doi:10.3772/j.issn.1002-0470.2018.06.003

Spark 框架下矢量多边形求交算法研究①

姚 晓②*** 邱 强* 肖茁建*** 方金云* 崔绍龙***

(*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)

(***中国科学院遥感与数字地球研究所 北京 100094)

摘 要 提出一种分布式内存计算框架 Spark 下的矢量多边形求交算法,解决了大数据环境下并行矢量多边形求交计算过程中网络数据传输成本高、冗余计算量大的问题。该算法根据空间填充曲线构建空间网格分区,并利用多边形最小外包矩形(MBR)进行网格填充,以传输 MBR 代替传统算法中直接传输多边形几何体的过程,减少了算法的网络数据传输量。针对复杂多边形跨越多个网格分区的场景,提出一种跨区数据交点定位策略,从而消除跨区多边形的冗余计算。实验结果表明,本文方法能够显著提高并行矢量多边形求交算法的计算效率。

关键词 Spark, 多边形求交, 最小外包矩形(MBR), 交点定位

0 引 言

近年来,随着遥感技术、移动计算技术的快速发展,空间数据如对地观测数据、LBS 签到数据等出现了爆炸性增长。空间数据规模海量化、数据格式多样化、数据处理实时化等需求给传统的空间分析算法带来极大的挑战,如何有效地利用大数据手段在云计算环境下提供高效快速的空间分析服务成为高性能空间分析领域的研究热点之一。

多边形求交算法作为矢量空间叠加分析算法中的代表性操作,是指将不同图层的矢量多边形按照一定的规则进行求交计算产生新矢量多边形的计算过程。由于图层中的多边形几何体本身结构复杂、空间占用高,在并行过程中直接进行传输导致网络数据量过大,影响算法的总体效率。另外,在处理面向网格分区的空间数据时,跨越多个分区的多边形会产生大量的冗余计算,同样影响算法的效率。针对此问题,本文在分布式计算框架 Spark 和分布式缓存下,充分利用弹性分布式数据集(resilient distributed datasets,RDD)的特性以及内存计算的高效性,提出基于 Spark 的矢量多边形求交算法。该算法首先基于空间填充曲线,利用多边形最小外包矩形(minimum bounding rectangle, MBR)对空间网格进行填充并构建网格内空间索引,通过传输 MBR 以减少网络传输数据量;其次根据跨区数据交点定位策略消除跨越多个分区的重复多边形,从而在叠加求交计算过程中消除冗余计算,提高算法整体性能。实验结果表明,本文方法极大地提高了矢量多边形求交算法的效率,能够满足大数据环境下矢量空间叠加分析算法的性能要求。

1 背景和相关研究

1.1 Spark 分布式框架

Spark 是一种基于内存的分布式并行计算框架,

① 国家重点研发计划(2016YFB0502300,2016YFB0502302)和国家自然科学基金(NO.41471430)资助项目。

② 男,1990 年生,博士生;研究方向:并行空间分析,空间数据存储与管理,搜索引擎技术等;联系人,E-mail: yaoxiao@ict.ac.cn

(收稿日期:2017-12-20)

该框架利用内存迭代计算,提高了数据处理的实时性,同时保证了高容错性和高可伸缩性。不同于 Hadoop 在 shuffle 运行阶段需要将临时数据写回到外存,Spark 将数据存放在内存,完全基于内存进行计算,其性能要比 Hadoop 快 10 到 100 倍。Spark 计算框架以 RDD 作为内存数据模型,所有的计算都是围绕 RDD 进行,在 Spark 任务的执行过程中,RDD 经过一系列 Transformation 算子之后,最后通过 Action 算子触发执行操作,完成整个计算过程。Spark 将需要重复使用的数据或者中间结果常驻内存,以提高下次使用该数据的读取效率,因此,Spark 特别适合矢量多边形求交分析这种一次读入、多次计算的迭代型分析算法。

1.2　相关研究

矢量多边形求交分析算法作为地理信息系统(geographic information system, GIS)的核心算法,其性能决定着整个空间分析系统的服务能力。传统的针对矢量多边形求交分析性能优化的研究大都围绕基于 OpenMP、MPI、CUDA 等框架的算法并行化改造进行展开[1-4],算法性能能够得到很大的提升,但由于其并行框架扩展性不足、容错能力差的缺陷,使得算法在海量空间数据的挑战下,仍然难以提供高效稳定的计算与服务。近年来,基于分布式计算平台的 GIS 空间分析相关研究成为新的研究热点,但大都集中在空间数据的存储与管理、空间查询等方面,针对空间叠加分析方面的研究相对较少。钟运琴等[5]基于 Hadoop 平台实现了大规模时空数据的分布式存储原型系统 extHDFS,取得了较高的 I/O 存取性能、系统吞吐率和系统扩展性能。SpatialHadoop[6]通过扩展 Hadoop 的核心模块,实现了 Hadoop 平台下的空间索引、空间数据 MapReduce 组件以及一系列空间操作,并提供 Hadoop 平台对于空间数据存储与计算的原生支持。Whitman 等[7]将 PMRquadtree 在 Hadoop 平台下进行适配,提出了一种不需要 MapReduce 任务和全表扫描的分布式空间查询方法。Puri 等[8]利用网格划分策略以及分布式缓存实现了 MapReduce 计算模式下的分布式空间叠加分析算法,但该方法将多边形数据用于网格填充,有着较大的数据划分代价以及 MapReduce 任务进行 shuffle 操作的数据传输成本。GeoSpark[9]实现了基于 Spark 的内存空间数据模型 SRDD 以及空间数据划分策略,相较于 SpatialHadoop 有着更好的空间数据查询处理性能,但其二级网格的数据划分策略存在数据的跨区复制问题,需要在子任务结束之后进行去重操作,影响系统的总体性能。Tang 等[10]实现了基于 Spark 的空间数据管理系统,较好地解决了分布式空间分析中查询倾斜以及网络通讯过载等问题。靳凤营等[11]提出了基于 Spark 的土地利用矢量数据叠加分析方法,该方法将原始数据以及离线创建的索引文件存储在 HDFS,计算时直接读取索引文件进行过滤计算,有效地提高了叠加分析算法的效率,该方法通过暴力方式将叠加图层的每一个空间对象与包含所有被叠加图层空间对象的索引进行过滤,然后进行计算,扩展性较差。

2　Spark 框架下的多边形求交算法

2.1　基于多边形 MBR 的数据分配策略

基于 Spark 的矢量多边形求交计算首先要进行任务分解,即数据划分,良好的数据划分策略是算法负载均衡性的保证。矢量多边形数据不同于普通的文本数据、日志数据,其包含的空间多边形结构复杂、大小不一、空间相关性强以及数据分布不均的特点容易导致分布式计算各个任务间产生较大的数据倾斜。例如我国行政区域划分,东南沿海与西部地区相比,数据量明显更多。根据地理学第一定律,空间对象在空间位置上具有越临近越相关的特性,在进行数据划分时,需要尽量将空间上较为临近的矢量多边形划分到同一个计算任务当中,这样只需要对同一任务分区内的数据进行外包过滤,可以极大地减少矢量多边形外包过滤的次数。为了保证划分到同一分区矢量多边形的空间临近性,减少数据倾斜,本文空间划分网格基于 Hilbert 空间填充曲线进行构建,每一个空间划分网格采用其中心点的 Hilbert 编码值作为唯一标识。

多边形及其 MBR 如图 1 所示,其在内存中的存储结构可以分别表示为:

polygon:[$<x_1,y_1>$, $<x_2,y_2>$, $<x_3,y_3>$,

$< x_4, y_4 >$, $< x_5, y_5 >$, $< x_6, y_6 >$],

MBR: [$< x_1, y_5 >$, $< x_4, y_2 >$]。

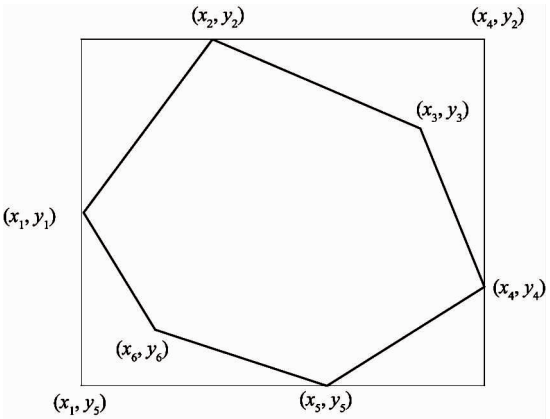

图 1 多边形与 MBR

可以看出矢量多边形的 MBR 无论在空间占用还是复杂程度上都远远小于所包含的多边形。因此,本文选取多边形 MBR 作为基本单元进行网格填充,这样不仅可以减少网格分区内数据过滤的计算量,而且 MBR 较小的空间占用也会降低算法进行 shuffle 操作时的网络数据传输成本。进行多边形 MBR 网格定位时,本文不同于文献[12]采用 MBR 中心点对多边形 MBR 进行 Hilbert 网格定位,该方式对于落在网格边界附近的多边形难以保证空间临近性。本文采用基于求交包含检测的方式将多边形 MBR 分配到具体的网格中,即将多边形 MBR 分配到与其存在相交、相邻或者包含关系的所有网格中。其处理逻辑如下:网格: [$< x_{g1}, y_{g1} >$, $< x_{g2}, y_{g2} >$], 多边形 MBR: [$< x_{m1}, y_{m1} >$, $< x_{m2}, y_{m2} >$]。

网格包含 MBR 如图 2(a)所示,其判断逻辑为:

$$b1 = x_{g1} \leqslant x_{m1} \&\& x_{g2} \geqslant x_{m2} \&\& y_{g1} \leqslant y_{m1} \&\& y_{g2} \geqslant y_{m2} \quad (1)$$

MBR 包含网格如图 2(b)所示,其判断逻辑为:

$$b2 = x_{m1} \leqslant x_{g1} \&\& x_{m2} \geqslant x_{g2} \&\& y_{m1} \leqslant y_{g1} \&\& y_{m2} \geqslant y_{g2} \quad (2)$$

网格与 MBR 相交如图 2(c)、(d)所示,其判断逻辑为:

$$b3 = !(x_{m1} \geqslant x_{g2} \parallel x_{m2} \leqslant x_{g1} \parallel y_{m1} \geqslant y_{g2} \parallel y_{m2} \leqslant y_{g1}) \quad (3)$$

因此,多边形 MBR 能否被分配到该网格,根据布尔值:

$$b1 \parallel b2 \parallel b3 \quad (4)$$

来进行判断。此外,在进行网格填充的过程中,会产生同一个多边形 MBR 被定位到多个网格的跨区重复分配问题,具体处理方式将在 2.2 节进行详细描述。

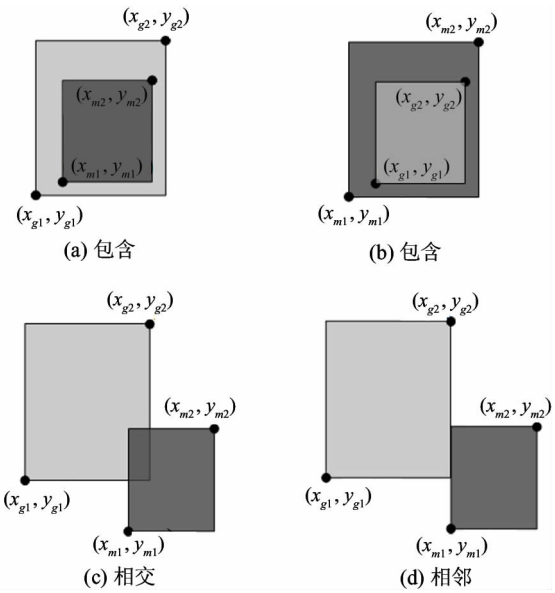

图 2 MBR 与网格位置关系

空间网格填充结束之后,为了在计算阶段加快过滤不相交的 MBR 对,需要对被叠加图层多边形 MBR 建立分区内索引,这里选取查询效率较高的 R-tree[13] 作为空间索引的数据结构,为了减少空间占用,其索引节点存储多边形 MBR 和多边形唯一标识 ID。算法 1 的伪代码描述了分区索引的构建过程。

算法 1:分区索引构建

```
输入:经过网格划分的 MBR: <hid, MBR>,其中 hid 表示网格分区的 Hilbert 编码
输出:分区 R-tree 索引: <hid, rtree>
<hid, MBRs> = <hid. MBR>. groupByKey()
<hid, MBRs>. repartition();
for each partition: p of <hid, MBRs>. partitions do
    build local R-tree: R
    for s in p. values() do
R. insert(s. MBR, s. MBR. ID)
end for
    emit <hid, R>
end for
```

2.2　跨区数据交点定位策略

Spark 框架下的矢量多边形求交算法,多边形 MBR 在进行数据划分过程中,存在跨区多边形 MBR 被分配到多个分区的问题,因此在求交计算阶段,会产生大量的冗余计算,如果不进行处理,就需要在各个分区任务结束之后进行去重操作,进一步增加系统的开销,降低算法整体性能。针对此问题,本文提出一种基于多边形 MBR 的跨区数据交点定位策略,通过计算两个多边形 MBR 的交点,并以左下交点所在的分区作为其最终进行多边形求交计算的网格分区。如图 3 所示,跨区多边形 MBR 可分为以下 3 种情况:图 3(a)中为相交关系、图 3(b)为包含关系、图 3(c)为相邻关系。以图 3(a)为例,来自叠加图层的 MBR:B1 被分配到分区 2、7、8、13,来自被叠加图层的 MBR:O1 被分配到分区 2、13,如果不进行冗余计算消除操作,则 B1 和 O1 所包含的多边形会分别在分区 2 和 13 进行计算,多次相同的计算会导致算法效率的下降。通过跨区数据交点定位的策略,B1 和 O1 进行求交计算得出的左下交点位于分区 2,因此 B1 和 O1 包含的多边形将在分区 2 进行具体的求交计算。同理图 3(b)和图 3(c)中 B2 与 O2、B3 与 O3 所包含的多边形求交计算任务也会只分配到分区 7 和分区 13 中进行具体的计算。

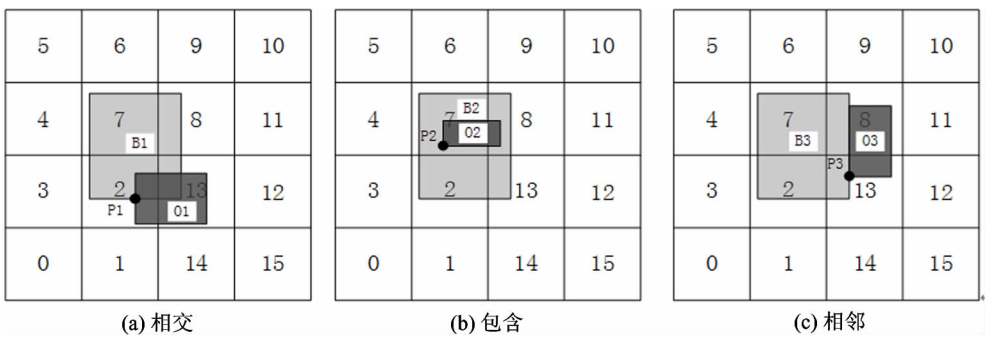

图 3　多边形 MBR 位置关系

2.3　多边形求交算法流程

本文提出的 Spark 框架下的矢量多边形求交算法流程如图 4 所示,可分为以下 5 个步骤:

步骤 1 输入图层进行多边形的 MBR 抽取;

步骤 2 基于其中一个图层(这里选择被叠加多边形图层)进行 Hilbert 网格构建,然后在数据分配阶段将图层的 MBR 集合填充到各网格分区,形成网格分区 MBR 集合;

步骤 3 对被叠加图层构建网格内 R-tree 索引,形成网格分区索引;

步骤 4 将被叠加图层的网格分区索引与叠加图层的网格分区 MBR 集合调用 zipPartition 算子进行 join 操作,并基于跨区数据交点定位策略进行跨区 MBR 去重;

步骤 5 各个网格分区从分布式缓存读取多边形数据进行求交计算,并输出结果。

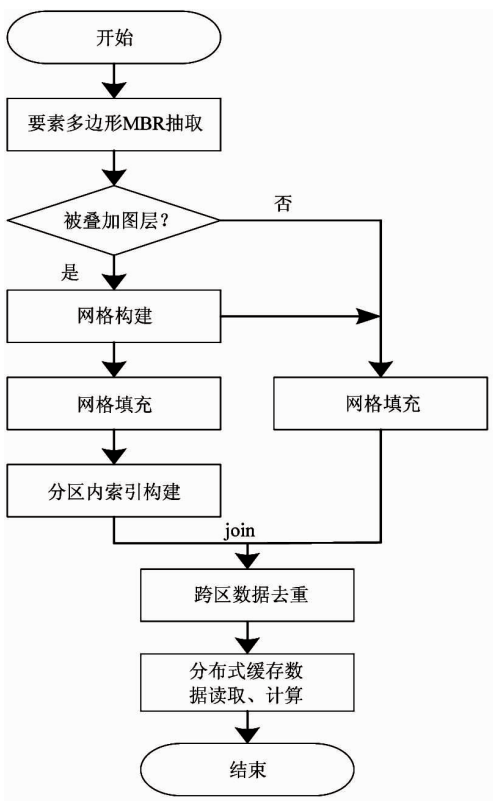

图 4　算法流程

3 实验与分析

3.1 实验环境

本文采用的实验环境由 7 台机架式服务器组成(node1 ~ node7),每台机器的硬件配置为:Intel XeonCPU 2.67GHz,内存为 64GB DDR3;外存为 1TB 7200RPM SATA 硬盘;采用千兆以太网连接各个节点。每个节点均部署 CentOS Linux release 7.2.1511、hadoop-2.7.0、Spark-2.2.0、JDK-1.8.0_65、Redis-3.2.6,其中 node1 作为 Spark 集群和 Hadoop 集群的主节点。

实验数据集包括真实地理数据集和经过抽稀、平移和复制的大规模地理数据集。真实地理数据集为包含 122552 个多边形的全国土地利用数据和 2448 个多边形的全国行政区划数据。大规模地理数据集为 1525568 个多边形的土地利用数据和包含 50816 个多边形的仿真数据。

3.2 评估指标

$$\text{冗余计算率:}\ \frac{M-N}{N}\times 100\% \tag{5}$$

其中,M 表示数据分配之后相交 MBR 的个数,N 表示实际相交 MBR 的个数。

$$\text{加速比:}\ S=\frac{T_s}{T_p} \tag{6}$$

其中,T_s 表示单线程串行算法总时间,T_p 表示基于本文方法的算法总时间。

$$\text{并行效率:}\ E=\frac{S}{p} \tag{7}$$

其中,p 表示并行度。

3.3 不同数据分配策略算法性能对比

实验 1 选取全国行政区划数据和土地利用数据进行求交计算,对比基于多边形 MBR 和基于多边形几何体 2 种数据分配策略下的算法性能。图 5 是网格分区数为 36,通过设置并行度,2 种策略下的算法时间对比。可以看出本文方法相较于基于多边形填充的策略有着明显的优势,其算法总时间减少 20.12% ~57.64%。这是由于一方面多边形相较于 MBR 有着更高的空间占用,因此在网络传输过程中,会有较高的时间延迟;另一方面,多边形数据结构比 MBR 复杂,因此在进行网格定位时会有更多的时间消耗。基于多边形几何体数据分配策略下的算法在并行度为 36 时算法性能达到最优,随着并行度个数的继续增加,开始出现算法总时间上升的趋势。这说明并行度不能设置过大,过大反而造成资源浪费以及算法性能下降。而本文基于多边形 MBR 数据分配策略下的算法性能则有着进一步的提升空间,在并行度为 54 时性能达到最优,并且相较于基于多边形的算法仍然有着较高的性能优势。

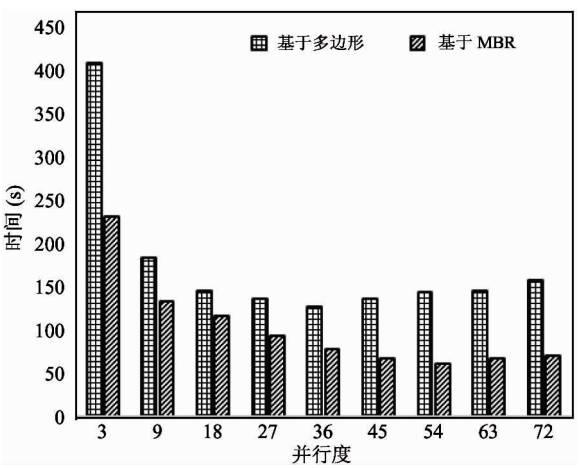

图 5 计算效率对比

3.4 冗余计算率对比

在跨区数据交点定位策略的验证实验中,选取全国行政区划数据和土地利用数据进行求交计算,分别统计在不同网格数的情况下,经过该策略和不使用该策略情况下的相交 MBR 被计算的次数,并给出冗余计算率和算法性能优化效果对比。

实验 2 首先如图 6 所示,为网格划分个数从 36 到 4900 的划分情况下,跨区 MBR 冗余计算率变化曲线。可以看出随着网格数量的不断增多,跨区数据的冗余计算率不断增加,即相同数据的重复计算次数不断增加。这是由于随着网格数的增加,每个分区网格越来越小,多边形 MBR 就更容易被划分到多个网格中。而本文方法无论网格数为多大,多边形只能被分配到一个网格分区,进而只会被计算一次,因此冗余计算率都是 0%。

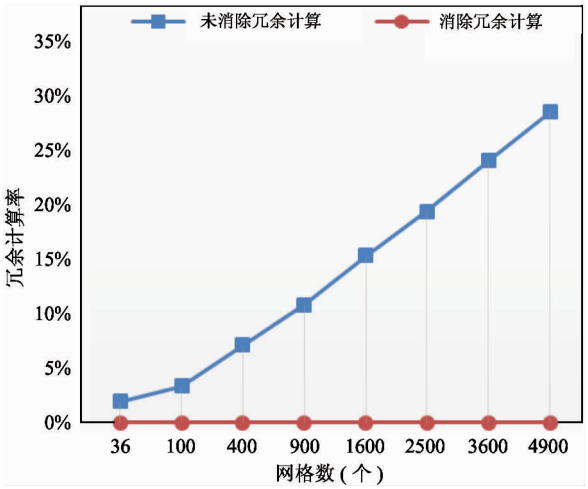

图 6　冗余计算率

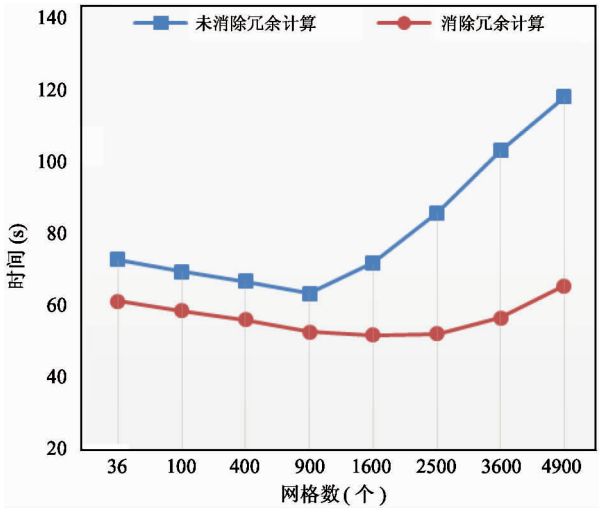

图 7　算法时间

其次对于算法性能的优化效果,如图 7 所示。经过跨区数据交点定位策略消除冗余计算之后,算法的性能有了明显提升,并且随着网格数的不断增加,优化效果会越来越明显。此外,从图 7 中可以发现,随着网格数的不断增加,算法出现性能下降的趋势。分析可知是由于网格数越多,进行数据分配的成本就会越来越大,两个多边形图层进行 join 操作的时间也会越长,因此并行算法需要选取合理的网格区间进行设置。

3.5　算法加速比和并行效率

本文通过实验 1 和实验 2 分别验证了两种策略对于并行多边形求交算法有着明显的优化效果,因此实验 3 综合 2.1 节和 2.2 节两种策略,在百万级数据规模下进行多边形求交实验。并行算法的加速比和并行效率是算法并行化程度的重要考量,但由于串行算法并行化过程中会带来数据划分以及通讯成本的增加,使得并行算法很难达到理想的加速比。图 8 为本文算法的加速比以及并行效率随并行度的

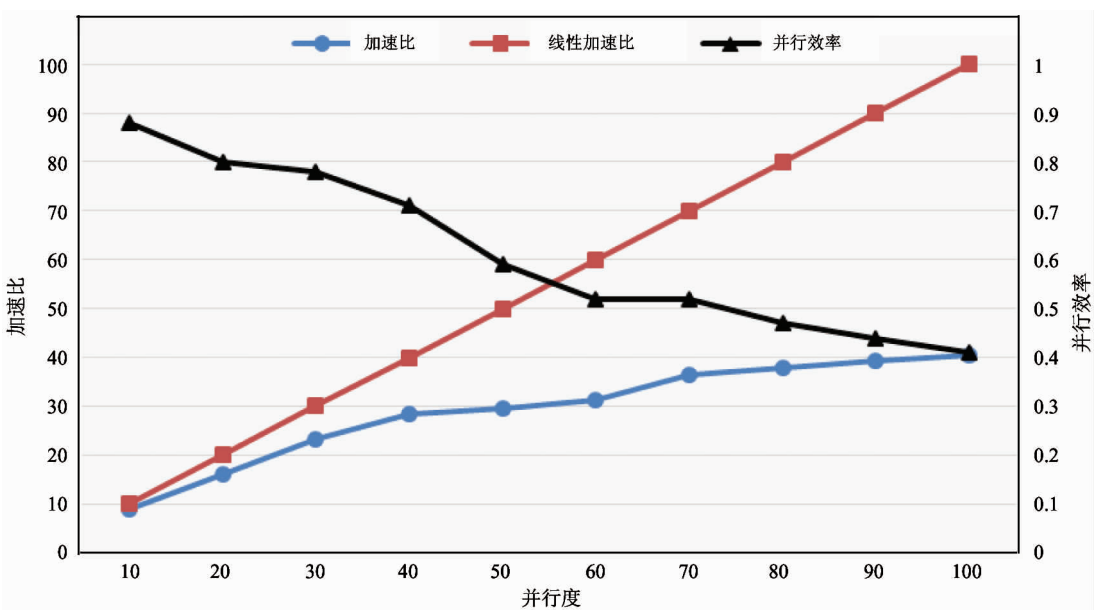

图 8　算法加速比和并行效率

变化情况,可以看出本文方法有着较好的加速比,并行效率最高可达到0.88。但随着并行度的不断提高,算法加速比以及并行效率下降越来越明显。分析可知是由于随着并行粒度的增加,计算任务之间计算任务量的不均衡性越来越明显,导致算法总体性能的进一步提升幅度有限,因此,下一步需要开展针对计算量预估的负载均衡性研究工作。

4 结 论

本文针对现有云计算环境下并行多边形求交算法存在的网络数据传输量大以及冗余计算量高的问题,提出了基于多边形 MBR 的数据分配策略以及跨区数据交点定位策略,并充分利用 Spark 内存计算的高效性以及分布式缓存的快速读写特性,实现了 Spark 框架下的矢量多边形求交算法。实验结果表明,本文方法相对于基于多边形几何体进行数据分配的策略有着更高的算法性能,并且扩展性得到进一步的提升。因此,本文提出的基于 Spark 的矢量多边形求交算法能够满足当下海量空间数据的实时分析需求。

空间数据结构复杂,基于空间填充曲线的数据划分仅仅能够实现针对空间位置和数据量大小进行考量的负载均衡策略,对于不同空间对象的计算复杂度难以实现较为准确的预估,因此,针对分布式环境下,基于计算复杂度预估的算法负载均衡性研究将是下一步的工作重点。

参考文献

[1] Akhter S, Aida K, Chemin Y. Grass GIS on high performance computing with MPI, OPENMP and NINF-G programming framework[C]. In: Proceedings of the International Archives of the Photogrammetry, Remote Sensing and Spatial Information Science, Kyoto, Japan, 2010. 580-585

[2] Puri S, Prasad S K. A parallel algorithm for clipping polygons with improved bounds and a distributed overlay processing system using MPI[C]. In: Proceedings of IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, Shenzhen, China, 2015. 576-585

[3] Audet S, Albertsson C, Murase M, et al. Robust and efficient polygon overlay on parallel stream processors[C]. In: Proceedings of ACM Sigspatial International Conference on Advances in Geographic Information Systems, Orlando, USA, 2013. 304-313

[4] 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

[5] 钟运琴, 方金云, 赵晓芳. 大规模时空数据分布式存储方法研究[J]. 高技术通讯, 2013, 23(12): 1219-1229

[6] Eldawy A. SpatialHadoop: towards flexible and scalable spatial processing using mapreduce[C]. In: Proceedings of the 2014 SIGMOD Phd Symposium, Snowbird, USA, 2014. 46-50

[7] 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, Dallas, USA, 2014. 73-82

[8] Puri S, Agarwal D, He X, et al. MapReduce algorithms for GIS polygonal overlay processing[C]. In: Proceedings of Parallel and Distributed Processing Symposium Workshops & Phd Forum, Cambridge, USA, 2013. 1009-1016

[9] Yu J, Wu J, Sarwat M. GeoSpark: a cluster computing framework for processing large-scale spatial data[C]. In: Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems, Seattle, USA, 2015. 1-4

[10] Tang M, Malluhi Q M, Malluhi Q M, et al. Location-Spark: a distributed in-memory data management system for big spatial data [J]. *Proceedings of the Vldb Endowment*, 2016, 9(13): 1565-1568

[11] 靳凤营, 张丰, 杜震洪,等. 基于 Spark 的土地利用矢量数据空间叠加分析方法[J]. 浙江大学学报(理学版), 2016, 43(1): 40-44

[12] 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): 327-333

[13] Guttman A. R-trees: a dynamic index structure for spatial searching [J]. *ACM Sigmod Record*, 1984, 14(2): 47-57

Research on vector polygon intersection algorithm in Spark framework

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

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

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

([***] Institute of Remote Sensing and Digital Earth, Chinese Academy of Sciences, Beijing 100094)

Abstract

A vector polygon intersection algorithm in Spark framework is proposed, which is a distributed memory computing framework. It solves the problems of current parallel polygon intersection algorithm, such as high cost of network data transmission and redundant computation. Firstly, a spatial grid partitioning strategy based on space filling curves and a grid filling method with polygon's minimum bounding rectangle (MBR) are introduced to reduce the network data transmission by transmitting MBR instead of polygon. Secondly, during the phase of data distribution, aiming at the scenario of complicated polygon which cross several grids, a polygon intersection location approach is adopted to eliminate the redundant computation of cross-district polygon. The results of the comprehensive experiments show that the proposed method can significantly improve the efficiency of the vector polygon intersection algorithm.

Key words: Spark, polygon intersection, minimum bounding rectangle (MBR), intersection location