Protecting Against Rare Event Failures in Archival Systems

Technical Report UCSC-SSRC-09-03
April 2009

Avani Wildani avani@cs.ucsc.edu
Thomas Schwarz tjschwarz@scu.edu
Ethan Miller elm@cs.ucsc.edu
Darrell Long darrell@cs.ucsc.edu

Storage Systems Research Center
Baskin School of Engineering
University of California, Santa Cruz
Santa Cruz, CA 95064
`http://www.ssrc.ucsc.edu/`

Protecting Against Rare Event Failures in Archival Systems

Avani Wildani
Univ. of California, Santa Cruz
avani@cs.ucsc.edu

Thomas J.E. Schwarz
Santa Clara University
tjschwarz@scu.edu

Ethan L. Miller
Univ. of California, Santa Cruz
elm@cs.ucsc.edu

Darrell D.E. Long
Univ. of California, Santa Cruz
darrell@cs.ucsc.edu

Abstract

Digital archives are growing rapidly, necessitating stronger reliability measures than RAID to avoid data loss from device failure. Mirroring, a popular solution, is too expensive over time. We present a compromise solution that uses multi-level redundancy coding to reduce the probability of data loss from multiple simultaneous device failures. This approach handles small-scale failures of one or two devices efficiently while still allowing the system to survive rare-event, larger-scale failures of four or more devices.

In our approach, each disk is split into a set of fixed size disklets *which are used to construct reliability stripes. To protect against rare event failures, reliability stripes are grouped into larger "über-groups," each of which has a corresponding "über-parity;" über-parity is only used to recover data when disk failures overwhelm the redundancy in a single reliability stripe. Über-parity can be stored on a variety of devices such as NV-RAM and always-on disks to offset write bottlenecks while still keeping the number of active devices low.*

Our calculations of failure probabilities found that the addition of über-groups allowed the system to absorb many more disk failures without data loss. Through discrete event simulation, we found that adding über-groups only negatively impacts performance when these groups need to be used for a rebuild. Since rebuilds using über-parity occur very rarely, they minimally impact system performance over time. Finally, we showed that robustness against rare events can be achieved for under 5% of total system cost.

1. Introduction

The amount of archival data in the world is growing rapidly as more corporations go paperless and more personal data is stored in the cloud. According to the Enterprise Strategy Group, a typical company experiences a 50% annual data store growth rate [15]. Much of this data, including medical records, photographs, and electronic correspondence is archival, which means that it is rarely read or updated yet may be valuable someday. Our goal is to propose a method to increase the reliability of storage systems aimed at this sort of archival data without significantly increasing either the startup or the operational costs of the underlying storage system.

Though individual disk failures are more common, large, correlated rare-event disk failures are a major concern for long-term storage. We want to increase reliability as much as we can afford to because even a very small loss rate corresponds to significant data loss. For example, a 10 PB system with an annual loss rate of 0.001% still loses a terabyte of data every decade. Recoverable correlated failures, *i.e.* those that only include a subset of the disks, may include power surges, rack failures, operator errors, cooling malfunctions, and disk batch failure. To inexpensively protect against these sorts of failures, we propose supplementing the normal redundancy in a storage system with relatively few devices worth of additional parity information.

The underlying storage system can be any declustered disk array where data is stored in reliability stripes, and we add additional protection by grouping a number of stripes into a larger group we refer to as the "über-group" with its own corresponding additional parity, the "über-parity." As before, failures of small, independent sets of disks are handled using the extent redundancy in the reliability stripe. In rare event failure cases, we can frequently rebuild all of the failed disks by reading all data in an über-group and using the über-parity, thus preventing data loss.

We propose a low cost method of adding reliability to archival systems while potentially reducing the number of disks spun up on write, resulting in fewer failed disks and better write performance. We also analyze the reliability trade-off between adding additional layers of reliability versus adding parity to existing layers.

The remainder of this paper is organized as follows. First, we present our system design after a short background on erasure codes. Then we cover related work in the field, followed by robustness analysis, a breakdown of the relevant costs, and experimental results. Finally, we discuss our results and the future directions of our work.

2. Related Work

Several studies have shown that storing data reliably over the archival time scale presents additional challenges to the already difficult field of storage reliability [3,17,18]. A challenge of this size requires combining known techniques of enhancing reliability with methods optimized for the expected workload and time scale of archival storage.

High-performance storage systems such as Ceph and GPFS that add reliability through mirroring are designed for systems where availability and performance trump costs [19,25]. For a long term system, keeping several times the number of disks you have for data on hand is infeasible. Coded systems such as RAID-5 [4] or even RAID-6 [5] reduce the disk overhead while distributing the risk for individual blocks of data, but do not provide enough reliability.

GFS [7] takes this a step further and adds the concept of multiple levels of redundancy to protect somewhat against correlated failures. Baker *et al.* [3] showed that the most important contributors to long-term storage reliability are fast detection of latent sector errors, independent replicas, and fast automatic repair. In archival systems, it is worth relaxing this fast repair constraint to have strong reliability without the cost overhead of multiple levels of replication.

Disk-based systems such as Oceanstore [14] and SafeStore [13] combine mirroring, erasure codes, and strategic data placement to add reliability to their storage networks. While these systems are fast and highly available, they are less suited for archival storage because they are not optimized to be low power and low cost. Other tiered reliability schemes similarly lack an emphasis on cost and power management [8,16]. We believe that our methods will work best over power-aware archival storage systems such as Pergamum [23] or PARAID [24].

Greenan *et al.* introduced the idea of using large-stripe erasure codes for storage [9]. We extend their model from coding over 2-way mirrored groups to looking at trade-offs between multiple levels of erasure coding. Since their reliability groups are smaller, they can store parity in NVRAM. This is a future direction for our model once prices are more competitive with an always-on disk.

3. System Design

Here, we cover our system architecture after presenting a brief review of erasure codes and their place in storage reliability.

3.1. Background: Erasure Codes

Erasure codes transform a message of m symbols into a message of n symbols, where $k = n - m$ is the number of redundant, or parity symbols in the code. Most codes that are used in storage are systemic, meaning that the original input data can be found in some part of the output. The changes required to turn one message into another is the *Hamming Distance* of the code, which for storage corresponds to the maximal set of failures that the code can tolerate. We assume a standard maximum distance separable (MDS) code with corresponding Hamming distance $k + 1$, meaning we can tolerate k failures in our n symbols. For calculating parity, we follow the same method as Greenan [10] by using an XOR-parity scheme to calculate single parity and a Reed Solomon scheme otherwise, giving us fast parity calculation. It is likely that a non-MDS code could provide better a better cost-benefit ratio for large stripe reliability groups, and we leave this for future work.

3.2. Architecture

Our system is based on a disk arrays with a large, varying number D of disks. Each disk is split into L *d*isklets. A typical value for L would be 32, 64, or 128. The disk array groups n disklets into a reliability stripe, so that m disklets are storing user data and the remaining k, $k = n - m$, disklets store parity. The parities are calculated using a Maximum Distance Separable Code so that a stripe can have up to k unavailable disklets without losing access to all the data stored in the stripe. We assume that the data is protected against corruption by scrubbing [20,23]. The system guarantees that all disklets making up a stripe are located in different disks. At this level, the disk array is guaranteed to not suffer data loss with up to k disk failures, and has a high probability of withstanding even more failures. When disks fail, the data is reconstructed on other disks in the system. Over time, the number of disks fluctuates as disks fails and more disks are added to the system

On top of this standard disk array, we introduce our additional protection mechanism that deals with the occasional data loss in a stripe. The system groups r reliability stripes into an *über-group*. Figure 1 shows a sample physical layout of this architecture. Note that the über-parity is kept on dedicated devices that may be different device types than the data disks. For each stripe, it adds s disklets

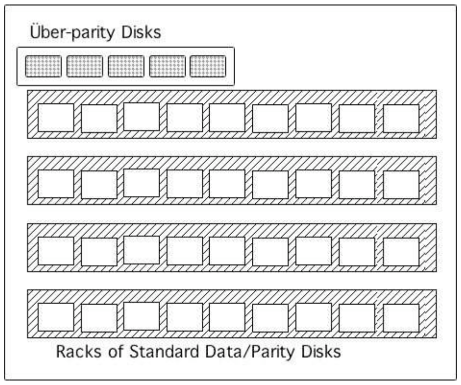


Figure 1: Sample physical system layout

storing *über-parity*. The über-parity is calculated from all user data in all disklets belonging to a stripe in the über-group using another erasure correcting code.

Unlike disks in the disk array, the storage devices storing über-parity only see read-modify-write traffic. The load at each über-parity disklet is r-times larger than at a reliability-stripe parity disklet. Thus, without a low write rate it would be hard to store the über-parity on the same device type as the data disks even if the über-parity disks were always left powered on. For a typical low-utilization archival system, however, the über-parity disks can be smaller, enterprise class disks that collectively handle the bandwidth. Today (2009), the disk array would contain 1 TB consumer disks, whereas the über-parity would be 200 GB enterprise disks. Other possible über-parity storage devices include Solid State Disks (SSD), Storage Class Memory (SCM), or even NV-RAM, all of which may alleviate a write bottleneck.

When we use über-parity to reconstruct otherwise lost data, we need to read all data in the über-group. Therefore, recovery from the effects of disk failures can be significantly more involved than for single stripe failure. We discuss this further in Sections 7 and 6.

We generate parity in our scheme with two different erasure correcting codes that together make up a pyramid code [11]. For our purposes, we can think of this class of codes as resulting from a large, linear, MDS code such as a Reed Solomon code with rm data symbols and $k+s$ parity symbols. We then remove the first k parity symbols from the code. Instead, we add k parity symbols to each stripe of m data symbols. These local parity symbols are calculated in the same way as in the original array except that we replace the data symbols outside this stripe of m by zeroes. Then, each of the old parity symbols is the sum of the corresponding new parity symbols. With this code, we can correct k erased symbols in a stripe using only the information from the stripe. Globally, our new code is stronger than the old one, which can only correct $k + s$ erasures. Assume a set of erased symbols. If there are f erased symbols in a stripe and $f > k$, then call $f - k$ the *excess* of the stripe. If the sum of the excesses is less than or equal to s, then we can still correct the erasures.

Table 1: Parameters and Sample Values

Parameter	Meaning
m	data disklets in a reliability strip
k	parity disklets in a reliability stripe
$n = m + k$	number of disklets in a reliability stripe
r	number of reliability stripes in a über-group
D	total number of disks
L	number of disklets per disk
$N = L \cdot D$	total number of disklets
$d = r \cdot n$	number of disklets per über-group
$u = \lceil \frac{D \cdot L}{n \cdot r} \rceil$	total number of über-groups
s	number of über-parities
f	number of failed disks
$U = \lceil \frac{D \cdot L \cdot s}{n \cdot r} \rceil$	number of disks storing über-parity

4. Analytical Model

We calculate the *robustness* of our disk array by computing the probability of data loss given a certain number of disk failures. Robustness calculations avoid several problems that occur when modeling the reliability of disk arrays. First, because we expect large scale failures, we cannot assume that disk failures are independent. Without a large-scale study of correlated data failure over time, modeling correlated disk failure would require assumptions that make the model meaningless. Similarly, given the required assumptions, MTTDL is difficult to translate into a real-world reliability metric. For example, we cannot simply assume that the failure and rebuild rates of disks are constant as they will vary by device type and installation. Since this analysis is difficult, we use simulation to get some idea of the expected MTTDL increase in Section 5.

4.1. Robustness

We define our parameters in Table 1. We assume that there are a small number f of failed disks at any point. Since disklets in a stripe come from different disks, the number x of unavailable disklets in a randomly chosen reliability stripe is binomially distributed:

$$p_x(f, n, D) = \binom{n}{x} \frac{f!(D-n)!(D-f)!}{D!(D-f-n+x)!(f-x)!}$$

Each reliability stripe protects itself against disklet unavailability through k parity disklets. Therefore, the probability p_S that all stripes remain accessible without resorting

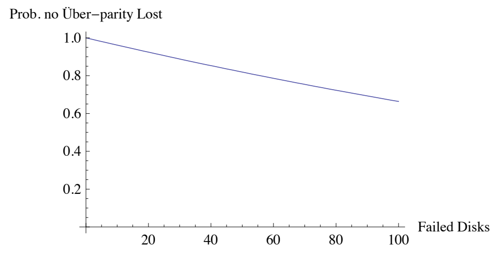


Figure 2: Probability that no über-parity bearing disk is lost. $n = 16, r = 16, D = 1024$

to the über-parities after f disk failures is:

$$p_S(k, f, n, D) = \sum_{i=0}^{k} p_i(f, n, D).$$

We denote the number of failed disks in reliability stripe i as f_i. The erasure correcting code can use the s über-parities to recover all data as long as $\sum_{i=1}^{r}(f_i - k) \le s$.

Define S to be the set of all tuples $\vec{f} = (f_1, f_2, \dots f_r)$ with non-negative integer coordinates f_j such that $\sum_{j=1}^{r} \max(f_j - k, 0) \le s$. The probability p_{Uber} that an über-group has failed is:

$$p_{\text{Über}}(s, k, f, n, D) = \sum_{\vec{f} \in S} \prod_{i=1}^{r} p_{f_i}(f, n, D).$$

For instance, an über-group with two parity disks per stripe ($k = 2$) and two über-parities per über-group protects against data loss if there is either one stripe with four unavailable disklets, one or two stripes with three unavailable disklets, or no stripe with more than two unavailable disklets. Correspondingly:

$$\begin{aligned} p_{\text{Uber}}(2, 2, f, n, D) = &\binom{n}{1} p_4(f, n, D) p_S(k, f, n, D)^{r-1} \\ &+ \binom{n}{1} p_3(f, n, D) p_S(k, f, n, D)^{r-1} \\ &+ \binom{n}{2} p_3(f, n, D)^2 p_S(k, f, n, D)^{r-2} \\ &+ p_S(k, f, n, D)^r \end{aligned}$$

4.2. Über-parities Failures

Potential failures of the über-parities have a disproportionately large effect on system reliability. Since we need to write to the corresponding über-parity whenever we write to the system, we analyze robustness with respect to two very different classes of über-parity storage devices. First, we consider über-parity stored on the same sort of devices as the data, which we assume are the inexpensive, low-power commodity drives typical in archival systems. There are $\lceil sD/nr \rceil$ of these devices, which means that many disks need to fail to lose a large percentage of the über-parity (Figure 2), though even losing a little über-parity still has a considerable impact. Alternatively, if we assume that the über-parity is stored on a device that "never" fails, the probability of data loss is given by $p_{\text{dl}}$the probability that at least one über-group has lost data:

$$p_{\text{dl}}(s, k, f, n, D, r, L) = 1 - (p_{\text{Uber}}(s, k, f, n, D, r, L))^u.$$

Table 2: Data Loss Probabilities: Case 1: über-parity disks can fail; Case 2: über-parity disks cannot fail. $k = 1, s = 1$ $n = 16, D = 1024, r = 16, L = 64$.

f	Case 1	Case 2
1	0.	0.
2	0.00159525	0.00160774
3	0.0286551	0.0265257
4	0.106224	0.0998437
5	0.248078	0.237821
6	0.442996	0.431747
7	0.651256	0.642184
8	0.82319	0.8177
9	0.930476	0.928003
10	0.979671	0.978861
11	0.995754	0.995567
12	0.99939	0.99936
13	0.999942	0.999939
14	0.999996	0.999996
15	1.	1.

Going back to the case where über-parity is stored on the same type of disks as the rest of the data: if we assume first that there is only one über-parity per über-group, i.e. $s = 1$, we can calculate p_{uf}, the probability that x of the über-parity disks have failed assuming that there are f total failures:

$$p_{uf}(x, f, D, U) = \binom{U}{x}\binom{D}{f-x}\binom{U+D}{f}^{-1}.$$

If x über-parity-disks have failed, then ux/U über-groups have no protection, whereas $(1-x/U)u$ do. An über-group without über-parity protection has not suffered data loss with probability:

$$p_S(k, f, n, D)^r.$$

Hence, the probability of data loss is a sum weighted by the probability of losing that many über-parities:

$$p_{\text{dl}}(s,k,f,n,D,r,L) = 1 - \sum_{x=0}^{f} p_{uf} \times p_S^{uxr/U} \times p_{\text{Uber}}^{(1-x/U)u}.$$

Table 2 compares calculated values in a typical configuration. Even though in Case 1 (über-parity can fail) the percentage of failed disks is predictably smaller than in Case 2 (über-parity cannot fail), the data loss probability is higher if $f \geq 3$. This indicates that über-parities pull more than their weight for reliability.

4.3. Parameter Sensitivities

We now investigate the impact of various parameters on the robustness of disk arrays. For simplicity's sake we assume that über-parity disks do not fail. Figure 3 gives the behavior of less reliable configurations with regards to n, the size of the reliability stripes. Figure 3 also shows that a single über-parity has almost the same effect on robustness as increasing $k = 1$, even though the latter increases the storage costs for parity by a factor of $r = 16$.

Our next step is to vary L, n, and r across disk layouts with $D = 1024$ to examine parameter effects. From Figures 4, 5, and 6, we see that robustness is most heavily affected by n, which makes sense given that n has the most effect on the parity overhead. Increasing L lowers robustness by increasing the chances of encountering a bad configuration of an über-group with the number of über-groups. Remarkably, increasing r only modestly hurts robustness. This is good news since in many architectures, the über-parities would be stored on more expensive devices. The effect is more pronounced for $s = 1$ than for $s = 2$. When comparing schemes with the same amount of parity information updated by a write (e.g. $k = 2, s = 2$ and $k = 3, s = 1$,) we notice that robustness does not change much.

4.4. Disk Rebuild Workload

To measure the costs of recovery, we calculated the expected costs of recovering data after f disk failures. In these calculations, we assumed that we recover even if there was some data loss in the array. We depict our results for the $k = 2, s = 2$ and $k = 3, s = 1$ configurations in Figure 4.4 with parameters $D = 1024$, $L = 64$, $n = 16$, and $r = 16$. The graphs show both the cumulative reconstruction load and the reconstruction load due to recovery within a single stripe. We measure the load with respect to accessing a complete disk. Thus, a value of 10 means that we are reading and writing 10 disks worth of data. The load is sub-linear in f for two reasons. First, with higher f, a stripe or an über-group is more likely to contain more

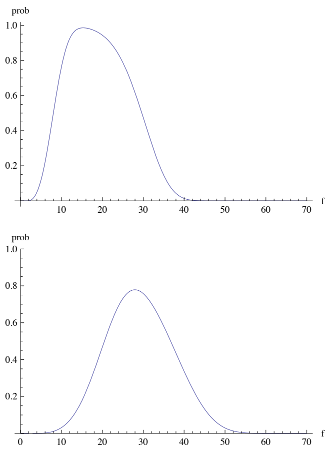


Figure 7: Rebuild workloads for $k = 2, s = 2$ (top) and $k = 3, s = 1$(bottom).

than one unavailable disklet or unrecoverable stripe. More failures also mean that it is more likely that the disk array cannot recover a stripe or an über-group. We can see that über-parity only becomes important after a relatively high number of disks in the array have failed.

4.5. Comparisons

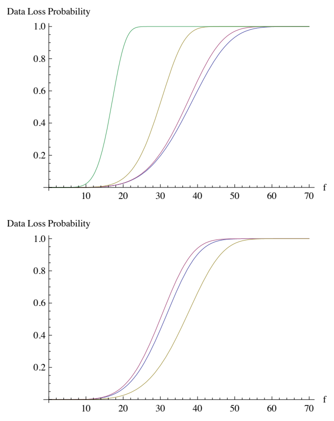
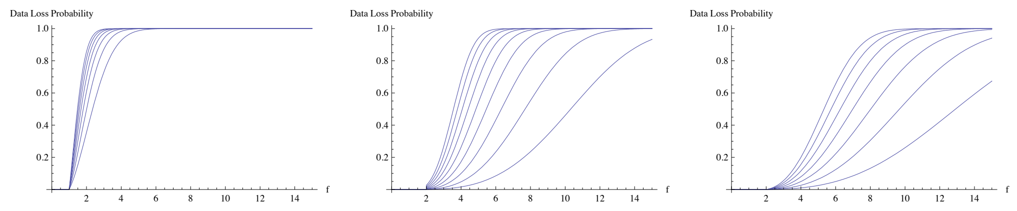


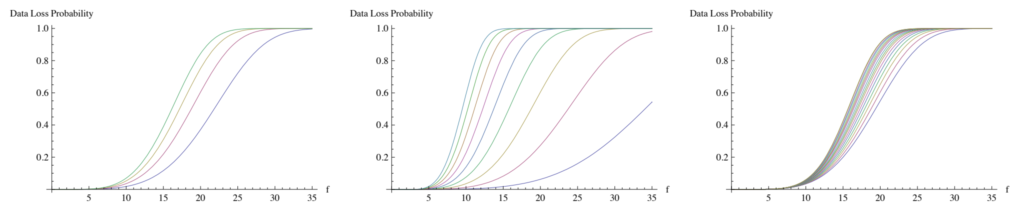
Figure 8: Naïve (top) versus Fair (bottom) comparisons between coding schemes.

We compare the robustness of four schemes, $k = 1, s = 3$; $k = 2, s = 2$; $k = 3, s = 1$; and $k = 4, s = 0$, all of which protect a data item with four parities, though they have different parity storage overheads, namely $19/256$, $34/256$, $49/256$, and $64/256$. Figure 8(a) shows that data loss is most likely when $k = 1$ and $s = 3$. If we assume for a moment that storage costs are the same for über-parity as for stripe parity, we can make a fairer comparison. Figure 8(b) compares the $k = 4, s = 0$ scheme with the $k = 3, s = 1$ scheme where we are using $n = 20$ and



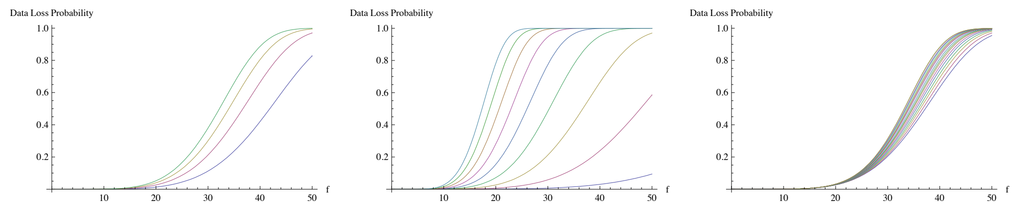
(a) $k = 1, s = 0$ (b) $k = 1, s = 1$ (c) $k = 2, s = 0$

Figure 3: Data Loss Probabilities varying $n = 8, 12, \ldots 32$ for different k, s pairs with $D = 1024, r = 16, L = 64$.



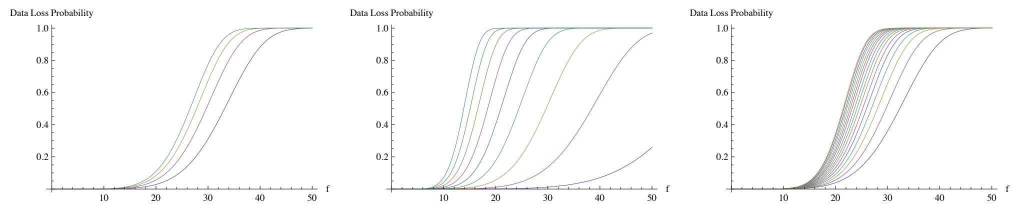
(a) $L = 32, 64, \ldots, 128$ (b) $n = 8, 12, \ldots, 40$ (c) $r = 8, 16, \ldots, 128$

Figure 4: Data Loss Probabilities for varying parameters from $k = 2$, $s = 1$, $D = 1024$, $r = 16$, and $L = 64$.



(a) $L = 32, 64, \ldots, 128$ (b) $n = 8, 12, \ldots, 40$ (c) $r = 8, 16, \ldots, 128$

Figure 5: Data Loss Probabilities for varying parameters from $k = 3$, $s = 1$, $D = 1024$, $r = 16$, and $L = 64$.



(a) $L = 32, 64, \ldots, 128$ (b) $n = 8, 12, \ldots, 40$ (c) $r = 8, 16, \ldots, 128$

Figure 6: Data Loss Probabilities for varying parameters from $k = 2$, $s = 2$, $D = 1024$, $r = 16$, and $L = 64$.

$n = 21$ in order to bracket the overhead of the $k = 3, s = 1$ scheme. Using the über-parity gives us the equivalent of tolerating seven additional disk failures. When we make the same comparison with the $k = 2, s = 2$ scheme, we need to set $n = 30, 31$ and find that the $k = 2, s = 2$ scheme tolerates six more disk failures. However, as we move more parity from the stripe into the über-group, the probability of having to use the über-parity for reconstruction increases (Figure 9). In summary, $k = 3, s = 1$ appears to be the best combination of standard parity and über-parity.

5. Simulation

Analyzing MTTDL directly is very hard (Section 4), but it is still a useful metric to see the benefits of über-parity. To examine the gains in MTTDL, we built a discrete event simulator that models the time to failure of disks and the rebuild time after a failure. Each iteration of the simulator runs until a data loss event is reached and the current time is recorded. Because we want to capture catastrophic failures, each iteration takes considerable time to run. Thus, we present results based on 100 iterations of the simulator, which has shown sufficient in earlier work [23].

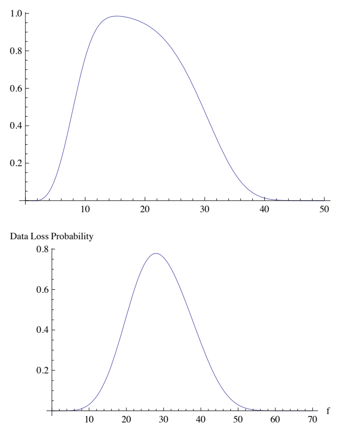


Figure 9: Probability of Reconstruction using the über-parity: $k = 2, s = 2$ (top) and $k = 3, s = 1$ (bottom), $n = 16, r = 16, L = 64$.

Our simulator was built on top of the Python SimPy library [22] and contains five core events: `DiskFail`, `DiskRebuild`, `LargeStripeRebuild`, `SectorFail` and `Scrub`. Values for disk failure time, sector failure time and disk scrub are all drawn from an exponential distribution. Though we believe that it would be interesting to look at write bottlenecks in the system, this system is designed for an archival workload where writes can be batched. Thus we consider exploring these bottlenecks as a secondary concern and do not model writes.

We model the effects of disk spin-up by subtracting 10 hours from the life of a disk every time it is spun up [20, 23]. This is a high estimate as newer drives become more robust, and so our simulation should correspond to a lower MTTDL in a real system.

We initialized the simulator to 1024 1 TB disks with reliability groups of 16 disks and über-groups of 256 disks. Über-parity is assumed to be stored on the same sort of disks as the data. Our baseline for comparison is having no über-parity represented by the 256-0 point on the graph. Figure 10 shows the percent increase in MTTDL over baseline from the addition of über-groups. We see the most gain in the 15 disk, 1 parity case since the über-parity has the greatest chance of being invoked. Note that this is a substantial improvement even though, as we discuss below, we are making worst case assumptions for disk rebuild. For systems with more parity, the line becomes effectively flat. This is due to failures being rare enough that extra parity matters less and the corresponding simulator results are more likely to fluctuate.

This is a different parity balance than our analytical result because it is measuring MTTDL instead of robustness. Our analytical results answer the question "What parity balance handles the most failures?" while our simulator results answer the question "What parity balance will result in the highest increase in MTTDL?" Also, since, making the assumptions needed to model correlated failures makes the model meaningless, we modeled failures as independent disk events pulled from an exponential distribution. We believe that a real-world scenario will have more correlated failures and thus will show über-parity being more useful for higher values of k.

We used a disk read rate of 7 MB/s to rebuild the disks, which leads to a standard rebuild time of approximately 10 hours and a worst case über-parity rebuild time of one week. This is a long period without data availability, but we do not expect disks in an archival system to be under a heavy read load and furthermore we expect the disk read time for commodity hard drives to approach that of current enterprise drives, which have read rates as high as 125 MB/s [1] within a few years, potentially decreasing the über-parity rebuild time by an order of magnitude. Finally, we stress this is a worst case, only triggered by all of the über-groups needing to rebuild simultaneously and each only operating at a quarter of the read rate.

We are simulating that rebuilding can only use up to a quarter of the disk bandwidth and only read from 16 disks at a time. However, the Reed Solomon style encoding we are using is completely parallelizable. Thus, if an installation could temporarily sacrifice disk bandwidth, the rebuild time would reduce to hours instead of days. Refer back to Section 7 for a more thorough analysis of rebuild performance.

6. Cost Analysis

Additional parity will always add reliability. The more interesting question is "Is the observed increase in reliability worth the cost?". An archival system is meant to run for the foreseeable future, so it is important to consider the operational cost projection as well the up-front costs of implementing any system meant to handle archival workloads. While it is important to keep the up-front cost low, we project that hardware prices will continue to decrease while operational costs, dominated by power, cooling, and data-center availability, will not decrease.

Assume a 10 PB system composed of 1 TB disks that are kept powered down unless needed for a write. Low power commodity terabyte drives are available for about \$100 [6]. For increased bandwidth, we want to store über-parity on smaller disks than the data disks. Say that we have 1 TB of über-parity stored on seven 160 GB enterprise disks, which can be found for about \$50 each [2]. This means that if we add 1 TB of über-parity disk every 100 disks, we are adding \$3850 per petabyte, or about 4%, to the startup cost of the system. NV-RAM currently costs about \$1 per gigabyte, leading to an 11% system overhead. While flash is more reliable than disk, there are few enough über-parity devices

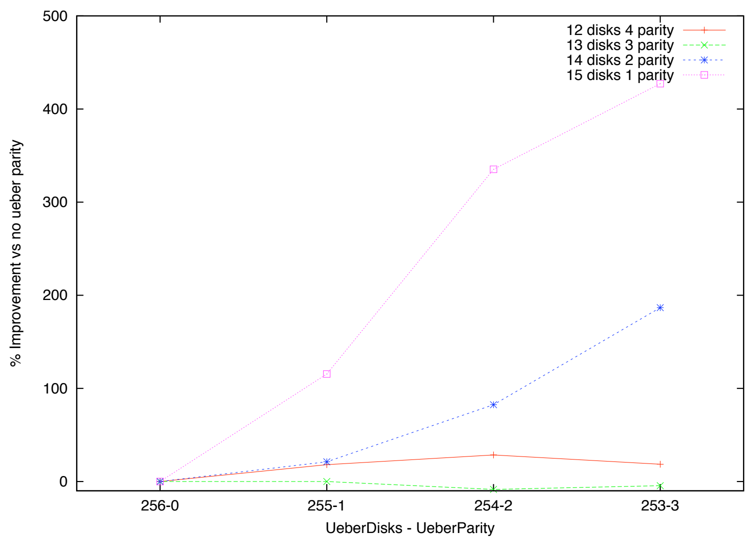


Figure 10: Percent increase in MTTDL with über-parity vs. no über-parity

that the impact of the device reliability is minimal, and thus we feel that the cost overhead is too high to justify having NV-RAM instead of having ten times the number of parity disks. In the future, solid state drives (SSDs) may also be an option, but as of now terabyte SSDs are expected to start at over $10,000 a piece [12].

The running cost of the additional disk is mainly the power it consumes. While running additional disks in a data center certainly makes the entire data center warmer, whether or not this will incur an increase in cooling costs is difficult to know [21]. On a power-aware archival system, we expect disks to typically be spun up about 5% of the time. Assuming $0.12 per KWatt with modern efficient drives, this results in an annual operating cost of at least $40,000. The additional annual operating cost is approximately $3,000, or 0.7% overhead, and thus a small enough overhead that the reliability increase is well worth the cost.

7. Future Work

We see several possible extensions to this project. First, our current analysis is restricted to MDS codes. Non-MDS codes could allow us to encode our domain knowledge about the reliability of different sets of disks into the code itself. For example, if an organization has servers in several stable areas and one volatile area, the large-stripe code could be biased to handle three failures in the unstable region and only one in the stable for less cost than adding three-disk parity across the entire stripe. This will be especially important as we hope to move this technique from always-on disks to NV-RAM as prices decrease. Another area for future development is to account for correlated failures. As we discussed in Section 1, catastrophic, large-scale failures are likely to correspond to real world events such as a fire or a batch failure, and thus it is more correct to treat them as correlated events. This presents several difficulties in modeling and simulation, especially as there are very few datasets from which to obtain priors for a real-world failure model.

8. Conclusion

We have presented a powerful technique for increasing the reliability of an archival system for minimal cost. In our analysis, we saw that using über-parity allowed us to tolerate up to seven additional disk failures. In the simulation, we see up to a four-fold improvement in MTTDL after adding über-parity. In short, we have a more robust system than RAID and require many fewer disks than mirroring. Thus, with its correspondingly low lost, we believe adding über-parity is a good value proposition for archival storage systems. Finally, adding über-parity to a system is also easy for an administrator to manage, and thus, given an amenable workload, is a clear choice for making an archival system more robust.

Acknowledgments

We would like to thank our colleagues in the Storage Systems Research Center (SSRC) who provided valuable feedback for this paper.

This research was supported by the Petascale Data Storage Institute under Department of Energy award DE-FC02-06ER25768, and by the industrial sponsors of the SSRC, including Los Alamos National Lab, Livermore National Lab, Sandia National Lab, Data Domain, Digisense, Hewlett-Packard Laboratories, IBM Research, LSI Logic, Network Appliance, Seagate, Symantec, and Yahoo!.

References

[1] Cheetah Hard Drive Family. `http://www.seagate.com/www/en-us/products/servers/cheetah/`, 2009.

[2] WD Caviar RE2 [RAID EDITION] Enterprise 160GB SATA 3.0Gb/s 3.5-inch Hard Disk Drive. `http://www.directron.com/wd1601abys.html`, 2009.

[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] S. Chen and D. Towsley. The design and evaluation of RAID 5 and parity striping disk array architectures. *Journal of Parallel and Distributed Computing*, 17(1-2):58–74, 1993.

[5] A. Dholakia, E. Eleftheriou, I. Iliadis, J. Menon, and K. Rao. Analysis of a new intra-disk redundancy scheme for high-reliability RAID storage systems in the presence of unrecoverable errors. In *Proceedings of the joint international conference on Measurement and modeling of computer systems*, pages 373–374. ACM New York, NY, USA, 2006.

[6] G. Gasior. Western Digital's Caviar GP hard drive. `http://techreport.com/articles.x/13379`, 2007.

[7] 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.

[8] K. Gopinath, N. Muppalaneni, N. S. Kumar, and P. Risbood. A 3-tier RAID storage system with RAID1, RAID5 and compressed RAID5 for Linux. In *Proceedings of the Freenix Track: 2000 USENIX Annual Technical Conference*, pages 21–34, June 2000.

[9] K. Greenan, E. Miller, T. Schwarz, and D. Long. Disaster recovery codes: increasing reliability with large-stripe erasure correcting codes. In *Proceedings of the 2007 ACM workshop on Storage security and survivability*, pages 31–36. ACM New York, NY, USA, 2007.

[10] K. Greenan, E. Miller, T. Schwarz, and D. Long. Disaster recovery codes: increasing reliability with large-stripe erasure correcting codes. In *Proceedings of the 2007 ACM workshop on Storage security and survivability*, pages 31–36. ACM New York, NY, USA, 2007.

[11] C. Huang, M. Chen, and J. Li. Pyramid Codes: Flexible Schemes to Trade Space for Access Efficiency in Reliable Data Storage Systems. In *Sixth IEEE International Symposium on Network Computing and Applications, 2007. NCA 2007*, pages 79–86, 2007.

[12] Z. Kerekes. 3.5" Terabyte Solid State Disks with Gigabyte / Sec Performance. `http://www.storagesearch.com/ssd-terabyte-art.html`, 2009.

[13] R. Kotla, L. Alvisi, and M. Dahlin. SafeStore: a durable and practical storage system. In *Proceedings of the 2007 USENIX Annual Technical Conference*, pages 129–142, June 2007.

[14] 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.

[15] W. Layton. Getting Ahead of the Data Storage Energy Crisis: The Case for MAID. `http://www.wwpi.com/top-stories/6493-getting-ahead-of-the-data-storage-energy-crisis-the-case-for-maid`, 2008.

[16] N. Muppalaneni and K. Gopinath. A multi-tier RAID storage system with RAID1 and RAID5. In *Parallel and Distributed Processing Symposium, 2000. IPDPS 2000. Proceedings. 14th International*, pages 663–671, 2000.

[17] 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.

[18] M. D. Panos Constantopoulos and Meropi Petraki. Reliability modelling for long-term digital preservation. In *9th DELOS Network of Excellence thematic workshop "Digital Repositories: Interoperability and Common Services", Foundation for Research and Technology - Hellas (FORTH)*, 2005.

[19] 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.

[20] 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, Oct. 2004.

[21] R. Sharma, C. Bash, C. Patel, R. Friedrich, and J. Chase. Balance of power: Dynamic thermal management for internet data centers. *IEEE Internet Computing*, 9(1):42–49, 2005.

[22] SimPy Team. SimPy homepage. `http://simpy.sourceforge.net/`, 2007.

[23] M. W. Storer, K. M. Greenan, E. L. Miller, and K. Voruganti. Pergamum: Replacing tape with energy efficient, reliable, disk-based archival storage. In *Proceedings of the 6th USENIX Conference on File and Storage Technologies (FAST)*, Feb. 2008.

[24] C. Weddle, M. Oldham, J. Qian, A.-I. A. Wang, P. Reiher, and G. Kuenning. PARAID : A gear-shifting power-aware RAID. In *Proceedings of the 5th USENIX Conference on File and Storage Technologies (FAST)*, Feb. 2007.

[25] 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.