SAND2015-4740C

RESAR: Reliable Storage at Exabyte Scale

Thomas Schwarz, SJ*, Ahmed Amer †, Thomas Kroeger‡, Darrell Long§, Jehan-François Pâris¶
* Universidad Centroamericana, La Libertad, El Salvador, Email: tschwarz@uca.edu.sv
†Santa Clara University, Santa Clara, CA, Email: aamer@scu.edu
‡Sandia National Laboratories, Livermore, CA,[1] Email: tmkroeg@sandia.gov
§SSRC, University of California, Santa Cruz, CA, Email: darrell@soe.ucsc.edu
¶University of Houston, Houston, TX, Email: jfparis@uh.edu

***Abstract*—Large-scale disk-based storage systems need to protect the data stored in them against individual disk failures, common component failures and latent disk errors. We present RESAR, a layout scheme that provides two failure tolerance by only using XOR operations to calculate parity data. Our layout has the same write overhead as that of a disk array whose layout is based on virtual RAID Level 6 disk arrays. If the size of a reliability stripe is k, our write overhead is $2/k$. We show that RESAR is actually more resilient than the RAID Level 6 layout.**

I. Introduction

In the age of big data, magnetic disk based storage is expanding, but also slowly moving to a more archival role; magnetic hard drives are replaced by solid state disks and more data is stored but proportionally less of it is accessed. These trends imply a continuous increase in the capacity and economic life span of large disk based storage systems. In order to save energy, most disks in such a large storage system will be powered off. The large number of components has made failures a daily occurrence. The value of data will determine the level of failure protection, but most storage systems will do fine with tolerance of two simultaneous failures, if the failures can be repaired very quickly. The need for two-failure tolerance arises from the presence of latent sector errors that can only be detected by frequent scrubbing or by reading an affected sector, which all too often happens during a repair. A repair here is the reconstruction of data stored originally in a lost disk and moved to different, live disks and is separate from a replacement of the physical drive, which takes place at a later time or is subsumed by adding many more drives to the storage system.

Traffic in a large storage center is dominated by writes with reads to random locations. It is best therefore to organize the data in the form of one or several log and store the incoming data on a few disks that are currently powered on. The use of disks with shingled write recording is another reason to store data in form of a log.

We can achieve two-failures tolerance using erasure correcting codes such as Row-Diagonal Parity or Reed-Solomon codes, but we present here an alternative that is using a flat XOR-code. As usual, we decluster, that is, we store data in relatively small "disklets" (of between 10GB and 50GB with today's disk capacities) that can be read and written within minutes. This means recovery times in the order of minutes versus hours for a standard RAID Level 6. If disk failures are detected fast, this narrows the "window of vulnerability" that opens when a disk fails and subsequent failures can lead to dataloss.

Our family of layouts called RESAR places each *data disklets* (so called because it contains user data) into two reliability stripes and adds to each reliability stripe an additional *parity disklet* that contains the exclusive-or of the data in the data disklets disklets in the stripe.

By introducing a disklet layer into the organization of a disk array, we need a metadata server that manages the mapping of disklets to disks, but obtain the following advantages
(1) We can deal with very large disk arrays consisting of heterogeneous disks.
(2) Only the metadata server has to deal with relocating data in order to balance the load between already present disks and a batch of new disks.
(3) By making it simple to store data in a log, RESAR allows to minimize the number of disks currently powered-on for writing.
(4) RESAR allows to change effectively the size of reliability stripes, allowing to make dynamic trade-offs between parity overhead and the effort needed to recover from disk failures.
(5) RESAR recovery from disk failure is in the order of minutes (compared to hours for a system composed of classical RAID Level 6 layouts).
(6) RESAR allows to recover using two distinct set of disks and is thus much less likely to interfere with an important read operation.

In the remainder of the article, we explain the way to create RESAR layouts and assess RESAR's reliability by comparing to a similar disk array that is also declustered and organizes disklets in reliability stripes of k disklets and two additional parity disklets. The presence of the second parity disklet in the stripe then requires the use of a more sophisticated code (such as generalized Reed-Solomon with Galois field calculation, Even-Odd, or Row-Diagonal Parity). Because the stripes are built as in a RAID Level 6 layout, we refer to this organization in short-hand as a RAID Level 6, though it should be properly called a declustered RAID Level 6.

[1] Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

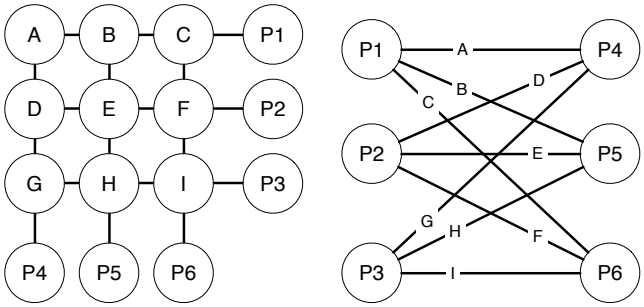

Fig. 1. Left: The two-dimensional layout for a disk array. A, B, ... I are data disks and $P1$, ..., $P6$ parity disks. Right: The corresponding graph visualization.

II. Two-Failure Tolerant Flat XOR Codes

Flat XOR codes are defined by Greenan *et alii* to be codes where parity symbols are calculated from certain subsets of data symbols [1]. We call these subsets reliability stripes. To each flat XOR-code corresponds directly a disk array layout where each data symbol corresponds to a data disk and each parity symbol to a parity disk.

Layouts based on flat XOR-codes and that tolerate two simultaneous failures need to place each data disk in at least two different reliability stripes. The intersection of two reliability stripes cannot contain more than a single disklet. We can *label* each data disk by the numbers of the two reliability stripes to which it belongs. Similarly, we can label each parity disk with the number of the reliability stripe to which it belongs. This defines a graph structure derived from the disk layout where each parity disk corresponds to a vertex and each data disk to an edge between vertices.

We use the graph to visualize the layout of disklets into reliability stripes. Figure 1 gives an example. In the graph layout, parity disklets corresponds to vertices and data disklets to edges. To our knowledge, the graph visualization was first exploited by Xu and colleagues in the definition of B-codes [4]. An observation by Zhou and colleagues characterizes minimal failure sets of disks as those containing either a cycle of edges or a path where the end vertices have also failed [5]. The graph visualization is a good way to determine the failure resilience of these type of layouts, as we did in previous work for the "complete layout" so-called because it derives from a complete graph. [3].

We investigate here a RESAR layout based on the bipartite graph, but allowing us to assign a parity layout for an arbitrary number of data disklets. The layout consists of two columns of parity disklets (the P and the D-disklets). Each P-parity disklet is "connected" (by a data disklet) to its opposite D-disklet and the next $k-1$ D-disklets. If we so desire, we can have the layout loop around so that all reliability stripes encompass exactly k data disklets.

We number all disklets in a RESAR layout by a pair of numbers (i, j) where $0 \leq j \leq k+1$ and i enumerates consecutively the pairs of P- and D-parities. The P-parity gets $i = 0$ and the D-parity gets $i = 1$. The data disklets are

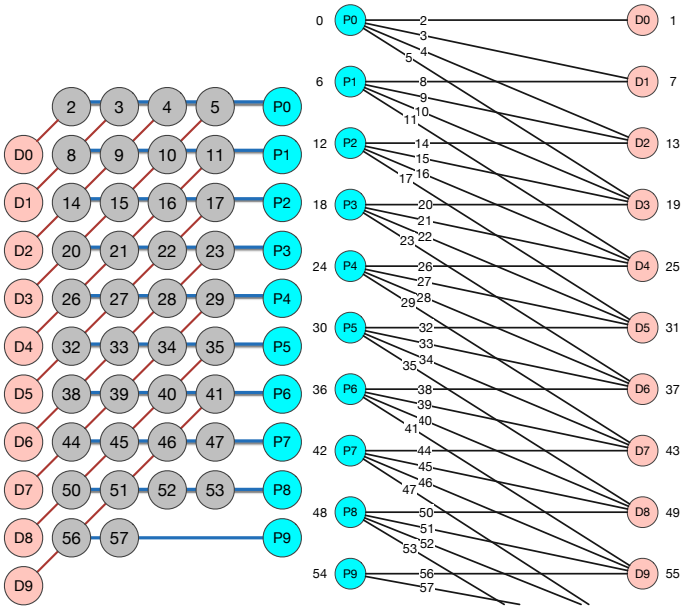

Fig. 2. A small bipartite RESAR layout (left) and its graph representation with $k = 3$. The disks are numbered for rack-aware layout (Section XXX)

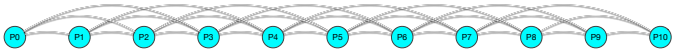

Fig. 3. A small complete RESAR layout in graph form.

numbered in order. We then convert the pair of numbers to a single one by the formula $(i, j) \rightarrow i * (k+2) + j$ that gives an enumeration of all disklets in the design such that neighboring disklets are assigned close indices. Since failure patterns with data loss are composed of neighboring disklets, we can use enumeration in order to insure that a single disk or rack failure cannot generate data loss. In our scheme, if the indices of two disklets is apart by more than $k^2 + k$, then they cannot be neighbors.

An alternative layout (the complete RESAR layout) is based on "stretching out" a complete graph and is presented in Figure 3. There, we have a single line of parity disklets (= graph vertices) and connect each disklet to the next k parity disklets down the line. Since each parity disklet also receives data disklets (= edges) from its left neighbors, each vertex (= parity disklet) in the graph is connected to $2k$ edges, which means that each reliability stripe is made up of $2k$ data disklets and the corresponding parity disklet. We show this layout only to make the point that RESAR defines families of disk array designs.

III. Recovery from Disk Failure in RESAR

To recover the data from a failed drive we must recover each of the disklets in the drive. Since each data disklet is in two reliability stripes (see Figure 4 for a depiction in the graph visualization), we can recover using either reliability stripe if all the other data disklet and the parity disklet in the corresponding stripe are still available. A parity disklet on a

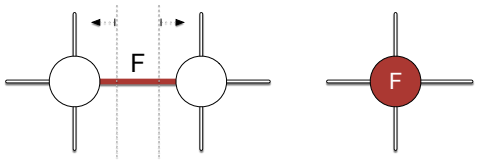

Fig. 4. Left: Failed data disklets with its neighbors. Right: Failed parity disklet with its neighbors.

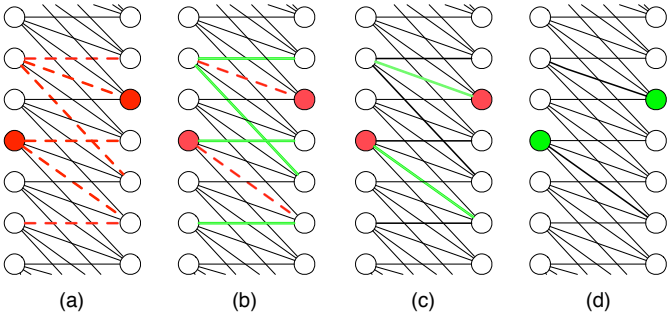

Fig. 5. Cascading recovery of multiple failures in RESAR.

failed disk can be reconstructed in another disk if all its data disklets can be recovered.

We can use the graph visualization to discuss recovery. The graph represents disklets, not disks, and a single disk failure results in multiple disklet failures. As we will discuss, a good disklet to disks mapping ensures that these disklet failures resulting from the failure of a single disk are widely spread over the (rather large) graph. We can recovery the data from a disklet (and then place the recovered disklet on another disk drive) if it is represented by an vertex (and is therefore a parity disklet) if all the edges (data disklets) adjacent to it are available. We can recover the data from a disklet represented by an edge (so this is a data disklet) if one of the adjacent vertices and all the edges adjacent to it are available. Cascading recovery in a RESAR layout

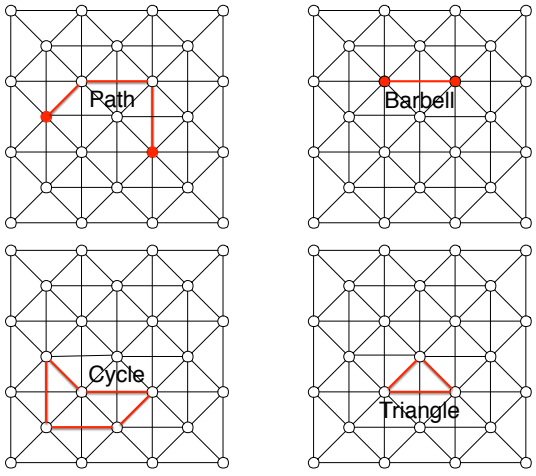

Fig. 6. Irreducible failure patterns.

happens if a disklet can only be recovered after some of its neighbors have been recovered. Figure 5 gives in column (a) a case with several failures in our graph layout. Failed elements are marked in red. In the upper failure cluster, there is one vertex on the left, which has lost three adjacent edges. For this reason, this vertex (or to be more precise, the reliability stripe represented by this vertex) cannot be used for recovery, but each data element in this stripe is also located in another reliability stripe. Two of the failed data elements can be recovered directly in the following step (column (b)), and the remaining failed element's data can be recovered in the third step (column (c)). The failed vertex in this cluster can only be reconstructed if all its adjacent edges are available. The lower failure cluster contains a failed vertex from which a path of length two of failed edges emanates. This pattern resolves itself only in the fourth step.

Arguments about failure tolerance are much easier in the graph than in the original primary design, as was previously observed [5]. Disk and sector failure induce a *failure pattern* in the graph. We are especially interested in patterns that represent data loss and that are minimal in the sense that removing one element of the pattern yields a pattern of failure from which we can recover. Any failure pattern that implies dataloss is or contains at least one minimal failure pattern. A key observation is that an edge, that is part of a minimal failure pattern, either has end-vertices that also have failed or an end-vertex where one other adjoining edge has also failed. This allows us to classify all minimal failure patterns. There are either a cycle consisting of failed edges or a path, which starts and ends at a failed vertex and otherwise consists of failed edges in between. The smallest minimal failure patterns are the *bar-bell* and the *triangle*. Figure 6 illustrates these concepts.

IV. RESAR Reliability

We now compare the reliability of a RESAR layout with that of a layout that organizes all disklets in the stripes of a RAID Level 6 layout with two parity disklets per stripe and k data disklets. The parity overhead in both organizations are equal (namely $2/k$). We also make some modeling assumptions that contradict the RESAR design goals of flexibility, but are necessary for calculations. In particular, we assume that each disk contains the same number L of disklets and that the disklets in the same position on the disk are organized in a separate RESAR layout where we connect the two ends in a single large circle. We make similar assumptions for the RAID Level 6 layout, namely that each reliability stripe contains disklets from the same relative location of the disk.

We first determine the number of failure patterns with n failed disklets out of N that cause dataloss in a single RESAR circle and in a RAID Level 6 layout, where in the latter all disklets form parts of a reliability stripe with k data and two parity disklets. We can use the notion of irreducible failure patterns (Figure 6) to determine the probability of dataloss for n failures, $n \leq 6$, before we run into an explosion of complexity in the derivation of the formulae and the formulae

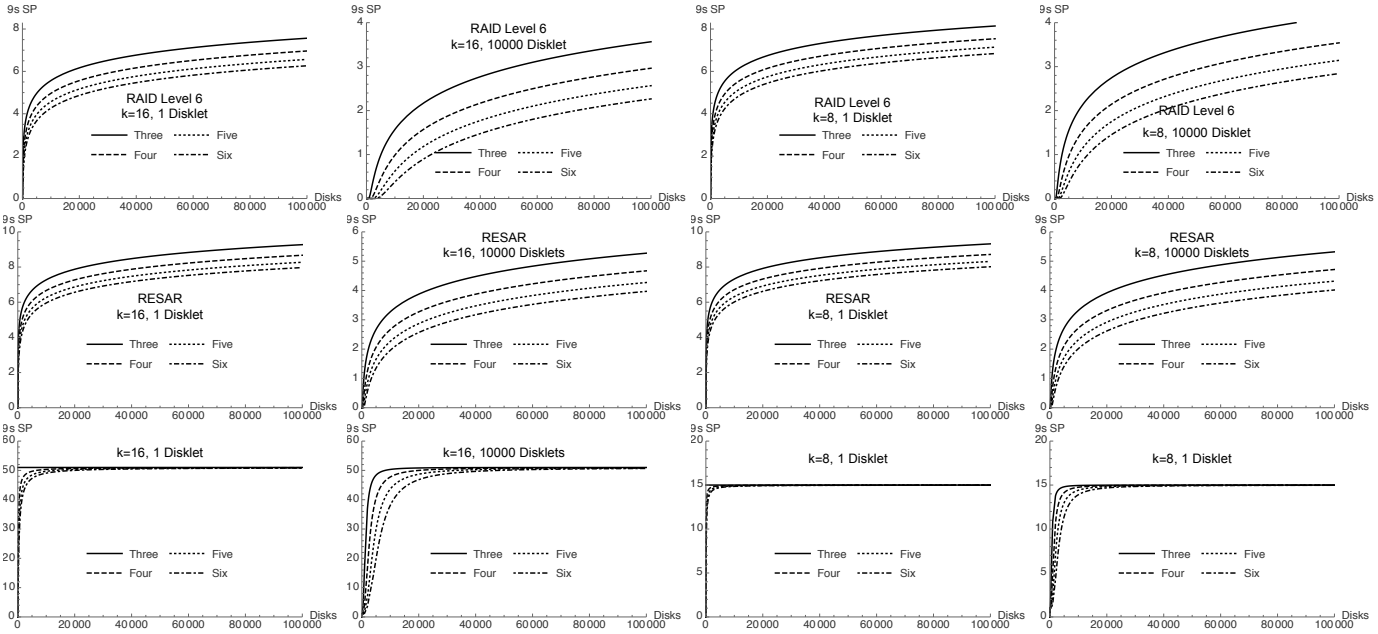

Fig. 7. Data survival probability in numbers of nine for the RAID Level 6 (top line) and the Resar layout and the increase in data survival probability by switching from the RAID Level 6 layout to the RESAR layout. Notice the different dimensions on the y-axis between the top and the middle row.

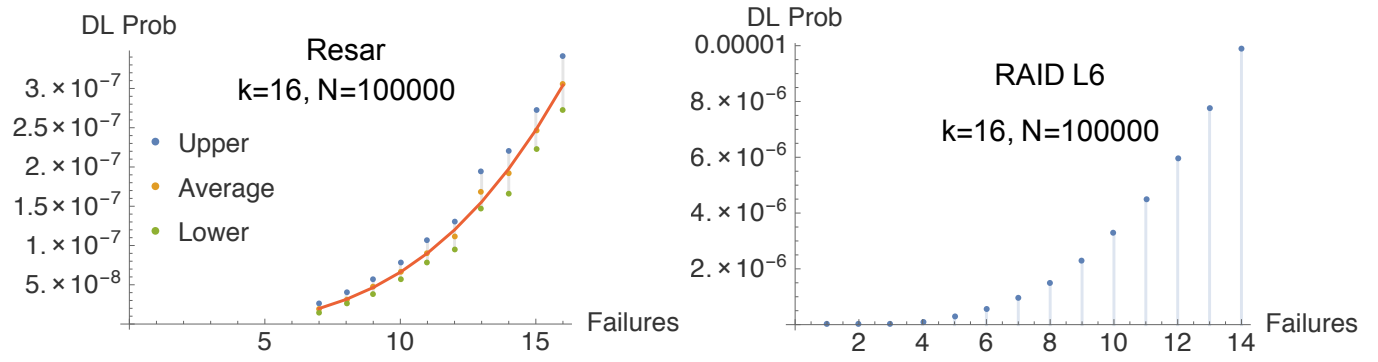

Fig. 8. Simulation results for the data loss probability of a RESAR layout (left) and exact values for a declustered RAID Level 6 layout (right), both with 100,000 disks. On the left, we give the average and the upper and lower value for the 99% confidence interval calculated with Wilson's formula.

itself. These numbers are actually quite useful for situations in which we have many disklets of few sites, since (a) recovery from dataloss is potentially so fast that the system is rarely ever in a state with more than 6 failed disks even if we ignore dataloss, and since (b) the probability of some dataloss with 7 failed disks is high (even though the amount of data lost is small). We give the results in Figure 7. As we can see, the RESAR layout is quite a bit more robust, for $k = 16$, the probability of dataloss is about 50 times higher and for $k = 8$, about 15 times higher.

Next, we simulated the resilience of a RESAR layout with 100,008 disklets (or disks and one disklet per disk). This simulation is very difficult and work-intensive (two weeks on 68 nodes) because failures are so rare. Accordingly, the results in Figure 8 (left) show great variation. We used a quadratic function for curve smoothing, giving in red.

The results show that RESAR with $k = 16$ is about fifty times less likely than an equivalent declustered RAID Level 6 layout to have suffered dataloss in the presence of the same number of failed disks. Figure 9 compares the dataloss probabilities for up to 500 failures and shows that RESAR remains considerably more robust than the declustered RAID, but at a slightly lower rate for larger numbers for failures.

Finally, Figure 10 shows the differences between the two organizations if we have many disklets per disk. As we can see, the effects are then much more pronounced. We used very small disklet sizes because even after using 60 nodes for several weeks, our simulation results have large confidence intervals that will be magnified if we calculate the robustness of layouts with many disklets per disk. $100,000$ disklets on a 10TB disk give a disklet size of 10MB, which is actually close to the block size on the new kinect drives from Seagate that use shingled write recording.

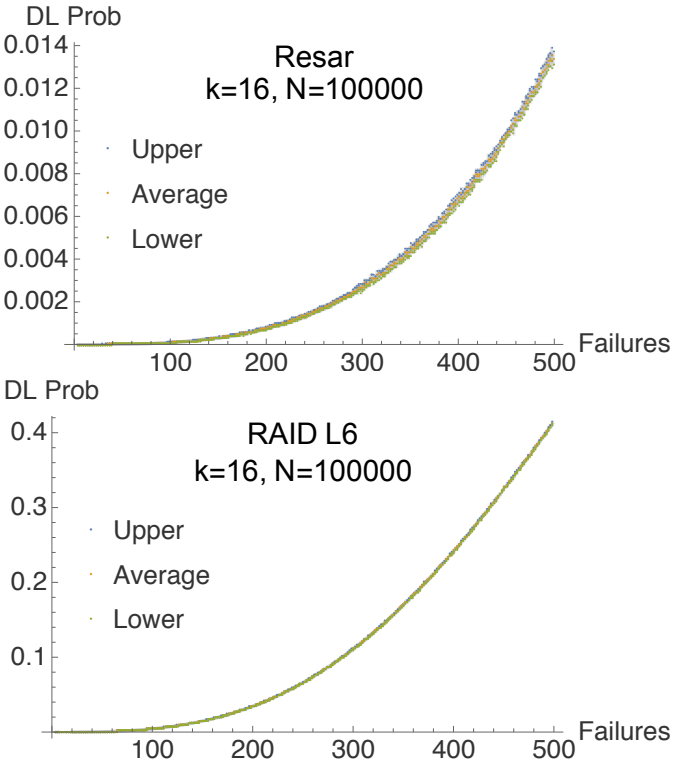

Fig. 9. Simulation results for the data loss probability of a RESAR layout (top) and a RAID Level 6 layout (bottom) with 100,000 disks. We give the average and the upper and lower value for the 99% confidence interval.

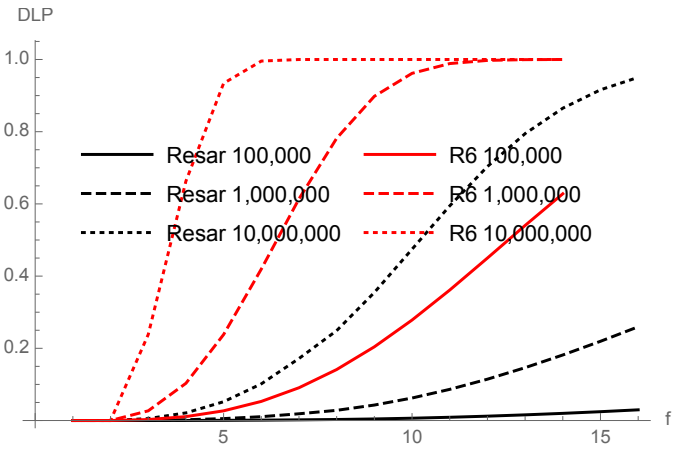

Fig. 10. Dataloss probabilities for a system with various numbers of disklets per disk.

V. Performance

RESAR is optimized for writes to consecutive data disklets. The need arises because we want to organize writes such that we write all data disklets in a reliability stripe so that we can update the parity disklet directly.

In the RAID Level 6 layout, there is only a single reliability stripe and we need to store the contents of k data disklets "in flux" before we can calculate the parities. Alternatively, we can maintain a buffer each for the two parity disklets and maintain

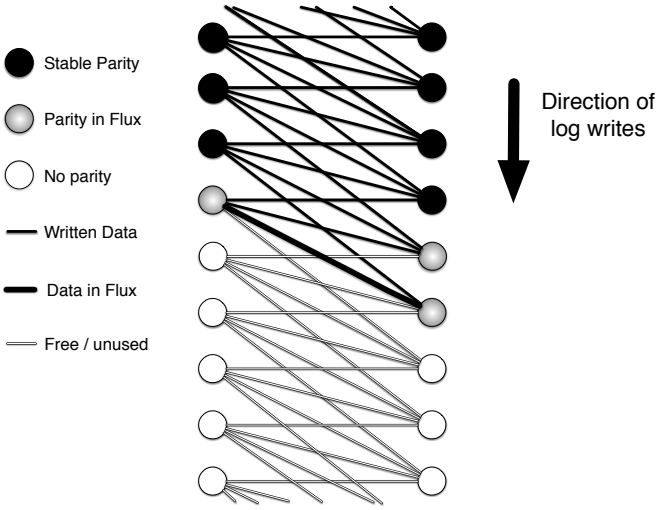

Fig. 11. Log write operation in a RESAR layout (with $k = 4$).

in it the parity of the data disklets in the stripe already written.

If we are writing a data disklet in a flat XOR-code, we can then write another data disklet in the same reliability stripe, but this leaves $k-1$ data disklets in the other reliability stripe, in which the original data disklets was, unwritten. As Figure 11 shows, the bipartite RESAR layout actually deals quite well with this problem. We write data disklets in order of their indices. A parity disklet is in flux, if some, but not all the adjoining data disklets is in flux. In Figure 11, there are three parity disklets in flux, one P-disklet and two D-disklets (the gray vertices on the right). If we write the next data disklets, one more D disklet would be in flux, but if we then move on to the next data disklet, one D disklet can be closed and no additional one enters a state of flux.

Since we are writing data disklets one at a time, only one data disklet (per log) can ever be in flux. Unlike for the RAID Level 6 layout, more than two parity disklets are in flux, namely a maximum of $k+1$.

Dealing with a large number of parity disklets in flux is in fact not difficult. The simplest mode is to store copies of data disklets contents in other disks until all data disklets in a reliability stripe have been written. At this moment, we can calculate the parity data, place it in a parity disklet and then free the disklets with the content copies. We do not need to keep the disks with the data disklets replica powered on, but only need to power them on if they are needed, first to store the replicated data, and then in order to be read for the parity calculation.

The difference between the two data layouts (RESAR and RAID Level 6) then only lies in the additional number of disklets to be stored, which is very small compared to the total capacity of disklets and costs less than the price of a single hard drive. The RAID Level 6 layout uses up to $2k$ disklets for temporary copies of data, while the RESAR layout would use up to $k(k+1)$ temporary disklets.

Most of the time, we would write to several data disklets

in parallel in order to digest the amount of incoming data. The small advantage of the RAID Level 6 layout due to its compactness would then be even smaller.

VI. Conclusions

In this paper, we have evaluated a new layout for very large disk arrays called RESAR. This layout places every disklet into two reliability groups with one parity disklet each. It has been designed to avoid small RAID writes (where we obtain the new parity from the delta of the new and the old data and the old parity). We compared it with a layout organized as a collection of RAID Level 6 reliability stripes and shown the RESAR layout to be much more resilient.

In the age of fast Galois field calculations [2], the fact that RESAR only uses exclusive-or operations to calculate parity might or might not be an advantage, but it frees us from the need to use powerful microprocessors.

We are well aware that many instances of data loss in disk arrays are due to faulty disk array controller software and that the reliability of the best layout can be destroyed for example by a faulty failure mechanism. The choice of the disk layout should not affect the complexities of implementing disk arrays.

References

[1] K. M. Greenan, X. Li, and J. J. Wylie, "Flat XOR-based erasure codes in storage systems: Constructions, efficient recovery, and tradeoffs," in *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*. IEEE, 2010, pp. 1–14.

[2] J. S. Plank, K. M. Greenan, and E. L. Miller, "Screaming fast Galois field arithmetic using Intel SIMD instructions," in *FAST*, 2013, pp. 299–306.

[3] T. Schwarz, D. D. Long, and J. F. Pâris, "Reliability of disk arrays with double parity," in *Dependable Computing (PRDC), 2013 IEEE 19th Pacific Rim International Symposium on*. IEEE, 2013, pp. 108–117.

[4] L. Xu, V. Bohossian, J. Bruck, and D. G. Wagner, "Low-density MDS codes and factors of complete graphs," *Information Theory, IEEE Transactions on*, vol. 45, no. 6, pp. 1817–1826, 1999.

[5] J. Zhou, G. Wang, X. Liu, and J. Liu, "The study of graph decompositions and placement of parity and data to tolerate two failures in disk arrays: Conditions and existance," *Chinese Journal of Computers (chinese edition)*, vol. 26, no. 10, pp. 1379–1386, 2003.