Adjustable flat layouts for Two-Failure Tolerant Storage Systems

Thomas Schwarz, SJ
Department of Mathematics, Statistics, and Computer Science
Marquette University
Milwaukee, Wisconsin
thomas.schwarz@marquette.edu

***Abstract*—Systems suffer component failure at sometimes unpredictable rates. Storage systems are no exception; they add redundancy in order to deal with various types of failures. The additional storage constitutes an important capital and operational cost and needs to be dimensioned appropriately. Unfortunately, storage device failure rates are difficult to predict and change over the lifetime of the system.**

Large disk-based storage centers provide protection against failure at the level of objects. However, this abstraction makes it difficult to adjust to a batch of devices that fail at a higher than anticipated rate. We propose here a solution that uses large pods of storage devices of the same kind, but that can re-organize in response to an increased number of failures of components seen elsewhere in the system or to an anticipated higher failure rate such as infant mortality or end-of-life fragility.

Here, I present ways of organizing user data and parity data that allow us to move from three-failure tolerance to two-tolerance and back. A storage system using disk drives that might be suffering from infant mortality can switch from an initially three-failure-tolerant layout to a two-failure-tolerant one when disks have been burnt in. It gains capacity by shedding failure tolerance that have become unnecessary. A storage system using Flash can sacrifice capacity for reliability as its components have undergone many write-erase cycles and thereby become less reliable.

Adjustable reliability is easy to achieve using a standard layout based on RAID Level 6 stripes where it is easy to convert components containing user data to ones containing parity data. Here, we present layouts that unlike the RAID layout use only exclusive-or operations, and do not depend on sophisticated, but power-hungry processors. There main advantage is a noticeable increase in reliability over RAID Level 6.

***Index Terms*—Disk Array Layout, Failure Tolerance**

I. Introduction

Any storage system stores data in fallible devices. Currently, these are mainly magnetic disk drives or flash memory, and in the near future, phase change memories. To protect against device failures and other types of data unavailability, data is stored redundantly. Since replicating data is expensive, many systems use erasure coding that adds parity data to a group of user data and that allows us to calculate some inaccessible parts of user data from the remaining user data and the parity data. We could arrange all devices with user data in a single reliability stripe and then add k parity devices in order to achieve tolerance against k failures, but since parity data needs to be updated with every change of user data in the stripe, such an arrangement would easily overwhelm the parity devices with updates. Additionally, in case of failure, reconstructing inaccessible data requires reading the same number of devices as there were those containing user data, which would be all of them in this scenario. The length of the reliability stripes are therefore chosen much smaller than the number of devices in the ensemble. The RAID Level 6 organization for instance arranges data in stripes each consisting of n user data devices and an additional two parity devices.

The hazard rate a.k.a. the failure rate (the probability to not survive the next time period) for many types of devices, including disk drives and flash, changes over their lifetime. We present here data layouts that adjust their level of erasure protection to these failure rates. The total number of storage devices does not change, which allows adjustment in prepackaged storage such as a large disk array or flash array with a fixed number of slots. This is easy to do if one chooses some form of RAID Level 6 design. We present here an organization that is based on a more resilient flat XOR layout.

Write cycles in Flash drives degrade their reliability. The resulting high error rate is controlled with Error Correcting Codes (ECC) that add parity data to each Flash page. Errors not controlled in this manner result in lost data. Meza and colleagues discern distinct rates of SSD failures during distinct periods of their useful life. Apparently, quite a number of devices fail early, and those that do not fail early, fail only towards the end of their useful life. [13].

The emergence of very large data centers with associated storage systems has allowed several studies on the failure behavior of disk drives. One result is that disks in these large installations fail at higher rates than the data sheet specifications suggest. For maybe 50% of batches, disk failure rates follow a bathtub curve – high failure rates at the beginning (burn-in phase) and the end of the economic lifespan (wear-out) and a lower rate in between [2], [18], [19], [20].

A recent study by Beach showed that individual disk batches can evidence much higher failure rates than usual, even considering that disk failure rates in data centers already tend to be higher than the specification sheets suggest. Beach reports a batch of 1163 Seagate Barracuda 7200.14 disks that failed at a rate of 43% per year in 2014, whereas other disks from the same manufacturer perform much better. Beach's observation are taken from Backblaze that buys the cheapest disks available and stores them in bundles of 45 in a Backblaze Storage Pod [3], [11]. The statistical significance of the Backblaze data can be doubted, but even if they are incapable of measuring the robustness of a specific drive family, they make the point that disk drive reliability differs unpredictably between batches and different time periods.

Data centers sometimes literally expand by adding a con-

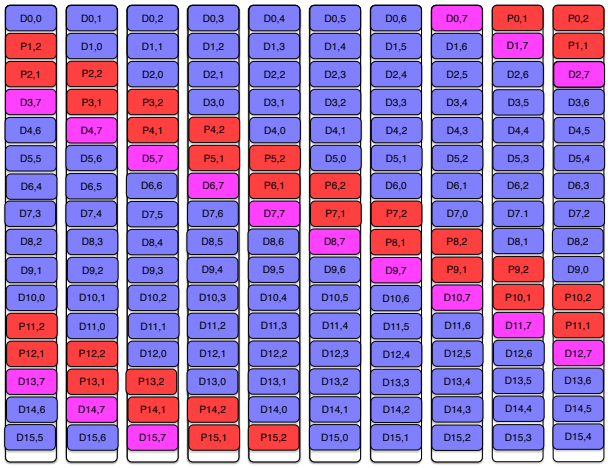

Fig. 1. A RAID 6 adjustable reliability stripe.

tainer with hardware to it. An organization usually does not buy containers and disks independently but rather an ensemble of enclosures and disks. If flash memory is used, then the diverse chips will also be packaged in an enclosure. Unless we are willing to organize reliability stripes across enclosures and disk arrays, we need to organize reliability with a fixed number of components. This is easy to do if we use RAID Level 6. Of course, very large disk farms do exactly that: they organize reliability across several racks and even storage facilities. The data is organized using sophisticated erasure correcting codes such as Local Reconstruction Codes [7], [28] or Partial MDS Codes [4], [5] in a manner that allows data reconstruction usually involving only disks from the same rack but that are still capable of dealing with larger failures using disks in other racks or even remote data centers.

These sophisticated layouts use individual disks as the management unit. I propose instead to use storage pods with even a few hundred disks as the unit of management. Each storage pod consists of a set of data and parity disks together with a few spare disks and is managed on an individual level. To obtain security against common cause failure (ranging from catastrophe to seemingly trivial causes such as a loose enclosure that allows vibration from one bad disk to affect others in the same enclosure) as well as temporary data unavailability such as one caused by a network problem at a data center, we can arrange the storage pods in reliability stripes as well, using a simple parity code, a linear MDS code, or a more sophisticated partial MDS code. The storage pods appear as highly reliable and very large storage units. They are responsible for trading capacity for reliability in case that the underlying disks are less reliable than expected or in order to combat infant mortality.

Since our construction applies to devices made of Flash drives or Phase Change Memories as well as disk drives, we call our organizations *s*torage arrays. We are proposing and evaluating the internal organization of storage arrays and their individual reliability, not how they are used in large scale distributed storage systems. A smaller organization might for legal reasons prefer to satisfy its data needs with a single array. A PCM storage manufacturer might decide to use the layout proposed here for a single very high capacity storage solution that gracefully react to the aging of devices by lowering its storage capacity.

Such a storage array (or pod) stores data internally using a number of storage devices such as disks or flash chips and protects again device failure in the usual manner by creating redundant parity data. It therefore organizes the set of devices in several reliability stripes. Each stripe contains a fixed number n of disks (or flash chips). Data is stored in groups of n buckets (or disklets), one on each physical disk. Each group consists of $n-k$ data buckets and k parity buckets. We simply adjust the resilience of a stripe and thereby the reliability of the storage array by rededicating one bucket from a data bucket to a parity bucket or vice versa. Figure 1 gives a simple example for a RAID 6 reliability stripe consisting of 10 disks or flash chips. In the two-failure tolerant layout, the stripe stores data in buckets Di, j and parity in buckets Pi, j. If a higher resilience is needed, then the data buckets $Di, 7$ becomes a parity bucket. The contents of the first parity bucket $Pi, 1$ in each stripe i is calculated using the exclusive-or operation, but the parity in the other parity buckets is calculated in a more involved way by using a linear erasure correcting code. Switching from two-failure tolerance to three-failure tolerance or back is an involved procedure. In the first case, data needs to be moved from the reassigned (violet) data buckets as the array is loosing capacity and the parity data needs to be recalculated. In the latter case, the parity bucket is conceptually emptied. But when data is written to these violet buckets, then the corresponding parity blocks (depicted in red) have to be rewritten with recalculated parity data. However, as we only envision one or maybe two changes of failure tolerance over the life-span of the array, the cost of the procedure is amortized to almost nothing over the economic life-span of the storage array.

Instead of using the RAID 6 architecture, we propose to use layouts based on *Flat XOR-codes* [6] because of their greater robustness than those based on RAID Level 6. The flat layouts also do away with the need for Galois field calculation and the need for power-hungry, sophisticated processors that can encode and decode general linear erasure correcting codes at access speeds [17]. These flat layouts have the same parameters as a corresponding RAID Level 6 layout, that is, they have the same storage overhead, the same amount of reconstruction traffic after a failure, and the same number of parity updates. Their biggest drawback is that they only exists for certain configurations (Table I below). Because each storage device belongs to two otherwise disjoint reliability stripes, reconstructing the contents of a failed device can be done in two different ways, implying greater flexibility in the handling of reconstruction traffic. To deal with the phenomenon of unexpectedly high failure rates, we proposed earlier "hardening" a disk array by adding additional parity disks on demand [16]. *Here, we propose to achieve hardening by merely reconfiguring the data layout.* Reconfiguration

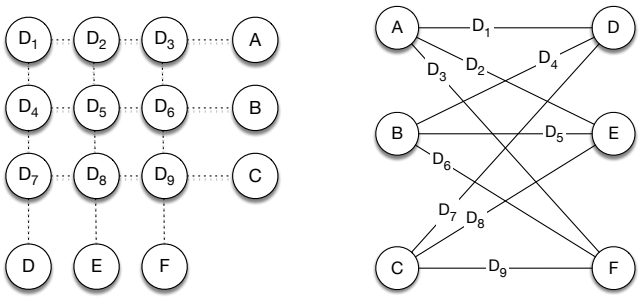

Fig. 2. Left: The two-dimensional layout for a storage array. D_1, D_2, . . ., D_9 are data devices and A, B, . . ., F parity devices. Right: The corresponding graph visualization.

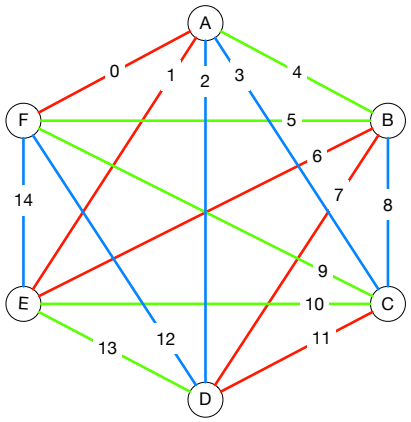

Fig. 3. The complete graph K_6.

instead of hardening is called for when storage needs are provided by ensembles with a fixed number of devices.

In the rest of this paper, we describe these layouts, calculate their reliability, and then compare with the reliability of RAID based layouts. In the final section, we point out that there are still other flat layouts that can be used for more reliable, but still adaptable layouts.

II. Layouts

Our storage arrays are laid out using *flat XOR-codes* [6]. This means that all devices are either parity or data devices and are organized into *reliability stripes* that consist of k data and one additional parity device. A two-failure tolerant layout places each device into (at least) two different reliability stripes. Reversely, if each data device is placed in two different reliability stripes, then the layout is two failure tolerant if reliability stripes are either disjoint or intersect in exactly one data disk [16], [23].

A. A Graph Representation

This frequently made observation implies that each data device is uniquely characterized by the two reliability stripes in which it is located. This allows us to represent a layout as a graph. The vertices of this graph are the reliability stripes and the edges of the graph are the data devices. Since each reliability stripe contains exactly one parity device, we can identify the vertices with parity devices as well. To our best knowledge, this representation of flat XOR-codes with two-failure tolerance as a graph was first used by Xu and colleagues in the definition of B-codes [27].

Figure 2 shows on the left a typical two-dimensional layout consisting of the nine data disks D_1, D_2, . . ., D_9 and six parity disks A, B, . . . , F. The dotted lines indicate the reliability stripes. Thus, the contents of B are the exclusive-or of the data disks D_4, D_5, and D_6. On the right, Figure 2 shows the corresponding graph. For example, the edge labeled D_8 on the right of Figure 2 connects vertices C and E and corresponds therefore to data disk D_8 on the left of the Figure located in the third row (with C) and second column (with E). As we can see, each data disk is determined by a pair of parity disks but not every pair of parity disks has an associated data disk.

We present a different example in Figure 3, where we start with the complete graph with six vertices. The colors of the edges are from the Lawless factorization that we will use below. The data devices correspond to the edges and the parity devices to the vertices. We can therefore read off the six reliability stripes. We present them as lines starting with the parity device and followed by the user data carrying devices in the left-to-right order of the edges at the vertex:

A	0	1	2	3	4
B	4	5	6	7	8
C	8	3	9	10	11
D	11	7	2	12	13
E	13	10	6	1	14
F	14	12	9	5	0

The construction guarantees that all data disks belong to exactly two reliability stripes and that each reliability stripe contains exactly five data devices. Two failure tolerance is also a consequence of the construction, but that is not as obvious. We can argue by case distinctions. If the two failed devices are parity, then their contents can be reconstructed from the user data that has survived in its entirety. If one is a parity and the other one a data device, then we can reconstruct the data in data device from one of the two reliability stripes in which the data device is located. Finally, if both are data devices, then they can only be in the same reliability stripe once and we can use the respective other stripe to reconstruct the data on the lost two devices.

B. Amended Layouts

Disk reliability can be much lower than advertised. Many, but far from all disk families investigated show high infant mortality. Layouts whose data survive three simultaneous failures are therefore desirable. In previous work [23] we advocated using a complete graph to define a two-failure tolerant layout and then divide the data disks into additional reliability stripes, also with k data disks. In the graph, the additional reliability stripe are an *edge factoring* in the sense of graph theory. We are using Lawless' construction of such a factoring from 1974 that results in triple-failure tolerant layouts [12].

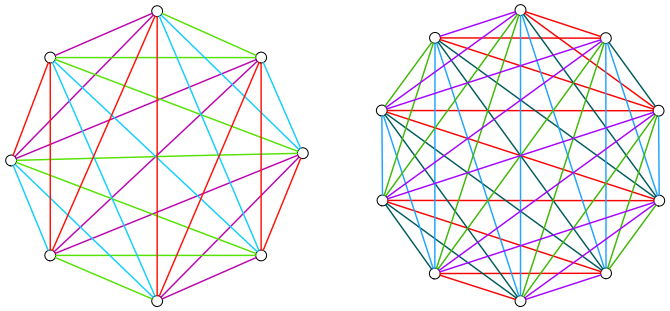

Fig. 4. Lawless factorizations of the complete graph for K_8 (left) and K_{10} (right).

TABLE I
DIMENSIONS OF PUNCTURED LAYOUTS. ON THE LEFT, WE GIVE THE NUMBERS FOR THE TWO-FAILURE TOLERANT AND ON THE RIGHT FOR THE THREE-FAILURE TOLERANT LAYOUT.

d	# Data	# Parity	# Total Disks	Stripe Sizes
3	15/12	6/9	21	5/4
4	28/24	8/12	36	7/6
5	45/40	10/15	55	9/8
6	66/60	12/18	78	11/10
7	91/84	14/21	105	13/12
8	120/112	16/24	136	15/14
9	153/144	18/27	171	17/16
10	190/180	20/30	210	19/18
11	231/220	22/33	253	21/20

Lawless designed the factorization in the context of design theory, and in particular, in order to give a "handcuffed design", a generalization of Kirkman's famous schoolgirl problem from 1847 that stands at the beginning of the theory of Combinatorial Designs [10]. A factor in the construction is a path that starts at one vertex, then moves to the vertex to the right, then two to the left, then three to the right, etc. and creating a zig-zag pattern. This factorization only exists if the number of vertices is even. The number of factors is half the number of vertices and the length of the factor is one less than the number of vertices. We give factorizations of K_8 and K_{10} in Figure 4, where factors are colored with the same color.

To change the resilience of a layout from two-failure tolerance to three-failure tolerance, we add n additional parity disks and create a new reliability stripe by grouping all data disks corresponding to edges in the same factor together with one of the parity disks. We called this the "amended layout". It is the basis for the current work.

C. Punctured Layouts

Adding and removing parity devices from an ensemble is possible, but not convenient. We present now a flat layout that switches from two-failure to three-failure tolerance and back without adding or removing devices. We start with an amended layout by coloring the edges of a complete graph K_{2d} with the design by Lawless. The edges colored with a given color form a path starting at a node i and then moving to nodes $i+1 \pmod{2d}$, $i-2 \pmod{2d}$, $i+3 \pmod{2d}$, $i-4 \pmod{2d}$, ..., Figure 5. Since all vertices are visited and since the number of vertices is even, the number of edges in the path is odd. Therefore, there is a middle edge in the path, the d^{th} one, which goes from $i-d+1 \pmod{2d}$ to $i+d \pmod{2d}$ if d is odd and which goes from $i+d-1 \pmod{2d}$ to $i-d \pmod{2d}$ if d is even. If we arrange the nodes in a cycle in ascending or descending order, then these edges are the ones that pass through the center of the cycle.

The two-failure tolerant layout is just defined by K_{2d} and disregards the colors. Thus, there are $2d$ reliability stripes, each with $2d-1$ data disks and – of course – one parity disk. To switch to the three-failure tolerant layout, we convert the data disks corresponding to the middle edges in the Lawless paths, indicated by dotted lines in Figure 5 to parity disks. We then create d additional parity stripes from the data disks colored by the same color in the Lawless coloring. Of course the middle edges in each path are no longer available as data disks, since they are now parity disks. The design now has $3d$ reliability stripes, each with $2d-2$ data disks and a parity disk.

We still need to show that the new design is three failure tolerant. Therefore assume that three devices have failed. If one or more of the failed devices is one of the d converted parity devices, then we really have at most a two-device failure in the non-amended layout, and we know that we can recover from that. The remaining case is that of three devices in the non-amended layout. There are exactly two failure patterns. The first one consists of a data device and the two parity devices belonging to the same reliability stripe as the data device. In this case, we can reconstruct the data on the data disk using the new reliability stripe containing the data device and write it to a spare disk. The second failure pattern consists of three failed data devices such that each pair of two of the three failed devices share a reliability stripe. These corresponds to three edges that form a triangle in the graph. The edges colored with the same color form a path and cannot therefore contain a triangle. This means that the edges in a triangle have two or three different colors. Therefore at least one of the edges has a different color than the other two in the triangle and since these are the only failed devices, we can use the new reliability stripe corresponding to its color to reconstruct the data on a spare drive. Now there are only two failures and we can deal with them using the old reliability stripes. This concludes the proof that the new layout provides three-failure tolerance.

Switching between two- and three-failure tolerance without introducing additional disks into the ensemble restricts the dimension of any disk array. However, the dimensions of our layouts cover a quite reasonable range, as can be seen from Table I.

III. RELIABILITY CALCULATION

Attempts to count patterns of failed devices for our 2-failure and 3-failure tolerant flat XOR layouts quickly run into too many inclusions and exclusions and become very difficult to calculate. Instead, we used simulation. In batches of 100,000

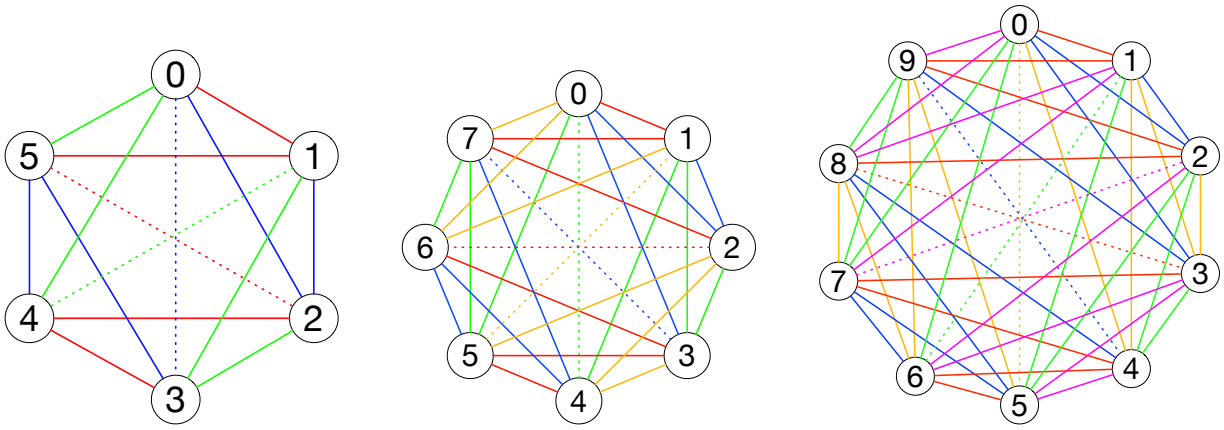

Fig. 5. Punctured layout of K_6, K_8, and K_{10}.

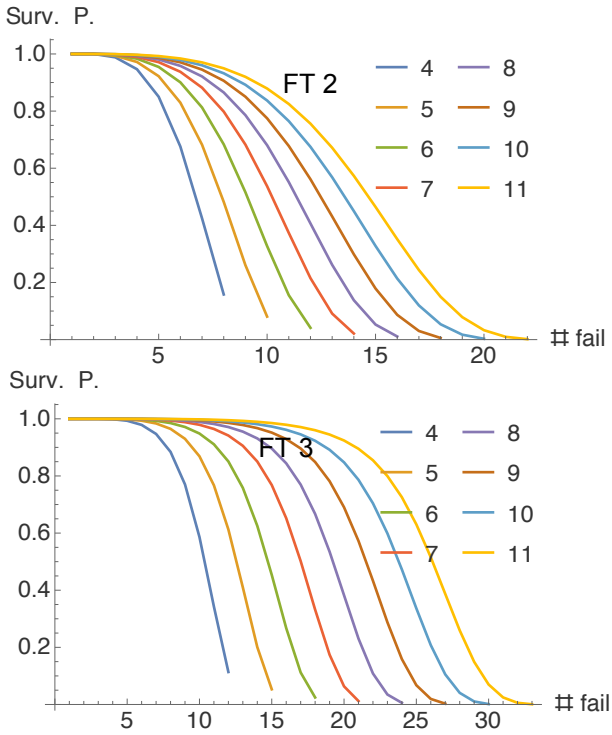

Fig. 6. Survival Rates of two-failure (top) and three-failure (bottom) tolerant layouts. The x-axis gives the number of failed devices, the y-axis the probability that the layout has not lost data.

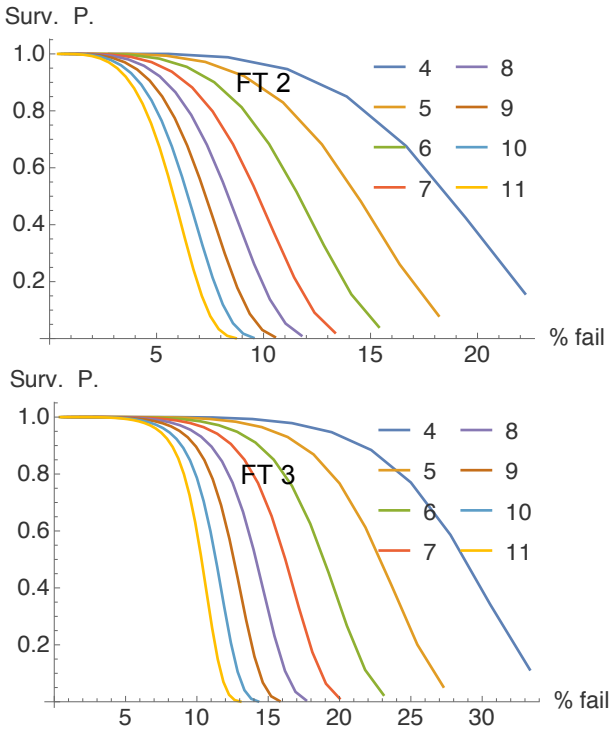

Fig. 7. Relative survival Rates of two-failure (top) and three-failure (bottom) tolerant layouts. The x-axis gives the percentage of failed devices, the y-axis the probability that the layout has not lost data.

or 1,000,000 runs, each run simulates the failure of f storage components in one of our layouts. For each batch, we count the number c of times per batch where we would not have been able to recover all data. We did this for at least 200 batches. We could assume that the value c is normally distributed. Statistical tests revealed this to be indeed the case with the exception of very small f for large layouts. Nevertheless, we assumed the Student's t distribution for c for the calculation of 99% confidence intervals. The average value of c divided by the number of runs in a batch estimates the robustness – the probability that a system with f failed devices has lost data. By adding batches if necessary the radius of the confidence intervals was made to be less than $1/10000$. We judged this precision to be necessary for trustworthiness as we calculate the data survival rate during five years for each layout.

We display the results of our robustness determining simulations in Figures 6 and 7. There are no error bars because we made the confidence intervals so small. The x-axis gives the number of failed devices in Figure 6 and the percentage of failed devices in Figure 7. The curves are labeled with the d-value in Table 1, which is half the number of vertices. The y-axis gives the probability of data survival. The typical S-

curves in Figure 6 show that data survival probability increases with increased size, whereas the ones in Figure 7 show the reversed order. This is not surprising since the size of the reliability stripes increases with the degree. The relative increase in robustness between the two-failure and the three-failure tolerant layouts are quite apparent.

We then used a standard continuous Markov model to calculate survival probability of a system after four and five years. We did not assume declustering. This means that all storage components are assigned a fixed role, either as data or as parity.

A. Markov Model

In order to avoid a plethora of states, our Markov models have one state for each number of failed devices in the ensemble. In addition, we have the failure state. We consider every dataloss catastrophic, so that we made the failure state self-absorbing.

There are two types of state transitions in our models, those modeling repair and those modeling failure. The rate of failure transitions is given by $n\lambda$ where λ is the failure rate and n is the number of currently functioning devices in the ensemble. We determine the destination of the transition – either to the next state or to the failure state – based on the previously determined robustness. We now derive these values.

Let A_i stand for the event that the ensemble has not lost data in the presence of exactly i failures and let $F_i = 1 - A_i$ stand for the opposite event that the device has suffered data loss in the presence of exactly i failures. The robustness is the probability $P(A_i)$. Clearly, for all $i \in \mathbb{N}$

$$A_i \supset A_{i+1}$$

and

$$F_i \subset F_{i+1},$$

as additional failure cannot restore an ensemble with dataloss to one without. Also, $P(A_0) = 1$ and $P(F_i) = 0$ for large enough i. The rate at which a failure transition from State i leads to the absorbing failure state is the conditional probability of dataloss in State $i+1$ given that there was no dataloss in State i and is

$$\begin{aligned} P(F_{i+1}|A_i) &= \frac{P(F_{i+1} \cap A_i)}{P(A_i)} \\ &= \frac{P(F_{i+1}) - P(F_i)}{1 - P(F_i)}. \end{aligned}$$

Correspondingly, if the total number of devices in the ensemble is denoted by N, then the transition from State i to State $i+1$ is taken at a rate of

$$\tau_{i,i+1} = (1 - \frac{P(F_{i+1}) - P(F_i)}{1 - P(F_i)})(N - i)\lambda$$

and from State i to the absorbing failure state at a rate of

$$\tau_{i,f} = \frac{P(F_{i+1}) - P(F_i)}{1 - P(F_i)}(N - i)\lambda.$$

There are repair transitions from State i to State $i-1$ taken at a rate of $i\rho$.

B. Implementation of Survival Rate Calculation

We used the Euler method to update the stay probabilities of the states every second. Discretizing updates is one source of numerical error, the other one is rounding error. We used the Apfloat (arbitrary precision) Java library experimenting with various degrees of precision until settling on one where there were no longer changes in the results.

C. Results

We display the results of our calculations in Figure 8. The x-axis gives the mean time to failure of the disks in hours, whereas the y-axis gives the number of nines in the survival probability after 5 or 6 years, respectively. The number of nines is defined as $-\log_1 0(1 - p_s)$ for a survival probability of p_s. For example, the number of nines of $p_s = 0.999$ is 3 and of $p_s = 0.9995$ is 3.30103.

For a given mean time to failure of the disks, the survival rate of the three-failure tolerant ensemble is better by about four nines. More importantly, if disks fail at a much higher rate, then the three-failure tolerant layout has an equivalent rate of survival than the two-failure tolerant layout. In the graphs, the difference between x-axis values for the same survival level is more important.

IV. Comparison with RAID Levels 6 and 6+1

We compare with RAID levels 6 and 6+1 in order to put our reliability numbers in context. A RAID level 6 is organized as n reliability stripes, each of which contains m data devices and an additional two parity devices. The first parity device contains the exclusive-or parity of the m data devices, whereas the contents of the second one are calculated via Galois field arithmetic. Using the Packed Shuffle Bytes (PSHUFB) instruction in the INTEL architecture's Supplemental Streaming SIMD Extensions 3 or its equivalent on other processors, Plank, Greenan, and Miller showed how this so-called Q-parity can be calculated at speeds faster than the speed of streaming data from a disk [17]. The drawback is of course the need for sophisticated, but power-hungry processors. For lack of a better name, we call RAID level 6+1 a similar design with m data disks and three parity disks per reliability stripe. By flipping one of the data disks to a parity disk, we change a level 6 RAID to a level 6+1 RAID and vice versa. For the convenience of the reader, we show the definition of the parameters in Figure 9.

It is normal to divide the blocks of a disk in a RAID level 6 or 6+1 into regions and cyclically change the assignments of each disk to the role of data or parity disk so that each disk contains the same number of blocks that serve as data blocks or parity blocks. This strategy equalizes read and write loads. It however has no influence on the reliability of the ensemble.

This is not true if we use declustering, where the assignment of a region in a disk also varies the reliability stripes. Declustering equalizes the reconstruction load over the remaining disks after one or more disk failures.

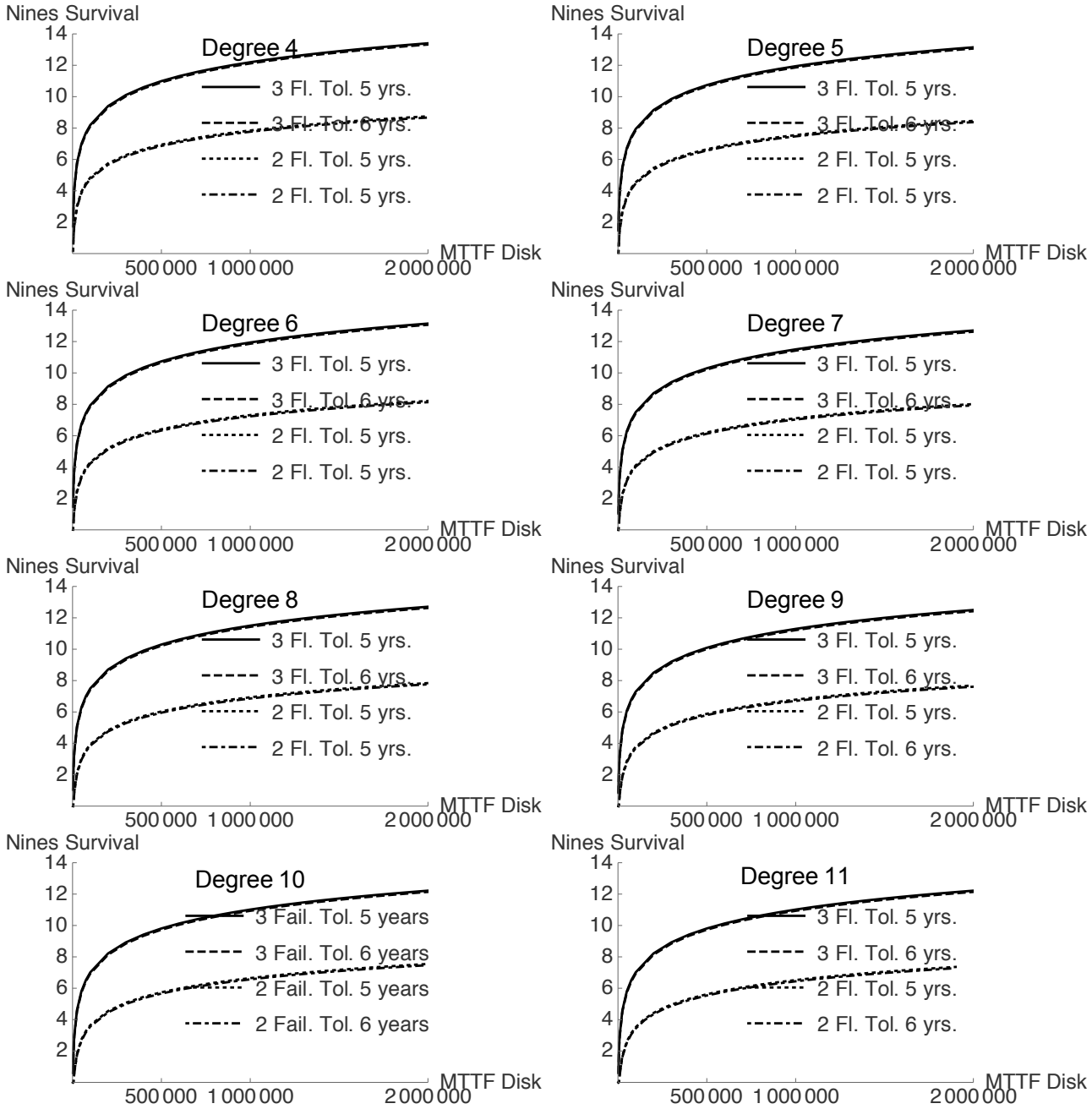

Fig. 8. Five and six year survival rate in number of nines in dependence on the disk mean time to failure (in hours) for layouts of degree 4, 5, 6, 7, 8, 9, 10, and 11.

A. Layout Equivalencies

The punctured layout of degree k has $2k$ parity disks and $k(2k-1)$ data disks in the two-failure tolerant version. These numbers are exactly the ones for a RAID Level 6 with k reliability stripes each consisting of $2k-1$ data disks and 2 parity disks. The three-failure tolerant version of the punctured layout has $3k$ parity disks and $k(2k-2)$ data disks, corresponding to a RAID Level 6+1 layout with k reliability stripes, each consisting of $2k-2$ data disks and 3 parity disks. This allows us direct comparisons between the two types of disk arrays.

In one point of comparison, flexibility, the RAID organization is a clear winner. There are configurations for any number of stripes and any number of data disks per stripe. As we will now see, the flat layouts offer considerably higher reliability.

B. RAID 6 and 6+1 Survivability Formulae

We now derive formulae for the survival of the data in a RAID level 6 organization after f failures. A failure pattern of f failed devices does *not* lead to dataloss and constitutes therefore a "good" pattern, if there is no stripe with more

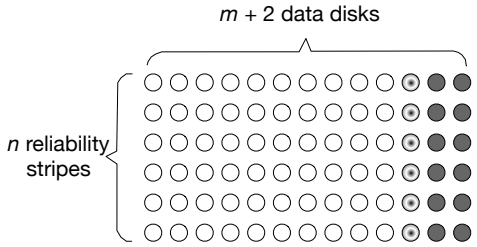

Fig. 9. Definition of parameters for a RAID Level 6 layout, where one data disk can be changed into an additional parity disk for a RAID Level 6+1 layout.

than two failed devices. We call the number of stripes with one failed device κ_1 and the number of stripes with two failed devices κ_2. Since the number of failed devices adds up to f, we have

$$\kappa_1 + 2\kappa_2 = f \qquad \Rightarrow \kappa_1 = f - 2\kappa_2.$$

In addition, we know that $\kappa_1 \geq 0$, $\kappa_2 \geq 0$, $\kappa_1 \leq n$, and $\kappa_2 \leq n$. Under those conditions, we know that the number of good failure patterns with these parameters is

$$n6_{\text{good}}(n, m, \kappa_1, \kappa_2) = \tbinom{n}{\kappa_1}\tbinom{n-\kappa_1}{\kappa_2}\tbinom{m+2}{1}^{\kappa_1}\tbinom{m+2}{2}^{\kappa_2},$$

otherwise it is 0. The total number of good patterns is obtained by summing over the various possibilities for κ_2, yielding

$$g6(n, m, f) = \sum_{\kappa_2=0}^{\lfloor f/2 \rfloor} n6_{\text{good}}(n, m, f - \kappa_2, \kappa_2).$$

While not exactly closed form, this equation can be evaluated precisely by using fractions of arbitrarily large integers with a tool like Mathematica. The same is true for the RAID level 6+1, though the formula and its evaluation are more involved. A failure pattern for RAID level 6+1 has lead to dataloss if one of the stripes contains more than three failed devices. If we denote the number of stripes with one, with two, and with three failed devices respectively with κ_1, κ_2, and κ_3, then the number of patterns with these characteristics are

$$\begin{aligned} & n7_{\text{good}}(n, m, \kappa_1, \kappa_2, \kappa_3) \\ = \quad & \tbinom{n}{\kappa_1, \kappa_2, \kappa_3, n-\kappa_1-\kappa_2-\kappa_3}\tbinom{m+3}{1}^{\kappa_1}\tbinom{m+3}{2}^{\kappa_2}\tbinom{m+3}{3}^{\kappa_3} \end{aligned}$$

and zero otherwise. The total number of good patterns is again obtained by summing up over the various possibilities

$$\sum_{\kappa_1+2\kappa_2+3\kappa_3=f;\kappa_1\geq 0;\kappa_2\geq 0;\kappa_3\geq 0;} n7_{\text{good}}(n, m, \kappa_1, \kappa_2, \kappa_3)$$

C. Robustness Comparison

We compare the resulting robustness in Figure 10. The black graphs are for the two-failure tolerant layouts and the red ones for the three-failure tolerant ones. The solid graphs give the robustness for the RAID layouts and the dotted ones for the punctured layouts. Both sets of graphs with the same failure tolerance start out and finish together; this is because the have the same number of failures they can tolerate for sure and equally naturally, both fail for sure when the number of failures exceeds the number of parity disks. However, if the number of failed devices is between these two extremes, the robustness, the probability that data survives in the presence of that number of failed devices, differs considerably. We also observe that as the degree and therefore the size of the ensemble increases, the punctured layout with two failure tolerance has closer robustness to that of the RAID layout with three failure tolerance. All together, these results show the positive effect of entanglement quite impressively.

D. RAID 6 and 6+1 Survival Rates

We again used the Euler Method to solve a Markov model to determine the survival rates in number of nines for the RAID 6 and RAID 6+1 layouts. The results are shown in Figure 11. As was the case for the punctured layouts, the extension of the economic lifespan of a disk ensemble by a year has much less of an impact then an increase in the reliability of the disks.

E. Comparison between RAID and Punctured Layouts

In Figure 12, we give the comparison between RAID 6 and RAID 6+1 on the one hand and the punctured layouts on the other hand of the survival probability after six years (in number of nines). The punctured layout is more reliable. The difference grows with the larger layouts, which is only natural.

V. Adjusting to Arbitrary Device Numbers

The total number of devices in a storage system arranged according to a punctured layout is a multiple of a large number. For $d = 8$, we obtain a 2 to 15 ratio of parity over data devices, but the device number needs to be 136 or a multiple thereof. This is however only necessary if we assign complete devices to the single function of either carrying user data or parity data.

Declustering distributes the role of parity and data devices so that each single device has about the same number of parity data and user data and therefore that the write-loads resulting from "small writes" (small changes in situ that affect only a single data block) tend to be more equal. It also allows us to use any number of devices for a certain layout as long as it is at least as large as the total number of devices from Table I. A very simple scheme uses the "left rotate" layout such as the one depicted in Figure 13. The vertical rectangles present 10 storage devices. Just as for RAID Level 6 (see Figure 1), they are broken up into disklets, depicted as the colored squares. Assume that we have a layout for 7 devices. We then organize the first disklets on the first seven devices according to the layout. We then organize the next set of seven disklets, three on the remaining three devices and four in the previous set of devices, again using the layout. The belonging to a certain seven-device layout is represented in Figure 13 through different colors. The different roles of a block (user data versus parity data) is indicated by the absence of presence of linear grading of each block. We can see that even though the numbers do not fit well, the left-rotate organizations uses

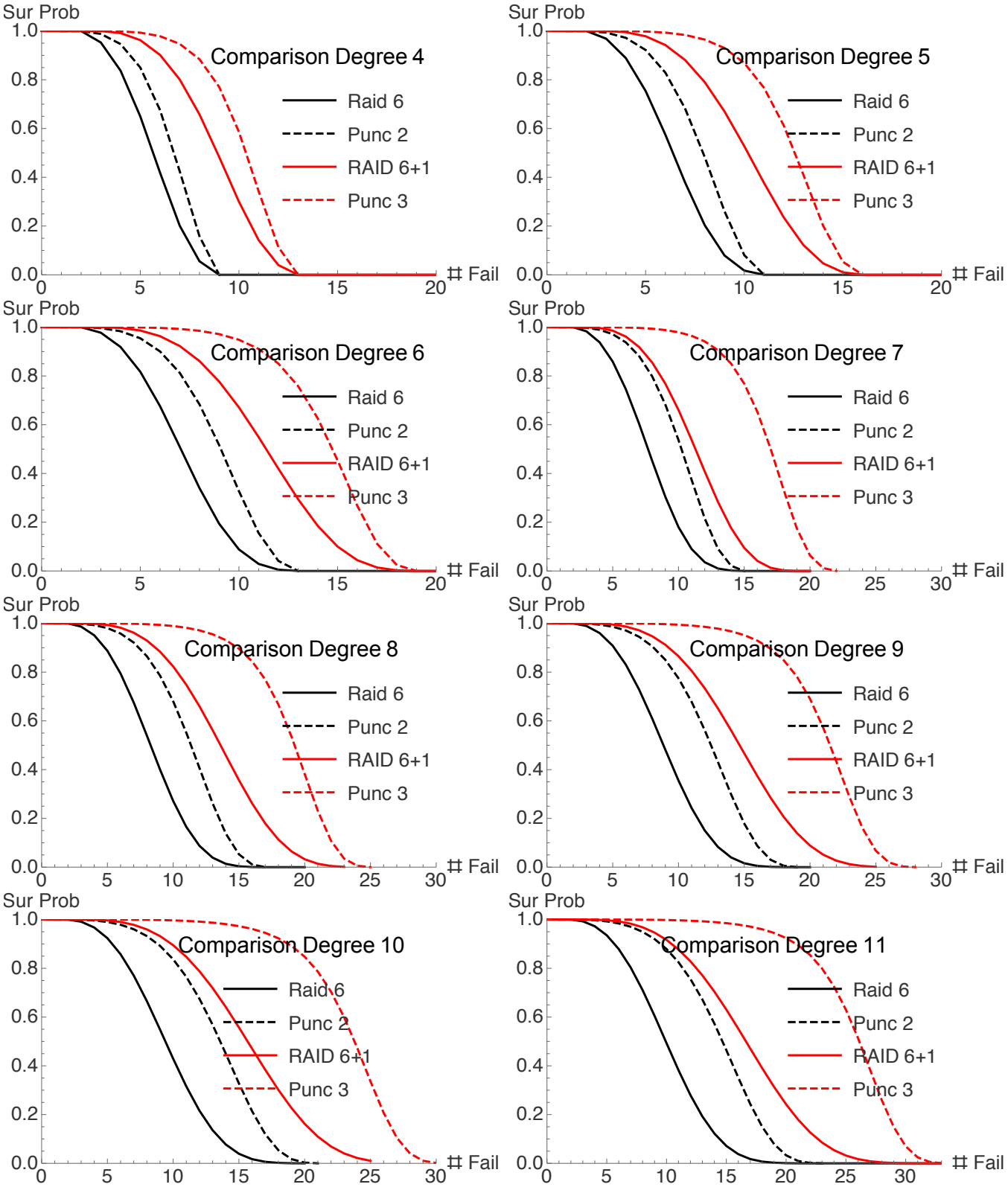

Fig. 10. Robustness comparisons for degrees 4 - 11 for the punctured and the RAID layouts with two and three failure tolerance.

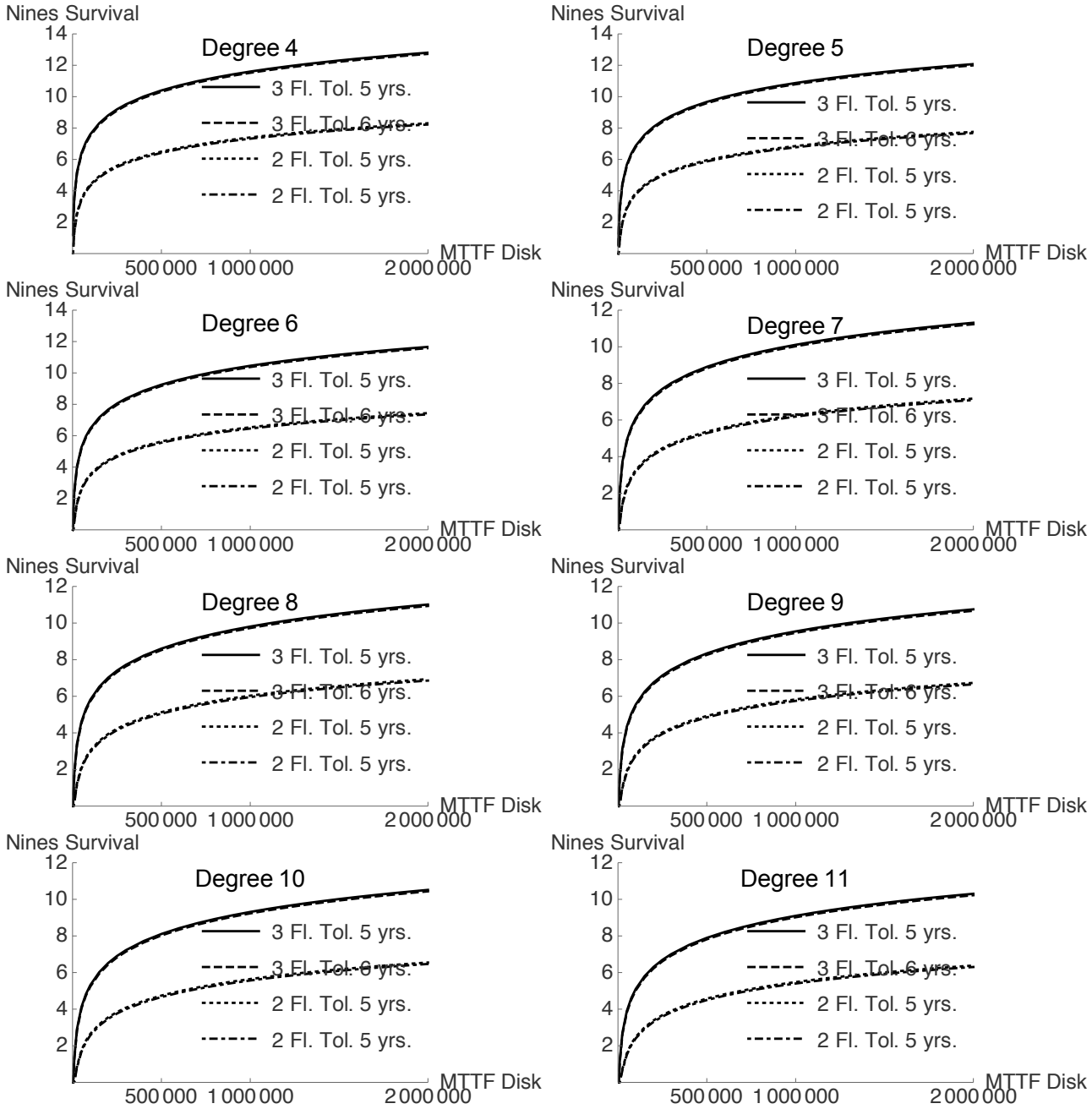

Fig. 11. Five and six year survival rate in number of nines in dependence on the disk mean time to failure (in hours) for layouts RAID 6 and RAID 6+1 corresponding degree 4, 5, 6, 7, 8, 9, 10, and 11.

up almost all of the available space and almost evens out the number of parity data carrying disklets on each device.

This particular layout is actually far from optimal. For example, if the first and the second device dies, then the third device's disklets are used for recovery six times, whereas the fifth device's disklets are only used four times. Finding better organizations is a combinatorial problem that needs to be left to future research.

At first glance, it seems that declustering does away with the reliability advantage of punctured layouts compared with an equally declustered RAID Level 6 organization, since all failures of three disks seem to imply data-loss. Indeed, if we use a random declustering organization with a very large number of disklets per device, then this is asymptotically the case. However, we can achieve good declustering with very moderate numbers of disklets per device, and in this case, the differences in survival rates are still present though not quite as marked.

Recently, a theoretical study by Iliadis extended a previous study on the reliability of erasure coded storage systems [25],

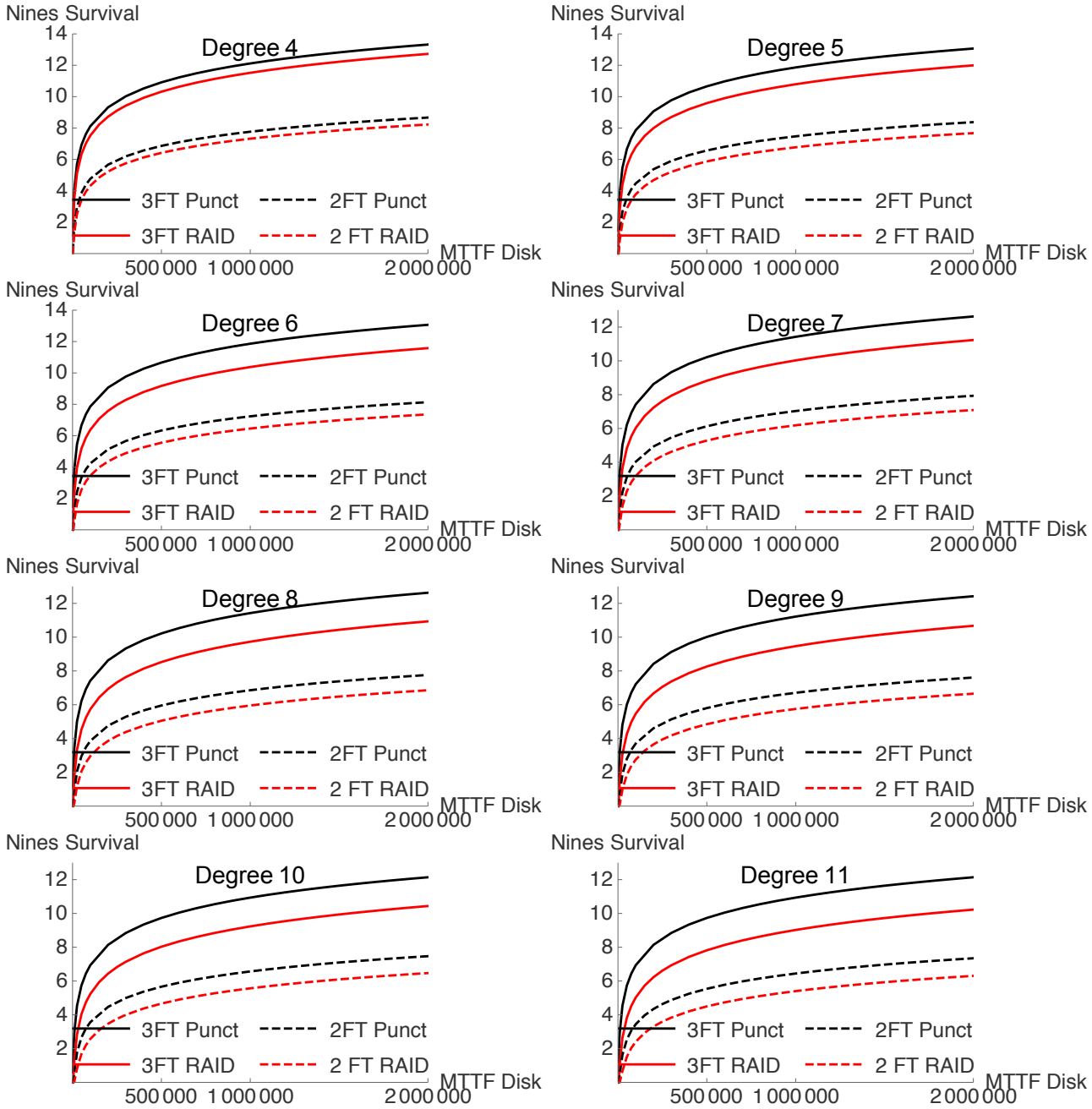

Fig. 12. Comparison of the six year survival rates (in number of nines) of analogue punctured and RAID layouts.

[9]. This study confirms the broad lines of behavior that were known since the seminal papers by Muntz [14] and Holland and Gibson [8] and the early works by various authors [1], [21], [22], [24], [26]. The load at individual devices is roughly inversely proportional to the number of devices over which we decluster and thus inversely linearly reduces the "window of vulnerability" during which additional failure can lead to dataloss. To our best knowledge, no published work discusses the fast, reliable detection of device failures (though presumably heart beat monitoring can achieve this), and how even the reconstruction load at individual devices is. The latter is not an issue when parity stripes are declustering over all disks in a large data center, since it is unlikely then that any device would have to be read more than once, but it becomes an issue in our situation. It should be noted that one potential benefit of flat layouts is that each failure can be repaired using two independent reliability stripes.

Whatever the underlying technology (disk drives, Flash memory, or newer memories such as PCM), our "super-storage" devices will have to contain spare components and will have to do some moderate amount of declustering.

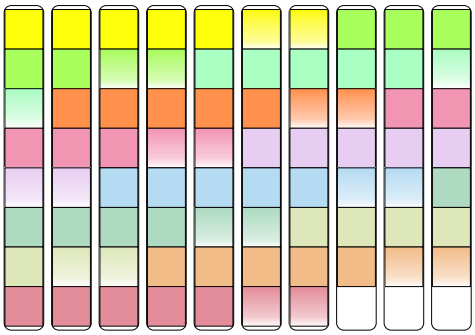

Fig. 13. Left rotate layout for declustering

VI. Future Work

The punctured layouts are not the only layouts possible. In Figure 14, we show a two-dimensional layout. It consists of $n \times n$ data disks arranged in a two-dimensional grid, to which for each column and row parity disks for a total of $2n$ are added. In the three-failure-tolerant layout, the data disks in the main diagonal are made into parity disks. Each row and column of data disks forms a reliability stripe. The new reliability stripes are formed by minor diagonals. In this layout, the assignment of disks to stripes is somewhat arbitrary. The right lower corner of Figure 14 shows that in this layout, some reliability stripes have an intersection of more than two, as the two disks in new stripe 1 are located in the same column. The only failure pattern of three disks in the square layout consists of a data disk together with the parity disk in the row and with the parity disk in the column. Since such a data disk is in one diagonal there is enough information in the system to make the three-failure-tolerant layout in fact three-failure tolerant.

From previous work [15] we know that designs based on complete graphs are somewhat less reliable than two- or three-failure tolerant square layouts which are exactly those based on complete bi-partite graphs. However, the former are considerably smaller than the latter, which makes them more attractive for flexible storage system layouts.

It is obvious that there is much potential research to be done evaluating different types of layouts.

Another issue for more research is that of declustering. With declustering, a layout for n disks can be used for layouts with $m, m > n$ disks. In addition, with declustering, the "window of vulnerability" after a storage device failure is greatly reduced since many reconstruction operations can be performed in parallel. In a very finely declustered layout, any failure of $t+1$ disks in a t-failure tolerant layout will lead to dataloss, because a bad configuration is going to happen in the many layouts that declustering superimposes on the array. However, a too finely declustered layout renders meta-data management complex. It is therefore natural to limit declustering by for example creating 1000 "disklets" on each hard drive and assign the disklets on a drive to different configurations. If this is a case, then many instances of $t+1$ and even some of $t+2$ failed devices do not lead to dataloss. Thus, the robustness of a moderately declustered system is of course

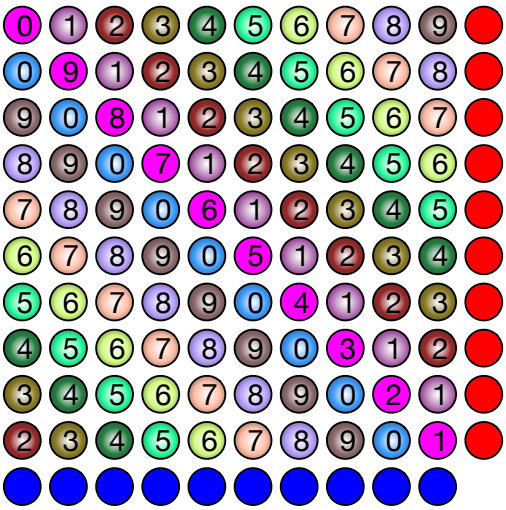

Fig. 14. An adjustable square layout.

less than the robustness of a non-declustered layout, but not quite as extreme as one might think. Finally, in a declustered layout, repair times are much smaller and the impact of the distribution of repair time on five-year survival rates should be higher. Modeling repair times as exponentially distributed introduces another source of modeling errors into the model. The lack of scientific activity on accurately modeling repair times (to my best knowledge) is attributable to the immense difficulty of making accurate modeling assumptions and of performing an analysis with them.

VII. Conclusions

Storage arrays consisting of individual components, whether flash or disk, need to deal with life-time variations of device reliabilities. Flash technology becomes more error prone with each erase cycle. An adjustable layout can sacrifice capacity for better failure protection when it is needed. Disk drives often, but not always suffer from infant disk mortality. A disk-based system can start out in a three-failure-tolerant configuration and move to a two-tolerant configuration once trust in the longevity of the hard drives has been established.

Since the punctured layouts presented here only use parity based on the exclusive-or operation, there is no need for the sophisticated and power-hungry CPUs that are needed to encode the linear codes used in RAID 6 and RAID 6+1.

The ease of parity generation in conjunction with the reliability gains should more than outweigh the flexibility of RAID based designs. We have argued (without proper investigation) that the lack of flexibility can be overcome using declustering with large disklet size. Declustering allows a trade-off between flexibility and the survival rate advantage of the resulting design.

References

[1] Alvarez, G., Burkhard, W., and Cristian, F. Tolerating multiple failures in RAID architectures with optimal storage and uniform declustering. In *Proceedings of the 24th Annual International Symposium on Computer Architecture* (1997), ACM, pp. 62–72.

[2] BAIRAVASUNDARAM, L. N., GOODSON, G. R., PASUPATHY, S., AND SCHINDLER, J. An analysis of latent sector errors in disk drives. In *ACM SIGMETRICS Performance Evaluation Review* (2007), vol. 35(1), ACM, pp. 289–300.

[3] BEACH, B. What is the best hard drive?, January 21 2015, www.backblaze.com/blog/best-hard-drive-q4-2014.

[4] BLAUM, M., HAFNER, J. L., AND HETZLER, S. Partial-MDS codes and their application to RAID type of architectures. *IEEE Transactions on Information Theory 59*, 7 (2013), 4510–4519.

[5] CALIS, G., AND KOYLUOGLU, O. A general construction for PMDS codes. *IEEE Communications Letters 21*, 3 (2017), 452–455.

[6] GREENAN, K. M., MILLER, E. L., AND WYLIE, J. J. Reliability of flat XOR-based erasure codes on heterogeneous devices. In *Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on* (2008), IEEE, pp. 147–156.

[7] GURUSWAMI, V., XING, C., AND YUAN, C. How long can optimal locally repairable codes be? *IEEE Transactions on Information Theory* (2019).

[8] HOLLAND, M., AND GIBSON, G. Parity declustering for continuous operation in redundant disk arrays. In *Proceedings of the 5th International Conference on Architectural Support for Programming Languages and Operating Systems* (1992), ACM, pp. 23–35.

[9] ILIADIS, I. Reliability of erasure coded systems under rebuild bandwidth constraints. In *Eleventh International Conference on Communications Theory, Reliability, and Quality of Service* (2018), IARIA, pp. 1–10.

[10] KIRKMAN, T. P. On a problem in combinations. *Cambridge and Dublin Math. J 2* (1847), 191–204.

[11] KLEIN, A. Backblaze hard drive stats for 2018, January 2019, www.backblaze.com/blog/hard-drive-stats-for-2018.

[12] LAWLESS, J. On the construction of handcuffed designs. *Journal of Combinatorial Theory, Series A 16*, 1 (1974), 76–86.

[13] MEZA, J., WU, Q., KUMAR, S., AND MUTLU, O. A large-scale study of flash memory failures in the field. In *ACM SIGMETRICS Performance Evaluation Review* (2015), vol. 43(1), pp. 177–190.

[14] MUNTZ, R., AND LUI, J. Performance Analysis of Disk Arrays Under Failure. In *Proceedings Very Large Data Bases* (1990), IEEE, p. 162.

[15] PÂRIS, J.-F., LONG, D. D., AND LITWIN, W. Three-dimensional redundancy codes for archival storage. In *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2013 IEEE 21st International Symposium on* (2013), IEEE, pp. 328–332.

[16] PÂRIS, J.-F., SCHWARZ, T. J., AMER, A., AND LONG, D. Protecting RAID arrays against unexpectedly high disk failure rates. In *Proceedings, 20th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC)* (2014).

[17] PLANK, J. S., GREENAN, K. M., AND MILLER, E. L. Screaming fast galois field arithmetic using intel simd instructions. In *FAST* (2013), pp. 299–306.

[18] SCHROEDER, B., DAMOURAS, S., AND GILL, P. Understanding latent sector errors and how to protect against them. *ACM Transactions on storage (TOS) 6*, 3 (2010), 9.

[19] SCHROEDER, B., AND GIBSON, G. A. Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you? In *Usenix Conference on File and Storage Technologies* (2007), vol. 7, pp. 1–16.

[20] SCHROEDER, B., AND GIBSON, G. A. Understanding failures in petascale computers. In *Journal of Physics: Conference Series* (2007), vol. 78(1), IOP Publishing, p. 012022.

[21] SCHWABE, E., AND SUTHERLAND, I. Improved parity-declustered layouts for disk arrays. In *Proceedings of the sixth annual ACM symposium on Parallel Algorithms and Architectures* (1994), ACM, pp. 76–84.

[22] SCHWARZ, T., AND BURKHARD, W. Almost complete address translation (ACATS) disk array declustering. In *Proceedings of SPDP'96: 8th IEEE Symposium on Parallel and Distributed Processing* (1996), IEEE, pp. 324–331.

[23] SCHWARZ, T., LONG, D. D. E., AND PÂRIS, J. F. Triple failure tolerant storage systems using only exclusive-or parity calculations. In *IEEE 21st Pacific Rim International Symposium on Dependable Computing (PRDC)* (2015), pp. 245–254.

[24] SCHWARZ, T. J., STEINBERG, J., AND BURKHARD, W. A. Permutation development data layout (pddl). In *Proceedings Fifth International Symposium on High-Performance Computer Architecture* (1999), IEEE, pp. 214–217.

[25] VENKATESAN, V., ILIADIS, I., HU, X.-Y., HAAS, R., AND FRAGOULI, C. Effect of replica placement on the reliability of large-scale data storage systems. In *2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems* (2010), IEEE, pp. 79–88.

[26] XIN, Q., MILLER, E., SCHWARZ, T., LONG, D., BRANDT, S., AND LITWIN, W. Reliability mechanisms for very large storage systems. In *Proceedings of the 20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies* (2003), IEEE, pp. 146–156.

[27] XU, L., BOHOSSIAN, V., BRUCK, J., AND WAGNER, D. G. Low-density MDS codes and factors of complete graphs. *Information Theory, IEEE Transactions on 45*, 6 (1999), 1817–1826.

[28] XU, Q., XI, W., YONG, K. L., AND JIN, C. Concurrent regeneration code with local reconstruction in distributed storage systems. In *Advanced Multimedia and Ubiquitous Engineering*. Springer, 2016, pp. 415–422.