# Cost Analysis of the X-code Double Parity Array

Alexander Thomasian [*] and Jun Xu
*New Jersey Institute of Technology - NJIT*
*alexthomasian@gmail.com;xujun.sh@gmail.com*

## Abstract

*The popular RAID5 disk arrays tolerate a single disk failure by using a parity code to reconstruct the contents of a failed disk on demand, but are susceptible to data loss if a second disk fails. The rebuild process which systematically reconstructs the contents of a failed disk on a spare disk may be unsuccessful due to media failures or a second disk failure. Two disk failure tolerant arrays dealing with both problems can be implemented using Reed-Solomon codes or multiple parity schemes such as EVENODD, RDP, X-code, and RM2. All methods incur the minimum level of redundancy in disk accesses and also capacity overhead (except RM2). An appropriate choice of symbol sizes in EVENODD and RDP results in the same access pattern as RAID6 and little disk load imbalance in degraded mode. In this study we consider the load increase and imbalance of the X-code method, since other methods were investigated in previous studies. We derive a general expression for disk loads and present graphs to quantify the load imbalance.*

## 1. Coding for Multiple Disk Failure Tolerant Arrays

Parity coding to recover from single disk failures was part of the original RAID proposal [7], which introduced five RAID levels. The popular RAID level 5 (RAID5) utilizes striping, which partitions large files into fixed size *stripe units - SUs* which are allocated in a round-robin manner across the disks of the array. The capacity of one disk out of *N* is dedicated to parity blocks, whose SUs according to the left symmetric organization are placed in left to right repeating diagonals. Distributing the parity blocks has the advantage of balancing the disk load due to updating activity.

The redundancy due to parity in RAID5 allows the on demand reconstruction of the blocks of a single failed disk. A data access to the failed disk entails fork-join requests to the corresponding blocks on the surviving disks. The load on these disks doubles when all requests are reads, so that there is a degradation in response times. More importantly, a RAID5 disk array with a single disk failure is susceptible to data loss if another disk fails. The rebuild process in RAID5 systematically reconstructs the contents of a failed disk on the spare disk, by reading and XORing the contents of surviving disks. In spite of the very small bit error rates for the RAID5 configuration considered in [1] the rebuild process cannot be completed in 4% of cases due to *latent sector failures - LSFs*.

Two-disk failure tolerant - 2DFTs arrays classified as RAID6 [3] are a solution to this problem. There are several schemes to recover from multiple disk failures, which from a coding viewpoint correspond to erasure correction. *Maximum Distance Separable - MDS* codes meeting the Singleton bound require *r* parity disks in order to correct *r* erasures. The most common class of MDS codes are Reed-Solomon (RS) codes, which have been implemented in StorageTek's Iceberg [3] and HP's RAID 5DP (double parity) [5].

RS codes which are based on arithmetic on finite fields are quite expensive to implement [2]. Several parity based methods have been proposed to reduce the computational cost. The EVENODD code has two parity columns that are independent from each other [1]. The Row-Diagonal Parity (RDP) family of codes, optimizes the number of XORs at the encoding [4]. With an appropriate choice of symbol sizes EVENODD and RDP incur the same disk access pattern as RAID6, i.e., three read-modify-writes to update data and parity blocks. The *P* and *Q* parity SUs are organized according to the left symmetric organization to balance disk loads due to updates.

RM2 [6] and the X-code [9] are distributed parity schemes, which have an advantage over dedicated parity in that there are no bottlenecks. No disk gets more accesses than others, so that there is no need to rotate the parity columns as in RAID5 and RAID6. While RM2 allows an odd or an even number of disks, the distributed parity codes cannot be shortened in general. For instance, the X-code consists of *N* columns, where *N* is a prime. Although EVENODD requires a prime number of disks, empty virtual disks with zero contents can be used to fill the gap.

As noted earlier the load of RAID arrays in processing read requests doubles with a single disk failure and triples in RAID6 with two disk failures. The load increase in RM2 is a function of the redundancy ratio and disk loads in RM2 tend to be quite unbalanced [8]. This study concentrates on the X-code organization. We show that the load increase with two disk failures is a function of the distance of the failed disks and derive a closed form expression in this case.

This paper is organized as follows. In Section 2 we describe the X-code organization. This is followed by the analysis of access costs in Section 3. Numerical results to characterize X-code performance are given in Section 4. We conclude with Section 5.

## 2. The X-code Organization

Data which is striped across $N$ disks is placed in the first $N-2$ rows of an $N \times N$ arrays referred to segments. The two $p$ and $q$ parity SUs are stored at rows $N-1$ and $N$, respectively. $p(i)$ is the $p$ parity at the $i^{th}$ column of the $(N-1)^{th}$ row, with slope of 1 for the *parity group - PG*. $q(i)$ is the $q$ parity at the $i^{th}$ column on $N^{th}$ row with slope -1 for the PG. [1] PGs are constructed from the SUs along several diagonal by the XOR operation. Figures 1 and 2 are the data layouts for the $p$ and $q$ PGs with $N=7$.

Let $B_{i,j}$ denote a data or parity block (or SU) at the $i^{th}$ row and $j^{th}$ column. Given $<X>_N = X \mod N$ the $p(i)$ and $q(i)$ for $0 \le i \le N-1$ are computed as follows:

$$B_{N-2,i} = B_{0,<i-2>_N} \oplus B_{1,<i-3>_N} \oplus \dots \oplus B_{N-3,<i-N+1>_N}.$$

$$B_{N-1,i} = B_{0,<i+2>_N} \oplus B_{1,<i+3>_N} \oplus \dots \oplus B_{N-3,<i+N-1>_N}.$$

For example, $B_{1,2}$ is protected by $p(3)$ in Figure 1 and $q(6)$ in Figure 2. Each parity group contains $N-1$ blocks, including the parity, so that the recovery of a block with a single disk failure requires $a = N-2$ disk accesses. Parity blocks unlike data blocks are protected by one PG, but the data SUs in each segment are protected by different $p$ and $q$ parities.

## 3. Cost Analysis

We evaluate access costs for the Xcode method with an *online transaction processing - OLTP* workload, i.e., accesses to small randomly placed data blocks. Updates associated with OLTP workloads incur the costly small write penalty, i.e., the updating of two parity blocks in addition to modified data block. We use the following notation for access costs [8]: (i) $D_{SR}$ cost of read access; (ii) $D_{SW}$ cost of write access; (iii) $D_{RMW}$ read-modify-write accesses for updating data and parity blocks. RMWs can be implemented as a read followed by a write after one disk rotation or an independent write request following a read.

The access cost in normal mode is the same as RAID6. $D_{SR}$ for read requests and $3D_{RMW}$ for write requests. In what follows we obtain the access costs with one and two disk failures.

[1]Note that the slopes defined in this paper are different from [9].

|  | Disk Number |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|
| Row | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | 2 | 3 | 4 | 5 | 6 | 0 | 1 |
| 2 | 3 | 4 | 5 | 6 | 0 | 1 | 2 |
| 3 | 4 | 5 | 6 | 0 | 1 | 2 | 3 |
| 4 | 5 | 6 | 0 | 1 | 2 | 3 | 4 |
| 5 | 6 | 0 | 1 | 2 | 3 | 4 | 5 |
| 6 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 7 |  |  |  |  |  |  |  |

**Figure 1. Parity groups for $p(i), 0 \le i \le 6$ with slope=1 and $N=7$.**

|  | Disk Number |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|
| Row | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | 5 | 6 | 0 | 1 | 2 | 3 | 4 |
| 2 | 4 | 5 | 6 | 0 | 1 | 2 | 3 |
| 3 | 3 | 4 | 5 | 6 | 0 | 1 | 2 |
| 4 | 2 | 3 | 4 | 5 | 6 | 0 | 1 |
| 5 | 1 | 2 | 3 | 4 | 5 | 6 | 0 |
| 6 |  |  |  |  |  |  |  |
| 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |

**Figure 2. Parity groups for $q(i), 0 \le i \le 6$ with slope=-1 and $N=7$.**

### 3.1. Access Costs with One Failed Disk

Data blocks to be read may be available or not. Given that all disks have the same number of blocks the fraction of available (resp. unavailable) data blocks is $f_1 = (N-1)/N$ (resp. $f_2 = 1/N$).

1. The block is available.

$$f_1 = \frac{N-1}{N} \Rightarrow C_{1F}^{Read/A} = D_{SR}.$$

2. The data block is unavailable.

$$f_2 = \frac{1}{N} \Rightarrow C_{1F}^{Read/U} = (N-2)D_{SR}.$$

Either the $p(i)$ or $q(i)$ parity can be used to reconstruct the unavailable data block.

The mean read cost is a weighted sum of the two cases.

$$C_{1F}^{Read} = f_1 C_{1F}^{Read/A} + f_2 C_{1F}^{Read/U} = \frac{2N-3}{N} D_{SR}. \qquad (1)$$

There are three cases when a data block is to be written.

1. Data and parity blocks are all available.
$$f_1 = \frac{N-3}{N} \Rightarrow C_1 = 3D_{RMW}.$$
2. The data block and one of two parity blocks is available.
$$f_2 = \frac{2}{N} \Rightarrow C_2 = 2D_{RMW}.$$
3. Data block is unavailable.
$$f_3 = \frac{1}{N} \Rightarrow C_3 = (N-3)D_{SR} + 2D_{RMW}.$$
We read data blocks and parity in one of the parity groups (say PG 1) to reconstruct $d_{old}$, to compute $d_{diff}$, and compute $p^1_{new} = p^1_{old} \oplus d_{diff}$. $d_{diff}$ computed in this manner is applied to both parity blocks, which are read and written as RMW accesses.

The mean write cost is the weighted sum of the three cases.

$$C_{1F}^{Write} = \sum_{i=1}^{3} f_i C_i = \frac{N-3}{N} D_{SR} + \frac{3(N-1)}{N} D_{RMW}. \qquad (2)$$

## 3.2. Access Costs with Two Failed Disks

For read cost with two failed disks there are two cases depending on whether the target block is available or not.

1. The block is available.
$$f_1 = \frac{N-2}{N} \Rightarrow C_{2F}^{Read/A} = D_{SR}.$$
2. The block is unavailable.
$$f_2 = \frac{2}{N} \Rightarrow C_{2F}^{Read/U}.$$

The read cost for an unavailable data block depends on the distance $d$ of the two failed disks $i$ and $j$, which is defined as follows:

$$d = min(j-i, i+N-j), \quad 1 \le i \le j \le N.$$

For example, the distance of disk 1 and disk 6 for $N = 7$ is 2, not 5. Given that $N$ is prime and an odd number, the maximum minimum distance is $(N-1)/2$.

The cost to read an unavailable data block varies according to the number of PGs that need to be read to reconstruct a data block. The access cost is a multiple of $a = N-2$, since each PG has $N-1$ blocks, one of which is the parity. Figures 3 and 4 can be used to determine the reconstruction cost with distance $d = 1$, e.g., disks 1 and 2 failed. $B_{3,1}$ cannot be reconstructed directly using PG $p(3)$ or $q(4)$. We need to reconstruct $B_{2,2}$ for $p(4)$ or $B_{4,2}$ for $q(3)$ first. So the cost to reconstruct $B_{3,1}$ is access to the blocks in one PG plus the minimum of costs to reconstruct $B_{2,2}$ or $B_{4,2}$. We repeat the analysis for $B_{2,2}$ and $B_{4,2}$ until we reach a block that can be reconstructed by one PG access. It is not possible to reconstruct $B_{4,2}$, because it needs $B_{5,1}$, which has two unavailable blocks (one of which is parity $q(1)$). $B_{2,2}$ needs $B_{1,1}$ which can be reconstructed by one PG access. The total cost for $B_{3,1}$ is accesses to data blocks in PGs $p(2)$, $q(5)$, and $p(4)$, i.e., $3(N-2)$ disk accesses. Table 1 displays the recovery path and read cost of all unavailable data blocks with $N = 7$ and $d = 1$.

| $slope = 1$ | Disk Number | | | | | | |
|---|---|---|---|---|---|---|---|
| Row # | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | 2 | 3 | 4 | 5 | 6 | 0 | 1 |
| 2 | 3 | 4 | 5 | 6 | 0 | 1 | 2 |
| 3 | 4 | 5 | 6 | 0 | 1 | 2 | 3 |
| 4 | 5 | 6 | 0 | 1 | 2 | 3 | 4 |
| 5 | 6 | 0 | 1 | 2 | 3 | 4 | 5 |
| 6 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 7 | | | | | | | |

**Figure 3.** Figure to determine recovery paths for blocks on the failed disks with the $p$ PGs, $d = 1$, and $N = 7$.

| $slope = -1$ | Disk Number | | | | | | |
|---|---|---|---|---|---|---|---|
| Row # | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | 5 | 6 | 0 | 1 | 2 | 3 | 4 |
| 2 | 4 | 5 | 6 | 0 | 1 | 2 | 3 |
| 3 | 3 | 4 | 5 | 6 | 0 | 1 | 2 |
| 4 | 2 | 3 | 4 | 5 | 6 | 0 | 1 |
| 5 | 1 | 2 | 3 | 4 | 5 | 6 | 0 |
| 6 | | | | | | | |
| 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |

**Figure 4.** Figure to determine recovery paths for blocks on the failed disks with the $q$ PGs, $d = 1$, and $N = 7$.

The read cost for data blocks on a failed disk has the pattern in Table 2. There are $(N-1)/2$ rows where each row represents the distance $d$ and the columns represent

| The list of parity groups needed to reconstruct an unavailable data block | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | ... | ... | $N-3$ | $N-2$ |
| 1 | 1 | 2 | 3 | ... | ... | $N-4$ | $N-3$ |
| 1 | 2 | 1 | 2 | ... | ... | $N-5$ | $N-4$ |
| 1 | 2 | 3 | 1 | ... | ... | $N-6$ | $N-5$ |
| | | | | ... | | | |
| 1 | 2 | ... | $(N-1)/2-2$ | 1 | 2 | ... | $(N-1)/2+1$ |
| 1 | 2 | 3 | ... | $(N-1)/2-1$ | 1 | ... | $(N-1)/2$ |

**Table 2. Read costs for data blocks $N-2$ on one failed disk with different distance and $N$ disks. Each number times $(N-2)D_{SR}$ is the cost of reconstructing a data block. The $(N-1)/2$ rows in the table do not represent the distances for failed disks.**

| Block | Recovery path | Cost |
|---|---|---|
| (1,1) | p(2) | 1 |
| (2,1) | q(6) + p(3) | 2 |
| (3,1) | p(2) + q(5) +p(4) | 3 |
| (4,1) | q(6) + p(3) + q(4) + p(5) | 4 |
| (5,1) | p(2) + q(5) + p(4) + q(3) + p(6) | 5 |
| (1,2) | q(6) | 1 |
| (2,2) | p(2) + q(5) | 2 |
| (3,2) | q(6) + p(3) +q(4) | 3 |
| (4,2) | p(2) + q(5) +p(4) +q(3) | 4 |
| (5,2) | q(6) + p(3) + q(4) + p(5) + q(2) | 5 |

**Table 1. Read cost for the failed disks 1 and 2 with $N=7$ and $d=1$ (cost is a multiple of $(N-2)D_{SR}$).**

the $N-2$ data blocks on failed disk. The read cost for the blocks on a failed disk for a given distance $d$ is the sum of the unshadowed and shadowed portions of the rows in Table 2:

$$C_d = (N-2)[\sum_{i=1}^{N-1-d} i + \sum_{j=0}^{d-1} j]D_{SR}.$$

Given that there are $N-2$ data blocks on each disk, the mean read cost per data block on a failed disk is:

$$C_d^{Read/F} = \frac{C_d}{N-2} = [\sum_{i=1}^{N-1-d} i + \sum_{j=0}^{d-1} j]D_{SR}.$$

The mean cost is the sum of costs for all distances $d$ divided by the number of possible distances $a=(N-1)/2$.

$$C_{2F}^{Read/U} = \frac{1}{a}\sum_{d=1}^{(N-1)/2} C_d^{Read/F} =$$

$$\frac{1}{a}\sum_{d=1}^{(N-1)/2}[\sum_{i=1}^{N-1-d} i + \sum_{j=0}^{d-1} j]D_{SR} = \frac{4a^2-1}{3}D_{SR}.$$

Substituting $a$ with $(N-1)/2$ and simplifying we have:

$$C_{2F}^{Read/U} = \frac{N^2-2N}{3}D_{SR}. \quad (3)$$

Given a fraction $f_1=(N-2)/N$ of data blocks are available and a fraction $f_2=2/N$ unavailable, then the average cost to access a data block is:

$$C_{2F}^{Read} = f_1 C_{2F}^{Read/A} + f_2 C_{2F}^{Read/U} = \frac{2N^2-N-6}{3N}D_{SR}. \quad (4)$$

There are five cases for writes depending on the availability of the data and two parity blocks. $D$, $P$, and $Q$ indicate that data and parity blocks are available, while $\overline{D}$, $\overline{P}$ and $\overline{Q}$ indicate that they are not, since the corresponding disk is failed. The costs and frequencies for all cases are:

1. All three blocks are available.

$$f_1 = \frac{(N-3)(N-4)}{N(N-1)} \Rightarrow C_1 = 3D_{RMW}.$$

2. Data and one of the parity blocks are both available.

$$f_2 = \frac{4(N-3)}{N(N-1)} \Rightarrow C_2 = 2D_{RMW}.$$

3. Data block is available, but both parity blocks are unavailable.

$$f_3 = \frac{2}{N(N-1)} \Rightarrow C_3 = D_{SW}.$$

4. Data block is unavailable, but both parity blocks are available.

$$f_4 = \frac{2(N-3)}{N(N-1)} \Rightarrow C_4 = \frac{N^2-2N-3}{3}D_{SR} + 2D_{RMW}.$$

5. Data and one of the parity blocks are unavailable.

$$f_5 = \frac{4}{N(N-1)} \Rightarrow C_5 = \frac{N^2-2N-3}{3}D_{SR} + D_{RMW}.$$

In cases 4 and 5 when the data block is not available, we treat it as a read request and reconstruct it at a cost given by Equation 3. If both parities survive they are written as $2D_{RMW}$ minus $D_{SR}$. If there is one surviving parity it is written as a $2D_{RMW}$ minus $D_{SR}$.

The overall mean cost for write operation with two disks failure is then the mean over the five cases.

$$C_{2F}^{Write} = \sum_{i=1}^{5} f_i C_i = \frac{2}{N(N-1)} D_{SW} +$$

$$\frac{2(N-3)(N+1)}{3N} D_{SR} + \frac{(3N^2-9N+4)}{N(N-1)} D_{RMW}.$$

The cost of operation for an Xcode disk array is summarized in Table 3.

## 4. Numerical Results

The results reported in this section were obtained via enumeration. Given the number of disks $N$, the number of failed disks, and the address of a block to be accessed, we determined the number of disk accesses. In the case of two disk failures the distance of the failed disks ($d$) needs to be specified. In the case of a single disk failure the rebuild cost per disk is $(N-2)^2$. In the case of a double disk failure at any distance the $2(N-2)$ unavailable blocks at the two failed disks can be recovered by accessing all $N(N-2)$ SUs. The parity SUs on the failed disks can be reconstructed from data blocks.

To determine disk access costs we systematically reconstruct all disk blocks (SUs more precisely), although each block access is treated as an individual request, as if we have random disk accesses. A recursive algorithm written for this purpose determines the least cost to access each block. It is observed that $p$ and $q$ PGs are utilized alternatively with two disk failures, although some blocks can be reconstructed by accessing one PG.

Figure 5 shows the mean read cost over all data blocks with two failed disks versus $N$ using Equation (4). The cost is almost linear in $N$.

Figure 6 shows the load increase on each disk with different distances for $N = 7$ and two failed disks. The X axis denotes the disk index, the Y axis denotes the distance ($d$), and the Z axis is the factor by which the load is increased. For example, the maximum load increase on disk one with $d = 1$ is 26 accesses, besides its own 5 read accesses. Hence the load increase is 6.2 folds of the normal load.

The mean load increase on surviving disks varies with $d$ and is the largest for $d = 1$. Figure 7 shows the mean load increase on surviving disks for varying distances for $N = 37$ and two disk failures. The mean load increase of a disk for a given distance is computed as follows. First

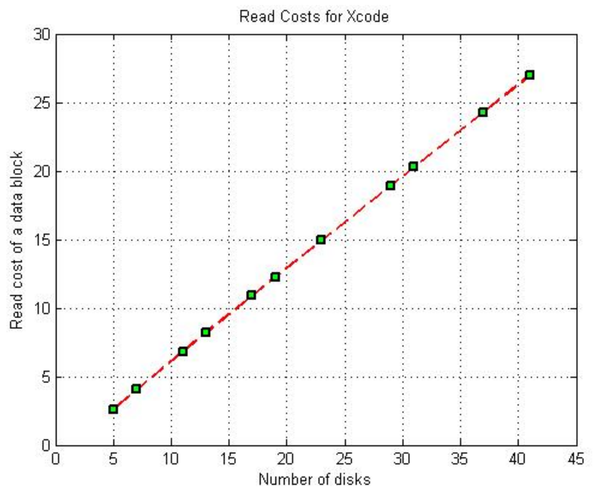

**Figure 5.** Mean read cost of a data block with two failed disks versus number of disks (the number needs to be a prime).

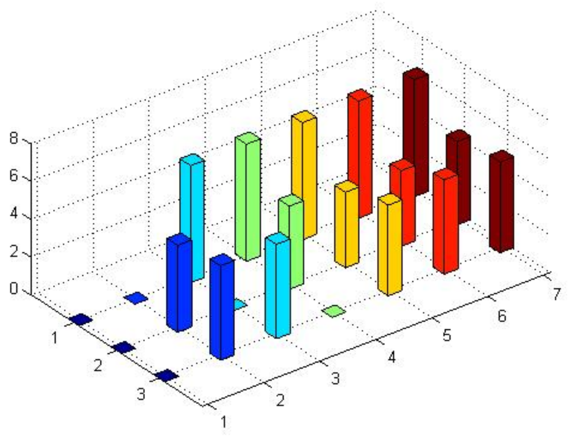

**Figure 6.** Load increase on each disk for reads with two failed disks and $N = 7$. X axis is $i^{th}$ disk, Y axis is the distance. Z axis is the times of load increase comparing that in the normal mode.

obtain load increase on each surviving disk as discussed above. Then obtain the mean of the load increase on each surviving disk to get the mean load increase on a disk with a given distance.

Given the sensitivity of load increase to the positions of the failed disks a nearly random permutation method applied at the level of segments can be used to balance disk loads. Successive segments are assigned increasing segment numbers which serve as seeds to a pseudo-random number generator to permute the columns of a segment.

| Mode | Read | Write |
|---|---|---|
| 0 | $D_{SR}$ | $3D_{RMW}$ |
| 1 | $\frac{2N-3}{N}D_{SR}$ | $\frac{N-2}{N}D_{SR}+\frac{3(N-1)}{N}D_{RMW}$ |
| 2 | $\frac{2N^2-N-6}{3N}D_{SR}$ | $\frac{2}{N(N-1)}D_{SW}+\frac{2(N-3)(N+1)}{3N}D_{SR}+\frac{(3N^2-9N+4)}{N(N-1)}D_{RMW}$ |

**Table 3. Cost of operation for Xcode with $N$ disks. Modes 0, 1, and 2 correspond to no, single, and double disk failures, respectively.**

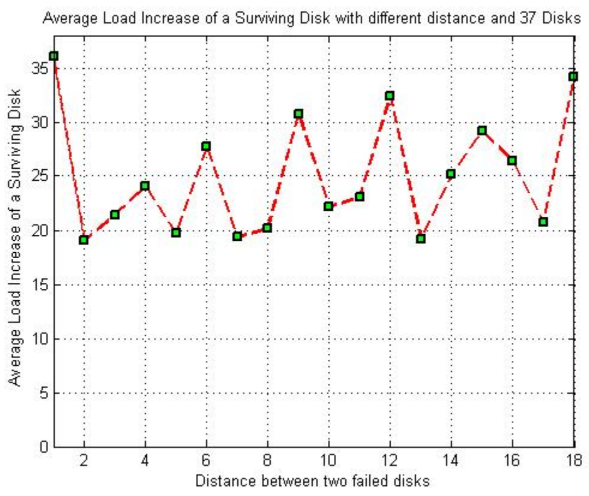

**Figure 7.** Mean load increase on surviving disks for reads with two disk failures and $N = 37$.

## 5. Conclusions

The X-codes method exhibits the same cost as RAID6 when operating in normal mode. Similarly to RAID6 the load in normal mode is almost doubled with a single disk failure. With two disk failures the read cost for unavailable blocks increases quadratically with $n$, while the overall cost increases linearly with $N$.

The load imbalance can be eliminated by assigning sequence numbers to consecutive segments ($N \times N$ array) and applying a random permutation based on the sequence number. A similar technique was applied to the RM2 method in [8].

## References

[1] M. Blaum, J. Brady, J. Bruck, and J. Menon. "EVEN-ODD: An Efficient scheme for tolerating double disk failures in RAID architectures", *IEEE Trans. Computers C-44*(2): 192–202 (Feb. 1995).

[2] M. Blaum. "An introduction to error-correcting codes", *Coding and Signal Processing for Magnetic Recording Systems*, B. Vasic and E. M. Kurtas, editors, Chapter 9, CRC Press, Orlando, FL, 2005.

[3] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson. "RAID: High-performance, reliable secondary storage". *ACM Computing Surveys 26*(2): 145–185 (June 1994).

[4] P. F. Corbett, B. English, A. Goel, T. Grcanac, S. Kleiman, J. Leong, and S. Sankar. "Row-diagonal parity for double disk failure correction", *Proc. 3rd USENIX Conf. File and Storage Technologies - FAST'04*, San Francisco, CA, March/April 2004.

[5] Hewlett-Packard, "An analysis of RAID 5DP", *HP White Paper*, `http://www.hp.com`.

[6] C.-I. Park, "Efficient placement of parity and data to tolerate two disk failures in disk array systems", *IEEE Trans. Parallel and Distributed Systems 11*(6): 1177-1184 (Nov. 1995).

[7] D. A. Patterson, G. A. Gibson, and R. H. Katz. "A case for redundant arrays of inexpensive disks - RAID". *Proc. ACM SIGMOD Int'l Conf. Management of Data*, Chicago, IL, June 1988, pp. 109-116.

[8] A. Thomasian, G. Fu, and C. Han, "Performance of two disk failure tolerant disk arrays", *IEEE Trans. Computers 56*(6): 799-814 (June 2007).

[9] L. Xu and J. Bruck. "X-Code: MDS array codes with optimal encoding", *IEEE Trans. Information Theory 45*(1): 272–276 (Jan. 1999).