Fractional-Overlap Declustered Parity: Evaluating Reliability for Storage Systems

Huan Ke, Haryadi S. Gunawi
University of Chicago
{*huanke, haryadi*}*@uchicago.edu*

Dominic Manno, David Bonnie, Bradley W. Settlemyer
Los Alamos National Laboratory
{*dmanno,dbonnie,bws*}*@lanl.gov*

***Abstract*—In this paper, we propose a flexible and practical data protection scheme, fractional-overlap declustered parity (FODP), to explore the trade-offs between fault tolerance and rebuild performance. Our experiments show that FODP is able to bring forth up to 99% less probability of data loss in the presence of various failure regimes. Furthermore, by adding one additional spare drive capacity within each server, FODP yields up to 99% reduction in granularity of data loss.**

1. Introduction

The emergence of Big Data as a major resource supporting scientific discovery, business intelligence, and social media has led to the creation of massive storage systems [1], [2], [3], [4] both in cloud infrastructure and in on-premise data centers. These storage systems often incorporate hundreds or thousands of storage nodes and tens of thousands of spinning disks. To support the massive storage capacities required to support Big Data analytics disk drive vendors are increasing the capacity of common disk drives with 16TB and 20TB drives commonplace and 24TB drives announced for 2021. At the same time, disk enclosures have also increased in size and now commonly house 104 or 106 drives in only 4U of rack space with current racks typically 42U or 48U tall. Thus, within a single data center rack it has become commonplace to provide greater than 20PB of data capacity provided by more than 1000 disk drives and data centers may host hundreds or thousands of such racks to support modern digital data.

While the number of disks in data centers is growing, disk deployments are becoming less reliable due to increases in track density, enclosure density and data center trends toward operating disks in more extreme environments. Distributed storage systems also introduce new failure modes such as rack failures [5], [6], cascading failures [7], [8], [9], and power and cooling failures [10] that trigger the loss of multiple storage system components within compressed time windows. In order to make data services continuously available for both ingest and analysis it has become necessary to employ data reliability schemes that protect against multiple types and sources of failures. Many existing reliability schemes work on two extremes, either enhancing the rebuild performance [11], [12], [13] or improving fault tolerance [14], [15], but how the interactions between fault tolerance and rebuild time together impact system reliability is still unclear. Motivated by that, we design a practical and flexible tool, fractional-overlap declustered parity (FODP) to explore the trade-offs between the number of failure domains and rebuild performance. This gives us fine-grained control to accommodate different reliability requirements and system sizes. By utilizing Mutually Orthogonal Latin Squares (MOLS), FODP can uniformly distribute data and parity blocks across disks and map the given logical units in the specified physical disks. This allows for adding additional parity on top of the FODP data layout and which can reduce the quantity of data loss during failure bursts. To the best of our knowledge, this is the first work to achieve the goal of reducing the probability of data loss in the presence of correlated failures while significantly reducing the magnitude of lost data.

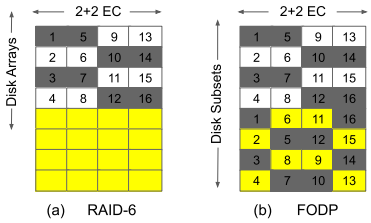

Figure 1. Comparison of data layouts in RAID-6 and FODP, where $2+2$ erasure code (EC) represents 2 data and 2 parity blocks configuration, which can tolerate up to 2 failures. RAID-6 separates disks into 4 independent disk arrays while FODP comprises of 8 disk subsets.

2. Fractional-Overlap Declustered Parity

Traditional declustered parity originally used Balanced Incomplete Block Designs (BIBD(v, k, λ)) [16], to distribute parity stripes of size k over v disks (e.g., declustered layout), where λ is the number of disk sets that each pairwise combination of disks appears in and $\lambda \geq 1$.When λ is at least 1 it ensures that all disks are able to equally participate in rebuilds because every disk has data overlapping with every other disk. For example, a declustered parity scheme that maps 2+2 stripes into a 16 disk enclosure, a λ value of 1 ensures that when a single disk fails the 15 remaining disks all participate equally in reading the surviving blocks of each degraded stripe to rebuild the failed disk. Less obvious is that the lower the λ value the more total failures that can be tolerated. Figure 1(a) illustrates

this property using 16 disks and 4 non-overlapping 2+2 RAID6 arrays (i.e. the minimum possible λ). In this case up to 8 disks can fail without data loss assuming that those failures are distributed evenly across RAID groups. Motivated by these observations, we propose *fractional-overlap declustered parity (FODP)* $\lambda < 1$, which relaxes the overlap constraint and allows disks to overlap less than once with every other disk. As a result, many pair-wise combinations exist, but a few pair-wise combinations are not generated. This $\lambda < 1$ scheme enables a tradeoff between rebuild performance and tolerating more disk failures by adjusting λ between the minimum and 1. Figure 1(b) re-organizes the RAID-6 data layout by adding 4 declustered disk subsets (e.g., the last 4 rows) in yellow and the 8 shaded failures are still tolerated, but each disk in FODP is involved in two disk subsets (i.e. 6 disks are able to participate in the reconstruction), which make the rebuild 2x faster than the RAID protection scheme.

	λ_{min}	$\lambda < 1$	$\lambda = 1$	$\lambda > 1$
Rebuild Efficiency	L	M	H	H
Fault Tolerance	H	H	M	L

TABLE 1. λ COMPARISON. H REPRESENTS HIGH, M REPRESENTS MEDIUM, AND L REPRESENTS LOW

Table 1 compares the trade-off space for different λ. In the case of λ_{min}, each disk is involved in a single disk subset, which is identical to a clustered RAID protection scheme with high failures tolerance and slow disk rebuild. The case of $\lambda < 1$ represents FODP, where placement groups only partially overlap which brings moderate rebuild performance and high fault tolerance. In the case of $\lambda = 1$, it is a special type of full declustering [17] that tolerates more disk failures compared to the case of $\lambda > 1$ with identical rebuild performance. Lastly, $\lambda > 1$ is widely used in declustered parity (DP) schemes and tolerates the fewest disk failures.

2.1. FODP Construction

One practical limitation for $\lambda = 1$ DP placement schemes is that they are hard to generate and are impossible for many disk configurations (e.g. 8+2 parity with a 20 disk enclosure). Because FODP relaxes the traditional constraint and allows $\lambda < 1$ it is able to achieve tradeoffs similar to the ideal DP configuration while also supporting more diverse parity schemes and disk enclosure sizes. In order to describe our latin squares approach of constructing FODP layouts we first quantify the *overlap fraction* λ for each disk as below.

$$\lambda = \frac{overlaps}{v - 1}$$

where each disk has (stripe-width-1) overlapping disks within a disk subset, and the *overlaps* is decided by the number of disk subsets * (stripe-width-1), which reflects the fraction that covers the remaining surviving (v-1) disks.

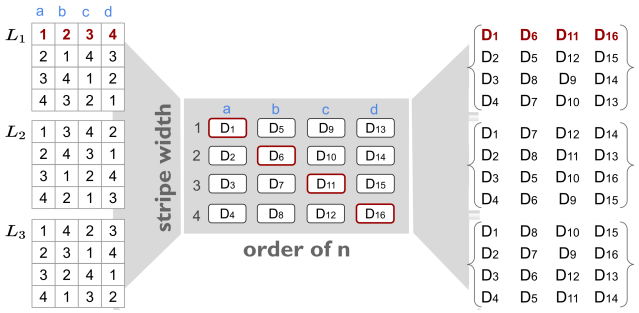

Figure 2. FODP layout by mapping MOLS into the disk matrix.

Definition 1. *A Latin square is a nxn array over n elements such that each element appears only once in each row and column.*

We start with a 4x4 Latin square L_1 in Figure 2, where each row and column are over elements of [1, 2, 3, 4]. To utilize the Latin square L_1, we organize the 16-disk enclosure into a disk matrix where the number of row is equal to stripe-width (e.g., rows 1, 2, 3, 4) and number of columns is equal to the Latin square order n (e.g., columns a, b, c, d). Similar to this, we specify columns a, b, c, d for Latin squares in top. By matching the 4x4 Latin square L_1 with the 4x4 disk matrix, each row is able to represent an independent 4-disk subset. For example, the first row [1, 2, 3, 4] in red represents the coordinate [(1 a), (2 b), (3 c), (4 d)], which translates into a new disk subset [D1, D6, D11, D16]. Likewise, it's the same for the other three rows. Therefore, the Latin square L_1 corresponds to 4 non-overlapping disk subsets that cover the whole disk matrix and the *overlap fraction* λ for each disk is 3/15.

Definition 2. *Two latin squares L_1 and L_2 are called mutually orthogonal when any ordered pair of entries from each Latin square in the same row and column occurs exactly once.*

As explained above, L_1 represents 4 non-overlapping disk subsets, which would be the same for every other Latin square. To generate more disk subsets, it is necessary to use additional Latin squares. The only concern is avoiding multiple overlaps with the existing disk subsets. In other words, the L_1 related disk subsets should overlap with L_2 related disk subsets by at most one disk. Figure 2 illustrates Mutually Orthogonal Latin Squares (MOLS) L_1, L_2 and L_3, where each row overlaps with each other row by at most one overlapping element (e.g., the only overlapping element of L_{11}([1, 2, 3, 4]) and L_{21}([1, 3, 4, 2]) is element 1). The idea behind it comes from unique order pair property in Mutually orthogonal Latin squares, which guarantees the single overlap property no matter how many Latin squares we use. In essence, by applying more MOLS, we are able to linearly increase the FODP overlap fraction λ. The n-order MOLS is known to exist when n is a power of a prime number [18]. Such valid n values exist for 4, 5, 7, 8, 9, 11, 13, 16, 17, 19, 23, 25, 27, 29, 31, 32, 37, 41,

43, 47, 49, 53, 59, 61, 64, 67, 71,73, 79, 81, 83, 89, 97 in the range of 100. If the disk matrix is rectangular and the number of columns meets the valid size in MOLS then this approach ensures each disk is included in an equal number of disk subsets guaranteeing an uniform and deterministic data distribution. If the number of disks do not form a rectangular matrix we can simply remove the last row of disks and make certain to randomly substitute those disks into subsequent MOLS. In this case additional overlaps are possible and uneven data distributions may occur, but all possible configurations are supported with similar fault tolerance and rebuild characteristics to perfectly rectangular matrices.

2.2. FODP-Plus-One

In FODP, if more than m failures occur simultaneously within a single disk subset then all data in this subset will be lost. Therefore, increasing the overlap fraction λ, which increase the number of disk subsets, will increase the probability of data loss because the $> m$ failures will hit one of the subsets with a higher probability. Thus the total number of disk subsets is a tradeoff between the probability of data loss and the *magnitude* of data lost during a loss event. FODP is originally designed for reducing the probability of data loss with the property of $\lambda < 1$. To avoid the loss of large amounts of data during a data loss event, we further propose *FODP-Plus-One* to add a layer of parity on top of the FODP stripes.

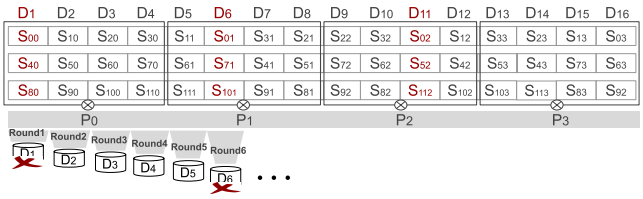

Figure 3. Additional parities P_1, P_2, P_3, P_4 on top of FODP stripes are placed in round robin fashion.

In Figure 3 we see the first blocks (e.g., $S_{00}, S_{10}...S_{11,0}$) in FODP stripes are always placed in column a (e.g., disks D_1, D_2, D_3, D_4), the second ones (e.g., $S_{01}, S_{11}...S_{11,1}$) are in column b, and so on. If we create an additional parity P_i for the i_{th} blocks in a set of FODP stripes, the parity P_i is able to further protect against FODP stripe loss by placing P_i on a disk in another column. This constructs a two-dimensional parity scheme, but uses an additional disk to store the vertical parity. Unlike existing schemes like BIBD, PDDL [13], and dRAID [11], the i_{th} blocks are always spread over the disk enclosure because the additional disk can be selected systematically. In the simplest rendition FODP-Plus-One simply places the additional parity data P_1, P_2, P_3, P_4 in round-robin fashion. A hypothetical refinement is to perform round-robin across the disks not in the original FODP disk subset, but the details of achieving balance with this approach are complicated compared to the round-robin approach presented here. Although the simple round-robin placement cannot further reduce the probability of data loss, it can maximally reduce the magnitude of lost data. In this work we will limit our analysis to data protection and address performance considerations in future work.

Overall, FODP-Plus-One reduces the magnitude of data loss because the additional parity on surviving drives can recover large amounts of the lost data. Continuing our example, assume D_1, D_6, and D_{11} fail simultaneously, only the stripe S_0 including blocks S_{00}, S_{01}, S_{02}, and S_{03} on disks [D_1, D_6, D_{11}, D_{16}] will lose data. The parities P_1, P_2, P_3, P_4 on surviving disks like D_2, D_3 D_4, D_5 and so on can help recover the vulnerable stripes. As a result, the magnitude of the lost data can be significantly reduced. If we configure FODP with the maximal overlap fraction within $\lambda < 1$, the total number of disk subsets is equal to the number of disk drives v. With one additional spare drive of capacity (e.g., $v+1$) the data lost is reduced by $(v-f)/v$ (where f is the number of disk failures).

3. Evaluation

In this section, we evaluate storage system reliability by addressing the following questions: (1) how do failure sizes and failure distributions impact reliability? (2) how does the overlap fraction explore the trade-offs between rebuild performance and fault tolerance? To mimic a real storage system, we consider Los Alamos National Laboratory's Campaign storage system [19] composed of 4 pods, each of which consists of 12 servers and each server is configured with two 84-disk JBODs (e.g., 168 disk drives per server). Note that to keep the FODP as similar to the Campaign storage systems 85% storage overhead we evaluate the protection schemes as if the servers have 169 disks per server and with parity configured at $11 + 2$. This is simply an artifact of trying to match the existing configuration most closely for fairer comparisons. To make these results applicable to emerging disk drive technology the simulation presented in this paper assumes that each disk drive is equipped with dual actuators and 16TB capacity, and the disk rebuild bandwidth has been set at approximately 50M/s and the copyback bandwidth is around 400MB/s.

3.1. Correlated Failures

Our study of correlated failures is limited due to the lack of contemporary failure data. In this section, we present a series of correlated failure models that model failures arriving closely in time, and we later evaluate the impact of different failure regimes on system designs.

3.1.1. Dense Failures. Some recent work [20] in high performance computing systems shows that failures are highly correlated in time resulting in time periods with higher failure density. It observes that 75% of failures just occur within 25% of the system lifetime. As a result, the periods of higher failure density could result in a MTBF multiples

higher than the average. In particular, failures in [6], [21] are sometimes time-correlated across hours and days due to environmental effects, firmware bugs, or transient workloads. Although these correlations do not hold over the life of systems, the existence of these highly correlated failures within compressed time windows may make existing storage system data protection schemes highly vulnerable to data loss.

3.1.2. Batch Failures. The assumption that failures occur separately from each other is not always valid because many failure types can be traced to batches of components (e.g. a run of disks manufactured with a less effective bearing lubricant). As shown in [8], if one disk has failed in a batch, it is more likely to trigger another disk failure in that same batch. To study batch failures, [22], [23] model the initial failures that happen randomly during the useful lifetime of disks in an independent way while the cascading characteristic results in failures that happen in rapid succession (e.g., 10 times faster [9]).

Type	Model
Dense failures	Exp($\frac{1}{MTBF}$) || Poisson($\frac{1}{MTBF}$)
Batch failures	Exp($\frac{1}{MTBF}$) & Exp($\frac{1}{0.1*MTBF}$)

TABLE 2. POISSON AND EXPONENTIAL DISTRIBUTION MODELS USED IN CORRELATED FAILURES.

Table 2 summarizes the failure models we will use in this paper. For dense failures, we use two separate distributions with failure events arriving according to a Poisson distribution or Exponential distribution. For batch failures, we use a model combining two Exponential distributions, where first failure may trigger an additional failure stream (e.g., 3 to 6 cascading failures [22]) that occurs at a 10x faster failure rate.

3.2. Impact of Failures

To study the effects of failure sizes and distributions on system reliability, we vary the percent of disk failures and set the ratio of MTBF to MTTR at 0.5 to parameterize the dense Poisson, dense Exponential, and batch cascade failures. We compare multiple parity schemes including RAID, traditional declustered parity (DP), Single-Overlap Declustered Parity(SODP), FODP (e.g., with maximal overlap fraction $13/14$), and FODP+1.

Figure 4 shows the probability of data loss (PDL) by varying the number of failed disks from 0.2% to 1.0%. The probability of data loss increases with the percent of failures for each failure distribution and protection scheme. Interestingly, the probability of data loss in DP slightly outperforms RAID for Poisson while RAID exceeds DP in Exponential. We note that the Exponential distribution results in more outlier failures which makes RAID, with its higher fault tolerance, stand out even more. Likewise, RAID is much more efficient than DP in batch failures, where tolerating more failures is most important. Note that,

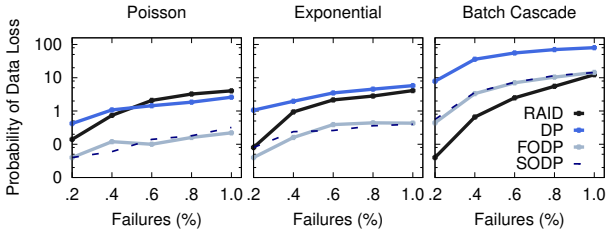

Figure 4. Comparison of failure sizes and distributions. From left to right, the figures show the probability of data loss (PDL) resulting from dense Poisson, dense Exponential, and batch cascade failures.

FODP+1 only improves FODP with respect to the amount of lost data instead of the probability of data loss, that is why we just compare FODP and SODP. Overall, FODP experiences almost the same probability of data loss as SODP with slightly lower rebuild performance and higher fault tolerance. Compared to RAID and DP, SODP and FODP experience lower PDL in Poisson and Exponential dense failures due to higher fault tolerance and faster rebuilds while they have slightly higher PDL rates in batch cascading failures where PDL is mainly governed by fault tolerance.

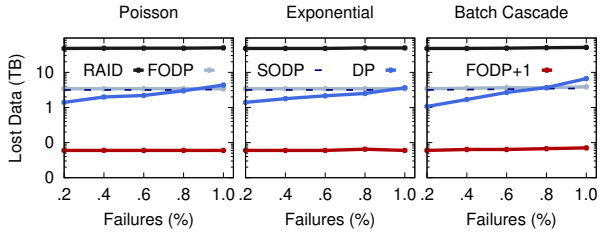

Figure 5. Comparison of the amount of lost data for each incident resulting from dense Poisson, dense Exponential, and batch cascade failures.

Figure 5 compares the average amount of lost data when a data loss event occurs. As discussed before, reducing the number of failure domains can reduces the probability of data loss at a cost of greater data lost during any data loss event. As expected, RAID experiences the highest amount of lost data, FODP and SODP have lower magnitudes of data loss with DP losing the least data. As the number of failed disks increases, the amount of lost data in RAID is slightly increased while the lost data in DP is greatly increased and even exceeds FODP and SODP in the presence of 1% failures in above three failure distributions. In SODP and FODP, additional failures increase the probability of data loss rather than the amount of lost data, which indicates that the data loss magnitude in FODP and SODP is not highly sensitive to such small variation in failures. Likewise, FODP+1 just experiences one data loss incident each time, thus the amount of lost data remains the same in Poisson and Exponential and just slightly increases in batch failures. With one additional spare drive capacity within each server FODP+1 not only reduces the probability of data loss like SODP and FODP but also significantly reduces the magnitude of data loss.

3.3. Impact of Overlap Fraction

To explore the effects of overlap fractions we adjust λ using FODP to compare the rebuild time and probability of data loss for Poisson, Exponential dense failures, and batch cascade failures.

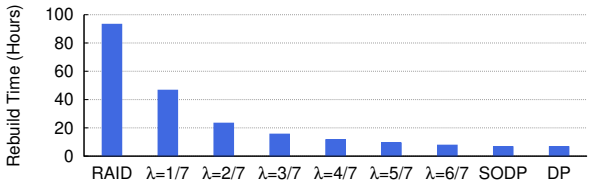

Figure 6. Rebuild time comparison.

Figure 6 compares the rebuild time among RAID, FODP with different overlap fractions λ (e.g., $1/7$-$6/7$), SODP and DP. As expected, there is a distinct correlation between overlap fraction and rebuild time. RAID takes the longest time (e.g., 93.2 hours) to rebuild a single disk failure. FODP with differing overlap fractions shows the diminishing rate of increase rebuild time as it approaches it's minimum rebuild time at $\lambda \geq 1$ configurations. We note that the improvement in rebuild times is not linear, but logarithmically decreasing due to the fixed amount of work being distributed over a larger number of disks.

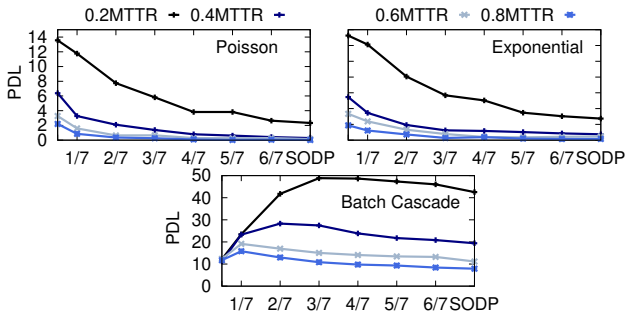

Figure 7. The figure shows the probability of data loss (PDL) resulting from 1% disk loss due to dense Poisson, dense Exponential and batch cascade failures.

Figure 7 compares the probability of data loss by varying overlap fraction λ from the minima $1/14$ (RAID) to the maxima $14/14$ (SODP) with 1% failures under various MTBF to MTTR ratios. As we can see, the probability of data loss in Poisson and Exponential failures decreases as λ increases. More interestingly we see that the PDL curves increase with λ up to some peak, but then decrease as λ approaches 1. This phenomena is due to the time required to achieve 1% disk failures and the rebuild rate of the selected λ. With the MTBF to MTTR ratio set at 0.2 it takes 22 hours for 1% of the disk population to fail. At $\lambda = 5/7$ the disk rebuild time is reduced to 11 hours and the PDL begins to decrease. We see this effect in all of the Batch Cascade curves where, as the MTBF is increased, the minimum necessary rebuild performance is decreased and lower lambda values become viable though better rebuild performance still provides incremental PDL improvements.

4. Related Work

Existing work for improving storage fault tolerance falls into two categories: (1) adding redundancy, and (2) reducing failure domains. In the case of adding redundancies, many erasure codes [14], [24] layouts the data with one horizontal and one vertical parity, which essentially protect data in two dimensions. By applying this design principle, CORE core creates additional vertical parities by XOR operations cross objects in different stripes. Furthermore, Uber Parity [25] and RAIDP [26] store the additional vertical parities on small auxiliary storage devices like NVRAM attached to disks. These local storage devices fail separately from disks, which can be used to recover data when disk failures overwhelm the single stripe redundancy. Copyset replication [15] was designed for reducing failure domains in data replication using three copies within large-scale clusters composed of thousands of nodes.

Many approaches to enhancing the reconstruction performance for declustered data have been described in literature. The construction of declustered layouts vary from deterministic distribution schemes like [12], [27] for small disk arrays to near-optimal schemes [11], [28] for large disk arrays. Single-Overlap Declustered Parity [17] applies the copyset idea to declustered parity to explore reducing the number of failure domains for declustered parity to dramatically improve fault tolerance while maintaining the identical rebuild performance. FODP, presented here, provides a more practical and flexible declustered parity scheme with similar failure and rebuild characteristics. Unlike HACFS [29] to use two different erasure codes, our FODP can adapt to different workload requirements with only one uniform erasure code. Furthermore, by adding one additional parity on top of FODP, our FODP plus one can dramatically reduce the magnitude of lost data.

5. Conclusion

Due to trends in disk drives, enclosures, and deployments we believe we are entering a period where failure events are becoming more common rather than less common. In this paper we revisit storage system fault tolerance and rebuild performance under a diverse set of correlated failure events. To study the joint effects of these reliability characteristics this paper describes a practical and flexible declustered parity scheme, fractional-overlap declustered parity (FODP), that enables the exploration of trade-offs between rebuild performance and the number of failure domains. Our experimental results suggest that equally emphasizing the two perspectives does not achieve a midpoint in system reliability – and is sometimes the least reliable configuration. Meanwhile, FODP can reduce probability of data loss by up to 99% compared to declustered parity for various failure regimes while FODP-Plus-One can further reduce 99% of the magnitude of data loss with a single additional spare drive capacity within each server.

Acknowledgments

This manuscript has been approved for unlimited release and has been assigned LA-UR-20-27110. This work has been co-authored by employees of Triad National Security, LLC which operates Los Alamos National Laboratory under Contract No. 89233218CNA000001 with the U.S. Department of Energy/National Nuclear Security Administration. The university authors were supported by funding from NSF(grant #CNS-1563956).

The publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of the manuscript, or allow others to do so, for United States Government purposes.

References

[1] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. The hadoop distributed file system. In *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)*, pages 1–10. Ieee, 2010.

[2] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The google file system. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 29–43, 2003.

[3] Sage A Weil, Scott A Brandt, Ethan L Miller, Darrell DE Long, and Carlos Maltzahn. Ceph: A scalable, high-performance distributed file system. In *Proceedings of the 7th symposium on Operating systems design and implementation*, pages 307–320, 2006.

[4] Brad Calder, Ju Wang, Aaron Ogus, Niranjan Nilakantan, Arild Skjolsvold, Sam McKelvie, Yikang Xu, Shashwat Srivastav, Jiesheng Wu, Huseyin Simitci, et al. Windows azure storage: a highly available cloud storage service with strong consistency. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, pages 143–157, 2011.

[5] Mi Zhang, Shujie Han, and Patrick PC Lee. A simulation analysis of reliability in erasure-coded data centers. In *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*, pages 144–153. IEEE, 2017.

[6] Daniel Ford, François Labelle, Florentina Popovici, Murray Stokely, Van-Anh Truong, Luiz Barroso, Carrie Grimes, and Sean Quinlan. Availability in globally distributed storage systems. 2010.

[7] Suman Nath, Haifeng Yu, Phillip B Gibbons, and Srinivasan Seshan. Subtleties in tolerating correlated failures in wide-area storage systems. In *NSDI*, volume 6, pages 225–238, 2006.

[8] Jehan-François Pâris and Darrell DE Long. Using device diversity to protect data against batch-correlated disk failures. In *Proceedings of the second ACM workshop on Storage security and survivability*, pages 47–52, 2006.

[9] Peter M Chen, Edward K Lee, Garth A Gibson, Randy H Katz, and David A Patterson. Raid: High-performance, reliable secondary storage. *ACM Computing Surveys (CSUR)*, 26(2):145–185, 1994.

[10] Weihang Jiang, Chongfeng Hu, Yuanyuan Zhou, and Arkady Kanevsky. Are disks the dominant contributor for storage failures? a comprehensive study of storage subsystem failure characteristics. *ACM Transactions on Storage (TOS)*, 4(3):1–25, 2008.

[11] Zfs draid. https://github.com/openzfs/zfs/wiki/dRAID-HOWTO, 2016.

[12] Mark Holland and Garth Gibson. Parity declustering for continuous operation in redundant disk arrays. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE, 1992.

[13] Thomas JE Schwarz, Jesse Steinberg, and Walter A Burkhard. Permutation development data layout (pddl). In *Proceedings Fifth International Symposium on High-Performance Computer Architecture*, pages 214–217. IEEE, 1999.

[14] Jehan-François Pâris, SJ Thomas JE Schwarz, Ahmed Amer, and Darrell DE Long. Highly reliable two-dimensional raid arrays for archival storage. In *2012 IEEE 31st International Performance Computing and Communications Conference (IPCCC)*, pages 324–331. IEEE, 2012.

[15] Asaf Cidon, Stephen Rumble, Ryan Stutsman, Sachin Katti, John Ousterhout, and Mendel Rosenblum. Copysets: Reducing the frequency of data loss in cloud storage. In *2013 {USENIX} Annual Technical Conference ({USENIX}{ATC} 13)*, pages 37–48, 2013.

[16] Block design. https://en.wikipedia.org/wiki/Block_design.

[17] H. Ke, H. S. Gunawi, D. Bonnie, N. DeBardeleben, M. Grosskopf, T. Grové, D. Manno, E. Moore, and B. Settlemyer. Extreme protection against data loss with single-overlap declustered parity. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 343–354, 2020.

[18] Raj Chandra Bose and Sharadchandra S Shrikhande. On the construction of sets of mutually orthogonal latin squares and the falsity of a conjecture of euler. *Transactions of the American Mathematical Society*, 95(2):191–209, 1960.

[19] Peter Braam and Dave Bonnie. Campaign storage.

[20] Leonardo Bautista-Gomez, Ana Gainaru, Swann Perarnau, Devesh Tiwari, Saurabh Gupta, Christian Engelmann, Franck Cappello, and Marc Snir. Reducing waste in extreme scale systems through introspective analysis. In *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 212–221. IEEE, 2016.

[21] Nezih Yigitbasi, Matthieu Gallet, Derrick Kondo, Alexandru Iosup, and Dick Epema. Analysis and modeling of time-correlated failures in large-scale distributed systems. In *2010 11th IEEE/ACM International Conference on Grid Computing*, pages 65–72. IEEE, 2010.

[22] Guillaume Aupy, Yves Robert, and Frédéric Vivien. Assuming failure independence: are we right to be wrong? In *2017 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 709–716. IEEE, 2017.

[23] Mary Baker, Mehul Shah, David SH Rosenthal, Mema Roussopoulos, Petros Maniatis, Thomas J Giuli, and Prashanth Bungale. A fresh look at the reliability of long-term digital storage. In *Proceedings of the 1st ACM SIGOPS/EuroSys European Conference on Computer Systems 2006*, pages 221–234, 2006.

[24] Mingqiang Li, Jiwu Shu, and Weimin Zheng. Grid codes: Strip-based erasure codes with high fault tolerance for storage systems. *ACM Transactions on Storage (TOS)*, 4(4):15, 2009.

[25] Avani Wildani, Thomas JE Schwarz, Ethan L Miller, and Darrell DE Long. Protecting against rare event failures in archival systems. In *2009 IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems*, pages 1–11. IEEE, 2009.

[26] Eitan Rosenfeld, Aviad Zuck, Nadav Amit, Michael Factor, and Dan Tsafrir. Raidp: replication with intra-disk parity. In *Proceedings of the Fifteenth European Conference on Computer Systems*, pages 1–17, 2020.

[27] Guangyan Zhang, Zican Huang, Xiaosong Ma, Songlin Yang, Zhufan Wang, and Weimin Zheng. Raid+: deterministic and balanced data distribution for large disk enclosures. In *16th {USENIX} Conference on File and Storage Technologies ({FAST} 18)*, pages 279–294, 2018.

[28] John Fragalla. Improving Lustre OST performance with ClusterStor GridRAID. In *2014 HPCAC Stanford HPC Exascale Conference*, 2014.

[29] Mingyuan Xia, Mohit Saxena, Mario Blaum, and David A. Pease. A tale of two erasure codes in HDFS. In *13th USENIX Conference on File and Storage Technologies (FAST 15)*, pages 213–226, Santa Clara, CA, February 2015. USENIX Association.