Cooperative Pipelined Regeneration in Distributed Storage Systems

Jun Li, Xin Wang
School of Computer Science
Fudan University, China

Baochun Li
Department of Electrical and Computer Engineering
University of Toronto, Canada

***Abstract*—In distributed storage systems, a substantial volume of data are stored in a distributed fashion, across a large number of storage nodes. To maintain data integrity, when existing storage nodes fail, lost data are regenerated at replacement nodes. Regenerating multiple data losses in batches can reduce the consumption of bandwidth. However, existing schemes are only able to achieve lower bandwidth consumption by utilizing a large number of participating nodes. In this paper, we propose a cooperative pipelined regeneration process that regenerates multiple data losses cooperatively with much fewer participating nodes. We show that cooperative pipelined regeneration is not only able to maintain optimal data integrity, but also able to further reduce the consumption of bandwidth as well.**

I. Introduction

Distributed storage systems provide a large-volume storage service by storing data in a large number of storage nodes. Due to the large number of storage nodes and their commodity nature, failures of storage nodes should be regarded as the *rule*, rather than *exceptions*. Thus, to maintain data integrity, redundancy should be stored in the system to tolerate failures of storage nodes. When some storage nodes fail to work, data remain available from other operating storage nodes.

The ability of tolerating node failures depends on how redundancy is achieved. Maximum Distance Separable (MDS) codes are able to achieve the optimal tolerance ability. By encoding the original file into n coded blocks and storing coded blocks into n different storage nodes, MDS codes can guarantee the *recoverability property* that any k storage nodes among the n storage nodes can recover the original file.

To maintain the ability to tolerate node failures, data losses incurred by such failures should be regenerated in replacement nodes, called *newcomers*. Traditional MDS codes, such as Reed-Solomon codes [1], can regenerate a coded block only after recovering the entire original data from at least k storage nodes, called *providers*. Inspired by network coding [2], random linear codes [3] and regenerating codes [4] can also maintain the recoverability property with the same storage overhead as traditional MDS codes. Different from traditional MDS codes and random linear codes, regenerating codes introduce an optimal trade-off between storage capacity and bandwidth consumption during regeneration. Specifically, if one storage node is allowed to store $\frac{M}{k}$ bits of a M-bit file, which is the same storage required by traditional MDS codes and random linear codes, Minimum-Storage Regenerating (MSR) codes minimize the consumption of bandwidth to approximately $\frac{M}{k}$ bits when the number of providers is large enough, yet with a higher computational cost than random linear codes [5].

In some distributed storage systems, such as Total Recall [6], in order to prevent unnecessary regeneration incurred by temporary node departures, the regeneration process will not be triggered until a certain number of node failures has been detected, and then data losses are regenerated in batches. Compared with regenerating multiple data losses in different newcomers independently, the cooperation among newcomers can further reduce the consumption of bandwidth [7], [8]. The more nodes participating during regeneration, the less bandwidth will be consumed. However, in practical systems, it is not desirable to engage a large number of nodes during regeneration, due to its weak resilience to node churns and high management complexity. A higher computational overhead will also be introduced by more participating nodes [5]. In some systems where nodes can sleep when they are idle and wake up on demand, more participating nodes in the regeneration process can also result in more energy consumption.

Some techniques (*e.g.,* simple regenerating codes [9]), have been proposed to reduce the number of participating nodes during regeneration and thus incur a much lower number of disk I/O operations on participating nodes during regeneration. However, they either fail to maintain the recoverability property, or are not designed for regenerating multiple data losses. In this paper, we propose a cooperative pipelined regeneration process to regenerate multiple data losses in batches. We introduce a new type of nodes into the regeneration process, called *apprentices*. During each round of pipelined regeneration, apprentices and newcomers all receive data from providers. After regeneration, newcomers are partially regenerated and become apprentices. Apprentices accumulate data in more than one round of pipelined regeneration. After apprentices have received enough data to regenerate coded blocks, they become fully regenerated and finally "graduate" to become storage nodes. As shown in Fig. 1, though newcomers can be fully regenerated after more than one round of cooperative pipelined regeneration, to compensate for data losses, a corresponding

This work was supported in part by the National Science Foundation of China under Grant 61171074, the National S&T Major Project of China under Grant 2010ZX03003-003-03, and Program for New Century Excellent Talents in University under Grant NCET-11-0113. Xin Wang (Email: xinw@fudan.edu.cn) is the corresponding author.

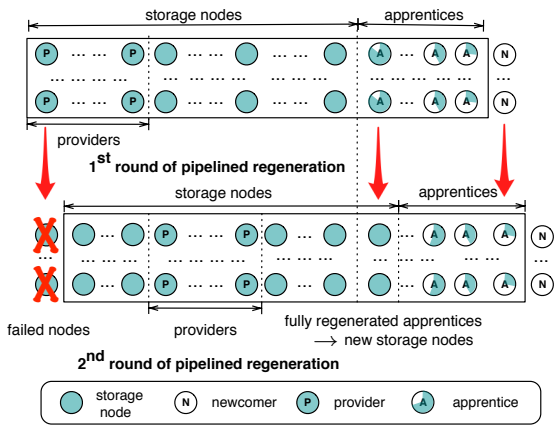

Fig. 1. Two consecutive rounds of cooperative pipelined regeneration.

number of apprentices that have been introduced as newcomers several rounds of regeneration before can be fully regenerated in this round of regeneration, maintaining the recoverability property. Moreover, even though additional apprentices are introduced, the pipelined regeneration process is still able to reduce the number of participating nodes since much fewer providers are utilized during regeneration, implying a much fewer disk I/O operations. Moreover, the bandwidth consumption can also be reduced, which can only be achieved by incurring even more participating nodes during conventional regeneration. We design cooperative pipelined regeneration processes based on random linear codes and regenerating codes, preserving the recoverability property of these codes. Though apprentices introduce additional storage overhead, we show that it is marginal in practical distributed storage systems.

II. Background and Related Work

Distributed storage systems store redundancy to tolerate node failures. Compared with simply storing replications, coded redundancy is able to provide a higher level of data integrity since every coded bit can provide innovative information for data recovery [10]. MDS codes can guarantee the optimal data integrity with the same storage overhead. However, supposing that the size of the original file is M bits and storage nodes store coded blocks of size $\frac{M}{k}$ bits, traditional MDS codes require both receiving M bits from at least k providers and recovering the original file before regenerating only one coded block.

Random linear codes [3] (with a high probability on a large Galois field) can perform as well as traditional MDS codes. Dividing the original file into k blocks and encoding them into random linear combinations on a Galois Field GF(2^q), random linear codes can regenerate a coded block by encoding any k coded blocks without recovering the original file. If the size of the Galois Field GF(2^q) is very large (such as $q = 16$ or 32), random linear codes can guarantee the recoverability property with a high probability after a reasonably large number of regeneration processes, as the upper bound of the probability of failures scales inversely proportionally to the size of the Galois field [11].

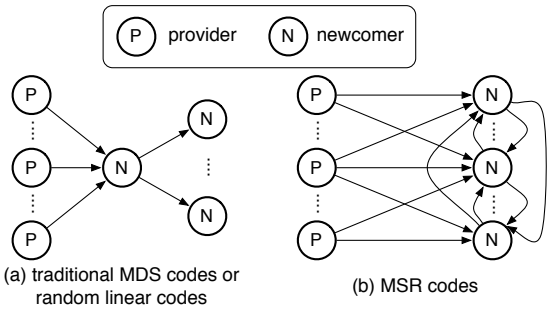

Fig. 2. Conventional regeneration processes with multiple newcomers for traditional MDS codes, random linear codes and MSR codes.

On the other hand, Minimum-Storage Regenerating (MSR) codes [4] are able to regenerate coded blocks with the least bandwidth consumption among MDS codes. MSR codes divide the original file into $k(d-k+1)$ segments ($d \geq k$), and a coded block contains $d-k+1$ coded segments that are linear combinations of these segments on GF(2^q). To regenerate a coded block, the newcomer needs to download only a linear combination of the $d-k+1$ coded segments in a coded block from each of d providers. Thus, the consumption of bandwidth is $\frac{Md}{k(d-k+1)}$ bits, converging to $\frac{M}{k}$ bits when d is large enough. MSR codes can be constructed in a static way [12]. When q is large enough, randomized MSR codes can also guarantee the recoverability property [4] with a high probability. However, Duminuco and Biersack [5] have shown that MSR codes incur a much higher computational overhead than random linear codes during regeneration.

Random linear codes and randomized MSR codes regenerate coded blocks in a functional manner, *i.e.,* the regenerated blocks are not exactly the lost ones, but can be used to recover the original file with any other $k-1$ coded blocks. The construction of exact MSR codes require either a certain combination of d and k [13], [14], or infinite sub-packetization [15], [16], which is impractical to our knowledge. In this paper, we focus on the functional regeneration.

Cooperation among newcomers can help to further reduce the bandwidth consumption during regeneration. As for traditional MDS codes and random linear codes shown in Fig. 2(a), one newcomer can be selected to receive data from k providers and regenerate coded blocks for itself and all other newcomers. If there are r newcomers, the average bandwidth consumed to regenerate one coded block is reduced to $\frac{M}{k} \cdot \frac{k+r-1}{r}$ bits. On the other hand, as for cooperative MSR codes [17][18], supposing that there are d providers and r newcomers during regeneration ($d \geq k, r > 1$), each storage node stores one coded block containing $d-k+r$ coded segments that are linear combinations of the $k(d-k+r)$ segments divided from the original file. During regeneration, each newcomer receives one linear combination of $d-k+r$ coded segments in each provider, and then sends one linear combination of received coded segments to each of other newcomers, as shown in Fig. 2(b). In this process, each newcomer receives $\frac{M}{k} \cdot \frac{d-1+r}{d-k+r}$ bits from d providers and other $r-1$ newcomers. When the number of providers does not change, cooperative regeneration

codes further reduce the consumption of bandwidth during regeneration with an increase of newcomers.

An increase in the number of participating nodes during regeneration may lead to higher management and computational complexity, weaker resilience to node churns and more energy consumption. Duminuco and Biersack [19] have proposed hierarchical codes that introduce a trade-off between data integrity and the consumption of bandwidth during regeneration. Hierarchical codes are able to regenerate a coded block from a subset of k coded blocks. The size of the subset, however, can vary from 2 to k and the recoverability property is no longer maintained. Self-repairing codes [8], [20] minimize the number of coded blocks required during regeneration. To regenerate one coded block, the newcomer can connect to specific pairs of two providers. Nevertheless, the recoverability property can not be maintained, either. Simple regenerating codes [9] and fractional repetition codes [21] can maintain the recoverability property while the newcomer need to connect to a low number of providers during regeneration. However, they both require storage nodes to store data more than what is required by MDS codes. Moreover, though these works can save the consumption of bandwidth by reducing the number of participating nodes during regeneration, none of them can reach the lower bound of the bandwidth consumption achieved by MSR codes, while still preserving the recoverability property.

A pipelined regeneration process with MSR codes [22] has been proposed for a single failure. However, it fails to utilize multiple newcomers cooperatively, and can only support MSR codes. In this paper, we propose new functional regeneration processes that can utilize multiple newcomers in a cooperative fashion, thus further saving the bandwidth consumption. In addition, both random linear codes and MSR codes can be supported by our regeneration processes, and thus the recoverability property can be maintained.

III. System Model and Cooperative Pipelined Regeneration

Suppose that one file of size M bits is stored in the distributed storage system. To maintain the data integrity, the system stores coded blocks of this file, of size $\frac{M}{k}$ bits, into n storage nodes. Coded blocks are encoded such that the recoverability property is maintained, in that any k storage nodes suffice to recover the original file. We discuss two codes that can achieve this property in this paper: random linear codes and Minimum-Storage Regenerating (MSR) codes. Specifically, we use a randomized implementation of MSR codes proposed by Duminuco and Biersack [5] in this paper.

Both random linear codes and MSR codes encode the original file into coded blocks of size $\frac{M}{k}$ bits. Each storage node stores at most one coded block. Specifically, with random linear codes, coded blocks are random linear combinations of k blocks divided from the original file, requiring coded blocks from k providers to regenerate one coded block. With randomized MSR codes, coded blocks contain $N-k$ coded segments that are random linear combinations of $k(N-k)$ segments divided from the original file ($N \geq k$), requiring N participating nodes (including at least k providers) during regeneration. We do not consider the impact of storage and communication overhead incurred by coefficients of coded blocks (coded segments), since the size of coded blocks (coded segments) is significantly larger than coefficients in practical distributed storage systems [11], [5].

As shown in Sec. II, the encoding operations of both random linear codes and MSR codes are performed on a Galois Field GF(2^q). When q equals 16 or 32, the recoverability can be achieved with a high probability. Thus, we do not consider the issue of linear dependence in this paper.

We now describe the basic idea of cooperative pipelined regeneration. Because random linear codes and MSR codes, on which we build the cooperative pipelined regeneration, regenerate data in a functional manner, cooperative pipelined regeneration we propose in this paper should also be categorized as functional regeneration. As long as a certain number of r coded blocks, which may be in storage nodes or apprentices, are lost ($r \geq 1$), a round of cooperative pipelined regeneration with r newcomers will be triggered. If $r = 1$, cooperative regeneration will be degraded to be independent. We use ν providers during cooperative pipelined regeneration, $\nu \geq 1$. Newcomers receive data (coded blocks with random linear codes or coded segments with MSR codes) from other participating nodes and encode received data into partially regenerated coded blocks. Thus, after this round of regeneration, newcomers will be partially regenerated and become apprentices. During the next round of cooperative pipelined regeneration, apprentices and r new newcomers receive data from other participating nodes with another set of ν providers. The value of ν is selected such that r apprentices will be fully regenerated, *i.e.,* they contact k providers with random linear codes or $N-1$ participating nodes with MSR codes, and "graduate" to become storage nodes after $\alpha + 1$ rounds ($\alpha \geq 1$) of cooperative pipelined regeneration (including the one as newcomers). Thus, there are always αr apprentices in the system. Throughout cooperative pipelined regeneration, r newcomers appear and r apprentices become storage nodes, keeping the number of storage nodes stable in the system.

It is required that there is no provider appearing more than once within $\alpha + 1$ consecutive rounds of cooperative pipelined regeneration, and thus newcomers and apprentices can always obtain innovative data during each round of regeneration. The distributed storage system can enforce this requirement by storing the identifiers of participating nodes in the latest α rounds of regeneration as the metadata of the corresponding file. If an apprentice A has received data from γ other participating nodes, the rank of its coded block B' is $\text{rank}(B') = \gamma$. When $\text{rank}(B') \geq k$ with random linear codes or $\text{rank}(B') \geq N-1$ with MSR codes, the block has a full rank and thus it can recover the original data with coded blocks in other $k-1$ storage nodes, preserving the recoverability property. We will show cooperative regeneration processes for random linear codes and MSR codes and analyze their respective performance in Sec. IV and Sec. V.

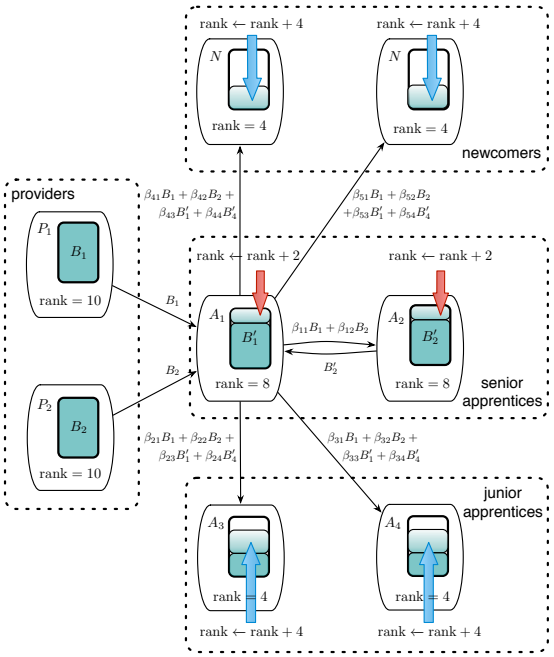

Fig. 3. An example of cooperative pipelined regeneration process ($r = \alpha = \nu = 2$) with random linear codes, including 2 newcomers, 2 junior apprentices, 2 senior apprentices and 2 providers. β_{ij} represents random coefficients of coded blocks conveyed through the corresponding connection.

IV. Regeneration with Random Linear Codes

A. Transmission

There are three types of nodes participating in the cooperative regeneration process with random linear codes, including r newcomers $(N_1, \ldots, N_r)$, ν providers $(P_1, \ldots, P_\nu)$, and αr apprentices $(A_1, \ldots, A_{\alpha r})$. The coded block of A_i is referred to as B'_i, $i = 1, \ldots, \alpha r$. Without loss of generality, we assume that $\text{rank}(B'_1) \geq \text{rank}(B'_2) \geq \ldots \geq \text{rank}(B'_{\alpha r})$. Among the αr apprentices, $A_1, \ldots, A_r$ are also denoted as *senior apprentices*, and one of the r senior apprentices is selected as the *root*. Other apprentices, $A_{r+1}, \ldots, A_{\alpha r}$, are referred to as *junior apprentices*.

During regeneration, all providers send their coded blocks to the root. The root updates its coded block by encoding these coded blocks with its own coded block. Meanwhile, it sends one random linear combination of received blocks to each of the other senior apprentices. These senior apprentices also encode their own coded blocks with received blocks. Thus, the rank of the coded block in each senior apprentice will be increased by ν. The values of ν and α are set such that all senior apprentices can be fully regenerated. All non-root senior apprentices return their coded blocks to the root. Now the root owns $\nu + r$ fully regenerated coded blocks (ν from providers, one from itself, and $r - 1$ from non-root senior apprentices), and it sends one linear combination of these $\nu + r$ coded blocks to each of the junior apprentices and the newcomers. All junior apprentices encode their coded blocks with received blocks, increasing the ranks of their coded blocks by $\nu + r$, and all newcomers store the received blocks, of rank $\nu + r$.

Fig. 3 illustrates a cooperative pipelined regeneration process with random linear codes. The value of k is 10 in this example. However, there are only 8 nodes participating during regeneration, including 2 newcomers ($r = 2$), 4 apprentices ($\alpha r = 4$, *i.e.,* $\alpha = 2$) and 2 providers ($\nu = 2$). In each round of cooperative pipelined regeneration, newcomers can get coded blocks of rank 4, junior apprentices increase the ranks of their coded blocks by 4, and senior apprentices increase the ranks of their coded blocks by 2. Thus, every newcomer will have a coded block with a rank of 10, and thus become fully regenerated after 3 rounds of cooperative pipelined regeneration. However, since the size of each coded block is $\frac{M}{10}$ bits ($k = 10$) and only 8 coded blocks are transferred during regeneration, the bandwidth consumed can be reduced from $\frac{6}{5}M$ bits, which is required to regenerate two coded blocks in batches during conventional regeneration (see Fig. 2(a)), to $\frac{4}{5}M$ bits, saving the consumption of bandwidth by 33%.

In $\alpha + 1$ consecutive rounds of cooperative pipelined regeneration, a storage node is required to act as a provider at most once, such that apprentices and newcomers can always obtain innovative coded blocks. In case any apprentices fail to work before they are fully regenerated, we use storage nodes that have not been used in recent $\alpha + 1$ rounds of cooperative pipelined regeneration to replace failed apprentices, when these apprentices would have acted as senior apprentices. Specifically, if a senior apprentice fails to work, we use a new storage node to replace it in the next round of cooperative pipelined regeneration. If a junior apprentice that has accomplished τ rounds of regeneration $(1 \leq \tau < \alpha)$ fails, one additional storage node should be selected to replace it in the next $(\alpha + 1 - \tau)^{\text{th}}$ round of cooperative pipelined regeneration, when it would have become a senior apprentice. Thus, the number of participating nodes does not increase, and newcomers and junior apprentices can still receive coded blocks of rank $\nu + r$. Moreover, the storage node selected to replace the failed apprentice just receives, encodes and sends out data, but does not need to update its own coded block, since its block has already been fully regenerated before. This storage node should not be used again in recent $\alpha + 1$ rounds of pipelined regeneration as a provider, or to replace a failed apprentice.

B. Number of participating nodes

The number of participating nodes depends on the values of ν and α. We first show feasible values of ν and α.

Since it is required that senior apprentices can be fully regenerated after each round of cooperative pipelined regeneration, the ranks of their coded blocks should be no less than k. During cooperative pipelined regeneration, a node can obtain a coded block with a rank of $\nu + r$ as a newcomer, and increase the rank of its coded block by $\nu + r$ as a junior apprentice and by ν as a senior apprentice. Therefore, after $\alpha + 1$ rounds of cooperative pipelined regeneration, the rank of the coded block in a senior apprentice is $\alpha(\nu + r) + \nu$. Since random linear codes encode at least k coded blocks to regenerate one coded block, the senior apprentice becomes fully regenerated

if and only if

$$\alpha(\nu+r)+\nu \geq k. \tag{1}$$

According to the analysis above, we obtain the following theorem.

Theorem 1: If $\alpha(\nu+r)+\nu \geq k$, the values of ν and α are feasible, *i.e.,* coded blocks in senior apprentices can always be fully regenerated after each round of cooperative pipelined regeneration.

We now investigate how to minimize $\nu+(\alpha+1)r$, the number of participating nodes, with the constraint of (1). Let $l=\nu+(\alpha+1)r$, and then substitute ν with $l-(\alpha+1)r$ in (1):

$$\begin{aligned} l &\geq (\alpha+1)r+\frac{k-\alpha r}{\alpha+1} && (2)\\ &= (\alpha+1)r+\frac{k+r}{\alpha+1}-r && (3)\\ &\geq 2\sqrt{r(k+r)}-r. && (4)\end{aligned}$$

The equality in (4) is achieved if and only if $\alpha=\sqrt{1+\frac{k}{r}}-1$. Then the minimum value of l is $l_{\min}=2\sqrt{r(k+r)}-r$, and $\nu=l_{\min}-(\alpha+1)r=\sqrt{r(k+r)}-r$. Note that $\alpha<1$ when $r>\frac{k}{3}$. Since cooperative pipelined regeneration requires that $\alpha \geq 1$, the regeneration process will be degraded to conventional regeneration when $r>\frac{k}{3}$. To support cooperative pipelined regeneration, we require that $r \leq \frac{k}{3}$. We believe that the feasible range of r is large enough in practical distributed storage systems where k is set large enough to improve the resilience to data losses (*e.g.*, $k=100$ in Wuala [23]).

Therefore, we obtain the following theorem.

Theorem 2: To regenerate r coded blocks with random linear codes ($r \leq \frac{k}{3}$), the minimum number of participating nodes during cooperative pipelined regeneration is $l_{\min}=2\sqrt{r(k+r)}-r$, while $\alpha=\sqrt{1+\frac{k}{r}}-1$ and $\nu=\sqrt{r(k+r)}-r$.

During conventional regeneration, we need at least k providers to regenerate coded blocks while maintaining the recoverability property. Thus, when there are r newcomers, there should be at least $k+r$ participating nodes. However, we can easily get

$$l_{\min}=2\sqrt{r(k+r)}-r \leq r+(k+r)-r=k+r. \tag{5}$$

The equality in (5) is achieved only when $r=k+r$, which is impossible in practical systems since $k>0$. Therefore, cooperative pipelined regeneration is always able to consume fewer participating nodes than conventional regeneration.

Since the values of ν and α should always be integers, in practice we can set the values of α and ν as $\left\lfloor\sqrt{1+\frac{k}{r}}\right\rfloor-1$ and $\left\lceil\frac{k-\alpha r}{\alpha+1}\right\rceil$, respectively.

Fig. 4 shows the practical numbers of participating nodes in the processes of conventional and cooperative pipelined regeneration when $k=60, 80$, and 100. When $r \leq \frac{k}{3}$, cooperative pipelined regeneration is able to save up to 81.1% participating nodes. We notice that though one coded block needs at least 100 providers to regenerate when $k=100$, the number of participating nodes utilized during cooperative pipelined regeneration never exceed 100, even though there are more than 30 newcomers.

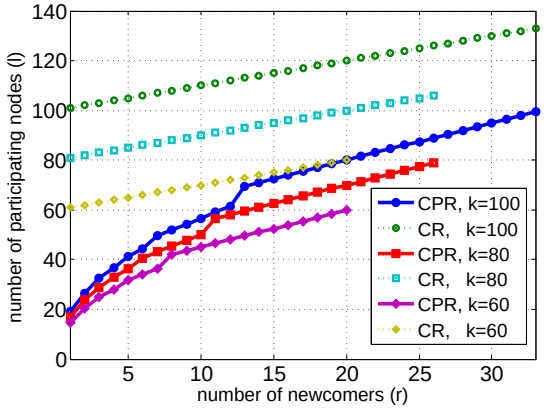

Fig. 4. A comparison of the respective number of participating nodes in the processes of cooperative pipelined regeneration (CPR) and conventional regeneration (CR), when $k=60, 80$, and 100.

C. Bandwidth consumption

Now we analyze the consumption of bandwidth during cooperative pipelined regeneration with random linear codes. Throughout the process of cooperative pipelined regeneration, there are ν connections from ν providers to the root, $2(r-1)$ connections between the root and $r-1$ non-root senior apprentices, and αr connections from the root to junior apprentices and newcomers. Thus, there are a total of $\nu+2(r-1)+\alpha r$ connections. One coded block is conveyed through each connection, and thus the bandwidth consumed is $\frac{M}{k}\cdot(\nu+2(r-1)+\alpha r)$ bits.

By (4), we can get

$$\begin{aligned} &\nu+2(r-1)+\alpha r && (6)\\ \geq\ & l_{\min}+r-2 && (7)\\ =\ & 2(\sqrt{r(k+r)}-1). && (8)\end{aligned}$$

Thus, the lower bound of the bandwidth consumption is $\frac{M}{k}\cdot 2(\sqrt{k(k+r)}-1)$ bits. The equality in (7) is achieved when $\alpha=\sqrt{1+\frac{k}{r}}-1$ and $\nu=\sqrt{r(k+r)}-r$.

During conventional regeneration, there are $k+r-1$ connections that also convey coded blocks of size $\frac{M}{k}$ bits. As r is required to be no more than $\frac{k}{3}$, Theorem 3 compares the bandwidth consumption between cooperative pipelined regeneration and conventional regeneration.

Theorem 3: When $r \leq \frac{k}{3}$, cooperative pipelined regeneration consumes less bandwidth than conventional regeneration.

Proof: When $r \leq \frac{k}{3}$, $2\sqrt{r(k+r)} \leq k+r$. Since $2(\sqrt{r(k+r)}-1) \leq k+r-2 < k+r-1$, cooperative pipelined regeneration that transfers $2(\sqrt{r(k+r)}-1)$ coded blocks consumes less bandwidth than conventional regeneration that transfers $k+r-1$ coded blocks. ■

Fig. 5 illustrates the bandwidth consumption of cooperative pipelined regeneration and conventional regeneration with

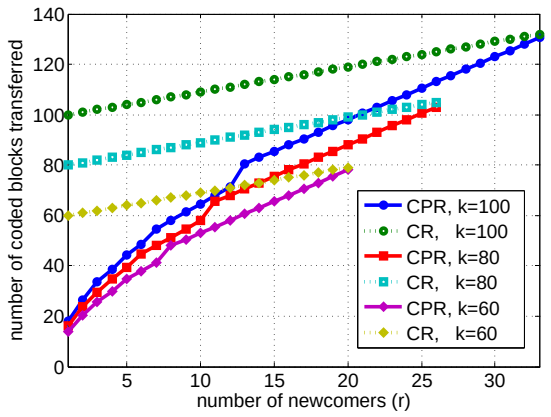

Fig. 5. Comparison of coded blocks transferred in the processes of cooperative pipelined regeneration (CPR) and conventional regeneration (CR), $k = 60, 80, 100$.

practical values of ν and α, in terms of the number of coded blocks conveyed during regeneration. When the condition $r \leq \frac{k}{3}$ is satisfied, cooperative pipelined regeneration can always incur less consumption of bandwidth.

D. Storage overhead

During conventional regeneration with random linear codes, at least k providers must be contacted. If the distributed storage system is designed to tolerate r failed storage nodes, there must be at least $k + r$ storage nodes in the system. We show that cooperative pipelined regeneration requires the system to not only maintain at least $k + r$ storage nodes, but also spend additional storage space for apprentices. However, this additional storage overhead is marginal.

Theorem 4: To support cooperative pipelined regeneration, the system should maintain at least $k + r$ storage nodes and $\sqrt{r(k+r)} - r$ apprentices.

Proof: By Theorem 2, there must be at least $\alpha_{\min} r = \sqrt{r(k+r)} - r$ apprentices in the system.

Since a coded block can be regenerated by contacting k providers, the amount of redundancy is enough if and only if ν providers that have not appeared in the previous α rounds of cooperative pipelined regeneration can always be found.

A round of cooperative pipelined regeneration is triggered when r node failures have been detected. If a failed node is a storage node that has not been used in recent $\alpha + 1$ rounds of cooperative pipelined regeneration, the required number of available storage nodes remains to be ν. If a failed node is an available storage node, the required number of available storage nodes should be increased by 1. If a failed node is a senior apprentice, one more storage node will be required to replace this apprentice during regeneration. If a failed node is a junior apprentice that has accomplished τ rounds of regeneration ($1 \leq \tau \leq \alpha$), the required number of storage nodes remains the same. However, in the next $(1 + \alpha - \tau)^{\text{th}}$ rounds regeneration, two more available storage nodes may be required. Therefore, the total number of required available storage nodes remains the same during $\alpha + 1$ rounds of cooperative pipelined regeneration. Overall, since there are

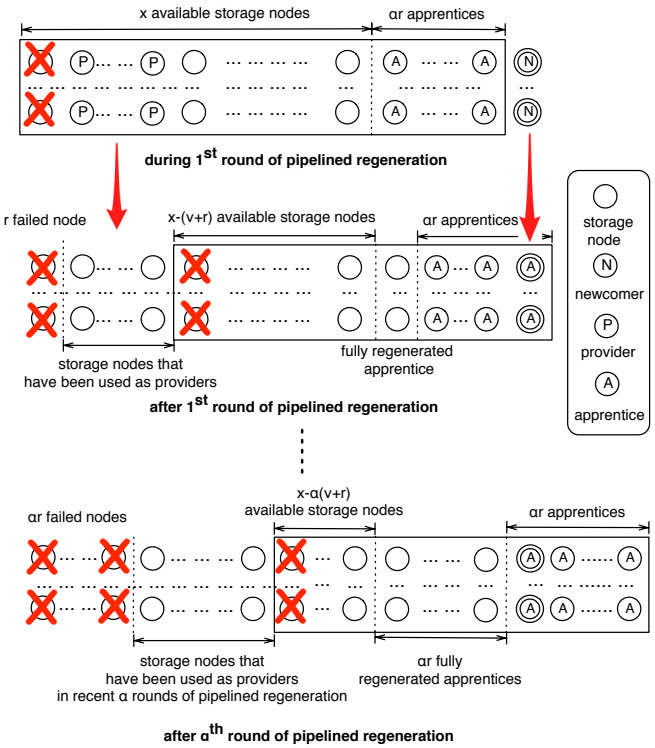

Fig. 6. The required number of available storage nodes in $\alpha+1$ consecutive rounds of cooperative pipelined regeneration, where failed nodes are all available storage nodes.

r failed nodes before regeneration, the average number of required available storage nodes is at most $\nu + r$.

We suppose that each round of cooperative pipelined regeneration requires $\nu + r$ available storage nodes. Without loss of generality, we assume that failed nodes are all available storage nodes. As shown in Fig. 6, $\nu + r$ available storage nodes become unavailable to the double-circled apprentices after each round of cooperative pipelined regeneration. Fully regenerated senior apprentices are also unavailable because they have also sent their coded blocks to the double-circled apprentices. Assume that there are x storage nodes available to the double-circled newcomer before its first round of regeneration. There are $x - \alpha(\nu + r)$ nodes available after α rounds of regeneration. Before the $(\alpha + 1)^{\text{th}}$ round of regeneration, $\nu + r$ additional available storage nodes are required, *i.e.,*

$$x - \alpha(\nu + r) \geq \nu + r. \tag{9}$$

By (1), $x \geq \alpha(\nu + r) + \nu + r \geq k + r$. Therefore, the system should maintain at least $k + r$ storage nodes. ■

Theorem 4 points out that apart from apprentices that are required by cooperative pipelined regeneration, the distributed storage system should still maintain at least $k + r$ storage nodes, since storage nodes can be used as providers at most once in $\alpha + 1$ consecutive rounds of cooperative pipelined regeneration. However, because $r \leq \frac{k}{3}$ during cooperative pipelined regeneration, it is easy to prove that $\sqrt{r(k+r)} - r \geq \frac{k}{3}$. Thus, the additional storage space required by apprentices will be no more than $\frac{1}{3}$ of the original file. Considering that

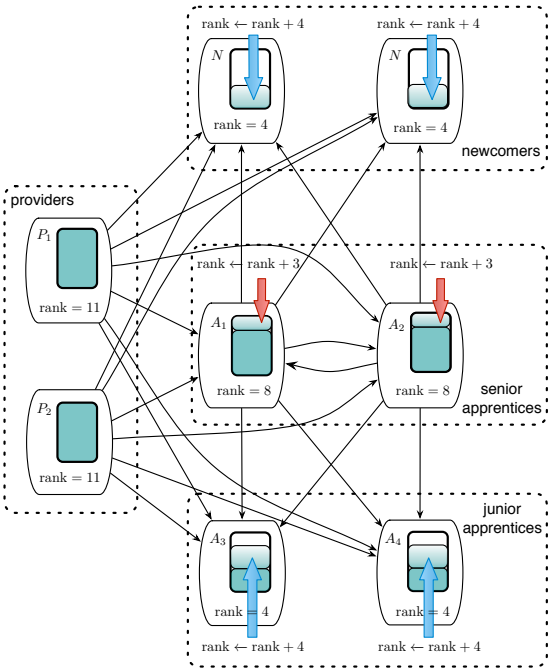

Fig. 7. An example of the cooperative pipelined regeneration process ($r = \alpha = \nu = 2$) with MSR codes ($k = 3, N = 12$), including 2 newcomers, 2 junior apprentices, 2 senior apprentices and 2 providers. Each connection conveys one coded segment.

in practical distributed storage systems, redundant data stored is usually more than three times of the original file (3x in Google File System by default [24] and 4.17x in Wuala [23]), this storage overhead is marginal.

V. Regeneration with Regenerating Codes

A. Transmission

Similar to cooperative pipelined regeneration with random linear codes, there are also three types of nodes in the cooperative pipelined regeneration process with Minimum-Storage Regenerating (MSR) codes: r newcomers $(N_1, \ldots, N_r)$, ν providers $(P_1, \ldots, P_\nu)$ and αr apprentices $(A_1, \ldots, A_{\alpha r})$. We refer to the coded block of A_i as B'_i, $i = 1, \ldots, \alpha r$, and assume that $\text{rank}(B'_1) \geq \text{rank}(B'_2) \geq \ldots \geq \text{rank}(B'_{\alpha r})$. $A_1, \ldots, A_r$ are referred to as senior apprentices and other apprentices are referred to as junior apprentices.

During regeneration, all providers send one coded segment, which is a random linear combination of coded segments in their coded blocks, to each of the r newcomers and the αr apprentices. Each senior apprentice then sends $(\alpha + 1)r - 1$ random linear combinations of coded segments it has received from providers in recent $\alpha + 1$ rounds of regeneration to all other apprentices and newcomers. Throughout this process, senior apprentices get $\nu + r - 1$ coded segments, and junior apprentices and newcomers receive $\nu + r$ coded segments. Since each coded block is supposed to contain $N - k$ coded segments in our model, as described in Sec. III, each senior apprentice encodes all coded segments they have received so far into $N - k$ coded segments, and groups them as one coded block.

Fig. 7 shows an example of cooperative pipelined regeneration ($r = \alpha = \nu = 2$). We suppose that 3 coded blocks can recover the original file ($k = 3$), and each fully regenerated coded block contains 9 coded blocks ($N = 12$). Since there are 22 connections that convey a total of 22 coded segments, only $\frac{M}{3} \cdot \frac{22}{9} M$ bits are transferred. As shown in Sec. II, 12 nodes are required to regenerate two coded blocks with the same bandwidth consumption during conventional regeneration. Thus, cooperative pipelined regeneration is able to save 33.3% of participating nodes by requiring only 8 nodes, while consuming the same amount of bandwidth. We will show later that the bandwidth consumption and participating nodes can both be saved by increasing the value of N.

Similar to cooperative pipelined regeneration with random linear codes, we use storage nodes that have not been used in recent $\alpha+1$ rounds of regeneration to replace the role of failed apprentices during regeneration. Like other storage nodes used as providers, they can not be used again in $\alpha + 1$ consecutive rounds of regeneration.

B. Number of participating nodes

We first analyze the feasible values of ν and α. We show that feasible values of ν and α can guarantee that $\alpha+1$ rounds of cooperative pipelined regeneration can be mapped to one round of conventional regeneration with N participating nodes.

As shown in Sec. II, in the conventional regeneration process with MSR codes, to regenerate r coded blocks [18], all r newcomers should receive coded segments from all providers, and then each newcomer sends linear combinations of coded segments it has received to all other newcomers. Newcomers then encode received segments to $N - k$ coded segments, and group them into one coded block that is able to preserve the recoverability property.

Given r senior apprentice $A_1, \ldots, A_r$, which are fully regenerated after one round of cooperative pipelined regeneration, they have received $(\alpha + 1)\nu$ coded segments from $(\alpha + 1)\nu$ providers $P_1, \ldots, P_{(\alpha+1)\nu}$ in the recent $\alpha + 1$ rounds of regeneration. There are also αr senior apprentices, $A_{r+1}, \ldots, A_{(\alpha+1)r}$, in the previous α rounds of regeneration. We map these $\alpha + 1$ rounds of cooperative pipelined regeneration into one conventional regeneration by mapping $A_1, \ldots, A_{(\alpha+1)r}$ to $(\alpha + 1)r$ newcomers $(A'_1, \ldots, A'_{(\alpha+1)r})$ and $P_1, \ldots, P_{(\alpha+1)\nu}$ to $(\alpha+1)r$ providers $(P'_1, \ldots, P'_{(\alpha+1)\nu})$. We also map the connection to the corresponding pair of nodes in the conventional regeneration process. Thus, A'_i $(i = 1, \ldots, r)$ receives coded segments from all providers $P'_1, \ldots, P'_{(\alpha+1)\nu}$ and all other newcomers.

Now we show that $A'_{r+1}, \ldots, A'_{(\alpha+1)r}$ also receive coded segments from $P'_1, \ldots, P'_{(\alpha+1)\nu}$ after mapping. Actually, $A_{r+1}, \ldots, A_{(\alpha+1)r}$ receive coded segments from some other providers that are not mapped into the conventional regeneration process. Assume that one such senior apprentice receives coded segments from providers $P_{i\nu+1}, \ldots, P_{(i+\alpha+1)\nu}, 1 < i \leq \alpha$. Since $(\alpha + 1)r \geq k$, coded segments stored in these providers can all be regarded as linear combinations of coded segments stored in $P_1, \ldots, P_{(\alpha+1)\nu}$. As all coded segments a

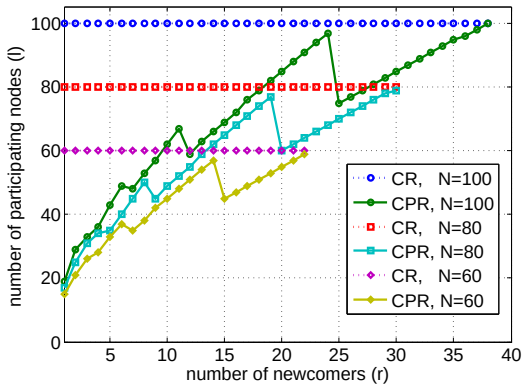

Fig. 8. Comparison of participating nodes in the processes of cooperative pipelined regeneration (CPR) and conventional regeneration (CR) with MSR codes, $N = 60, 80$, and 100.

newcomer receives are random linear combinations of coded segments stored in providers, it is functionally equivalent with the fact that, these coded segments received are random linear combinations of coded segments stored in $P_1, \ldots, P_{(\alpha+1)\nu}$. Therefore, coded blocks regenerated in $A'_1, \ldots, A'_r$ are regenerated equivalently with $(\alpha+1)\nu$ providers and $(\alpha+1)r$ newcomers in a conventional regeneration process, which requires N participating nodes and k providers to preserve the recoverability property. By the analysis above, we can easily obtain the following theorem.

Theorem 5: If $(\alpha+1)(\nu+r) \geq N$ and $(\alpha+1)\nu \geq k$, all senior apprentices can be fully regenerated after cooperative pipelined regeneration with MSR codes.

By the conditions required by Theorem 5, we analyze the values of ν and α that require the minimum number of participating nodes.

Theorem 6: To regenerate r coded block containing $N-k$ coded segments in one round of pipelined cooperative regeneration, the minimum number of participating nodes is $l_{\min} = 2\sqrt{Nr} - r$, when $\alpha = \sqrt{\frac{N}{r}} - 1$, $\nu = \sqrt{Nr} - r$, and $k \leq N - \sqrt{Nr}$.

Proof: By Theorem 5, $N \leq (\alpha+1)(\nu+r)$. Thus, $\nu \geq \frac{N}{\alpha+1} - r$. The total number of participating nodes during cooperative pipelined regeneration is

$$(\alpha+1)r + \nu \geq (\alpha+1)r + \frac{N}{\alpha+1} - r \geq 2\sqrt{Nr} - r. \quad (10)$$

Equalities in (10) are achieved when $\alpha = \sqrt{\frac{N}{r}} - 1$ and $\nu = \sqrt{Nr} - r$. Thus, k is required to be no more than $(\alpha+1)\nu = N - \sqrt{Nr}$. ■

Since the minimum number of participating nodes during cooperative pipelined regeneration with MSR codes is $2\sqrt{Nr} - r \leq (N+r) - r = N$, the number of participating nodes can always be saved unless $r = N$.

In practice, we can set the values of α and ν as $\left\lceil \sqrt{\frac{N}{r}} \right\rceil - 1$ and $\left\lceil \sqrt{Nr} \right\rceil - r$, respectively. In such settings, when $r < N$, α is always no less than 1, which guarantees a round of cooperative pipelined regeneration. Moreover, when $r < \frac{3-\sqrt{5}}{2} \cdot N$, we can always reduce the number of participating nodes. Similar to cooperative pipelined regeneration with random linear codes, we also believe that this range of r is large enough in practical distributed storage systems, where k is usually set to be large enough and N should be larger than k to save bandwidth consumption.

Fig. 8 illustrates the number of participating nodes with practical values of α and ν, when $N = 60, 80$, and 100, assuming that k is always feasible (*e.g.,* $k = 30$). We set the number of newcomers (r) no more than $\frac{3-\sqrt{5}}{2} \cdot N$ and find that the number of participating nodes can always be less than N. We notice that the saving in the number of participating nodes is more significant when the number of newcomers is small, since the values of α and ν both increase with the increasing of r, while conventional regeneration always requires N nodes participating no matter how many newcomers there are.

C. Bandwidth consumption

According to the transmission scheme during cooperative pipelined regeneration with MSR codes, there are $\nu(\alpha+1)r$ connections that convey coded segments from providers to apprentices and newcomers, and $r\left[(\alpha+1)r-1\right]$ connections from r senior apprentices to all other apprentices and newcomers. Thus, by Theorem 5, there are a total of $\nu(\alpha+1)r + r\left[(\alpha+1)r-1\right] \geq r(N-1)$ connections. Since each connection conveys one coded segment of size $\frac{M}{k(N-k)}$ bits, the minimum bandwidth consumption to regenerate r coded blocks is $\frac{M(N-1)}{k(N-k)} \cdot r$ bits, which is exactly the same amount of the bandwidth consumption required by conventional regeneration with r newcomers using MSR codes.

On the other hand, if we keep using the same number of participating nodes, we can store data encoded by MSR codes with a much higher N with the help of cooperative pipelined regeneration, further saving the consumption of bandwidth. Supposing that at most n nodes can participate during regeneration, by (10), the maximum N can be supported by cooperative pipelined regeneration is $N_{\max} = \frac{(n+r)^2}{4r}$, while N must be no more than n for conventional regeneration.

In practice, we can let $\alpha = \left\lfloor \frac{n-r}{2r} \right\rfloor$ and $\nu = \left\lfloor \frac{n-r}{2} \right\rfloor$

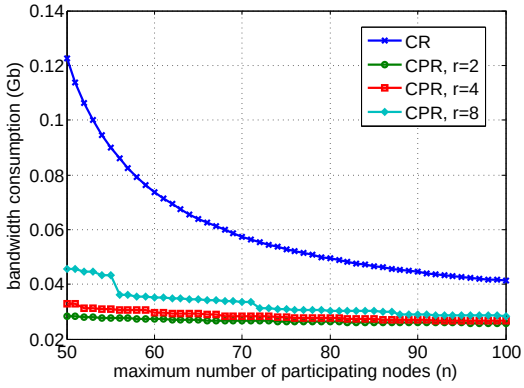

Fig. 9. Comparison of the bandwidth consumption in the processes of cooperative pipelined regeneration (CPR, $k = 40, r = 2, 4, 8$) and conventional regeneration (CR). The size of the original file is 1 Gb.

where $r < n$. Thus, the maximum N that can be supported by pipelined regeneration is $(\alpha+1)(\nu+r) \leq \frac{(n+r)^2}{4r}$. By storing data encoded by MSR codes with a large N, we can further reduce the bandwidth consumption during regeneration significantly. For example, we assume the size of the original file is 1 Gb. When $k = 40$ and $r = 2$, the bandwidth consumption can be saved by $38\% - 77\%$, as shown in Fig. 9. The bandwidth consumption of conventional regeneration does not change with r. During cooperative pipelined regeneration, the bandwidth consumption will increase slightly with r. However, the saving of the bandwidth consumption remains significant. Moreover, to save both participating nodes and the consumption of bandwidth, the system can select the value of N between n and $\lfloor \frac{n+r}{2r} \rfloor \cdot \lfloor \frac{n+r}{2} \rfloor$.

D. Storage overhead

We use the same method in Theorem 4 to analyze the storage overhead brought by cooperative pipelined regeneration with MSR codes. Assume that at least x storage nodes must be maintained in the system. Since no storage node can be used twice in $\alpha+1$ consecutive rounds of regeneration, (9) still holds. By Theorem 5, $x \geq (\alpha+1)(\nu+r) \geq N$. In other words, at least N storage nodes must be maintained in the system, which is the same number required by conventional regeneration with MSR codes.

Though cooperative pipelined regeneration with MSR codes does not require more storage nodes than conventional regeneration, there are also αr apprentices in the system, incurring additional storage overhead. Since the minimum number of participating nodes is achieved when $\alpha r = \sqrt{Nr} - r$ by Theorem 6, we obtain the following theorem.

Theorem 7: To support cooperative pipelined regeneration with MSR codes, the distributed storage system should maintain at least N storage nodes and $\sqrt{Nr} - r$ apprentices.

Since $\sqrt{Nr} - r \leq \frac{N}{4}$, the storage overhead brought by apprentices is marginal in practical distributed storage systems.

VI. Conclusion

In this paper, we study the process of data regeneration in failed storage nodes in distributed storage systems. The highlight of this paper is to propose a new *cooperative pipelined* regeneration mechanism, which depends on both node cooperation and pipelined regeneration to significantly reduce the number of participating nodes and the consumption of bandwidth during regeneration, as compared to existing schemes in the literature. We show that cooperative pipelined regeneration can maintain the optimal integrity of data by supporting both random linear codes and minimum-storage regenerating codes. Though cooperative pipelined regeneration incurs additional storage overhead in the system, we show that such overhead is marginal in practical distributed storage systems.

References

[1] I. Reed and G. Solomon, "Polynomial Codes over Certain Finite Fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[2] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network Information Flow," *IEEE Trans. Inform. Theory*, vol. 46, pp. 1204–1216, 2000.

[3] T. Ho, R. Koetter, M. Medard, D. Karger, and M. Effros, "The Benefits of Coding over Routing in a Randomized Setting," in *Proc. IEEE International Symp. Inform. Theory (ISIT)*, 2003, pp. 442–442.

[4] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, "Network Coding for Distributed Storage Systems," *IEEE Tran. Inform. Theory*, vol. 56, no. 9, Sept. 2010.

[5] A. Duminuco and E. Biersack, "A Practical Study of Regenerating Codes for Peer-to-Peer Backup Systems," in *Proc. 29th IEEE International Conference on Distributed Computing Systems*, 2009, pp. 376–384.

[6] R. Bhagwan, K. Tati, Y.-C. Cheng, S. Savage, and G. M. Voelker, "Total Recall: System Support for Automated Availability Management," in *Proc. USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2004.

[7] Y. Hu, Y. Xu, X. Wang, C. Zhan, and P. Li, "Cooperative Recovery of Distributed Storage from Multiple Losses with Network Coding," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 2, pp. 268–276, Feb. 2010.

[8] F. Oggier and A. Datta, "Self-repairing Homomorphic Codes for Distributed Storage Systems," in *Proc. IEEE INFOCOM*, Apr. 2011.

[9] D. S. Papailiopoulos, J. Luo, A. G. Dimakis, C. Huang, and J. Li, "Simple Regenerating Codes: Network Coding for Cloud Storage," in *Proc. IEEE INFOCOM*, 2012.

[10] H. Weatherspoon and J. Kubiatowicz, "Erasure Coding vs. Replication: A Quantitative Comparison," *Peer-to-Peer Systems*, vol. 2429/2002, pp. 328–337, 2002.

[11] S. Acedanski, S. Deb, M. Médard, and R. Koetter, "How Good is Random Linear Coding based Distributed Networked Storage?" in *Proc. 1st Workshop on Network Coding, Theory, and Applications (NetCod)*, 2005.

[12] Y. Wu, A. Dimakis, and K. Ramchandran, "Deterministic Regenerating Codes for Distributed Storage," in *Proc. 45th Annual Allerton Conference on Communication, Control, and Computing*, 2007.

[13] 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. Inform. Theory*, vol. 57, no. 8, pp. 5227–5239, 2011.

[14] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran, "Interference Alignment in Regenerating Codes for Distributed Storage: Necessity and Code Constructions," *IEEE Trans. Inform. Theory*, vol. 58, no. 4, pp. 2134–2158, 2012.

[15] V. R. Cadambe, S. A. Jafar, and H. Maleki, "Distributed Data Storage with Minimum Storage Regenerating Codes - Exact and Functional Repair are Asymptotically Equally Efficient," *CoRR*, vol. abs/1004.4299, 2010.

[16] C. Suh and K. Ramchandran, "On the Existence of Optimal Exact-Repair MDS Codes for Distributed Storage," *CoRR*, vol. abs/1004.4663, 2010.

[17] K. W. Shum, "Cooperative Regenerating Codes for Distributed Storage Systems," in *Proc. IEEE International Conference on Communications (ICC)*, 2011.

[18] A.-M. Kermarrec, N. Le Scouarnec, and G. Straub, "Repairing Multiple Failures with Coordinated and Adaptive Regenerating Codes," in *Proc. IEEE International Symposium on Network Coding (NetCod)*, 2011.

[19] A. Duminuco and E. Biersack, "Hierarchical Codes: How to Make Erasure Codes Attractive for Peer-to-Peer Storage Systems," in *Proc. 8th International Conference on Peer-to-Peer Computing*, 2008, pp. 89–98.

[20] F. Oggier and A. Datta, "Self-Repairing Codes for Distributed Storage - A Projective Geometric Construction," *arXiv:1105.0379*, May 2011. [Online]. Available: http://arxiv.org/abs/1105.0379

[21] S. E. Rouayheb and K. Ramchandran, "Fractional Repetition Codes for Repair in Distributed Storage Systems," in *48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2010, pp. 1510–1517.

[22] J. Li, X. Wang, and B. Li, "Pipelined Regeneration with Regenerating Codes for Distributed Storage Systems," in *Proc. IEEE International Symposium on Network Coding (NetCod)*, 2011.

[23] D. Grolimund, "Wuala - A Distributed File System," *Google Tech Talk*. [Online]. Available: http://www.youtube.com/watch?v=3xKZ4KGkQY8

[24] S. Ghemawat, H. Gobioff, and S. Leung, "The Google file system," *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 29–43, 2003.