doi:10.3799/dqkx.2010.057

GSHR-Tree:一种基于动态空间槽和哈希表的网格环境下的空间索引树

陈占龙[1,2],吴信才[1],谢 忠[1*],马丽娜[1,2]

1. 中国地质大学信息工程学院,湖北武汉 430074
2. 地理信息系统软件及其应用教育部工程中心,湖北武汉 430074

摘要:为提高网格环境下海量空间数据管理与并行化处理效率,将网格环境下的分布并行处理技术与空间索引相融合,提出了一种空间索引框架(grid slot and hash R tree,GSHR-Tree).该索引树结构基于散列hash表和动态空间槽,结合R树结构的范围查询优势和哈希表结构的高效单key查询,分析改进了索引结构的组织和存储.构造了适合于大规模空间数据的网格并行空间计算的索引结构,该索引树算法根据空间数据划分策略,动态分割空间槽,并将它们映射到多个节点机上.每个节点机再将其对应空间槽中的空间对象组织成R树,以大节点R树方式在多个节点上分布索引数据.以空间范围查询并行处理的系统响应时间为性能评估指标,通过模拟实验证明,该GSHR-Tree索引满足了当前网格环境空间索引的需要,并具有设计合理、性能高效的特点.

关键词:空间数据索引;分布式空间索引;R-树;散列hash表;动态空间槽;网格计算;地理信息系统.

中图分类号:TP311 **文章编号**:1000-2383(2010)03-0463-08 **收稿日期**:2010-01-15

GSHR-Tree: A Spatial Index Tree Based on Dynamic Spatial Slot and Hash Table in Grid Environments

CHEN Zhan-long[1,2], WU Xin-cai[1], XIE Zhong[1*], MA Li-na[1,2]

1. *Faculty of Information Engineering, China University of Geosciences, Wuhan* 430074, *China*
2. *Engineering Research Center for GIS Software and Applications, Ministry of Education, Wuhan* 430074, *China*

Abstract: In order to improve the efficiency of parallel processing of a spatial mass data under the distributed parallel computing grid environment, this paper presents a new grid slot hash parallel spatial index GSHR-Tree structure established with the parallel spatial indexing mechanism. Based on the hash table and dynamic spatial slot, we have improved the structure of the classical parallel R-tree index. The GSHR-Tree index makes full use of the good qualities of R-Tree and hash data structure. A new parallel spatial index is constructed to meet the needs of parallel grid computing about the magnanimous spatial data in the distributed network. This arithmetic splits space into multi-slots by multiplying and reverting and maps these slots to sites in distributed and parallel system. Each site constructs the spatial objects in its spatial slot into an R-tree. On the basis of this tree structure, the index data is distributed among multiple nodes in the grid networks by using large node R-tree method. Instead of spatial object's recursive comparison where original R-tree has been used, the algorithm builds the spatial index by applying binary code operation in which computer runs more efficiently, and extends dynamic hash code for bit comparison, using the system response time of the parallel processing of spatial scope query algorithm as the performance evaluation factor. The result of the simulated the experiments shows GSHR-Tree is performed to prove the reasonable design and the high performance of the indexing structure presented in the paper.

Key words: geospatial data index; distributed parallel index; R-Tree; hash index; dynamic spatial slot; grid computing; geographic information system (GIS).

基金项目:国家重点"863"项目(No. 2007AA120503);中央高校基本科研业务费专项资金(No. CUGL090251);国家自然科学基金(No. 40771165).

作者简介:陈占龙(1980-),男,博士,讲师,主要从事地理信息系统研究与应用. * 通讯作者:谢忠,E-mail: xiezhong68@gmail.com

随着地理信息系统(geographic information system,GIS)处理空间数据规模的急剧增加和计算机网络技术的迅速发展,海量空间数据势必要分布存储在网络环境下的各个服务器中,分布存储和分布式处理计算变得越来越重要.随着网格计算技术在 GIS 中的应用逐渐成为研究热点,其分布式并行计算模式与系统架构将有助于提高 GIS 的整体性能和运行效率(吴信才和吴亮,2006).因此,分布式并行计算将成为解决传统 GIS 计算能力不足问题的重要方法.空间数据索引机制是分布式空间数据处理计算的基础,研究分布式环境下高效的空间数据检索机制必然成为趋势.近年来对分布式索引的研究引起了国内外学者的广泛关注,已有的研究成果包括:Hu *et al*.(2005)提出的用于在网格环境下定位 GIS 资源的 SR-Tree 索引生成算法;Miyazaki *et al*.(2004)设计的实现索引结点拷贝和索引数据快速回取的 Fat-Btree;Fornari and Iochpe(2004)提出了用以实现动态索引数据的分配动态 hash 联结方法;郭鹏等(2005)提出了基于 R-tree 的适用于 P2P 环境的多维空间索引结构 PR-tree;唐继勇等(2005)研究了涉及索引数据分配、索引复制策略、索引数据迁移和重构等方面的 DPB+Tree 索引结构.在这些索引树的研究过程中,其中以对具有良好搜索性能的 R 树的研究尤为活跃(罗忠文,2002;谢忠等,2006),网格环境下对 R 树的改进主要有:左朝树等(2006)提出了一种 DPSlR$^+$ 树;du Mouza *et al*.(2007)提出了分布式环境中的海量空间索引框架 SD-R 树并对其存储负载均衡和消息处理进行了试验测试.这些新的分布式并行索引树大多存在更新开销大、并行化程度低、副本采用后冗余太多等性能问题.本文研究了网络环境下分布式索引技术实现的关键技术,提出一种基于动态空间槽和哈希表的网格环境下的空间索引树,该分布式并行索引树结构综合考虑了以上问题,力求满足对网格环境下海量空间数据的查询分析运算的性能要求.

本文首先给出了网格环境下的空间索引(grid slot and hash R tree,GSHR-Tree)的框架,分析了 GSHR-Tree 所涉及的概念并进行了解释,并对 GSHR-Tree 索引树进行了严格定义.然后给出 GSHR-Tree 在网格环境下的结构,以便对 GSHR-Tree 索引树插入删除算法进行形式化描述.详细介绍了 GSHR-Tree 的插入删除算法以及利用 hash 表的处理,对 GSHR-Tree 的生成、查找节点算法进行了详细分析,同时对本文提出的算法进行了分析.

1 GSHR-Tree 的提出

首先对空间槽概念进行解释.在数据库领域,可以将空间数据划分成各个空间槽,空间槽划分技术是解决划分后的数据块在各空间数据服务器节点之间分布不均衡问题(即数据倾斜现象)的有效方法.同样,空间数据划分技术在空间数据库中也发挥着重要作用,在避免产生数据倾斜的前提下,可提高空间数据的并行查询与检索效率.空间槽是指在 n 维线性空间中由 $d_i<x_i<u_i$,x_i 为 n 维线性空间中的第 i 维,且 d_i 和 u_i 满足($-\infty=<d_i,u_i<=+\infty$),由 x_i 围成的一个 n 维线性空间中的域 D 称为 n 维线性空间中的一个空间槽.用 SS 表示一个空间槽(du Mouza *et al*., 2007).

空间槽的划分方法:为了满足空间索引在网格环境下并行查询的需要和空间数据划分策略的需求,采用赵春宇等(2006)中基于 Hilbert 空间填充曲线的空间数据划分算法(spatial data partitioning based on Hilbert curve,HCSDP).在充分考虑空间信息的海量特征以及矢量数据存储记录的不定长等特点的前提下,该算法可实现并行空间数据库中海量空间数据记录在多个存储设备上的均衡划分,以避免出现数据倾斜现象,从而提高了空间数据的检索与查询效率.利用 Hilbert 空间填充曲线的空间数据划分算法,可以将空间数据库集群中的空间数据划分成各个空间槽,存储在各个服务器节点中,设计了基于动态空间槽和哈希表的网格环境下的空间索引框架(GSHR-Tree).

本文关注于海量空间数据实体的检索,每个空间实体由唯一的对象实体 ID 和最小外包矩形 mbb 标识.GSHR-Tree 继承于经典的空间索引 R 树,使得 R 树适用于网格环境下的空间实体的分布.提出了一种网格环境下的空间索引树结构,建立了相应的索引管理策略,其中索引管理策略包括索引的旋转平衡策略、索引数据平衡策略、索引的存储均衡策略等.该框架结构如图 1 所示.

GSHR-Tree 的设计遵循了以下原则:

(1)没有利用集中式目录管理来进行空间数据的检索;(2)服务器节点可以按需动态加入到 GSHR-Tree 中;(3)客户端可能通过过期的映像来存取 GSHR-Tree 中的空间实体.

GSHR-Tree 是一种动态平衡的二叉空间索引树,GSHR-Tree 中的每一个服务器节点中,都记录

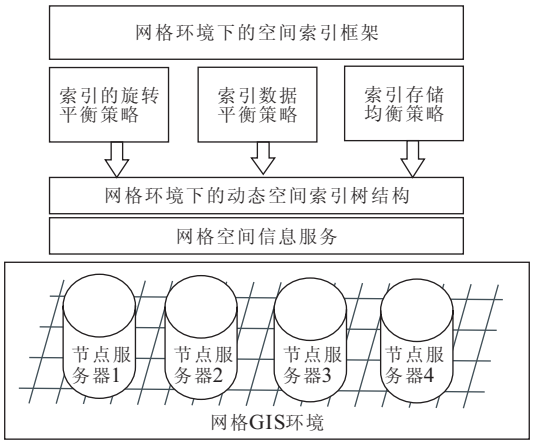

图 1　网格环境下的动态空间数据索引框架

Fig. 1　The dynamic geospatial data index framework in grid environment

了该服务器节点的 ID 和最小外包矩形 MBB. GSHR-Tree 被部署在网格环境中的各个服务器节点上. 如果一个服务器节点溢出了，该服务器节点就需要分裂，分裂出的空间实体被加入到 GSHR-Tree 的新服务器节点上.

应用程序通过客户端组件来存取由 GSHR-Tree 索引的空间实体，网格环境下空间实体的存取并不需要集中式的组件或者多播消息. 客户端组件通过存储在客户端的服务器节点的映像文件来选择合适的服务器节点，一些已经存在的服务器节点可能不在客户端的映像文件中，这些服务器节点可能是其他服务器节点分裂的时候动态加入到 GSHR-Tree 中的新服务器节点，在这种情况下，查询操作可能选择一个错误的服务器节点，接收到查询操作的服务器节点认识到这个查询错误，并把该查询操作转发到其他服务器节点，直到该查询操作到达正确的服务器节点.

如果该服务器节点的空间实体对象的最小外包矩形符合查询条件，则该节点将利用 R 树索引的查询算法进行空间实体的查询，否则需要对 GSHR-Tree 进行向上遍历，直到有服务器节点的最小外包矩形符合查询条件为止. 也可能有这样的情况，即没有找到合适的服务器节点，这在查询中是没有成功查询到结果的情况.

在各个服务器节点空间数据的检索过程中，通过 R 树索引对空数据进行搜索操作，得到了索引的结果集. 为提高 R 树自身的存取效率，经典的 R 树的设计使单个数据页面聚集尽可能多的记录项，索引的结果集中只包括对空间数据的连接指针，即 R 树的节点中只存放了空间对象的标志符，真正的空间数据还需要再次获取，R 树的查询操作还没有完成. 因此，基于 R 树索引的查询操作可以明显分为两个阶段：一是产生索引结果集的阶段；二是产生最终的查询结果集阶段. 两个阶段都有制约查询效率的不同特点，因而有不同的优化策略，但根本方法都是减少磁盘读取次数. 经典的 R 树在构造时使用的聚类算法只能保证对 R 树本身存储磁盘页面的存取性能得到优化，即取得索引结果集的性能可以得到优化. 但根据索引结果集直接去获取最终查询结果集，性能并不是很高，在实际的实现中也成为了 R 树性能的瓶颈. 根据索引结果集取得查询结果集，需要考虑实际数据页面的读取效率，通过最少的磁盘读取次数获得集合中的所有数据. R 树索引结果集是从空间聚类中取得的结果，其对应的结果数据对存储来说并不聚集，空间实体的聚集是存储空间上的聚集.

为了克服经典 R 树在实际应用中的弊端，GSHR-Tree 的服务器节点索引的叶子结点采用 hash 结构来存储数据桶的桶号，在实际存取空间数据时对索引结果集按照桶号进行空间数据聚集方向进行排序，以减少磁盘的 I/O 次数，提高空间数据的检索效率.

提出 GSHR-Tree 的定义如下：

设索引框架的每个结点用结点标识符 i 来标识，结点 i 的高度为 $G(i)$. 当结点 i 是结点 j 的父亲时，$P(j)=i$，并且 $G(j)=G(i)+1$. 对于根结点 r，则 $G(r)=1$. 设 S_i 为网格环境下的各个服务器节点机的集合，Hash(key)为一选定的 hash 函数. GSHR-Tree 树结构满足以下条件：

(1)如果 $i=P(j)$，则 $S_i \supseteq S_j$；

(2)S_i 以动态平衡的二叉空间索引树的组织形式分布在网格环境中，可以按需动态加入到二叉空间索引树中；

(3)如果 $G(i)=H$ 树服务器节点 i 以 R 树索引空间实体，且 R 树索引的叶子节点以 hash 方式存储，对应的 hash 函数为 Hash(key)，则称为 GSHR-Tree.

2　GSHR-Tree 框架的结构

GSHR-Tree 网格环境下的各个服务器节点中的各种节点定义如表1. GSHR-Tree的结构如图2

表 1 GSHR-Tree 节点定义

Table 1 The definition of DR-H index node

GSHR-Tree Hash 表节点的结构定义	GSHR-Tree 叶节点的定义	GSHR-Tree 路由节点的定义

struct R-Entry{
int m_i0id;//对象 ID
Rect m_rect;;//外包矩形
R-Entry * link;};
Typedef R-Entry * R-EntryPtr;
struct Menties{int m_iOid;//对象 ID
Rect m_rect;}
	struct LefNode{
	int PagNo;//本节点页号
	int ParPNo;//父节点页号
	short nIndex;//父节点下标
	int count;//指针个数
	int sytle;//数据库类型
	R-EntryPtr[MAXLN];}
		struct MidNode{
		int PagNo;//本节点页号
		int ParPNo;//父节点页号
		short nIndex;//父节点下标
		int count;//索引项数
		int Menties[MAXMN];//索引项数组}

注:MAXLN 定义为:#define MAXLN(int)((PGSIZE−(6 * sizeof(int)))/sizeof(struct LefNode));MAXMN 定义为:#define MAXCARD(int)((PGSIZE−(6 * sizeof(int)))/sizeof(struct MidNode)).

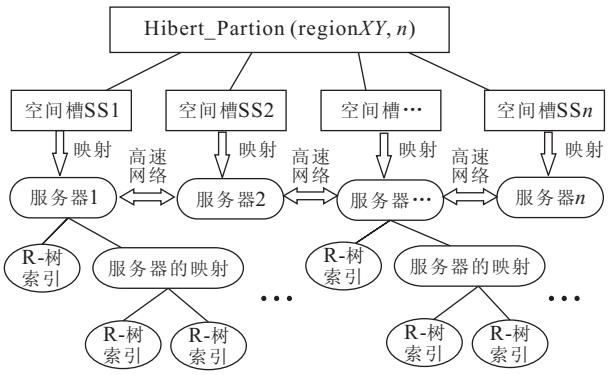

图 2 GSHR-Tree 的结构

Fig. 2 The structure of GSHR-Tree

所示,首先利用空间数据划分算法将空间对象映射到空间槽 SSi 中,空间槽 SSi 对应于网格环境下的一个节点服务器,为了避免传统的 R 树索引在根节点访问的负载瓶颈,服务器 i 将其对应的空间槽中的空间对象组织成平衡二叉空间索引树,二叉空间索引树的每个节点的空间数据以经典的 R 树索引进行组织存储,并通过 R 树分解函数将 R 树分解成多个子 R 树分别存入一个本地磁盘中.

在网格环境下的每个节点服务器中,包含由数据节点和路由节点组成的平衡二叉索引树,其中路由节点有左右路由孩子节点组成,路由节点的外包矩形记录了其左右两个孩子节点的最小外包矩形. 每一个数据节点的空间数据以经典的 R 树索引进行组织存储. 拥有 n 个服务器节点的 GSHR-Tree 索引将由 n 个叶子数据节点和 n-1 个中间路由节点组成. 在网格环境中的每一个服务器节点 S_i 由二元组$<r_i, d_i>$表示,r_i 表示路由节点,d_i 表示数据节点. 路由节点和数据节点的最小外包矩形组成了该服务器节点 S_i 的外包矩形.

GSHR-Tree 逻辑上是一种二叉树索引,映射为网格环境下服务器节点的集合. GSHR-Tree 的每一个内部节点都有两个孩子节点,且孩子节点的高度最多相差 1. 这就保证了 GSHR-Tree 的高度为节点服务器的对数. 路由节点维护了其左右两个孩子节点的目录矩形,目录矩形为左右两个孩子节点的最小外包矩形. 每一个叶节点存储了由经典的 R 树索引的空间实体. 在具有 n 个服务器节点的网络中,GSHR-Tree 有 n 个叶节点和 n-1 个中间节点. 在 GSHR-Tree 中,每个节点的大小恒定为 Pagesize,Pagesize 可根据实际情况在 4K 字节、1K 字节、512 字节和 256 字节中选择.

其中,叶节点对单个服务器节点的索引树进行了描述. 特别地,路由节点的目录矩形为其左右两个孩子的目录矩形的几何并集,路由节点的高为 Max(left. height, right. height)+1.

3 GSHR-Tree 算法

3.1 GSHR-Tree 的生成

图 3 展示了 GSHR-Tree 的生成过程:最初只有一个数据节点 d_0 存储在 S_0 上. 在第一次分裂之后,新的服务器节点 S_1 被添加到 GSHR-Tree 中,在 S_1 中存储了(r_1, d_1),其中 r_1 是路由节点,d_1 是数据节点. 空间实体被部署在这两个服务器节点中,每个服务器节点(d_0, d_1)中的空间实体的组织都遵从于基于外包矩形约束的改进的 R 树索引. r_1 的目录矩形为 a,d_0 和 d_1 的目录矩形分别为 b 和 c,满足 $a=\mathrm{mbb}(b \cup c)$,目录矩形 a、b 和 c 存储在 r_1 中的目的是指引 GSHR-Tree 的插入和删除操作. 随着空间实体中的不断插入,假设 S_1 必须分裂,目录矩形 c 必须再次分裂,存储在 d_1 中的空间实体被重新部署在 S_1 和新添加的服务器节点 S_2 中,新的路由节

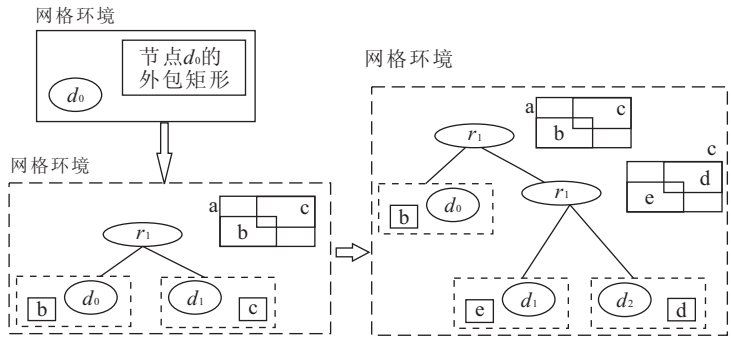

图 3　GSHR-Tree 的生成过程

Fig. 3　Generation process for GSHR-Tree

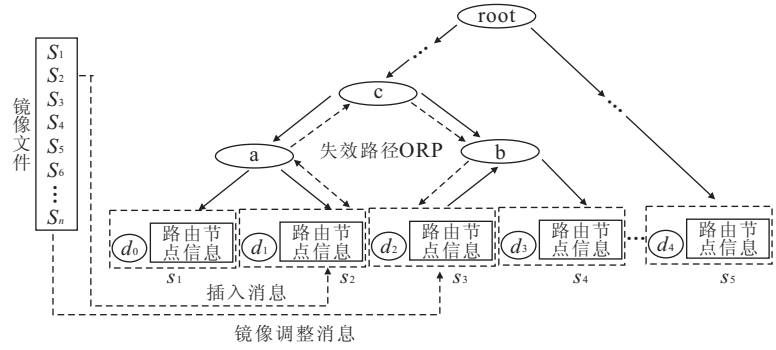

图 4　GSHR-Tree 服务器节点的查找过程

Fig. 4　The search process for GSHR-Tree servers nodes

点 r_2 和数据节点 d_2 存储在服务器节点 S_2 中，r_2 中存储了目录矩形 c 和 r_2 中的左右两个孩子节点的外包矩形 d 和 e，且满足 $c=\mathrm{mbb}(d\cup e)$. 因此每一个节点的矩形表现在两层节点：在节点本身和节点的父亲节点上. 路由节点维护了其父亲节点的 id 和其左右两个孩子节点的链表，链表是一个四元组 (id，dr，height，type)，其中 id 为存储该节点的服务器节点的 id，dr 为该节点的目录矩形，height 为该节点的高度，type 表示该节点是数据节点还是路由节点.

3.2　GSHR-Tree 插入算法

GSHR-Tree 插入算法的基本思路是找到合适的叶子结点，在其中插入索引项，并由此可能产生结点的分裂. 其中查找合适节点的过程如下(图 4)：

刚开始，客户端 C 的镜像文件为空，由客户端 C 发起的第一次插入查询操作将发给客户端依附的服务器节点 S. 第一次插入操作很可能使得服务器节点 S 范围溢出(即节点 S 的目录矩形不满足查询条件)，范围溢出使得节点 S 的发起从该节点到其祖先节点的递归遍历直到找到其祖先节点的目录矩形满足查询条件，最坏的情况直到遍历到 GSHR-Tree 的根节点. 在遍历和插入过程中，遍历过的节点的链表都加入到镜像调整消息中，该镜像调整消息最终被插入空间实体的节点并发送给客户端 C，使得客户端 C 初始化其原本为空的镜像文件. 节点的分裂和新的服务器节点的加入将导致客户端镜像文件的过期. 最坏的情况：客户端向服务器节点发送查询操作，服务器节点的范围溢出消息将触发一连串的不成功的消息，递归遍历直到 GSHR-Tree 的根节点，向上遍历过程花费了 $\log N$ 条消息，而从向上遍历的节点向下找到合适的插入节点需要 $\log N$ 条消息. 最后，如果插入空间实体的节点发生了分裂，需要另一个从底向上的遍历一直到 GSHR-Tree 的根来调整遍历路径上的节点的高度. 该分裂可能导致其他网络中的其他节点服务器的加入，从而导致整棵树的调整. GSHR-Tree 非叶节点的插入与经典 R-树相同，只是在叶节点需要特殊处理. 具体插入算法过程如下：

算法 GSHR-Insert：

插入一个新索引项 E 到 GSHR-Tree 中；

输入：T：树的根结点；E：待插入的项；

输出：无；

I1:调用 ChooseLeaf 来选择一个合适的叶子结点 L 以容纳索引项 E;

I2:若 L 中还能容纳 E,调用 InsertLNod 加入叶节点,否则调用 GSHR-SplitNode 来获得两个结点 L 和 LL,它们包含了 E 和 L 中原有的所有索引项;

I3:若产生了结点分裂,调用 GSHR-Adjus,传递参数 L,LL;

I4:若结点分裂向上传播导致根结点的分裂,则需要新的服务器节点加入;

I5:对调整生成的两个叶节点 L,LL 调用 HashLNode 进行 Hash 处理.

在 GSHR-Tree 的插入算法中,特殊之处就是对叶节点插入处理以及分裂的处理,即调用了 InsertLNod 函数,该函数的基本思想是:当在叶节点插入新索引项 E 时,根据空间数据的对象 m__i0id,使用散列函数 Hash(m__i0id),计算得到桶号 R-EntryPtr[i],然后将 R-Entry 插入到 R-EntryPtr[i]的链列表中,在服务器节点 S_i 达到其负载需要分裂时,新的服务器节点 $S_i+1(r_i+1,d_i+1)$ 加入到 GSHR-Tree 索引框架中,r_i+1 为服务器节点 S_i+1 的路由节点,d_i+1 为服务器节点 S_i+1 的数据节点,则利用经典的 R 树的分类插入算法将需要分裂的空间对象分别插入到节点服务器 S_i 和 S_i+1 中,同时 S_i 的路由节点中记录了两个服务器节点的最小外包矩形,同时将 S_i+1 作为 S_i 的子节点插入到 GSHR-Tree 中.

3.3 GSHR-Tree 删除算法

叶节点中给定矩形的删除是空间索引树动态维护的主要操作之一,准确地查找目标矩形是删除操作的前提. 与匹配操作不同的是,确切查找的结果只能是唯一的矩形(或者找不到). 为了提高删除矩形的效率,笔者应用 Hash 索引技术对 GSHR-Tree 的叶结点的组织方式进行改进. 蒋夏军等(2006)的思想是坐标空间中的矩形除了其自身的长、宽信息外,另一类重要信息是其位置信息. 利用矩形的这些信息,采用 Hash 索引,用它来辅助矩形的确切查找. 设矩形的长和宽分别为 a 和 b,矩形的中心在空间中的坐标为(x,y),P,Q 为常量,数组 $A[P,Q]$为指针数组,数组元素为指向存储桶(bucket)的指针,P、Q 的意义为将空间的 x 方向的长度(设为 LX)P 等分;将所有矩形外接圆半径的最大可能值(设为LR)Q等分. 设 $i=\mathrm{int}(x/(Lx/P))$,$j=\mathrm{int}(\sqrt{a^2+b^2}/2)/(Lp/Q)$. 则该矩形的指针存放在 $A\{[i,J]\}$所指向的存储桶中,存储桶可能存放不止一个矩形的指针. 运用上述 Hash 法既考虑了矩形的形状信息,同时又考虑了它的位置信息,因而使矩形尽量分散在不同的存储桶中.

GSHR-Tree 删除算法的基本思路是找到合适的叶子结点,利用上述的 hash 方法 Hash(m__i0id)计算出桶号在 R-EntryPtr[i]的链列表中进行删除.

算法 GSHR-Delete:从 GSHR-Tree 中删除索引项 E

输入:T:树的根结点;E:将被删除的索引项;

输出:无;

Il:调用 FindLeaf 来寻找存放 E 的叶子结点 L;若没有找到则停止;

I2:调用 DR-HdeleLNod 和 Hash(m__i0id)从 L 中删去 E;

I3:调用 DR-HCondenseTree,传递参数 L;

I4:若根结点只有一个孩子,则让该孩子结点成为新根.

3.4 GSHR-Tree 查询算法

本文以窗口查询为例,GSHR-Tree 的客户端以窗口 SR 向网格环境中的服务器节点 S_i 中发送消息,给定查找矩形区域 SR,查找所有与 SR 重叠的空间目标,需要先用经典的 R-查找算法找到与 SR 重叠的 GSHR-Tree 的叶节点,然后对叶节点的各个桶链进行搜索,最后得到查询结果. 但因 GSHR-Tree 叶节点用 Hash 表结构进行存储,考虑了空间数据的实际存储,减少了磁盘的 I/O 次数,提高了 GSHR-Tree 树的整体查询性能.

算法 GSHR-Wquery:从 GSHR-Tree 查询实体;

输入:(W:SR);

输出:与 W 相交的空间对象实体集;

Il:调用 ChooseServer(W)来寻找目标服务器节点 S_i;

I2:判断 S_i 上的数据节点 node 是否与 W 相交;

I3:如果 S_i 上的数据节点 node 不与 W 相交则向上搜索其父亲节点 S_i;

I4:若 S_i 与 W 相交,则用经典的 R 树算法搜索 S_j 的数据节点和子结点,并将数据返回;

I4:若 S_i 与 W 不相交,则返回 I3,继续搜索 S_i 的父亲节点直到 GSHR-Tree 的根.

查找到的服务器节点的数量取决于查询范围

SR，一旦包含 SR 的服务器节点找到，该服务器节点将向网络中的其他服务器节点广播消息，这些广播消息的最大条数为 $O(\log N)$，N 为网络中服务器节点的数量.

4　性能分析

为测试 GSHR-Tree 空间查询性能的优化情况，在此采用 C 语言实现分布式并行空间索引子系统. 在实践中遇到了通过 R-树索引来求取索引结果集的性能很高，但通过索引结果集来求取查询结果集的性能很低的矛盾. 空间索引树理论模型与实际应用实现中的性能差距是解决该问题的关键，通过分析出实际应用中制约 R-树索引性能的主要因素，可以得到解决这一矛盾的途径是在通过索引结果集来求取查询结果集的阶段做针对磁盘读取的优化. GSHR-Tree 即是解决这一问题的关键，GSHR-Tree 的叶子节点的存储结构充分考虑了空间数据的物理磁盘存储，由于 GSHR-Tree 通过对服务器节点上 R 树索引的改进，叶子结点使用 hash 结构和溢出结点改善了更新的效率，叶子节点的存储结构充分考虑了空间数据的物理磁盘存储，与经典分布式空间索引树相比，有以下特性(图 5)：

(1)没有利用集中式目录管理来进行空间数据的检索，考虑靠近索引根服务器节点的负载问题，而该问题是设计分布式索引树的关键，GSHR-Tree 解决了分布式索引越靠近根的服务器节点其接收到的查询消息越多、负载越大的情况. 在最坏的情况，所有的查询消息都要先经过根节点所在的服务器节点，这在实际应用过程中是不可以接受的.

(2)服务器节点可以按需动态加入到 GSHR-Tree 中，客户端可能通过过期的映像来存取 GSHR-Tree 中的空间实体.

(3)在每个服务器节点的索引树的设计中，叶子结点使用 hash 结构和溢出结点改善了更新的效率，经典的 R 树结点的查找必须不断的对 key 值进行比较，而叶子结点采用了 hash 之后，可使得查询速度进一步提高(hash 函数在记录的存储位置和已知的关键值之间建立了一个确定的对应关系). 叶子结点拆分的几率减小，那么整个树结构的改变几率也相应减小，从而使得整个的系统的更新开销减小.

(4)较高的查询效率，由于 GSHR-Tree 框架的设计充分利用了 R 树的范围查询和散列 hash 表的高效单 key 查询的优点，所以对范围查询和点查询都具有较好的查询效率.

(5)支持多种空间位置的查询，优化了删除算法，设计了批量删除算法：支持通过实体 ID 查询范围，delete 和 update；采用面向对象的设计，使结构更加清晰.

(6)采用多线程来优化内外存交换的部分. 因为 I/O 操作非常费时，而且 R 树的效率很低，因为 R 树的查询是按范围而不是按 ID 的大小进行的，加入了一个专门负责 I/O 的线程很大程度上提高了效率.

笔者利用大量的空间实体对 GSHR-Tree 的性能进行了实验分析，本节对 GSHR-Tree 框架进行了插入空间实体的分析. 笔者首先利用 5 000 个空间实体对 GSHR-Tree 进行初始化，在对 GSHR-Tree 的初始化避免了由于刚开始只向一个服务器节点插入时的数据偏差，因为只向单个服务器节点插入时，各种消息的耗费几乎为 0. 基于服务器节点接收到的消息的条数和服务器节点之间的负载均衡，对空间实体的插入进行了分析，图 5 展示了在初

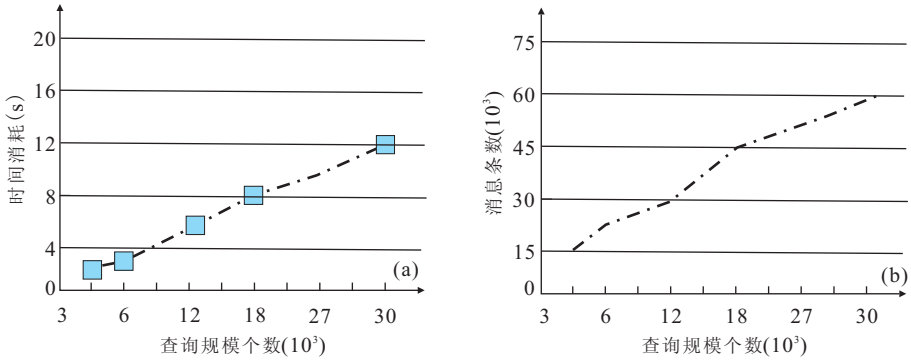

图 5　GSHR-Tree 存取的性能分析

Fig. 5　Performance analysis of GSHR-Tree access

始化基础上插入空间实体的个数与各个服务器节点收到的消息条数和耗时之间的关系.

5 结语

空间查询速度是空间数据库发展应用的关键.空间索引对空间查询的影响极大.本文将空间索引与网格环境下的并行处理技术融合在一起,研究了利用分布式并行技术建立网络环境下海量空间数据的大规模索引机制的并发处理,并借助于 hash 表提高性能.设计了顾及空间数据的物理存储的 GSHR-Tree 索引树结构.实验测试证明,该索引树结构能降低空间数据的提取时间.通过插入空间实体消耗的时间和消息个数的实验测试表明 GSHR-Tree 具有良好的查询性能. GSHR-Tree 索引树的进一步优化、提高网格环境下索引系统的可靠性将是本文后续工作的研究重点.

References

du Mouza, C., Litwin, W., Rigaux, P., 2007. SD-Tree: a scalable distributed Rtree. In: ICDE, Paris, France, 296－305.

Fornari, R. M., Iochpe, C., 2004. A spatial hash join algorithm suited for small buffer size. Proceedings of the 12th annual ACM International workshop on geographic information systems. Washington, D. C., U. S. A.. 118－126.

Guo, P., Wang, B., Wang, G. R., et al., 2005. PR-Tree: a multidimensional distributed index for peer-to-peer systems. *Journal of Huazhong University of Science and Technology (Nature Science Edition)*, 33 (Suppl.): 221－225 (in Chinese with English abstract).

Hu, H. B., Li, J., Chen, Y. H., 2005. The supplementary R-Tree (SRT) algorithm used for GIS resources allocation in model base system under grid environment. 2005 IEEE International Geoscience and Remote Sensing Symposium, 2: 4. Seoul Korea.

Jiang, X. J., Wu, H. Z., Li, W. Q., 2006. R-tree method of matching algorithm for data distribution management. *Journal of Computer Research and Development*, 43 (2): 362－367 (in Chinese with English abstract).

Luo, Z. W., 2002. Using ORDBMS to store GIS data. *Earth Science—Journal of China University of Geosciences*, 27(3): 267－270 (in Chinese with English abstract).

Miyazaki, J., Abe, Y., Yokota, H., 2004. Availabilities and costs of reliable Fat-Btrees. Proceedings of the 10th IEEE Pacific Rim International symposium on dependable computing (PRDC'04). Washington, D. C., U. S. A..

Tang, J. Y., Bai, X. Y., Yang, F., et al., 2005. Index replication strategy study based on DPB＋Tree. *Computer Science*, 32(11): 112－114 (in Chinese with English abstract).

Wu, X. C., Wu, L., 2006. Service-oriented distributed spatial information supporting system. *Earth Science—Journal of China University of Geosciences*, 31(5): 585－589 (in Chinese with English abstract).

Xie, Z., Feng, M., Ma, C. J., 2006. Index strategies for embedded-GIS spatial data management. *Earth Science—Journal of China University of Geosciences*, 31(5): 653－658 (in Chinese with English abstract).

Zhao, C. Y., Meng, L. K., Lin, Z. Y., 2006. Spatial data partitioning towards parallel spatial database system. *Geomatics and Information Science of Wuhan University*, 31(11): 962－965 (in Chinese with English abstract).

Zuo, C. S., Liu, X. S., Chen, X. H., et al., 2006. DPSlR$^+$: a distributed and parallel spatial index tree based ondynamic spatial slot. *Computer Science*, 33(2): 121－125 (in Chinese with English abstract).

附中文参考文献

郭鹏,王斌,王国仁,等,2005. PR-tree: P2P 环境下一种多维数据的分布式索引结构.华中科技大学学报(自然科学版),33(增刊):221－225.

蒋夏军,吴慧中,李蔚清,2006.数据分发管理匹配算法的 R-树实现.计算机研究与发展,43(2):362－367.

罗忠文,2002.应用对象关系型数据库存储 GIS 数据.地球科学——中国地质大学学报,27(3):267－270.

唐继勇,白新跃,杨峰,等,2005.基于 DPB＋Tree 的索引复制策略研究.计算机科学,32(11):112－114.

吴信才,吴亮,2006.面向服务的分布式空间信息支撑平台.地球科学——中国地质大学学报,31(5):585－589.

谢忠,凤鸣,马常杰,2006.嵌入式空间索引策略.地球科学——中国地质大学学报,31(5):653－658.

赵春宇,孟令奎,林志勇,2006.一种面向并行空间数据库的数据划分算法研究.武汉大学学报(信息科学版),31(11):962－965.

左朝树,刘心松,陈小辉,等,2006. DPSlR$^+$:一种基于动态空间槽的分布式并行空间索引树.计算机科学,33(2):121－125.