(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 111198660 A
(43)申请公布日 2020.05.26

(21)申请号 201911371660.X

(22)申请日 2019.12.26

(71)申请人 天津中科曙光存储科技有限公司
地址 300000 天津市滨海新区华苑产业区(环外)海泰华科大街15号3层

(72)发明人 石胜男 王云飞 沈海嘉 郭照斌

(74)专利代理机构 北京德恒律治知识产权代理有限公司 11409
代理人 章社杲 卢军峰

(51)Int.Cl.
G06F 3/06(2006.01)
G06F 9/52(2006.01)

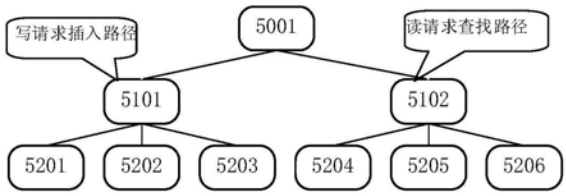
权利要求书2页 说明书7页 附图4页

(54)发明名称

一种B+树遍历的方法及装置

(57)摘要

本发明公开了一种B+树遍历的方法,包括在读请求过程中,给某一节点加读锁时,并给其父节点解锁;在写请求过程中,给某一节点加写锁时,并给其父节点解锁。本发明对传统B+树遍历机制进行改进,通过及时归还父节点的锁,解决了在IO并发量急剧增加的时候,由于非叶子节点持锁时间长导致的性能下降的问题;通过控制持锁区间,降低了由于不必要的持锁时间导致的开销;有助于提高bcache的IO并发处理能力。



CN 111198660 A

1.一种B+树遍历的方法,其特征在于,包括

在读请求过程中,给某一节点加读锁时,并给其父节点解锁;

在写请求过程中,给某一节点加写锁时,并给其父节点解锁。

2.根据权利要求1所述的B+树遍历的方法,其特征在于,所述“在读请求过程中,给某一节点加读锁时,给其父节点解锁”中,包括

根节点加读锁;

遍历根节点的bkey,找到符合条件的非叶子节点;

给任一非叶子节点加读锁时,并给该非叶子节点的父节点解锁;

遍历非叶子节点的bkey,直至找到符合条件的叶子节点;

给叶子节点加读锁时,并给该叶子节点的父节点解锁;

在叶子节点中查找匹配的数据;

给叶子节点解锁;

结束。

3.根据权利2所述的B+树遍历的方法,其特征在于,所述“在写请求过程中,给某一节点加读锁时,并给其父节点解锁”中,包括

根节点加读锁;

遍历根节点的bkey,找到符合条件的非叶子节点;

给任一非叶子节点加读锁时,给该非叶子节点的父节点解锁;

遍历非叶子节点的bkey,直至找到符合条件的叶子节点;

给叶子节点加写锁时,给该叶子节点的父节点解锁;

在叶子节点中插入新增bkey;

给叶子节点解锁;

结束。

4.根据权利要求3所述的B+树遍历的方法,其特征在于,当所述在叶子节点中插入新增bkey时该节点中bkey已满,则进行节点分裂;任一节点分裂时,该节点下新增节点的bkey插入到该节点的上级节点中;且给任一节点加写锁时,并给上级节点解锁。

5.一种B+树遍历的装置,其特征在于,包括处理器和存储器,存储器中存储有程序,程序被处理器运行时,执行:

在读请求过程中,给某一节点加读锁时,并给其父节点解锁;

在写请求过程中,给某一节点加写锁时,并给其父节点解锁。

6.根据权利要求5所述的B+树遍历的装置,其特征在于,程序执行所述“在读请求过程中,给某一节点加读锁时,给其父节点解锁”中,包括

根节点加读锁;

遍历根节点的bkey,找到符合条件的非叶子节点;

给任一非叶子节点加读锁时,并给该非叶子节点的父节点解锁;

遍历非叶子节点的bkey,直至找到符合条件的叶子节点;

给叶子节点加读锁时,并给该叶子节点的父节点解锁;

在叶子节点中查找匹配的数据;

给叶子节点解锁;

结束。

7.根据权利6所述的B+树遍历的方法，其特征在于，程序执行所述“在写请求过程中，给某一节点加读锁时，并给其父节点解锁”中，包括

根节点加读锁；

遍历根节点的bkey，找到符合条件的非叶子节点；

给任一非叶子节点加读锁时，给该非叶子节点的父节点解锁；

遍历非叶子节点的bkey，直至找到符合条件的叶子节点；

给叶子节点加写锁时，给该叶子节点的父节点解锁；

在叶子节点中插入新增bkey；

给叶子节点解锁；

结束。

8.根据权利要求7所述的B+树遍历的装置，其特征在于，当所述在叶子节点中插入新增bkey时该节点中bkey已满，则进行节点分裂；任一节点分裂时，该节点下新增节点的bkey插入到该节点的上级节点中；且给任一节点加写锁时，并给上级节点解锁。

一种B+树遍历的方法及装置

技术领域

[0001] 本发明涉及计算机处理技术领域,具体而言,为一种B+树遍历的方法及装置。

背景技术

[0002] bcache是linux内核块设备层cache(缓存),它使用类似SSD(固态硬盘)来作为HDD(机械硬盘)硬盘的cache,从而起到加速的作用。如图1所示的是linux Kernel v4.16.12里的bcache版本,SSD盘sdb为sdc系列后端HDD低速磁盘提供bcache服务。bcache将随机写转化为顺序写,它将SSD看做是缓存设备,首先将IO数据写到SSD上,然后通过bcache内部特有的回刷机制,将这些缓存数据有序的写到磁盘或者阵列上,从而获得更高的随机写性能。

[0003] bcache是用B+树来管理缓存数据与HDD上数据块的对应关系的。B+树索引的k-v(key-value,键-值)结构在bcache中称为bkey。如图2所示,bcache会将系统中的多块HDD空间编址为一个地址空间。以HDD盘的id(编号)与IO请求的LBA(Logical Block Address,逻辑区地址)为索引建立B+树,叶子节点中存放的是实际的映射。根据这些bkey可以找到缓存数据以及在HDD设备上的位置。

[0004] IO的写流程,会先将IO数据写到SSD盘,并根据这笔IO请求的HDD盘的id和LBA地址(HDD offset)、长度(HDD length),以及写入的SSD盘的地址(SSD offset)生成一个bkey,插入到这棵B+树中。IO的读流程,就是将请求的HDD盘的id和LBA地址,作为查询条件去B+树上检索,根据检索到的bkey,找到SSD盘的地址(SSD offset),从而根据这个地址,从SSD盘读取上来相应的缓存数据,返回给请求发起者,如果未从B+树上检索到数据,那么bcache就会从HDD盘读上来相应的数据,同时将其缓存到SSD盘上,将对应的bkey插入到B+树上,便于下次查找命中。可以看出,这棵B+树访问效率对整个系统的性能起着至关重要的作用;它就相当于一个小型的数据库,是整个bcache的核心。

[0005] 所有的IO请求都会访问B+树,B+树作为公共的资源,使用了读写锁来确保数据访问的一致性。以读请求举例,假设是一个三层结构的B+树(图3-1),要查找的数据对应的bkey在3204中,通过遍历B+树查找到3204中的某个bkey的过程如图3-2所示。

[0006] 遍历B+树的步骤如下:

[0007] (1)根节点3001加读锁;

[0008] (2)遍历3001里面的bkey,找到3102;

[0009] (3)给3102加读锁;

[0010] (4)遍历3102里面的bkey,找到3204;

[0011] (5)给3204加读锁;

[0012] (6)遍历3204里面的bkey,查找匹配数据;

[0013] (7)给3204解锁;

[0014] (8)给3102解锁;

[0015] (9)给3001解锁;

[0016] (10)结束。

[0017] 由此可以看出根节点3001的加锁时间占据了整个查找过程。B+树用的是读写锁，B+树的访问者分读者和写者，IO的读请求，要对B+树进行读访问，需要获取到读锁；IO的写请求，需要插入bkey，对B+树进行修改，这个时候需要获取到写锁。读写锁的原理是，当读写锁是加写锁状态时，在写锁释放之前，所有的读或写请求都会被阻塞；当读写锁是加读锁状态时，所有试图获取读锁的请求都可以或取到读锁，但是写请求拿不到锁，必须等所有的读者都释放读锁时，才可以拿到写锁。当读写锁处于读模式时，此时如有一个线程试图以写模式或得该锁时，读写锁会拒绝后续的读请求，这样可以避免读者长期占用锁，而等待的写锁请求得不到满足。按照现有的B+树的遍历机制，可以推测，当IO读写请求压力增大的时候，不必要的持锁区间必然导致性能下降。

[0018] 有鉴于此，特提出本发明。

发明内容

[0019] 针对现有技术中的缺陷，本发明提供一种B+树遍历的方法和装置，以利于提高bcache的IO并行处理能力。

[0020] 为了实现上述目的，本发明的技术方案为：

[0021] 一种B+树遍历的方法，包括

[0022] 在读请求过程中，给某一节点加读锁时，并给其父节点解锁；

[0023] 在写请求过程中，给某一节点加写锁时，并给其父节点解锁。

[0024] 优选的，上述的B+树遍历的方法中，所述“在读请求过程中，给某一节点加读锁时，给其父节点解锁”中，包括

[0025] 根节点加读锁；

[0026] 遍历根节点的bkey，找到符合条件的非叶子节点；

[0027] 给任一非叶子节点加读锁时，并给该非叶子节点的父节点解锁；

[0028] 遍历非叶子节点的bkey，直至找到符合条件的叶子节点；

[0029] 给叶子节点加读锁时，并给该叶子节点的父节点解锁；

[0030] 在叶子节点中查找匹配的数据；

[0031] 给叶子节点解锁；

[0032] 结束。

[0033] 优选的，上述的B+树遍历的方法中，所述“在写请求过程中，给某一节点加读锁时，并给其父节点解锁”中，包括

[0034] 根节点加读锁；

[0035] 遍历根节点的bkey，找到符合条件的非叶子节点；

[0036] 给任一非叶子节点加读锁时，给该非叶子节点的父节点解锁；

[0037] 遍历非叶子节点的bkey，直至找到符合条件的叶子节点；

[0038] 给叶子节点加写锁时，给该叶子节点的父节点解锁；

[0039] 在叶子节点中插入新增bkey；

[0040] 给叶子节点解锁；

[0041] 结束。

[0042] 优选的，上述的B+树遍历的方法中，当所述在叶子节点中插入新增bkey时该节点

中bkey已满,则进行节点分裂;任一节点分裂时,该节点下新增节点的bkey插入到该节点的上级节点中;且给任一节点加写锁时,并给上级节点解锁。

[0043] 一种B+树遍历的装置,包括处理器和存储器,存储器中存储有程序,程序被处理器运行时,执行:

[0044] 在读请求过程中,给某一节点加读锁时,并给其父节点解锁;

[0045] 在写请求过程中,给某一节点加写锁时,并给其父节点解锁。

[0046] 优选的,上述的B+树遍历的装置中,程序执行所述“在读请求过程中,给某一节点加读锁时,给其父节点解锁”中,包括

[0047] 根节点加读锁;

[0048] 遍历根节点的bkey,找到符合条件的非叶子节点;

[0049] 给任一非叶子节点加读锁时,并给该非叶子节点的父节点解锁;

[0050] 遍历非叶子节点的bkey,直至找到符合条件的叶子节点;

[0051] 给叶子节点加读锁时,并给该叶子节点的父节点解锁;

[0052] 在叶子节点中查找匹配的数据;

[0053] 给叶子节点解锁;

[0054] 结束。

[0055] 优选的,上述的B+树遍历的装置中,程序执行所述“在写请求过程中,给某一节点加读锁时,并给其父节点解锁”中,包括

[0056] 根节点加读锁;

[0057] 遍历根节点的bkey,找到符合条件的非叶子节点;

[0058] 给任一非叶子节点加读锁时,给该非叶子节点的父节点解锁;

[0059] 遍历非叶子节点的bkey,直至找到符合条件的叶子节点;

[0060] 给叶子节点加写锁时,给该叶子节点的父节点解锁;

[0061] 在叶子节点中插入新增bkey;

[0062] 给叶子节点解锁;

[0063] 结束。

[0064] 优选的,上述的B+树遍历的装置中,当所述在叶子节点中插入新增bkey时该节点中bkey已满,则进行节点分裂;任一节点分裂时,该节点下新增节点的bkey插入到该节点的上级节点中;且给任一节点加写锁时,并给上级节点解锁。

[0065] 本发明的有益效果为:

[0066] 本发明方法对传统B+树遍历机制进行改进,通过及时归还父节点的锁,解决了在IO并发量急剧增加的时候,由于非叶子节点持锁时间长导致的性能下降的问题;通过控制持锁区间,降低了由于不必要的持锁时间导致的开销;有助于提高bcache的IO并发处理能力。

附图说明

[0067] 为了更清楚地说明本发明具体实施方式或现有技术中的技术方案,下面将对具体实施方式或现有技术描述中所需要使用的附图作简单地介绍。在所有附图中,类似的元件或部分一般由类似的附图标记标识。附图中,各元件或部分并不一定按照实际的比例绘制。

[0068] 图1为传统技术中bcache服务机制示意图；

[0069] 图2为传统技术中B+树管理缓存数据与HDD上数据块的对应关系示意图；

[0070] 图3中3-1为传统技术一个实施例中B+树结构；3-2为遍历图3-1中B+树查找数据的流程图；

[0071] 图4中：4-1为本发明方法的一个具体实施例的B+树结构；

[0072] 4-2为图4-1所示B+树的遍历查找流程图；

[0073] 4-3为图4-1所示B+树的插入流程图；

[0074] 4-4为图4-1所示B+树的节点分裂示意图；

[0075] 4-5为读流程加解锁顺序示意图；

[0076] 4-6为分裂流程的加解锁顺序示意图；

[0077] 图5为本发明系统一个具体实施例中对B+树的插入流程与查找流程示意图。

具体实施方式

[0078] 下面将结合附图对本发明技术方案的实施例进行详细的描述。以下实施例仅用于更加清楚地说明本发明的技术方案，因此只作为示例，而不能以此来限制本发明的保护范围。

[0079] 需要注意的是，除非另有说明，本申请使用的技术术语或者科学术语应当为本发明所属领域技术人员所理解的通常意义。

[0080] 实施例1

[0081] 一种B+树遍历的方法，对传统的遍历机制进行改进，主要包括

[0082] S1.在读请求过程中，给某一节点加读锁时，给其父节点解锁；

[0083] S2.在写请求过程中，给某一节点加写锁时，给其父节点解锁。

[0084] 为了提高bcache的IO并行处理能力，本发明方法中对B+树的遍历机制进行了优化，降低读写锁的粒度，因为只有叶子节点中存的是真正的缓存数据的信息，非叶子节点的锁没有必要一直持有，用完锁可以立即归还。因此在读请求时，以三层B+树为例，如图4-1、4-2所示，假设需要访问节点4201中的一个bkey，则步骤S1.包括：

[0085] S11.根节点4001加读锁；

[0086] S12.遍历根节点4001的bkey，找到符合条件的节点4101；

[0087] S13.给节点4101加读锁，并给根节点4001解锁；

[0088] S14.遍历节点4101的bkey，找到符合条件的叶子节点4201；

[0089] S15.给叶子节点4201加读锁，并给节点4101解锁

[0090] S16.在叶子节点4201中查找匹配的数据；

[0091] S17.给叶子节点4201解锁；

[0092] S18.结束。

[0093] 与传统的遍历方法相比，本发明中的便利方法在非叶子节点(4001、4101)的持锁时间明显变短。

[0094] 若在图4-1中的节点4201中未查找到期望的数据，也就是SSD盘上没有保存这份数据，那么需要从HDD盘将未命中的数据读上来，写到SSD盘，将对应的bkey插入B+树中，并将数据返回给上层调用者，下次再有同样IO请求的话，就能从SSD盘上直接获取返回给上层调

用者;因此本发明方法还包括B+树的插入流程,如图4-3所示,步骤S2.包括

[0095] S21.根节点4001加读锁;

[0096] S22.遍历根节点4001的bkey,找到符合条件的节点4101;

[0097] S23.给节点4101加读锁,并给根节点4001解锁;

[0098] S24.遍历节点4101的bkey,找到符合条件的叶子节点4201;

[0099] S25.给叶子节点4201加写锁,并给节点4101解锁

[0100] S26.在叶子节点4201中插入新增bkey;

[0101] S17.给叶子节点4201解锁;

[0102] S18.结束。

[0103] 若图4-3中描述的在节点4201中插入一个bkey的操作没有预期的顺序,节点4201中bkey已满,因为B+树节点的大小是有限的,在内存中仅256KB,这就涉及到了节点的分裂。如图4-4,4201这个节点分裂为42010和42011两个节点,这两个新增节点的bkey需要插入到4201的上级节点4101,如果4101节点大小也满,4101也需要分裂成2个节点,41010和41011两个节点,这两个新增节点的bkey要插入到4001中。因为无法预知本次分裂是否会导致父节点的继续分裂,因此bcache原有逻辑,分裂是一直持有根节点的写锁的,直到分裂结束。

[0104] 当B+树节点要分裂的时候,在分裂结束前需要一直持有根节点的写锁。大量的IO请求访问或修改B+树的时候,原逻辑会有很多根节点的读锁持有者,分裂的操作要拿到写锁必须等待读锁全部释放,如果根节点的读锁未及时归还,获取到根节点写锁的等待时间会加长,整个IO流程的延时必然会加大。但以本发明上述遍历方法,根节点的读锁持锁时间明显变短,获取写锁等待的时间相应的就缩短了。本发明通过上述方法中分区间加锁方式,进行B+树遍历过程,解决了传统技术中关键节点不合理的持锁方式造成的性能瓶颈;即,本发明方法中通过及时归还父节点的锁,解决了在IO并发量急剧增加的时候,由于非叶子节点持锁时间长导致的性能下降的问题;通过控制持锁区间,降低了由于不必要的持锁时间导致的开销。

[0105] 假设访问节点4201中的数据(如图4-4),经过本发明优化后的读流程加解锁顺序为:

[0106] 根节点4001加读锁→节点4101加读锁→根节点4001解读锁→节点4201加读锁→节点4101解读锁→节点4201解读锁;如图4-5所示。

[0107] 假设要往节点4201中插入一个bkey,导致分裂(图4-4),分裂流程的加解锁顺序为:

[0108] 根节点4001加写锁→节点4101加写锁→节点4201加写锁→节点4201解写锁→节点4101解写锁→节点4001解写锁;如图4-6所示。

[0109] 则可以得出,首先加锁解锁的数量是匹配的,不会由于遗漏解锁导致死锁。4.6.1的节点4001要拿到写锁,需要等4.5.3的节点4001解读锁,4.6.2的节点4101要拿到写锁,需要等4.5.5的节点4101解读锁完成,4.6.3的节点4201拿到写锁,需要4.5.6的解写锁。假设4.6.3拿到了节点4201的写锁,在4.6.4的解节点4201的写锁之前,4.5.4是无法拿到读锁的。拿锁顺序和归还锁的顺序也是匹配的;不会导致循环等待死锁;因此本发明B+遍历的方法不会引发死锁问题。

[0110] 实施例2

[0111] 本发明还提供了实施上述方法的装置，包括处理器和存储器，存储器中存储有程序，程序被处理器运行时，执行：

[0112] 在读请求过程中，给某一节点加读锁时，并给其父节点解锁；

[0113] 在写请求过程中，给某一节点加写锁时，并给其父节点解锁。

[0114] 本发明B+树遍历的装置对应实施例1中的方法步骤，则程序执行所述“在读请求过程中，给某一节点加读锁时，给其父节点解锁”中，包括

[0115] 根节点加读锁；

[0116] 遍历根节点的bkey，找到符合条件的非叶子节点；

[0117] 给任一非叶子节点加读锁时，并给该非叶子节点的父节点解锁；

[0118] 遍历非叶子节点的bkey，直至找到符合条件的叶子节点；

[0119] 给叶子节点加读锁时，并给该叶子节点的父节点解锁；

[0120] 在叶子节点中查找匹配的数据；

[0121] 给叶子节点解锁；

[0122] 结束。

[0123] 程序执行所述“在写请求过程中，给某一节点加读锁时，并给其父节点解锁”中，包括

[0124] 根节点加读锁；

[0125] 遍历根节点的bkey，找到符合条件的非叶子节点；

[0126] 给任一非叶子节点加读锁时，给该非叶子节点的父节点解锁；

[0127] 遍历非叶子节点的bkey，直至找到符合条件的叶子节点；

[0128] 给叶子节点加写锁时，给该叶子节点的父节点解锁；

[0129] 在叶子节点中插入新增bkey；

[0130] 给叶子节点解锁；

[0131] 结束。

[0132] 当所述在叶子节点中插入新增bkey时该节点中bkey已满，则进行节点分裂；任一节点分裂时，该节点下新增节点的bkey插入到该节点的上级节点中；且给任一节点加写锁时，并给上级节点解锁。

[0133] 如图5所示的，假设一个IO读请求，要访问节点5205里面的数据，一个IO写请求要在节点5203里插入一个bkey，此时节点5203的内容将满，增加的bkey会触发节点5203的分裂。当前读请求正在进行，如果按照传统的逻辑，读请求所在的线程持有的锁包括节点5001的读锁，节点5102的读锁，节点5205的读锁，分裂逻辑要获取到写锁，需要等到节点5205，节点5102，节点5001的读锁释放之后才可以获取。本发明装置实施过程中，读请求查找到节点5102后便可释放节点5001的锁，此时写请求可顺利拿到节点5001，节点5101，节点5203的写锁，减少了拿锁等待的时间。因B+树节点的内存是256K，bcache中IO请求压力比较大的时候，分裂比较频繁，分裂的时延降低将有助于提高bcache的IO并发处理能力；因为bcache是存在于内核态，对linux操作系统的并行处理能力也大大提升。

[0134] 最后应说明的是：以上各实施例仅用以说明本发明的技术方案，而非对其限制；尽管参照前述各实施例对本发明进行了详细的说明，本领域的普通技术人员应当理解：其依然可以对前述各实施例所记载的技术方案进行修改，或者对其中部分或者全部技术特征进

行等同替换;而这些修改或者替换,并不使相应技术方案的本质脱离本发明各实施例技术方案的范围,其均应涵盖在本发明的权利要求和说明书的范围当中。

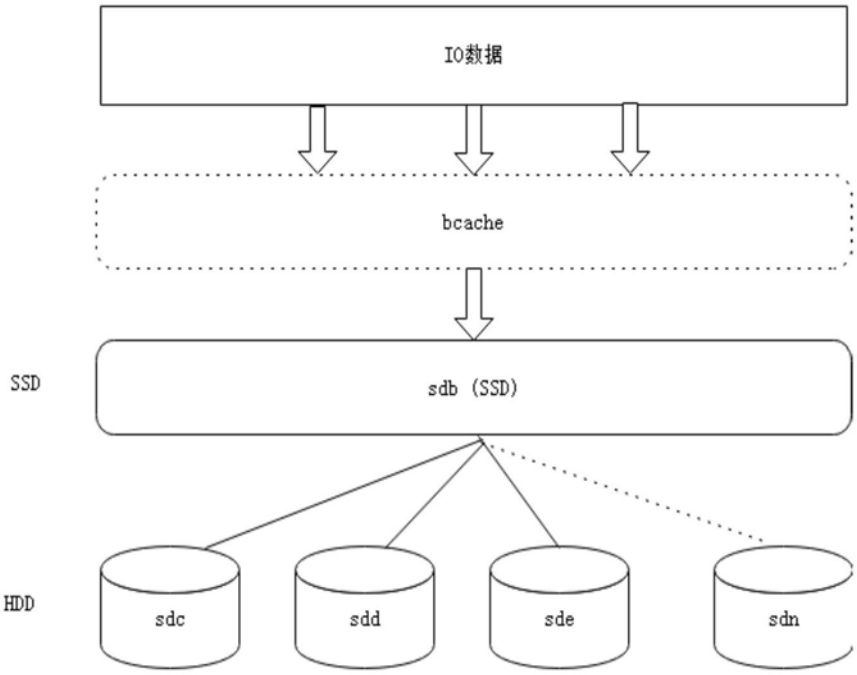

IO数据
bcache
SSD
sdb (SSD)
HDD
sdc
sdd
sde
sdn

图1

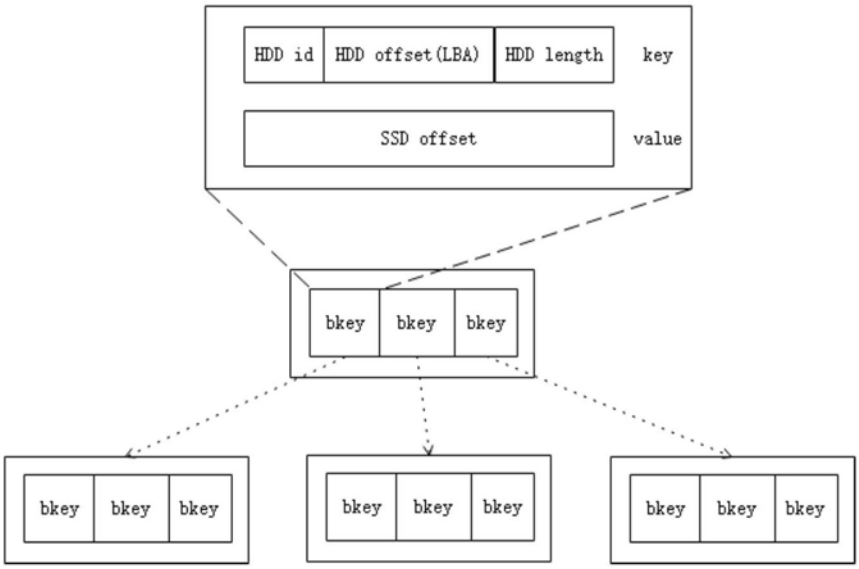
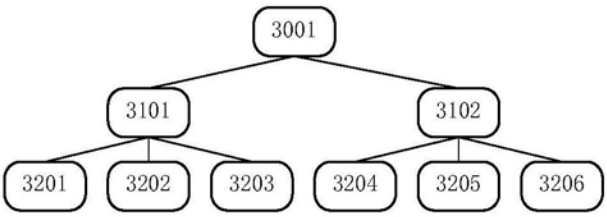
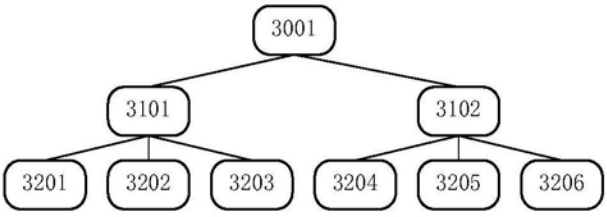

HDD id
HDD offset(LBA)
HDD length
key
SSD offset
value
bkey
bkey
bkey
bkey
bkey
bkey
bkey
bkey
bkey
bkey
bkey
bkey

图2

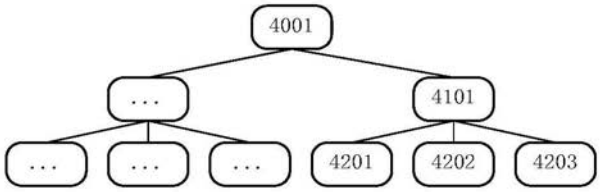

3001
3101
3102
3201
3202
3203
3204
3205
3206

（3-1）

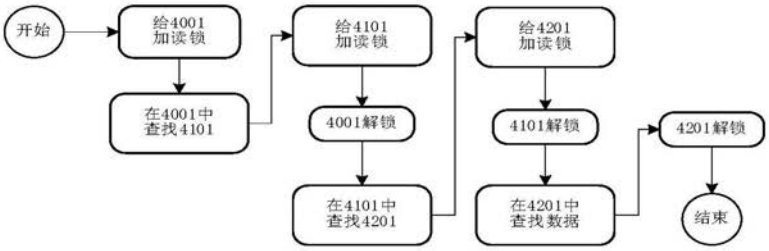

3001
3101
3102
3201
3202
3203
3204
3205
3206

（3-2）

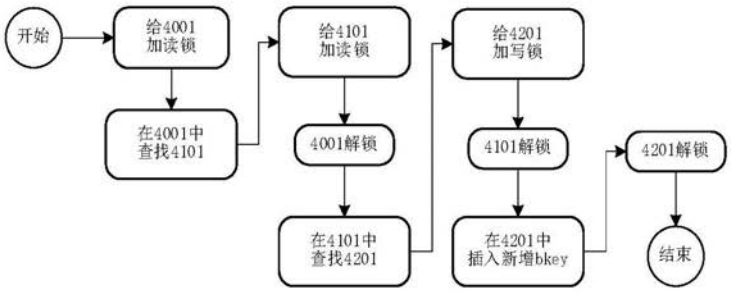
图3



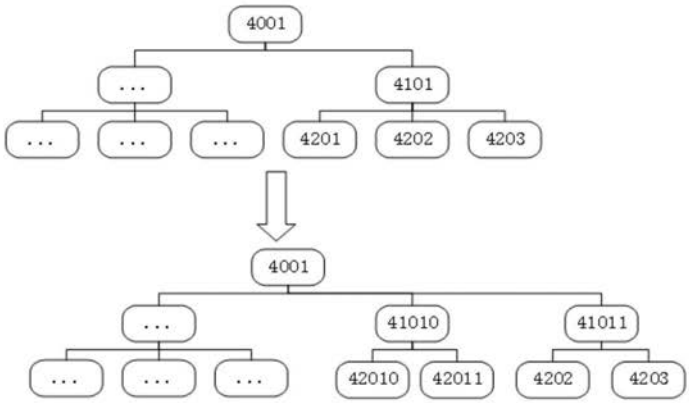
（4-1）



（4-2）



（4-3）



（4-4）

（4-5）

（4-6）

图4

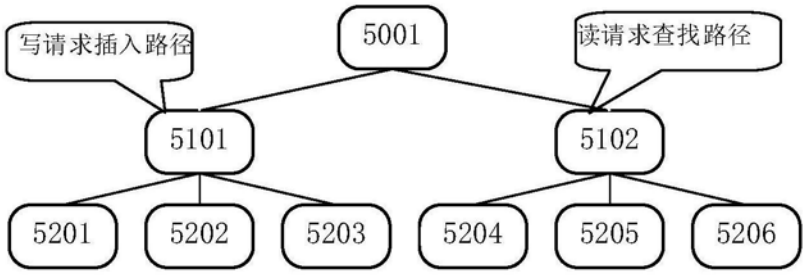

写请求插入路径
5001
读请求查找路径
5101
5102
5201
5202
5203
5204
5205
5206

图5