Constructing Double- and Triple-erasure-correcting Codes with High Availability Using Mirroring and Parity Approaches*

Gang Wang, Xiaoguang Liu, Sheng Lin, Guangjun Xie, Jing Liu
Dept. of Computer, College of Information Technical Science,
Nankai University, 300071, Tianjin, China
wgzwp@163.com

Abstract

With the rapid progress of the capacity and slow pace of the speed/MTTF of hard disks, and increasing size of storage systems, the reliability and availability of storage systems become more and more serious. This paper discusses the method of constructing double- and triple-erasure-correcting codes via combining mirroring and parity approaches in details, and presents a double-erasure code MPDC and a triple-erasure code MPPDC based on one-factorizations of complete graphs. The two codes are simple, easy to implement, and have no disk number limitation. They achieve perfect fault-free load balance and approximately optimal reconstruction load balance. The simulation results show that, compared with other double- and triple-erasure codes, MPDC and MPPDC have comparative light-load and moderate-load performance and better heavy-load performance in fault-free mode. Because parity declustering is used, the two codes are far superior to the other double- and triple-erasure codes in degraded- and reconstruction-mode performance.

1. Introduction

Recently, there are some new tendencies in storage field: more and more storage systems are constructed through in virtue of network technology; the size of storage systems becomes much larger; and the size of elementary storage device improves quickly, but the performance and reliability improves very slowly. We can see easily the effects of these tendencies, higher component failure rate, higher unrecoverable read error rate, and longer recover time, consequently the worse availability and reliability. So, current storage systems are in urgent need of multi-erasure-correcting codes. But unfortunately, although multi-erasure codes for storage applications have been studied for over two decades, none of known multi-erasure codes is as popular as mirroring and parity technologies. In this paper, we will present a double-erasure code MPDC (Mirroring Parity DeClustering) and a triple-erasure code MPPDC (Mirroring Parity-Parity DeClustering). They are based on mirroring and parity, so they are very simple and efficient. They lay over anything else of the kind in availability and reliability because of using parity declustering technology.

The rest of the paper is organized as follow. In the next section we review the related works about multi-erasure codes and parity declustering technology. In Section 3 we present the construction method of MPDC and MPPDC. The simulation results and analysis are given in Section 4. Finally, Section 5 summarizes this study.

2. Related work

The multi-erasure codes studies can be described as: given n data disks which store user data, design a coding scheme which encodes the content on n data disks onto m coding disks to provide the ability that can recover any $\leq t$ erasures. There are two trivial but effective schemes - replication and parity for n=1 and m=t=1 respectively. The former provides extremely high reliability and high performance, and the latter has optimal space efficiency and optimal access cost. But unfortunately, for t>1, and n>1, m>1, the known codes all have tradeoffs, and none of them is used as widely as the above two solutions [1].

Reed-Solomon code [2] is the only known MDS codes for arbitrary m (=t) and n up to now. But it's based on finite field arithmetic. Although some optimizations have been developed [3], the

* This paper is partly supported by NSF of China (90612001), Science and Technology Development Plan of Tianjin, (043185111-14), Nankai university R&D innovation fund and ISC.

encoding/decoding complexity is still a serious problem, especially for software implementation. RS code is used in RAID level 6 which is the most widely used multi-erasure code till now.

The binary linear codes presented by Gibson et al [4] are XOR-based, so they have perfect encoding/decoding complexity. Bad storage efficiency is their inherent defect. Recently, the studies of LDPC codes for storage applications have emerged [5, 6]. It's easy to see that LDPC codes are virtually irregular linear codes. The studies of LDPC codes focus on "average fault tolerance" instead of traditional "Hamming (threshold) fault tolerance". This distinctive idea leads to good tradeoff between reliability and storage efficiency.

Another category is so-called array codes, but we prefer thinking of them as data layouts of linear codes. EVENODD code [7] (and its generalization [8]) is the first MDS array code, perhaps also the most important one. Almost all the double- and triple-erasure array codes presented since then can be regarded as the varieties of EVENODD code and its generalization, such as X-Code [9], RDP code [10], STAR code [11], and so on. Poor flexibility is a big problem: all of these codes require prime size, and the code height is linear with the number of the disks.

B-CODE [12] and BG-HEDP/Latin Code [13,14] are based on perfect one-factorizations of complete graphs and complete bipartite graphs (Hanmiltonian Latin Squares) respectively. They have alleviative size limitation.

Most of the above codes seek for MDS or near-MDS property. But MDS array codes definitely cause large group size [13] which perhaps induces poor performance in distributed storage systems. WEAVER Codes [15] act in an opposite way - its best storage efficiency is 50%! Just the high redundancy brings about low stripe group size and good localization which are helpful to the performance of distributed storage applications. But the construction of WEAVER codes is time-consuming.

Besides lack of flexibility, the above works gave practice problems little thought. Parity declustering [16] is an effective technology for improving performance in fault mode. The key idea is to distribute the reconstruction load induced by the failed disks over all disks instead of the disks within the same parity group (the case in grouped RAID5/RAID6 arrays). Then, the per disk load increase ratio is $(G-1)/(N-1)$ (where G is the size of parity groups and N is the size of the array), while it is 100% in grouped RAID5/RAID6 arrays! Many parity declustering data layouts have been developed since mid 1990s', such as PRIME/RELPR [17], DATUM [18], PDDL [19], and so on. Most of their aims are single-erasure, and require prime size to achieve perfect reconstruction load balance. Combing Randomization and simulated annealing optimization [20] is suitable for any size, but can't guarantee perfect reconstruction load balance and require an in-memory mapping table.

Notably, although mirroring approach induces bad storage efficiency, two-way mirroring is used widely in many industries because of its advantage in performance and reliability. We combine mirroring, parity and parity declustering approaches to construct double-erasure code MPDC and triple-erasure code MPPDC which have high performance, high availability and high reliability. Today, the per MB cost of hard disk decreases quickly, the new codes are certainly affordable for the users who use two-way mirroring originally.

Li et al studied several combinations of mirroring and parity [21], but they did only theoretic performance analysis, and hadn't study any practical problems. Hsiao et al presented chained declustering to improve fault-mode performance in mirrored arrays [22], but their results are only partially parity declustering, and didn't consider combining mirroring and parity approaches. Recently, Miller et al developed a large-scale storage system based on object-oriented storage targets, mirroring and hash technology [23]. Compared with this work, our solution is simpler and suitable for small and moderate storage applications.

3. Design of MPDC and MPPDC

3.1. Combine mirroring and parity approaches

We add a parity disk to a RAID1+0 array (stripe of mirror pairs) then get a double-erasure code, we call it MP (Mirroring Parity). For describe convenience, we define some terms: the ***twin*** of a disk means its mirror disk, and vice versa; ***half failure*** denotes the case that a disk fails and its twin is ok; and ***pair failure*** denotes the case that a disk and its twin fail simultaneously. For MPDC and MPPDC, these terms are in terms of unit instead of disk.

MP code can tolerate any double erasures certainly. Any half failures can be recovered by coping data between twins easily, and pair failures and parity disk failure can be recovered through XORing all surviving pairs (and the parity disk). MP is a non-MDS code, 2 is its Hamming fault tolerance, so it is able to recover many triple (or more) erasures except those contain a pair and the parity disk or two pairs.

Based on a MP code, a triple-erasure code MPP (Mirroring Parity-Parity) can be constructed easily by

adding a mirror disk for the parity disk. It is obviously a triple-erasure code. All triple-erasures are composed of three half failures or a pair failure and a half failure. The former can be recovered by twin-coping. And the latter can be recovered by XORing all surviving pairs and the "half-healthy" pair. Any t-erasure containing two pair failures will break the array.

MP and MPP are actually particular cases of 2d-parity code [4] and its extension respectively. But degenerating from parity to mirroring in one dimension brings qualitative change. Load sharing between twins will improve performance greatly. Moreover, mirrored arrays needn't XOR operations when do updating, and then needn't to read parity units and data units. Thus MP/MPP are superior to other double- and triple-erasure codes in encoding/decoding/updating performance in terms of both computational complexity and the number of disk operations.

3.2. Design of MPDC

In view of the reconstruction load imbalance of MP and MPP, we think parity declustering. But the problem is more complex than those that the related works had faced. Most related works treated (grouped) RAID5 whose structure is one-dimension, the only failure type is single erasure, and the decoding method is unique. So the researchers only need to distribute independent parity stripes across all disks evenly. The reconstruction load balance will be achieved if each pair of disks participates in a fix number of stripes.

But the structures of MP and MPP are two-dimension, and there are several failure types in a MP/MPP array. We must optimize reconstruction load distribution not only for single-erasure mode but also for double-/triple-erasure mode. Maybe all possible recoverable erasures need to be considered in order to design "perfect" solutions. Moreover, various possible reconstruction methods for given single- or double-/triple-erasures add extra difficulty.

Fortunately, the twins can share load, this property simplifies the problem. We can get a good solution via one-factorizations of complete graphs. A factor of a graph $G=(V, E)$ is a spanning subgraph of G and a one-factor of G is a one-regular spanning subgraph of G. A factorization of G is a set of factors of G which are pairwise edge disjoint - no two have a common edge - whose union is G. A one-factorization (1F) of G is a factorization of G consisting of only one-factors. Formula 1 shows a construction method of 1Fs of even complete graph K_{p+1}.

$$E_i = \{(i, p)\} \cup \bigcup_{j=1}^{(p-1)/2} \{(i-j, i+j)\} \quad (1)$$

All arithmetic operations in Formula 1 are *mod p* arithmetic. E_i is the edge set of the *ith* factor for $0<=i<=p-1$. Apparently, Formula 1 constructs the famous perfect one-factorization (P1F) family - *GK* [24]. A 1F $F=\{F_0, F_1, ..., F_{k-1}\}$ is a P1F if for any distinct pair F_i, F_j of factors, $F_i \cup F_j$ induces a Hamiltonian cycle within G for $0 \le i, j < k$. GK_{p+1} is a perfect 1-factorization for all prime numbers p, but it is a 1-factorization for all integers p.

We can let the vertices of K_N denote the disks, and the edges denote the twin relationships. Then the problem is transformed into a graph factorization problem. Certainly, we still need to add parity units in, so using a 1-factorization of K_N directly to construct a code with N disks is not feasible. We can do a transformation on GK_{2N+2} to produce a MDPC code with $2N+1$ disks for all integers N. The algorithm is described as follow:

Algorithm 1 MPDC codes construction algorithm
Input: $GK_{2N+2} = \{F_0, F_1, \ldots, F_{2N+1}\}$
Output: A MPDC code with $2N+1$ disks
Method:

1. Rearrange GK_{2N+2}, let F_i contains edges $(2N-i, 2N+1)$ for $0 \le i \le 2N$.
2. Let vertex i of K_{2N+2} denote the *ith* disk of the array for $0 \le i \le 2N$, and vertex $2N+1$ be "parity vertex".
3. Construct the *ith* rows of the code via F_i for $0 \le i \le 2N$: for each edge (u, v) $(0 \le u, v \le 2N)$ of F_i, place a data unit on disk u, and place its twin on disk v; place the parity unit on disk $2N-i$.

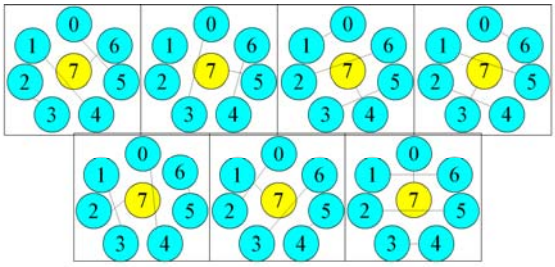
Figure 1 shows the construction of the MPDC code with 7 disks. Where *Di* denotes the *ith* data units, *Mi* denotes the twin unit of *Di*, and *Pi* denotes the parity unit of the *ith* row (stripe).

It is easy to see that because the parity unit is distributed evenly and each pair of disks contains exactly one unit twins, the fault-free mode load is distributed evenly in a MPDC array. The reconstruction load is not distributed absolutely evenly, but the distribution is near-optimal.

When a single-erasure occurs, the lost data units on the failed disk can be recovered by coping exactly one data unit from each surviving disk, and the lost parity unit can be recovered by reading all surviving unit twins at the same row. Obviously, the reconstruction of the lost parity unit involves only half of the surviving disks. But notably, the layout showed in Figure 1 is just a "period" which will be filled into disk arrays repeatedly. Thus the twins can share the reconstruction load caused by the lost parity units. Moreover, the

array still bear user load in fault mode, thus (approximate) load balance is achieved naturally.

Now let's consider double-erasure mode. We need recover a pair failure, two parity failure and 4N-2 single failures. It is easy to calculate that there are two cases: in order to recover a period on the two failed disks, 4 units must be read from a surviving disk, 3 units from other two disks, and 3.5 from the others; or 3 units from a surviving disk, and 3.5 from the others. However, the reconstruction load carried by each disk is close. If the user workload is taken into consideration, load balance can be achieved. Obviously, MPDC improves fault-mode performance greatly compared with RAID6 and MP. Its per-disk reconstruction load is fixed against the array size. And the larger the disk array, the more superiority is MPDC to other codes.



ROW(Stripe)	DISK0	DISK1	DISK2	DISK3	DISK4	DISK5	DISK6
0	M0	M1	M2	D2	D1	D0	P0
1	M4	M5	D5	D4	D3	P1	M3
2	M8	D8	D7	D6	P2	M6	M7
3	D11	D10	D9	P3	M9	M10	M11
4	D13	D12	P4	M12	M13	M14	D14
5	D15	P5	M15	M16	M17	D17	D16
6	P6	M18	M19	M20	D20	D19	D18

Figure 1. **Constructing MPDC code with 7 disks**

3.3. Design of MPPDC

We hope to construct triple-erasure parity declustering codes MPPDC based on MPP codes through a method like algorithm 1. But it is difficult to achieve the four objectives: distributing parity units evenly, and distributing single-/double-/triple-erasure reconstruction load evenly. So we relax the metrics of perfect parity declustering data layouts, and let twins adjust the load distribution adaptively. The algorithm 2 describes the construction steps of a MPPDC code with 2N+2 disks via GK_{2N+2}.

Algorithm 2 MPPDC codes construction algorithm
Input: $GK_{2N+2} = \{F_0, F_1, \ldots, F_{2N+1}\}$
Output: A MPDC code with 2N+2 disks
Method:

1. Rearrange GK_{2N+2}, let F_i contains edges (2N-i, 2N+1) for $0 \le i \le 2N$.
2. Let vertex i of K_{2N+2} denote the *ith* disk of the array for $0 \le i \le 2N+1$.
3. Construct the *ith* rows of the code via F_i for $0 \le i \le 2N$: place the parity twins on disk (2N-i-1) and disk (2N-i+1), and for each edge (u, v) ($0 \le u$, $v \le 2N$) of F_i except (2N-i-1, 2N-i+1), place a data unit on disk u, and place its twin on disk v. (all arithmetic operations are *mod* (2N+1) arithmetic)

Our basic way is that constructing the codes based on GK_{2N+2}, selecting one edge from each 1-factor (row) of GK_{2N+2}, and letting the adjacent vertices of the edges be the disks which hold the parity units. In order to get perfect parity load distribution, we must design an edge selection scheme which satisfies that the selected N+1 edges from any N+1 consecutive factors just cover all vertices - each disk contains exactly one parity units. If for any GK_{2N+2}, there exists a 1-factor F of K_{2N+2} which satisfies that the intersection of F and each factor of GK_{2N+2} is exactly one edge, the problem will fade out. In graph theory, a *rainbow* 1-factor is a 1-factor with the property that no two edges of it are in the same 1-factor of a 1-factorization. Graph theorists have proven that given any 1-factorization, there exists a rainbow [24]. But unfortunately, the property of rainbow is slightly different from our requirement. In fact, since the size of GK_{2N+2} is 2N+1 and the size of a 1-factor of K_{2N+2} is N+1, it is almost impossible to design a perfect solution. Remarkably, we have only discussed the first objective. The scheme which can solve all the four objectives perfectly seems an unachievable goal.

ROW (Stripe)	DISK0	DISK1	DISK2	DISK3	DISK4	DISK5	DISK6	DISK7
0	P0	M1	M2	D2	D1	PM0	D0	M0
1	M4	M5	D5	D4	PM1	D3	P1	M3
2	M8	D8	D7	PM2	D6	P2	M7	M6
3	D11	D10	PM3	D9	P3	M10	M11	M9
4	D13	PM4	D12	P4	M13	M14	D14	M12
5	PM5	D15	P5	M16	M17	D17	D16	M15
6	D18	P6	M19	M20	D20	D19	PM6	M18

Figure 2. **MPPDC code with 8 disks**

So we select a near-optimal but relative simple method as algorithm 2 described. The main idea is that let the two adjacent vertices of the second edge of F_i store the parity twins at the *ith* row. Figure 2 shows a MPPDC code with 8 disks, where *PMi* denotes the twin of the parity units at the *ith* row. Obviously, the parity units are distributed evenly neither globally nor locally. Especially disk 2N+1 contains no parity units. But it can share the load on any other disk, thus the fault-free load balance can be achieved.

When any single-erasure occurs, the reconstruction of one period of the failed disk reads exactly one unit from each surviving disk. When any double-erasure

occurs, 2.5 units must be read from each surviving disk during recover one period.

As for triple-erasures, the lost units compose three pair failures and 6*N*-3 half failures. We can prove that 4 units must be read from three surviving disks and 4.5 units from the other surviving disks in order to recover a triple-erasure.

4. Performance evaluation

4.1. Performance metrics analysis

Gibson et al presented 5 metrics for multi-erasure coding schemes: reliability, check disk overhead, update penalty, group size and extensibility. We will compare MPDC/MPPDC with other coding schemes at these aspects. In this subsection, *N* denotes array size.

MP/MPP are particular cases of 2d-parity codes, therefore their fault tolerance are not confined to 2/3, they can recover all 3-/4-erasures except bad 3-/4-erasures [4]. Though parity declustering is applied, MPDC/MPPDC retain the high fault tolerance. Of course, the definition of "bad" is different. In MP arrays, bad 3-erasures are 3-erasures composed of a data, its twin and the parity. And in MPP arrays, bad 4-erasures are 4-erasures composed of two twins.

In MPDC arrays, we use a triple (*i*, *j*, *k*) denote a 3-erasure - *i*, *j* and *k* are the serial numbers of the failed disks. According to algorithm 1, we call (*i*, *j*, *k*) bad if it satisfies one of the following equations: (1) $i + j \equiv 2k \ mod\ N$; (2) $i + k \equiv 2j \ mod\ N$; (3) $k + j \equiv 2i \ mod\ N$.

It is easy to see that we can get all bad 3-erasures by applying equation 1 to each pair of serial numbers. Thus, there are $N(N-1)/2$ bad (unrecoverable) 3-erasures in total $N(N-1)(N-2)/6$ 3-erasures. But when $N \equiv 0 \ mod\ 3$, there are $N/3$ triples satisfy the three equations simultaneously, namely they are calculated three times. So the number of bad 3-erasures is $N(N-1)/2 - 2N/3$. Then the unrecoverable 3-erasure rate of MPDC is $O(1/N)$, while the rate of MP (the best in 2d-parity) is $O(1/N^2)$. Apparently, MPDC is inferior to MP. But the recoverable rates of them are asymptotically identical. Similarly, we can prove that the unrecoverable 4-erasure rate of MPPDC is $O(1/N)$, while the rate of MPP is $O(1/N^2)$. As for MTTDL (Mean Time To Data Loss), we have developed a simulator which is more precise than Markov model method. The simulation results show that the MTTDL of MPDC/MPPDC is better than RAID6 because of their better average fault tolerance, and is better than MP/MPP because of shorter reconstruction time (see subsection 4.2). The results are not showed in this paper for lack of space.

Storage efficiency is the main drawback of MPDC/MPPDC. But their efficiency is close to two-way mirroring unless *N* is very small. Moreover, as described above, the capacity of hard disk increases dramatically and price decreases dramatically today, so the not good check disk overhead is not a fatal flaw.

MPDC/MPPDC are far superior to other codes in encoding/decoding/updating complexity. We use the number of per data unit XOR operations to evaluate encoding performance fairly. The values of this criterion in MPDC/MPPDC arrays are $1-2/(N-1)$ and $1-2/(N-2)$ respectively, while RDP is $2-2/(N-2)$, EVENODD is $2-1/(N-3)$, 2d-parity is about $2-2/\sqrt{N}$, and STAR Code is $3-1/(N-4)-1/(N-3)(N-4)$. That is to say that in order to do encoding, MPDC only need to do half of what the other 2-erasure codes do and MPPDC only need to do one-third of what the other 3-erasure codes do. Decoding/updating cases are similar. The reason is simple - mirroring needn't to do XOR operations.

Mirroring also benefits the update penalty in terms of the number of disk operations. The update penalties of MPDC and MPPDC are 5 and 6 respectively, while the update penalties of typical 2- and 3-erasure codes are 6 and 8 respectively. The advantage of MPDC and MPPDC is evident.

MPDC and MPPDC also have good parity group size. Though the size of the biggest parity group is about *N/2*, the twins have extreme small group size - 2, therefore the average group size is about 3. MPDC and MPPDC are comparable to WEAVER CODE at this aspect. In fact, Gibson et al hope to evaluate fault mode performance through this criterion - small group size means little impact on arrays during reconstruction. Obviously, per disk load increase rate during reconstruction is a better criterion. As described in subsection 3.2 and 3.3, MPDC and MPPDC are far superior to the other codes at this aspect.

4.2. Simulation

In order to compare the performance of MPDC and MPPDC with other 2- and 3-erasure codes precisely, we did simulation. The simulation platform is modified DiskSim [25]. The disk model is DEC DZ26. Unless otherwise noted, the array size is 14 (2*7 for grouped RAID6, and 13 for MP and MPDC), the stripe unit size is 8KB, the workload is synthetic, the request size and alignment are all 8KB, the request distribution is uniform, the request type is 80% read and 20% write,

the request inter-arrival time distribution is exponential, and the generator works in non time-critical, non time-limited mode. All the simulations are repeated 5 times and the results showed in this paper are all average. We test six kinds of codes: RAID6, grouped RAID6, MP, MPP, MPDC and MPPDC. The simulations evaluate not CPU performance but disk performance, thus RAID6 can represent (near) MDS horizontal codes such as EVENODD, RDP and so on because their disk operation models are identical. Similarly, MP and MPP can represent binary linear codes.

Figure 3 plots the average user response time vs. the achieved user I/O operations per second when each array is fault-free. MP and MPP have the worst load capacity because of extremely uneven parity distribution. Other codes have almost the same load capacity. Under moderate and heavy workloads, the advantage of mirroring emerges. The response time of MPDC and MPPDC is only about 60% ~ 67% of RAID 6 and grouped RAID6.

Figure 4 shows the single-erasure mode performance. It is easy to see the advantage of parity declustering. MPDC and MPPDC have almost no load capacity degradation, while RAID6 and grouped RAID6 have about 40% and 35% degradation respectively. MPDC and MPPDC are also superior in response time degradation. And the response time gap between the two kinds of codes becomes wider compared with fault-free mode. Figure 5 and Figure 6 show the double-erasure mode performance and triple-erasure mode performance respectively. "FailP" means that there is a pair failure in a MP/MPP array. It is easy to see that the general trends are similar to single-erasure mode. And the performance gap between MPDC/MPPDC and RAID6/gRAID6 becomes wider as the number of failed disks climbs. Namely, MPDC/MPPDC have better "erasure scalability".

We have also tested the performance under reconstruction-mode. We cannot list all the results in this paper for lack of space. Only the double-erasure reconstruction time is showed in Figure 7. It is easy to see that MPDC and MPPDC are also superior to other codes in reconstruction time. This advantage will bring better MTTDL. Unlisted results are similar to Figure 3 - Figure 7 in the view of performance comparison.

In order to compare the scalability of the codes, we did the above simulations for some bigger arrays which have 28, 42 and 56 disks respectively. Certainly, the sizes of MPDC arrays are 27, 41 and 55. The parity group sizes of grouped RAID6 arrays are fixed to 7 in favor of their performance. We only show the response time of grouped RAID6, MPDC and MPPDC under fault-free mode, single-erasure mode and double-erasure mode in Figure 8 - Figure 10 for lack of space. The suffixes "f1" and "f2" denote single- and double-erasure respectively. Obviously, as the size increases, the MPDC/MPPDC arrays retain little performance degradation when failures occur. And the performance gap between MPDC/MPPDC and grouped RAID6 becomes wider as the array size increases. Certainly, the performance gap between MPDC and MPPDC also becomes wider slowly because the former has better update penalty and better load balance. In a word, MPDC/MPPDC win again.

5. Conclusions and future works

In this paper, we developed a double-erasure code MPDC and a triple-erasure code MPPDC based on one-factorizations of complete graphs using mirroring and parity approaches and parity declustering technology. The theoretical analysis and simulation results show that their fault-free performance is far higher than MP and MPP, and is superior to RAID6 and grouped RAID6 at heavy workloads. Their degraded- and reconstruction-mode performance is close to their fault-free mode, and is far superior to other codes. Their storage efficiency is close to RAID1, but they provide double- and triple-erasure-correcting ability with excellent availability and reliability. Their coding/decoding/updating complexity is also better than that of other 2- and 3-erasure codes. Moreover, they have no limit in the number of disks.

The future works will focus on the implementation of MPDC and MPPDC, and performance evaluation via the workloads close to real world workloads (such as TPC-C, TPC-W, and so on). We also plan to construct MPDC and MPPDC via other one-factorizations of K_N instead of GK_N. There are 396 nonisomorphic one-factorizations of K_{10}, and the number is 526,915,620 for K_{12} [24]! This good variety will be helpful to performance optimization. In addition, we may optimize fault-mode performance via distributed spare technology.

References

[1] J. S. Plank, "Erasure Codes for Storage Applications", Tutorial of the 4th Usenix Conference on File and Storage Technologies, San Francisco, CA, Dec, 2005.

[2] J. S. Plank, "A Tutorial on Reed-Solomon Coding for Fault-Tolerance in RAID-like Systems", Software - Practice & Experience, Vol. 27, No.9, Sep, 1997, pp.995-1012.

[3] J. S. Plank and Lihao Xu, "Optimizing Cauchy Reed-Solomon Codes for Fault-Tolerant Network Storage Applications", In Proceedings of the 5th IEEE

International Symposium on Network Computing and Applications, Cambridge, MA, Jul, 2006, pp.173-180.

[4] Lisa Hellerstein, Garth A. Gibson, Richard M. Karp, Randy H. Katz and David A. Patterson, "Coding techniques for handling failures in large disk arrays", Algorithmica, Vol. 12, No. 2/3, Aug,1994, pp.182-208.

[5] M. G. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman and V. Stemann, "Practical Loss-Resilient Codes", In Proceedings of the 29th Annual ACM Symposium on Theory of Computing, El Paso, Texas, May, 1997, pp.150-159.

[6] J. S. Plank and M. G. Thomason. "A practical analysis of low-density parity-check erasure codes for wide-area storage applications", In Proceedings of the International Conference on Dependable Systems and Networks, Florence, Italy, Jun, 2004, pp.115–124,.

[7] M. Blaum, J. Brady, J. Bruck, J. Menon, "EVENODD: an efficient scheme for tolerating double disk failures in RAID architectures", IEEE Trans. on Computers, Vol. 44, No. 2, pp. Feb, 1995, 192-202.

[8] M. Blaum, J. Bruck, and A. Vardy, "MDS array codes with independent parity symbols", IEEE Trans. on Information Theory, Vol. 42, No. 2, Mar, 1996, pp. 529-542.

[9] L. Xu and J. Bruck, "X-Code: MDS Array Codes with Optimal Encoding", IEEE Trans. on Information Theory, Vol. 45, No. 1, Jan, 1999, pp.272-276.

[10] P. Corbett, B. English, A. Goel, T. Grcanac, S. Kleiman, J. Leong and S. Sankar, "Row-Diagonal Parity for Double Disk Failure Correction", In Proceedings of the 3th USENIX Conference on File and Storage Technologies, San Francisco, CA, USA, Mar, 2004, pp.1-14.

[11] Cheng Huang, Lihao Xu, "STAR: An Efficient Coding Scheme for Correcting Triple Storage Node Failures", In Proceedings of the 4th USENIX Conference on File and Storage Technologies, San Francisco, Dec, 2005, pp.197-210.

[12] L. Xu, V. Bohossian, J. Bruck, and D.G. Wagner, "Low-Density MDS Codes and Factors of Complete Graphs", IEEE Trans. on Information Theory, Vol. 45, No. 6, Sep, 1999, pp.1817-1826.

[13] Wang Gang, Dong Sha-sha, Liu Xiao-guang, Lin Sheng, Liu Jing, "Construct double-erasure-correcting Data Layout Using P1F", ACTA ELECTRONICA SINICA, Vol. 34, No. 12A, 2006, pp.2447-2450.

[14] Gang Wang, Sheng Lin, Xiaoguang Liu, Guangjun Xie, Jing Liu, "Combinatorial Constructions of Multi-Erasure-Correcting Codes with Independent Parity Symbols for Storage Systems", (to appear) IEEE PRDC 2007, Melbourne, Victoria, Austrilia, Dec, 2007.

[15] J. L. Hafner, "WEAVER Codes: Highly Fault Tolerant Erasure Codes for Storage Systems", In Proceedings of the 4th Usenix Conference on File and Storage Technologies, San Francisco, Dec, 2005, pp.211-224.

[16] M. Holland, G. A. Gibson, D. P. Sieworuk, "Architectures and Algorithms for On-Line Failure Recovery in Redundant Disk Arrays", Journal of Parallel and Distributed Databases, Vol. 2, No. 3, Jul, 1994, pp.295-335.

[17] G. A. Alvarez, W. A. Burkhard, L. J. Stockmeyer, F. Cristian, "Declustered Disk Array Architectures with Optimal and Near-Optimal Parallelism", ACM SIGARCH Computer Architecture, Vol. 26, No. 3, Jun, 1998, pp.109-120.

[18] G. A. Alvarez, W. A. Burkhard, F. Cristian, "Tolerating Multiple Failures in RAID Architectures with Optimal Storage and Uniform Declustering", In Proceedings of the 24th Annual ACM/IEEE International Symposium on Computer Architecture, Denver, Colorado, United States, Jun, 1997, pp.62-72.

[19] T. J. E. Schwarz, J. Steinberg, W. A. Burkhard, "Permutation Development Data Layout (PDDL) Disk Array Declustering", In Proceedings of the 5th International Symposium on High-Performance Computer Architecture, Orlando, FL, USA, Jan, 1999, pp.214-217.

[20] Eric J. Schwabe, Ian M. Sutherland, Bruce K. Holmer, "Evaluating Approximately Balanced Parity-Declustered Data Layouts for Disk Arrays", Parallel Computing, Vol. 23, No. 4-5, Jun, 1997, pp.501-523.

[21] C. S. Li, M. S. Chen, P. S. Yu and H. I. Hsiao, "Combining Replication and Parity Approaches for Fault-Tolerant Disk Arrays", In Proceedings of the 6th IEEE Symp. on Parallel and Distributed Processing, Arlington, USA, Oct. 1994, pp.360-367.

[22] H. I. Hsiao and D. J. DeWitt, "A Performance Study of Three High-Availability Data Replication Strategies", In Proceedings of the International Conference on Parallel and Distributed Information Systems, Miami, Florida, USA, 1991, pp.18-29.

[23] Sage A. Weil, Scott A. Brandt, Ethan L. Miller, Darrell D. E. Long, and Carlos Maltzahn, "Ceph: A Scalable, High-Performance Distributed File System", In Proceedings of the 7th Conference on Operating Systems Design and Implementation, Seattle, WA, Nov, 2006, pp.307-320.

[24] Charles. J. Colbourn, Jeffrey H. Dinitz, et al, "Handbook of Combinatorial Designs (Second Edition)", CRC Press, 2007.

[25] J. S. Bucy, G. R. Ganger, "The DiskSim Simulation Environment Version 3.0 Reference Manual", Technical Report CMU-CS-03-102, Carnegie Mellon University, 2003.

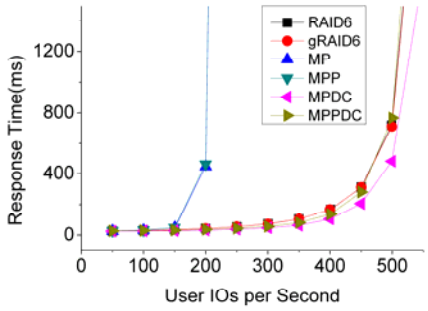


Figure 3. **Fault-free mode performance**

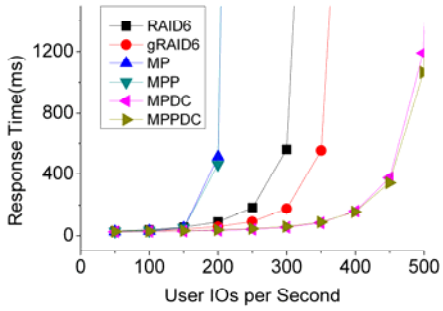


Figure 4. **Single-erasure mode performance**

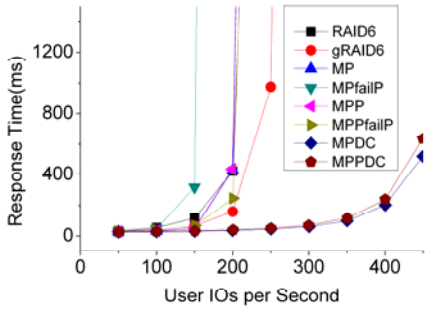


Figure 5. **Double-erasure mode performance**

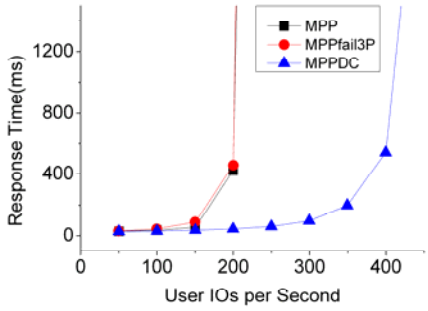


Figure 6. **Triple-erasure mode performance**

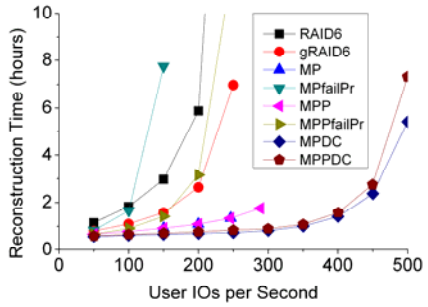


Figure 7. **Double-erasure reconstruction time**

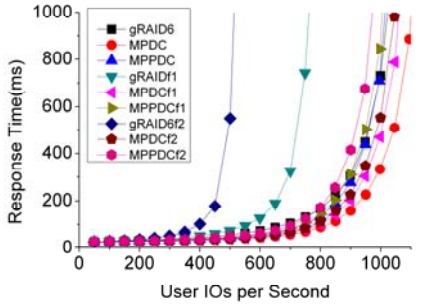


Figure 8. **Performance of arrays with 28 disks**

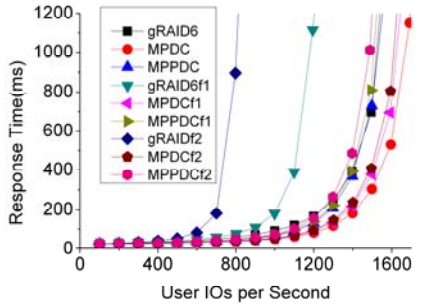


Figure 9. **Performance of arrays with 42 disks**

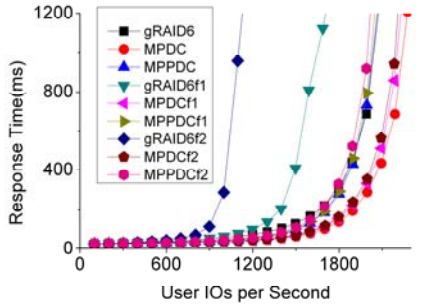


Figure 10. **Performance of arrays with 56 disks**