Latency Optimal Storage and Scheduling of Replicated Fragments for Memory Constrained Servers

Rooji Jinan, *Graduate Student Member, IEEE*, Ajay Badita, *Member, IEEE*, Pradeep Kiran Sarvepalli, and Parimal Parag, *Senior Member, IEEE*

***Abstract*— We consider the setting of a distributed storage system where a single file is subdivided into smaller fragments of same size which are then replicated with a common replication factor across servers of identical cache size. An incoming file download request is sent to all the servers, and the download is completed whenever the request gathers all the fragments. At each server, we are interested in determining the set of fragments to be stored, and the sequence in which fragments should be accessed, such that the mean file download time for a request is minimized. We model the fragment download time as an exponential random variable independent and identically distributed for all fragments across all servers, and show that the mean file download time can be lower bounded in terms of the expected number of useful servers summed over all distinct fragment downloads. We present deterministic storage schemes that attempt to maximize the number of useful servers. We show that finding the optimal sequence of accessing the fragments is a Markov decision problem, whose complexity grows exponentially with the number of fragments. We propose heuristic algorithms that determine the sequence of access to the fragments which are empirically shown to perform well.**

***Index Terms*— Distributed storage systems, mean download time, projective plane designs, replication storage codes, scheduling.**

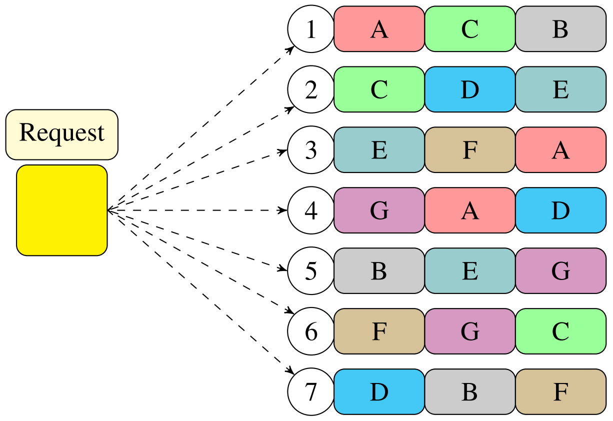

Fig. 1. An example of $\frac{3}{7}$-$(7,7,3)$ replication coded system, where a single file is fragmented into $V = 7$ fragments, and each fragment is repeated $R = 3$ times over $B = 7$ servers each with storage of $K = 3$ fragments. The corresponding block replication code is denoted by $(21, 7)$.

I. Introduction

THE recent years have seen a widespread deployment of distributed storage systems, consisting of large number of storage nodes. These nodes are prone to failures and unpredictable download times [2]. A primary challenge in such systems is to provide resilience against such events. One approach for improving the robustness of a distributed storage system is adding redundancy through error-correcting codes. It turns out that coded systems also offer fast access to the data due to parallelization gains. Latency redundancy trade-off has been studied for maximum distance separable (MDS) and replication codes in various articles [3]–[8].

For simplicity, we consider a single file of unit size. A distributed storage system of B servers that can each store α fraction of this file, is referred to as an α-B system. A single file of unit size divided into V fragments, encoded into VR fragments, and stored over an α-B system, is called an α-(B, V, R) coded storage scheme. This suggests that we can use (n, k) error-correcting codes which encode $k = V$ information symbols into $n = VR$ encoded symbols, such that the file can be recovered by downloading certain coded symbols. In replication coding, we replicate each of the V file fragments R times, and the file can be recovered by downloading a single replica of each fragment. On the other hand, in a (VR, V) MDS code the V file fragments are encoded into VR fragments, and the entire file can be reconstructed from any of the V encoded fragments. For an α-B system, it is assumed that per server storage is $K = \alpha V$ in terms of the fragments. Fig. 1 shows such a scheme for $\frac{3}{7}$-$(7,7,3)$ replication storage scheme.

Manuscript received October 4, 2020; revised August 3, 2021; accepted January 29, 2022. Date of publication February 24, 2022; date of current version May 20, 2022. This work was supported in part by the Science and Engineering Research Board, Department of Science and Technology, Government of India, under Grant DSTO-1677; in part by the Department of Telecommunications, Government of India, under Grant DOTC-0001; in part by the Robert Bosch Centre for Cyber-Physical Systems; and in part by the Centre for Networked Intelligence (a Cisco Corporate Social Responsibility (CSR) Initiative) at Indian Institute of Science, Bengaluru. An earlier version of this paper was presented in part at the 2021 IEEE International Symposium on Information Theory [1] [DOI: 10.1109/ISIT45174.2021.9517901]. *(Corresponding author: Parimal Parag.)*

Rooji Jinan is with the Robert Bosch Centre for Cyber-Physical Systems, Indian Institute of Science, Bengaluru, Karnataka 560012, India (e-mail: roojijinan@iisc.ac.in).

Ajay Badita was with the Indian Institute of Science, Bengaluru 560012, India. He is now with the IOTA Foundation, 10405 Berlin, Germany (e-mail: ajay.badita@iota.org).

Pradeep Kiran Sarvepalli is with the Department of Electrical Engineering, Indian Institute of Technology Madras, Chennai, Tamil Nadu 600036, India (e-mail: pradeep@ee.iitm.ac.in).

Parimal Parag is with the Department of Electrical Communication Engineering, Indian Institute of Science, Bengaluru, Karnataka 560012, India (e-mail: parimal@iisc.ac.in).

Communicated by C. Hollanti, Associate Editor for Coding and Decoding.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIT.2022.3152182.

Digital Object Identifier 10.1109/TIT.2022.3152182

A commonly studied α-(B, V, R) system is where per server storage $\alpha = \frac{1}{V}$ [2], [9], [10]. That is, the number of coded fragments on each of the B parallel server caches is $K = 1$. For this case it was shown that MDS codes provide optimal performance [10] in terms of the mean access delay. However, if we allow for an additional degree of freedom, namely a larger subpacketization of the fragments stored on the servers, then even non-MDS codes can become competitive. For instance, when $K > 1$ the staircase codes proposed in [11] for secure computation have been used to improve upon the MDS codes[1]. The mean download time of staircase codes was shown to be smaller than that of a $(B, B/R)$ MDS code in [12], [13].

Fixing the subpacketization at V fragments, the code rate at $1/R$, and the number of servers at B, it can be shown that among all (VR, V) codes stored over an α-B system, an MDS code has the smallest file download time for a class of distributions. Even though MDS codes are latency optimal, they have certain drawbacks. First, encoding and decoding require complex finite field arithmetic. Popular erasure decoding algorithms for MDS codes such as Reed-Solomon codes have a decoding complexity $O(V^2)$, see [14] for a recent overview. For a slow processing system, this could lead to a non-trivial decoding latency. Second, file sizes often change in storage systems with frequent writes [15]. In this case, the entire file has to be encoded again [16]. Third, to be able to code V fragments of a file into VR MDS coded symbols, the symbols must belong to a sufficiently large alphabet [17, Theorem 4.1]. This can be achieved by grouping multiple bits together in each file fragment. This puts a constraint that each fragment should be sufficiently large.

Block replication codes score well on all of these fronts. As the replication codes are binary, encoding and decoding of replication is trivial and the file sizes can be small. Further, the file size changes can be accommodated by change in the associated fragments and their replicas. This is reflected in widespread adoption [18]–[21] of replication codes in distributed systems. Furthermore, we will show that the latency performance of block replication codes becomes comparable to that of MDS codes with increase in either the number of fragments or the storage size per server. Hence, we focus on (VR, V) replication codes that offer good mean download time for an α-B system. For replication codes, we need to determine on which servers each of the replicated fragments should be stored. In addition, we need to consider the order in which the fragments are accessed at each server. *We consider the problem of optimal storage and access sequence of the replicated fragments at each of the B servers, such that the mean download time is minimized.* There is no obvious relation between the fragments stored on each of the servers and the mean download time. Therefore, determining what fragments should be stored on each of the servers appears to be a difficult problem as there are exponentially many ways to store the fragments on the servers. For a given fragment storage on each server, the problem of optimal access sequence can be posed as a finite horizon Markov Decision Problem (MDP), which can be solved by standard backward induction algorithm [22]. For the proposed MDP, this algorithm requires exponentially large memory in number of fragments, and cannot be implemented efficiently for large system parameters.

[1]We view the staircase codes as (BK, V) codes over a q-ary alphabet, where each server has K q-ary symbols. It is also possible to view them as (B, V) MDS codes over a q^K alphabet. See also [11, Theorem 1].

A. Related Work

Coding techniques have emerged as a popular technique to provide reliability in distributed storage systems with fault-prone network [2], [23]–[26]. Storage codes can also be designed to achieve additional objectives such as low repair bandwidth [2], [9], [18], [27], [28], low regeneration bandwidth [24], [29], [30], high locality [31], [32], low latency [33]–[36], among others. In this work, we are interested in low latency performance of distributed storage systems, by using codes. Specifically, we study replication codes where files are stored redundantly over the system.

It has been shown that redundant storage can reduce latency as well. In this case, a download request can be served in parallel by multiple servers storing the requested file [3], [7], [10], [36]–[41]. Trade-off between latency and cost of availing redundancy was observed empirically in [37], and subsequently studied theoretically in [7], [21], [34], [38], [40], [42], [43]. Two well studied file encoding strategies used in distributed systems with redundant storage are MDS coding [10], [23], [34], [42] and replication [19]–[21], [24]. It has been shown that MDS coding outperforms replication in mean file access latency [4], [10], [44].

In all these works, it was assumed that each server stores a single coded fragment of the file. In other words, the size of a file fragment is equal to the memory of the server. If we divide the file into smaller fragments than the memory available at the server, then we can store multiple file fragments on each server. This fact was exploited by [12], [13] to show that the mean access latency for Staircase codes can be smaller than MDS codes storing a single coded fragment on each server. In Staircase codes, the fragments stored on the servers are not all of equal size. We note that if the file is uniformly subfragmented and a larger code is used, then performance of MDS codes can also be improved.

Our work differs from these existing works in the following ways. We focus on replication codes for equal sized fragments of a single file stored over multiple servers, such that each server can store multiple fragments. Each server stores a different set of fragments introducing an asymmetry among the servers during the download process. The storage scheme has a direct impact on the mean download time. Furthermore, the sequence of access of fragments at each server affects the download time significantly. We address the problem of constructing good storage schemes as well as the access sequence at each server in this work. These aspects have not been explored in the literature to the best of our knowledge.

B. Main Contributions

We briefly summarize the main contributions of the paper.

i) We study replication codes for file fragments stored over distributed storage systems. Our study differs from previous work in that each server can store multiple fragments.

ii) We characterize the mean download time of a file when fragment download times are random and independent and identically distributed (*i.i.d.*) exponential, and find a lower bound in terms of the expected sum of number of useful servers for each fragment download.

iii) We provide bounds on the number of useful servers for any storage scheme in an α-B system employing (VR, V) replication code.

iv) We propose storage schemes that maximize the aforementioned lower bounds on number of useful servers.

v) We establish that finding the optimal fragment access sequence is an MDP. We propose efficient suboptimal algorithms that are easy to implement.

vi) We show that among all the (VR, V) codes stored on an α-B system, an MDS code minimizes the mean download time. In addition, we show that a (VR, V) replication code matches the MDS code performance when $\alpha \geqslant 1$, i.e. $K \geqslant V$.

vii) We propose a random fragment storage scheme along with a random scheduling policy for (VR, V) replication codes that performs competitively with respect to (VR, V) MDS codes when $K < V$, for large V.

viii) We support our analyses with numerical studies which provide additional insights into α-(B, V, R) coded storage schemes. They also illustrate the performance of various proposed algorithms.

A key takeaway from our work is that a good choice of storage scheme and access sequence can enable replication codes to be a practical alternative to MDS codes in situations where the implementation of MDS codes is not preferable.

C. Organization

We present the system model in Section II and the problem formulation in Section III. We provide universal performance bounds in Section IV, based on which we propose deterministic storage policies in Section V. Algorithms for fragment access sequence are studied in Section VI. A random storage scheme for replication codes together with a random scheduling policy is presented in Section VII. The performance of replication and MDS codes are compared in Section VIII. Empirical studies are provided in Section IX, and we conclude with some final remarks in Section X.

Notation: We briefly summarize the notation used throughout this article. We denote the set of first N consecutive positive integers by $[N] \triangleq \{1, \dots, N\}$, the set of positive integers by $\mathbb{N}$, the set of non-negative integers by $\mathbb{Z}_+$, the set of non-negative reals by $\mathbb{R}_+$. For a set A, we denote the collection of all subsets by 2^A, and the cardinality by $|A|$.

II. System Model

We consider the storage of a single file split into V distinct fragments on a finite number of servers B. Each of these servers are assumed to have identical storage capacity of K such fragments. We will initially consider $K \leqslant V$, and define

TABLE I
NOTATIONS FOR VARIOUS STORAGE PARAMETERS

Storage parameter	**Notation**
Number of file fragments	V
Number of servers	B
Storage capacity at each server	K
Replication factor for each fragment	R
Fragment sets at servers	$(S_b : b \in [B])$
Occupancy sets of fragments	$(\Phi_v : v \in [V])$

the storage capacity per server in terms of fraction of file fragments

$$\alpha \triangleq \frac{K}{V}. \tag{1}$$

We will see in Section VIII-B, that $K \geqslant V$ is a simpler case and can be studied independently. For (VR, V) block codes that encode V fragments into VR coded fragments, the *code rate* is $1/R$. We study (VR, V) replication codes where each file fragment is replicated R times, and R is called the *replication factor* of the code. The system should have sufficient storage capacity to store all VR fragments, and this requires that $VR \leqslant KB$, or equivalently $R \leqslant \alpha B$.

We assume any download request for the stored file is forked to all B servers. This request leaves the system only when it has collected all the fragments needed to decode the file. We further assume that only one symbol can be downloaded at a time from each server and the information stored on the servers are never lost. At each server $b \in [B]$, the request starts downloading stored fragments in some order. The fragment to be downloaded at each server depends on the fragments that have already been downloaded. The download time for the file is affected by which fragments are stored on each server, download time of each fragment, and the temporal sequence in which the fragments are downloaded. In the following sections, we consider each of these aspects in more detail.

A. Storage Model

For a specific code which encodes the V file fragments into V R encoded fragments, we refer to the mapping of fragments stored on each server as the *storage scheme*.

Definition 1 (Occupancy Set): The set of servers on which a fragment $v \in [V]$ is replicated is called the *occupancy set*, and is denoted by $\Phi_v \subseteq [B]$ such that $|\Phi_v| \leqslant R$.

The storage scheme is completely determined by the collection of occupancy sets $\Phi \triangleq (\Phi_v : v \in [V])$.

Definition 2 (Fragment Set): The set of fragments stored on a server $b \in [B]$ is called the *fragment set*, and is denoted by $S_b \subseteq [V]$, such that $|S_b| \leqslant K$ and

$$S_b \triangleq \{v \in [V] : b \in \Phi_v\}. \tag{2}$$

The notations used for the various storage parameters is summarized in Table I.

We classify the storage of (VR, V) codes on α-B systems as *completely utilizing* and *underutilizing*. A storage scheme is said to be completely utilizing if $\sum_{v\in[V]} |\Phi_v| = BK$ and underutilizing if $\sum_{v\in[V]} |\Phi_v| < BK$. If $VR < BK$, then an

α-B system is clearly underutilizing. A (VR, V) replication code with $VR = BK$ can still be underutilizing, if there exists a fragment v whose multiple replicas are stored on some server b. In this case, $|\Phi_v| < R$ and $|S_b| < K$. Here, the storage of extra replicas on the same server provide no parallelization benefit, and they are discarded as soon as one of the replicas gets downloaded. Therefore, we focus on storage schemes where each server stores at most a single replica of each fragment.

Definition 3: For (VR, V) replication coded file fragments stored on B servers with storage capacity of K fragments satisfying $VR = BK$, the *completely utilizing α-(V, R) replication storage ensemble* is defined as the collection

$$\mathcal{S} \triangleq \left\{ \Phi \in (2^{[B]})^{[V]} \middle| |\Phi_v| = R, |S_b| = K \text{ for all } v, b \right\}. \quad (3)$$

B. Fragment Download Time Model

The fragment download time at each server is modeled by a random variable that captures uncertainty due to network delays and server background processes [45]. We denote the fragment download time for fragment v at the server b by a nonnegative random variable T_{bv}. We assume that all fragments are of equal size and the servers are identical, in the sense that the marginal distribution of T_{bv} is identical for all fragments $v \in S_b$ at all servers $b \in [B]$. The background processes vary widely across the servers and with respect to time. Together with this observation and motivated by analytical tractability, we assume that the fragment download times T_{bv} are independent across servers and fragments. That is, we assume fragment download times to be *i.i.d.*, which is a popular assumption in the literature [3], [5], [10], [20], [40], [46].

It has been shown that shifted exponential distribution is a good model for the random download time in data center networks [8], [12], [40], [47], where the constant shift is the startup time for servers, and the memoryless part accounts for the uncertainty. In the case when the startup time is negligible when compared to the mean download time, exponential distribution is a good approximation for the download time distribution. Therefore, we also assume the common distribution function F for the random fragment download time to be exponentially distributed with rate μ, such that

$$F(x) \triangleq P(\{T_{bv} \leqslant x\}) = 1 - e^{-\mu x}, \quad \text{for all } x \geqslant 0.$$

C. Download Sequence and Scheduling

Recall that the request is completely serviced when all the V distinct fragments have been downloaded. Due to the stochastic nature of fragment download, the order in which the request downloads the fragments is random. We assume that the request only downloads the fragments which have not been downloaded before, and such download sequences are referred to as *minimal downloading sequences*. A minimal downloading sequence for replication codes consists of unique fragments. The ℓth downloaded fragment is denoted by v_ℓ, and the sequence of downloaded fragments until ℓth download is called download subsequence and denoted by $I_\ell \triangleq (v_1, \dots, v_\ell)$. Since all the downloaded fragments are unique, sometimes we regard the download subsequence I_ℓ as a set and ignore the ordering of fragments. The distinction between the set and the subsequence will be clear from the context.

Given the sequence of downloaded fragments, the process of determining which fragment is made available for downloading at each server is referred to as *scheduling*. We will only be interested in the class of scheduling policies that result in minimal download sequences, and refer to them as *work-conserving*. These scheduling policies do not waste the servers' work by downloading replicas of the fragments that have already been downloaded. In other words, the scheduler only selects the remaining fragments at each server. Therefore, after a fragment download, the request immediately stops downloading that fragment from other servers. Instead, it starts downloading the scheduled fragment among one of the remaining fragments at that server. After the ℓth download, the set of remaining fragments on a server $b \in [B]$ is denoted as

$$S_b^\ell \triangleq S_b \setminus I_\ell, \quad \ell \in [V]. \quad (4)$$

Definition 4 (Work-Conserving Scheduling): The work-conserving scheduling can be formally defined as a function $\Psi : \mathcal{S} \times 2^{[V]} \to [V]^{[B]}$, that selects one of the remaining fragments at each server b after ℓ downloads for a storage scheme $\Phi \in \mathcal{S}$. That is, for all $\ell \in \{0, \dots, V-1\}$, we have

$$\Psi(\Phi, I_\ell)(b) \in S_b^\ell, \quad \text{for all } b \in [B] \text{ such that } S_b^\ell \neq \emptyset. \quad (5)$$

We denote the restriction of work-conserving scheduling policy Ψ to a fixed storage scheme $\Phi \in \mathcal{S}$ by $\Psi_\Phi : 2^{[V]} \to [V]^{[B]}$ such that

$$\Psi_\Phi(I_\ell)(b) = \Psi(\Phi, I_\ell)(b).$$

Motivated by analytical tractability, we have assumed an idealized model of perfect and immediate cancellation at all parallel servers serving the same fragment.

III. Problem Formulation

In this section, we give a precise formulation of the main problem we study. Our main goal is to minimize the mean download time for a file stored using a storage scheme Φ from the completely utilizing α-(V, R) replication storage ensemble and work-conserving scheduling scheme Ψ, when the service times at each server are independent and exponentially distributed with rate μ. We denote the download time of ℓth distinct fragment v_ℓ by D_ℓ, where $D_0 \triangleq 0$ and $\ell \in \{0, \dots, V\}$. This indicates that a download subsequence I_ℓ of ℓ fragments has been downloaded at time D_ℓ. With this notation, the file download time is denoted by D_V.

If the request has downloaded all the fragments available at a server, the corresponding server is called *useless*. A server that has fragments not yet downloaded by the request, is called *useful*. We borrow this nomenclature of useful and useless servers from [10]. The set of useful servers after ℓth download is denoted by $U(I_\ell)$ and its cardinality by $N(I_\ell)$. More precisely, we have

$$U(I_\ell) \triangleq \bigcup_{v \notin I_\ell} \Phi_v, \quad N(I_\ell) \triangleq |U(I_\ell)|. \tag{6}$$

The request is being served by $N(I_\ell)$ parallel servers in the duration $[D_\ell, D_{\ell+1})$. From the independent and memoryless service assumption at all servers of rate μ, we have

$$\mathbb{E}\left[D_{\ell+1} - D_\ell \middle| I_\ell\right] = \frac{1}{N(I_\ell)\mu}.$$

The download time for V fragments can be written as the sum of download time of individual fragments, i.e. $D_V = \sum_{\ell=0}^{V-1}(D_{\ell+1} - D_\ell)$. From the linearity and the tower property of expectation, it follows that the mean download time averaged over all fragments is

$$\frac{1}{V}\mathbb{E}\left[D_V\right] = \frac{1}{V}\mathbb{E}\left[\sum_{\ell=0}^{V-1} \frac{1}{N(I_\ell)\mu}\right]. \tag{7}$$

We see that the mean download time depends on $U(I_\ell)$, the set of useful servers remaining after ℓth download, which in turn depends on the storage scheme Φ and the scheduling policy Ψ. We present the following lemma that provides us a lower bound on the mean file download time in terms of the sum of mean number of useful servers.

Lemma 1: For any positive random vector $X \in \mathbb{R}_+^V$, we have $\frac{1}{V}\mathbb{E}\left[\sum_{i=1}^V \frac{1}{X_i}\right] \geqslant \frac{V}{\sum_{i=1}^V \mathbb{E}[X_i]}$.

Proof: Since the arithmetic mean is always larger than the harmonic mean, we can write $\frac{1}{V}\sum_{i=1}^V \frac{1}{X_i} \geqslant V/(\sum_{i=1}^V X_i)$. Taking expectation on both sides, and applying Jensen's inequality [48] to the convex function $f(x) = \frac{1}{x}$ and positive random variable $\sum_{i=1}^V X_i$, we get the result. ■

Remark 1: It follows from Lemma 1 that the mean download time can be lower bounded as

$$\frac{1}{V}\mathbb{E}\left[D_V\right] \geqslant \frac{V}{\mu \sum_{\ell=0}^{V-1} \mathbb{E}\left[N(I_\ell)\right]}. \tag{8}$$

Remark 2: Suppose that the random download times at the servers are *i.i.d.* having shifted exponential distribution with rate parameter μ and shift parameter $\theta \geqslant 0$. Then, we will get $\frac{1}{V}\mathbb{E}\left[D_V\right] = \theta + \frac{1}{V}\mathbb{E}\left[\sum_{\ell=0}^{V-1} \frac{1}{\mu N(I_\ell)}\right]$. Applying Lemma 1 to the second term, we get

$$\frac{1}{V}\mathbb{E}\left[D_V\right] \geqslant \theta + \frac{V}{\mu \sum_{\ell=0}^{V-1} \mathbb{E}\left[N(I_\ell)\right]}.$$

In some important settings, the sum $\sum_\ell \mathbb{E}\left[N(I_\ell)\right]$ is analytically more tractable when compared to $\sum_{\ell=0}^{V-1} \mathbb{E}\left[1/N(I_\ell)\right]$. Motivated by this fact, instead of minimizing the mean download time in Eq. (7), we minimize the lower bound in Eq. (8), which is then equivalent to maximizing $\sum_{\ell=0}^{V-1} \mathbb{E}\left[N(I_\ell)\right]$. Guided by this observation, we now pose the following problem.

Problem 1: Find the storage scheme Φ in completely utilizing α-(V, R) replication storage ensemble and restriction Ψ_Φ of work-conserving scheduling policy Ψ to this storage scheme, that maximizes the mean number of useful servers averaged over all fragments, i.e.

$$(\Phi^*, \Psi^*_{\Phi^*}) = \arg\max_{(\Phi, \Psi_\Phi)} \frac{1}{V}\sum_{\ell=0}^{V-1} \mathbb{E}\left[N(I_\ell)\right].$$

We divide this problem into two subproblems. The first subproblem is to find the optimal scheduling policy given a fixed storage scheme.

Problem 2: 1-A Find the optimal work-conserving scheduling policy restricted to a fixed completely utilizing α-(V, R) replication storage scheme Φ, i.e.

$$\Psi^*_\Phi = \arg\max_{\Psi_\Phi} \frac{1}{V}\sum_{\ell=0}^{V-1} \mathbb{E}\left[N(I_\ell)\right].$$

The optimal work-conserving scheduling policy Ψ^* is the collection of restrictions Ψ^*_Φ for all storage schemes $\Phi \in \mathcal{S}$.

The second subproblem is to find the optimal storage scheme given a fixed scheduling policy.

Problem 3: 1-B Given a work-conserving scheduling policy Ψ, find the optimal completely utilizing α-(V, R) replication storage scheme $\Phi \in \mathcal{S}$, i.e.

$$\Phi^\dagger(\Psi) = \arg\max_{\Phi \in \mathcal{S}} \frac{1}{V}\sum_{\ell=0}^{V-1} \mathbb{E}\left[N(I_\ell)\right].$$

By solving Problem 2 for each storage scheme $\Phi \in \mathcal{S}$, we can find the optimal work-conserving scheduling policy Ψ^*. Subsequently, we can obtain the optimal storage scheme $\Phi^* = \Phi^\dagger(\Psi^*)$ by solving Problem 3 for the optimal scheduling Ψ^*. It follows that if we can solve the above two sub-problems, then we can find the optimal solution $(\Phi^*, \Psi^*_{\Phi^*})$ to Problem 1.

It turns out that Problem 2 can be posed as an MDP, for a given storage scheme Φ. This MDP suffers from the well-known curse of dimensionality [49], and becomes intractable for large values of number of fragments. We propose heuristic scheduling algorithms that are computationally efficient. These algorithms are empirically shown to have a good performance for given storage schemes.

In contrast to Problem 2, it is not clear at the outset, how to efficiently solve Problem 3. A brute force way of solving this problem would be to compute the mean download time for all storage schemes $\Phi \in \mathcal{S}$ under the optimal scheduling policy Ψ^*_Φ, or its surrogate suboptimal heuristic algorithms, and searching among all storage schemes. Since this brute force search is computationally expensive, we propose an alternative suboptimal approach that involves two steps.

Step 1. We first find universal lower bounds $L_1^\Phi(\ell), L_2^\Phi(\ell)$ for the number of useful servers $N(I_\ell)$, such that for thresholds $0 < \nu_1, \nu_2 \leqslant V - 1$,

$$N(I_\ell) \geqslant \begin{cases} L_1^\Phi(\ell), & \ell < \nu_1, \\ L_2^\Phi(\ell), & \ell \geqslant \nu_2. \end{cases} \tag{9}$$

The lower bounds depend only on the storage scheme Φ and not on the specific work-conserving scheduling policy Ψ.

Step 2. We next find storage schemes that maximize the lower bounds $L_1^\Phi(\ell)$ and $L_2^\Phi(\ell)$. The resulting storage schemes behave well even with the worst possible work-conserving scheduling policy.

IV. Performance Bounds

In this section, we study the aggregate number of useful servers for completely utilizing α-(V,R) replication storage schemes $\Phi \in \mathcal{S}$, in conjunction with a work-conserving scheduling policy Ψ. In particular, we provide bounds on the number of useful servers $N(I_\ell)$ after each download $\ell \in \{0,\dots,V-1\}$.

A. Upper Bound on $N(I_\ell)$

First, we prove a simple upper bound for the number of useful servers for any α-(V,R) storage scheme.

Theorem 1: For a completely utilizing α-(V,R) replication storage scheme $\Phi \in \mathcal{S}$ defined in Eq. (3), the number of useful servers $N(I_\ell)$ after ℓ downloads is upper bounded in terms of $m \triangleq \lceil B/R \rceil$,

$$N(I_\ell) \leqslant \begin{cases} B, & \ell \leqslant V-m \\ (V-\ell)R, & \ell > V-m. \end{cases} \tag{10}$$

Proof: From the construction of a completely utilizing α-(V,R) storage scheme, it follows that all servers are useful before any download is initiated, i.e. $N(I_0) = B$. Further, no servers are useful after all fragments have been downloaded, i.e. $N(I_V) = 0$. Taking cardinality of the set of useful servers given in Eq. (6), we get

$$N(I_\ell) = |\cup_{v \notin I_\ell} \Phi_v| \leqslant \sum_{v \notin I_\ell} |\Phi_v| = (V-\ell)R.$$

Since the number of useful servers cannot exceed the total number of servers B, we get $N(I_\ell) \leqslant \min\{B,(V-\ell)R\}$ for any ℓ. We verify that $B \leqslant (V-\ell)R$ if and only if $m \leqslant V-\ell$, and the result follows. ■

From Theorem 1, we can obtain an upper bound on the number of useful servers averaged over the number of fragments and the number of servers.

Remark 3: Recall that for a completely utilizing α-(V,R) replication storage scheme $B/R = V/K = 1/\alpha$. When B/R is an integer, summing up both sides of Eq. (10), and dividing both sums by the product BV, we obtain

$$\frac{1}{BV}\sum_{\ell=0}^{V-1} N(I_\ell) \leqslant 1 - \frac{(m+1)}{2V}. \tag{11}$$

This gives us a normalized upper bound on the sum of the number of useful servers.

Remark 4: Note that this upper bound is true for all completely utilizing α-(V,R) replication storage schemes and all work-conserving scheduling policies. That is,

$$\sup_{\Phi \in \mathcal{S}} \sup_{\Psi_\Phi} \frac{1}{BV}\sum_{\ell=0}^{V-1} N(I_\ell) \leqslant 1 - \frac{(m+1)}{2V}.$$

Remark 5: Consider an underutilizing α-(V,R) replication storage scheme where each fragment $v \in [V]$ is assumed to be replicated on R_v servers, where $R_v \leqslant R$ and $\frac{1}{V}\sum_{v\in[V]} R_v < R$. Since, the number of useful servers before the commencement of download can only be smaller than or equal to B and $R_v \leqslant R$ for all $v \in [V]$, following the exact steps as above shows that the above given upper bound holds for an underutilizing α-(V,R) replication storage scheme.

B. Lower Bound on $N(I_\ell)$

Next, we proceed to find a lower bound for the number of useful servers $N(I_\ell)$ after ℓ downloads for a completely utilizing α-(V,R) storage scheme. To this end, we define two important properties of a storage scheme $\Phi \in \mathcal{S}$. For servers $a,b \in [B]$ and fragments $v,w \in [V]$, we define the maximum overlap of fragment sets and occupancy sets as

$$\tau_M \triangleq \max_{a \neq b} |S_a \cap S_b|, \qquad \lambda_M \triangleq \max_{v \neq w} |\Phi_v \cap \Phi_w|. \tag{12}$$

In the initial stages of download when the number of fragments downloaded is less than $O(K^2)$, the number of useful servers is primarily determined by the overlap between the fragment sets. More precisely, we have the following lower bound on $N(I_\ell)$.

Theorem 2: Consider a completely utilizing α-(V,R) replication storage scheme Φ with the maximum overlap of fragment sets τ_M, as defined in Eq. (12). Let $\ell_i \triangleq iK - i(i-1)\frac{\tau_M}{2}$ for $i \in \left\{0,1,\dots,\lfloor \frac{K}{\tau_M} \rfloor\right\}$, then the number of useful servers after ℓ downloads can be lower bounded as

$$\begin{aligned} N(I_\ell) &\geqslant L_1^\Phi(\ell) \\ &\triangleq \sum_{i=0}^{\lfloor K/\tau_M \rfloor} (B-i)\mathbb{1}_{\{\ell_i \leqslant \ell < \ell_{i+1}\}}, \text{ for } \ell < \nu_1 \triangleq \ell_{\lfloor K/\tau_M \rfloor + 1}. \end{aligned} \tag{13}$$

Proof: For $i \leqslant \frac{K}{\tau_M}$, we will show that $N(I_\ell) \geqslant B - i$ for $\ell < \ell_{i+1}$ by contradiction. We assume that the number of downloaded fragments $\ell < \ell_{i+1}$, and the number of useful servers $N(I_\ell) < B - i$. That is, the number of useless servers is $B - N(I_\ell) > i$. Then, there exist $(i+1)$ servers $\{b_1,\dots,b_{i+1}\} \subset [B]$, which are no longer useful after ℓ downloads. This implies that the union of their fragment sets $\cup_{j=1}^{i+1} S_{b_j}$ is included in the downloaded fragment set I_ℓ. Since any two servers can have at most τ_M file fragments in common, jth server has at least $(K-(j-1)\tau_M)$ fragments distinct from the fragments stored in first $(j-1)$ servers. Thus,

$$\ell \geqslant \left|\cup_{j=1}^{i+1} S_{b_j}\right| \geqslant \sum_{j=0}^{i}(K - j\tau_M) = \ell_{i+1}.$$

This contradicts our assumption. It follows that for each $i \in \left\{0,\dots,\lfloor \frac{K}{\tau_M} \rfloor\right\}$, the largest lower bound for $N(I_\ell)$ in the interval $\ell_i \leqslant \ell < \ell_{i+1}$ is $(B-i)$. Thus, we get the result. ■

Remark 6: From Theorem 2, we observe that for any $i \in \{0,\dots,\lfloor K/\tau_M \rfloor\}$, the derived lower bound on the number of useful servers after ℓ downloads is $L_1^\Phi(\ell) = B - i$ for $\ell_i \leqslant \ell < \ell_{i+1}$. Maximizing the lower bound $L_1^\Phi(\ell)$ is equivalent to maximizing the intervals $\{\ell_i,\dots,\ell_{i+1}-1\}$. From Theorem 2, it follows that the interval duration is $\ell_{i+1} - \ell_i = K - i\tau_M$. We observe that the interval durations are maximized for the overlap parameter $\tau_M = 1$.

We can also give a lower bound on the number of useful servers when most of the fragments have been downloaded. In this case the number of useful servers is determined by the overlap between occupancy sets. By considering the largest possible overlap of the occupancy sets, we can show the following lower bound on $N(I_\ell)$.

Theorem 3: Consider a completely utilizing α-(V,R) replication storage scheme $\Phi \in \mathcal{S}$ with the maximum overlap of occupancy sets λ_M, as defined in Eq. (12). Then, the number of useful servers is lower bounded as

$$N(I_\ell) \geqslant L_2^\Phi(\ell) \triangleq (V-\ell)\left(R-(V-\ell-1)\frac{\lambda_M}{2}\right), \quad (14)$$

for $\ell \geqslant \nu_2 \triangleq V - \lfloor R/\lambda_M \rfloor - 1$.

Proof: After ℓ downloads, the set of downloaded fragments is I_ℓ, and the set of remaining fragments is $[V] \setminus I_\ell$. Since $\ell = V - i$, we denote the set of remaining fragments as $\{w_1, \dots, w_i\}$. Recall that the set of useful servers is the union of servers storing the remaining fragments $U(I_\ell) = \cup_{v \notin I_\ell} \Phi_v = \cup_{j=1}^{i} \Phi_{w_j}$. We can write the number of useful servers as

$$N(I_\ell) = \left|\cup_{j=1}^{i} \Phi_{w_j}\right| = \sum_{j=1}^{i}\left(\left|\Phi_{w_j}\right| - \left|\cup_{r=1}^{j-1}(\Phi_{w_j} \cap \Phi_{w_r})\right|\right).$$

Since any two occupancy sets can have at most λ_M servers in common, we get $N(I_\ell) \geqslant \sum_{j=1}^{i}(R-(j-1)\lambda_M)$, and the result follows. ■

Remark 7: We observe from Theorem 3 that $L_2^\Phi(\ell)$ is maximized for overlap parameter $\lambda_M = 1$.

When the maximum overlap of fragment sets is $\tau_M = 1$, then we have the following lower bound on the number of useful servers.

Lemma 2: For a completely utilizing α-(V,R) replication storage scheme with the maximum overlap of fragments τ_M set to be 1, the number of useful servers $N(I_\ell)$ satisfies

$$N(I_\ell) \geqslant N(I_{\ell-1}) - \min\left\{R, \left\lfloor \frac{\ell-1}{K-1} \right\rfloor\right\}.$$

Proof: We denote the set of servers that turn useless after ℓth download by $G_\ell \triangleq U(I_{\ell-1}) \setminus U(I_\ell)$, and its cardinality by $|G_\ell| = N(I_{\ell-1}) - N(I_\ell)$. Each server $b \in G_\ell$ has the ℓth downloaded fragment v_ℓ, and all its remaining fragments have been downloaded in first $(\ell-1)$ downloads. Further, since the maximum overlap of fragment sets for these storage schemes is $\tau_M = 1$, it follows that the sets $(S_b \setminus \{v_\ell\} : b \in G_\ell)$ are disjoint. That is, we can write

$$\cup_{b \in G_\ell}(S_b \setminus \{v_\ell\}) \subseteq I_{\ell-1}.$$

Since $|I_{\ell-1}| = \ell - 1$ and $|S_b \setminus \{v_\ell\}| = K-1$ for all $b \in G_\ell$, we get $|G_\ell| \leqslant \frac{\ell-1}{K-1}$. Further noting that the ℓth downloaded fragment can occur on a maximum of R servers, we obtain the result. ■

V. Deterministic Placement Schemes for Replication Codes

In this section, we establish a connection between storage schemes and combinatorial designs. This connection allows us to systematically construct new storage schemes for a given α-B system. As discussed in Section III, instead of the objective function $\frac{1}{V}\sum_{\ell=0}^{V-1}\mathbb{E}[N(I_\ell)]$, we focus on the universal lower bounds $L_1^\Phi(\ell)$ and $L_2^\Phi(\ell)$ defined in Eq. (9), on the number of useful servers $N(I_\ell)$. From Remark 6 and Remark 7 in Section IV, we observe that the overlap parameter $\tau_M = 1$ maximizes the lower bound $L_1^\Phi(\ell)$ in the region $\ell < \nu_1$, and the overlap parameter $\lambda_M = 1$ maximizes the lower bound $L_2^\Phi(\ell)$ in the region $\ell \geqslant \nu_2$. The connection to designs allows us to optimize the storage schemes for λ_M and in some cases τ_M as well. We then study the performance of a particular class of storage schemes with desired overlap parameters, constructed from a combinatorial design called projective plane. Numerical studies are reported in Section IX.

A. Storage Schemes From Combinatorial Designs

In this section, we establish a correspondence between storage schemes and designs. Then using this correspondence we propose storage schemes from designs, with desirable overlap parameters τ_M and λ_M.

Definition 5 (Design): A design is a pair $(\mathcal{P}, \mathcal{B})$ satisfying the following conditions:

D1) $\mathcal{P}$ is a set of elements called points.

D2) $\mathcal{B}$ is a collection of nonempty subsets of $\mathcal{P}$ called blocks.

Remark 8: Every completely utilizing α-(V,R) replication storage scheme corresponds to a design $([V], \mathcal{B})$ where $|\mathcal{B}| = B = \frac{R}{\alpha}$ and

$$\mathcal{B} = \{S_b \subseteq [V] : b \in [B]\}.$$

Conversely, every design $(\mathcal{P}, \mathcal{B})$ leads to a replication storage scheme with $|\mathcal{P}|$ fragments stored on $B = |\mathcal{B}|$ servers, where bth server is storing the fragments indexed by the bth block of $\mathcal{B}$.

In addition, if every block has the same size K, and every point occurs R times, then we say that R is the replication number. From such a design, we can obtain an α-$(|\mathcal{P}|, R)$ replication storage scheme where $\alpha = R/|\mathcal{B}|$.

The class of designs that are suitable for completely utilizing α-(V,R) replication storage schemes, where all the servers have the same storage capacity, are the so-called t-designs. A design $(\mathcal{P}, \mathcal{B})$ is said to be a t-design with parameters t-(V,K,λ) if

1) there are V points in $\mathcal{P}$,
2) every block in $\mathcal{B}$ contains exactly K points,
3) every t-subset of $\mathcal{P}$ is contained exactly in λ blocks of $\mathcal{B}$.

The following result is well known from design theory, see for instance [50].

Proposition 1 ([50]): For every t-(V,K,λ) design with B blocks and replication number R, we have

$$BK = VR, \quad (15)$$

$$B\binom{K}{t} = \lambda\binom{V}{t}, \quad (16)$$

From the one to one correspondence between designs and replication storage schemes shown in Remark 8, it follows that the set of blocks in which a point p appears is precisely

TABLE II
CORRESPONDENCE BETWEEN DESIGNS AND STORAGE CODES

t-(V, K, λ) designs to codes	
Design parameter	**Storage parameter**
$\mathcal{P}$: Points	$[V]$: File fragments
$\mathcal{B}$: Blocks	$(S_b : b \in [B])$: Fragment sets at servers
$\|\mathcal{P}\|$: Number of points	V: Number of file fragments
$\|\mathcal{B}\|$: Number of blocks	B: Number of servers
K: Size of each block	K: Storage capacity at each server
R: Replication factor for each point	R: Replication factor for each fragment

the occupancy set of the fragment corresponding to point p. Further, the total number of blocks that contain the point p is the replication factor R of the corresponding fragment. In our setting, the points are the fragments, and the blocks are the set of fragments on each server. Table II summarizes the mapping between design and storage parameters.

In practical systems, typically, the number of servers and the storage capacity at each server is fixed. In addition, Eq. (15) tells us that among V and R, only one can be chosen independently. That is, among the two design parameters, the number of fragments into which each file is fragmented and the replication factor, choosing one determines the other one. Similarly, having chosen V, the parameter λ is completely known for any t-design.

In general, it is an open question whether a t-design with a given set of parameters exists and if it exists how to construct that design. For some specific parameters, there are explicit constructions of designs. A popular and well studied case is where $t = 2$. These designs are called balanced incomplete block designs (BIBDs). Then, from Eq. (16), we have

$$BK(K-1) = \lambda V(V-1), \tag{17}$$

and using Eq. (15) we obtain

$$\lambda = \frac{R(K-1)}{(V-1)}. \tag{18}$$

A special case of BIBDs of interest is where the number of points is equal to the number of blocks. In other words, the number of servers is equal to the number of fragments. Such BIBDs are also called symmetric BIBDs. Then, from Eq. (15), we note that the replication factor is equal to the memory at each server. Further, we also have the property that any two distinct blocks intersect in λ points, see [50, Theorem 2.2]. An important symmetric design is the projective plane for which explicit constructions are known.

Another well known 2-design for which an explicit construction is known is the affine plane which can be obtained from a projective plane. The memory requirements at each server scale as $O(\sqrt{B})$ for storage schemes derived from projective planes (and affine planes) with B servers. If we are interested in storage schemes with a fixed size of memory, then we could consider constructing a storage scheme from a Steiner triple systems [50]. They are also 2-designs, but with the block size fixed at $K = 3$. In the following table we summarize the storage schemes from the designs we discussed above. As can be seen from the Table III, there is a flexibility in the choice of the design parameters allowing us to construct storage schemes for various α-B systems. Additional storage schemes can be constructed using results from design theory. For instance a t-(V, K, λ) design implies the existence of $(t-i)$-$(V-i, K-i, \lambda)$ design for $i < t$, see [50, Theorem 9.2]. Therefore a $\frac{K}{V}$-(V, R) storage scheme obtained from a t-(V, K, λ) design implies a $\frac{K-i}{V-i}$-$(V-i, R')$ storage scheme where $R' = \lambda\frac{\binom{V-i-1}{t-i-1}}{\binom{K-i-1}{t-i-1}}$ and $i < t$.

TABLE III
PARAMETERS OF VARIOUS COMPLETELY UTILIZING α-(V, R) REPLICATION STORAGE SCHEMES FROM DESIGNS

t-**design**	**Parameters**	α-(V, R) **system**
General t-design	t-(V, K, λ)	$\frac{K}{V}$-(V, R) $B = \lambda\binom{V}{t}/\binom{K}{t}$ $R = \lambda\binom{V-1}{t-1}/\binom{K-1}{t-1}$
BIBD	2-(V, K, λ)	$\frac{K}{V}$-(V, R) $B = VK/R,$ $R = \lambda(V-1)/(K-1)$
Symmetric BIBD	2-(V, K, λ)	$\frac{K}{V}$-(V, K) $B = V,\ R = \lambda(V-1)/(K-1)$
Projective plane	2-$(n, q+1, 1)$, $n = q^2+q+1$ $q = p^m$, p prime	$\frac{q+1}{n}$-$(n, q+1)$ $B = n,\ R = q+1$ $K = q+1$
Affine plane	2-$(q^2, q, 1)$ $q = p^m$, p prime	$\frac{q}{q^2}$-$(q^2, q+1)$ $B = q^2+q,\ K = q$
Steiner triple system	2-$(V, 3, 1)$ $V \equiv 1, 3 \bmod 6$	$\frac{3}{V}$-$(3V/R, V, R)$ $R = (V-1)/2$

B. Storage Schemes From Projective Planes

As mentioned earlier, Theorems 2 and 3 motivate us to construct storage schemes where λ_M and τ_M are small. Since they are both nonnegative, one might try to make them both zero. However, $\lambda_M = 0$ implies that $K = 1$ which has been studied extensively. Similarly, $\tau_M = 0$ implies that $R = 1$ in which case there is no redundancy. For these reasons, we do not study these two cases in this paper. The next possible choice would be $\lambda_M = 1$ and $\tau_M = 1$. Note, that $\lambda_M = 1$ implies that the maximum occupancy overlap is less than or equal to one. For simplicity, we consider the symmetric case, where for any two distinct fragments u, v we have $|\Phi_u \cap \Phi_v| = 1$. Such a storage scheme immediately leads us to Steiner systems which are BIBDs with $\lambda = 1$. This motivates the study of storage schemes from such 2-designs. If we similarly restrict that the fragment sets also satisfy a similar overlap property, i.e. $|S_a \cap S_b| = 1$ for distinct blocks a, b, then such a BIBD must also be symmetric in that number of blocks is identical to the number of fragments [50, Corollary 2.5]. A well studied class of symmetric BIBDs is that of projective planes.

A projective plane is a 2-$(q^2+q+1, q+1, 1)$ design and can be constructed from $\mathbb{F}_q^3$, where q is power of a prime and $\mathbb{F}_q$ is a finite field with q elements. From these designs we obtain an α-$(q^2+q+1, q+1)$ replication storage scheme,

TABLE IV
A $\frac{3}{7}$-$(7,3)$ REPLICATION STORAGE SCHEME BASED ON A PROJECTIVE PLANE

Placement of fragments							
Server	1	2	3	4	5	6	7
S_b	1,2,3	3,4,5	1,5,6	1,4,7	2,5,7	3,6,7	2,4,6

TABLE V
A $\frac{3}{7}$-$(7,3)$ REPLICATION STORAGE SCHEMES BASED ON CYCLIC SHIFT

Placement of fragments							
Server	1	2	3	4	5	6	7
S_b	1,2,3	2,3,4	3,4,5	4,5,6	5,6,7	6,7,1	7,1,2

where $\alpha = \frac{(q+1)}{(q^2+q+1)}$. A $\frac{3}{7}$-$(7,3)$ replication storage scheme constructed from a projective plane is shown in Table IV.

If we relax the constraint that two blocks do not necessarily intersect in λ_M points, then also it is possible to construct a storage scheme from a 2-design with $\lambda_M = \tau_M = 1$. Specifically, the affine planes lead to storage schemes with these parameters. These are 2-$(q^2, q, 1)$ designs. From these, we can construct $\frac{1}{q}$-$(q^2, q+1)$ replication storage schemes.

Remark 9: Recall that the overlap parameters $\tau_M = \lambda_M = 1$ for any α-(V,R) replication storage scheme derived from a 2-$(q^2+q+1, q+1, 1)$ design. Aggregating results from Theorem 2 for $\tau_M = 1$, Theorem 3 for $\lambda_M = 1$, and Lemma 2 for $\tau_M = 1$, we can obtain a lower bound on the number of useful servers for number of downloads $\ell \in \{0, \dots, V-1\}$.

C. Cyclic Shift Based Storage

We conclude this section by considering a simple α-(V,R) storage scheme. The set of servers storing replicas of a fragment $v \in [V]$ are cyclically shifted by v. More precisely, the occupancy set of a fragment v is given by

$$\Phi_{v+R-1} = \{v, v+1, \cdots, v+R-1\}, \qquad (19)$$

where the addition is modulo B whenever the sum exceeds B. We observe that $|\Phi_v \cap \Phi_{v+j}| = |\Phi_v \cap \Phi_{v-j}| = R-j$ for $j \in [R-1]$, $v \in [V]$ where the addition and subtraction is modulo V whenever the sum exceeds V or difference go below 1. Thus, the maximum overlap between the occupancy sets $\lambda_M = R-1$.

In addition, we can write the fragment set at server $b \in [B]$ for this storage scheme as

$$S_b = \{b, b+1, \dots, b+K-1\}, \qquad (20)$$

where the addition is modulo V whenever the sum exceeds V. Here, we observe that $|S_{b+j} \cap S_b| = |S_{b-j} \cap S_b| = K-j$ for $j \in [K-1]$, $b \in [B]$ where the addition and subtraction is modulo B whenever the sum exceeds B or difference go below 1. Thus, the maximum overlap between the servers $\tau_M = K-1$. An example is given in Table V.

While the maximum overlap parameters for this scheme are $\tau_M = K-1$ and $\lambda_M = R-1$, the overlap parameters have a wide spread. Any server will have a fragment set overlap i, where $i \in [K-1]$, with exactly two other servers and will have zero overlap with the remaining $\max(0, B-2K+1)$ servers. Similarly, the occupancy set of each fragment will have an overlap of j, where $j \in [R-1]$, with the occupancy sets of exactly two other fragments and will have zero overlap with the occupancy sets of remaining $\max(0, V-2R+1)$ fragments.

From the bounds in Theorems 2 and 3, we expect that cyclic shift based storage schemes should perform somewhat poorly compared to the projective plane designs based schemes presented previously, as we will corroborate in Section IX. If $B = V$, the fragments in the ith locations of the servers can be arranged to be a permutation of $[V]$ giving a natural nonadaptive scheduling policy. Later, in Section IX we will study the performance of this scheme along with an adaptive scheduling policy to highlight the importance of scheduling. The worst case overlap parameters for cyclic shift based storage schemes are very large, and this is reflected in their large mean download time with nonadaptive scheduling. However, the performance is not only affected by λ_M and τ_M but also by the spread of the occupancy set overlaps and fragment set overlaps. Taking advantage of this fact we can improve the performance of this scheme with adaptive scheduling. Specifically, the adaptive scheduling algorithms exploit the spread of the overlap parameters, to drive the system state towards good residual fragment sets with small overlap.

VI. WORK-CONSERVING SCHEDULING

As already discussed in Section III, Problem 2 can be reformulated as an MDP, given a completely utilizing α-(V,R) storage scheme. In view of the complexity of the MDP, we propose two classes of efficient suboptimal work-conserving scheduling algorithms. One class of algorithms are nonadaptive while the other are adaptive. The nonadaptive algorithms fix the schedule at each server ahead of the file download. In adaptive algorithms, the schedule of remaining fragments to be downloaded at each server is causally aware of the sequence of fragment downloads. We show that both classes of algorithms offer good performance through numerical studies. As will be seen in Section IX, adaptive algorithms can give a better performance than the nonadaptive ones. Adaptive work-conserving scheduling provides more flexibility in the download process which can be exploited to reduce the mean download time. Nonadaptive scheduling policies may be preferred in some cases, when adaptive fragment selection incurs non-negligible delay.

A. Nonadaptive Work-Conserving Scheduling

Given a completely utilizing α-(V,R) replication storage scheme Φ, we can find the fragment set S_b at each server $b \in [B]$. In the nonadaptive case, the scheduling decisions are embedded in the placement order $\pi_\Phi^b : [K] \to S_b$ of fragment set at each server $b \in [B]$, and is fixed prior to the commencement of download. A nonadaptive work-conserving scheduling is defined by the collection of placement order $\pi_\Phi \triangleq (\pi_\Phi^b : b \in [B])$. Given the set of downloaded fragments I_ℓ after ℓ downloads, the fragment to be downloaded from a server b is the first residual fragment stored at this server in the order of placement. This fragment is denoted by $\pi^b_\Phi(k^b_{\ell+1})$, where

$$k^b_{\ell+1} \triangleq \inf\{k \in [K] : \pi_b(k) \notin I_\ell\}, \quad b \in U(I_\ell).$$

A placement order π induces a scheduling policy Ψ such that $\Psi_\Phi(I_\ell)(b) = \pi^b_\Phi(k^b_{\ell+1})$. Finding a nonadaptive work-conserving scheduling policy is equivalent to finding a placement order. Therefore, given an α-(V,R) replication storage scheme $\Phi \in \mathcal{S}$, the optimal nonadaptive work-conserving scheduling policy is given by

$$\pi^\dagger(\Phi) = \arg\max_{\pi(\Phi)} \frac{1}{V}\sum_{\ell=0}^{V-1} \mathbb{E}\left[N(I_\ell)\right].$$

One way to find the optimal nonadaptive work-conserving scheduling policy is to search over all possible placement orderings. This search is highly computationally intensive, and it is not clear how to efficiently find the optimal nonadaptive work-conserving scheduling policy. As such, we discuss heuristic nonadaptive work-conserving scheduling policies in Section IX that attempt to maximize the mean number of useful servers, aggregated over all downloads.

B. Adaptive Work-Conserving Scheduling

Recall that, for a fixed completely utilizing α-(V,R) replication storage scheme Φ, the corresponding adaptive scheduling policy is a map $\Psi_\Phi : 2^{[V]} \to [V]^{[B]}$. In particular, the policy schedules a residual fragment $\Psi_\Phi(I_\ell)(b) = w^b_{\ell+1} \in S^\ell_b$ on each useful server $b \in U(I_\ell)$, after ℓ downloads. That is,

$$\Psi_\Phi(I_\ell) : b \mapsto S^\ell_b, \text{ for all } b \in U(I_\ell), \text{ all } \ell \in \{0,1,\dots,V-1\}.$$

We first show that the evolution of downloaded fragments can be modelled as a Markov chain for any work-conserving scheduling policy Ψ_Φ given a fixed completely utilizing α-(V,R) replication storage scheme Φ. We then pose optimal dynamic scheduling problem defined in Problem 2 as an MDP.

Let $X_t \subseteq [V]$ be the set of downloaded fragments at time t. We can write ℓth download instants in terms of the process $X \triangleq (X_t \in 2^{[V]} : t \in \mathbb{Z}_+)$ as

$$D_\ell = \inf\{t > 0 : |X_t| = \ell\}.$$

At the ℓth download instant D_ℓ, the set of downloaded fragments is $I_\ell = X_{D_\ell}$. Further, the set of useful servers at this instant is $U(I_\ell)$. At time D_ℓ, the fragment scheduled on any useful server $b \in U(I_\ell)$ is the scheduling decision $\Psi_\Phi(I_\ell)(b) \in S^\ell_b$. Since the process X is piecewise constant and only changes at decision epochs, we are interested in the associated discrete time process sampled at the decision epochs $\{D_0, D_1, \dots, D_{V-1}\}$. Defining $Y_\ell \triangleq X_{D_\ell}$ for all $\ell \in \{0,1,\dots,V-1\}$, we can write the sampled process as $Y \triangleq (Y_\ell : \ell \in \{0,1,\dots,V-1\})$.

Lemma 3: For a fixed completely utilizing α-(V,R) replication storage scheme Φ, scheduling policy Ψ_Φ, and *i.i.d.* exponential fragment download times, the continuous time process $X = (X_t \in 2^{[V]} : t \in \mathbb{Z}_+)$ is Markov. Hence, the associated sampled process Y is a discrete time Markov chain with the transition probabilities given by

$$p_{I_\ell, I_\ell \cup \{v\}} = \frac{1}{N(I_\ell)} \sum_{b \in U(I_\ell)} \mathbb{1}_{\{\Psi_\Phi(I_\ell)(b) = v\}}, \quad v \notin I_\ell. \tag{21}$$

Proof: Refer Appendix A. ■

1) MDP Formulation: Since we have already shown that the set of downloaded fragments evolves as a discrete-time Markov chain at the decision epochs, we can reformulate Problem 2 as an MDP. Recall that the objective function $\frac{1}{V}\sum_{\ell=0}^{V-1} \mathbb{E}\left[N(I_\ell)\right]$ is additive over the downloads. Therefore, we can consider this to be a finite MDP with V stages, where the reward in stage ℓ is

$$r_\ell(I_\ell) \triangleq \frac{N(I_\ell)}{V}, \quad \ell \in \{0,\dots,V-1\}. \tag{22}$$

Given a fixed completely utilizing α-(V,R) replication storage scheme Φ, our goal is to find the optimal work-conserving scheduling policy Ψ_Φ such that the aggregate reward is maximized over the finite time horizon, i.e.

$$\Psi^*_\Phi = \arg\max \mathbb{E}\left[\sum_{\ell=0}^{V-1} r_\ell(I_\ell)\right]. \tag{23}$$

Since the rewards are additive for our current problem, we can define a reward-to-go function $u_{\ell+1}(\Psi_\Phi(I_\ell)) = \mathbb{E}\left[\sum_{j=\ell+1}^{V-1} r_j(I_j)\right]$ at the ℓth download time instant D_ℓ, of the current state I_ℓ and the decision rule $\Psi_\Phi(I_\ell)$. We can re-write this reward-to-go function as

$$u_{\ell+1}(\Psi_\Phi(I_\ell)) = \sum_{v \notin I_\ell} p_{I_\ell, I_\ell \cup \{v\}}\Big[r_{\ell+1}(I_\ell \cup \{v\}) + u_{\ell+2}(\Psi_\Phi(I_\ell \cup \{v\}))\Big].$$

Then, the optimal reward-to-go function from $(\ell+1)$th stage is given by

$$u^*_{\ell+1}(I_\ell) = \max_{\Psi_\Phi} u_{\ell+1}(\Psi_\Phi(I_\ell)), \quad \ell \in \{0,\dots,V-1\}.$$

This is the well known *Bellman's optimality equation* and it follows from [22, Theorem 4.3.3] that

$$u^*_0 = \max_{\Psi_\Phi} \mathbb{E}\left[\sum_{\ell=0}^{V-1} r_\ell(I_\ell)\right]$$

and the optimal work-conserving scheduling policy is the one that achieves the optimal reward-to-go function i.e.

$$\Psi^*_\Phi = \arg\max_{\Psi_\Phi} u_0(\Psi_\Phi).$$

In order to solve this optimal scheduling decision we can use the standard backward induction algorithm [22]. However, the computational complexity of the backward induction algorithm to solve the optimal scheduling problem grows exponentially with the number of file fragments.

2) Greedy Scheduler: A greedy solution to MDP maximizes the immediate reward $\mathbb{E}\left[r_{\ell+1}(I_{\ell+1})\right]$ after ℓ downloads.

Theorem 4: For a given completely utilizing α-(V,R) storage scheme, the adaptive work-conserving scheduler that maximizes the expected immediate reward $\mathbb{E}\left[r_{\ell+1}(I_{\ell+1})\middle|I_\ell\right]$ after ℓ downloads is given by

$$\Psi_\Phi(I_\ell)(b) = \arg\min_{v\in S_b^\ell} \rho_\ell^g(v), \quad \ell \in \{0,\cdots,V-1\}, \tag{24}$$

where the greedy ranking function $\rho_\ell^g(v)$ for a fragment v after ℓ downloads is defined as

$$\rho_\ell^g(v) \triangleq \sum_{b\in\Phi_v} \mathbb{1}_{\{|S_b^\ell|=1\}}. \tag{25}$$

Proof: After each download ℓ, maximizing the immediate reward of the mean number $\mathbb{E}\left[N(I_{\ell+1})|I_\ell\right]$ of useful servers after next download, is equivalent to minimizing the expected additional number of servers that become useless after next download i.e., $N(I_\ell) - \mathbb{E}\left[N(I_{\ell+1})|I_\ell\right]$. Conditioned on the set of downloaded fragments I_ℓ, the number of useful servers $N(I_\ell)$ is deterministic. Therefore, we can write the conditional expectation of the reduction in the number of useful servers after $(\ell+1)$ downloads, as

$$\mathbb{E}\left[N(I_\ell) - N(I_{\ell+1})\middle|I_\ell\right] = \sum_{b\in U(I_\ell)} \mathbb{E}\left[\mathbb{1}_{\left\{S_b^\ell=\{v_{\ell+1}\}\right\}}\middle|I_\ell\right],$$

where $v_{\ell+1}$ is the $(\ell+1)$th downloaded fragment. Note that $v_{\ell+1}$ is a random variable given I_ℓ, and it takes value among all scheduled fragments with probability distribution given by Eq. (21). We can re-write the conditional expectation of the reduction in the number of useful servers given ℓ downloads, as

$$\mathbb{E}\left[N(I_\ell) - N(I_{\ell+1})\middle|I_\ell\right] = \sum_{v\notin I_\ell} p_{I_\ell, I_\ell\cup\{v\}} \sum_{a\in\Phi_v} \mathbb{1}_{\{|S_a^\ell|=1\}}. \tag{26}$$

The above sum is a convex combination of greedy rank $\rho_\ell^g(v)$ over the probability distribution of scheduled fragments v. It follows that the greedy algorithm must schedule the fragment with the lowest greedy rank, at each useful server $b \in U(I_\ell)$ after ℓ downloads. ■

Remark 10: Note that the greedy rank $\rho_\ell^g(v)$ is equal to the number of servers that become useless if fragment v gets downloaded at the $(\ell+1)$th download instant.

3) Ranked Schedulers: Recall that a scheduling algorithm has to schedule a remaining fragment to be downloaded at each of the useful servers, at each download instant. The greedy scheduler discussed previously, computes a function $\rho_\ell^g(v)$ at each download instant ℓ for each remaining fragment v and schedules the fragment with smallest rank at each useful server. Instead of $\rho_\ell^g(v)$, we could consider other functions for remaining fragments giving us a class of algorithms for various rank functions. The rank function ρ_ℓ quantifies the suitability of the fragment to be scheduled for download. Fragments with lower rank are prioritized over fragments with higher rank while they are scheduled for downloading. The complete algorithm for a choice of $\rho_\ell : [V]\setminus I_\ell \to \mathbb{R}$ is given below.

One big issue with the greedy approach is that it does not optimize the expected number of useful servers over all downloads. The greedy approach is oblivious to the evolution of I_ℓ and the choices it makes can steer the algorithm in a direction that does not minimize the mean download time. In particular, in the initial stages of download when ℓ is small, the greedy ranking function $\rho_\ell^g(v) = 0$ for many fragments and the likelihood of not making the optimal choice is high.

Algorithm 1 Suboptimal Adaptive Work-Conserving Scheduler

```
Input: α-(V,R) replication storage scheme Φ
Output: ((Ψ_Φ(I_ℓ)(b) : b ∈ U(I_ℓ) : ℓ ∈ {0,...,V−1})
1: Set $S_b^0 = S_b = \{v\in[V] : b\in\Phi_v\}$ for all $b$ in $[B]$
2: for $\ell \in \{0,\cdots,V-1\}$, do
3:     for $b$ in $U(I_\ell)$ do
4:         $\Psi_\Phi(I_\ell)(b) = \arg\min_{v\in S_b^\ell}\rho_\ell(v)$ ▷ Ties can be broken
           randomly or by any other rule.
5:     end for
6:     Update $S_b^{\ell+1} = S_b^\ell \setminus \{v_{\ell+1}\}$ for all $b$ in $U(I_{\ell+1})$
7: end for
```

A better choice for the ranking function should be more sensitive to the download sequence and be able to assign a nonzero value even in the initial stages. This implies that a good ranking function must have some desirable properties. In the following discussion we attempt to derive some of them. First, we make the simplifying assumption that the ranking function $\rho_\ell(v)$ for a remaining fragment $v \notin I_\ell$ only depends on the collection $\{S_b^\ell : b \in \Phi_v\}$.

Recall that our goal is to maximize the number of useful servers, not just at the ℓth download but over all the subsequent downloads. Intuitively, a choice of the metric $\rho(v)$ that ensures scheduling a fragment which is less likely to lead to servers storing single fragments in the future, might perform better compared to a greedy approach. To this end, we make the following two observations. The first observation is that a server becomes useless if all its stored fragments are downloaded. That is, $N(I_{\ell+1}) = \sum_{b\in[B]} \mathbb{1}_{\left\{S_b^{\ell+1}\neq\emptyset\right\}}$. The second observation is that a fragment download reduces the number of remaining fragments on each of the server it is stored. If a remaining fragment $v \notin I_\ell$ is downloaded next, then

$$\left|S_b^{\ell+1}\right| = \left|S_b^\ell\right| - \mathbb{1}_{\{b\in\Phi_v\}}, \quad b \in U(I_\ell).$$

This suggests that we want to schedule fragments that are stored on servers with large cardinality of remaining fragment set. This implies that good ranking functions should have the following monotonicity property.

Definition 6 (Monotonicity): Consider two fragments $v, w \notin I_\ell$ after ℓ downloads, and a bijection $f : \Phi_w \to \Phi_v$ such that $\left|S_b^\ell\right| \leqslant \left|S_{f(b)}^\ell\right|$ for all servers $b \in \Phi_w$. Then, we say that $v \geqslant_\ell w$. A ranking function ρ_ℓ is said to be monotonic if $\rho_\ell(v) \leqslant \rho_\ell(w)$ for all $v \geqslant_\ell w$.

Remark 11: We note that greedy ranking function ρ_ℓ^g has the monotonic property, however, it maps to zero for many fragments for small ℓ.

Unfortunately, there are many collections of remaining fragment sets which can not be compared. We still need to rank such fragments, and therefore we propose the following

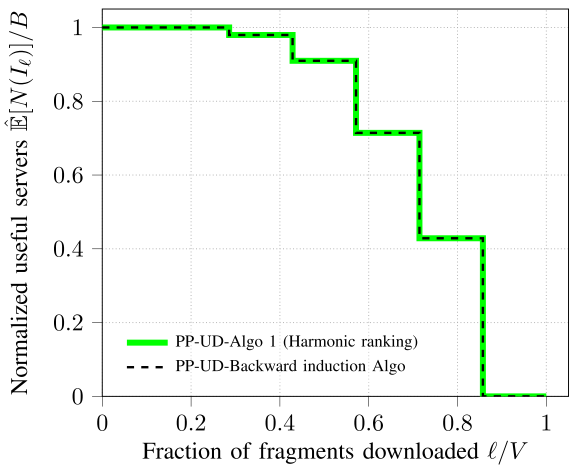

Fig. 2. This plot shows the comparison of empirical average of the normalized number of useful servers $\mathbb{E}[\hat{N}(I_\ell)]/B$ between the optimal scheduling using the backward induction algorithm and a ranked scheduler given in Algorithm 1 with the harmonic ranking function. The storage scheme is the projective plane (PP) based $\frac{3}{7}$-$(7,3)$ replication storage, and the initial arrangement follows uniform diversity (UD) at each layer as in Table VII.

harmonic ranking function $\rho_\ell^h : I_\ell^c \to \mathbb{R}$ at the ℓth download instant, for each of the remaining fragments $v \notin I_\ell$, as

$$\rho_\ell^h(v) \triangleq \sum_{a\in\Phi_v} \frac{1}{|S_a^\ell|}, \quad v \notin I_\ell, \quad \ell \in \{0,\dots,V-1\}. \tag{27}$$

Remark 12: Note that the ranking function ρ_ℓ^h has the monotonic property, and the value $\rho_\ell^h(v)$ for a fragment v is the harmonic sum of the cardinality of the set of remaining fragments $\left|S_a^\ell\right|$ at each server $a \in \Phi_v$ after ℓ downloads.

We observe that, as the reciprocals of numbers increases steeply as the numbers get smaller, this harmonic sum is highly sensitive to low size of remaining fragment set at servers. That is, the rank of a fragment increases significantly if the servers on which it is hosted has very few remaining fragments to be downloaded. Then, scheduling the fragment with the least value of the ranking function helps in decreasing the download probability of fragments which are stored on servers with low number of remaining fragments. This in turn reduces the probability of reduction of number of useful servers at each download instant.

Algorithm 1 is computationally efficient, easy to implement, and requires us to only keep track of the downloaded fragments for a given completely utilizing α-(V, R) storage scheme. We know that ideally, the best algorithm should provide a solution that is jointly optimal over all download instants. However, this algorithm only provides the best solution for the current download instant and does not take into account the impact of the current scheduling decision on the future evolution of the system. Yet, we observe that its performance is comparable with the optimal backward induction algorithm when implemented for a small number of fragments as seen from Fig. 2.

We conclude this section with a brief discussion on the impact of scheduling and decoding on latency. For replication codes, the decoding cost is negligible since the fragments are uncoded. However, there is a scheduling cost associated with adaptive scheduling. We measure the increase in latency in terms of the computational cost of adaptive scheduling at each server, after each fragment download. We estimate this computation cost[2] for Algorithm 1. Assuming that the computation of metric $\rho_\ell(v)$ requires at most c computations, we need to compute the rank $\rho_\ell(v)$ for $|S_b^\ell| \leqslant K$ fragments at each server. Finding the minimum of these ranks takes $O(|S_b^\ell|)$ time. Therefore, there is an additional delay of $O(cK)$ at each server, due to scheduling computations. Since there are V fragments, the total delay is $O(cVK)$ over all the downloads. For harmonic rank scheduler the cost of computing the rank $\rho_\ell(v)$ is $O(R)$. Thus, the adaptive ranked scheduling for replication coded storage increases the latency by $O(VRK)$.

In case of MDS codes, the scheduling cost is negligible since any fragment can be scheduled and this could be decided before the current download is complete. However, for MDS codes there is an additional cost due to decoding. For Reed-Solomon codes, a popular class of MDS codes, practical erasure decoding algorithms have complexity $O(V^2)$, see for instance [14]. Thus the increase in latency due to decoding complexity of MDS codes is $O(V^2)$.

VII. Randomized Storage Ensemble and Scheduling

In the preceding sections we had considered deterministic strategies for placement and scheduling. It is natural to consider randomized approaches for these tasks. In this section we take up this point of view. Our goal is to demonstrate the competitiveness of replication coded placement schemes vis-a-vis the MDS coded placement schemes. To achieve this, we propose a random storage coding model and a random scheduling policy and analyze their joint performance with respect to the number of useful servers. To analyze the performance, we need to decouple the placement with the scheduling and the sequence of downloaded fragments. To this end we modify the system model. The modified system approaches the system we had considered so far in the limit of large number of fragments giving us useful results for the system of interest. More precisely, we consider a hypothetical system of B servers, each having a storage capacity of VR fragments and random exponential service rates that can be altered at every download stage. We propose a random storage scheme and a random scheduling policy for a (VR, V) replication code stored over this heterogenous B servers system. For the proposed random storage scheme and randomized scheduling, we show the following two results when the number of fragments grow large. We first show that for any fixed fraction $1 - \beta$ of fragment downloads, the download rate at each server converges to a constant rate μ almost surely. We next show that the fraction of fragments stored per server converges to $\alpha \triangleq \frac{R}{B}$ almost surely. On average, a random storage scheme chosen from this ensemble together with a randomized work-conserving scheduling policy is arbitrarily close to the solution of Problem 1, asymptotically as the number of fragments grows large.

[2]Note that complexity of Algorithm 1 is different from the cost on the latency since the scheduling at each server is in parallel.

A. Randomized Replication Coded Storage Over B Servers

A storage scheme for a (VR, V) replication code, stored on a system with B servers each having storage capacity of VR fragments, is called a (B, V, R) *replication storage scheme*.

Definition 7: A (B, V, R) replication storage scheme, where the rth replica of fragment v is stored on server $\Theta_{v,r} \in [B]$, chosen independently and uniformly at random from B servers, is called a *randomized* (B, V, R) *replication storage scheme*. The collection of all possible realizations of the random (B, V, R) replication storage scheme is referred to as the *random* (B, V, R) *replication storage ensemble.*

Remark 13: A randomized (B, V, R) replication storage scheme can be determined by *i.i.d.* random vectors $\Theta_v = (\Theta_{v,r} : r \in [R]) \in [B]^R$ for all fragments $v \in [V]$, where

$$P\{\Theta_{v,r} = b\} = \frac{1}{B}, \quad \text{for all } b \in [B].$$

We observe that the number of replicated fragments VR is smaller than the total system storage capacity BVR for $B > 1$, and hence this is an underutilizing storage scheme.

Note that, the random vector Θ_v is not a set but a vector. In particular, more than one replica of a file fragment can be stored on a single server. For each fragment v, we can compute the number of replicas of this fragment stored at server b as

$$N_{vb} \triangleq \sum_{r\in[R]} \mathbb{1}_{\{\Theta_{v,r}=b\}}. \tag{28}$$

We note that for a fixed server b, $(N_{vb} : v \in [V])$ are *i.i.d.* random variables with mean $\alpha \triangleq R/B$.

Lemma 4: For the random (B, V, R) replication storage scheme defined in Definition 7, we have

$$P(\cup_{b\in[B]} \{N_{vb} \geqslant 2\}) \geqslant 1 - e^{\frac{-\alpha(R-1)}{2}}, \quad v \in [V].$$

Proof: The event of no server storing more than a single replica of a fragment v in the random replication storage scheme is given by

$$E \triangleq \{\Theta_{v,1} \neq \Theta_{v,2} \neq \cdots \neq \Theta_{v,R}\}.$$

Since placement of each replica $r \in [R]$ for each fragment $v \in [V]$ is *i.i.d.* uniform over servers in $[B]$, the probability of this event is

$$P(E) = \prod_{r=0}^{R-1}\left(1 - \frac{r}{B}\right) \leqslant e^{-\sum_{r=0}^{R-1}\frac{r}{B}} = e^{-\frac{\alpha(R-1)}{2}}.$$

The result follows as the event $\cup_{b\in[B]}\{N_{vb} \geqslant 2\}$ is the complement of the event E. ■

As the choice of fragment v in the above Lemma was arbitrary, it implies that the probability of each file fragment repeating on some server is non-zero. However, we can construct an occupancy set Φ_v for each fragment v from this vector Θ_v, by throwing away the repeated entries. That is,

$$\Phi_v = \{b \in [B] : \Theta_{v,r} = b \text{ for some } r \in [R]\}.$$

This implies that $|\Phi_v| \leqslant R$, and this inequality is strict if any entry in the vector Θ_v is repeated twice.

We will consider a family of (B, V, R) random replication storage schemes for increasing values of number of fragments V, while keeping the ratio $\alpha = \frac{R}{B}$ constant for the system. In this case, we will show that the fraction of file fragments stored by each server converges to α for the proposed random (B, V, R) replication storage scheme. Recall that the ratio α is the normalized storage capacity of each server in a completely utilizing α-(V, R) storage scheme. The normalized number of fragments stored at any server $b \in [B]$ is defined as

$$\alpha_b^{\text{rep}} \triangleq \frac{1}{V}\sum_{v\in[V]}\sum_{r\in[R]} \mathbb{1}_{\{\Theta_{v,r}=b\}}. \tag{29}$$

Remark 14: If the normalized number of fragments $\alpha_b^{\text{rep}} = \alpha$, then the randomized (B, V, R) replication storage scheme defined in Definition 7 in terms of *i.i.d.* vectors $(\Theta_v : v \in [V])$[3], is an underutilizing α-(V, R) storage scheme with high probability, since there exists servers storing redundant replicas of the same fragment with high probability.

Definition 8: We say that a randomized (B, V, R) replication storage scheme is an α-(V, R) storage scheme asymptotically in V, if for each server b, $\lim_{V\to\infty} \alpha_b^{\text{rep}} = \alpha$ almost surely.

Theorem 5: The randomized (B, V, R) storage scheme defined in Definition 7 is an α-(V, R) storage scheme asymptotically in V.

Proof: From Eq. (28), the number of replicas of a fragment v stored at server, $N_{vb} = \sum_{r\in[R]} \mathbb{1}_{\{\Theta_{v,r}=b\}}$. Note that $(N_{vb} : v \in [V])$ is a sequence of *i.i.d.* random variables for a fixed b and has mean $\alpha = \frac{R}{B}$. Since N_{vb} takes only non-negative values, we get $\mathbb{E}|N_{vb}| = \mathbb{E}[N_{vb}] = \alpha < \infty$. It follows from the L^1 strong law of large numbers [51, Theorem 2.4.1], that $\alpha_b^{\text{rep}} = \frac{1}{V}\sum_{v=1}^{V} N_{vb} \xrightarrow[V\to\infty]{\text{a.s}} \alpha$. ■

B. Randomized Scheduling Over Heterogenous B Servers

The number of fragments on each server $b \in [B]$ after downloading set of fragments I_ℓ is given by $N_b^\ell \triangleq \sum_{v\notin I_\ell} N_{vb}$. We consider the system of B servers with independent random download times distributed exponentially, where the download rate of a server $b \in [B]$ after ℓth download is set to

$$\mu_b^\ell \triangleq \frac{N_b^\ell}{\alpha(V-\ell)}\mu. \tag{30}$$

We next introduce the randomized scheduling for the random (B, V, R) replication storage scheme.

Definition 9: We define a *random scheduler* that schedules a fragment replica stored on a server, independently and uniformly at random. That is, after ℓth download, the probability of scheduling a fragment $v \notin I_\ell$ to be downloaded from server $b \in U(I_\ell)$ is given by

$$P(\Psi(I_\ell)(b) = v \Big| I_\ell, \Theta) = \frac{N_{vb}}{N_b^\ell}. \tag{31}$$

[3]Even though $\Theta : [V] \to [B]^R$ is a collection of *i.i.d.* random variables, we observe that normalized number of fragments on each server α_b^{rep} are dependent random variables. To see this, we observe that $P\{\Theta_{v,r} = b, \Theta_{v,r} = a\} = 0 \neq \frac{1}{B^2}$ for any $b \neq a$. It follows that

$$\mathbb{E}\left[\alpha_b^{\text{rep}}\alpha_a^{\text{rep}}\right] = \alpha^2 - \frac{1}{B}\alpha \neq \alpha^2 = \mathbb{E}\left[\alpha_b^{\text{rep}}\right]\mathbb{E}\left[\alpha_a^{\text{rep}}\right].$$

Lemma 5: Consider the heterogeneous B server system with download rates defined in Eq. (30), under the random (B, V, R) replication storage defined in Definition 7 and random scheduling defined in Definition 9. For any $\ell \in \{0, \dots, V-1\}$, the fragment download set I_ℓ is independent of the storage vector Θ.

Proof: It suffices to show that $P(I_{\ell+1} = I_\ell \cup \{v\} \Big| I_\ell, \Theta) = \frac{1}{V-\ell}$ for all $\ell \in \{0, \dots, V-1\}$. To this end, we fix the storage vector Θ and the downloaded fragments I_ℓ after ℓ downloads. We observe that the exponential download rate at each useful server $b \in U(I_\ell)$ is proportional to the number of remaining useful fragment replicas N_b^ℓ. Therefore, the probability of download completion from a useful server $b \in U(I_\ell)$ is given by $\frac{N_b^\ell}{\sum_{a \in U(I_\ell)} N_a^\ell}$. Together with Eq. (31) for random selection probability of a fragment and the fact that $\sum_{b \in U(I_\ell)} N_{vb} = R$, we can compute the probability of $(\ell+1)$th download being a remaining useful fragment $v \notin I_\ell$, as

$$\sum_{b \in U(I_\ell)} \frac{N_b^\ell}{(V-\ell)R} P(\Psi(I_\ell)(b) = v \Big| I_\ell, \Theta) = \frac{1}{(V-\ell)}.$$

■

Definition 10: Consider the heterogeneous B server system with download rates μ_b^ℓ for server b and download ℓ. For any $\beta \in (0,1)$ and $\ell \leqslant V(1-\beta)$, we say that this is a homogeneous B server system with download rate μ asymptotically in V, if $\lim_{V\to\infty} \mu_b^\ell = \mu$ almost surely for each server b.

Lemma 6: Consider the heterogeneous B server system with download rates μ_b^ℓ defined in Eq. (30), under the random (B, V, R) replication storage defined in Definition 7 and random scheduling defined in Definition 9. For any $\beta \in (0,1)$ and $\ell \leqslant V(1-\beta)$, this system converges to a homogeneous B server system with download rate μ asymptotically in V.

Proof: Consider the system at the ℓth stage of download where $\ell \leqslant V(1-\beta)$. We observe that $1 = \sum_{S \subseteq [V]:|S|=\ell} \mathbb{1}_{\{I_\ell = S\}}$, and hence we can write

$$\frac{\sum_{v \notin I_\ell} N_{vb}}{V-\ell} = \sum_{S \subseteq [V]:|S|=\ell} \mathbb{1}_{\{I_\ell = S\}} \frac{\sum_{v \notin S} N_{vb}}{|S^c|}.$$

From Lemma 5, the downloaded fragment set I_ℓ and storage vector Θ are independent. It follows that for a fixed $\ell \in \{0, \dots, V-1\}$, realization $\{I_\ell = S\}$, and server $b \in U(S)$, the random sequence $(N_{vb} : v \notin S)$ is *i.i.d.* and has mean α. Since N_{vb} takes only non-negative values, we get $\mathbb{E}|N_{vb}| = \mathbb{E}[N_{vb}] = \alpha < \infty$. Thus, the result follows from the L^1 strong law of large numbers [51, Theorem 2.4.1] which implies that $\mathbb{1}_{\{I_\ell = S\}} \frac{\sum_{v \notin S} N_{vb}}{|S^c|} \xrightarrow[V\to\infty]{\text{a.s}} \alpha$. ■

We can compute the sum of mean number of useful servers aggregated over first $\ell \leqslant V(1-\beta)$ downloads.[4]

Theorem 6: Consider the heterogeneous B server system with download rates μ_b^ℓ defined in Eq. (30), under the random (B, V, R) replication storage defined in Definition 7 and random scheduling defined in Definition 9. For any $\beta \in (0,1)$ and $\ell_\beta \triangleq \lfloor V(1-\beta) \rfloor$, we can write

$$\frac{1}{BV} \sum_{\ell=0}^{\ell_\beta} \mathbb{E}[N(I_\ell)] = \frac{\ell_\beta + 1}{V} + \frac{\left(1 - \frac{1}{B}\right)^{R(V-\ell_\beta)} \left(1 - (1-\frac{1}{B})^{R(\ell_\beta+1)}\right)}{V\left(1 - (1-\frac{1}{B})^R\right)}.$$

Proof: The expectation $\mathbb{E}[N(I_\ell)]$ is taken over the randomness in the download sequence $(I_0, \dots, I_{V-1})$, resulting from the randomness in the service times, scheduling, and the storage vector $\Theta = (\Theta_v : v \in [V])$ of the random (B, V, R) replication storage ensemble. For a random storage vector Θ, we will obtain one of the random storage schemes $\Phi = (\Phi_v : v \in [V])$ defined as $\Phi_v \triangleq \cup_{r \in [R]} \{\Theta_{vr}\}$. For a download sequence I_ℓ, we can write the set of useful servers as $U(I_\ell) = \cup_{v \notin I_\ell} \Phi_v = \cup_{v \notin I_\ell} \cup_{r \in [R]} \{\Theta_{vr}\}$. This implies that a server $b \notin U(I_\ell)$ if and only if $\Theta_{vr} \neq b$ for any $v \notin I_\ell$ and $r \in [R]$. As the total number of servers is B, we can deduce that the number of useful servers is given by $N(I_\ell) \triangleq |U(I_\ell)| = B - \sum_{b \in [B]} \prod_{v \notin I_\ell} \prod_{r \in [R]} \mathbb{1}_{\{\Theta_{vr} \neq b\}}$. From Lemma 5, the storage vector Θ and download sequence I_ℓ are independent, and hence

$$\mathbb{E}\left[\prod_{v \notin I_\ell} \prod_{r \in [R]} \mathbb{1}_{\{\Theta_{vr} \neq b\}} \sum_{S \subseteq [V]:|S|=\ell} \mathbb{1}_{\{I_\ell = S\}}\right] = \sum_{S \subseteq [V]:|S|=\ell} \mathbb{E}\left[\mathbb{1}_{\{I_\ell = S\}}\right] \mathbb{E}\left[\prod_{v \notin S} \prod_{r \in [R]} \mathbb{1}_{\{\Theta_{vr} \neq b\}}\right].$$

From the randomized construction of the storage scheme, the random storage vector Θ is *i.i.d.* and $P\{\Theta_{vr} \neq b\} = (1 - 1/B)$. From the linearity of expectation, it follows that

$$\frac{1}{BV} \mathbb{E}[N(I_\ell)] = \frac{1}{V}\left(1 - \left(1 - \frac{1}{B}\right)^{R(V-\ell)}\right). \quad (32)$$

Result follows from summing up the above equation on both sides over $\ell \in \{0, \cdots, \lfloor V(1-\beta) \rfloor\}$. ■

Corollary 1: For asymptotically large number of fragments, a random (B, V, R) replication storage scheme defined in Definition 7 together with the random scheduler defined in Definition 9, is arbitrarily close to the solution to the Problem 1 almost surely.

Proof: As the number of fragments V tends to infinity, the following three results hold.

1) From Theorem 5, the proposed random (B, V, R) replication storage scheme almost surely converges to an α-(V, R) replication storage scheme. The resulting scheme is underutilizing with high probability.
2) From Lemma 6, the download rates μ_b^ℓ of the servers $b \in U(I_\ell)$ in the heterogenous B server system converges to the constant service rate μ of α-B server systems, for downloads $\ell \leqslant V(1-\beta)$ and $\beta \in (0,1)$ can be made arbitrarily small.
3) From Theorem 6, we obtain $\lim_{V\to\infty} \frac{1}{BV} \sum_{\ell=0}^{V-1} \mathbb{E}[N(I_\ell)] \geqslant 1 - \beta$ for any $\beta \in (0,1)$.

[4]Theorem 6 appeared originally as Theorem 9 in [1] which had a gap in the proof. This is corrected in the current manuscript.

From Eq. (11) and Remark 5, we observe that the ensemble mean of normalized number of useful servers for the proposed random (B, V, R) replication storage scheme under the random scheduling, is arbitrarily close to the upper bound for any α-(V, R) replication storage scheme, asymptotically in V. ■

VIII. Comparison With MDS Codes

So far we have looked at storage schemes based on (VR, V) replication codes. In this section we consider storage schemes based on (VR, V) MDS codes assuming that the field is large enough so that a (VR, V) MDS code exists. MDS codes are known to outperform replication codes in many settings. For example, MDS codes have better code rates for same fraction of erasure correction [52], and are shown to be latency optimal for class of symmetric codes in single fragment storage [10].

In this section, we first show that among all α-(V, R) coded storage schemes, the ones based on MDS codes minimize the mean download time. Second, we find the bounds on the number of useful servers for MDS coded storage, and show that replication coded storage is asymptotically order optimal. That is, when the number of fragments grows large, the average number of useful servers per fragment can be achieved by random replication. Third, we show that when each server can store the whole file, i.e. $K \geqslant V$, then the replication coded storage is as good as a MDS coded storage, even in the non-asymptotic regime.

Definition 11: Consider a file with V fragments encoded to VR coded fragments, and completely utilizing storage of this (VR, V) code on an α-B system. Such storage schemes are referred to as α-(V, R) *coded storage schemes*, where the normalized storage capacity per server is $\alpha = \frac{R}{B}$ and the code rate is $\frac{1}{R}$.

Remark 15: For any α-(V, R) coded storage scheme, the number of useful servers $N(I_\ell)$ after ℓ downloads is always upper bounded by the total number of servers B, and hence $\frac{1}{BV}\sum_{\ell=0}^{V-1} N(I_\ell) \leqslant 1$.

Definition 12: Any V subset of VR coded fragments that suffices to decode a (VR, V) code, i.e., reconstruct the V uncoded fragments, is called an *information set* [10], [53]. For an α-(V, R) coded storage scheme, we can define the collection of all information sets [10, Section II], as $\mathcal{I}$.

For a completely utilizing α-(V, R) replication storage scheme, information sets consist of distinct V fragments. For a completely utilizing α-(V, R) MDS coded storage, information sets are any V coded fragments, and hence $\mathcal{I} = \{S \subset [VR] : |S| = V\}$. This implies that the collection of information sets for MDS code includes collection of information sets for any other (VR, V) code.

Theorem 7: Among all α-(V, R) coded storage schemes, MDS codes minimize the mean download time.

Proof: For any completely utilizing α-(V, R) coded storage scheme, each server b is storing a set $S_b \subseteq [VR]$ of $|S_b| = \alpha V$ out of VR coded fragments. Further, the first ℓ downloaded symbols I_ℓ are an ℓ-subset of some information set $S \in \mathcal{I}$. Then, we can write the set of useful servers after ℓ downloads as those that have the remaining coded symbols for such information sets. That is,

$$U(I_\ell) = \cup_{S \in \mathcal{I}} \{b \in [B] : (S \setminus I_\ell) \cap S_b \neq \emptyset\}.$$

Recall that the collection of information sets for MDS codes includes collection of information sets for any other (VR, V) code. Using this fact together with the definition of the set of useful servers, it follows that the largest possible set of useful servers among all α-(V, R) coded storage schemes, is the one achieved by MDS coded storage for the same download sequence I_ℓ. From coupling arguments for the download sequence and Eq. (7) for mean download time, the result follows by induction on the number of downloaded fragments. ■

A. Asymptotic Order Optimality of Replication Codes

We established that the among all α-(V, R) coded storage schemes, an MDS code has the largest number of useful servers. We next find bounds on the number of useful servers for MDS coded storage, which can be used as a benchmark to compare replication coded storage.

Lemma 7: For a completely utilizing α-(V, R) MDS storage scheme, the number of useful servers $N^{\text{mds}}(I_\ell)$ is bounded as

$$B - \left\lfloor \frac{\ell}{K} \right\rfloor \leqslant N^{\text{mds}}(I_\ell) \leqslant \min\{B, VR - \ell\}.$$

Proof: For an α-(V, R) MDS coded storage scheme, each VR coded fragment is useful and downloading any V coded fragments suffices to reconstruct the entire file. Therefore, if $\ell < V$ fragments are downloaded, then all the servers that store any of the remaining $VR - \ell$ fragments are useful. As each server can store $K = \alpha V$ fragments each, a server can become useless if and only if all its K coded fragments have been downloaded. Therefore, the maximum number of servers that can be useless after ℓ downloads is $\lfloor \ell/K \rfloor$.

To obtain the upper bound on the number of useful servers, we make the following observations. First, that all servers remain useful at the ℓth download if less then K coded fragments have been downloaded from them. However, if the number of remaining coded fragments $VR - \ell < B$, then at most $VR - \ell$ unique servers are useful. ■

The previous lemma will immediately give us the following result by taking average over all V fragments.

Corollary 2: For a completely utilizing α-(V, R) MDS coded storage scheme with code rate $\frac{1}{R} \leqslant \frac{V}{B+V}$, the normalized aggregate number of useful servers is bounded as

$$1 - \frac{1}{2R}\left(1 - \frac{1}{V}\right) \leqslant \frac{1}{BV}\sum_{\ell=0}^{V-1} N^{\text{mds}}(I_\ell) \leqslant 1. \tag{33}$$

Remark 16: From Corollary 1 we observe that the limit of average number of useful servers for the random (B, V, R) replication storage scheme together with the random scheduling can be brought arbitrarily close to the upper bound for MDS codes in Eq. (33), as the number of fragment grows. This implies that replication coded storage is asymptotically order optimal.

TABLE VI

A NONADAPTIVE SCHEDULING POLICY FOR THE $\frac{3}{7}$-$(7,3)$ REPLICATION STORAGE CODING SCHEME WITH THE FRAGMENTS SCHEDULED IN INCREASING ORDER OF THEIR INDICES

Smallest index first scheduling							
Server	1	2	3	4	5	6	7
Layer 1	1	3	1	1	2	3	2
Layer 2	2	4	5	4	5	6	4
Layer 3	3	5	6	7	7	7	6

B. Optimality of Replication Codes for Large Storage

So far, we have considered the case $\alpha \leqslant 1$. In other words, it means that storage capacity per server is $K = \alpha V \leqslant V$, i.e. each server can store at most a single file. We now show that, when $K \geqslant V$, then there exists a (VR, V) replication code such that the number of useful servers remains B after every download, for any work-conserving scheduling policy. Thus, the average number of useful servers for replication code meets the universal upper bound for all fragment size V, when the storage can store at least the entire file.

Remark 17: Suppose that the servers are not memory constrained i.e. $\alpha \geqslant 1$, then we can store the entire file on each of the servers. To see this observe that we have $R = BK/V = B\alpha \geqslant B$. Therefore, we can place all the distinct V fragments on every server. Thus every server is able to provide a useful fragment until the entire file is downloaded. The number of useful servers is always B, which is the best case.

IX. NUMERICAL STUDIES

In this section, we present the results of our numerical studies for completely utilizing α-(V, R) replication storage schemes and work-conserving scheduling policies. The interplay of the storage scheme and work-conserving scheduling policies determines the overall download time for a file. We use the storage schemes proposed in Section V. We study the performance of these storage codes in conjunction with various nonadaptive and adaptive scheduling policies.

Before presenting the numerical results, we first review some scheduling policies and illustrate them by considering a completely utilizing $\frac{3}{7}$-$(7,3)$ replication storage scheme. We consider the storage scheme constructed from a projective plane.

Smallest Index First Scheduling: A straightforward nonadaptive scheduling policy is to schedule the fragments based on their indices. We could arrange the fragments in the increasing order of the fragment index. After a fragment is downloaded, the fragment with the next highest index is moved to the head of the server. This scheduling policy applied to the storage scheme in Table IV is shown in Table VI.

Uniform Diversity Scheduling: The previous scheduling policy leads to an asymmetric scheduling in that all fragments are not equally distributed at the heads of the servers. For instance, in Table VI, fragments 4 to 7 are not scheduled in the first layer. We can make the policy more symmetric, by scheduling as many distinct fragments at the head of each server and in each subsequent layer. The motivation being that having a diversity of fragments at the heads of the servers leads to larger number of fragments being downloaded in parallel. One such scheduling for the design based $\frac{3}{7}$-$(7,3)$ replication storage is shown in Table VII. For storage schemes based on projective planes and the cyclic shift storage scheme, where $B = V$, it is always possible to place fragments in each layer across the servers as a permutation of all the fragments. Such a uniform scheduling may not be possible for every storage scheme.

TABLE VII

A SCHEDULING FOR $\frac{3}{7}$-$(7,3)$ REPLICATION STORAGE CODING SCHEME WITH UNIFORM DIVERSITY AT EACH LAYER

Uniform diversity scheduling							
Server	1	2	3	4	5	6	7
Layer 1	1	3	5	7	2	6	4
Layer 2	3	4	6	1	5	7	2
Layer 3	2	5	1	4	7	3	6

TABLE VIII

COMBINING PUSHBACK (PB) POLICY WITH THE UNIFORM DIVERSITY (UD) POLICY FOR THE $\frac{3}{7}$-$(7,3)$ REPLICATION STORAGE CODING SCHEME. THE FRAGMENTS OF THE FIRST SERVER (IN BLUE) ARE PLACED IN THE LAST LAYER, DUE TO WHICH THEY ARE SCHEDULED LAST IN THOSE SERVERS

PB with UD scheduling							
Server	1	2	3	4	5	6	7
Layer 1	1	4	5	7	5	6	4
Layer 2	3	5	6	4	7	7	6
Layer 3	2	3	1	1	2	3	2

Pushback scheduling: This scheduling policy aims to maximize the number of useful servers toward the end when a large number of fragments have been downloaded. A heuristic scheduling policy that aims to maximize $N(I_\ell)$ in this range is as follows: Pick a server b and schedule the fragments of S_b last in the other servers i.e. $[B] \setminus \{b\}$. In a projective plane based storage scheme, the pushback policy will schedule the fragments stored in bth server on $R(R-1)$ disjoint servers. Comparing with the bound given in Theorem 3 for the number of useful servers, we can see this will lead to larger number of useful servers at the end.

The pushback policy can be combined with any other scheduling policy. To combine the pushback policy with another scheduling policy, we first apply that policy and then modify this scheduling according to the pushback scheduling. So when this is applied in conjunction with the uniform diversity scheduling, we take the scheduling for uniform diversity as in Table VII and the modify it according to the pushback policy. If we choose the first server fragments to be placed last then the resulting scheduling is given in Table VIII. Similarly, when we combine pushback with smallest index first policy we obtain the scheduling in Table VI.

Adaptive scheduling based on harmonic rank: We now illustrate an example of an adaptive ranked scheduler with harmonic ranking function defined in Eq. (27), with design based $\frac{3}{7}$-$(7,3)$ replication storage code given in Table VII. We will look at one sample path of download sequence in Fig. 3.

TABLE IX

COMBINING PUSHBACK (PB) POLICY WITH THE SMALLEST INDEX FIRST POLICY IN TABLE VI FOR THE $\frac{3}{7}$-$(7, 3)$ REPLICATION STORAGE SCHEME

PB with smallest index first scheduling							
Server	1	2	3	4	5	6	7
Layer 1	1	4	5	4	5	6	4
Layer 2	2	5	6	7	7	7	6
Layer 3	3	3	1	1	2	3	2

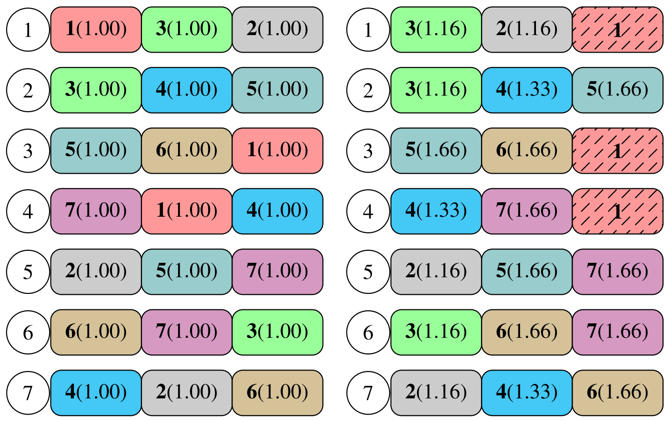

Fig. 3. We show first two steps of a sample path of download sequence for a design based $\frac{3}{7}$-$(7, 3)$ replication storage scheme given in Table VII. The ranked adaptive scheduling defined in Algorithm 1 with harmonic rank function defined in Eq. (27) is used. For each remaining fragment, we have listed its identity and rank.

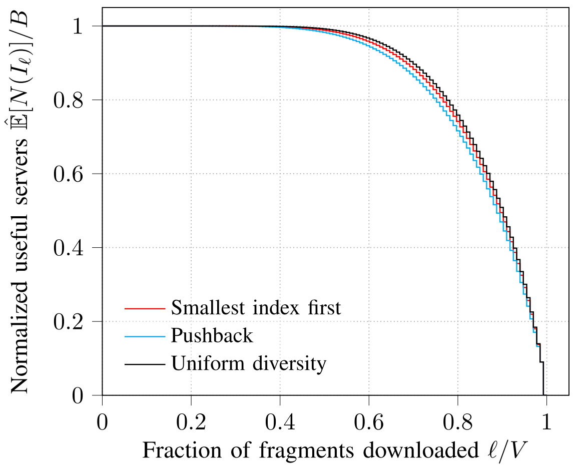

Fig. 4. This plot shows an empirical average of the normalized number of useful servers $\hat{\mathbb{E}}[N(I_\ell)]/B$ for the cyclic shift based $\frac{12}{133}$-$(133, 12)$ replication storage scheme with nonadaptive scheduling policies.

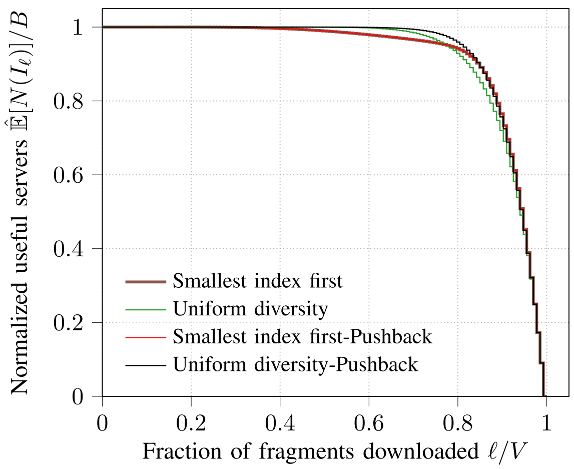

Fig. 5. This plot shows an empirical average of the normalized number of useful servers $\hat{\mathbb{E}}[N(I_\ell)]/B$ for the projective plane based $\frac{12}{133}$-$(133, 12)$ replication storage scheme with nonadaptive scheduling policies.

Initially harmonic rank of all fragments is identically unity, and any stored fragment can be scheduled at any of the 7 servers. Let us assume that the first downloaded fragment is 1. We now compute the harmonic rank of the remaining six fragments $\{2, 3, 4, 5, 6, 7\}$ as $\{1.16, 1.16, 1.33, 1.66, 1.66, 1.66\}$ respectively. Thus, Algorithm 1 schedules either fragment 2 or 3 on server 1, fragment 3 on server 2, either fragment 5 or 6 on server 3, fragment 4 on server 4, fragment 2 on server 5, fragment 3 on server 6, and fragment 2 on server 7.

We now present the results of our numerical studies on a completely utilizing α-(V, R) replication storage scheme constructed from a projective plane of order $q = 11$. This results in a symmetric (VR, V) replication code stored on an α-B system, where the number of servers $B = V = q^2+q+1$, the replication factor of each fragment $R = q + 1$, and the storage capacity of each server is a fraction $\alpha = R/B$ of all V fragments. We also considered an alternative completely utilizing α-(V, R) replication storage scheme based on cyclic shift of fragments, for the identical parameters α, V, R. As mentioned earlier, the download time for each fragment is modelled as an independent random variable that has an exponential distribution with rate $\mu = 10^{-5}$, chosen to amplify the differences between various storage schemes and scheduling policies. We performed Monte Carlo simulations of our system setup with 1×10^5 runs. We computed the normalized empirical mean $\hat{\mathbb{E}}[N(I_\ell)]/B$, of number of useful servers $N(I_\ell)$ after ℓ downloads, averaged over all simulation runs. Performance of various storage schemes and scheduling policies are compared by plotting the normalized empirical average $\hat{\mathbb{E}}[N(I_\ell)]/B$ as the fraction of downloads ℓ/V grows. From Eq. (7), we know that the performance of the scheme is better if the number of useful servers remains high as the download progresses. That is, a uniformly higher plot is indicative of a better performance.

The simulation results for the cyclic shift based storage scheme for nonadaptive scheduling policies, are shown in Fig. 4. We observe that the uniform diversity scheduling has the best performance among the proposed nonadaptive scheduling policies. The results for the projective plane based scheme for nonadaptive scheduling policies, are shown in Fig. 5. In this case, we observe that the scheduling policy that combines the uniform diversity with pushback has the best performance.

For the following numerical studies, we consider ranked schedulers with greedy and harmonic ranking functions, as adaptive scheduling policies. In Fig. 6 these adaptive scheduling policies are compared with the best nonadaptive policies

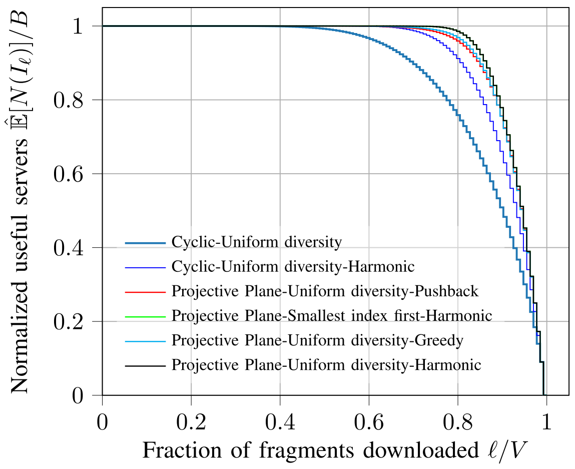

Fig. 6. This plot shows an empirical average of the normalized number of useful servers $\hat{\mathbb{E}}[N(I_\ell)]/B$ for $\frac{12}{133}$-$(133, 12)$ replication storage schemes based on projective plane and cyclic shift, with best nonadaptive scheduling and rank based adaptive scheduling policies.

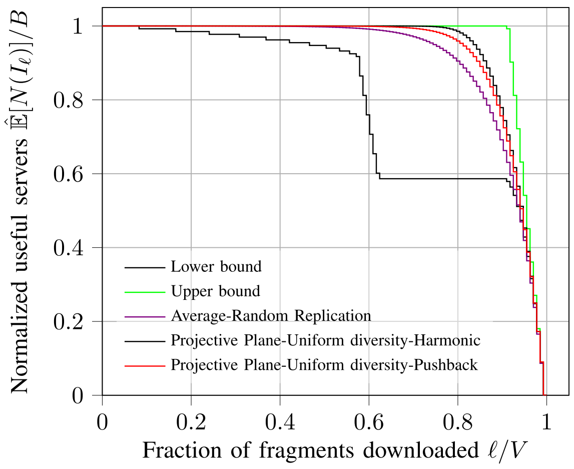

Fig. 7. Comparison of the best nonadaptive and adaptive scheduling policies for the projective plane based $\frac{12}{133}$-$(133, 12)$ replication storage scheme with the normalized universal lower bound obtained as given in Remark 9, the normalized upper bound in Eq. (10), and the average performance of $(133, 133, 12)$ random replication (Eq. (32)).

for both cyclic shift based and projective plane based storage schemes. In this case, we have the freedom to choose the initial schedule, since the rank of all fragments remains identical before the first download. We explored two types of initial schedules: i) the smallest index first, and ii) uniform diversity where all fragments are present. The performance is very similar in both cases as can be seen in Fig. 6, with the uniform initialization performing slightly better with respect to the total download time. This suggests that for a given storage scheme, the initialization does not affect the overall performance as much. Note also, that the cyclic storage scheme with adaptive scheduling still does not perform as well as nonadaptive scheduling with design based storage scheme. Therefore, it is important to find a good storage scheme. As mentioned in Section V-C, the performance of cyclic shift based replication storage with nonadaptive scheduling is poor due to the large value of overlap parameters. However, this storage scheme has low overlap for certain fragment and occupancy sets. An adaptive scheduling exploits this property by driving the system state to ensure the good fragment sets such that the remaining fragments have low overlap in their occupancy sets.

In Fig. 7, we compare the performance of projective plane based storage scheme for the best performing nonadaptive and adaptive scheduling policies against the bounds, which includes the universal lower bounds on $N(I_\ell)$ derived in Theorems 2 and 3, and the lower bound on the $N(I_\ell)$ obtained from Remark 9 for projective plane based storage scheme, the upper bound for $N(I_\ell)$ random replication storage given in Eq. (10), the average number of useful servers given in Eq. (32) for heterogeneous server system under random replication storage and random scheduling. One important observation is that the performance of the deterministic storage scheme is superior to the average performance of the random code ensemble. Therefore, it is worthwhile to develop good deterministic storage schemes.

Finally, in Fig. 8, we show the variation of the normalized number of useful servers with increase in the number of

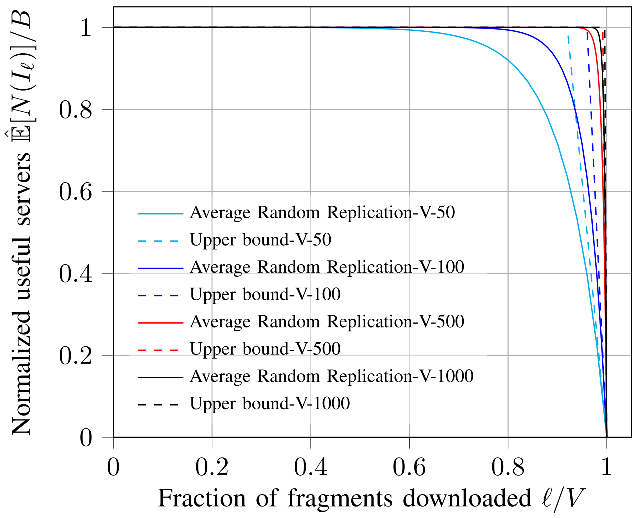

Fig. 8. We plot the upper bound on the normalized number of useful servers in Eq. (10) and the empirical average of the normalized number of useful servers given in Eq. (32) for random replication scheme with random scheduling, as the number of fragments V increases in the set $\{50, 100, 500, 1000\}$ for a fixed storage capacity per server $\alpha = K/V = 0.25$.

servers. As indicated in Corollary 1, we observe that the ensemble mean of normalized number of useful servers for the proposed random (B, V, R) replication storage scheme with a random scheduler is arbitrarily close to the upper bound for any α-(V, R) replication storage scheme given in Eq. (11) when V is asymptotically large.

Explicit expression for computation of the mean download time for replication storage schemes is given in Eq. (7), in terms of $\mathbb{E}[1/N(I_\ell)]$ for download $\ell \in \{0, , \dots, V-1\}$. Even for our simpler setup, analytical computation of the mean download time remains elusive. We had provided a lower bound on the mean download time in terms of the analytically tractable means $\mathbb{E}[N(I_\ell)]$ for $\ell \in \{0, \dots, V-1\}$. We computed the empirical mean of the file download time for

TABLE X
AVERAGE DOWNLOAD TIMES OF VARIOUS NONADAPTIVE AND ADAPTIVE POLICIES

Storage code	**Average download time**
Cyclic shift based storage-nonadaptive scheduling	
Uniform diversity	139629.39
Smallest index first	141507.86
Pushback	145146.52
Projective plane based storage-nonadaptive scheduling	
Smallest index first	122378.76
Smallest index first-Pushback	122394.19
Uniform diversity	122897.40
Uniform diversity-Pushback	121678.81
Projective plane based storage-adaptive scheduling	
Smallest index first-Harmonic ranking	121002.69
Uniform diversity-Harmonic ranking	120886.04
Smallest index first-Pushback-Harmonic ranking	120940.41
Uniform diversity-Pushback-Harmonic ranking	120993.85
Uniform diversity-Greedy	121617.66
Cyclic shift based storage-adaptive scheduling	
Uniform diversity-Harmonic ranking	126722.19
Smallest index first-Harmonic ranking	126769.84
Pushback-Harmonic ranking	126783.50

proposed replication storage schemes and scheduling policies, and the corresponding lower bound. We observe that for all values of selected system parameters, they remain very close. The following Table X shows the empirical mean download times for various storage schemes and scheduling policies.

Discussion. Storage schemes based on combinatorial designs perform better than the naive schemes such as the cyclic shift based storage schemes. The scheduling policy significantly affects the performance for the same storage code.

For any storage code, adaptive scheduling leads to better performance over nonadaptive scheduling. The initial set of fragments placed at the head of the servers does not noticeably affect the performance of the adaptive scheduling algorithms. Adaptive scheduling also reduces the dependence on the storage code as can be seen that both cyclic and design based storage schemes lead to comparable performance.

X. CONCLUSION

A. Summary

We considered a single file that is divided into finitely many fragments. Each fragment is replicated an identical number of times, and the replicas are stored on a finite server system with finite storage capacity. A file is considered downloaded, if one can reconstruct the file from the downloaded fragments. We posed the problem of optimal storage scheme and scheduling policy to minimize the mean download time of the entire file, given that each fragment download time is random *i.i.d.* and memoryless. We modified the problem from the minimization of mean download time to the maximization of the aggregate number of useful servers over all fragment downloads. We subdivided this problem into two subproblems. The first subproblem was to find the optimal scheduling policy given a storage scheme. The second subproblem was to find the optimal storage scheme for a given scheduling policy. We provided lower bounds on the number of useful servers, and proposed design based storage schemes that maximize this lower bound. Further, we posed the optimal adaptive scheduling policy as an MDP, and provided suboptimal solutions. We empirically verified that the proposed solution is close to the upper bound on the number of useful servers, when the number of fragments is large.

B. Discussion and Further Directions

Our storage scheme and scheduling policy can be generalized to other codes as well. The problem of optimal storage and scheduling policy exists for all coded storage schemes, except for MDS coded storage. For MDS coded storage scheme, all fragments are useful until V coded fragments are downloaded. We showed that MDS coded storage minimizes the mean download time. However, we observed that either when the servers have sufficiently large storage or when the number of fragments is large, the mean download time performance of replication coded storage is competitive to that of MDS coded storage.

We note that the exponential service distribution assumption was needed to compute the expression for mean download time in terms of number of useful servers. Our study can be extended to any service distribution that leads to a smaller mean download time for a more dominant sequence of number of useful servers. The mean download time for a general distribution for fragment download time can also be minimized if it was a decreasing function of the sequence of number of useful servers. In this case, the design based storage schemes will continue to do well, since they attempt to maximize the number of useful servers. However, for non-memoryless distribution, the scheduling policies are more complicated, since one needs to take care of age of previously scheduled fragment replicas, which are not yet downloaded.

We also note that, even though our study was for a single file, our proposed framework can be extended to multiple files. Specially, when we assume that each server has a predetermined fraction of storage for each file, one can find optimal storage scheme for each of these files separately. However, the scheduling for multiple files becomes somewhat more involved in this case. One has to schedule a fragment at each server that could be useful for one of these files. For a specific file, one of our proposed scheduling algorithms can select the fragment to be scheduled. However, it is not clear *apriori*, which one of the files should be scheduled.

In our studies, we have ignored the delays in cancelling the scheduled replicas. We observe that these delays affect other coding policies as well. There is a subtle difference though. Each download for a replication coded storage, leads to cancellation at all other servers where the identical replica was being downloaded. This can lead to a maximum of $(R-1)$ cancellations per download. In MDS coded storage, there are no cancellations until the Vth download, when every other scheduled fragment at other $N(I_{V-1}) - 1$ servers should be cancelled.

As can be seen from the preceding discussion, there are many interesting directions of practical import to explore; we hope that this work motivates further research in this topic.

Appendix A
Proof of Lemma 3

A countable state process with sample paths that are right continuous with left limits, is Markov if (a) the inter-transition times are memoryless, (b) conditioned on the current state the future inter-transition times are independent of the past, and (c) the jump probabilities depend only on the current state [54, Chapter 5]. We will show that the countable state process X satisfies all three conditions given a completely utilizing replication storage scheme $\Phi \in \mathcal{S}$ and associated work conserving scheduling policy Ψ_Φ.

Recall that each sample path of the process X is piece-wise constant, and transitions only at the download instants. Thus, the process X is right continuous with left limits. After the ℓth download, the time to download the next fragment $v_{\ell+1}$ is the minimum of the residual download time at each of the $N(I_\ell)$ useful servers given the current state I_ℓ. As the service times at all servers are *i.i.d.* and exponentially distributed with rate μ, it follows that the time for next download $D_{\ell+1} - D_\ell$ is exponentially distributed with rate $N(I_\ell)\mu$. The memoryless property of the service times also implies that the residual download time at each useful server is independent of the past. Thus, future inter-transition times $(D_{j+1} - D_j : j \geqslant \ell)$ are independent of the past conditioned on the current state I_ℓ. In addition, since residual fragment download times are identically exponentially distributed, it follows that probability of any of the useful servers finishing first is $\frac{1}{N(I_\ell)}$. Since the scheduled fragment on each useful server depends only on the current state I_ℓ, the transition probability from current state I_ℓ to the next state $I_{\ell+1}$ depends only on the current state. This transition probability is denoted $p_{I_\ell, I_\ell \cup \{v\}} = \frac{\sum_{b \in U(I_\ell)} \mathbb{1}_{\{\Psi_\Phi(I_\ell)(b)=v\}}}{N(I_\ell)}$. We observe that the three conditions outlined above are met by the process X and hence, the result holds. ■

Acknowledgment

The authors thank the reviewers for their comments which helped improve the presentation. In particular, they thank one of the reviewers for pointing out a gap in the proof of Theorem 6 in a preliminary version of this manuscript. In addition, the they would like to acknowledge helpful discussions with Prof. Manjunath Krishnapur. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

References

[1] R. Jinan, A. Badita, P. Sarvepalli, and P. Parag, “Low latency replication coded storage over memory-constrained servers,” in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2021, pp. 2340–2345.
[2] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, “Network coding for distributed storage systems,” *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4539–4551, Sep. 2010.
[3] 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.
[4] B. Li, A. Ramamoorthy, and R. Srikant, “Mean-field analysis of coding versus replication in large data storage systems,” *ACM Trans. Modeling Perform. Eval. Comput. Syst.*, vol. 3, no. 1, pp. 1–28, Feb. 2018.
[5] K. Lee, R. Pedarsani, and K. Ramchandran, “On scheduling redundant requests with cancellation overheads,” *IEEE/ACM Trans. Netw.*, vol. 25, no. 2, pp. 1279–1290, Apr. 2017.
[6] K. Gardner, M. Harchol-Balter, A. Scheller-Wolf, and B. Van Houdt, “A better model for job redundancy: Decoupling server slowdown and job size,” *IEEE/ACM Trans. Netw.*, vol. 25, no. 6, pp. 3353–3367, Dec. 2017.
[7] P. Parag, A. Bura, and J.-F. Chamberland, “Latency analysis for distributed storage,” in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2017, pp. 1–9.
[8] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, “Speeding up distributed machine learning using codes,” *IEEE Trans. Inf. Theory*, vol. 64, no. 3, pp. 1514–1529, Mar. 2018.
[9] Y. Wu, A. G. Dimakis, and K. Ramchandran, “Deterministic regenerating codes for distributed storage,” in *Proc. Annu. Allerton Conf. Control, Comput., Commun. (Allerton)*, Urbana-Champaign, IL, USA, Sep. 2007, pp. 1–5.
[10] A. Badita, P. Parag, and J.-F. Chamberland, “Latency analysis for distributed coded storage systems,” *IEEE Trans. Inf. Theory*, vol. 65, no. 6, pp. 4683–4698, Apr. 2019.
[11] R. Bitar and S. E. Rouayheb, “Staircase codes for secret sharing with optimal communication and read overheads,” *IEEE Trans. Inf. Theory*, vol. 64, no. 2, pp. 933–943, Feb. 2018.
[12] R. Bitar, P. Parag, and S. E. Rouayheb, “Minimizing latency for secure distributed computing,” in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 2900–2904.
[13] R. Bitar, P. Parag, and S. E. Rouayheb, “Minimizing latency for secure coded computing using secret sharing via staircase codes,” *IEEE Trans. Commun.*, vol. 68, no. 8, pp. 4609–4619, Aug. 2020.
[14] S. E. Mann, “The original view of Reed–Solomon coding and the Welch–Berlekamp decoding algorithm,” Ph.D. dissertation, Dept. Appl. Math., Univ. Arizona, Tucson, AZ, USA, 2013. [Online]. Available: http://hdl.handle.net/10150/301533
[15] K. Ren, Y. Kwon, M. Balazinska, and B. Howe, “Hadoop’s adolescence: An analysis of Hadoop usage in scientific workloads,” *Proc. VLDB Endowment*, vol. 6, no. 10, pp. 853–864, Aug. 2013.
[16] F. Maturana, V. S. C. Mukka, and K. V. Rashmi, “Access-optimal linear MDS convertible codes for all parameters,” in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2020, pp. 577–582.
[17] R. M. Roth, *Introduction to Coding Theory*. Cambridge, U.K.: Cambridge Univ. Press, 2006.
[18] S. Pawar, N. Noorshams, S. El Rouayheb, and K. Ramchandran, “DRESS codes for the storage cloud: Simple randomized constructions,” in *Proc. IEEE Int. Symp. Inf. Theory Proc.*, Jul. 2011, pp. 2338–2342.
[19] B. Zhu, H. Li, H. Hou, and K. W. Shum, “Replication-based distributed storage systems with variable repetition degrees,” in *Proc. 20th Nat. Conf. Commun. (NCC)*, Feb. 2014, pp. 1–5.
[20] D. Wang, G. Joshi, and G. Wornell, “Using straggler replication to reduce latency in large-scale parallel computing,” *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 43, no. 3, pp. 7–11, 2015.
[21] A. Badita, P. Parag, and V. Aggarwal, “Optimal server selection for straggler mitigation,” *IEEE/ACM Trans. Netw.*, vol. 28, no. 2, pp. 709–721, Apr. 2020.
[22] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Hoboken, NJ, USA: Wiley, 1994.
[23] C. Suh and K. Ramchandran, “Exact-repair MDS codes for distributed storage using interference alignment,” in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Dec. 2010, pp. 161–165.
[24] S. El Rouayheb and K. Ramchandran, “Fractional repetition codes for repair in distributed storage systems,” in *Proc. 48th Annu. Allerton Conf.*, Sep. 2010, pp. 1510–1517.
[25] K. V. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, “A solution to the network challenges of data recovery in erasure-coded distributed storage systems: A study on the Facebook warehouse cluster,” in *Proc. USENIX Conf. Hot Topics Storage File Syst.*, 2013, p. 8.
[26] A. S. Rawat, D. S. Papailiopoulos, A. G. Dimakis, and S. Vishwanath, “Locality and availability in distributed storage,” *IEEE Trans. Inf. Theory*, vol. 62, no. 8, pp. 4481–4493, Aug. 2016.
[27] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran, “Explicit codes minimizing repair bandwidth for distributed storage,” in *Proc. IEEE Inf. Theory Workshop (ITW)*, Jan. 2010, pp. 1–5.
[28] D. Papailiopoulos, A. G. Dimakis, and V. Cadambe, “Repair optimal erasure codes through Hadamard designs,” *IEEE Trans. Inf. Theory*, vol. 59, no. 5, pp. 3021–3037, May 2013.

[29] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction," *IEEE Trans. Inf. Theory*, vol. 57, no. 8, pp. 5227–5239, Aug. 2011.
[30] K. W. Shum and Y. Hu, "Cooperative regenerating codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 11, pp. 7229–7258, Nov. 2013.
[31] A. Wang and Z. Zhang, "Repair locality with multiple erasure tolerance," *IEEE Trans. Inf. Theory*, vol. 60, no. 11, pp. 6979–6987, Nov. 2014.
[32] A. Wang, Z. Zhang, and M. Liu, "Achieving arbitrary locality and availability in binary codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2015, pp. 1866–1870.
[33] J. Dean and L. A. Barroso, "The tail at scale," *Commun. ACM*, vol. 56, no. 2, pp. 74–80, Feb. 2013.
[34] 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.
[35] G. Liang and U. C. Kozat, "Use of erasure code for low latency cloud storage," in *Proc. 52nd Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2014, pp. 576–581.
[36] G. Joshi, E. Soljanin, and G. Wornell, "Efficient replication of queued tasks for latency reduction in cloud systems," in *Proc. Allerton Conf. Commun., Control, Comput.*, Sep. 2015, pp. 107–114.
[37] A. Vulimiri, P. B. Godfrey, R. Mittal, J. Sherry, S. Ratnasamy, and S. Shenker, "Low latency via redundancy," in *Proc. 9th ACM Conf. Emerg. Netw. Exp. Technol.*, Dec. 2013, pp. 283–294.
[38] G. Joshi, E. Soljanin, and G. W. Wornell, "Queues with redundancy: Latency-cost analysis," *SIGMETRICS Perform. Eval. Rev.*, vol. 43, no. 2, pp. 54–56, Sep. 2015.
[39] K. Gardner *et al.*, "Reducing latency via redundant requests: Exact analysis," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 43, no. 1, pp. 347–360, 2015.
[40] Y. Xiang, T. Lan, V. Aggarwal, and Y.-F. R. Chen, "Joint latency and cost optimization for erasure-coded data center storage," *IEEE/ACM Trans. Netw.*, vol. 24, no. 4, pp. 2443–2457, Aug. 2016.
[41] P. Parag and J.-F. Chamberland, "Novel latency bounds for distributed coded storage," in *Proc. Inf. Theory Appl. Workshop (ITA)*, Feb. 2018, pp. 1–9.
[42] A. Badita, P. Parag, and V. Aggarwal, "Sequential addition of coded subtasks for straggler mitigation," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Jul. 2020, pp. 746–755.
[43] A. Badita, P. Parag, and V. Aggarwal, "Single-forking of coded subtasks for straggler mitigation," *IEEE/ACM Trans. Netw.*, vol. 29, no. 6, pp. 2413–2424, Dec. 2021.
[44] L. Huang, S. Pawar, H. Zhang, and K. Ramchandran, "Codes can reduce queueing delay in data centers," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2012, pp. 2766–2770.
[45] D. Cheng, J. Rao, Y. Guo, C. Jiang, and X. Zhou, "Improving performance of heterogeneous MapReduce clusters with adaptive task tuning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 3, pp. 774–786, Mar. 2017.
[46] A. O. Al-Abbasi, V. Aggarwal, and T. Lan, "TTLoC: Taming tail latency for erasure-coded cloud storage systems," *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 4, pp. 1609–1623, Dec. 2019.
[47] A. O. Al-Abbasi and V. Aggarwal, "Video streaming in distributed erasure-coded storage systems: Stall duration analysis," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1921–1932, Aug. 2018.
[48] T. M. Cover and J. A. Thomas, *Elements of Information Theory* (A Wiley-Interscience Publication). Hoboken, NJ, USA: Wiley, 2006.
[49] R. Bellman, *Dynamic Programming* (Rand Corporation Research Study). Princeton, NJ, USA: Princeton Univ. Press, 1957.
[50] D. R. Stinson, *Combinatorial Designs: Constructions and Analysis*. Cham, Switzerland: Springer, 2003.
[51] R. Durrett, *Probability: Theory Examples*. Cambridge, U.K.: Cambridge Univ. Press, 2019, vol. 49.
[52] R. Durrett, *Probability: Theory and Examples*, vol. 49. Cambridge, U.K.: Cambridge Univ. Press, 2019.
[53] P. Gopalan, G. Hu, S. Kopparty, S. Saraf, C. Wang, and S. Yekhanin, "Maximally recoverable codes for grid-like topologies," in *Proc. 28th Annu. ACM-SIAM Symp. Discrete Algorithms*, Jan. 2017, pp. 2092–2108.
[54] S. M. Ross, *Stochastic Processes (Wiley Series in Probability and Statistics)*, 2nd ed. Hoboken, NJ, USA: Wiley, Feb. 1995.

Rooji Jinan (Graduate Student Member, IEEE) received the B.Tech. degree in electronics and communication engineering and the M.Tech. degree in communication engineering and signal processing from the University of Calicut, Kerala, in 2012 and 2015, respectively. She is currently pursuing the Ph.D. degree with the Robert Bosch Centre for Cyber-Physical Systems, Indian Institute of Science, Bengaluru. She worked as an Assistant Professor at the Christ College of Engineering, Kerala, from 2016 to 2017. Her research interests include real time communication systems, low latency distributed storage, and compute systems.

Ajay Badita (Member, IEEE) received the B.Tech. degree in electronics and communication engineering from SSCE, affiliated to JNTU Kakinada in 2011, the M.Tech. degree in electronics and communication engineering from NIT Rourkela in 2015, and the Ph.D. degree in electronics and communication engineering from the Indian Institute of Science (IISc), Bengaluru, in 2021.

He has been a Research Scientist at the IOTA Foundation, Berlin, since 2021. His research interests include distributed ledgers, delay-sensitive communication, computation, and storage in distributed systems.

Pradeep Kiran Sarvepalli received the B.Tech. degree in electrical engineering from IIT Madras and the master's degree in electrical engineering and the Ph.D. degree in computer science from Texas A&M University. Following his Ph.D. degree, he was a Post-Doctoral Fellow at The University of British Columbia and the Georgia Institute of Technology. Before his doctoral studies, he worked as an IC Design Engineer with Texas Instruments India, Bengaluru. He is currently an Associate Professor with the Department of Electrical Engineering, IIT Madras. His research interests include quantum and classical error correcting codes, quantum cryptography, quantum computation, and distributed storage.

Parimal Parag (Senior Member, IEEE) received the B.Tech. and M.Tech. degrees in electrical engineering from IIT Madras in 2004 and the Ph.D. degree in electrical engineering from Texas A&M University in 2011. He joined the Indian Institute of Science in 2014, where he is currently an Associate Professor with the Department of Electrical Communication Engineering. Prior to that, he was a Senior System Engineer (Research and Development) with ASSIA Inc., Redwood City, CA, USA, from 2011 to 2014. His research interests include the design and analysis of large scale distributed systems. He was the coauthor of the 2018 IEEE ISIT Student Best Paper. He was a recipient of the 2017 Early Career Award from the Science and Engineering Research Board.