Proc. 8th International Symposium on Stabilization, Safety and Securiity of Distributed Systems (SSS 2006), Dallas, TX, Nov. 2006, to appear

Self-Adaptive Disk Arrays

Jehan-François Pâris[1*], Thomas J. E. Schwarz [2], and Darrell D. E. Long [3*]

[1] Dept. Of Computer Science, University of Houston
Houston, TX 77204-3010
paris@cs.uh.edu
[2] Dept. of Computer Engineering, Santa Clara University
Santa Clara, CA 95053
tjschwarz@scu.edu
[3] Dept. of Computer Science, University of California
Santa Cruz, CA 95064
darrell@cs.ucsc.edu

Abstract. We present a disk array organization that adapts itself to successive disk failures. When all disks are operational, all data are mirrored on two disks. Whenever a disk fails, the array reorganizes itself, by selecting a disk containing redundant data and replacing these data by their exclusive or (XOR) with the other copy of the data contained on the disk that failed. This will protect the array against any single disk failure until the failed disk gets replaced and the array can revert to its original condition. Hence data will remain protected against the successive failures of up to one half of the original number of disks, provided that no critical disk failure happens while the array is reorganizing itself. As a result, our scheme achieves the same access times as a mirrored organization under normal operational conditions while having a much lower likelihood of loosing data under abnormal conditions. In addition it tolerates much longer repair times than mirrored disk arrays.

Keywords: fault-tolerant systems, storage systems, repairable systems, *k*-out-of-*n* systems.

1 Introduction

Today's disks have mean time to failures of more than ten years, which means that a given disk has a less than ten percent probability of failing during any given year of its useful lifetime. While this reliability level is acceptable for all the applications that only require the storage of a few hundreds of gigabytes of non-critical information over relatively short time intervals, it does not satisfy the needs of applications having to store terabytes of data over many years.

Backups have been the traditional way of protecting data against equipment failures. Unfortunately, they suffer from several important limitations. First, they do not scale well; indeed the amount of time required to make a copy of a large data set

* Supported in part by the National Science Foundation under award CCR-0204358.

can exceed the interval between daily backups. Second, the process is not as trustworthy as it should be due to both human error and the frailty of most recording media. Finally, backup technologies are subject to technical obsolescence, which means that saved data risk becoming unreadable after only ten to twenty years. A much better solution is to introduce redundancy into our storage systems

The two primary ways of introducing that redundancy are mirroring and *m*-out-of-*n* codes. Both techniques have their advantages and disadvantages. Mirroring offers the two main advantages of reducing read access times and having a reasonable update overhead. Identifying failed disks can always be done by detecting which replicas have become unavailable. On the other hand, *m*-out-of-*n* codes provide much higher data survivability. Consider, for instance, the case of a small disk array consisting of eight disks. A mirrored organization that maintains two copies of each file on separate disks would protect data against all single disk failures and most double disk failures. A simultaneous failure of three disks would have a bigger impact as it would result in data loss in 43 percent of the cases. This is much worse than an optimal 4-out-of-8 code that protects data in the presence of up to four arbitrary disk failures. In fact, this is such an improbable event that erasure codes that tolerate more than two simultaneous failures are never used in actual storage systems.

We propose a self-adaptive disk array organization that combines most of advantages of mirroring and erasure coding. As long as most disks are operational, it will provide the same read and write access times as a mirrored organization. Whenever a disk fails, it will reorganize itself and quickly return to a state where data are again protected against a single failure. As a result, data will remain protected against the consecutive failures of up to one half of the original number of disks, provided that no critical disk failure happens while the array is reorganizing itself. This is a rather unlikely event as the reorganization process will normally take less than a few hours.

The remainder of this paper is organized as follows. Section 2 will introduce our self-adaptive disk array organizations. Section 3 will compare the mean times to data loss (MTTDL) achieved by self-adaptive arrays with those achieved by mirrored disk arrays. Section 4 will review previous work and Section 5 will have our conclusions.

2 Our Approach

Consider the small disk array displayed on Fig. 1. It consists of four pairs of disks with data replicated on each pair of disks. For instance, disks A_1 and A_2 contain the same data set A. Assume now that disk B_1 fails. As a result, only one remaining copy of data set B remains and the array will become vulnerable to a failure of disk B_2. Waiting for the replacement of disk B_1 is not an attractive option as the process make take several days. To adapt itself to the failure, the array will immediately locate a disk containing data that are replicated elsewhere, say, disk A_1, and replace its contents by the exclusive or (XOR) of data sets A and B thus making the array immune to a single disk failure. Fig. 2 displays the outcome of that reconfiguration.

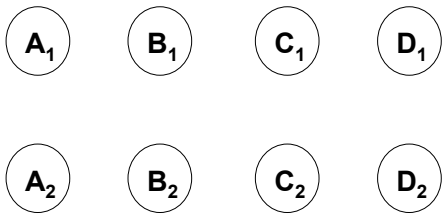

Fig. 1. A small disk array consisting of four pairs of disks with data replicated on each pair of disks.

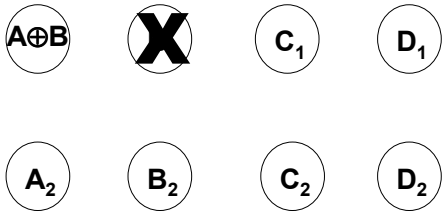

Fig. 2. The same disk array after disk B_1 has failed and the array is reconfigured.

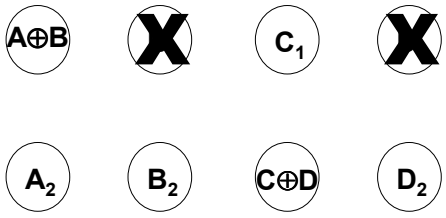

Fig. 3. The same disk array after disk D_1 has failed and the array is reconfigured.

The array can again tolerate any single disk failure. The sole drawback of the process is that accesses to data sets *A* and *B* will now be slightly slower. In particular, updates to these two data sets will be significantly slower as each update will now require one additional read operation. This condition is only temporary as the array will revert to its original condition as soon as the failed disk is replaced.

Consider now what would happen if a second disk failed, say, disk D_1, before disk B_1 was repaired. This second failure would remove one of the two copies of data set *D* and make the array vulnerable to a failure of disk D_2. To return to a safer state, the array will locate a disk containing data that are replicated elsewhere, say, disk C_2, and replace its contents by the exclusive or (XOR) of data sets C and D. Fig. 3 displays the outcome of this reorganization.

Under most circumstances, the two failing disks will be replaced before a third failure could occur. Delays in the repair process and accelerated disk failures resulting

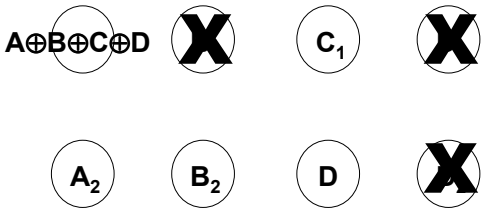

Fig. 4. The same disk array after disk D_2 has failed.

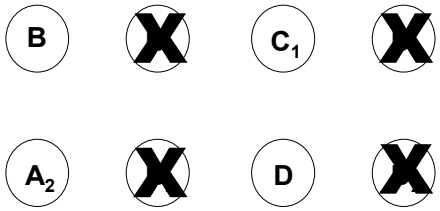

Fig. 5. The same disk array after disk B_2 has failed.

from environmental conditions could however produce the occurrence of a third disk failure before disks B_1 and D_1 are replaced. Let us assume that disk D_2 fails this time. Observe that this failure destroys the last copy of data set D. The fastest way to reconstitute this data set is to send the contents of disk C_1 to the disk that now contains $C \oplus D$ and to XOR the contents of these two disks *in situ*. While doing that, the array will also send the old contents of the parity disk, that is, $C \oplus D$, to the disk that contains $A \oplus B$ in order to obtain there $A \oplus B \oplus C \oplus D$. As seen on Fig. 4, the disk array now consists of four disks holding data and one parity disk.

Let us now consider for the sake of completeness the rather improbable case of a fourth disk failure occurring before any of the three failed disks can be replaced. Assume that disk B_2 fails this time. As Fig. 5 indicates, the sole option left is to reconstitute the contents of the failed disk by XORing the contents of the parity disk ($A \oplus B \oplus C \oplus D$) with those of disks A_2, C_1 and D and store these contents on the former parity disk. This would keep all four data sets available but would leave all of them vulnerable to a single disk failure.

In its essence, our proposal is to let the array adapt itself to the temporary loss of disks by switching to more compact data representations and selecting when possible a configuration that protects the data against a single disk failure. That process will involve introducing parity disks, merging them and sometimes using them to reconstitute lost data.

Figs. 6 and 7 give a more formal description of our scheme. The first algorithm describes how the array reacts to the loss of a data disk. Two main cases have to be considered, depending on whether the contents of the failed disk D can be found on

Assumptions:

disk *D* is failed data disk

Algorithm:

```
begin
    find disk E  having same contents as disk D
    if found then
        find a disk F whose contents are replicated on another disk G
        if found then
            replace contents (F) by contents (E) XOR contents(F)
        else
            find parity disk Z whose contents are XORed contents of fewest
            data disks
            if found then
                replace contents (Z) by contents (E) XOR contents(Z)
            else
                do nothing
            endif
        endif
    else
        find sufficient set S of disks to reconstitute contents (D)
        if found then
            reconstitute contents (D) on a parity disk X in S
            replace parity disk X
        else
            declare failure
        endif
    endif
end
```

Fig. 6. Replacing a failed data disk

another disk *E*. When this is the case, the array will protect the contents of disk *E* against of a failure of that disk by storing on some disk *F* the XOR of the contents of *E* and the contents of one or more disks. To select this disk *F*, the array will first search for disks whose contents are replicated on some other disk. If it cannot find one, it will then select the parity disk *Z* whose contents are the XORed contents of the fewest data disks. The second case is somewhat more complex. When the contents of the failed disk *D* cannot be found on another disk, the array will attempt to find a sufficient set *S* of disks to reconstitute the contents of the lost disk. If this set exists, it will reconstitute the contents of the lost data disk *D* on a parity disk *X* in S. Once this is done, the array will try to remedy the loss of the parity data on disk *X* by calling the second algorithm.

Our second algorithm describes how the array reacts to the loss of a parity disk *X*. This loss can either be the direct result of a disk failure or a side-effect of the recovery of the contents of a data disk *D*. In either case, the array checks first if it can reconstitute the contents of the failed parity disk *X*. This will be normally possible unless the array has experienced two simultaneous disk failures. If the contents of *X*

Assumptions:
disk X is failed parity disk

Algorithm:

```
begin
    find sufficient set S of disks to reconstitute contents(X)
    if found then
        find a disk F whose contents are mirrored on another disk G
        if found then
            replace contents (F) by contents (E) XOR contents (F)
        else
            find parity disk Z whose contents are XORed contents of fewest
            data disks
            if found then
                replace contents (Z) by contents(Z) XOR contents(X)
            else
                do nothing
            endif
    else
        declare failure
    endif
end
```

Fig. 7. Replacing a failed parity disk.

can be reconstituted, the array will try to XOR them with the contents of a data disk that was replicated elsewhere. If no such data disk exists, the array will XOR the reconstituted contents of X with the contents of the parity disk Z whose contents are the XORed contents of the fewest data disks.

Space considerations prevent us from discussing in detail how the array will handle disk repairs. In essence, it will attempt to return to its original configuration, first by splitting the parity disks whose contents are the XORed contents of the largest number of parity disks then by replacing the remaining parity disks by pairs of data disks. A more interesting issue is how the self-adapting array would react to the loss of a disk involved in a reconfiguration step. Let us return to our previous example and consider what would happen if disk B_2 failed after disk B_1 failed but before the contents of disk A_1 could be completely replaced by the XOR of the contents of disks A_1 and B_2. Assuming that we do this replacement track by track, disk A_1 would be left in a state where it would contain some of its original tracks and some tracks containing the XOR of the corresponding tracks of disks A_1 and B_2. This means that some but not all the contents of disk B_2 would be recoverable and that some but not all contents of disk A_1 would have become vulnerable to a single disk failure.

3 Reliability Analysis

Self-adaptive disk arrays occupy a place between mirrored disk organizations and organizations using erasure coding. As long as most disks are operational, they provide the same read and write access times as static mirrored organizations. In addition, they are more resilient to disk failures. We propose to evaluate this resilience and to compare it with that of mirrored disk organizations.

Estimating the reliability of a storage system means estimating the probability R(t) that the system will operate correctly over the time interval [0, t] given that it operated correctly at time $t = 0$. Computing that function requires solving a system of linear differential equations, a task that becomes quickly unmanageable as the complexity of the system grows. A simpler option is to focus on the mean time to data loss (MTTDL) of the storage system. This is the approach we will take here.

Our system model consists of a disk array with independent failure modes for each disk. When a disk fails, a repair process is immediately initiated for that disk. Should several disks fail, the repair process will be performed in parallel on those disks. We assume that disk failures are independent events exponentially distributed with rate λ, and that repairs are exponentially distributed with rate μ.

The MTTDL for data replicated on two disks is [9]

$$MTTDL = \frac{3\lambda + \mu}{2\lambda^2}$$

and the corresponding failure rate L is

$$L = \frac{2\lambda^2}{3\lambda + \mu}.$$

Consider an array consisting of n disks with all data replicated on exactly two disks. Since each pair of disk fails in an independent fashion, the global failure rate $L(n)$ of the array will be $n/2$ times the failure rate L of a single pair of disks

$$L(n) = \frac{n}{2} L = \frac{n\lambda^2}{3\lambda + \mu}$$

and the global mean time to data loss $MTTDL(n)$ will be

$$MTTDL(n) = \frac{1}{L(n)} = \frac{3\lambda + \mu}{n\lambda^2}.$$

Fig. 8 shows the state transition diagram for a very small self-adaptive array consisting of two pairs of disks with each pair storing two identical replicas of the same data set. Assume that disks A_1 and A_2 contain identical copies of data set A while disks B_1 and B_2 store identical copies of data set B. State <2, 2> represents the normal state of the array when its four disks are all operational. A failure of any of these disks, say disk A_1 would bring the array to state <2, 1>. This state is a less than

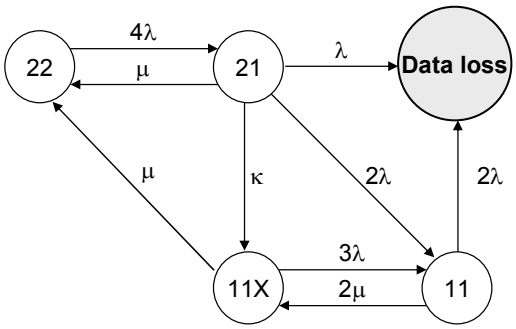

Fig. 8. State transition diagram for a self-adaptive disk array consisting of two pairs of mirrored disks.

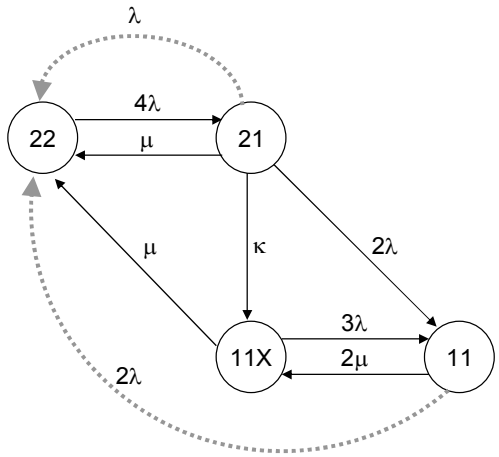

Fig. 9. Modified state transition diagram for the same disk array. The two dotted gray arcs returning the array to state <2, 2> represent data losses.

desirable state because the array has now a single copy of data set A on disk A_2. Hence a failure of that disk would result in a data loss.

To return to a more resilient state, the array will immediately start replacing the contents of either disk B_1 or disk B_2 with the XOR of data sets A and B, thus bringing the system from state <2, 1> to state <1, 1, X>. We assume that the duration of this self-adaptive process will be exponentially distributed with rate κ.

Once this reorganization is completed, the array will have single copies of both data sets A and B on two of the three surviving disks as well as their XOR on the third disk. A failure of either of the two redundant disks present in state <2, 1> or a failure of any of the three disks in state <1, 1, X> would leave the array in state <1, 1>, incapable of tolerating any additional disk failure.

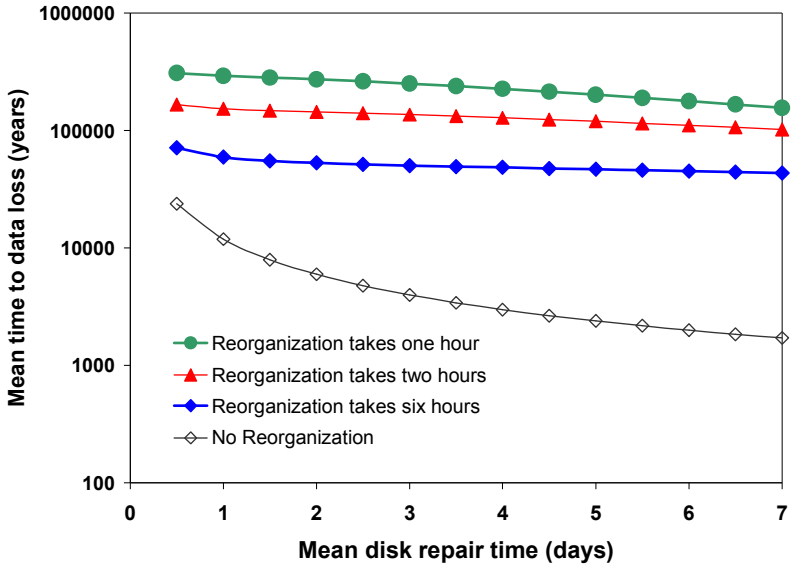

Fig. 10. Mean times to data loss achieved by a self-adaptive disk array consisting of two pairs of mirrored disks.

Recovery transitions correspond to the repair of one of the disks that failed. They would bring the array first from state <1, 1> to state <1, 1, X> and then from state <1, 1, X> to state <2, 2>. A third recovery transition would bring the array from state <2, 1> to state <2, 2>. It corresponds to situations where the failed disk was replaced before the self-adaptive process can be completed.

Since data losses are essentially irrecoverable, the state corresponding to such a loss is an absorbing state. Hence a steady state analysis of the array would provide no insight on its performance.

Let us now consider the state transition diagram displayed in Fig, 9. It has the same states and the same transitions as that of Fig. 8 but for the two transitions leading to a data loss, which are now redirected to state <2, 2>. This diagram represents what would happen if the array went through continuous cycles during which it would first operate correctly then lose its data and get instantly repaired and reloaded with new data [7]. The corresponding system of equations is

$$\begin{aligned} 4\lambda p_{22} &= \mu(p_{21} + p_{11X}) + \lambda p_{21} + 2\lambda p_{11} \\ (3\lambda + \mu + \kappa) p_{21} &= 4\lambda p_{22} \\ (3\lambda + \mu) p_{11X} &= \kappa p_{21}, +2\mu p_{11} \\ (2\lambda + 2\mu) p_{11} &= 3\lambda p_{11X} + 2\lambda p_{21} \end{aligned} \qquad (1)$$

together with the condition that $p_{22} + p_{21} + p_{11X} + p_{11} = 1$, where p_{ij} represents the steady-state probability of the system being in state <*i, j*>. In addition, the rate at which the array will fail before returning to its normal state is

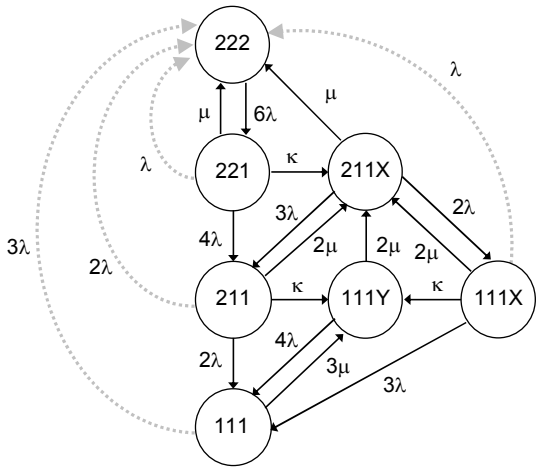

Fig. 11. State transition diagram for a self-adaptive disk array consisting of three pairs of mirrored disks. The four dotted gray arcs returning the array to state <2, 2, 2> represent data losses.

$$L = \lambda p_{21} + 2\lambda p_{11}$$

Solving system (1), we obtain

$$L = \frac{4\lambda^2(9\lambda^2 + 3\kappa\lambda + 3\lambda\mu + \mu^2)}{33\lambda^3 + 13\kappa\lambda^2 + 5\kappa\lambda\mu + 8\lambda\mu^2 + \kappa\mu^2 + \mu^3}.$$

The mean time to data loss of our disk array (MTTDL) is then

$$MTTDL = \frac{1}{L} = \frac{33\lambda^3 + 13\kappa\lambda^2 + 5\kappa\lambda\mu + 8\lambda\mu^2 + \kappa\mu^2 + \mu^3}{4\lambda^2(9\lambda^2 + 3\kappa\lambda + 3\lambda\mu + \mu^2)}.$$

Fig. 10 displays on a logarithmic scale the MTTDLs achieved by the self–adaptive array for selected values of κ and repair times varying between half a day and seven days. We assumed that the disk failure rate λ was one failure every one hundred thousand hours, that is, slightly less than one failure every eleven years and conservative relative to the values quoted by disk manufacturers. Disk repair times are expressed in days and MTTDLs expressed in years.

Fig. 11 displays the state transition diagram for a self-adaptive disk array consisting of three pairs of mirrored disks. Assume that disks A_1 and A_2 contain identical copies of data set A, disks B_1 and B_2 store identical copies of data set B and disks C_1 and C_2 have identical copies of data set C. State <2, 2, 2> represents the normal state of the array when its six disks are all operational. A failure of any of these six disks, say disk A_1, would leave the system in state <2, 2, 1>. This state is a less than desirable state as the array is left with only one copy of data set A. To return to a more resilient state, the array will immediately start replacing the contents of one of the four redundant disks, say, disks B_1, with the XOR of data sets A and B, thus bringing the system from state <2, 2, 1> to state <2, 1, 1, X> with the XOR of data

sets A and B on disk B_1. Failure of a second disk would bring the array to either state <2, 1, 1> or state <1, 1, 1, X>. Both states are less than desirable states, as they leave

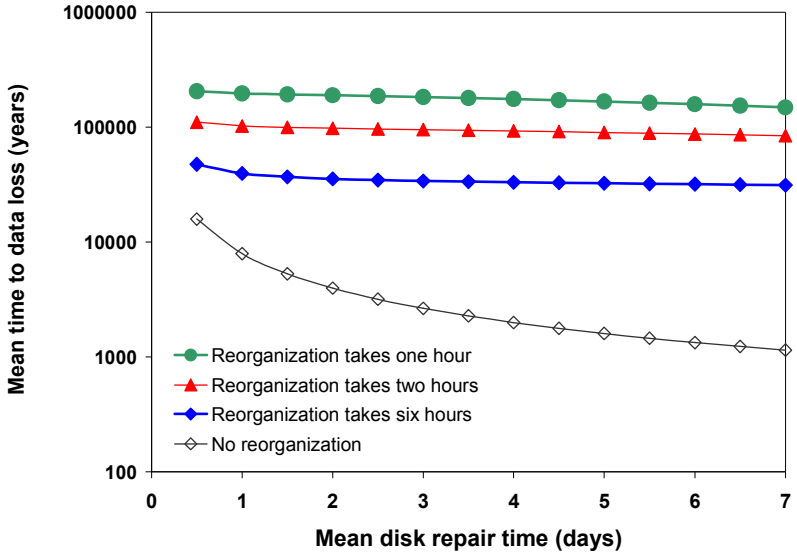

Fig. 12. Mean times to data loss achieved by a self-adaptive disk array consisting of three pairs of mirrored disks.

the array vulnerable to a single disk failure. To return to a more resilient state, the array will bring itself to state <1, 1, 1, Y>, having a single copy of each data set on separate disks and their XOR ($A \oplus B \oplus C$) on disk Y.

Finally, a failure of any of these four disks would bring the array from state <1, 1, 1, Y> to state <1, 1, 1>, having survived three successive data losses and being unable to tolerate a fourth disk failure. Less desirable outcomes would result from the failure of the critical disk in state <2, 2, 1> and <1, 1, 1, X> or from the failure of either of the two critical disks in state <2, 1, 1>. All these failures would result in permanent data loss. As in Fig. 9, all failure transitions that result in a data loss are represented by dotted gray arcs returning to the normal state of the array.

Using the same techniques as in our previous model, we can compute the steady-state probabilities of the system of being in any of its seven possible states and derive from them the rate L at which the array will fail before returning to its normal state

$$L = \lambda p_{221} + 2\lambda p_{211} + \lambda p_{111X} + 3\lambda p_{111}$$

and its MTTDL

$$MTTDL = \frac{1}{L} = \frac{1}{\lambda p_{221} + 2\lambda p_{211} + \lambda p_{111X} + 3\lambda p_{111}}$$

The outcome of these computations is a quotient of polynomials that is too large to be displayed. We refer instead the reader to Fig. 12, which displays on a semi-

logarithmic scale the MTTDLs achieved by the self-adaptive array for selected values of κ and repair times varying between half a day and seven days.

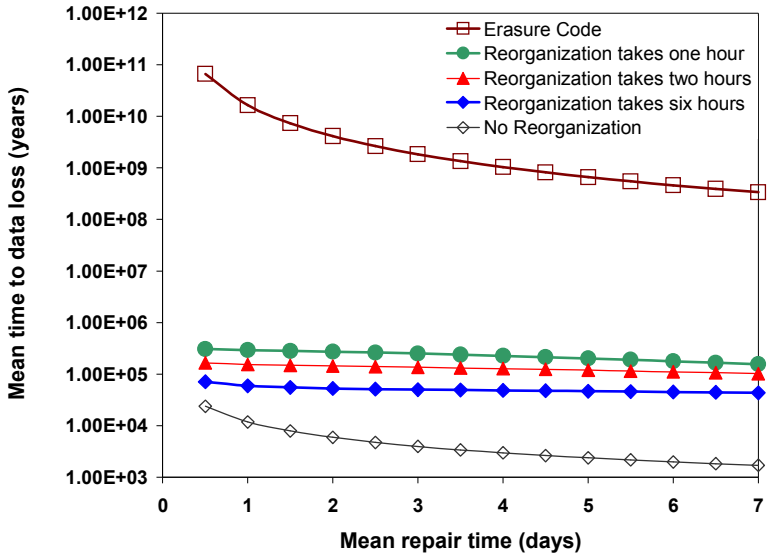

Fig. 13. Mean times to data loss achieved by (a) a self-adaptive disk array consisting of two pairs of mirrored disks and (b) a 2-out-of-4 erasure code.

Let us now see how our technique compares with erasure coding. Rather than storing our data on two pairs of mirrored disks, we could use a 2-out-of-4 erasure code that would tolerate the simultaneous loss of two disks. We can easily derive the MTTDLs achieved by these erasure codes by observing they correspond to the limit case when the reorganization rate κ goes to infinity. Hence, we would have

$$MTTDL_{2-\text{out}-\text{of}-4} = \lim_{\kappa\to\infty} \frac{33\lambda^3 + 13\kappa\lambda^2 + 5\kappa\lambda\mu + 8\lambda\mu^2 + \kappa\mu^2 + \mu^3}{4\lambda^2(9\lambda^2 + 3\kappa\lambda + 3\lambda\mu + \mu^2)} = \frac{13\lambda^2 + 5\lambda\mu + \mu^2}{12\lambda^3}.$$

Fig. 13 compares the MTTDLs achieved by a 2-out-of-4 erasure code, a pair of self-adaptive mirrored disks and a pair of conventional mirrored drives. As we can see, the 2-of-out-4 erasure code achieves much higher MTTDLs than a self-adaptive disk array with four disks. These excellent results need however to be qualified in two important ways. First, all our analyses have assumed that disk failures were the only causes of data losses. We did not consider other types of system malfunctions such as media errors, human errors, power failures, fires, floods and other acts of God. As we consider solutions minimizing the impact of disk failures, these other malfunctions will quickly become the main cause of data losses. Second, 2-out-of-4 erasure codes will result in much costlier write accesses that mirroring or even conventional RAID level 5.

We can make four main observations from our results. First, our self-adaptive array provides much better MTTDLs than a static array that makes no attempt at reconfiguring itself after disk failures. The improvements vary between a minimum of 200 percent and a maximum of almost 13,000 percent depending on the disk repair rate and the array reorganization rate, with the best results achieved for a combination of a slow disk repair rate and a fast array reconfiguration rate. This is a very significant result as we have only considered arrays consisting of two and three pairs of mirrored drives. Larger self-adaptive disk arrays should perform even better as they can perform many more corrective actions to protect their data after successive disk failures.

Second, these benefits remain evident even when the reconfiguration process takes six hours. Since the reconfiguration process normally consists of reading the whole contents of a disk and XORing these contents with the contents of a second disk, this is clearly an upper bound. In reality we expect most reconfiguration tasks to take between one and two hours depending on the disk bandwidths and capacities.

Third, the MTTDLs achieved by our self-adaptive organization remain nearly constant over a wide range of disk repair times. This is a significant advantage because fast repair times require maintaining a local pool of spare disks and having maintenance personnel on call 24 hours a day. Since our self-adaptive organization tolerates repair times of up to one week, if not more, it will be cheaper and easier to maintain than a static mirrored disk organization with the same number of disks.

Finally, erasure codes ought to be seriously considered whenever we want to provide the highest level of protection to data that are very unlikely to be ever updated.

4 Previous Work

The idea of creating additional copies of important data in order to increase their chances of survival is likely to be as old as the use of symbolic data representations by mankind and could well have preceded the discovery of writing. Erasure coding appeared first in RAID organizations as $(n-1)$-out-of-n codes [3, 4, 6, 8, 9]. RAID level 6 organizations use $(n-2)$-out-of-n codes to protect data against double disk failures [1].

The HP AutoRAID [11] automatically and transparently manages migration of data blocks between a replicated storage class and a RAID level 5 storage class as access patterns change. This system differs from our proposal in several important aspects. First, its objective is different from ours. AutoRAID attempts to save disk space without compromising system performance by storing data that are frequently accessed in a replicated organization while relegating inactive data to a RAID level 5 organization. As a result, data migrations between the two organizations are normally caused by changes in data access patterns rather than by disk failures. Self-adaptive disk arrays only reconfigure themselves in response to disk failures and repairs. Second, AutoRAID actually migrates data between its two storage classes while self-adaptive disk arrays keeps most data sets in place. Finally, the sizes of the transfer units are quite different. A self-adaptive disk array manages its resources at the disk level. The transfer units managed by AutoRAID are *physical extent groups* (PEGs)

consisting of at least three *physical extents* (PEXes) whose typical size is a megabyte. Consequently, AutoRAID requires a complex addressing structure to locate these PEGs while a self-adaptive array must only keep track of what happened to the contents of its original disks. Assuming that we have n data sets replicated on $2n$ disks, the actual locations of data sets and their parities can be stored in $2n^2$ bits. In addition, this information is fairly static as it is only updated after a disk failure or a disk repair.

Another form of adaptation to disk failure is *sparing*. Adding a spare disk to a disk array provides the replacement disk for the first failure. Distributed sparing [10] gains performance benefits in the initial state and degrades to normal performance after the first disk failure.

5 Conclusions

We have presented a disk array organization that adapts itself to successive disk failures. When all disks are operational, all data are replicated on two disks. Whenever a disk fails, the array will immediately reorganize itself and adopt a new configuration that will protect all data against any single disk failure until the failed disk gets replaced and the array can revert to its original condition. Hence data will remain protected against the successive failures of up to one half of the original number of disks, provided that no critical disk failure happens while the array is reorganizing itself. As a result, our scheme achieves the same access times as a mirrored disk organization under normal operational conditions while having a much lower likelihood of loosing data under abnormal conditions. Furthermore, the MTTDLs achieved by our self-adaptive organization remain nearly constant over a wide range of disk repair times.

More work is still needed to investigate larger disk arrays. As the number of possible reconfiguration steps increases with the size of the array, simulation will become an increasingly attractive alternative to Markov models. We also plan to investigate self-adaptive strategies for disk arrays where some data are more critical than other and thus deserve a higher level of protection. This is the case for archival storage systems implementing chunking to reduce their storage requirements. Chunk-based compression, or chunking, partitions files into variable-size chunks in order to identify identical contents that are shared by several files [5]. Chunking can significantly reduce the storage requirements of archival file systems. Unfortunately, it also makes the archive more vulnerable to the loss of chunks that are shared by many files. As a result, these chunks require a higher level or protection than chunks that are only present in a single file [2].

References

1. Burkhard, W. and J. Menon: Disk Array Storage System Reliability. Proc. 23rd International Symposium on Fault-Tolerant Computing (FTCS-23), pp. 432-441, 1993.

2. Bhagwat, D., K. Pollack, D. D. E. Long, E. L. Miller, T. J. Schwarz and J.-F. Pâris: Providing High Reliability in a Minimum Redundancy Archival Storage System. Proc. 14th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, to appear, Sep. 2006.
3. Chen, P. M., E. K. Lee, G. A. Gibson, R. Katz, and D. Patterson: RAID, High-Performance, Reliable Secondary Storage. ACM Computing Surveys, Vol. 26, No. 2, pp. 145–185, 1994.
4. Ganger, G., B. Worthington, R. Hou, Y. Patt: Disk arrays: High-performance, high-reliability storage subsystems. IEEE Computer vol. 27(3), p. 30–36. 1994.
5. Muthitacharoen, A., B. Chen, and D. Mazieres: A Low-Bandwidth Network File System. Proc. 18th Symposium on Operating Systems Principles, pp. 174-187, 2001.
6. Patterson, D. A., G. A. Gibson, and R. H. Katz: A Case For Redundant Arrays Of Inexpensive Disks (RAID). Proc. SIGMOD 1988 International Conference on Data Management, pp. 109–116, June 1988.
7. Pâris, J.-F., T. J. E. Schwarz and D. D. E. Long: Evaluating the Reliability of Storage Systems. Technical Report UH-CS-06-08, Department of Computer Science, University of Houston, June 2006.
8. Schwarz, T. J. E., and W. A. Burkhard: RAID Organization and Performance. Proc. 12th International Conference on Distributed Computing Systems, pp. 318–325, June 1992.
9. Schulze, M., G. Gibson, R. Katz and D. Patterson: How Reliable is a RAID? Proc. Spring COMPCON '89 Conference, pp. 118–123, March 1989.
10. Thomasian, A. and J. Menon: RAID 5 Performance with Distributed Sparing. IEEE Transactions on Parallel and Distributed Systems, Vol. 8(6), pp. 640–657, June 1997.
11. J. Wilkes, R. Golding, C. Stealin, C. and T. Sullivan: The HP AutoRAID hierarchical storage system. ACM Transactions on Computer Systems, Vol. 14(1), pp. 1–29, Feb. 1996.