Disaster Recovery Codes: Increasing Reliability with Large-Stripe Erasure Correcting Codes

Kevin M. Greenan
Computer Science Dept.
Univ. of California, Santa Cruz
Santa Cruz, CA, USA
kmgreen@cs.ucsc.edu

Ethan L. Miller
Computer Science Dept.
Univ. of California, Santa Cruz
Santa Cruz, CA, USA
elm@cs.ucsc.edu

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

Darrell D. E. Long
Computer Science Dept.
Univ. of California, Santa Cruz
Santa Cruz, CA, USA
darrell@cs.ucsc.edu

ABSTRACT

Large-scale storage systems need to provide the right amount of redundancy in their storage scheme to protect client data. In particular, many high-performance systems require data protection that imposes minimal impact on performance; thus, such systems use mirroring to guard against data loss. Unfortunately, as the number of copies increases, mirroring becomes costly and contributes relatively little to the overall system reliability. Compared to mirroring, parity-based schemes are space-efficient, but incur greater update and degraded-mode read costs. An ideal data protection scheme should perform similarly to mirroring, while providing the space efficiency of a parity-based erasure code.

Our goal is to increase the reliability of systems that currently mirror data for protection without impacting performance or space overhead. To this end, we propose the use of large parity codes across two-way mirrored reliability groups. The secondary reliability groups are defined across an arbitrarily large set of mirrored groups, necessitating a small amount of non-volatile RAM for parity. Since each parity element is stored in non-volatile RAM, our scheme drastically increases the mean time to data loss without impacting overall system performance.

Categories and Subject Descriptors

D.4.2 [**Software**]: Storage Management—*Secondary storage*; D.4.5 [**Software**]: Operating Systems—*reliability*

General Terms

Reliability

Keywords

erasure coding, storage reliability

StorageSS'07, October 29, 2007, Alexandria, Virginia, USA.

1. INTRODUCTION

Storage systems use redundancy in order to store data reliably. High performance storage systems, such as Ceph [12] and GPFS [9] use mirroring to store client data over possibly thousands of disks. While mirroring is well suited for high-performance workloads, full data replication results in a great deal of overhead and contributes little to the overall probability of not losing data during the economic lifespan of the file system [7]. For instance, 2-way mirroring could yield two or three nines of data survivability, but pushing this number to four or five nines requires much higher redundancy. Systems such as Oceanstore [6] and FAB [8] have the ability to provide higher redundancy and pay less in terms of space consumption using parity-based erasure correcting codes for data protection. These systems trade space efficiency for the read-write-modify update and degraded-mode read penalties incurred when using parity-based protection schemes.

One quality most mirrored and erasure-encoded systems share is a single level of replication; trading off either space efficiency or performance in the process. Our aim is to maintain the performance advantages of a mirrored system while providing very high reliability. Instead of providing additional redundancy using a single parity-based encoding, we propose a two-level scheme that adds a second layer of redundancy; resulting in data protection that can handle correlated and massive failures without incurring unacceptable space and performance overhead. Our scheme creates primary redundancy groups using two-way mirroring, which enables fast recovery in the case of single disk failure. A second, erasure-encoded redundancy group is computed across a large set of primary copies of mirrored data. The second layer of redundancy augments a traditional mirroring scheme with additional protection; making recovery attainable in the case of massive or correlated failures.

In an effort to minimize the parity update costs across the second layer of reliability groups, we store all parity data in non-volatile RAM (NVRAM), which is distributed throughout the storage devices. By creating very large stripes in the second layer of redundancy, our solution only requires a small fraction of NVRAM. For example, suppose a single data strip encompasses 1 GB and a single parity strip is computed from 1000 data strips. Such a configuration would result in 1 MB of parity for every gigabyte of data, or $\frac{1}{1000}$ parity overhead. We chose this number because currently, the price of NVRAM, such as compact flash, is less than 1000 times that of disk measured in $/GB. Since NVRAM can be updated at a much higher rate, our solution is cheaper and faster than adding

disk space to store additional parity data for a much smaller reliability stripe. In addition, we find that data reliability increases substantially even when using single parity across mirrored groups.

The main contribution of this paper is the addition and analysis of large parity groups across mirrored reliability groups. We believe that high-performance storage systems that currently rely on mirroring for reliability will benefit from the extra redundancy, without compromising system performance. As our preliminary results show, our scheme results in much higher reliability than 2-way mirroring. Since this work concentrates on reliability, we postpone a detailed performance analysis.

2. OVERVIEW

Our scheme is not specific to any single system; however, for clarity, we present our analysis in a system similar to Ceph [12]. The system stores client data in fixed-sized blocks, called *buckets*. The buckets themselves are objects assigned to object-based storage devices (OSD), which are assumed to contain at least a disk and some NVRAM. All client data is stored in *data buckets* on disk, while parity data is placed in *parity buckets* on NVRAM.

Our storage system stores client data in buckets of size approximately 1 GB, so each device stores about $1,000$ data buckets. The storage system consists of about $10,000$ devices and hence stores about 10^7 buckets, for a total of 10 PB of raw storage. In the future, we expect the size of storage installations to increase and the capacity of disk drives to increase, though perhaps not at the historical rate of 60% to 80%. Our purpose in giving these numbers is to give a more concrete picture of the system, but we expect the utility of our approach to increase with larger systems.

For our analysis, we assume the system uses mirroring to protect buckets against device failure. Each bucket is stored on two devices selected in a pseudo-random manner using algorithms similar to RUSH [5]. A bucket is considered lost if the two devices are lost or sector failures corrupt both copies of the bucket. Buckets are addressed by a simple number and might migrate during the lifetime of the system as new devices are added to the system and old ones are removed because of failure or obsolescence.

Assume that a disk fails every 5 years. Because of the distributed nature of our storage devices, the system is reconfigured and protected much faster than the several hours it takes to read a disk completely. In such a system 30 minutes is a very conservative estimate for reconfiguration time, even including time to detect the failure. This means that each bucket is vulnerable to single OSD failure for 30 minutes / 5 years = 10^{-5} of its life. Thus, a bucket is lost at a probability of 10^{-10} times the lifespan of its data. Unfortunately, there are many buckets, leading to a much greater likelihood of data loss.

We increase the already excellent survival chances of mirrored data by maintaining additional parity data stored in a NVRAM calculated over a reliability stripe of thousands of disks. In the rare case of a bucket annihilation on disk (e.g. loss of both replicas), we shut down the system and use the NVRAM based parity together with the other buckets in the same reliability group to reconstruct the missing buckets. The calculation is tedious (involving reading thousands of disks) but highly likely to succeed.

We propose a technique that calculates NVRAM parity data over a large set of data buckets. Each bucket in the system is assigned to a single reliability stripe with R buckets total to which we add 1, 2 or 3 NVRAM parity buckets. In each of the three scenarios, we calculate the parity using a maximum distance separable (MDS) code. A (n,k) MDS code that computes $n-k$ parity buckets over k data buckets can sustain the failure of any $n-k$ buckets. We assume that an XOR-parity scheme is used to calculate single parity, while

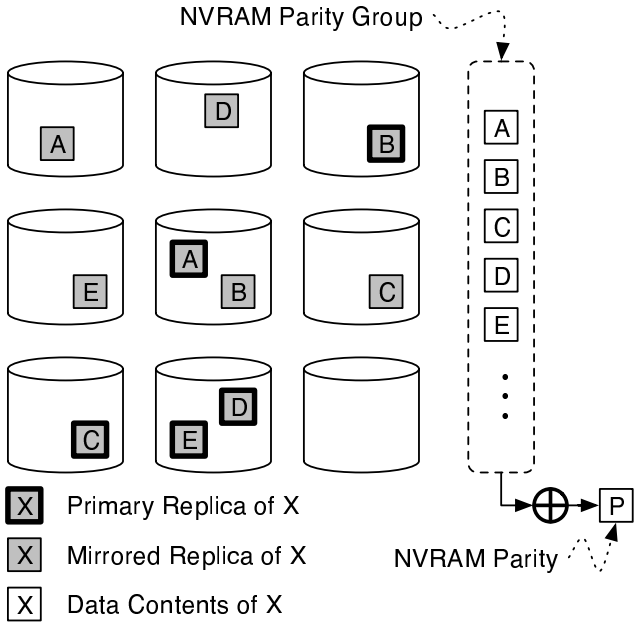


Figure 1: *The layout of our disaster recovery encoding scheme. Primary and mirrored replicas are distributed such that no two replicas of a single data bucket are stored on the same disk. The data contents of each object contribute to the parity of a single parity group, achieving another level of redundancy. We assume that at least one physical copy of each data element in a parity group exists on a distinct disk.*

	Disk Overhead	NVRAM Overhead
Mirror	$\frac{N}{2}$ *TB*	0
Mirror+P	$\frac{N}{2}$ *TB*	$\left(P \cdot \frac{1}{R} \cdot \frac{N}{2} \cdot B\right)$ *GB*

Table 1: *Storage overhead for each reliability scheme given N 1 TB OSDs, B 1 GB buckets per OSD, secondary reliability groups with R data buckets and P parity elements per group.*

Reed-Solomon codes are used for double and triple parity.

Figure 1 shows how both mirrored replicas and parity groups are defined in our system. Each *data bucket* is stored in two physical locations, as the *primary replica* and *mirrored replica*. Each replica must be stored on two distinct OSDs. The contents of each data bucket also contribute to a single *parity group*, with the constraint that each data element in a parity group must exist on a distinct OSD. This constraint is easily maintained if we consider the physical location of the primary and mirrored replica of a data bucket. The data elements of a parity group are used to compute a parity element, which is stored in distributed NVRAM banks across the OSDs themselves.

Each parity group protects R data buckets from data loss. If a data bucket's corresponding primary and mirrored replicas are lost due to disk failure, the remaining data buckets and parity buckets in the parity group are used to reconstruct the lost bucket. Once the lost data bucket is reconstructed, the primary and mirrored replica are redistributed. Reconstruction during whole OSD failure, though more complicated and time consuming, works in a similar fashion. As long as each parity group uses an (n,k) MDS code, our scheme can tolerate any $n-k$ lost data buckets.

Figure 2 shows the overhead associated with 2-way mirroring and parity groups over the mirrored buckets, assuming $1,000$ (1 GB) buckets per OSD, $10,000$ OSDs and $3,000$ buckets per parity group.

If the parity groups utilize an encoding that adds a single parity element, then the total storage overhead is roughly 5,000 TB of disk redundancy and 1.63 TB of NVRAM redundancy. Since the NVRAM storage is distributed throughout the system, placing 1 GB of NVRAM on each OSD should be sufficient. Also note that the storage overhead in our example (excluding 2-way mirroring overhead) is 1.63 TB for every 5,000 TB of data, or 0.03%.

An issue arises when using a Reed-Solomon code to calculate the additional parity. In most fast software implementations of Reed-Solomon, a Galois field with 256 elements is used to calculate parity. Unfortunately, this field will only support 257 buckets per reliability group. Following byte boundaries, the next largest field, $GF(2^{16})$, contains $65,536$ elements, which is sufficient for our purposes. We have developed an efficient implementation of $GF(2^{16})$ and larger fields [4].

3. RELIABILITY ANALYSIS

Intuitively, the addition of NVRAM parity would be expected to lead to much higher data reliability than standalone mirroring. We quantify and compare the reliability of the mirrored and NVRAM parity schemes using both probabilistic and stochastic analysis. The analysis shows that augmenting a mirrored system with NVRAM parity results in a substantial increase in reliability.

The target storage system in our analysis contains a total of N OSDs, though strictly speaking, N varies due to failure and replacement. We assume that B buckets are stored on each device, resulting in $N \cdot B$ buckets of storage in the entire system. Each data bucket is mirrored; thus, the system stores $\frac{N \cdot B}{2}$ buckets of data. We distinguish between physical buckets and data buckets, where the latter is stored in two physical buckets. In the following, we simply say bucket for a data bucket.

3.1 Probabilistic Analysis

By design, we never store the two physical buckets of a (data) bucket on the same OSD. Assume now that k OSDs have failed in a Ceph-like system that relies on 2-way mirroring for data reliability. With probability $p(k) = \frac{\binom{k}{2}}{\binom{N}{2}}$, a given bucket is located on these k OSDs and hence lost. Then, with probability $q(k) = 1 - p(k)$, the failures have not led to data loss. Given k failures and no NVRAM parity, the system survival probability is $Q_{0\mathrm{P}}(k) = q(k)^{\frac{N \cdot B}{2}}$; the failure probability is $1 - Q_{0\mathrm{P}}(k)$.

Suppose a 2-way mirrored scheme is augmented with the NVRAM parity scheme by adding n parity buckets to each group containing R data buckets. This allows the whole group to survive up to n bucket failures—situations where *both* mirrors of a bucket are lost. Given k failures, the probability that $l \leq k$ of the n NVRAM parity buckets in a single group are located on a failed OSD is $\frac{\binom{n}{l}\binom{N-n}{k-l}}{\binom{N}{k}}$. Since there are $n-l$ available NVRAM parity buckets in the group, the chance of data survival is the probability that at most $n-l$ of the R data buckets are unavailable; given by the cumulative binomial distribution. The survival probability of a single parity group is calculated as

$$Q(n,R,k) = \sum_{l=0}^{n} \left(\frac{\binom{n}{l} \cdot \binom{N-n}{k-l}}{\binom{N}{k}} \sum_{v=0}^{n-l} \binom{R}{v} p(k)^v q(k)^{R-v} \right).$$

Assuming k failures, the survival probability of a system with n NVRAM parity buckets per R bucket group is $Q_{\mathrm{nP}}(k) = Q(n,R,k)^{\frac{N \cdot B}{2 \cdot R}}$, since there are $\frac{N \cdot B}{2 \cdot R}$ parity groups in the system.

Figure 2 shows the effect of computing NVRAM parity across data buckets. These results show that using of parity groups with just a single parity element has a dramatic effect on reducing the probability of data loss given multiple OSD failures. We find that calculating a single NVRAM parity across every $1,000$ buckets gives even odds to survive roughly 35 OSD failures, while mirroring alone has the same odds to survive about 4 OSD failures.

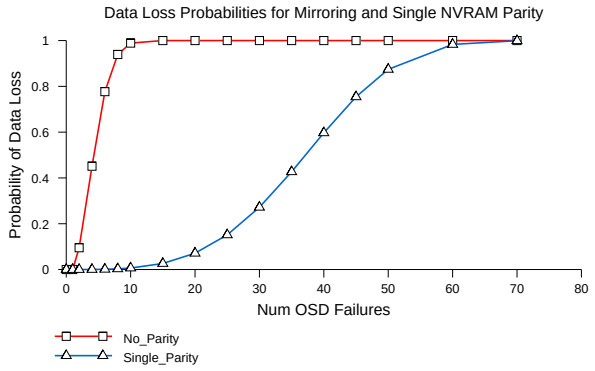


Figure 2: *Data loss probabilities for an increasing number of OSD failures, across mirroring and the single parity reliability schemes. Both schemes use* 2*-way mirroring as the primary redundancy scheme. The single parity scheme computes parity across groups of* $1,000$ *data buckets.*

Figure 3 gives the data loss probabilities for double and triple parity NVRAM layouts. First, notice that adding more NVRAM protection per group greatly increases the number of tolerated failures. This figure also shows the effect of group size on the probability of data loss, which is lower for smaller R. This observation is rather intuitive because the probability of two or more OSD failures occurring in the same group decreases with group size. The reliability benefit associated with group size introduces a cost/reliability tradeoff; decreasing the size of the NVRAM parity groups will increase NVRAM utilization.

3.2 Calculating Mean Time to Data Loss

The probability calculations given above demonstrate that computing large NVRAM parity groups across mirrored data results in very high data survival probabilities. We now use the probabilistic analysis to derive and compare the expected mean time to data loss.

Figure 4 shows our generic Markov model with which we calculate Mean Time to Data Loss (MTTDL). The system is in a state S_k, where k denotes the number of OSDs currently unavailable. The initial state is S_0. For accuracy, the model presumes that we replace failed OSDs with new OSDs, whereas in reality, the OSDs are not replaced; instead, the system is replenished from time to time with new batches of OSDs. While these OSDs would probably also have a higher capacity and carry more buckets, modeling these details is quite difficult and would not yield additional insights.

Since the baseline redundancy scheme is 2-way mirroring, the failure of two or more OSDs may result in data loss. If $Q_{\mathrm{nP}}(k)$ denotes the chances of survival when k disks have failed, then we calculate the probability that the additional failure caused data loss as the conditional probability

$$P_k = \Pr(DL_k | NDL_{k-1}) = 1 - \frac{Q_{\mathrm{nP}}(k)}{Q_{\mathrm{nP}}(k-1)},$$

where DL_k is the event of data loss after k OSD failures and NDL_{k-1} is the event of no data loss after $k-1$ OSD failures. Since

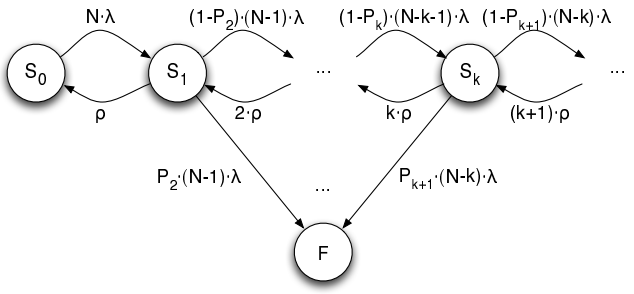


Figure 4: *Our Markov model resembles a standard Birth-Death model with an additional absorbing failure state (F). Transitions between the non-absorbing states are OSD failures that do not result in data loss and OSD repairs. A transition to the failure state is a data loss event.*

all buckets are mirrored across distinct OSDs, we set $P_0 = 0$ and $P_1 = 0$.

Our Markov model is closely related to a standard birth and death model, but with additional failure transitions. The model assumes that disk drives fail at a constant rate λ and "repair" replicates all buckets from the failed OSD elsewhere and that the rate of repair is ρ.

As shown in Figure 4, an OSD failure corresponds to a birth and a repair to a death. An OSD failure without data loss is the transition from state S_k going to S_{k+1} with rate $(1-P_{k+1})(N-k)\lambda$. If an OSD failure leads to data loss, then the model transitions to the failure state F, taken with frequency $P_{k+1}(N-k)\lambda$. The repair transition from state S_k to S_{k-1} is taken with rate $k\rho$, where the multiplier, k, accounts for concurrent OSD recovery. We cannot expect numerically reliable answers for large numbers of state and we cut off our Markov model at a state S_S where $p_S \approx 1$ and then set $P_S = 1$.

Figure 5 displays the system MTTDL for a mirrored system without NVRAM parity and a system with single NVRAM parity. We measure system MTTDL as a function of OSD mean time to failure (MTTF) and repair rate. The analysis assumes repair rates of $\frac{1}{4}$, $\frac{1}{2}$ and 1 hour. Due to the seemingly random allocation of buckets to disks, these repair times are sufficient to "copy" all of the failed buckets on a OSD to other OSDs in the system. The MTTDL numbers indicate that the use of single NVRAM parity per group results in a much more reliable system than standalone mirroring. We also find that repair rate has a non-trivial effect on system reliability. Even though the use of a single NVRAM parity bucket per group results in a $4,000$ fold increase in MTTDL, applying these results to the provisioning of an actual storage system needs to be done with caution because the basic assumption-the Markov property- makes MTTDL values (millions of years) very hard to comprehend.

Unfortunately, we could not numerically handle the large number of required states for double and triple parity with any confidence. In light of this, we still believe the analysis as a whole justifies the use of NVRAM parity in a mirrored system.

4. RELATED WORK

Most large-scale distributed systems rely on redundancy for data availability and reliability. High performance systems such as GPFS [9] (which also supports RAID 5) and Ceph [12] utilize mirroring for redundancy. FAB [8] has the ability to use either erasure codes or mirroring for redundancy, while FARSITE [1] uses mirroring instead of erasure codes. All of these systems use a single level of redundancy, which may fall prey to correlated failures or failure during rebuild. We believe that our techniques could be incorporated into these systems resulting in much higher reliability.

Other systems use multiple levels of redundancy for greater fault-tolerance and availability. POTSHARDS [11] performs a single availability-centric split using threshold cryptography before storing data across archives using secure distributed RAID. Oceanstore [6] replicates so-called active data and disperses copies of this data into deep archival data using erasure codes. Our scheme independently generates two-levels of redundancy in a way that has a minimal impact on average-case performance, while dramatically improving data reliability.

A variety of studies analyze ways to further improve storage system reliability. Xin *et al.* [13] propose mechanisms for mirrored and mirrored RAID 5 systems that increase system recovery rate and recover from nonrecoverable read errors. While Xin *et al.* analyzed mirrored and mirrored-RAID 5 configurations, we propose and analyze large parity codes over mirrored groups.

Schwarz *et al.* [10] propose and analyze a mechanism, called disk scrubbing, used to actively check data integrity in large storage systems. Baker *et al.* [3] and Bairavasundaram *et al.* [2] further validate the importance of active data scrubbing (or auditing) for detecting and recovering from latent faults as quickly as possible. Although we did not consider active checking in our analysis, we expect disk scrubbing to be integral in future analysis—especially when considering latent errors.

5. CONCLUSION AND FUTURE WORK

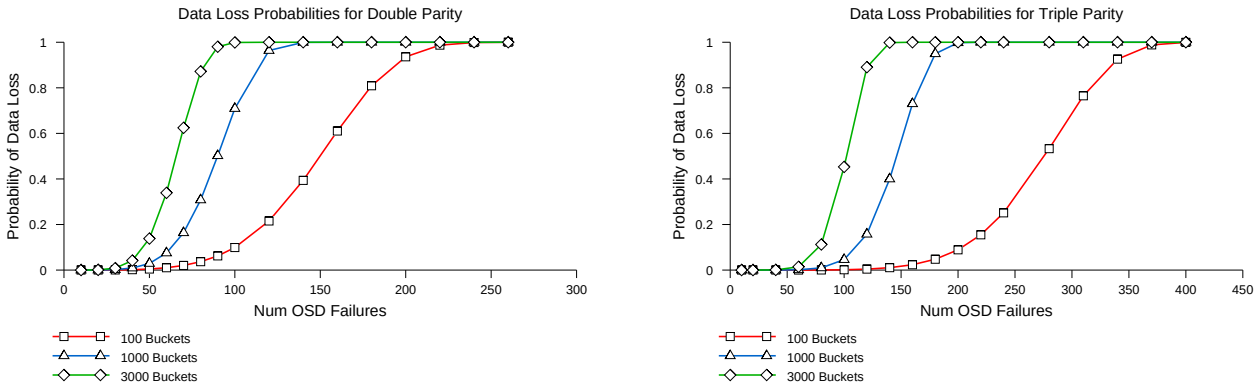
There exists a great deal of work to be completed in our disaster recovery encoding scheme. Our preliminary analysis does not include a performance evaluation. In the future, we plan to compare the update, degraded-mode read and rebuild overhead of our two-level scheme to a one-level mirroring scheme. In addition, we plan to consider the effect of correlated failures, latent sector faults and "bad" batches of disks. Finally, this scheme may be extended to a multi-level hierarchy of arbitrary erasure encoding schemes. We plan to investigate the utility and reliability of this and other hierarchical encoding schemes.

We have presented a method that drastically increases the reliability of mirrored systems, while imposing relatively small space and expected performance overhead. Our scheme generates an extra level of redundancy, using large parity-based erasure codes, computed across mirrored groups. All parity is stored in distributed NVRAM banks, resulting in a faster implementation compared to storing additional redundancy on disk.

Our preliminary analysis shows that even the use of a parity group with a single parity element contributes greatly to the overall reliability of the system. Our probabilistic analysis shows that the NVRAM parity scheme increases resilience against multiple OSD failures that might occur, for example, as the result of a bad batch of disks. System MTTDL (defined as loss of a single data bucket), also increases, about 4000-fold by introducing a single RAM parity per 1000 buckets. As the standalone mirroring results show, recourse to the NVRAM parity is rare, but may be necessary since such events are quite traumatic.

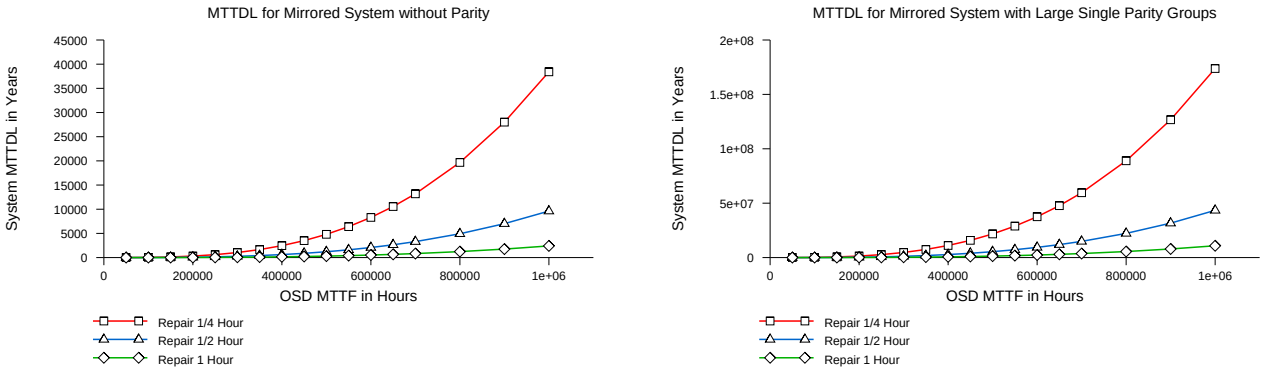
Acknowledgments

We would like to thank our colleagues in the Storage Systems Research Center (SSRC) who provided valuable feedback on the ideas in this paper. This research was supported by the Petascale Data Storage Institute under Dept. of Energy award DE-FC02-06ER25768, UCSC/LANL Institute for Scalable Scientific Data Management



(a) (b)

Figure 3: *Data loss probabilities for an increasing number of OSD failures and three parity group sizes, across double (a) and triple (b) parity groups. Both schemes use* 2*-way mirroring as the primary redundancy scheme.*



(a) (b)

Figure 5: *Mean Time to Data Loss (years) with* 2*-way mirroring (a) and with one NVRAM parity per 3000 bucket group (b) for a system with 10,000 disks, 1000 blocks per disk, and a repair times of 1/4, 1/2 and 1 hour. We find that repair repair time has a non-trivial effect on reliability. MTTF is in hours.*

and by SSRC sponsors including Los Alamos National Lab, Livermore National Lab, Sandia National Lab, Agami, Digisense, Hewlett-Packard Laboratories, IBM Research, Intel, LSI Logic, Microsoft Research, Network Appliance, Seagate, Symantec, and Yahoo.

6. 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] L. N. Bairavasundaram, G. R. Goodson, S. Pasupathy, and J. Schindler. An analysis of latent sector errors in disk drives. In *Proceedings of the 2007 SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, June 2007.

[3] M. Baker, M. Shah, D. S. H. Rosenthal, M. Roussopoulos, P. Maniatis, T. Giuli, and P. Bungale. A fresh look at the reliability of long-term digital storage. In *Proceedings of EuroSys 2006*, pages 221–234, Apr. 2006.

[4] K. M. Greenan. Efficient Galois field and data encoding library. http://www.soe.ucsc.edu/ kmgreen/html/sw.html, May 2007.

[5] 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.

[6] J. Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. OceanStore: An architecture for global-scale persistent storage. In *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Cambridge, MA, Nov. 2000. ACM.

[7] S. Nath, H. Yu, P. B. Gibbons, and S. Seshan. Subtleties in tolerating correlated failures in wide-area storage systems. In *Proceedings of the 3rd Symposium on Networked Systems Design and Implementation (NSDI)*, 2006.

[8] Y. Saito, S. Frølund, A. Veitch, A. Merchant, and S. Spence. FAB: Building distributed enterprise disk arrays from commodity components. In *Proceedings of the 11th*

International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), pages 48–58, 2004.

[9] F. Schmuck and R. Haskin. GPFS: A shared-disk file system for large computing clusters. In *Proceedings of the 2002 Conference on File and Storage Technologies (FAST)*, pages 231–244. USENIX, Jan. 2002.

[10] T. J. E. Schwarz, Q. Xin, E. L. Miller, D. D. E. Long, A. Hospodor, and S. Ng. Disk scrubbing in large archival storage systems. In *Proceedings of the 12th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '04)*, pages 409–418. IEEE, Oct. 2004.

[11] M. W. Storer, K. M. Greenan, E. L. Miller, and K. Voruganti. POTSHARDS: secure long-term storage without encryption. In *Proceedings of the 2007 USENIX Annual Technical Conference*, pages 143–156, June 2007.

[12] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, and C. Maltzahn. Ceph: A scalable, high-performance distributed file system. In *Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI)*, Seattle, WA, Nov. 2006. USENIX.

[13] 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.