Journal of Computer Applications
计算机应用,2014,34(11):3218-3221,3278
ISSN 1001-9081
CODEN JYIIDU
2014-11-10
http://www.joca.cn

文章编号:1001-9081(2014)11-3218-04
doi:10.11772/j.issn.1001-9081.2014.11.3218

基于 KD 树和 R 树的多维云数据索引

何　婧[1,2],吴　跃[1],杨　帆[2],尹春雷[3],周　维[2*]
(1.电子科技大学 计算机科学与工程学院,成都 611731;　2.云南大学 软件学院,昆明 650091;
3.云南农业大学 建筑工程学院,昆明 650201)
(*通信作者电子邮箱 wz.weihou@gmail.com)

摘　要:针对云存储系统大多基于键值对<key,value>模型存储数据,多维查询需要对整个数据集进行完全扫描,查询效率较低的问题,提出了一种基于 KD 树和 R 树的多维索引结构(简称 KD-R 索引)。KD-R 索引采用双层索引模式,在全局服务器建立基于 KD 树的多维全局索引,在局部数据节点构建 R 树多维本地索引。基于性能损耗模型,选取索引代价较小的 R 树节点发布到全局 KD 树,从而优化多维查询性能。实验结果表明:与全局分布式 R 树索引相比,KD-R 索引能够有效提高多维范围查询性能,并且在出现服务器节点失效的情况下,KD-R 索引同样具有高可用性。

关键词:云计算;云存储;云数据管理;多维索引;范围查询
中图分类号:TP311.13　　**文献标志码**:A

Multi-dimensional cloud index based on KD-tree and R-tree

HE Jing[1,2], WU Yue[1], YANG Fan[2], YIN Chunlei[3], ZHOU Wei[2*]
(1. *School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu Sichuan* 611731, *China*;
2. *School of Software, Yunnan University, Kunming Yunnan* 650091, *China*;
3. *School of Architecture Engineering, Yunnan Agricultural University, Kunming Yunnan* 650201, *China*)

Abstract: Most existing cloud storage systems are based on the <key, value> model, which leads to a full dataset scan for multi-dimensional queries and low query efficiency. A KD-tree and R-tree based multi-dimensional cloud data index named KD-R index was proposed. KD-R index adopted two-layer architecture: a KD-tree based global index was built in the global server and R-tree based local indexes were built in local server. A cost model was used to adaptively select appropriate R-tree nodes to publish into global KD-tree index. The experimental results show that, compared with R-tree based global index, KD-R index is efficient for multi-dimensional range queries, and it has high availability in the case of server failure.

Key words: cloud computing; cloud storage; cloud data management; multi-dimensional index; range query

0　引言

云计算是随着存储、计算以及通信技术的飞速发展而出现的一种共享基础资源的计算模型[1]。数据中心是云计算提供服务的基础,因此,作为数据中心基础设施之一的云存储系统在云计算中扮演了很重要的角色。

当前典型的云存储系统包括:Google 公司的 BigTable[2]、Amazon 公司的 Dynamo[3]、Facebook 的 Cassandra[4],以及开源的 HBase[5-6]等。上述云存储系统主要基于<key,value>方式存储数据,这样的存储方式具有高可扩展性、高可用性和容错性等特点,能够实现对海量数据的高效存储和处理。然而,<key,value>方式主要提供基于键值的高效的点查询和范围查询,针对复杂查询(如:范围查询、多维查询、*K* 最近邻(*K*-Nearest Neighbor, KNN)查询等)则需要运行 MapReduce[7]任务去扫描整个数据集。虽然 MapReduce 技术的并行性可以在一定程度上提高数据访问的效率,但是,在大数据情况下,数据访问的时效性仍然不能满足实际应用的需求。在不移动数据的前提下,索引是提高查询性能、扩展查询方式的一种有效措施,因此,近年来,云计算平台上的数据索引研究已成为了学术界和工业界关注的核心问题之一。

文献[8]提出了一种使用全局分布式 B 树(Distributed B-tree)构建云中大规模数据集的索引结构,该方法实现简单,具有高扩展性和容错性,但是在实现时需要在客户端缓存大量信息,内存消耗大,且由于采用的是 B 树索引,多维查询效率较低。文献[9]提出了一种改进的 B^+ 树索引方案,该方案是一种双层索引方案,对每个计算机节点的数据建立一个本地的 B^+ 树局部索引,然后选取每个局部索引中的部分 B^+ 树节点构成全局索引,全局索引采用了 P2P 网络 Baton 对全局索引节点进行组织。该方案对单个属性上的查询效率较高,但仍然无法有效支持多维查询。RT-CAN[10](R-Tree based index in Content Addressable Network)则是针对多维查询设计的一种双层索引方案,局部索引采用了 R 树结构,全局索引

收稿日期:2014-05-14;**修回日期**:2014-07-10。
基金项目:国家自然科学基金资助项目(61363021);云南省教育厅科学研究基金资助项目(2014Y013)。
作者简介:何婧(1978-),女,云南红河人,讲师,博士研究生,CCF 会员,主要研究方向:云数据管理、数据挖掘、知识发现;　吴跃(1958-),男,四川成都人,教授,博士生导师,主要研究方向:云计算、数据挖掘;　杨帆(1990-),男,湖北武汉人,硕士研究生,主要研究方向:云存储、分布式计算;　尹春雷(1974-),男,云南蒙自人,讲师,硕士研究生,主要研究方向:计算机网络、建筑自动化;　周维(1974-),男,云南昆明人,副教授,博士,主要研究方向:分布式计算、云计算。

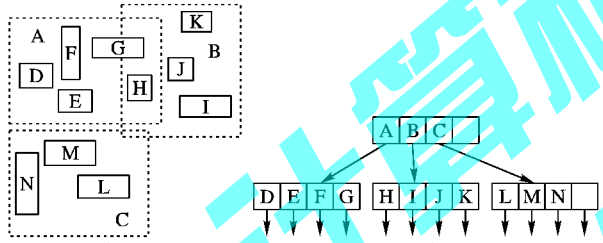
采用了支持多维查询的 P2P 网络 CAN(Content Addressable Network),该方案具有较高的多维查询效率。文献[11]提出了另一种支持多维查询的双层索引结构 QT-Chord,局部索引采用了改进的四叉树,全局索引采用了 Chord 覆盖网络,通过对四叉树进行编码,使得局部索引到全局索引的节点发布策略更简洁。以上两种多维索引结构[10-11]的全局索引结构都采用的是 P2P 网络,可扩展性高,但是,目前云计算的主流计算平台是 Hadoop[12],Hadoop 是一种 Master-Slave 结构,要将 P2P 网络移植到 Hadoop 平台上较困难。

基于以上分析可看出:云存储多维索引是当前研究的热点,虽然已经有部分工作提出基于双层索引的云存储多维索引,但是现有工作还未见在上层索引中采用 KD 树、下层索引中采用 R 树的结构。因此,本文提出了一种基于 KD 树和 R 树的多维云数据索引——KD-R 索引。

1　数据结构分析

1.1　R 树

R 树是 B 树在多维上的扩展,它基于空间划分的思想把数据分割为多个数据区,每个数据区用一个最小外包矩形(Minimal Bounding Rectangle,MBR)表示,R 树中每个节点的 MBR 包含其所有孩子的 MBR[13]。典型的二维 R 树结构如图 1 所示。



(a) R树平面图　　(b) R树结构图

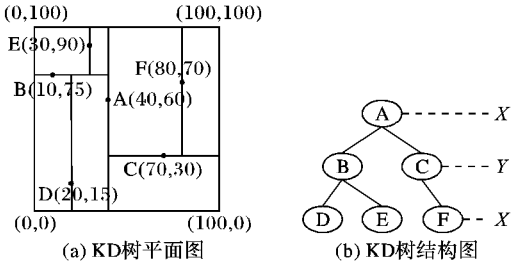
图 1　二维 R 树结构

图 1 中,节点 A 的 MBR 包含节点 D、E、F、G 的 MBR,节点 B 的 MBR 包含节点 H、I、J、K 的 MBR,以此类推。实际数据信息存放在叶子节点指向的数据中,R 树上的所有内部节点都不包含数据信息。

R 树索引与 B 树索引不同,不需要为多维查询建立多个索引,它们又有类似之处,基于 R 树索引的多维查询只需要遍历可能包含查询结果的数据,查询效率高;同时,R 树是一种动态的数据结构,查询、插入、删除可以交叉进行,不需要定期的全局结构重组,索引维护代价低。因此,本文采用 R 树作为本地数据节点的多维索引结构。

1.2　KD 树

KD 树是把二叉搜索树推广到多维数据的一种主存数据结构[13]。KD 树是一棵二叉树,它的内部节点包含一个属性 a 和一个值 V,从而将数据点分成 $a \leqslant V$ 和 $a > V$ 两个部分。由于所有维的属性在层间循环,所以树的不同层上的属性是不同的。图 2 展示了采用 KD 树划分二维空间数据点的结构图。图 2 中,在 $K=2$ 的情况下,首先在 X 属性上进行划分,选取 A 节点作为划分节点,将二维空间划分为 $X \leqslant 40$ 和 $X > 40$ 两个部分,左子空间包含 B、D、E 节点,右子空间包含 C、F 节点。然后在每个子空间上对 Y 属性进行划分,左子空间选择 B 节点作为划分节点,右子空间选择 C 节点作为划分节点。以此类推,直到空间中只包含一个节点为止。



(a) KD树平面图　　(b) KD树结构图

图 2　二维 KD 树结构

在基于 KD 树的查询中,每次只需要对多维查询中的一个维度值进行处理,计算简单,另外,交替对不同属性的取值进行检测,可以更快速地将搜索范围缩减到包含查询结果的节点。

文献[14]对 R 树和 KD 树在云计算环境中处理多维查询的效率进行了研究,结果表明 KD 树的效率更高。本文分析认为原因是 KD 树的中间节点保存了数据信息,不需要查询到叶节点就可以发现查询命中数据,并且 KD 树索引的空间划分不会出现区域重叠现象,更适合用作云计算环境中的上层全局索引,从而在多维查询过程中尽快发现包含查询结果的局部数据节点。因此,文本采用 KD 树构建全局索引。

2　KD-R 索引

2.1　KD-R 索引机制

分布式存储系统通常采用“水平分区”的方式将大规模数据集划分为多个小的数据单元,称为“数据片”,然后将这些数据片按照负载均衡的原则存储到云计算环境中不同的计算机节点上。为了提高数据查询性能,可以依据传统方法为整个数据集建立统一的全局分布式索引,但是在大数据情况下,全局分布式索引本身占用的存储空间会急剧增长,同时索引的维护也变得困难。

为了优化这种为整个数据集建立全局分布式索引的方法,KD-R 索引采用了分层结构来建立索引,即为每个局部数据节点中存储的数据建立一个 R 树索引,用于管理本地计算机服务器上的数据,同时,按照一定的算法选择策略,将每个 R 树索引中的部分节点选择出来,构建全局 KD 树索引,全局索引可以看成是在本地 R 树索引之上的第二层索引。具体结构见图 3。

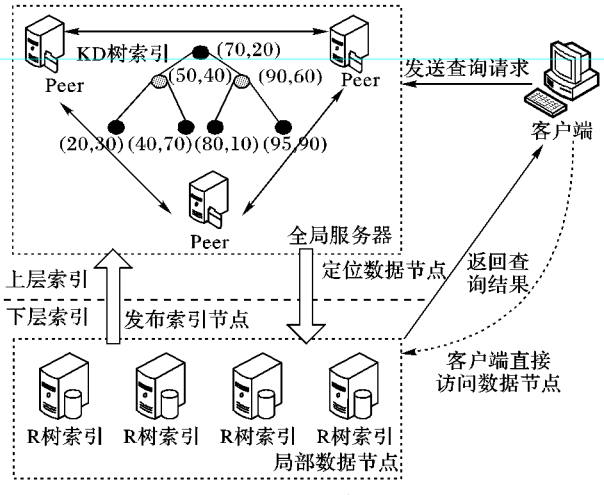


图 3　KD-R 索引

KD-R 索引方案把查询过程分成两个阶段:第一阶段,应用服务器将客户端提出的查询请求发送到全局索引层,通过检索 KD 树,找到可能包含查询结果的 KD 树节点;第二阶段,对于命中的 KD 树节点,根据节点上存储的 IP 信息,找到其对应的局部数据节点,然后通过对局部数据节点上本地 R 树索引的检索,最终找到满足条件的查询结果。最后,将查询结果返回给客户端,客户端针对查询结果的修改操作可以直接提交到局部数据节点,不需要再次经过全局索引,从而提高数据处理的效率。

在图 3 的设计中,全局索引可以采用集中式方案实现,也可以采用分布式方案实现。在分布式方案中,每个局部数据节点在 KD-R 索引中有两种身份:一种身份是作为下层索引结构的数据存储节点,保存本地数据、本地的 R 树索引,并负责向上层索引发布它的本地索引节点;另一种身份是作为上层索引由 KD 树管理的一个对等节点(peer),多个对等节点按照 KD 树的组织方式来共同构成上层的全局索引。

2.2 自适应发布算法

KD-R 索引是一种双层索引,在查询执行过程中,通过上层的全局索引可以实现高效的节点路由,快速发现包含查询结果的局部数据节点。由于存储空间的限制和查询效率的要求,不能把每个局部数据节点上的本地 R 树索引的全部节点发布到全局索引中。但是,如果发布的信息过少,又有可能导致局部数据信息遗漏,使查询无法正确完成。

因此,需要研究一种高效的自适应发布算法,该算法既要保证发布的数据信息的完整性,又要使得索引维护代价尽量小。

算法 1 AdaptiveIndexPublishing(N_j)。

```
输入  N_j: the local R-tree index server。
1)    N_j publishes the root node of its local R-tree
2)    While true do
3)      //n_j is the published local R-tree node
4)      for each n_j in the N_j's published set
5)        if isLessCost(n_j. children) then
6)          index n_j's children instead of n_j;
7)        else
8)          if ((cost(n_j) + cost(n_j. siblings)) <
              cost(n_j. parent)) then
9)            index n_j's parent instead of n_j and n_j's sibings;
10)     endfor
11)   endwhile
```

算法 1 描述了本文采用的自适应发布算法。为了保证发布 R 树索引节点的正确性和完整性,初始情况下,将每个局部数据节点中的 R 树索引的根节点发布到全局索引中。然后,每个局部数据节点周期性地检查其发布的节点集。如果某个已发布的 R 树节点的孩子节点的索引代价更小,则用孩子节点替换该节点;如果某个已发布的节点及其兄弟节点的索引代价高于其父亲节点的索引代价,则用父亲节点替换它们:由此保证整个 KD-R 索引的完整性和高效性。

2.3 性能损耗模型

为了计算 KD-R 索引节点的索引代价,本文定义了相应的性能损耗模型。在云环境下的多维查询过程中,性能损耗主要来自两个方面:查询过程产生的损耗和索引维护产生的损耗。定义性能损耗模型如下:

$$C(n) = C_q(n) + C_m(n) \tag{1}$$

其中:$C(n)$ 表示某个 KD-R 索引节点 n 的性能损耗,$C_q(n)$ 表示查询过程的损耗,$C_m(n)$ 表示索引维护的损耗。

查询损耗主要指对全局 KD 树进行搜索的时间耗费,因此,平均情况下查询耗费为:

$$C_q(n) = \mathrm{lb}(N) \tag{2}$$

其中 N 表示全局 KD 树的节点总数。因为局部数据节点上的 R 树索引可以缓存在内存中,所以 R 树查询的耗费通常可以忽略。

在局部数据节点上发生的数据插入或者删除会触发相应的 R 树索引节点的更新,这样会导致全局索引上存储的信息与局部索引节点不一致,此时就需要对局部索引节点和全局索引节点的信息进行同步操作。本文的索引维护损耗指的就是同步操作的损耗。

如果一个局部数据节点上的 R 树索引的节点 N_i 发生了分裂,同步过程分为两步:1)检查 N_i 的标识位,以确定该节点是否是一个已发布节点,如果状态是“已发布”,则分裂后每个节点的状态设定为“正在发布”,找到 N_i 对应的 KD 树节点 N_j,将节点 N_j 删除,处理时间为 lb(N);2)在全局索引 KD 树中插入 N_i 分裂后的 m 个节点,总的插入时间为 m lb(N),插入操作完成后将 R 树中这 m 个节点的状态更新为“已发布”。一个节点 n 的索引维护时间为:

$$C_m(n) = (m+1)\mathrm{lb}(N) \tag{3}$$

同理,R 树节点合并的索引维护时间与节点分裂的时间一样,也是 $(m+1)$lb(N)。

2.4 查询处理

本文主要研究了云环境下基于 KD-R 索引的多维点查询和多维范围查询的处理。

2.4.1 点查询处理

基于 KD-R 索引的多维点查询定义为 Query(key),其中 $key = (v_1, v_2, \cdots, v_d)$,表示一个 d 维情况的下的点。给定一个 R 树索引节点 n,$[l_1, m_1]$、$[l_2, m_2]$、…、$[l_d, m_d]$ 表示它的数值范围的区间,如果 key 满足 $l_i \leqslant v_i \leqslant m_i (1 \leqslant i \leqslant d)$,则这个节点 n 就是一个满足查询条件的结果。

由于 R 树内节点存储的是一个多维范围,在将 R 树节点发布到全局索引时,算法 1 发布的是 R 树节点的下界,为了保证查询的完整性,在搜索全局索引 KD 树找到第一个满足查询条件的数据节点 N_{init} 后,还需要检查该节点的父亲节点和兄弟节点是否满足查询。具体算法见算法 2。

算法 2 PointQueryProcess(key)。

```
输入  key: keyword for search。
输出  S_result: the result of local R-tree index。
1)    S_result = 0; S_kd = 0
2)    N_init = Kd-tree. lookup(key);
3)    add N_init, N_init. parent, and N_init. siblings into S_kd
4)    for each S_j in S_kd do
5)      for i = 1 to d do
6)        if key. vi < S_j. vi then
7)          delete S_j from S_kd;
8)    endfor
9)    While S_kd is not null do
10)     S_result. add(LocalSearch(R-tree, key)
11)   return S_result;
```

算法 2 在处理点查询 Query(key) 时,首先把查询发送到全局 KD 树索引,根据 KD 树查询算法找到第一个满足查询条

件的节点 N_{init},N_{init} 查询它的缓存中存储的全局索引,找到它的父亲节点和兄弟节点,完成全局索引查询,生成中间结果集 S_{kd}。针对 S_{kd} 集合中任一个节点进行计算,在 d 维空间中如果存在 $v_i < l_i(1 \leqslant i \leqslant d)$,则该节点不可能包含查询结果,将其删除;否则再根据该节点的 IP 地址找到局部数据存储节点,执行 R 树索引查询,最终将查询结果返回给客户端。

2.4.2　范围查询处理

基于 KD-R 索引的范围查询定义为 Query(*key*, *dist*),表示查询与节点 *key* 距离为 *dist* 的范围内的所有节点,其中 $key = (v_1, v_2, \cdots, v_d)$,表示一个 d 维情况的下的点。以 *key* 为圆心,*dist* 为半径的圆就是查询范围,记为 Q。

假设一个 R 树节点 n 的数值范围区间是 $n_r = ([l_1, u_1], [l_2, u_2], \cdots, [l_d, u_d])$,如果查询范围 Q 与节点 n 的范围 n_r 有交集,则节点 n 是一个满足查询要求的节点。

以二维情况为例,给定范围查询 $Q1$(*key*, *dist*),其中:$key = (v_1, v_2)$,$dist = r$。由于局部 R 树索引发布到全局 KD 树索引的是 R 树节点的下界,因此,在搜索全局 KD 树时,初始节点(0,0)到 $Q1$ 的监测节点 N_{flag} 的区间内包含的节点都有可能是满足查询的节点,$N_{flag} = (v_1 + dist, v_2 + dist)$。

算法 3　RangeQueryProcess(*key*, *dist*)。

输入　*key*: keyword for search; *dist*: search radius。

输出　S_{result}: the result of local R-tree index

1)　$S_{result} = 0$; $S_{kd} = 0$;

2)　N_{init} = Kd-tree. nearestpoint (*region*, N_{flag});

　　// *region* represents the search data range of Kd-tree

3)　*region* = generateSearchRange(N_{init});

4)　S_{kd}. add(GlobleSearch(Kd-tree, *region*)

5)　While S_{kd} is not null do

6)　　S_{result}. add(LocalSearch(R-tree, *key*, *dist*)

7)　return S_{result};

在 KD-R 索引中,用户发送的查询请求首先发送到全局 KD 树索引,算法 3 的第 2)行首先找到距离用户查询范围的监测节点 N_{flag} 最近的 KD 树节点 N_{init},生成满足用户查询的数据区间 *region*,该区间内的节点都有可能满足查询。N_{init} 首先查询它的缓存中存储的满足 *region* 区间的全局索引,再将查询请求提交到其他相关节点,以执行各个局部数据节点中本地的 R 树索引查询,最后将查询结果返回给客户端。

3　实验与分析

本文采用 Peersim[15] 模拟器进行模拟实验,对 KD-R 索引的多维查询性能进行了评估,Peersim 可以方便地模拟不同规模的多节点数据存储环境。实验用机器配置为 Intel Core i3-350M 2.26 GHz CPU,2 GB 内存,320 GB 磁盘,操作系统为 Ubuntu 11.1.0(64 bit)。设定每个局部数据节点管理 1 000 个数据文件,文件大小从 32 KB 到 64 KB 随机分布。数据节点数量增加,则文件数增加,在本文的实验中,数据节点数目从 8 开始按照 2 的幂依次增加,直到增加到 256 为止。通过 KD-R 索引与全局分布式 R 树索引的对比实验,评估 KD-R 索引的性能,实验以响应时间作为评估指标。

图 4 表示的是多维点查询情况,表明全局分布式 R 树索引在处理多维点查询时比 KD-R 树索引效率高。这是由于 R 树索引在执行查询时,时间复杂度为 lb N, 是一种高效的数据结构。但是,从图 4 可看出:随着数据节点数的增加,KD-R 索引的查询时间增长率变平缓,点查询效率在可以接受的范围内。

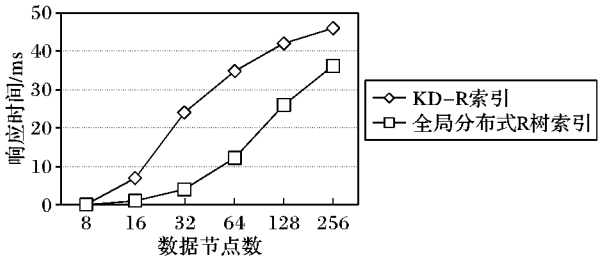


图 4　多维点查询

图 5 表示的是范围查询的情况,很明显,KD-R 索引的范围查询效率比全局分布式 R 树索引高。特别是随着数据节点数的增加,KD-R 索引的优势更突出。原因是 KD-R 索引在处理范围查询时,通过全局索引,有效降低了局部数据节点的搜索范围,同时通过多个局部数据节点上的并行查询,提高了查询效率。而在全局分布式 R 树索引中,由于索引节点的随机分布的特性,导致查询性能较差。

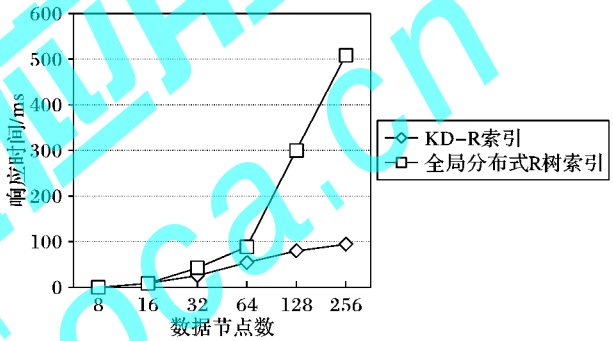


图 5　多维范围查询

可用性测试是分布式系统的重要测试指标之一。由于云存储系统是由大量廉价的计算机服务器组成,更容易出现计算机节点失效的情况,而节点失效后,会影响存储系统处理查询请求的正确率。因此,在实验中,本文针对不同规模的局部数据节点值,随机设置了 5% 的节点失效率。图 6 表明 KD-R 索引比全局分布式 R 树索引的查询成功率更高,因此,KD-R 索引具有更高的可用性。

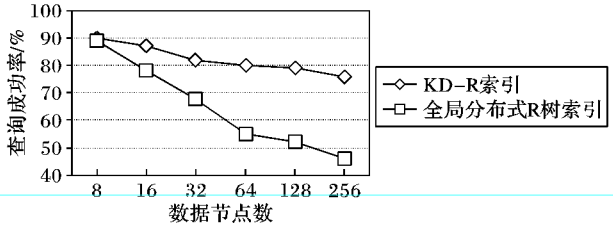


图 6　可用性测试

4　结语

本文基于 R 树和 KD 树设计了一种双层的多维云数据索引。该索引对云计算环境中的每个局部数据服务器上的数据构建 R 树索引,以实现高效的本地多维查询;同时,根据性能损耗模型,选择局部数据节点中的部分本地 R 树索引节点发布到全局索引中,全局索引按 KD 树结构进行组织。实验结果表明,KD-R 索引具有查询高效性和高可用性。然而,由于云存储系统通常要同时支持事务操作和数据分析操作,频繁的数据更新会与数据查询产生冲突,从而降低数据查询效

(下转第 3278 页)

配应该基于自然语言理解基础上才准确与公平;方法考虑因素较少,以致尚不能单独作为评定科技论文的可信与质量的指标,所得出的评估值作为一种评价体系的一部分较为合理。下一步研究仍然从科技论文内容入手,在完善此方法基础上,研究科技论文的内容可信判断方法。

参考文献:

[1] GARFIELD E. Forms for literature citations[J]. Science, 1954, 120(3129):1038 - 1041.

[2] BERGSTROM C T. Eigenfactor: measuring the value and prestige of scholarly journals[J]. College and Research Libraries News, 2007, 68(5):314 - 316.

[3] ZHOU J, ZENG G, ZENG Y. The comprehensive assessment method of trust level for scientific papers based on retrieval features[J]. Application Research of Computers, 2013, 30(3):820 - 824. (周静, 曾国荪, 曾媛. 基于检索特征的科技论文可信等级的综合评估方法[J]. 计算机应用研究, 2013, 30(3):820 - 824.)

[4] WANG Y, MA J. A literature of science and technology quality evaluation algorithm based on PageRank [J]. Journal of Guangxi Normal University: Natural Science, 2009, 27(1):165 - 168. (王向阳, 马军. 一个基于 PageRank 的科技文献质量评价算法[J]. 广西师范大学学报: 自然科学版, 2009, 27(1):165 - 168.)

[5] LIU L. The research of paper in quality evaluation based on science-paper online[D]. Changchun: Changchun University of Technology, 2011. (刘乐. "中国科技论文在线"论文质量评价研究[D]. 长春: 长春工业大学, 2011.)

[6] FENG X, ZHU P. The structure of the scientific papers and writing format [J]. Journal of Shandong Meteorology, 2005, 25(3):12 - 15. (冯晓云, 朱平盛. 科技论文的构成与编写格式[J]. 山东气象, 2005, 25(3):12 - 15.)

[7] PANG J. Research and development of Web text feature extraction method [J]. Information Studies: Theory and Application, 2006, 29(3):338 - 340. (庞景安. Web 文本特征提取方法的研究与发展[J]. 情报理论与实践, 2006, 29(3):338 - 340.)

[8] LI L. Chinese science and technology literature summarization system [D]. Chengdu: University of Electronic Science and Technology of China, 2006. (李立燕. 中文科技文献自动摘要系统 [D]. 成都: 电子科技大学, 2006.)

[9] YU L. Research and applications on text features extraction from science and technical literatures [D]. Beijing: Beijing University of Posts and Telecommunications, 2009. (于亮. 科技文献的文本特征抽取研究与应用[D]. 北京: 北京邮电大学, 2009.)

[10] LIU Q, ZHANG H, YU H, *et al.* Chinese lexical analysis based on layered hidden Markov model[J]. Journal of Computer Research and Development, 2004, 41(8):1421 - 1429. (刘群, 张华平, 俞鸿魁, 等. 基于层叠隐马模型的汉语词法分析[J]. 计算机研究与发展, 2004, 41(8): 1421 - 1429.)

[11] XIE J. Chinese keyword extraction method based on word span and application in text classification [D]. Hangzhou: Zhejiang University of Technology, 2011. (谢晋. 基于词跨度的中文文本关键词提取及在文本分类中的应用[D]: 杭州: 浙江工业大学, 2011.

[12] HUANG S, ZENG G, WANG W. Web text credibility calculation method based on trust model validation[J]. Computer Science, 2011, 38(1): 177 - 180. (黄帅彪, 曾国荪, 王伟. 基于信任模式验证的论述性 Web 文本可信性判定方法[J]. 计算机科学, 2011, 38(1): 177 - 180.)

[13] PENG J, YANG D, TANG S, *et al.* Text similarity calculation based on concept of similarity[J]. China Science F: Information Science, 2009, 39(5):534 - 544. (彭京, 杨东青, 唐世渭, 等. 基于概念相似度的文本相似计算[J]. 中国科学 F 辑: 信息科学, 2009, 39(5):534 - 544.)

(上接第 3221 页)

率,如何在保证查询效率的前提下提高数据一致性是下一步研究的内容。

参考文献:

[1] ARMBRUST M, FOX A, GRIFFITH R, *et al.* A view of cloud computing [J]. Communications of the ACM, 2010, 53(4): 50 - 58.

[2] CHANG F, DEAN J, GHEMAWAT S, *et al.* Bigtable: a distributed storage system for structured data [J]. ACM Transactions on Computer Systems, 2008, 26(2): 1 - 26.

[3] DECANDIA G, HASTORUN D, JAMPANI M, *et al.* Dynamo: Amazon's highly available key-value store [C]// Proceedings of the 21st ACM Symposium on Operating Systems Principles. New York: ACM Press, 2007: 205 - 220.

[4] LASKHMAN A, MALIK P. Cassandra: a decentralized structured storage system[J]. ACM SIGOPS Operating Systems Review, 2010, 44(2): 35 - 40.

[5] CARSTOIU D, CERNIAN A, OLTEANU A. Hadoop HBase-0.20.2 performance evaluation[C]// Proceedings of the 4th International Conference on New Trends in Information Science and Service Science. Piscataway: IEEE Press, 2010: 84 - 87.

[6] FRANKE C, MORIN S, CHEBOTKO A, *et al.* Distributed semantic Web data management in HBase and MySQL cluster [C]// Proceedings of the 2011 IEEE International Conference on Cloud Computing. Piscataway: IEEE Press, 2011: 105 - 112.

[7] DEAN J, GHEMAWAT S. MapReduce: simplified data processing on large clusters [C]// Proceedings of the 6th Symposium on Operating System Design and Implementation. Berkeley: USENIX, 2004: 137 - 150.

[8] AGUILERA M K, GOLAB W, SHAH M A. A practical scalable distributed B-tree [J]. Proceedings of the VLDB Endowment, 2008, 1(1): 598 - 609.

[9] WU S, JIANG D, OOI B C, *et al.* Efficient B-tree based indexing for cloud data processing[J]. Proceedings of the VLDB Endowment, 2010, 3(1/2): 1207 - 1218.

[10] WANG J, WU S, GAO H, *et al.* Indexing multi-dimensional data in a cloud system [C]// Proceedings of the 2010 ACM SIGMOD International Conference on Management of data. New York: ACM Press, 2010: 591 - 602.

[11] DING L, QIAO B, WANG G, *et al.* An efficient quad-tree based index structure for cloud data management [C]// Proceedings of the 12th International Conference on Web-Age Information Management, LNCS 6897. Berlin: Springer-Verlag, 2010: 238 - 250.

[12] BHANDARKAR M. Hadoop: a view from the trenches [C]// Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM Press, 2013: 1138 - 1138.

[13] GAEDE V, GUNTHER O. Multidimensional access methods[J]. ACM Computing Surveys, 1998, 30(2): 170 - 231.

[14] LO Y, TAN C. Evaluation on multi-attribute indexing for cloud databases [J]. International Journal of Advanced Information Technologies, 2012, 6(2):215 - 222.

[15] MONTRESOR A, JELASITY M. PeerSim: a scalable P2P simulator [C]// Proceedings of the 9th IEEE International Conference on Peer-to-Peer Computing. Piscataway: IEEE Press, 2009: 99 - 100.