文章编号:1674-2474(2017)08-0152-09 DOI:10.16339/j.cnki.hdxbzkb.2017.08.023

基于柯西矩阵的最小带宽再生码研究*

宋海龙[1,2†],王伟平[2],肖亚龙[1]
(1. 中南大学 信息科学与工程学院,湖南 长沙 410083;2. 吉首大学 信息科学与工程学院,湖南 吉首 416000)

摘 要:节点的失效在大规模分布式存储系统中是常见现象.为防止数据的丢失,系统必须解决失效节点的自修复问题.利用再生码可以在无需下载整个源文件的情况下即可恢复出失效节点的数据,从而能有效节省修复带宽.本文利用柯西矩阵作为编码矩阵,构造了一种精确修复最小带宽再生码(ER-MBR),可以精确修复失效节点,并通过实例演示了在有限域上进行编码解码及节点修复的过程.理论分析和仿真实验都表明利用柯西矩阵作为编码矩阵,其算法的运算效率优于利用范德蒙矩阵或者随机矩阵.

关键词:纠删码;再生码;网络编码;柯西矩阵;范德蒙矩阵;分布式存储
中图分类号:TP393 **文献标志码**:A

Study of Minimum Bandwidth Regeneration Codes Based on Cauchy Matrix

SONG Hailong[1,2†],WANG Weiping[2],XIAO Yalong[1]
(1. School of Information Science and Engineering,Central South University,Changsha 410083,China;
2. School of Information Science and Engineering,Jishou University,Jishou 416000,China)

Abstract:The failures of node are the common phenomena in the massive distributed storage system. To prevent the data loss,the system must solve the problem of self-repairing for failed nodes.Using regenerating codes,the data of failed nodes can be recovered without downloading the whole source file,so repairing bandwidth can be effectively saved.This paper presents an exact-repair minimum bandwidth regeneration code (ER-MBR) by using Cauchy matrix as coding matrix.ER-MBR can exactly repair the failed nodes.The whole process of coding,decoding and node repairing is demonstrated in the finite fields by an instance.Theoretical analysis and simulation experiments prove that by using Cauchy matrix as coding matrix,the operation efficiency of this algorithm is better than that of using Vandermonde matrix or random matrix.

Key words:erasure codes;egenerat codes;network coding;Cauchy matrix;Vandermonde matrix;distributed storage

近几年来,大规模数据存储的需求迅速增长.许多应用如社交网络、文件共享、流媒体点播、云存储系统等都要求对大规模数据的无缝存储、访问和安全保护.这些大规模数据都是通过分布式存储系统(如 RAID-6[1], OceanStore[2], Total Recall[3], DHash++[4]等)利用多个数据节点进行存储的,其

* **收稿日期**:2016-12-11
基金项目:国家自然科学基金资助项目(61173169,61363073),National Natural Science Foundation of China(61173169,61363073)
作者简介:宋海龙(1977—),男,山东平度人,中南大学博士生,吉首大学讲师
†通讯联系人,E-mail:highsong@126.com

目标是要保持数据的长期可靠存储.然而数据规模发展到今天,存储节点的损坏已经是一种常态,而不是偶然事件.例如 Facebook 部署的 Hadoop[5] 集群中共有 3 000 多个节点,涉及 45 PB 数据,日平均失效节点数为 22 个,最高日失效节点数达到 100 个.因此如何保障数据存储的可靠性就成为分布式存储系统研究的首要问题.

保证数据存储可靠性的方法主要是引入冗余信息来提高系统的容错能力,从而在发生节点失效时能够进行修复.通常有两种策略:一种是数据复制策略[6-8],一种是纠删编码策略.数据复制策略的好处是简单易实现,但是数据量太大.如 HDFS[9] 和 GFS[10] 就是采用 3 副本复制的策略,其数据存储量相当于膨胀了 3 倍.利用纠删编码策略数据量膨胀较小,但要求节点有一定的计算能力.目前关于纠删码的研究是分布式存储方案中的热点.

纠删码本来起源于通信领域的信道编码,现已广泛应用于存储系统中.研究人员已提出了许多可行的纠删码方案用于分布式存储系统.Wu[11] 利用网络编码理论设计了一种部分精确修复纠删码,能够精确修复系统部分的单节点故障.Papailiopou-los 等[12] 利用异或运算设计了一种简单再生码,先进行纠删编码,再进行某种方式的异或编码,它能够对单节点损坏进行精确修复.Salim 等[13] 利用重复编码设计了能修复部分节点故障的 MDS 编码[14].Changho 等[15] 利用网络编码理论以及矩阵特征向量设计了一种 MDS 纠删码,可以精确修复单节点故障.蔡鸾佳[16] 考虑到安全问题,将同态指纹检验码技术应用到纠删编码方案中去,设计了具有拜占庭容错性能的存储协议.李谢华等[17] 考虑到云存储中的访问控制安全问题,设计了基于属性加密的访问控制方案.然而这些方案都没有考虑编码解码时的效率问题.由于纠删码在编码解码过程中都要用到大量的有限域上的矩阵运算,运算代价是比较高的,所以编解码效率是个不能忽视的问题.

本文利用一种特殊的纠删码——再生码来实现分布式存储方案,它除了具有纠删码的容错性能之外,还能使修复损坏节点时从各个帮助节点下载的数据量——修复带宽达到最小,并且由于采用柯西型矩阵作为编码矩阵,使得修复效率和数据重建效率都有较大提高.

1　相关知识

纠删编码是一种重要的数据容错策略,常用于构建高可靠性分布式存储系统.再生码是在其基础上发展起来的一种特殊的纠删码,它继承了纠删码的性质,同时在修复损坏的节点时无需下载源数据大小的数据量,从而可以节省网络带宽.下面介绍纠删码和再生码的相关知识.

1.1　MDS 纠删码

定义 1　(n,k)纠删码.所谓(n,k)纠删码[14]是指这样一种编解码方法:发送方将 k 个长度均为α的源数据包编码为 n 个长度为α 的数据包,其中要求 $n>k$.接收方收到了这 n 个编码数据包中的m 个后(这里 $m\geqslant k$),通过解码算法就能够重构出原来的 k 个源数据,而收到的数据包少于 k 个时则不能重构出源数据.其原理如图 1 所示.

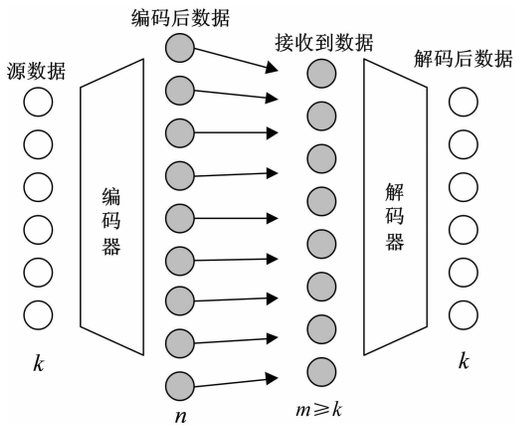

图 1　(n,k)纠删码原理图

Fig.1　Principle of (n,k) erasure coding

如果接收方能够从收到的 m 个数据包中只要任意选k 个都能重构出原来的k 个源数据包的话,那么就称这种编码为极大距离可分码(Maximum Distance Separable,MDS).通常也称这种编码满足 MDS 性.从冗余折中角度来讲,MDS 码是一种最优纠删码,因为 k 个数据包是能够包含所有源数据信息的最小值.

根据纠删码的性质,可以将其运用到分布式存储系统的冗余容错中.当某一个节点损坏的时候,只需要从剩余的节点中选取 k 个或者k 个以上节点下载数据,并利用解码算法恢复出源数据,再对源数据进行编码生成损坏节点的数据包,这就完成了节点的修复.然而在这一过程中,整个的网络通信量是$k\alpha$,通常也称之为修复带宽.可见对于一般的 MDS 码来说,修复一个损坏节点需要下载整个源数据文件大小的数据量,带宽消耗是相当高的.事实上,如果只是修复一个损坏节点,修复带宽是可以小于 $k\alpha$ 的,这就要用到再生码.

1.2 再生码

Dimakis 等[18]于 2007 年首先提出了利用网络编码的方法来优化和设计纠删码,并将其应用于分布式存储系统中,有效降低了数据修复过程中的通信开销,并首次把这种基于网络编码的纠删码命名为再生码.

定义 2 再生码.假设源数据文件中单个符号是有限域 $GF(q)$ 中的元素,文件长度为 B,将其编码存储到 n 个节点,每个节点存储大小为 α 的编码数据.如果某个节点损坏,替代节点就从剩余的 $n-1$ 个节点中连接任意 d(其中 $k\leqslant d\leqslant n-1$)个并下载相关数据进行修复,从每个节点下载的数据量为 $\beta\leqslant\alpha$.这 d 个参与帮助修复的节点就称为帮助节点(Helping Node),用于修复而下载的总数据量 $d\beta$ 就称为修复带宽(Repair Bandwidth),它小于原始文件总大小 B.这样就可以用$\{n,k,d,\alpha,\beta,B\}$这 6 个参数来表示一个再生码.

再生码的所谓"再生性"主要体现在对损坏节点数据的再生上,如果要恢复源数据,则只要按其纠删码的解码过程操作即可.文献[19]指出,根据修复出来的数据包与损坏节点的数据包是否完全一致,再生码又可以分为 3 类修复模式:精确修复、部分精确修复和功能性修复.在精确修复再生码中,损坏节点的数据包能被完全精确的再生出来;而在功能性修复再生码中,修复出来的数据包可能与损坏节点的数据包并不完全一致,但是仍然满足 MDS 性;而部分精确修复再生码则介于两者之间,其对损坏节点的系统部分数据是精确修复的,而对冗余部分则可能是功能性修复的.

再生码的概念提出以后,Wu 等[20]根据网络流理论中的最大流最小割原理给出了再生码参数必须满足的一个条件:

$$B\leqslant\sum_{i=0}^{k-1}\min\{\alpha,(d-i)\beta\} \tag{1}$$

在 d 固定的情况下,最小化 α 可以使得总冗余存储量最小;最小化 β 可以使得总修复带宽最小.以上这两种情况也就分别称为最小存储再生码(Minimum Storage Regeneration,简记为 MSR)和最小带宽再生码(Minimum Bandwidth Regeneration,简记为 MBR).然而根据式(1)可以推知,不可能同时最小化 α 和 β.文献[17]给出了实现 MSR 和 MBR 需满足的条件:

$$(\alpha_{\mathrm{MSR}},\beta_{\mathrm{MSR}})=\left(\frac{B}{k},\frac{B}{k(d-k+1)}\right) \tag{2}$$

$$(\alpha_{\mathrm{MSR}},\beta_{\mathrm{MSR}})=\left(\frac{2dB}{k(2d-k+1)},\frac{2B}{k(2d-k+1)}\right) \tag{3}$$

从式(3)可以看出 $\alpha_{\mathrm{MSR}}/\beta_{\mathrm{MSR}}=d$,$\frac{B}{\beta_{\mathrm{MBR}}}=kd-\frac{k(k-1)}{2}$,这两个值都是整数,也就是说此时 α_{MBR} 和 B 都是 β_{MBR} 的倍数.为了简化运算,不失一般性,不妨假设 $\beta_{\mathrm{MBR}}=1$,则 α_{MBR} 和 B 的值就相应变为:

$$\alpha_{\mathrm{MBR}}=d$$

$$B=kd-\frac{k(k-1)}{2}=\frac{k^2+k}{2}+kd-k^2=C_{k+1}^2+k(d-k) \tag{4}$$

也就是说,在构造的再生码方案中,其参数只要满足以上条件,就可以使修复带宽达到最小.本文主要研究精确修复模式的最小带宽再生码(Exact-Repair MBR,简记为 ER-MBR).此时,由于 $\beta_{\mathrm{MBR}}=1$,$\alpha_{\mathrm{MBR}}=d$,因而参数集$\{n,k,d,\alpha,\beta,B\}$就可以简化为$\{n,k,d,B\}$.下面详细介绍 ER-MBR 的实现原理.

2 实现原理

将源数据中的单个符号看作是有限域 $GF(q)$ 中的元素,则 ER-MBR 的构造过程可采用有限域上的矩阵运算来表示.将源数据向量$[b_1,b_2,\cdots,b_B]$(其中 $b_i\in GF(q);i=1,\cdots,B$)按一定规则存放在阶为 $d\times d$ 的数据矩阵 $\boldsymbol{M}$ 中,再选择一个阶为 $n\times d$ 的矩阵 $\boldsymbol{R}$ 作为编码矩阵,它是预先定义并独立于数据矩阵的,并且在选取时要求它的每个行数为 d 的子方阵都是可逆的.编码后的数据用再生码矩阵 $\boldsymbol{C}$ 来表示,它是一个阶为 $n\times d$ 的矩阵,则再生码可以用以下矩阵乘法公式表示:

$$\boldsymbol{C}_{n\times d}=\boldsymbol{R}_{n\times d}\times\boldsymbol{M}_{d\times d} \tag{5}$$

下面从数据的编码分发、源数据的重构和损坏节点的修复 3 个方面讲解 ER-MBR 的实现原理.

2.1 数据的编码分发

首先将源数据向量按要求组装存放到数据矩阵 $\boldsymbol{M}$ 中,根据式(4)的特点,我们将源数据存放在对称矩阵 $\boldsymbol{M}=\begin{bmatrix}\boldsymbol{S} & \boldsymbol{T}\\ \boldsymbol{T}^{\mathrm{t}} & 0\end{bmatrix}$ 中,其中 t 表示矩阵的转置.$\boldsymbol{S}$ 是一个 $k\times k$ 阶对称矩阵,用来存放 C_{k+1}^2 个源数据中的符号,$\boldsymbol{T}$ 是一个 $k\times(d-k)$ 阶的矩阵,用来存放剩下的 $k(d-k)$ 个数据符号.

再选择一个 $n\times d$ 阶的矩阵 $\boldsymbol{R}$ 作为编码矩阵,

按照式(5)进行运算,得到再生码矩阵 $\boldsymbol{C}$,$\boldsymbol{C}$ 的第 i 行($i=1,\cdots,n$)包含的 $d=\alpha_{\mathrm{MBR}}$ 个元素构成的向量 $\boldsymbol{c}_i$ 就由第 i 个节点存储.这样就把原始数据编码分布到了 n 个存储节点上.记编码矩阵 $\boldsymbol{R}$ 的第 i 行的向量为 $\boldsymbol{r}_i$,则第 i 个存储节点的数据向量就可表示为:$\boldsymbol{c}_i=\boldsymbol{r}_i\boldsymbol{M}$.其实现过程如图 2 所示.

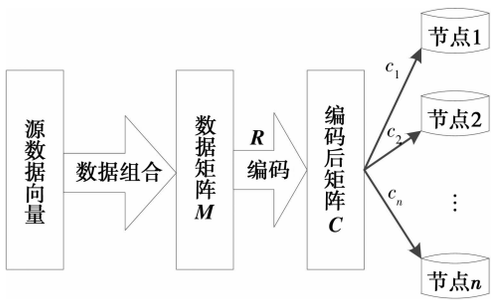

图 2　源数据的编码分发

Fig.2　Distribution of coded source data

2.2 源数据的重构

要重构源数据,数据汇聚点 DC 需要从 n 个存储节点中的任意 k 个节点$\{i_1,i_2,\cdots,i_k\}$中下载编码数据向量,组成再生码数据矩阵 $\boldsymbol{C}_{\mathrm{DC}}$,再从编码矩阵 $\boldsymbol{R}$ 中选取相对应的行向量$\{r_{i1},r_{i2},\cdots,r_{ik}\}$组成解码矩阵 $\boldsymbol{R}_{\mathrm{DC}}$,根据解码算法即可重构出源数据文件,其实现过程如图 3 所示.以下定理 1 说明了数据重构的充分性.

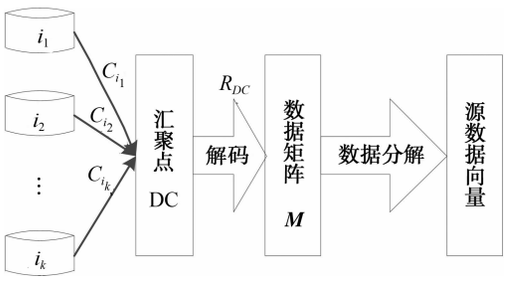

图 3　源数据的重构

Fig.3　Reconstruction of source data

定理 1　在当前的 ER-MBR 编码方式下,所有的 B 个源数据符号都可以通过连接 n 个存储节点中的任意 k 个节点并下载相应数据进行线性变换而予以重构.

证　记 $\boldsymbol{R}_{\mathrm{DC}}=[\boldsymbol{W}_{\mathrm{DC}}\quad\boldsymbol{Z}_{\mathrm{DC}}]$表示编码矩阵 $\boldsymbol{R}$ 中与 k 个汇聚节点相关的行组成的 $k\times d$ 阶子矩阵,$\boldsymbol{W}_{\mathrm{DC}}$是 $k\times k$ 阶方阵,由编码矩阵的性质可知它是可逆的.

由于,

$$\boldsymbol{C}_{\mathrm{DC}}=\boldsymbol{R}_{\mathrm{DC}}\boldsymbol{M}=[\boldsymbol{W}_{\mathrm{DW}}\quad\boldsymbol{Z}_{\mathrm{DC}}]\begin{bmatrix}\boldsymbol{S} & \boldsymbol{T}\\ \boldsymbol{T}^{\mathrm{t}} & 0\end{bmatrix}=$$

$$[\boldsymbol{W}_{\mathrm{DC}}\boldsymbol{S}+\boldsymbol{Z}_{\mathrm{DC}}\boldsymbol{T}^{\mathrm{t}}\quad\boldsymbol{W}_{\mathrm{DC}}\boldsymbol{T}]$$

是从 k 个节点接收到的数据,是已知的,从而 $\boldsymbol{W}_{\mathrm{DC}}\boldsymbol{T}$ 是已知的.由于 $\boldsymbol{W}_{\mathrm{DC}}$也是已知的而且是可逆的,故通过 $\boldsymbol{T}=\boldsymbol{W}_{\mathrm{DC}}^{-1}(\boldsymbol{W}_{\mathrm{DC}}\boldsymbol{T})$可以计算得到 $\boldsymbol{T}$,再根据已知的子矩阵 $\boldsymbol{W}_{\mathrm{DC}}\boldsymbol{S}+\boldsymbol{Z}_{\mathrm{DC}}\boldsymbol{T}^{\mathrm{t}}$ 及 $\boldsymbol{Z}_{\mathrm{DC}}$可以计算得到:

$$\boldsymbol{S}=\boldsymbol{W}_{\mathrm{DC}}^{-1}((\boldsymbol{W}_{\mathrm{DC}}\boldsymbol{S}+\boldsymbol{Z}_{\mathrm{DC}}\boldsymbol{T}^{\mathrm{t}})+\boldsymbol{Z}_{\mathrm{DC}}\boldsymbol{T}^{\mathrm{t}}) \tag{6}$$

从而可以重构整个 $\boldsymbol{M}$.

证毕.

2.3　失效数据节点的修复

在式(5)所定义的 ER-MBR 分布式存储系统中,如果某一个节点损坏,则替换节点可以从剩下的 $n-1$ 个节点中任意连接 d 个并下载编码数据,再根据这 d 个帮助节点所对应的行数从编码矩阵$\boldsymbol{R}$中选取相应的行向量组成修复矩阵 $\boldsymbol{R}_{\mathrm{repair}}$,显然 $\boldsymbol{R}_{\mathrm{repair}}\boldsymbol{M}$ 就是替换节点从帮助节点处下载的再生码数据矩阵,然后根据 ER-MBR 的解码算法即可计算出损坏节点的数据.在这一过程中无需重建源数据就可以直接生成损坏节点的数据,这就比普通的纠删码简化了运算.其实现过程如图 4 所示.以下定理 2 说明了失效节点数据修复的充分性.

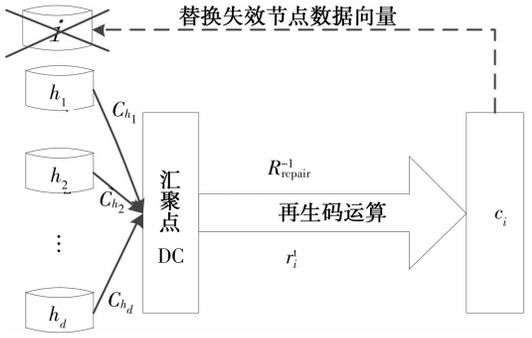

图 4　失效节点数据的再生

Fig.4　Data regenerating of failed node

定理 2　在当前的 ER-MBR 编码方式下,如果第 i 个节点损坏,则替换节点通过从剩下的 $n-1$ 个节点中任意连接 d 个并下载相应数据即可再生出该损坏节点存储的数据向量 $\boldsymbol{c}_i$.

证　由于编码矩阵是预先定义的,所以第 i 个损坏节点所对应的编码向量 $\boldsymbol{r}_i$ 是已知的,它就是编码矩阵 $\boldsymbol{R}$ 的第 i 行.假设替换节点连接的帮助节点记为$\{h_j\mid j=1,\cdots,d\}$,则这些帮助节点所对应的编码向量也是已知的,不妨记为 $r_{h1},\cdots,r_{hd}$,它们组成的 $d\times d$ 阶方阵记为 $\boldsymbol{R}_{\mathrm{repair}}=[r_{h1},\cdots,r_{hd}]^{\mathrm{t}}$,也就是第 i 个损坏节点的修复矩阵,显然它是编码矩阵 $\boldsymbol{R}$ 的子方阵,根据编码矩阵的性质,它的逆矩阵 $\boldsymbol{R}_{\mathrm{repair}}^{-1}$ 是一定存在的.而 $\boldsymbol{R}_{\mathrm{repair}}\boldsymbol{M}=(c_{h1},c_{h2},\cdots,c_{hd})^t$ 是从

帮助节点下载到的再生码数据,也是已知的,且由于 $\boldsymbol{M}$ 是个对称矩阵,因此有

$$\begin{aligned}\boldsymbol{c}_i &= \boldsymbol{r}_i\boldsymbol{M} = \boldsymbol{r}_i\boldsymbol{M}\boldsymbol{R}_{\text{repair}}^{\text{t}}(\boldsymbol{R}_{\text{repair}}^{\text{t}})^{-1} = \\ &((\boldsymbol{r}_i\boldsymbol{M}\boldsymbol{R}_{\text{repair}}^{\text{t}}(\boldsymbol{R}_{\text{repair}}^{\text{t}})^{-1})^{\text{t}})^{\text{t}} = \\ &((\boldsymbol{R}_{\text{repair}}^{\text{t}})^{-1})^{\text{t}}(\boldsymbol{R}_{\text{repair}}\boldsymbol{M}^{\text{t}}\boldsymbol{r}_i^{\text{t}})^{\text{t}} = \\ &(\boldsymbol{R}_{\text{repair}}^{-1}(\boldsymbol{R}_{\text{repair}}\boldsymbol{M}^{\text{t}})\boldsymbol{r}_i^{\text{t}})^{\text{t}} = \\ &(\boldsymbol{R}_{\text{repair}}^{-1}(\boldsymbol{R}_{\text{repair}}M)\boldsymbol{r}_i^{\text{t}})^{\text{t}} = \\ &(\boldsymbol{R}_{\text{repair}}^{-1}(c_{h1},c_{h2},\cdots,c_{hd})^{\text{t}},\boldsymbol{r}_i^{\text{t}})^{\text{t}} \end{aligned} \tag{7}$$

由于在式(7)中,$\boldsymbol{R}_{\text{repair}}^{-1}$,$(c_{h1},c_{h2},\cdots,c_{hd})^{\text{t}}$,$\boldsymbol{r}_i^{\text{t}}$ 都是已知的,故第 i 个损坏节点存储的数据向量 $\boldsymbol{c}_i$ 可以被精确再生出来.证毕.

3 基于柯西矩阵的 ER-MBR 的实现

从上述实现原理可以看出,再生码的运算主要是有限域上的矩阵运算.由于目前计算机是以 8 比特的一个字节为信息存储的最小单位,所以可将源数据的最小符号看作是有限域 $GF(2^8)$ 中的元素.有限域中元素的加法和减法等价,都是异或运算.乘法运算则需要用到本原多项式,具体方法是将有限域中元素的二进制表示式看作是降幂多项式的系数,两个元素相乘就相当于多项式相乘,再对本原多项式求模,所得结果多项式再转化为向量.本实现中取本原多项式 $m(x)=x^8+x^4+x^3+x^2+1$ 作为模多项式.从式(5)可以看出,再生码的构造主要取决于编码矩阵 $\boldsymbol{R}$ 的选取.原则上只要该矩阵的任何一个子方阵都是可逆的就可以作为编码矩阵,在此我们采用柯西型矩阵来作为编码矩阵,下面先给出柯西矩阵的定义.

定义 3 柯西矩阵(Cauchy Matrix)[21].

如果$\{x_i\mid i=1,\cdots,m\}$和$\{y_j\mid j=1,\cdots,n\}$都是域 $GF(q)$中的元素,且满足:1)$\{x_i\mid i=1,\cdots,m\}$两两不同;2) $\{y_j\mid j=1,\cdots,n\}$两两不同;3)$x_i+y_j\neq 0$;则形式为

$$\begin{bmatrix}\frac{1}{x_1+y_1} & \frac{1}{x_1+y_2} & \cdots & \frac{1}{x_1+y_n}\\ \frac{1}{x_2+y_1} & \frac{1}{x_2+y_2} & \cdots & \frac{1}{x_2+y_n}\\ \cdots & \cdots & \cdots & \cdots\\ \frac{1}{x_m+y_1} & \frac{1}{x_m+y_2} & \cdots & \frac{1}{x_m+y_n}\end{bmatrix}$$

的 $m\times n$ 阶矩阵就称为柯西矩阵.

对于一个 n 阶的柯西方阵,其行列式为:

$$\begin{vmatrix}\frac{1}{x_1+y_1} & \frac{1}{x_1+y_2} & \cdots & \frac{1}{x_1+y_n}\\ \frac{1}{x_2+y_1} & \frac{1}{x_2+y_2} & \cdots & \frac{1}{x_2+y_n}\\ \cdots & \cdots & \cdots & \cdots\\ \frac{1}{x_n+y_1} & \frac{1}{x_n+y_2} & \cdots & \frac{1}{x_n+y_n}\end{vmatrix} = \frac{\prod\limits_{i<j}(x_i-x_j)\prod\limits_{i<j}(y_i-y_j)}{\prod\limits_{i,j=1}^{n}(x_i+y_j)} \tag{8}$$

由柯西矩阵的定义易知,以上行列式(8)永不等于零.同时易知柯西矩阵的任何子方阵也是柯西型的,这也就意味着柯西矩阵的任何子方阵都是可逆的.另外柯西矩阵的乘法运算及求逆运算效率都比较高,这些良好的性质,使我们考虑以其作为编码矩阵.

下面以一个 $GF(2^8)$ 上的参数为 $(n,k,d,B)=(6,3,4,9)$的 ER-MBR 为实例来说明实现过程.

3.1 数据矩阵的组装

由参数为$(n,k,d,B)=(6,3,4,9)$可知源文件长度为 9 个字节,不妨记为$[u_1,u_2,\cdots,u_9]$,其中 u_i 都是有限域 $GF(2^8)$ 中的元素.则组装后的数据矩阵为:

$$\boldsymbol{M}=\begin{bmatrix}\boldsymbol{S} & \boldsymbol{T}\\ \boldsymbol{T}^{\text{t}} & 0\end{bmatrix}=\begin{bmatrix}u_1 & u_2 & u_3 & u_7\\ u_2 & u_4 & u_5 & u_8\\ u_3 & u_5 & u_6 & u_9\\ u_7 & u_8 & u_9 & 0\end{bmatrix}$$

3.2 编码矩阵的选取

按柯西矩阵的定义,从 $GF(2^8)$ 中选取合适的元素即可构成编码矩阵 $\boldsymbol{R}$,比如取 $x_i=0,1,2,3,4,5$; $y_j=6,7,8,9$ 则有:

$$\boldsymbol{R}=\begin{bmatrix}\frac{1}{0+6} & \frac{1}{0+7} & \frac{1}{0+8} & \frac{1}{0+9}\\ \frac{1}{1+6} & \frac{1}{1+7} & \frac{1}{1+8} & \frac{1}{1+9}\\ \frac{1}{2+6} & \frac{1}{2+7} & \frac{1}{2+8} & \frac{1}{2+9}\\ \frac{1}{3+6} & \frac{1}{3+7} & \frac{1}{3+8} & \frac{1}{3+9}\\ \frac{1}{4+6} & \frac{1}{4+7} & \frac{1}{4+8} & \frac{1}{4+9}\\ \frac{1}{5+6} & \frac{1}{5+7} & \frac{1}{5+8} & \frac{1}{5+9}\end{bmatrix}=\begin{bmatrix}\frac{1}{6} & \frac{1}{7} & \frac{1}{8} & \frac{1}{9}\\ \frac{1}{7} & \frac{1}{6} & \frac{1}{9} & \frac{1}{8}\\ \frac{1}{4} & \frac{1}{5} & \frac{1}{10} & \frac{1}{11}\\ \frac{1}{5} & \frac{1}{4} & \frac{1}{11} & \frac{1}{10}\\ \frac{1}{2} & \frac{1}{3} & \frac{1}{12} & \frac{1}{13}\\ \frac{1}{3} & \frac{1}{2} & \frac{1}{13} & \frac{1}{12}\end{bmatrix}=\begin{bmatrix}122 & 186 & 173 & 157\\ 186 & 122 & 157 & 173\\ 71 & 167 & 221 & 152\\ 167 & 71 & 152 & 221\\ 142 & 244 & 61 & 170\\ 244 & 142 & 170 & 61\end{bmatrix}$$

3.3　数据的编码分发

根据式(5)易知,编码后的再生码矩阵为:

$$\boldsymbol{C}=\boldsymbol{RM}=\begin{bmatrix}122u_1+186u_2+173u_3+157u_7 & 122u_2+186u_4+173u_5+157u_8 & 122u_3+186u_5+173u_6+157u_9 & 122u_7+186u_8+173u_9\\ 186u_1+122u_2+157u_3+173u_7 & 186u_2+122u_4+157u_5+173u_8 & 186u_3+122u_5+157u_6+173u_9 & 186u_7+122u_8+157u_9\\ 71u_1+167u_2+221u_3+152u_7 & 71u_2+167u_4+221u_5+152u_8 & 71u_3+167u_5+221u_6+152u_9 & 71u_7+167u_8+221u_9\\ 167u_1+71u_2+152u_3+221u_7 & 167u_2+71u_4+152u_5+221u_8 & 167u_3+71u_5+152u_6+221u_9 & 167u_7+71u_8+152u_9\\ 142u_1+244u_2+61u_3+170u_7 & 142u_2+244u_4+61u_5+170u_8 & 142u_3+244u_5+61u_6+170u_9 & 142u_7+244u_8+61u_9\\ 244u_1+142u_2+170u_3+61u_7 & 244u_2+142u_4+170u_5+61u_8 & 244u_3+142u_5+170u_6+61u_9 & 244u_7+142u_8+170u_9\end{bmatrix}$$

再生码矩阵 $\boldsymbol{C}$ 的每一行向量就相应分发给每一节点,即第 i 行向量 $\boldsymbol{c}_i$ 就分发给第 i 个节点($i=1,\cdots,6$).如分发给节点 1 的向量就为:

$$\boldsymbol{c}_1=\begin{bmatrix}122u_1+186u_2+173u_3+157u_7\\ 122u_2+186u_4+173u_5+157u_8\\ 122u_3+186u_5+173u_6+157u_9\\ 122u_7+186u_8+173u_9\end{bmatrix}^t$$

3.4　源数据的重构

由于 ER-MBR 是基于纠删码的,所以通过连接 k 个节点就可以重构出源数据,本例中只要连接 3 个节点即可.假设从节点 1,2,3 下载相应数据来重建源数据.从这 3 个节点可接收到的数据为:

$$\underbrace{\begin{bmatrix}122u_1+186u_2+173u_3+157u_7 & 122u_2+186u_4+173u_5+157u_8 & 122u_3+186u_5+173u_6+157u_9 & 122u_7+186u_8+173u_9\\ 186u_1+122u_2+157u_3+173u_7 & 186u_2+122u_4+157u_5+173u_8 & 186u_3+122u_5+157u_6+173u_9 & 186u_7+122u_8+157u_9\\ 71u_1+167u_2+221u_3+152u_7 & 71u_2+167u_4+221u_5+152u_8 & 71u_3+167u_5+221u_6+152u_9 & 71u_7+167u_8+221u_9\end{bmatrix}}_{\boldsymbol{R}_{\mathrm{DC}}}\underbrace{\qquad\qquad}_{\boldsymbol{M}}$$

而编码矩阵 $\boldsymbol{R}$ 的子方阵

$$\boldsymbol{W}_{\mathrm{DC}}=\begin{bmatrix}\frac{1}{6} & \frac{1}{7} & \frac{1}{8}\\ \frac{1}{7} & \frac{1}{6} & \frac{1}{9}\\ \frac{1}{4} & \frac{1}{5} & \frac{1}{10}\end{bmatrix}=\begin{bmatrix}122 & 186 & 173\\ 186 & 122 & 157\\ 71 & 167 & 221\end{bmatrix}$$ 是已知的,从而其逆也是已知的,即

$$\boldsymbol{W}_{\mathrm{DC}}^{-1}=\begin{bmatrix}144 & 216 & 68\\ 144 & 214 & 72\\ 72 & 112 & 48\end{bmatrix}$$,从而由 $\boldsymbol{T}=\boldsymbol{W}_{\mathrm{DC}}^{-1}(\boldsymbol{W}_{\mathrm{DC}}\boldsymbol{T})=\begin{bmatrix}u_7\\ u_8\\ u_9\end{bmatrix}$ 可以求出 $\boldsymbol{T}$.另外 $\boldsymbol{Z}_{\mathrm{DC}}=\begin{bmatrix}\frac{1}{9}\end{bmatrix}=\begin{bmatrix}157\\ 173\\ 152\end{bmatrix}$ 是已知的,从而根据式(6)可以求出 $\boldsymbol{S}$,重建出整个数据矩阵 $\boldsymbol{M}$.

3.5　失效节点的数据修复

因为修复带宽为 $d=4$,所以若某一节点失效,只要连接 4 个帮助节点就可以进行修复替换.假设节点 5 失效,我们选取节点 2,3,4,6 作为帮助节点来进行修复,首先易知修复矩阵为:

$$\boldsymbol{R}_{\mathrm{repair}}=\begin{bmatrix}r_2\\ r_3\\ r_4\\ r_6\end{bmatrix}=\begin{bmatrix}\frac{1}{7} & \frac{1}{6} & \frac{1}{9} & \frac{1}{8}\\ \frac{1}{4} & \frac{1}{5} & \frac{1}{10} & \frac{1}{11}\\ \frac{1}{5} & \frac{1}{4} & \frac{1}{11} & \frac{1}{10}\\ \frac{1}{3} & \frac{1}{2} & \frac{1}{13} & \frac{1}{12}\end{bmatrix}=\begin{bmatrix}186 & 122 & 157 & 173\\ 71 & 167 & 221 & 152\\ 167 & 71 & 152 & 221\\ 244 & 142 & 170 & 61\end{bmatrix}$$

从而它的逆矩阵为:

$$R_{\text{repair}}^{-1}=\begin{bmatrix}36 & 237 & 116 & 191\\ 56 & 133 & 237 & 233\\ 192 & 57 & 99 & 229\\ 162 & 150 & 40 & 221\end{bmatrix}$$

从节点 2,3,4,6 收到的数据组成的再生码矩阵为:

$$\begin{bmatrix}c_{h_2}\\ c_{h_3}\\ c_{h_4}\\ c_{h_6}\end{bmatrix}=\begin{bmatrix}186u_1+122u_2+157u_3+173u_7 & 186u_2+122u_4+157u_5+173u_8 & 186u_3+122u_5+157u_6+173u_9 & 186u_7+122u_8+157u_9\\ 71u_1+167u_2+221u_3+152u_7 & 71u_2+167u_4+221u_5+152u_8 & 71u_3+167u_5+221u_6+152u_9 & 71u_7+167u_8+221u_9\\ 167u_1+71u_2+152u_3+221u_7 & 167u_2+71u_4+152u_5+221u_8 & 167u_3+71u_5+152u_6+221u_9 & 167u_7+71u_8+152u_9\\ 244u_1+142u_2+170u_3+61u_7 & 244u_2+142u_4+170u_5+61u_8 & 244u_3+142u_5+170u_6+61u_9 & 244u_7+142u_8+170u_9\end{bmatrix}$$

易知节点 5 对应的编码向量为:$\boldsymbol{r}_5=[142\quad 244\quad 61\quad 170]$,从而可得修复后节点的数据向量为:$\boldsymbol{c}_5=(\boldsymbol{R}_{\text{repair}}^{-1}(c_{h2},c_{h3},c_{h4},c_{h6})^{\mathrm{t}}r_5^{\mathrm{t}})^{\mathrm{t}}$,图 5 为节点 5 修复示意图.

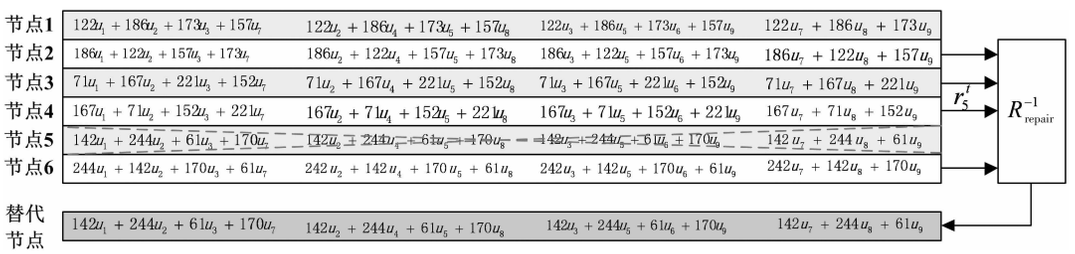

图 5 失效节点 5 的修复

Fig.5 Repairing of failed node five

4 理论分析

在基于 ER-MBR 的分布式存储系统中,由于已实现了修复带宽的最小化,网络性能已达到最优,所以决定系统性能的关键因素就是编码、修复及重建数据的效率了.数据的编码分发只须进行有限域上矩阵的乘法运算,运算代价不是很大.节点修复和源数据重建则要用到矩阵的乘法和编码矩阵的求逆运算,比如损坏节点的修复过程中要计算 $\boldsymbol{R}_{\text{repair}}^{-1}$,数据重构时要计算 $\boldsymbol{W}_{DC}^{-1}$ 等等.有限域上矩阵的求逆运算代价是比较高的.如果选取较好的编码矩阵,就可以有效提高修复效率和重建效率.本方案选取柯西型矩阵作为编码矩阵,就是因为它的求逆运算比目前已知的其他类型矩阵效率要高一些.下面来对比分析一下一般矩阵、柯西矩阵和范德蒙矩阵求逆的计算复杂性.

对于一般的矩阵 $\boldsymbol{R}$,如果其元素是在有限域 $GF(2^8)$ 中随机选取,Chou 等[22]指出,这种矩阵可逆的概率可以达到 99.6%,所以理论上也是可以用来作为再生码的编码矩阵的.然而对于一般的矩阵,其逆矩阵的计算复杂度是相当高的.Copersmith 等[23]已经将其复杂度上界降为 $O(n^{2.496})$,是目前已知最好的结果.

再来看范德蒙矩阵求逆的情况.若 $x_i\in GF(2^8)$,$i=1,\cdots,n$,且 x_i 两两不同,则形如

$$\boldsymbol{W}=\begin{bmatrix}1 & 1 & \cdots & 1\\ x_1 & x_2 & \cdots & x_n\\ \cdots & \cdots & \cdots & \cdots\\ x_1^{n-1} & x_2^{n-1} & \cdots & x_n^{n-1}\end{bmatrix}$$

的矩阵就称为范德蒙矩阵,易知其行列式为:$|\boldsymbol{W}|=\prod_{1\leqslant j<i\leqslant n}(x_i-x_j)$,只要选取两两不同的元素 $x_1,\cdots,x_n$,就可以保证该式是非零的,也即矩阵是可逆的,因而也可以作为再生码的编码矩阵.文献[24]中利用基本对称函数乘积表算法得到了范德蒙矩阵求逆的固有复杂度为 $O(n^2)$.

最后来看柯西矩阵的求逆.文献[25]指出,记一个 n 阶的柯西型矩阵 $\boldsymbol{C}$ 的逆矩阵为 $\boldsymbol{C}^{-1}=(d_{ij})_{n\times n}$,$i,j=1,2,\cdots,n$ 其元素 d_{ij} 可由下式给出:

$$d_{ij}=(-1)^{i+j}\frac{e_jf_i}{a_jb_i(x_j+y_i)}$$

$$a_k=\prod_{i<k}(x_i-x_k)\prod_{k<j}(x_k-x_j)$$

$$b_k=\prod_{i<k}(y_i-y_k)\prod_{k<j}(y_k-y_j)$$

$$e_k=\prod_{i=1}^{n}(x_i-y_i),f_k=\prod_{i=1}^{n}(y_k-x_i)$$

利用该算法,可以将柯西矩阵求逆的复杂度降低到 $O(n\lg^2 n)$.

比较以上 3 种复杂度 $O(n^{2.496})$,$O(n^2)$ 和

$O(n\lg^2 n)$，易知柯西矩阵求逆的复杂度是最低的.另外还要指出的是，如果采用范德蒙型矩阵作为再生码的编码矩阵，则在节点修复过程中，其修复矩阵未必还是范德蒙型的，只能按一般的矩阵来处理，这样其求逆的运算量将会更大.而采用柯西矩阵的再生码，其修复矩阵仍然是柯西型矩阵，存在如文献[25]所指出的快速求逆算法，因此修复效率更高.

5　实验分析

为了验证本文构造方法的可行性，我们选取随机生成的文件，通过 ER-MBR 码编码算法生成分布到各节点的数据，再模拟当某一节点失效时，采用不同的编码矩阵时的节点修复效率.硬件环境是联想 Y700 电脑，英特尔酷睿 i7-6700 CPU(2.6 GHz 主频)，16 G 内存.软件环境是在 windows 10 操作系统下采用 MatlabR2014a 编程语言进行编程实现.实验分为两种，一是在测试文件大小为 10 kB 的固定值情况下，采用参数为(16，3，d，B)的 ER-MBR 码，其中修复带宽 d 进行变化，相应地参数 B 随 d 值而变化；一种是采用固定参数为(16，3，10，52) 的 ER-MBR码情况下，测试文件大小进行变化.所有运算都是在有限域 $GF(2^8)$ 上进行.实验结果如图 6 和图 7 所示.

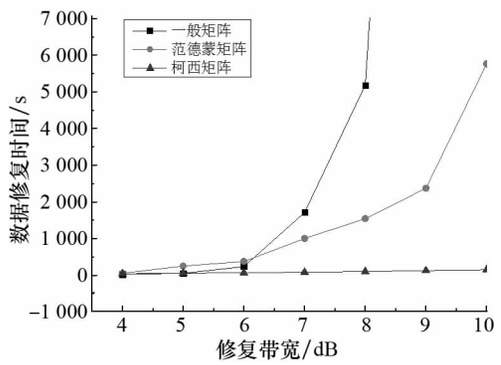

图 6　文件大小固定时修复效率对比

Fig.6　Comparison of repairing efficiency while file length unchanging

从图 6 可以看出，随着修复带宽 d 的增大，采用一般矩阵或范德蒙矩阵作为编码矩阵的 ER-MBR 码，数据修复时间变化较快，而采用柯西矩阵则变化不明显，说明采用柯西矩阵作为编码矩阵的 ER-MBR 码其节点修复效率受 d 的影响不大，在采用多节点修复策略的时候优势非常明显.

从图 7 可以看出，参数固定时，采用各种编码矩阵的 ER-MBR 码其修复时间都与文件大小成正比关系，但是采用柯西矩阵作为编码矩阵的 ER-MBR 码其节点修复效率明显更高.

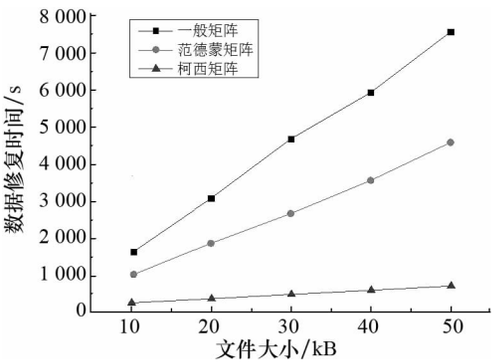

图 7　文件大小变化时修复效率对比

Fig.7　Comparison of repairing efficiency while file length changing

6　结　论

再生码是一种特殊的纠删码，不但能重建源数据，还能在不需要恢复源数据的前提下恢复失效节点的数据，从而节省修复带宽.我们描述了精确修复最小带宽再生码的概念并给出其实现条件，并利用柯西矩阵作为编码矩阵对其进行了优化.理论分析和实验结果表明，在分布式存储系统中，以柯西矩阵作为 ER-MBR 码的编码矩阵，能有效提高修复效率和重建效率，但是否还存在更好的编码矩阵呢？这是个开放问题，有待进一步研究.另外，柯西矩阵本身可能还可以进一步优化，以构造更容易实现解码效率的柯西矩阵出来，这也是个值得研究的方向.

参考文献

[1] ELERATH J G. RAID-6 system reliability dependence on recovery, disk scrubbing, and group size[C]//Proceedings of 2016 Annual Reliability and Maintainability Symposium. Tucson, Arizona, USA: IEEE Computer Society, 2016: 25－28.

[2] RHEA S, WELLS C, EATON P, *et al*. Maintenance-free global data storage[J]. IEEE Internet Computing, 2001, 10(9): 40－49.

[3] BHAGWAN R, TATI K, CHENG Y C, *et al*. Total recall: system support for automated availability management[C]//Proceedings of USENIX the First Symposium on Networked Systems Design and Implementation. San Francisco, California, USA, 2004: 337－350.

[4] DABEK F, LI J, SIT E, *et al*. Designing a DHT for low latency and high throughput[C]//Proceedings of USENIX the First Symposium on Networked Systems Design and

Implementation. San Francisco, California, USA, 2004: 85－89.

[5] SATHIAMOORTHY M, ASTERIS M, PAPAILIOPOULOS D, *et al*. Xoring elephants: novel erasure codes for big data [C]//Proceedings of the 39th International Conference on Very Large Data Bases. Riva del Garda, Trento, Italy, 2013: 325－336.

[6] 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 Yijie, SUN Weidong, ZHOU Song, *et al*. Key technologies of distributed storage for cloud computing[J]. Journal of Software, 2012, 23(4): 962－986. (In Chinese)

[7] 罗象宏,舒继武.存储系统中的纠删码研究综述[J].计算机研究与发展,2012,49(1):1－11.
LUO Xianghong, SHU Jiwu. Summary of research for erasure code in storage system[J]. Journal of Computer Research and Development, 2012, 49(1): 1－11. (In Chinese)

[8] 李挥,张宇蒙,陈俊.大数据环境下的可靠存储技术思考[J].中兴通讯技术,2015,21(5):27－31.
LI Hui, ZHANG Yumeng, CHEN Jun. Reliable storage technologies for big data[J]. ZTE Technology Journal, 2015, 21(5): 27－31. (In Chinese)

[9] SHVACHKO K, KUANG H, RADIA S, *et al*. The hadoop distributed file system [C]//Proceedings of the 26th IEEE Symposium on Mass Storage Systems and Technologies. Lake Tahoe, Nevada, USA: IEEE Computer Society, 2010: 1－10.

[10] WANGMengdi, LI Bo, ZHAO Yongxin, *et al*. Formalizing google file system[C]//Proceedings of the 20th IEEE Pacific Rim International Symposium on Dependable Computing. Singapore, 2014: 190－191.

[11] WU Yunnan. A construction of systematic MDS codes with minimum repair bandwidth[J]. Information Theory, 2011, 57(6): 3738－3741.

[12] PAPAILIOPOULOS D S, LUO Jianqiang, DIMAKIS A G, *et al*. Simple regenerating codes: network coding for cloud storage [C]//Proceedings of the 31st Annual IEEE International Conference on Computer Communications. Orlando, Florida, USA, 2012: 2801－2805.

[13] SALIM E R, RAMCHANDRAN K. Fractional repetition codes for repair in distributed storage systems[C]//Proceedings of the 51st Annual Allerton Conference on Communication. Control, and Computing. Monticello, Illinois, USA, 2010: 1510－1517.

[14] RIZZO L. Effective erasure codes for reliable computer communication protocols[J]. ACM Computer Communication, 1997, 27(2): 24－36.

[15] CHANGHO S, RAMCHANDRAN K. Exact-repair MDS code construction using interference alignment[J]. Information Theory, 2011, 57(3): 1425－1442.

[16] 蔡鸾佳.拜占庭容错纠删码分布式存储协议[J].计算机系统应用,2012,21(2):98－103.
CAI Luanjia. Byzantine fault-tolerant erasure coded distributed storage protocol[J]. Computer Systems & Applications, 2012, 21(2): 98－103. (In Chinese)

[17] 李谢华,张蒙蒙,刘鸿,等. 基于 MA-ABE 的云存储访问控制方法[J].湖南大学学报:自然科学版,2015,42(10):133－140.
LI Xiehua, ZHANG Mengmeng, LIU Hong, *et al*. Multi-authority ABE for access control in cloud storage[J]. Journal of Hunan University: Natural Sciences, 2015, 42(10): 133－140. (In Chinese)

[18] DIMAKIS A G, GODFREY P B, WU Y, *et al*. Network coding for distributed storage systems[J]. IEEE Transactions on Information Theory, 2010: 4539－4551.

[19] DIMAKISA 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.

[20] WU Y, DIMAKIS A G, RAMCHANDRAN K. Deterministic regenerating codes for distributed storage[C]//Proceedings of the 45th Annual Allerton Conference on Control, Computing and Communication. Urbana-Champaign, Illinois, USA, 2007: 1354－1362.

[21] 慕建君,路成业,王新梅.关于纠删码的研究与进展[J].电子与信息学报,2002,24(9):1276－1281.
MU Jianjun, LU Chengye, WANG Xinmei. Research and development about erasure code[J]. Journal of Electronics and Information Technology, 2002, 24(9): 1276－1281. (In Chinese)

[22] CHOU P A, WU Y, JAIN K. Practical network coding [C]//Proceedings of the 51st Annual Allerton Conference on Communication, Control, and Computing. Monticello, Illinois, USA, 2003: 114－120.

[23] COPERSMITH D, WINOGRAD S. On the asymptotic complexity of matrix multiplication[C]//Proceedings of the 22nd Annual Symposium on Foundations of Computer Science. Nashville, Tennessee, USA, 1981: 82－90.

[24] FINCKT, HEINIG G, ROST K. An inversion formula and fast algorithm for Cauchy-Vandermonde matrices [J]. Linear Algebra and Its Applications, 1993, 183(1): 179－191.

[25] WUX, WANG H, YAN Z. Erasure codes for broadcasting small files over erasure channels with low bandwidth[C]//Proceedings of the 42nd Annual Conference on Information Sciences and Systems. Princeton, New Jersey, USA, 2008: 556－561.