# 基于简单再生码的分段编码方案

王 静[1] 罗 威[1] 欧阳明生[1] 姜 灿[1] 王新梅[2]
(长安大学信息工程学院 西安 710064)[1]
(西安电子科技大学综合业务网国家重点实验室 西安 710071)[2]

**摘 要** 简单再生码将可容多错的RS纠删码与简单的异或运算相结合,在达到容忍任意 $n-k$ 个节点故障可靠性的基础上,可以实现对单个失效节点的高效快速修复。对简单再生码的失效节点修复过程进行改进,提出一种新的基于简单再生码的分段编码方案,将 $f$ 个具有相同下标的编码块分成两段,将每段中的编码块进行异或操作,生成一个新的校验块。对该方案的存储开销、磁盘读取的开销以及修复带宽开销进行性能分析和仿真实验,结果表明提出的基于简单再生码的分段编码方案在增加少量存储开销的同时,其修复带宽和磁盘读取的开销性能有了很大程度的优化,进一步验证了改进方案的正确性和有效性。

**关键词** 分布式存储,简单再生码,节点修复,分段编码
**中图法分类号** TP302.8 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.8.031

## Segmentation Coding Scheme Based on Simple Regenerating Codes

WANG Jing[1] LUO Wei[1] OUYANG Ming-sheng[1] JIANG Can[1] WANG Xin-mei[2]
(School of Information Engineering, Chang'an University, Xi'an 710064, China)[1]
(State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China)[2]

**Abstract** Combining RS erasure codes with simple XOR operation, simple regenerating codes can realize single failed node repair on the basis of tolerating any $n-k$ node failure. In this paper, based on the repair process of simple regenerating codes, we proposed a new segmentation coding scheme. Concretely, we divided the $f$ coded blocks with the same index into two sections, and then XOR operation was used on the coded blocks of each section to generate a new parity block. Performance analysis and simulation results show that, at a little expense of storage overhead, the proposed segmentation coding scheme based on simple regeneration code can optimize the repair bandwidth and disk I/O overhead, which proves the correctness and effectiveness of this scheme.

**Keywords** Distributed storage, Simple regenerating codes, Node repairing, Segmentation code

## 1 引言

随着信息与互联网技术的迅猛发展,信息技术已经由以计算机为核心转变为以存储为核心[1],数据海量化已成为一种发展趋势。以网络技术为基础的分布式存储技术,将服务器系统中的数据处理和数据存储分离,实现了对数据的海量存储[2]。分布式存储系统通过在多个节点存储冗余数据来提高系统的可靠性,当一小部分节点故障时,可以从其他节点中恢复出故障节点中的数据。

常见的冗余策略包括复制和纠删码[3,4]。复制策略是最简单的冗余策略之一,PAST和Freenet等采用复制策略的分布式存储系统[5],将创建的多个文件副本分发到各个节点,只要有一个节点有效,就可以还原原文件,副本节点越多,系统可靠性越高,且还原原文件不需要编解码操作,因此系统的读取性能较好。

作为另一种重要的冗余策略,$(n,k)$ 纠删码将大小为 $M$ 的原文件分成 $k$ 块,每块大小为 $M/k$,然后将这 $k$ 块文件编码成 $n$ 块分别存储在 $n$ 个节点中,每个节点存储一个编码块。当汇聚节点(Data Collector,DC)连接 $n$ 个节点中的任意 $k$ 个节点时,就可以恢复出原文件。常见的采用纠删码策略的分布式存储系统包括RAID-6[6]、Ocean-Store[7]、Total-Recall[8]等,纠删码策略不需要像复制策略那样大量增加存储开销并且可以实现系统的高容错性[3]。然而在单节点失效的情况下,纠删码在修复失效节点时需要还原整个文件,浪费了网络带宽资源。

Dimakis等人在纠删码的基础上,将网络编码引入分布式存储系统,提出了再生码(regenerating codes)的概念[9],仿真实验证明再生码可以有效降低分布式存储系统中故障节点的修复带宽。局部性修复编码[10-13]修复节点故障时在保障较低修复带宽的同时,可以减少修复过程中连接的节点数,进一步减少修复过程中的磁盘I/O开销,具有较好的修复局部性。Rawat和Vishwanath于2012年提出了基于多项式运算

到稿日期:2015-07-28 返修日期:2015-11-01 本文受国家自然科学基金(61040005,61271262),陕西省自然科学基金(2014JQ8300,2015JM6307),大学生创新创业训练计划(201510710131),长安大学中央高校基金(2013G1241117)资助。

王 静(1982-),女,博士,副教授,主要研究方向为分布式存储、网络编码及再生码,E-mail:jingwang@chd.edu.cn;罗 威(1989-),男,硕士生,主要研究方向为分布式存储中的再生码;欧阳明生(1990-),男,硕士生,主要研究方向为网络编码、协作通信;姜 灿(1989-),男,硕士生,主要研究方向为分布式存储中的再生码;王新梅(1937-),男,教授,博士生导师,主要研究方向为信息论及信道编码理论等。

的局部性修复编码[14]，并采用 Reed-Muller(RM)码存储节点数据，修复局部性可以达到 2 和 3。Papailiopoulos 等人将可容多错的$(n,k)$RS 纠删码与简单的异或运算相结合，提出了简单再生码(Simple Regenerating Codes, SRC)[15]，通过连接少量节点快速修复单节点失效，实现了更高的系统可靠性。紧接着，Rawat 等人基于最大秩度量 Gabidulin 码，给出了能达到最小距离的最优局部性修复编码的构造方法[16,17]，推导出了给定修复带宽情况下，在分布式存储系统中采用最优的局部性修复编码时，系统可以存储的数据量的上界。对于给定的最小码距和修复局部性，Goparaju 等人构造了一类具有最佳码率的循环局部性修复编码[18]。进一步地，Papailiopoulos 等从理论上证明了最优局部性修复编码的存在性[12]，基于 RS 分组码，给出了可达任意高数据速率的局部性修复编码的最佳构造。Xie 等给出了具有两层编码结构的局部性修复编码的构造方法[19]，通过两层编码结构确保数据重构及修复局部性。

然而，上述局部性修复编码方案在修复组内采用 MDS 阵列码来确保较低的修复局部性，当修复组内存在节点故障时，需要进行 MDS 阵列码译码，修复复杂度较高。简单再生码在修复组内采用简单的异或运算，当修复组内有节点发生故障时，可以通过对修复组内相关节点数据进行异或操作，实现对故障节点的快速高效修复。为此，本文提出一种新的基于简单再生码的分段编码方案，通过在每个节点扩容存储一个校验块，进一步降低故障节点的修复带宽开销和磁盘读取开销。仿真实验结果表明，基于简单再生码的分段编码方案在增加少量存储开销的同时，其修复带宽开销和磁盘读取开销性能有了很大程度的优化。

## 2 简单再生码

对于简单再生码，假设原文件大小为 $M$，将原文件平均分为 $f$ 个子文件 $M_1,M_2,\cdots,M_f$，并且对这 $f$ 个子文件分别进行$(n,k)$RS 编码，将具有相同下标的编码块进行简单的异或运算以生成 $n$ 个校验块，并且以下标循环的方式依次将生成的编码块存储在 $n$ 个节点中，如图 1 所示。

| 节点 1 | 节点 2 | … | 节点 n−1 | 节点 n |
|---|---|---|---|---|
| $x_1^{(1)}$ | $x_2^{(1)}$ | | $x_{n-1}^{(1)}$ | $x_n^{(1)}$ |
| $x_2^{(2)}$ | $x_3^{(2)}$ | | $x_n^{(2)}$ | $x_1^{(2)}$ |
| $x_3^{(3)}$ | $x_4^{(3)}$ | | $x_1^{(3)}$ | $x_2^{(3)}$ |
| … | … | … | … | … |
| $x_f^{(f)}$ | $x_{f\oplus 1}^{(f)}$ | | $x_{f\oplus(n-2)}^{(f)}$ | $x_{f\oplus(n-1)}^{(f)}$ |
| $S_{f\oplus 1}$ | $S_{f\oplus 2}$ | | $S_{f\oplus n-1}$ | $S_{f\oplus n}$ |

图 1 $(n,k,f)$简单再生码的一般构造

仿真实验证明，简单再生码与基于 MDS 编码的分布式存储系统相比具有相同可靠性的同时，在修复单个失效节点时具有更小的修复带宽[15]。当系统中某个节点失效时，新生节点可以通过连接 $f$ 个存活节点对其进行修复。以节点 $i$ 为例，对于失效节点 $i$ 中的第一个编码块 $x_i^{(1)}$，需要 $f+1$ 步来对其完成修复，如图 2 所示。首先从节点 $i$ 之前的 $f-1$ 个连续节点中下载与 $x_i^{(1)}$ 具有相同下标的其他 $f-1$ 个编码块 $x_i^{(2)},x_i^{(3)},\cdots,x_i^{(f)}$，然后从节点 $i$ 之前的第 $f$ 个节点中下载校验块 $s_i$，由于 $s_i=\sum_{l=1}^{f}x_i^{(l)}$，因此可以通过异或运算求出丢失的编码块 $x_i^{(1)}$，即 $x_i^{(1)}=s_i-\sum_{l=2}^{f}x_i^{(l)}$。

| 步骤 | 修复失效节点 i 中的编码块 $x_i^{(1)}$ |
|---|---|
| 1 | 读取磁盘 i⊖1 并且下载 $x_i^{(2)}$ |
| 2 | 读取磁盘 i⊖2 并且下载 $x_i^{(3)}$ |
| ⋮ | ⋮ |
| f−1 | 读取磁盘 i⊖(f−1)并且下载 $x_i^{(f)}$ |
| f | 读取磁盘 i⊖f 并且下载 $s_i$ |
| f+1 | 还原 $x_i^{(1)}$：$x_i^{(1)}=s_i-\sum_{l=2}^{f}x_i^{(l)}$ |

图 2 失效节点 $i$ 中的编码块 $x_i^{(1)}$ 的修复过程

由图可知，修复失效节点中的每个编码块都需要下载 $f$ 个编码块，并且需要 $f$ 次磁盘访问。在基于简单再生码的分布式存储系统中，每个节点存储 $f+1$ 个编码块，因此简单再生码的存储开销为 $\alpha=(f+1)M/fk$，即修复单个失效节点中的所有编码块需要下载 $f(f+1)$个数据块，因此修复单个失效节点总共需要$(f+1)M/k$ 的带宽开销。

## 3 基于简单再生码的分段编码方案

### 3.1 方案描述

简单再生码在存储各编码块时采用下标循环的方式，因此修复单个失效节点时只需要 $2f$ 次磁盘访问，即新生节点只需连接 $2f$ 个存活节点即可完成失效节点修复。由于简单再生码中的 $n$ 个校验块 $s_1,s_2,\cdots,s_n$ 均是由 $f$ 个下标相同的编码块进行异或运算编码产生的，因此修复失效节点 $i$ 中的每个编码块均需要下载 $f$ 个编码块。由此可见，简单再生码的单节点修复带宽取决于生成校验块 $s_i$ 的编码块的数量，并且生成校验块的编码块的数量越少，修复单个编码块的带宽开销越小。

为了进一步降低分布式存储系统中单节点修复带宽，提出一种基于简单再生码的分段编码方案。该方案同样使用 RS 纠删码作为外部编码，即将原文件 $M$ 平均分为 $f$ 个子文件 $M_1,M_2,\cdots,M_f$，同时对 $f$ 个子文件分别独立进行$(n,k)$RS 纠删码编码，分别生成 $n$ 个编码块，即 $x_1^{(i)},x_2^{(i)},\cdots,x_n^{(i)}$，其中 $1\leqslant i\leqslant f$。经过$(n,k)$RS 纠删码编码之后，系统总共生成 $f\cdot n$ 个编码块。为了降低修复带宽，首先将 $f$ 个具有相同下标的编码块分成两段，每一段生成一个校验块，则每段中的单个失效编码块可以利用该段中的存活编码块以及该段生成的校验块来完成修复。分段过程分为以下两种情况。

(1)当 $f$ 为偶数时，可以直接将 $f$ 个编码块 $x_i^{(1)},x_i^{(2)},\cdots,x_i^{(f)}$ 平均分为两段，每段含有 $f/2$ 个编码块，即$(x_i^{(1)},x_i^{(2)},\cdots,x_i^{(f/2)})$，$(x_i^{(f/2+1)},x_i^{(f/2+2)},\cdots,x_i^{(f)})$；分别对每段的编码块进行异或编码，各生成一个校验块 $s_i^{(1)}$ 和 $s_i^{(2)}$，其中 $s_i^{(1)}=\sum_{l=1}^{f/2}x_i^{(l)}$，$s_i^{(2)}=\sum_{l=f/2+1}^{f}x_i^{(l)}$；因此系统中共有$(f+2)n$ 个编码块，其中包括经过$(n,k)$RS 纠删码编码生成的 $f\cdot n$ 个编码块以及 $2n$ 个校验块，将$(f+2)n$ 分别存储在 $n$ 个节点中，如图 3 所示。

图 3 给出了 $f$ 为偶数时的分段编码存储方案示意图。将子文件 $M_1,M_2,\cdots,M_{f/2}$ 生成的 $n\cdot(f/2)$个编码块$(x_1^{(1)},x_2^{(1)},\cdots,x_n^{(1)},x_1^{(2)},x_2^{(2)},\cdots,x_n^{(2)},\cdots,x_1^{(f/2)},x_2^{(f/2)},\cdots,x_n^{(f/2)})$以及 $n$ 个校验块 $s_1^{(1)},s_2^{(1)},\cdots,s_n^{(1)}$ 按照简单再生码的存储方式依次存储在节点 1 到节点 $n$ 中。同样地，将子文件 $M_{f/2+1}$，$M_{f/2+2},\cdots,M_f$ 生成的 $n\cdot(f/2)$个编码块 $x_1^{(f/2+1)},x_2^{(f/2+1)},\cdots,x_n^{(f/2+1)},x_1^{(f/2+2)},x_2^{(f/2+2)},\cdots,x_n^{(f/2+2)},\cdots,x_1^{(f)},x_2^{(f)},\cdots,x_n^{(f)}$ 以及 $f$ 个校验块 $s_1^{(2)},s_2^{(2)},\cdots,s_n^{(2)}$ 按照简单再生码的方式依次存储在节点 1 到节点 $n$ 中。

| 节点 1 | 节点 2 | … | 节点 n−1 | 节点 n |
|---|---|---|---|---|
| $x_1^{(1)}$<br>$x_2^{(2)}$<br>…<br>$x_{\frac{f}{2}}^{(\frac{f}{2})}$ | $x_2^{(1)}$<br>$x_3^{(2)}$<br>…<br>$x_{\frac{f}{2}\oplus 1}^{(\frac{f}{2})}$ | … | $x_{n-1}^{(1)}$<br>$x_n^{(2)}$<br>…<br>$x_{\frac{f}{2}\oplus(n-2)}^{(\frac{f}{2})}$ | $x_n^{(1)}$<br>$x_1^{(2)}$<br>…<br>$x_{\frac{f}{2}\oplus(n-1)}^{(\frac{f}{2})}$ |
| $s_{\frac{f}{2}\oplus 1}^{(1)}$ | $s_{\frac{f}{2}\oplus 2}^{(1)}$ | … | $s_{\frac{f}{2}\oplus(n-1)}^{(1)}$ | $s_{\frac{f}{2}\oplus n}^{(1)}$ |
| $x_1^{(\frac{f}{2}+1)}$<br>$x_1^{(\frac{f}{2}+2)}$<br>…<br>$x_{\frac{f}{2}}^{(f)}$ | $x_2^{(\frac{f}{2}+1)}$<br>$x_3^{(\frac{f}{2}+2)}$<br>…<br>$x_{\frac{f}{2}\oplus 1}^{(f)}$ | … | $x_{n\frac{f}{2}1}^{(\frac{f}{2}+1)}$<br>$x_{n\frac{f}{2}1}^{(\frac{f}{2}+2)}$<br>…<br>$x_{\frac{f}{2}\oplus(n-2)}^{(f)}$ | $x_n^{(\frac{f}{2}+1)}$<br>$x_n^{(\frac{f}{2}+2)}$<br>…<br>$x_{\frac{f}{2}\oplus(n-1)}^{(f)}$ |
| $s_{\frac{f}{2}\oplus 1}^{(2)}$ | $s_{\frac{f}{2}\oplus 2}^{(2)}$ | … | $s_{\frac{f}{2}\oplus(n-1)}^{(2)}$ | $s_{\frac{f}{2}\oplus n}^{(2)}$ |

图 3　$f$ 为偶数时的分段编码存储方案

(2)当 $f$ 为奇数时，同样地将 $f$ 个具有相同下标的编码块分为两段，其中第一段包含 $\frac{(f+1)}{2}$ 个编码块，即 $(x_i^{(1)}, x_i^{(2)}, \cdots, x_i^{((f+1)/2)})$，第二段包含 $\frac{(f-1)}{2}$ 个编码块，即 $(x_i^{((f+1)/2+1)}, x_i^{((f+1)/2+2)}, \cdots, x_i^{(f)})$，其存储方案与 $f$ 为偶数时相同，如图 4 所示。

| 节点 1 | 节点 2 | … | 节点 n−1 | 节点 n |
|---|---|---|---|---|
| $x_1^{(1)}$<br>$x_2^{(2)}$<br>…<br>$x_{\frac{f+1}{2}}^{(\frac{f+1}{2})}$ | $x_2^{(1)}$<br>$x_3^{(2)}$<br>…<br>$x_{\frac{f+1}{2}\oplus 1}^{(\frac{f+1}{2})}$ | … | $x_{n-1}^{(1)}$<br>$x_n^{(2)}$<br>…<br>$x_{\frac{f+1}{2}\oplus(n-2)}^{(\frac{f+1}{2})}$ | $x_n^{(1)}$<br>$x_1^{(2)}$<br>…<br>$x_{\frac{f+1}{2}\oplus(n-1)}^{(\frac{f+1}{2})}$ |
| $S_{\frac{f+1}{2}\oplus 1}^{(1)}$ | $S_{\frac{f+1}{2}\oplus 2}^{(1)}$ | … | $S_{\frac{f+1}{2}\oplus(n-1)}^{(1)}$ | $S_{\frac{f+1}{2}\oplus n}^{(1)}$ |
| $x_1^{(\frac{f+1}{2}+1)}$<br>$x_2^{(\frac{f+1}{2}+2)}$<br>…<br>$x_{\frac{f-1}{2}}^{(f)}$ | $x_2^{(\frac{f+1}{2}+1)}$<br>$x_3^{(\frac{f+1}{2}+2)}$<br>…<br>$x_{\frac{f-1}{2}\oplus 1}^{(f)}$ | … | $x_{n-1}^{(\frac{f+1}{2}+1)}$<br>$x_n^{(\frac{f+1}{2}+2)}$<br>…<br>$x_{\frac{f-1}{2}\oplus(n-2)}^{(f)}$ | $x_n^{(\frac{f+1}{2}+1)}$<br>$x_1^{(\frac{f+1}{2}+2)}$<br>…<br>$x_{\frac{f-1}{2}\oplus(n-1)}^{(f)}$ |
| $s_{\frac{f-1}{2}\oplus 1}^{(2)}$ | $s_{\frac{f-1}{2}\oplus 2}^{(2)}$ | … | $s_{\frac{f-1}{2}\oplus(n-1)}^{(2)}$ | $s_{\frac{f-1}{2}\oplus n}^{(2)}$ |

图 4　$f$ 为奇数时的分段编码存储方案

### 3.2　单节点修复过程

在分布式存储系统中采用基于简单再生码的分段编码方案，当系统中某个节点(以节点 $i$ 为例)失效时，分为以下两种情况进行讨论。

(1)当 $f$ 为偶数时，节点 $i$ 中存储的 $f+2$ 个编码块分别为：

$$x_i^{(1)}, x_{i\oplus 1}^{(2)}, \cdots, x_{i\oplus(f/2-1)}^{(f/2)}, s_{i\oplus(f/2)}^{(1)}; x_i^{(f/2+1)}, x_{i\oplus 1}^{(f/2+2)}, \cdots, x_{i\oplus(f/2-1)}^{(f)}, s_{i\oplus(f/2)}^{(2)}$$

其中，$(x_i^{(1)}, x_{i\oplus 1}^{(2)}, \cdots, x_{i\oplus(f/2-1)}^{(f/2)})$ 为节点 $i$ 中的第一段编码块，其相应的校验块为 $s_{i\oplus(f/2)}^{(1)}$。$(x_i^{(f/2+1)}, x_{i\oplus 1}^{(f/2+2)}, \cdots, x_{i\oplus(f/2-1)}^{(f)})$ 为节点 $i$ 中的第二段编码块，其相应的校验块为 $s_{i\oplus(f/2)}^{(2)}$。

对于节点 $i$ 中的第一段编码块及其校验块 $(x_i^{(1)}, x_{i\oplus 1}^{(2)}, \cdots, x_{i\oplus(f/2-1)}^{(f/2)}, s_{i\oplus(f/2)}^{(1)})$，可以采用简单再生码的修复方案进行修复。

图 5 给出了当 $f$ 为偶数时修复第一段编码块及其校验块的示意图，对于该段中的每个编码块，新生节点从连续的 $\frac{f}{2}$ 个存活节点中依次下载与需要修复的编码块具有相同下标的编码块和校验块，其中包括 $\frac{f}{2}-1$ 个编码块以及 1 个校验块。将这 $\frac{f}{2}-1$ 个编码块与具有相同下标校验块进行异或操作，即可修复失效的编码块。

| 步骤 | 修复失效节点 i 中的编码块 $x_i^{(1)}$ |
|---|---|
| 1 | 读取磁盘 i⊖1 并且下载 $x_i^{(2)}$ |
| 2 | 读取磁盘 i⊖2 并且下载 $x_i^{(3)}$ |
| ⋮ | ⋮ |
| $\frac{f}{2}-1$ | 读取磁盘 i⊖($\frac{f}{2}-1$)并且下载 $x_i^{(\frac{f}{2})}$ |
| $\frac{f}{2}$ | 读取磁盘 i⊖$\frac{f}{2}$ 并且下载 $s_i^{(1)}$ |
| $\frac{f}{2}+1$ | 还原 $x_i^{(1)}$：$x_i^{(1)}=s_i^{(1)}-\sum_{l=2}^{\frac{f}{2}} x_i^{(l)}$ |

图 5　第一段中编码块 $x_i^{(1)}$ 的修复过程($f$ 为偶数)

对于该段中的校验块，新生节点通过从连续的 $\frac{f}{2}$ 个存活节点中依次下载 $\frac{f}{2}$ 个与该校验块具有相同下标的编码块，所下载的编码块的上标依次为 $1,2,\cdots,\frac{f}{2}$，然后将这 $\frac{f}{2}$ 个编码块进行异或操作即可完成修复。由于修复第一段中的每个编码块需要 $f/2$ 次磁盘访问，因此，当 $f$ 为偶数时，修复第一段编码块及其校验块总共需要 $f$ 次磁盘访问。

对于第二段编码块以及第二个校验块 $(x_i^{(f/2+1)}, x_{i\oplus 1}^{(f/2+2)}, \cdots, x_{i\oplus(f/2-1)}^{(f)}, s_{i\oplus f/2}^{(2)})$，该段中的每个编码块的修复过程与第一段相同，如图 6 所示。由于第一段编码块和第二段编码块的数量相等，并且存储模式完全一样，因此在修复第一段编码块时所访问的磁盘必定包含修复第二段编码块所需要的存活编码块，从而可以直接从已连接的磁盘中直接下载所需要的存活编码块，故修复第二段编码块及其校验块不会产生多余的磁盘读取开销。

| 步骤 | 修复失效节点 i 中的编码块 $x_i^{(\frac{f}{2}+1)}$ |
|---|---|
| 1 | 读取磁盘 i⊖1 并且下载 $x_i^{(\frac{f}{2}+2)}$ |
| 2 | 读取磁盘 i⊖2 并且下载 $x_i^{(\frac{f}{2}+3)}$ |
| ⋮ | ⋮ |
| $\frac{f}{2}-1$ | 读取磁盘 i⊖($\frac{f}{2}-1$)并且下载 $x_i^{(f)}$ |
| $\frac{f}{2}$ | 读取磁盘 i⊖$\frac{f}{2}$ 并且下载 $s_i^{(2)}$ |
| $\frac{f}{2}+1$ | 还原 $x_i^{(\frac{f}{2}+1)}$：$x_i^{(\frac{f}{2}+1)}=s_i^{(2)}-\sum_{l=\frac{f}{2}+2}^{f} x_i^{(l)}$ |

图 6　第二段中编码块 $x_i^{(f/2+1)}$ 的修复过程($f$ 为偶数)

(2)当 $f$ 为奇数时，对于节点 $i$ 中存储的第一段编码块和第一个校验块 $(x_i^{(1)}, x_{i\oplus 1}^{(2)}, \cdots, x_{i\oplus((f+1)/2-1)}^{((f+1)/2)}, s_{i\oplus(f+1)/2}^{(1)})$，依旧采用简单再生码的修复方案进行修复。

图 7 给出了当 $f$ 为奇数时修复第一段编码块及其校验块的示意图。对于该段中的每个编码块，其修复过程与 $f$ 为偶数时的一样，不同之处在于该段含有 $\frac{(f+1)}{2}$ 个编码块以及 1 个由 $\frac{(f+1)}{2}$ 个编码块异或生成的校验块，因此新生节点必须从连续的 $\frac{(f+1)}{2}$ 个存活节点中依次下载与需修复编码块具有相同下标的编码块和校验块，其中包括 $\frac{(f+1)}{2}-1$ 个编码

块和1个校验块。将这$\frac{(f+1)}{2}-1$个编码块与具有相同下标校验块进行异或操作，即可修复失效的编码块。

| 步骤 | 修复失效节点 i 中的编码块 $x_i^{(1)}$ |
|---|---|
| 1 | 读取磁盘 i⊖1 并且下载 $x_i^{(2)}$ |
| 2 | 读取磁盘 i⊖2 并且下载 $x_i^{(3)}$ |
| ⋮ | ⋮ |
| $\frac{f+1}{2}-1$ | 读取磁盘 i⊖($\frac{f+1}{2}-1$)并且下载 $x_i^{(\frac{f+1}{2})}$ |
| $\frac{f+1}{2}$ | 读取磁盘 i⊖$\frac{f+1}{2}$并且下载 $s_i^{(1)}$ |
| $\frac{f+1}{2}+1$ | 还原 $x_i^{(1)}$：$x_i^{(1)}=s_i^{(1)}-\sum_{l=2}^{\frac{f+1}{2}}x_i^{(l)}$ |

图7　第一段中编码块 $x_i^{(1)}$ 的修复过程($f$为奇数)

对于该段中的校验块，新生节点从连续的$\frac{f+1}{2}$个存活节点中依次下载$\frac{f+1}{2}$个与该校验块具有相同下标的编码块，所下载的编码块的上标依次为 $1,2,\cdots,\frac{f+1}{2}$，然后将这$\frac{f+1}{2}$个编码块进行异或操作即可完成修复。由于修复第一段中的每个编码块需要$\frac{f+1}{2}$次磁盘访问，因此当 $f$ 为奇数时，修复节点 $i$ 中第一段中的所有编码块及其校验块总共需要 $f+1$ 次磁盘访问。

第二段编码块总共包含$\frac{f-1}{2}$个编码块和1个校验块，即$(x_i^{((f+1)/2+1)},x_{i\oplus 1}^{((f+1)/2+2)},\cdots,x_{i\oplus((f-1)/2-1)}^{(f)},s_{i\oplus(f-1)/2}^{(2)})$，且该段中的校验块由$\frac{f-1}{2}$个编码块异或生成。因此，修复该段中的每个编码块时，新生节点只需连接连续的$\frac{f-1}{2}$个存活节点并且下载$\frac{f-1}{2}$个与所需修复编码块具有相同下标的编码块和校验块即可完成修复，如图8所示。

| 步骤 | 修复失效节点 i 中的编码块 $x_i^{(\frac{f+1}{2}+1)}$ |
|---|---|
| 1 | 读取磁盘 i⊖1 并且下载 $x_i^{(\frac{f+1}{2}+2)}$ |
| 2 | 读取磁盘 i⊖2 并且下载 $x_i^{(\frac{f+1}{2}+3)}$ |
| ⋮ | ⋮ |
| $\frac{f-1}{2}-1$ | 读取磁盘 i⊖($\frac{f-1}{2}-1$)并且下载 $x_i^{(f)}$ |
| $\frac{f-1}{2}$ | 读取磁盘 i⊖$\frac{f}{2}$并且下载 $s_i^{(2)}$ |
| $\frac{f-1}{2}+1$ | 还原 $x_i^{(\frac{f+1}{2}+1)}$：$x_i^{(\frac{f+1}{2}+1)}=s_i^{(2)}-\sum_{l=\frac{f+1}{2}+2}^{f}x_i^{(l)}$ |

图8　第二段中编码块 $x_i^{(\frac{(f+1)}{2}+1)}$ 的修复过程($f$为奇数)

由于修复第二段编码块和校验块中的每个编码块只需访问$\frac{f-1}{2}$个存活节点，少于修复第一段编码块和校验块中的每个编码块时所需访问节点的数量($\frac{f+1}{2}$)，因此当 $f$ 为奇数时，修复第二段的所有编码块所访问的节点必定是修复第一段的所有编码块所访问节点的子集，故修复第二段编码块同样不会造成多余的磁盘读取开销。

### 3.3　分段编码方案中 $f$ 的取值

文献[15]证明了简单再生码在修复单个失效节点时的磁盘读取开销为 $\min(2f,n-1)$，即当 $2f\leqslant n-1$ 时，修复单个失效节点的磁盘读取开销为 $2f$，当 $2f>n-1$ 时，其磁盘读取开销为 $n-1$。

然而，为了保证采用简单再生码的分布式存储系统在修复单个失效节点时能够产生尽可能小的磁盘读取开销，在设定 $f$ 大小时应优先考虑 $2f\leqslant n-1$，即 $f\leqslant\frac{n-1}{2}$。

简单再生码的修复带宽为 $\gamma_1=(f+1)\frac{M}{k}$，而RS纠删码的修复带宽为 $\gamma_2=M=k\cdot\frac{M}{k}$。为了使简单再生码比RS纠删码具有更低的修复带宽，应确保 $f+1<k$，即 $f<k-1$。

基于简单再生码的分段编码方案中，将 $f$ 个下标相同的编码块分为两段之后，每段中的编码块需要异或产生一个校验块，因此每个编码段中应当至少包含2个编码块。可见，在分段编码方案中，$f$ 的最小值为4。进一步地，该分段编码方案同样需要满足简单再生码对 $f$ 的限制，则分段编码方案中 $f$ 的取值应当满足 $4\leqslant f\leqslant\frac{(n-1)}{2}$ 且 $4\leqslant f<k-1$。

## 4　性能分析

### 4.1　带宽开销

对于简单再生码，修复单个失效节点中的每一个编码块，新生节点都需要下载 $f$ 个编码块来对其进行还原，因此简单再生码总共需要下载 $f(f+1)$ 个编码块来修复失效节点中的所有数据，其带宽开销为 $\gamma_1=(f+1)\frac{M}{k}$。

对于本文提出的基于简单再生码的分段编码方案，分为以下两种情况：

(1)当 $f$ 为偶数时，由于编码生成的两个校验块均是由$\frac{f}{2}$个编码块异或生成，对于失效节点中的每个编码块只需要下载$\frac{f}{2}$个存活编码块即可完成修复，因此当 $f$ 为偶数时基于简单再生码的分段编码方案对失效节点中的前$\frac{f}{2}+1$个编码块的修复方案与简单再生码的修复方案相同，因此修复这前$\frac{f}{2}+1$个编码块需要总共从其余存活节点中下载$\frac{f(\frac{f}{2}+1)}{2}$个编码块；同理，对于失效节点的后$\frac{f}{2}+1$个编码块，同样需要下载$\frac{f(\frac{f}{2}+1)}{2}$个存活编码块来完成修复，因此当 $f$ 为偶数时，单节点的修复带宽为 $\gamma_3=(\frac{f}{2}+1)\cdot f\cdot(\frac{M}{fk})$。

(2)当 $f$ 为奇数时，失效节点的前一段中的$\frac{f+3}{2}$个编码块的校验块由$\frac{f+1}{2}$个编码块异或生成，因此对于该段中的每个编码块，需要下载$\frac{f+1}{2}$个编码块来完成修复；后一段包含$\frac{f+1}{2}$个编码块，其校验块由$\frac{f-1}{2}$个编码块异或生成，因此修复该段中的每个编码块需要下载$\frac{f-1}{2}$个编码块来完成修复。因此当 $f$ 为奇数时，单节点修复带宽为：

$$\gamma_3=(\frac{f+3}{2}\cdot\frac{f+1}{2}+\frac{f+1}{2}\cdot\frac{f-1}{2})\cdot\frac{M}{fk}$$

图 9 给出了相同文件大小情况下简单再生码和分段编码方案的带宽开销，在仿真程序中提前设定系统的各参数，文件大小 $M=1000\text{Mb}$，$k=36$。假设 $n$ 足够大，为了保证系统的高效性以及在分段之后各段中的编码块依然可以编码，令 $4\leqslant f\leqslant 34$。从仿真结果可以看出，在文件大小固定的情况下，相比于简单再生码，分段编码方案的带宽开销约等于简单再生码的一半。也就是说，分段编码方案节省了近一半的带宽开销，并且随着 $f$ 的增大，简单再生码与分段编码的带宽开销差值也越来越大。例如，当 $f=34$ 时，原简单再生码方案的修复带宽为 972.222Mb，分段编码的修复带宽为 500Mb，比原有方案节省了 472.222Mb 的带宽开销。因此，分段编码方案可以显著降低单节点修复的带宽开销。

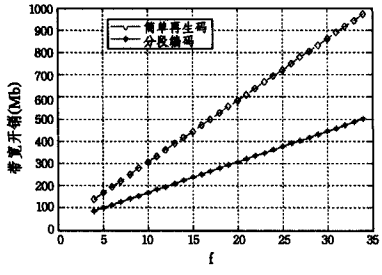


图 9 相同文件大小情况下简单再生码和分段编码方案的带宽开销

图 10 对比了不同文件大小，即 $M$ 不同取值的情况下简单再生码和分段编码方案的单节点修复带宽开销。仿真实验过程中，同样将程序中的相关参数设定为 $k=36$，$f=14$，$1000\text{Mb}\leqslant M\leqslant 7000\text{Mb}$。根据图 10 的仿真结果可知，简单再生码和分段编码方案的带宽开销都随着文件大小 $M$ 的增加而线性增大，且分段编码方案的带宽开销约等于简单再生码的一半。如当文件大小 $M=3000\text{Mb}$ 时，简单再生码的修复带宽为 1250Mb，而分段编码方案的修复带宽为666.667Mb，节省了 583.333Mb 的带宽开销。因此相对于简单再生码，分段编码方案节省了近一半的带宽开销，显著降低了单节点修复的带宽开销。

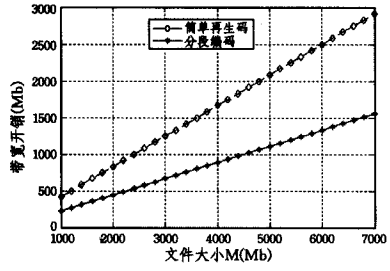


图 10 不同文件大小情况下简单再生码和分段编码方案的带宽开销

### 4.2 磁盘读取开销

文献[15]提出并证明了简单再生码在修复单个失效节点时的磁盘读取开销为 $2f$(这里需确保 $f\leqslant\frac{n-1}{2}$)，对于失效节点中的每一个编码块，系统会连接 $f$ 个连续节点进行修复。而对于下一个编码块，由于简单再生码采用下标滚动的方式存储数据块，因此，新生节点连接的 $f$ 个节点的标号会向前移动一位，也就是新增加一次磁盘读取开销，其余读取的磁盘与之前重合。在简单再生码中，因为每个节点含有 $f+1$ 个编码块，因此修复单失效节点总共需要 $2f$ 次磁盘读取开销。

提出的基于简单再生码的分段编码方案中，当 $f$ 为偶数时，分段编码方案的磁盘读取开销为 $f$；当 $f$ 为奇数时，分段编码方案的磁盘读取开销为 $f+1$。因此，本文提出的分段编码方案可以显著降低单节点修复过程中的磁盘读取开销。

图 11 给出了分段编码方案和简单再生码在修复单个失效节点时磁盘读取开销的仿真结果，实验仿真过程中将 $f$ 设定为 $4\leqslant f\leqslant 34$。通过图 11 可以看出，相比于简单再生码，分段编码方案可以有效降低单节点修复过程中的磁盘读取开销，通过实验仿真进一步验证了上述理论分析的正确性。

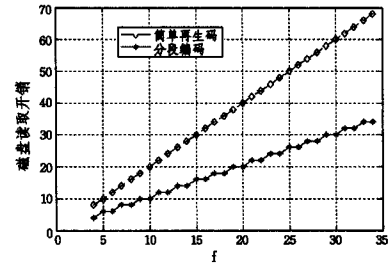


图 11 简单再生码和分段编码方案的磁盘读取开销

### 4.3 存储开销

由于简单再生码中原文件被平均分成 $fk$ 块未经编码的原始文件块，因此系统中的每个编码块大小均为 $\frac{M}{fk}$，而每个节点均存储 $f+1$ 个编码块，故简单再生码的存储开销为 $\alpha_1=\frac{(f+1)M}{fk}$。同理，基于简单再生码的分段编码方案中，每个编码块的大小为 $\frac{M}{fk}$，而每个节点存储 $f+2$ 个编码块，因此分段编码的存储开销为 $\alpha_2=\frac{(f+2)M}{fk}$。分段编码方案相对于简单再生码，增加了 $\frac{M}{fk}$ 的存储开销。

图 12 给出了两种方案存储开销的仿真结果，同样地，在程序中将相关参数设定为 $M=1000\text{Mb}$，$k=36$，$4\leqslant f\leqslant 34$。由图 12 可以看出，随着 $f$ 的增大，两种方案的存储开销变小，并且基于简单再生码的分段编码方案的存储开销只是略高于简单再生码，且两者存储开销的差值随着 $f$ 的增大也逐渐变小。例如，当 $f=34$ 时，简单再生码的存储开销为28.594Mb，分段编码方案的存储开销为 29.412Mb，比简单再生码只增加了 0.818Mb 的存储开销。

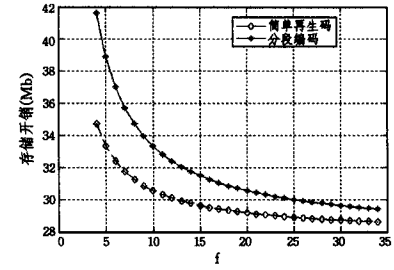


图 12 简单再生码和分段编码方案的存储开销

**结束语** 由于 $(n,k,f)$ 简单再生码单节点失效需要下载 $f$ 个编码块进行修复，为进一步降低简单再生码的修复带宽开销和磁盘读取开销，提出一种新的分段编码方案，通过在每个节点扩容存储一个校验块来进一步降低单节点的修复带宽。仿真实验结果表明，在 $f\geqslant 4$ 的情况下，基于简单再生码的分段编码方案比简单再生码具有更小的修复带宽，随着 $f$ 的增大，修复带宽的优化效果也越来越明显。进一步地，基于简单再生码的分段编码方案在增加少量存储开销的同时，获得了更高的磁盘读取开销性能。

## 参 考 文 献


[1] Wang Yu. Research on data redundancy and maintenance technology in distributed storage system [D]. Guangzhou: South China University of Technology, 2011(in Chinese)
王禹. 分布式存储系统中的数据冗余与维护技术研究[D]. 广州:华南理工大学,2011

[2] Zhao Hao-tian. Study on fault-tolerant and scaling problems of distributed storage systems based on network coding [D]. Hefei: University of Science and Technology of China, 2013(in Chinese)
赵浩天. 基于网络编码的分布式存储容错及扩容问题研究[D]. 合肥:中国科学技术大学,2013

[3] Zeng T, Liu Lei, Zhao J, et al. Router-supported data regeneration protocols in distributed storage systems[C]// Proceeding of 2011 Third International Conference on Ubiquitous and Future Networks (ICUFN). Dalian, 2011: 315-320

[4] Weatherspoon H, Kubiatowicz J. Erasure coding vs. replication: a quantitative comparison[C]// Proceeding of the First International Workshop on Peer-to-Peer Systems. 2002: 328-338

[5] Clarke I, Miller S G, Hong T W, et al. Protecting free expression online with Freenet[J]. IEEE Internet Computing, 2002, 6(1): 40-49

[6] Patterson D A, Gibson G, Katz R H. A case for redundant arrays of inexpensive disks (RAID) [C] // Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data. 1988: 109-116

[7] Kubiatowicz J, Bindel D, Chen Y, et al. OceanStore: an architecture for global-scale persistent store[C] // Proceedings of the ninth International Conference on Architectural Support for Programming Languages and Operating Systems. 2000: 190-201

[8] Bhagwan R, Tati K, Cheng Y, et al. Total recall: system support for automated availability management[C]// Proceedings of the 1st Conference on Networked Systems Design and Implementation. 2004: 25-28

[9] 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

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

[11] 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

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

[13] Khan O, Burns R, Plank J, et al. In search of I/O-optimal recovery from disk failures[C] // Proceeding of 3rd Workshop on Hot Topics in Storage and File Systems (HotStorage). USENIX, 2011

[14] Rawat A S, Vishwanath S. On locality in distributed storage systems[C]// Proceeding of 2012 IEEE Information Theory Workshop (ITW). Lausanne, 2012: 497-501

[15] Papailiopoulos D S, Luo J, Dimakis A G, et al. Simple regenerating codes: network coding for cloud storage[C]// Proceeding of 2012 IEEE INFOCOM. Orlando, FL, 2012: 2801-2805

[16] Rawat A S, Silberstein N, Koyluoglu O O, et al. Optimal locally repairable codes with local minimum storage regeneration via rank-metric codes[C]// Proceeding of 2013 Information Theory and Applications Workshop(ITA). San Diego, CA, 2013: 1-8

[17] Silberstein N, Rawat A S, Koyluoglu O O, et al. Optimal locally repairable codes via rank-metric codes[C]// Proceeding of 2013 IEEE International Symposium on Information Theory (ISIT). Istanbul, 2013: 1819-1823

[18] Goparaju S, Calderbank R. Binary cyclic codes that are locally repairable[C]// Proceeding of 2014 IEEE International Symposium on Information Theory (ISIT). Honolulu, HI, 2014: 676-680

[19] Xie Hong-mei, Yan Zhi-yuan. Two-layer locally repairable codes for distributed storage systems[C]// Proceeding of 2014 IEEE International Conference on Communications (ICC). Sydney, NSW, 2014: 3902-3907


(上接第 117 页)


朱奎龙,侯丽敏. 抗解压缩/压缩攻击的 MP3 压缩域音频水印[J]. 上海大学学报(自然科学版),2008,14(4):331-335

[5] Petitcolas A P. MP3Stego 1.1.16 [ED/OL]. (1997)[2002-03-19]. http://www.cl.cam.ac.uk/-fapp2/steganography/mp3-stego

[6] Zhong S P, Chen T K. Approaches of Improving Performance of MP3Stego and their Implementations[J]. Computer Engineering and Applications, 2006, 35(1): 95-100(in Chinese)
钟尚平,陈铁客. 提高 MP3Stego 性能的方法与实现[J]. 计算机工程与应用,2006,35(1):95-100

[7] Gao H Y. The MP3 Steganography Algorithm Based on Huffman Coding [J]. Journal of Sun Yatsen University (Natural Science Edition), 2007, 46(4): 32-35(in Chinese)
高海英. 基于 Huffman 编码的 MP3 隐写算法[J]. 中山大学学报(自然科学版),2007,46(4):32-35

[8] Liu X J, Guo L. High Capacity Audio Steganography in MP3 Bitstreams [J]. Computer Simulation, 2007, 24(5): 110-113(in Chinese)
刘秀娟,郭立. 大容量 MP3 比特流音频隐写算法[J]. 计算机仿真,2007,24(5):110-113

[9] Pohlmann K C. Principles of Digital Audio(第六版)[M]. 夏田,译. 北京:人民邮电出版社,2013:421

[10] Zhang L G, Wang Rang-ding. Psychoacoustic Model and its Application to MP3 Encoding[J]. Journal of Ningbo University (Natural Science Edition), 2010, 23(3): 27-30(in Chinese)
张力光,王让定. 心理声学模型及其在 MP3 编码中的应用[J]. 宁波大学学报(理工版),2010,23(3):27-30

[11] Ritchey P C, Rego V J. Hiding Secret Messages In Huffman Trees[C]// 2012 Eighth International Conference on Intelligent Information Hiding and Multimedia Signal Processing. IEEE Conference Publications, 2012: 71-74

[12] Zhang Z Y. Research and Implementation of MP3 Encodin Algorithm[D]. Taiwan: National Chiao Tung University, 2002(in Chinese)
张芷燕. MP3 编码法之研究与实现[D]. 台湾:国立交通大学,2002

[13] ITU. Methods for subjective assessment of small impairments in audio systems including multichannel sound systems: ITU-R Rec. BS. 1116[S]: 1994

[14] Deng K Y, Zhang R, Tian Y, et al. Steganalysis of the MP3 Steganographic Algorithm Based on Huffman Coding[C]// Intelligent Computing and Integrated Systems (ICISS). 2010: 79-82