

矢量多边形并行栅格化数据划分方法*

周琛, 李满春, 陈振杰, 姜朋辉, 陈东
(南京大学地理与海洋科学学院, 江苏南京 210023)

摘要:针对多边形并行栅格化中的负载不均衡问题提出一种新的数据划分方法, 主要包括: 迭代计算划分线的位置, 在每次迭代中保证分块间的计算量大致均衡, 完成数据划分、实现负载均衡; 提出基于二叉树的划分结果融合策略, 以解决跨边界多边形的融合问题。在多核 CPU 环境下实现并行算法, 选用多个典型土地利用现状数据集进行测试。结果表明: 针对不同类型多边形数据集, 所提方法较传统方法可获得更高的并行加速比和更好的负载均衡; 针对大数据量数据集, 以多边形节点数为度量标准可更精确地估算分块计算量, 从而更好地实现负载均衡。

关键词:地理信息系统; 并行计算; 多边形栅格化; 数据划分; 负载均衡

中图分类号: TP751 **文献标志码:** A **文章编号:** 1001-2486(2015)05-021-08

A novel data decomposition method for rapid parallel processing of vector polygon rasterization

ZHOU Chen, LI Manchun, CHEN Zhenjie, JIANG Penghui, CHEN Dong

(School of Geographic and Oceanographic Sciences, Nanjing University, Nanjing 210023, China)

Abstract: According to the load balance problem of large-scale parallel vector polygon rasterization, a novel data decomposition method was proposed. Firstly, the number of polygon nodes or the number of polygons was employed to evaluate the amount of calculations of a subset. The spatial locations of decomposed lines were computed iteratively and the balanced calculations between decomposed subsets were guaranteed, so as to realize data decomposition and load balancing. Secondly, a binary-tree based fusion strategy was put forth to merge the polygons across multiple subsets. The proposed parallel algorithm was implemented under a multi-core CPU-based environment and multiple China land use datasets were employed. Experimental results show that the presented method can outperform conventional methods for different datasets and can achieve a higher speed-up ratio and good load balancing. Moreover, when dealing with a large-scale vector dataset, the number of polygonal nodes is more appropriate to be the metric to evaluate the calculation of a subset precisely.

Key words: geographical information system; parallel computing; vector polygon rasterization; data decomposition; load balancing

矢量数据和栅格数据是地理信息系统 (Geographic Information System, GIS) 中的基本数据类型^[1]。栅格数据更适合进行空间分析和空间模拟, 能够高效地处理空间尺度问题, 因此经常需要进行矢量多边形数据的栅格化处理^[2]。近年来, 随着对地观测技术的快速发展, 利用并行计算技术实现对大规模多边形数据的快速、实时处理显得十分迫切和必要^[3-7]。在多边形并行栅格化中, 数据划分方法的优劣将极大地影响各划分分块计算量的均衡性, 进而影响并行计算效率^[8-9]。同时, 多边形具有数据量大、形态各异和复杂度差异大的特点^[10], 这对研究负载性良好的数据划分方法提出了挑战。

传统的数据划分方法包括基于多边形 ID 顺序和基于空间位置的划分方法。基于 ID 顺序的划分方法根据多边形 ID 的存储顺序均匀划分成多个分块^[11-13]; 该方法易于实现, 但划分较为粗略, 忽略了多边形复杂程度不同对并行效率的影响, 因而效率不高。基于空间位置的划分方法根据数据集的空间位置进行规则划分, 包括行划分、列划分、格网划分和四叉树划分等^[14-18]。在此基础上, Lee 等^[19]提出了一种启发式划分方法, 可将给定的空间范围划分成任意个面积相等的格网。该方法实现速度快, 且能在一定程度上保证多边形的空间聚集性, 因而应用广泛。然而, 该方法采用面积相等作为划分的标准, 忽略了多边形

* 收稿日期: 2015-06-16

基金项目: 国家 863 计划资助项目 (2011AA120301)

作者简介: 周琛 (1990—), 男, 江苏宿迁人, 博士研究生, E-mail: njzhouc@gmail.com;

李满春 (通信作者), 男, 教授, 博士, 博士生导师, E-mail: limanchun@yahoo.com

的大小、形状等特征对并行效率的影响,因而很难实现负载均衡。范俊甫等^[13]针对不同多边形图层叠置分析的并行处理提出了一种分组间关联最小化的划分方法,通过将所有相交多边形分组实现对多边形数据的划分。该方法划分规则明确,但仅适用于多边形叠置分析,不具有通用性,且极易造成不同分组间的计算量失衡,导致数据倾斜。为此,提出一种负载性较好的多边形数据划分方法。

本文提出一种改进的基于启发式划分的数据划分方法,主要包括:①基于传统启发式划分方法,将多边形节点数或多边形数作为划分的度量标准,通过迭代计算划分线的位置,进而完成数据划分;②提出一种基于二叉树的划分结果融合策略,将空间上相邻的分块依次进行两两融合,以解决跨边界多边形的融合问题。在多核 CPU 环境下实现并行栅格化算法,选用多个典型的中国土地利用现状数据集进行测试,并从运行时间、并行加速比和负载均衡三个方面对数据划分方法的有效性和稳定性进行评价。

1 算法并行性分析

典型的多边形栅格化算法包括内部点扩散法、复数积分法、扫描线算法和边界代数法等^[20-21]。内部点扩散法通过重复设定种子点,填充位于多边形内部及边界上的种子点栅格,直至多边形内部区域被填满;复数积分法对每个栅格单元逐个判定其是否包含在多边形之内,并将多边形内部的栅格单元进行填充;扫描线算法通过逐行扫描,识别多边形内部栅格像元条带,并用多边形的属性值将其填充。边界代数法通过加减代数运算将属性值赋给多边形内部及边界上的栅格单元。其中,边界代数法实现简便、运算速度快,本文主要实现边界代数法的并行栅格化。

多边形栅格化算法实现原理不同,但具有相同的并行性,表现为:主要过程为判定多边形内部及边界上的栅格单元;对单个多边形的处理都在最小外包矩形内,不涉及其他多边形;不依赖于具体的栅格化填充算法;对多边形节点数敏感性极强。上述分析表明,多边形栅格化属于数据密集型的局部计算类型,具有良好的可并行性。

2 改进的启发式数据划分方法

2.1 改进的启发式划分过程

本文基于传统的启发式划分方法,将分块计算量相等作为划分的标准,通过重复迭代计算划

分线位置使得划分后各分块计算量大致相当,从而实现负载均衡。在估算计算量时可采用多边形节点数或多边形数作为度量计算量的标准。改进后的启发式划分方法的基本过程如下(见图 1):

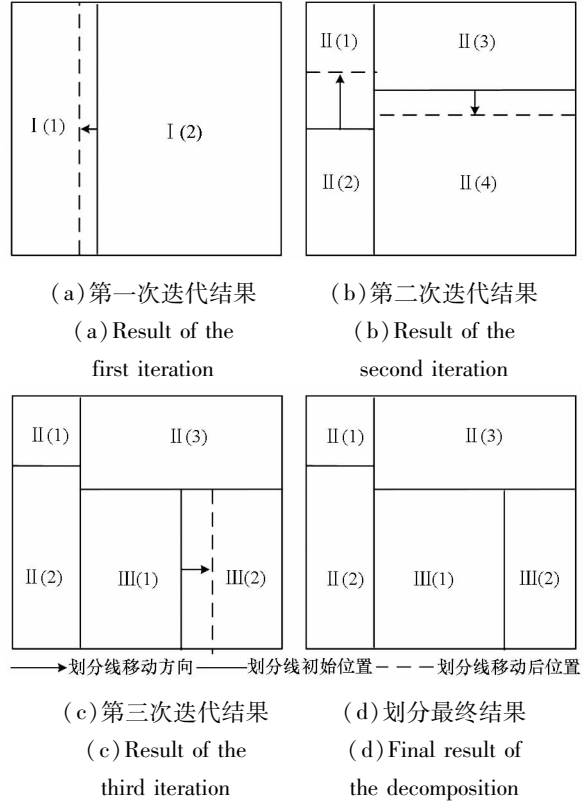


图 1 改进的启发式划分方法示意图

Fig. 1 Improved heuristic decomposition method

步骤 1:确定分块数 p 、最大迭代次数 n 、迭代中划分线每次移动的距离 d 和分块间节点数(或多边形数)相差阈值 s ;

步骤 2:若 $p = 1$ 则停止迭代计算;

步骤 3:若 p 为偶数,则将待划分区域沿着宽度较长的边划分成面积相等的分块 A 和 B ;通过空间查询分别获取 A 和 B 的节点数(或多边形数) N_A 和 N_B ,若 $|N_A - N_B| \leq s$,则满足要求,进行下一轮迭代。否则,需要对划分线位置进行调整,具体调整过程如下:①当 $N_A - N_B > s$ 时,将划分线的位置向使得分块 A 面积减少的方向移动距离 d ;②当 $N_B - N_A > s$ 时,将划分线的位置向使得分块 B 面积减少的方向移动距离 d ;③重复过程①、过程②直至满足 $|N_A - N_B| \leq s$ 或达到最大迭代次数 n ,在迭代过程中若当前划分线移动方向与上一次移动方向相反,则改变 d 的值,使得 $d = d/2$ 。

步骤 4:若 p 为奇数,则将待划分区域沿着宽度较长的边划分成分块 A 和 B ,使得两者面积比为 $[p/2]:[p/2] + 1$;通过空间查询分别获取 A 和 B 的节点数(或多边形数) N_A 和 N_B ,若

$|N_A - ([p/2]:[p/2] + 1)N_B| \leq s$, 则满足要求, 进行下一轮迭代。否则, 需要对划分线位置进行调整, 具体调整过程如下: ①当 $N_A - ([p/2]:[p/2] + 1)N_B > s$ 时, 将划分线的位置向使得分块 A 面积减少的方向移动距离 d ; ②当 $([p/2]:[p/2] + 1)N_B - N_A > s$ 时, 将划分线的位置向使得分块 B 面积减少的方向移动距离 d ; ③重复过程①、过程②直至满足 $|N_A - ([p/2]:[p/2] + 1)N_B| \leq s$ 或达到最大迭代次数 n , 在迭代过程中若当前划分线移动方向与上一次移动方向相反, 则改变 d 的值, 使得 $d = d/2$ 。

步骤 5: 对子分块 A 和 B 重复步骤 1 至步骤 4, 直至划分完毕。

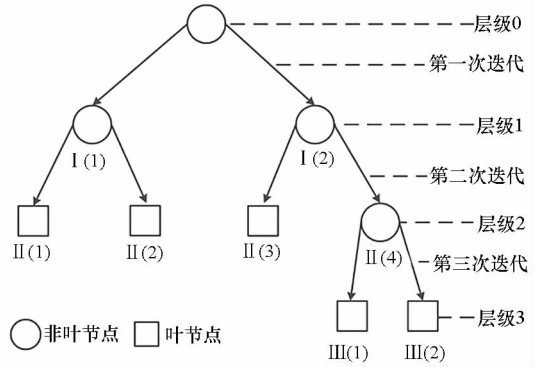
2.2 分块处理结果融合策略

在上述启发式划分后, 各任务分块的边界处存在大量跨边界多边形。若该类型多边形不经处理, 则会导致该类型多边形的重复处理, 从而降低并行效率。本文提出一种结果融合策略, 对并行处理后的跨边界多边形进行快速融合, 主要包括两个步骤: 基于二叉树结构的启发式划分结果构建及分块跨边界多边形的迭代处理。

首先, 根据启发式划分迭代划分空间位置的特性逐级构建二叉树, 每次划分当前空间形成的两个分块分别为当前层级的左右结点 (如图 2(a) 所示)。这样, 当指定分块数为 p 时, 形成的二叉树最大层级为 $\lceil \log_2 p \rceil$ 。当上述二叉树构建完毕后按照二叉树层级从底端逐层向上进行迭代计算, 迭代次数为 $\lceil \log_2 p \rceil$ 。在每次迭代中, 参与计算的分块为当前层级包含的分块 (如图 2(b) 所示)。主要过程如下: ①不属于该层级的分块直接进入下一次迭代; ②在当前二叉树层级中, 将拥有相同根结点的右结点分块中的跨边界多边形传递给处理左结点分块的进程, 并由该进程负责两个分块中跨边界多边形的融合; ③左结点进程剔除两分块中的相同多边形, 并将仅与该分块有交集的跨边界多边形写入目标文件, 将跨多个分块的多边形保留, 进入下一次迭代, 在下次迭代中, 该进程作为根结点的虚拟处理进程, 其处理的两个分块整体作为根结点的虚拟分块; ④重复步骤①~③, 直至迭代结束。该策略可保证每次参与融合的两个分块在空间上邻近, 且融合次数最少。

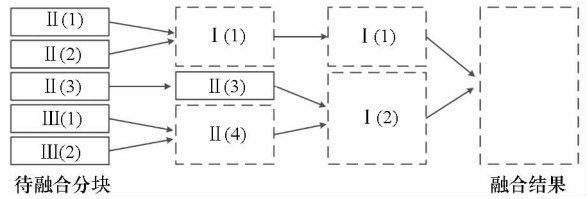
2.3 并行算法实现流程

多边形并行栅格化过程可分为预处理、并行执行和后处理过程。预处理过程包括多边形数据划分和任务分发; 并行执行过程, 即对各多边形进行并行栅格化填充计算; 后处理过程针对跨边界



(a) 基于二叉树结构的启发式划分结果构建

(a) Binary-tree based construction of the decomposition results



(b) 划分分块跨边界多边形的迭代处理

(b) Iterative processing of the polygons across multiple decomposed sub-datasets

图 2 划分结果融合策略示意图

Fig. 2 Fusion strategy for the decomposed results

多边形进行结果融合处理。并行算法采用标准 C++ 编程语言在 Linux 开发平台下开发, 并在消息传递接口 (Message Passing Interface, MPI) 并行环境下实现, 矢量数据的读写操作通过地理数据处理类库 (Geospatial Data Abstraction Library, GDAL) 实现。具体并行实现流程总体上分为以下步骤 (如图 3 所示):

步骤 1: 并行环境初始化, 接收数据划分参数, 包括计算量度量标准 (多边形节点数或多边形数)、分块数 (进程数) p 、最大迭代次数 n 、迭代中划分线移动距离 d 和分块间节点数 (或多边形数) 相差阈值 s 。

步骤 2: 读取源矢量多边形数据, 并获取最小范围内的空间位置。

步骤 3: 对多边形数据集进行改进的启发式划分, 完成数据划分, 并将划分结果传递给各并行进程。各并行进程分别处理 1 个任务分块。

步骤 4: 各并行进程读取各自任务分块中的多边形, 并调用多边形栅格化算法进行并行计算。

步骤 5: 并行执行过程结束后构建基于二叉树的启发式划分结果, 并根据二叉树结构进行迭代计算, 将空间上邻近的分块两两进行融合, 以解决跨边界多边形的融合问题。

步骤 6: 输出最终结果并退出并行环境。

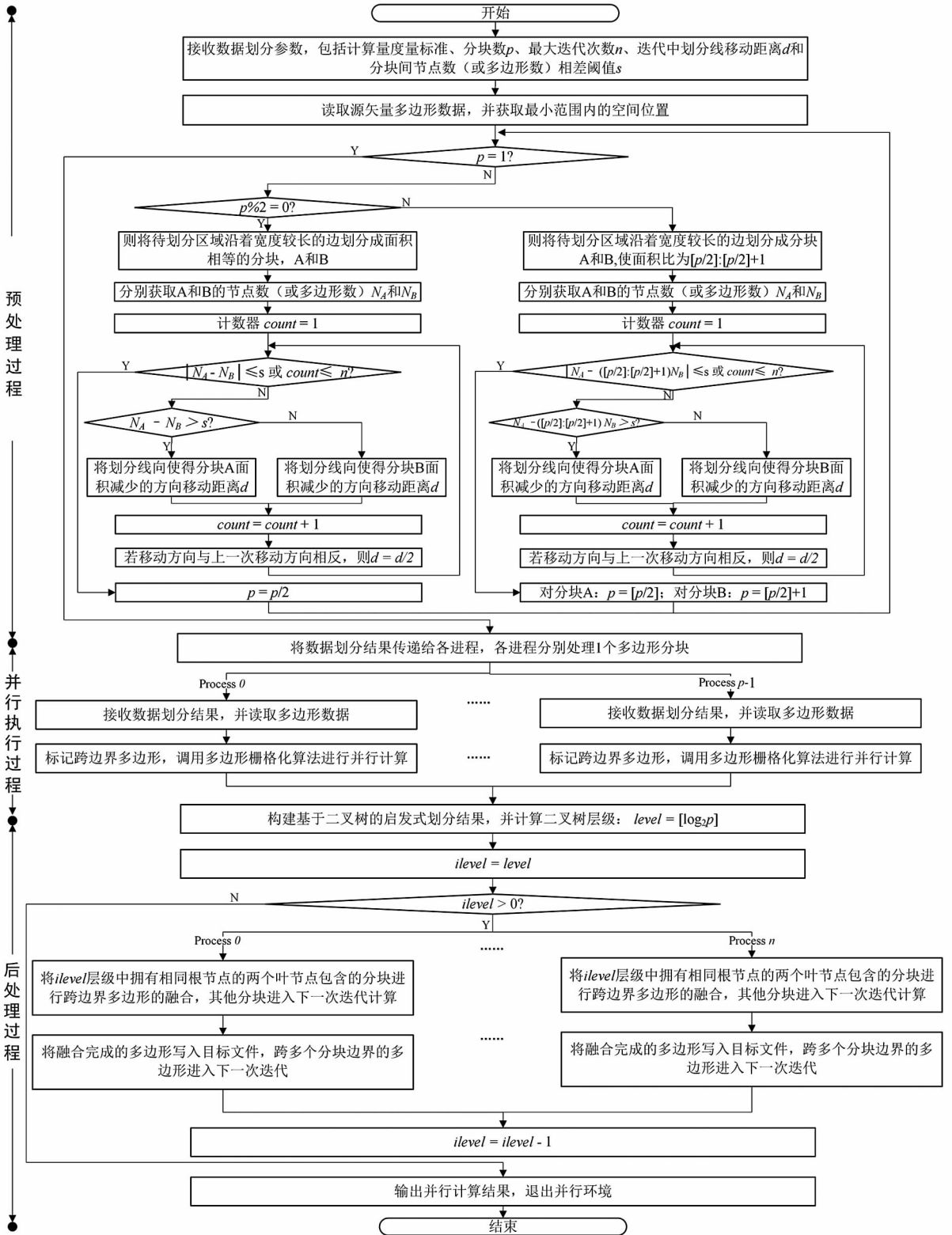


图 3 并行算法实现流程图

Fig. 3 Implementation flow of the parallel polygon calculation algorithm

3 实验结果与分析

3.1 性能评价方法

本文将多边形数据集分为三类:不同数据量、不同空间分布和不同数据复杂度。其中,数据量

用数据集的内存占用量和多边形数表示;空间分布用数据集实际面积与其最小外接矩形面积的比值表示;数据复杂度用平均多边形节点数表示。本文分别采用运行时间、加速比和负载均衡指数来评价算法的并行性能。其中,运行时间是从算

法启动直到最后一个进程执行完所花费的时间。加速比是同一个任务在串行环境下和并行环境下运行时间的比值,如式(1)所示。

$$Speedup = \frac{T_{\text{sequential}}}{T_{\text{parallel}}} \quad (1)$$

其中, $T_{\text{sequential}}$ 为串行时间, T_{parallel} 为并行时间。负载均衡指数等于并行进程执行的最长时间与最短时间的比值,该指数越接近于1,表明进程间运行时间越接近、算法负载性能越好,如式(2)所示。

$$Loadbalancing = \frac{\max_{i=1 \sim p} \{T_i\}}{\min_{i=1 \sim p} \{T_i\}} \quad (2)$$

其中, p 为进程数, $\max_{i=1 \sim p} \{T_i\}$ 为并行进程执行的最长时间, $\min_{i=1 \sim p} \{T_i\}$ 为最短时间。

本文将所提数据划分方法与传统方法进行对比,分别测试三类多边形数据集,并从并行运行时间、加速比和负载均衡性能三个方面对算法并行性能进行评价。







3.2 并行环境与实验数据

程序运行选择 IBM 并行集群,包含5个计算节点,每个节点的硬件配置为:CPU 2 颗,规格为 Intel (R) Xeon (R) CPU E5 - 2620 (主频 2.00GHz,6核12线程);内存为16GB(4根4GB内存条,规格为 DDR3 RDIMM1600MHz);硬盘为2TB,网络为集成的双口千兆以太网。软件配置:操作系统为 Centos Linux 6.3,文件系统为 lustre 系统,MPI 的实现产品选择 OpenMPI 1.4.1。

测试数据为中国不同区域的土地利用现状数据(见表1)。其中,数据1和2代表不同数据量的数据集,其数据量分别为5.5 GB和1.6 GB,多边形数分别为12 126 100和2 300 723;数据3和4代表不同空间分布的数据集,实际面积与其最小外接矩形(Minimum Bounding Rectangle, MBR)面积比值分别为54.9%和27.7%;数据5和数据6代表不同复杂度的数据集,其数据量相近、但平均节点数相差很大,分别为35.05和643.15。

表1 测试数据集基本参数

Tab. 1 Datasets used in the experiments

数据集名称	不同数据量		不同空间分布		不同数据复杂度	
	数据1	数据2	数据3	数据4	数据5	数据6
示意图						
数据量	5.5 GB	1.6 GB	1.0 GB	1.2 GB	589 MB	537 MB
多边形数	12 126 100	2 300 723	702 199	896 348	541 562	53 646
面积/km ²	100 320	29 143	11 702	3 730 436	4323	21 528
(实际面积/ MBR 面积)/%	46.1%	56.5%	54.9%	27.7%	48.5%	62.3%
平均节点数	21.96	33.32	48.95	30.58	35.05	643.15

3.3 不同数据划分方法性能对比

实验的目的为比较不同划分方法的性能,将本文方法与传统的基于多边形 ID 顺序的划分方法和启发式划分方法分别应用于并行算法中。在本文方法中,采用多边形节点数作为度量计算量的标准;测试从数据1至数据6,分别计算运行时间、加速比和负载均衡指数(见图4)。

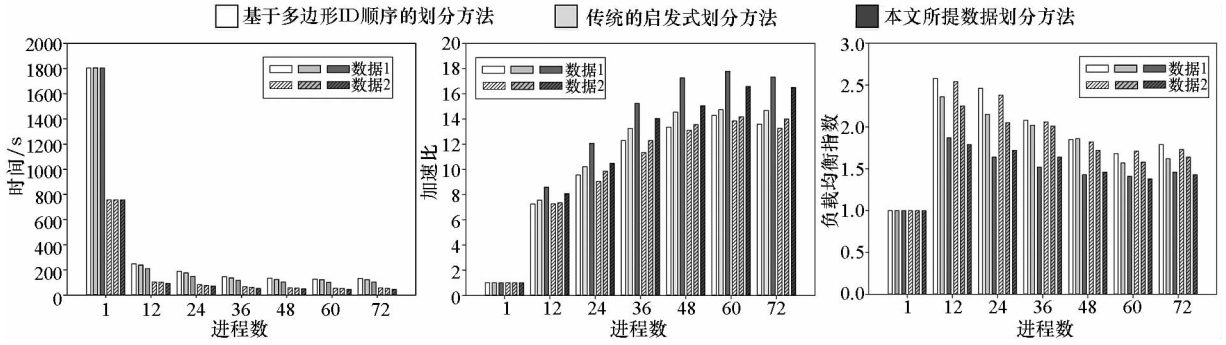
图4(a)描述了对不同数据量数据集的测试结果。不同数据划分方法表现出相同的变化趋势:运行时间随着进程数的增加逐渐减少;加速比逐渐上升,当达到并行环境的最大核数时达到最

优;负载均衡指数逐渐降低。这表明上述三种方法均能降低算法的运行时间;比较而言,本文方法能更有效地降低算法运行时间、获得更高的并行加速比。对数据1,串行时间为1805.34 s,三种方法的最少运行时间分别为126.34 s,122.40 s和101.48 s,最大加速比为14.29,14.75和17.79;对数据2,串行运行时间为756.89 s,最少运行时间分别为54.65 s,53.38 s和45.62 s,最大加速比为13.85,14.18和16.59。同时,对不同数据量的数据集,传统方法负载均衡指数较大,这表明各进程间计算量均衡性较差;而本文方法负载均衡

指数低于其他两种方法,表明采用本文方法的并行算法各进程间计算量较为均衡。

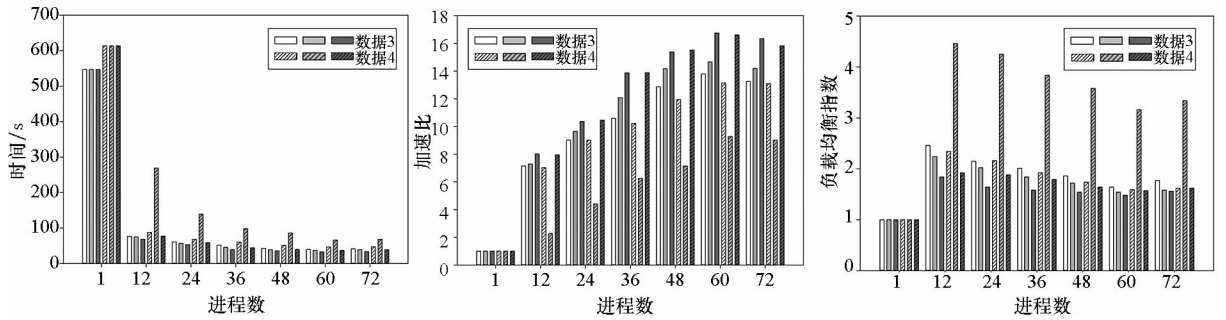
图 4(b)描述了对不同空间分布数据集的测试结果。对于空间分布较为均匀的数据 3,不同方法均表现出较好的性能。对于空间分布不均的数据 4,传统的启发式划分方法受空间分布影响

较大,仅获得 9.28 的加速比;同时,其负载均衡指数较高,高于 3.16。原因在于传统启发式划分方法仅能保证各分块面积相等,而不能保证各分块计算量相等,从而不适用于空间分布不均的数据集。而本文方法基本不受空间分布的影响,对不同数据集均能表现出稳定的加速效果。



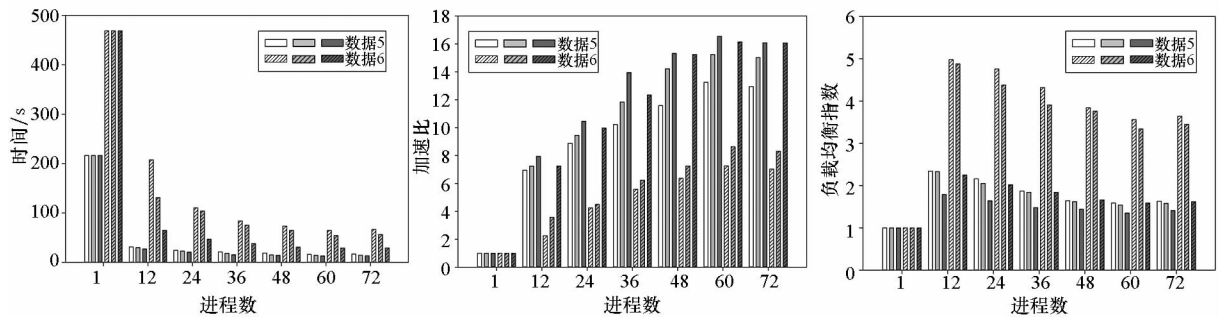
(a) 不同方法执行不同数据量数据集的计算结果

(a) Experimental results of different methods when performing on datasets with different data volumes



(b) 不同方法执行不同空间分布数据集的计算结果

(b) Experimental results of different methods when performing on datasets with different spatial distribution



(c) 不同方法执行不同复杂度数据集的计算结果

(c) Experimental results of different methods when performing on datasets with different complexity

图 4 不同数据划分法性能对比结果图

Fig. 4 Experimental results of different methods on execution time, speedup ratio and load balancing

图 4(c)描述了对不同数据复杂度数据集的测试结果。对于复杂度较低的数据 5,不同方法表现出较好的性能。对于复杂度较高的数据 6,传统的两种方法均不能适用,具体表现在:对于相同的串行时间 468.95 s,传统方法的最少运行时间分别为 64.59 s 和 54.27 s,最大加速比为 7.26

和 8.64;两者的负载均衡指数均较高,最小值为 3.34。主要原因在于数据 6 多边形节点数相差巨大,存在多个包含大量节点数的大多边形,上述两种划分方法均无法有效分配大多边形,导致数据严重倾斜,从而产生并行处理过程中的负载不均衡问题。比较而言,本文方法可以较好地处理复

杂度较高的数据集,获得良好的加速比(16.15)。

总结来说,本文提出的数据划分方法较传统方法能更稳定、更有效地处理不同类型的矢量多边形数据集,获得更高的并行加速比、更好的负载均衡性能。

3.4 不同度量标准对运行时间的影响

本文提出的数据划分方法中可采用节点数或多边形数作为度量分块计算量的标准。并行算法采用的度量标准不同,其并行执行效率也不同。实验将并行算法执行总时间分为预处理、并行执行、后处理和 I/O 时间;分别采用多边形节点数和多边形数为度量标准,执行并行多边形栅格化算法计算数据 1 和数据 2,并统计其运行时间(见表 2)。

在总时间上,以多边形节点数为度量标准的并行算法运行时间少于以多边形数为度量标准的并行算法,其中以大数据量的数据 1 更为明显:两者执行数据 1 的最少时间分别为 104.54s 和

124.38s;执行数据 2 的最少时间分别为 50.26s 和 49.49s。这表明针对大数据量的矢量多边形数据集,以多边形节点数为度量标准比以多边形数为度量标准的并行算法可获得更好的加速效果。

在各部分执行时间上,两者的后处理时间和 I/O 时间相差不大,而在预处理时间和并行执行时间上相差较大,表现在:以多边形节点数为度量标准的并行算法预处理时间高于以多边形数为标准的算法,而其并行执行时间明显较少。原因在于:以多边形节点数为度量标准的并行算法在数据划分中需要重复统计分块中的多边形节点数,因而耗时长;但其时间收益大于以多边形数为标准的并行算法,主要表现在其并行执行时间明显降低较快。这表明针对大数据量的多边形数据集,以多边形节点数为标准可更精确地度量计算量,从而使得各划分分块计算量更加均衡,获得更高的并行执行效率。

表 2 不同度量标准对并行效率影响的测试结果
Tab. 2 Results of the influence of different metrics on parallel efficiency

数据 1 测试结果										
进程数	以多边形节点数为度量标准					以多边形数为度量标准				
	预处理	并行执行	后处理	I/O	总时间	预处理	并行执行	后处理	I/O	总时间
1	0.00	1783.76	0.00	21.58	1805.34	0.00	1784.12	0.00	21.22	1805.34
12	14.88	169.14	2.68	23.47	210.17	7.79	196.34	2.66	23.65	230.44
24	16.97	104.54	3.75	24.19	149.45	9.28	134.08	3.85	24.04	171.25
36	18.16	69.06	5.09	26.07	118.38	11.37	96.08	5.69	26.64	139.78
48	21.87	49.28	6.24	27.15	104.54	14.74	75.19	6.44	28.01	124.38
60	28.54	36.42	7.88	28.64	101.48	16.34	59.97	8.02	28.81	113.14

数据 2 测试结果										
进程数	以多边形节点数为度量标准					以多边形数为度量标准				
	预处理	并行执行	后处理	I/O	总时间	预处理	并行执行	后处理	I/O	总时间
1	0.00	746.30	0.00	10.59	756.89	0.00	746.30	0.00	10.47	756.77
12	7.95	72.64	1.36	11.72	93.67	3.32	88.64	1.42	11.66	105.04
24	8.82	48.33	2.24	12.76	72.15	5.58	56.57	2.09	12.81	77.05
36	10.26	25.50	4.52	13.59	53.87	6.49	38.87	4.33	13.62	63.31
48	12.28	18.37	4.89	14.72	50.26	7.27	22.87	4.92	14.43	49.49
60	16.68	8.84	5.24	14.86	45.62	8.51	18.68	5.18	14.91	47.28

4 结论

提出一种新的数据划分方法,主要包括:将多

边形节点数或多边形数作为度量计算量的标准;迭代计算划分线的位置,在每次迭代中保证分块间的计算量大致均衡,完成数据划分;提出了基于

二叉树的划分结果融合策略,解决了跨边界多边形的快速融合问题。在多核 CPU 环境下实现并行算法;选用多个土地利用现状数据集进行测试。结果表明:本文提出的数据划分方法较传统方法针对不同类型多边形数据集可获得更高的并行效率、更好的负载均衡性能;针对大数据量的多边形数据集,以多边形节点数为度量标准可更精确地估算分块计算量,从而更好地实现负载均衡。

参考文献 (References)

- [1] Goodchild M F. Scale in GIS: an overview[J]. *Geomorphology*, 2011, 130(1-2): 5-9.
- [2] 吴立新, 史文中. 地理信息系统原理与算法[M]. 北京: 科学出版社, 2003.
WU Lixin, SHI Wenzhong. *Geographic information systems principles and algorithms*[M]. Beijing: Science Press, 2003. (in Chinese)
- [3] Mineter M J. A software framework to create vector-topology in parallel GIS operations[J]. *International Journal of Geographical Information Science*, 2003, 17(3): 203-222.
- [4] 刘军志, 朱阿兴, 刘永波, 等. 基于栅格分层的逐栅格汇流算法并行化研究[J]. 国防科技大学学报, 2013, 35(1): 123-129.
LIU Junzhi, ZHU Axing, LIU Yongbo, et al. Parallelization of a grid-to-grid routing algorithm based on grids layering [J]. *Journal of National University of Defense Technology*, 2013, 35(1): 123-129. (in Chinese)
- [5] Yang C W, Goodchild M F, Huang Q Y, et al. Spatial cloud computing: how can the geospatial sciences use and help shape cloud computing? [J]. *International Journal of Digital Earth*, 2011, 4(4): 305-329.
- [6] 吴立新, 杨宜舟, 秦承志, 等. 面向新型硬件构架的新一代 GIS 基础并行算法研究 [J]. 地理与地理信息科学, 2013, 29(4): 1-8.
WU Lixin, YANG Yizhou, QIN Chengzhi, et al. On basic geographic parallel algorithms of new generation GIS for new hardware architectures [J]. *Geography and Geo-Information Science*, 2013, 29(4): 1-8. (in Chinese)
- [7] 程果, 景宁, 陈萃, 等. 栅格数据处理中邻域型算法的并行优化方法 [J]. 国防科技大学学报, 2012, 34(4): 114-119.
CHENG Guo, JING Ning, CHEN Luo, et al. Parallel optimization methods for raster data processing algorithms of neighborhood-scope [J]. *Journal of National University of Defense Technology*, 2012, 34(4): 114-119. (in Chinese)
- [8] Shekhar S, Ravada S, Chubb D, et al. Declustering and load-balancing methods for parallelizing geographic information systems [J]. *IEEE Transactions on Knowledge and Data Engineering*, 1998, 10: 632-655.
- [9] Ferhatosmanoglu H, Tosun A S, Canahuate G, et al. Efficient parallel processing of range queries through replicated declustering[J]. *Distributed and Parallel Databases*, 2006, 20(2): 117-147.
- [10] Meng L K, Huang C Q, Zhao C Y, et al. An improved hilbert curve for parallel spatial data partitioning [J]. *Geospatial Information Science*, 2007, 10(4): 282-286.
- [11] Hawick K A, Coddington P D, James H A. Distributed frameworks and parallel algorithms for processing large-scale geographic data [J]. *Parallel Computing*, 2003, 29(10): 1297-1333.
- [12] Ye J Y, Chen B, Chen J, et al. A spatial data partition algorithm based on statistical cluster[C]//Proceedings of the 19th International Conference on Geoinformatics, NY: IEEE, 2011: 1-6.
- [13] 范俊甫, 马廷, 季民, 等. GIS 中 8 种图层级多核并行多边形叠置分析工具的实现及优化方法 [J]. 地理科学进展, 2013, 32(12): 1835-1844.
FAN Junfu, MA Ting, JI Min, et al. Implementation and optimization of eight parallel polygon overlapping tools with OpenMP at the feature layer level in GIS [J]. *Progress in Geography*, 2013, 32(12): 1835-1844. (in Chinese)
- [14] Wang S W, Cowles M K, Armstrong M P. Grid computing of spatial statistics: using the TeraGrid for $G_i^*(d)$ analysis [J]. *Concurrency and Computation: Practice and Experience*, 2008, 20(14): 1697-1720.
- [15] Armstrong M P, Pavlik C E, Marciano R. Experiments in the measurement of spatial association using a parallel supercomputer [J]. *Geographical Systems*, 1994, 1(4): 267-288.
- [16] Mineter M J, Dowers S. Parallel processing for geographical applications: a layered approach [J]. *Journal of Geographical Systems*, 1999, 1(1): 61-74.
- [17] Agarwal D, Puri S, He X, et al. A system for GIS polygonal overlay computation on Linux cluster—an experience and performance report [C]//Proceedings of the 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum, NY: IEEE, 2012: 1433-1439.
- [18] Wang Y F, Chen Z J, Cheng L, et al. Parallel scanline algorithm for rapid rasterization of vector geographic data [J]. *Computers & Geosciences*, 2013, 59: 31-40.
- [19] Lee C K, Hamdi M. Parallel image processing applications on a network of workstations [J]. *Parallel Computing*, 1995, 21(1): 137-160.
- [20] Liao S B, Bai Y. A new grid-cell-based method for error evaluation of vector-to-raster conversion [J]. *Computational Geosciences*, 2010, 14(4): 539-549.
- [21] Zhou C H, Ou Y, Yang L, et al. An equal area conversion model for rasterization of vector polygons [J]. *Science in China Series D: Earth Science*, 50(1): 169-175.