DRESS Codes for the Storage Cloud: Simple Randomized Constructions

Sameer Pawar, Nima Noorshams, Salim El Rouayheb, Kannan Ramchandran
Dept. of Electrical Engineering and Computer Sciences
University of California, Berkeley
{spawar, nshams, salim, kannanr}@eecs.berkeley.edu

***Abstract*—We introduce an efficient family of exact regenerating codes for data storage in large-scale distributed systems. We refer to these new codes as Distributed Replication-based Exact Simple Storage (DRESS) codes. A key property of DRESS codes is their very efficient *distributed* and *uncoded repair* and *growth* processes that have minimum bandwidth, reads and computational overheads. This property is essential for large-scale systems with high reliability and availability requirements.**

DRESS codes will first encode the file using a Maximum Distance Separable (MDS) code, then place multiple replicas of the coded packets on different nodes in the system. We propose a simple and flexible randomized scheme for placing those replicas based on the balls-and-bins model. Our construction showcases the power of the probabilistic approach in constructing regenerating codes that can be efficiently repaired and grown.

I. Introduction

Cloud storage is a growing paradigm for providing online storage of data and, thereby, making it accessible anywhere and anytime. Such services are now being offered by data centers, such as the ones run by Google [1] and Amazon [2], or peer-to-peer (p2p) systems such as Wuala [3]. These distributed storage systems (DSS) rely on large distributed networks of inexpensive and individually unreliable storage nodes to reliably store the data by leveraging the power of redundancy. New nodes are frequently added to the system to *repair* it from failures and maintain its *reliability*. This is accomplished by replacing the nodes that were lost, due for example to hardware failure or peer churning, by new nodes. Moreover, new nodes are added to *grow* the system and increase the *availability* of popular data. This is needed to be able to serve the data to a larger number of users, or to reduce delays by storing the data on nodes that are geographically closer to new or mobile users. In either case, it is important to make this process fast and efficient in order to lower the cost on the system and reduce its downtime.

Most practical DSS nowadays use "off-the-shelf" erasure codes, such as replication [1] or Reed-Solomon (RS) codes [3], to achieve the reliability and availability requirements. These codes offer different system tradeoffs. While RS codes are more efficient than replication in terms of storage space, they incur a higher bandwidth cost since regenerating lost coded data requires downloading and decoding the whole file. As distributed storage systems keep scaling and become more and more distributed, many of the system resources, such as storage capacity, bandwidth, disk I/O and energy, are becoming a bottleneck. Therefore, it is important to understand the theoretical trade-offs that exist among these different resources and construct codes that can achieve these tradeoffs, thus offering practitioners more choices depending on the application.

This research was funded by an NSF grant (CCF-0964018), a DTRA grant (HDTRA1-09-1-0032), and in part by an AFOSR grant (FA9550-09-1-0120).

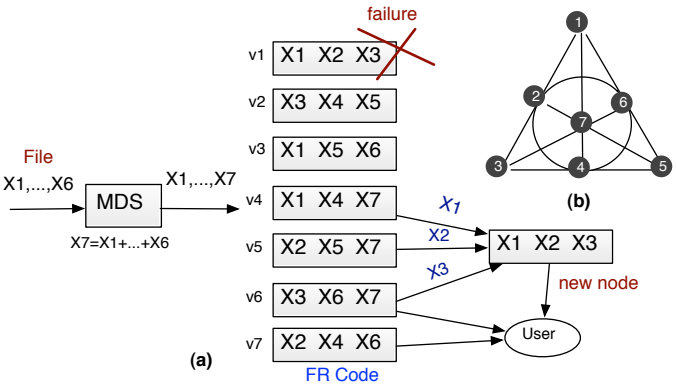

Fig. 1. (a) An example of the minimum-bandwidth regenerating codes proposed in [4] having exact and uncoded repair. These codes form the motivation for the DRESS code construction proposed in this paper. The DSS here has parameters $(n, k, d) = (7, 3, 3)$, where n is the total number of nodes, k is the number of nodes contacted by the user, and d is the number of packets per node. A file of size 6 packets (see Eq. (1)) is first encoded using an MDS code that appends to it a parity check packet. Then, an inner code, referred to as Fractional Repetition (FR) code, places 3 replicas of each coded packet in the system. Any user contacting 3 nodes will observe 6 distinct packets and can recover any missing file packet by a simple xor operation. Notice also the uncoded repair process illustrated when node v_1 fails and is replaced by a new node that downloads its data from the helper nodes v_4, v_5 and v_6. The system is then repaired with minimum reads, minimum download bandwidth and no computations. (b) The Projective plane of order 2, also known as the Fano plane. The pattern of the FR code is derived by associating points in the Fano plane to coded packets and lines (including the circle) to storage nodes (see [4] for more details).

Recently, Dimakis et al. demonstrated in [5] a fundamental tradeoff between storage capacity and bandwidth in DSS. They also introduced a new class of codes, called *regenerating codes*, that achieve any point in this tradeoff. While regenerating codes studied in the literature minimize the bandwidth needed for repair, they generally incur high disk I/O (for reads) and computational overheads. They require a helper node to read all its stored data in order to form linear combinations of it and send it to the new node joining the system. As the storage per node keeps increasing, this means that each helper node has to read and process data in the order of terabytes while the read bandwidth of disks is still in the order of 100 MB/s [6]. This can result in high computational cost

and excessive delays that cannot be tolerated in systems with stringent reliability and availability requirements.

In this paper, we are interested in constructing regenerating codes with efficient and low-complexity distributed repair and growth processes that require minimum bandwidth, minimum disk reads and no computations. This is achieved by imposing what we refer to as the *uncoded repair and growth* property on the code. Moreover, we want the repair process to be *exact*, *i.e.*, we want to always reproduce an exact copy of the lost data on a new node replacing a failed one[1].

Codes that minimize reads when repairing systematic nodes were studied in [8] for RAID-like systems (few parity nodes) where storage is minimized instead of bandwidth. In contrast, we are interested in large-scale distributed systems with low-rate codes (large number of parity nodes). Such systems are intended to provide ubiquitous and pervasive presence of the data for applications that demand high availability and serving mobile users. In these systems, the failure of parity nodes is a frequent event that should be handled efficiently along with growth. Deterministic constructions of codes for such systems that have exact and uncoded repair were proposed in [4] by a subset of the authors. These codes are formed by the concatenation of an outer Maximum Distance Separable (MDS) code with an inner code dubbed *Fractional Repetition* (FR) code that replicates the coded packets across the system (see Fig. 1 for a detailed example). The existence of these codes was shown to depend on the existence of some combinatorial objects such as projective spaces and Steiner systems.

The codes in [4] were constructed to maximize the performance of the system for the worst-case user. In this paper, we advocate a probabilistic approach that permits more flexibility in terms of possible system parameters and produces codes with the desired repair and growth properties. Our proposed codes are based on the same concatenated design of Fig. 1. However, the inner FR code here randomly places the coded packet replicas on the nodes based on a bins-and-balls model. We call these codes Distributed Replication-based Exact Simple Storage (DRESS) codes. The benefits of the probabilistic construction comes at the expense of sacrificing the "perfect" MDS property of the code and allowing a small overhead in the number of nodes contacted by a user, much like the case of Fountain codes. Moreover, our codes require a repair table that indicates to a new node joining the system which specific nodes it can contact to download its data.

The rest of the paper is organized as follows: In Section II we describe our model for distributed storage systems. In Section III, we describe our DRESS code construction and motivate it using simulation results. Section IV provides summary of our main results followed by a discussion in Section V. Section VI contains the theoretical analysis of our constructed codes. We conclude the paper in Section VII and discuss some related open problems.

[1]Exact repair is a desired property for system reasons such as preserving the systematic form of the data and keeping the code alphabet size small, and also for security purposes [7]. Moreover, exactness guarantees that the repair process does not compromise desired properties in the initial code (before any repairs). For instance, starting with a Fountain code, the decoding at the user side will always have a complexity that is linear in the size of the file at any point in the lifetime of the system.

II. Background and Model

A distributed storage system is defined by the triplet (n, k, d), where n is the total number of storage nodes, $v_1, \ldots, v_n$, in the system, $k < n$ is the number of nodes contacted by the user to retrieve the stored file, and d is the number of helper nodes contacted by a new node when added to the system .

We consider DSS operating in the *minimum-bandwidth regime* (MBR) on the storage vs. bandwidth tradeoff curve described in [9]. Our focus on this regime is motivated by the asymmetrical cost of resources in typical applications where bandwidth is more expensive than storage. In this case, the bandwidth needed for repair or growth, *i.e.*, the total amount of data downloaded by a new node from the helper nodes, is minimized and is equal to the amount of data lost. For load-balancing requirements, we assume that the new node downloads and stores an equal amount of data, referred to as a packet, from each of the d contacted nodes. Therefore, d also represents the node storage capacity expressed in packets. Under this model, the storage capacity C_{MBR} of the DSS in packets, representing the information-theoretic limit on the maximum file size that can be delivered to any user contacting k out of the n nodes, was shown in [9] to be

$$C_{MBR}(n, k, d) = kd - \binom{k}{2}. \tag{1}$$

This expression is based on a *functional repair* model where any d nodes can help in repair and where the only constraint on the data stored on a new node is to maintain the MDS property of the entire code. Thus, the regenerated data can be different from the lost data, but it should be "functionally" equivalent. In this paper, we require a stricter form of repair where an exact copy of the lost data should be reproduced on a replacement node. Recently, Rashmi et al. showed in [10] that, in the minimum bandwidth regime, there is no loss in the DSS capacity incurred by requiring exact repair and constructed codes that achieve the capacity C_{MBR} of (1). We are interested here in constructing codes with exact but also *uncoded* repair that minimizes the disk reads and does not involve data processing when repairing or growing the system.

The literature on regenerating codes implicitly imposes a stringent requirement on the repair process which is required to efficiently recover from up to $n-d$ failures (any d nodes should be able to help in repair). However, this may be unnecessary in large-scale systems where $n >> d$. Typically, DSS are repaired at periodic intervals. Therefore, we assume here that, with high probability, the repair process should be able to tolerate efficiently $\rho' < n - k$ failures, where ρ' is determined based on the disk failure statistics and on the inter-repair time. In the unlikely event of more than ρ', but less then $n - k$, failures the system can implement a costly repair where each new node downloads and decodes the whole file.

Consider a file F of size R packets. A regenerating code with exact and uncoded repair can be constructed in the following way. First, F is encoded using an outer (θ, R) MDS code, where $\theta = \frac{nd}{\rho}$, and ρ a code parameter called *replication factor*. The coded packets are then replicated on ρ

distinct nodes following a certain pattern dictated by an inner Fractional Repetition code. By choosing $\rho = \rho' + 1$, there will always be, with high probability, a surviving copy of any lost packet which can be efficiently replicated on new nodes when repairing the system[2]. A good design rule for the FR code, that guarantees that all users observe at least C_{MBR} packets, is to make sure that nodes have at most one packet in common. Combinatorial constructions based on this rule were studied in [4]. We are interested here in code construction that allow more flexibility in terms of admissible system parameters and that can be easily grown when appending more nodes to the system. Next, we will describe a simple randomized construction that achieves these desired properties.

III. Construction and Simulation

A DRESS code for a DSS with parameters (n, k, d) having a replication factor ρ is constructed in the following way. First, a file of size R is encoded using a (θ, R) MDS code with $\theta = \frac{nd}{\rho}$. We think of the n storage nodes as n bins and the θ MDS coded packets as θ balls having distinct colors. The code is then obtained by throwing successively in each bin a set of d distinct balls picked uniformly at random from the possible $\binom{\theta}{d}$ subsets of balls. The sets of balls corresponding to the different bins are assumed to be picked independently and with replacement.

In this setting, a user contacts k bins and counts the number of distinct colors it can observe. This corresponds to the number of distinct coded packets that it can download. By the MDS property of the outer code, the user can always decode the stored file whenever the number of observed colors is larger or equal to the file size R. The number of observed colors will be relatively small when the contacted bins have a large number of colors in common. We will show that, for the system parameters of interest, this is a rare event and that the number of colors observed by a user is highly concentrated around its mean.

Simulations: To motivate our proposed construction, we provide simulation results for the performance of DRESS codes for an example of a DSS with $(n, k, d) = (400, 10, 15)$ and a replication factor $\rho = 20$. The obtained histogram depicted in Fig. 2 shows a concentration of the number of colors observed by a user around its mean which is roughly 118 colors. Moreover, a user can decode a file of size $R = C_{MBR} = 105$ packets with more than 99% probability.

The simulations results indicate the benefits of abandoning a worst-case design of the system in favor of a probabilistic one that allows a small percentage of users that cannot decode the file. This way we can easily construct codes for distributed storage that have efficient repair and growth capabilities.

IV. Theoretical Results

To simplify the theoretical analysis, we relax the condition that the d balls thrown in each bin are of distinct colors and analyze a slightly different construction where these d balls that go in each bin are picked independently and uniformly

[2]We assume that there is a repair table in the system that stores information on which packets are stored on each node. This repair table can be contacted by the new nodes added to the system.

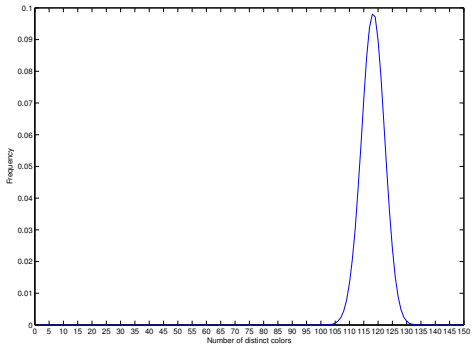

Fig. 2. Performance of DRESS codes based on the balls-and-bins randomized construction for a distributed storage system with parameters $(n, k, d) = (400, 10, 15)$ and $\rho = 20$. The graph shows the histogram of the number of distinct colors (coded packets) observed by a single user contacting, without loss of generality, the first k nodes. The numbers are based on randomly generating 10^7 code instances.

at random (with replacement) from the set of θ balls. It is clear that the DRESS code construction of Section III always performs better than this construction. Moreover, when the number of colors θ is large (large n and small ρ and d), which is the typical range of system parameters, this relaxation will give a good approximation of the performance of DRESS codes. The analysis in the remainder of the paper will focus only on this relaxed construction.

Let $\mathcal{I}, \mathcal{I} \subset \{1, \ldots, n\}$, be a set of size k containing the indices of the k different storage nodes contacted by a user. Denote by $F_{\mathcal{I}}$ the random variable representing the total number of distinct colors (coded packets) stored on the nodes indexed by $\mathcal{I}$. Recall also that, by the MDS property of the outer code, the user will be able to recover the stored file whenever $F_{\mathcal{I}}$ is larger or equal then the size of the stored file, *i.e.*, $R \geq F_{\mathcal{I}}$. It can be shown that the probability distribution of $F_{\mathcal{I}}$ has the following expression [11], [12] ,

$$\mathbb{P}\left(F_{\mathcal{I}} = f\right) = \binom{\theta}{f} \sum_{i=0}^{f} (-1)^i \binom{f}{i} \left(\frac{f-i}{\theta}\right)^{kd}, \qquad (2)$$

for $f = 1, \ldots, \min\{\theta, kd\}$. In addition to the above classical result, we provide here a new upper bound on the probability distribution of $F_{\mathcal{I}}$ that is more amenable to engineering interpretation and that highlights the interplay among the different system parameters. This bound is given in the following theorem which says that for DRESS codes $F_{\mathcal{I}}$ is concentrated around its mean F_{ave}. Therefore, any user will observe roughly F_{ave} colors with a high probability.

Theorem 1: Consider a DRESS code of replication factor ρ obtained from the randomized construction described in Sec. III and used for an (n, k, d) DSS. Then we have:

a) The average distinct colors observed by a user is

$$F_{ave} := \mathbb{E}[F_{\mathcal{I}}] = \theta \left(1 - (1 - \frac{1}{\theta})^{kd}\right). \qquad (3)$$

b) Furthermore, $F_{\mathcal{I}}$ is concentrated around its mean. More precisely, for any fixed $B > 0$, we have

$$\mathbb{P}\left(|F_{\mathcal{I}} - F_{ave}| \geq B\right) \leq 2 \exp\left(-\frac{B^2}{2\sigma^2}\right), \qquad (4)$$

where $\sigma^2 := \frac{\theta^2(1-(1-\frac{1}{\theta})^{2kd})}{2\theta-1}$.

c) We also have

$$\mathbb{P}\left(|F_{\mathcal{I}} - F_{ave}| \geq B\right) \leq 2\exp\left(-\frac{B^2}{2kd}\right). \quad (5)$$

The bound in (4) is stronger, but equation (5) gives better intuition on the system performance.

Corollary 2: For any given $0 < \delta < 1$, a user will be able to decode a file of size $F_{ave} = \theta(1-(1-1/\theta)^{kd})$ with a success probability equal to $1-\delta$ as long as he contacts at least $k' = k(1+\epsilon)$ nodes, where

$$\epsilon \geq \frac{\log\left(1 - \frac{\sqrt{2\sigma^2\log(1/\delta)}}{\theta(1-1/\theta)^{kd}}\right)}{kd\log(1-1/\theta)}. \quad (6)$$

V. Discussion

Before providing the proof to Th. 1, we discuss the effect of the probabilistic construction on the system parameters.

A. File Size

Since the number of colors observed by a user is concentrated around its mean F_{ave}, the size of the stored file can be chosen to be $R = F_{ave}(k)$. If a user contacting k nodes happens to observe less than F_{ave}, we allow him to contact a few extra nodes so that with high probability it will be able to recover the file. The idea is similar to the Fountain codes setting, where a small overhead for k is allowed. Using the one-sided version of the bound in (4), one can compute the probability of not being able to decode the file when the user contacts $k' \geq k$ nodes as,

$$\mathbb{P}\left(F_{\mathcal{I}}(k') < R\right) \quad \leq \quad \exp\left(-\frac{(F_{ave}(k') - R)^2}{2\sigma^2}\right). \quad (7)$$

Cor. 2 follows immediately from (7) by taking $R = F_{ave}(k)$. Consider our example with $(n,k,d) = (400,10,15)$ and $\rho = 20$, Cor. 2 suggests contacting up to $k' = 13$ nodes to guarantee that a user can recover the file of size $R = F_{ave} = 118$ with a 96.5% success probability. Indeed, numerical simulations show that for $k' = 10, 11, 12$ the probability of recovering the file is $47\%, 97.5\%$ and 99.99%, respectively.

Alternatively, if the number of nodes a user contacts cannot exceeds k, one can back off from F_{ave} and choose $R = F_{ave} - B < F_{ave}$ depending on the probability of error $\delta < 0.5$ that can be tolerated. The file size R can be deduced from Eq. (4) which gives $R = F_{ave} - \sqrt{2\sigma^2\log(1/\delta)}$. For instance, for the system with $(n,k,d) = (400,10,15)$ and $\rho = 20$, Eq. (4) gives that taking $R = 88$ packets guarantees that each user can decode the file with probability equal to 99%. Note that the simulation results of Fig. 2 show that it is possible to take $R = 108$ packets and still guarantee a 99% success probability.

B. Replication Factor

Recall that the system requires each packet to be replicated ρ times. However, with the randomized construction the number of replicas of each coded packet is random. Let r denote the random variable indicating the number of replication of a particular packet, say without loss of generality the first one. In other words, r represents the number of distinct bins containing the ball of color 1. It can be shown that r is a binomial random variable $B(n,p)$, with $p = 1 - \left(1-\frac{1}{\theta}\right)^d$. Therefore, r is concentrated around its average $\bar{r} = n\left(1-\left(1-\frac{1}{\theta}\right)^d\right)$ as indicated by the following Chernoff bound:

$$P(r \leq (1-\epsilon)\bar{r}) \quad \leq \quad e^{-\bar{r}\epsilon^2/2}. \quad (8)$$

Note that for large values of θ, the expectation of r is approximatively equal to the desired replication factor ρ, since $\bar{r} \approx nd/\theta = \rho$. To guarantee that the number of replicas is at least ρ with high probability, the balls-and-bins random construction can be applied with an over-provisioned value $\rho^* > \rho$ of the repetition factor which can be obtained from the bound in (8). Going back to our example with $(n,k,d) = (400,10,15)$ and $\rho = 20$, by taking $\rho^* = 34$ one can guarantee that with 90% probability each packet is replicated at least $\rho = 20$ times. Note that the effect of over-provisioning ρ is a decrease in the average file size as dictated by (3). For this example, the average file size drops from 118 (for $\rho = 20$) to 101 packets (for $\rho^* = 34$).

C. Repair and Growth

As mentioned earlier, DRESS codes are intended for large-scale distributed storage systems providing ubiquitous data storage. Therefore, in addition to repair, it is essential to be able to efficiently and distributively grow the system by adding more nodes to it. With the proposed randomized construction, the system can be easily grown or repaired in the following way. In case of growth, each added node will independently pick d distinct colors at random from a total of θ colors, while for repair these d colors correspond to the ones that were stored on the failed node[3]. It then consults the repair table to know which helper nodes can be contacted to download the packets corresponding to its chosen colors. The new system can again be repaired or grown in the same way while reading and downloading the minimum amount of data with no computations, while retaining the same guarantees on the file size delivered to the user. Moreover, in the case of growth, the number of replicas of each packet will increase resulting in better guarantees on the replication factor ρ.

VI. Theoretical Analysis

In this section we present the proof for Th. 1. Recall that there are n bins and θ distinct balls. Each bin selects d balls with replacement from a pool of θ balls, each time drawing a ball uniformly at random.

a) Computing the average: Each user contacts k distinct bins. Hence effectively each user draws kd balls from the pool of θ balls. Let S_i denote the number of distinct colors after picking i balls. By construction we know that $\{S_i\}_{i=0}^{kd}$ forms a Markov chain with transition probability

$$S_{i+1} = \begin{cases} S_i + 1 & \text{with probability } \frac{\theta - S_i}{\theta} \\ S_i & \text{with probability } \frac{S_i}{\theta} \end{cases},$$

[3]The monitoring process that initiates the repair knows the identity of the failed node and can convey it to new node joining the system.

for $i = 0, 1, 2, \cdots, kd-1$, where $S_0 = 0$. Taking the conditional expectation yields

$$\mathbb{E}\left[S_{i+1}|S_i\right] = (S_i+1)\frac{\theta - S_i}{\theta} + S_i\frac{S_i}{\theta} = 1 + (1-\frac{1}{\theta})S_i.$$

Therefore, using the tower property of expectation we obtain $\mathbb{E}\left[S_{i+1}\right] = 1 + (1-\frac{1}{\theta})\mathbb{E}\left[S_i\right]$. Using the initial condition $\mathbb{E}\left[S_0\right] = 0$ and induction on i, one can easily show that $\mathbb{E}\left[S_i\right] = \sum_{j=0}^{i-1}(1-\frac{1}{\theta})^j$. Finally, simplifying this geometric series, we obtain $F_{ave} = \mathbb{E}\left[S_{kd}\right] = \theta\left(1-(1-\frac{1}{\theta})^{kd}\right)$.

b) Concentration result: Let X_i denote the color of the ball drawn at i^{th} round by a user for $i = 1, 2, \cdots, kd$. To reduce the notation overhead, we use X_1^i to denote the the sequence $(X_1, X_2, \cdots, X_i)$. Define $Y_i = \mathbb{E}\left[S_{kd}|X_1^i\right]$ for $i = 0, 1, \cdots, kd$, where S_{kd} is the total number of distinct colors observed by the user. It is known that Y_i forms a Doob martingale, where it can be seen that $F_{\mathcal{I}} = Y_{kd}$, and $\mathbb{E}\left[F_{\mathcal{I}}\right] = F_{ave} = Y_0$. Since S_i is a function of X_1^i, we have $Y_i = \mathbb{E}\left[S_{kd}|X_1^i, S_i\right]$. Moreover, it is clear that knowing S_i, the random variable S_{kd} is independent of X_1^i (S_i is a sufficient statistic). Therefore, we have $Y_i = \mathbb{E}\left[S_{kd}|S_i\right]$ for $i = 0, 1, \cdots, kd$, and

$$\begin{aligned}
\mathbb{E}\left[S_{i+2}|S_i\right] &= \mathbb{E}\left[\mathbb{E}\left[S_{i+2}|S_{i+1}, S_i\right]|S_i\right] \\
&= \mathbb{E}\left[1 + (1-\frac{1}{\theta})S_{i+1}|S_i\right] \\
&= 1 + (1-\frac{1}{\theta})\mathbb{E}\left[S_{i+1}|S_i\right] \\
&= (1-\frac{1}{\theta})^0 + (1-\frac{1}{\theta})^1 + (1-\frac{1}{\theta})^2 S_i.
\end{aligned}$$

In general, one can show that

$$\mathbb{E}\left[S_{i+(kd-i)}|S_i\right] = \sum_{j=0}^{kd-i-1}(1-\frac{1}{\theta})^j + (1-\frac{1}{\theta})^{kd-i}S_i,$$

for $i = 0, 1, 2, \cdots, kd-1$. Therefore, we have

$$\mid Y_i - Y_{i+1} \mid = \mid (1-\frac{1}{\theta})^{kd-(i+1)}\left[1 + (1-\frac{1}{\theta})S_i - S_{i+1}\right] \mid . \tag{9}$$

It can be verified that for two possibilities $S_{i+1} \in \{S_i + 1, S_i\}$, it holds that $|1 + (1-\frac{1}{\theta})S_i - S_{i+1}| \leq 1$, and hence $\mid Y_i - Y_{i+1} \mid \leq (1-\frac{1}{\theta})^{kd-(i+1)}$.

Now we use Azuma-Hoeffding inequality to bound the concentration probability. Having a martingale $\{Y_i\}_{i=0}^{kd}$ with bounded differences $\mid Y_i - Y_{i+1} \mid \leq c_i$ for $i = 0, 1, \cdots, kd-1$, the inequality states that

$$\mathbb{P}\left(|Y_{kd} - Y_0| \geq B\right) \leq 2\exp\left(-\frac{B^2}{2\sum_{i=0}^{kd-1} c_i^2}\right).$$

In our setup, we have $c_i = (1-\frac{1}{\theta})^{kd-(i+1)}$, thus

$$\sum_{i=0}^{kd-1} c_i^2 = \frac{\theta^2(1-(1-\frac{1}{\theta})^{2kd})}{2\theta - 1} := \sigma^2.$$

Putting the pieces together, we obtain

$$\mathbb{P}\left(|F_{\mathcal{I}} - F_{ave}| > B\right) \leq 2\exp\left(-\frac{(2\theta-1)B^2}{2\theta^2(1-(1-\frac{1}{\theta})^{2kd})}\right).$$

To show equation (5), we use the fact that $0 < c_i \leq 1$. Therefore $\sigma^2 = \sum_{i=0}^{kd-1} c_i^2 \leq \sum_{i=0}^{kd-1} c_i \leq kd$. This concludes the proof of Theorem 1.

VII. Conclusion

We proposed new low-complexity regenerating codes for data storage in large-scale distributed systems. A key property of these codes is their very lightweight distributed repair and growth processes which is essential for systems with high availability requirements. This was achieved by requiring the repair and growth processes to be *uncoded*, thus minimizing their associated bandwidth, Disk I/O and computational costs. Our proposed construction consisted of first encoding a file using an MDS codes then placing copies of the coded packets randomly in the system using the balls-and-bins model. We showed that by allowing a small overhead in the number of contacted nodes, the user is able to recover his file with high probability. While one can think of other randomized constructions that may be a better fit in different scenarios, such as when the original data is not co-located or when it is being streamed, our goal here was to showcase the power of the probabilistic approach for constructing efficient codes for distributed storage. Our codes assume the existence of a repair table that stores the system blueprint. In some systems such table may not exist or is hard to handle. An important question that we are currently investigating is how to replace the repair table with an efficient distributed gossip-style algorithm for finding the location of any needed packets.

References

[1] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," in *19th ACM Symposium on Operating Systems Principles*, 2003.

[2] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: Amazon's highly available key-value store," in *ACM SIGOPS*, 2007.

[3] http://www.wuala.com/en/learn/technology.

[4] S. E. Rouayheb and K. Ramchandran, "Fractional repetition codes for repair in distributed storage systems," in *Allerton*, 2010.

[5] A. Dimakis, P. B. Godfrey, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," in *INFOCOM*, 2007.

[6] V. Venkatesan, "Fast rebuilds in distributed storage systems using network coding," tech. rep., IBM Research GmbH, Zurich Research Laboratory, 2009.

[7] S. A. Pawar, S. El Rouayheb, and K. Ramchandran, "Securing dynamic distributed storage systems against eavesdropping and adversarial attacks," in *arXiv:1009.2556v2*, 2011.

[8] I. Tamo, Z. Wang, and J. Bruck, "MDS array codes with optimal rebuilding," *arXiv:1103.3737v1*, 2011.

[9] A. Dimakis, P. Godfrey, Y. Wu, M. Wainright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inform. Theory*, vol. 56, pp. 4539–4551, Sep. 2010.

[10] 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," in *arxiv:1005.4178*, 2010.

[11] W. Feller, *An Introduction to Probability and Its Applications*, vol. 1. Wiley, 1968.

[12] http://www.math.uah.edu/stat/urn/Birthday.pdf.