International Journal of Geo-Information

MDPI

Article

Using High-Performance Computing to Address the Challenge of Land Use/Land Cover Change Analysis on Spatial Big Data

Xiaochen Kang [1,*], Jiping Liu [1,2,*], Chun Dong [1] and Shenghua Xu [1]

[1] Research Center of Government Geographic Information System, Chinese Academy of Surveying and Mapping, Beijing 100830, China; dongchun@casm.ac.cn (C.D.); xushh@casm.ac.cn (S.X.)
[2] Institute of Geographical Sciences, Henan Academy of Sciences, Zhengzhou 450052, China
* Correspondence: kangxc@casm.ac.cn (X.K.); liujp@casm.ac.cn (J.L.)

Received: 30 April 2018; Accepted: 6 July 2018; Published: 11 July 2018

Abstract: Land use/land cover change (LUCC) analysis is a fundamental issue in regional and global geography that can accurately reflect the diversity of landscapes and detect the differences or changes on the earth's surface. However, a very heavy computational load is often unavoidable, especially when processing multi-temporal land cover data with fine spatial resolution using more complicated procedures, which often takes a long time when performing the LUCC analysis over large areas. This paper employs a graph-based spatial decomposition that represents the computational loads as graph vertices and edges and then uses a balanced graph partitioning to decompose the LUCC analysis on spatial big data. For the decomposing tasks, a stream scheduling method is developed to exploit the parallelism in data moving, clipping, overlay analysis, area calculation and transition matrix building. Finally, a change analysis is performed on the land cover data from 2015 to 2016 in China, with each piece of temporal data containing approximately 260 million complex polygons. It took less than 6 h in a cluster with 15 workstations, which was an indispensable task that may surpass two weeks without any optimization.

Keywords: LUCC analysis; spatial big data; high-performance computing; spatial decomposition

1. Introduction

Land surface changes result from numerous driving forces, including tropical deforestation, rangeland modification, agricultural intensification, urbanization, social-political factors and the worldwide interconnectedness of places and people [1–5]. Telecoupling provides a new avenue of research that enables natural and social scientists to understand and generate information for managing how humans and nature sustainably coexist [6–10]. With global issues such as climate change, surging population growth and continued energy shortages, more focus is being directed toward global environmental changes [10–12]. As landscapes are enormously complex systems, most methods to study the driving forces decompose the system into different functional components and then describe the state of the landscape, the processes within the landscape, and the reactions of the landscape over time [3,13,14]. In addition, monitoring the locations and distributions of land cover change is important for establishing links between policy decisions, regulatory actions and the resulting land-use activities [15,16]. Land use/land cover change (LUCC) analysis has become a fundamental component of environmental change and sustainability research [17,18]. As stated by Li [19], except for the analysis of the driving mechanism of LUCC, other existing studies are currently dedicated to designing reasonable methods to detect the changes more accurately [11,20–22] or predicting the change trends of the spatial-temporal progress of LUCC [23–26], which can be performed at a fine scale [27–32] or a global scale [33–37].

Over the last few decades, dramatic changes have occurred throughout the world. LUCC is fundamentally a spatial process, resulting from the complex interaction of social, ecological and geophysical processes [38]. Land use maps indicating the status are more detailed for agricultural and urban land use classes than for others. Satellite data has always been the primary source for obtaining information about the earth's land surface, and a wide variety of techniques ranging from unsupervised algorithms to parametric supervised algorithms to machine learning algorithms (MLA) have been used to classify the land cover over large areas from satellite [39,40]. Generally, post-classification comparison and multi-date composite image change detection are the two most commonly used methods in change detection [41–43], and the techniques of satellite remote sensing and GIS can also be used to examine the spatial and temporal patterns of the LUCC [44]. Moreover, after classification, overlaying two layers in different time series, in raster or vector format, can directly reflect the changes in pixels or polygons [45–47]. In addition, a hybrid classification by combining different types of methods or incorporating different data sources may also provide a better choice [48,49]. In recent years, the classification accuracy of these methods has been greatly enhanced with the increase of spatial resolution, such as very high-resolution (VHR) remote sensing images [50], and the optimization of knowledge-based approaches which play a role in use-specific photointerpretation and machine learning-based algorithms [51,52]. In the near future, it is likely that more and more robust and highly adjusted approaches will emerge for recognizing objects from images.

When multi-temporal images are available, how to detect the changes among these images has become a disparate, highly variable and ever-expanding area of research [53]. After classification, the land surface can be easily partitioned into different classes of land use/land cover parcels. On this basis, some research on change analysis addresses post classification methods, termed as a post-classification comparison [54,55]. For such comparisons, pixel-by-pixel [56,57] or object-by-object comparisons [58–60] are usually conducted. As suggested by Zhou et al. [59], object-based approaches provide a better means for change detection than pixel-based methods because they can provide a more effective way to incorporate spatial information and expert knowledge into the change detection process. With the resultant objects in polygon layers from the segmentation, land parcels are further split through GIS overlay functions, and then the classification can be well preserved at the object level [1,61]. Generally, two types of spatial models are widely used when representing the earth's phenomena, vector and raster, which are the data formats for performing spatial operations [62]. Therefore, to detect the changes from multi-temporal data, GIS operations can be used on raster images, vector polygon layers, or a combination of both representing the land classifications.

With the advances in sensor and computer technology, the approach for collecting, managing and analyzing the spatial data is going through a revolution. Monitoring geographical conditions can provide consistent measurements of landscape conditions, allowing detection of both abrupt changes and slow trends over time [54]. National geographical conditions are viewed as an important part of the basic national conditions, including spatial distribution and relationships among natural elements, cultural elements, and socio-economic information [63]. Routine monitoring provides a persistent data source for the LUCC analysis, a very large amount of data and the complicated computing processes require that high-performance computing (HPC) technologies should be used to conquer these challenges. In geographical information science, geocomputation refers to the adoption of a large-scale computationally intensive approach to the problems of physical and human geography in particular, and the geosciences in general [64]. In essence, HPC is used as an important component of geospatial cyberinfrastructure (GCI) and is critical to large-scale geospatial data processing and problem-solving [65,66]. In the era of spatial big data, cluster computing, cloud computing, and GPU computing play complementary, overlapping and even crucial roles in the processing of geospatial data [67]. In fact, the utilization of HPC systems in spatial big data processing has become more and more widespread in recent years [68], such as image classification [69], land use change analysis [70], urban growth simulation [71], etc. To perform land use change analysis over massive spatial data at a national scale, a very heavy computational load is often unavoidable, especially when processing the

multi-temporal land cover data with finer resolution using more complicated procedures. Particularly, to solve this geospatial problem, a theoretical approach towards capturing the computational loads can be used to direct the two necessary steps: spatial decomposition and task scheduling [72,73].

This paper presents a new approach to this problem: a cluster computing paradigm for conducting the data moving, clipping, overlay operation, area calculation and the matrix building. As a validation case, we used it to quickly detect the land use changes over China from multi-temporal layers. It should be noted that the overlay operation is performed on digital maps in a vector format for different temporal land cover layers. However, we guess that the overlay operation could also be applied to raster maps with small changes.

The remainder of the paper is organized as follows. Section 2 introduces the detailed method, including the LUCC procedures, the manner of spatial decomposition and the stream scheduling mechanism. Next, two individual experiments based on the estimating factors are given in Section 3. Section 4 provides a detailed discussion of the obtained results and finally the main conclusions of the work are presented in Section 5.

2. Materials and Methods

2.1. Procedures of LUCC Analysis Based on Multi-Temporal Vector Objects

In China, the Geographical Conditions Monitoring (GeoCM) project aims to monitor all kinds of indexes for the land surfaces in a dynamic and quantitative way, thereby achieving geographical descriptions of the spatial distributions and spatiotemporal changes of natural, economic and social factors. Since 2013, China has initiated the first national survey of geographical conditions. By the end of 2015, the complete land cover/land use information had been collected through interpretation from high-resolution imagery. Since 2016, it has entered a routine monitoring stage, and the land cover data for 2016 and 2017 are now both available.

The produced land cover data are all in vector format, which can therefore clearly depict the vegetation coverage, water and wetlands, desert and bare land, traffic networks, residential sites and facilities, etc. In 2015, for example, over 260 million complex polygons were collected, and each of the produced polygons belonged to a specific class of land cover. At a rough level, ten classes of land parcels are involved, i.e., farm-lands, garden lands, wood-lands, grass-lands, buildings, roads, constructed bodies, digging lands, desert lands, and waters, which form a complete coverage of the land area. Meanwhile, at the finest level, there are more than 80 classes involved. The detailed classification is provided in Supplementary Materials.

With the high volume of land cover data, a natural workflow is described in Figure 1, which involves five procedures, i.e., data moving, clipping, overlay of multi-temporal layers, area calculation and transition matrix building.

The data moving is the first procedure (P_1), which downloads a partition of the data from the distributed file sharing system to the local storage, according to some spatial boundaries. Particularly, the volume of the partial data to be moved should be predetermined with reference to the computing power of the local CPU resources. In fact, the following procedures should also be considered for assessing the computational load. As the original data is produced and stored in physical chunks, some redundant chunks should also be moved to a local computer, which ensures data integrity when performing the LUCC analysis according to semantic partitions, such as administrative divisions. As data moving is typically an input/output (I/O)-intensive operation, the memory consumption and time usage are both closely related to the size of the dataset.

With the available local dataset, the clipping procedure (P_2) partitions them into some semantic chunks for the subsequent overlay operation (P_3) and area calculation (P_4). Herein, the semantic chunks are usually bounded by administrative divisions. As the data clipping may cause heavy pressures both on the CPU and hard disks, its time usage is often highly relevant to the number of

polygon vertices. According to our experiences, when managing a large dataset, the main proportion of the consumed time in this procedure is used in computation, instead of I/O.

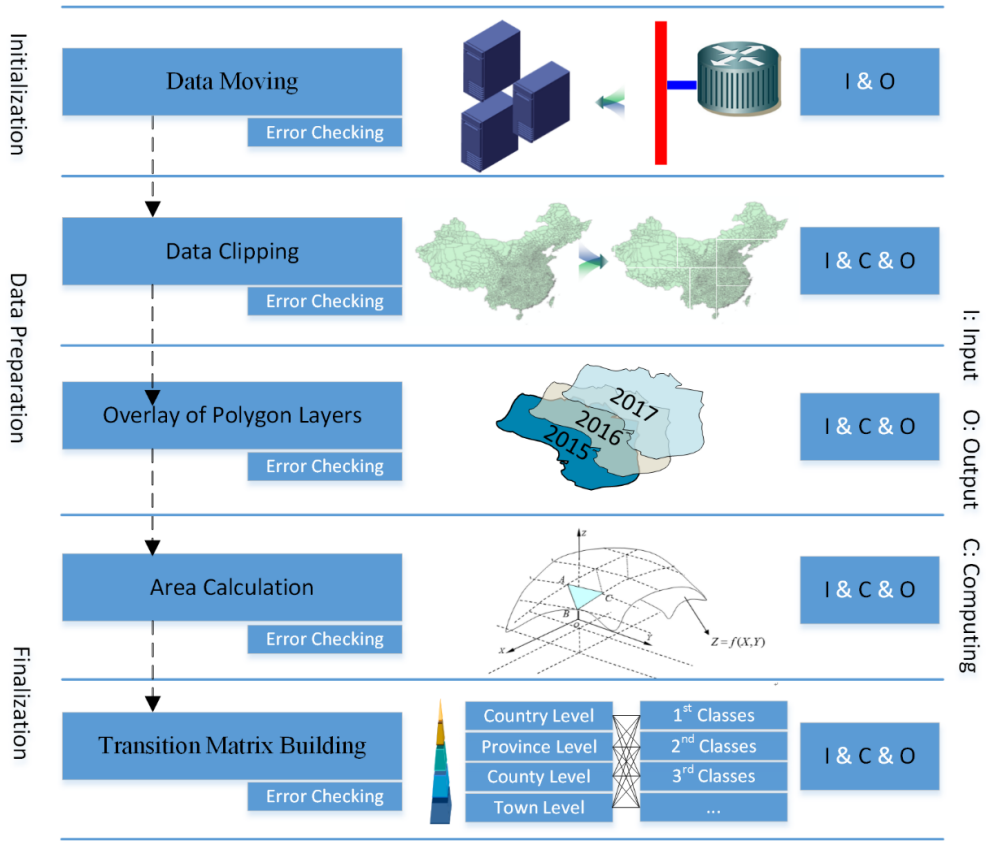

Figure 1. Workflow of the LUCC analysis.

The overlay operation creates a new layer according to two temporal layers produced in the last procedure (P_3). More importantly, the attributes of both original layers are assigned to the new layer by the intersection of the polygons. In other words, the new layer represents the LUCC information from different years. This operation is typically computationally intensive, and the time usage is determined by the number of polygon vertices to a great degree.

Area calculation is conducted to calculate the area for each of the intersected land parcels. Specifically, the area for each polygon is stored in a table row that contains the relevant information, as shown in Table 1. Naturally, area calculation is closely related to the count of the vertices in each polygon. After the above four procedures, a transition matrix is built in an MS-SQL database (P_5), which can produce the matrixes for different levels of administrative districts and different levels of land cover classes through GROUP BY and SUM operators. In comparison with the first four procedures, the time used in this procedure is negligible.

Table 1. Structure of the results for area calculation.

No.	Field Name	Notes
1	Region Code	Spatial identifier of each land parcel after intersection
2	Polygon ID	Unique identifier of each land parcel after intersection
3	Land Cover Class (from)	Land cover class before change
4	Land Cover Class (to)	Land cover class after change
5	Area	Spherical area of the land parcel after overlay operation
6	Other	A spare field

2.2. Graph-Based Spatial Decomposition

2.2.1. Administrative Level Decomposition

To perform LUCC analysis over large areas, administrative districts provide a direct and effective way for partitioning the large-scale landform, and a hierarchical structure can be drawn from the study area. Shown in Figure 2 is a 4-level hierarchy of administrative districts in China.

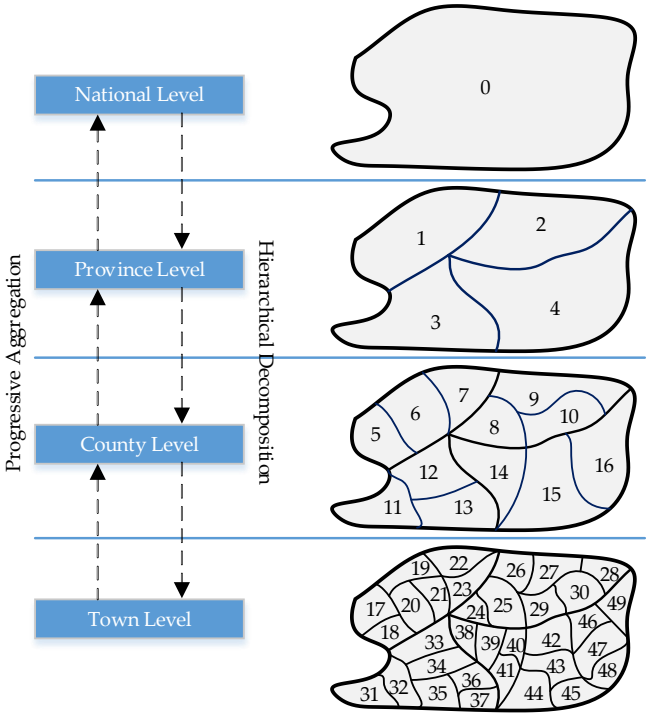

Figure 2. Administrative level decomposition in China (The numbers in these areas are unique identifiers).

From the top down, the country can be decomposed into some lower-level areas hierarchically. Meanwhile, from the bottom up, the transition matrix can be built level by level. Herein, each administrative district at the same level can be viewed as an independent partition of the country, and moreover, they can be processed simultaneously. However, different divisions usually have different areas, different numbers of land parcels, different numbers of polygon vertices, etc., which can be used to evaluate the time complexity of an algorithm or procedure and thus contribute to measuring the computational load for each of the of involved procedures. As the computational load may vary significantly in different procedures, a better strategy for load measuring requires referencing both the data and relevant operations in a combined way.

In fact, the administrative division is for the purpose of administration, and most countries in the world have three or more levels. Therefore, this initial decomposition can also be used in managing the data for other countries.

2.2.2. Computational Load Evaluation and Representation

Under the administrative level decomposition, each district in the hierarchical level involves a certain amount of computational load. How to quantitatively measure the load is a primary step for grouping the districts and scheduling the procedures. Among the five procedures in the LUCC analysis in Figure 1, the first four procedures are the most time consuming due to heavy loads in computations or I/O operations, and are thus performed in distributed workstations. Theoretically, the time usage

in data moving and area calculation should present a linear correlation with the data volume to be processed, i.e., the number of polygons or the constituting vertices. Both the clipping and overlay are essentially intersection operations. According to Becker, et al. [74], the optimal time complexity of the overlay operation on two sets of polygons is $O\ (N \times \log N + K)$, where N is the number of line segments, and K is the number of line intersections. When K is significantly smaller than N, a normal selection for measuring the computational loads is the number of vertices composing the polygons. In addition, I/O consumption is another factor that cannot be ignored, which is an impediment to the scalable speedups when parallelizing the overlay operations in a shared file system [75–77]. It is noteworthy that the jitter of the I/O process is unavoidable, which is often caused by the resource contention within multicore architectures, the communication, the kernel process scheduling and the cross-application contention [78].

Wang and Armstrong [73] proposed a theoretical approach which transforms the field-based and object-based spatial dataset to a spatial computational domain and then uses a composite function to represent the I/O consumption and computing time as compu-bands.

$$f = df + of \tag{1}$$

where df is a data-centric function, which transforms knowledge about spatial data characteristics in the context of spatial operations into either memory or I/O band. of is an operation-centric function, which takes into consideration the spatial operations that are directly or indirectly related to spatial data characteristics and transforms knowledge about the characteristics of spatial operations into a computing time band.

For the LUCC analysis, multiple procedures can be identified. Herein, we concentrate on how to finish the entire workflow as soon as possible, and the first four procedures are represented as four transforming functions. $f_1(p)$ is a typical I/O operation, and its time usage is mainly determined by the data volume of a data partition p, which can be measured by the number of polygons, the number of geometry vertices, or the areas. $f_2(p)$ and $f_3(p)$ measure the computing time consumed in clipping and overlaying operations, and they have a similar measurement for the time complexity [79]. $f_4(p)$ measures the area calculation, and the number of polygon vertices is naturally used as the main influencing factor.

$$f_{1,2,3,4}(p) \propto \{L_O(p), L_V(p), L_A(p), \ldots\} \tag{2}$$

where $L_O(p)$ is the number of polygons of p, $L_V(p)$ is the number of polygon vertices of p, and $L_A(p)$ is the area of p. Moreover, the Pearson correlation coefficient (PCC) is used to measure the linear correlation between the possible variables and the time usage, and then determine which factor should be selected to measure the loads. Finally, a compressive measurement is defined based on these functions.

$$T = T_{P1}(f_1) + T_{P2}(f_2) + T_{P3}(f_3) + T_{P4}(f_4) \tag{3}$$

where T_{P1}, T_{P2}, T_{P3} and T_{P4} are fitting functions used to convert the values calculated from (2) to computing time units (e.g., seconds).

During the entire workflow, the computational loads result from both the inner part of a district and the neighboring districts. Specifically, the inner loads are closely related to the inner part of the data and the operations performed on it, while the outer loads are closely related to the neighboring divisions due to the clipping operations. According to Kang and Lin [80], each district can be represented as a graph vertex with weights (i.e., the load produced from P_1 to P_4), and the relations among the districts can be represented as graph edges with weights (i.e., the load in clipping neighboring districts). After this representation, a graph that completely illustrates the data distribution and the intensity of the computational loads in the LUCC analysis can be obtained. This graph can be formally modeled by the following formula:

$$\mathrm{Graph}(LUCC) = \mathrm{G}(\mathrm{V}[i, f_v, w_v], \mathrm{E}[j, f_w, w_e]) \tag{4}$$

where V is a graph vertex set, and E is a graph edge set. Each vertex in V contains three elements: i (the identifier for the vertex), f_v (the vertex function) and w_v (the vertex weight). f_v denotes the specific operation upon a particular object, and w_v denotes computational load of the function on that object. Similarly, each edge in E contains three elements: j (the identifier for the edge), f_e (the edge function), and w_e (the edge weight). f_e is used to merge these results from f_v, and w_e denotes the computational load of the edge function on the spatial relation between two objects. In fact, f_e is optional and can be empty if output dependence does not exist. w_v and w_e may vary greatly for different vertices and edges due to the spatial heterogeneity. For more details about the graph representation, please refer to the definition of data dependence in spatial operations [80], which also provides an operable means for generating the graph vertices and edges.

Figure 3 illustrates a sample instance by simulating seven districts, which are all represented as graph vertices. The neighboring relations are represented as graph edges. The background grids are the originally produced chunks, each of which is filled by many land class parcels. In addition, the value in each grid is the corresponding load value, and the load of a district can be obtained by summing load values from these topologically contained or intersected grids. As a matter of fact, the shapes of these grids are often irregular, and they provide complete coverage of the entire area.

In Figure 3, the weights both exist on the graph vertices and the edges. As shown in Table 2, the weight on each vertex is obtained by summing the loads of the related grids, while the weight on the edges is obtained by summing the loads of the crossing grids that intersect with the neighboring boundaries.

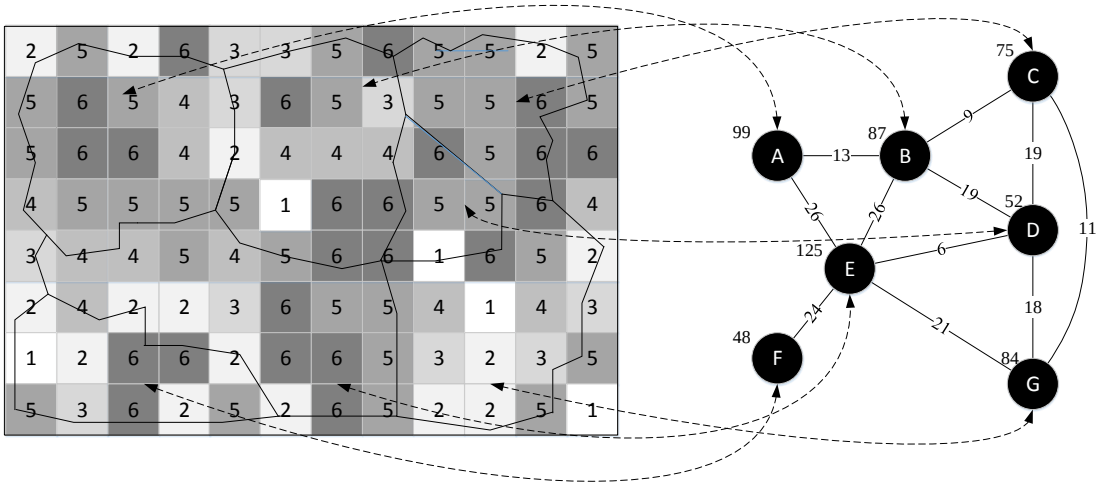

Figure 3. Graph representation of the computational loads over administrative districts (The value in each grid is the corresponding load).

Table 2. Graph weight evaluation.

Type	Vertex ID	Weights
Vertex	A	$W_A = 2+5+2+6+3+5+6+5+4+3+5+6+6+4+2+4+5+5+5+5+3+4+4 = 99$
	B	$W_B = 3+3+5+6+3+6+5+3+2+4+4+4+5+1+6+6+4+5+6+6 = 87$
	C	$W_C = 6+5+5+2+5+3+5+5+6+5+6+5+6+5+6 = 75$
	D	$W_D = 3+5+4+6+5+6+5+5+6+1+6 = 52$
	E	$W_E = 5+5+5+3+4+4+5+4+5+6+6+2+4+2+2+3+6+5+5+6+6+2+6+6+5+2+6+5 = 125$
	F	$W_F = 2+4+2+1+2+6+6+2+5+3+6+2+5+2 = 48$
	G	$W_G = 5+6+4+6+1+6+5+2+5+4+1+4+3+5+3+2+3+5+5+2+2+5 = 84$
Edges	A	$W_{AB} = W_{BA} = 3+3+2+5 = 13$; $W_{AE} = W_{EA} = 3+4+4+5+5+5 = 26$
	B	$W_{BC} = W_{CB} = 6+3 = 9$; $W_{BD} = W_{DB} = 3+4+6+6 = 19$; $W_{BE} = W_{EB} = 5+4+5+6+6 = 26$
	C	$W_{CD} = W_{DC} = 3+6+5+5 = 19$; $W_{CG} = W_{GC} = 5+6 = 11$
	D	$W_{DE} = W_{ED} = 6$; $W_{DG} = W_{GD} = 6+1+6+5 = 18$
	E	$W_{EF} = W_{FE} = 2+4+2+6+6+2+2 = 24$; $W_{EG} = W_{GE} = 6+5+5+5 = 21$

2.2.3. Graph Partitioning of the LUCC Analysis Graph

When the calculations are described as graphs in which the vertices and edges represent the computation and data dependence, they can be divided among the CPU cores of a parallel computer by partitioning the vertices of a graph [81]. In mathematics, the graph partitioning problem is defined in the form of a graph G = (V, E), with V vertices and E edges, such that it is possible to partition G into smaller components with specific properties [82]. Graph partitioning arises as a pre-processing step to divide and conquer algorithms, where it is often viewed as an effective way to divide things into roughly equal-sized pieces [83]. Partitioning a graph into equal-sized components while minimizing the number of edges between the different components can contribute to a balanced task assigning strategy in parallel computing [84,85]. Thus, the computational load can be divided among CPU cores. Moreover, some typical spatial operations have been optimized using the graph-based divide and conquer method by representing the geometry-level computation as vertices and dependence as edges, and then partitioning and encapsulating the sub-graphs into sub-tasks [80]. Specifically, each processing task over a spatial partition is viewed as a sub-task that may have a dependence on the other tasks over the neighboring spatial partitions.

After transforming the task of the LUCC analysis into a graph, it can thus be partitioned into several parts that can be simultaneously processed on different CPU cores. Two requirements should be met to complete the task efficiently, i.e., inter-independence and balanced load allocation. The inter-independence requires that each partitioned part can be processed independently without referencing the neighboring partitions in the computation, while balanced load allocation requires different partitions be processed in approximately equal time [73,86]. As geospatial science problems often have constraints in spatial and temporal dimensions (e.g., topological relations and order relations), relevant principles should be systematically considered in exploiting the parallelism toward optimizing the problems [86–89]. Two general rules can be concluded: (1) the independence among different parts from the physical phenomena are more intensive for those that are closer (localization); (2) the data access, computation and communication all contribute to the spatial distribution of the computational loads (heterogeneity). The localization enables the geographical domains to be divided and conquered, while the heterogeneity in the distribution of the computational loads requires that a reasonable load balancing strategy is adopted to partition the problem.

Many algorithms have been developed to find a reasonably good partition. Typical methods include spectral portioning methods [90], geometric partition algorithms [91–93], and the multilevel graph partitioning algorithms [94–97]. The multilevel graph partitioning algorithms work by applying one or more stages. Each stage reduces the size of the graph by collapsing the vertices and edges, partitioning the smaller graph, then mapping back and refining this partition of the original graph [97]. In many cases, this approach can provide both fast execution speed and very high-quality results. One widely used example of such an approach is called METIS [84], which was developed in the Karypis lab (http://glaros.dtc.umn.edu/gkhome/). METIS provides a set of programs for partitioning graphs, partitioning finite element meshes, and producing fill-reducing orderings for sparse matrices. The algorithms implemented in METIS are based on the multilevel recursive bisection, multilevel k-way, and multi-constraint partitioning schemes. Herein, the multilevel k-way partitioning scheme is selected as the partitioner in the proposed method. METIS was chosen because METIS's multilevel k-way partitioning algorithm provides the advantageous capabilities of the minimized resulting sub-domain connectivity graph, enforced contiguous partitions and minimized alternative objectives.

2.3. Architecture and Scheduling

2.3.1. Master-Slave Architecture for Distributed Computers

We implemented a system employing a dynamic load balancing strategy, as shown in Figure 4. The system enables a workflow to support load measuring on the master side, spatial decomposition on the master side, data moving from the master side to multiple slave sides, data clipping on the

slave sides, overlay operation on the slave sides, area calculation on the slave sides and the final matrix building on the master side.

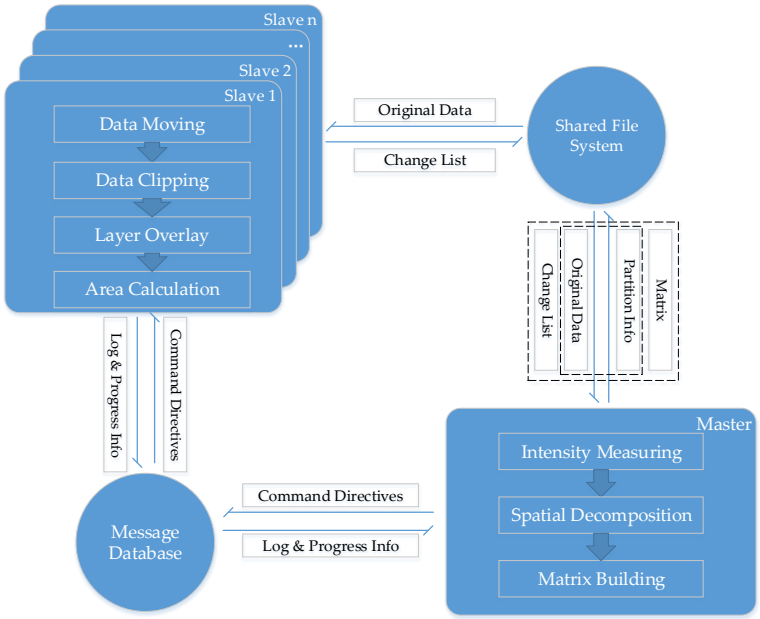

Figure 4. Master-slave architecture for distributed workstations.

Measuring the computational loads in the original chunks is performed on the server side. After this step, the LUCC analysis graph is constructed on the master side, and then partitioned to a certain number equal to that of the slaves. Next, the data partitions are moved from the master to the slave sides. Thus, data clipping, overlay operation and area calculation can be performed concurrently on each of the slave sides. Finally, the resultant data will be transferred to the master side.

2.3.2. Stream Scheduling of Tasks

With the above system, communications between any pair of workstations are supported. Furthermore, we adopted a stream scheduling method to exploit the parallelism among the slave workstations. Herein, each administrative division was viewed as an independent compute unit for scheduling, as illustrated in Figure 5.

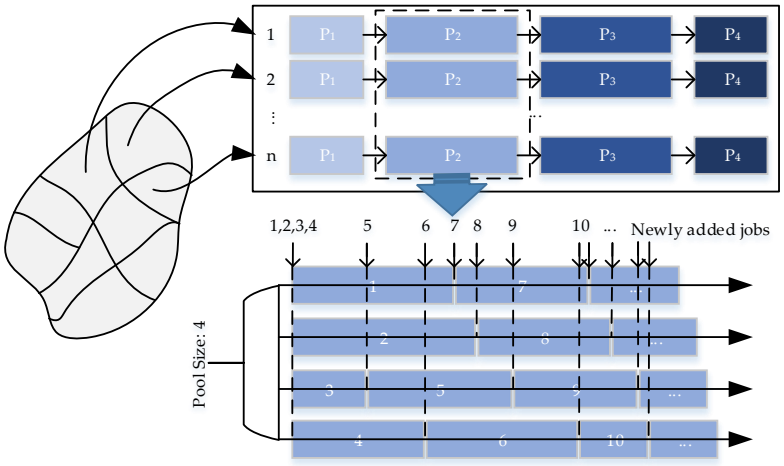

Figure 5. Stream scheduling of asynchronous compute units (P_1, P_2, P_3 and P_4 are four different procedures; P_1 is data moving, P_2 is data clipping, P_3 is overlay operation, and P_4 is area calculation).

Different administrative divisions often have different computational loads. The different computational loads mean the time usage for different compute units may vary greatly due to load differences. A process pool, whose size is set to the number of CPU cores on each of the slave workstations, is first initialized to support the computation. Moreover, in the pool, different units are scheduled asynchronously. As illustrated in Figure 5, four compute units were initiated at startup. When a unit finished, a new one was scheduled to push into the pool. When all the units finished, the whole task was terminated.

2.3.3. Implementation

On the side of each slave workstation, the stream scheduling was realized in the Python language and used to fully exploit the parallel performance in the process pool, which was supported in the package named Multiprocessing. A shared table was established on the server side to distribute the jobs to different workstations to work smoothly. At the same time, the working status of the workstations is continuously transferred to the server side.

Moreover, to support all the spatial operations from P_1 to P_3, we used the ArcGIS API for Python, which is a widely used Python library for working with maps and geospatial data. It provides simple and efficient tools for sophisticated vector and raster analysis, geocoding, map making, routing and directions. Furthermore, a series of error checks were developed to guarantee that all the procedures had correct inputs and outputs. Error checking enabled the workflow to work smoothly. For P_4, an area calculation was developed in the C++ language, and the File Geodatabase API provided by ESRI Company was used to access the spatial data. In comparison with the Python language, a very optimal performance could be obtained.

In addition, an MS-SQL database was used as the message database for exchanging the command directives, and the log and progress information between the master and the slave workstations.

3. Performance Evaluation

3.1. Testing Environment

A computer cluster was used to support the LUCC analysis, which included 15 workstations running Microsoft Windows 7 (×64) with a six-core Intel Xeon E5-1650 CPU (3.50 GHz) and 16 GB DDR4-2133. The total number of CPU cores was 90. In addition, Hyper-Threading Technology makes a single physical core appear as two logical cores, and the maximum number of threads for a workstation was 12. These workstations were connected through a ten-thousand-megabyte bandwidth switch.

3.2. Determination of the Factors in Estimation

To select the proper factors in measuring the computational loads, we used the PCC coefficient to measure the linear correlation between the possible factors and the time usage in each of the procedures. Values near 1 indicated that the curve shapes have high comparability. This is defined as follows.

$$\rho = \frac{\mathrm{cov}(X+Y)}{\sigma_X \sigma_Y} \tag{5}$$

where cov is the covariance, σ_X is the standard deviation of X, σ_Y is the standard deviation of Y. Herein, we randomly selected 89 counties in China, and the testing result is provided in the Supplementary Materials. Shown in Table 3 is the correlation analysis result obtained through calculating the PCC between the time usage in each of procedures and the potential influencing factors.

Table 3. Comparison of PCC values from different factors (The green items indicated higher correlations).

Factors\Time	T_{P1}	T_{P2}	T_{P3}	T_{P4}	T
F	0.969167	0.530038	0.568205	0.740030	0.582032
N	0.718055	0.973142	0.994402	0.964224	0.990629
A	0.513562	0.790036	0.825275	0.779288	0.809465
$N \times log(N)$	0.706571	0.976781	0.994454	0.960296	0.991878

Herein, F is the count of polygons in the layers, N is the number of vertices that constitute the polygons. According to Table 3, F was the most influencing factor that determined T_{P1} in data movement, and thus it was selected for measuring the loads in this procedure. As the land cover data were stored in vector objects, the movement of the data layers was realized through applying the query condition to each of the vector features. Even N should be a more effective parameter in theory for assessing the data volume, F showed a closer linear relationship with time T_{P1}. Thus, f_1 was set to F. Herein, a linear curve-fitting method was used to capture the trend of time with the changes in F. As illustrated in Figure 6, the time usage in this procedure was modeled using the following formula.

$$T_{P1} = 0.0001743 \times F + 14.395 \quad (6)$$

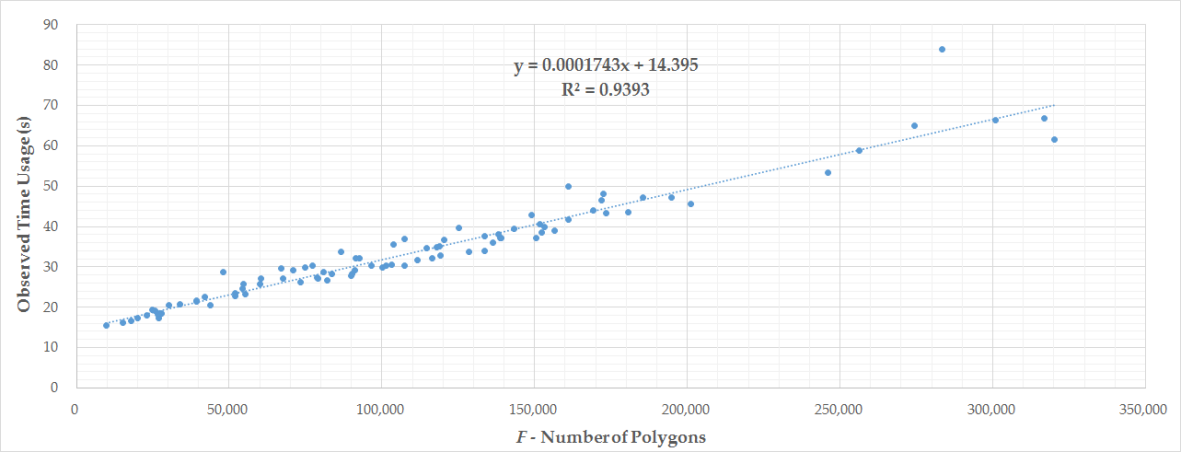

Figure 6. Linear regression of P_1 (This is the relation between the number of polygons and the time usage in data movement, where F is the number of polygons).

As the data clipping (P_2) and overlay (P_3) are essentially intersection operations, the time usage should be measured by the number of vertices. In P_2, the number of the vertices constituting the clipping boundaries was much less than that of the land cover polygons. Thus, factor N could be used in the time complexity format to measure the computational load, i.e., $f_2 = N \times log(N)$. In addition, clipping the two polygon layers involved more data I/O operations, and thus a linear fitting curve may not well match. In Figure 7, a second-order curve-fitting method was used to capture the trend of time with the change in the number of vertices.

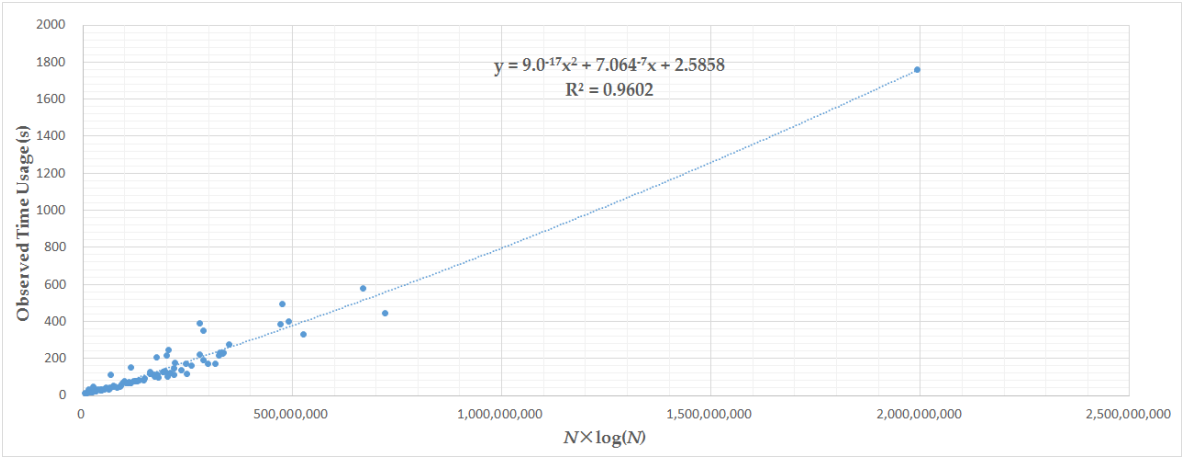

Figure 7. Second-order regression of P_2 (This is the relation between $N \times log(N)$ and the time usage in data clipping, where N is the number of vertices).

Then, we have

$$T_{P2} = 9.0^{-17} \times (N \times log(N))^2 + 0.0000007064 \times N \times log(N) + 2.5858 \quad (7)$$

Meanwhile, in P_3, the overlay of the different layers involved two polygon layers from different years. Therefore, the number of vertices from the two participating layers was used, i.e., $f_3 = N \times log(N)$. As the time usage was mainly determined by the computation, we found a linear fitting curve was more suitable than other curves, as illustrated in Figure 8.

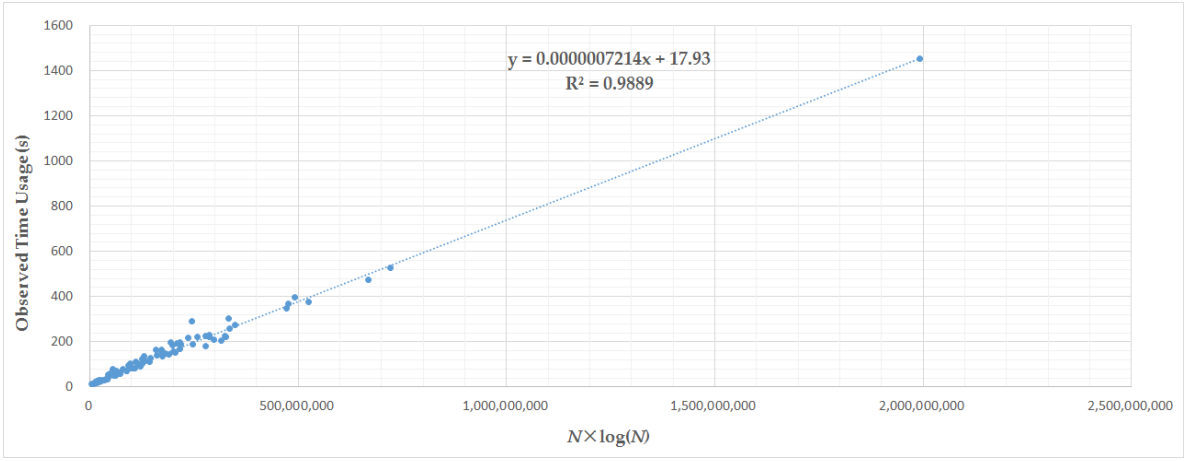

Figure 8. Linear order regression of P_3 (This is the relation between $N \times log(N)$ and the time usage in polygon overlay, where N is the number of vertices).

Then, we have

$$T_{P3} = 0.0000007214 \times N \times log(N) \quad (8)$$

In P_4, the computational load in the area calculation was positively proportional to the N, and a linear relation was most suitable, i.e., $of_4 = N$. The fitting curve was shown in Figure 9.

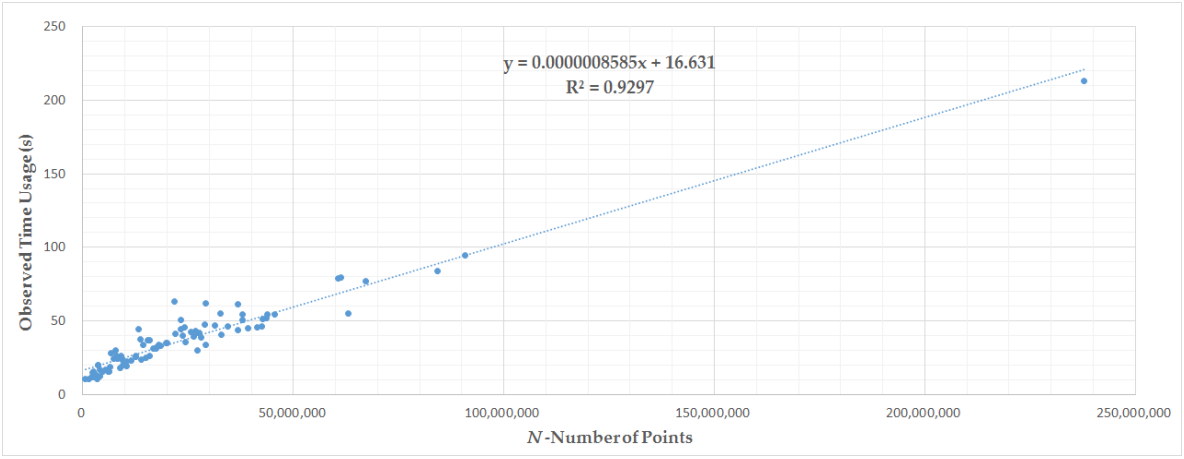

Figure 9. Linear regression of P_4 (This is the relation between N and the time usage in area calculation, where N is the number of vertices).

Then, we have

$$T_{P4} = 0.0000008585 \times N + 16.631 \tag{9}$$

Finally, the total time used in the four procedures can be measured as follows.

$$T = T_{P1} + T_{P2} + T_{P3} + T_{P4} \tag{10}$$

It is important to note that these time functions (T_{P1}, T_{P2}, T_{P3} and T_{P4}) were all obtained under a certain testing hardware environment, including the CPU, the physical memory, the disk, etc. Different hardware conditions, including the CPU cores, memory bandwidth, disk transfer rate, etc., may cause certain differences in these functions. After transforming the related factors into an estimable function of the time, the computational load can be measured according to the data size and the related procedures.

3.3. Experimental Results

3.3.1. Testing data

The testing data covered all 31 provinces and autonomous regions in mainland China (Hong Kong, Macao and Taiwan regions were not covered), and 2858 counties were involved. Specifically, the data were organized into 2858 chunks, each of which was stored in an individual File Geodatabase. Moreover, these data chunks were located in a shared-disk file system, and each computer could freely access any part of the data independently. According to the time functions proposed in Section 3.2, we estimated the computational loads for all these chunks. Shown in Figure 10 is the estimated time usage for all the counties. Clearly, there were obvious differences among these counties due to the differences in the number of geometries and number of vertices. It could be seen that the computation on some large counties cost more than two hours, while computation on some small counties cost less than one minute.

In general, larger counties often had higher loads, while for some areas with complicated land cover conditions, the time usage per square kilometer showed a large difference. In other words, the computational loads in different locations were different. As shown in Figure 11, it can be concluded that in eastern and southern China, the time density was relatively higher. In fact, these areas were usually fragmented, and the representative polygons were often narrow and twisted. Thus, more points were essential for constraining the boundaries of the land cover parcels.

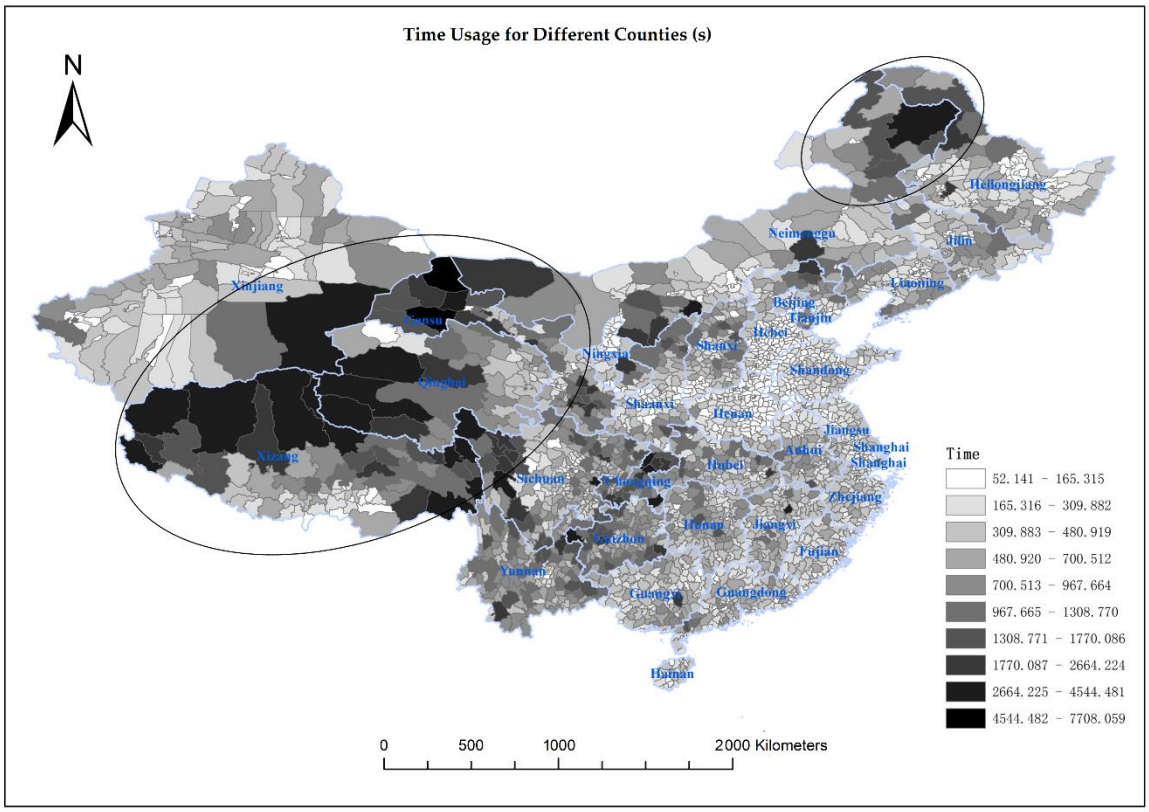

Figure 10. Estimated time usage in the 2858 counties in China.

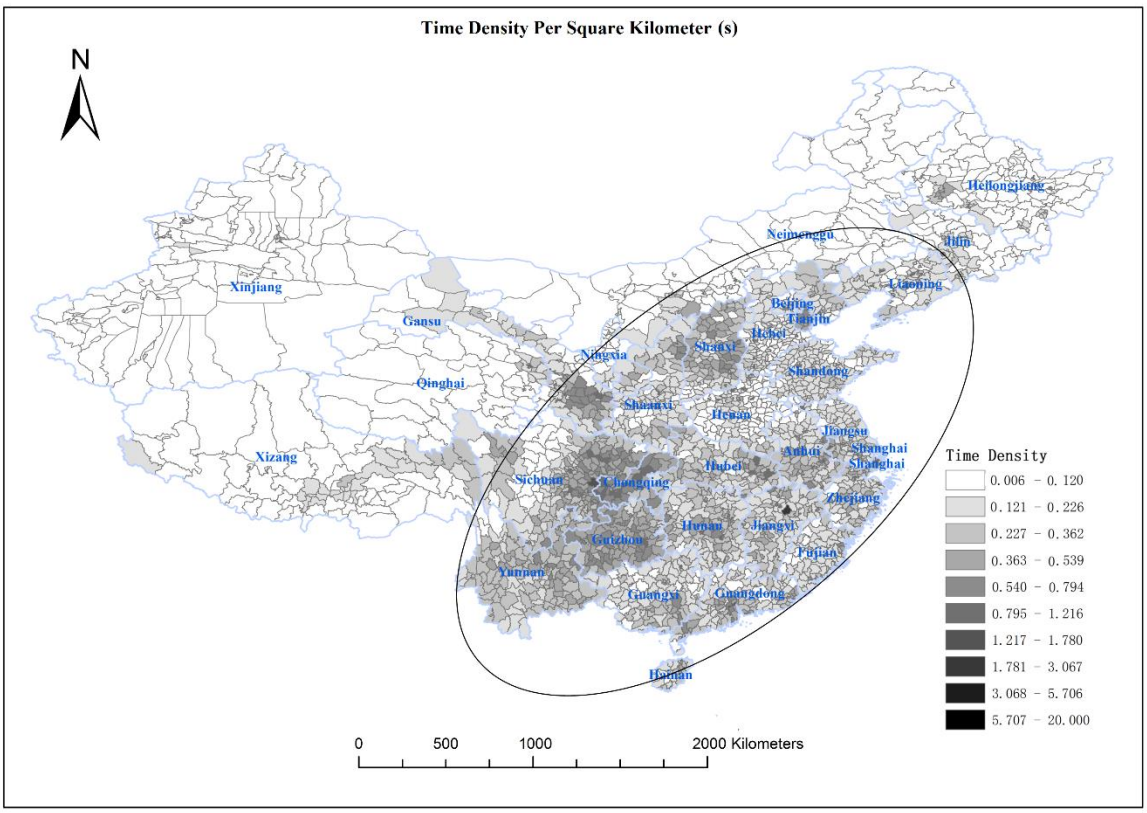

Figure 11. Estimated time density for the counties in China.

3.3.2. Results

Two individual experiments were conducted to verify the performance of the proposed method. The first one was to test the performance of the LUCC analysis over 7 provinces on a single workstation, which could validate the load measurement and the scalability of the program under a multi-core environment. The second one was performed in a cluster environment, which aimed at testing the validity of the spatial decomposition and the entire efficiency.

In the first experiment, Shanxi, Heilongjiang, Shanghai, Zhejiang, Shandong, Xizang and Qinghai were selected, and detailed testing results are provided in Table S3 in the Supplementary Materials. First, seven provinces were tested in a serial manner towards testing the time loads. Shown in Table 4 are the observed and estimated time usage.

Table 4. Comparison of the observed time and estimated time.

Order	Name	Number of Counties	Time Usage (s)		Difference	
			Observed	Estimated	Absolute	Ratio
1	Shanxi	117	54,158.173	51,664.532	2493.640	4.6%
2	Heilongjiang	128	39,550.255	37,914.468	1635.787	4.1%
3	Shanghai	16	2743.127	2529.132	213.995	7.8%
4	Zhejiang	90	28,432.039	21,489.041	6942.998	24.4%
5	Shandong	137	35,075.643	27,635.252	7440.391	21.2%
6	Xizang	74	74,659.131	77,058.161	2399.029	3.2%
7	Qinghai	46	32,129.256	34,644.451	2515.195	7.8%
-	Total/Avg	-	266,747.624	252,935.037	23,641.035	8.9%

The maximum difference ratio was approximately 24.4%, and the minimum difference ratio was 3.2%. On the whole, the average difference ratio was approximately 8.9%. Considering some unstable factors, such as the disk transfer rate and geometry complexity, a further improvement in accuracy for measuring the loads should be difficult to achieve. Actually, the absolute differences could be further reduced when performing the test in a multicore CPU. Furthermore, we tested the performance improvement when giving different computing processes (CPU cores), as illustrated in Table 5, and the performance curves are shown in Figure 12.

Table 5. Performance improvement in a multicore CPU environment (P is the number of processes).

Order	Name	Time (s)					Speedup
		P = 1	P = 2	P = 4	P = 6	P = 12	
1	Shanxi	54,158.173	29,733.984	18,042.815	12,955.973	8994.652	6.021
2	Heilongjiang	39,550.255	27,368.083	19,710.915	13,951.313	10,218.074	3.871
3	Shanghai	2743.127	1658.247	1083.171	877.641	681.561	4.025
4	Zhejiang	28,432.039	15,515.316	8185.116	6340.825	4595.972	6.186
5	Shandong	35,075.643	19,257.271	10,199.049	7279.027	5601.673	6.262
6	Xizang	74,659.131	43,597.761	26,888.392	18,922.136	15,557.339	4.799
7	Qinghai	32,129.256	18,202.262	12,600.972	9032.178	6538.687	4.914
-	Total/Avg	266,747.624	155,332.924	96,710.430	69,359.093	52,187.958	5.111

In the second experiment, the proposed method was tested on a cluster. As all the 2858 counties would be processed, spatial decomposition should be conducted first. The target of partitioning the task of the LUCC analysis was to distribute it to different slave workstations, and then coordinate them to complete the entire task. Therefore, a partitioning method that can both maintain the balancing loads and the smallest possible cost of cutting should be pursued. Before partitioning the data, the LUCC analysis graph was first constructed by estimating the loads in the focused counties and the neighboring counties. The loads in the counties focused upon contributed to the major computational

loads from P_1 to P_4, which were represented as graph vertices, while the loads in neighboring counties mainly contributed to the data moving load in P_1, which were represented as graph edges.

Thus, we obtained the LUCC graph that completely described the task. Before partitioning the graph, we stored the vertices and edges as a graph file by referencing the METIS soft manual (http://glaros.dtc.umn.edu/gkhome/views/metis). Moreover, to minimize the edge cut costs and the total communication volume, multilevel k-way partitioning was used. The LUCC graph was partitioned into 15 parts, as shown in Figure 13, to be in accordance with the testing environment.

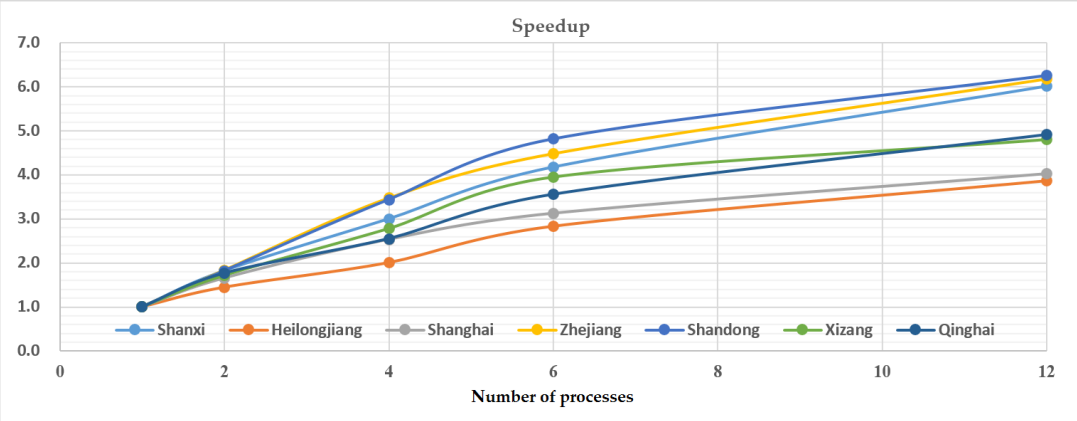

Figure 12. Performance improvement in different numbers of processes.

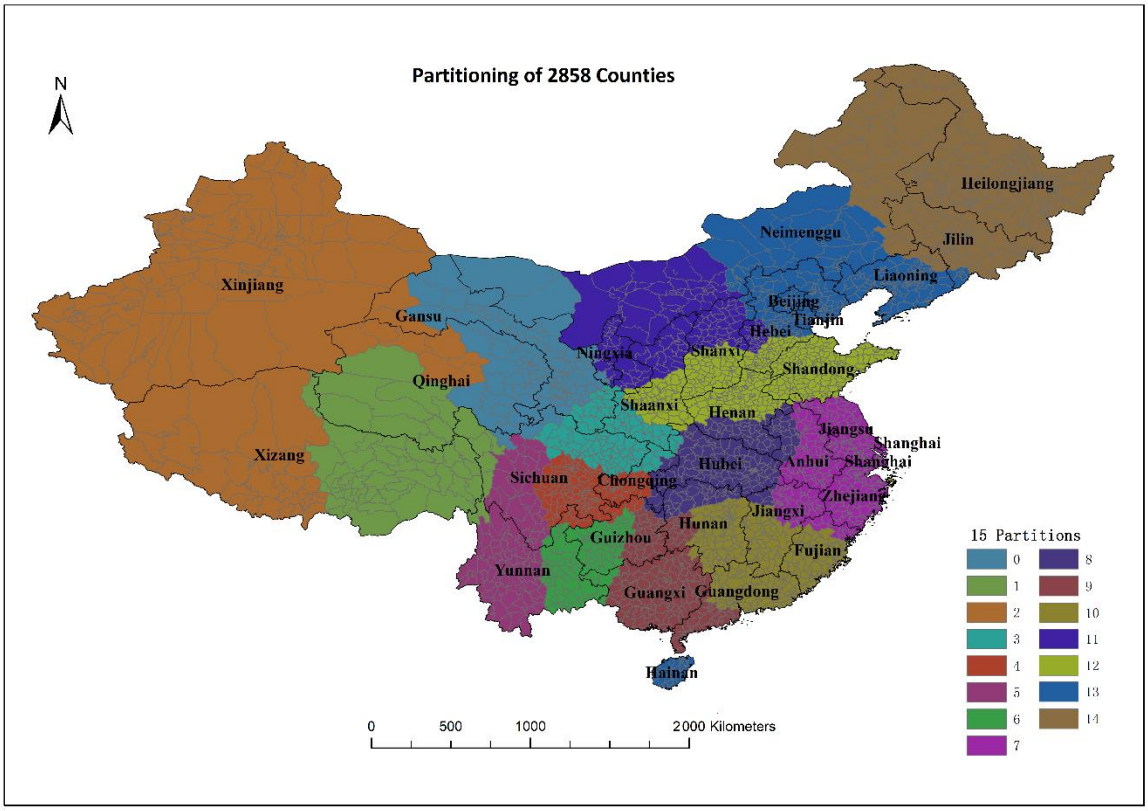

Figure 13. Partitioning of the LUCC analysis graph (the graph was partitioned into 15 parts, each of which was labeled with a single color).

After partitioning the graph, 15 sub-graphs were formed, and the related cutting edges were also recorded. As shown in Table 6, 15 sub-tasks were built. Then, we estimated the time loads in the sub-graphs and the cutting edges, respectively. Moreover, the total loads for each sub-task were obtained by summing the sub-graph loads and the cutting-edge loads.

Table 6. Computing time loads for the partitioning domains and total loads.

Computer ID	Graph Loads (s)	Cutting Loads (s)	Costs Ratio	Task Loads (s)	Balance
192.168.3.1	78,737.800	7834.684	9.95%	86,572.484	6.52%
192.168.3.2	78,945.698	2513.631	3.18%	81,459.329	6.13%
192.168.3.3	78,444.883	6984.443	8.90%	85,429.326	6.43%
192.168.3.4	83,135.226	13,525.892	16.27%	96,661.118	7.28%
192.168.3.5	82,848.954	8524.572	10.29%	91,373.526	6.88%
192.168.3.6	80,668.012	7148.928	8.86%	87,816.940	6.61%
192.168.3.7	83,146.467	6905.600	8.31%	90,052.067	6.78%
192.168.3.8	83,042.980	6721.402	8.09%	89,764.382	6.76%
192.168.3.9	82,401.041	11,897.392	14.44%	94,298.433	7.10%
192.168.3.10	80,064.289	11,565.392	14.45%	91,629.681	6.90%
192.168.3.11	79,672.611	12,012.495	15.08%	91,685.106	6.90%
192.168.3.12	78,525.364	2017.754	2.57%	80,543.118	6.06%
192.168.3.13	78,805.905	3550.061	4.50%	82,355.966	6.20%
192.168.3.14	80,616.409	5116.888	6.35%	85,733.297	6.45%
192.168.3.15	82,511.703	10,519.652	12.75%	93,031.355	7.00%

According to Table 6, the LUCC analysis task was decomposed into 15 sub-tasks. These tasks were distributed in a relatively even manner. In addition, extra costs caused by decomposition were approximately 10%. The proportions of the computational loads ranged from 6.06% to 7.28%.

Table 7 shows the results of the entire testing data. Obviously, a large proportion of the time was reduced on each of the slave workstations, and the average time load was approximately 5.049 h. The performance improvement ranged from 3.834 to 5.573, and the maximum gap in time was approximately 4790 s, with reference to the maximum task load on workstation 192.168.3.4 and the minimum task load on workstation 192.168.3.10.

Table 7. Performance improvement in a cluster (All the CPU cores were used).

Slave ID	Counties	Task Loads (s)	Observed Time (s)	Speedup
192.168.3.1	176	86,572.484	16,713.703	5.180
192.168.3.2	212	81,459.329	21,245.131	3.834
192.168.3.3	189	85,429.326	18,003.370	4.745
192.168.3.4	327	96,661.118	18,944.093	5.102
192.168.3.5	294	91,373.526	17,064.811	5.354
192.168.3.6	333	87,816.940	18,688.152	4.699
192.168.3.7	207	90,052.067	17,794.495	5.061
192.168.3.8	268	89,764.382	18,774.252	4.781
192.168.3.9	197	94,298.433	19,458.336	4.846
192.168.3.10	106	91,629.681	16,454.875	5.569
192.168.3.11	88	91,685.106	18,048.192	5.080
192.168.3.12	137	80,543.118	17,363.678	4.639
192.168.3.13	71	82,355.966	18,177.326	4.531
192.168.3.14	111	85,733.297	19,216.351	4.461
192.168.3.15	142	93,031.355	16,692.415	5.573
Avg	190.5	88,560.409	18,175.945	4.872

When the total task was performed on a single workstation without adopting any parallel optimization method, the estimated time was more than 14 days by summing all the estimated graph loads (i.e., the total task was performed on a single workstation, and no cutting loads were incurred).

Whereas, in a cluster with the proposed method, the total time usage was approximately 5.901 h considering the maximum task load. On the whole, the speedup was approximately 57.028 in the cluster in comparison with the estimated time on a single workstation.

4. Discussion

Generally, time complexity is simply a measure of the time it takes for a function or expression to complete its task, as well as the name of the process to measure that time. However, the hardware environment should also be considered. This is because different CPU frequencies, memory bandwidth and disk transfer rates will no doubt exert an influence on the time usage. Therefore, there exists a gap between time complexity and actual time usage. Through a polynomial regression analysis, we built the relevance between the actual time usage and the data volume factors, such as the number of geometries and vertices. Moreover, the estimated time usage from different procedures could be transformed into a unified function. According to the testing results in the Supplementary Materials, the PCC coefficient for the observed time and the estimated time was more than 0.99. Thus, the estimated time function can exactly reflect the actual time usage by referencing the number of geometry vertices or the number of polygons. According to the analysis of the testing data, spatial heterogeneity is a common phenomenon that is determined both by the data and the corresponding procedures. Herein, two individual experiments were provided. The first one was performed on a single workstation, while the second one was performed on a cluster.

The results from the first experiment indicated that an average speed improvement of 5.1 times could be achieved over the sequential version with 12 processes. When the number of allocating processes was fewer than 6, a stable improvement could be achieved. However, there was no continuous performance improvement when the allocating processes were more than 6. The main reason was due to the I/O bottlenecks. When multiple processes read the data from disk or dump data to the disk with conflicts, the overall performance can easily decline due to the I/O speed limit of the storage drive. In addition, multiple procedures that were working on different CPU cores may also have caused some bad effects on the computation speed when assessing the shared memory space. Therefore, to alleviate the competition from slow disks, a flexible method is to adopt a faster drive, such as a solid-state drive (SSD). If conditions allowed, loading all the data into the main memory could definitely provide an ideal way of conquering large-scale data processing. According to our early tests [79], using a streaming framework that can coordinate the I/O events with the computation may also provide an effective way of alleviating this bottleneck. Thus, an individual procedure should be allocated to continuously reading the data and producing a continuous data block stream; then, other procedures can concurrently process these available data blocks acquired from the stream.

When performing the task on a single workstation, we focused on how to fully exploit the parallelism from multicore CPUs. In a cluster environment, the difficulty lies in how to allocate the computational loads to different workstations. Therefore, a quantitative representation of the task should first be accomplished. According to the results from the second experiment, the average difference ratio of the partitioned loads maintained a relatively low level, which demonstrated that the estimation and the partitioning method could be used to decompose the computational loads. Furthermore, two temporal land cover layers covering mainland China were decomposed into multiple sub-datasets with even loads. Thus, this extremely time-consuming task was solved in a very short period of time.

In addition, two improvements should also be made in the near future. One regards the lack of assessment on how the geometry complexity influences the time consumption. Though time functions have given an approximate estimation of the observed time, some deep rules might have been covered up. Another improvement is in terms of applicability. All the tests were performed in a homogeneous environment, and all the workstations had an identical configuration. Actually, heterogeneous environments are very common, and some adjustments should be made.

As a matter of fact, the earth's surface is changing at an amazing speed, and thus, more and more large-scale, short-period and high-resolution data are becoming available with the development of sensor technology. The proposed method was only used to achieve a two-temporal-layer LUCC analysis, and no geomorphic layers were involved for a deep-level analysis. When dealing with more temporal layers and thematic layers, the time costs will definitely increase sharply. Therefore, methods for how to perform the change analysis for a large-scale region, such as a country, a continent or even the globe, especially using high-performance computing facilities, is still a challenge that should be addressed.

5. Conclusions

This work presented a novel graph-based spatial decomposition to conquer the very heavy computational load when overlaying multi-temporal land cover data. This approach quantitatively represented the computational loads as graph vertices and edges, and balanced subtasks were formed through graph partitioning. Basically, PCC analysis was first conducted towards selecting the determining factors that influence the time usage. Then, stream scheduling was used to achieve an efficient parallel LUCC method. Together with a master-slave architecture connecting the distributed workstations and a stream scheduling method, we performed two experiments for testing the performance improvements on a single workstation and a cluster.

Clearly, selecting the proper factors and regression methods can contribute to a unified representation that quantitatively transforms each of the procedures in a complex workflow to a time function with the data size as measurable variables. On this basis, graph-based spatial decomposition gives a balanced load allocation method, which not only considers the volume of the data to be processed, but also the spatial operations performed on the data. As a main research object, the LUCC analysis is thus optimized efficiently. It is worth mentioning that the consumed time was reduced from a two-week period to less than 6 h, for detecting the two-temporal changes over China.

In the near future, research should also be done to validate the feasibility of approach to raster maps and explore the potential influence of the geometric complexity that may also governs the parallel efficiency to some degree. In addition, a heterogeneous environment should also be considered.

Supplementary Materials: The following are available online at http://www.mdpi.com/2220-9964/7/7/273/s1, Table S1: Factor-Selecting, Table S2: Factor-Assessment, Table S3: Performance-Single, Table S4: Performance-Cluster, Table S5: Classification-Landcover.

Author Contributions: X.K. conceived the original idea and designed the experiments; X.K., J.L. and C.D. performed and analyzed the experiments; X.K. and S.X. wrote the paper. All authors read and approved the final manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (Grant No. 41701461), the National Key Research and Development Program of China (Grant No. 2016YFC0803101) and Basic Research Fund of CASM (Grant No. 7771701).

Acknowledgments: We thank the editors and the anonymous reviewers for their insightful comments which have helped to improve the quality of the paper.

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

References

1. Reid, R.S.; Kruska, R.L.; Muthui, N.; Taye, A.; Wotton, S.; Wilson, C.J.; Mulatu, W. Land-use and land-cover dynamics in response to changes in climatic, biological and socio-political forces: The case of Southwestern Ethiopia. *Landsc. Ecol.* **2000**, *15*, 339–355. [CrossRef]
2. Lambin, E.F.; Turner, B.L.; Geist, H.J.; Agbola, S.B.; Angelsen, A.; Bruce, J.W.; Coomes, O.T.; Dirzo, R.; Fischer, G.; Folke, C. The causes of land-use and land-cover change: Moving beyond the myths. *Glob. Environ. Chang.* **2001**, *11*, 261–269. [CrossRef]
3. Bürgi, M.; Hersperger, A.M.; Schneeberger, N. Driving forces of landscape change-current and new directions. *Landsc. Ecol.* **2005**, *19*, 857–868. [CrossRef]

4. Zhang, J.; Zhang, Y. Remote sensing research issues of the national land use change program of China. *ISPRS J. Photogramm. Remote Sens.* **2007**, *62*, 461–472. [CrossRef]
5. Yin, H.; Pflugmacher, D.; Li, A.; Li, Z.; Hostert, P. Land use and land cover change in Inner Mongolia-understanding the effects of China's re-vegetation programs. *Remote Sens. Environ.* **2018**, *204*, 918–930. [CrossRef]
6. Liu, J.; Hull, V.; Batistella, M.; DeFries, R.; Dietz, T.; Fu, F.; Hertel, T.; Izaurralde, R.C.; Lambin, E.; Li, S. Framing sustainability in a telecoupled world. *Ecol. Soc.* **2013**, *18*. [CrossRef]
7. Eakin, H.; Defries, R.; Kerr, S.; Lambin, E.F.; Liu, J.; Marcotullio, P.J.; Messerli, P.; Reenberg, A.; Rueda, X.; Swaffield, S.R. Significance of telecoupling for exploration of land-use change. In *Rethinking Global Land Use in an Urban Era*; Seto, K.C., Reenberg, A., Eds.; MIT Press: Cambridge, MA, USA, 2014; pp. 141–162.
8. Liu, J. Forest sustainability in China and implications for a telecoupled world. *Asia Pac. Policy Stud.* **2014**, *1*, 230–250. [CrossRef]
9. Friis, C.; Nielsen, J.O. Exploring the potential of the telecoupling framework for understanding land change. *IRI THESys Discuss. Pap.* **2014**, *1*, 1–32.
10. Liu, J.; Hull, V.; Luo, J.; Yang, W.; Liu, W.; Viña, A.; Vogt, C.; Xu, Z.; Yang, H.; Zhang, J. Multiple telecouplings and their complex interrelationships. *Ecol. Soc.* **2015**, *20*. [CrossRef]
11. Liu, J.; Deng, X. Progress of the research methodologies on the temporal and spatial process of LUCC. *Chin. Sci. Bull.* **2010**, *55*, 1354–1362. [CrossRef]
12. Liu, J.; Mooney, H.; Hull, V.; Davis, S.J.; Gaskell, J.; Hertel, T.; Lubchenco, J.; Seto, K.C.; Gleick, P.; Kremen, C. Systems integration for global sustainability. *Science* **2015**, *347*, 1258832. [CrossRef] [PubMed]
13. Liu, J.; Zhang, Z.; Xu, X.; Kuang, W.; Zhou, W.; Zhang, S.; Li, R.; Yan, C.; Yu, D.; Wu, S. Spatial patterns and driving forces of land use change in China during the early 21st century. *J. Geogr. Sci.* **2010**, *20*, 483–494. [CrossRef]
14. Plieninger, T.; Draux, H.; Fagerholm, N.; Bieling, C.; Bürgi, M.; Kizos, T.; Kuemmerle, T.; Primdahl, J.; Verburg, P.H. The driving forces of landscape change in Europe: A systematic review of the evidence. *Land Use Policy* **2016**, *57*, 204–214. [CrossRef]
15. Lunetta, R.S.; Knight, J.F.; Ediriwickrema, J.; Lyon, J.G.; Worthy, L.D. Land-cover change detection using multi-temporal modis ndvi data. *Remote Sens. Environ.* **2006**, *105*, 142–154. [CrossRef]
16. Wu, W.; Yu, Q.; You, L.; Chen, K.; Tang, H.; Liu, J. Global cropping intensity gaps: Increasing food production without cropland expansion. *Land Use Policy* **2018**, *76*, 515–525. [CrossRef]
17. Yang, C.; Yu, M.; Hu, F.; Jiang, Y.; Li, Y. Utilizing cloud computing to address big geospatial data challenges. *Comput. Environ. Urban Syst.* **2016**, *61*, 120–128. [CrossRef]
18. Liu, J.; Dou, Y.; Batistella, M.; Challies, E.; Connor, T.; Friis, C.; Millington, J.D.; Parish, E.; Romulo, C.L.; Silva, R.F.B. Spillover systems in a telecoupled anthropocene: Typology, methods, and governance for global sustainability. *Curr. Opin. Environ. Sustain.* **2018**, *33*, 58–69. [CrossRef]
19. Li, X. A review of the international researches on land use/land cover change. *Acta Geogr. Sin.* **1996**, *51*, 553–558.
20. Chen, J.; Gong, P.; He, C.; Pu, R.; Shi, P. Land-use/land-cover change detection using improved change-vector analysis. *Photogramm. Eng. Remote Sens.* **2003**, *69*, 369–379. [CrossRef]
21. Xiao, J.; Shen, Y.; Ge, J.; Tateishi, R.; Tang, C.; Liang, Y.; Huang, Z. Evaluating urban expansion and land use change in Shijiazhuang, China, by using gis and remote sensing. *Landsc. Urban Plan.* **2006**, *75*, 69–80. [CrossRef]
22. Shalaby, A.; Tateishi, R. Remote sensing and gis for mapping and monitoring land cover and land-use changes in the Northwestern coastal zone of Egypt. *Appl. Geogr.* **2007**, *27*, 28–41. [CrossRef]
23. Parker, D.C.; Manson, S.M.; Janssen, M.A.; Hoffmann, M.J.; Deadman, P. Multi-agent systems for the simulation of land-use and land-cover change: A review. *Ann. Assoc. Am. Geogr.* **2003**, *93*, 314–337. [CrossRef]
24. Gibon, A.; Sheeren, D.; Monteil, C.; Ladet, S.; Balent, G. Modelling and simulating change in reforesting mountain landscapes using a social-ecological framework. *Landsc. Ecol.* **2010**, *25*, 267–285. [CrossRef]
25. Ralha, C.G.; Abreu, C.G.; Coelho, C.G.; Zaghetto, A.; Macchiavello, B.; Machado, R.B. A multi-agent model system for land-use change simulation. *Environ. Model. Softw.* **2013**, *42*, 30–46. [CrossRef]

26. Liu, X.; Liang, X.; Li, X.; Xu, X.; Ou, J.; Chen, Y.; Li, S.; Wang, S.; Pei, F. A future land use simulation model (flus) for simulating multiple land use scenarios by coupling human and natural effects. *Landsc. Urban Plan.* **2017**, *168*, 94–116. [CrossRef]
27. Walker, R.; Solecki, W. Theorizing land-cover and land-use change: The case of the Florida everglades and its degradation. *Ann. Assoc. Am. Geogr.* **2004**, *94*, 311–328. [CrossRef]
28. Rindfuss, R.R.; Entwisle, B.; Walsh, S.J.; Mena, C.F.; Erlien, C.M.; Gray, C.L. Frontier land use change: Synthesis, challenges, and next steps. *Ann. Assoc. Am. Geogr.* **2007**, *97*, 739–754. [CrossRef]
29. Dong, L.; Wang, W.; Ma, M.; Kong, J.; Veroustraete, F. The change of land cover and land use and its impact factors in upriver key regions of the yellow river. *Int. J. Remote Sens.* **2009**, *30*, 1251–1265. [CrossRef]
30. Wang, K.; Zhang, Q.; Chen, Y.D.; Singh, V.P. Effects of land-use/cover change on hydrological processes using a GIS/RS-based integrated hydrological model: Case study of the east river, China. *Hydrol. Sci. J.* **2015**, *60*, 1724–1738. [CrossRef]
31. Meneses, B.; Reis, R.; Vale, M.; Saraiva, R. Land use and land cover changes in zêzere watershed (portugal)—Water quality implications. *Sci. Total Environ.* **2015**, *527*, 439–447. [CrossRef] [PubMed]
32. Yu, W.; Zang, S.; Wu, C.; Liu, W.; Na, X. Analyzing and modeling land use land cover change (LUCC) in the daqing city, China. *Appl. Geogr.* **2011**, *31*, 600–608. [CrossRef]
33. Turner, B.; Meyer, W.B.; Skole, D.L. Global land-use/land-cover change: Towards an integrated study. *Ambio Stockholm.* **1994**, *23*, 91–95.
34. Bartholomé, E.; Belward, A.S. Glc2000: A new approach to global land cover mapping from earth observation data. *Int. J. Remote Sens.* **2005**, *26*, 1959–1977. [CrossRef]
35. Verburg, P.H.; Crossman, N.; Ellis, E.C.; Heinimann, A.; Hostert, P.; Mertz, O.; Nagendra, H.; Sikor, T.; Erb, K.-H.; Golubiewski, N. Land system science and sustainable development of the earth system: A global land project perspective. *Anthropocene* **2015**, *12*, 29–41. [CrossRef]
36. Ringeval, B.; Augusto, L.; Monod, H.; Apeldoorn, D.; Bouwman, L.; Yang, X.; Achat, D.L.; Chini, L.P.; Van Oost, K.; Guenet, B. Phosphorus in agricultural soils: Drivers of its distribution at the global scale. *Glob. Chang. Biol.* **2017**, *23*, 3418–3432. [CrossRef] [PubMed]
37. Li, X.; Chen, G.; Liu, X.; Liang, X.; Wang, S.; Chen, Y.; Pei, F.; Xu, X. A new global land-use and land-cover change product at a 1-km resolution for 2010 to 2100 based on human–environment interactions. *Ann. Assoc. Am. Geogr.* **2017**, *107*, 1040–1059. [CrossRef]
38. Munroe, D.K.; Müller, D. Issues in spatially explicit statistical land-use/cover change (LUCC) models: Examples from western honduras and the central highlands of Vietnam. *Land Use Policy* **2007**, *24*, 521–530. [CrossRef]
39. Defries, R.S.; Chan, J.C.W. Multiple criteria for evaluating machine learning algorithms for land cover classification from satellite data. *Remote Sens. Environ.* **2000**, *74*, 503–515. [CrossRef]
40. Cracknell, M.J.; Reading, A.M. Geological mapping using remote sensing data: A comparison of five machine learning algorithms, their response to variations in the spatial distribution of training data and the use of explicit spatial information. *Comput. Geosci.* **2014**, *63*, 22–33. [CrossRef]
41. Jensen, J.R. *Introductory Digital Image Processing: A Remote Sensing Perspective*, 3rd ed.; Prentice Hall: Upper Saddle River, NJ, USA, 2005.
42. Lu, D.; Batistella, M.; Moran, E. Multitemporal spectral mixture analysis for amazonian land-cover change detection. *Can. J. Remote Sens.* **2004**, *30*, 87–100. [CrossRef]
43. Coppin, P.; Jonckheere, I.; Nackaerts, K.; Muys, B.; Lambin, E. Review articledigital change detection methods in ecosystem monitoring: A review. *Int. J. Remote Sens.* **2004**, *25*, 1565–1596. [CrossRef]
44. Weng, Q. Land use change analysis in the Zhujiang delta of China using satellite remote sensing, GIS and stochastic modelling. *J. Environ. Manag.* **2002**, *64*, 273–284. [CrossRef]
45. Mas, J.-F. Change estimates by map comparison: A method to reduce erroneous changes due to positional error. *Trans. GIS* **2005**, *9*, 619–629. [CrossRef]
46. Zhang, H.; Shu, N. Land use/cover change detection using feature database based on vector-image data conflation. In Proceedings of the 8th International Symposium on Spatial Accuracy Assessment in Natural Resources and Environmental Sciences, Shanghai, China, 25–27 June 2008; pp. 255–262.
47. Lu, Y.; Cao, M.; Zhang, L. A vector-based cellular automata model for simulating urban land use change. *Chin. Geogr. Sci.* **2015**, *25*, 74–84. [CrossRef]

48. Rozenstein, O.; Karnieli, A. Comparison of methods for land-use classification incorporating remote sensing and GIS inputs. *Appl. Geogr.* **2011**, *31*, 533–544. [CrossRef]
49. Khatami, R.; Mountrakis, G.; Stehman, S.V. A meta-analysis of remote sensing research on supervised pixel-based land-cover image classification processes: General guidelines for practitioners and future research. *Remote Sens. Environ.* **2016**, *177*, 89–100. [CrossRef]
50. Cheng, G.; Han, J.; Guo, L.; Liu, Z.; Bu, S.; Ren, J. Effective and efficient midlevel visual elements-oriented land-use classification using VHR remote sensing images. *IEEE Trans. Geosci. Remote Sens.* **2015**, *53*, 4238–4249. [CrossRef]
51. Huang, X.; Lu, Q.; Zhang, L. A multi-index learning approach for classification of high-resolution remotely sensed images over urban areas. *ISPRS J. Photogramm. Remote Sens.* **2014**, *90*, 36–48. [CrossRef]
52. Hu, F.; Xia, G.-S.; Hu, J.; Zhang, L. Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery. *Remote Sens.* **2015**, *7*, 14680–14707. [CrossRef]
53. Tewkesbury, A.P.; Comber, A.J.; Tate, N.J.; Lamb, A.; Fisher, P.F. A critical synthesis of remotely sensed optical image change detection techniques. *Remote Sens. Environ.* **2015**, *160*, 1–14. [CrossRef]
54. Ahlqvist, O. Extending post-classification change detection using semantic similarity metrics to overcome class heterogeneity: A study of 1992 and 2001 us national land cover database changes. *Remote Sens. Environ.* **2008**, *112*, 1226–1241. [CrossRef]
55. Tiede, D. A new geospatial overlay method for the analysis and visualization of spatial change patterns using object-oriented data modeling concepts. *Cartogr. Geogr. Inf. Sci.* **2014**, *41*, 227–234. [CrossRef] [PubMed]
56. Yang, X. Satellite monitoring of urban spatial growth in the atlanta metropolitan area. *Photogramm. Eng. Remote Sens.* **2002**, *68*, 725–734.
57. Yuan, F.; Sawaya, K.E.; Loeffelholz, B.C.; Bauer, M.E. Land cover classification and change analysis of the twin cities (minnesota) metropolitan area by multitemporal landsat remote sensing. *Remote Sens. Environ.* **2005**, *98*, 317–328. [CrossRef]
58. Im, J.; Jensen, J.; Tullis, J. Object-based change detection using correlation image analysis and image segmentation. *Int. J. Remote Sens.* **2008**, *29*, 399–423. [CrossRef]
59. Zhou, W.; Huang, G.; Troy, A.; Cadenasso, M. Object-based land cover classification of shaded areas in high spatial resolution imagery of urban areas: A comparison study. *Remote Sens. Environ.* **2009**, *113*, 1769–1777. [CrossRef]
60. Dingle Robertson, L.; King, D.J. Comparison of pixel- and object-based classification in land cover change mapping. *Int. J. Remote Sens.* **2011**, *32*, 1505–1529. [CrossRef]
61. Myint, S.W.; Gober, P.; Brazel, A.; Grossman-Clarke, S.; Weng, Q. Per-pixel vs. Object-based classification of urban land cover extraction using high spatial resolution imagery. *Remote Sens. Environ.* **2011**, *115*, 1145–1161. [CrossRef]
62. Chrisman, N.R. *Exploring Geographic Information Systems*; Wiley: Hoboken, NJ, USA, 1997.
63. Deren, D.L.; Shao, Z. Reflections on issues in national geographical conditions monitoring. *Geomat. Inf. Sci. Wuhan Univ.* **2016**, *41*, 143–147.
64. Openshaw, S.; Alvanides, S. Applying geocomputation to the analysis of spatial distributions. In *Geographic Information Systems: Principles and Technical Issues*; Longley, P.A., Michael, M., Maguire, D.J., Rhind, D.W., Eds.; John Wiley and Sons Inc.: New York, NY, USA, 1999; Volume 1, pp. 267–282.
65. Wang, S.; Liu, Y. Teragrid giscience gateway: Bridging cyberinfrastructure and giscience. *Int. J. Geogr. Inf. Sci.* **2009**, *23*, 631–656. [CrossRef]
66. Yang, C.; Raskin, R.; Goodchild, M.; Gahegan, M. Geospatial cyberinfrastructure: Past, present and future. *Comput. Environ. Urban Syst.* **2010**, *34*, 264–277. [CrossRef]
67. Zhang, J. Towards personal high-performance geospatial computing (HPC-G): Perspectives and a case study. In Proceedings of the ACM SIGSPATIAL International Workshop on High Performance and Distributed Geographic Information Systems, San Jose, CA, USA, 2 November 2010; ACM: New York, NY, USA, 2010; pp. 3–10.
68. Lee, C.A.; Gasster, S.D.; Plaza, A.; Chang, C.I.; Huang, B. Recent developments in high performance computing for remote sensing: A review. *IEEE J. STARS* **2011**, *4*, 508–527. [CrossRef]
69. Plaza, A.; Benediktsson, J.A.; Boardman, J.W.; Brazile, J.; Bruzzone, L.; Camps-Valls, G.; Chanussot, J.; Fauvel, M.; Gamba, P.; Gualtieri, A. Recent advances in techniques for hyperspectral image processing. *Remote Sens. Environ.* **2009**, *113*, S110–S122. [CrossRef]

70. Jin, X.; Sieber, R.E. A land use/land cover change geospatial cyberinfrastructure to integrate big data and temporal topology. *Int. J. Geogr. Inf. Sci.* **2016**, *30*, 573–593.
71. Pijanowski, B.C.; Tayyebi, A.; Doucette, J.; Pekin, B.K.; Braun, D.; Plourde, J. A big data urban growth simulation at a national scale: Configuring the gis and neural network based land transformation model to run in a high performance computing (HPC) environment. *Environ. Model. Softw.* **2014**, *51*, 250–268. [CrossRef]
72. Wang, S.; Armstrong, M.P. A Theory of the Spatial Computational Domain. In Proceedings of the 8th International Conference on GeoComputation University of Michigan, Ann Arbor, MI, USA, 31 July–3 August 2005.
73. Wang, S.; Armstrong, M.P. A theoretical approach to the use of cyberinfrastructure in geographical analysis. *Int. J. Geogr. Inf. Sci.* **2009**, *23*, 169–193. [CrossRef]
74. Becker, L.; Giesen, A.; Hinrichs, K.H.; Vahrenhold, J. Algorithms for performing polygonal map overlay and spatial join on massive data sets. In Proceedings of the International Symposium on Spatial Databases, Hong Kong, China, 20–23 July 1999; Springer: Berlin/Heidelberg, Germany, 1999; pp. 270–285.
75. 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.
76. Zhou, C.; Chen, Z.; Liu, Y.; Li, F.; Cheng, L.; Zhu, A.-X.; Li, M. Data decomposition method for parallel polygon rasterization considering load balancing. *Comput. Geosci.* **2015**, *85*, 196–209. [CrossRef]
77. Liu, J.; Kang, X.; Dong, C.; Xu, S. A stream tilling approach to surface area estimation for large scale spatial data in a shared memory system. *Open Geosci.* **2017**, *9*, 622–634. [CrossRef]
78. Dorier, M.; Antoniu, G.; Cappello, F.; Snir, M.; Orf, L. Damaris: How to efficiently leverage multicore parallelism to achieve scalable, jitter-free I/O. In Proceedings of the 2012 IEEE International Conference on Cluster Computing (CLUSTER), Beijing, China, 24–28 September 2012; IEEE: New York, NY, USA, 2012; pp. 155–163.
79. Kang, X.; Liu, J.; Lin, X. Streaming progressive tin densification filter for airborne lidar point clouds using multi-core architectures. *Remote Sens.* **2014**, *6*, 7212–7232. [CrossRef]
80. Kang, X.; Lin, X. Graph-based divide and conquer method for parallelizing spatial operations on vector data. *Remote Sens.* **2014**, *6*, 10107–10130. [CrossRef]
81. Hendrickson, B.; Kolda, T.G. Graph partitioning models for parallel computing. *Parallel Comput.* **2000**, *26*, 1519–1534. [CrossRef]
82. Lipton, R.J.; Tarjan, R.E. A separator theorem for planar graphs. *SIAM J. Appl. Math.* **1979**, *36*, 177–189. [CrossRef]
83. Skiena, S. *The Algorithm Design Manual*, 2nd ed.; Springer Science & Business Media: New York, NY, USA, 2008.
84. Karypis, G.; Kumar, V. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.* **1998**, *20*, 359–392. [CrossRef]
85. Andreev, K. Balanced graph partitioning. *Theor. Comput. Syst* **2006**, *39*, 929–939. [CrossRef]
86. Yang, C.; Goodchild, M.; Huang, Q.; Nebert, D.; Raskin, R.; Xu, Y.; Bambacus, M.; Fay, D. Spatial cloud computing: How can the geospatial sciences use and help shape cloud computing? *Int. J. Digit. Earth* **2011**, *4*, 305–329. [CrossRef]
87. Goodchild, M.F.; Yuan, M.; Cova, T.J. Towards a general theory of geographic representation in GIS. *Int. J. Geogr. Inf. Sci.* **2007**, *21*, 239–260. [CrossRef]
88. De Smith, M.J.; Goodchild, M.F.; Longley, P. *Geospatial Analysis: A Comprehensive Guide to Principles, Techniques and Software tools*, 2nd ed.; Troubador: Leicester, UK, 2008.
89. Yang, C.; Wu, H.; Huang, Q.; Li, Z.; Li, J. Using spatial principles to optimize distributed computing for enabling the physical science discoveries. *Proc. Natl. Acad. Sci. USA* **2011**, *108*, 5498–5503. [CrossRef] [PubMed]
90. Hendrickson, B.; Leland, R. An improved spectral graph partitioning algorithm for mapping parallel computations. *SIAM J. Sci. Comput.* **1995**, *16*, 452–469. [CrossRef]

91. Nour-Omid, B.; Raefsky, A.; Lyzenga, G. Solving finite element equations on concurrent computers. In Proceedings of the 1986 Symposium on Parallel Computations and Their Impact on Mechanics, Boston, MA, USA, 13–18 December 1987; pp. 209–227.
92. Miller, G.L.; Teng, S.H.; Vavasis, S.A. A unified geometric approach to graph separators. In Proceedings of the 1991 Symposium on Foundations of Computer Science, San Juan, PR, USA, 1–4 October 1991; pp. 538–547.
93. Miller, G.L.; Teng, S.; Thurston, W.; Vavasis, S.A. *Automatic Mesh Partitioning*; Springer: New York, NY, USA, 1993; pp. 57–84.
94. Garbers, J.; Promel, H.J.; Steger, A. Finding clusters in vlsi circuits. In Proceedings of the 1990 IEEE International Conference on Digest of Technical Papers, Santa Clara, CA, USA, 11–15 November 1990; pp. 520–523.
95. Cheng, C.K.; Wei, Y.C.A. An improved two-way partitioning algorithm with stable performance [VLSI]. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **1991**, *10*, 1502–1511. [CrossRef]
96. Bui, T.N.; Jones, C. A heuristic for reducing fill-in in sparse matrix factorization. *Proc. Parallel Process. Sci. Comput.* **1993**, *1*, 445–452.
97. Hendrickson, B.; Leland, R. A multi-level algorithm for partitioning graphs. In Proceedings of the 1995 ACM/IEEE Conference on Supercomputing, New York, NY, USA, 4–8 December 1995; pp. 28–41.