similarly for f_2 or f_3, there are one systematic server and three recovery groups.

B. Content Access Model

We consider the Fork-Join (FJ) model for data access. FJ model for a system with an (r,t)-availability code consists of two levels of FJ queues. In the first level, each request upon arrival is *forked* (replicated) into $t+1$ copies, which are sent to the systematic server that stores the requested object and its t recovery groups. The request completes as soon as either its copy at the systematic server or a copy at any one of its recovery groups finishes service. Once a request is completed, all its outstanding copies are immediately removed from the system. The second level of FJ queues is formed at each recovery group. A request copy that is assigned to a recovery group is forked into r *sub-copies* and each enters the queue at its respective server. A request copy at a recovery group finishes when all its r forked sub-copies finish service and *join*. (See Fig. 1 for an illustration.) We assume cancellation of the outstanding request copies (either in queue or in service), or reconstructing an object from the coded copies incurs negligible delay in comparison with the overall data access latency. Indeed, it is observed in practice that the latency of reconstructing an object is several orders of magnitude smaller than the overall data access latency [23].

C. Request Arrival and Service Models

For tractability, request arrival process is assumed to be Poisson with fixed rate λ. We model object popularities as follows: each request arrival asks for object f_i independently with probability p_i for $i=1,\dots,k$, where $0 \le p_i \le 1$ and $\sum_{i=1}^k p_i = 1$. We denote each request-j as a pair (t_j, o_j), where t_j is the arrival time and o_j is the requested object. We consider two arrival regimes:

Low-traffic regime in which the system completes serving a request before the next one arrives, and can thus contain at most one request at any time. This case serves as a starting point for the download time analysis, and it is also useful to understand the download performance under low offered load.

Queuing regime, in which subsequent requests might arrive before the current request(s) is finished. In this case, system resources needs to be shared across the requests, and multiple requests sent to a storage server are queued at the server. We model resource sharing at the servers with FCFS queues.

Request copies are served at the servers by reading and streaming the requested object. We model this by a random *service time*. For tractability, service times are assumed to be independently and identically distributed (i.i.d.) across different requests and servers as Exponential random variables (r.v.'s) of rate μ. Note that, as argued in [20], the shifted-exponential distribution is a good model of service times in data centers. However, the exponential distribution is analytically tractable because it is memoryless, and is commonly used to model service times in queuing theoretic analysis (see, *e.g.*, [12], [17], [18], [26]). Insights derived from this study can serve as a stepping stone towards understanding the system performance under general service time distributions.

Our focus is on analyzing the *download time* T for an individual object, which essentially is the response time (or sojourn time) of a request. In particular, we define a request's download time as $T = t_{\text{departure}} - t_{\text{arrival}}$, where t_{arrival} is the time the request arrives to system and $t_{\text{departure}}$ is the time the request download is completed. Since we consider multiple coding schemes and variants of the FJ system, we typically denote the download time in system x with r.v. T_x.

Notation: S denotes an Exponential random variable (r.v.) with rate μ, i.e, $S \sim \text{Exp}(\mu)$. For r.v.'s X and Y, $X \le Y$ if and only if $\text{P}\{X > s\} \le \text{P}\{Y > s\}$ for all s. Notation $X_{n:i}$ denotes the ith smallest of n i.i.d. samples of a r.v. X. The beta function is given by $\beta(x,y) = \int_0^1 v^{x-1}(1-v)^{y-1}\text{d}v$.

III. Download Time Under Low-Traffic Regime

A. Download Time Characterization for Availability Codes

In the following, we characterize the download time $T_{\text{FJ-}(r,t)}$ in the FJ system implemented by an (r,t)-availability code.

Theorem 1: Under the low-traffic regime,

$$P\left\{T_{FJ\text{-}(r,t)} > s\right\} = \exp(-\mu s)\left(1-(1-\exp(-\mu s))^r\right)^t,$$
$$\mathbb{E}\left[T_{FJ\text{-}(r,t)}\right] = \frac{1}{\mu r}\beta(t+1, 1/r).$$

Proof: $T_{\text{FJ-}(r,t)} = \min\{S, R_1, \dots, R_t\}$ where S is the service time at the systematic server and R_i is the service time at the ith recovery group. R_i are independent and distributed as $S_{r:r}$, since the request copy at a recovery group finishes service once all its copies at the r recovery servers finish service. We have

$$\begin{aligned}\text{P}\left\{T_{\text{FJ-}(r,t)} > s\right\} &= \text{P}\{\min\{S, R_1, \dots, R_t\} > s\} \\ &\overset{(a)}{=} \text{P}\{S > s\}\left(1-\text{P}\{S \le s\}^r\right)^t \\ &\overset{(b)}{=} \exp(-\mu s)\left(1-(1-\exp(-\mu s))^r\right)^t \quad (1)\end{aligned}$$

where (a) follows from the independence of the service times and (b) comes from substituting $\text{P}\{S > s\} = \exp(-\mu s)$.

Observing that $T_{\text{FJ-}(r,t)}$ is a non-negative r.v., we obtain

$$\begin{aligned}\mathbb{E}\left[T_{\text{FJ-}(r,t)}\right] &= \int_0^\infty \text{P}\left\{T_{\text{FJ-}(r,t)} > s\right\}\text{d}s \\ &\overset{(a)}{=} \frac{1}{\mu r}\int_0^1 v^t(1-v)^{(1/r-1)}\text{d}v \overset{(b)}{=} \frac{1}{\mu r}\beta\Big(t+1, \frac{1}{r}\Big) \quad (2)\end{aligned}$$

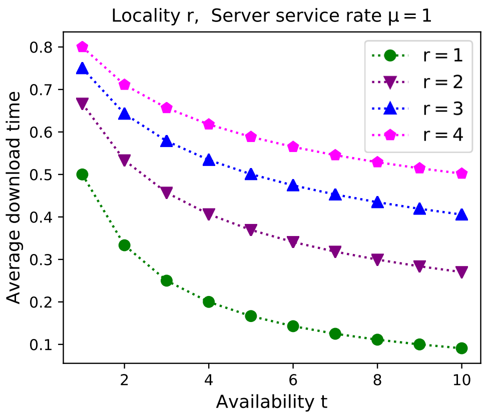

Fig. 2. Average download time under the low-traffic regime with respect to code locality r and availability t.

where (a) comes from (1) and $1-(1-\exp(-\mu s))^r = v$, and (b) follows from the definition of the beta function. ■

Remark 1: Expression for $\mathbb{E}\left[T_{FJ\text{-}(r,t)}\right]$ *allows us to examine the effect t on the average download time while fixing r. Using the equality* $\beta(x, y+1) = \frac{y}{x+y}\beta(x,y)$, *it is straightforward to verify that the relative reduction in average download time per increment in t is given by* $(r(t+1)+1)^{-1}$, *which is approximately* $1/rt$ *for large t and r. Thus incrementing t yields diminishing returns in reducing the average download time. This can be observed in Fig. 2, which plots the average download time with respect to r and t. Observe that increasing the locality r significantly slows down the download.*

1) Comparison With Replication Codes: Replication schemes are an important contestant of availability codes as they are commonly deployed in production, see *e.g.*, [27]. We here consider a t_{rep}-replication code that stores t_{rep} copies for each of the k objects. For a fair comparison, we assume the cumulative service rate (sum of the service rates across all the servers) is fixed. The proof of the following lemma follows from the order statistics, similar to the proof of Theorem 1.

Lemma 1: In a system with t_{rep}*-replication code and service times i.i.d. as* $\text{Exp}(\mu)$, *the average download time under the low-traffic regime is* $\mathbb{E}\left[T_{FJ\text{-}t_{\text{rep}}}\right] = 1/(t_{\text{rep}}\mu)$. *If the cumulative service rate of the system is the same as that of a system with an* (n,k,r,t)*-LRC, then* $\mathbb{E}\left[T_{FJ\text{-}t_{\text{rep}}}\right] = k/(n\mu)$.

Remark 2: When the cumulative service rate is equal to that of an (n,k) *code, the average download time in the* t_{rep}*-replication system depends only on* k/n, *and not on* t_{rep}.

2) Comparison With MDS Codes: An (n,k) code is said to be MDS if any k of the n coded objects are sufficient to reconstruct the original k objects. Recent work have shown that MDS codes are faster than replication in downloading the entire set of k objects, see *e.g.*, [12]. It is therefore natural to evaluate MDS codes in downloading the individual objects.

In the FJ access model for MDS coded storage, each request is replicated upon arrival into n copies, which are then assigned across all the n servers. A request is completed once either its copy at the systematic server finishes service or any k out of the remaining $n-1$ recovery servers jointly finish serving the request. The download time $T_{\text{FJ-}(n,k)}$ is given by $\min\{S,\ S_{(n-1):k}\}$. This allows us to derive the average download time under the low-traffic regime as follows. The proof follows from the order statistics, similar to Theorem 1.

TABLE I

COMPARISON OF THREE COMMERCIAL CODES AND AN AVAILABILITY CODE BASED ON THREE PERFORMANCE METRICS

Erasure Code	$\mathbb{E}[T]\,\mu$	Storage Overhead	Fault Tolerance
3-replication	0.33 (0.67)	3×	2
$(9,6)$-MDS code	0.67 (0.67)	1.5×	3
$(10,6,3,1)$-LRC	0.6 (0.83)	1.5×	3
$(14,6,2,3)$-LRC	0.45 (0.71)	2.33×	3

Lemma 2: In a system with an (n,k)*-MDS code and service times i.i.d. as* $\text{Exp}(\mu)$, *the expected download time under the low-traffic regime is* $\mathbb{E}\left[T_{FJ\text{-}(n,k)}\right] = k/(n\mu)$.

Remark 3: The average download time in an (n,k)*-MDS coded system depends only on* k/n, *and is the same as that in a* t_{rep}*-replication coded system when the cumulative mean service rates of both the systems are the same.*

B. Performance Comparison of Erasure Codes

We here compare availability codes with the following state-of-the-art erasure codes: (i) 3-replication, which is commonly used in many distributed storage systems, *e.g.*, Amazon's Dynamo [27]; (ii) $(9,6)$-MDS code, which is used in Google file system [28]; (iii) $(10,6,2,1)$-LRC, which is used in Windows Azure Storage [23]; and (iv) $(14,6,2,3)$-availability code obtained as a direct sum of two $(7,3,2,3)$-Simplex codes. Binary Simplex code, which is a *dual* of the Hamming code, forms a well-known class of availability codes with locality two [29], and a direct sum of simplex codes yields codes with availability [30]. We here adopt the same data access and request service model introduced in Section II. This means requests are replicated to all servers upon arrival, and the request service times are i.i.d. as $\text{Exp}(\mu)$. We use Lemma 2 and Theorem 1 to compute the average download time for the systems with the erasure codes listed above.

In order to get a holistic performance comparison, we consider the following metrics in addition to average download time: (i) storage overhead and (ii) fault tolerance. In particular, it is well-known that a coding scheme that encodes k objects across n nodes with minimum distance d can tolerate any $d-1$ node failures. We refer to $d-1$ as the fault tolerance of the coded system.[1] We emphasize that, even though the storage overhead and fault tolerance have been known to be important metrics in evaluating real-world distributed storage systems (see, *e.g.*, [23], [24], [28]), the critical metric of (average) download time is missing in the literature.

We present a detailed comparison in Table I. The $(14,6,2,3)$-LRC with $(2,3)$ availability achieves smaller

[1] More generally, reliability analysis can be done in terms of mean time to data loss (MTTDL) using standard techniques, see e.g., [23], [24], [28]. It is well-known that MTTDL grows exponentially with the minimum distance. For simplicity, we focus on the fault-tolerance instead of MTTDL.

average download time as compared to the state-of-the-art erasure codes, at the expense of worse storage overhead. On the other hand, its storage overhead is smaller compared to 3-replication, at the cost of slightly worse download time. Indeed, it achieves a favorable trade-off between the storage overhead and the average download time, with large fault-tolerance. Availability codes are therefore attractive for storing hot data that requires small download latency. On the other hand, when the cumulative service rate in the system is kept fixed, $(9, 6)$-MDS code achieves the smallest download time as well as the smallest storage overhead, with high fault-tolerance. Hence, for storage systems with limited cumulative service rate, MDS codes are favorable candidates.

IV. Download Time Under Queuing Regime

In this section we remove the assumption that at most one request is present in the system at any time. In this case, multiple requests might be simultaneously present in the system, hence queues might build up at the servers. In order to highlight the structure of the queues in our system, let us first review the notion of an (ℓ, m)-Fork-Join (FJ) queue [12]. In an (ℓ, m)-FJ queue, each request is forked into ℓ copies upon arrival, and a request completes once any m out of its ℓ copies finish service. Once a request completes, its remaining $\ell - m$ copies are immediately removed from the system.

In the system that uses availability codes, there are two levels of FJ queues present under the FJ access model. In the *outer* level, each arriving request is replicated into $t+1$ copies, which are then sent to the systematic server and its t recovery groups. A request completes as soon as any one of its copies finishes service, hence forming a $(t + 1, 1)$-FJ queue. In the *inner* level, request copies sent to a recovery group fork into r sub-copies and complete once all the forked copies finish service. Thus, each recovery group acts as an (r, r)-FJ queue. Note that FJ queues at the recovery groups are not independent due to the cancellation of outstanding request copies.

Analysis of the FJ queues is a long standing open problem. Its average response time is known only for the simplest case of $(2, 2)$-FJ queue [31], [32]. Moreover, the inner FJ queues implemented by the recovery groups in our system are inter-dependent. This inter-dependence makes it intractable to exactly analyse our system. Our approach to understand the system performance is therefore deriving bounds and approximations on the download time.

A. Bounds on Download Time

We here present bounds on the download time by considering more restricted counterparts of our system. These restrictions are carefully imposed on the system to make it possible to find exact or approximate expressions.

1) Fork-Join General Arrival (FJ-GA): We refer to our availability coded system under the Fork-Join access model and object popularity probabilities $(p_1, \ldots, p_k)$ as the Fork-Join General-Arrival (FJ-GA) system.

2) Fork-Join Fixed Arrival (FJ-FA): In this restrictive model, every request arrival within a busy period asks for the same object. A busy period refers to the time interval during which there is at least one request in the system. This will hold for instance if the object popularities exhibit extreme skew, i.e., when requests ask for a particular object with probability one. In general, the FJ-FA model is useful to understand the performance of the FJ model under highly skewed object popularities, which is typical in practice [22].

3) Fork-Join Split-Merge (FJ-SM): In this restrictive model, requests are buffered in a centralized FCFS queue, and are sent to service one by one only when all the servers are idle.

Recall that each download request-i is of the form (t_i, o_i) where t_i is the arrival time and o_i is the requested object. The download time refers to the steady state system response time, i.e., the time spent by an individual request in the system. In the following, we denote the download time in system x under an aggregate arrival rate of λ as $T_{x,\lambda}$. We avoid writing λ explicitly when the arrival rate is clear from the context.

Theorem 2: Download time distributions for the FJ-GA, FJ-FA, FJ-SM systems satisfy the following inequalities:

$$\begin{aligned} P\left\{T_{\textit{FJ-GA},\lambda} > t\right\} &\overset{(a)}{\geq} \sum_{i=1}^{k} p_i \, P\left\{T_{\textit{FJ-FA},\lambda p_i} > t\right\} \\ &\overset{(b)}{\geq} \sum_{i=1}^{k} p_i \, \exp\left(-\left((t+1)\mu - p_i\lambda\right) t\right), \quad (3) \end{aligned}$$

$$\begin{aligned} P\left\{T_{\textit{FJ-GA},\lambda} > t\right\} &\overset{(c)}{\leq} P\left\{T_{\textit{FJ-SM},\lambda} > t\right\}, \\ P\left\{T_{\textit{FJ-FA},\lambda} > t\right\} &\overset{(d)}{\leq} P\left\{T_{\textit{FJ-SM},\lambda} > t\right\}. \quad (4) \end{aligned}$$

Lemma 3: When code locality is two, $P\{T_{\textit{FJ-GA},\lambda} > t\} \leq P\{T_{\textit{FJ-FA},\lambda} > t\}$ for all λ and t.

Remark 4: It is possible to interpret the lower bound (a) as the response time in a system of k separate FJ-FA's, which are fed by splitting the Poisson arrival process across with the probabilities $p_1, \ldots, p_k$. In fact, this lower bound is derived using this interpretation as outlined later in Section VI. In this system of k FJ-FA's, minimum response time is achieved when the load is perfectly balanced across the FJ-FA's, that is when $p_i = 1/k$ for $i = 1, \ldots, k$. The response time will increase $(p_1, \ldots, p_k)$ becomes skewed. This means the lower bound (a) will take its minimum possible value when $p_i = 1/k$ for $i = 1, \ldots, k$. The value of the lower bound will increase as $(p_1, \ldots, p_k)$ becomes skewed, e.g., when $p_1 > p_i$ for $i = 2, \ldots, k$. Lemma 3 together with the upper bounds (c) and (d) says that, when the code locality is two, the lower bound (a) becomes equal to the upper bound (c) when the object popularities are polarized, i.e., when every arrival asks for the same object. Then, when code locality is two, the upper bound (c) suggests that the performance of FJ-GA becomes worst when the object popularities are highly skewed. The bounds in Theorem 2 become looser as the request arrival rate gets larger. Figure 9 shows this for the system with availability one and locality two. For this reason, we also find an approximation for the download time in Section VII.

Theorem 2 gives us the following bounds on the average download time in the FJ-GA system. We note that the bounds on λ in the theorem statement are due to stability constraints.

Theorem 3: The average download time in the FJ-GA system $\mathbb{E}[T_{\textit{FJ-GA}}]$ is bounded as follows.

$$\mathbb{E}[T_{\textit{FJ-GA}}] \geq \sum_{i=1}^{k} \frac{p_i}{(t+1)\mu - p_i\lambda},$$

$$\mathbb{E}[T_{\textit{FJ-GA}}] \leq \eta + \frac{\lambda \sum_{j=0}^{t}\binom{t}{j}(-1)^j \sum_{l=0}^{rj}(-1)^l\binom{rj}{l}(l+1)^{-2}}{\mu^2(1-\lambda\eta)}, \tag{5}$$

where $\eta = \frac{1}{\mu r}\beta\Big(t+1, \frac{1}{r}\Big)$. *The lower bound holds for* $\lambda < (t+1)\mu/\max\{p_1, \dots, p_k\}$, *and the upper bound, for* $\lambda < 1/\eta$.

We defer proving Theorem 2 and Theorem 3 to Section VI after we establish the characteristics of the FJ-FA and FJ-SM systems in Section V-A and V-B, which are used in the proofs.

B. Simulation Results

Fig. 3 shows the average download time vs. request arrival rate for the coding schemes given in Table I. Note that all the four coding schemes used here store 6 objects. However the number of servers n dictated by the code varies across the schemes. For a fair comparison, we also plot results when the cumulative service rate is fixed at 10, which is evenly allocated across the servers. For instance, server service rate is 1 in the system with $(10, 6, 3, 1)$-LRC while it is $10/14$ in the system with $(14, 6, 2, 3)$-Availability code. For high arrival rates, these systems are ordered in terms of their average download time as $(18, 6)$-Replication $<$ $(14, 6, 2, 3)$-Availability $<$ $(10, 6, 3, 1)$-LRC $<$ $(9, 6)$-MDS. Notice that this is the same ordering given by their storage efficiency. This means availability codes in the Fork-Join access model serve as an intermediate point in the storage vs. download time tradeoff between the two extremes: MDS and replication codes. This observation holds regardless of the skews in object popularities.

V. Variants of the Fork-Join System

A. Fork-Join System in the Fixed-Arrival Model (FJ-FA)

Recall that, in the FJ-FA model, all request arrivals within a busy period ask for the same object. Therefore, each server in the system acts either as the systematic server or as a recovery server within the same busy period. Further, only one server takes the role of systematic server in the busy period.

Leading and Slow Servers: A request copy is forked into r sibling servers once it arrives to a recovery group. Some of the servers within the recovery group can be ahead of their siblings in service at any given time. We refer to such servers as *leading*, and to those behind with service as *slow* servers. For instance, in the example shown in Fig. 6 (in Section VII-C), the server that hosts $a+b$ is ahead of its sibling (the one that hosts b) by one request copy. In this case, we say the server hosting $a+b$ is leading while the server hosting b is slow.

Observe that up to $1 + r(t-1)$ requests can be served simultaneously, and different copies of the same request can start service at different times. These properties significantly complicate the system analysis. In order to address this challenge, we redefine the request service start times as follows.

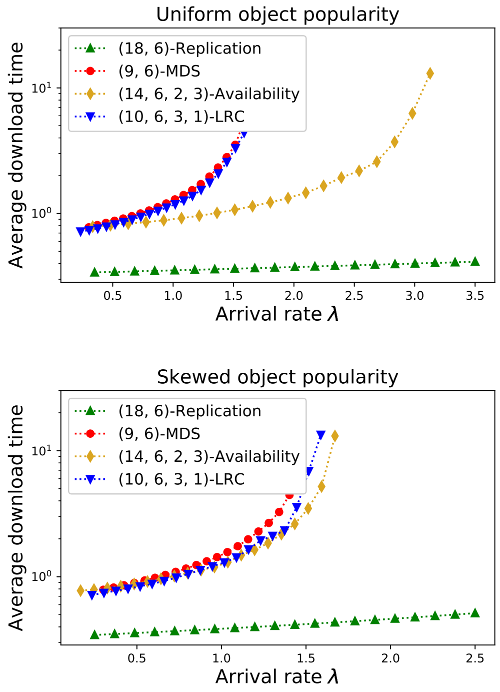

Fig. 3. Average download time versus λ for the coding schemes in Table I. At the top, all objects have equal popularity. At the bottom, $1/3$ of the objects have 90% of the popularity and the rest share 10% of popularity equally, e.g., $p_1 = p_2 = 0.45$, and $p_3 = p_4 = p_5 = 0.05$ for $(14, 6, 2, 3)$-availability.

Definition 1: We say that a request is at the head of the line (HoL) once all its copies remaining in the system are in service, and the request starts service once it moves to HoL.

Under this definition, we have the following observation.

Observation 1: In the FJ-FA system, requests depart the system in the order they arrive and there can be at most one request in service at any time.

The first part follows from the fact that request copies depart in the order of arrival both at the systematic server and at any recovery group. To see the second part, note that, for two requests to be simultaneously in service, all remaining copies of each must be in service simultaneously, which is impossible given that requests depart in the first-in first-out order.

At each recovery group, a request can have up to $r-1$ of its copies depart before the request moves to HoL. We refer to these as *early departing copies*. If d copies of a request depart early at a recovery group, service time of the request copy at that recovery group will be distributed as $S_{r-d:r-d}$. More generally, let d_i denote, for a request, the number of its early departing copies at the ith recovery group. Then the service time for the request, once it moves to HoL, is given as $S_{d_1,\dots,d_t} = \min\{S,\ S_{r-d_1:r-d_1},\ \dots,\ S_{r-d_t:r-d_t}\}$. For a multiset $\{d_1, \dots, d_t\}$ and each $d_i \in \{0, 1, \dots, r-1\}$, let ν_d denote the number of occurrences of d in the multiset. Given that the recovery groups are indistinguishable, $S_{d_1,\dots,d_t}$ is solely determined by the vector $\boldsymbol{\nu} = (\nu_0, \nu_1, \dots, \nu_{r-1})$, and we say in this case that the request has *type-*$\boldsymbol{\nu}$ service time, and denote it as $S_{\boldsymbol{\nu}}$. The set of all possible $\boldsymbol{\nu}$ is given by

$$\mathbb{N} = \Big\{(\nu_0, \nu_1, \dots, \nu_{r-1}) \mid \sum_{0}^{r-1} \nu_i = t,\ \nu_i \in \mathbb{Z}_{\geq 0}, \Big\} \tag{6}$$

where $\mathbb{Z}_{\geq 0}$ is the set of non-negative integers. We formalize these observations in the following lemma.

Lemma 4: In the FJ-FA system, the set of all possible request service time distributions is $\mathbb{S} = \{S_{\boldsymbol{\nu}} \mid \boldsymbol{\nu} \in \mathbb{N}\}$*, where* $\mathbb{N}$ *is given in* (6) *and* $|\mathbb{S}| = |\mathbb{N}| = \binom{t+r-1}{r-1}$*. The distribution function for the type-*$\boldsymbol{\nu}$ *service time is*

$$P\{S_{\boldsymbol{\nu}} > s\} = \exp(-\mu s) \prod_{d=0}^{r-1} \left(1 - (1 - \exp(-\mu s))^{r-d}\right)^{\nu_d}.$$

Observe from the expression of $\mathrm{P}\{S_{\boldsymbol{\nu}} > s\}$ that the more copies of a request depart early (before the request moves to HoL), the faster its service will be. Recall that $\sum_i \nu_i = t$ for any $\boldsymbol{\nu} \in \mathbb{N}$. Distribution $\mathrm{P}\{S_{\boldsymbol{\nu}} > s\}$ gets smaller as the mass t of $\boldsymbol{\nu}$ is shifted on ν_i with larger i. Thus, we have the following partial ordering between the possible request service time distributions: $S_{\boldsymbol{\nu}} > S_{\boldsymbol{\nu}'}$, if there exists a j such that $\nu_i \leq \nu'_i$ for all $i \geq j$. This ordering implies that $S_{\boldsymbol{\nu}}$ is the *fastest* and the *slowest* when $\boldsymbol{\nu}$ is equal to $(0, \ldots, 0, t)$ and $(t, 0, \ldots, 0)$ respectively. We refer to $S_{(0,\ldots,0,t)}$ as S_{fastest}, and $S_{(t,0,\ldots,0)}$ as S_{slowest}. Their distributions are

$$\begin{aligned} \mathrm{P}\{S_{\text{fastest}} > s\} &= e^{-\mu(t+1)s}, \\ \mathrm{P}\{S_{\text{slowest}} > s\} &= e^{-\mu s}(1 - (1 - e^{-\mu s})^r)^t. \end{aligned} \tag{7}$$

B. Fork-Join System in the Split-Merge Model (FJ-SM)

Recall that in the FJ-SM system, requests wait in a centralized FCFS queue and are admitted to service one at a time. FJ-SM therefore does not introduce any dependence between the service time distributions of different requests. In fact, it is straightforward to see that the FJ-SM system implements an M/G/1 queue with an arrival rate of λ and the service time distribution $T_{\text{FJ-}(r,t)}$ given in Theorem 1.

In FJ-SM, all servers are blocked until the request at the HoL finishes service, whereas in FJ-GA or FJ-FA, each server can independently start serving from their queues. FJ-SM therefore performs slower than both FJ-GA and FJ-FA. This serves us to find an upper bound on the download time in FJ-GA and FJ-FA as stated below. It should be noted that the Split-Merge model is also used in [12] for deriving upper bounds on the response time of (n, k)-FJ queue.

Lemma 5: The download time in FJ-SM is stochastically dominant over that in FJ-FA and FJ-GA, i.e., $P\{T_{\text{FJ-GA}} > t\} \leq P\{T_{\text{FJ-SM}} > t\}$ *and* $P\{T_{\text{FJ-FA}} > t\} \leq P\{T_{\text{FJ-SM}} > t\}$*.*

Fast-Split-Merge Model: Observe that the request service time distribution in FJ-SM, given in Theorem 1, is the same as $\mathrm{P}\{S_{\text{slowest}} > s\}$ in (7). Recall that S_{slowest} denotes the slowest possible request service time distribution in the FJ-FA system. In other words, FJ-SM is a modification of FJ-FA such that it forces every request to have the slowest possible service time.

We next consider the *Fast-Split-Merge model*, denoted as FJ-FSM, which operates at the other extreme of FJ-SM and serves every request with the fastest possible service time S_{fastest} in (7). Using Observation 1, it is straightforward to show that the FJ-FSM model implements an M/M/1 queue with an arrival rate of λ and service time distribution $\mathrm{Exp}((t+1)\mu)$. Just as we used FJ-SM to find an upper bound on $T_{\text{FJ-FA}}$ (Lemma 5), we next use FJ-FSM to find a lower bound.

Lemma 6: $P\{T_{\text{FJ-FA}} > x\} \geq \exp\left(-\left((t+1)\mu - \lambda\right) x\right)$ *for* $\lambda < (t+1)\mu$*.*

Proof: FJ-FA implements a FCFS queue (Observation 1), and each request is served with one of the $\binom{t+r-1}{r-1}$ distributions given in Lemma 4. In FJ-FSM, all requests are served with the fastest possible service time. Response time of FJ-FSM therefore serves as a lower bound for $T_{\text{FJ-FA}}$. As the FJ-FSM model implements an M/M/1 queue with arrival rate λ and service time distribution $\mathrm{Exp}((t+1)\mu)$, we get the result. ■

The immediate corollary of Lemma 6 gives a lower bound on the average download time in FJ-FA as $\mathbb{E}[T_{\text{FJ-FA}}] \geq 1/\left((t+1)\mu - \lambda\right)$ for $\lambda < (t+1)\mu$.

VI. Proofs of Theorems 2 and 3

Proof of Theorem 2 Lower bounds (a) *and* (b)*:* Consider an enlarged system that consists of k copies, system-i for $i = 1, \ldots, k$, of the original FJ system. Let us forward the requests for object i only to system-i. Then, system-i is equivalent to a FJ-FA operating under the arrival rate of $p_i\lambda$. Each request will experience a smaller response time in this enlarged system than in the original. Hence the lower bound (a). The Fast-Split-Merge model gives a lower bound on the response time of system-i (Lemma 6), which together with (a) gives (b). **Upper bounds** (c) **and** (d) follow from Lemma 5. ■

Proof of Theorem 3: Lower bound (a) given on $T_{\text{FJ-GA}}$ in (3) is a mixture distribution with components distributed as $\mathrm{Exp}((t+1)\mu - p_i\lambda)$ for $i = 1, \ldots, k$. Expected value of this mixture distribution is the lower bound for $\mathbb{E}[T_{\text{FJ-GA}}]$. By (d) given in (4), average response time $\mathbb{E}[T_{\text{FJ-SM}}]$ in FJ-SM system yields an upper bound on $\mathbb{E}[T_{\text{FJ-GA}}]$. FJ-SM implements an M/G/1 queue with the service time distribution S_{slowest} (Section V-B). Then the Pollaczek-Khinichin formula [33] gives

$$\mathbb{E}[T_{\text{FJ-SM}}] = \mathbb{E}[S_{\text{slowest}}] + \frac{\lambda \mathbb{E}\left[S^2_{\text{slowest}}\right]}{2\Big(1 - \lambda \mathbb{E}[S_{\text{slowest}}]\Big)}. \tag{8}$$

$\mathbb{E}[S_{\text{slowest}}]$ is given in Theorem 1. We find $\mathbb{E}\left[S^2_{\text{slowest}}\right]$ as

$$\begin{aligned} \mathbb{E}\left[S^2_{\text{slowest}}\right] &\overset{(i)}{=} \int_0^\infty 2s\, \mathrm{P}\{S_{\text{slowest}} > s\}\, \mathrm{d}s \\ &\overset{(ii)}{=} \int_0^\infty 2s \exp(-\mu s) \left(1 - (1 - \exp(-\mu s))^r\right)^t \mathrm{d}s \\ &\overset{(iii)}{=} \sum_{j=0}^{t} \binom{t}{j} (-1)^j \sum_{l=0}^{rj} (-1)^l \binom{rj}{l} \frac{2}{\mu^2 (l+1)^2}, \end{aligned} \tag{9}$$

where (i) holds because $\mathbb{E}\left[X^2\right] = \int_0^\infty 2s\, \mathrm{P}\{X > s\}\, \mathrm{d}s$ for a non-negative r.v. X, (ii) comes from (7), and (iii) follows from the binomial expansion of $\left(1 - (1 - \exp(-\mu s))^r\right)^t$ and interchanging the order of integration and summation. Finally, substituting in (8) the first and second moments of S_{slowest} gives us the upper bound in (5). ■

Proof of Lemma 3: Consider a sample arrival sequence L of requests in the FJ-GA system. Each request-i denotes a pair (t_i, o_i) where t_i is the arrival time and o_i is the requested object. Let ℓ be the index of the first request in L that is asking for an object different from the first requested object o_1. Let Γ be an operator such that $\Gamma(L)$ returns a copy of L with only one modification: the object being asked by request-ℓ is changed to the first requested object o_1. Note that Γ keeps the request arrival times t_i the same. Let $L^{(m)}$ be the sequence obtained by applying Γ on L repeatedly m times.

Let T_L be the download time under L and $T_{L^{(1)}}$ under $L^{(1)}$. System under $L^{(1)}$ behaves as FJ-FA for requests-($\leq \ell$), that is, requests-($\leq \ell$) depart the system in order. However, under L, request-ℓ might finish early at the leading servers. This implies that request-ℓ might depart before even reaching HoL. Conditioned on the event that request-ℓ does not have an early departure, it is not difficult to show that $T_{L^{(1)}}$ is stochastically the same as T_L when code locality is two. Further, conditioned on the event that that request-ℓ has an early departure, it is not difficult to show that $T_L \leq T_{L^{(1)}}$ stochastically when code locality is two. We skip showing these two observations here due to space constraints. This overall gives us $T_L \leq T_{L^{(1)}}$.

Using the same arguments given in the previous paragraph, we can show that $T_{L^{(i)}} \leq T_{L^{(i+1)}}$ for all i. Let $L^{(*)}$ denote a sequence after applying Γ on L sufficient number of times such that $L^{(*)}$ has all requests asking for the same object. Notice that $L^{(*)}$ is a sample arrival sequence for FJ-FA. The previous inequality we showed implies $T_L \leq T_{L^{(1)}} \leq T_{L^{(2)}} \leq \ldots \leq T_{L^{(*)}}$. Let us also define $\mathbb{L}$ (resp. $\mathbb{L}^{(*)}$) as the set of all request arrival sequences for FJ-GA (resp. FJ-FA). Every L in $\mathbb{L}$ will be transformed into an $L^{(*)}$ in $\mathbb{L}^{(*)}$ by repeated application of operator Γ. Thus, Γ is a surjective function from $\mathbb{L}$ to $\mathbb{L}^{(*)}$. Let us define the subset of all L in $\mathbb{L}$ that map to $L^{(*)}$ as $\Gamma^{-1}(L^{(*)})$. Then it is easy to see

$$\mathrm{P}\left\{L^{(*)}\right\} = \sum_{L \in \Gamma^{-1}(L^{(*)})} \mathrm{P}\left\{L\right\}. \tag{10}$$

where $\mathrm{P}\left\{L\right\}$ (resp. $\mathrm{P}\left\{L^{(*)}\right\}$) denotes the probability of sampling L (resp. $L^{(*)}$) from $\mathbb{L}$ (resp. $\mathbb{L}^{(*)}$).

The FJ-GA download time distribution can be written as

$$\begin{aligned}\mathrm{P}\left\{T_{\text{FJ-GA}} > t\right\} &\overset{(a)}{=} \sum_{L \in \mathbb{L}} \mathrm{P}\left\{L\right\} \mathrm{P}\left\{T_L > t\right\} \\ &\overset{(b)}{=} \sum_{L^{(*)} \in \mathbb{L}^{(*)}} \sum_{L \in \Gamma^{-1}(L^{(*)})} \mathrm{P}\left\{L\right\} \mathrm{P}\left\{T_L > t\right\} \\ &\overset{(c)}{\leq} \sum_{L^{(*)} \in \mathbb{L}^{(*)}} \sum_{L \in \Gamma^{-1}(L^{(*)})} \mathrm{P}\left\{L\right\} \mathrm{P}\left\{T_{L^{(*)}} > t\right\},\end{aligned} \tag{11}$$

where (a) follows from the law of total probability, (b) follows from the discussion previously given on $\Gamma^{-1}(L^{(*)})$, and (c) follows by using $T_L \leq T_{L^{(*)}}$ Then, we find

$$\begin{aligned}\mathrm{P}\left\{T_{\text{FJ-GA}} > t\right\} &\overset{(a)}{\leq} \sum_{L^{(*)} \in \mathbb{L}^{(*)}} \mathrm{P}\left\{T_{L^{(*)}} > t\right\} \sum_{L \in \Gamma^{-1}(L^{(*)})} \mathrm{P}\left\{L\right\} \\ &\overset{(b)}{=} \sum_{L^{(*)} \in \mathbb{L}^{(*)}} \mathrm{P}\left\{L^{(*)}\right\} \mathrm{P}\left\{T_{L^{(*)}} > t\right\},\end{aligned}$$

where (a) follows from (11) and (b) follows from (10). The proof concludes by observing that the right hand side $\sum_{L^{(*)} \in \mathbb{L}^{(*)}} \mathrm{P}\left\{L^{(*)}\right\} \mathrm{P}\left\{T_{L^{(*)}} > t\right\} = \mathrm{P}\left\{T_{\text{FJ-FA}} > t\right\}$. ∎

VII. M/G/1 Queue Approximation for FJ-FA

A. Background

While discussing the FJ-FA system, we re-defined the request service start times (see in Section V-A, Definition 1). That allowed us to dissect the system dynamics and lead us to make two important observations: i) requests depart the FJ-FA system in the order they arrive (Observation 1), and ii) there are $\binom{t+r-1}{r-1}$ possible request service time distributions, as given in Lemma 4. These two observations, together with a the one we make below, allow us to argue that the FJ-FA system can be approximated as an M/G/1 queue.

The service time distribution for a request is dictated by the system state at its service start time. Queue lengths carry memory between the service starts of the subsequent requests. For instance, as request-i starts service, if the difference between the queue lengths across all the servers is at least 2, then this difference will be at least 1 as request-$(i+1)$ starts service. Therefore in general, service times are not independent across the requests. However, very importantly, request service times are only *loosely coupled*. Once a request is finished, its outstanding redundant copies get removed from service. This helps the slow servers to catch up with the leading servers. It is "hard" for the leading servers to keep leading as they compete with every other server. Queues across all the servers are therefore expected to frequently level up. Every time the queues level up while they are non-empty corresponds to an epoch at which a request start service (i.e., moving to HoL) with type-0 service time distribution. Given a time epoch t at which the queues level up, requests that move to HoL before t or those that move after t have independent service time distributions. Therefore, queue levelling time epochs break the dependence between the service times. Given that such time epochs occur frequently, request service times constitute a series of independent small-size batches.

Observation 2: The FJ-FA system experiences frequent time epochs across which request service times are independent.

The observations we have made so far lead to *an approximate method* for analyzing the FJ-FA system. Requests depart the system in the order they arrive (Observation 1), hence the system as a whole acts as a FCFS queue. There are $\binom{t+r-1}{r-1}$ possible distributions for the request service times (Lemma 4). Although request service times are not independent, they are loosely coupled (Observation 2). Putting all these together, we propose the following approximation for the FJ-FA system.

Approximation 1: The FJ-FA system can be approximated as an M/G/1 queue, and it holds that

$$\mathbb{E}\left[T_{\textit{FJ-FA}}\right] \approx \mathbb{E}\left[T_{\textit{M/G/1}}\right] = \mathbb{E}\left[S\right] + \frac{\lambda \mathbb{E}\left[S^2\right]}{2(1 - \lambda \mathbb{E}\left[S\right])}. \tag{12}$$

Here, the moments of S are given as

$$\mathbb{E}\left[S\right] = \sum_{\boldsymbol{\nu}\in\mathbb{N}} P\left\{\text{Type-}\boldsymbol{\nu}\,\text{service}\right\}\mathbb{E}\left[S_{\boldsymbol{\nu}}\right],$$
$$\mathbb{E}\left[S^2\right] = \sum_{\boldsymbol{\nu}\in\mathbb{N}} P\left\{\text{Type-}\boldsymbol{\nu}\,\text{service}\right\}\mathbb{E}\left[S_{\boldsymbol{\nu}}^2\right], \tag{13}$$

where the set $\mathbb{N}$ is defined in (6)*, $P\left\{\text{Type-}\boldsymbol{\nu}\,\text{service}\right\}$ is the probability that the service time of an arbitrary request is sampled from $S_{\boldsymbol{\nu}}$, whose distribution is in Lemma 4.*

Note that (12) follows from the Pollaczek-Khinichin formula [33], and (13) follows from Lemma 4. Moreover, the approximation becomes exact for the system with locality one, that is, when the Fork-Join content access model is implemented on the system with replicated storage [12].

In the M/G/1 approximation given above, the only unknown quantity is $\mathrm{P}\left\{\text{Type-}\boldsymbol{\nu}\,\text{service}\right\}$. Recall that we study the system dynamics in steady state. Given that, $\mathrm{P}\left\{\text{Type-}\boldsymbol{\nu}\,\text{service}\right\}$ denotes the probability of serving request i with type-$\boldsymbol{\nu}$ distribution in the limit $i \to \infty$. By ergodicity the limiting value of this probability is equal to the limiting fraction $f_{\boldsymbol{\nu}}$ of the requests served with type-$\boldsymbol{\nu}$ distribution [34]. We rely on $f_{\boldsymbol{\nu}}$ in the derivations presented in the sequel.

B. FJ-FA With Locality Two

We next focus on the FJ-FA model for availability codes with locality two. This class of codes is of interest because they are minimally different from replication with locality one.

Given that recovery groups consist of two servers, there can be at most one leading server in each. Hence a request can have at most one early departure at each of the recovery groups. (See Section V-A for the definition of early departing request copies.) Request service type vector $\boldsymbol{\nu}$ in this case is given by (ν_0, ν_1) where ν_0 (resp. ν_1) denotes the number of recovery groups at which the request has no (resp. one) early departing copy. Given that $\nu_0+\nu_1$ is fixed and equal to t, it is sufficient to only keep track of ν_1 to determine the service time distribution of a request. In other words, service time distribution for a request is defined by its number of early departing copies, *i.e.*, number of recovery groups at which the request had one early departure. Given that ν_1 lies in $[0, t]$, the set $\mathbb{S}$ of all possible service time distributions is of size $t+1$. If a request has an early departing copy at i recovery groups before it moves to HoL, i.e., $\nu_1 = i$, then the request will be served with type-i service time distribution. (Here, we follow the convention introduced on service types in Section V-A.) Type-i service time distribution is given by

$$\mathrm{P}\left\{S_i > s\right\} = \exp(-\mu s)^{i+1}\left(1-(1-\exp(-\mu s))^2\right)^{t-i} \tag{14}$$

for $i = 0, 1, \ldots, t$. We have $\mathrm{P}\left\{S_{i+1} > s\right\}/\mathrm{P}\left\{S_i > s\right\} < 1$ for $s > 0$. This implies that S_i are stochastically ordered as $S_0 > S_1 > \cdots > S_t$. Recall from Section V-A that for the code with general locality, we have a partial ordering between the service time distributions. This partial ordering turns into a complete ordering in this case with locality of two.

Deriving the request service time probabilities f_i requires an exact analysis of the system, which is intractable due to the state explosion problem. We next conjecture a relation between f_i for the system with locality two. This relation will be used later to derive several estimates for f_i.

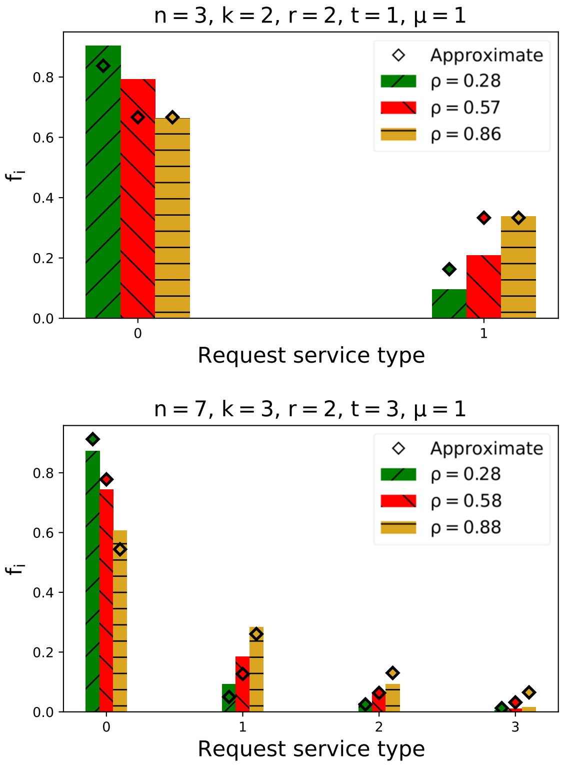

Fig. 4. Simulated request service time probabilities f_i for systems with (3, 2, 2, 1)- and (7, 3, 2, 3)-availability codes. Here ρ is the average system load.

Conjecture 1: In the FJ-FA system with locality two, $f_{i-1} > f_i$ for $i = 1, \ldots, t$.

We next briefly discuss the reasoning behind the conjecture. Observation 2 states that the queues at the servers frequently level up. This is because the leading servers in the recovery groups compete with every other server in the system to keep leading. For a request to have type-i distribution for service, it needs to have one early departure at exactly i recovery groups. This requires one server in each of the i recovery groups to be leading, which gets less likely for larger i. We have validated the conjecture with extensive simulations. Fig. 4 shows the simulated values for f_i in a system that employs the binary Simplex code with availability one or three, and locality two. Furthermore, we found a strong pointer for the conjecture (can be found in [25]). This pointer says that given a request is served with type-i distribution, the subsequent request is more likely to be served with type-j distribution for $j < i$. We prove the conjecture for the system with availability one and locality two. This is implied by the bounds given on f_i in Theorem 6.

Fig. 4 shows that, as the offered load increases, the frequency of type-i service time increases for larger i, which agrees with the discussion above. To have type-i distribution, a request needs to have one early departure at i recovery groups. This is possible only if there are i leading servers. Recovery servers can go ahead of their peers only if their queues are non-empty. The queues are more likely to build up

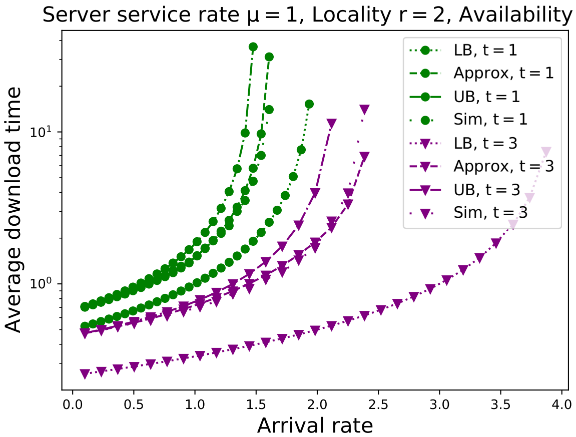

Fig. 5. Average download time versus λ for FJ-FA with locality two. *Sim* refers to the simulated values. *UB* (resp. *LB*) refers to the upper (resp. lower) bounds in Lemma 5 (resp. 6). *Approx* refers to Approximation 2.

under greater offered load. This allows the leading servers to progress even further than they could under smaller offered load. Using Conjecture 1, we find estimates for f_i. (see [25] for details). Then substituting these estimates in the M/G/1 approximation (cf. (12)), we derive the following approximation.

Approximation 2: The FJ-FA system with locality two is approximately an M/G/1 queue with service time distribution

$$P\{S>s\} = \Big(1+\sum_{i=1}^{t}\prod_{j=0}^{i-1}\rho_j\Big)^{-1}\sum_{i=0}^{t}P\{S_i>s\}\prod_{j=0}^{i-1}\rho_j,$$

where ρ_j are recursively computed as ($i = 1, \ldots, t$)

$$\rho_0 = \frac{\lambda\mathbb{E}[V]}{t\big(1-\lambda\mathbb{E}[V]\big)},$$
$$\rho_i = \frac{1-\Big(1-\lambda\mathbb{E}[V]\Big)\Big(1+\sum_{k=0}^{i-1}\prod_{l=0}^{k}\rho_l\Big)}{\big(1-\lambda\mathbb{E}[V]\big)(t-i)\prod_{k=0}^{i-1}\rho_k},$$

where $\mathbb{E}[V] = \frac{1}{t+1}\sum_{i=0}^{t}\mathbb{E}[S_i]$ and S_i are defined in (14).

Comparison of the Approximations and Bounds: Fig. 5 gives a comparison between the M/G/1 approximation (Approximation 2) and the bounds presented in Section V-A.

C. FJ-FA With Locality Two and Availability One

We here focus on availability codes with locality two and availability one. This represents the simplest possible availability code, i.e., objects a, b are stored together with $a+b$ over three servers. The corresponding FJ system is thus the simplest of all FJ systems with availability codes. Requests in this case are served with either type-0 or type-1 service time distribution. Although the exact analysis is still formidable, the system state is simple enough to find tighter bounds on the request service time probabilities f_0 and f_1. Substituting these into Approximation 2 leads to a tighter M/G/1 approximation.

The simplicity of the setting here allows us to address a more general service time model than identical Exp(μ) r.v.'s. We continue assuming the service times to be independent across different servers and request copies. However this time,

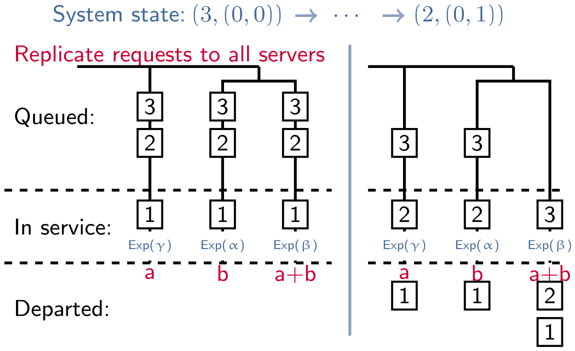
System state: $(3,(0,0)) \rightarrow \cdots \rightarrow (2,(0,1))$
Replicate requests to all servers
Queued:
In service:
Departed:
3
2
1
Exp(γ)
Exp(α)
Exp(β)
a
b
a+b

Fig. 6. FJ-FA with availability one and locality two. Snapshot on the left (resp. right) illustrates the time epoch at which request 1 (resp. 2) starts service and its service time distribution S_0 (resp. S_1).

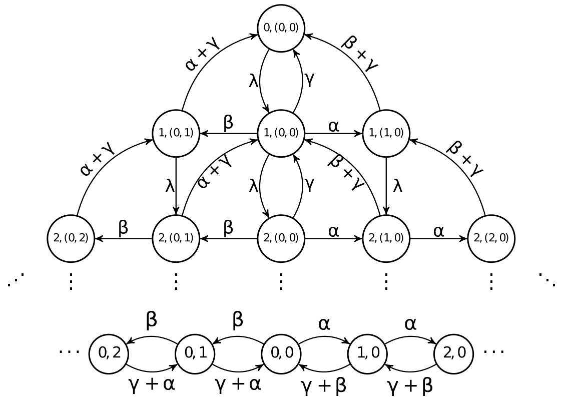

Fig. 7. Markov state process for the FJ-FA system with availability one and locality two (top), and its high traffic approximation (bottom).

we let the service time at the systematic server to be distributed as Exp(γ), and the service times at the recovery servers to be distributed as Exp(α) and Exp(β). In the following we refer to server with service rate x as server-x.

The system state at time τ can be described with a triple $\boldsymbol{s}(\tau) = (N(\tau),(n_\alpha(\tau),n_\beta(\tau)))$. $N(\tau)$ denotes the total number of requests in the system at time τ. This is given by the number of request copies present in the systematic server. $n_\alpha(\tau)$ (resp. $n_\beta(\tau)$) denotes by how many request copies server-α (resp. server-β) is leading the other recovery server. In other words, denoting the queue length at server-x at time τ with len-$x(\tau)$, we can express $n_\alpha(\tau)$ and $n_\beta(\tau)$ as

$$n_\alpha(\tau) = \max\{\text{len-}\alpha(\tau) - \text{len-}\beta(\tau), 0\},$$
$$n_\beta(\tau) = \max\{\text{len-}\beta(\tau) - \text{len-}\alpha(\tau), 0\}. \quad (15)$$

It follows that $n_\alpha(\tau)n_\beta(\tau) = 0$ for all τ. This is because there can be only one leading recovery server at any time since the system has one recovery group of two servers (availability one, locality two). (Recall the definition of leading server from the second paragraph of Section V-A.) The system state is illustrated in Fig. 6 with two different snapshots of the system.

The system state $\boldsymbol{s}(\tau)$ at time τ is a Markov process as illustrated in Fig. 7 (top). Let us refer to $\mathrm{P}\{\boldsymbol{s}(\tau) = (k,i,j)\}$ by $p_{k,i,j}(\tau)$. Suppose that the system stability is imposed and $\lim_{\tau\to\infty} p_{k,i,j}(\tau) = p_{k,i,j}$. Balance equations for the system are summarized for $k, i, j \geq 0$ as

$$\begin{aligned}&\big(\gamma \mathbb{1}(k \geq 1) + \alpha \mathbb{1}(i \geq 1) + \beta \mathbb{1}(j \geq 1)\big) p_{k,i,j} \\ &\quad = \lambda \mathbb{1}(k \geq 1, i \ \geq 1, j \geq 1)\ p_{k-1,i-1,j-1} + \gamma\ p_{k+1,i+1,j+1} \\ &\qquad + (\gamma+\alpha) p_{k+1,i+1,j} + (\gamma+\beta) p_{k+1,i,j+1}.\end{aligned}$$

These balance equations do not permit exact analysis of the system's steady state behavior, because the state space is infinite in two dimensions. We use two methods to analyse the process approximately: 1) the *local balance method* with a guess-based analysis [35, Chapter 17], and 2) the *matrix analytic method* [35, Chapter 21], which involves truncating the process and numerically finding $p_{k,i,j}$ with an iterative procedure. The analysis is given in [25]. The matrix analytic method gives the following upper bound on the average download time. This bound is provably tighter than the Split-Merge upper bound previously given in Theorem 3. This is because the Split-Merge model truncates the pyramid process and keeps only the central column, while in our application of the matrix analytic method, we keep the five central columns.

Theorem 4: In the FJ-FA system with availability one and locality two, the average data download time is bounded as

$$\begin{aligned}\mathbb{E}\left[T_{\text{FJ-FA}}\right] < \frac{1}{\lambda}\Big(&\boldsymbol{\pi}_0 \mathbf{1}_0^T - \pi_{0,(0,0)} \\ &+ \boldsymbol{\pi}_1 \left((\boldsymbol{I}-\boldsymbol{R})^{-2} + (\boldsymbol{I}-\boldsymbol{R})^{-1}\right) \mathbf{1}_1^T\Big).\end{aligned}$$

Here $\mathbf{1}_0 = [1,1,1,1]$, $\mathbf{1}_1 = [1,1,1,1,1]$. *and* $\mathbf{1}_0^T$ *and* $\mathbf{1}_1^T$ *refer to their transpose. Vectors* $\boldsymbol{\pi}_0$ *and* $\boldsymbol{\pi}_i$ *are given as*

$$\begin{aligned}\boldsymbol{\pi}_0 &= \left[\pi_{0,(0,0)},\ \pi_{1,(0,1)},\ \pi_{1,(0,0)},\ \pi_{1,(1,0)}\right], \\ \boldsymbol{\pi}_1 &= \left[\pi_{2,(0,2)},\ \pi_{2,(0,1)},\ \pi_{2,(0,0)},\ \pi_{2,(1,0)},\ \pi_{2,(2,0)}\right],\end{aligned}$$

where $\pi_{n_\gamma,(n_\alpha,n_\beta)}$ *is the steady-state probability that queue lengths at the servers are len-*$\gamma = n_\gamma$, *len-*$\alpha = n_\alpha$ *and len-*$\beta = n_\beta$. *Matrices* $\boldsymbol{I}$ *and* $\boldsymbol{R}$ *are* 5×5. $\boldsymbol{I}$ *is the identity matrix.* $\boldsymbol{R}$ *is numerically computed in terms of the arrival rate* λ *and service rates* γ, α *and* β. *The algorithm is described in [25].*

1) High-Traffic Approximation: We next present a method to approximately analyze the system. As in Section V-A, we first find estimates for the request service time probabilities f_0 and f_1, and then substitute these in the M/G/1 approximation (cf. (12)). Here we are able to find tighter estimates for f_0 and f_1 by exploiting the simplicity of the system state.

Suppose that the system operates close to its stability limit, such that the servers are always busy serving a request copy. This reduces the system complexity, as we can now describe the system state keeping track of $n(\tau) = (n_\alpha(\tau), n_\beta(\tau))$ defined in (15). System state in this case implements a birth-death Markov process as shown in Fig. 7 (bottom). Referring to $\lim_{\tau\to\infty} \mathrm{P}\{n(\tau) = (i,j)\}$ as $p_{i,j}$, and using the balance equations $\alpha\ p_{i,0} = (\gamma+\beta) p_{i+1,0}$ and $\beta\ p_{0,i} = (\gamma+\alpha) p_{0,i+1}$ for $i \geq 0$, we find the limiting state probabilities as

$$\begin{aligned}&p_{0,0} = \frac{\gamma^2 - (\alpha-\beta)^2}{\gamma(\alpha+\beta+\gamma)}, \qquad p_{i,0} = \left(\frac{\alpha}{\beta+\gamma}\right)^i p_{0,0}, \\ &p_{0,i} = \left(\frac{\beta}{\alpha+\gamma}\right)^i p_{0,0} \quad \text{for } i \geq 1.\end{aligned}$$

We next use these expressions to find bounds on the fraction of the requests completed by the systematic server, w_s, or by the recovery group, w_r. Recall that a request completes as soon as either its copy at the systematic server or both copies at the recovery group finish service. For the sake of simplicity, let the service rates α and β at the recovery servers be μ. We keep the service rate at the systematic server fixed at γ.

Theorem 5: In FJ-FA system with availability one and locality two, $w_s \geq \frac{\gamma\nu}{\gamma\nu+2\mu^2}$ *and* $w_r \leq \frac{\mu^2}{\gamma\nu+2\mu^2}$ *for* $\nu = \gamma + 2\mu$.

Proof: Suppose the system operates close to its stability limit. Under this high-traffic assumption, let us refer to the values of w_s and w_r as $\hat{w_s}$ and $\hat{w_r}$. The recovery servers regularly go idle under stability. Therefore, fraction of the request completions at the recovery group is smaller under high-traffic approximation than it is under stability. Hence $w_r \leq \hat{w_r}$, and $w_s = 1 - w_r$ implies $w_s \geq \hat{w_s}$.

Next, we derive $\hat{w_s}$ and $\hat{w_r}$ from the steady state probabilities of the Markov chain embedded in the state process $n(\tau)$ (Fig. 7, bottom). System stays at each state for an exponential duration of rate $\nu = \gamma + 2\mu$. Therefore, steady state probabilities p_i of $n(\tau)$ (i.e., the limiting fraction of the time spent in state i) and the steady state probabilities π_i of the embedded Markov chain (i.e., the limiting fraction of the state transitions into state i) are equal. This is easily seen by the equality $\pi_i = p_i\nu \Big/ \sum_{i\geq 0} p_i\nu = p_i$.

Let f_s be the limiting fraction of the state transitions that represent request completions by the systematic server. Let f_r denote the same quantity for the recovery group. We have

$$\begin{aligned}f_s &= \pi_{0,0}\gamma/\nu + \sum_{i=1}^{\infty} \pi_{i,0}\ \gamma/\nu + \sum_{i=1}^{\infty} \pi_{0,i}\ \gamma/\nu = \gamma/\nu, \\ f_r &= \sum_{i=1}^{\infty} (\pi_{0,i} + \pi_{i,0})\mu/\nu = 2\left(\mu/\nu\right)^2.\end{aligned}$$

The limiting fraction of the state transitions that correspond to request departures is $f_d = f_s + f_r = (\gamma\nu + 2\mu^2)/\nu^2$. Thus the fraction of requests completed by the systematic server and the recovery group are

$$\hat{w_s} = \frac{f_s}{f_d} = \frac{\gamma\nu}{\gamma\nu + 2\mu^2}, \quad \hat{w_r} = \frac{f_r}{f_d} = \frac{2\mu^2}{\gamma\nu + 2\mu^2}.$$

■

Fig. 8 shows the general tightness of the high traffic bounds given in Theorem 5, which increases with the offered load as expected. We next find bounds on the request service time probabilities f_0 and f_1 using the high-traffic approximation. Note that bounds given below prove Conjecture 1 for the system with locality two and availability one.

Theorem 6: In the FJ-FA system with availability one and locality two, $f_0 \geq \frac{\gamma\nu}{\gamma\nu+2\mu^2}$ *and* $f_1 \leq \frac{2\mu^2}{\gamma\nu+2\mu^2}$ *for* $\nu = \gamma + 2\mu$.

Proof: Suppose that the system operates close to its stability limit. Under this high-traffic assumption, let us refer to the values of f_0 and f_1 as $\hat{f}_0$ and $\hat{f}_1$. Under stability, system has to empty out regularly. Every request that finds the system empty, makes type-0 service start. However, such idle periods never happen under high-traffic approximation. That is why f_0 under the high traffic approximation is smaller than its value under stability. Thus we conclude $f_0 \geq \hat{f}_0$ and $f_1 \leq \hat{f}_1$.

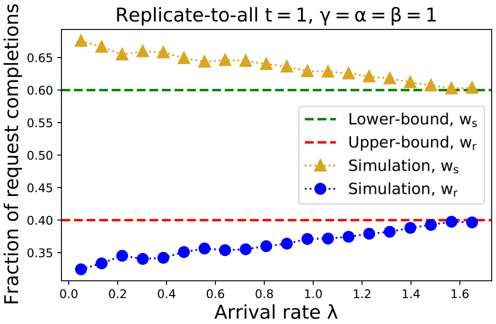

Fig. 8. Simulated fraction of the request completions by the systematic server, w_s, and by the recovery group, w_r. The lower and the upper bound are computed using the expressions given in Theorem 5.

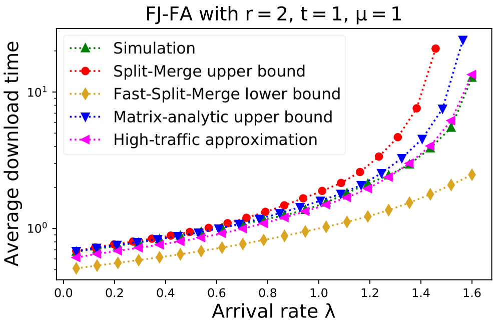

Fig. 9. Comparison of the bounds on average download time versus λ for FJ-FA with availability one and locality two. Split-Merge and Fast-Split-Merge bounds are given in lemmas 5 and 6. Matrix analytic bound is given in Theorem 4. High-traffic approximation comes from substituting in (12) the bounds given in Theorem 6.

Next, we derive $\hat{f}_0$ and $\hat{f}_1$ using the steady state probabilities of the Markov chain embedded in $n(\tau)$ (Fig. 7, bottom). State transitions towards state $(0,0)$ represent request completions. Let f_d be the fraction of such state transitions. Under high-traffic, every time the request at HoL departs there is a subsequent request that starts service. Thus, every time system transitions into $(0,0)$ (or any other transition towards it), a new request makes a type-0 (resp. type-1) service start. Let $f_{\to 0}$ and $f_{\to 1}$ be the fraction of state transitions that represent type-0 and type-1 service starts. Then

$$\begin{aligned} f_d &= \pi_{0,0}\gamma/\nu + \sum_{i=1}^{\infty}(\pi_{0,i} + \pi_{i,0})(\mu+\gamma)/\nu \\ &= (2\mu^2 + 2\mu\gamma + \gamma^2)/\nu^2 \\ f_{\to 0} &= \pi_{0,0}\gamma/\nu + \pi_{1,0}(\mu+\gamma)/\nu + \pi_{0,1}(\mu+\gamma)/\nu \\ &= \pi_{0,0}\Big(\frac{\gamma}{\nu} + \frac{2\mu}{\mu+\gamma}\frac{\mu+\gamma}{\nu}\Big) = \pi_{0,0} = \frac{\gamma}{\gamma+2\mu}. \end{aligned}$$

Thus, the limiting fraction $\hat{f}_0$ (resp. $\hat{f}_1$) of the requests that make type-0 (type-1) service start are, for $\nu = \gamma + 2\mu$,

$$\hat{f}_0 = \frac{f_{\to 0}}{f_d} = \frac{\gamma\,\nu}{\gamma\,\nu + 2\mu^2}, \quad \hat{f}_1 = 1 - \hat{f}_0 = \frac{2\mu^2}{\gamma\,\nu + 2\mu^2}$$

■

2) Comparison of Approximations: We approximate the FJ-FA system as an M/G/1 queue (Approximation 1), which together with the PK formula gives us an approximate expression for the average download time (12). This approximation requires the first and second moments of the service time distribution (13). Substituting the bounds in Theorem 6 in place of the actual probabilities f_0 and f_1 yields the following lower bounds on the service time moments:

$$\begin{aligned} \mathbb{E}\,[S] &\geq \frac{1}{3}\Big(\frac{2}{\gamma+\mu} - \frac{1}{\gamma+2\mu}\Big) + \frac{2}{3}\frac{1}{\gamma+\mu}, \\ \mathbb{E}\,[S^2] &\geq \frac{1}{3}\Big(\frac{4}{(\gamma+\mu)^2} - \frac{2}{(\gamma+2\mu)^2}\Big) + \frac{2}{3}\frac{2}{(\gamma+\mu)^2}. \end{aligned}$$

Substituting these bounds in (12) gives a lower bound on the download time. However, this lower bound can only be treated as an approximation since (12) is not exact but an approximation. As shown in Fig. 9, this approximation performs very well in predicting the actual average download time, especially compared to the Split-Merge and Fast-Split-Merge bounds given in lemmas 5 and 6.

VIII. Conclusion and Future Directions

Storage systems with availability codes allow simultaneous downloads of each data object from multiple recovery groups. However, downloading an object from a recovery group requires fetching the coded objects from all the servers in the group. Such downloads are slower than those from a single server because of straggling servers. In this paper, we asked if availability codes improve hot data download performance. We adopted the Fork-Join (FJ) access scheme that employs redundant requests to mitigate stragglers' impact.

We found that availability codes lie between replication and MDS codes in the storage overhead vs. download latency tradeoff. Also, we found that availability codes achieve a favorable tradeoff between the storage overhead and download latency compared with the state-of-the-art erasure codes. We made these observations by deriving expressions for the download latency. An exact analysis was possible under the low traffic regime. Otherwise, availability codes give rise to multi-layer inter-dependent FJ queues. The state space explosion makes the exact analysis intractable in this case. Here, we derived bounds and approximations on the download time by devising systems that are tractable variants of the FJ system. For the case of locality two, which is minimally different from replication (i.e., locality one), we conjectured an order relationship between the service time probabilities, and derived approximations on the download time based on this conjecture. We demonstrated, via simulations, that these approximations are close to the download time.

We considered the FJ access model that replicates every request to all the servers upon arrival. This redundancy strategy treats all the requests uniformly, resulting in fairness. However, it aggressively adds redundant load on the system, which might be suboptimal. An interesting future direction is to investigate strategies that employs redundancy selectively based on the system state such as the current load offered on the system. Another future direction is to devise redundancy policies that are provably optimal for systems employing a given availability code. Overall, the ultimate goal is to design, for a given set of parameters, coding schemes and accompanying redundancy strategies that are optimal in terms of the download delay.

Acknowledgment

Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

References

[1] S. Kadhe, E. Soljanin, and A. Sprintson, "Analyzing the download time of availability codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2015, pp. 1467–1471.

[2] M. F. Aktas, E. Najm, and E. Soljanin, "Simplex queues for hot-data download," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 45, no. 1, pp. 35–36, Sep. 2017.

[3] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, "A survey on network codes for distributed storage," *Proc. IEEE*, vol. 99, no. 3, pp. 476–489, Mar. 2011.

[4] F. Oggier and A. Datta, "Self-repairing homomorphic codes for distributed storage systems," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 1215–1223.

[5] P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin, "On the locality of codeword symbols," *IEEE Trans. Inf. Theory*, vol. 58, no. 11, pp. 6925–6934, Nov. 2012.

[6] I. Tamo and A. Barg, "Bounds on locally recoverable codes with multiple recovering sets," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2014, pp. 691–695.

[7] A. S. Rawat, D. S. Papailiopoulos, A. G. Dimakis, and S. Vishwanath, "Locality and availability in distributed storage," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2014, pp. 681–685.

[8] L. Pamies-Juarez, H. D. L. Hollmann, and F. Oggier, "Locally repairable codes with multiple repair alternatives," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2013, pp. 892–896.

[9] A. Wang and Z. Zhang, "Repair locality with multiple erasure tolerance," *IEEE Trans. Inf. Theory*, vol. 60, no. 11, pp. 6979–6987, Nov. 2014.

[10] S. Melnik *et al.*, "Dremel: Interactive analysis of Web-scale datasets," *Proc. VLDB Endowment*, vol. 3, nos. 1–2, pp. 330–339, 2010.

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

[12] G. Joshi, Y. Liu, and E. Soljanin, "Coding for fast content download," in *Proc. 50th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Oct. 2012, pp. 326–333.

[13] L. Huang, S. Pawar, H. Zhang, and K. Ramchandran, "Codes can reduce queueing delay in data centers," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2012, pp. 2766–2770.

[14] G. Joshi, Y. Liu, and E. Soljanin, "On the delay-storage trade-off in content download from coded distributed storage systems," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 989–997, May 2014.

[15] G. Joshi, E. Soljanin, and G. Wornell, "Queues with redundancy: Latency-cost analysis," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 43, no. 2, pp. 54–56, Sep. 2015.

[16] M. F. Aktas and E. Soljanin, "Heuristics for analyzing download time in MDS coded storage systems," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 1929–1933.

[17] B. Li, A. Ramamoorthy, and R. Srikant, "Mean-field-analysis of coding versus replication in cloud storage systems," in *Proc. IEEE 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2016, pp. 1–9.

[18] P. Parag, A. Bura, and J.-F. Chamberland, "Latency analysis for distributed storage," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2017, pp. 1–9.

[19] G. Joshi, E. Soljanin, and G. Wornell, "Efficient redundancy techniques for latency reduction in cloud systems," *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 2, no. 2, pp. 1–30, May 2017.

[20] G. Liang and U. C. Kozat, "FAST CLOUD: Pushing the envelope on delay performance of cloud storage with coding," *IEEE/ACM Trans. Netw.*, vol. 22, no. 6, pp. 2012–2025, Dec. 2014.

[21] G. Joshi, E. Soljanin, and G. Wornell, "Efficient replication of queued tasks for latency reduction in cloud systems," in *Proc. 53rd Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2015, pp. 107–114.

[22] G. Ananthanarayanan *et al.*, "Scarlett: Coping with skewed content popularity in mapreduce clusters," in *Proc. 6th Conf. Comput. Syst. (EuroSys)*, 2011, pp. 287–300.

[23] C. Huang *et al.*, "Erasure coding in windows azure storage," in *Proc. USENIX Annu. Tech. Conf.*, 2012, pp. 15–26.

[24] M. Sathiamoorthy *et al.*, "XORing elephants: Novel erasure codes for big data," *Proc. VLDB Endowment*, vol. 6, no. 5, pp. 325–336, Mar. 2013.

[25] M. F. Aktas, "Performance evaluation of redundancy techniques for distributed storage and computing systems," Ph.D. dissertation, Dept. ECE, Rutgers Univ.-Graduate School-New Brunswick, New Brunswick, NJ, USA, 2020.

[26] N. B. Shah, K. Lee, and K. Ramchandran, "When do redundant requests reduce latency?" *IEEE Trans. Commun.*, vol. 64, no. 2, pp. 715–722, Feb. 2016.

[27] G. DeCandia *et al.*, "Dynamo: Amazon's highly available key-value store," *ACM SIGOPS Operating Syst. Rev.*, vol. 41, no. 6, pp. 205–220, Oct. 2007.

[28] D. Ford *et al.*, "Availability in globally distributed storage systems," in *Proc. 9th USENIX Symp. Operating Syst. Design Implement.*, Vancouver, BC, Canada, 2010, pp. 61–74.

[29] V. R. Cadambe and A. Mazumdar, "Bounds on the size of locally recoverable codes," *IEEE Trans. Inf. Theory*, vol. 61, no. 11, pp. 5787–5794, Nov. 2015.

[30] S. Kadhe and R. Calderbank, "Rate optimal binary linear locally repairable codes with small availability," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 166–170.

[31] L. Flatto and S. Hahn, "Two parallel queues created by arrivals with two demands I," *SIAM J. Appl. Math.*, vol. 44, no. 5, pp. 1041–1053, Oct. 1984.

[32] R. Nelson and A. N. Tantawi, "Approximate analysis of fork/join synchronization in parallel queues," *IEEE Trans. Comput.*, vol. C-37, no. 6, pp. 739–743, Jun. 1988.

[33] H. Tijms, *A First Course in Stochastic Models*. Hoboken, NJ, USA: Wiley, 2003.

[34] R. W. Wolff, "Poisson arrivals see time averages," *Oper. Res.*, vol. 30, no. 2, pp. 223–231, Apr. 1982.

[35] M. Harchol-Balter, *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*, 1st ed. New York, NY, USA: Cambridge Univ. Press, 2013.

Mehmet Fatih Aktaş received the B.S. degree in electrical and electronics engineering from Bilkent University, and the M.S. and Ph.D. degrees in electrical and computer engineering from Rutgers University. He is currently a Post-Doctoral Researcher with Rutgers University. His main research interests include to make distributed computer systems faster and more robust to runtime changes. He has been doing research in both systems development and theoretical analysis. In his research, he mostly relies on probabilistic modeling and on tools from applied probability, such as queueing theory, order statistics, and reinforcement learning.

Swanand Kadhe (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from Texas A&M University, in 2017. He is currently a Post-Doctoral Researcher with the University of California, Berkeley. His research interests include broadly in Information and coding theory, distributed machine learning, privacy and security, and blockchains. He was a recipient of the 2016 Graduate Teaching Fellowship from the College of Engineering, Texas A&M University. He has been a Visiting Researcher with Nokia Bell Labs, Duke University, and The Chinese University of Hong Kong. From 2009 to 2012, he was a Researcher with the TCS Innovation Labs, Bangalore.

Emina Soljanin (Fellow, IEEE) is currently a Professor with Rutgers University. Before moving to Rutgers in 2016, she was a (Distinguished) member of Technical Staff for 21 years in the Mathematical Sciences Research of Bell Labs. Her research interests are wide. Over the past quarter of the century, she has participated in numerous research and business projects, as diverse as power system optimization, magnetic recording, color space quantization, hybrid ARQ, network coding, data and network security, distributed systems performance analysis, and quantum information theory. She served as an Associate Editor for Coding Techniques, for IEEE Transactions on Information Theory, on the Information Theory Society Board of Governors, and in various roles on other journal editorial boards and conference program committees. She has been an outstanding alumnus of the Texas A&M School of Engineering, the 2011 Padovani Lecturer, a 2016–2017 Distinguished Lecturer, and the 2019 President of the IEEE Information Theory Society.

Alex Sprintson (Senior Member, IEEE) received the B.S. degree *(summa cum laude)*, M.S., and Ph.D. degrees in electrical engineering from Technion, in 1995, 2001, and 2003, respectively. From 2013 to 2019, he served as an Associate Editor for IEEE Transactions on Wireless Communications. He is currently a Faculty Member with the Department of Electrical and Computer Engineering, Texas A&M University. He is also serving as a rotating Program Director with the U.S. National Science Foundation (NSF). He has been a member of the Technical Program Committee of the IEEE Infocom from 2006 to 2020.