(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 105843905 B
(45)授权公告日 2019.06.11

(21)申请号 201610169762.3

(22)申请日 2011.11.21

(65)同一申请的已公布的文献号
申请公布号 CN 105843905 A

(43)申请公布日 2016.08.10

(30)优先权数据
61/415,928 2010.11.22 US

(62)分案原申请数据
201180056305.9 2011.11.21

(73)专利权人 日立数据管理有限公司
地址 美国加利福尼亚州

(72)发明人 丹尼尔·J·N·皮肯
尼尔·柏林顿

(74)专利代理机构 北京银龙知识产权代理有限公司 11243
代理人 范胜杰 杨继平

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

(56)对比文件
US 2007106706 A1,2007.05.10,
US 2005246397 A1,2005.11.03,
US 2005246503 A1,2005.11.03,
CN 101710323 A,2010.05.19,
CN 101743546 A,2010.06.16,
CN 101178677 A,2008.05.14,

审查员 谢晓琦

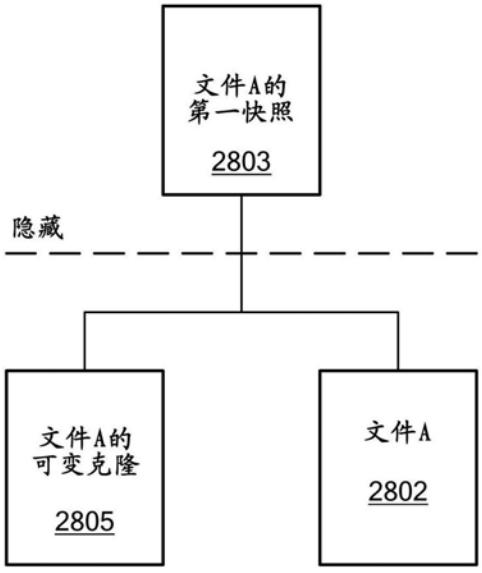
权利要求书2页 说明书27页 附图25页

(54)发明名称

用于管理文件系统中的文件系统对象的设备和方法

(57)摘要

公开了一种用于管理文件系统中的文件系统对象的设备和方法。用于管理文件系统中的文件系统对象的设备包括存储装置,用于存储文件系统的对象,该文件系统包括快照对象和文件系统对象,快照对象包括引用数据块的第一指针,并且文件系统对象包括引用数据块的第二指针。通过快照对象的第一指针所引用的数据块包括由快照对象拥有的数据块;并且用于管理文件系统中的文件系统对象的设备还包括处理装置,用于执行:针对由快照对象拥有的并且通过快照对象的第一指针以及文件系统对象的第二指针所引用的每个数据块,将每个数据块的所有权从快照对象转移到文件系统对象,以及在这样的所有权转移之后,从文件系统中移除快照对象。



CN 105843905 B

1.一种用于管理文件系统中的文件系统对象的设备，

所述用于管理文件系统中的文件系统对象的设备包括存储装置，所述存储装置用于存储文件系统的对象，所述文件系统包括快照对象(2809)和文件系统对象(2802')，所述快照对象包括引用一个或多个数据块的一个或多个第一指针，并且所述文件系统对象(2802')包括引用一个或多个数据块的一个或多个第二指针，

其中，通过所述快照对象(2809)的一个或多个第一指针所引用的一个或多个数据块包括由所述快照对象(2809)拥有的一个或多个数据块；并且

所述用于管理文件系统中的文件系统对象的设备还包括一处理装置，所述处理装置用于执行：

-针对由所述快照对象(2809)拥有的并且通过所述快照对象(2809)的一个或多个第一指针以及所述文件系统对象(2802')的一个或多个第二指针所引用的每个数据块，将每个数据块的所有权从所述快照对象(2809)转移到所述文件系统对象(2802')，以及

-在这样的从所述快照对象(2809)到所述文件系统对象(2802')的所有权转移之后，从所述文件系统中移除所述快照对象(2809)。

2.根据权利要求1所述的用于管理文件系统中的文件系统对象的设备，其中，

所拥有的数据块的所有权指示负责释放所拥有的数据块。

3.根据权利要求1或2所述的用于管理文件系统中的文件系统对象的设备，其中，

所述文件系统对象(2802')与检查点内克隆编号(CCN)相关联，并且一个或多个第二指针中的每个指针与相应的检查点编号(CN)相关联，并且

所述将每个数据块的所有权从所述快照对象(2809)转移到所述文件系统对象(2802')包括：

将与引用每个检查点的一个或多个第二指针中的指针相关联的检查点编号(CN)设定为大于或等于与所述文件系统对象(2802')相关联的检查点内克隆编号(CCN)的并且小于所述文件系统的当前检查点编号的值；以及

使得所述快照对象(2809)的一个或多个第一指针不指向任何东西。

4.根据权利要求3所述的用于管理文件系统中的文件系统对象的设备，其中，

所述快照对象(2809)与另一检查点内克隆编号(CCN)相关联，并且一个或多个第一指针中的每个指针与每个检查点编号(CN)相关联。

5.根据权利要求4所述的用于管理文件系统中的文件系统对象的设备，其中，

如果对象包括指针，则所述对象拥有数据块，所述指针引用每个数据块，并且每个数据块与一检查点编号相关联，所述检查点编号大于或等于与所述对象相关联的检查点内克隆编号。

6.根据权利要求3所述的用于管理文件系统中的文件系统对象的设备，其中，

与所述文件系统对象(2802')相关联的检查点内克隆编号指示所述文件系统对象(2802')的数据偏离所述快照对象(2809)的数据的文件系统的最早检查点。

7.根据权利要求1所述的用于管理文件系统中的文件系统对象的设备，其中，

所述文件系统对象(2802')与所述快照对象(2809)链接，并且所述快照对象(2809)与另一快照对象(2803)链接；并且

当从所述文件系统移除所述快照对象(2809)时，所述文件系统对象(2802')被链接到

另一快照对象(2803)。

8.根据权利要求1所述的用于管理文件系统中的文件系统对象的设备,其中,

当确定所述快照对象(2809)的引用计数变为一时,执行从所述快照对象(2809)到所述文件系统对象(2802')的所有权转移以及移除所述快照对象(2809)。

9.根据权利要求8所述的用于管理文件系统中的文件系统对象的设备,其中,

当删除与所述快照对象(2809)链接的另一文件系统对象(2811)时,所述快照对象(2809)的引用计数变为一,并且另一文件系统对象(2811)先前被创建为所述文件系统对象(2802')的克隆。

10.根据权利要求9所述的用于管理文件系统中的文件系统对象的设备,其中,

另一文件系统对象(2811)与另一检查点内克隆编号相关联,所述另一检查点内克隆编号指示另一文件系统对象(2811)的数据偏离所述快照对象(2809)的数据和所述文件系统对象(2802')的数据的文件系统的最早检查点。

11.根据权利要求1所述的用于管理文件系统中的文件系统对象的设备,其中,

所述文件系统对象(2802')是文件系统的可写入文件。

12.根据权利要求1所述的用于管理文件系统中的文件系统对象的设备,其中,

所述快照对象(2809)是文件系统的隐藏文件系统对象。

13.根据权利要求1所述的用于管理文件系统中的文件系统对象的设备,其中,

所述快照对象(2809)是文件系统的只读文件系统对象。

14.一种用于管理文件系统中的文件系统对象的方法,

所述文件系统包括快照对象(2809)和文件系统对象(2802'),所述快照对象(2809)包括引用一个或多个数据块的一个或多个第一指针,并且所述文件系统对象(2802')包括引用一个或多个数据块的一个或多个第二指针,其中,通过所述快照对象(2809)的一个或多个第一指针所引用的一个或多个数据块包括由所述快照对象(2809)拥有的一个或多个数据块;并且

所述方法包括:

-针对由所述快照对象(2809)拥有的并且通过所述快照对象(2809)的一个或多个第一指针以及所述文件系统对象(2802')的一个或多个第二指针所引用的每个数据块,将每个数据块的所有权从所述快照对象(2809)转移到所述文件系统对象(2802'),以及

-在这样的从所述快照对象(2809)到所述文件系统对象(2802')的所有权转移之后,从所述文件系统中移除所述快照对象(2809)。

用于管理文件系统中的文件系统对象的设备和方法

[0001] 本申请是2011年11月21日提出的、申请号为201180056305.9、名称为“数据存储系统中的文件克隆和去克隆”的发明申请的分案申请。

[0002] 相关申请的交叉引用

[0003] 本专利申请要求于2010年11月22日以Daniel J.N.Picken和Neil Berrington的名义提交的(代理机构卷号No.2337/126)题为FILING CLONING IN A DATA STORAGE SYSTEM(数据存储系统中编档克隆)的美国临时专利申请No.61/415,928的优先权权益,通过引用将其全部内容合并在本文中。

[0004] 本专利申请与下列专利申请相关,通过引用将他们中的每个的全部内容都合并在本文中:

[0005] 于2008年6月30日以Christopher J.Aston,Simon L.Benham和Neil Berrington的名义提交的题为MULTI-WAY CHECKPOINTS IN A DATA STORAGE SYSTEM(数据存储系统中的多向检查点)的美国专利申请NO.12/164,730(代理机构卷号No.2337/110),其是于2008年1月16日以Christopher J.Aston名义提出的,题为VALIDATING OBJECTS IN A DATA STORAGE SYSTEM(数据存储系统中证实对象)的美国专利申请No.12/015,192的部分继续申请(代理机构卷号No.2337/113),并因此要求后者的优先权。

[0006] 本专利申请还可能与一个或多个下列专利申请相关,通过引用将他们中的每个的全部内容合并在本文中:

[0007] 于本申请同一日期提出的,题为DYNAMIC WRITE BALANCING IN A DATA STORAGE SYSTEM(数据存储系统中的动态写入平衡)的美国专利申请(代理机构卷号NO.2337/111);

[0008] 于2008年10月9日以John C.Holtom名义提交的,题为SYSTEM,DEVICE,AND METHOD FOR VALIDATING DATA STRUCTURES IN A STORAGE SYSTEM(存储系统中的用于验证数据结构的系统、装置和方法)的美国专利申请NO.12/248,300(代理机构卷号No.2337/117),其要求于2007年10月12日提出的,题为SYSTEM,DEVICE,AND METHOD FOR VALIDATING DATA STRUCTURES IN A STORAGE SYSTEM(在存储系统中用于验证数据结构的系统、装置和方法)的美国临时专利申请No.60/979,561(代理机构卷号No.2337/118)的权益。

[0009] 于2001年6月12日提出的,题为APPARATUS AND METHOD FOR HARDWARE IMPLEMENTATION OR ACCELERATION OF OPERATING SYSTEM FUNCTIONS(用于硬件实现或操作系统功能加速的设备和方法)的美国专利申请No.09/879,798,即现在的美国专利No.6,826,615(代理机构卷号No.2337/103);

[0010] 于2004年7月12日提出的,题为APPARATUS AND METHOD FOR HARDWARE IMPLEMENTATION OR ACCELERATION OF OPERATING SYSTEM FUNCTIONS(用于硬件实现或操作系统功能加速的设备和方法)的美国专利申请No.10/889,158(代理机构卷号No.2337/108);

[0011] 于2002年11月1日以Geoffrey S.Barrall等人名义提出的,题为APPARATUS AND METHOD FOR HARDWARE-BASED FILE SYSTEM(用于基于硬件的文件系统的设备和方法)的美国专利申请No.10/286,015(代理机构卷号No.2337/104);和

[0012] 于2007年8月20日以Geoffrey S.Barrall等人名义提出的,题为APPARATUS AND METHOD FOR HARDWARE-BASED FILE SYSTEM(用于基于硬件的文件系统的设备和方法)的美国专利申请No.11/841,353(代理机构卷号No.2337/117)。

技术领域

[0013] 本发明涉及数据存储系统,尤其涉及数据存储系统中的克隆和去克隆文件。

背景技术

[0014] 于2002年11月1日以Geoffrey S.Barrall等人名义提交的,题为Apparatus and Method for Hardware-Based File System(用于基于硬件的文件系统的设备和方法)的美国专利申请No.10/286,015(代理机构卷号No.2337/104)和于2007年8月20日以Geoffrey S.Barrall等人名义提交的,题为Apparatus and Method for Hardware-Based File System(用于基于硬件的文件系统的设备和方法)的美国专利申请No.11/841,353(代理机构卷号No.2337/117),通过引用将两者的全部内容合并在本文中,它们描述了各种文件系统结构,除此之外,所述文件系统结构还允许文件服务器维护文件系统的两个副本,即文件系统的当前版本和文件系统之前的“检验点”版本。特别是使用包括特定根节点的树结构来维护文件系统,所述特定根节点实际上是被称为左手侧(LHS)和右手侧(RHS)的一对结构。在实践中,一侧被用来保持文件系统的“检验点”副本,同时另一侧被用于文件系统的持续管理(包括文件系统对象的创建、删除和修改)。有时,两侧的角色倒转,使得用于文件系统的持续管理的一侧结构执行“检验点”,并且文件系统的持续管理继续使用持有之前“检验点”的一侧结构。保持两个所谓的动态超级块(dynamic superblock)来保持文件系统的当前版本和检验点版本的踪迹。

[0015] “检验点”的一个目的在于,一旦文件系统的持续管理过程中发生错误,则存储文件系统的副本。在某些情况下,文件系统可以被恢复到“检验点”版本。在这种系统中的一个风险在于,文件系统的当前版本和“检验点”版本都可能会损坏。另一个风险在于,一条重要的信息将被移除或被改变,并且文件系统的当前版本和“检验点”版本都将不包括那条原始信息。

发明内容

[0016] 在本发明的某些实施例中,例如当用户制作文件副本时,采用文件克隆机制来允许在文件系统内快速创建文件的副本(克隆)。在示例性实施例中,源对象的克隆至少最初表述为包含对源对象的各种元素(例如,间接onode节点、直接onode节点和数据块)的引用的结构。可以创建只读和可变克隆。源文件和克隆最初共享这种元素并且当更改源文件或可变克隆时继续共享未修改的元素。在创建克隆时,不需要复制用户数据块或描述与源文件相关联的数据流(即间接/直接onode节点)的元数据块。这种文件克隆的一些特点包括:

[0017] -由于不需要复制构成数据流的用户数据块,因此无论源对象的数据流的大小如何,文件系统对象的数据流都可以被迅速地并且是在相对固定的时间量中有效克隆。而且,不需要复制用于描述数据流的元数据块(即,间接/直接onode节点)。非常小的和固定数量的元数据块被突变。

[0018] -对克隆的/克隆对象操作I/O的复杂性相当于对普通对象的操作。

[0019] -可以被克隆的文件或克隆的次数仅受限于文件系统中空闲空间的数量。

[0020] -文件系统可支持的克隆数目仅受限于文件系统中空闲空间的数量。

[0021] -该文件克隆具有固有的文件去复制,其特征在于,克隆实际上是被创建为与源文件共享数据和元数据块的去复制化文件,而不是创建源文件的完整副本并且后续执行去复制。

[0022] -尽管通过冗余存储(即,RAID控制器)和其他机制,数据损坏得到减轻,但是共享块的损坏会影响多个文件。

[0023] 根据本发明的一个方面,提供一种在文件存储系统中克隆源文件系统对象的方法。源文件系统对象包括至少一个数据块和直接或间接引用至少一个数据块的指针组。该方法包括在文件存储系统中创建只读的数据流快照对象并且在数据流快照对象中存储指针组的副本;并且在文件存储系统中创建可变克隆对象并在克隆对象中存储指针组的副本,其中,数据流快照对象和克隆对象与源文件系统对象共享至少一个数据块,而不为数据流快照对象和克隆对象制作至少一个数据块的单独副本,并且其中,源文件系统对象和克隆对象有效地成为数据流快照对象的可变版本并有效地存储数据流快照对象所代表的对象的只读副本中的改变。

[0024] 根据本发明的另一方面,提供用于克隆文件系统对象的装置,该文件系统对象包括文件存储系统中的源文件系统对象。源文件系统对象包括至少一个数据块和直接或间接引用至少一个数据块的指针组。该装置包括至少一个存储设备;并且存储处理器与至少一个存储设备通信,该存储处理器被配置为在文件存储系统中创建只读的数据流快照对象并在数据流快照对象中存储指针组的副本;以及在文件存储系统中创建可变克隆对象并在克隆对象中存储指针组的副本,其中,数据流快照对象和克隆对象与源文件系统对象共享至少一个数据块,而不为数据流快照对象和克隆对象制作至少一个数据块的单独副本,并且其中,源文件系统对象和克隆对象有效地成为数据流快照对象的可变版本并有效地存储数据流快照对象所代表的对象的只读副本的改变。

[0025] 在多种替选实施例中,每个对象可以包括根onode节点,并且指针组可以被存储在对象根onode节点中。指针组可以从源文件系统对象被复制到数据流快照对象,并且然后从数据流快照对象被复制到克隆对象。

[0026] 实施例还可以包含在数据流快照对象中存储对源文件系统对象的引用和克隆对象的引用;在源文件系统对象中存储对数据流快照对象的引用;和在克隆对象中存储对数据流快照对象的引用。

[0027] 实施例还可以包含保持数据流快照对象中的引用计数,该引用计数用于指示引用数据流快照对象的文件系统中的对象的数量。

[0028] 实施例还可以包含将源文件系统对象的尺寸作为克隆对象的属性,其中从所述源文件系统对象创建所述克隆对象。

[0029] 实施例还可以包含,当可变源文件系统对象或可变克隆对象被修改时,分配至少一个数据块用于存储所述修改并且将至少一个分配的数据块与修改后的对象进行关联,修改后的对象包括修改后的指针组。修改后的对象可以被克隆,例如,通过在文件存储系统中创建第二只读数据流快照对象并且在第二数据流快照对象中存储修改后的指针组的副本;

以及在文件存储系统中创建第二可变克隆对象并在第二克隆对象中存储修改后的指针组的副本。该克隆还可以包含在第二数据流快照对象中存储对修改后的对象的引用、对第二克隆对象的引用、和对第一数据流快照对象的引用;在修改后的文件系统对象中存储对第二数据流快照对象的引用;并且在第二克隆对象中存储对第二数据流快照对象的引用。

[0030] 实施例还可以包含使用数据流快照对象进一步创建源对象的克隆。

[0031] 实施例还可以包含去克隆对象。

附图说明

[0032] 通过参照下面的具体实施方式,参考附图,本发明的上述特征将会更容易理解,其中:

[0033] 图1是依照本发明示例性实施例的文件存储系统的示意框图;

[0034] 图2是显示了依照本发明示例性实施例的文件系统的普通格式的示意框图;

[0035] 图3是显示依照本发明示例性实施例的对象树结构的普通格式的示意框图;

[0036] 图4是显示依照本发明示例性实施例,使用没有其他onode节点的根onode节点的示意框图;

[0037] 图5是显示依照本发明示例性实施例,具有直接onode节点的根onode节点的使用的方框图;

[0038] 图6是显示依照本发明示例性实施例,具有间接onode节点和直接onode节点的根onode节点的使用的方框图;

[0039] 图7是示出依照本发明示例性实施例,使用位于根onode节点和直接onode节点之间的多层间接onode节点的方框图;

[0040] 图8显示了对于本发明示例性实施例,对象编号分配的表述;

[0041] 图9是显示依照本发明示例性实施例的间接对象的普通格式的示意框图;

[0042] 图10是依照本发明示例性实施例,演示DSB、间接对象、根直接对象和文件对象之间的普通关系的示意框图;

[0043] 图11是显示依照本发明示例性实施例,在1号检查点的包括四个数据块和各种onode节点的示例性对象的结构的示意图;

[0044] 图12是显示依照本发明实施例,为经修改的对象创建新的根节点之后,图11的示例性对象的结构的示意图;

[0045] 图13是显示依照本发明实施例,在创建数据块的经修改的副本之后,图12的示例性对象的结构的示意图;

[0046] 图14是显示依照本发明实施例,在创建新的直接onode节点以指向数据块的经修改的副本之后,图13的示例性对象的结构的示意图;

[0047] 图15是显示依照本发明实施例,在创建新的间接onode节点以指向新的直接onode节点之后,图14的示例性对象的结构的示意图;

[0048] 图16是显示依照本发明实施例,在更新新的根节点以指向新的间接onode节点之后,图15的示例性对象的结构的示意图;

[0049] 图17是显示依照使用DSB的循环列表来记录检查点的本发明的示例性实施,在执行检查点之前各种文件系统结构的示意图;

[0050] 图18是显示依照使用DSB的循环列表来记录检查点的本发明示例性实施例，在执行检查点之后，图17的各种文件系统结构的示意图；

[0051] 图19是显示依照使用DSB循环列表来记录检查点的本发明示例性实施例，在修改间接对象之后，图18的各种文件系统结构的示意图；

[0052] 图20是显示依照本发明示例性实施例，其中再次使用一个DSB以创建继承检查点，在执行检查点之前的各种文件系统结构的示意图；

[0053] 图21是显示依照本发明示例性实施例，其中再次使用一个DSB以创建继承检查点，在执行检查点之后，图20的各种文件系统结构的示意图；

[0054] 图22是显示依照本发明示例性实施例，其中再次使用一个DSB以创建继承检查点，在修改间接对象之后，图21的各种文件系统结构的示意图；

[0055] 图23示意性地显示了依照本发明示例性实施例，源对象(文件A)2802、隐藏的数据流快照对象2803、和可变副本2805之间的关系；

[0056] 图24示意性地显示了依照本发明示例性实施例，在4号概念性检查点上的，克隆图11中表述的文件系统对象的之后的对象2802、2803和2805；

[0057] 图25示意性地显示了依照本发明示例性实施例，源对象(文件A)2802、隐藏的数据流快照对象2803、以及两个可变副本2805和2807之间的关系；

[0058] 图26示意性地显示了依照本发明示例性实施例，在6号概念性检查点上，克隆第二可变副本之后的对象2802、2803、2805和2807；

[0059] 图27示意性地示出了经修改的源对象2802’(由单引号代表源对象的修改版本)、具有两个原始源对象克隆2805和2807的第一数据流快照对象2803、第二数据流快照对象2809、以及第二数据流快照对象2809的可变副本2811之间的关系；

[0060] 图28示意性地显示了依照本发明示例性实施例，在8号概念性检查点上，克隆经修改的源对象2802’之后的对象2802’、2803、2809和2811；

[0061] 图29示意性地显示了依照本发明示例性实施例，与特定源对象相关联的DSS对象是如何保留在文件系统中，直到源对象和所有副本都被删除；和

[0062] 图30(包含子部分30A-30C)示例性显示了依照示例性实施例的对象去克隆。

[0063] 应该注意的是，前述的附图和其中描述的元素不一定被画成一致的比例或任何比例。除非上下文给出相反的暗示，否则类似的元素用类似的附图标记指代。

具体实施方式

[0064] 用于本说明和所附权利要求中，除非上下文做出相反的要求，下列术语应当具有所指示出的含义：

[0065] “存储设备”是用于存储数据的设备或系统。存储设备可以包括一个或多个磁性或磁光或光盘驱动器、固态存储设备、或者磁带。为了方便，存储设备有时被称为“磁盘”或“硬盘”。数据存储系统可以包括具有相同或不同存储容量的相同或不同类型的存储设备。

[0066] “RAID控制器”是将几个存储设备的存储容量合并到一块虚拟的存储空间中的设备或系统，所述虚拟的存储空间可以被替选地称为是“系统驱动器”(“SD”)、“逻辑单元”(“LU”或“LUN”)或者是“卷”。通常，SD比单一存储设备大，从几个存储设备提取空间，并且包括冗余信息，以便它能承受磁盘的一定数量的故障而不丢失数据。在示例性实施例中，每个

SD与下文中被称为“逻辑单元标识符”或“LUID”的唯一标识符相关联，并且每个SD将不会大于预定的最大尺寸，诸如2TB-64TB或更多。当命令被发送至SD时，RAID控制器典型地在同一时间将命令转发至SD的所有存储设备。RAID控制器有助于克服典型的存储设备的三个主要局限，即存储设备典型地是存储系统中最慢的组件，他们典型地最可能经历灾难性故障，并且他们典型地具有相对较小的存储容量。

[0067] “RAID系统”是包括一个或多个RAID控制器和多个存储设备的设备或系统。通常，RAID系统将包含两个RAID控制器(以便如果一个发生故障，另一个可以继续工作，并且当两个都正常时还可以分担负荷)和数十个存储设备。在示例性实施例中，RAID系统典型地被配置有两个到三十二个之间的SD。当文件服务器需要存储或检索数据时，它将指令发送到RAID系统的RAID控制器，其继而负责将命令向前路由到单个存储设备并且按需要存储或检索数据。使用一些RAID系统，可以建立SD之间的镜像关系，以使得为了冗余的目的，被写入一个SD(被称为“主SD”)的数据被RAID系统自动地写入另一个SD(本文中被称为“次SD”或“镜像SD”)。次SD可以由和主SD一样的相同的RAID系统或由不同的本地或远程RAID系统管理。为了从在某些情况下SD或者可能是甚至多个SD的丢失或损坏中提供恢复，镜像SD有效地提供了SD之间的RAID 1+0功能。

[0068] “文件系统”是在文件存储系统中存储的文件和目录(文件夹)的结构。在文件存储系统中，典型地使用多个虚拟存储结构来管理文件系统，并且在示例性实施例中，使用被称为范围(range)、条带集(stripeset)和跨度(span)的虚拟存储结构的层次管理文件系统。“范围”由主SD自身或者由主/次SD对组成，所述主/次SD对应当包含同样的数据，并且因此提供与单一SD相同的存储容量。“条带集”由一个或多个范围组成。“跨度”由一个或多个条带集组成。因此，最终，跨度由一个或多个SD(典型地是四至五十个SD)组成。跨度可以被分为一个或多个文件系统，每个文件系统具有单独的名称和标识符以及潜在不同的特征(诸如，一个文件系统可以被格式化为具有多个32KB集群并且另一个具有多个4KB集群，一个文件系统可以是“一写多读”存储器(Worm)而另一个不是，等等)。跨度上的每个文件系统被单独格式化、安装和卸载。文件系统可以以任何顺序和在任何时间被创建和删除。文件系统可以被配置为自动扩展(或者替选地，防止或限制自动扩展)或可以手动扩展。

[0069] 值的“集合”可以包含一个或多个值。

[0070] 在下面使用的标题是为了方便，并且不被理解为以任何方式限制本发明。

[0071] 在本发明的某些实施例中，例如当用户制作文件的副本时，采用文件克隆机制来允许在文件系统内快速创建文件的副本(克隆)。在示例性实施例中，源对象的克隆至少最初被表述为包含涉及源对象的各种元素的结构(例如间接onode节点、直接onode节点和数据块)。只读和可变克隆都可以被创建。源文件和克隆最初共享这种元素并且在对源文件或可变克隆进行改变时继续共享未修改的元素。在创建克隆时，不需要复制描述与源文件相关的数据流(即间接/直接onode节点)的元数据块或用户数据块。在适当的时间，克隆的文件可以被“去克隆”。

[0072] 虽然结合示例性文件系统描述本发明示例性实施例，但是应该注意各种克隆和去克隆的概念还可以被应用到其他类型的文件系统。

[0073] 示例性文件系统

[0074] 图1是依照本发明示例性实施例的文件存储系统的示意性方框图。除此之外，文件

存储系统包括许多文件服务器(为了简单和方便,显示了单一文件服务器9002),所述服务器在例如互联网协议网络(如互联网)的通信网络9004上与多种客户端装置9006_1-9006_M通信以及在例如光纤通道网络的存储网络9010上与各种RAID系统9008_1-9008_N通信。客户端装置9006_1-9006_M和文件服务器9002利用例如CIFS和/或NFS的一个或多个网络文件协议进行通信。文件服务器9002和RAID系统9008_1-9008_N利用例如SCSI的存储协议进行通信。应该注意的是,文件存储系统可以包括以多种配置互联的多个文件服务器和多个RAID系统,包括全网状配置,其中任何文件服务器可以与任何RAID系统在冗余的和切换的光纤通道网络上进行通信。

[0075] 文件服务器9002包括用来管理一个或多个文件系统的存储处理器。文件服务器9002可以被配置为允许客户端访问文件系统的部分,如在指定名称下的树或子树。在CIFS用语中,这样的访问可以被称作“共享”而在NFS用语中,这样的访问可以被称作“输出”。在内部,文件服务器9002可以包括多种硬件实现和/或硬件加速子系统,例如,如同以上以引用的方式合并于本文中的美国专利申请No.09/879,798和No.10/889,158所描述的一样,并且可以包括包含多个链接的子模块的基于硬件的文件系统,例如,如同以上以引用的方式合并于本文中的美国专利申请No.10/286,015和No.11/841,353所描述的一样。

[0076] 每个RAID系统9008典型地包括至少一个RAID控制器(并且通常是两个RAID控制器用于冗余),还包括由RAID控制器管理的许多物理存储设备(如磁盘)。RAID系统9008将其存储资源聚合成许多SD。例如,每个RAID系统9008可以被配置为具有2到32个之间的SD。每个SD可以被限制为预定的最大尺寸(如,2TB-64TB或更多)。将几个存储设备组合到SD可以提供许多好处,包括提高的速度(单个存储设备相对较慢,但数据可以是跨越几个存储设备的以扩大瓶颈)、增加的容量(单个的存储设备比较小,但是可以组合几个存储设备来提供更多的可用空间)、抽象(所使用的空间量可以大于或小于单一存储设备的尺寸)和恢复能力(奇偶校验或冗余信息可以被存储在每个存储设备上,以便SD能承受存储设备的丢失)。

[0077] 文件服务器9002被配置为使用一个或多个SD,所述SD可以来自单一的RAID系统或来自多个RAID系统。文件服务器9002通常可以询问RAID系统来找出每个SD是主的还是次的。文件服务器9002使用SD进行控制的方法在本文中被称为“许可(licensing)”。因此,实际上,文件服务器9002典型地对于一些SD是被许可的,而对于其他的则是不被许可的。

[0078] 在内部,文件服务器9002能够将几个SD组合成更大的在本文中被称为“跨度”的存储池。跨度本质上是几个SD的RAID 0阵列。将几个SD组合成跨度可以提供许多类似于通过将多个物理磁盘组合成SD所获得的好处,包括提高的速度(在多个RAID系统的多个SD之间铺开I/O可以进一步扩大存储瓶颈)、增加的存储容量(跨度可以比单一的SD大,该SD可以被限制为两百万兆字节)和附加的抽象,它允许更灵活的存储空间分配。

[0079] 文件系统树结构

[0080] 文件服务器9002将多种类型的对象存储在文件系统里。对象一般可以被分为系统对象和文件对象。文件对象被创建用于存储用户数据和相关属性,比如文字处理器或电子表格文件。系统对象由文件存储系统创建用来管理信息,并且包括诸如根目录对象、空闲空间分配对象、修改的检查点对象列表对象、修改的保留对象列表对象和软件元数据对象之类的东西,这里仅举一些例子。更具体来说,目录对象被创建用于存储目录信息。空闲空间分配对象被创建用于存储空闲空间分配信息。修改的检查点对象列表对象和修改的保留对

象列表对象(两者在下面都将更详细的描述)分别被创建用来存储涉及检查点和保留的检查点的信息。软件元数据对象(在下面将更详细的描述)是特殊的对象,被用来保持与文件或目录对象有关的过量文件属性(即,如下面所描述的,不能符合文件或目录对象内的预先指定区域内的文件属性,如CIFS安全属性),并且,该软件元数据对象是由文件或目录对象的创建者所创建,它包含对文件或目录对象内软件元数据对象的引用。

[0081] 使用具有根节点的树结构(被称为动态超级数据块或DSB)管理文件系统的实例,所述树结构优选地被存储在存储系统内的固定位置。除此之外,将DSB存储在固定位置使得文件服务器9002很容易定位DSB。文件服务器9002可以维护多个DSB以存储代表不同检查点的文件系统的不同版本(如,当前的“工作”版本和一个或多个“检查点”版本)。在示例性实施例中,DSB包括指向间接对象的指针(在下面详细描述),间接对象继而包括指向其他对象的指针。

[0082] 图2是显示依照本发明示例性实施例的文件系统实例的一般格式的示意性方框图。DSB 202是表示文件系统树结构的根的特殊结构。除此之外,DSB 202包括指向间接对象204的指针,其继而包括指向包括系统对象206和文件对象208的文件系统中的其他对象的指针。

[0083] 在本发明的实施例中,N个动态超级数据块(N>2)被保持用于文件系统,只有其中一个被认为是在任何给定的时间点上最新的。DSB的数量可以是固定的或是可配置的。DSB位于固定位置上并且被用来记录磁盘上的检查点的状态。每个DSB指向一个间接对象。

[0084] 除此之外,以下的信息被存储在每个动态超级数据块中:

[0085] ●与此动态超级数据块相关的检查点编号。

[0086] ●用于该检查点的修改后检查点对象列表对象的句柄。

[0087] ●来自最后的保留检查点的修改的保留对象列表对象的对象编号。

[0088] ●此检查点的状态(即,是否已经创建检查点)。

[0089] ●允许检查DSB和其他结构(如间接对象)有效性的CRC和多种其他信息。

[0090] 在示例性实施例中,DSB被当作循环列表(即,第一动态超级数据块被认为连续地跟随最后的动态超级数据块),并且在循环列表中,每个连续的检查点使用下一个连续的动态超级数据块。当文件服务器9002打开卷时,它通常读入所有动态超级数据块并且在DSB上执行多种检查。具有检查点状态被标记为完成的最新检查点编号和通过了多种其他完整性检查(sanity check)的DSB被认为代表了这个卷上最新的有效检查点。对于下一个检查点,文件服务器9002开始使用循环列表中的下一个DSB。

[0091] 在下面讨论间接对象204的一般格式。

[0092] 对象树结构

[0093] 一般来说,包括间接对象204的文件系统里的每个对象、每个系统对象206和每个文件对象208,都是使用包括单独的对象根节点和可选地包括许多间接节点、直接节点和存储块的单独的树结构来实现的。DSB 202包括指向间接对象204的根节点的指针。间接对象204包括指向其他对象的根节点的指针。

[0094] 图3是显示依照本发明示例性实施例的对象树结构的一般格式的示意性方框图。根(“R”)节点302可以指向多个的间接(“I”)节点304,其中的每一个间接节点304可以指向许多直接(“D”)节点306,其中的每一个直接节点306可以指向许多存储块(“B”)308。实际

上，对象树结构可以例如根据对象的尺寸广泛地变化。并且，当信息被添加到对象或从对象中被删除时，特定对象的树结构可以随着时间变化。例如，当更多存储空间被用于对象时，节点可以被动态地添加到树结构，并且可以按照需要使用不同程度的间接(例如，间接节点可以指向直接节点或其他间接节点)。

[0095] 当创建对象时，为对象创建对象根节点。最初，这种“空”对象的根节点没有指向任何间接节点、直接节点、或者数据块的指针。

[0096] 当数据被添加到对象时，它首先被放入从根节点直接指向的数据块中。这在图4的图表中被示出，显示了没有其他节点的根节点的使用。要注意的是，在该图和所有以下的图表中，为了简单起见，根节点和直接节点被显示为只有两个数据指针，并且间接节点被显示为只有两个间接或直接节点指针。

[0097] 一旦根节点内的所有直接块指针被填满，那么直接节点A被创建为具有从根节点指向直接节点的指针。图5显示了具有这种直接节点A的根节点的使用。要注意的是，根节点具有多个数据块指针，但是只具有指向直接或间接节点的单一指针。

[0098] 如果对象中的数据增长到填满直接节点内的所有数据指针，那么创建间接节点B，如图6所示。图6显示了具有间接节点和直接节点的根节点的使用。指向直接节点A的根节点中的指针被更改为指向间接节点B，并且在间接节点B中的第一指针被设置为指向直接节点A。同时，创建新的直接节点C，它也被从间接节点B所指向。随着更多的数据被创建，更多的直接节点被创建，所有这些直接节点都被从间接节点所指向。

[0099] 一旦间接节点B中的所有直接节点指针都已经被使用了，另一个间接节点D被创建，其被插入在根节点和第一间接节点B之间。另一个间接节点E和直接节点F也被创建，以允许更多的数据块被引用。这些情况都在图7中被显示，说明了被放置在根节点和直接节点之间的多个层的间接节点的使用。

[0100] 添加间接节点以创建更高程度的间接的过程被重复，以容纳对象所包含的无论多少的数据。

[0101] 对象根节点包括检查点编号以标识最后修改的对象所在的检查点(检查点编号最初标识被创建的对象所在的检查点，其后，在新检查点中，每次修改对象时，检查点编号都发生改变)。在示例性实施例中，被创建的对象所在的检查点编号也被存储在对象根节点中。同样，在对象根节点内的是标识对象类型的参数，对象根节点为该对象提供元数据。对象类型可以是例如任何空闲空间、文件或目录。除了对象类型以外，对象根节点还具有针对块中对象长度的参数。

[0102] 对象根节点还具有一系列的指针。其中之一是指向对象根节点的任何上一个版本的指针。如果出现保留的检查点已经被合适的检查点所替换，那么存在那里有可能已经存储了对象根节点的上一个版本的疑问，并且指针标识对象根节点的这种前一个版本的扇区编号。

[0103] 对于对象根节点对应的实际数据，对象根节点包括指向与相应对象相关的每个数据块的单独的指针。多达18块数据块的位置被存储在对象根节点内。对于数据超出18块，额外地需要直接节点，在这种情况下，对象根节点还具有指向直接节点的指针，所述直接节点在对象根节点中被磁盘上的扇区编号所识别。

[0104] 直接节点包括检查点编号并且被布置为存储与对象相一致的一定数量的块(例如

约60或61块)的位置。

[0105] 当第一直接节点被充分利用于标识数据块时,那么一个或多个间接节点被用来标识第一直接节点以及具有与该对象相对应的数据块的额外的直接节点。在这种情况下,对象根节点具有指向间接节点的指针,并且间接节点具有指向相对应的直接节点的指针。当间接节点被充分利用时,随后当必要时,额外的介入间接节点被利用。这种结构允许部分文件的快速识别,而无关于文件的分片。

[0106] 本发明的多种实施例可以包括促进大型文件创建的机制,他们通常是在最初被创建时,用零填充的稀疏文件。

[0107] 一种这样的机制允许分配由零填充的数据块而不将零实际地写入数据块。具体来说,对象根节点和直接节点包括用于每个块指针来指示是否相应的块被逻辑上用零填充(块并不需要实际用零填充)的标志。因此,例如,当数据块被分配而不是用零填充该数据块时,与数据块有关的比特可以被设置为指示数据块被零填充,并且对数据块的读取访问将返回零,而并不从数据块实际上读取数据。

[0108] 类似的机制允许在不为文件实际分配所有数据块和节点的情况下创建文件。具体来说,指向块和其他节点的指针可以包括用来指示是否块或其他节点已经被实际创建的比特。在相关的块和节点还没有被创建的情况下,那么块和节点被创建为需要适应写入需求,并且分配比特被相应地切换。注意,创建块需要空间的分配,将数据写入块,并为合适的节点设置比特标志。在一个特定的示例性实施例中,这种机制仅仅被用来创建文件而不分配所有的数据块;其他文件节点如上所述被分配。

[0109] 事务日志,也在元数据高速缓存内被保持为运行。

[0110] 在示例性实施例中,也可以以进一步减少与节点结构有关的磁盘写入的方式建立节点结构。最后,节点结构不仅需要容纳文件内容的存储还容纳文件属性的存储。文件属性包括多种参数,包括文件尺寸、文件创建时间和日期,文件修改时间和日期,只读状态和访问权限,以及其他。这种联系利用这样的事实:由于对象根节点尚未被写入磁盘(即,如上所述,因为对象根节点的磁盘写入被延迟),所以在给定的检查点期间可以频繁执行对象根节点的内容的改变。因此,在示例性实施例中,一部分对象根节点被保留用于文件属性的存储。

[0111] 更为普遍地,在示例性实施例中定义下面用于文件属性存储的结构:

[0112] 以太网节点(enode)(更新的开销少,容量有限)。在示例性实施例中,这个结构被限定在对象根节点中并且是128字节。

[0113] 软件元数据对象(更新的开销高,接近无限容量)。这是一种用于元数据存储的专门对象,因此在磁盘上有它自己的存储位置;在以太网节点中识别对象。

[0114] 因此,在示例性实施例中,每个对象根节点存储以下类型的信息:

[0115] ●检查点的编号。

[0116] ●用于该对象版本的数据长度。

[0117] ●在运行列表中,用于该对象的间接程度的编号。

[0118] ●对象的类型。在请求进入访问对象时,这主要被用作完整性检查。

[0119] ●为保留的检查点(如果存在的话)制作的指向较旧的根节点版本的指针。

[0120] ●指向较新的根节点版本的指针(仅仅在这是为保留的检查点所做的根节点的副

本时有效)。

[0121] ●多达19个数据块描述符。每个数据块描述符包括指向数据块的指针、检查点编号、和表示块是否被零填充的比特。

[0122] ●指向直接节点或者指向间接节点的单一指针。

[0123] ●用于该对象的以太网节点数据的128字节。

[0124] ●允许根节点被检查有效性的CRC和多种完整性双字。

[0125] 如下面所讨论的,对象可以包括每次执行保留的检查点时所创建的根节点的副本。指向较旧根节点版本的指针和指向较新根节点版本的指针允许根节点的双链表被创建为包括当前根节点以及为保留的检查点所创建的根节点的任何副本。双链表辅助保留的检查点的创建和删除。

[0126] 如上面所讨论的,间接节点提供了根节点和直接节点之间的间接程度。在示例性实施例中,下列信息被存储在间接节点中:

[0127] ●检查点的编号。

[0128] ●指向间接节点或者指向直接节点的指针(如,多达122个这样的指针)。

[0129] ●允许间接节点被检查有效性的CRC和多种完整性双字。

[0130] 如上面所讨论的,直接节点提供了指向磁盘上的数据块的直接指针。在示例性实施例中,以下信息被存储在直接节点中:

[0131] ●检查点的编号。

[0132] ●许多数据块描述符(如,多达62个这样的描述符)。每个数据块描述符包括指向数据块的指针、检查点编号、和表示块是否被零填充的比特。

[0133] ●允许间接节点被检查有效性的CRC和多种完整性双字。

[0134] 从对象和数据块和直接以及间接节点删除的数据是不再需要的,它们被返回到空闲空间分配控制器。

[0135] 根据一个实施例,间接程度的编号随着对象的变小而减少,直到对象中的所有数据通过根节点中的直接块指针可以被引用时,其余的直接和间接节点全部被释放并且间接程度将被设置为零。

[0136] 如果向特定文件对象的写入操作具有超出对象当前末尾的开始偏移或对象的长度被设置为大于当前长度,那么对象的未定义部分(例如,在对象当前的末尾和新写入数据的开始之间)通常用零填充。在典型的实施中,这涉及到为对象的所有由零填充的部分分配磁盘块并且用零填充那些磁盘块。对于1GB的文件,这可能要花10秒钟的量级。对于1TB的文件,可能要花3小时的量级。

[0137] 在本发明的示例性实施例中,并不实际上用零填充与对象的未定义部分相关的数据块,而是那些数据块的内容是被保留未被写入,并且每个数据块指针内的比特被用来说明块是被认为由零填充的。如果文件服务器9002(并且,特别是对象存储子模块)看到这个比特设置,那么,即使在磁盘上,块可以包含完全不同的东西,文件服务器也知道块应该被零填充。如果块被读取,那么文件服务器9002为这个块返回零,而不返回其实际存在于磁盘上的内容。如果块以不填满整个块的方式被写入,那么文件服务器9002将零写入到没有被写入的块的部分,并且,随后为这个块重置“由零填充”比特。

[0138] 另一个与将对象的长度设置为一些非常大的值相关的考虑是分配数据块和创建

所需的直接和间接节点结构所花费的时间。例如，在示例性实施例中，使用尺寸为4K的磁盘块，1TB对象需要大约4百万个直接节点以及较少数量的间接节点。这可以花费40秒的量级来向磁盘写入。所有所需数据块的空闲空间分配、以及后续更新到空闲空间位图也将大大增加这个时间。如果在文件创建开始之后立即执行检查点，那么整个系统通常会在整个这个时间中停止服务请求(至任何卷)。

[0139] 在本发明的替选实施例中，这个问题可以通过不实际为文件的由零填充的部分分配磁盘块被解决，如上所述的。这意味着当对象存储看到对零填充块的写入时，它首先必须将磁盘空间分配给那个块，并且把指向它的指针放入相关的节点结构。

[0140] 在另一个替选实施例中，除了不实际将磁盘块分配给文件的由零填充的部分以外，这个问题也可以通过不创建相应的节点结构而被解决。要实施这个方面，每个节点指针可以包括用来表示它指向的节点是否被分配的比特。如果节点没被分配，那么当出现要求节点是有效的操作时，只有那时，磁盘空间才会被分配给它，并且插入正确的指针。通过这种方式，巨大的由零填充的对象可以只有能够很快被创建的根节点。

[0141] 对象编号和间接对象

[0142] 在文件存储系统内，每个对象都和用于引用该对象的对象编号相关联。系统对象通常有固定的、预定义的对象编号，因为他们通常总是存在于系统内的。文件对象通常是从有效的对象编号池中被动态分配的对象编号。这些文件对象编号在某些情况下可以被重复使用(例如，当文件被删除时，它的对象编号可以被释放以便由后来的文件对象重复使用)。

[0143] 图8显示了本发明示例性实施例中的对象编号分配的表现。具体来说，文件系统可以包括Z对象编号(其中，Z是可变的，并且可以随对象数量的增加而随着时间增长)。一定范围的对象编号被保留给系统对象206(在这个例子中，对象编号1-J)，并且其余对象编号(在这个例子中，对象编号K-Z)被分配给文件对象208。通常，系统对象206的编号是固定的，而文件对象208的编号可以变化。

[0144] 在示例性实施例中，间接对象204逻辑上被组织为表格，用对象编号索引的每个对象有一个表项目。如图9所示，表格中的每个项目502包括对象类型字段和指针字段。对象类型字段定义许多不同的值，但为了讨论，一组值被定义为“已使用”对象并且另一组值被定义为“空闲”对象。因此，特定表项目的对象类型字段中的值将表明相应的对象编号是否已使用还是空闲。

[0145] 每个已使用的表项目的指针字段包括指向对象的根节点的指针。当对象被创建时，对象根节点被创建(如上面所讨论的)，并且对象编号被分配给对象。指向对象根节点的指针被存储在间接对象204中，特别是被存储在与已分配的对象编号相关的表项目内。因此，基于对象的对象编号，特别是通过索引间接对象表格结构并且访问指针字段，文件服务器9002可以容易地定位任何对象的对象根节点。最初，这种“空”对象的根节点没有指向任何间接节点、直接节点、或者数据块的指针，尽管数据块、间接节点和直接节点可以随着时间被添加到对象树结构。

[0146] 空闲表项目的指针字段被用来维持一个或多个空闲对象编号列表(并且最好两个单一链接的、非循环的空闲对象编号列表，例如，在于2007年10月12日提交的、题为“System,Device,and Method for Validating Data Structures in a Storage System(存储系统中验证数据结构的系统、装置和方法)”的美国临时专利申请No.60/979,561中所

描述的,并且通过引用将其全部合并于此。

[0147] 具体来说,与每个空闲对象编号相关联的表项目包括对在其空闲对象编号列表中的下一个空闲对象编号的引用,而不是指向对象的根节点的指针。因为在示例性实施例中空闲对象编号列表是单一链接的、非循环列表,因此在空闲对象编号列表内与最后空闲对象编号相关联的表项目包括“空”引用(例如,该值为零)。

[0148] 从理论上说,在间接对象中维持单一空闲对象列表编号是可能的。随着新对象被创建,列表中的空闲对象编号可以被重复利用和从列表中移除,并且随着将对象从系统删除,空闲对象编号可以被加入到列表中。

[0149] 然而,在本发明示例性实施例中,两个单独的空闲对象编号列表被维持在间接对象中,一个列出可立即被用于重复利用的空闲对象编号,另一个列出不能立即被用于重复利用的最新释放为空闲的对象编号。在这个示例性实施例中,文件服务器有时执行文件系统的“检查点”或“快照”,(例如,以下描述的,或于2002年11月1日以Geoffrey S.Barrall等人名义提交的题为“Apparatus and Method for Hardware-Based File System(用于基于硬件的文件系统的设备和方法)”的美国专利申请No.10/286,015和于2007年8月20日以Geoffrey S.Barrall等人名义提交的题为“Apparatus and Method for Hardware-Based File System(用于基于硬件的文件系统的设备和方法)”的美国专利申请No.11/841,353所讨论的,通过引用的方式将这两个申请全部合并于此),这样,在任何给定的时间,文件服务器都具有文件系统的“工作副本”,例如当对象被创建、删除和修改时,该文件系统的“工作副本”都可以改变。由于种种原因,对于这个示例性实施例来说很重要的是确保随着时间的推移,对被分配了特定重复利用的对象编号的不同对象给予不同的对象句柄。因此,在这个示例性实施例中,两个空闲对象编号列表被用于确保特定的对象编号不能在文件系统的相同的工作副本内被释放和被重复利用(即,通过将已被释放的对象编号添加到一个列表,但是从其他列表分配对象编号),并且当对象被创建时,当前检查点编号的后面32比特被包括在对象句柄中,以便在不同检查点中创建的对象会有不同的句柄。因此,在文件系统的任何特定的工作副本期间,文件服务器重复利用来自一个列表的空闲对象编号,同时将新释放的对象编号添加到其他列表。在每个“检查点”,两个列表的角色都被“交换”,以便在以前的检查点期间被释放的对象编号可以被重复利用,同时在当前的检查点期间被释放的新的对象编号不可以在该检查点期间被重复利用。在新对象被创建时,如果被重复利用的空闲对象编号所在的列表是空的,那么为了提供额外的空闲对象编号,间接对象被扩展(即便在另一列表中可能实际上存在一些空闲对象编号)。然而实际上,由于在每个检查点上的角色交换,两个列表通常都会随着时间的推移积累许多空闲对象编号,所以在稳定状态下间接对象应该不需要经常被扩展。

[0150] 在示例性实施例中,DSB 202包括指向间接对象204(并且,更具体地说,指向间接对象204的根节点)的指针,并且还包括两个指针,每个指针用于间接对象204中的一个空闲对象编号列表。每个指针指向表项目的各自空闲对象编号列表的开始处。因为DSB 202被存储在存储系统内的固定位置上,并且包括指向间接对象204和间接对象204内的空闲对象编号列表的指针,因此文件服务器9002可以容易地定位间接对象204(并因此定位任何其他对象的根节点)以及使用DSB 202的空闲对象编号列表。

[0151] 因此,再次参照图4中所示的间接对象204的表格结构,特定表项目的对象类型字

段的值将表示相应的对象编号是否被使用还是空闲的。如果该对象编号被使用了，那么该表项目的指针字段将包括指向相应对象的根节点的指针。然而，如果该对象编号是空闲的，那么该表项目的指针字段将包括对在其空闲对象编号列表中的下一个空闲对象编号的引用。

[0152] 一般来说，已释放的对象编号被添加到空闲对象编号列表的头部部分，并且被重复利用的对象编号也来自空闲对象编号列表的头部部分。

[0153] 如在题为“System,Device,and Method for Validating Data Structures in a Storage System(存储系统中验证数据结构的系统、装置和方法)”的美国临时专利申请No.60/979,561中所述的，有时，为了确保所有空闲对象编号被包括在空闲对象编号列表中并且没有“已使用”的对象编号被包括在空闲对象编号列表中，验证间接对象204(包括空闲对象编号列表)可能是必要或期望的。验证间接对象204(包括空闲对象编号列表)的一种方法是从开始到结束遍历每个空闲对象编号列表，以确保不存在循环并确保列表末尾具有空引用。然而，在工作数据存储系统中，大量的对象随着时间被创建和删除并不罕见，从而空闲对象编号列表可以变得相当长。此外，空闲对象编号列表没有被排序，而是随着对象编号发生被使用和被释放而更新，所以遍历空闲对象编号列表通常需要根据对单一链接列表的引用，逐一跳跃间接对象204。空闲对象编号列表的这种遍历通常是缓慢和低效的。

[0154] 因此，在本发明的示例性实施例中，间接对象表格结构被从顶到底地循序遍历，并且使用位图或其他适当的数据结构跟踪“已使用”和“空闲”对象编号。具体来说，如果特定的对象编号被使用，那么位图中相应的比特被标记；如果该比特已经被标记了，那么间接对象是损坏的(如，因为“已使用”的对象编号被较早的“空闲”项目错误地引用)。如果特定对象编号是空闲的，那么在间接对象表格结构中的相应项目包括对空闲对象编号列表中下一个空闲对象编号的引用，所以对应于这种下一个空闲对象编号的比特被标记在位图中；如果该比特已经被标记，那么间接对象是损坏的(如，因为空闲对象编号列表包括了“已使用”对象编号或因为空闲对象编号列表包括循环引用)。在整个间接对象表格结构已经被遍历后，由DSB 202中的指针所指向的两个开始表格条目被检查，如果任一表格项目是“已使用”，那么间接对象是损坏的。此外，在整个间接对象表格结构被遍历后，应该被保留为不做标记的唯一比特是与两个空闲对象编号列表的两个开始表项目相关联的比特，该两个比特由DSB 202中的指针所指。如果这些比特中的任一个被标记，那么间接对象是损坏的。如果位图中任何其他比特是未被标记的，那么相应的对象编号既没有被使用也没有被包括在空闲对象编号列表中，在这种情况下，间接对象是可用的(因为在正常过程中，这种“未链接”的空闲项目不会被重复利用)。可以执行额外的处理以确保每个空闲对象编号列表以空引用终止。

[0155] 在多种替选实施例中，位图可以被初始化为全部是零，并且位图中的比特可以通过设置比特(即，设置成一)被“标记”；就这一点而言，所谓的“测试并设置”操作既可被用于测试比特的值又可被用于在单一操作中设置比特。替选地，位图可以被初始化为全部为一，并且位图中的比特可以通过清空比特(即，为零)被“标记”。当然，其他类型的数据结构和其他类型的标记方案可以被用于其他实施例。本发明并不局限于位图的使用或数据结构或标记方案的任何特定类型。

[0156] 在示例性实施例中，间接对象可以被实现为没有实际存储块的“伪文件”。在示例

性实施例中，不具有指向对象树结构中的实际数据块的指针(例如，如图2所示)，而是在间接对象树结构中的这种指针指向对应对象的根节点。因此，在示例性实施例中，间接对象将每个对象编号映射为与对应的文件系统对象相关的根节点的扇区地址。间接对象树结构可以基于对象编号随后被遍历，以获取指向对应对象的根节点的指针。

[0157] 在示例性实施例中，间接对象“伪文件”被结构化，以便一个共用的代码可以被用来基于对象编号遍历间接对象树结构，以获取指向对应对象的根节点的指针并且基于文件偏移遍历其他对象树结构，以获取指向对应数据块的指针。在这种实施例中，对象编号基本上被转换成虚拟文件偏移，然后间接对象树结构以使用实际文件偏移遍历其他对象树结构相同的方式被遍历。具有可用于遍历间接对象“伪文件”树结构和其他对象树结构的共用代码的一个优势在于，单一逻辑块可以被用于两个函数，这对于硬件中的树遍历函数是特别有利的。示例性系统对象

[0158] 如上所述，文件系统包括多种类型的系统对象。一般来说，尽管某些系统对象可以具有可变的对象编号，但是系统对象有固定的、预定义的对象编号。以下是本发明示例性实施例中的一些系统对象的描述。

[0159] 根目录对象是系统对象(即，它具有根节点和固定的预定义对象编号)，其将文件名称映射为对应的对象编号。因此，当文件被创建时，文件存储系统为文件分配根节点，为文件指定对象编号，将项目添加到将文件名映射为对象编号的根目录对象，并且为文件向将项目添加到将对象编号映射为根节点的磁盘地址的间接对象。间接对象中的项目将根目录对象编号映射为根目录对象的根节点的磁盘地址。

[0160] 图10是原理方框图，展示了根据本发明示例性实施例，DSB 202、间接对象204、根目录对象606和文件对象208之间的一般关系。如上所述，间接对象中的项目将根目录对象编号映射为根目录对象的根节点的磁盘地址，根目录对象将文件名映射为对象编号，并且间接对象将对象编号映射为对象。因此，当文件服务器9002需要基于对象的文件名称定位对象时，文件服务器9002可以通过间接对象定位根目录对象606(即，使用与根目录对象606相关的对象编号)，使用根目录对象606将文件名映射为与其对应的对象编号，然后使用对象编号通过间接对象定位对象。

[0161] 空闲空间位图对象是系统对象(即，它具有根节点和固定的预定义的对象编号)，其显示文件存储系统中的空闲存储块。间接对象中的项目将空闲空间位图对象编号映射为空闲空间位图对象的根节点的磁盘地址。

[0162] 修改后的检查点对象列表对象是系统对象(即，它具有根节点和固定的预定义的对象编号)，其识别在检查点循环期间已经被创建或修改的对象。在每个检查点的开始，修改后的检查点对象列表对象被创建。当每次作为该检查点的一部分创建或修改不同的对象时，其对象编号被写入到修改后的检查点对象列表对象，以便在检查点被创建时，存在列出在那个检查点中所有被创建或修改的对象的对象。

[0163] 空闲块对象是系统对象，其被用于在特定的检查点期间，由文件系统已经变为未被使用的数据块的跟踪。空闲块对象列出了可被释放的扇区地址。特定数据块被文件系统已经变为未被使用的事实并不一定意味着数据块可以被释放以重复使用，因为数据块可以与较早的检查点和/或保留的检查点相关联。因此，其他机制(如，后台清理任务)通常被用于决定如何以及何时可以释放块。

[0164] 在一个预期的实施例中，文件存储系统会为N个检查点保持N个空闲块对象(其中N通常大于二)，而间接对象包括用于使用固定的预定空闲块对象编号的N个空闲块对象的单独的项目(即，在间接对象中的N个项目)。在这种实施例中，当特定检查点(如，第N+1个检查点)被删除时，文件存储系统会处理与那个检查点相关联的空闲块对象，以便包含在其中的信息不会丢失。

[0165] 在一个替换的实施例中，文件存储系统可以保持超过N个空闲块对象(即使只有N个检查点被保持)，以便空闲块对象可以使用后台清理过程而不是运行时过程被处理。在这种实施例中，由于系统中空闲块对象的数量可以变化，所以在间接对象中具有固定数量的项目并不实用，因此空闲块对象的目录(如，空闲块目录对象)可以被替代使用。在这里，间接对象中的单一项目可以被用于空闲块目录对象，并且空闲块目录对象可以保持指向单个空闲块对象的指针。

[0166] 保留的检查点配置对象是系统对象，其被用于维持保留的检查点的列表。在间接对象中的项目将保留的检查点配置对象编号映射为保留的检查点配置对象的根节点的磁盘地址。保留的检查点配置对象在下面进一步详细讨论。

[0167] 多路检查点

[0168] 在特定实施例中，多个检查点可以被执行以便文件系统的多个版本可以随着时间而被保持。例如，多个单独的根结构(在下文中被称为“动态超级块”或“DSB”)被用于管理文件系统的多个实例。为了易于访问，DSB优选地被存储在存储系统中的固定位置中以便于访问，虽然DSB可以替选地以其他方式被存储。通常存在超过两个DSB，DSB的数量可以是固定的或变化的。对于DSB的数量没有理论上的限制(虽然对于多种实施可以有实际上的限制)。通过这种方式，如果必须或希望将文件系统恢复回之前的“检查点”，那么存在多个可供选择的“检查点”，这提供存在将文件系统可以恢复成的文件系统的完整版本或存在包含文件系统特定版本的检查点的更好机会。

[0169] 文件系统请求的处理由一系列检查点描述，以不少于一些用户指定时间间隔的频繁程度，比如每10秒预定出现检查点。检查点可以在其他时间被执行，例如，如果超过一半的正用于当前检查点的非易失性RAM已经满了、如果扇区高速缓存正在变满、如果用户请求保留检查点(下面讨论)、或在其他适当的情况下。

[0170] 对每个连续的检查点，存在在磁盘上存储的当前文件结构信息，其取代来自前一个检查点的以前存储的文件结构信息。检查点被循序编号并被用于文件请求的临时分组处理。

[0171] 如上所述，本发明的示例性实施例保持N个DSB(其中N大于二，例如，16)。DSB被用来执行连续的检查点。

[0172] 因此，在任何给定的时间，存在文件系统的当前(工作)版本和文件系统的一个或多个检查点版本。由于存储系统通常是非常动态的，所以文件系统的当前版本几乎肯定会在执行检查点之后几乎立即开始改变。例如，文件系统对象可以随时间被添加、删除或修改。然而，为了保持检查点，没有一种与存储的检查点相关联的结构可以被允许改变，至少直至特定的检查点被删除或重写。因此，随着文件系统的当前版本中的对象被添加、删除和修改，对象树结构的新版本根据需要被创建，并且多种指针被相应的更新。

[0173] 例如，图11示意性显示了在1号检查点被创建的示例性对象的对象结构。该对象包

括四个数据块，即数据块0(2310)，数据块1(2312)，数据块2(2314)和数据块3(2316)。直接节点2306包括指向数据块0(2310)的指针和指向数据块1(2312)的指针。直接节点2308包括指向数据块2(2314)的指针和指向数据块3(2316)的指针。间接节点2304包括指向直接节点2306的指针和指向直接节点2308的指针。根节点2302包括指向间接节点2304的指针。所有节点和所有数据块都被标记有1号检查点。

[0174] 假设现在，数据块0(2310)要在3号检查点被修改。因为根节点2402是较早检查点的部分，它不能被修改。相反，文件服务器9002的对象存储子模块保存旧的根节点2302的副本存储到磁盘上的空闲空间，并将这个新的根节点标记为3号检查点(即，新的根节点被创建的检查点)。图12示意性地显示了创建新的根节点2403后的对象结构。在这个点上，根节点2402和新的根节点2403都指向间接节点2304。

[0175] 对象存储子模块随后遍历在根节点开始的对象结构，直到它到达数据块0(2310)的描述符。由于数据块0(2310)是较早检查点的部分，因此它不能被修改。相反，对象存储子模块在磁盘上的空闲空间中创建数据块2310的修改后的副本，并将这个新的数据块标记为3号检查点(即，新的数据块被创建的检查点)。图13示意性地显示了新的数据块2510的创建之后的对象结构。

[0176] 对象存储子模块现在需要将指向新的数据块2510的指针放入直接节点，但是对象存储子模块不能将指向新的数据块2510的指针放入直接节点2306，因为直接节点2306是较早检查点的组件。对象存储子模块因此将直接节点2306的修改后的副本创建到包括指向新的数据块0(2510)和旧的数据块1(2312)的指针的磁盘上的空闲空间中，并将这个新的直接节点标记为3号检查点(即，新的数据块被创建的检查点)。图14示意性地显示了新的直接节点2606被创建后的对象结构，新的直接节点2606包括指向新的数据块0(2510)和旧的数据块1(2312)的指针。

[0177] 对象存储子模块现在需要将指向新的直接节点2606的指针放入间接节点，但是对象存储子模块不能将指向新的直接节点2606的指针放入间接节点2304，因为间接节点2304是较早检查点的组件。对象存储子模块因此创建具有指向新的直接节点2606和旧的直接节点2308的指针的间接节点2304的修改后的副本。图15示意性地显示了新的间接节点被创建后的对象结构，该新的间接节点包括指向新的直接节点2606和旧的直接节点2308的指针。

[0178] 最后，对象存储子模块将指向新的间接节点2704的指针写入新的根节点2403。图16示意性地显示了指向新的间接节点2704的指针被写入新的根节点2403后的对象结构。

[0179] 应该注意的是，在完成数据块0的修改后，块2402、2304、2306和2310是对象的检查点1版本的组件但不是当前检查点3版本的组件；块2308、2312、2314和2316既是对象的检查点1版本的组件也是当前检查点3版本的组件；块2403、2704、2606和2510是对象的当前检查点3版本的组件但不是检查点1版本的组件。

[0180] 还应当注意的是，新节点并不必需要按照上述的顺序被创建。例如，新的根节点可以最后被创建而不是最先被创建。

[0181] 因此，当文件系统对象被修改时，通过对象树结构传播修改，因此对已修改的对象创建新的根节点。只需要在给定的检查点中为对象创建一次新的根节点；新的根节点可以在单一的检查点期间被修订多次。

[0182] 为了在文件系统的当前版本中包括对象的新版本，当前的间接对象被修改为指向

已修改对象的根节点，而不指向对象的之前版本的根节点。例如，再次参照图16，对于与该对象相关联的对象编号，当前间接对象可以被更新为指向根节点2403而非指向根节点2402。

[0183] 相似的，如果在文件系统当前版本中创建新的对象或删除现有对象时，当前的间接对象被相应的更新。例如，如果新的对象被创建，则间接对象被修改为包括指向新对象的根节点的指针。如果现有的对象被删除，则间接对象被修改为将相应的对象编号标记为空闲的。

[0184] 由于间接对象也是具有根节点的树结构，所以间接对象的修改也通过树结构传播，以使得可以为修改后的间接对象创建新的根节点。同样，只需要在给定的检查点中为间接对象创建一次新的根节点；新的根节点在单一的检查点期间可以被修订多次。

[0185] 因此，在特定的检查点期间创建间接对象的新版本时，对于修改后的间接对象，与检查点相关联的DSB被更新为指向用于修改后的间接对象的新的根节点。因此，文件系统的每个版本(即，当前版本和每个检查点版本)通常将包括间接对象的单独版本，每个版本具有不同的间接对象根节点(但可能共享一个或多个间接节点、直接节点、和/或数据块)。

[0186] 在一个示例性实施例中，DSB被当作循环列表处理，并且在预定的时间间隔继续执行检查点，这样在稳定状态期间，每个新的检查点“重写”旧的检查点，造成由“重写的”检查点所表述的文件系统的旧版本的丢失。

[0187] 图17是显示了根据本发明示例性实施例的，在执行检查点之前的多种文件系统结构的示意图。具体来说，显示了编号202和203的两个DSB。DSB 202与文件系统的当前版本相关联，并包括指向间接对象204的当前版本的根节点的指针。DSB 203是下一个可用的DSB。

[0188] 为了从文件系统的当前版本创建检查点，在循环列表中的下一个DSB(即，本示例中的DSB 203)为新的检查点被初始化。除此之外，这种初始化包括将下一个检查点编号写入到DSB 203中并且将指向间接对象204的根节点的指针存储到DSB 203中。图18显示了依据本发明示例性实施例的，在执行检查点之后的多种文件系统结构的示意图。在该点上，DSB 202代表文件系统的最近的检查点版本，而DSB 203代表文件系统的当前(工作)版本。

[0189] 如上所述，当对象被创建、修改和删除时，文件系统的当前版本可以改变。此外，如上所述，当文件系统的当前版本改变时，间接对象的新版本(具有新的根节点)被创建。因此，如在图18中所描述的，当间接对象的当前版本在执行检查点之后变化时，从而新的间接对象根节点被创建，当前文件系统版本的DSB(即，图18中的DSB 203)被更新为指向新的间接对象根节点而不是指向之前的间接对象根节点。图19是显示了根据本发明示例性实施例的，在间接对象的修改之后的多种文件系统结构的示意图。在这里，与文件系统的检查点版本相关联的DSB 202指向间接对象204的检查点版本，同时与文件系统的当前版本相关联的DSB 203指向新的间接对象205的根节点。

[0190] 如上所述，通常定期执行检查点，以便文件系统的多个版本随着时间被保持。在每个检查点，文件系统的当前(工作)版本移到循环列表中的下一个连续的DSB。当从系统删除特定检查点版本时(例如，因为它的DSB已经被重复使用)，与已删除的检查点相关联的存储器在适当的时候可以被恢复，例如，使用标识并释放不再被使用的存储器的后台任务。

[0191] 在一个替选实施例中，特定DSB可以作为连续检查点的当前的DSB被重复使用，其他DSB被用于保存文件系统的检查点版本。

[0192] 图20是显示依据本发明示例性实施例的，在执行检查点之前的多种文件系统结构的示意图。具体来说，显示了编号202和203的两个DSB。DSB 202与文件系统的当前版本有关，并包括指向间接对象204的当前版本的根节点的指针。DSB 203是下一个可用的DSB。

[0193] 为了从文件系统的当前版本创建检查点，为新的检查点初始化下一个DSB 203。除此之外，这种初始化包括将下一个检查点编号写入DSB 203中并将指向间接对象204的根节点的指针存储到DSB 203中。图18是显示依据本发明示例性实施例的，在执行检查点之后的多种文件系统结构的示意图。在该点中，DSB 203代表文件系统最新的检查点版本，同时DSB 202继续代表文件系统的当前(工作)版本。

[0194] 如上所述，当对象被创建、修改和删除时，文件系统的当前版本可以改变。同样，如上所述，当文件系统的当前版本改变时，间接对象的新版本(具有新的根节点)被创建。因此，当间接对象的当前版本在执行图21所示的检查点之后变化时，这种新的间接对象根节点被创建，当前文件系统版本的DSB(即，图21中的DSB 202)被更新为指向新的间接对象根节点而不指向之前的间接对象根节点。图22是显示依据本发明示例性实施例的，在间接对象的修改之后的多种文件系统结构的示意图。在这里，与文件系统的检查点版本相关联的DSB 203指向间接对象204的检查点版本，同时继续与文件系统的当前版本相关联的DSB 202指向新的间接对象205的根节点。

[0195] 当卷被安装好时，系统通常将要回到最后的有效检查点。然而，可能会有系统或用户选择恢复到较早的有效检查点的时候。在本发明的实施例中，文件服务器9002能够保持超过一个的检查点，所以可以存在文件系统可被恢复到的文件系统的多个版本。可以提供实用程序，以允许操作者检验多种检查点版本的内容，以便辅助用于返回文件系统的检查点版本的选择。

[0196] 尽管N个检查点可以被保持在系统中(其中，N通常大于二，并且可以是用户可配置的)，可以提供用户触发机制以保留检查点，以便它将维持有效并可访问(只读)，直到用户选择删除它。保留的检查点基本上是在特定检查点上的文件系统结构的只读版本。可以执行多个保留的检查点，并且包括用于删除所选定的保留的检查点或将文件系统返回到所选定的保留的检查点的机制(例如，将文件系统返回到紧随灾难的已知状态)。只要保留的检查点仍然有效，包含保留的检查点的节点和数据块就不能被修改或返回到空闲空间。应该注意的是，节点或数据块可以是多个保留的检查点的组件，并且只要节点或数据块是至少一个保留的检查点的组件，则特定的节点或数据块就不能被返回到空闲空间。

[0197] 在示例性实施例中，除了别的之外，执行保留的检查点涉及在磁盘上的空闲空间中存储相应的DSB的副本并且在保留的检查点配置对象中存储所存储的DSB副本的引用。只要保留的检查点被存储在文件系统中，与保留的检查点相关联的结构就不能被删除。即使从执行保留的检查点的检查点已经被重写了，也是这样。在示例性实施例中，文件服务器9002包括用于防止与保留的检查点相关联的结构被删除的机制。

[0198] 文件克隆

[0199] 在本发明的某些实施例中，比如当用户制作文件的副本时，采用文件克隆机制，以允许在文件系统内快速创建文件的副本(克隆)。在示例性实施例中，源对象的克隆至少最初是由包含源对源对象的多种元素(例如间接onode节点、直接onode节点和数据块)的引用的结构来表述的。只读和可变的克隆都可以被创建。源文件和克隆起初共享这种元素，并且

在更改源文件或更改可变克隆时继续共享未被修改的元素。在克隆被创建时，没有一个用户数据块或描述与源文件相关联的数据流(即，间接/直接onode节点)的元数据块需要被复制。这种文件克隆的一些特点包括：

[0200] -文件系统对象的数据流可以被有效地迅速克隆，并且在相对固定量的时间中而无关于源对象数据流的尺寸，因为没有一个包含数据流的用户数据块需要被复制。此外，没有一个描述数据流(即，间接/直接onode节点)的元数据块需要被复制。非常小的和固定数量的元数据块被改变。

[0201] -对克隆的/克隆对象的I/O句柄的复杂性相当于对普通对象。

[0202] -可以被克隆的文件或克隆的次数仅受限于文件系统中空闲空间的数量。

[0203] -文件系统可以支持的克隆数目仅受限于文件系统中空闲空间的数量。

[0204] -该文件克隆具有固有的文件去复制的特征，因为克隆实际上是被创建为与源文件共享数据和元数据块的去复制化文件，而不是创建源文件的完全副本并且稍后执行去复制。

[0205] -尽管通过冗余存储(即RAID控制器)和其他机制，数据损坏得到减轻，但是共享块的损坏会影响多个文件。

[0206] 在示例性实施例中，首先通过创建代表源对象的只读克隆(快照)的新对象，在下文中被称为“数据流快照”对象或“DSS”，然后创建对象的可变克隆，从而文件系统对象被克隆。在克隆对象的根onode节点中的块指针和onode节点块指针最初被设置为指向与源对象相同的块。来自源对象的某些元数据(如，文件时间、安全等)和命名的数据流不被复制到克隆对象。元数据被保持在源对象和克隆对象中，以将数据流快照对象与源对象和可变的克隆对象联系起来，也将源对象和可变的克隆对象与数据流快照对象联系起来。在示例性实施例中，数据流快照对象是“隐藏”的对象，因为它是对于文件系统用户不可见的。源对象和可变的克隆对象都有效地成为DSS对象的可写入版本并有效存储他们与DSS对象的差异。

[0207] 在创建数据流快照对象之前，系统优选地确保源对象是静态的。在示例性实施例中，这包括以下步骤：

[0208] 步骤A1.将源对象锁定以不发生突变。

[0209] 步骤A2.执行文件系统检查点，这有效地串行化在给定文件系统上的克隆的创建(虽然创建的速度将受限于文件系统可以将之前的检查点提交到磁盘的速度，从而致使存储侧延误将导致更长的创建时间)。

[0210] 然后，在检查点完成后，系统创建数据流快照对象，这包括以下步骤：

[0211] 步骤A3.创建数据流快照对象。

[0212] 步骤A4.将块指针从源对象的根onode节点复制到数据流快照对象的根onode节点。

[0213] 步骤A5.将当前检查点编号记录在源对象的根onode节点中。这是对象的检查点内克隆的编号(“CCN”)；它定义了最早的检查点，其中对象的数据流可以与其相关联的数据流快照对象的数据流不同。

[0214] 系统还保持以下元数据，以使源对象与数据流快照对象相联系：

[0215] 步骤A6.数据流快照对象的句柄被记录在源对象的元数据中。

[0216] 步骤A7.引用数据流快照对象的对象引用计数和列表被记录在数据流快照对象的

元数据中。

[0217] 如果源对象已经是克隆(即,正在被克隆的克隆),那么在步骤4和5之间存在两个额外的步骤:

[0218] 步骤A4a.把新的数据流快照对象与源文件的当前数据流快照对象相关联。

[0219] 步骤A4b.将源文件的当前检查点内克隆的编号记录在新的数据流快照对象的根onode节点中。

[0220] 具有数据流快照对象的数据流的可变克隆的另一个对象可以如下方式被创建:

[0221] 步骤B1.创建新的文件系统对象。

[0222] 步骤B2.将块指针从数据流快照对象的根onode节点复制到新对象的根onode节点。

[0223] 步骤B3.将当前检查点编号记录在新对象的根onode节点中。

[0224] 步骤B4.将数据流快照对象的句柄记录在新对象的元数据中。

[0225] 步骤B5.递增数据流快照对象的引用计数并将新对象的句柄添加到数据流快照对象的引用列表。

[0226] 应该注意的是,检查点内克隆的编号(CCN)不同于对象的检查点编号(在图11中被标记为“CN”),后者记录了对象的最后修改的检查点。两者都被存储在对象根onode节点中。

[0227] 当修改用户数据或元数据块时,当决定是否必须将块写入新空间时,文件系统考虑块是否已经偏离了克隆对象相关联的数据流快照对象:

[0228] ●通过具有小于克隆的检查点内克隆编号的检查点编号(未偏离的块)的指针对用户/元数据块进行的改变必须被写入新空间。

[0229] ●通过具有大于或等于克隆的检查点内克隆编号的检查点编号(已偏离的块)的指针对用户/元数据块进行的改变基本上遵循如上所述的针对“有效的”文件系统中的对象的通常规则。

[0230] 以上所描述的一些文件克隆概念可以由下列示例说明,所述示例基于图11中所呈现的文件系统对象。

[0231] 图23示意性地显示了依据本发明示例性实施例的,源对象(文件A)2802、隐藏的数据流快照对象2803和可变克隆2805之间的关系。

[0232] 图24示意性地显示了依据本发明示例性实施例的,图11中所呈现的在概念层次的4号检查点的文件系统对象的克隆之后,对象2802、2803和2805。

[0233] 如上所述,在源对象被锁定并且检查点被执行(步骤A1和A2)之后,数据流快照对象2803被创建(步骤A3),并且来自源对象的根onode节点2302的块指针被复制到数据流快照对象2803根onode节点(步骤A4)。当前检查点编号被记录在源对象2802根onode节点中(步骤A5)。数据流快照对象2803的句柄被记录在源对象2802元数据中(步骤A6)。引用数据流快照对象2803的对象的引用计数和列表被记录在数据流快照对象2803元数据中(步骤A7)。此时,只有源对象2802引用数据流快照对象2803,并且引用计数(暂时)被设置为一。

[0234] 同样如上所述,可变克隆2805被创建(步骤B1),并且来自数据流快照对象2803根onode节点的块指针被复制到对象2805根onode节点(步骤B2)。当前检查点编号被记录在对象2805根onode节点中(步骤B3)。数据流快照对象2803的句柄被记录在对象2805元数据中(步骤B4)。数据流快照对象2803中的引用计数被递增,并且对象2805的句柄被记录在引用

的数据流快照对象2803列表中(步骤B5)。

[0235] 应该注意的是,2802和2803之间的虚线的双面箭头代表了这两个结构之间的链接,并且相似地,2803和2805之间的虚线的双面箭头代表了这两个结构之间的链接。

[0236] 应该注意的是,当可变克隆2805被创建时,例如块指针的信息可以从源对象2802被复制而不是从DSS对象2803被复制,虽然从DSS对象2803复制是优选的,并且可以允许源对象2803从静态状态更快地被释放(如,在DSS对象2803创建后但是在可变克隆2805创建前)。

[0237] 如果在被修改之前再次克隆源对象2802,则第二可变克隆被创建。图25示意性地显示了依据本发明示例性实施例的,源对象(文件A)2802、隐藏的数据流快照对象2803、以及两个可变克隆2805和2807之间的关系。

[0238] 图26示意性地显示了依据本发明示例性实施例的,在概念层次6号检查点,第二可变克隆创建之后的对象2802、2803、2805和2807。具体来说,第二可变克隆2807被创建,并且来自数据流快照对象2803根onode节点的块指针被复制到对象2807根onode节点。当前检查点编号被记录在对象2807根onode节点中。数据流快照对象2803的句柄被记录在对象2807元数据中。数据流快照对象2803中的引用计数被递增,并且对象2807的句柄被记录在引用的数据流快照对象2803的列表中。

[0239] 由于源对象2802是有效的数据流快照对象2803的可变副本,因此源对象2802可以随着时间的推移被修改,这使源对象2802的数据流偏离数据流快照对象和其他的文件克隆的数据流。例如,再次参照图24和图11-16中所示的对象结构,比如说,来源于2802的源对象的数据块0(2310)在概念性的5号检查点中的修改会导致发散的树结构,使根onode节点2802指向新的间接onode节点(类似于在图12-16所示的概念性的3号检查点中,当图11中所示的对象被修改时,根onode节点2403最终指向间接onode节点2704的方法),使根onode节点2803和2805继续指向间接onode节点2304。相似的,数据流快照对象的可变克隆可以随着时间的推移被修改,这使可变的副本的数据流偏离数据流快照对象和其他克隆的数据流。

[0240] 如果源对象被修改后,修改后的源对象的副本被制作,那么使用包括额外的步骤A4a和A4b的上述过程,为修改后的源对象创建第二数据流快照对象,并且随后第二数据流快照对象的可变克隆被创建。

[0241] 图27示意性显示了修改后的源对象2802'(单引号代表源对象的修改后的版本)、具有两个原始源对象克隆2805和2807的第一数据流快照对象2803、第二数据流快照对象2809、以及第二数据流快照对象2809的可变克隆2811之间的关系。可以看到,数据流快照对象2803和2809在逻辑上被分等级地链接。

[0242] 图28示意性地显示了依据本发明示例性实施例的,在概念层次的8号检查点,已修改的源对象2802'的克隆之后的对象2802'、2803、2809和2811。

[0243] 如上所述,数据流快照对象2809被创建(步骤A3),并且来自源对象2802'根onode节点的块指针被复制到数据流快照对象2809根onode节点(步骤A4)。数据流快照对象2809与源对象的当前数据流快照对象2803相关联(步骤A4a),具体来说,通过将DSS 2803的句柄记录到DSS 2809中,将DSS 2809的句柄记录到DSS 2803的对象列表,并且递增DSS 2803中的引用计数。源文件2802'的当前检查点内克隆的编号被记录在DSS 2809根onode节点中(步骤A4b)。当前检查点编号被记录在源对象2802'根onode节点中(步骤A5)。数据流快照对

象2809的句柄被记录在源对象2802'的元数据中(步骤A6)。引用数据流快照对象2809的对象的引用计数和列表被记录在数据流快照对象2809元数据中(步骤A7)。此时,只有源对象2802'引用数据流快照对象2809,并且引用计数被(暂时)设置为一。

[0244] 同样如上所述,可变文件副本2811被创建(步骤B1),并且来自数据流快照对象2809根onode节点的块指针被复制到对象2811根onode节点(步骤B2)。当前检查点编号被记录在对象2811根onode节点中(步骤B3)。数据流快照对象2809的句柄被记录在对象2811元数据中(步骤B4)。数据流快照对象2809的引用计数被递增,并且对象2811的句柄被记录在数据流快照对象2809的引用列表中(步骤B5)。

[0245] 应该注意的是,源对象2802'可进一步随着时间的推移被修改,并且对象的未来版本的克隆会导致额外的数据流快照对象分等级地与第一DSS 2803和第二DSS 2809链接。相似地,克隆2805和/或克隆2807可随时间的推移被修改,并且这些克隆的副本(包括未修改的和已修改的)可以不时地基本上如上所述相同地被制作,使额外的DSS对象根据需要被分等级地添加。未修改的对象2805的克隆基本上与未修改的源对象2802的克隆一样,这会导致与第一DSS对象2803链接的另一个可变克隆,而对象2805的修改版本的克隆会导致与新的DSS链接的修改后对象的可变副本的创建,其继而被链接到第一DSS对象2803。

[0246] 在示例性实施例中,DSS对象与特定源对象相关联保留在文件系统中,直到源对象和所有克隆被删除。因此,例如,即使图27中所示的克隆2805、2807和2811被删除,DSS 2803、DSS 2809和对象2802'依然保留,如图29中示意性呈现的。在这个示例中,文件系统包括对象的当前版本以及两个之前的快照版本。除非和直到源对象明显地偏离对象的快照版本,对象应该共享许多数据和元数据块,并且因此保持快照对象不应消耗大量的存储空间。如果有必要或期望,快照对象可以例如通过去克隆操作被移除,该操作基本上重新构造对象2802',以如同原始的(即,未被克隆的)对象出现并且移除DSS对象,并从DSS对象释放不与对象2802'共享的数据和元数据块。一些或所有这些功能可以作为"后台"任务被执行。

[0247] 应该注意的是,以上参照步骤A1-A7所述的包括步骤A4a和A4b的逻辑流,被用于演示在本发明示例性实施例中多种对象是如何被创建和链接的。实际上,可选步骤A4a和A4b可以是虚拟的步骤,因为常见的逻辑可以被用于初始克隆和对克隆进行克隆两者。例如,每个根onode节点基本上包括"向上指针"以指向在等级上更高的根onode节点。最初,源对象中的向上指针是空(NULL)的,因为源对象不指向等级上更高的DSS对象。当源对象被克隆时,常见的逻辑可以从源对象根onode节点将向上指针复制到新创建的DSS对象(即,第一DSS对象),然后设置源对象根onode节点中的向上指针指向DSS对象,并且相似地,可以从源对象将当前检查点编号复制到第一DSS对象,然后将当前检查点编号记录到源对象中。在图24中呈现了作为结果得到的向上指针。源对象可以随后被修改。当修改后的源对象被克隆时,常见的逻辑可以从修改后的源对象(其指向第一DSS对象)将向上指针复制到新创建的第二DSS对象,然后设置在修改后的源对象中的向上指针指向第二DSS对象,相似的,可以从修改后的对象将检查点编号复制到第二DSS对象并记录修改后的对象中的当前检查点编号。在图28中呈现了作为结果得到的向上指针。因此,这种公用代码高效地,不需要在克隆未被克隆的文件和克隆已被克隆的文件之间进行区分,并且这种公用代码将创建任何数量的链接的等级程度。这种公用代码的特别优势在于,在基于硬件的文件系统中易于实现。

[0248] 在如上所述的示例性实施例中,使用与多种对象相关的文件句柄,DSS对象被链接

到源对象和一个或多个克隆对象，反之亦然。除此以外，这种链接允许存储系统控制器快速地识别特定的文件是否已经被克隆并且还定位与已克隆文件相关的对象。应该注意的是，本发明并不局限于使用文件句柄来链接根onode节点。相反，比如对象编号的其他信息可以被用于补充或代替文件句柄。

[0249] 应该注意的是，在示例性实施例中，成为已克隆文件属性的是源文件的尺寸，从所述源文件创建克隆。因此，例如，克隆1G字节的文件将导致1G字节被占用为与已克隆的文件相关的配额。配额不考虑在克隆文件之间共享的任何块。

[0250] 依据性能，克隆对象的读取和写入应符合常规非克隆文件的读取和写入。与未克隆的文件一样，基于硬件的文件系统可以自动查询和自动应答相对于克隆的文件的NFS/CIFS操作。客户端侧对已克隆文件的删除(如，使用“rm”命令)可以立即完成，随之，已克隆文件被实际删除并且在后台执行DSS对象。

[0251] 应该注意的是，克隆对象可以被不同的用户和组“拥有”，并且除了源对象和其他克隆以外，可以位于不同的目录树。

[0252] 应该的注意的是，如上所述的文件克隆结构被用于管理文件系统内的文件，且不影响如何在文件系统外访问文件。与非克隆的文件一样，已克隆文件的转移(如，HSR/NDMP)转移整个文件内容，导致了在目的地的“臃肿(fat)”的文件。

[0253] 应该注意的是，在本发明的示例性实施例中，上面所描述的克隆逻辑优选地主要在作为基于硬件的文件系统的一部分的硬件中实施。

[0254] 已克隆对象的去克隆

[0255] 如上所述，在示例性实施例中，与特定源对象相关的DSS对象保留在文件系统中，直到源对象和所有克隆被删除。除非和直到源对象明显偏离对象的快照版本，对象应该共享许多数据和元数据块，因此保持快照对象不应消耗大量的存储空间。如果有必要或希望，快照对象可以通过去克隆操作被移除，去克隆操作基本上将对象重构为以原始的(即，未克隆的)对象出现并且移除DSS对象并从DSS对象释放不与对象共享的数据和元数据块。一些或所有这些功能可以作为“后台”任务被执行。

[0256] 在示例性实施例中，这种“去克隆”被执行如下。当DSS对象的引用计数变为一(如在图29中)并且依然存在的引用是克隆(不是另一个DSS对象)时，克隆可以从DSS对象被“去克隆”并且DSS对象可以被删除。

[0257] 在示例性实施例中，这种“去克隆”是通过从DSS将用户数据块所有权转移到其最后存在的有效文件可变克隆来执行的。就这一点而言，如果块指针的检查点编号大于或等于对象的检查点内克隆编号(其中所有权意味着负责释放)，那么对象(DSS或可变克隆)被认为拥有块。共享用户数据块的所有权可以通过以下方法被转移到有效文件:

[0258] 1.制作有效文件的块指针的检查点编号:

[0259] a.大于或等于有效文件的检查点内克隆编号。

[0260] b.并且小于文件系统的当前CP编号(例如，如果它随后在当前检查点被修改，为了损坏时的一致性，确保块被保存)。

[0261] 2.使DSS的块指针稀疏。

[0262] 在示例性实施例中，为了满足步骤1中的条件，有效文件的检查点内克隆编号被使用。在改变任何块指针之前发出检查点，以确保检查点内克隆编号小于文件系统的当前的

CP编号。

[0263] 在示例性实施例中，这两个步骤的顺序很重要，因为到DSS的更新将潜在地释放一些它拥有的(和在步骤1之前，可能已经与有效文件共享的)onode节点。

[0264] 图30(包含子部分的30A-30C)被用来演示如本发明示例性实施例的文件去克隆的多方面。

[0265] 图30A示意性地显示了描述在每个对象的根onode节点中最初的三个块指针的对象链。

[0266] 具体来说，第一快照对象具有10号检查点内克隆。其块指针分别指向存储在块1-3中的数据。

[0267] 第二快照对象具有20号检查点内克隆。它(仍然)与第一快照对象共享它的第一个块。它的第二和第三块已经偏离，所以它的第二和第三块指针现在分别指向存储在块4和5中的数据。

[0268] 有效文件对象具有30号检查点内克隆。它与第一快照和第二快照对象共享它的第一块并且与第二快照对象共享它的第二块。它的第三块已经偏离，所以它的第三块指针现在指向存储在块6中的数据。

[0269] 如果第二快照对象(即，父DSS对象)的引用计数下降到一，那么它拥有并与有效文件对象共享的用户数据块所有权可以通过自动改变相应的块指针被转移到有效文件对象。具体来说，对于被DSS对象拥有并与有效文件对象共享的每个数据块，在DSS对象中的相应块指针被形成为“稀疏”的(如下面更多讨论的，更新块指针的检查点编号)，并且与有效文件对象中相应的块指针相关的检查点编号被更新为有效文件对象的检查点内克隆编号。图30B示意性地描述了从第二快照对象(即，DSS对象)将块4的所有权转移到有效文件对象。具体来说，指向块4的第二快照指针被形成为“稀疏”的，并且指向块4的有效文件指针的检查点编号被更新为有效文件对象的检查点内克隆编号。

[0270] 遵循着保留修改后的onode节点的通常规则，执行这些转移。具体来说，无论块指针何时被更新，与块指针相关的检查点编号被更新为当前的检查点编号。因为块指针已被修改，包含的onode节点被写入到新的空间。如果该onode节点是直接或间接的onode节点，那么指向父onode节点中的onode节点的块指针也被更新，这样父onode节点被修改并且被写入新的空间，以此类推，直到根onode节点被处理。在所有权转移过程结束时，有效文件对象和DSS对象不再共享任何onode节点。

[0271] 因此，在示例性实施例中，为了从DSS对象将共享块的所有权转移到有效文件对象，有效文件对象被遍历，并且对于发现的由DSS对象共享的并且所拥有的每个区域，在有效文件对象中的相关块指针被“触及”(这更新了受影响的块指针的检查点编号，类似于在相应的块被写入但是没有任何数据被实际写入时更新检查点编号，以便有效文件对象现在拥有这些块)，以及在DSS对象中的相关块指针被形成为稀疏(这使受影响的块指针不指向任何东西，并且也更新了检查点编号，实际上在DSS对象中创建了“洞”，以便当DSS对象最终被删除时，它不再指向被转移到有效文件对象的用户数据块)。一旦以前由DSS对象所拥有的所有共享的用户数据块的块指针已经以这种方式被转换，DSS对象就可以被安全地删除。

[0272] 再次参照图30B，在第二快照对象的删除开始之前，有效文件对象被从第二快照对象的引用列表删除并被添加到第一快照对象的引用列表，借此使第一快照对象成为有效文

件对象的父DSS。如果并且当第一快照对象的引用计数达到一(即,以至于有效文件对象是第一快照的独子),那么第一快照对象可以如上所述被去克隆。例如,如图30C示意性的描述,块1的所有权从第一快照对象被转移到有效的文件对象。在第一快照对象所拥有并且与有效文件对象共享的所有块的所有权被转移之后,第一快照对象可以被删除。

[0273] 在示例性实施例中,通过每次有效文件被锁定时转移有限数量的用户数据块的所有权,“去克隆”过程应对被同时改变的有效文件。同样,如果在这个过程中有效文件被克隆,则这个过程将被中止。这种“去克隆”过程潜在的“弄脏”许多间接/直接onode节点,但没有“弄脏”用户数据块。虽然,一个有益的副产品是它留下了有效文件与“正确”的对象编号和重复使用计数以前共享的onode节点。在删除所有有效文件的前任DSS对象之后,有效文件可以被转换回普通(非克隆)文件。

[0274] 其他方面

[0275] 应该注意的是,“服务器”这个属于在本文中可以被用来描述用于本发明某些实施例中的设备,并且除非上下文做出相反的指示,否则不应被解释为将本发明限制于任何特定的设备类型。因此,设备可以包括但不限于,桥、路由器、桥-路由器(桥路器)、开关、节点、服务器、计算机、电器、或其他类型的设备。这样的设备通常包括用于在通信网络上通信的一个或多个网络接口和被配置为相应地执行设备功能的处理器(例如,具有存储器和其他外围设备和/或应用程序专用硬件的微处理器)。通信网络通常可以包括公共和/或私人网络;可以包括局域网、广域网、城域网、存储器、和/或其他类型的网络;并且可以采用的通信技术包括但不限于,模拟技术、数字技术、光学技术、无线技术(如,蓝牙)、网络技术和网络互连技术。

[0276] 还应该注意的是设备可以使用通信协议和消息(例如,被该设备创建、传输、接收、存储、和/或处理的消息),这些消息可以由通信网络或介质传送。除非上下文做出相反要求,否则本发明不应被解释为限制于任何特定的通信消息类型、通信消息格式,或通信协议。因此,通信消息通常可以包括但不限于,框架、分组、数据电报、用户数据报、存储格或其他类型的通信消息。除非上下文做出相反要求,否则,具体的通信协议的引用是作为示例的,并且应该可以理解的是,可选实施例可以酌情使用这种通信协议的变形(例如,可以不时制作协议修改或扩展)或其他已知的或者将来开发的协议。

[0277] 还应该注意的是,逻辑流可以在本文中被描述来演示本发明的多个方面,不应被解释为将本发明限制为任何特定的逻辑流或逻辑的实施。所描述的逻辑可以被划分成不同的逻辑块(如,程序、模块、功能或子程序),而不改变整体结果或者不偏离本发明的真正范围。很多时候,逻辑元素可以被添加、修改、省略、以不同的顺序执行、或使用不同的逻辑结构来实施(例如,逻辑门、循环基元、条件逻辑、和其他逻辑结构)而不改变整体的结果或者不偏离本发明的真正范围。

[0278] 本发明可以以许多不同的形式体现,包括但不限于,与处理器一起使用的计算机程序逻辑(例如,微处理器、微控制器、数字信号处理器、或通用计算机)、与可编程逻辑器件一起使用的可编程逻辑(例如,现场可编程门阵列(FPGA)或其他PLD)、离散组件、集成电路(例如,专用集成电路(ASIC))、或包括他们的任何组合的任何其他设备。实施一些或全部所描述功能的计算机程序逻辑通常实现为这样一组计算机程序指令:被转换成计算机可执行格式、这样存储在计算机可读介质中、并且在操作系统控制下被微处理器执行。实施一些或

全部所描述功能的基于硬件的逻辑可以使用一个或多个适当配置的FPGA来实现。

[0279] 实施全部或部分本文中之前所描述的功能的计算机程序逻辑可以以多种形式体现实施，包括但绝不限于，源代码形式、计算机可执行形式、以及多种中间形式(例如，由汇编器、编译器、链接器、或定位器所产生的形式)。源代码可以包括以多种编程语言实施的一系列计算机程序指令(例如，对象代码、汇编语言、或例如Fortran、C、C++、JAVA或HTML的高级语言)以便与不同的操作系统或操作环境一起使用。源代码可以定义和使用多种数据结构和通信消息。源代码可以是计算机可执行形式(例如，通过解释程序)、或可以被转换(例如，通过翻译器、汇编程序、或者编译器)为计算机可执行形式的源代码。

[0280] 实施全部或部分本文中之前所述功能的计算机程序逻辑可以在单一处理器上(例如，同时地)在不同的时间被执行，或者可以在多个处理器上在相同的或不同的时间被执行，并且可以在单一操作系统进程/线程下或在不同的操作系统进程/线程下运行。因此，术语“计算机进程”通常指一组计算机程序指令的执行，而不管不同的计算机进程是否在相同或不同的处理器上执行并且不管不同计算机进程是否在相同的操作系统进程/线程或者不同的操作系统进程/线程下运行。

[0281] 计算机程序可以以任何形式永久或者暂时地被固定(例如，源代码形式、计算机可执行形式、或者中间形式)在有形的存储介质中，如半导体存储器装置(例如，RAM、ROM、PROM、EEPROM或Flash可编程RAM)、磁性存储器设备(例如，软盘或硬盘)、光学存储设备(例如，CD-ROM)、PC卡(例如，PCMCIA卡)、或其他存储设备。计算机程序可以以任何形式被固定在使用多种通信技术中的任何技术可传送到计算机的信号内，这种通信技术包括但不限于，模拟技术、数字技术、光学技术、无线技术(例如，蓝牙)、网络技术和网络互连技术。计算机程序可以以任何形式被分发作为可移动存储介质(例如，缩绕软件(shrink wrapped software))，伴随有打印的或电子文档、被计算机系统预先加载(例如，在系统ROM或硬盘上)、或在通信系统上(例如，互联网或万维网)从服务器或者电子公告板上被分发。

[0282] 实施全部或部分本文中之前所描述功能的硬件逻辑(包括与可编程逻辑器件一起使用的可编程逻辑)可以使用传统的手工方法被设计，或可以使用各种工具设计、捕获、模拟或电子存档，这种工具例如是计算机辅助设计(CAD)、硬件描述语言(例如，VHDL或AHDL)、或PLD编程语言(例如，PALASM、ABEL、或CUPL)。

[0283] 可编程逻辑可以被永久地或者暂时地固定在有形的存储介质中，例如半导体存储设备(例如，RAM、ROM、PROM、EEPROM或Flash可编程RAM)、磁性存储设备(例如，软盘或硬盘)、光学存储设备(例如，CD-ROM)、或其他存储设备。可编程逻辑可以被固定在可使用多种通信技术中的任何技术传送到计算机的信号中，这种通信技术包括但不限于，模拟技术、数字技术、光学技术、无线技术(例如，蓝牙)、网络技术和网络互连技术。可编程逻辑可以被分发作为可移动存储介质，伴随有打印的或电子文档(例如，缩绕软件)、，计算机系统预先加载的(例如，在系统ROM或硬盘上)、或在通信系统上(例如，互联网或万维网)从服务器或者电子公告板分发。当然，本发明的一些实施例可以被实现为软件(例如，计算机程序产品)和硬件两者的结合。本发明的其他实施例是被实现为完全硬件或完全软件。

[0284] 本发明可以以其他特定形式被实施，而不偏离本发明的真正范围。任何对“发明”的引用都意指本发明示例性实施例，不应被解释为指本发明的所有实施例，除非上下文做出相反要求。所描述的实施例在所有方面都被视为只是说明性的和非限制性的。

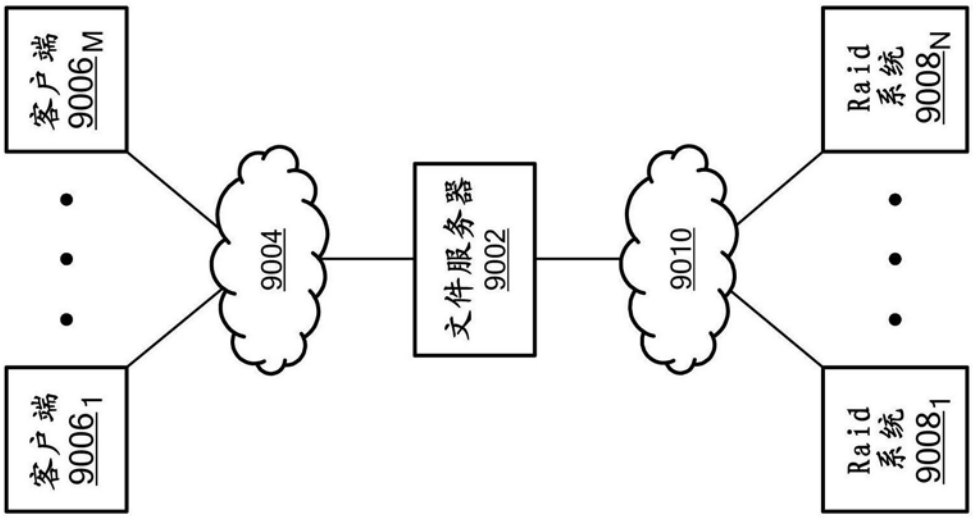

客户端 9006M
客户端 9006 1
9004
文件服务器 9002
9010
Raid 系统 9008N
Raid 系统 9008 1

图1

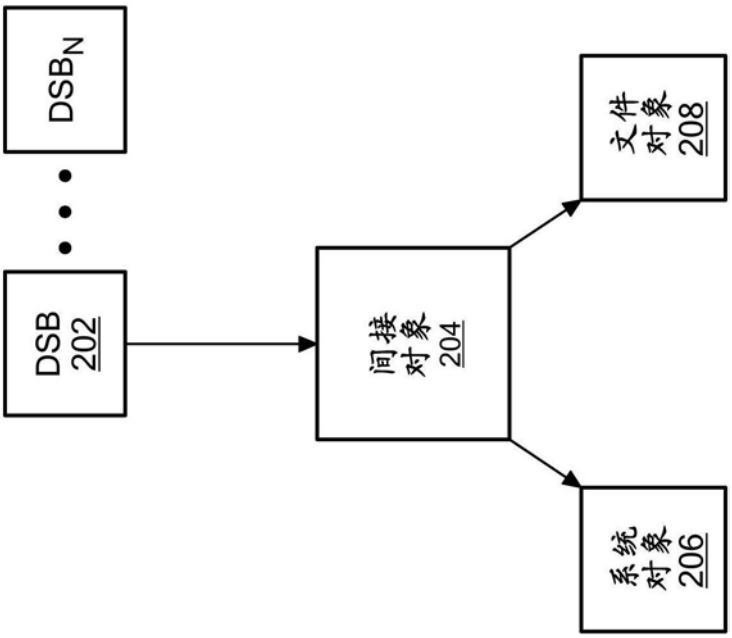

DSBN
DSB 202
间接对象 204
文件对象 208
系统对象 206

图2

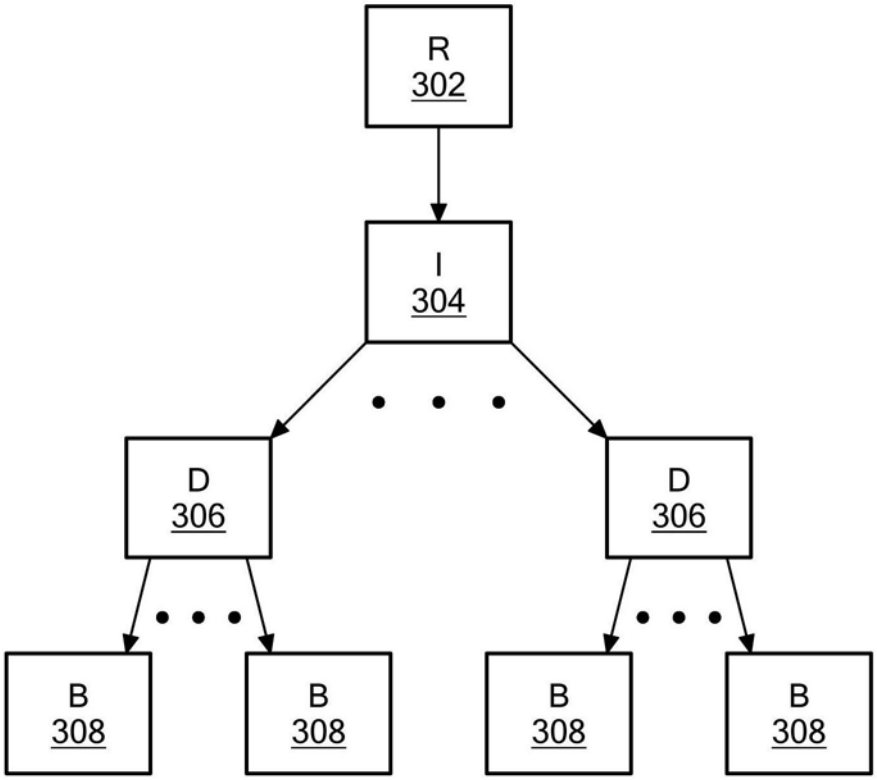


图3

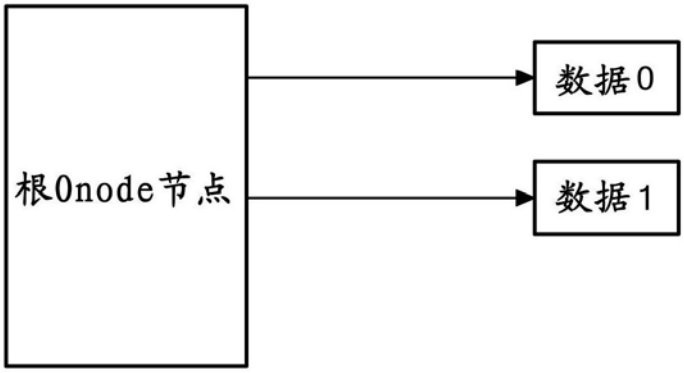


图4

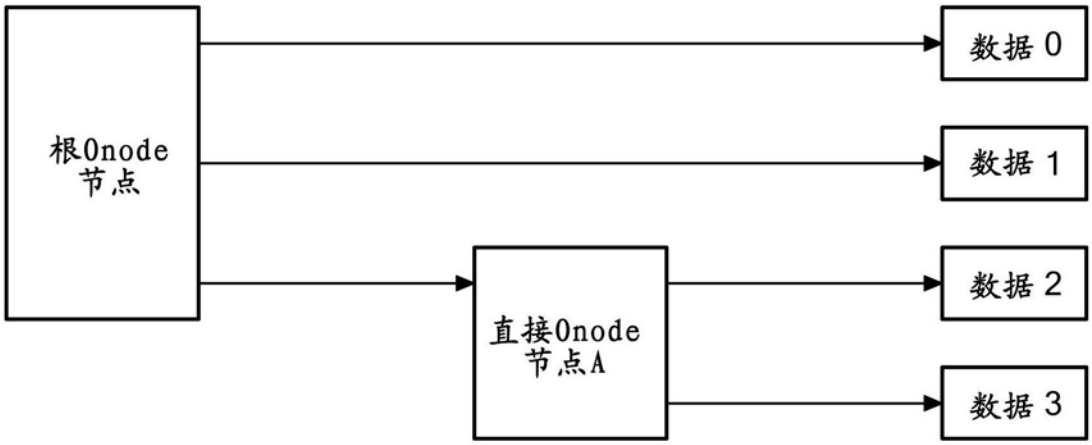

根Onode
节点
数据 0
数据 1
直接Onode
节点A
数据 2
数据 3

图5

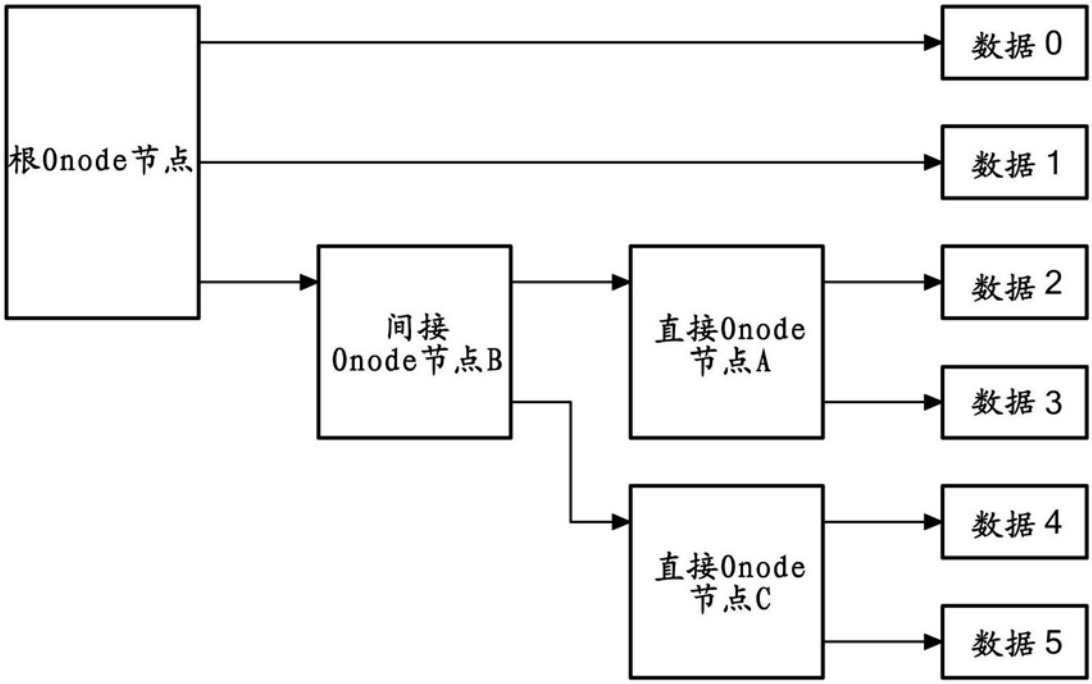

根Onode节点
数据 0
数据 1
间接
Onode节点B
直接Onode
节点A
数据 2
数据 3
直接Onode
节点C
数据 4
数据 5

图6

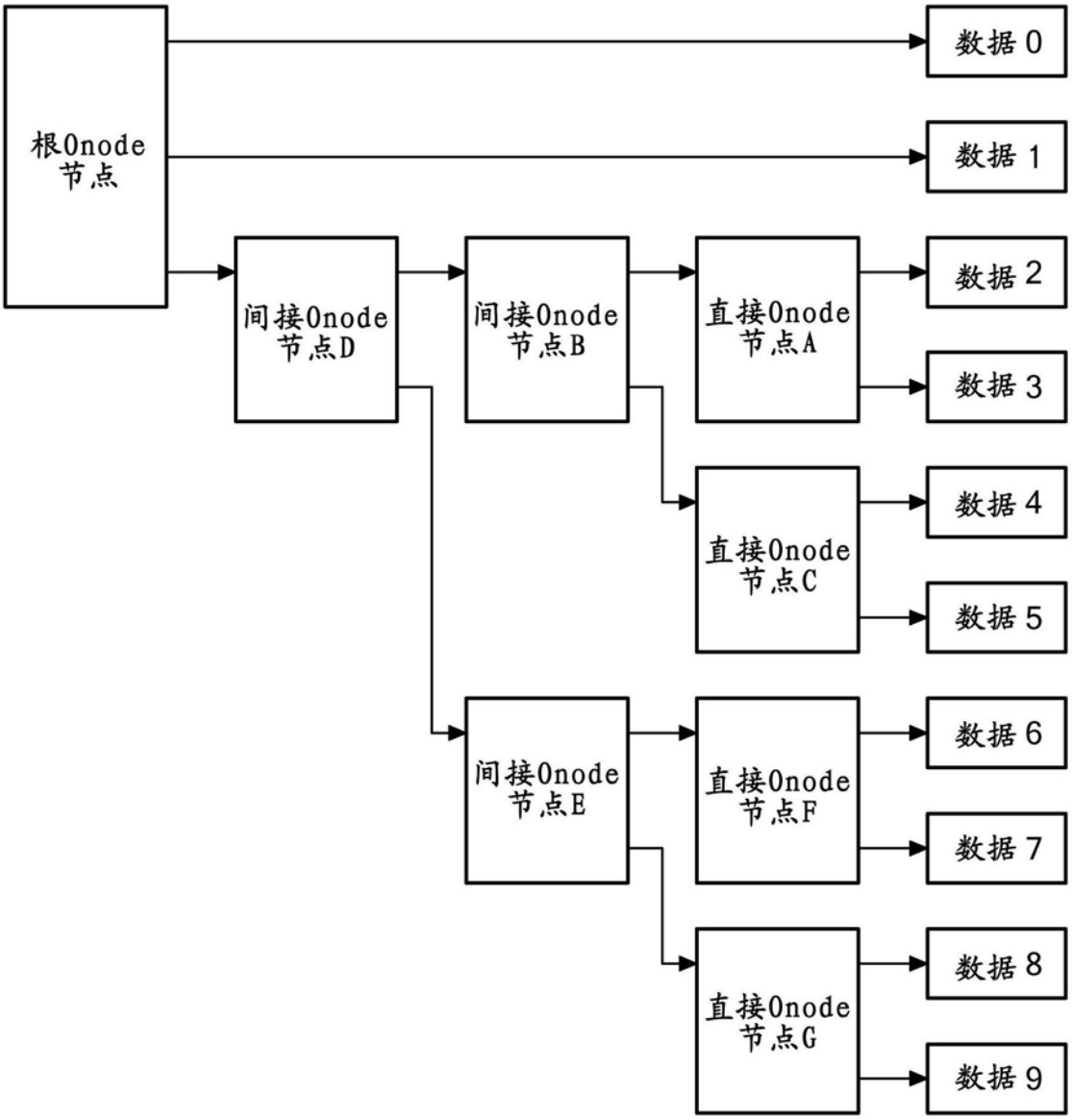

根Onode
节点
数据 0
数据 1
间接Onode
节点D
间接Onode
节点B
直接Onode
节点A
数据 2
数据 3
直接Onode
节点C
数据 4
数据 5
间接Onode
节点E
直接Onode
节点F
数据 6
数据 7
直接Onode
节点G
数据 8
数据 9

图7

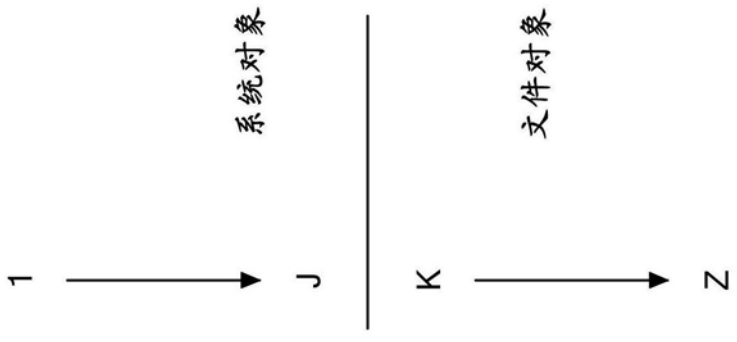

系统对象
文件对象
1
J
K
Z

图8

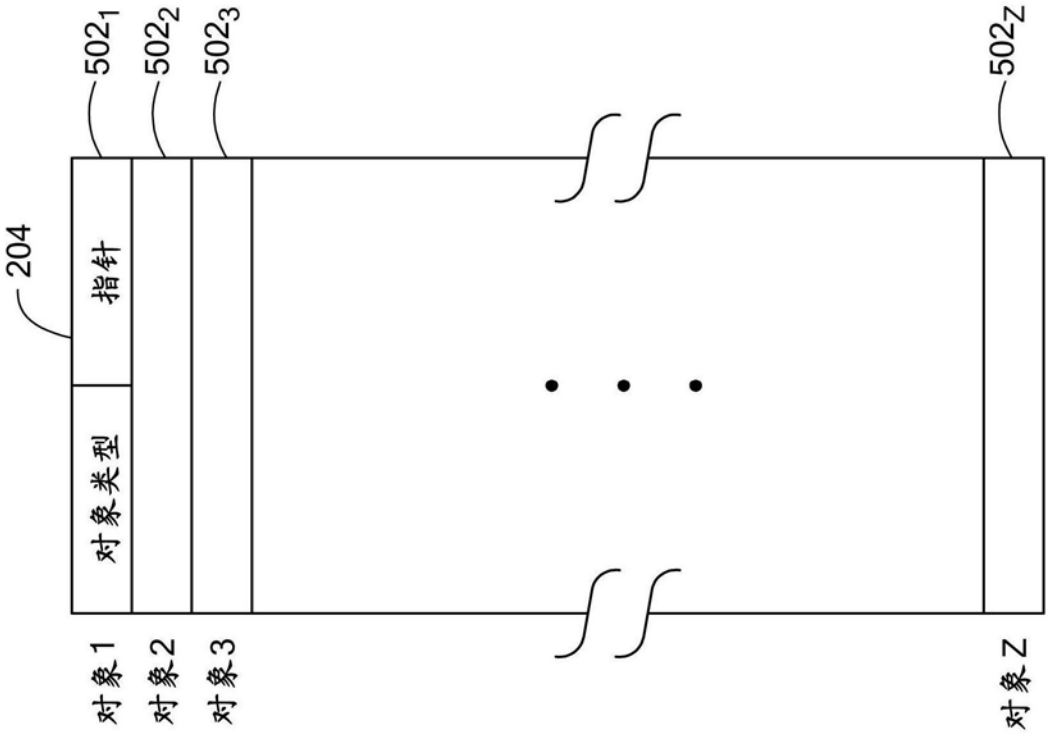


图9

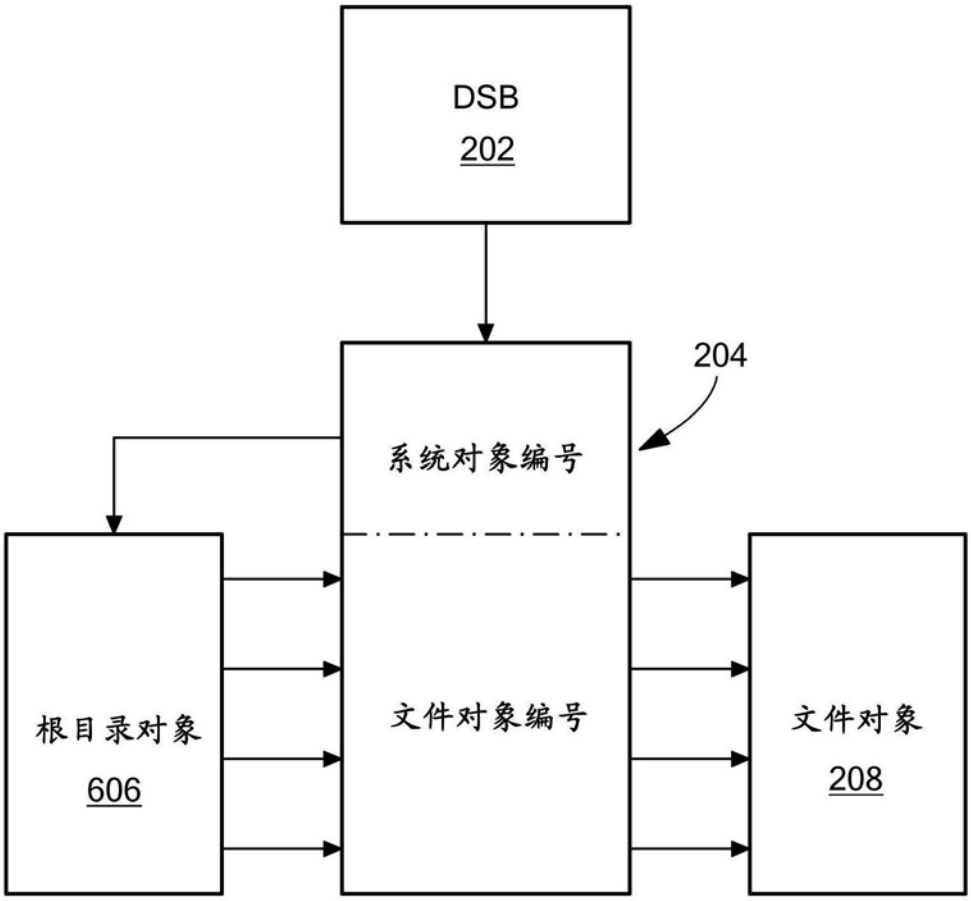


图10

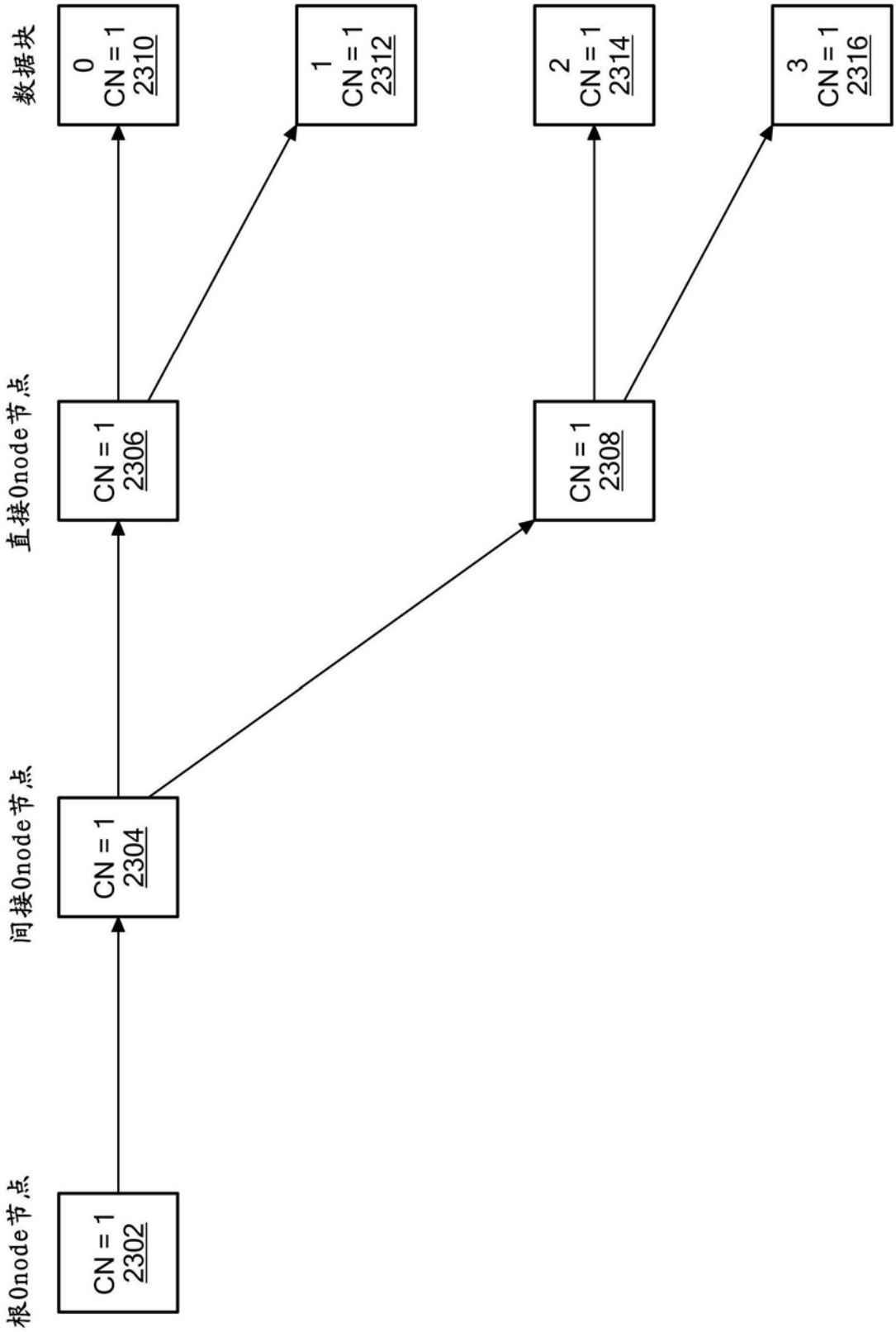

根0node节点
间接0node节点
直接0node节点
数据块
CN = 1
2302
CN = 1
2304
CN = 1
2306
CN = 1
2308
0
CN = 1
2310
1
CN = 1
2312
2
CN = 1
2314
3
CN = 1
2316

图11

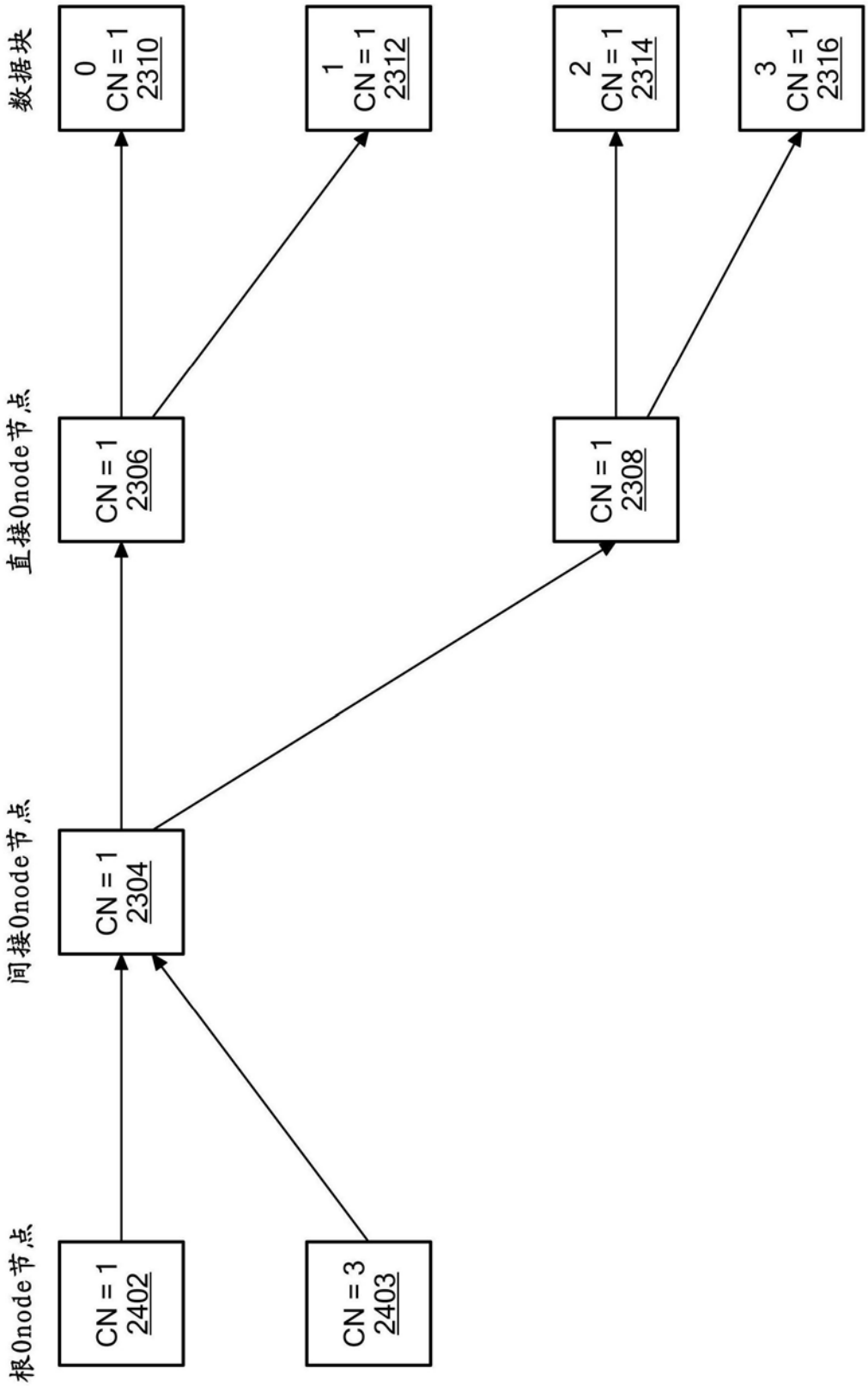

数据块
0
CN = 1
2310
1
CN = 1
2312
2
CN = 1
2314
3
CN = 1
2316
直接0node节点
CN = 1
2306
CN = 1
2308
间接0node节点
CN = 1
2304
根0node节点
CN = 1
2402
CN = 3
2403

图12

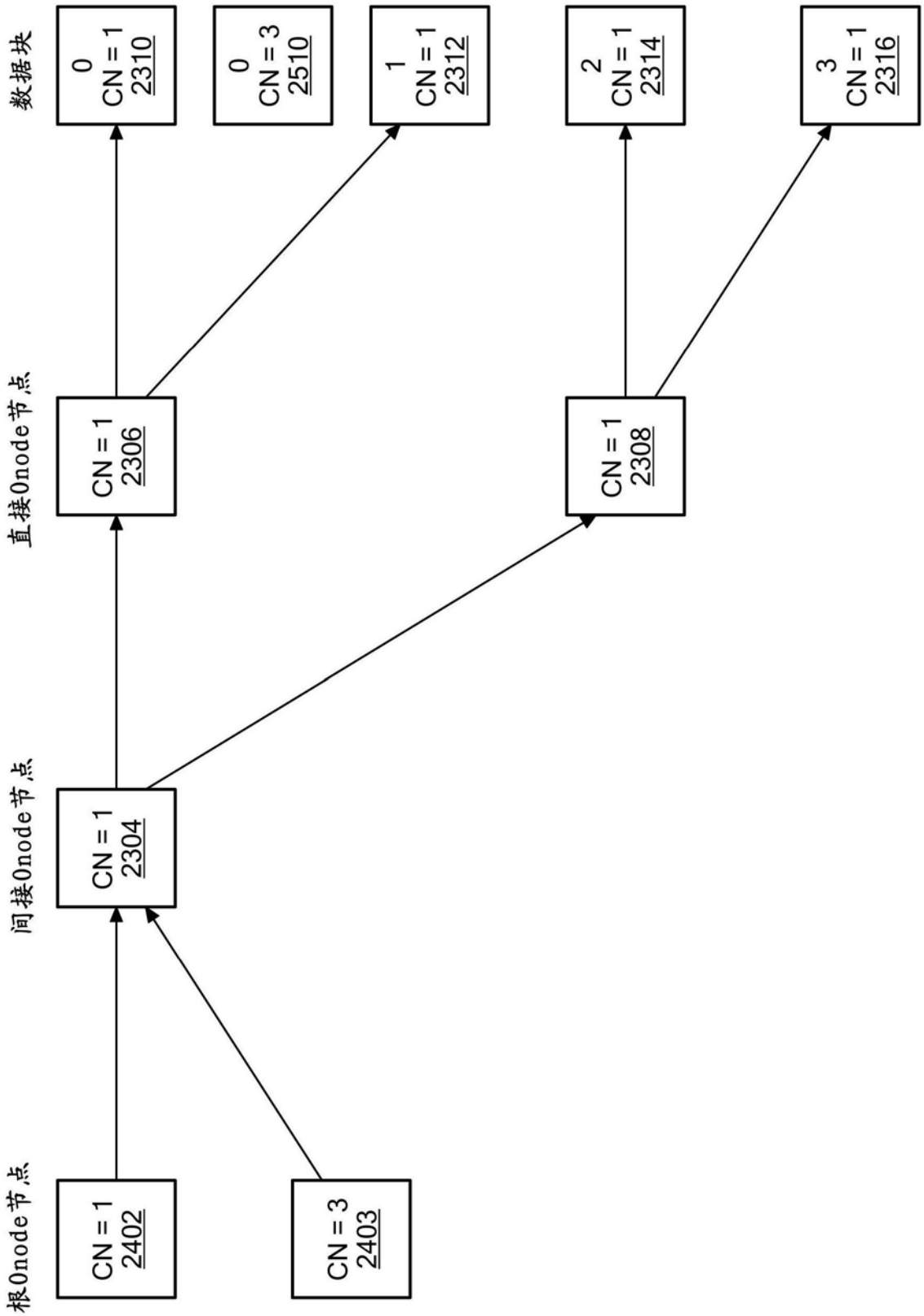

数据块
直接0node节点
间接0node节点
根0node节点
0
CN = 1
2310
0
CN = 3
2510
1
CN = 1
2312
2
CN = 1
2314
3
CN = 1
2316
CN = 1
2306
CN = 1
2308
CN = 1
2304
CN = 1
2402
CN = 3
2403

图13

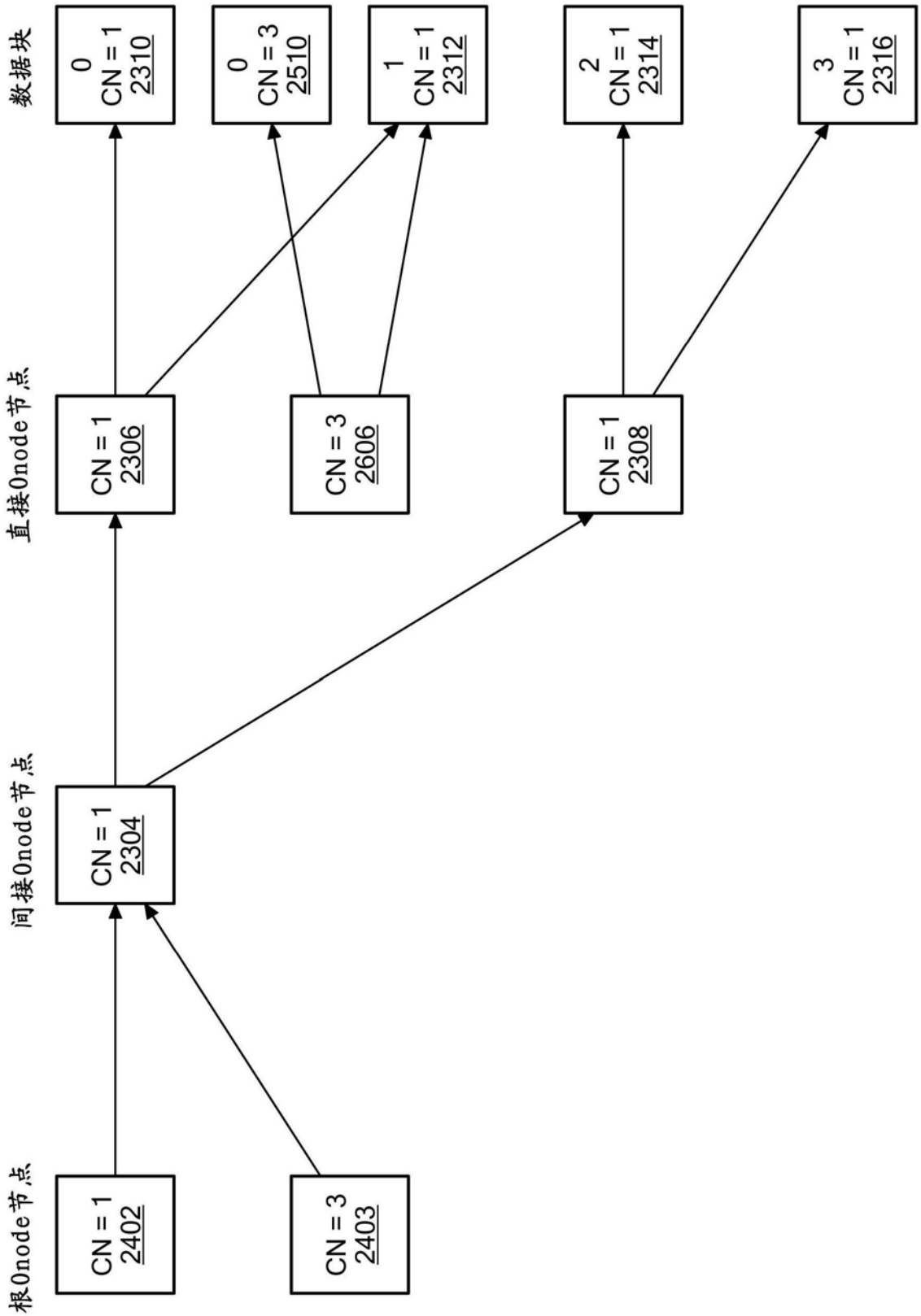

数据块
直接0node节点
间接0node节点
根0node节点
0
CN = 1
2310
0
CN = 3
2510
1
CN = 1
2312
2
CN = 1
2314
3
CN = 1
2316
CN = 1
2306
CN = 3
2606
CN = 1
2308
CN = 1
2304
CN = 1
2402
CN = 3
2403

图14

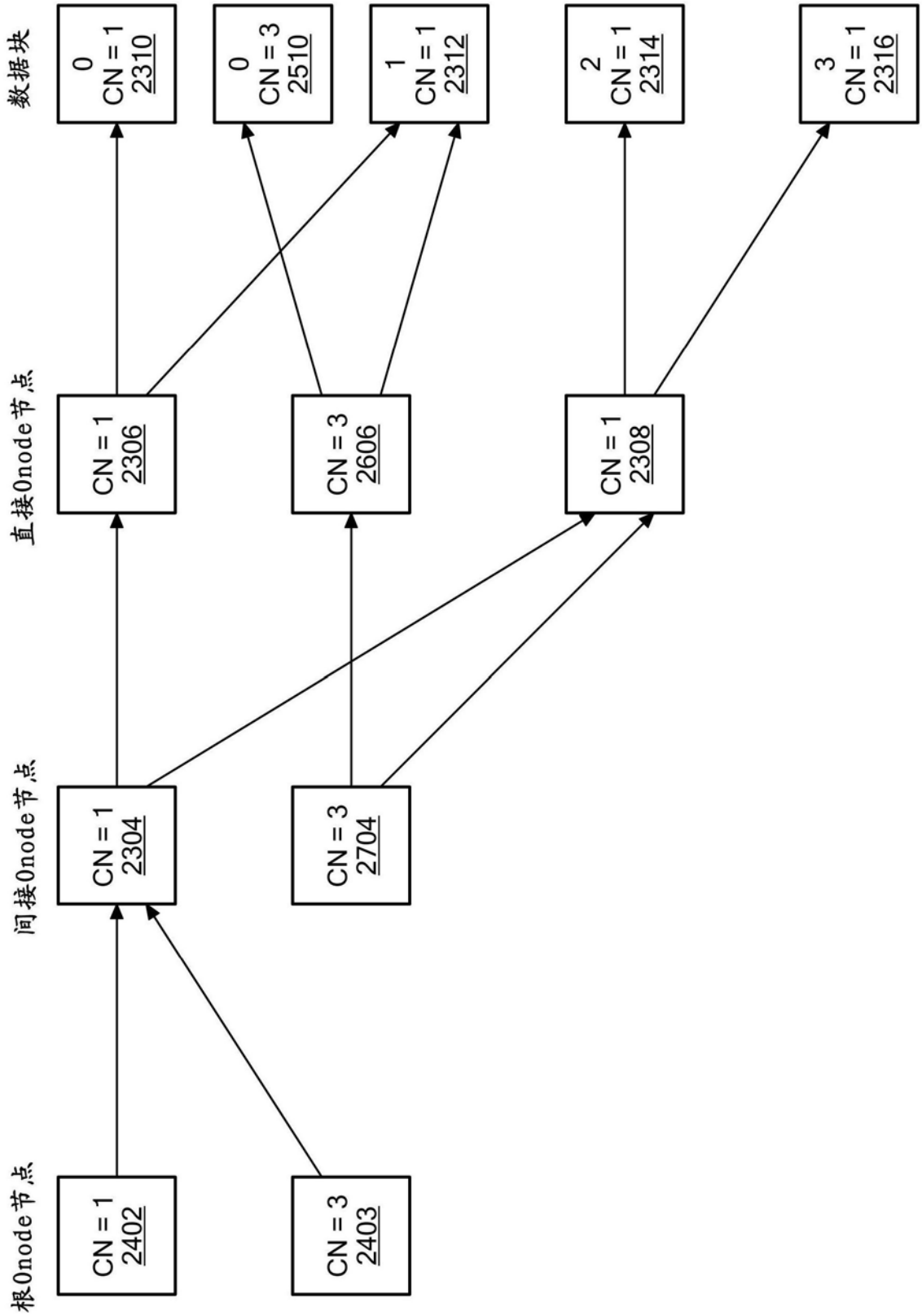

数据块
直接0node节点
间接0node节点
根0node节点
0
CN = 1
2310
0
CN = 3
2510
1
CN = 1
2312
2
CN = 1
2314
3
CN = 1
2316
CN = 1
2306
CN = 3
2606
CN = 1
2308
CN = 1
2304
CN = 3
2704
CN = 1
2402
CN = 3
2403

图15

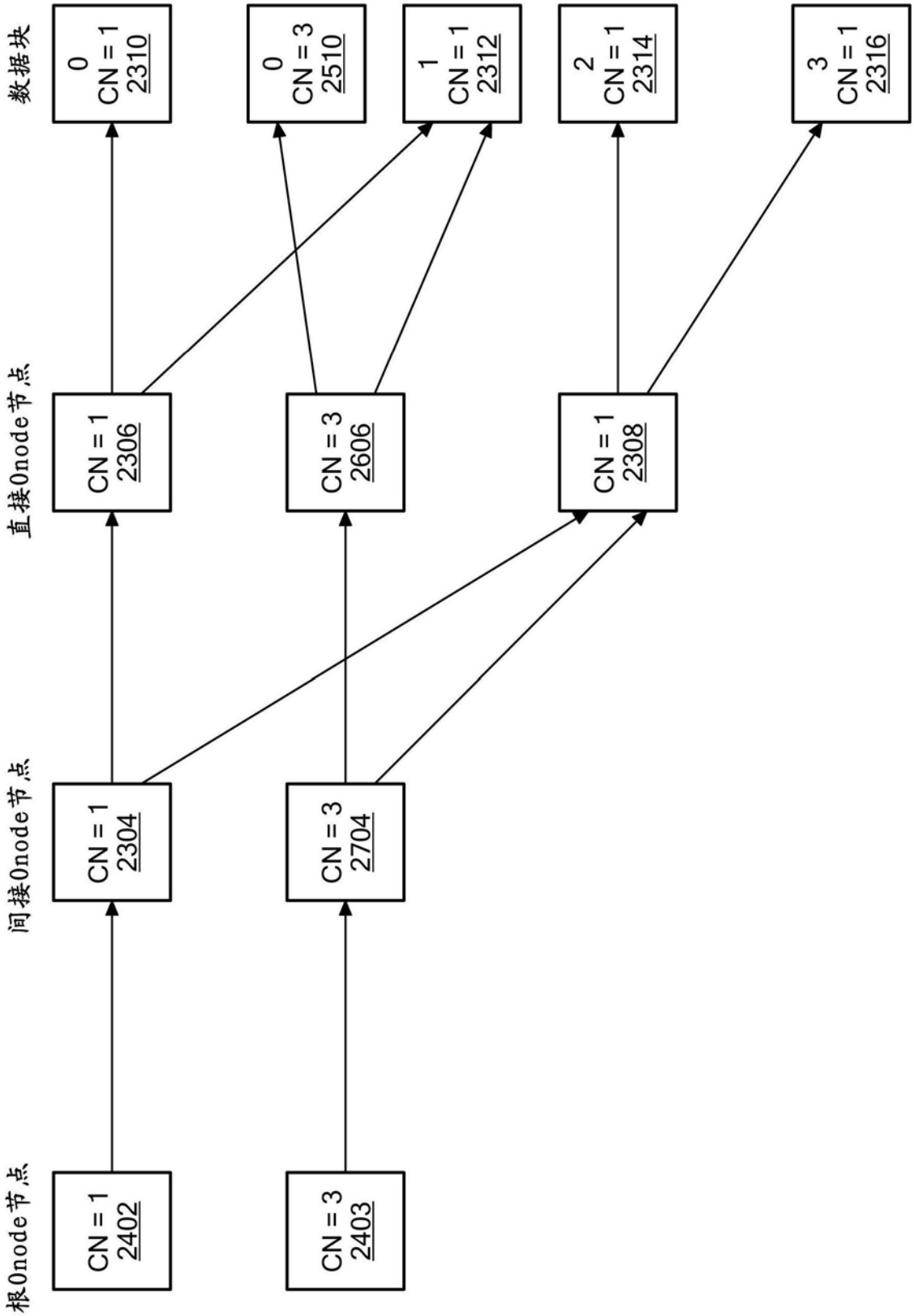

数据块
直接0node节点
间接0node节点
根0node节点
0
CN = 1
2310
0
CN = 3
2510
1
CN = 1
2312
2
CN = 1
2314
3
CN = 1
2316
CN = 1
2306
CN = 3
2606
CN = 1
2308
CN = 1
2304
CN = 3
2704
CN = 1
2402
CN = 3
2403

图16

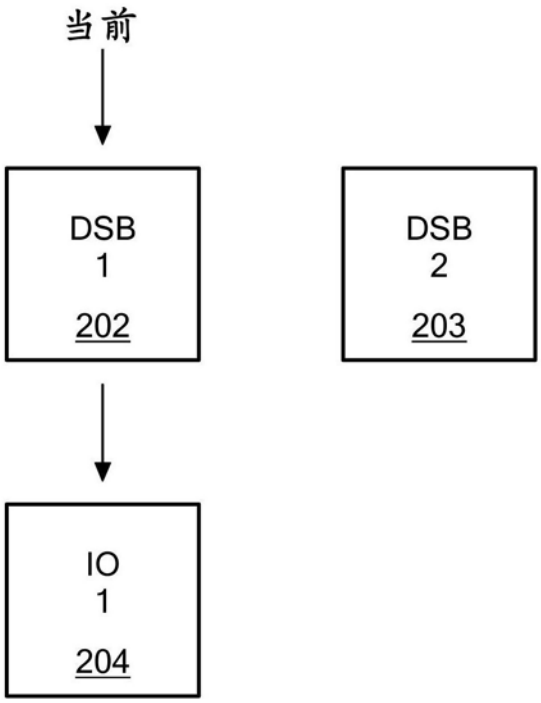


图17

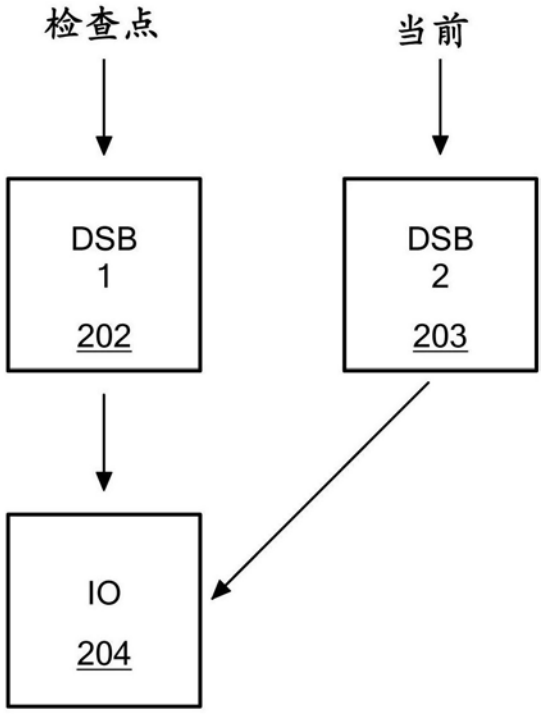


图18

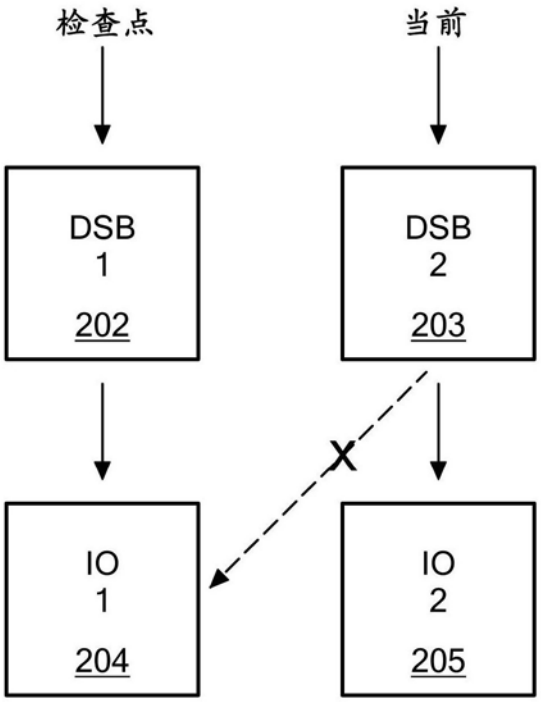

检查点
当前
DSB
1
202
DSB
2
203
X
IO
1
204
IO
2
205

图19

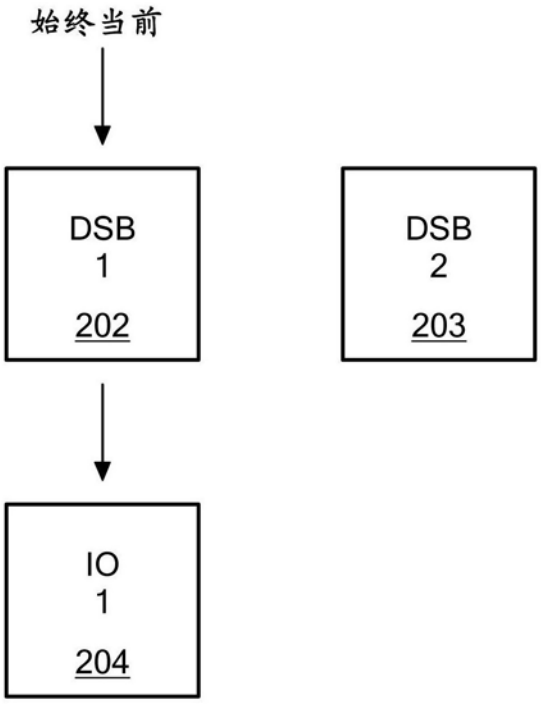

始终当前
DSB
1
202
DSB
2
203
IO
1
204

图20

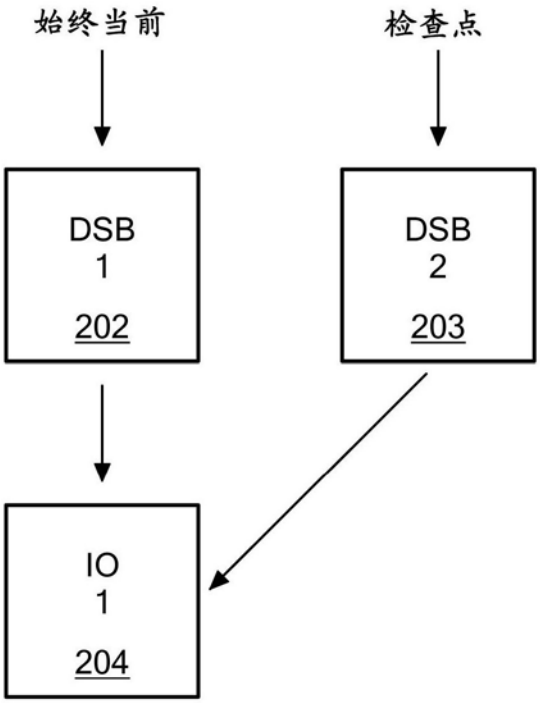

始终当前
检查点
DSB
1
202
DSB
2
203
IO
1
204

图21

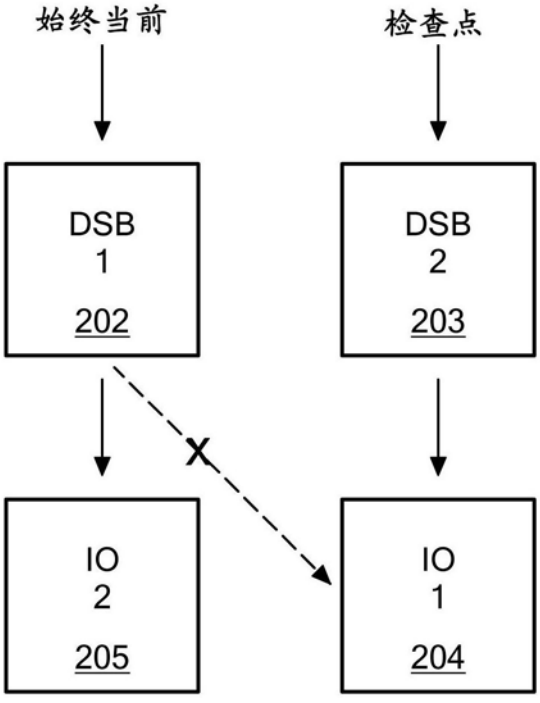

始终当前
检查点
DSB
1
202
DSB
2
203
X
IO
2
205
IO
1
204

图22

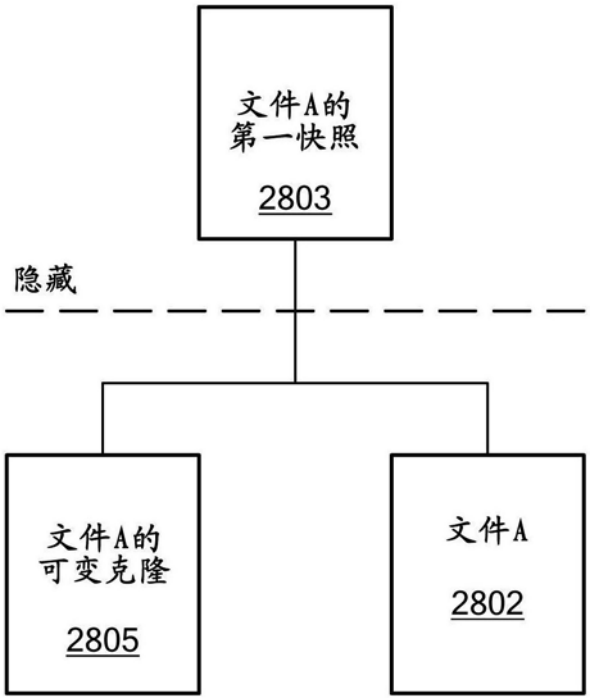


图23

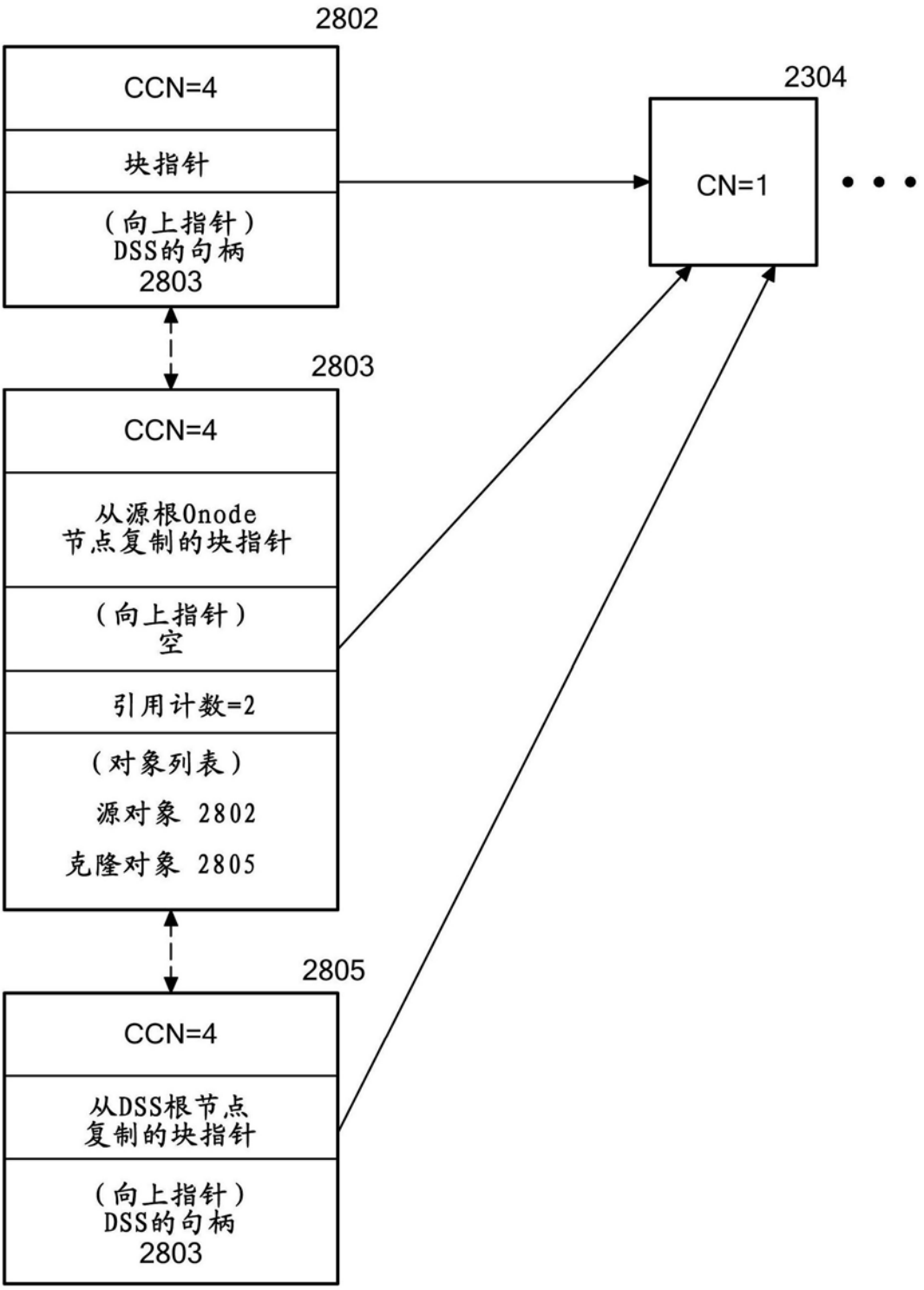

2802
CCN=4
块指针
（向上指针）
DSS的句柄
2803
2304
CN=1
2803
CCN=4
从源根0node
节点复制的块指针
（向上指针）
空
引用计数=2
（对象列表）
源对象 2802
克隆对象 2805
2805
CCN=4
从DSS根节点
复制的块指针
（向上指针）
DSS的句柄
2803

图24

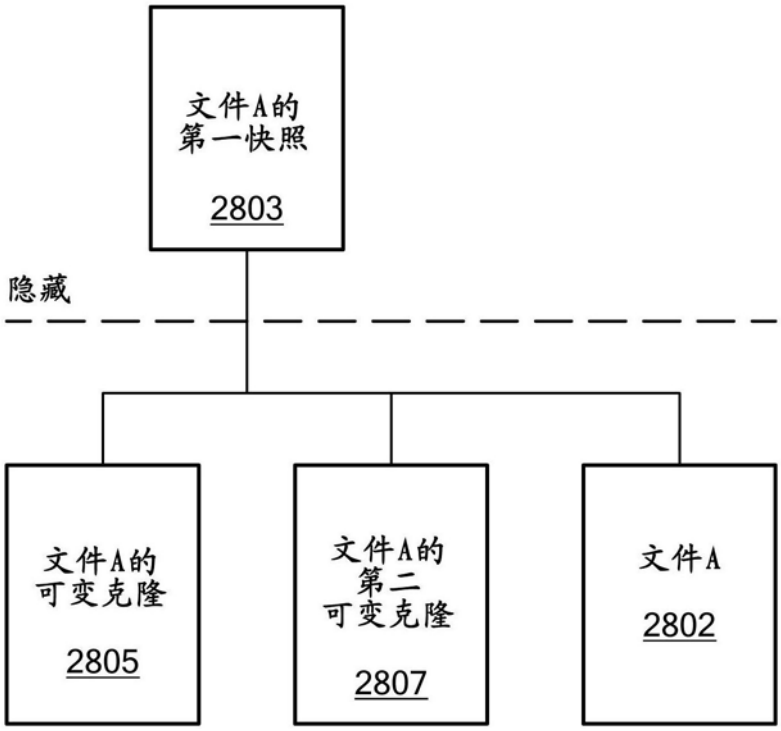


图25

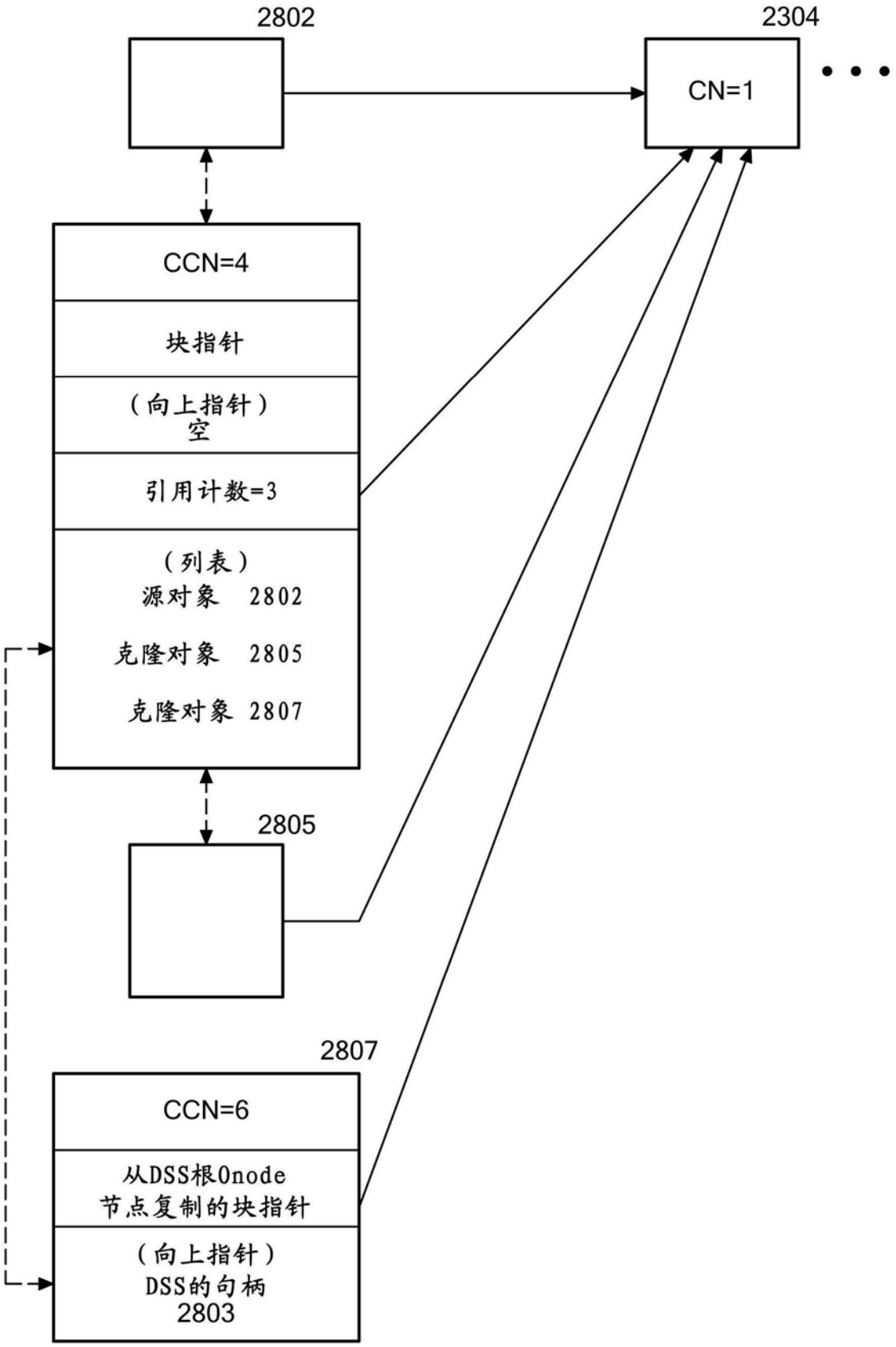

2802
2304
CN=1
CCN=4
块指针
（向上指针）
空
引用计数=3
（列表）
源对象 2802
克隆对象 2805
克隆对象 2807
2805
2807
CCN=6
从DSS根0node
节点复制的块指针
（向上指针）
DSS的句柄
2803

图26

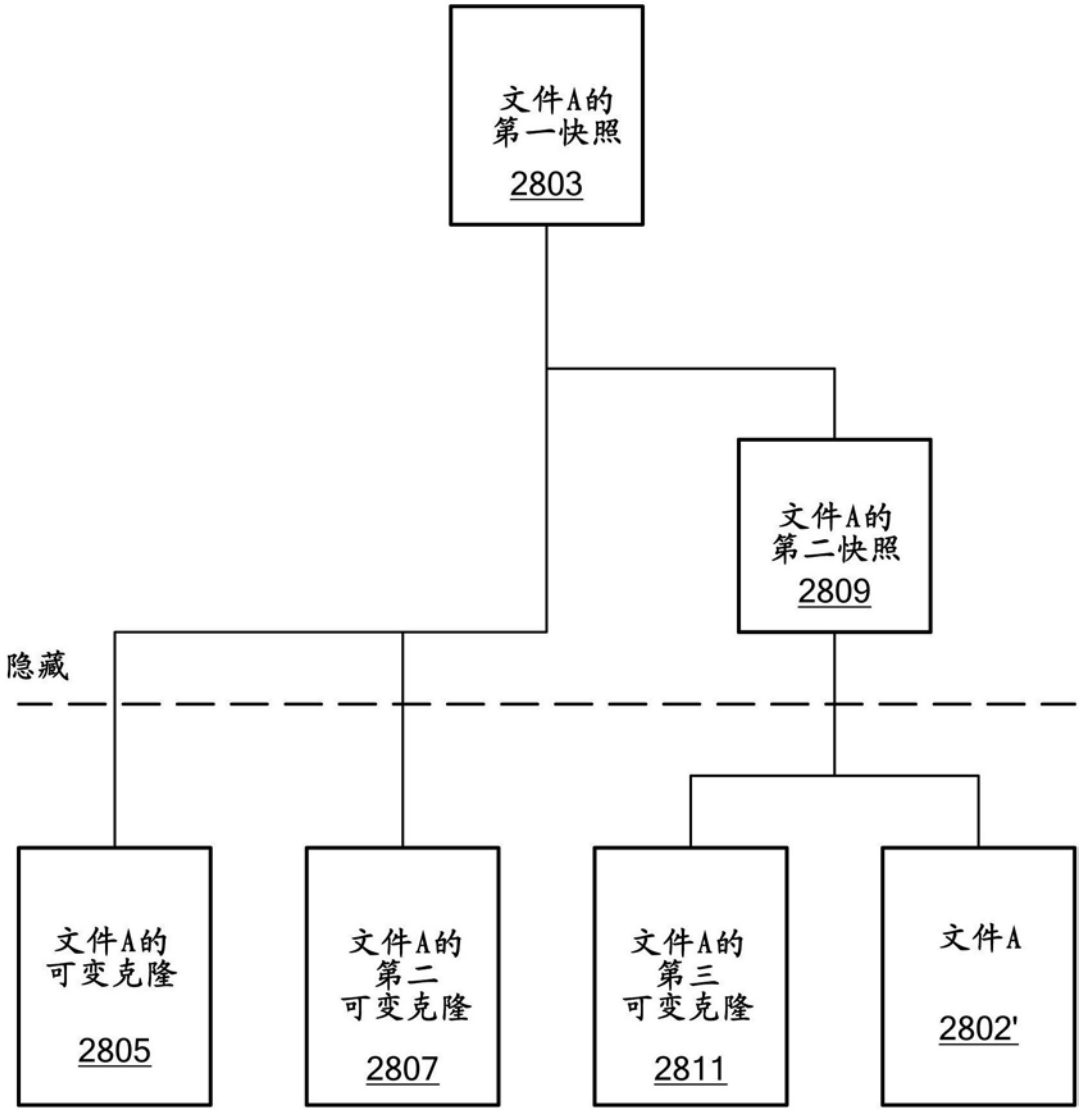

文件A的
第一快照
2803
文件A的
第二快照
2809
隐藏
文件A的
可变克隆
2805
文件A的
第二
可变克隆
2807
文件A的
第三
可变克隆
2811
文件A
2802'

图27

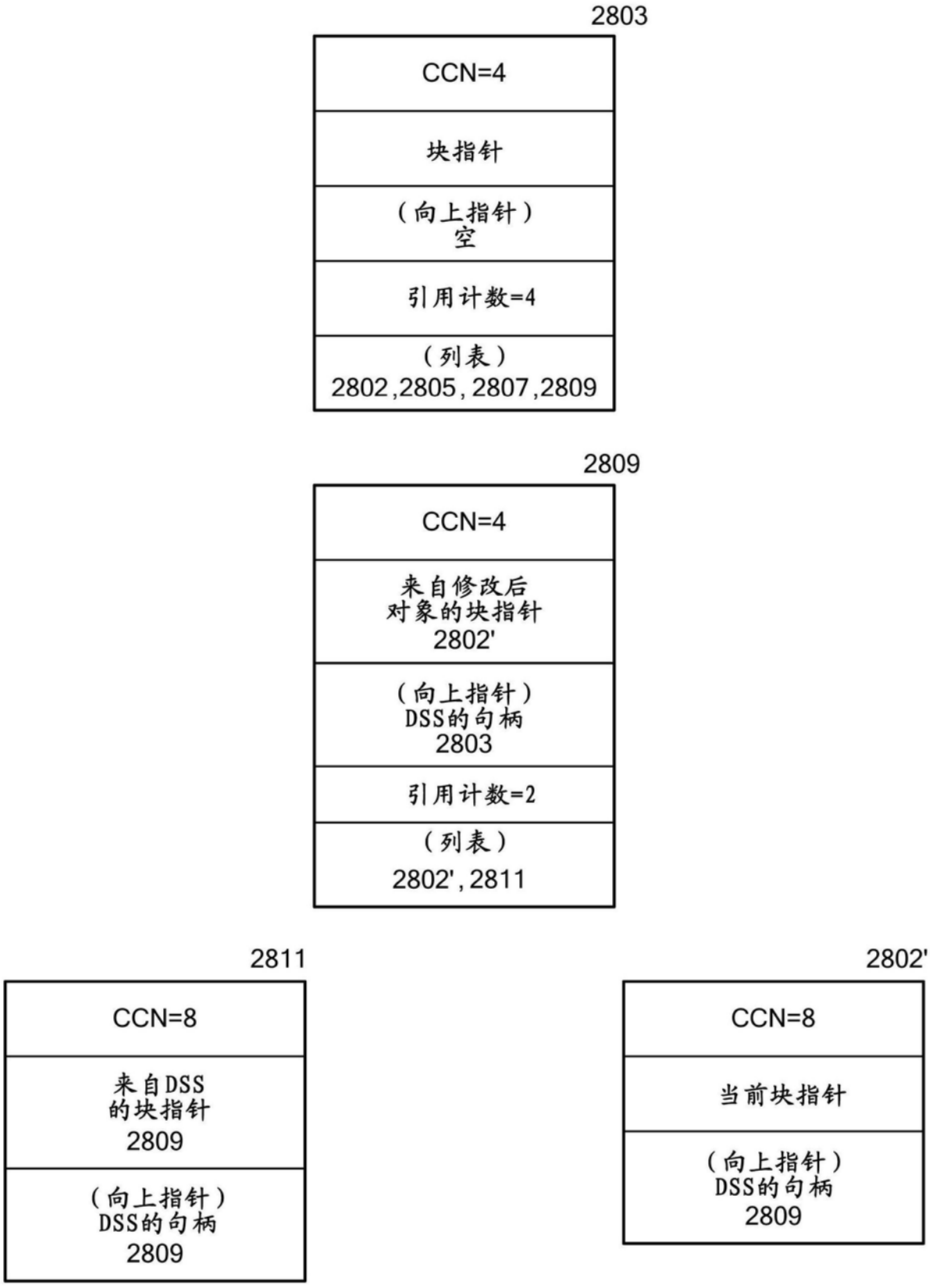

2803
CCN=4
块指针
(向上指针)
空
引用计数=4
(列表)
2802,2805,2807,2809
2809
CCN=4
来自修改后
对象的块指针
2802'
(向上指针)
DSS的句柄
2803
引用计数=2
(列表)
2802',2811
2811
CCN=8
来自DSS
的块指针
2809
(向上指针)
DSS的句柄
2809
2802'
CCN=8
当前块指针
(向上指针)
DSS的句柄
2809

图28

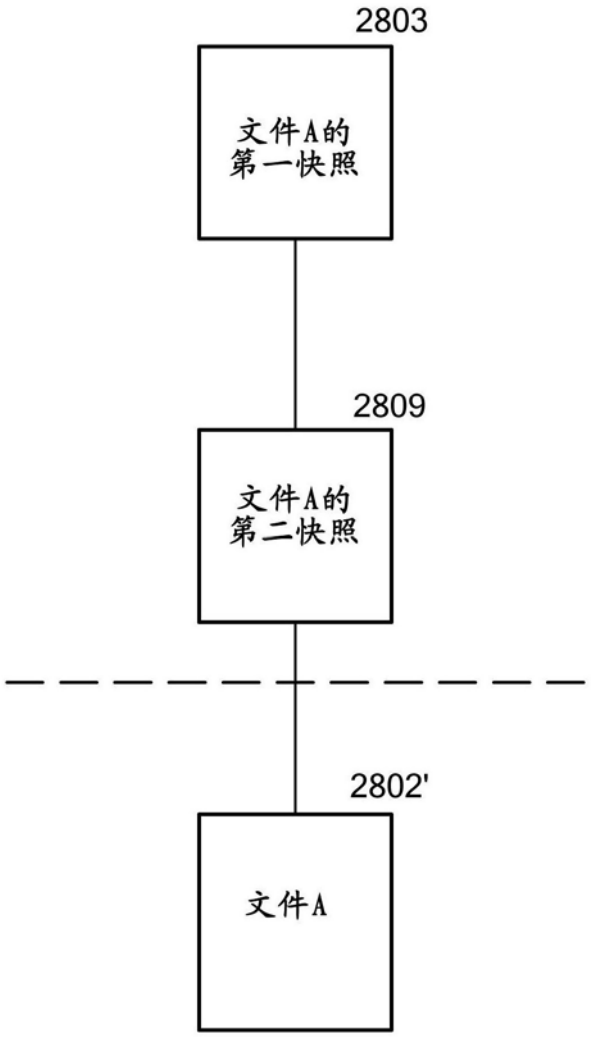


图29

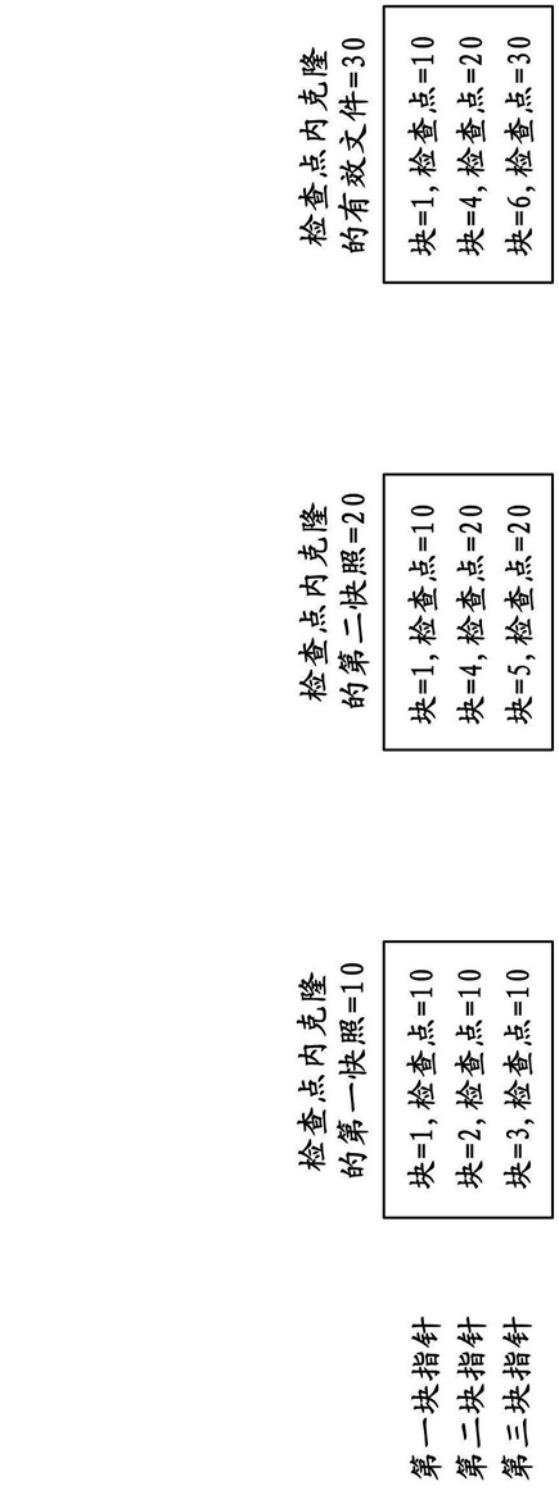

检查点内克隆
的第一快照=10
块=1,检查点=10
块=2,检查点=10
块=3,检查点=10
检查点内克隆
的第二快照=20
块=1,检查点=10
块=4,检查点=20
块=5,检查点=20
检查点内克隆
的有效文件=30
块=1,检查点=10
块=4,检查点=20
块=6,检查点=30
第一块指针
第二块指针
第三块指针

图30A


第一块指针
第二块指针
第三块指针
检查点内克隆
的第一快照=10
块=1,检查点=10
块=2,检查点=10
块=3,检查点=10
检查点内克隆
的第二快照=20
块=1,检查点=10
块=,检查点=有效的
块=5,检查点=20
检查点内克隆
的有效文件=30
块=1,检查点=10
块=4,检查点=30
块=6,检查点=30

图30B


检查点内克隆
的有效文件=30
块=1,检查点=30
块=4,检查点=30
块=6,检查点=30
检查点内克隆
的第一快照=10
块=,检查点=有效的
块=2,检查点=10
块=3,检查点=10
第一块指针
第二块指针
第三块指针

图30C