第 35 卷　第 2 期　　计　算　机　工　程　　2009 年 1 月
Vol.35　No.2　　Computer Engineering　　January 2009
·软件技术与数据库·　文章编号：1000—3428(2009)02—0068—02　文献标识码：A　中图分类号：TP311.13

基于分区技术的静态 R 树索引并行计算技术

周　芹[1,2]，钟耳顺[1]，黄耀欢[3]
(1. 中国科学院地理科学与资源研究所，北京 100101；2. 中国科学院研究生院，北京 100039；3. 中国水利水电科学研究院，北京 100044)

摘　要：海量空间数据静态 R 树索引的加载时耗很大。该文利用关系数据库的优势，以空间数据分区存储技术为基础，提出针对自上而下的贪婪分裂算法的静态 R 树并行加载方法。该方法提高了海量数据批量加载效率，支持分区粒度的索引重建。论证与实验结果表明，并行构建的 R 树在合理空间数据分区下可以获得更高查询效率。

关键词：空间索引；静态 R 树；分区；并行计算

Parallel Computation Technique for Static R-tree Index Based on Partition Technology

ZHOU Qin[1,2], ZHONG Er-shun[1], HUANG Yao-huan[3]
(1. Institute of Geographic Sciences and Natural Resources Research, Chinese Academy of Sciences, Beijing 100101;
2. Graduate University of Chinese Academy of Sciences, Beijing 100039;
3. China Institute of Water Resources and Hydropower Research, Beijing 100044)

【Abstract】Bulk-loading of static R-tree index for massive spatial data is time consuming. This paper utilizes the advantage of relational database. Aiming at the Top-down Greedy-Split(TGS) algorithm, it proposes parallel bulk-loading method of static R-tree based on the storage technology of spatial data. This method accelerates the mass data bulk loading efficient, and supports the index rebuild of partition grading. Argumentation and experimental results show that the parallel built R-tree has higher query efficiency under reasonable spatial data partition.

【Key words】spatial index; static R-tree; partition; parallel computation

R树索引性能优良，被广泛用于原型研究和商用空间数据库系统，它是目前最流行、最成功的多维索引结构之一[1]。但在海量空间数据的管理过程中，存在R树构建时耗过大、数据更新效率低、全局索引维护困难等问题。因此，本文针对静态R树加载过程良好的可并行性，采用并行计算技术并行化R树的构建过程，以提高索引构建效率。利用大型商用数据库分区技术管理海量空间数据，在逻辑上保证数据的无缝存储，确保查询效率并从物理层次上提高海量空间数据及其索引的可管理性、可用性和性能。

1　分区技术在空间数据库引擎中的应用

1.1　分区技术

数据库提供的分区功能可以提高许多应用程序的可管理性、性能和可用性。在 GIS 领域，将商用关系数据库作为空间数据的容器，采用分区技术提高空间数据访问的性能需要合理确定分区字段。在分区的基础上建立高效、易于管理维护的空间索引是成功应用分区技术的关键。

1.2　海量空间数据的高效存储

1.2.1　分区方案选择

常见的分区方法包括范围分区、Hash 分区、列表分区和组合分区。空间数据的特殊性在于其空间特性，根据 GIS 应用的特点，范围分区和列表分区较适合海量空间数据的大表分区存储。本文采用范围分区方法，通过预先设定图幅范围避免范围分区带来的各分区数据不均匀问题。

1.2.2　数据装载

为保证分区内空间对象地理范围的连续性，先根据数据的空间分布特征划分图幅范围，各图幅应该是整个数据范围的一个划分。如图 1 所示，根据数据的空间分布情况，将数据分为 4 个区，尽量保证各区数据量均衡。存储在数据库中 4 个不同的表空间内，跨图幅的数据单独存放在分区 4 中。

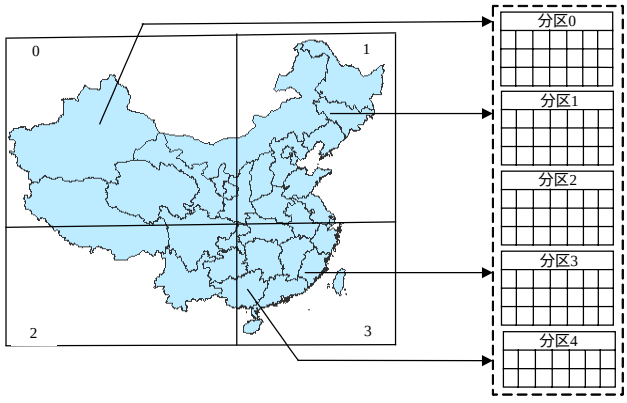


图 1　空间数据分区存储

1.2.3　数据维护

基于分区技术的空间数据存储使空间数据能以分区为单位被维护并管理，如数据更新、索引维护等工作可以在单个分区内进行，而不影响其他分区中的数据，提高了海量数据的可管理性和可维护性。

作者简介：周　芹(1982－)，女，博士研究生，主研方向：GIS 软件技术；钟耳顺，研究员、博士生导师；黄耀欢，博士研究生
收稿日期：2008-07-20　**E-mail：**zhouq.06b@igsnrr.ac.cn

2 静态 R 树的并行构建

2.1 存在的问题

空间数据的R树构建[2]是CPU密集型计算，且涉及大量I/O操作，其时耗很大。已有很多方法可以加速R树的构建，如Packed R树、Hilbert Packed R树、Bercken's Buffer R树、Arge's Buffered R树等静态R树，但针对海量数据的R树构建时间仍然不能满足实际应用的要求。且全局空间索引维护困难，海量空间数据的R树索引会导致树深度增加，查询效率下降。存在小部分数据“脏区”时，必须对全局数据重建空间索引。

针对以上问题，在空间数据分区存储的基础上对分区数据并行计算空间索引，减少索引创建时间。在保证查询效率的同时，提高全局索引的可维护性。

2.2 基于 TGS 算法的 R 树并行构建

2.2.1 TGS 算法

Garcia 等人于 1998 提出自上而下的贪婪分裂(Top-down Greedy-Split, TGS)算法，其特点是自上而下地递归构建 R 树。

在 R 树的递归分裂过程中，通常采用扫描线分割空间中 N 个对象的 MBR 面积作为 TGS 的自定义代价函数选择扫描线，即

$$f(r1, r2)=S_{\mathrm{Area}(r1)}+S_{\mathrm{Area}(r2)}$$

其中，$S_{\mathrm{Area}(ri)}$为被扫描线分割的第i个部分的面积。

每次递归过程中选择分割当前子集中 N 个对象的 MBR 面积最小的扫描线，即

$$S=\min(fi(r1, r2))$$

其中，i=1,2,…,n，n 为扫描线总数；S 为选取的扫描线；$fi(r1, r2)$表示第 i 条扫描线。

2.2.2 R 树的并行构建

如图 2 所示，并行计算R树索引的基本思想如下：采用并行GIS处理常用的典型策略[3]，即DCSO(Decomposition, Conquer, Stitch, Output)策略，本文数据分区存储实现了数据的自然分解，即问题分解；指定处理节点分别处理各分区数据，生成R树的子树；将各子树作为根节点的分支节点插入，完成整体索引创建，得到最终结果。

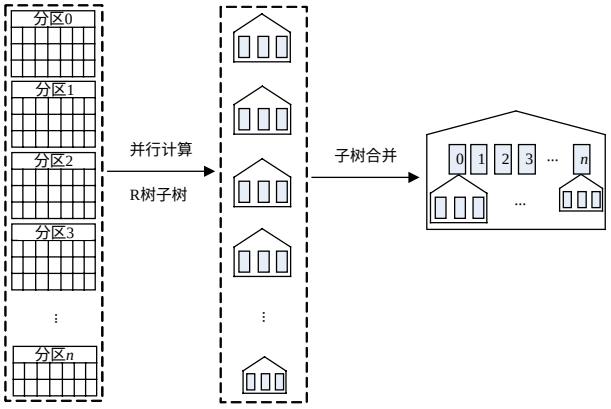


图 2 R 树索引的并行计算

2.3 查询效率分析

R树的查询效率与节点的总面积和边长相关，在其他变量一定的情况下，总面积和边长越小，查询代价(磁盘I/O)越小[4]。在基于分区的R树索引的并行计算中，人为地在 R 树顶端对数据按分区范围生成n个分支节点，以提高数据的空间聚集度。在保证数据量合理分区的情况下，R树节点将更趋近正方形，可以提高R树的整体质量和查询效率。

3 实验结果

3.1 并行计算平台的选择

用集群实现并行计算具有很高的灵活性。本文构建一个小型集群系统实现海量数据空间静态 R 树索引的并行计算。

3.2 实验环境

4 个计算节点硬件配置如下：Intel(R) Pentium(R) 4 CPU 2.60 GHz，操作系统为 Microsoft Windows 2000 Professional。一个数据服务器配置如下：Intel(R) Xeon(TM) CPU 2.40 GHz，操作系统为 Microsoft Windows 2003 Enterprise Server Edition, (05.02.3790.00)，数据库为 Oracle 10g。

单机串行计算实验结果如表 1 所示。

表 1 单机串行计算实验结果

计算节点	对象数	计算时间/s
节点 1	1 504 110	1 233.5

实验数据为全国 1: 250 000 等高线数据，共 1 504 110 条记录。数据表分 5 个区存储，编号为 0~4，其中，4 区存储跨区对象。集群计算实验结果见表 2，其中，总时间包括数据传输、计算、数据存储的时间。

表 2 集群计算实验结果

分区号	对象数	计算节点编号	计算时间/s
0	286 819	0	90.5
1	419 166	1	120.3
2	327 038	2	108.6
3	469 538	3	150.2
4	1 549	0	1.2
子树合并	-	0	0.1
总计	-	-	196.5

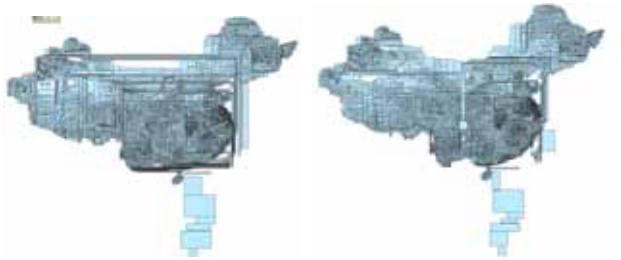
3.3 实验结果及分析

3.3.1 索引构建时间对比

采用 TGS 算法进行单机串行计算和集群计算的时间如下：4 个节点进行并行计算获得的加速比为 6.28。基于 TGS 算法的静态 R 树构建的计算量与数据量密切相关，对分区数据计算局部空间索引减少了计算量，因此，可以获得超线性加速比。

3.3.2 查询效率对比

采用上述算法构建的 R 树与串行构建的 R 树结构不同，根据 2.3 小节的论述，比较 2 种方法所建立索引的叶子节点分布，如图 3 所示。串行计算的索引包在数据密度较小的区域容易出现“长条”形状，通过分区方法能避免跨分区的“长条”出现，获得更好的查询效率，且 R 树结构将有所改善。



(a)串行计算结果　　(b)并行计算结果

图 3 R 树叶子节点对比

在实际操作中对查询效率进行测试，由于点查询可以看作是域查询的特殊情况，因此本文采用域查询方式进行实验。

分别取图幅范围的 1%, 2%, 3%, 4%, 5%作为查询窗口进行 200 次随机域查询，统计磁盘 I/O 数和查询时间，结果见图 4、图 5。

(下转第 73 页)