(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 116955348 A
(43)申请公布日 2023.10.27

(21)申请号 202210897014.2

(22)申请日 2022.07.28

(71)申请人 中移(上海)信息通信科技有限公司
地址 201206 上海市浦东新区新金桥路27号金桥现代产业服务园区10号楼二楼
申请人 上海交通大学
中国移动通信集团有限公司

(72)发明人 姚建国 周威 彭博 许寒旭 张萍 茬凯旋 李晗东

(74)专利代理机构 北京银龙知识产权代理有限公司 11243
专利代理师 吴立臣

(51)Int.Cl.
G06F 16/22(2019.01)
G06F 16/21(2019.01)

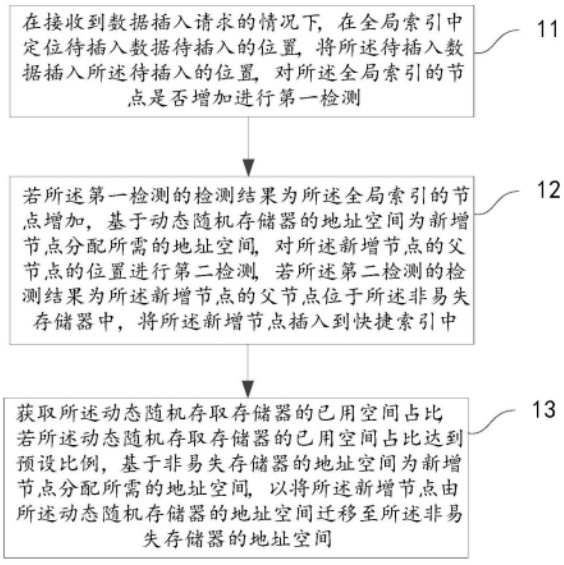
权利要求书3页 说明书12页 附图5页

(54)发明名称

一种数据库索引构建方法及装置

(57)摘要

本发明提供一种数据库索引构建方法及装置,属于信息存储技术领域,方法包括:在接收到数据插入请求的情况下,将待插入数据插入待插入的位置,若第一检测的检测结果为全局索引的节点增加,基于动态随机存储器的地址空间为新增节点分配所需的地址空间,若第二检测的检测结果为新增节点的父节点位于非易失存储器中,将新增节点插入到快捷索引中;若动态随机存取存储器的已用空间占比达到预设比例,将新增节点由动态随机存储器的地址空间迁移至非易失存储器的地址空间;其中,全局索引以及快捷索引均采用跳表数据结构。本发明采用跳表数据结构,优化了内存型数据库的索引结构,在保证高数据响应速度的前提下减少索引对DRAM的空间消耗。



CN 116955348 A

1.一种数据库索引构建方法,其特征在于,应用于包括动态随机存取存储器和非易失存储器的混合内存系统,所述方法包括:

在接收到数据插入请求的情况下,在全局索引中定位待插入数据待插入的位置,将所述待插入数据插入所述待插入的位置,对所述全局索引的节点是否增加进行第一检测;

若所述第一检测的检测结果为所述全局索引的节点增加,基于动态随机存储器的地址空间为新增节点分配所需的地址空间,对所述新增节点的父节点的位置进行第二检测,若所述第二检测的检测结果为所述新增节点的父节点位于所述非易失存储器中,将所述新增节点插入到快捷索引中;

获取所述动态随机存取存储器的已用空间占比,若所述动态随机存取存储器的已用空间占比达到预设比例,基于非易失存储器的地址空间为所述新增节点分配所需的地址空间,以将所述新增节点由所述动态随机存储器的地址空间迁移至所述非易失存储器的地址空间;

其中,所述全局索引以及所述快捷索引均采用跳表数据结构。

2.根据权利要求1所述的方法,其特征在于,所述在接收到数据插入请求的情况下,在全局索引中定位待插入数据待插入的位置包括:

根据起始节点的高度,设置当前节点高度,并以所述起始节点为初始的当前节点;

获取步骤:根据所述当前节点高度,从所述当前节点开始获取下一个节点作为新的当前节点;

若当前节点不为空指针且对应的键值小于所述待插入数据的搜索码,则将所述当前节点高度减去预设数值,执行所述获取步骤直至当前节点为空指针或对应的键值小于所述待插入数据的搜索码,对所述当前节点的父节点组数是否为空指针进行第一核验;

若所述第一核验的核验结果为所述当前节点的父节点组数不为空指针,对所述当前节点高度是否为零进行第二核验,若所述第二核验的核验结果为所述当前节点高度为零,返回所述当前节点并结束定位;若所述第二核验的核验结果为所述当前节点高度不为零,将所述当前节点高度减去预设数值,执行所述获取步骤直至所述当前节点高度为零。

3.根据权利要求2所述的方法,其特征在于,所述若所述第二检测的检测结果为所述新增节点的父节点位于所述非易失存储器中,将所述新增节点插入到快捷索引中包括:

将当前节点设置为跳表头,将当前节点高度设置为随机高度;

若当前节点高度超过所述非易失存储器允许的最大高度,得到所述当前节点高度超过所述非易失存储器允许的最大高度的超出部分,将所述超出部分对应的父节点数组索引的每一项设置为跳表头,将所述非易失存储器允许的最大高度设置为当前节点高度。

4.根据权利要求1所述的方法,其特征在于,基于非易失存储器的地址空间为所述新增节点分配所需的地址空间,以将所述新增节点由所述动态随机存储器的地址空间迁移至所述非易失存储器的地址空间包括:

若所述新增节点的父节点在所述动态随机存取存储器中具有后继节点,则将所述后继节点确定为目标节点,基于非易失存储器的地址空间为所述目标节点分配所需的地址空间,将所述目标节点由所述动态随机存储器的地址空间迁移至所述非易失存储器的地址空间。

5.根据权利要求4所述的方法,其特征在于,基于非易失存储器的地址空间为所述新增

节点分配所需的地址空间，以将所述新增节点由所述动态随机存储器的地址空间迁移至所述非易失存储器的地址空间包括：

递进步骤：逐一获取所述后继节点作为当前节点；

若当前节点的高度小于所述新增节点在所述非易失存储器中的父节点的高度，则将当前节点迁移到所述非易失存储器中，执行所述递进步骤直至当前节点的高度大于或等于所述非易失存储器中的父节点的高度。

6.一种数据库索引构建装置，其特征在于，应用于包括动态随机存取存储器和非易失存储器的混合内存系统，所述装置包括：

第一插入模块，用于在接收到数据插入请求的情况下，在全局索引中定位待插入数据待插入的位置，将所述待插入数据插入所述待插入的位置，对所述全局索引的节点是否增加进行第一检测；

第二插入模块，用于若所述第一检测的检测结果为所述全局索引的节点增加，基于动态随机存储器的地址空间为新增节点分配所需的地址空间，对所述新增节点的父节点的位置进行第二检测，若所述第二检测的检测结果为所述新增节点的父节点位于所述非易失存储器中，将所述新增节点插入到快捷索引中；

迁移模块，用于获取所述动态随机存取存储器的已用空间占比，若所述动态随机存取存储器的已用空间占比达到预设比例，基于非易失存储器的地址空间为新增节点分配所需的地址空间，以将所述新增节点由所述动态随机存储器的地址空间迁移至所述非易失存储器的地址空间；

其中，所述全局索引以及所述快捷索引均采用跳表数据结构。

7.根据权利要求6所述的装置，其特征在于，所述第一插入模块包括：

第一获取单元，用于根据起始节点的高度，设置当前节点高度，并以所述起始节点为初始的当前节点；

第二获取单元，用于获取步骤：根据所述当前节点高度，从所述当前节点开始获取下一个节点作为新的当前节点；

定位单元，用于若当前节点不为空指针且对应的键值小于所述待插入数据的搜索码，则将所述当前节点高度减去预设数值，执行所述获取步骤直至当前节点为空指针或对应的键值小于所述待插入数据的搜索码，对所述当前节点的父节点组数是否为空指针进行第一核验；

所述定位单元，还用于若所述第一核验的核验结果为所述当前节点的父节点组数不为空指针，对所述当前节点高度是否为零进行第二核验，若所述第二核验的核验结果为所述当前节点高度为零，返回所述当前节点并结束定位；若所述第二核验的核验结果为所述当前节点高度不为零，将所述当前节点高度减去预设数值，执行所述获取步骤直至所述当前节点高度为零。

8.根据权利要求7所述的装置，其特征在于，所述第一插入模块包括：

将当前节点设置为跳表头，将当前节点高度设置为随机高度；

若当前节点高度超过所述非易失存储器允许的最大高度，得到所述当前节点高度超过所述非易失存储器允许的最大高度的超出部分，将所述超出部分对应的父节点数组索引的每一项设置为跳表头，将所述非易失存储器允许的最大高度设置为当前节点高度。

9.根据权利要求6所述的装置,其特征在于,所述迁移模块包括:

迁移单元,用于若所述新增节点的父节点在所述动态随机存取存储器中具有后继节点,则将所述后继节点确定为目标节点,基于非易失存储器的地址空间为所述目标节点分配所需的地址空间,将所述目标节点由所述动态随机存储器的地址空间迁移至所述非易失存储器的地址空间。

10.根据权利要求9所述的装置,其特征在于,所述迁移模块包括:

第三获取单元,用于递进步骤:逐一获取所述后继节点作为当前节点;

停止单元,用于若当前节点的高度小于所述新增节点在所述非易失存储器中的父节点的高度,则将当前节点迁移到所述非易失存储器中,执行所述递进步骤直至当前节点的高度大于或等于所述非易失存储器中的父节点的高度。

一种数据库索引构建方法及装置

技术领域

[0001] 本发明涉及信息存储技术领域,尤其涉及一种数据库索引构建方法及装置。

背景技术

[0002] 随着云计算场景中数据规模的扩大,内存型数据库运行时所需承载的数据内存总量也不断增大。由于内存数据库的性能直接受限于其可使用的内存空间大小,当数据规模超过内存数据库可用的内存容量时,数据库管理者只能选择增大存储成本以购置额外的硬件设备来提高DRAM(Dynamic Random Access Memory,动态随机存取存储器)容量,或者牺牲性能将部分数据驱逐到磁盘中。

[0003] 为了减少因数据规模超过可用内存容量而导致的数据库系统性能退化的问题,许多工作围绕利用数据访问的局部性特征来对数据分区存储,将冷数据从DRAM移动到二级存储介质中。然而,尽管冷数据已经从DRAM中删除,但数据库系统还需要将所有的索引保留在内存中。而索引所占的DRAM空间消耗有时候会达到数据库总空间消耗的一半以上,故如何在保证高数据响应速度的前提下减少索引对DRAM的空间消耗是当下热门的研究问题。

发明内容

[0004] 有鉴于此,本发明提供一种数据库索引构建方法及装置,用于在保证数据的响应速度的前提下解决目前索引占用DRAM过多空间,导致DRAM资源浪费的问题。

[0005] 为解决上述技术问题,第一方面,本发明提供一种数据库索引构建方法,应用于包括动态随机存取存储器和非易失存储器的混合内存系统,该方法包括:

[0006] 在接收到数据插入请求的情况下,在全局索引中定位待插入数据待插入的位置,将所述待插入数据插入所述待插入的位置,对所述全局索引的节点是否增加进行第一检测;

[0007] 若所述第一检测的检测结果为所述全局索引的节点增加,基于动态随机存储器的地址空间为新增节点分配所需的地址空间,对所述新增节点的父节点的位置进行第二检测,若所述第二检测的检测结果为所述新增节点的父节点位于所述非易失存储器中,将所述新增节点插入到快捷索引中;

[0008] 获取所述动态随机存取存储器的已用空间占比,若所述动态随机存取存储器的已用空间占比达到预设比例,基于非易失存储器的地址空间为新增节点分配所需的地址空间,以将所述新增节点由所述动态随机存储器的地址空间迁移至所述非易失存储器的地址空间;

[0009] 其中,所述全局索引以及所述快捷索引均采用跳表数据结构。

[0010] 可选地,所述在接收到数据插入请求的情况下,在全局索引中定位待插入数据待插入的位置包括:

[0011] 根据起始节点的高度,设置当前节点高度,并以所述起始节点为初始的当前节点;

[0012] 获取步骤:根据所述当前节点高度,从所述当前节点开始获取下一个节点作为新

的当前节点；

[0013] 若当前节点不为空指针且对应的键值小于所述待插入数据的搜索码，则将所述当前节点高度减去预设数值，执行所述获取步骤直至当前节点为空指针或对应的键值小于所述待插入数据的搜索码，对所述当前节点的父节点组数是否为空指针进行第一核验；

[0014] 若所述第一核验的核验结果为所述当前节点的父节点组数不为空指针，对所述当前节点高度是否为零进行第二核验，若所述第二核验的核验结果为所述当前节点高度为零，返回所述当前节点并结束定位；若所述第二核验的核验结果为所述当前节点高度不为零，将所述当前节点高度减去预设数值，执行所述获取步骤直至所述当前节点高度为零。

[0015] 可选地，所述若所述第二检测的检测结果为所述新增节点的父节点位于所述非易失存储器中，将所述新增节点插入到快捷索引中包括：

[0016] 将当前节点设置为跳表头，将当前节点高度设置为随机高度；

[0017] 若当前节点高度超过所述非易失存储器允许的最大高度，得到所述当前节点高度超过所述非易失存储器允许的最大高度的超出部分，将所述超出部分对应的父节点数组索引的每一项设置为跳表头，将所述非易失存储器允许的最大高度设置为当前节点高度。

[0018] 可选地，基于非易失存储器的地址空间为所述新增节点分配所需的地址空间，以将所述新增节点由所述动态随机存储器的地址空间迁移至所述非易失存储器的地址空间包括：

[0019] 若所述新增节点的父节点在所述动态随机存取存储器中具有后继节点，则将所述后继节点确定为目标节点，基于非易失存储器的地址空间为所述目标节点分配所需的地址空间，将所述目标节点由所述动态随机存储器的地址空间迁移至所述非易失存储器的地址空间。

[0020] 可选地，基于非易失存储器的地址空间为所述新增节点分配所需的地址空间，以将所述新增节点由所述动态随机存储器的地址空间迁移至所述非易失存储器的地址空间包括：

[0021] 递进步骤：逐一获取所述后继节点作为当前节点；

[0022] 若当前节点的高度小于所述新增节点在所述非易失存储器中的父节点的高度，则将当前节点迁移到所述非易失存储器中，执行所述递进步骤直至当前节点的高度大于或等于所述非易失存储器中的父节点的高度。

[0023] 第二方面，本发明提供一种数据库索引构建装置，应用于包括动态随机存取存储器和非易失存储器的混合内存系统，所述装置包括：

[0024] 第一插入模块，用于在接收到数据插入请求的情况下，在全局索引中定位待插入数据待插入的位置，将所述待插入数据插入所述待插入的位置，对所述全局索引的节点是否增加进行第一检测；

[0025] 第二插入模块，用于若所述第一检测的检测结果为所述全局索引的节点增加，基于动态随机存储器的地址空间为新增节点分配所需的地址空间，对所述新增节点的父节点的位置进行第二检测，若所述第二检测的检测结果为所述新增节点的父节点位于所述非易失存储器中，将所述新增节点插入到快捷索引中；

[0026] 迁移模块，用于获取所述动态随机存取存储器的已用空间占比，若所述动态随机存取存储器的已用空间占比达到预设比例，基于非易失存储器的地址空间为新增节点分配

所需的地址空间，以将所述新增节点由所述动态随机存储器的地址空间迁移至所述非易失存储器的地址空间；

[0027] 其中，所述全局索引以及所述快捷索引均采用跳表数据结构。

[0028] 可选地，所述第一插入模块包括：

[0029] 第一获取单元，用于根据起始节点的高度，设置当前节点高度，并以所述起始节点为初始的当前节点；

[0030] 第二获取单元，用于获取步骤：根据所述当前节点高度，从所述当前节点开始获取下一个节点作为新的当前节点；

[0031] 定位单元，用于若当前节点不为空指针且对应的键值小于所述待插入数据的搜索码，则将所述当前节点高度减去预设数值，执行所述获取步骤直至当前节点为空指针或对应的键值小于所述待插入数据的搜索码，对所述当前节点的父节点组数是否为空指针进行第一核验；

[0032] 所述定位单元，还用于若所述第一核验的核验结果为所述当前节点的父节点组数不为空指针，对所述当前节点高度是否为零进行第二核验，若所述第二核验的核验结果为所述当前节点高度为零，返回所述当前节点并结束定位；若所述第二核验的核验结果为所述当前节点高度不为零，将所述当前节点高度减去预设数值，执行所述获取步骤直至所述当前节点高度为零。

[0033] 可选地，所述第一插入模块包括：

[0034] 将当前节点设置为跳表头，将当前节点高度设置为随机高度；

[0035] 若当前节点高度超过所述非易失存储器允许的最大高度，得到所述当前节点高度超过所述非易失存储器允许的最大高度的超出部分，将所述超出部分对应的父节点数组索引的每一项设置为跳表头，将所述非易失存储器允许的最大高度设置为当前节点高度。

[0036] 可选地，所述迁移模块包括：

[0037] 迁移单元，用于若所述新增节点的父节点在所述动态随机存取存储器中具有后继节点，则将所述后继节点确定为目标节点，基于非易失存储器的地址空间为所述目标节点分配所需的地址空间，将所述目标节点由所述动态随机存储器的地址空间迁移至所述非易失存储器的地址空间。

[0038] 可选地，所述迁移模块包括：

[0039] 第三获取单元，用于递进步骤：逐一获取所述后继节点作为当前节点；

[0040] 停止单元，用于若当前节点的高度小于所述新增节点在所述非易失存储器中的父节点的高度，则将当前节点迁移到所述非易失存储器中，执行所述递进步骤直至当前节点的高度大于或等于所述非易失存储器中的父节点的高度。

[0041] 第三方面，本发明还提供一种计算机可读存储介质，其上存储有计算机程序，该计算机程序被处理器执行时实现上述第一方面中任一种数据库索引构建方法中的步骤。

[0042] 本发明的上述技术方案的有益效果如下：

[0043] 本发明实施例中，跳表数据结构的节点由多个指针域构成，在插入过程中不存在分裂或合并操作，仅涉及指针域的修改，同时其节点内部不存在空余空间，由此，本发明实施例通过采用跳表数据结构优化了包括动态随机存取存储器和非易失存储器的混合内存系统的索引结构，保证了高数据响应速度；当存储于动态随机存储器的地址空间的新增节

点的父节点位于非易失存储器中，将新增节点插入到快捷索引中，由于快捷索引结构中的跳表可以使用节点的搜索码作为当前节点的描述信息，表示当前节点的子节点均比当前节点大，由此，将新增节点插入到快捷索引中加快了对新增数据的响应速度；当动态随机存取存储器的已用空间占比达到预设比例，基于非易失存储器的地址空间为新增节点分配所需的地址空间，将新增节点由动态随机存储器的地址空间迁移至非易失存储器的地址空间，本发明实施例在保证高数据响应速度的前提下减少索引对DRAM的空间消耗。

附图说明

[0044] 图1为本发明实施例一提供的一种数据库索引构建方法的流程示意图；

[0045] 图2为本发明实施例一提供的一种跳表结构的示意图；

[0046] 图3为本发明实施例一提供的实验结果的示意图之一；

[0047] 图4为本发明实施例一提供的实验结果的示意图之二；

[0048] 图5为本发明实施例一提供的实验结果的示意图之三；

[0049] 图6为本发明实施例一提供的实验结果的示意图之四；

[0050] 图7为本发明实施例一提供的实验结果的示意图之五；

[0051] 图8为本发明实施例一提供的实验结果的示意图之六；

[0052] 图9为本发明实施例一提供的实验结果的示意图之七；

[0053] 图10为本发明实施例一提供的实验结果的示意图之八；

[0054] 图11是本发明实施例二提供的一种数据库索引构建装置的结构示意图。

具体实施方式

[0055] 为使本发明实施例的目的、技术方案和优点更加清楚，下面将结合本发明实施例的附图，对本发明实施例的技术方案进行清楚、完整地描述。显然，所描述的实施例是本发明的一部分实施例，而不是全部的实施例。基于所描述的本发明的实施例，本领域普通技术人员所获得的所有其他实施例，都属于本发明保护的范围。

[0056] 下面对相关技术进行说明。

[0057] 随着云计算场景中数据规模的扩大，内存型数据库运行时所需承载的数据内存总量也不断增大。由于内存数据库的性能直接受限于其可使用的内存空间大小，当数据规模超过内存数据库可用的内存容量时，数据库管理者只能选择增大存储成本以购置额外的硬件设备来提高DRAM(Dynamic Random Access Memory，动态随机存取存储器)容量，或者牺牲性能将部分数据驱逐到磁盘中。

[0058] 为了减少因数据规模超过可用内存容量而导致的数据库系统性能退化的问题，许多工作围绕利用数据访问的局部性特征来对数据分区存储，将冷数据从DRAM移动到二级存储介质中。例如，H-Store的反缓存(Anti-caching)结构体系利用LRU(Least Recently Used，最近少使用)机制来追踪数据的访问模式，异步地将冷数据驱逐到磁盘的反缓冲区中；Volt DB(VoltDB，一个内存数据库，提供了NoSQL数据库的可伸缩性和传统关系数据库系统的ACID一致性)利用操作系统虚拟内存机制将冷页迁移到磁盘中；HANA(High-Performance Analytic Appliance，高性能分析应用软件)系统中，所有的新数据都会被插入到针对OLTP(On-Line Transaction Processing，联机事务处理过程)工作负载特征设计

的行存储引擎中，随着数据的老化，逐步被移动到优化的字典压缩的列式存储中。Hyper对数据的冷热程度进行了更进一步的划分，根据数据的访问频次分为热数据、温数据、冷数据以及不变的冷冻数据，热数据被放入能够快速定位查询的小页中，而冷冻数据则进行压缩并移动到二级存储的大页中，以实现高效、内存消耗友好的快照。

[0059] 然而，尽管冷数据已经从DRAM中删除，但数据库系统还需要将所有的索引保留在内存中，而索引所占的DRAM空间消耗有时候会达到数据库总空间消耗的一半以上，故如何减少索引对DRAM的空间消耗也是当下热门的研究问题。

[0060] SILT存储系统是一个基于闪存的键值存储系统，通过建立不同数据结构的多级存储层级索引结构来实现高性能和较小的内存占用。SILT的第一层是支持快速读写的Log Structured Structure(日志存储结构)；第二层使用传统的哈希表来进行缓冲，哈希表中并不存放完整的键，而是存储键的标志位来节约内存，当查询键时，首先检查标志位是否匹配，若匹配再从磁盘中获取键值进行比较；最后一层是前缀树，通过压缩算法，可以达到非常小的内存开销。一种非关系型的动态混合索引方法根据字段的冷热来进行动态索引划分，通过统计周期内的字段访问历史，更新非主键字段的权重，将字段划分为高频字段和低频字段，对高频字段建立详细索引，对低频字段建立摘要索引，减少对低频字段部分索引的空间开销。另有方法通过建立一个二阶段的索引结构来节省索引对内存空间的占用，第一阶段索引用于对新增数据进行索引，周期性触发迁移算法，将一阶段的数据移动到第二阶段的读优化、内存效率高的索引结构中。这些研究的不同之处在于如何确定需要迁移的内容以及迁移数据的机制。

[0061] 上述技术方案中，存在以下局限性：

[0062] 一、尽管冷数据已经从DRAM中删除，但数据库系统还需要将所有的索引保留在内存中。而索引所占的DRAM空间消耗有时候会达到数据库总空间消耗的一半以上，仍会造成大量的DRAM资源的浪费。

[0063] 二、对索引结构的分区存储主要指的是对位于不同存储介质上的数据建立不同的索引，导致了管理索引结构的成本增大，数据的迁移需要对多个索引进行维护，同时缺少扩展性，不利于将该模式应用到多种数据库索引结构上。

[0064] 三、现有技术中少有根据数据的冷热性来对索引结构进行分区域存储。因为传统的SSD(Solid State Disk或Solid State Drive，固态硬盘)等二级存储介质，其读写延迟和DRAM相比差距过大，如果将索引部分放入SSD中，势必会造成严重的性能损失。

[0065] 非易失性内存(Non-volatile Memory，NVM)新硬件的产生，为索引的分区存储方案提供了可能。

[0066] 由此，请参阅图1，图1为本发明实施例一提供的一种数据库索引构建方法的流程示意图，该方法应用于包括动态随机存取存储器和非易失存储器的混合内存系统，包括以下步骤：

[0067] 步骤11：在接收到数据插入请求的情况下，在全局索引中定位待插入数据待插入的位置，将所述待插入数据插入所述待插入的位置，对所述全局索引的节点是否增加进行第一检测；

[0068] 步骤12：若所述第一检测的检测结果为所述全局索引的节点增加，基于动态随机存储器的地址空间为新增节点分配所需的地址空间，对所述新增节点的父节点的位置进行

第二检测，若所述第二检测的检测结果为所述新增节点的父节点位于所述非易失存储器中，将所述新增节点插入到快捷索引中；

[0069] 步骤13：获取所述动态随机存取存储器的已用空间占比，若所述动态随机存取存储器的已用空间占比达到预设比例，基于非易失存储器的地址空间为新增节点分配所需的地址空间，以将所述新增节点由所述动态随机存储器的地址空间迁移至所述非易失存储器的地址空间；

[0070] 其中，所述全局索引以及所述快捷索引均采用跳表数据结构。

[0071] 本发明实施例中，数据库索引构建方法包括两个阶段，第一阶段是接收数据插入的阶段，当接收到一个数据的插入请求时，可以从全局索引的根节点开始向下搜索插入的目标节点，当待插入数据的插入引发了全局索引的节点增加，则在DRAM中分配新增节点需要的地址空间；如果该新增节点的父节点位于NVM地址空间，则将新增节点插入到快捷索引中进行维护，以加快对新增数据的响应速度。第二阶段是迁移阶段，也即当DRAM空间占用达到一定比例后，将触发迁移线程将新增节点(即目标节点)迁移到NVM空间中，从而降低DRAM的空间使用。

[0072] 本实施例中，全局索引以及所述快捷索引均采用跳表数据结构，支持高效的插入、查询和删除操作。

[0073] 本实施例中，可选地，采用两个索引数据结构，即全局索引和快捷索引，其中，全局索引用于对数据库的全量数据进行维护，在NVM地址空间进行构建，而快捷索引用于新增数据和节点的维护，以便加快新增数据查询更新的响应速度。

[0074] 本实施例中，可选地，对于全局索引的DRAM节点和NVM节点，跳表数据结构的节点由多个指针域构成，在插入过程中不存在分裂或合并操作，仅涉及指针域的修改，同时其节点内部不存在空余空间。

[0075] 本实施例中，可选地，对于快捷索引，采用跳表数据结构的情况下，可以使用节点的搜索码作为当前节点的描述信息，可表示当前节点的子节点均比当前节点大。具体的，跳表在不同层级都维持了链表的有序性，后续节点的搜索码的值都比当前节点的搜索码的值大，因此可以使用节点的搜索码作为当前节点的描述信息，以表示当前节点的子节点均比当前节点大。

[0076] 本发明实施例中，通过采用跳表数据结构，优化了包括动态随机存取存储器和非易失存储器的混合内存系统的索引结构，当存储于动态随机存储器的地址空间的新增节点的父节点位于非易失存储器中，将新增节点插入到快捷索引中；当动态随机存取存储器的已用空间占比达到预设比例，基于非易失存储器的地址空间为新增节点分配所需的地址空间，将新增节点由动态随机存储器的地址空间迁移至非易失存储器的地址空间，本发明实施例在保证高数据响应速度的前提下减少索引对DRAM的空间消耗。

[0077] 下面举例说明上述数据库索引构建方法。

[0078] 本发明实施例中，采用的NVM_SkipList(非易失存储器跳表)中，节点Node包含成员key，value和height，分别代表该节点的搜索码，数据值以及节点高度，next数组表示不同层级的后继节点，getNext函数和setNext函数分别用于获取和设置某一高度的后继节点值。NVM_SKipList中具有全局索引根节点成员nvmRoot和快捷索引根节点成员dramRoot，另外以maxNVMHeight(非易失存储器最高高度)和maxDRAMHeight(非易失存储器最高高度)分

别记录当前两个索引(全局索引和快捷索引)的最高高度。

[0079] 请参考图2,图2为本发明实施例一提供的一种跳表结构的示意图。如图2所示,在图2的跳表中进行查询操作时,包括以下流程:

[0080] 步骤1.1:记当前节点为头节点head,当前层级level为跳表的最高高度(对应图中level=4);

[0081] 步骤1.2:查看当前节点第level层的后继节点,如果后继节点为空指针或后继节点的搜索码比目标搜索码大,则将level减去一,返回步骤1.2;如果后继节点的搜索码值小于目标搜索码,则将当前节点置为后继节点,返回步骤1.2;如果后继节点的搜索码等于目标搜索码,则直接返回结果。

[0082] 对跳表执行插入操作的流程和查询流程类似,在寻找插入位置时,记录父节点集合,以图2中的node5为例,node5的父节点集合为{node4,node2,head},然后为新节点赋予随机的高度值作为指针域个数,然后将对应父节点的后继节点指针修改为新增节点。

[0083] 其中一种可选的具体实施方式中,所述在接收到数据插入请求的情况下,在全局索引中定位待插入数据待插入的位置包括:

[0084] 根据起始节点的高度,设置当前节点高度,并以所述起始节点为初始的当前节点;

[0085] 获取步骤:根据所述当前节点高度,从所述当前节点开始获取下一个节点作为新的当前节点;

[0086] 若当前节点不为空指针且对应的键值小于所述待插入数据的搜索码,则将所述当前节点高度减去预设数值,执行所述获取步骤直至当前节点为空指针或对应的键值小于所述待插入数据的搜索码,对所述当前节点的父节点组数是否为空指针进行第一核验;

[0087] 若所述第一核验的核验结果为所述当前节点的父节点组数不为空指针,对所述当前节点高度是否为零进行第二核验,若所述第二核验的核验结果为所述当前节点高度为零,返回所述当前节点并结束定位;若所述第二核验的核验结果为所述当前节点高度不为零,将所述当前节点高度减去预设数值,执行所述获取步骤直至所述当前节点高度为零。

[0088] 具体来说,由于跳表的每一个节点只能存储一个数据,因此每次新增数据的插入都会触发节点的增加。在进行新节点的构建时,第一步是定位目标位置,可以采用findGreaterOrEqual方法,具体包括以下步骤:

[0089] 步骤2.1:根据起始节点的高度,将起始节点高度减一,设置为当前高度;

[0090] 步骤2.2:根据当前高度,从起始节点开始获取下一个节点;

[0091] 步骤2.3:如果下一个节点不为空指针且对应的键值小于待插入数据的搜索码,则跳转至步骤2.4,否则跳转至步骤2.5;

[0092] 步骤2.4:将x设置为下一个节点,并跳转至步骤2.2,其中,x为伪代码内的节点指针的赋值;

[0093] 步骤2.5:如果父节点数:不为空指针,则将父节点数组对应当前高度的下标的值设置为x;

[0094] 步骤2.6:若当前高度为0,则返回下一个节点,定位结束,否则当前高度减一,跳转至步骤2.2。

[0095] 本发明的另一些实施例中,所述若所述第二检测的检测结果为所述新增节点的父节点位于所述非易失存储器中,将所述新增节点插入到快捷索引中包括:

[0096] 将当前节点设置为跳表头,将当前节点高度设置为随机高度;

[0097] 若当前节点高度超过所述非易失存储器允许的最大高度,得到所述当前节点高度超过所述非易失存储器允许的最大高度的超出部分,将所述超出部分对应的父节点数组索引的每一项设置为跳表头,将所述非易失存储器允许的最大高度设置为当前节点高度。

[0098] 本实施例中,如前所述,跳表节点拥有多个父节点组成的父节点数组(父节点集合),则在数据库构建的第一阶段中,需要考虑如何从父节点集合中选择节点加入到快捷索引中。以图2中的节点node5为例,对于加入到快捷索引的父节点组合有三种选择,如果选择将节点node4加入到快捷索引中,则当节点node5被迁移到NVM,地址位置改变后,节点node2以及节点head无法修改对应的后继节点指针,程序会发生错误。因此,只能选择和节点高度同等级的父节点,即对于节点node5需要将节点head加入到快捷索引,对于节点node3需要将节点node2加入到快捷索引。

[0099] 具体来说,将新增节点插入到快捷索引中包括以下步骤:

[0100] 步骤3.1:将当前节点设置为跳表头;

[0101] 步骤3.2:将x设置为根据待插入数据的搜索码、前一个节点以及跳表头定位到的节点;

[0102] 步骤3.3:将当前高度设置为随机高度;

[0103] 步骤3.4:如果当前高度超过NVM允许的最大高度,将flag设置为true,否则设置为false;

[0104] 步骤3.5:如果当前高度超过NVM允许的最大高度,则将当前高度超过NVM允许最大高度的部分对应的父节点数组索引的每一项设置为跳表头,并将NVM允许的最大高度设置为当前高度;

[0105] 步骤3.6:如果flag为true或者父节点数组索引的第一项是跳表头或者父节点数组索引的最后一项是跳表头,则转入步骤3.7,否则转入步骤3.8;

[0106] 步骤3.7:将x设置为新的NVM节点,转入步骤3.9:

[0107] 步骤3.8:将x设置为新的节点,转入步骤3.9;

[0108] 步骤3.9:对于从0开始一直到当前高度的索引,将x的next值设置为父节点数组索引中对应项的next,再将父节点索引中对应的next设置为x;

[0109] 步骤3.10:如果flag为true或者父节点数组索引的第一项是跳表头或者父节点数组索引的最后一项是跳表头,将父节点设置为父节点数组索引的最后一项,并调用插入捷径函数插入父节点的搜索码。

[0110] 本发明的一些实施例中,基于非易失存储器的地址空间为所述新增节点分配所需的地址空间,以将所述新增节点由所述动态随机存储器的地址空间迁移至所述非易失存储器的地址空间包括:

[0111] 若所述新增节点的父节点在所述动态随机存取存储器中具有后继节点,则将所述后继节点确定为目标节点,基于非易失存储器的地址空间为所述目标节点分配所需的地址空间,将所述目标节点由所述动态随机存储器的地址空间迁移至所述非易失存储器的地址空间。

[0112] 本实施例中,数据库构建的第二阶段需要考虑何时停止对当前节点的迁移迭代。对于每个需要进行迁移的NVM父节点,首先判断其是否具有DRAM后继节点,若不存在,则可

以直接返回,即停止迁移,否则进行后继节点迁移。

[0113] 本发明的另一些实施例中,基于非易失存储器的地址空间为所述新增节点分配所需的地址空间,以将所述新增节点由所述动态随机存储器的地址空间迁移至所述非易失存储器的地址空间包括:

[0114] 递进步骤:逐一获取所述后继节点作为当前节点;

[0115] 若当前节点的高度小于所述新增节点在所述非易失存储器中的父节点的高度,则将当前节点迁移到所述非易失存储器中,执行所述递进步骤直至当前节点的高度大于或等于所述非易失存储器中的父节点的高度。

[0116] 示例性地,记height为NVM父节点的高度,对prev数组进行初始化。通过调用getNext(0)函数不断获取后继节点,若后继节点的高度超过了height值或者后继节点高度等于height且为NVM节点,则终止迭代。对于位于DRAM地址空间的后继节点,调用allocateNVMNode函数,将节点迁移到NVM地址空间,然后修改prev数组的值,继续进行节点的迭代迁移。

[0117] 本实施例中,可选地,跳表结构的迭代迁移包括以下步骤:

[0118] 步骤4.1:初始化当前节点高度,声明父节点集合数组;

[0119] 步骤4.2:判断当前节点是否存在DRAM后继节点,若存在,则转入步骤4.11,若不存在,则转入步骤4.3;

[0120] 步骤4.3:初始化父节点集合;

[0121] 步骤4.4:获取当前节点的后继节点,若为空,则将当前高度超过NVM允许最大高度的部分对应的父节点数组索引的每一项设置为跳表头,并将NVM允许的最大高度设置为当前高度,若不为空,则转入步骤4.5;

[0122] 步骤4.5:获取后继节点高度,若超过当前节点高度,则转入步骤4.6,若等于当前节点高度且后继节点为NVM节点,则转入步骤4.11;

[0123] 步骤4.6:判断后继节点是否存在DRAM后继节点,若存在,则转入步骤4.7,若不存在,则转入步骤4.11;

[0124] 步骤4.7:为后继节点插入快捷索引,并转入步骤4.11;

[0125] 步骤4.8:判断后继节点是否为NVM节点,若是NVM节点,则转入步骤4.9,若不是,则转入步骤4.10;

[0126] 步骤4.9:更新父节点集合,将后继节点设置为当前节点,重复执行步骤4.4;

[0127] 步骤4.10:将后继节点迁移到NVM地址空间,更新父节点集合,重复执行步骤4.4;

[0128] 步骤4.11:迭代迁移操作结束。

[0129] 请参考图3至图10,图3为本发明实施例一提供的实验结果的示意图之一,图4为本发明实施例一提供的实验结果的示意图之二,图5为本发明实施例一提供的实验结果的示意图之三,图6为本发明实施例一提供的实验结果的示意图之四,图7为本发明实施例一提供的实验结果的示意图之五,图8为本发明实施例一提供的实验结果的示意图之六,图9为本发明实施例一提供的实验结果的示意图之七,图10为本发明实施例一提供的实验结果的示意图之八。本发明实施例中,使用Yahoo!Cloud Serving Benchmark(雅虎云服务基础测试工具,简称YCSB)测试基准程序对本发明实施例中的索引结构进行性能评估。YCSB是用于对云服务进行基础测试的工具,提供了六种不同读写比例的OLTP工作负载。本实施例中选

取了读写均衡工作负载、只读工作负载以及读密集工作负载对索引结构进行测试。工作负载的数据初始化阶段作为只写工作负载进行测试与分析。对于每个工作负载，采用64位单调递增整数以及64位随机整数两种基本类型的搜索码进行实验，索引值为64位整数。工作负载的数据局部性遵循Latest分布模式，分布幅度为10％，即90％的访问集中在10％的数据上。其中，工作负载模型主要包括只写(Insert-only)，读写均衡(Balanced)，只读(Read-only)，以及读密集(Read-heavy)四种类型；搜索码类型包括64位单调递增整数(Mono)以及64位随机整数(Rand)。本实施例中，对比方案包括原始索引结构以及一种已公开的二阶段索引压缩方案，为方便区分，将原始索引结构方案、二阶段索引压缩方案以及本发明实施例中的方案分别记作SkipList，Hybrid以及NVM。

[0130] 如图3至图10所示，每张图中，性能指标为吞吐量，图的标题表示该测试工作负载的配置，横轴为线程个数(threands)，纵轴为吞吐量(throughput)，曲线①为原始索引结构方案，曲线②为二阶段索引压缩方案，曲线③为本发明实施例中的方案。通过吞吐量的测试，可以看出，当搜索码为自增整数时，本发明实施例中的方案能够在降低DRAM资源使用量的基础上获得和原始索引结构方案相当的吞吐量，和二阶段索引压缩方案相比，本发明实施例中的方案的吞吐量的表现更优。而当搜索码为随机整数时，本发明实施例中的方案会有一定的性能损失，其原因在于随机整数导致新增节点的位置不确定，会进行频繁的快捷索引插入，从而导致整体的性能降低。

[0131] 只写工作负载：二阶段索引压缩方案主要是需要定期进行节点的迁移，从实验结果上可以看出，本发明实施例中的方案与二阶段索引压缩方案对只写工作负载都有一定的性能损失，而在不同的工作负载场景下，本发明实施例中的方案都比二阶段索引压缩方案性能更优，这是因为二阶段索引压缩方案在迁移过程中会阻塞正常的数据库请求直到完成迁移，而本发明实施例中的方案在迁移过程中只会和涉及迁移节点的请求发生锁争用，性能影响比二阶段索引压缩方案小。

[0132] 读写均衡工作负载：当运行读写均衡工作负载时，在不同搜索码、不同索引结构设置下，本发明实施例中的方案均比二阶段索引压缩方案更优，其原因在于，二阶段索引压缩方案对于更新请求，会将新数据插入到动态索引结构中，之后再和静态结构进行融合，会导致较高的迁移频率，引起性能下降。与原始索引结构对比需要区分不同的搜索码，当搜索码为自增整数时，本发明实施例中的方案能够取得和原始相近的吞吐量，而当搜索码为随机整数时，性能会有所下降。

[0133] 只读工作负载：二阶段索引压缩方案的问题在于，单点查询请求可能会遍历两个索引结构，整体的查询性能会有所降低，同时由于全局索引的大部分节点位于NVM地址空间，NVM的读写延迟将会进一步带来性能损失。

[0134] 读密集工作负载：本发明实施例中的方案在各工作负载上都比二阶段索引压缩方案更优。

[0135] 为了更直观的对比本发明实施例中的方案比现有两种方案在存储成本降低的效果，下表1对各方案的内存消耗情况进行了一个汇总，表1为三种方案的存储成本对比，其中，在计算存储成本时，记每单位GB(Gigabyte，吉字节)的DRAM存储成本为1，而每单位GB的NVM存储成本为0.3。

[0136] 表1：数据库内存与存储成本对比

[0137]

	SkipList 方案		Hybrid 方案		NVM 方案		
	DRAM 成本	总成本	DRAM 成本	总成本	DRAM 成本	NVM 成本	总成本
NVM_SkipList Mono	3.85	3.85	3.85	3.85	0.60	3.25	1.58
NVM_SkipList Rand	3.85	3.85	3.85	3.85	2.35	1.2	2.71

[0138] 从上表1可以看出，对于六种不同配置的索引结构和工作负载，和原始索引结构相比，本发明实施例中的方案分别能够降低59％，29％的存储成本；与二阶段索引压缩方案相比，本发明实施例中的方案能够降低的存储成本比例分别为59％，29％。

[0139] 以上实验结果可以说明，本发明实施例中的方案中，基于DRAM-NVM混合内存和跳表结构的数据库索引构建方法可以比现有技术在混合内存系统支撑的数据库提供更高的索引性能、提高混合内存中高速DRAM资源的有效利用率、从总体设计上降低数据库系统的内存成本。

[0140] 总之，本发明实施例中，通过采用跳表数据结构，优化了内存型数据库的索引结构，将热数据和其索引放在读写速度更快的DRAM中，将访问频率小的冷数据及其索引放在NVM中，在维持数据库的高吞吐量的同时可容纳更多的数据，节省了大量DRAM空间，提高了索引结构的可拓展性，提升了数据库性能和资源利用率。

[0141] 请参阅图11，图11是本发明实施例二提供的一种数据库索引构建装置的结构示意图，应用于包括动态随机存取存储器和非易失存储器的混合内存系统，该装置110包括：

[0142] 第一插入模块111，用于在接收到数据插入请求的情况下，在全局索引中定位待插入数据待插入的位置，若所述待插入数据导致所述全局索引的节点增加，在所述动态随机存取存储器中分配新增节点所需的地址空间；

[0143] 第二插入模块112，用于若所述新增节点的父节点位于所述非易失存储器中，则将所述新增节点插入到快捷索引中；

[0144] 迁移模块113，用于在所述动态随机存取存储器的已用空间占比达到预设比例的情况下，将目标节点迁移到所述非易失存储器中；

[0145] 其中，所述全局索引以及所述快捷索引均采用跳表数据结构。

[0146] 本发明实施例中，通过采用跳表数据结构，优化了内存型数据库的索引结构，将热数据和其索引放在读写速度更快的DRAM中，将访问频率小的冷数据及其索引放在NVM中，在维持数据库的高吞吐量的同时可容纳更多的数据，节省了大量DRAM空间，提高了索引结构的可拓展性，提升了数据库性能和资源利用率。

[0147] 可选的，所述第一插入模块包括：

[0148] 第一获取单元，用于根据起始节点的高度，设置当前节点高度，并从所述起始节点开始获取下一个节点作为当前节点；

[0149] 第二获取单元，用于若当前节点不为空指针且对应的键值小于所述待插入数据的搜索码，则将当前节点高度减一并继续获取下一个节点作为当前节点，直至当前节点为空指针或对应的键值小于所述待插入数据的搜索码，且当前节点的父节点数组不为空指针；

[0150] 定位单元，用于若当前节点为空指针或对应的键值不小于所述待插入数据的搜索

码，且当前节点的父节点数组不为空指针，则在当前节点高度为零时返回当前节点并结束定位，在当前节点高度不为零时将当前节点高度减一，并继续获取下一个节点作为当前节点，直至当前节点高度为零。

[0151] 可选的，所述第二插入模块包括：

[0152] 第一设置单元，用于将当前节点设置为跳表头，将当前节点高度设置为随机高度；

[0153] 第二设置单元，用于若当前节点高度超过所述非易失存储器允许的最大高度，则将所述当前节点高度超过所述非易失存储器允许的最大高度的部分对应的父节点数组索引的每一项设置为跳表头，并将所述非易失存储器允许的最大高度设置为当前节点高度；

[0154] 插入单元，用于将所述新增节点的父节点设置为所述父节点数组索引的最后一项，并调用插入捷径函数在所述快捷索引中插入所述新增节点的父节点的搜索码。

[0155] 可选的，所述迁移模块包括：

[0156] 迁移单元，用于若所述新增节点的父节点在所述动态随机存取存储器中具有后继节点，则将所述后继节点确定为目标节点，并迁移到所述非易失存储器中。

[0157] 可选的，所述迁移模块包括：

[0158] 第三获取单元，用于逐一获取所述后继节点作为当前节点；

[0159] 停止单元，用于若当前节点的高度小于所述新增节点在所述非易失存储器中的父节点的高度，则将当前节点迁移到所述非易失存储器中，若当前节点的高度大于或等于所述非易失存储器中的父节点的高度，则停止获取所述后继节点并停止将当前节点迁移至所述非易失存储器中。

[0160] 本发明实施例是与上述方法实施例一对应的产品实施例，故在此不再赘述，详细请参阅上述实施例一。

[0161] 本发明实施例三提供一种计算机可读存储介质，其上存储有计算机程序，该计算机程序被处理器执行时实现上述实施例一中任一种数据库索引构建方法中的步骤。详细请参阅以上对应实施例中方法步骤的说明。

[0162] 上述计算机可读存储介质包括永久性和非永久性、可移动和非可移动媒体可以由任何方法或技术来实现信息存储。信息可以是计算机可读指令、数据结构、程序的模块或其他数据。计算机的存储介质的例子包括，但不限于相变内存(PRAM)、静态随机存取存储器(SRAM)、动态随机存取存储器(DRAM)、其他类型的随机存取存储器(RAM)、只读存储器(ROM)、电可擦除可编程只读存储器(EEPROM)、快闪记忆体或其他内存技术、只读光盘只读存储器(CD-ROM)、数字多功能光盘(DVD)或其他光学存储、磁盒式磁带，磁带磁磁盘存储或其他磁性存储设备或任何其他非传输介质，可用于存储可以被计算设备访问的信息。

[0163] 以上所述是本发明的优选实施方式，应当指出，对于本技术领域的普通技术人员来说，在不脱离本发明所述原理的前提下，还可以作出若干改进和润饰，这些改进和润饰也应视为本发明的保护范围。

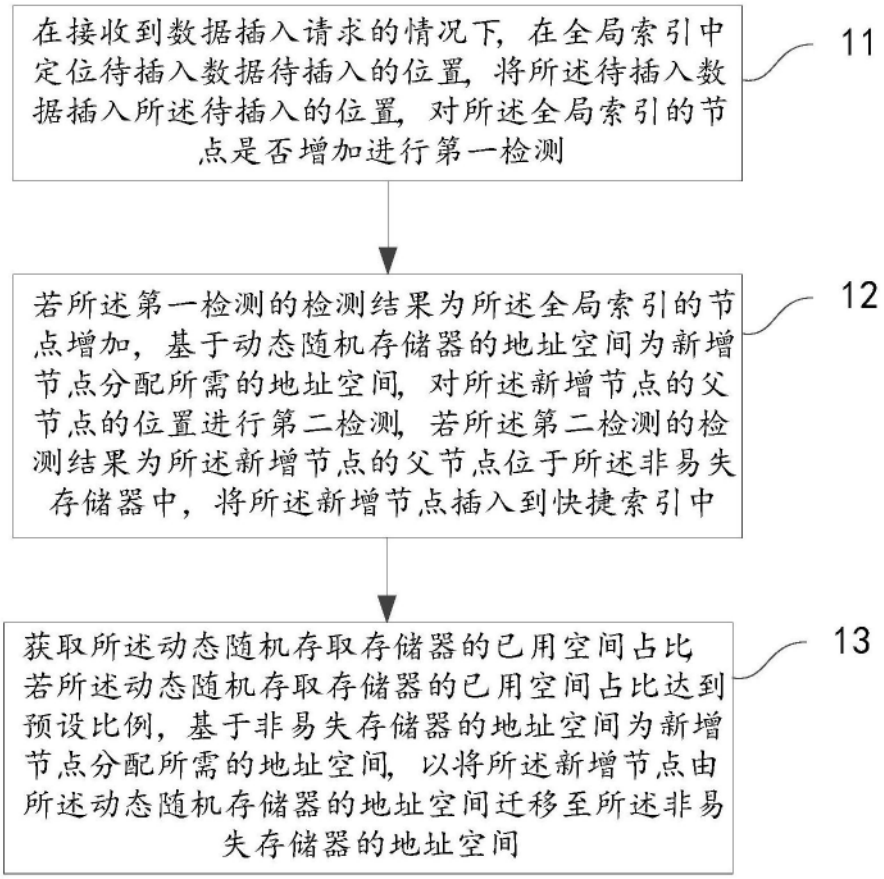

在接收到数据插入请求的情况下，在全局索引中定位待插入数据待插入的位置，将所述待插入数据插入所述待插入的位置，对所述全局索引的节点是否增加进行第一检测
11
若所述第一检测的检测结果为所述全局索引的节点增加，基于动态随机存储器的地址空间为新增节点分配所需的地址空间，对所述新增节点的父节点的位置进行第二检测，若所述第二检测的检测结果为所述新增节点的父节点位于所述非易失存储器中，将所述新增节点插入到快捷索引中
12
获取所述动态随机存取存储器的已用空间占比，若所述动态随机存取存储器的已用空间占比达到预设比例，基于非易失存储器的地址空间为新增节点分配所需的地址空间，以将所述新增节点由所述动态随机存储器的地址空间迁移至所述非易失存储器的地址空间
13

图1

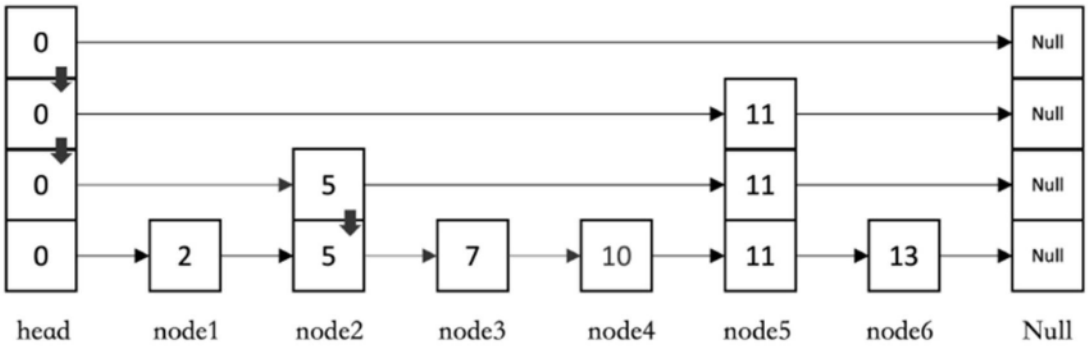

0
0
0
0
2
5
5
7
10
11
11
11
13
Null
Null
Null
Null
head
node1
node2
node3
node4
node5
node6
Null

图2

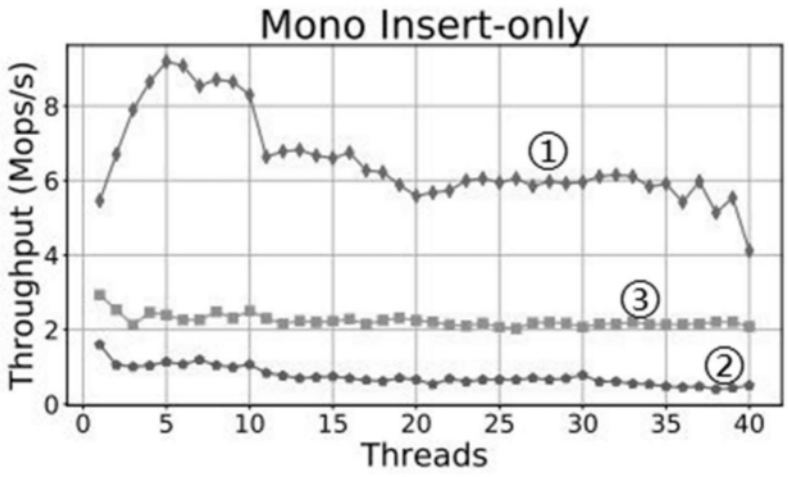

Mono Insert-only
Throughput (Mops/s)
Threads
①
②
③

图3

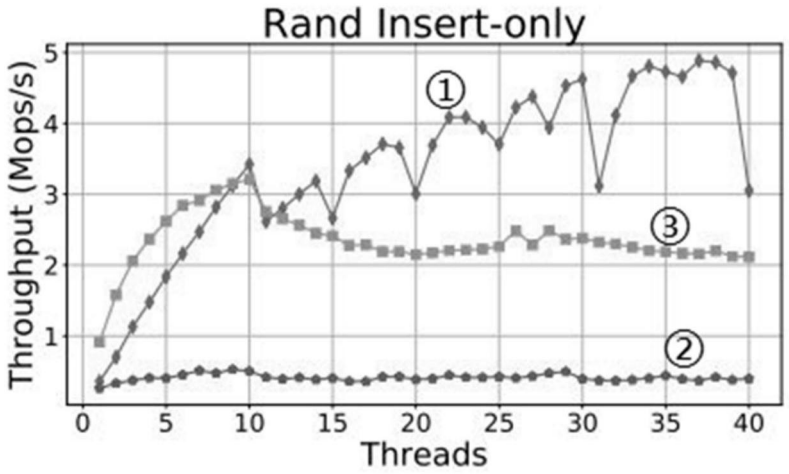

Rand Insert-only
Throughput (Mops/s)
Threads
①
②
③

图4

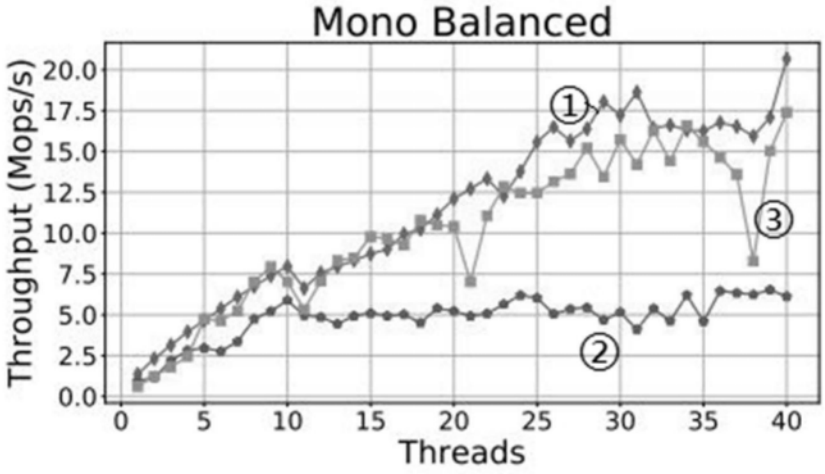

Mono Balanced
Throughput (Mops/s)
Threads
①
②
③

图5

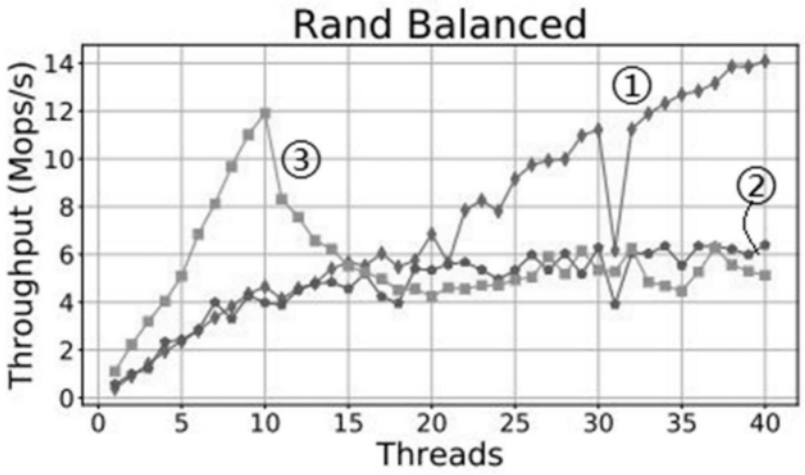

Rand Balanced
Throughput (Mops/s)
Threads
①
②
③

图6

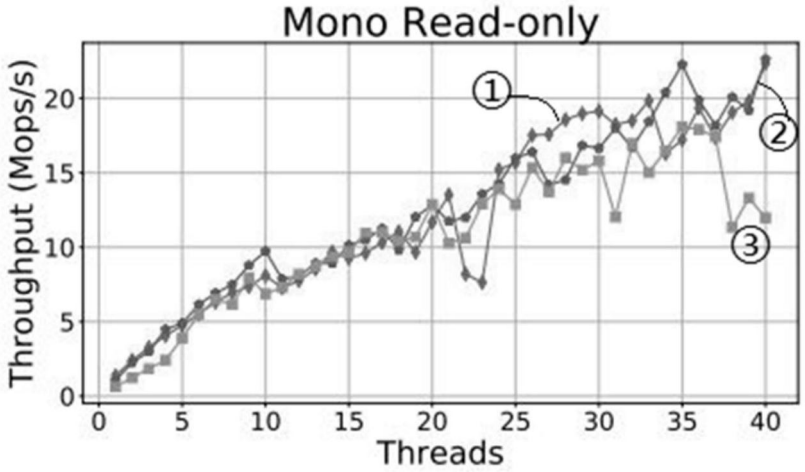

Mono Read-only
Throughput (Mops/s)
Threads
①
②
③

图7

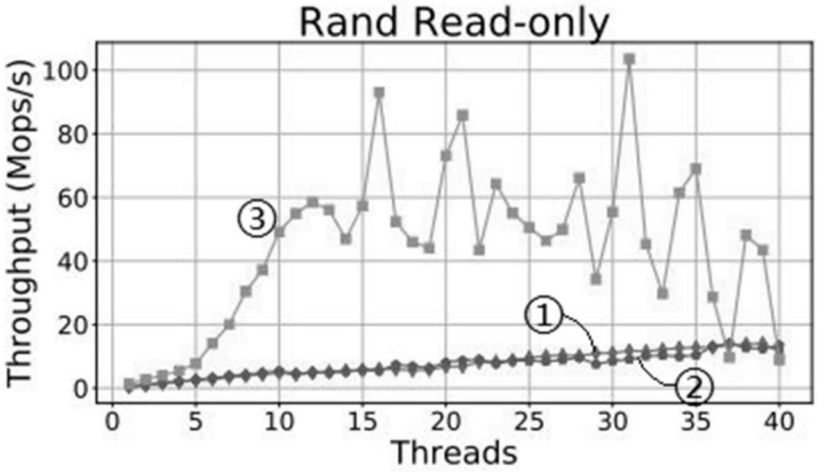

Rand Read-only
Throughput (Mops/s)
Threads
①
②
③

图8

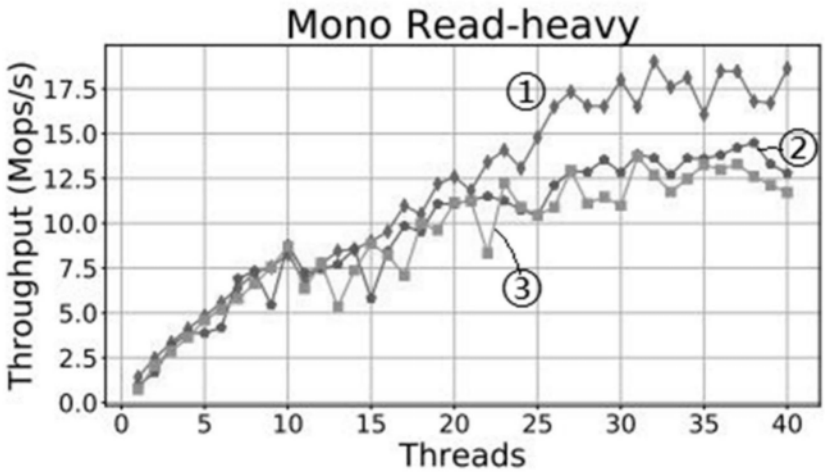

Mono Read-heavy
Throughput (Mops/s)
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0
5
10
15
20
25
30
35
40
Threads
①
②
③

图9

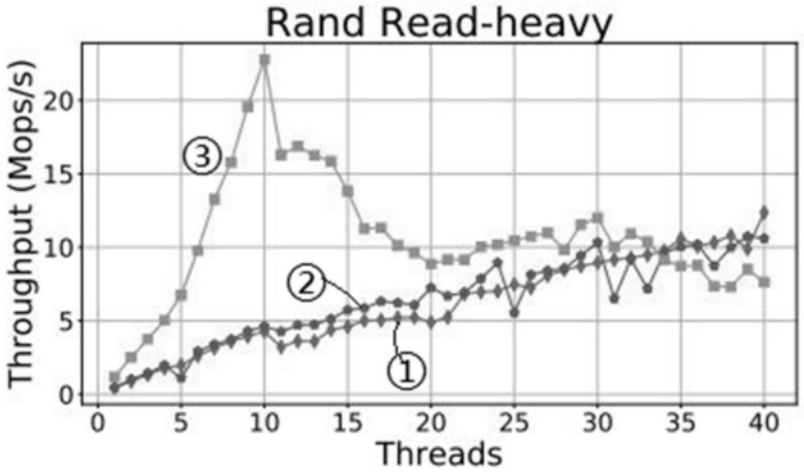

Rand Read-heavy
Throughput (Mops/s)
0
5
10
15
20
0
5
10
15
20
25
30
35
40
Threads
③
②
①

图10

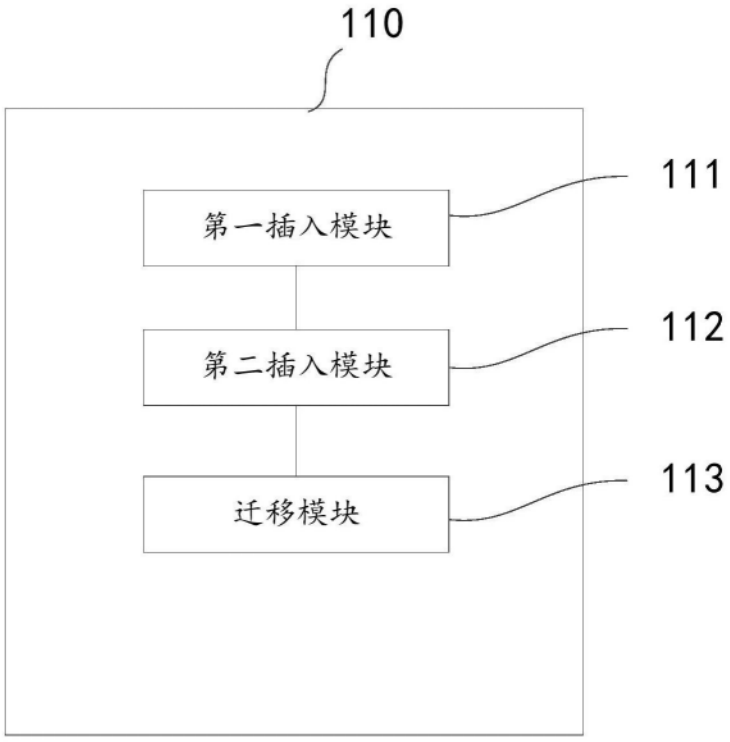


图11