[19]中华人民共和国国家知识产权局

[51]Int.Cl[7]
G06F 11/00
G06F 17/30

[12]发明专利申请公开说明书

[21]申请号 98808073.7

[43]公开日 2000年9月20日

[11]公开号 CN 1267379A

[22]申请日 1998.8.14 [21]申请号 98808073.7
[30]优先权
[32]1997.8.15 [33]US [31]60/055,897
[86]国际申请 PCT/US98/16956 1998.8.14
[87]国际公布 WO99/09479 英 1999.2.25
[85]进入国家阶段日期 2000.2.12
[71]申请人 西加特技术有限公司
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 沈昭坤

权利要求书2页 说明书25页 附图页数21页

[54]发明名称 面向对象的数据存储设备的冗余实现

[57]摘要

一种数据存储系统(100)包括具有存储在其上的并配置成多个对象(12 4—126)的数据的存储媒体(132),每个对象具有表示所述对象(1 24—126)特征的属性。所述对象(124—126)包含存储冗余信息 的冗余对象(412)。一个控制部件(150)可操作地耦合至存储媒体(132)并配置成以向所述对象(124—126)提供界面(128)。该 界面(128)展示可被调用以访问对象(124—126)的方法(Φ-N)。



ISSN1008-4274

知识产权出版社出版

权　利　要　求　书

1．一种数据存储系统，其特征在于，它包括：

第一存储媒体，其上存有配置成多个对象的数据，每个对象具有表示所述对象特征的属性，所述对象包括存储冗余信息的冗余对象；和

第一控制部件，可操作地耦合至所述第一存储媒体并配置成为所述对象提供界面，所述界面展示存取所述对象的调用方法。

2．一种在数据存储系统中维护冗余的方法，所述存储系统包括多个存储设备，每个存储设备包括存储媒体和可操作地耦合至所述存储媒体的控制部件，其特征在于，该方法包括下述步骤：

在所述存储媒体上存储数据，所述数据配置成多个对象，每个对象包括表示所述对象特征的属性，所述对象包括存储冗余信息的冗余对象；和

向所述对象提供展示存取所述对象的调用方法的界面。

3．如权利要求 2 所述的方法，其特征在于，该方法还包括下述步骤：对多个存储设备组调用定义冗余方法把所述组配置成实现期望冗余。

4．如权利要求 3 所述的方法，其特征在于，该方法还包括下述步骤：响应所述调用步骤，在所述组的每个存储设备上创建冗余对象。

5．如权利要求 4 所述的方法，其特征在于，所述创建步骤包括：计算分配给所述冗余对象的部分存储空间。

6．如权利要求 3 所述的方法，其特征在于，该方法还包括下述步骤：对所述多个存储设备中的第一个设备调用写方法，使所述第一存储设备上的第一控制部件更新所述第一存储设备的第一存储媒体上的指定数据对象。

7．如权利要求 6 所述的方法，其特征在于，该方法还包括下述步骤：响应所述写方法，对所述多个存储设备中的第二个设备调用奇偶校验写方法，更新存储在所述第二存储设备上的第二存储媒体中，与所述指定数据对象关联的奇偶校验对象。

8．如权利要求 7 所述的方法，其特征在于，所述调用写方法的步骤进一步包括下述步骤：从所述指定数据对象读老数据；及向所述数据对象写新数据。

9．如权利要求 7 所述的方法，其特征在于，所述调用冗余写方法的步

骤包括下述步骤：

对所述老数据和新数据执行异或操作以获得中间冗余数据；和

对提供中间冗余数据和冗余对象标识符(ID)的所述第 2 存储设备调用冗余写方法。

10．如权利要求 9 所述的方法，其特征在于，该方法响应于调用冗余写方法，还包括下述步骤：

从所述冗余对象读老冗余数据；

对所述老冗余数据与中间冗余数据作异或操作以获得新冗余数据；

向所述冗余对象写所述新冗余数据。

11．如权利要求 9 所述的方法，其特征在于，该方法还包括下述步骤：对包含所述冗余对象的存储设备组进行计算，所述冗余对象与所述指定数据对象关联，所述计算基于所述组的存储设备数和用新数据写的所述指定数据对象内的位置。

12．如权利要求 11 所述的方法，其特征在于，该方法还包括下述步骤：根据所述组内存储设备数及在用所述新数据写入的指定数据对象中的位置，计算被更新的所述冗余对象中的位置。

13．一种数据存储系统，其特征在于，它包括：

第一盘驱动器，它包含：存有配置成多个对象数据的的第一存储媒体，每个对象具有表示所述对象特征的属性，所述对象包括存储冗余信息的第一冗余对象；第二盘驱动器，它包括可操作地耦合至所述第一存储媒体并配置成向所述对象提供界面的第一控制部件，所述界面展示存取所述对象的调用方法；

第二盘驱动器，可操作地耦合至所述第一盘驱动器和具有存有配置成多个对象的数据的媒体，每个对象具有表示所述对象特征的属性，所述对象包括存有与所述第一冗余对象中的所述冗余信息对应数据的第一数据对象；和第二控制元件，可操作地耦合至第二存储媒体并配置成向所述对象提供界面，所述界面展示存取所述对象的调用方法。

说　　明　　书

面向对象的数据存储设备的冗余实现

技术领域

本发明涉及数据存储设备。具体而言，本发明涉及其中数据作为对象加以组织和存取的诸如磁盘驱动器、磁带驱动器或光盘驱动器之类的数据存储设备。

背景技术

在计算机产业界，熟知两类常规的计算机模型。一种是主机计算模型，另一种是簇（群集）计算模型。

在主机计算模型中，最终用户的通常过程是，先购买一初始系统，而在需要追加处理能力时，再换一个更大的系统。在这周期的不同点会产生损伤性的不连续。例如，如果用户成长到不满意初始系统的体系结构，则用户可能需从一种操作系统转换至另一种操作系统，或甚至当购买第二套升级的主机系统时，要从一个供应商专有的体系结构转换到另一个供应商的专有体系结构。为实施升级，这种改变在费用与雇员时间两方面均需巨大代价。所以在很多情况下，要避免这种转换。

此外，主机模型遗留的计算机设备的剩余价值较差。这样，当初始系统由升级的系统更换时，系统更换经常使以前投资基本上完全丧失。何况，更大的升级系统与较小系统相比，销售量较小。因此，升级系统的价格常比以前系统要高。

在簇计算模型中，用一簇较小的、标准服务器来代替主机。这可提供比主机更多优点。因为簇可作为一个单一系统启动，因而进入簇模型的阈值较低。进而，这种较小系统常常大量销售，使计算机成本较低。而且，这种系统的标准是不依赖于专有体系结构。这为现有设备提供了多种来源的可能性，从而允许用户续后购买时选择到最佳产品。

簇计算模型还具有以下优点。通过仅增加满足目前及近期需求的附加资源，可更精确地控制升级成本。进而，用户可从广泛的供应商中进行选择而

不必担心转移或转换至一个升的体系结构。类似地，有一个正确的体系结构，可能决不需转换至另一操作系统。

簇计算模型也有其不利的问题。例如，簇计算模型难于提供在承担与单个主机完成同等工作量前提下而具有共享数据能力的簇系统。例如，通常非常难于实现簇中每个服务器都处理相同数据事务的簇机型。

一些这种应用的例子包括航空公司订票系统或金融机构的完整交易清单。

簇机型的第二个缺点仅涉及在管理主机环境中现存数据缺少广泛经验。这种经验已引伸至在标准的簇环境中还没可用的管理软件。

本发明致力于解决这些和其它问题，并对已有技术提供另一些优点。

发明内容

数据存储系统包括存储媒体，其上存储配置成多个对象的数据，每个对象具有表示该对象特征的属性。这些对象中还包括一个存储冗余信息的冗余对象。控制部件可操作地耦合至该存储媒体并配置成为所述对象提供一个界面。所述界面展示存取对象的调用方法。

附图概述

图 1 是根据本发明一个方面的附加于网络的存储系统框图。

图 2 是根据本发明一个方面的对象模型的说明图。

图 3－1 是第一种配置框图，其中一个请求者存取存储设备中的一个对象。

图 3－2 是第二种配置框图，其中一个请求者存取存储设备中的一个对象。

图 4 是根据本发明一个方面的磁盘驱动器的透视图。

图 5 是说明请求者访问对象的功能框图。

图 6 是根据本发明一个方面分区的部分存储媒体的说明图。

图 7－1 和图 7－2 是根据本发明的一个方面，请求者存取对象的流程图。

图 8 是根据本发明一个方面，创建对象的流程图。

图 9 是根据本发明一个方面，打开和更新对象的流程图。

图 10 是根据本发明一个方面，写对象的流程图。

图 11 是根据本发明一个方面，打开和只读对象的流程图。

图 12 是根据本发明一个方面，读对象的流程图。

图 13 是根据本发明一个方面，关闭对象的流程图。

图 14 是根据本发明一个方面，删除对象的流程图。

图 15 是根据本发明一个方面，创建分区的流程图。

图 16 是根据本发明一个方面，删除分区的流程图。

图 17 是根据本发明一个方面，输出对象的流程图。

图 18 是根据本发明一个方面，获得对象属性的流程图。

图 19 是根据本发明一个方面，设置或修改对象属性的流程图。

图 20 是根据本发明一个方面，读锁定属性的流程图。

图 21 是根据本发明一个方面，设置锁定属性的流程图。

图 22 是根据本发明一个方面，重置锁定属性的流程图。

图 23 是根据本发明一个方面，获取设备关联的流程图。

图 24 是根据本发明一个方面，设置设备关联的流程图。

图 25 是根据本发明一个方面，实现磁盘驱动阵列的框图。

图 26 是根据本发明一个方面的目标磁盘驱动器的框图。

图 27 是根据本发明一个方面的奇偶磁盘驱动器的框图。

图 28 是根据本发明一个方面，创建奇偶群的流程图。

图 29 是根据本发明一个方面，更新奇偶信息的写操作流程图。

较佳实施例的详细说明

图 1 是根据本发明一个方面的数据存储系统 100 的框图。系统 100 包括面向对象的数据存储设备 110 和 112，文件服务器 114，请求者 116、118 和 112，及互连 122。系统 100 表示一种附加于网络的存储配置，它由来自许多不同供应商的设备和软件构件，对用户来说如同单个大计算机系统。

面向对象的存储设备 110－112 是执行系统 100 的数据存储功能的存储部件。存储设备 110－112 最好包括磁盘驱动器、独立冗余盘阵列（RAID）子系统、磁带驱动器、磁带库、光盘驱动器、连接光盘(juke boxes)或其它可共享的存储设备。存储设备 110 和 112 还向请求者 116、118 和 120 提供输入/输出通道以便存取设备 110 和 112。

请求者 116、118 和 120 是类似服务器或客户器之类的部件，它们能共

享设备 110 和 112 中的存储信息。请求者 116－120 最好配置成直接存取存储设备 110 和 112 中的信息。

文件服务器 114 执行管理和安全功能，诸如请求验证和资源位置等。在较小系统中，最好不使用专用文件服务器。而用 116－120 中的一请求者承担文件服务器 114 本该执行的监督系统 100 操作的职责。此外，在不需要或不希望文件服务器 114 提供的安全和功能时，或更需请求者 116－120 群与存储设备 110 和 112 直接交谈时，可从系统 100 中去掉文件服务器 114。

在一个较佳实施例中，互连 122 是物理上的基础结构，通过它，带有存储系统 100 的网络中的所有部件能相互通信。

操作上，当系统 100 启动时，所有设备最好相互或相对于一个共同参照点（例如文件服务器 114 或互连 122）标识自身。例如，在基于光纤通道的系统 100 中，面向对象的存储设备 110 和 112 及请求者 116－120 登记在系统组织结构(fabric)。在这样实施后，任何希望确定操作配置的系统 110 的任何部件可使用结构服务来识别所有其它部件。从文件服务器 114，请求者 116－120 得知请求者 116－120 可以存取的存储设备 110 和 112 的存在。同样，存储设备 110 和 112 得知确定系统 100 中的其它设备的位置信息及用于调用备份之类管理服务所必须的地址。类似地，在一个较佳实施例中，文件服务器 114 从结构服务得知存储设备 110 和 112 的存在。

取决于特定系统 100 的安全实践，请求者 116－120 或其中任一个可能被拒绝存取系统 100 的某些部件。从对每个请求者都适用的一组存储设备 110 和 112，请求者可识别文件、数据库和它可用的空闲空间。

同时，系统 100 中的各部件最好能识别与文件服务器 114 关联的任一特写考虑。例如，一旦任一存储设备将层次服务(level service)属性通知文件服务器，则系统 100 中的所有其它部件即可从文件服务器 114 得知这些属性。例如，特定请求者 116－120 可能希望及时通知它附加存储设备的介入，并启动之。当例如只要请求者在向文件服务器 114 登录时声明，即可提供这种属性。这样，每当有新存储设备加入系统 100 时，文件服务器 114 自动告知该特定请求者 116－120。通常，文件服务器 114 还可向请求者传送其它重要特性，例如存储设备是否 RAID5、镜象等。

根据本发明的一个方面，存储在存储设备 110 和 112 中的信息是以图 2 中清楚说明的系统加以存储。存储设备 110 和 112 中的每一个最好是面向对

象的设备，在它的操作模式中，数据是作为对象 124－126，而不是作为一定序列的扇区来组织和存取的。面向对象的设备 110 和 112 用一个对象文件系统来管理对象 124－126，该系统包括对特定设备上的每个分区，有一个单层对象表。这也可看作为平直文件系统。存储在每个设备 110 和 112 的存储媒体上的对象 124－126 最好是以面向对象设备模式操作的设备 110 或 112 上的容量分配的最小可视单元。在这种存储设备上的一个对象包括与唯一标识符关联的一组有序扇区。数据用标识符参照且对象有一偏移量。按存储设备 110 或 112，将对象分配，存入存储媒体。操作系统管理由对象组成的文件和元数据，而不像已有体系结构中管理扇区数据。

通过界面 128 存取对象 124－126，其中对象展示多个可被请求者 116－120 调用的方法，以存取和操作对象 124－126 中的属性和数据。这样，如图 2 所示，请求者 116－120 发出请求 130。在较佳实施例中，请求者 116－120 是计算机系统，或是某簇系统网络中的一员，它们对包含对象 124－126 的存储设备发出请求 130。从而，请求者 116－120 既可是客户，又可以是服务器。在任何情况下，由请求者 116－120 之一发出的请求 130 调用界面 128 中的一个方法，从而操作一个或多个对象 124－126，如本申请中后文将详细说明的。

图 3－1 和 3－2 是可用于存取存储设备 110－112 中所存对象的二个不同配置的框图。为简单起见，在图 3－1 和 3－2 中仅说明单个请求者 116 和单个面向对象的存储设备 110。当请求者 116 希望打开对象（例如对象 124－126）时，请求者 116 可直接存取存储设备 110，或为存取存储设备 110 上的一个对象，它可能需要文件服务器 114 的请求许可和位置信息。文件服务器 114 控制对存储设备 110 的存取扩展成系统 100 特定执行的安全要求功能。

在图 3－1 说明的框图中，假定系统 100 是安全的。即不要求保护在请求者 116 和存储设备 110 间进行命令信息和数据的传送。在这种实施中，仍有一带管理功能的文件服务器 114，但不需它监督请求者与存储设备 110 的交互作用。

在这种实施中，请求者 116 可直接对存储设备 110 存取和创建对象。这样，请求者 116 可以打开、读、写和关闭对象，就象它们原本就附加于请求者 116 一样。该操作后文将详细叙述。但为清楚起见，这里先作一概述。为

了读存储设备 110 中的对象，请求者 116 最好先读揭示存储设备 110 的逻辑卷或分区的一个或多个对象并弄懂如何开始检索其上存储的对象。请求者 116 然后打开并读一个对象，它可能是根目录。从该对象，且基于根目录的内容一直向前寻找其它对象。请求者 116 重复上述过程直到找到所要的数据。数据由对象标识符（对象 ID）参照，并在对象内有偏移。

在示于图 3－2 的第二个实施例中，要求安全性。所以文件服务器 114 插入请求者 116 和存储设备 110 间的 I/O 链至期望保护级所必需的程度。在一个较佳实施例中，请求者 116 必须首先执行一系列 I/O 操作得到文件服务器 114 的请求许可。文件服务器 114（它可能为了附加安全而向请求者 116 隐瞒存储位置信息）然后通过返回足够信息以允许请求者 116 直接与存储设备 110 通信而认可来自请求者 116 的请求。由于当存储设备 110 在文件服务器 114 上登录时，它最好被告知安全参数，因而存储设备 110 最好不允许 I/O 请求，除非它被适当地构成并包括含来自文件服务器 114 的有效许可的编码数据。

然后，过程以与图 3－1 所叙述过的相似方式进行。但是，与每个命令关联的有效负载可能很不相同。例如，在要求完全的情况下（如图 3－2 所示），在请求者 116 和存储设备 110 间传送的命令和数据均可能加密。此外，最好将许可信息追加至请求者 116 向存储设备 110 提供的命令参数。

由于在一个较佳实施例中，存储设备 110 和 112 可包括硬盘驱动器，因而简单讨论硬盘驱动器是必要的。图 4 是可用作存储设备 110 的硬盘驱动器的透视图。在磁盘驱动器 110 中，多个盘片 132 围绕机壳 136 中的电机部件 134 的轴套叠，每个盘片 132 有许多同心圆的记录磁道，如 138 所示。每个磁道 138 再分成多个分区（参照图 6 细述）。通过指定某个磁道 138 内的分区，数据可存储在磁盘 132 上或从中检索到。致动臂构件 140 最好可旋转地安装在机壳 136 的一个角上，致动臂构件 140 带有多个头万向架构件 142，每个构件带有具有读/写头即传感器（未图示）的滑块，用于从磁盘 132 读出信息或把信息写入磁盘 132。

音圈电机 144 精确地前后旋转致动臂构件 140，使滑块 142 上的传感器在磁盘 132 的表面沿箭头 146 所指的弧形移动。图 4 还以框图形式表示磁盘驱动控制器 148，用于以熟知的方式控制磁盘驱动器 110 的某些操作。但是，根据本发明，磁盘驱动器控制器 148 也用于对磁盘 132 上存储的对象 124－

126 实施界面 128。

图 5 是磁盘驱动器 110 应用于示于图 1 的系统 100 时，其一部分的框图。图 5 中，磁盘驱动器控制器 148 包括实施界面 128 的控制部件 150。对象 124－126 存储在构成磁盘 132 的存储媒体上。请求部件 152 被请求者 116－120 执行并调用界面 128 中方法构成逻辑公式化请求，一旦调用某一方法，控制部件 150 执行某种任务，以设定方式操作被标定对象。控制部件 150 返回一个事件，它可包括与任何识别对象关联的数据或属性。该事件也可以根据请求者 116－120 特定的调用方法来返回。

为了使面向对象的设备 110－112 能提供与面向块的设备连用的操作系统相同功能，设备 110－112 中的存储空间必须管理至相似程度。这样，在一个较佳实施例中，在存储设备 110－112 中，在存储在其上的对象 124－126 上面设置组织层。在一个较佳实施例中，面向对象的设备 110－112 把磁盘空间分配成一个或多个独占区域，称为分区。分区参照图 6 详细说明。在一个分区中，请求者 116－120 可创建对象。在一个较佳实施例中，分区中的结构是一个简单的、平直组织。在该组织上，任何操作系统都可映射其自身的结构。

图 6 说明存储媒体（例如一个磁盘 132）上的部分存储空间。该存储空间包括许多对象，诸如设备控制对象 154、设备关联对象 156、以及标为分区 0（还用 158 表示），分区 1（还用 160 表示），分区 N（还用 162 表示）的多个分区。每个分区还包括多个对象，例如分区控制对象 164、分区对象表 166 和多个数据对象 168（用数据对象 0－数据对象 N 来表示）。

与每个对象关联的一组属性。根据本发明的一个方面，提供存取控制属性。用 Set Attribute 方法进行设置（后文将详述）并提供一种手段，借助于它可存取被控制的特定对象。通过改变存取控制属性的版本号，可拒绝或给出某些请求者 116－120 对该特定对象的存取。

分簇对象（The clustering object）是一种属性，它表示在存储系统中特定对象是否应希望位于另一对象附近。再生属性(the cloning attribute)表示特定对象是否通过复制存储系统中另一对象而创建。一组大小属性(size attributes)确定特定对象的大小特征。例如，这组大小属性包括：表示可写入对象的最大偏移信息，分配给对象的块数，对象中存储数据的块数及对象中每块的字节数。

一组时间属性表示：对象何时创建，对象中数据最后一次被修改的时间，对象中最后一次被修改属性的时间。对象最好还包括一组属性，它确定文件系统中任何数据最后一次被修改的时间和任何属性最后一次被修改的时间。为了表示其它参数、特征或任何给定对象的特点，也可提供其它属性。

每一个对象还与一个对象标识符相关联，它由特定的存储设备 110－112 选定。并作为对创建对象命令的响应返回给请求者 116－120。该标识符最好是不带符号的，定长整数。在一个较佳实施例中，标识符的长度缺省为由特定存储设备 110－112 规定的大小，即可设置成一设备属性。进而，在一个较佳实施例中，为熟知对象、特定应用和希望实行的其它特殊功能保留预定的标识符子集（Ids）。

图 6 说明，存储媒体典型地包括许多熟知的对象，它们总是有特定的对象 ID。在某些情况下，这些熟知的对象存在于每个设备或每个分区中。

例如，一个熟知对象是设备控制对象 154，它最好包含由每个设备 110－112 所维护的属性，并且与设备本身或设备中的所有对象都有关。由 Set_Attribute 方法维护的属性将在后面的应用中详述。在一个较佳实施例中，每个设备 110－112 都有一个设备控制对象 154。

表 1 说明一组较佳设备控制对象(DCO)属性

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 控制设备密钥
	设备密钥	8	设备密钥 控制分区密钥
	保护级	1	确定保护任选
分区	分区计数	1	设备分区数
设备属性	对象属性	8	确定与设备上所有对象关联的特性

在一个较佳实施例中，DCO 属性包括时钟(简单地是一个单调计数器)，包含加密密钥或控制设备上所有其它密钥的其它主密钥的主密钥，和控制分区密钥且可用于锁定分区的设备密钥。属性还包括能识别预定保护级且与安全政策关联的保护级密钥，确定设备上分区数的分区计数器和确定被存取特

定设备上所有对象关联特性的对象属性。

为了胜任管理跨接于多个存储设备 110－112 的对象，每个存储设备 110－112 最好有一个能确定各设备间相关性的设备关联对象 156。例如，什么地方存储设备 110 和 112 是一对镜象设备或是一个阵列组的成员，并联对象 156 会识别这种关系。表 2 说明关联对象 156 的较佳属性。

表 2

名称	字节	语义
关联标识符	2	该组唯一 ID
关联类型	2	关联种类
成员表	n	
关联标识符	2	
关联类型	2	
成员表	n	

这些属性中最好包括一关联标识符，它是每个给定组的唯一标识符。属性中最好还包括确定设备间关联种类(例如镜象对、RAID5 等)的关联类型。属性中最好进一步包括一张成员表，它简单标明上述确定关联的成员是设备 110－122。

存储设备 110－112 上每个分区 158，160 和 162 最好包括含单个分区特性的分区控制对象 164。对象 164 最好不仅描述分区，也描述任何与该分区中所有对象相关的对象属性。每个设备 110－112 最好对设备上定义的每个分区包含一个分区控制对象 164，虽然图 6 说明存储在每个分区中的分区控制对象，但并不一定是这种情况。分区控制对象可以存储在平面文件系统中，而不是存储在上述分区中。

表 3 表示许多常包含在分区控制对象 168 中的属性。

类型	名称	字节	语义
	主密钥	8	加密密钥
	当前工作密钥	8	
	以前工作密钥	8	
分区属性	对象属性	8	定义与分区中所有

			对象关联的特性

这类属性中最好包含一个能定义整个分区加密密钥且可用于设置当前工作密钥的主密钥。属性中最好还包括用于加密和解密命令和数据信息的当前工作密钥和以前工作密钥。分区控制对象 164 最好还包括与指定分区中所有对象关联的对象属性。

图 6 还说明，每个分区最好包含分区对象表 166，它是在存储媒体上创建分区时，由控制部件 150 建立的对象。分区对象表 166 最好对每个分区具有相同的标识符，指定在存储媒体上导航对象文件系统的出发点。表 4 说明与每个分区对象表关联的属性表。

表 4

字段	字节	
对象 ID	8	打开、读、写、关闭该对象用的 ID
用户数据	N	用 POL 属性进行设置，用 GET ATTRIBUTE 得到数据值

如表 4 所示，该对象最好包含一张分区中所有对象的对象标识符表(即对象 ID)及分配给每个对象的用户空间量。请求者使用对象标识符打开、读、写和关闭该对象。此外，用户最好为每个对象 ID 分配用户空间便于设定分区对象表中的用户数据属性。在分区对象表 166 后，每个分区最好包含多个数据对象 168。每个数据对象 168 最好包含表 1 列出的一个或多个属性组，由数据存储系统的具体实施来确定。

面向对象的存储设备 110－112 最好能支持请求者 116－120 提供数据或存储数据的请求。而且，存储设备 110－112 最好承担对其它部件很可能是在操作系统或已有体系结构中完成的功能。空间管理及对与设备 110－112 中对象关联的属性维护最好由设备 110－112 自身完成。这些功能最好通过调用界面 128 支持的方法来实现，该界面 128 是通过每个存储设备 110－112 中的控制部件 150 来实现。下文详细讨论可以调用的多个方法。但是，为了便于更好地理解这些方法，用 7－1 和 7－2 提供流程图，说明根据本发明的一个方面，导航面向对象的文件系统。相信在下文详细讨论每个方法前，先讨论图 7－1 和 7－2 将便于理解本发明。

从框 170 延伸至 204 的图 7—1 和 7—2 说明如何在一个存储设备 110—112 的指定分区中查找对象。首先，请求者 116 获得设备控制对象 154 中的设备属性。这由框 172 表示。调用 Get_DCO_Attributes 方法使控制部件 150 返回存储设备在控制对象 154 中的属性。这由框 174 表示。然后根据从装置控制对象 154 返回的属性，请求者 116 选择某个给定的分区。这由框 176 表示。

请求者 116 一旦选定分区，请求者 116 立即调用 Get_DAO_Attributes 方法，如框 173 所示。它使控制部件 150 从存储在存储媒体 110 上的设备关联对象 156 获取属性。然后，控制部件 150 向请求者 116 返回设备关联属性，如框 175 所示。根据设备关联属性和控制属性，请求者 116 选定要询问的分区。如框 176 所示。

接着，请求者 116 调用 Get_PCO_Attributes 方法，它使控制部件 158 找到与请求者 116 要询问的指定分区关联的分区控制对象 164 中的属性。控制部件 150 获得并返回分区控制对象属性。如框 178 和 180 所示。如果选定分区中的对象不是请求者感兴趣的对象，则如框 182 和 176 所示，请求者可选择另外分区。

但是，假定请求者 116 已经找到感兴趣的分区，则请求者对选定的分区调用 Get_POL_Attributes，如框 184 所示。该方法使控制部件 150 从选定分区相关联的分区对象表 166 获得属性。然后把这些属性提供给请求者 116，如框 186 所示。

接着，请求者 116 调用 Open_Read_Only_POL 方法。如框 188 所示。如下文将详细讨论的，控制部件 150 获得存储在与选定分区关联的分区对象表 166 中的数据，但因为接供的数据是只读的，不能被修改或扩充，所以控制部件不能修改该对象的属性。如框 190 所示。

请求者然后调用 Read_POL 方法，它使控制部件 150 得到选定分区的对象表，提供给请求者 116 回顾。如框 194 所示。在选定的分区中选择好所要对象后，请求者 116 调用 Close_POL 方法，它使控制部件 150 关闭分区对象表。如框 196 所示。

找到所要对象或所有对象的 ID 后，请求者接着调用 Open_×××Objectx 方法。其中×××表明请求者根据所要的特定数据操作而调用的规定打开方法。Objectx 表明请求者从分区对象表中选定要操作或存取对象的 ID。符号

×××例如可表示 Open_Update 操作或或 Open_Read_Only 操作。这些在下文还要讨论且这些步骤如框 198 所示。

接着，请求者对控制部件 150 返回的对象执行希望的操作。下文将详细讨论可用于操作对象的各种方法。如框 200 所示。

最后，请求者一旦对所要对象完成所期望的操作或存取，请求者 116 立即调用下文将详述的 Close_Objectx 方法，该操作将关闭请求者 116 存取过的对象。

图 8－24 是说明为对存储在面向对象的存储设备（如设备 110）中的对象，实现所要的功能和所要的操作，请求者所能调用的各种示范方法的流程图。

图 8 是具体说明 Open_Create_Object 方法的流程图。当请求者 116 调用该方法，如框 208 所示，控制部件 150 创建一新的对象 ID 并将它输入与被创建对象所在分区关联的分区对象表中。如框 210 所示。然后控制部件 150 通过分配与对象相关联的块数等，并修改对象属性以表明对象创建时间和设置表 1 中所列的、与对象关联的其它属性，从而创建新的对象。如框 212 所示。接着，控制部件 150 返回请求状态，连同刚创建的新对象的 ID。如框 214 所示。

除简单创建一个对象外，请求者 116 可指定许多任选项。例如，在一个较佳实施例中，请求者 116 可指定：对象是否受口令保护、对象是否加密、一定的质量服务阈值(例如，对象是否备份)、锁定特性(例如，对象是否用一个对象锁或其它锁如分区锁或设备锁来锁定)、存取控制版本、镜象或其它备份支持(它会使所有更新被镜象成另一对象或以另一指定方式备份)，以表明将在以指定最小大小为单位来分配空间并设置冲突特性(例如写入一 UNIX 系统)。

为了调用该方法，请求者 116 向控制部件 150 提供的特定信息包括系统安全所需的许可信息，要创建对象的设备分区及上述任何选项。作为响应，在一个示范实施例中，控制部件 150 返回：设备上可用的容量、请求状态及新对象的 ID。

应注意，可调用该方法的特定例子包括与对象关联的所有数据。在这种情况下，可调用一个方法来创建对象、写入对象及关闭对象。

图 9 是说明 Open_Update_Objectx 方法的流程图。如框 220 所示，请求

者 116 调用此方法时，允许请求者 116 读和写特定对象。还可扩展该对象的长度。当调用此方法时，控制部件 150 对指定对象中设置一个属性以表明该对象正在使用中。请求者 116 提供许可信息、含对象的分区 ID、要存取对象的标识符、要进行的操作类型(如更新或写)及上述任何选项。作为响应，控制部件 150 返回请求状态和指定对象的长度及请求者 116 还可用的剩余容量。

图 10 是说明 Write_Object 方法的流程图。当请求者如框 242 所示调用此方法时，控制部件 150 在指定对象中的一定位置上写一定块数。

写方法也能引起调用其它方法。例如，若被存取设备 110－112 支持奇偶校验，则写操作会自动调用异或方法，它对被写数据执行异或操作，然后，奇偶数据被写入一个或多个预定的奇偶校验设备中。

为了调用此方法，请求者 116 提供许可信息、对象标识符、分区 ID、对象中被写的起始块位置、对象中能写的块数、任选信息和要写的数据。一旦调用此方法，控制部件 150 用提供的具体数据修改指定的对象。如框 244 所示。然后，控制部件 150 修改指定对象的必要属性，如对象长度、与该对象关联的时间记号等。如框 246 所示。然后，控制部件 150 在需要的地方修改其它对象的必要属性，诸如分区对象表中涉及的地方。如框 248 所示。然后，控制部件 150 向相应请求者返回请求状态。如框 250 所示。

图 11 是说明 Open_Only_Object x 方法的流程图。调用此方法时，控制部件 150 允许请求者 116 仅为读目的存取指定对象。这样，当调用此对象时，如框 230 所示，请求者提供许可信息、分区 ID、对象 ID 和任选信息。然后，控制部件 150 对指定对象设置一个属性表明对象正在使用中。如框 232 所示。接着，控制部件 150 在此对象设置只读属性。表明请求者不能写此对象。如框 234 所示。最后，控制部件 150 返回请求状态和指定对象的长度。如框 236 所示。

图 12 是说明 Read_Objectx 方法的流程图。当请求者 116 希望设备 110 从特定对象返回数据时，调用此方法。请求者提供许可信息、对象 ID、分区 ID、要读块的起始位置、要读的块数和其它所要的任选信息。作为响应，控制部件 150 返回请求状态、要返回的数据长度及响应于此方法而返回的实际数据。如框 258 所示。

图 13 是说明 Close_Object x 方法流程图。当请求者 116 如框 264 所示

调用该方法时，请求者提供许可信息、对象 ID 和任何所要的任选信息。作为响应，控制部件 150 如框 266 所示，修正指定对象中的数据。此外，作为写对象的结果而引起对象会有些改变。如果，以前对存储媒体没写成，则这时还可写。控制部件 150 如框 268 所示，更新对象 X 的属性。例如，若对象是新创建的对象，则其属性用创建时间及其它要求的属性信息加以更新。此外，被修改的属性表明对象中数据被最后一次修改的时间，若数据长度被改变，则控制部件 150 设置一属性，表明请求者不再使用该对象。作为任选，控制部件 150 还能更新与对象关联并反映在对象属性中的驻留高速缓存信息。如框 270 所示。例如，若特定请求者 116 将请求配置成要通知存储设备 110，关闭对象的数据仍要高速缓存或不再高速缓存，以便存储设备 110 的操作系统为对象快速连续关闭和再次开启的应用，保留高速缓存信息。但同时，存储设备 110 能为系统 100 中哪些因另一请求者在平均时间内存取同一对象而造成相干冲突事件中必须通知到的部件保留磁道。最后，控制部件 150 返回请求状态，如框 270 所示。

图 14 是说明 Remove_Object x 方法的流程图。当调用该方法时，如框 278 所示，控制部件 150 采取必要步骤从存储媒体删除该对象。如框 280 所示。然后，控制部件 150 修改与被删除对象所在分区关联的分区对象表，以及映指定对象 ID 又可被用了。如框 282 所示。最后，控制部件 150 返回请求状态，如框 284 所示。为了调用此方法，请求者 116 要提供许可信息、分区 ID、对象 ID 和任何所要的任选信息。最后，控制部件 150 返回请求状态，如框 284 所示。

图 15 是说明 Create_Partition 方法的流程图，该方法可由请求者调用，如框 290 所示。以便在存储设备 110 上创建一分区。应注意，当 Greate_Partitionx 方法把驱动器分成一个或多个区域时，不必计及存储媒体上的所有空间。此外分区也可跨越磁盘上不同区域。

在一个实施例中，用该方法创建平铺排列的分区，它代表设备上存储空间的真实划分。用这种排列按服务级划分空间，如数据阵列。这种分区不能改变大小但可删除和重建。

根据本发明的另一方面，把分区用作逻辑分区，便于逻辑上组织对象，而不是按服务级来管理空间。在第二个实施例中，分区可动态地改变大小。

为了调用该方法，请求者提供许可信息、任何所要的任选、分区 ID 和

为识别被分配给具体标定部分的空间而制定的初始空间分配。作为响应，控制部件 150 将存储媒体上的空间分配或指定分区，如框 292 所示。然后，控制部件 150 建立分区控制对象和分区对象表，如框 294 和 296 所示。如上所述，不能删除分区对象表。它可用作导航分区中对象的起始点。最后，控制部件 150 返回请求状态和说明已进行划分的分区图。如框 298 所示。

图 16 是说明 Remove_Partition X 方法的流程图。为了调用该方法，请求者 116 提供许可信息、任选信息和标识被删除分区的分区 ID。如框 304 所示。作为响应，控制部件 150 将先前与分区关联的空间去分配，如框 306 所示。接着，控制部件 150 删去与被删除分区关联的分区对象表中的所有对象，删除分区对象表并删除分区控制对象。如框 308、310 和 312 所示。最后，控制部件 150 返回状态状态和显示划分变化的分区图。如框 314 所示。

根据本发明的一个方面，数据管理政策要通知每个存储设备 110－112，以便存储设备在相互独立地动作时执行该管理政策。这提供的重要优点在于，不仅较少人的干预且可进行更可预测的和及时的管理控制。

例如，存储设备 110－112 上数据可按需地在每周进行备份。常规系统典型地在周末空闲期间备份，使系统可利用率不至于在一周业务期间中断。但是，在系统容量不断增大的同时，可利用率的窗口已逐渐缩小。这样，为备份可能达兆兆字节的数据，很难找出有如此长的系统中断时间。

根据本发明的一个方面，基于被赋予属性决定对该对象动作，面向对象的存储设备 110－112 能在对象到达其要备份的正确状态的任何时候通知备份功能。而且，全部文件的备份可分布在较长期间，在此期间其余文件仍在更新，而不影响数据完整性。

面向对象的存储设备 110－112 所能调用的属性的其它例子包括加密、压缩、空版本(versioning)和奇偶冗余。在每个例子中，仅需通知存储设备 110－112 与指定对象或对象组有关的政策。接着，设备本身可执行该功能或通知提供该服务的代理。

例如，存储设备 110－112 本身能完成压缩和加密。从而仅需通知设备的是对一个对象要求压缩或加密这一事实。对于由代理执行的管理功能，不仅管理功能政策必须通知存储设备，而且要告知如何识别执行该功能的代理，致使存储设备在代理执行该功能时，可访问代理。

根据本发明的一个方面，在对象间建立关联，便于识别具有相同属性或

相关的对象。例如，假设一个数据库包含 6 个文件或对象，一个也不能备份，除非所有的文件或对象都关闭或指定一个与其它全无关系的对象已经关闭。可能要用文件服务器 114 来管理对象间的这类相互关系。此外，本发明在阵列奇偶设置时，还建立内部设备相关。通过建立这样的组，其中，一个设备或对象弄清楚，则群中其余设备或对象也具有实质相同的特性，所以组管理会更有效。

图 17－24 是说明通过调用存储设备上对象展示的方法而完成的管理功能的流程图。调用这些方法使控制部件 150 和/或相关的控制部件采取步骤以完成与调用方法关联的管理功能。

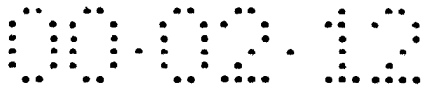
图 17 是说明 Export__Objectx 方法的流程图。如框 130 所示，请求者 116 通过提供许可信息、任选信息、对象 ID、目标设备 ID 和目标分区 ID，而调用该方法。该输出方法使存储设备 110－112 根据表示给定对象关联的属性的规则动作。例如，它可用于启动备份或支持给其它设备的对象定版本。

当调用 Export_Objectx 方法时，控制部件 150 从存储媒体获得指定对象，如框 322 所示。然后，控制部件 150 对请求者 116 指定的目标设备调用 Open_Create 方法。如框 324 所示。然后，控制部件 150 对提供指定对象的数据和属性的目标设备调用写方法。如框 326 所示。最后，控制部件 150 对目标设备调用 Close 方法，以便在完成对目标设备写以后，关闭目标设备上的对象。如框 328 所示。最后，控制部件 150 向请求者返回请求状态连同已被写入目标设备的对象的新对象 ID。如框 330 所示。

由控制部件 150 实施的界面 128 也支持允许请求者为先回顾获得对象属性并设置对象属性的方法。图 18 和 19 是分别说明相应 Get_Object x_Attributes 及 Get_Object x_Attributes 方法的流程图。

一旦调用图 18 所示方法，如框 336 所示，使控制部件 150 获得指定对象的属性。在一个说明性实施例中，请求者提供许可信息、对象 ID 或对象 ID 表、和任选信息。然后，控制部件 150 获得与该对象 ID 或对象 ID 表关联的属性并连同请求状态一起向请求者返回这些属性。如框 338 所示。

图 19 说明 Get_Objectx_Attributes 方法。如框 344 所示，可通过请求者向控制部件 150 提供许可信息、对象 ID 和任选信息来加以调用。然后，控制部件 150 用请求者提供的信息修改指定对象的属性，并返回请求状态及被修改的指定对象的属性。如框 346 和 346 所示。



根据本发明的另一方面，可以锁定对象，使它们只能在对对象加锁的服务器解锁后才能被存取。在一个说明性实施例中，对象可按对象级、分区级或设备级加锁。加锁机构为内部服务器提供存取方案。在一个较佳实施例中，用这种锁调度并行更新及在维护期间禁止存取。图 20、21 和 22 是说明可作为 Get_Attribute 和 Set_Attribute 方法例子的加锁方法的流程图。但是，对这些方法的具体例子还提供附加细节，使之可用于在请求者的簇之间共享数据。

图 20 是说明 Read_Lock_Attributes 方法的流程图。通过请求者 116 向控制部件 150 提供许可信息、对象、分区或设备 ID、加锁参数及所要的任选信息来调用此方法，如框 354 所示。作为响应，控制部件 150 确定指定对象是否设置有锁。然后，控制部件 150 为拥有锁的请求者返回请求状态。如框 356 所示。

图 21 是说明 Set_Lock_Attributes 方法的流程图。通过提供许可信息、对象、分区或设备标识符信息、加锁参数和任选信息，请求者可加以调用，如框 362 所示。当调用该方法时，控制部件 150 检查与标识对象关联的锁。如框 364 所示。然后，控制部件试图用请求者的标识完成加锁或解锁操作。如框 366 所示。若请求操作的请求者是锁的主人，则执行该操作。否则不执行该操作。不管哪种情况，控制部件 150 返回请求状态及锁主人服务器的 ID。如框 368 所示。

图 22 是说明 Reset_Lock_Attribute 方法的流程图。当锁主人的服务器不再有该功能时用该功能重置锁。通过提供许可信息、对象、分区或设备标识符信息、锁参数及所要的任选信息而加以调用此方法。如框 374 所示。作为响应，控制部件 150 对指定对象、分区或设备加锁，如框 376 所示。并返回请求状态及锁主人服务器的标识符。如框 378 所示。

图 23 和 24 是说明 Get 和 Set_Device_Association 方法的流程图。这些方法定义或询问设备 110－112 间的关系。这种关系的一个说明性实施包括：将存储设备 110－112 之一认作该组设备的第一设备或主设备而其余设备为该组的从属组员。该组的第一设备或主设备负责将组属性的变化分解到其它组员。若属性设置不是从该组第一设备或主设备提供的，则其它成员应拒绝这些属性设置。为使存储设备 110－112 执行这些功能，它们具有自检查的能力。这使这些设备可检查自身以确定是否是更大设备组的一个成员。

图 23 中说明 Get_Device_Associations 方法。通过提供许可信息和任选信息而调用该方法，如框 384 所示。作为响应，控制部件 150 返回请求状态及该设备成为组员所请求的关系；如框 386 所示。图 24 是说明 Set_Device_Associations 方法的流程图。通过提供许可信息、任选信息、成员表及定义关联的属性来调用此方法，如框 392 所示。作为响应，控制部件 150 修改存储媒体上所含的设备关联对象 156，如框 394 所示。被修改的设备关联对象包含请求者提供的属性及表示对象最后被修改的时间记号等。控制部件 150 返回请求状态，如框 396 所示。

上述的许可信息允许文件服务器 114 用门来控制对存储器的存取。请求者 116－120 给文件服务器必要的凭证，以得到存储设备 110－112 的响应。文件服务器 114 还命令存储设备 110－112 他们必须是坚持遵守设备安全政策的 I/O 请求者。通过 Set_Object_Attributes 方法以图例说明将从属于许可安全能力的密钥传达给存储设备 110－112。如果对存储设备 110－112 设置一定的安全级，则存储设备可配置成为安全性而检查每个 I/O 命令。但是，如上所述，某些应用不需采用安全性。进而，若一个特定的服务器簇具有一些位于另一设施上的设备，可以设想，与远处设备的通信定义较高的安全级，而对本地通信则不必如此。这允许对远处请求者或服务器采用安全性，但避免因对本地请求者或服务器采用同样安全性而引起的性能损失。

进而，每个存储设备 110－112 最好包含可读单调递增时钟，用于给安全信息和对象打时间记号。在一个说明性实施例中，各设备的时钟与全系统时基同步。在另一个说明性实施例中，文件服务器 114 提供逐个存储设备之间的偏差和值。

由此可见，本发明提供诸如磁盘驱动器之类的面向对象的存储设备，它比常规存储设备有显著优点。面向对象的存储设备显著改进了簇体系结构。例如，通过以面向对象的方式存储数据，该数据可由存储设备自身管理。对象向存储设备为驻留数据提供充分知识，足以承担管理其自身空间的职责。而且，当设备具有关于逻辑实体的构成信息时，可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设备中的数据，则必须控制所有元数据的并行存取活动。与此相反，在面向对象的设备中，许多元数据活动对访问它的系统是不透明的。这样，系统仅需关心对用户数据的存取冲突。此外，设备本身完成空间管理，就可避免因两个系统试图在同一时间管理同一

存储设备空间而产生的争论和混乱。

此外，抽象成对象更容易进行多机种计算。面向对象的存储设备提供一种至少操作系统可中断的组织的能力。

进而，使用面向对象的存储设备有许多理由增强簇系统的性能。例如，元数据决不需脱离设备本身，从而消除了一定数量的 I/O 操作。

此外，设备在任何时刻都知道哪些对象是打开的或关闭的，利用该信息可更有效高速缓存数据。因为设备知道正读取对象的格式，因而预取更有效。存储设备可更有效地确定顺序存取模式。设备中的高速缓存可一次为多个正存取它的系统保存元数据。此外，设备可参与服务决议的质量，诸如什么地方存放数据更合适。仅在设备有责任分配存储器时要这样做。与此相反，几乎没有一个操作系统能按磁盘驱动器的存储区域分配数据。所以，设备自身具备该能力也就是增强了设备性能。

本发明还可在排列成阵列的盘驱动器上实施。因为存储在盘驱动器阵列上的信息通常比盘驱动器本身更有价值，所以，常称驱动器阵列为廉价磁盘冗余阵列(Redundant Arrays of Inexpensive Discs)(RAID)。几种类型的 RAID 系统或几级 RAID 是已知的。例如，第一级 RAID，如上所述，其特点是提供镜象盘。到第五级 RAID，阵列中存储的数据和奇偶校验或冗余数据布及一组内的所有盘驱动器。第五级 RAID 把数据和检验信息交叉分布到包括检验盘在内的所有盘。其余级(例如 2－4 级)的 RAID 在题为“具有阵列支持控制器和接口的盘阵列”，专利号为 5617425 的美国专利中有详细讨论。

图 25－29 说明根据本发明一个方面执行的写操作，其中，存储数据作为阵列盘驱动器的对象。在图 25 所示的实施例中，文件服务器 114、请求者(或主机)116 和互连 125 连至由目标驱动器 402 和奇偶校验驱动器 404 构成的存储设备(如 110－112)的盘驱动器阵列。目标驱动器 402 保持住一个对象或要写的一部分，而奇偶校验驱动器 404 保持住目标驱动器 402 上所存目标对象关联的奇偶校验信息。

在图 25 中，实施的驱动器阵列作成 RAID5 阵列，其中数据和奇偶校验交叉分布在该组的所有驱动器中。从而，仅对当前写操作，驱动器 402 是目标驱动器，且驱动器 404 是奇偶校验驱动器。换言之，目标驱动器 402 也可以保持奇偶校验信息，而奇偶校验驱动器 404 也可以保持数据。但是，对下述的单个写操作，驱动器 402 是目标驱动器，而驱动器 404 是对应的奇偶校

验驱动器。应注意的是，本发明也可用其它级 RAID 而不是五级 RAID 来实施。本发明对这种 RAID 系统的应用，对于本领域技术人员是显而易见的。

在图 25 中，目标驱动器 402 和奇偶校验驱动器 404 经光纤通道接口或其它适当接口，诸如其它串行接口相互连接。

图 26 和 27 分别说明目标驱动器 402 和奇偶校验驱动器 404。每个驱动器包括控制部件 150 和一个或多个盘 132。每个驱动器还包括读/写电路 406(例如上述的数据头)和异或(XOR)电路 408。目标驱动器 402 包括存储被写目标对象的盘空间 410。奇偶校验驱动器 404 包括存储对应奇偶校验对象的磁盘空间 412。下面，图 28 和 29 将详细讨论驱动器 402 和 404 的操作。

用小型计算机系统接口(SCSI)实现常规盘阵列，XOR 命令使盘驱动器进行必需的位操作，实施防止驱动器故障的奇偶校验保护。这种命令要求主机(或请求者)有一个存取该盘的扇区，这样，作为写入盘驱动器的任一扇区含奇偶校验信息的另一个盘驱动器的相应扇区可被适当更新。但是，上述面向对象的盘驱动器在主机和该盘驱动器的实际存储扇区之间引入一提取层。具体而言，盘驱动器把盘空间作为对象来管理，使主机(或请求者)存取不到寻址方案的下面扇区。盘驱动器自身有空间管理职责，使请求者或主机不可能把写入一个盘驱动器的一部分数据与另一个盘驱动器上的一个位置相关。这样，请求者就不知道盘驱动器上被写块的地址，也就算不出相应的奇偶校验地址。这使得如上所述，在面向对象的盘驱动器中使用常规的 XOR 功能，即使不是不可能，也是十分困难的。

从而，本发明提供一种称为 Define_Parity_Group 方法，该方法可对组成奇偶校验组的一组盘驱动器中的每个盘驱动器调用。该方法完成两件事。首先，它提供足够信息。可调用标准 Write_Object 方法来完成与常规驱动器阵列中基于扇区的 XOR 命令相同的功能。它也使在该组每个驱动器上创建的一个对象能保持特定的奇偶校验数据的驱动器共享。奇偶校验对象的 ID 是众所周知的，每个驱动器都知道，从而任何试图更新奇偶校验信息的驱动器，都知道正确的对象标识符，可用它找到所要的请求。

图 28 详细叙述 Define_Parity_Group 方法。首先，请求者或主机对奇偶校验组中的每个驱动器调用该方法。如框 420 所示。为了调用该方法，请求者要提供如下信息：

1．一张有序的，组成奇偶校验组的驱动器表。在简单的说明性实施例

中，还包括每个驱动器的序号和地址。

2．用于计算奇偶校验的算法。在简单说明性实施例中，对被写数据的块地址执行模数运算。该运算既得出奇偶校验驱动器地址(根据上述有序表第 1 项)，还给出奇偶校验驱动器上奇偶校验对象中的相对块地址(含所需奇偶校验信息的奇偶校验对象的相对部分)。

3．奇偶条的数据量，实例中以块为单位。如果奇偶校验数据分散在每个驱动器的空间，则数据量按基本分配单位。

4．奇偶校验对象的标识符。调用 Write_Object 方法更新奇偶校验对象的驱动器，把奇偶校验对象标识符发向该奇偶校验驱动器的该对象 ID，定作上述两项的开始。还应注意到，最好实施多级奇偶校验(例如两级奇偶校验)。这样，每个驱动器可具有多至两个奇偶校验对象。在一个说明性实施中，在驱动器用于具有两级奇偶校验的盘阵列时，每个驱动器分配和保留二个熟知的对象 ID。第二个奇偶校验对象的存在表示正采用两级奇偶校验。

5．奇偶校验对象分配政策。它表示每个驱动器是否把奇偶校验对象分配成盘空间的单个邻接区域或是否要用用户数据对象来点缀奇偶校验对象。当奇偶校验对象和数据对象如图 26 和 27 所示作为邻接盘空间，这仅是为了例示。应注意，若用数据来点缀奇偶校验对象时，它仍可预分配。

作为调用 Define_Parity_Group 方法的响应，奇偶校验组内每个盘驱动器中的控制部件 150 计算奇偶校验数据所需空间的百分比。如框 422 所示。奇偶校验对象所需的空间是取决于奇偶校验组表中的盘驱动器数目。例如，若表中有 9 个盘驱动器，则每个驱动器必须将其九分之一空间分配给奇偶校验信息。用请求者或主机按调用的方法提供的，熟知的奇偶校验对象 ID 来识别空间量。如框 424 所示。

奇偶组或组表中的每个驱动器保存定义奇偶校验组的信息，这样，盘驱动器每次被驱动或复位时，就能验证该奇偶校验组是否受到损害。为此，该信息应存储在非易失性存储器中，如框 426 所示。

因为已创建有奇偶校验的盘驱动器，并在每个盘驱动器上分配一定空间来保持一个或多个奇偶校验对象，所以，以数据对象存储在一个或多个驱动器上的数据就能被更新。图 29 是说明根据本发明的一个方面，更新数据对象并更新对应奇偶校验对象的框图。

为了更新数据，请求更新数据的请求者 116 对该奇偶校验组内的一个盘

驱动器调用上述 Write_Object 方法。在图 25－27 所示的实施例中，请求者 116 调用 Write_Object 方法。如图 26 中对目标驱动器 402 的箭头 428 所示。在实施例中，为了调用此方法，请求者 116 提供标识被更新对象的对象标识符、分区 ID、对象中被写的起始块位置、对象中被写的块数、任选信息和要写的数据。目标驱动器 402 知道服务于 Write_Object 方法，必须包含与被更新对象关联的更新的奇偶校验信息。目标驱动器 402 知道这个信息。因为它把执行 Define_Parity_Group 方法期间提供和产生的信息已存在非易失性存储器中。

为了更新奇偶校验信息，目标驱动器 402 执行一系列步骤。首先，它从目标对象中的指定位置读出老数据，并把它连同在该位置上要写的新数据提供给 XOR 电路 408。如图 29 的框 432 和图 26 的箭头 434、436 和 438 所示。

其次，目标驱动器 402 用老数据异或(XOR)新数据获得中间奇偶校验信息。如图 29 的框 440 所示。目标驱动器 402 在图 26 的一个输出口 442，给出中间奇偶校验信息。接着，目标驱动器 402 把新数据写入目标对象 410 中的目标位置，这样，更新目标对象。如图 29 的框 444 所示。

接着，目标驱动器 402 对奇偶驱动器 404 调用 Write_Object 方法。对应目标对象 410 的奇偶校验对象正好被更新。如图 29 的框 446 和图 27 的箭头 448 所示。目标驱动器 402 可用许多方法算出奇偶校验对象的目标位置。例如，目标驱动器 402 可从被写目标对象块的相对扇区地址计算此位置。该相对地址被奇偶校验组的驱动器数除，得到奇偶校验驱动器 404 上奇偶校验对象的相对地址。用 Define_Parity_Group 方法中规定的算法确定奇偶校验驱动器地址。这样，目标驱动器 402 构成 Write_Object 方法并对奇偶校验驱动器进行调用，来识别奇偶校验对象 412，并用相对地址识别该对象内的适当位置。

例如，为了计算被更新驱动器 404 上奇偶校验对象中的相对块，目标驱动器 402 可用下述公式：

$$B=INT(S/D-1) \qquad \text{式 1}$$

其中：B 是奇偶校验对象中的相对块；

S 是目标驱动器 402 上被写的相对扇区地址；

D 是奇偶校验组中的驱动器数。

为了计算奇偶校验驱动器地址，目标驱动器 402 可用下列公式：

$$P = \text{Mod}(S/D-1) \qquad \text{式 2}$$

其中：P 是至奇偶校验驱动器的奇偶校验组内驱动器表中的偏移(用于计算 P 的表必须包含目标驱动器 402 的地址)。

响应该 Write_Object 方法，奇偶校验驱动器 404 认为该命令是对其奇偶校验对象写，并完成奇偶校验操作。这些操作包括读老的奇偶校验数据，如图 29 框 450 及图 27 箭头所示。然后，奇偶校验驱动器 404 从目标驱动器 402，取中间奇偶校验数据对老的奇偶校验信息进行 XOR 操作。如图 29 框 454 和图 27 箭头 456 和 458 所示。异或操作结果是更新了奇偶校验信息并把它写入盘 132 的奇偶校验对象。如图 29 的框 460 和图 27 的箭头 462 和 464 所示。从而完成奇偶校验对象的更新。

这样，本发明相对通常使用的基于扇区的奇偶校验方案具有多个优点。例如，不再用盘上的用户数据来点缀奇偶校验信息。从而可能相连存储用户文件(或对象)，性能更好。此外，对象可备份或输出为真实用户数据文件。由于没有点缀的奇偶校验数据，因而它们自然有意义且有用。

同样，当下载一个数据库到奇偶校验组时，应用程序可在数据库一下载立即从请求者存取它，而不必等计算奇偶校验。作为后台处理，可构成奇偶校验信息而不必中断数据存取。这有利于更快地使用数据。常规系统不可能在装入所有数据包括奇偶校验数据结构之前，存取数据。奇偶校验数据点缀整个用户数据，用户数据不能独立装载，而只能与奇偶校验计算连接。

进而，由于奇偶校验组中的每个盘驱动器知道该奇偶校验组中的其它成员，盘驱动器可验证奇偶校验更新对准的驱动器是否正确。若驱动器被替换或其地址更改，则可以检测出来并放弃操作，从而避免奇偶校验数据讹误。

调用奇偶校验保护时不需用特定的 XOR 命令。发出 Define_Parity_Group 命令，驱动器得知简单的 Write_Object 命令扩展到包括奇偶校验保护。这可由驱动器透明地进行。

此外，请求者无需向驱动器发出附加 I/O 命令即可支持两级奇偶校验。对一个驱动器调用两个 Define_Parity_Group 方法，它启动了双重故障保护。驱动器知道它必须更新两个奇偶校验对象，每个奇偶校验组中一个。驱动器也可检查以确信两个奇偶校验组仅共用一个驱动器(否则这种配置也许是无效的)。

本发明包括一数据存储系统，它包括第一存储媒体 132，构成多个对象

124－126 的数据存储在上面。每个对象具有表示该对象特征的属性。这些对象包括存储冗余信息的冗余对象 412。第一控制部件 150 可操作地耦合至存储媒体 132，并配置成为对象 124－126 提供界面 128。界面 128 展示存取对象 124－126 的调用方法(方法 O－方法 N)。

在一个较佳实施例中，方法 O－N 包括 Define_Redundancy 方法，调用该方法时，控制部件 150 对冗余对象 412 分配存储空间。在另一个实施例中，调用该 Define_Redundancy 方法时，控制部件 150 存储控制部件 150 所属冗余组的信息。

根据另一个实施例，当调用 Define_Redundancy 方法时，控制部件 150 计算要分配给冗余对象 412 的存储空间大小。

根据另一个实施例，调用 Write_Object 方法使控制部件 150 用新数据更新指定的数据对象并根据新数据更新对应冗余对象 412。

根据另一个实施例，存储系统包括多个存储设备 402 和 404，第一个存储设备 402 包括第一存储媒体 132 和第一控制部件 150，而第二个存储设备 404 包括第二存储媒体 132 和第二控制部件 150。在这个实施例中，指定的数据对象至少部分对象和对应冗余对象存储在第一个存储媒体上并有部分对象和对应冗余对象存储在第二存储媒体上。在一个说明性实施例中，第一控制部件 150 对第二存储设备调用 Write_Object 方法来更新冗余对象。

本发明也可实施为一种维护数据存储系统中冗余的方法，该数据存储系统具有多个存储设备，每个存储设备都包括存储媒体 132 和控制部件 150。该方法包括在所述存储媒体 132 上存储数据，而把被存数据配置成多个对象 124－126，且每个对象含有表示所述对象特征的属性。所述对象包括存储冗余信息的冗余对象 412。所述方法进一步提供展示存取对象调用方法(方法 O－方法 N)的界面 128。

在一个说明性实施例中，在步骤 420 所述方法包括在对多个存储设备组调用 Define_Parity 方法，以配置所述组完成期望的冗余方案。在一个说明性实施例中，所述方法包括响应调用步骤，对所述组的每个存储设备创建一个冗余对象 412。在步骤 422，所述方法进一步包括计算分配给所述冗余对象 412 的部分存储空间。

应该理解，虽然上文已叙述了本发明各个实施例的特点和优点并详述其结构和功能，但这种揭示仅是例示性的，在本发明的原理内，可作出许多变

换，尤其是对于结构和配置方面，以使本发明包括所附权利要求中的术语的广义含义所表示的整个范围。例如，不脱离本发明的范围和精神，特定元件可取决于使用的特定界面方法和冗余方案改变，而实质上仍保持相同的功能。

说 明 书 附 图

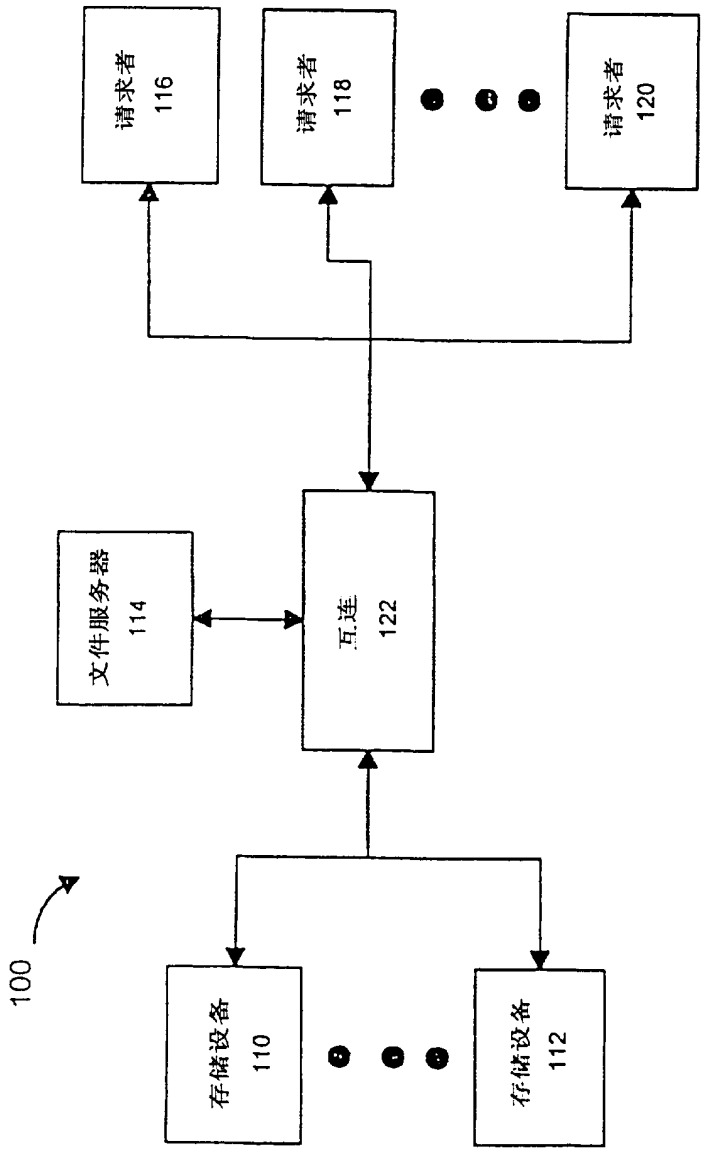


图 1

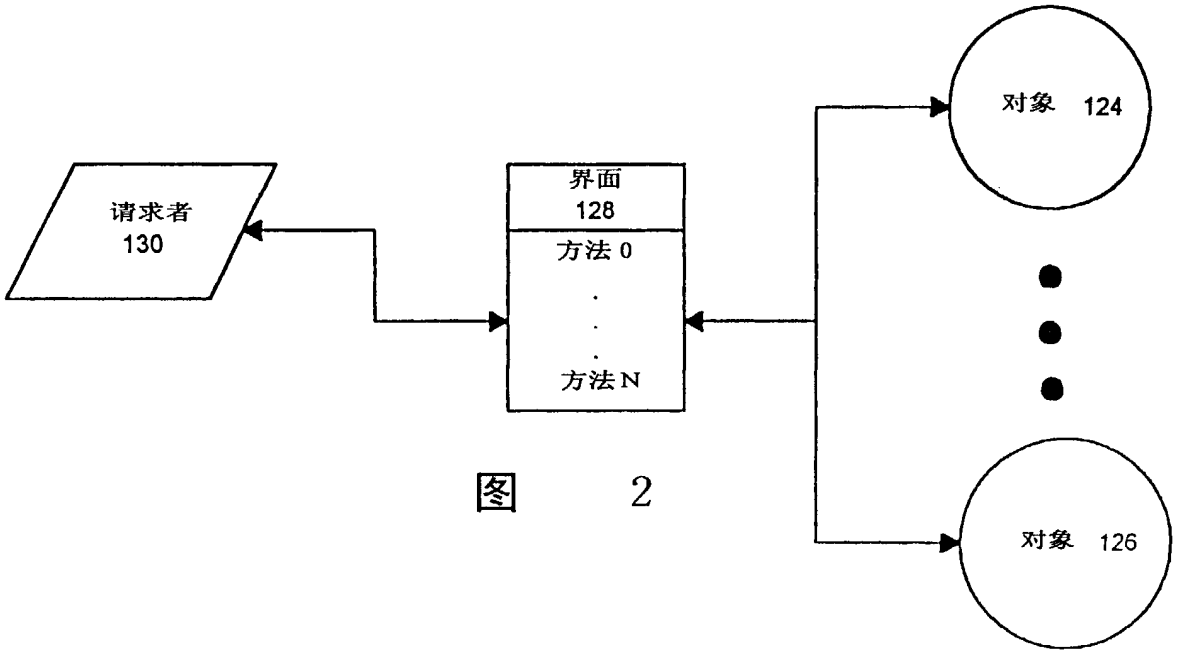


图 2



图 3－1

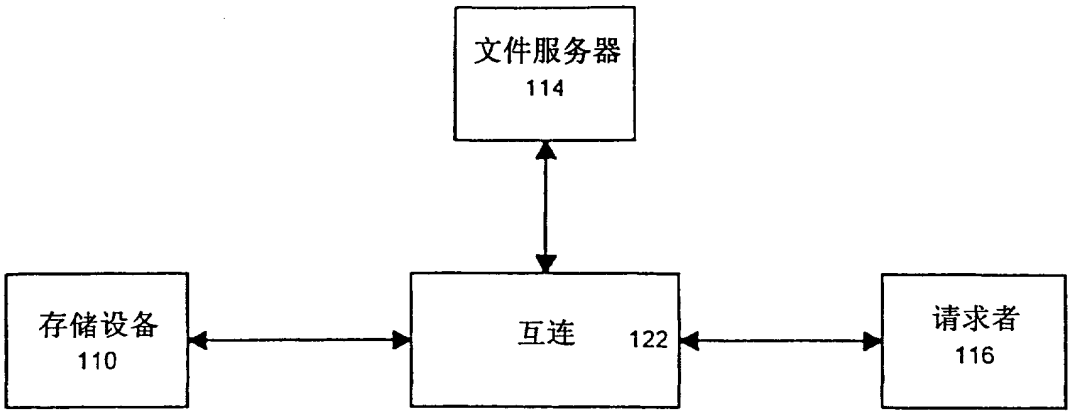


图 3－2

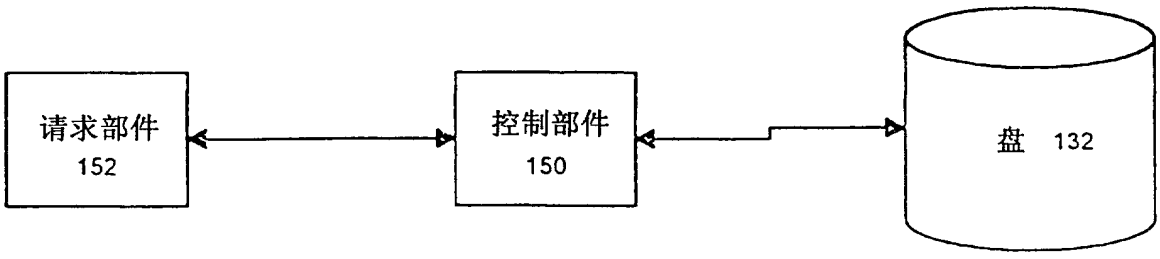


图 5

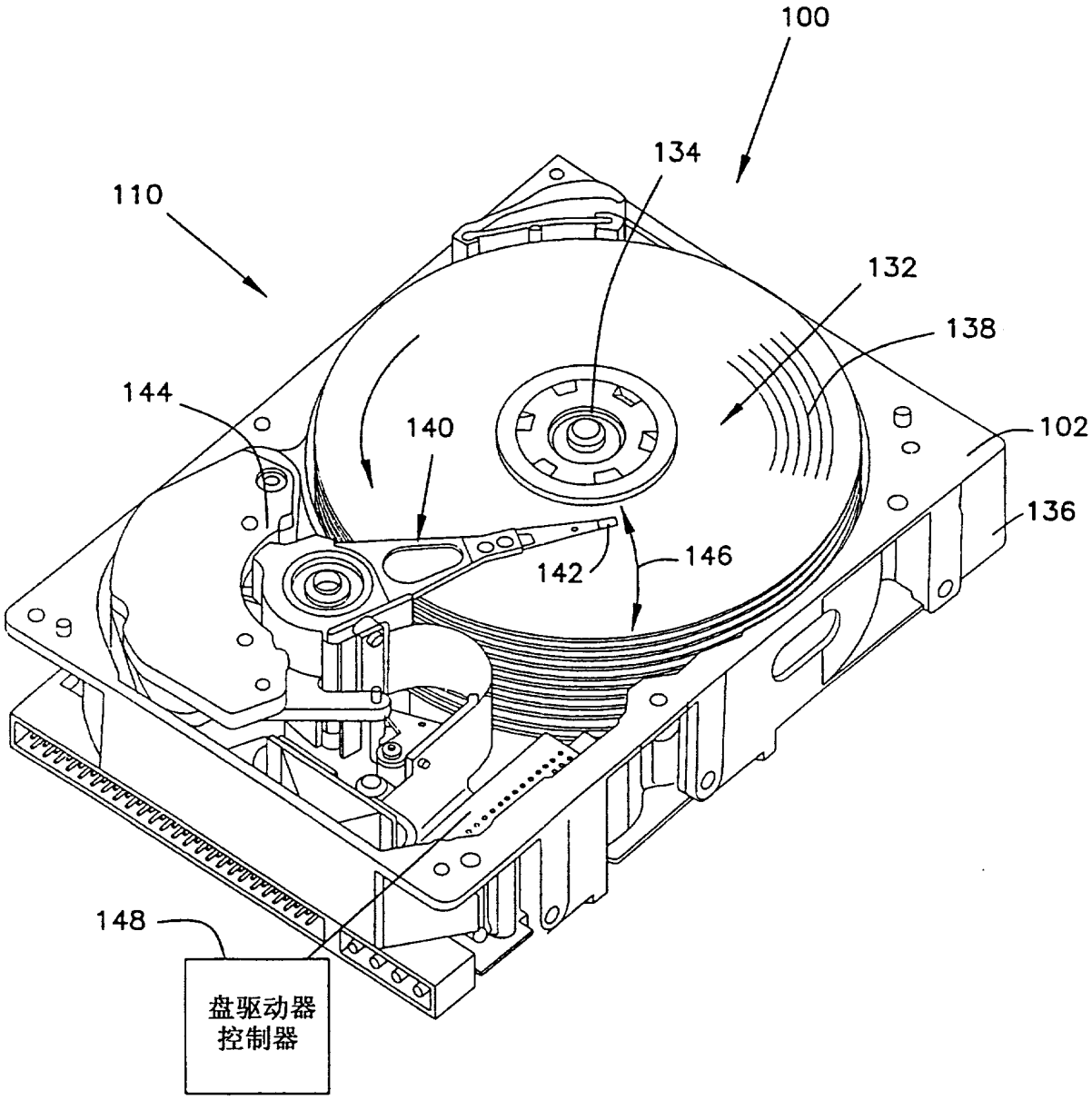


图 4

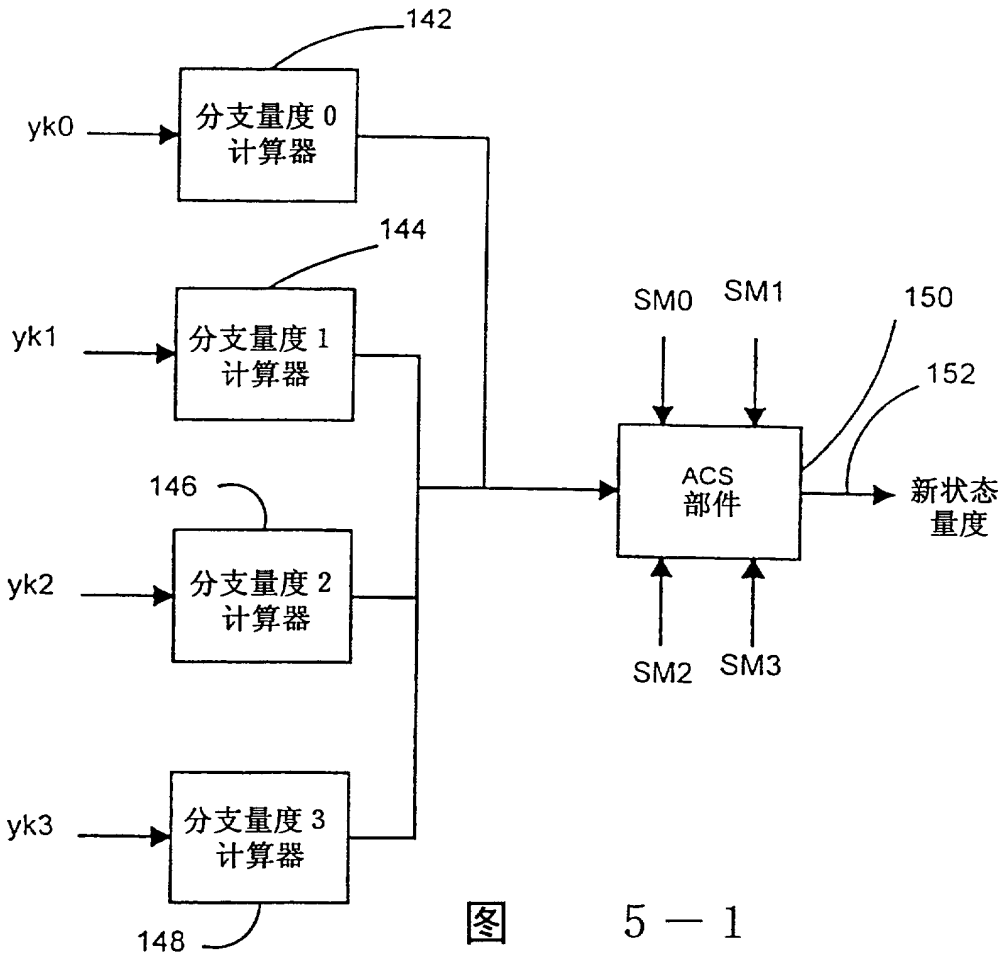

142
yk0
分支量度 0
计算器
144
yk1
分支量度 1
计算器
146
yk2
分支量度 2
计算器
yk3
分支量度 3
计算器
148
SM0
SM1
150
152
ACS
部件
新状态
量度
SM2
SM3

图 5－1

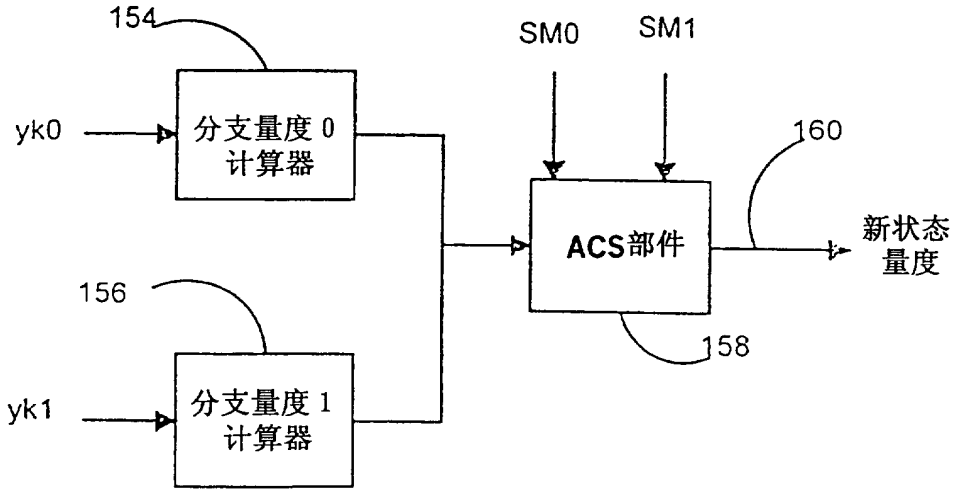

154
yk0
分支量度 0
计算器
156
yk1
分支量度 1
计算器
SM0
SM1
160
ACS部件
新状态
量度
158

图 5－2

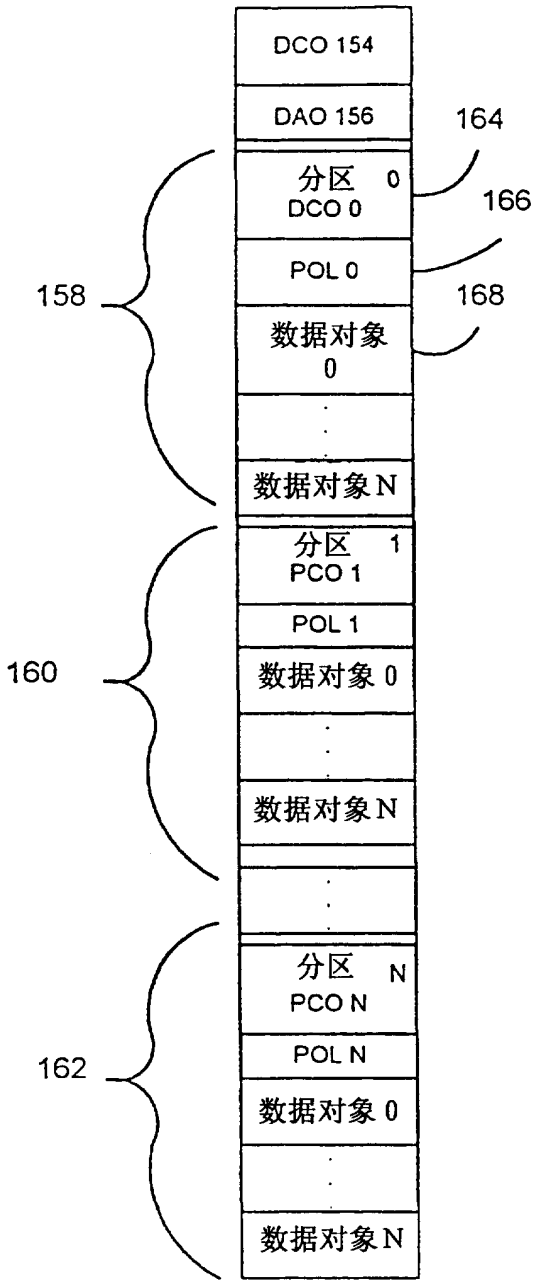

DCO 154
DAO 156
164
分区 0
DCO 0
166
POL 0
158
168
数据对象
0
数据对象N
分区 1
PCO 1
POL 1
160
数据对象 0
数据对象N
分区 N
PCO N
POL N
162
数据对象 0
数据对象N

图 6

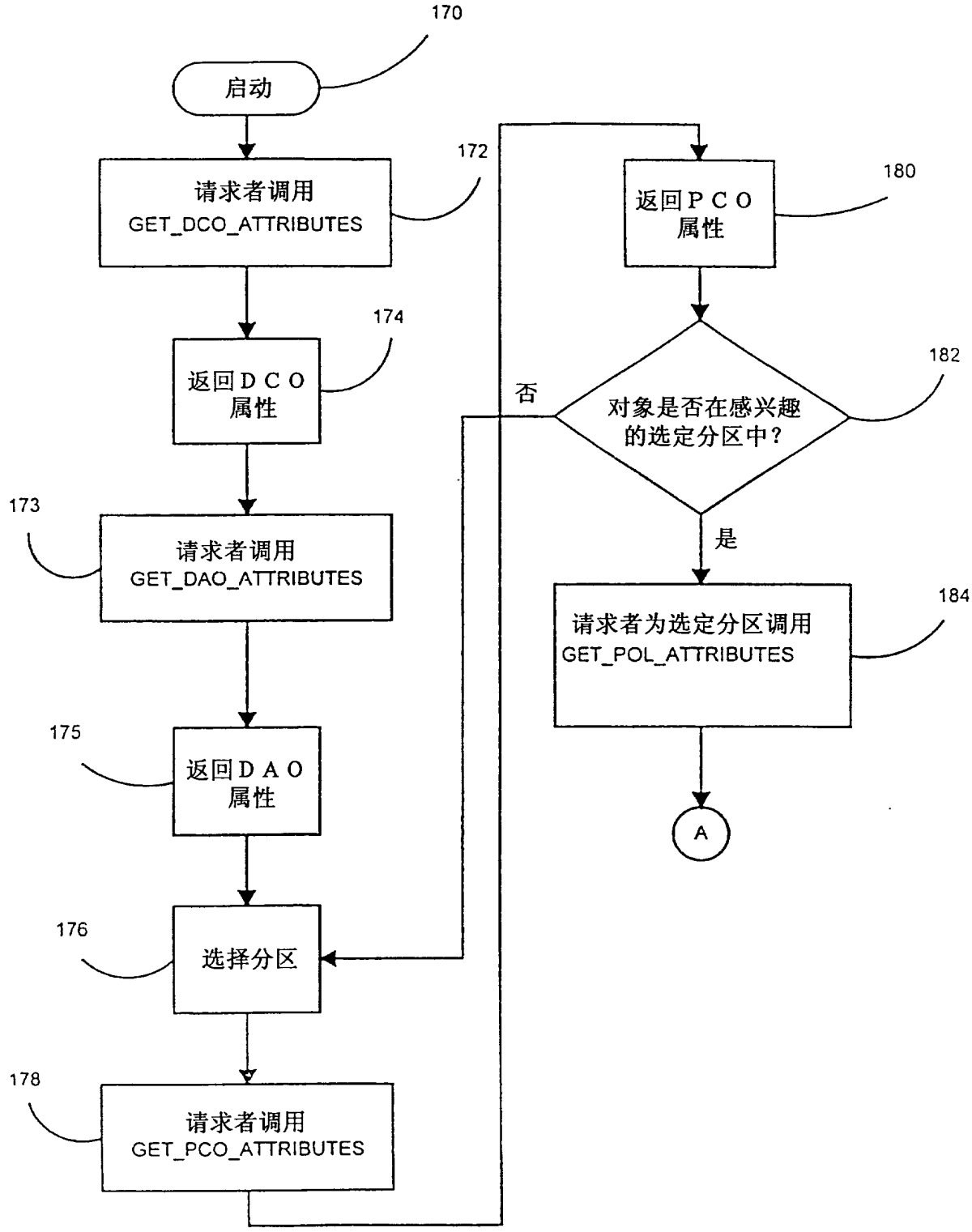

170
启动
172
请求者调用
GET_DCO_ATTRIBUTES
174
返回ＤＣＯ
属性
173
请求者调用
GET_DAO_ATTRIBUTES
175
返回ＤＡＯ
属性
176
选择分区
178
请求者调用
GET_PCO_ATTRIBUTES
180
返回ＰＣＯ
属性
182
对象是否在感兴趣
的选定分区中？
否
是
184
请求者为选定分区调用
GET_POL_ATTRIBUTES
A

图　　7－1

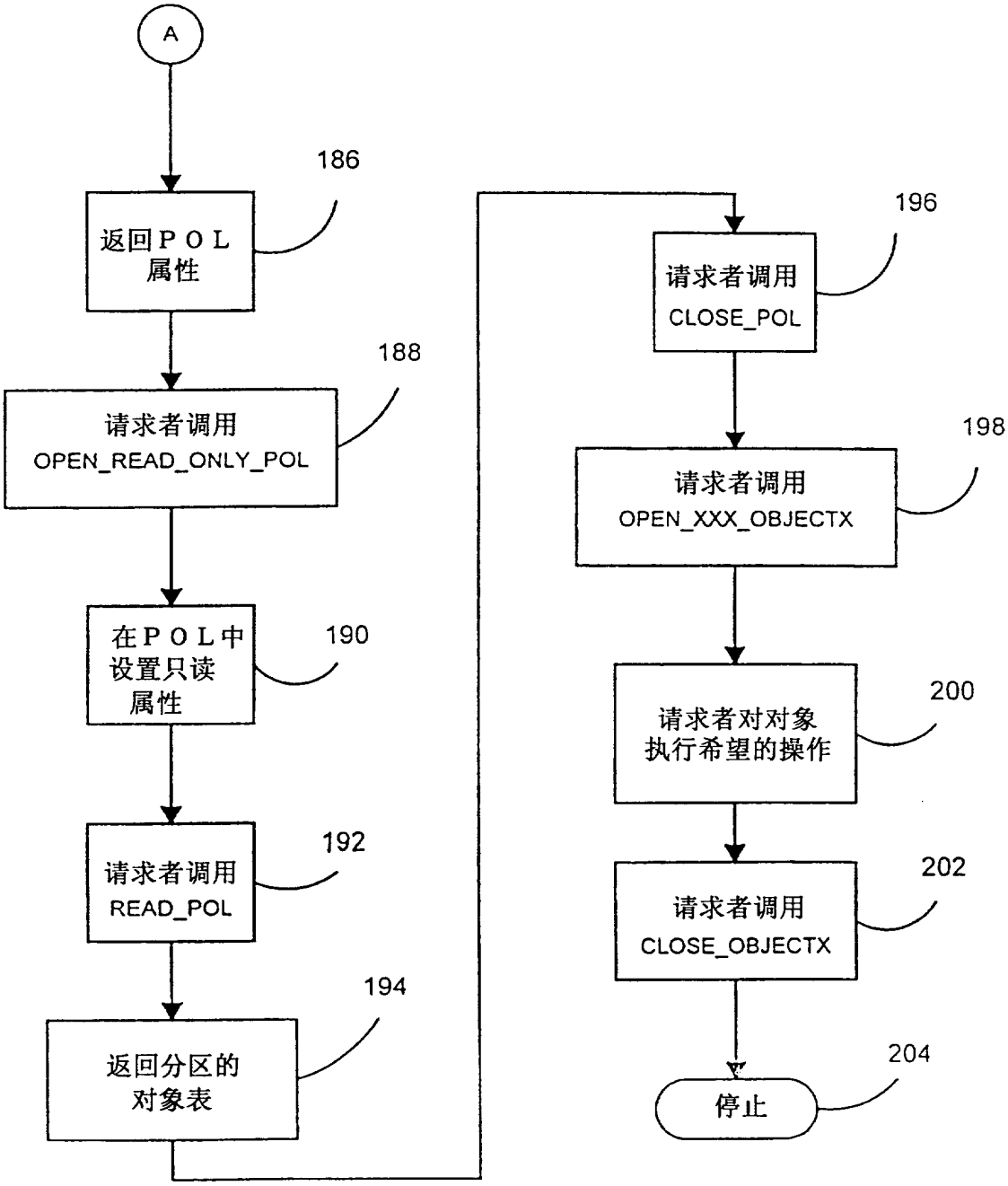

A
186
返回ＰＯＬ属性
188
请求者调用 OPEN_READ_ONLY_POL
190
在ＰＯＬ中设置只读属性
192
请求者调用 READ_POL
194
返回分区的对象表
196
请求者调用 CLOSE_POL
198
请求者调用 OPEN_XXX_OBJECTX
200
请求者对对象执行希望的操作
202
请求者调用 CLOSE_OBJECTX
204
停止

图　　7－2

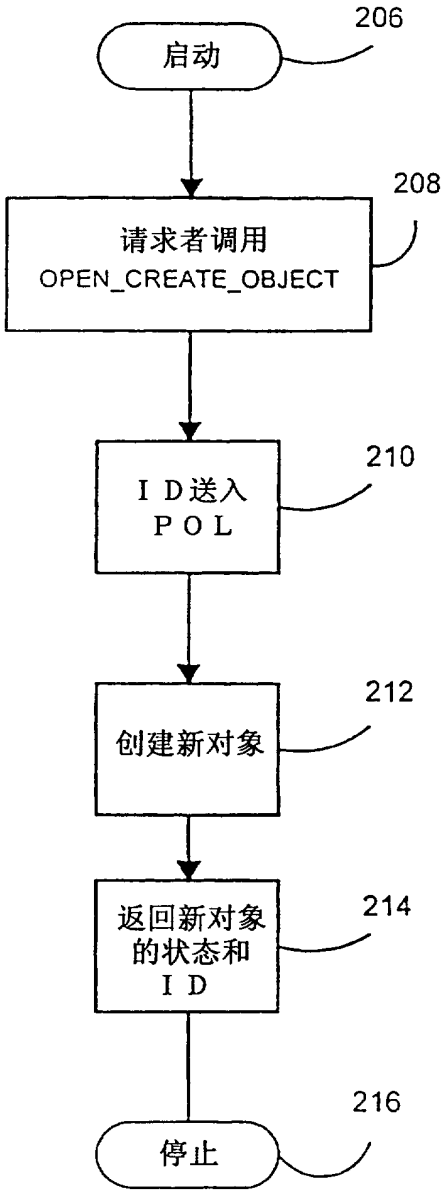

启动
206
请求者调用
OPEN_CREATE_OBJECT
208
ＩＤ送入
ＰＯＬ
210
创建新对象
212
返回新对象
的状态和
ＩＤ
214
停止
216

图　　8

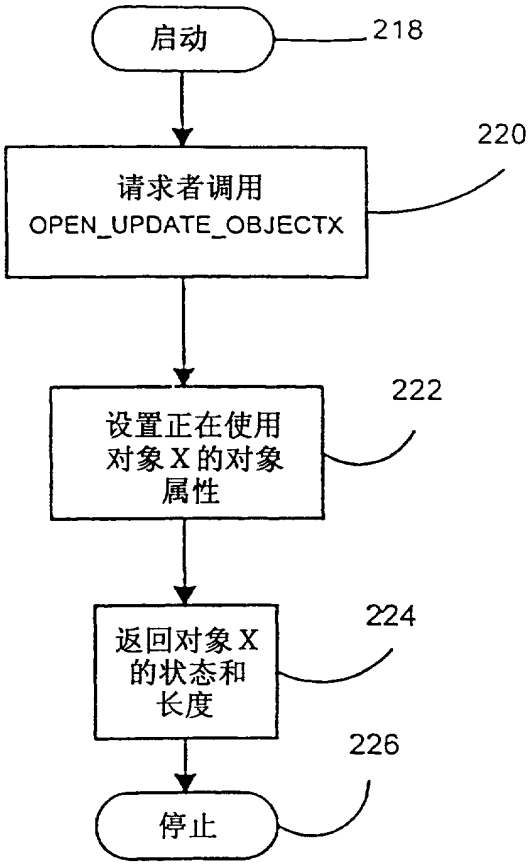

启动
218
请求者调用
OPEN_UPDATE_OBJECTX
220
设置正在使用
对象Ｘ的对象
属性
222
返回对象Ｘ
的状态和
长度
224
停止
226

图　　9

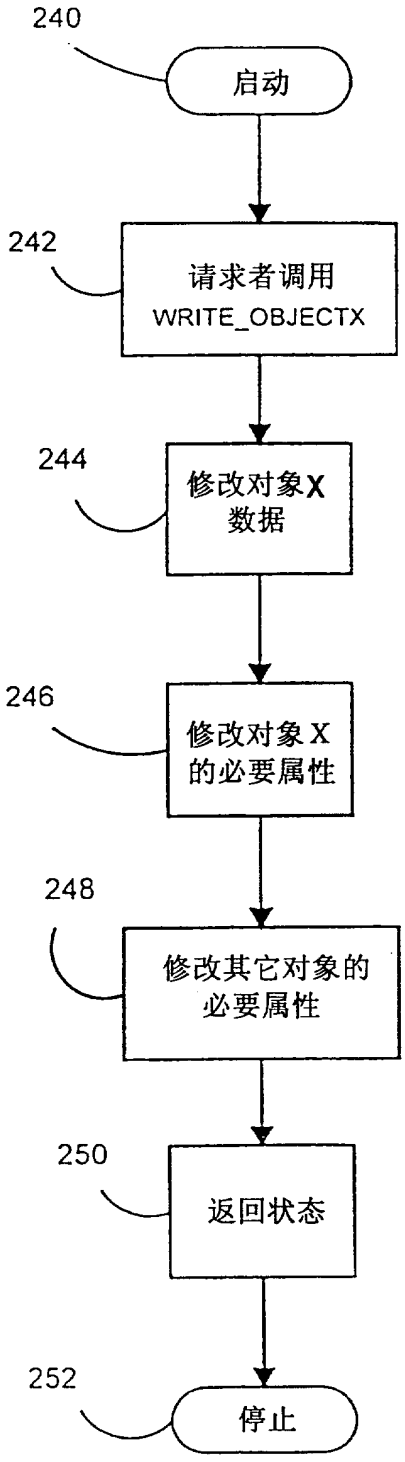

240
启动
242
请求者调用
WRITE_OBJECTX
244
修改对象X
数据
246
修改对象X
的必要属性
248
修改其它对象的
必要属性
250
返回状态
252
停止

图 10

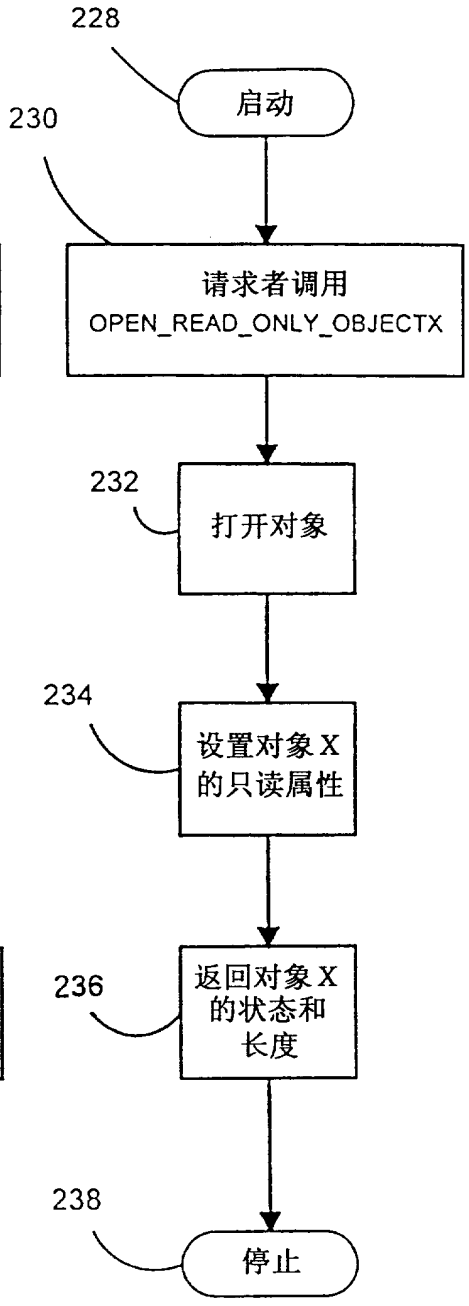

228
启动
230
请求者调用
OPEN_READ_ONLY_OBJECTX
232
打开对象
234
设置对象X
的只读属性
236
返回对象X
的状态和
长度
238
停止

图 11

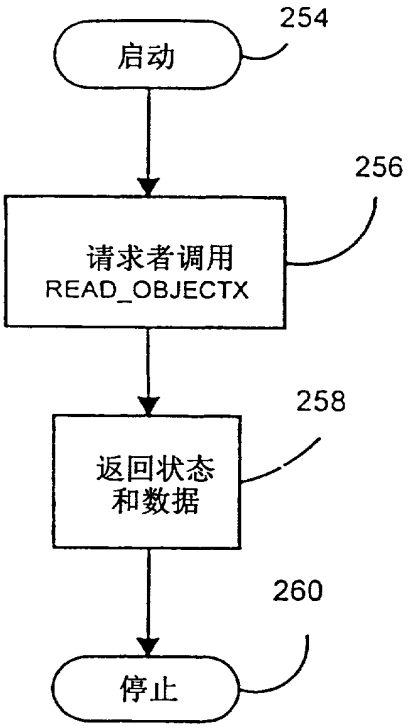

启动
254
请求者调用
READ_OBJECTX
256
返回状态
和数据
258
停止
260

图 12

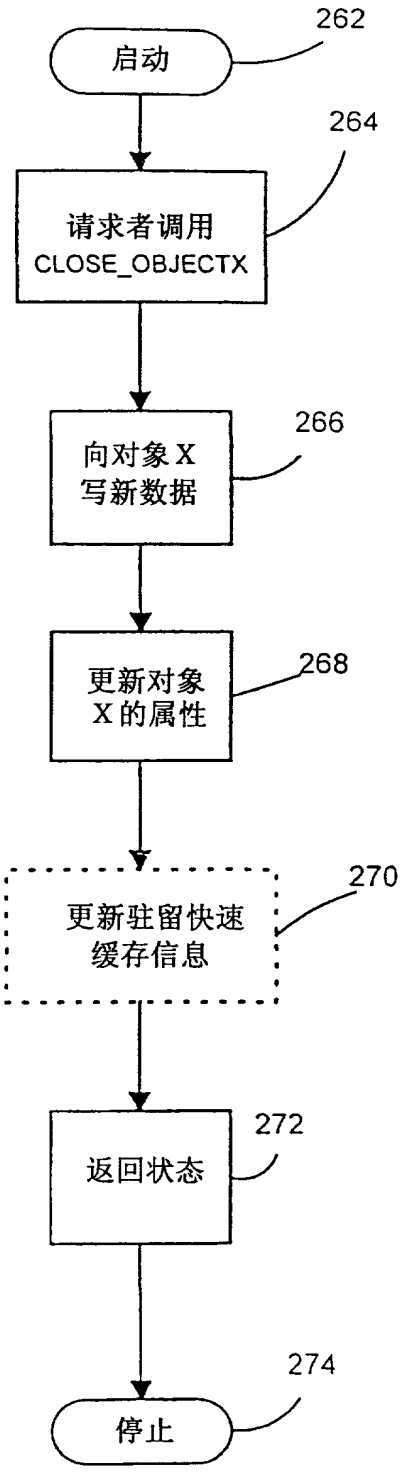

启动
262
请求者调用
CLOSE_OBJECTX
264
向对象X
写新数据
266
更新对象
X的属性
268
更新驻留快速
缓存信息
270
返回状态
272
停止
274

图 13

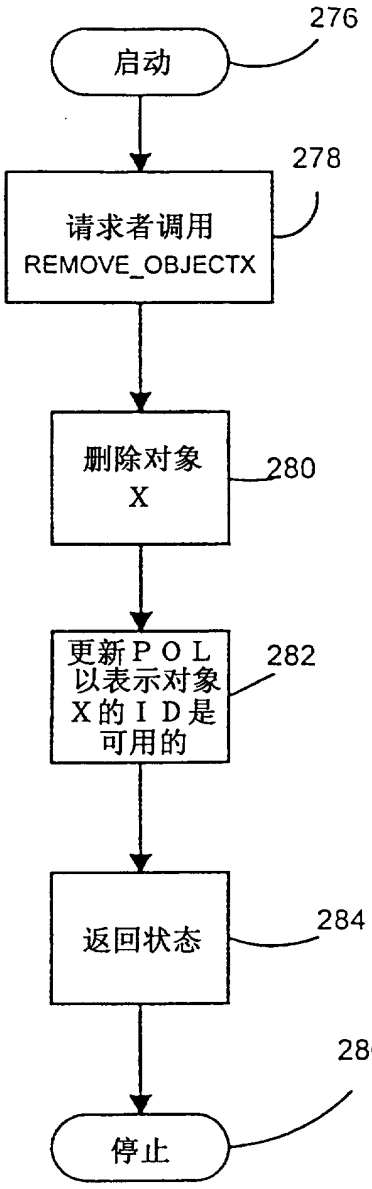

启动
276
请求者调用
REMOVE_OBJECTX
278
删除对象
X
280
更新ＰＯＬ
以表示对象
X的ＩＤ是
可用的
282
返回状态
284
停止
286

图　　１４

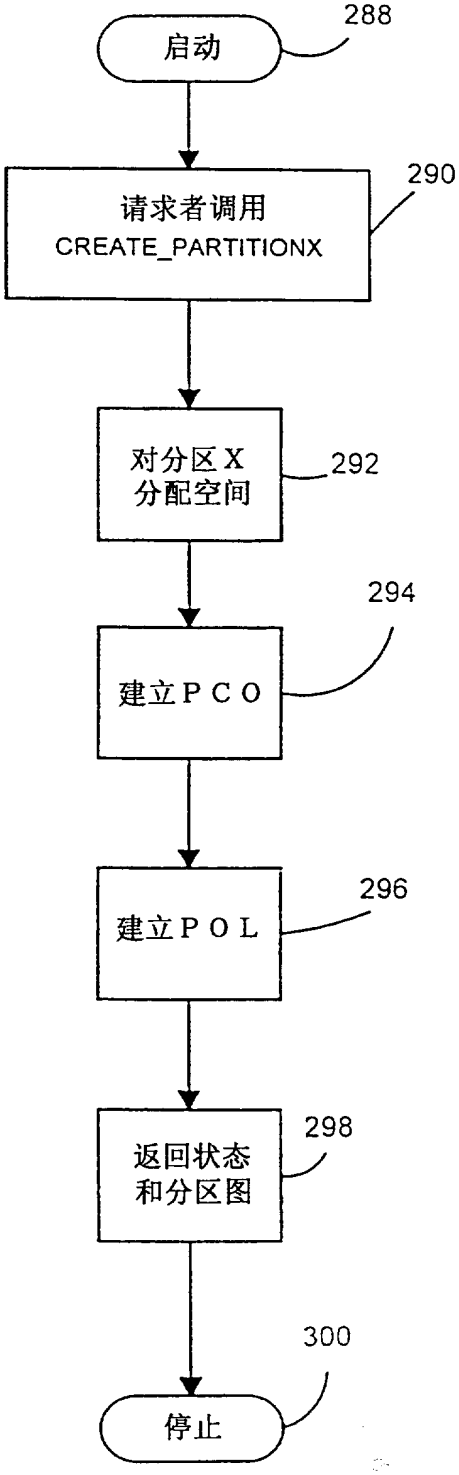

启动
288
请求者调用
CREATE_PARTITIONX
290
对分区Ｘ
分配空间
292
建立ＰＣＯ
294
建立ＰＯＬ
296
返回状态
和分区图
298
停止
300

图　　１５

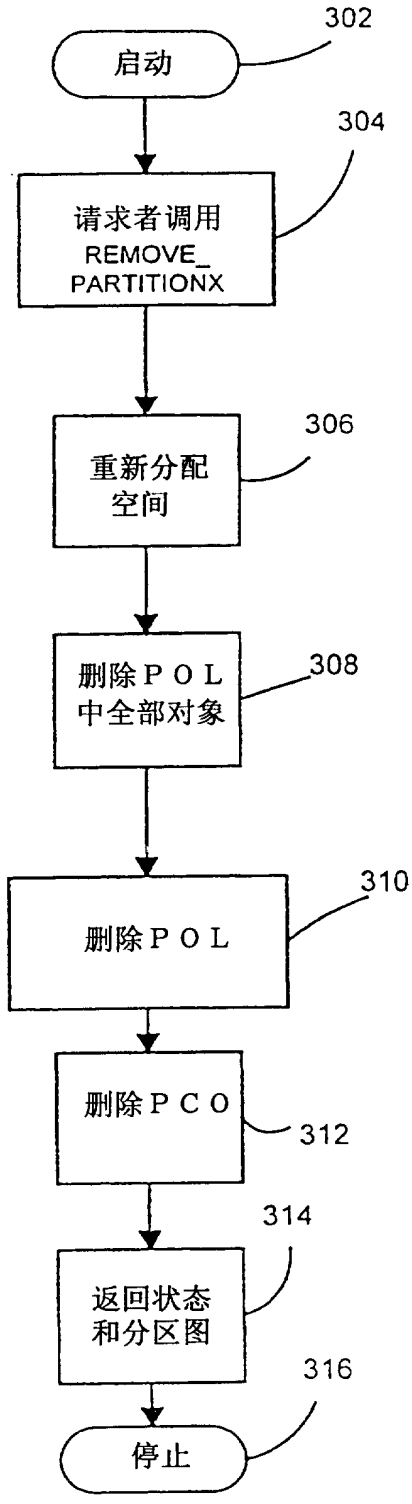

302
启动
304
请求者调用
REMOVE_
PARTITIONX
306
重新分配
空间
308
删除ＰＯＬ
中全部对象
310
删除ＰＯＬ
312
删除ＰＣＯ
314
返回状态
和分区图
316
停止

图　　１６

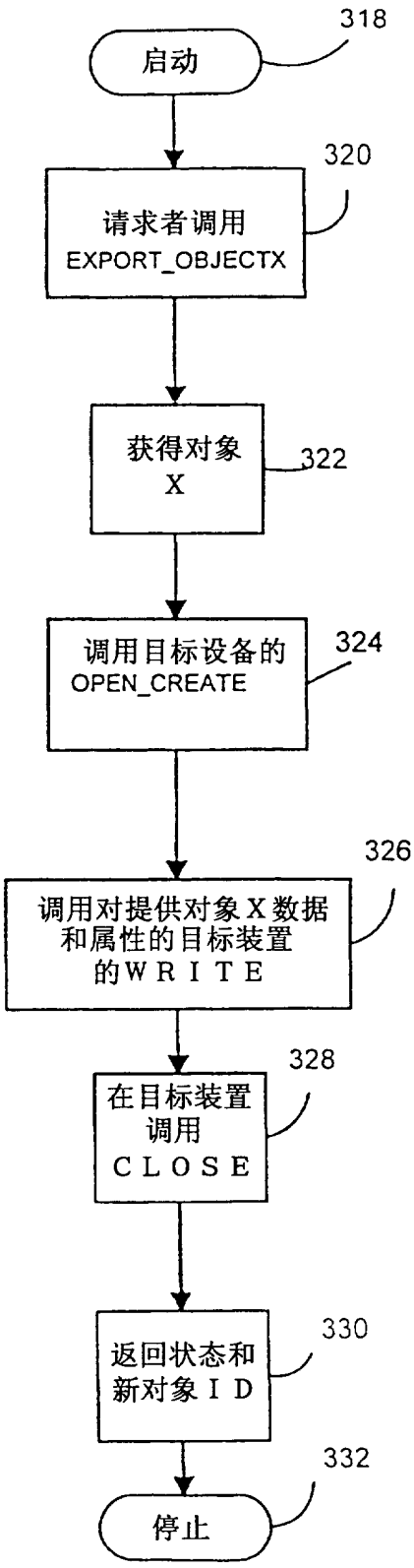

318
启动
320
请求者调用
EXPORT_OBJECTX
322
获得对象
X
324
调用目标设备的
OPEN_CREATE
326
调用对提供对象Ｘ数据
和属性的目标装置
的ＷＲＩＴＥ
328
在目标装置
调用
ＣＬＯＳＥ
330
返回状态和
新对象ＩＤ
332
停止

图　　１７

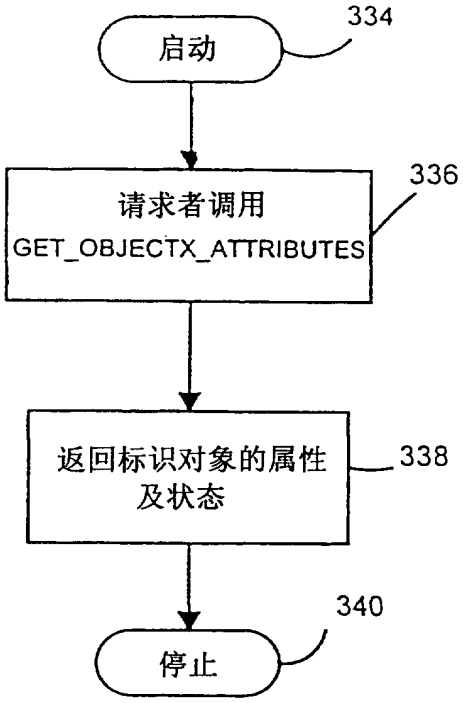

启动
334
请求者调用
GET_OBJECTX_ATTRIBUTES
336
返回标识对象的属性
及状态
338
停止
340

图 18

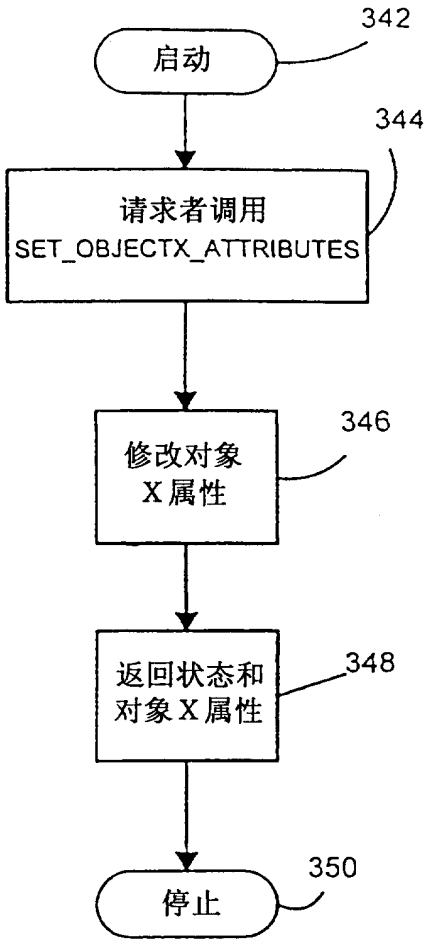

启动
342
请求者调用
SET_OBJECTX_ATTRIBUTES
344
修改对象
X属性
346
返回状态和
对象X属性
348
停止
350

图 19

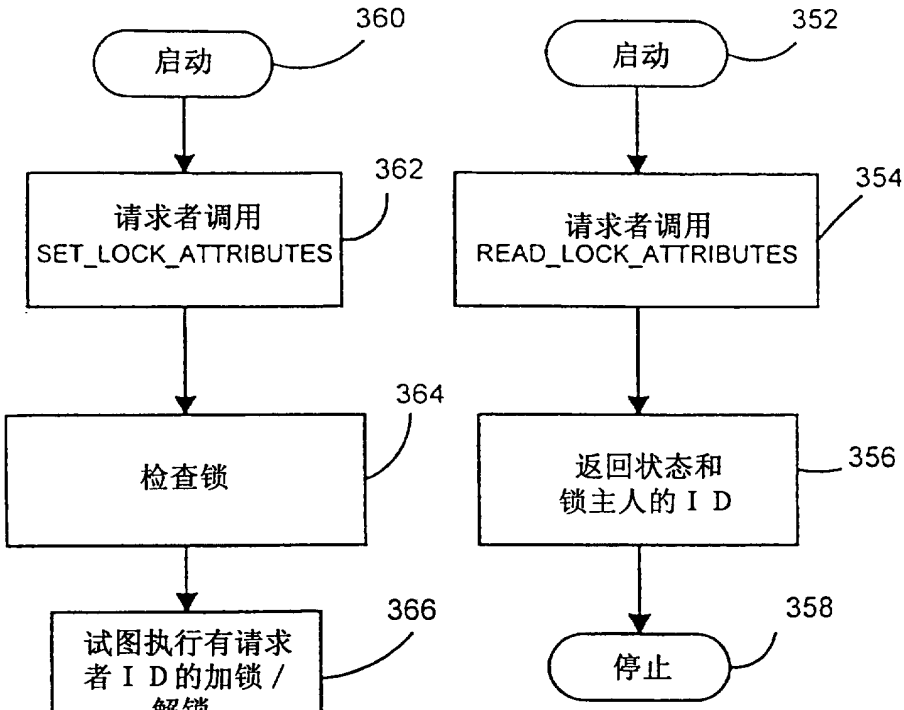

启动
352
请求者调用
READ_LOCK_ATTRIBUTES
354
返回状态和
锁主人的ＩＤ
356
停止
358

图　　２０

启动
360
请求者调用
SET_LOCK_ATTRIBUTES
362
检查锁
364
试图执行有请求者ＩＤ的加锁／解锁
366
返回状态和锁主人的ＩＤ
368
停止
370

图　　２１

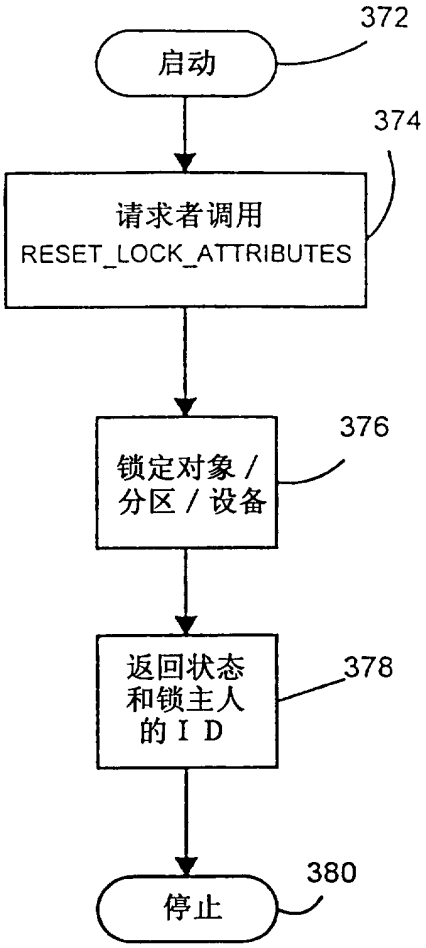

372
启动
374
请求者调用
RESET_LOCK_ATTRIBUTES
376
锁定对象 /
分区 / 设备
378
返回状态
和锁主人
的 I D
380
停止

图 2 2

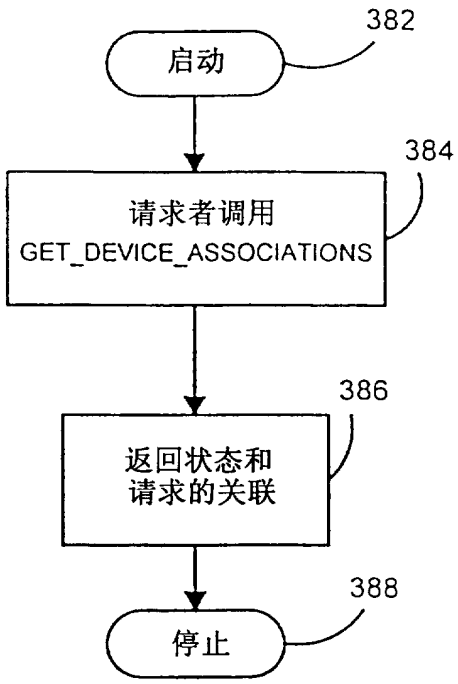

382
启动
384
请求者调用
GET_DEVICE_ASSOCIATIONS
386
返回状态和
请求的关联
388
停止

图 2 3

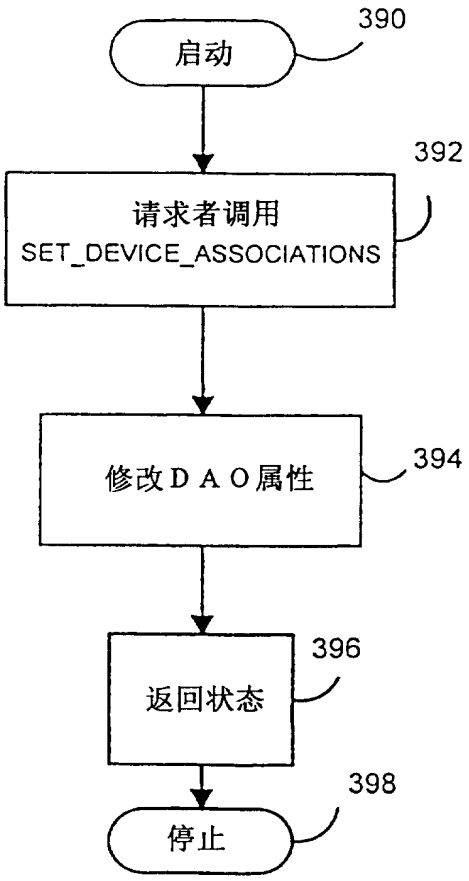

390
启动
392
请求者调用
SET_DEVICE_ASSOCIATIONS
394
修改ＤＡＯ属性
396
返回状态
398
停止

图　　２４

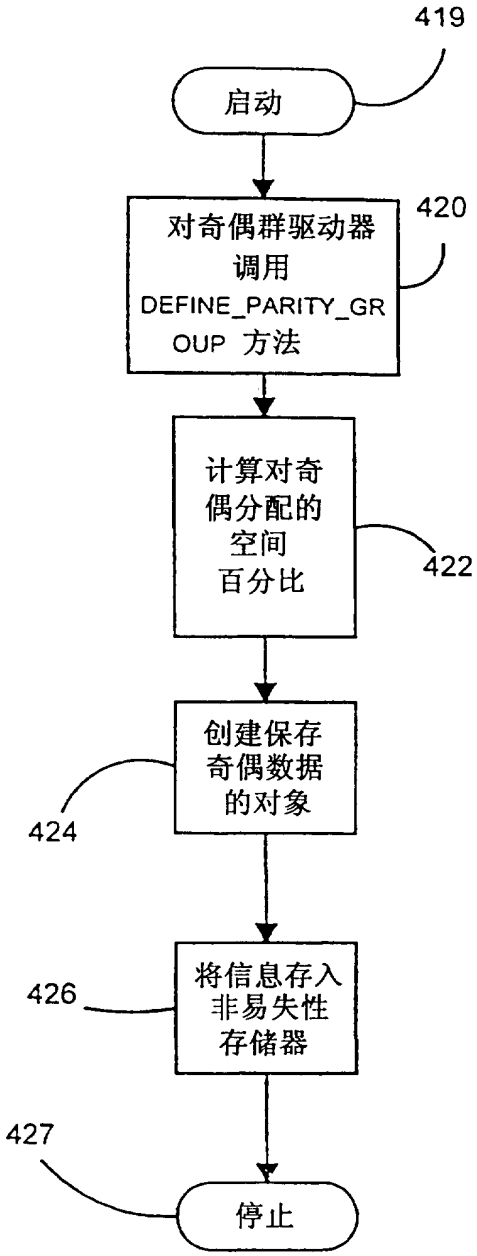

419
启动
420
对奇偶群驱动器
调用
DEFINE_PARITY_GR
OUP 方法
计算对奇
偶分配的
空间
百分比
422
创建保存
奇偶数据
的对象
424
426
将信息存入
非易失性
存储器
427
停止

图　　２８

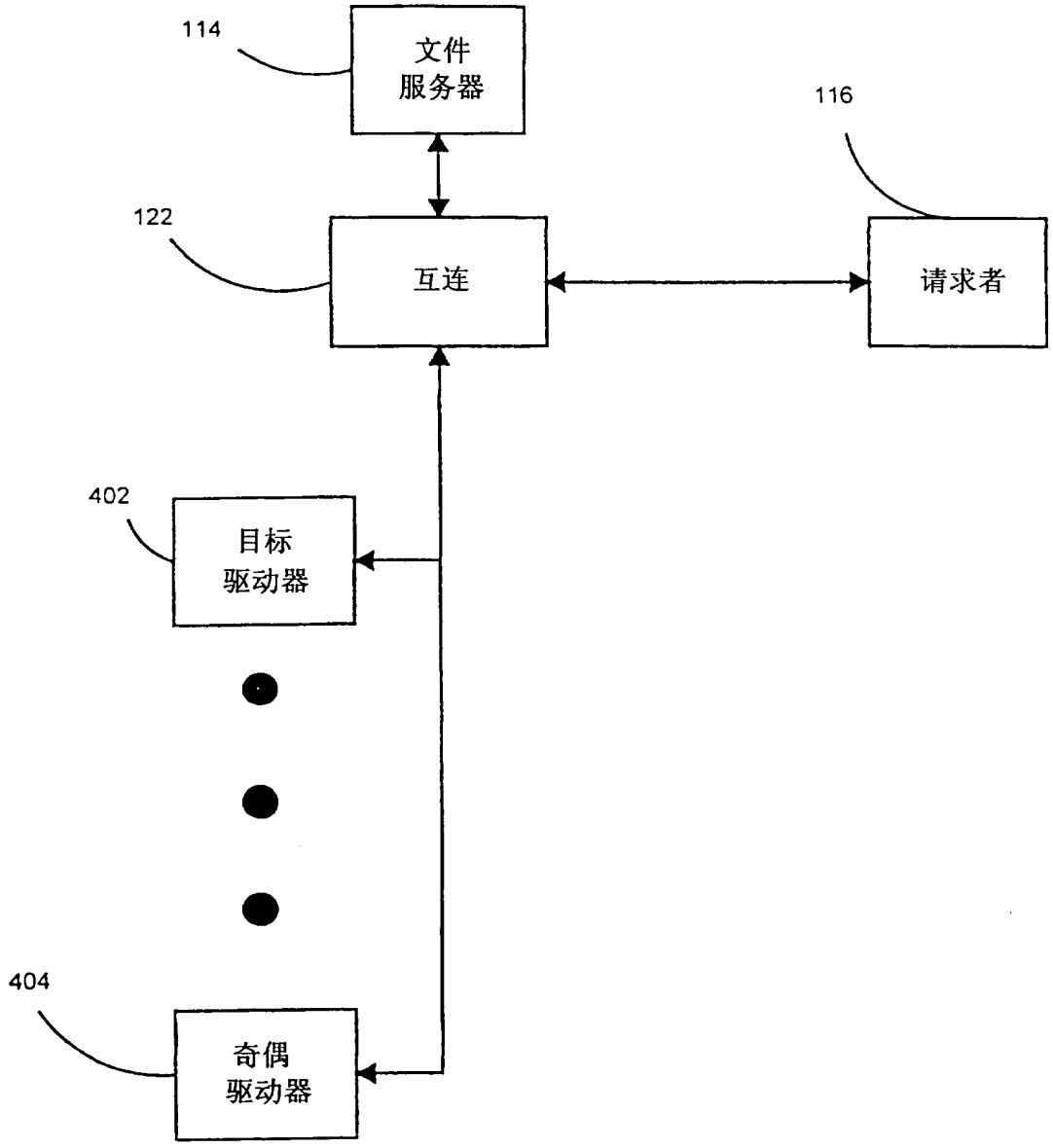


图 25

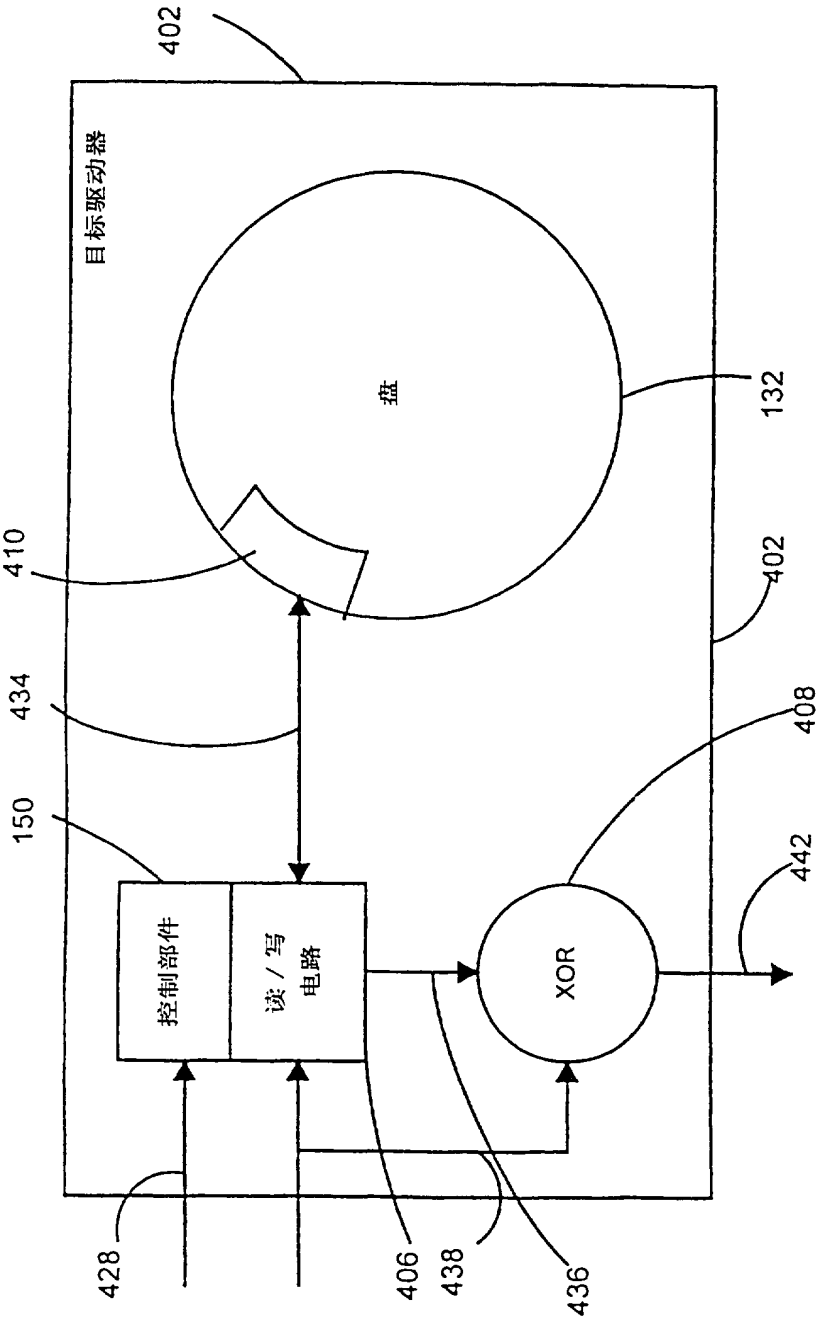

402
目标驱动器
盘
132
410
434
150
控制部件
读/写
电路
XOR
408
442
428
406
438
436
402

图 26

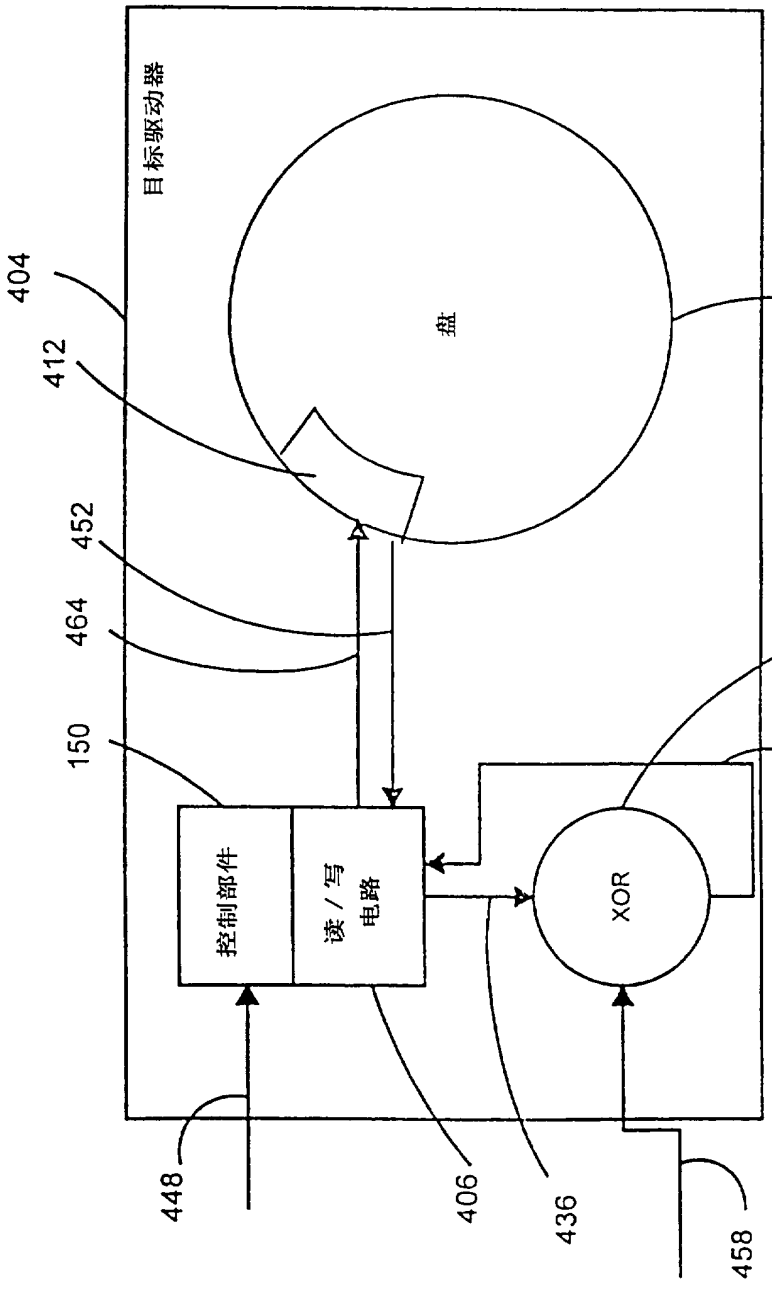

目标驱动器
盘
控制部件
读/写
电路
XOR
404
412
452
464
150
132
408
462
448
406
436
458

图 27

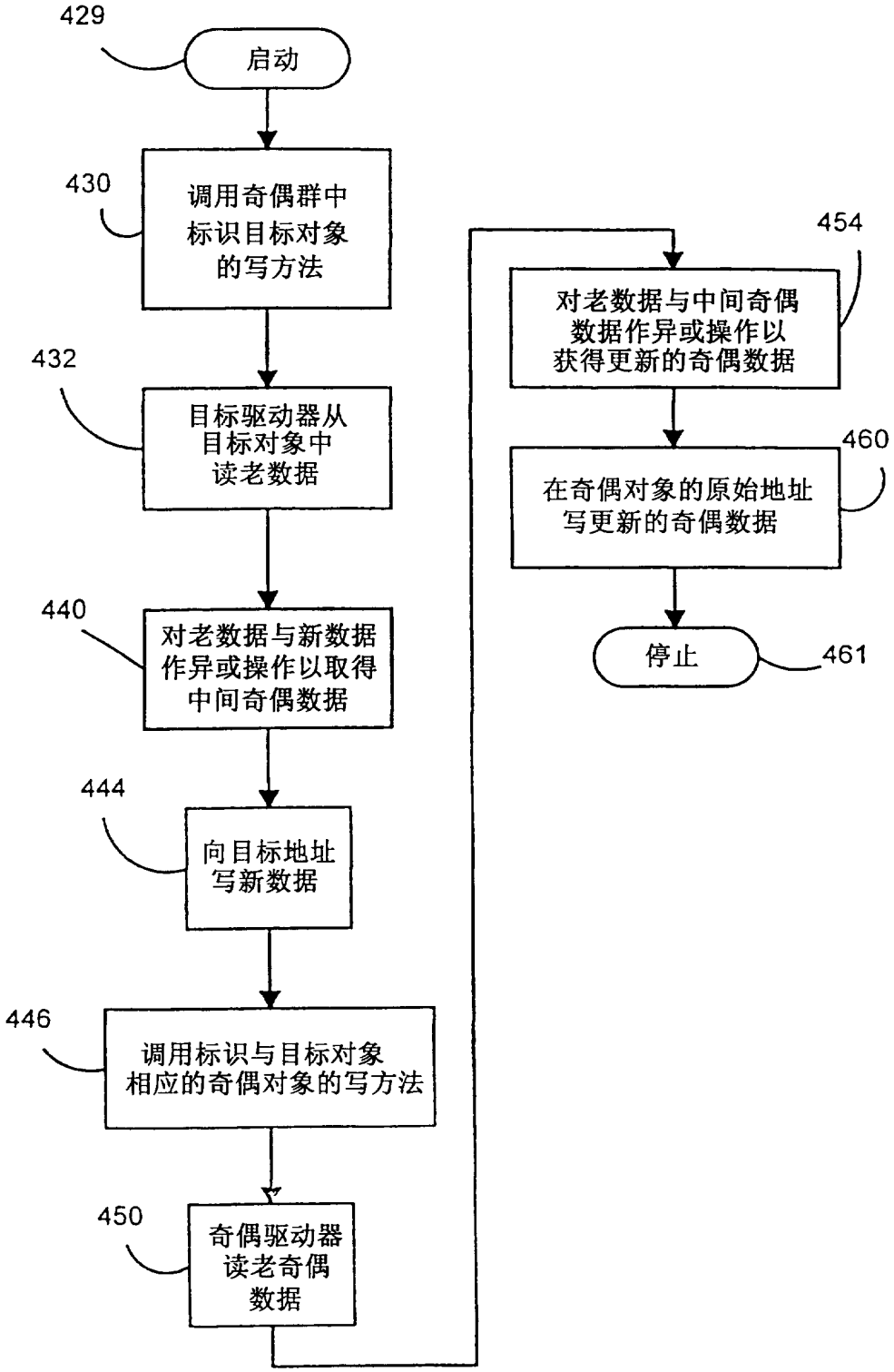

429
启动
430
调用奇偶群中
标识目标对象
的写方法
432
目标驱动器从
目标对象中
读老数据
440
对老数据与新数据
作异或操作以取得
中间奇偶数据
444
向目标地址
写新数据
446
调用标识与目标对象
相应的奇偶对象的写方法
450
奇偶驱动器
读老奇偶
数据
454
对老数据与中间奇偶
数据作异或操作以
获得更新的奇偶数据
460
在奇偶对象的原始地址
写更新的奇偶数据
停止
461

图 29