第33卷 第15期 计 算 机 工 程 2007年8月
Vol.33 No.15 Computer Engineering August 2007
·博士论文· 文章编号：1000—3428(2007)15—0023—03 文献标识码：A 中图分类号：TP311

基于CRS算法的高可靠性存储系统的设计

那宝玉，张毓森
(解放军理工大学指挥自动化学院，南京 210007)

摘 要：阐述了国际上典型的数据可靠性算法，分析了CRS算法在存储系统中的编、解码原理，基于CRS算法实现数据存储中间件，结合CRS算法和网格技术提出了高可靠性存储系统的整体架构，在理论上对系统的可靠性进行了证明，在局域网中对存储系统的性能进行了测试。结果表明，基于CRS算法实现的存储系统在保证数据存储可靠性的同时具有极高的存储性能。

关键词：磁盘阵列；Reed-Solomon码；柯西Reed-Solomon码

Design of High Reliability Storage System Based on CRS Algorithm

NA Bao-yu, ZHANG Yu-sen
(Institute of Command Automation, PLA University of Science and Technology, Nanjing 210007)

【Abstract】 Several main data reliability algorithms are described. Encode and decode principles of CRS algorithm used in storage system are analyzed. Principle of realization of data storage middleware based on CRS is presented. Based on CRS and grid technology, a whole structure of a high reliability storage system is proposed. System reliability is proved theoretically. Storage system performance is tested in LAN environment. Results show that the CRS algorithm based storage system not only ensures data storage reliability but possesses high storage performance.

【Key words】 RAID; Reed-Solomon codes; Cauchy Reed-Solomon codes

RAID是一种磁盘冗余技术，典型的RAID系统包括RAID5和RAID6，分别解决1块和2块的磁盘损失时的数据恢复问题。

随着对存储容量的需求不断增加，需要成百上千块磁盘组成大规模的存储系统，这种情况下仅通过RAID5和RAID6很难保证数据的可靠性。针对这种情况，国际上提出了很多数据存储可靠性算法，解决多块磁盘损失下的数据恢复问题[1]。文献[2~4]提出的RS算法作为真正意义上的MDS算法，保证n+m块磁盘中任意损失m块数据还是可以恢复的，由于实现简单，受到了欢迎，无论在软件方面还是硬件方面都有了大量应用。文献[5]提出了CRS算法，对RS算法进行了改进，取消了GF域，只使用异或运算进行乘除操作，在保证数据可靠性的同时在时间性能上较RS算法有了很大提高。文献[6]中提出的STAR算法，可以解决3个磁盘同时损失情况下数据恢复问题。文献[7]中提出的WEAVER算法和在文献[8]中提出的HoVer算法可以解决3个以上磁盘错误，因为其特有的编码方式受到极大的重视。Tornado编码作为LDPC编码的一种，使用二向树进行编码，特别当磁盘量增大时，Tornado码的编码和解码速度更快，但是某些数据丢失情况下，数据可能无法恢复。

由于CRS算法保证在任意容错度下算法仍具有MDS特性，并且具有很高的编解码性能，选取CRS算法应用于存储系统中，保证系统具有高度的容错性和存取性能。

1 CRS算法在数据存储中的应用

1.1 基于CRS算法的数据存储机制

CRS算法在存储系统中的作用是恢复“可定位错误”，当一个设备崩溃时，它就会关闭，系统可以实时地察觉到这种行为。CRS算法在网络存储中的实现思想如下：

将对应用于网络存储中的CRS算法命名为CRS-RAID，CRS-RAID(n, m)表示共有n块数据盘和m块校验盘的存储系统，用D_j表示第j块数据盘，C_i表示第i块校验盘。为了计算得出C_i $(1\le i\le m)$中的内容，共需要m个校验函数$F_i()(1\le i\le m)$对所有的数据盘进行校验计算。算法的实施以word为基本单元，每个word的大小是w bits，w的大小可以根据具体的系统来定，每个磁盘被划分为若干个word。大小为k字节的磁盘包含的word数为

$$l=(\text{kbytes})\left(\frac{8\text{bits}}{\text{byte}}\right)\left(\frac{1\text{word}}{\text{wbits}}\right)=\frac{8k}{w}\ \text{words}$$

每块校验盘的校验数据产生，如图1所示。

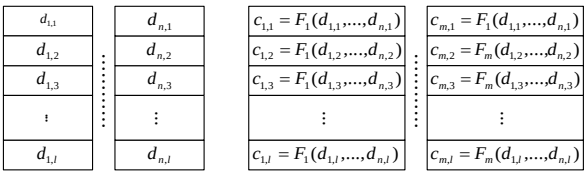

图1 CRS-RAID中校验数据的计算过程

图1中，定义$d_{i,j}$为第i块磁盘的第j个word。校验码的计算公式为$c_{i,j}=F_i(d_{1,j},d_{2,j},d_{3,j},\cdots,d_{n,j})=\sum_{i=1}^{n}d_{i,j}f_{j,i}$。

定义矩阵$\boldsymbol{B}$为$m\times n$阶的柯西矩阵，$\boldsymbol{F}$为$n\times n$阶单位矩阵，$\boldsymbol{D}$为数据矢量，$\boldsymbol{C}$为校验矢量，则有

$$\begin{pmatrix}\boldsymbol{F}\\ \boldsymbol{B}\end{pmatrix}\boldsymbol{D}=\begin{pmatrix}\boldsymbol{D}\\ \boldsymbol{C}\end{pmatrix}\tag{1}$$

式(1)可以用来计算校验盘中的数据。

基金项目：国家自然科学基金资助项目(60403043)
作者简介：那宝玉(1979－)，男，博士研究生，主研方向：网络存储，数据容灾，安全操作系统；张毓森，教授、博士生导师
收稿日期：2007-03-13 **E-mail：**nby627@163.com

当系统中有磁盘崩溃且崩溃磁盘数小于等于 m，则可以根据下面的方法进行数据的恢复：

令 $\boldsymbol{K}=\begin{pmatrix}\boldsymbol{F}\\ \boldsymbol{B}\end{pmatrix}$，$\boldsymbol{X}=\begin{pmatrix}\boldsymbol{D}\\ \boldsymbol{C}\end{pmatrix}$，则式(1)变为 $\boldsymbol{KD}=\boldsymbol{X}$。当系统中有磁盘崩溃时，相应的在矩阵 $\boldsymbol{K}$ 和 $\boldsymbol{X}$ 中删去对应的行，在删除后的矩阵 $\boldsymbol{K}$ 中任意选择 n 行形成一个 $n\times n$ 矩阵 $\boldsymbol{K'}$，得到等式 $\boldsymbol{K'D}=\boldsymbol{X'}$，通过高斯消元法解得矢量 $\boldsymbol{D}$。这样就恢复了数据盘中的内容，利用式(1)重新计算校验盘中数据，整个系统中丢失磁盘数据恢复完毕。

1.2 CRS-RAID 数据存取中间件的实现

结合 CRS 算法与 RAID 技术开发的 CRS-RAID 数据存储中间件体系结构如图 2 所示。

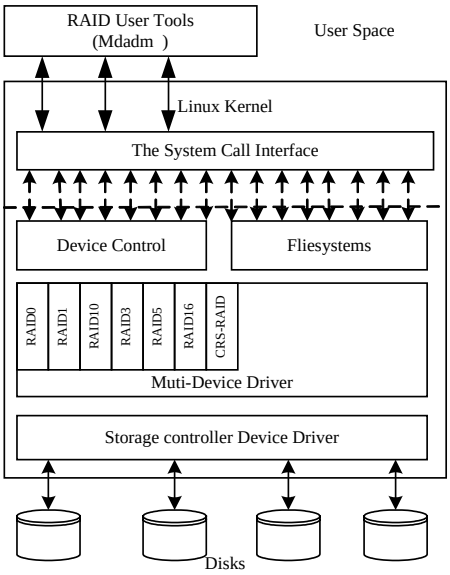

图 2　中间件体系结构

CRS-RAID 的实现借助于 Linux 内核中 Block 设备层中称为 MD(Multi-Device Driver)的特殊 Block 设备，在 MD 之下又形成了一个逻辑层，来管理不同级别的 RAID 设备。MD 在设备驱动中对应一个 Major Number，下面不同的 RAID 设备每个都有一个 Minor Number，这样，每个 RAID 设备都由(Major Number，Minor Number)唯一决定。在 MD 之下建立设备 CRS-RAID，其特性在结构 crsraid_personality 中进行定义，包括对 CRS-RAID 的所有操作。设备初始化时，通过调用 MD 中的注册函数把 CRS-RAID 注册成为 MD 下面的一个逻辑设备。

CRS-RAID 设备具有热插拔、任意添加删除磁盘和错误自动检测等特性。

2　基于 CRS 算法的高可靠性存储系统整体架构

海量高可靠性存储系统整体架构如图 3 所示，网格中心由目录服务器、调度服务器和 CA 认证中心等构成。目录服务器登记并动态反映所有可用的存储服务器和 RAID-M 存储器的状态；调度服务器对存储资源进行统一管理，具有负载均衡和就近服务功能。存储服务器接受来自用户或应用的存储访问请求，并利用目录服务器中所保存的全局信息，将请求转化为对(n+m)台 RAID-M 存储器的具体访问，通过 CRS-RAID 对用户存储数据进行编码后发送到选定的 RAID-M 存储器；CA 认证中心进行所有服务器、存储器和客户端的安全认证和授权。

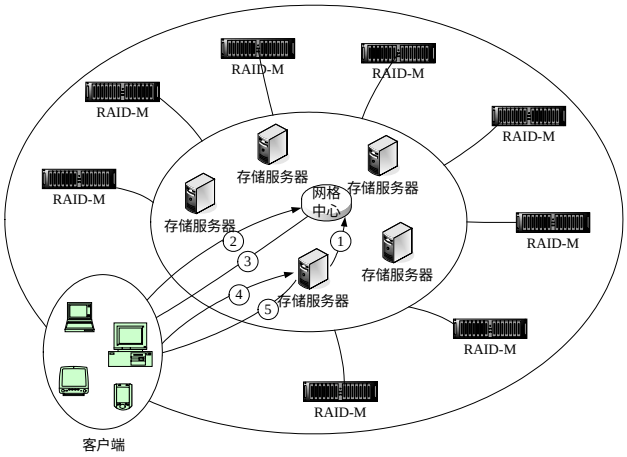

图 3　海量高可靠性存储系统整体架构

海量高可靠性存储系统的主要工作步骤如下：

(1)所有 RAID-M 存储器和存储服务器都向网格目录服务器发布自己的服务；

(2)存储访问客户端向网格调度服务器提出访问请求；

(3)由调度服务器根据负载均衡和就近服务策略选择一台存储服务器为其服务；

(4)客户端连接指定的存储服务器，提交或检索数据。

(5)存储服务器将客户请求转化为对 RAID-M 的访问请求，并将访问结果返回给客户端。

存储服务器在进行数据存储时，主要分为以下几步：

(1)从网格中心查找负载最轻、足够分散的(n+m)台 RAID-M 存储器；

(2)运用 CRS-RAID 对存储数据进行编码，并将编码后的数据和 SCSI 命令封装到 TCP/IP 包中通过网络转发给选定的 RAID-M 存储器；

(3)RAID-M 存储器收到 TCP/IP 包之后，将其还原为 SCSI 命令和数据，执行 SCSI 命令完成数据的存储，通知存储服务器数据存储已完成。

当客户端需要读取数据时，存储服务器首先发送通知到 n 台存储原始数据的 RAID-M 存储器，如果 n 台存储器正常工作，则返回数据到存储服务器，存储服务器把数据打包发送回客户端；如果 n 台存储原始数据的 RAID-M 存储器中某 k 台($1\le k\le m$)损坏无法返回数据，则在 m 台存储校验数据的 RAID-M 存储器中选择 k 台读取校验数据，CRS-RAID 对读取的所有数据进行解码后得到全部原始数据返回客户端。

3　系统可靠性分析

从可靠性理论来看，CRS-RAID 属于马尔可夫串联型的可修复系统。为了分析系统的可靠性，作如下假定：

(1)每个磁盘的失效率为 λ，即 $\lambda=1/MTBF_{disk}$（$MTBF_{disk}$ 为单个磁盘正常工作时间，即磁盘在连续两次失效之间的工作时间）；

(2)每个磁盘的修复率为 μ，即 $\mu=1/MTTR_{disk}$（$MTTR_{disk}$ 为单个磁盘的平均修复时间，即磁盘驱动器从失效到恢复正常工作的时间）；

(3)磁盘的崩溃相互独立，失效事件和修复事件时间上彼此相互独立。

k 个盘失效且系统可恢复时 CRS-RAID 的可靠性及其数学模型的分析如图 4 所示。

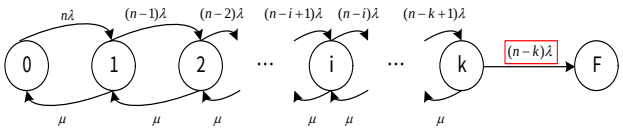

图 4　CRS-RAID 的马尔可夫可靠性模型

在图 4 中，状态“ 0 ”表示组内每个盘都正常工作，状态“ 1 ”表示有一个盘失效，状态“ 2 ”表示有两个盘失效，以此类推，状态“ k ”表示有 k 个盘失效。由于采用了冗余容错技术，因此直至状态“ k ”失效时，系统仍可以正常工作。状态“ k+1 ”为系统无法正常工作状态。由马尔可夫模型，可以得到如下方程组：

$$\begin{cases} P_0(t) = -n\lambda P_0(t) + \mu P_1(t) \\ P_1(t) = n\lambda P_0(t) - [\mu + (n-1)\lambda] P_1(t) + \mu P_2(t) \\ P_2(t) = (n-1)\lambda P_1(t) - [\mu + (n-2)\lambda] P_2(t) + \mu P_3(t) \\ \cdots \\ P_i(t) = (n-i+1)\lambda P_{i-1}(t) - [\mu + (n-i)\lambda] P_i(t) + \mu P_{i+1}(t) \\ \cdots \\ P_k(t) = (n-k+1)\lambda P_{k-1}(t) - [\mu + (n-k)\lambda] P_k(t) \end{cases}$$

设初始条件为 $P_0(0)=1$，$P_1(1)=P_2(1)=\ldots=P_k(1)=0$；将上面的方程式进行拉氏变换最终得到系统的可靠度为

$$MTTF_{CRS-RAID} = \sum_{j=0}^{k} \frac{(-1)^{j-1}\{[\mu+(n-k)\lambda]\Phi_{k-1}^{(i)} - \mu(n-k+1)\lambda\Phi_{k-2}^{(i)}\}}{(-1)[\mu+(n-k)\lambda]\Psi_{k-1} - \mu(n-k+1)\lambda\Psi_{k-2}}$$

其中，$\Psi_0 = n\lambda$ ；$\Phi_1^{(i)} = \lambda^i \prod_{j=0}^{i-1}(n-j)\ (k >> i+1)$。

设系统有 N_g 个单组，则系统的 $MTTF_{CRS-RAID}$ 为

$$MTTF_{CRS-RAID} = \sum_{j=0}^{k} \frac{(-1)^{j-1}\{[\mu+(n-k)\lambda]\Phi_{k-1}^{(i)} - \mu(n-k+1)\lambda\Phi_{k-2}^{(i)}\}}{N_g\{(-1)[\mu+(n-k)\lambda]\Psi_{k-1} - \mu(n-k+1)\lambda\Psi_{k-2}\}}$$

下面举例来说明多冗余对系统的可靠性的提高：当 m=10，k=3 时，已知$MTBF_{disk}$=4000000h，N_g=1，$MTTR_{disk}$= 0.25h，计算得到：$MTTF_{CRS-RAID}$=1.0×10^{16}h。

4　性能测试

为了进行性能对比，本文对 RS 算法予以实现，称为 RS-RAID，分别将 RS-RAID 和 CRS-RAID 应用于存储系统中，在局域网环境下对其编码和解码性能进行了测试，测试结果如图 5、图 6 所示。

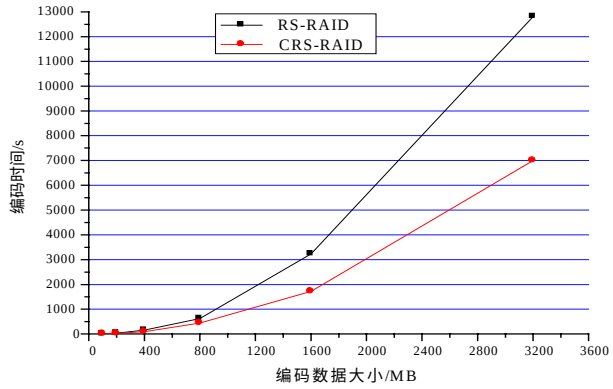

图 5　RS-RAID 和 CRS-RAID 编码性能测试

RS 算法在进行编码时采用范德蒙德矩阵，当进行除运算时，为了解决运算的“ 封闭性 ”，引入了 GF 域。每次进行乘除运算时都要进行两次 GF 域转换，例如，A 乘 B，首先通过函数 M 对 A 和 B 进行转换得到 A1 和 B1，然后对 A1 和 B1 进行异或操作得到 C，C=A1⊕B1，通过函数 N 对 C 进行反转换得到 D，D 即为 A 乘 B 的值。每次乘除运算进行两次 GF 域转换大大增加了计算复杂性。

CRS 算法采用 Cauchy 矩阵替代范德蒙德矩阵，在进行乘除运算时不需要进行 GF 域的转换，而只使用异或操作，因此，无论在编码时间和解码时间上都要明显优于 RS 算法。特别在解码过程中，随着恢复数据大小的增加，RS 算法的解码时间迅速增加，远远超过 CRS 算法的解码时间。

测试结果表明，CRS 具有极高的编解码性能，极大地提高了存储系统的数据读取性能指标。

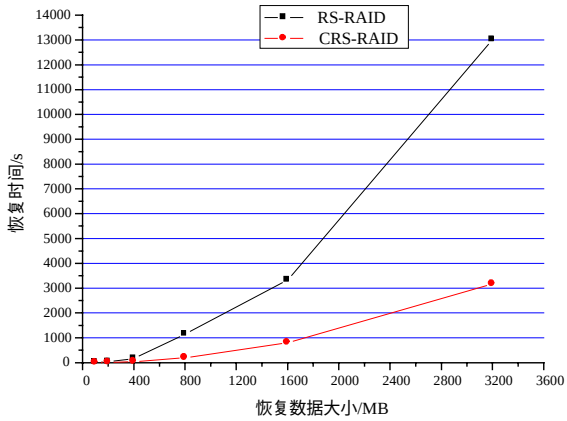

图 6　RS-RAID 和 CRS-RAID 解码性能测试

5　结束语

本文提出了海量高可靠性存储系统的整体架构，基于 CRS 算法开发实现的数据存储中间件最大限度地提高了数据的可靠性，即使在发生大规模破坏的情况下也可以很好地保证数据的完整性。同时，经过性能测试，整个系统具有极高的编解码性能，大大缩短了数据存取的时间。本文的研究对建设高可靠性的信息系统，具有一定的指导意义。

参考文献

1 那宝玉，张毓森. 存储系统中数据可靠性算法研究[J]. 解放军理工大学学报(自然科学版), 2007, 8(2)

2 Wicker S B, Bhargava V K. Reed-solomon Codes and Their Applications[M]. IEEE Press, 1994

3 Plank J S. A Tutorial on Reed-Solomon Coding for Fault-Tolerance in RAID-like Systems[J]. Software——Practice & Experience, 1997, 27(9): 95-101.

4 Plank J S, Ding Y. Note: Correction to the 1997 Tutorial on Reed-Solomon Coding[J]. Software——Practice & Experience, 2005, 35(2): 189-194.

5 Plank J S, Xu Lihao. Optimizing Cauchy Reed-Solomon Codes for Fault-Tolerant Network Storage Applications[C]//Proc. of the 5th IEEE International Symposium on Network Computing and Applications, Cambridge, MA. 2006.

6 Huang C, Xu L. STAR: An Efficient Coding Scheme for Correcting Triple Storage Node Failures[C]//Proc. of the 4th Usenix Conference on File and Storage Technologies. 2005.

7 Hafner J L. WEAVER Codes: Highly Fault Tolerant Erasure Codes for Storage Systems[C]//Proc. of the 4th Usenix Conference on File and Storage Technologies. 2005.

8 Hafner J L. HoVer Erasure Codes for Disk Arrays[R]. IBM Research Division, Research Report: RJ10352 (A0507-015), 2005.

9 都志辉，陈　渝，刘　鹏. 网格计算[M]. 北京：清华大学出版社, 2002: 3-8.

10 Foster I, Kesselman C. 网格计算[M]. 2 版. 金　海，袁平鹏，石　柯，译. 北京：电子工业出版社, 2004: 2-7.