Author Guidelines for MSS Symposium Proceedings

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

Rodney Van Meter
Nokia
rdv@alumni.caltech.edu

Paolo Ienne
Swiss Federal Inst. of Technology
Paolo.Ienne@di.epfl.ch

Abstract

The abstract is to be in fully-justified italicized text, at the top of the left-hand column, below the author and affiliation information. Use the word "Abstract" as the title, in 12-point Times, boldface type, centered relative to the column, initially capitalized. The abstract is to be in 10-point, single-spaced type. The abstract may be up to 3 inches (7.62 cm) long. Leave two blank lines after the Abstract, then begin the main text.

1. Introduction

Please follow the steps outlined below when submitting your manuscript to the publications chairs.

2. Instructions

Please read the following instructions carefully. You may use this LaTeX document as a template if you're using LaTeX to prepare your document. There is also a Microsoft Word template if you prefer. However, you may use any word processor you like, as long as you follow the guidelines listed here. Note that all manuscripts must be in English.

2.1. Acceptable formats

Your paper *must* be submitted electronically. The only acceptable formats for your final paper are PDF (preferred) or PostScript. PostScript in particular can be finicky; if your PostScript doesn't convert to PDF at our site, we will ask you to provide PDF directly.

2.2. Margins and page numbering

All printed material, including text, illustrations, and charts, must be kept within a print area 6-7/8 inches (17.5 cm) wide by 8-7/8 inches (22.54 cm) high. Nothing should be printed outside this area, *including page numbers*. Your paper should not have page numbers!

2.3. Formatting your paper

All text must be in a two-column format. The total allowable width of the text area is 6-7/8 inches (17.46 cm) wide by 8-7/8 inches (22.54 cm) high. Columns are to be 3-1/4 inches (8.25 cm) wide, with a 3/8 inch (0.95 cm) space between them. The main title (on the first page) should begin 1-3/8 inch (3.49 cm) from the top edge of the page. The second and following pages should begin 1.0 inch (2.54 cm) from the top edge. On all pages, the bottom margin should be 1-1/8 inches (2.86 cm) from the bottom edge of the page for 8.5 $\times$ 11-inch paper; for A4 paper, approximately 1-5/8 inches (4.13 cm) from the bottom edge of the page.

2.4. Type-style and fonts

Wherever Times is specified, Times Roman may also be used. If neither is available on your word processor, please use the font closest in appearance to Times that you have access to. If possible, use only Times, Helvetica, Courier, Symbol, and Dingbats fonts in your paper. Any other fonts must be embedded in your document to ensure that it will print properly. LaTeX users: you *must* use the `times` package; the Computer Modern Roman font produces poor PostScript.

2.5. Main title

Center the title 1-3/8 inches (3.49 cm) from the top edge of the first page. The title should be in Times 14-point, boldface type. Capitalize the first letter of nouns, pronouns, verbs, adjectives, and adverbs; do not capitalize articles, coordinate conjunctions, or prepositions (unless the title begins with such a word). Leave two blank lines after the title.

Table 1. Sample table.

Level	Size	Style	Before	After
1	12 point	Bold	1 line	1 line

Stuff that goes into the figure itself.

Figure 1. Example of short caption.

Things that go bump in the night.

Figure 2. Example of long caption requiring more than one line.

2.6. Author names and affiliations

Author names are to be centered beneath the title and printed in Times 12-point, non-boldface type. Affiliations and email addresses are to be below each author's name, and set in Times 12-point italic type. If all authors have the same affiliation, the affiliation can appear centered below all authors' names. This information is to be followed by two blank lines.

2.7. Paper body

The body of the paper (including the abstract) must be in a two-column format.

The main text should be set in 10-point Times, single-spaced. All paragraphs should be indented 1 pica (approx. 1/6 inch or 0.422 cm). Make sure your text is fully justified—that is, flush left and flush right. Please do not place any additional blank lines between paragraphs.

2.8. Figures and tables

Figure and table captions should be 10-point Helvetica boldface type as shown in Figures 1 and 2. Short captions (single line) should be centered, as in Figure 1. Long captions should be aligned on both sides and indented 1 pica on both left and right, as demonstrated in Figure 2. Figure captions should be below their figures, while table captions should be *above* the table, as shown in Table 1.

Figures and tables must be centered, and should be placed in a single column if possible; however, large figures and tables may span the entire page if necessary. Page-spanning figures and tables should be at the top or bottom of the page. Figures and tables *must* be included as part of the PostScript or PDF document—no cut and paste!

We will be printing the proceedings in black and white. However, color in your document is acceptable as long as the black and white version is still readable and comprehensible; the CD-ROM will allow readers to see color figures.

3. First order headings

First-order headings should be Times 12-point boldface, flush left, with one blank line before, and one blank line after. Only the first word of first-, second-, and third-order headings should be capitalized. All numbers in the section number should be followed by periods.

3.1. Second order headings

Second order headings should be Times 11-point boldface, flush left, with one blank line before, and one after.

3.1.1. Third order headings. If you require a third-order heading (we discourage it), use 10-point Times, boldface, flush left, preceded by one blank line, followed by a period and your text on the same line.

3.2. Footnotes

Please use footnotes sparingly[1]; instead, include necessary peripheral observations in the text (within parentheses, if you prefer).

3.3. References

List and number all bibliographical references in 9-point Times, single-spaced, at the end of your paper. When referenced in the text, enclose the citation number in square brackets, for example [1]. Multiple citations [1, 2] should be in a single set of square brackets.

3.4. Copyright forms

You must include your signed IEEE copyright release form when you submit your finished paper. We MUST have this form before your paper can be published in the proceedings.

3.5. Conclusions

Please direct any questions to your session chair or tothe publications chairs Ethan Miller (elm@cs.ucsc.edu) or Rod Van Meter (Rod.VanMeter@nokia.com).

References

[1] I. M. Author. Some related article I wrote. *Some Fine Journal*, 99(7):1–100, January 1999.

[2] A. N. Expert. *A Book He Wrote*. His Publisher, Erewhon, NC, 1999.

[1] Or, better still, try to avoid footnotes altogether. If you do use footnotes, place them at the bottom of the column on the page on which they are referenced using 8-point single-spaced Times.

Title of the Paper

Ann Author
Foo University
ann@foo.edu

Joe Researcher
Company, Inc.
joe@company.com

Chris Scientist
The Research Lab
chris@trl.gov

Reliability Mechanisms for Very Large Storage Systems

Qin Xin[†]
University of California, Santa Cruz
qxin@cs.ucsc.edu

Ethan L. Miller[†]
University of California, Santa Cruz
elm@cs.ucsc.edu

Darrell D. E. Long[†]
University of California, Santa Cruz
darrell@cs.ucsc.edu

Scott A. Brandt[†]
University of California, Santa Cruz
scott@cs.ucsc.edu

Thomas Schwarz[*]
Santa Clara University
tschwarz@calprov.org

Witold Litwin
Université Paris 9 Dauphine
Witold.Litwin@dauphine.fr

Abstract

Reliability and availability are increasingly important in large-scale storage systems built from thousands of individual storage devices. Large systems must survive the failure of individual components; in systems with thousands of disks, even infrequent failures are likely in some *device. We focus on two types of errors: nonrecoverable read errors and drive failures. We discuss mechanisms for detecting and recovering from such errors, introducing improved techniques for detecting errors in disk reads and fast recovery from disk failure. We show that simple RAID cannot guarantee sufficient reliability; our analysis examines the tradeoffs among other schemes between system availability and storage efficiency. Based on our data, we believe that two-way mirroring should be sufficient for most large storage systems. For those that need very high reliabilty, we recommend either three-way mirroring or mirroring combined with RAID.*

1. Introduction

System designers have been making computer systems with better performance, lower cost, and larger scale over the last 20 years, but high availability and reliability issues have been somewhat neglected in most designs. Siewiorek and Swarz [15] noted four reasons for an increasing concern on fault tolerance and reliability: harsher environments, novice users, increasing repair costs, and larger systems. For storage systems, increasing the number of devices increases the likelihood that failure will occur in at least one device in the system, potentially causing loss of data.

We are primarily concerned with two types of failures: disk failures and nonrecoverable read errors. In petabyte-scale file systems, disk failures will be a daily (if not more frequent) occurence—data losses with this frequency cannot be tolerated. Moreover, disk rebuild times are becoming longer as disk capacity outpaces bandwidth [7], increasing the “window of vulnerability” during which a second disk failure would cause data loss from a RAID.

Nonrecoverable read errors are also a problem of increasing importance. Disk drives such as the Seagate ST3146807LW quote nonrecoverable read error rates of 1 in 10^{15} bits. At 100 GB/second aggregate bandwidth (20 MB/s per disk), nonrecoverable read errors occur several times per hour across the whole system even when a disk has not completely failed.

It is unacceptable to lose data in large storage system simply because a few components have failed. Our work explores the causes of such failures and presents redundancy mechanisms to reduce data loss when drives do fail.

We investigate the availability issues in a very large-scale storage system built from object-based storage devices (OBSDs) [17]. An OBSD is a network-attached storage device [6] that presents an interface of arbitrarily-named objects of variable size. In our OBSD storage system, files are broken into objects, and the objects are then distributed across many devices. The total storage capac-

[†] Supported in part by Lawrence Livermore National Laboratory, Los Alamos National Laboratory, and Sandia National Laboratory under contract B520714.

[*] Supported in part by IBM Research Grant 41102-COEN-RSCH-IG-IG09.

ity of this OBSD system is expected to be 2 PB[1]. In such a large-scale distributed storage system with thousands of nodes, there is a high likelihood that at least one node will be down at any given time.

Data for a single file will be distributed across hundreds of individual OBSDs. This distribution must be done in a decentralized fashion so that there is no central bottleneck when a single file is accessed by thousands of clients. Individual OBSDs can manage low-level storage allocation themselves, leaving the problems of allocating data to OBSDs and replication to the higher-level file system

If an OBSD consists of multiple drives, we can use RAID or other redundancy mechanisms within the OBSD to safeguard data. This approach is feasible and could increase the availability of a single OBSD. However, it is expensive because of the need for custom hardware and software to manage the redundancy and the need for a higher-speed network to connect the collection of drives. Additionally, RAID hardware and software are often complex and have greater chance of failures. If the non-disk hardware or software in an OBSD fails, the data is unavailable, though recovery time from such faults is often lower than that for rebuilding a failed disk. An alternate approach would be to put the OBSD software on each drive and attach it directly to the network. This approach might be less expensive in the long run, but would require more redundancy between OBSDs to provide a comparable level of reliability.

A guiding principle in our research is that disk failures should not significantly affect overall system behavior by losing data or significantly decreasing bandwidth if the file system is expected to be in a production system. If the system contains 4000 disks with 500 GB per disk, and each disk has a mean time to failure (MTTF) of 10^5 hours, a disk will fail every 25 hours on average. Simply using RAID is not enough because it takes too long to rebuild a disk. Assuming the RAID must still provide data to clients while rebuilding, it would take more than a day to rebuild a 500 GB disk at 5 MB/second. With 5 disks in a RAID, the chance of a second failure during a one day rebuild is about 0.1%, resulting in a mean time to data loss (MTTDL) from a 2 PB system of less than three years.

As disk prices keep dropping, so does the cost of providing redundant storage, allowing us to adopt more reliable replication mechansims including RAID, mirroring, or both. Our research is concerned with providing guidance to find the right balance between cost and reliability.

2. Related work

There has been some research beyond RAID [4] in reliability and recovery for large-scale systems, though most of it has focused on the use of storage in wide-area systems. For example, Oceanstore [9, 18] is designed to have a very long MTTDL, but at the cost of dramatically increasing the number of disk requests per block written. Pangaea [13] allows individual servers to continue serving most of their data even when disconnected; however, this approach is designed for wide-area systems, and does not permit files to be striped across dozens of servers for higher bandwidth. Microsoft's Farsite project [1, 5] investigated the issue of replication for systems built from relatively unreliable commodity workstations. They focused on reliability, investigating replica placement strategies that take server reliability into account. However, Farsite is not designed for high-bandwidth applications, and must deal with servers that are less reliable than those in a single large-scale file system.

Other researchers have studied the question of how much replication is really necessary as well as techniques to reduce that level of replication. The Recovery Oriented Computing project [12] is trying to reduce the recovery time in order to get higher availability and lower total cost of ownership. Castro and Liskov [3] propose a secure replication system to tolerate Byzantine faults and make the window of vulnerability small. Litwin and Schwarz [10] present a family of linear hashing methods for distributing files. The algorithms reduce the number of messages and scale to the growth or shrink of files efficiently. Schwarz [14] has built a Markov model to estimate system availability; we will apply this model to the scalable file system we are designing.

3. Nonrecoverable errors

By storing vast quantities of data on commodity disks, we will reach the point where the built-in error detection (and correction) of the disks no longer prevents undetected errors. For example, error rates of one undetected error of 1 in 10^{15} bits are common. A single drive running at 25 MB/s would experience such an error about once a year. In a large system with 10,000 disks, however, such an error will occur once per hour *somewhere* in the system. While the rate is too small to worry about in a typical commercial database where human errors far outnumber machine-generated errors, there are applications where very large amount of data need to be stored without any data corruption, *e.g.* nuclear test simulation data files for the United States Department of Energy.

If we were to change the error control code that hard

[1] A petabyte (PB) is 10^{15} bytes.

drives deploy to detect and (hopefully) correct slight data corruption so that internal errors are flagged instead of corrected, the hard drive would have far more nonrecoverable read errors, but the rate of an undetected read error would decrease substantially. Obviously, this is not possible for commodity hard drives and would even be complicated for specialized hard drives since the ECC and the magnetic code are increasingly interwoven [8]. We could embed the application data into an ECC, but the storage overhead is high.

3.1. Signature scheme

We have designed a scheme that detects and corrects small errors in blocks stored on disk in a large-scale system. Our scheme consists of two components: a signature scheme that *flags* corrupted data, and a RAID 5-like [4] mechanism that creates groups of blocks spread across different disks and stores the parity of the block on yet another disk. This redundancy allows us to reconstruct any corrupted block of data.

To flag corrupted data, we associate a *signature* with each data block. The signature is a bit string of fixed length f that is calculated from the contents of the block. A signature resembles a hash function, and should change if the block is only slightly changed. We calculate the signature when we store a data block and store the signature on the same disk separately from the data block. When we read the data block, we recompute the signature and compare it with the previously stored signature value. If the two signature values agree, we conclude that the block is correct; otherwise, we flag an error. This error is likely caused by an incorrect block, but could instead be caused by an incorrect signature.

To correct corrupt data, we introduce redundancy into the storage scheme. Blocks are assigned into redundancy sets, each of which contains n or fewer blocks. In addition, each set contains k parity blocks. The parity blocks are calculated using a standard erasure correction code. The parameter k is a constant of the file system. If k is one, then the single parity block is the normal (XOR) parity of the n blocks containing application data. If $k = 2$, a modern erasure correction code such as Even-Odd is used to calculate the two parity blocks. Beyond this, we can use generalized Reed-Solomon codes or other codes [2, 14]. We might want to use $k > 1$ because this also protects the data against disk failure.

3.2. Block signatures

A good signature scheme has the following properties:

1. The probability that two random blocks have the same signature is constant 2^{-f}.

2. The signature changes for small changes in the contents of the block.

3. The signature can be calculated using a simple scan of the block.

4. The signatures of a compound object can be calculated from the signature of constituent objects, *e.g.* a *file signature* can be calculated from the file block signatures. The file signature changes if two blocks are interchanged.

5. The signature scheme can use arbitrary f within a wide range.

We can construct such a signature using Galois fields [11], as follows. We break a block of length g bits into s bit symbols just as a written word in a western language is broken into characters. s is a constant of the scheme, with $s = 8$ or $s = 16$ being good choices. Each symbol is interpreted as an element of the Galois field with 2^s elements. A definition of Galois fields is beyond the scope of this paper; however, much of the complex math can be precalculated and stored in lookup tables, allowing signatures to be computed quickly.

We pick an element α of the Galois field such that the powers of α make up all the non-zero elements of the field; such an element is called primitive. Formally, we define the α signature of a block $B = (b_0, b_1, \cdots, b_{l-1})$ as

$$sig_\alpha(b_0, b_1, \cdots, b_{l-1}) = \sum_{i=0}^{l-1} \alpha^i b_i$$

This signature condenses a block of ls bits into s bits, but this signature is not long enough for our purposes. We therefore use a compound signature made up of several components:

$$\begin{gathered} sig_{\alpha_0,\alpha_1,\cdots,\alpha_{t-1}}(b_0, b_1, \cdots, b_{l-1}) = \\ \big(sig_{\alpha_0}(b_0, b_1, \cdots, b_{l-1}), \\ sig_{\alpha_1}(b_0, b_1, \cdots, b_{l-1}), \cdots, \\ sig_{\alpha_{t-1}}(b_0, b_1, \cdots, b_{l-1})\big) \end{gathered}$$

This signature yields ts bits, with $t = 4$ and $s = 8$ being typical values that yield a 32 bit signature. We use a single table lookup into a table with 2^{s+1} entries together with a single zero comparison and an integer addition to avoid the use of a logarithm and antilogarithm table.

The compound signature has several attractive properties, but is not cryptographically secure—indeed, it has these properties precisely because it is not secure. We can define a file signature in a completely analogous way. If the file is stored in several blocks with possible zero-padding, then the file signature can be quickly calculated from the block signature. The signature is guaranteed to change if up to t symbols are changed. We can calculate the signature of a block after the swap of two sub-strings from the sub-string signature. Finally, we can calculate the signature of a parity block (generated by XOR parity or general Reed-Solomon coding) from the data blocks. Obviously, the signature can be calculated in a single sweep of the block while using little memory. All of these properties make this signature attractive for use in a large-scale system where data may be distributed across multiple OBSDs.

4. Disk failures

Protecting against nonrecoverable disk errors guarantees that individual disk blocks read from "working" are correct, but does not handle the case that an entire disk has failed. As discussed in Section 1, a 2 PB storage system will experience about one disk failure per day. Given the long rebuild times that current techniques require, disk failures would hurt performance and result in unacceptably high risk of data loss from a second disk failure. We investigated several redundancy mechanisms and developed mechanisms for fast recovery in a large-scale storage system, resulting in lower risk of data loss at an acceptable storage overhead.

4.1. Redundancy mechanisms

We assume that our storage system holds 2 PB of data, and is built from 500 GB disk drives. Such drives are not yet available, but will be by 2004–2005, assuming that current growth rates in disk capacity [7] continue. Note that a storage system containing 2 PB of data will require more capacity for redundancy. The ratio between data capacity and total storage capacity is the *storage efficiency*. We further assume that our disks have a mean time to failure (MTTF) of 10^5 hours. This is significantly shorter than that specified by drive manufacturers, but is longer than the 50,000 hours reported by sites such as the Internet Archive [16]. For simplicity, we assume the failure rates of the disks in the system are identical and independent, though this may not be true if many disks are from the same manufacturing batch.

We use the term *redundancy set* to refer to a block group composed of data blocks and their associated replicas or parity blocks. A single redundancy set will typically contain 1 MB to 1 TB. We consider three methods to configure the organization of a redundancy set: two-way mirroring (Mirror-2), three-way mirroring (Mirror-3), and RAID-5 with mirroring (RAID 5+1). In n-way mirroring, each data block in the redundancy set is stored n times, with each replica stored on a different OBSD. Under RAID 5+1, each OBSD consistes of multiple disks organized as a RAID 5, and each data block is mirrored on two OBSDs. Table 1 summarizes the cost and storage efficiency of the three redundancy schemes using a current disk price of \$1/GB and an estimated disk price of \$0.1/GB in 2005.

Table 1. Cost and overhead of different reliability mechanisms.

Method	Cost ($million)		Storage Efficiency
	2002	2005	
Mirror-2	$2	$0.2	50%
Mirror-3	$4	$0.4	33%
RAID 5+1	$3	$0.3	40%

4.2. Fast recovery mechanisms

When we use any of the redundancy mechanisms described above, there is still a small chance that the system will lose data. For example, Mirror-3 will fail when two of the three OBSDs in the redundancy set fail and the third fails while the other two are being rebuilt. We can use one of two mechanisms to deal with this situation: Fast Mirroring Copy (FMC) and Lazy Parity Backup (LPB).

Fast Mirroring Copy quickly provides additional redundancy for a redundancy set that has lost one copy. Rather than attempting to immediately rebuild an entire disk, FMC recreates the lost replicas throughout the storage system. Since the "peer" for each replica that was on the lost disk is on a different OBSD, this process takes time proportional to the size of a single redundancy set. For example, it would take only 100 seconds to create a new replica for a single 500 MB redundancy set. In a system in which a 500 GB disk contains replicas for 1000 redundancy sets, this technique could rebuild a missing disk within 2 minutes. The rebuilt disk is distributed across the entire storage system, but all of the data is protected as if the original disk had not failed.

Lazy Parity Backup has the same goal—protecting data by replication—but works by creating parity blocks in the background when the associated data blocks have not been modified for some time. This method can be used to create RAID-like structures across OBSDs. If FMC is used for rapidly-changing data, LPB can be used for more static data to gain the same reliability with lower storage overhead. This technique is somewhat similar to AutoRAID [19], but operates on a far larger storage system.

Table 2. Parameters for a 2 PB storage system.

Parameter	Value
Z (total data in the system)	2 PB
γ (recovery rate)	10^2 GB/hr
N (number of redundancy sets)	Z/S
$MTTF_{disk}$	10^5 hours
D (disks in an OBSD RAID 5)	5
S (data in one redundancy set)	varies

4.3. System availability

Using Markov models, we compared the mean time to data loss in a 2 PB of Mirror-2, Mirror-3, and RAID 5+1. The parameters of the storage system we used in our comparison is listed in Table 2.

To compare the MTTDL of the three redundancy schemes, we used the following equations which approximate (to within 1%) the MTTDL for each of the redundancy mechanisms[2]:

$$MTTDL_{Mirror-2} = \frac{MTTF_{disk}^2 \cdot \gamma}{2 \cdot Z} \tag{1}$$

$$MTTDL_{Mirror-3} = \frac{MTTF_{disk}^3 \cdot \gamma^2}{3 \cdot S \cdot Z} \tag{2}$$

$$MTTDL_{RAID\ 5+1} = \frac{(D-1)^2 \cdot MTTF_{disk}^4 \cdot \gamma^3}{2 \cdot D \cdot S^2 \cdot Z} \tag{3}$$

Using Equations 1, 2, and 3 and Table 2, we calculated the MTTDL for each redundancy mechanism using different sizes for a single redundancy set, as shown in Figure 1. For each mechanism, we show MTTDL for both $MTTDL_{disk} = 10^5$ hours and the manufacturers' claims of $MTTDL_{disk} = 10^6$ hours.

Our first result is that MTTDL for the system does not vary with the size of a redundancy set for Mirror-2, as expected from Equation 1. Though larger redundancy sets require more time for recovery, there are fewer of them. These two effects are balanced for Mirror-2, so the size of a single redundancy set does not affect overall MTTDL. For Mirror-3 and RAID 5+1 mechanisms, however, MTTDL decreases as the size of a single redundancy set increases. In both cases, the decrease in reliability due to longer recovery time overwhelms the increased reliability from having fewer, larger redundancy sets.

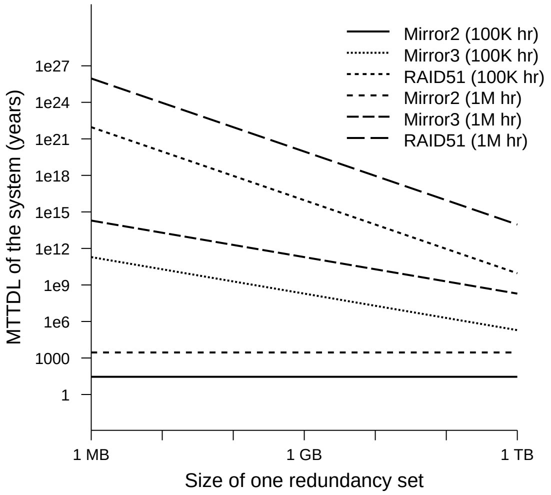

Figure 1. Mean time to data loss in a 2 PB storage system

Figure 1 seems to indicate that smaller redundancy sets provide longer MTTDL. However, this approach has several limitations. First, overall file system bandwidth will decrease if redundancy sets are too small because individual disk transfers will be too small. This sets a lower limit of 256 KB–4 MB for redundancy sets. Second, we assume that redundancy sets fail independently. If there are too many redundancy sets, however, many will share multiple disks, causing correlated failures. Third, the bookkeeping necessary for millions of small redundancy sets will be overwhelming. For all these reasons, we believe it unlikely that redundancy sets will be much smaller than 200 MB–1 GB.

Disk lifetime is another important factor in calculating MTTDL for the entire storage system. An order of magnitude in $MTTDL_{disk}$ from 10^5 hours to 10^6 hours can improve overall MTTDL by a factor of 100 for Mirror-2, 1000 for Mirror-3, and 10,000 for RAID 5+1. The use of a controlled environment to ensure longer disk lifetimes will result in a major benefit in overall system reliability.

Increasing the recovery rate γ can also improve overall system reliability. Placing a higher priority on disk transfer rate and recovering faster will greatly improve reliability by reducing the "window of vulnerability" during which the system may lose data. Doubling the recovery rate will double the reliability of Mirror-2, but will increase the reliability of RAID 5+1 by a factor of 8.

For a system with 2 PB of storage, we believe that Mirror-2 will provide sufficient redundancy at an acceptable cost. MTTDL for such a system will be about 30 years regardless of the redundancy set size, allowing us to use larger redundancy sets to reduce bookkeeping over-

[2] Due to space limitations, we omit the derivations, which will be included in the full paper.

head. Mirror-3 and RAID 5+1 can provide much longer MTTDL—up to 2×10^{11} years for Mirror-3, and 10^{22} years for RAID 5+1. Although other schemes provide much greater MTTDL, Mirror-2 is considerably simpler to implement than Mirror-3 and RAID 5+1, and provides good reliability at relatively low cost.

5. Conclusions

We have discussed two major sources of data loss in large-scale storage systems—nonrecoverable read errors and disk failures—and have presented mechanisms for dealing with each. By using signatures on individual disk blocks and redundancy across multiple storage devices, we can reduce the risk of data loss. Techniques such as fast mirror copy further decrease the chance of data loss to the point where simple two-way mirroring in a 2 PB file system can still have a mean time to data loss of 30 years without the use of expensive RAID hardware. If this is not sufficiently long, techniques such as three-way mirroring and RAID 5 with mirroring can virtually guarantee that data will never be lost.

References

[1] A. Adya, W. J. Bolosky, M. Castro, R. Chaiken, G. Cermak, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, and R. Wattenhofer. FARSITE: Federated, available, and reliable storage for an incompletely trusted environment. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI)*, Boston, MA, Dec. 2002. USENIX.

[2] G. A. Alvarez, W. A. Burkhard, and F. Cristian. Tolerating multiple failures in RAID architectures with optimal storage and uniform declustering. In *Proceedings of the 24th International Symposium on Computer Architecture*, pages 62–72, Denver, CO, June 2002. ACM.

[3] M. Castro and B. Liskov. Proactive recovery in a Byzantine-fault-tolerant system. In *Proceedings of the 4th Symposium on Operating Systems Design and Implementation (OSDI)*, 2000.

[4] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson. RAID: High-performance, reliable secondary storage. *ACM Computing Surveys*, 26(2), June 1994.

[5] J. R. Douceur and R. P. Wattenhofer. Optimizing file availability in a secure serverless distributed file system. In *Proceedings of the 20th Symposium on Reliable Distributed Systems (SRDS '01)*, pages 4–13, New Orleans, LA, Oct. 2001. IEEE.

[6] G. A. Gibson, D. F. Nagle, K. Amiri, J. Butler, F. W. Chang, H. Gobioff, C. Hardin, E. Riedel, D. Rochberg, and J. Zelenka. A cost-effective, high-bandwidth storage architecture. In *Proceedings of the 8th Interational Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 92–103, San Jose, CA, Oct. 1998.

[7] J. L. Hennessy and D. A. Patterson. *Computer Architecture A Quantitative Approach*. Morgan Kaufmann Publishers, 3rd edition, 2003.

[8] A. S. Hoagland and J. E. Monson. *Digital magnetic recording*. Wiley, 2nd edition, 1991.

[9] 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 Interational Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Cambridge, MA, Nov. 2000. ACM.

[10] W. Litwin and T. Schwarz. LH*$_{\mathrm{RS}}$: A high-availability scalable distributed data structure using Reed Solomon codes. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 237–248, Dallas, TX, May 2000. ACM.

[11] F. J. MacWilliams and N. J. Sloane. *The Theory of Error Correcting Codes*. Elsevier Science B.V., 1983.

[12] D. A. Patterson, A. Brown, P. Broadwell, G. Candea, M. Chen, J. Cutler, P. Enriquez, A. Fox, E. Kiciman, M. Merzbacher, D. Oppenheimer, N. Sastry, W. Tetzlaff, J. Traupman, and N. Treuhaft. Recovery-Oriented Computing (ROC): Motivation, Definition, Techniques, and Case Studies. Technical Report UCB//CSD-02-1175, University of California, Berkeley, Mar. 2002.

[13] Y. Saito and C. Karamanolis. Pangaea: A symbiotic wide-area file system. In *Proceedings of the 2002 ACM SIGOPS European Workshop*. ACM, Sept. 2002.

[14] T. Schwarz. Generalized Reed Solomon codes for erasure correction in SDDS. In *Workshop on Distributed Data and Structures (WDAS 2002)*, Paris, France, Mar. 2002.

[15] D. Siewiorek and R.S.Swarz. *Reliable Computer Systems Design and Evaluation*. The Digital Press, 2nd edition, 1992.

[16] B. Tofel. Personal communication, Aug. 2002.

[17] F. Wang, S. A. Brandt, E. L. Miller, and D. D. E. Long. OBFS: A file system for object-based storage devices. Submitted to the 2003 Conference on File and Storage Technologies (FAST).

[18] H. Weatherspoon and J. Kubiatowicz. Erasure coding vs. replication: A quantitative comparison. In *Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS 2002)*, Mar. 2002.

[19] J. Wilkes, R. Golding, C. Staelin, and T. Sullivan. The HP AutoRAID hierarchical storage system. In *Proceedings of the 15th ACM Symposium on Operating Systems Principles (SOSP '95)*, pages 96–108, Copper Mountain, CO, 1995. ACM Press.

Telekinesis in Petabyte-Scale Storage Systems*

Yendor Q. Galollipop
University of Foo
yendor@foo.edu

Evad X. Galilypad
Bar , Inc.
evad@bar.com

Chris Frobnitz
Universal Dispensing Authority
frobnitz@uda.gov

Abstract

This is our abstract. In it we simply state that we've figured out how to use telekinesis in mass storage systems.

We'd say more here if there were more to say.

1. Introduction

Here we introduce the topic of telekinesis and explain its relevance to mass storage.

2. Our work

Here we explain what's nifty, keen, cool, rad and bad about our work. We're the first people in this part of the galaxy to use telekinesis as a key architectural component of mass storage systems.

2.1. Telekinetic bits

We figured out how to move bits using telekinesis. Fortunately, they are not heavy, so it was not hard.

2.2. Telekinetic readers and writers

The TKbits don't do any good if you can't read them and write them, so we figured out how, as shown in Figure 1.

2.2.1. Telekinetic readers. It's not that hard to read TKbits as long as you have the power.

2.2.2. Telekinetic writers. You can also write bits telekinetically if you know how.

3. Related work

Here's where we explain why it's niftier, keener, cooler, radder and badder than anything else in the known universe, including Fredkin gates, reversible computing and jellied beet consommé [1,2,3].

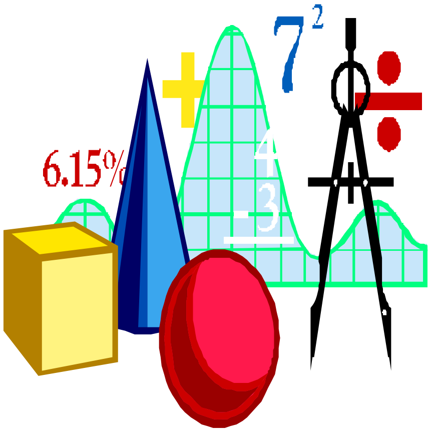

Figure 1. How to figure it out.

4. Future work

Here's where we explain how much of the work we really didn't get done before the paper was due.

5. Conclusions

Recap and indicate how totally cool it really all is. Be sure to state the conclusion: Necessity is the Mother of Invention [4].

References

[1] R. P. Feynman, *Feynman Lectures on Computation*. (Reading MA: Addison Wesley, 1996)

[2] I. S. Rombauer and M. R. Becker, *Joy of Cooking*. (New York, NY: Bobbs-Merrill, 1975 edition, 1931)

* This research was sponsored by the Universal Dispensing Authority under Contract No. UDA-1864-AbeLincoln-98. Views and conclusions contained in this report are the authors' and should not be interpreted as representing the official opinion or policies, either expressed or implied, of the UDA, the U.D. Government, or any person or agency connected with them. In fact, the authors aren't even responsible for this work.

[3] R. S. Mulliken, "Miss Molecule," *J Chem Phys*, **22** (1945) 345-355.

[4] Johannes Gutenbergus, “Dispensation Franchising and Neighborhood Loansharking: essential prerequisites to invention,” *Poggendorf’s Annalen der Druckindustrie* **1** (1644) 1-15.