2015 The 5th International Workshop on Computer Science and Engineering
doi: 10.18178/wcse.2015.04.106

Double-Erasure Correcting Code Methods for Parity Disk to Recover Failure Data

Ghulam Muhammad Shaikh[1, a], Quanxin Zhang[1, b], Qing Mu[1, c], Fei Wu[2, d] and Yu-an Tan[1, e]

[1]Beijing Engineering Research Center of Massive Language Information Processing and Cloud Computing Application, School of Computer Science and Technology, Beijing Institute of Technology, Beijing, 100081, China

[2]Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, Wuhan, 430074, China

[a]gm.shaikh1@gmail.com, [b]zhangqx@bit.edu.cn, [c]jennifermu@bit.edu.cn, [d]wufei@mail.hust.edu.cn, [e]tan2008@bit.edu.cn

Keywords: Redundant array of Independent Disk (RAID), Parity Disk Architecture, Parity in *t* dimension Double-Erasure Correcting Code with proved theorem.

Abstract. In a redundant disk array the protection of data against disaster disk failure is very critical. Nowadays, single disk is highly responsible when a redundant disk array failure occurs. The results saving several other disks and reduce the probability of failure that at least one disk will fail. The data is distributed on several disks and the distributed data are store an independently. We present a double-erasure correcting code with proved theorem and evaluate code to recover a failure data from parity disk.

Introduction

Redundant Array of Independent Disk (RAID) has been a vast area of research for the past several years. The configuration and the concept of RAID were proposed for reliable storage of data and it is a kind of disk redundant technology which was introduced by David Patterson[1]. RAID has relatively high performance of writing or reading data with its own distinctive characteristics and the performance can be improved as array provides confident level of fault tolerance.

This research shows that the problem of designing double erasure-correcting code and proved theorems that secures against the loss of data caused by disk failure or parity disk. It is not uncommon for multiple disks to fault simultaneously and results are paralysis in system. It is risky to take corresponding disk fault tolerating mechanisms for rapid recovery and repair of classified data.

This paper is organized as follows: In Section 2, RAID is explained with the previous work on disk parity. Afterwards, parity in *t* Dimension is presented in Section 3. Then, storage erasure codes are briefly defined in Section 4. Finally, results and discussions and conclusions are presented in section 5 and 6 respectively.

Raid and Disk Parity

Many organizations design the RAID storage subsystem to increase parallelism. This trend has been past several years, the system can and will perform better in terms of I/O by increasing the number rather than the performance of individual disks. This is generally part of a RAID in which one or more disk drives are connected together to act as a single system[2].

The disk array consists of multiple disk drives having a relatively small capacity. Small piece of write data transferred from the CPU is subdivided into a multiple drives in a parallel mode. Conversely, when data is read out the available data are combined which will be transferred to the CPU. It should also be noted that a group of multiple data will be called a "parity group". In this specification, this terminology will be employed in such a case when the error correction data does not correspond to parity data. This parity data is used to recover data stored in one drive where a fault

occurs. Since the probability of the fault occurrence is increased due to an increased number of components, such parity is prepared to improve the reliability of the disk array[3][4].

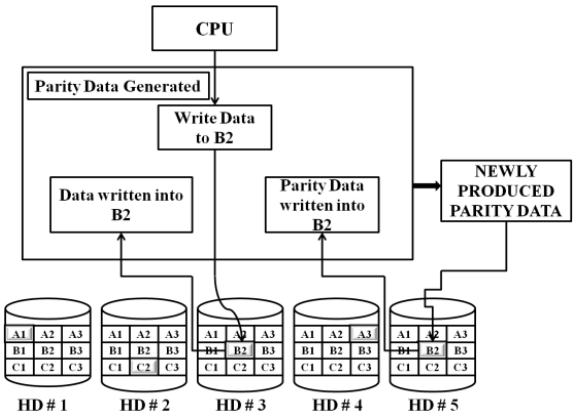

Fig. 1: Descriptive diagram of the conventional redundant disk array

In Fig. 1[5], the addresses fixed. It is absolutely required to set the parity data in order to improve the reliability of this system. In above array, the parity data is produced by the data at the same address A1 in the multiple drives, and then the resultant parity data is stored into the address A1 of the drive used to store the parity data. The relevant data in the respective drives is to be accessed when read or write process is performed. Similarly, when data is to be written on an address B2 of the drive # 3 and the old parity data is stored at the address B2 of the parity drive #5. These read data are exclusive-OR gate with the data to be written to newly produced parity data[6]. After the parity data has been formed, the write data is stored into the address B2 of the drive # 3 and the new parity data is stored into the address B2 of the drive # 5.

Parity in t Dimension

The erasure-correcting code in large disk arrays proposed a method for recovering lost data. The single-erasure-correcting scheme divides up the information disks into sets of *G* disk, and to connect a parity check disk with each of those sets. In Fig. 2 we show these *G* disks (for *G* =4) in a row with parity in the disk on the right side. This coding scheme has an overhead of 1/ *G*, an updated penalty of 1 (since one check disk update is needed for every information disk update), and has a group size *G* +1 (since *G* +1 disk are involved in the reconstruction of any failure). Two failures in any group leads to lost data. The method of reliability, which we call *1disk-parity* is shown in Fig. 2(a) [7].

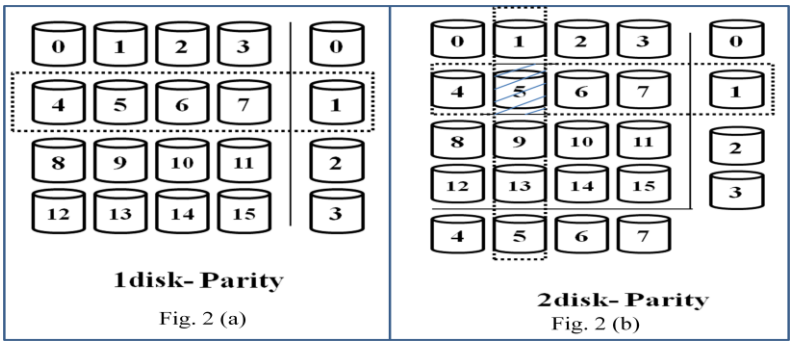

Fig. 2: The parity can be compute as horizontally and vertically is called 1disk parity and 2disk parity

A simple extension of 1disk-parity called *2disk-parity* which is shown in Fig. 2(b) (for *G*=4). A set G^2 information disks are arranged in a two dimensional array[8]. On one end of each row and column a check disk stores parity for that row or column. Since failed disks belong to two potential group row

and columns, it can be reconstructed from the data. Thus 2d-parity is double-erasure-correcting. This coding scheme has a check disk overhead of $2G/G^2$, an updated penalty of 2 and group size of G+1. These coding schemes have a check disk overhead of $tG^{(t-1)}/G^t$=t/G, since there are $tG^{(t-1)}$ check disks and G^tinformation disks. However, the group size remains G+1 because only G+ 1 disk are involved in the reconstruction of any single failure. The *t* disk-parity coding scheme are a member of the class of product codes that have been commonly used in magnetic tape systems. A common but expensive technique for protecting disk system from disk failures is known as shadowing. A shadowing code is equivalent to *t* disk-parity code with G =1 because the parity of single bit duplicates the value of that bit.

We do not envision practical disk arrays large enough to require *t* greater than 2 or 3, because other sources of the performance degradation associated with increasing numbers of check disk updates. Although *t* disk-parity has an easily visualized structure, it is necessarily the best coding scheme for our research. In the next section we discuss more about Double-erasure code with proved theorem that how to implement on updated parity on a failure data[9].

Double-Erasure correcting code

In our redundant disk array update penalty is the dominant performance cost. We restrict our attention to codes that minimize it. Since any code that corrects *t* erasure must leave evidence of every write on at least *t* different check disks its minimum update penalty is *t*. Thus, in designing good *t*-erasure-correcting codes, we require that the column corresponding to each information disk has weight exactly *t*[10].

Within the class of double or 2-erasure-correcting code with minimum update penalty (the information columns of H have weight 2 or 3, respectively). Present in below section, double-erasure-correcting codes that are designed to achieve high reliability when fault occurs. We also prove theorems.

High Reliable Double-erasure Correcting Code

Theorem 1: if a code is 2-erasure correcting that has minimum update penalty, and corrects all sets of 3-eraure except bad 3-erasures, then its check disk overhead is at least 4/c, where c is the number of check bits[11][12].

Proof: consider any 2-erasure-correcting code with c check bits and minimum update penalty that corrects all sets of more than two erasures excepts dreadful erasures. We will prove that the minimum number of information bits in such a code is $c^2/4$. Therefore, the check disk overhead is least c/(c^2/4)=4/c.

The parity check matrix H = [$P \mid I$] of this code can be represented as a graph. The graph contains c vertices, which correspond to the c rows of P. The columns of P have weight two. The graph contains an edge two vertices v_1 and v_2 if and only if there exits a column in P that contain 1's in the row corresponding to $v_1$andv_2. Suppose, for the purpose of contradiction, that the graph contains a clique of size three. The vertices of the clique correspond to three rows of P. the edges correspond to three columns of P, let the three rows of P be i_1 , i_2 and i_3, and the corresponding vertices be v_1 , v_2 and v_3. Let the three column of P be j_1 , j_2 and j_3, and the corresponding edges be e_1 , e_2 and e_3,. Without loss of generality, assume ej_1= (vi_1 .vi_2), ej_2=(vi_1 .vi_3) and ej_3= (vi_2 .vi_3). Consider columnj_1. Since j_1 corresponds to the edge ej_1 between $vi_1$and vi_2, it follows from the definition of the graph that the only 1's in column j_1 appear in rows i_1 andi_2. Similarly, the only 1's in column j_2 appear in rows $i_1$and i_3 and the only 1's in column j_3 appear in rows i_2 and i_3. It follows that the sum of columns j_1 , j_2 and j_3 (mod 2) is the zero vector. Therefore, columns j_1 , j_2 and $j_3$constitute an uncorrectable 3-erasure that is not bad. It follows that the graph corresponding to a code's parity check matrix cannot contain a clique of size three. By Turan's theorem See, for example, (32) (33) a graph with c vertices that does not contain at most $c^2/4$ edges. Therefore the graph corresponding tp the parity check can contain at most $c^2/4$ edges. Since each edge corresponds to a column of P, and each column of P corresponds to an information bit, the code has at most $c^2/4$ information bits.

Follows that the check disk overhead at least 4/*c*. We have used groups with 4 information disks and marked the groups that contain data disk .

The 2disk Parity Code is Double-erasure Correcting Code and has Minimal Update Penalty.

Theorem 2 the 2disk-parity code

Proof: The description of the 2disk-parity code. Some of the check disks compute parity along rows, and some compute parity along columns. Each information disk is checked by a "row check disk," and a "column check disk." To show that 2disk-parity corrects all sets. We must show that any set of three columns of $H_{2diskparity} = [P_{2diskparity} \mid I]$ which does not correspond to an information bit and its two check bits, is linearly independent[13][14].

For simplicity, assume that the top *c*/2 rows of $P_{2diskparity}$ correspond to the row check disk, and the bottom *c*/2 rows of $P_{2diskparity}$ correspond to the column. Then $P_{2diskparity}$consist of all possible column of weight 2 with the property that one of the 1's in the column occur in the first *c*/2 rows, and the other occurs in the last *c*/2 rows. Consider any two columns of $P_{2diskparity}$. If the two columns do contain a 1 in a common row, then their sum will be a column which either contains 1's in its first *c*/2 rows, or two 1's in its last *c*/2 rows. It follows that the sum of any two or three columns of $P_{2diskparity}$ cannot be the zero vector, and the sum of the two columns of $P_{2dparity}$ and one column of *I* cannot be the zero vector, and the sum of two columns of P2diskparity and one column of *I* cannot be the zero vector either. A set of two columns from I and one column from $P_{2diskparity}$ can only sum to zero if the columns correspond to an information disk and its two associated check disk. Finally, any three columns of *I* are linearly independent. The check disk overhead of the 2d-parity code is *c*/4, where *c* is the number of check bits. Thus the 2d-parity code achieves the optimal check disk overhead possible for 2-erasure correcting codes with minimum update penalty that correct all sets of 3-erasure excepts bad 3-erasures.

Code	Updated Penalty	Check Disk Overhead	Group Size	Daily Repair	Weekly Repair
Non-redundant	**0**	**0%**	**0**	**0.003**	**0.003**
1disk-parity	**1**	**0.50%**	**5**	**1.05**	**0.015**
2disk-parity	**2**	**3.10%**	**16.05**	**12,500**	**14**

Table 1: Codes for about 1000 information disk on daily and weekly basis

Results and Discussions

Disk arrays are a promising solution to the increasing demand for I/O bandwidth and access parallelism. Fig. 4 and Fig. 5 shows a execution for repair is periodic visits (for example, daily and weekly) by maintenance personnel. With this model the mean time to data loss (MTTDL) is the expected number of repair periods until an unrecoverable set of failures occurs. Using independent, exponential disk lifetimes with mean *M*, we can calculate the probability of y failures (erasures) in a repair period *T* [15][16].

$$\binom{N}{y}(1-e^{-\frac{T}{M}})^{y}(e^{-T/M})^{(N-y)} \quad (1)$$

where *N* is the total number of disks. We have used Monte Carlo simulation on the columns of code's parity check matrix to estimate the fraction of *y* failures (*y*-erasure) that are unrecoverable (linearly

dependent) in each of our sample codes. The mean number of repair periods until data loss is then just the reciprocal of the probability that an unrecoverable failure occurs in a single repair period.

For our simulation we use codes that have close to 1000 information disks and we assume inexpensive disks; that disks is with a mean time to failure of 12,500 hours. Our results are shown in Table 1. The updated penalty is the number of additional accesses to check disks for each information disk write. The daily and weekly repair model on Monte Carlo simulation has the probability to set failures at the end of the unrecoverable repair period.

First, these results reaffirm our introductory comments about single-erasure-correcting, the 1disk-parity code is less reliable than a single disk, even if repair is daily. So we turn to double erasure-correcting codes and pay the additional check disk update penalty on every write. The 2disk-parity code has many times the reliability of a single disk, even if it is weekly. The 2-disk parity codes do better in overhead and still have very good reliability.

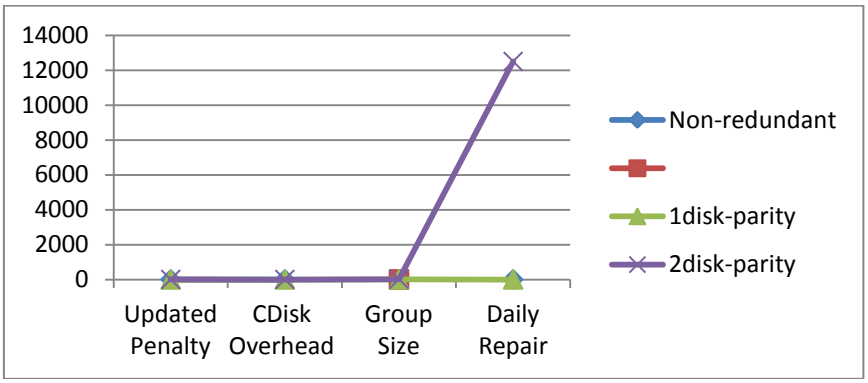

Fig. 4: shows a execution of repair when periodic visits daily

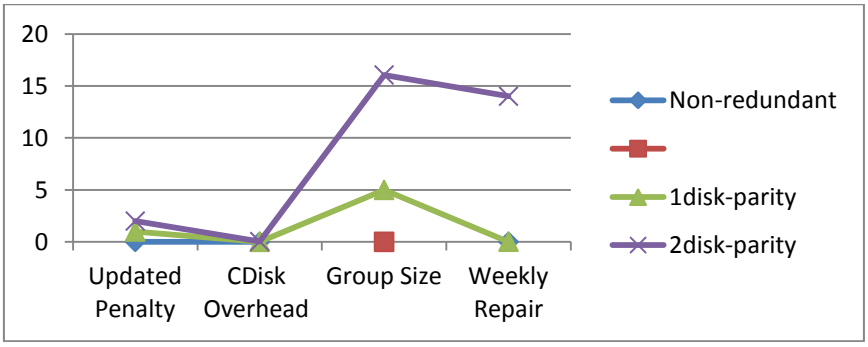

Fig. 5: shows a execution of repair when periodic visits weekly

Conclusion

Disk arrays are a promising solution to the increasing demand for I/O bandwidth and access parallelism. If high reliability is to be preserved as the size of these arrays grows, redundancy encodings may be required guarantee of failures.

In this paper we explain that implementation of redundancy codes for the practical control of disk array. In Table 1, the characteristics of the main code which is discussed and the number of check disks that can be updated whenever information needs to be updated. By using the Monte Carlo simulator, the reliability of explained double-erasure-correcting code for arrays about 500 information disks is so good and 2-erasure correction is achieved.

Acknowledgement

We would like to acknowledge and appreciate the help extended during language editing of the paper by Mr. Zahid Ali Memon, PhD scholar, School of Management and Economics, Beijing Institute of

Technology (BIT), China. This work is supported by 863 program of China(No.2013AA01A212) and Natural Science Foundation of China(No.61370063and 61300047).

References

[1] P. David, G. Gibson and R.H. Katz: *A case for redundant arrays of inexpensive disks (RAID)*. Vol. 17. No. 3. ACM, (1988)

[2] B.R, T. Cortes, and H. Jin: "A Case for Redundant Arrays of Inexpensive Disks (RAID)." (2009): 2-14.

[3] S. Yasushi et al. "FAB: building distributed enterprise disk arrays from commodity components." *ACM SIGARCH Computer Architecture News*. Vol. 32. No. 5. ACM, (2004)

[4] R. David and T. Aaron Gulliver: "Single parity check product codes." *Communications, IEEE Transactions on* 49.8 (2001): 1354-1362.

[5] D. Alexandros et al: "Network coding for distributed storage systems." *Information Theory, IEEE Transactions on* 56.9 (2010): 4539-4551.

[6] L. Xiaozhou, M. Lillibridge and M. Uysal: "Reliability analysis of deduplicated and erasure-coded storage." *ACM SIGMETRICS Performance Evaluation Review* 38.3 (2011): 4-9.

[7] T. Alexander and M. Blaum: "Higher reliability redundant disk arrays: Organization, operation, and coding." *ACM Transactions on Storage (TOS)* 5.3 (2009): 7.

[8] B. Mario, J.L. Hafner and S. Hetzler: "Partial-MDS codes and their application to RAID type of architectures." *Information Theory, IEEE Transactions on* 59.7 (2013): 4510-4519.

[9] C. Peter et al: "Row-diagonal parity for double disk failure correction."*Proceedings of the 3rd USENIX Conference on File and Storage Technologies*. (2004)

[10] G. Stuart and K. Kochanek: "Dynamic programming and the graphical representation of error-correcting codes." *Information Theory, IEEE Transactions on* 47.2 (2001): 549-568.

[11] L. Yi, X. Zhou, and C. Wu: "Two-and three-dimensional topological insulators with isotropic and parity-breaking Landau levels."*Physical Review B* 85.12 (2012): 125122.

[12] W. Gang et al: "Construct double-erasure-correcting data layout using P 1 F." *Dianzi Xuebao(Acta Electronica Sinica)* 34.12 (2006): 2447-2450.

[13] F. Bin et al: "DiskReduce: RAID for data-intensive scalable computing."*Proceedings of the 4th Annual Workshop on Petascale Data Storage*. ACM, (2009)

[14] F. Bin et al: "Diskreduce: Replication as a prelude to erasure coding in data-intensive scalable computing." *Proceedings of the International Conference for High Performance Computing Networking, Storage and Analysis (SC'11)*. (2011)

[15] P. Nattakan et al: "A new class of MDS erasure codes based on graphs." *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*. IEEE, (2009)

[16] G. Kevin M, J. S. Plank, and J.J. Wylie: "Mean time to meaningless: MTTDL, Markov models, and storage system reliability."*Proceedings of the 2nd USENIX conference on Hot topics in storage and file systems*. USENIX Association, (2010)