

Evaluation of Distributed Recovery in Large-Scale Storage Systems

Qin Xin Ethan L. Miller
Storage Systems Research Center
University of California, Santa Cruz
{qxin, elm}@cs.ucsc.edu

Thomas J. E. Schwarz, S. J.
Computer Engineering Dept.
Santa Clara University
tjschwarz@scu.edu

Abstract

Storage clusters consisting of thousands of disk drives are now being used both for their large capacity and high throughput. However, their reliability is far worse than that of smaller storage systems due to the increased number of storage nodes. RAID technology is no longer sufficient to guarantee the necessary high data reliability for such systems, because disk rebuild time lengthens as disk capacity increases. In this paper, we present FAsT Recovery Mechanism (FARM), a distributed recovery approach that exploits excess disk capacity and reduces data recovery time. FARM works in concert with replication and erasure-coding redundancy schemes to dramatically lower the probability of data loss in large scale systems. We have examined essential factors that influence system reliability, performance, and costs, such as failure detections, disk bandwidth usage for recovery, disk space utilization, disk drive replacement, and system scales, by simulating system behavior under disk failures. Our results show the reliability improvement from FARM and demonstrate the impacts of various factors on system reliability. Using our techniques, system designers will be better able to build multi-petabyte storage systems with much higher reliability at lower cost than previously possible.

1. Introduction

Five exabytes (5×2^{60} bytes) of new information were generated in 2002, and new data is growing annually at the rate of 30% [22]. System designers are building ever-larger storage clusters to meet such rapidly increasing demand on data capacity with high I/O performance. The national labs, for instance, are planning to build a two petabyte storage system for use in large-scale scientific simulations and experiments. This application motivates our research.

Advances in storage technology have reduced cost and improved performance and capacity; however, disk drive

reliability has only improved slowly. In a typical environment for large storage systems, such as supercomputing systems, data loss is intolerable: losing just the data from a single drive, while it might hold less than 0.1% of the total storage, can result in the loss of a large file spread over thousands of drives. A failure in a single device might be rare, but a system with thousands of devices will experience failures and even groups of almost simultaneous failures much more frequently. As an example, consider the Internet Archive [19], a digital library that contains over 100 terabytes of compressed data and suffers about thirty disk failures per month. To make matters worse, the time to rebuild a single disk is becoming longer as increases in disk capacity outpace increases in bandwidth [16]. These phenomena make it challenging to ensure high reliability for large-scale storage systems.

In this paper, we propose FARM (FAsT Recovery Mechanism), which makes use of declustering [2, 23, 25] that reduces the time to deal with a disk failure. RAID designers have long recognized the benefits of declustering for system performance. Declustering distributes the mirrored copies or redundancy groups across the disk array, so that, after a disk failure, the data needed to reconstruct the lost data is distributed over a number of drives in the disk array. Thus, declustering leads to good performance for storage systems in degraded mode. FARM uses declustering not only to improve performance during data recovery, but, more importantly, to improve reliability. FARM deals with the consequences of failures much faster and thus limits the chance that additional failures lead to data loss during the window of vulnerability. We examine distributed recovery in a very large-scale system and show how FARM improves reliability across a wide range of system characteristics.

To the best of our knowledge, no research yet has been directed towards the architecture of such a high-performance, large-scale system with such high reliability demands. OceanStore [28] aims for a high availability and high security world-wide peer-to-peer system, but does not provide high bandwidth. FARSITE [1] stores data in free disk space on workstations in a corporate network.

While both systems have concerns similar to ours, they have less control over the individual storage devices and they provide primarily read-only access at relatively low bandwidths (megabytes per second). In principle, we are close to the classical storage solutions such as RAID [6]. However, the size of our system is so much larger that simply using traditional RAID techniques alone will not provide sufficient reliability.

We studied FARM in a petabyte-scale storage system with thousands of storage devices. Storage systems built from Object-based Storage Devices (OSDs), which are capable of handling low-level storage allocation and management, have shown great promise for superior scalability, strong security, and high performance [13]. Our mechanisms not only provide high reliability for object-based storage systems, but also are applicable to general large-scale data storage systems built from block devices. We use the term “disk drives” to refer to both OSDs and traditional block devices.

Our simulation results show that FARM is successful across most data redundancy techniques. We also investigate the factors that influence system reliability, including failure detection latency, data recovery bandwidth, disk space utilization, disk drive replacement, and overall system scales. With our reliability schemes and analysis for related factors, system designers can choose the techniques necessary to ensure that their storage system is sufficiently reliable for their users.

2. Fast Recovery Mechanism

As the increase in capacity outpaces that of bandwidth, disk rebuild time is increasing. During rebuilding, additional disk failures can lead to data loss. Traditional recovery schemes such as RAID that rebuild the data on a failed drive onto a single replacement drive cannot guarantee sufficient reliability in large scale storage systems, as we will see in Section 3. We propose FARM, a FASt Recovery Mechanism uses declustering to speed up recovery from disk failure and thus achieve high reliability for very large-scale storage systems. In FARM, failure and recovery are transparent to users.

2.1. Redundancy Groups

In order to achieve high reliability in large systems, user data is stored redundantly using either replication or some form of erasure correcting code such as storing the parity of a group of blocks. We call a group of data blocks composed of user data and their associated replicas or parity / erasure code blocks a *redundancy group*. The size of a redundancy group is the total user data in the group, not including the replicas or parity blocks.

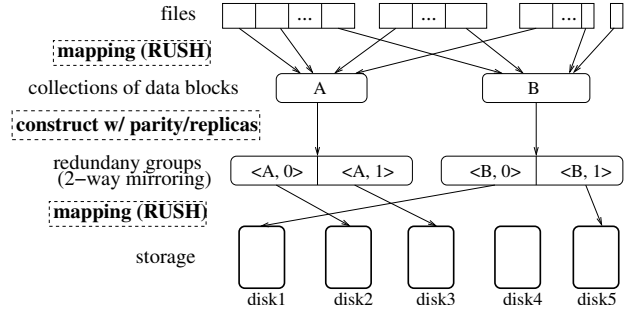


Figure 1. Redundancy groups construction.

2.1.1. Data Distribution

Before discussing the configuration schemes for redundancy groups, we give a brief illustration of our system: a petabyte-scale storage system built from thousands of disk drives. Files are broken up into fixed-size *blocks*; the default size of a block is 1 MB. A number of blocks are gathered in a *collection* through the mapping of block to collection done by RUSH [17]. A collection is then assigned to a *redundancy group* by redundancy schemes, such as replication or adding parity blocks. Collections of blocks enable efficient data management and redundancy groups provide enhanced reliability. We use RUSH again to allocate redundancy groups to storage devices. Figure 1 illustrates the construction process of redundancy group A and B with two-way mirroring configuration. Each data block is marked as $\langle grp_id, rep_id \rangle$, where grp_id is the identifier of the group to which it belongs and rep_id is its identifier in the group. Data blocks that reside in the same redundancy group are called “buddies;” they share one grp_id with various values of rep_id .

2.1.2. Configuration of Redundancy Groups

No redundancy scheme is simpler than replication. An n -way mirroring scheme simply maintains n copies of each user data block, each residing on a different disk. Alternatively, a parity scheme adds a block containing the (XOR) parity of the user data blocks. For higher failure tolerance, we can use an erasure correcting code (ECC) to generate more than a single parity block. These *generalized parity blocks* are made up of the check symbols of the code.

There are many good candidates for an ECC. Since disk access times are comparatively long, time to compute an ECC is relatively unimportant. A good ECC will create k parity blocks of the same size as the m user data blocks. It will be able to reconstruct the contents of any block out of m parity or data blocks. Examples of such ECCs are generalized Reed-Solomon schemes [21, 27] and Even-Odd [4]. These are generically called m/n schemes, where $n = m + k$. An m/n scheme gives us m -availability, but must

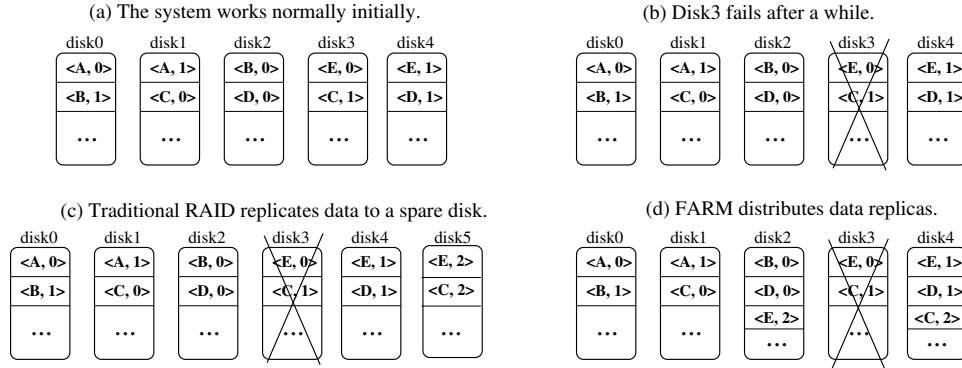


Figure 2. Upon disk failure, different behaviors with and w/o FARM.

update all k parity blocks whenever a data block changes. If only a single block changes, this can often be done as in RAID 5 by calculating the difference between the new and the old user data. The difference is then propagated to all parity blocks, which are updated by processing the difference and the old parity values.

The blocks in a redundancy group reside on different disks, but each disk contains data that belongs to different redundancy groups. We allocate redundancy groups to disk drives in a fully distributed fashion using an algorithm [17] that gives each disk statistically its fair share of user data and parity data, so that read and write loads are well balanced. A good data placement algorithm limits the amount of data any disks contributes to the data recovery process; the one we are using has this property.

The trade-offs between the different redundancy mechanisms are ease of implementation, the complexity of parity management and recovery operations, the bandwidth consumed by recovery [36], and the storage efficiency, *i. e.*, the ratio between the amount of user data and the total amount of storage used. Two-way mirroring is easy to implement and simple to run, but only has a storage efficiency of $1/2$. An m/n scheme is more complex and write performance is worse, but it has better storage efficiency of m/n . There are some schemes that put a user data block into more than one redundancy group [15] and mixed schemes that structure a redundancy group by data blocks and an (XOR-)parity block, and a mirror of the data blocks with parity.

The consequences for reliability of each scheme depend greatly on the system in which they are deployed. For example, the FARSITE implementers noticed that failure of storage sites could no longer be considered independent for a replication factor $m > 4$ [9]. In a large storage system, placement and support services to the disk introduce common failure causes such as a localized failure in the cooling system.

2.2. Design Principles

The traditional recovery approach in RAID architectures replicates data on a failed disk to one dedicated spare disk upon disk failure. Such a scheme works properly in a small system consisting of up to one hundred disk drives, but it fails to provide sufficient reliability for systems with thousands of disks.

Menon and Mattson [23] proposed the distribution of a dedicated spare disk in a RAID 5 system among all the disks in the disk array. Each disk then stores blocks for user data, blocks for parity data, and blocks reserved for data from the next disk to fail. Some time after a failure, a replacement disk is rebuilt from the data distributed to the storage array from the failed disk. *Distributed sparing* results in better performance under normal conditions because the load is divided over one more disk, but has the same performance in “degraded mode” (after a disk failure). In addition, reliability benefits greatly from reduced data reconstruction time after a disk failure [33], because of a smaller window of vulnerability to further drive losses.

The large storage systems that we envision do not differ only in scale from disk arrays; they are also dynamic: batches of disk drives will be deployed to replace failed or old disks and to provide additional capacity. Our proposal for systems of this sort reduces recovery time by parallelizing the data rebuild process and increases redundancy adaptively when system load and capacity utilization allow. We can characterize FARM as follows: A RAID is *declustered* if parity data and spare space are evenly distributed throughout the disk array. Declustering is motivated by performance. FARM is declustering at a much higher scale, with a primary focus on reliability, though performance also benefits.

Figure 2 illustrates the principles of FARM in a small storage system, compared with a traditional RAID. For simplicity, we use two-way mirroring. With disk 3 fails, FARM creates a new replica for each redundancy group that had a replica on disk 3, blocks C and E in Figure 2(b).

Rather than create all of the replicas on a spare disk, say disk 5 shown in Figure 2(c), FARM distributes all new replicas to different disks, as shown in Figure 2(d). In a storage system with thousands of disks, replication can proceed in parallel, reducing the window of vulnerability from the time needed to rebuild an entire disk to the time needed to create one or two replicas of a redundancy group, greatly reducing the probability of data loss.

Our data placement algorithm, RUSH [17] provides a list of locations where replicated data blocks can go. After a failure, we select the disk on which the new replica is going to reside from these locations. We call the selected disk the *recovery target*, and the locations of the buddies that help to rebuild the *recovery sources*. The recovery target chosen from the candidate list (a) must be alive, (b) should not contain already a buddy from the same group, and (c) must have sufficient space. Additionally, it should currently have sufficient bandwidth, though if there is no better alternative, we will stick to it. If we use S.M.A.R.T. (Self Monitoring and Reporting Technology) [18] or a similar system to monitor the health of disks, we are able to avoid unreliable disks. Unlike FARSITE [9], replicas are not moved once placed.

Even with S.M.A.R.T., the possibility that a recovery target fails during the data rebuild process remains. In this case, we merely choose an alternative target. If a recovery source fails, and there is no alternative, a data loss occurs. Otherwise, we replace the failed source with an alternative one. The occurrence of this problem, which we call *recovery redirection* is rare. We found that, at worst, it happened to fewer than 8.0% of our systems even once during simulated six years.

2.3. Reliability Factors

The reliability of a large storage system employing FARM depends on the size and structure of redundancy groups, the size of the blocks, the latency of disk failure detection, the bandwidth utilized for recovery, the amount of data stored on disks, the number of disks in the system, and the way we replace disks.

Two inherent factors affect the probability of data loss: the total number of the redundancy groups across the system and the size of a single redundancy group. Our previous study [37] showed that these two factors balance each other out in the case of two-way mirroring, and that the data loss probability is independent of the redundancy group size under the idealizing assumption of zero failure detection time and independent redundancy groups.

Strategies for efficient failure detection are beyond the scope of this paper; we merely measure the impact of failure detection latencies, which add to the rebuild times. The speed of a rebuild depends also on the data transfer rate for

Table 1. Disk failure rate per 1000 hours [10].

Period (month)	0-3	3-6	6-12	12-72
failure rate	0.5%	0.35%	0.25%	0.2%

Table 2. Parameters for a petabyte-scale storage system.

Parameter	Base Value	Examined Value
total data in the system	2 PB	0.1–5 PB
size of a redundancy group	10 GB, 50 GB	1–100 GB
group configuration	two-way mirroring	varied
latency to failure detection	300 sec.	0–3600 sec.
disk bandwidth for recovery	16 MB/sec	8–40 MB/sec

the rebuild. This recovery bandwidth is not fixed in a large storage system. It fluctuates with the intensity of user requests, especially if we exploit system idle time [14] and adapt recovery to the workload.

Storage overhead in a large system is costly. At \$1/GB, the difference between two- and three-way mirroring amounts to millions of dollars in a petabyte-scale storage system. For this reason, we investigated disk space utilization.

FARM is a general approach to combat disk failures in large storage systems. As storage systems grow, relatively rare failure types become statistically significant. FARM might not make a difference for small systems, but is needed for large-scale systems.

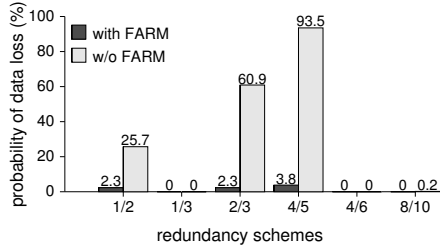
Finally, the batch size—the number of new disks introduced into the system at any one time—has an effect on reliability. When new disk drives are introduced to the system, data should be migrated from old disks to the new ones. The number of the replacement processes determines the frequency of data reorganization and also affects system reliability, because of the possible failures in a batch.

3. Experimental Results

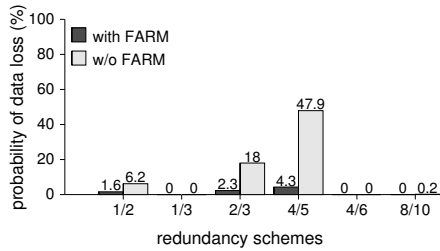
Since building a petabyte-scale real large system is expensive, and running multi-year reliability tests is impractical at best, we ran our experiments using discrete event-driven simulations built with the PARSEC simulation tool [24].

3.1. System Assumptions

Our experiments explored the behavior of a two-petabyte (PB) storage system, except as noted. Depending on the redundancy scheme employed, the system contains up to 15,000 disk drives, each with an extrapolated capacity of 1 TB and an extrapolated sustainable bandwidth of



(a) redundancy group size = 10 GB.



(b) redundancy group size = 50 GB.

Figure 3. Reliability comparisons of systems with and without FARM data protection, assuming that latency to failure detection is zero (1000 runs for each). m/n schemes are discussed in Section 2.1.2.

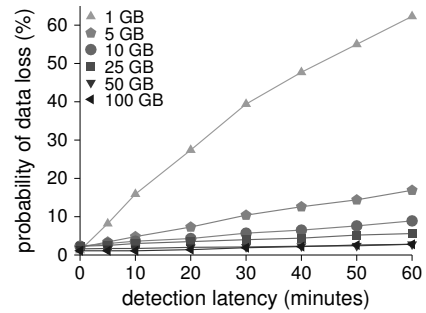
80 MB/sec (based on the 56 MB/sec of the current IBM Deskstar [8]). We assume that recovery can use at most 20% of the available disk bandwidth, and that each device reserves no more than 40% of its total capacity at system initialization for recovered data. We define the size of a redundancy group to be the amount user data stored in it, and vary this amount from 1 GB to 100 GB.

It is well-known that disks do not fail at a constant rate; the failure rates are initially high, then decrease gradually until disks reach their End Of Design Life (EODL). The industry has proposed a new standard for disk failure distribution [10, 35]. We assume our disk drives have a typical EODL of 6 years and follow Elerath [10] for the failure rates enumerated in Table 1.

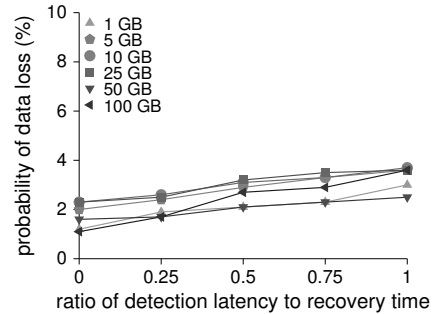
Table 2 lists the default values of the system parameters together with the range of values that we used to quantify the tradeoffs in the design of our system.

3.2. Reliability Improvement

We first explored how much system reliability FARM would gain. We constructed redundancy groups using two-way and three-way mirroring (1/2 and 1/3), two RAID 5 schemes (2/3 and 4/5), and two ECC configurations (4/6 and 8/10), respectively. We then compared 2 PB systems built using the base parameters in Table 2, both with and without FARM, assuming the latency to failure detection is zero. We further varied the size (usable capacity) of the re-



(a) The effect of detection latency on the probability of data loss.



(b) The effect of the ratio of detection latency to recovery time on the probability of data loss.

Figure 4. The effect of latency to detect disk failures on overall system reliability under various redundancy group sizes.

dundancy group between 10 GB and 50 GB. We simulated our system for six years. At the end of six years, the remaining disks would be near the end of their lives and be ready to be replaced.

Our results, shown in Figure 3, demonstrate that FARM always increases reliability. RAID 5-like parity without FARM fails to provide sufficient reliability. With two-way mirroring, FARM reduces probability of data loss down to 1–3%, as compared to 6–25% without FARM. 3-way mirroring limits the probability of data loss to less than 0.1% during the first six years, similar to the probability of 4/6 and 8/10 with FARM.

Figures 3(a) and 3(b) show that the size of redundancy groups has little impact on systems using FARM, but *does* matter for systems without FARM. Our earlier study [37] found that data loss probability is independent of group size under two-way mirroring with FARM, if the latency of failure detection is zero. Without FARM, reconstruction requests queue up at the single recovery target. Data loss occurs if any of the recovery sources or their alternatives fail before the block is reconstructed. With smaller redundancy groups, there are more recovery sources that can fail during that time, so that the probability of data loss increases.

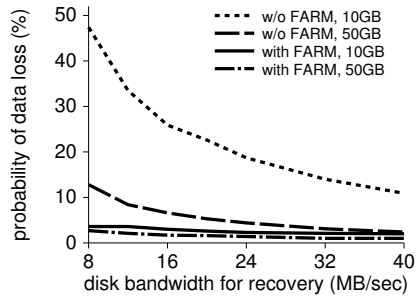


Figure 5. System reliability at various levels of recovery bandwidth with size of redundancy groups varies as 10 GB and 50 GB, under FARM and traditional recovery scheme, respectively.

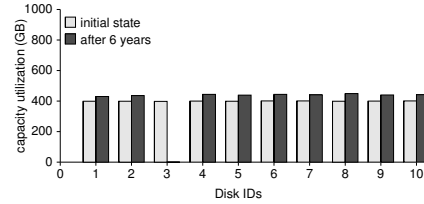
3.3. Latency of Failure Detection

After a disk fails, we first need to identify the failed disk and then reconstruct the data that was on it. The latency of detecting a failure and the time of constructing data constitute the window of vulnerability. Discovering failure in such a large system is not trivial and failure detection latency cannot be neglected. When we investigated its impact on systems with redundancy group sizes ranging from 1 GB to 100 GB under two-way mirroring plus FARM, we found that systems with smaller group sizes are more sensitive (Figure 4(a)). It takes less time to reconstruct smaller-sized groups, so a constant failure detection latency makes up a much larger relative portion of the window of vulnerability. For example, it takes 64 seconds to reconstruct a 1 GB group, assuming reconstruction runs at a bandwidth of 16 MB/sec, while it takes 6400 seconds for a 100 GB group. If it takes 10 minutes to detect a failure, detection latency represents 90.4% of the window of vulnerability for the former, and only 0.86% for the latter. We hypothesized that the *ratio* of failure detection latency to actual data recovery time determines the probability of data loss. Our results, summarized in Figure 4(b), show that this is the case.

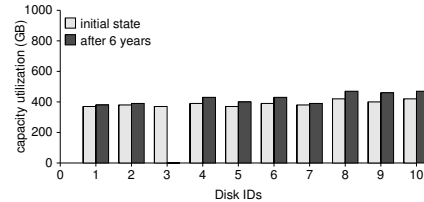
3.4. Disk Bandwidth Usage for Recovery

Data rebuild time can be shortened by allocating a higher portion of device bandwidth to recovery. To gauge the impact of recovery on usable bandwidth, we examined various disk bandwidths contributed to data recovery in a 2 PB storage system with two-way mirroring under the assumption that failure detection latency is 300 seconds.

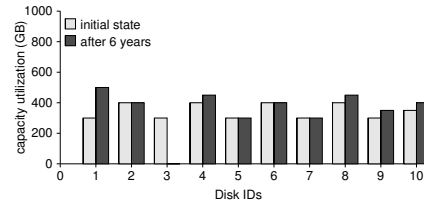
As expected, the probability of data loss decreases as the recovery bandwidth increases (Figure 5). In all cases,



(a) redundancy group size = 1 GB.



(b) redundancy group size = 10 GB.



(c) redundancy group size = 50 GB.

Figure 6. Disk utilization for ten randomly selected disks. Utilization was measured at the start of the simulation period and at the end of the six year period. The size of redundancy groups is varied as 1 GB, 10 GB and 50 GB.

we observed that the chance of data loss is higher for a smaller group size, due to the impact of failure detection latency. High recovery bandwidth improves system reliability dramatically for the systems without FARM, but does not have a pronounced effect when FARM is used. The advantage of higher recovery bandwidth is to reduce the data rebuild time, but FARM has already cut recovery time dramatically, so that further reductions from higher bandwidth utilization achieve little improvement. Without FARM, recovery time is quite long due to the single recovery target, so high recovery bandwidth can greatly improve reliability. In systems where disks are much less reliable, or storage capacity exceeds petabytes, high recovery bandwidth can be effective even when FARM is used.

3.5. Disk Space Utilization

FARM distributes both data and redundancy information across the whole system. As disks fail, data stored on them is redistributed to the rest of the system; it is never recollected to a single disk. This approach has the potential

statistical values	1 GB		10 GB		50 GB	
	initial state	six years later	initial state	six years later	initial state	six years later
mean	400 GB	442.33 GB	400 GB	442.33 GB	400 GB	442.33 GB
standard deviation	1.41 GB	6.44 GB	18.03 GB	26.41 GB	81.52 GB	92.53 GB

Table 3. Mean and standard deviation of disk utilization in the system initial state and after the end of disk lifetime (six years). Redundancy groups are configured as 1 GB, 10 GB and 50 GB, respectively.

to introduce imbalances in the actual amount of data stored on each individual drive. However, our technique does not suffer from this problem, as demonstrated by an experiment summarized in Figure 6. We first used our placement algorithm to distribute data on 10,000 1 TB disks with an average utilization of 40%, including both primary and mirror copies of data. We then simulated disk failures and data reconstruction for six years of simulated time; the mean and standard deviation of capacity utilization are listed in Table 3. Smaller-sized redundancy groups result in a lower standard deviation on capacity, although the mean values stay the same. Figure 6 shows the load for ten randomly-chosen disk drives both before and after the six years of service. Disk 3 failed during the service period so it does not carry any load. The other nine disk drives have increased their disk space usage due to the distributed redundancy created by FARM. The unevenness in data distribution is caused by the relatively large ratio of redundancy group size to disk size. Reducing redundancy group size to 1 GB would alleviate this problem and balance disk utilization better, but at the cost of lower reliability, as described in Section 3.2.

3.6. Disk Drive Replacement

Large-scale storage systems are always dynamic: old disk drives are removed when they fail or retire, and new disk drives are added to satisfy the demands of disk replacement and data capacity growth. It is typically infeasible to add disk drives one by one into large storage systems because doing so would require daily (if not more frequent) drive replacement. Instead, a cluster of disk drives, called a *batch*, is added. The choice of batch size determines the replacement frequency and the amount of data that needs to migrate onto new drives in order to keep the system in balance.

The newly-added disks come from different disk vintages and have various storage capacities. The reorganization of data should be based on the *weight* of disks, determined by disk vintage, reliability properties, and capacity. The determination of these weights is not within the scope of the paper; currently, the weight of each disk is set to that of the existing drives for simplicity.

Large batch sizes can have a negative impact on system

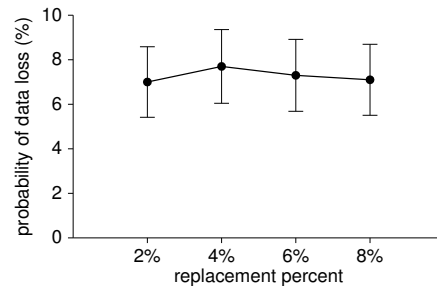
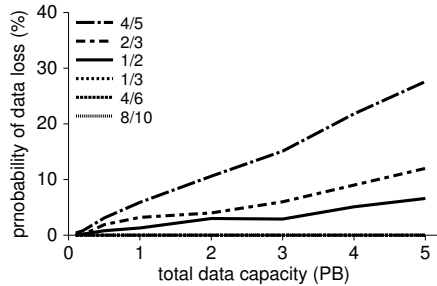


Figure 7. Effect of disk drive replacement timing on system reliability, with 95% confidence intervals. New disks are added in the system after losing 2%, 4%, 6%, and 8% of the total disks. The size of redundancy groups is 10 GB.

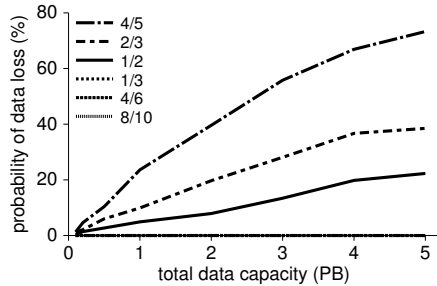
reliability because they introduce a large number of new, and hence more failure-prone, disks into the system. We call this the *cohort effect*. We experimented by replacing failed disk drives once the system has lost 2%, 4%, 6%, and 8% of the total number of disk drives. As Figure 7 reports, the cohort effect is not visible using a redundancy group size of 10 GB, in large part because only about 10% of the disks fail during the first six years. As a result, disk replacement happens about five times at the batch size of 2% and about once at 8%, assuming that total data capacity remains unchanged. The number of disks in each replacement batch is 200 for 2% and 800 for 8%, so that only 2% and 8% of the data objects migrate to newly-added disks. This number is too small for the cohort effect, so batch size and replacement frequency does not significantly affect system reliability. Thus, there is little benefit beyond delaying some cost to just-in-time replacement.

3.7. System Scale

Our findings apply not only to a 2 PB system, but to any large-scale storage system. As expected, the probability of data loss, shown in Figure 8(a), tends to increase approximately at a linear rate as system scales from 0.1 PB to 5 PB. For a 5 PB storage system, FARM plus two-way mirroring achieves a data loss probability as low as 6.6%.



(a) Disks with the failure rate listed in Table 1.



(b) Disks with a failure rate twice that listed in Table 1.

Figure 8. The probability of data loss in a storage system under FARM is approximately linear in the size of the storage system. The size of redundancy groups is 10 GB.

However, RAID 5-like parity cannot provide enough reliability even with FARM. Using a 3-way mirroring, 6 out of 8, or 8 out of 10 scheme with FARM, the probability of data loss is less than 0.1%. This result is not unexpected—it is well-known that a system with twice as many disks is approximately twice as likely to fail given otherwise identical parameters.

Disk vintage [10] is an important aspect in system reliability. Disk drives with various vintages differ in failure rates and even failure modes. We set up disk drives with failure rates as twice high as the disk vintage listed in Table 1 and vary the system scale. We observed a similar trend in increase of data loss probability from the results shown in Figure 8(b). As the failure rate of individual drives doubled and the rest of the configuration stayed the same, the probability of data loss more than doubled. This is due to several factors, including an increased likelihood of failure when one or both disks are new. We believe that keeping disk failure rates low is a critical factor in ensuring overall system reliability because system reliability decreases at a rate faster than individual disk reliability decreases.

4. Related Work

System designers have long tried to build more reliable storage systems. Techniques such as disk mirroring [3] and RAID (Redundant Arrays of Independent Disks) [6] have been used for many years to improve both system reliability and performance. The use of more powerful erasure codes for RAID [5, 27, 32] can improve reliability to the point where it may be sufficient for a multi-petabyte file system, but the overhead of using such erasure codes will likely reduce system performance.

Traditionally, system designers were more concerned with system performance during recovery than they were with reliability, since smaller systems can be highly reliable even with relatively simple redundancy mechanisms. To address the problem of reduced performance during recovery from a failure, Menon and Mattson [23] proposed “distributed sparing,” in which the spare disk is broken up and distributed through the array. In such an array, the reads for a data rebuild are distributed to all disks, and the disk itself is “rebuilt” onto the spare space available in the array, distributing the recovery load to all of the disks in the system and reducing the performance penalty of recovery. Muntz and Lui [25] proposed that a disk array of n disks be declustered by grouping the blocks in the disk array into reliability sets of size g . Later, Alvarez, *et al.* [2] developed DATUM, a method that can tolerate multiple failures by spreading reconstruction accesses uniformly over disks based on information dispersal as a coding technique. DATUM accommodates distributed sparing as well. The principle of these ideas comes close to the spirit of fast recovery schemes. However, our fast recovery mechanisms are not only used for avoiding performance degradation under disk drive failures but also, more importantly, for improving reliability. In addition, the previous studies did not use a bathtub curve as disk failure rates, reducing the accuracy of their experiments.

There has been some research beyond RAID in reliability and recovery for large-scale systems, though most has focused on the use of storage in wide-area systems. OceanStore [28] is an infrastructure that provides high availability, locality, and security by supporting nomadic data. It uses erasure coding to provide redundancy without the overhead of strict replication and is designed to have a very long mean time to data loss. An analysis of the reliability of OceanStore [36] showed that erasure codes had higher reliability than pure redundancy for a given amount of storage overhead; however, the study did not explore general characteristics of large-scale storage systems. Other peer-to-peer systems, such as Pangaea [31], PAST [30], CFS [7], and Gnutella [29] were designed with high replication since they are limited to read-only data. However, in read/write file systems, very large replication

factors would not be practical because of both storage efficiency and overhead necessary to maintain data consistency. Ivy [26], a read/write peer-to-peer system, does not address replication issues.

In the Google file system [12], a single master makes decisions on data chunk placement and replications. A data chunk is re-replicated when its number of replicas falls below a limit specified by users. Grid DataFarm [34] stores files as replicated fragments in distributed storage resources. We consider more general redundancy schemes like erasure coding. In FARSITE [1], a directory group ensures that the files they store are never lost, again restricting replication for a given block to a particular group of machines. FARSITE assumes that recovery is sufficiently fast that data is never lost; in a multi-petabyte file system, this assumption may simply not hold.

Petal [20], a distributed storage system that is highly available and easy to manage, uses chained-declustering data placement scheme and provides high failure tolerance. Recently, FAB, federated array of bricks [11], extended Petal with better replication and load balancing algorithms. FAB provides reliability by grouping data segments together and replicating the groups over several bricks. The management of groups in FAB comes close to that of redundancy groups in our system. However, they did not address the various issues in dynamic storage systems such as failure detection latency and disk bandwidth.

5. Conclusions

Traditional redundancy schemes can not guarantee sufficient reliability for petabyte-scale storage systems. Simply increasing the number of replicas is cumbersome and costly for read/write systems, and using m out of n schemes, while cheaper, is also more complex. In response, we developed FARM, a fast recovery scheme that distributes redundancy on demand effectively to improve reliability. Data objects are clustered into redundancy groups that may use various replication and erasure-coding schemes. We have demonstrated that FARM provides much higher reliability for large-scale storage systems. Our results show that failure detection latency affects reliability greatly for small redundancy groups, and that the ratio of recovery time to failure detection latency is crucial. We found that allocating a higher portion of bandwidth to recovery does not have a pronounced influence for FARM because data rebuild times have already been greatly reduced by FARM. We have validated FARM for general large-scale storage systems of different sizes.

Our work offers a more effective redundancy mechanism for designers of large-scale storage systems, especially those that must provide high data reliability. By developing more effective redundancy techniques for large-

scale storage systems and quantifying the effects of different parameters on the reliability of such systems, we have provided designers of petabyte-scale systems the techniques they need to build the high-performance, high-capacity systems demanded by scientific computation and “computing utilities.” By applying these techniques, designers can ensure that their systems feature high reliability as well as high performance.

Acknowledgments

We would like to thank the members of the Storage Systems Research Center for their help and guidance. Qin Xin and Ethan Miller were supported in part by Lawrence Livermore National Laboratory, Los Alamos National Laboratory, and Sandia National Laboratory under contract B520714. Thomas Schwarz was supported in part by IBM Research Grant 41102-COEN-RSCH-IG-IG09.

References

- [1] A. Adya, W. J. Bolosky, M. Castro, R. Chaiken, G. Cermak, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, and R. Wattenhofer. FARSITE: Federated, available, and reliable storage for an incompletely trusted environment. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI)*, Boston, MA, Dec. 2002. USENIX.
- [2] G. A. Alvarez, W. A. Burkhard, and F. Cristian. Tolerating multiple failures in RAID architectures with optimal storage and uniform declustering. In *Proceedings of the 24th International Symposium on Computer Architecture*, pages 62–72, Denver, CO, June 1997. ACM.
- [3] D. Bitton and J. Gray. Disk shadowing. In *Proceedings of the 14th Conference on Very Large Databases (VLDB)*, pages 331–338, 1988.
- [4] M. Blaum, J. Brady, J. Bruck, and J. Menon. EVENODD: An efficient scheme for tolerating double disk failures in RAID architectures. *IEEE Transactions on Computers*, 44(2):192–202, 1995.
- [5] W. A. Burkhard and J. Menon. Disk array storage system reliability. In *Proceedings of the 23rd International Symposium on Fault-Tolerant Computing (FTCS '93)*, pages 432–441, June 1993.
- [6] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson. RAID: High-performance, reliable secondary storage. *ACM Computing Surveys*, 26(2), June 1994.
- [7] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with CFS. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01)*, pages 202–215, Banff, Canada, Oct. 2001. ACM.
- [8] Disk drive specification: IBM Deskstar TM 180GXP hard disk drives.

- [9] J. R. Douceur and R. P. Wattenhofer. Large-scale simulation of replica placement algorithms for a serverless distributed file system. In *Proceedings of the 9th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '01)*, pages 311–319, Cincinnati, OH, Aug. 2001. IEEE.
- [10] J. G. Elerath. Specifying reliability in the disk drive industry: No more MTBF's. In *Proceedings of the 2000 Annual Reliability and Maintainability*, pages 194–199. IEEE, 2000.
- [11] S. Frølund, A. Merchant, Y. Saito, S. Spence, and A. Veitch. FAB: Enterprise storage systems on a shoestring. In *Proceedings of the 9th Workshop on Hot Topics in Operating Systems (HotOS-IX)*, Kauai, HI, May 2003.
- [12] S. Ghemawat, H. Gobioff, and S.-T. Leung. The Google file system. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP '03)*, Bolton Landing, NY, Oct. 2003. ACM.
- [13] G. A. Gibson, D. F. Nagle, K. Amiri, J. Butler, F. W. Chang, H. Gobioff, C. Hardin, E. Riedel, D. Rochberg, and J. Zelenka. A cost-effective, high-bandwidth storage architecture. In *Proceedings of the 8th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 92–103, San Jose, CA, Oct. 1998.
- [14] R. Golding, P. Bosch, C. Staelin, T. Sullivan, and J. Wilkes. Idleness is not sloth. In *Proceedings of the Winter 1995 USENIX Technical Conference*, pages 201–212, New Orleans, LA, Jan. 1995. USENIX.
- [15] L. Hellerstein, G. A. Gibson, R. M. Karp, R. H. Katz, and D. A. Patterson. Coding techniques for handling failures in large disk arrays. *Algorithmica*, 12:182–208, 1994.
- [16] J. L. Hennessy and D. A. Patterson. *Computer Architecture—A Quantitative Approach*. Morgan Kaufmann Publishers, 3rd edition, 2003.
- [17] R. J. Honicky and E. L. Miller. Replication under scalable hashing: A family of algorithms for scalable decentralized data distribution. In *Proceedings of the 18th International Parallel & Distributed Processing Symposium (IPDPS 2004)*, Santa Fe, NM, Apr. 2004. IEEE.
- [18] G. Hughes, J. Murray, K. Kreutz-Delgado, C. Elkan, and W. Tran. Improved disk-drive failure warnings. In *IEEE Transactions on Reliability*, 2000.
- [19] http://www.archive.org/web/researcher/data_available.php.
- [20] E. K. Lee and C. A. Thekkath. Petal: Distributed virtual disks. In *Proceedings of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 84–92, Cambridge, MA, 1996.
- [21] W. Litwin and T. Schwarz. LH*_{RS}: A high-availability scalable distributed data structure using Reed Solomon codes. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 237–248, Dallas, TX, May 2000. ACM.
- [22] P. Lyman, H. R. Varian, P. Charles, N. Good, L. L. Jordan, and J. Pal. How much information? 2003. <http://www.sims.berkeley.edu/research/projects/how-much-info-2003/>.
- [23] J. Menon and R. L. Mattson. Distributed sparing in disk arrays. In *Proceedings of Comcon '92*, pages 410–416, Feb. 1992.
- [24] R. A. Meyer and R. Bagrodia. PARSEC user manual, release 1.1. <http://pcl.cs.ucla.edu/projects/parsec/>.
- [25] R. R. Muntz and J. C. S. Lui. Performance analysis of disk arrays under failure. In *Proceedings of the 16th Conference on Very Large Databases (VLDB)*, pages 162–173, 1990.
- [26] A. Muthitacharoen, R. Morris, T. M. Gil, and B. Chen. Ivy: A read/write peer-to-peer file system. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI)*, Boston, MA, Dec. 2002.
- [27] J. S. Plank. A tutorial on Reed-Solomon coding for fault-tolerance in RAID-like systems. *Software—Practice and Experience (SPE)*, 27(9):995–1012, Sept. 1997. Correction in James S. Plank and Ying Ding, Technical Report UT-CS-03-504, U Tennessee, 2003.
- [28] S. Rhea, P. Eaton, D. Geels, H. Weatherspoon, B. Zhao, and J. Kubiatowicz. Pond: the OceanStore prototype. In *Proceedings of the 2003 Conference on File and Storage Technologies (FAST)*, pages 1–14, Mar. 2003.
- [29] M. Ripeanu, A. Iamnitchi, and I. Foster. Mapping the Gnutella network. *IEEE Internet Computing*, 6(1):50–57, Aug. 2002.
- [30] A. Rowstron and P. Druschel. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01)*, pages 188–201, Banff, Canada, Oct. 2001. ACM.
- [31] Y. Saito, C. Karamanolis, M. Karlsson, and M. Mahalingam. Taming aggressive replication in the Pangaea wide-area file system. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI)*. USENIX, Dec. 2002.
- [32] T. J. Schwarz. Generalized Reed Solomon codes for erasure correction in SDDS. In *Workshop on Distributed Data and Structures (WDAS 2002)*, Paris, Mar. 2002.
- [33] T. J. Schwarz and W. A. Burkhard. Reliability and performance of RAID5. *IEEE Transactions on Magnetics*, 31(2):1161–1166, 1995.
- [34] A. Takefusa, O. Tatebe, S. Matsuoka, and Y. Morita. Performance analysis of scheduling and replication algorithms on grid datafarm architecture for high energy physics applications. In *Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing (HPDC)*, pages 34–43, Seattle, WA, June 2003. IEEE.
- [35] The International Disk Drive Equipment & Materials Association (IDEMA). R2-98: Specification of hard disk drive reliability.
- [36] H. Weatherspoon and J. Kubiatowicz. Erasure coding vs. replication: A quantitative comparison. In *Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS 2002)*, Cambridge, Massachusetts, Mar. 2002.
- [37] Q. Xin, E. L. Miller, T. J. Schwarz, D. D. E. Long, S. A. Brandt, and W. Litwin. Reliability mechanisms for very large storage systems. In *Proceedings of the 20th IEEE / 11th NASA Goddard Conference on Mass Storage Systems and Technologies*, pages 146–156, Apr. 2003.