Proc. of the International Conference on Electrical, Computer and Energy Technologies (ICECET)
9-10 December 2021, Cape Town-South Africa

Three-dimensional RAID Arrays with Fast Repairs

Jehan-François Pâris
Department of Computer Science
University of Houston
Houston, TX 77204-3010*, USA*
jfparis@uh.edu

Thomas Schwarz, SJ
Department of Computer Science
Marquette University
Milwaukee, WI 53233*, USA*
thomas.schwarz@marquette.edu

***Abstract*—Large data storage systems often use Reed-Solomon erasure codes to protect their static data against triple or even quadruple device failures. A main drawback of this approach is the high cost of recovering the contents of failed devices, as it requires accessing the contents of a large number of surviving devices. We present a three-dimensional RAID organization that adds vertical parity devices to a stack of identical two-dimensional RAID arrays. These new vertical parity devices will let the organization recover faster from all single device failures while greatly reducing the risk of data loss. Depending on the way the vertical parities are defined, the new arrays will either tolerate all triple failures and more than 99.9 percent of all quadruple failures, or all quintuple failures and more than 99.995 percent of all sextuple failures.**

Keywords— Data storage systems, fault tolerant systems, parity check codes

I. Introduction

As we store ever-increasing amounts of data in digital form, traditional fault-tolerant solutions such as mirroring or RAID level 6 arrays fail to provide the required levels of protection [1] [2]. To give one example, both the Google File System [3] and Windows Azure Storage [4] maintain three replicas of all their active data. While this approach protects the data against all double failures and nearly all triple failures, it also multiplies by three the data storage costs. The preferred way to slash these costs is to use Reed-Solomon erasure codes [5] to store inactive data. One good example is the erasure code used by BackBlaze cloud backup service [6]. It splits every incoming file into 17 equal-size shards and calculates 3 parity shards so that the file contents can be reconstituted from any 17 of these 20 shards. The main advantage of the approach is its very low space overhead, as only 3 out of the 20 shards contain redundant data.

A common disadvantage of Reed-Solomon codes is the higher cost of data recovery operations after the loss of a disk. Consider, for instance, the Backblaze erasure code. Reconstituting the contents of a single failed disk requires reading in the entire contents of at least 17 of the remaining 19 operational disks. The process is likely to take at least half a day and involve the transfer of hundreds of terabytes. This cannot be done without stressing the communication layer of the storage system and slowing down all other data requests for the entire duration of the process. In addition, requests directed to the data not yet reconstituted will be insufferably slow, as each request will require accessing data spread over 17 different disks.

Such service disruptions may be acceptable in an online backup service since most of its data are likely to be never retrieved. However, they cannot be tolerated in a distributed file system or a cloud storage service. This situation has led to the development of several new coding schemes that require considerably fewer disk accesses to recover from the failure of a single disk. Two of these schemes deserve our attention because they were developed for well-known large-scale storage solutions. These are the Windows Azure local reconstruction code (LRC) [7] and the HDFS-XORBAS locally repairable code [8]. Both schemes add local parity blocks to the parity blocks of an extant Reed-Solomon code. Other proposals include Rotated Reed-Solomon Codes [9], Shingled Erasure Codes [10], Hitchhiker [11] and Bundles of RAID Arrays [12].

A common thread in most of these codes is their complexity. To give one example, deriving the correct coding equations for the local parities is not a trivial task. Our proposal only uses exclusive-or operations, which does away with the need for processors that can do Galois field calculation at blazing speed [12]. Even more importantly, it avoids having to set up decoding by, for instance, inverting a matrix as a generalized linear code does. While there are several implementations of RAID 6 arrays that use only exclusive-or operations to compute parity [14], they all require more complex parity calculations than our technique.

We propose an even simpler approach that does not require any kind of fine-tuning, and only involves elementary parity calculations. Our proposal stacks together a small number of *layers* all containing identical two-dimensional RAID arrays and top these layers with vertical parities such that each vertical parity stripe will contain one disk from each layer. The resulting three-dimensional structure allows us to define two levels of protection depending on the number of vertical parity stripes each organization includes. These are:

1. A basic protection level where we only create vertical parity stripes for the data disks of each layer. As a result, each data disk will participate in three parity stripes, namely the two in-layer parity stripes at the intersection of which the disk is located, and its vertical parity stripe. Since each data update is propagated to three disks, the array will be able to tolerate the simultaneous failure of three arbitrary disks without any data loss. In addition, most but not all quadruple and quintuple failures will not result in a data loss.
2. An expanded protection level where both data and parity disks of each layer participate in the vertical parity

978-1-6654-4231-2/21/$31.00 ©2021 IEEE

scheme. As a result, each data update will be propagated to (a) the vertical parity of the data disk, (b) the parity disks of its two in-layer parity stripes, and (c) the vertical parity disks of these two disks. Because each data update is now propagated to five separate disks, the array will be able to tolerate all quintuple disk failures. Moreover, most but not all sextuple failures will not result in a data loss.

Unlike most other proposals, our solution can be deployed without having to compute the value of any coefficient. The sole parameters to specify are the number of planes and the number of disks per plane. In addition, predicting the fault-tolerance of a specific installation only requires the application of one or two simple formulas.

This flexibility provides users with a useful tool for managing the space overhead of their disk arrays and take advantage of the low space overheads of large two-dimensional arrays. While our smallest arrays have space overheads of up to 50 percent, much larger arrays have lower overheads than the best existing solutions while providing equal or better data protection.

The remainder of this paper is organized as follows. Section II reviews relevant previous work. Section III introduces our three-dimensional arrays and discusses their vulnerability to multiple failures. Section IV evaluates their performance and compares them to those of other codes with fast repairs. Section V has our conclusions.

II. Previous Work

In this section, we review the most significant previous work on locally repairable codes. Space considerations have prevented us from being more complete.

A. *Windows Azure Local Reconstruction Codes*

Azure Local Reconstruction Codes (LRC) [7] partition k data blocks into l sets of k/l blocks each and will have l local parity blocks and r global parity blocks. The code will be able to recover from the loss of either any single data block or any single local parity block using exactly k/l block reads. At the same time, the recovery performance of the code depends on the coefficients selected for the linear expressions defining the $l + r$ parity blocks. For instance, a (6, 2, 2) LRC can recover all single data block failures using three read operations, has a space overhead of $4/10$ = 40 percent and can tolerate all triple failures and most quadruple failures without data loss.

B. *HDFS-XORBAS locally repairable codes*

The HDFS-XORBAS locally repairable code [8] comprises 10 data blocks and 7 parity blocks. The four global parity blocks P_1, P_2, P_3, and P_4 are built with a standard Reed-Solomon code and ensure that the code can tolerate the loss of four arbitrary blocks. Blocks S_1, S_2, and S_3 are local parity blocks whose aim is to reduce the cost of recovering from single block failures. Block S_1 is a linear combination

$$S = c_1D_1 \oplus c_2D_2 \oplus c_3D_3 \oplus c_4D_4 \oplus c_5D_5$$

of the contents of data blocks D_1 to D_5 and block S_2 is similarly obtained from data blocks D_6 to D_{10}. Block S_3 is an *implied* parity block. It is not stored anywhere but will be created on demand. Overall, HDFS-XORBAS can recover from all single block failures using five read operations and will protect its contents against all quadruple disk failures. Its space overhead is $6/16$ = 37.5 percent.

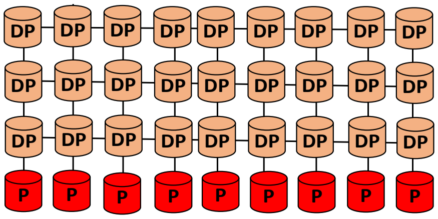


Fig. 1. A bundle of three RAID arrays protected by nine column parity blocks. All disks in the three RAID arrays contain both data (D) and parity (P) blocks.

C. *Rotated Reed-Solomon codes*

Khan et al. [9] analyzed some of the most popular erasure codes and proposed a new class of codes that perform degraded reads more efficiently than all known codes, but otherwise keep the reliability and performance properties of extant Reed-Solomon codes. The emphasis of their work was on minimizing overall data transfers instead of minimizing the number of disks involved in the reconstruction.

D. *Shingled Erasure Codes (SHEC)*

Miyamae et al. [10] have proposed a shingled disk array organization consisting of k data disks and m parity disks. Each of these m parity disks contains the XOR of the contents of l data disks, which are said to form a *locality*. As $ml > k$, we can assign each of the k data disks to exactly ml/k distinct localities in a way that ensures that the array will tolerate the simultaneous failure of up to ml/k disks. In addition, the organization guarantees that recovering from a single block failure will require exactly l block reads. Since all their parities are local, these codes can recover from double and triple failures without involving any of the remaining data disks.

E. *Hitchhiker*

Hitchhiker [11] uses a modified Reed-Solomon code tailored to reduce both network traffic and disk I/O by around 25 to 45 percent during the reconstruction of missing data, without requiring any additional storage.

F. *Bundles of RAID arrays*

Bundles of RAID arrays [12] provide both high reliability and fast repairs by adding column parity disks to a bundle of RAID level 5 or level 6 arrays [1] [2] [15]. Fig. 1 depicts one such bundle. It consists of three RAID arrays each having nine disks and nine additional column parity disks such that each column parity stripe contains one disk from each RAID array and each of the disks in the three RAID arrays belongs to one parity stripe. Thanks to these nine column parities, recovering from a single disk failure will only require three read operations. In addition, bundles of RAID level 5 arrays can recover without data loss from all triple and at least 96 percent of quadruple disk failures while bundles of RAID level 6 arrays can recover from all quintuple and 99.9 percent of all sextuple disk failures.

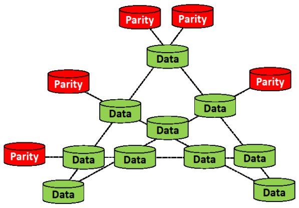
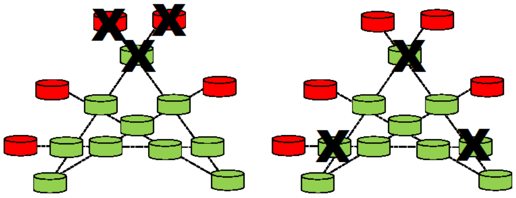


Fig.2. A two-dimensional RAID array with five parity disks and ten data disks.



Type A Type B

Fig. 3. Types of fatal triple failures.

III. Our Proposal

Two-dimensional RAID arrays [2] [16] protect the data they store against all double disk failures and most triple disk failures by requiring that:

1. Each data disk belongs to two distinct parity stripes.
2. Two distinct parity stripes have at most one common data disk.

Among these organizations, the most space-efficient are those who also require all parity stripes to intersect with each other [17]. Given that we can place one data disk at each of these intersections, an array with n_p parity disks will be able to protect the contents of $n_d = \binom{n_p}{2}$ data disks. As an example, the disk array of Fig. 2 contains five parity disks and ten data disks, thus having a space overhead of 33.3 percent. While this overhead is higher than that of a RAID level 6 array with the same number of data disks, it quickly decreases as the array size increases. For instance, an array with 12 parity disks will be able to accommodate 66 data disks, which corresponds to a space overhead of less than 16 percent.

Even so, these arrays do not fulfill all the requirements of modern storage systems. First, they do not protect their data against all triple disk failures. As we can see on Fig. 3, two kinds of triple failures can result in a data loss, namely:

1. Type A fatal triple failures that correspond to the simultaneous failure of two parity disks and the data disk located at the intersection of the parity stripes they define.
2. Type B fatal triple failures that correspond to the simultaneous failure of three data disks such that each of the three data disks has one parity stripe in common with each of the two other data disks.

Overall, the array will not be able to tolerate $n_d + \binom{n_p}{3}$ of the $\binom{n_p + n_d}{3}$ possible triple failures.

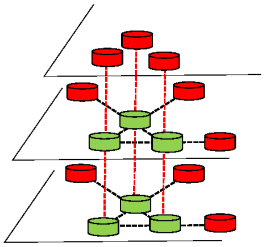


Fig. 4. A very small three-dimensional organization using three vertical parity blocks (on top) to provide basic protection to two two-dimensional RAID arrays (middle and bottom layer). We expect actual three-dimensional organizations to have up to five distinct layers and many more disks per layer.

Second, the number of disks that we must access to recover from the loss of a single disk quickly grows as the size of the array grows. This is not a problem with small arrays such as the array of Fig. 2. Since each parity stripe has only four data disks per array, recovering from a single disk failure will require combining the contents of four disks. Consider now the case of an array with 12 parity disks and 66 data disks. Since each data disk has to be on two distinct parity stripes, each parity stripe will contain 11 data disks, which means we will now need to combine the contents of 11 separate disks to recover from the failure of a single disk.

We propose to address these two limitations by combining a small number m of identical two-dimensional arrays into a three-dimensional structure where each two-dimensional array will form a separate layer of the structure. We add to these layers vertical parity disks such that each of the parity stripes defined by these vertical parity disks contains one disk from each layer of the organization. In addition, we will define two distinct levels of protection based on the number of vertical parity disks included in the organization.

A. Basic protection level

In the basic protection level, the vertical parity strips only contain the data disks of each layer. Fig. 4 displays a very small instance of that organization. Its two bottom layers contain two identical two-dimensional RAID arrays with three data disks and three parity disks. The top layer contain vertical parity disks that define vertical parity stripes such that:

1. Each data disk belongs to exactly one vertical parity stripe.
2. Each vertical parity stripe contains exactly one data disk from each two-dimensional RAID array.

There are thus as many vertical parity disks as there are data disks in each array and our organization will contain a total of $mn_p + n_d$ parity disks and mn_d data disks. Its space overhead will thus be

$$\frac{mn_p + n_d}{m(n_p + n_d) + n_d}.$$

Since each of the mn_d data disks belongs to a vertical parity stripe, recovering the contents of any data disk will never require accessing the contents of more than m different disks. Thus, keeping m low will suffice to guarantee the fast recovery

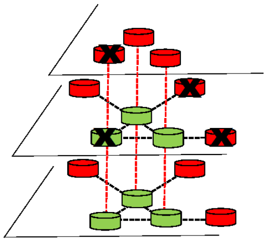


Fig. 5. A fatal quadruple failure resulting from a type A failure in one of the two two-dimensional RAID arrays accompanied by the failure of the vertical parity disk of the failed data disk.

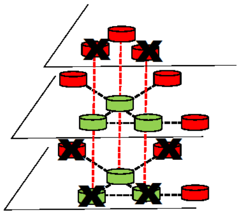


Fig. 6. A fatal sextuple failure resulting from the simultaneous failure of two data disks sharing a common parity stripe, the two parity disks of their other parity stripes and their two vertical parity disks.

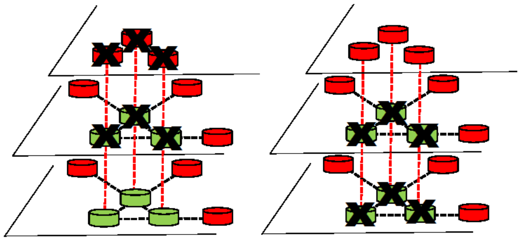


Fig.7. Two fatal sextuple failures caused by type B failures.

of any single data disk failure. As the parity disks of each layer are not included in the vertical protection scheme, recovering from the failure of any of them will still require accessing the contents of the $2n_d/n_p$ data disks in its stripe.

Updating the contents of any data disk will now require three additional updates, as we will now have to update the vertical parity of the data disk in addition to the parity disks of its two layer parity stripes. Given that all updates are now propagated to three distinct parity disks, we can expect that the structure will tolerate all possible failures without experiencing any data loss. We can indeed quickly check that no type A nor type B failure in any of the m layers can now result in a data loss as the vertical parity stripes always provide a way to recover the contents of at least one of the three sites that failed.

This is not true of all quadruple failures. We observe instead that the structure will not be able to recover from type A failure in any of its m layers when the type A failure is accompanied with the failure of the vertical parity disk that would otherwise have been used to recover the contents of the failed data disk. Fig. 5 displays one of these mn_d fatal quadruple failures. Conversely, the structure remains able to tolerate all type B failures in any of its m layers as all three data disks involved in the failure have recovery paths independent of each other. Overall, mn_d of the $\binom{m(n_p+n_d)+n_d}{4}$ quadruple failures will result in a data loss.

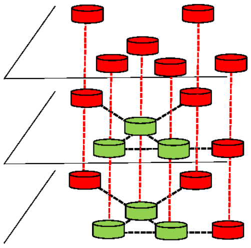


Fig. 8. A very small three-dimensional organization using six vertical parity blocks (on top) to provide expanded protection to two two-dimensional RAID arrays (middle and bottom layer).

The sole fatal quintuple failures are caused by the failure of the four sites involved in any of the mn_d possible fatal quadruple failures plus the failure of any of the remaining $m(n_p+n_d)+n_d-4$ disks for a total of

$$mn_d\left(\left(n_p+n_d\right)+n_d-4\right)$$

fatal quintuple failures.

Sextuple fatal failures can occur in many distinct forms. We can distinguish:

1. The failure of the four disks involved in any of the mn_d fatal quadruple failures plus the failure of two of the remaining disks. There are

$$mn_d\binom{\left(n_p+n_d\right)+n_d-4}{2}$$

distinct failures of that type.

2. The failure of two data disks sharing a common parity stripe and the failure of the two parity disks of their other parity stripes plus the failure of the vertical parity disks of the two data disks. Fig. 6 displays one of the

$$m\binom{n_p}{3}$$

distinct failures of that type.

3. A type B failure in one of the m two-dimensional RAID arrays plus the failures of the vertical parity disks of the three failed disks or a second type B failure involving the same disks in one of the remaining $m-1$ two-dimensional arrays. Fig. 7 displays two of the

$$\binom{m+1}{2}\binom{n_p}{3}$$

distinct failures of that type.

B. Expanded protection level

As Fig. 8 shows, the expanded protection level adds n_p additional vertical parity stripes for the n_p parity disks of each layer. There are thus as many vertical parity disks as there are disks in each array and our organization will contain a total of

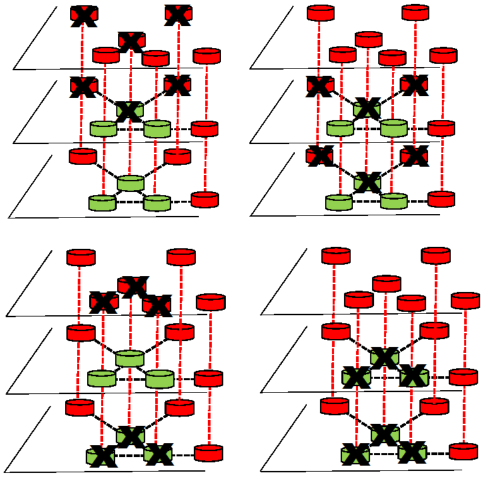


Fig. 9. Four instances of fatal sextuple failures for a three-dimensional organization providing extended protection to its two-dimensional RAID arrays.

$(m+1)n_p + n_d$ parity disks for its mn_d data disks. Its space overheads will thus be

$$\frac{(m+1)n_p + n_d}{(m+1)(n_p+n_d)}.$$

Since each of the $m(n_p + n_d)$ disks in any of the m layers now belongs to a vertical parity stripe, recovering the contents of any data disk will never require accessing the contents of more than m different disks.

Updating the contents of any data disk will now require five additional updates as we will have to update (a) the vertical parity of the data disk, (b) the parity disks of its two layer parity stripes, and (c) the vertical parity disks of these two disks. Given that all updates are now propagated to five distinct parity disks, we can expect that the structure will tolerate all possible failures without experiencing any data loss. We can quickly check that no type A nor type B failures in any of the m layers combined with two additional arbitrary failures can result in a data loss since the vertical parity stripes always provide three separate ways to recover the contents of at least one of the three sites that failed.

As we can expect, some sextuple failures will result in a data loss. As Fig. 9 shows, fatal failures include a type A or a type B failure in one of the layers of the organization. More specifically, we can distinguish:

1. Fatal sextuple failures resulting from either a type A failure and the failure of the three associated parity disks or two type A failures involving six disks sharing the same three parity stripes. There are $\binom{m+1}{2} n_d$ such failures.
2. Fatal sextuple failures resulting from either a type B failure plus the failure of the three associated parity disks, or two type B failures involving six disks sharing the same three parity stripes. There are $\binom{m+1}{2}\binom{n_p}{3}$ such failures.

Observing that $n_d = \binom{n_p}{2}$, the total number of fatal sextuple failures becomes

$$\binom{m+1}{2}\binom{n_p}{2} + \binom{m+1}{2}\binom{n_p}{3} = \binom{m+1}{2}\binom{n_p+1}{3}$$

The sole fatal failures involving seven disks are a failure of the six sites involved in any fatal sextuple failures plus the failure of any of the remaining $(m+1)(n_p+n_d) - 6$ disks.

IV. Performance Evaluation

A cursory comparison of our two organizations could lead us to prefer the extended protection scheme over the basic protection scheme because it tolerates two more failures without significantly increasing the space overhead of the array. While this is true, there are other factors to consider.

First, all arrays managed by our basic protection scheme will tolerate nearly all quadruple and quintuple disk failures. Even the smallest array will tolerate 99.9 percent of all quadruple failures and 99 percent of all quintuple failures while arrays totaling more than 20 data disks will tolerate 99.99 percent of all quadruple failures and 99.9 percent of all quintuple failures. To give a point of comparison, a 60-disk array that maintains three replicas of all its data will only tolerate 99.77 percent of all quadruple failures and 99.5 percent of all quintuple failures.

Second, our extended protection scheme has significantly higher update costs due to the need to propagate all updates to five different parity disks. Recall that the basic protection scheme propagates all updates to three parity disks, namely to the two parity disks of the two parity stripes at the intersection of which the updated data disk is located and the vertical parity disk of the data disk. The extended protection scheme also propagates the updates of two of these three parity disks to their own vertical parity disks, which is a more complex process. As a result, the extended protection scheme is poorly suited to applications where the data are likely to be ever updated.

This cumbersome update process offers nevertheless the major advantage of providing an exceptional level of data protection. Even the smallest array will tolerate 99.995 percent of all sextuple failures and 99.96 percent of all septuple failures.

One of the best ways to measure the fault-tolerance of a storage array is to evaluate the survival probability of the array in the presence of increasing numbers of disk failures. We present in Figs. 10 to 12 the results of simulations that measure the survival probability of a basic three-dimensional array with 45 data disks and 10 parity disks per layer and 45 vertical parity disks. We used runs of 1,000,000 randomly chosen sets of f failed disks and repeated each run at least 12 times. The 95% confidence intervals for the true probability of survival given f failed disks was less than 1% of the value, which explains why the error intervals show up as single short horizontal lines instead of an "I" structure. We also indicate the range where the survival probability is higher than 99.9% (or three nines). For example, the three-layered structure survives 12 failures with a

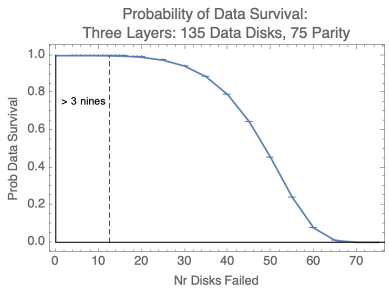


Fig. 10. Data survival probability of a three-layered basic three-dimensional disk array with 135 data disks and 75 parity disks as a function of the number of disks that failed.

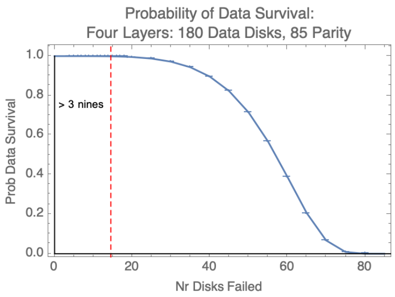


Fig. 11. Data survival probability of a four-layered basic three-dimensional disk array with 180 data disks and 85 parity disks as a function of the number of disks that failed.

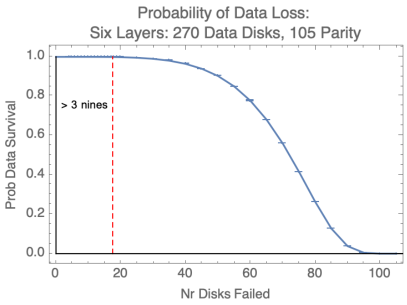


Fig. 12. Data survival probability of a six-layered three-dimensional disk array with 270 data disks and 105 parity disks as a function of the number of disks that failed.

probability of 0.99914424 plus or minus 0.00000723. When the number of failed disks exceeds the number of parity disks, then data loss is certain, but as we can see, the disk array can suffer large failure numbers while still being more likely to retain all of its data. For instance, in the three layer disk array, 47 disks can have failed and the chance of all data surviving is still greater than 50 percent.

Comparing the performance of our three-dimensional array organization with those of other organizations that allow local recovery of single disk failures is not an easy task due to a lack of data about the performance of those organizations. In addition, these organizations typically involve small numbers of devices while our three-dimensional array organization requires many more devices to operate at peak efficiency.

TABLE I COMPARING THE PERFORMANCE OF THE BASIC PROTECTION LEVEL FOR 3D ORGANIZATIONS WITH THOSE OF BUNDLES OF RAID 5 ARRAYS.

Number of units (arrays/layers)	Disks per unit	Storage capacity (disks)	Space overhead	Fatal quadruple failures
3 RAID 5	12	33	31.3%	0.204%
2 layers	31	30	47.4%	0.031%
2×3 RAID 5	12	66	31.3%	> 0.204%
3 layers	28	63	40.0%	0.001%
4×3 RAID 5	12	132	31.3%	> 0.204%
3 layers	55	135	35.7%	0.0002%
6×3 RAID 5	12	198	31.3%	> 0.204%
3 layers	66	198	34.0%	6×10^{-7}

TABLE II COMPARING THE PERFORMANCE OF THE EXTENDED PROTECTION LEVEL FOR 3D ORGANIZATIONS WITH THOSE OF BUNDLES OF RAID 6 ARRAYS.

Number of units (arrays/layers)	Disks per unit	Storage capacity (disks)	Space overhead	Fatal sextuple failures
3 RAID 6	12	30	37.5%	0.007%
3 layers	15	30	50.0%	2×10^{-6}
2×3 RAID 6	12	60	37.5%	> 0.007%
3 layers	28	63	43.8%	9×10^{-8}
4×3 RAID 6	12	120	37.5%	> 0.007%
3 layers	55	135	38.6%	4×10^{-9}
5×3 RAID 6	12	150	37.5%	> 0.007%
3 layers	66	165	37.5%	2×10^{-9}
6×3 RAID 6	12	180	37.5%	> 0.007%
3 layers	78	198	36.5%	$9 \times 10^{-10\times}$

The data we present here compare the performance of our new array organization with those of bundles of RAID arrays, a relatively recent proposal that was found to perform as well as the Windows Azure Local Reconstruction Codes and HDFS XORBAS. Like three-dimensional RAID arrays, bundles of RAID arrays offer two levels of data protection depending on the types of RAID arrays being clustered together.

Basic bundles of RAID arrays group together a small number of RAID level 5 arrays and add to them an array of parity disks that act in the same way as the vertical parity disks in our organization. As a result, each data update is propagated to three disks, namely,

1. The RAID disk that contains the corresponding parity block.
2. The column parity disk associated with the disk that contains the updated block.
3. The column parity disk associated with the RAID disk that contains the updated parity block.

Since the array propagates each update to three distinct devices, the bundle will tolerate all triple failures and most quadruple failures without data loss.

Comparing the performance of our basic protection scheme with those of bundles of RAID 5 arrays required us to make some assumptions about the two systems. We decided first to compare three-dimensional RAID arrays that had three layers with bundles of three RAID 5 arrays because both systems were able to recover from all single disk failures by accessing the contents of three disks. We also assumed that each RAID array consisted of 12 disks using 11/12 of its capacity to store data blocks and the remaining 1/12 to store parity blocks. As a result, each bundle of 36 disks contained the equivalent of 33 disks full of data. Finally, we decided that larger disk arrays would consist of multiple instances of the same original bundle. Therefore, a disk array capable of storing the equivalent of 66 disks full of data would be comprised of 2 independent bundles each consisting of 3 RAID 5 arrays of 12 disks each plus 12 additional parity blocks. We would refer to this organization as a 2×3 RAID 6.

Table I summarizes the results of our comparison. As we can see, bundles of RAID 5 arrays provide significantly lower space overheads than three-dimensional disk arrays for very small disk arrays but this advantage slowly erodes as the disk capacity increases. This should not surprise us because the RAID level 5 arrays that compose the bundles have a very low space overhead as parity data occupy only 1/12 of the total array capacity. At the same time, the two-dimensional RAID arrays that we use in our three-dimensional organizations are not very space efficient as long as they remain small. For instance, a two-dimensional array with 15 data disks will require 6 data disks and use 28.6 percent of its disk space to store parity information.

We also compared the performance of our extended protection scheme with those of bundles of RAID arrays after replacing their RAID 5 arrays by RAID 6 arrays. The change had a significant impact on the space overhead of the bundles, as 2/12 of the storage space of each array is now dedicated to parity data. In contrast, moving from a basic to an extended protection scheme only added n_p vertical parity disks to the n_d parity disks of the basic protection scheme.

Table II summarizes these new results. As we expected, the bundles of RAID 6 arrays have a significantly higher space overhead than the bundles of RAID 5 arrays. In addition, the differences between the space overheads of the bundles of RAID level 6 arrays and the three-dimension RAID arrays decreases much faster as the array sizes increase. We note indeed that our three-dimensional organization becomes more space-efficient than the bundle of RAID 6 arrays once the array contains more than 165 data disks.

Space overheads between 35 and 40 percent are within what we should expect. The Windows Azure local reconstruction code [7] has a space overhead of 40 percent even though it does not protect data better than our basic scheme. Similarly, HDFS-XORBAS [8] has a space overhead of 37.5 percent even though it does not tolerate all quintuple failures and requires accesses to five disks, instead of three, to recover from a single disk failure.

V. Conclusion

We have presented a three-dimensional RAID organization that adds vertical parity disks to a stack of two-dimensional RAID arrays. As we have seen, these new vertical parity disks serve two purposes. First, they provide faster recovery paths for all single data disk failures. Second, these vertical parity disks can greatly enhance the fault-tolerance of the organization. Depending on the number of vertical parity disks that they require, our organization can offer:

1. A basic protection scheme where only the data disks of each array are included in the vertical parity stripes defined by the vertical parity disks: this basic protection scheme can tolerate all triple failures and more than 99.9 percent of all quadruple failures and 99 percent of all quintuple failures without data loss.
2. An extended protection scheme where both the data disks and the parity disks of each array are included in the same vertical parity stripes. This extended protection scheme will tolerate all quintuple failures and more than 99.995 percent of all sextuple failures without data loss.

References

[1] W. A. Burkhard and J. Menon, “Disk Array Storage System Reliability,” in *Proc. FTCS*-23, pp. 432–441, June 1993.

[2] T. Schwarz, SJ, “Reliability and performance of disk arrays,” PhD Dissertation, CSE Dept., University of California, San Diego, 1994.

[3] S. Ghemawat , H. Gobioff, and S.-T. Leung, “The Google file system,” in *Proc.19th* SOSP, Oct. 2003.

[4] B. Calder, et al. “Windows Azure storage: A highly available cloud storage service with strong consistency,” in *Proc. 23rd* SOSP, Oct. 2011.

[5] I. S. Reed and G. Solomon, “Polynomial codes over certain finite fields", *SIAM Journal*, Vol. 8, No. 2, pp. 300–304, 1960.

[6] B. Beach, “BackBlaze open sources Reed-Solomon erasure coding source code,” https://www.backblaze.com/blog/reed-solomon/, June 16, 2015, retrieved Nov. 10, 2021.

[7] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, and S,Yekhanin, “Erasure coding in Windows Azure storage,” in *Proc. 2012 USENIX ATC*, June 2012.

[8] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, and D. Borthakur, “XORing elephants: novel erasure codes for big data,” in *Proc. of the VLDB Endowment*, Vol. 6, No. 5, pp. 325-336, 2013.

[9] O. Khan, R. Burns, J. Plank, W. Pierce, and C. Huang, “Rethinking erasure codes for Cloud file systems: minimizing I/O for recovery and degraded reads,” *Proc 10th USENIX FAST Conf.*, Feb. 2012.

[10] T. Miyamae, T. Nakao, and K. Shiozawa, “Erasure code with shingled local parity groups for efficient recovery from multiple disk failures,” in *Proc. 10th USENIX Workshop on Hot Topics in System Dependability* (HotDep '14) Broomfield, CO, Oct. 2014.

[11] K. V. Rashmiet al. “A hitchhiker’s guide to fast and efficient data reconstruction in erasure-coded data centers,” in *Proc. 2014 ACM* SIGCOMM *Conf.*, Chicago, IL, Aug. 2014.

[12] J.-F. Pâris, “Bundling together RAID disk arrays for greater protection and easier repairs” in *Proc.27th MASCOTS Symp.*, Oct. 2019.

[13] J. Plank, K. Greenan, E. L. Miller, “Screaming Fast Galois Field Arithmetic Using Intel SIMD Extensions,” in *Proc. 11th USENIX FAST Conf.*, Feb. 2013.

[14] P. Corbett, B. English, A. Goel, T. Grcanac, S. Kleiman, J. Leong, and S. Sankar, “Row-diagonal parity for double disk failure correction,” *Proc. 3rd USENIX FAST Conf.* \, pp. 1–14, Mar.-Apr. 2004.

[15] D. A. Patterson, G. Gibson, and R. H. Katz, “A case for redundant arrays of inexpensive disks (RAID),” in *Proc. 1988 ACM SIGMOD Conf.*, pp. 109–116, June 1988.

[16] L. Hellerstein, G. A. Gibson, R. M. Karp, R. H. Katz, D. A. Patterson, “Coding Techniques for Handling Failure in Large Disk Arrays,” *Algorithmica*, Vol. 12. No.3–4, pp. 182–208, June 1994.

[17] J.-F. Pâris, A. Amer, and T. J. E. Schwarz, “Low-redundancy two-dimensional RAID arrays,” in *Proc. 2012 ICCCN, Data Storage Technology and Applications Symp.*, Jan.–Feb. 2012.