(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 114328373 A
(43)申请公布日 2022.04.12

(21)申请号 202011051335.8

(22)申请日 2020.09.29

(71)申请人 伊姆西IP控股有限责任公司
地址 美国马萨诸塞州

(72)发明人 赵朝俊 罗明艺 黄佳 曾泓源 王豪

(74)专利代理机构 北京市金杜律师事务所 11256
代理人 黄倩

(51)Int.Cl.
G06F 16/11(2019.01)
G06F 16/174(2019.01)

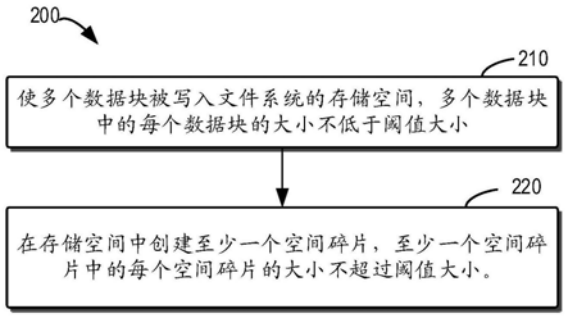
权利要求书2页 说明书10页 附图10页

(54)发明名称

管理文件系统的方法、电子设备和计算机程序产品

(57)摘要

根据本公开的示例实施例,提供了一种管理文件系统的方法、电子设备和计算机程序产品。该方法包括:使多个数据块被写入文件系统的存储空间,多个数据块中的每个数据块的大小不低于阈值大小;以及在存储空间中创建至少一个空间碎片,至少一个空间碎片中的每个空间碎片的大小不超过阈值大小。由此,本方案可以快速、高效地使文件系统老化。



CN 114328373 A

1.一种管理文件系统的方法,包括:

使多个数据块被写入所述文件系统的存储空间,所述多个数据块中的每个数据块的大小不低于阈值大小;以及

在所述存储空间中创建至少一个空间碎片,所述至少一个空间碎片中的每个空间碎片的大小不超过所述阈值大小。

2.根据权利要求1所述的方法,其中创建所述至少一个空间碎片包括:

使所写入的所述多个数据块中的至少两个数据块从所述存储空间被移除,所述至少两个数据块在所述存储空间中不相邻。

3.根据权利要求2所述的方法,其中所述多个数据块的大小均为预定大小。

4.根据权利要求1所述的方法,其中使多个数据块被写入所述存储空间包括:

以第一压缩率来压缩所述多个数据块中的第一数据块,以得到第一压缩数据块;以及

使所述第一压缩数据块被写入所述存储空间中的第一子空间。

5.根据权利要求4所述的方法,其中创建所述至少一个空间碎片包括:

使所述第一压缩数据块从所述第一子空间中被移除;

以第二压缩率来压缩所述第一数据块,以得到第二压缩数据块,所述第一压缩率小于所述第二压缩率;以及

使所述第二压缩数据块被写入所述第一子空间。

6.根据权利要求1所述的方法,其中使多个数据块被写入所述存储空间包括:

使多个数据块中的第二数据块被写入所述存储空间中的第二子空间;

以第三压缩率来压缩所述多个数据块中的第三数据块,以得到第三压缩数据块;以及

使所述第三压缩数据块被写入所述存储空间中的第三子空间,所述第二数据块的大小大于所述第三数据块的大小,所述第二子空间和所述第三子空间不相邻。

7.根据权利要求6所述的方法,其中创建所述至少一个空间碎片包括:

使所述第二数据从所述第二子空间中被移除;

以第四压缩率来压缩所述第三数据块,以得到第四压缩数据块,所述第四压缩率小于所述第三压缩率;以及

使所述第四压缩数据块被写入所述第二子空间。

8.根据权利要求7所述的方法,还包括:

使所述第三压缩数据块从所述第三子空间中被移除。

9.根据权利要求1所述的方法,还包括:

在所述至少一个空间碎片被创建后确定所述文件系统的性能,所述性能包括以下中的至少一项:所述文件系统的响应时间、所述文件系统读取/写入数据的平均带宽、所述文件系统每秒进行读写操作的数目以及所述文件系统的故障率。

10.一种电子设备,包括:

至少一个处理单元;

至少一个存储器,所述至少一个存储器被耦合到所述至少一个处理单元并且存储用于由所述至少一个处理单元执行的指令,所述指令当由所述至少一个处理单元执行时,使得所述设备执行动作,所述动作包括:

使多个数据块被写入所述文件系统的存储空间,所述多个数据块中的每个数据块的大

小不低于阈值大小;以及

在所述存储空间中创建至少一个空间碎片,所述至少一个空间碎片中的每个空间碎片的大小不超过所述阈值大小。

11.根据权利要求10所述的设备,其中创建所述至少一个空间碎片包括:

使所写入的所述多个数据块中的至少两个数据块从所述存储空间被移除,所述至少两个数据块在所述存储空间中不相邻。

12.根据权利要求11所述的设备,其中所述多个数据块的大小均为预定大小。

13.根据权利要求10所述的设备,其中使多个数据块被写入所述存储空间包括:

以第一压缩率来压缩所述多个数据块中的第一数据块,以得到第一压缩数据块;以及

使所述第一压缩数据块被写入所述存储空间中的第一子空间。

14.根据权利要求13所述的设备,其中创建所述至少一个空间碎片包括:

使所述第一压缩数据块从所述第一子空间中被移除;

以第二压缩率来压缩所述第一数据块,以得到第二压缩数据块,所述第一压缩率小于所述第二压缩率;以及

使所述第二压缩数据块被写入所述第一子空间。

15.根据权利要求10所述的设备,其中使多个数据块被写入所述存储空间包括:

使多个数据块中的第二数据块被写入所述存储空间中的第二子空间;

以第三压缩率来压缩所述多个数据块中的第三数据块,以得到第三压缩数据块;以及

使所述第三压缩数据块被写入所述存储空间中的第三子空间,所述第二数据块的大小大于所述第三数据块的大小,所述第二子空间和所述第三子空间不相邻。

16.根据权利要求15所述的设备,其中创建所述至少一个空间碎片包括:

使所述第二数据从所述第二子空间中被移除;

以第四压缩率来压缩所述第三数据块,以得到第四压缩数据块,所述第四压缩率小于所述第三压缩率;以及

使所述第四压缩数据块被写入所述第二子空间。

17.根据权利要求16所述的设备,还包括:

使所述第三压缩数据块从所述第三子空间中被移除。

18.根据权利要求10所述的设备,还包括:

在所述至少一个空间碎片被创建后确定所述文件系统的性能,所述性能包括以下中的至少一项:所述文件系统的响应时间、所述文件系统读取/写入数据的平均带宽、所述文件系统每秒进行读写操作的数目以及所述文件系统的故障率。

19.一种计算机程序产品,所述计算机程序产品被有形地存储在非瞬态计算机可读介质上并且包括机器可执行指令,所述机器可执行指令在被执行时使机器执行根据权利要求1至9中任一项所述的方法的步骤。

管理文件系统的方法、电子设备和计算机程序产品

技术领域

[0001] 本公开的实施例总体涉及管理文件系统的方法、设备和计算机程序产品。

背景技术

[0002] 用户在文件系统中进行操作的过程中，大量的数据不可用和数据丢失时间是由定时、锁、线程和内容冲突引起的，这在长期使用后的文件系统中尤其明显，其中经常出现内存溢出(OOM)、紧急情况、文件系统脱机、线程阻塞等问题。长期使用的文件系统会使其老化并且性能降低，这在固态硬盘(SSD)上的老化的文件系统的性能损失可能比机械硬盘(HDD)上的损失还要严重。

[0003] 在传统解决方案中，为了发现这些问题，通常会进行大量的测试操作，例如单元测试，功能测试，集成测试，压力测试和耐力测试。然而，由于用户的文件系统通常经过了长时间的使用已经老化、存储阵列从许多故障中恢复、很少执行不间断升级且很少执行重新初始化，而当前大多数系统测试都在新创建存储器阵列上运行，所以在系统测试环境与用户文件系统之间存在显著的差异。然而，在一般的系统环境测试中很难发现上述问题。

发明内容

[0004] 本公开的实施例提供了管理文件系统的方法、设备和计算机程序产品。

[0005] 在本公开的第一方面，提供了一种管理文件系统的方法。该方法包括：使多个数据块被写入文件系统的存储空间，多个数据块中的每个数据块的大小不低于阈值大小；以及在存储空间中创建至少一个空间碎片，至少一个空间碎片中的每个空间碎片的大小不超过阈值大小。

[0006] 在本公开的第二方面，提供了一种电子设备。该设备包括至少一个处理单元和至少一个存储器。至少一个存储器被耦合到至少一个处理单元并且存储用于由至少一个处理单元执行的指令。该指令当由至少一个处理单元执行时使得设备执行动作，该动作包括：使多个数据块被写入文件系统的存储空间，多个数据块中的每个数据块的大小不低于阈值大小；以及在存储空间中创建至少一个空间碎片，至少一个空间碎片中的每个空间碎片的大小不超过阈值大小。

[0007] 在本公开的第三方面，提供了一种计算机程序产品。计算机程序产品被有形地存储在非瞬态计算机可读介质上并且包括机器可执行指令，机器可执行指令在被执行时使机器实现根据本公开的第一方面所描述的方法的任意步骤。

[0008] 提供发明内容部分是为了以简化的形式来介绍对概念的选择，它们在下文的具体实施方式中将被进一步描述。发明内容部分无意标识本公开的关键特征或必要特征，也无意限制本公开的范围。

附图说明

[0009] 通过结合附图对本公开示例性实施例进行更详细的描述，本公开的上述以及其它

目的、特征和优势将变得更加明显，其中，在本公开示例性实施例中，相同的参考标号通常代表相同部件。在附图中：

[0010] 图1示出了本公开的实施例可以在其中被实现的示例环境的示意图；

[0011] 图2示出了根据本公开的实施例的管理文件系统的方法的流程图；

[0012] 图3示出了根据本公开的实施例的管理文件系统的示意图；

[0013] 图4示出了根据本公开的实施例的管理文件系统的环境的示意图；

[0014] 图5示出了根据本公开的实施例的管理文件系统的示意图；

[0015] 图6示出了根据本公开的实施例的管理文件系统的示意图；

[0016] 图7示出了根据本公开的实施例的管理文件系统的示意图；

[0017] 图8示出了根据本公开的实施例的在经老化的文件系统上进行测试的示意图；

[0018] 图9示出了根据本公开的实施例的用于记录文件系统管理过程的设备的示意图；

[0019] 图10示出了根据本公开的实施例的文件系统性能的示意图；

[0020] 图11示出了根据本公开的实施例的文件系统性能的示意图；

[0021] 图12示出了根据本公开的实施例的文件系统性能的示意图；

[0022] 图13示出了根据本公开的实施例的文件系统性能的示意图；以及

[0023] 图14示出了可以用来实施本公开的实施例的示例设备的框图。

[0024] 在各个附图中，相同或对应的标号表示相同或对应的部分。

具体实施方式

[0025] 下面将参照附图更详细地描述本公开的优选实施例。虽然附图中显示了本公开的优选实施例，然而应该理解，可以以各种形式实现本公开而不应被这里阐述的实施例所限制。相反，提供这些实施例是为了使本公开更加透彻和完整，并且能够将本公开的范围完整地传达给本领域的技术人员。

[0026] 在本文中使用的术语“包括”及其变形表示开放性包括，即“包括但不限于”。除非特别申明，术语“或”表示“和/或”。术语“基于”表示“至少部分地基于”。术语“一个示例实施例”和“一个实施例”表示“至少一个示例实施例”。术语“另一实施例”表示“至少一个另外的实施例”。术语“第一”、“第二”等等可以指代不同的或相同的对象。下文还可能包括其他明确的和隐含的定义。

[0027] 通常而言，文件系统具有其自己的生命周期。终端用户在文件系统中创建、读取、写入、截断、删除和复制文件、目录和链接。文件系统可以从一个地方迁移或复制到另一个地方。可以为文件系统拍摄快照，从而导致块共享和写入拆分。有时文件系统可能会脱机或损坏。可以执行FSCK(FSCK用于检查和维护不一致的文件系统，若系统掉电或磁盘发生问题，可利用fsck命令对文件系统进行检查)对文件系统进行恢复。从终端用户和文件系统的存储实施的角度来看，文件系统老化的因素很多，包括但不限于以下各项：碎片化、累计操作、重用数据结构(inode、间接块等)、资源重用(索引节点编号)、重新分配数据块、从故障中恢复(离线)FSCK等。

[0028] 传统上，在新创建的文件系统中执行压力测试、耐久性测试、数据迁移测试、客户升级测试等，但是无法揭示文件系统、尤其是文件系统中的存储空间老化后的潜在问题和性能瓶颈。

[0029] 为了至少部分地解决上述问题以及其他潜在问题中的一个或者多个，本公开的示例实施例提出了一种管理文件系统的方案。在该方案中，首先在文件系统的存储空间的多个子存储空间中写入数据。然后对写入的数据进行操作以产生空间碎片，从而老化文件系统。以此方式，本方案可以在短时间内模拟长时间使用的客户文件系统，有助于揭示在新设计的文件系统中的潜在问题并且可以识别文件系统的性能瓶颈，提高了后续测试的测试效率和有效性。

[0030] 在下文中，将结合图1至图14更详细地描述本方案的具体示例。图1示出了根据本公开的实施例的备份系统100的示例的示意图。图1示出了本公开的实施例能够在其中被实现的示例环境100的框图。如图1所示，环境100包括主机110、存储管理器120以及文件系统130。应当理解，仅出于示例性的目的描述环境100的结构和功能，而不暗示对于本公开的范围的任何限制。例如，本公开的实施例还可以被应用到与环境100不同的环境中。

[0031] 文件系统130可以包括一个或多个存储空间，例如磁盘、光盘、机械硬盘(HDD)或固态盘(SSD)等。每个存储空间可以被划分成多个子存储空间。例如，每个子存储空间可以具有相同大小。根据所存储的不同数据类型或根据所划分的逻辑层，文件系统130可以包括各种类型的存储空间，例如，用于存储用户数据的存储空间(也称为“用户数据存储空间”)、用于存储与存储系统有关的元数据的存储空间(也称为“元数据存储空间”)等。元数据存储空间可以存储与存储系统有关的映射信息、索引信息、状态信息等，例如存储空间到物理盘的映射信息、存储空间的状态(诸如，正常状态或故障状态)等。存储空间中的多个子存储空间中所存储的数据可以是互相关联的。

[0032] 存储管理器120可以包括处理器121和存储器122。存储器122可以是任何目前已知或者将来开发的易失性存储介质、非易失性存储介质、或者两者的组合。存储管理器120可以被配置为管理文件系统130，并且处理来自主机110的输入/输出(I/O)请求。主机110可以是运行用户应用的任何物理计算机、虚拟机、服务器等等。

[0033] 主机110可以向存储管理器120发送I/O请求，例如用于从文件系统130中的存储空间移除数据和/或向其写入数据等。目标存储空间的元数据可以被存储在元数据存储空间中。响应于接收到来自主机110的I/O请求，存储管理器120可以首先从元数据存储空间获取目标存储空间的元数据，该元数据可以指示目标存储空间到物理盘的映射信息、目标存储空间的状态等。响应于该I/O请求为读请求，存储管理器120可以基于所获取的元数据向目标存储空间转发该I/O请求以从目标存储空间读取数据，并且将所读取的数据返回给主机110。响应于该I/O请求为写请求，存储管理器120可以基于所获取的元数据向目标存储空间转发该I/O请求以向目标存储空间写入数据。

[0034] 图2示出了根据本公开的实施例的用于管理存储盘的示例方法200的流程图。方法200例如可以由如图1所示的存储管理器120来执行。应当理解，方法200还可以包括未示出的附加动作和/或可以省略所示出的动作，本公开的范围在此方面不受限制。以下结合图1来详细描述方法200。

[0035] 如图2所示，在框210处，存储管理器120使多个数据块被写入所述文件系统130的存储空间，多个数据块中的每个数据块的大小不低于阈值大小。例如，存储管理器120可以响应于主机110发起的写入请求，将大小相同或者不同的数据块写入文件系统130的存储空间中的多个子存储空间中，这里，阈值大小可以是文件系统130的最小子存储空间大小，例

如为8KB。请注意，对于不同的文件系统，最小存储空间大小可以不同，因此可以设置不同的阈值小大小，本公开在此不做限制。

[0036] 在框220处，存储管理器120在存储空间130中创建至少一个空间碎片，至少一个空间碎片中的每个空间碎片的大小不超过所述阈值大小。例如，存储管理器120可以响应于主机110发起的擦除请求或者重写请求，对在210中写入文件系统130的存储空间中的多个子存储空间中的多个数据块进行移除、改变压缩率、重新写入等操作，以在文件系统中创建多个空间碎片。这里，空间碎片是指存储空间中的多个子存储空间中的不连续的空间，该空间例如小于最小存储空间大小，并且无法直接被写入数据。

[0037] 在下面将结合图3至图7详细描述上述步骤210和220。

[0038] 图3示出了根据本公开的实施例的管理文件系统的示意图300。存储管理器120使多个数据块被写入文件系统130的存储空间的子空间301、302、303、304、305、306和307中，在此多个字空间可以为新的空白的存储空间，本公开在此不做限制。如图130'所示，写入后的多个数据块在存储空间中各自占有子空间。然后存储管理器120使所写入的所述多个数据块中的至少两个数据块从所述存储空间被移除，例如，如130"所示，空白的矩形框代表被移除的数据块所占据的子空间，也即空间碎片，所示出的数据块在被移除前在存储空间中所占据的子空间不相邻。由此，创建了如130"中的空白所示的空间碎片，该空间碎片所占据的空间不大于所写入的数据块的大小且不相邻。

[0039] 在一个实施例中，存储管理器120经由I/O工具将大小相同的数据块(例如8KB，在此8KB为最小子存储空间大小)写入文件系统130的存储空间310中，然后将所写入的数据块间隔移除，并且在移除时禁用FSR等空间回收功能，该空间回收功能禁止将空白的子存储空间重新组合，从而实现更高的碎片率。请注意，相同大小的8KB的数据块仅仅是示例性的。

[0040] 大量的8KB的数据块也可以帮助配置3级的DUCH之类的复杂文件结构。

[0041] 在一个示例中，可以通过FIO工具实现上述8KB数据块的写入，FIO是一种IO测试工具，支持多引擎和多系统测试，在此仅仅作为示例，还可以应用诸如内部团队开发的内部工具或者诸如LDX工具的外部工具来实现数据的写入或者移除，本公开在此不做限制。FIO工具实现8KB数据块的写入的代码可以参照如下，其中文件可以是上述过程中的数据块：

[0042] [global]

[0043] do_verify＝0

[0044] ioengine＝libaio

[0045] iodepth＝32

[0046] direct＝1

[0047] readwrite＝write

[0048] openfiles＝40

[0049] refill_buffers＝0

[0050] numjobs＝1

[0051] create_on_open＝1

[0052] bs＝8k

[0053] [_mnt_testAged_2]\创建文件名称\

[0054] directory＝/mnt/obd2151/afp01/\加载指针\

[0055] filesize＝8k\创建文件大小\

[0056] nrfiles＝65536\创建文件的数目,如果数目太大,主机将内存不足\

[0057] create_serialize＝1\如果设置,将具有很高的碎片率,但是将花费更多的时间来创建文件\

[0058] 上述代码仅仅是示例性的,不旨在限制本公开。

[0059] 上面描述的情况为不存在数据块压缩和数据块重复存储的情况。数据压缩和重复数据删除是提高存储效率的常用功能,它们在文件系统中得到了广泛的使用,下面进一步结合图4至图7描述支持数据压缩和重复数据删除的文件系统的情况。

[0060] 图4示出了根据本公开的实施例的管理文件系统的环境的示意图400。响应于接收到来自主机110的I/O请求,存储管理器120可以首先从元数据存储空间获取元数据Leaf IB-A、Leaf IB-B、Leaf IB-C和Leaf IB-D/E,其中Leaf IB-B、Leaf IB-C和Leaf IB-D/E分别与压缩数据块B470、压缩数据块C(为了方便后续描述,在图中和下文被称为第一压缩数据块480)470、以及压缩数据块D相对应,与Leaf IB-A相对应的压缩数据块A在被移除。该多个元数据可以指示存储空间到物理盘的映射信息、目标存储空间的状态等,该多个元数据可以被进一步经由偏移量A、偏移量B、偏移量C、偏移量D和偏移量E被索引到元数据ILC-VBM-i,该元数据为VBM类型的元数据,ILC代表其可以存储压缩数据,这仅仅是一个示例,还可以应用其他类型的数据或者元数据,也可以以其他方式命名数据,本公开在此不做限制。

[0061] 以元数据ILC-VBM-i中索引1为例,其中w:10指示元数据Leaf IB-B在元数据ILC-VBM-i中存储的所有元数据40中的权重为10,数据长度B指示元数据数据Leaf IB-B所对应的压缩数据块B在存储空间中的数据长度。其中元数据Leaf IB-D/E存储有两个相同的压缩数据,其所对应的数据权重为20,并且其所对应的数据在数据,即压缩数据块D存储空间中仅占据一个字存储空间。通过不同数据的不同权重,可以快速地确定是哪个数据出现问题。Zipheader-B、Zipheader-C、Zipheader-D表示所存储的压缩数据块B470、第一压缩数据块480、以及压缩数据块D490的地址信息等。其中与元数据Leaf IB-A相对应的压缩数块A(未示出)被从文件系统130的存储空间中移除,图4中的Leaf IB-A上的叉形状表示其被移除,所以其在元数据ILC-VBM-i中的索引0中的权重、长度和偏移量为0,并且由于其被移除在存储空间中存在空闲存储空间460。压缩数块A被移除仅为了描述本公开的实施例,其不旨在限制。在下面的描述中,将不再赘述元数据,而直接描述存储管理器120对存储空间中的数据块的操作。

[0062] 图5示出了根据本公开的实施例的管理文件系统的示意图500。首先参考上述关于图4描述的,存储管理器120可以响应于主机110的写入请求,通过第一压缩率来压缩多个数据块中的第一数据块,以得到第一压缩数据块480,并且将其写入文件系统130的存储空间中的第一子空间中。写入过程可以参照图4中的描述,在此不再赘述。后续描述的第一子空间、第二子空间等是文件系统130的存储空间中的多个子存储空间,其可以是大小相同的或者不同的存储空间,本公开在此不做限制。

[0063] 例如,存储管理器120可以将8KB的第一数据块通过第一压缩率80％得到大小为6.4KB的第一压缩数据块480,然后将其写入第一子空间,这里第一子空间相应地在存储空间占据6.4KB的存储空间。上述数据块的大小和压缩率仅仅是示例性的,根据不同的存储和数据结构,还可以应用其他的数据大小和压缩率大小。

[0064] 然后，存储管理器120可以使第一压缩数据块从第一子空间中被移除，例如存储管理器120可以将与该第一压缩数据块480相对应的Offset-C移除，然后第一压缩数据块480从第一子空间被移除并且用于Offset-C的元数据ILC-VBM-i被移除并且第一子空间被释放。接着存储管理器120通过大于第一压缩率的第二压缩率来重新压缩所述第一数据块，以得到第二压缩数据块580。由于第一压缩率小于第二压缩率，所以第二压缩数据块580的大小小于第一压缩数据块480，因此存储管理器120可以将第二压缩数据块580重新写入第一子空间，因此产生空间碎片510。

[0065] 例如，继续上面示例描述，存储管理器120可以将8KB的第一数据块通过第二压缩率70％得到大小为5.6KB的第二压缩数据块580，然后存储管理器120可以将5.6KB的第二压缩数据块580写入存储空间中的第一子空间，并且数据长度C被更新为5.6KB，由此创建了大小为6.4KB-5.6KB＝0.8KB的空间碎片510，在此空间碎片是指分离的存储空间，其在没有空间回收的操作的情况下无法再存储数据。

[0066] 在一个备选实施例中，第一压缩率可以为100％，即第一数据块未被压缩，第二压缩率可以为任何小于100％的压缩率，从而创建空闲的存储空间。

[0067] 还可以对上述压缩数据块B和压缩数据块D执行类似的操作，在此不再赘述。

[0068] 通过反复多次以不同的压缩率同一数据块或者不同数据块，可以创建诸如最小存储空间的10％甚至更小的空间碎片，这些空间碎片无法被进一步写入数据，从而老化文件系统。此外，通过不同压缩数据率的重写IO可以触发数据块的写入拆分，在原有的元数据不是和存储压缩数据块的情况下，还可以导致有关实现数据块共享的元数据的增加和删除，从而进一步老化了文件系统。

[0069] 图6示出了根据本公开的实施例的管理文件系统的示意图600。图6与图5的不同之处在于，重新写入的压缩数据块的大小大于原先写入的压缩数据块的大小，所以其无法写入原来所占据的存储空间。存储管理器120首先响应于主机110的多个写入请求，执行多个写入操作，存储管理器120使多个数据块中的第二数据块被写入存储空间中的第二子空间，然后以第三压缩率来压缩多个数据块中的第三数据块，以得到第三压缩数据块，以及使第三压缩数据块被写入存储空间中的第三子空间，第二数据块的大小大于第三数据块的大小，第二子空间和第三子空间不相邻。

[0070] 例如，首先参考图5描述，存储管理器120可以首先将8KB的压缩数据块A520写入文件系统130的存储空间中的第二子空间，然后将可以将8KB的第三数据块通过第三压缩率70％得到大小为5.6KB的第三压缩数据块(图中示出为第二压缩数据快)，然后存储管理器120可以将5.6KB的第三压缩数据块写入存储空间中的第三子空间，如图所示压缩数据块A和第三子空间之间还有压缩数据块B。

[0071] 然后，存储管理器120使第二数据从第二子空间中被移除，使第三压缩数据块从第三子空间中被移除，以小于第三压缩率的第四压缩率来压缩第三数据块，以得到第四压缩数据块，最后使所述第四压缩数据块被写入所述第二子空间。

[0072] 例如，存储管理器120可以将8KB的压缩数据块A520从文件系统130的存储空间中的第二子空间移除，如图4所示，移除后的VBM元数据中的各项数据都更新为0，然后使5.6KB的第三压缩数据块从第三子空间被移除。存储管理器120通过第四压缩率90％得到大小为7.2KB的第四压缩数据块610，由于第四数据块610大于第三子空间的大小，即第三压缩数据

块的大小5.6KB,所以其无法被写入第三子空间。由于第二子空间的大小,即压缩数据块A520的大小8KB大于第四数据块610的大小,存储管理器120可以将7.2KB的第四压缩数据块610写入存储空间中的第二子空间,并且数据长度C被更新为7.2KB。由此,如图6所示,创建了大小为8KB-7.2KB＝0.8KB的空间碎片620和大小为5.6KB的空间碎片630。

[0073] 在一个备选实施例中,由于元数据ILC-VBM-i尽可以存储12个(数字12是示例性的,其仅仅代表不同类型的元数据的属性)对应的压缩数据条目,如果元数据ILC-VBM-i没有合适的块来存储上述新写入的压缩数据,则应创建一个新的元数据VBM来存储压缩数据。

[0074] 关于上述描述了通过对压缩数据的写入和移除来创建各种空间碎片,下面描述关于重复数据的情况。

[0075] 图7示出了根据本公开的实施例的管理文件系统的示意图700。如上文关于图4所讨论的,Offset-D和Offset-E为重复数据,其共享一个数据块。存储管理器120以不同的压缩/重复数据删除速率重写Offset-E,导致Offset-D和Offset-E无法共享一个数据块。Offset-E必须进行写拆分,即重新写入经重新压缩的数据块,写入过程与上文图5至图7所描述的重写Offset-C相似,在此不再赘述。存储管理器120还在ILC-VBM-i中添加有关Offset-E的新记录,例如其权重、长度等。

[0076] 如图7所示,以不同压缩率重写的压缩数据块E710占据了先前被移除的压缩数据块A所占据的存储空间,由此创建了空间碎片720。

[0077] 在一个实施例中,存储管理器120还可以对压缩数据块B470、压缩数据块C730执行压缩重写等上文所描述的操作以创建更多的空间碎片。

[0078] 在一个备选实施例中,同时执行上述数据块的写入、移除操作,可以导致具有不同文件大小IO的快照创建/删除操作以及创建更多的空间碎片,这些操作可以测试更多元数据区域,例如inode(inode缓存,具有生成编号的inode ID重用)并且加速文件系统的老化。

[0079] 图8示出了根据本公开的实施例的在经老化的文件系统上进行测试的示意图800。其中可以通过上面关于图2至图7所描述的空间碎片创建操作来老化文件系统1(810)、文件系统2(820)至文件系统n(830)。测试用例1(840)、测试用例2(850)和测试用例N(860)可以包括例如单元测试,功能测试,集成测试,压力测试和耐力测试。可以存在两种方式进行文件系统的老化和测试。方案一(如实线所示),存储管理器120可以首先将多个文件系统老化,然后经由多个测试用例对经老化的文件系统进行测试。方案二(如虚线所示),存储管理器120可以在对文件系统进行老化的同时对其进行测试。

[0080] 图9示出了根据本公开的实施例的用于检测文件系统的老化过程的设备的示意图900。本公开提出了在文件系统130的老化过程中使用Elastic Stack系统来检测和记录日志跟踪和事件。Filebeat920是安装在文件系统的存储空间上的工具,用于读取目标跟踪并使用对背压敏感的协议将数据发送到Logstash930。Logstash930可以提取和转换数据,然后加载到Elasticsearch940,Elasticsearch940是Elastic core的核心软件,用于记录转换后的日志跟踪和事件。最后Kibana950将在GUI中显示日志跟踪的详细信息,它可以告诉用户事件发生的时间,地点和方式,以及对老化操作的反馈。老化配置文件(工作负载和操作)将根据从Elastic Stack获得的结果进行修改。

[0081] 例如,存储管理器120可以经由上述系统来获取文件系统130的数据碎片、元数据碎片、自由空间碎片和内存碎片等以确定文件系统130的老化率。存储管理器120还可以经

由上述系统获取冲突操作(块共享,块重新分配,空间缩小/扩展等)和中断操作的频率等来确定文件系统130的老化率。

[0082] 在一个实施例中,存储管理器120可以通过如下步骤来老化文件系统130:(1)首先使用内存和磁盘上的客户数据来配置文件系统130。客户数据例如根据客户经常使用的数据结构(哈希表,DB,DUCH,dentry等)、本地文件系统还是全局文件系统而被预先限定;(2)经由限定的工作负载配置文件(IO工作负载和文件系统操作),以使用内部工具和外部(FIO,LDX)工具老化文件系统130,具体的老化方法可以参考图2至图7描述的创建碎片的方法;(3)经由Elastic Stack系统监测并确定文件系统130的老化率;(4)基于不同的测试目的,不断进行步骤(2)的操作以获得预定的老化率。

[0083] 图10至图13示出了根据本公开的实施例的文件系统性能的示意图1000、1100、1200和1300。其中曲线1030、1130、1230和1330表示用8KB的数据块填充50GB的文件系统,然后将数据块间隔删除,配置该文件系统需要15个小时。曲线1020、1120、1220和1320表示用16KB的数据块填充50GB的文件系统,然后将数据块间隔删除。配置该文件系统需要6个小时。曲线1010、1110、1210和1310表示新创建的50GB文件系统。可以看出,利用8KB数据块并且间隔删除的老化方式,即曲线1030、1130、1230和1330在文件系统响应时间上最长,I/O最小,带宽最低,每秒I/O最高。即对于该特性文件系统,使用8KB数据填充并间隔删除的老化效果最好。该示例仅仅是示例性的,还可以根据不同的文件系统和数据结构采取不同大小的小数据块进行老化操作。

[0084] 上述性能图仅仅是示例性的,在一个实施例中,存储管理器120还可以下通过以下至少一项来确定文件系统130的性能:文件系统的响应时间、文件系统读取/写入数据的平均带宽、文件系统每秒进行读写操作的数目以及文件系统的故障率。

[0085] 图14示出了可以用来实施本公开内容的实施例的示例设备1400的示意性框图。例如,如图1所示的存储管理器120可以由设备1400来实施。如图所示,设备1400包括中央处理单元(CPU)1401,其可以根据存储在只读存储器(ROM)1402中的计算机程序指令或者从存储空间1408加载到随机访问存储器(RAM)1403中的计算机程序指令,来执行各种适当的动作和处理。在RAM 1403中,还可以存储设备1400操作所需的各种程序和数据。CPU 1401、ROM 1402以及RAM 1403通过总线1404彼此相连。输入/输出(I/O)接口1407也连接至总线1404。

[0086] 设备1400中的多个部件连接至I/O接口1470,包括:输入单元1406,例如键盘、鼠标等;输出单元1407,例如各种类型的显示器、扬声器等;存储空间1408,例如磁盘、光盘等;以及通信单元1409,例如网卡、调制解调器、无线通信收发机等。通信单元1409允许设备1400通过诸如因特网的计算机网络和/或各种电信网络与其他设备交换信息/数据。

[0087] 上文所描述的各个过程和处理,例如方法200,可由处理单元1401执行。例如,在一些实施例中,方法200可被实现为计算机软件程序,其被有形地包含于机器可读介质,例如存储空间1408。在一些实施例中,计算机程序的部分或者全部可以经由ROM 1402和/或通信单元1409而被载入和/或安装到设备1400上。当计算机程序被加载到RAM1403并由CPU 1401执行时,可以执行上文描述的方法200的一个或多个动作。

[0088] 本公开可以是方法、装置、系统和/或计算机程序产品。计算机程序产品可以包括计算机可读存储介质,其上载有用于执行本公开的各个方面的计算机可读程序指令。

[0089] 计算机可读存储介质可以是可以保持和存储由指令执行设备使用的指令的有形

设备。计算机可读存储介质例如可以是——但不限于——电存储设备、磁存储设备、光存储设备、电磁存储设备、半导体存储设备或者上述的任意合适的组合。计算机可读存储介质的更具体的例子(非穷举的列表)包括:便携式计算机盘、硬盘、随机存取存储器(RAM)、只读存储器(ROM)、可擦式可编程只读存储器(EPROM或闪存)、静态随机存取存储器(SRAM)、便携式压缩盘只读存储器(CD-ROM)、数字多功能盘(DVD)、记忆棒、软盘、机械编码设备、例如其上存储有指令的打孔卡或凹槽内凸起结构、以及上述的任意合适的组合。这里所使用的计算机可读存储介质不被解释为瞬时信号本身,诸如无线电波或者其他自由传播的电磁波、通过波导或其他传输媒介传播的电磁波(例如,通过光纤电缆的光脉冲)、或者通过电线传输的电信号。

[0090] 这里所描述的计算机可读程序指令可以从计算机可读存储介质下载到各个计算/处理设备,或者通过网络、例如因特网、局域网、广域网和/或无线网下载到外部计算机或外部存储设备。网络可以包括铜传输电缆、光纤传输、无线传输、路由器、防火墙、交换机、网关计算机和/或边缘服务器。每个计算/处理设备中的网络适配卡或者网络接口从网络接收计算机可读程序指令,并转发该计算机可读程序指令,以供存储在各个计算/处理设备中的计算机可读存储介质中。

[0091] 用于执行本公开操作的计算机程序指令可以是汇编指令、指令集架构(ISA)指令、机器指令、机器相关指令、微代码、固件指令、状态设置数据、或者以一种或多种编程语言的任意组合编写的源代码或目标代码,所述编程语言包括面向对象的编程语言—诸如Smalltalk、C++等,以及常规的过程式编程语言—诸如“C”语言或类似的编程语言。计算机可读程序指令可以完全地在用户计算机上执行、部分地在用户计算机上执行、作为一个独立的软件包执行、部分在用户计算机上部分在远程计算机上执行、或者完全在远程计算机或服务器上执行。在涉及远程计算机的情形中,远程计算机可以通过任意种类的网络—包括局域网(LAN)或广域网(WAN)—连接到用户计算机,或者,可以连接到外部计算机(例如利用因特网服务提供商来通过因特网连接)。在一些实施例中,通过利用计算机可读程序指令的状态信息来个性化定制电子电路,例如可编程逻辑电路、现场可编程门阵列(FPGA)或可编程逻辑阵列(PLA),该电子电路可以执行计算机可读程序指令,从而实现本公开的各个方面。

[0092] 这里参照根据本公开实施例的方法、装置(系统)和计算机程序产品的流程图和/或框图描述了本公开的各个方面。应当理解,流程图和/或框图的每个方框以及流程图和/或框图中各方框的组合,都可以由计算机可读程序指令实现。

[0093] 这些计算机可读程序指令可以提供给通用计算机、专用计算机或其它可编程数据处理装置的处理单元,从而生产出一种机器,使得这些指令在通过计算机或其它可编程数据处理装置的处理单元执行时,产生了实现流程图和/或框图中的一个或多个方框中规定的功能/动作的装置。也可以把这些计算机可读程序指令存储在计算机可读存储介质中,这些指令使得计算机、可编程数据处理装置和/或其他设备以特定方式工作,从而,存储有指令的计算机可读介质则包括一个制造品,其包括实现流程图和/或框图中的一个或多个方框中规定的功能/动作的各个方面的指令。

[0094] 也可以把计算机可读程序指令加载到计算机、其它可编程数据处理装置、或其它设备上,使得在计算机、其它可编程数据处理装置或其它设备上执行一系列操作步骤,以产

生计算机实现的过程，从而使得在计算机、其它可编程数据处理装置、或其它设备上执行的指令实现流程图和/或框图中的一个或多个方框中规定的功能/动作。

[0095] 附图中的流程图和框图显示了根据本公开的多个实施例的系统、方法和计算机程序产品的可能实现的体系架构、功能和操作。在这点上，流程图或框图中的每个方框可以代表一个模块、程序段或指令的一部分，所述模块、程序段或指令的一部分包含一个或多个用于实现规定的逻辑功能的可执行指令。在有些作为替换的实现中，方框中所标注的功能也可以以不同于附图中所标注的顺序发生。例如，两个连续的方框实际上可以基本并行地执行，它们有时也可以按相反的顺序执行，这依所涉及的功能而定。也要注意的是，框图和/或流程图中的每个方框、以及框图和/或流程图中的方框的组合，可以用执行规定的功能或动作的专用的基于硬件的系统来实现，或者可以用专用硬件与计算机指令的组合来实现。

[0096] 以上已经描述了本公开的各实施例，上述说明是示例性的，并非穷尽性的，并且也不限于所披露的各实施例。在不偏离所说明的各实施例的范围和精神的情况下，对于本技术领域的普通技术人员来说许多修改和变更都是显而易见的。本文中所用术语的选择，旨在最好地解释各实施例的原理、实际应用或对市场中的技术的技术改进，或者使本技术领域的其它普通技术人员能理解本文披露的各实施例。

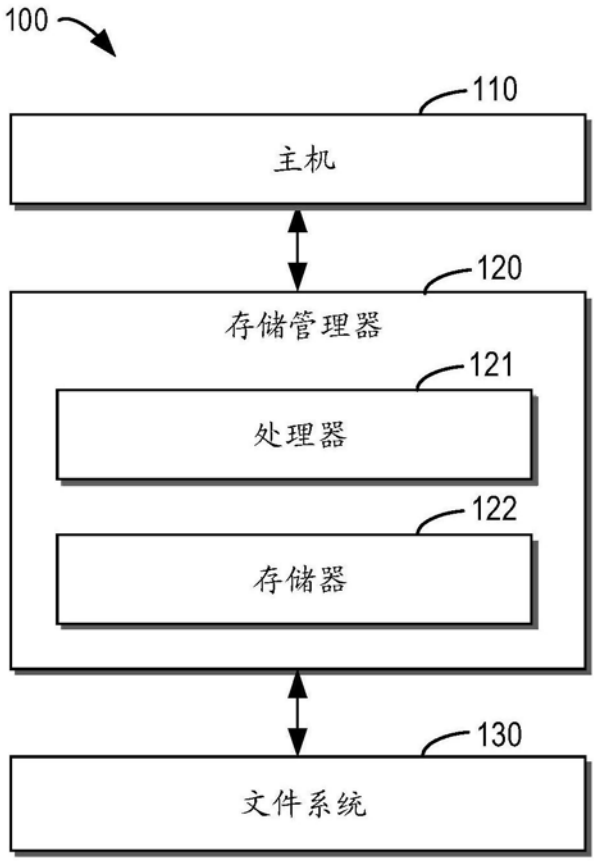

100
110
主机
120
存储管理器
121
处理器
122
存储器
130
文件系统

图1

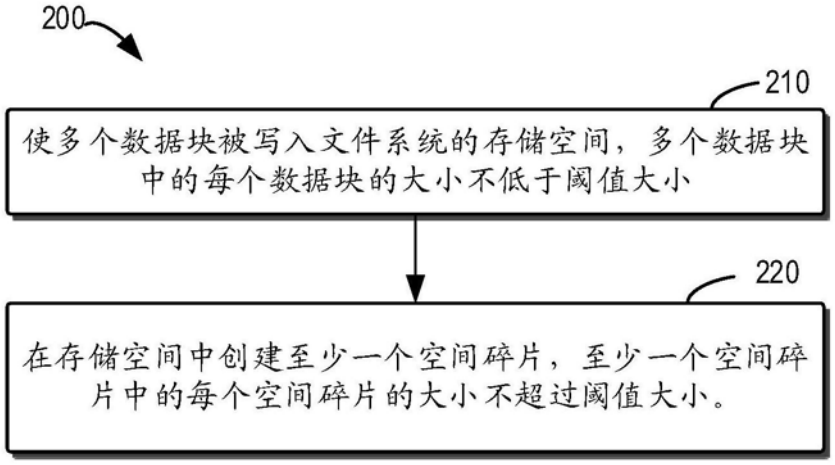

200
210
使多个数据块被写入文件系统的存储空间，多个数据块中的每个数据块的大小不低于阈值大小
220
在存储空间中创建至少一个空间碎片，至少一个空间碎片中的每个空间碎片的大小不超过阈值大小。

图2

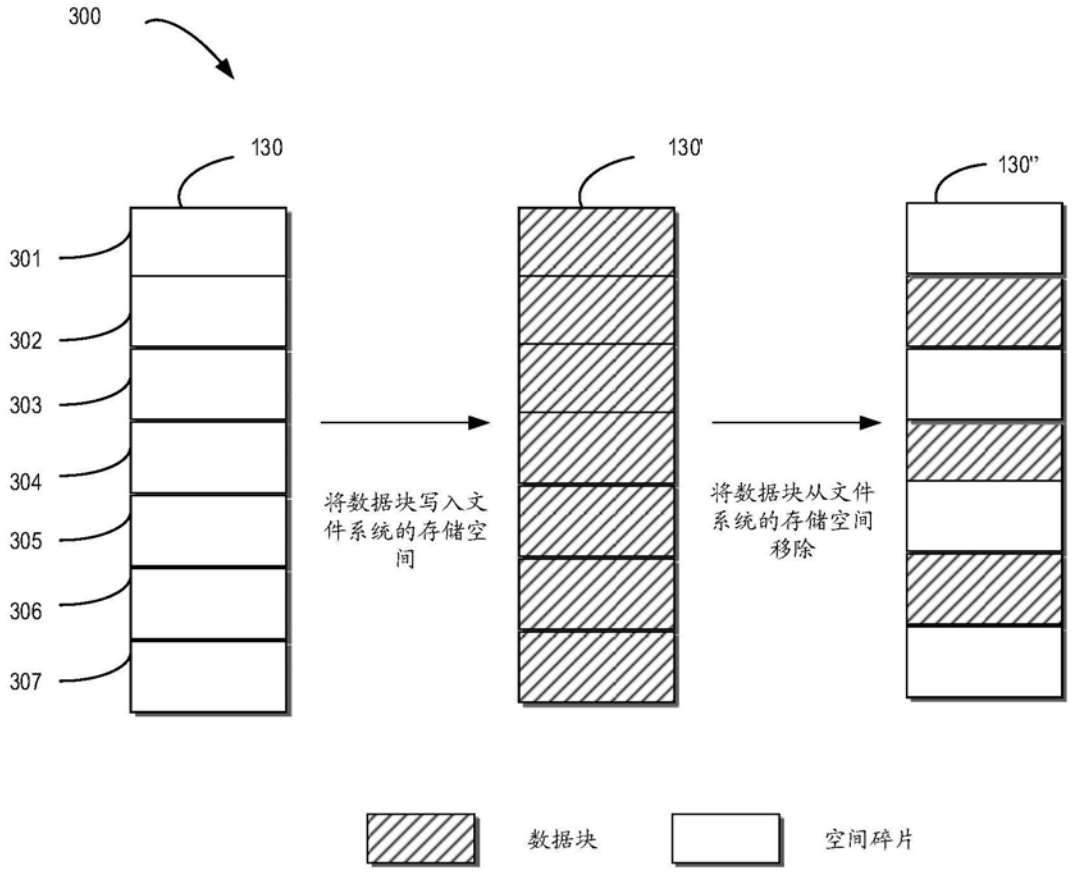

300
130
130'
130''
301
302
303
304
305
306
307
将数据块写入文件系统的存储空间
将数据块从文件系统的存储空间移除
数据块
空间碎片

图3

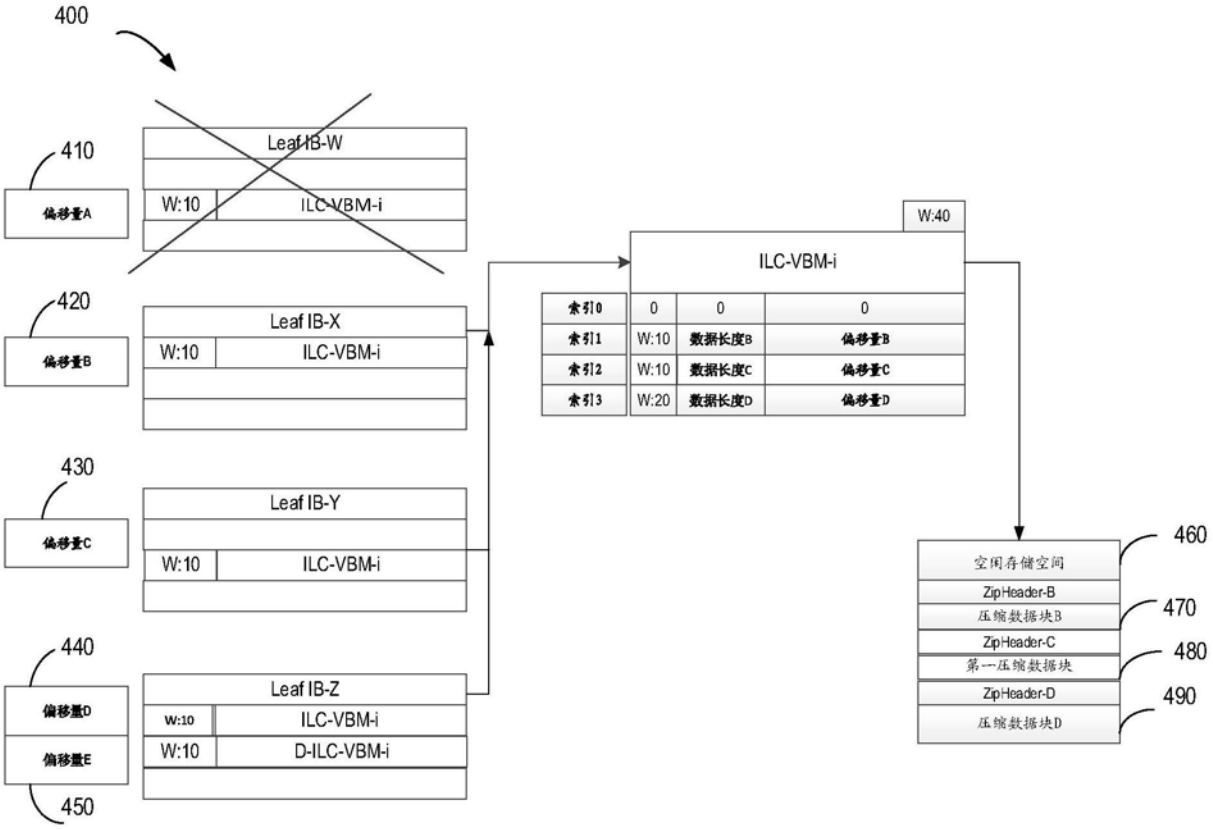

400
410
偏移量A
Leaf IB-W
W:10
ILC-VBM-i
420
偏移量B
Leaf IB-X
W:10
ILC-VBM-i
430
偏移量C
Leaf IB-Y
W:10
ILC-VBM-i
440
偏移量D
偏移量E
Leaf IB-Z
W:10
ILC-VBM-i
W:10
D-ILC-VBM-i
450
W:40
ILC-VBM-i
索引0 0 0 0
索引1 W:10 数据长度B 偏移量B
索引2 W:10 数据长度C 偏移量C
索引3 W:20 数据长度D 偏移量D
460
空闲存储空间
ZipHeader-B
压缩数据块B
470
ZipHeader-C
第一压缩数据块
480
ZipHeader-D
压缩数据块D
490

图4

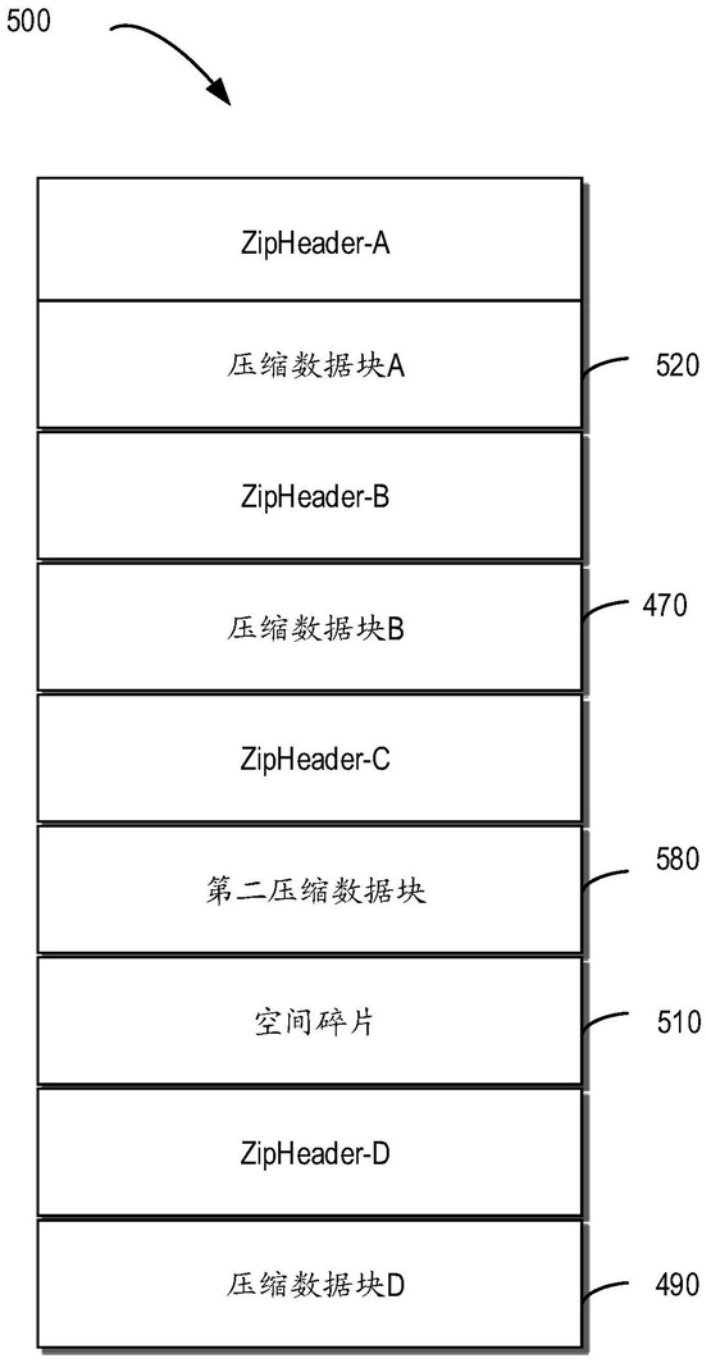

500
ZipHeader-A
压缩数据块A
520
ZipHeader-B
压缩数据块B
470
ZipHeader-C
第二压缩数据块
580
空间碎片
510
ZipHeader-D
压缩数据块D
490

图5

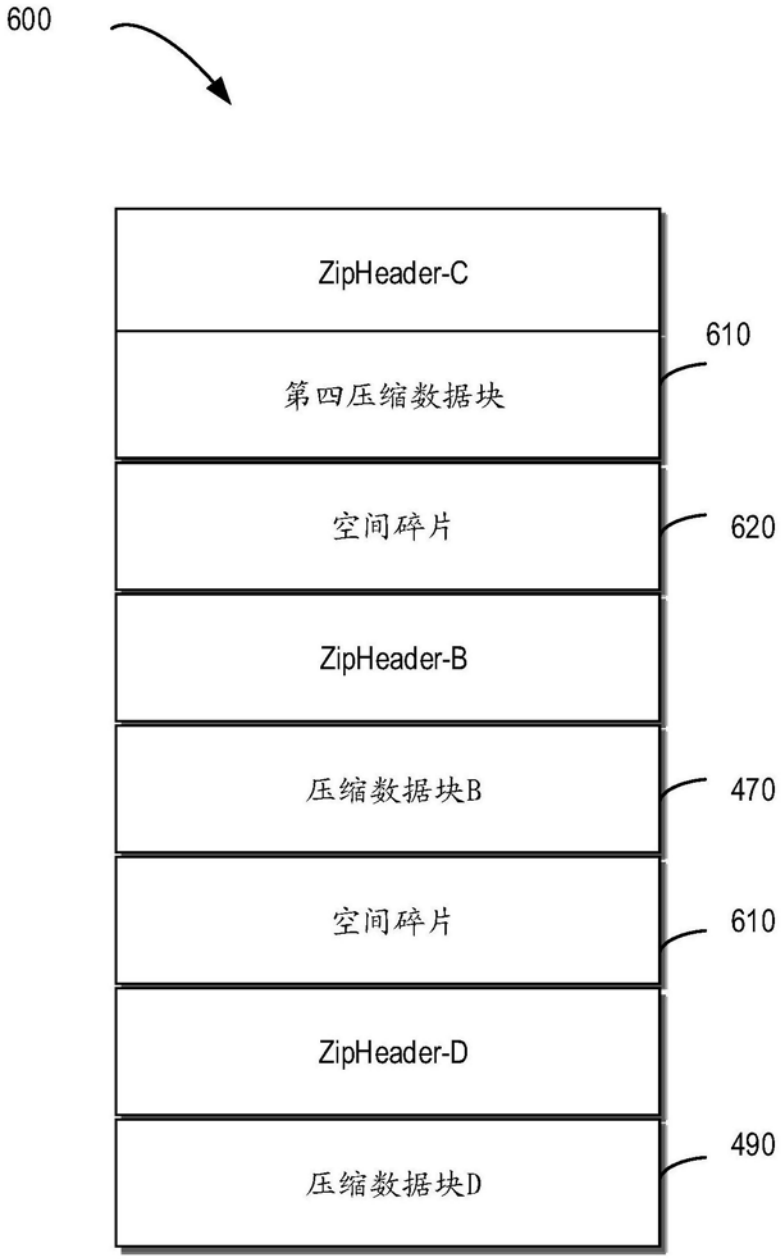

600
ZipHeader-C
610
第四压缩数据块
空间碎片
620
ZipHeader-B
压缩数据块B
470
空间碎片
610
ZipHeader-D
压缩数据块D
490

图6

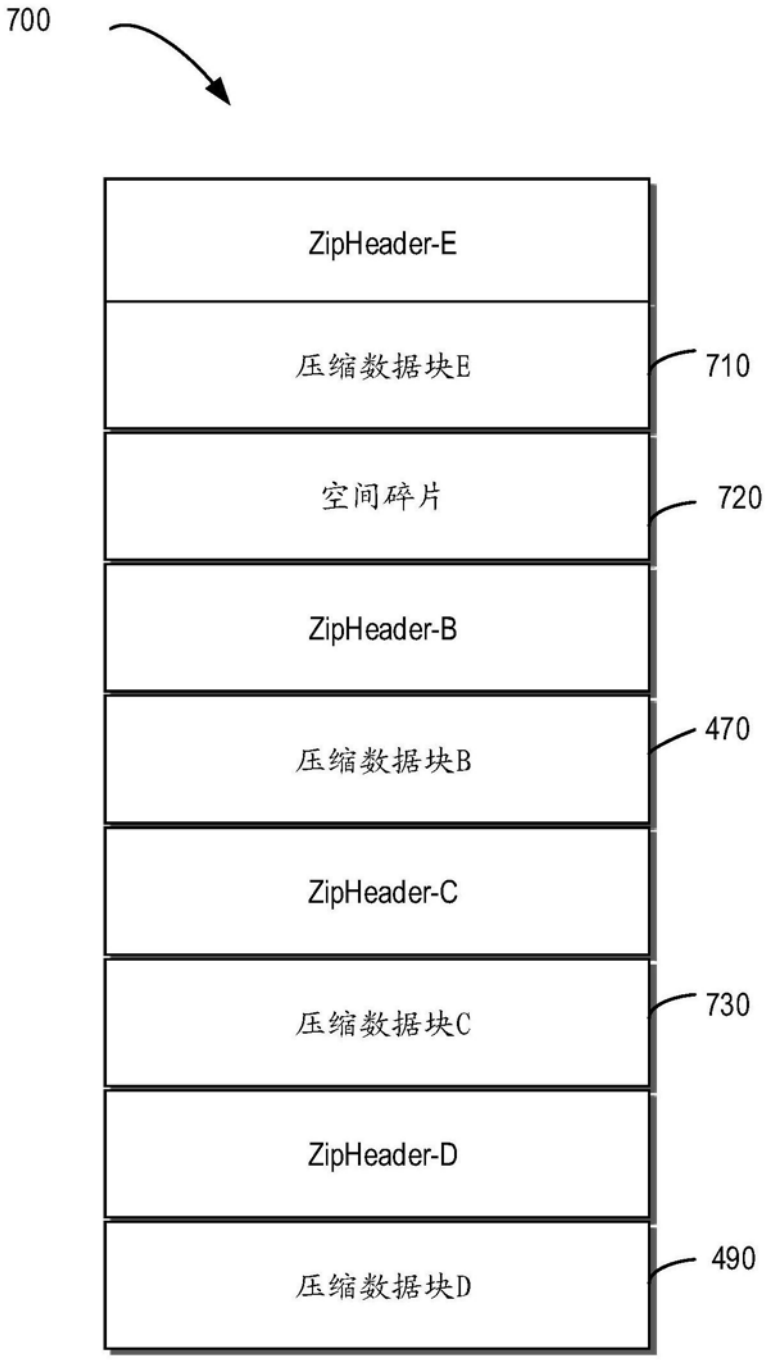

700
ZipHeader-E
压缩数据块E
710
空间碎片
720
ZipHeader-B
压缩数据块B
470
ZipHeader-C
压缩数据块C
730
ZipHeader-D
压缩数据块D
490

图7

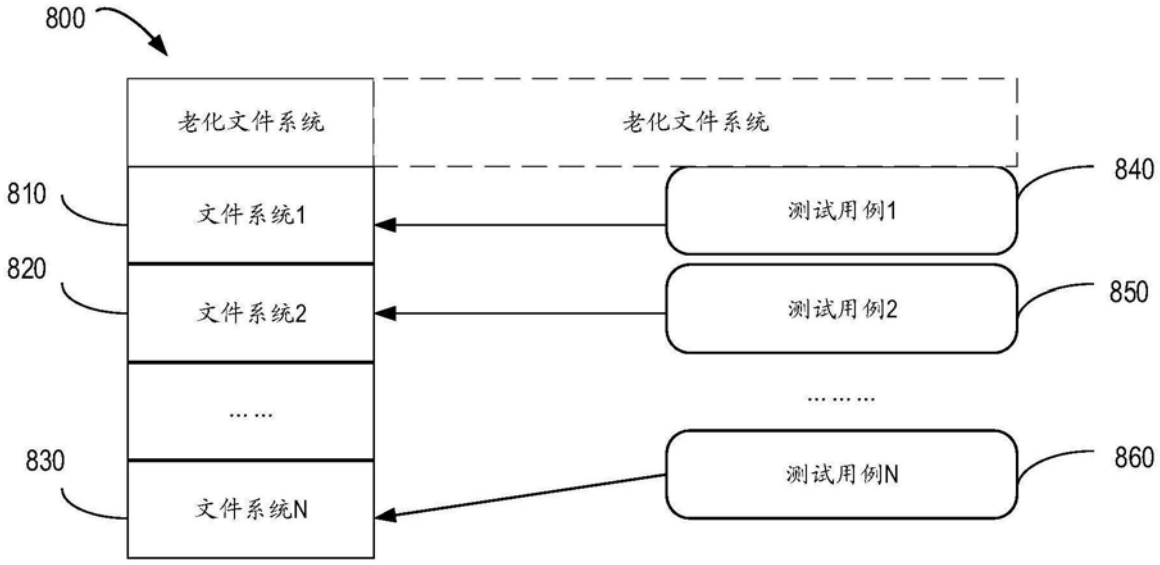

800
老化文件系统
老化文件系统
810
文件系统1
820
文件系统2
……
830
文件系统N
840
测试用例1
850
测试用例2
……
860
测试用例N

图8

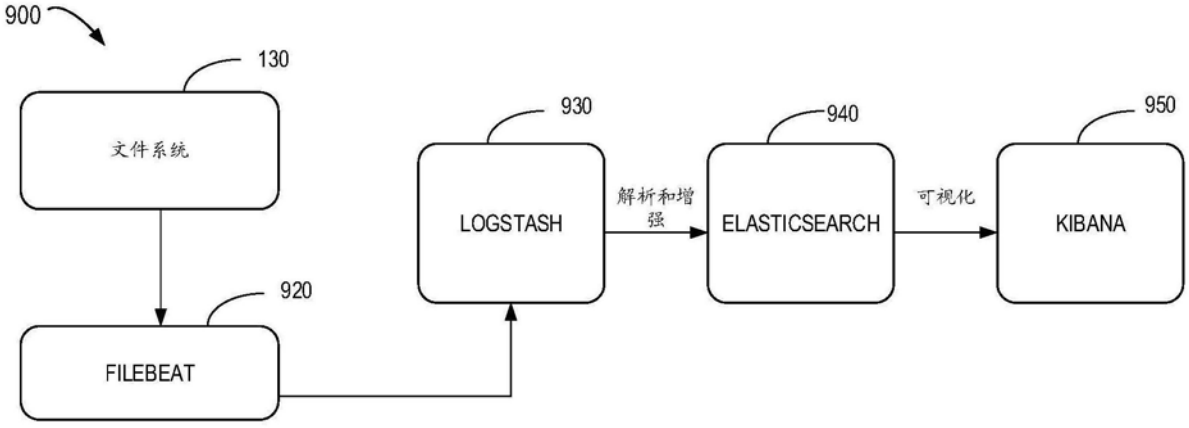

900
130
文件系统
920
FILEBEAT
930
LOGSTASH
解析和增强
940
ELASTICSEARCH
可视化
950
KIBANA

图9

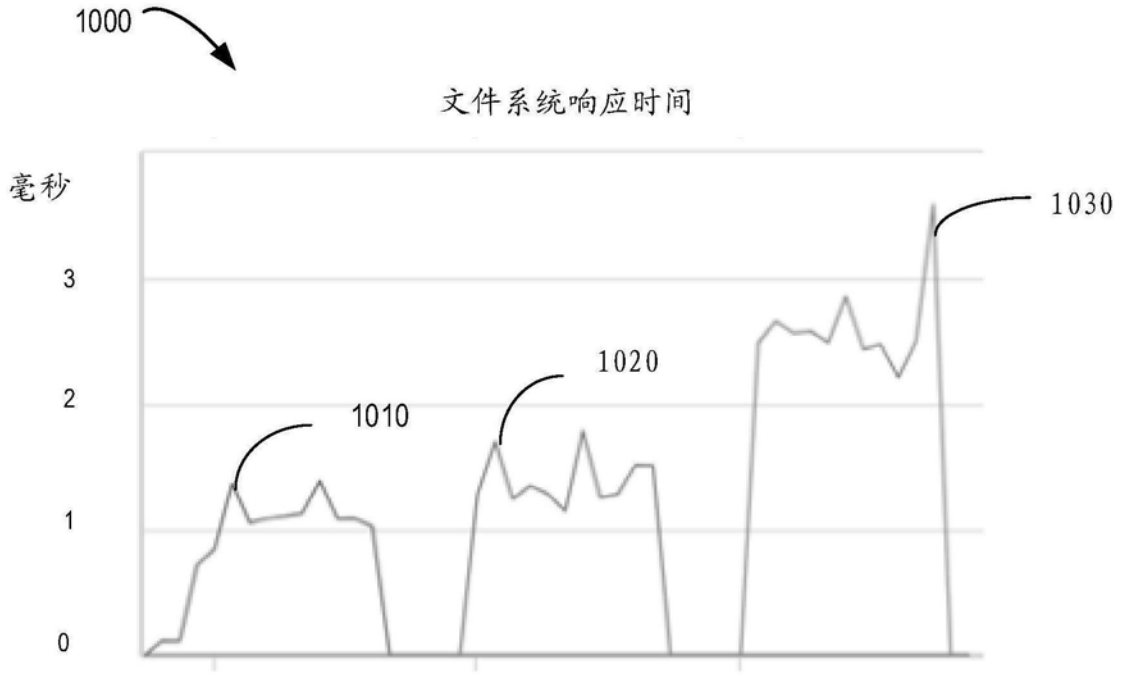

1000
文件系统响应时间
毫秒
3
2
1
0
1010
1020
1030

图10

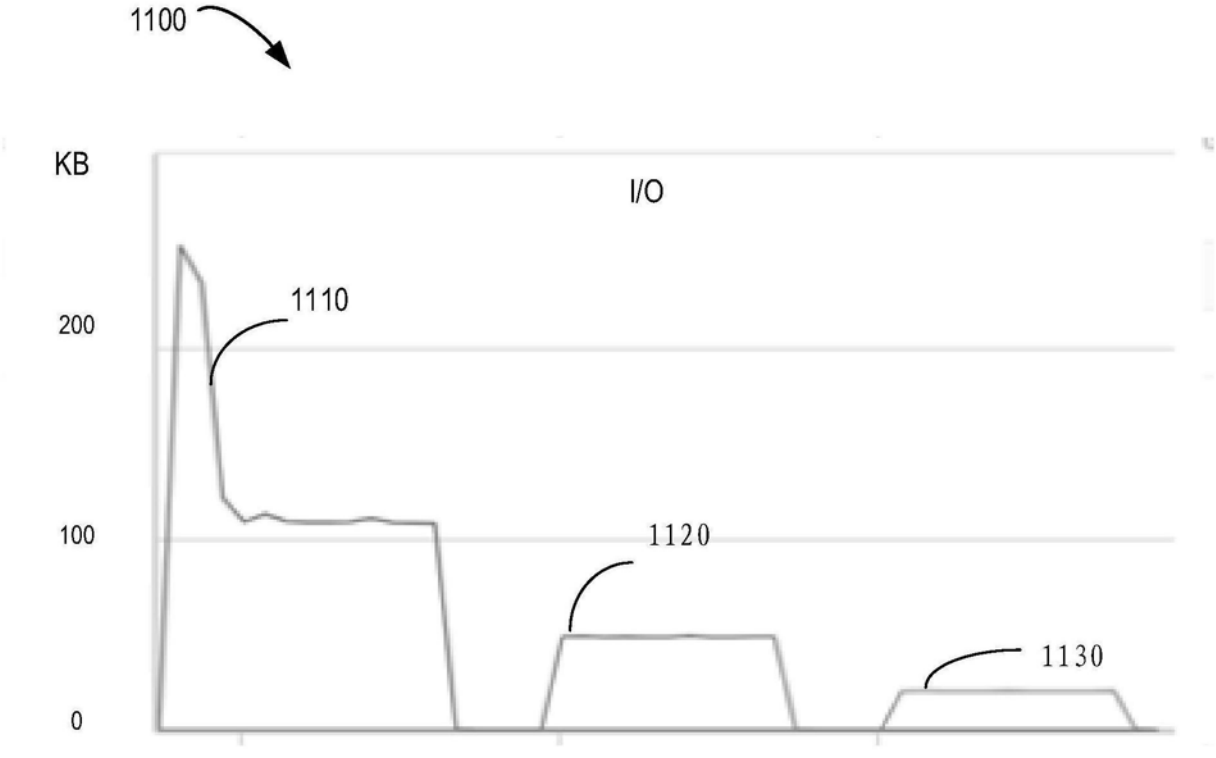

1100
KB
I/O
200
100
0
1110
1120
1130

图11

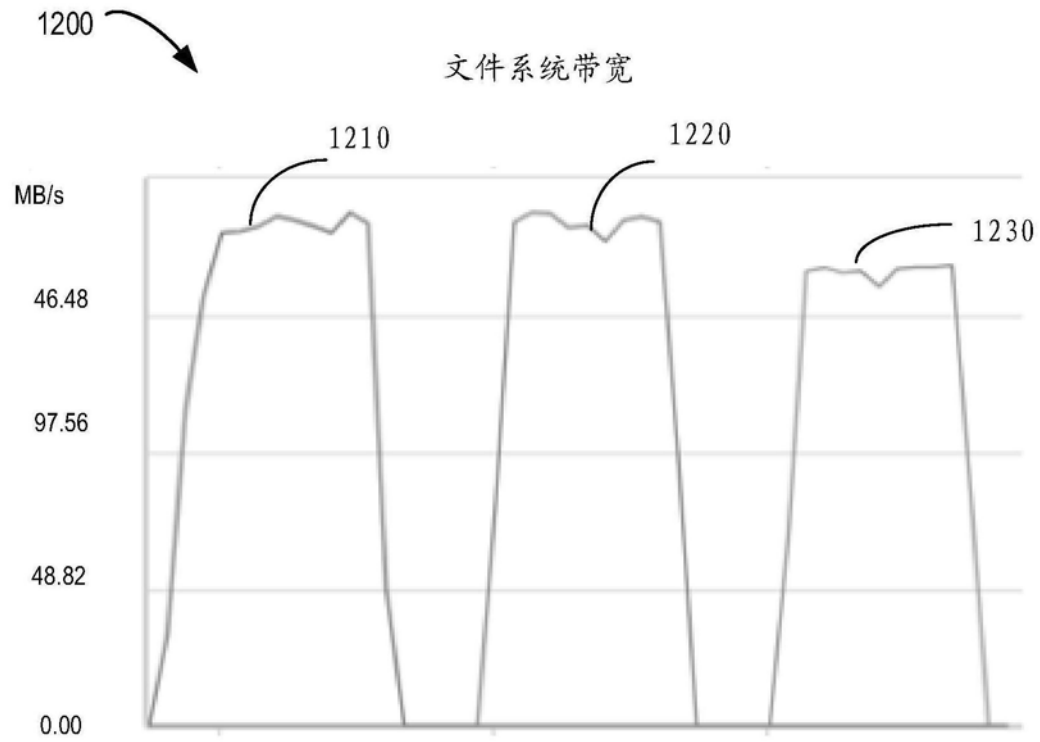

1200
文件系统带宽
1210
1220
1230
MB/s
46.48
97.56
48.82
0.00

图12

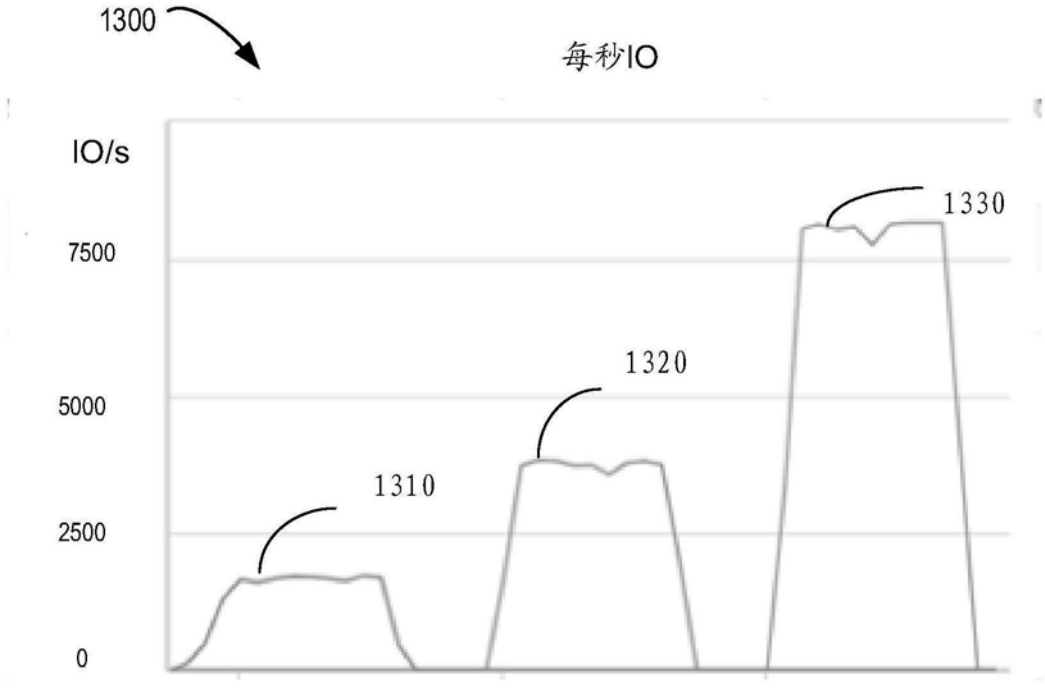

1300
每秒IO
IO/s
7500
5000
2500
0
1330
1320
1310

图13

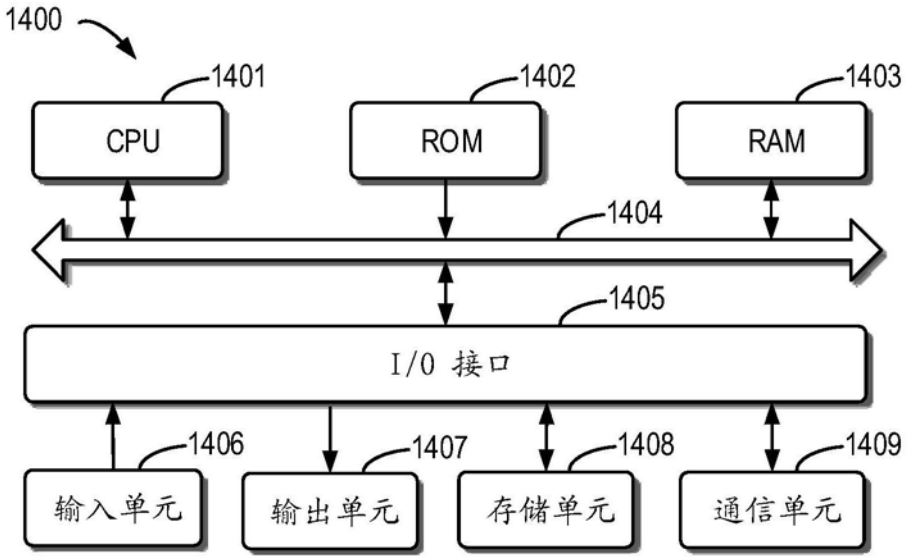

1400
1401
CPU
1402
ROM
1403
RAM
1404
1405
I/O 接口
1406
输入单元
1407
输出单元
1408
存储单元
1409
通信单元

图14