双容错数据布局算法 DP-RAID 扩展研究*)

刘卫平　蔡皖东　任建奇
(西北工业大学计算机学院　西安 710072)

摘　要　基于单容错编码的数据布局已经不能满足存储系统对可靠性越来越高的要求。对基于多容错编码的数据布局的研究受到了广泛的关注，并且出现了一些多容错的布局算法，如多维 Parity，DH1，DH2 等。但这些布局算法普遍存在冗余度较差、计算负载大等缺点。DP-RAID 是一种基于水平方向和对角方向双重奇偶校验的双容错数据布局算法。该布局计算负载小，实现简单，但该布局要求校验条纹长度为素数。本文对 DP-RAID 进行扩展，使其能够应用于校验条纹长度为素数减一的环境。与其他双容错布局算法比较表明，该布局算法在保证同样可靠性的情况下，性能有了明显的提高。

关键词　RAID5，奇偶校验，双容错编码，可靠性

Research on Double-Error Tolerance Data Placement Scheme DP-RAID

LIU Wei-Ping　CAI Wan-Dong　Ren Jian-Qi
(College of Computer Science and Engineering, Northwestern Polytechnical University, Xi'an 710072)

Abstract　As the data placement scheme based on single-erasure correcting codes can not satisfy the requirement of storage system on reliability. Now there are more and more researches on the data placement scheme based on multi-erasure correcting codes such as multi-dimension parity, DH1, DH2. But these data placement schemes have some disadvantages such as poor redundancy, heavy computing workload. DP-RAID is a double error tolerating data placement scheme based on horizontal parity and diagonal parity. Its computing load is low, and easy to implement. But it requires that the length of parity stripe is prime number. This paper extends the data placement scheme to the environment that the length of parity stripe is prime number minus 1. The performance of the new data placement scheme improves obviously and reliability of the new data placement scheme doesn't change, comparing with other double-error tolerance data placement scheme.

Keywords　RAID5, Parity, Double-erasure correcting codes, Reliability

1　引言

RAID5[1~3]是一种基于单容错码(Single-Erasure Correcting Codes)的数据布局方式，可以为阵列中的单个磁盘故障提供足够的保护，故障磁盘数据可以通过奇偶校验信息和阵列中其余磁盘中的数据来重构。在数据重构时，整个阵列系统处于降级模式。在降级模式中，RAID5 不再具有容错能力，一旦此时再有磁盘出现故障，则会出现故障磁盘数据不可恢复的状况。

随着网络技术的发展以及多媒体技术的普及，人类社会需要存储的数据信息呈现一种爆炸性的增长。越来越多的应用领域要求使用大容量的存储系统。当存储系统规模较大时，同时发生多个磁盘故障的概率就比较高，基于单容错码的数据布局已不能满足大容量存储系统对可靠性的要求，所以，要求阵列系统能够采用基于多容错码(Multi-Erasure Correcting Codes)的数据布局，来提高大规模阵列系统的数据可靠性并保证系统的高性能。

2　双容错码数据布局研究现状

按照编码理论，如果一种编码能够容许 n 个磁盘同时故障，那么它至少应该拥有 n 个存储校验信息的冗余磁盘[5]。目前，在阵列系统中，针对多容错数据布局的研究工作有很多。具体来说，按照一个条纹组中校验单元的数目，这些工作可被分为两类：单校验条纹和多校验条纹。

单校验条纹布局是指在一个条纹组中只包含一个校验单元的数据布局。单校验条纹大多基于奇偶校验实现。除了 RAID5，还包括 2D-Parity、2D-Parity 改进、3D-Parity 等多维 Parity 布局[6]。这些布局方法虽然可靠性较好，但是冗余度较差。而且这些数据布局或多或少存在着映射函数复杂、难以实现的问题。

多校验条纹布局是指在一个条纹组中包括多个校验单元的布局设计。多校验条纹设计主要包括扩展二进制海明码、MDS 码(如 Reed-Solomon 码)等[5]。传统的二进制海明码可以纠正一个错、检测两个错误。扩展二进制海明码可以纠正两个错误，利用本身所包含的更多的校验信息，扩展海明码还可以纠正三个或者三个以上的错误。但是与奇偶校验编码相比，海明码的计算负载和数据更新代价都很高。同海明码一样，MDS 码也是编码理论中经常使用的一种容错编码，在阵列系统的应用中，MDS 可以提供很高的可靠性。与多维校验数据布局相比，MDS 码可以实现最低的数据冗余度，即使用

*)本文受航空基础科学基金项目(项目号：03F53031)和西安市工业攻关项目(项目号：GG200312)的资助。刘卫平　在读博士生，主要从事网络容灾、网络存储、信息安全等方面研究；蔡皖东　教授，博士生导师，主要从事计算机网络、网络信息安全、网络容灾等方面研究；任建奇　在读硕士生，主要从事网络容灾、信息安全等方面的研究。

C个校验单元可纠正C个同时出现的错误。它的缺点在于计算负载较大。最常见的MDS码是Reed-Solomon码。RAID6就是采用基于Reed-Solomon码的P+Q校验方式，它采用两个校验盘，可以在保证同时有两个磁盘故障时数据不丢失，具有很高的可靠性。但Reed-Solomon编码计算较为复杂，计算负载较大。

此外，基于双奇偶校验的DH1和DH2[7]布局方法，将校验单元散布到阵列中，可以解决校验盘的瓶颈问题。但要求条纹长度为素数，而且两重校验条纹组的大小和校验条纹的长度均不同，导致编码和解码较为复杂。作者提出的基于双奇偶校验的DP-RAID，虽然固定校验条纹长度，具有较好的性能，但也要求条纹长度为素数。因此本文对DP-RAID数据布局进行扩展，使其在一定条件下能够适应条纹长度为偶数的阵列环境，扩展了布局算法的应用领域。

3 Double Parity RAID数据布局及其扩展

3.1 DP-RAID-odd布局算法

由于普通的DP-RAID布局算法要求校验条纹长度为素数，为了与后面介绍的算法区别，将其命名为DP-RAID-odd。

首先对磁盘数为N(N为素数，且$N\geqslant 3$)的RAID5阵列进行变形，使得条纹组中的条纹数为$N-1$。为了实现双容错，另外添加一个校验磁盘，使得整个阵列系统磁盘总数为$N+1$，如图1所示($N=5$)。因此这样一个条纹组可用$(N-1)\times(N+1)$的矩阵M来表示。$M[i,j]$表示第j个磁盘的第i个逻辑条纹单元，其中$1\leqslant i\leqslant(N-1)$，$1\leqslant j\leqslant(N+1)$。

DP-RAID-odd数据布局算法通过沿水平方向和对角方向的两重校验来容许两个磁盘故障。

首先是沿水平方向分布校验条纹：在前N个磁盘中，按照RAID5的布局思想沿水平方向选择数据单元构成校验条纹，并且将校验单元均匀分布在各个磁盘中，条纹分布见图1。

H1	1	1	1	1	
2	H2	2	2	2	
3	3	H3	3	3	
4	4	4	H4	4	

图1 水平方向校验条纹的分布($N=5$)

用H_i($1\leqslant i\leqslant N-1$)表示第$i$个水平校验单元。$H_i$可以用公式(1)来计算，并存入$M[i,N+1-i]$。可知$H_i$由位于同一行中的其他条纹单元进行XOR运算得到。水平校验条纹的长度为N。

$$H_i=\bigoplus_{j=1}^{N}M[i,j]\quad 1\leqslant i\leqslant(N-1),\ j\neq -i \tag{1}$$

其次是沿对角方向分布校验条纹：将前N个磁盘作为数据区，按照正向对角的方向选择数据单元来构成条纹，并将校验单元依次存入校验磁盘。为了便于实现，规定条纹长度与水平校验条纹长度一样，条纹分布见图2。

	1	2	3	4	
4		1	2	3	P1
3	4		1	2	P2
2	3	4		1	P3
					P4

图2 对角方向校验条纹的分布($N=5$)

用P_i($1\leqslant i\leqslant N-1$)表示第$i$个对角校验单元。$P_i$可以用公式(2)来计算，并存入$M[i,N+1]$。可知$P_i$由正向对角线方向的条纹单元进行XOR运算得到。对角校验条纹的长度也为N。

$$P_i=\begin{cases}\bigoplus_{j=1}^{N-1}M[j,j+i]\cdots(i+j)\leqslant N\\ \bigoplus_{j=1}^{N-1}M[j,(j+i)\bmod N]\cdots(i+j)>N\end{cases} \tag{2}$$

3.2 条纹长度为偶数的布局算法DP-RAID-even

3.1节中给出的布局算法要求阵列的校验条纹长度为素数，这一特点限制了算法的应用领域。本节给出的DP-RAID-even是对DP-RAID-odd布局算法的扩展，突破了校验条纹长度为素数这一限制。

首先选择磁盘数为N($N+1$为素数，且$N\geqslant 4$)的RAID5阵列，规定条纹组中的条纹数为N。此外添加一个校验磁盘，使得系统磁盘数为$N+1$，如图3所示($N=4$)。因此这样一个条纹组可用$N\times(N+1)$的矩阵M来表示。$M[i,j]$表示第j个磁盘的第i个逻辑条纹单元，其中$1\leqslant i\leqslant N$，$1\leqslant j\leqslant(N+1)$。

与DP-RAID-odd算法一样，仍然沿水平方向和对角方向来分布校验条纹。

首先是沿水平方向分布校验条纹：在前N个磁盘中，按照RAID5的布局思想沿水平方向选择数据单元构成校验条纹，并且将校验单元均匀分布在各个磁盘中，条纹分布见图3。

H1	1	1	1	
2	H2	2	2	
3	3	H3	3	
4	4	4	H4	

图3 水平方向校验条纹的分布($N=4$)

用H_i($1\leqslant i\leqslant N$)表示第i个水平校验单元。H_i可以用公式(3)来计算，并存入$M[i,N+1-i]$。可知H_i由同一行中的其他条纹单元进行XOR运算得到。数据区有N个磁盘，所以水平校验条纹的长度为N。

$$H_i=\bigoplus_{j=1}^{N}M[i,j]\quad 1\leqslant i\leqslant N,\ j\neq i \tag{3}$$

其次是沿对角方向分布校验条纹：将前N个磁盘作为数据区，按照正向对角的方向选择数据单元来构成条纹，并将校验单元依次存入校验磁盘。为了便于实现，规定条纹长度与水平校验条纹长度一样，条纹分布见图4。

	1	2	3	
4		1	2	P1
3	4		1	P2
2	3	4		P3
				P4

图4 对角方向校验条纹的分布($N=4$)

用P_i($1\leqslant i\leqslant N$)表示第i个对角校验单元。P_i可以用公式(4)来计算，并存入$M[i,N+1]$。可知P_i由正向对角线方向的条纹单元进行XOR运算得到。对角校验条纹的长度也为N。

$$P_i=\begin{cases}\bigoplus_{j=1}^{N-1}M[j,j+1]\cdots\cdots(i+j)\leqslant N\\ \bigoplus_{j=i}^{N-1}M[j+1,(j+i)\bmod N]\cdots\cdots(i+j)>N\end{cases} \tag{4}$$

由图4可以看出，对于任意一个对角条纹，其中的 N 个条纹单元均匀分布在 N 个磁盘中，而且没有两个条纹分布在同样的 N 个磁盘中，这两个特点使得该数据分布具备了双容错的能力。此外，对角校验条纹没有包含水平校验条纹中的校验单元，这一特点大大提高了该布局算法的小数据写性能。

3.3 数据重构

数据重构可分为三种情况：

(1)单个磁盘出现故障。如果是数据区中任意一个磁盘发生故障，那么可根据水平条纹重构故障磁盘数据。如果是校验磁盘发生故障，则可根据对角条纹重构校验磁盘数据。

(2)数据区中一个磁盘和校验磁盘出现故障。此时首先根据水平条纹重构数据区中故障磁盘数据，然后根据对角条纹重构校验磁盘数据。

(3)数据区域中两个磁盘出现故障。这种情况下的数据重构算法最为复杂，具体算法如下：

定义阵列中磁盘编号为 $1\sim N+1$。其中，$1\sim N$ 为数据磁盘，$N+1$ 为校验磁盘。假设数据区中两个磁盘 i,j 同时出现故障($1\leqslant i<j\leqslant N$)，根据下面步骤可重构磁盘数据：

```
k=j−i, a=k, b=N−k+1, x=1;
do {
    if (j>a)
        根据对角条纹 P(j−a)恢复 M[a, j];
    else
        根据对角条纹 P(N+j+1−a)恢复 M[a, j];
        根据水平条纹 Pa 恢复 M[a, i];
    if (a=i)
        break;
    a=a+k;
    if (a>N)
        a=(a−1) mod N;
}
While (a≠i)
if (x≤N)
{
    do {
        if (i>b)
            根据对角条纹 P(i−b)恢复 M[b, i];
        else
            根据对角条纹 P(N+i+1−b)恢复 M[b, i];
        根据水平条纹 Pb 恢复 M[b, j];
        if (b=j)
            break;
        if (b>k)
            b=b−k;
        else
            b=b+N+1−k;
    }
    While (b≠j)
}
```

3.4 校验单元散布

DP-RAID-even 布局算法采用了单独的校验磁盘。因此，频繁写数据时，校验盘的负载过重，会造成系统的瓶颈。因此按照校验散布的思想，将校验盘中的部分的数据散布到阵列中其它磁盘当中，可以实现阵列中各磁盘之间的负载均衡。

该布局中，水平校验条纹中的校验单元已均匀散布到前 N 个磁盘中。但是对角校验条纹中的 N 个校验单元 $P_1\sim P_N$ 集中分布在校验盘中。本文将这 N 个校验单元的一半散布到前 N 个磁盘当中。具体的散布思想如下：从 $P_1\sim P_N$ 中选择一半的校验单元，将选择的校验单元分别与各自所在条纹中并且与自己处于同一行的数据单元进行置换，即可将选择的校验单元散布到前 N 个磁盘中，可得图5所示布局。

校验散布算法：

```
for (i=1; i≤N/2; i++)
{
    k=2i−1;
    if (2k≤N)
    {
        Cache=M[k, 2k];
        M[k, 2k]=M[k, N+1];
        M[k, N+1]=Cache;
    }
    if (2k>N)
    {
        Cache=M[k, (2k−1)mod N];
        M[k, (2k−1)mod N]=M[k, N+1];
        M[k, N+1]=Cache;
    }
}
```

由于该散布仅限于同一条纹内单元的置换，而且没有跨行置换，因此只要对进行过置换的单元做好标记，就不会对水平校验和对角校验运算产生影响。因此，校验单元的这种散布方式不会影响数据重构，而且该置换算法简单易行，不会增加系统的实现复杂度。

	P1	2	3	1
4		1	2	P2
P3	4		1	3
2	3	4		P4

图5 校验单元散布后的布局

4 DP-RAID 可靠性与性能分析

4.1 可靠性分析

Gibson 计算出了 RAID5 数据布局的 MTTDL(mean time to data loss)。$MTTDL_{RAID5}=\frac{MTTF_{disk}^2}{N(N+1)MTTR_{disk}}$，其中 N 是条纹中数据单元的数目；$MTTF_{disk}$ 是磁盘的平均生命周期；$MTTR_{disk}$ 是在阵列系统中，从发现一个磁盘故障，系统进入降级模式，到故障被排除、数据恢复，系统恢复正常的平均时间[1,3]。

本文按照 Gibson 的思路来计算 DP-RAID 的 MTTDL[8]。

$$MTTDL_{DP\text{-}RAID}=\frac{MTTF_{disk}^3}{N(N+1)(N+2)MTTR_{disk}^2}$$

将 DP-RAID 的 MTTDL 与标准 RAID5 的 MTTDL 比较，得

$$\frac{MTTDL_{DP\text{-}RAID}}{MTTDL_{RAID5}}=\frac{MTTF_{disk}}{(N+2)MTTR_{disk}} \tag{5}$$

按照通常的标准，$MTTF_{disk}$ 一般为 500000h，$MTTR_{disk}$ 一般分为 1h 和 12h 两种情况[4]。根据公式(5)可得图4。阵列中磁盘数较少时，即 N 较小的情况下，DP-RAID-even 的可靠性是 RAID5 的数千倍。即使因为某些意外，使得 $MTTR_{disk}$ 增大，或者构成阵列的磁盘数大幅度增加的情况下，DR-RAID 的可靠性仍然可以达到标准 RAID5 可靠性的数百倍。可见，DR-RAID-even 的可靠性要远远高于 RAID5 的可靠性。

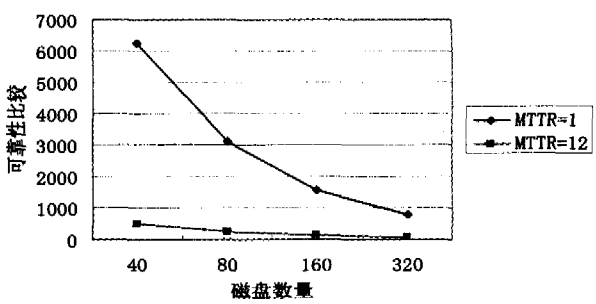

图6 DP-RAID 与 RAID5 可靠性比较示意图

4.2 性能分析

DP-RAID系列布局算法与文[7]中提出的两种双容错算法HD1、HD2的思想比较相似，但是在校验条纹长度、校验数据的分布上存在差异。因此本文从数据读、写操作的角度对DH系列布局算法和DP-RAID系列布局算法的效率进行比较，来说明DP-RAID系列布局算法的性能特点。

在进行读操作的时候，不涉及到对校验单元的写，也就不会有额外的计算负载。因此在进行读操作的时候，两种布局方式具有完全相同的性能。即使与同类型的RAID5相比，其性能也没有下降。

在进行大数据写的时候，DP-RAID系列布局算法的性能与DH系列布局算法基本相同。但是下降与条纹长度相等的RIAD5相比，二者之比为$\frac{G}{G+1}$（G为校验条纹的长度）。可知其性能略有下降，但是下降幅度较小。尤其当阵列规模较大（G较大）时，其性能非常接近于RAID5。

而DP-RAID系列布局算法与DH系列布局算法在小数据写的情况下，性能比RIAD5都有一定幅度的下降。这主要是因为一个小写操作会引起多个校验条纹中条纹单元的多次读和写操作。

DP-RAID-even与DH1布局算法在小数据写情况下的效率比为$\frac{2N-3}{3N-4}$（N为阵列中磁盘数量），由此可得图7。可见，DP-RAID-even在小数据写时的效率远好于DH1。其原因主要是因为DH1布局算法中，水平校验条纹中的校验单元也被当作对角校验条纹的数据单元。因此，对任意一个数据单元的写操作，除了会引起其所在的两重校验条纹中对应的校验单元的写操作之外，由于水平校验单元也是对角校验中的数据单元，因此水平校验单元的写操作还会额外引起一个对角条纹的读写操作。即在DH1数据布局算法中，一个小写操作会涉及到三个校验条纹的读写操作。而在DP-RAID-even布局算法中，水平校验条纹中的校验单元没有参与到对角校验条纹中。那么任意一个数据单元的写操作，只会引起其所在的两重校验条纹中对应的读写操作，因此在小写操作时负载较小。

DP-RAID-odd与DH2布局算法在小数据写情况下的效率基本相等。其原因主要是在DP-RAID-odd和DH2布局算法中，水平校验单元都没有参与对角校验。那么任意一个数据单元的写操作，只会引起其所在的两重校验条纹中对应的读写操作。因此，从小写负载的角度来讲，DP-RAID-odd与DH2布局算法的小写性能基本相等。

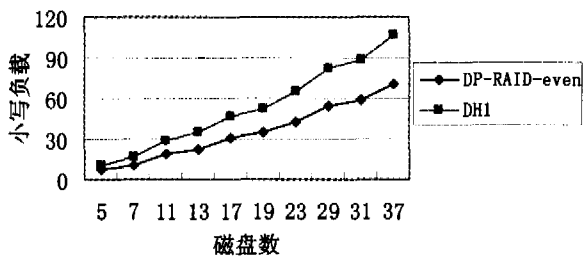

图7 DP-RAID-even与DH1布局算法小数据写效率比较

根据上面的比较，可知DP-RAID-even算法在小数据写操作时性能虽然比DH1有很大的改进，但并没有彻底解决双奇偶校验布局算法在小数据写操作时性能较差的问题。但上述的比较仅仅考虑的是一个小写操作引起的其它逻辑单元读写的次数，是一种最坏状况下的比较。而在阵列系统的具体实现时，适当的缓存机制和磁盘间读写操作的并发执行，可以大幅度改善双奇偶校验布局算法的小数据写性能。此外，DP-RAID系列布局算法中的两重校验条纹长度相等，与两重校验条纹长度不相等DH系列布局算法相比，不仅简化了系统的实现，而且有助于通过完整条纹读写机制来解决小数据写问题。

4.3 空间率用率

DP-RAID系列布局算法要求阵列系统中包含两个校验磁盘。这一特点对磁盘空间利用率有少许的影响。传统的RAID5布局方法的空间利用率是$(N-1)/N$，DP-RAID的空间利用率是$(N-2)/N$。对于小规模的阵列系统来讲，DP-RAID的空间利用率有所下降。但是小规模的存储系统包含的磁盘数量较少，那么同时出现两个磁盘故障的概率也很低，因此不需要采用DP-RAID布局。而对于大规模的阵列系统，系统中包含很多的磁盘，那么因DP-RAID布局而导致的空间利用率的下降就非常微小。与此同时，包含多个磁盘的大规模阵列系统，同时发生两个磁盘故障的几率会急剧增加，所以在大规模存储系统中采用DP-RAID布局方式刚好可以缓解这一问题。

结论 传统的基于单容错码的数据布局方式，如RAID5，只能容许阵列系统中的一个磁盘发生故障，已经不能满足大容量存储系统对可靠性提出的越来越高的要求。DP-RAID-odd是一种基于双奇偶校验的能够容许两个磁盘错误的双容错数据布局。与同类型的双容错布局，如RAID6相比，DP-RAID-odd布局在提供相同可靠性的前提下，计算负载小，具有很好的实用性。但是目前，该布局方式要求条纹长度为素数，具有一定的局限性。因此本文对该布局进行了一定的扩展，使其能够适应校验条纹长度为偶数的阵列环境，得到了预期的效果。与同类的双奇偶校验布局算法DH1和DH2相比，DP-RAID-odd和DP-RAID-even算法在性能上有较大幅度的提高，而且编码解码简单，易于实现。

参考文献

1 Patterson D A, Gibson G, Katz R H. A Case for Redundant Arrays of Inexpensive Disk (RAID). In: Proc. ACM SIGMOD, 1988. 109～116

2 Patterson D A, Chen P M, Gibson G, et al. Introduction to Redundant Arrays of Inexpensive Disk (RAID). In: Proc. IEEE COMPCON, 1989. 112～117

3 Chen P M, Lee E K, Gibson G, et al. RAID: High Performance, Reliable Secondary Storage. ACM Computing Surveys, 1994, 26 (2): 145～185

4 Gibson G, Hellerstein L, Karp R M, et al. Coding Techniques for Handing Failures in Large Disk Arrays. ASPLOS Ⅲ, 1989

5 MacWilliams F J, Sloane N J A. The Theory of Error-Correcting Codes. North-Holland Mathematical Library, 1977

6 Gibson G A, Hellerstein L, Karp R M, et al. Failure Correction Techniques for Large Disk Arrays. In: International Conference on Architectural Support for Programming Language and Operating System, 1989. 123～132

7 Lee N K, Yang S B, Lee K W. Efficient Parity Placement Schemes for Tolerating up to Two Disk Failures in Disk Array. Journal of Systems Architecture, 2000, 46: 1383～1402