applied sciences

MDPI

Article

An Efficient Parallel Algorithm for Polygons Overlay Analysis

Yuke Zhou [1], Shaohua Wang [2,*] and Yong Guan [3]

1 Key Laboratory of Ecosystem Network Observation and Modeling, Institute of Geographic Sciences and Natural Resources Research, Chinese Academy of Sciences, Beijing 100101, China; zhouyk@igsnrr.ac.cn
2 Department of Geography, University of California, Santa Barbara, CA 93117, USA
3 Department of Information System and Technology, Claremont Graduate University, Claremont, CA 91711, USA; yong.guan@cgu.edu
* Correspondence: shaohua@geog.ucsb.edu; Tel.: +1-805-284-3507

Received: 29 September 2019; Accepted: 11 November 2019; Published: 13 November 2019

Abstract: Map overlay analysis is essential for geospatial analytics. Large scale spatial data pressing poses challenges for geospatial map overlay analytics. In this study, we propose an efficient parallel algorithm for polygons overlay analysis, including active-slave spatial index decomposition for intersection, multi-strategy Hilbert ordering decomposition, and parallel spatial union algorithm. Multi-strategy based spatial data decomposition mechanism is implemented, including parallel spatial data index, the Hilbert space-filling curve sort, and decomposition. The results of the experiments showed that the parallel algorithm for polygons overlay analysis achieves high efficiency.

Keywords: parallel algorithm; map overlay analysis; Hilbert ordering decomposition; spatial analysis

1. Introduction

Map overlay analysis is a fundamental operator in geospatial analytics and widely used in geospatial applications [1,2]. Large scale spatial data pressing poses challenges for geospatial map overlay analytics [3].

The parallel GIS algorithm is an efficient way to conduct map overlay analysis [4–7]. Spatial data decomposition is the basis of parallel computing architecture based on the spatial data partitioning mechanism [8]. Spatial data domain decomposition in parallel GIS refers to the decomposition of object sets in the study area according to a certain granularity and is assigned to different computing units for processing to achieve high concurrency. Spatial data domain decomposition from the perspective of geographic data storage mainly refers to the database domain to allocate spatially adjacent geographical elements to the same physical medium storage according to a certain decomposition principle. The feature elements form different groups in space in the form of clusters, and the spatially separated clusters are divided into different storage areas to realize parallelized spatial data extraction mode. The parallelized map overlay analysis algorithm technology route is based on data division and behavior.

Parallel spatial data decomposition needs to take into consideration the data storage and geo-computation in each child node from the perspective of the spatial distribution of feature objects, while spatial data has multidimensionality [9]. In the process of parallel overlay analysis, the core of parallelization is a fast intersection judgment of geometric objects and the interactive communication between geospatial data [10,11]. Therefore, the critical principle of layer element decomposition is to maintain the spatial proximity of data. The main feature of geographic data is that it has a strong spatial correlation, and its data parallel strategy should be compatible with spatial data types. The spatial feature is the difference between ordinary numerical parallel computing and the key technology of parallel GIS system [12,13]. The purpose of spatial data decomposition is to

implement the local process of spatial analysis operations (to reduce the synchronization operation between computing nodes). The modeling of spatial elements can be accelerated by computational localization. With the improvement of computer hardware performance and the increasing cost of storage, the usual strategy is to exchange storage space for computing time [12,14].

In this paper, we propose an efficient parallel algorithm for polygons overlay analysis. We decompose vector space data based on space filling-curve (Hilbert curve) to better maintain the spatial proximity of data decomposition, which is conducive to the parallelization of spatial proximity and sensitive overlay operations, and with the basis of the original Hilbert ordering decomposition, the spatial data is decomposed using different sorting strategies. This paper is organized as follows. Section 2 introduces related work. Section 3 shows the methods of a parallel algorithm for polygons overlay analysis. The experimental results are given in Section 4. Following this, the last section contains the conclusion and further work.

2. Related Work

The most time-consuming operation of polygon overlay analysis continues to be the intersection of line segments. Ching studied the load balancing problem for the parallel map overlay calculation based on the line segment set [15] but did not deeply discuss the communication and merge-efficiency of each node after parallel calculation, and cannot guarantee the constant acceleration ratio of the whole process.

Parallel spatial data region decomposition needs to take into consideration the storage and calculation in each child node from the perspective of the spatial distribution of feature objects while spatial data has multidimensionality. In traditional data partitioning methods, such as token rotation, hash table segmentation, and simple region partitioning, the spatial relationship between objects is split during decomposition, which does not reflect the proximity between spatial data.

The purpose of using Hilbert space sorting is to maximize the mapping of high-dimensional data to low-dimensional data [16,17] and to close the geographically adjacent points in computer storage to accelerate data extraction and improve the efficiency of data operations in the first-level storage. The access to spatial data in memory is performed randomly. For spatial data with unbalanced distribution, if the point data comparison is too dense in a certain area, data redundancy in the index sub-node is caused. In order to maintain the uniqueness of spatial data mapping, a more detailed division of the index grid is required. However, if the division is too detailed, it will increase the difficulty and computational complexity of spatial sorting coding and also increase the size of the spatial query.

The spatial data decomposition can be divided by the spatial indexing mechanism. Based on the spatial index, the search space of the candidate dataset in the overlay analysis can be effectively reduced. At the same time, the false intersection can be further filtered in the proximity analysis of the candidate geometry data into a map overlay object with real intersections. The data decomposition in the parallel superposition analysis method is based on the vector topology data model. The key difficulty is how to assign the elements with large spatial proximity to the same node. The vector data capacity is usually between megabytes to gigabytes, and the current computer hard disk is measured by the terabyte level storage capacity. The equalization of storage capacity is not a critical issue. The key problem is also how to effectively equalize the computational tasks of vector data and reduce unnecessary intersection detection and parameter communication between distributed nodes. Because the input spatial data will have different feature density, the division of spatial data into distributed nodes in the GIS parallel algorithm does not have a conventional experience.

Most spatial decomposition methods are based on planar space, where one point on one side of the plane defines one area while the other side determines another area. However, as points on the plane can be arbitrarily divided into one area, using the plane to divide the space recursively will eventually generate a Binary Space Partition Tree (BSP Tree) [18]. Using spatial data decomposition structures to store objects can easily speed up specific geometric query operations, such as conflict detection, to determine if two objects are close together or if a ray intersects the object. The quadtree index belongs

to the vertical decomposition mode of the plane [19]. The generation process is to recursively divide the geospatial by four points until the termination condition is set by itself (for example, each area contains no more than two points, if it exceeds four points). Finally, a hierarchical quadtree is formed. Quadtree decomposition is a typical planar recursive decomposition [20]. This method is suitable for the case where the data isomorphism and distribution are relatively balanced [21].

The decomposition of the spatial index is determined by geometric objects (object priority). Decomposing spatial regions into sub-regions according to spatial objects are also called bucket. Therefore, this method is usually called bucket partitioning. The object-oriented data decomposition method needs to follow certain principles. The most classic strategy is that the B-tree rule uses a separate point or line to decompose the spatial extent recursively [22]. Another classic decomposition principle is to keep the outer bounding rectangle of the object to be the smallest, and R-tree is its important implementation [23]. Use the R-tree spatial data domain to decompose the main packing or bucket mode of the R-tree. R-tree optimization is mainly for the packing and sorting of sub-node index packets. Kamel and Faloutsos proposed the MBR sorting method based on the Hilbert curve [24]. Roussopoulos proposed a sorting method based on Nearest-X in a certain direction. In the classic R-tree implementation, Guttman (1984) proposed two heuristic tree node decomposition methods, including Quadratic-Split and Linear-Split [23]. The performance of the R-tree depends on the quality of the data outsourcing the rectangular clustering algorithm in the node. The Hilbert R-tree uses the space filling curve, especially the Hilbert curve, to organize the data outsourcing the rectangle linearly. The Hilbert R-tree comes in two forms: one is a static database form, and the other is a dynamic database form. In the paper research scheme, the Hilbert fill curve is used to achieve better ordering of high-dimensional data in nodes. This order ensures that similar objects outsourcing rectangles are grouped into groups, keeping the area and perimeter of the resulting outer rectangle as small as possible, so the Hilbert curve is a good sorting method in the sense of this layer. The compact Hilbert R-tree is suitable for static databases [25], and there are few update operations in the static database or no update operations at all. The dynamic Hilbert R-tree is suitable for dynamic databases [26–28], which require real-time insert, delete, and update operations. The dynamic Hilbert R-tree uses an elastic segmentation delay mechanism to increase space utilization. Each node in the tree has a well-defined set of sibling nodes. By establishing the order of the nodes in the R-tree and adjusting the Hilbert R-tree data partitioning strategy, the ideal space utilization degree can be achieved. The Hilbert R-tree is ordered based on the Hilbert value of the center point of the object rectangle, and the point Hilbert value is the length from the start of the Hilbert curve to the point. In contrast, other R-tree variants do not have control over space utilization. Leutenger proposed a new R-tree variant, the Sort-Tile-Recursive tree (STR-tree). The algorithm uses the recursive idea. For the set of spatial rectangles with r in the k-dimensional plane, let the maximum capacity of the leaves of the R-tree be n, and the rectangles are sorted according to the x value of the center point. The concept of the tile is to use $\sqrt{(r/n)}$ vertical cuttings. The line divides the sorted rectangle so that each strip can be loaded close to $\sqrt{(r/n)}$ nodes. Each slice continues to be sorted according to the y value of the center point of the rectangle, and a leaf node is pressed every n rectangle; the top-down reclusiveness processes the slice to generate the entire R-tree. One of the measures of efficiency and accuracy of the R-tree index is the area and perimeter of the sub-node MBR in the tree. The smaller the area and perimeter, the higher the spatial aggregation. Therefore, the analysis of the R-tree proposed by Guttman (1984) has some shortcomings: long loading time, insufficient subspace optimization, and long data extraction time for the spatial query.

No matter the equilibrium grid decomposition, quadtree decomposition, and traditional R-tree decomposition, the problem of large spatial distribution and density imbalance cannot be avoided. The regularized partitions of these decomposition methods are divided into different degrees.

In the algorithm of parallel overlay analysis, a lot of frequent data extraction from the cluster environment and the intersection of geometric objects are involved [6]. In the single-disk and single-processor environment, the traditional spatial data extraction method uses the index structure of the spatial database. However, the single-point spatial index storage and access mechanism in

multi-disk and multi-processor environments cannot meet the requirements of high-performance computing for data extraction speed [29]. Therefore, it is necessary to implement a fast filtering and extraction mechanism for spatial data matching with a distributed shared-nothing mode in the computing environment of parallel overlay analysis. Spatial index is an important criterion for measuring the quality of the spatial database. In the spatial database, there are usually millions of data tables. If the traditional spatial indexing method of the database is adopted, the data query efficiency will be seriously reduced. At the same time, spatial data has spatial object uncertainty, and the intersection, inclusion, and separation are complex. It is difficult to classify and sort spatial data by processing ordinary data. The spatial index of the spatial database field can be divided into two types: embedded and external [30]. The embedded spatial index structure is incorporated into the database as part of the database itself, while the plug-in database is usually in the form of middleware, which performs a similar proxy and forwarding mechanism at the data request and data layers. For example, the default indexing methods for Oracle spatial and PostGIS are R-tree and B-tree [31,32]; ESRI ArcSDE is an external spatial database management mechanism, which has no specific spatial data carrier but is based on the traditional RDBMS system. The extension, such as ArcSDE, can implement spatial data management and indexing mechanisms based on SqlServer and Oracle. The spatial database established by ArcSDE is called Geodatabase [33]. The default indexing strategy for geodatabase is a spatial grid index for the feature class. Secondly, in the aspect of distributed spatial index data decomposition, Kamel (1993) and other research applied R-tree to the deployment of single-processor and multi-disk hardware structures and implemented an R-tree-based parallel query mechanism. Zhang et al. used a multivariate R-tree (Multiplexed-R-tree) structure to optimize R-tree in combination with proximity index constraints [34]. Experiments show that parallel domain query performance is better when dealing with spatially balanced data. In order to improve the efficiency of massive spatial data management and parallelization processing in the distributed parallel computing environment, Tanin et al. implemented the distributed Quadtree-based spatial query function in the peer–to peer network environment [35].

The parallel spatial index has gradually formed an essential branch of the spatial index family with the development of high-performance parallel GIS applications, which can solve the problem of the simple data decomposition method in this study. The most typical parallel spatial index is the MC-R-tree (Master-client R-tree) method proposed by Schnitzer [36]. The method is characterized in that all non-leaf nodes in the spatial index tree are stored in the main cluster. In the node, each subtree in the index tree is stored in the sub-computing node of the cluster. The disadvantage is that the space utilization of the conventional R-tree is low, and the number of MBR overlaps higher when the subtree index are assigned to the child nodes.

3. Methods

This section includes active-slave spatial index decomposition for intersection, multi-strategy Hilbert ordering decomposition, and the parallel spatial union algorithm.

3.1. Active-Slave Spatial Index Decomposition for Intersection

From the dynamic nature of the overlay analysis, the superimposed layers are divided into the active layer (overlay layer) and passive layer (base layer). The point of parallel acceleration is the fast query of the geometric elements in the active layer. The intersection part of the layer and the spatial index is the key technology to achieve acceleration. According to the characteristics of the parallel overlay analysis in this paper, the storage of spatial data adopts a completely redundant mechanism, and each child node maintains a complete set of spatial data tables to be superimposed.

A data decomposition method is proposed for the parallel intersection operation. The process is to spatially decompose the active layer in advance and send the partition location information to the corresponding child node according to the FID (The name primarily used in the spatial data layer), the child node then locally extracts the active layer data in the range, then, the geographic elements in the

passive layer all participate in the establishment of the whole spatial index tree; the elements in the active layer query the passive layer index and perform superposition operations on the intersecting candidate sets. Since the process of spatial query and the intersection operation in the overlay analysis are mostly the same operation, the data decomposition mode is adapted to be combined with the parallel intersection operation. In Figure 1, O_obj represents the entity object in the active overlay layer O_lyr. After the decomposition strategy, it is distributed to each child node, and the base layer B_lyr is parallelized.

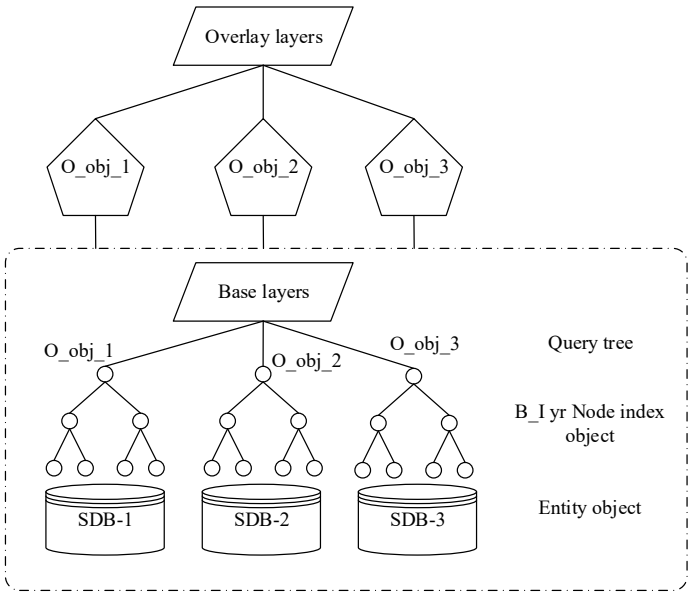

Figure 1. Parallel data decomposition using R-tree.

The actual data decomposition is only the division and treatment of the overlay layer O_lyr, and the base layer is still a query as a whole. This solution enables acceleration in a reasonable computational complexity in small to medium data scale applications. The data decomposition characteristics of the above map overlay algorithm shows that if each child node resides in a base layer, the spatial index of the entire tree has two defects: first, when processing a layer with a large number of geographic features, each node needs to consume a lot of time and memory resources; second, each query is for the traversal of the entire tree, and there are a large number of invalid queries in the intersecting query of a single geometric object.

To improve the efficiency of massive spatial data management and processing in a parallel environment, an improved master-slave spatial indexing method MC-STR-tree was designed according to the current research results. The rapid management of data management and geometric intersections provides conditions.

The basic index structure of the index adopts STR-tree, and the modification of the R-tree object node is improved, and the conflict of data distribution among the nodes is reduced. It has good spatial clustering characteristics and extremely low spatial index packet overlap rate, so it has excellent parallelization characteristics. STR-tree is a highly balanced tree with simple and maximized space utilization.

The processing steps of the improved master-slave parallel space index MC-STR-tree decomposition (Figure 2) are as follows:

(1) Data preprocessing

The master node divides the MBR of the spatial data index node into n parts according to specific rules. The commonly used rules are Hilbert ordering and STR partitioning method, and n is an integer

not larger than the cluster computing node. The master node needs to record the node tag to which each subtree is sent and the root node MBR of the subtree. In this paper, the leaf nodes are divided according to the STR rule, and each computing node corresponds to a stripe in the STR. To control the number of subtrees less than the number of computable nodes, node Vol is used to set the maximum number of indices sub-nodes allowed for each computed node in the STR-tree.

(2) Sending of the subtree index

The primary node sends the index entries to the child nodes according to a certain replication and allocation policy. The data structure uses an in-memory spatial index. Unlike the traditional hard disk-based spatial index, the in-memory spatial index has the characteristics of dynamic construction and fast query.

(3) The master node sets up the Master-R-tree

The master node builds a new spatial index into all non-leaf nodes in the original entire R-tree index tree in computer memory. The index tree records the index item information allocated by each computing node, and its function is equivalent to an overall index controller. The intersecting query of the active overlay layer is first filtered by the overall index tree.

(4) Computing node to build a Client-tree

According to the agreed STR-tree node organization rule, the index items sent by the master node are built into the client terminal tree. The entity object stored in the computer node is associated with the index item of the subtree, and the entity space data is extracted according to the index item MBR.

The parallel R-tree index decomposition based on the STR rule is analyzed. The essence is to replace the random partition in the form of elemental fid with the domain decomposition partition considering the neighboring spatial rule. The master node controls the overall spatial index R-tree of a large node, where the id information of the spatial data domain decomposition block in the cluster is recorded. The active layer first performs a query operation with the overall index tree. If one or more intersecting subdomains are queried, the master node notifies the child nodes having the subdomains to extract the active layer locally. The element is further intersected with the subtree, and the process will be analyzed in the parallelization of the intersection.

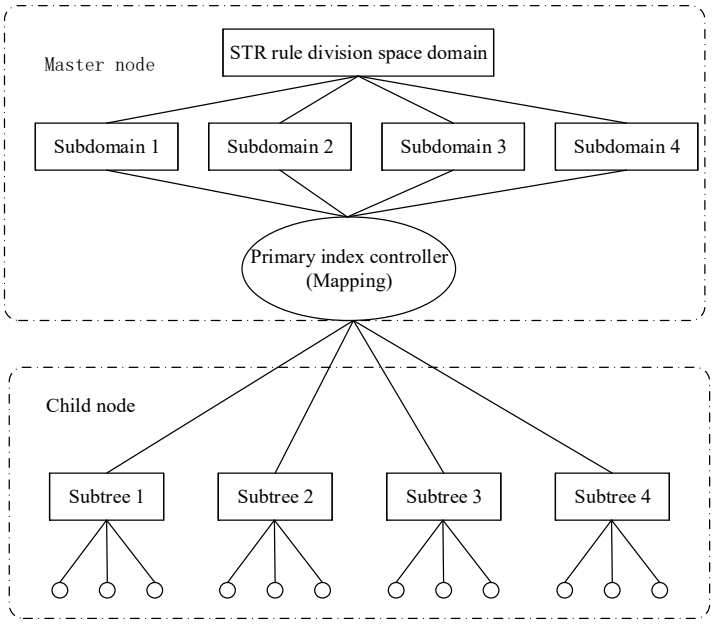

Figure 2. Sort-Tile-Recursive (STR)-tree data decomposition.

3.2. Multi-Strategy Hilbert Ordering Decomposition

It can be seen from the process of generating Hilbert code values that the determination of the code value is determined by the center point of the space grid, which is called the order method of the midpoint strategy. This section studies the difference between the middle strategy and the median strategy. Hilbert curve sorting is used to optimize the problem so that the sorting method can better take into account the spatial proximity of map features. The spatial sort is the reversible 1:1 correspondence between spatial feature points and consecutive integers in one-dimensional space as key values.

The middle strategy is easier to understand. In the process of dividing the spatial hierarchy, each cell is divided according to a fixed size, that is, at the middle point of the cell. The midpoint strategy is suitable for point set objects with a more balanced distribution in the low dimension in practical applications.

Unlike the middle strategy, the median strategy divides the cells in the space according to the median point in the x or y direction of the point to be sorted (Figure 3). The geometric midpoint is an effective solution to the positioning of resource rules. In the high dimensional space, the median point can better represent the centrality of the clustering point. The median strategy is adapted to high-dimensional point sets or point set objects with uneven distribution.

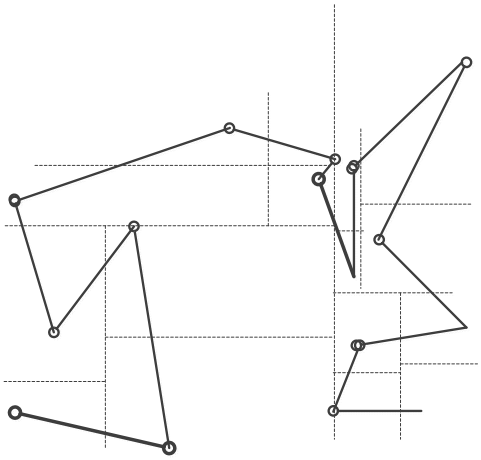

Figure 3. The demonstration of the Hilbert median sort demo.

From the definition of the median and the application context, the characteristics of maintaining clustering in Hilbert ordering can be better understood. The median is a concept in sequential statistics. In a set of n elements, the nth order statistic is the i-th small element in the set. For example, in a set of elements, the minimum is the first order statistic ($i = 1$), and the maximum is the nth order statistic ($n = 1$). An informal definition of a median is the "midpoint element" of the collection it is in. When n is an odd number, the set has a unique median at $i = ((n + 1))/2$ out; when n is even, it has two medians, which appear at $i = n/2$ and $i = n/(2 + 1)$. If not considering the parity of n, the median always appears at $i = \lceil((n + 1))/2\rceil$ (upper median) and $i = \lfloor((n + 1))/2\rfloor$ (lower median). In our study, the Hilbert median strategy sorting adopts the lower median by default. The CGAL library is used to assist Hilbert in sorting in the decomposition process of spatial data. Because the spatial data cannot be extracted from the data source directly according to the Hilbert code value, it must be the main processing data in parallel computing. After the node establishes the mapping relationship with the spatial database, the data is decomposed. Figure 4 shows the world map divided by two different Hilbert sorting methods. Each color denotes the different classes for the result of Hilbert sorting methods.

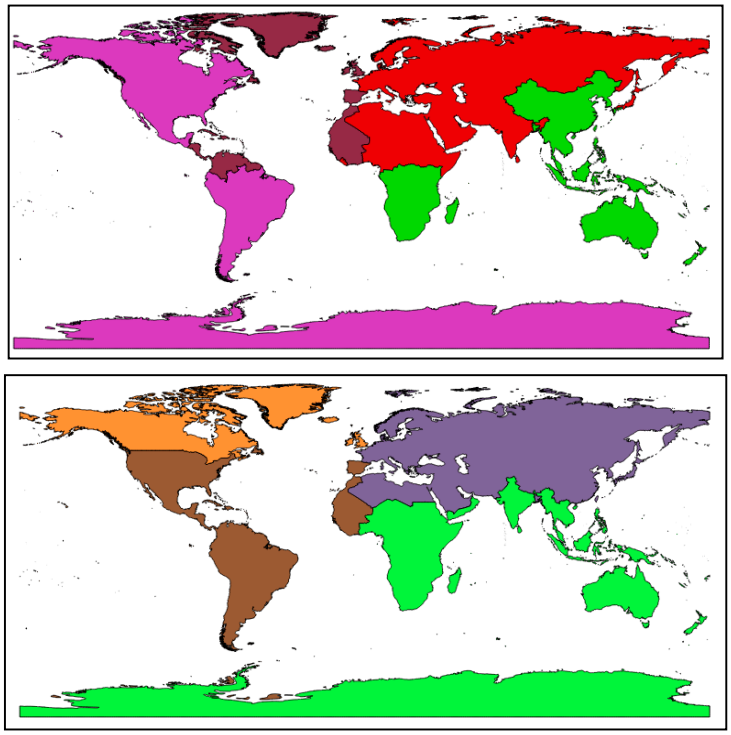

Figure 4. Hilbert middle and median sort for world map (4 nodes).

The process based on Hilbert's sorting decomposition is shown in Figure 5. First, the feature points are extracted from the feature elements, and then the appropriate Hilbert space sorting strategy is selected according to the distribution characteristics of the spatial data. After Hilbert sorting, the main computing nodes follow the order of the one-dimensional Hilbert curve according to the multi-channel form, and the data-polling type is generated to each sub-node according to the interval division. Because the storage of vector data in this paper takes the form of a spatial database, what is referred to here is the location tag of the object in the spatial database table.

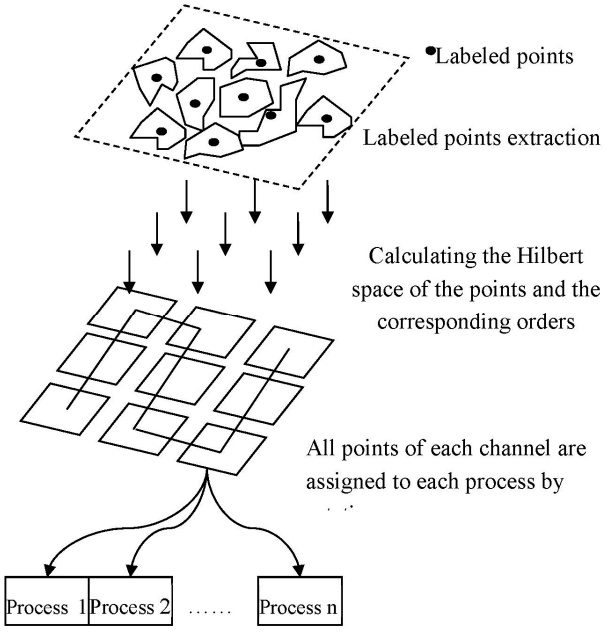

Figure 5. Spatial data decomposition using Hilbert sort.

The most crucial purpose of spatial data domain decomposition in parallel GIS computing systems is that each child node can perform spatial data extraction or query work at the same time, without pulling information from the global control node; however, the traditional data partitioning method is based on the static clustering method. Reasonable spatial data partitioning should consider the spatial location distribution and length uncertainty of geographic data. It can be considered from two aspects whether the decomposition strategy of Hilbert ordering is the best, including whether each piece of sub-data can maintain spatial proximity to the maximum extent and whether the storage capacity of the data blocks in each node is equalized.

3.3. Parallel Spatial Union Algorithm

The parallel spatial union algorithm process is as follows:

(1) Polygon data feature point extraction

The Hilbert curve sorted objects are point objects with integer values, and the ordering of polygons was converted to the ordering of their representative feature points. The feature points of a polygon usually have an outer center point of the rectangle or four corner points, a geometric center point, and a mass center point. Since the calculation of the center point and the center of gravity of the polygon is time-consuming, it is not conducive to the parallel acceleration of the map overlay analysis. Therefore, the feature point of the outsourced rectangular is the center point. All the layers S are regarded as a set of all the layers L, and the feature point extraction is represented as, among them, a feature point set. The feature point set is constructed by the mapping function map.

(2) Feature point Hilbert space ordering

The function corresponding to the Hilbert curve sort is described as follows. Among them are integer coordinate values, and d is the Hilbert code value. For floating-point data in practical applications, it is necessary to scale to the nearest integer value. Constructing a Hilbert curve in a two-dimensional space maps the interval to a unit square interval, and each cell is recursively divided into four parts until each cell contains only one feature point. The computational complexity of constructing the Hilbert sorting curve algorithm is, where n is the number of hierarchical divisions of the Hilbert space. In the calculation process of Hilbert coded values in this paper, the value of n is not forced to be constrained so as to ensure that the point object clustering is denser and still maintains a good mapping unity.

In the parallel overlay analysis, Hilbert curve ordering is performed according to the distribution characteristics of the feature elements using two strategies: middle strategy and median strategy.

(3) Distribution of nearby polygons

In the parallel overlay analysis system, the Master node is responsible for loading all the map layers involved in the calculation and establishing the Hilbert sort number for its content. The elements in each layer are evenly distributed to each node according to the number of compute nodes. The number of layer elements and number of processors is n. It is necessary to map the code on the Hilbert curve string to the FID of the map element. The generation of the one–to–one correspondence depends on the previous Hilbert ordering process in which there are two mapping relationships, and the two mapping relationships are connected by feature point coordinates. The specific process can be described by the following formula, which is the Hilbert code value, the FID marked element, and the data extraction condition as the overlay query.

The implementation process of the Hilbert joint algorithm is as follows:

(i) A data partitioning strategy. The experiment uses three data distribution strategies in the order of fid, Hilbert_median, and Hilbert_middle. First, the master node determines the number of nodes participating in the calculation, and then equally distributes the ordered sets separately.

(ii) Independent calculations, generating intermediate results. Each computing node is responsible for the joint operation of the local polygons and writes the generated intermediate results to a temporary table in the MySQL spatial database.

(iii) Results are collected in conjunction. The master node loads the middle layer into the local memory in the order in which the child nodes complete the task, and then jointly outputs the final result layer to these layers.

The previous part is shown the operation of a single layer, the data decomposition hierarchy object is single, and it is easier to divide and conquer. For the overlaying and merging of two or more layers, it is necessary to consider the spatial data distribution of multiple layer species, such as the case of the line surface overlaying, to prevent the layer overlaying of the spatial distribution imbalance.

From the perspective of spatial data decomposition parallelism, the parallel stacking of multiple layers can use two methods.

(i) According to the above design, the Hilbert sorting decomposition parallel method is based on the spatial ordering of multi-level two-dimensional plan layers through high-dimensional Hilbert curves. For the processing of two-dimensional plane map data, the specific process can be divided into two processes of reducing the dimension and raising the dimension, but the two are not contradictory. Dimensionality reduction is for a polygon layer, extracting a single point from a polygon in the layer, such as the center point of the outer frame, a corner point, or the Centre's CentreID point to uniquely identify the polygon as a mapping to the one-dimensional Hilbert function.

The polygon objects in the two layers were loaded into the same container, and the geometric objects in the container were manipulated according to the Hilbert parallel sorting algorithm to merge the individual layers. The advantage of this method is that it is not necessary to consider the combined correspondence of polygons in the container, and it can be divided according to Hilbert ordering. Polygons that do not want to intersect in the original single layer may be indirectly connected due to the addition of new layer features, as shown in Figure 6. Indirect connectivity increases the complexity of parallel association. If the polygons in the two layers are only a one–to–one relationship, they can be directly merged in batch form. The polygons may be spread in pieces when considering indirect connectivity. In this way, the parallel joint process needs to continue the binary tree merging of the data decomposition results.

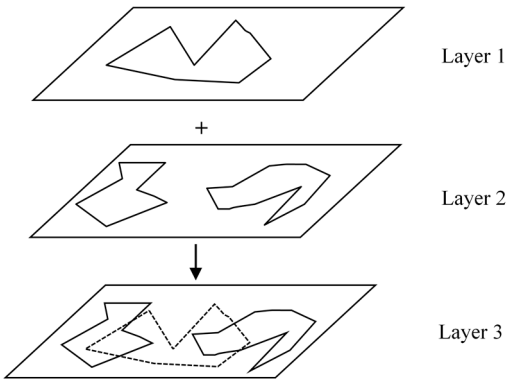

Figure 6. Connect relationship between two layers.

(ii) A combination of multi-layer filtering mechanisms using parallel master-slave indexes.

The master node establishes a coarse-grained spatial index for the two layers, which is equivalent to establishing a distribution for each layer to establish m and n spatial index subtrees. First, test the intersection of the root nodes of these subtrees and filter the separated subtrees. The information of the intersecting subtrees is sent to the child nodes for the merge operation of the independent subtrees. The merge mode can be understood as spatial analysis of different spatial data of different spatial granularity levels, which can be performed based on spatial data with lower spatial granularity level, and spatial analysis of different spatial data of the same spatial granularity level must be superimposed

to generate Spatial data with a lower spatial granularity level, which is then based on spatial data with a lower spatial granularity level.

According to the implementation principle of the parallel master-slave index union, with data decomposition based on a spatial index, it is difficult to completely eliminate the redundancy of index objects, so different nodes will have the same geometric object, and the subtree merge is a recursive consumption. At the time of operation, the last master still needs to recycle and combine the intermediate results. The sprawling characteristics of joint operations determine that they cannot be parallelized in the form of batch processing. The parallel combination of master-slave index tree mode is difficult to implement in practical applications. The parallel joint research in this paper uses Hilbert as the experimental subject for the two layers as a whole, and the parallel master-slave index joint will be further studied in the next step.

In our study, two joint layers are treated as an overall virtual layer and parallelized according to the strategy of polygon merging in the same layer (Figure 7). The specific process is as follows:

(i) First, create a data structure featureInfo for each feature in each layer. The main member variables are fid, layer_num, hilbert_code, and mbr_pt. The fid is used to record the cursor of the feature in the layer. As a query condition for quickly extracting data in the future, layer_num records which layer is one of the filtering conditions of the data query. The hilbert_code defaults to a null value when the FeatureInfo is initialized, which is used to record the encoding of the Hilbert space sorting. Mbr_pt records the center point of the desired MBR as a feature point for Hilbert ordering.

```
typedef struct
{
    long fid, hilber_code;
    GTPoint* mbr_pt;
} featureInfo;
```

(ii) Put the featureInfo in the two layers into the vector array Vector<featureInfo*> and use the Hilbert spatial sorting function hilbert_sort() to sort and encode the elements in the array.

(iii) According to the number k of available computing nodes, the total n elements are sent to one node every n/k according to the Hilbert coding order. The polling method only sends the Hilbert space sorting number interval instead of sending the specific geometry object.

(iv) The child node receives the Hilbert space sorting coding interval, extracts the geometric objects of the feature from the local database according to fid and layer_num in featureInfo, and starts to merge the intersecting polygons in the interval.

(v) Continue the merge in the form of a single layer binary tree merge until all intersecting polygons are merged.

According to this layer, the overall layer is added to perform Hilbert sorting decomposition. The advantage is that the data is divided into non-redundant polygons of each node. The unique characteristics of Hilbert coded values can ensure the low coupling of joint calculation between processes. The binary tree parallel combination can reduce the pressure of the main node recovery. Therefore, the scheme is feasible, and theoretically, an acceleration effect can be obtained. The following is an experimental verification of the multi-layer Hilbert joint method using actual data.

The computational complexity of this algorithm is O(N+K), which is derived from the count of polygons for the overlayed map layers. Given N polygons in each layer, if using the brute-force overlay test, the complexity of this algorithm should be $O(N^2)$. However, we performed an intersecting filter process that can exclude the unnecessary computing for polygons that do not overlay to each other in the space. Thus, there would be K polygons in the overlayed layer that participate in overlay computation. Ultimately, the complexity of this algorithm should be O(N+K).

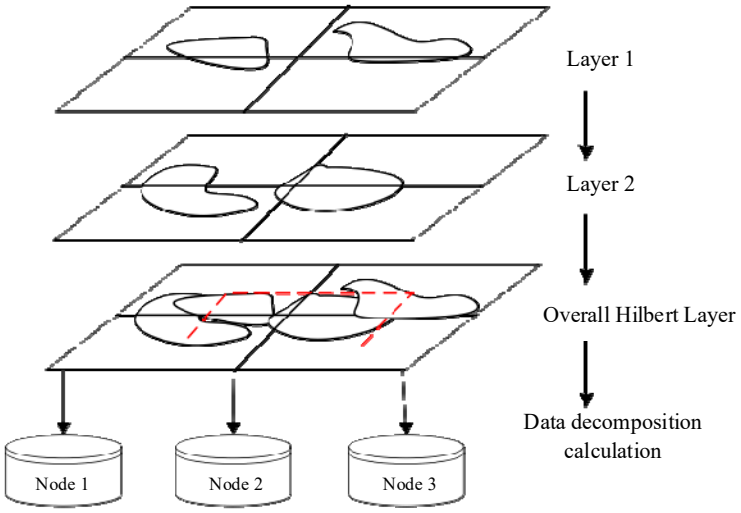

Figure 7. Hilbert decomposition for two layers.

4. Experiments

The parallel algorithm for polygons overlay analysis was implemented, and the performance of the parallel spatial union was verified. Multi-strategy based spatial data decomposition mechanism was implemented, including parallel spatial data index, the Hilbert space-filling curve sort, and spatial data decomposition. The programming language was C++. The operating system of each computation node was 64-bit Ubuntu Linux (16.04), and run on Intel CPU based hardware architecture (Model: Intel Core i7-3960X, 3.8 GHz, 12 cores).

4.1. The Experiment for Parallel Spatial Union

The experiment used a spatial union in map overlay analysis as an example to conduct parallelization experiments. The experimental data shows the Chinese 1:4 million land use map. The data type was multi-polygon, the number of elements was 15,615, and the number of points was 886,547 (Figure 8). The visualization strategy in Figure 8 is FID based strategy. The experimental dataset is a type of data commonly used in GIS applications, representing the general characteristics of spatial data decomposition. The spatial distribution of these maps data is irregular and unbalanced, and the amount of data is large. It is representative of experiments and has guiding significance for practical applications. Figures 9–11 are the results for the parallel spatial union using fid decomposition, median decomposition, and middle decomposition. The number of the results for the decomposition method is 2, 4, 8, and 16, respectively. Different color denotes different class in these maps.

Figure 8. The landuse map of China.

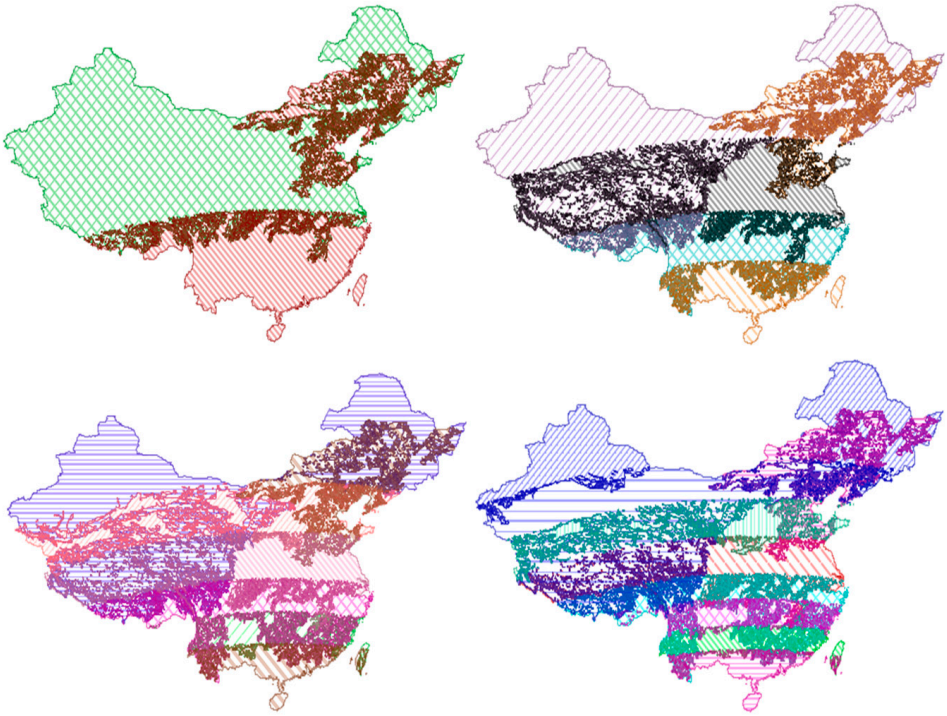

Figure 9. Parallel spatial union using the fid decomposition method.

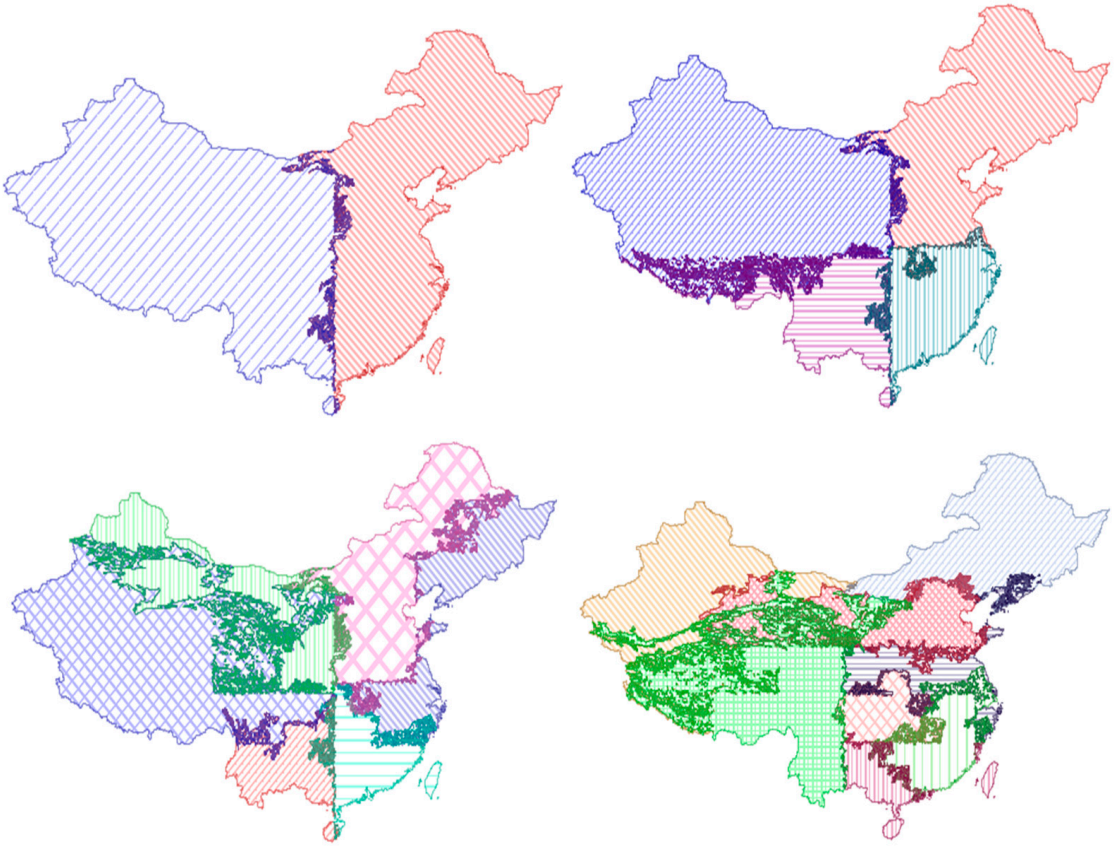

Figure 10. Parallel spatial union using the median decomposition method.

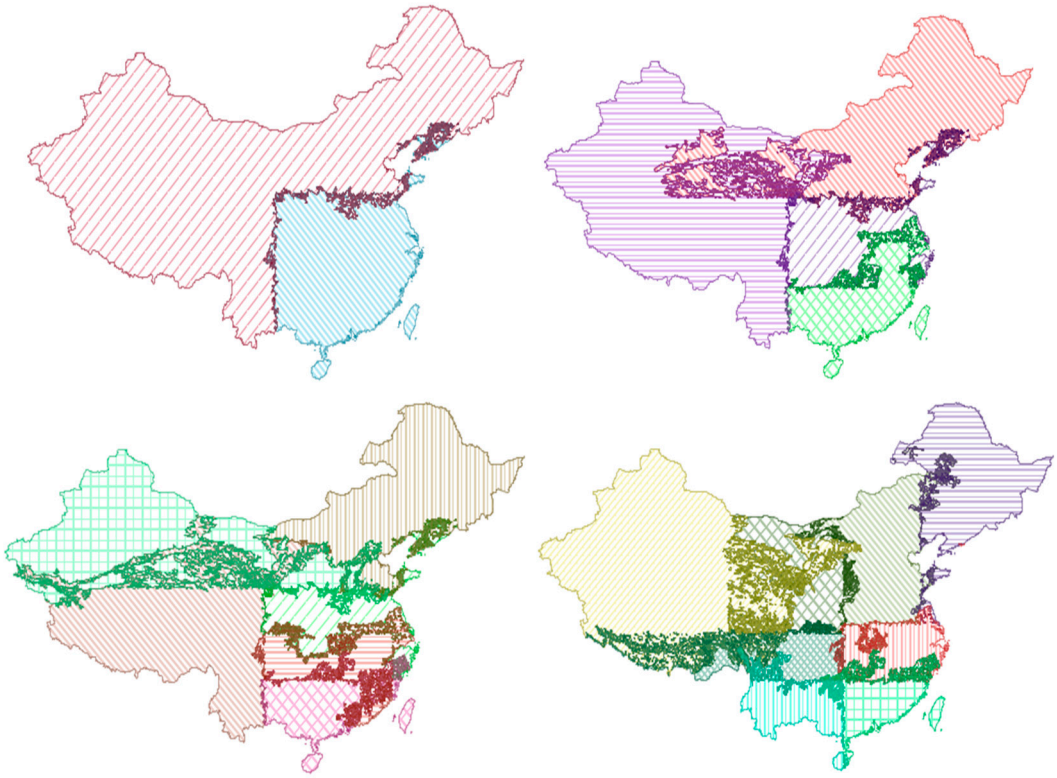

Figure 11. Parallel spatial union using the middle decomposition method.

Figure 12 shows the resource consumption of a computer when the parallel overlay is used to implement Hilbert ordering using the MPI method. The system starts a total of 10 computing processes. It can be seen from the figure that the CPU usage of each process is basically the same in the parallel computing process, and the memory usage averages about 1.3%.

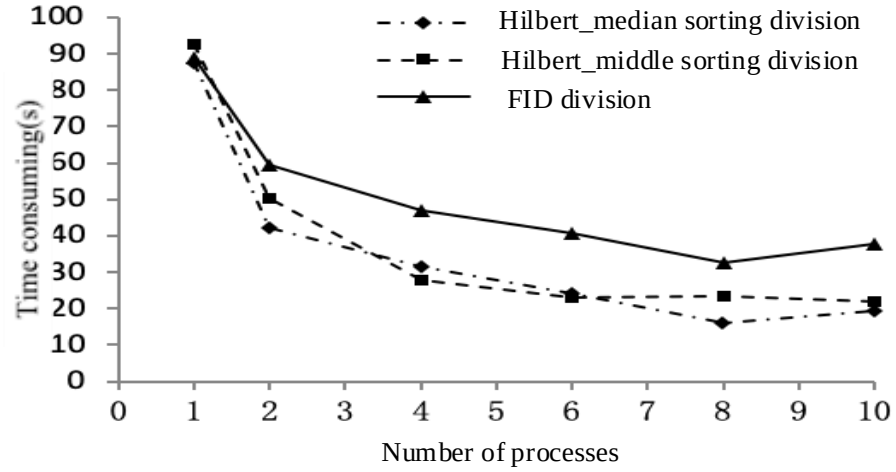

Figure 12. Computing time for parallel spatial union.

According to the experimental results, the parallel merging strategy is analyzed. From the comparison of FID partitioning and Hilbert partitioning, the results of FID partitioning are scattered. The merging results in the same sub-node are not necessarily adjacent. Hilbert divides the merging result better, for example, if the FID is merged, even if there are only two computing nodes, the combined result of the first node also includes part of the combined result of the second node. At the same time, it can be observed that the combined results of FID partitioning are basically horizontal

strips, and the Hilbert partitioning results are basically blocky. It can also be observed from the merger of the Taiwan Province that the fid division strategy will be divided into multiple blocks (the whole map), the islands will be divided into multiple blocks for processing, and the Hilbert spatial sorting combination will ensure a small area and approximate shape. The circular polygonal area maintains functional integrity.

The parallel computing time and acceleration ratio effects of the three merge modes are shown in Figures 13 and 14. The parallel computing time is the sum of the data decomposition time, the independent calculation time, and the last merge time of the master node. The calculation time has decreased with the increase of the number of processes; however, the acceleration effect is obvious when the acceleration effect is less than four processes, which is higher than the linear acceleration ratio, but the acceleration effect is significantly reduced when there are more than four processes because of two aspects: one is that if there are multiple processes in one child node, the data is concurrent with I/O, and the other is that multiple child nodes send writing database requests to the master node. The number of polygons that the master node needs to merge increases, and the database is locked as the writing process takes time. When the number of threads is greater than eight, the median strategy shows a trend of decreasing the acceleration ratio, indicating that the consumption of the collection and consolidation of the primary node is greater than the acceleration of the parallel computing. The solution is to increase the number of physical machines and try to keep each child node running a small number of processes, reduce read-write conflicts, allow as many binary tree merges as possible between child nodes, and reduce the number of merged polygons at the primary node.

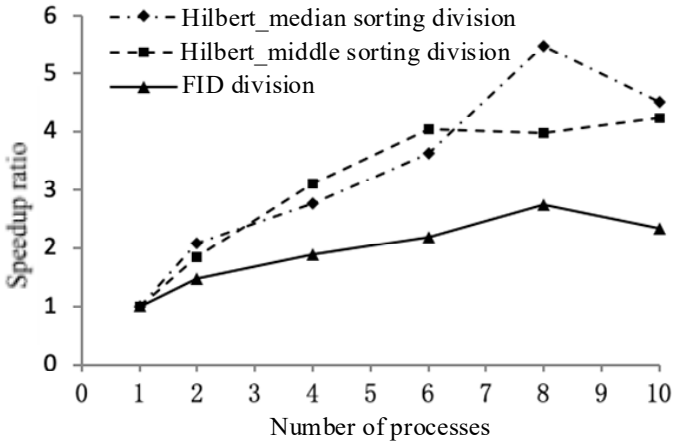

Figure 13. Speedup ratio for parallel polygon union.

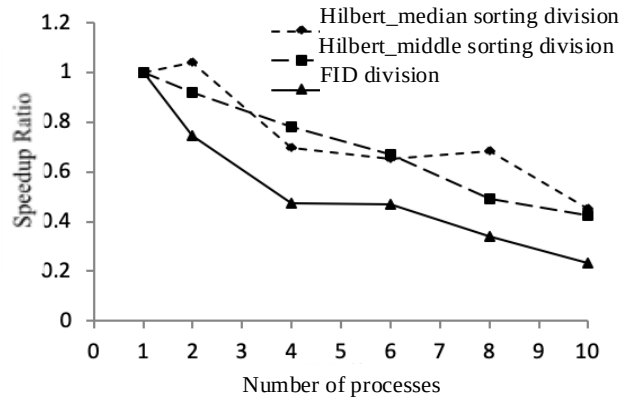

Figure 14. Speedup ratio for parallel polygon union.

From the accelerated efficiency analysis of parallelization, the acceleration efficiency is the lowest according to the strategy of the fid division. The acceleration efficiency change from 1 process to 10 processes is from 1 to 0.234, which is a large change. The reason is that the polygon data is allocated according to fid. The number of nodes actually intersecting is small, which causes many polygons to merge again when the master node collects, causing the overload of the master node to affect the overall acceleration efficiency. The acceleration efficiency of the Hilbert_median partitioning strategy varies between 2~4 processes and 8~10 processes, and the acceleration efficiency is basically unchanged in 4~8 processes. The Hilbert_middle strategy divides the efficiency with the number of processes. The increase has a linear downward trend. From the perspective of the accelerated efficiency degradation of these parallel joints, although the data decomposition strategy for estimating the spatial proximity characteristics can achieve a specific acceleration effect, there is still room for further optimization in the parallelization algorithm of the data in the reduction phase of the data.

4.2. The Experiment for Parallel Hierarchical Spatial Union

In the experimental data, the basic layer is the planning map of a certain urban area, and the overlayed layer is the post-translation planning diagram (Figure 15). The result of the spatial union operation is shown in Figure 16.

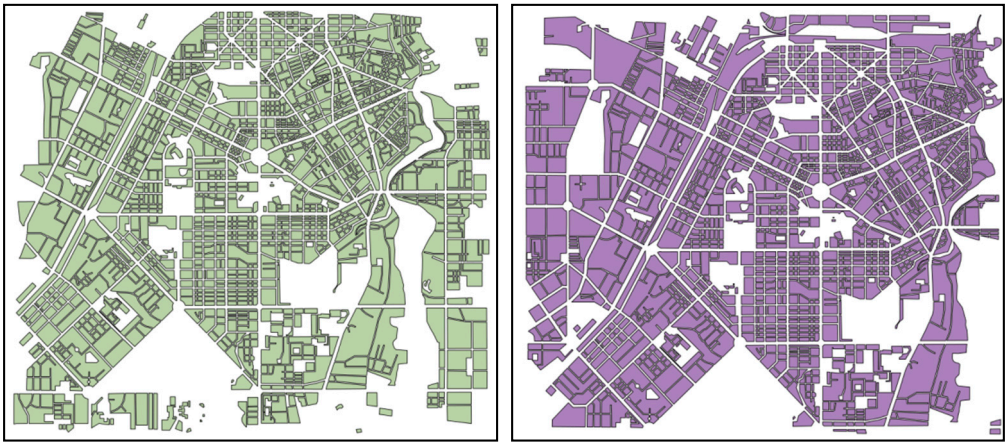

Figure 15. Test data for the Hilbert decomposition of two layers.

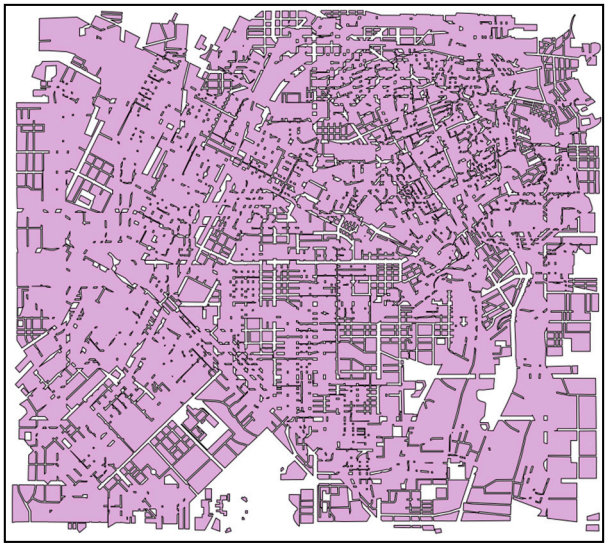

Figure 16. Two data layers' union result.

5. Conclusions and Future Work

In this study, we propose an efficient parallel algorithm for polygons overlay analysis, including active-slave spatial index decomposition for intersection, multi-strategy Hilbert ordering decomposition, and a parallel spatial union algorithm. A multi-strategy based spatial data decomposition mechanism is implemented, including parallel spatial data index, the Hilbert space-filling curve sort, and data decomposition. The results of the benchmark experiments showed that the parallel algorithm for polygons overlay analysis achieves high efficiency.

However, there are some limitations to this study. Firstly, in this study, we have not discussed cloud computing and edge computing for map overlay analysis. The new computing framework can improve the efficiency of spatial analysis for large-scale geospatial data [37–39]. Secondly, graphics processing unit (GPU)-based geo-computation is not considered. GPU is a potential factor that could improve the speedup ratio [40]. In the future, we will improve the study by addressing those limitations. Firstly, the map overlay algorithm based on cloud computing will be proposed. Secondly, we would take account of GPU-based parallel algorithms for map overlay in the future.

Author Contributions: Conceptualization, S.W.; data curation, Y.Z., S.W., and Y.G.; funding acquisition, Y.Z.; investigation, Y.Z. and S.W.; methodology, Y.Z. and S.W.; project administration, Y.Z. and S.W.; software, Y.Z., S.W. and Y.G.; supervision, Y.Z. and S.W.; validation, Y.G.; visualization, Y.Z. and Y.G.; writing—original draft, Y.Z., S.W., and Y.G.

Funding: This research was funded by the National Natural Science Foundation of China (grant number 41601478) and the National Key Research and Development Program (grant numbers 2018YFB0505301, 2016YFC0500103).

Acknowledgments: We would like to acknowledge the Institute of Geographic Sciences and Natural Resources Research of the Chinese Academy of Science for providing the research grant to conduct this work.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wang, S.; Zhong, E.; Lu, H.; Guo, H.; Long, L. An effective algorithm for lines and polygons overlay analysis using uniform spatial grid indexing. In Proceedings of the 2015 2nd IEEE International Conference on Spatial Data Mining and Geographical Knowledge Services (ICSDM), Fuzhou, China, 8–10 July 2015; pp. 175–179.
2. Zhao, K.; Jin, B.; Fan, H.; Song, W.; Zhou, S.; Jiang, Y. High-performance overlay analysis of massive geographic polygons that considers shape complexity in a cloud environment. *ISPRS Int. J. Geo Inf.* **2019**, *8*, 290. [CrossRef]
3. You, S.; Zhang, J.; Gruenwald, L. Large-scale spatial join query processing in cloud. In Proceedings of the 2015 31st IEEE International Conference on Data Engineering Workshops, Seoul, Korea, 13–17 April 2015; pp. 34–41.
4. Zhou, Q.; Zhong, E.; Huang, Y. A parallel line segment intersection strategy based on uniform grids. *Geo Spat. Inf. Sci.* **2009**, *12*, 257–264. [CrossRef]
5. Puri, S.; Agarwal, D.; He, X.; Prasad, S.K. Mapreduce algorithms for gis polygonal overlay processing. In Proceedings of the 2013 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum, Cambridge, MA, USA, 20–24 May 2013; pp. 1009–1016.
6. Agarwal, D.; Puri, S.; He, X.; Prasad, S.K. A system for gis polygonal overlay computation on linux cluster-an experience and performance report. In Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum, Shanghai, China, 21–25 May 2012; pp. 1433–1439.
7. Zhou, C.; Chen, Z.; Li, M. A parallel method to accelerate spatial operations involving polygon intersections. *Int. J. Geogr. Inf. Sci.* **2018**, *32*, 2402–2426. [CrossRef]
8. Wang, S.; Armstrong, M.P. A quadtree approach to domain decomposition for spatial interpolation in grid computing environments. *Parallel Comput.* **2003**, *29*, 1481–1504. [CrossRef]
9. Kantere, V.; Skiadopoulos, S.; Sellis, T. Storing and indexing spatial data in p2p systems. *IEEE Trans. Knowl. Data Eng.* **2008**, *21*, 287–300. [CrossRef]
10. Wang, F. A parallel intersection algorithm for vector polygon overlay. *IEEE Comput. Graphics Appl.* **1993**, *13*, 74–81. [CrossRef]

11. Agarwal, D.; Puri, S.; He, X.; Prasad, S.K.; Shi, X. *Crayons—A Cloud Based Parallel Framework for Gis Overlay Operations*; Distributed & Mobile Systems Lab: New York, NY, USA, 2011.
12. Healey, R.; Dowers, S.; Gittings, B.; Mineter, M.J. *Parallel Processing Algorithms for Gis*; CRC Press: Boca Raton, FL, USA, 1997.
13. Mineter, M.J. A software framework to create vector-topology in parallel gis operations. *Int. J. Geogr. Inf. Sci.* **2003**, *17*, 203–222. [CrossRef]
14. Xiao, N. *GIS Algorithms*; Sage: Thousand Oaks, CA, USA, 2015.
15. Ching, Y.-T. Load balancing for the parallel map overlay-operation in the geographic information system. *J. Inf. Sci. Eng.* **1999**, *15*, 441–449.
16. Xun, L.; Wenfeng, Z. Parallel spatial index algorithm based on hilbert partition. In Proceedings of the 2013 International Conference on Computational and Information Sciences, Shiyang, China, 21–23 June 2013; pp. 876–879.
17. Papadopoulos, A.; Manolopoulos, Y. Parallel bulk-loading of spatial data. *Parallel Comput.* **2003**, *29*, 1419–1444. [CrossRef]
18. Naylor, B. Binary space partitioning trees as an alternative representation of polytopes. *Comput. Aided Des.* **1990**, *22*, 250–252. [CrossRef]
19. Samet, H. *The Design and Analysis of Spatial Data Structures*; Addison-Wesley Reading: Reading, MA, USA, 1990; Volume 85.
20. Peuquet, D.J. A conceptual framework and comparison of spatial data models. *Cartogr. Int. J. Geogr. Inf. Geovis.* **1984**, *21*, 66–113. [CrossRef]
21. Minasny, B.; McBratney, A.B.; Walvoort, D.J. The variance quadtree algorithm: Use for spatial sampling design. *Comput. Geosci.* **2007**, *33*, 383–392. [CrossRef]
22. Chen, S.; Ooi, B.C.; Tan, K.-L.; Nascimento, M.A. St 2 b-tree: A self-tunable spatio-temporal b+-tree index for moving objects. In Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, Ancouver, CB, Canada, 9–12 June 2008; pp. 29–42.
23. Guttman, A. *R-Trees: A Dynamic Index Structure for Spatial Searching*; ACM: New York, NY, USA, 1984; Volume 14.
24. Kamel, I.; Faloutsos, C. Hilbert R-Tree: An Improved R-Tree Using Fractals; College Park, MD, USA. 1993. Available online: https://www.researchgate.net/publication/2303411_Hilbert_R-Tree_An_Improved_R-Tree_Using_Fractals (accessed on 13 November 2019).
25. Hamilton, C.H.; Rau-Chaplin, A. Compact hilbert indices for multi-dimensional data. In *Proceedings of the First International Conference on Complex, Intelligent and Software Intensive Systems (CISIS'07), Vienna, Austria, 10–12 April 2007*; IEEE: New York, NY, USA, 2007; pp. 139–146.
26. Arge, L.; Hinrichs, K.H.; Vahrenhold, J.; Vitter, J.S. Efficient bulk operations on dynamic r-trees. *Algorithmica* **2002**, *33*, 104–128. [CrossRef]
27. Leutenegger, S.T.; Lopez, M.A.; Edgington, J. Str: A simple and efficient algorithm for r-tree packing. In Proceedings of the 13th International Conference on Data Engineering, Birmingham, UK, 7–11 April 1997; pp. 497–506.
28. Zhu, Q.; Gong, J.; Zhang, Y. An efficient 3d r-tree spatial index method for virtual geographic environments. *ISPRS* **2007**, *62*, 217–224. [CrossRef]
29. Ferhatosmanoglu, H.; Tuncel, E.; Agrawal, D.; El Abbadi, A. Vector approximation based indexing for non-uniform high dimensional data sets. In Proceedings of the 9th International Conference on Information and Knowledge Management, McLean, VA, USA, 6–11 November 2000; pp. 202–209.
30. Zhang, C.; Pan, M. A study on topological reconstruction of gis vector data based on grid index. *Geogr. Geo Inf. Sci.* **2006**, *22*, 20–24.
31. Obe, R.O.; Hsu, L.S. *Postgis in Action*; Manning Greenwich: New York, NY, USA, 2011; Volume 2.
32. Hu, Y.; Ravada, S.; Anderson, R.; Bamba, B. Topological relationship query processing for complex regions in oracle spatial. In Proceedings of the 20th International Conference on Advances in Geographic Information Systems, Redondo Beach, CA, USA, 6–9 November 2012; pp. 3–12.
33. Gaskill, J.; Brooks, D. Understanding arcsde. In Proceedings of the Annual ESRI International User Conference, Paris, France, 3–7 July 2000.

34. Zhang, M.; Lu, F.; Cheng, C. A forced transplant algorithm for dynamic r-tree implementation. In *Proceedings of the International Conference on Database and Expert Systems Applications, Kraków, Poland, 4–8 September 2006*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 459–466.
35. Tanin, E.; Harwood, A.; Samet, H. Using a distributed quadtree index in peer-to-peer networks. *VLDB J.* **2007**, *16*, 165–178. [CrossRef]
36. Schnitzer, B.; Leutenegger, S.T. Master-client r-trees: A new parallel r-tree architecture. In Proceedings of the Eleventh International Conference on Scientific and Statistical Database Management, Cleveland, OH, USA, 30 July 1999; pp. 68–77.
37. Wang, S.; Zhong, Y.; Wang, E. An integrated gis platform architecture for spatiotemporal big data. *Future Gener. Comput. Syst.* **2019**, *94*, 160–172. [CrossRef]
38. Gui, Z.; Yang, C.; Xia, J.; Huang, Q.; Liu, K.; Li, Z.; Yu, M.; Sun, M.; Zhou, N.; Jin, B. A service brokering and recommendation mechanism for better selecting cloud services. *PLoS ONE* **2014**, *9*, e105297. [CrossRef] [PubMed]
39. Yang, C.; Yu, M.; Hu, F.; Jiang, Y.; Li, Y. Utilizing cloud computing to address big geospatial data challenges. *Comput. Environ. Urban Syst.* **2017**, *61*, 120–128. [CrossRef]
40. Heitzler, M.; Lam, J.C.; Hackl, J.; Adey, B.T.; Hurni, L. Gpu-accelerated rendering methods to visually analyze large-scale disaster simulation data. *J. Geovis. Spat. Anal.* **2017**, *1*, 3. [CrossRef]