(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 110162528 A
(43)申请公布日 2019.08.23

(21)申请号 201910441720.4

(22)申请日 2019.05.24

(71)申请人 安徽芃睿科技有限公司
地址 230000 安徽省合肥市庐阳区耀远路372号

(72)发明人 陆夏根 杨子江 于俊凤 李思思 徐蓓蓓 魏墨济 杨爱芹

(74)专利代理机构 济南圣达知识产权代理有限公司 37221
代理人 张庆骞

(51)Int.Cl.
G06F 16/22(2019.01)
G06F 16/2455(2019.01)
G06F 16/182(2019.01)

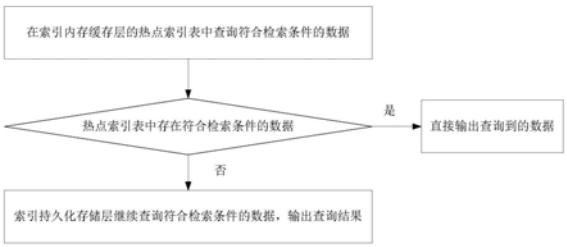
权利要求书2页 说明书8页 附图3页

(54)发明名称

海量大数据检索方法及系统

(57)摘要

本公开提供了一种海量大数据检索方法及系统。其中，一种海量大数据检索方法，包括在索引内存缓存层的热点索引表中查询符合检索条件的数据；所述热点索引表中存在有访问频度不小于预设访问频度阈值的数据；当热点索引表中存在符合检索条件的数据时，则直接输出查询到的数据；当热点索引表中不存在符合检索条件的数据时，则转到索引持久化存储层继续查询符合检索条件的数据，输出查询结果。



CN 110162528 A

1.一种海量大数据检索方法,其特征在于,包括:

在索引内存缓存层的热点索引表中查询符合检索条件的数据;所述热点索引表中存在有访问频度不小于预设访问频度阈值的数据;

当热点索引表中存在符合检索条件的数据时,则直接输出查询到的数据;

当热点索引表中不存在符合检索条件的数据时,则转到索引持久化存储层继续查询符合检索条件的数据,输出查询结果。

2.如权利要求1所述的一种海量大数据检索方法,其特征在于,所述索引持久化存储层中持久化存储有HBase,所述HBase中存储有多个非主键属性索引表,每个非主键属性索引表用来对应存储管理用户表中的一个待查询非主键属性的索引;

或所述索引内存缓存层为分布式内存缓存,使用一致性哈希来确定数据所在的服务器节点;在节点发生变化时,只有和变化节点相邻的节点数据需要迁移。

3.如权利要求1所述的一种海量大数据检索方法,其特征在于,在索引内存缓存层中执行查询请求时,内存缓存的服务进程对访问到的每条索引数据记录本周期内的访问次数,直到查询请求次数达到预设访问次数阈值,即到达热度计算周期时,服务进程触发缓存的更新替换。

4.如权利要求3所述的一种海量大数据检索方法,其特征在于,根据索引集合在相应周期内的被访问次数计算所有记录的热度,根据热度排序,将热度排序位于前K位的集合记录缓存到内存中;其中,K为热度门限值;根据缓存空间能够容纳的记录条数限制计算出热度门限K,高于热度门限的集合被缓存到内存中。

5.一种海量大数据检索系统,其特征在于,包括:

第一查询模块,其用于在索引内存缓存层的热点索引表中查询符合检索条件的数据;所述热点索引表中存在有访问频度不小于预设访问频度阈值的数据;

当热点索引表中存在符合检索条件的数据时,则直接输出查询到的数据;

第二查询模块,其用于当热点索引表中不存在符合检索条件的数据时,则转到索引持久化存储层继续查询符合检索条件的数据,输出查询结果。

6.如权利要求5所述的一种海量大数据检索系统,其特征在于,在所述第二查询模块中,所述索引持久化存储层中持久化存储有HBase,所述HBase中存储有多个非主键属性索引表,每个非主键属性索引表用来对应存储管理用户表中的一个待查询非主键属性的索引;

或在所述第二查询模块中,所述索引内存缓存层为分布式内存缓存,使用一致性哈希来确定数据所在的服务器节点;在节点发生变化时,只有和变化节点相邻的节点数据需要迁移。

7.如权利要求5所述的一种海量大数据检索系统,其特征在于,在所述第一查询模块中,在索引内存缓存层中执行查询请求时,内存缓存的服务进程对访问到的每条索引数据记录本周期内的访问次数,直到查询请求次数达到预设访问次数阈值,即到达热度计算周期时,服务进程触发缓存的更新替换。

8.如权利要求7所述的一种海量大数据检索系统,其特征在于,在所述第一查询模块中,根据索引集合在相应周期内的被访问次数计算所有记录的热度,根据热度排序,将热度排序位于前K位的集合记录缓存到内存中;其中,K为热度门限值;根据缓存空间能够容纳的

记录条数限制计算出热度门限K,高于热度门限的集合被缓存到内存中。

9.一种计算机可读存储介质,其上存储有计算机程序,其特征在于,该程序被处理器执行时实现如权利要求1-4中任一项所述的海量大数据检索方法中的步骤。

10.一种计算机设备,包括存储器、处理器及存储在存储器上并可在处理器上运行的计算机程序,其特征在于,所述处理器执行所述程序时实现如权利要求1-4中任一项所述的海量大数据检索方法中的步骤。

海量大数据检索方法及系统

技术领域

[0001] 本公开属于大数据处理领域，尤其涉及一种海量大数据检索方法及系统。

背景技术

[0002] 本部分的陈述仅仅是提供了与本公开相关的背景技术信息，不必然构成在先技术。

[0003] Hbase(Hadoop Database)是一个高可靠性、高性能、面向列、可伸缩的分布式存储系统，利用HBase技术可在廉价PC Server上搭建起大规模结构化存储集群。HBase提供两种常用编程访问方法：(1)JavaAPI，用户应用程序需要把HBase客户端函数库(jar包)拷贝到CLASSPATH指定的路径中，即可调用HBase的API进行数据读写。(2)Thrift方式。Thrift提供多种语言的接口库，包括C++，Java，Python，Perl，C#.用户程序根据需要调用这些接口。在HBase集群中部署Thrift服务器集群，Thrift服务器负责将请求转发给HBase集群。Thrift服务器可以和HBase集群共有相同的物理服务器。

[0004] 发明人发现，当面向海量大数据检索时，采用HBase非主键查询的过程中要对全表扫描，这样降低了索引表查询速度；另外，每次查询访问HBase表会涉及到很多磁盘访问，这样也影响了索引表查询速度。

发明内容

[0005] 为了解决上述问题，本公开的第一个方面提供一种海量大数据检索方法，其通过将索引热点数据缓存在内存中，部分查询可以直接在内存中命中结果集，从而降低了磁盘访问开销，提高整体查询性能。

[0006] 本公开的第一个方面的一种海量大数据检索方法的技术方案为：

[0007] 一种海量大数据检索方法，包括：

[0008] 在索引内存缓存层的热点索引表中查询符合检索条件的数据；所述热点索引表中存在有访问频度不小于预设访问频度阈值的数据；

[0009] 当热点索引表中存在符合检索条件的数据时，则直接输出查询到的数据；

[0010] 当热点索引表中不存在符合检索条件的数据时，则转到索引持久化存储层继续查询符合检索条件的数据，输出查询结果。

[0011] 为了解决上述问题，本公开的第二个方面提供一种海量大数据检索系统，其通过将索引热点数据缓存在内存中，部分查询可以直接在内存中命中结果集，从而降低了磁盘访问开销，提高整体查询性能。

[0012] 本公开的第二个方面的一种海量大数据检索系统的技术方案为：

[0013] 一种海量大数据检索系统，包括：

[0014] 第一查询模块，其用于在索引内存缓存层的热点索引表中查询符合检索条件的数据；所述热点索引表中存在有访问频度不小于预设访问频度阈值的数据；

[0015] 当热点索引表中存在符合检索条件的数据时，则直接输出查询到的数据；

[0016] 第二查询模块，其用于当热点索引表中不存在符合检索条件的数据时，则转到索引持久化存储层继续查询符合检索条件的数据，输出查询结果。

[0017] 为了解决上述问题，本公开的第三个方面提供一种计算机可读存储介质，其通过将索引热点数据缓存在内存中，部分查询可以直接在内存中命中结果集，从而降低了磁盘访问开销，提高整体查询性能。

[0018] 本公开的第三个方面的一种计算机可读存储介质的技术方案为：

[0019] 一种计算机可读存储介质，其上存储有计算机程序，该程序被处理器执行时实现上述所述的海量大数据检索方法中的步骤。

[0020] 为了解决上述问题，本公开的第四个方面提供一种计算机设备，其通过将索引热点数据缓存在内存中，部分查询可以直接在内存中命中结果集，从而降低了磁盘访问开销，提高整体查询性能。

[0021] 本公开的第四个方面的一种计算机设备的技术方案为：

[0022] 一种计算机设备，包括存储器、处理器及存储在存储器上并可在处理器上运行的计算机程序，所述处理器执行所述程序时实现上述所述的海量大数据检索方法中的步骤。

[0023] 本公开的有益效果是：

[0024] (1)本公开首先到索引内存缓存层查询热点索引数据，若缓存中不存在该记录，则将查询转发到索引持久化存储层进行检索，通过将索引热点数据缓存在内存中，部分查询可以直接在内存中命中结果集，从而降低了磁盘访问开销，提高整体查询性能，这对于具有倾斜的数据访问分布特性的应用来说尤为有效。

[0025] (2)本公开的索引持久化存储层中持久化存储有HBase，所述HBase中存储有多个非主键属性索引表，每个非主键属性索引表用来对应存储管理用户表中的一个待查询非主键属性的索引，这样借助HBase的特性获得良好的可扩展性和容错性，避免了对HBase非主键查询时的全表扫描，提供快速的非主键查询能力引。

[0026] (3)本公开在索引内存缓存层中执行查询请求时，内存缓存的服务进程对访问到的每条索引数据记录本周期内的访问次数，直到查询请求次数达到预设访问次数阈值，即到达热度计算周期时，服务进程触发缓存的更新替换，这样不仅考虑了数据的访问时间远近，同时考虑了数据的访问频率，能够提升5-15倍的查询性能。

附图说明

[0027] 构成本公开的一部分的说明书附图用来提供对本公开的进一步理解，本公开的示意性实施例及其说明用于解释本公开，并不构成对本公开的不当限定。

[0028] 图1是本公开实施例提供的HDFS框架；

[0029] 图2是本公开实施例提供的MapReduce程序的具体执行过程；

[0030] 图3是本公开实施例提供的一种海量大数据检索方法流程图；

[0031] 图4是本公开实施例提供的一种分层式索引存储模型；

[0032] 图5是本公开实施例提供的一种海量大数据检索系统结构示意图。

具体实施方式

[0033] 应该指出，以下详细说明都是例示性的，旨在对本公开提供进一步的说明。除非另

有指明，本文使用的所有技术和科学术语具有与本公开所属技术领域的普通技术人员通常理解的相同含义。

[0034] 需要注意的是，这里所使用的术语仅是为了描述具体实施方式，而非意图限制根据本公开的示例性实施方式。如在这里所使用的，除非上下文另外明确指出，否则单数形式也意图包括复数形式，此外，还应当理解的是，当在本说明书中使用术语“包含”和/或“包括”时，其指明存在特征、步骤、操作、器件、组件和/或它们的组合。

[0035] 术语解释部分：

[0036] Hbase(Hadoop Database)是一个高可靠性、高性能、面向列、可伸缩的分布式存储系统。HBase利用Hadoop HDFS作为其文件存储系统；利用Hadoop MapReduce来处理HBase中的海量数据。

[0037] HBase提供给了两种查询方法和多种组合查询条件：

[0038] (1)根据主键直接查找对应的记录。这是速度最快的查询操作。

[0039] (2)提供快速扫描的方法，可以让用户组合多种条件进行扫描，例如：

[0040] 1)指定行主键的特征，例如满足某个前缀、包含某个子字符串或者满足某个正则表达式；

[0041] 2)指定列前缀的特征，与行主键一样，可以描述多种条件；

[0042] 3)指定某列的值的特征，例如比较关系、字符串包含关系；

[0043] 4)指定时间戳范围，例如大于某个时间点的记录，或者最近3次的记录值。

[0044] 除此以外，针对一些特殊应用需要采用SQL查询，可以在现有集群基础上另外建设Hive集群。HBase集群作为Hive的数据源。Hive把类SQL的语句编译成Map/Reduce分布式程序进行并行查询，并把结果汇总。对于更复杂的应用，例如模式识别、流量预测等，需要建立对应的数据模型，编写Map/Reduce分布式程序，对HBase上的数据进行分析。

[0045] 其中，Hadoop实现了一个分布式文件系统(Hadoop Distributed File System)，简称HDFS。HDFS最开始是作为Apache Nutch搜索引擎项目的基础架构而开发的。

[0046] HDFS主要由Client、Datanode和Namenode组成，其框架如图1所示。一个使用Hadoop技术架构的集群中，一般有一到两台主机作为Namenode，若干台主机作为Datanode。Client代表使用HDFS的客户程序；Namenode是Hadoop集群中的一台主机，负责保存数据节点的信息、计算任务的分发以及最终规约等任务；Datanode负责数据存储与处理。为保证数据的安全性，HDFS适度增加了冗余数据。具体的做法是在不同的Datanode中保存同一数据的多份拷贝，一般为三份拷贝。

[0047] 一个客户端创建一个文件的请求并不会立即转发到Namenode。实际上，一开始HDFS客户端将文件数据缓存在本地的临时文件中。应用程序的写操作被透明地重定向到这个临时本地文件。当本地文件堆积到一个HDFS块大小的时候，客户端才会通知Namenode。Namenode将文件名插入到文件系统层次中，然后为它分配一个数据块。Namenode构造包括Datanode ID(可能是多个，副本数据块存放的节点也有)和目标数据块标识的报文，用它回复客户端的请求。客户端收到后将本地的临时文件刷新到指定的Datanode数据块中。

[0048] 当文件关闭时，本地临时文件中未残留的数据就会被转送到Datanode。然后客户端就可以通知Namenode文件已经关闭。此时，Namenode将文件的创建操作添加到到持久化存储中。假如Namenode在文件关闭之前死掉，文件就丢掉了。

[0049] 当客户端写数据到HDFS文件中时，如上所述，数据首先被写入本地文件中，假设HDFS文件的复制因子是3，当本地文件堆积到一块大小的数据，客户端从Namenode获得一个Datanode的列表。这个列表也包含存放数据块副本的Datanode。当客户端刷新数据块到第一个Datanode。第一个Datanode开始以4kb为单元接收数据，将每一小块都写到本地库中，同时将每一小块都传送到列表中的第二个Datanode。同理，第二个Datanode将小块数据写入本地库中同时传给第三个Datanode，第三个Datanode直接写到本地库中。一个Datanode在接前一个节点数据的同时，还可以将数据流水式传递给下一个节点，所以，数据是流水式地从一个Datanode传递到下一个。

[0050] 在数据处理过程中Hadoop采用MapReduce技术。MapReduce是一种编程模型，用于大规模数据集(大于1TB)的并行运算。概念“Map(映射)”和“Reduce(归约)”，和它们的主要思想，都是从函数式编程语言里借来的，还有从矢量编程语言里借来的特性。它极大地方便了编程人员在不会分布式并行编程的情况下，将自己的程序运行在分布式系统上。当前的软件实现是指定一个Map(映射)函数，用来把一组键值对映射成一组新的键值对，指定并发的Reduce(归约)函数，用来保证所有映射的键值对中的每一个共享相同的键组。

[0051] MapReduce程序的具体执行过程如图2所示：

[0052] 首先对数据源进行分块，然后交给多个Map任务去执行，Map任务执行Map函数，根据某种规则对数据分类，写入本地硬盘；Map阶段完成后，进入Reduce阶段，Reduce任务执行Reduce函数，具有同样Key值的中间结果，从多个Map任务所在的节点，被收集到一起(shuffle)进行合并处理，输出结果写入本地硬盘(分布式文件系统)。程序的最终结果可以通过合并所有Reduce任务的输出得到。

[0053] 实施例1

[0054] 如图3所示，本实施例提供的一种海量大数据检索方法，至少包括：

[0055] S101：在索引内存缓存层的热点索引表中查询符合检索条件的数据；所述热点索引表中存在有访问频度不小于预设访问频度阈值的数据；

[0056] 当热点索引表中存在符合检索条件的数据时，则直接输出查询到的数据。

[0057] 本实施例的海量大数据以科研项目数据为例：

[0058] 首先，在索引内存缓存层的热点索引表中查询符合检索条件的数据；其中，检索条件为热力科研项目且项目属于中外合作项目；

[0059] 所述热点索引表中存在有访问频度不小于预设访问频度阈值(例如：10次/天)的数据。

[0060] 具体地，访问频度可采用在一定周期内的访问次数来表示。

[0061] 当热点索引表中存在符合检索条件的数据时，则直接输出查询到的数据。

[0062] 在具体实施中，所述索引内存缓存层为分布式内存缓存，使用一致性哈希来确定数据所在的服务器节点；在节点发生变化时，只有和变化节点相邻的节点数据需要迁移。

[0063] 在分布式内存缓存中，使用一致性哈希来确定数据所在的服务器节点。在节点发生变化时(如节点失效或节点加入)，只有和变化节点相邻的节点数据需要迁移，从而可减少节点的加入和退出带来的计算和数据传输开销。

[0064] S102：当热点索引表中不存在符合检索条件的数据时，则转到索引持久化存储层继续查询符合检索条件的数据，输出查询结果。

[0065] 在具体实施中，索引表将为HBase表实现索引数据的持久化存储。由于索引数据是存放在HBase中，每次查询访问HBase表会涉及到很多磁盘访问，进一步考虑把索引中那些访问频度高的索引数据作为热点数据缓存在内存中，形成基于HBase和分布式内存的分层式索引存储和查询机制，进一步提高索引表查询速度。分层式索引存储模型，如图4所示，分层式索引存储模型基本的查询过程是：首先到索引内存缓存层查询热点索引数据，若缓存中不存在该记录，则将查询转发到索引持久化存储层进行检索。可以看出，通过将索引热点数据缓存在内存中，部分查询可以直接在内存中命中结果集，从而降低了磁盘访问开销，提高整体查询性能，这对于具有倾斜的数据访问分布特性的应用来说尤为有效。

[0066] 具体地，所述索引持久化存储层中持久化存储有HBase，所述HBase中存储有多个非主键属性索引表，每个非主键属性索引表用来对应存储管理用户表中的一个待查询非主键属性的索引。

[0067] 在另一实施例中，通常缓存的容量远远小于保存全部数据的磁盘数据库的容量，所以当缓存满了之后需要选择合适的牺牲者淘汰出缓存，这就是缓存替换策略。本实施例采用热度累积的缓存替换策略，其基本设计思想是周期性地累积记录被访问的次数。

[0068] 在内存中缓存的索引热点数据基于Redis的集合(Set)存储，Redis Set也是以<key,value>格式来组织数据。索引热点数据的索引主键做Redis Set的key，而索引集合作为Redis Set的value保存在内存缓存中。显然，具有相同索引列值的记录被绑定在同一个集合中，基于索引列值的查询命中是以集合为单位的。同时，它们也是热度累积的基本单位，每个集合都会累积它在一个计算周期内的访问次数。热度累积的缓存替换策略基于与LRU算法相同的假设：最近被访问的数据在最近的未来最有可能被重复访问。算法周期性地计算集合的累积热度，对所有的记录累积热度排序，选择累积热度TOP-K的索引记录缓存到内存中，这就是热度累积的缓存替换策略。

[0069] 具体地，在索引内存缓存层中执行查询请求时，内存缓存的服务进程对访问到的每条索引数据记录本周期内的访问次数，直到查询请求次数达到预设访问次数阈值，即到达热度计算周期时，服务进程触发缓存的更新替换。

[0070] 热度累积缓存替换策略的热度计算公式1如下：

[0071] $$score_n = \alpha \times \frac{visitCount}{countPeriod} + (1-\alpha) \times score_{n-1} \qquad \text{公式 1}$$

[0072] 其中0<a<1。公式中的countPeriod即热度计算周期，visitCount指当前热度计算周期中，该索引集合被访问的次数。历史热度scoren-1则反映集合累积的历史热度。参数a是衰减系数，用来确定当前周期累积的热度和历史热度在$score_n$中各自所占的权重。a越大，则最近的访问在数据访问热度中所占的权重越大，历史访问记录对数据热度的影响越小，反之亦然。

[0073] 为了降低热度计算带来的计算和更新开销，在执行查询请求时，内存缓存的服务进程将会对访问到的每条索引数据记录本周期内的访问次数，此时并不对内存缓存的数据进行替换。直到查询请求次数达到countPeriod，即到达热度计算周期时，服务进程触发缓存的更新替换。按照热度累积公式对所有的记录计算热度，根据热度排序，将热度排序TOP-K的集合记录缓存到内存中，集合中包含的记录条数是不固定的，所以选择TOP-K时，根据缓存空间能够容纳的记录条数限制计算出热度门限，高于门限的集合被缓存到内存中。

[0074] 然而，在系统初始阶段，缓存是大量空闲的。LRU算法在系统初始阶段的命中率提升很快，这是由于LRU算法中数据记录是访问即进入缓存，最长时间没有被访问的数据记录会在缓存充满后被淘汰。所以LRU可以快速地进入稳定状态。而热度累积的访问如果在系统初始阶段通过周期性地计算热度，等被访问数据记录的热度累积到门限时才可以进入缓存的话，初始阶段预热代价大。所以我们的热度累积算法在缓存空闲阶段做了优化，只要缓存有空闲，就采用“访问即进入”的策略，将所有访问到的记录都插入缓存。而当缓存充满以后，热度累积的缓存替换策略根据记录的热度累积评分选择“牺牲者”淘汰出内存，选择获得热度高分的记录保存在缓存中。

[0075] 本实施例的热度累积的缓存替换策略不仅考虑了数据的访问时间远近，同时考虑了数据的访问频率，所以比LRU更准确。从实验结果看出，热度累积的缓存替换策略明显优于LRU算法，和不使用内存缓存策略相比，可以提升5-15倍的查询性能。

[0076] 实施例2

[0077] 如图5所示，本实施例的一种海量大数据检索系统，至少包括：

[0078] (1)第一查询模块，其用于在索引内存缓存层的热点索引表中查询符合检索条件的数据；所述热点索引表中存在有访问频度不小于预设访问频度阈值的数据；

[0079] 当热点索引表中存在符合检索条件的数据时，则直接输出查询到的数据。

[0080] 本实施例的海量大数据以科研项目数据为例：

[0081] 首先，在索引内存缓存层的热点索引表中查询符合检索条件的数据；其中，检索条件为热力科研项目且项目属于中外合作项目；

[0082] 所述热点索引表中存在有访问频度不小于预设访问频度阈值(例如：10次/天)的数据。

[0083] 具体地，访问频度可采用在一定周期内的访问次数来表示。

[0084] 在所述第一查询模块中，在索引内存缓存层中执行查询请求时，内存缓存的服务进程对访问到的每条索引数据记录本周期内的访问次数，直到查询请求次数达到预设访问次数阈值，即到达热度计算周期时，服务进程触发缓存的更新替换。

[0085] 在所述第一查询模块中，根据索引集合在相应周期内的被访问次数计算所有记录的热度，根据热度排序，将热度排序位于前K位的集合记录缓存到内存中；其中，K为热度门限值；根据缓存空间能够容纳的记录条数限制计算出热度门限K，高于热度门限的集合被缓存到内存中。

[0086] 通常缓存的容量远远小于保存全部数据的磁盘数据库的容量，所以当缓存满了之后需要选择合适的牺牲者淘汰出缓存，这就是缓存替换策略。本实施例采用热度累积的缓存替换策略，其基本设计思想是周期性地累积记录被访问的次数。

[0087] 在内存中缓存的索引热点数据基于Redis的集合(Set)存储，Redis Set也是以<key,value>格式来组织数据。索引热点数据的索引主键做Redis Set的key，而索引集合作为Redis Set的value保存在内存缓存中。显然，具有相同索引列值的记录被绑定在同一个集合中，基于索引列值的查询命中是以集合为单位的。同时，它们也是热度累积的基本单位，每个集合都会累积它在一个计算周期内的访问次数。热度累积的缓存替换策略基于与LRU算法相同的假设：最近被访问的数据在最近的未来最有可能被重复访问。算法周期性地计算集合的累积热度，对所有的记录累积热度排序，选择累积热度TOP-K的索引记录缓存到

内存中，这就是热度累积的缓存替换策略。

[0088] 具体地，在索引内存缓存层中执行查询请求时，内存缓存的服务进程对访问到的每条索引数据记录本周期内的访问次数，直到查询请求次数达到预设访问次数阈值，即到达热度计算周期时，服务进程触发缓存的更新替换。

[0089] 热度累积缓存替换策略的热度计算公式1如下：

[0090] $$score_n = \alpha \times \frac{visitCount}{countPeriod} + (1-\alpha) \times score_{n-1}$$ 公式1

[0091] 其中0<a<1。公式中的countPeriod即热度计算周期，visitCount指当前热度计算周期中，该索引集合被访问的次数。历史热度scoren-1则反映集合累积的历史热度。参数a是衰减系数，用来确定当前周期累积的热度和历史热度在$score_n$中各自所占的权重。a越大，则最近的访问在数据访问热度中所占的权重越大，历史访问记录对数据热度的影响越小，反之亦然。

[0092] 为了降低热度计算带来的计算和更新开销，在执行查询请求时，内存缓存的服务进程将会对访问到的每条索引数据记录本周期内的访问次数，此时并不对内存缓存的数据进行替换。直到查询请求次数达到countPeriod，即到达热度计算周期时，服务进程触发缓存的更新替换。按照热度累积公式对所有的记录计算热度，根据热度排序，将热度排序TOP-K的集合记录缓存到内存中，集合中包含的记录条数是不固定的，所以选择TOP-K时，根据缓存空间能够容纳的记录条数限制计算出热度门限，高于门限的集合被缓存到内存中。

[0093] 然而，在系统初始阶段，缓存是大量空闲的。LRU算法在系统初始阶段的命中率提升很快，这是由于LRU算法中数据记录是访问即进入缓存，最长时间没有被访问的数据记录会在缓存充满后被淘汰。所以LRU可以快速地进入稳定状态。而热度累积的访问如果在系统初始阶段通过周期性地计算热度，等被访问数据记录的热度累积到门限时才可以进入缓存的话，初始阶段预热代价大。所以我们的热度累积算法在缓存空闲阶段做了优化，只要缓存有空闲，就采用“访问即进入”的策略，将所有访问到的记录都插入缓存。而当缓存充满以后，热度累积的缓存替换策略根据记录的热度累积评分选择“牺牲者”淘汰出内存，选择获得热度高分的记录保存在缓存中。

[0094] 本实施例的热度累积的缓存替换策略不仅考虑了数据的访问时间远近，同时考虑了数据的访问频率，所以比LRU更准确。从实验结果看出，热度累积的缓存替换策略明显优于LRU算法，和不使用内存缓存策略相比，可以提升5-15倍的查询性能。

[0095] (2)第二查询模块，其用于当热点索引表中不存在符合检索条件的数据时，则转到索引持久化存储层继续查询符合检索条件的数据，输出查询结果。

[0096] 具体地，在所述第二查询模块中，所述索引持久化存储层中持久化存储有HBase，所述HBase中存储有多个非主键属性索引表，每个非主键属性索引表用来对应存储管理用户表中的一个待查询非主键属性的索引。

[0097] 具体地，在所述第二查询模块中，所述索引内存缓存层为分布式内存缓存，使用一致性哈希来确定数据所在的服务器节点；在节点发生变化时，只有和变化节点相邻的节点数据需要迁移。

[0098] 实施例3

[0099] 一种计算机可读存储介质，其上存储有计算机程序，该程序被处理器执行时实现

如图3所示的海量大数据检索方法中的步骤。

[0100] 实施例4

[0101] 一种计算机设备，包括存储器、处理器及存储在存储器上并可在处理器上运行的计算机程序，所述处理器执行所述程序时实现如图3所示的海量大数据检索方法中的步骤。

[0102] 本领域内的技术人员应明白，本公开的实施例可提供为方法、系统、或计算机程序产品。因此，本公开可采用硬件实施例、软件实施例、或结合软件和硬件方面的实施例的形式。而且，本公开可采用在一个或多个其中包含有计算机可用程序代码的计算机可用存储介质(包括但不限于磁盘存储器和光学存储器等)上实施的计算机程序产品的形式。

[0103] 本公开是参照根据本发明实施例的方法、设备(系统)、和计算机程序产品的流程图和/或方框图来描述的。应理解可由计算机程序指令实现流程图和/或方框图中的每一流程和/或方框、以及流程图和/或方框图中的流程和/或方框的结合。可提供这些计算机程序指令到通用计算机、专用计算机、嵌入式处理机或其他可编程数据处理设备的处理器以产生一个机器，使得通过计算机或其他可编程数据处理设备的处理器执行的指令产生用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的装置。

[0104] 这些计算机程序指令也可存储在能引导计算机或其他可编程数据处理设备以特定方式工作的计算机可读存储器中，使得存储在该计算机可读存储器中的指令产生包括指令装置的制造品，该指令装置实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能。

[0105] 这些计算机程序指令也可装载到计算机或其他可编程数据处理设备上，使得在计算机或其他可编程设备上执行一系列操作步骤以产生计算机实现的处理，从而在计算机或其他可编程设备上执行的指令提供用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的步骤。

[0106] 本领域普通技术人员可以理解实现上述实施例方法中的全部或部分流程，是可以通过计算机程序来指令相关的硬件来完成，所述的程序可存储于一计算机可读取存储介质中，该程序在执行时，可包括如上述各方法的实施例的流程。其中，所述的存储介质可为磁碟、光盘、只读存储记忆体(Read-Only Memory，ROM)或随机存储记忆体(Random AccessMemory，RAM)等。

[0107] 上述虽然结合附图对本公开的具体实施方式进行了描述，但并非对本公开保护范围的限制，所属领域技术人员应该明白，在本公开的技术方案的基础上，本领域技术人员不需要付出创造性劳动即可做出的各种修改或变形仍在本公开的保护范围以内。

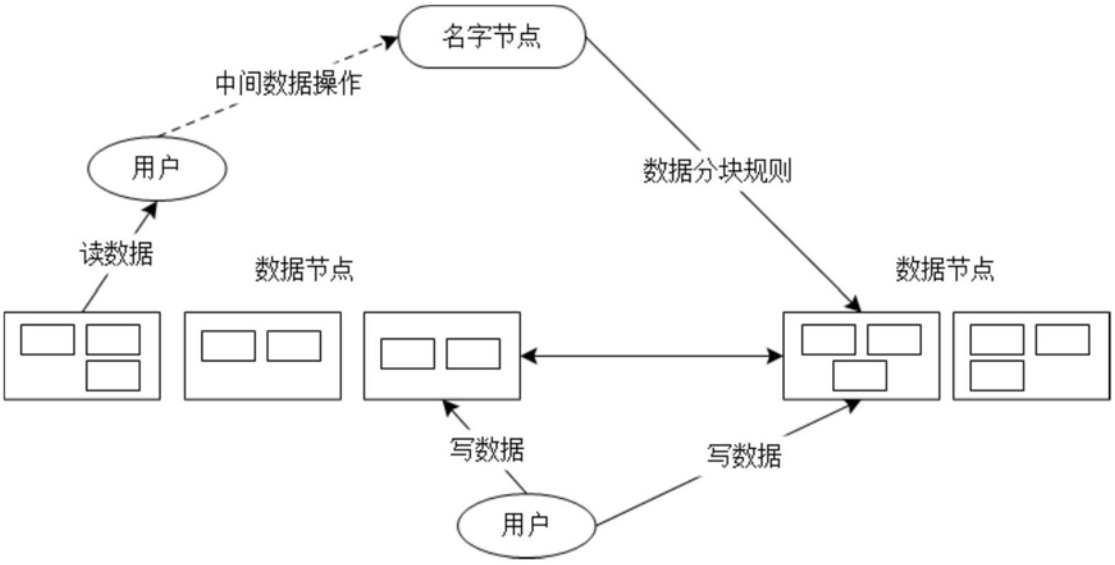

名字节点
中间数据操作
用户
数据分块规则
读数据
数据节点
数据节点
写数据
写数据
用户

图1

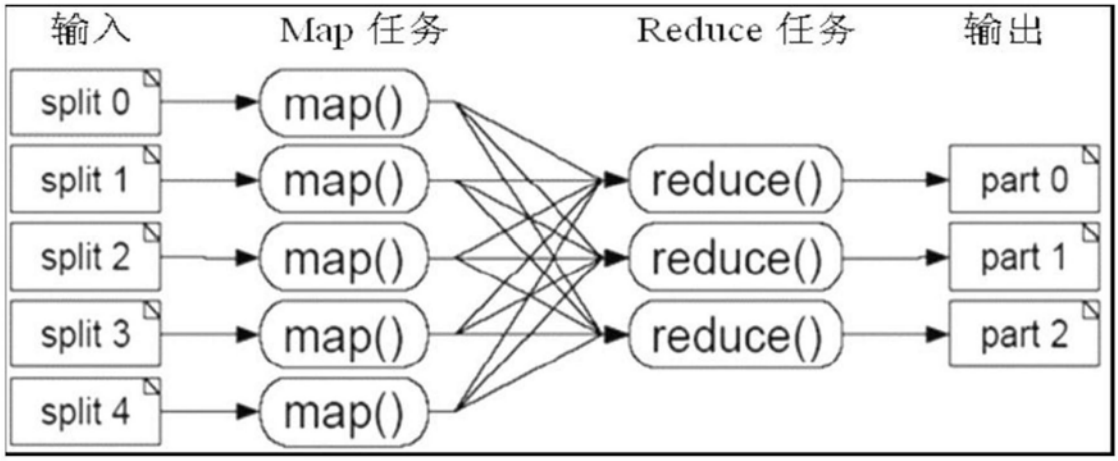

输入
Map 任务
Reduce 任务
输出
split 0
split 1
split 2
split 3
split 4
map()
map()
map()
map()
map()
reduce()
reduce()
reduce()
part 0
part 1
part 2

图2

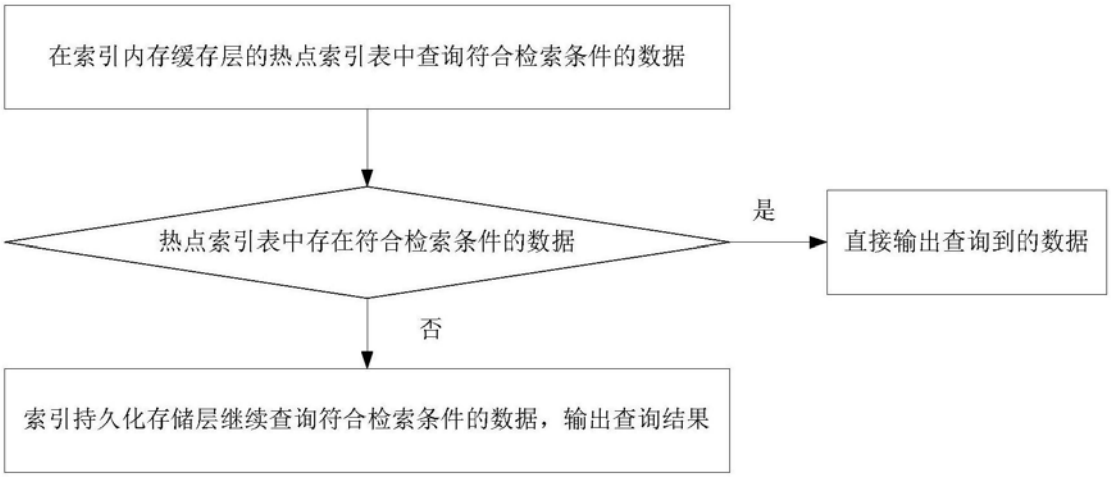

在索引内存缓存层的热点索引表中查询符合检索条件的数据
热点索引表中存在符合检索条件的数据
是
直接输出查询到的数据
否
索引持久化存储层继续查询符合检索条件的数据，输出查询结果

图3

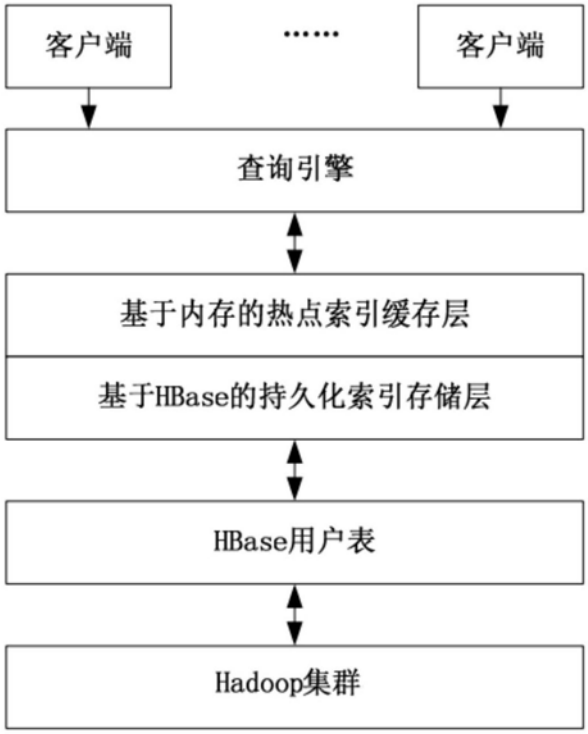

客户端
……
客户端
查询引擎
基于内存的热点索引缓存层
基于HBase的持久化索引存储层
HBase用户表
Hadoop集群

图4

图5