[19]中华人民共和国国家知识产权局

[51]Int. Cl[7]
G06F 11/08

[12]发明专利申请公开说明书

[21]申请号 98811922.6

[43]公开日 2001年1月24日

[11]公开号 CN 1281560A

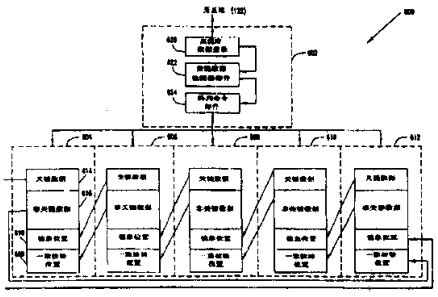
[22]申请日 1998.10.7 [21]申请号 98811922.6
[30]优先权
[32]1997.10.8 [33]US [31]60/062,663
[86]国际申请 PCT/US98/21080 1998.10.7
[87]国际公布 WO99/18507 英 1999.4.15
[85]进入国家阶段日期 2000.6.8
[71]申请人 西加特技术有限责任公司
地址 美国加利福尼亚州
[72]发明人 D·B·安德森
[74]专利代理机构 上海专利商标事务所
代理人 徐 泰

权利要求书2页 说明书37页 附图页数26页

[54]发明名称 混合数据存储和重建系统以及用于数据存储装置的方法

[57]摘要

一种用于存储装置的混合数据重建系统(600)和方法(630/660)。按照两种或多种冗余性方案(618/620)之一,有选择地存储数据(6 14/616),从而按照具有较高冗余度的方案(618)存储关键数据(614)。



ISSN1008-4274

00·06·12

权　利　要　求　书

1. 一种磁盘驱动器阵列，其特征在于，包括：

多个磁盘驱动器；

在操作上耦合至多个磁盘驱动器的至少一个控制器，构造控制器以接收数据和按照第一冗余性方案在磁盘驱动器上存储数据的第一部分，并且按照第二冗余性方案在磁盘驱动器上存储数据的第二部分。

2. 如权利要求 1 所述的磁盘驱动器阵列，其特征在于，第一冗余性方案比第二冗余性方案提供更大的冗余度。

3. 如权利要求 2 所述的磁盘驱动器阵列，其特征在于，数据的第一部分包括冗余性数据，它不同于数据的第一部分，并且构造控制器以按照第一冗余性方案在磁盘驱动器上存储数据的第一部分和冗余性数据。

4. 如权利要求 3 所述的磁盘驱动器阵列，其特征在于，构造控制器以在多个磁盘驱动器的第一组磁盘驱动器上存储数据的第二部分，并且在多个磁盘驱动器的第二组磁盘驱动器上镜象存储数据的第一部分。

5. 如权利要求 4 所述的磁盘驱动器阵列，其特征在于，构造控制器以在多个磁盘驱动器的第一磁盘驱动器上存储数据的第二部分，并且在多个磁盘驱动器的第二磁盘驱动器上镜象存储数据的第一部分。

6. 如权利要求 2 所述的磁盘驱动器阵列，其特征在于，数据的第一部分包括存取频度比数据的第二部分的存取频度更高的数据。

7. 如权利要求 2 所述的磁盘驱动器阵列，其特征在于，构造控制器以在一种结构安排中把数据的第一和第二部分作为目标存储，并且数据的第一部分包括结构目标，它包含指出结构安排的信息。

8. 如权利要求 7 所述的磁盘驱动器阵列，其特征在于，构造控制器以存储分区中的目标，并且结构目标包括装置控制目标、装置相关联性目标、分区控制目标和分区目标列表之一。

9. 如权利要求 2 所述的磁盘驱动器阵列，其特征在于，把数据的第一和第二部分作为目标存储，每个目标包括属性，并且数据的第二部分包括属性。

10. 一种在磁盘驱动器中的磁盘上存储数据的方法，其特征在于，包括下述步骤：

(a)按照第一冗余性方案存储数据的第一部分；以及

(b)按照与第一冗余性方案不同的第二冗余性方案存储数据的第二部分。

11. 一种磁盘驱动器阵列，其特征在于，包括：

多个磁盘驱动器；

在操作上耦合至多个磁盘驱动器的至少一个控制器，构造控制器以接收数据和按照第一冗余性方案在磁盘驱动器上存储数据的第一部分，并且按照直接编码方案在磁盘驱动器上存储数据的第二部分。

00·06·12

说　明　书

混合数据存储和重建系统以及用于数据存储装置的方法

发明领域

本发明涉及数据存储装置。说得更具体一些，本发明涉及用于数据存储装置(诸如磁盘驱动器、磁带驱动器或光盘驱动器)的数据重建。

发明背景

在计算产业中，众所周知有两种常规的计算机模型。第一种是主机计算模型，而第二种是群集(clustered)计算模型。

在主机计算模型中，最终用户的传统的进展是购买一套起始的系统，并且当需要额外的处理能力时，就用较大的系统来替换起始的系统。

在此循环中的各个时刻，会发生外伤性的不连续。例如，如果用户扩充起始系统，当他购买第二个升级的主机系统时，他可能需要从一种操作系统转换到另一种操作系统，或者甚至从一个卖主专有的体系结构转换到另一个卖主专有的体系结构。这些改变对于购买升级的机构，在金钱和雇员工时方面都需承担巨大花费。因此，在许多场合避免这种转换。

此外，主机模型使得计算机装备的剩余的价值很低。于是，当用升级的系统替代起始系统时，系统替换往往导致投资资本基本上完全丧失。此外，较大的升级的系统的销售量要比较小的系统小。于是，每个典型地升级的新系统的计算成本要比先前的系统的计算成本高。

在群集计算模型中，一台主机计算机被一群较小的基于标准的服务器替代。与主机模型相比这可以提供许多优点。由于此群可以作为只是单个系统出发，因此进入此群集模型的门限较低。此外，一般这些较小的系统销售量很大，使得计算成本较低。还有，这些系统是基于标准的，即，它们不呈现对专有体系结构的依赖。这就提供了可以从多个来源得到设备的能力，由此使得用户在每次后续的购买时可以选择最佳的替换设备。

群集计算模型本身也呈现出另一些优点。通过只添加满足现有的和不久的将来的需要所需的额外资源量能够更精确地控制升级成本。此外，用户能从广泛的卖方中进行选择，而不涉及转移至或转换至一种新的体系结构。类

似地，采用正确的体系结构，可能从来不需要转换至另一种操作系统。

还有，群集计算模型也有缺点和问题。例如，群集计算模型在向群集系统以允许此群采取单个主机能进行的工作负荷的方式提供共享数据的能力方面遇到困难。例如，当前实施这样的群集模型极为困难，即，群中的每个服务器需要对相同的数据处理具体事项。某些应用的例子包括航空公司预约系统或财政机关全部的业务往来的清查。

群集计算模型的第二个缺点只是缺少在管理存储于主机环境中的存储和数据方面的广泛的经验。这些经验已经包括在管理软件中，但在基于标准的群集环境中还没有得到。

常规的磁盘驱动器还包括了这样的缺点，它与丢失操作系统信息有关。例如，常规的磁盘驱动器包含数以百万计的数据扇区。由于任何数目的不同原因，一个或多个扇区可能变得不可读出或有毛病。如果变得不可读出的扇区是操作系统用于特殊目的的一个扇区，则即使能够读出其整个其余部分，在磁盘驱动器中的整个磁盘空间也可能无法使用。例如，在个人计算机环境中，主引导记录、分区引导记录、文件属性表(FAT)或根目录可能变得不可读出或有毛病。这能够导致丢失磁盘驱动器的基本上所有的内容。常规操作系统没有能力来恢复丢失这些关键的文件系统管理数据的盘面中的所有可读的数据。这意味着用户的巨大损失，并且特别令人遗憾。因为丢失的数据是与操作系统相关的，而与存储在磁盘驱动器上的实际数据(它不能读出)很少或毫无关系。

至今，任何在这些情形下的恢复数据的服务一般是极为麻烦的。这些服务一般需要把磁盘驱动器从其操作环境中移开，并且将它送至从事于恢复这些数据的服务的公司或服务提供者。提供这项服务不保证成功，并且对于后续的保密性不加保护，为此得放弃为保密目的而对磁盘驱动器的看管。

本发明致力于这些问题和别的问题，并且提供胜过现有技术的其他优点。

发明概要

本发明得出一种用于数据存储装置的混合数据再现系统和方法。按照两种或多种冗余性方案之一，有选择地存储数据，从而按照具有较高冗余度的一种方案存储关键性的数据。

00.06.12

附图简述

图 1 是按照本发明的一个方面的与网络相连的存储系统的方框图。

图 2 描绘按照本发明的一个方面的目标模型。

图 3－1 是第一结构的方框图，其中，请求器访问在存储装置上的一个目标。

图 3－2 是第二结构的方框图，其中，请求器访问在存储装置上的一个目标。

图 4 是按照本发明的一个方面的磁盘驱动器的透视图。

图 5 是描绘请求器访问目标的功能方框图。

图 6 描绘按照本发明的一个方面划分的存储媒体的一部分。

图 7－1 和 7－2 示出流程图，它描述按照本发明的一个方面的请求器访问目标。

图 8 是流程图，描绘按照本发明的一个方面创建目标。

图 9 是流程图，描绘按照本发明的一个方面打开和更新目标。

图 10 是流程图，描述按照本发明的一个方面写至目标。

图 11 是流程图，描绘按照本发明的一个方面为了只读而打开目标。

图 12 是流程图，描述按照本发明的一个方面读出目标。

图 13 是流程图，描述按照本发明的一个方面关闭目标。

图 14 是流程图，描绘按照本发明的一个方面移去目标。

图 15 是流程图，描绘按照本发明的一个方面创建一个分区。

图 16 是流程图，描绘按照本发明的一个方面移去除一个分区。

图 17 是流程图，描绘按照本发明的一个方面输出目标。

图 18 是流程图，描绘按照本发明的一个方面得到目标属性。

图 19 是流程图，描绘按照本发明的一个方面设置或修改目标属性。

图 20 是流程图，描绘按照本发明的一个方面读取加锁属性。

图 21 是流程图，描绘按照本发明的一个方面设置加锁属性。

图 22 是流程图，描绘按照本发明的一个方面复位目标的加锁属性。

图 23 是流程图，描绘按照本发明的一个方面得到装置关联。

图 24 是流程图，描绘按照本发明的一个方面设置装置关联。

图 25 是方框图，描绘按照本发明的一个方面实施的磁盘驱动器阵列。

图 26 是方框图，描绘按照本发明的一个方面的目标磁盘驱动器。

图 27 是方框图，描绘按照本发明的一个方面的一致校验(parity)磁盘驱动器。

图 28 是流程图，描绘按照本发明的一个方面创建一致校验组。

图 29 是流程图，描绘按照本发明的一个方面的写操作，其中，一致校验信息被更新。

图 30 描绘按照本发明的一个方面的数据结构。

图 31 是按照本发明的一个方面的使用嵌入位置信息的磁盘驱动器的方框图。

图 32 是流程图，描绘示于图 31 的系统的操作。

图 33 是方框图，描绘按照本发明的另一方面的使用嵌入位置信息的数据存储装置的另一实施例。

图 34 是方框图，描绘按照本发明的一个方面实现混合数据重建系统的磁盘驱动器阵列。

图 35 和 36 是流程图，分别对于面向块的数据和面向目标的数据进行混合数据重建方法的写操作。

较佳实施例的详细描述

图 1 是按照本发明的一个方面的数据存储系统 100 的方框图。系统 100 包括面向目标的数据存储装置 110 和 112；文件服务器 114；请求器 116、118 和 120；以及互连器 122。系统 100 表示一个与网络连接的存储结构，它能够用购自许多不同卖主的装备和软件构成，并且它作为一个单个的大型计算机系统呈现给用户。

面向目标的存储装置 110－112 是完成系统 100 的数据存储功能的存储部件。存储装置 110－112 最好包括磁盘驱动器、廉价磁盘冗余阵列(RAID)子系统、磁带驱动器、磁带库、光盘驱动器、投币式自动电唱机(juke box)或任何其他的能被共享的存储装置。存储装置 110 和 112 还设有至请求器 116、118 和 120 的输入/输出(I/O)通道连接件，这些请求器将访问装置 110 和 112。

请求器 116、118 和 120 是一些部件，诸如服务器或客户机，它们共享存储在装置 110 和 112 上的信息。请求器 116－120 最好也构造得直接访问在存储装置 110 和 112 上的信息。

文件服务器 114 完成管理和保安功能，诸如请求证实和资源定位。在较

小的系统中，最好不用专用文件服务器。作为替代，请求器 116－120 之一承担这项监视由文件服务器 114 执行的系统 100 的操作的功能和职责。此外，如果不需要或不想要由文件服务器 114 提供的保安和功能，或者性能的首要需要要求请求器组 116－120 直接与存储装置 110 和 112 对话，可把文件服务器 114 从系统 100 中去掉。

在一较佳实施例中，互连 122 是一种物理的基础结构，在连接网络的存储系统 100 中的所有部件经过该基础结构互相沟通。

在运作中，当系统 100 通电时，所有的装置最好或是互相或是对一公共参考点(诸如文件服务器 114 或互连 122)识别它们自己。例如，在基于纤通道(fiber channel)的系统 100 中，面向目标的存储装置 110 和 112 以及请求器 116－120 登录在系统的构造(fabric)上。在这样的实施中，系统 110 的任何部件(它想确定操作结构)能使用构造服务来识别所有其他的部件。请求器 116－120 从文件服务器 114 得知存储装置 10 和 112 的存在(请求器 116－120 能对存储装置 110 和 112 进行访问)。类似地，存储装置 110 和 112 得知在系统 100 中定位其他装置所需的信息的位置和必须用来调用管理服务(诸如备份)的地址。类似地，在一个较佳实施例中，文件服务器 114 从构造服务从得知存储装置 110 和 112 的存在。

取决于特殊系统 100 的保安实施，请求器 116－126（或它们中的任何一个）可以被拒绝访问系统 100 的某些部件。根据每个请求器可用的存储装置 110 和 112 的组，于是该请求器能够识别它可用的文件、数据库、和自由空间。

与此同时，系统 100 中的每个部件最好对文件服务器 114 识别任何与其相关联的特定考虑。例如，任何存储装置级别的服务属性可以一次传送至文件服务器 114，于是系统 100 中的所有其他部件从文件服务器 114 得知它们的属性。例如，某个特定的请求器 116－120 在启动后可能希望知道额外存储装置的引入。例如，当请求器记录在文件服务器 114 上时，可以提供这一属性。于是每当新的存储装置添加至系统 100 时，文件服务器 114 就自动地通知该特定的请求器。于是文件服务器 114 一般也可以向请求器传送其他重要特性(诸如存储装置是否为 RAID5、镜象存储装置、等等)。

按照本发明的一个方面，存储在存储装置 110 和 112 上的信息最好存储中图 2 中更好画出的系统中。每个存储装置 110 和 112 最好是面向目标的装置，它以这样的模式运作，其中，把数据作为目标 124－126 加以组织和存取，

而不是作为扇区的有序序列。面向目标的装置 110 和 112 用目标文件系统来管理目标 124－126，目标文件系统对于特定装置上的每个分区说明性地包括目标的一个单层列表。目标文件系统也称为平坦文件系统。存储在每个装置 110 和 112 中的存储媒体上的目标 124－126 最好是安置在装置 110 或 112(它以面向目标装置模式运作)上的容量分配的最小可见单元。在这样一个存储装置上的目标包括与唯一的识别符相关联的一组有序的扇区。数据通过识别符以及在目标中的偏置而引用。当操作系统管理在这些目标结构中的文件和元数据(metadata)(而不是如在现有技术的体系结构中管理数据扇区)的同时，由存储装置 110 或 112 本身把目标定位和放置在存储媒体上。

目标 124－126 由接口 128 存取，其中，目标暴露出可由请求器 116－120 调用的多个方法，以访问和处理在目标 124－126 中的属性和数据。于是，如图 2 所示，从一个请求器 116－120 发出一个请求 130。在一较佳实施例中，请求器 116－120 是计算机系统或者是系统的群或网络中的一个单元，该单元为对包含目标 124－126 的存储装置采取行动而提交请求 130。于是，请求器 116－120 可以是客户机和服务器。在任何一种情形中，由请求器 116－126 之一发出的请求 130 调用在接口 128 中的方法之一，如后文将要详细描述的，这又造成对一个或多个目标 124－126 的处理。

图 3－1 和 3－2 是能够用来存取存储在存储装置 110－112 中的目标的两种不同结构的方框图。为简单起见，在图 3－1 和 3－2 中只画出单个请求器 116 和单个面向目标的存储装置 110。当请求器 116 希望打开目标(诸如目标 124－126)时，请求器 116 可以直接访问存储装置 110，或者它需要从文件服务器 114 请求允许和位置信息，以访问在存储装置 110 上的目标。文件服务器 110 控制对存储装置 110 的存取的程度主要随系统 100 的特定实现的安全性要求而变。

在图 3－1 示出的方框图中，假设系统 100 是安全的。即，没有要求对在请求器 116 和存储装置 110 之间传输的命令信息和数据进行保护。在这种实施中，仍然可以存在用于管理功能的文件服务器 114，但不需文件服务器 114 来监视请求器与存储装置 110 的交互作用。

在这样的实施中，请求器 116 处于直接在存储装置 110 上存取和创建目标的地位。于是请求器 116 能够打开、读出、写入和关闭目标，仿佛它们本来就附着于请求器 116。在后文的应用中会更详细描述这些操作。然而，只是

为了清楚起见，这里只提供简短的概述。为了读取存储装置 110 上的一个目标，请求器 116 最好首先读取这样的一个或多个目标，它们揭示在存储装置 110 上的逻辑卷名或分区，并且揭示如何开始搜索存储在其上的目标。然后请求器 116 打开并读取一个目标，它可以是一个根目录。根据这个目标，定位其它目标是直接的，并且是根据目录的内容的。请求器 116 重复这一过程直到定位出所需的数据。由目标识别符(目标 ID)和目标内的位移引用数据。

在示于图 3—2 的第二个实施中，需要保安。因此，把文件服务器 114 插入请求器 116 和存储装置 110 之间的输入/输出(I/O)链，达到需要的保护等级所需的程度。在一个较佳实施例中，请求器 116 必须首先向文件服务器 114 请求允许作一组 I/O 操作。于是文件服务器 114(它可能为了额外的安全曾经拒绝向请求器 116 提供存储位置信息)通过返回足够的信息信任来自请求器 116 的请求，以允许请求器 116 直接与存储装置 110 联系。因为当存储装置 110 在文件服务器 114 上登录时，存储装置 110 最好被通知有关安全性参数，存储装置 110 最好不允许 I/O 请求，除非该请求是适当构造的，并且包括编码数据(它包括了来自文件服务器 114 的有效允许)。

于是，过程以一种类似于对图 3—1 所描述的方式进行。然后，与每个命令相关联的有效载荷可以很不相同。例如，在需要保安的情形下(示于图 3—2)，在请求器 116 和存储装置 110 之间传送的命令和数据都可能是加密的。此外，最好必须将允许信息添加至从请求器 116 提供给存储装置 110 的命令参数中。

由于在一个实施例中存储装置 110 和 112 能够包括硬盘驱动器，因此有必要简短讨论一下磁盘驱动器。图 4 是硬盘驱动器的透视图，该硬盘驱动器能够作为存储装置 110。在磁盘驱动器 110 中，在一个壳体 136 内，多个磁盘 132 设置在主轴电动机组件 134 的轴颈部。每个磁盘 132 具有多条同心的圆形记录磁道，它们示意地用 138 标出。每条磁道 138 划分或多个分区(将对于图 6 详加描述)。通过访问一条磁道 138 中的一个特定的分区，可以将数据存储在磁盘 132 上，或从磁盘 132 上检索。最好地一个执行机构 (actuator)臂组件 140 可旋转地安装在壳体 136 的一个角隅。执行机构臂组件 140 装有多个磁头万向架组件 142，每个磁头万向组件装有一个带读/写磁头或换能器(未示出)的滑块，用于从磁盘 132 读出信息和在磁盘 132 上写入信息。

音圈电动机 144 适用于精确地前后旋转执行机构臂组件 140，从而在滑块

142 上的换能器沿一般地由箭头 146 指出的弧线在磁盘 132 的表面上移动。图 4 还以方框图的形式画出磁盘驱动器控制器 148，用它以已知方式来控制磁盘驱动器 110 的某些操作。然而，按照本发明，磁盘驱动器控制器也用来实现与存储在磁盘 132 上的目标 124－126 的接口 128。

图 5 是当把磁盘驱动器 110 装在示于图 1 的系统 100 中时，它的一部分的方框图。在图 5 中，磁盘驱动器控制器 148 包括实现接口 128 的控制部件 150。目标 124－126 存储在构成磁盘 132 的存储媒体上。请求部件 152 在请求器 116－120 上实现，并且构造来逻辑上表达调用接口 128 中的方法的请求。在调用了一个方法后，控制部件 150 执行某些任务，以用所需的方式处理被识别的目标。控制部件 150 返回一个事件，它能够包括与任何被识别的目标相关联的数据或属性。还可根据由请求器 116－120 调用的特定方法，返回该事件。

为了使面向目标的装置 110－112 提供带有面向块的装置的操作系统传递的相同的功能，在装置 110－112 上的存储空间必须达到类似的可处理程度。于是，在一个较佳实施例中，提供了在存储装置 110－112 上的目标 124－126(它们存储在其上)上层的组织层。在一较佳实施例中，面向目标的存储装置 110－112 将磁盘空间分派成一个或多个独占的区域，称为分区。将对于图 6 更详细地描述分区。在一个分区中，请求器 116－120 能够创建目标。在一个较佳实施例中，一个分区中的结构是一个简单的、平坦的组织。任何操作系统能把其自己的结构映射在这个组织上。

图 6 画出在存储媒体(诸如磁盘 132 之一)上的存储空间的一部分。存储空间包括多个目标，诸如装置控制目标(DCO)154、装置相关目标联性(DAO)156 和多个分区，这些分区标以分区 0(也用标号 158 指出)、分区 1(也用标号 160 指出)和分区 N(也用标号 162 指出)。每个分区还包括多个目标，诸如分区控制目标(PCO)164。分区目标列表(POL)166 和多个数据目标 168(标为数据目标 0－数据目标 N)。

与每个目标相关联的是一组属性。按照本发明的一个方面，提供了一个存取控制属性，它由设置属性(Set Attribute)方法设置(将在后文详加讨论)，并且提供了使访问特定目标受控的方法。通过改变存取控制属性的版本号(version number)，拒绝或给予某些请求器 116－120 对特定目标的访问。

群集(clustering)目标是这样属性，它指出该特定目标是否应该合乎要求地放置得靠近存储系统中的另一个目标。复制(cloning)属性指出，特定目

标是否通过复制存储系统中的另一个目标而创建。一组规模(size)属性确定特定目标的规模特性。例如，规模属性的组包括这样的信息，它指出在目标内写的最大偏移、分配给目标的块数、用于在目标内存储数据的块数和在目标内每块的字节数。

一组时间属性指出何出创建目标、修改目标中的数据的最后时间以及修改在目标中的属性的最后时间。目标最好还包括这样一组属性，它们确定修改文件系统中任何数据的最后时间和修改文件系统中任何属性的最后时间。为了指出任何给定目标的其他参数、特性或特征，还可以提供其他的属性。

每个目标还与一个目标识别符相关联，该目标识别符由特定的存储装置 110－112 选择，并且响应于创建目标的命令而返回至请求器 116－120。识别符最好是一个规定长度的不带符号的整数。在一个较佳实施例中，识别符的长度默认为由特定的存储装置 110－112 规定的大小，或者能设置为装置属性。此外，在一个较佳实施例中，为公知的目标、特殊使用和其他希望进行的特定功能保留预定的识别符子组。

图 6 指出，存储媒体一般包括时常具有特定的目标识别符的公知的目标。在某些情形中，这些公知的目标存在于每个装置之上或在每个分区之中。

例如，一个公知的目标是装置控制目标 154，它最好包含由每个装置 110－112 保留的属性，并且它与装置本身有关，或者与装置上的所有目标有关。由“设置属性”(Set_Attribute)方法(该方法在后文描述)保留属性。在一个较佳实施例中，每个装置 110－112 有一个装置控制目标 154。

表 1 描述一组值得推荐的装置控制目标(DCO)属性。

表 1

类型	名称	字节	语 义
安全性	时钟	8	单调计数器
	主密钥	8	主密钥，控制装置密钥
	装置密钥	8	装置密钥，控制分区密钥
	保护等级	1	确定保护选项
分区	分区计数	1	装置上的分区数
装置属性	目标属性	8	确定与装置上所有的目标相关联的性质

在一个较佳实施例中，DCO 属性包括：时钟，它只是一个单调计数器；包括加密密钥的主密钥或者控制装置上所有其他密钥的其他主密钥；以及装置密钥，它控制分区密钥，并且可以用来锁定分区。属性还包括：保护等级密钥，它识别预定的保护等级并且具有相关联的保安方针；分区计数，它确定装置上的分区数；以及目标属性，它确定与正在访问的特定装置上的所有目标相关联的性质。

为了合适地管理跨越多个存储装置 110－112 的目标，每个存储装置 110－112 最好也包括一个装置相关联目标 156，它规定各个装置 110－112 之间的相关联性。例如，如果存储装置 110 和 112 是装置的镜象对，或者是一个阵列组的部件，则装置相关联目标 156 识别这种关系。表 2 示出装置相关联目标 156 的一些值得推荐的属性。

表 2

名称	字 节	语 义
相关联性识别符	2	此组的唯一的 ID
相关联性类型	2	相关联性类型
成员表	n	
相关联性识别符	2	
相关联性类型	2	
成员表	n	

这些属性最好包括相关联性识别符，它是相关联的装置的每个给定组的唯一的识别符。属性最好还包括相关联性类型，它确定装置之间的相关联性的类型(例如，镜象对、RAID5、等等)。属性最好还包括成员表，它只是识别装置 110－112，它们是上述相关联性的成员。

在存储装置 110－112 上的每个分区 158、160 和 162 最好还包括分区控制目标 164，它包含了单个分区的性质。目标 164 最好不仅描述了该分区，而且描述属于该分区中的所有目标的任何目标属性。每个装置 110－112 最好包括用于在装置上确定的每个分区的一个分区控制目标 164。虽然图 6 描绘了存储在每个分区中的分区控制目标，但情形不必总是如此。分区控制目标可以不存储在分区中，而存储在其上方的平坦文件系统中。

表 3 指出最好包括在分区控制目标 168 中的多个属性。

表 3

类型	名称	字节	语 义
	主密钥	8	加密密钥
	当前工作密钥	8	
	先前工作密钥	8	
分区属性	目标属性	8	确定与分区中所有的目标相关联的性质

这些属性最好包括主密钥，它确定用于整个分区的加密密钥，并且能够用来设置当前工作密钥。属性最好还包括当前工作密钥和先前工作密钥，最好用它们来对命令和数据消息加密和解密。分区控制目标 164 最好还包括目标属性，它与指定分区中的所有的目标相关联。

图 6 还画出，每个分区最好包括分区目标列表 166，它是当在存储媒体上创建一个分区时，由控制部件 150 建造的一个目标。分区目标列表 166 最好在每个分区中具有相同的识别符，并且组成引导在存储媒体上实施的目标文件系统的出发点。表 4 示出最好与每个分区目标列表相关联的属性列表。

表 4

字 段	字 节	
目标 ID	8	用于任何打开、读出、写入、关闭此目标的 ID
用户数据	N	POL 属性设置此数据，用 GET ATTRIBUTEC(得到属性)来得知其值

如表 4 所示，对于驻留在该分区中的所有目标，目标最好包括目标识别符(目标 ID)列表，以及分派给每个目标的用户空间的卷大小。由请求器使用目标识别符，以打开、读出、写入和关闭目标。此外，通过在分区目标列表中设置用户数据属性，用户最好能够对每个目标 ID 分派用户空间。在分区目标列表 166 之后，每个分区最好包括多个数据目标 168。根据数据存储系统的特定实施，每个数据目标 168 最好包括一个或多个在表 1 中设置的属性，并且能够包括附加属性。

面向目标的存储装置 110－112 最好支持请求，以向请求器 116－120 提

供数据或者为其存储数据。此外，在现有技术系统构造中，存储装置110－112最好承担对可能已在其他部件部件处(最可能在操作系统中)实行的其他功能的责任。最好由装置110－112自己来完成空间管理以及保持与在装置110－112上的目标相关联的属性。最好通过调用由接口 128 支持的方法来完成这些功能，接口 128 是由在每个存储装置 110－112 中的控制部件 150 实现的。在说明书的后文中会详述能调用的多种方法。然而，为了便于更好地理解这些方法，图 7－1 和 7－2 提供了流程图，它画出了按照本发明的一个方面对面向目标的文件系统的引导，可以相信，在详述后面给出的方法之前讨论图 7－1 和 7－2，将有助于理解本发明。

从方框 170 延伸至 204 的图 7－1 和 7－2 示出在存储装置 110－112 之一的特定分区中寻找目标。首先，请求器 116 在装置控制目标 154 中得到装置属性。这由方框 172 指出。调用 Get_DCO_Attributes(得到装置控制目标属性)方法使得控制部件 150 返回存储在装置控制目标 154 中的属性。这由方框 174 指出。然后请求器 116 根据从装置控制目标 154 返回的属性选出给定的分区。这由方框 176 指出。

如方框 173 指出的那样，一旦请求器 116 选出了分区，请求器 116 就调用 Get_DAO_Attributes(得到装置相关性目标属性)方法。这使得控制部件 150 从存储在存储媒体 110 上的装置相关联媒体 156 得到属性。然后控制部件 150 把装置相关联属性返回至请求器 116，如方框 175 指出的那样。根据装置相关联属性和装置控制属性，请求器 116 选出一个分区来询问。这由方框 176 指出。

然后请求器 116 调用 Get_PCO_Attributes(得到分区控制目标属性)方法，这将使得控制部件 158 得到在分区控制目标 164 中找到的属性，分区控制目标 164 与要由请求器 116 询问的特定的分区相关联。这使得控制部件 150 得到并且返回分区控制目标属性。这由方框 178 和 180 指出。如果在选出的分区中的目标不是请求器所关心的目标，则请求器选择另一个分区，如方框 182 和 176 所指出的那样。

然而，假设请求器已经找到了所关的分区，于是请求器对于选出的分区调用 Get_PCO_Attributes(得到分区目标列表属性)方法，如方框 184 所示。此方法使得控制部件 150 从与选出的分区相关联的分区目标列表 166 得到属性。然后把这些属性提供给请求器 116，如方框 186 指出的那样。

接着，请求器调用 Open_Read_Only_POL(打开只读分区目标列表)方法。这由方框 188 指出。如后文要详细讨论的，控制部件 150 得到存储在分区目标列表 166 中的与选出的分区相关联的数据，但修改在该目标中的一个属性，以指出正在只读的基础上提供数据，从而数据不能修改或扩展。这由方框 190 指出。

然后请求器调用 Read_POL(读出分区目标列表)方法，它使得控制部件 150 提供选出分区中的目标的列表，供请求器 116 审查。这由方框 194 指出。在选出的分区中选出所要的目标后，请求器 116 调用 Close_POL(关闭分区目标列表)方法，它使得控制部件 150 关闭分区目标列表。这由方框 196 指出。

在对于所要的一个或一些目标发现了目标 ID 之后，于是请求器 116 调用 Open_xxx_Objectx(打开 xxx 目标 x)方法。xxx 指出根据请求器所要的特殊的数据处理。而由请求器调用的一种特定的打开方法。Objectx 指出来自分区目标列表的目标 ID，该分区目标列表识别要由请求器处理或访问的目标。例如，xxx 指示可以代表 Open_Update(打开更新)操作，或 Open_Read_Only(打开只读)操作。这些将在后文讨论，而这一步骤由方框 198 指出。

然后请求器对由控制部件 150 返回的目标作所要的处理。后文将讨论能用来处理目标的各种方法。这由方框 200 指出。

最后，一旦请求器完成所要的目标处理或访问，请求器 116 就调用 Close_Objectx(关闭目标 x)方法，它也将在后文详述，而运作该方法以关闭由请求器 116 访问的目标。

图 8—24 是流程图，画出能由请求器调用的各种例示的方法，以完成对存储在面向目标的存储装置(诸如装置 110)上的目标所要的功能和所要的处理。

图 8 是流程图，具体画出 Open_Create_Object(打开创建目标)方法。当请求器 116 调用此方法时，如方框 208 指出的，控制部件 150 创建一个新的目标 ID 并且将这个目标 ID 输入与特定的分区(目标将在其中创建)相关联的分区目标列表。这由方框 210 指出。然后通过分派与目标相关联的块数等等，以及通过修改目标属性以指出目标创建时间和设置在表 1 中列出并且与目标相关联的其他属性，控制部件 150 创建一个新的目标。这由方框 212 指出。接下来，控制部件 150 返回请求状态以及刚创建的目标的新的 ID。这由方框 214 指出。

除了只是创建一个目标之外，请求器 116 还能够规定许多可选项。例如，在一个较佳实施例中，请求器 116 能够规定目标是否由口令保护、目标是否加密、某些质量服务门限(例如，目标是否备份)、加锁特性(例如，目标是否要由目标锁以及任何其他的锁(诸如分区锁和装置锁)锁定)、存取控制版本、镜象或其他备份支撑(它将使得所有的更新要在另一个目标上生成镜象，或用规定的其他方法备份)，以指出将以规定的最小规模为单位分派空间以及设置冲突(collision)特性(诸如在 UNIX 型系统中的写入)。

请求器 116 为调用此方法而向控制部件 150 提供的特殊的信息包括系统中的允许信息(这些系统为保安而需要此信息)、要在其中创建的装置的分区、以及上面提代的任何可选项。在一个例示的实施例中，作为响应，控制部件 150 返回在装置上可用的容量、请求的状态、以及新的目标的 ID。

还应该注意，能够调用这个方法的一个特例，它包括与一个目标相关联的所有数据。在此情形中，能够调用这样一种方法，它能创建目标、写入目标和关闭目标。

图 9 是流程图，示出 Open_Update_Objectx(打开更新目标 x)方法。当请求器 116 调用此方法时，如由方框 220 指出的，这允许请求器 116 读出和写入规定的目标。它还提供目标的长度的扩展。当调用此方法时，控制部件 150 在规定的目标中设置一属性，以指出正在使用该目标。请求器 116 提供允许信息、包含该目标的分区 ID、要被访问的目标的识别符、要采取的行动的类型(诸如更新或写入)以及上面提到的任何可选项。作为响应，控制部件 150 返回请求状态和规定目标的长度，以及请求器 116 可用的剩余容量。

图 10 是描绘 Write_Objectx（写目标 x）方法的流程图。当请求器 116 调用此方法时，如方框 242 指出的，这使得控制部件 150 在规定的位置处在指定的目标中写入至规定数据的块。

写入方法也能调用其他方法。例如，如果对于要访问的装置 110－112 要求一致校验支持，则写入可自动地调用“异或”方法(它对要写入的数据进行“异或”操作)，而一致校验数据要被写至一个或多个预先规定的一致校验装置。

为了调用此方法，请求器 110 提供允许信息、目标识别符、分区 ID、在目标内要写入的块的起始位置、要写入至目标的块数、可选项信息、以及要写入的数据。一旦调用此方法，控制部件 150 就用提供的规定数据修改规定

的目标。这由方框 244 指出。然后控制部件 150 修改规定的目标中的必要属性(诸如目标长度、与目标相关联的时间戳记、等等)。这由方框 246 指出。然后如果有需要，控制部件 150 修改其他目标的必要属性(诸如分区目标列表)。这由方框 248 指出。然后控制部件 150 返回请求状态至规定的请求器。这由方框 250 指出。

图 11 是描绘 Open_Read_Only_Objectx(打开只读目标 x)方法的流程图。当调用此方法时，控制部件 150 允许请求器 116 为了只读的目的而访问规定目标。于是，当调用此方法时，如方框 230 指出的，请求器提供允许信息、分区 ID、目标 ID、和可选项信息。然后控制部件 150 在规定目标中设置指出该目标正在使用的属性。这由方框 232 指出。然后控制部件 150 在目标中设置指出目标不能被请求器写入的属性。这由方框 234 指出。然后控制部件 150 返回请求状态和规定目标的长度。这由方框 236 指出。

图 12 是描绘 Read_Objectx(读出目标 x)方法的流程图。当请求器 116 想要装置 110 返回来自规定目标的数据时，由请求器 116 调用此方法。请求器提供允许信息、目标 ID、分区 ID、要读取的块的起始位置、要读取的块数、以及任何其他所要的可选项信息。作为响应，控制部件 150 返回请求状态、正在返回的数据长度、以及响应于此方法正在返回的实际数据。这由方框 256 和 258 指出。

图 13 是描绘 Close_Objectx(关闭目标 x)方法的流程图。如方框 264 所指出的，当请求器 116 调用此方法时，请求器提供允许信息、目标 ID、以及任何所要的可选项信息。作为响应，控制部件 150 修改在规定目标中的数据，如方框 266 指出的那样。此外，如果作为写至目标的结果的目标的任何改变不曾写至存储媒体，则在此时刻写入。控制部件 150 还更新目标 x 的属性，如方框 268 所指出的那样。例如，如果目标是一个新创建的目标，则其属性用创建时间和其他所需的属性信息来更新。此外，修改属性以指出在目标中的数据被修改的最后时间，数据长度(如果它被改变)和属性由控制部件 150 设置，它指出给定的请求器不再使用该目标。

可选地，控制部件 150 也能更新与目标相关联的剩下的高速缓存信息，并且反映在目标属性中。这由方框 270 指出。例如，作出请求的规定的请求器 116 构造来通知存储装置 110，对于关闭的目标，数据仍然存储在高速缓存中，或者不再存储在高速缓存中，存储装置 110 的操作系统能够对那些应用

保持高速缓存信息，在这些应用中，目标将以迅速的顺序关闭和再次打开。然而，与此同时，存储装置110能够始终跟踪在接连冲突(Coherency collision)的事件中需要告知系统 100 中的哪个部件，是否在同时应由另一个请求的请求访问此目标。然后控制部件 150 返回请求状态，如方框 272 指出的那样。

图 14 是描绘 Remove_Objectx(移去目标 x)方法的流程图。如方框 278 所指出的，当调用此方法时，控制部件 150 采取必要的步骤从存储媒体中删除目标。这由方框 280 指出。然后控制部件 150 修改与分区(从该分区删除对象)相关联的分区对象列表，以反映出该指定的目标 ID 可供使用。这由方框 282 指出。然后控制部件 150 返回请求状态，如方框 284 所指出的那样。为了调用此方法，请求器 116 提供允许信息、分区 ID、目标 ID、以及任何所要的可选项信息。然后控制部件 150 返回请求状态，如方框 284 指出的那样。

图 15 是描绘 Create_Partitionx(创建部分 x)方法的流程图，该方法能被请求器调用，如方框 290 所指出的那样，以在存储装置 110 上创建分区。应该注意，虽然 Create_Partitionx 方法把驱动器划分为一个或多个区域，但在存储媒体上的所有空间不必计及。此外，划分区域也能在磁盘上跨越不同的区域。

在一个实施例中，用此方法以铺砌(tiling)结构创建分区，这些分区代表在装置的存储空间的真实划分。通过服务等级(诸如数据阵列)，使用这种结构来划分空间。这些分区不能调整大小，但能移去和重新创建。

按照本发明的另一方面，这些分区用作逻辑分区，以逻辑地组织目标而不是按照服务等级来管理空间。在此第二实施例中，可以动态地调整这些分区的大小。

为了调用此方法，请求器提供允许信息、任何所要的可选项、分区 ID、以及起始的空间分配(它识别要分配给被识别的规定分区的空间)。作为响应，控制部件 150 对于规定的分区分配在存储媒体上的空间，如方框 292 所指出的那样。然后控制部件 150 建立分区控制目标和分区目标列表，如方框 294 和 296 所指出的那样。如上所述，部分目标列表不能移去，并且用作引导分区的目标的起点。然后控制部件 150 返回请求状态和分区映射(它描述已经作出的划分方式)。这由方框 298 指出。

图 16 是描述 Remove_Partitionx(移去分区 x)方法的流程图，为了调用此方法，请求器 116 提供允许信息、可选项信息、以及识别要被移去的分区

的分区 ID。这由方框 304 指出。作为响应，控制部件 150 对先前与分区相关联的空间重新分配，如方框 306 指出的那样。然后控制部件 150 移去在分区目标列表(它与要删除的分区相关联)中的所有目标，删除分区目标列表并删除分区控制目标。这由方框 308、310 和 312 指出。然后控制部件 150 返回请求状态和示出对分区作出的改变的分区映射。这由方框 314 指出。

按照本发明的一个方面，把数据管理方针通知每个存储装置 110－112，从而存储装置能够相互独立地动作，以执行管理方针。这提供了明显的优点，即，它不仅导致较少的人为干预而且导致更可预测的和适时的管理控制。

例如，在存储装置 110－112 上的数据可能希望每周作备份。常规的系统一般在周末的空闲时间作此备份，从而在工作日期期间系统的可用性不受妨碍。然而，在系统容量增加的同时，可用性的窗口已逐渐收缩。于是打算找出一段足够长的时间来中断系统以备份可能是兆兆字节的数据已经变得非常困难。

于是，按照本发明的一个方面，通过根据分配给目标的属性而对其采取行动，一当一个目标达到对其备份的正确状态，面向目标的存储装置 110－112 就能通知备份功能。还有，备份所有的文件可以分散在一段较长的时间内(在此期间内其他的仍在更新)进行，而不影响数据的完整性。

可以由面向目标的存储装置 110－112 调用动作的其他的属性例子包括加密、压缩、改版本(versioning)和一致校验冗余度。在这些例子的每个例子中，存储装置 110－112 最好只需要告知关于规定目标或目标组的方针。于是，装置本身能够完成功能或通知指定的代理者提供服务。

例如，在存储装置 110－112 上本身能进行压缩和加密。因此，所需通知装置的唯一事情是需对一个目标进行压缩和加密。对于由代理者进行的管理功能，不仅必须把管理功能的方针通知存储装置，还要把对进行这项功能的代理者的标识通知存储装置，从而在进行此项功能时，存储装置能够访问该代理者。

按照本发明的一个方面，在目标之间建立相关联性，从而能够识别具有相同属性或具有依从关系的目标。例如，假设一个数据库包括 6 个文件或目标，或者是在所有的文件或目标都关闭或者是指定为其他所有的文件或目标都依赖于它的一个目标关闭之前，不能备份任何一个文件或目标。为了管理目标之间的此类关系，可能需要文件服务器 114。此外，本发明还建立装置之

间的依从关系，如在成阵列的一致校验组的情形中那样。通过可以建立一些组，其中，一个装置或目标确信的其余成员具有相同的基本性质，则组的管理效能更高和更有用。

图 17－24 是一些流程图，它们说明了通过调用由存储装置上的目标揭示的方法而能够完成的管理功能。调用这些方法使得控制部件 150 和/或有关的控制部件采取步骤，以进行与被调用的方法相关联的管理功能。

图 17 是说明 Export_Objectx(输出目标 x)方法的流程图。如方框 320 指出的，通过提供允许信息、可选项信息、目标 ID、对象装置 ID 和对象分区 ID，请求器 116 调用此方法。输出方法使得存储装置 110－112 能够根据在与一给定目标相关联的属性中表述的规则采取行动。例如，能够使用该方法来启动至其他装置的目标备份或支持改版本。

当调用 Export_Objectx 方法时，如方框 322 指出的，控制部件 150 从存储媒体得到规定的目标。然后控制部件 150 在由请求器 116 规定的对象装置处调用 Open_Create(打开创建)方法。这由方框 324 指出。然后控制部件 150 在对象装置处调用写入方法提供规定目标的数据和属性。这由方框 326 指出。然后控制部件 150 在对象装置处调用 Close(关闭)方法，在已写入至对象装置后关闭在对象装置上的目标。这由方框 328 指出。最后，控制部件返回请求状态连同目标的新的目标 ID(它已写入至对象装置)至请求器。这由方框 330 指出。

由控制部件 150 实现的接口 128 也支持允许请求器得到用于审查的目标属性和设置目标属性的方法。图 18 和 19 是分别说明相应的 Get_Objectx_Attributes(得到目标 x 属性)方法和 Set_Objectx_Attributes (设置目标 x 属性)方法的流程图。

如方框 336 指出的那样，一当调用图 18 中描述的方法，就使得控制部件 150 从规定的目标得到属性。在一个例示的实施例中，请求器提供允许信息、目标 ID(或目标 ID 的列表)、以及可选项信息。然后控制部件 150 得到与目标 ID(或目标 ID 列表)相关联的属性，并且将那些属性连同请求状态返回至请求器。这由方框 338 指出。

如方框 344 指出的那样，通过请求器能够调用在图 19 中描绘的 Set_Objectx_Attributes 方法，以提供允许信息、目标 ID、和可选项信息至控制部件 150。然后控制部件 150 用由请求器提供的信息修改规定目标的属性，

并且返回请求状态连同规定目标的属性(已修改)，这由方框 346 和 348 指出。

按照本发明的另一方面，目标能够被加锁使得一旦目标被拥有锁(它已放置在目标上)的服务器解锁，才能对目标访问，在一个较佳实施例中，能够将目标锁定在目标层次、分区层次或装置层次上。加锁机构提供了解决服务器间的访问方案。在一个较佳实施例中，使用这些锁以调度迸发的更新以及在维护功能期间禁止访问。图 20、21 和 22 是说明加锁方法的流程图，它们可视为 Get_Attribute 和 Set_Attribute 方法的例子。然而，对于那些方法的这些特例给出了额外的细节，从而在请求器群之间共享数据时可使用它们。

图 20 是描绘 Read_Lock_Attributes(读出锁属性)方法的流程图。如方框 354 指出的那样，通过提供来自请求器 116 的允许信息、目标 ID、分区 ID 或装置 ID、锁参数、以及任何所要的可选项信息至控制部件 150，能够调用此方法。响应时，控制部件 150 确定规定的目标是否具有设置的锁。控制部件 150 然后返回拥有锁的请求者的请求状态。这由方框 356 指出。

图 21 是描绘 Set_Lock_Attribute(设置锁属性)方法的流程图。如方框 362 指出的那样，通过提供允许信息、目标、分区或装置识别符信息、锁信息和可选项信息，请求器能够调用此方法。当调用此方法时，控制部件 150 检查与经识别的目标相关联的锁。这由方框 364 指出。然后控制器部件试图用请求器的标识来进行加锁或解锁操作。这由方框 366 指出。如果请求操作的请求器是锁的拥有者，则将完成操作，如果不是，则将不完成操作。在任何情形下，控制部件 150 返回请求状态连同拥有锁的服务器的 ID。这由方框 368 指出。

图 22 是描绘 Reset_lock_Attribute(复位锁属性)方法的流程图。在拥有锁的服务器不再行使职责的事件中若打算对锁复位，就使用该功能。如方框 374 所示，通过提供允许信息、目标、分区或装置识别符信息、锁参数、以及任何所要的可选项信息，能够调用此方法。在响应时，如方框 376 指出的那样，控制部件 150 对规定目标、分区或装置加锁，并且返回请求状态连同拥有锁的服务器的标识。这由方框 378 指出。

图 23 和 24 是描绘 Get_Device_Association(得到装置相关联性)方法和 Set_Device_Association(设置装置相关联性)方法的流程图。这些方法确定或询问装置 110－112 之间的关系。这样的关系的一种例示的实施包括把存储装置 110－112 之一识别为第一组装置的主装置，而其余的是该组的从属成员。

该组的第一或主装置担负着把组属性的改变传递给其它成员的任务。如果该组的第一或主装置没有提供属性设置，则其他成员可拒绝这些属性设置。存储装置 110－112 为了实现这些功能，它们具有进行自检的能力。这允许装置检查自己，以确定它们是否包括在一个较大的装置组中的成员关系中。

在图 23 中，描绘了 Get_Device_Associations 方法。如方框 384 指出的，通过提供允许信息和可选项信息，能够调用此方法。在响应时，控制部件 150 返回请求状态以及请求的相关联性(装置是其一员)，这由方框 386 指出。

图 24 是描绘 Set_Device_Assocations 方法的流程图。如方框 392 指出的那样，通过提供允许信息、可选项信息以及确定相关联性的成员和属性的列表，能够调用此方法。在响应时，控制部件 150 修改包含在存储媒体中的装置相关联性目标 156，如方框 394 指出的那样。修改装置相关联性目标，以包括由请求器提供的属性，并且包括时间戳记，以指出目标属性何时进行过最后一次修改，等等。如方框 396 指出的那样，控制部件 150 返回请求状态。

上述允许信息例示性地允许文件服务器 114 选通访问存储装置，其做法是控制某个请求器 116－120，文件服务器 114 把从存储装置 110－112 得到响应所需的凭据(credential)给了该请求器。文件服务器 114 还命令存储装置 110－112，它们必须只尊重 I/O 请求(它们依附于安装安全方针)。由 Set_Object_Attributes 方法把构成允许保安能力的基础的密钥传送至存储装置 110－112。如果对于存储装置 110－112 设置合适的安全等级。就可使该存储装置对于安全依从性检查每个 I/O 命令。然而，如上所述，某些应用不需使用保安性。此外，如果一个特殊的服务器群具有位于另一个物理设备中的一些装置，就可能希望对于与位于远处的装置通信，(但不是对于与本地通信)定义较高的保安等级。这允许对于位于远处的请求器或服务器使用保安性，但避免性能损失(这种性能损失将与对于本地请求器或服务器使用这种保安性相伴随)。

此外，每个存储装置 110－112 最好包括可读取的单调递增的时钟，它将用于时间戳记安全消息和目标。在一个例示的实施例中，在系统范围的基础上使得用于各种装置的时钟同步。在另一个例示的实施例中，文件服务器 114 适应于从存储装置到存储装置间的差异和不同的值。

于是，可以看出，本发明提供了面向目标的存储装置(诸如磁盘驱动器)，它具有胜过常规的存储装置的显著优点。面向目标的存储装置显著地改进了

群集体系结构。例如，通过以面向目标的方式存储数据，能由存储装置本身来管理数据。目标向存储装置提供有关其驻留数据的足够资料，从而能够担当它们管理其自身空间的职责。此外，当装置具有关于哪些构成了逻辑实体的信息时，就能更智能地控制数据共享。例如，如果两个系统共享存储在面向块的装置上的数据，则必须控制对于迸发访问的元数据活动。相反，在面向目标的装置中，大部分元数据活动对于访问它的系统来说，是不透明的。于是，系统只需关心对用户数据的存取冲突。此外，由装置自己进行空间管理消除了任何竞争或混乱，而当两个系统试图同时管理同一存储装置上的空间时可能会出这种情况。

此外，通过目标的抽象使异种机计算容易得多，面向目标的存储装置提供了这样的能力，从而至少具有操作系统能解释的组织。

此外，通过使用面向目标的存储装置，有多个原因使得群集系统中的性能提高。例如，元数据不需离开装置本身，这就免除了一定数目的 I/O 操作。

此外，装置知道在任何时刻哪些目标是打开或关闭的，并且能够使用这个信息以更有效地高速缓存数据。因为装置知道正在被读取的目标的布局，因此预取也能有效得多。存储装置能够更有效地确定顺序访问的方式。装置中的高速缓存器也能一次对于多个访问它的系统保存元数据。此外，装置能参与服务质量的判定，诸如在哪里放置数据更合适。如果装置有分派存储的职责，则一般它只能做这件事。相反，几乎没有一个操作系统能在磁盘驱动器上用区域来分派数据。于是，向磁盘驱动器本身提供这种能力就提高了其性能。

也能在安排为驱动器阵列的磁盘驱动器中实现本发明。由于存储在磁盘驱动器阵列上的信息时常比磁盘驱动器本身更有价值，因此时常把驱动器阵列称为廉价磁盘的冗余阵列(RAID)。已知数种类型的 RAID 系统或 RAID 层次。例如，第一层 RAID 的特征是提供镜象磁盘，如上所述。在第五层 RAID 中，要存储至阵列的数据以及一致校验或冗余数据在组中的所有的磁盘驱动器中散布。第五层 RAID 对所有的磁盘(包括校验磁盘)分发数据和校验信息。其他 RAID 层次(例如，层次 2－4)在名为“具有阵列支持控制器和接口的磁盘阵列”的 No. 5,617,425 美国专利中有详细描述。

图 25－29 描绘了按照本发明的一个方面进行的写入操作，其中，数据作为目标存储在阵列中的磁盘驱动器上。在图 25 描绘的实施例中，示出文件服

务器 114、请求器(或主计算机)116 和互连 122 连至磁盘驱动器阵列，它包括对象驱动器 402 和一致校验驱动器 404，它们构成存储装置(诸如存储装置 110－112)。对象驱动器 402 保存有一个要对其写入的目标，或者它的一部分，而一致校验驱动器 404 保存有与存储在对象驱动器 402 上的对象目标相关联的一致校验信息。

在图 25 中，驱动器阵列作为 RAID5 阵列实现，其中，跨过组中所有的驱动器分布数据和一致校验。因此，驱动器 402 是对象驱动器而驱动器 404 是一致校验驱动器，只对于此写入操作。换言之，对象驱动器 402 也保存有一致校验信息而一致校验驱动器 404 也保存有数据。然而，对于下述的单个写入操作，驱动器 402 是对象驱动器而驱动器 404 是相应的一致校验驱动器。也应当注意，除了 RAID 层次 5 之外，本发明也能用其他的 RAID 层次来实现。在这些 RAID 系统中，本发明对于熟悉本领域的人而言将是显而易见的。

在图 25 中，对象驱动器 402 和一致校验驱动器 404 通过纤通道(Fiber Channel)接口或者其他合适的接口(诸如其他的串行接口)互相连接。

图 26 和 27 分别描绘对象驱动器 402 和一致校验驱动器 404。每个驱动器包括控制部件 150 以及一个或多个磁盘 132。每个磁盘还包括读/写电路 406(诸如上面描述的数据磁头)和一个“异或”(XOR)电路。对象驱动器 402 包括磁盘空间 410，它存储要被写入的对象目标。一致校验驱动器 404 包括磁盘空间 412，它存储相应的一致校验目标。下面对于图 28 和 29 更详细地讨论驱动器 402 和 404 的操作。

常规的磁盘阵列实现的小型计算机系统接口(SCSI)XOR(“异或”)命令使得磁盘驱动器执行防止驱动器失效而实行的一致校验保持所需的位处理。这些命令要求主计算机(或者请求器)具有对磁盘的扇区存取，从而对于写至一个磁盘驱动器的任何扇区，在包含一致校验信息的另一个磁盘驱动器上的相应的扇区能被合适地更新。然而，上述面向目标的磁盘器在主计算机和磁盘驱动器上的实际存储扇区之间引入了一个抽象层(abstraction layer)。具体地说，磁盘驱动器把磁盘空间作为目标管理，从而主计算机(或请求器)不访问基本的扇区寻址方案。磁盘驱动器自己负责空间管理，使得请求器或主计算机不可能把写在一个磁盘驱动器上的数据一部分与在另一个磁盘上的位置相关。于是，请求器不知道在一个磁盘驱动器上的已经写入的块的地址，并且它不能计算相应的一致校验地址。如上所述，在面向目标的磁盘驱动器上

使用常规的 XOR(“异或”)功能如果不是不可能的话，那也是极为困难的。

因此，本发明提供了一种称之为 Define_Parity_Group(定义一致校验组)的方法，在构成一致校验组的一组磁盘驱动器的每个驱动器处调用该方法。此方法完成两件事情。第一，它提供足够的信息，使得能够引用标准的 Write_Object(写入目标)方法来完成作为在常规的驱动器阵列中基于扇区的 XOR 命令的相同的功能。它也使得要在组中的每个驱动器中创建的目标保存该特定驱动器对一致校验数据的共享。一致校验对象 ID 是公知的 ID(对于每个驱动器知道)，从而要更新一致校验信息的任何驱动器知道正确的目标识别符(它们对该识别符提出其请求)。

对于图 28 详地描述 Define_Parity_Group 方法。首先，请求器(或主计算机)在一致校验组的每个驱动器中引用该方法。这由方框 420 指出。为了调用该方法，请求器提供许多事项如下：

1. 构成一致校验组的驱动器的排序的列表。这可以包括(例示地)每个驱动器的编号和地址。

2. 用于计算一致校验的算法。在一种简单的例示性的实施中，对要写入的数据的块地址进行模算术运算。此算术运算得到一致校验驱动器地址(根据排序列表，从上述项目 1 得出)和在一致校验驱动器上的一致校验目标中的有关的块地址(它是包含所要的一致校验信息的一致校验目标的有关部分)。

3. 例示地以块为单位的在一致校验条(stripe)中的数据量。如果一致校验数据要在遍及每个驱动器的空间上散布，由此信息是分配的原子(atomic)单元。

4. 一致校验目标识别符。调用 Write_Object(写入目标)方法以更新一致校验目标的驱动器将一致校验目标识别符发送给一致校验驱动器上的这个目标 ID，此一致校验驱动器可如上述项目 2 提出的那样确定。还应该注意，也能实现多层一致校验(诸如两层一致校验)。于是，每个驱动器可以具有多至两个的一致校验目标。在一个例示的实施中，如果在具有两层一致校验的磁盘阵列中使用驱动器，则分派两个公知的目标 ID，并由每个驱动器保留。存在第二个一致校验目标指出正在使用两层一致校验。

5. 一致校验目标分派方针。这指出是每个驱动器分派一致校验目标作为磁盘空间的单个连续的范围还是随同用户数据目标散布一致校验目标。于是，虽然一致校验目标和数据目标在图 26 和 27 中作为连续的磁盘空间示出，但

这只是说明性的。应该注意，如果一致校验目标随同数据散布，则它仍能是预先分配的。

响应于调用 Define_Parity_Group 方法，在一致校验组中的每个磁盘驱动器中的控制部件 150 计算一致校验数据所需的空间的百分数。这由方框 422 指出。根据在一致校验组列表中的磁盘驱动器数据确定一致校验目标所需的空间大小。例如，如果在列表中有 9 个磁盘驱动器，则每个驱动器必须为一致校验信息分配其 1/9 的空间。这个空间大小用公知的一致校验目标 ID 来识别，一致校验目标 ID 由请求器或主计算机在调用所述方法后提供。这由方框 424 指出。

在一致校验组列表中的每个驱动器保存确定一致校验组的信息，从而磁盘驱动器每次通电或复位时，能够证实未曾连累一致校验组。于是，把信息存储在非易失性存储器中，如方框 426 指出的那样。

由于已经创建了磁盘驱动器的一致校验组，并且由于已经在每个磁盘驱动器上分配空间以保存一个或多个一致校验目标，因此能够更新存储在一个或多个驱动器上的数据目标中的数据。图 29 是方框图，描绘按照本发明的一个方面的数据目标的更新，以及相应的一致校验目标的更新。

为了更新数据，请求要更新数据的请求器 116 在一致校验组中的一个磁盘驱动器上调用上述 Write_Object 方法。在图 25—27 描绘的实施例中，请求器 116 在对象驱动器 402 上调用 Write_Object 方法。这由图 26 中的箭头 428 和图 29 中的方框 430 指出。为了调用此方法，请求器 116 例示性地提供一个识别要更新的目标的目标识别符、分块 ID、要在目标内写入的块的起始位置、要在目标内写入的块数、可选项信息、以及要写入的数据。对象驱动器 402 知道，运行 Write_Object 方法必须包括更新与正在更新的目标相关联的一致校验信息。对象驱动器 402 知道这一点。因为它已经存储了在非易失性存储器执行 Define_Panty_Group 方法期间提供和产生的信息。

为了更新一致校验信息，对象驱动器 402 完成多个步骤。首先，它从对象目标中的规定的位置读取老数据，并且将它连同要写至该位置的新数据提供给“异或”电路 408。这由图 29 中的方框 432 和图 26 中的箭头 434、436 和 438 指出。

接下来，对象驱动器 402 把老数据与新数据作“异或”运算，以得出中间一致校验信息。这由图 29 中的方框 440 指出。对象驱动器 402 在图 26 中

的输出端 442 处提供中间一致校验信息。接下来，目标驱动器 402 把新数据写至对象目标 410 中的对象位置，于是更新了对象目标。这由图 29 中的方框 444 指出。

然后，对象驱动器 402 自己在一致校验驱动器 404 上调用另一个 Write_Object 方法，以识别相应于刚更新的对象目标 410 的一致校验目标。这由图 29 中的方框 446 和图 27 中的箭头 448 指出。对象驱动器 402 能够用多种方法计算一致校验目标的对象位置。例如，对象驱动器 402 能够根据正在写入的块对象目标的相对扇区地址计算位置。相对地址被一致校验组中的驱动器数所除，以提供一致校验驱动器 404 上的一致校验目标中的相对地址。由在 Define_Parity_Group 方法中规定的算法确定一致校验驱动器地址。对象驱动器 402 于是构造 Write_Object 方法并且在一致校验驱动器 404 上调用它，以使用此相对地址识别一致校验目标 412 和在该目标中的适当位置。

用举例的方式，为了计算要更新的驱动器 404 上的一致校验目标中的相对块，对象驱动由 402 能够使用下式：

$$B=INT(S/D-1) \qquad \text{式 1}$$

这里 B 是一致校验目标中的相对块；

S 是正在对象驱动器 402 处写入的相对的扇区地址；以及

D 是一致校验组中的驱动器数目。

为了计算一致校验驱动器地址，对象驱动器 402 可以使用下式

$$P=Mod(S/D-1) \qquad \text{式 2}$$

这里 P 是进入一致校验驱动器的一致校验组中的驱动器列表的位移(用于计算 P 的列表必须把对象驱动器 402 的地址除外)

响应于此 Write_Object 方法，一致校验驱动器 404 理解命令为写至其一致校验目标，并且进行一致校验操作。这些操作包括读取老的一致校验数据，如图 29 中的方框 450 和图 27 中的箭头所指出的那样。然后一致校验驱动器 404 把老的一致校验信息与来自对象驱动器 402 的中间一致校验数据作“异或”运算。这由图 29 中的方框 454 和图 27 中的箭头 456 和 458 指出。“异或”运算的结果是更新写至磁盘 132 的一致校验目标的一致校验信息。这由图 29 中的方框 460 和图 27 中的箭头 462 和 464 指出。这样完全了一致校验目标的更新。

图 30 是一个数据目标 500(诸如示于图 6 中的那些)的更详细的图。按照

本发明的一个方面，数据目标 500 包括许多部分，其中包括包含属性的部分 502 以及多个数据部分 504、506 和 508，每个数据部分分别带有相关联的纠错码部分 510、512 和 514。虽然示出纠错码部分 510、512 和 514 与相应的数据部分相邻，但在磁盘上不必用这种方式记录，如此表示只是为了方便。于是，按照一个较佳实施例，使用目标 500 的每个数据部分(以及事实上可能还有属性部分)，用已知方式来产生纠错码(ECC)信息。当读回相应的用户数据，能够使用此信息来确定用户数据是否包含任何差错，并且(取决于所用代码)定位并纠正这些差错。在一个较佳实施例中，使用 Reed Solomon 码产生 ECC 信息。然而，能够使用任何合适的代码来产生 ECC 信息。

在现有技术的磁盘驱动器中，如果一个扇区不可读取，并且该扇区由操作系统为了特殊目的而使用，则这基本上能使得整个磁盘不可读。例如，如果主引导记录、分区引导记录、FAT 表、或者根目录变得不可读，则这能够造成丢失基本上整个磁盘内容。常规的操作系统面对丢失这些关键的文件系统管理数据没有能力来恢复磁盘面上的可读数据。因此，按照本发明的一个方面，在磁盘驱动器上的面向目标的数据组织使得磁盘驱动器负责保持基本文件系统结构(这在以前是操作系统的事)。按照本发明的一个方面，基本文件系统数据的冗余拷贝连同每个数据块(或数据部分)存储在其相关联的 ECC 部分。因为 ECC 部分已存储在磁盘上，把文件系统信息嵌入数据目标的 ECC 部分丝毫不会影响性能或用户容量。

图 31 是描绘了在磁盘上记录信息之前文件系统信息如何与 ECC 信息组合或嵌入 ECC 信息之中的方框图。图 31 和图 32 还描绘了按照本发明的一个方面然后如何使用文件系统信息来重建文件系统数据。图 31 示出了编码器 516、ECC 发生器 518、“异或”电路 520、磁盘 132 和读/写电路 406、ECC 发生器 522、译码器 524 和“异或”电路 518。应该注意，编码器 516、ECC 发生器 518、“异或”电路 520、译码器 524、ECC 发生器 522 和“异或”电路 526 都能在磁盘驱动器上的控制部件 156 中实现，或者能单独实现。

首先把用户数据从主计算机、请求器或文件服务器提供给编码器 516。编码器 516 按照预定的编码算法编码(一般地实施算法用以降低差错率)。然后把经编码的用户数据提供 ECC 发生器 518。ECC 发生器根据经编码的数据用已知的方式产生 ECC 信息。产生的 ECC 信息将取决于所用的纠错编码方案的特殊类型。又将 ECC 信息提供给“异或”电路 520。在输入端 521 处把文件系统

数据也提供给“异或”电路 520。在图 31 描绘的实施例中，文件系统数据是位置信息，它识别要把用户数据写在磁盘 132 上的位置。例如，在上述的面向目标的系统中，位置信息包括目标识别符，它识别用户数据所属的目标。位置信息还包括相对位置信息，它识别在被识别的目标内的相关联的数据部分的相对位置。于是“异或”电路 520 的输出提供其中嵌有(或种有)定位信息的 ECC 信息。对于包含由编码器 516 提供的经编码的用户数据的数据部分，把这个信息提供给读/写电路 406 并且作为相关联的 ECC 部分写至磁盘 132。

当从磁盘 132 读回信息(为了完成正常的读出操作)时，控制部件 150 通过提供所希望的位置信息至“异或”电路并且把该信息与 ECC 信息(它包含嵌入的位置信息)作“异或”运算而执行 Read_Object 功能。“异或”电路的输出端得到与正在读取的用户数据相关联的 ECC 信息。把这个信息提供给 ECC 发生器，它判定是否在经编码的用户数据中出现任何差错。如果没有，则把经编码的用户数据提供给译码器，在该处，把无差错的信息呈现给请求器或用户。如果出现差错，则控制部件 150 可能打算识别和纠正差错，这取决于所用的特殊的差错编码方案。另一种做法，控制部件 150 可以只提供差错标志，该标志指出数据包含有一个或多个无法纠正的差错。

然而，如果系统信息(在给出的例子中是位置信息)已经丢失，则控制部件 150 以不同的方式运作。控制部件 150 使得在磁盘 132 上的数据被读取。这由图 32 中的方框 528 指出。把经编码的用户数据提供给译码器 524 和 ECC 发生器 522。应该注意，ECC 发生器 522 和 ECC 发生器 518 可以是采用合适的多路复用电路的同一个发生器。然而，为了清楚起见，在图 31 中把它们画成分开的部件。

ECC 发生器 522 根据经编码的用户数据产生 ECC 信息，如图 32 中的方框 530 指出的那样。把这个信息提供给“异或”电路 526。ECC 信息(它包含嵌入的位置信息)也从磁盘 132 读出，并且提供给“异或”电路 526。这由方框 532 指出。与 ECC 发生器 522 那样，“异或”电路 526 与“异或”电路 520 可以是采用合适的多路复用电路的同一个电路。然而，为了清楚起见，把这两个电路分别示出。

通过把由 ECC 发生器 522 提供的 ECC 信息和包含嵌入的位置信息的 ECC 信息作“异或”运算，在至“异或”电路 526 的两个输入中的 ECC 信息将互相抵消，导致只是位置信息的输出。这由方框 534 指出。能够把这个信息与

由译码器 524 输出的用户数据连用，以重建已经丢失的文件系统信息。这由方框 536 和 538 指出。例如，使用从磁盘检索得的位置信息，以及与用户数据相关联的信息亦从磁盘读出，现在能够重建目标目录。

按照本发明的另一方面，通过使用伪随机数(或伪噪声)发生器，能够使 ECC 发生器 518 产生的 ECC 信息随机化。图 33 是描绘这一实施例的方框图。图 33 所示的许多项目与图 31 所示的相似，而标号也相似。示于图 33 的方框图的运作基本上与图 31 所示的方框图的运作相同。然而，不是在输入端 521 处提供位置信息至“异或”电路，而是用位置信息作为种子由伪噪声(PN)发生器产生一个随机数。于是，在输入端 542 处把位置信息提供给 PN 发生器 540。根据种子值，PN 发生器产生一个提供给“异或”电路 521 的输出，把该输出与由 ECC 发生器 518 提供的 ECC 信息(它与相关联的经编码的用户数据记录在磁盘 132 上)作“异或”运算。

为了重建文件系统信息(例如，位置信息)，从磁盘 132 读取经编码的用户数据，并且把它提供给 ECC 发生器 522，该 ECC 发生器产生 ECC 信息并且把它提供给“异或”电路 526。包含嵌入伪随机值的 ECC 信息也从磁盘 132 读出，并且提供给“异或”电路 526。“异或”电路 526 的输出端得到随机值，它已以位置信息为种子。将此值提供给逆 PN 发生器 544，它把由 PN 发生器 540 提供的随机化过程倒过来，并在春输出端 546 提供位置信息种子值。如图 31 所示的实施例那样，能够用此信息连同由译码器 524 提供的经译码的用户数据，来重建先前丢失的文件系统结构信息。

这里描述的“异或”门 520 和 526 例示地是字节宽的“异或”电路，用于对数据字节内的各个位作“异或”运算。于是，“异或”电路实在是 8 个分开的“异或”门，以提供 8 位字节的“异或”功能。此外，虽然这里描述的本发明涉及“异或”门，但对于作为纠错和检错码基础的域任何合适的 Galois 域处理(或相加)、或其他合适的处理应视为在本发明的范围内，并能被熟悉本领域的人实施。

还有，在一个较佳实施例中，在 1998 年 2 月 10 授予 French 等人的第 5，717，535 号美国专利更详细地描述了 PN 发生器 540。该专利把发生器描述为具有 33 寄存器单元、带有连至逻辑块的输入端和输出端。寄存器单元是一位宽，并且如数据字节那样在相同的时钟上计时，发生器足以保存高至 4 字节(32 位)长的一个目标识别符和相对位置信息，但能容易地扩展，以容纳更大的位

置信息或其他的文件系统信息(大于 4 字节)。例示地，该发生器还包含一个额外的寄存器单元，使用它从而具有零值的位置信息将不在 PN 发生器 540 的输出端产生全零。没有理由必须把这个额外的单元包括在 PN 发生器 540 中，如果使用它完全是为了用 ECC 信息来种植文件系统信息。然而，如果为了其他理由(即，容错)用发生器 540 来使数据随机化，则应该包括额外的单元，从而全零输入将提供非零输出。例示地用时钟对数据计时，其速率为每 8 个数据位一次(即每个字节一次)。

在例示的实施例中，发生器 540 包括多个下面的式 3 和式 4 运作的触发器，这里 B 代表至触发器的输入，而 A 代表从触发器的输出。

$$B_I = A_{I+8} \quad ; \qquad \text{式 3}$$

对于 I＝0 至 24；而

$$B_I = A_m \text{XOR } A_{m+13} \quad ; \qquad \text{式 4}$$

对于 I＝25 至 32，M＝(I＋8)模 33。

应该注意，发生器 540 例示地等效于一个基于本原多项式 $X^{33}+X^{13}+1$ 的二进制反馈移位寄存器，并且每时钟节拍移位 8 次。驱动输入至寄存器单元的逻辑块代表这些 8 次位移的结果。根据这个类似，很显然在发生器 540 输出端处提供的字节的序列每 $2^{33}-1$ 个字节重复一次。

于是，可以看出，本发明提供了胜过现有系统的显著的优点。本发明允许读出用户数据，即使它在丢失关键的文件系统信息〔诸如文件系统结构数据〕的磁盘面上。本发明把文件系统信息(诸如文件系统结构信息)嵌在与目标的数据部分对应的 ECC 信息中。于是，只通过读取在磁盘上的数据和逆转嵌入过程，就能够读出文件系统信息，于是能够重建文件系统信息。

图 34－36 描绘了按照本发明的另一方面的另一种写入操作，其中，驱动器阵列作为混合 RAID 构造来实现。如前面提到的，存储在磁盘驱动器阵列中的数据时常比磁盘驱动器本身更有价值。因此，除了主数据之外，提供了一个廉价磁盘冗余阵列(RAID)来存储冗余或一致校验数据，以在主数据变得有毛病或不可存取时，能够重建它。已知数种类型的 RAID 系统或 RAID 层次。

第一层 RAID(RAID 层次 1)以提供镜象磁盘驱动器为特征。在第一层 RAID 中，阵列中所有的驱动器都加倍。于是，如果一个驱动器失效，信息也不会丢失，因为确切的信息被镜象存储在另一个磁盘驱动器上。对于实现磁盘驱动器阵列而言，这是一种很昂贵的选择方案，因为硬件要加倍。

第二层 RAID 包括用于纠错的 Hamming 码。在第二层 RAID 中，跨越阵列的驱动器对于数据作位交织(bit-interleave)，并且添加校验驱动器，以检测和纠正单个差错。这样做有缺点，即，如果只读取少量数据，但仍须读取来自组中的每个位交织驱动器的整个扇区。同样，写单个单元仍然包括在组中的所有驱动器上的读一修改一写(read-modify-write)循环。

第三层 RAID 以每组驱动器有单个校验驱动器为特征。在第三层 RAID 中，取消了在第二层 RAID 中用于存储纠错码信息的额外的校验驱动器。相反，由于数据正在存至磁盘驱动器，把 ECC 信息添加至数据。还有，相应于存储在阵列中的数据，用单个磁盘驱动器来存储冗余数据。当从阵列读取信息时，用 ECC 信息来确定是否出现差错以及哪个驱动器包含有差错。于是，通过计算其余好的驱动器的一致校验性并且加以比较，以及对起初的全组数据计算并且存储冗余或一致校验磁盘驱动器上的一致校验作逐位比较，即可在出故障的驱动器上重建信息。

第四层RAID以安排得对独立的读和写提供为特征。在第二层和第三层RAID的实施中，存储在阵列中的信息要跨越组中所有的驱动器而散布。因此，对于组中一个驱动器的任何读和写的操作需要对组中所有的驱动器读和写。第四层 RAID 通过提供在任何给定时刻对每个驱动器组做一个以上的 I/O 操作的能力而改进了小传递的性能。不再跨越几个驱动器散布每个数据扇区。相反，把存储在阵列中的每个数据扇区作为在单个驱动器上的单独的单元保存。存储在阵列中的信息在数据磁盘中是在扇区的水平上而不是在位的水平上交织。

在第五层 RAID 中，要存储至阵列的数据以及一致校验或冗余数据都经组中的所有驱动器散布。虽然第四层 RAID 允许在任何给定的时刻在每组中进行多于一个的读操作，它仍限于每组一个写操作，因为每个写操作需要访问校验驱动器。第五层 RAID 把每个扇区的数据和校验信息跨越所有的驱动器(包括校验驱动器)散布。因此，第五层 RAID 能够支持每组的多次单独的写操作。因为对于每个顺序位置的检验信息是在组中的不同的驱动器上，因此写操作能并行进行，因为无需在给定时刻顺序访问任何一个驱动器。

虽然上面的讨论提供了不同层次的 RAID 系统之间的某些主要区别的概述，但对于这些区别的更详细的描述以及说明性的例子在 Patterson、Gibson 和 Katz 的题为“一种用于廉价磁盘的冗余阵列的情形”论文中提供。

由于在 RAID3 和 RAID4 与 5 型系统之间的特性区别，不同的系统特别适合于不同的需要。RAID3 系统已经典型地适合于需要呈现高数据传递速率的阵列系统并且显示在其中具有良好的性能。另一方面，RAID4 和 5 系统典型地在高聚集输入/输出(I/O)应用中的磁盘阵列中显示出良好的性能。在事务应用或具有许多个 UNIX 用户的应用中时常找到这些实现。

虽然 RAID 层次 5 对于事务应用是最常用的，但它不如 RAID1 系统可靠。特别，RAID5 系统使用普遍，因为它对磁盘空间的使用最有效，因而在硬盘方面需要较少的投资。然而，如果丢失的主数据的数量超过根据在 RAID5 系统中的冗余数据来恢复数据的能力，则不能重建主数据。因此，RAID5 系统易受完全的数据丢失的影响。

相反，RAID1 系统不易受完全的数据丢失的影响，因为在另一个驱动器上作了主数据的完全备份(即，数据被镜象存储)。于是，如果丢失了任何或所有的主数据，可以使用镜象数据来完全重建主数据。然而，由于 RAID1 系统需要完全加倍磁盘驱动器，因此它可能成本过高。

本发明的混合数据重建系统既提供了 RAID5 系统的成本上的优点，又提供了 RAID1 系统的性能上的优点。通过把 RAID1 系统应用于被确定为对用户是重要的或关键的数据，而把其他的 RAID 技术之一(例如 RAID5)应用于其他的非关键数据。虽然下面的描述说明了本发明的实施例为一种混合的 RAID1/RAID5 构造，但熟悉本领域的人将理解，通过组合任何两个冗余方案(诸如 RAID1 构造与 RAID2、RAID3 或 RAID4 构造)可以得到类似的好处。为了简单起见，并且只是作为说明，下面的描述集中在混合的 RAID1/RAID5 技术上。

由于对于用户来说最重要和关键的数据往往是存取最频繁的数据，因此一种识别关键数据的有效技术是识别高频度数据。高频度数据是指比对于在磁盘驱动器上的其他数据访问次数明显多的数据。熟悉本领域的人将理解，本发明的混合 RAID 技术既应用于块数据系统，又可应用于面向目标的系统。

最重要和关键的数据也可由用户指定，而与数据被存取的频度无关。例如，用户可以指定某个文件名、某种文件类型、某个目录、或者它们的组合作为关键数据。通过将关键数据与块地址相关(当用在面向块的数据中时)或者与目标类型或目标属性相关(当用在面向目标的数据中时)，可以对此数据作出标志。

如果把关键数据作为该数据被存取的频度的函数来识别，则磁盘驱动器

控制器 148、或磁盘驱动器阵列控制器 602 或计算机系统的其余合适的部分可以用来自动地记录数据或数据目标被存取的次数。这样，最好将每个数据块或数据目标被存取的次数记录入非易失性存储器。使用这个直方图(histographical)数据，如果数据块或数据目标例如正在被存取超一个阈值频度(即，超过存取次数的阈值)，就用 RAID1 技术(即，将它们作镜象存储)。所有正在存取的低于阈值频度的其他数据用 RAID2、3、4 或 5 技术存储。因此，由于关键数据完全被镜象存储在另一个磁盘驱动器上，因此它不容易完全丢失。相反，非关键数据用空间更节约的 RAID 技术(例示地，用 RAID5 技术)备份。这适用于或者由用户指派的标志来识别关键数据，或者由磁盘驱动器控制器 148 或阵列控制器 602 来检测高频度使用而识别。

图 34 描绘按照本发明的一个方面的实现混合数据存储和重建的磁盘驱动器阵列 600 的方框图。磁盘驱动器阵列 600 包括连至磁盘驱动器 604、606、608、610 和 612 的阵列的阵列控制器 602。每个驱动器说明性地包括一个驱动器控制器 148(在图 34 中未示出)。阵列控制器 602 还连至互连 122，如图 1 和 25 所示。在描绘的特定的实施中，每个磁盘驱动器 604、606、608、610 和 612 包括关键数据部分 614、非关键数据部分 616、镜象存储位置 618 和一致校验位置 620。

分配给镜象存储位置的空间 618 最好正比于关键数据 614。类似地，在每个磁盘上为一致校验位置提供的空间 620 最好正比于相应于由非关键数据 616 占有空间的一致校验数据。然而，熟悉本领域的人将理解，根据用户对于特殊应用所需的性能和经济之间的平衡，可以修改分配给镜象存储的位置 618 和一致校验的位置 620。然而，说明性地，关键数据 614 占据了磁盘空间的相对较小的数量(例如，9GB 驱动器中的 20MB)，因而相应的镜象存储位置 618 成比例地占有磁盘空间的较小数量。因此，由于在每个磁盘上用于镜象存储的位置 618 的磁盘空间的总量相对较小，因此在磁盘空间和性能之间的折衷不是实质性的。

图 34 描绘了混合的 RAID1/RAID5 驱动器阵列 600，其中，主数据、镜象数据和一致校验数据在组中跨越所有的驱动器 604、606、608、610 和 612 分布。因此，驱动器 604 为在驱动器 606 中的关键数据保存镜象数据。驱动器 604 还包含驱动器 606 中的非关键数据的一致校验数据。如所描绘的，在驱动器 606/608、608/610、610/612、612/604 之间也存在类似的关系。通过合适的

接口(诸如纤通道接口或其他串行接口)，每个驱动器 604、606、608、610 和 612 互相连接。

各个驱动器 604－612 的构造和操作基本上类似于在图 26 和 27 中描绘的驱动器。特别，磁盘驱动器与 RAID 层次 5 的体系结构相似，而与 RAID 层次 1 的体系结构不同。具体而言，在每个驱动器 604－612 中的每个磁盘除了一致校验位置 620 外包括镜象存储位置 618，而在图 26 和 27 中描绘的驱动器没有镜象存储位置。然而，除了这些描述的之外，结构和操作基本上是相同的。

阵列控制器 602 包括命令部件 624，它对每个驱动器 604－612 发出指令，以将新数据作为镜象数据或一致校验数据写入。命令部件 624 按照从关键数据检测器部件 622 接收到的输出而对驱动器 604－612 发出指令。关键数据检测器部件 622 对输入的新数据扫描，以发现由用户预定的或由高频度数据登录 626 分配的标志。如果由关键数据检测器 622 检测得一个标志(例如，块地址或目标属性)，则阵列命令部件 624 使新数据被写至其目的地并被镜象存储至位置 618。如果未检测到标志，则阵列命令部件 624 使新数据写至其目的地，并使相关联的一致校验信息写至一致校验位置 620。

可以使用高频度数据登录 626，以登录特定的数据块或数据目标被存取的次数。如果特定的数据块或数据目标已被存取的次数超过阈值次数(它可由用户规定)，则高频度数据登录 624 对于由关键数据检测器 622 的检测指定该数据块或数据目标为关键数据。如果使用面向目标的数据，并且目标属性之一已经包括存取频度，则可省略高频度数据登录 626。此外，如果作为除存取频度之外的某个量的函数指派关键数据，则可省略数据登录 626。

图 35 和 36 描绘了图 34 所示的磁盘阵列 600 的写入操作 630 和 660。具体而言，图 35 描绘对于面向块的数据的写入操作，而图 36 描绘对于面向目标的数据的写入操作。

特别参看图 35，写入操作 630 在方框 634 处开始，但用户可以预先指定一个标志，它附加于数据或对于数据预先考虑，并且它把数据识别为关键数据，如在方框 632 中那样。这样一个标志可以是(但不限于)与数据相关联的块地址。如果用户对于关键数据预先指定标志，则可跳过在方框 633 中描述的登录步骤。如果没有预先指定标志，则写入操作 630 可以用在方框 633 中描述的登录步骤开始。

假设没有预先指定标志，控制器 602 的命令部件 624 读取和登录在登录

626 中的新数据的块地址，如方框 633 所示。块地址被存取的次数(即，频度)在登录 626 中被跟踪和记录。如果存取次数(频度)超过阈值，则标志该块地址为关键数据。

在接收到新数据后，控制器 600 检查登录 626 并且判定作为新数据的目的地的块地址是否已被标记为关键性的，如在方框 636 中反映出的那样。如果如判定框 638 所指出的那样标志被检测，则把新数据作为关键数据处理，并且写入至其目的地，和写至镜象存储位置 618，如方框 640 所反映的那样。如果判定框 638 指出未检测到标志，则部件 624 更新频度值，如方框 639 指出的那样，并且实施 RAID5 例行程序，如方框 642－650 指出的那样。熟悉本领域的人将理解，可以使用其他合适的 RAID 或别的冗余性例行程序，包括(但不限于)RAID2、3 和 4 例行程序。

采用 RAID 5 例行程序，在对象位置处读取老数据，如方框 642 指出的那样。在一致校验位置处还读取老的一致校验数据，如方框 644 指出的那样。在对象位置处，把老数据与新数据作“异或”运算，得出新的一致校验数据，如方框 648 指出的那样。然后把新的一致校验数据写至相关联的一致校验位置 620，如方框 650 指出的那样。根据判定框 638 来启动 RAID1 例行程序或 RAID5 例行程序，把新数据也写至对象位置，如方框 652 指出的那样。在把新数据写至对象位置后，写入操作就完成了，如方框 654 指出的那样。

参看图 36，它描绘了对于使用面向目标的数据的写入操作。写入操作 660 在方框 662 处开始。根据目标类型，可以预先指定块的关键性。例如，可能希望在任何时刻把目录信息(诸如 POL166)或任何确定存储装置结构或在其上存储的数据的任何其他目标(诸如 DCO154、DAO158 或 PCO164)、或它们的任何组合加以镜象存储。此外，由用户或其他实体，可在逐个目标的基础上使用 OID、文件名、或某些其他属性来预先指定某些目标。于是，命令部件 624 接收和检索目标，以判定是否预先指定关键性。这由方框 664 和 666 指出。说明性地把这种预先指定存储在控制器 602 的存储器中。如在判定框 668 中指出的那样，如果检测到关键性，则把新数据如方框 670 指出的那样写至镜象存储位置 618 和如方框 682 指出的那样写至其目的地。

如果未检测到预先指定的关键性，如在判定框 668 中指出的那样，则部件 624 根据它被存取的频度判定正在被存取的目标是否关键性的。通过检查高频度数据登录 626(说明性地它包括 OID 和指出存取频度的相关联的频度

值)，部件 624 做这件事。这由方框 669 示出。如果根据频度，目标是关键性的(其相关联的频度值超过预定的阈值或自适应地调节的阈值)，则部件 624 把数据作镜象存储，如方框 670 指出的那样。

如果根据频度，目标不是关键性的，则部件 624 把 OID 登录在登录 626 中(如果它是一个新的目标)，并且更新相关联的频度值。这由方框 671 指出。然后把数据按照 RAID2、3、4 或 5 实施或其他冗余性方案加以存储。

采用 RAID5 实施，在对象位置处读取老数据，如方框 672 指出的那样。然后在一致校验位置处读取老的一致校验数据，如方框 674 指出的那样。在对象位置处把老数据与新数据作“异或”运算，导致新的一致校验数据，如方框 678 指出的那样。把新的一致校验数据写至一致校验位置，如方框 680 指出和在图 34 描绘的那样。不管在判定框 668 中是否检测出关键性，都把新数据写至对象位置，如方框 682 指出的那样。在把新数据写至对象位置之后，完成了写入操作 660，如方框 684 指出的那样。

还应该注意，在图 35 和 36 中提出的步骤能够在与任何一个或数个驱动器 604－612 相关联的一个驱动器控制器 604－612 上执行，然后把数据传送至要在合适的驱动器上写的合适的驱动器控制器。在此时，驱动器控制器设置有部件 622－626。处理基本上与图 35 和 36 示出的那些相同。

应该注意，根据驱动器使用，能够把频度值预先指定为一个原始数目或者一个变数。例如，能够把阈值设置为总的驱动器存取的一个百分数，并且能够随每个存取(即，随每个读写操作)而自适应地更新。此外，能够使用多个阈值，例如，对于每个目标类型或者目标类型组有一个阈值。也能够使用其它自适应技术。如果自适应地设置阈值，则在每次存取时必须对每个块或目标更新频度值。

此外，使用冗余性方案来存储关键数据且同时用直接编码方案来存储其余的数据能够得到本发明的好处。直接编码方案是指，经编码的数据只是存储在一个磁盘驱动器上，而不增添冗余性数据。因此，作为单个例子根据任何其他的冗余性方案，能够镜象存储或存储关键数据，而非关键数据只是以编码形式加以存储。在这种情形下，并且再参看图 34，非关键数据 616 没有相应的存储在一致校验位置 620 中的一致校验数据。于是，在图 34 中作为一致校验位置 620 所花费的空间可以用来存储额外的关键数据、非关键数据或相应于关键数据的冗余数据。类似地，能够使用两种或多种冗余性方案以及

直接编码方案来实现本发明。在该情形下，能够按照高度冗余的冗余性方案来存储高度关键性的信息，按照较少冗余的冗余性方案来存储中等关键性的数据，而按照直接编码方案来存储非关键数据。

还应该注意，术语“阵列”是指磁盘驱动器的一个集合，这些磁盘驱动器存放在地理上互相远离的位置中，诸如在不同的房间、不同的建筑物或由很大的距离分隔开的位置中。

如从上面可见，除了在图 34 中描述的混合阵列系统构造 600 之外，在图 35 和 36 中描述的混合存取操作 630 和 660 提供了 RAID5 系统的成本方面的好处和 RAID1 系统的可靠性方面的好处。

本发明的一个实施例是磁盘驱动器 600，它包括多个磁盘驱动器 604－612 和至少一个控制器 602(该控制器在操作上耦合至多个磁盘驱动器 604－612)，构造来接收数据和按照第一冗余性方案在磁盘驱动器 604－612 上存储数据的第一部分，并按照第二冗余性方案在磁盘驱动器上存储数据的第二部分。第一冗余性方案最好比第二冗余性方案提供更大的冗余性自由度。

按照第一冗余性方案，数据的第一部分包括冗余性数据，它与数据的第一部分不同，而控制器 602 构造来存储数据的第一部分和在磁盘驱动器 604－612 上的冗余性数据。

控制器 602 构造来在多个磁盘驱动器的第一驱动器或第一组驱动器上存储数据的第二部分，并且把数据的第一部分镜象存储在多个磁盘驱动器 604－612 的第二驱动器或或第二组驱动器上。

数据的第一部分可以包括诸如元数据这样的数据，它要比数据的第一部分更频繁地被存取。

控制器 602 可以构造来存储作为结构安排中的目标的数据的第一部分和第二部分，而数据的第一部分可以包括结构目标，它包括指出结构安排的信息。

在此上下文中，控制器 602 可以构造来以分区的形式存储目标，而结构目标可以包括装置控制目标、装置相关性目标、分区控制目标或分区目标列表。

数据的第一和第二部分可以作为目标存储，每个目标包括属性，其中，数据的第二部分包括属性。

第一冗余性方案可以是 RAID 层 2、3、4 或 5，而第二冗余性方案可以是

RAID 层 1。

控制器 602 可以构造来确定数据被存取的频度，并且根据其被存取的频度把数据分为第一和第二部分。

在此上下文中，控制器 602 可以构造来存储作为对象的第一和第二数据部分(因目标具有相关联的文件名)并且可构造来跟踪根据文件名存取数据的频度。

另外，控制器 602 可以构造来存储作为目标的第一和第二数据部分，每个目标具有相关联的目标类型，并且可构造来根据目标类型，在第一和第二数据部分之间划分数据。

根据用户输入，可以在第一和第二数据部件之间划分数据。

每个磁盘驱动器 604－612 可以包括一个驱动器控制器 148，其中，控制器 602 包括一个或多个驱动器控制器 148。

主机控制器可以耦合至驱动器控制器 148，并且控制器 602 可以是主机控制器。

在本发明的另一实施例中，磁盘驱动器阵列 600 包括磁盘驱动器 604－612 的一个阵列和阵列控制器装置 602，用于有选择地在磁盘驱动器 604－612 之间镜象存储数据。

在本发明的还有一个实施例中，在磁盘上、在磁盘驱动器中的一种存储数据的方法包括按照第一冗余性方案存储数据的第一部分；以及按照不同于第一冗余方案的第二冗余方案存储数据的第二部分。

此方法可以包括根据用户输入，根据例如数据被存取的频度或者根据数据的内容，判定数据是在第一部分或第二部分。

应该明白，即使在上面的描述中已经提出了本发明的各个实施例的许多特征和优点，还给出了本发明的各个实施例的结构细节和功能，这个揭示还是说明性的，因此可在细节方面作改变，尤其是在本发明的原理的整个范围之内，对各个部分的结构和安排作改变，由所附的权利要求书用术语的广泛的一般意义表述本发明的原理的范围。例如，不偏离本发明的范围和精神，根据特殊的连接方法，冗余性方案或检错或纠错方案可以改变特殊的元件，与此同时保持基本上相同的功能。

说 明 书 附 图

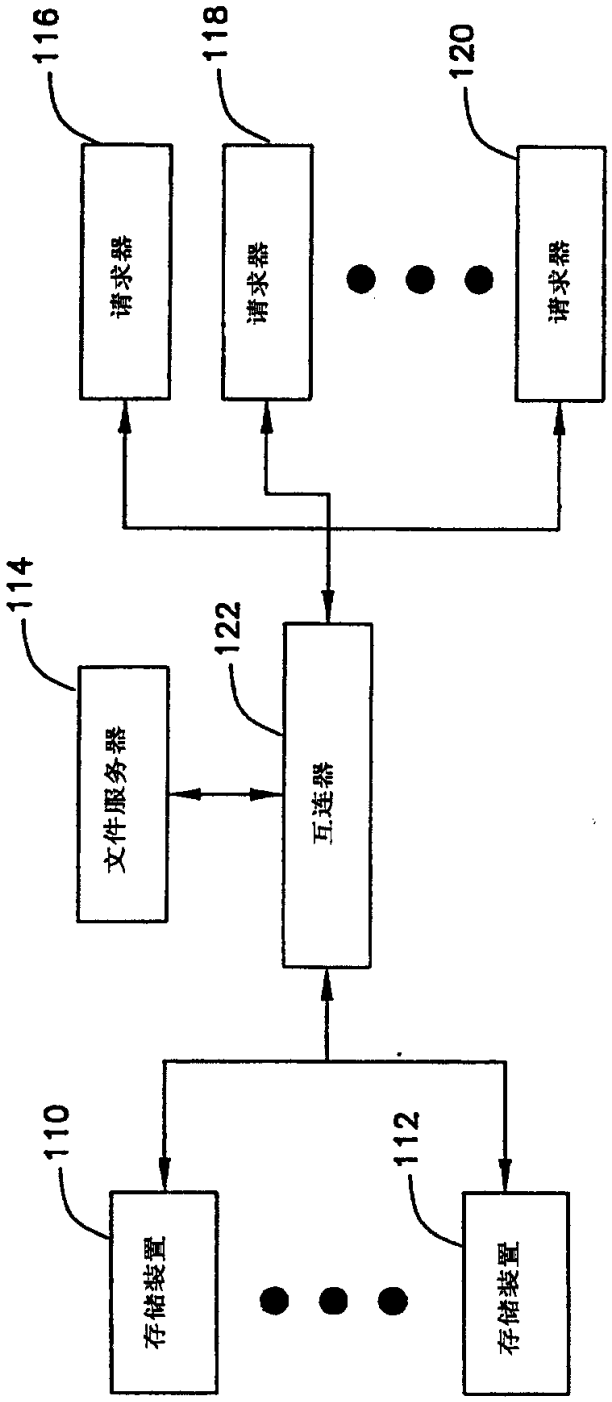


图 1

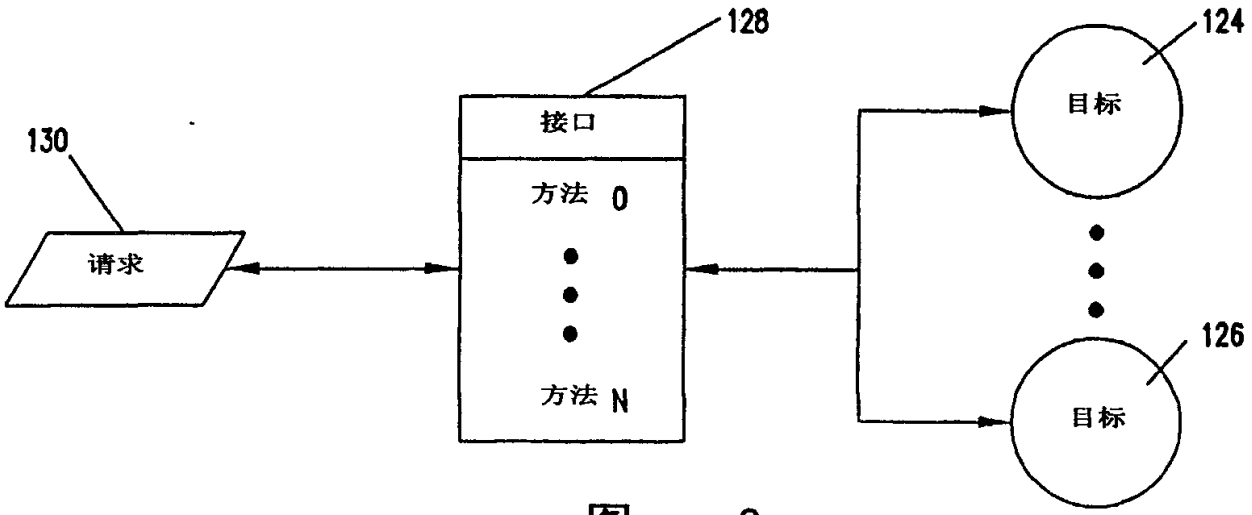


图 2

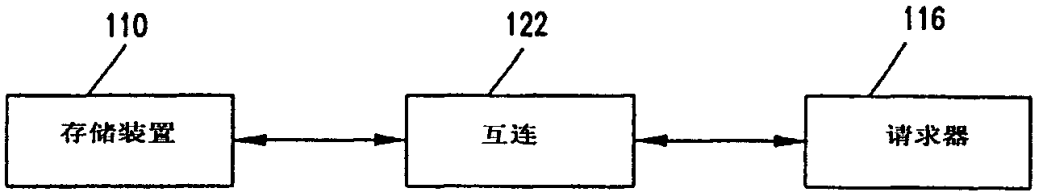


图 3-1

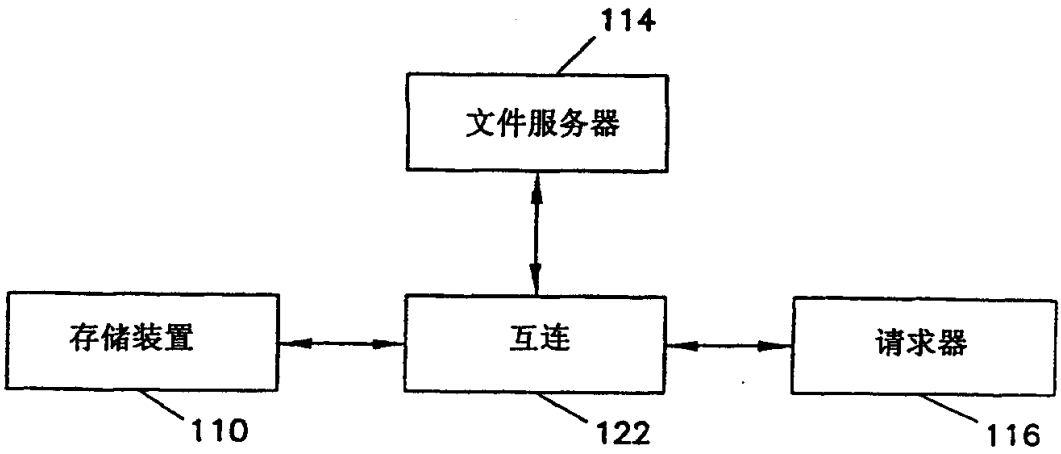


图 3-2

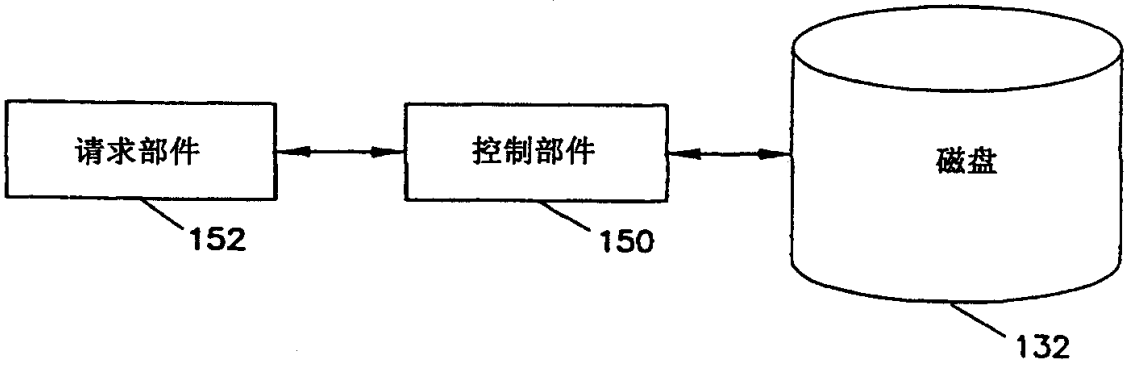


图 5

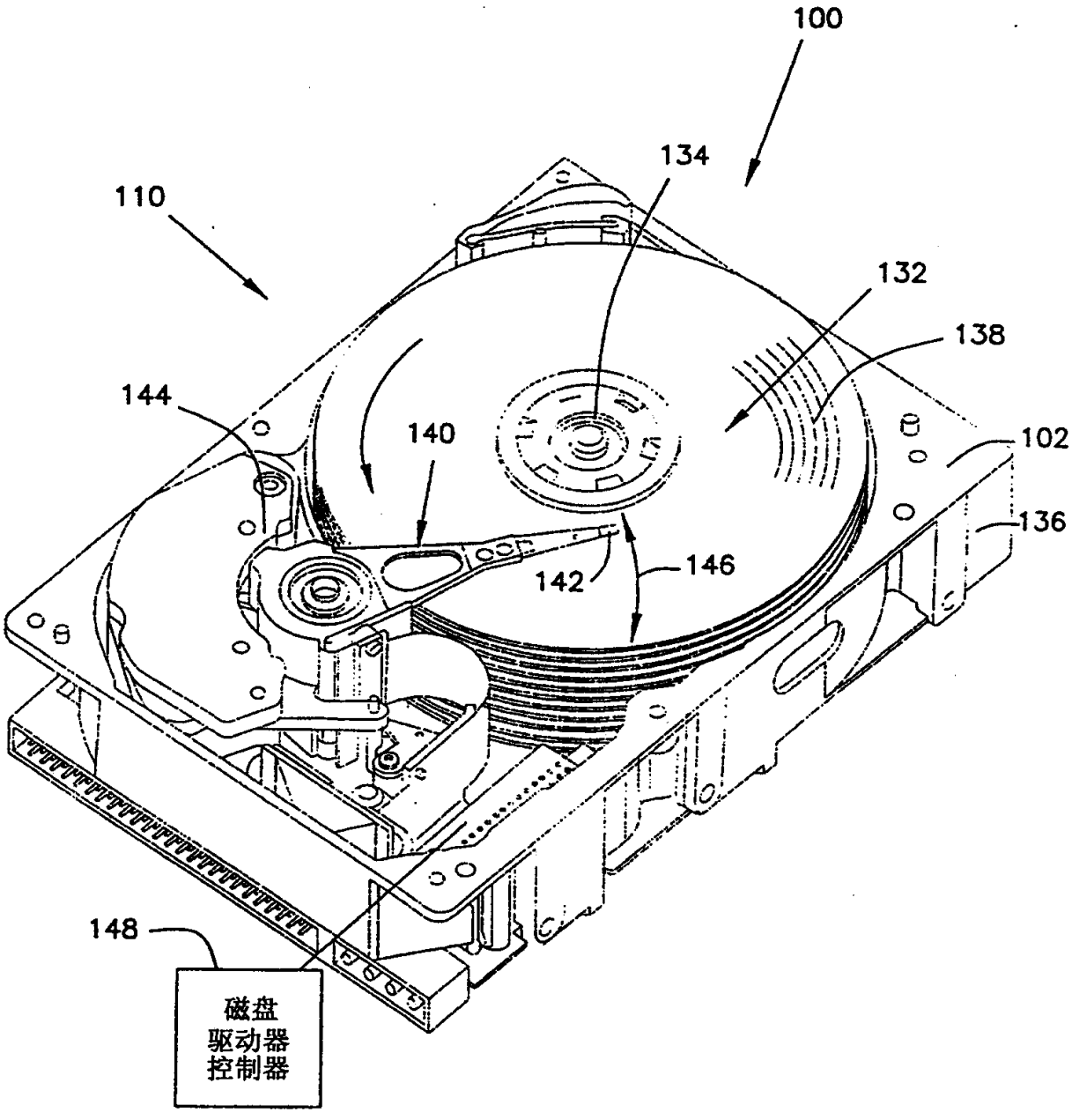


图 4

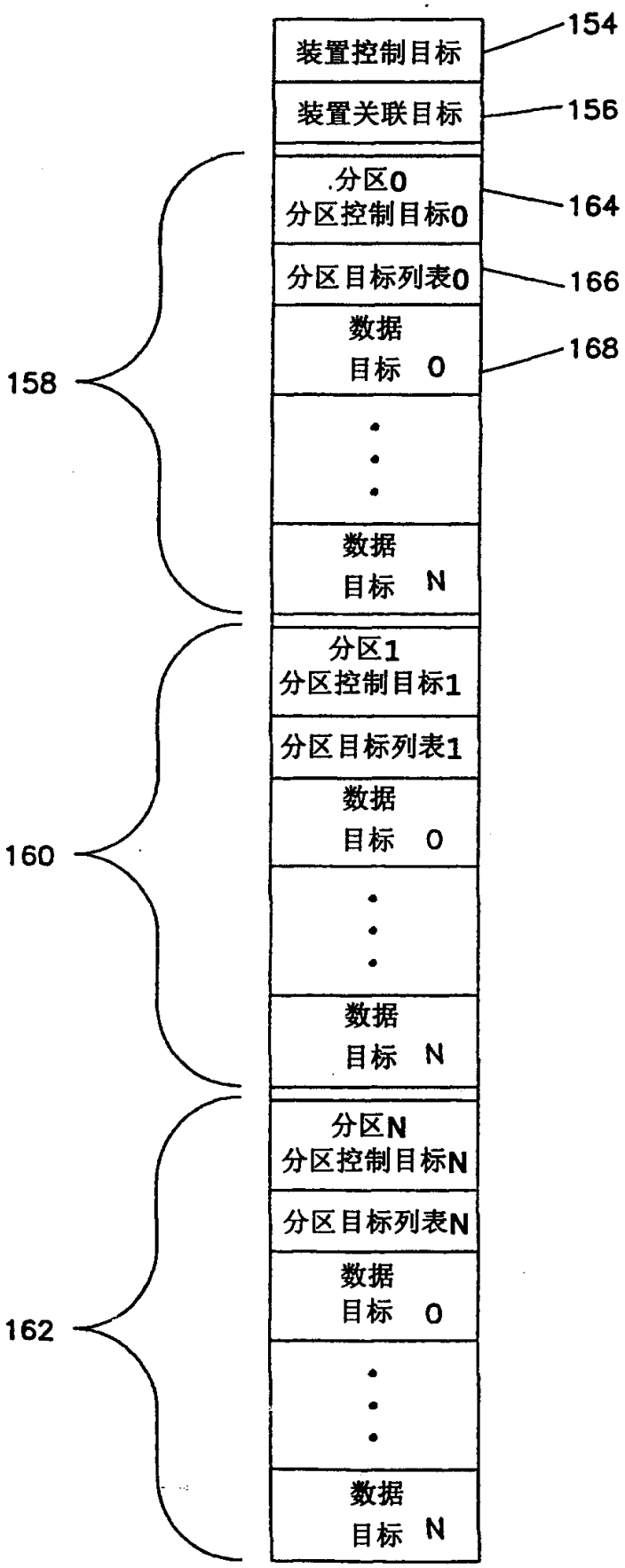

装置控制目标
154
装置关联目标
156
.分区0
分区控制目标0
164
分区目标列表0
166
数据
目标 0
168
158
数据
目标 N
分区1
分区控制目标1
分区目标列表1
数据
目标 0
160
数据
目标 N
分区N
分区控制目标N
分区目标列表N
数据
目标 0
162
数据
目标 N

图 6

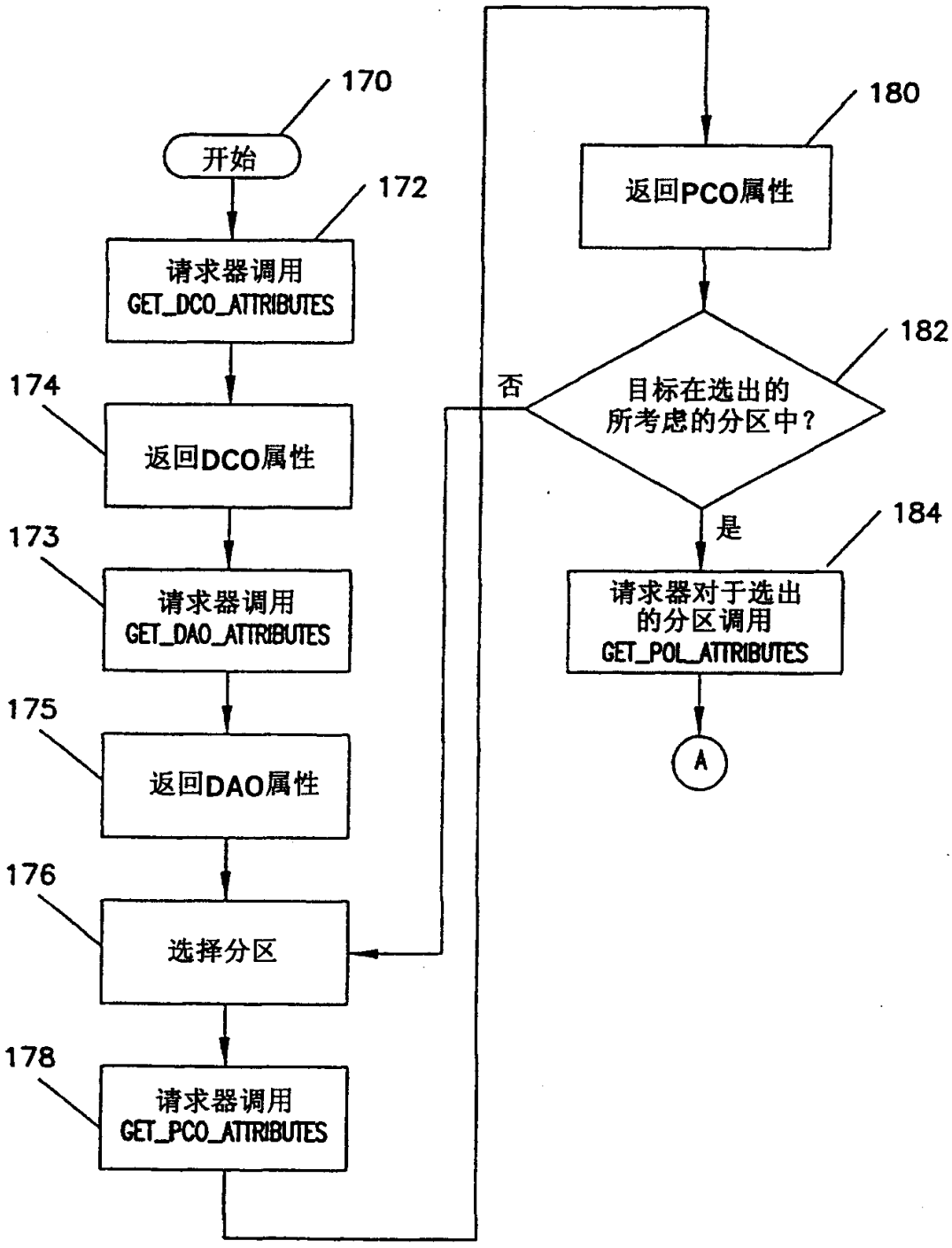

170
开始
172
请求器调用
GET_DCO_ATTRIBUTES
174
返回DCO属性
173
请求器调用
GET_DAO_ATTRIBUTES
175
返回DAO属性
176
选择分区
178
请求器调用
GET_PCO_ATTRIBUTES
180
返回PCO属性
182
目标在选出的
所考虑的分区中？
否
是
184
请求器对于选出
的分区调用
GET_POL_ATTRIBUTES
A

图 7-1

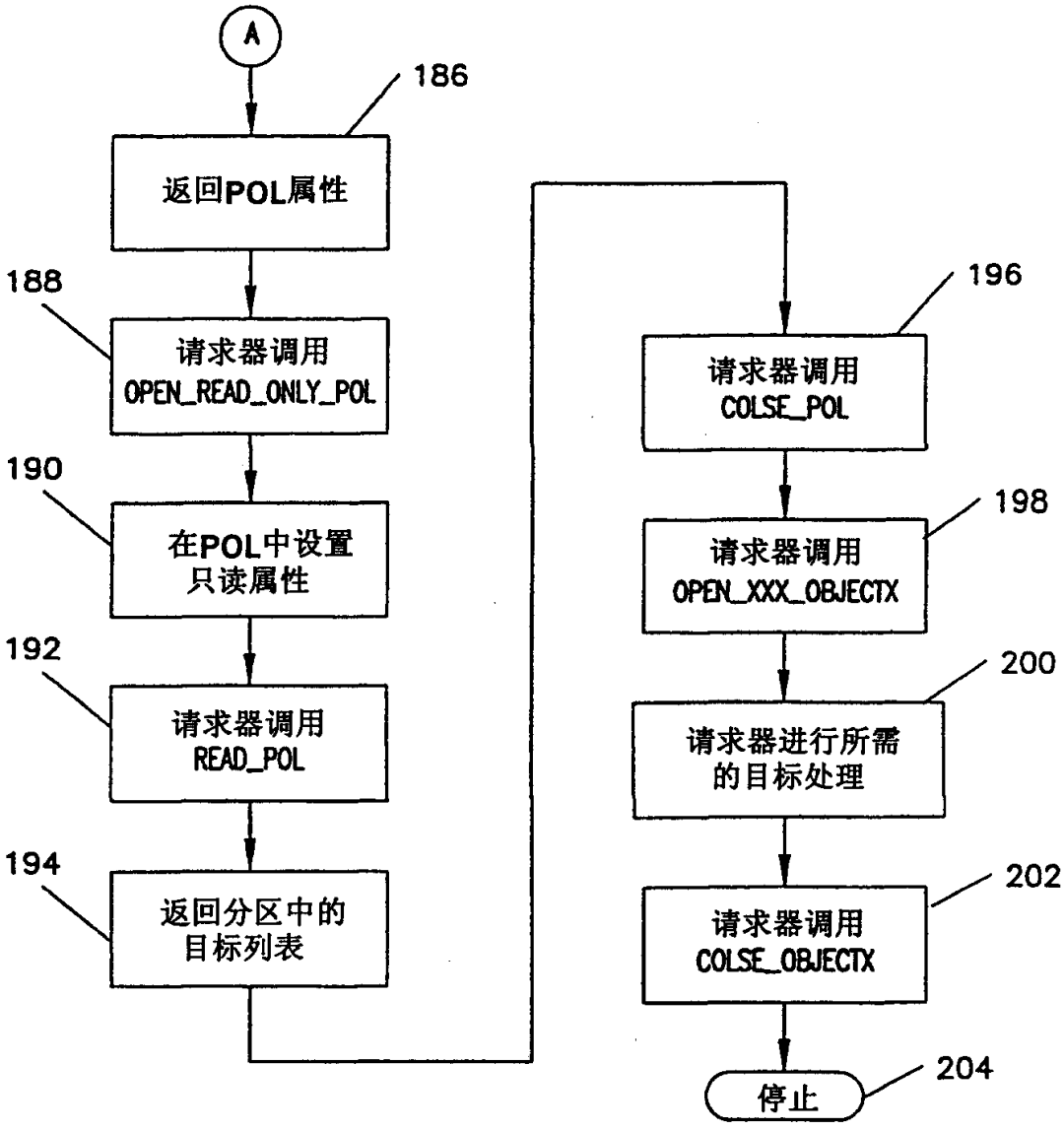

A
186
返回POL属性
188
请求器调用
OPEN_READ_ONLY_POL
190
在POL中设置
只读属性
192
请求器调用
READ_POL
194
返回分区中的
目标列表
196
请求器调用
COLSE_POL
198
请求器调用
OPEN_XXX_OBJECTX
200
请求器进行所需
的目标处理
202
请求器调用
COLSE_OBJECTX
204
停止

图 7-2

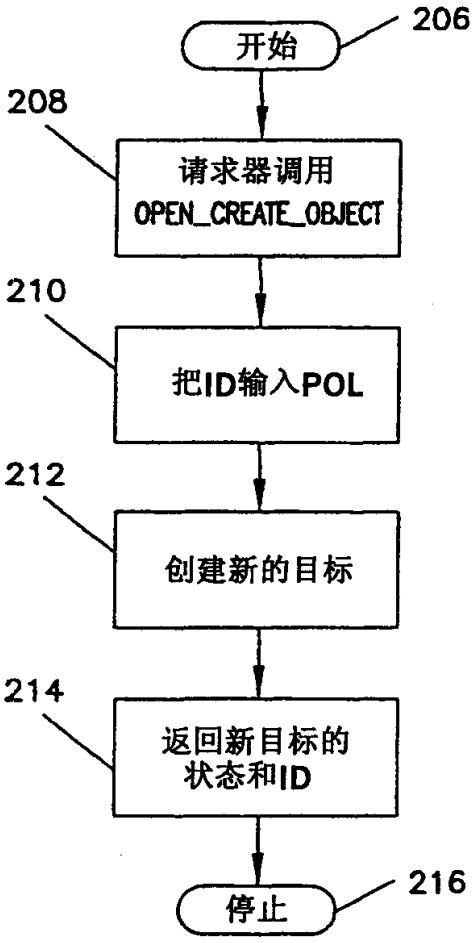

开始
206
208
请求器调用
OPEN_CREATE_OBJECT
210
把ID输入POL
212
创建新的目标
214
返回新目标的
状态和ID
停止
216

图 8

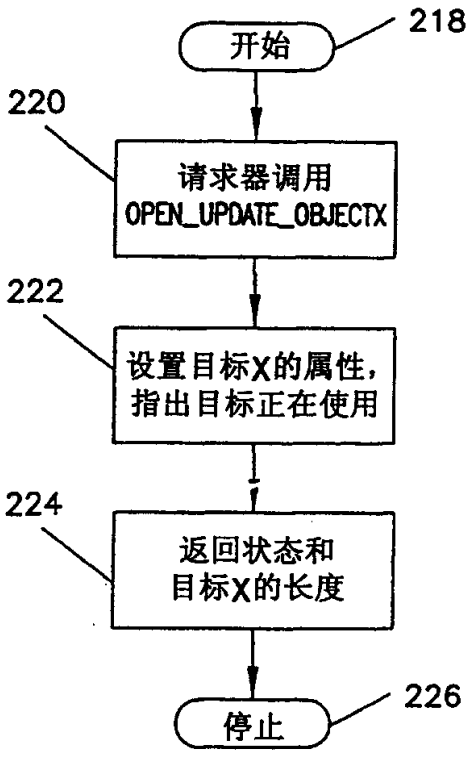

开始
218
220
请求器调用
OPEN_UPDATE_OBJECTX
222
设置目标X的属性，
指出目标正在使用
224
返回状态和
目标X的长度
停止
226

图 9

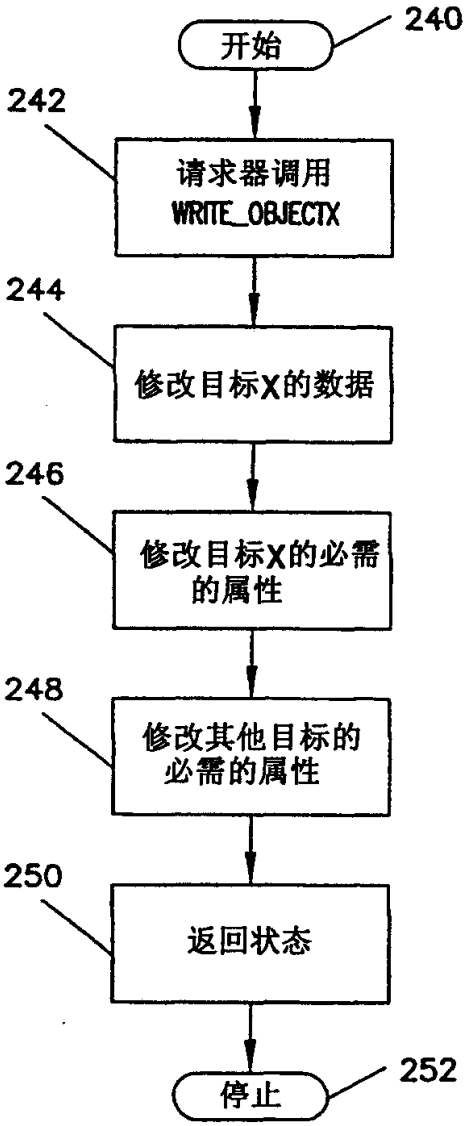

开始
240
242
请求器调用
WRITE_OBJECTX
244
修改目标X的数据
246
修改目标X的必需
的属性
248
修改其他目标的
必需的属性
250
返回状态
停止
252

图 10

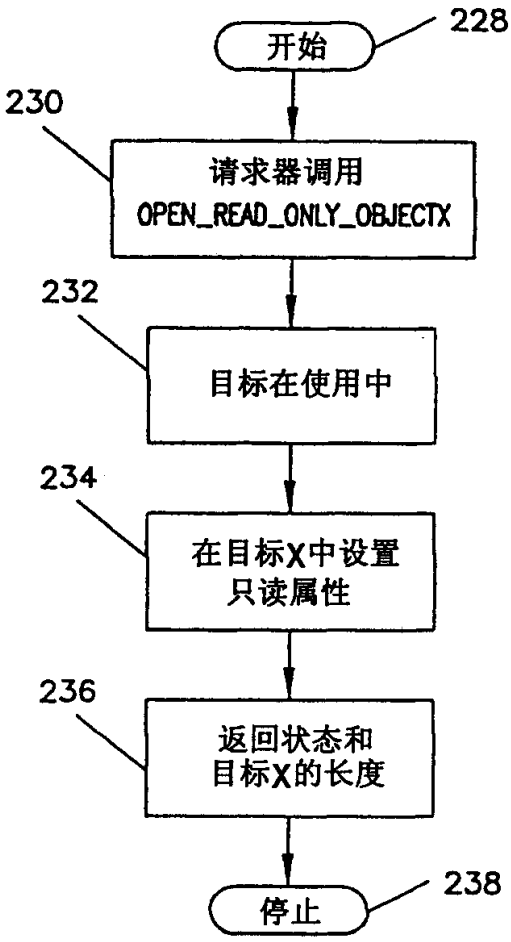

开始
228
230
请求器调用
OPEN_READ_ONLY_OBJECTX
232
目标在使用中
234
在目标X中设置
只读属性
236
返回状态和
目标X的长度
停止
238

图 11

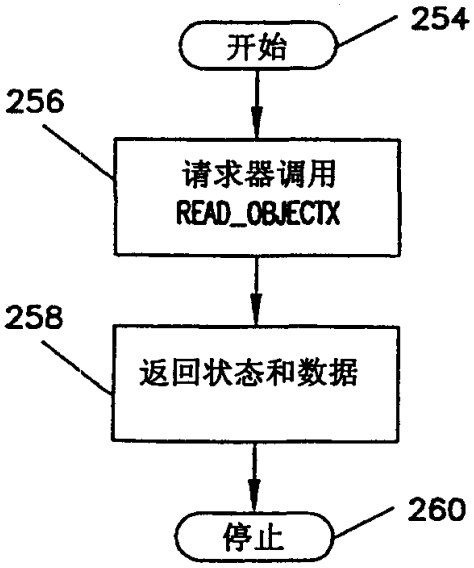

开始
254
256
请求器调用
READ_OBJECTX
258
返回状态和数据
停止
260

图 12

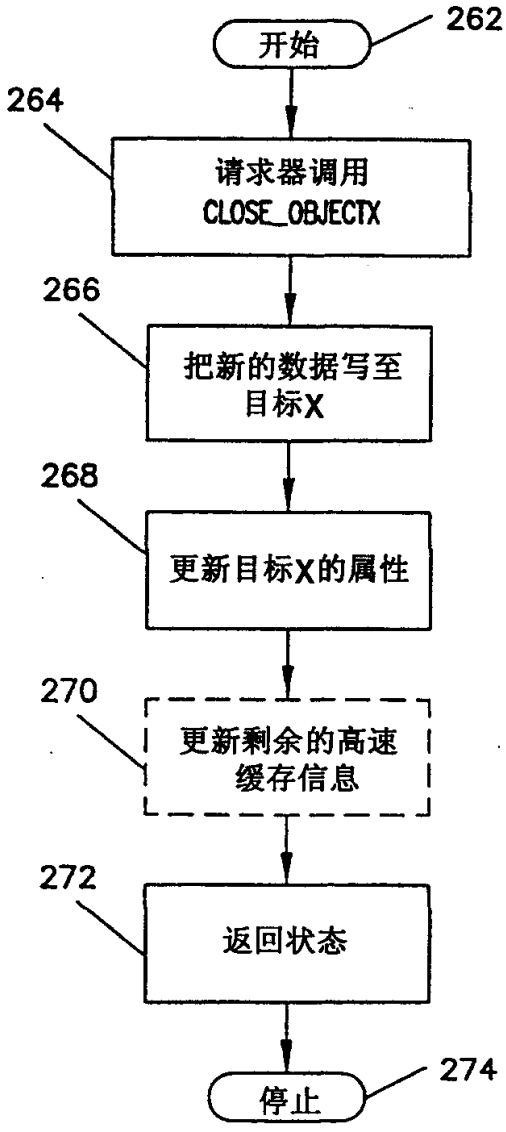

开始
262
264
请求器调用
CLOSE_OBJECTX
266
把新的数据写至
目标X
268
更新目标X的属性
270
更新剩余的高速
缓存信息
272
返回状态
停止
274

图 13

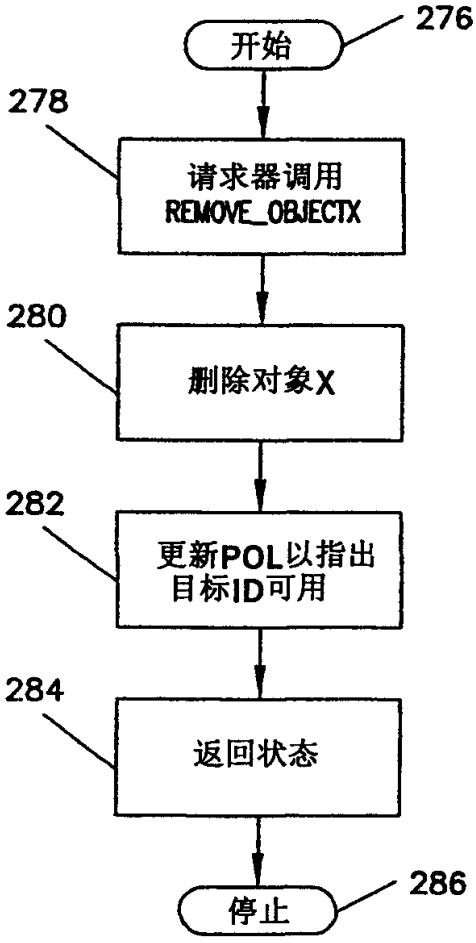

开始
276
278
请求器调用
REMOVE_OBJECTX
280
删除对象X
282
更新POL以指出
目标ID可用
284
返回状态
停止
286

图 14

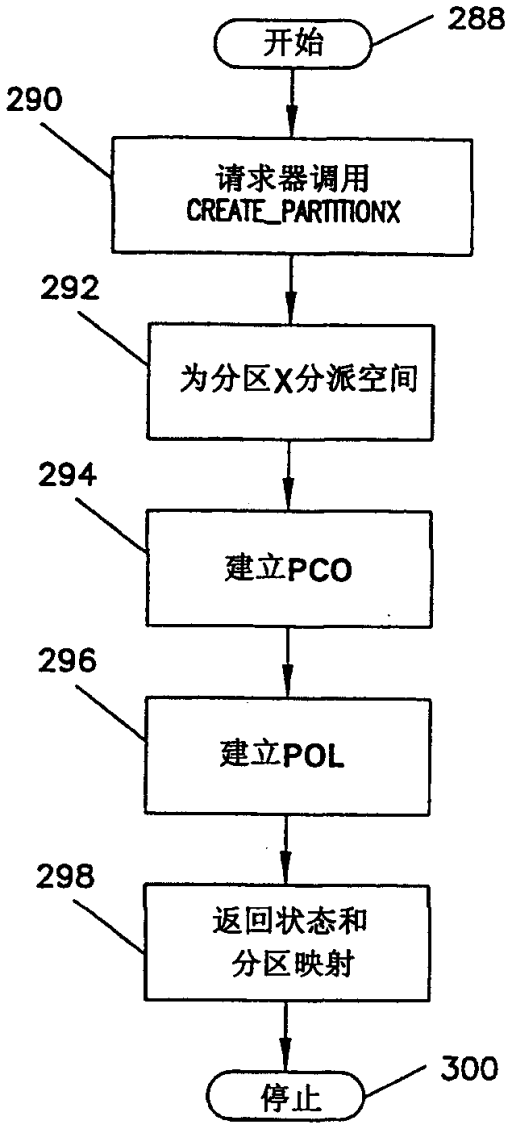

开始
288
290
请求器调用
CREATE_PARTITIONX
292
为分区X分派空间
294
建立PCO
296
建立POL
298
返回状态和
分区映射
停止
300

图 15

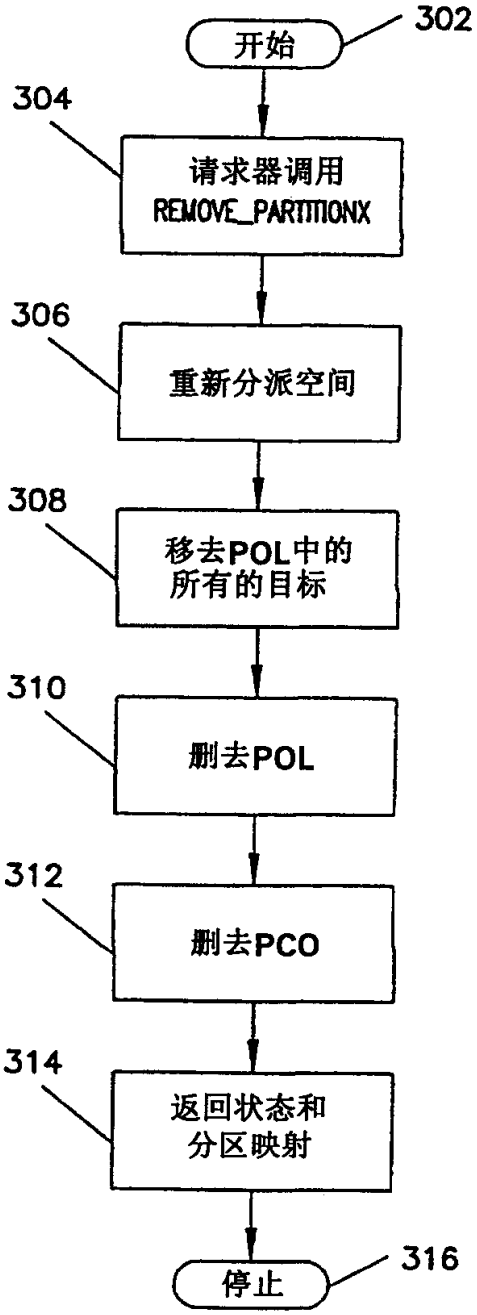

开始
302
304
请求器调用
REMOVE_PARTITIONX
306
重新分派空间
308
移去POL中的
所有的目标
310
删去POL
312
删去PCO
314
返回状态和
分区映射
停止
316

图 16

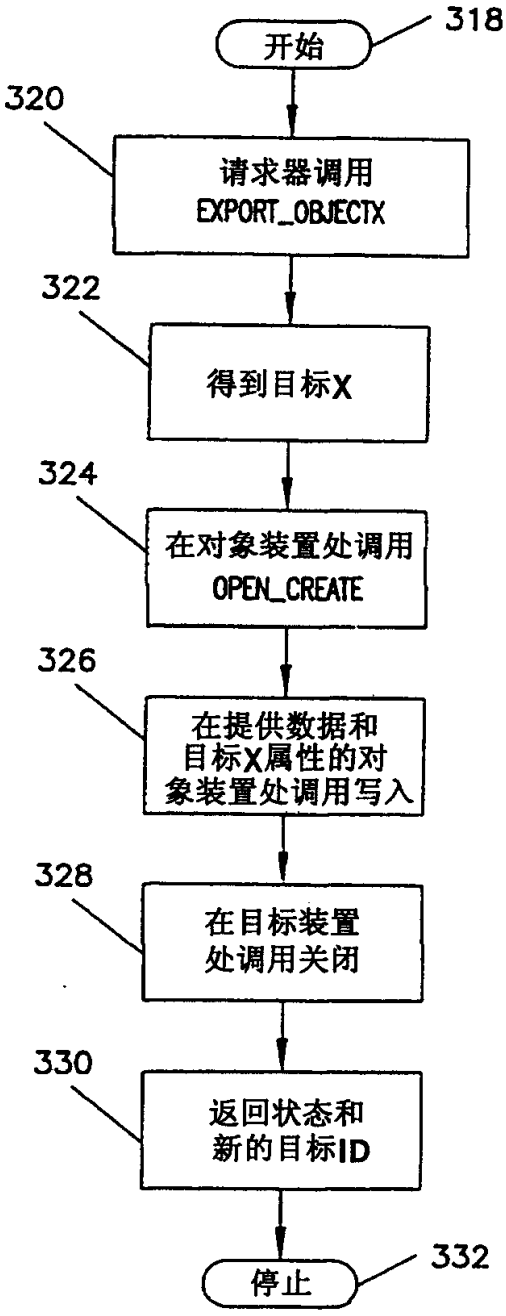

开始
318
320
请求器调用
EXPORT_OBJECTX
322
得到目标X
324
在对象装置处调用
OPEN_CREATE
326
在提供数据和
目标X属性的对
象装置处调用写入
328
在目标装置
处调用关闭
330
返回状态和
新的目标ID
停止
332

图 17

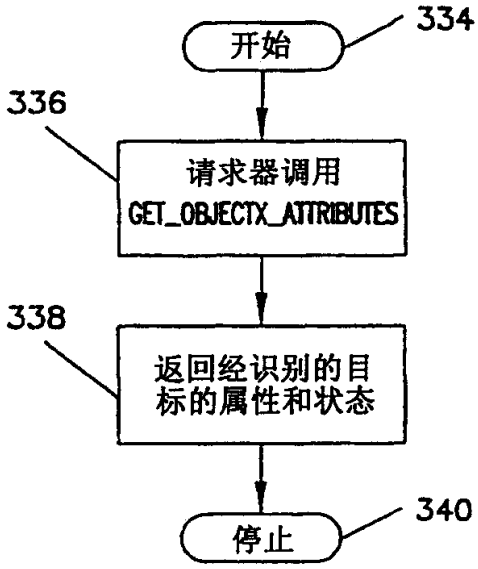

开始
334
336
请求器调用
GET_OBJECTX_ATTRIBUTES
338
返回经识别的目
标的属性和状态
停止
340

图 18

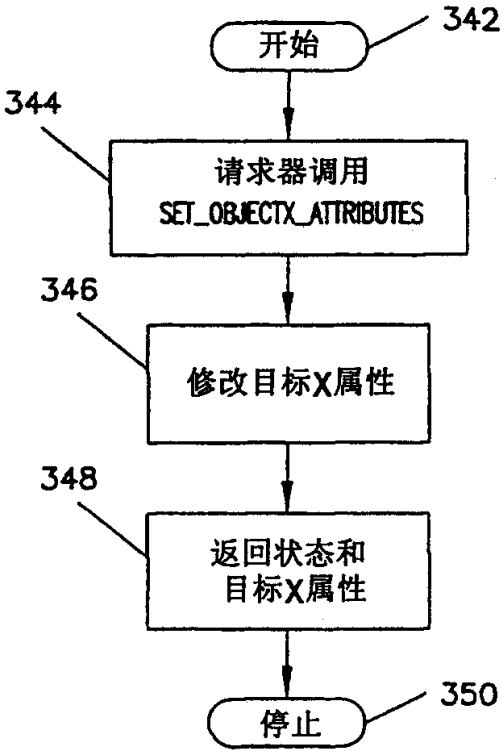

开始
342
344
请求器调用
SET_OBJECTX_ATTRIBUTES
346
修改目标X属性
348
返回状态和
目标X属性
停止
350

图 19

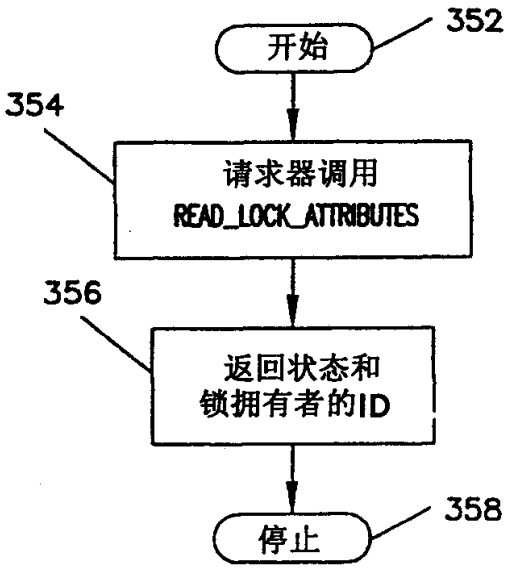

开始
352
354
请求器调用
READ_LOCK_ATTRIBUTES
356
返回状态和
锁拥有者的ID
停止
358

图 20

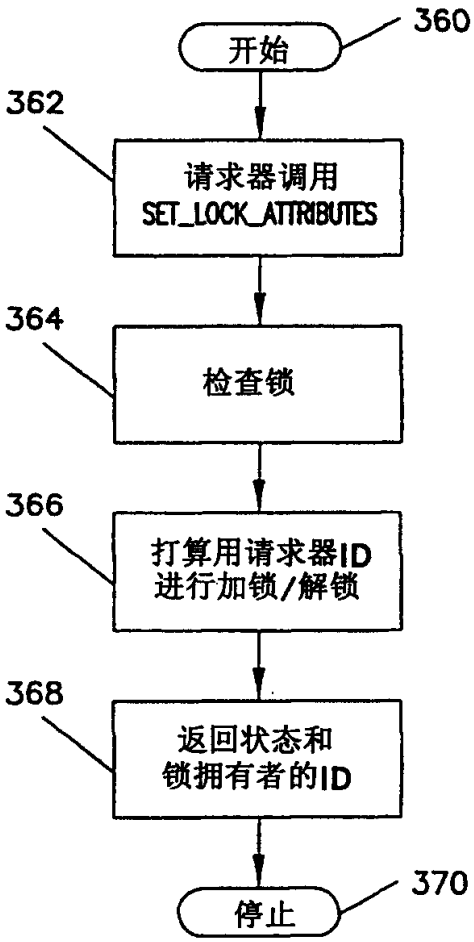

开始
360
362
请求器调用
SET_LOCK_ATTRIBUTES
364
检查锁
366
打算用请求器ID
进行加锁/解锁
368
返回状态和
锁拥有者的ID
停止
370

图 21

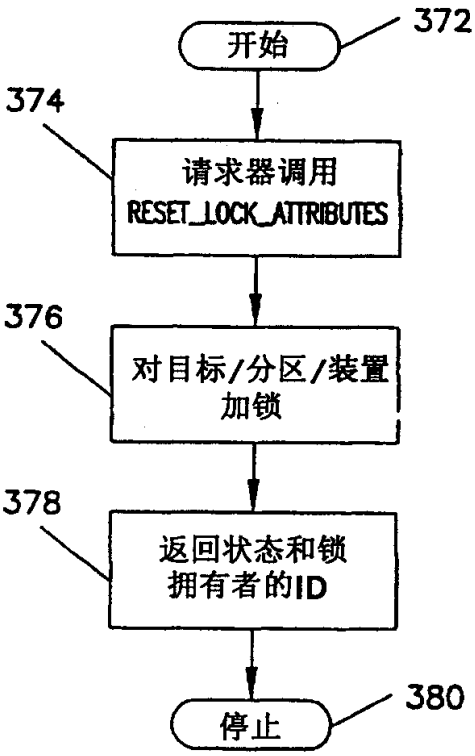

开始
372
374
请求器调用
RESET_LOCK_ATTRIBUTES
376
对目标/分区/装置
加锁
378
返回状态和锁
拥有者的ID
停止
380

图 22

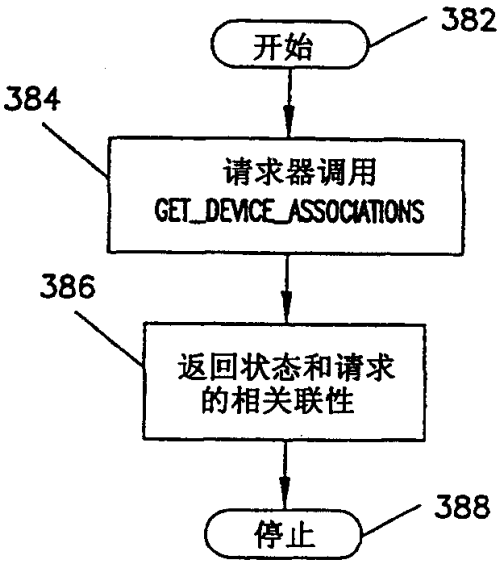

开始
382
384
请求器调用
GET_DEVICE_ASSOCIATIONS
386
返回状态和请求
的相关联性
停止
388

图 23

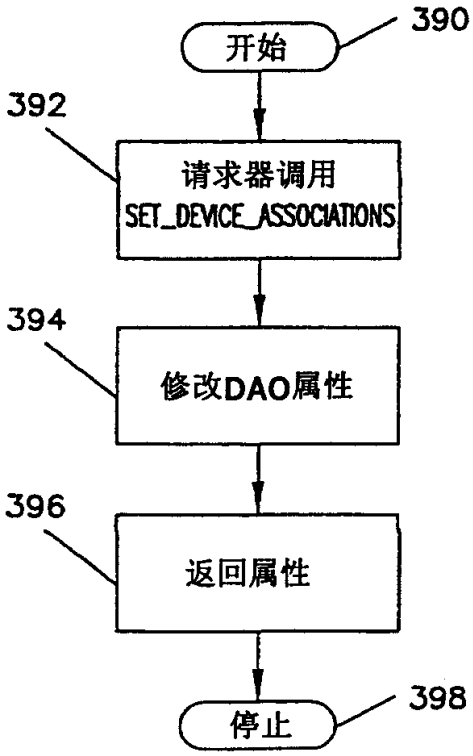

开始
390
392
请求器调用
SET_DEVICE_ASSOCIATIONS
394
修改DAO属性
396
返回属性
停止
398

图 24

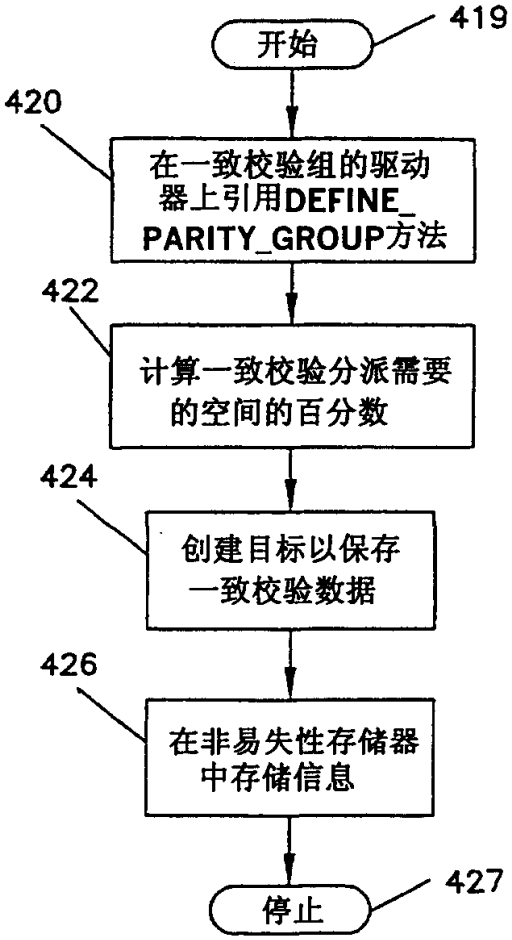

开始
419
420
在一致校验组的驱动
器上引用DEFINE_
PARITY_GROUP方法
422
计算一致校验分派需要
的空间的百分数
424
创建目标以保存
一致校验数据
426
在非易失性存储器
中存储信息
停止
427

图 28

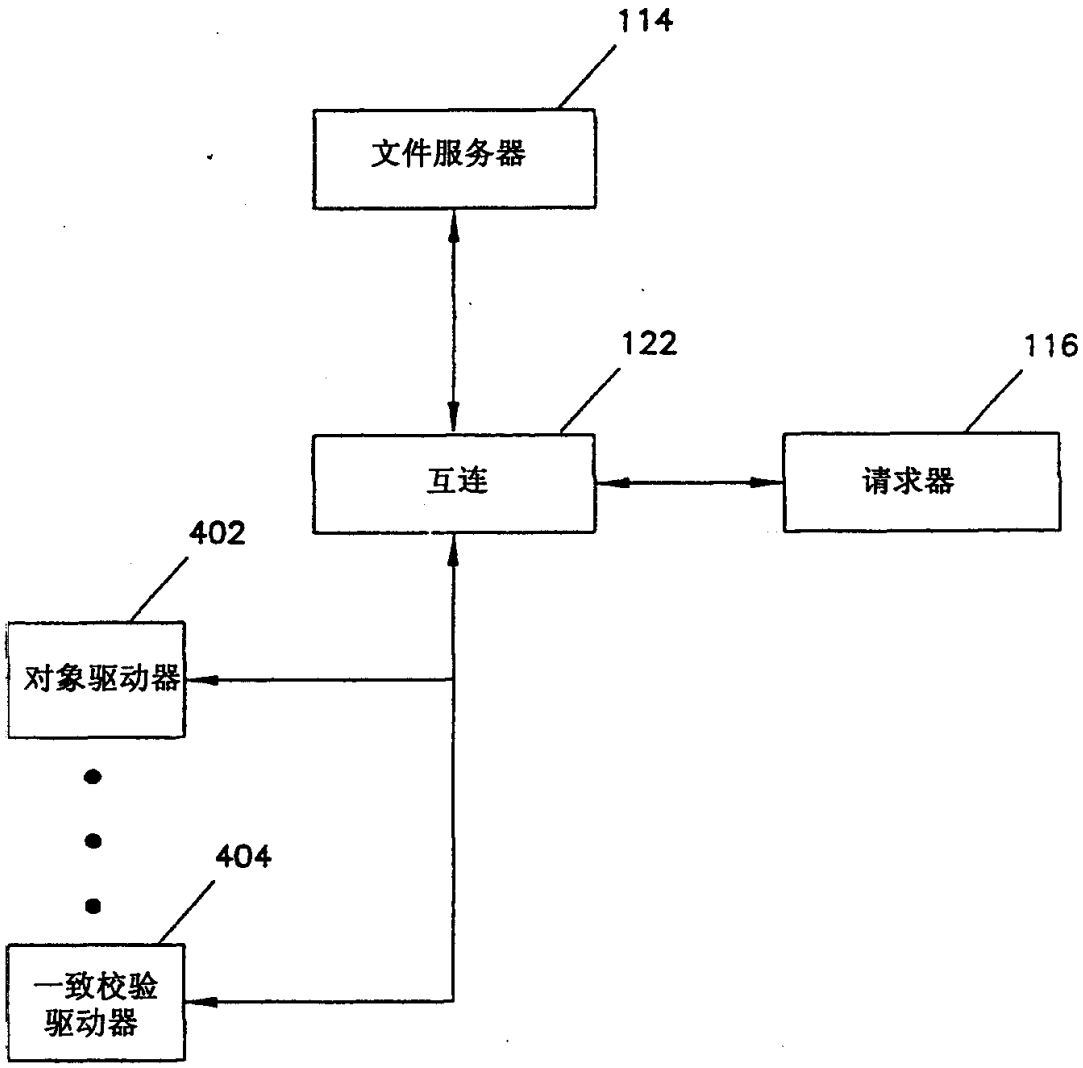


图 25

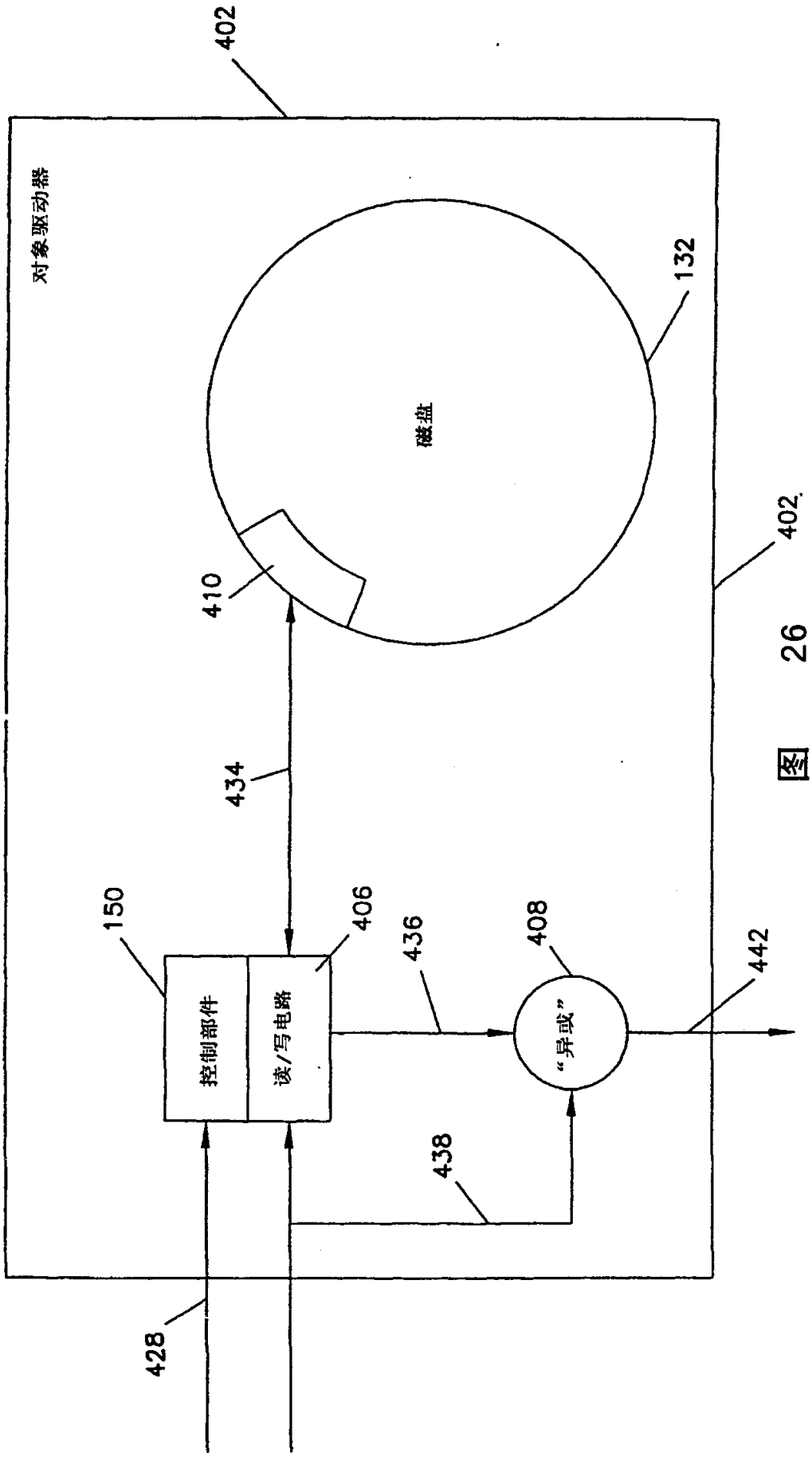

对象驱动器
402
磁盘
132
410
434
150
控制部件
读/写电路
406
436
408
"异或"
442
438
428
402

图 26

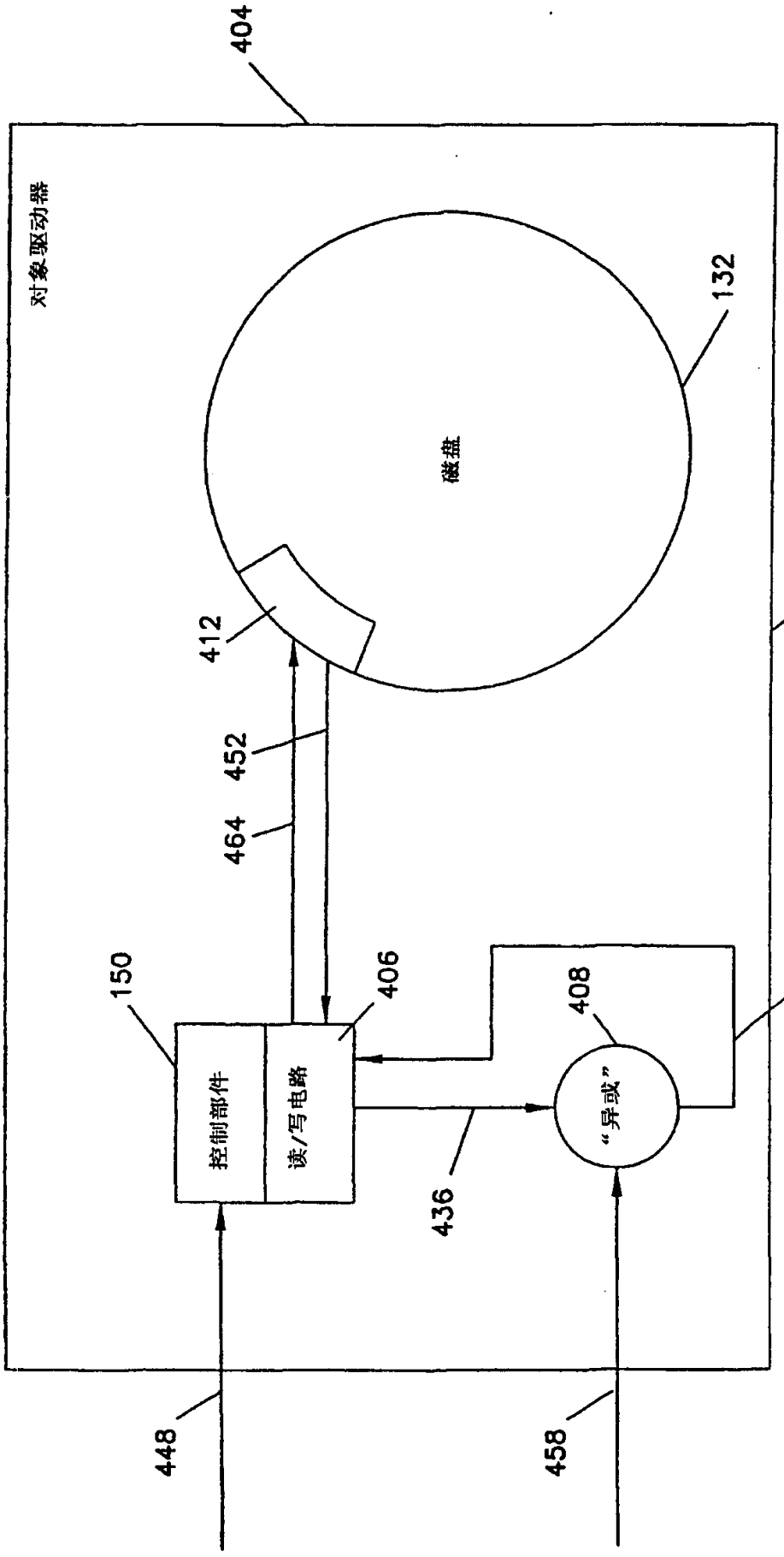

404
对象驱动器
132
磁盘
412
452
464
402
150
406
408
462
控制部件
读/写电路
"异或"
436
448
458

图 27

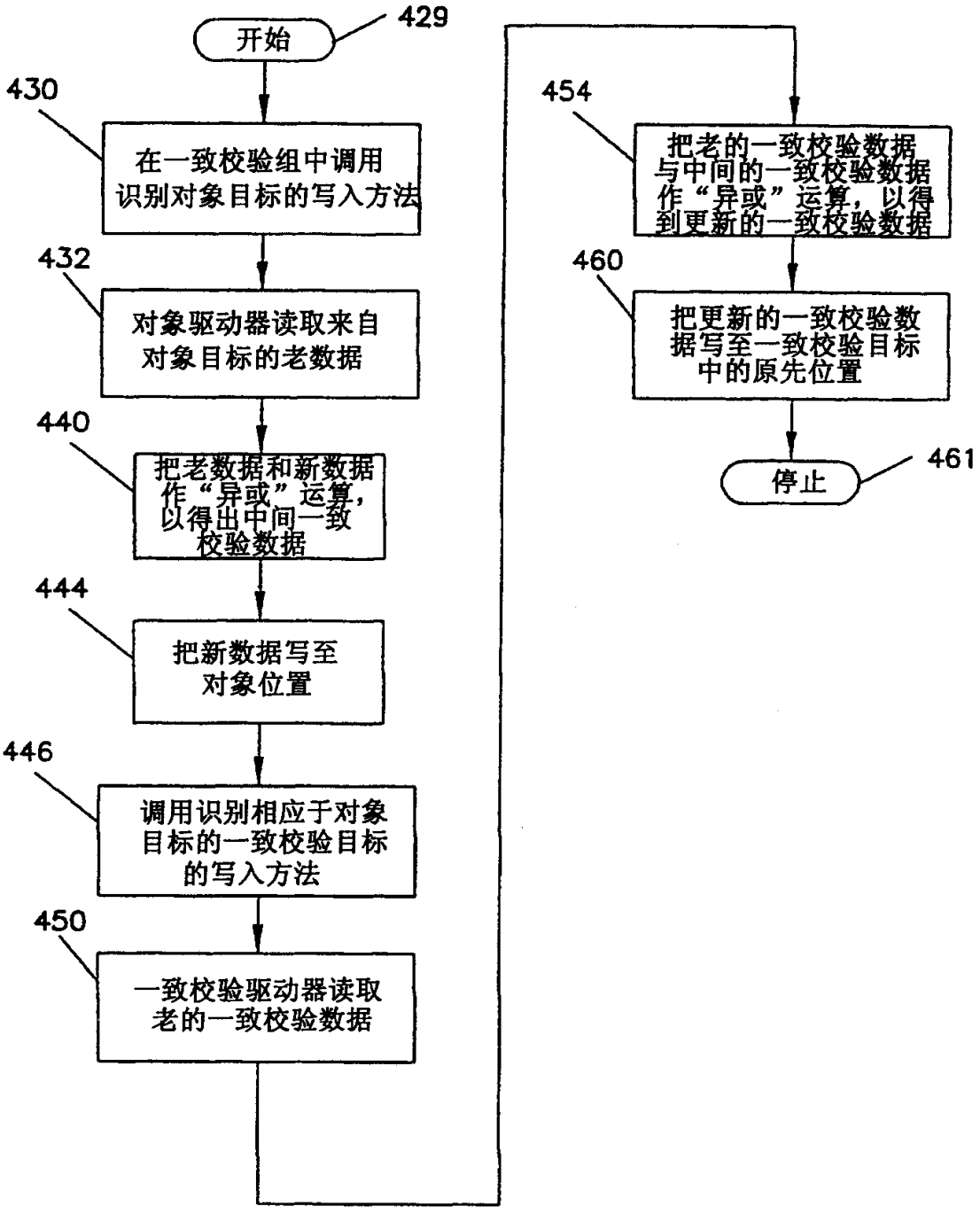

开始
429
430
在一致校验组中调用识别对象目标的写入方法
432
对象驱动器读取来自对象目标的老数据
440
把老数据和新数据作“异或”运算，以得出中间一致校验数据
444
把新数据写至对象位置
446
调用识别相应于对象目标的一致校验目标的写入方法
450
一致校验驱动器读取老的一致校验数据
454
把老的一致校验数据与中间的一致校验数据作“异或”运算，以得到更新的一致校验数据
460
把更新的一致校验数据写至一致校验目标中的原先位置
停止
461

图 29

属性	数据0	纠错码0	数据1	纠错码1	. . . . .	数据N	纠错码N

图 30

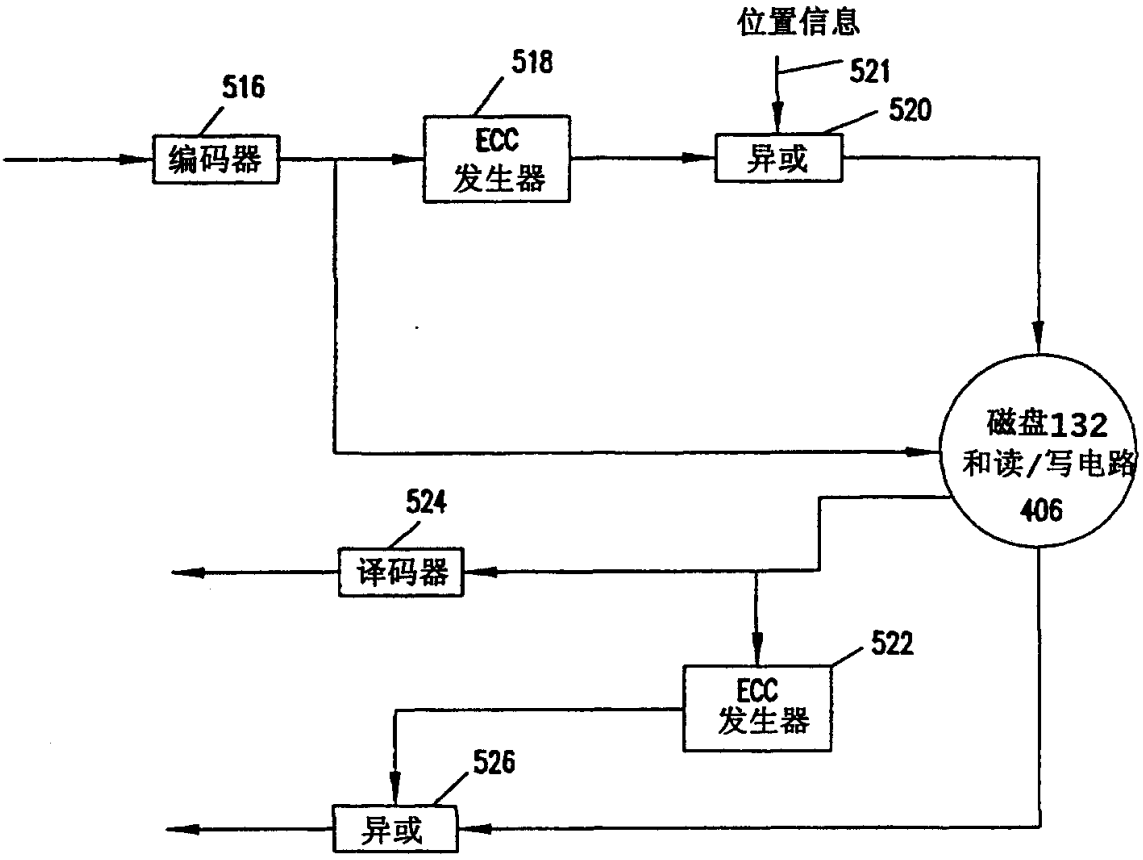


图 31

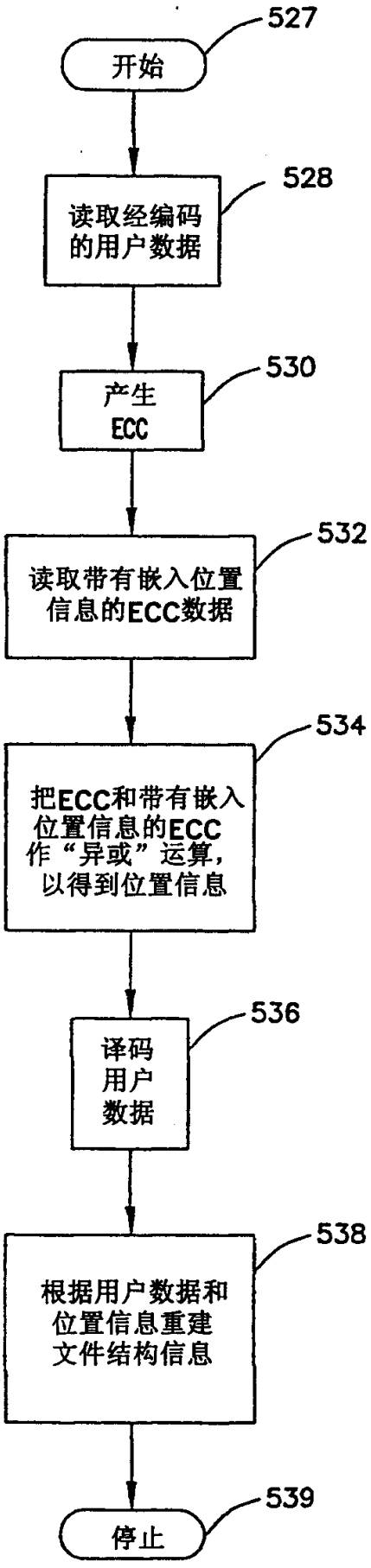

527
开始
528
读取经编码的用户数据
530
产生ECC
532
读取带有嵌入位置信息的ECC数据
534
把ECC和带有嵌入位置信息的ECC作“异或”运算，以得到位置信息
536
译码用户数据
538
根据用户数据和位置信息重建文件结构信息
539
停止

图 32

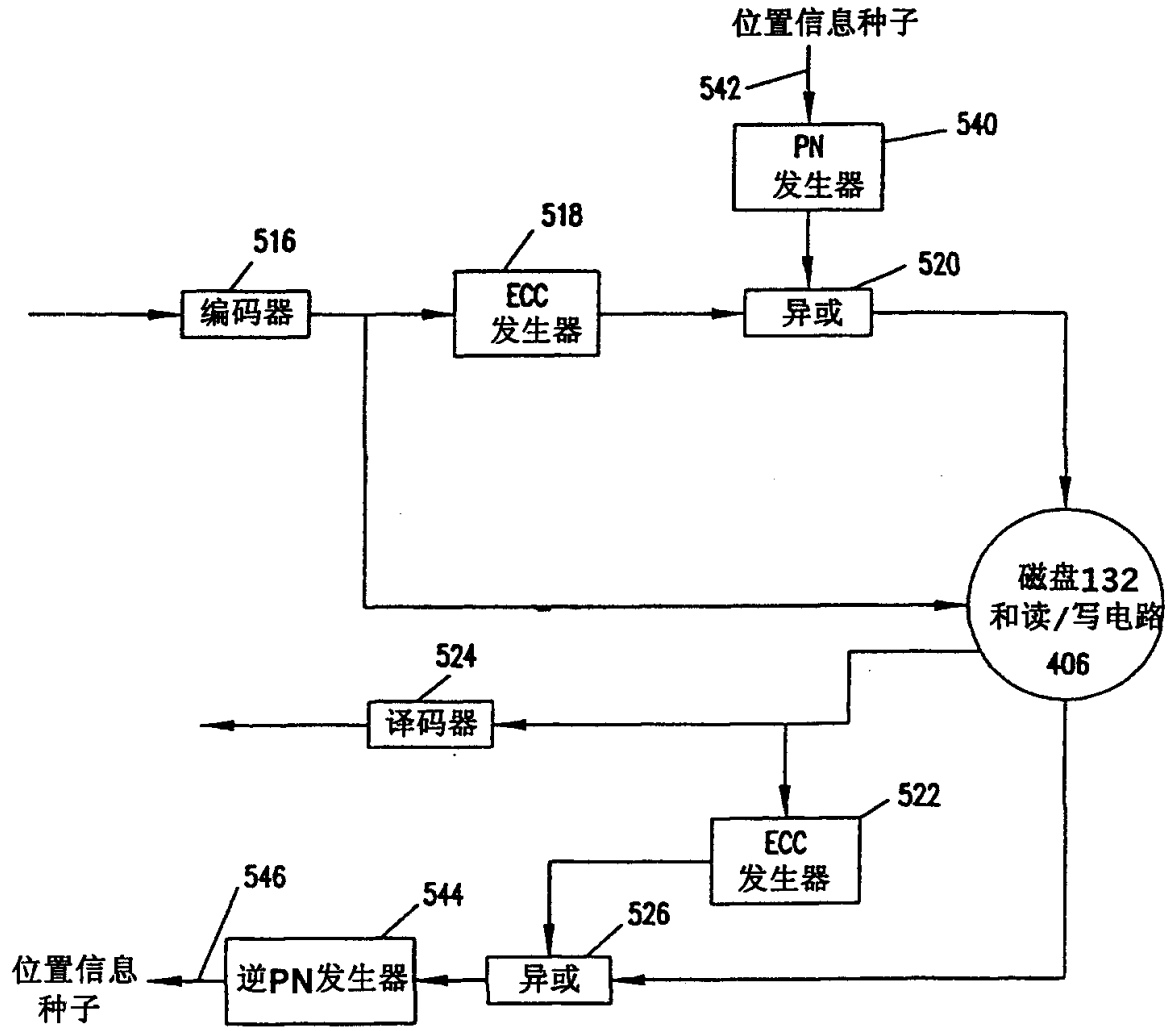
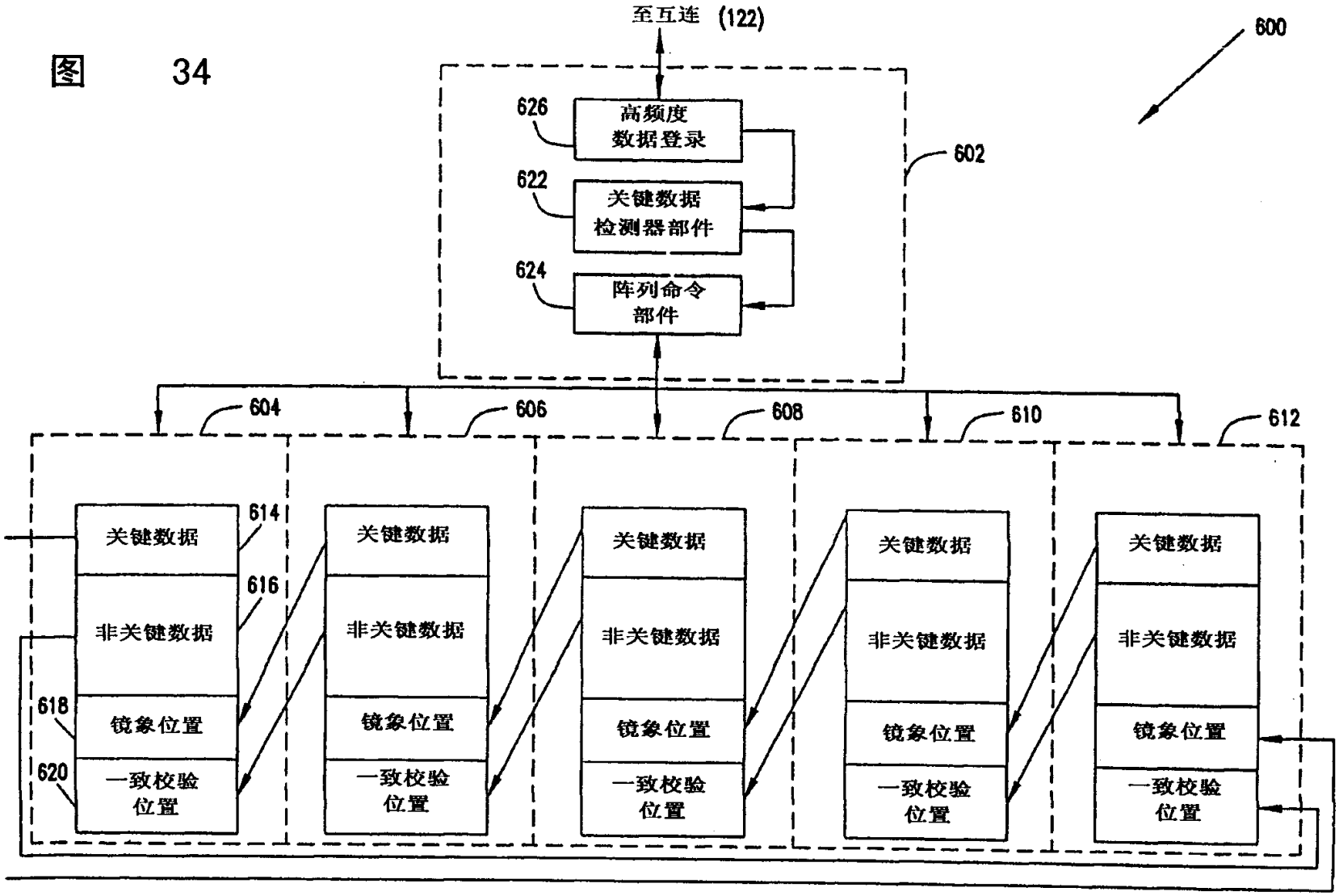

位置信息种子
542
PN
发生器
540
516
编码器
518
ECC
发生器
520
异或
磁盘132
和读/写电路
406
524
译码器
522
ECC
发生器
546
544
526
位置信息
种子
逆PN发生器
异或

图 33

图 34


至互连 (122)
600
626
高频度
数据登录
602
622
关键数据
检测器部件
624
阵列命令
部件
604
606
608
610
612
关键数据
614
非关键数据
616
镜象位置
618
一致校验
位置
620

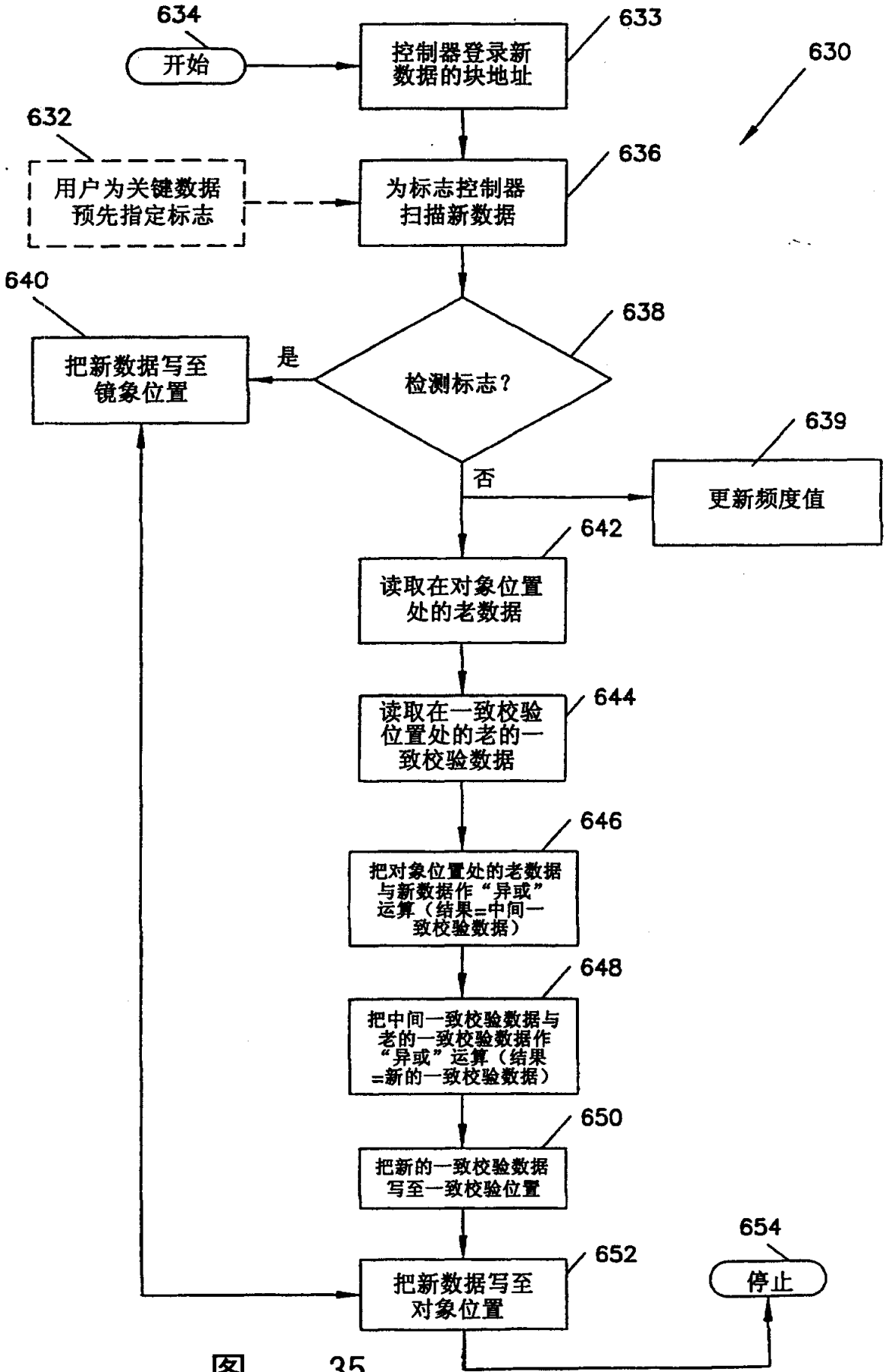

634
开始
633
控制器登录新数据的块地址
630
632
用户为关键数据预先指定标志
636
为标志控制器扫描新数据
640
把新数据写至镜象位置
是
638
检测标志？
639
否
更新频度值
642
读取在对象位置处的老数据
644
读取在一致校验位置处的老的一致校验数据
646
把对象位置处的老数据与新数据作“异或”运算（结果=中间一致校验数据）
648
把中间一致校验数据与老的一致校验数据作“异或”运算（结果=新的一致校验数据）
650
把新的一致校验数据写至一致校验位置
654
停止
652
把新数据写至对象位置

图 35

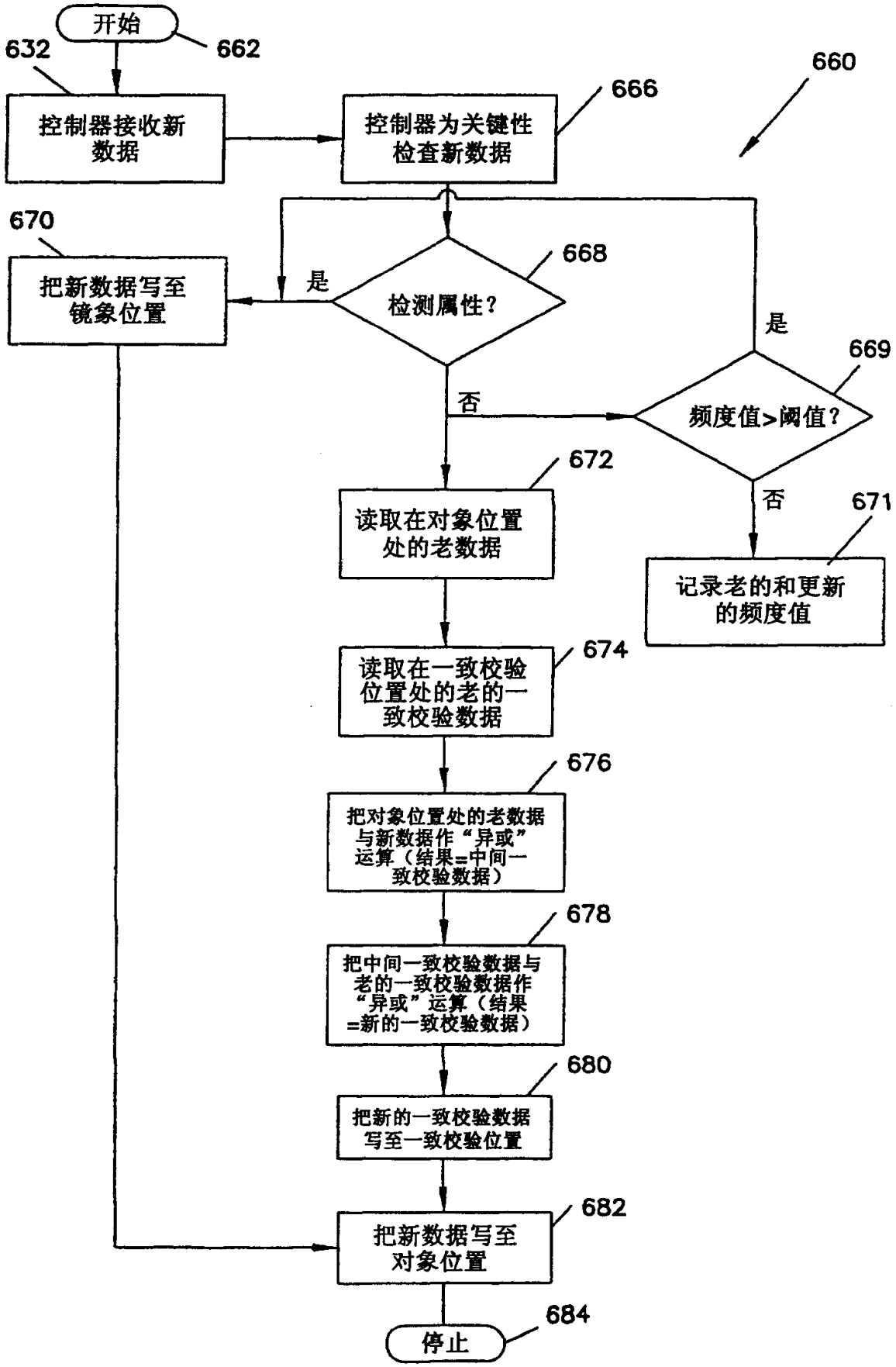

开始
662
632
控制器接收新数据
控制器为关键性检查新数据
666
660
670
把新数据写至镜象位置
是
检测属性？
668
是
669
否
频度值>阈值？
否
671
记录老的和更新的频度值
672
读取在对象位置处的老数据
674
读取在一致校验位置处的老的一致校验数据
676
把对象位置处的老数据与新数据作“异或”运算（结果=中间一致校验数据）
678
把中间一致校验数据与老的一致校验数据作“异或”运算（结果=新的一致校验数据）
680
把新的一致校验数据写至一致校验位置
682
把新数据写至对象位置
684
停止

图 36