中国科学: 信息科学 2016 年 第 46 卷 第 4 期: 523–538
SCIENTIA SINICA Informationis

论 文

一类多容错的阵列纠删码

唐聃, 舒红平*

成都信息工程大学, 成都 610225
* 通信作者. E-mail: shp@cuit.edu.cn

收稿日期: 2015–06–15; 接受日期: 2015–08–02; 网络出版日期: 2016–04–13
国家自然科学基金项目 (批准号: 61501064) 和四川省教育厅科技成果转换重大培育项目 (批准号: 15CZ0019) 资助

摘要 阵列纠删码具有构造方法简洁、运算效率高等诸多优点, 不仅是增强存储系统可靠性的理想方法之一, 也在秘密共享、多路传输等诸多领域有着重要应用. 但容错能力较低一直是阻碍阵列码实用化的一大障碍. 目前已知的阵列码中, 具有最大容错能力的是 Grid 码, 其容错能力通常情况下也只能达到 15. 针对这一情况, 本文提出了一类可多容错的阵列码, 称作斜率码. 该码的编码和数据恢复等操作所需的所有计算均为二进制的异或运算, 具有极高的运算效率, 而其简单的构造形式有利于软硬件的实现. 虽然不具备 MDS 性质, 但斜率码的存储效率可以通过加大条块尺寸而不断增加. 该编码具有理论可达的最小更新代价, 这可以为整个存储系统并发存取操作的高效性提供保障.

关键词 纠删码 阵列码 水平码 存储系统 多容错

1 引言

随着信息技术对各个行业和领域的不断渗透, 数据量正呈现出持续爆炸式增长的趋势. 为了应对由数据量的快速增长而带来的数据存储可靠性问题, 同时也为了提高数据访问的并发效率并降低成本, 通常有效的做法是使用多个存储节点 (每个节点可以是一台 PC、一台服务器、一个阵列或移动终端等可用作数据存储的设备) 共同构建一个存储系统 (通常是基于网络的分布式存储系统, 其雏形可以追溯到集中式的 RAID 系统). 而各个行业和领域存储数据量的持续增加导致存储系统的规模 (节点数) 在不断扩大, 文献 [1] 表明 Google 和 Facebook 等企业已拥有多个节点规模在 3000 到 4000 的存储系统. 而在单个节点平均失效概率一定的情况下, 总的节点数目增加就意味着一个系统在任意时间段内可能同时失效的节点数目也随之增长. 此外, 地震、洪水、火灾和泥石流等自然灾害, 黑客攻击、恐怖袭击或操作失误等故意或非故意的人为破坏等突发事件也可能造成一个存储系统中的多个节点同时失效. 因此, 高容错能力是保障一个大规模存储系统高可靠性的关键, 而针对存储系统的多容错技术研究对保障存储系统的数据安全具有重要的意义.

目前, 最常见的实用数据容错方法主要为基于镜像备份的可靠性增强策略, 本文简称为复制备份策略 (严格地讲, 镜像备份与复制策略在具体实现的技术细节上存在差异, 但从逻辑实质上来讲这两种策略又具有统一性, 所以本文统称为复制备份). 基于复制备份的数据存储系统容错策略把多个相同

引用格式: 唐聃, 舒红平. 一类多容错的阵列纠删码. 中国科学: 信息科学, 2016, 46: 523–538, doi: 10.1360/N112015-00130

的数据副本分别存储到不同的存储节点上, 冗余数据即是原文件的多个副本. 显然, 在使用基于复制备份的策略构建一个具有容错能力的存储系统时, 如果需要在最多 $k(\geqslant 1)$ 个节点同时失效后有效恢复数据 (即 k 容错或容错能力为 k), 则需要把原始数据复制 k 份, 并分别存放在不同的节点上. 基于复制备份策略的容错方法和体系是目前研究最深入, 也是在各种商用存储系统中应用最广的一种数据存储可靠性增强方法, 著名的云存储系统 GFS [2], Hadoop [3] 和 Dynamo [4] 等都采用了此种方法. 基于复制备份策略的存储容错系统具有方案简洁、易于实施、构建成本低、便于扩展、无需任何运算等明显优势, 但是其随着容错能力提升而不断下降的存储效率和不断增加的更新代价却是其巨大缺陷.

基于编码 (主要是纠删码) 的存储系统容错方法 (本文中简称编码冗余策略), 是一种近年来日益引起业界重视的存储系统可靠性增强方法. 编码冗余策略对随机失效模型和关联性失效模型都具有很好的预防效能, 与复制备份策略相比, 编码冗余策略最大的优势在于保证容错能力的前提下可以大大减少修复带宽, 降低更新代价, 同时提高存储效率. 因此, 学者们逐渐认可了基于纠删码的容错方法在提高存储系统可靠性上的重要性, 而纠删码也逐步成为了存储系统可靠性增强的重要方法和研究热点. 对基于纠删码来构建存储系统的容错机制研究, 国外 [5~9] 主要有 IBM 的 Blaum 团队、加州理工的 Bruck 团队和田纳西大学的 Plank 团队等. 国内 [10~13] 对该领域研究较为深入的有清华大学的舒继武团队、华中科技大学的冯丹团队、中国科技大学许胤龙团队、西北工业大学李战怀团队等.

而目前对基于编码冗余策略的存储系统可靠性增强机制的研究, 主要集中在如下几个领域. 一是 RS (Reed-Solomon) 类纠删码. RS 类纠删码是目前已知唯一一类可纠任意个删除错的 MDS 码, 达到香农限 (Shannon limit), 因而基于 RS 码的数据容错方案的容错能力理论上不受限制且具有最优的存储效率. 然而, 由于 RS 码涉及多元有限域上的运算, 计算复杂度高 (特别是多元有限域上的乘法运算), 其计算代价将难以被任何大规模实用系统接受. 针对这一问题, 学者们也提出了诸多有益的局部改进方法, 最典型的方法是将多元有限域的运算转换到二元域上进行 [14,15], 从而提高运算速度. 而近年来, 也有学者在大幅度提高多元有限域运算效率方面作出了卓有成效的工作, 如 Plank 等 [16] 提出的 GF-Complete. 一旦运算速度的问题得到彻底解决, RS 码无疑将是存储容错核心方案的首选方法之一. 不过, 目前仍然只有一些实验型的存储系统, 如 OceanStore 和 TotalRecall [17] 等使用了该方法. 二是 LDPC (low-density parity-check code) 类纠删码. LDPC 类纠删码 [18] 是一类编码和译码过程完全基于异或运算的纠删码, 因此与 RS 码相比, 其纠删能力不受限且运算效率要高出几个数量级. 尽管 LDPC 类纠删码不是 MDS 码, 但其在纠删能力相同时, 其码率十分接近 MDS 码的码率. 虽然 LDPC 码有着众多的优点, 但是其应用于存储系统时仍然面临不少问题. 例如码字结构的不规则性以及成功译码具有概率性等, 都极大地阻碍了 LDPC 码在存储系统可靠性增强领域的发展, 因此基于 LDPC 码的容错方法在实际存储系统中的应用还很少, 且已有方法多是以理论研究或实验为主, 如 SpreadStore [19]. 除了以上两种编码体系, 阵列纠删码则是引起业界日益关注的另外一种可增加存储系统可靠性的技术手段. 阵列纠删码 (通常简称为阵列码) 的编码过程只使用简单的二进制异或运算, 具有实现过程简单、运算效率高、计算域不会随着码元的增大而增加等特点, 这些特性都使得阵列码非常适合应用于大规模存储阵列. 按照存储效率, 阵列码可分为 MDS 编码和非 MDS 编码. 其中以 EvenOdd 码 [5] 和 X 码 [7] 为代表的二容错阵列码, 以及以 Star 码 [8] 和扩展 X 码 [11] 为代表的三容错阵列码都是非常典型的 MDS 阵列码. 但是多容错 (可恢复 4 个以上的节点删除错误) 的 MDS 阵列码的研究工作却举步维艰, 一直到现在也没有容错能力大于 3 且完全使用二进制异或运算的 MDS 阵列码被提出. 研究者们在牺牲了一定的存储效率后设计出了可以多容错但不具备 MDS 性质的阵列码, 包括 Weaver 码 [20], Hover 码 [21], E 码 [12] 和 Grid 码 [13] 等. 这类阵列码的容错能力与 MDS 阵列码相比有了一定幅度提高, 通常可以达到 10 以上甚至更高, 但是在实用化的过程中却仍然存在很多

困难: 首先是大多编码的容错能力缺少理论证明, 或者其容错能力是通过计算机实验 (穷举法等) 得出的结论, 或者其容错能力结论带有概率性 (如某种编码的容错能力达到 n 的概率很大, 在大多数情况下可行, 但是某些条件下又会出现 n 甚至小于 n 个错误无法恢复的个案), 所以当存储系统需要扩展或容错数目要继续增加时很难利用这些编码. 二是限制条件较为严格, 包括存储阵列中条带 (Stripe) 或条块 (Strip) 尺寸需要为一个素数或与素数保持某种线性关系, 或者多容错能力只能在一种比较特殊的条件下才能达到等, 这导致了整个系统可扩展性差. 三是存储效率, 计算复杂度等性能指标随着容错数目的增加而快速降低, 例如, 当构建一个可纠 10 个错误的 Weaver 码系统时, 其存储效率将不足 20%, 仅略高于复制冗余策略. 四是没有考虑到大规模存储系统的其他性能需求, 如更新代价等. 而最大容错能力仍然较低也造成了阵列码实用性不高的问题. 针对这一问题, 本文提出了一类新的阵列纠删码, 称作斜率阵列纠删码, 简称斜率码, 该编码使用了最简洁的几何形式构造方法进行编码和译码, 具备理论上可达的最低更新代价, 存储效率也能在一定条件下不断趋近香农限, 最重要的是在理论上可容许任意数量错误, 这对提高阵列码在存储容错领域的实用性, 以及扩大其应用范围具有一定的意义.

本文的组织结构如下: 下一节将对文中使用到的存储容错编码领域的概念, 术语及符号进行必要的定义或说明, 这些定义包括一些通用的概念, 也包含部分本文特定使用的术语. 第 3 节将对斜率码的构造过程进行详细说明, 包括编码和数据恢复. 斜率码基本特性的证明和分析将在第 4 节中给出. 最后是全文的总结.

2 基本概念及符号表示

对于存储系统可靠性增强领域中的一些通用的概念, 本文将沿用目前大多数文献的提法和说明, 如数据、冗余、校验、元素、条带、条块等概念和定义可参考文献 [6]. 本节仅列出一些文中特定使用的符号和概念说明.

取模运算符 %: 为了表达的简洁, 文中使用符号 % 表示取模运算, 如 $a\%p = \mathrm{mod}(a, p)$.

取整、上取整和下取整: 文中使用符号 $[]$, $\lceil\rceil$ 和 $\lfloor\rfloor$ 分别表示取整、上取整和下取整. 例如, $[3/2] = 2$, $\lceil 3/2\rceil = 2$, $\lfloor 3/2\rfloor = 1$.

码链及斜率: 将存储系统的数据阵列 (本文特指所有数据元素构成的尺寸为 $m \times n$ 的阵列) 作为一个二维坐标系, 以向右的水平轴作为坐标系的正向 x 轴, 以向下的垂直轴作为坐标系的正向 y 轴, 所有阵列中的元素作为坐标系上的一个有着 x 坐标和 y 坐标的点, 则计算某个校验元素所需要的所有数据元素可以看作是该坐标系上的一条直线, 而该直线上所有数据元素按其在阵列中的所在行有前后之分, 所以更像是一个链条. 我们通过这个链条上的各个元素对阵列码进行编码和译码, 因此文中称为码链. 值得一提的是, 此概念并非本文首次提出, 在之前的多篇阵列码文献中均有提及, 比如 X 码将链条叫做译码链 (原文中称作 decoding chain). 假设一条码链上所有元素的坐标为 $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$, 且所有元素的坐标满足如下条件:

$$(y_2 - y_1)\%n/(x_2 - x_1)\%n = (y_3 - y_2)\%n/(x_3 - x_2)\%n = \cdots = (y_m - y_{m-1})\%n/(x_m - x_{m-1})\%n, \quad (1)$$

则该条码链有斜率, 且将该码链斜率定义为 $s = (y_m - y_1)\%n/(x_m - x_1)\%n$. 显然, 如果一个元素集不满足等式 (1) 时则不能算作是二维坐标系上的一条直线, 更不能称为码链. 而对于所有能满足等式 (1) 的元素集均可以看作是一条直线或码链. 如图 1 所示, 在该坐标系上有明显标记的元素构成的集合就可以作为一条码链, 且该码链的斜率为 2 或 -3.

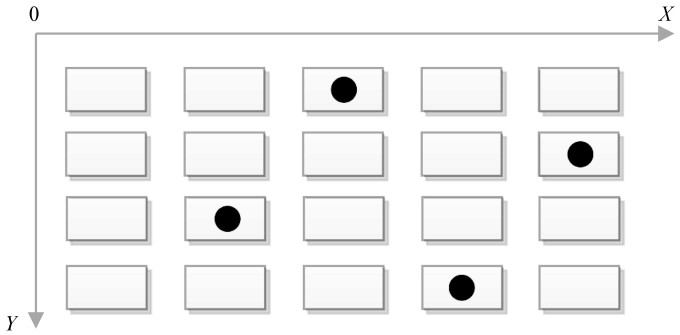

图 1　存储阵列坐标体系

Figure 1　The coordinate system of the storage array

有效码链: 当一条码链对应的校验元素及所有在该码链上的数据元素中, 最多有一个元素为未知元素 (处于失效节点上) 时, 该未知元素可以由这条码链上的其余元素进行有效恢复, 称该码链为当前未知元素的有效码链, 否则称为无效码链. 需要注意的是, 在一个阵列码系统中, 某些无效码链可以在一些未知元素得到恢复后变为有效码链.

$El(i,j)$: 当 i 和 j 均为一个确定正整数时, 可以使用符号 $El(i,j)$ 唯一确定存储系统阵列中的第 i 行第 j 列的元素. 当表示多个元素时, 可以使用符号 ':' 表示出一个确定的范围, 如 $El(i,1:4)$ 表示阵列中第 i 行上第 1 到 4 个的元素集合, $El(i,:)$ 表示阵列中第 i 行上的所有元素.

$Chain^s_{El(i,j)}$: 通过元素 $El(i,j)$ 且斜率为 s 的码链, 当码链通过的元素不直接指明时, 下标也可以是一个元素的集合. 例如 $Chain_{El(:,j)}$ 表示与列 j 相交的所有斜率的码链集合. $Chain^s$ 则泛指斜率为 s 的码链, 注意, s 可以是一个确定的数字也可以是一个集合, 例如 $Chain^{1\leqslant s\leqslant 5}$ 表示斜率大于 1 小于 5 的所有码链的集合. 至于 s 表示的是一个数字还是一个集合很容易根据上下文环境进行判断.

列间距离: 使用符号 D^i_j 表示列 i 到列 j 的距离, 定义 $D^i_j=(j-i-1)\%n+1$, 其中 n 为阵列的长度. 如果是计算错误列之间的距离则特别需要说明的是, 如果错误列 i 和错误列 j 之间还存在一个及以上错误列, 则错误列 i 和 j 之间不存在距离, 记作 $D^i_j=0$. 假设一个阵列中有 f 个错误, 记作错误列 $1,2,\ldots,f$, 错误列 $(i-2)\%f+1$ 是错误列 i 的左邻错误列, 错误列 $i\%f+1$ 称作错误列 i 的右邻错误列.

R: 表明某个元素可以用到某条码链 (或某个码链集) 进行正确恢复. 例如 $R(El(i,j))=Chain^s_{El(i,j)}$ 表示元素 $El(i,j)$ 可以通过自身且斜率为 s 的码链正确恢复, 也可以简记为 $R(El(i,j))=Chain^s$. 同时列出多个这类表达式, 则可以说明阵列中某些错误的恢复过程. 但是特别需要说明的是, 当有多个这类表达式同时出现时, 应注意各个表达式的先后顺序, 以保证其表达的错误恢复过程的正确性.

以上定义了文中后续使用频率较高或较为重要的概念或符号. 若无特别说明, 本文所有章节使用到的上述概念和符号均以本节的表述为准.

3　斜率码: 一类具有高容错能力的阵列纠删码

为了保证存储系统的整体运行效率, 提高编译码的计算速度, 除了所有运算均采用二进制的异或运算外, 本文所研究的斜率码还将采用几何形式的构造策略来进行设计 (其他的阵列码构建形式, 如图论、代数几何方法等, 运算代价较大, 因此用于大规模存储系统将受到很多限制). 目前已有的采用

相同技术路线的阵列码均是使用水平、正斜向和反斜向三类码链来组织数据元素和校验元素之间的关系, 从理论上讲这种策略的最大容错能力不超过 3, 更为严峻的是, 为了适应大规模存储系统, 水平码链显然不再适用, 否则单个节点上一个最小单位数据更新时将涉及存储系统上的所有其他数据节点的读取操作. 因此, 为了扩大容错能力, 本文采用了在满足存储阵列行列关系约束条件下从正负两个方向不断递增码链斜率的方法.

3.1 编码方法

假设存储系统需要最大的容错数目为 f, 阵列的行数为 m, 则数据阵列部分的列数 n 需要满足如下不等式: $n \geqslant m \cdot f - f + 1$, 其中 m 和 f 均为正整数, 且 $m \geqslant 2$. 校验阵列部分的列数为 $t = \lceil f \cdot n/m \rceil$. 显然, 这是典型的水平阵列码的数据布局结构, 阵列中的前 n 列完全存储的是数据信息, 称为数据阵列部分, 而后面 t 列则全部存储的是校验信息, 称为校验阵列部分或冗余阵列部分. 整个存储阵列的规模为 $m \times (n+t)$, 显然 t 为正整数. 若无特别说明, 下文在叙述过程中的符号 f, m, n, t 具有和本段描述相同的含义.

在 $m \times n$ 的数据阵列部分以列优先顺序部署 f 条码链: 当 f 为奇数时, 部署斜率为 $\pm 1, \pm 2, \ldots, \pm\lfloor f/2 \rfloor$, 以及斜率为 $\lceil f/2 \rceil$ 的码链各一条. 当 f 为偶数时, 部署斜率为 $\pm 1, \pm 2, \ldots, \pm[f/2]$ 的码链各一条. 各条码链上所有元素异或产生的校验值以列优先顺序存放于校验阵列中的各个位置上. 根据如上描述, $El(:, 1:n)$ 为存储阵列中的数据阵列部分, 即数据阵列部分的每个元素可以表示为 $El(i, j)$, $1 \leqslant i \leqslant m$, $1 \leqslant j \leqslant n$. 而 $El(:, n+1:n+t)$ 为存储阵列中的校验阵列部分, 即校验阵列部分的每个元素可以表示为 $El(i, j)$, $1 \leqslant i \leqslant m$, $n+1 \leqslant j \leqslant n+t$. 任意校验元素的生成可以表述为如下形式:

$$El_{\text{check}} = \sum_{\text{row}=1}^{m} El(\text{row}, ((\text{row}-1) \cdot \text{sign} \cdot \text{slope} + 1 + \text{beta})\%n), \tag{2}$$

其中, 符号 El_{check} 表示任意一个校验元素, 符号 sign 表示当前码链的斜率为正或负, 符号 slope 表示当前码链斜率的绝对值, 而符号 beta 则表示当前码链的当前数据元素所在列相对于上一个码链元素的列增量数值. 而表达式中的其他符号变量定义如下:

$$\text{slope} = \lceil i/2 \rceil, \quad \text{beta} = j - 1, \quad \text{sign} = \begin{cases} -1, & i\%2 = 0, \\ 1, & i\%2 = 1, \end{cases}$$

其中 i 表示当前部署的是容 i 错的码链, 符号 j 表示用于容 i 错的第 j 条码链, $1 \leqslant i \leqslant f, 1 \leqslant j \leqslant n$.

由斜率码编码方法的前提条件可知, 有 $n \geqslant m \cdot f - f + 1$ 成立, 而容错能力每增加 1 个则需要多部署 n 条码链, 即多增加 n 个校验元素, 显然在斜率码容错系统中校验元素的总数至少不小于 $f \cdot (f \cdot m - f + 1)$ 个, 则对于一个 m 行且最大容错能力为 f 的存储阵列, 其编码的时间复杂度为 $O(m \cdot n \cdot f)$, 其中 $n \geqslant m \cdot f - f + 1$.

以一个 4 容错的存储阵列为例, 同时假设存储阵列的行数为 3, 而根据前文可得如下结论, 需要部署斜率分别为 ± 1 和 ± 2 的码链各 1 条, 一共 4 条码链. 存储阵列中的数据阵列部分至少为 9 列, 而校验位共 36 个, 即校验阵列部分需要 12 列. 根据上述编码流程, 图 2 展示了所有校验位的生成过程. 需要说明的是, 在图 2 中所有具有相同纹理的数据及校验元素的二进制异或和为 0.

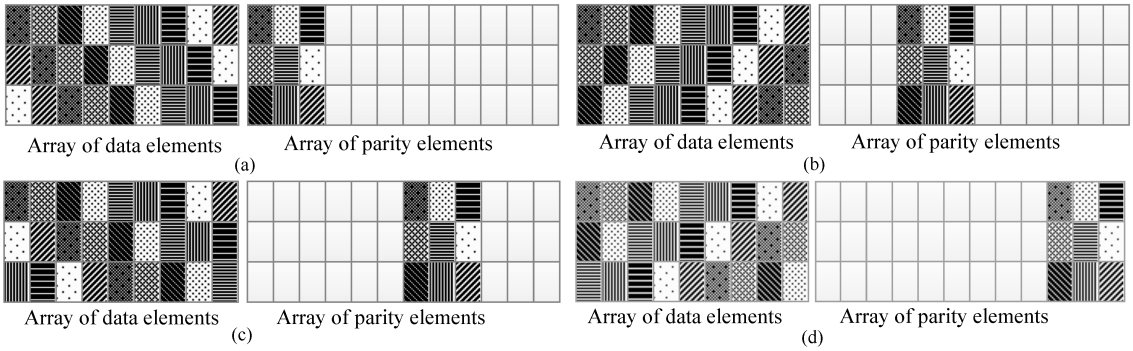

图 2　使用斜率 ±1,±2 的码链生成校验位

Figure 2　Generating parity elements by coding chains with slope ±1 and ±2. (a) The deployment of coding chain with a slope of 1; (b) the deployment of coding chain with a slope of −1; (c) the deployment of coding chain with a slope of 2; (d) the deployment of coding chain with a slope of −2

3.2　删除错误的数据恢复方法

斜率码是一类可容任意错误的阵列纠删码, 其容错能力在下文中将有专门的章节予以证明, 本节仅针对错误全部发生在数据阵列部分而给出丢失数据的恢复方法, 后续部分给出的容错能力证明将保障数据恢复方法的可行性. 说明一下, 本文所指的错误均指节点错误, 即整个存储节点 (服务器、整个磁盘等) 失效而导致的数据完全丢失, 不单独考虑存在存储节点中部分数据丢失的情况 (如磁盘的某些扇区错误等).

在一个存储阵列中的节点错误可以按照错误节点的类型分为 3 类情况, 包括错误全部发生在校验阵列部分, 错误全部发生在数据阵列部分以及校验阵列和数据阵列部分均有错误, 而错误全部发生在数据阵列部分是数据恢复最困难的情况. 在斜率码的编码体系下, 可以采用多种已知的阵列纠删码的译码方法对节点错误上的数据进行恢复, 而为了保证在大规模的存储环境中能高效地进行数据恢复, 本节将介绍一种针对错误全部发生在数据阵列部分的数据恢复方法, 该方法最核心的思想是寻找有效码链来进行错误节点上丢失元素的恢复, 类似于 Star 码中的 Zig-Zag 译码方法 [8], 但是由于斜率码体系本身的特性以及采用了更加简洁的有效码链寻找方法, 因而具有更高的执行效率.

假设在存储阵列的数据阵列部分存在 f 个删除错误, 即数据阵列部分上有 f 个列的所有元素未知, 顺序标记 f 个错误列为 $1,2,\ldots,f$. 不妨假设列 f 和列 1 之间的列间距离 D_1^f 为这 f 个错误列中相邻错误列的列间距离的最大值. 算法的基本思想如下: 码链 $Chain^s_{El(i,j)}$ 为当前删除元素的有效码链, 即元素 $El(i,j)$ 可以被斜率绝对值为 1 的码链恢复, 则未知元素 $El(i,f+1-j)$, $El(m+1-i,j)$ 和 $El(m+1-i,f+1-j)$ 也可以在当前被恢复, 即以下码链 $Chain^{-s}_{El(m+1-i,j)}$, $Chain^{-s}_{El(i,f+1-j)}$ 和 $Chain^{s}_{El(m+1-i,f+1-j)}$ 均为有效码链, 其中 $1 \leqslant i \leqslant \lfloor f/2 \rfloor$, $1 \leqslant j \leqslant \lfloor m/2 \rfloor$. 保证上述策略正确性的相关性质及证明将在后文中给出.

而由该数据恢复流程可知, 当错误全部发生在数据阵列部分时, 其数据恢复的时间复杂度可以控制在 $O(m \cdot \lceil f/2 \rceil \cdot \lceil m/2 \rceil)$ 之内.

仍然以一个 4 容错的存储阵列为例, 并且存储阵列的行数为 3, 如前所述, 存储阵列中的数据阵列部分至少有 9 列, 校验阵列部分需要 12 列. 假设数据阵列中的第 3, 4, 6 列发生了严重错误, 其中所有的数据全部丢失. 则按照前文关于数据恢复方法的描述, 所有丢失数据的恢复过程如下: 首先, 未知元素 $El(1,3)$ 可以由码链 $Chain^{-1}$ 进行恢复, 因此可以得知未知元素 $El(3,3)$, $El(1,6)$ 和 $El(3,6)$ 可

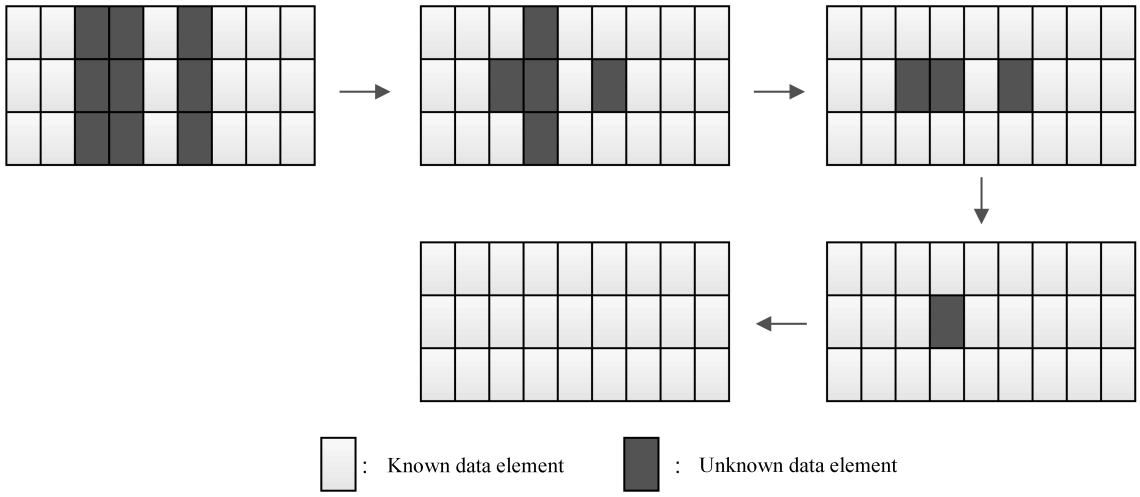

图 3 数据恢复示意

Figure 3 Data recovery process of 3 erasures

以分别被码链 $Chain^1$, $Chain^1$ 和 $Chain^{-1}$ 恢复. 此时未知元素 $El(1,4)$ 可被码链 $Chain^{-2}$ 恢复, 由此可得未知元素 $El(3,4)$ 可以被码链 $Chain^2$ 有效恢复. 接下来是未知元素 $El(2,3)$ 被码链 $Chain^{-1}$ 恢复, 因此可得未知元素 $El(2,5)$ 可以被码链 $Chain^1$ 恢复. 最后仅剩一个未知元素 $El(3,4)$ 可被任意一条穿过该元素的码链恢复. 至此, 所有丢失元素全部被恢复. 其详细过程如图 3 所示, 由于和整个数据恢复过程关联度不大, 校验阵列部分没有绘出.

3.3 性质与定理

由于独特的编码方法, 使得整个斜率码体系具有一系列独特的性质和定理, 这些性质和定理不仅保障了斜率码在理论上任意的容错能力, 也保证了该编码体系在大规模存储系统的容错领域上的其他优良特性和实用基础.

性质 1 斜率码中码链部署的对称性: 由斜率码的定义可知, 斜率码可容许一个存储阵列中最多 f 个删除错误, 当 f 为偶数时, 存储阵列中将部署斜率为 $1,2,\ldots,[f/2]$ 以及 $-[f/2],-[f/2]+1,\ldots,-1$ 的码链各一条. 当 f 为奇数时, 除了一条斜率为 $\lceil f/2\rceil$ 的码链外, 存储阵列中仍将部署斜率为 $1,2,\ldots,[f/2]$ 以及 $-[f/2],-[f/2]+1,\ldots,-1$ 的码链各一条. 即数据阵列部分中的每个元素均有斜率不同的 f 条码链通过, 且斜率为对称的 $\pm1,\pm2,\ldots,\pm\lfloor f/2\rfloor$, 换句话说当一个数据元素由一条斜率为 i 的码链通过时, 就一定有一条斜率为 $-i$ 的码链通过, 其中 $i\in1,2,\ldots,\lfloor f/2\rfloor$.

由上述性质, 可以容易得出如下结论: 在一个容错能力为 f 的存储阵列中, 如果在数据阵列部分有斜率为 s 的码链 $Chain^s$ 通过元素 $El(i,j)$, 则元素 $El(i,f+1-j)$ 和 $El(m+1-i,j)$ 上一定存在码链 $Chain^{-s}$ 通过, 元素 $El(m+1-i,f+1-j)$ 上一定存在码链 $Chain^s$ 通过, 其中 $1\leqslant i\leqslant\lfloor f/2\rfloor,1\leqslant j\leqslant\lfloor m/2\rfloor$. 利用斜率码的对称性以及相关结论能极大减少搜寻有效码链的次数, 降低数据恢复过程的复杂度, 如本章上一节中给出的数据恢复方法就有效利用了斜率码的该特性, 与 Star 码中的 Zig-Zag 译码方法相比, 将有效码链的搜索次数减少了近 3/4.

定理1 一个 f 容错的斜率码的体系中, 阵列的规模为 $m\times(n+t)$, 假设错误全部发生在数据阵列部分, 则一定存在至少两个错误列 i 和 j, 满足 $D_j^i\geqslant m$. 其中列 j 为列 i 的右邻错误列, $1\leqslant i,j\leqslant n$

且 $i \neq j$.

证明 假设错误全部发生在数据阵列部分, 如果需要平均每个错误列的距离达到 $m-1$ 时, 数据阵列部分的列数需要满足不等式 $n \geqslant (m-1)\cdot f$, 即 $n \geqslant m\cdot f-f$. 由斜率码的定义可知 $n \geqslant m\cdot f-f+1$ 成立, 根据鸽笼原理可得, 当错误全部发生在数据阵列部分时, 一定存在两个相邻错误列之间的距离不小于 m. t 为正整数, 因此 $n+t>n$ 成立. 所以, 至少存在两个错误列 i 和 j, 其距离不小于 m, 即 $D_j^i \geqslant m$ 成立. 证毕.

由该定理, 对于有 f 个错误全部发生在数据阵列部分的存储阵列中, 可能存在多个不同的错误列对 i 和 j 之间的列距离满足条件 $D_j^i \geqslant m$ (列 j 为列 i 的右邻错误列, $1 \leqslant i,j \leqslant n$, 且 $i \neq j$), 而通常对数据恢复的起始点选择起到关键作用的是列间距最大的两个错误列.

定理2 假设删除错误全部发生在数据阵列部分, 且存在两个错误列 i, j 满足不等式 $D_j^i \geqslant m$ 时, 若 $D_i^{(i-2)\%f+1}=d$, 则 $R(El(1:d,i))=Chain^1$ 成立, 即错误列 i 的前 d 个元素及后 d 个元素可以被斜率为 1 的码链恢复, 其中 d 为正整数, 且 $1 \leqslant d \leqslant m$.

证明 当 $d=1$ 时, 由条件 $D_j^i \geqslant m$ 可得 $R(El(1,i))=Chain^1$ 成立, 而当 $d>1$ 时, 错误列 i 与其左邻的错误列之间存在 $d-1$ 个正确列, 即通过 i 列第 j 个元素且斜率为 1 的码链的前 $j-1$ 个元素为已知元素, 而由条件 $D_j^i \geqslant m$ 可得, 该码链的后 $m-j$ 个元素为已知元素, 因此等式 $R(El(1:d,i))=Chain^1$ 成立. 证毕.

而由斜率码的码链对称性可知, 如下结论仍然成立: 若 $D_{j\%f+1}^j=d$, 则 $R(El(m:m-d+1,i))=Chain^{-1}$ 成立, 即错误列 j 的前 d 个元素及后 d 个元素可以被斜率为 -1 的码链恢复. 该结论也可以被看作是定理 2 的另外一种表述形式.

定理3 假设 f 个删除错误全部发生在数据阵列部分, 且当 $D_j^i \geqslant m$ 成立时, 若 $D_i^{(i-2)\%f+1} \geqslant \lceil m/2 \rceil$ 则 $R(El(:,i))=Chain^s$ 成立 ($s=\pm 1$), 即错误列 i 的所有元素可以由斜率为 1 或 -1 的码链恢复.

证明 因为 $D_i^{(i-2)\%f+1} \geqslant \lceil m/2 \rceil$ 成立, 即错误列 i 与其左邻错误列的距离大于等于 $\lceil m/2 \rceil$, 由定理 2 可得等式 $R(El(1:\lceil m/2 \rceil,i))=Chain^1$ 与 $R(El(1:m-\lceil m/2 \rceil+1,i))=Chain^{-1}$ 成立, 因此可得等式 $R(El(1:m,i))=Chain^1$ 成立, 即 $R(El(:,i))=Chain^s$ 成立. 证毕.

由斜率码的码链对称性可知, 如下结论也成立: 若 $D_{j\%f+1}^j \geqslant \lceil m/2 \rceil$ 时, 则 $R(El(:,j))=Chain^s$ 成立 ($s=\pm 1$), 即错误列 j 的所有元素可以被斜率为 1 或 -1 的码链恢复. 该结论也可以被看作是定理 3 的另外一种表述形式.

定理4 假设 f 个删除错误全部发生在数据阵列部分, 标记 f 个错误列为 $1,2,\ldots,f$, 当 $1 \leqslant i \leqslant \lfloor f/2 \rfloor$ 时, 对错误列 i 使用码链集 $Chain^{-1 \geqslant s \geqslant i}$ 进行未知元素恢复, 当 $\lceil f/2 \rceil \leqslant i \leqslant f$ 时, 对错误列 i 使用码链集 $Chain^{i \geqslant s \geqslant 1}$ 进行未知元素恢复, 则每条码链的末端一定在错误列 f 到错误列 1 之间区域.

证明 根据斜率码的定义, 数据阵列部分的尺寸为 $m \times n$, 且数据阵列部分的尺寸满足如下约束条件: $n \geqslant m\cdot f-f+1$. 首先证明 $\lceil f/2 \rceil \leqslant i \leqslant f$ 的情况, 由于错误列 i 使用码链集 $Chain^{i \geqslant s \geqslant 1}$ 进行未知元素恢复, 而穿过错误列 1 长度最长的码链斜率为 i, 长度为 $(m-1)\times i$. 根据前文假设, $D_{(i-1)\%f+1}^i=d_i$, 且 $\max(d_1,d_2,\ldots,d_f)=d_f$. 由定理 1 可知, $d_f \geqslant m$, 错误列 1 到错误列 f 长度为 $m-d_f$. 而 i 最大为 $\lceil f/2 \rceil$, 而 m 为大于等于 2 的正整数, 因此 $(m\cdot f-f+1)/2+m>(m-1)\cdot\lceil f/2 \rceil$, 可得 $n/2+m>(m-1)\cdot i$. 在此条件下定理成立. 而对于 $1 \leqslant i \leqslant \lfloor f/2 \rfloor$ 的情况, 由斜率码的码链对称性可知, 该条件下定理同样成立. 证毕.

定理5 假设存在 f 个删除错误列 $1,2,\ldots,f$, 使用相同的码链集 $Chain^s$ 进行数据恢复, 则有如

下结论成立: 当 $i<j<\lfloor f/2\rfloor$ 时, 错误列 i 可恢复的未知元素的数量 $\geqslant$ 错误列 j 可恢复的未知元素的数量. 当 $\lceil 2/f\rceil \leqslant i<j$ 时, 错误列 i 可恢复的未知元素的数量 $\leqslant$ 错误列 j 可恢复的未知元素的数量. 其中 $1\leqslant s\leqslant\lceil f/2\rceil$ 或者 $-1\geqslant s\geqslant-\lfloor f/2\rfloor$.

证明 首先证明定理的前半部分. 根据条件, 两个错误列 i, j 满足 $i<j<\lfloor f/2\rfloor$, 即列 i 更加靠近阵列的正确区域. 最简单的情况, 当 $i=1$ 且 $j=2$ 时, 即 $D_j^i=1$, 根据定理 2 可得 $R(El(1,i))=Chain^{-1}$, 而 $R(El(1,i))\neq Chain^{-1}$, 显然定理成立. 如果 $R(El(1,j))=Chain^s$ 成立, 而与列 i 左邻的正确元素更多, 显然存在有效码链, 则 $R(El(1,i))=Chain^z$ 成立, 其中 $\mid z\mid\leqslant\mid s\mid$. 由斜率码的码链对称性易得, 定理的后半部分也成立. 证毕.

定理6 全部 f 个删除错误均发生在数据阵列部分时, 标记前 k 个错误列为 $1,2,\ldots,k$, 列 k 和右邻错误列 $k+1$ 的距离 $\geqslant k+1$, 则 $R(El(1:2,1:k))=Chain^{-k\leqslant s\leqslant-1}$ 成立, 即使用码链集 $Chain^{-k\leqslant s\leqslant-1}$ 可以正确对前 k 列的前两行所有未知元素进行恢复, 其中 $2\leqslant k<\lfloor f/2\rfloor$.

证明 假设错误列 k 和 $k+1$ 的距离为 $k+1$, 即 $D_{k+1}^k=k+1$ 成立. 当 $k=2$ 时, 根据前提条件 $D_3^2=3$, 即列 2 右侧有 2 个正确列, 因此 $R(El(2,2))=Chain^{-2}$ 成立. 此时定理成立. 当 $k=3$, 根据前提条件有 $D_4^3=4$. 有如下等式成立: $R(El(1,1))=Chain^{-1}$, $R(El(1,2))=Chain^{-2}$, $R(El(2,1))=Chain^{-1}$, $R(El(1,3))=Chain^{-3}$, $R(El(2,2))=Chain^{-2}$, 而由于 $D_4^3=4$, 即列 3 右侧有 3 个正确列, 等式 $R(El(2,3))=Chain^{-3}$ 成立. 以此类推, 可以得出结论, 当 $D_{k+1}^k=k+1$, 等式 $R(El(2,k))=Chain^{-k}$ 成立, 而由前述未知元素恢复过程可知, 前 k 列的前两行所有错误未知元素可以被恢复. 证毕.

由斜率码的码链对称性易得如下结论成立: 全部 f 个删除错误均发生在数据阵列部分时, 标记后 k 个错误列为 $f,f-1,\ldots,f-k+1$, 列 $f-k+1$ 和左邻错误列 $f-k$ 的距离 $\geqslant k+1$, 则 $R(El(1:2,f-k+1))=Chain^{1\leqslant s\leqslant k}$ 成立, 即使用码链集 $Chain^{1\leqslant s\leqslant k}$ 可以对后 k 列的前两行所有未知元素进行有效恢复, 其中 $2\leqslant k<\lfloor f/2\rfloor$. 该结论也可以被看作是定理 6 的另外一种表述形式.

定理7 假设阵列中存在 f 个删除错误, 且错误均出现在数据阵列部分, 各个错误列标记为 $1,2,\ldots,f$, 则错误列 i 的第一个元素可由斜率不小于 $-i$ 的负斜率码链正确恢复. 相同条件下, 错误列 $f-i+1$ 的第一个元素可由斜率不大于 i 的正斜率码链正确恢复, 其中 $i\leqslant\lceil f/2\rceil$.

证明 首先证明定理的前半部分. 从最简单的情况入手, 第 1 个错误列的第 1 个元素显然可以被斜率为 -1 的码链恢复. 而第 2 个错误列的第 1 个元素可以由如下过程进行恢复: $R(El(1,1))=Chain^{-1}$, $R(El(1,2))=Chain^{-2}$. 特别地, 如果当 $D_2^1\geqslant 2$ 时, 错误列 2 的第 1 个元素可以由斜率为 -1 的码链恢复, 即 $R(El(1,2))=Chain^{-1}$ 成立. 以此类推, 假设错误列 i 与左邻错误列的距离不小于 i, $D_i^{(i-2)\%f+1}\geqslant i$, 则根据定理 5 可得, 错误列 $1,2,\ldots,i-1$ 的前两行所有未知元素可以被恢复, 即等式 $R(El(1:2,1:i-1))=Chain^{-i-1\leqslant s\leqslant-1}$ 成立. 而由定理 5 可得, 列序号越小的错误列可以被恢复出的未知元素越多, 因此 $R(El(1,i))=Chain^{-i}$ 成立, 即错误列 i 的第一个元素可以被恢复. 而 $D_i^{(i-2)\%f+1}<1$ 时, 不妨假设 $D_i^j=i$, 分两种情况, 一是列 i 为正确列, 则显然 $R(El(1,i))=Chain^{-i}$ 成立, 二是列 j 为错误列, 显然 $j=1,2,\ldots,i-2$, 则有 $R(El(1,j+1:i-1))=Chain^{-1\geqslant s>-i}$, $R(El(2,j))=Chain^{-j}$ 成立, 因此错误列 i 的第一个元素可以恢复. 由斜率码的码链对称性易得, 定理的后半部分也成立. 证毕.

定理8 f 个删除错误全部发生在数据阵列部分时, 若有 $R(El(1,:))=Chain$ 成立, 则 $R(El(2,:))=Chain$ 即阵列中第 1 行的所有错误元素可以被恢复, 则第 2 行的所有错误元素也可以被恢复.

证明 第 1 行的未知元素全部被恢复后, 仍然假设其余行的所有错误元素还没有被恢复 (虽然在

恢复第 1 行错误元素的过程中一定会有不少其余行的错误元素被恢复). 此时, 将第 i 行作为第 $i-1$ 行, 其中 $2 \leqslant i \leqslant m$, 重复最初恢复第 1 行所有未知元素的过程, 即可恢复第 2 行上所有未知元素. 证毕.

4 性能分析及对比

上一节给出了斜率码的一些重要性质, 这些性质主要阐述了斜率码体系中码链、编码和译码 (数据恢复) 之间的关系. 而本节将针对存储系统, 特别是大规模存储系统对容错编码的需求进一步分析斜率码体系的性能.

4.1 理论上不受限制的容错能力

容错能力一直是限制阵列纠删码实用性的主要因素, 而斜率码区别于其他基于二元域异或运算的阵列码之处是其容错能力在理论上的不受限制. 阵列码体系中, 删除错误的发生可分为如下 3 种情况: 一是错误全部发生在校验阵列部分, 二是校验阵列部分和数据阵列部分均有错误发生, 三是错误全部发生在数据阵列部分. 下面按照这 3 种错误情况分别进行讨论.

f 个删除错误全部发生在校验阵列部分是纠错, 即数据恢复最为简单的情况, 只需重新做一次校验位的生成运算即可. 校验阵列部分和数据阵列部分均有错误发生的情况也相对容易处理, 由斜率码的构造过程可知, 在 f 容错的斜率码阵列中, 每个数据元素均有 f 条码链通过, 而每条码链的校验值存放于一个校验元素中, 因此一个数据元素与 f 个校验元素相关. 由于在数据阵列部分部署码链时采用了行优先的顺序, 而将每条码链校验得到的值存放于校验位时采用了列优先, 又因为数据阵列部分的长度 n 严格大于其高度 m, 所以两个与同一数据元素相关的校验元素一定不会出现在同一列. 假设在校验阵列部分有 i 个错误, 则在数据阵列部分有 $f-i$ 个错误, 换句话说 $m\cdot(f-i)$ 个数据元素和 $m\cdot i$ 个校验元素失效, 即每个数据元素至多有 $f-1$ 个相关的校验元素失效. 因此, 数据阵列部分的全部错误可以被恢复, 而数据阵列部分的错误元素被恢复后, 就又转换成为了错误全部发生在校验阵列部分的情况, 所以剩余错误可以容易地被恢复. 下面讨论数据恢复最困难的情况, 即 f 个错误全部发生在数据阵列部分.

采用归纳法进行分析, 为了便于叙述, 不妨假设其后的所有论述将遵循如下条件: 将阵列中的 f 个错误列顺序地记作 $E_1, E_2, \ldots, E_f$, 将错误列 E_i 与其右邻错误列之间的距离记作 d_i, 即 $D^{E_i}_{E_{(i-1)\%f+1}} = d_i$, 则 $\max(d_1, d_2, \ldots, d_f) = d_f$. 其中 $i = 1, 2, \ldots, f$. 由于 $n \geqslant m\cdot f - f + 1$, 所以根据定理 1 可知, 不等式 $d_f \geqslant m$ 一定成立.

当 $f = 1$ 时, 标记该错误列为 E_1, 根据条件 $n \geqslant m\cdot f - f + 1$ 可得 $n \geqslant m$. 由斜率码的定义可知, 此时在数据阵列部分部署了一条斜率为 1 的码链, 因此错误列上每个错误元素均有一条斜率为 1 的码链通过, 且码链上唯一的未知元素来自该错误列, 即 $R(El(:, E_1)) = Chain^1$ 成立. 显然当 $f = 1$ 时, 错误可以顺利恢复.

当 $f = 2$ 时, 标记两个错误列为 E_1 和 E_2, 由条件 $n \geqslant m\cdot f - f + 1$ 可得 $n \geqslant 2m - 1$. 又根据前提条件可知 $d_2 \geqslant m$. 由列间距离的定义可知所有距离均为正整数, 根据定理 2 可得 $R(El(1, E_1)) = Chain^{-1}$, $R(El(1, E_2)) = Chain^1$ 成立. 因此, 阵列中第一行所有的错误元素被恢复, 即 $R(El(1, :)) = Chain^{\pm 1}$ 成立. 由定理 8 可知, 所有错误均可成功恢复.

假设 $f = 2k$ 时, 所有错误列上未知元素可以被恢复. 下面证明 $f = 2k+1$ 和 $f = 2k+2$ 的情

况. 当 $f=2k+1$ 时, 标记所有错误列为 $E_1,E_2,\ldots,E_{2k},E_{2k+1}$. 由前提条件可知 $d_{2k+1}\geqslant m$ 为最大距离. 当 $d_1\geqslant\lceil m/2\rceil$ 成立时, 根据定理 3 可得 $R(El(:,E_1))=Chain^{\pm1}$ 成立. 同理, 当 $d_{2k}\geqslant\lceil m/2\rceil$ 成立时, 有 $R(El(:,E_{2k+1}))=Chain^{\pm1}$ 成立. 上述两种情况均可以转换为 $f=2k$ 的情况, 根据假设, 全部错误数据能顺利恢复. 而当 $d_1<\lceil m/2\rceil$ 且 $d_{2k}<\lceil m/2\rceil$ 时, 由定理 7 可知如下等式成立: $R(El(1,E_1:E_k))=Chain^{-1\geqslant s\geqslant -k}$, $R(El(1,E_{k+2}:E_{2k}))=Chain^{1\leqslant s\leqslant k}$, $R(El(1,E_{k+1}))=Chain^{k+1}$. 此时所有存在于第一行的未知元素被恢复, 根据定理 8 可知, 所有错误均可正确恢复.

当 $f=2k+2$ 时, 标记所有错误列为 $E_1,E_2,\ldots,E_{2k+1},E_{2k+1}$. 由前提条件可知 $D_{2k+2}\geqslant m$ 为最大距离. 当 $d_1\geqslant\lceil m/2\rceil$ 成立时, 根据定理 3 可得 $R(El(:,E_1))=Chain^{\pm1}$ 成立. 同理, 当 $d_{2k+1}\geqslant\lceil m/2\rceil$ 成立时, 有 $R(El(:,E_{2k+1}))=Chain^{\pm1}$ 成立. 上述两种情况均可以转换为 $f=2k+1$ 的情况, 根据前述证明, 显然全部错误数据能顺利恢复. 而当 $d_1<\lceil m/2\rceil$ 且 $d_{2k+1}<\lceil m/2\rceil$ 时, 由定理 7 可知如下等式成立:

$$R(El(1,E_1:E_k))=Chain^{-1\geqslant s\geqslant -k},\quad R(El(1,E_{k+3}:E_2k))=Chain^{1\leqslant s\leqslant k},$$

$$R(El(1,E_{k+2}))=Chain^{k+1},\quad R(El(1,E_{k+1}))=Chain^{-k-1}.$$

此时所有存在于第一行的删除错误元素被恢复, 根据定理 8 可知, 所有错误列上未知元素均可正确恢复.

4.2 弱约束条件

阵列纠删码在构造过程中对于阵列尺寸的限制, 即对条带或条块的约束则是另外一个影响其实用性的因素. 表 1 列出了存储容错领域当前的主要几类阵列码的容错能力, 条带及条块尺寸的要求对比情况. 从该表中不难发现, 绝大部分阵列纠删码对于存储阵列的列数 (条带尺寸、存储阵列中的节点数) 和单一条块中元素数目 (条块的尺寸, 存储阵列的行数) 都有非常严格的要求, 且通常都要求存储阵列的列数为素数, 而这一约束条件对存储阵列造成了很强的限制, 从而导致了这些编码的可扩展性差和实用性的降低. 当然, 目前也有研究者们提出了一些折中的解决方案, 最典型的方法为当存储阵列中的节点数目不为素数时则添加 1 个或多个所有存储数据值均为 0 的虚拟节点以补足成素数, 当编码或数据恢复时这些虚拟节点共同参与运算. 但是, 随着存储规模的扩大, 存储节点数据的增加, 相邻素数之间的间隔也随之扩大, 即为了补足素数所需的虚拟节点数目将急剧增加, 与之相伴随的则是编码译码的无效运算量也不断增加. 因此, 此类方案仅仅是在小规模存储系统中适用. Weaver 编码的容错能力较高, 且对于条带和条块尺寸没有任何特殊需求, 但是该码没有系统的编码方法并且缺乏理论支撑, 在某种特定条件下构造出的码字具有的容错能力依赖计算机工程试验, 因而很难在大规模存储系统上进行扩展. Grid 码具有更强的容错能力, 实现简单且存储效率最高可达 80% 以上, 但是该码在构造过程中需要找到其他阵列码 (必须是典型的水平码或垂直码) 形成对偶码, Grid 码的容错能力依赖于我们确定的对偶码的容错能力, 此外 Grid 码并不是典型的水平阵列码或垂直阵列码, 它的存储布局兼具了水平和垂直两种阵列编码的特点, 因此, Grid 码的扩展和应用于大规模存储系统时都将受到很大的制约.

相比其他阵列码, 本文所提出的斜率码对于条带或条块的尺寸也没有任何素数的限制, 对于一个容错能力为 f 的斜率码仅需要条带在存储阵列部分的长度 n 和条块尺寸 m 满足如下条件: $n\geqslant m\cdot f-f+1$, 其中 f 显然为正整数, m 为存储阵列的行数, 即条块的尺寸, n 为数据阵列部分的列数, 而 $n+\lceil(f\times n)/m\rceil$ 即为条带的尺寸. 即在斜率码体系中, 条带和条块尺寸仅存在很弱的限制条件 (阵

表 1　阵列码的容错能力、条带及条块尺寸要求对比

Table 1　Fault tolerance, stripe and strip constraints of array codes

Array codes	Fault-tolerant ability	Requirement for the stripe size	Requirement for the strip size
EVENODD codes	2	Prime	The stripe size−1
RDP codes	2	Prime−1	Equal to the stripe size
Liberation codes	2	None	A prime not less than the strip size
Star codes	3	Prime	The stripe size−1
X codes	2	Prime	Equal to the stripe size
B codes	2	Solution for corresponding mathematics problem	The stripe size×2, or the stripe size×2+1
P codes	2	Prime, or prime−1	The stripe size/2
Weaver codes	12	None	None
Grid codes	15	Determined by the stripe size of matched codes	Determined by the stripe size of matched codes
Slope codes	Arbitrary	Greater than ((the stripe size× fault−tolerant number)−fault−tolerant number)	A positive integer not less than 2

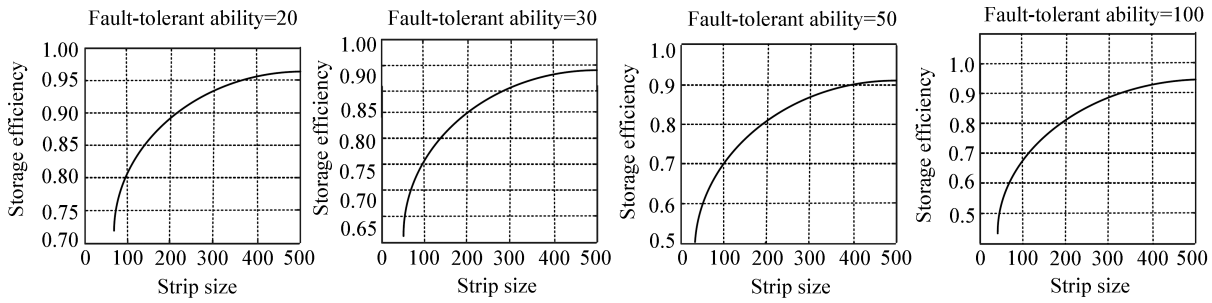

图 4　条块尺寸对存储效率的影响

Figure 4　Effect of strip size to storage efficiency

列的行列约束). 因此, 斜率码可以具有良好的扩展能力, 且适用于存储系统的实用性较其他阵列码有很大的增强.

4.3　存储效率及提升方法

基于复制备份策略的存储系统可靠性增强方案最大的问题便是其极低的存储效率, 特别是当存储规模巨大时这种对存储空间的浪费将变得不可接受. 而以编码策略为核心实现的容错体系则可以有效提高相同容错能力条件下的存储效率, 虽然需要付出一定的计算量为代价. 存储效率是一个衡量容错编码的重要性能指标, 若在相同或相当的容错能力前提下, 经过编码后的存储效率与复制冗余策相比近似甚至更低, 则该编码方案至少从 “计算代价 — 存储效率” 的角度看是失败的.

本文提出的斜率码并不具备 MDS 性质, 不能达到香农限, 即理论最优的存储效率. 尽管如此, 由斜率码的构造过程可知, 在确定一个固定容错能力的前提下, 斜率码可以通过加大条块尺寸的方法来提高其存储效率. 图 4 显示了在容错能力一定的前提下条块尺寸和存储效率的关系, 即存储效率可以随着条块尺寸的增加而不断增长.

表 2 斜率码与 RS 码的对比

Table 2 Slope codes vs. Reed-Solomon codes

	Fault tolerance	Storage efficiency	Operation
Slope codes	Arbitrary	Non-optimal	XOR
Reed-Solomon codes	Arbitrary	Optimal	Finite field arithmetic

4.4 低运算工作量

任意一个纠删码在存储系统, 特别是大规模存储系统中的实用性很大程度取决于其在编码和数据重构阶段的运算开销情况, 而编码及数据重构的计算工作量则成为了评价纠删码的一大重要指标. 其中一个最重要的例子就是 RS 码, 该码具有理论上不受限制的容错能力, 且能在存储效率上达到香农限的最优值. 而长期以来制约 RS 码应用于存储系统的关键正是在于其编译码使用了效率较低的有限域运算, 一旦运算效率的问题得以解决, RS 码将无疑是存储系统中容错核心方案的首选之一, 所以长期以来很多研究者们一直致力于提高 RS 编码的计算效率, 也取得了不小的进展, 且随着计算机系统 CPU 计算能力的不断增强, RS 编码的计算效率瓶颈将大大降低. 尽管如此, 根据文献 [22] 可知, 目前通常有限域上乘法的带宽仍然只能达到约 150 M/s (假设计算有限域为 $GF(2^{16})$), 远远小于异或运算的 3 GB/s, 因此 RS 码的计算效率对于目前规模巨大且不断增加的存储系统来说依然很难接受. 相对而言, 基于异或的容错方法在计算效率方面的表现则更为理想, 表 2 列出了本文提出的斜率码与 RS 码之间的情况对比.

对于目前大多数非 MDS 的阵列码, 其构造通常不具有很强的规范性, 构造过程相对复杂, 所以我们很难使用一个确切的公式来表达其编码或数据重构的运算量, 因而通常使用生成单个校验位所需要经过异或运算的次数来衡量其编码的复杂度, 采用恢复单个丢失数据位需要经过的异或运算的次数来衡量其数据恢复的复杂度.

由斜率码的定义可知, 对于生成所有单个校验位所需的异或运算次数, 以及恢复单个丢失数据位所需要的异或运算次数均为确定的 $m-1$ 次, 其中 m 为存储阵列的行数, 即条块尺寸. 然而在斜率码体系下, 存储阵列的列数, 即条带尺寸会随着存储阵列的行数 m 或最大容错能力 f 的扩大而增长, 这就意味着校验位的数量也将增加, 即总体运算量的加大. 因此, 存储阵列的行数以及最大容错能力的变化对编码和数据恢复运算量的影响是我们所不能不考虑的一个因素. 图 5 则显示了条块尺寸为 2, 4, 8, 16 的斜率码, 随着最大容错能力的增强, 编码和数据恢复总体运算量增加的情况. 不难发现, 编码及数据恢复的运算量会随着条块尺寸以及最大容错能力的增长而增加, 但是其增长幅度近似于线性, 即在容错能力一定的条件下运算量的增长近似于随着条块尺寸的增加而线性增长, 或在条块尺寸一定的情况下运算量的增长近似于随着最大容错能力的增加而线性增长, 这种情况在实用系统中是能够被普遍接受的. 此外, 在容错数量较大时, 其总的运算量仍然被控制在一个极低的范围内, 如条带尺寸为 8 最大容错能力为 100 的情况, 生成得到所有校验位或恢复所有丢失的数据位所需要的二进制异或运算次数小于 4.7×10^5 次, 即使再加上存储系统运行中数据缓存、读取、传输、排队等环节的运算, 这显然也是当前一个拥有最普通运算能力的计算节点能够轻松完成的工作量.

4.5 最优更新代价

在水平阵列码体系中, 更新代价 (update penalty) 是指当信息节点上 1 个最小的信息位变化时所需要涉及的校验节点的数量. 由于在水平阵列码中, 每一个信息位的变化都会导致多个校验节点上的

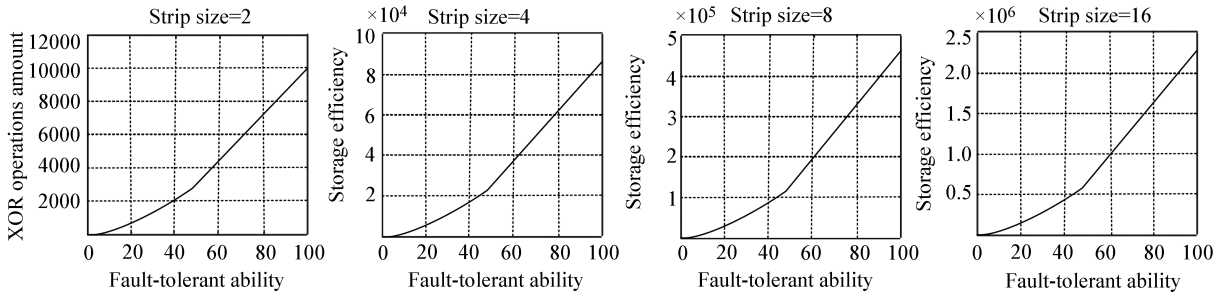

图 5　容错能力增加对运算量的影响

Figure 5　Effect of fault tolerance to calculations

数据的存取操作, 显然更新代价过高将极大的限制存储系统数据并行处理的速度, 因此有效降低更新代价将整体提升存储系统的运行效率. 由信息论可知 [22], 任意 f 容错的水平阵列码每一次信息更新操作至少会涉及到 f 个不同的校验节点, 即最优的更新代价为 f.

由斜率码的构造过程可知, 在一个最大容错能力为 f 的的斜率码存储阵列中, 任意一个数据元素有且仅有 f 条码链通过, 即每一个数据位与 f 个校验位存在关联. 由 4.1 小节可得, 这 f 个与相同数据元素相关联的校验元素一定处在不同的节点上. 因此, 每一次对数据元素的更新, 有且仅有 f 个校验节点会有存取操作的发生, 达到理论上的最优更新代价.

5　结束语

考虑到错误恢复时间, 节点间内部通信机制, 以及系统的整体运行效率等诸多因素, 对于一个存储系统的容错方案而言, 并非可容错数量越高其实用价值就越大. 而本文工作的出发点在于构建出一类容错能力理论上不受限制且容错数量能根据应用环境预先设定的阵列纠删码, 且能避免目前大多阵列码在构造时需要满足的强约束条件, 从而提升阵列码在存储系统中的实用性以及扩展阵列码的应用范围. 经实验表明, 本文提出的斜率码可以很好地达到这一目标. 此外, 斜率码的整个编码及数据恢复计算操作全部由二元域上的异或运算完成, 且采用了运算量最小的几何形式构造方法设计, 具有很高的运算效率. 而该纠删码所具备的最优更新代价可以优化冗余节点的读写访问量, 从访问效率的角度改善整个系统的运行性能. 虽然可以在固定容错能力条件下通过一定的技术手段对存储效率进行提高, 但由于该码不具备 MDS 性质, 因而存储效率不能达到理论最优, 这也是斜率码体系需要继续完善之处. 从目前已知的文献及参考资料上看, 本文所提出的斜率码是第一个完全使用二进制异或运算且理论上可容任意数量错误的阵列纠删码, 因此该码的提出对于提升阵列码的应用范围及实用价值可能会起到一定的积极作用.

致谢　特别感谢中国科学院成都计算机应用研究所王晓京研究员对本文工作的悉心指导和宝贵意见.

参考文献

1　Sathiamoorthy M, Asteris M, Papailiopoulos D, et al. Xoring elephants: novel erasure codes for big data. In: Proceedings of the VLDB Endowment. Almaden: VLDB Endowment, 2013. 325–336

2 Ghemawat S, Gobioff H, Leung S. The Google file system. In: Proceedings of 9th ACM Symposium on Operating Systems Principles Proceedings, New York, 2003. 29–43

3 Shvachko K, Kuang H, Radia S, et al. The Hadoop distributed file system. In: Proceedings of IEEE Twenty-Sixth Symposium on Massive Storage Systems and Technologies Proceedings, Washington, 2010. 1–10

4 DeCandia G, Hastorun D, Jampani M, et al. Dynamo: Amazon's highly available key-value store. In: Proceedings of Twenty-First ACM SIGOPS Symposium on Operating Systems Principles Proceedings, New York, 2007. 205–220

5 Blaum M, Brady J, Bruck J, et al. EVENODD: an efficient scheme for tolerating double disk failures in RAID architectures. IEEE Trans Comput, 1995, 45: 192–202

6 Blaum M, Brady J, Bruck J, et al. The EVENODD code and its generalization. High Perform Mass Stor Parall I/O, 2001, 34: 187–208

7 Xu L, Bruck J. X-code: MDS array codes with optimal encoding. IEEE Trans Inform Theory, 1999, 45: 272–276

8 Huang C, Xu L. STAR: an efficient coding scheme for correcting triple storage node failures. IEEE Trans Comput, 2008, 57: 889–901

9 Plank J, Mario Blaum. Sector-disk (sd) erasure codes for mixed failure modes in Raid systems. Trans Stor, 2014, 10: 4–17

10 Li M, Shu J. On the equivalence between the B-Code constructions and perfect one-factorizations. In: Proceedings of the 2010 IEEE International Symposium on Information Theory, Austin, 2010. 993–996

11 Meng Q C. Research of erasure codes applied in distributed storage system. Dissertation for Ph.D. Degree. Beijing: Graduate University of the Chinese Academy of Sciences, 2007 [孟庆春. 应用于分布式存储系统上的纠删码技术研究. 博士学位论文. 北京: 中国科学院研究生院, 2007]

12 Chen Z. A class of array erasure codes and their applications. Dissertation for Ph.D. Degree. Beijing: Graduate University of the Chinese Academy of Sciences, 2009 [陈峥. 一类新的阵列纠删码理论及应用研究. 博士学位论文. 北京: 中国科学院研究生院, 2009]

13 Li M Q, Shu J W, Zheng W M. Grid codes: strip-based erasure codes with high fault tolerance for storage systems. Acm Trans Stor, 2009, 4: 15–22

14 Zheng Z, Sinha P. XBC: XOR-based buffer coding for reliable transmissions over wireless networks. In: Proceedings of Broadband Communications, Networks and Systems, Raleigh, 2007. 76–85

15 Plank J S, Xu L. Optimizing Cauchy Reed-Solomon codes for fault-tolerant network storage applications. In: Proceedings of Network Computing and Applications, Massachusetts, 2006. 173–180

16 Plank J S, Greenan K M, Miller L. Screaming fast Galois field arithmetic using Intel SIMD instructions. In: Proceedings of 11th USENIX Conference on File and Storage Technologies (FAST 13), San Jose, 2013. 298–306

17 Calder B, Wang J, Ogus A, et al. Windows azure storage: a highly available cloud storage service with strong consistency. In: Proceedings of Twenty-Third ACM Symposium on Operating Systems Principles (SOSP 2011), New York, 2011. 143–157

18 Gallager R. Low-density parity-check codes. IEEE Trans Inform Theory, 1962, 8: 21–28

19 Harihara G, Janakiram B, Chandra M G, et al. SpreadStore: a LDPC erasure code scheme for distributed storage system. In: Proceedings of the 2010 International Conference on Data Storage and Data Engineering, Bangalore, 2010. 87–93

20 Hafner J L. Weaver codes: highly fault tolerant erasure codes for storage systems. In: Proceedings of Usenix Conference on File and Storage Technologies, San Francisco, 2005. 16–16

21 Hafner J L. Hover erasure codes for disk arrays. In: Proceedings of International Conference on Dependable Systems and Networks, Philadelphia, 2006. 217–226

22 Plank J S, Buchsbaum A L, Collins R L, et al. Small parity-check erasure codes-exploration and observations. In: Proceedings of International Conference on Dependable Systems and Networks, Yokohama, 2005. 326–335

A class of array erasure codes with high fault tolerance

Dan TANG & Hongping SHU*

Chengdu University of Information Technology, Chengdu 610225*, China*
*E-mail: shp@cuit.edu.cn

Abstract Array-erasure coding technology is not only one of the ideal methods to enhance the reliability of storage systems, but it also has important applications in many other research areas, such as secret-sharing schemes and multipath transportation. This is because it has many special advantages, such as a simple construction, a high calculation efficiency, etc. However, low fault tolerance is the biggest obstacle to the actual implementation of array codes. Grid codes have the highest fault tolerance of all currently known array codes, which only achieve a capacity of 15. In view of this situation, this paper presents a class of array erasure codes with high fault tolerance, which are referred to as slope codes. Slope codes have a high computing efficiency because all calculations of encoding and data reconstruction are binary XOR operations, and that implementation in hardware or software is convenient due to its simple construction method. The new codes have a non-MDS property, but its storage efficiency can be close to the Shannon Limit with the increase of the size of strips in a storage array. Additionally, the minimum update penalty provides a guarantee of efficient concurrent data access. The current literature and reference materials indicate that the slope code proposed in this paper is the first array code that can tolerate any number of errors.

Keywords erasure codes, array codes, horizontal codes, storage systems, high fault tolerance

Dan TANG was born in 1982. He received a Ph.D. degree from the Graduate University of Chinese Academy of Sciences in 2010. Currently, he is an associate professor of Computer Science and Technology at Chengdu University of Information Technology. His research interests include coding theory and secret sharing.

Hongping SHU was born in 1974. He received a Ph.D. degree from Sichuan University in 2010. Currently, he is a professor of Computer Science and Technology at Chengdu University of Information Technology. His research interests include data mining and software engineering.