2016 IEEE 35th Symposium on Reliable Distributed Systems

ProCode: A Proactive Erasure Coding Scheme for Cloud Storage Systems

Peng Li, Jing Li, Rebecca J. Stones, Gang Wang✉, Zhongwei Li, Xiaoguang Liu✉
Nankai-Baidu Joint Lab, College of Computer and Control Engineering
Nankai University, Tianjin, China
Email: {lipeng, lijing, rebecca.stones82, wgzwp, lizhongwei, liuxg}@nbjl.nankai.edu.cn

***Abstract*—Common distributed storage systems use data replication to improve system reliability and maintain data availability, but at the cost of disk storage. In order to lower storage costs, data may instead be stored according to erasure codes, but this results in greater network and disk traffic when data blocks are reconstructed following an erasure. These methods are also passive, i.e., they only reconstruct data after failures occur.**

In this paper, we present a proactive erasure coding scheme (ProCode). We monitor the health of disks via drive failure prediction and automatically adjust the replication factor of data blocks on at-risk disks to ensure data safety. In this way, we achieve fast recovery after disk failures without significantly increasing the storage overhead. ProCode is implemented as an extension to HDFS-RAID used by Facebook.

Compared with replication storage and erasure coding, ProCode improves system reliability and availability. Specifically, experimental results show 2 or more orders of magnitude reduction in the average number of data loss events over a 10-year period, a 63% or greater drop in degraded read latency, and a 78% drop in recovery time.

I. Introduction

With modern disk technology, an individual disk failure might be rare. However, in a cloud data center with thousands of storage disks, we may frequently experience failures and even simultaneous failures [35]. For these systems, providing satisfactory data reliability is a significant challenge. Two traditional data storage methods are:

Data replication: Data is simply stored multiple times (typically 3) over different storage units on different nodes. This gives high reliability in practice, but incurs significant storage costs, which is problematic particularly for large-scale systems.

Erasure coding: Data is stored according to an erasure code. This method requires less disk space than replication, but in case of disk failures, the computation and communication overhead is increased during the reconstruction process (possibly by a factor of 10).

Erasure codes have outstanding performance at providing higher reliability at significantly lower storage costs when used in large distributed storage systems. Recently much research has been carried out focusing on how to reduce the high degraded read latency and shorten the long reconstruction time, both in theory and practice [12], [16], [23], [30], [34]. Some researchers have employed new erasure codes to optimize the tradeoffs. For example, Huang et al. presented LRC [12], which divides the parity blocks into local parity blocks, which reduces the single failure recovery cost, and global parity blocks, to provide high reliability, and has been successfully applied in the Microsoft Windows Azure storage system. Rashmi et al. presented Hitchhiker [23], which "rides" on top of RS codes and reduces both network traffic and disk IO during reconstruction of missing or otherwise unavailable data with no additional storage. There is also research into the optimization of encoding and decoding [16], [30], [34], which reduces network transfer loads without increasing storage.

Some two-phase techniques [1], [6], [31], [33], which are mostly inspired by tiering RAID architectures, have been put into practice. AutoRAID [31] provides a two-level storage hierarchy within the storage controller which automatically and transparently migrates data blocks between different RAID levels as access patterns change. It adopts two copies for active data and requires low storage overhead due to the use of RAID 5 protection for inactive data. DiskReduce [6] and Facebook's HDFS-RAID [1] go a little further and employ two-phase techniques, asynchronously migrating data from replication to erasure coding if it has not been recently modified. HACFS [33] extends this scheme further by splitting the erasure-coded storage tier into two parts to optimize both storage overhead and recovery performance. This hybrid method results in lower reconstruction cost than both LRC and Google ColossusFS [7].

Only a few researchers focus on proactive disk warning, i.e., predicting drive failures before they actually occur. Aiming to improve prediction accuracy, some statistical and machine learning methods are used [13], [17], [20], [29] to build hard drive failure prediction models based on SMART attributes [3]. For example, the classification tree prediction model predicted over 95% of failures at a false alarm rate (FAR) under 0.1% [17]. A threshold-based predictor, PLATE [19], which monitors the health of each drive by tracking the number of reallocated sectors, is capable of detecting up to 65% of impending whole-disk failures with up to 2.5% FAR.

In the context of storage systems, failure prediction by itself is not enough; we need to combine the prediction results with disk pre-warning methods to improve system reliability and availability. A few research has been carried out to apply failure prediction models to storage systems. E.g., RAIDShield [19] proposed PLATE to predict the single drive failure, and ARMOR as well to estimate the health status of RAID group using joint probability.

In this paper, we present a *proactive erasure coding scheme* (ProCode) for distributed storage systems that combines the storage efficiency of erasure coding with the low recovery costs of simple data replication by utilizing drive failure prediction. We implement a prototype as an extension to HDFS-RAID.

1060-9857/16 $31.00 © 2016 IEEE
DOI 10.1109/SRDS.2016.37

Traditional two-phase storage systems generally adjust data redundancy according to file access patterns and give the "hot" (frequently accessed) data higher redundancy. For example, HDFS-RAID [1] focuses on the tradeoff of read performance and storage space, replicating hot data and erasure coding cold data. ProCode is designed from a different angle: distinct from HDFS-RAID-like systems, ProCode instead stores at-risk data (blocks with low "block health degree") with replication while other data is erasure coded. The aim is to reduce the recovery cost in erasure coding when failure happens. ProCode also adopts a fine-grained replication scheme where blocks are duplicated, rather than the entire stripe or file. By using this scheme, the units from the same stripe can have different numbers of replicas corresponding to the units' individual health degrees.

Replication systems achieve better read performance in read-intensive workloads but perform worse in write-intensive workloads than erasure coding system. ProCode offers a balance, performing the same as erasure coding for write-intensive workloads. For read-intensive workloads, while the 3-way replication system performs the best, the methods similar to [2], [15], [33], who give "hot" data higher redundancy can help ProCode to reduce the performance gap (between replication and erasure coding).

One stripe example for ProCode(6,2) is shown in Figure 1. Blocks are either predicted to fail (i.e., resident on a drive that is predicted to fail) or are healthy (i.e., resident on a healthy drive). This stripe has six unique data blocks and two unique code blocks (parity blocks). Some blocks like A_{21} are healthy blocks so one copy of each of them is considered enough to ensure their reliability. However, A_{11}, A_{31}, A_{32}, A_{51}, and P_{11} are predicted to fail, so we use a healthy replica to ensure their content's safety. In this way, we ensure each block has a healthy copy.

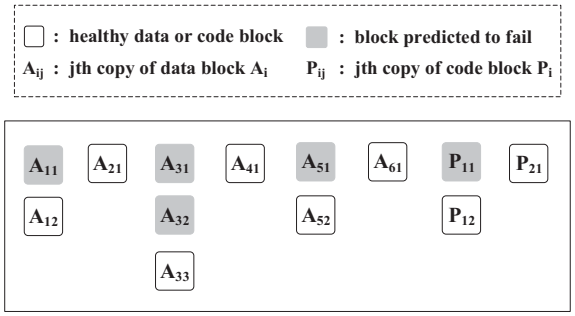


Fig. 1: Illustrating one stripe for ProCode(6,2) with some blocks residing on disks which are predicted to fail.

ProCode aims to eliminate most resource-consuming block recovery operations in erasure coding by using hard drive failure predictors with high prediction accuracy [17], [19]. Moreover, as previous work [17], [20] shows, disk failure prediction can provide sufficient time for block replication, thereby enabling the bandwidth for data migrating to be throttled, reducing the impact on the system during migration.

The remainder of the paper is organized as follows. We further motivate our research in Section II and describe ProCode in detail in Section III. The results of experimental testing are given in Section IV. We put this work into context and consider some possible extensions in Section V.

II. Background

A. Passive fault tolerance

At present, replication and erasure coding are two practical passive fault-tolerant methods typically trade off storage overhead against recovery overhead, which have been used in [1], [7], [8], [28]. There is also some research focusing on the optimization of replication or erasure coding to reduce recovery costs and/or storage overhead. For example, Cidon et al. [5] proposed Copyset, which distributes replicas in limited node groups to reduce the probability of data loss and recovery costs, but this unavoidably incurs a high storage overhead due to the replication strategy. Local Reconstruction Codes [12], Regenerating codes [21] and Weaver codes [10] also both reach a balance between recovery costs and storage utilization. There are also some researchers [16], [24], [27], [34] focusing on the optimization of encoding or decoding processes to minimize disk IO and network overhead such as Diagonal Optimal Recovery (RDOR) [34], which can perform single disk failure recovery efficiently.

Table I tabulates the properties of some popular storage systems and their recovery costs and storage overhead. Recovery costs (third column) are the average-case number of blocks needed to recover a lost block and storage overhead (fourth column) represents the proportion of storage required for a given storage method vs. the storage required for a single copy of the raw data. Google's ColossusFS and Facebook's HDFS-RAID use two different Reed-Solomon codes whereas Microsoft Azure encodes 12 data blocks into 2 global and 2 local code blocks.

System	Code	Recovery	Storage
HDFS [28]	—	1	3
Facebook HDFS-RAID [1]	$\mathrm{RS}(10,4)$	10	1.4
Google ColossusFS [7]	$\mathrm{RS}(6,3)$	6	1.5
Microsoft Azure [12]	$\mathrm{LRC}(12,2,2)$	6	1.3

TABLE I: The recovery cost and storage overhead of some popular storage systems.

B. Proactive fault tolerance

Self-Monitoring, Analysis, and Reporting Technology (S-MART) is implemented on most modern hard disks [3]. SMART monitors and compares disk attributes with preset thresholds, and issues warnings in the event that some attribute exceeds its threshold. SMART allows administrators to take appropriate actions in advance before disk failures actually occur, i.e., proactive fault tolerance, which fundamentally improves system reliability. However, to avoid false alarms, disk manufacturers set conservative thresholds to keep the false

alarm rate low, which comes at the expense of the failure detection rate [20].

SMART, by itself, can not reach a satisfactory prediction performance. To improve prediction accuracy, many statistical and machine learning methods have been proposed to build prediction models based on SMART attributes [13], [17], [20], [29], some of which have achieved good prediction performance. Li et al. [17] proposed a Classification Tree (CT) model which predicts over 95% of failures at a false alarm rate (FAR) under 0.1% on a real-world dataset containing 25,792 drives; we incorporate it into ProCode.

In addition to disk failure prediction, a few studies focus on disk failure pre-warning mechanisms, which give time to backup at-risk data and replace at-risk disks before failure occurs, improving system reliability. Fatman [22] and IDO [32] simply replace at-risk disks by using prediction results without taking the availability of at-risk disks and the migration impact on the system performance into consideration; and SSM [14] proactively migrates data from soon-to-fail disks predicted by failure prediction models, which inspired this work. RAIDShield [19] is an active defense method for RAID storage systems which predicts the possibility of data loss in one code group. We incorporate a pre-warning mechanism into ProCode via the health degree model.

III. ProCode

In the proposed method, ProCode, the system initially uses an erasure code strategy for a low storage overhead. Blocks that are determined to be at risk (i.e., residing on a disk with poor disk health) are duplicated before failure occurs (i.e., proactive fault tolerance). We have the following simultaneous goals:

Goal 1: Achieve high storage efficiency, comparable to erasure coding and significantly better than 3-way replication system.

Goal 2: Provide superior failure recovery performance, significantly better than erasure coding, with higher reliability as well.

Goal 3: Maintain high data availability (low degraded read latency), in the same ballpark as the 3-way replication system.

A. Architecture and design

An overview of the architecture of ProCode is given in Figure 2. ProCode is implemented as an extension to HDFS-RAID, which has a master/slave architecture and does not support random write operations. There is only one master, called NameNode, which manages the file system metadata and controls client access while slave nodes, called DataNodes, store application data using small blocks. The RaidNode is responsible for erasure encoding/decoding and also activates the recovery of corrupted blocks.

DataNode predicts its associated drives' health statuses, which are reported to NameNode. NameNode is responsible for automatically adjusting the storage policy (e.g., increasing or decreasing block replication) accordingly. Thus, when failures occur, the system should have predicted it, and replicated

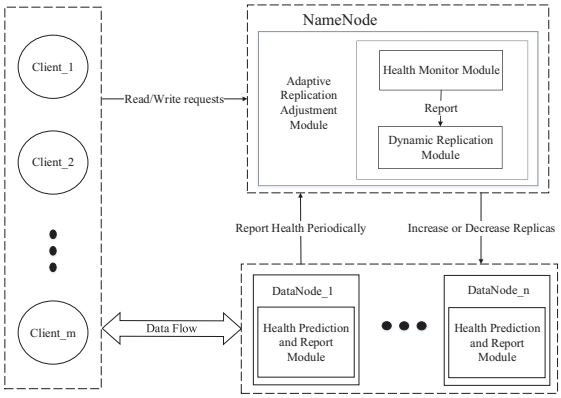


Fig. 2: Architecture and design of ProCode.

the data in response to the early warning. However, this depends on accurate failure predictions.

1) Health Prediction and Report Module: These modules monitor the health status of individual disks and report the prediction result to NameNode. For the drives that are protected by a local RAID group, we can use joint probability to predict the health of RAID group (as described in [19]). This module starts a daemon thread to collect SMART attributes of drives periodically and transfers the collected information to a trained disk failure predictor to predict the disk health after necessary pre-processing such as data standardization and feature selection. We use the classification tree method proposed by Li et al. [17] for disk failure prediction.

2) Adaptive Replication Adjustment Module: ProCode adjusts the redundancy of data blocks according to their health statuses.

a) Block Health: For each block, the Health Monitor Module in NameNode periodically collects its health reports from the DataNodes. We define a block's health by

$$H := \sum_i h_i \tag{1}$$

where $h_i \in \{0,1\}$ is the health status of the drive containing the i-th copy of the block, with 1 meaning healthy and 0 meaning predicted to fail. A drive's health status is determined using the CT method by Li et al. [17].

b) Health Monitor Module: The Health Monitor Module receives the drive health reports from DataNodes and updates the blocks' health accordingly. When a block's health changes, its replication level may also be changed, which is decided by the Dynamic Replication Module.

c) Dynamic Replication Module: ProCode extends erasure coding by dynamically replicating blocks according to their health determined from (1). We describe the details of transitioning states in this section. Files are initially stored compactly using erasure coding, and we subsequently increase or decrease (or leave unmodified) the number of copies of each block.

If $H = 0$ for a given block, this module will trigger a process to copy that block onto a healthy node, which will

increase the block's H-value. If $H \geq 1$, there is at least one copy of that block on a healthy disk, but to reduce storage costs, if $H \geq 2$, we delete a copy. However, to safeguard against noisy predictions we build in a time delay before deleting blocks. Drive failure prediction model gives the *time in advance* (TIA) before drive failures actually happen [17], which can be used as the time delay. TIA varies with the model updates and ProCode will adjust the time delay accordingly. A time delay exceeding the TIA will be a stronger safeguard, but will result in extra data storage. The optimal choice of time delay will vary according to the TIA and false alarm rate, and should be matched to the storage system.

When one or more disk warnings occur, a large number of blocks with $H = 0$ will need to be replicated. A priority-based method is used for increasing block replication, where blocks with the fewest copies are copied first. Deleting copied blocks is done in a First In First Out manner while for blocks which need to have a copy deleted, we prioritize unhealthy copies.

Previous studies [19], [22], [32] on disk fault prediction regard at-risk drives as failed, replacing them with healthy drives immediately. In contrast, ProCode retains at-risk drives still available in the system until failure actually occurs, and this can be superior in two ways: (a) There is a possibility of intermittent false alarms, which would ordinarily result in a disk being classed as "failed" and consequently disused and replaced. However, ProCode will instead continue to monitor its health, where it would be later reclassified as healthy, thereby avoiding replacing a healthy disk. (b) Drives that are about to fail are still capable of serving read requests for clients and replicating tasks before they are replaced or actually fail, improving data availability.

Disk warning events trigger the process of increasing replication, which results in greater storage costs than replacing disks immediately. In storage systems, most drives are healthy and only a small fraction are classed as at risk. For this reason, ProCode won't result in excessive additional storage overhead vs. traditional erasure coding but will achieve higher recovery performance. Section III-B analyzes the ProCode's extra storage overhead in detail.

3) Disk Failure: When a disk failure actually occurs, ProCode will first check the block health degrees of the blocks residing on that disk. If a drive was predicted to be at risk and all of its blocks have been copied onto other healthy disks, the recovery process was completed in advance. If a drive had not been deemed at risk or only part of its blocks had been copied, its lost blocks will be repaired by RaidNode which will reconstruct lost blocks as usual in erasure coding. And a spare healthy disk will be added into the system because of the actually failed disk. In the event of simultaneous disk failures, ProCode proceeds the same as with single disk failure.

B. Theoretical analysis

Here we construct mathematical models for the system performance. We focus on three important parameters for disk failure prediction: the failure detection rate (FDR), the false alarm rate (FAR), and the warning time before drive failure, i.e., time in advance (TIA).

a) Storage overhead: The storage overhead owing to erasure coding can be described by

$$S_{\text{erasure}} = S_{\text{orig}} + S_{\text{code}}$$

where S_{erasure} is the storage required for erasure coding while S_{orig} and S_{code} represents the storage cost for original data blocks and code blocks, respectively. For $\mathrm{RS}(k, m)$ codes, where each stripe has k data blocks and m code blocks, we have

$$\frac{S_{\text{erasure}}}{S_{\text{orig}}} = \frac{k+m}{k}. \tag{2}$$

The total storage required by ProCode can be described by

$$S_{\text{erasure}} + S_{\text{FDR}} + S_{\text{FAR}}$$

where S_{FDR} and S_{FAR} are the extra storage overhead owing to correct (FDR) and incorrect (FAR) failures predictions, respectively. Predictions not to fail, whether correct or incorrect, do not result in increased storage.

To simplify the mathematical model, we make the assumption that blocks that are stored multiple times are stored precisely twice, contributing once to S_{erasure} and once to either S_{FDR} or S_{FAR}. While ProCode can result in blocks being stored three or more times, this requires that there are multiple copies of a block, at least two of which are on distinct at-risk disks, which we expect is sufficiently unlikely as to be negligible.

Alarms (whether correct or not) are resolved within a time period of length equal to the TIA: when an alarm is raised, a disk will either actually fail by time TIA, or be recognized as a false alarm, in which case ProCode deletes the unnecessary replicas. Thus, at a given time t, precisely those disks which are predicted to fail within the time interval $[t - \mathrm{TIA}, t)$ will have their blocks replicated. If the proportion of disks failing within that interval is df, then $\mathrm{df} \cdot \mathrm{FDR}$ of these are predicted and trigger replication. So

$$S_{\text{FDR}} = S_{\text{erasure}} \cdot \mathrm{df} \cdot \mathrm{FDR}, \tag{3}$$

and similarly we have

$$S_{\text{FAR}} = S_{\text{erasure}} \cdot (1 - \mathrm{df}) \cdot \mathrm{FAR}. \tag{4}$$

The proportional increase in storage in using ProCode (vs. unduplicated original blocks) is given by

$$\begin{aligned}
&\frac{S_{\text{erasure}} + S_{\text{FDR}} + S_{\text{FAR}}}{S_{\text{orig}}} \\
&= \frac{S_{\text{erasure}} + S_{\text{FDR}} + S_{\text{FAR}}}{S_{\text{erasure}}} \cdot \frac{k+m}{k}, \qquad \text{[by (2)]} \\
&= (1 + \mathrm{df} \cdot \mathrm{FDR} + (1 - \mathrm{df}) \cdot \mathrm{FAR}) \cdot \frac{k+m}{k} \\
&\qquad\qquad \text{[by (3) and (4)]} \\
&\leq (1 + \mathrm{df} + \mathrm{FAR}) \cdot \frac{k+m}{k}. \qquad (5)
\end{aligned}$$

Ordinarily, we expect that df will be relatively small (since hard drives don't fail often) and that FAR will be small (as

required by disk manufacturers). Thus we expect (5) to be close to that from pure erasure coding, i.e., $(k+m)/k$, which is the theoretical best possible (when no errors occur).

Experimental results for the classification tree method were given in [17], who found $\mathrm{FAR} \leq 0.001$, and [26] gave $\mathrm{df} \simeq 0.0015$ (with $\mathrm{TIA} \simeq$ two weeks). In this setting, (5) implies we can expect the storage costs to increase by a factor of around 0.0025 vs. pure erasure coding. Versus storing the data without redundancy, (5) instead gives a factor of 1.336 when using ProCode$(6, 2)$ or 1.203 when using ProCode$(10, 2)$.

b) Recovery time: In ProCode, additional recovery time can arise in three situations as a result of disk failure: (a) unpredicted disk failures, (b) disk failures occurring during the warning process, owing to insufficient TIA, and (c) predicted disk failures with sufficient TIA.

Let T denote the time required to reconstruct one block that has been erasure coded, and let $\overline{T}$ denote the time required to recover one block that has been duplicated. For simplicity, we consider the reconstruction time T of fail-in-warning disks with insufficient time in advance equal to disks which fail without any warning. Let FIW denote the proportion of fail-in-warning disks. The average block recovery time can therefore be modeled as follows:

$$(1-\mathrm{FDR})\cdot T + \mathrm{FDR}\cdot\mathrm{FIW}\cdot T + \mathrm{FDR}\cdot(1-\mathrm{FIW})\cdot\overline{T} \quad (6)$$

Disk failures can typically be correctly predicted for several days in advance [17], [36], which is sufficient time to generate a replica (particularly for an automated process) if adequate bandwidth is provided [18], which results in FIW close to 0, and thus implying a negligible second term in (6). We also expect FDR to be close to 1 in practice, e.g. around 95% as found by [17]. Thus, we bound (6) as

$$\leq \overline{T} + 0.05\,T. \quad (7)$$

To access lost blocks, RaidNode will launch a MapReduce job to recover it by retrieving the surviving blocks in the same stripe and then perform decoding. For $\mathrm{RS}(6, 2)$ for example, 6 blocks are required, so without even factoring in computation time, we have $T \geq 6\,\overline{T}$. So (7) gives an upper bound on the average block recovery time of $0.21\,T$, implying a lower bound on the speedup of 4.8 (vs. pure erasure coding).

Since ProCode does nothing for disks that are not predicted to fail, the average block recovery time will be at least $(1 - \mathrm{FDR}) \cdot T \simeq 0.05\,T$, i.e., the first term in (6). This gives an upper bound on the speedup of 20, which cannot be improved upon (without improving the FDR).

The degraded read latency, i.e., the time it takes to read a block on an failed drive, admits an analogous mathematical analysis to recovery time, and is omitted.

C. Prototype implementation

We implement ProCode as an extension to HDFS-RAID. Our prototype spans more than 1000 lines of code by modifying the modules of NameNode, RaidNode, and DataNode.

In HDFS-RAID, a degraded read can happen when a block's CRC check error occurs while system reconstruction is performed owing to a disk or node failure. During system reconstruction or degraded read, HDFS-RAID triggers a process called RaidNode which launches a MapReduce job on different DataNodes to decode corrupted blocks.

HDFS-RAID simply protects data using reactive fault tolerance and reconstructs lost data after failure happens. In contrast, ProCode proactively protects at-risk data before failures occur, adding some features such as health reporting, health monitoring, and dynamic replication, as shown in Figure 2.

IV. Evaluation

We (a) run simulations using sequential Monte Carlo methods [4] to compare ProCode's reliability (infrequency of data loss) to 3-way replication and erasure coding, and (b) build a small cluster to evaluate the performance of ProCode experimentally.

We write $\mathrm{ProCode}(k, m)$ to indicate the underlying erasure code, where k and m respectively denote the number of data blocks and code blocks in a stripe.

A. Methods

1) Simulation: In a single simulation, one possible system operating chronology is represented by random events. Virtual time is accumulated until a specified mission time is exceeded. During the simulation, we count the number of times data loss occurs.

We begin with two basic types of events that drive the virtual time forward: (a) *failure events*, which are triggered by Weibull distributed and independent disk failures, and (b) *failure rebuild-complete events*, which are triggered at the time when a disk failure repair session is finished.

To simulate ProCode, we add two additional events: (a) *warning events*, which are triggered by Weibull distributed and independent disk warnings predicted by the failure predictor, and (b) *warning rebuild-complete events*, which are triggered at the time when a disk warning backup session is finished.

Through simulation, we can estimate how many times data loss occurs during a time interval. To achieve this, we use the probability that a given minimal set of drive failures, which could possibly cause data loss, actually causes data loss (P_{dup} and P_{eras} in what follows). These probabilities depend on how blocks are stored and are used in Section IV-B1 to evaluate system reliability. At present, storage systems typically use rack-aware placement strategies where data blocks are dispersed across racks and nodes, which we take into account for simulations.

a) 3-way replication: In a 3-way replication system, the 3 copies of a data block will be stored in 3 distinct nodes, exactly 2 of which belonging to the same rack.

Suppose we have r racks, n nodes per rack, and d disks per node in a storage system. Call a selection of three disks such that two disks belong to distinct nodes on one rack, and the remaining disk belongs to another rack, a *suitable* configuration. Suppose three disks in a random suitable configuration fail, and that disk A is one of those disks. There are two types

of such suitable configurations, illustrated in Figure 3. Thus, the number of suitable configurations A belongs to is

$$\overbrace{(n-1)d \times (r-1)nd}^{\text{Figure 3a}} + \overbrace{(r-1)\binom{n}{2}d^2}^{\text{Figure 3b}}$$
$$= \frac{3(r-1)n(n-1)d^2}{2}.$$

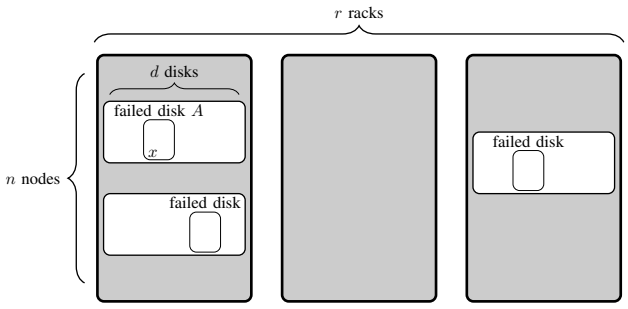
So the probability that a data block x stored on disk A is also stored on the two other disks that are lost is

$$p_{\text{dup}} := \frac{2}{3(r-1)n(n-1)d^2}.$$

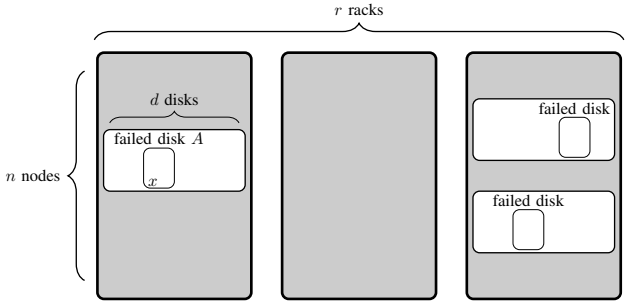
If there are b blocks per disk, then the probability of any block on A being lost caused by these failures is

$$P_{\text{dup}} := 1 - (1 - p_{\text{dup}})^b$$

assuming duplicate blocks are stored in random suitable configurations.



(a) Disk A belongs to a node on the same rack as another node containing an failed disk, and another failed disk belongs to a node on a different rack.



(b) Disk A belongs to a node on a different rack than the two other failed disks, which belong to two nodes on the same rack.

Fig. 3: The two ways in which disk A can belong to a suitable configuration of three disks, which are all failed.

b) Erasure coding: In an erasure system tolerant of m erasures, there will be k data blocks and m code blocks in each stripe, and each of the $k+m$ blocks will be stored on disks in separate racks. For data loss to occur, we need $m+1$ simultaneous failures; suppose disk A is one of those $m+1$ disks which contains some data block x. There are

$$(nd)^{k+m-1}\binom{r-1}{k+m-1}$$

ways in which the remaining blocks in the stripe containing x could be distributed among disks. Of these, there are

$$(nd)^{k-1}\binom{r-1-m}{k-1}$$

ways in which a given set of $m+1$ disks, each of which containing blocks in the stripe containing x, and one of which being disk A, occur. So the probability that a given set of $m+1$ data blocks in the stripe containing x, including disk A, are lost is

$$p_{\text{eras}} := \frac{(nd)^{k-1}\binom{r-1-m}{k-1}}{(nd)^{k+m-1}\binom{r-1}{k+m-1}}.$$

If there are b blocks per disk, then the probability that erasure of those $m+1$ disks results in data loss is therefore

$$P_{\text{eras}} := 1 - (1 - p_{\text{eras}})^b$$

assuming blocks are stored randomly on distinct racks.

2) Experiment: ProCode is developed based on HDFS-RAID and deployed in our cluster with 13 nodes, one NameNode and 12 DataNodes, linked by gigabit network, and each node is configured with CentOS 6.3, four Intel Xeon CPUs @ 2.80GHz, 1GB memory, and 500GB SATA disk storage. We use the default HDFS file system block size of 64 MB and the default policy of block placement for erasure coding. We upload files whose size distribution has a lognormal distribution to the HDFS-RAID cluster. In ProCode, disk warning and failure events are generated (matching some desired failure prediction rate and false alarm rate) while in 3-way replication and erasure code systems, only failure events are produced.

B. System Comparison

In this section, we use simulation and experiments to evaluate the reliability and performance of ProCode. We experiment with both $\mathrm{RS}(6,2)$ and $\mathrm{RS}(10,2)$ as the underlying erasure code in ProCode.

1) System Reliability: We simulate the various systems, estimating the expected number of data loss events that occur over 10 years. Fewer data loss events implies greater system reliability. We use P_{dup} and P_{eras} discussed in Section IV-A1 to evaluate whether data loss would actually occur when a given minimal set of drive failures occurs.

We use a Weibull distribution to simulate failure and warning events (setting the shape parameter $\beta = 1.12$ in the Weibull expression [9]). In the simulation, we set the mean time to failure (MTTF) to 100000 hours, and set TIA to 360 hours. We vary the FDR from 60% to 90%.

We perform two related experiments:

1) We simulate a system with a varying number of racks, with each rack containing 20 nodes, and each node containing 10240 blocks. We set the mean time to repair (MTTR) to 24 hours.
2) We simulate a system as the system MTTR varies. Here, the number of racks is set to 640.

The results are plotted in Figure 4.

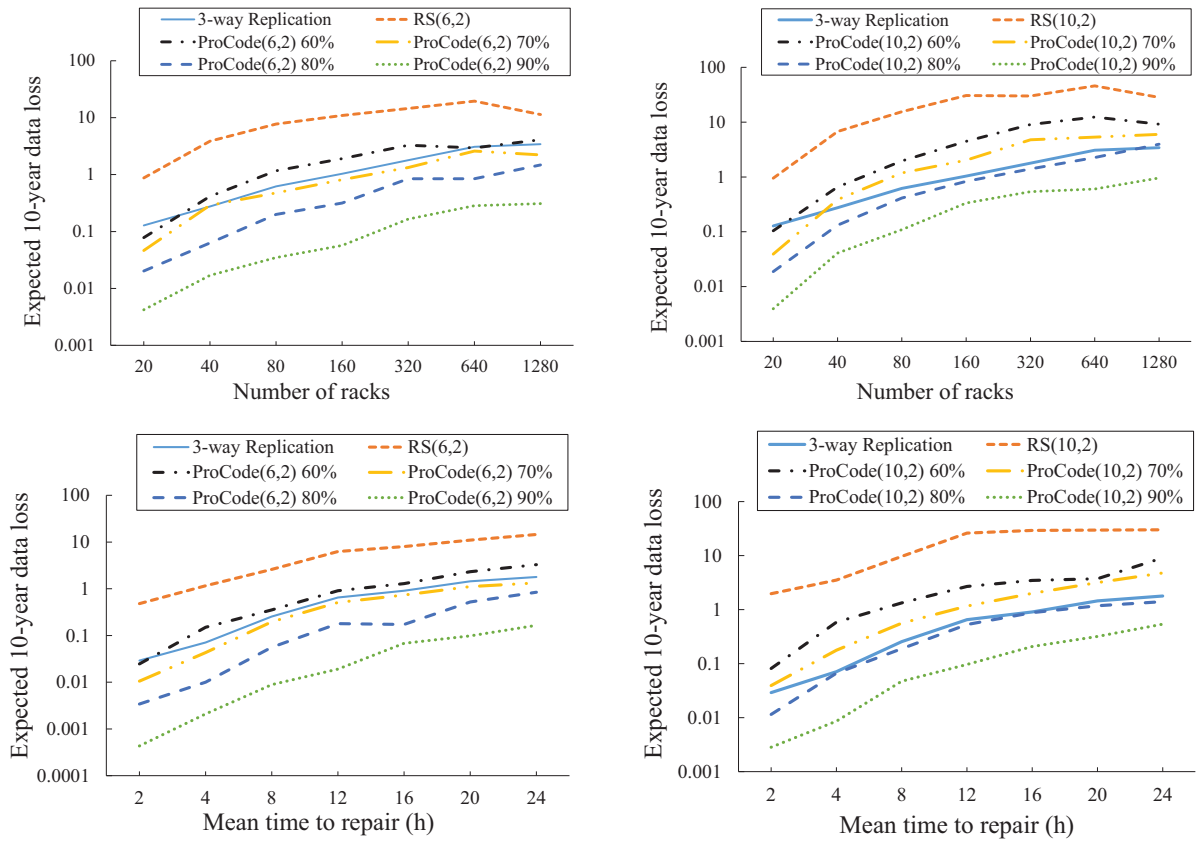


Fig. 4: Expected number of data loss events over a 10-year period for $\mathrm{ProCode}(6,2)$ (left column) and $\mathrm{ProCode}(10,2)$ (right column) vs. $\mathrm{RS}(6,2)$, $\mathrm{RS}(10,2)$, and 3-way replication as the number of racks varies (top row) and the mean time to repair varies (bottom row). The percentage next to ProCode in the key is the failure detection rate.

As we would expect, more accurate failure predictions result in higher reliability. With each 10% increase in FDR, the reliability is enhanced nearly by an order of magnitude; this highlights the benefit of a proactive fault tolerance. Also as we would expect, a system's reliability tends to decrease as MTTR increases, as it provides more time for simultaneous disk failures to occur in.

The expected number of data loss events over 10 years in the default 3-way replication system is one order of magnitude lower than the erasure code system of $\mathrm{RS}(6,2)$ and two orders lower than that of $\mathrm{RS}(10,2)$. With proactive fault tolerance, $\mathrm{ProCode}(6,2)$ with an FDR of 70% and $\mathrm{ProCode}(10,2)$ with an FDR of 80% achieves nearly the same reliability compared with default 3-way replication system.

While both $\mathrm{ProCode}(6,2)$ and $\mathrm{ProCode}(10,2)$ can recover from any two erasures, $\mathrm{ProCode}(6,2)$ has a higher reliability but uses more storage space. While 3-way replication is designed to be highly reliable (at a significant cost in terms of storage space), we find that ProCode incurs fewer data loss events when failure detection rate is around 80% or more.

With a MTTR of 4 hours, the expected number of 10-year data loss events for 3-way replication is 0.07. With a FDR of 90%, the same level of reliability can be achieved with a MTTR of around 20 hours with $\mathrm{ProCode}(6,2)$ and around 12 hours with $\mathrm{ProCode}(10,2)$.

2) Performance Results: We conduct experiments on the 13-node cluster to evaluate two aspects of ProCode's performance: degraded read latency and recovery time. We use a FDR of 80%, FAR of 0.1%, and TIA of 360 hours which were found to be reasonable in [17].

We measure the degraded read latency as follows: we inject disk events with given parameters and use a single client to randomly read 16000 data blocks which are either corrupted (and need to be reconstructed by a MapReduce job) or having a healthy replica (owing to disk warning) and calculate the block's expected read latency and average storage overhead. During the reconstruction time phase, the disk events are produced in the same way as when measuring degraded read. Some disk failures can be recovered quickly owing to pre-warning while others which weren't predicted accurately are network and IO intensive and can be time consuming. We measure the average recovery time and average storage overhead of different kinds of failures.

a) Degraded read latency: Figure 5 plots the degraded read latency and storage overhead of ProCode vs. the 3-way replication system, and the erasure code systems using $\mathrm{RS}(6,2)$ and $\mathrm{RS}(10,2)$. In ProCode, during a warning process, resources are consumed to create new replicas for the at-risk blocks, which we term *migration*. Consequently, we test ProCode's degraded read latency in two circumstances: (a) where network bandwidth used for migration is not throttled ("best-effort" strategy), which we call *BE-ProCode*, and (b) where network bandwidth used for migration is throttled to about 10% ("leisure" strategy), which we call *LS-ProCode*.

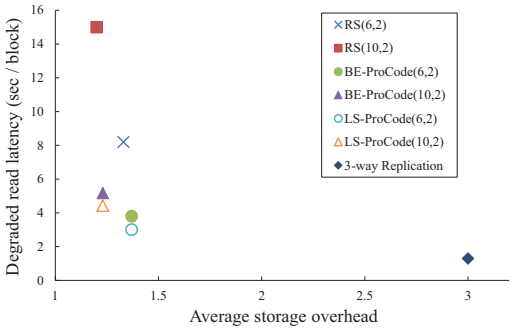


Fig. 5: The degraded read latency and the storage overhead of ProCode under various conditions, two erasure coded systems, and the 3-way replication system.

We observe that $\mathrm{ProCode}(6,2)$ has a 63% lower degraded read latency than $\mathrm{RS}(6,2)$, and $\mathrm{ProCode}(10,2)$ has a 70% lower degraded read latency than $\mathrm{RS}(10,2)$. This improvement is due to failure pre-warning: at-risk blocks will be copied to other healthy disks in advance, so when a disk failure actually occurs, there is a healthy replica for use and there's no need to rebuild the lost block.

In ProCode, a degraded read takes only twice as long as the 3-way replication scheme when the leisure copy strategy is used, and about three times as long when the best-effort strategy is used. In comparison, a degraded read will take about 6.3 and 11.5 times longer for erasure-coded system with $\mathrm{RS}(6,2)$ and $\mathrm{RS}(10,2)$, respectively. This is consistent with the theoretical analysis because a degraded read for $\mathrm{RS}(6,2)$ and $\mathrm{RS}(10,2)$ will require 6 and 10 times of network transfers and disk IOs respectively, and extra decoding operations as well.

The average storage overhead is also presented in Figure 5, displaying the trade-off in storage space and degraded read latency. The 3-way replication system requires 3 times as much storage space as a system without any redundancy, whereas erasure-coded systems respectively take around 1.2 and 1.3 times as much time as the erasure-coded systems with $\mathrm{RS}(6,2)$ and $\mathrm{RS}(10,2)$. ProCode takes a little more space than that of its underlying erasure code. This slight additional storage cost is the price paid for significantly better degraded read latency in ProCode. Versus 3-way replication, ProCode uses much less storage space with only a slight reduction in degraded read performance.

b) Recovery time: Figure 6 compares the node recovery time after the failure of a 110GB disk using erasure coding, ProCode and 3-way replication. And if not specified, the following experiments are conducted using "best-effort" strategy for migration.

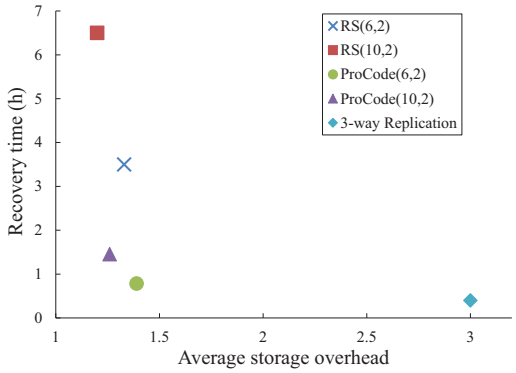


Fig. 6: Time required for the recovery of 110GB node and the average proportional storage overhead. The data points for ProCode have been modified from their original values as described in the main text.

In 3-way replication, the server takes about 0.22 hours to recover because the lost blocks on the failed disk still have two available replicas distributed evenly on other healthy disks which can be used to launch block recovery simultaneously. At the other extreme, it takes up to 6.5 hours in $\mathrm{RS}(10,2)$ and 3.5 hours in $\mathrm{RS}(6,2)$ to reconstruct the same number of corrupted blocks in erasure-coded system. The reason for slow reconstruction is that when one block is lost, the system need to read several other blocks in the same stripe to rebuild it.

In ProCode, most of the disk failures are predicted sufficiently early, giving adequate time to protect at-risk blocks by creating replicas on other healthy drives, therefore when a disk failure occurs, the recovery process amounts to metadata modification. Therefore, for failure predictions with sufficient TIA, the recovery time can be reduced from hours to minutes. However, there will still be some disk failures that are not properly predicted, and some failures which happen before the at-risk blocks are completely copied. These cases form a small proportion of all failures, so ProCode shows almost the same level of recovery performance as the 3-way replication system. If a higher FDR is achieved, ProCode can even outperform the 3-way replication system.

On the cluster available to the authors, consisting of only 13 nodes, we find hour-level recovery for one drive failure in RS coding. In a large cluster with thousands of nodes, the recovery time for one drive failure can be decreased to minute-level by spreading the network and computation overhead to multiple nodes. However, the total amount of data to be transferred upon disk failure will remain constant.

C. Sensitivity Study

The ProCode performance is influenced by several important factors, including FDR, FAR, and TIA. In this section,

we test how these factors affect the total amount of data transferred (TADT) for data recovery which indicates the level of consumption of system resources (disk bandwidth, network bandwidth, etc.) during recovery.

As in [26], an annual disk failure rate of 2% to 4% is common (i.e., daily, 0.01% to 0.03% of disks fail). Li et al. [17], observed $\leq 0.1\%$ FAR in a two-day time window. With this in mind, we set the number of drive failure events equal to the number of false alarm events in a fixed time period with FAR of 0.1% in the following experiment.

The available cluster has 13 nodes and it may require several years to have even a single actual drive failure, so we simulate failures on the cluster. Given a required number of failure events, we use a simulation process to produce the failure events, disk warning events, and false alarm events to give desired FDR, FAR, and TIA values and apply them to the disks in the cluster. If the next event is e.g. "in 100 days, disk X is falsely predicted to fail", we skip time forward 100 days.

c) FDR: To evaluate how the FDR affects the system, we conduct experiments to measure the TADT to perform system recovery as FDR varies from 0% to 90%. We set the FAR at 0.1%, and run the generation process until 100 failure events and 100 false alarm events have occurred. Figure 7 plots the results.

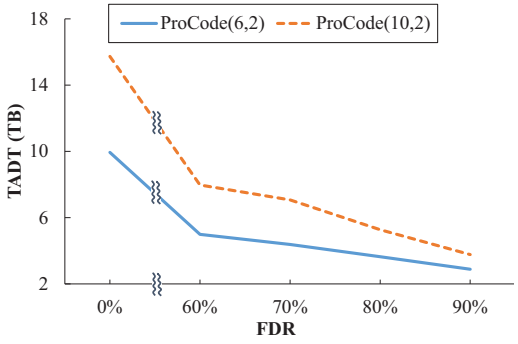


Fig. 7: Total amount of data transferred with $\mathrm{ProCode}(6,2)$ and $\mathrm{ProCode}(10,2)$ with varying FDR. Here, FAR is 0.1% and TIA is 360 hours. (Note: the horizontal axis has a discontinuity between 0% and 60%.)

As expected, TADT generally decreases as FDR increases, since with a higher FDR, more disk failures are (correctly) predicted in advance so that the warning process can duplicate at-risk data blocks. Theoretically, $\mathrm{ProCode}(6,2)$ recovering from an unpredicted drive failure should require about 6 times the amount of data transferred vs. from a predicted drive failure. However, experiment results are not completely consistent with this ideal situation; this could be caused e.g. by insufficient TIA to migrate blocks or false alarms, which bring additional and unnecessary data transfers.

d) FAR: We measure the total amount of data transferred with ProCode as FAR varies from 0% to 0.8%. This will result in the number of false alarm events varying from 0 to 800. The results are plotted in Figure 8.

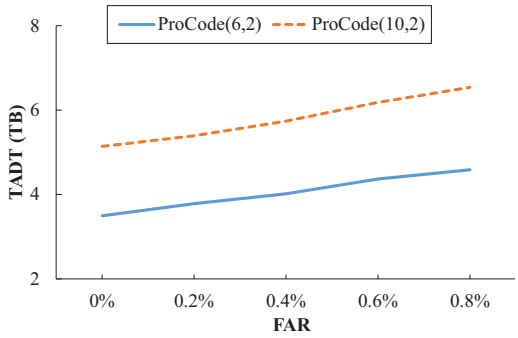


Fig. 8: Total amount of data transferred with $\mathrm{ProCode}(6,2)$ and $\mathrm{ProCode}(10,2)$ with varying FAR. Here, FDR is 80% and TIA is 360 hours.

As we would expect, we see that TADT increases with an increasing number of false alarms. With each false alarm, additional and unnecessary data will be transferred. ProCode eventually deletes these unnecessary block replicas when false alarms occur.

e) TIA: To investigate the effect of TIA on ProCode, we conduct experiments with TIA varying from 24 to 384 hours. The results are plotted in Figure 9.

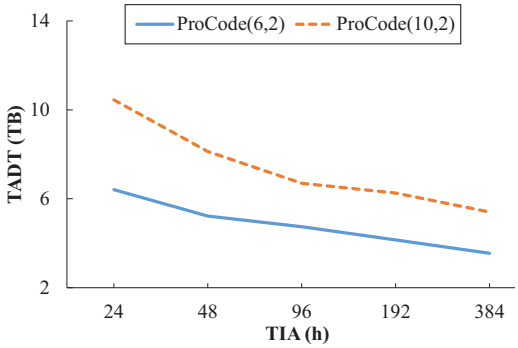


Fig. 9: Total amount of data transferred with $\mathrm{ProCode}(6,2)$ and $\mathrm{ProCode}(10,2)$ with varying TIA. Here, FAR is 0.1% and FDR is 80%.

We can see that TADT decreases with increasing TIA. If the TIA is sufficient, when failures actually occur, at-risk blocks will have been migrated to healthy drives, and thus less resources are required to recover from these failures.

f) Trade-off: In general, high FDR can be obtained at the expense of FAR. In ProCode, higher FDR, lower FAR, and longer TIA will result in improved performance, but ProCode is more sensitive to changes in FDR than FAR (cf. Figures 7 and 8). Therefore, a ProCode system would benefit from a higher-than-usual FDR. Zhu et al. [36] reported that two different parameter pairs lead to an FDR of 68.5% with 0.03% FAR, and an FDR of 80.0% with 0.3% FAR; we should choose the latter to reduce the recovery overhead. The trade-off between FDR and FAR, however, will affect TIA, which also should be sufficiently large (see Figure 9).

V. Concluding remarks

Petabyte-scale data levels are becoming common in large-scale storage system [12], [25], [28] and disk failures occur frequently. Two primary approaches have drawbacks: replication increases storage costs and erasure coding results in a large communication overhead when reconstructing lost or corrupted blocks.

Recent studies focus on the optimization of erasure coding to improve reconstruction performance. For example, LRCs [12], Weaver codes [10], and Hover codes [11] are all specifically designed for cloud storage systems, and use additional storage for efficient data reconstruction. In contrast, ProCode takes advantage of drive failure prediction to almost eliminate the need for block recovery using erasure coding. It combines ideas behind previous approaches, such as RAIDShield [19], which proactively monitors the disk health and quantifies the vulnerability of RAID groups. This is achieved in ProCode by using the classification tree model from [17].

In future research, it would be interesting to employ the underlying idea of ProCode to other kinds of codes, aiming to apply them to cloud storage systems in order to optimize the trade-off of storage overhead vs. recovery performance.

In this paper, we focus on individual drive failure, but in a real data center containing tens of thousands of servers, whole server failures can happen. A server failure can simultaneously erase many hard drives, which would result in a much higher recovery cost than a single hard drive erasure. This is another possible future work direction.

Acknowledgments

The authors thank Mingze Li and Rui Ma for their input.

References

[1] "HDFS RAID wiki," http://wiki.apache.org/hadoop/HDFS-RAID.

[2] C. L. Abad, Y. Lu, and R. H. Campbell, "Dare: Adaptive data replication for efficient cluster scheduling," in *Cluster Computing (CLUSTER), 2011 IEEE International Conference on*. Ieee, 2011, pp. 159–168.

[3] B. Allen, "Monitoring hard disks with SMART," *Linux Journal*, no. 117, pp. 74–77, 2004.

[4] C. L. Borges, D. M. Falcão, J. C. O. Mello, and A. C. Melo, "Composite reliability evaluation by sequential Monte Carlo simulation on parallel and distributed processing environments," *IEEE Trans. Power Systems*, vol. 16, no. 2, pp. 203–209, 2001.

[5] A. Cidon, S. M. Rumble, R. Stutsman, S. Katti, J. K. Ousterhout, and M. Rosenblum, "Copysets: Reducing the frequency of data loss in cloud storage." in *Proc. ATC*, 2013, pp. 37–48.

[6] B. Fan, W. Tantisiriroj, L. Xiao, and G. Gibson, "Diskreduce: Raid for data-intensive scalable computing," in *Proceedings of the 4th Annual Workshop on Petascale Data Storage*. ACM, 2009, pp. 6–10.

[7] A. Fikes, "Colossus, successor to Google File System," http://static.googleusercontent.com/media/research.google.com/en/us/university/relations/facultysummit2010/storage_architecture_and_challenges.pdf, [Online; accessed 14th September 2015].

[8] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," in *ACM SIGOPS operating systems review*, vol. 37, no. 5. ACM, 2003, pp. 29–43.

[9] K. Gopinath, J. Elerath, and D. Long, "Reliability modelling of disk subsystems with probabilistic model checking," Technical Report UCSC-SSRC-09-05, University of California, Santa Cruz, Tech. Rep., 2009.

[10] J. L. Hafner, "WEAVER codes: Highly fault tolerant erasure codes for storage systems." in *Proc. FAST*, vol. 5, 2005, pp. 16–16.

[11] ——, "Hover erasure codes for disk arrays," in *Dependable Systems and Networks, 2006. DSN 2006. International Conference on*. IEEE, 2006, pp. 217–226.

[12] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, S. Yekhanin *et al.*, "Erasure coding in Windows Azure storage," in *Proc. ATC*, 2012, pp. 15–26.

[13] G. F. Hughes, J. F. Murray, K. Kreutz-Delgado, and C. Elkan, "Improved disk-drive failure warnings," *IEEE Trans. Reliability*, vol. 51, no. 3, pp. 350–357, 2002.

[14] X. Ji, Y. Ma, R. Ma, P. Li, J. Ma, G. Wang, X. Liu, and Z. Li, "A proactive fault tolerance scheme for large scale storage systems," in *Algorithms and Architectures for Parallel Processing*. Springer, 2015, pp. 337–350.

[15] R. T. Kaushik and M. Bhandarkar, "Greenhdfs: towards an energy-conserving, storage-efficient, hybrid hadoop compute cluster," in *Proceedings of the USENIX Annual Technical Conference*, 2010, p. 109.

[16] O. Khan, R. C. Burns, J. S. Plank, W. Pierce, and C. Huang, "Rethinking erasure codes for cloud file systems: minimizing I/O for recovery and degraded reads." in *Proc. FAST*, 2012, p. 20.

[17] J. Li, X. Ji, Y. Jia, B. Zhu, G. Wang, Z. Li, and X. Liu, "Hard drive failure prediction using classification and regression trees," in *Proc. DSN*, 2014, pp. 383–394.

[18] J. Li, R. J. Stones, G. Wang, Z. Li, and X. Liu, "Being accurate is not enough: New metrics for disk failure prediction," in submission.

[19] A. Ma, F. Douglis, G. Lu, D. Sawyer, S. Chandra, and W. Hsu, "RAIDShield: characterizing, monitoring, and proactively protecting against disk failures," in *Proc. FAST*, 2015, pp. 241–256.

[20] J. F. Murray, G. F. Hughes, and K. Kreutz-Delgado, "Machine learning methods for predicting failures in hard drives: A multiple-instance application," in *J. Mach. Learn. Res.*, 2005, pp. 783–816.

[21] D. S. Papailiopoulos, J. Luo, A. G. Dimakis, C. Huang, and J. Li, "Simple regenerating codes: Network coding for cloud storage," in *Proc. INFOCOM*, 2012, pp. 2801–2805.

[22] A. Qin, D. Hu, J. Liu, W. Yang, and D. Tan, "Fatman: Cost-saving and reliable archival storage based on volunteer resources," *Proc. VLDB Endowment*, vol. 7, no. 13, pp. 1748–1753, 2014.

[23] K. V. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, "A hitchhiker's guide to fast and efficient data reconstruction in erasure-coded data centers," in *Proc. SIGCOMM*, 2014, pp. 331–342.

[24] 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, 2011.

[25] F. B. Schmuck and R. L. Haskin, "GPFS: A shared-disk file system for large computing clusters." in *Proc. FAST*, vol. 2, no. 19, 2002.

[26] B. Schroeder and G. A. Gibson, "Disk failures in the real world: What does a MTTF of 1,000,000 hours mean to you?" in *Proc. FAST*, vol. 7, 2007, pp. 1–16.

[27] N. B. Shah, K. Rashmi, P. V. Kumar, and K. Ramchandran, "Distributed storage codes with repair-by-transfer and nonachievability of interior points on the storage-bandwidth tradeoff," *IEEE Trans. Inf. Theory*, vol. 58, no. 3, pp. 1837–1852, 2012.

[28] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop distributed file system," in *Proc. MSST*, 2010, pp. 1–10.

[29] Y. Wang, Q. Miao, E. W. Ma, K.-L. Tsui, and M. G. Pecht, "Online anomaly detection for hard disk drives based on Mahalanobis distance," *IEEE Trans. Reliability*, vol. 62, pp. 136–145, 2013.

[30] Z. Wang, A. G. Dimakis, and J. Bruck, "Rebuilding for array codes in distributed storage systems," in *Proc. GLOBECOM Workshops*, 2010, pp. 1905–1909.

[31] J. Wilkes, R. Golding, C. Staelin, and T. Sullivan, "The hp autoraid hierarchical storage system," *ACM Transactions on Computer Systems (TOCS)*, vol. 14, no. 1, pp. 108–136, 1996.

[32] S. Wu, H. Jiang, and B. Mao, "Proactive data migration for improved storage availability in large-scale data centers," *IEEE Trans. Comput.*, vol. 64, no. 9, pp. 2637–2651, 2015.

[33] M. Xia, M. Saxena, M. Blaum, and D. A. Pease, "A tale of two erasure codes in HDFS," in *Proc. FAST*, 2015.

[34] L. Xiang, Y. Xu, J. Lui, and Q. Chang, "Optimal recovery of single disk failure in RDP code storage systems," in *ACM SIGMETRICS*, vol. 38, no. 1, 2010, pp. 119–130.

[35] Q. Xin, E. L. Miller, T. Schwarz, D. D. E. Long, S. Brandt, and W. Litwin, "Reliability mechanisms for very large storage systems," in *Proc. MSST*, 2003, pp. 146–156.

[36] B. Zhu, G. Wang, X. Liu, D. Hu, S. Lin, and J. Ma, "Proactive drive failure prediction for large scale storage systems," in *Proc. MSST*, 2013, pp. 1–5.