(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 113946580 A
(43)申请公布日 2022.01.18

(21)申请号 202111193982.7

(22)申请日 2021.10.13

(71)申请人 中国舰船研究设计中心
地址 430064 湖北省武汉市武昌区紫阳路268号

(72)发明人 王兴众 陈曦 黄金锋 谭亮 李莉丽

(74)专利代理机构 湖北武汉永嘉专利代理有限公司 42102
代理人 胡建平 李丹

(51)Int.Cl.
G06F 16/22(2019.01)
G06F 16/2453(2019.01)
G06F 16/27(2019.01)

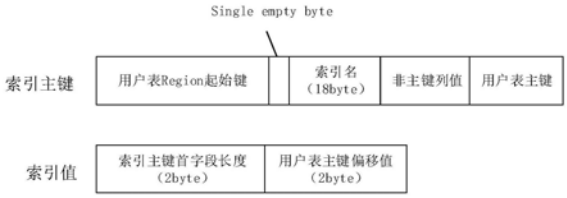
权利要求书1页 说明书4页 附图4页

(54)发明名称

一种海量异构日志数据检索中间件

(57)摘要

本发明公开了一种海量异构日志数据检索中间件,所述日志数据检索中间件为适用于日志数据的二级索引数据模型;二级索引结构包括索引主键和索引值;所述索引主键包括以下字段:原始表Region起始键字段RegionStartKey;索引名字段IndexName,所述IndexName字段用于唯一标识一个二级索引;非主键列值字段IndexKey,该字段由原始数据表中非主键列值字段拼接而成;用户表主键字段UserRowKey,该字段用于映射原始数据表中的主键;所述索引值包括用于存储索引主键原始表Region起始键字段长度的字段和用于存储用户表主键偏移值的字段。本发明设计的二级索引结构中,原始数据表数据只与索引主键字段有关,空间开销可控可计算,每个索引使用不到8%的空间开销而获得10倍以上的查询性能提升。



CN 113946580 A

1.一种海量异构日志数据检索中间件,其特征在于,所述日志数据检索中间件为适用于日志数据的二级索引数据模型;

二级索引结构包括索引主键和索引值;

所述索引主键包括以下字段:

原始表Region起始键字段RegionStartKey;该字段用于存储Region中的第一行;所述Region为HBase中所有数据的一个子集,

索引名字段IndexName,所述IndexName字段用于唯一标识一个二级索引;

所述原始表Region起始键字段RegionStartKey与索引名字段IndexName之间设有一个长度为1的空字节数组;

非主键列值字段IndexKey,该字段由原始数据表中非主键列值字段拼接而成;

用户表主键字段UserRowKey,该字段用于映射原始数据表中的主键,映射关系生成如下:通过IndexKey与查询条件的比较获得符合条件的索引表主键结果集合,通过该集合获得数据表主键,得到符合条件的记录;原始数据表的主键唯一性决定了该字段的唯一性,保证索引记录与数据表记录的唯一映射关系;

所述索引值包括用于存储索引主键原始表Region起始键字段长度的字段和用于存储用户表主键偏移值的字段。

2.根据权利要求1所述的海量异构日志数据检索中间件,其特征在于,所述索引名字段IndexName采用空字节数组填充策略保持该字段长度为18byte定长。

3.根据权利要求1所述的海量异构日志数据检索中间件,其特征在于,所述二级索引结构的存储方式如下:采用HT-Tree构建内存索引存储。

4.根据权利要求3所述的海量异构日志数据检索中间件,其特征在于,所述采用HT-Tree构建内存索引具体如下:每个叶子节点有四个哈希表,每个哈希表中有三个哈希桶,在使用HT-Tree构建内存索引时,根据关键字的键值通过查找算法查找到关键字可以插入的哈希表,然后通过计算,找到关键字可插入的哈希桶,判断哈希桶是否已满,若已满则分裂该节点,将该关键字插入,若未满则直接插入哈希桶。

5.根据权利要求4所述的海量异构日志数据检索中间件,其特征在于,所述关键字的键值为用户表主键字段中将主键原始字符串经过hash函数映射成整数形式得到主键的键值。

一种海量异构日志数据检索中间件

技术领域

[0001] 本发明涉及大数据检索技术,尤其涉及一种海量异构日志数据检索中间件。

背景技术

[0002] 在检索海量异构日志数据时,传统的数据存储与管理方法已难以适应当前大规模数据管理对效率的需求,因此非关系型数据库(Not Only SQL,NoSQL)得以迅速发展。HBase作为NoSQL数据库的代表,已被广泛应用于各行各业的数据存储与管理中。HBase在行键上建立了类B+树索引,可以高效地支持基于行键的快速数据查询,但对非行键的列没有建立索引,故进行非行键列查询时,需要对全表进行扫描,这样的查询效率十分低下。在日志检索与分析场景下,大多数查询都要求在非主键上进行,且需要综合各列数据综合进行日志分析,这就导致HBase对于日志数据的检索能力较弱。因此,需要一个高效快速的海量异构日志检索中间件。

发明内容

[0003] 本发明要解决的技术问题在于针对现有技术中的缺陷,提供一种海量异构日志数据检索中间件。

[0004] 本发明解决其技术问题所采用的技术方案是:一种海量异构日志数据检索中间件,所述日志数据检索中间件为适用于日志数据的二级索引数据模型;

[0005] 二级索引结构包括索引主键和索引值;

[0006] 所述索引主键包括以下字段:

[0007] 原始表Region起始键字段RegionStartKey;该字段用于存储Region中的第一行;所述Region为HBase中所有数据的一个子集,

[0008] 索引名字段IndexName,所述IndexName字段用于唯一标识一个二级索引;

[0009] 所述原始表Region起始键字段RegionStartKey与索引名字段IndexName之间设有一个长度为1的空字节数组;

[0010] 非主键列值字段IndexKey,该字段由原始数据表中非主键列值字段拼接而成;

[0011] 用户表主键字段UserRowKey,该字段用于映射原始数据表中的主键,映射关系生成如下:通过IndexKey与查询条件的比较获得符合条件的索引表主键结果集合,通过该集合获得数据表主键,得到符合条件的记录;原始数据表的主键唯一性决定了该字段的唯一性,保证索引记录与数据表记录的唯一映射关系;

[0012] 所述索引值包括用于存储索引主键原始表Region起始键字段长度的字段和用于存储用户表主键偏移值的字段。

[0013] 按上述方案,所述索引名字段IndexName采用空字节数组填充策略保持该字段长度为18byte定长。

[0014] 按上述方案,所述二级索引结构的存储方式如下:采用HT-Tree构建内存索引存储。

[0015] 按上述方案,所述采用HT-Tree构建内存索引具体如下:每个叶子节点有四个哈希表,每个哈希表中有三个哈希桶,在使用HT-Tree构建内存索引时,根据关键字的键值通过查找算法查找到关键字可以插入的哈希表,然后通过计算,找到关键字可插入的哈希桶,判断哈希桶是否已满,若已满则分裂该节点,将该关键字插入,若未满则直接插入哈希桶。

[0016] 按上述方案,所述用户表主键字段中将主键原始字符串经过hash函数映射成整数形式得到主键的键值。

[0017] 本发明产生的有益效果是:本发明设计的二级索引结构中,原始数据表数据只与索引主键字段有关,索引值存储解析索引主键的字节信息,且长度固定,所以空间开销可控可计算。每个索引使用不到8%的空间开销而获得10-100倍以上的查询性能提升。

附图说明

[0018] 下面将结合附图及实施例对本发明作进一步说明,附图中:

[0019] 图1是本发明实施例的二级索引结构示意图;

[0020] 图2是本发明实施例的HT-Tree结构示意图;

[0021] 图3是本发明实施例的IndexMapper示意图。

[0022] 图4是本发明实施例的批量构建索引流程图;

[0023] 图5是本发明实施例的生成索引表put操作列表流程图;

[0024] 图6是本发明实施例的协处理器工作流程。

具体实施方式

[0025] 为了使本发明的目的、技术方案及优点更加清楚明白,以下结合实施例,对本发明进行进一步详细说明。应当理解,此处所描述的具体实施例仅用以解释本发明,并不用于限定本发明。

[0026] 如图1所示,一种海量异构日志数据检索中间件,所述日志数据检索中间件为适用于日志数据的二级索引数据模型;

[0027] 二级索引结构包括索引主键和索引值;

[0028] 所述索引主键包括以下字段:

[0029] 原始表Region起始键字段RegionStartKey;该字段用于存储Region中的第一行;所述Region为HBase中所有数据的一个子集;region有三个要素:a.它所属于哪张表;b.它所包含的第一行(startkey);c.它所包含的最后一行(endkey)。该字段存储的就是上述startkey。在HBase中的记录都按照RowKey字典序排列,加上该字段后,具有相同主键前缀的索引表和数据表记录都会被分配到相同RegionServer。保证查询过程对于两表的请求只需要一次RPC即可完成,如果查询规模越大,这种优化就会越明显。

[0030] 索引名字段IndexName,所述IndexName字段用于唯一标识一个二级索引;采用空字节数组填充策略保持该字段长度为18byte定长;

[0031] 所述原始表Region起始键字段RegionStartKey与索引名字段IndexName之间设有一个长度为1的空字节数组;

[0032] 非主键列值字段IndexKey,该字段由原始数据表中非主键列值字段拼接而成;该字段变长,在构造索引元数据时就定义其最大长度及数据类型,若拼接后的字节数组长度

不足,采用空字节进行填充。

[0033] 用户表主键字段UserRowKey,该字段用于映射原始数据表中的主键,映射关系生成如下:通过IndexKey与查询条件的比较获得符合条件的索引表主键结果集合,通过该集合获得数据表主键,得到符合条件的记录;原始数据表的主键唯一性决定了该字段的唯一性,保证索引记录与数据表记录的唯一映射关系;

[0034] 所述索引值包括用于存储索引主键原始表Region起始键字段长度的字段和用于存储用户表主键偏移值的字段。

[0035] 在上述定义的索引数据和用户数据中,主键添加的前缀可以将主键原始字符串经过hash函数映射成整数形式。比较良好的hash函数,比如MD5、BKDRHash函数等,可以将字符串均匀、随机的分配到一个整数范围。如果经过一次hash函数映射之后的结果依然不符合要求,可以在此基础上再次进行散列,使得散列结果能够在整体上比较均匀的分散到各个Region Server上。前缀构造完成后,就可以将前缀和原始数据的主键拼接在一起,构成新的主键,插入HBase,同时构建相应的索引主键,同样插入Hbase。读取和插入HBase时,都需要对查询和插入的主键或者非主键构建的索引数据进行散列,然后再进行常规检索和插入。

[0036] 上述设计的二级索引在逻辑上实现了索引结构,我们还对其进行物理存储上的优化,对上述二级索引采用特定的物理存储结构,采用内存索引设计,这样就提高了该索引中间件的索引速度。本中间件使用了HT-Tree构建内存索引,如图2所示,每个叶子节点有四个哈希表,每个哈希表中有三个哈希桶,在使用HT-Tree构建内存索引时,需要通过查找算法查找到关键字可以插入的哈希表,然后通过计算,找到关键字可插入的哈希桶,判断哈希桶是否已满,若已满则分裂该节点,将该关键字插入,若未满则直接插入哈希桶。

[0037] 由于数据量巨大,列的种类繁多并持续动态增加,我们还提供了一种方法来动态的根据原始数据表来管理构建对应的索引表。采用Hadoop软件体系中提供的丰富MapReduce工具库,来加速索引构建过程。图3和图4展示了通过MapReduce来构建索引流程,它的输入是<Rowkey,Result>,输出是<Object,Put>,其中RowKey表示原始数据表主键,Result表示RowKey对应的具体记录,Object是输出表,Put是对于索引表操作的集合。具体算法步骤:

[0038] 1、读取JSON索引配置文件,生成索引元数据。

[0039] 2、向原始数据表写入索引元数据、创建索引表以及配置MapReduce任务。上述操作在执行MapReduce之前就已提前完成,之后才会对命名空间中所有原始数据表进行索引构建。

[0040] 3、执行MapReduce的map阶段。主要是删除标记为DeleteType的数据单元,将过滤后的Cell加入到结果集中。

[0041] 4、将上述结果集作为输入,经过图5所示的流程可以生成一个针对索引表的Put操作列表。

[0042] 生成了索引表Put操作列表后,只需将该操作列表作为MapReduce执行环境中的输入,因为HBase原生的在数据存储层集成了MapReduce,所有的任务将会在HBase中自动执行完成。由图3可以看出,批量构建阶段的MapReduce程序不进行reduce操作只进行map操作,避免MapReduce shuffle过程,使得索引构建速度更快,效率更高。

[0043] 基于协处理器RegionObserver来实现索引的流式更新以及内存索引构建。协处理器是HBase提供的一个数据处理框架，通过它我们可以把计算逻辑推到托管数据的RegionServer节点上，并且跨越所有的RegionServer并行执行，其工作流程如附图6所示。协处理器允许用户在RegionServer上运行自己的代码，即用户可以执行Region级别的操作，原先这种级别的操作没有开放给用户权限，因此在老版本的HBase中有许多令人诟病的问题，比如无法轻易建立二级索引，难以执行求和、技术、排序等操作。在HBase引入了协处理器后，可以直接将计算过程放置在server端，减少了通讯开销，从而获得很好的性能提升。HBase的协处理器框架主要包含Observer(观察者)和endpoint(终端)两类组件。其中endpoint可以理解为RDBMS中的存储过程，通过添加一些远程过程调用来动态扩展RPC协议。Observer类似于RDBMS中的Trigger(触发器)，当数据表进行增删改等操作时就会触发钩子函数，紧接着被服务端调用，最终实现索引表的流式更新逻辑。

[0044] 设计了JSON格式的索引配置文件，以便灵活轻便的创建以及维护二级索引，同时提供了索引生成、解析及查询过程中的必要决策信息。为了更方便的将JSON格式数据反序列化为Java对象，采用GSON(一种Google开源的用于处理JSON格式数据的类库)进行配置文件的解析。

[0045] 应当理解的是，对本领域普通技术人员来说，可以根据上述说明加以改进或变换，而所有这些改进和变换都应属于本发明所附权利要求的保护范围。

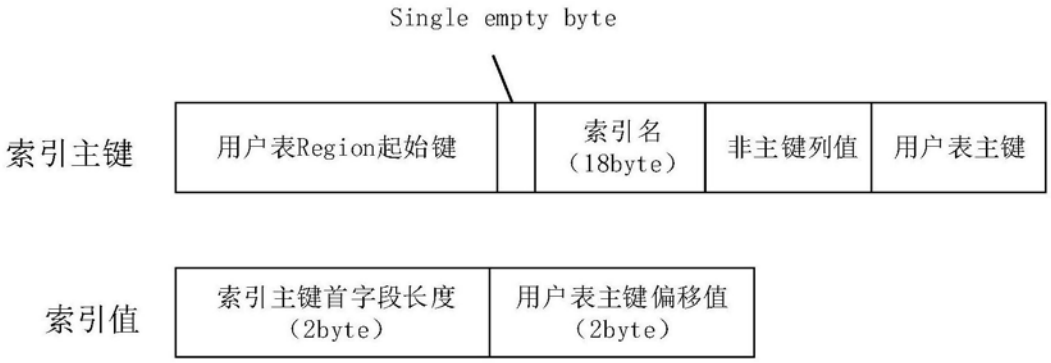

Single empty byte
索引主键
用户表Region起始键
索引名
（18byte）
非主键列值
用户表主键
索引值
索引主键首字段长度
（2byte）
用户表主键偏移值
（2byte）

图1

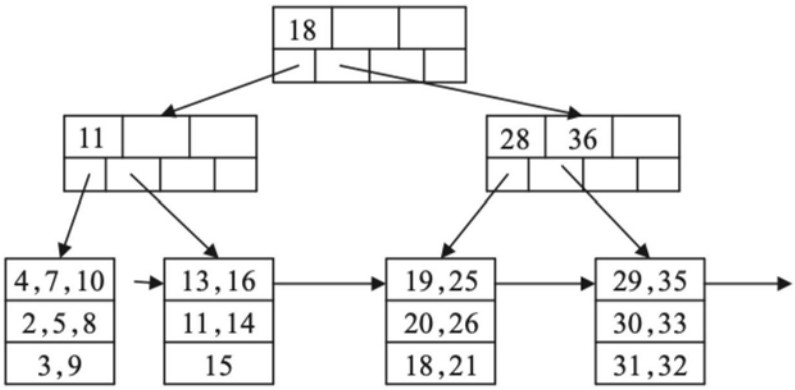

18
11
28
36
4,7,10
2,5,8
3,9
13,16
11,14
15
19,25
20,26
18,21
29,35
30,33
31,32

图2

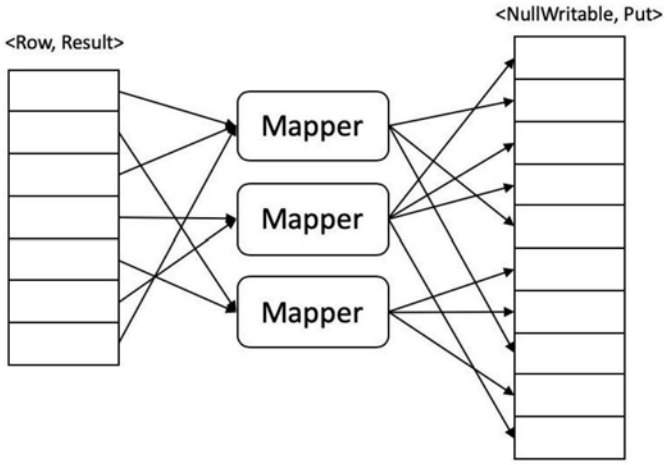

<Row, Result>
Mapper
Mapper
Mapper
<NullWritable, Put>

图3

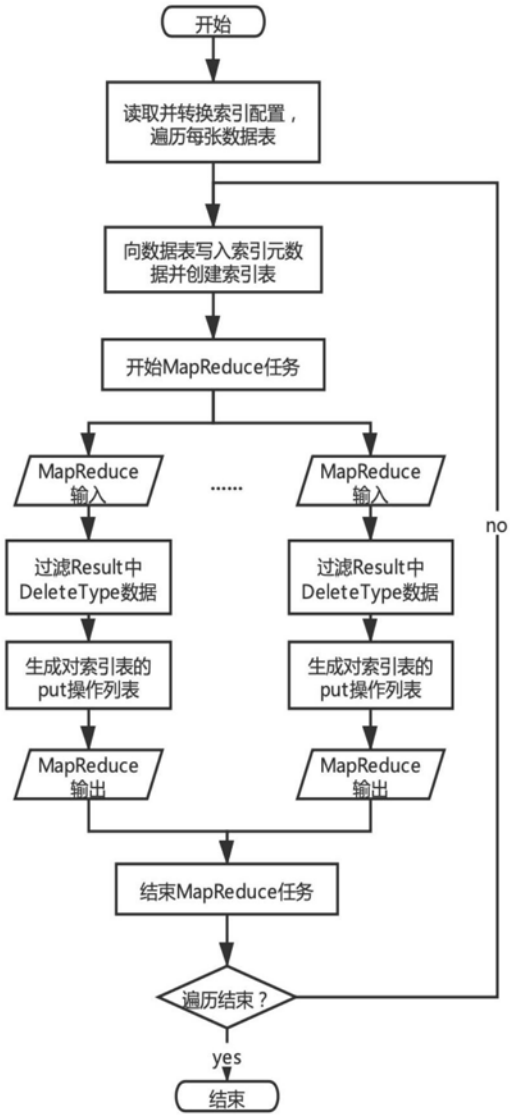

开始
读取并转换索引配置，遍历每张数据表
向数据表写入索引元数据并创建索引表
开始MapReduce任务
MapReduce输入
.....
MapReduce输入
过滤Result中DeleteType数据
过滤Result中DeleteType数据
生成对索引表的put操作列表
生成对索引表的put操作列表
MapReduce输出
MapReduce输出
结束MapReduce任务
遍历结束？
no
yes
结束

图4

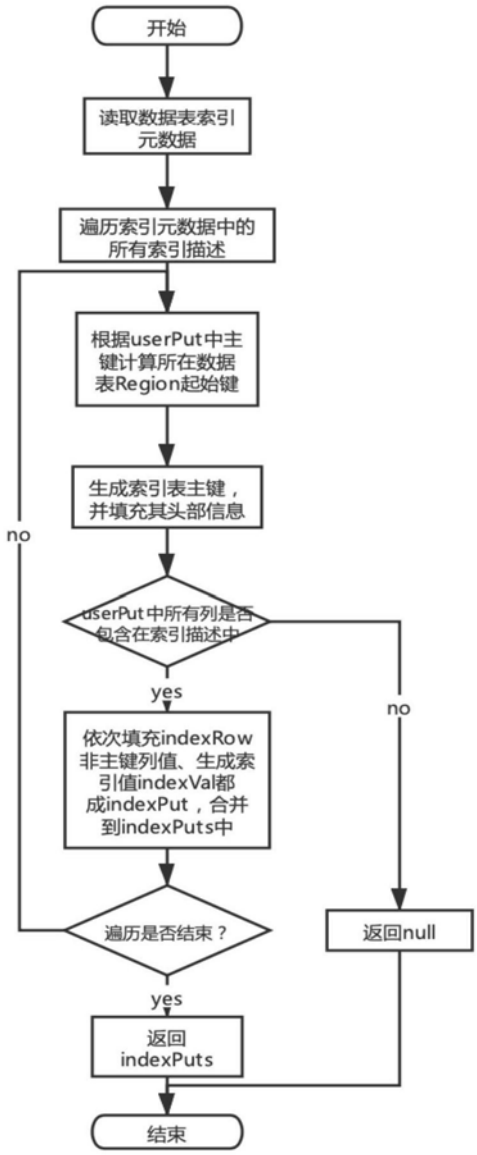

开始
读取数据表索引元数据
遍历索引元数据中的所有索引描述
根据userPut中主键计算所在数据表Region起始键
生成索引表主键，并填充其头部信息
userPut中所有列是否包含在索引描述中
no
yes
no
依次填充indexRow非主键列值、生成索引值indexVal都成indexPut，合并到indexPuts中
遍历是否结束？
返回null
yes
返回indexPuts
结束

图5

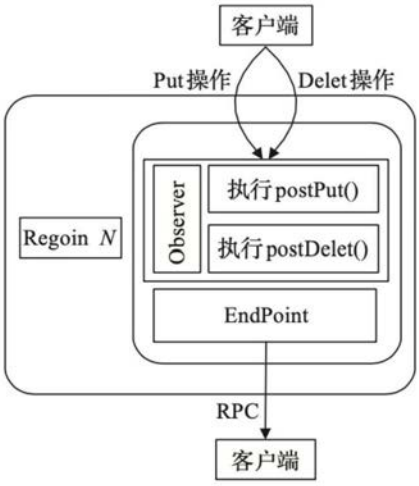

客户端
Put操作
Delet操作
Regoin N
Observer
执行postPut()
执行postDelet()
EndPoint
RPC
客户端

图6