一种高性能阵列架构研究

姜国松[1,3] 丁 红[2] 狄 平[3] 谢长生[1]
(华中科技大学计算机学院外存储系统国家重点实验室 武汉 430074)[1]
(上海第二工业大学计算机与信息学院计算机科学与技术系 上海 201209)[2]
(武汉邮电科学研究院 武汉 430074)[3]

摘 要 现今的应用程序需要更可靠的数据存储。到目前为止,数据存储的可靠性都是依靠不同的RAID级别来保证数据的可靠性,一般采用5个RAID级别[12]中的一种。存储方面的数据已经很明显地增长了,但是磁盘的可靠性并没有多大的改善。所以,为了控制存储的成本,有必要提供多元化的存储。在现有系统中加入新的RAID代码需要大量的开发、测试和调试工作,从成本上来讲是不现实的。因此,提出了一种新的通用RAID架构,此架构是基于异或的纠删码的,并将任意扇区和磁盘故障的组合作为基础,因而具有通用性。

关键词 冗余磁盘阵列,逻辑单元号,条带,条块,单元,扇区

Research about a New Architecture of High-performance Disk Array

JIANG Guo-song[1,3] DING Hong[2] DI Ping[3] XIE Chang-sheng[1]
(National Storage System Laboratory, Huazhong University of Science and Technology, Wuhan 430074, China)[1]
(School of Computer Science and Technology, Shanghai Second Polytechnic University, Shanghai 201209, China)[2]
(Wuhan Research Institute of Post & Telecommunication, Wuhan 430074, China)[3]

Abstract Current applications need more reliable data storage. Present day we use one of five commonly available RAID levels to protect against data loss due to media or disk failures. Data storage has been very significant growth, but the reliability of disk is not much improvement. Therefore, in order to control the storage cost, it is necessary to provide a wide range of storage. Adding a new RAID code in the existing system requires a large amount of development, testing and debugging work, such is unrealistic in terms of cost. Therefore, we proposed a new common RAID architecture, the architecture is based on XOR-based erasure (RAID) code, and makes arbitrary sector and the combination of disk failure as a basis, so it is universal.

Keywords RAID, LUN, Stripe, Strip, Element, Sector

1 引言

目前宽带网络建设中,热建冷用的现象非常严重。为了向客户提供增值网络服务而提出的流媒体服务项目,可以有效地缓解这一现象。伴随着视频运营市场的不断扩大,基于传统架构的服务器不能满足视频数据的特点,这主要表现在3个方面:其一,网络接口带宽的瓶颈因素;其二,存储带宽的瓶颈;其三,主机处理能力的不足。其中,尤以存储瓶颈的限制比较大。适合流媒体服务的高性能阵列控制卡就是为解决视频服务器在存储方面出现的瓶颈现象而设计开发的。

到目前为止,各种机构的客户数据非常重要。要防止客户数据由于媒体错误或设备故障导致的丢失,就意味着使用RAID来存储数据。为了满足更高的性能和可靠性要求,存储厂商已经采用了如RAID51的分级编码。这些编码虽然有改进,但是也带来了很大的风险,对于存储效率和性能并没有任何的好处。唯一可取的是它们可以重用基本的RAID编码,这样需要增加的源代码是最少的,这也意味着新产品的测试成本较低。

为什么在传统的RAID控制器上只支持少数的RAID编码?这有很好的理由:固件的复杂性随着支持的RAID编码而增加,并且需要越来越多的开发和测试成本。当固件成为大量RAID编码合集时,固件将变得很难做路径长度的优化。从软件维护的角度来说,if…then…else…代码块的堆积使得固件可读性更难,更容易发生错误。并且,每推出一个RAID编码可能需要进行现场升级。

由于改变固件的做法很难,普遍的心态是尽量避免之。但是,近来在存储技术方面的趋势和存储客户的关注点迫使重新评估现有的存储系统。

首先,没有单一的RAID编码能够满足所有方面的数据存储需要。而在有效的信息生命周期下,支持多种RAID编

到稿日期:2008-04-29 本文受国家自然科学基金项目(60603074)资助。
姜国松(1976—),男,博士生,研究方向为计算机存储系统,E-mail:gsjiang@fiberhome.com.cn;丁 红(1975—),女,硕士,讲师,研究方向为嵌入式系统开发;狄 平(1962—),男,工程师,研究方向为通讯系统管理;谢长生(1957—),男,教授,博士生导师,研究方向为NAS与SAN存储体系、iSCSI存储设备、图像存储与处理和计算机系统结构。

码是非常有价值的，数据的存放在性能、可靠性和效率水平上应当与其商业价值成正比。

第二，当数据量由 gigabytes 增长到 petabytes 时，其可靠性必须保持相对的恒定。然而，磁盘故障率随着磁盘容量的增加而增大[8,13]，对于各种规模的数据集，使用相同的 RAID 编码是不切实际的。

第三个原因是，模块化的系统日益普及，其分系统甚至使用 fail-in-place 组件简化管理。另一个趋势是使用低成本的串口 ATA(SATA)磁盘建设大型存储系统。相比 SCSI 磁盘，SATA 磁盘的硬件故障率是 SCSI 磁盘的 10 倍多，而成本要便宜 30%～50%[11,1,13,14]。使用较不可靠的磁盘提供高可靠性的数据，需要更多种类的 RAID 编码。

提供多种磁盘阵列码而不损害固件的质量、性能以及可维护性是矛盾的。根据这个矛盾，可以得出一个理想解决办法的需求：

(1) 它应考虑在加入新的 RAID 编码时不增加固件的复杂性；

(2) 它应该很容易支持各种流行的 RAID 编码，如基于异或的 RAID 编码能在硬件和/或软件中有效地实现；

(3) 应采用自动处理任何关于 RAID 编码相关的错误，例如在扇区故障或磁盘故障下的读错误；

(4) 由于在任何 RAID 的实现中错误处理占有很大比例，因此理想的情况下，解决方案应把 fault-free 和 fault-ridden 纳入共同代码路径；

(5) 应简化嵌套错误路径，例如在重构由于先前故障而丢失的块数据过程中，发现了一个新的扇区或磁盘故障。只有在 RAID 编码允许的条件下，这种重构才能完成，理想的解决方案必须弄清楚如何自动重构；

(6) 应通过动态的状态来自动调整 I/O 性能，例如缓存机制。

2 RAID 现状及相关研究

到目前为止，并不缺乏 RAID 编码，例如 EVENODD[2]，generalized EVENODD[3]，X-code[15]，RDP[6]，WEAVER[7]。最近有少量 non-XOR 编码已经实现了，但是相比简单的基于异或的编码，它们并没有提供特定的好处，这些 non-XOR[8] 编码仍然只在小范围使用。

过去的研究侧重于提供允许快速原型并评估冗余磁盘阵列，这样架构的固件环境典型的是 raidframe 。raidframe 将 RAID 架构的基本功能模块化——映射、编码和缓存。这种分解允许每一个环节可以根据新的设计独立地修改。阵列操作被描述为有向无环图(DAGS)，DAGS 图指定了不同原语之间的结构依赖关系。同时，它允许一个架构指定异常处理，使得 raidframe 缺乏自动性能调整的能力。

最近，RAID 的系统集成芯片(SOC)[10,5,9]产品已经可用了。aristos 的 SoC 充分体现了这个类别，它包含一个嵌入式处理器、DMA/XOR 引擎以及主机和磁盘接口逻辑。由于处理器是可编程的，可以想象，它们能支持多种磁盘阵列码。然而，问题是所有的错误路径必须指定为回调(很像 raidframe)，且这些回调必须由开发人员开发。此外，目前还不清楚如何自动调节性能。

3 存储系统方案

建立一个通用的 RAID 架构，通过 RAID 引擎和优化器来满足上述要求。此架构是基于异或的纠删码，并将任意扇区和磁盘故障的组合作为基础，因而具有通用性。

典型的运行中，通过 I/O 栈中块数据缓存来调用 EOM，使用相应的 RAID 编码来读、写、擦除、重构或迁移存储在磁盘上的数据。EOM 可以推断出重构和更新策略，并根据这些策略执行。此外，EOM 内的在线优化器可以根据当前缓存的内容为每一个读写操作选择最低开销的策略。可以将最小化系统级目标作为优化器的配置，例如最小的磁盘 I/O 或者最少的内存总线带宽目标。通过确定故障状态的参数，EOM 能够防止包括嵌套恢复在内不断出现的情况到一个单一的代码路径。最后，通过在已证明的成果上建立系统。

EOM 是一套程序集，当向 RAID 算法所编码的卷读写数据时，由 CACHE 管理调用 EOM 程序来完成读写。如图 1 所示，应用程序生成读写请求，将读写请求发送到 I/O 子系统，I/O 子系统通过数据 CACHE 来提供服务。在写回策略下，应用程序要写的数据首先被拷贝到数据 CACHE 中，并标记为脏页。在一段时间后，当替换策略决定把这些脏页替换出去的时候，脏页被写到磁盘上。在读数据的情况下，应用程序首先读数据 CACHE。当读缺失时，CACHE 模块首先从磁盘中取出数据，然后将数据返回给应用程序。大多数 CACHE 动态地区分写回和读，来处理各种应用负载。在大多数 RAID 控制器中，写回 CACHE 具有掉电保护。

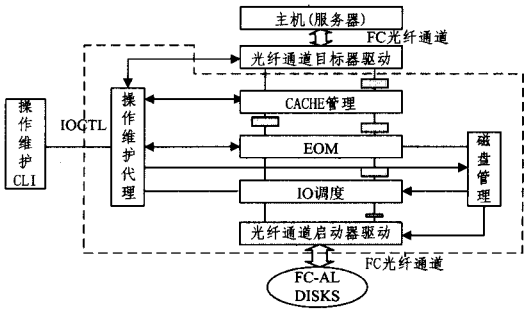

图 1 RAID 软件系统总体结构图

3.1 EOM 操作

在下列情况下调用 EOM 程序：

1)读缺失。数据请求的虚拟块地址被转换成实际的磁盘物理地址。在磁盘无故障情况下，磁盘读取数据并返回。如果磁盘有故障，此时必须启动 EOM 的重构程序。在重构不可能的情况下，返回读取错误。

2)清理脏页。当缓存替换策略选择将一个脏页写到磁盘时，将脏页的虚拟块地址转换成实际的物理磁盘地址。EOM 必须识别 RAID 的校验块，在脏页写入磁盘时，必须知道怎么更新。这同样适用于写穿策略。

3)重建丢失页。当数据由于磁盘故障而丢失时，由 RAID 系统内部的辅助程序调用重建进程。首先，EOM 利用 RAID 中的冗余信息重构丢失的数据页，然后将重构完成的数据页写到一个新的位置。这里重建可以看成是一组操作-重构的读操作和重构读完成后的写操作。

4)迁移数据页。数据迁移是由管理平台来触发的，调用迁移是为了改变数据页的 RAID 编码。迁移包括改变 RAID

磁盘数量以及 RAID 编码。类似重建，迁移可以看成是用旧的 RAID 编码读数据，然后用新的编码写数据。

3.2 EOM 的两个组成部分

如图 2 所示，EOM 程序从功能上可以划分为两个组件：RAID 引擎指出该做什么；执行引擎指出如何做。RAID 引擎将输入参数转换成一个 I/O 计划，此 I/O 计划由一组将被读取的块操作、一组将被异或的操作和一组将被写的块操作组成。这个 I/O 计划被输入到执行引擎，执行引擎执行必要的磁盘读/写和异或操作。

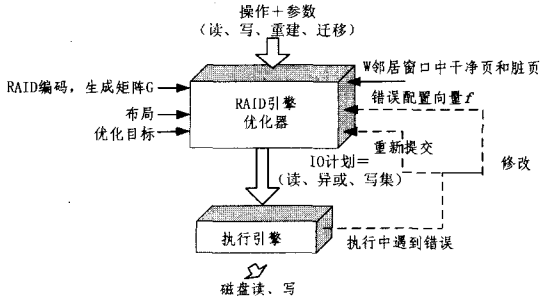

图 2　EOM 内部结构功能图

除了基本的操作型及其参数（页的块地址、起始虚拟块地址、读写字节数）外，RAID 引擎要求输入下列参数来产生 I/O 计划：

1）对 RAID 编码的描述，可从元数据中获得。

2）数据块的物理布局描述，在 RAID 编码中称之为 layout，也可以从元数据中获得。

3）对已知的扇区或磁盘故障的描述，称之为错误配置，可以从系统管理数据结构中获取。

4）在数据 CACHE 中将要被读/写的数据页周围的干净和脏页的列表，可以从数据 CACHE 目录中获得。

最后输入的 RAID 引擎是资源优化的目标。这可包括（但不限于）像最小化磁盘 I/O 或最小化存储总线带宽之类的标准。这个输入参数指导优化器对读/写操作选择一个策略。

由于在无故障的读操作情况下唯一合理的选择是直接从适当的磁盘读取需要的页，因此优化器几乎没有变化。但是对于写操作就有很多选择了。每个策略选择都要求不同数量的磁盘读和/或异或操作。在这种情况下，用配置目标函数来指导优化器进行选择。

在开始 I/O 计划之前，执行引擎获得了所有必要的数据页和分条锁。然后，执行引擎分 3 步执行 I/O 计划：首先，执行引擎提交磁盘读操作；然后执行引擎执行相应的异或操作；最后，执行引擎提交磁盘写操作。在任何一个执行步骤中，如果遇到了扇区或磁盘故障，执行引擎重新将整个操作提交给 RAID 引擎，同时提交最新的被发现的错误状态。在图 2 中展示了重新提交的步骤。这样设计的架构，其好处是恢复代码的路径不同于主代码的路径。

4　RAID 引擎

4.1　RAID 编码表示

在基于异或的擦除码中（RAID 编码），任何冗余的比特都是一定数量的数据比特的异或。为了更有效地表示，将这个联系应用于固定大小的 chunk，称之为单元。典型的一个单元由磁盘上一个或多个连续的页组成。每个页由多个扇区组成。一个单元要么是数据页，要么是校验页，但不可能既是数据页又是校验页。一个条带是数据元素集和所有相关的校验元素。在一个条带的一个校验单元是在这个条带内数据单元的某个子集的异或。校验单元依赖其所在的条带内的数据单元。组成一个条带的元素个数既依赖于磁盘个数（称为 rank），又依赖于编码方案。在每一个条带内，e 个连续的元素在每个存储设备上连续排列起来组成一个条块。为了简化，假定 RAID 编码有统一大小的单元和条块。

RAID 编码的矩阵表示是将数据元素与校验元素之间的异或关系表示成一个系统方程。这样组织成的矩阵称之为生成矩阵 G[4]，它是一个 $N\times M$ 矩阵，这里 N 是在一个条带内数据单元的个数，M 是在一个条带内所有单元（数据单元和校验单元）的个数。生成矩阵 G 的一列对应于一个条带内的数据或校验单元。生成矩阵 G 的一列对应一个条带内数据元素时通常只有唯一的“1”，生成矩阵 G 的一列对应在一个条带内的校验元素时有多个“1”，每一个“1”依赖于数据单元。

假如每个单元有 k 个页，用大小为 k 的单位矩阵替代每一个单元元素，那么生成矩阵 G 能够根据页而不是单元来重新生成。不失一般性，同时为了简单描述，假定每一个单元对应于单个页并且可交替使用词汇“单元”和“页”。

Disk1	Disk2	Disk3	Disk4	Disk5
D1	D3	D5	P1	P3
D2	D4	D6	P2	P4

图 3　5 磁盘 2 容错效验码中一个条带的物理排列

D1	D2	D3	D4	D5	D6	P1	P2	Q1	QQ2
1	0	0	0	0	0	1	0	1	0
0	1	0	0	0	0	0	1	0	1
0	0	1	0	0	0	1	0	0	1
0	0	0	1	0	0	0	1	1	1
0	0	0	0	1	0	1	0	1	1
0	0	0	0	0	1	0	1	1	0

图 4　5 磁盘 2 容错效验码生成矩阵 G 对应物理排列

4.2　布局表示

布局是在一个条带内数据和校验页的物理排列。除了像每页大小的配置参数外，布局的更多参数可以通过针对 RAID 编码和 e（每个条块的页数）参数的生成矩阵 G 来识别。生成矩阵 G 可以被看成是 e 列的块，每块对应于磁盘上每个条块的物理排列。当校验页分散在数据页当中时，布局是交叉存储的。

为方便起见，很值得使用列向量来归纳条带内校验页的位置。此列向量索引生成矩阵 G 中对应的校验页，称此向量为 $1\times(M-N)$ 维校验列向量。

考虑到对所有的磁盘的负载均衡，许多布局是周期性变换的。也就是说，基本字码的列是旋转分布，并均匀地将校验元素分布在所有磁盘上。此变换可以使用一个有符号的数字 s 来表示。s 定义了每个条带内条块的周期性变换。此符号表示变换的方向：负数代表左对称，正数代表右对称。

4.3　故障表示

$1\times n$ 维的故障配置向量为 f。假如页 i 发生故障，就将对应页 i 的元素标记成 1，否则标记为 0。在执行 I/O 计划

时，如果发生新的错误，故障配置向量 f 将被修改。这个流程如图 2 所示。

4.4 缓存表示

W-neighborhood 是所有干净页和脏页的集合，在数据 CACHE 中，它以脏页所在条带为中心，同时包含周边的条带，共 $2W+1$ 个条带的窗口。从图 5，可以很清楚地看到 W-neighborhood 定义。设置 $W>0$，EOM 能将多页的请求结合成一个大请求，然后再写回磁盘，因此极大地提高了磁盘的吞吐率。

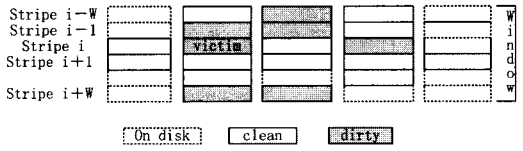

图 5 W-neighborhood

在 W-neighborhood 内的页可以分成一组干净页和一组脏页。每个页组可以用二进制向量编码，用 1 表示页在 CACHE 中。用字符表示这两个向量：干净数据向量 cv 和脏数据向量 dv。

对于写穿 CACHE，采用 0-neighborhood。

4.5 I/O 计划输出

如图 2 所示，I/O 计划是由 RAID 引擎基于前面所描述的参数输入所产生的。在形式上，一个 I/O 计划是一个三元组 (r,X,w)。I/O 计划是由 RAID 引擎基于前面所描述的参数输入所产生的。在形式上，一个 I/O 计划是一个三元组。r 是一个二进制向量，表示必要的磁盘读操作；1 表示 1 所在位置的页需要读取。同样地，w 是一个二进制向量，表示必要的磁盘写操作；X 表示异或操作，每一个 X 是将被异或的页的列表。X 能用一个 $M\times M$ 维的方形矩阵表示，这里列 i 是一组页，异或操作这组页将计算出校验页 i。

4.6 EOM 读

假如需要读取的页能够从磁盘中读取，那么需要在 r 中设置从磁盘中读取的页所对应的位置。在无故障的 EOM 读操作中，X 和 w 都设置成 0。

对于读操作，具有挑战性的情况是由于扇区或磁盘故障而导致重构进程的执行。为了得出重构策略，采用了 Hafner 等人描述的方案。为了表述方便，概述一下他们的重构技术。由生成矩阵 G 开始，然后修改生成矩阵 G 得到矩阵 $\overset{\vee}{G}$；对于每一个故障扇区，在 G 中对应列的元素为 0；对于每一个故障磁盘，磁盘上的页所对应的列为 0。形式上，$\overset{\vee}{G}$ 是如下计算得来的：

$$\overset{\vee}{G}=G(I_M-\mathrm{diag}(f))$$

其中 I_M 是大小为 M 的单位矩阵；$\mathrm{diag}(f)$ 是一个矩阵，它是通过将错误配置向量 f 应用 $M\times M$ 在矩阵的对角线得到的。

接下来使用高斯消元法，计算逆矩阵：

$$R=(\overset{\vee}{G}-1)$$

R 是 $M\times N$ 维矩阵，这里列 i 对应于继续存在页（数据页或校验页）的描述，而且这些页必须被读取或异或来重构数据页 i。

此方案的两个方面是 RAID 引擎结构的核心，这两个方面由 Hafner 等人讨论过，并且已经证明了。第一个方面是：只要 RAID 编码允许，矩阵逆技术只使用现存的页时，总能找到一个重构方案。第二个方面是 R 的不唯一性。根据线性代数，由于 $\overset{\vee}{G}$ 描述了一个非常详细的系统方程，那么它的反操作结果将不会唯一。每一个 $\overset{\vee}{G}$ 的逆操作定义一种读策略。给定一个资源优化目标，EOM 的在线优化器能够选择一个合适的策略并得到相应的逆矩阵 R。丢失的或者需要页对应的 R 的列被提取到 r 和 X。由于需求的页可能一部分已经在 CACHE 中，在逻辑上 r 应当与干净 CACHE 向量 cv 进行逻辑与运算得到执行引擎必须从磁盘读取的页组。注意：在重构读时，w 是 0。

4.7 EOM 写

4.7.1 确定受影响的校验页

将 W-neighborhood 内的所有脏页一并写回磁盘。脏页向量 dv 是包括脏页在内的将被写到磁盘的页的集。在任何 RAID 编码中，数据页的改变必须反映到它们所依赖的所有校验页。因此，首先是确定所有依赖的校验页。可以通过脏页向量与在 G 中的校验页的列的逻辑与运算来找出所有依赖的校验页。每一个非 0 的结果向量意味着继续存在的校验页必须更新，并作为写向量 dv 的一部分。为 0 的结果向量列意味着要么是这些校验页没有受影响，要么是这些向量由于扇区或磁盘故障不能写入磁盘。

4.7.2 写策略的选择

确定了受影响的校验页，下一步就是选择怎样更新每一个受影响的校验页。在 RAID5 中，任何校验元素可以两种方法更新：检验增量（PI）和校验计算（PC）。关于 PI（又可称作读-修改-写），RAID 控制器首先将被修改的数据和校验页从磁盘上读出来，然后计算被修改的新数据和从磁盘上读出来的未修改之前的旧数据之间的校验差，最后将得到的校验差作为旧的校验值的增量计算得到新的校验值。关于 PC，则是先从磁盘读取所有未修改的校验页所依赖的数据页，然后将这些数据页与脏页进行异或运算，从而计算出校验页。

问题是怎样将这些策略应用于任意错误配置下的任何 RAID 编码。为了解决这个问题，首先假定无故障配置，将用于 RAID 的方法扩展到应用于所有的 RAID 编码，然后再归纳到允许任意的故障。

写无故障的 RAID 编码卷在扩展后是：在 RAID 的条带中的任何脏页的正确更新是校验增量（PI）或校验计算（PC）的组合。为另一个校验重新使用一个校验更新的结果在某种形式上可以重新写，表现得就好像每一个校验页是单独更新的。对于在受影响校验向量的非零的元素，每一个 PI 或 PC 的实例定义了一个写策略。

这个概括允许系统的列举所有可能的写策略。对任何写，假如 p 个校验页受到了影响，那么将有 2^p 个写策略。每个写策略转换成不同的 I/O 计划，优化器可以选择一个最匹配资源优化目标的 I/O 计划。

为了处理扇区或磁盘故障，必须修正前面提到的扩展。修正后的方法允许在更新校验页之前用 PI 或 PC 重构所需要的页。

4.7.3 导出 I/O 计划

给定一个写策略，对于每一个受影响的校验页，可以独立

（下转第 37 页）

方案中，假设异常检测算法在每个节点上独立运行，低复杂度的协作算法可能会改善检测和遏制入侵的过程。一旦发现入侵，邻居节点将共同协作牵制入侵。目前实施方案还不完善，还存在一定的检测误差和误报警率，提高检测效率、降低误报警率将是下一步研究的具体目标。

参 考 文 献

[1] Chee Y, Rabaey J, Niknejad A. A class A/B low power amplifier for wireless sensor networks[C]// Proceedings of the 2004 International Symposium on Circuits and Systems. 2004, 4: 409-412

[2] Karlof C , Wagner D . Secure routing in wireless sensor networks: Attacks and countermeasures[C]// Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications. May 2003: 113-127

[3] Huang Y A , Lee W . A cooperative intrusion detection system for ad hoc networks[C]// Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks. ACM Press, 2003: 135-147

[4] Vigna G, Gwalani S, Srinivasan K, et al. An intrusion detection tool for AODV-based ad hoc wireless networks[C]// Proceedings of the 20th Annual Computer Security Applications Conference. 2004: 16-27

[5] 覃伯平，周贤伟，杨军. 无线传感器网络路由协议的攻击检测模型[J]. 计算机工程，2007，33(2)：117-119

[6] Xu Y, Heidemann J, Estrin D. Geography-informed Energy Conservation for Ad-hoc Routing[C]// Proc. of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking. 2001: 70-84

[7] Perrig A, Stankovic J, Wagner D. Security in Wireless Sensor Networks[J]. Communications of the ACM, 2005, 47(6)

[8] Lamport L, Shostak R, Pease M. The Byzantine Generals Problem[J]. ACM Transaction Programming Languages and Systems, 1982, 4(3): 382-401

[9] 裴庆祺，沈玉龙，马建峰. 无线传感器网络安全技术综述[J]. 通信学报，2007，28(8)：113-122

[10] Chan H , Perrig A , Song D . Random Key Predistribution Sch emes for Sensor Networks[C]// IEEE Symposium on Research in Security and Privacy. May 2003

[11] Zhang Y, Lee W, Huang Y A. Intrusion detection techniques for mobile wireless networks[J]. Wireless Networks, 2003, 9(5): 545-556

[12] Estrin D, Sayeed A, Srivastava M. Wireless sensor networks, Mobicom Tutorial. http://nesl. ee. ucla. edu/tutorials/mobicom02, 2002

[13] Zuniga M, Krishnamachari B. Analyzing the transitional region in low power wireless links[C]// Proceedings of the IEEE International Conference on Sensor and Ad hoc Communications and Networks (SECON). 2004

[14] 周贤伟，王培，覃伯平，等. 一种无线传感器网络异常检测技术研究[J]. 传感技术学报，2007，20(8)：1870-1874

[15] Onat I, Miri A. An intrusion detection system for wireless sensor networks[C]// Wireless And Mobile Computing, Networking And Communications, 2005. (WiMob'2005)/IEEE International Conference on Volume 3, Aug. 2005: 253-259

(上接第 29 页)

地计算出要被读、异或和写的页。EOM 通过结合针对受影响的校验页的字计划计算出 I/O 计划。这个可通过从逆矩阵中选择列，并将写策略转换成必要的读、异或和写操作。读操作集在留意干净页向量时被计算。

假如受影响的校验页 k 的子计划表示成 r_k，X_k，W_k，那么组合的 I/O 计划是通过总结所有的单个子计划得来的。

$$r=\bigvee_{k\in parity} r_k; X=\bigvee_{k\in parity} X_k; W=\bigvee_{k\in parity} w_k$$

结束语 已经显示了 EOM 对于在不增加固件复杂性的情况下提供多种 RAID 码的问题，是一个理想的解决方案。根据现有的知识，EOM 在能力上是独特的，同时具有灵活性（支持任何基于异或的 RAID 码）、简单性（统一的无故障和故障清除路径）和自我调节性。相对于现有的 RAID 实现，EOM 不仅具有竞争力，而且对于很多负载提供适当的性能提升。

对于将来可能的工作，是使 EOM 在适应数据布局上平衡一下性能、可靠性和工作效率。

参 考 文 献

[1] Archivas. http://www. archivas. com/

[2] Blaum M , Brady J , Bruck J , et al. EVENODD : An optimal scheme for tolerating double disk failure in RAID architectures. IEEE Transactions on Computers, 1995, 44(2): 192-202

[3] Blaum M, Bruck J, Vardy A. MDS array codes with independent parity symbols. IEEE Transactions on Information Theory, 1996, 42(2): 529-542

[4] Deenadhayalan V, Hafner J L, Rao K K, et al. Matrix methods for lost data reconstruction in erasure codes. Technical Report RJ 10354. San Jose, CA: IBM Research, 2005

[5] Aarohi Communications. http://www. aarohi. net

[6] Corbett P, et al. Row-diagonal parity for double disk failure// Proceedings of the Third USENIX Conference on File and Storage Technologies. 2004: 1-14

[7] Hafner J . WEAVER codes : Highly Fault Tolerant Erasure Codes for Storage Systems// Proceedings of the Fourth USENIX Conference on File and Storage Technologies. San Francisco, CA USA, December 2005: 211-224

[8] Intel Digital Enterprise Group Storage Components Division . SCD roadmap update. Powerpoint slide deck, March 2005

[9] iVivity. http://www. ivivity. com/

[10] Aristos Logic. http://www. aristoslogic. com/

[11] LeftHand Networks. http://www. lefthandnetworks. com/

[12] Patterson D, Gibson G, Katz R. A case for redundant arrays of inexpensive disks (RAID)// Proceedings ACM SIGMOD. ACM, June 1988: 109-116

[13] Isilon Systems. Uncompromising reliability through clustered storage. http://www. isilon. com/media/pdf/Isilon _ Uncompromising_Reliability. pdf, May 2005

[14] Terrascale Technologies. http://www. terrascale. com/

[15] Xu L, Bruck J. X-code: MDS array codes with optimal encoding. IEEE Transactions on Information Theory, 1999, 45: 272-276