[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 3/00 (2006.01)
G06F 9/06 (2006.01)
G06F 9/30 (2006.01)
G06F 12/06 (2006.01)

[12] 发明专利说明书

专利号 ZL 03106425.6

[45] 授权公告日 2007 年 3 月 7 日

[11] 授权公告号 CN 1303498C

[22] 申请日 2003.2.25 [21] 申请号 03106425.6

[30] 优先权

[32] 2002. 2.27 [33] US [31] 10/087,672

[73] 专利权人 微软公司

地址 美国华盛顿

[72] 发明人 杰雷德·多纳尔德·阿西姆 杨永奇

[56] 参考文献

JP2000231483A 2000.8.22

JP000285095A 2002.10.13

审查员 吴 敏

[74] 专利代理机构 中国国际贸易促进委员会专利商标事务所

代理人 付建军

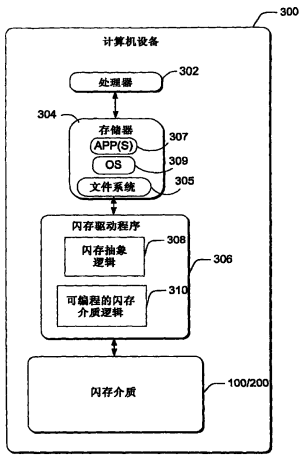
权利要求书 5 页 说明书 23 页 附图 15 页

[54] 发明名称

闪存控制器以及使用闪存介质来存储数据的处理设备

[57] 摘要

本发明公开了一种开放式体系结构闪存驱动程序。 闪存驱动程序体系结构可以作为大多数类型的文件系统和闪存介质之间的接口来运行，不管制造商是谁。 闪存驱动程序包括闪存抽象逻辑模块，该逻辑模块充当一个管理器，用于操作许多不同类型的闪存介质通用的特征。 闪存驱动程序还可以包括可编程的闪存介质逻辑模块，该逻辑模块执行与闪存介质的直接通信中的更具体的操作。 计算机设备的用户/制造商可以最佳地选择与闪存介质逻辑关联的一组可编程的入口点，以确保在计算机的用户/制造商所选择的文件系统和闪存介质之间的均匀而无缝的操作。



1.在文件系统和一个或多个闪存介质之间形成接口的闪存控制器，包括:

闪存抽象逻辑模块，该逻辑模块可以由文件系统调用，以管理闪存操作，而不考虑一个或多个闪存介质的类型如何；以及

闪存介质逻辑模块，被配置为与不同类型的闪存介质进行交互；

其特征在于，闪存抽象逻辑模块调用闪存介质逻辑模块以执行潜在地取决于闪存介质的类型以不同的方式执行的存储器操作。

2.根据权利要求 1 所述的闪存控制器，其特征在于，其中一个闪存操作包括执行与闪存介质关联的磨损调整操作。

3.根据权利要求 1 所述的闪存控制器，其特征在于，其中一个闪存操作包括维护闪存介质的数据完整性。

4.根据权利要求 1 所述的闪存控制器，其特征在于，其中一个闪存操作包括在发生电源故障之后处理与闪存介质关联的数据恢复。

5.根据权利要求 1 所述的闪存控制器，其特征在于，其中一个闪存操作包括映射与闪存介质的物理扇区关联的状态信息，以供文件系统使用。

6.根据权利要求 1 所述的闪存控制器，其特征在于，闪存介质逻辑模块被进一步配置为将从文件系统接收到的命令转换为物理扇区命令，以便发送到闪存介质。

7.根据权利要求 1 所述的闪存控制器，其特征在于，闪存介质逻辑模块是用户可编程的，以向闪存介质写入数据和从中读取和擦除数据。

8.根据权利要求 1 所述的闪存控制器，其特征在于，闪存介质逻辑模块被配置执行与闪存介质关联的错误代码纠正。

9.闪存控制器，包括:

在文件系统和闪存介质之间插入的闪存抽象逻辑模块，被配置为:

(a)将逻辑扇区状态从文件系统映射到闪存介质的物理扇区状态；

以及

(b)维护与操作闪存介质关联的存储器要求。

10.根据权利要求 9 所述的闪存控制器，其特征在于，进一步包括用户可编程的闪存介质逻辑模块，该逻辑模块被配置为向闪存介质写入数据和从中读取和擦除数据。

11.根据权利要求 9 所述的闪存控制器，进一步包括用户可编程的闪存介质逻辑模块，该逻辑模块被配置为从文件系统接收并转换与读取数据和将数据写入到闪存介质关联的特定操作命令。

12.根据权利要求 9 所述的闪存控制器，其特征在于，存储器要求包括管理与闪存介质关联的磨损调整操作。

13.根据权利要求 9 所述的闪存控制器，其特征在于，存储器要求包括维护闪存介质的数据完整性。

14.根据权利要求 9 所述的闪存控制器，其特征在于，存储器要求包括在发生电源故障之后处理与闪存介质关联的数据恢复。

15.根据权利要求 9 所述的闪存控制器，进一步包括闪存介质逻辑模块，该逻辑模块可由用户以编程方式配置为执行与闪存介质关联的错误代码纠正。

16.闪存控制器，包括:

用户可编程的闪存介质逻辑模块，该逻辑模块被配置为向闪存介质写入数据和从中读取和擦除数据；以及

在文件系统和闪存介质之间插入的闪存抽象逻辑模块，用于维护闪存介质的操作的通用要求。

17.根据权利要求 16 所述的闪存控制器，其特征在于，闪存抽象逻辑模块将与某些类型的闪存介质关联的特定命令直接传递到闪存介质逻辑模块，以便进行转换和执行。

18.根据权利要求 16 所述的闪存控制器，其特征在于，闪存抽象逻辑模块是闪存介质逻辑模块和文件系统之间的接口。

19.根据权利要求 16 所述的闪存控制器，其特征在于，通用要求包括维护闪存介质的数据完整性。

20.根据权利要求 16 所述的闪存控制器，其特征在于，通用要求包括管理与闪存介质关联的磨损调整操作。

21.根据权利要求 16 所述的闪存控制器，其特征在于，通用要求包括在发生电源故障之后处理恢复。

22.根据权利要求 16 所述的闪存控制器，其特征在于，闪存介质逻辑模块包括一组可编程的入口点，这些入口点可以由用户实现，以便与所选择的闪存介质的类型连接。

23.使用闪存介质来存储数据的处理设备，包括:

一个文件系统，该系统被配置为控制该处理设备的数据存储;

闪存介质逻辑模块，该逻辑模块被配置为基于物理扇区命令对闪存介质执行物理扇区操作，其特征在于，闪存介质逻辑模块包括一组可编程的入口点，这些入口点可以由用户实现，以便与所选择的闪存介质的类型连接；以及

闪存抽象逻辑模块，被配置为维护运行闪存介质所必需的闪存要求。

24.根据权利要求 23 所述的处理设备，其特征在于，闪存抽象逻辑模块将与某些类型的闪存介质关联的物理逻辑命令直接传递到闪存介质逻辑模块，以便进行转换和执行。

25.根据权利要求 23 所述的处理设备，其特征在于，闪存抽象逻辑模块基于物理扇区地址成为闪存介质逻辑模块和文件系统之间的接口。

26.根据权利要求 23 所述的处理设备，其特征在于，闪存要求包括维护闪存介质的数据完整性。

27.根据权利要求 23 所述的处理设备，其特征在于，闪存要求包括管理与闪存介质关联的磨损调整操作。

28.根据权利要求 23 所述的处理设备，其特征在于，闪存要求包括在发生电源故障之后处理恢复。

29.根据权利要求 23 所述的处理设备，其特征在于，要求对于许多不同的闪存介质是通用的。

30.根据权利要求 23 所述的处理设备，其特征在于，闪存介质逻辑模块包括一组可编程的入口点，这些入口点可以由用户实现，以便执行与处理设备中使用的闪存介质的类型相关的错误代码纠正。

31.根据权利要求 23 所述的处理设备，其特征在于，闪存介质逻辑模块用于将从闪存抽象逻辑模块或文件系统接收到的逻辑命令转换为物理扇区命令。

32.根据权利要求 23 所述的处理设备，其特征在于，物理扇区操作包括读取、写入和执行与闪存介质关联的错误代码纠正命令。

33.在使用闪存介质来存储数据的处理设备中，用于驱动闪存介质的方法，包括:

从文件系统读取和写入命令并将它们转换为物理扇区命令;

在闪存抽象逻辑模块中管理与操作闪存介质关联的规则;

发出一组可编程的入口点，这些入口点可以由用户可选地选择，以便与处理设备中使用的闪存介质的类型连接; 以及

将物理扇区命令从闪存介质逻辑模块直接发送到闪存介质。

34.根据权利要求 33 所述的方法，其特征在于，其中一个规则包括维护闪存介质的数据完整性。

35.根据权利要求 33 所述的方法，其特征在于，其中一个规则包括管理与闪存介质关联的磨损调整操作。

36.根据权利要求 33 所述的方法，其特征在于，其中一个规则包括在发生电源故障之后处理介质的恢复。

37.根据权利要求 33 所述的方法，进一步包括发出一组可编程的入口点，这些入口点可以由用户实现，以便执行与处理设备中使用的闪存介质的类型相关的错误代码纠正。

38.根据权利要求 33 所述的方法，进一步包括接收来自文件系统的读取和写入命令。

39.闪存控制器，包括一个模块，用于指示所述闪存控制器在从许多不同文件系统中选择的一个文件系统和从许多不同闪存介质中选择的一个闪存介质之间提供一个接口。

40.闪存控制器，包括下列模块，用于指示所述闪存控制器：

在从许多不同文件系统中选择的一个文件系统和从许多不同闪存介质中选择的一个闪存介质之间提供一个接口；以及

管理闪存抽象逻辑模块中的许多不同闪存介质通用的一组特征。

41.闪存控制器，包括下列模块，用于指示所述闪存控制器：

在从许多不同文件系统中选择的一个文件系统和从许多不同闪存介质中选择的一个闪存介质之间提供一个接口；

管理闪存抽象逻辑模块中的许多不同闪存介质通用的一组特征；

提供可编程的入口点，这些入口点可以由用户可选地选择，以便与所选择的闪存介质的类型连接。

闪存控制器以及使用闪存介质来存储数据的处理设备

技术领域

本发明涉及闪存驱动程序，更具体来说，涉及开放式体系结构闪存驱动程序。

背景技术

闪存是非易失性存储介质，它可以在没有电源的情况下保留信息，并可以将信息删除和重新编程。许多便携式计算机设备，如膝上型计算机、便携数字助手 (PDA)、便携通信/计算机设备，以及许多其他类型的相关的设备使用闪存作为用于进行信息存储的主要介质，因为闪存在计算机设备中占用的空间非常小，并且不要求连续的电源便能保留其存储内容。此外，闪存还可以抵抗许多便携式计算机设备时常发生的摇动和意外摔落。因此，闪存越来越成为大多数便携式计算机设备所选择的存储介质。

大多数闪存制造商与通常被称为闪存驱动程序的专用控制器一起销售闪存。通常，这些闪存驱动程序与其他制造商制造的闪存不兼容。这就降低了便携式计算机设备的制造商的灵活性，因为制造部署的操作系统和/或文件系统常常与特定的专用闪存驱动程序紧密联系。更改闪存介质常常要求对文件系统进行修改，以确保与那些与闪存介质关联的特定闪存驱动程序的兼容性。在某些情况下，某些操作系统以及文件系统与某些闪存介质不兼容，从而由于缺乏驱动程序兼容性可能会使便携式计算机设备的某些制造商在某些程度上被局限于特定的闪存。

发明内容

下面将描述开放式体系结构闪存驱动程序。闪存驱动程序可以作为大多数类型的文件系统和闪存介质之间的接口来运行，不管制造商是谁。在描述的一个实施例中，闪存驱动程序在由计算机执行时会在

文件系统和一个或多个闪存介质之间提供一个接口。闪存驱动程序使用闪存抽象逻辑，该逻辑可以由文件系统调用，以管理闪存操作，而不考虑一个或多个闪存介质的类型如何。闪存驱动程序还使用被配置为与不同类型的闪存介质进行交互的闪存介质逻辑。闪存抽象逻辑调用闪存介质逻辑以执行潜在地取决于闪存介质的类型以不同的方式执行的存储器操作。

附图说明

下面将参考附图进行详细的说明。在图中，引用号码的最左边的数字标识引用号码在其中首次出现的图。

图 1 说明了 NAND 闪存介质的逻辑表示。

图 2 说明了 NOR 闪存介质的逻辑表示。

图 3 说明了使用一个或多个闪存设备来存储信息的计算机设备的相关组件。

图 4 说明了闪存抽象逻辑的方框图。

图 5 说明了闪存介质逻辑的示范方框图。

图 6A 显示了用于存储逻辑扇区地址和物理扇区地址之间的对应关系的数据结构。

图 6B 显示了与图 6A 中的数据结构相同的数据结构，只不过其内容已经更新。

图 7 说明了用于当文件系统向闪存驱动程序发出写入请求时跟踪闪存介质上的数据的过程。

图 8 说明了保护诸如如图 6A 和 6B 中所示的数据结构之类的易失性数据结构中存储的逻辑到物理扇区地址信息的映射的过程。

图 9 说明了逻辑扇区地址可以存储以便防止电源故障的闪存介质内的位置。

图 10 说明了用于跟踪闪存介质中存储的数据的动态查询数据结构。

图 11 说明了动态地分配用于跟踪闪存介质上的数据的查询数据结构的过程。

图 12 是被闪存驱动程序作为连续的圆周看待的闪存介质的示意图。

图 13 描述了作为连续圆周看待的介质的另一个图表。

图 14 说明了被扇区管理器用来确定闪存驱动程序在介质上存储数据的下一个可用的空闲扇区位置的过程。

图 15 说明了作为连续圆周看待的介质的另一个视图。

图 16 是说明了被压缩工具用来回收扇区的过程的流程图。

图 17 显示了图 16 中说明的过程的一个示范结果。

图 18 说明了为更好地支持闪存驱动程序实现的过程和技术划分的 NOR 闪存介质的逻辑表示。

具体实施方式

下面的讨论针对闪存驱动程序。下面将有针对性地对主题进行描述以满足法定的要求。然而，描述本身不仅限于本专利的范围。发明人认为权利要求书所述的主题还可以以其他方式实施，以与其他当前或未来的技术一起包括不同的元素或类似于本文档中描述的元素的元素的组合。

概述

本讨论假设读者熟悉闪存介质的基本操作原理。虽然如此，还是提供了对两种常见的非易失性随机存取存储器 — NAND 和 NOR 闪存介质 — 的一般介绍，以便更好地理解此处描述的示范实施例。之所以选择这两个示例闪存介质，是因为它们当前比较流行，但对它们的描述不仅将所描述的实施例限于这些类型的闪存介质。其他以电的方式可擦除的和可编程的只读存储器 (EEPROM) 也行。为便于说明，在本详细描述中使用的大多数示例中，数据结构中显示的数字是十进制格式。

通用闪存介质操作特征

图 1 和图 2 分别说明了示例 NAND 和 NOR 闪存介质 100、200 的逻辑表示。两种介质分别具有对每一种介质通用的操作特征，不管制造商是谁。例如，请参看图 1，NAND 闪存介质通常分为相邻

的区块（0、1 到 N）。每一个区块 0、1、2 等等进一步细分为 K 个扇区 102。标准商业性 NAND 闪存介质通常每个区块都包含 8、16 或 32 个扇区。然而，制造商不同，区块和扇区的数量也会不同。某些制造商将扇区"称为"页"。此处使用的两个术语是等效的，并可互换。

每一个扇区 102 进一步被分成两个不同的部分，用于存储信息的数据区域 103，和用于存储诸如错误纠正代码 (ECC) 之类的多余信息的备用区域 104。数据区域 103 的大小通常为 512 字节，但是随着制造商的不同，也可能多一些，也可能少一些。采用 512 个字节时，闪存介质可使大多数文件系统将该介质作为诸如固定磁盘（硬盘）之类的非易失性存储器设备对待。此处使用的 RAM 通常是指诸如 DRAM、SRAM、VRAM、VDO 等等之类的随机存取存储器系列。通常，备用区域 104 的大小为 16 字节，用于 NAND 闪存介质设备的多余信息存储。此外，也可以选择其他大小，可以稍大一些，也可以稍小一些。在大多数情况下，备用区域 104 用于存储错误纠正代码和状态信息。

NOR 存储器介质 200 与 NAND 存储器介质不同的是，区块不细分为物理扇区。与 RAM 相似的是，NOR 存储器介质的区块内存储的每一个字节是可分别寻址的。然而，在实际中，NOR 存储器介质上的区块可以从逻辑上细分为物理扇区，并附带备用区域。

除总体布局和操作比较以外，闪存设备的某些通用电学特征（此处也被称为"存储器要求"或"规则"）可以概述如下：

1. 对扇区的写入操作可以将单个位从逻辑"1"更改为逻辑"0"，但不能将逻辑"0"更改为逻辑"1"（下面的第 2 种情况除外）；

2. 擦除一个区块会将该区块中的所有位设置为逻辑"1"；

3. 在没有擦除一个区块内的所有扇区/字节的情况下要擦除同一区块中的单个扇区/字节/位通常是不可能的；

4. 区块的擦除生存期是有限的，大约在 100,000 到 1,000,000 周之间；

5. NAND 闪存设备使用 ECC 来防止由于漏泄电流造成的

数据损坏；以及

6. 读取操作不计入写入/擦除生存期内。

闪存驱动程序的体系结构

图 3 说明了使用一个或多个闪存设备来存储信息的计算机设备 300 的相关组件。通常，可以使用各种不同的一般用途或特殊用途计算系统配置来作为计算机设备 300，包括但不仅限于个人计算机、服务器计算机、手提或膝上型设备、便携通信设备、多处理器系统、基于微处理器的系统、可编程的消费类电子产品、游戏系统、多媒体系统、上述示例设备和/或系统的任何一种组合等等。

计算机设备 300 通常包括处理器 302、存储器 304 和闪存介质 100/200。计算机设备 300 可以包括上述元素中的多个元素。诸如电源、键盘、触摸板、I/O 接口、显示器、LED、音频生成器、振动设备等等之类的其他元素未显示，但它们也可以轻松地成为示范计算机设备 300 的组成部分。

存储器 304 通常包括易失性存储器（例如，RAM）和非易失性存储器（例如，ROM、PCMCIA 卡等等）。在下面描述的大多数实施例中，存储器 304 被用作计算机设备 302 的缓存，使得应用程序数据被很快地访问到，而不必永久地将数据存储在诸如闪存介质 100/200 之类的非易失性存储器上。

操作系统 309 驻留在存储器 304 中并在处理器 302 上执行。一个示例操作系统实施例包括 Microsoft Corportaion 提供的 Windows® CE 操作系统，但也可以选择其他操作系统，如 DOS、UNIX 等。为便于说明，诸如操作系统之类的程序和其他可执行程序组件此处作为不连续的区块来显示，虽然这样的程序和组件在不同的时间驻留在计算机的不同的存储组件中，并由计算机设备 300 的处理器来执行。

一个或多个应用程序 307 被加载到存储器 304 中，并在操作系统 309 上运行。应用程序的示例包括，但不仅限于，电子邮件程序、文字处理程序、电子表格程序、因特网浏览器程序等等。

也在操作系统 309 上运行的文件系统 305 也加载到存储器 304 中。文件系统 305 通常负责管理数据向诸如硬盘驱动器和此示范实施例闪存介质 100/200 之类的存储器设备的存储和检索。大多数文件系统 305 根据文件系统 305 运行所在的操作系统的约定的逻辑级别访问和存储信息。文件系统 305 也可以成为操作系统 309 的组成部分或作为单独的逻辑模块以代码形式嵌入其中。

闪存驱动程序 306 作为文件系统 305 和闪存介质 100/200 之间的直接接口来运行。闪存驱动程序 306 能使计算机设备 300 通过文件系统 305 控制闪存介质 100/200 并最终发送/检索数据。然而，如下文更详细地描述的，闪存驱动程序 306 不仅负责读/写操作。闪存驱动程序 306 维护数据完整性，执行闪存介质的磨损调整，在计算机设备 300 的电源中断期间最大限度地降低数据丢失，并允许计算机设备 300 的 OEM 支持它们相应的闪存设备，不管制造商是谁。闪存驱动程序 306 不依赖于文件系统。这意味着闪存驱动程序 306 支持许多不同类型的文件系统，如文件分配数据结构文件系统 (FAT 16)、(FAT32) 和其他文件系统。此外，闪存驱动程序 306 也不依赖于闪存介质，这同样意味着驱动程序 306 支持闪存设备，不管该闪存设备的制造商是谁。即，闪存驱动程序 306 具有在闪存介质上读取/写入/擦除数据的能力，并可以支持大多数闪存设备，如果不是全部的话。

在示范实施例中，闪存驱动程序 306 作为操作系统 309 内的组件，当它执行时，充当文件系统 305 和闪存介质 100/200 之间的逻辑接口模块。闪存驱动程序 306 以一个单独的框 306 的方式来显示，为便于演示闪存驱动程序充当接口的示例。虽然如此，闪存驱动程序 306 也可以驻留在其他应用程序中，作为文件系统 305 的组成部分或独立地作为与硬件/固件设备一起执行的计算机可读的介质上单独的代码。

在一个实施例中，闪存驱动程序 306 包括：闪存抽象逻辑 308 和可编程的闪存介质逻辑 310。闪存抽象逻辑 308 和可编程的介质逻辑

310 是可以支持闪存驱动程序 306 执行的各种功能的编码指令。虽然显示的示范实施例包括这两个元素，但是可以选择闪存抽象逻辑 308 和闪存介质逻辑 310 中每一个逻辑中的各种功能，以执行下面描述的某些更具体的实施例。因此当描述的实施例显示了两个不同的逻辑层 308/310 时，可以实现下面描述的许多技术，而不一定要求这两层逻辑中的全部或部分功能。此外，实现技术时也可以不必有下面描述的准确的职责划分。

在一个实施例中，闪存抽象逻辑 308 管理对闪存介质普遍通用的那些操作特征。这些通用的存储器要求包括磨损调整、维护数据完整性，以及在发生电源故障之后处理数据恢复。此外，闪存抽象逻辑 308 负责将闪存介质 100/200 上的物理扇区域上存储的信息映射到与文件系统 305 关联的逻辑扇区域。即，闪存抽象逻辑 308 跟踪从逻辑到物理扇区地址和/或从物理到逻辑扇区地址发出的数据。驱动程序 306 在执行读/写操作时都使用逻辑到物理扇区地址。驱动程序 306 在驱动程序初始化过程中在创建查询表（下面将作介绍）时从物理到逻辑扇区地址出发。依赖于某些类型的闪存介质的文件系统发出的某些比较具体的命令被直接发送到闪存介质逻辑 310 以便执行和转换。如此，闪存抽象逻辑 308 充当那些通用操作的管理器，这些通用操作对闪存介质是通用的，不管介质的制造商是谁，如磨损调整、维护数据完整性、在发生电源故障之后处理数据恢复。

图 4 说明了闪存抽象逻辑的示范方框图。闪存抽象逻辑 308 包括扇区管理器 402、逻辑到物理扇区映射模块 404 和压缩工具 406。简而言之，扇区管理器 402 提供一个指针，指向可用的扇区，即，“空闲”并可用于接收新数据。逻辑到物理扇区映射模块 404 在数据从逻辑扇区寻址的文件系统域向物理扇区寻址的闪存介质域出发时对数据进行管理。压缩工具 406 提供一个机制，用于清除数据区块（在业界通常也称为“擦除”），以确保有足够多的空闲扇区可用于写入数据。此外，压缩工具 406 还帮助驱动程序 306 系统执行一致而均匀的磨损调整。下面将比较详细地介绍所有这些元素。

再回到图 3，使用闪存介质逻辑 310，将从闪存抽象逻辑 308 或文件系统 305 接收到的逻辑命令转换为物理扇区命令，以便发送到闪存介质 100/200。例如，闪存介质逻辑 310 从闪存介质读取和擦除数据和/或向其中写入数据。闪存介质逻辑 310 在必要时也负责执行 ECC。在一个实施例中，闪存介质逻辑 310 是可编程的，以允许用户满足特定制造商的特定闪存介质要求。因此，闪存介质逻辑 310 被配置为处理与控制闪存介质 100/200 的物理方面关联的特定细节、ECC、和特定的命令。

图 5 说明了闪存介质逻辑 310 的示范方框图。如图所示，闪存介质逻辑 310 包括可编程的入口点模块 502、I/O 模块 504 和 ECC 模块 506。可编程的入口点模块 502 定义了一组编程接口，以便在闪存抽象逻辑 308 和闪存介质 100/200 之间通信。换句话说，可编程的入口点允许计算机设备 300 的制造商对闪存介质逻辑 310 进行编程，以便与计算机设备 300 中使用的实际闪存介质 100/200 连接。I/O 模块 504 包含发送到闪存介质 100/200 的读取/写入/擦除命令所必需的特定代码。用户可以对 ECC 模块 506 进行编程，以便根据用户选择的任何特定的 ECC 算法运行。

跟踪数据

文件系统 305 使用逻辑扇区寻址，以读取和存储闪存介质 100/200 上的信息。逻辑扇区地址是文件系统从中读取和向其中写入数据的地址位置。它们是“逻辑”，因为它们是相对于文件系统的。在现实中，数据可以存储在闪存介质 100/200 上的完全不同的物理位置。这些物理位置被称为物理扇区地址。

闪存驱动程序 306 负责将所有的逻辑扇区地址请求（即，读取 & 写入）链接到物理扇区地址请求。链接逻辑到物理扇区地址的过程此处也被称为“映射”。从逻辑到物理扇区地址的映射允许闪存驱动程序 306 在决定将数据存储在闪存介质 100/200 的什么位置时具有最大的灵活性。逻辑到物理扇区映射模块 404 允许数据灵活地分配到闪存介质上的任何物理位置，从而使诸如磨损调整和从电源故障中恢复之

类的其他任务有较高的效率。它还允许文件系统 305 以所设计的方式存储数据，而不必知道数据实际上是以不同的方式存储在闪存介质上的。

图 6A 显示了闪存驱动程序 306 生成的数据结构（即，表）600A 的示范实施例。数据结构 600A 存储在存储器 304 的易失性部分，例如，RAM。数据结构 600A 包括具有对应的逻辑扇区地址 604 的物理扇区地址 602。下面将参考图 7 介绍表 600A 是如何生成的。

图 7 说明了用于当文件系统 305 向闪存驱动程序 306 发出写入请求时跟踪闪存介质 100/200 上的数据的过程 700。过程 700 包括步骤 702-718。请参看图 6A 和 7，在步骤 702 中，闪存抽象逻辑 308 接收到将数据写入到指定的逻辑扇区地址 604 的请求。

在步骤 704 中，扇区管理器 402 确定闪存介质 100/200 上可以接受与写入请求关联的数据的空闲物理扇区地址位置（下面将比较详细地讲述扇区管理器 402 如何选择物理扇区地址）。空闲的物理扇区是不必首先擦除就可以接受数据的任何扇区。一旦扇区管理器 402 接收与空闲物理扇区位置关联的物理扇区地址，逻辑到物理扇区映射模块 404 就将物理扇区地址分配到写入请求指定的逻辑扇区地址 604，从而形成对应的关系。例如，物理扇区地址 0 到 N 可以分配到任何逻辑扇区地址 0 到 N。

接下来，在步骤 706 中，逻辑到物理扇区映射模块 404 在诸如存储器 305 中的示范表 600A 之类的数据结构中存储物理扇区地址到逻辑扇区地址的对应关系。如图示范数据结构 600A 所示，三个逻辑扇区地址 604 被分配到对应的物理扇区地址 602。

接下来，在步骤 708 中，与逻辑扇区地址写入请求关联的数据存储在步骤 704 中分配的物理扇区地址位置中的闪存介质 100/200 上。例如，数据将存储在介质 100/200 上的对应于逻辑扇区地址 11 的物理扇区地址位置 0。

现在，在步骤 710 中，假设文件系统 305 发出了另一个写入请求，但在这种情况下，是为了修改与以前步骤 702 中发出的逻辑扇区

地址关联的数据。然后，闪存驱动程序 306 分别执行步骤 712 到 714，它们与上文描述的步骤 704 到 708 完全相同。

然而，在步骤 718 中，在与步骤 710 关联的更新的数据成功地存储在闪存介质 100/200 之后，逻辑到物理扇区映射模块 404 将在步骤 704 中分配的旧的物理扇区地址标记为“脏的”。在新数据被写入介质 100/200 之后旧数据被标记为脏的，如此，在写入操作的中间发生电源故障的情况下，逻辑到物理扇区映射模块 404 将不会丢失旧的数据。可能会丢失新的或从步骤 702 或 710 更新的数据，但由于没有必要执行擦除操作，因此在发生电源故障的情况下新的或被修改的数据中只有一项丢失。

图 6B 显示了与数据结构 600A 相同的数据结构 600B，只不过其内容已经更新。在此示例中，文件系统 305 具有与逻辑扇区地址 11 关联的更新数据。相应地，闪存驱动程序 306 将逻辑扇区地址 11 重新分配到物理扇区地址 3，并将这两个地址之间的重新分配的对应关系存储在数据结构 600B 中。如数据结构 600B 所示，逻辑扇区 11 的内容实际上被写入物理扇区地址 3，数据内容被成功地写入到物理扇区地址 3 之后，扇区 0 的内容被标记为“脏的”，关于这一点，参考步骤 710-718 进行了描述。

当以前存储的数据被文件系统 305 更新时重新分配逻辑到物理扇区地址的此过程，允许写入操作不必等待移动全部数据区块便可以执行，并执行擦除操作。因此，过程 700 允许数据结构得到很快的更新，然后可以在实际物理介质 100/200 上执行物理写入操作。闪存抽象逻辑 308 使用诸如 600A/600B 之类的数据结构正确地维护逻辑到物理的映射关系。

当文件系统 305 发出了读取请求时，闪存抽象逻辑 308，通过逻辑到物理映射模块 404，检索数据结构 600A/600B，以获得与关联于读取请求的逻辑扇区具有对应关系的地址物理扇区地址。然后，闪存介质逻辑 310 使用该物理扇区地址作为一个基础，以便将与读取请求关联的数据发送回文件系统 305。文件系统 305 不必知道其对逻辑扇

区地址的请求实际映射到物理扇区地址。

电源中断保护

写入操作在与区块级别相对的扇区级别上执行，从而在发生电源故障的情况下最大限度地降低数据丢失的可能性。用于存储数据的扇区是相对于大多数文件系统 305 的最细微的粒度级别。因此，如果闪存驱动程序 306 每个扇区地运行，在发生电源故障的过程中数据丢失的可能性会降低。

如上文所述，数据结构 600A、600B 存储在存储器 304 中，该存储器在一个示范实施例中通常是一个易失性存储器设备，在发生电源故障的情况下将会被完全擦除。为保护闪存介质 100/200 上的数据完整性，数据结构 600A/600B 中存储的逻辑到物理映射信息被备份在闪存介质上。

在一个示范实施例中，为降低与在闪存介质 100/200 上存储整个数据结构关联的开销，逻辑扇区地址存储在介质的备用区域 104，并带有逻辑扇区地址与其具有对应关系的每一个物理扇区。

图 8 说明了保护诸如示范数据结构 600A 和 600B 之类的易失性数据结构中存储的逻辑到物理扇区地址信息的映射的过程 800。过程 800 包括步骤 802-814。描述该过程的顺序不能解释为限制。此外，该过程可以以任何适当的硬件、软件、固件或它们的组合来实现。在步骤 802 中，与实际数据关联的逻辑扇区地址存储在分配到逻辑扇区地址的物理扇区地址上的闪存介质 100/200 的物理扇区中。在 NAND 闪存介质 100 的情况下，逻辑扇区地址存储在介质的备用区域 104。使用此方案，逻辑到物理扇区映射信息以反向搜索格式来存储。因此，在发生电源故障之后，必须扫描介质上的每一个物理扇区的备用区域，确定对应的逻辑扇区地址，然后相应地更新存储器中的查询表。图 9 说明了介质 100/200 中的可以存储逻辑扇区地址的位置。正如前面所提到的，NOR 闪存的区块可以从逻辑上细分为物理扇区，每一个物理扇区都带有一个备用区域（与 NAND 相似）使用此技术，逻辑扇区地址存储在每一个物理扇区的备用区域，与 NAND

闪存使用的过程相似（图 15 中显示为空间 1504，下面将参考图 15 对其进行描述）。

在发生电源中断以及数据结构 600A、600B 丢失的情况下，正如图 8 的判断步骤 804 的“是”分支所指出的，然后，闪存抽象逻辑 308 使用闪存介质逻辑 310 扫描闪存介质，以定位与每一个物理地址中的数据一起存储的逻辑扇区地址（参见图 9），正如步骤 806 所指出的。在步骤 808 中，包含了数据的物理扇区地址被重新分配到与介质上的数据在一起的逻辑扇区地址。由于物理和逻辑扇区地址被恢复，因此，它们被存储回数据结构 600A、600B，闪存介质逻辑 310 进入包含数据的下一个扇区，正如步骤 812 所指出的。步骤 806-812 不断重复，直到包含数据的所有扇区都已经被扫描，数据结构被恢复。通常，这种情况将在初始化计算机设备 300 时发生。

相应地，当发生电源故障时，过程 800 能使闪存抽象逻辑 308 扫描介质 100/200，并在诸如示范数据结构 600 之类的数据结构中重建逻辑到物理映射。过程 800 可以确保映射信息在发生电源故障期间不会丢失，并且可以保持数据的完整性。

用于跟踪数据的动态查询数据结构

图 10 说明了用于跟踪闪存介质 100/200 中存储的数据的动态查询数据结构 1000。数据结构 1000 包括主数据结构 1002 和一个或多个辅助数据结构 1004、1006。数据结构将由闪存驱动程序 306 生成和维护。数据结构存储在存储器 304 的易失性部分。一个或多个辅助表 1004、1006 包含逻辑到物理扇区地址的映射。正如下文将讲述的，辅助数据结构 1004、1006 中每一个数据结构都具有预先确定的映射容量。主数据结构 1002 包括一个指针，指向一个或多个辅助数据结构 1004、1006 中的每一个数据结构。每一个辅助数据结构是按需要分配的，以便映射用于存储数据的那些逻辑到物理地址。一旦辅助数据结构 1004、1006 等等的容量被超过，然后将分配另一个辅助数据结构，然后再分配下一个，等等，直到最后闪存介质 100/200 上的所有可能的物理扇区地址都被映射到逻辑扇区地址。每次分配辅助

表时，闪存驱动程序 306 将启用主数据结构 1002 中包含的指针，以指向它。

相应地，闪存驱动程序 306 将基于闪存介质本身中存储的永久数据量动态地分配一个或多个辅助数据结构 1004、1006。在运行时将使用闪存介质 100/200 的特定属性计算辅助数据结构的大小特征。不分配辅助数据结构，除非以前分配的辅助数据结构已满或不足以处理文件系统 305 所需的逻辑地址空间量。因此，动态查询数据结构 1000 最大限度地降低存储器 304 的使用。动态查询数据结构 1000 适用于使用日历、收件箱、文档等等的计算机设备 300，在此大多数逻辑扇区地址空间将不必映射到物理扇区地址。在这些应用程序中，只有有限的逻辑扇区范围被反复地访问，只有在该应用程序要求更多存储区域时才写入新逻辑扇区。

主数据结构 1002 包含一个指针阵列，0 到 N，它们指向已分配的那些辅助数据结构。在图 10 的示例中，位置 0 和 1 的指针分别指向辅助数据结构 1004 和 1006。此外，在图 10 的示例图表中，指针 2 到 N 不指向任何辅助数据结构，并包含默认设置“NULL”，以使逻辑到物理扇区映射模块 404 知道没有进一步分配辅助数据结构。

每一个辅助数据结构 1004、1006 都类似于数据结构 600，但只有全部可能的介质的一部分映射到辅助数据结构。辅助数据结构允许闪存抽象逻辑 308 将存储器 304 所需要空间量降低到只包括文件系统发出的逻辑扇区地址的那些部分。每一个辅助数据结构是 (b*k) 字节大小，其中 k 是数据结构中包含的物理扇区地址的数量，b 是用于存储每一个物理扇区地址的字节的数量。

图 11 说明了动态地分配用于跟踪闪存介质 100/200 上的数据的查询数据结构的过程 1100。过程 1100 包括步骤 1102 到 1106。描述该过程的顺序不能解释为限制。此外，该过程可以以任何适当的硬件、软件、固件或它们的组合来实现。

在步骤 1102 中，生成了包含指向一个或多个辅助数据结构

1004、1006 的主数据结构 1002。此示范实施例中的主数据结构 1002 的大小是固定的。在计算机设备 300 启动时，闪存介质逻辑 310 确定闪存介质 100/200 的大小，并将此信息中继到闪存抽象逻辑 308。基于闪存介质的大小，闪存抽象逻辑 308 计算物理地址的范围。即，假设闪存介质的大小是 16 MB，那么，NAND 闪存介质 100 通常将包含 32768 个扇区，每一个扇区的大小为 512 字节。这意味着，闪存抽象逻辑 308 可能必须将总数为 0 到最坏的情况下的 32768 个逻辑扇区，假设闪存介质上的所有存储器空间都已被使用。已知介质上有 2^{15} 个扇区，闪存抽象逻辑 308 可以使用 2 个字节存储每一个逻辑扇区地址的物理扇区地址。如此主数据结构以 256 个（N=256) DWORD 的阵列的方式来实现，涵盖了将由文件系统发出的最大的逻辑扇区地址数量（例如，32768）。如此，共有 256 个潜在的辅助数据结构。

在步骤 1104 中，分配辅助数据结构。首先，闪存抽象逻辑确定每一个潜在的辅助数据结构的可能的最小大小。使用简单的除法，每一个数据结构支持 32768 / 256 = 128 个逻辑扇区地址。如上所述，整个物理空间都可以使用 2 个字节映射，b=2，因此，每一个辅助数据结构的大小为 256 个字节（b=2*k=128）。

现在，已知每个辅助数据结构的大小，假定文件系统 305 请求写入到逻辑扇区地址 50-79，也被称为 LS50-LS79。为满足来自文件系统 305 的写入请求，闪存抽象逻辑 308 计划将主数据结构 1002 中的第一个指针用于逻辑扇区地址 LS0-LS127 或数据结构 1004。假设第一个指针是 NULL，闪存抽象逻辑 308 分配存储器 304 中的数据结构 1004，大小为 256 个字节。正如步骤 1106 所指出的，闪存抽象逻辑 308 能使主数据结构的位置 0 的指针指向数据结构 1004。因此，在此示例中，使用数据结构 1004 存储逻辑扇区 LS50-LS79 的映射信息。

如果文件系统 305 写入到闪存介质 100/200 中的对应的区域，闪存抽象逻辑 308 分配辅助数据结构。通常，只有已被使用的逻辑扇

区地址被闪存抽象逻辑 308 映射。因此，在最糟的情况下，当文件系统 305 访问整个逻辑地址空间时，那么对所有 256 辅助数据结构（在图 10 的示例中只显示将分配两个，1004、1006），每个将被分配 256 个字节，共要求存储器 304 中的 64KB 空间。

当分配的数据结构 1004 变得不足以存储文件系统 305 发出的逻辑扇区地址空间时，那么，闪存抽象逻辑 308 分配类似于数据结构 1006 的另一个数据结构。如果数据结构 1004 在稍后的时间再次足以处理文件系统作出的所有逻辑扇区地址请求，那么，这种动态地分配辅助数据结构的过程也适用。在此示例中，指向数据结构 1006 的指针将被闪存抽象逻辑 308 禁用；数据结构 1006 将成为存储器 304 中的空闲空间。

扇区的均匀磨损调整和回收

图 12 是被闪存驱动程序 306 作为连续的圆周 1200 看待的闪存介质 100/200 的示意图。在物理上闪存介质与如图 1 和 2 所示的介质 100/200 相同，只是闪存抽象逻辑 308 组织闪存介质，好像它是包含 0 到 N 个区块的连续圆周 1200。相应地，区块 N 内的最高的物理扇区地址（单个扇区未在图 12 中显示，以简化图表，但可以在图 1 和 2 中看到）和区块 0 内的最低的物理扇区地址被视为相邻的。

图 13 说明了作为连续圆周 1200 看待的介质 100/200 的另一个视图。在此示范图表中，扇区管理器 402 维护一个写入指针 1302，该指针指出介质上的可用于接收数据的下一个空闲扇区。下一个可用的空闲扇区是不必按规定的顺序首先擦除就可以接受数据的扇区。写入指针 1302 作为两个计数器的组合：计数扇区的扇区计数器 1306 和计数区块的区块计数器 1304。两种计数器组合起来可以指出用于接收数据的下一个可用的空闲扇区。

在另一个实施例中，写入指针 1302 可以作为单个计数器，并指出在写入操作期间可用于接受数据的下一个物理扇区。根据此实施例，扇区管理器 402 维护介质上可用于接收数据的所有物理扇区地址的

列表。扇区管理器 402 存储介质上的第一个和最后一个物理扇区地址（相邻的地址），并减去两个地址以确定空闲扇区的完整列表。然后，写入指针 1302 循环并连续地在列表中向前移动。这就减少了需要由扇区管理器 402 存储的信息量。

图 14 说明了被扇区管理器 402 用来确定闪存驱动程序 306 在介质 100/200 上存储数据的下一个可用的空闲扇区位置的过程 1400。过程 1400 还能使扇区管理器 402 提供每一个物理扇区地址（对于下一个空闲扇区），以便分配到文件系统 305 作出的每个逻辑扇区地址写入请求，如上文所述。过程 1400 包括步骤 1402-1418。描述该过程的顺序不能解释为限制。此外，该过程可以以任何适当的硬件、软件、固件或它们的组合来实现。

在步骤 1402 中，X 区块计数器 1304 和 Y 扇区计数器 1306 最初被设置为零。此时假设没有数据驻留在介质 100/200 上。

在步骤 1404 中，驱动程序 306 接收写入请求，将查询扇区管理器 402，以将下一个可用的空闲物理扇区地址发送到逻辑到物理扇区映射模块 404。写入请求可以来自文件系统 305 和/或来自用于回收扇区的压缩工具 406 内部，如下文所详述。

在步骤 1406 中，数据被写入到如写入指针 1302 所指出的扇区。由于在此示范图表中两种计数器最初被设置为零，假定写入指针 1302 指向扇区 0、区块 0。

在步骤 1408 中，扇区计数器 1306 向前移动一个有效的扇区。例如，在步骤 1406 的示例之后，写入指针向前移动到扇区 1、区块 0。

接下来，在判断步骤 1410 中，扇区管理器 402 检查扇区计数器 1306 是否超过一个区块中的扇区数量 K。如果 Y 计数器未超过区块的最大扇区大小，那么，根据判断步骤 1410 的“否”分支，对下一个写入请求重复步骤 1404-1410。

另一方面，如果 Y 计数器超过区块的最大扇区大小，那么，该区块的最高的物理扇区地址被写入到该区块，该区块已满。然后根据

步骤 1410 的“是”分支，在步骤 1412 中，Y 计数器被重置为零。接下来，在步骤 1414 中，X 区块计数器 1304 增加 1，从而将写入指针 1302 向前移动到该区块的最低的有效物理扇区地址 0 的下一个区块。

接下来，在判断步骤 1416 中，压缩工具 406 检查 X 区块计数器是否指向坏区块。如果是，那么，X 区块计数器 1304 将增加 1。在一个实施例中，压缩工具 406 负责检查此状况。如上文所提及的，扇区管理器存储可用于处理写入请求的所有物理扇区地址。物理扇区地址的整个区块始终在压缩或在初始化期间由压缩工具添加。因此，扇区管理器 402 不必检查区块是否为坏的，虽然扇区管理器可以这样做。还应注意，在其他实施例中，步骤 1416 可以在过程 1400 开始时执行。

在步骤 1417 中，X 区块计数器 1304 增加，直到它指向好的区块。为避免连续循环，如果所有区块都是坏的，那么过程 1400 将在步骤 1416 处停止，并给用户提供一个指示，说明所有区块都是坏的。

接下来在判断步骤 1418 中，扇区管理器检查 X 区块计数器 1304 是否超过最大区块数量 N。这将指出写入指针 1302 已经使圆周完整（在圆周 1200 的顶端）。如果是这种情况，那么根据步骤 1418 的“是”分支，过程 1400 重复，X 和 Y 计数器被重置为零。否则，根据步骤 1418 的“否”分支，过程 1400 返回到步骤 1404 并继续向前移动。

在此示范过程 1400 中，写入指针 1302 最初以最低的地址区块的最低的物理扇区地址开始。写入指针 1302 一次将扇区向前移动到最高的地址区块的最高的物理扇区地址，然后回到最低的地址区块，等等。此连续和循环的过程 1400 确保数据被均衡地写入到介质 100/200 的每个扇区。没有特定的区块或扇区被写入多次，从而可确保在整个介质 100/200 中有均匀的磨损调整。相应地，过程 1400 允许数据被非常快地写入到下一个可用的空闲扇区，不必使用用于确定在哪里写入新数据的昂贵处理算法，同时又可维护均匀的磨损调整。

此类传统的算法会影响计算机设备的写入速度。

在另一个实施例中，写入指针 1302 可以按逆时针方向移动，从最高的区块地址 N 的最高的物理扇区地址开始，并减少其计数器。在任何一种情况下，坏的区块都可以被扇区管理器完全忽略。此外，计数器可以被设置为任何值，不一定需要从计数器的最高的或最低的值开始。

图 15 说明了作为连续圆周 1200 看待的介质 100/200 的另一个视图。如图 15 所示，写入指针 1302 已经从区块 0 向前移动到 7，大约为整个圆周 1200 的一半。相应地，区块 0 到 7 包含脏的、有效的数据，或坏的区块。即，区块 0 到 7 中的每个好的扇区都不空闲，因此，无法用于接收新的或修改的数据。箭头 1504 表示区块 0 到 7 包含已被使用的扇区。最后，写入指针 1302 将要么用完空闲扇区，无法写入，除非清除和回收被标记为脏的或无效的扇区。要清除扇区，就是说，扇区被重置为可写状态，或换句话说，被“擦除”。为了释放扇区，必须至少一次擦除一个区块。然而，在可以擦除区块之前，所有好的扇区的内容都被复制到介质的不同部分的空闲扇区。然后，将扇区标记为“脏的”，再将区块擦除。

压缩工具 406 负责监视介质 100/200 的状况，以确定何时应擦除区块，以便将空闲扇区回收到扇区管理器 402。压缩工具 406 还负责执行清除操作。要完成清除操作，压缩工具 406 与扇区管理器 402 类似，维护了一个指针。在这种情况下，压缩工具 406 维护了一个清除指针 1502，图 15 显示了该指针。清除指针 1502 指向物理区块，能使压缩工具 406 在区块被清除时跟踪介质 100/200 上的扇区，如下文所述。压缩工具 406 可以维护一个指向下一个要压缩的区块的指针，因为擦除操作影响整个区块。即，当压缩工具 406 不在压缩区块时，压缩工具 406 指向一个区块。

图 16 是说明了被压缩工具用来回收扇区的过程的流程图。过程 1600 包括步骤 1602-1612。描述该过程的顺序不能解释为限制。此外，该过程可以以任何适当的硬件、软件、固件或它们的组合来实现。在

步骤 1602 中，压缩工具 406 监视文件系统向闪存介质 100/200 写入或更新的频率。这是通过专门监视介质 100/200 上的空闲和脏的扇区的数量来完成的。空闲扇区和脏的扇区的数量可以通过计算上文描述的表 600 和/或 1000 中存储的空闲和脏的扇区来判断。

在判断步骤 1604 中，压缩工具 406 执行两个比较，以确定是否应该回收扇区。第一个比较涉及将空闲扇区的数量与脏的扇区的数量进行比较。如果脏的扇区的数量超过空闲扇区的数量，那么，压缩工具 406 就认为应该执行回收操作，该操作在这种情况下被称为“服务压缩”。因此，当脏的扇区的数量超过空闲扇区的数量时，将进行服务压缩。

如果应该进行服务压缩，那么在步骤 1606 中，压缩工具在控制介质以执行步骤 1608-1612 以清除脏数据的区块之前等待低优先级线程 1606。服务压缩也可以在可以将脏的扇区回收到空闲扇区的其他方便的时间进行。例如，在另一个实施例中，当全部扇区的三分之一是脏扇区时，闪存抽象逻辑 308 可以执行服务压缩。在任何一个实施例中，压缩工具 406 通常等待优先级较高的线程，以放弃对处理器 302 和/或闪存介质 100/200 的控制。一旦已经有低优先级线程可用，过程将进入步骤 1608。

回过头来参看步骤 1604，第二个比较涉及比较介质上剩下的空闲扇区的数量，以确定写入指针 1302 是否即将或已经用完要指向的空闲扇区。如果果真是这种情况，那么，压缩工具 406 认为有必要预定一个“关键压缩”以回收扇区。压缩工具不等待低优先级线程并立即启动步骤 1608。

在步骤 1608 中，压缩工具 406 根据步骤 1604 的情况将在高优先级线程或低优先级线程上操作。如果在高级别线程上操作（关键压缩），那么，压缩工具 406 仅限于将少量的，例如，16 个脏扇区，回收到空闲扇区，并将对处理器的控制返回到计算机设备 300，以避免在这样的中断期间对处理器 302 的独占。

对于闪存介质，通常每个区块制造三十二个扇区，但对于关键压

缩，也可以其他数量的扇区，稍微大一些，或稍微小一些。不管这些大小特征如何，在关键压缩期间回收的扇区的数量都是任意的，但至少必须为 1(为了满足当前的写入请求)。关键压缩妨碍文件系统 305 能够完成写入；因此，尽快地完成压缩非常重要。在关键压缩的情况下，压缩工具 406 至少必须将一个脏的扇区回收到空闲扇区中，以便在介质上有空间满足挂起的写入请求。如果一次回收多个扇区，如 16 个，就可以避免有多个挂起的写入请求以及多个关键压缩紧接着执行的情况，有效地防止无限期地控制处理器。因此，尽管为关键压缩选择的回收的扇区的数量可以不同，但在示范描述中选择了足以防止紧接着的关键压缩的数量。

因此，在步骤 1608 中，压缩工具 406 将使用清除指针 1502 扫描扇区中的有效数据，将数据重写到空闲扇区，并在成功地移动数据之后将扇区标记为脏的。相应地，当移动数据时，压缩工具使用参考过程 700 描述的相同过程，该过程的代码与文件系统 305 写入新的数据和/或更新数据时使用的代码相同。压缩工具 406 在移动数据时查询扇区管理器 402 是否有空闲扇区，其方式与参考过程 1400 描述的方式相同。

在步骤 1610 中，压缩工具 406 使用类似于图 13 中所示的写入计数器 1306 的扇区计数器一个扇区一个扇区地移动清除指针 1502，只是此扇区计数器针对的是清除指针 1502 的位置。压缩工具 406 还通过计数器跟踪区块，其方式与参考写入指针 1302 描述的方式类似。然而，清除的区块的数量是由脏的扇区的数量确定的，关键压缩的情况除外。在关键压缩中，压缩工具只压缩足够的区块以回收少量的物理扇区（即，16 个扇区）。

在步骤 1612 中，压缩工具擦除（清除）包含完全被标记为脏的的好的扇区的那些区块。图 17 显示了过程 1600 的示范结果。在此示例中，在有必要进行另一个压缩的情况下，区块 0 和 1 被清除，清除指针被移到区块 2 的第一个扇区。因此，压缩工具 406 从区块 0 和 1 回收两个区块的扇区，这就向扇区管理器 402 提供了更多的

空闲扇区。已使用的扇区 1504 构成了在此实施例中按顺时针方向旋转的数据流（下文中将称之为“数据流”1504）。写入指针 1302 保持在数据流 1504 的首部，清除指针 1502 保持在数据流 1504 的“尾部”。数据流 1504 可以在数据被删除时收缩，或者随着新数据的添加而增长，但指针始终指向数据流 1504 的两个相对的尾部：头部和尾部。

对待闪存介质的方式为好像物理扇区地址构成一个连续的圆周 1200，并且使用上文描述的过程，能使闪存抽象逻辑 308 在整个介质 100/200 中执行均匀的磨损调整。压缩工具 406 选择给定区块相同的次数，以便通过擦除回收扇区。由于闪存区块具有有限的写入/擦除周期，压缩工具以及扇区管理器在区块 0-N 内尽可能均衡地分布这些操作。在这点上，数据流 1504 在圆周 1200（即，介质 100/200）旋转，均衡地在闪存介质 100/200 上提供完美的磨损调整。

在发生电源故障的情况下，闪存抽象逻辑 310 包含简单的编码逻辑，该逻辑扫描闪存介质 100/200，并确定哪些位置被标记为空闲和脏的。然后，该逻辑能够推断，数据流 1504 驻留在被标记为空闲和脏的位置之间，例如，图 17 中描述的圆周 1200 的数据流 1504 部分。数据流 1504 的头部和尾部可以通过定位包含首部数据的最高的物理扇区地址和通过定位包含尾部数据的最低的物理扇区地址来轻松地确定。

NOR 闪存设备

虽然此详细描述部分中的上述所有章节适用于 NAND 和 NOR 闪存设备，但是，如果使用了 NOR 闪存介质 200，也需要某些额外的实施例，才能使闪存介质逻辑支持在介质 200 的每个物理扇区中存储数据。每个 NOR 区块 0、1、2 等都可以被闪存介质逻辑 310 当作 NAND 闪存介质 100 来对待。具体来说，每个 NOR 区块都细分为一些页，其中每一页都包括 512 字节的“数据区域”，用于存储扇区数据，和 8 字节“备用区域”，用于存储诸如逻辑扇区地址、状态位等等之类的东西（如上文所述）。

图 18 说明了为更好地支持闪存驱动程序实现的过程和技术划分的 NOR 闪存介质 200 的逻辑表示。在此实施例中，扇区 1802 包含 512 字节的数据区域 1803，用于存储与扇区相关的数据，和 8 字节的数据区域，用于作为备用区域 1804。区段 1806 表示 NOR 区块的未使用的部分，因为 NOR 闪存区块通常大小为 2，它不是均衡可分的。例如，请看一个 16 MB 的 NOR 闪存设备，该设备具有 128 个闪存区块，每一个区块的大小为 128 KB。使用 520 字节的页面大小，每一个 NOR 闪存区块都可以分成 252 个不同的扇区，32 个字节是未使用的。不幸的是，在示范实施例中，这每个区块的 32 个字节被闪存介质逻辑 310“浪费”，未被用来存储扇区数据。然而，折中是增强的写入吞吐量，均匀的磨损调整、将数据丢失减到最小限度等等，如上文所述的示范闪存驱动程序 306 的闪存抽象逻辑 308 提供的全部。另外一些实施例可以通过将介质 200 分成不同的扇区大小的方法来完成。

计算机可读的介质

使用如上文所述的闪存驱动程序的示范主题的实施例可以存储在计算机可读的介质上或通过计算机可读的介质的某些形式进行传输。计算机可读的介质可以是计算机可以访问的任何可用的介质。作为示例，而不仅限于，计算机可读的介质可以包括“计算机存储介质”和“通信介质”。

“计算机存储介质”包括以任何方法或技术实现的易失性的和非易失性的、可移动的和非可移动的介质，以用于存储诸如计算机可读的指令、数据结构、程序模块或其他数据之类的信息。计算机存储介质包括，但不仅限于，RAM、ROM、EEPROM、闪存或其他存储器技术，CD-ROM、数字多功能磁盘 (DVD) 或其他光盘存储器、磁带机、磁带、磁盘存储器或其他磁存储设备，或者可用于存储所需要的信息并且计算机可以访问的任何其他介质。

“通信介质”通常在诸如载波之类的调制数据信号或其他传输机制中实施计算机可读的指令、数据结构、程序模块或者其他数据。通信

介质还包括任何信息提供介质。

术语“调制的数据信号”是指具有一个或多个其特征集或以这样的方式以便对信号中的信息进行编码的信号。作为示例，而不仅限于，通信介质包括有线介质，如有线网络或者直接有线连接，以及无线介质，如声控、RF、红外和其他无线介质。上述任何几项的组合也应包括在计算机可读的介质的范围内。

结束语

虽然本发明是以结构功能和/或方法操作所特有的语言描述的，可以理解，在所附的权利要求书中定义的本发明不一定仅限于描述的特定功能或操作。特定功能和操作可以作为实现权利要求所述的发明的示范形式来描述。

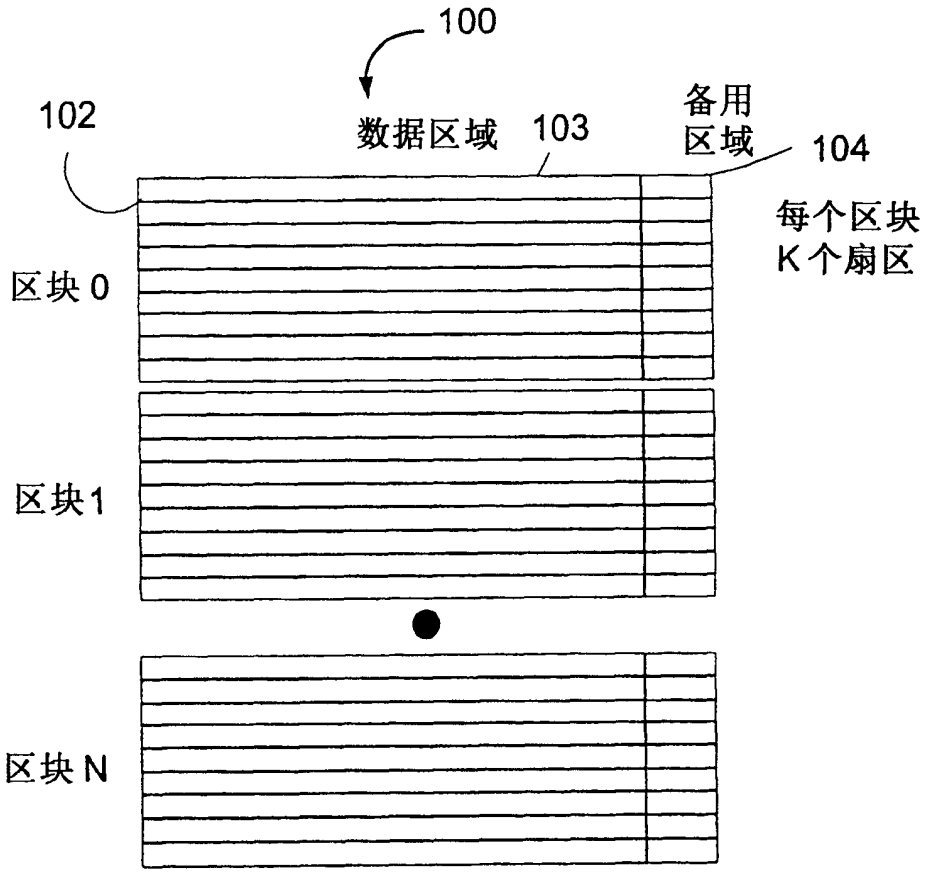


图1

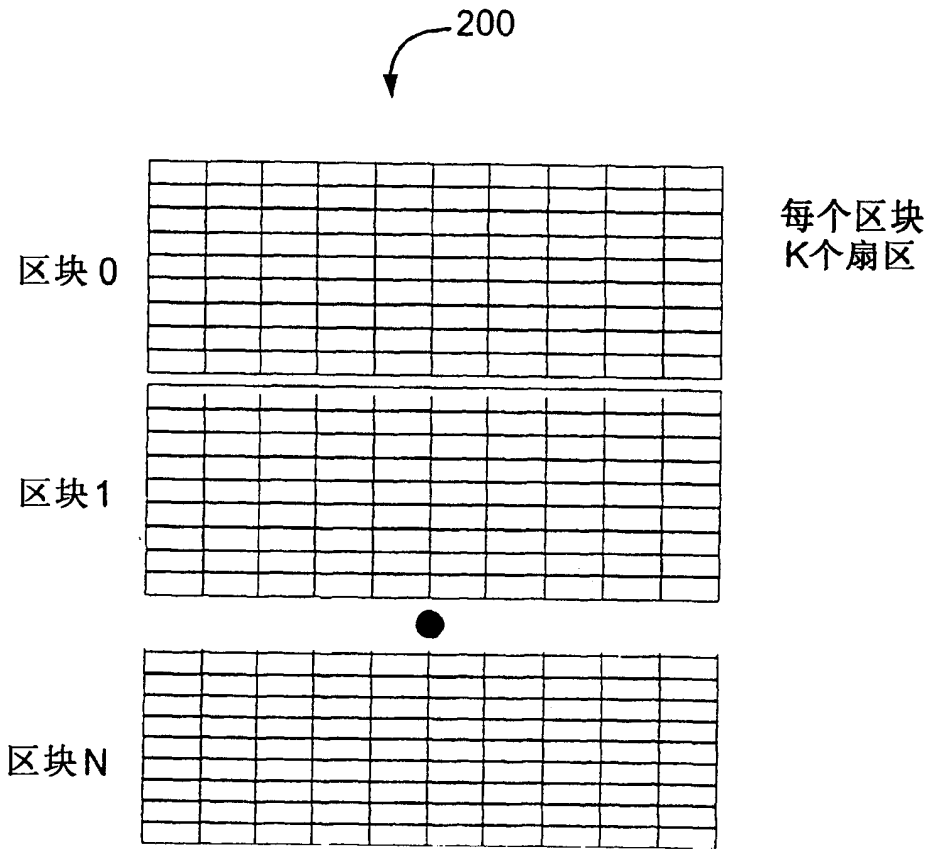


图2

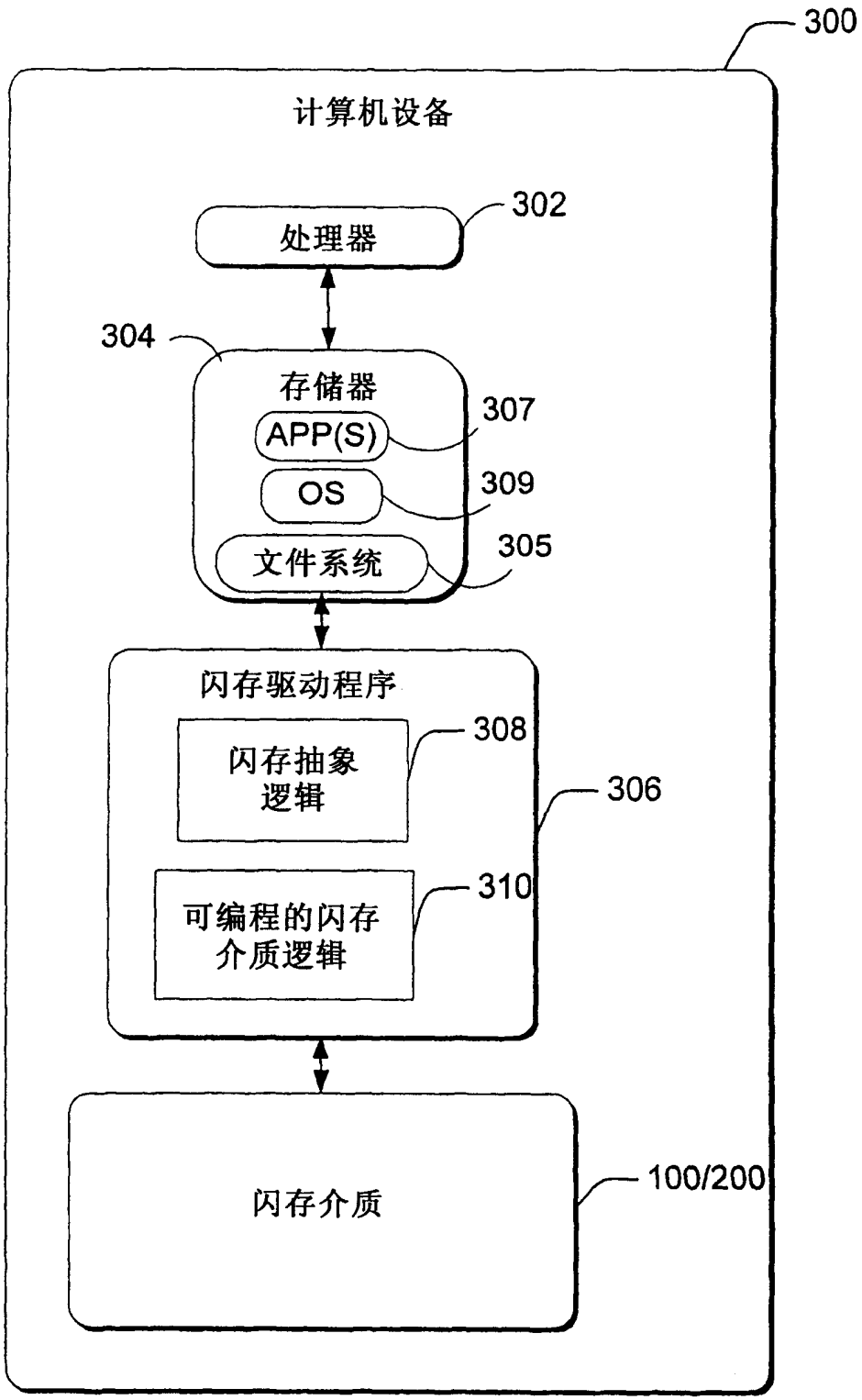

300
计算机设备
302
处理器
304
存储器
307
APP(S)
309
OS
305
文件系统
闪存驱动程序
308
闪存抽象逻辑
306
310
可编程的闪存介质逻辑
100/200
闪存介质

图3

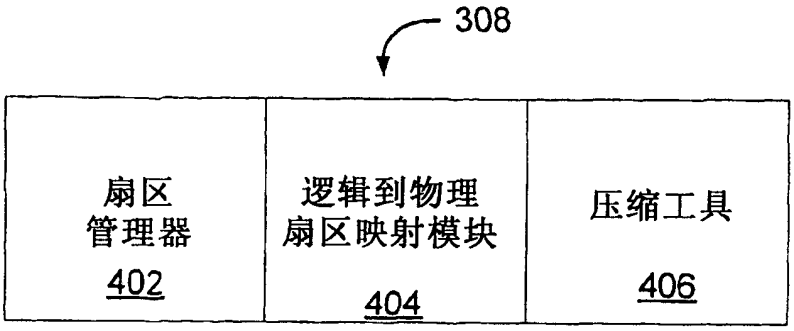

308
扇区
管理器
402
逻辑到物理
扇区映射模块
404
压缩工具
406

图4

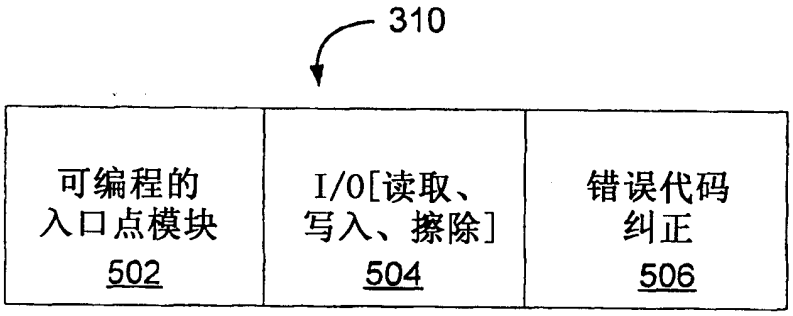

310
可编程的
入口点模块
502
I/O[读取、
写入、擦除]
504
错误代码
纠正
506

图5

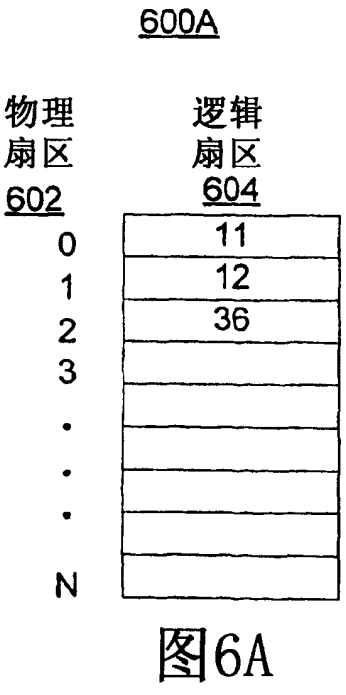

600A
物理
扇区
602
逻辑
扇区
604
0
1
2
3
N
11
12
36

图6A

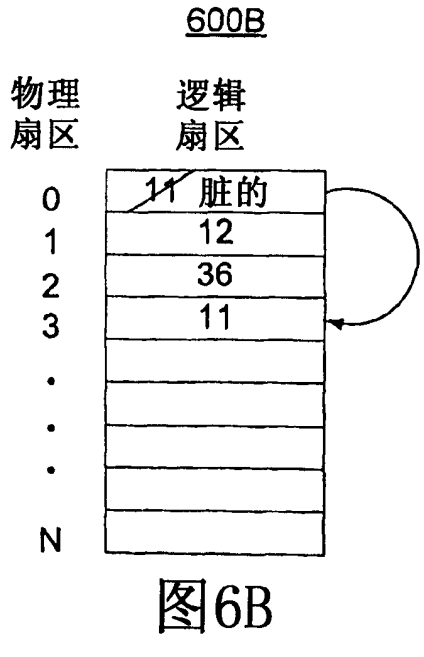

600B
物理
扇区
逻辑
扇区
0
1
2
3
N
11 脏的
12
36
11

图6B

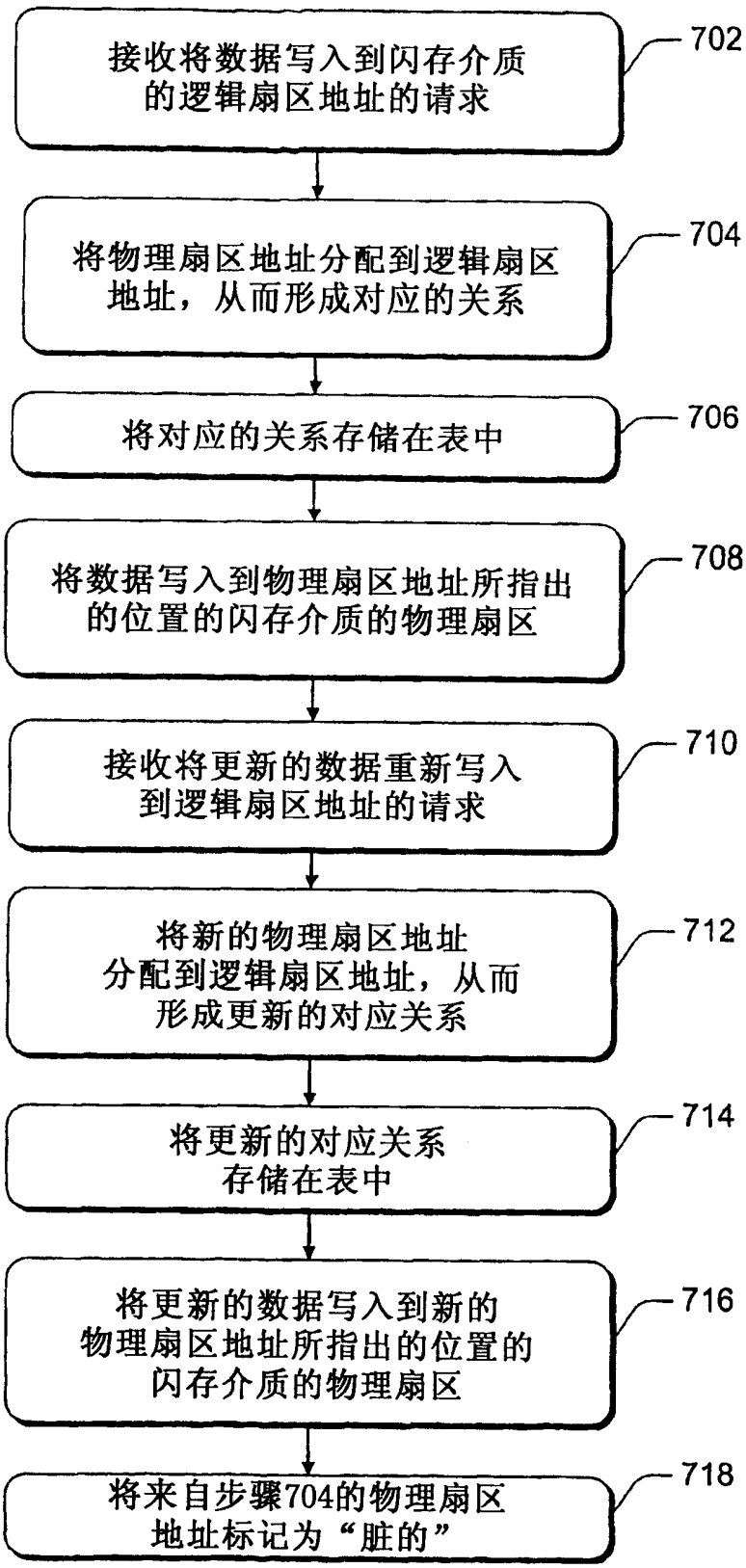

700
接收将数据写入到闪存介质的逻辑扇区地址的请求
702
将物理扇区地址分配到逻辑扇区地址，从而形成对应的关系
704
将对应的关系存储在表中
706
将数据写入到物理扇区地址所指出的位置的闪存介质的物理扇区
708
接收将更新的数据重新写入到逻辑扇区地址的请求
710
将新的物理扇区地址分配到逻辑扇区地址，从而形成更新的对应关系
712
将更新的对应关系存储在表中
714
将更新的数据写入到新的物理扇区地址所指出的位置的闪存介质的物理扇区
716
将来自步骤704的物理扇区地址标记为“脏的”
718

图7

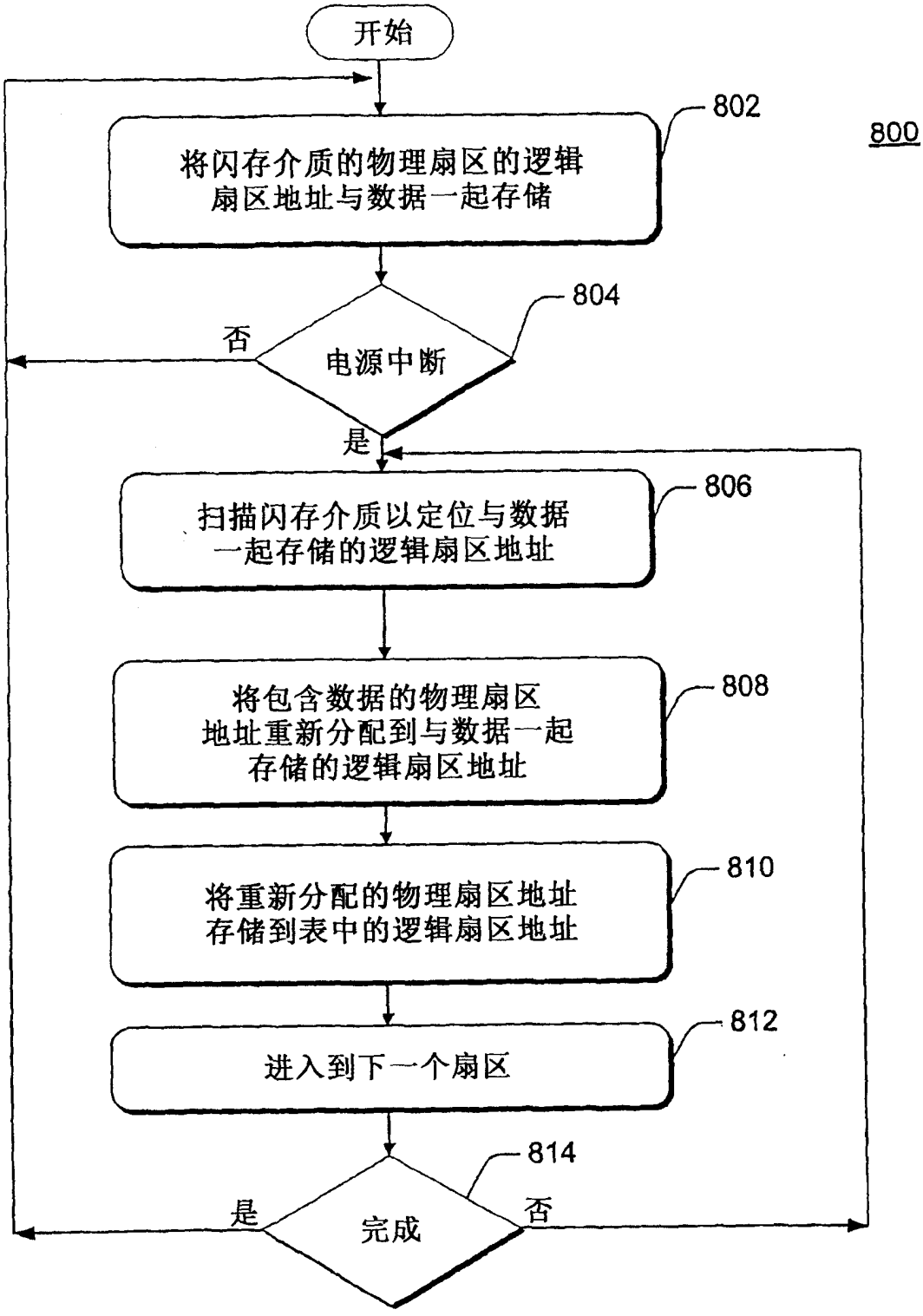

开始
802
800
将闪存介质的物理扇区的逻辑扇区地址与数据一起存储
804
否
电源中断
是
806
扫描闪存介质以定位与数据一起存储的逻辑扇区地址
808
将包含数据的物理扇区地址重新分配到与数据一起存储的逻辑扇区地址
810
将重新分配的物理扇区地址存储到表中的逻辑扇区地址
812
进入到下一个扇区
814
是
完成
否

图8

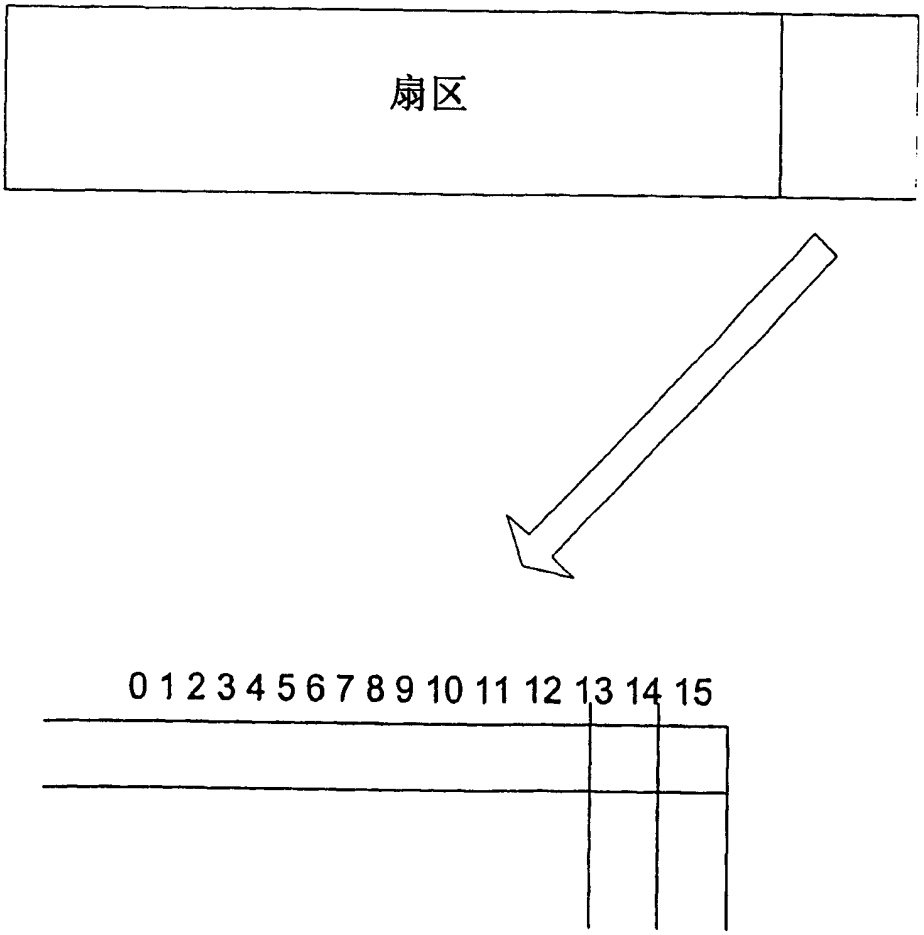


图9

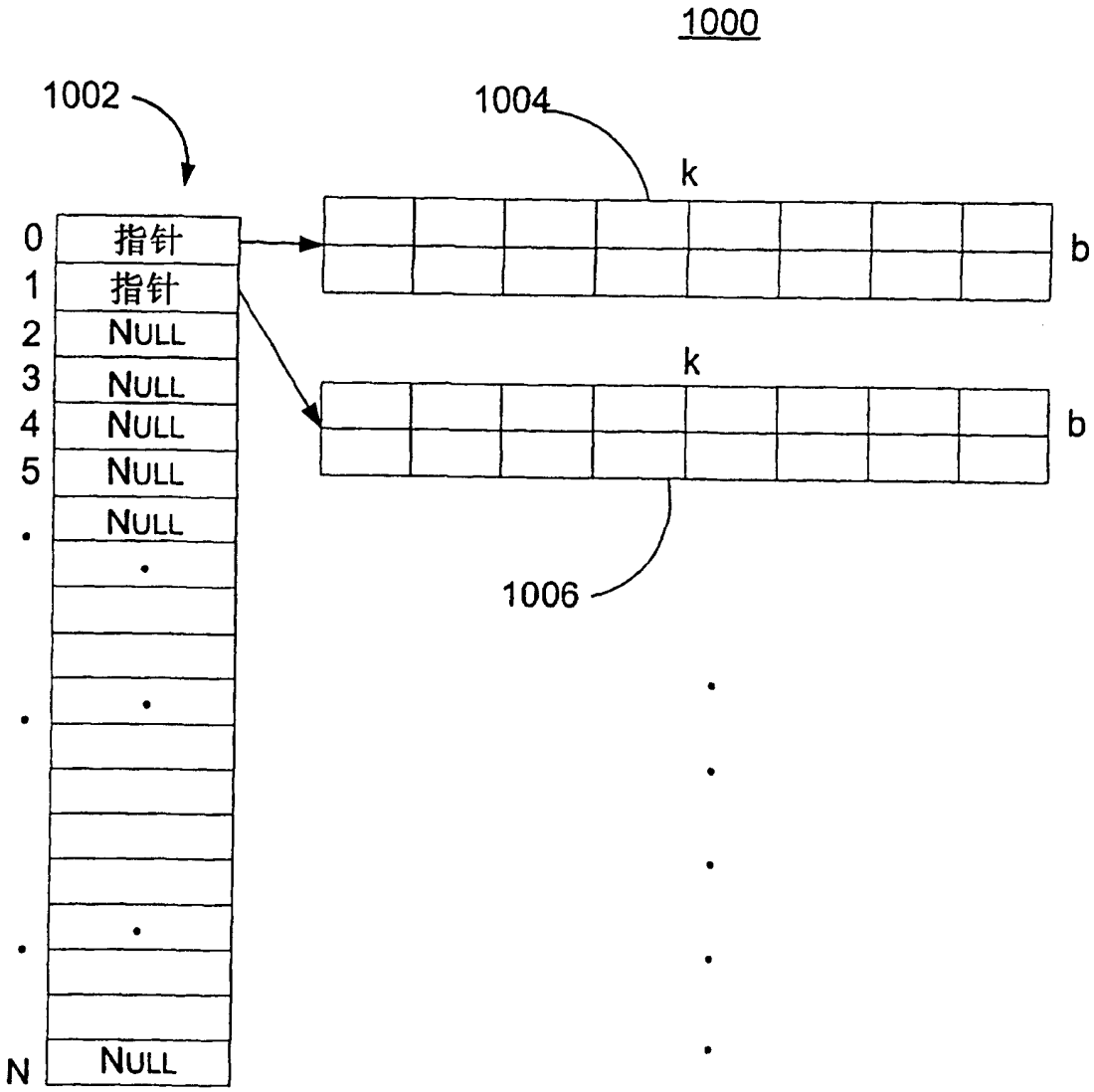

1000
1002
1004
k
0
指针
1
指针
2
NULL
3
NULL
4
NULL
5
NULL
NULL
b
k
b
1006
N
NULL

图10

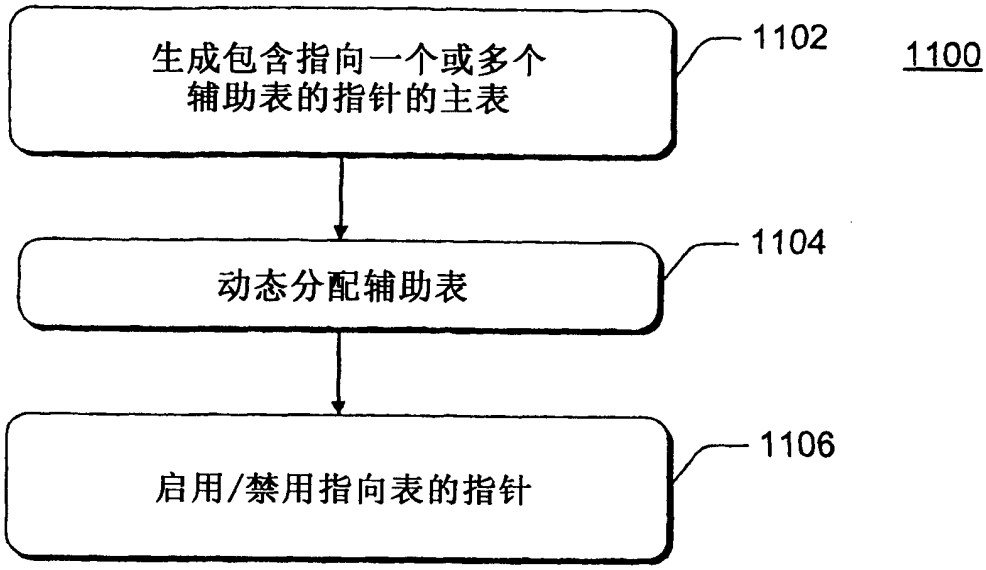

生成包含指向一个或多个辅助表的指针的主表
1102
1100
动态分配辅助表
1104
启用/禁用指向表的指针
1106

图11

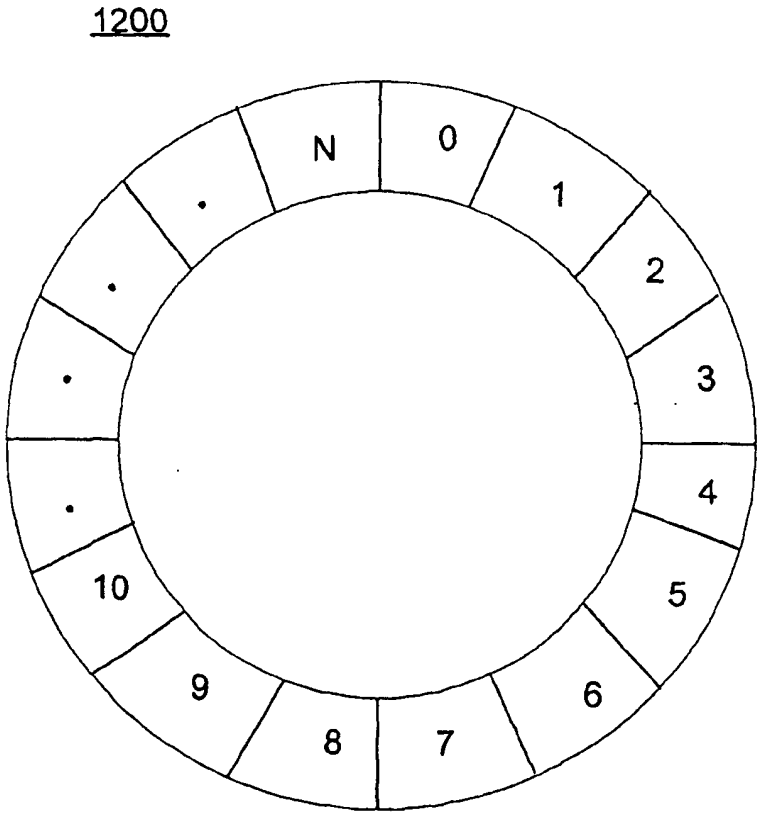

1200
N
0
1
2
3
4
5
6
7
8
9
10

图12

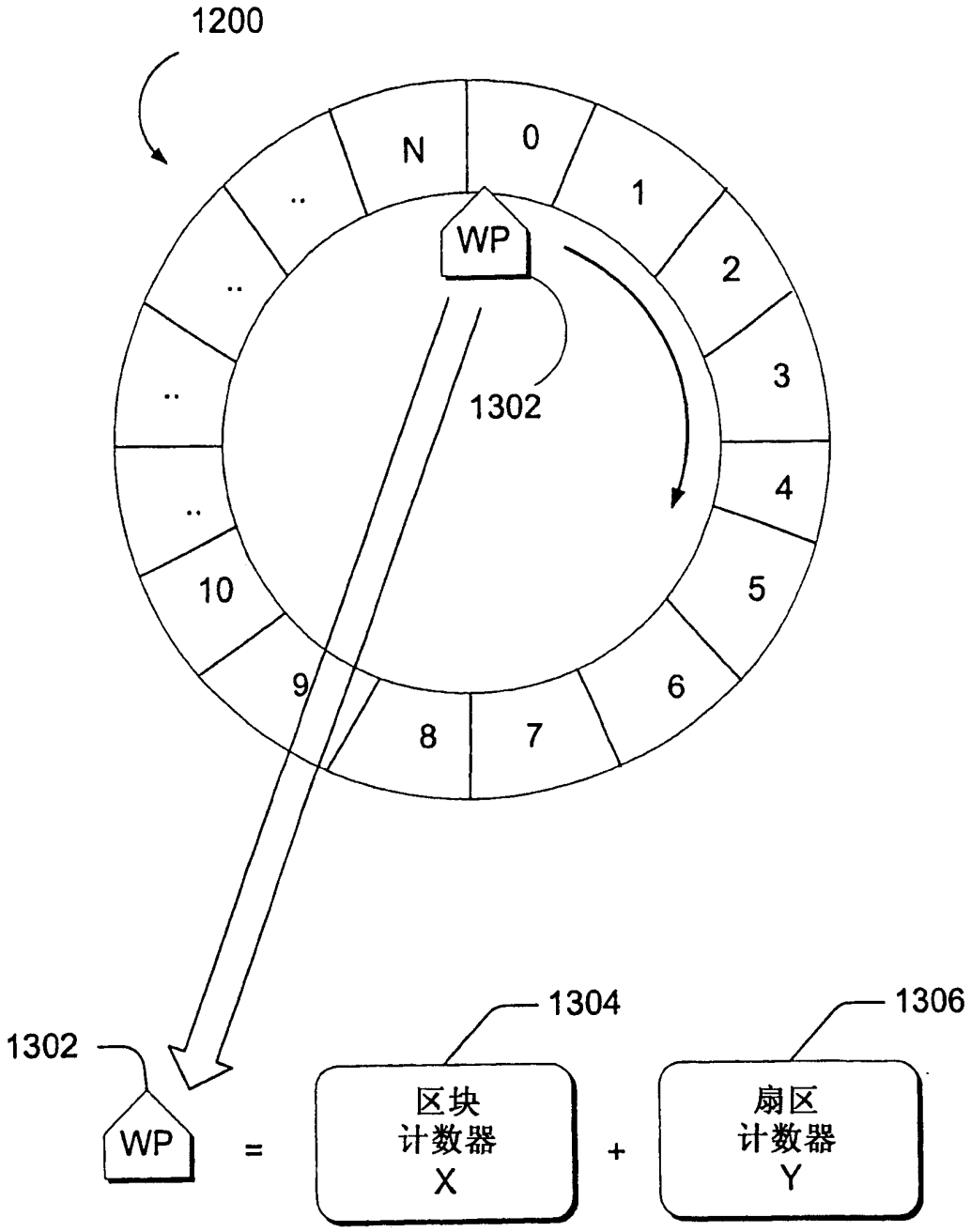


图13

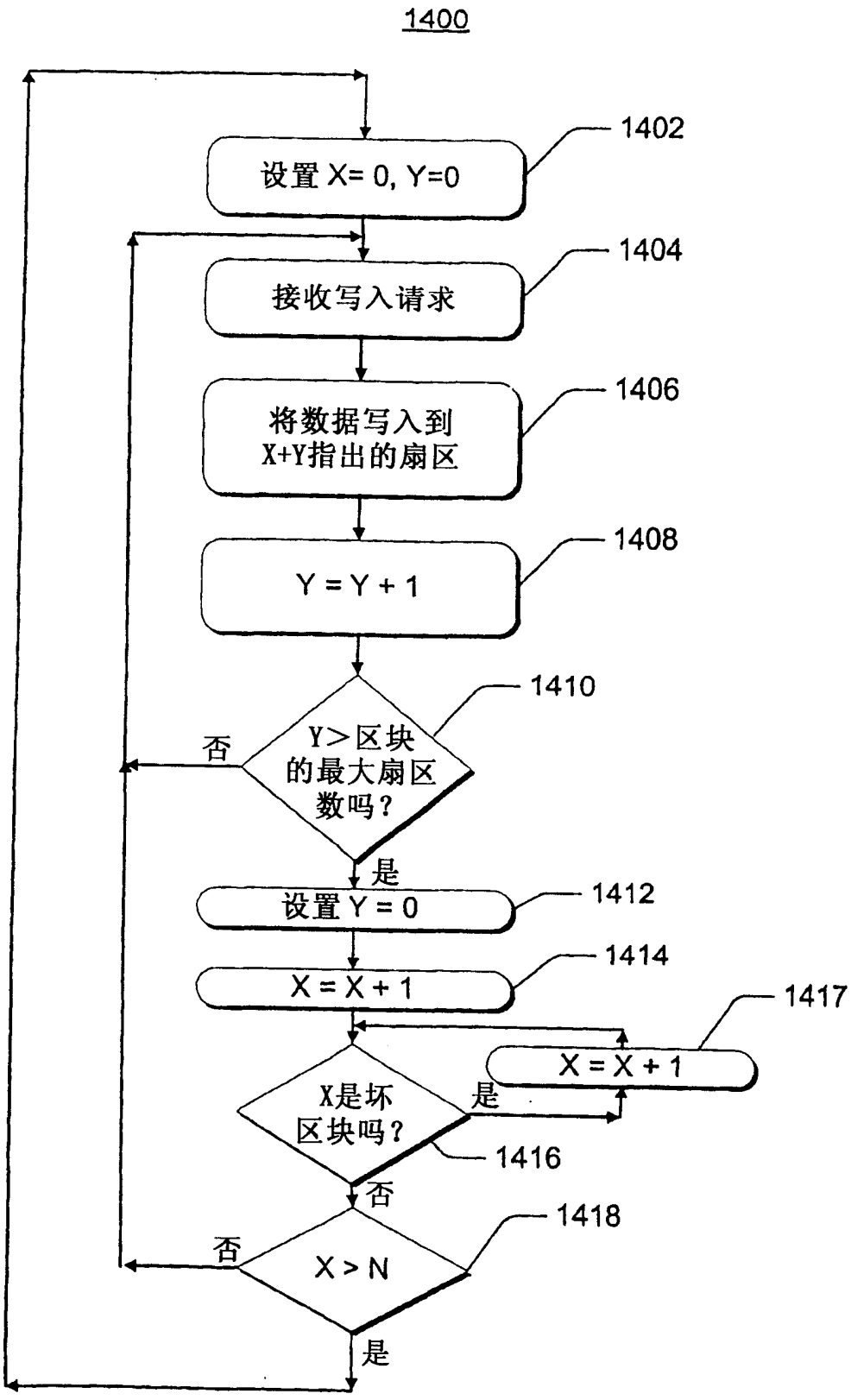

1400
设置 X= 0, Y=0
1402
接收写入请求
1404
将数据写入到
X+Y指出的扇区
1406
Y = Y + 1
1408
Y>区块
的最大扇区
数吗?
1410
否
是
设置 Y = 0
1412
X = X + 1
1414
X = X + 1
1417
X是坏
区块吗?
是
1416
否
X > N
1418
否
是

图14

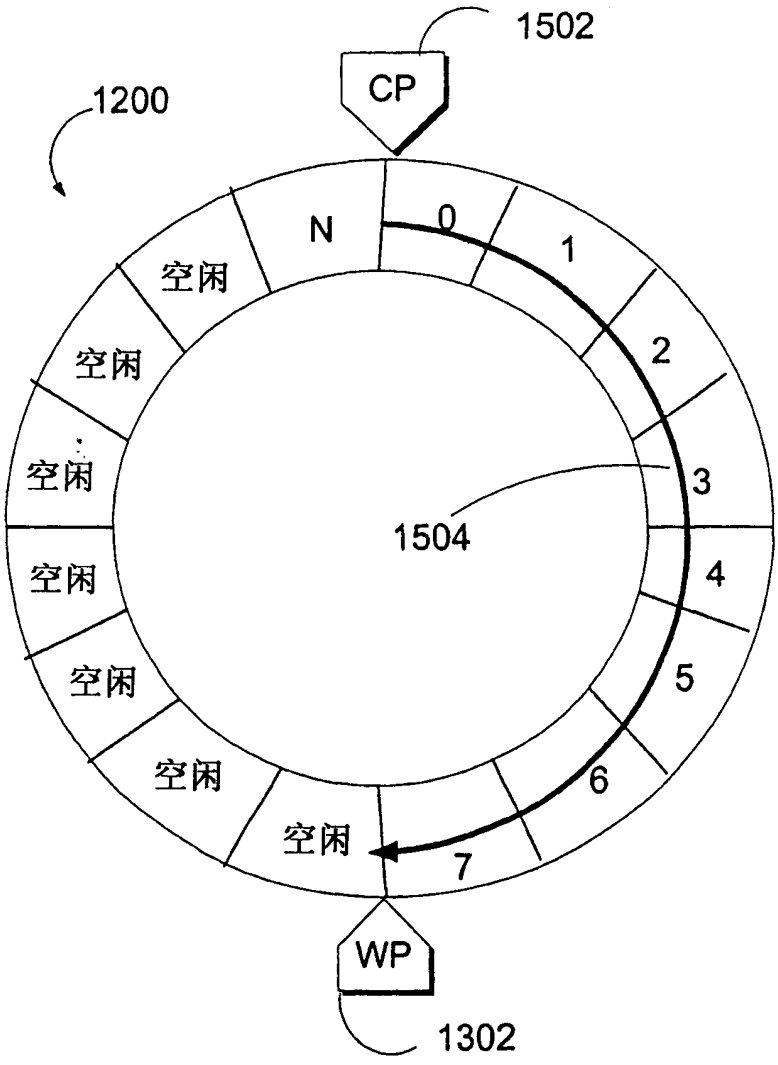


图15

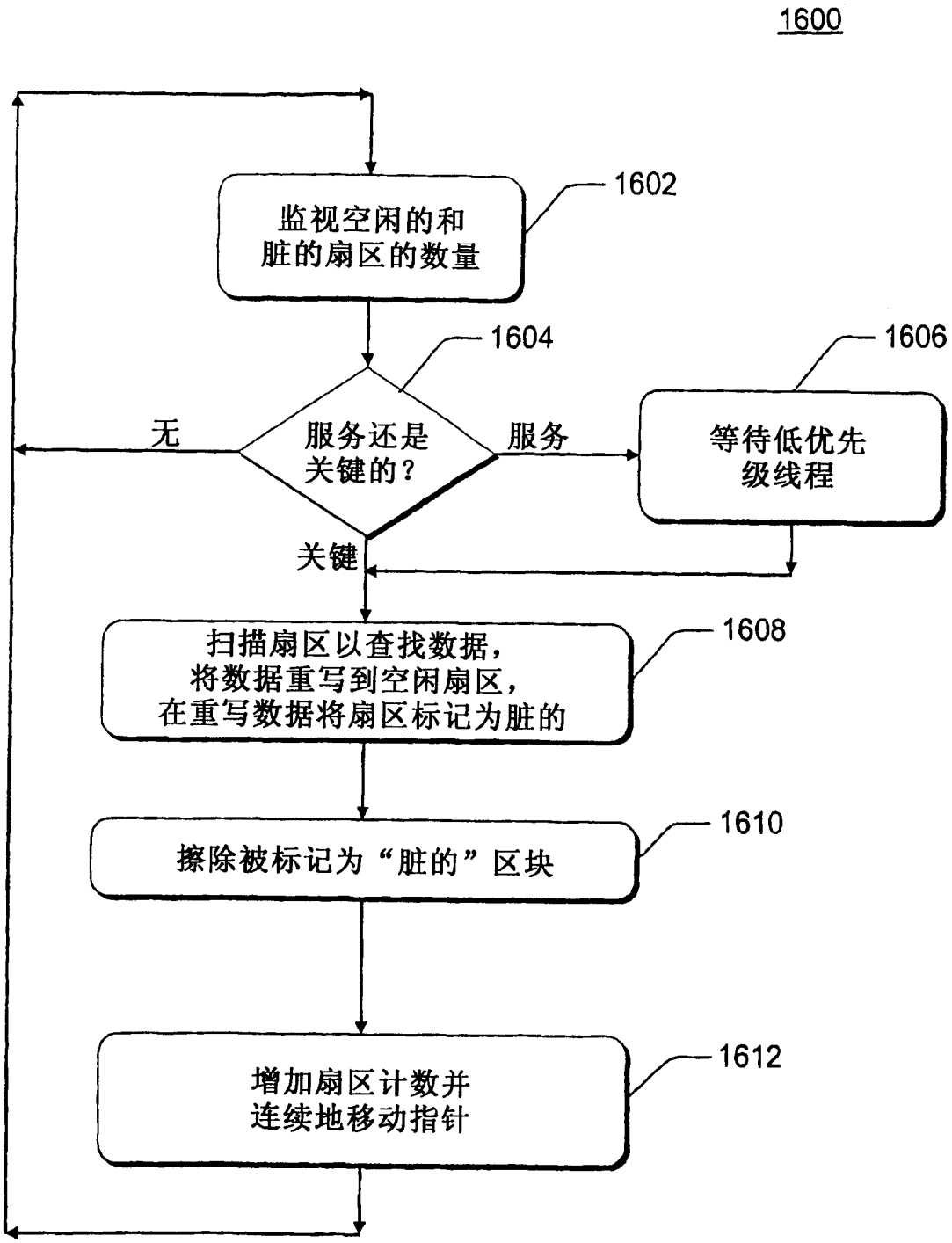

1600
监视空闲的和
脏的扇区的数量
1602
1604
服务还是
关键的？
无
服务
关键
1606
等待低优先
级线程
扫描扇区以查找数据，
将数据重写到空闲扇区，
在重写数据将扇区标记为脏的
1608
擦除被标记为“脏的”区块
1610
增加扇区计数并
连续地移动指针
1612

图16

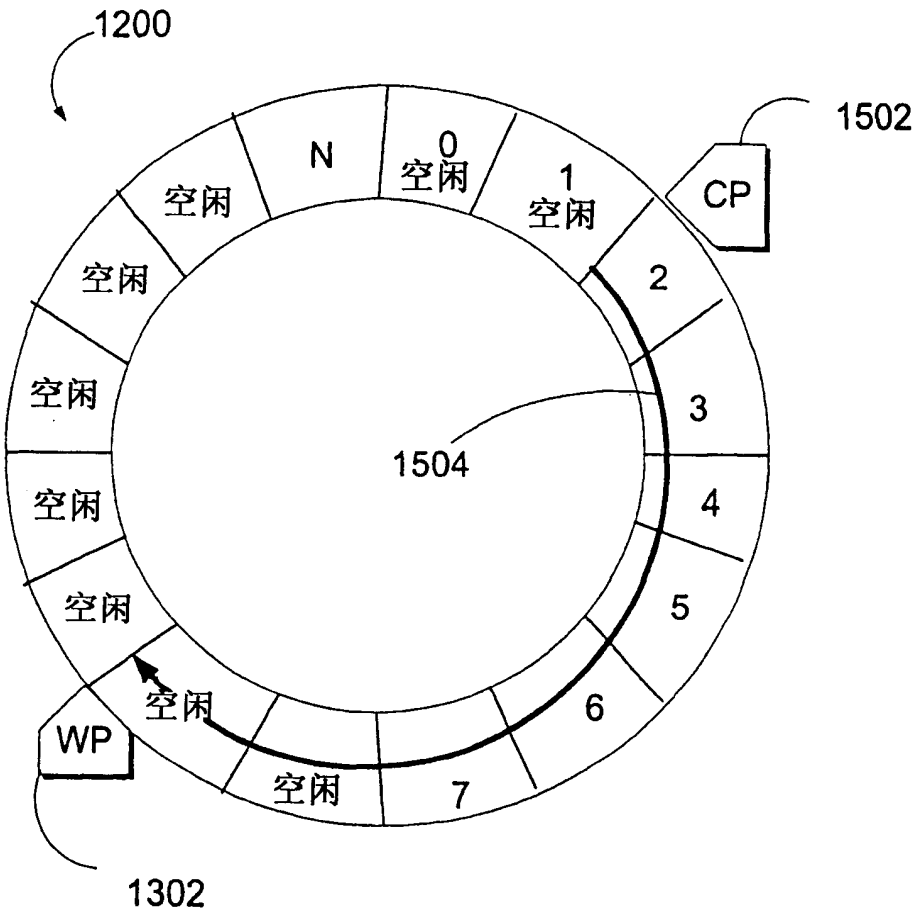


图17

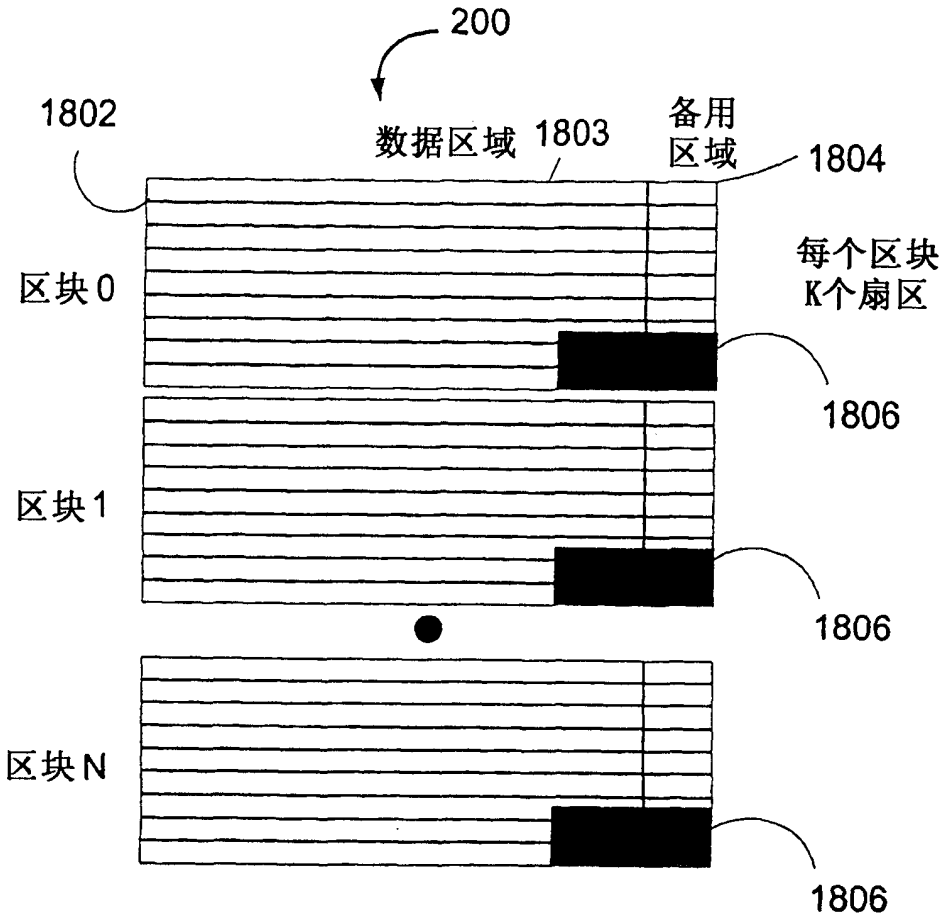

200
1802
数据区域
1803
备用
区域
1804
每个区块
K个扇区
区块0
1806
区块1
1806
区块N
1806

图18