Protecting Against Rare Event Failures in Archival Systems

Avani Wildani*, Thomas J. E. Schwarz†, Ethan L. Miller*, and Darrell D.E. Long*
*Storage Systems Research Center
University of California, Santa Cruz
{avani,elm,darrell}@cs.ucsc.edu
†Computer Engineering Department
Santa Clara University
tjschwarz@scu.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 *super-groups*, each of which has a corresponding *super-parity*; super-parity is only used to recover data when disk failures overwhelm the redundancy in a single reliability stripe. Super-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 show that adding super-parity allows our system to absorb many more disk failures without data loss. Through discrete event simulation, we found that adding super-groups has a significant impact on mean time to data loss and that rebuilds are slow but not unmanageable. Finally, we showed that robustness against rare events can be achieved for a fraction of total system cost.

I. Introduction

The amount of archival data in the world is growing rapidly as more corporations go paper-less and more personal data is stored in the cloud. The IDC claims that we produce 281 exabytes of data a year but throw much of it away [8]. According to the Enterprise Strategy Group, a typical company experiences a 50% annual data store growth rate [17]. 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.

As archival systems approach the petabyte scale, retaining data for longer lengths of time, such as a human lifetime, becomes even more difficult. In 2004, Xin *et al.* showed that the increased probability of disk failures in petascale installations already called for more redundancy than typical data centers provide [32]. Sector corruption alone can necessitate additional parity on top of the standard RAID-5 configuration such as RAID-6, intra-disk redundancy [14], or three-way replication [9].

We propose a method to increase the reliability of storage systems for archival data without significantly increasing either the startup or the operational costs of the underlying storage array. When discussing reliability, one must note that 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.

Rare failures can include power surges, rack failures, operator errors, cooling malfunctions, and disk batch failure. To protect against these sorts of failures inexpensively, we propose supplementing the normal redundancy in a storage system with relatively few devices worth of additional parity information.

The underlying storage system is a generic declustered disk array where each disk holds a variable number of fixed-size disklets. It is important to restrict the number of disklets per disk to something fairly small, typically < 100, to gain the benefits of declustering without adding an excessive amount of metadata to track the mappings of data to disklets. Disklets are placed in reliability stripes, and one or more parity disklets are calculated per stripe. We add additional protection by aggregating reliability stripes into *super-groups* with their own, independently calculated parity, the *super-parity*. If the system discovers failed disks or corrupt sectors, it first tries to recover the lost data by using the redundancy in the reliability stripe. Only in the rare event that this does not succeed will the system read all data in a super-group and use super-parity to reconstruct. Our additional protection functions as an insurance policy against these rare failures at the cost of few additional devices and write operations.

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 instances of data loss and better write performance. We also analyze the reliability trade-off between adding additional layers of reliability versus adding parity to existing layers. We investigate the robustness, defined as the probability of dataloss after several disk failures, of our schemes, use simulation to assess the Mean Time To Data Loss (MTTDL), and discuss the costs of adding super-parity to archival systems.

The remainder of this paper is organized as follows. After a review of related work, we present our system design. We then analytically calculate robustness, report on our simulation results to assess MTTDL of systems, and conclude with a discussion of costs and future directions.

II. 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 [2, 21, 22]. 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 [25, 31]. 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 [7] reduce the disk overhead while distributing the risk for individual blocks of data, but do not provide enough reliability.

GFS [9] takes this a step further and adds the concept of multiple levels of redundancy to protect somewhat against correlated failures. Baker *et al.* [2] 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 competing reliability schemes.

Disk-based systems such as Oceanstore [16] and SafeStore [15] 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 [10, 20]. We believe that our methods will work best over power-aware archival storage systems such as Pergamum [29] or PARAID [30].

Greenan *et al.* introduced the idea of using large-stripe erasure codes for storage [11]. 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 NV-RAM. We intend to explore using NV-RAM to store parity when NV-RAM prices become more competitive with always-on disks.

III. System Design

In order to protect against data loss, large storage installations need to store data redundantly. In this section, we cover our system architecture after presenting a brief review of erasure codes and their place in achieving storage reliability.

A. Background: Erasure Codes

Systematic erasure codes take m data symbols and add to them k *parity* symbols. A Maximum Distance Separable (MDS) code is defined as allowing the reconstruction of all $m+k$ symbols from *any* m symbols. Most often, the symbols are interpreted as elements of a Galois field with 2^f elements, with typical values $f = 8$ or $f = 16$. Linear MDS codes allow parity symbols to be updated based on the old parity symbol value and the difference between the old and new data contents. We can always insure that the first parity symbol is calculated as the sum of all data symbols [12, 19].

A storage system can use an MDS code by grouping m blocks for data into a reliability group and calculating the contents of k parity blocks by taking the first symbol from each data block as the data symbols of the chosen MDS code and obtaining the first symbols in each parity block as the parity symbol. Equally space efficient codes such as Even-Odd [3], row-diagonal parity [6], and Liberation codes [23] exist for $k = 2$ and $k = 3$. Other codes used in storage systems trade the space efficiency for code complexity [18, 24]. While we do not consider non-MDS codes in our analysis, we could use the same analysis techniques with these codes.

B. Archival Storage Systems

Archival systems are designed to store data for arbitrarily long periods of time, where the data is written once and often never read. Though this data may never be read, the data can be irreplaceable and losing it can incur a significant cost. For example, a corporate e-mail archive could be crucial for defending against a lawsuit. Data retention policies demand minimum times during which all stored data is accessible and instances of data loss might come under intense legal scrutiny. In contrast to other storage systems, archival systems rarely update data in place and need to store very large amounts of data for longer than the lifetime of the storage medium As disks become larger and cheaper, archival systems are increasingly being built disk arrays instead of traditional tape libraries. Depending on the workload, a disk-based archival storage system can save operating and cooling costs by powering off unused disks [5].

C. Architecture

Our architecture is based on a disk array with a large, varying number D of disks. We expect our disk array to be a heterogeneous mix of disk brands and capacities as new disks are incorporated into the array and old disks fail or are decommissioned. We split each disk into L *disklets* where each disklet stores either data or parity. L typically ranges from 20 to 150. Breaking up a disk into disklets has two primary advantages: it allows us to scale to disks of arbitrary size and, by carefully assigning data to disklets, it makes it easier for us to distribute parity across our system

The disk array groups n disklets into a *reliability stripe* such that m disklets store user data and the remaining $k = n - m$ disklets store parity data. The controller calculates the parity using a linear MDS code. As a result, a stripe can tolerate $\leq k$ unavailable disklets without loss of access to all data stored in the stripe. We assume that data is protected against block corruption by scrubbing [26, 29]. To maximize robustness, the

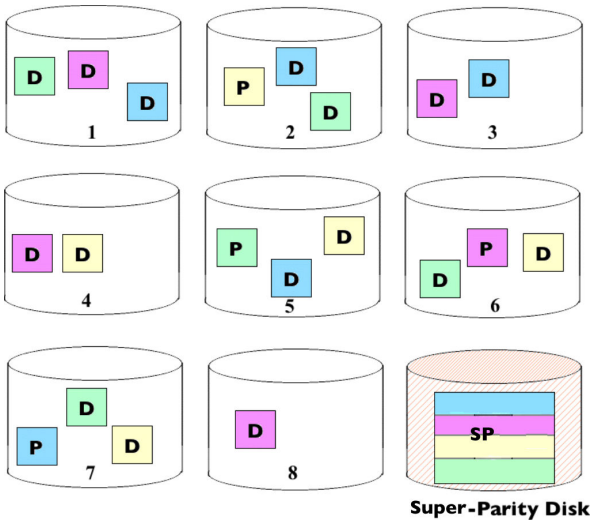


Fig. 1. Four reliability groups are split over eight disks with the super-parity device holding super-parity for the entire super-group. Each small square is a disklet.

system places all disklets that compose a stripe onto different disks whenever possible. The array can tolerate up to k disk failures without incurring data loss. The data from failed disks is reconstructed onto standby disks in the disk array.

On top of this standard disk array, we introduce our additional protection mechanism that deals with less predictable data loss in a stripe. The system groups r reliability stripes into a *s*uper-group. Figure 1 shows a sample physical layout of this architecture. Using standard parity, we can reconstruct any one failed disk. Super-parity is linearly independent from standard stripe parity, and so we can reconstruct any two failed disks using the super-parity to augment the stripe parity. Disk #9 is a dedicated *super-parity device*. Unlike the other disks in the system, these devices only see write traffic, mostly read-modify-write traffic. By keeping the super-parity on dedicated disks, we can take advantage of hardware that is better at handling a heavy write workload than traditional disks.

An archival storage system can improve read access times by striping data over disklets in a reliability group and thus avoiding read-modify-write operations for the stripe parity. Additionally, archival workloads rarely update data in place. We do not envision the system trying to stripe over super-groups, in part because of the size of the groups. Thus, write workload at the super-parity devices is heavy. Depending on this load, we can use a variety of devices such as:

1) Inexpensive, always-on SATA disks
2) Enterprise class disks. Ideally, we would use a larger number of lower-capacity disks than we do when using inexpensive disks.
3) Solid state memory and future storage class memory devices
4) Massively distributed non-volatile RAM (where each RAM device is very small).

Unlike for reliability stripes, we make no assumption that the rn disklets in a super-group (other than the super-parity disklets) are located in different disks. This policy greatly simplifies the construction of super-groups, especially as we expect data to migrate from failed disks to other disks. When we use super-parity to reconstruct otherwise lost data, we need to read all data in the super-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 IV-D and VI.

We generate parity in our scheme with two different erasure correcting codes that together make up a pyramid code [13]. 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 and 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 each stripe of m by zeroes, thus making the parities independent, as 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, we can still correct the erasures. Grid codes [18] have a similar structure and the same functionality, and our analysis applies to them as well.

IV. Analytical Model

We calculate the *robustness* of our disk array by computing the probability of data loss given a certain number of disk failures. Modeling robustness avoids several problems that occur when modeling the reliability of disk arrays. First, since we expect this system to run for a very long time, we must include correlated failures in our reliability analysis. Without a large-scale empirical study of correlated data failures over time in actual storage systems, modeling correlated disk failure would require enough assumptions to make the model meaningless. Similarly, many assumptions are required to translate Mean Time To Data Loss (MTTDL) into a realistic reliability metric. For example, we cannot simply assume that the failure rates of disks are constant as they will vary by age and device type. Additionally, data reconstruction times depend heavily on both expected workload and service guarantees, since handling reads and writes limit the disk and network bandwidth available to the rebuild process. We use simulation to get some idea of the expected MTTDL increase in Section V.

A. Robustness

We define our parameters in Table I. We assume that there are currently a small number f of failed disks. Since disklets in a stripe come from different disks, the number x of unavailable disklets in a randomly chosen reliability stripe is given as the quotient of the number of ways to allocate x failures among the n disks making up the stripe multiplied by the number of

TABLE I
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 super-group
D	total number of disks (excluding super-parity devices)
L	number of disklets per disk
$N = L \cdot D$	total number of disklets
$d = r \cdot n$	number of disklets per super-group
$u = \lceil \frac{D \cdot L}{n \cdot r} \rceil$	total number of super-groups
s	number of super-paritiesper super-group
f	number of failed disks
$U = \lceil \frac{D \cdot L \cdot s}{n \cdot r} \rceil$	number of disks storing super-parity

ways to allocate $f - x$ failures among the remaining $D - n$ disks and the number of ways to distribute f failures among all D disks:

$$p_x(f, n, D) = \binom{n}{x}\binom{D-n}{f-x}\binom{D}{f}^{-1}.$$

Each reliability stripe protects itself against disklet unavailability through its k parity disklets. Therefore, the probability p_S that all data in a given single reliability stripe remains accessible without resorting to super-parity after f disk failures is:

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

Since the total number of stripes is DL/n, the probability of dataloss without any super-parity given f failed disks is

$$p_{DL0}(f, k, n, D) = 1 - p_S(f, k, n, D)^{DL/n}.$$

We now calculate the robustness of individual super-groups. We denote the number of failed disks within reliability stripe i in a super-group as f_i. The numbers of failed disks per stripe over the r stripes in the super-group are independent stochastic variables. We assume that all s super-parities are available. The erasure correcting code can use these s disklets to recover all data in the super-group as long as $\sum_{i=1}^{r} \max(f_i - k, 0) \leq s$.

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

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

For instance, a super-group with $k = 2$ parity disks per stripe and $s = 2$ super-parity disklets per group protects against data loss if there is either

1) One stripe with $k + 2$ unavailable disklets and no other stripes with more than k unavailable disklets.
2) One or two stripes with $k + 1$ unavailable disklets and no other stripes with more than k unavailable disklets, or
3) No stripe with more than k unavailable disklets.

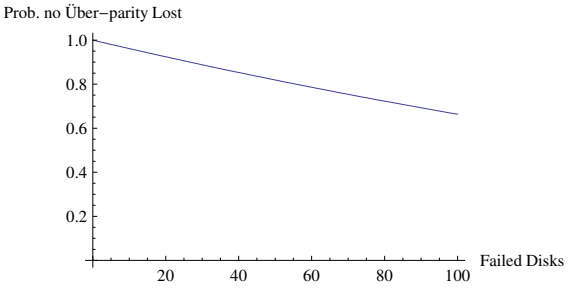


Fig. 2. Probability that no super-parity bearing disk is lost. $n = 16$, $r = 16$, $D = 1024$, $L = 64$

Correspondingly:

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

B. Super-parity Failures

Adding super-parity devices increase the robustness of the system disproportionate to the storage overhead. We will show super-parity failures have a disproportionately large effect on system reliability.

Depending on the expected load the archive has to sustain, we write super-parity on devices ranging from the inexpensive, high-capacity, low-power commodity drives typical in archival systems to enterprise disks to SSDs, NV-RAM, and other storage class memory solutions. To model super-parity failures more accurately given these different classes of devices, we consider super-parity on both high and zero failure devices. We expect a real system to fall somewhere between these two extremes, so we need to examine the impact of both assumptions.

First, we consider super-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. As a result, the super-parity will fail at the same rate as an ordinary disk.

There are only $\lceil sD/nr \rceil$ of these super-parity devices out of typically a much larger number of total disks, which means that many disks need to fail across the system to lose a large percentage of the super-parity (Figure 2). However, losing even a little super-parity still has a considerable impact on robustness.

Given f failures among the $D + U$ devices, the probability that x super-parity devices have failed is:

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

We now further restrict discussion to the case $s = 1$. If x of the $U = \frac{DL}{nr}$ super-parity have failed, then $v = xL$ of all super-groups have lost super-parity protection and $w = \left(\frac{D}{nr} - x\right) L$ have not. The probability that one of the parity-less super-groups has not lost data is equivalent to the probability that all reliability stripes in the super-group have not lost data, or $p_1 = p_S(f, k, n, D)^r$. Correspondingly, the probability that the super-groups that still have parity have lost data is $p_2 = p_S(f, k, n, D)^r + r p_{k+1}(f, n, D) p_S(f, k, n, D)^{r-1}$, which is the probability that either all super-groups have not lost data or exactly 1 super-group only suffered $k+1$ unavailable disklets.

We can now calculate the probability $p_{DL1'}$ of data loss as a sum of data loss suffered given x failed super-parity devices weighted by the probability of x failures among the super-parity devices:

$$p_{DL1'} = 1 - \sum_{x=0}^{U} (p_1^v \cdot p_2^w \cdot p_s f(x).$$

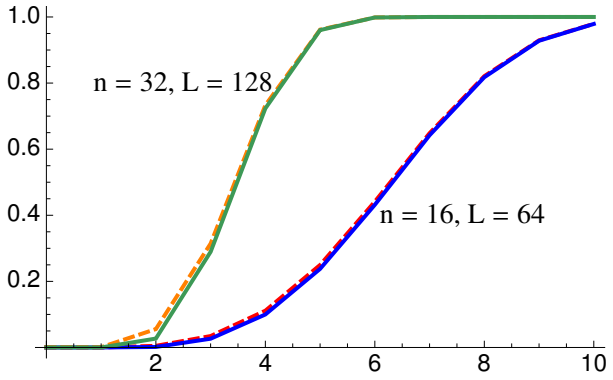


Fig. 3. Comparison of robustness with (dashed) and without super-parity device failures. Parameters are $D = 1024$, $r = 16$, $k = 1$, and $s = 1$ with $n = 32, L = 128$ or $n = 16, L = 64$

Alternatively, if we assume that the super-parity is stored on a device that "never" fails, the probability of data loss is given by p_{dl}, the probability that at least one super-group has lost data:

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

Figure 3 compares these two assumptions for two typical configurations. We set $D = 1024$, $r = 16$, $k = 1$, and $s = 1$. For each assumption, we plot curves for the parameter sets $n = 32, L = 128$ and $n = 16, L = 64$. In the case where super-parity devices can fail (the dashed lines in Figure 3), the curves are just slightly above the curves without failure. Since these curves are nearly indistinguishable, we can use the failure-free case to reasonably approximate a real system.

C. Parameter Sensitivities

We now investigate the impact of various parameters on the robustness of disk arrays. For simplicity's sake we assume that super-parity disks do not fail. Figure 4 compares the robustness of declustered RAID-5, declustered RAID-6, and RAID-5 with the addition of either one or two super-parity devices per super-group. We see that RAID-5 with one super-parity is on par with RAID-6, while RAID-5 with two super-parities is significantly more robust. While the operational overhead of the super-parity approaches are obviously higher, since each write touches more disks, the storage overhead is much smaller. For example, with n disklets per reliability stripe and $r = 16$ reliability stripes per super-group the storage overhead for the approach with two super-parities per super-groupis $\frac{1}{n}$ for the RAID-5 parity added to $\frac{2}{16n}$, representing the super-parity, for a total of $\frac{9}{8n}$ units of overhead versus the $\frac{2}{n}$ needed for RAID-6. Thus, super-parity clearly adds more robustness for its storage overhead compared to traditional parity.

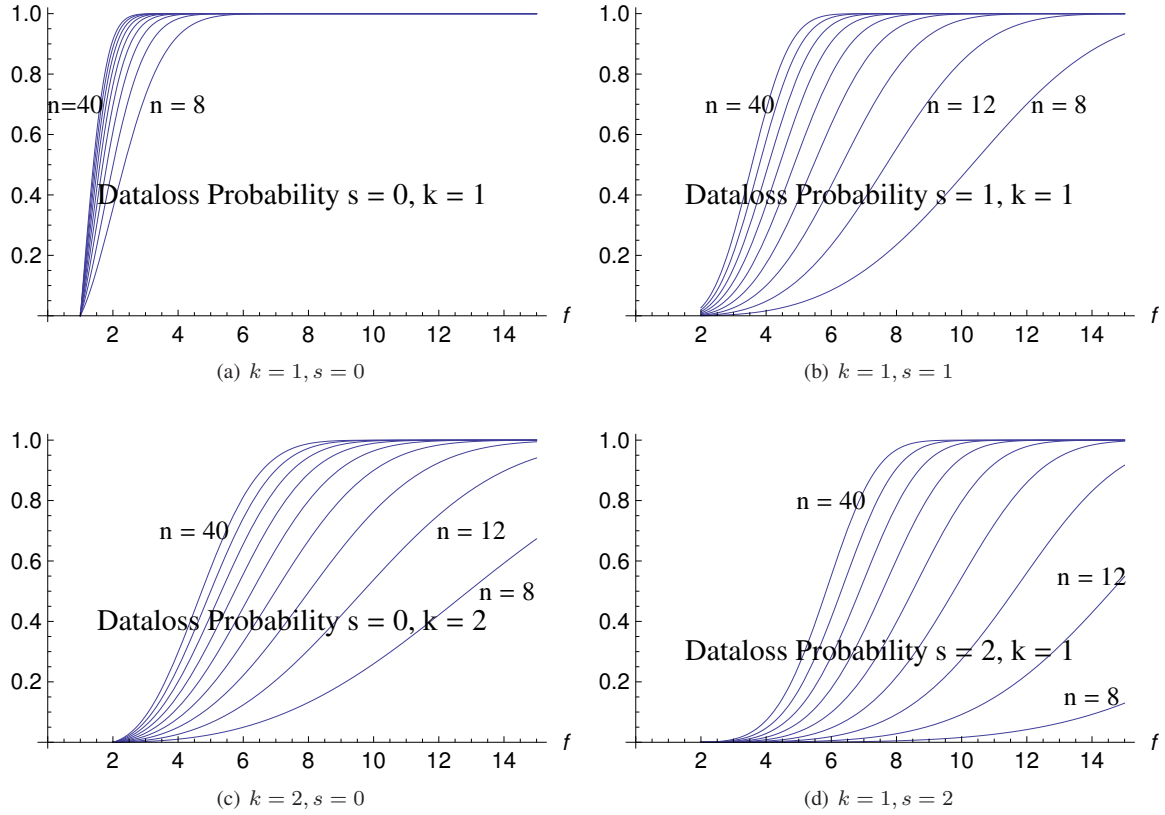
Our next step is to vary L, n, and r across disk layouts with $D = 1024$ to examine parameter effects. From Figures 5, 6, and 7, we see that robustness is most heavily affected by n, which makes sense given that n has the most effect on the parity overhead. Increasing L, the number of disklets per disk, lowers robustness by increasing the chance that a set of disk failures will impact enough stripes to reach an unrecoverable configuration of failures across the super-group. Also, as L increases the number of unrecoverable stripes, the amount of data lost during an unrecoverable failure increases accordingly. Remarkably, increasing r only modestly hurts robustness. This is good news since this allows us to spend our budget on a few faster, more resilient devices for super-parity without suffering a significant hit in added robustness. 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.

D. 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 $k = 1, s = 1$; $k = 2, s = 1$; and $k = 2, s = 2$ configurations in Figure IV-D with parameters $D = 1024$, $L = 64$, $n = 16$, and $r = 16$. Here, the workload is measured in the total number of disks accessed over all reads and writes. The different lines on the graph correspond to cumulative reconstruction load, the reconstruction load from recovering a single-stripe, and the reconstruction load within a single super-group.

The individual components of the workload show a single peak. As the number of failed disks (f) increases, there are more stripes or super-groups with disklets that need to be recovered. The peak occurs when we pass the point where enough disks have failed that the probability that all data can be recovered approaches zero. As expected, the recovery workload for the super-parity case has a steeper rise and fall than the standard stripe recovery case and peaks at a different f, resulting in the total workload potentially having two maxima.



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

(c) $k = 2, s = 0$ (d) $k = 1, s = 2$

Fig. 4. Data Loss Probabilities varying $n = 8, 12, \ldots 32$ for different k, s pairs with $D = 1024, r = 16, L = 64$.

E. Comparisons

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. These configurations have parity storage overheads of $19/256$, $34/256$, $49/256$, and $64/256$, respectively. Figure 9 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 super-parity as for stripe parity, we can make a fairer comparison (Figure 10).

In Figure 11, we see that setting $n = 16$ for the $k = 4, s = 0$ configuration gives us a storage overhead similar to setting $n = 20$ in the $k = 3, s = 1$ configuration. At this storage overhead, the configuration with super-parity can tolerate approximately seven additional 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 super-group, the probability of having to use the super-parity for reconstruction increases. In summary, $k = 3, s = 1$ appears to analytically be the best combination of standard parity and super-parity.

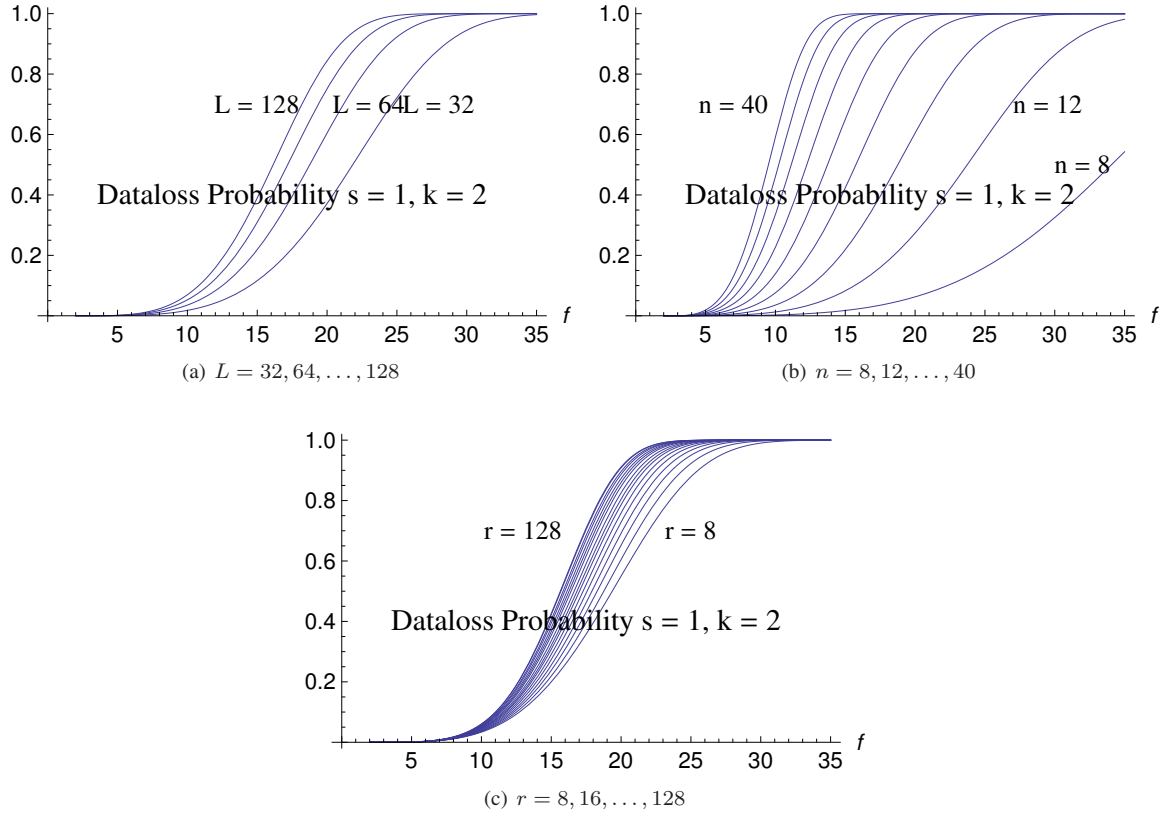
V. Simulation

Analyzing MTTDL directly is very hard (Section IV), but it is still a useful metric to see the benefits of super-parity. To examine the gains in MTTDL, we built a discrete event simulator that models independent disk failures 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 [29].

Our simulator was built on top of the Python SimPy library [28] 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 [26, 29]. 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 super-groups of 256 disks. Super-parity is assumed to be stored on the same sort of disks as the data. Our baseline for comparison is having no super-parity represented by the 256-0 point on the graph. Figure 12 shows the percent increase in MTTDL after adding super-



(a) $L = 32, 64, \ldots, 128$

(b) $n = 8, 12, \ldots, 40$

(c) $r = 8, 16, \ldots, 128$

Fig. 5. Data Loss Probabilities for varying parameters from $k = 2$, $s = 1$, $D = 1024$, $n = 16$, $r = 16$,and $L = 64$.

groupsand super-parity. On the x-axis we show the different super-groupsuper-parity configurations we simulated (*e.g.*, 255-1 corresponds to the case where there is 1 super-parity for every 255 disklets). We see the most gain in the 15 disk, 1 parity case since the super-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.

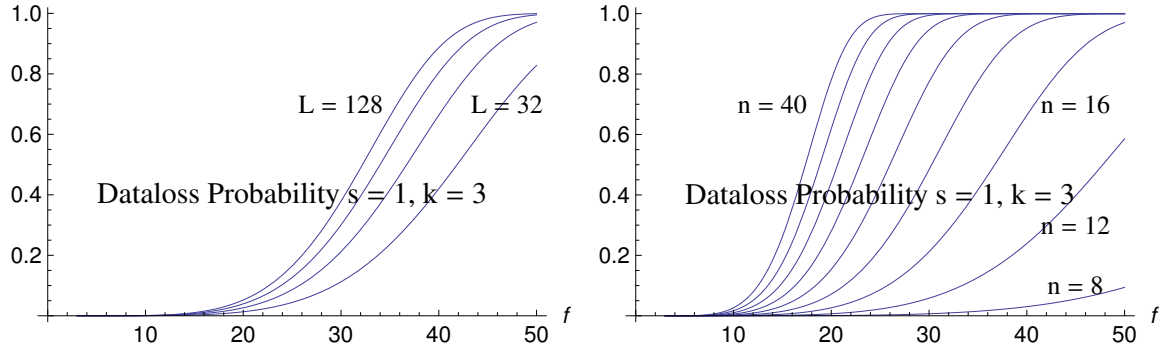
This is a different parity balance than our analytical result because it is measuring MTTDL instead of robustness. Our analytical results answer the question "What parity balance handles the most failures?" while our simulator results answer the question "What parity balance will result in the highest increase in MTTDL?" Since we lack real data for failure correlation, 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 more super-parity being more useful.

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 super-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 super-parity rebuild time by an order of magnitude. Finally, we stress this is a worst case, only triggered by all of the super-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. We expect that in a real system we could use more disk bandwidth and rebuild more disks in parallel. Thus, if an installation could temporarily sacrifice disk bandwidth, the rebuild time would reduce to hours instead of days. Refer back to Section IV-D for a more thorough analysis of rebuild performance.

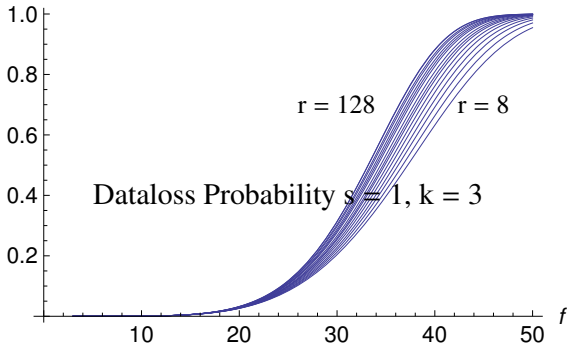
VI. Cost Analysis

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



(a) $L = 32, 64, \ldots, 128$

(b) $n = 8, 12, \ldots, 40$



(c) $r = 8, 16, \ldots, 128$

Fig. 6. Data Loss Probabilities for varying parameters from $k = 3$, $s = 1$, $D = 1024$, $n = 16$, $r = 16$,and $L = 64$.

TABLE II
SUPER-PARITY DEVICE COSTS

Super-parity Device	Initial Equipment Overhead	Annual Power Overhead	5-Year Overhead	Write Rate
Western Digital Caviar: 4x250 GB SATA Disks	$10,320	$3,210	$26,370	100 MB/s per disk
Distributed NV-RAM: 1 GB Devices	$44,032	$10· avg. simultaneous writes	$48,997 with < 100 simultaneous writes *	Variable: 100× write = 15 MB/s per device.
Axiom: 4x256 GB SSDs	$522,708	$460	$525,008	200 MB/s per disk

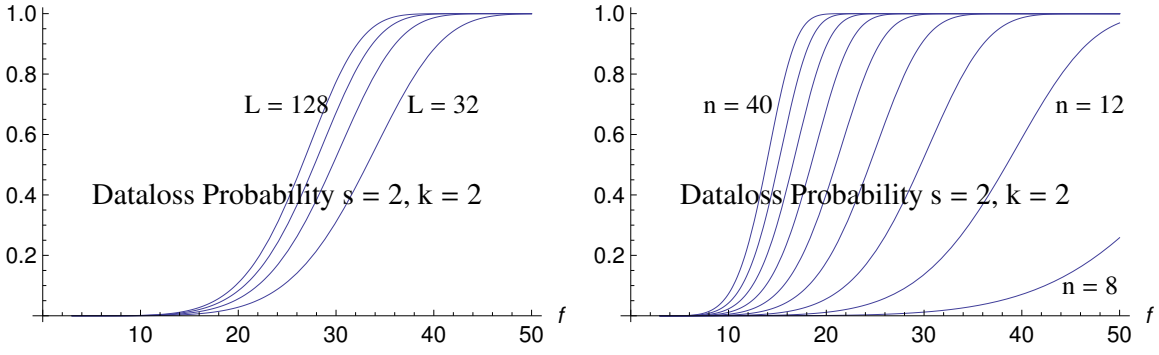
that are kept powered down unless needed for a write. Table II outlines the cost and corresponding write rates of different super-parity devices. Regardless of the super-parity device we use, we want to store super-parity on devices smaller than the data disks for increased bandwidth.

While flash is more reliable than disk, there are few enough super-parity devices 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 a very write-heavy workload, solid state drives (SSDs) may be attractive, even though they are initially expensive. One advantage of our design is that super-parity can be stored on an assortment of devices as technologies and financial situations change.

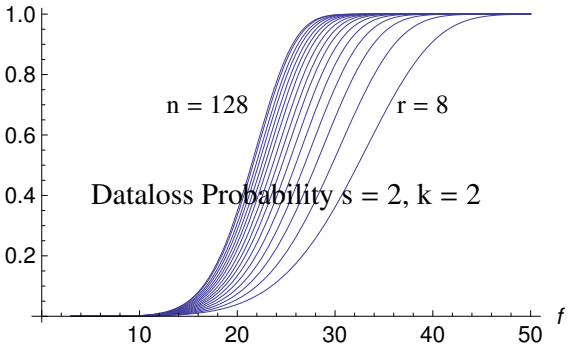
The running cost of the additional disk is mainly the power it consumes. While running additional disks in a data center certainly makes the entire data center warmer, whether or not this will incur an increase in cooling costs is difficult to know [27]. On a power-aware archival system, we expect disks 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 for storing super-parity on SATA drives adds an overhead of about 0.7%, which we believe is a small enough overhead that the reliability increase is well worth the cost.

VII. FUTURE WORK

We see several possible extensions to this project. First, our current analysis is restricted to homogeneous, static codes. Different coding structures could allow us to encode our domain knowledge about the reliability of different sets of disks. For example, newer disks are more likely to fail than older, burned in disks. Thus, we may want to give stripes with many young disks extra parity to protect against the higher probability of failure. Alternatively, 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



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



(c) $r = 8, 16, \ldots, 128$

Fig. 7. Data Loss Probabilities for varying parameters from $k = 2$, $s = 2$, $D = 1024$, $n = 16$, $r = 16$,and $L = 64$.

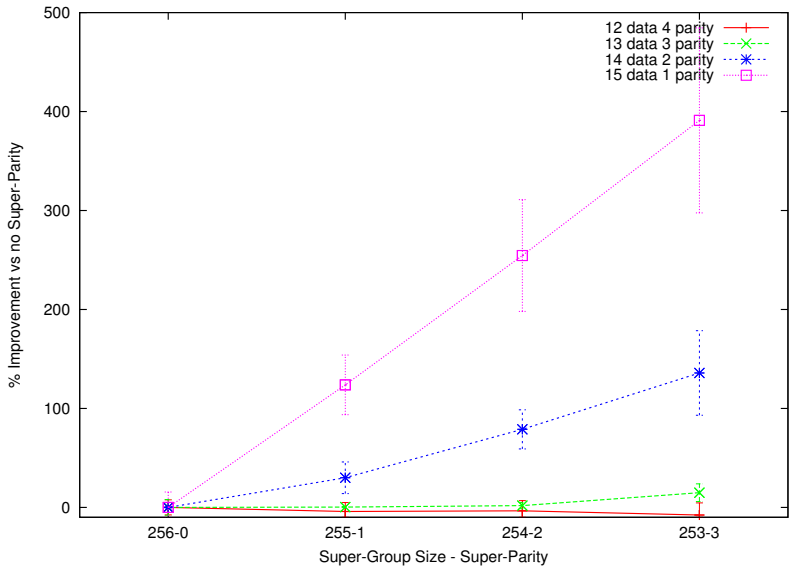
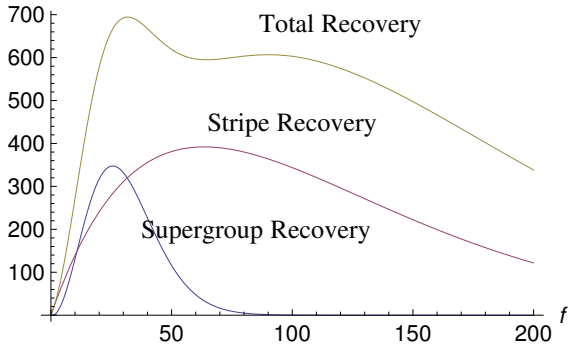


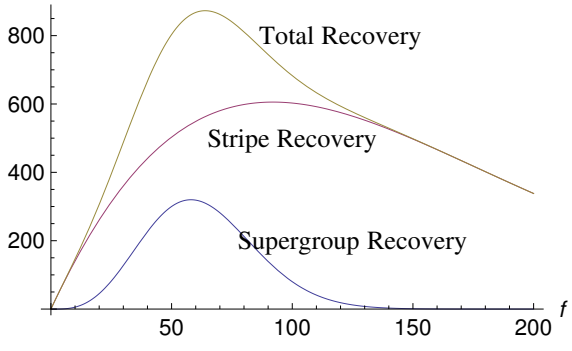
Fig. 12. Percent increase in MTTDL with super-parity vs. no super-parity

the unstable region and only one in the stable for less cost than adding three-disk parity across the entire stripe.

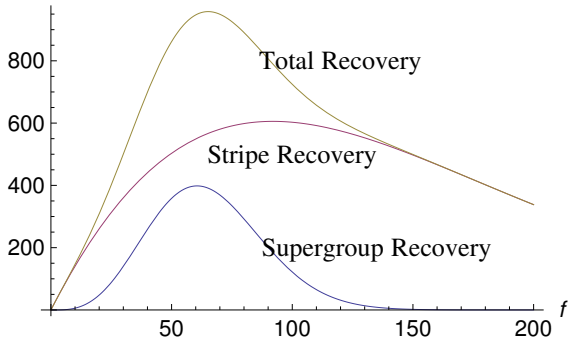
One major limitation to our analysis is the lack of experimental data covering correlated failures in large storage systems. As we discussed in Section I, we believe that the sorts of rare-event failures we protect against most likely have a single origin such as a cooling malfunction, disk batch failure, fire, or natural disaster. While super-parity can not provide the same level of reliability as multi-site redundancy, knowing the prevalence and distribution of correlated failures would enable us to better define the compromise between costs and reliability in an archival system that uses super-parity.



(a) $k = 1, s = 1$



(b) $k = 2, s = 1$



(c) $k = 2, s = 2$

Fig. 8. Rebuild workloads for different parity configurations with $n = 16$, $r = 16$, $D = 1024$, and $L = 64$. Workload is defined as the total number of disks accessed across all reads and writes.

VIII. Conclusion

We have presented a powerful technique for increasing the reliability of an archival system for little cost. In our analysis, we saw that using super-parity allowed us to tolerate several additional disk failures over a system with comparable storage overhead. In the simulation, we see up to a four-fold improvement in MTTDL after adding super-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 super-parity is a good value proposition for archival storage systems.

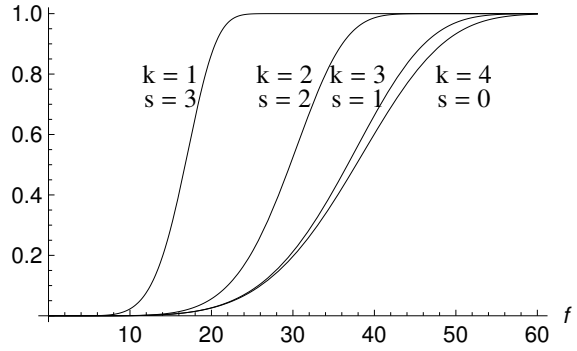


Fig. 9. Naïve comparisons between coding schemes. ($n = 16$, $r = 16$, $D = 1024$, and $L = 64$.)

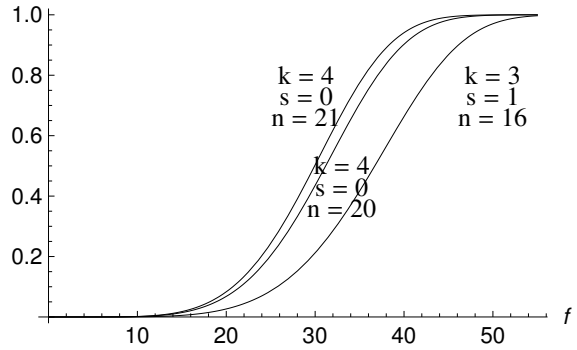


Fig. 10. Comparisons between coding schemes with approximate equal storage overhead. ($r = 16$, $D = 1024$, and $L = 64$.)

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!. Dr. Schwarz was supported by a grant from the Stiles Family Fund.

References

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

[2] 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*, Apr. 2006, pp. 221–234.

[3] 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*, vol. 44, no. 2, pp. 192–202, 1995.

[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*, vol. 17, no. 1-2, pp. 58–74, 1993.

[5] D. Colarelli and D. Grunwald, "Massive arrays of idle disks for storage archives," in *Proceedings of the 2002 ACM/IEEE Conference on Supercomputing (SC '02)*, Nov. 2002.

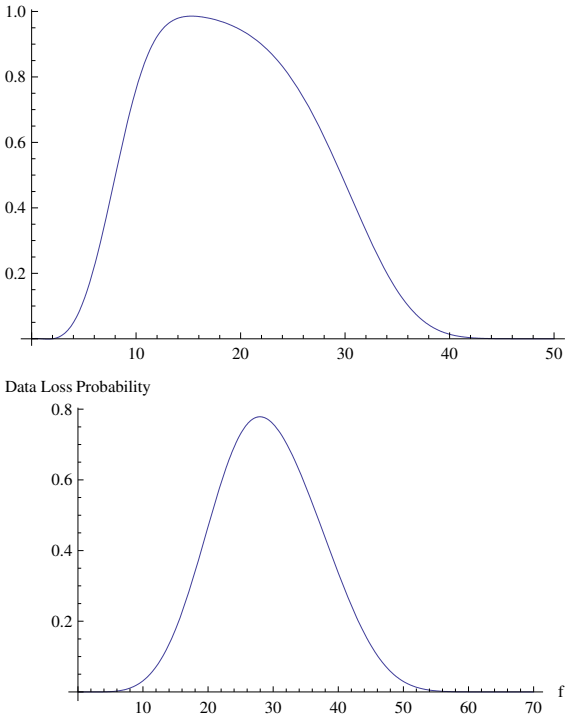


Fig. 11. Probability of Reconstruction using the super-parity: $k = 2, s = 2$ (top) and $k = 3, s = 1$ (bottom), $n = 16$, $r = 16$, $L = 64$.

[6] P. Corbett, B. English, A. Goel, T. Grcanac, S. Kleiman, J. Leong, and S. Sankar, "Row-diagonal parity for double disk failure correction," in *Proceedings of the Third USENIX Conference on File and Storage Technologies (FAST)*, 2004, pp. 1–14.

[7] 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*. ACM New York, NY, USA, 2006, pp. 373–374.

[8] J. Gantz, C. Chute, A. Manfrediz, S. Minton, D. Reinsel, W. Schlichting *et al.*, "The Diverse and Exploding Digital Universe," *IDC White Paper*, vol. 2, 2008.

[9] 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: ACM, Oct. 2003.

[10] 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*, Jun. 2000, pp. 21–34.

[11] 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*. ACM New York, NY, USA, 2007, pp. 31–36.

[12] ——, "Disaster recovery codes: increasing reliability with large-stripe erasure correcting codes," in *Proceedings of the 2007 ACM workshop on Storage security and survivability*. ACM New York, NY, USA, 2007, pp. 31–36.

[13] 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*, 2007, pp. 79–86.

[14] I. Iliadis, R. Haas, X.-Y. Hu, and E. Eleftheriou, "Disk scrubbing versus intra-disk redundancy for high-reliability RAID storage systems," in *Proceedings of the 2008 SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, Jun. 2008, pp. 241–252.

[15] R. Kotla, L. Alvisi, and M. Dahlin, "SafeStore: a durable and practical storage system," in *Proceedings of the 2007 USENIX Annual Technical Conference*, Jun. 2007, pp. 129–142.

[16] 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: ACM, Nov. 2000.

[17] 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.

[18] M. Li, J. Shu, and W. Zheng, "Grid codes: Strip-based erasure codes with high fault tolerance for storage systems," *Trans. Storage*, vol. 4, no. 4, pp. 1–22, 2009.

[19] W. Litwin, R. Moussa, and T. Schwarz, "LH*RS – a highly-available scalable distributed data structure," *ACM Transactions on Database Systems*, vol. 30, no. 3, pp. 769–811, 2005.

[20] 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*, 2000, pp. 663–671.

[21] 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.

[22] 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.

[23] J. Plank, "The RAID-6 liberation codes," in *Proceedings of the 6th USENIX Conference on File and Storage Technologies table of contents*. USENIX Association Berkeley, CA, USA, 2008.

[24] J. Plank and M. Thomason, "A practical analysis of low-density parity-check erasure codes for wide-area storage applications," in *DSN-2004: The International Conference on Dependable Systems and Networks*. IEEE, 2004, pp. 115–124.

[25] 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)*. USENIX, Jan. 2002, pp. 231–244.

[26] 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)*, Oct. 2004, pp. 409–418.

[27] R. Sharma, C. Bash, C. Patel, R. Friedrich, and J. Chase, "Balance of power: Dynamic thermal management for internet data centers," *IEEE Internet Computing*, vol. 9, no. 1, pp. 42–49, 2005.

[28] SimPy Team, "SimPy homepage," http://simpy.sourceforge.net/, 2007.

[29] 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.

[30] 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.

[31] 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: USENIX, Nov. 2006.

[32] Q. Xin, E. L. Miller, and T. J. E. Schwarz, "Evaluation of distributed recovery in large-scale storage systems," in *Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing (HPDC)*, Honolulu, HI, Jun. 2004, pp. 172–181.