计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2023,32(2):394−399 [doi: 10.15888/j.cnki.csa.008969]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

图因子分解的故障节点快速修复①

余春雷[1,3], 王　娥[2], 刘　星[1], 谢　锐[1], 冉　彪[1]

[1](四川文理学院 智能制造学院, 达州 635002)
[2](长安大学 信息工程学院, 西安 710064)
[3](智能制造产业技术研究院, 达州 635002)
通信作者: 余春雷, E-mail: yclinfinite@163.com

摘　要: 为了提高分布式存储系统中故障节点的修复效率, 提出一种新的部分重复 (fractional repetition, FR) 码的构造算法. 该算法利用完全图的因子分解进行构造, 称为 CGFBFR (complete graph factorization based FR) 码. 该算法首先对完全图进行因子分解, 分解完成以后确定完全图的因子分解个数, 根据需要存储数据块的重复度来选择完全图的因子个数, 将完全图选中的因子所有顶点当做分布式存储系统中需要存储的数据块, 然后对选中因子图的边进行标记, 标记的边当做分布式数据节点进行存储. 最后根据选中的因子的顶点和边生成编码矩阵, 在分布式存储系统中按照编码矩阵中的数据对数据块分别进行存储. 实验仿真结果显示, 本文提出的一种新的部分重复码构造算法, 与分布式存储系统中的里所 (reed-solomon, RS) 码、简单再生码 (simple regenerating codes, SRC) 以及最新的循环可变部分重复 (variable fractional repetition, VFR) 码相比, 在系统修复故障节点时, 能够快速地修复故障节点, 有效降低了故障节点的修复带宽开销、修复局部性、修复复杂度, 而且构造过程简单, 同时可以灵活选择构造参数, 广泛适用于分布式存储系统中.

关键词: 图因子分解; 完全图; 存储节点; 修复; 部分重复码; 故障诊断

引用格式: 余春雷,王娥,刘星,谢锐,冉彪.图因子分解的故障节点快速修复.计算机系统应用,2023,32(2):394–399. http://www.c-s-a.org.cn/1003-3254/8969.html

Fast Repair of Faulty Nodes Based on Graph Factorization

YU Chun-Lei[1,3], WANG E[2], LIU Xing[1], XIE Rui[1], RAN Biao[1]

[1](School of Intelligent Manufacturing, Sichuan University of Arts and Science, Dazhou 635002, China)
[2](School of Information Engineering, Chang'an University, Xi'an 710064, China)
[3](Intelligent Manufacturing Industry Technology Research Institute, Dazhou 635002, China)

Abstract: To improve the efficiency of repairing faulty nodes in a distributed storage system, this study proposes a new construction algorithm for fractional repetition (FR) codes. The algorithm resorts to the factorization of the complete graph for node construction, and the nodes constructed are referred to as complete graph factorization based FR (CGFBFR) nodes. Specifically, the complete graph is factorized, and the number of factors generated from the complete graph after the factorization is completed is determined. The number of factors of the complete graph is selected according to the repetition degrees of the data blocks that need to be stored. All the vertices of the selected factors of the complete graph are regarded as the data blocks that need to be stored in the distributed storage system. Then, the edges of the selected factor graph are marked and stored as distributed data nodes. Finally, an encoding matrix is generated with the vertices and edges of the selected factors, and the distributed storage system stores the data blocks respectively according to the data in the encoding matrix. The experimental simulation results show that compared with the Reed-Solomon (RS) codes, the simple regenerating codes (SRCs), and the latest cyclic variable FR (VFR) codes in the

① 基金项目: 陕西省重点研发计划 (2021GY-019), 智能制造产业技术研究院开放基金 (ZNZZ2106)
收稿时间: 2022-06-15; 修改时间: 2022-09-07; 采用时间: 2022-09-14; csa 在线出版时间: 2022-11-04
CNKI 网络首发时间: 2022-11-16

distributed storage system, the codes generated by the new FR code construction algorithm proposed in this study can quickly repair the faulty node when the system repairs the node. The proposed algorithm can be widely applied to distributed storage systems as it reduces the repair bandwidth overhead, repair locality, and repair complexity of the faulty node, provides a simple construction process, and allows flexible selection of construction parameters.

Key words: graph factorization; complete graph; storage node; repair; fractional repetition (FR) codes; fault diagnosis

如今的社会是一个互联网高速发展的社会, 大量互联网公司和政府部门每天运营中产生海量的数据[1], 数据的价值和重要性不言而喻, 因此如何维护数据的可靠性和稳定性[2,3] 是急需要解决的关键技术. 现有的集中式存储主要靠数据服务器和存储设备[4]. 但是集中式存储, 设备价格昂贵, 而且需要大量的管理时间以及维护成本[5]. 集中式存储一旦遭到破坏, 数据将永久毁灭. 相较于集中式存储, 具有可扩展、设备成本低、高性能、易用性的分布式存储系统成为目前主流存储系统[6]. 分布式存储对数据的存储采用的策略是复制策略和纠删码策略[7,8]. 复制策略就是把需要存储的原始文件复制n份, 然后对复制的所有数据进行存储, 通过复制的副本数据来保障数据的可靠性, 当出现数据损坏, 可以直接复制副本数据快速修复, 整个过程简单易实施, 但复制策略的缺陷是存储开销大[9].

与复制策略相比, 纠删码策略只需要存储少量的编码数据块. 使系统的存储开销得到有效地降低, 同时纠删码策略通过编码产生的校验数据块, 使系统具有更好的容错性. 但在分布式存储系统中通过纠删码对故障节点进行修复时, 系统需要消耗较大的修复带宽开销[10]. 目前里所 (reed-solomon, RS) 码和磁盘阵列码[11] 等典型的纠删码被广泛应用于分布式存储系统中. 如图 1 所示为 RS 编码和重构原理过程. 当其中任意一个数据块发生故障, 都可以通过解码恢复出故障数据块.

针对复制策略和纠删码策略等局限性[12,13], 研究者通过研究发现在存储开销跟修复带宽开销之间折中, 提出简单再生码 (simple regenerating codes, SRC) 的编码方式[14]. 简单再生码是通过大量的有限域运算编码构造[15]. 不需要消耗大量的存储开销, 并且在修复故障节点时, 只需要连接相应节点就可以修复数据, 修复带宽开销较低. 但修复过程需大量解码运算, 使得修复过程非常复杂, 修复时间增多, 而且还需要连接多个节点进行修复, 因此修复局部性高[16].

部分重复 (fractional repetition, FR) 码[17] 无需进行有限域运算, 同时也可以降低节点存储开销[18]. 在对故障节点修复时, 具有较低的修复局部性和修复带宽开销[19,20]. 因此, 部分重复码被广泛研究. 目前主流的 FR 码构造方法有如下几种: 基于正则图、分组设计[21] 及可分解设计[22], 但构造过程复杂, 且不能灵活选择部分重复码的构造参数, 不适应实际的存储系统[23,24]. 因此提出基于完全图图因子分解的部分重复 (complete graph factorization based FR, CGFBFR) 码的构造算法.

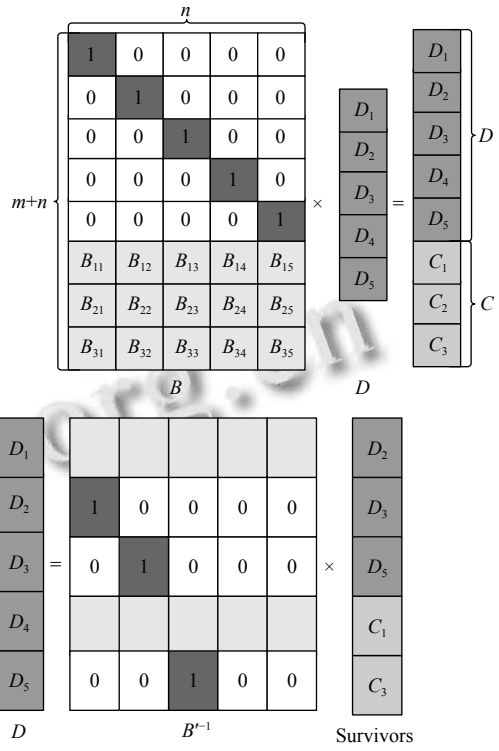

图 1 RS 编码和重构原理

1 图的因子分解

图的因子分解是图分解的一种方法, 记图G表示为$G=(V, E)$, 图H为图G的子图, 满足$V(H)\subseteq V(G)$且$E(H)\subseteq E(G)$, 记为$H\subseteq G$. 若 $V(H)=V(G)$, 图H是图G的生成子图. 对于n阶完全图记为K_n, 图G的阶就是图G所有的顶

点个数.

如果图G可以分解成生成子图$F_1,F_2,\cdots,F_n$,且$V(F_i)=V(G)$,图G分解的每个生成子图满足$\bigcup_{i=1}^{n}E(F_i)=E(G)$,$E(F_i)\cap E(F_j)=\phi\ (i\neq j)$,称这些生成子图$F_i\ (i=1,2,\cdots,n)$为$G$的因子,称为图$G$的因子分解. 若每个分解出来的因子都是$m$正则图时,则称图$G$是可$m$因子分解的.

2 部分重复码的构造

在(n,k,d)分布式存储系统 (distributed storage system, DSS) 中,其中n表示存储节点个数,即分布式系统中存储数据的节点. $d\geqslant k$是修复故障节点时连接的节点个数,$k<n$表示恢复出原文件需要节点个数.

定义 1. 在(n,k,d)系统中[25], (n,d,θ,ρ)部分重复码可以由n个子集的集合$N=\{N_1,\cdots,N_n\}$表示,ρ是数据块的重复度,每个子集中的符号均属于符号集$[n]=\{1,2,\cdots,\theta\}$. 该$(n,d,\theta,\rho)$部分重复码的构造过程满足如下条件.

(1) 每个节点存储d个数据块,即$|N_i|=d\ (1\leqslant i\leqslant n)$.

(2) $[n]$中的每个元素都出现于N中ρ个不同的子集.

在(n,d,θ,ρ) FR 码的构造过程中,其中参数满足$\theta\rho=nd$,首先对原文件进行分割为m个数据块,如图 2 所示,分别为数据块 m_0 到 m_8,其次通过最大距离可分 (maximum distance separable, MDS) 码对 m_0 到 m_8 进行编码,产生 m_0 到 m_8、P_9 等编码数据块,P_9 为校验块数据块,保护数据的可靠性. 然后采用复制策略,对 m_0 到 m_8、P_9 等编码数据块复制ρ份,最后通过部分重复码的内部构造算法把所有的数据块存储在n个节点里.

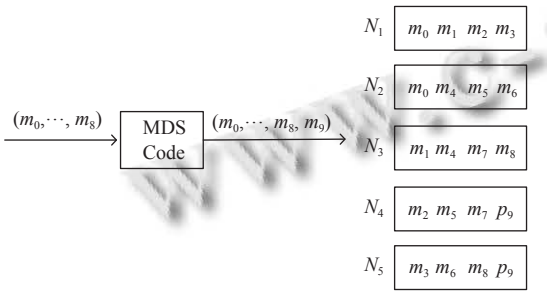

图 2 部分重复码构造过程

3 基于图因子分解的部分重复码的构造

本文提出一种新的部分重复码的构造算法,通过完全图的因子分解进行构造,构造过程简单,同时也解决了传统 FR 码存在的不足. 具体构造步骤如下.

步骤 1. 对完全图K_{2k}的图G做可 1 因子分解,得到$F_1,F_2,\cdots,F_l$,图G中的每个顶点$v_i\ (1\leqslant i\leqslant n)$视为分布式存储系统中需要存储的数据块$d_i\ (1\leqslant i\leqslant n)$,其中$l$满足公式如下:

$$l=\frac{(2n)!}{2^n n!} \tag{1}$$

步骤 2. 确定分布式存储系统中 FR 码构造过程每个数据块$d_i\ (1\leqslant i\leqslant n)$的重复度$\rho\ (1\leqslant\rho\leqslant l)$. 因为图$G$的所有顶点都包含于每个因子$F_\sigma\ (1\leqslant\sigma\leqslant l)$之中,因此可以构造$\binom{l}{\rho}$种重复度为$\rho$的 FR 码.

步骤 3. 在分解的所有因子$F_1,F_2,\cdots,F_l$中,根据数据块的重复度ρ,选择任意ρ个因子的边分别编号为$e_\mu\ (1\leqslant\mu\leqslant n\rho)$.

步骤 4. 构造数据矩阵,通过数据矩阵$m_{\mu,i}\ (1\leqslant\mu\leqslant n\rho,1\leqslant i\leqslant n)$对部分重复码进行构造,$m_{\mu,i}$的横坐标代表部分重复码的存储节点$N_\mu\ (1\leqslant\mu\leqslant n\rho)$,在数据矩阵$m_{\mu,i}$中,对矩阵中相应数字为 1 的编码数据块$d_i\ (1\leqslant i\leqslant 2n)$进行存储,其中$m_{\mu,i}$数据矩阵构造公式如下:

$$m_{\mu,i}=\begin{cases}1, \text{若}v_i\text{与}e_\mu\text{相关联}\\ 0, \text{其他情况}\end{cases} \tag{2}$$

具体地,可以对K_6进行因子分解,然后构造一个重复度为$\rho\ (1\leqslant\rho\leqslant 15)$的 FR 码. 如果取重复度$\rho=3$,因此可以在$F_1,F_2,\cdots,F_{15}$所有因子中任选 3 个进行构造数据矩阵,这里我们在 15 个因子中选取F_1,F_2,F_3,然后对选中F_1,F_2,F_3因子的边进行依次编号为$e_\mu\ (1\leqslant\mu\leqslant 9)$,如图 3 所示.

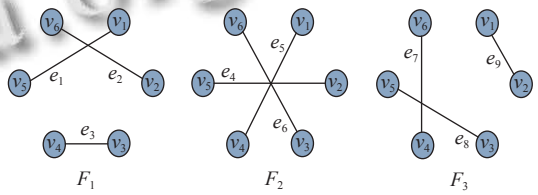

图 3 K_6的部分因子分解

通过图 2,我们可根据式 (2) 求出数据矩阵m_1如下:

$$m_1=\begin{bmatrix}1&0&0&0&1&0\\0&1&0&0&0&1\\0&0&1&1&0&0\\0&1&0&0&1&0\\1&0&0&1&0&0\\0&0&1&0&0&1\\0&0&0&1&0&1\\0&0&1&0&1&0\\1&1&0&0&0&0\end{bmatrix}$$

基于完全图因子分解的部分重复码如图 4 所示, 部分重复码的存储节点$N_\mu(1\leqslant\mu\leqslant 9)$为数据矩阵$m_1$的横坐标, 对标记为 1 数据块$d_i$ $(1\leqslant i\leqslant 6)$进行存储. 利用$K_6$的可 1 因子分解, 重复度$\rho=3$的 FR 码的构造方法总共有$\binom{15}{3}$种. 因此, CGFBFR 码具有灵活的参数选择性, 而且有多种构造方法.

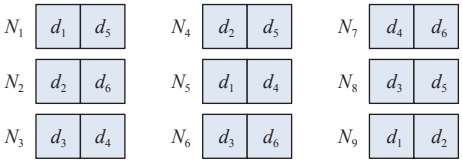

图 4 基于K_6因子分解的 CGFBFR 码

4 性能分析

对基于完全图K_{2k}可 1 因子分解构造的 CGFBFR 码与目前分布式系统主要研究的 RS 码、SRC 以及最新的循环可变部分重复 (variable fractional repetition, VFR) 码性能进行分析.

表 1 给出了 SRC、RS 码、三副本、以及 VFR 码与 CGFBFR 码几种编码方案性能的分析. 为了便于比较, 文件大小取$M=2000$ Mb, t为单位时间内寻址时间, SRC 的子文件数$f=3$, 且 CGFBFR 码外部编码采用$(k+3,\ k)$码.

表 1 编码方案的性能分析

编码方案	(n,k,f)SRC	(n,k)RS	CGFBFR	三副本	VFR
存储开销	$(f+1)M/k$	M/k	$2M/k$	M	$3M/k$
修复复杂度	$(f-1)(f+1)$	k^2+k	0	0	0
寻址时间	$(f-1)(f+1)tM$	$(k^2+k)tM$	tM	tM	tM

4.1 修复带宽开销

在修复故障节点时下载的数据量就是修复带宽开销. 当单节点故障时, 如表 2 所示, 若采用(n,k,f)SRC, 因为 SRC 每个节点存储的数据块为$f+1$, 每一个数据块的大小是M/fk, 修复时需要从其他数据节点下载f个数据块, 因此(n,k,f)SRC 修复单节点故障的带宽开销为$(f+1)M/k$; $(n,\ k)$RS 码修复时需要靠原文件修复故障节点, 所以带宽开销为M; 对于 VFR 码, 单节点故障时, 修复带宽开销为$3M/k$. 对于 CGFBFR 码, 每个节点存放的数据块大小为 2, 数据块每个的大小为M/k, 因此 CGFBFR 码修复带宽开销为$2M/k$.

当两个节点发生故障时, 如表 2 所示, $(n,\ k)$RS 编码在修复数据时仍需要通过原文件来修复故障节点, 因此修复带宽开销仍为M. 对于 $(n,\ k,\ f)$SRC, 修复带宽开销为M. VFR 码的两节点故障时修复带宽开销也为M. 对于 CGFBFR 码, $2M/k$为单个存储节点存储开销, 因此$4M/k$是两节点故障时的修复带宽开销. 图 5 给出了 4 种编码方式的修复带宽开销, 显而易见, CGFBFR 码的修复带宽更低.

表 2 编码方案的性能分析

编码方案		(n,k,f)SRC	(n,k)RS	CGFBFR	VFR
修复带宽开销	单节点故障	$(f+1)M/k$	M	$2M/k$	$3M/k$
	两节点故障	M	M	$4M/k$	M
修复局部性	单节点故障	$2f$	k	2	2
	两节点故障	k	k	max{4}	$\left\lceil\frac{k}{3}\right\rceil+1$

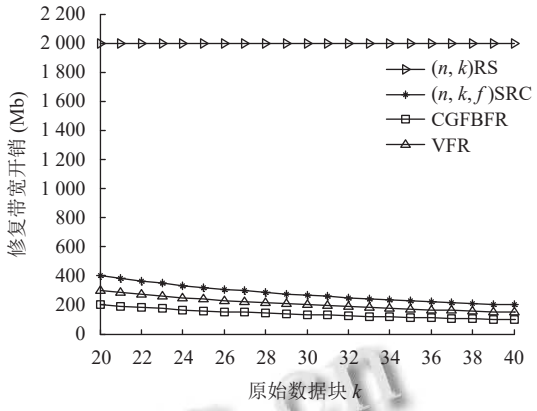

图 5 单节点故障的修复带宽开销对比

4.2 修复局部性

当分布式存储系统中出现单节点故障时, 如图 6 所示. 对于 $(n,\ k,\ f)$SRC 需要$2f$ $(2f\leqslant n)$个节点修复故障节点, 因此 $(n,\ k,\ f)$SRC 系统中修复局部性为$2f$; 对于$(n,\ k)$RS 码的修复局部性是k; VFR 码单节点故障故障修复局部性为 2; 采用 CGFBFR 码, 修复故障节点需要连接的节点数为 2, 因此 CGFBFR 码修复局部性为 2.

图 7 给出了两节点故障时, 4 种不同的编码修复局部性. 对于 $(n,\ k,\ f)$SRC, 恢复原文件需要k存储节点, 通过原文件对故两节点故障进行修复, 因此 $(n,\ k,\ f)$SRC 修复局部性是k; 对于$(n,\ k)$RS 码需要连接k个节点恢复原文件, 然后再对两故障节点进行修复, 所以$(n,\ k)$RS 码的修复局部性也是k; VFR 码的两节点修复局部性为$\lceil K/3\rceil+1$. 对于 CGFBFR 码的修复局部性最大 4.

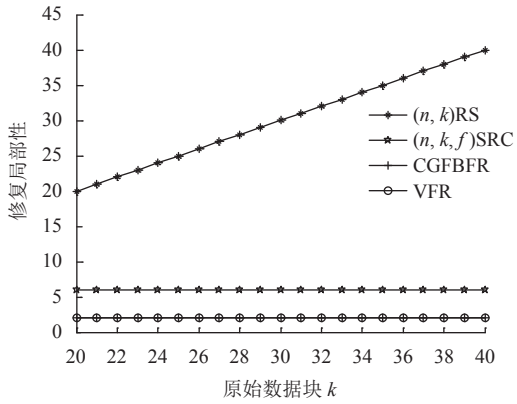

图 6 单节点故障修复局部性

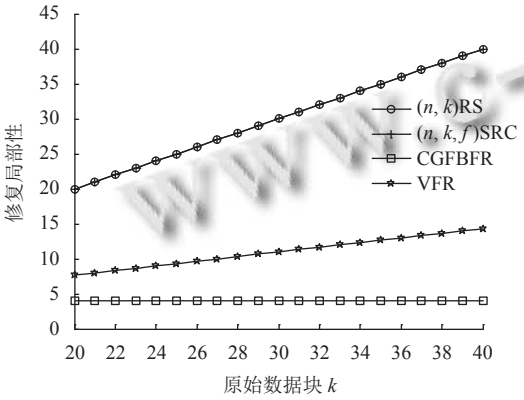

图 7 两节点故障修复局部性

4.3 修复复杂度

对于 RS 码, 在分布式存储系统中修复故障节点时, 系统需要经过k^2-1次有限域加法运算, k^2+k次有限域乘法运算, 因此 RS 码在修复单节点的故障时它的修复复杂度为$\mathrm{O}\left(2k^2+k-1\right)$. 对于 SRC, 系统需要执行$(f-1)(f+1)$次异或运算, 所以 SRC 的修复复杂度为$\mathrm{O}\left(f^2-1\right)$. 对于 CGFBFR 码, 不需要经过有限域运算和异或运算, 直接通过复制数据就可以进行修复, 因此 CGFBFR 码相比于 RS 码和 SRC 具有较低修复复杂度, 能够快速修复故障节点, 降低修复时间.

5 结论

本文提出一种 CGFBFR 码能够满足分布式存储系统的实际需求, 提高了分布式存储系统中故障节点的修复效率, 并且能够灵活的选择构造参数. 在修复故障节点时, CGFBFR 相较于分布式存储系统中其他编码, 在修复局部性、修复复杂度、修复带宽开销都有较大的性能提升. 更加具有发展的前景和应用背景.

参考文献

1 Siddiqa A, Karim A, Gani A. Big data storage technologies: A survey. Frontiers of Information Technology & Electronic Engineering, 2017, 18(8): 1040–1070.

2 Mazumdar S, Seybold D, Kritikos K, *et al*. A survey on data storage and placement methodologies for cloud-big data ecosystem. Journal of Big Data, 2019, 6(1): 15. [doi: 10.1186/s40537-019-0178-3]

3 Lv Z, Li X, Lv H, *et al*. BIM big data storage in WebVRGIS. IEEE Transactions on Industrial Informatics, 2019, 16(4): 2566–2573.

4 Li RN, Song TY, Mei B, *et al*. Blockchain for large-scale Internet of Things data storage and protection. IEEE Transactions on Services Computing, 2019, 12(5): 762–771. [doi: 10.1109/TSC.2018.2853167]

5 Yang Y, Zheng XH, Guo WZ, *et al*. Privacy-preserving smart IoT-based healthcare big data storage and self-adaptive access control system. Information Sciences, 2019, 479: 567–592. [doi: 10.1016/j.ins.2018.02.005]

6 Liang W, Fan YK, Li KC, *et al*. Secure data storage and recovery in industrial blockchain network environments. IEEE Transactions on Industrial Informatics, 2020, 16(10): 6543–6552. [doi: 10.1109/TII.2020.2966069]

7 Lee OT, Akash GJ, Kumar SDM, *et al*. Storage node allocation methods for erasure code-based cloud storage systems. Arabian Journal for Science and Engineering, 2019, 44(11): 9127–9142. [doi: 10.1007/s13369-019-03983-8]

8 Knauft E, Goggin EJ, Li RC, *et al*. Automatically removing dependency on slow disks in a distributed storage system: US, 10168942. 2019-01-01.

9 余春雷, 王静, 王秘, 等. 图因子分解的部分重复码构造. 中国科技论文, 2019, 14(11): 1260–1264.

10 Vaishampayan VA. Lattice erasure codes of low rank with noise margins. Proceedings of the 2018 IEEE International Symposium on Information Theory (ISIT). Vail: IEEE, 2018. 961–965.

11 Balaji SB, Krishnan MN, Vajha M, *et al*. Erasure coding for distributed storage: An overview. Science China Information Sciences, 2018, 61(10): 100301. [doi: 10.1007/s11432-018-9482-6]

12 余春雷, 王静, 杨成福, 等. 哈夫曼树的异构部分重复码构造. 北京邮电大学学报, 2021, 44(6): 116–121.

13 Papailiopoulos DS, Dimakis AG, Cadambe VR. Repair optimal erasure codes through hadamard designs. IEEE Transactions on Information Theory, 2013, 59(5): 3021–3037. [doi: 10.1109/TIT.2013.2241819]

14 Goparaju S, Fazeli A, Vardy A. Minimum storage regenerating codes for all parameters. IEEE Transactions on Information Theory, 2017, 63(10): 6318–6328. [doi: 10.1109/TIT.2017.2690662]

15 Kralevska K, Gligoroski D, Jensen RE, *et al*. HashTag erasure codes: From theory to practice. IEEE Transactions on Big Data, 2018, 4(4): 516–529. [doi: 10.1109/TBDATA.2017.2749255]

16 朱兵, 李挥, 陈俊, 等. 基于可分组设计的部分重复码研究. 通信学报, 2015, 36(2): 98–105. [doi: 10.11959/j.issn.1000-436x.2015038]

17 Ahmad I, Wang CC. Flexible fractional repetition codes for distributed storage networks. Proceedings of the 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton). Monticello: IEEE, 2018. 805–812.

18 Zhu B. A study on universally good fractional repetition codes. IEEE Communications Letters, 2018, 22(5): 890–893. [doi: 10.1109/LCOMM.2018.2813391]

19 Porter A, Silas S, Wootters M. Load-balanced fractional repetition codes. Proceedings of the 2018 IEEE International Symposium on Information Theory (ISIT). Vail: IEEE, 2018. 2072–2076.

20 Wang J, Wang TT, Liu XY, *et al*. Locally repairable codes based on fractional repetition codes in distributed storage systems. 14th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM 2018). Lancaster, PA: DEStech Publications, Inc., 2018, 306: 578–587.

21 Dukes PJ, Lamken ER, Ling ACH. Resolvable group divisible designs with large groups. The Electronic Journal of Combinatorics, 2016, 23(4): 4.24. [doi: 10.37236/5435]

22 Zhu B, Zhang SG, Wang WP. Expandable fractional repetition codes for distributed storage systems. Proceedings of the 2021 IEEE Information Theory Workshop (ITW). Kanazawa: IEEE, 2021. 1–5.

23 Zhu B, Shum KW, Li H, *et al*. On the optimal reconstruction degree of fractional repetition codes. Proceedings of the 2019 IEEE International Symposium on Information Theory (ISIT). Paris: IEEE, 2019. 1557–1561.

24 Su YS. Optimal pliable fractional repetition codes that are locally recoverable: A bipartite graph approach. IEEE Transactions on Information Theory, 2019, 65(2): 985–999. [doi: 10.1109/TIT.2018.2876284]

25 Aghayev A, Weil S, Kuchnik M, *et al*. File systems unfit as distributed storage backends: Lessons from 10 years of Ceph evolution. Proceedings of the 27th ACM Symposium on Operating Systems Principles. Huntsville: ACM, 2019. 353–369.

(校对责编: 牛欣悦)