计算机研究与发展
Journal of Computer Research and Development
ISSN 1000-1239/CN 11-1777/TP
49(1): 1-11, 2012

存储系统中的纠删码研究综述

罗象宏　　舒继武
(清华大学计算机科学与技术系　北京　100084)
(luo-xh09@mails.tsinghua.edu.cn)

Summary of Research for Erasure Code in Storage System

Luo Xianghong and Shu Jiwu
(*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084)

Abstract　With the development of massive storage system and its application in complex environments, there is a big challenge in the reliability of storage system. Erasure code is drawing more and more attention since it is the main technology for fault tolerance in storage systems. In this paper, we firstly introduce the current-status of some typical and popular erasure codes, then make careful comparison and analysis on current erasure codes with the important metrics that are used to evaluate them. Afterwards, we point out some shortages and improvement suggestions on fault tolerance, requirements for the number of disks, storage efficiency, encoding efficiency, updating efficiency and decoding efficiency for different erasure codes. What' more, we discuss the different requirements on erasure code in disk array systems, P2P systems, distributed storage systems and archival storage systems. Finally, we indicate the unresolved problems in erasure code and their future trends. From the analysis, we found a lot of drawbacks on fault tolerance, storage efficiency and computation efficiency (including encoding efficiency, updating efficiency and decoding efficiency) for different erasure codes. It is an issue worthy of further study in a long period to make a balance on these factors and create new erasure code with higher fault tolerance, greater storage efficiency, and faster computation efficiency.

Key words　storage systems; erasure code; fault tolerance; storage efficiency; computation efficiency

摘　要　随着海量存储系统的发展和在复杂环境中的应用,存储系统的可靠性受到了严重的挑战.纠删码作为存储系统容错的主要方法越来越受到重视.首先介绍了当前典型和常见的纠删码技术的发展现状,从评价纠删码性能的各项重要指标的角度详细地对比和分析了现有的纠删码技术,给出了不同纠删码在容错能力与磁盘要求、空间利用率、编码效率、更新效率、重构效率等方面的不足和可能的改进见解,并讨论了磁盘阵列系统、P2P 存储系统、分布式存储系统、归档存储系统等不同存储系统对于纠删码各类性能的差别要求,并进一步指明了当前存储系统纠删码研究中尚未解决的一些难题和未来纠删码可能的发展方向.通过分析得出,目前不同纠删码在容错能力、计算效率、存储利用率等方面都存在不同程度的缺陷,如何平衡这些影响纠删码性能的因素,设计出更高容错能力、更高计算效率及更高存储利用率的纠删码,仍是未来很长一段时间内值得不断深入研究的问题.

关键词　存储系统;纠删码;容错率;存储利用率;计算效率

中图法分类号　TP302.8

收稿日期:2011-01-13;**修回日期**:2011-10-02

基金项目:国家自然科学基金项目(60925006,60873066);国家"八六三"高技术研究发展计划基金重大项目(2009AA01A403);Intel 国际合作项目(Intel-CRC-2010-06)

通信作者:舒继武(shujw@tsinghua.edu.cn)

随着计算机技术和网络技术的发展,数据正以爆炸式的速度增长,海量信息对存储系统提出了巨大的挑战[1]. 特别是随着存储系统中存储介质数目的增加,以及存储介质的多样化和复杂化,存储系统的存储介质错误[2-3]或者存储介质中的潜在扇区错误(latent sector errors[4])出现的概率也越来越高. 为了提高数据的可靠性和可用性,需要研究高效并易于实现的存储容错技术,目前,这些技术都是通过增加冗余的信息来查找并修复存储介质故障的.

通常的冗余备份机制包括 2 类:一类是完全的数据备份机制,即镜像方法(mirrored method);而另一类就是纠删码(erasure code)的方法.

镜像方法又称为多副本技术,就是把数据复制多个副本分别存储起来,以实现冗余备份. 这种方法不涉及专门的编码和重构算法,容错性能较好,但存储利用率极低,当存放 N 个副本时,磁盘利用率仅有 $\frac{1}{N}$,尤其是当系统规模很大时,镜像技术带来的额外存储空间的开销很大,导致存储成本非常高.

纠删码起源于通信传输领域,起初主要是用于解决数据传输中的检错和纠错问题. 后来纠删码逐渐应用到存储系统中的数据检错和纠错问题中,以提高存储系统的可靠性,并根据存储系统应用的特点逐步得到改进和推广[5]. 另外,在通信传输研究中,纠删码的编码算法和重构算法得到了更多的关注,而对于存储系统中的纠删码,其更新效率更受关注,并成为评价存储系统纠删码最重要的标志(这将在下文中阐述),在下文中,对纠删码的讨论将约定在存储系统应用中. 在存储系统中,纠删码技术主要是通过利用纠删码算法将原始的数据进行编码得到冗余,并将数据和冗余一并存储起来,以达到容错的目的. 其基本思想是将 k 块原始的数据元素通过一定的编码计算,得到 m 块冗余元素. 对于这 $k+m$ 块的元素,当其中任意的 m 块元素出错(包括数据和冗余出错)时,均可以通过对应的重构算法恢复出原来的 k 块数据. 基于纠删码的方法与传统的镜像副本技术相比,具有冗余度低、磁盘利用率高等优点.

随着存储系统规模的不断扩大,镜像方法已难以适应海量存储系统对于冗余备份机制在磁盘利用率和容错能力方面的要求,因此研究者更多地关注在存储系统纠删码的研究上. 本文将通过介绍一些典型和常用的纠删码来展示当前纠删码研究的现状,并从评价纠删码的各项重要指标的角度比较和分析了现有的纠删码,指出当前存储系统纠删码研究中尚未解决的难题以及未来可能的研究方向.

1 存储系统中纠删码的相关概念

目前对存储系统中的纠删码及相关概念还没有一个统一的定义. 为便于理解,基于文献[6-7],我们给出如下一些常用的基本概念的说明和定义:

1) 数据(data or information)——原始的用于存储真实用户需要的信息的一块数据串.

2) 冗余(parity or redundant)——利用纠删码算法,通过计算数据而得到的存储冗余信息的一块数据串,这些冗余的存在是为了保证纠删码的容错能力.

3) 元素(element or symbol)——在编码理论中,元素是信息存储的最小单位,它存储一定量的连续数据信息或者冗余信息. 在纠删码的计算过程中,将一个元素视为计算中的一个基本计算单位.

4) 条带(stripe)——独立地与同一纠删码算法相关的所有信息的集合. 条带是独立地构成一个纠删码算法的信息集合,等价于一个算法的"实例". 一个存储系统可以看成是很多条带的集合,但是条带之间相互独立. 因此逻辑上条带中的磁盘与真实的物理磁盘的一一对应关系并不一定是恒定不变的,不同的条带可以有不同的对应方法,这样就可以通过条带的轮转来实现磁盘的负载均衡.

5) 条块(strip)——处于同一存储设备上的属于同一条带的数据的集合,条块的大小由条块所包含的元素的个数决定,记为 s. 一个条块可以只包含数据或者冗余,也可以同时包含数据和冗余.

元素、条块与条带之间的相互关系如图 1 所示[8]:

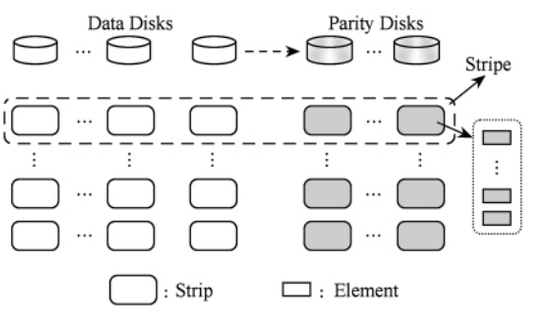


Fig. 1 Relationship between element, strip and stripe[8].

图 1 元素、条块与条带之间的相互关系[8]

6) 横式编码(horizontal codes)——冗余独立于数据条块,并单独存储在冗余条块中的编码方式. 即在横式编码中,属于同一个条带的冗余元素与数据元素存储在不同的条块中,每个条块只存储数据或者只存储冗余,没有混合的条块. 例如 EVENODD

编码[9]、RDP 编码[10]等都是横式编码.在横式编码中,通常将存储数据的磁盘数目记为 n,存储冗余的磁盘数目记为 m.

7)纵式编码(vertical codes)——冗余存储在数据磁盘中的编码方式,即在纵式编码中,某些条块中既存储了数据元素,又存储了冗余元素,例如 X-code 编码[11]等.

8)容错率(fault tolerance)——可以容忍的最多任意条块出错数目,记为 t.即当任意不多于 t 块条块出错时,均一定可以通过相应的重构方法恢复出所有出错的条块.而存在某 $t+1$ 块条块,如果这 $t+1$ 块条块出错,将没有办法通过重构方法恢复出所有出错的条块.

9)存储利用率(storage efficiency)——存储系统的存储效率即是指存储的数据量与所有的信息量(所有的信息量是指数据量与冗余量之和)之比,记为 E.

10)MDS 编码(maximum distance separable codes)——满足了 Singleton 边界条件[12]的编码方式,即达到了理论上最优的存储利用率的编码.

11)编码(encoding)——在纠删码算法下由数据计算出冗余的过程.

12)更新(updating)——当数据元素需要进行修改时,在纠删码算法下,重新计算与该数据相关联的冗余的过程.在存储系统中,数据更新非常频繁,因此更新效率是评价纠删码算法的最重要标志.

13)重构(decoding)——当部分磁盘或者磁盘中的某些数据块出错时,利用纠删码对应的重构算法,从剩余的未出错信息中恢复出所有数据的过程.

2 存储系统中典型的纠删码

本节主要介绍当前典型和常见的各类纠删码,并对这些纠删码进行分析和比较.

2.1 RS(Reed-Solomon)编码

RS 编码是唯一可以满足任意的数据磁盘数目(n)和冗余磁盘数目(m)的 MDS(maximum distance separable)的编码方法.RS 编码起源于 1960 年[13],经过长期的发展,已经具有较为完善的理论基础.RS 编码是在 Galois 域 $GF(2^w)$ 上进行所对应的域元素的多项式运算(包括加法运算和乘法运算)的编码方式,它属于横式编码.RS 编码通常分为 2 类:一类是范德蒙 RS 编码(Vandermonde RS codes[13]);另一类是柯西 RS 编码(Cauchy RS codes[14]).范德蒙 RS 编码使用的生成矩阵是范德蒙矩阵,而柯西 RS 编码所用的生成矩阵则为柯西矩阵.

2.1.1 范德蒙 RS 编码

图 2 所示[15]为 RS 编码的编码和重构原理.RS 编码实际上就是利用生成矩阵与数据列向量的乘积来计算得到信息列向量的.其重构算法实际上也是利用未出错信息所对应的残余生成矩阵的逆矩阵与未出错的信息列向量相乘来恢复原始数据的,具体算法可参见文献[13].但是在 RS 编码重构原理的推导过程中,有一个条件至关重要,那就是未出错信息所对应的残余生成矩阵在 $GF(2^w)$ 域上必须要满足可逆的条件.更进一步来说,为了满足 RS 编码在任何情况下均是可重构的,对于任意的 m 个元素出错时,均要能够通过重构算法恢复出数据,这就要求对于任意的 n 个元素(n 的含义是指去掉 m 个出错元素以后剩下的 n 个未出错的元素),其对应的残余生成矩阵均要在 $GF(2^w)$ 域上满足可逆,这就对 RS 编码的生成矩阵提出了很高的要求.在生成矩阵中,要求任意 n 个行向量均必须在 $GF(2^w)$ 域上满足线性无关(行向量线性无关是矩阵可逆的充要条件).

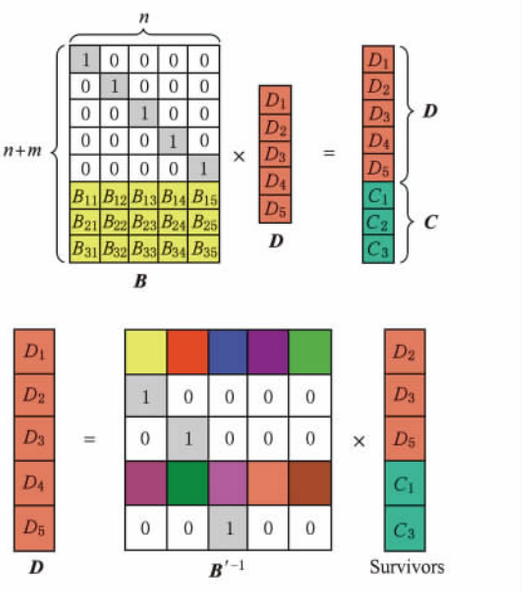


Fig. 2 Encoding and decoding principle for Reed-Solomon code[15].

图 2 RS 编码的编码和重构原理[15]

在数学上应用极广的范德蒙矩阵(Vandermonde matrix)正好就可以满足 RS 码的重构原理对于生成矩阵提出的要求[13].在 $GF(2^w)$ 域上,将范德蒙矩阵进行初等变换,将其前 n 行变成一个单位矩阵,就可以得到满足 RS 编码要求的生成矩阵.基于这个生成矩阵的 RS 编码就是范德蒙 RS 编码.

在 $GF(2^w)$ 域上，加法的定义实际上就是异或(XOR)运算，而乘法则要复杂得多，通常需要借助离散对数运算和查表才能进行. 因此，标准 RS 编码的计算开销太大，无法适应存储系统对于计算效率的要求.

2.1.2 柯西 RS 编码

在 $GF(2^w)$ 域上，复杂的乘法运算和矩阵求逆运算成为制约标准 RS 编码进一步发展的障碍. 除了范德蒙矩阵以外，是否还存在别的矩阵同样可以满足 RS 编码的重构算法对于生成矩阵提出的要求？是否存在一种生成矩阵可以将 $GF(2^w)$ 域上的乘法化繁为简？柯西 RS 编码的出现给予了这些问题肯定的回答.

柯西 RS 编码使用柯西矩阵[14]来代替范德蒙行列式，使得生成矩阵变得更为简单. 同时，文献[14]中提出的位运算转换方法可以对于柯西 RS 编码中的乘法运算进行改进. 经过简单的改造，柯西 RS 编码可以将 $GF(2^w)$ 域上的乘法运算转化成二进制的乘法运算. 因此，整个 RS 编码的运算就可以转化成只包含异或(XOR)的简单运算. 如图 3 所示就是经过改造过后的 RS 码的编码过程示意图[16]：

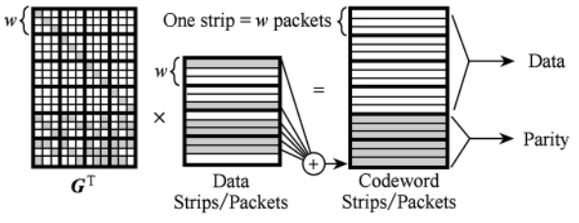


Fig. 3 Modified encoding principle for Cauchy Reed-Solomon code[16].

图 3 改造后的柯西 RS 码的编码原理[16]

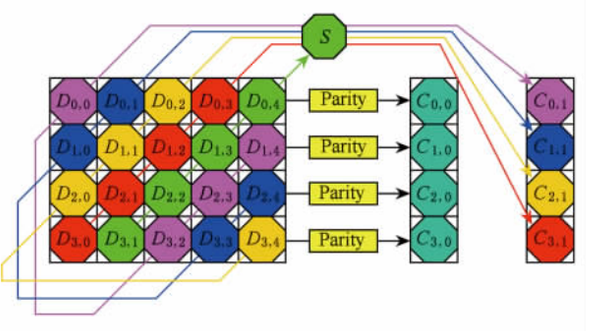
仔细观察改造后的 RS 码的编码过程，很容易发现，编码的效率直接决定于生成矩阵中“1”的数目，同时，这个“1”的数目还影响着 RS 码的更新效率. 那么，如何尽量减少生成矩阵中“1”的数目，从而得到最优的编码和更新效率呢？文献[17]对此专门作了深入的研究，由于篇幅有限，此不赘述.

2.2 横式阵列码

与 RS 编码相比，阵列码(array code[18])完全基于异或(XOR)运算，这是纠删码研究的重点. 阵列码的含义就是将原始的数据和冗余都存储在一个 2 维(或者多维)的阵列中. 与传统的 RS 编码相比，在阵列码中，仅使用异或(XOR)操作，实现容易，编码、更新、以及重构的过程都相对比较简单，因此应用也最广. 下面我们首先介绍横式阵列码，然后在 2.3 节中介绍纵式阵列码.

横式阵列码(horizontal parity array codes)是指冗余独立于数据条块，单独存储在冗余条块中的阵列编码方式. 它的结构特点是让冗余单独存放在独立的磁盘中(这些磁盘被称为冗余磁盘)，而让剩下的磁盘专门用于存储数据，这样的排布方式使得整个磁盘阵列具有非常好的可扩展性. 横式阵列码的容错率一般都不是很大，例如，容错率为 2 的编码包括 EVENODD[9]，RDP[10]，EEO[19]，Liberation Code[20]等. 容错率大于 2 的编码目前还研究得较少，但是也涌现了一些成果，例如 STAR[21]编码等.

图 4 给出了 2 类典型的横式编码的示意图. EVENODD 编码[9]被公认是阵列码的始祖，RDP[10]也是一类应用很广的横式编码. 在这些编码体系中都对磁盘数目提出了严格的要求(要求磁盘数目必须是素数或素数−1)，同时还要求在单个条块中的数据元素的个数必须与磁盘数目相匹配(具体要求可参看文献[9-10]，此不赘述). 从空间利用率上看，这 2 类编码在理论上满足了最优的存储效率，都是 MDS(maximum distance separable)的编码.



(a) EVENODD code[15]

Data Disk 0	Data Disk 1	Data Disk 2	Data Disk 3	Row Parity	Diag. Parity
0	1	2	3	4	0
1	2	3	4	0	1
2	3	4	0	1	2
3	4	0	1	2	3

(b) RDP code[10]

Fig. 4 Two kinds of typical horizontal codes.

图 4 两类典型的横式编码

由文献[6]可知，任意的编码结构在数学上实际都可以使用一个生成矩阵来进行描述. 那么，从这样的角度来看，决定一种纠删码的编码效率和更新效率实际上就是生成矩阵中“1”的数目，如何能够减少生成矩阵中“1”的数目，从而提高纠删码的效率，成为一

个新兴的研究问题. 文献[20,22-23]对于 RAID-6 中的这个问题进行了深入的研究,并形成了一个最低密度 RAID-6 编码的体系(minimal density RAID-6 codes).

从传统的 EVENODD 和 RDP 编码出发,为了提高编码的容错能力,又派生出一系列新的可以容忍更高错误的编码,例如 STAR[21], Generalized EVENODD code[24], Feng's Code[25-26]等. 它们都属于横式阵列码的范畴,都具有较好的编码效率和存储利用率,限于篇幅,此不赘述.

2.3 纵式阵列码

纵式阵列码(vertical parity array codes)是指冗余存储在数据条块中的阵列编码方式,即在纵式编码中某些条块中既存储了数据元素,又存储了冗余元素. 纵式阵列码一般都具有较简单的几何构造结构,其计算复杂度的开销均匀地分摊到各块磁盘上. 在横式编码中,连续不断的写操作会带来磁盘热点非常集中的瓶颈问题(即冗余数据盘在连续的写操作中会连续用到). 而在纵式编码中,由于其结构上的均衡性,这一瓶颈得到了天然的解决. 但是纵式编码的均匀性也导致了磁盘相互之间的依赖性很强,这也就导致了其可扩展性很差. 随着研究的深入,各种各样的纵式编码被陆续提出,例如 X-code[11], P-code[27], WEAVER[28]等.

X-code[11]是一类 MDS 的纵式编码,与横式编码相比,它在编码、更新、重构阶段均具有理论上的最优效率,但是由于其磁盘之间的关系紧密,因此可扩展性较差. 如图 5 所示[15],在 X-code 的编码体系中,包括 n 块磁盘,在每块磁盘中包含 n 个元素,在这其中数据元素有 $n-2$ 个,冗余元素有 2 个,特别需要注意的是,n 必须恰好是一个大于 2 的素数.

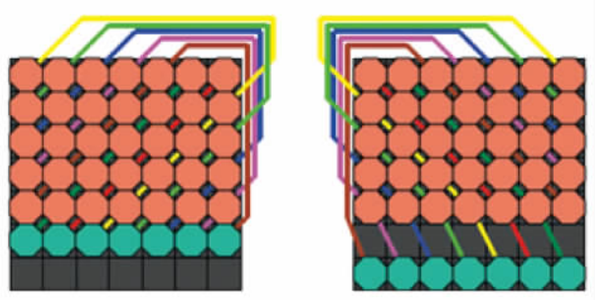


Fig. 5 X-code[15].

图 5 X-code[15]

X-code 在运算的各阶段均能够达到理论上最优的效率,这是横式编码所不具有的优点. 而且在存储利用率上,X-code 也是 MDS 的编码. 但是 X-code 对于磁盘数目 n 的要求非常严格,要求 n 必须是一个素数,才能保证在任意 2 块磁盘出错的情况下均能恢复出所有的数据. 由于 n 必须是素数的限制,以及各个冗余元素与几乎每块盘的数据元素均有关系,因此 X-code 的可扩展性很差. 在 X-code 中,冗余元素与数据元素联系紧密,这导致了很难从 X-code 推广出容错率更高的编码. 另外,在重构的过程中,必须按一定的顺序进行迭代的重构,不能利用并行操作提高其重构效率,这些均是 X-code 的不足之处. 但是总体来讲,X-code 仍然是一类非常好的纠删码技术. 目前大部分垂直码均是 MDS 的编码,其容错能力一般也不高. 但是 WEAVER[28]编码是一个例外,它可以提供高达 12 的容错能力,但是其空间利用率始终不能超过 50%.

2.4 LDPC 编码

LDPC(low-density parity-check code[29])码是又一类所有运算过程均完全基于异或(XOR)运算的编码. LDPC 编码不是 MDS 的编码,但是它非常接近 MDS 的存储效率,因此可以认为是渐进 MDS(asymptotically MDS[30])的. 并且从计算效率上来看,LDPC 编码比其他的 MDS 编码要快得多,运算复杂度可以达到 $O(m)$(m 表示 LDPC 编码所在的图上的边数).

LDPC 编码的编码方式实际上是构建在一个二部图之上. 比较典型的 LDPC 编码是构建在 Tanner 图上的编码方式[31],在这种编码方式中,二部图的左端包括了所有的数据元素和冗余元素,右端的每个点则代表了一个等式关系,表示与其相连的左端的各数据元素的异或之和为 0. 如图 6 所示[15],在 Tanner 图形式下的 LDPC 编码相当灵活,在重构运算中,可以忽略数据元素与冗余元素之间的区别,将它们视为同一类数据元素,直接采用以二部图理论为基础的重构算法. LDPC 编码最大的特点就是重构过程简单、速度很快.

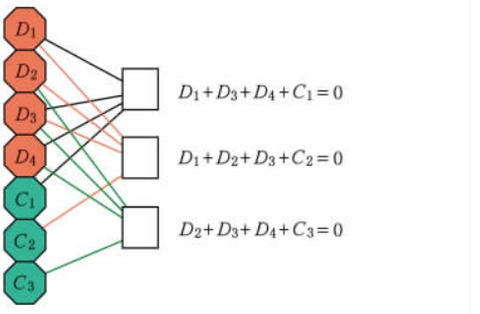


Fig. 6 LDPC code under Tanner graph[15].

图 6 Tanner 图下的 LDPC 编码[15]

对于 LDPC 编码的研究还包括基于矩阵表示方法的重构算法,以及纵式 LDPC 编码等等,这里限于篇幅,不再详细讨论.但是总体来说,对于 LDPC 码的研究更多的还是处于理论研究阶段,在实践中使用得很少,这个主要与 LDPC 编码的构造相对较难以及大部分 LDPC 码都是不规则编码有关,因此在 LDPC 编码领域还需要更为深入的研究.

2.5 若干新式编码

以上介绍的几类纠删码都是比较传统而又经典的.近年来,随着研究的深入,一些新式的纠删码不断涌现出来,丰富了纠删码的算法类型.如对于前面所提到的横式阵列码和纵式阵列码,它们各有优缺点.如何构造出横纵相结合的新式编码,以充分发挥出阵列码的优势,成为一个研究的新兴热点.目前对于这类编码的研究尚处于初级阶段,只出现少部分的编码,例如 HoVer 编码[32]等.

在传统的纠删码结构中,都是采用元素作为计算的基本单位,在这样的计算方法中,纠删码在容错能力、计算效率、存储利用率方面都不同程度地存在较大的缺陷,难以同时使得 3 个目标都达到理想的状态.以条块为计算的基本单位的 GRID 编码[8]解决了这个问题,它在本质上仍是横纵相结合的阵列码,但是它最高可以达到 15 的容错能力,并且实现起来也比较简单,在存储利用率方面最高可达 80% 以上.

在数学中,对于完全图的完美 1-因子分解的研究也为阵列码的构造提供了新的方法,文献[33-34]对于这个问题进行了深入的研究,严格地证明了它们之间的相互转化关系,为阵列码的构造提供了一大类新的方法.其中文献[33]的编码通常称为 B-code,另外,P-code[27]也正是利用这种方法构造出来的.

对于存储系统纠删码的易实现性也成为研究中一个值得关注的问题,为此研究人员又提出一类新的称为 Cyclic Code[35-36]的编码.对于这类编码的研究还处于起步阶段,也许会成为一个未来研究的热点.

3 评价纠删码的重要指标及未来改进趋势

纠删码的容错能力、存储利用率以及计算效率是在存储系统中评价纠删码的重要指标,其中,计算效率又分为编码、更新和重构三大计算效率.在本节中,我们将从这几个方面对常见的纠删码作出比较和分析,并指出纠删码在各个指标上未来的改进趋势.

3.1 容错能力和磁盘要求

容错能力是纠删码算法的基石,同时,很多纠删码对于磁盘数目和单一条块中的元素数目也有着很多要求,我们在本节中将首先对这两方面展开分析.如表 1 所示,我们列出了当前较为常见的纠删码的容错能力以及编码对于数据磁盘数目和单一条块中元素数目的要求.

Table 1 Fault Tolerance Ability and Requirements for the Number of Data Disks and the Number of Elements in Single Strip in Erasure Codes

表 1 纠删码的容错能力以及编码对于数据磁盘数目和条块中元素数目的要求

Erasure Code	Fault Tolerance Ability	Requirement for the Number of Data Disks	Requirement for the Number of Elements in Single Strip
Reed-Solomon code	Arbitrary	Depend on the length of codeword	Depend on the length of codeword
EVENODD	2	Prime	Number of disks−1
RDP	2	Prime−1	Number of disks
Liberation code	2	None	Prime that not less than the number of disks
STAR	3	Prime	Number of disks−1
X-code	2	Prime	Number of disks
WEAVER	≤12	None	None
B-code	2	Solution for corresponding mathematics problem	2×Number of disks or 2×Number of disks+1
P-code	2	Prime or Prime−1	Number of disks divides 2

在表1中,大部分编码对于数据磁盘数目和单一条块中元素数目的要求都非常严格,一般都要求数据磁盘的数目为素数,这就对磁盘阵列提出了很高的要求,同时也导致了这些编码的可扩展性较差.在容错能力为2的编码中只有B-code编码和Liberation编码没有对于数据磁盘数目的素性提出要求,但是它们也有各自的问题:B-code的构造方法对应着数学中的一个公开难题,因此对于小规模的B-code只可以使用搜索技术找寻其构造方法,而对于大规模的B-code的构造则只能寄希望于该数学问题的圆满解决.Liberation编码同样对于磁盘数目没有非常苛刻的要求,但是该编码要求在单一条块中包含的元素数目必须恰好是一个素数,并且要求数据磁盘的数目不能超过单一条块中的元素数目,这样的条件同样很难得到满足.

在横式编码中,对于数据磁盘数目的要求是可以灵活变通的.因为在横式编码中,大部分磁盘是专用于存储数据的,于是可以在逻辑上,令这些磁盘上的所有数据均恰好为"0",而在实际应用中去除掉这些数据磁盘,这样就可以使得横式编码在实际应用中摆脱对于数据磁盘数目的限制.这样的磁盘扩展方法是由于这些磁盘上没有存储冗余,所以这样的方法对于纵式编码来说并不适用.

从表1中还可以发现,大部分纠删码的容错能力都不太高,但是也有两类编码的容错能力是不错的,并且它们对于数据磁盘数目和单一条块中的元素数目都没有特殊要求,但是这两类编码也有各自的问题.RS编码虽然可以适应任意的容错能力的需求,但是RS编码是基于Galois域$GF(2^w)$上进行运算的,其运算效率很差,并且RS编码对于生成矩阵的要求极为苛刻,导致RS编码的构造很难,目前已知构造出来的运算都比较复杂.同样,WEAVER编码虽然也拥有较好的容错能力,但是它没有系统的构造方法,大多都需要依靠搜索技术才能确定它的编码构造,这样的方法难以推广.

3.2 空间利用率

纠删码的使用产生了冗余的数据,这就带来了额外的空间开销.如何充分地利用存储资源,提高空间利用率,成为纠删码研究中的一个重要问题.

在目前常见的纠删码中,MDS的编码以其理论上最优的存储利用率而被广泛推崇.前面介绍的大部分编码都是满足MDS的编码,例如RS,EVENODD,RDP,Liberation Code,STAR,X-code等.但也有一些编码是非MDS的,这些编码往往在数据磁盘数目限制、容错率、计算效率等方面有着其独到的优势.下面,我们将讨论这些编码的特点,并分析空间利用率和这些优势之间的关系.

WEAVER编码[28]与前面介绍到的众多编码不同的是,它不再一定是MDS的编码,其空间利用率不到50%,但是容错率却可以高达12.另外,WEAVER编码突破了素数的限制,对于任意不超过12的磁盘数目均存在对应的WEAVER编码的构造方法.这些优点都是从它的空间损失上换取到的,可以说,它在数据磁盘数目限制、容错率、空间效率和计算效率等方面达到了一个比较好的平衡.

LDPC编码是另一类非MDS的编码,但是它非常接近MDS的存储效率,因此可以认为是渐进MDS的.LDPC编码在空间上的损失同样换来了计算效率上的优势,从计算效率来看,LDPC编码比其他的MDS编码都要快,其运算复杂度可以达到$O(m)$(m在这里表示LDPC编码所在的图上的边数).

3.3 编码效率

一个纠删码的实用性往往取决于其在编码阶段的开销情况,因此,编码效率成为评价纠删码的又一重要指标.

RS编码是定义在Galois域$GF(2^w)$上的纠删码,在它的运算中,加法是较为简单的,但是乘法的运算却非常复杂,甚至需要借助离散对数运算和查表才能实现,这样的纠删码的编码效率是远低于只采用异或运算的编码的.尽管柯西RS编码克服了标准RS编码中的乘法问题,但是其生成矩阵中"1"的数目较多仍然导致了其编码效率不高,尽管文献[17]对此作了深入的研究,最优化了柯西RS纠删码的编码效率,但是这个编码效率与其他的阵列码相比还是有较大的差距.

对于非MDS的编码,它们的构造一般比较复杂,不具有规律性,因此难以系统地讨论它们的编码效率.下面,我们将采用在编码过程中生成单个冗余元素需要作的平均异或(XOR)次数来度量各类MDS纠删码的编码效率.很显然,在MDS的横式纠删码中,对于数据磁盘数目为n的编码,生成单个冗余元素的理论最少异或次数为$n-1$,因此,我们将生成单个冗余元素的平均异或次数除以$n-1$,来正规化地表示各类横式纠删码的编码开销.如图7所示,图中纵轴表示的是各类纠删码中生成单个冗余元素的平均异或次数与理论最少异或次数之比(很显然,纵轴上的1代表的是理论最优的编码效率).

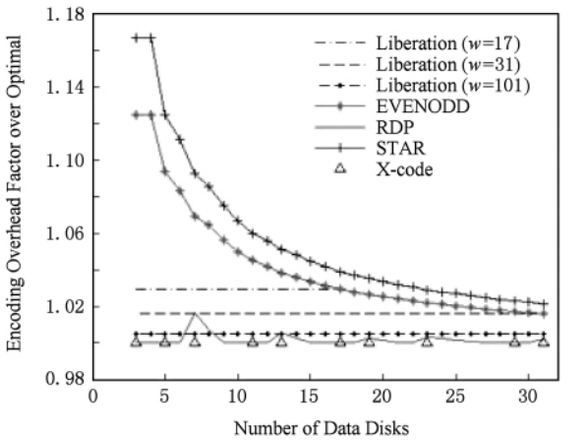


Fig. 7 Encoding efficiency comparison for erasure codes.

图 7 各类纠删码的编码效率对比

在图 7 中,我们使用 X-code 来代表 MDS 的纵式编码,它们在编码效率上可以达到理论上的最优效率,但是由于他们对于磁盘数目的限制非常严格,只能当磁盘数目为素数时才能满足编码构造,因此,在图 7 中,X-code 的图像是一些离散在磁盘数目为素数上的点.

如同 4.1 节中提到的,横式编码可以在实际应用中"减少"其磁盘数目,在图 7 中给出了各类横式纠删码的编码效率对比. RDP 的编码效率相对较好,但是当"缩短"其磁盘数目时,效率也受到了较大的影响. 对于磁盘数目的"缩短"方案,由于编码的不对称性,在 RDP 编码中"去掉"靠前的磁盘是最好的选择,而相反的是,EVENODD 和 STAR 编码中"去掉"靠后的磁盘则比较好. 从图 7 中还可以看出,Liberation 编码较以往传统的横式编码而言,在编码效率上得到了大大的提高,特别是随着编码中 w 这一参数的增大,其编码效率越来越接近理论上的最优编码效率[20].

对于编码效率的提高,除了研究在生成矩阵中"1"的数量更少的纠删码以外,另一类可以提高编码效率的技术就是 Parity Scheduling 技术[20]. 如图 8 所示[15],Parity Scheduling 技术通过改变各冗余的计算顺序以及生成矩阵中某些重叠部分的预计算,使得后续计算充分利用到前缀计算的结果,以此来减少编码运算中的异或次数. Parity Scheduling 的方法不仅可以应用于编码运算中,在重构运算中同样可以发挥巨大的作用. Parity Scheduling 技术目前还只能适用于少数生成矩阵,还没有一种有效的可以使得任意生成矩阵都能得到最优 Parity Schedule 的一般算法. 对于这类算法的研究目前还仍是一个开放问题,还需要更深入的研究.

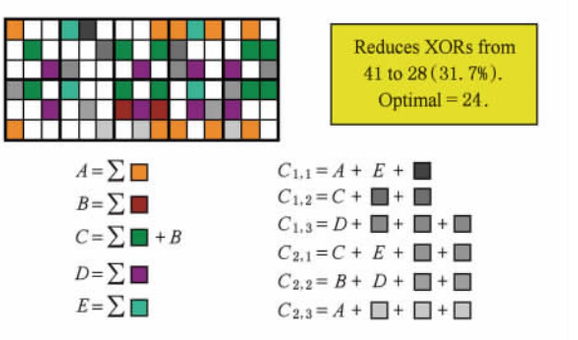


Fig. 8 Parity Scheduling technique[15].

图 8 Parity Scheduling 技术[15]

3.4 更新效率

纠删码应用在通信传输中时数据不存在更新的问题,因此在过去的研究中,编码算法和重构算法得到了更多的关注. 然而在存储系统中,连续不断的 I/O 是其重要的特点,由于系统中有了冗余,当数据被更新时,同时也就需要去更新与它相关联的所有冗余,这就增加了存储系统的更新开销. 由于数据是不断被更新的,因此,对于存储系统中的纠删码,其更新效率更受关注,更新效率成为了评价一类纠删码最重要的标志.

在本节中,我们使用更新单个数据元素时平均需要同步更新的冗余元素数目来表示一类纠删码的更新效率. 如图 9 所示,我们对比了各类容错率为 2 的纠删码的更新效率:

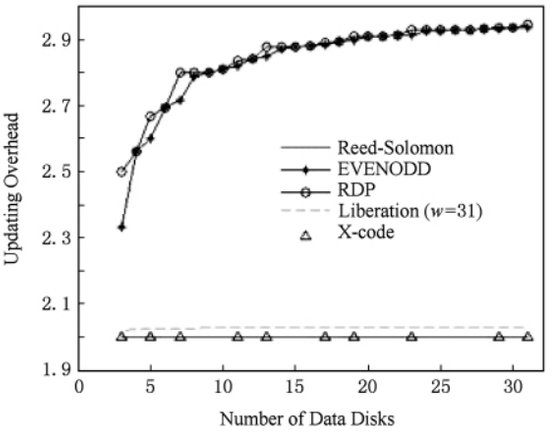


Fig. 9 Updating efficiency comparison for erasure codes.

图 9 各类纠删码的更新效率对比

尽管在 RS 编码中,更新单个数据元素时只需要同步更新 2 个冗余元素,是理论上最优数目,但是由于 RS 编码的运算是定义在 Galois 域 $GF(2^w)$ 上的,它的运算效率远低于只采用异或运算的编码,因此实际上 RS 编码的更新效率很低. 而对于柯西 RS

编码,与编码效率的分析一样,其生成矩阵中"1"的数目较多导致了其更新效率不高,尽管文献[17]对此作了深入的研究,最优化了柯西 RS 纠删码的生成矩阵,但是它的更新效率与其他的阵列码相比还是有较大的差距.

与 3.3 节一样,我们采用 X-code 来代表所有纵式编码的更新效率,对于磁盘数目的严格限制导致纵式编码在图 9 中仍然表现为一些离散的点,但是它的更新效率仍然可以达到理论上最优效率.

EVENODD 和 RDP 的更新效率随着磁盘数目的增长而逐步降低,如图 9 所示,同时在理论上可以证明的是,在 EVENODD 和 RDP 编码中,更新单个数据元素时,平均需要同步更新的冗余元素的数目上界是 3 个.由于 STAR 的容错率为 3,与容错率为 2 的编码在更新效率上不具有可比性,因此未在图 9 中标示出来,但是由于 STAR 是 EVENODD 的一种扩展,因此通过理论分析和证明可知,STAR 的更新效率也是随着磁盘数目的增长而逐步降低的,并且更新单个数据元素时,其平均需要同步更新的冗余元素的数目上界是 5 个.

文献[20,22-23]证明了 MDS 横式编码的更新效率不可能达到数学上的理论最优效率. Liberation 编码的生成矩阵中"1"的数目就是横式编码在理论上的下界.因此在图 9 中,Liberation 编码的更新效率远好于其他横式阵列码.由此可知,最优更新效率与 MDS 的特性不可兼得.

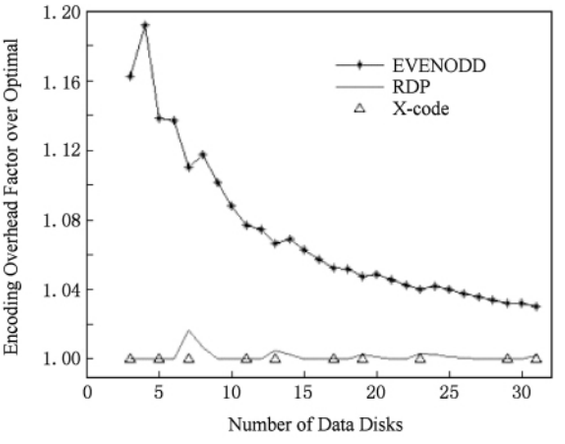
3.5 重构效率

在纠删码研究中,重构过程的效率决定了纠删码算法对于整个存储系统的影响,那么,各类纠删码的重构效率到底怎样呢?我们将在本节中进行讨论.

RS 编码的重构过程通常分为 2 步,首先需要作矩阵求逆,然后才能进行矩阵乘法运算恢复出错的元素.而且 RS 编码是定义在 Galois 域 $GF(2^w)$ 上的纠删码,其乘法运算非常复杂,矩阵的求逆运算就更加复杂,因此 RS 编码的重构效率非常低下,对存储系统效率的影响很大.

在容错率为 2 的编码中,当存储系统出现单个磁盘的错误时,如果出错的磁盘是冗余磁盘,那么就可以完全按照编码的方法来重构出错的磁盘,否则,也可以很轻易地按照类似于 RAID-5 的方法来重构出错的元素.因此下面我们将只讨论恰好 2 块磁盘出错情况下的重构效率.

在图 10 中,我们枚举了在各类编码中所有可能出错的磁盘组合情况,用以严谨地分析编码的重构效率.与 4.3 节中分析编码效率时类似,我们采用重构运算中恢复单个错误元素需要的平均异或次数作为评价一类纠删码的重构效率的标准.在 MDS 的横式编码中,对于磁盘数目为 n 的存储系统,恢复单个错误元素理论上需要的最少异或次数是 $n-1$,因此在图 10 中,与 4.3 节中分析编码效率时一样,我们将恢复单个错误元素需要的平均异或次数除以 $n-1$,用以正规化地表示各类横式纠删码的重构开销.如图 10 所示,图中纵轴表示的是各类纠删码中恢复单个出错元素需要的平均异或次数与理论最少异或次数之比(很显然,纵轴上的 1 代表的是理论上最优的重构效率).



ig. 10 Decoding efficiency comparison for erasure codes.

图 10 各类纠删码的重构效率对比

与 3.3 节、3.4 节一样,我们使用 X-code 来代表纵式编码,特殊的排布结构决定了它的重构效率依然可以达到理论上的最优效率,但是大部分纵式编码对于磁盘数目严格的限制导致 X-code 仍然表现为一些离散的点.

Liberation 编码和柯西 RS 编码并未在图 10 中标明,这是因为它们的重构矩阵都比较复杂,导致其重构效率不高.然而,4.3 节中提到的 Parity Scheduling 技术[20]也同样适用于重构运算,这样可以显著地提高重构效率,提高后的重构效率甚至可以接近理论上的最优效率,具体的应用方法可参考文献[17,20],此不赘述.

3.6 不同存储系统对于纠删码各类性能的需求

不同的存储系统有着其独到的对于纠删码的需求,在本节中,我们将从应用的角度对这个问题进行分析.

1) 磁盘阵列系统:这类系统的特点是数据磁盘的数量很大,但是冗余磁盘的数目相对较少.在这类

系统中,计算效率通常是最受关注的一个因素,纠删码通常是安装在磁盘控制器上的,因此阵列码对于这类系统比较合适,RAID[37-39]是这类编码系统最常见的模型.

2) P2P 系统:这类系统的特点是用于存储数据和冗余的介质都非常多,这类系统具有廉价性.另一方面,由于 P2P 系特有的流动性,存储介质出错或者离开网络的概率也非常高,因此需要高容错的纠删码支持.基于以上特点,利用多副本技术实现容错是 P2P 系统的最佳选择.

3) 分布式存储系统:在这类系统中,存储介质的数目同样是相当多的,且它对可用性的要求很高.因此,纠删码的重构效率成为影响这类系统性能的关键,而存储利用率并不是关注的焦点.

4) 归档存储:数据的高可靠性是这类系统唯一的要求,需要高容错率的纠删码才能适用于这类系统.

4 总结和展望

随着计算机科技的进步和存储系统海量化的发展,为了保证存储系统的高可靠性和可用性,纠删码越来越受到存储系统研究者们的重视.本文介绍了当前典型和常见的各类纠删码,并从评价纠删码性能的各项重要指标的角度对比和分析了现有的纠删码技术,为未来纠删码可能的研究方向提供了理论依据.

经过多年的发展,RS 编码作为最早的纠删码类型,已被研究得较为透彻.它无论是在编码效率、更新效率,还是重构效率上,都难已再有提高的空间[8].而反观阵列码,因它是建立在异或运算之上的一类纠删码,由于其运算简单,容易实现,因而在近年来得到了广泛的关注,还有很多工作值得继续研究.

在阵列码中,由于目前提出的各类 RAID-6 编码总是或多或少地存在着各自的缺陷,还没有一类被公认为 RAID-6 最佳解决方案的纠删码出现,因此在未来的一段时间以内,对于 RAID-6 编码的研究仍会是一个热点.同时,随着大规模存储系统的应用与发展,低容错率的纠删码已难以适应存储系统对于高可靠性的需求,因此,更高容错率的纠删码也必然成为未来发展的一个方向.

在对于影响纠删码性能的各项参数的分析和对比中,我们发现,各类纠删码都有各自的优势和缺点,例如纵式编码在各项计算效率上都可以达到理论上的最优效率,但是却受到了磁盘数目严格的限制,而横式编码虽然可以"减少"其磁盘数目,却难以在计算效率上达到最优.如何利用这些编码结构的优点构造出新型的纠删码,也是一个未来很有趣的问题.

目前纠删码在容错能力、计算效率、存储利用率方面都不同程度地存在着较大的缺陷,难以同时使得 3 个目标都达到理想的状态.如何构造出各方面具优的纠删码仍是未来研究中任重道远的问题.

参 考 文 献

[1] Layman P, Varian H R. How much information 2003? [EB/OL]. [2010-10-18]. http://www2.sims.berkeley.edu/research/projects/how-much-info-2003

[2] Pinheiro E, Weber W D, Barroso L A. Failure trends in a large disk drive population [C] //Proc of the 5th USENIX Conf on File and Storage Technologies. Berkeley, CA: USENIX Association, 2007: 17-28

[3] Schroeder B, Gibson G A. Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you? [C] //Proc of the 5th USENIX Conf on File and Storage Technologies. Berkeley, CA: USENIX Association, 2007: 1-16

[4] Bairavasundaram L N, Goodson G R, Pasupathy S, et al. An analysis of latent sector errors in disk drives [C] //Proc of 2007 ACM SIGMETRICS Int Conf on Measurement and Modeling of Computer Systems. New York: ACM, 2007: 289-300

[5] Zheng Qingji. Research on erasure code for secure storage system [D]. Shanghai: Shanghai Jiaotong University, 2009 (in Chinese)
(郑清吉. 安全存储系统中纠删码技术研究[D]. 上海: 上海交通大学, 2009)

[6] Hafner J M, Deenadhayalan V, Rao K, et al. Matrix methods for lost data reconstruction in erasure codes [C] //Proc of the 4th USENIX Conf on File and Storage Technologies. Berkeley, CA: USENIX Association, 2005: 183-196

[7] Hafner J M, Deenadhayalan V, Kanungo T, et al. Performance metrics for erasure codes in storage systems, RJ 10321 [R]. San Jose, CA: IBM Research, 2004

[8] Li M, Shu J, Zheng W. GRID Codes: Strip-based erasure codes with high fault tolerance for storage systems [J]. ACM Trans on Storage, 2009, 4(4): 1-22

[9] Blaum M, Brady J, Bruck J, et al. EVENODD: An efficient scheme for tolerating double disk failures in RAID architectures [J]. IEEE Trans on Computer, 1995, 44(2): 192-202

[10] Corbett P, English B, Goel A, et al. Row-diagonal redundant for double disk failure correction [C] //Proc of the 3rd USENIX Conf on File and Storage Technologies. Berkeley, CA: USENIX Association, 2004: 2-15

[11] Xu L, Bruck J. X-code: MDS array codes with optimal encoding [J]. IEEE Trans on Information Theory, 1999, 45(1): 272-276

[12] MacWilliams F J, Sloane N J A. The Theory of Error-Correcting Codes [M]. New York: North-Holland, 1977

[13] Reed I S, Solomon G. Polynomial codes over certain finite fields [J]. Journal of the Society for Industrial and Applied Mathematics, 1960, 8(2): 300-304

[14] Roth R M, Lempel A. On MDS codes via Cauchy matrices [J]. IEEE Trans on Information Theory, 1989, 35(6): 1314-1319

[15] Plank J. Erasure codes for storage applications [EB/OL]. [2009-12-11]. http://web.eecs.utk.edu/~plank/plank/papers/FAST-2005.html

[16] Plank J, Luo J, Schuman C, et al. A performance evaluation and examination of open-source erasure coding library for storage [C] //Proc of the 7th USENIX Conf on File and Storage Technologies. Berkeley, CA: USENIX Association, 2009: 253-266

[17] Plank J, Xu L. Optimizing cauchy reed-solomon codes for fault-tolerant network storage applications [C] //Proc of the 5th IEEE Int Symp on Network Computing Applications. Piscataway, NJ: IEEE, 2006: 1-8

[18] Blaum M, Farrell P, Tilborg H. Array Codes [M]. Amsterdam, Netherlands: Elsevier Science B V, 1998

[19] Feng J, Chen Y, Summerville D. EEO: An efficient MDS-like RAID-6 code for parallel implementation [C] //Proc of the 33rd IEEE Sarnoff Symp. Piscataway, NJ: IEEE, 2010: 1-5

[20] Plank J. The RAID-6 liberation codes [C] //Proc of the 6th USENIX Conf on File and Storage Technologies. Berkeley, CA: USENIX Association, 2008: 97-110

[21] Huang C, Xu L. STAR: An efficient coding scheme for correcting triple storage node failures [C] //Proc of the 4th USENIX Conf on File and Storage Technologies. Berkeley, CA: USENIX Association, 2005: 197-210

[22] Blaum M, Roth R M. On lowest density MDS codes [J]. IEEE Trans on Information Theory, 1999, 45(1): 46-59

[23] Plank J. The RAID-6 Liber8Tion code [J]. Int Journal of High Performance Computing Applications, 2009, 23(3): 242-251

[24] Blaum M, Bruck J, Vardy A. MDS array codes with independent parity symbols [J]. IEEE Trans on Information Theory, 1996, 42(2): 529-542

[25] Feng G, Deng R H, Bao F, et al. New efficient MDS array codes for RAID Part I: Reed-Solomon-like codes for tolerating three disk failures [J]. IEEE Trans on Computers 2005, 54(9): 1071-1080

[26] Feng G, Deng R H, Bao F, et al. New efficient MDS array codes for RAID Part II: Rabin-like codes for tolerating multiple (≥4) disk failures [J]. IEEE Trans on Computers 2005, 54(12): 1473-1483

[27] Jin C, Jiang H, Feng D, et al. P-code: A new RAID-6 code with optimal properties [C] //Proc of the 23rd Int Conf on Supercomputing. New York: ACM, 2009: 360-369

[28] Hafner J L. WEAVER codes: Highly fault tolerant erasure codes for storage systems [C] //Proc of the 4th USENIX Conf on File and Storage Technologies. Berkeley, CA: USENIX Association, 2005: 211-224

[29] Gallager R G. Low-density parity-check codes [J]. IEEE Trans on Information Theory, 1962, 8(1): 21-28

[30] Luby M G, Mitzenmacher M, Shokrollahi M A, et al. Efficient erasure correcting codes [J]. IEEE Trans on Information Theory, 2001, 47(2): 569-584

[31] Tanner R M. A recursive approach to low complexity codes [J]. IEEE Trans on Information Theory, 1981, 27(5): 533-547

[32] Hafner J L. HoVer erasure codes for disk arrays [C] //Proc of the 2006 Int Conf on Dependable Systems and Networks. Piscataway, NJ: IEEE, 2010: 217-226

[33] Xu L, Bohossian V, Bruck J, et al. Low-density MDS codes and factors of complete graphs [J]. IEEE Trans on Information Theory, 1999, 45(6): 1817-1826

[34] Li M, Shu J. On the equivalence between the B-code constructions and perfect one-factorizations [C] //Proc of the 2010 IEEE Int Symp on Information Theory. Piscataway, NJ: IEEE, 2010: 993-996

[35] Cassuto Y, Bruck J. Cyclic lowest density MDS array codes [J]. IEEE Trans on Information Theory, 2009, 55(4): 1721-1729

[36] Li M, Shu J. On cyclic lowest density MDS array codes constructed using starters [C] //Proc of the 2010 IEEE Int Symp on Information Theory. Piscataway, NJ: IEEE, 2010: 1315-1319

[37] Patterson D A, Chen P M, Gibson G, et al. Introduction to redundant arrays of inexpensive disks [C] //Proc of the IEEE COMPCON'89. Piscataway, NJ: IEEE, 1989: 112-117

[38] Patterson D A, Gibson G, Katz R H. A case for redundant arrays of inexpensive disks [C] //Proc of the ACM SIGMOD'88. New York: ACM, 1988: 109-116

[39] Chen P M, Lee E K, Gibson G, et al. RAID: High-performance, reliable secondary storage [J]. ACM Computing Surveys, 1994, 26(2): 145-185

Luo Xianghong, born in 1987. PhD candidate. His main research interests include storage systems, coding theory.

Shu Jiwu, born in 1968. PhD, professor and PhD supervisor. Senior Member of China Computer Federation. His main research interests include network storage, storage security and reliability, parallel process technologies, etc(shujw@tsinghua.edu.cn).