doi:10.3772/j.issn.1002-0470.2019.01.006

一种三容错节能型数据布局①

江培瑞② 孟利民 蒋 维 商宇洲 胡杨添秀

(浙江工业大学信息工程学院 杭州 310023)

摘 要 随着网络通讯技术的迅猛发展,数字化信息呈爆炸式增长,数据存储的可靠性以及数据中心的能耗问题更为人们所关注。基于独立冗余磁盘阵列(RAID)存储系统,本文提出了一种可以恢复任意 3 个失效磁盘的数据且具有一定节能效果的节能交叉奇偶校验(EECPC)数据布局。该方案是在 STAR 码基础上的提高并结合了优化后的 CRUSH 数据布局算法。通过分析,该方案与已有的一些 3 容错阵列码相比具有更小的计算复杂度且具有一定的节能效果,拥有一定的实用价值。

关键词 数据存储,可靠性,独立冗余磁盘阵列(RAID),节能

0 引 言

随着网络技术的飞速发展,数据存储成为了网络信息领域中至关重要的技术。而随着数据量的不断增大,存储系统的规模也随之变大,为解决存储庞大数据量的问题,Patterson[1]在 1988 年首次提出了独立冗余磁盘阵列(redundant arrays of independent disks, RAID)结构。RAID 存储技术,可以大大地提高存储容量,提高系统输入/输出请求处理能力,并且通过数据的分布式存储、并行访问和信息冗余技术提高数据的可靠性。随着存储系统的规模变大,各种各样的不确定因素往往会造成不可预知的系统错误,从而导致数据的遗失,数据的可靠性也随之下降,且存储系统的能耗也会随存储规模的变大而上升。

在 RAID 存储系统的数据可靠性和节能性研究中,其保障 RAID 存储系统的数据可靠性常见于 RAID-5[2]。但是 RAID-5 只能容许一个磁盘失效,随着存储规模的变大相应需要的磁盘数量也变多,多磁盘失效的概率也会随之变大继而数据的可靠性下降,所以 RAID-5 的单盘容错并不能很好地应用于大规模存储环境下,因而学者们研究了基于纠删码的数据布局,以保障数据的可靠性。文献[3]中提出的 EVENODD 码一种经典的纠删码,在 2 个磁盘同时失效的情况下,通过解码算法恢复丢失的数据,该方案的编码和解码均为简单异或运算,具有较小的计算复杂度。文献[4]提出的编码方案,理论上保证了最大距离可分(maximum-distance-separable, MDS)、最优恢复带宽、修复灵活性、系统性和快速编码这几个属性,综合保障了存储系统的数据可靠性。文献[5]介绍了 zigzag decodable (ZD)编码,其编码和解码可在线性时间内完成,并只需简单的异或运算和位操作,是一种可靠性较高的数据布局策略。文献[6]给出了理论上能恢复任意个失效磁盘的数据的里所(Reed-Solomon, RS)码编码算法的完整规范以及实现它的细节,方便了人们在 RAID 存储系统中对该数据布局的实现。文献[7]提出的三独立奇偶校验(three independent parity, TIP)码基于简单的异或运算,并且能接受磁盘阵列中任意 3 个磁盘失效,并恢复其中的数据,从而保障了数据的可靠性。这些方案将基于纠删码的数据布局策略应

① 国家自然科学基金(61372087)资助项目。

② 男,1993 年生,硕士生;研究方向:数据存储,数据安全;E-mail: 2111603011@zjut.edu.cn

(收稿日期:2018-05-15)

用于 RAID 存储系统中虽然可以提升系统中数据的可靠性,但是忽略了对存储系统节能性的考虑。而在 RAID 存储系统节能性的相关研究中,文献[8]提出了一种能量感知策略(striping-based energy-aware,SEA),在保证响应速度的同时,也显著地节省能量。文献[9]提出了一种动态块交换算法,基于磁盘组中所观察到的负载量,在各个磁盘间切换数据,使得频繁访问的数据块最终驻留在几个"热"盘上,从而允许磁盘组中的其他磁盘处于更长的空闲周期以达到节能的目的。文献[10]中提到的方案,通过观察和预测磁盘的负载量,自适应控制磁盘组的功率状态来达到节能的效果。文献[11]提出的(power-aware RAID, PARAID)方案,可根据不同的性能需求,按照不同的档位运行不同数量的磁盘,若性能需求高则切换到高档位,即运行更多的磁盘,反之则切换至低档位,关闭一些磁盘,节约能量。文献[12]提出了 ThinRAID 结构,该方案主要根据数据集的容量,使用主磁盘阵列的一个子集构建自适应的 RAID 阵列,其他非必要的磁盘则被关停以节省能源。文献[13]提出的方案是将磁盘分为热组和冷组,数据迁移只在两个组之间进行从而降低整个系统中数据迁移产生的能耗。虽然这些方案能起到一定的节能效果,但是它们并未考虑到磁盘失效的情况,这样就造成了数据可靠性的下降。

综合考虑数据可靠性和系统的节能性这两个因素,本文提出了一种节能交叉奇偶校验(energy-efficient cross parity check, EECPC)数据布局,该方案是基于 STAR 纠删码[14]并结合了优化后的 CRUSH 数据布局算法,能允许任意 3 个磁盘失效,在一定程度上保证了 RAID 存储系统的数据可靠性且拥有一定的节能性。

1 数据布局模型

在 RAID 存储系统中,数据放置的算法是影响系统可靠性、节能性和可扩展性的重要因素。本节主要论述了将改进后的 CRUSH 数据布局算法融入一种计算复杂度更低的 3 容错纠删码后的数据布局策略。

1.1 CRUSH 数据布局

CRUSH 算法[15]是由 Weil 等在 2006 年提出的一种数据分布算法。该算法基于伪随机 Hash 算法产生确定且均匀的数据分布。此方案能很好地应用于 RAID 存储系统中且具有一定的可靠性和节能性。如在一个由 4 个磁盘构成的磁盘阵列中,设数据副本的分布策略为 select(2,disk),即数据的 2 个副本分别放置在不同的磁盘上,通过 CRUSH 算法产生的数据布局如图 1 所示。

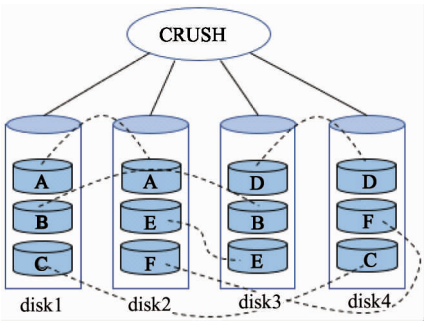

图 1 CRUSH 算法数据布局

在该种数据布局方案下,从节能性分析,此方案存在一定的局限性。如图 1 所示,为达到节能的效果可以关闭任意 1 个磁盘,但是如果关闭 2 个及 2 个以上的磁盘就会导致 A ~ F 中的某些数据的不可用。且在低能耗状态下,即关闭 1 个磁盘的情况下,如果存在其他磁盘因为不可预知的因素而损坏,那么就会导致整个系统中某些数据永久失效的后果。假设在有 n 个磁盘组成的磁盘阵列中,在分布策略 select(a,disk)下,最多可以关闭 $a-1$ 个磁盘,可以达到最大节能比例为$(a-1)/n$。当磁盘阵列的规模变大时,节能效果就越发不明显,而且因磁盘失效导致数据可靠性下降的概率也随之变大。因此本文提出了一种新的数据布局,优化节能比例并提高了数据可靠性。

1.2 EECPC 数据布局

该数据布局是在具有 MDS 性质的 STAR 纠删码的基础上进行改进,将 RAID 存储系统中的磁盘分为源数据磁盘和冗余数据磁盘即校验数据磁盘,且该方案结合了优化后的 CRUSH 数据布局算法。为了简化描述,文中假设在一个由 $m+3$ 个磁盘构成的磁盘阵列中,前 m 个磁盘存储源数据,后 3 个

存储冗余数据，而为了达到 MDS 特性，m 为素数[3]。设在每个源数据磁盘中分别存储 $m-1$ 个数据符号，即构成了一个 $(m-1)\times(m+3)$ 的编码矩阵。本小节先介绍优化后 CRUSH 数据布局算法。

如图 2 所示，在 $m=5$ 的磁盘阵列中，对 disk1 到 disk4 的磁盘进行数据布局优化。若按 1.1 节的 CRUSH 算法中的 select(2,disk) 布局策略，RAID 存储系统处于节能状态时，只能关闭 1～4 号磁盘中的 1 个磁盘。所以文中对 1.1 节中的数据布局进行了优化，该方案将磁盘阵列中的 1 到 $(m-1)$ 号磁盘分为 2 个能耗组 EnergyGroup1 和 EnergyGroup2，因为数据的备份分别位于 2 个不同的能耗组中，所以在关闭任意一个能耗组的情况下，磁盘中的数据都是可用的。在此情况下节能比例达到 $(m-1)/2m$，明显优于 1.1 节中的数据布局方案。具体优化算法描述如下。

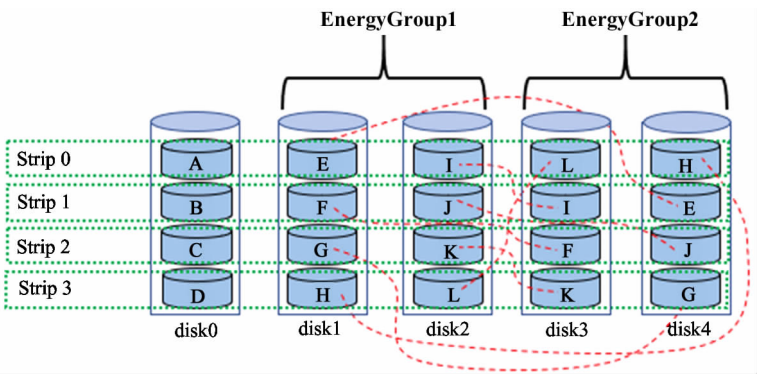

图 2　优化后的 CRUSH 数据布局

优化算法：

$x=(m-1)/2$ # 计算一个能耗组中磁盘数#

for id in [1, x]:

for j in [0, $m-2$]:

if $<n-id+j>_x \neq 0$:# $<x>_m$ 表示运算 x 模 m #

$data[id][S_j] \rightarrow data[<n-id+j>_x + x][S_{<j+1>_{n-1}}]$

$data[id][S_j]$　#表示 disk(id)，条带 Strip(j) 的数据#

Else if $<n-id+j>_x = 0$:

data $[id][S_j] \rightarrow$ data $[<n-id+j>_x + 2x][S_{<j+1>_{n-1}}]$

End

但随着负载的加重，在节能状态下，即只开启 1 个能耗组来响应客户请求。这样势必会造成一定的延时，难以保证用户的体验质量(quality of experience, QoE)。所以在高负载情况下，则同时开启 2 个能耗组来提高服务质量。

在普通模式下，即全部磁盘运行的情况下，此时如果发生多个磁盘同时失效的情况，如 disk0、disk1、disk4 同时损坏，某些数据就会永久丢失，数据的可靠性难以得到保障，本文就融入了纠删码的思想来保证数据的可靠性，在之前的数据优化布局的基础上，介绍编码算法。

同理假设一个 $m=5$ 的磁盘阵列，然后引入 3 个冗余数据磁盘存放冗余数据，构成一个 $(m-1)\times(m+3)$ 的编码矩阵。先定义 $a_{i,j}(0\leqslant i\leqslant m-2, 0\leqslant j\leqslant m+1)$ 表示在第 j 个磁盘中的第 i 个条带上的数据符号。假设在矩阵的最后一行是全为符号 0 的假想行，即 $a_{m-1,j}=0, 0\leqslant j\leqslant m-1$。

在计算冗余数据时，只运用简单异或运算，其编码公式如下。

Disk(m)编码：

$$a_{l,m}=\bigoplus_{t=0}^{m-1}a_{l,t} \tag{1}$$

Disk($m+1$)编码：

$$a_{l,m+1}=\bigoplus_{t=0}^{m-1}a_{<l+t>_m,t} \tag{2}$$

Disk($m+2$)编码：

$$a_{l,m+2}=\bigoplus_{t=0}^{m-1}a_{<l-t>_m,t} \tag{3}$$

EECPC 数据布局多用了 3 个磁盘，用于存储冗余数据，但是在磁盘阵列正常运行的情况下，可以关

闭冗余磁盘,只有在进行数据恢复的时候才会开启它们,所以在正常模式下并不会因为 3 个冗余数据磁盘而增加能源消耗。虽然多用 3 个磁盘增加了存储成本,但是该方案能提高数据的可靠性,减小因磁盘毁坏造成的数据永久丢失带来的损失。

2 数据恢复策略

在 EECPC 数据布局下,有效数据都存储在前 m 个源数据磁盘中,若其中任意 3 个源数据磁盘失效,失效数据都可以通过冗余数据磁盘和 $m-3$ 个有效磁盘中的数据通过简单的异或运算恢复出来。因为用户所需的数据都存储在源数据磁盘中,所以本文主要介绍 3 个源数据磁盘同时失效情况下的数据恢复策略,分为对称情况和非对称情况。假设失效磁盘为 disk(a)、disk(b)和 disk(c),$a<b<c$,$h=b-a$,$k=c-b$,若 $h=k$ 则为对称情况,$h\neq k$ 则为非对称情况。在 2 种不同的情况下,其数据恢复策略的主要思想都是先恢复出 disk(b)中的数据(见 2.1 和2.2 节),然后再重建 disk(a)和 disk(c)中的数据(见 2.3 节)。

2.1 对称情况

本文用具体的实例来直观地描述数据恢复算法,假设 $m=5$,$a=1$,$b=2$,$c=3$,为了方便算法的描述,本文先定义了一些概念。

$$S_m=\bigoplus_{l=0}^{m-2}a_{l,m} \tag{4}$$

$$S_{m+1}=\bigoplus_{l=0}^{m-2}a_{l,m+1} \tag{5}$$

$$S_{m+2}=\bigoplus_{l=0}^{m-2}a_{l,m+2} \tag{6}$$

$$S_{-1}=S_m\oplus S_{m+1}=\bigoplus_{t=0}^{m-1}a_{<t-1>_m,t} \tag{7}$$

$$S_1=S_m\oplus S_{m+2}=\bigoplus_{t=1}^{m-1}a_{m-1-t,t} \tag{8}$$

第 $m+1$ 列算子:

$$c_{i,0}=a_{i,m}\oplus\left(\bigoplus_{t=0}^{m-1}a_{l,t}\right)\quad 0\leqslant i\leqslant m-1,t\neq a,b,c \tag{9}$$

第 $m+2$ 列算子:

$$c_{i,1}=\begin{cases}a_{i,m+1}\oplus\left(\bigoplus_{t=0}^{m-1}a_{<i+t>_m,t}\right) & 0\leqslant i\leqslant m-2\\ S_{-1}\oplus\left(\bigoplus_{t=0}^{m-1}a_{<i+t>_m,t}\right) & i=m-1\end{cases}$$

$$t\neq a,b,c \tag{10}$$

第 $m+3$ 列算子:

$$c_{i,2}=\begin{cases}a_{i,m+2}\oplus\left(\bigoplus_{t=0}^{m-1}a_{<i-t>_m,t}\right) & 0\leqslant i\leqslant m-2\\ S_1\oplus\left(\bigoplus_{t=0}^{m-1}a_{<i-t>_m,t}\right) & i=m-1\end{cases}$$

$$t\neq a,b,c \tag{11}$$

交叉算子:

$$C_{i,a}=c_{<i-a>_m,1}\oplus c_{<i+c>_m,2} \tag{12}$$

$c_{i,1}$ 表示计算 $a_{i,m+1}$ 时,丢失的数据的异或和,如 $a_{i,m+1}=\bigoplus_{t=0}^{m-1}a_{<i+t>_m,t}$,当磁盘 a、b、c 同时失效时,数据 $a_{<i+a>_m,a}$、$a_{<i+b>_m,b}$、$a_{<i+c>_m,c}$ 丢失,$c_{i,1}$ 则表示这 3 个丢失数据的异或和。当 $i=m-1$ 时,对应是编码矩阵中的假想行,所以式(10)中采用分段讨论,式(11)和式(10)同理。交叉算子 $C_{i,a}$ 则表示 2 组丢失数据的异或和,在后面所述算法中结合第 $m+1$ 列算子可求得 disk(b)中丢失的数据。

文中将磁盘阵列抽象为一个 $(m-1)\times(m+3)$ 的矩阵,每一列表示一个磁盘,每一行表示一个数据条带,如图 3 所示,空白列表示失效磁盘,最后一行为假想行。

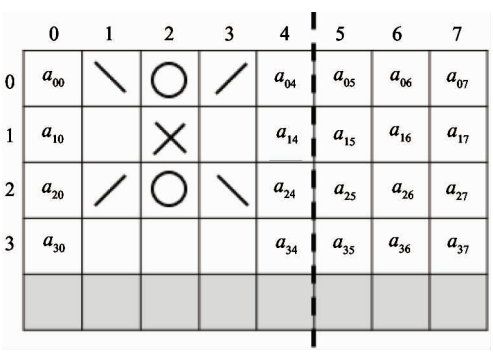

图 3 对称情况解码示意图

当 $i=0$ 时,先求 $C_{0,1}=c_{4,1}\oplus c_{3,2}$,$c_{4,1}=S_{-1}\oplus a_{3,4}=a_{0,1}\oplus a_{1,2}\oplus a_{2,3}$,在图中用斜率为 -1 的短线标注;$c_{3,2}=a_{3,7}\oplus a_{3,0}=a_{2,1}\oplus a_{1,2}\oplus a_{0,3}$,在图中用斜率为 1 的短线标注;为了简略表述,其中涉及到假想行中的数据未予标注。根据式(9),$c_{0,0}=a_{0,5}\oplus a_{0,0}\oplus a_{0,4}=a_{0,1}\oplus a_{0,2}\oplus a_{0,3}$,$c_{2,0}=a_{2,5}\oplus a_{2,0}\oplus a_{2,4}=a_{2,1}\oplus a_{2,2}\oplus a_{2,3}$,所以 $c_{0,0}\oplus c_{2,0}\oplus c_{4,1}\oplus c_{3,2}=(a_{0,1})\oplus a_{0,2}\oplus a_{0,3})\oplus(a_{2,1}\oplus a_{2,2}\oplus a_{2,3})\oplus(a_{0,1}\oplus a_{1,2}\oplus a_{2,3})\oplus(a_{2,1}\oplus a_{1,2}\oplus a_{0,3})=a_{0,2}$

$\oplus a_{2,2}$，图中用圆圈标注。这里定义根据交叉算子解得一个数据异或对的运算，称为一次有效运算。同理根据不同的交叉算子 $C_{i,a}$，可以计算出 $a_{1,2} \oplus a_{3,2}$、$a_{2,2} \oplus a_{4,2} = a_{2,2}$、$a_{3,2} \oplus a_{0,2}$。综上可得一共进行 4 次有效运算，然后逐一得到 $a_{2,2}$、$a_{0,2}$、$a_{3,2}$、$a_{1,2}$，从而恢复出 disk(b)中的数据。算法思路大致如下。

for i in [0, $m-2$]:

figure out $C_{i,a}$

figure out $a_{i,b} \oplus a_{<i+(c-a)>_m,b}$ #根据相关的第 $m+1$ 列算子#

end

$a_{<m-1>_m,b} = 0$

for i in [0, $m-2$]:

figure out $a_{<m-1-(c-a)\times(i+1)>_m,b} \oplus a_{<m-1-(c-a)\times i>_m,b}$

work out $a_{<m-1-(c-a)\times(i+1)>_m,b}$

END

2.2 非对称情况

在非对称情况下，disk(b)的数据恢复策略和对称情况下类似，同样根据交叉算子和第($m+1$)列算子，依次恢复出 disk(b)中的所有数据。与对称情况相比，不同点在于非对称情况下的一次有效运算需要多个交叉算子。可以用下式来解得一次有效运算中所需的交叉算子的个数 l_d。

$$< h + l_d k >_m = 0,\ 0 \leqslant h,\ k < m \tag{13}$$

在一次有效运算中，交叉算子满足后一个交叉算子由前一个交叉算子向下平移 k 个单位所得。同样非对称情况下数据恢复算法可由具体实例直观可得，如图 4 所示。

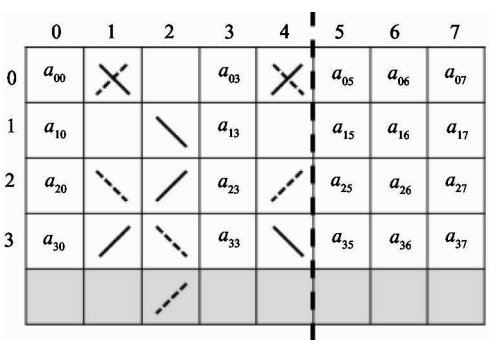

图 4 非对称情况解码示意图

设 $m=5$，$a=1$，$b=2$，$c=4$，可得 $h=1$，$k=2$，$l_d=2$，取第 1 个交叉算子为 $C_{0,1}$，下平移 2 个单位得到第 2 个交叉算子 $C_{2,1}$，这样得到一次有效运算所需的所有交叉算子。根据这 2 个交叉算子可得：$C_{0,1} = c_{4,1} \oplus c_{4,2}$，$c_{4,1} = S_{-1} \oplus a_{2,3} = a_{0,1} \oplus a_{1,2} \oplus a_{3,4}$（斜率为 -1 的线段标记），$c_{1,2} = a_{1,7} \oplus a_{1,0} \oplus a_{3,3} = a_{0,1} \oplus a_{2,4}$（斜率为 1 的线段标记）。所以 $C_{0,1} \oplus C_{2,1} = c_{4,1} \oplus c_{4,2} \oplus c_{1,1} \oplus c_{1,2} = a_{1,2} \oplus a_{2,1} \oplus a_{2,2} \oplus a_{2,4} \oplus a_{3,1} \oplus a_{3,2} \oplus a_{3,4}$，根据第 $m+1$ 列算子 $c_{2,0} = a_{2,0} \oplus a_{2,5} = a_{2,1} \oplus a_{2,2} \oplus a_{2,4}$，$c_{3,0} = a_{3,0} \oplus a_{3,5} = a_{3,1} \oplus a_{3,2} \oplus a_{3,4}$，可得 $C_{0,1} \oplus C_{2,1} \oplus c_{2,0} \oplus c_{3,0} = a_{1,2} \oplus a_{4,2} = a_{1,2}$，同理可得 $a_{2,2} \oplus a_{0,2}$、$a_{3,2} \oplus a_{1,2}$、$a_{4,1} \oplus a_{2,2}$，然后逐一求得 disk(b)中的数据。算法思路如下。

input: a, b, c

figure out h, k, l_d

$a_{<m-1>_m,\,b} = 0$

for i in [0, $m-2$]:

figure out $x = \bigoplus_{t=0}^{l_d-1} C_{i+kt,a}$

figure out $a_{<i+h>_m,b} \oplus a_{<i+kl_d>_m,b}$

#根据相应的第 $m+1$ 列算子#

end

work out $a_{i,b}$ one by one

2.3 双磁盘数据恢复

2.1 节和 2.2 节主要介绍了 disk(b)中的数据恢复策略，将 3 磁盘失效问题简化为 disk(a)和 disk(c)的双磁盘失效问题。本小节主要介绍双盘数据恢复策略。

定义斜 1 对角算子 $S_l^{d_1}$：

$$S_l^{d_1} = \begin{cases} a_{l,m+2} \oplus (\bigoplus_{t=0}^{m-1} a_{<l-t>_m,t}),\ 0 \leqslant l \leqslant m-2 \\ S_1 \oplus (\bigoplus_{t=0}^{m-1} a_{<l-t>_m,t}),\ l = m-1 \end{cases} \tag{14}$$

定义水平算子 S_l^h：

$$S_l^h = \bigoplus_{t=0,\,t\neq a,c}^{m} a_{l,t},\ 0 \leqslant l \leqslant m-1 \tag{15}$$

同时利用水平特征值 S_l^h 和斜 1 对角算子 $S_l^{d_1}$，凭借下列步骤可恢复 disk(a)和 disk(c)的数据。

(1) 设 $\delta \leftarrow <-(c-a)-1>_m$，对于任意 $0 \leqslant t \leqslant m-1$ 都有 $a_{m-1,t} = 0$

(2) 让 $a_{\delta,c} \leftarrow S^{d_1}_{<\delta+c>_m} \oplus a_{<\delta+(c-a)>_m,a}$ 以及 $a_{\delta,a}$

$\leftarrow S_l^h \oplus a_{\delta,c}$

(3) 设 $\delta \leftarrow <\delta-(c-a)>_m$。如果 $\delta = m-1$ 则停止循环,如果 $\delta \neq m-1$ 则回到第(2)步。

所以,根据 2.1、2.2、2.3 小节则可以恢复任意 3 个失效的源数据磁盘中的数据。

3 相关证明

3.1 式(7)和式(8)的证明

$$(\bigoplus_{l=0}^{m-2} a_{l,m}) \oplus (\bigoplus_{l=0}^{m-2} a_{l,m+1})$$

$$= (\bigoplus_{l=0}^{m-2}(\bigoplus_{t=0}^{m-1} a_{l,t})) \oplus (\bigoplus_{l=0}^{m-2}(S \oplus \bigoplus_{t=0}^{m-1} a_{<l+t>_m,t}))$$

$$= (\bigoplus_{l=0}^{m-2}(\bigoplus_{t=0}^{m-1} a_{l,t})) \oplus \underbrace{S \oplus S \cdots \oplus S}_{m-1} \oplus (\bigoplus_{l=0}^{m-2}(\bigoplus_{t=0}^{m-1} a_{<l+t>_m,t}))$$

$$= (\bigoplus_{l=0}^{m-2}(\bigoplus_{t=0}^{m-1} a_{l,t})) \oplus (\bigoplus_{l=0}^{m-2}(\bigoplus_{t=0}^{m-1} a_{<l+t>_m,t}))$$

因为 m 是素数,所以 $(m-1)$ 必为偶数,所以 $\underbrace{S \oplus S \cdots \oplus S}_{m-1} = 0$, 易得:

$$(\bigoplus_{l=0}^{m-2}(\bigoplus_{t=0}^{m-1} a_{l,t})) \oplus (\bigoplus_{l=0}^{m-2}(\bigoplus_{t=0}^{m-1} a_{<l+t>_m,t})) \oplus (\bigoplus_{t=0}^{m-1} a_{<t-1>_m,t}) = 0$$

所以:

$$(\bigoplus_{l=0}^{m-2}(\bigoplus_{t=0}^{m-1} a_{l,t})) \oplus (\bigoplus_{l=0}^{m-2}(\bigoplus_{t=0}^{m-1} a_{<l+t>_m,t})) = (\bigoplus_{t=0}^{m-1} a_{<t-1>_m,t})$$

$$S_{-1} = \bigoplus_{t=0}^{m-1} a_{<t-1>_m,t} = (\bigoplus_{l=0}^{m-2} a_{l,m}) \oplus (\bigoplus_{l=0}^{m-2} a_{l,m+1})$$

同理可得:

$$S_1 = \bigoplus_{t=0}^{m-1} a_{m-1-t,t} = (\bigoplus_{l=0}^{m-2} a_{l,m}) \oplus (\bigoplus_{l=0}^{m-2} a_{l,m+2})$$

3.2 式(13)的证明

在非对称情况的解码算法中,在一次运算交叉解码运算中所需的交叉算子的个数 l_d 满足式(13)。因为 m 是一个素数,根据伽罗华域的相关概念,正整数对 m 取模运算后定义了一个有限域 GF(m)。在这个有限域内让 k^{-1} 表示为 k 的唯一逆元,所以存在唯一 l_d, 使得 $l_d = (m-h)k^{-1}$, 即证得式(13)成立

3.3 EECPC 数据布局的 MDS 特性证明

因为 EECPC 方案是在 STAR 码[14]基础上的改进,降低了计算复杂度,且同样能最大恢复任意 3 个失效磁盘的数据。又因为 STAR 码具有 MDS 特性,且 EECPC 方案中的校验磁盘数也为 3,根据编码理论可得,EECPC 具有 MDS 特性。

4 性能分析

本文所提出的 EECPC 数据布局策略主要以纠删码为基础,将编码复杂度和解码复杂度作为衡量指标,并与已有的三容错 STAR 码[14]和 EEOD 码[16]作比较。在节能性方面,与仅采用 STAR 码的数据布局策略相比,EECPC 也体现出了一定的优势。

4.1 编码复杂度

根据编码式(1)、(2)、(3),以及文献[14,16]可得,编码过程只需要异或运算,这里将每 bit 所需的异或运算作为编码的复杂度。在 CPC 码中,假设每个信息位有 8 bits,源数据磁盘数量为 m,式(1)所需的异或运算次数为 $8(m-1)^2$ 次,式(2)和(3)所需的异或运算次数分别也为 $8(m-1)^2$ 次,所以总共需要 $24(m-1)^2$ 次异或运算,而 CPC 码总共有 $8(m-1)m$ bits,所以 CPC 码的编码复杂度为 $3-1/m$。

同理,由文献[14,16]可得 EEOD 码的编码复杂度为 $3+(m+1)/(m-1)/m$, STAR 码的编码复杂度为 $3+(m-3)/(m-1)/m$,三者的复杂度比较如图 5 所示。

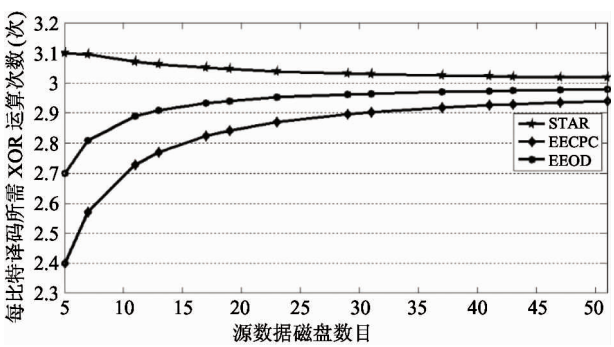

图 5 编码复杂度比较

4.2 解码复杂度

同理解码过程也只需要简单的异或运算,因为 EECPC 数据布局只讨论了 3 个源数据磁盘同时失效的情况,所以这里主要对此情况进行解码复杂度的分析。根据前文提到的数据恢复策略,可得 EECPC 的解码复杂度 O_c 为

$$O_c = 3 - \frac{3}{m} - \frac{2}{m(m-1)} + \frac{2l_d + l_h}{m}$$

文献[14]中定义 l_d 为在一次有效运算中，交叉算子的个数，l_h 为一次有效运算中消元后剩余的行数。根据文献[14,16]可得 3 种译码算法的复杂度如图 6 所示。

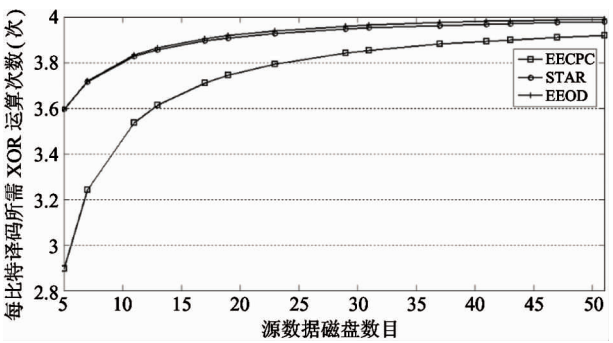

图 6　解码复杂度比较

4.3　节能性分析

本文采用 Disksim，并引入能量指标来分析 EECPC 数据布局的性能。其中磁盘配置参数如表 1 所示。

表 1　磁盘参数配置

操作系统	Linx
磁盘容量	2 TB
转速	7200 rps
闲置功率	4.6 W
运行功率	6.2 W

在仿真实验中，本文将 EECPC 数据布局和采用 STAR 码[14]的数据布局相比较，在 RAID 存储系统运行的时候，因为冗余数据磁盘中的数据非用户可用数据，所以默认冗余数据磁盘处于关闭状态。分析在不同平均负载率的情况下比较EECPC节能模

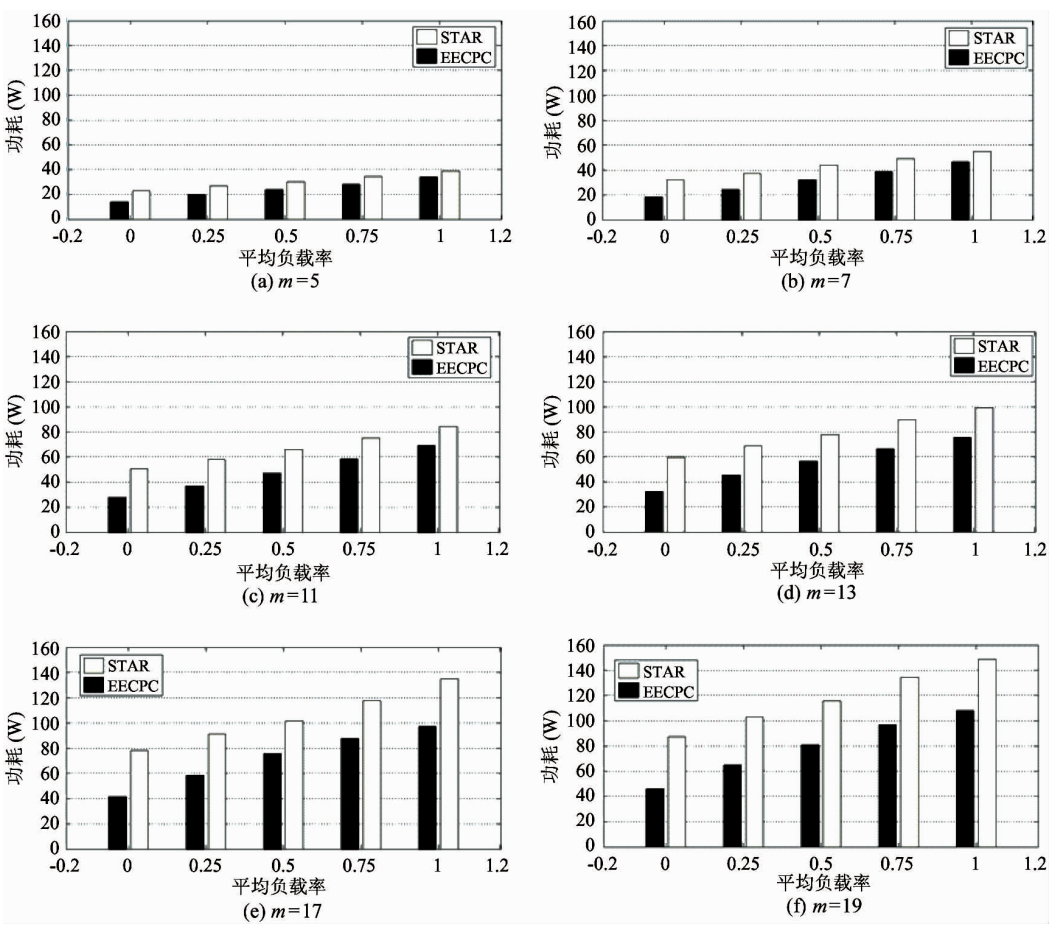

图 7　能耗对比图

式和 STAR 方案的能耗。其中平均负载率指的是在磁盘阵列中,平均每个磁盘的数据量占磁盘总容量的比例。文中给出了不同规模下的 RAID 存储系统的能耗对比图。如图 7 所示。

定义节能率即 EECPC 数据布局策略和 STAR 数据布局的能耗差与 STAR 数据布局的能耗的比例。由图 8 可得,随着 RAID 存储系统的规模增大,节能率随之变大,最后稳定在 0.32 ~0.34 之间。由此可知,EECPC 数据布局策略的节能性较为可观。

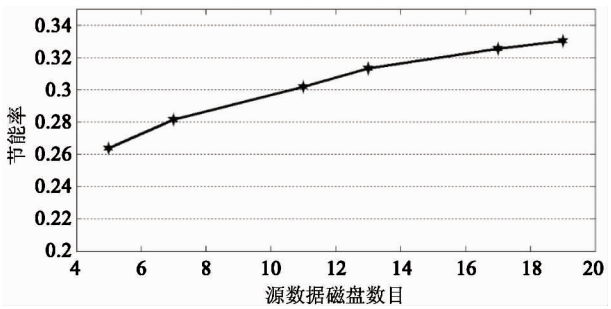

图 8　节能率

图 9 分析对比了 2 种数据布局下数据的平均读取时间,在 EECPC 数据布局下,虽然数据平均存取时间变长,但是毫秒级的变化并不会很大地影响用户的 QoE。该方案是性能和能耗之间的一种折中,以损失少量的服务性能为代价,节省了较多的能源,具有一定的实用意义。

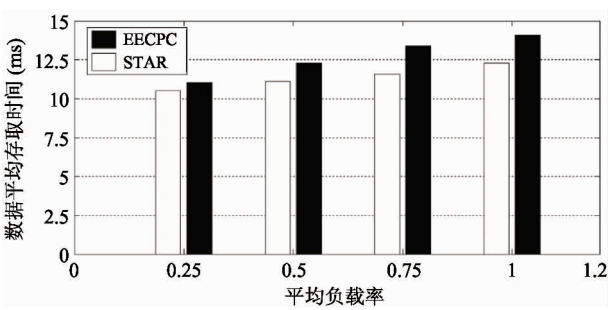

图 9　数据平均存储时间

5　结　论

本文基于 RAID 存储系统,在 STAR 纠删码的可靠性数据布局上融入了节能的思想。本文的 EECPC 数据布局策略,在 STAR 码和 CRUSH 算法的基础上进行了改进,降低了编解码的计算复杂度,达到了一定的节能性,并且与已有的一些 RAID 节能结构相比,能同时恢复 3 个失效磁盘中的数据,提高了系统中数据的可靠性。仿真结果表明,在 EECPC 数据布局的节能模式下,与未采用节能算法的 STAR 数据布局相比,虽然数据平均存取时间有所上升,但是节能率达到了 30% 左右,以较小的代价换取了较大的收益,因此 EECPC 是有一定意义的节能可靠型数据布局。

参考文献

[1] Patterson D A. A case for redundant arrays of inexpensive disks[C]. In: Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data, Chicago, USA, 1988. 109-116

[2] Gibson G A, Hellerstein L, Karp R M, et al. Coding techniques for handling failures in large disk arrays[J]. *Algorithmica*, 1994, 12(2-3):182-208

[3] Blaum M, Brady J, Bruck J, et al. EVENODD: an optimal scheme for tolerating double disk failures in RAID architectures[J]. *IEEE Transactions on Computers*, 1995, 44(2):245-254

[4] Nakkiran P, Rashmi K V, Ramchandran K. Optimal systematic distributed storage codes with fast encoding[J]. *Computer Science*, 2015:430-434

[5] Gong X, Chi W S. Zigzag decodable codes: linear-time erasure codes with applications to data storage[J]. *Journal of Computer & System Sciences*, 2017, 89:190-208

[6] Plank J S. A tutorial on reed-solomon coding for fault-tolerance in RAID-like systems[J]. *Software Practice & Experience*, 2015, 27(9):995-1012

[7] Zhang Y, Wu C, Li J, et al. TIP-Code: a three independent parity code to tolerate triple disk failures with optimal update complextiy[C]. In: Proceedings of the IEEE/IFIP International Conference on Dependable Systems & Networks, Rio de Janeiro, Brazil, 2015. 136-147

[8] Xie T. SEA: a striping-based energy-aware strategy for data placement in RAID-structured storage systems[J]. *IEEE Transactions on Computers*, 2008, 57(6):748-761

[9] Otoo E, Rotem D, Tsao S C. Dynamic data reorganization for energy savings in disk storage systems[J]. *Scientific and Statistical Database Management*, 2010, 6187:322-

341

[10] Kim J, Rotem D. Energy proportionality for disk storage using replication[C]. In: Proceedings of the International Conference on Extending Database Technology, Uppsala, Sweden, 2011. 81-92

[11] Weddle C, Oldham M, Qian J, et al. PARAID: a gear-shifting power-aware RAID[C]. In: Proceedings of the Usenix Conference on File and Storage Technologies, Oakland, USA, 2007. 30-30

[12] Wan J, Qu X, Zhao N, et al. ThinRAID: thinning down RAID array for energy conservation[J]. *IEEE Transactions on Parallel & Distributed Systems*, 2015, 26(10): 2903-2915

[13] Jin P Q, Xie X, Jensen C S, et al. HAG: an energy-proportional data storage scheme for disk array systems[J]. *Journal of Computer Science and Technology*, 2015, 30(4):679-695

[14] Huang C, Xu L. STAR: an efficient coding scheme for correcting triple storage node failures[J]. *IEEE Transactions on Computers*, 2008, 57(7):889-901

[15] Weil S A, Brandt S A, Miller E L, et al. CRUSH:controlled, scalable, decentralized placement of replicated data[C]. In: Proceedings of the ACM/IEEE SC2006 Conference on High Performance Networking and Computing, Tampa, USA, 2006. 122

[16] 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): 1721-1730

An energy-efficient data layout with fault tolerance of three

Jiang Peirui, Meng Limin, Jiang Wei, Shang Yuzhou, Hu Yangtianxiu

(College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023)

Abstract

With the rapid development of network communication technology, the digital information is increasing explosively, the reliability of data storage and the energy consumption of the data center have been concerned more and more. Based on redundant arrays of independent disks (RAID) storage system, a data layout scheme of energy-efficient cross parity check (EECPC) is proposed, which can recover data from any 3 invalid disks and has certain effect of saving energy. This scheme takes into account two factors of data reliability and energy consumption in the storage system, which is improved by STAR erasure code and combined with optimized CRUSH data layout algorithm. The experiment results show that the scheme has smaller computational complexity compared with some existing erasure codes which can tolerate 3 faults, and has certain energy saving effect.

Key words: data storage, reliability, redundant arrays of independent disks (RAID), energy-efficient