doi:10.11959/j.issn.1000-436x.2016197

面向云存储容错系统的 RS 再生码

鄢喜爱[1,2]，张大方[1]，杨金民[1]，张波云[2]
（1. 湖南大学信息科学与工程学院，湖南 长沙 410082；2. 湖南警察学院信息技术系，湖南 长沙 410138）

摘 要：面向云存储容错系统提出了一种 RS 再生纠删码，该编码继承了 RS 编码容多错的可靠性，又能实现容三错的高效性。对 RS 再生码中单节点故障混合修复方法进行了介绍，并求出了混合修复时磁盘读取数的理论下界。从理论上对 RS 再生码的存储开销、译码效率、修复带宽进行了性能评估。实验结果表明，RS 再生纠删码比同类纠删码的修复性能有较大的提升，特别是采用混合修复算法以后，系统单故障恢复时间下降 20.8%~28.2%。
关键词：云存储；容错；纠删码；RS 码；RDP 码
中图分类号：TP391 **文献标识码**：A

RS regenerating codes for cloud storage fault-tolerant system

YAN Xi-ai[1,2], ZHANG Da-fang[1], YANG Jin-min[1] ,ZHANG Bo-yun[2]
(1. College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China;
2. Department of Information Technology,Hunan Police Academy, Changsha 410138, China)

Abstract: RS(Reed-Solomon) regenerating erasure codes was proposed for cloud storage fault-tolerant system, which not only inherited the reliability of the RS encoding, but also achieved the high efficiency of tolerance three faults. Hybrid recovery method of the single fault node based on RS regenerating erasure codes was introduced. And the theoretical lower bound of the number of accessing disks was computed. In theory, the performance evaluation of the storage overhead, decoding efficiency, and repair bandwidth of the RS regenerating erasure codes was carried out. Experiments results show that the repair performance of RS regenerating erasure codes is improved greatly than the similar erasure codes, and the total recovery time of the system is reduced by 20.8%~28.2% using hybrid recovery algorithm in the case of single fault.
Key words: cloud storage, fault tolerance, erasure codes, RS encoding, RDP encoding

1 引言

在云存储系统中，数据中心一般由非常多的数据节点组成，存储的数据达 EB 级甚至 ZB 级，数据失效已成为一种常态，构建云存储系统必须考虑容错机制。存储容错通常采取复制和纠删码 2 种方法。复制方法操作简单，故障恢复快，但存储开销成倍增长，并存在副本一致性维护难的问题；纠删码在故障恢复时会消耗更多的网络带宽，但在容错能力和存储空间上占有绝对优势[1,2]。由于云平台上存储的数据呈指数级增长，云存储系统的容错方法逐渐从复制向纠删码转换。

纠删码可分为阵列码和 RS（Reed-Solomon）编码两大类。阵列码只有简单的异或运算，运算速度快，但一般只能容单错或双错。常用的阵列码有 EVENODD 码、RDP 码和 X-Code 码等[3~5]。RS 编码是可以容多错的编码，并且具备 MDS（maximum distance separable）性质，常用于对存储要求较高的

收稿日期：2016-02-22；**修回日期**：2016-05-23
基金项目：国家自然科学基金资助项目（No.61472130，No.61471169）；国家重点基础研究发展计划（“973”计划）基金资助项目（No. 2012CB315805）；公安部公安理论及软科学研究计划基金资助项目（No.2013LLYJHNST040）；湖南省科技厅科研基金资助项目（No. 2014FJ3049）；网络侦查技术湖南省重点实验室基金资助项目（No. 2016WLZC006）
Foundation Items: The National Natural Science Foundation of China(No.61472130，No. 61471169), The National Key Basic Research and Development Program of China (973 Program) (No.2012CB315805), Ministry of Public Security Public Security Theory and Soft Science Research Projects (No.2013LLYJHNST040), Hunan Provincial Science and Technology Department Research Projects (No.2014FJ3049), Hunan Provincial Key Laboratory of Network Investigational Technology Research Projects (No. 2016WLZC006)

存储系统中。目前一些主流的云平台（Microsoft Azure、Google、HDFS 等）逐步采用了 RS 编码。

标准 RS 编码建立在二元有限域 $G(2^{\omega})$ 上。在 $G(2^{\omega})$ 中，加法的运算只需利用简单的 XOR 实现；乘法的运算较为复杂，若 ω 很小，可通过查询对数表和反对数表获得，否则，必须通过反复的迭代计算才能获得[6]。对 RS 编码的性能优化主要体现在解决有限域上乘法计算难的问题，主要的相关工作如下。BURGISSER 等[7]提出借助拉格朗日插值方法可使 RS 的运算复杂度大大降低，但限制了纠删码的长度，并且当数据量增大时，时间消耗量仍然很大。LACAN 等[8]提出用 2 个 Vandermonde 矩阵来构造编码，比一个 Vandermonde 矩阵加一个单位矩阵的构造效率要高。Plank 等[9]提出将 Vandermonde 矩阵变成 Cauchy 矩阵，将整个 RS 编码的运算全部转化为 XOR 运算，但编译码的时间复杂度仍是平方级，分别为 $O(n\log n)$和 $O(n^2)$。KALCHER S 等[10]针对 RS 编码提出一个基于矢量的多项式乘法的实现算法，并采用 GPU 进行加速。KHANO 等[11]通过旋转和移位来降低 RS 编译码的复杂度，但更新和重构的效率都有不同程度的降低。李小兵等[12]提出了 RS 编码与 X-code 编码相结合生成再生码，在双错情况下采用 X-code 编码，有效地提高了修复效率，但 X-code 编码可扩展性不好，重构时必须按照一定的顺序进行迭代，不方便进行并行操作。

云存储属于归档式存储，对系统的容错能力、编译码效率、重构效率都有较高的要求。针对有中心的云存储系统，本文设计了一种 RS 码与扩展 RDP（ERDP）码相结合的 RS 再生纠删码。再生码兼顾了 RS 码与阵列码的优良特性，当节点故障在三错以上仍采用容错能力强的 RS 编码进行修复，当节点故障在三错以下，启用只有 XOR 运算的 ERDP 编码进行修复；因单节点故障是云存储系统中的主要故障类型，本文讨论了 RS 再生纠删码中单节点故障的磁盘读取方法，求出了磁盘读取数的理论下界。从理论上对 RS 再生纠删码的存储开销、译码效率、修复带宽进行了分析，最后通过实验对系统性能进行了评估。

2 RS 再生纠删码

2.1 RS 编码

RS 编码时，将每个存储节点被分为 t 个存储单元，每个存储节点的字长是 ω bit，假设数据盘有 n 个，分别 $\{D_1,D_2,\cdots,D_n\}$，校验盘有 m 个，分别为 $\{C_1,C_2,\cdots,C_m\}$，校验盘存储单元的数据的生成函数为 F。

$$c_i = F_i(d_1,d_2,\cdots,d_n)$$

RS 校验编码的生成矩阵一般采用 Vandermonde 矩阵[13]，定义 $\boldsymbol{F}$ 为 $m\times n$ 的 Vandermonde 矩阵，其中，$f_{i,j}=j^{i-1}$，则编码的生成可转化为以下形式。

$$\begin{bmatrix} f_{1,1} & f_{1,2} & \cdots & f_{1,n} \\ f_{2,1} & f_{2,2} & \cdots & f_{2,n} \\ \vdots & \vdots & & \vdots \\ f_{m,1} & f_{m,2} & \cdots & f_{m,n} \end{bmatrix} \times \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 2 & \cdots & n \\ \vdots & \vdots & & \vdots \\ 1 & 2^{m-1} & \cdots & n^{m-1} \end{bmatrix} \times \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}$$

图 1 是当 n=6，m=3 时 RS 编码的生成框架。

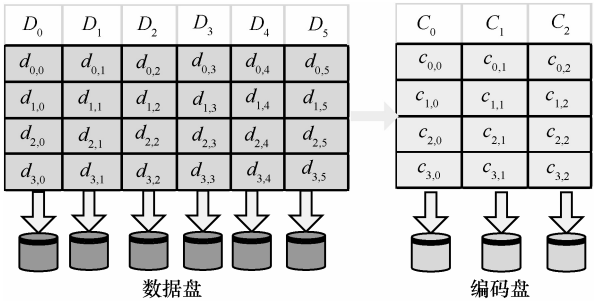

图 1 RS 编码生成框架（n=6, m=3）

2.2 标准 RDP 编码

标准 RDP 编码是一种容双错横式阵列码，所有编码存放在一个$(p-1)\times(p+1)$的阵列中，其中，p 为大于 2 的素数，前 $p-1$ 列存放的是原始数据块，后 2 列存放的是冗余校验数据[14]。RDP 编码的生成公式如式(1)和式(2)所示，其中，式(1)生成的是行校验码，式(2)生成的是正对角线校验码。

$$c_{i,p-1} = \bigoplus_{j=0}^{p-2} c_{i,j} \tag{1}$$

$$c_{i,p} = \bigoplus_{j=0}^{p-1} c_{<i-j>_p,j} \tag{2}$$

$c_{i,j}$ 表示第 i 行第 j 列数据位的值，式(2)中 $<i-j>_p$ 表示 $(i-j) \bmod p$。

图 2 是当 p=5 时，标准 RDP 编码的形成过程描述。

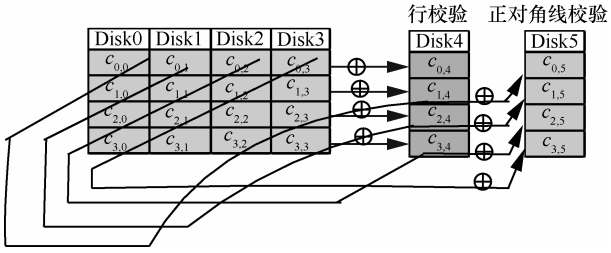

图 2　RDP 码的形成过程描述（p=5）

2.3　RS 再生码（RS-ERDP）构造

2.3.1　有限域上的计算加速处理

设 $F(x)=f_m x^m+f_{m-1}x^{m-1}+\cdots+f_1x^1+f_0$ 为不可约多项式，$G(2^\omega)$ 上 2 个元素 A 和 B 可表示为：$A(x)=a_{m-1}x^{m-1}+\cdots+a_1x^1+a_0$ 和 $B(x)=b_{m-1}x^{m-1}+\cdots+b_1x^1+b_0$，其中，$(f_m f_{m-1}\cdots f_0)$、$(a_{m-1}a_{m-2}\cdots a_1a_0)$、$(b_{m-1}b_{m-2}\cdots b_1b_0)$ 的取值范围为 {0,1}，则 $A(x)$ 和 $B(x)$ 在 $G(2^\omega)$ 上乘积可表示为

$$C(x)=A(x)B(x)\bmod F(x)=\sum_{i=0}^{m-1}\sum_{j=0}^{m-1}a_ib_jx^{i+j}\bmod F(x) \tag{3}$$

对于不可约多项式有

$$x^m=f_0+f_1x+\cdots+f_{m-1}x^{m-1} \tag{4}$$

根据式(4)和有限域上交换律，式(3)可改写为

$$\begin{aligned}\boldsymbol{C}&=\boldsymbol{AB}\bmod F(x)=(\boldsymbol{A}\bmod F(x))b_0+\\&(\boldsymbol{A}x\bmod F(x))b_1+\cdots+(\boldsymbol{A}x^{m-1}\bmod F(x))b_{m-1}\end{aligned} \tag{5}$$

将式(5)改写为矩阵形式

$$\begin{aligned}\boldsymbol{C}&=\left[\boldsymbol{A}^{(0)}\quad\boldsymbol{A}^{(1)}\quad\cdots\quad\boldsymbol{A}^{(m-1)}\right]\times\boldsymbol{B}\\&=\begin{bmatrix}a_0^{(0)}&a_0^{(1)}&\cdots&a_0^{(m-1)}\\a_1^{(0)}&a_1^{(1)}&\cdots&a_1^{(m-1)}\\\vdots&\vdots&&\vdots\\a_{m-1}^{(0)}&a_{m-1}^{(1)}&\cdots&a_{m-1}^{(m-1)}\end{bmatrix}\times\begin{bmatrix}b_0\\b_1\\\vdots\\b_{m-1}\end{bmatrix}\end{aligned} \tag{6}$$

若用 $\boldsymbol{A}^{(i)}$ 表示矩阵的列向量，则

$$\begin{cases}\boldsymbol{A}^{(0)}=\boldsymbol{A}\\\boldsymbol{A}^{(i)}=\boldsymbol{A}^{(i-1)}x\bmod F(x),1\leqslant i\leqslant m-1\end{cases} \tag{7}$$

根据式(6)和式(7)，对于矩阵 $\boldsymbol{C}$ 的第 k 行，当 $i\geqslant1$ 时，可得

$$a_k^{(i)}=\begin{cases}a_{m-1}^{(i-1)}f_0,k=0\\a_{k-1}^{(i-1)}+a_{m-1}^{(i-1)}f_k,1\leqslant k\leqslant m-1\end{cases} \tag{8}$$

式(8)可进一步简化为

$$a_k^{(i)}=\begin{cases}a_0^i=a_{m-1}^{(i-1)}f_0,\quad k=0\\a_k^i=a_{k-1}^{(i-1)},\quad 1\leqslant k\leqslant m-1,a_{m-1}^{(i-1)}=0\\a_k^i=a_{k-1}^{(i-1)}\oplus f_k,1\leqslant k\leqslant m-1,a_{m-1}^{(i-1)}=1\end{cases} \tag{9}$$

从式(9)可知，矩阵的第一列是 $(a_0a_1\cdots a_{m-1})$，随后每一列是前一列左移一位，若前一列高位有溢出，则与 $(f_0f_1\cdots f_{m-1})$ 进行 XOR 运算。

根据以上的推导，有限域上的乘法就可演化为只含移位和 XOR 运算的计算，计算复杂度明显降低。后续标准的 RS 编码和译码计算将按此方法处理。如在 $G(2^4)$ 中，A=5，B=12，设定不可约多项式为 $F(x)=x^4+x+1$，则乘积 $\boldsymbol{C}$ 可通过下面方法获得。

$\boldsymbol{A}^{(0)}=5=(a_3a_2a_1a_0)=(0101)_2$，$a_3^{(0)}=0$，只需将 $\boldsymbol{A}^{(0)}$ 左移 1 位，得 $A^{(1)}=(1010)_2$；$a_3^{(1)}=1$，将 $\boldsymbol{A}^{(1)}$ 左移 1 位，再与 $(f_0f_1\cdots f_{m-1})$ 进行异或运算，$\boldsymbol{A}^{(2)}=10100\oplus10011=0111$；因为 $a_3^{(2)}=0$，只需将 $\boldsymbol{A}^{(2)}$ 左移 1 位，得 $\boldsymbol{A}^{(3)}=1110$。计算结果与查询对数表和反对数表一致。

$$\begin{aligned}\boldsymbol{C}=\boldsymbol{A}\times\boldsymbol{B}\bmod F(x)&=\begin{bmatrix}1&0&1&0\\0&1&1&1\\1&0&1&1\\0&1&0&1\end{bmatrix}\times\begin{bmatrix}0\\0\\1\\1\end{bmatrix}\\&=(1001)_2=9\end{aligned}$$

2.3.2　扩展 RDP 编码

为了增加 RDP 编码的容错能力，本文构建了一种可扩展 RDP 编码（ERDP）。编码过程中，增加一个校验节点，所有编码存放在一个$(p-1)\times(p+2)$的阵列中，p 为大于 2 的素数，前 $p-1$ 列存放的是原始数据块，后 3 列存放的是冗余校验数据。在后 3 列中，前 2 列存放的编码与标准 RDP 的校验码一样，最后一列存放的编码为反对角线校验码，通过式(10)生成。

$$c_{i,p+1}=\bigoplus_{j=0}^{p-1}c_{<i+j>_p,j} \tag{10}$$

$c_{i,j}$ 表示第 i 行第 j 列数据位的值，$<i+j>_p$ 表示 $(i+j)\bmod p$。

图 3 是当 p=5 时，ERDP 编码的形成过程描述。

ERDP 编码是一个可容三错的横式阵列码，在单错或双错情况下，解码方法与标准 RDP 码完全相同，当出现三错时，启用反对角线校验码。假设

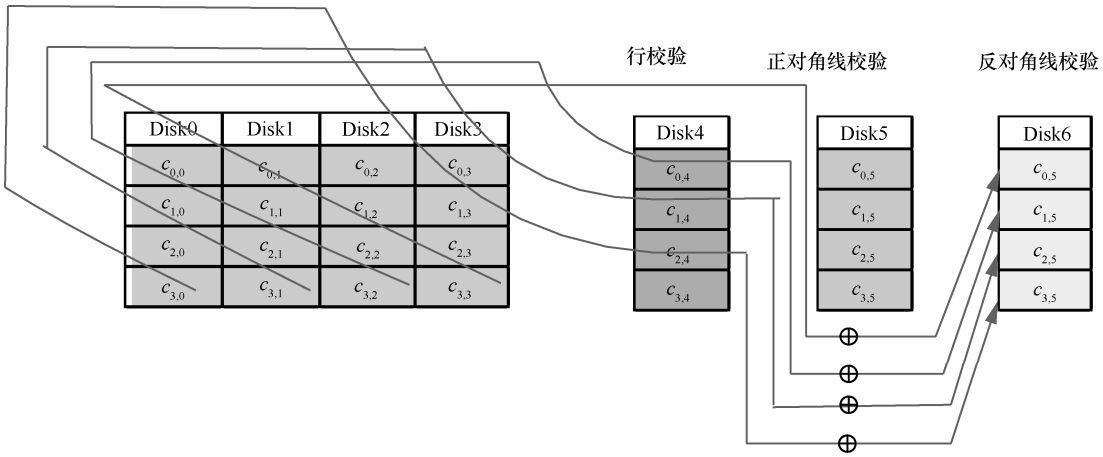

图 3 ERDP 码的形成过程描述（p=5）

3 个失效盘为 Dx、Dy、Dz，3 个修复的校验算子分别为

$$S_i^{(1)} = c_{i,p-1} \oplus \bigoplus_{\substack{j=0 \\ j\neq x,y,z}}^{p-2} c_{i,j} \tag{11}$$

$$S_i^{(2)} = c_{i,p} \oplus \bigoplus_{\substack{j=0 \\ j\neq x,y,z}}^{p-1} c_{<i-j>_p,j} \tag{12}$$

$$S_i^{(3)} = c_{i,p+1} \oplus \bigoplus_{\substack{j=0 \\ j\neq x,y,z}}^{p-1} c_{<i+j>_p,j} \tag{13}$$

2.3.3 RS 再生码的生成

结合 RS 编码与 ERDP 编码的特性，本文构造了一种 RS-ERDP 的容错纠删码。为了能容忍多错，内部编码仍采用标准 RS 编码，为了达到 3 个错误以内高效地修复，外部采用只含 XOR 运算的 RS 再生码，即外部采用在 RS 编码基础上生成的 ERDP 编码。由于 RDP 编码对磁盘数提出了要求（p 为素数），为了不破坏 RS 编码对磁盘数无要求的优良特性，在生成 RS 编码之后，先分组再生成 ERDP 编码。

假设 RS 编码中数据盘数为 n，RS 编码校验盘数为 m，ERDP 编码磁盘数为 p+2，RS-ERDP 编码的构造步骤如下。

Step1 将文件按条带分成 n 块，利用 Vandermonde 矩阵生成 RS 编码，一个条带中会存在 m+n 个编码块。

Step2 将条带分组，每组编码块的个数为 $p-1$(p 为素数)，$k=\dfrac{m+n}{p-1}$，$RS=\{RS_1,RS_2,\cdots,RS_k\}$。

Step3 每组增加 3 个容错节点，将每一组的原 RS 编码生成 ERDP 编码。其中，p 列存放行校验码，p+1 列存放正对角线校验码，p+2 列存放反对角线校验码。

表 1 是 n=3，m=3，p=7 时，一个 RS 单元的 ERDP 编码生成情况。原始数据块存储在 D_0、D_1、D_2 列，RS 编码存储在 D_3、D_4、D_5 列，ERDP 编码存储在 D_6、D_7、D_8 列。

3 RS-ERDP 编码中单节点故障修复的读盘优化

在云存储系统中，单节点故障发生的概率占 90%以上。一旦出现第 1 个失效节点，其他节点失效的概率会大大增加，更多节点失效会随后发生[15]。因此，当系统中出现单节点失效时，需要尽快修复至正常

表 1 **一个 RS 单元的 ERDP 编码生成过程**

D_0	D_1	D_2	D_3	D_4	D_5	$D_6(c_{i,6})$	D_7	D_8
$d_{0,0}$	$d_{0,1}$	$d_{0,2}$	$c_{0,3}$	$c_{0,4}$	$c_{0,5}$	$d_{0,0}\oplus d_{0,1}\oplus d_{0,2}\oplus c_{0,3}\oplus c_{0,4}\oplus c_{0,5}$	$c_{0,5}\oplus c_{1,4}\oplus c_{2,3}\oplus d_{3,2}\oplus c_{4,1}\oplus c_{5,0}$	$c_{5,5}\oplus c_{4,4}\oplus c_{3,3}\oplus d_{2,2}\oplus d_{1,1}\oplus d_{0,0}$
$d_{1,0}$	$d_{1,1}$	$d_{1,2}$	$c_{1,3}$	$c_{1,4}$	$c_{1,5}$	$d_{1,0}\oplus d_{1,1}\oplus d_{1,2}\oplus c_{1,3}\oplus c_{1,4}\oplus c_{1,5}$	$c_{0,4}\oplus c_{1,3}\oplus d_{2,2}\oplus d_{3,1}\oplus c_{4,0}\oplus c_{5,6}$	$c_{5,4}\oplus c_{4,3}\oplus d_{3,2}\oplus d_{2,1}\oplus d_{1,0}\oplus c_{0,6}$
$d_{2,0}$	$d_{2,1}$	$d_{2,2}$	$c_{2,3}$	$c_{2,4}$	$c_{2,5}$	$d_{2,0}\oplus d_{2,1}\oplus d_{2,2}\oplus c_{2,3}\oplus c_{2,4}\oplus c_{2,5}$	$c_{0,3}\oplus d_{1,2}\oplus d_{2,1}\oplus d_{3,0}\oplus c_{5,5}\oplus c_{4,6}$	$c_{5,3}\oplus d_{4,2}\oplus d_{3,1}\oplus d_{2,0}\oplus c_{0,5}\oplus c_{1,6}$
$d_{3,0}$	$d_{3,1}$	$d_{3,2}$	$c_{3,3}$	$c_{3,4}$	$c_{3,5}$	$d_{3,0}\oplus d_{3,1}\oplus d_{3,2}\oplus c_{3,3}\oplus c_{3,4}\oplus c_{3,5}$	$d_{0,2}\oplus d_{1,1}\oplus d_{2,0}\oplus c_{5,4}\oplus c_{4,5}\oplus c_{3,6}$	$d_{5,2}\oplus d_{4,1}\oplus d_{3,0}\oplus c_{0,4}\oplus c_{1,5}\oplus c_{2,6}$
$d_{4,0}$	$d_{4,1}$	$d_{4,2}$	$c_{4,3}$	$c_{4,4}$	$c_{4,5}$	$d_{4,0}\oplus d_{4,1}\oplus d_{4,2}\oplus c_{4,3}\oplus c_{4,4}\oplus c_{4,5}$	$d_{0,1}\oplus d_{1,0}\oplus c_{5,3}\oplus c_{4,4}\oplus c_{3,5}\oplus c_{2,6}$	$d_{5,1}\oplus d_{4,0}\oplus c_{0,3}\oplus c_{1,4}\oplus c_{2,5}\oplus c_{4,6}$
$d_{5,0}$	$d_{5,1}$	$d_{5,2}$	$c_{5,3}$	$c_{5,4}$	$c_{5,5}$	$d_{5,0}\oplus d_{5,1}\oplus d_{5,2}\oplus c_{5,3}\oplus c_{5,4}\oplus c_{5,5}$	$d_{0,0}\oplus d_{5,2}\oplus c_{4,3}\oplus c_{3,4}\oplus c_{2,5}\oplus c_{1,6}$	$d_{5,0}\oplus d_{0,2}\oplus c_{1,3}\oplus c_{2,4}\oplus c_{3,5}\oplus c_{5,6}$

状态。影响单节点修复的关键因素是磁盘的读取，读取的磁盘数越少，则修复越快。如果失效节点是校验节点，因为生成校验时最多只参与了一次，所以只能重新生成编码，磁盘的读取没有优化空间；如果失效节点是数据节点，因为生成校验时可能参与了多次，所以磁盘的读取可以进行优化。

在 RDP 编码中，传统的节点故障修复的方法是通过单一的行校验来进行修复，即读取行校验节点和存储节点的所有数据，数据节点的读取次数为 $(p-1)^2$。传统的修复方法忽视了对角线校验的存在，行校验与对角线校验会存在一些公共数据节点（同时参与了行校验与对角线校验）。对于公共数据节点，如果采用行校验和对角线校验进行混合修复，节点只需读一次就可以了，这样可以有效降低磁盘的读取次数。XIANG 等[16]提出了一种容双错 RDP 编码中单节点故障的混合修复方法，行校验和对角线校验修复各取一半，得出数据节点的读取次数的理论下界为 $\frac{3(P-1)^2}{4}$。

在 RS-ERDP 编码中，增加了一个反对角线校验，存在的公共数据节点会更多。下面本文来探讨容三错 RS-ERDP 编码中单节点故障修复的读盘优化问题。

定义 1

1) 设行校验集合为

$$R=\{R_i \mid 0\leqslant i\leqslant p-2\}=\{d_{i,r} \mid 0\leqslant i\leqslant p-2, 0\leqslant r\leqslant p-2\}$$

2) 设对角线校验集合 D 为

$$D=\{D_j \mid 0\leqslant j\leqslant p-2\}=\{d_{i,r} \mid (i-r)\bmod p=j, 0\leqslant i\leqslant p-2, 0\leqslant r\leqslant p-1\}$$

3) 设反对角线校验集合 B 为

$$B=\{B_t \mid 0\leqslant t\leqslant p-2\}=\{d_{i,r} \mid (i+r)\bmod p=t, 0\leqslant i\leqslant p-2, 0\leqslant r\leqslant p-1\}$$

从定义可知 $|R|=|D|=|B|=p-1$。

引理 1

1) $R_i\cap D_j\neq\varnothing$，$|R_i\cap D_j|=1$

2) $R_i\cap B_t\neq\varnothing$，$|R_i\cap B_t|=1$

3) $\exists D_j\cap B_t=\varnothing$

证明　1) 根据定义 1 可知，当 $r=(i-j)\bmod p$ 时，若 $0\leqslant r\leqslant p-2$，有 $d_{i,<i-j>_p}\in R_i$ 若 $i+r\equiv i+(i-j)=j(\bmod p)$，有 $d_{i,<i-j>_p}\in D_j$，从而可推导 $d_{i,<i-j>_p}\in R_i\cap D_j$，也即 $R_i\cap D_j\neq\varnothing$；由于 p 是素数，有且仅有 $d_{i,<i-j>_p}\in R_i\cap D_j$，即 $|R_i\cap D_j|=1$。

2) 证明方法同 1)。

3) 在矩阵中增加一行虚拟节点，$R_i=\{d_{i,r} \mid 0\leqslant i\leqslant p-1, 0\leqslant r\leqslant p-2\}$，对角线与反对角线可视为斜率为 ±1 的 2 条直线，根据定义 1 可知 $d_{p-1,r}\in D_j$，$d_{p-1,r}\in B_t$，$1\leqslant r\leqslant p-1$，从而得知 $d_{p,r}\in D_j\cap B_t$，也即在虚拟节点层存在 2 条直线交叉的节点，2 条直线相交只有一个交点，故这 2 条直线可能在实际阵列中没有相交的节点。

假设在进行单节点修复时，选用 3 种校验方案进行混合修复，假设有 λ 个行校验组，ρ 个对角线校验组，γ 个反对角线校验组。又假设行校验组与对角线校验组公共节点数为 S_{RD}，行校验组与反对角线校验组公共节点数为 S_{RB}，对角线校验组与反对角线校验组公共节点数为 S_{DB}，总公共节点数为 S。根据引理 1，有

$$S_{RD}=\lambda\rho,\ S_{RB}=\lambda\gamma,\ S_{DB}\leqslant\rho\gamma \tag{14}$$

$$\lambda+\rho+\gamma=p-1 \tag{15}$$

结合式(14)、式(15)有

$$\begin{aligned}S&=(S_{RD}+S_{RB}+S_{DB})\leqslant\lambda\rho+\lambda\gamma+\rho\gamma\\&=(p-1)(\lambda+\rho)-[(\lambda+\rho)^2-\lambda\rho]\\&\leqslant(p-1)(\lambda+\rho)-\frac{3}{4}(\lambda+\rho)^2\\&=\frac{1}{3}(p-1)^2-\frac{1}{3}(p-1)^2+(p-1)(\lambda+\rho)-\frac{3}{4}(\lambda+\rho)^2\\&=\frac{1}{3}(p-1)^2-\frac{3}{4}[\frac{4}{9}(p-1)^2+2\cdot\frac{2}{3}(p-1)(\lambda+\rho)+(\lambda+\rho)^2]\\&=\frac{1}{3}(p-1)^2-\frac{3}{4}[\frac{2}{3}(p-1)-(\lambda+\rho)]^2\\&\leqslant\frac{1}{3}(p-1)^2\end{aligned} \tag{16}$$

总公共节点数的理论上界为 $\frac{1}{3}(p-1)^2$，设混合修复时节点读取数为 W，单一修复数据节点的读取次数为 $(p-1)^2$，则 W 的理论下界为

$$W=(p-1)^2-\frac{1}{3}(p-1)^2=\frac{2}{3}(p-1)^2 \tag{17}$$

由式(17)可知，当采用行校验、对角线校验、反对角线校验三者进行混合修复时，且 3 组的取值相近 $\lambda,\rho,\gamma\in\left(\left\lfloor\frac{p-1}{3}\right\rfloor,\left\lceil\frac{p-1}{3}\right\rceil\right)$，会逼近最少的磁盘读取数 $\frac{2}{3}(p-1)^2$。

4 RS-ERDP 编码的性能分析

4.1 存储开销

假设原文件大小为 1 MB，容错时若采用目前常用的 3 副本完全复制容错方式，则存储空间占用 3 MB；若采用 RS 编码方式，每个编码块的大小为 $\frac{1}{n}$ MB，则存储空间占用 $\frac{m+n}{n}$ MB。

在 RS-ERDP 编码中，具有 $k=\frac{m+n}{p-1}$ 个修复组，每组的源数据的大小为 $(p-1)\frac{1}{n}$ MB，每个修复组增加 3 个校验编码块，则整个文件存储空间大小为

$$S=k\left(\frac{p-1}{n}+\frac{3}{n}\right)=\frac{p+2}{p-1}\frac{m+n}{n}\ \text{MB}$$

若假设原文件大小为 50 MB，3 副本完全复制、RS 编码、RS-ERDP 编码 3 种容错的存储开销比较如图 4 所示。从图 4(a)可看出，当 p 值固定，且 m 不变时，RS 编码、RS-ERDP 编码的存储开销都随着 n 的增大而降低，S 趋向于 $\frac{p+2}{P-1}\cdot 50$ MB；从图 4(b)可看出，当（n,m）固定时，RS-ERDP 编码的存储开销随着 p 的增大而降低，S 趋向于 $\frac{m+n}{n}\cdot 50$ MB。

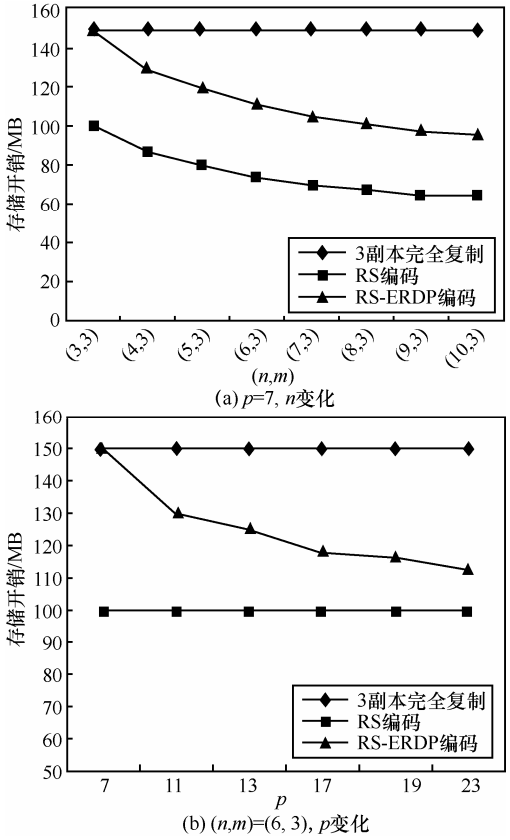

图 4 存储开销对比

4.2 译码效率

为了便于比较，定义每个数据块的大小为 1 bit，译码效率每个数据源信息块的平均异或次数（修复失效数据总异或次数与源数据比特数之比）。对容三错的 RS-ERDP 编码、RS 编码、EVENODD 编码的译码性能进行比较。

由文献[5]得知，容三错 EVENODD 编码异或的总次数为 $(4l_d-2+3p)(p-1)-3$。由文献[17]得知，基于 XOR 的 RS 编码异或的总次数约为 krL^2，k 表示码的信息位，r 表示冗余校验位长度，L 表示有限域的大小，为了便于比较，取相同的冗余校验位 $r=3$。

在容三错 RS-ERDP 编码中，根据式(11)~式(13)容易得知，3 个校验算子总异或次数之和为 $3p^2-11p+6$，将校验算子作为方程组，通过消元计算出恢复三错中一个失效列总异或次数之和为 $3(l_d-1)(p-1)+(p-2)$，剩余两错按常规的 RDP 容双错处理，总异或次数之和为 $4(p-2)-1$ [18]，RS-ERDP 译码总异或次数为上述 3 部分之和。

数据源比特数为 $(p-1)^2$，RS-ERDP 码、RS 码、EVENODD 码的译码效率如图 5 所示。

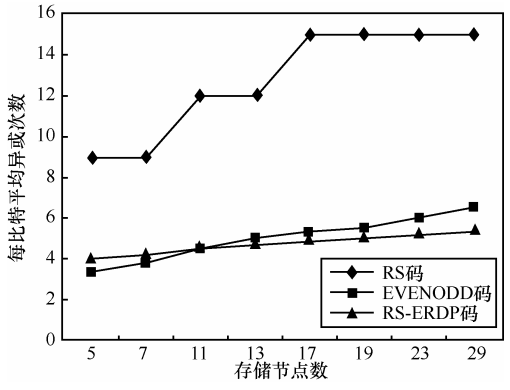

图 5 译码效率比较

从图 5 可看出，当节点数量少（p<11）时，RS-ERDP 码的译码效率在略高于 EVENODD 码，但当节点数增长（p>11）时，RS-ERDP 码的平均异或次数较 EVENODD 码增长更缓慢，而且译码效率要高。RS 码的译码效率最低，主要与有限域的大小相关，并且随着域的扩大而增加。

4.3 单节点故障修复带宽

假设每个编码块的大小为 1 MB，在 RS 码中，

根据 RS（n,m）思想[19]，恢复一个失效数据块，需要获取 n 个数据块，一个节点含有 $p-1$ 个数据块，所以期间若单节点发生故障其修复带宽为 $(p-1)n$ MB，对于传统 RDP 码单故障修复带宽视 p 的大小决定，在 RS-ERDP 码单故障修复时，本文采用混合修复的方法，由于行、对角线、反对角线存在公共节点，修复带宽可进一步减少。RS 码、RDP 码、RS-ERDP 码单节点故障修复带宽如表 2 所示。RS-ERDP 码只是局部单元修复组修复，一般 $p<n$，所以 RS-ERDP 修复带宽最低。

表 2　　单节点故障修复带宽比较

编码	修复带宽/MB
RS 码	$(p-1)n$
RDP 码传统修复	$(p-1)^2$
RS-ERDP 码混合修复	$\frac{2}{3}(p-1)^2$

5　基于 RS-ERDP 编码的云存储容错架构

基于 RS-ERDP 编码的有中心云存储容错架构如图 6 所示。存储客户端将待存储的文件生成 RS-ERDP 编码发至云端文件服务器。文件服务器包括元数据信息管理模块和心跳感知模块。元数据信息管理模块负责处理编码块的名字空间、索引记录、编码块到存储节点映射等方面的信息。心跳感知模块通过心跳感知存储节点是否发生故障，并根据故障节点数来决定采用 RS 编码还是 RS-ERDP 编码，并将相关信息反馈给元数据信息管理模块。心跳感知模块执行编码选择的方案如表 3 所示。文件读取时，客户端首先从服务器获取文件编码块的偏移信息，再从存储集群节点中读取，读取分为正常读取和故障读取 2 种方式。

表 3　　编码选择方案

$m+n$ 中节点故障数(u)	ERDP 中校验节点故障数(v)	方案选择
$u>m$	$\forall v$	无法修复
$3<u\leqslant m$	$\forall v$	RS
$u=3$	$v>0$	RS
$u=3$	$v=0$	RS-ERDP
$u=2$	$v>1$	RS
$u=2$	$v\leqslant 1$	RS-ERDP
$u=1$	$v>2$	RS
$u=1$	$v\leqslant 2$	RS-ERDP

文件服务器是系统架构的核心部分，为增强其可靠性，本文采用双机热备的方式进行工作。两台同构的服务器同时加电工作，一个任务同时发送到主从服务器。正常情况下从服务器不指挥工作，当主服务器发生物理故障时才接管指挥工作。主从服

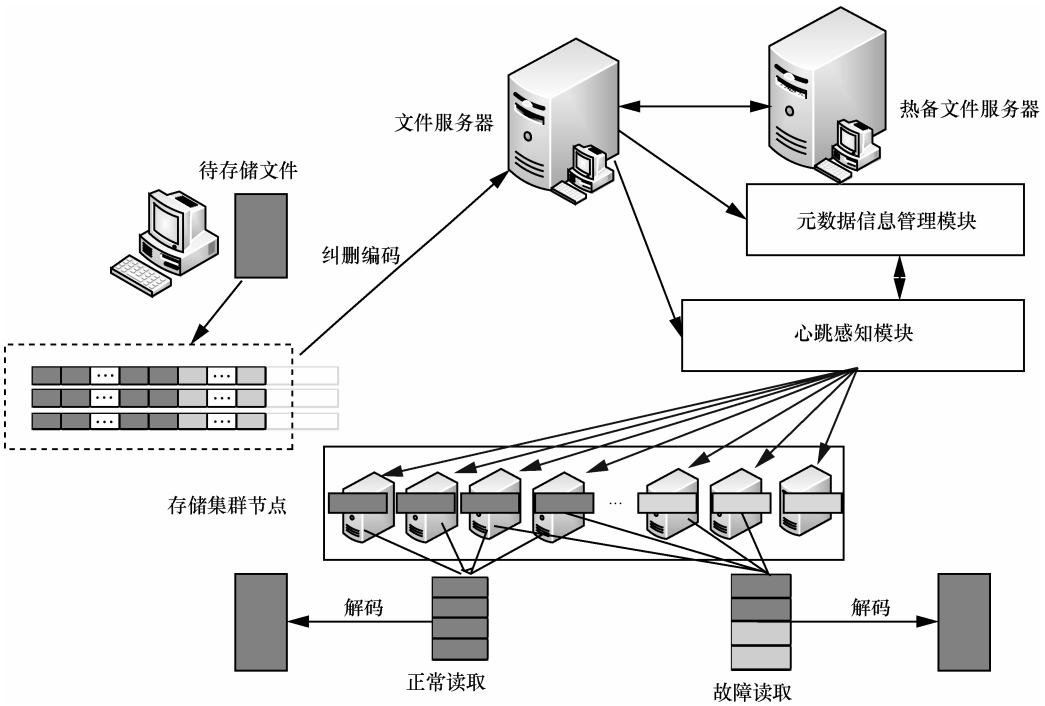

图 6　基于 RS-ERDP 编码的云存储容错架构

务器周期性地做检查点并比较检查点结果（状态）是否相同，若相同，说明系统正常，若不相同，判定其中一台服务器发生软故障，主从服务器同时回卷到上一个检查点状态，重新执行工作。

6 实验性能评估

6.1 实验环境

本实验在开源的分布式存储系统 NCFS 中进行[20]。NCFS 连接各存储设备,并根据配置的不同编码机制将数据分块并按照条带编码分散存储在相关的存储设备中。

实验的硬件环境如表 4 所示。

表 4 实验环境的硬件配置

身份	CPU	硬盘	内存	网络带宽
存储客户端	酷睿 i5，3.3 GHz	WD4T	DDR3, 8 GB	100 Mbit/s
主从文件服务器	酷睿 i7，4.0 GHz	WD 6T	DDR3, 8 GB	100 Mbit/s
存储节点	酷睿 i5，3.3 GHz	WD 4T	DDR3, 8 GB	100 Mbit/s

6.2 节点故障修复时间比较

为了比较不同编码的修复性能，本文在 NCFS 存储系统中分别实现了 RS 码、RS-cauchy 码、RS-ERDP 码，为了便于比较，还构建了容三错的 RS-EVENODD 码。节点失效利用断电的方式离线处理。实验中通过调整 p 值的大小来比较系统的整体修复时间（下载数据时间、修复时间、写数据时间）。实验测试了不同编码在参数 p 值不同的情况下的修复性能。实验结果如图 7 所示。

实验结果表明，在容三错的范围内，由于全部采用只含 XOR 运算的阵列码来容错，RS-ERDP 码的恢复性能较 RS 编码、RS-cauchy 都有明显的提升，而且随着存储节点、故障节点的增多，优化性能越明显。此外，随着存储规模地增大，RS-ERDP 码的修复性能变化平缓，非常适合于大规模的云存储容错。RS-ERDP 码与 RS-EVENODD 码的比较结果是：节点数量小（p<11）时，RS-ERDP 码性能略低；当节点数量增多（p>11）时，RS-ERDP 码性能要高于 RS-EVENODD 码。这是由于 RS-ERDP 编码多一列参与对角线校验，节点数量少时，每比特译码的平均 XOR 次数略高于 RS-EVENODD 码，当节点数增多时，每比特译码的平均 XOR 次数会低于 RS-EVENODD 码，并且 RDP 的小写性能和数据读取平衡性优于 EVENODD。

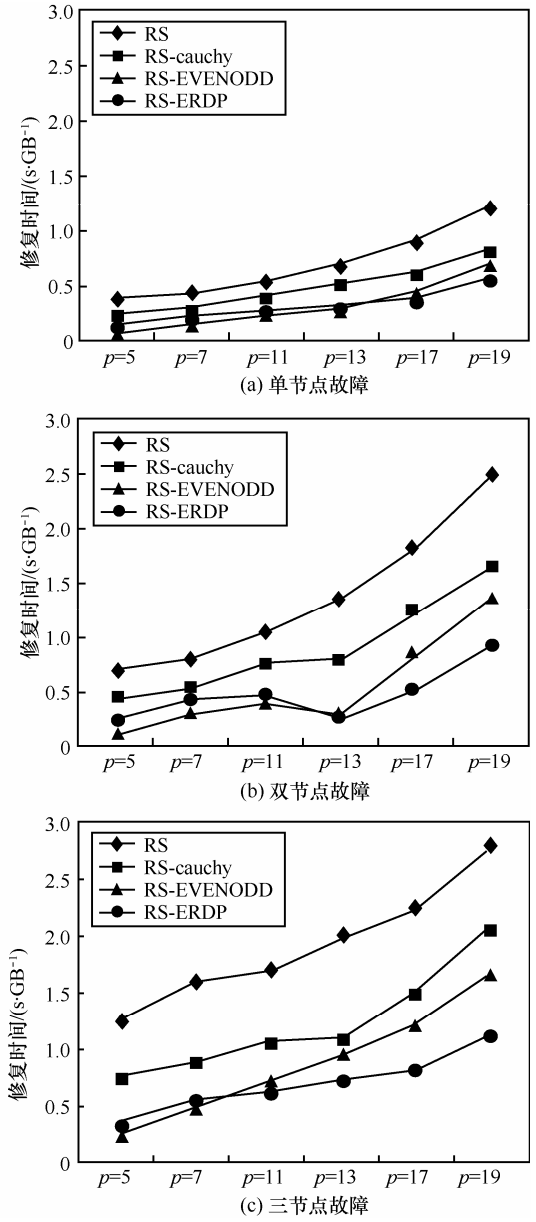

图 7 不同节点数情况下各类编码的恢复性能比较

6.3 单节点故障修复性能比较

针对于单节点故障修复，本文对传统的修复与混合修复进行了比较。实验测试了 RS-ERDP 编码在参数 p 不同的情况下的修复性能。实验中数据块大小分别取值为 1 024 KB、2 048 KB，$p\in\{5,7,13,17,19,23\}$，传统修复只采用行校验，混合修复采用行校验、对角线校验、反对角线校验三者混合修复，且 3 组取值接近 $\lambda,\rho,\gamma\in\left(\left\lfloor\frac{p-1}{3}\right\rfloor,\left\lceil\frac{p-1}{3}\right\rceil\right)$，实验结果如图 8 所示。

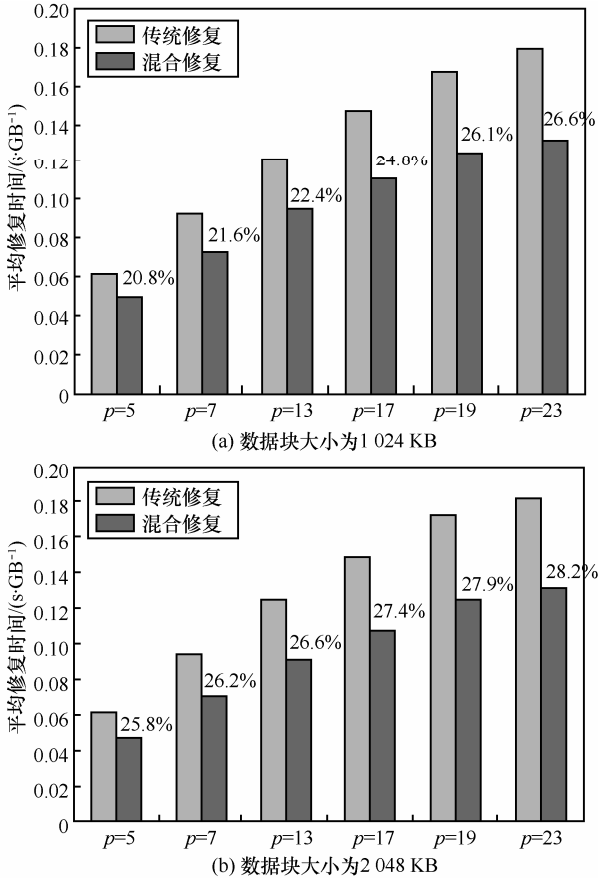

图 8　传统修复与混合修复在不同节点个数情况下的性能对比

柱状图顶端的数字表示优化百分比。实验结果表明，当数据块的大小为 1 024 KB 时，数据修复时间减少 20.8%~26.6%，当数据块的大小为 2 048 KB 时，数据修复时间减少 25.8%~28.2%；从变化趋势可看出，节点个数越多，优化效果越好，这是由于存在更多公共数据块的原因。理论的下界是减少 33.3%，优化的结果离理论下界还有一定的差距，这是由于混合修复时磁头不是连续读，在寻道上有一定的延时。

7　结束语

云存储是大规模的海量存储，必须具有可靠而又高效的容错特性。本文设计了兼备 RS 码与阵列码优良特性的 RS-ERDP 容错纠删码，当发生多个节点故障时可启用内部 RS 编码进行修复，当节点故障在 3 个以下时可启用只含 XOR 运算的 ERDP 码，并讨论了容三错 ERDP 码中单节点故障的磁盘读取问题，通过混合修复降低了磁盘读取数，求出了三容错中磁盘读取的理论下界。从理论和实验 2 个方面对编码的容错性能维护开销进行了分析，结果表明 RS-ERDP 容错纠删码通过少量的额外存储代价可获得准确高效的数据修复效果，适应于对实时性、可靠性要求较高的云存储系统。

参考文献：

[1] 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.

[2] 谭鹏许, 陈越, 兰巨龙, 等. 用于云存储的安全容错编码[J]. 通信学报, 2014, 35(3): 109-115.
TAN P X, CHEN Y, LAN J L, et al. Secure fault-tolerant code for cloud storage[J]. Journal on Communications, 2014, 35(3): 109-115.

[3] LUO J Q, MOCHAN S, XU L H, et al. Efficient encoding schedules for XOR-based erasure codes[J]. IEEE Transactions on Computers, 2014, 63(9): 2259-2272.

[4] LI M, SHU J. On cyclic lowest density MDS array codes constructed using starters[J]. IEEE International Symposium on Information Theory, 2010, 41(3): 1315-1319.

[5] 万武南, 吴震, 陈运, 等. 一种基于 3 容错阵列码的 RAID 数据布局[J]. 计算机学报, 2007, 30(10): 1722-1730.
WAN W N, WU Z, CHEN Y, et al. A data placement based on toleration on triple failures array codes in RAID[J]. Chinese Journal of Computers, 2007, 30(10): 1722-1730.

[6] KVASHENNIKOV V V. Application of fast polynoamial transformations over GALOIS GF(2m) fields in Reed-Solomon coding and decoding[J]. Telecommunications and Radio Engineering, 2012, 71(10): 85-90.

[7] BURGISSER P, CLAUSEN M, SHOKROLLAHI MA. Algebraic complexity theory[M]. Springer Verlag Heidelberg. 1996.

[8] LACAN J, FIMES J. Systematic MDS erasure codes based on Vandermonde matrices[J]. IEEE Communications Letters, 2004, 8(9): 570-582.

[9] PLANK J S, XU L.Optimizing cauchy Reed-Solomon codes for fault-tolerant network storage applications [C]//The 5th IEEE International Symposium on Network Computing and Applications (IEEE NCA06). Cambridge, MA, 2006: 1-8.

[10] KALCHER S, LINDENSTRUTH V. Accelerating Galois field arithmetic for Reed-Solomon erasure codes in storage applications[C]// IEEE International Conference on Cluster Computing. 2011: 290-298.

[11] KHAN O, BURNS R, PLANK J S. Rethinking erasure codes for cloud file systems: minimizing I/O for recovery and degraded reads[C]// USENIX. FAST 2012: 10th USENIX Conference on File and Storage Technologies. San Jose, CA, 2012: 1-14.

[12] 李小兵, 许胤龙, 林一施, 等. 再生码:一类适用于云存储的准确修复编码[J]. 计算机应用与软件, 2014, 31(8): 241-244.

LI X B, XU Y L, LIN Y S, et al. X regenerating codes: a class of accurate repair codes for cloud storage [J]. Computer Applications and Software, 2014, 31(8): 241-244.

[13] PLANK J S. A tutorial on Reed-Solomon coding for fault-tolerance in RAID-like systems[J]. Software: Practice and Experience, 1997, 27(9): 995-1012.

[14] CORBETT P, ENGLISH B, GOEL A, et al. Row diagonal parity for double disk failure correction [C]//Proceedings of the Third USENIX Conference on File and Storage Technologies. Berkeley, CA, USA, 2004: 1-14.

[15] 邱丽娜, 王芳, 李楚, 等. 一种容三盘失效纠删码的单数据盘失效快速重建方法[J]. 计算机学报, 2013, 36(10): 2041-2051.
QIU L N, WANG F, LI C, et al. EDS: a novel scheme for boosting single disk failure recovery of triple erasure correcting code storage systems[J]. Chinese Journal of Computers, 2013, 36(10): 2041-2051.

[16] XIANG L，XU Y, LUI J C S, et al. Optimal recovery of single disk failure in RDP code storage systems[J]. ACM Sigmetrics Performance Evaluation Review[J]. ACM, 2010, 38(1): 119-130.

[17] BLOEMER J M, KALFANE M, KARPINSKI R. An XOR-based erasure-resilient coding scheme[R]. Technical Report at ICSI, 1995.

[18] 万武南, 王拓, 索望. 一种三容错数据布局[J]. 电子与信息学报, 2013, 35(10): 2341-2346.
WAN W N, WANG T, SUO W, et al. A data placement based on toleration triple failures[J]. Journal of Electronics & Information Technology, 2013, 35(10): 2341-2346.

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

[20] HU Y, YU M C, LEE P P C, et al. NCFS: on the praticatity and extensibility of a network-coding based distributed file system[C]// International Symposium on Network Coding. Beijing, 2011: 1-6.

作者简介：

鄢喜爱（1972-），男，湖南长沙人，湖南大学博士生，湖南警察学院教授，主要研究方向为分布式存储、容错计算。

张大方（1959-），男，湖南长沙人，湖南大学教授、博士生导师，主要研究方向为可信系统与网络、系统容错。

杨金民（1967-），男，湖南长沙人，博士，湖南大学教授，主要研究方向为软件工程、系统容错。

张波云（1972-），男，湖南长沙人，博士，湖南警察学院教授，主要研究方向为信息安全、系统容错。