**文章编号**:1007-5321(2019)05-0075-08 **DOI**:10.13190/j.jbupt.2019-026

# 非均匀故障保护的分组修复码构造

王 静[1], 刘 艳[1], 余春雷[1], 王 秘[1], 刘向阳[2]
(1. 长安大学 信息工程学院, 西安 710064; 2. 国防科技大学 信息通信学院, 西安 710106)

**摘要**: 考虑到实际分布式存储系统中存在热度不同的文件,构造了一种基于非均匀故障保护的分组修复码(GRC-NFP),可对热文件和高故障概率节点提供更高等级保护,并降低多故障节点修复的磁盘读取开销. 在文件冷热分组后,用所存目标节点故障概率表征数据块故障概率,并排序,存入长度依次递增的多个数据分组,并生成组编码块. 性能分析和实际系统部署结果表明,与里德-所罗门码和分组修复码相比,GRC-NFP 可在存储开销较小的条件下拥有较高的容错能力和较低的修复局部性,并且使热文件能够受到更有效地保护. 系统部署下较少的编码和故障修复时间进一步证明了 GRC-NFP 的可行性.

**关 键 词**: 分布式存储系统; 非均匀故障保护; 分组修复码; 文件可靠性
**中图分类号**: TN911.2 **文献标志码**: A

# Construction of Group Repairable Codes for Non-Uniform Fault Protection

WANG Jing[1], LIU Yan[1], YU Chun-lei[1], WANG Mi[1], LIU Xiang-yang[2]
(1. School of Information Engineering, Chang'an University, Xi'an 710064, China;
2. College of Information and Communication, National University of Defense Technology, Xi'an 710106, China)

**Abstract**: Considering that there are files with different heat in actual distributed storage systems, a class of group repairable codes based on non-uniform fault protection (GRC-NFP) is proposed. GRC-NFP provides higher protection for hot files and nodes with high fault probability, and reduces the disk I/O overhead for repairing multiple failed nodes. Specifically, after hot and cold grouping, the fault probabilities of data blocks are represented and sorted by that of the stored target nodes. Data blocks are stored into multiple data groups with increasing lengths, and group encoded blocks are further generated. Performance analysis and actual system deployment showed that GRC-NFP had higher fault tolerance and lower repair locality under less storage overhead compared with Reed-Solomon codes and group repairable codes. Moreover, the hot files can be protected more effectively by adopting GRC-NFP. The fewer coding and fault repair time under system deployment further proved the feasibility of GRC-NFP.

**Key words**: distributed storage system; non-uniform fault protection; group repairable codes; file reliability

**收稿日期**: 2019-03-15
**基金项目**: 陕西省自然科学基金项目(2019JM-386); 中央高校基本科研业务费专项资金项目(300102248104, 300102248201, 300102248401); 大学生创新创业训练计划项目(201910710071)
**作者简介**: 王 静(1982—), 女, 教授, 硕士生导师, E-mail: jingwang@chd.edu.cn.

随着互联网技术的发展,海量数据存储引起研究者的广泛关注. 传统的数据存储方案已经不能适应当前海量数据的存储,分布式存储系统逐渐成为主流的数据存储方式. 通过将数据存储在多个独立物理存储设备上,分布式存储系统不仅能分担存储负载,而且成本低廉,可扩展性高,适用当前的海量数据存储. 在分布式存储系统中,通常利用存储冗余数据确保数据存储的可靠性. Hadoop 分布式文件系统、Google 文件系统等利用复制方式来保证数据存储的高可靠性[1-2],但需要存储多个文件副本,系统存储开销过大. 相比于复制方式,纠删码策略有效降低了存储开销[3-4],但是在修复故障节点时需要下载整个文件大小的数据量,修复带宽开销过大. Wu 等[5-7]指出,将网络编码技术应用于分布式存储系统,可有效降低数据修复过程中的带宽开销. 基于该冗余方式,Dimakis 等[8]提出再生码的概念,实现了存储开销和修复带宽开销之间的最佳折中. 再生码可以极大地减少故障节点修复时的数据传输量,但是需要连接较多的存活节点,修复局部性较高. 局部性修复编码(LRC, locally repairable codes)通过将存储节点分组并产生组编码块来降低故障节点修复时需要连接的节点数[9-11],在单节点故障时具有较好的修复局部性,但是在多节点故障时仍需连接较多存活节点. 基于 LRC,分组修复码(GRC, group repairable codes)在分组内生成多个组编码块,降低了多故障节点修复时需要读取的数据量[12].

上述冗余方案中,视所有节点故障概率相同. 受磁盘物理参数的影响,存储系统中每个存储节点故障概率呈现出不均匀分布的特征. 在分布式系统中根据"帕累托"原则,80% 的访问集中在 20% 的文件中,这 20% 的文件称为热文件,剩余的文件则统称为冷文件. Chang 等[13]将文件大小作为文件热度区分的标准,这显然对于目前的文件系统是不合理的. Yim 等[14]指出热数据可以被有效地压缩,而冷数据因为被编码不能被有效压缩,故 Kim 等[15]利用文件的压缩比与特定数值的大小来判定文件热度,这种方法的缺点在于计算数据压缩比的开销很大. 目前,针对高故障概率节点和冷热文件进行不同的等级保护,邓俊杰[16]利用分组思想将原始数据块分成长度不等的数据,分组并异或组内数据,生成一个组编码块,该方案可以对高故障概率节点进行高等级保护,但是是否可以有效保护热文件没有经过验证,并且在多节点故障时磁盘读取开销仍较大.

为此,为了对分布式存储系统中不同热度文件采取不同等级保护,在节点非均匀故障的基础上结合分组修复码,提出一种新的编码方案——非均匀故障保护的分组修复码(GRC-NFP, group repairable codes based on non-uniform fault protection). 具体地,冷热文件划分完成后,将数据块按照目标存储节点故障概率进行排序,存入长度不等的数据分组中,并且根据故障概率生成数目不等的组编码块. 理论分析和平台部署结果表明,该非均匀故障保护的分组修复码在提高热文件保护等级的同时确保节点故障的修复局部性始终小于分组修复码. 与里德-所罗门码(RS codes, Reed-Solomon codes)相比,存储效率虽然无法达到最优,但优于分组修复码. 实际平台下,编码时间较少的同时拥有高等级保护下的低故障恢复时间,证明了 GRC-NFP 的部署使得系统性能达到一定的提升.

## 1 分组修复码

最大距离可分码(MDS codes, maximal distance separable codes)把大小为 $M$ 的文件平均分为 $k$ 个数据块,利用生成矩阵编码生成 $n=k+m$ 个相同大小的编码块,并分别存放在分布式存储系统中的 $n$ 个节点中.

为了描述方便,称 MDS 码为原纠删码. GRC 的具体构造如下:利用分组思想将原纠删码 $k$ 个数据块分成 $L$ 个组,记为 $S_l(l=1,2,\cdots L)$,组 $S_l$ 包含 $k_l$ 个数据块. 令原纠删码的全局编码块 $m=m_0+m_1$ 保持原纠删码前 $m_0$ 个全局编码块不变,为每个组计算 $m_1$ 个组编码块,每个组的组编码块和原纠删码生成剩余 $m_1$ 个全局编码块的计算方法相同,生成矩阵中保留本组数据块编码系数,其余组全部置零. 另外,将前 $m_0$ 个全局编码块记为 $S_{l+1}$ 组,异或组内数据,生成组编码块. 具体的(13, 6)GRC 编码过程如图 1 所示.

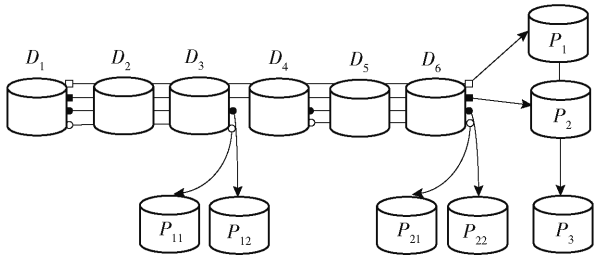


图 1 (13,6)GRC 编码示意图

如图 1 所示,(13, 6)GRC 将数据块 $D_1 \sim D_6$ 均分为两组 $S_1$ 和 $S_2$,每组生成 $m_1=2$ 个组编码块 $P_{11}$、

$P_{12}$和 $P_{21}$、$P_{22}$，保留原纠删码 $m_0=2$ 个全局编码块 $P_1$ 和 $P_2$，并且将 $P_1$ 和 $P_2$ 视为一组生成组编码块 $P_3$.

## 2　非均匀故障保护的分组修复码

### 2.1　文件热度计算

基于以上研究，将文件分为首次存储和非首次存储并分别使用不同计算方式定义其文件热度.

**定义 1**　定义文件的冷热分类 $\boldsymbol{C}=\{F_{\text{cold}},F_{\text{hot}}\}$，其中 $\boldsymbol{C}$ 为文件冷热的假设. $F_{\text{cold}}$、$F_{\text{hot}}$分别指冷文件和热文件，并且包含了系统中的所有文件.

对于首次存储的文件，文件在指定计算周期内的热度为

$$H=\sum_{i=1}^{n}\upsilon_i\chi \tag{1}$$

其中：$\upsilon_i$ 表示系统文件中数据块 $i$ 的影响因子，$\chi$ 表示数据块 $i$ 在指定计算周期内的访问次数，$n$ 为文件中编码数据块的总数. 若编码数据块 $i$ 在指定计算周期内被用户访问，$\upsilon_i$ 较大.

对于非首次存储的文件，使用数据块的重要特性引用量来表征文件热度，故文件在指定计算周期内的热度为

$$H=\sum_{t=1}^{r}Z_t \tag{2}$$

其中：$Z_t$ 表示文件中数据块的引用量，$r$ 表示编码数据块总数. 式(1)和式(2)中考虑了首次和非首次存储这一重要因素. 接下来，利用阈值函数来判断文件的冷热性. 参照强度不变性质设置阈值：

$$f(\omega)=\left(\prod\omega\right)^{1/n} \tag{3}$$

其中：对于首次存储文件，$\omega$ 表示用户在指定计算周期内对数据块的访问次数($\omega=\chi$)；对于非首次存储文件，$\omega$ 表示用户在指定计算周期内的数据块引用量($\omega=Z_t$)；$n$ 表示文件中编码数据块总数，当文件热度计算值高于上述阈值则为 $F_{\text{hot}}$；反之则为 $F_{\text{cold}}$.

### 2.2　冷热文件的转化

由于用户访问行为的不确定性，文件访问热度在一段周期之后可能发生变化，即文件热度为动态变化. 因此，可根据以下过程来进行冷热文件之间的动态转换.

**步骤 1**　当文件按照首次存储与非首次存储计算热度后，建立冷文件访问量表 $P$ 和热文件访问量表 $Q$，其中 $P$ 为存储后冷文件的集合，$Q$ 为存储后热文件的集合.

**步骤 2**　考虑文件热度的更新周期 $T$，在更新周期后，重新计算文件热度，将 $Q$ 中低于式(3)中阈值的文件重新存储在 $P$ 中，同时将 $P$ 中高于式(3)中阈值的文件重新存储在 $Q$ 中，达到量表的动态更新. 对于更新周期 $T$ 的选取，若更新周期 $T$ 选择较短，频繁更新文件热度的操作会影响系统对外整体性能，但若更新周期 $T$ 选择较长，会弱化文件热度的变化. 因此，综合考虑对于更新周期 $T$ 的选择，当 $T$ 为 2 个计算周期时，进行冷热文件的转化.

### 2.3　非均匀故障保护的分组修复码构造

下面详细论述$(k_0,m_0,L,\tau)$GRC-NFP 的构造过程. 原纠删码 $k$ 个原始数据块分成 $L$ 个数据分组，且第 $i(1\leqslant i\leqslant L)$ 个数据分组存储容量为 $k_0+2(i-1)$，即第 $i$ 个数据分组可存储 $k_0+2(i-1)$ 个数据块. 根据数据块目标存储节点的故障概率从高到低排序，第 1 个数据分组放入故障概率最高，即最易故障的 $k_0$ 个数据块；第 2 个数据分组放入次易故障的 $k_0+2$ 个数据块；按此类推，第 $L$ 个数据分组放入最不易故障的 $k_0+2(L-1)$ 个数据块.

保持原纠删码后 $m_0$ 个全局编码块不变，分别记为 $P_{i+L-m_1}(m_1\leqslant i\leqslant m)$. 对于故障概率大于 $\tau$ 的 $\mu$ 个高故障数据分组，根据 MDS 码分别生成 $m_1$ 个组编码块$P_{li}(1\leqslant i\leqslant m_1,1\leqslant l\leqslant\mu)$，只需将除本组之外的其余组数据置零，则 $m=m_0+m_1$. 组编码块 $P_{li}(1\leqslant i\leqslant m_1,1\leqslant l\leqslant\mu)$是原纠删码剩余 $m_1$ 个全局编码块在每个数据分组的投影. 对于故障概率小于 $\tau$ 的 $L-\mu$ 个低故障数据分组，分别异或组内数据块生成一个组编码块 $P_i(\mu\leqslant i\leqslant L)$. 将 $m_0$ 个全局编码块作为 $S_{l+1}$组，生成组编码块 $P_{i+L+m_0}$. 编码公式表示为

$$P_{i+L+m_0}=\sum_{j=L+1}^{L+m_0}P_j \tag{4}$$

$$P_i=\sum_{j=1+\sum_{n=1}^{i-1}[k_0+2(n-1)]}^{\sum_{n=1}^{i}[k_0+2(n-1)]}D_j,\ \mu<i\leqslant L \tag{5}$$

$$P_{i+L-m_1}=\sum_{j=1}^{k}g_{ij}D_j,\ m_1<i\leqslant m \tag{6}$$

$$P_{li}=\sum_{j=1+k-\sum_{n=l}^{L}[k_0+2(n-1)]}^{\sum_{n=1}^{l}[k_0+2(n-1)]}g_{ij}D_j,\ 1\leqslant i\leqslant m_1,1\leqslant l\leqslant\mu \tag{7}$$

基于上述构造过程，可得总编码块数 $n=\mu m_1+$

$L-\mu+m_0+1+\sum_{i=1}^{L}k_i, k_i=k_0+2(i-1)$，存储开销如下：

$$\frac{n}{\sum_{i=1}^{L}k_i}=\frac{L+(m_1-1)\mu+m_0+1+\sum_{i=1}^{L}k_i}{\sum_{i=1}^{L}k_i} \quad (8)$$

根据式(8)，选取不同的 $k_0$ 值可以影响编码的存储效率. 由上述定义可知，将高故障概率数据节点放入小分组中，显而易见，小分组的数据可用性高一些. 数据可用性是指某些存储节点故障时，数据块能成功修复的概率. 小分组中的数据块数目少，在拥有相同组编码块数目的情况下，小分组中保存的冗余信息就多一些，能够成功恢复的概率就高一些. 所以，决定小分组中数据块数目的参数 $k_0$ 对于文件整体的可靠性具有一定影响.

下面以图2为例说明 GRC-NFP 的具体编码过程. 为表示一般性，$\tau$ 取 0.2.

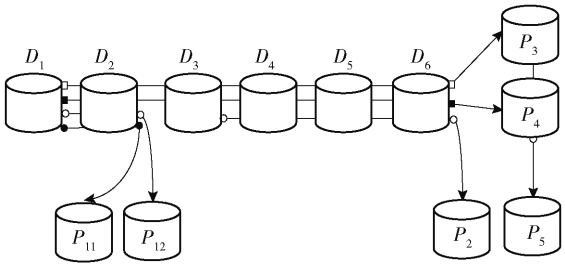


图2 (2, 2, 2, 0.2)GRC-NFP 编码示意图

(2, 2, 2, 0.2)GRC-NFP 的编码过程及生成矩阵表示为

$$\boldsymbol{C}=\boldsymbol{GD}=$$

$$\left[\begin{array}{cccccc}
\multicolumn{6}{c}{I_6} \\
\hdashline
g_{71} & g_{72} & 0 & 0 & 0 & 0 \\
g_{81} & g_{82} & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 \\
g_{91} & g_{92} & g_{93} & g_{94} & g_{95} & g_{96} \\
g_{10,1} & g_{10,2} & g_{10,3} & g_{10,4} & g_{10,5} & g_{10,6} \\
g_{91}+g_{10,1} & g_{92}+g_{10,2} & g_{93}+g_{10,3} & g_{94}+g_{10,4} & g_{95}+g_{10,5} & g_{96}+g_{10,6}
\end{array}\right]
\left[\begin{array}{c} D_1 \\ D_2 \\ D_3 \\ D_4 \\ D_5 \\ D_6 \end{array}\right]=$$

$$[D_1 \ \cdots \ D_6 \ P_{11} \ P_{12} \ P_2 \ P_3 \ P_4 \ P_5]^{\mathrm{T}} \quad (9)$$

(2, 2, 2, 0.2)GRC-NFP 将 $D_1 \sim D_6$ 数据块非均匀划分成2个数据分组，$(D_1, D_2)$ 为第1个数据分组，该分组数据块故障概率大于 0.2，在组内生成 $m_1=2$ 个组编码块. $(D_3, D_4, D_5, D_6)$ 为第2个数据分组，该分组数据块失效概率小于 0.2，在组内异或生成一个组编码块. 保留原纠删码的全局编码块 $P_3$ 和 $P_4$，将 $P_3$、$P_4$ 视为一个分组生成组编码块 $P_5$.

### 2.4 采用 GRC-NFP 的分布式文件存储

由于 CPU 运算速度一直在急剧增长，然而与 CPU 相比，磁盘 I/O 速度仍然落后约6个数量级，使磁盘操作成为系统的主要性能瓶颈. 在访问文件时，磁盘寻道时间通常是 I/O 延迟的主要部分. 对于小于一个磁盘块的文件，它仍然需要一个磁盘块，占用的空间比它需要得多. 这不仅会导致 I/O 操作过多，而且还会浪费磁盘空间. 故文件分组是减少磁盘 I/O 延迟的一种有效方法. 文件分组是一种通过将相关文件紧密放在磁盘上来减少 I/O 查找时间的方法. 笔者将文件自适应地分为多个小文件分组，在小文件分组中对数据块使用 GRC-NFP 进行纠删保护.

图3为自适应分组编码示意图. 当文件需要进行存储时，根据是否是首次存储进行冷热判断，判断完成后为减少磁盘 I/O 延迟对文件进行自适应分组，每个小文件分组使用 GRC-NFP 进行编码，编码完成后存储在系统中供用户访问.

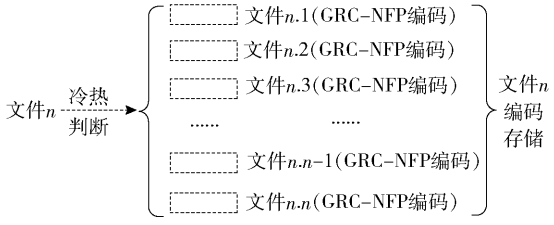


图3 自适应分组编码示意图

## 3 基于非均匀故障保护分组修复码的故障节点修复

按照上述的 GRC-NFP 构造方法将原始文件存储到分布式存储系统中的 $n$ 个节点中，下文所述数据块故障为存储数据块的节点故障. 本节主要讨论 GRC-NFP 的故障节点修复问题.

### 3.1 故障节点的不同等级保护

分析 GRC-NFP 可以为高故障概率节点提供更高等级保护. 假设图1所示(13, 6)GRC 中的 $(D_1, D_2, D_3)$ 分组以及该组的组编码块 $P_{11}$ 和 $P_{12}$ 全部故障. 这种情况下是无法修复的，因为含有3个数据块信息的全局编码块只有 $P_1$ 和 $P_2$，无法从2个方程组中解出3个未知量. 若(2, 2, 2, 0.2)GRC-NFP 中 $(D_1, D_2)$ 分组以及 $D_3$、$P_{11}$ 和 $P_{12}$ 全部故障是可以进行数据修复的，这时由于 $P_2$ 的存在可以先进

行 $D_3$ 的修复，然后利用 $P_3$ 和 $P_4$ 两个全局编码块便可以解出 $D_1$ 和 $D_2$. 这样所有故障的数据块都可以修复完成. 将所有 2 个编码方案中 5 个节点同时故障的情况列举出，(13, 6) GRC 中第 1 分组不能修复率为 2.4%，(2, 2, 2, 0.2) GRC-NFP 中第 1 分组不能修复率仅为 1%. 由此可见，GRC-NFP 可以有效提高对高故障概率节点的保护等级.

### 3.2　多节点故障修复

**定理 1**　GRC-NFP 最多能容忍 $\lambda=(\mu m_1+L-\mu+m_0+1)$ 个数据块和组编码块同时故障.

**证明**　编码块为数据块的冗余数据，编码块的数目也就是包含的冗余数据块数目. 故编码方案中生成的编码块个数为该方案的容错能力. 根据 GRC-NFP 的构造过程，在前 $\mu$ 数据分组中，每组生成 $m_1$ 个组编码块，因此在前 $\mu$ 组可以容忍 $\mu m_1$ 个数据块和组编码块故障. 对后 $L-\mu$ 个数据分组，每组仅有 1 个组编码块，故可以容忍 $L-\mu$ 个数据块和组编码块在数据分组内故障. 并且，存在 $m_0$ 个全局编码块和由全局编码块生成的一个组编码块. 故 GRC-NFP 最多能容忍 $\lambda=(\mu m_1+L-\mu+m_0+1)$ 个数据块和组编码块同时故障. 证毕.

对于 $(n,k)$ MDS 码，根据故障数据块数目可以直接判断该故障情况是否可以成功修复. 若故障数据块数目小于 $n-k$ 则可以修复；否则无法修复. 但是对于 GRC-NFP 来说，由于全局编码块 $m_0$ 和组编码块 $m_1$ 的存在使得可以有组内修复和全局修复 2 种修复. 组内修复指利用组编码块进行修复，全局修复指利用全局编码块进行修复. 换句话说，也就是无法根据故障块数目进行直接判断. 即使故障块数目不大于 $\lambda$，也有可能无法进行修复. 由编码方案将故障情况分为编码块故障和数据块与编码块同时故障. 在编码块故障情况下，只需根据 GRC-NFP 编码规则重新进行编码即可. 故主要讨论另外一种故障情况的修复.

针对单数据块故障，由于组编码块的存在，使得所有单一数据块故障都能修复. 所以，重点讨论 2 个及多个故障块的修复. 在数据块和编码块同时故障的情况下，当故障块数目不大于 $\lambda$ 时进行修复. 整体修复原则是先在组内修复后在全局修复，以期尽可能降低修复开销，最后在全局修复使修复条件发生变化之后再进行组内修复.

当分布式存储系统采用 GRC-NFP 来存储数据，故障节点的具体修复过程包括以下步骤.

**步骤 1**　进行数据分组内修复，判断每个数据分组内数据块故障数目 $\varepsilon$ 和未发生故障的组编码块数目 $\varphi$. 若 $\varepsilon\leqslant\varphi$，则进行数据分组内修复；若 $\varepsilon>\varphi$，则等待进行全局修复.

**步骤 2**　对于无法在数据分组内修复的数据块进行全局修复. 计算未故障的全局编码块与组编码块(处于故障数据块所在分组)数之和，若小于剩余无法修复的数据块数，则无法进行全局修复；反之，则利用未故障的全局编码块与组编码块(处于故障数据块所在分组)可以修复剩余的故障数据块.

**步骤 3**　由于在步骤 2 中全局修复完成后可能会使修复条件发生变化，若在步骤 2 中可以完成全局修复，则再进一步进行数据分组内修复. 重复步骤 1 过程，利用步骤 1、步骤 2 中已经修复好的数据块去修复未修复完成的编码块. 若只有数据块故障，则修复过程只进行步骤 1 和步骤 2.

如图 4 所示，深色数据块代表故障数据块. 在进行修复时，首先进行数据分组内修复. 第 1 数据分组故障数据块数目 4 大于未故障组编码块数目 0，该组无法在数据分组内修复，而第 2 数据分组没有故障块不需要进行修复. 全局编码块所在的分组故障块数目 2 大于未故障组编码块数目 1，$P_3$ 与 $P_4$ 均无法修复. 其次，全局修复剩余的数据块. 由于第 1 数据分组数据块故障的数目 2 大于未故障的全局编码块与组编码块(故障数据块所在分组)数目之和 0，故无法进行全局修复. 故得出结论在该故障情况下无法成功修复.

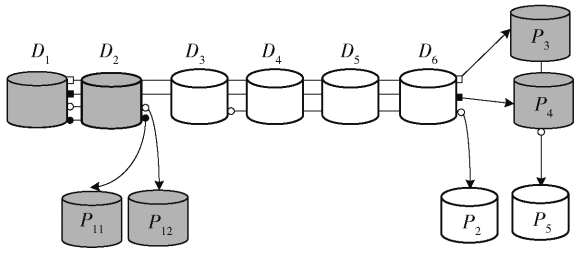


图 4　(2, 2, 2, 0.2) GRC-NFP 码无法修复故障示意图

图 5 中深色数据块同样代表故障数据块. 在进行修复时，首先进行组内修复. 第 1 数据分组故障数据块数目 1 不大于未故障组编码块数目 1，该组可以在数据分组内修复. 而第 2 分组的故障数据块数目 2 大于未故障组编码块数目 0，所以无法进行数据分组内修复. $P_3$ 可以通过 $P_4$ 和组编码块 $P_5$ 进行组内修复. 其次，全局修复剩余的数据块. 由于第 2 分组数据块故障的数目 2 等于未故障的全局编

码块与组编码块(故障数据块所在分组)数目之和2,故 $D_3$、$D_4$ 可以通过 $P_3$ 和 $P_4$ 进行修复. 最后再进行组内修复,利用式(2)中修复完成的数据块可以成功修复 $P_2$. 故得出结论:在该故障情况下可以成功修复.

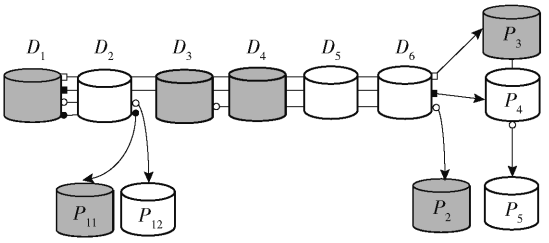


图5 (2, 2, 2, 0.2)GRC-NFP 成功修复故障示意图

## 4 性能分析

下面主要讨论非均匀故障保护的分组修复码 GRC-NFP 的存储开销、文件可靠性和修复局部性,并与现有的 RS 码和 GRC 进行比较.

### 4.1 存储开销

存储开销定义为编码后编码块存储空间与原始数据块存储空间的比值大小. $(n,k)$ RS 码将 $k$ 个数据块利用生成矩阵编码生成 $n$ 个编码块,因此 $(n,k)$ RS 码的存储开销为 $n/k$;根据 GRC 构造过程得总编码块数 $k+m_0+Lm_1+1$,故 GRC 存储开销为 $(k+m_0+Lm_1+1)/k$;根据式(9),GRC-NFP 的存储开销为 $[k+L+(m_1-1)\mu+m_0+1]/k, k=\sum_{i=1}^{L}[k_0+2(i-1)]$. 取 $n=k+4$, $m_0=m_1=2$, GRC 和 GRC-NFP 中 $L=2$,存储开销和数据块数之间的关系如图6所示.

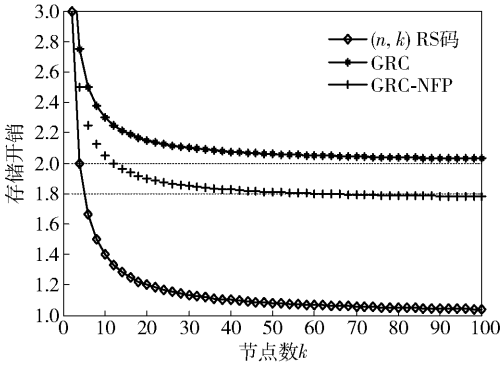


图6 存储开销对比

因为 GRC 码和 GRC-NFP 码均不满足 MDS 性质,相比 RS 码没有达到最优的存储开销. 图6中,RS 码的存储开销随着 $k$ 的增大而减小,达到了最优的存储开销. GRC-NFP 和 GRC 的存储开销随着 $k$ 增大而减小,且 GRC-NFP 的存储开销整体小于 GRC 的存储开销.

### 4.2 文件可靠性

利用成功修复故障节点的比率来衡量 GRC-NFP 的文件可靠性. 若可以成功修复的比率越高,则文件的可靠性越高. 在 $k_0$ 和故障节点数目取值后,计算 GRC-NFP 的不可成功修复概率即可知可成功修复概率. 取 $k_0=10$ 时,总编码块数 $n=28$. 当故障节点数为4时,总可能的故障情况数目为 $C_{28}^4$,根据 GRC-NFP 中故障节点修复原则,统计不可修复的情况数目为 $C_{12}^4+C_{12}^3\times3+C_{12}^1\times4+C_{12}^2\times5$. 故不可成功修复概率为 0.075,可成功修复概率为 0.925,其余不同 $k_0$ 和故障节点数目情况下同理. 仿真过程中,取 $L=2$, $\mu=1$, $m_0=m_1=2$.

图7给出了分组规模 $k_0$ 分别取2、10和20时,采用 GRC-NFP 的文件可靠性. 故障节点数在3以内时,所有故障情况都可以成功修复. $k_0=2$ 时,可成功恢复节点故障的概率都在0.9以上,随着 $k_0$ 的增加,当 $k_0=10$ 和 $k_0=20$ 时,可成功恢复节点故障的概率均小于 $k_0=2$ 时的恢复概率. 当故障节点数为5时,相比 $k_0=2$ 的情况下降0.2左右,且 $k_0=20$ 可恢复故障概率小于 $k_0=10$ 的可恢复故障概率. 因此,对于存储系统中的冷热文件,通过分组规模 $k_0$ 的选择可以提供不同等级保护.

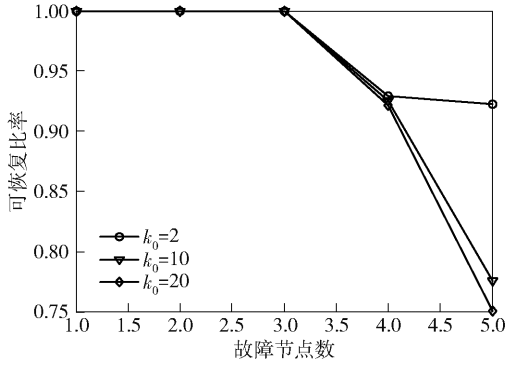


图7 不同 $k_0$ 对文件可靠性的影响

### 4.3 修复局部性

修复局部性为修复故障节点时需读取的磁盘数目,即磁盘读取开销. 这里主要讨论单节点、两节点以及三节点故障时的修复局部性. 为使故障修复都在一个数据分组里,取 GRC 和 GRC-NFP 中 $m_1=3$,并且在三节点故障时 GRC-NFP 中节点数目从8开始取值. 当三节点以内故障时,$(n,k)$ RS 码需要连

接 $k$ 个节点来修复故障节点，即修复局部性为 $k$；GRC 至少需要连接 $k_l$ 个节点来修复故障节点，修复局部性为 $k_l$（$k_l = k/L$）；GRC-NFP 关注高故障概率节点的修复局部性，前 $\mu$ 组内出现单个故障节点时需要至少连接 $k_0 + 2(i-1)$（$1 \leqslant i \leqslant \mu$）个节点来修复故障节点，修复局部性为 $k_0 + 2(i-1)$（$1 \leqslant i \leqslant \mu$）. 这里取 $L = 2, \beta = 1$ 进行以下比较.

图 8 给出了各种编码方案的修复局部性对比. $(n,k)$ RS 码的修复局部性随 $k$ 呈线性增长. GRC 和 GRC-NFP 相比 RS 码的修复局部性显著减少，并且 GRC-NFP 相比 GRC 的修复局部性更小. 系统中数据节点数 $k = 6$ 时，(2, 2, 2, 0.2) GRC-NFP 在 $D_1$、$D_2$ 两节点故障的情况下，需读取 2 个数据块 $P_{11}$ 和 $P_{12}$. 相比同样情况下，(13, 6) GRC 需读取 3 个数据块 $D_1$、$P_{11}$ 和 $P_{12}$，磁盘读取开销减小了 1. 在高故障概率数据节点修复占主要修复情形下，GRC-NFP 可以有效降低多节点故障修复时的磁盘读取开销. 这对于节点数目较大的存储系统故障节点的快速修复具有重要意义.

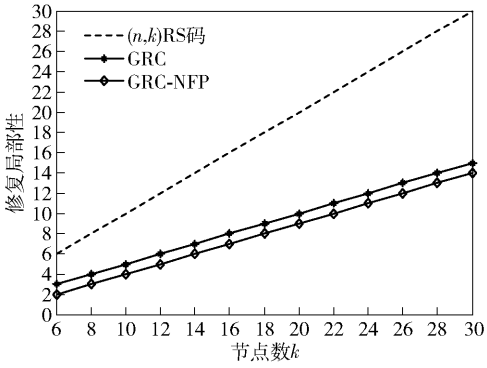


图 8　修复局部性对比

通过与 RS 码和 GRC 的分析与对比可得，在存储开销方面，在同样不满足 MDS 性质的情况下，GRC-NFP 的存储开销要小于 GRC，并且 GRC-NFP 的修复局部性要优于 RS 码与 GRC. GRC-NFP 以相比 RS 码少量的存储开销换取了较优的修复局部性和数据可靠性.

### 4.4　实验平台仿真结果

采用 FastDFS 开源的轻量级分布式文件系统验证所提编码的性能，具体进行编码时间开销及故障恢复时间的实验. 该系统服务器的配置为 Intel(R) Corn(TM) i5-3337U 1.80 GHz，操作系统为 RHEL7 (Linux 内核 3.10.0)，FastDFS 采用 C/S(服务端/客户端)模式，整体框架为使用虚拟机(8 GB 内存和 1 024 ×4 MHz)作为跟踪服务器与存储服务器，并设置客户端. 每台虚拟机需装有 nginx 进行连接通信，并通过配置服务脚本设置为开机自启.

**实验 1**　存储服务器中设置节点 $n = 13$，并分别部署(10, 6) RS 码、(13, 6) GRC 码以及(2, 2, 2, 0.2) GRC-NFP. 在节点中数据块大小分别为 4、8、16 MB 的情况下，统计部署这 3 种码分别需要的时间，为保证可靠性，取 5 次实验数据计算平均值. 如图 9 所示，(2, 2, 2, 0.2) GRC-NFP 所需编码时间最少，比(13,6) GRC 码平均减少 14%，(10,6) RS 码所需时间平均最多.

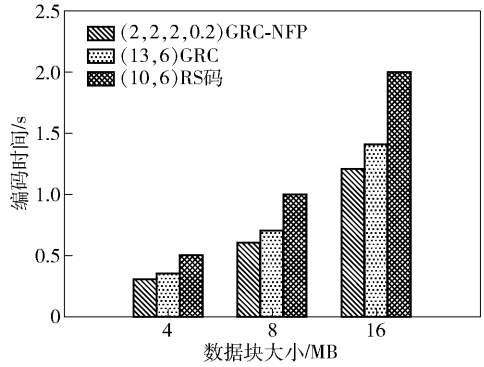


图 9　编码时间对比

从编码矩阵角度考虑 3 种编码的计算复杂度. (10,6) RS 码编码算法基于有限域 GF($2^8$)，基础运算为 $g_{xj}D_k$（$7 \leqslant x \leqslant 10, 1 \leqslant j, k \leqslant 6$），编码块为 $k = 6$ 个基础运算的线性组合. GRC 由于分组的存在，组编码块所需的基础运算为 RS 码编码块的一半. GRC-NFP 不仅存在组编码块，低故障数据分组中组编码块的基础运算为异或运算，减少了有限域上基础运算的操作，故编码时间较 GRC 和 RS 码有所减少.

**实验 2**　考虑分组规模 $k_0$ 与故障修复时间之间的关系. 故障修复时间直接影响系统整体性能. 存储服务器中设置节点 $n = 13$，节点存储容量为 128 MB，分组规模 $k_0$ 分别取 2、10 和 20 时，分别统计故障节点均能被修复的情况下(单节点、两节点以及三节点故障)所需修复时间. 同样，为保证可靠性，取 5 次实验数据计算平均值. 如图 10 所示，在分组规模 $k_0$ 取值依次增加的情况下，节点故障修复时间同样呈增加趋势，这是因为 GRC-NFP 的高等级保护机制是通过分组中所存数据块个数调节的，修复故障节点需顺序连接每个分组中数据块与所需编码块，$k_0$ 的增加意味着所需连接数据块个数的增加.

通过实验 1 和实验 2 的实验结果可知，部署该

码后系统的性能良好,较少的编码时间与高保护等级下的低修复时间,使得用户可以安全可靠地上传访问文件.

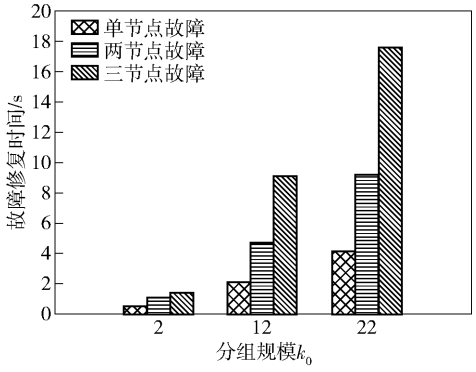


图10 不同 $k_0$ 对故障修复时间的影响

## 5 结束语

构造了一种基于非均匀故障保护的分组修复码. 考虑到实际分布式存储系统中用户访问文件的不均衡性,通过分组规模的选择为热文件提供高于冷文件的保护. 将高概率节点中的数据存入不同数据分组中,并生成数目不等的组编码块,实现保护高概率节点的同时降低了磁盘读取开销. 理论分析与系统部署实验结果表明,相比 RS 码与 GRC,GRC-NFP 拥有较小存储开销的同时修复局部性最小,并且可以提高热文件的保护等级,同时拥有较小的编码时间和故障修复时间,可以更好地适应系统.

**参考文献:**

[1] Shahabinejad M, Khabbazian M, Ardakani M. An efficient binary locally repairable code for Hadoop distributed file system [J]. IEEE Communications Letters, 2014, 18(8): 1287- 1290.

[2] Ghemawat S, Gobioff H, Leung S T. The Google file system [J]. ACM SIGOPS Operating Systems Review, 2003, 37(5): 29-43.

[3] Lee O T, Kumar S D M, Chandran P. Erasure coded storage systems for cloud storage-challenges and opportunities [C]//3rd International Conference on Data Science and Engineering (ICDSE). Cochin: IEEE Press, 2016: 52-58.

[4] Li J, Li B. Erasure coding for cloud storage systems: a survey [J]. Tsinghua Science and Technology, 2013, 18(3): 259-272.

[5] Wu Y. Existence and construction of capacity-achieving network codes for distributed storage [J]. IEEE Journal on Selected Areas in Communications, 2010, 28(2): 277-288.

[6] Koetter R, Medard M. An algebraic approach to network coding [J]. ACM Transactions on Networking, 2003, 11(5): 782-795.

[7] Ahlswede R, Cai N, Li S Y R, et al. Network information flow [J]. IEEE Transactions on Information Theory, 2000, 46(4): 1204-1216.

[8] Dimakis A G, Godfrey P B, Wu Y, et al. Network coding for distributed storage systems [J]. IEEE Transactions on Information Theory, 2010: 56(9): 4539-4551.

[9] Oggier F, Datta A. Self-repairing homomorphic codes for distributed storage systems [C]//2011 Proceedings IEEE INFOCOM. Shanghai: IEEE Press, 2011: 1215-1223.

[10] Gopalan P, Huang C, Simitci H, et al. On the locality of codeword symbols [J]. IEEE Transactions on Information Theory, 2012, 58(11): 6925-6934.

[11] Papailiopoulos D S, Dimakis A G. Locally repairable codes [C]//2012 IEEE International Symposium on Information Theory Proceedings (ISIT). Cambridge: IEEE Press, 2012: 2771-2775.

[12] 林轩. GRC: 一种适用于多节点失效的高容错低修复成本纠删码[J]. 计算机研究与发展, 2014, 51(S): 172-181.
Lin Xuan. GRC: a high fault tolerant low repair cost erasure code for multi-node failures [J]. Journal of Computer Research and Development, 2014, 51(S): 172-181.

[13] Chang L P. Hybrid solid-state disks: combining heterogeneous NAND flash in large SSDs [C]//Proceedings of the 13th ACM International Conference on Asia and South Pacific Design Automation Conference. Seoul: IEEE Press, 2008: 428-433.

[14] Yim K S, Bahn H, Koh K. A flash compression layer for smart media card systems [J]. IEEE Transactions on Consumer Electronics, 2004, 50(1): 192-197.

[15] Kim K, Jung S, Song Y H. Compression ratio based hot/cold data identification for flash memory[C]//2011 IEEE International Conference on Consumer Electronics. Las Vegas: IEEE Press, 2011: 33-34.

[16] 邓俊杰. 云存储中基于非均匀保护策略的纠删码技术研究与实现[D]. 长沙: 湖南大学, 2017.