doi:10.3772/j.issn.1002-0470.2018.11－12.002

一种基于预存交点的矢量空间叠加分析算法①

肖茁建②*** 邱 强*** 姚 晓*** 方金云*

(*中国科学院计算技术研究所 北京 100190)

(**中国科学院大学 北京 100190)

摘 要 针对矢量空间叠加分析服务实时性的需求,提出了一种基于预存交点信息的矢量空间叠加分析算法。在叠加分析算法中实现了存储计算一体化的交点数据结构,能够满足交点额外空间占用率小的存储需求和快速获取交点信息的计算需求。在叠加分析时,将图层之间的交点离线计算并存储,以查询交点的方式代替传统算法中计算交点的方式,用少量的空间代价避免了计算交点的时间开销。在保证叠加分析算法结果正确性的同时,极大提高了叠加分析算法的效率。实验结果表明,与传统计算模式相比,本文方法在低于10%的额外空间占用率的代价下,使得算法计算的时间减少92.4%,并且并行算法能够取得较为理想的并行加速比。

关键词 叠加分析,空间数据管理,预存交点,存储计算一体化

0 引 言

矢量空间叠加分析是一种重要的空间分析方法,常用于提取和分析多个图层中同一区域的空间数据和属性信息,是地理信息系统的核心操作之一。同时,矢量空间叠加分析也是一个计算密集型的分析操作。随着互联网技术特别是移动互联网的发展,一方面,网络地图服务中对叠加分析等空间分析服务的实时性要求越来越高,而另一方面,地理数据无论是在数据规模还是数据多样性上也在快速地增长,对叠加分析算法的性能提出了更高的要求。

以面面叠加分析为例,首先需要对所有相交面要素中的线段求交点,再通过这些交点构造出结果要素集合。文献[1]对叠加分析中各个步骤做了详细的时间复杂度分析,同时通过实验统计发现线段求交点的时间占整个算法时间的80%～90%。线段求交点过程是空间叠加分析算法主要的性能瓶颈,线段的交点作为算法过程中关键的中间计算结果,具有以下特点:相同图层多次叠加分析时,交点数据可以复用;交点数据结构简单,数据量小。

针对线段交点的特点,本文将线段求交点的过程从叠加分析中分离出来,设计了线段交点的数据结构,满足线段交点额外空间占用率小的存储需求和快速获取交点信息的计算需求,提出了存储计算一体化[2]的矢量叠加分析模式:将传统叠加分析算法中关键的中间计算结果(线段交点)离线计算并存储,在叠加分析时,以查询交点的方式代替传统算法中计算交点的方式,以少量的空间代价降低叠加分析的时间成本。

1 研究现状

1.1 矢量空间分析

近年来随着并行计算、分布式计算的发展,许多矢量空间分析中引入了相关技术来解决空间分析数

① 国家重点研发计划(2016YFB0502300,2016YFB0502302)资助项目。

② 男,1991年生,博士生;研究方向:并行空间分析,空间数据存储与管理,分布式计算;E-mail:xiaozhuojian@ict.ac.cn

(收稿日期:2018-04-09)

据量大、计算密集等问题。文献[3]总结了近几年并行矢量空间分析的相关研究。文献[4]设计了基于空间聚类的矢量空间数据并行计算划分方法,在保持空间算法邻近性的基础上提高了并行空间分析算法的负载均衡度。文献[5]基于 GPU 实现了一种多边形裁剪算法,利用 GPU 解决叠加分析中计算密集的问题。文献[6]在单机多核环境下基于 OpenMP 实现并优化了大量线段求交算法,提出了利用动态内存分配技术来保证并行计算的效率。文献[7]提出的并行空间分析算法,利用二级均匀网格索引提高并行计算的负载均衡度,并且提出了一种没有舍入误差的矢量分析算法。文献[8,9]基于 MapReduce 框架实现了矢量数据分析算法,算法并行效率进一步提升。文献[10,11]针对并行空间分析中的 IO 瓶颈,提出了一种全内存的矢量空间数据存取技术,利用内存计算和内存数据库极大地提高了并行矢量空间分析的 IO 效率。文献[12]提出了一种估计空间计算强度的方法,在并行计算任务划分时,按照计算强度划分任务,使得计算任务更加均衡。这些研究在并行矢量空间分析的任务划分、负载均衡和资源管理方面都做了很多有意义的工作,但是依然没有改变空间分析中线段求交点和构造结果集合耦合在一起的计算模式,交点的计算过程依然是矢量叠加分析效率的瓶颈。

1.2 矢量数据存储管理

HBase 是一个高可靠、高性能的分布式存储系统。由于 HBase 具有面向列存储、易于扩容等特性,被引入到地理信息系统领域,以期解决海量地理数据的高效存储、管理和发布的问题。文献[13]研究了基于 HBase 的矢量空间数据存储模型。文献[14]提出了并行构建网格空间索引的方法,有效地加快了索引构建的处理速度。文献[15]提出了 HBase 中的空间数据模型 HBasespatial,并和 MongoDB、MySQL 对比,空间查询的速度明显提高。这些研究工作为空间大数据的分布式存储查询提供了思路。

本文将矢量叠加分析算法与 HBase 做出了详细的特性对比,如表 1 所示。从表 1 可以看出,HBase 特征非常适合空间数据管理。依据本文提出的存储计算一体化的矢量叠加分析模式,设计了适合矢量地图叠加分析计算的矢量数据存储结构和图层交点存储结构,将在后文具体阐述。

表 1 矢量叠加分析算法和 HBase 特征对比

	矢量地图叠加分析算法	HBase 特点
数据特征	数据量大、数据增长快速;数据结构复杂,包含几何要素、属性信息等	分布式存储,可伸缩;适合存储非结构化数据、半结构化数据,列式存储方式
计算特征	计算密集、数据密集;分布式计算时,计算节点之间的网络 IO 开销成为瓶颈	通过行健对数据分片,接近平均的将数据存储在多个节点上;当计算任务落在对应的数据节点时能有效降低网络 IO 开销

2 叠加分析实时服务架构

如图 1 所示,本文设计了存储计算一体化模式下的叠加分析实时服务架构,该架构基于 HBase 管理和存储矢量数据和交点数据。存储计算一体化过程通过加载矢量数据并离线计算的方式完成:在图层加载的同时,计算出图层之间要素的交点,并将交点存储在 HBase 中。

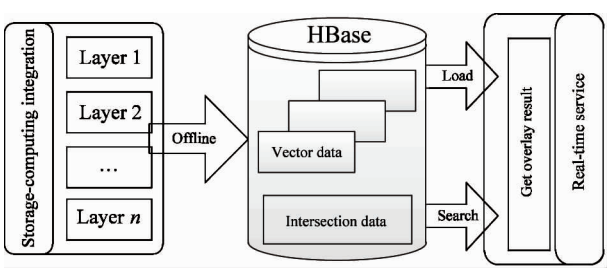

图 1 叠加分析实时服务架构

图层之间的交点一旦计算并存储,就可以在叠加分析时多次复用。当面对大量的叠加分析请求时,由于交点数据可以直接查询得到,整个算法过程变为加载图层要素、查询要素交点、构造结果集合 3 个步骤。在整个计算过程中,最耗时的计算交点的过程变成了查询交点,由此可以极大地提高算法的

效率。

3 线段交点存储计算一体化

3.1 线段交点数据组织管理

存储计算一体化的关键在于高效组织管理线段交点数据。本文根据线段交点的数据特点和叠加分析算法需要的信息,设计了图 2 所示的交点数据结构。

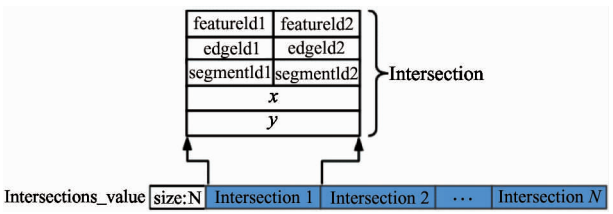

图 2 线段交点组织结构

矢量数据的叠加分析以空间要素的两两叠加作为最基本的计算单元。两个空间要素之间有可能有多个线段交点。图 2 中,Intersection 表示一个线段交点,为了完成叠加分析的计算,一个线段交点需要携带以下信息:(1)产生交点的两个空间要素 ID(featureId1,featureId2);(2)产生交点的两个边(edgeId1,edgeId2);(3)产生交点的两个线段(segmentId1,segmentId2);(4)线段交点的坐标(x, y)。图 2 中,Intersections_value 表示 2 个要素之间的线段交点集合,这个交点集合在叠加分析时会被整体使用,不会被割裂。所以将这些交点数据组织存储在一块连续的区域中,并且在区域头部存储线段交点的个数。这种交点组织结构既是叠加分析时交点在内存中的组织结构,同时也是线段交点的存储结构。

本文根据线段交点数据的组织结构,设计了 HBase 中的图层交点表,如表 2 所示。

表 2 HBase 图层交点表结构

Rowkey	Column Family: Qualifier	
Layer1 _ layer2	Intersection:i _j	Intersections _ value

从表 2 可以看出,一个存储单元(Cell)中存储一个图 2 结构的交点集合。在交点表中,以 2 个图层名拼接的字符串(layer1 _ layer2)作为行键(Rowkey),以 Intersections 作为列族名(Column Family),以产生交点的 2 个要素编号拼接的字符串(i _j)作为列名(Qualifier)。

在 HBase 中,一个存储单元由行健、列族名、列名共同确定,即 Cell = < Rowkey, Colunm Family, Qualifier > 。例如查询条件 < layer1 _ layer2, Intersections, i _j > 可以查询出 layer1 图层中编号为 i 的空间要素和 layer2 图层中编号为 j 的空间要素的交点集合。同时,也可以通过 Rowkey 一次取出两个图层之间的所有交点。

3.2 线段交点存储计算算法

线段交点的存储计算过程是在矢量数据存储到 HBase 的同时,离线完成的。在计算两个图层所有要素之间的交点时,先利用 RTree[16] 索引过滤不可能产生交点的要素对,然后再采用 BO[17] 算法计算交点,具体描述见算法 1。

算法 1 线段交点存储计算算法

输入:两个计算图层 *Layer*1 和 *Layer*2

输出: *Layer*1 与 *Layer*2 之间的交点

1. 首先对 *Layer*1 和 *Layer*2 中空间要素数多的一个图层建立 Rtree 索引(这里假设对 *Layer*1 建立 Rtree 索引)。
2. 依次获取 *Layer*2 中的空间要素,进行如下计算:
   2-1. 查询 Rtree 索引,在 *Layer*1 上找到可能与当前要素产生交点的空间要素。
   2-2. 将当前要素和查询到的要素中的线段求交点,并将交点根据表 2 的格式,存储到 HBase 图层交点表中。

算法 1 中 2-2 步所用到的线段求交点的算法为 BO 算法,算法时间复杂度为 $f(BO) = O((n + k)\log n)$,其中 n 为线段的个数,k 为交点个数。可以看出,BO 算法是一个数据敏感型算法,其算法时间复杂度与要素的交点个数相关,真实的计算时间难以预估。算法 1 的整体算法复杂度还依赖于 2-1 步骤中通过 RTree 的过滤操作来减少计算量,在最坏

情况下的时间间复杂度为 $O(m \times nf(BO))$，其中 m 和 n 分别为 2 个图层的要素数目。由于真实地理数据空间分布不均匀和空间聚集的特点，最坏情况一般不会出现，通过 Rtree 会过滤掉大量不会产生交点的要素对。

线段交点存储计算算法与传统算法相比，虽然在算法时间复杂度和效率上都没有改进，但通过离线计算存储的方式将这一过程从叠加分析中分离，会极大地提升叠加分析算法的效率。同时，由于交点结构简单，不会占用大量的存储空间，能够满足叠加分析服务的总体要求。

4 基于预存交点的矢量叠加分析算法

4.1 矢量数据存储结构

矢量数据中包含了空间要素的几何信息和属性信息，几何信息通常以 OGC(Open Geospatial Consortium)标准的 WKB(Well-known binary)格式组织，属性信息以(属性名:属性值)的 K-V 格式组织。本文设计了 HBase 中的矢量数据表结构存储矢量空间数据。

在 HBase 中，每个图层对应 1 张表，表结构如表 3所示。RowKey 为图层要素的编号 featureId，表结构中包含 2 个列族用以存储不同的信息。列族 Geo 主要存储空间要素的几何信息，列族 Attr 主要存储空间要素的属性数据。在叠加分析时，为了减少运算量，通常会采用 RTree 索引过滤不可能产生结果的要素对，本文算法的 Rtree 以空间要素的最小外包矩形(minimum bounding rectangle, MBR)构建。所以，为了更高效地构建 RTree 和过滤要素对，在 Geo 列族中，除了存储空间要素的 WKB 数据外，还存储 MBR 数据，这 2 类数据分别以 WKB 和 MBR 作为列名，数据存储在对应的存储单元中。空间要素的属性数据存储在列族 Attr 中，以属性名为列名，属性值存储在对应的存储单元中。

以 featureId 为 Rowkey 组织数据，可以快速地通过要素编号查询要素不同维度的数据。例如使用查询条件 $<i$, Geo, MBR $>$ 可以查询编号为 i 的空间要素的 MBR 数据；通过 $<i$, Attr, key _ 1 $>$ 可以取出空间要素 i 的 key _ 1 属性值；当然也可以通过要素 *id* 直接取出空间要素的所有数据。

表 3 HBase 中矢量数据的存储结构

RowKey	ColumnFamily:Qualifier		ColumnFamily:Qualifier	
featureId	Geo:MBR	MBR _ Value	Attr:key _ *i*	ATTR _ I _ Value
	Geo:WKB	WKB _ Value		

4.2 并行矢量叠加分析算法

在矢量叠加分析算法的并行实现中，本文采用线程池和任务队列结合的方式管理线程和分配计算任务。面面叠加分析算法是矢量叠加分析算法中最复杂、也是最具有代表性的一个算法，本文以面面叠加求交为例，具体描述如算法 2。

算法 2 并行矢量叠加算法

输入：参与叠加分析的矢量地图 *Layer*1 和 *Layer*2

输出：叠加分析的结果要素集合

1. 加载矢量数据：从 HBase 中取出 *Layer*1 和 *Layer*2 的所有空间要素。

2. 加载交点数据：从 HBase 中以 *Layer*1 _ *Layer*2 为 RowKey，查询两个图层的所有要素交点。

3. 建立索引：选取 *Layer*1 和 *Layer*2 中要素数多的图层，建立 Rtree 索引(这里假设为 *Layer*1 建立索引)。

4. 构造叠加结果：依次取 *Layer*2 中的空间要素 *geo*2，进行如下操作：

4-1. 以 *geo*2 的 MBR 查询 Rtree 索引，找到 *Layer*1 中所有可能产生结果的空间要素 *geos*1。

4-2. 将数据 *geo*2 和 *geos*1 放入任务队列，等待线程池调度空闲线程进行计算。

算法 2 中 4-2 步为并行计算的步骤，本文采用

线程池和任务队列的方式进行线程调度和任务分配。线程池中存在一定数量的线程,当某一线程的计算任务结束并且空闲时,线程池会在任务队列中取出一个没有完成的任务分配给该线程,让其继续进行计算,直到任务队列为空。

线程池中的每个线程完成相同的操作,即计算任务队列中的空间要素的叠加结果。承接算法 2,线程计算 *geo*2 和 *geos*1 的叠加结果如算法 3 所示。

算法 3 空间要素叠加算法

输入:空间要素 *geo*2 和空间要素集合 *geos*1

输出:*geo*2 和 *geos*1 的叠加结果

1. 依次取 *geos*1 中的每一个空间要素 *geo*1

1-1. 根据 *geo*1 和 *geo*2 的空间要素 id:*featureId*1 和 *featureId*2 查询两个要素的交点。

1-2. 如果存在交点,则根据交点信息将交点插入到空间要素中,将要素对应的环打断。

1-3. 构造 *geo*1 和 *geo*2 的叠加结果。

算法 3 中的 1-1 步通过查询条件 < layer1 _ layer1, Intersections, *featureId*1 _ *featureId*2 > 直接从 HBase 的交点信息表中查询出 2 个空间要素的交点,这样就避免了在叠加分析时计算要素之间的交点,大幅减少了叠加分析的时间。由于本文在设计线段交点的组织结构(图 2 所示)时,充分考虑了叠加分析计算的需求,所以在 1-2 步中,可以通过交点信息快速地定位出交点在空间要素中的位置,并将其插入到空间要素中,打断交点对应的环,为构造叠加结果做准备。步骤 1-3 采用的叠加算法为 Weiler-Atherton[18]算法,其时间复杂度为 $f(WA) = O(n + k)$,其中 k 为交点数目,n 为 2 个空间要素中顶点数之和。

5 实验结果与分析

本文实验分为 3 组,实验数据基本信息如表 4 所示。实验选用的机器的 CPU 配置为:Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40 GHz × 2,64 GB 内存,操作系统版本为 Ubuntu 14.04,程序运行时环境为 JDK1.8。

表 4 实验数据说明

数据集	要素个数	数据类型	数据说明
Area	1	面要素	包含复杂地理条件的人工数据
Railways	1 484	线要素	全国铁路路网
Provinces	46	面要素	部分省界行政图
Counties	2 449	面要素	部分县界行政图
LandUse	122 552	面要素	全国土地利用数据

5.1 算法鲁棒性验证实验

由于真实的地理环境非常复杂,在图层叠加时,会出现很复杂的线段交叠情况,而能否正确处理这些复杂的情况是算法鲁棒性的一个标志。在实验 1 中,本文首先假设工业界普遍使用的 ArcGIS 分析的结果为正确结果,然后选用相同的测试数据,对比 ArcGIS 和本文算法的结果,以便分析本文算法的鲁棒性。

在实验时,选用的数据集为 Area 和 Counties,其中 Counties 为真实的地理数据,Area 不是简单的多边形数据,而是是包含了大量复杂地理边界条件(内含湖泊、岛屿等)的人造数据。这组测试数据能更加客观地说明本文算法的鲁棒性。实验结果如图 3所示,图 3(a),3(b)分别为两个测试数据,图 3(c)为本文算法结果,图 3(d)为 ArcGIS 分析结果。2 个结果图层对比得到,2 个结果图层要素个数一致(均为 686 个),且 2 个结果图层在 ArcGIS 中对称差结果为 0,说明 2 个结果图层要素边界信息完全一致。

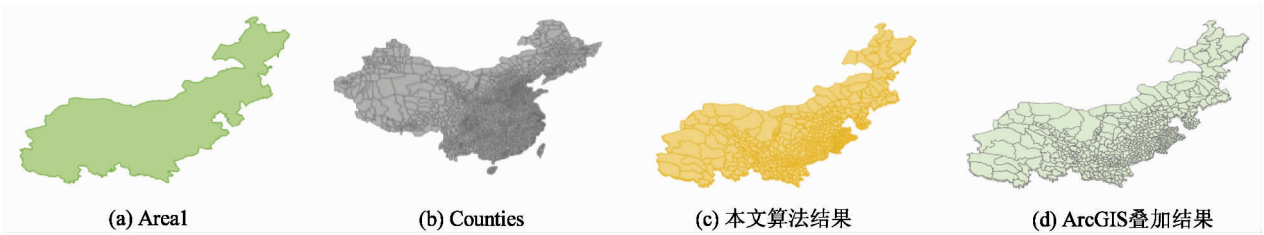

图 3　实验 1 数据

5.2　算法并行效率验证实验

实验 2 中选用 Counties 和 LandUse 2 个数据集，用来分析在较大数据量情况下本文算法相较于没有预存交点的算法在效率上的提升以及算法的并行效率。用于实验的两个数据集均为真实的地理数据，叠加后的结果图层中包含了157 622个要素，表 5 中分别统计了算法 2 和没有预存交点的算法在 2 个数据集上的叠加分析时间。

表 5　实验 2 计算时间统计表(s)

本文预存交点算法(算法 2)						没有预存交点的算法
线程数	数据加载	查询交点	构建 RTree 索引	构造结果集	总时间	总时间
1	2.25	1.24	0.13	56.69	60.32	779.73
2	2.19	1.30	0.15	29.40	33.06	
3	2.12	1.31	0.10	22.58	26.12	
4	2.03	1.33	0.11	17.17	20.65	

在实验中，没有预存交点的算法采用单线程计算，与本文提出的算法在单线程下进行对比。从表5中的数据可以看出，本文方法所用的时间要明显少于没有预存交点的算法所用的时间，叠加分析的用时相较没有预存交点的算法减少 92.4%。叠加分析时间的大幅度减少符合本文方法设计的初衷，把耗时的线段求交运算从叠加分析中分离出来，通过离线计算的方式提前计算线段交点并存储，使得在叠加分析时只需要查询交点数据就可以构造出结果集合。

如表 5 数据所示，算法 2 中构造结果集合的步骤为主要的计算步骤，用时最多。本文算法中，通过将这一步并行化来减少分析用时。随着线程数的增多，构造结果集合的时间在逐渐减少；而其他步骤，由于没有并行化，所以耗时基本没有变化。

由于算法 2 中存在没有并行计算的步骤，同时，还有线程调度等的影响，实际的加速比无法达到线性加速比。从图4中可以看出，虽然随着线程数目

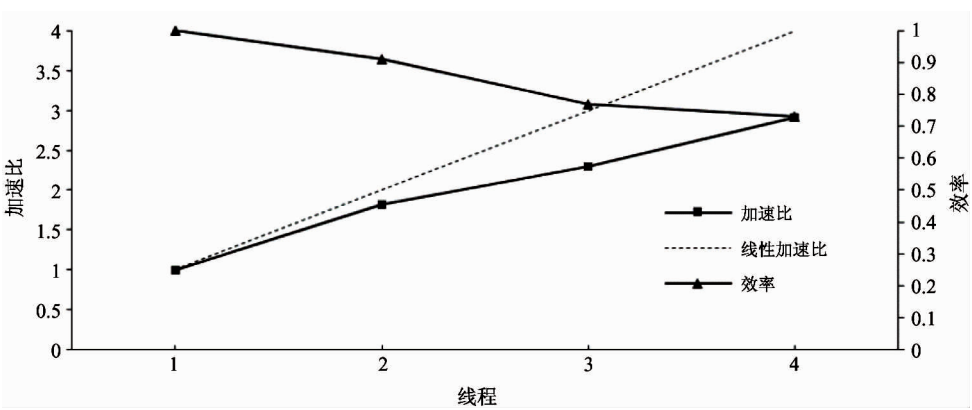

图 4　加速比及并行效率

的增多,算法的加速比与线性加速比的差距变大,算法的并行效率降低,但加速比依然比较理想,算法的并行效率在73%以上。

5.3 算法额外空间占用率分析

由于本文算法采用以空间换时间的策略提升算法的效率,所以额外的空间占用率也是衡量算法好坏的一个重要指标。没有预存交点的算法不需要存储交点信息。在资源占用方面,本文算法与没有预存交点的算法相比,需要存储和管理额外的交点信息,所以将交点数据的空间占用率作为衡量本文算法额外空间占用率的指标。从上文叙述可知,本文设计的交点结构简单,每个交点占40Byte的存储空间(交点结构中,交点的坐标为float类型,其余均为int类型)。本实验选取4个真实的地理数据集Railway、Provinces、Counties和LandUse,通过统计数据集两两之间的交点数目,确定本文算法的额外空间占用量。

表6分别统计了多组叠加分析中,原始数据集占用的存储空间大小(*DatasetSize*)和交点数据占用的存储空间大小(*InterctionsSize*),并计算了交点的额外空间占用率,计算公式如下。

$$ratio = \frac{InterctionsSize}{DataSetSize} \times 100\% \quad (1)$$

从表6可以看出,不同的图层之间,交点额外空间占用率变化波动较大(0.23~9.24%),这与数据集的数据分布有关。但交点的额外空间占用率整体都低于10%,甚至大部分都低于1%。算法的额外空间开销较小,能满足存储和计算的要求。

表6 数据存储量统计

叠加数据集	数据集存储大小(MB)	交点存储大小(MB)	交点额外空间占用率(%)
Railway & Counties	39.5	0.3	0.78
Railway & Provinces	9.1	0.02	0.23
Railway & LandUse	97.5	0.8	0.82
Provinces & LandUse	101.6	1.8	1.80
Provinces & Counties	43.6	0.3	0.69
Counties & LandUse	132.0	12.2	9.24

6 结 论

本文提出的空间分析算法能明显地提高空间分析的效率,并且本文方法能处理复杂地理环境的空间要素,算法高效鲁棒。算法在多线程环境下的加速效果明显。本文提出的空间分析算法为大数据环境下的空间分析实时性服务提供了新的解决方案。因此,将本文提出的空间分析算法做主流分布式计算平台的适配,以便对空间大数据提供有效的分析服务是下一步工作的重点。

参考文献

[1] Qiu Q, Yuan M, He F, et al. A parallel algorithm for line segment intersection[C]. In: Proceedings of the 21st International Conference on Geoinformatics (GEOINFORMATICS), Melbourne, Australia, 2013. 1-4

[2] 李荣亚. 双态云支持下高分辨率遥感存储与计算一体化研究[D]. 杭州:浙江大学地球科学系, 2014

[3] 邱强, 秦承志, 朱效民, 等. 全空间下并行矢量空间分析研究综述与展望[J]. 地球信息科学学报, 2017, 19(9): 1217-1227

[4] 邱强, 方雷, 姚晓, 等. 基于空间聚类的矢量空间数据并行计算划分方法[J]. 高技术通讯, 2015, 25(4): 327-33

[5] 赵斯思, 周成虎. GPU加速的多边形叠加分析[J]. 地理科学进展, 2013, 32(1): 114-20

[6] 朱效民, 潘景山, 孙占全, 等. 基于OpenMP的两个地学基础空间分析算法的并行实现及优化[J]. 计算机科学, 2013, 40(2): 8-11+39

[7] Magalhaes S V, Andrade M V, Franklin W R, et al. Fast exact parallel map overlay using a two-level uniform grid [C]. In: Proceedings of the 4th International ACM SIGSPATIAL Workshop on Analytics for Big Geospatial Data, Seattle, USA, 2015. 45-54

[8] Puri S, Agarwal D, He X, et al. MapReduce algorithms for GIS polygonal overlay processing[C]. In: Proceedings of the IEEE 27th International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), Boston, USA, 2013. 1009-1016

[9] Whitman R T, Park M B, Ambrose S M, et al. Spatial indexing and analytics on hadoop[C]. In: Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Forth Worth, USA, 2014. 73-82

[10] Yao X, Qiu Q, Zhang M, et al. Research on vector spatial data access based on main memory database[C]. In: Proceedings of the 2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Milan, Italy, 2015. 4704-4707

[11] Xie D, Li F, Yao B, et al. Simba: Efficient in-memory spatial analytics[C]. In: Proceedings of the 2016 International Conference on Management of Data, San Francisco, USA, 2016. 1071-1085

[12] Guo M, Guan Q, Xie Z, et al. A spatially adaptive decomposition approach for parallel vector data visualization of polylines and polygons[J]. *International Journal of Geographical Information Science*, 2015, 29(8): 1419-1440

[13] Wang Y, Li C, Li M, et al. HBase storage schemas for massive spatial vector data[J]. *Cluster Computing*, 2017(2): 1-10

[14] 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

[15] Zhang N, Zheng G, Chen H, et al. Hbasespatial: A scalable spatial data storage based on hbase[C]. In: Proceedings of the 13th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Beijing, China, 2014. 644-651

[16] Guttman A. R-trees: A Dynamic Index Structure for Spatial Searching [M]. ACM, 1984

[17] Bentley J L, Ottmann T A. Algorithms for reporting and counting geometric intersections[J]. *IEEE Transactions on computers*, 1979, 28(9): 643-647

[18] Weiler K, Atherton P. Hidden surface removal using polygon area sorting[C]. In: Proceedings of the ACM SIGGRAPH Computer Graphics, New York, USA, 1977. 214-222

A vector spatial overlay analysis algorithm based on pre-storing intersections

Xiao Zhuojian[*,**], Qiu Qiang[*], Yao Xiao[*,**], Fang Jinyun[*]

([*] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

([**] University of Chinese Academy of Sciences, Beijing 100190)

Abstract

Vector spatial overlay analysis is a key operation in the geographic information system (GIS). Recently, it is challenging to provide real-time service, especially for large scale vector spatial data. Therefore, an overlay algorithm based on pre-storing information of intersections is proposed. The structure of intersection for storage and computation integration is designed in vector map overlay algorithm. It can satisfy the storage requirement of low storage space overhead, and meet the computation requirement of intersections access. This work calculates and stores the information of intersections offline. Then, it gets the information of intersections by querying instead of calculating, which is different from the traditional method, effectively reduces the overhead of intersection calculation with less storage cost. This method greatly improves the efficiency of real-time special analysis without impairing the correctness of analysis results. The experiment shows that this work reduces the overlay algorithm time by 92.4%, with the overhead of the storage space less than 10% compared with the traditional algorithm. Besides, the proposed method can effectively improve the speedup and parallel efficiency.

Key words: overlay analysis, spatial data management, pre-storing intersections, storage and computation integration