Simple Data Entanglement Layouts with High Reliability

Vero Estrada-Galiñanes*
Computer Science Department
University of Neuchâtel
Neuchâtel, NE, Switzerland

Jehan-François Pâris
Department of Computer Science
University of Houston
Houston, TX, USA

Pascal Felber
Computer Science Department
University of Neuchâtel
Neuchâtel, NE, Switzerland

***Abstract*—We study the reliability of open and close entanglements, two simple data distribution layouts for log-structured append-only storage systems. Both techniques use equal numbers of data and parity drives and generate their parity data by computing the exclusive or (XOR) of the most recently appended data with the contents of their last parity drive. While open entanglements maintain an open chain of data and parity drives, closed entanglements include the exclusive or of the contents of their first and last data drives. We evaluate five-year reliabilities of open and closed entanglements, for two different array sizes and drive failure rates. Our results show that open entanglements provide much better five-year reliabilities than mirroring and reduce the probability of a data loss by at least 90 percent over a period of five years. Closed entanglements perform even better and reduce the same probability by at least 98 percent.**

***Index Terms*—storage systems; magnetic disks; system reliability; fault-tolerance.**

I. Introduction

Despite recent advances in solid state storage, the lower cost of magnetic disk drives ensures that they remain today the most widespread storage medium in large data centers. One of the main disadvantage of these drives is their poor reliability [1]–[4]. As a result, all disk-based long-term storage solutions incorporate enough data redundancy to be able to reconstruct the data stored on any individual drive. These solutions are as diverse as mirroring, RAID level 5 [5], [6], RAID level 6 [7], [8], two-dimensional RAID arrays [9], [10], and SSPiRAL arrays [11].

Entanglements [12], [13] trade space for increased reliability and faster updates, especially in the case of log-structured append-only storage systems. Simple entanglements require equal numbers of data and parity drives; therefore, they have the same space overhead as mirroring. In counterpart, a simple open entanglement chain with $2n$ drives will tolerate the failure of any of its drives and the simultaneous failure of any two of them, except for the two last drives, which is much better than a mirrored organization. At the same time, appending a block to the entanglement will require one read and two writes while RAID level 6 and two-dimensional RAID arrays will require two reads and three writes.

We present here a full probabilistic analysis of the reliability offered by simple entanglements. We note first that entanglement chains can be closed to increase their reliability and show that the conversion process is both fast and reversible. We then model each entanglement as a Markov chain under standard stochastic assumptions and use our model to investigate the behavior of open and closed entanglement chains for four different scenarios, namely:

1) A small array consisting of 20 fairly reliable drives with a mean time to failures, MTTF, of 200,000 hours.
2) The same array with drives having an MTTF of only 35,000 hours. These disks would fail at the rate of 25 percent per year.
3) A medium-size array consisting of 50 fairly reliable drives with an MTTF of 200,000 hours.
4) The same array with drives having an MTTF of only 35,000 hours.

Our results show that open entanglements provide much better five-year reliabilities than mirroring and reduce the probability of a data loss by at least 90 percent over a period of five years, which corresponds to the maximum useful lifetime of most consumer-class drives [2]. Closed entanglements perform even better and reduce that probability by at least 98 percent.

The remainder of this paper is organized as follows. Section II introduces simple entanglements. Section III discusses possible array organizations. Section IV discusses the vulnerability of open and close entanglements to double, triple, and quadruple drive failures. Section V introduces our Markov model and presents the results of our investigation. Section VI sketches possible extensions, Section VIII presents our conclusions.

II. Simple Data Entanglement Layouts

The idea behind simple data entanglement is to intertwine data and parity blocks using the bitwise XOR operation with the goal of *refreshing* previously calculated parities and increasing the scope of redundant information. The process builds a chain that associates old information to the new data. In this way, old blocks will become indirectly dependent on blocks that will be inserted in future without the need to run the algorithm more than once.

A pseudocode description of the simple entanglement encoder algorithm is shown in Algorithm 1. The encoder has time complexity $\mathcal{O}(n)$ with n being the number of data blocks that are entangled and space complexity $\mathcal{O}(2n)$ since it requires 100 percent extra storage, i.e., the overhead equivalent

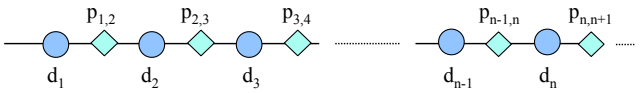
* VEG was partially supported by *SNSF Doc.Mobility Project 162014*

978-1-5090-5252-3/16/$31.00 ©2016 IEEE

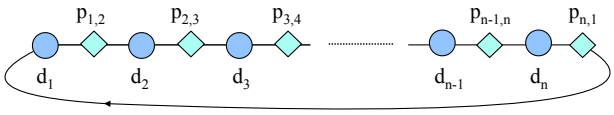
Algorithm 1 — Simple entanglement encoder.

1: *pool*: Queue containing blocks ready to be encoded

2: **while** ¬IsEmpty(*pool*) **do**
3: $u \leftarrow$GetParity() ▷ Get from cache
4: $v \leftarrow$Dequeue(*pool*) ▷ Get oldest element
5: $w \leftarrow$Entangle(u, v) ▷ Compute $u \oplus v$
6: WriteData(v) ▷ Write to array
7: WriteParity(w) ▷ Write to cache and array
8: **end while**



(a) Open Entanglements



(b) Closed Entanglements

Fig. 1: Open and closed entanglement chains

to mirroring. To improve performance, the encoder uses a write-through cache to keep in memory the last parity used in the entanglement. Data blocks are kept in a queue and they are encoded sequentially per their arrival order. We use the following notations:

d_i is a data block. Any object that is stored in the system is split in one or more d_i. The index i indicates the order in which the blocks are entangled and their position in the chain.
$p_{i,j}$ is the parity block associated with data blocks d_i and d_j.
D_i is a drive that stores data blocks.
P_i is a drive that stores parity blocks.

The general expression that defines a data entanglement is

$$p_{i,i+1} \leftarrow d_i \oplus p_{i-1,i}\,, \tag{1}$$

for i greater than zero. An equivalent formula was used to build the more complex entanglements presented in [12].

We define two classes of entanglements, namely, open and closed entanglement chains (Figure 1).

A. Open Entanglements

The encoder algorithm creates a never-ending chain. As long as space is available, the encoder will keep adding entangled blocks at the right extremity of the continuously growing chain. When the encoder starts for the first time, there is no parity block $p_{0,1}$ to use on the right part of Equation 1. The encoder will instead copy d_1 into $p_{1,2}$, because it is equivalent to assuming the existence of a fictitious parity block $p_{0,1}$ that only contains zeroes. The value of the next parity block, block $p_{2,3}$, is computed using Equation 1, that is:

$$p_{2,3} \leftarrow d_2 \oplus p_{1,2}\,.$$

Similarly, the decoding equations are derived from Equation 1. They result from the associative property of the exclusive or (XOR) operator and the iterative process that entangles blocks while building the entanglement chain. If a data block is not available, it can be rebuilt using:

$$d_i \leftarrow p_{i-1,i} \oplus p_{i,i+1}\,.$$

In addition, there are two ways of recovering parity blocks. One way is using Equation 1 and the other is using:

$$p_{i,i+1} \leftarrow d_{i+1} \oplus p_{i+1,i+2}\,.$$

Note that this second expression does not apply to the last parity block $p_{n,n+1}$ of an entanglement chain as neither d_{i+1} nor $p_{i+1,i+2}$ exist. As a result, the content of the last data block d_n will be irrecoverably lost if both blocks d_n and $p_{n,n+1}$ are lost.

B. Closed Entanglements

Closing the entanglement chain eliminates this fatal double failure by entangling the last data block d_n with the first data block d_1. A closed entanglement is built exactly as an open entanglement until there are no more data to be added and the chain can be closed. When this happens, the content of the first parity block is recomputed using:

$$p_{1,2} \leftarrow d_1 \oplus p_{n,1}\,,$$

where $p_{n,1}$ is the last parity block of the closed entanglement chain. As a result, the content of $p_{n,1}$ can now be reconstituted using:

$$p_{n,1} \leftarrow d_1 \oplus p_{1,2}\,.$$

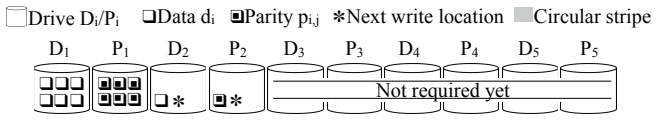
The content of parity disk $p_{2,3}$ will remain unchanged but it will now be defined as:

$$p_{2,3} \leftarrow d_2 \oplus d_1\,,$$

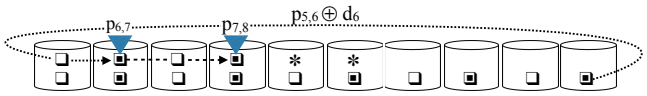
where d_1 replaces $p_{1,2}$. As the conversion process is $\mathcal{O}(1)$ with respect to the length of the chain, closing an open entanglement is both fast and easy. The reverse process is even simpler: it only requires overwriting the content of parity block $p_{1,2}$ with the content of data block d_1. As a result, it is always possible to add elements to a closed entanglement chain by reopening it.

III. Array Organizations

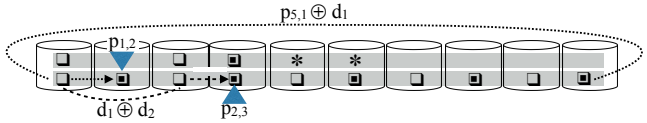
The entanglement algorithm presented in the previous section organizes blocks into a log. There are several choices for writing the blocks to disks. We sketch here two possible array organizations: *full-partition* write and *block-level striping*. As both organizations will require equal numbers of data drives and parity drives, all arrays will have an even number of drives.



(a) Full-partition write with unknown entanglement class (not yet defined)

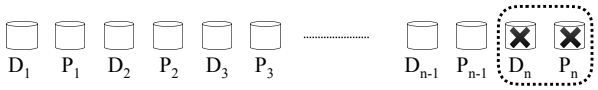


(b) Block-level striping with open entanglement chain

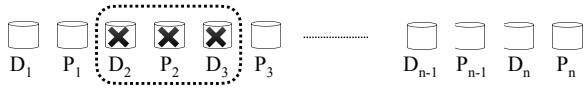


(c) Block-level striping with closed entanglement chain

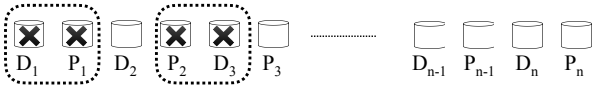
Fig. 2: Array organizations in combination with open and closed entanglements



(a) Type A failure



(b) Type B failure



(c) Type C failure

Fig. 3: The three irreducible fatal failure patterns of open entanglements

A. Full Partition Write

In this approach, blocks are written sequentially on the same drive. The process does not spread content across drives. Instead, it waits until the current drive is full before writing into a new drive. This data layout is best suited for archival applications with very low read to write ratio. Most of the drives will remain idle and can be powered off as in a massive arrays of idle disks, MAID, configuration [14], which could result in significant energy savings.

Another advantage of the approach is its scalability as drives can be added to the array at any time. The minimum requirement is two drives, but this would correspond to a mirror configuration. To create a true entanglement chain, at least four drives are needed. In addition, a closed entanglement chain that can tolerate all double drive failures would require at least three data drives, for a total of six drives.

The main limitation of this data layout is its write penalty as three drives are involved in the operation. To complete a write, the system needs to read a parity from an extant parity drive, XOR that parity with the new data and write the new data block and the new parity block on two different drives. A new parity is computed for every write. The general encoder algorithm presented in section II uses a write-through cache to avoid the extra read. The problem is that we would need a cache large enough to store the contents of a whole parity drive in order to eliminate the read penalty.

Figure 2a shows an example of a ten-disk array with four active drives. As writes only involve the last two most recently written drives of the array, the array could grow indefinitely if the application requires it.

B. Block-level striping

This second approach distributes data over all available drives to improve performance. New content is split into data blocks of size b. These data blocks are then spread across some or all of the n data drives.

Block-level striping requires to decide between an open and a closed entanglement when the array is set up. Figure 2 shows instances of both organizations. Their main difference is the way parities $p_{1,2}$ and $p_{2,3}$ are calculated. In an open chain, unless otherwise stated, the elements stored in the same disk or partition are part of the same growing chain. Elements located at n hops are written to the same disk. For example, Figure 2b shows how to calculate $p_{6,7}$. It is the second element stored in drive P_1. The figure also shows how to use $p_{6,7}$ to compute $p_{7,8}$, which is stored in drive P_2. Special elements $p_{1,2}$ and $p_{2,3}$ are calculated only once. Figure 2c shows how to calculate the special elements in a closed chain. Each closed chain forms a circular stripe. One of the chains is not yet finished, therefore, its element $p_{1,2}$ is not yet calculated. As a result, all blocks stored in the same disk will be part of independent stripes.

IV. Vulnerability Analysis

In this section, we will evaluate the probabilities that open entanglements, closed entanglements and mirrored organizations will not be able to tolerate double, triple and quadruple drive failures. Our focus will be on entanglements using the full partition write approach as it simplifies our analysis. In all three cases, we will begin by searching for irreducible failure patterns, that is, the specific failure patterns that involve a minimum number of failed drives [15].

A. Open Entanglements

As we can see on Figure 3, open entanglements exhibit three irreducible failure patterns, namely:

1) The failure of the last data drive of the entanglement, say, drive D_n in our example, and its associated parity drive P_n. We will call this failure a *type A* failure.

2) The failure of two consecutively numbered data drives, say, drives D_2 and D_3 in our example, and the parity drive in between the two failed data drives, that is, parity drive P_2. We will call this failure a *type B* failure.
3) The failure of two data drives, say, data drives D_1 and D_3 in our example, and all the parity drives between them, say parity drives P_1 and P_2 in our example. We will call this failure a *type C* failure.

In all three cases, the array will lack the information to reconstruct the contents of the failed data drive(s).

Consider now an open entanglement with $2n$ drives and assume it experiences the simultaneous failure of two of its drives. Out of the $\binom{2n}{2}$ possible double failures, only the type A failure will result in a data loss. As a result, the probability α that the entanglement will not tolerate a double drive failure is:

$$\alpha = \frac{1}{\binom{2n}{2}}$$

The triple failures that will result in a data loss include:

1) The type A double failure mentioned above combined with the failure of any of the $2n-2$ remaining drives.
2) Any of the $n-1$ type B triple failures mentioned above.

Hence, the probability β that the entanglement will not tolerate a triple drive failure is:

$$\beta = \frac{3n-3}{\binom{2n}{3}}$$

The quadruple failures that will result in a data loss include:

1) The type A double failure mentioned above combined with the failure of two of the $2n-2$ remaining drives for a total of $\binom{2n-2}{2}$ fatal quadruple failures.
2) Any of the $n-1$ type B failures mentioned above combined with the failure of any of the remaining $2n-3$ drives for a total of $(n-1)(2n-3)$ fatal quadruple failures. We need to subtract one from that product not to count twice the failure of drives D_{n-1}, P_{n-1}, D_n and P_n.
3) Any of the $n-2$ type C failures involving a data drive D_i, a data drive D_{i+2}, and the parity drives P_i and P_{i+1}.

Hence, the probability γ that the entanglement will not tolerate a quadruple drive failure is:

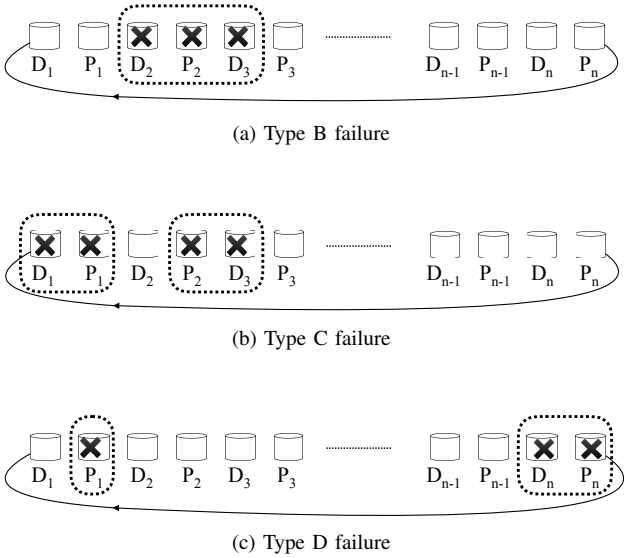
$$\gamma = \frac{\binom{2n-2}{2} + (n-1)(2n-3) - 1 + (n-2)}{\binom{2n}{4}}$$

B. Closed Entanglements

Closed entanglements can tolerate all double drive failures without experiencing a data loss. As we can see on Figure 4, they share the same B and C irreducible failure patterns as open entanglements and have an additional irreducible triple failure that involves drives D_n, P_n and P_1. We will call this failure a *type D* failure.

Given that closed entanglements tolerate all double failures without data loss, the probability α that the entanglement will not tolerate a double failure is zero.

The triple failures that will result in a data loss include:



(a) Type B failure

(b) Type C failure

(c) Type D failure

Fig. 4: The three irreducible fatal failure patterns of closed entanglements

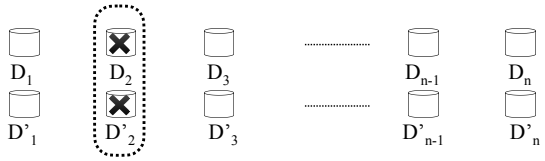


Fig. 5: The sole irreducible failure pattern of mirrored organizations

1) Any of the same $n-1$ type B triple failures as open entanglements.
2) One type D triple failure.

As a result, the probability β that the entanglement will not tolerate a triple drive failure is:

$$\beta = \frac{n}{\binom{2n}{3}}$$

In the same way, the quadruple failures that will result in a data loss are:

1) Any of the $n-1$ type B failures combined with the failure of any of the remaining $2n-3$ drives.
2) The single type D failure combined with the failure of any of the remaining $2n-3$ drives.
3) Any of the $n-2$ type C failures involving a data drive D_i, a data drive D_{i+2}, and the parity drives P_i and P_{i+1}.

The probability γ that the entanglement will not tolerate a quadruple drive failure is:

$$\gamma = \frac{n(2n-3) + (n-2)}{\binom{2n}{4}}$$

C. Mirrored organizations

As we can see on Figure 5, the sole irreducible failure pattern for mirrored organizations is the failure of a mirrored pair. As

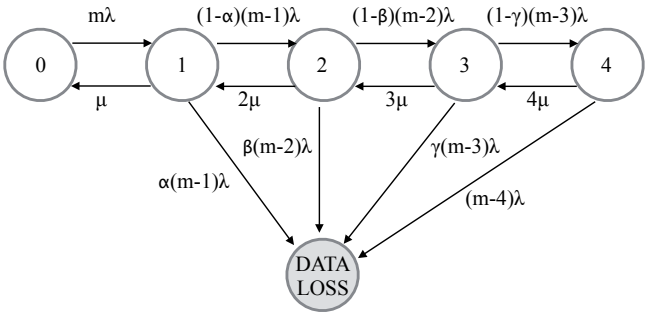


Fig. 6: State transition probability diagram

a result, the probability α that a mirrored organization with $2n$ drives will not tolerate a double drive failure is:

$$\alpha = \frac{n}{\binom{2n}{2}}$$

The triple failures that will result in a data loss is the failure of any of its n mirrored pairs combined with the failure of any of the $2n-2$ remaining drives. Hence, the probability β that the mirrored organization will not tolerate a triple drive failure is:

$$\beta = \frac{n(n-2)}{\binom{2n}{3}}$$

In the same way, the number of quadruple failures that will result in a data loss comprise the failure of any of its n mirrored pairs combined with the failure of two of the $2n-2$ remaining drives. We must however subtract from this total the $\binom{n}{2}$ quadruple failures consisting of the failure of two mirrored pairs of drives in order not to count them twice. As a result, the probability γ that the organization will not tolerate a quadruple drive failure is:

$$\gamma = \frac{n\binom{2n-2}{2} - \binom{n}{2}}{\binom{2n}{4}}$$

V. 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 intractable as the complexity of the system grows. Instead, a simpler option is to use the five-year reliability of the array. As this value is typically very close to 1, we will express it in "nines" using the formula number_of_nines $= -\log_{10}(1-R_d)$, where R_d is the five-year reliability of the array. Thus a reliability of 99.9 percent would be represented by three nines, a reliability of 99.99 percent by four nines, and so on.

We develop first a generic Markov model that will apply to open entanglements, closed entanglements and mirrored organizations. The specific behavior of each fault-tolerant drive array will be represented by the four parameters m, α, β, and γ, where $m = 2n$ is the number of disks in the array and α, β, and γ are the respective probabilities that the array will not tolerate the simultaneous failures of two, three or four drives. In all three cases, we will neglect the probability that the array will tolerate a quintuple drive failure, assuming that this probability is small enough to be neglected.

The model consists of an array of drives with independent failure modes. Whenever a drive fails, a repair process is immediately initiated for that drive. Should several drives fail, this repair process will be performed in parallel on those drives. We assume that drive failures are independent events and are exponentially distributed with mean λ. In addition, we require repairs to be exponentially distributed with mean μ. Both hypotheses are necessary to represent our system by a Markov process with a finite number of states.

Figure 6 displays the state transition probability diagram. State <0> is the initial state where all m drives are operational and no drive has failed. Should any of the drives fail, the system would move to state <1> with an aggregate failure rate $m\lambda$. Since some double failures can be fatal, the two possible failure transitions from state <1> are:

1) A transition to the data loss state with rate $\alpha(m-1)\lambda$ where the actual value of the α parameter will depend on the specific storage organization, as computed in previous section.
2) A transition to state <2> with rate $(1-\alpha)(m-1)\lambda$.

In the same way, the two failure transition from state <2> are:

1) A transition to the data loss state with rate $\beta(m-2)\lambda$ where the actual value of the β parameter will depend on the specific storage organization.
2) A transition to state <3> with rate $(1-\beta)(m-2)\lambda$.

Following the same pattern, the two failure transition from state <3> are:

1) A transition to the data loss state with rate $\gamma(m-3)\lambda$ where the actual value of the γ parameter will depend on the specific storage organization.
2) A transition to state <4> with rate $(1-\gamma)(m-3)\lambda$.

As we did not take into account the possibility that the array could survive a quintuple failure, there is a single failure transition leaving state <4>.

Recovery transitions are more straightforward: they bring the array from state <4> to state <3>, then from state <3> to state <2> and so on until the system returns to its initial state <0>.

The Kolmogorov system of differential equations that

describes the behavior of the storage organization is:

$$\frac{dp_0(t)}{dt} = -m\lambda p_0(t) + \mu p_1(t)$$
$$\frac{dp_1(t)}{dt} = -((m-1)\lambda + \mu)p_1(t) + m\lambda p_0(t) + 2\mu p_2(t)$$
$$\frac{dp_2(t)}{dt} = -((m-2)\lambda + 2\mu)p_2(t) + (1-\alpha)(m-1)\lambda p_1(t) + 3\mu p_3(t)$$
$$\frac{dp_3(t)}{dt} = -((m-3)\lambda + 3\mu)p_3(t) + (1-\beta)(m-2)\lambda p_2(t) + 4\mu p_4(t)$$
$$\frac{dp_4(t)}{dt} = -((m-4)\lambda + 4\mu)p_4(t) + (1-\gamma)(m-3)\lambda p_3(t)\,,$$

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 $i \neq 0$.

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

$$\text{MTTDL} = \sum_{i=0}^{4} p_i^*(0)\,,$$

where $p_i^*(s)$ is the Laplace transform of $p_i(t)$, we compute the Laplace transforms of the above equations and we solve them for $s = 0$ and a fixed value of m [16]. We then use this result to compute the mean time to data loss (MTTDL) of our system and convert this MTTDL into a five-year reliability, using the formula:

$$R_d = \exp\left(-\frac{d}{\text{MTTDL}}\right),$$

where d is a five-year interval expressed in the same units as the MTTDL. Observe that the above formula implicitly assumes that long-term failure rate $1/\text{MTTDL}$ does not significantly differ from the average failure rate over the first five years of the array.

We analyzed the reliability of both open and close entanglements and compared them with the reliability of mirrored solutions under four different array configurations, namely:

1) A small array of 20 drives and a drive mean time to failure (MTTF) of 200,000 hours.
2) The same array with a drive MTTF of 35,000 hours.
3) A medium size array of 50 drives and a drive MTTF of 200,000 hours.
4) The same array with a drive MTTF of 35,000 hours.

A drive MTTF of 200,000 hours corresponds to an annual failure rate of 4.28 percent, which represents what can be expected from an array built with very good drives. The annual failure rate is calculated using the inverse of the MTTF expressed in years. A drive MTTF of 35,000 hours corresponds to an annual failure rate of 25 percent. While this failure rate is pathological, it is neither exceptional nor confined to disks of dubious origin. Beach [2] reported just such a rate for a batch of 539 disks coming from a reputable manufacturer.

Figure 7 summarizes the findings. It shows the five-year reliabilities of the four configurations described above for mean time to repair (MTTR) varying between half a day and one week. Since reliabilities are expressed in nines, each unit increment on the vertical scale corresponds to a 90 percent reduction of the probability of a data loss.

As we can see, open entanglements provide much better five-year reliabilities than mirroring and reduce the probability of a data loss by 90 percent for the small array and by 98 percent for the large array. The very good performance of open entanglements for large arrays should not surprise us given that these entanglements only have a single fatal double failure.

As expected, closed entanglements perform even better and reduce the probability of a data loss:

- by at least 99.87 percent for the small disk array and a 200,000-hour MTTF,
- by at least 99.93 percent for the large disk array and a 200,000-hour MTTF,
- by at least 99.21 percent for the small disk array and a 35,000-hour MTTF,
- by at least 98.68 percent for the large disk array and a 35,000-hour MTTF.

We should note that these are minimum values observed for very large drive repair times. The improvements observed for a more reasonable two -day drive repair time vary between 99.79 and 99.98 percent.

Two features of our model may affect the accuracy of our results. First, we assumed that drive failures were independent events, which is not always true. Second, we assumed that all quintuple drive failures were fatal. This assumption is fairly reasonable as long as the quintuple disk failures remain rare events, which is certainly true for small arrays consisting of drives with high MTTFs and low MTTRs. This is less true for larger arrays especially if their drives have lower MTTFs and higher MTTRs. As these arrays will experience more drive failures and have their failed drives remain for longer periods of time in that state, quintuple drive failures will become less uncommon. Assuming that all these failures are fatal will then provide pessimistic evaluations of the array five-year reliability.

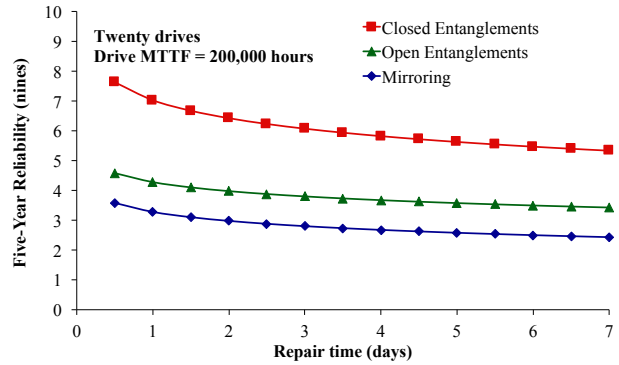
VI. Possible Extensions

While both open and closed entanglements provide much higher five-year array reliabilities than mirroring, there are circumstances where even higher levels of data protection must be sought. This could be the case if the array includes disk drives that exhibit high failure rates or if geographic considerations result in longer repair times. Let us show how we can address that issue by eliminating all fatal double drive failures in the case of open entanglements and all fatal triple drive failures in the case of their closed counterparts.

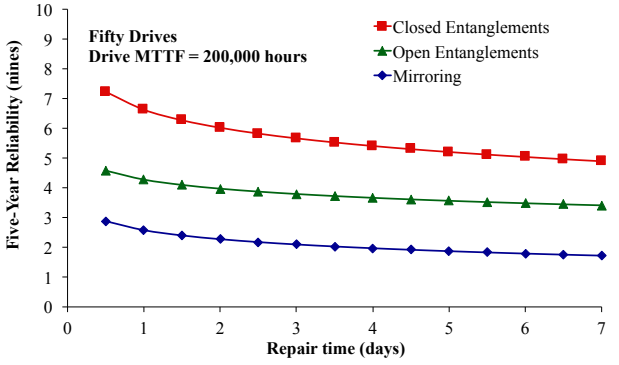
A. Open Entanglements

As we have seen in Section IV, an open entanglement with n data drives and n parity drives will not tolerate the combined failure of its last data drive D_n and its associated parity drive, drive P_n.

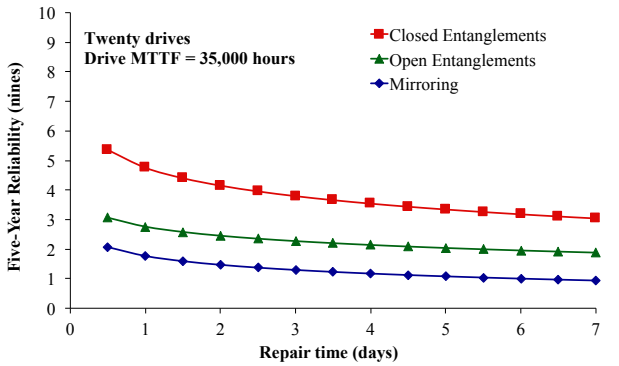
As we can see on Figure 8, the simplest solution is to mirror data drive D_n, thus adding an additional data drive D'_n.



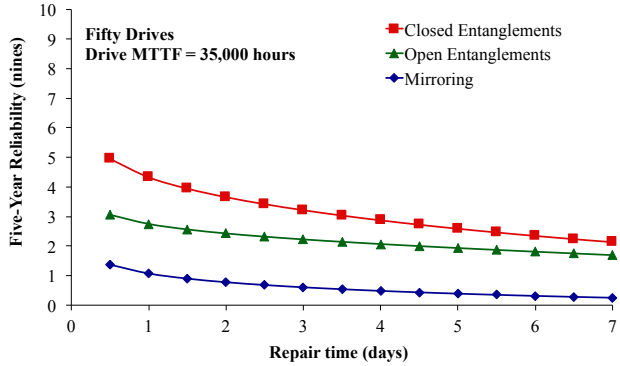
(a) 20-disk array with MTTF of 200,000 hours



(b) 50-disk array with MTTF of 200,000 hours



(c) 20-disk array with MTTF of 35,000 hours



(d) 50-disk array with MTTF of 35,000 hours

Fig. 7: Five-year reliability of four array configurations for mean time to repair varying between half a day and one week

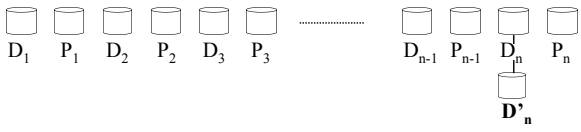


Fig. 8: An open entanglement that tolerates all double failures

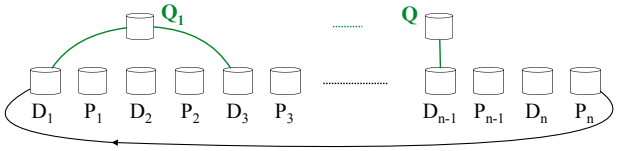


Fig. 9: A closed entanglement that tolerates all triple failures

The outcome will be an array with $2n + 1$ drives that would tolerate double drive failures and provide the same reliability as a closed entanglement with $2n$ drives.

B. Closed Entanglements

Recall that the sole fatal triple failures in closed entanglements are (a) type B triple failures involving D_k, its associated parity drive P_k, and the next data drive D_{k+1} and (b) a single type D triple failure involving D_n, P_n and P_1. So, if we number sequentially all data drives starting with D_1, all type B triple fatal failures will involve both an odd-numbered and an even-numbered data drive. Thus any mechanism that will allow the recovery of any failed odd-numbered data drive will eliminate all fatal type B triple failures.

To achieve this goal, we group all odd-numbered data drives into pairs (D_1, D_3), (D_5, D_7) and add to each pair an extra parity drive Q_j such that:

$$Q_j = D_{4k+1} \oplus D_{4k+3}\,,$$

where $0 \leq k \leq \left\lfloor \frac{n-3}{4} \right\rfloor$ and n is the number of data drive in the entanglement. In half the cases, the pairing process will leave the last odd-numbered data drive alone. Then, two cases need to be considered:

1) If n is odd, the last odd-numbered data drive is drive D_n. We do not need extra protection since the drive is already entangled with the first drive of the chain, namely drive D_1.

2) If n is even, the last odd-numbered data drive is drive D_{n-1}. To protect data against the simultaneous failure of drives D_{n-2}, P_{n-2} and D_{n-1} and parity drive P_{n-2}, we must mirror drive D_{n-1} and have

$$Q_x = D_{n-1}\,.$$

Figure 9 illustrates this last case.

VII. Previous Work

RAID arrays were the first disk array organizations to utilize erasure coding in order to protect data against disk failures [5]–[7]. While RAID levels 3, 4 and 5 only tolerate single disk failures, RAID level 6 organizations use $(n-2)$-out-of-n codes to protect data against double disk failures [8]. EvenOdd, Row-Diagonal Parity and the Liberation Codes are three implementations of RAID level that use only XOR operations to construct their parity information [17]–[21]. Huang and Xu proposed a coding scheme correcting triple failures [22].

Two-dimensional RAID arrays, or 2d-Parity arrays, were investigated by Schwarz [23] and Hellerstein et al. [9] who noted that these arrays tolerated all double disk failures but did not investigate how they reacted to triple or quadruple disk failures. More recently, Lee patented a two-dimensional disk array organization with prompt parity updates in one dimension and delayed parity updates in the second dimension [24].

SSPiRAL (Survivable Storage using Parity in Redundant Array Layouts) [11] layouts use simple t parity computations to provide high reliability and maintainability. Every SSPiRAL arrangement is defined by its number of unique data nodes (its degree), the number of nodes that contribute to constructing a parity node (its x-order), and the total number of nodes. For instance, a SSPiRAL layout of degree 3 and x-order 2 would use two nodes to build a parity node, and consist of three data nodes and two to four parity nodes.

VIII. Conclusion

Our study reveals that open and closed entanglements provide better five-year reliability than mirroring, reducing the probability of data loss by respectively 90 and 98 percent. Furthermore, these techniques are very efficient and simple to implement, either in software or hardware, and are hence of practical interest for cloud storage systems.

Acknowledgement

V.E.G. thanks Prof. Ethan Miller for hosting her during part of this work. Thanks are also due to the Computer Science Department of University of Houston for their hospitality during the author trip to Houston that motivated initial discussions for this work.

References

[1] L. N. Bairavasundaram, G. R. Goodson, S. Pasupathy, and J. Schindler, "An analysis of latent sector errors in disk drives," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 35. ACM, 2007, pp. 289–300.

[2] B. Beach, "What hard drive should i buy?" Backblaze Blog from January, January 2014.

[3] E. Pinheiro, W.-D. Weber, and L. A. Barroso, "Failure trends in a large disk drive population." in *FAST*, vol. 7, 2007, pp. 17–23.

[4] B. Schroeder and G. A. Gibson, "Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you?" in *FAST*, vol. 7, 2007, pp. 1–16.

[5] 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 (CSUR)*, vol. 26, no. 2, pp. 145–185, 1994.

[6] D. A. Patterson, G. Gibson, and R. H. Katz, "A case for redundant arrays of inexpensive disks (RAID)," in *Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '88. New York, NY, USA: ACM, 1988, pp. 109–116. [Online]. Available: http://doi.acm.org/10.1145/50202.50214

[7] T. J. Schwarz and W. A. Burkhard, "RAID organization and performance." in *ICDCS*, 1992, pp. 318–325.

[8] W. A. Burkhard and J. Menon, "Disk array storage system reliability," in *Fault-Tolerant Computing, 1993. FTCS-23. Digest of Papers., The Twenty-Third International Symposium on*. IEEE, 1993, pp. 432–441.

[9] L. Hellerstein, G. A. Gibson, R. M. Karp, R. H. Katz, and D. A. Patterson, "Coding techniques for handling failures in large disk arrays," *Algorithmica*, vol. 12, no. 2-3, pp. 182–208, 1994.

[10] T. Schwarz, A. Amer, T. Kroeger, E. L. Miller, D. D. E. Long, and J.-F. Pâris, "RESAR: Reliable storage at exabyte scale," in *Proceedings of the 24th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2016)*, Sep. 2016.

[11] A. Amer, D. D. Long, T. Schwarz, and J.-F. Paris, "Increased reliability with SSPiRAL data layouts," in *2008 IEEE International Symposium on Modeling, Analysis and Simulation of Computers and Telecommunication Systems*. IEEE, 2008, pp. 1–10.

[12] V. E. Galinanes and P. Felber, "Helical entanglement codes: An efficient approach for designing robust distributed storage systems," in *Symposium on Self-Stabilizing Systems*. Springer, 2013, pp. 32–44.

[13] ——, "Ensuring data durability with increasingly interdependent content," in *2015 IEEE International Conference on Cluster Computing*. IEEE, 2015, pp. 162–165.

[14] D. Colarelli and D. Grunwald, "Massive arrays of idle disks for storage archives," in *Proceedings of the 2002 ACM/IEEE conference on Supercomputing*. IEEE Computer Society Press, 2002, pp. 1–11.

[15] I. Corderí, T. Schwarz, A. Amer, D. D. Long, and J.-F. Pâris, "Self-adjusting two-failure tolerant disk arrays," in *Petascale Data Storage Workshop (PDSW), 2010 5th*. IEEE, 2010, pp. 1–5.

[16] M. Rausand and A. Høyland, *System reliability theory: models, statistical methods, and applications*. John Wiley & Sons, 2004, vol. 396.

[17] M. Blaum, J. Brady, J. Bruck, and J. Menon, "EVENODD: An efficient scheme for tolerating double disk failures in raid architectures," *IEEE Transactions on computers*, vol. 44, no. 2, pp. 192–202, 1995.

[18] P. Corbett, B. English, A. Goel, T. Grcanac, S. Kleiman, J. Leong, and S. Sankar, "Row-diagonal parity for double disk failure correction," in *FAST-2004: 3rd Usenix Conference on File and Storage Technologies*, 2004.

[19] W. Gang, L. Xiaoguang, L. Sheng, X. Guangjun, and L. Jing, "Generalizing RDP codes using the combinatorial method," in *Network Computing and Applications, 2008. NCA'08. Seventh IEEE International Symposium on*. IEEE, 2008, pp. 93–100.

[20] J. S. Plank, "A new minimum density RAID-6 code with a word size of eight," in *Network Computing and Applications, 2008. NCA'08. Seventh IEEE International Symposium on*. IEEE, 2008, pp. 85–92.

[21] ——, "The RAID-6 liber8tion code," *International Journal of High Performance Computing Applications*, 2009.

[22] C. Huang and L. Xu, "STAR: An efficient coding scheme for correcting triple storage node failures," *IEEE Transactions on Computers*, vol. 57, no. 7, pp. 889–901, 2008.

[23] T. J. E. Schwarz, "Reliability and performance of disk arrays," Ph.D. dissertation, UNIVERSITY OF CALIFORNIA, SAN DIEGO, 1994.

[24] W. S. Lee, "Two-dimensional storage array with prompt parity in one dimension and delayed parity in a second dimension," Jan. 6 2004, uS Patent 6,675,318.