Self-Adaptive Two-Dimensional RAID Arrays

Jehan-François Pâris[1]
Dept. of Computer Science
University of Houston
Houston, TX 77204-3010
paris@cs.uh.edu

Thomas J. E. Schwarz
Dept. of Computer Engineering
Santa Clara University
Santa Clara, CA 95053
tjschwarz@scu.edu

Darrell D. E. Long[1]
Dept. of Computer Science
University of California
Santa Cruz, CA 95064
darrell@cs.ucsc.edu

Abstract

We propose increasing the survivability of data stored in two-dimensional RAID arrays by letting these arrays reorganize themselves whenever they detect a disk failure. This reorganization will rebalance as much as possible the redundancy level of all stored data, thus reducing the potential impact of additional disk failures. It remains in effect until the failed disk gets repaired. We show how our technique can be applied to two-dimensional RAID arrays consisting of n^2 data disks and 2n parity disks and show how it can increase the mean time to data loss of the array by at least 200 percent as long as the reorganization process takes less than half the time it takes to replace a failed disk.

1 Introduction

Archival storage systems are systems designed for long-term stable storage of information. Their importance is growing as more organizations maintain larger amounts of their archival data online. This trend is due to many factors; among these are lower costs of online storage, regulatory requirements and the increasing rate at which digital data are produced.

Archival storage systems differ from conventional storage systems in two important ways. First, the data they contain often remain immutable once they are stored. As a result, write rates are a much less important issue than in other storage systems. Second, these data must remain available over time periods that can span decades. Achieving this longevity requires a special emphasis on data survivability.

The best way to increase the survivability of data is through the use of redundancy. Two well-known examples of this approach are mirroring and *m*-out-of-*n* codes. Mirroring maintains multiple replicas of the stored data while *m*-out-of-*n* codes store data on *n* distinct disks along with enough redundant information to allow access to the data in the event that $n - m$ of these disks fail. The best-known organizations using these codes are RAID level 5, which uses an $(n - 1)$-out-of-*n* code, and RAID level 6, which uses an $(n - 2)$-out-of-*n* code.

While both mirroring and *m*-out-of-*n* codes can be used to obtain any desired level of data survivability, they achieve that objective by either maintaining extra copies of the stored data or implementing more complex omission-correction schemes. Both techniques will have a significant impact on data storage and update costs.

We propose to control these costs by having storage organizations adapt to failures and reorganize themselves in a way that minimizes the risk of a data loss. As a result, these organizations will achieve higher levels of data survivability without increasing the redundancy levels of the stored data. Consider data stored on a disk array using some arbitrary redundant organization and assume that one of the disks has failed. While this failure will not result in a data loss, it is likely to have an unequal impact on the protection level of the data: some data will be left less protected—or even totally unprotected—while other data will not be affected by the failure. This is clearly an undesirable situation as it increases the risk of a data loss. We propose to let the disk array adjust to the failure by readjusting the protection levels of its data in a way that ensures that no data are left significantly less protected than the others. This new organization will then remain in effect until the failed disk gets replaced and the array returned to its original condition. The whole process will be done automatically and remain transparent to the user.

Our proposal has the main advantage of increasing the survivability of data stored on almost any redundant organization without requiring any additional hardware. As we will see, it will also reduce the impact of disk repair times on the survivability of the archived data.

2 Previous Work

The idea of creating additional copies of critical data in order to increase their chances of survival is probably as old as the use of symbolic data representations by mankind. Erasure coding for disk storage first appeared in RAID organizations as $(n - 1)$-out-of-*n* codes [2, 4–6]. RAID level 6

[1] Supported in part by the National Science Foundation under award CCR-0204358.

organizations use $(n-2)$-out-of-n codes to protect data against double disk failures [1, 9].

Much less work has been dedicated to self-organizing fault-tolerant disk arrays. The HP AutoRAID [8] automatically and transparently manages migration of data blocks between a mirrored storage class and a RAID level 5 storage class as access patterns change. Its main objective is 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, it reacts to changes in data access patterns rather than to disk failures.

Sparing is more relevant to our proposal as it provides a form of adaptation to disk failures. Adding a spare disk to a disk array provides the replacement disk for the first failure. Distributed sparing [7] gains performance benefits in the initial state and degrades to normal performance after the first disk failure.

Pâris et al. [3] have recently presented a mirrored 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 starts using$(n-1)$ out-of-n codes in such a way that no data are left unprotected. We extend here a similar approach to two-dimensional RAID arrays.

3 Self-Adaptive Disk Arrays

While our technique is general and applies to most redundant disk arrays, its application will depend on the actual array organization. Hence we will present it using a specific disk array organization.

Consider the two dimensional RAID array of Fig. 1. It consists of nine data disks and six parity disks. Parity disks P_1, P_2 and P_3 contain the exclusive or (XOR) of the contents of the data disks in their respective rows while parity disks Q_1, Q_2 and Q_3 contain the XOR of the contents of the data disks in their respective columns. This organization offers the main advantage of ensuring that the data will survive the failure of an arbitrary pair of disks and most failures of three disks. As seen on Fig. 2, the sole triple failures that result in a data loss are the failure of one arbitrary data disk, the parity disk in the same row and the parity disk in the same column.

We propose to eliminate this vulnerability by having the disk array reorganize itself in a transparent fashion as soon as it has detected the failure of a single disk. We will first consider how the disk array will handle the loss of a parity disk and then how it will handle the loss of a data disk.

A Handling the loss of a parity disk

Consider first the loss of a parity disk, say parity disk Q_2. As Fig. 3 shows, this failure leaves the array vulnerable to a simultaneous failure of disks D_{12} and P_1 or disks D_{22} and P_2 or disks D_{32} and P_3.

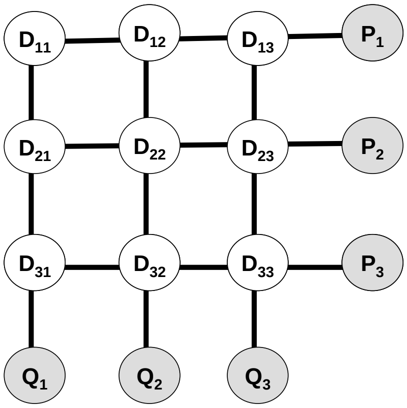


Fig. 1 – A two dimensional RAID array with nine data disks and six parity disks.

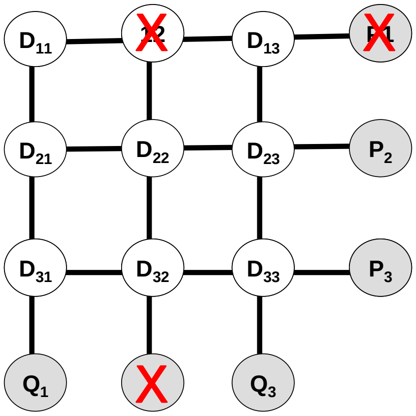


Fig. 2 – The same array experiencing the simultaneous failures of one arbitrary data disk, the parity disk in the same row and the parity disk in the same column.

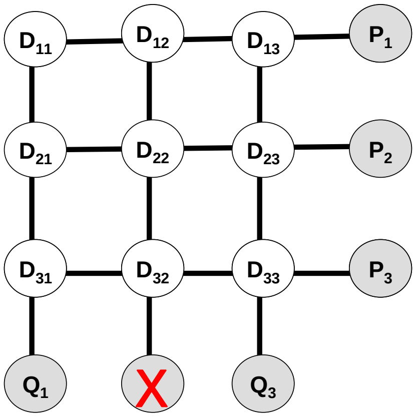


Fig. 3 – How the array is affected by the failure of an arbitrary parity disk.

We can eliminate this vulnerability by selecting a new array organization such that:

1. Each data disk will belong to two distinct parity stripes.

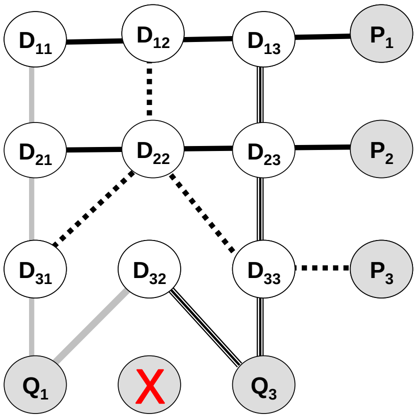


Fig. 4 – A new array organization protecting the data against two simultaneous disk failures after the failure of a parity disk.

2. Two distinct parity stripes will have at most one common data disk.

The first condition guarantees that the array will always have two independent ways to reconstitute the contents of any failed data disk. Without the second condition, two or more data disks could share their two parity stripes. As a result, the array would be unable to recover from the simultaneous failure of these two disks.

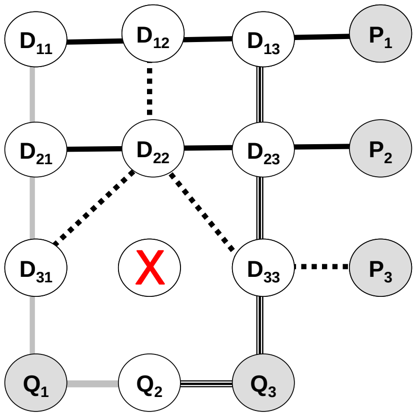
Fig. 4 displays an array organization satisfying these two conditions. It groups the nine data disks into five parity stripes such that:

1. Disks D_{11}, D_{12} and D_{13} keep their parities stored on disk P_1
2. Disks D_{21}, D_{22} and D_{23} keep their parities stored on disk P_2.
3. Disks D_{31}, D_{12}, D_{22}, and D_{33} have their parities stored on disk P_3.
4. Disks D_{11}, D_{21}, D_{31}, and D_{32} have their parities stored on disk Q_1.
5. Disks D_{13}, D_{23}, D_{33}, and D_{32} have their parities stored on disk Q_2.

Table 1 itemizes the actions to be taken by the fourteen operational disks to achieve the new organization. As we can see, six of the nine data disks and two of the five remaining parity disks do not have to take any action. The two busiest disks are data disk D_{32}, which has to send its contents to its old parity disk $P3$ and its two new parity disk Q_1 and $Q_{2,}$ and parity disk $P3$, which has to XOR to its contents the contents of data disks D_{12}, D_{22} and D_{32}. This imbalance is a significant limitation of our scheme, as it will slow down the array reorganization process, thus delaying its benefits.

Table 1 – Actions to be taken by the fourteen operational disks after the failure of parity disk Q_2.

Disk	*Action to be taken*
D_{11}	Do nothing
D_{12}	Send contents to P_3
D_{13}	Do nothing
D_{21}	Do nothing
D_{22}	Send contents to P_3
D_{23}	Do nothing
D_{31}	Do nothing
D_{32}	Send contents to P_3, Q_1 and Q_3
D_{33}	Do nothing
P_1	Do nothing
P_2	Do nothing
P_3	XOR to its contents the contents of D_{12}, D_{22} and D_{32}
Q_1	XOR to its contents the contents of D_{32}
Q_3	XOR to its contents the contents of D_{32}



(Q_2 replaces D_{32})

Fig. 5 – A new array organization protecting the data against two simultaneous disk failures after the failure of a data disk.

B Handling the loss of a data disk

Let us now discuss how the array should react to the failure of one of its data disks, say disk D_{32}.

The first corrective step that the array will take will be to reconstitute the contents of the failed disk on one of its parity disks, say, disk Q_2. Once this process is completed, the reorganization process will proceed in the same fashion as in the previous case.

Figure 5 displays the final outcome of the reorganization process. Each data disk—including disk Q_2—now belongs to two distinct parity stripes and two distinct parity disks have at most one common disk.

Table 2 itemizes the actions to be taken by the fourteen operational disks to achieve the new organization. Observe that the reorganization process will now require two steps since the array must first compute the new contents of disk Q_2 before updating disks Q_1 and Q_3.

C Discussion

Our technique can be trivially extended to all two-dimensional RAID arrays consisting of n^2 data disks and $2n$ parity disks with $n \geq 3$. The array will always handle the loss of a parity disk by assigning a

Table 2 – Actions to be taken by the fourteen operational disks after the failure of data disk D_{32}.

Disk	*Action to be taken*
D_{11}	Do nothing
D_{12}	Send contents to P_3 and Q_2
D_{13}	Do nothing
D_{21}	Do nothing
D_{22}	Send contents to P_3 and Q_2
D_{23}	Do nothing
D_{31}	Do nothing
D_{33}	Do nothing
P_1	Do nothing
P_2	Do nothing
P_3	XOR to its contents the contents of D_{12}, D_{22} and D_{32}
Q_1	XOR to its contents new contents of Q_2
Q_2	XOR to its contents the contents of D_{12} and D_{22} Send new contents to Q_1 and Q_3
Q_3	XOR to its contents new contents of Q_2

new parity stripe to each of the n disks belonging to same parity stripe as the disk that failed. Hence the reorganization process will only involve n out of the n^2 data disks and n out of the remaining $2n – 1$ parity disks. Handling the failure of a data disk will involve one less data disk and one extra parity disk.

4 Reliability Analysis

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. We will first consider a disk array consisting of 9 data disks and 6 parity disks and then a larger array with 16 data disks and 8 parity drives.

Building an accurate state-transition diagram for either two-dimensional disk array is a daunting task as we have to distinguish between failures of data disks and failures of parity disks as well as between failures of disks located on the same or on different parity stripes. Instead, we present here a simplified model based on the following approximations.

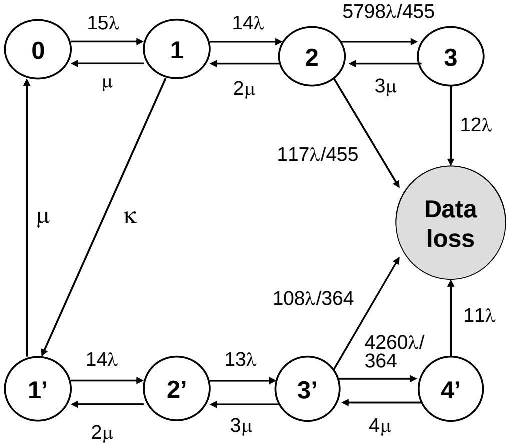


Fig. 6 – Simplified state transition probability diagram for a two-dimensional array consisting of 9 data disks and 6 parity disks.

1. Whenever the disk repair rate μ is much higher than the disk failure rate λ, each individual disk will be operational most of the time. Hence the probability that the array has four failed disks will be almost negligible when we compare it to the probability that the array has three failed disks. We can thus obtain a good upper bound of the array failure rate by assuming that the array fails whenever it has three failed disks in any of the nine critical configurations discussed in section 3 or at least four failed disks regardless of their configuration. In other words, we will ignore the fact that the array can survive some, but not all, simultaneous failures of four or more disks.
2. Since disk failures are independent events exponentially distributed with rate λ, the rate at which an array that has already two failed disks will experience a third disk failure is $(16 – 2)\lambda = 14\lambda$. Observing there are 455 possible configurations with 3 failed disks out of 15 but only 9 of them result in a data loss, we will assume that the rate at which an array that has already two failed disks will not fail will be $(455 – 9)\times 13\lambda/455 = 5798\lambda/455$.

Fig. 6 displays the simplified state transition probability diagram for a two-dimensional array consisting of 9 data disks and 6 parity disks. State <0> represents the normal state of the array when its fifteen disks are all operational. A failure of any of these disks would bring the array to state <1>. A failure of a second disk would bring the array into state <2>. A failure of a third disk could either result in a data loss or bring the array to state <3>. As we stated earlier, we assume that any third failure occurring while the array already has three failed disks will result in a data loss.

Repair transitions bring back the array from state <3> to state <2> then from state <2> to state <1> and, finally, from state <1> to state <0>. Their rates are equal to the number of failed disks times the disk repair rate μ.

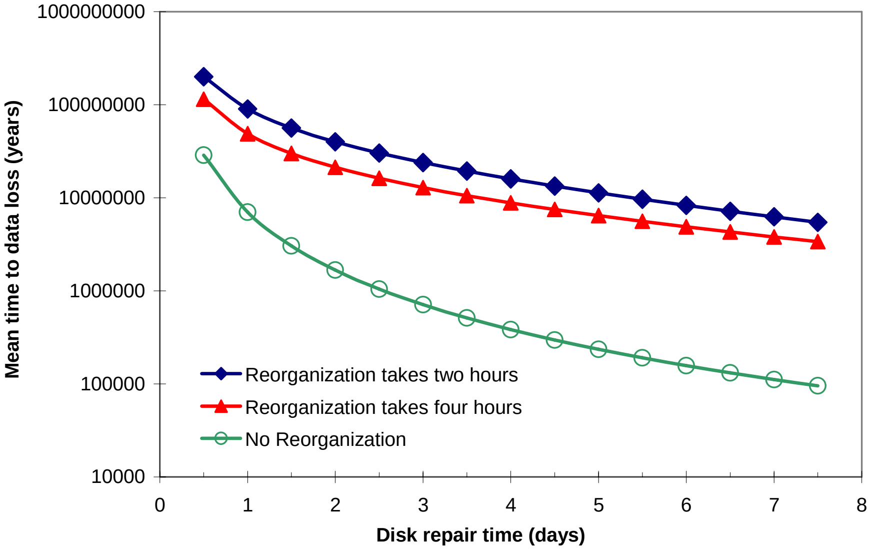


Fig. 7 – Mean times to data loss achieved by a two dimensional disk array consisting of nine data disks and six parity disks.

When the array reaches state <1>, it will start a reorganization process that will bring it into state <1'>. State <1> is a more resilient state than state <1'> as the reorganized array tolerates two arbitrary additional failures. We will assume that this duration of the reorganization process is exponentially distributed with rate *κ*. A failure of a second disk while the array is in state <1'> would bring the array into state <2'> and a failure of a third disk would bring the array to state <3'>. When the array is in state <3'>, any failure that would occasion the simultaneous failure of one of the nine data disks and its two parity disks would result in a data loss. Observing there are 364 possible configurations with 3 failed disks out of 14 but only 9 of them result in a data loss, we see that the transition rate between state <3'> and state <4'> is given by $(364 - 9)\times 12\lambda/364 = 4260\lambda/364$.

The Kolmogorov system of differential equations describing the behavior of the array is

$$\frac{dp_0(t)}{dt} = -15\lambda p_0(t) + \mu p_1(t) + \mu p_{1'}(t)$$

$$\frac{dp_1(t)}{dt} = -(14\lambda + \kappa + \mu)p_1(t) + 15\lambda p_0(t) + 2\mu p_2(t))$$

$$\frac{dp_{1'}(t)}{dt} = -(14\lambda + \kappa + \mu)p_1(t) + 15\lambda p_0(t) + 2\mu p_2(t))$$

$$\frac{dp_2(t)}{dt} = -(13\lambda + 2\mu)p_2(t) + 14\lambda p_1(t) + 3\mu p_3(t)$$

$$\frac{dp_{2'}(t)}{dt} = -(13\lambda + 2\mu)p_{2'}(t) + 14\lambda p_{1'}(t) + 3\mu p_{3'}(t)$$

$$\frac{dp_3(t)}{dt} = -(12\lambda + 3\mu)p_3(t) + \frac{5798}{455}\lambda p_2(t)$$

$$\frac{dp_{3'}(t)}{dt} = -(12\lambda + 3\mu)p_{3'}(t) + 13\lambda p_{2'}(t) + 4\mu p_{4'}(t)$$

$$\frac{dp_{4'}(t)}{dt} = -(11\lambda + 4\mu)p_{4'}(t) + \frac{4260}{364}\lambda p_{3'}(t)$$

where $p_i(t)$ is the probability that the system is in state <*i*> with the initial conditions $p_0(0) = 1$ and $p_i(0) = 0$ for $i \neq 0$.

The Laplace transforms of these equations are

$$sp_0^*(s) - 1 = -15\lambda p_0^*(s) + \mu p_1^*(s) + \mu p_{1'}^*(s)$$

$$sp_1^*(s) = -(14\lambda + \kappa + \mu)p_1^*(s) + 15\lambda p_0^*(s) + 2\mu p_2^*(s)$$

$$sp_{1'}^*(s) = -(14\lambda + \mu)p_{1'}^*(s) + \kappa p_1^*(s) + 2\mu p_{2'}^*(s)$$

$$sp_2^*(s) = -(13\lambda + 2\mu)p_2^*(s) + 14\lambda p_1^*(s) + 3\mu p_3^*(s)$$

$$sp_{2'}^*(s) = -(13\lambda + 2\mu)p_{2'}^*(s) + 14\lambda p_{1'}^*(s) + 3\mu p_{3'}^*(s)$$

$$sp_3^*(s) = -(12\lambda + 3\mu)p_3^*(s) + \frac{5798}{455}\lambda p_2^*(s)$$

$$sp_{3'}^*(s) = -(12\lambda + 3\mu)p_{3'}^*(s) + 13\lambda p_{2'}^*(s) + 4\mu p_{4'}^*(s)$$

$$p_{4'}^*(s) = -(11\lambda + 4\mu)p_{4'}^*(s) + \frac{4260}{364}\lambda p_{3'}^*(s)$$

Observing that the mean time to data loss (MTTDL) of the array is given by

$$MTTDL = \sum_i p_i^*(0),$$

we solve the system of Laplace transforms for $s = 0$ and use this result to compute the MTTDL.. The expression we obtain is quotient of two polynomials that are too large to be displayed.

Fig. 7 displays on a logarithmic scale the MTTDLs achieved by a two-dimensional array with 9 data disks and 6 parity disks for selected values of the reorganization rate κ and repair times that vary 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. Disk repair times are expressed in days and MTTDLs expressed in years. As we can see, our technique increases the MTTDL of the array by at least 200 percent as long as the reorganization process takes less than half the time it takes to replace a failed disk, that is, $\kappa > 2\mu$. In addition, it also reduces the impact of disk repair times on the MTTDL of the array. This is an important advantage of our technique as short repair times require both maintaining a local pool of spare disks and having maintenance personnel on call 24 hours a day.

Let us now consider the case of a larger two-dimensional array consisting of 16 data disks and 8 parity disks. As Fig. 8 shows, the simplified state transition probability diagram for the new array is almost identical to the one for the array .consisting of 9 data disks and 6 parity disks, the sole difference being the weights of the failure transitions between the states. Some of these changes are self-evident: since the new array has 24 disks, the transition rate between state <0> and state <1> is now 24λ. Let us focus instead on the transitions leaving states <2> and states <3'>.

Recall that state <2> is a state where the array has already lost 2 of its 24 disks and has not yet been reorganized. Hence it remains vulnerable to the loss of a third disk. Observing there are 2,204 possible configurations with 3 failed disks out of 24 but only 16 of them result in a data loss, we will assume that the rate at which the array will fail will be given by $16{\times}22\lambda/2204$ or $352\lambda/2204$. Conversely, the transition rate between state <2> and state <3> will be $(2204 - 16){\times}22\lambda/2204$ or $44176\lambda/2204$.

State <3'> is a state where the reconfigured array has lost two additional disks and has become vulnerable to the failure of a fourth disk. Observing there are 364 possible configurations with 3 failed disks out of 23 but only 16 of them result in a data loss, we see that the hat the rate at which an array will fail will be equal to $16{\times}21\lambda/1771$ or $336\lambda/1771$. As a result, the transition rate between state <3'> and <4'> will be equal to $(1771 - 16){\times}21\lambda/1771$ or $26855\lambda/1771$.

Using the same techniques as in the previous system, we obtain the MTTDL of our disk array by computing the Laplace transforms of the system of differential equations describing the behavior of the array and solving the system of Laplace transforms for $s = 0$.

Fig. 9 displays on a logarithmic scale the MTTDLs achieved by a two-dimensional array with 16 data disks and 8 parity disks for selected values of the reorganization rate κ and repair times varying between half a day and a week. As before, we assumed that the disk failure rate λ was equal to one failure every one hundred thousand hours one failure every one hundred thousand hours. Here too, we can see that our technique increases the MTTDL of the array by at least 200 percent as long as $\kappa > 2\mu$ and reduces the impact of disk repair times on the MTTDL of the array.

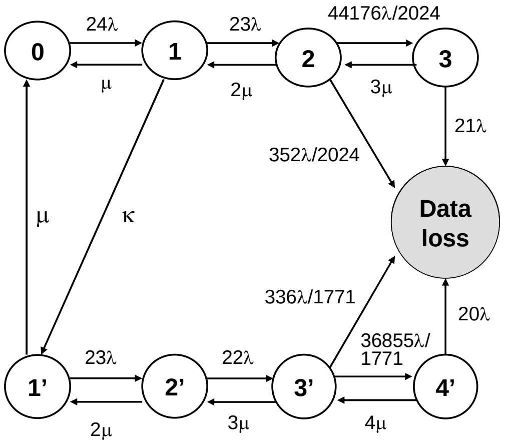


Fig. 8 – Simplified state transition probability diagram for a two-dimensional array consisting of 16 data disks and 8 parity disks.

4 An Alternate Organization

Another way of organizing n^2 data disks and $2n$ parity disks is to partition them into n RAID level 6 stripes each consisting of n data disks and two parity disks. (We may prefer to call now these two disks *check disks.*) This organization is displayed on Fig. 10. It would protect data against the failure of up to two disks in any of its n stripes. We propose to evaluate its MTTDL and compare it to those obtained by our two-dimensional array.

Fig. 11 displays the state transition probability diagram for a single RAID level 6 stripe consisting of three data disks and two check disks. State <0> represents the normal state of the stripe when its five disks are all operational. A failure of any of these disks would then bring the stripe to state<1>. A failure of a second disk would bring the stripe into state <2>. A failure of a third disk would result in a data loss. Repair transitions bring back the strip from state <2> to state <1> and then from state <1> to state <0>.

The system of differential equations describing the behavior of each RAID level 6 stripe is

$$\frac{dp_0(t)}{dt} = -5\lambda p_0(t) + \mu p_1(t)$$

$$\frac{dp_1(t)}{dt} = -(4\lambda + \mu)p_1(t) + 5\lambda p_0(t) + 2\mu p_2(t)$$

$$\frac{dp_2(t)}{dt} = -(3\lambda + 2\mu)p_2(t) + 4\lambda p_1(t)$$

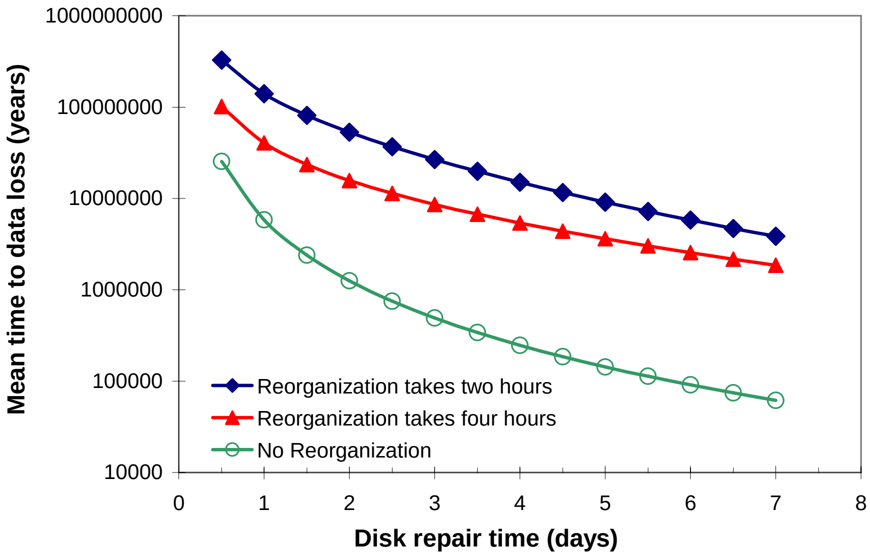


Fig. 9 – Mean times to data loss achieved by a two dimensional disk array consisting of 16 data disks and 8 parity disks.

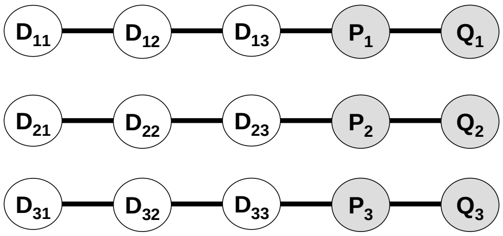


Fig. 10 – An alternative organization with nine data disks and six check disks.

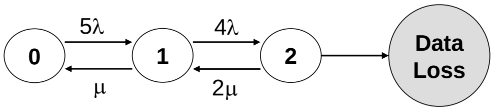


Fig. 11 – State transition probability diagram for a stripe consisting of three data disks and two check disks.

Applying the same techniques as in the previous section, we obtain the MTTDL of each stripe.

$$MTTDL_s = \sum_i p_i^*(0) = \frac{47\lambda^2 + 13\lambda\mu + 2\mu^2}{60\lambda^3}$$

Since our array configuration consists of three stripes, the MTTDL of the whole array is

$$MTTDL_a = \frac{MTTDL_s}{3} = \frac{47\lambda^2 + 13\lambda\mu + 2\mu^2}{180\lambda^3}.$$

Fig. 12 displays on a logarithmic scale the MTTDLs achieved by the new array configuration and compares them with the MTTDLs achieved by a two-dimensional array with the same number of data disks and parity disks. As in Fig. 7, the disk failure rate λ is assumed to be equal to one failure every one hundred thousand hours and the disk repair times vary between half a day and seven days.

As we can see, the new organization achieves MTTDLs that are significantly lower than these achieved by the two-dimensional array even when we assume that no reorganization can take place. While this gap narrows somewhat when the mean repair time τ increases, the MTTDL achieved by the new organization never exceed 41 percent of the MTTDL achieved by a static two-dimensional array with the same number of data disks and parity disks.

We can explain this discrepancy by considering the number of triple failures that will ause either disk organization to fail. As we saw earlier, 9 triple failures out of 465 result in a data loss for a two-dimensional array consisting of nine data drives and six parity drives. In the case of a RAID level 6 organization, any failure of three disks in any of the three stripes would result in a data loss. Since each stripe consists of 5 disks, there are exactly 10 distinct triple failures to consider in each of the 3 stripes. Hence the total number of triple failures that will result in a data loss is 30 out of 465, that is, slightly more that three times the corresponding number of triple failures for the two-dimensional disk organization.

We should also observe that this performance gap will increase with the size of the array. Consider, for instance, a two-dimensional array consisting of 25 data drives and 10 parity drives. The only triple failures that would result in a data loss for our two dimensional array involve the simultaneous failures of an arbitrary data disk and

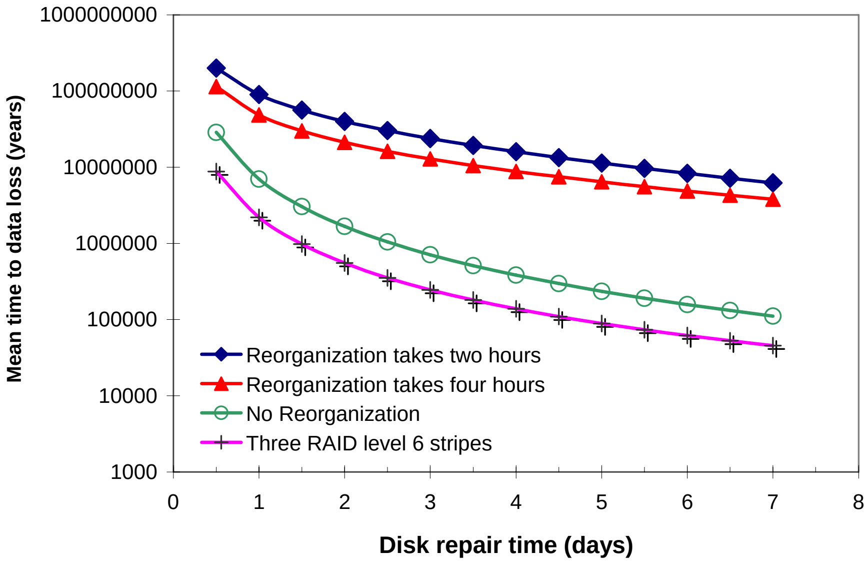


Fig. 12 – Mean times to data loss achieved by various disk arrays consisting of nine data disks and six parity disks.

its two parity disks. Since our array has 25 data disks, this corresponds to 25 out of the 6545 possible triple failure. Consider now an alternate organization consisting of 5 stripes each consisting of 5 data disks and 2 check disks and observe that a organization consisting of 5 stripes each consisting of 5 data disks and 2 check disks and observe that a failure of three disks in any of the five stripes would result in a data. Since each stripe consists of 7 disks, there are exactly 35 distinct triple failures to consider in each stripe. Hence the total number of triple failures that will result in a data loss is 175 out of 6545, that is, seven times the corresponding number of triple failures for the two-dimensional disk organization.

5 Conclusion

We have presented a technique for improving the survivability of data stored on archival storage systems by letting these systems reorganize themselves whenever they detect of a disk failure and until the failed disk gets replaced.

This reorganization will rebalance as much as possible the redundancy level of all stored data, thus reducing the potential impact of additional disk failures. It will remain in effect until the failed disk gets repaired. We show how our technique can be applied to two-dimensional RAID arrays consisting of n^2 data disks and $2n$ parity disks and discuss its impact on the mean time to data loss of arrays with 15 and 24 disks. We found out that the reorganization process was especially beneficial when the repair time for individual disks exceeded one to two days and concluded that a self-adaptive array would tolerate much longer disk repair times than a static array making no attempt to reorganize itself in the presence of a disk failure.

In addition, we found out that this two-dimensional disk organization achieved much better MTTDLs than a set of RAID level 6 stripes, each having *n* data disks and two check disks.

References

[1] W. A. Burkhard and J. Menon. Disk array storage system reliability. In *Proc. 23rd Int. Symp. on Fault-Tolerant Computing*, pp. 432-441, 1993.

[2] P. M. Chen, E. K. Lee, G. A. Gibson, R. Katz and D. A. Patterson. RAID, High-performance, reliable secondary storage, *ACM Computing Surveys* 26(2):145–185., 1994,.

[3] J.-F. Pâris, T. J. E. Schwarz and D. D. E. Long. Self-adaptive disk arrays. In *Proc. 8th Int. Symp. on Stabilization, Safety, and Security of Distributed Systems*, pp/ 469–483, Nov. 2006.

[4] D. A. Patterson, G. A. Gibson and R. Katz. A case for redundant arrays of inexpensive disks (RAID). In *Proc. SIGMOD 1988 Int. Conf, on Data Management*, pp. 109–116, June 1988.

[5] T. J. E. Schwarz and W. A. Burkhard. RAID organization and performance. In *Proc. 12th Int. Conf. on Distributed Computing Systems*,, pp. 318–325 June 1992.

[6] M. Schulze, G. A. Gibson, R. Katz, R. and D. A. Patterson. How reliable is a RAID? In *Proc. Spring COMPCON 89 Conf.*, pp. 118–123, Mar. 1989.

[7] A. Thomasian and J. Menon RAID 5 performance with distributed sparing. *IEEE Trans. on Parallel and Distributed Systems* **8**(6):640–657, June 1997.

[8] J. Wilkes, R. Golding, C. Stealin and T. Sullivan. The HP AutoRaid hierarchical storage system. *ACM Trans. on Computer Systems* **14**(1): 1–29, Feb. 1996

[9] L. Xu and J. Bruck: X-code: MDS array codes with optimal encoding. *IEEE Trans. on Information Theory*. **45**(1):272–276, Jan. 1999.