Protecting Single Shingled Write Drives Against Latent Sector Failures

Thomas Schwarz, SJ
Marquette University
Milwaukee, WI, USA
Thomas.Schwarz@Marquette.edu

ABSTRACT

The recently introduced shingled write technology is necessary for reaching drive capacities of 10 TB, but no longer allows blocks to be updated in place. This inconvenience however simplifies the use of intra-disk redundancy to protect disks against latent sector failures, necessary when the disk drive does not form part of a larger ensemble. Latent disk sector errors usually befall single disk blocks and are only discovered when repeated attempts to read the block have failed. It is known that they afflict a sizable portion of modern disks. Because of the nature of shingled writing, content is written to disks in large contiguous bands. Redundant information can be calculated over large segments of each band and embedded as parity blocks into the segment. We investigate strategies for generating redundant information, evaluate various codes using distributions of latent sector locations obtained by Schroeder and colleagues, and propose a simple pyramid code that combines easy creation and easy recovery of unreadable blocks with low storage overhead and good protection.

CCS CONCEPTS

• **Computer systems organization** → **Reliability**;

KEYWORDS

Intra-disk redundancy scheme, shingled-write disks

1 INTRODUCTION

Shingled Write Technology (SWT) [1, 8, 21, 24] became necessary to maintain the pace of data density increases in magnetic disks. A shingled write head uses a stronger, but asymmetric magnetic field. A stronger field allows the bit density in a track to be increased. Due to the asymmetry, shingled writing can overlap the currently written track with the previous track and leave only a small part of the previous track untouched. This results in higher track density. The large increase in data density comes at the cost of no longer allowing in-place updates, since a write now destroys data in between 4 and 8 adjacent tracks in one direction. Disks with shingled write technology organize data in bands separated by unused tracks and only appends data to a band.

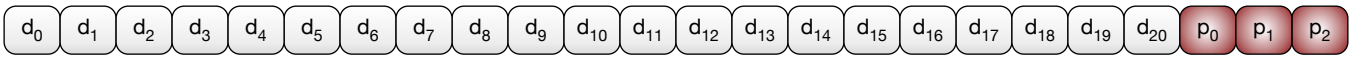
As disks sizes have increased, consumers, disk producers, and storage researchers have taken note of the existence of latent disk errors [6, 7, 10, 11, 20, 23]. A latent disk error is only discovered when we try to read an affected disk block. Their exact cause still remains unclear, but thanks to the work by Bairavasundaram, Goodson, Pasupathy, and Schindler [3] and by Schroeder, Damouras, and Gill [19] much more is now known. These researchers gained access to failure data of about 1.53 million disks from NetApp. In this set, 3.45% of all disks developed latent disk errors during a period of 32 months.

In installations with several or many disks, inter-disk redundancy such as provided by RAID Level 6 protects against complete drive failure as well as latent disk errors. Scrubbing [2, 15, 20] periodically reads all sectors in a disk and thereby detects latent sector failures, whose data then can be recovered. A variant of scrubbing verifies all data immediately after a write [18, 22], at least if the disk is not busy. Since the data to be written is still in buffer, an unreadable block has caused no harm. For isolated disks, intra-disk redundancy, introduced by Dholakia and colleagues [6, 7, 10], is the only option to protect against latent sector errors.

The work by Schroeder and colleagues has yielded some insights into the distribution of latent sector errors on disk [19]. The vast majority of latent sector errors affect a single block, but large bursts also occur. The distribution of inter-burst distance and burst length is best approximated by a Pareto distribution. This is unfortunate news for the designer of the erasure correcting code, since the Pareto distribution is tail-heavy. Long bursts that exceeds the erasure-correcting capability of a chosen code are more probable.

SYSTOR '18, June 4–7, 2018, HAIFA, Israel

ACM ISBN 978-1-4503-5849-1/18/06…$15.00
https://doi.org/10.1145/3211890.3211893



$$\mathbf{G} \cdot \begin{pmatrix} d_0 \\ d_1 \\ \vdots \\ d_{20} \end{pmatrix} = \begin{pmatrix} \gamma_{0,0} \cdot d_0 \oplus \gamma_{0,1} \cdot d_1 \oplus \ldots \oplus \gamma_{0,20} \cdot d_{20} \\ \gamma_{1,0} \cdot d_0 \oplus \gamma_{1,1} \cdot d_1 \oplus \ldots \oplus \gamma_{1,20} \cdot d_{20} \\ \gamma_{2,0} \cdot d_0 \oplus \gamma_{2,1} \cdot d_1 \oplus \ldots \oplus \gamma_{2,20} \cdot d_{20} \end{pmatrix} = \begin{pmatrix} p_0 \\ p_1 \\ p_2 \end{pmatrix}$$

Figure 1: Construction of a Linear MDS Code.

In the remainder of this contribution, we investigate a few promising intra-disk codes for their capacity to correct latent sector failures. Because shingled write disks cannot update single disk blocks, maintaining intra-disk parity is less challenging and code design simpler. We then investigate the candidate codes using the parameters of the burst-length and burst distance distributions obtained by Schroeder and colleagues [19]. We use these parameters to extrapolate the behavior of the much larger shingled disk drives. Presumably, more modern disks are better at controlling for latent sector failures, so that our test set is quite pessimistic. We still claim that it represents a "reasonably worst" scenario. We are able to identify a simple pyramid code [9] that only uses parity calculated with exclusive-or operations that provides large erasure correcting capability with low operational costs and low storage overhead. We also note that the effectiveness of any code depends on the propensity of a drive to exhibit latent sector errors and propose to investigate adaptive schemes.

2 INTRA-DISK REDUNDANCY CODING

Any intra-disk redundancy scheme incurs operational costs and storage overhead. Operational costs are comprised of the complexity of the parity calculation and the complexity of data recovery. The storage costs are given by the number of redundant sectors per coding block. All shingled write disks maintain their data in bands to which new data is appended. This simplifies intra-disk redundancy schemes since once written, data is never overwritten until garbage collection.

2.1 Linear MDS Codes

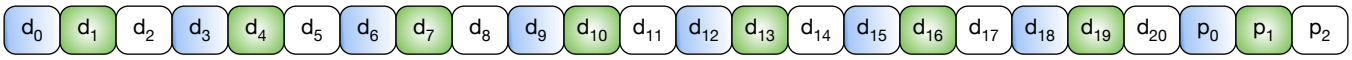
For the calculation of parity codes, we only consider linear codes. The simplest instance of a linear code takes an information word, n disk blocks, and adds a single parity block, which is obtained as the bit-wise exclusive-or of the data blocks. A *linear MDS code* is a natural generalization of this *simple parity code*. We interpret all data as a sequence of elements in a Galois field(finite field) $\mathscr{F}_{2^l}$ where each Galois field element is a bit string of length l. The contents of a disk block become an element of a vector space over $\mathscr{F}_{2^l}$ where each Galois field element is a bit string of length l. For example, a 4KB block can be interpreted as a 4096-dimensional vector over $\mathscr{F}_{2^8}$, as a 2048-dimensional vector over $\mathscr{F}_{2^{16}}$, or as a 1024-dimensional vector over $\mathscr{F}_{2^{32}}$. A linear MDS code with p parity is defined by an $n \times p$ *generator matrix* $\mathbf{G}$. To qualify as a generator matrix, all $n \times n$ submatrices of the matrix $(I_n|\mathbf{G})$ obtained by concatenating an $n \times n$ identity matrix with $\mathbf{G}$ need to be invertible. Suitable generator matrices exists in abundance as long as the *segment size* $n+p$ does not exceed 2^l, the number of Galois field elements. Using matrix inversion, any n of the $n+p$ parity blocks suffice to reconstruct all $n+p$ blocks. Codes such as RDP are also linear MDS codes but are implemented without Galois field operations [5].

Formally, the parity blocks are calculated as $\vec{p} = \mathbf{G} \cdot \vec{d}$. We create the parity blocks by multiplying all the data in a block with a certain Galois field element and then obtain a parity block as the exclusive-or of the results. Plank, Greenan, and Miller [17] show how this can be done very efficiently through the judicious use of the PSHUFB instruction (or their equivalents) in newer CPU architectures. We give an example for a code with segment size 24 = 21+3 in Figure 1. For such a small segment size, the matrix coefficients would presumably be in $\mathscr{F}_{2^8}$ and each data symbol d_i correspond to 4096 B interpreted as a column vector of dimension 4096.

2.2 Interleaved Parity Check Codes

Speedy parity calculation and data recovery with a linear MDS code relies on sophisticated, power-hungry processors. This makes code using only exclusive-or operations attractive. The interleaved parity check code has the same capability as a linear MDS code to deal with a single burst of unreadable blocks [7]. The code takes a contiguous set of n blocks numbered from 0 to $n-1$, calculates the exclusive-or of the blocks with the same reminder modulo p, $p < n/2$, and writes the result into a parity block, giving rise to p parity blocks. In Figure 2, $n = 21$ and $p = 3$. Any error burst of up to p blocks affects only one block in each stripe $S_k = \{d_i, i = k \bmod p\} \cup \{p_k\}$ and can therefore be recovered.

Operationally, the interleaved parity check code has advantages. First and foremost, the basic operation is the exclusive-or (XOR). Second, when we generate parity, we only need to process a data block once, by XOR-ing it with one of several parity blocks kept in the disk's RAM. In contrast, the linear MDS code needs to update all parity blocks with the single data block. Third, for recovery, the linear MDS code needs to invert a submatrix and then multiply this matrix with a vector



$p_0 = d_0 \oplus d_3 \oplus d_6 \oplus d_9 \oplus d_{12} \oplus d_{15} \oplus d_{18}$ $p_1 = d_1 \oplus d_4 \oplus d_7 \oplus d_{10} \oplus d_{13} \oplus d_{16} \oplus d_{19}$ $p_2 = d_2 \oplus d_5 \oplus d_8 \oplus d_{11} \oplus d_{14} \oplus d_{17} \oplus d_{20}$

Figure 2: Construction of an Interleaved Parity Code.

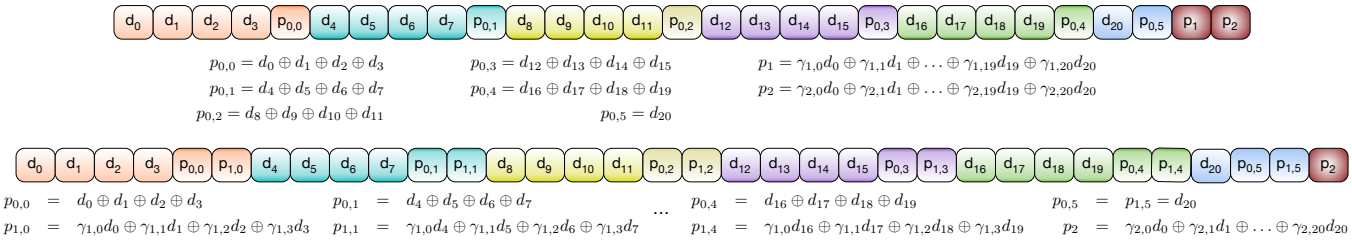


Figure 3: Two constructions of a Pyramid Code.

calculated from the remaining data and parity blocks in the segment.

The erasure correcting capability of a linear MDS code are of course optimal. One with p parity blocks can recover from the unavailability of *any* p blocks. In contrast, the interleaved parity code can recover from the unavailability of any p *contiguous* blocks. If there is a burst of $p-1$ contiguous, unreadable blocks and a single unreadable block in a large segment, then the probability of recovery is approximately $1/p$. Similarly, the probability that the code can recover from l isolated unreadable block in a large segment is approximately $P_{p,l} = (p(p-1)\ldots(p-l+1))/p^l$, which is small for even moderate l and p. For example, for $l = 2$ and $p = 4$, $P_{p,l} = 0.75$, for $l = 5$ and $p = 10$, $P_{p,l} = 0.302$, for $l = 10$ and $p = 20$, $P_{p,l} = 0.065$, and for $l = 20$ and $p = 40$, the probability is $P_{p,l} = 0.003$. This observation explains the relative poor showing of this code in our experiments reported below.

2.3 Pyramid Codes based on linear MDS codes

Finally, we can use a class of codes that are known as Pyramid codes [9]. We start with a linear MDS code on a large segment of n data blocks. The code generates k parity blocks as a linear combination of the data blocks. We divide k into $k_l + k_s = k$. We also divide the n data blocks into m about equally-sized small segments. We then replace the k_l parity blocks with $m \times k_l$ parity blocks (for a total of $k_l + m \times k_l$ parity blocks) that are calculated as a linear combination of the blocks in one of the m small segments. The calculation is the same, but the data blocks outside of the small segments are set to be zero.

Figure 3 gives two examples, where the data blocks are subdivided into four groups of four (and one group of one). In the top construction, we assume that the first parity of a linear code is just the exclusive-or parity of the data blocks.

Data recovery with a pyramid code starts with using small segments and then progressing to the use of the available redundancy for the complete large segment. This code is attractive since the $m \times k_s$ parity blocks can deal effectively with small bursts of unreadable blocks while still protecting against larger bursts using the parities of the large segment.

2.4 Exclusive-Or Parity Based Pyramid Codes

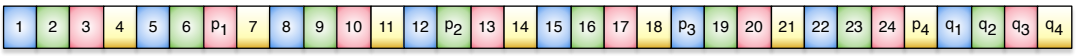
The idea of a pyramid code can also be applied to an interleaved parity code. We recall that the interleaved parity code adds a separate parity block as the exclusive-or of every m^{th} data block. Several layouts are possible, we investigate the layout depicted in Figure 4. The layout starts by placing r contiguous data blocks into a reliability stripe – a *small segment* –, to which it adds an exclusive-or parity. A contiguous set of these small segments (each comprising $r+1$ blocks) are then a grouped into a large segment, to which an interleaved parity code adds additional parity blocks.

In comparison to the previously discussed pyramid code, calculating parity is simpler. A data block contributes to two parity blocks, namely one because of its membership in the reliability stripe of m contiguous blocks and one from the interleaved parity check code. In contrast, writing data blocks using the pyramid code based on linear MDS codes involves $k_l + k_s$ writes.

Just as before, the small segments can be used to deal with isolated latent sector failures. However, if not used for this purpose, they can still be sometimes used in order to recover from a burst just larger than the number of parities in the interleaved parity check code.

3 EXPERIMENTAL SET-UP

The analysis by Schroeder and colleagues [19] of failure data some ten years ago has shown that the susceptibility for latent sector errors is surprisingly high but differs widely among different disk models and batches. The vast majority of latent sector errors (always $\geq 90\%$) affect single blocks, but very long bursts also happen. Both the length of error bursts and



$$p_1 = d_1 \oplus d_2 \oplus d_3 \oplus d_4 \oplus d_5 \oplus d_6 \qquad p_4 = d_{19} \oplus d_{20} \oplus d_{21} \oplus d_{22} \oplus d_{23} \oplus d_{24}$$
$$p_2 = d_7 \oplus d_8 \oplus d_9 \oplus d_{10} \oplus d_{11} \oplus d_{12} \qquad q_1 = d_1 \oplus d_5 \oplus d_8 \oplus d_{12} \oplus d_{15} \oplus p_3 \oplus d_{22}$$
$$p_3 = d_{13} \oplus d_{14} \oplus d_{15} \oplus d_{16} \oplus d_{17} \oplus d_{18} \qquad q_2 = d_2 \oplus d_6 \oplus d_9 \oplus d_{12} \oplus p_2 \oplus d_{16} \oplus d_{23}$$
$$q_3 = d_3 \oplus p_1 \oplus d_{10} \oplus d_{13} \oplus d_{17} \oplus d_{20} \oplus d_{24}$$
$$q_4 = d_4 \oplus d_7 \oplus d_{11} \oplus d_{14} \oplus d_{18} \oplus d_{21} \oplus p_4$$

Figure 4: A Pyramid Code with Exclusive-Or Parity.

Table 1: Parameters from distribution fitting for latent failure sectors (Table I, [19]).

Parameter	A-1	D-2	E-1	E-2	k-2	k-3	n-3	o-2
Single burst probability	0.9	0.98	0.98	0.96	0.97	0.97	0.93	0.97
Burst Length Pareto-α	1.21	1.79	1.35	1.17	1.2	1.15	1.25	1.44
Distance Pareto-α	0.008	0.022	0.158	0.128	0.017	0.00045	0.077	0.05

Table 2: Percentage of disks with a latent sector error for our extrapolated families

Type	% of Disks	Type	% of Disks
A-1	16.26	k-2	31.41
D-2	38.61	k-3	0.993
E-2	96.99	n-3	81.88
E-3	94.15	o-2	10.50

the distance between affected bursts are Pareto distributed. This is bad news for the protection against latent disk failures, since it means that longer bursts are more common then one would assume.

Luckily, the probability that a given disk sector suffers an error is very low. In fact, this makes analysis of the phenomenon very difficult. In particular, according to the results of Schroeder and colleagues [19] latent sector errors defy any simple explanation. For instance, one would expect a strong correlation between utilization and prevalence of latent sector errors, but this is just not the case.

Modeling under this type of uncertainty and a dearth of publicly available traces is by necessity speculative. *In our experiments we therefore use the parameters from the distribution fitting by Schroeder and colleagues [19] for various disk types and apply them to families of theoretical shingled write disk models with much higher capacity, namely 8TB organized in sectors of 4KB.* This extrapolation therefore no longer describes actual families of disks. Shingled write disks have a much larger bit density in a track and a much larger track density. Also, the actual disks have a sector size of 512B. It is unclear how changing the parameters will affect the frequency of latent sector errors. Since writes are the culprit for generating these errors, we can assume that changing the block size has no great effect. If disk drive geometry is a factor [19], then our methodology overestimates the frequency.

Our extrapolation shows the variety in disk behavior that we would expect. We are probably erring on the side of caution; actual disks should show a lower rate of latent sector incidences. The parameters for our hypothetical large disks are the same as for the families of disks investigated previously [19] and are presented in Table 1. The percentage of disks with a latent sector error for these hypothetical disks are given in Table 2.

We used extensive simulation (amounting to years of CPU time) in order to obtain the percentage of disks with uncorrectable latent sector errors after assuming various intra-disk redundancy schemes.

4 RESULTS

Our results show that all types of intra-disk redundancy are successful in reducing the proportion of disks with latent sector failures. No scheme with reasonable parity overhead can correct all latent sector errors with high confidence. This is a function of the relatively high likelihood of very long bursts. The best protection scheme for a given disk family depends primarily on the susceptibility of the disk family for latent sector errors.

We present the results in terms of a contour graph. The lines connect points with parameters as coordinates that give rise to the same probability that a disk drive contains one or more unrecoverable latent sector failures. These probabilities are given in percentages. The colors correspond to points representing parameters with approximately the same data loss probability due to latent sector failures and change color from blue through orange to beige as the probability increases.

4.1 Interleaved Parity Code versus linear MSD code

We first compared the protection offered by the interleaved parity code to that of the linear MDS code. Figures 5 to 12 give our results. We used segment sizes between 100 and

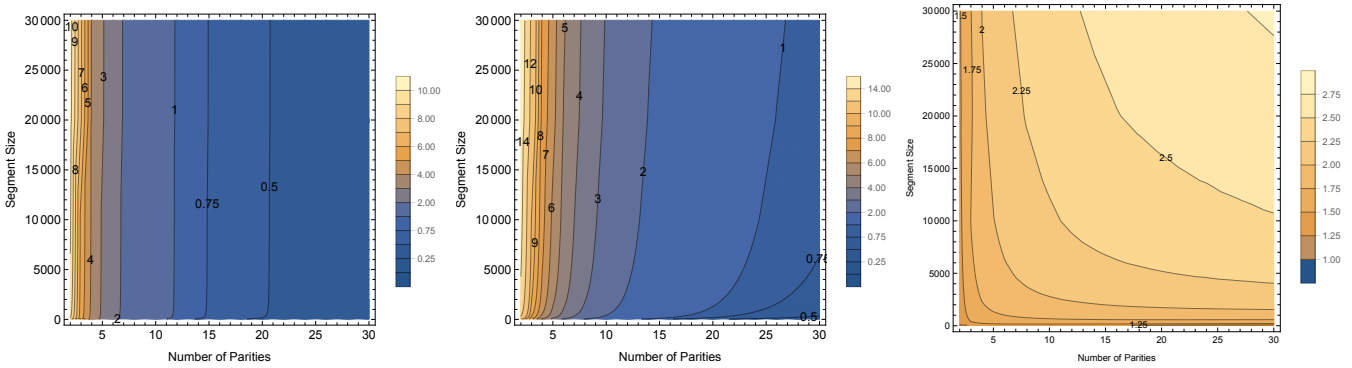


Figure 5: Comparison of the Interleaved Parity Code against a linear MDS code for A-1. Left: linear MDS, Middle: Interleaved Parity, Right: Comparison.

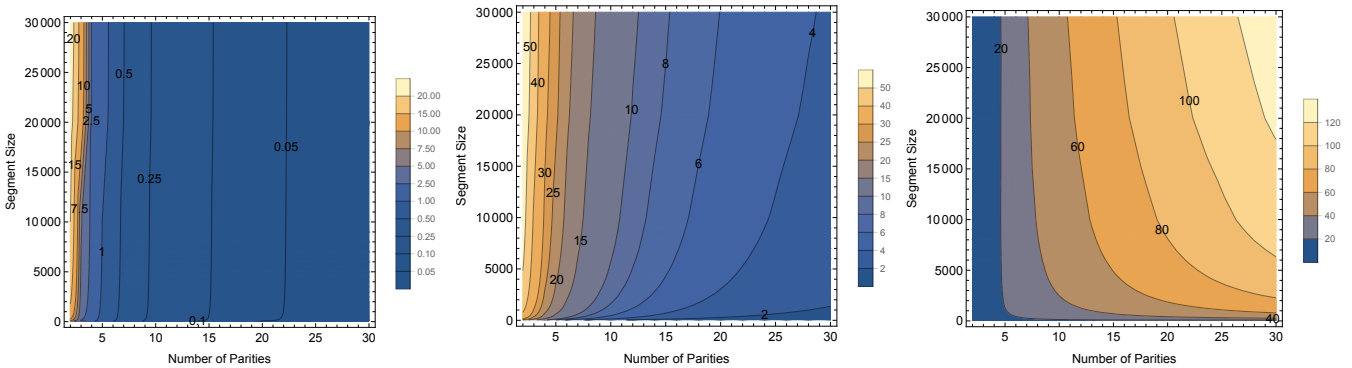


Figure 6: Comparison of the Interleaved Parity Code against a linear MDS code for D-2.

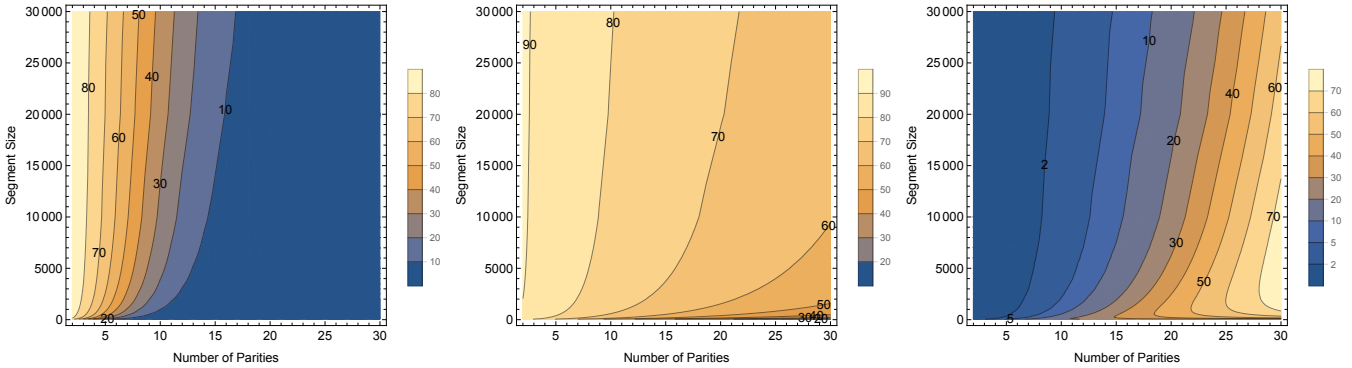


Figure 7: Comparison of the Interleaved Parity Code against a linear MDS code for E-1.

30,000 blocks and varied the total number of parity per segment between 1 and 30. At the left of each Figure, we give the results for the linear MDS code, in the middle for the Interleaved Parity Code, and on the right the ratio of dataloss probability of Interleaved over MDS. This ratio is always at least one, since MDS codes have optimal recovery capabilities for a given storage overhead.

We make several observations:

(1) For most disk families, the contour lines run almost in parallel to the y-axis, at least for the MDS code. In this case, then dataloss probability of the whole disk depends little on the segment size and almost completely on the number of parities. With other words, for a given storage overhead, one should then choose the largest segment size compatible with operational restrictions. One should prefer to add parities to complete bands instead of breaking the bands into smaller segments. The highly susceptible disk families E-1, E-2, and n-3 are an exception, presumably because they combine bursts of large size with relatively frequent small bursts. Only for k-3 are the contour lines in parallel for the Interleaved Parity Code, but A-1, D-2, and k-2 come close. Here, the influence of segment size becomes visible for larger number of parities.

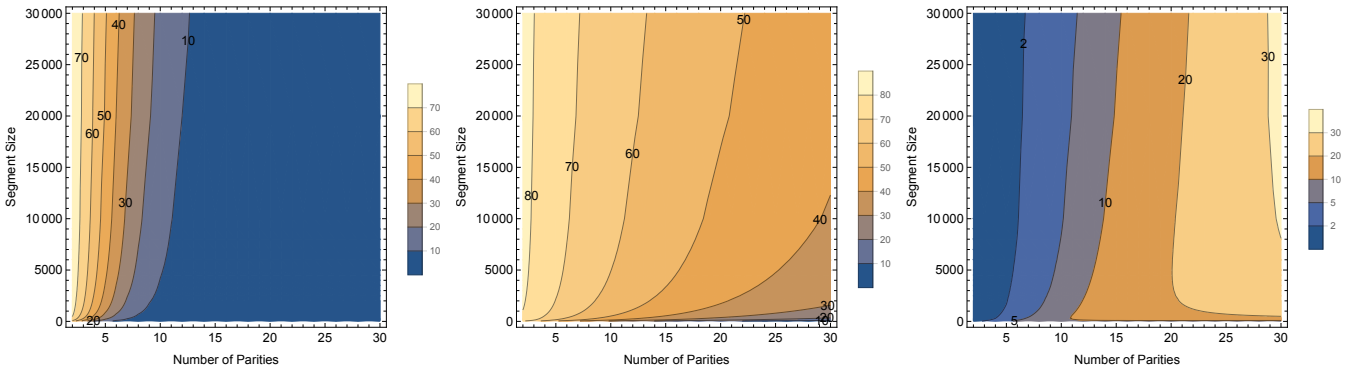


Figure 8: Comparison of the Interleaved Parity Code against a linear MDS code for E-2.

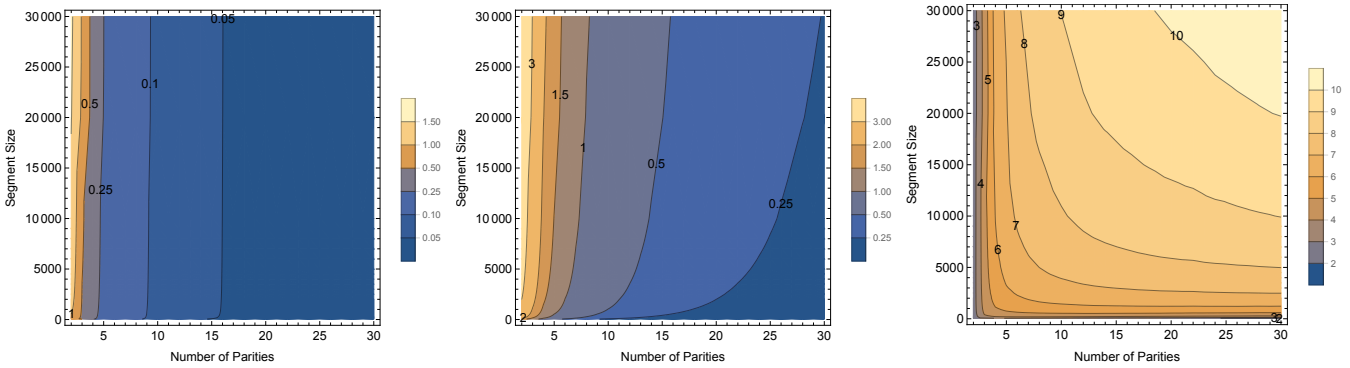


Figure 9: Comparison of the Interleaved Parity Code against a linear MDS code for k-2.

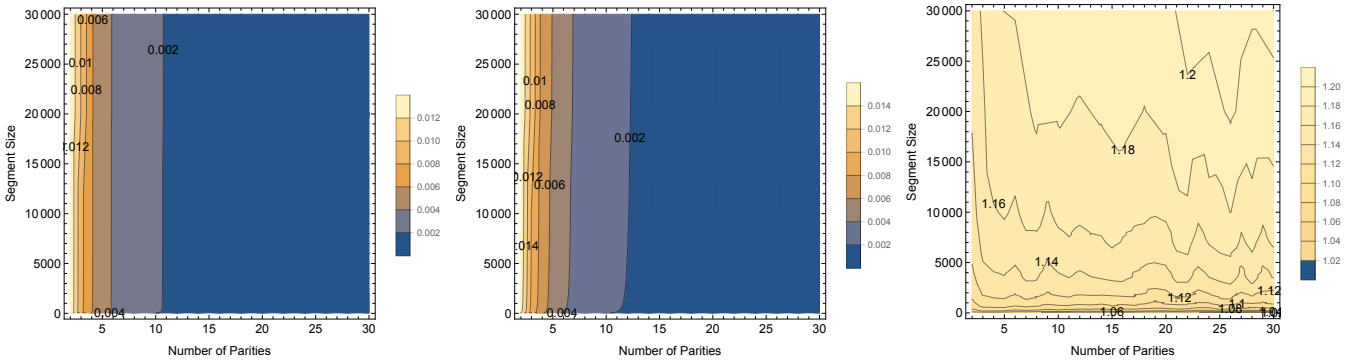


Figure 10: Comparison of the Interleaved Parity Code against a linear MDS code for k-3.

(2) The linear MDS code is of course always better, but the difference is quite small for k-3 and moderate for A-1, k-2 and o-2. It is however drastic for the more susceptible disk families D-2, E-1, E-2, and n-3. That the contour lines in the comparison for k-3 jiggle is a result of the paucity of dataloss for this family.

(3) The relative benefit of linear MDS codes over the Interleaved Parity Code is largest for larger number of parity and larger segment sizes, but E-1 is an exception.

4.2 Pyramid Codes based on linear MDS codes

We then investigated whether a pyramid code would help to improve the reliability of the linear MDS code. We break a segment (referred to as the large segment) of size 10,000 blocks into smaller segments. We also obtained results for a large segment size of 1000 blocks, but the results are similar. The size of these small segments is one of our parameters. Each small segment calculates k_s redundant parity blocks. In addition, we independently calculate k_l parity blocks for the large segment. The total number of parity blocks depends of

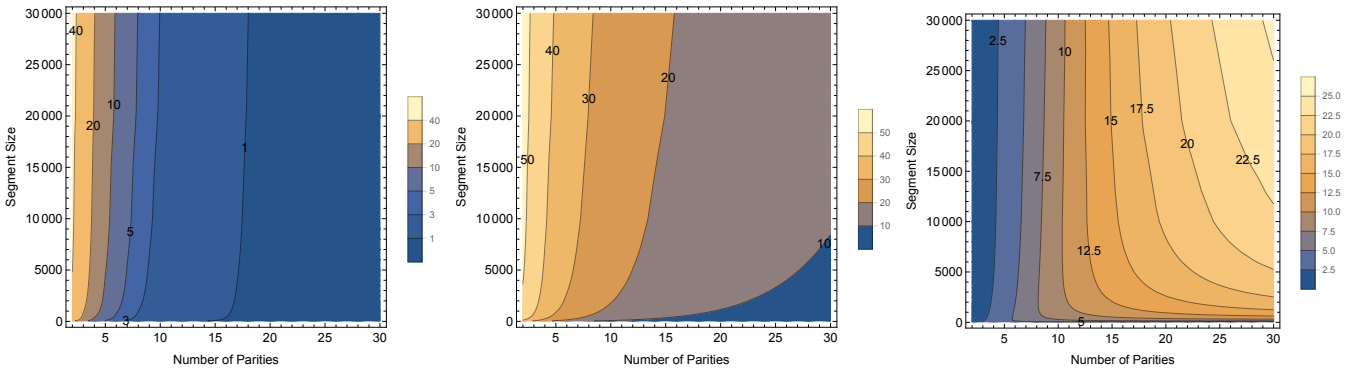


Figure 11: Comparison of the Interleaved Parity Code against a linear MDS code for n-3.

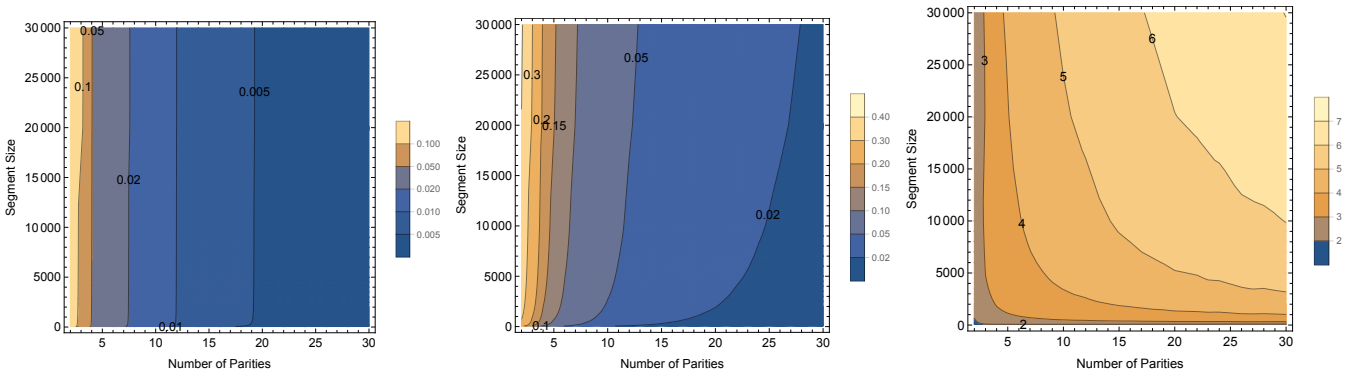


Figure 12: Comparison of the Interleaved Parity Code against a linear MDS code for o-2.

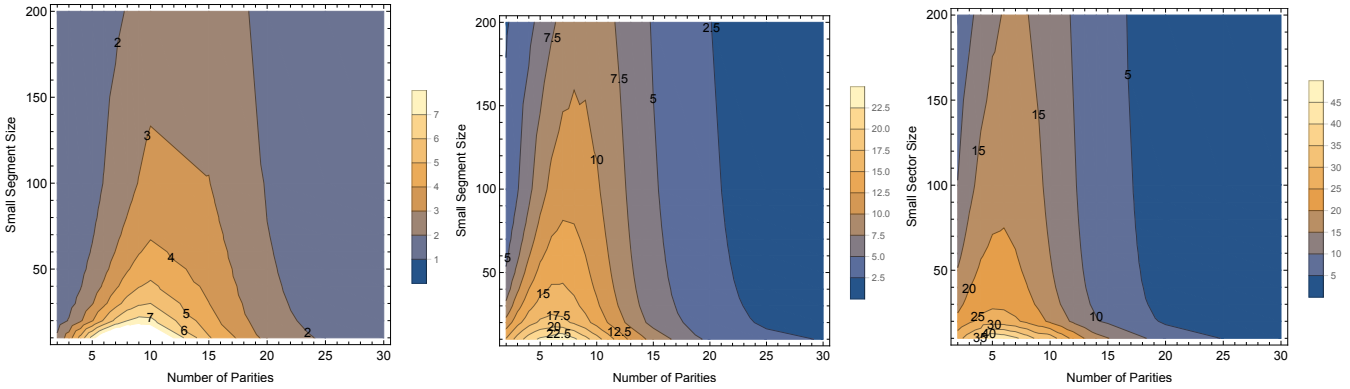


Figure 13: Comparison of a linear MDS code with a linear MDS Pyramid code with $k_s = 1$ (left), $k_s = 3$ (middle), and $k_s = 5$ (right) for the E-1 family. The x-axis is given by $k_s + k_l$.

course on the size of the small segments. This encoding is guaranteed to be able to correct the dataloss due to a burst of $k_s + k_l$ unreadable sectors, but can correct a burst of $2k_s + k_l$ sectors, if the burst is divided over two small segments with at least k_s unreadable sectors in each small segment. We simulated this layout for all families with $k_s = 1, 3$, and 5 parity blocks per small segment. For space reasons, we only display typical behavior. We varied the size of the small segment and $k_s + k_l$, the number of parity blocks that would be affected if we were to change a single data block. (Of course, we do not update single blocks in a shingled-write disk.)

In Figure 13, we show the improvement disks of type E-1. This type is the most susceptible to latent sector errors. As we can see, a large improvement in reliability can be realized, but only if the small groups are small and $k_s + k_l$, the number of parities is also small. Unfortunately, small groups create considerable storage overhead. To some extent, this behavior is shared by E-2 for $k_s = 5$.

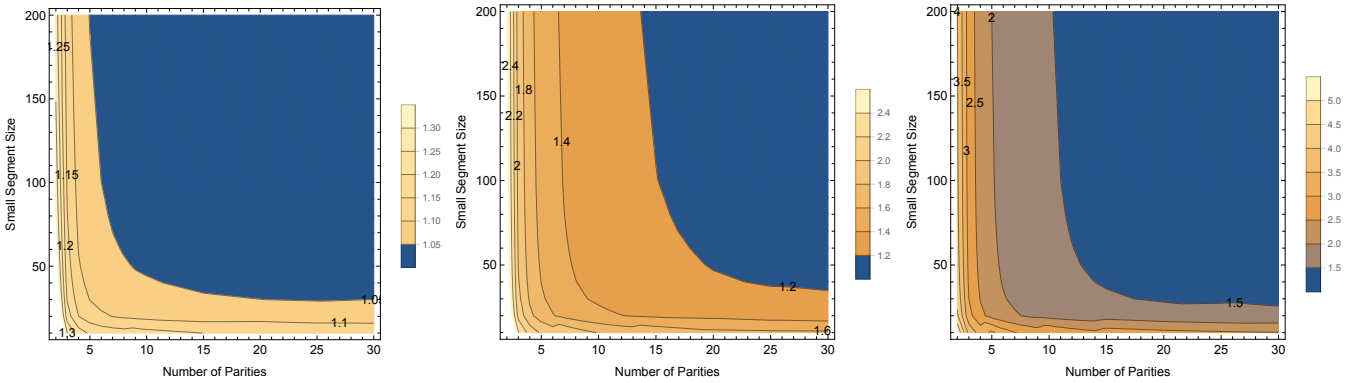


Figure 14: Comparison of a linear MDS code with a linear MDS Pyramid code with $k_s = 1$ (left), $k_s = 3$ (middle), and $k_s = 5$ (right)for the k-2 family.

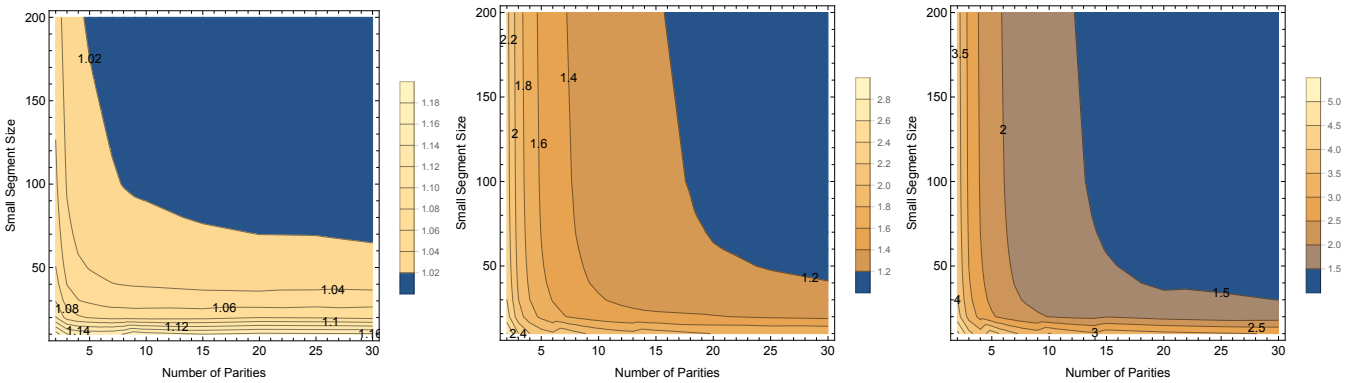


Figure 15: Comparison of a linear MDS code with a linear MDS Pyramid code with $k_s = 1$ (left), $k_s = 3$ (middle), and $k_s = 5$ (right) for the o-2 family.

The behavior for more reliable disks is different. In Figures 14 and 15, we give the same comparison for o-2 and k-2 as typical examples. The improvements are more modest, and are realized in small band around the x- and the y-axes, namely when the size of the small groups is small or the number of parities is very small. The contour graphs are similar for A-1, D-2, E-2 if $k_s = 1$ or $k_s = 3$, and k-3.

4.3 Pyramid Codes based on the Interleaved Parity code

Finally, we investigated the pyramid code based on the Interleaved Parity Code. Recall that we break the large segment into small segments. To each small segment we add a single parity calculated as the exclusive-or of the blocks in the segment. We simulated large segments of 10000 and 1000 blocks and varied the size of the small segments and the number of parities for the interleaved parity code. We recall that a single data block only contributes to two parity blocks and that all encoding and recovery operations only involve exclusive-or operations.

We give our results as contour graphs in Figures 16, 17, and 17 for a large segment size of 10,000. If we compare the graphs with those depicting the reliability of linear MDS codes, we see that just increasing the number of parities in the interleaved code results in protection that is equal to that provided by the MDS code.

The parity overhead consists mostly of the one caused by the exclusive-or parity block for each of the small sectors. In the case of the less susceptible families of disks, namely k-2, k-3, and o-2, it is possible to choose a large size for the small segments (for example 100) and in addition 40 to 50 interleaved parities. This choice of parameters reduces the percentage of disks with incorrigible latent sector failures to values for below 0.1%. The parity overhead would be $\frac{1}{100} + \frac{50}{10000} = 1.5\%$ if we choose large segment sizes of 10000 blocks. As the disks writes a block to a band, it updates the parity of the small segment and one parity of the interleaved parity check code. It needs to keep the parities for the interleaved parity check code and in addition the small segment parity block in RAM. If a disk can be equipped with sufficient RAM, this code combines good data protection with simple encoding and simple erasure correction.

A parity overhead of around 1.5% is puny when compared to the usual overhead in actual storage systems such as the

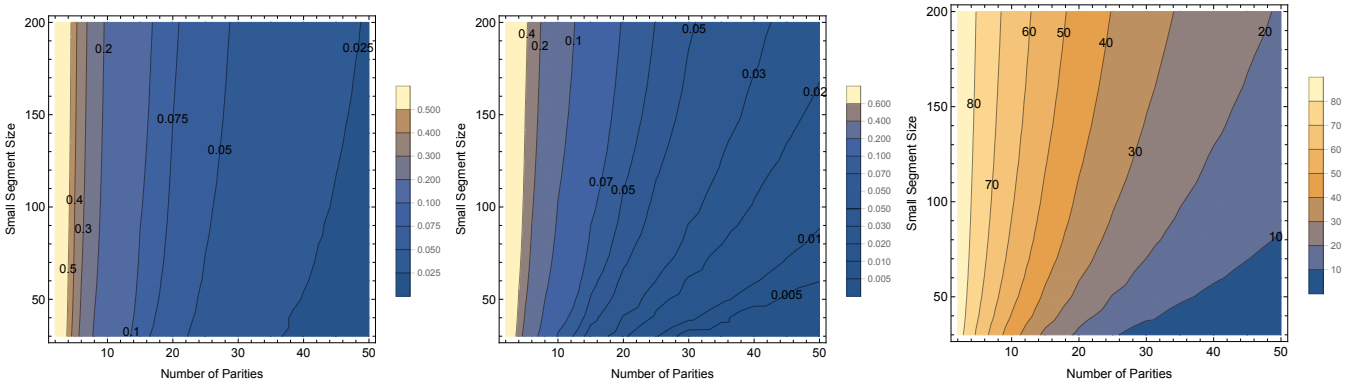


Figure 16: Percentage of disks with a latent sector error using a pyramid code based on the Interleaved Parity Code for A-1 (left), D-2 (middle), and E-1 (right) disks. The x-axis gives the number of parity blocks in a large segment. The sector size is 10,000 blocks.

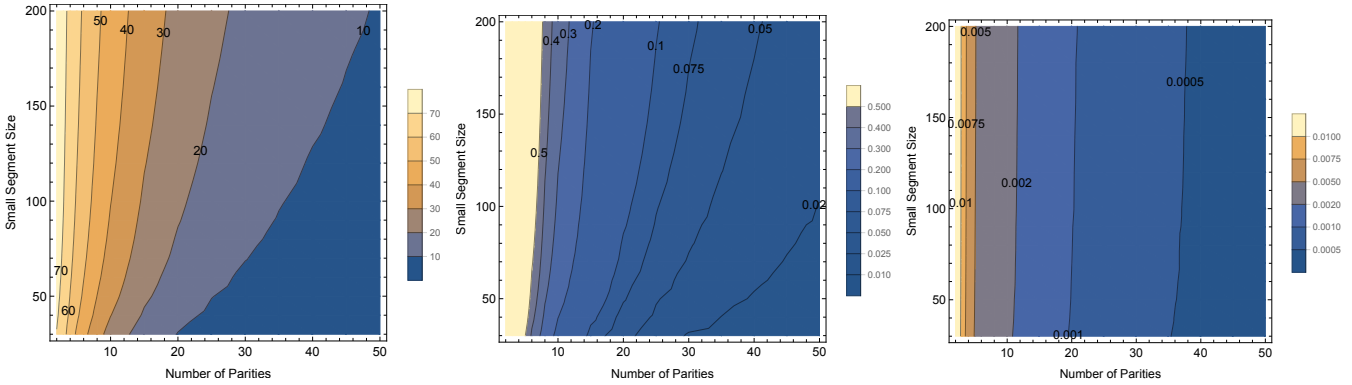


Figure 17: Percentage of disks with a latent sector error using a pyramid code based on the Interleaved Parity Code for E-2 (left), k-2 (middle), and k-3 (right) disks.

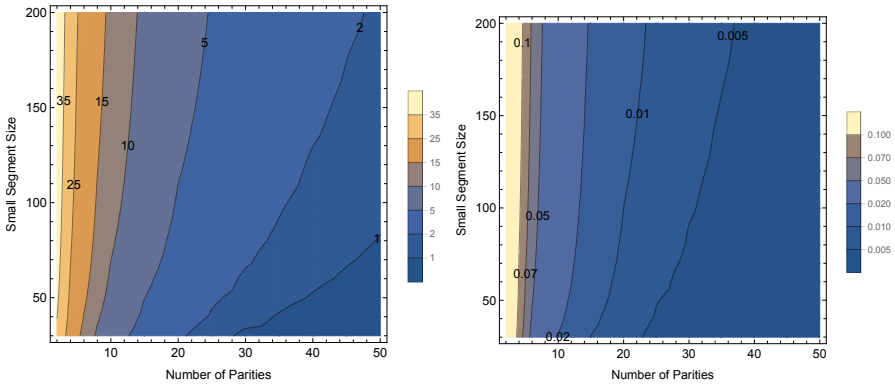


Figure 18: Percentage of disks with a latent sector error using a pyramid code based on the Interleaved Parity Code for n-3 (left) and o-2 (right) disks.

12.5% of a RAID Level 6 with a 14+2 stripes. If a disk manufacturer decides to build intra-disk redundancy into every disk sold, then this overhead is no longer negligible.

4.4 Overall Evaluation

If the only criterion for the selection of an intra-parity code is storage overhead, then one has to select Linear MDS codes. Unfortunately, we have seen that the number of parity blocks per segment has to be quite high in order to protect against

large bursts. The linear MDS codes exact a high computational overhead since every data block written needs to be processed for each parity block.

If a disk drives encounters a latent sector error, then the drive has to read all sectors in a reliability stripe in order to reconstruct the error. If we choose a large segment size, then we need to read all sectors in it, which leads to a high *tail latency*, a phenomenon described at length by Li and colleagues [14]. Luckily, most latent sector errors affect sectors in isolation, and therefore any of the pyramid codes discussed will do away with the worst tail latencies. Data accesses are not uniform and most disk reads will be to previously read data. Since bit rot (a good sector becomes unreadable) is extremely rare, data once read can be read again without running into a latent sector error. We conclude that it is sufficient to choose a pyramid code with a single exclusive-or parity for the small segment. We then have to choose for the method of generating parity for the large segment, namely between a linear MDS code or the interleaved parity code. Since we recommend making the large segment truly large, the additional storage overhead of the interleaved parity code needed to obtain the same erasure-correcting capacity as a linear MDS code becomes insignificant. It seems to us that this code reconciles best the divergent goals of an intradisk redundancy code.

5 RECOMMENDATIONS

Our experiments show that we cannot expect to control latent sector failures with intra-disk redundancy completely. It is however possible to reduce the occurrence of incorrigible latent sector errors to tolerable levels. We base this assessment on the expectation that our extrapolated susceptible disk present worst cases that are not typical for actual disk families. We have also seen that changing the parameters of a code can deal with susceptibility, namely one can lower the small segment size and one can increase the number of parities per large segment.

For disks that are components of a larger storage system, inter-disk redundancy also protects against latent sector errors. Plank, Blaum, and Schwartz as well as Li and Lee discuss codes appropriate for this dual purpose [4, 12, 13, 16]. In this setting, the purpose of intra-disk redundancy is mainly to allow fast recovery from small bursts and most prominently, single block failure. It remains to be investigated whether intra-disk redundancy can allow the storage system designer to lower the inter-disk redundancy. For example, instead of using RAID Level 6, one might be able to use RAID Level 5.

The results of Schroeder and colleagues [19] indicate that bit-rot is not an important cause of latent sector failures, or, in other words, that disk blocks do not go bad on their own. Read-after-Write is therefore the single most effective protection [18, 22]. As a burst of write requests renders this strategy difficult, this still leaves the need for intra-disk redundancy. The peculiarities of shingled write technology however gives the Read-after-Write strategy another opportunity. In order to retrieve freed blocks, a shingled write disk needs to use garbage collection. A band that contains freed blocks can be rewritten, skipping of course over the freed blocks. The new band can then be verified before the old one is declared to be unallocated. The verifying read after the write of the new band can be delayed as long as the old band is not overwritten. This strategy is successful in limiting intra-disk redundancy to bands containing new user data, lowering its costs.

Finally, the necessary strength of the code depends heavily on the prevalence of latent sector failures. Naturally enough, a disk without failures does not need protection. As the results of Schroeder [19] do not indicate disk age as an important cause for the prevalence of latent sector failures, we can *learn* by observations from an individual disk whether the disk family is prone to latent sector errors. The lone disks needs to use periodic scrubbing and can use the number of discovered (and hopefully recovered latent sector errors) as an indicator of the proneness. A very simple decision tree can use the extrapolated number of latent error sectors and the expected burst length to change code parameters. For instance, if we encounter a large number of unreadable blocks, then we could subdivide small segments in a pyramid code or increase the number of parity blocks for the large segment. Similarly, if we stop finding latent sector errors, then we can increase the size of small segments or decrease the number of parity blocks for the large segment. We leave it to future work to ascertain whether there are good predictor for the tail-heaviness of the burst length distribution.

6 CONCLUSIONS AND FUTURE WORK

We investigated the protection given by various forms of intra-disk redundancy in the context of shingled write disks. No code can guarantee protection against data loss due to latent sector errors. However, a pyramid code using only exclusive-or operations such that each data sector contributes to only two parity blocks, combines excellent protection with slight overhead.

Future work needs to obtain data on latent sector failures in shingled write disks and confirm whether Schroeder's results are still valid. With this data, we can then validate the recommendations of this article. In addition, we can also evaluate the codes that combine inter-disk and intra-disk redundancy for larger storage systems. We can also then validate the possibility of having a lone disk adapt to the observed number of unreadable blocks.

Acknowledgment. I would like to thank my shepherd, Keith Smith, Netapp, for the many helpful comments in improving the presentation of the paper.

REFERENCES

[1] Ahmed Amer, Darrell D.E. Long, Ethan L. Miller, J.-F. Paris, and Thomas Schwarz SJ. 2010. Design issues for a shingled write disk system. In *26th Symposium on Mass Storage Systems and Technologies (MSST)*. IEEE, Santa Clara, CA, 1–12.

[2] George Amvrosiadis, Alina Oprea, and Bianca Schroeder. 2012. Practical scrubbing: Getting to the bad sector at the right time. In *42nd Annual International Conference on Dependable Systems and Networks (DSN)*. IEEE / IFIP, 1–12.

[3] Lakshmi N Bairavasundaram, Garth R Goodson, Shankar Pasupathy, and Jiri Schindler. 2007. An analysis of latent sector errors in disk drives. In *SIGMETRICS Performance Evaluation Review*, Vol. 35(1). ACM, 289–300.

[4] Mario Blaum, James S. Plank, Moshe Schwartz, and Eitan Yaakobi. 2014. Partial MDS (PMDS) and sector-disk (SD) codes that tolerate the erasure of two random sectors. In *International Symposium on Information Theory (ISIT)*. IEEE, 1792–1796.

[5] Peter Corbett, Bob English, Atul Goel, Tomislav Grcanac, Steven Kleiman, James Leong, and Sunitha Sankar. 2004. Row-Diagonal Parity for Double Disk Failure Correction. In *Proceedings of the 3rd USENIX Conference on File and Storage Technologies*. USENIX Association, 1–14.

[6] Ajay Dholakia, Evangelos Eleftheriou, Xiao-Yu Hu, Ilias Iliadis, Jai Menon, and KK Rao. 2006. Analysis of a new intra-disk redundancy scheme for high-reliability RAID storage systems in the presence of unrecoverable errors. In *ACM SIGMETRICS Performance Evaluation Review*, Vol. 34. ACM, 373–374.

[7] Ajay Dholakia, Evangelos Eleftheriou, Xiao-Yu Hu, Ilias Iliadis, Jai Menon, and KK Rao. 2008. A new intra-disk redundancy scheme for high-reliability RAID storage systems in the presence of unrecoverable errors. *ACM Transactions on Storage (TOS)* 4, 1 (2008), 1.

[8] Simon Greaves, Yasushi Kanai, and Hiroaki Muraoka. 2009. Shingled Recording for 2–3 Tbit/in^2. *IEEE Transactions on Magnetics* 45, 10 (2009), 3823–3829.

[9] Cheng Huang, Minghua Chen, and Jin Li. 2013. Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems. *ACM Transactions on Storage (TOS)* 9, 1 (2013), 3.

[10] Ilias Iliadis, Robert Haas, Xiao-Yu Hu, and Evangelos Eleftheriou. 2008. Disk scrubbing versus intra-disk redundancy for high-reliability raid storage systems. In *SIGMETRICS Performance Evaluation Review*, Vol. 36(1). ACM, 241–252.

[11] Hannu H Kari. 1997. *Latent sector faults and reliability of disk arrays*. Ph.D. Dissertation. Helsinki University of Technology Espoo, Finland.

[12] Mingqiang Li and Patrick PC Lee. 2014. Stair codes: A general family of erasure codes for tolerating device and sector failures. *ACM Transactions on Storage (TOS)* 10, 4 (2014), 14.

[13] Mingqiang Li and Patrick PC Lee. 2014. STAIR codes: a general family of erasure codes for tolerating device and sector failures in practical storage systems.. In *USENIX Conference on File and Storage Technologies, (FAST)*. 147–162.

[14] Yin Li, Hao Wang, Xuebin Zhang, Ning Zheng, Shafa Dahandeh, and Tong Zhang. 2017. Facilitating Magnetic Recording Technology Scaling for Data Center Hard Disk Drives through Filesystem-Level Transparent Local Erasure Coding.. In *USENIX Conference on File and Storage Technologies, (FAST)*. 135–148.

[15] Alina Oprea and Ari Juels. 2010. A Clean-Slate Look at Disk Scrubbing.. In *USENIX Conference on File and Storage Technologies (FAST)*. 57–70.

[16] James S Plank and Mario Blaum. 2014. Sector-disk (SD) erasure codes for mixed failure modes in RAID systems. *ACM Transactions on Storage (TOS)* 10, 1 (2014), 4.

[17] James S Plank, Kevin M Greenan, and Ethan L Miller. 2013. Screaming fast Galois field arithmetic using Intel SIMD instructions.. In *USENIX Conference on File and Storage Technologies*. 299–306.

[18] Alma Riska and Erik Riedel. 2008. Idle Read After Write-IRAW. In *USENIX Annual Technical Conference*. 43–56.

[19] Bianca Schroeder, Sotirios Damouras, and Phillipa Gill. 2010. Understanding latent sector errors and how to protect against them. *ACM Transactions on Storage (TOS)* 6, 3 (2010), 9.

[20] Thomas J.E. Schwarz, Qin Xin, Ethan L. Miller, Darrell D.E. Long, Andy Hospodor, and Spencer Ng. 2004. Disk scrubbing in large archival storage systems. In *12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS 2004)*, IEEE (Ed.). 409–418.

[21] Y. Shiroishi, K. Fukuda, I. Tagawa, H. Iwasaki, S. Takenoiri, H. Tanaka, H. Mutoh, and N. Yoshikawa. 2009. Future options for HDD storage. *Transactions on Magnetics* 45, 10 (2009), 3816–3822.

[22] John Tillson. 1999. Disk drive incorporating read-verify after write method. (Aug. 24 1999). US Patent 5,941,998.

[23] Vinodh Venkatesan and Ilias Iliadis. 2013. Effect of latent errors on the reliability of data storage systems. In *21st International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS)*, IEEE (Ed.). 293–297.

[24] Roger Wood, Mason Williams, Aleksandar Kavcic, and Jim Miles. 2009. The feasibility of magnetic recording at 10 terabits per square inch on conventional media. *IEEE Transactions on Magnetics* 45, 2 (2009), 917–923.