doi:10.11959/j.issn.1000-436x.2015038

基于可分组设计的部分重复码研究

朱兵，李辉，陈俊，侯韩旭，周泰
（北京大学深圳研究生院 大数据技术研究院 深圳融合网络集成播控技术工程实验室，广东 深圳 518055）

摘 要：针对最小带宽再生情形下的有效修复问题，提出了一种新型部分重复（FR, fractional repetition）码设计。该设计由外部最大距离可分（MDS, maximum distance separable）码和内部重复码组成，称为 GDDBFR（group divisible design based FR）码，可以达到随机访问模式下的系统存储容量，并且能够在很大范围内选择构造参数。理论分析指出，尽管 GDDBFR 码采用基于表格的修复方式，但通常具有大量的节点修复选择方案。此外，实验结果表明，与传统的 RS（Reed-Solomon）码和再生码相比，GDDBFR 码可以显著地减少失效修复时间。
关键词：部分重复码；可分组设计；存储容量；节点修复选择度；修复时间
中图分类号：TP393 **文献标识码**：A

Research on fractional repetition codes based on group divisible designs

ZHU Bing, LI Hui, CHEN Jun, HOU Han-xu, ZHOU Tai
(Institute of Big Data Technologies & Shenzhen Engineering Laboratory of Converged Network Technology, Peking University Shenzhen Graduate School, Shenzhen 518055, China)

Abstract: A novel design of FR (fractional repetition) codes was proposed which aims at providing efficient repair at the minimum bandwidth regenerating point. The design consisted of an outer MDS (maximum distance separable) code and an inner repetition code, called GDDBFR (group divisible design based FR) codes. The proposed codes can achieve the system storage capacity under the random access model and are available for a wide range of parameters. Despite of the table-based repair, theoretical analysis identifies that GDDBFR codes generally have large node repair alternatives. Furthermore, experimental results show that GDDBFR codes can significantly reduce the failure repair time when compared with legacy RS (Reed-Solomon) codes and regenerating codes in the domain.
Key words: fractional repetition codes; group divisible designs; storage capacity; node repair alternativity; repair time

1 引言

近年来，随着计算机技术的飞速发展和网络信息的高速增长，大数据对存储系统提出了严峻的挑战。与此同时，海量数据存储技术及系统设计的研究也迈上了新的台阶。分布式存储系统以其高效的存储性能，如高可用性、高可扩展性等，日益成为现代主流存储系统。然而，大规模的存储系统中经常发生节点离线、突发断电等故障，如何在不可靠的存储节点上提供可靠的数据存储服务，已经成为分布式存储领域一个重要研究课题。

实际的存储系统通常引入冗余来保证容错性。传统的方案采用备份（replication）机制，原文件复制若干倍后分别存储在不同的节点上。只要系统中存在一个未损坏的数据拷贝，该文件就可以正常使用。其中，3 倍复制策略的使用最为常见，如 GFS[1]（google file system）等。备份机制的优点在于简单、易于实施，但是存储效率较低。在冗余信息相等的

收稿日期：2013-10-23；**修回日期**：2013-11-25
基金项目：国家重点基础研究计划(“973 计划”)基金资助项目(2012CB315904)；国家自然科学基金资助项目(61179028)；广东省自然科学基金资助项目(S2013020012822)；深圳市基础研究基金资助项目(JCYJ20140417144423192, JCYJ20130331144502026)
Foundation Items: The National Basic Research Program of China (973 Program) (2012CB315904); The National Natural Science Foundation of China (61179028); The Natural Science Foundation of Guangdong Province (S2013020012822); The Basic Research Program of Shenzhen (JCYJ20140417144423192, JCYJ20130331144502026)

情况下，纠删码（erasure codes）技术的引入可以大幅提高系统的可靠性[2]。特别地，原文件被均分成 k 份，编码生成 n 个数据块（$k<n$），并且满足任意 k 个块就可以恢复出原始数据，该特性进一步称为 MDS 属性。作为一种经典的 MDS 码，RS 码目前正广泛研究并且逐步应用于实际存储系统中[3]。如果系统中出现节点失效，修复过程需要恢复出整个原文件，重新编码并将生成的编码块传输到新的替换节点。可见，为了修复一个失效的数据块，使用 MDS 码的系统带宽消耗相对较高。

从实际的角度来看，每个节点数据存储量和失效修复带宽是衡量系统存储性能的 2 个重要指标。为此，文献[4]提出了再生码（regenerating codes），并且确定了节点存储量和修复带宽之间的折中曲线。当节点失效时，新引入的替换节点可以随机连接 d 个可用节点，每个节点传输大小为 β 的数据来进行修复。再生码修复过程中并不需要重构原文件，相比 MDS 码降低了系统的修复带宽。目前，再生码的研究主要集中于折中曲线的两端极值点，分别对应于最小存储再生 MSR（minimum storage regenerating）码和最小带宽再生 MBR（minimum bandwidth regenerating）码。文献[5~7]给出了再生码的一些具体构造方法。

然而，再生码的修复过程计算复杂度比较高，通常涉及大量的有限域运算。参与修复的节点读出所存储的数据块并进行特定的线性运算，再向替换节点传递组合后的数据块。考虑到实际系统中节点读写带宽小于网络带宽[8]，因此读写带宽很容易成为系统性能瓶颈。为了降低修复过程运算复杂度，El Rouayheb 和 Ramchandran[9]在 MBR 码的基础上提出了部分重复码的概念，指出了 FR 码可以提供精确有效的修复。典型的 FR 码包含 2 个部分：一个外部 MDS 码以及一个内部重复码。数据块经过 MDS 编码后，将输出的编码块复制 f 倍再分散到各存储节点。系统中发生节点失效时，可以通过从其他节点直接下载数据并存储到替换节点来完成修复，不需要额外的运算。当复制倍数 f=2 时，文献[9]提出了基于正则图的 FR 码设计；而对于多节点失效（f>2），FR 码的构造采用 Steiner 系，但参数不在 Steiner 系范围内的 FR 码设计是一个开放性问题。文献[10~12]给出了 FR 码的其他构造方法，如相互正交拉丁方、仿射几何、可分解设计等。

本文提出了一种新型基于可分组设计（GDD，group divisible design）的 FR 码构造，称为 GDDBFR 码。该编码可以在很大范围内选择构造参数，同时通过调整设计的分组，可以构造出不同的 FR 码。如果构造过程中采用的可分组设计是可分解的，系统节点规模则可以灵活地选择。GDDBFR 码的构造解决了文献[9]提出的开放性问题，证实了参数在 Steiner 系之外的 FR 码也是普遍存在的。相比现有的 FR 码构造方法，采用可分组设计更加简洁、直观。进一步分析得出，GDDBFR 码可以达到随机访问模式下的系统存储容量，达到了理论上的最优。尽管 GDDBFR 码采用基于表格的修复方式，系统中的节点仍然具有大量的修复选择方案。此外，实验结果表明与传统的 RS 码和再生码相比，GDDBFR 码可以显著地减少失效修复时间。

2　预备知识

2.1　系统模型

本文用(n,k,d)来确定一个分布式存储系统，其中 n 表示存储系统的节点总数，k 表示重构原文件所需最少节点数，d 表示修复一个失效节点所需的可用节点数，并且满足 $k\leqslant d\leqslant n-1$。当节点失效时，系统从每个连接节点下载 $\beta=1$ 的数据进行修复（每个节点传输一个数据块）。同时，修复模型采用精确修复，即再生的数据和失效数据完全相同。因此，节点的存储容量同样为 d。在这种修复模式下，用 C_{MBR} 表示分布式存储系统的存储容量，是指从系统中任意 k 个节点所能获得的最大文件大小。文献[4]指出，$\beta=1$ 时系统的存储容量为

$$C_{MBR}(n,k,d)=kd-\binom{k}{2} \tag{1}$$

与精确修复不同，功能修复模型中替换节点不必和失效节点完全相同，只需修复完成后系统仍然保留 MDS 属性。尽管上述存储容量从功能修复模型中得出，文献[6]进一步证明了采用精确修复的 MBR 码同样可以达到系统存储容量。

2.2　部分重复码

MDS 码的研究已经相对成熟，几乎可以满足任何符合条件的参数。所以，部分重复码的构造难点在于内部重复码设计。FR 码的实质是复制倍数为 f 的 θ 个数据块在节点上的一种排列，同时保证每个数据块的副本分别存储在不同的节点上。

定义 1　一个适用于参数为(n,k,d)分布式存储

系统的部分重复码 C=(U,M)，复制倍数为 f，是指特定 n 个子集的集合 $M=\{M_1,\cdots,M_n\}$，其中每个子集的元素均来自于符号集 $U=\{1,\cdots,\theta\}$。同时满足以下 2 个条件。

1) 每个子集的大小均为 d；

2) U 中的每个元素属于 M 中 f 个子集。

在上述定义中，每个子集 M_i 中的元素表示经过 MDS 编码后数据块的下标，这些数据块相应地存储在节点 $N_i(i=1,\cdots,n)$。可见，每个子集对应于一个存储节点。所有数据块分布在 n 个不同的节点上，且每个节点的存储容量为 d。因此，有

$$f\theta=nd \tag{2}$$

由于 2 个子集之间可能相交若干元素，不同节点之间可能存储相同的数据块。因此进一步定义 FR 码率 $R_C(k)$，是指通过任意 k 个节点所能获得不同数据块的个数。

$$R_C(k)=\min\left|\bigcup_{i\in K}M_i\right|$$

其中，$K\subset\{1,\cdots,n\},|K|=k$。

从定义可以看出，FR 码满足 MDS 属性，因此码率表示系统通过 k 个节点能解码的最大文件大小。如果一个 FR 码率大于或等于所应用的分布式系统存储容量，即 $R_C(k)\geqslant C_{MBR}(n,k,d)$，那么称该 FR 码是最优的。

2.3 FR 码实例

假设 $X=(X_1,\cdots,X_5)\in F_q^5$ 表示一个包含 5 个数据块的文件，F_q^5 表示大小为 q 的有限域。经过参数为(6,5)的 MDS 编码，输出 6 个数据块 $Y_1,\cdots,Y_6$。其中 $Y_i=X_i,i=1,\cdots,5$；$Y_6=\sum_{i=1}^{5}X_i$。每个输出的编码块均复制 2 倍，将生成的数据块存储在 4 个节点上，如图 1 所示。方框中的数字表示编码块的下标，如节点 N_1 存储的 3 个数据块依次为 Y_1,Y_3,Y_5。任意 2 个节点存储的数据可以重构出原文件，因此有 k=2。当节点失效时，可以从其他 3 个节点下载数据进行修复，则 d=3。

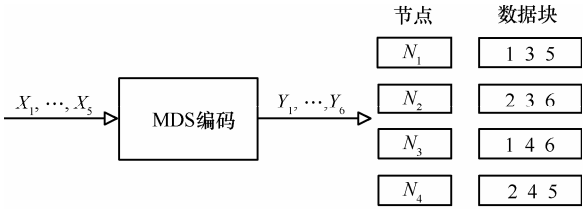


图 1 采用 FR 码的（4,2,3）分布式存储系统

可以看出，任意 2 个节点包含一个相同的数据块。随机连接 2 个节点能够得到 5 个不同数据块，根据定义该 FR 码率为 5。此外，由式(1)可得系统存储容量为 5，因此所设计的 FR 码是最优的。

3 GDDBFR 码设计

3.1 可分组设计

定义 2 设 v 与 λ 为给定的正整数，S 与 T 为给定的正整数集。设 $D=(V,G,A)$为有限关联结构，其中，V 为一个 v 元集，G 构成 V 的一个划分。V 中的元素叫点（point），A 中的元素叫做区组（block），G 中的元素叫做组（group）。若以下条件满足：

1) 对任意 $B\in A$，都有 $|B|\in S$；

2) 对任意 $G\in G$，都有 $|G|\in T$；

3) 对任意 $B\in A$ 与 $G\in G$，都有$|B\cap G|\leqslant 1$；

4) V 中的任意一对属于不同组的元素恰好同时包含在 λ 个区组中。

那么，称 D 为一个可分组设计（group divisible design）或者 GD 设计，记作 GD(S, λ, T; v)。如果每个组的容量相同，每个区组大小相同，即 $S=\{s\}$，$T=\{t\}$，把 GD($\{s\}$, λ, $\{t\}$; v)简记作 GD(s, λ, t; v)，并称其为均匀(uniform)可分组设计。若对 $1\leqslant i\leqslant h$，G 包含 m_i 个容量为 t_i 的组，且 $v=\sum_{i=1}^{h}m_it_i$，则称 D 为一个型（type）为 $t_1^{m_1}t_2^{m_2}\cdots t_h^{m_h}$ 的 GD 设计。

设 $D=(V,G,A)$为一个 GD 设计，令 $P\subset A$，若 V 中的每一点都正好与 P 中唯一的区组相关联，则称 P 为一个平行类(parallel class)。如果一个 GD(s, λ, t; v)的全部区组能够划分成若干平行类，则称为一个可分解 GD 设计。

当 $v=st$ 时，GD(s, λ, t; v)叫做 λ 重横截设计(λ-fold transversal design)，记作 TD(s, λ; t)，简称 TD 设计。如果参数 λ=1，横截设计 TD(s, 1; t)的存在性等价于相互正交的拉丁方的存在性。若每个组均只包含一个点，即 t=1，则该 TD 设计相当于一个 Steiner 系[13]。虽然 Steiner 系是一种特殊的 GD 设计，但并不是所有的 GD 设计都属于 Steiner 系。

对于均匀 GD(s, λ, t; v)，V 中的每一点都属于特定数目的区组（记为 r），称为此设计的重复数，并满足下述参数关系

$$r=\lambda(v-t)/(s-1) \tag{3}$$

同时，用 b 表示该 GD 设计包含的区组总数，

从而有式(4)成立。

$$b = \lambda v(v-t)/s(s-1) \tag{4}$$

例 1　设 V= {1, 2,⋯, 6}，3 个大小相等的组分别取为 G: {1, 2}，{3, 4}，{5, 6}，所生成的区组为 {1, 3, 5}, {2, 3, 6}, {1, 4, 6}, {2, 4, 5}、则(V, G, A)构成一个均匀 GD(3, 1, 2; 6)。其中，任一给定的点属于 2 个不同的区组。因此，r=2，b=4。

3.2　构造方法

为了构造能够达到随机访问模式下的系统存储容量的 FR 码，GD 设计中应取 λ=1，任意一对属于不同组的点恰好同时包含在唯一的区组中。并且设计中节点存储容量相同，这里采用均匀 GD 设计。一个 GD 设计可能存在若干同构，如例 1 中若取 $G=\{\{1,3\},\{2,5\},\{4,6\}\}$，将得到不同的区组。本文仅考虑一种具体的设计（对应于特定的分组），相应的构造方法同样适用于所有其他同构设计。

GDDBFR 码构造　取一个给定的 GD(s, 1, t; v)，其中 $t\geqslant 2$。该设计全体区组为 $A=\{B_1,\cdots,B_b\}$，则可以生成一个 FR 码 C=(V, A)。这里，所构造的 FR 码参数为：$\theta = v$，f = (v−t)/(s−1)。相应存储系统的节点规模为 $n = v(v-t)/s(s-1)$，每个节点可以存储 $d = s$ 个数据块。

其中，复制倍数 f 和系统节点数 n 可以分别从式(3)和式(4)求出。采用例 1 中的均匀 GD(3, 1, 2; 6)设计，构造出的 FR 码如图 1 所示。该系统可以容纳一个节点失效且保证精确无编码地数据再生。如果节点 N_1、N_2 同时失效，那么需重构原文件才能得到编码块 Y_3。一般地，对于一个复制倍数为 f 的 FR 码，系统可以承受 f−1 个节点失效而不丢失精确无编码修复特性，此时系统中所有数据块都至少存在一个备份。下面给出一个允许多节点失效的 GDDBFR 码构造实例。

例 2　令元素集 V= {1, 2,⋯, 8}, G = {{1, 2}, {3, 4}, {5, 6}, {7, 8}}，从而可以得到 8 个区组

{1, 3, 5}, {2, 4, 6}, {1, 4, 7}, {2, 3, 8}

{1, 6, 8}, {2, 5, 7}, {3, 6, 7}, {4, 5, 8}

取一个包含 6 个数据块的文件，记为 $X_1,\cdots,X_6$。经过参数为(8,6)的 MDS 编码，输出 8 个编码块 $Y_1,\cdots,Y_8$。运用所述均匀 GD 设计，系统中的数据块存储方式如图 2 所示。

如果一个用户连接 3 个节点，至少可以获得 6 个不同的数据块。例如，连接节点 N_1, N_2, N_3，将得到 7 个不同的数据块。若从 N_1、N_3 以及 N_7 3 个节点下载数据，可以取得 6 个不同的数据块。根据 FR 码率的定义，则有 $R_C(3) = 6$。此外，由式(1)得该系统的存储容量为 C_{MBR} = 6，所设计的 GDDBFR 码是最优的。

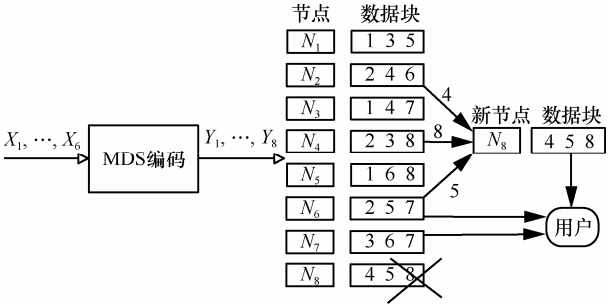


图 2　参数为(8, 3, 3)的存储系统，所采用 FR 码的复制倍数为 3

文献[10]指出，如果构造过程中所使用的设计能够分解成 ρ 个平行类，则可以通过选取其中任意 $f(\leqslant\rho)$ 个来生成复制倍数为 f 的 FR 码。每个平行类包含了符号集中的所有元素，因此只要系统中存在一个完整的平行类，那么节点修复就可以正常进行。相应地，如果 GDDBFR 码构造过程中所应用的 GD 设计是可分解的，则可以灵活地选择编码块的复制倍数和系统中节点规模。

例 3　考虑一个均匀 GD(3, 1, 3; 9)，其中 3 个分组依次为{1, 2, 3}, {4, 5, 6}, {7, 8, 9}。该设计所生成的 9 个区组可以分成 3 个平行类（每一行的区组构成一个平行类）

{1, 4, 7}, {2, 5, 9}, {3, 6, 8}

{1, 6, 9}, {2, 4, 8}, {3, 5, 7}

{1, 5, 8}, {2, 6, 7}, {3, 4, 9}

如果选取其中任意 2 个平行类，由构造方法可以得到一个复制倍数 f=2 的 GDDBFR 码，适用于参数为(6,3,3)的分布式存储系统；如果取 3 个平行类，则可以生成一个复制倍数 f=3 的 GDDBFR 码，对应存储系统参数为(9,3,3)。这种灵活的参数选取，给系统设计提供了很大的便利。

GDDBFR 码涵盖了 FR 码的所有特性。每个数据块的复制倍数一致，同时系统每个节点的存储容量相同。值得注意的是，与传统随机访问模式不同，GDDBFR 码采用基于表格（table-based）的修复方式。具体地说，修复表格指明了每个特定失效节点可选择的修复方案。如例 2 所示，如果节点 N_8 失效，可以通过节点 N_2、N_4、N_6 来进行修复，而非节点 N_1、N_2 及 N_3。对于一个应用于参数为(n, k, d)分布式存储系

统的GDDBFR码，复制倍数为f，其修复表格生成过程如下。

1) 建立数据块修复方式：记录编码后的数据块修复方式。由于复制倍数为f，每个数据块均存在f种修复选择方案。

2) 创建单节点修复方案：每个节点的修复方案就是该节点所包含的d个数据块可选的修复方式，所以时间复杂度为$O(df)$。

3) 构造完整的修复表格：遍历系统中所有节点，生成n个节点的修复表格。因此，整个过程时间复杂度为$O(ndf)$。

实际存储系统部署中通常包含一个追踪服务器（tracker server），用于记录系统元数据。因此，可以将修复表格信息写入元数据，便于失效修复的快速访问读取。就降低修复过程的复杂度而言，建立和维护节点修复表格的代价是值得的[9]。

3.3 构造方法比较

目前，FR 码构造方式主要基于组合设计理论（combinatorial designs），如正则图[9]、Steiner 系[9]、射影几何[11]及可分解设计[12]。当系统中节点数n和节点容量d的乘积为偶数且复制倍数为f=2时，正则图可以提供相应的 FR 码设计。Steiner 系适用于允许多节点失效的系统，其中任意一对不同的数据块恰好存储在唯一的节点上。然而，构造参数在 Steiner 系之外的 FR 码设计是一个开放性问题[9]。

如之前所述，Steiner 系是一种特殊的 GD 设计，但并不是所有的均匀 GD 设计都属于 Steiner 系。对于一个区组大小为 α，元素总数为 v 的 Steiner 系 $S(m,\alpha,v)$，满足任意一个 m 元子集均包含在唯一的区组中。根据 GD 设计的定义，所有属于同一组的点不可能出现在同一个区组。因此，基于 GD 设计的 FR 码构造不同于基于 Steiner 系的设计，参数在 Steiner 系之外的 FR 码是广泛存在的。

如果系统的节点数远远大于 FR 码的复制倍数，射影几何（两偶图、相互正交拉丁方等）能够提供这样大规模的 FR 码构造，同时系统还可以方便地进行扩展，不需频繁的重配置。并且，若构造过程中所采用的设计是可分解的，FR 码的复制倍数可以灵活地进行选择。

与现有 FR 码构造方式相比，GDDBFR 码的构造方式更加简洁直观。GD 设计不需要生成特定的图或者抽象几何，降低了 FR 码构造过程的复杂度。一旦元素集 V 及分组 G 确定（如果该设计确实存在），就可以快速地生成相应的区组，并且应用于 GDDBFR 码构造中。

4 GDDBFR 码性质及研究

GDDBFR 码在 MBR 码的基础上构造的，满足 MDS 属性，同时节点失效采用精确修复模式。本节将进一步讨论 GDDBFR 码的其他特性。

4.1 最优性

前面已经介绍了 GDDBFR 码的具体构造方法，且相关构造实例表明所设计的 FR 码是最优的。进一步观察可以发现，复制倍数为 2 的 GDDBFR 码与文献[9]中基于正则图的 FR 码构造方式相似。例如，图 1 中 GDDBFR 码等价于图 3 中的 FR 码设计。

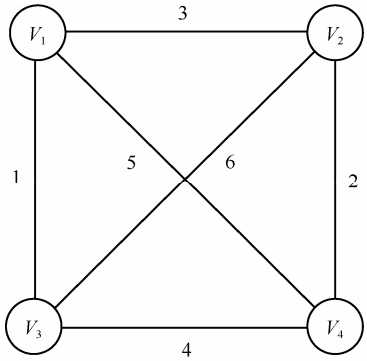


图 3 基于正则图的 FR 码构造

每个顶点代表一个存储节点，每条边表示一个特定的数据块，其中边上的数字是编码块下标。每个节点存储其关联边所表示的特定数据块，如节点 4（对应于顶点 V_4）存储 3 个数据块 Y_2、Y_4、Y_5。节点发生失效时，可以从该节点相邻的节点下载数据进行修复。

当复制倍数 f=2 时，每个数据块存储在 2 个不同的节点上。由 GDDBFR 码具体构造方法可知，所应用 GD 设计的重复数为 r=2。结合式(3)和式(4)，可得 b=2v/d，从而有 $\theta = nd/2$。因此，就数据块的标识而言，基于 GD 设计和正则图的构造是等价的，所生成的 FR 码均是最优的[9]。可以看出，尽管参数为 f=2 的 GDDBFR 码本质上与基于正则图的 FR 码相同，本文的设计省去了正则图的生成、数据块的分配等操作，构造方式更加简洁直观。此外，如例 2 所示，复制倍数为 3 的 GDDBFR 码同样也是最优的。综上所述，可得如下定理。

定理 1 复制倍数为 $f \geqslant 2$ 的 GDDBFR 码均是最优的。

证明 从 GD 设计的定义可知，点集 $\boldsymbol{V}$ 中每一

对不同的点可能存在 2 种情况：属于同一个组或者 λ 个区组，任意一对来自同一组的点不可能出现在相同的区组中。给定一个均匀 GD(s, 1, t; v)，任意一对属于不同组的点恰好同时包含在唯一的区组中，2 个不同的区组相交于不多于一点。结合具体构造方法，当连接 k 个不同节点时，最多可以得到 $\binom{k}{2}$ 个重复的数据块。因此，GDDBFR 码可以达到系统在随机访问模式下的存储容量 C_{MBR}。

证毕。

从以上定理可以看出，GDDBFR 码率可以超出系统存储容量，即 $R_C(k) > C_{MBR}$。例如，随机选取例 3 中 GD 设计的 2 个平行类，可以构造出一个适用于参数为(6,3,3)存储系统的 FR 码，复制倍数为 f=2。任意连接系统中 3 个节点，均可以得到 7 个不同的数据块，则有 $R_C(3) = 7$。而由式(1)可知，该系统存储容量仅为 6。因此，采用 GDDBFR 码的系统所能存储的最大文件可以超出随机访问模式下的系统存储容量。这种存储增益来源于修复条件的放宽，即 GDDBFR 码的修复方式是基于表格的。

4.2　存在性

GDDBFR 码与均匀 GD 设计的存在性紧密相关，选择适当的 GD 设计可以构造出所需参数的 FR 码。作为组合设计理论的重要分支，目前 GD 设计的研究已经相对成熟，参数在一定范围内的 GD 设计也基本确定。特别地，文献[13]给出了均匀 GD 设计存在的必要条件。

引理 1　GD(s, λ, t; v)设计存在的必要条件是

1) $v \geqslant st$;

2) $\lambda(v-t) = 0(\mathrm{mod}(s-1))$;

3) $\lambda v(v-t) = 0(\mathrm{mod}(s-1)s)$ 。

根据均匀 GD(s, λ, t; v)的定义，设计中组的数目应不小于区组的容量（如条件(1)所示），否则找不到满足条件的点来构成一个区组。并且，引理中条件(2)和条件(3)分别保证了 GD 设计的重复数以及区组数为正整数。目前，区组容量较小的均匀 GD 设计的存在性问题已经完全解决。对于 GD(3, λ, t; v)设计，上述的必要条件也是充分的。同时，除去 GD(4, 1, 2; 8)和 GD(4, 1, 6; 24)这 2 个设计不存在之外，GD(4, λ, t; v)存在的必要条件也是充分的。此外，区组大小 s=5 时，均匀 GD 设计存在性问题的研究也已经取得了重要进展。以上结果结合 GDDBFR 码的具体构造方法，可得如下定理。

定理 2　GDDBFR 码存在的必要条件是当 λ=1 时，引理 1 中的 3 个条件均满足。

表 1 列出了一些点数在 15 以内的均匀 GD 设计存在实例，其中区组大小 $s \in \{3,4\}$。同时，表中还给出了相应的 GDDBFR 码设计($f \in \{2,\cdots,6\}$)及具体系统参数。可以看出，GDDBFR 码可以在很大范围内选择构造参数。

表 1　GDDBFR 码的存在实例

点集 v (符号 θ)	组容量 s (节点容量 d)	型 (type)	重复数 r (复制倍数 f)	区组数 b (节点数 n)
6	3	2^3	2	4
8	3	2^4	3	8
9	3	3^3	3	9
12	3	2^6	5	20
12	3	4^3	4	16
12	4	3^4	3	9
14	4	2^7	4	14
15	3	3^5	6	30
15	3	5^3	5	25
15	4	3^5	4	15

4.3　节点修复选择度

对 MDS 码来说，系统中发生节点失效时可以通过从其他 n−1 个可用节点中随机选择 k 个下载数据，重构出原始文件再进行编码修复。因此，对于任一节点失效，MDS 码存在 $\binom{n-1}{k}$ 种修复方案。这种指明了节点失效修复可选的方案数，称为该节点的修复选择度。

与随机访问模式不同，GDDBFR 码采用基于表格的修复方式，其中表格给出了节点具体的修复方案。由于每个数据块的 f 个副本分布在不同的节点并且一对不同的数据块存储在唯一的节点上，当一个节点失效时，可以连接其他与该节点存储相同数据块的节点，下载所丢失数据块的副本构造出替换节点。由此可知，给定一个容量为 d 的存储节点，系统存在$(f-1)^d$ 种失效修复方案。图 4 给出了复制倍数为 3 的 GDDBFR 码的节点修复选择度与存储容量 d 之间的关系。

从图 4 中可以看出，虽然 GDDBFR 码的修复方式基于表格，其节点修复选择度依然可以达到很高的水平。对于复制倍数一定的 GDDBFR 码，节点修复选择度随着节点存储容量 d 呈指数倍增长。

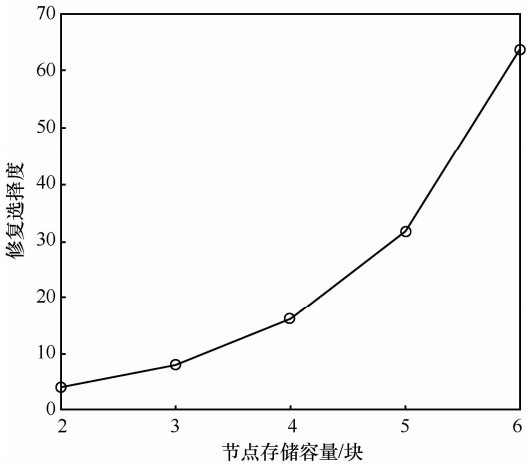


图 4 节点修复选择度与存储容量 d 之间的关系，其中 f=3

4.4 实验结果及分析

采用业内流行的 Hadoop 分布式文件系统，实现了本文提出的 GDDBFR 码，完成了文件的编解码以及失效恢复功能。实验中系统服务器的 CPU 配置为 Intel(R) Xeon(R) E5-2609 2.40 GHz，内存大小为 24 GB。采用普通 PC 机（CPU 为 AMD A8-5600k 3.0 GHz，4 GB 内存）作为数据存储节点，配置了相同的实验环境，并且实验过程中每个节点无任何其他作业。在节点存储容量相同的的条件下，从不同的(n,k)值比较分析了 GDDBFR 码与经典的 RS 码、MBR 码在修复时间上的差异。

实验 1 设定节点数量 n=9，任意 6 个节点存储的数据可以重构出原文件。同时，实验中采用复制倍数为 2 的 GDDBFR 码，分别在节点存储容量为 100 MB、200 MB、300 MB 的情况下测试 3 种编码的单节点失效修复时间。在相同条件下运行多次取平均测试值，实验结果如图 5 所示。从图中可以看出，与 RS 码和 MBR 码相比，GDDBFR 码大幅降低了节点失效恢复时间。

实验 2 设定节点数量 n=16，且 k=10。实验中采用复制倍数为 f=3 的 GDDBFR 码，结果如图 6 所示。当节点存储容量增加时，GDDBFR 码在修复时间上的优势更为明显。

传统的 RS 码节点修复过程中需恢复出原始文件，重新编码再将生成的编码块存储到替换节点，因此修复时间比较长。对于最小带宽再生 MBR 码，参与修复的节点对存储的数据进行线性运算，再将组合后的数据块传送到替换节点。该节点对接收的所有数据块进一步整合，进而恢复出失效的数据。整个过程涉及大量的有限域运算，增加了修复时间。当检测出节点失效时，系统首先判定具体哪一个节点失效，根据 GDDBFR 码修复表格（存储在系统元数据中）确定修复方案。同时连接方案中指定的可用节点，下载相应数据块并直接存储到替换节点中。可见，整个修复过程仅仅涉及文件读取工作，并不引入其他复杂运算。虽然在一定程度上增加了系统的冗余量，实验结果表明，GDDBFR 码可以大幅降低失效修复时间。

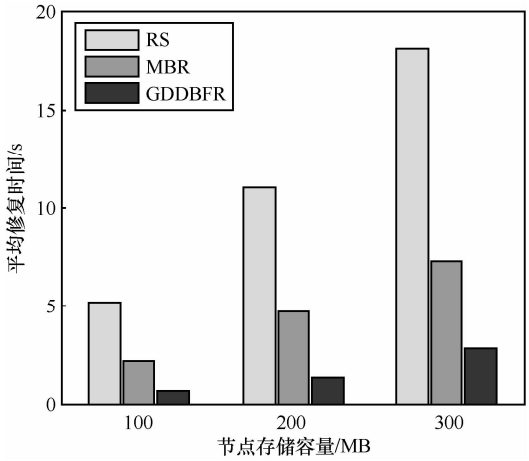


图 5 参数为(9, 6)的 3 种编码修复时间对比

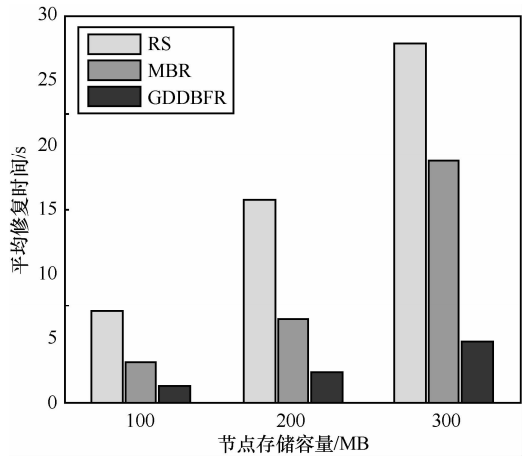


图 6 参数为(16, 10)的 3 种编码修复时间对比

5 结束语

本文提出了一种新型的基于可分组设计的部分重复码构造——GDDBFR 码，可以在很大范围内选择设计参数。相比现有的构造方式，基于 GD 设计更加简洁直观。并且，GDDBFR 码可以达到随机访问模式下的系统存储容量，实现了理论上的最优。理论分析表明，虽然 GDDBFR 码采用基于表

格的修复方式，但通常具有广泛的修复选择方案。此外，相比传统的 RS 码和再生码，GDDBFR 码可以很大程度上减少节点修复时间，这使其在实际大规模存储系统中具有可观的应用前景。为了进一步使本文提出的 FR 码设计具有更好的实用性，下一步将引入大数据平台进行分析。

参考文献：

[1] GHEMAWAT S, GOBIOFF H, LEUNG S. The Google file system[A]. The 19th ACM Symposium on Operating Systems Principles[C]. Lake George, New York, USA, 2003. 29-43.

[2] DIMAKIS A G, RAMCHANDRAN K, WU Y, *et al*. A survey on network codes for distributed storage[J]. Proceedings of the IEEE, 2011, 99(3): 476-489.

[3] FAN B, TANTISIRIROJ W, XIAO L, *et al*. Diskreduce: Replication as a Prelude to Erasure Coding in Data-Intensive Scalable Computing[R]. Parallel Data Laboratory, Carnegie Mellon University, Pittsburgh, PA, USA, 2011.

[4] 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.

[5] RASHMI K V, SHAH N B, KUMAR P V. Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction[J]. IEEE Transactions on Information Theory, 2011, 57(8): 5227-5239.

[6] RASHMI K V, SHAH N B, KUMAR P V, *et al*. Explicit construction of optimal exact regenerating codes for distributed storage[A]. The 47th Annual Allerton Conference on Communication, Control, and Computing[C]. Monticello, IL, USA, 2009.1243-1249.

[7] RASHMI K V, SHAH N B, KUMAR P V, *et al*. Explicit and optimal exact-regenerating codes for the minimum-bandwidth point in distributed storage[A]. IEEE International Symposium on Information Theory Proceedings[C]. Austin, TX, USA, 2010. 1938-1942.

[8] VENKATESAN V. Fast Rebuilds in Distributed Storage Systems Using Network Coding[R]. Zurich Research Laboratory, IBM Research GmbH, Zurich, 2009.

[9] ROUAYHEB S E, RAMCHANDRAN K. Fractional repetition codes for repair in distributed storage systems[A]. The 48th Annual Allerton Conference on Communication, Control, and Computing[C]. Allerton, IL, USA, 2010. 1510-1517.

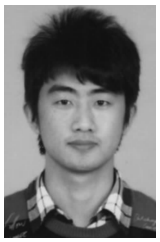
[10] PAWAR S, NOORSHAMS N, ROUAYHEB S E, *et al*. Dress codes for the storage cloud: simple randomized constructions[A]. IEEE International Symposium on Information Theory Proceedings[C]. St. Petersburg, Russia, 2011. 2338-2342.

[11] KOO J, GILL J. Scalable constructions of fractional repetition codes in distributed storage systems[A]. The 49th Annual Allerton Conference on Communication, Control, and Computing[C]. Monticello, IL, USA, 2011. 1366-1373.

[12] OLMEZ O, RAMAMOORTHY A. Repairable replication-based storage systems using resolvable designs[A]. The 50th Annual Allerton Conference on Communication, Control, and Computing[C]. Monticello, IL, USA, 2012. 1174-1181.

[13] COLBOURN C J, DINITZ J H. Handbook of Combinatorial Designs, Second Edition[M]. Chapman and Hall/CRC, 2006.

作者简介：



朱兵（1990-），男，安徽庐江人，北京大学深圳研究生院博士生，主要研究方向为网络编码、分布式存储系统等。

李挥（1964-），男，广东潮州人，博士，北京大学深圳研究生院教授、博士生导师，主要研究方向为三网合一流媒体云计算网络视频关键技术、下一代网络体系结构、网络路由和宽带交换结构、网络编码理论及其应用、嵌入式系统开发。

陈俊（1990-），男，广东揭阳人，北京大学深圳研究生院硕士生，主要研究方向为分布式存储编码及实现。

侯韩旭（1987-），男，安徽砀山人，北京大学深圳研究生院博士生，主要研究方向为网络编码、分布式存储系统等。

周泰（1989-），男，河北邢台人，北京大学深圳研究生院硕士生，主要研究方向为分布式存储系统。