Journal of Computer Applications
计算机应用,2020,40(10):2942 - 2950
ISSN 1001-9081
CODEN JYIIDU
2020-10-10
http://www.joca.cn

文章编号:1001-9081(2020)10-2942-09
DOI:10.11772/j.issn.1001-9081.2020010127

分布式存储系统中的低修复成本纠删码

张　航,刘善政,唐　聃*,蔡红亮
(成都信息工程大学 软件工程学院,成都 610225)
(*通信作者电子邮箱 tangdan@foxmail.com)

摘　要:纠删码技术是分布式存储系统中典型的数据容错方法,与多副本技术相比,能够以较低的存储开销提供较高的数据可靠性;然而,纠删码修复成本过高的特点限制了其应用。针对现有纠删码修复成本高、编码复杂和灵活性差的问题,提出一种编码简单的低修复成本的纠删码——旋转分组修复码(RGRC)。RGRC首先将多个条带组合成条带集,然后利用条带之间的关联关系对条带集内的数据块进行分层旋转编码,以此得到相应的冗余块。RGRC大幅度地减少了单节点修复过程中所需要读取和传输的数据量,从而能节省大量的网络带宽资源。同时RGRC在解决单节点修复成本高的问题时,依然保留着较高的容错能力,且为满足分布式存储系统的不同需求,可以灵活地权衡系统的存储开销和修复成本。在分布式存储系统中进行的对比实验分析结果展示,与其他常用的RS(Reed-Solomon)码、LRC(Locally Repairable Codes)、basic-Pyramid、DLRC(Dynamic Local Reconstruction Codes)、pLRC(proactive Locally Repairable Codes)、GRC(Group Repairable Codes)、UFP-LRC(Unequal Failure Protection based Local Reconstruction Codes)相比,RGRC只需要增加少量的存储开销,就能降低单节点修复14%~61%的修复成本,同时减少14%~58%的修复时间。

关键词:分布式存储系统;数据修复;单节点修复;纠删码;低修复成本
中图分类号:TP3　　**文献标志码:**A

Erasure code with low recovery-overhead in distributed storage systems

ZHANG Hang, LIU Shanzheng, TANG Dan*, CAI Hongliang
(*School of Software Engineering*, *Chengdu University of Information Technology*, *Chengdu Sichuan* 610225, *China*)

Abstract: Erasure code technology is a typical data fault tolerance method in distributed storage systems. Compared with multi-copy technology, it can provide high data reliability with low storage overhead. However, the high repair cost limits the practical applications of erasure code technology. Aiming at problems of high repair cost, complex encoding and poor flexibility of existing erasure codes, a simple-encoding erasure code with low repair cost — Rotation Group Repairable Code (RGRC) was proposed. According to RGRC, multiple strips were combined into a strip set at first. After that, the association relationship between the strips was used to hierarchically rotate and encode the data blocks in the strip set to obtain the corresponding redundant blocks. RGRC greatly reduced the amount of data needed to be read and transmitted in the process of single-node repair, thus saving a lot of network bandwidth resources. Meanwhile, RGRC still retained high fault tolerance when solving the problem of high repair cost of a single node. And, in order to meet the different needs of distributed storage systems, RGRC was able to flexibly weigh the storage overhead and repair cost of the system. Comparison experiments were conducted on a distributed storage system, the experimental analysis shows that compared with RS (Reed-Solomon) codes, LRC (Locally Repairable Codes), basic-Pyramid, DLRC (Dynamic Local Reconstruction Codes), pLRC (proactive Locally Repairable Codes), GRC (Group Repairable Codes) and UFP-LRC (Unequal Failure Protection based Local Reconstruction Codes), RGRC can reduce the repair cost of single node repair by 14%-61% through adding a small amount of storage overhead, and reduces the repair time by 14%-58%.

Key words: distributed storage system; data recovery; single-node repair; erasure code; low repair cost

0　引言

随着信息技术的发展,人们产生的数据越来越多,各企业为了满足人们的数据需求,纷纷搭建企业自己的数据中心并使用分布式存储系统来存储海量数据。据英特尔预测,全球数据总量在2020年将达到44 ZB(1 ZB=10^9TB),而中国产生的数据量将达到8 ZB,大约占据全球总数据量的1/5[1]。在商业存储领域,数据增长极大地推动了分布式存储系统的发展。为了保障分布式存储系统海量数据的安全性和可靠性,常采

收稿日期:2020-02-13;**修回日期:**2020-04-16;**录用日期:**2020-04-28。　**基金项目:**四川省科技计划项目(2020YFG0150);四川人工智能重大专项(2018GZDZX0030);四川省科技成果转移转化示范项目(2018CC0093)。

作者简介:张航(1995—),男,四川乐山人,硕士研究生,主要研究方向:编码理论、分布式存储;　刘善政(1995—),男,河南濮阳人,硕士研究生,主要研究方向:图像秘密共享;　唐聃(1982—),男,四川绵阳人,教授,博士,CCF会员,主要研究方向:编码理论、分布式存储;　蔡红亮(1983—),男,山西临汾人,讲师,博士,主要研究方向:编码理论、视觉密码。

用多副本[2]和纠删码[3]等容错技术。

多副本技术是将原始数据复制多份，并分别存储在不同的存储节点上。Hadoop 分布式文件系统（Hadoop Distributed File System，HDFS）[4]以及 Ceph 分布式存储[5]系统均采用了多副本中常见的三副本。多副本技术操作简单，易于实施，但是该方法存储效率低。以常见的三副本为例，该方法将原始数据复制成三份，保存在三个不同的存储节点上，磁盘存储利用率仅有 1/3，同时数据中心构建成本高，存储成本过大，造成严重的资源浪费。

纠删码容错技术是一种数据保护方法，它将数据分割成片段，利用编码算法生成冗余数据块存储在不同的位置，比如磁盘、存储节点上。纠删码技术相比多副本技术能以更低的存储开销存储更多的数据，Hadoop3.0 以及 Ceph 分布式存储系统均采用了纠删码技术作为系统容错技术之一，同时微软的存储系统、亚马逊的 AWS（Amazon Web Services）[6]等也都支持纠删码技术作为容错技术。

然而，纠删码过高的修复成本极大地限制了其技术在分布式存储系统中的实用性。纠删码在修复失效的数据块时，需要在多个节点读取数据并传输，会占用大量的网络带宽。众所周知，带宽资源一直是分布式存储系统中的稀缺资源。占用较高的网络带宽，会降低分布式存储系统的数据读取速度，从而影响到整个系统的稳定性。

纠删码的修复成本主要由其自己的特性决定，设计纠删码的编码结构能从根本上降低修复成本[7]。目前，根据编码结构的不同，关于低修复成本的纠删码的主要分为两类：分组码和再生码。分组码是将一个条带的数据块进行分组，利用组内的原始数据块生成局部冗余块，在原始数据块失效时，利用组内的局部冗余块来恢复数据块，能有效降低数据修复时需要的修复成本；再生码是通过适当增加冗余并且使新生节点从尽量多的节点下载数据块来降低修复时需要传输的数据量[8]。

Huang 等[9-10]提出了 LRC（Locally Repairable Codes）、Pyramid 等典型的层次分组码。LRC 和 Pyramid 码都是通过增加局部冗余的方式来降低数据块修复成本，但它们的修复成本依然过高。Facebook 采用了针对 LRC 和 Pyramid 上作出改进的 LRCs[11]和 EXPyramid[12]层次分组码。LRCs 主要针对全局冗余块再编码了一个局部冗余块，能降低全局冗余块的修复成本；EXPyramid 码是将 Pyramid 码的编码数据块改为阵列结构，同时在横向和纵向两个方向编码局部冗余块，进一步降低了数据块的修复成本。林轩等[13]继续在 LRC、Pyramid 的基础上提出了 GRC（Group Repairable Codes）层次分组码，GRC 不仅将条带进行分组生成局部冗余块，而且同时为编码条带生成局部冗余块，在数据块失效时，利用两种局部冗余块参与恢复，不需要条带中所有数据块参与修复，进一步降低了修复成本。但这三种改进方法都需要消耗大量的存储空间，并不适用大规模的分布式存储系统。Meng 等[14]提出了一种新的分组码 DLRC（Dynamic Local Reconstruction Code），通过利用参数动态地调整存储开销和重构开销的平衡；但在满足多节点容错的情况下，单节点修复成本过大。Miyamae 等[15]提出了交叉分组码 SHEC（Shingled Erasure Code）。SHEC 是将条带上的数据块逻辑上进行重叠并分组，进而编码出局部冗余块，在数据块失效时，利用较少的组进行局部修复，从而降低修复成本；但 SHEC 重叠编码难以维持较低的修复成本和较高的容错能力，依然不适用于现有分布式存储系统。

再生码的研究主要关注 MBR（Minimun Bandwidth Rrgenerating）码[16]和 MSR（Minimun Storage Regenerating）码[17]。MSR 码主要拥有最低的存储开销，MBR 码主要拥有最低的数据修复成本。在 2011 年，Rashmi 等[18]首次利用矩阵乘的方法，用统一的方法构造出 MBR 码和 MSR 码。Liu 等[19]提出了再生码 GFR（General Functional Regenerating），通过设定一个可自由调节的参数来实现分布式存储系统的修复成本和存储成本之间的平衡，同时使用一种启发式算法来找到修复单节点失效数据的最小修复成本。Liu 等[20]同时提出了最小再生码 Z 码，Z 码把置换矩阵作为元矩阵，组合构造出最优修复成本的生成矩阵，同时利用矩阵的张量乘积，迭代出任意参数下的生成矩阵，并依然保持最优修复成本。Xie 等[21]提出了 AZ-CODE（Availability Zones Codes），AZ-CODE 结合了 MSR 码和 LRC 的编码方式，利用其混合优点，能有效降低修复成本。

再生码虽然可以大幅度地减少修复时的数据传输量，但在数据修复过程中，参与修复的节点需要把自己存储的所有数据都读取出来进行组合，会消耗大量的读取成本。同时，再生码编码复杂度高、编码耗时长等因素都不适用于现有的分布式存储系统。

根据数据中心对分布式存储系统故障的调查研究报告显示，其中 99.75% 的故障来源于分布式存储系统单节点失效[22]。因此针对现有的分布式存储系统中频繁的单节点失效且纠删码修复成本过高的问题，提出了一种低修复成本纠删码——旋转分组修复码（Rotation Group Repairable Codes，RGRC），并验证了低成本修复性。RGRC 是将条带组合成条带集，对条带集内的数据块进行分层旋转编码来减少大容量单节点失效的修复带宽。同时 RGRC 在解决单节点修复成本高的问题时，依然保留着较高的容错能力，且为满足分布式存储系统的不同需求，可以灵活地权衡其存储开销和修复成本。

1　分布式存储系统中纠删码的相关概念

分布式存储系统使用纠删码作为容错技术，能消耗更少的存储空间，从而减少存储硬件的使用，大幅降低存储中心的构建成本。可见，纠删码空间利用率高的特点在存储海量数据时具有重大意义。

纠删码技术主要是通过纠删码算法将原始的数据块进行编码得到冗余块，并将原始数据块和冗余块一并存储起来，以达到容错的目的。用 (n,k) 表示一个纠删码，K 个原始数据块 $D_1, D_2, \cdots, D_K$ 经过计算产生了 n 个块 $P_1, P_2, \cdots, P_n$，这一过程称为编码。当 $P_1, P_2, \cdots, P_n$ 中失效的块小于该纠删码最大的容错个数时，可以选取其中剩余的块计算恢复出失效块，这一过程称为解码。相关原理如图 1 所示。

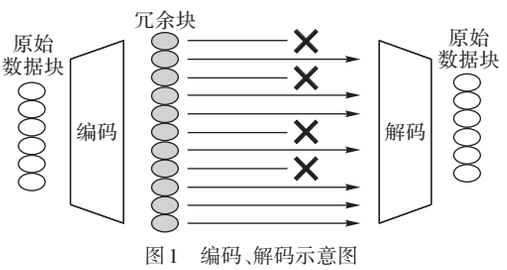

图 1　编码、解码示意图

Fig. 1　Schematic diagram of encoding and decoding

为了便于理解，现结合图2给出如下一些纠删码常用的基本概念的说明和定义：

1）极大距离可分（Maximum Distance Separable，MDS）码：这一类码在消耗相同存储开销的情况下，有最优的容错能力，因此被分布式存储系统广泛使用。对于一个参数为(n,k)的MDS纠删码来讲，容错能力为$n-k$。

2）系统性纠删码：数据块经过计算产生的块中包含原始的数据块，因良好的访问性能而成为分布式存储系统的首选。

3）容错度：纠删码可以容忍的丢失数据块个数。

4）原始数据块：用户上传的原始数据对象被系统划分后得到的块。

5）冗余块：数据块经过纠删码算法后产生的所有块。

6）条带：由多个数据块和冗余块组成，满足同一纠删码算法。

7）条带集：由多个条带组成的集合。

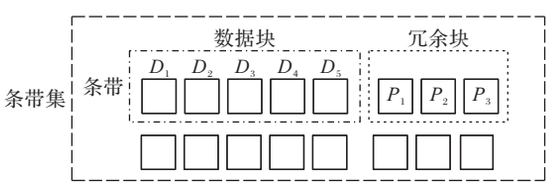

图2 数据块、冗余块、条带和条带集之间的相互关系

Fig. 2 Correlation between data blocks, redundant blocks, stripes and strip set

2 旋转分组修复码

2.1 纠删码数据修复问题定义

为了便于理解，基于文献给出纠删码数据修复问题相关的定义。

定义1 纠删码修复成本。纠删码在进行数据修复过程中需要读取的数据量。在分布式存储系统中，网络带宽资源一直是稀缺资源，在修复的过程中读取的数据量越小，传输的数据量就小，占用的网络带宽的资源就越少，从而避免了修复过程对分布式存储系统中其他任务的影响，提升了系统的稳定性。

定义2 纠删码的修复时间。纠删码在进行数据修复时所消耗的时长。分布式存储系统中，修复的快慢直接影响到系统的稳定性，如果在数据量庞大的存储系统中，一旦不能快速恢复数据，有可能导致在修复过程中因其他任务的影响而使其余数据块再次失效，因此，对于分布式存储系统中的纠删码来说，修复时间应该越小越好。

定义3 纠删码的修复率。修复率是纠删码重要的指标之一。不同失效节点数时的修复率侧面反映出了该纠删码的容错能力。

定义4 全局冗余块。条带中所有原始数据块参与编码计算得到的冗余块。

定义5 局部冗余块。条带中组内原始数据块参与编码计算得到的冗余块。

表1为常用符号说明。

表1 常用符号

Tab. 1 Common symbols

符号	含义
n	一个条带中块的总个数
m	条带中冗余块个数
k	条带中原始数据块个数
k_0	条带内小组的原始数据块个数
m_0	全局冗余块数
m_1	局部冗余块个数
$\boldsymbol{G}$	编码矩阵
D_i	条带中第i个原始数据块
P_i	条带中第i个全局冗余块
P_{ij}	条带中第i个小组中第j个局部冗余块
S	条带集中条带个数

2.2 编码算法

本文提出的RGRC是在系统型MDS码上作出改进的一种分组纠删码，所以本节先介绍系统型MDS码的编码过程，然后介绍RGRC的编码过程，最后举例演示RGRC的编码过程。

系统型MDS码的编码过程为：首先在有限域上构造编码矩阵，编码矩阵与原始数据块在有限域中执行乘法运算得到编码块。

如编码公式（1）所示，$D_1,D_2,\cdots,D_K$总共K个原始数据块与编码矩阵相乘，计算产生了$C_1,C_2,\cdots,C_n$，共n个块，其中产生的n个块中包含了k个原始数据块。

$$\boldsymbol{G}\times\boldsymbol{D}=\begin{bmatrix} g_{1,1} & g_{1,2} & \cdots & g_{1,k} \\ g_{2,1} & g_{2,2} & \cdots & g_{2,k} \\ \vdots & \vdots & & \vdots \\ g_{n,1} & g_{n,2} & \cdots & g_{n,k} \end{bmatrix}\times\begin{bmatrix} D_1 \\ D_2 \\ \vdots \\ D_k \end{bmatrix}=\begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_n \end{bmatrix} \tag{1}$$

RGRC的编码过程如下：

步骤1 根据系统性MDS码的编码方式，生成m个全局冗余块表示成$P_1,P_2,\cdots,P_m$，记为$P_i(i=1,2,\cdots,m)$。数据条带分为L个小组，记为$L_i(i=1,2,\cdots,l;0<l<k)$，每个小组包含$k_0$个原始数据块。保持纠删码前$m_0$个全局冗余块不变，为每个小组计算$m_l=m-m_0$个组内局部冗余块（其中$m_1\geqslant 3$），假设每个组内前$m_1-2$个局部冗余块称为$R$，小组$L_i$的组内局部冗余块$R_i(i=1,2,\cdots,m_l-2)$的计算方法和$P_i$一样，只是将其他数据块置0。

步骤2 多个条带组合成条带集，形成每个条带集包含S个条带，每个条带包含L个小组的结构。小组L_i的倒数第二个组内局部冗余块称为前段冗余块（Rotation Front block，RF），最后一个组内局部冗余块称为后段冗余块（Rotation Back block，RB），RF通过前段数据旋转编码方式得到，RB的通过后段数据旋转编码得到。相应的编码结构如图3所示。

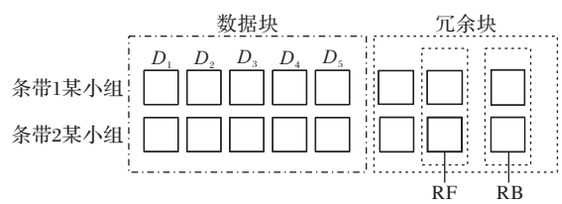

图3 RF、RB示意图

Fig. 3 Schematic diagram of RF and RB

步骤3 L_i小组内包含k_0个原始数据块，设前t个原始数据块为前段数据块，其中$0<t<k_0$，则$t_1=k_0-t$个原始数据块为后段数据块。前段数据旋转编码方式以条带集为编码单

位。取每个小组前t个原始数据块，加上前一个条带内同一位置小组内的后段数据块组合成新的数据块集合，用新的数据块集合参与编码计算，计算方法和P_i一样，只是将其他数据块置0。后段数据旋转编码方式与前段数据旋转编码类似，只不过新的数据块集合由该小组的后段数据块加上前一个条带内同一位置小组内的前段数据块组成。

所有类型的冗余块编码公式如下。

1）全局冗余块生成公式：

$$P_i = \sum_{j=1}^{k} g_{i,j} D_j;\ 0 < i \leqslant m_0 \tag{2}$$

2）组内局部冗余块生成公式：

$$R_i = \sum_{j=(l-1)\times(k/L)+1}^{l\times(k/L)} g_{i,j} D_j;\ 0 < l \leqslant L \tag{3}$$

设条带集包含S个条带，$RF_{s,l}$表示条带集中第s条带中第l小组的前段冗余块，$RB_{s,l}$表示条带集中第s条带中第l小组的后段冗余块。

3）前段、后段冗余块生成公式如下：

$$RF_{s,l} = \sum_{j=(l-1)\times(k/L)+1}^{l\times(k/L)-(k/L-t)} g_{n-1,j} D_{s,j} + \sum_{j=(l-1)\times(k/L)+t}^{l\times(k/L)} g_{n-1,j} D_{(s-1)\bmod S,j};$$
$$0 < l \leqslant L, 0 < t \leqslant (k/L), 0 < s \leqslant S \tag{4}$$

$$RB_{s,l} = \sum_{j=(l-1)\times(k/L)+1}^{l\times(k/L)-(k/L-t)} g_{n,j} D_{s,j} + \sum_{j=(l-1)\times(k/L)+t}^{l\times(k/L)} g_{n,j} D_{(s+1)\bmod S,j};$$
$$0 < l \leqslant L, 0 < t \leqslant (k/L), 0 < s \leqslant S \tag{5}$$

下面从(14, 10)的MDS码出发，构造(18, 10)RGRC演示编码过程。图4展示的(14, 10)的编码结构，$D_1, D_2, \cdots, D_{10}$为10个原始数据块，$P_1, P_2, \cdots, P_4$为经过编码计算后产生的4个全局冗余块。

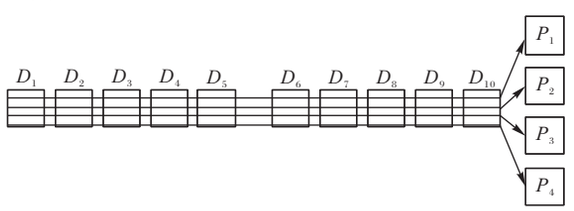

图4　(14, 10) MDS码的编码结构

Fig. 4　Encoding structure of (14, 10) MDS code

如图5所示，将MDS码的10个原始数据块分成2个小组，每个小组包含5个原始数据块，其中$L_1 = \{D_1, D_2, D_3, D_4, D_5\} L_2 = \{D_6, D_7, D_8, D_9, D_{10}\}$，保持1个全局冗余块不变，则每个小组产生的组内局部冗余块个数为3。L_1组的局部冗余块为$\{P_{21}, P_{31}, P_{41}\}$，$L_2$组的局部冗余块为$\{P_{22}, P_{32}, P_{42}\}$。

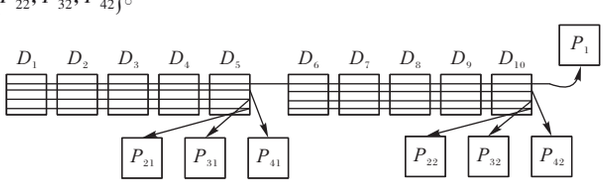

图5　条带分组后局部冗余块的编码

Fig. 5　Encoding of local redundant blocks after strip grouping

如图6所示，将4个条带组合成条带集，以第一个小组为例。设$t = 2$，则D_1、D_2列为前段数据块列，D_3、D_4、D_5列为后段数据块列。P_{31}列为前段冗余块列。按照$RF_{s,l}$生成公式构造出如图6的前段冗余块。

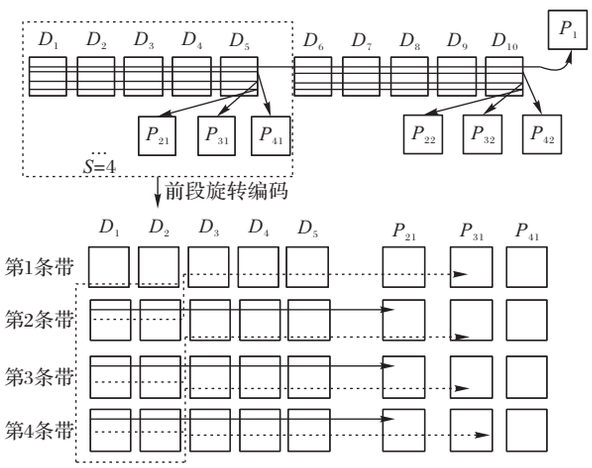

图6　前段旋转编码结构

Fig. 6　Anterior rotary coding structure

如图7所示，P_{41}为后段冗余块列。按照$RB_{s,l}$生成公式构造出如图7的后段冗余块。

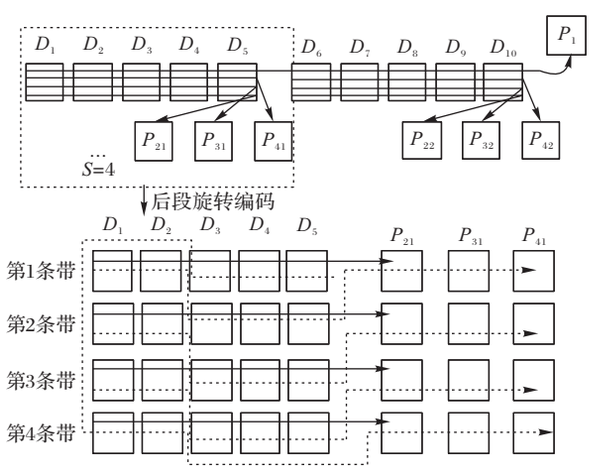

图7　后段旋转编码结构

Fig. 7　Posterior rotary coding structure

相较于MDS码，RGRC先通过对数据条带进行分组产生组内局部冗余块，组合条带构成条带集进行旋转编码，这种设计可以保障在单节点失效时快速在组内恢复，同时以条带集为单位进行恢复，大幅减少修复需要读取的数据量，从而降低修复时的修复成本和数据传输量。

2.3　解码算法

RGRC的解码过程大致为：利用现有数据组合出对应的剩余编码矩阵的逆矩阵与剩余活跃的块在有限域中进行乘法运算，从而求得丢失的数据块。

在进行解码时，单节点失效虽然在节点故障中占比大，但是多节点失效在分布式存储系统中并不少见，因此，本文针对RGRC的解码分为单节点解码和多节点解码。

2.3.1　单节点解码步骤

当失效节点为单节点时，以条带集为单位进行解码。假设失效单节点在L_i组，且S为偶数时，解码过程按如下步骤进行解码：

步骤 1　条带集以 2 个条带进行集合划分 $A_i = \{S_q, S_p\}$，(其中 $i = 1, 2, \cdots, S/2$)，S_q、S_p 为旋转编码相关联的两个条带。

步骤 2　当失效节点在前段数据块列时，恢复 S_q 条带中的失效数据块时，读取 S_q 条带中 L_i 组的前 k_0 个未失效的原始数据块和局部冗余块组成新的块集合，并保存在缓存中。恢复 S_p 条带中的失效数据块时，读取 S_p 条带中 L_i 组的未失效的前段数据块，前段旋转编码产生的 RF 块和缓存中需用到的块组成新的块集合。新的块集合和对应的剩余编码矩阵的逆矩阵相乘来恢复 2 个条带中的失效数据块。照此方法循环迭代，恢复条带集中所有的失效数据块，并进一步恢复整个节点失效的数据块。

步骤 3　当失效节点在后段数据块列时，恢复 S_q 条带中的失效数据块，读取 S_q 条带中 L_i 组的前 k_0 个未失效的原始数据块和局部冗余块组成新的数据块集合，并保存在缓存中。恢复 S_p 条带中的失效数据块时，读取 S_p 条带中 L_i 组的未失效的后段数据块，后段旋转编码产生的 RB 块和缓存中需用到的块组成新的块集合。

新的块集合和对应的剩余编码矩阵的逆矩阵相乘来恢复 2 个条带中的失效数据块。照此方法循环迭代，恢复条带集中所有的失效数据块，并进一步恢复整个节点失效的数据块。

假设失效单节点在 L_i 组，且 S 为奇数时，解码过程按如下步骤进行解码：

步骤 1　条带集以 2 个条带进行集合划分 $A_i = \{S_q, S_p\}$ (其中 $i = 1, 2, \cdots, (S-1)/2$)，$B = \{S_S\}$，$S_q$、$S_p$ 为旋转编码相关联的两个条带，S_S 为条带集中最后一个条带。

步骤 2　恢复 A_i 集合与 S 为偶数时一样。恢复 B 集合中的失效数据块时，读取 S_S 条带中 L_i 组的前 k_0 个未失效的原始数据块和局部冗余块组成新的块集合，新的块集合和对应的剩余编码矩阵的逆矩阵相乘来恢复失效数据块。

下面以 (18, 10)，$S = 4$ 的 RGRC 在 L_1 组单节点失效为例，演示其解码过程。图 8 中阴影表示数据块失效。条带集以 2 个条带进行集合划分 $A_1 = \{S_1, S_2\}$，$A_2 = \{S_3, S_4\}$。在 A_1 中恢复第 1 条带的 D_1 块，需读取第 1 个条带的 $(D_2, D_3, D_4, D_5, P_{21})$ 5 个块；恢复第 2 条带的 D_1 块，需读取第 2 个条带的 D_2 和第一条带的 P_{31} 共计两个块。A_2 集合恢复失效块方法和 A_1 一样。相较于 (14, 10) 的 MDS 码恢复 4 个条带的 D_1 列失效需读取 40 块，RGRC 只需读取 14 块数据，比 MDS 码相比修复成本降低 65%。

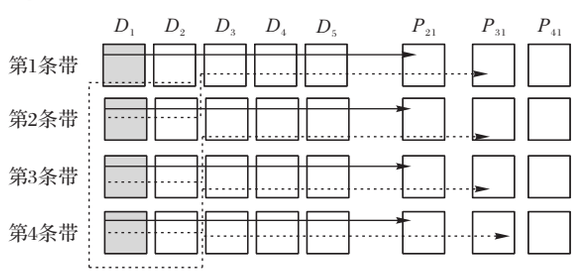

图 8　单节点失效解码

Fig. 8　Decoding of single-node failure

2.3.2　多节点解码步骤

RGRC 与 MDS 码有所区别，并不满足当失效块个数大于 $n-k$ 则不能恢复，而是某些情况下当失效块数大于 $n-k$ 依然可以恢复。基于 RGRC 的特征，本文提出了贪心策略的解码算法，该解码算法分为 2 类解码：条带集组解码、条带集全局解码。

条带集组解码是指利用和失效节点在同一小组内的条带集中未失效的块来解码恢复失效块。由于条带集中小组内块数较少，在恢复失效块时需读取的块数较少，解码成本能降到最低。当条带集组内解码无法恢复失效块时，使用条带集全局解码恢复失效块。条带集全局解码利用条带集内全局未失效块来恢复失效块，为了降低读取成本，当利用条带集全局解码恢复出某些失效块后，为可以满足条带集组解码条件时，条带集全局解码应马上转换成条带集组解码进行解码。

根据以上原则，对多节点失效的贪心策略的解码算法过程分为两个阶段：

1)条带集组解码。对于条带集中每个小组，当失效块数小于局部冗余块数，则启用条带集组解码恢复数据，当解码成功后，标记为活跃块，可以参与下一步的解码。若失效块全部解码成功完全恢复，则程序结束。

2)条带集全局解码。当失效块不能被条带集组解码恢复数据时，则启用条带集全局解码，并标记条带集中未失效块为活跃块。在全局层面，如果条带集内每个条带的活跃块大于失效块个数，则解码条带集中每个条带同一位置的失效块，并标记为活跃。同时检查可否进入第一阶段条带集组解码：如果可以，则转入第一阶段解码；若不行，程序结束。

算法 1　RGRC 贪心策略的解码算法。

输入　条带中失效位置的数组 err_loc；需要读取的块在条带集中的位置的二维数组 read_loc；读数据缓存数组 read_buff。

输出　写数据缓存数组 write_buff。

```
err_length=err_loc[].length;
for i in range(err_length)
   if check_group(err_loc[])==True
      s_groupdecode();                          //条带集组解码
      结果存于 write_buff[];
      更新 err_loc[];
      if err_loc[].length==0
         return True
      else
         s_overalldecode();                     //条带集全局解码
         结果存于 write_buff[];
         更新 err_loc[];
         if check_group(err_loc[]) == True
                                                //检查是否条带集组解码
         s_groupdecode()
         结果存于 write_buff[]
         更新 err_loc[]
      if err_loc[].length == 0 return True
end for
```

多节点解码算法根据 RGRC 的编码特点设计，条带集组解码优先，在条带集组解码失效后，再转换成条带集全局解码，一旦系统检查满足组内解码条件，再转换成条带集组解码。这种方式保证了在解码过称中读取的数据量较少，大幅减少系统的修复成本，同时将修复时间降到最低。

2.4　修复率分析

修复率是衡量一个纠删码性能的重要指标之一。在本小节中用于测试纠删码修复率的方法如下：

假设一个参数为 (n, k) 的纠删码，在分布式存储系统中，由于硬盘发生故障，失效的节点个数为 x，根据概率学，共有 $\binom{n}{x}$ 种失效方式，在这些失效方式中，能够修复的失效方式和

全部失效方式的比值则为该纠删码失效x节点的修复率。不同失效节点数时的修复率侧面反映出了该纠删码的容错能力。

表2为不同参数下RGRC在多节点失效时修复率和容错能力。从表2可以看出在数据块个数不变的情况下,冗余块个数越多,RGRC的容错能力就越强,多节点失效时的修复率也越高。此外,冗余块个数越接近数据块个数时,相应修复节点的修复率也将越高。由表2可知RGRC的容错能力以及修复率可以满足大部分分布式存储系统的需求,可以根据具体的业务情况,合理地定制不同参数下的编码方案。

表2　RGRC的修复能力

Tab. 2　Repair capability of RGRC

编码方案	数据块数	局部冗余块数	全局冗余块数	容错能力	不同节点数失效时的修复率/%			
					4个节点	5个节点	6个节点	7个节点
(17, 10) RGRC	10	6	1	7	100.00	95.92	82.30	56.43
(19, 10) RGRC	10	8	1	9	100.00	100.00	99.11	92.02
(15, 10) RGRC	10	4	1	5	89.74	63.63	0.00	0.00
(15, 8) RGRC	8	6	1	7	100.00	96.27	83.21	57.10

图9展示的分别是(14, 10)的RS(Reed-Solomon)码[23]、(10, 2, 3)的LRC、(17, 10)的basic-Pyamid码、(10, 2, 4, 3)的DLRC、(10, 2, 3)的pLRC(proactive Locally Repairable Codes)[24]、(17, 10)的GRC、(3, 3, 3), ρ = 1的UFP-LRC(Unequal Failure Protection based Local Reconstruction Code)[25]与(17, 10)的RGRC的修复率对比;在失效节点数为4时,所有的纠删码都可以完全修复;当失效节点数为5时,(14, 10)RS码数据丢失;当失效节点数为6时,(10, 2, 3)LRC、(10, 2, 3)pLRC数据丢失;当失效节点数为7时,(3, 3, 3), ρ = 1的UFP-LRC数据丢失,而RGRC依然保持着较高的修复率。而RGRC因为和basic-Pyamid码消耗的冗余存储空间一样,所以它们的容错能力和修复率几乎一样。综合来说,(17, 10)RGRC的容错能力和修复率均强于(14, 10)的RS码、(10, 2, 3)的LRC、(10, 2, 4, 3)的DLRC、(10, 2, 3)的pLRC、(17, 10)的GRC和(3, 3, 3), ρ = 1的UFP-LRC。

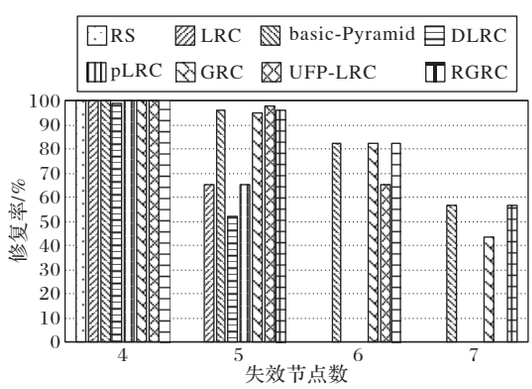

图9　修复率对比

Fig. 9　Comparison of repair rate

3　实验与结果分析

为了在真实的分布式环境下对RGRC的各方面进行测试,并与现在常用的纠删码进行比较,基于Ceph分布式存储系统搭建了纠删码测试平台。纠删码测试平台系统主要分为三个部分,分别为OSD(Object Storage Device)存储节点、Monitor监测节点以及客户端。OSD节点负责存储数据、恢复数据、平衡数据,Monitor负责监视集群的健康状态以及控制集群节点相关操作,客户端用于提供用户操作集群界面。纠删码测试平台体系结构如图10所示。

3.1　实验环境

实验使用的纠删码测试平台包含20个节点,包括:1个客户端、1个Monitor节点和18个OSD存储节点。每个节点的机器参数为:CPU Intel Core i7、内存8 GB、磁盘500 GB,所有节点安装Centos 7系统和Python 2.7运行环境。

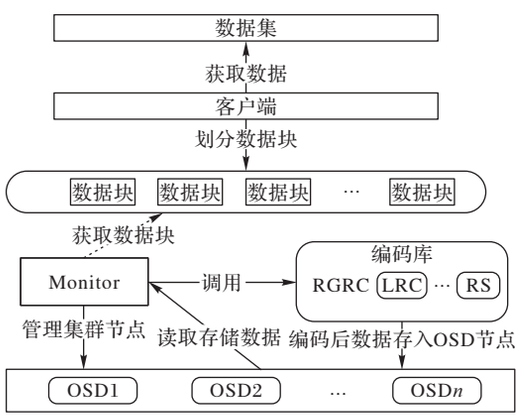

图10　纠删码测试平台体系结构

Fig. 10　Erasure code test platform architecture

3.2　实验对比指标和方法

实验对比指标有修复成本、修复时间和存储开销。修复成本指修复过程中实际的数据读取量,也反映修复操作对网络带宽资源的占用情况;修复时间指修复过程的快慢,反映出算法的实时能力;存储开销指原始文件经纠删码算法计算后实际存储所占用的存储空间,存储开销越小,分布式存储系统的可靠性越好。针对以上实验对比指标,根据单节点修复和多节点修复两种情况分别设计实验来监测和统计其数据。

3.2.1　修复成本实验

1)单节点失效修复成本测试。

实验设计为随机单节点故障来模拟实际应用中的节点故障规律。上传至纠删码测试平台的测试文件大小分别为1 MB~10 MB依次递增,数据块默认大小为1 KB。读取纠删码测试平台记录的各个纠删码修复读取的数据量进行对比。

2)多节点修复成本测试。

实验设计为随机生成失效节点,假设失效节点个数为x,且x不断增大。根据纠删码测试平台记录的每次失效x节点时用于恢复所需要的数据读取量进行对比。实验用的测试文件大小为10 MB,数据块默认大小为1 KB。

3.2.2　修复时间实验

1)单节点失效修复时间实验。

实验设计为以10 MB的存储数据作为测试数据,数据块分块大小默认为1 KB,分别设置10次单节点失效数据修复测试,读取纠删码测试平台记录的各个纠删码修复时间进行对比。

2)多节点失效修复时间实验。

实验设计为以 10 MB 的存储数据作为测试数据,数据块分块大小默认为 1 KB,分别设置随机不同失效节点个数情况下的数据修复测试,读取纠删码测试平台记录的各个纠删码修复时间进行对比。

3.2.3 存储开销实验

实验分别上传 10 MB、25 MB、50 MB 的文件至纠删码测试平台,统计平台记录的各个纠删码在节点中占据的实际存储空间进行对比测试。

为了探究数据块和冗余块个数对 RGRC 的容错能力和单节点修复成本的影响,分别使用不同编码方案的 RGRC 测试其容错能力和单节点修复成本。

3.3 实验对比纠删码

实验中对比的纠删码分别是(14, 10)的 RS 码、(10, 2, 3)的 LRC、(17, 10) 的 basic-Pyamid 码 、(10, 2, 4, 3) 的 DLRC、(10, 2, 3) 的 pLRC、(17, 10) 的 GRC 和 (3, 3, 3),ρ = 1 的 UFP-LRC。(10, 2, 3)的 LRC 和 pLRC 初始都将原始数据块分成两组,每组产生 1 个组内局部冗余块,为整个数据条带生成 3 个全局冗余块;(17, 10)basic-Pyamid 码将原始数据块分为 2 个小组,每组产生 3 个组内局部冗余块,为整个数据条带生成 1 个全局冗余块;(10, 2, 4, 3)的 DLRC 为整个数据条带生成 2 个全局冗余块,将总共 12 个块分成 3 组,每组产生 1 个组内局部冗余块;(17, 10)的 GRC 将原始数据块分成 2 组,每组产生 2 个组内局部冗余块,为整个数据条带生成 2 个全局冗余块,同时 2 个全局冗余块生成 1 个额外冗余块;(3, 3, 3),ρ = 1 的 UFP-LRC 将原始数据块分成 3 个组,每组产生 1 个组内局部冗余块,为整个数据条带生成 3 个全局冗余块;(17, 10)RGRC 将原始数据块分成 2 组,每组产生 3 个组内局部冗余块,为整个数据条带生成 1 个全局冗余块。

3.4 实验结果和分析

3.4.1 单节点修复

1) 单节点修复成本实验。

图 11 为单节点失效平均修复成本对比,修复成本为单节点失效恢复需要的平均数据读取量。RGRC 因在修复多条带单节点失效时,只需读取少量组内数据块和组内局部冗余块,同时可以利用缓存中的块数据,使得 RGRC 的修复成本可以大幅度减少,从图 11 可知(17, 10)RGRC 相较于(14, 10)RS 码修复成本约降低 61.8%,相较于(10, 2, 3)LRC 修复成本约降低 36.4%,相较于(17, 10)basic-Pyamid 码修复成本约降低 29.4%,相较于(10, 2, 4, 3)DLRC 修复成本约降低 25.3%,相较于(17, 10)GRC 修复成本约降低 29.5%,相较于(3, 3, 3),ρ = 1UFP-LR 码修复成本约降低 14.8%。pLRC 虽然修复单节点时会转换成(2, 1)模式进行修复来降低修复成本,但前期需要读取数据块进行转移,间接增加了其修复成本,综合 pLRC 码前期转移数据块的读取成本和后期修复时的读取成本,(17, 10)RGRC 相较于(10, 2, 3)pLRC 修复成本约降低 57.6%。另一方面,由于 RGRC 的修复成本与条带数量相关,随着上传文件数据量的增大,相应条带变多,其单节点修复成本相较于 RS 码、LRC、basic-Pyamid 码、DLRC、pLRC、GRC、UFP-LRC 将会进一步降低。

2) 单节点修复时间实验。

图 12 为单节点平均修复时间对比。

RGRC 恢复时读取的块数量较少,修复成本缩减,同时单节点失效解码结构简单,相比其他纠删码能大幅减少修复时间。从图 12 可知,(17, 10)RGRC 相较于(14, 10)RS 码修复时间约减少 58.7%,相较于(10, 2, 3)LRC 修复时间约减少 34.6%,相较于(17, 10)basic-Pyamid 码修复时间约减少 30.2%,相较于(10, 2, 4, 3)DLRC 修复时间约减少 14.2%,相较于(10, 2, 3)pLRC 修复时间约减少 53.2%,相较于(17, 10)GRC 修复时间约减少 33.1%,相较于(3, 3, 3),ρ = 1UFP-LRC 修复时间约减少 23.6%。

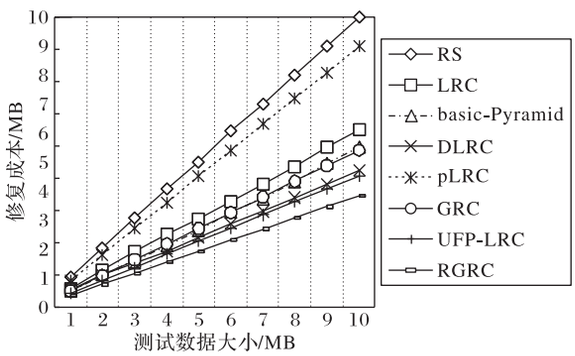

图 11 单节点失效时各纠删码的平均修复成本对比

Fig. 11 Comparison of average repair cost of different erasure codes with single-node failure

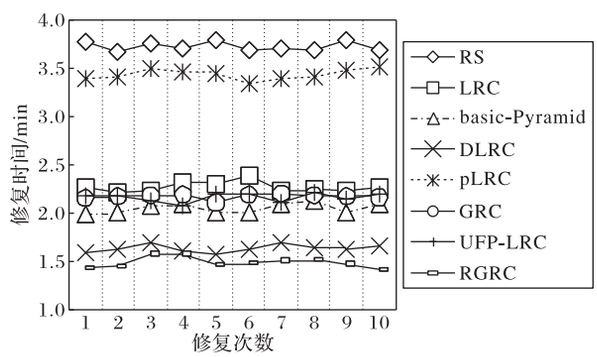

图 12 各纠删码码的单节点平均修复时间对比

Fig. 12 Comparison of average repair time of single node by different erasure codes

3.4.2 多节点修复

1) 多节点修复成本实验。

图 13 为多节点失效的平均修复成本对比,修复成本为节点失效恢复需要的平均数据读取量。由于随着失效节点个数的增多,横向对比的码会超出自身的容错能力,因此,实验设计节点最大失效个数为 4,来测试其平均修复成本。

RGRC 在修复单节点失效时,可以利用分层旋转编码的方式大幅降低修复成本。在修复多个节点失效时,由于牵涉到组解码恢复和全局解码恢复,修复成本会相应增多,低修复成本优势会逐渐减小。但综合平均修复成本,进行多节点修复时,RGRC 的修复成本相较于 RS 码、LRC、basic-Pyamid 码、DLRC、pLRC、GRC、UFP-LRC 依然是有所减少的。

2) 多节点修复时间实验。

图 14 为多节点失效的平均修复时间对比。从图 14 可知,单节点修复时 RGRC 的修复时间最少,随着节点失效个数的增加,相应的修复时间逐渐增加,相较于其他纠删码,其优势逐渐变小。综合平均修复时间,相较于 RS 码、LRC、basic-Pyamid 码、DLRC、pLRC、GRC、UFP-LRC,RGRC 在多节点时的修复时间,虽不及单节点的幅度,但依然是有所减少的。

分布式存储系统中大多数的故障来源于单节点失效,快

速修复单节点失效可以有效降低存储系统中多节点失效情况。由于本次设计的纠删码主要对单节点修复进行改良,降低其修复成本和修复时间。因此,相较于单节点修复实验,RGRC在多节点的修复成本和修复时间无法达到单节点修复的降低幅度,关于多节点修复的改良将会在下一步的研究中进一步展开。

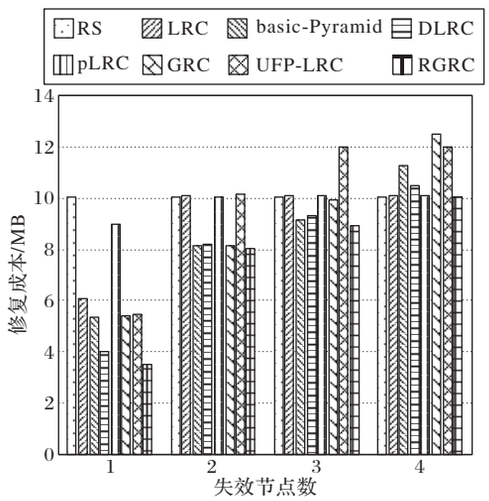

图13　各纠删码的多节点失效平均修复成本对比

Fig. 13　Comparison of average repair cost of multi-node failure by different erasure codes

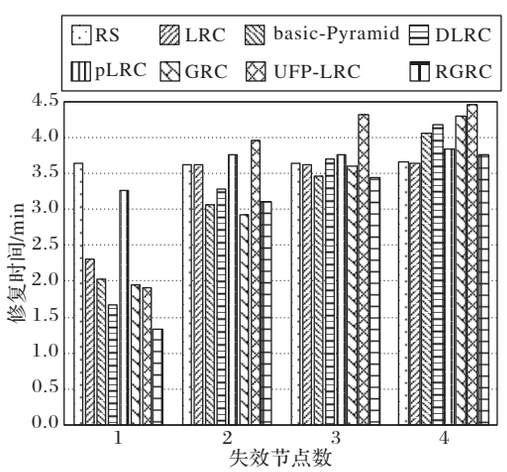

图14　各纠删码的多节点平均修复时间对比

Fig. 14　Comparison of average repair time of multiple nodes by different erasure codes

3.4.3　存储开销实验

RGRC因为其旋转编码方式,能降低单节点修复成本,减少修复时间,但却消耗了一定的存储空间。图15为纠删码存储开销对比。从图15可知,(17, 10)RGRC相较于(14, 10)RS码存储开销约增加21%,相较于(10, 2, 3)LRC、(10, 2, 3)pLRC、(10, 2, 4, 3)DLRC和(3, 3, 3),ρ = 1的UFP-LRC存储开销约增加13%,相较于(17, 10)basic-Pyamid码和(17, 10)GRC不增加额外的存储开销。根据实验数据的对比,虽然增加了一定的存储开销,但相比其降低的修复成本和减少的修复时间,仍可在接受的范围内。

图16展示了不同参数下的RGRC的最大容错能力。从图16可以看出RGRC的容错能力与冗余块个数有关,随着冗余块个数的增加,RGRC相应的容错能力也会随着增加。

实验结果显示,RGRC在单节点修复时,相较于对照的其他纠删码能大幅度降低修复成本和修复时间。在多节点的修复时,随着节点个数的增加,优化幅度逐渐减小,但综合在多个节点修复时的修复成本和修复时间后,RGRC的修复成本和修复时间依然有所改善。虽然RGRC增加了额外的存储开销,但其本身的修复能力增强,最大容错个数以及多节点修复率都有所提高。综合来说,RGRC通过增加冗余同时利用分层旋转编码的方式,改善修复能力、修复成本和修复时间是行之有效的。

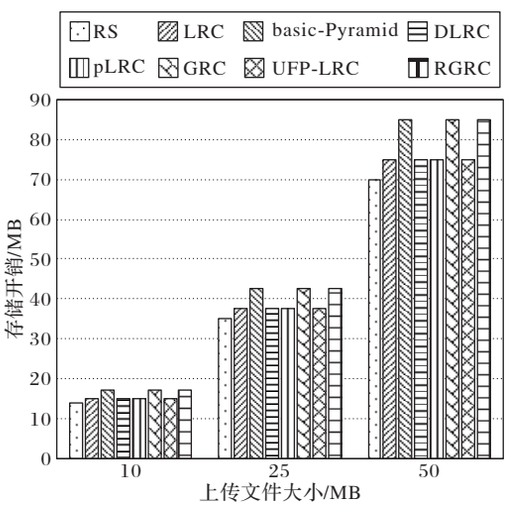

图15　各纠删码的存储开销对比

Fig. 15　Comparison of storage overhead by different erasure codes

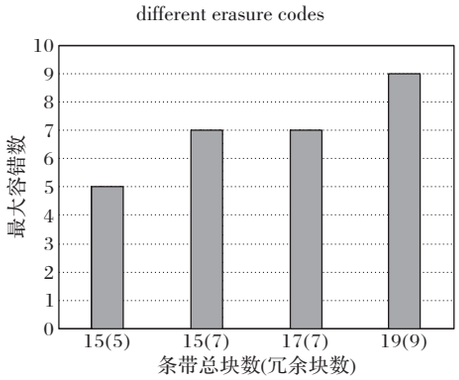

图16　最大容错数和冗余块数的关系

Fig. 16　Relationship between maximum number of failure tolerance and number of redundant blocks

4　结语

在分布式存储系统中其中大约99.75%故障为单节点失效故障,过高的修复成本将影响分布式存储系统的系统性能,为此,针对现有纠删码中单节点失效修复成本过大、修复时间过长的问题,提出了RGRC。

RGRC对条带进行分组编码,同时把条带组合成条带集,以条带集为单位编码组内局部冗余块。RGRC在拥有多容错能力的同时,具备低修复成本和低修复时间的特性,特别针对单节点失效,具有最佳的修复成本和读取成本。实验结果表明,与RS码相比,RGRC能降低单节点修复成本约61.8%,修复时间减少约58.7%,仅需增加约21%的存储开销;与LRC、DLRC、pLRC、UFP-LRC相比,RGRC能降低单节点修复成本14.8%~57.6%,修复时间减少14.2%~53.2%,仅需增加约13%的存储开销;与basic-Pyamid码、GRC相比,RGRC能降低单节点修复成本约29%,修复时间减少约30%,不需要增加额

外的存储开销。同时,RGRC 在多节点失效时也有较高的恢复率和低修复成本性,同时 RGRC 灵活性高,能很好地嵌入到分布式存储系统中,具有很好的前景性。

参考文献(References)

[1] 企业网. 2020 年数据量将达数十万亿 GB 数据中心如何应对[EB/OL]. [2019-01-16]. http://dc. idcquan. com/ywgl/157398. shtml. (China IDC Circle. Data volume will reach tens of trillions of GB in 2020, how the data center deal with it?[EB/OL]. [2019-01-16]. http://dc. idcquan. com/ywgl/157398. shtml.)

[2] 王意洁,孙伟东,周松,等. 云计算环境下的分布存储关键技术[J]. 软件学报, 2012, 23(4): 962-986. (WANG Y J, SUN W D, ZHOU S, et al. Key technologies of distributed storage for cloud computing[J]. Journal of Software, 2012, 23(4): 962-986.)

[3] WANG Y, LI S. Research and performance evaluation of data replication technology in distributed storage systems[J]. Computers and Mathematics with Applications, 2006, 51(11): 1625-1632.

[4] SHVACHKO K, KUANG H, RADIA S, et al. The Hadoop distributed file system [C]// Proceedings of the IEEE 26th Symposium on Mass Storage Systems and Technologies. Piscataway: IEEE, 2010: 1-10.

[5] WEIL S A, BRABDT S A, MILLER E L, et al. Ceph: a scalable, high-performance distributed file system [C]// Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation. Berkeley: USENIX Association, 2006: 307-320.

[6] 杜忠岩,张伟强,鲁华伟. 亚马逊 AWS 云服务分析[J]. 中国新通信, 2015(17): 106-108. (DU Z Y, ZHANG W Q, LU H W. Analysis of Amazon AWS cloud services [J]. China New Telecommunications, 2015(17): 106-108.)

[7] 王意洁,许方亮,裴晓强. 分布式存储中的纠删码容错技术研究[J]. 计算机学报, 2017, 40(1): 236-255. (WANG Y J, XU F L, PEI X Q, Research on erasure code-based fault-tolerance technology for distributed storage [J]. Chinese Journal of Computers, 2017, 40(1): 236-255.)

[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] HUANG C, SIMITCI H, XU Y, et al. Erasure coding in windows azure storage [C]// Proceedings of the 2012 USENIX Conference on Annual Technical Conference. Berkeley: USENIX Association, 2012: 15-26.

[10] HUANG C, CHEN M, LI J. Pyramid codes: flexible schemes to trade space for access efficiency in reliable data storage systems [C]// Proceedings of the 6th IEEE International Symposium on Network Computing and Applications. Piscataway: IEEE, 2007: 79-86.

[11] SATHIAMOORTHY M, ASTERIS M, PAPAILIOPOULOS D, et al. XORing elephants: novel erasure codes for big data [J]. Proceedings of the VLDB Endowment, 2013, 6(5): 325-336.

[12] 周松,王意洁. EXPyramid:一种灵活的基于阵列结构的高容错低修复成本编码方案[J]. 计算机研究与发展, 2011, 48(S1): 30-36. (ZHOU S, WANG Y J. EXPyramid: an array-based flexible coding scheme with high fault-tolerance and low recovery-overhead [J]. Journal of Computer Research and Development, 2011, 48(S1): 30-36.)

[13] 林轩,王意洁,裴晓强,等. 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(S2): 172-181. (LIN X, WANG Y J, PEI X Q, et al. GRC: a high fault-tolerance and low repair cost erasure code for multiple losses [J]. Journal of Computer Research and Development, 2014, 51(S2): 172-181.)

[14] MENG Y, ZHANG L, XU D, et al. A dynamic erasure code based on block code [C]// Proceedings of the 2019 International Conference on Embedded Wireless Systems and Networks. New Westmisnter, Canada: Junction Publishing, 2019: 379-383.

[15] MIYAMAE T, NAKAO T, SHIOZAWA K. Erasure code with shingled local parity groups for efficient recovery from multiple disk failures [C]// Proceedings of the 10th Workshop on Hot Topics in System Dependability. Berkeley: USENIX Association, 2014: 5-5.

[16] RASHMI K V, SHAH N B, KUMAR P V, et al. Explicit construction of optimal exact regenerating codes for distributed storage [C]// Proceedings of the 47th Annual Allerton Conference on Communication, Control, and Computing. Piscataway: IEEE, 2009: 1243-1249.

[17] WU Y, DIMAKIS A G. Reducing repair traffic for erasure coding-based storage via interference alignment [C]// Proceedings of the 2009 IEEE International Symposium on Information Theory. Piscataway: IEEE, 2009: 2276-2280.

[18] 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.

[19] LIU Q, FENG D, HU Y, et al. High-performance general functional regenerating codes with near-optimal repair bandwidth [J]. ACM Transactions on Storage, 2017, 13(2): No. 15.

[20] LIU Q, FENG D, JIANGY H, et al. Z codes: general systematic erasure codes with optimal repair bandwidth and storage for distributed storage systems [C]// Proceedings of the IEEE 34th Symposium on Reliable Distributed Systems. Piscataway: IEEE, 2015: 212-217.

[21] XIE X, WU C, GU J, et al. AZ-code: an efficient availability zone level erasure code to provide high fault tolerance in cloud storage systems [C]// Proceedings of the 35th Symposium on Mass Storage Systems and Technologies. Piscataway: IEEE, 2019: 230-243.

[22] SCHROEDER B, GIBEON G A. Disk failures in the real world: What does an MTTF of 1 000 000 hours mean to you? [C]// Proceedings of the 5th USENIX Conference on File and Storage Technologies. Berkeley: USENIX Association, 2007: 1-16.

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

[24] 张晓阳,许佳豪,胡燏翀. 云存储系统中的预测式局部修复码[J]. 计算机研究与发展, 2019, 56(9): 1988-2000. (ZHANG X Y, XU J H, HU Y C. Proactive locally repairable codes for cloud storage systems [J]. Journal of Computer Research and Development, 2019, 56(9): 1988-2000.)

[25] HU Y, LIU Y, LI W, et al. Unequal failure protection coding technique for distributed cloud storage systems [J]. IEEE Transactions on Cloud Computing, 2017(Early Access): 1-1.

This work is partially supported by the Sichuan Science and Technology Program (2020YFG0150), the Sichuan Artificial Intelligence Major Project (2018GZDZX0030), the Sichuan Science and Technology Achievement Transfer and Transformation Demonstration Project (2018CC0093).

ZHANG Hang, born in 1995, M. S. candidate. His research interests include coding theory, distributed storage.

LIU Shanzheng, born in 1995, M. S. candidate. His research interests include secret image sharing.

TANG Dan, born in 1982, Ph. D., professor. His research interests include coding theory, distributed storage.

CAI Hongliang, born in 1983, Ph. D., lecturer. His research interests include coding theory, visual cryptography.