RJ10352 (A0507-015) July 12, 2005
Computer Science

IBM Research Report

HoVer Erasure Codes for Disk Arrays

James Lee Hafner
IBM Research Division
Almaden Research Center
650 Harry Road
San Jose, CA 95120-6099

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

HOVER ERASURE CODES FOR DISK ARRAYS

James Lee Hafner

IBM Research
Almaden Research Center
650 Harry Road
San Jose, CA 95120
hafner@almaden.ibm.com

ABSTRACT: We present a new family of XOR-based erasure codes primarily targeted for use in disk arrays. These codes have a unique data/parity layout that gives rise to the name HoVer codes. We give constructions that tolerate up to four disk failures. Though the codes are only approximately maximum distance separable (MDS), they have performance advantages over other codes at many common array sizes. In addition, they have fewer parameter constraints than many other codes which enables greater choices and flexibility in efficiency and performance trade-offs.

Contents

1 Introduction **2**
- 1.1 Vocabularly And Notations . . . 3
- 1.2 Related Work – Other Codes . . . 4

2 HoVer Codes – Data/parity Layout **5**
- 2.1 Comparison to Nanda and Samsung . . . 7
- 2.2 Generic Parameter Restrictions . . . 7

3 Constructions **10**
- 3.1 Two Fault Tolerance . . . 11
  - 3.1.1 Rotating The Parity Row . . . 14
- 3.2 Three Fault Tolerance . . . 16
  - 3.2.1 Two V-Parity Rows . . . 16
  - 3.2.2 One V-Parity Row . . . 21
- 3.3 Higher Fault Tolerance . . . 24

4 Features **24**
- 4.1 Storage Efficiency . . . 25
- 4.2 Short Write IO Costs . . . 27
- 4.3 Reconstruction Costs . . . 27
- 4.4 Extended Fault Tolerance . . . 29
- 4.5 Parameter Flexibility . . . 29

5 Open Problems **30**

6 Summary **30**

7 Acknowledgements **30**

A Partial Proof Of Theorem 5 **32**

1. Introduction

Disk array systems have, for the most part, depended on the fault tolerance of RAID5 to provide reliability for customer data in the presence of single-point-of-failure models. Continuing increases in disk capacity along with little improvement in constant bit error rate has reduced the overall reliability of RAID5 systems to unacceptable levels. (Other factors also contribute as is noted in [5].) Many erasure codes have been proposed over the last 40 years (and the last 15 years in particular – see Section 1.2) in anticipation of such a need (and also for their purely scientific and theoretical value). However, none of the proposed codes has become even a *de facto* standard in the storage industry. Each code requires some trade-offs and may not be suitable within a given system's constraints and requirements.

In this paper we present a new family of XOR-based erasure codes for storing data in a disk array or other reliable storage system. We call these codes HoVer codes because the unique data and parity layout has both "horizontal" and "vertical" characteristics not found in (almost) any other code. Within a stripe, most codes either place all the parity on separate disks from the data (that is, horizontally aligned across the disks) or they place parity on the same disks with the data of the stripe (that is, vertically aligned within the disks). Reed-Solomon [10] and EVENODD [2] are examples of horizontal codes, whereas the X-code [12] and ZZS [13] are examples of vertical codes. The HoVer codes introduced in this paper have parity in both horizontal and vertical positions giving rise to their name.

The unique layout of HoVer codes provides some advantages over exclusively horizontal or vertical codes. These advtanges are detailed in Section 4. One key disadvantage of these new codes is that they are *not* MDS codes (in coding theory, MDS stands for "maximum distance separable" which for our purposes implies maximum storage efficiency for the given fault tolerance). However, as we will show, the unique layout provides for a range of implementation options that cover a large portion of the performance/efficiency trade-off space.

We provide four different constructions of HoVer codes (see Section 3), that can tolerate two, three (two constructions) and four disk failures. Our 2-fault tolerant construction is completely described theoretically. For the first of our 3-fault tolerant constructions, we provide some "negative" theoretical results (parameter choices which fail to provide a valid code) and some incomplete "positive" results. However, we have done extensive experiments (with array sizes into the 300s), so for practical purposes, our results are more than adequate. The other 3-fault tolerant code is given only by example, but it is clear that there is a theoretical relationship between the two 3-fault tolerant constructions. Finally, our 4-fault tolerant construction is presented only briefly; we have not studied this construction in much detail. The possibility of

other constructions within the HoVer layout are open questions, though we mention some alternatives in passing.

The paper is organized as follows. Our next subsection contains vocabularly and notation. We conclude the introduction with remarks on related work. Section 2 contains a description of the HoVer codes in general, with their unique data/parity layout and very general theorems on constructions. We next describe in detail the constructions we have discovered so far; in particular, Section 3.1 contains the complete theory of our 2-fault tolerant construction. In Section 4 we detail some of the key advantages and trade-offs of the HoVer codes. Open problems are briefly outlined in Section 5. Finally, we close with a brief summary and acknowledgements.

1.1. Vocabularly And Notations

We present our terminology and notation here for convenience and to prevent confusion as many terms have inconsistent usage in the literature.

element: a fundamental unit of data or parity; this is the building block of the erasure code. In coding theory, this is the data that is assigned to a bit within a symbol. For XOR-based codes, this is typically a set of sequential sectors and is the maximally sized data unit of an XOR formula.

stripe: a complete (connected) set of data and parity elements that are dependently related by parity computation relations. In coding theory, this is a code word; we use "code instance" synonymously.

strip: a unit of storage consisting of all continguous elements (data, parity or both) from the same disk and stripe. In coding theory, this is associated with a code symbol. It is sometimes called a stripe unit. The set of strips in a code instance form a stripe. Typically, the strips are all of the same size (contain the same number of elements).

vertical code: an erasure code in which a (typical) strip contains both data elements and parity elements (e.g., X-code [12]).

horizontal code: an erasure code in which each strip contains either data elements or parity elements, never both (e.g., EVENODD [2]).

array A collection of disks on which one or more instances of a RAID erasure code is implemented.

The number of elements in a strip is refered to as the number of "rows" in the code (in coding theory, this is the number of bits per symbol). A code with more than one row is called two-dimensional. This is clarified in Section 2. The array size is the total number of strips in a stripe. We use the abbreviations v-parity and h-parity for "vertical parity" (aligned vertically with respect to data elements in a stripe) and "horizontal parity" (aligned horizontally with respect to data elements in a stripe).

1.2. Related Work – Other Codes

There are many erasure codes that have been presented in the literature for application to disk arrays. For ease of comparison to HoVer codes, we divide the set of known erasure codes into different categories and give (non-exhaustive) examples in each category. In a category by themselves are the Reed-Solomon codes [10], which are MDS but require complex finite field arithmetic. Second are XOR-based codes that are MDS. These come in two types: vertical codes such as the X-code [12], BCP [1] or ZZS codes [13] and horizontal codes such as EVENODD [2, 3], Blaum-Roth [4], or Row-Diagonal Parity [5]. Finally, there are non-MDS codes that are XOR-based. These subdivide into three categories based on efficiency. The Gibson *et al* codes [6] with efficiency larger than 50%, codes with efficiency 50% exactly such as the LSI [11], the Blaum-Roth [4] 3-fault tolerant code and some of the WEAVER codes [7], and N-way mirroring (trivially XOR-based) with efficiency less than 50%. There are WEAVER codes [7] of efficiency less than 50% but we ignore them here.

With the exception of the Reed-Solomon codes, N-way mirroring and some WEAVER codes, none of the codes have exceptionally high fault tolerance. There are variations of EVENODD [3] that are 4 fault tolerant; the Blaum-Roth [4] binary code of efficiency 50% and two codes in [6] are 3 fault tolerant. As far as we know, none of the other codes have variants that can tolerate more than 2 failures.

Throughout the paper, we will highlight the features (both positive and negative) of the HoVer codes when compared against these categories (typically by specific example). We disregard a comparison with Reed-Solomon, since we are concerned primarily with XOR-based codes. Similarly, since the HoVer codes are nearly MDS (but not quite – see Section 4.1), we do not consider comparisons with the last class of non-MDS, low efficiency codes, with the following exception. The WEAVER codes provide a performance advantage over MDS codes because they constrain the number of data elements that touch each parity (so-called parity in-degree). This in turn implies a bound on efficiency (50% and below). It is remarked in [7] that WEAVER codes provide an efficiency/performance trade-off for high fault tolerant codes, interpolating (at the low end of the efficiency range) between Reed-Solomon and N-way mirroring. The HoVer codes provide a similar efficiency/performance trade-off on

the upper end (above 50%) of the efficiency range. MDS codes have large parity in-degree for all parity elements. HoVer codes have the feature that some parity elements have small parity in-degree and others large parity in-degree. Large parity in-degree typically reflects better storage efficiency, whereas smaller parity in-degree typically reflects better performance (particularly for rebuild or reconstruction). This asymmetry of the HoVer codes is a key feature that enables their performance/efficiency trade-offs.

There are two other codes that deserve mention, both from the patent literature. These are the Nanda code [9] and the Samsung DH2 code [8]. These both have a layout similar to HoVer (see the next section). We discuss their relationship to HoVer codes in what follows. The other codes in [8] have no specific h-parity strips; they can be viewed as codes analogous to a non-MDS X-code, so are considered in the category of purely vertical codes above.

2. HoVer Codes – Data/parity Layout

The HoVer codes are two-dimensional codes: the diagram in Figure 1 shows their general data/parity layout. We parameterize the HoVer codes as $\text{HoVer}^t_{v,h}[r,n]$ where t is the fault tolerance, v is the number of rows of v-parity elements (exclusively), h is the number of strips of h-parity elements (exclusively), r is the number of data elements per strip (not on an h-parity strip), and n is the number of strips containing data elements. The user data substripe is the $r \times n$ upper-left block with $X(i,j)$ entries. By abuse of terminology, when we refer to a "data strip" we mean a strip with both data and parity, i.e., a strip which is *not* an h-parity strip.

	n-data disks				h h-parity		
r data rows	$X(0,0)$	$X(0,1)$	$\cdots$	$X(0,n-1)$	$H(0,0)$	$\cdots$	$H(0,h-1)$
	$X(1,0)$	$X(1,1)$	$\cdots$	$X(1,n-1)$	$H(1,0)$	$\cdots$	$H(1,h-1)$
	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$
	$X(r-1,0)$	$X(r-1,1)$	$\cdots$	$X(r-1,n-1)$	$H(r-1,0)$	$\cdots$	$H(r-1,h-1)$
v v-parity rows	$V(0,0)$	$V(0,1)$	$\cdots$	$V(0,n-1)$	$\square$	$\cdots$	$\square$
	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$
	$V(v-1,0)$	$V(v-1,1)$	$\cdots$	$V(v-1,n-1)$	$\square$	$\cdots$	$\square$

Figure 1: General data layout for $\text{HoVer}^t_{v,h}[r,n]$ codes. $X(i,j)$ represents data symbols, $H(i,j)$ and $V(i,j)$ represent horizontal and v-parity symbols, respectively. The $\square$ symbol represents unused blocks or possibly other h-parity elements.

For symmetry reasons, the $\square$ elements in the array are ideally left unused. This space on disk can be ignored (left blank) or used for meta-data about the stripe or array. Alternatively, with the appropriate rotations, packing and logical addressing, this space can be occupied by elements of other stripes in the array, thereby not wasting the physical space. In the first case, this block is a penalty in the efficiency formulas, whereas in the second it is not (see Section 4.1). It is possible to also use these spaces for additional data or parity elements, but we do not consider such constructions here. However, since we have v and h small (see equation (2.1)), wasting this space is insignificant.

Clearly, any erasure code can be put in this form, with perhaps an additional set of strips that contain data exclusively. Purely horizontal codes have $v = 0$ and purely vertical codes have $h = 0$. As examples, RAID4 is a $\text{HoVer}^1_{0,1}[1, n]$ code, EVENODD is $\text{HoVer}^2_{0,2}[p-1, p]$, the X-code is $\text{HoVer}^2_{2,0}[p-2, p]$ but the ZZS code with $v = 1, h = 0, t = 2, r = (p-3)/2, n = p-1$ requires an additional data-only strip with $r+1$ data elements. (We use the symbol p exclusively to represent a prime number.)

The novelty of the HoVer codes is when both v and h are non-zero, so we reserve the name HoVer for these cases. In all our constructions, each data element "touches" (or contributes to the parity formula) for exactly t parity elements. Erasure codes with this property are optimal for t fault tolerance. With this restriction, we also make the assumption

$$v + h \leq t \tag{2.1}$$

since the most efficient use of the parity element space puts each data element in each parity strip and each parity row at least once (for a total of exactly t times). Consequently, a true HoVer code must have $t \geq 2$ (the interesting case!).

This general description of HoVer codes is incomplete since we have only specified where data and parity elements are placed on the strips; we have not specified how the parity values are computed. It turns out that only certain combinations of the parameters can provide for a valid code *regardless* of how the parity values are computed. In the next subsection, we discuss such necessary conditions for HoVer codes. In Section 3 we give formal definitions for parity value XOR formulas that yield valid codes under certain additional necessary and in some cases sufficient conditions.

In addition, we assume that the first h-parity strip is the simple XOR of the data substrips (that is, we assume a RAID4 subcode). So, for $0 \leq i \leq r-1$, we set

$$H(i, 0) = \bigoplus_{j=0}^{n-1} X(i, j). \tag{2.2}$$

This is not strictly required but is a reasonable assumption in practice. When there is only one h-parity strip, we abbreviate $H(i) = H(i, 0)$.

2.1. Comparison to Nanda and Samsung

We can now draw comparisons of the HoVer codes with the Nanda [9] and Samsung DH2 [8] codes which have a layout as in Figure 1. Both of these codes are only 2-fault tolerant, in constrast to the HoVer constructions of 3 and 4-fault tolerance. Both codes have an h-parity strip defined by (2.2).

In the Nanda construction $h = 2$ and $v = 1$, and the diagonal parities are computed for all diagonals in the array *without* wrap around. This set of parities is layed out in order down the second h-parity strip, then along the parity row until all the diagonal parities have been placed. This has two consequences. First it requires more strips/rows for parity placement than are strictly necessary for 2-fault tolerance (i.e., it does not satisfy the bound in (2.1)). Second, there is non-uniform distribution of parity in-degrees for the diagonal parity elements because there is no wrap-around (e.g., the first and last diagonal parity elements are simple mirrors of the upper left and lower right corners of the data element subarray, respectively). For HoVer codes, we desire condition (2.1) and better symmetry in the parity formulas.

The Samsung DH2 code is, in fact, a special case of the $\text{HoVer}^2_{1,1}$ construction that we give in Section 3.1. Though we discovered the HoVer codes independently of the Samsung work (we discovered this patent only after completing our initial study), the DH2 was in fact discovered earlier, so it is perhaps more accurate to say that the $HoVertvh$ codes (and $\text{HoVer}^2_{1,1}$ codes in particular) are broad generalizations of this Samsung DH2 code. The DH2 code is limited only to $n = p$, a prime and $r = p - 2$, whereas our constructions, even for $t = 2$, $h = v = 1$, remove both of these constraints. More remarks on this code are given in Section 3.1.

2.2. Generic Parameter Restrictions

In this section we prove some simple necessary conditions on the parameters of HoVer codes as well as some other useful results. Our first theorem shows a fundamental necessary condition for any HoVer code to achieve the specified fault tolerance. It is independent of the XOR formulas and depends only on the data/parity layout of Figure 1.

Theorem 1. *For any* $\text{HoVer}^t_{v,h}[r, n]$ *code the following relationship must hold:*

$$r \leq v\frac{n-t}{t-h}.$$

or equivalently,

$$n \geq r\frac{t-h}{v} + t,$$

Proof. The proof of this is a simple dimensionality argument. The generator matrix for this code has dimensions $rn \times (r+v)(n+h)$ where we place zeros in hv columns (one set of v columns of zeros after each set of r columns of h-parity) to simulate the $\square$ subarray of the stripe (see Figure 1). The columns are grouped in blocks of $r+v$ to simulate data layout on the disks. That is, each block of $r+v$ columns defines a strip on a disk (or a code symbol). Such a block defines a failure domain. The generator matrix with this block structure defines a set of equations that map the user data values to the data/parity layout.

If the code has fault tolerance t, then it must be able to tolerate the loss of t strips. From the generator matrix perspective, this can be simulated by removing t of the blocks of $r+v$ columns. The remaining matrix defines a set of equations that map data to the remaining (known) data and parity in the stripe. If this matrix is full row rank (using modulo 2 or binary arithmetic), then the system of equations is solvable and the lost data can be recovered (and so any lost parity as well).

For the matrix to be full row rank, it must be the case that the number of *non-zero* columns in the reduced matrix must be at least equal to the number of rows. The number of rows is rn. The number of non-zero columns is at least $(r+v)(n+h) - t(r+v) - hv$; we substract the term hv assuming that no h-parity blocks are removed (these blocks have v columns of zeros in each block). So, we require,

$$rn \leq (r+v)(n+h) - t(r+v) - hv. \tag{2.3}$$

By isolating r and then n in this inequality, we prove the inequalities in the theorem. $\square$

Inequality (2.3) implies that for purely horizontal codes there is no *generic* restriction on the number of rows (only that $h \geq t$, so $h = t$ by (2.1); i.e., there is enough parity columns for the required fault tolerance) – this is the case for RAID4. Other restrictions on r and n for horizontal codes (for example, see [2, 4, 5]) come from considerations other than dimension (the same holds for certain HoVer codes as we show later). For purely vertical codes of fault tolerance 2, we deduce $r + v \leq vn/2$, which is tight for the X-code (with $v = 2$, provided n is prime) and BCP codes [1] (with $v = 1$, provided n is even).

For HoVer codes in general, this inequality in Theorem 1 is not a sufficient condition for a valid code. However, we make the following conjecture based on our computer searches:

Conjecture 1. *If n is prime and greater than t, then there exists a* $\mathrm{HoVer}^t_{v,h}[r,n]$ *code for* every *pair $[r,n]$ satisfying the condition of Theorem 1.*

More generally, for any n sufficiently large, there is a number $r_0 = r_0(n, t, h, v)$ so that there exists a HoVer code provided $r \leq r_0$.

Alternatively, for any r, there is a number $n_0 = n_0(r, t, h, v)$ so that there exists a HoVer code provided $n \geq n_0$.

The first part of the conjecture states that the inequalities in Theorem 1 are both necessary and sufficient when n is prime. As we show in Section 3.1, the conjecture holds for $t = 2$ and $v = h = 1$. More refined conjectures and more numerical evidence is given for $t \geq 3$ in Sections 3.2 and 3.3.

On the other hand, we prove the following theorem related to the last two statements of the conjecture in the special case that $h = 1$.

Theorem 2. *Suppose there exists a* $\text{HoVer}^t_{v,1}[r_0, n]$ *for some fixed choice of* $v \geq 1$, $t \geq 2$, n *and* r_0. *Then there exists a* $\text{HoVer}^t_{v,1}[r, n]$ *code for every* $1 \leq r \leq r_0$.

Proof. For any $r \leq r_0$, we can construct the required $\text{HoVer}^t_{v,1}[r, n]$ simply by logically zeroing any $r_0 - r$ rows of data and the corresponding h-parity elements. □

This theorem implies that HoVer codes (with $h = 1$) have great flexibility in their array parameters, particularly, in the number of rows of data elements *provided* some construction works at all. We show in Section 4 that large r implies better efficiency, but small r implies better performance. So having this range for r enables a wide (and continuous) range of choices in the performance/efficiency trade-off space for any given array size. For fixed n (and v, t), our goal is to find the largest value $r_0 \leq v(n-t)/(t-1)$ that supports a given construction.

Conversely, we will see that if a construction works for some value of n_0 (for fixed r), then it does not necessarily work for all values of $n \geq n_0$. That is, there can be (and are) gaps in the sequence of valid n for a given r, but it is "continuous" after some n_0 sufficiently large.

We contrast the statement of this theorem with a typical purely horizontal code with $t \geq 2$. As an example, consider the EVENODD code [2]. In this case, the $r = p - 1$ and it must be the case that $n \leq p - 2 = r + 1$. That is, for fixed n, there is a *lower* bound on the number of rows in such codes. Conversely, for fixed r, there is an *upper* bound on n. (These two bounds are the reverse of the bounds for HoVer codes.) This analysis applies to essentially every horizontal code, including Reed-Solomon [10]. See Section 4 for additional comments.

So far, we have not given any specification of how parity is computed in the HoVer codes. Beyond the above theorems, we know very little in the general case. In the next section, we provide details of our known constructions.

3. Constructions

In the next few subsections, we give some constructions for HoVer codes with fault tolerance two, three (two constructions) and four, respectively. For 2-fault tolerance, we give one construction and provide necessary and sufficient conditions on the parameters. For the other constructions, we have extensive experimental results, some necessary conditions for the parameters and partial proofs of sufficient conditions. In some cases (unfortunately), we do not even have good conjectures. The patterns and mathematical analysis in these cases are quite complex, even though the constructions are very simple. However, our experiments are sufficient to cover practical ranges of the parameters.

Our experiments are performed by exhaustively checking all possible failure scenarios for a presumed fault tolerance t. The following pseudo-code describes the procedure:

1. Compute the generate matrix for the given design parameters in column block form (as in the proof of Theorem 1).
2. For each t element subset T of the set $\{0, 1, \ldots, N-1\}$ of strip labels, perform the following:
   (a) Remove the blocks from the generator matrix indexed by the elements of T (simulate disk/strip loss).
   (b) Compute the row rank of the reduced matrix (under binary arithmetic).
   (c) If the row rank equals the number of rows, continue; else return "Invalid".
3. Return "Valid".

As in the proof of Theorem 1, each reduced matrix represents a set of equations that map the "unknown" data values (as variables) into the "known" data and parity (those that have not been lost). The row rank of the reduced matrix equals the number of rows if and only if the system of equations is solvable. Consequently, the specific generator matrix defines a valid t fault tolerant code if and only if the loop completes. Note that there are $\binom{N}{t}$ subsets T to consider, which can grow quite large for high fault tolerance and/or large array size. This methodology can be applied to *any* erasure code whatsoever. Symmetries and other specific properties of codes (as in our HoVer constructions), can reduce this combinatorial complexity. Such details are beyond the scope of this paper, however.

3.1. Two Fault Tolerance

In this section we provide the complete theory of a family of $\mathrm{HoVer}^2_{1,1}[r,n]$ codes (2-fault tolerant with one v-parity row and one h-parity strip). Our theorems provide necessary and sufficient conditions for r and n. We also analyze a variation on the basic code which has some additional performance advantages.

We need some notation. Let $\mathrm{mod}_n(k)$ be the value of $(k \bmod n)$ in the set $\{0,\ldots,n-1\}$. Also, let $\mathrm{pr}(n)$ be the smallest prime divisor of n. For example, if n is prime, then $\mathrm{pr}(n)=n$; if n is even, then $\mathrm{pr}(n)=2$. For later use, we let $\mathrm{pr}_2(n)$ be the second largest distint prime factor of n; for a prime or prime power, $n=p^k$, we set $\mathrm{pr}_2(n)=\infty$.

Relabel the entries in the first (and only) row $V(0,j)$ of v-parity by $U(j)$ and define them by the simple equation:

$$\begin{aligned} U(j) &= \bigoplus_{k=0}^{r-1} X(r-1-k, \mathrm{mod}_n(j+k+1)) \\ &= \bigoplus_{k=0}^{r-1} X(k, \mathrm{mod}_n(j+r-k)). \end{aligned} \tag{3.1}$$

Visually, this is parity computed along the up-diagonal (hence the symbol $U(j)$ for "up") with wrapping from left to right (modulo n). Each diagonsl starts on the next disk to the right (with wrap-around) of the parity disk. The following graphic shows the example when $r=4, n=7$:

3	4	5	6	**0**	1	2	$H(0)$
4	5	6	**0**	1	2	3	$H(1)$
5	6	**0**	1	2	3	4	$H(2)$
6	**0**	1	2	3	4	5	$H(3)$
U(0)	$U(1)$	$U(2)$	$U(3)$	$U(4)$	$U(5)$	$U(6)$	

Data elements are labeled by the index (strip number) of their v-parity; the diagonal for $U(0)$ is highlighted (in bold).

Note that the definition has the up-diagonal parity lines wrapping from right-to-left similar to the X-code [12] (because of the mod_n in (3.1)). There is no wrapping from top to bottom as common in horizontal codes such as EVENODD [2] or RDP [5] codes.

With this definition of the v-parity, we have the following result:

Theorem 3. *The* $\mathrm{HoVer}^2_{1,1}[r,n]$ *code with v-parity defined by (3.1) has fault tolerance two if and only if the following relation holds:*

$$r \le n - n/\mathrm{pr}(n) - 1.$$

We give the proof below, but first give some examples. If n is prime, then $\mathrm{pr}(n) = n$ so that this relation is just $r \leq n-2$. This is the necessary condition of Theorem 1 and shows that Conjecture 1 holds in this special of $t = 2$ and $v = h = 1$.

When $n = p$ and $r = n - 2 = p - 2$, formula (3.1) defines a code identical to the Samsung DH2 [8] code. The construction given above generalizes the Samsung construction to n composite and also allows for variability in the choice of r. In the next section, we alter these formulas just a bit and gain a performance advantage over the Samsung DH2 code even in the case $n = p$ and $r = p - 2$.

As we will show in Section 4.1, larger admissible values for r yield better efficiency. Consequently, this is optimal when n is prime. When n is even, $\mathrm{pr}(n) = 2$, we get the other extreme: the relation of Theorem 3 is that $r \leq n/2 - 1$. Odd values of n support a larger maximum value for r in relation to n.

Proof. We have to show that we can reconstruct lost data (and parity) when any one or two strips are lost (e.g., because their disk(s) fail). There are two cases of single strip loss: (a) the strip is the h-parity strip or (b) the strip is a data/parity strip. In the first case, no data is lost so the parity can be recomputed by reading the data in the stripe and recomputing. In the second case, we can recompute the lost data using either the h-parity or the v-parity. Using the horizontal parity, a total of rn elements are read from a total of n strips and each data reconstruction formula is an n-ary XOR. The lost v-parity reconstruction is an r-ary XOR, but requires no additional reads. On the other hand, using the v-parity each data element can be reconstructed from $r - 1$ data elements (from the diagonal) and its v-parity element (so an r-ary XOR). The v-parity is reconstructed as before. In this case, a total of $r(r + 1)$ elements are required from $\min(n - 1, 2r)$ strips and each reconstruction formula is an r-ary XOR. As $r \leq n - 2$, using the v-parity is more efficient in both IO and XOR costs (this exemplifies performance advantage for smaller values of r; see Section 4.3 for additional comments).

When two strips are lost, there are two cases:

Case 1: h-parity and one data/parity strip are lost. The v-parity algorithm above can be used to reconstruct the lost data/parity strip. Then the h-parity can be recomputed from the reconstructed data and the remainder of the data elements in the array. In this case, all the (good) data element substripe and most of the v-parity row needs to be read (though this is still not necessarily the entire good stripe).

Case 2: two data/parity strips are lost. By rotational symmetry (wrapping modulo n), we only need to consider the case where strip(0) and strip(Δ) are lost for $1 \leq \Delta \leq n/2$. View the v-parity element in column c as being in position (r, c) in the stripe. We need a lemma whose proof requires some very simple bookkeeping (which we leave to the reader):

Lemma 1. *For $1 \leq \Delta \leq n/2$, the diagonal through the element at position (x, c) intersects strip $c + \Delta$ at position $(x - \Delta, c + \Delta)$, provided $x - \Delta \geq 0$. Otherwise, it does not intersect at all.*

With this lemma, we can continue with the proof of Case 2. Consider a data element on one of the lost strips. It touches two parity elements, one horizontal and one vertical. The h-parity element is assumed to be good and the h-parity relation through the given data element must intersect the other lost strip. That is, this relation has two unknown data elements. If we reconstruct either of these elements, the other is immediately reconstructable. In contrast, the v-parity element touched by the given data element falls into three subcases: (a) it may be on the other lost strip, so is a lost parity element itself, or (b) it may be good (not on the other lost strip) and the diagonal parity relation (3.1) does not intersect the other lost strip, or (c) it may be good and the diagonal parity relation intersects the other lost strip. In subcase (a), the v-parity is useless for any reconstruction. In subcase (b), there is only one unknown in the parity relation (the given data element) and so the element can be reconstructed. In case (c), there is an h-parity element that leads back to the initial lost strip.

The h-parity and diagonal v-parity relations then build a set of zig-zag chains through the lost data elements. Each data element is of one of three types: (a) at the intersection of a horizontal segment and a diagonal segment whose endpoint is a lost v-parity, or (b) at the end-point of a horizontal segment and can be reconstructed from a diagonal v-parity, or (c) is an interior point on a zig-zag chain.

These chains partition the set of lost data elements. If a chain contains an element of type (b), all elements on its chain are reconstructable. If a chain contains no such elements (equivalently, the two endpoints must be of type (a)), then no element on the chain can be reconstructed. This is the case if and only if there is a chain that connects the two lost v-parity elements. If such a chain exists, it is unique.

Using the lemma, it is easy to see that the dependency chain from the lost parity on strip(0) contains:

$$\begin{aligned} U(0) &\rightarrow X(r - \Delta, \Delta) \\ &\rightarrow X(r - \Delta, 0) \\ &\rightarrow X(r - 2\Delta, \Delta) \\ &\rightarrow X(r - 2\Delta, 0) \\ &\rightarrow \cdots \\ &\rightarrow X(r - k\Delta, 0), \end{aligned}$$

provided $r - k\Delta \geq 0$. The chain from the lost v-parity on strip(Δ) starts with

$$U(\Delta) \rightarrow X(r - (n - \Delta), 0).$$

So, the system is unrecoverable if and only if there exists a number k such that

$$r - k\Delta = r - (n - \Delta) \geq 0. \tag{3.2}$$

Suppose the system is unrecoverable. Then such a k exists and we have

$$(k + 1)\Delta = n, \quad \text{and} \quad (n - \Delta) \leq r.$$

which implies that $\Delta | n$. We already have $\Delta \leq n/2$. The smallest value possible then for the $n - \Delta$ occurs when Δ is maximal, that is, when $\Delta = n/\operatorname{pr}(n)$. So, if the system is unrecoverable then $r \geq n - n/\operatorname{pr}(n)$ as claimed.

Finally, suppose $\Delta = n/\operatorname{pr}(n) \leq n/2$ and suppose that $r \geq n - \Delta$. Then $k = \operatorname{pr}(n) - 1$ satifies the conditions in (3.2) and the system is unrecoverable.

This completes the proof. □

The proof of Case 2 shows that the set of dependency chains give a complete description of the reconstruction process (partitioning and recursion rules). It is not hard to see that each chain has an even number of elements (they come in pairs connected by a horizontal segment). When a chain has two endpoints of type (b), half the lost data elements can be recovered from one end of the chain, and the other half from the other end. This chain splitting minimizes the recursion for a specific element and so reduces reconstruction costs. For all failures cases, except when two adjacent strips are lost ($\Delta = 1$), every chain is dual-ended.

In the exceptional case when $\Delta = 1$, there is only one chain and it has one type (b) endpoint (the other endpoint is of type (a)); the chain is maximally long, containing $2r$ elements. In all other cases, the chains can be split into at most r elements each (and sometimes as little as one element each). In the next section, we modify the construction so that every chain has two type (b) endpoints. This reduces the worst case reconstruction to two r-step recursions.

3.1.1. Rotating The Parity Row In the above construction, we have placed the v-parity on the strip immediately to the left of the diagonal defining it. The only obvious (necessary) requirement for the placement of the v-parity is that it is not placed on the same strip as any data it contains.

In addition, we saw that failure of two neighboring strips resulted in a single reconstruction chain of length $2r$. If we rotate the v-parity to the left (so that it is

separated from the dependent data by an additional strip, but subject to the necessary condition above), we find that there are always at least two chains. In other words, all chains have length at most r and are more efficient for reconstruction.

More generally, we can rotate the v-parity s steps to the left. The equations for the v-parity (with s-shifted placement) are now:

$$\begin{aligned} U_s(j) &= \bigoplus_{k=0}^{r-1} X(r-1-k, \operatorname{mod}_n(j+k+s)) \\ &= \bigoplus_{k=0}^{r-1} X(k, \operatorname{mod}_n(j+r-1-k+s)) \end{aligned} \tag{3.3}$$

(compare this to (3.1)). The subscript s indicates the number of positions the diagonal parity is shifted to the left before placement in its row, or equivalently, the offset from the parity element to the start of its up-diagonal. The equations in (3.1) have $s = 1$ (that is, $U(j) = U_1(j)$), so are placed on disk to the left of the starting data element. An example with $s = 2$, $r = 4$ and $n = 7$ is presented in the following diagram (where we leave off the h-parity since that is unchanged):

2	3	4	5	6	**0**	1
3	4	5	6	**0**	1	2
4	5	6	**0**	1	2	3
5	6	**0**	1	2	3	4
$\mathbf{U_2(0)}$	$U_2(1)$	$U_2(2)$	$U_2(3)$	$U_2(4)$	$U_2(5)$	$U_2(6)$

Unfortunately, such shifts reduce the range of r for which the code has 2-fault tolerance (and so reduces the maximal efficiency, see Section 4.1). The following theorem summarizes the situation.

Theorem 4. *The s-shift* $\mathrm{HoVer}^2_{1,1}[r, n]$ *code defined by (3.3) has fault tolerance two if and only if*

$$r \leq \begin{cases} n - s & \text{if } s > n/\operatorname{pr}(n) \\ n - n/\operatorname{pr}(n) - s & \text{otherwise} \end{cases}$$

We do not give a proof here. It follows along the same lines as the case of $s = 1$ given in the proof of Theorem 3, with the addition of some special cases when $s > n/\operatorname{pr}(n)$. We remark that for $n = p$, a prime, both the 1-shift and the 2-shift allow for $r \leq n-2$, so that there is no efficiency penalty for $s = 2$ even though we gain the reconstruction advantage of the 2-shift layout (and in particular, provides an advantage over the Samsung DH2 code [8])

3.2. Three Fault Tolerance

There are three natural extensions of the 2-fault tolerance code to 3-fault tolerance: (a) add an additional h-parity strip, $v = 1$ and $h = 2$; (b) add an additional v-parity row, $v = 2$ and $h = 1$; (c) pack parity elements more densely keeping $v = 1$ and $h = 1$. In the latter case, it only makes sense to pack into the v-parity row because the h-parity strip is a single failure domain. This denser packing is fundamentally how the ZZS [13] and BCP [1] codes (one parity row, 2-fault tolerant) differ from the X-code (two parity rows, 2-fault tolerant). We have not yet discovered an extension of type (a), though we expect that such codes exist, perhaps as extensions of Blaum-Roth [4] or EVENODD [2]. (The Nanda code [9] has $h = 2$ but is only 2-fault tolerant.)

In the next two sections, we give constructions when $h = 1$ for $v = 2$ (Section 3.2.1) and for $v = 1$ (Section 3.2.2). Our theory is weaker here than in the previous section, but our experimental constructions contain all array sizes of practical interest.

3.2.1. Two V-Parity Rows In this section we give a construction of a 3-fault tolerant code in the case of $v = 2$ and $h = 1$. Amongst the many possible choices, the most obvious seems to be to extend the $\mathrm{HoVer}^2_{1,1}[r, n]$ construction of the previous section by the addition of a row of v-parity computed via the "down-diagonals" in the array. This can be viewed as extending (a subcode of) the X-code to 3-fault tolerance by adding the h-parity strip (and dropping the primality constraint on the array size). Alternatively, one could define the v-parity row by lines of a different slope (see Section 3.2.1.1 where we consider alternative paths).

However, the obvious extension with the down-diagonal parity placed in the strip to the right of its last entry does not work. We need to provide for shifting left of the up-diagonals (as in Section 3.1.1) and shifting right of the down-diagonals as well in order to get the maximal range for r.

The following diagram shows an example with both up- and down-diagonals shifted by 2. In each cell the data subarray, the first number in the pair indicates the strip number of the up-diagonal that data element touches; the second number is the down-diagonal's strip number. We highlight the diagonal for $U_2(0)$ (in bold) and for $D_2(5)$ in underlined italics.

2:*5*	3:6	4:0	5:1	6:2	**0**:3	1:4
3:4	4:*5*	5:6	6:0	**0**:1	1:2	2:3
4:3	5:4	6:*5*	**0**:6	1:0	2:1	3:2
5:2	6:3	**0**:4	1:*5*	2:6	3:0	4:1
$\mathbf{U_2(0)}$	$U_2(1)$	$U_2(2)$	$U_2(3)$	$U_2(4)$	$U_2(5)$	$U_2(6)$
$D_2(0)$	$D_2(1)$	$D_2(2)$	$D_2(3)$	$D_2(4)$	$\underline{D_2(5)}$	$D_2(6)$

Mathematically, we compute the two parity rows via:

$$\begin{aligned} U_{s_0}(j) &= \bigoplus_{k=0}^{r-1} X(r-1-k, \mathrm{mod}_n(j+k+s_0)) \qquad (3.4) \\ &= \bigoplus_{k=0}^{r-1} X(k, \mathrm{mod}_n(j+r-1-k+s_0)), \\ D_{s_1}(j) &= \bigoplus_{k=0}^{r-1} X(r-1-k, \mathrm{mod}_n(j-k-s_1)). \qquad (3.5) \\ &= \bigoplus_{k=0}^{r-1} X(k, \mathrm{mod}_n(j-r+1+k-s_1)) \end{aligned}$$

The following symmetries are inherent in these formulas.

Proposition 1. *If the parameters* $[r, n; s_0, s_1]$ *provide a valid* $\mathrm{HoVer}^3_{2,1}[r, n]$ *code then the following sets of parameters do also:*

- $[r, n; s_1, s_0]$
- $[r, n; n+1-r-s_1, n+1-r-s_0]$
- $[r', n; s_0, s_1]$, *for any* $r' \leq r$

Furthermore, if $n - r < s_0 < n$ *or* $n - r < s_1 < n$, *then the code is invalid.*

Proof. The first comes from flipping the data and v-parity with respect to the vertical line throught the center of the data subarray and swapping the two v-parity rows; that is, $X(k,t) \to X(k, n-1-t)$, $U_s(k) \leftrightarrow D_s(n-1-k)$. The second is obtained by flipping the data symmetrically with respect to the horizontal line through the center of the data subarray and swapping the two v-parity rows; that is $X(k,t) \leftrightarrow X(r-1-k,t)$ and $U_s(k) \leftrightarrow D_s(k)$.

By the construction, $r' < r$ can be achieved by logically zero-ing the top $r - r'$ rows (including elements in the h-parity column). Since this does not change the validity of the code or the offsets, the parameters $[r', n, s_0, s_1]$ must also provide a valid code. This is a constructive version of a special case of Theorem 2.

If $n - r < s_0 < n$, then every up-diagonal parity element contains a data element from its own strip. In that case, the code is invalid. The same argument applies to s_1 and the down-diagonal parity elements. □

We do not have a general theorem that determines all combinations of parameters $[r, n; s_0, s_1]$ for which the formulas (3.4) and (3.5) provide valid codes. Recall that our goal is to determine, for a given n, the largest value of r for which some construction works (that is, the most storage efficient design). We performed extensive

computer searches over all possible choices of offsets and all $r \le n-3$ (the bound of Theorem 1). That lead to the following theorem and conjectures that (almost) complete the picture; they depend heavily on the small prime factors of n, analogous to Theorem 3.

We give some notation first. Let $\mathrm{r}_0(n,[s_0,s_1])$ be the largest value of r so that equations (3.4) and (3.5) define a valid $\mathrm{HoVer}^3_{2,1}[r,n]$ code with offsets $[s_0,s_1]$. Let $\mathrm{r}_0(n)$ be the maximum of $\mathrm{r}_0(n,[s_0,s_1])$ over all offsets $[s_0,s_1]$, that is, the best we can achieve with the up-diagonal and down-diagonal parity formulas (3.4) and (3.5). Recall that $\mathrm{pr}(n)$ is the smallest prime divisor and $\mathrm{pr}_2(n)$ is the second smallest distinct prime divisor of n (see the beginning of Section 3.1).

Theorem 5. *For $n \ge 4$ there are valid $\mathrm{HoVer}^3_{2,1}[r,n]$ codes in the following cases:*

- *if $n = 4, 6, 10$, then $\mathrm{r}_0(n) = \mathrm{r}_0(n,[1,2]) = n/2 - 1$ (these are the exceptional cases);*
- *if $\mathrm{pr}(n) = 2$ and $n = 8$ or $n \ge 12$ (not exceptional), then $\mathrm{r}_0(n) = \mathrm{r}_0(n,[1,2]) = n/2$;*
- *if $\mathrm{pr}(n) > 2$, then $\mathrm{r}_0(n,[1,2]) = (n-3)/2$;*
- *if $\mathrm{pr}(n) = 3$ and $\mathrm{pr}_2(n) = 5$, then $\mathrm{r}_0(n) \le 3n/5 - 3$;*
- *if $\mathrm{pr}(n) = 3$ and $\mathrm{pr}_2(n) > 5$, then $\mathrm{r}_0(n) \le 2n/3 - 2$;*
- *if $\mathrm{pr}(n) \ge 5$, then $\mathrm{r}_0(n) \le n - 2n/\mathrm{pr}(n) - 1$.*

Conjecture 2. *With the notation of the theorem above, the following should hold:*

- *if $\mathrm{pr}(n) = 3$ and $\mathrm{pr}_2(n) = 5$, then $\mathrm{r}_0(n) = \mathrm{r}_0(n,[1,3]) = 3n/5 - 3$.*
- *if $\mathrm{pr}(n) = 3$ and $\mathrm{pr}_2(n) > 5$, then*
  a. *$\mathrm{r}_0(n,[1,3]) = 2n/3 - 3$, and*
  b. *whenever $\mathrm{r}_0(n) = \mathrm{r}_0(n,[s_0,s_1]) = 2n/3 - 2$, then $[s_0,s_1] = [2, n/3+1]$.*
- *if $\mathrm{pr}(n) \ge 5$, then $\mathrm{r}_0(n) = \mathrm{r}_0(n,[s_0,s_1]) = n - 2n/\mathrm{pr}(n) - 1$, when $s_0 = s_1 = n/\mathrm{pr}(n) + 1$;*

Theorem 6. *Conjecture 2 holds for all $n \le 120$. In addition, if $\mathrm{pr}(n) = 3$, then the conjecture holds for all $n \le 300$.*

Before we discuss these rather complicated theorems and the conjecture in detail, we remark that Theorem 6 is established by a purely numerical calculation as described in Section 3. So, for practical array sizes, the two theorems provide a complete description of this design for $\mathrm{HoVer}^3_{2,1}$ codes. We extended the search when $\mathrm{pr}(n) = 3$ in hopes of refining the second statement of the conjecture when $\mathrm{pr}(n) = 3$ and $\mathrm{pr}_2(n) > 5$ (see the additional remarks below).

The first two items of Theorem 5 completely characterize $\mathrm{r}_0(n)$ for n even. In particular, it shows that the optimal choice for offsets is $[s_0, s_1] = [1, 2]$. Note that for the non-exceptional cases, this code supports one more row than the $\mathrm{HoVer}^2_{1,1}[r, n]$ code (for even n), which was a surprise.

The rest of the theorem and conjecture are for odd n, i.e., when $\mathrm{pr}(n) \geq 3$. The third item covers odd n for the specific offsets $[1, 2]$. This gives a lower bound on $\mathrm{r}_0(n)$ that is clearly not optimal as is seen in the rest of theorem and in the conjecture.

For the case where $\mathrm{pr}(n) \geq 5$, the last item in the theorem provides an upper bound on $\mathrm{r}_0(n)$ and last item in the conjecture claims that this upper bound is tight for the offsets $s_0 = s_1 = n/\mathrm{pr}(n) + 1$. For prime $n = p \geq 5$, this would give

$$\mathrm{r}_0(p) = \mathrm{r}_0(p, [2, 2]) = n - 3,$$

which is a special case of Conjecture 1. The bound

$$\mathrm{r}_0(n) \leq n - 2n/\mathrm{pr}(n) - 1$$

is less than that in Theorem 3 (for $\mathrm{HoVer}^2_{1,1}$) by an additional term $n/\mathrm{pr}(n)$.

The remainder of the theorem and conjecture deal with the case that $\mathrm{pr}(n) = 3$. This subdivides into two additional cases, depending on whether or not $\mathrm{pr}_2(n) = 5$ (that is, whether or not 5 divides n). If $\mathrm{pr}_2(n) = 5$, the theorem gives the upper bound and the conjecture gives the lower bound, and further identifies the best offsets as $[1, 3]$. We have a partial proof of the lower bound, but failed (because of time constraints) to finish analysing all the cases.

For the remaining case when $\mathrm{pr}_2(n) > 5$, there is a gap of one between the upper bound of the theorem and the lower bound of the conjecture. Generally, offsets $[1, 3]$ give the best value for $\mathrm{r}_0(n)$, but erratically, the offsets $[2, n/3 + 1]$ did one better (achieving the upper bound of the theorem). We examined this case extensively (up to $n \leq 300$ and a few cases beyond) and found only $n = 9, 27, 33, 63, 87, 117, 123, 153, 207, 243, 297$ achieve the upper bound (and only with offsets $[2, n/3 + 1]$). There appeared to be no pattern to these cases, which is why we do not have a more precise conjecture.

We believe that the methods we applied to our partial proof of the first statement of the conjecture can be mined (given enough time) to completely prove the conjecture (perhaps excluding the exceptional cases when $\mathrm{pr}(n) = 3$ and $\mathrm{pr}_2(n) > 5$).

In the appendix, we provide most of the proof of the theorem (with the exception of the case mentioned above). The proofs are instructive about the reconstruction algorithms. As we saw in Section 3.1, for the $\text{HoVer}^2_{1,1}[r,n]$ code, there were reconstruction chains that zig-zag through the two lost strips. In this 3-fault tolerant construction, there are some complex paths that can be used to show non-reconstructability for specific failure sets (to give the upper bounds). Proving the lower bounds is significantly more difficult.

3.2.1.1. Alternative Slope Choices There is nothing magic about the slopes ± 1 that we chose in forumulas (3.4) and (3.5); they provide only the most obvious and simplest design. Because we have a natural notion of wrap-around from left-to-right (modulo n), a slope of $\pm 1/k$ may be used as well where k is any integer. Such a design easily puts one element per row into a parity value. For example, the following formula is the Knight's move to the right (slope $+1/2$), with offset s:

$$K_s(j) = \bigoplus_{k=0}^{r-1} X(k, \text{mod}_n(j + 2(r-1-k) + s)). \tag{3.6}$$

Alternatively, it is also possible to use patterns that follow lines of other slopes (e.g., ± 2). However, because there is no natural notion of top-to-bottom wrap-around, there is additional complications in this case. For example, with slope $+2$, every other row is skipped. To address this, we can use two independent lines, both of slope $+2$, the first starting in the last row and the second starting in the next to last row (and with independent offsets):

$$\begin{aligned} K^e_{s_0}(j) &= \bigoplus_{\substack{0 \le k \le r-1 \\ k \text{ even}}} X(k, \text{mod}_n(j + (r + \delta - 2 - k)/2 + s_0)) \\ K^o_{s_1}(j) &= \bigoplus_{\substack{1 \le k \le r-1 \\ k \text{ odd}}} X(k, \text{mod}_n(j + (r - \delta - 1 - k)/2 + s_1)) \\ V_{[s_0,s_1]}(j) &= K^e_{s_0}(j) \bigoplus K^o_{s_1}(j) \end{aligned} \tag{3.7}$$

where δ is zero for r even, and is one for r odd.

Curiously, combinations of these types of parity formulas with those of slope $+1$ or -1 sometimes yield better ranges for r, though never consistently for all n. The following chart shows the largest r, denoted r_0', that we discovered for each $n \le 24$ when it exceeded the values given by the Theorems above. In many other cases, the value matched that of the above theorems and in the remaining cases any other combination we examined did worse (e.g., whenever $n \ge 7$ was prime). The first column gives the value of n. In the second column is the value given by Theorems 5 and 6. The third column contains the best value we determined using alternative

patterns. For all $n \neq 10$ in the table, using equations (3.4) and (3.7) produced a better result. For $n = 10$, equations (3.5) and (3.7) yielded the best r.

Valid $[r, n]$ for $\text{HoVer}^3_{2,1}$		
n	old r_0	new r_0'
10	4	6
14	7	9
15	6	8
16	8	9
18	9	10
20	10	12
21	11	12
22	11	15
24	12	13

In summary, there are many different alternative designs for a $\text{HoVer}^3_{2,1}$ code; we have not discovered any design which uniformly has the best efficiency over all ranges of n. Furthermore, as indicated by the imcompleteness of the theory, even the simplest design with slopes ± 1 is extremely difficult to analyse.

3.2.2. One V-Parity Row In the last section, we gave a construction and (incomplete) theory for a HoVer code with 3-fault tolerance, that used one h-parity strip and two v-parity rows. In this section, we discuss one of many possible HoVer codes with one v-parity row and 3-fault tolerance. This code weakly generalizes BCP [1] or ZZS [13], as these are 2-fault tolerant and have only one v-parity row. As in these codes, each data element has to touch two parity elements in the one v-parity row. For a given array size, the codes with one v-parity row (BCP and ZZS) have data row count approximately half that of the X-code which has two v-parity rows. A similar phenomenon occurs here comparing $\text{HoVer}^3_{1,1}$ and $\text{HoVer}^3_{2,1}$; see also the remarks at the end of this section. This is also seen in Theorem 1, where for a $\text{HoVer}^3_{1,1}$ code, we must have

$$r \leq (n-3)/2. \tag{3.8}$$

One possible construction is simply to add an h-parity strip to one of these known vertical codes such as BCP or ZZS. It turns out that this does not work; however, it might be possible to modify this construction in an attempt to achieve a valid 3-fault tolerant code. We did not pursue this line of reasoning, primarily because these codes have little geometric or rotational symmetries – a feature we believe makes for simpler implementations. Instead we offer the following alternative.

For two offsets $[s_0, s_1]$, define the v-parity elements (in the one row) by the equations:

$$V_{[s_0,s_1]}(j) = U_{s_0}(j) \bigoplus D_{s_1}(j), \tag{3.9}$$

where U_s and D_s are the up- and down-diagonals defined in (3.4) and (3.5), respectively. Visually, for a given v-parity element, this is a valley with the v-parity element at the bottom of the valley, and with the right hill offset s_0 units to the right and the left hill offset s_1 units to the left. Another way to view this is as an example of the $\text{HoVer}^3_{2,1}$ code with the two parity rows added together into a single row.

The following diagram shows how each data element contributes to the parity elements for $[r, n] = [4, 10]$ and $s_0 = s_1 = 1$ (subscripts $[s_0, s_1] = [1, 1]$ are suppressed in the parity row). Each data element is labeled with two v-parity strip numbers, the first indicates the strip number of the v-parity element it touches on the up-slope and the second label corresponds to the down-slope. The parity element $V(4) = V_{[1,1]}(4)$ and its data elements are highlighted (in bold).

$\text{HoVer}^3_{1,1}$ Up/Down Parity Relation Examples									
6:**4**	7:5	8:6	9:7	0:8	1:9	2:0	3:1	**4**:2	5:3
7:3	8:**4**	9:5	0:6	1:7	2:8	3:9	**4**:0	5:1	6:2
8:2	9:3	0:**4**	1:5	2:6	3:7	**4**:8	5:9	6:0	7:1
9:1	0:2	1:3	2:**4**	3:5	**4**:6	5:7	6:8	7:9	8:0
$V(0)$	$V(1)$	$V(2)$	$V(3)$	$\mathbf{V(4)}$	$V(5)$	$V(6)$	$V(7)$	$V(8)$	$V(9)$

(This particular instance of the code does not have 3-fault tolerance; it is given only for visualization purposes – a small valid example would not have enough rows to see the pattern. See below for valid sets of parameters.)

This construction is clearly analogous to the $\text{HoVer}^3_{2,1}$ construction and so assumes many of the same symmetry properties. In fact, Proposition 1 holds for the $\text{HoVer}^3_{1,1}$ code without change.

The following table summarizes a small sample (for array sizes up to 32) of the experimentally discovered constructions and the conjecture immediately after provides our best summary guess to what the theorem should be. By the remarks above (and Proposition 1), we only considered $s_0, s_1 \leq n - r$. For all cases, we set $s_0 = s_1 = 1$ (with the exception of $n = 16, 24, 28$ where we use $s_0 = 3$ and $s_1 = 6, 10, 12$, respectively, though other choices work in each case as well).

Valid $[r, n]$ for $\mathrm{HoVer}^3_{1,1}$					
n	r_0	n	r_0	n	r_0
5	1	14	3	23	10
6	1	15	4	24	6
7	2	16	4	25	9
8	1	17	7	26	6
9	2	18	4	27	8
10	2	19	8	28	7
11	4	20	4	29	13
12	2	21	6	30	7
13	5	22	5	31	14

Conjecture 3. *Let* $\mathrm{pr}(n)$ *be as before. Let* $\mathrm{r}_0(n.[s_0, s_1])$ *be the largest value of* r *so that equation (3.9) defines a valid* $\mathrm{HoVer}^3_{1,1}$ *code of parameters* $[r, n]$ *for the offsets* s_0, s_1 *and let* $\mathrm{r}_0(n)$ *be the largest such value achieved by any offset. We have the following statements.*

- *If* $\mathrm{pr}(n) \geq 3$*, then* $\mathrm{r}_0(n) = \mathrm{r}_0(n, [1, 1]) = \frac{1}{2}(n - n/\mathrm{pr}(n) - 2)$.
- *If* $\mathrm{pr}(n) = 2$ *(that is,* n *is even), then* $\mathrm{r}_0(n, [1, 1]) = \lfloor (n-2)/4 \rfloor$.
- *If* $n \equiv 2 \pmod 4$*, or* $n = 8, 12, 20$*, then* $\mathrm{r}_0(n) = \mathrm{r}_0(n, [1, 1]) = (n-2)/4$.
- *If* $n \neq 8, 12, 20$ *and* $n \equiv 0 \pmod 4$*, then* $\mathrm{r}_0(n) = \mathrm{r}_0(n, [s_0, s_1]) = n/4$ *where* $s_1 = n/2 + 1 - s_0$ *and*

$$\begin{array}{rcll} s_0 & = & n/4 - 1 & \text{if } n \not\equiv 0 \pmod 3 \\ s_0 & \in & \{3, 4, 6, 7\} & \text{if } n \equiv 0 \pmod 3, \text{ that is, if } n \equiv 0 \pmod{12} \end{array}$$

Theorem 7. *Conjecture 3 holds for all* $n \leq 120$*. In addition, if* $n \equiv 0 \pmod{12}$*, then the conjecture holds for all* $n \leq 600$.

We do not have a more precise conjecture that specifies the s_0 in the last statement of the conjecture/theorem, but, as the theorem claims, we have examined this up to $n = 600$. Unfortunately, no pattern emerges as to the exact value for s_0 even with this much numerical evidence. Notice that the bound in this case is one more than the bound for the specific offsets $[1, 1]$ (in the second statement). When n is prime, this conjecture (and the values from the table) are consistent with Conjecture 1. Furthermore, for n odd, this conjecture provides a bound better than $1/2$ of the $\mathrm{HoVer}^3_{2,1}$ bound, which was somewhat unexpected considering the remarks at the beginning of this section.

We further remark that the $\text{HoVer}^3_{1,1}$ codes with $r_0(n) = 1$ can be viewed as extensions, by adding the h-parity strip, to the 2-fault tolerant simple WEAVER codes in [7].

3.3. Higher Fault Tolerance

In this section we briefly describe our limited understanding of higher fault tolerant HoVer codes. First, we give one construction that can tolerate four faults, and list some experimentally tested cases for practical array sizes up to 32 strips ($n \leq 31$). We next briefly describe what we expect may be true in more generality.

As always there are numerous ways to try to define a $\text{HoVer}^4_{3,1}$ code with three v-parity rows. The most natural is to extend the $\text{HoVer}^3_{2,1}$ code by defining the first two rows of v-parity by the equations (3.4) and (3.5). We experimented with a number of alternatives for the third row of parity. The best results in most cases (but not all) came from using the formulas in (3.7). However, in one case ($n = 11$), replacing K^o with a pattern of slope -2 produced a better r_0.

The following table shows some experimental results for the $\text{HoVer}^4_{3,1}$ codes defined by (3.4),(3.5) and (3.7) – we do not give the offsets.

Valid $[r, n]$ for $\text{HoVer}^4_{3,1}$					
n	r_0	n	r_0	n	r_0
5	1	14	7	23	15
6	1	15	8	24	9
7	2	16	6	25	16
8	3	17	10	26	13
9	4	18	8	27	17
10	4	19	12	28	13
11	6*	20	9	29	21
12	5	21	13	30	15
13	7	22	11	31	22

The exceptional case $n = 11$ is indicated by $*$. Note that for prime n, the best we could obtain in these constructions was significantly below the bounds of Conjecture 1. We expect that there are alternative constructions that we have simply missed, so far.

4. Features

In this section we summarize the key features of these HoVer codes, both the advantages and disadvantages. In general, there are direct trade-offs between the two.

4.1. Storage Efficiency

There are two ways to measure the storage efficiency of these HoVer codes, depending on whether or not an array is "packed" (to fill the □ portion of one stripe with data or parity of another stripe – see earlier remarks in Section 2). The packed case measures efficiency in the sense of coding theory (the rate of the code); we call this Eff_C. The unpacked version assumes that the □ subarray of the data/parity layout (Figure 1) is physically wasted disk space; we use Eff_D for this type of efficiency.

A comparison of these functions to MDS codes should apply only when the number of disks (or strips) in the array is the same. So we compare MDS horizontal codes of t-fault tolerance with stripe size equal to N and data strips equal to $N-t$. We label this efficiency as $\mathit{Eff}_O(t,N)$, for optimal. We then compare a $\mathrm{HoVer}^t_{v,h}[r,n]$ code with $\mathit{Eff}_O(t,n+h)$, so $N=n+h$.

We have the formulas

$$\begin{aligned}
\mathit{Eff}_D(r,n,v,h) &= \frac{rn}{(r+v)(n+h)} \\
\mathit{Eff}_C(r,n,v,h) &= \frac{rn}{(r+v)(n+h)-vh} \\
\mathit{Eff}_O(t,N) &= \frac{N-t}{N}
\end{aligned}$$

and the relations

$$\mathit{Eff}_D(r,n,v,h) < \mathit{Eff}_C(r,n,v,h) < \mathit{Eff}_O(t,n+h)$$

(recall that $v+h \le t$).

Clearly, Eff_C and Eff_D are increasing functions of r and n (independently). That is, for fixed n, the efficiences are maximal for the largest admissible r. By Theorem 1,

$$\begin{aligned}
\mathit{Eff}_D(r,n,v,h) &= \frac{n+h-t}{n+h} - O\left(\frac{1}{n}+\frac{1}{r}\right) \\
&= \mathit{Eff}_O(t,n+h) - O\left(\frac{1}{r}\right).
\end{aligned}$$

and so approximates (but is strictly less than) the optimal efficiency for a t-fault tolerant code. These "inefficiencies" appear in two places. First, the optimal efficiency for an r-row horizontal code would have tr redundant bits, whereas for our codes we have $nv+rh$; this is strictly larger than tr because of Theorem 1. The second inefficiency comes from the fact that Theorem 1 provides only an upper bound on r and we saw in our constructions that this was not always achieved.

Figure 2 gives a comparison of the best efficiency values versus MDS optimal efficiency for the $\text{HoVer}^2_{1,1}$ codes of Section 3.1. Note that n represents the number of data strips, so is one less than the array size. Because r does not scale linearly with n, neither do the efficiencies. Observe that the worst efficiencies are for n even, that is, for an odd array size. The best efficiencies are for n prime, which includes common array sizes of $8, 12$ (but not 16, unfortunately).

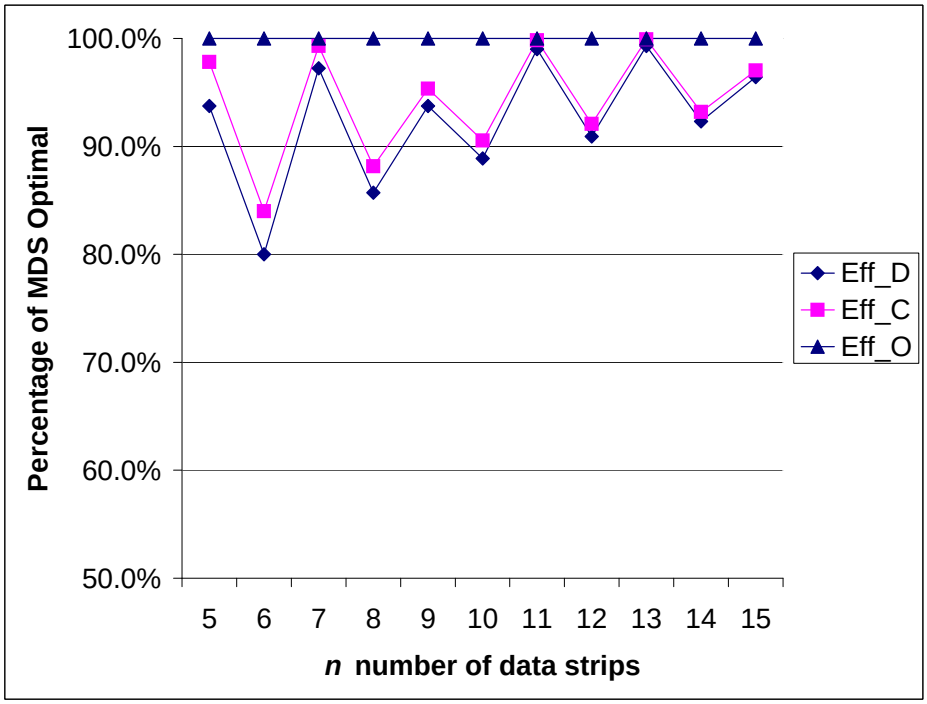

Figure 2: Efficiency Comparison for $\text{HoVer}^2_{1,1}$ constructions (with $s = 1$)

Similarly, Figure 3 shows the comparison to MDS optimal efficiency for the $\text{HoVer}^3_{2,1}$ and $\text{HoVer}^3_{1,1}$ codes. The efficiency values for $v = 2$ and $v = 1$ are the same with the exception of $n = 8, 12, 14, 15$. This is because in all the other cases, the value of r is halved from $v = 2$ to $v = 1$ and this leaves the HoVer efficiency formulas unchanged. For $n = 8, 12, 14$, the $\text{HoVer}^3_{1,1}$ is marginally less efficient, but for $n = 15$ it is more efficient (because in this case, and for many larger values of n, $\text{r}_0(n)$ for $\text{HoVer}^3_{1,1}$ is greater than $1/2$ the value for $\text{HoVer}^3_{2,1}$).

We leave the efficiency table for $\text{HoVer}^4_{3,1}$ to the reader.

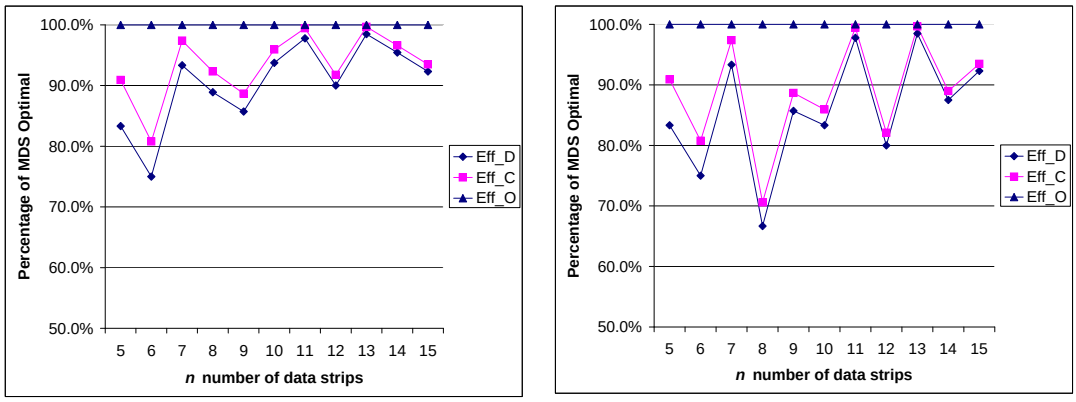

Figure 3: Efficiency Comparison for best known $\text{HoVer}^3_{2,1}$ (on left) and $\text{HoVer}^3_{1,1}$ (on right) constructions

4.2. Short Write IO Costs

We use the term "short write" to indicate a host write to a single user data sector in a stripe. More generally, a short write is a host write to any portion of a data element. A feature of our HoVer code design is that every data element touches exactly t parity elements. Consequently, the short write update cost is exactly $2(t+1)$ IO seeks, achieved using a standard read-modify-write algorithm for the t parity values. This is clearly optimal for most codes (the exception are symmetric codes, or near symmetric codes, with the same number of parity elements as data elements, such as the WEAVER codes [7]). This bound is better than that for most horizontal codes such as EVENODD or Blaum-Roth where some data elements touch more than $t = 2$ parity elements.

4.3. Reconstruction Costs

The HoVer codes show their most significant advantage over most other codes with respect to reconstruction costs. Consider the $\text{HoVer}^2_{1,1}$ code (a similar argument applies to $\text{HoVer}^t_{t-1,1}$ codes for $t = 3, 4$ when less than t strips are lost and a variation applies to $\text{HoVer}^3_{1,1}$ as well). We discuss two scenarios: (a) reconstruction when one strip is lost, and (b) reconstruction when two strips are lost. The former is called a Degraded array, and the latter a Critical array. In the discussion below, we ignore cases where the h-parity strip is lost (as that is the easy case).

Degraded: When one data strip is lost, there are r data elements that are lost. A host read to one of these strips requires reconstruction of the lost data. RAID4 (or RAID5) with n data strips requires reading n chunks of data from n strips (the $n-1$

remaining data strips and the parity strip). An XOR computation with n inputs is then required to reconstruct the lost data. This same argument applies to any horizontal code such as EVENODD or RDP (since these have a RAID4 subcode). Use of the other parity elements in these codes requires at least as many disk reads and XOR computations. It is easy to verify that each of the vertical codes (X-code, BCP, or ZZS) requires reading from $n-2$ strips (on an array of size n) and an XOR computation with $n-2$ inputs.

This method of using the h-parity may also be used for the HoVer code, since they contain a RAID4 subcode. But, the HoVer codes also have the v-parity elements available for reconstruction. Use of these parity elements requires reading from only r strips. Since $r \leq n-2$ (for the $\text{HoVer}^2_{1,1}$ code), we have a net performance advantage of the HoVer codes over the other known codes.

By Theorem 2, we can vary the size of r below the bound of Theorem 3. A smaller r improves the reconstruction costs linearly (with a trade-off in efficiency, however).

Furthermore, even for complete reconstruction of all the data and parity on the lost strip, we see from the proof of Theorem 3, that only a subportion of the entire (remaining) stripe need be accessed for reconstruction of a single strip and the XOR computations all have r inputs. And smaller values for r improve this considerably. In constrast for most other codes, the entire stripe needs to be accessed and most reconstruction formulas have order n inputs (or more).

Critical: Now suppose two data strips are down in a $\text{HoVer}^2_{1,1}$ array. We saw in the proof of Theorem 3 and the discussion in Section 3.1.1 that the recovery algorithm involved following reconstruction chains. For most failure scenarios in the 1-shift code and all scenarios in the 2-shift code, there are at least two such chains. Most scenarios, in fact, have quite a few more chains, and in some instances, chains are only one element long. For all vertical MDS codes and most horizontal codes, reconstruction follows such chains as well, but in these cases the chains are generally quite long and there are at most two such chains. Longer chains implies more work to reconstruct the elements near the end of the chains. In addition, the maximum length of the chains is proportional to the number of rows. We see that the HoVer codes which support constructions with variable r have more efficient and parameterizable reconstruction chains.

In addition, it is easy to see that if two lost strips are separated by $r+s$ consecutive strips, then each strip can be recovered independently, using *only* the v-parity row. This follows because no diagonal intersects both strips so that all recovery chains have length one. (In fact, as the gap between the lost strips increases (modulo n), the number of chains of length one increases smoothly.)

4.4. Extended Fault Tolerance

The ability to vary the parameter r has an additional advantage. Suppose r is small compared to n. In this case, the array can recover from certain instances of more than t failures. As we saw above for the $\text{HoVer}^2_{1,1}$ case, if the gap between two lost strips is large enough, each can be recovered as an independent instance of a single failure (this is a property also shared by WEAVER codes [7]). More precisely, the $\text{HoVer}^2_{1,1}$ code can recover from any number of failures so long as in any set of $r+s$ consecutive strips (with wrap-around), there are no more than two lost strips. (If r is too large, then there is no room for such combinations.) None of the MDS codes have this property (as a direct consequence of the MDS property). Similar extended fault tolerance is available for the other HoVer code constructions we have given, provided r is sufficiently small relative to n.

4.5. Parameter Flexibility

In general, most codes that tolerate $t \geq 2$ failures have a very tight restriction on the relationship between the parameters r and n (e.g., BCP [1] requires $r = n/2 - 1$ with n even). Other codes have primality restrictions on some of the parameters. For example, the X-code [12] requires n prime *and* $r = n - 2$. The horizontal codes such as EVENODD [2], Blaum-Roth [4] and RDP [5] require $r = p - 1$ for some prime p and $n \leq p$ (or $n \leq p - 1$ for RDP). In these cases, the number of rows must increase in order to increase the size of the array. Increasing the number of rows implies performance penalties. For example, more rows implies more fragmentation of the parity formulas, more formulas and so more computation costs. In addition, as we saw above, decreasing the number of rows can improve reconstruction costs and add additional fault tolerance.

The HoVer codes are unique in that the number of rows is bounded *above* by a function of the array size. All array sizes are supported in a uniform way (unlike the BCP codes, the X-code or ZZS [13]). The implementator may choose an r value that works for the entire range of array sizes supported by the system. In this way, the array can support changes in the array size without significant rearrangement of data or parity elements. In addition, the system will have uniform performance characteristics across the entire range of array sizes even when one disk is lost (this property does not even hold for RAID4 or RAID5). However, this comes at the cost of efficiency. In short, the HoVer codes provide a range of constructions allowing an implementor flexibility in balancing the efficiency penalty with the performance advantages.

5. Open Problems

There are many open problems in the context of HoVer codes. We list a few of the more important ones here (in our order of importance).

- Determine (and prove) the precise conjecture for the $\text{HoVer}^3_{2,1}$ code when $\text{pr}(n) = 3$.
- Prove the lower bounds in Conjecture 2 for $\text{HoVer}^3_{2,1}$ codes (e.g., when n is prime).
- Determine (and prove) the precise formulas for $\text{HoVer}^3_{1,1}$ (Conjecture 3).
- Find a construction for a $\text{HoVer}^3_{1,2}$ code (one parity row and two h-parity strips).
- Find a reasonable general construction for $\text{HoVer}^t_{t-1,1}$ code for any t.

6. Summary

In this paper we presented the HoVer codes, a new family of XOR-based erasure codes for disk arrays. They combine the notions of h-parity (parity stored on separate disks from the data) and v-parity (parity stored on the same disk with unrelated data) as a hybrid data/parity layout. We gave a complete characterization of one construction of these codes (and a simple variation) that tolerates 2 lost disks. In addition we provided two alternative constructions of 3-fault tolerant codes and one 4-fault tolerant code. These latter constructions were supported by a mixture of theory, conjecture and experiment, but cover all practical array sizes. We argued that the HoVer codes can provide performance advantages over traditional codes with a trade-off in efficiency. More to the point, the HoVer codes have parameter flexibility that gives implementators and designers the ability to adjust the performance/efficiency trade-off to achieve the best balance of these two metrics in a specific environment or system.

7. Acknowledgements

The author thanks KK Rao, Tapas Kanungo, and Veera Deenadhayalan for their contributions to this work. We also want to thank the BlueGene/L support team at IBM's Almaden Research Center for the opportunity to run many of the larger experiments on their system (and their assistence). Testing array sizes in the 300–600 range (as we did for Theorems 6 and 7) required considerable computing power.

References

[1] S. Baylor, P. Corbett, and C. Park. Efficient method for providing fault tolerance against double device failures in multiple device systems, January 1999. U. S. Patent 5,862,158.

[2] M. Blaum, J. Brady, J. Bruck, and J. Menon. EVENODD: an efficient scheme for tolerating double disk failures in RAID architectures. *IEEE Transactions on Computers*, 44:192–202, 1995.

[3] M. Blaum, J. Brady, J. Bruck, J. Menon, and A. Vardy. The EVENODD code and its generalization. In J. Jin, T. Cortest, and R. Buyya, editors, *High Performance Mass Storage and Parallel I/O: Technologies and Applications*, chapter 14, pages 187–208. IEEE and Wiley Press, New York, 2001.

[4] M. Blaum and R. M. Roth. On lowest density MDS codes. *IEEE Transactions on Information Theory*, 45:46–59, 1999.

[5] P. Corbett, B. English, A. Goel, T. Grcanac, S. Kleiman, J. Leong, and S. Sankar. Row-diagonal parity for double disk failure. In *Proceedings of the Third USENIX Conference on File and Storage Technologies*, pages 1–14, 2004.

[6] G. A. Gibson, L. Hellerstein, R. M. Karp, R. H. Katz, and D. A. Patterson. Failure correction techniques for large disk arrays. In *Proceedings of International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 123–132, Boston, MA, 1989.

[7] J. L. Hafner. WEAVER erasure codes for disk arrays. Technical Report RJ xxxxx, IBM Research, San Jose, CA, 2005.

[8] T.-D. Han, S.-D. Kim, S.-B. Yang, K.-W. Lee, and S. Chang. Method for storing parity and rebuilding data contents of failed disks in an external storage subsystem and apparatus thereof, December 2000. U. S. Patent 6,158,017.

[9] S. Nanda. Method and system for disk fault tolerance in a disk array, April 2004. U. S. Patent Application US 2004/0078642 A1.

[10] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8:300–304, 1960.

[11] A. Wilner. Multiple drive failure tolerant raid system, December 2001. U. S. Patent 6,327,672 B1.

[12] L. Xu and J. Bruck. X-code: MDS array codes with optimal encoding. *IEEE Transactions on Information Theory*, IT-45:272–276, 1999.

[13] G. V. Zaitsev, V. A. Zinovev, and N. V. Semakov. Minimum-check-density codes for correcting bytes of errors. *Problems in Information Transmission*, 19:29–37, 1983.

A. Partial Proof Of Theorem 5

The propositions (and proofs) of this section address some of the more interesting cases in Theorem 5 for $\text{HoVer}^3_{2,1}$ codes. They illustrate the complexity and difficulty of the theoretical analysis of these codes.

Proposition 2. *Suppose* $\text{pr}(n) \geq 5$*, and set* $r = n - 2n/\text{pr}(n)$*. For any* $[s_0, s_1]$*, the equations (3.4) and (3.5) do not provide a valid 3-fault tolerant code.*

Proof. (Outline) Let $p = \text{pr}(n) \geq 5$, $\Delta = n/p$, $q = (p-1)/2$ and assume that strips 0, Δ and $q\Delta$ have been lost. By Proposition 1, we can assume that $1 \leq s_0 < \Delta$. Consider the intersection of the rows $s_0 - 1 + k\Delta$ for $k = 0, \ldots, (p-3)$ with the three lost strips. Of the up-diagonals that intersect these lost strips, exactly p cross at these rows and no others. They are $U(k\Delta)$ for $k = 0, 1, \ldots, p-1$. Of these, $U(0)$, $U(\Delta)$ and $U(q\Delta)$ are assumed lost. Similarly, there are exactly p total down-diagonals intersecting these lost strips at these rows: $D(r + s_1 - s_0 - k\Delta)$ for $k = 0, 1, \ldots, p-1$.

On the intersection of these horizontal, up-diagonal and down-diagonals, there are $3(p-2)$ unknown (lost) elements. The horizontal parities provide for $(p-2)$ equations. There are a total of p up-diagonal equations intersecting through these lost values, but of these, as noted above, three are lost so that we have only $p-3$ valid up-diagonal parity equations. Similarly, there are at most p down-diagonal parity equations available through these lost values (some down-diagonals may be lost). If at least two of these are also lost, there remains at most $(p-2)+(p-3)+(p-2) = 3p-7$ equations. These provide an underdetermined system of equations for the lost values and prove that this configuration is unsolvable.

Now suppose that all the down-diagonal parities are still available. This provides us with a total of $3p-5$ equations for $3p-6$ variables. But since the sum of all the horizontal equations must equal the sum of the down-diagonal equations, we have at least one redundant equation so that our system is now square (same number of variables as equations).

If in fact only one down-diagonal is lost, then again we have a square system. In addition, we can recompute the lost down-diagonal as the sum of all the horizontal equations plus (xor) the sum of the known down-diagonal equations.

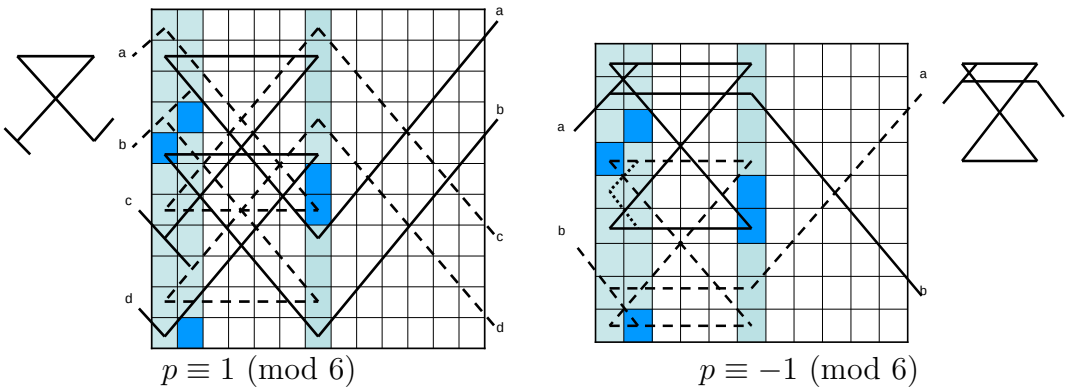

Figure 4: Examples of patterns that show a redundancy relationship for $p = \mathrm{pr}(n) \equiv \pm 1 \pmod 6$. Shaded strips are assumed lost, darkened cells have lost up-diagonals; building block structure is shown. Letter labels indicate diagonals that are connected under wrap-around. Shaded and darkened cells have degree either zero or two.

As a result, if either all or one of the down-diagonals is lost, the system is square and we can assume that all the down-diagonal values are available. To conclude the proof, we need to show that there is a redundant equation in the system. This is done by determining a pattern (drawing a graph) of up, down and horizontal lines through the array in which each lost value is touched either zero or exactly twice (the degree is either zero or two) and no element is touched by a diagonal line if that diagonal is lost. Each line corresponds to an accessible parity equation; the pattern implies that the set of equations is interdependent, and so unsolvable.

Two such patterns occur, one for $p \equiv 1 \pmod 6$ and one for $p \equiv -1 \pmod 6$. Two examples are given in the Figure 4; the general case can be extracted from this. In the figures, the shaded strips are assumed lost, and the darkened cells have lost up-diagonal parities. In the case $p \equiv 1 \pmod 6$, there is a basic structure that always occurs with its vertical mirror and there are $(p-1)/6$ such mirrored pairs. The horizontal bar of the basic structure occurs at rows $1, 4, \ldots, (p-7)/2+1$.

In the case $p \equiv -1 \pmod 6$ and $p \geq 11$, there is one basic structure that also always occurs with its mirror, and there are $(p+1)/6$ such pairs. The top horizontal bar is aligned at rows $0, 3, \ldots, (p-5)/2$. There is one additional small structure (a wedge) indicated by the dotted lines that also occurs in middle left portion of the picture. The structure for $p = 5$ contains only the wedge and a rotated bow-tie. $\square$

The next two propositions give upper bounds for r when $\mathrm{pr}(n) = 3$.

Proposition 3. *Suppose* $\mathrm{pr}(n) = 3$ *and* n *is divisible by* 5. *Then* $\mathrm{r}_0(n) \leq 3n/5 - 3$ *for the construction provided by equations (3.4) and (3.5); that is, for no values of* $[s_0, s_1]$ *do these equations provide 3-fault tolerant code.*

Proof. This is the most complicated of the upper bound proofs; there is no single failure pattern that covers all the cases.

Case 1: If $\Delta_5 = n/5$ and we loose strips 0, Δ_5 and $2\Delta_5$, then for any s_1 in the range $3 \leq s_1 \leq \Delta_5$, we have a non-recoverable system of six variables in three rows with only one down-diagonal parity, three horizontal parities and three up-diagonal parities. The top row at offset $k = s_1 - 3$ has elements in strips Δ_5 and $2\Delta_5$ (label them B and C respectively); the second row at offset $k = s_1 - 3 + \Delta_5$ has elements in strips 0 and $2\Delta_5$ (label them D and F); the third row at offset $k = s_1 + 2\Delta_5$ has elements in strips 0 and Δ_5 (label them G and H). It is easy to see that this system is underdetermined: the sum of the horizontal parities equals the sum of the up-diagonal parities and in addition one up-diagonal equation $(B + D)$ is the sum of a down-diagonal equation $(B + F)$ and a horizontal equation $(D + F)$. We use symmetry to prove the result for the second interval $\Delta_5 + 3 \leq s_1 \leq 2\Delta_5 = n - r - 2$.

Case 2: If $\Delta_3 = n/3$ and we loose strips 0, Δ_3 and $2\Delta_3$, then for any s_1 in the range $n/15 + 3 \leq s_1 \leq n/3 = n - r - (n/15 + 2)$, we have a non-recoverable system of 6 variables with at best five equations (two horiz and three ups) because all the downs are lost. The row offsets are $k = r - 1 + s_1 - 2\Delta_3$ and $k = r - 1 + s_1 - \Delta_3$.

Case 3: If $s_0 = s_1$ and we loose strips 0, $\Delta_3 = n/3$ and $2\Delta_3$, then we have either one row or two rows in which all up and down-diagonals are lost, leaving only one or two equations for the three or six variables. Rows are $k = r - 1 + s_0 - \Delta_3$ and possibly $k = r - 1 + s_0 - 2\Delta_3$.

Case 4: Let $\Delta_t = (n - 1)/2 - t$ for some t and assume that strips 0, Δ_t and n (the h-parity) are lost. Then if $s_1 = s_0 + 1 + 2t$, the element at row $k = r - 1 + s_0 + \Delta_t$ has all its parities lost (the up and down diagonals both are on strip 0.

Case 5: Let $\Delta_t = (r - 1)/2 + s_0 + t$ for some t and suppose that strips 0, Δ_t and n again are lost. Then if $s_1 = n - r - 2t - s_0 + 1$, the up-diagonal and down-diagonal on strip 0 again meet at strip Δ_t at row $k = (r-1)/2 - t + s_0$, which is unrecoverable.

Case 6: Let $\Delta_5 = n/5$ and assume that strips 0, Δ_5 and $3\Delta_5$ are lost. At rows $k = \Delta - 2 = r - 2\Delta_5$ and $k = 2\Delta_5 - 2 = r - \Delta_5$, we have three lost elements, touched by the five up-diagonals $U(0), U(\Delta_5), \ldots, U(4\Delta_5)$ and the five down-diagonals $D(0), D(\Delta_5), \ldots, D(4\Delta_5)$. Of these elements, the two on strip $3\Delta_5$ can be reconstructed directly from the two parities on strip $2\Delta_5$. However, what remains are four elements with four equations (two horizontal, one each surviving up and down diagonal). One equation is redundant (because the sum of the up and down diagonal

equations equals the sum of the two horizontal equations); consequently, the system is underdetermined.

It is easy to see that Cases 1 and 2, together with symmetries, give all values of $s_0 \geq 3$ and for $s_0 = 1, 2$, all s_1 in range $3 \leq s_1 \leq n - r - 2$. Case 3 covers when $s_0 = s_1$ so we can assume $s_0 < s_1 \leq 2$. With that, there remains only a few cases: $[s_0, s_1] = [1, 2], [1, n-r-1], [1, n-r], [2, n-r-1]$. The first of these is covered when $t = 0$ in Case 4. The last two are covered by Case 5, also when $t = 0$. Finally, the remaining case is handled by Case 6 again when $t = 0$. □

Proposition 4. *Suppose* $\mathrm{pr}(n) = 3$ *and* 5 *does not divide* n. *For* $r = 2n/3 - 2$ *and* $[s_0, s_1] = [1, 2]$, *the equations (3.4) and (3.5) do not provide a valid code. If* $r = 2n/3 - 1$, *then no values for* $[s_0, s_1]$ *provide a valid code.*

Proof. The proof is similar to the upper-bound proof for $\mathrm{pr}(n) \geq 5$. Assume that strips $0, \Delta_3 = n/3$ and $2\Delta_3$ are lost. First, consider the case of $[s_0, s_1] = [1, 3]$ and $r = 2n/3 - 2$. There are six elements in rows 0 and Δ_3 on these three lost strips, but only 5 surviving equations (two h-parity, three up-diagonal parity, but no down-diagonal parity). As such these elements cannot be reconstructed.

Next assume that $r = 2n/3 - 1$. If $s_0 = s_1 = 1$, then the three elements in row $\Delta_3 - 1$ have all up and down diagonals lost. Hence this is unrecoverable.

If $s_0 \neq 1$ and s_1 is any value, then the six elements in rows $s_0 - 2$ and $s_0 - 2 + \Delta_3$ have all three up-diagonals lost, which leaves at most 5 equations for the six variables. □

In a similar manner, it is not hard to prove the following upper bound.

Proposition 5. *Let the* $\mathrm{HoVer}^3_{2,1}$ *layout be defined by equations (3.4) and (3.5). If* n *is even* ($\mathrm{pr}(n) = 2$), *then* $\mathrm{r}_0(n) \leq n/2$. *If* n *is odd, then* $\mathrm{r}_0(n, [1, 2]) \leq (n - 3)/2$.

Proof. In the case that n is even and $r = n/2 + 1$, then the layout is not even 2-fault tolerant. Loss of strips 0 and $\Delta = n/2$, and so elements at rows 0 and Δ provide an underdetermined system of 4 equations in 4 variables.

For n odd, $r = (n-1)/2$, and $[s_0, s_1] = [1, 2]$, the loss of strips 0, $\Delta = (n+1)/2$ and $n + 1$ (the h-parity strip) implies that the element of strip 0, row 0 is lost, along with all of its parity (the up-diagonal and down-diagonal parities are both on strip Δ). Hence, this element cannot be reconstructed. □

The above three propositions provide proofs of all the upper-bound statements in Theorem 5 (with the exception of the special cases $n = 4, 6, 10$ which can be checked by hand). The lower-bounds are considerably more difficult (because all failure cases need to be considered). For brevity, we leave out the details of the proofs of those cases we have completed.