doi:10.19509/j.cnki.dzkq.2019.0527
朱静,刘振华,乔栋.基于HBase的海量地理空间数据的空间索引模型构建与优化[J].地质科技情报,2019,38(5):253-260.

基于HBase的海量地理空间数据的空间索引模型构建与优化

朱 静[a,b],刘振华[a,b],乔 栋[a,b]

(中国地质大学(武汉) a.计算机学院; b.智能地学信息处理湖北省重点实验室,武汉 430074)

摘 要:传统关系型数据库在海量地理空间数据的存储与管理上面临着高并发访问规模限制、数据库扩展能力不足等困难。非关系数据库如HBase等以其强大的扩展能力与计算能力为该问题提供了新的思路与方法。空间索引模型和分布式存储模式设计是影响基于非关系数据库的海量地理空间数据的存储与查询效率的关键因素。对当前主要基于HBase的索引模型和空间数据存储设计进行了研究,设计了基于行政区划编码与矢量要素编码结合的RowKey(行键),使空间数据在HBase存储中得到很好的聚类效果,并针对要素重叠与边界划分等问题提出了一种基于四叉树-R树的改进的空间索引模型。该模型基于四叉树结构将空间数据划分为多个子网格,为每一个子网格构建R树索引,利用Hilbert(希尔伯特)曲线对子网格进行编码,并设计了基于MapReduce的并行化索引构建算法和相应的空间查询算法。经实验测试,该存储设计和空间索引模型具有较好的查询效率。

关键词:海量地理空间数据;分布式存储; RowKey;四叉树—R树空间索引模型;空间查询
中图分类号:TB115 **文献标志码**:A **文章编号**:1000-7849(2019)05-0253-08

Construction and Optimization of Spatial Index Model for Massive Geospatial Data Based on HBase

Zhu Jing[a,b], Liu Zhenhua[a,b], Qiao Dong[a,b]

(a.Faculty of Computer; b.Hubei Key Laboratory of Intelligent Geo-Information Processing, China University of Geosciences(Wuhan), Wuhan 430074, China)

Abstract: The traditional relational database is faced with difficulties in the storage and management of massive geospatial data, such as the limitation of high concurrent access scale and the lack of database expansion ability. Non-relational databases such as HBase provides new ideas and methods for this problem. Distributed storage mode and spatial index model are key factors which affect the distributed storage and query efficiency of massive geo-spatial data based on non-relational database. This paper studies the storage mode of HBase and the current spatial index model based on non-relational database, and designs a kind of RowKey based on the combination of administrative division coding and vector element coding which leads to a good clustering effect for the geo-spatial data in HBase storage. The paper improves a spatial index model based on quadtree-R tree through proposing to solve the problems of feature overlap and boundary division. A parallel index construction algorithm based on MapReduce is designed in this paper to further

收稿日期:2019-06-05 编辑:杨 勇
基金项目:智能地学信息处理湖北省重点实验室开放基金项目"基于HBase的空间大数据分布式存储与查询优化"(KLIGIP-2017B11)
作者简介:朱 静(1974—),女,副教授,主要从事地学大数据处理、机器学习、网络技术应用等教学与科研工作。E-mail:zhujing@cug.edu.cn
通信作者:刘振华(1995—),男,现正攻读计算机技术专业硕士学位,主要从事地学大数据研究工作。E-mail:416251221@qq.com

improve efficiency of algorithm. In the model, the index space is divided into multiple subspace based on the quadtree structure, and the subspace is encoded by Hilbert curve. The experiments have verified the model has higher time efficiency and accuracy than the existing models in spatial data storage and query.
Key words: massive geospatial data; distributed storage; RowKey; Quadtree-R tree spatial index model; spatial query

据麦肯锡研究所报告,2016年全球地理空间数据总量已经超越了6 000 PB,且每年仍以PB级别的速度在增加[1]。针对海量的地理空间数据进行高效的存储与管理已成为当前地理信息业界及学界关注的重要问题。

Hadoop是当前主流的大数据存储系统框架,提供了分布式文件系统HDFS(hadoop distributed file system)、分布式数据库HBase和分布式计算框架MapReduce。HBase能够提供高效的随机访问,多格式的数据存储,以及高并发的数据读写[2]。但HBase不支持空间数据存储与空间查询,需要通过构建并实现空间索引模型,设计查询算法,将其扩展应用至地理空间数据的处理[3]。

空间索引的目的是提高空间数据的查询效率[4]。目前关系型数据库的空间索引主要有空间填充曲线、空间网格、基于树的空间索引3类。基于树的空间索引主要有四叉树、K-D树、R树以及R树的变种R^+树等[5]。这些方法在进行海量地理空间数据的存储和管理时,存在数据存储组织困难、难以满足实时查询需求等诸多问题[6]。针对这些问题,分布式数据库HBase和并行计算框架MapReduce提供了可行的解决方案,将海量的地理空间数据存储到NoSQL(非关系数据库)中,利用NoSQL的分布式特性,提供高并发的访问[7]。

基于HBase的海量地理空间数据的高效分布式存储和管理,需要设计适当的存储模式和构建高效的空间索引模型。其中空间数据划分方法是构建空间索引模型的关键问题。目前空间数据划分方法相关研究中,由于空间数据的分布特征造成每个磁盘间的数据存储容量不均衡而产生的数据倾斜现象一直没有得到很好的解决。

笔者拟对当前主要基于NoSQL的海量地理空间数据的分布式存储模式和空间索引模型进行研究,设计基于行政区划编码与矢量要素编码结合的RowKey(行键),使得地理空间数据在HBase存储中得到很好的聚类效果。针对要素重叠与边界划分等问题提出一种基于四叉树—R树的改进空间索引模型,并设计基于MapReduce的并行化索引构建算法和相应的空间查询算法以进一步提高索引构建效率。

1 相关研究

1.1 分布式存储模式设计

目前国内外较多采用的方法是将HBase数据表中需要索引的数据直接以RowKey作为索引项的Value来实现[8]。Xiong等[9]针对MySQL在处理大规模地理空间数据上的缺点,提出了HBaseSpatial模型,通过实验证明了其在处理地理大数据上优于MySQL。

范建永等[10]设计了矢量数据的分布式存储模式,使用矢量要素的唯一ID作为行键。其缺点是在处理多版本数据以及海量数据时,无法保证ID是唯一的,且聚类效果差。陈俊欣[11]设计了一种空间数据复合键拼接加入分隔符的行键存储模式,并取消了过滤列族的设计。Aji等[12]提出了基于MapReduce的高性能Hadoop数据仓库系统,利用Hadoop分布式特点来实现对地理空间数据的高并发访问,并提供了基于多字段与位置的查询接口。

1.2 空间数据划分方法

传统空间数据划分方法主要有轮转法、散列划分、范围划分、混合划分等,这些方法只能在选定关系的一个属性上划分,不支持非划分属性上的查询[13]。赵春宇等[14]基于Hilbert曲线提出了一种提高各存储节点间存储均衡的划分方法。陆锋等[15]提出了一种Hilbert快速编码方法,通过空间层次分解提高了Hilbert曲线的编码速度。周艳等[16]针对现有方法不考虑相邻对象空间关系对数据划分的影响等问题,提出了一种基于Hilbert空间填充曲线层次分单元的空间划分方法。Cary等[17-18]提出将空间数据划分算法与并行计算技术相结合,基于MapReduce对空间数据划分策略算法进行了并行化处理。

这些研究面对TB级、PB级及以上的海量数据,依然没有很好解决空间数据划分效率低的问题。

1.3 R树索引模型

R树是树形结构的空间查询索引,衍生出了诸多变种如R^+树,R^*树等。R树是一个平衡的多路查找树,非叶子节点不存放数据,所有数据均存储在叶子节点一层,且所有叶子节点均在同一层[19]。R树空间允许相交,查询总是从根节点开始,最坏情况下需查找整棵树,最好情况是查找单个分支,因此当索引的数据量变得很大时,R树的深度与索引空间的重叠都会增大,查找性能急剧下降[20]。

1.4 四叉树索引模型

四叉树每个非叶子节点都有 i 个子节点($1 \leqslant i \leqslant 4$)。四叉树常用于二维空间数据的分类与索引。Lee 等[21]提出了一种基于 KD$^+$-Tree 的轻量级空间索引模型,在针对点类型数据的查询(K 近邻查询)上,得到了更高效的查询效率。但由于地理空间数据的区域分布总是不均匀的,因此四叉树索引在空间数据的存储上存在以下几个缺点[22]:①所有的空间数据都只在叶子节点中存储。存储海量的空间数据时,四叉树深度会很大,查询效率低下。②四叉树的划分是简单的四等分,对于跨索引空间边界的空间对象,会存在一个空间对象存储在多个叶子节点的要素重叠情况。③空间数据的分布不均衡会导致四叉树深度参差不齐,特别对于满四叉树而言,会极大地消耗存储空间。

2　四叉树-R 树空间索引模型

针对上述 R 树与四叉树索引模型的缺点,本次研究提出一种基于四叉树划分索引空间的方法,并设计提出了一个四叉树－R 树空间索引模型。

2.1 存储设计

2.1.1　地理空间数据

本次研究的存储模式设计主要针对矢量数据。矢量地理空间对象一般包括二类数据,一是几何数据,描述矢量要素的位置、形状、大小及其分布特征;二是属性数据,描述矢量对象具有的属性信息,有时间、面积、长度等其他描述类属性信息。以中国国家发布县级及乡镇行政区域数据(Shp 文件)为例,其逻辑模型如表 1 所示。

表 1　地理空间数据模型

Table 1　Geo-spatial data model

属性项	数据
GEOMETRY	几何数据集合
feature_ID	矢量要素码
TimeStamp	时间
province_c	省级区域代码
city_c	地级区域代码
district_c	县级区域代码
town_c	乡镇级区域代码
province	省级名称
city	地级市名称
district	县级名称
town	乡镇(街道)名称
poptotal	人口总数

2.1.2　RowKey 设计

在 HBase 表中,一个矢量对象存储一行,作为一个完整的记录。每一行需要有一个唯一的标识符 RowKey,所有的行按照 RowKey 的字典序排列。HBase 为 RowKey 建立了高效的 B$^+$ 树索引,对单个矢量要素,RowKey 的设计是影响 HBase 检索效率的关键因素。

HBase 并不提供类似关系型数据库中的多属性查询。一般做法是合理设计 RowKey,通过对 RowKey 的解析实现多属性查询。以此为基础,笔者设计了行政区划结合矢量要素名称 RowKey。为保证属于相同行政区的矢量要素在 HBase 中实际的存储位置聚集,以 region_ID＋feature_ID 的编码方式作为矢量要素在 HBase 中存储时的唯一 RowKey。其中 region_ID 为 12 位的行政区划编码,省、地、县、镇 4 级行政区编码,编码规则采用中国民政局发布的中国县级以及县级以下行政区划分编码。例如,甘肃省酒泉市敦煌县月牙泉乡的 region_ID 为 620982106000。其中 4 级行政区编码为 9 位,后三位补 0,留作备用。feature_ID 是 8 位的矢量要素编码。

每个矢量要素对应的 RowKey 的组成结构如表 2 所示。HBase 存储时每一条记录对应一个 TimeStamp,从而可以实现维护多版本的数据管理。

表 2　RowKey 结构组成

Table 2　Structure composition of RowKey

RowKey	region_ID＋feature_ID

2.1.3　列族

在 Hbase 中,同一个列族下面的所有列存储在同一个文件中,因此在相同列族中应存储可能被一起访问的数据,以减少磁盘 I/O,提高数据访问效率。因此,本次研究在数据表中设计了 2 个列族:COLUMNFAMILY_GEOMETRY(其中包含列 Geometry)和 COLUMNFAMILY_PROPERTY,存放矢量要素的属性信息。表 3 为 HBase 地理空间数据表的逻辑视图结构。

表 3　HBase 地理空间数据表的逻辑视图结构

Table 3　Logical table view of HBase geospatial data table

RowKey	TimeStamp	COLUMNFAMILY: GEOMETRY	COLUMNFAMILY: PROPERTY		
		Geometry	Fea_ID	Area	…
RowKey	T3	Pologyn	xxx	xxx	
	T2	Pologyn	xxx	xxx	
	T1	Pologyn	xxx	xxx	

2.2 空间数据划分方法

笔者提出的索引模型将整个索引空间分解为多个小的子索引空间,即用若干简单 R 树代替一棵复杂

R树，由此可将空间数据查询定位到小的子索引空间，大大减少空间要素重叠。划分后的索引空间呈金字塔状，每个节点代表一个索引空间。为每个索引空间构建一个R树，将需查询的矢量要素所在索引空间缩小。对划分后的子索引空间进行Hilbert曲线编码，使得物理上相邻的索引空间在实际存储中也在相邻的位置。整个索引空间结构如图1所示。

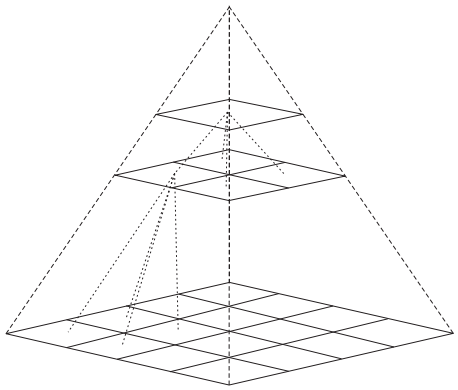

图1 四叉树索引空间划分

Fig.1 Quadtree index space partition

为保留空间数据邻近性特征，避免要素重叠，设计的空间对象放置策略应遵循以下规则：①索引空间能完全覆盖空间对象的最小外接矩形。②放置空间对象的索引空间需保证空间对象最小外接矩形与该索引空间的增量应是所有空间对象最小外接矩形与索引空间增量中最小的。

如图2所示，由规则①可知R1应放置到第一层级的子索引空间，由规则②可知，R2应放置到第二层级的左上索引空间；同理可推出R3、R4应放置到第三层级的不同子索引空间。

Hilbert曲线可以对多维空间进行离散值近似表示，将其线性化为一维，但保留了空间邻近性。因此将二维空间划分后，用Hilbert曲线进行连接，使得相邻区域的物理存储空间也连续，如图3所示。

2.3 索引表模型

为保证空间对象与空间索引映射关系的一一对应，本次研究在HBase中建立了索引表，经四叉树划分之后的每个索引空间都对应索引表的一行，其中，RowKey字段是每个索引空间的唯一编码标识。

经四叉树划分后的金字塔型索引空间中，每个索引空间都由层级(Level)、行(Row)、列(Column)三元组进行唯一标识。每一层级用Hilbert曲线进行统一编码得到M，使得同一层级物理位置相邻的空间对象的物理存储空间位置也相邻。Level+M即为唯一的RowKey，并命名其RowKey为QTID。RowKey逻辑结构如表4所示。

表4 经四叉树划分金字塔索引空间的RowKey

Table 4 RowKey of pyramid index space divided by quadtree

RowKey(QTID)	
Level	M

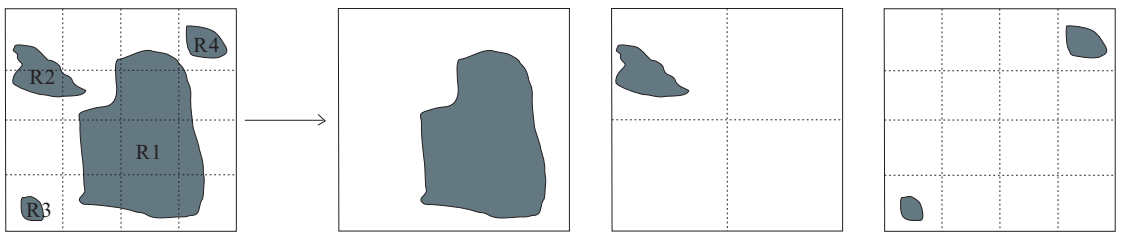

图2 空间对象放置策略

Fig.2 Space object placement policy

表5 四叉树索引空间的行结构

Table 5 Row structure of quadtree index space

RowKey (QTID)	Family:RTree		Family:ChildNodes		Family:Parent	
	mbr	R-RowKey	childMbrs	childRowKeys	Pmbr	PRowKey
Level+M	rectangle	R_RowKey	retangles	k1 $ k2 $ k3 $ k4	rectangle	P_RowKey

行结构如表5所示，其中，每一行代表一个索引空间，并且指向一棵R树索引。

本索引模型中为每个索引空间建立R树索引。其中R树的索引表中，分为叶子节点与非叶子节点。非叶子节点结构中，使用R树所属索引空间的QTID、当前层级level与MBR三元一体来唯一标

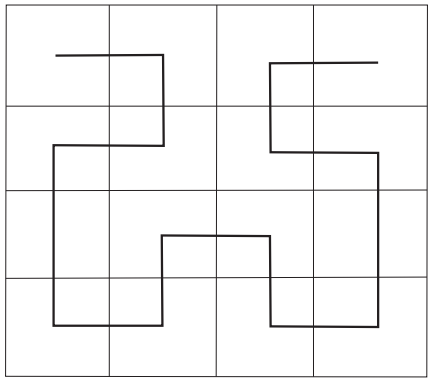

图 3　Hilbert 曲线空间划分

Fig. 3　Schematic diagram of space division by Hilbert curve

识每一个节点,使用当前索引空间的 QTID 作为行键的开头,这样可使同一索引空间下所有 R 树空间索引聚集在一起。R 树 RowKey 结构如表 6 所示。R 树非叶子节点属性字段包含所有子节点与指向子节点的指针、父节点与指向父节点的指针,其结构如表 7 所示。叶子节点的属性段应包括所有空间对象的物理存储空间 RowKey 与父节点及其指向父节点的指针。其结构如表 8 所示。

2.4 基于 MapReduce 的四叉树—R 树索引生成算法

在划分好的索引空间上,需要为每个索引空间建立 R 树索引,为提高构建索引效率,设计了基于 MapReduce 的并行化构建索引的算法。

表 6　R 树索引节点的 RowKey

Table 6　RowKey of R-tree index node

RowKey		
QTID	level	MBR

MapReduce 计算框架只需要通过 map 和 reduce 接口定义好自己的处理函数,然后执行基于 MapReduce 映射规则的并行计算,即可生成需要的 key/value(键值对)。

MapReduce 框架集群中的节点由一个 JobTracker 和若干个 TaskTracker 组成。JobTracker 负责任务调度,TaskTracker 负责并行执行任务。任务的执行主要分为 Input,Map,Reduce,Output 4 个阶段。首先将输入数据文件进行划分,文件划分大小可通过函数参数进行配置,一般为 16 MB 到 64 MB 大小,然后通过 JobTracker 为 M 个 Map 和 R 个 Reduce 分配任务。Map 阶段从 HDFS 读取对应的输入数据块,通

表 7　R 树索引非叶子节点的行结构

Table 7　Row structure of non-leaf nodes indexed by R-tree

RowKey	Family:Parent		Family:ChildNodes	
	mbr	parentRowKey	childMbrs	childRowKeys
QT+Level+MBR	[a,b,c,d]	RowKey	[]/[]/[]/[]	k1 $ k2 $ k3 $ k4

表 8　R 树索引叶子节点的行结构

Table 8　Row structure of Leaf Node in R-Tree Index

RowKey	Family:Parent		Family:Childs
	mbr	parentRowKeys	childRowKeys
QT+Level+MBR	[a,b,c,d]	RowKey	k1 $ k2 $ k3 $ k4

计算输出中间结果;Reduce 阶段将 Map 阶段的输出结果按 Key 值分为 R 份,每个划分对应一个 Reduce,并按 Key 值做 merge 处理,即将具有相同 Key 值的 key/value 合并,形成新的 key/value 数据集并传递给 Reduce 函数,Reduce 函数对传入的每个 key/value 进行分析计算,输出最终结果。

构建索引过程如下。

Input 阶段:从 HBase 中读取出存储的空间数据,获取矢量要素 Geo 及其属性信息。

Map 阶段:并行处理数据,根据矢量要素对象的 geometry 计算出空间对象的 MBR,并根据 MBR 将其映射到对应的四叉树索引空间中,并保存 MBR 结果。

Reduce 阶段:将映射到同一个索引空间数据整合,开始插入 R 树操作。

Output 阶段:将所有结果导入到索引表中。

算法伪代码描述如下 Algorithm 1 所示。

```
Algorithm1:GetIndexSpace(geometry ,root,level)
Input:矢量对象的几何要素 geometry、四叉树索引空间的入口
      root 与层级 level
Output:矢量对象所放置的索引空间的根节点
1:R ← Φ
2:L ← 0
3:MBR←getMBR(geometry)
4:While root.MBR←root.MBR∪MBR || L<root.level
5:do root←root.child
6:If root.level=level && root.MBR<root.MBR∪MBR
```

```
Algorithm1:GetIndexSpace(geometry ,root,level)
7:      return root
8:else
9:return root.parent
```

3 实验测试与分析

3.1 实验环境

本实验的运行环境是由4台虚拟机节点组成小集群,由1个Master节点与3个Slave节点组成,部署了Hadoop2.7.4与HBase1.2.6。每台机器配置4G内存、200G硬盘以及CentOS 7操作系统。

为了便于测试本次研究的存储设计与索引模型,实验数据集选取了中国行政区划矢量数据、乡镇分布矢量数据以及矢量数据,其中要素总量近五百万。

3.2 聚类测试

本实验通过HBase提供的基于RowKey的直接查找Get和范围查找Scan在不同数据量上的查找效率的对比测试RowKey的聚类效果。经过实验测试,比较在查询连续范围不同数据量的记录时,Get查找与Scan查找所用的时间对比如图4所示。

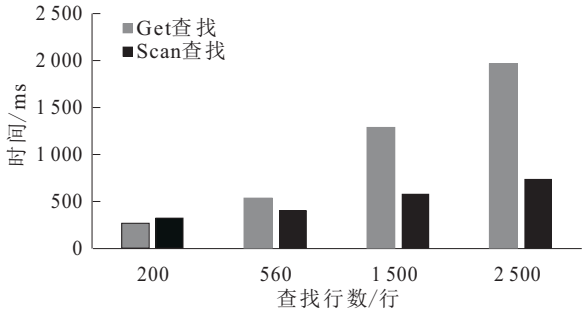

图4 不同数据量应用Get查找与Scan查找时间对比

Fig.4 Comparison of Get lookup and Scan lookup time with different amount of data

实验结果显示,当查询记录数目达到一定程度后(大于600行),Scan范围查找相比Get直接查找所用的时间更短,查找1 500,2 500行数据的效率分别高出124.8%和199.9%。可以看出本次研究设计的RowKey具有优秀的聚类效果,且查询数据量越大,优势越明显。

3.3 索引模型构建效率测试与分析

本次实验在10万、20万、30万3种矢量对象数量的数据集上,分别基于R树、四叉树以及四叉树-R树索引模型进行空间索引构建,分析对比其索引构建效率。本次实验在单线程环境下进行,四叉树采用动态构建方式、R树采用动态插入方式,四叉树-R树采用静态构建、四叉树采用动态构建R树的方式。构建完成的索引存放在HBase的索引表中。测试结果如图5和表9所示。

从表9可以看出,矢量对象数量为10万时,四叉树-R树索引模型构建时间效率比R树和四叉树分别提高了55.5%,40.7%,但三者时间效率绝对值相差并不大。矢量对象数量为20万时,四叉树-R树模型时间效率分别比两者提高了76.9%和52%。矢量要素数量增长到30万时,四叉树-R树模型时间效率分别比两者提高了81.7%和73%。数据量越大,优势越明显。

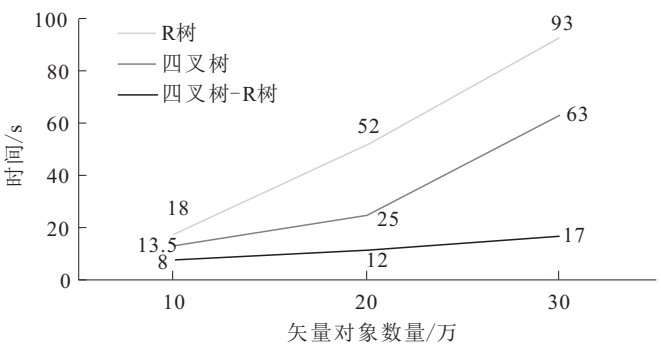

图5 基于3种空间索引模型的索引构建时间效率对比

Fig.5 Comparison of time efficiency of index construction based on three spatial index models

表9 基于3种空间索引模型的索引构建时间效率对比

Table 9 Comparison of time efficiency of index construction based on three spatial index models

矢量对象数量/万	R树	四叉树	四叉树-R树
	查找时间/s		
10	18	13.5	8
20	52	25	12
30	93	63	17

为验证本次研究设计的基于MapReduce的并行化索引构建算法效率,本次研究设计并实现了单线程构建四叉树—R树索引与MapReduce并行化构建索引的对比实验。实验结果如图6所示。可以看出在矢量要素分别为20万、80万、140万和200万个时,基于MapReduce的并行化索引构建时间效率分别高于单线程构建四叉树-R树索引7.7%,56%,71.2%,70.6%,并行化索引构建算法时间效率大大优于单线程索引构建算法。

3.4 空间查询测试与分析

3.4.1 范围查询

本实验中首先测试了四叉树-R树不同的索引空间层级对查询性能的影响。实验测试结果如图

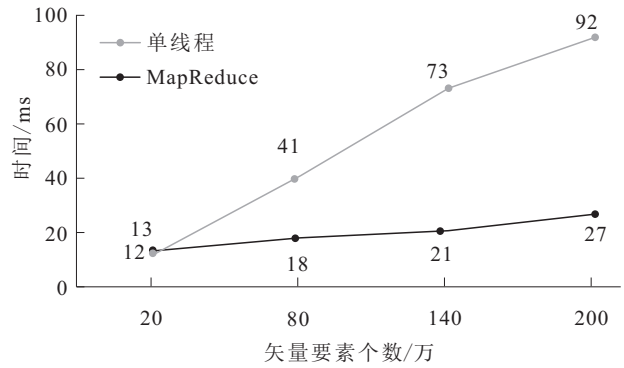

图 6　MapReduce 并行化算法与单线程构建索引时间效率对比

Fig.6　Comparison of time efficiency between MapReduce parallelization algorithm and single thread index construction

7,从用时看索引空间的最大层级设定 10 相对比较合理,设置较大或较小,时间效率都会降低。

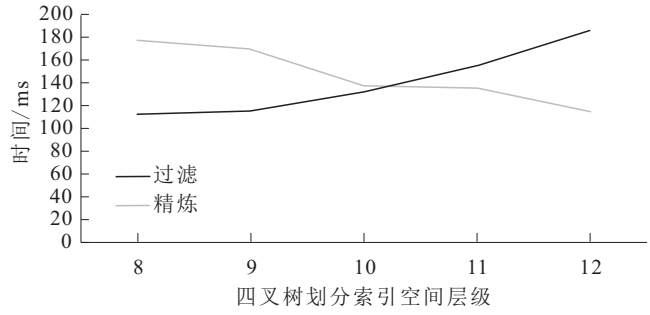

图 7　四叉树划分索引空间层级对查询效率的影响

Fig.7　The influence of Quad Tree partition index space level on query efficiency

为对比范围查询效率在 3 种索引模型上的表现,在本次实验中,定义了 3 个不同的多边形,各自包含不同数量的矢量要素,结果如图 8 所示。

实验结果表明,查询范围内矢量要素为 2 248,4 783,11 045 个时,四叉树-R 树索引模型时间效率

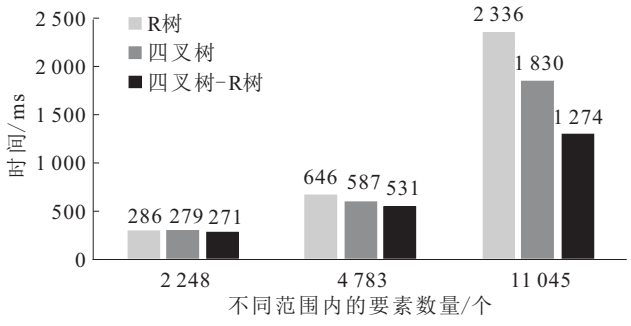

图 8　基于 3 种空间索引模型的范围查询效率对比

Fig.8　Comparison of range query efficiency based on three spatial index models

优于 R 树0.52%、17.8%、45.5%优于四叉树0.03%、0.95%、30.3%,充分说明了在大数据情况下四叉树-R 树索引模型在范围查询上的优越性能。

3.4.2　*K* 近邻查询

本实验首先测试了 *K* 近邻查询中半径 *r* 对于实验结果的影响。选取坐标点 *P*(105.389 723 46,29.923 872 16)作为查询中心点,设置 *K* 值为 5,10,15,20 四个查询值,作为 *K* 近邻查询输入,得到实验结果如表 10 所示。

表 10　圆形区域半径 *r* 对查询效率的影响

Table 10　Influence of circular area radius *r* on query efficiency

查询半径 *r*	是否满足查询	对象个数	10 个点/ms
0.05	否	5	2 334
0.07	是	16	538
0.1	是	51	1 184
0.2	是	339	2 983
0.4	是	1 227	8 216

由表 10 可以看出,半径 *r* 设置在 0.07～0.1 范围内查询效果较好。以此半径范围为基础,本次研究设计了实验进行 *K* 近邻查询的效率测试。在相同环境下,除去代码影响,对比了不同 *K* 值 5,10,15,20,来测试 *K* 近邻查询的效率。测试结果如图 9 所示。实验测试结果表明,四叉树-R 树空间索引模型在 *K* 近邻查询点数据时,耗时远低于四叉树与 R 树模型,效率最高。

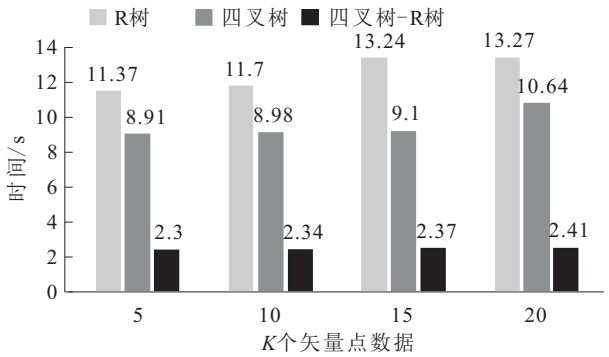

图 9　基于 3 种空间索引模型的 *K* 近邻查询时间效率对比

Fig.9　Comparison of time efficiency of *K*-nearest neighbor query based on three Spatial Index models

4　结论与展望

(1)研究地理空间数据在基于 HBase 的分布式存储模式,设计了基于行政区划编码与矢量要素编码结合的行键,实验证明得到了很好的聚类效果。

(2)针对海量地理空间数据分布式存储中的要素重叠与边界问题,提出了一种四叉树-R 树空间索

引模型，大大降低了R树索引的复杂度，提高了空间查询效率。

(3)基于MapReduce计算框架，设计了一种并行化构建空间索引的算法，进一步优化提高了索引构建的时间效率。

(4)基于四叉树-R树索引空间结构模型，结合地理空间数据的查询需求，设计了范围查询与K近邻查询的查询算法。

本次研究与实验为海量地理空间数据的存储与管理提供了新的方法与思路，但是仍然存在不足。从一般数据的索引结构而言，通常格网类型与树形结构结合能得到较好的索引性能，但是空间数据存在特有的分布均衡性问题，如何平衡格网层级、树形索引与空间数据的分布均衡性之间的关系，是下一步值得研究的方向。

在地理空间数据查询方面，本次研究只涉及2种基本的查询方式：范围查询与K近邻查询(点查询)。针对地理空间数据的多种拓扑关系，如何设计不同的查询算法来满足多样化的查询需求也是未来值得研究的一个方向。

参考文献：

[1] Zhong Y, Han J, Zhang T, et al. Towards parallel spatial query processing for big spatial data[C]// 2012 IEEE 26th international parallel and distributed processing symposium workshops & PhD Forum (IPDPSW). Shanghai: IEEE, 2012: 2085-2094.

[2] Hadoop A. Hadoop[EB/OL]. (2009-03-06)[2019-5-18]http://hadoop.apache.org.

[3] Dimiduk N, Khurana A, Ryan M H. HBase in action[M]. Shelter Island: Manning, 2013.

[4] 熊才权，马乐乐，孙贤斌.空间索引技术研究[J].计算机技术与发展，2010，20(10)：219-223.

[5] 刘刚.空间索引技术研究[J].科技经济导刊，2016(33)：40.

[6] Gao J, Xing H, Tian Z, et al. Lattice Boltzmann modeling and evaluation of fluid flow in heterogeneous porous media involving multiple matrix constituents[J]. Computers & Geosciences, 2014, 62(2): 198-207.

[7] Van der Veen J S, Van Der Waaij B, Meijer R J. Sensor data storage performance: SQL or NoSQL, physical or virtual[C]// 2012 IEEE fifth international conference on cloud computing. Honolulu: IEEE, 2012: 431-438.

[8] Zhang N, Zheng G, Chen H, et al. Hbasespatial: A scalable spatial data storage based on hbase[C]// 2014 IEEE 13th international conference on trust, security and privacy in computing and communications. Beijing: IEEE, 2014: 644-651.

[9] Xiong Y, Zuo R. Recognition of geochemical anomalies using a deep autoencoder network[J]. Computers & Geosciences, 2016, 86: 75-82.

[10] 范建永，龙明，熊伟.基于HBase的矢量空间数据分布式存储研究[J].地理与地理信息科学，2012，28(5)：39-42.

[11] 陈俊欣.基于Hadoop的空间矢量数据的分布式存储与查询研究[D].成都：电子科技大学，2016.

[12] Aji A, Wang F, Vo H, et al. Hadoop gis: A high performance spatial data warehousing system over mapreduce[J]. Proceedings of the VLDB Endowment, 2013, 6(11): 1009-1020.

[13] Shekhar S, Chawla S. 空间数据库[M].谢昆青，马修军，杨冬青，等译.北京：机械工业出版社，2004：10-15.

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

[15] 陆锋，周成虎.一种基于空间层次分解的Hilbert码生成算法[J].中国图象图形学报：A辑，2001，6(5)：465-469.

[16] 周艳，朱庆，张叶廷.基于Hilbert曲线层次分解的空间数据划分方法[J].地理与地理信息科学，2007，23(4)：13-17.

[17] Cary A, Sun Zhengguo, Hristidis V, et al. Experiences on processing spatial data with MapReduce[C]// Anon. International conference on scientfic and statistical database management. Berlin: Springer, 2009: 302-319.

[18] Cary A, Yesha Y, Adjouadi M, et al. Leveraging cloud computing in geodatabase management[C]// Anon. 2010 IEEE international conference on granular computing. San Jose: IEEE, 2010: 73-78.

[19] Li W, Schmitt D R, Zou C, et al. A program to calculate pulse transmission responses through transversely isotropic media[J]. Computers & Geosciences, 2018, 114: 59-72.

[20] Kothuri R K V, Ravada S, Abugov D. Quadtree and R—tree indexes in oracle spatial: A comparison using GIS data[C]// Anon. Proceedings of the 2002 ACM SIGMOD international conference on management of data. Madison, Wisconsin: ACM, 2002: 546-557.

[21] Lee K, Ganti R K, Srivatsa M, et al. Efficient spatial query processing for big data[C]// Anon. Proceedings of the 22nd ACM SIGSPATIAL international conference on advances in geographic information systems. Fort Worth: ACM, 2014: 469-472.

[22] 赵波，边馥苓.面向移动GIS的动态四叉树空间索引算法[J].计算机工程，2007，33(15)：86-87.