symmetry

MDPI

Article

Parallel Spatial-Data Conversion Engine: Enabling Fast Sharing of Massive Geospatial Data

Shuai Zhang [1,2,3], Manchun Li [1], Zhenjie Chen [1,*], Tao Huang [1], Sumin Li [4], Wenbo Li [5] and Yun Chen [6,*]

1 Jiangsu Key Laboratory of Geographic Information Science and Technology, Nanjing University, Nanjing 210093, China; zhangshuainj@gmail.com (S.Z.); manchun@nju.edu.cn (M.L.); ht.anglenx@gmail.com (T.H.)
2 Department of Geography, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA
3 National Center for Supercomputing Applications (NCSA), University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA
4 School of Architecture, Changsha University of Science and Technology, Changsha 610059, China; lisumin57@gmail.com
5 Hefei Institute of Technology Innovation, Chinese Academy of Sciences, Hefei 230031, China; wbli@iim.ac.cn
6 Shanghai Aerospace Control Technology Institute, Shanghai 201109, China
* Correspondence: chenzj@nju.edu.cn (Z.C.); chenyun3305@sohu.com (Y.C.)

Received: 25 February 2020; Accepted: 23 March 2020; Published: 1 April 2020

Abstract: Large-scale geospatial data have accumulated worldwide in the past decades. However, various data formats often result in a geospatial data sharing problem in the geographical information system community. Despite the various methodologies proposed in the past, geospatial data conversion has always served as a fundamental and efficient way of sharing geospatial data. However, these methodologies are beginning to fail as data increase. This study proposes a parallel spatial data conversion engine (PSCE) with a symmetric mechanism to achieve the efficient sharing of massive geodata by utilizing high-performance computing technology. This engine is designed in an extendable and flexible framework and can customize methods of reading and writing particular spatial data formats. A dynamic task scheduling strategy based on the feature computing index is introduced in the framework to improve load balancing and performance. An experiment is performed to validate the engine framework and performance. In this experiment, geospatial data are stored in the vector spatial data defined in the Chinese Geospatial Data Transfer Format Standard in a parallel file system (Lustre Cluster). Results show that the PSCE has a reliable architecture that can quickly cope with massive spatial datasets.

Keywords: spatial data sharing; spatial data conversion; parallel computing; computing cluster; vector geo-spatial data transfer

1. Introduction

The current tools and equipment for capturing geospatial data at both mega and milli scales are insufficient. Spatiotemporal data acquired through remote sensors (e.g., remote sensing images), widespread location-aware mobile devices, and large-scale simulations (e.g., climate data) have always been "big" [1–3]. Furthermore, considerable data have accumulated via the geographical information systems (GIS) of different institutions, organizations, and communities worldwide. However, reusing existing spatial data for new applications remains challenging [4,5].

The complexity of data sharing is believed to originate from two main characteristics within data sources, namely, distribution and heterogeneity [6–8]. Heterogeneity problems, including syntactic, schematic, and semantic heterogeneity, can be very complicated [6,8,9]. In syntactic heterogeneity, various systems use different data types, structures, languages, or formats to describe data. Schematic problems are encountered when different information systems maintain their data using different data models, database structures, or interfaces. Moreover, semantic heterogeneity often occurs at the information level, requiring systems to understand the content of an item of information and its meaning, that is, semantics [9,10].

Despite such obstacles, the reusability of existing data is still essential simply because of the high cost of acquiring new geographical data from scratch. Many researchers have presented various approaches that enable spatial data sharing and example systems [6,8–20]. These approaches may be classified into three general categories, namely, spatial data conversion, spatial federated database, and mediator-based data sharing.

Spatial data conversion techniques are the basic methods for spatial data sharing. They can be traced to the beginning of the GIS industry when a huge amount of spatial data were collected in isolated systems or departments, and the need for data sharing increased among geospatial data producers. People have attempted to convert spatial data from data sources directly into their own systems to overcome data sharing problems [8,9,14]. Currently, spatial data conversion techniques have developed extremely well and gained the power to transfer between two formats in over a hundred cases. However, such techniques often cause data redundancy and result in improper stress when coping with large datasets.

Federated database techniques soon took the focus because they had several advantages over data conversion methodologies. They have allowed each database owner to define the subsets of local data and then integrate them into one virtual database by establishing a global schema in a common data model, thereby allowing users to access the federated database like a centralized database, without having to worry about the actual physical location of data or the type of a local DBMS (DBMS, Database Management System) [6,17].

The constant improvement of distributed computing, networks, and middleware technology enables GIS practitioners to develop complicated and powerful mediator-based systems, allowing them to share data in a large scale. Such systems were designed to be constructed from numerous and relatively autonomous sources of data and services, which could communicate with one another over a standard protocol, enabling users to issue a single query to the mediator and provide the capability to return a result that seamlessly combines information from multiple sources [15,21,22].

Federated database tech and mediator-based data sharing systems bring substantial flexibility under control, which is good but can be really complicated and unreliable if autonomous databases are imbalanced and change frequently. Furthermore, as large quantities of data and massive requests keep pouring in the big data era, federated databases and mediator-based systems are slow to respond because they have to overcome great overhead to achieve interoperability. Moreover, many agencies do not want others to access their databases due to security and other issues. Therefore, spatial data conversion techniques are still a valuable and efficient way of achieving geospatial data sharing.

However, current situations are quite different. A traditional spatial data conversion method that runs on a single computer is increasingly becoming a bottleneck for geospatial data sharing as data increase rapidly. A computing pattern shift, known as data intensive computation, can be observed throughout the IT industry [3,23–25]. Multicore CPUs (CPU, Central Processing Unit) and multi-nodes computing cluster are increasingly utilized. Therefore, a new approach that enables massive spatial data sharing by utilizing parallel computing and high-performance computing technics is necessary. Shuai proposed a parallel geo-raster data conversion engine (PGRCE) in 2015 aimed at geospatial raster data sharing, which achieved a 7.54 speedup of eight computing nodes [26]. However, geospatial data in a vector format are much more complicated and demand a closer concern.

This study proposes a novel parallel spatial data conversion engine (PSCE) to achieve the efficient sharing of massive geodata especially in vector format by utilizing high-performance computing technics. Compared to PGRCE, a feature computing index (FCI) was introduced to evaluate the actual workload from each feature object in order to achieve better load balancing. The rest of this paper is structured as follows. Related works on spatial data conversion techniques and the problems they encounter are presented in Section 2. The framework and architecture of the proposed PSCE are discussed in detail in Section 3. A use case, which converts a large dataset of the Chinese standard vector geo-spatial data transfer (VCT) format into GeoJSON format, is utilized to demonstrate the framework in Section 4. Finally, the conclusions are discussed in Section 5.

2. Spatial Data Conversion Techniques

Spatial data conversion techniques, which mainly aim to input spatial data from one system or format and then output to another, have existed for a long time and are well documented in the literature [8,9,14]. However, the key point in the conversion process lies in the spatial models (the way people view the geospatial world) behind the spatial datasets, which actually determine the quality of the outcomes to a large extent. In other words, if the two models between are compatible or even based on the same model, then the conversion might be satisfactory. Otherwise, the conversion might be poor.

However, the conversion engine must consider several rules to obtain satisfactory results. The first rule concerns geometric data, that is, no geometries shall be lost or twisted unless the destination format does not support such a type. Second, the attributes associated with the geometries shall be considered and matched with one another. Other important information includes the spatial relationship between features, spatial reference, metadata, and symbol styles. An ideal conversion engine should also provide an expandable and customized way to handle all these aspects.

Spatial data conversion techniques have gone through three generations. Converting spatial data from system A directly into system B is the first attempt in this sense. The transfer program is designed for specific formats. Therefore, it has great control and usually acts fast.

However, the file formats in both directions, which are sometimes difficult to achieve or even impossible, must all be accessible and fully understood, especially when the data format is proprietary to an organization. Furthermore, developing transfers between two data formats is laborious. Even after they are developed, all the transfers related with this format will have to be revised accordingly if one format evolves into a new version.

Therefore, converting spatial data directly from format to format seems inefficient in a large-scale setting. However, an exchange format can be created to reduce the workloads, allowing formats to be transferred first into the exchange format and then to the destination format [13,27]. Many exchange formats exists, such as commercial releases (e.g., DXF for AutoCAD, MIF for MapInfo, and E00 for Arc/Info) and national standard exchange formats from all over the world (e.g., the Spatial Data Transfer Standard by the American National Standards Institute [11], Chinese National Geo-spatial data transfer format [13], the Standard Procedure and Data Format for Digital Mapping in Japan [28]). However, many redundant datasets may be produced in exchange format methods, and the quality of the conversion largely depends on the compatibility between the exchange format and the two ends. Moreover, excessive exchange formats exist worldwide, rendering all efforts in this direction futile.

The third generation of spatial data conversion technology has emerged through sharing a common interface library. A common interface library mediates the conversion process, interpreting from the source and serializing into the destination and enabling the transfer between two data formats directly. The entire process, from reading and transferring to writing, is quite flexible and controllable. Hence, specific adjustments can be made according to particular data formats. Furthermore, the architecture of the conversion engine is scalable, and new data formats can be simply plugged into the architecture without requiring the reconstruction of the engine itself or its other parts. GDAL/OGR, an open source translator library for geospatial data formats, and FME, a spatial data transformation software, are two famous examples of these formats.

3. Engine Framework

The PSCE, which evolved from the common interface library methodology, is a framework for massive spatial data sharing. It provides an infrastructure for data exchange among different spatial data formats in high-performance computing environments. The PSCE framework aims to simplify different data sources for users working on large computing clusters. The framework was designed to support the following six requirements for the massive spatial data converting process:

1. Effectiveness—reliability of data conversion and ability to reduce information loss;
2. Efficiency—ability to handle massive data conversion quickly under the HPC environment;
3. Expandability—can easily plug a new data format into the framework and participate in the conversion;
4. Concurrency—the potential to run on a large computing cluster concurrently;
5. Independence—independence of formats and ability to act separately;
6. Transparency—users should not see the complexities associated with data conversion.

3.1. Architecture

An abstract standard model for spatial data as a common interface is the first step toward developing a solution for the data conversion engine. The PSCE uses the OpenGIS standard [29,30] as the common data model to understand geospatial data in various data formats. The overview of the common interface set is shown in Figure 1. Each data format in the PSCE should implement its own provider, a specific way of loading or unloading data from a data format. The extendable architecture of the PSCE is symmetric and shown in Figure 2. It is mainly comprised of three parts, namely, *TRANSFER ENGINE*, *PROVIDER HARBOR*, and *PROVIDERs*. Each *PROVIDER* can be a customized wrapper for the particular spatial data format to load and unload the spatial data and to bridge the given data format and the abstract spatial data model.

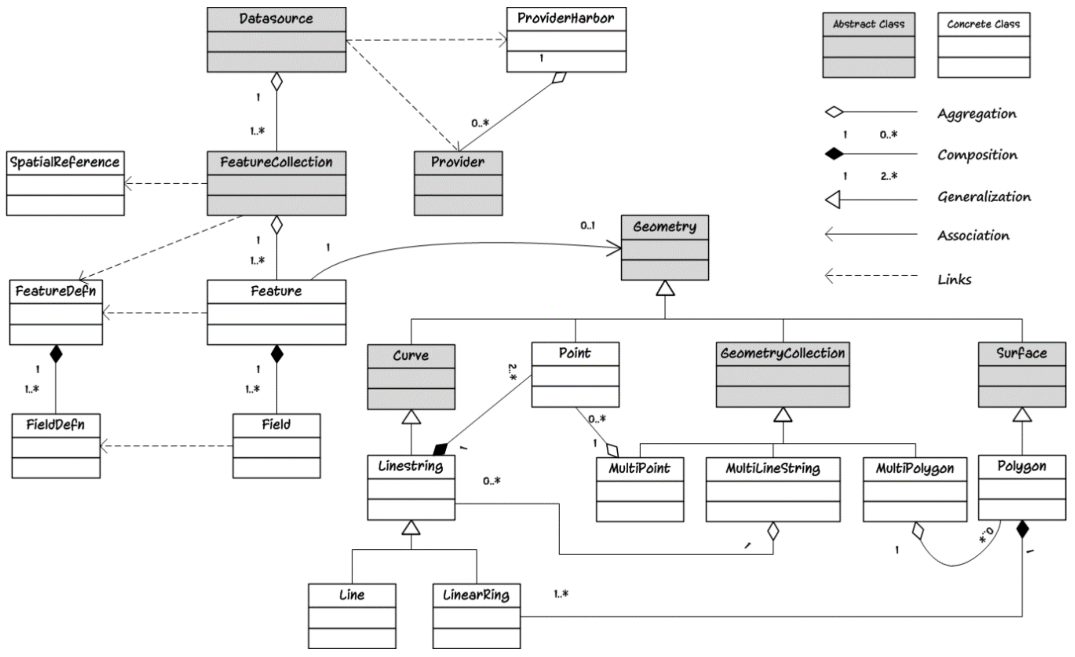


Figure 1. Common interface set of the parallel spatial data conversion engine (PSCE). Feature includes geometry and attributes. FeatureCollection represents a set of Features with access methods. FeatureDefn contains schema information for FeatureCollection. Field is the Feature attribute value union, and FieldDefn describes the definition of a Field. Provider acts as operational format driver, normally exist for each file format registered in the Provider Harbor.

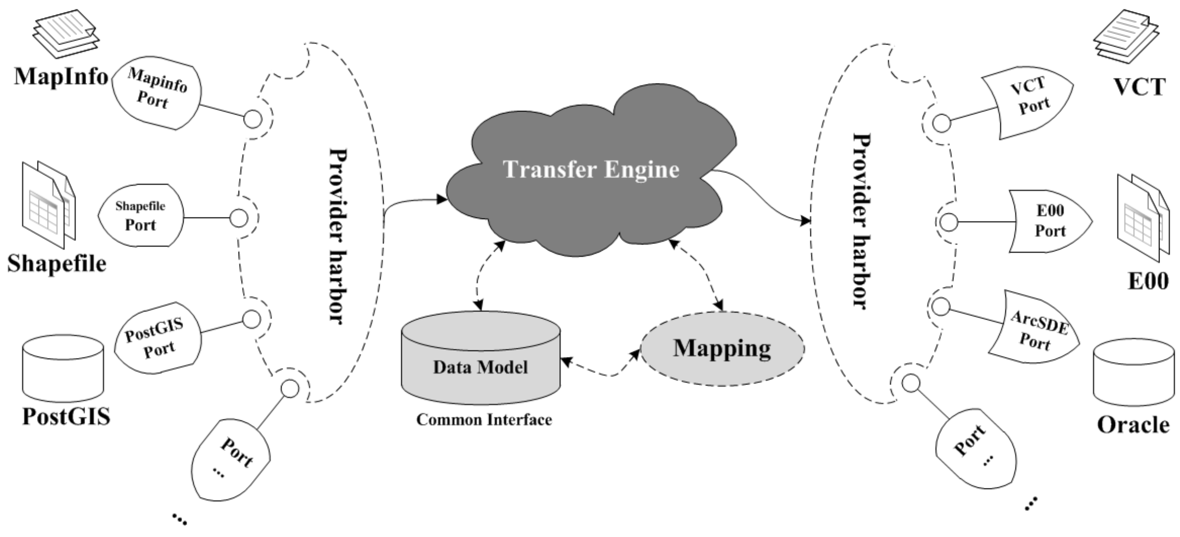


Figure 2. Symmetric architecture of the PSCE.

The conversion part, which includes at least three aspects, namely, geometry transferring, attributes mapping, and metadata conveying, is performed by the *TRANSFER ENGINE* (*TE*). Geometry transferring can sometimes be difficult because many spatial data models exist and are never perfectly compatible with one another. For example, the vector data format of the Chinese National Geo-spatial data transfer format defines a type of line called Circular Arc, which is described by only three points. Moreover, it cannot be found in OpenGIS SFA (OpenGIS Simple Feature Access) and can only be approximated via LinearString. The attribute information attached to the geometry is maintained in two lists (i.e., one for the origin and one for the destination). Before conversion actually starts, a mapping relationship between the two is established to identify which part of the information should be delivered to the destination and thus prevent mismatch.

3.2. Task Scheduler

The PSCE utilizes parallel computing technology, which employs a manager–worker paradigm of two entities, namely, a manager and multiple workers. On the one hand, the manager decomposes the problem logically into small tasks, distributes the tasks among a set of workers, and gathers the computing statistics from each worker. On the other hand, the workers act in a very simple cycle, that is, they obtain a task instruction from the manager, process the task, and send the computing statistics back to the manager.

The manager decomposes the entire problem according to a domain partitioning strategy, which deems one Feature object as the minimum unit of a typical geospatial data. By dividing the input spatial data into parts without any dependencies, the partitioning strategy is to achieve one of the most important goals, that is, to decouple the processes. As such, communication or collaboration between workers will not be necessary, thereby greatly simplifying the relationship between processes and thus accomplishing high computational speedups.

A dynamically load-balanced strategy was used to allocate tasks to worker processes, which enable it to adapt to the changing system conditions. A FIFO (FIFO, First Input First Output) queue (an idle worker queue) and a FILO stack (barrels to be done, called a cellar) are maintained. PSCE combines one idle worker from the worker queue and one barrel from the cellar to form a task instruction, then sends the instruction to the worker until the cellar is empty. The workers receive the task instructions, fulfill the task, and report feedback to the manager. Thus, the cycle can continue until all the spatial data are converted, as shown in Figure 3. This dynamically load-balanced strategy results in PSCE's robust performance, allowing it to respond appropriately for the failure of certain processors.

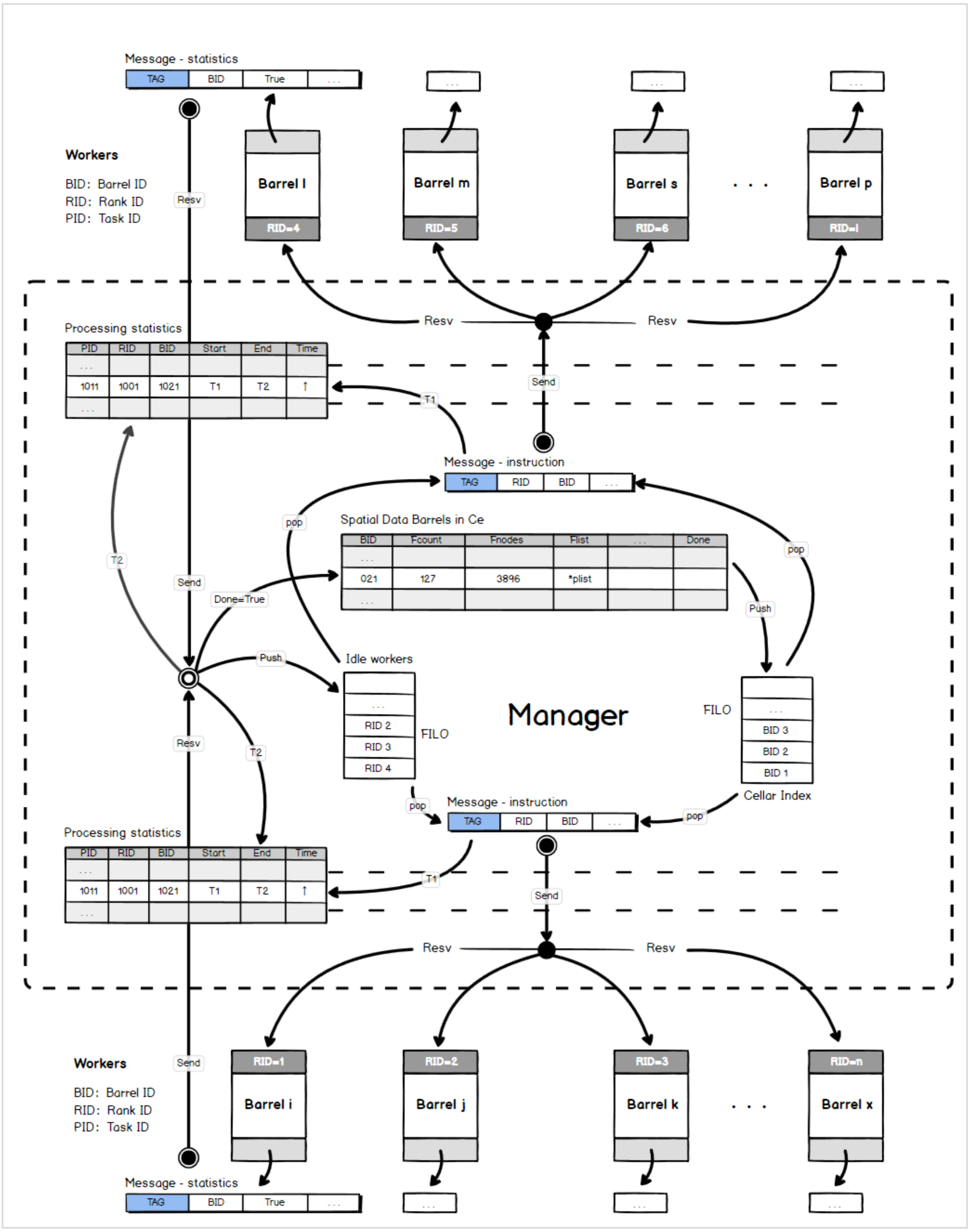


Figure 3. Task scheduler of the manager–worker paradigm in the PSCE. The cellar is a container of barrels who describes the workload unit, such as BID (Barrel ID, the identifier of barrels), Fcount (the total number of Feature objects in the barrel), feature computing index (FCI) (the total computing index of the barrel), Flist (the list of the Feature objects in the barrel). Processing statistics table keep the processing information of each task, including the processor (Rank ID, RID), barrel (BID), and the start–end time.

3.3. Domain Parititioning Strategy

Geospatial data conversion is a typical data-intensive problem [3,25], and the most time-consuming part is the data themselves. Load balance is important because the slowest worker will determine the overall performance if all workers are subject to barrier synchronization point. The same number of Feature objects (EFC, Equal Feature Count) may not be a good partition choice because the data size of a single Feature object can vary in a large range due to the complexity of the geometry within. Furthermore, the attributes actually have the same data size because each Feature in a FeatureCollection has the same schema. Thus, the PSCE proposes a feature computing index (FCI) as an estimation of the actual workload from each Feature object. The FCI of a feature object can be described as follows:

$$F_i = Atr + G_i \tag{1}$$

where *Atr* is the total bytes of the attribute data divided by one POINT bytes. G_i is the total number of POINTs in the geometry contained in Feature *I*. F_i is the total feature computing index of Feature object *i*.

The PSCE first examines all the spatial data and builds a computing index of each Feature object, calculates their volume, and then places them into barrels by Feature ID. As such, each piece of the data is quite computationally intensive. Nevertheless, the number of Feature objects in one barrel may vary from a dozen to a thousand, depending on the following formula:

$$\sum\nolimits_{i=1}^{n} F_i = \mathrm{n}{*}Atr + \sum_{i=1}^{n} G_i \leq V_0 \tag{2}$$

where n is the amount of Feature objects in the barrel. V_0 is the total volume of the given barrel. And Atr, G_i, and F_i have the same meaning as in Equation (1).

Therefore, each of the barrels as a single task would have a similar FCI and thus a similar number of workloads. However, the efficiency of massive spatial data conversion is severely limited by the relatively low I/O performance of most current computing platforms [31]. The PSCE utilizes a parallel file system, Lustre, to manage large spatial data files and takes advantage of its powerful I/O capacity to promote overall performance. All spatial data files are arranged in the Lustre file system, and each data format provider is implemented under the support of the MPI-IO to achieve parallel access to a large spatial dataset. However, the manager process in the PSCE does not read or write spatial data directly. It only fetches certain metadata to make a wise decision, whereas workers communicate with the I/O scheduler of the file system directly. The message between the manager and workers are only instructions or reports of small data size. As such, the communication within will not suffer, as shown in Figure 4.

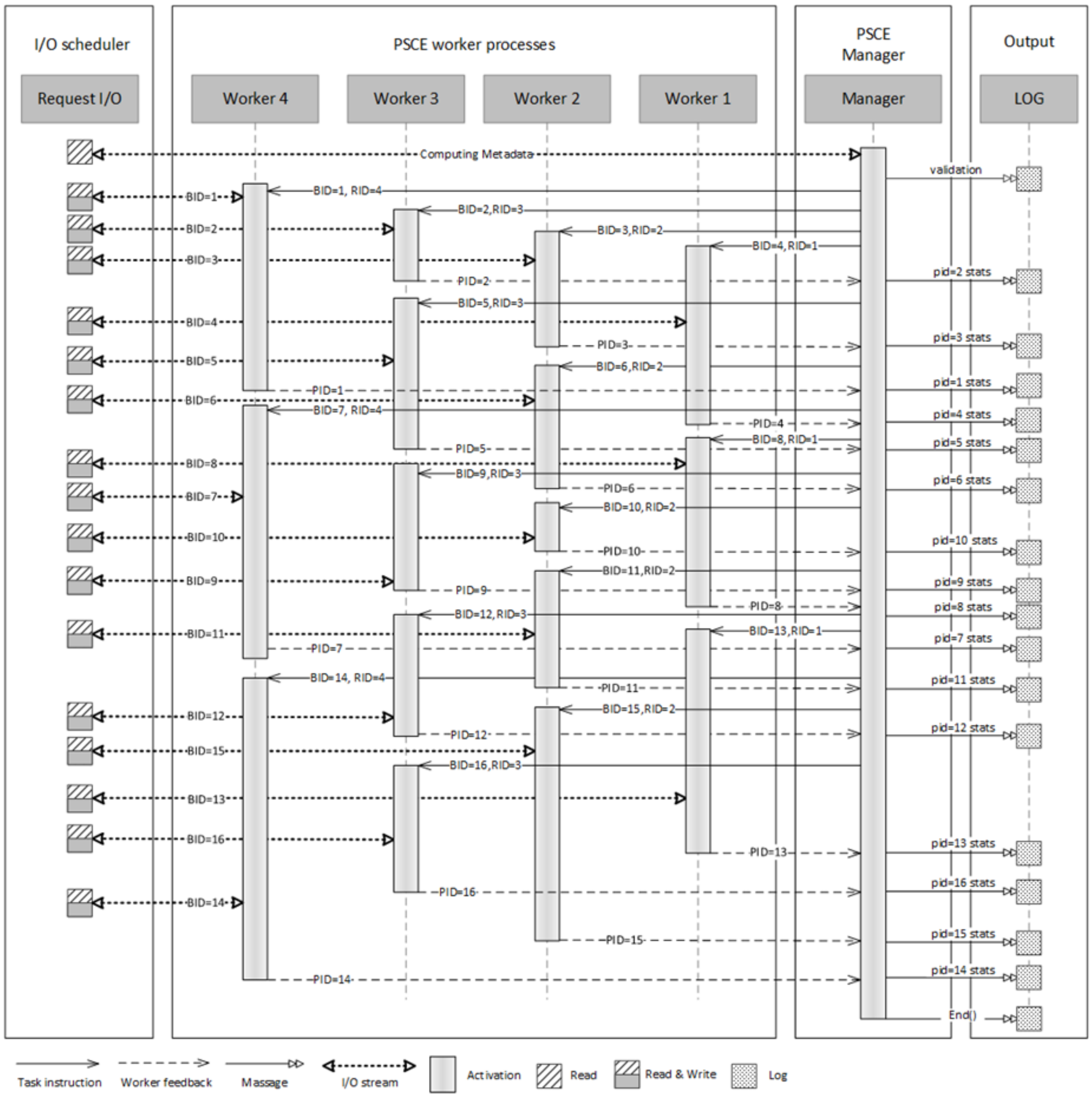


Figure 4. Communication and data stream in the PSCE. Assuming that 16 barrels and four workers exist, the manager process dynamically assigns the tasks to the workers and print out the reports into logs. Each worker then directly reads and writes data from files or database.

4. Use Case

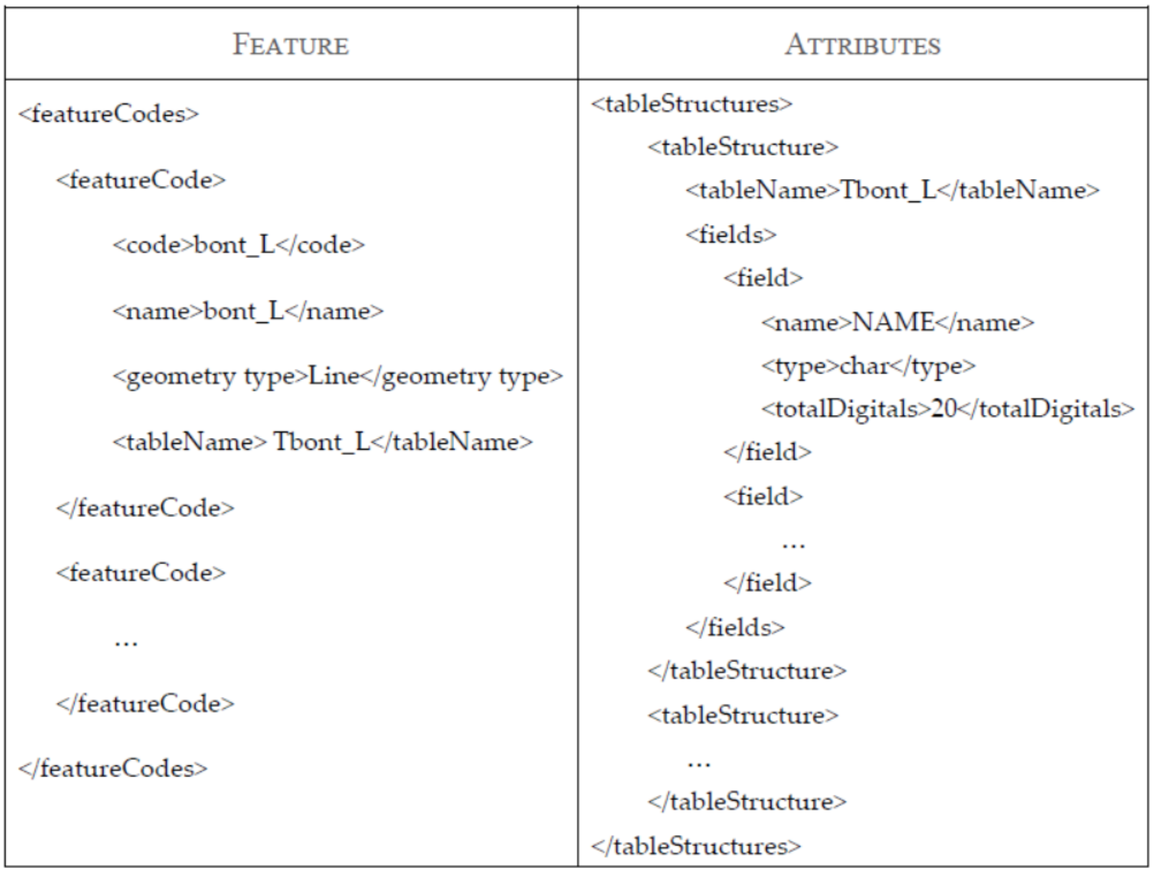
To validate the framework and evaluate the performance of the PSCE described in the previous sections, an experiment was conducted to benchmark the performance of the PSCE. The experiment was performed on a Linux cluster (with one metadata target (MDT) and nine object storage targets (OSTs) to import a large VCT dataset into a GeoJSON format. The GeoJSON specification is quite similar to the geometric hierarchical structure designed in the OpenGIS SFA. Therefore, writing a GeoJSON Provider is theoretically easy.

However, a VCT file can be complicated because points, lines, and polygons are stored alongside annotation and symbol styles. Moreover, four types of points, eight lines, five polygons, and even a solid (3D) data exist within the file. This work attempts to present how the PSCE understands the VCT format and pays close attention to the core geographical information, geometric information, and linked attributes.

4.1. VCT Provider

The first step to recruit the VCT format is to create a new Provider for the VCT format, following the common interface and registering it in the ProviderHarbor. In the PSCE, each VCT document is referred to as a Datasource and may contain several FeatureCollections whose geometry type might vary from point to polygon. The metadata described between <vectorMetadata> < /vectorMetadata > in the head of a VCT file maintains the geographic coordinate system information and is parsed to instance the SpatialReference associated with Datasource.

After the <vectorMetadata> part, it comes in the order with the <featureCodes>, <tableStructures>, <pointFeatures>, <curveFeatures>, <polygonFeatures>, and <attributeTables>, in which the actual spatial data are described. The PSCE regards every <featureCode> part as a FeatureCollection. For example, a FeatureCollection named bont_L, which has a geometry type of line, is defined in Figure 5, and the curves belonging to bont_L will appear in the latter <curveFeatures> part. The <tableStructures> part presents all the attribute tables, and the definitions of FeatureDefn and FieldDefn to describe the structure of the FeatureCollection. Consequently, with this structure knowledge, the PSCE can interpret the latter three data parts.



FEATURE	ATTRIBUTES
<featureCodes>	<tableStructures>
	<tableStructure>
<featureCode>	<tableName>Tbont_L</tableName>
<code>bont_L</code>	<fields>
	<field>
<name>bont_L</name>	<name>NAME</name>
<geometry type>Line</geometry type>	<type>char</type>
	<totalDigitals>20</totalDigitals>
<tableName> Tbont_L</tableName>	</field>
</featureCode>	<field>
	…
<featureCode>	</field>
…	</fields>
	</tableStructure>
</featureCode>	<tableStructure>
</featureCodes>	…
	</tableStructure>
	</tableStructures>

Figure 5. Sample data to illustrate vector geo-spatial data transfer (VCT) data structure. In this example, a FeatureCollection called bont_L with its attribute table Tbont_L is defined.

4.2. Geometric Transfer

The VCT spatial data model was not built under the OpenGIS SFA standard. Therefore, compatibility problems exist between them, and they may not always be easy to fuse them together. The transfer tables for geometries between VCT and SFA are shown in Table 1, Table 2, and Table 3 for points, lines, and polygons, respectively. The points defined in the VCT file are quite simple and can be directly converted to POINT or MULTIPOINT in the SFA, as shown in Table 1.

Table 1. Point data transfer rules.

VCT CODE	VCT TYPE	VCT DEFINITION	SFA TYPE
1	Single Point	<1> [25] {<node coordinate> [25]}	POINT
2	Joint Point	<2> [25] {<node coordinate> [25]}	POINT
3	Directed Point	<3> [25] {<node coordinate> [25]}	POINT
4	Point Set	<4> [25] <NON*> [25] {<nodes coordinates> [25]}$_n^n$	MULTIPOINT

* Number of nodes.

Table 2. Line data transfer rule.

VCT CODE	VCT TYPE	VCT DEFINITION	SFA TYPE
11	Polyline	<NON> [25] {<nodes coordinates> [25]}$_n^n$	LINESTRING
12	Circular Arc	<3> [25] {<nodes coordinates> [25]}$_3^3$	LINESTRING
13	Circular Arc	<Centre> [25] <Radius> [25] <SA^1> [25] <EA^2> [25]	LINESTRING
14	Elliptical Arc	< Centre> [25]< SMANC^3> [25] <Ratio> [25] <SA^1> [25] <EA^2> [25]	LINESTRING
15	Cubic Spline	<NON> [25]{<nodes coordinates> [25]}$_n^n$<start coordinate> [25]<end coordinate > [25]	LINESTRING
16	Basis Spline	<NON> [25] {<nodes coordinates> [25]}$_n^n$ <Degree> [25]<Description> [25]	LINESTRING
17	Bézier Curve	<NON> [25] {<nodes coordinates> [25]}$_n^n$ <Degree> [25]<Description> [25]	LINESTRING
100	Integrated line	<NOL^4> [25] {<line id> [25]}$_0^\infty$	MULTILINESTRING

^{1} Start angle, ^{2} End angle, ^{3} Semi-major axis node coordinate, ^{4} Number of lines

Table 3. Polygon data transfer rule.

VCT CODE	VCT TYPE	VCT DEFINITION	SFA TYPE
11	Polygon	<NON> [25] {<nodes coordinates> [25]}$_n^n$	POLYGON
12	Circle	<3> [25] {<nodes coordinates> [25]}$_3^3$	POLYGON
13	Circle	<Centre> [25] <Radius> [25]	POLYGON
14	Elliptical Arc	<Centre> [25]< SMANC> [25] <Ratio> [25]	POLYGON
21	Integrated lines	<NOL> [25] {<line id> [25]}$_0^\infty$	POLYGON
22	Integrated polygons	<NOP*> [25] {< polygon id> [25]}$_0^\infty$	MULTIPOLYGON

*NOP: Number of polygon.

For the lines in VCT, polyline (CODE 11) is quite similar to the LINESTRING in the SFA. So is the integrated line (CODE 100) for MULTILINESTRING. Thus, direct conversion can be adopted. For the rest of the line types, the following formula is employed for resampling and for optimal approximation (illustrated in Figure 6):

$$CURVE(X_i, Y_i) = 0,$$
$$X_j < X_i < X_{j+1},\ i \in \{1, 2, 3, 4, 5, 6\},\ X_j, X_{j+1} \in \{Point \big| CURVE(X, Y) = 0\} \tag{3}$$

where $CURVE(X,Y) = 0$ is the function to describe the curve defined in the VCT file; $\{Point \mid CURVE(X,Y) = 0\}$ means that the existing points provided in the definition of the curve for line type 13,14 the start and end points are used, and X_j and X_{j+1} are two adjacent points along the way toward the X increasing direction.

Polygons exhibit a similar situation. The integrated lines (CODE 21), integrated polygons (CODE 22) in VCT and POLYGON, and the MULTIPOLYGON in SFA are nearly peer to peer, and thus suitable for a straight transfer. By contrast, VCT polygon types 12, 13, and 14 have to be resampled and converted first into LINEARRING before building the polygon in SFA with the following formula:

$$CURVE(R, \theta_i) = 0,\ 0 \le \theta < 2\pi,\ i \in \{1, 2, \ldots, 16\} \tag{4}$$

where R is the radius; θ is the angle; and $CURVE(R, \theta_i) = 0$ is the function to describe the polygon curve defined in the VCT file.

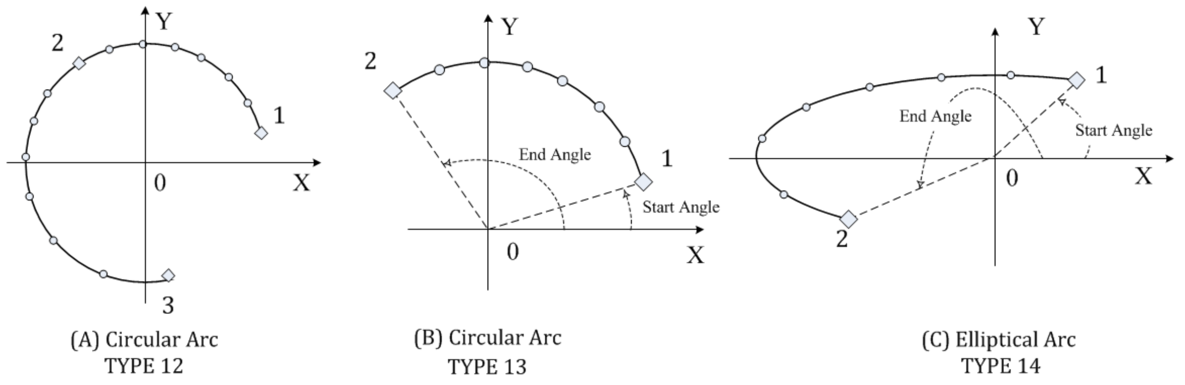


Figure 6. Illustration for the resampling of the VCT Line Type 12, 13, and 14. ∈{*Point*| *CURVE*(*X*,*Y*) = 0}, ∈{*Point*| *CURVE*(X_i,Y_i) = 0}.

4.3. Performance Evaluation

Experiments were conducted for benchmarking and to demonstrate the framework and evaluate the performance of the PSCE. The testing data included 420,000 polygons, which were generated in a heavily uneven distribution with a feature size standard deviation (STD) of 99.30, as shown in Table 4. The minimum polygon only contains three points, with an attribute size of 1.11 KB and a maximum feature size of 401.18 KB. The data were then partitioned and converted with the EFC and FCI methods, respectively. Table 4 indicate that in the EFC method, whose partitions have the same number of features, 5000 features results in a heavily uneven distribution with a minimum partition data size of 3.60 MB and a maximum partition data size 1876.83 of MB. Meanwhile, the FCI method achieves a much even partition, with a data size STD of only 6.29.

Table 4. Statistics of the testing VCT data.

FEATURES	COUNT	MAX	MIN	AVG	STD	OVERALL
DATA SIZE	420,000	401.18 KB	1.11 KB	53.79 KB	99.30	21.55 GB
EFC	84	1876.83 MB	3.60 MB	261.54 MB	434.75	21.55 GB
FCI	81	271.64 MB	218.21 MB	269.92 MB	6.29	21.55 GB

The experiments were conducted in a Lustre cluster with one metadata target (MDT) and eight object storage targets (OSTs). Each node has eight Intel(R) Xeon(R) CPU E5-2603 cores and 16 GB of memory. EFC and FCI methods invoked eight workers, one manager, and one process on each node in which the manager partitions the data and distributes the tasks among workers. The workers deal with the conversion of each part until all data are processed. Table 5 shows the benchmark results. Given that the PSCE is an I/O intensive algorithm and the EFC partitioned the data into heavily uneven distributions, the performance of each partition in the EFC was also very diverse, from a minimum time of 1.53 s to a maximum time 2269.23 s. By contrast, the FCI exhibited a quite stable performance, with an average time of 155.46 s and a STD 23.75. Overall, FCI could achieve 6.99 of acceleration, whereas EFC could only achieve 3.30 (shown in Figure 7).

Table 5. Benchmark performance between the EFC and FCI methodologies.

TYPE	COUNT	MAX	MIN	AVG	STD	OVERALL
EFC	84	2269.23 s	1.53 s	233.05 s	486.37	3583.27 s (59.72 min)
FCI	81	198.67 s	133.92 s	155.46 s	23.75	1692.26 s (28.20 min)

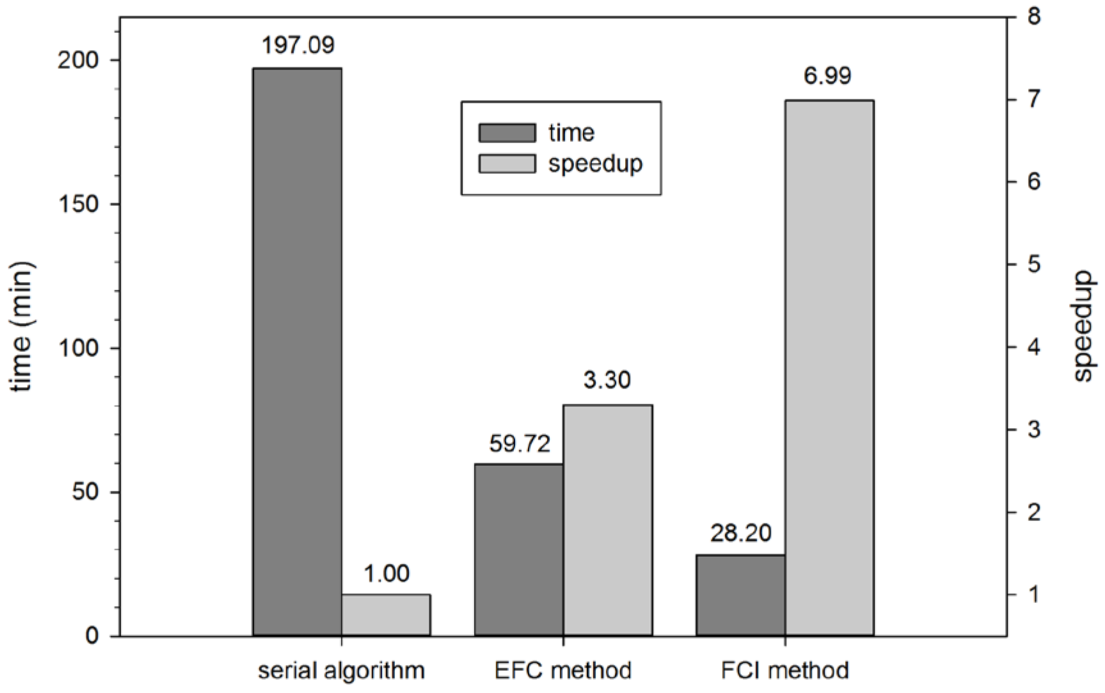


Figure 7. Speedup comparison of the PSCE.

5. Conclusions

In the era of big data, large amounts of geospatial data have been accumulated in isolated systems throughout the world. Large scale of geospatial data sharing between organizations in the GIS communities has always been a challenge. Traditional spatial data conversion methods that are running on a single computer are increasingly becoming bottlenecks for data sharing. Meanwhile multicore CPUs and multimode computing clusters are increasingly utilized. Therefore, a new approach that enables massive spatial data sharing by utilizing parallel computing and high-performance computing technics is critical.

The PSCE presented in this paper is a generic framework for massive spatial data sharing, which is built upon large computing clusters in an attempt to enhance the speed and reliability of spatial data sharing. The framework utilizes a common interface based on the OpenGIS Simple Feature Model, which bridges two spatial data formats. The architecture is also flexible and extendable, and every data format can have a customized way of reading and writing its spatial data.

The PSCE could handle large datasets because it is designed to run in high-performance computing clusters with the support of a parallel file system or distributed spatial database. Hence, it had unmatched performance in comparison with sequential approaches. The dynamic task scheduling strategy based on the FCI was introduced in the framework to improve load balancing, allowing PSCE to cope with massive spatial datasets in a fast and stable manner.

However, as the number of worker processes is increasing, the PSCE will encounter its I/O limitations, and the speedup will decline. In future study, we will try to improve the geospatial data distribution in the computing cluster to break through the I/O bottleneck. We also will improve our method by hiring technologies such as temporal graph convolutional networks [32] and deep inferring network technologies [33] to predict the performance bottleneck in applications.

Author Contributions: All authors contributed equally to the conception of the idea, the design of experiments, the analysis and interpretation of results, and the writing and improvement of the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National Key Research and Development Program of China (Grant No.2017YFB0504205), the national Natural Science Foundation of China (No. 41571378), the Scientific Research

Foundation of Graduate School of Nanjing University (Grant No 2013CL09), and the China Scholarship Council (award to Shuai Zhang for 1 year's study abroad at the University of Illinois at Urbana–Champaign).

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

References

1. Vatsavai, R.R.; Ganguly, A.; Chandola, V.; Stefanidis, A.; Klasky, S.; Shekhar, S. Spatiotemporal data mining in the era of big spatial data: Algorithms and applications. In Proceedings of the 1st ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data; ACM: Redondo Beach, CA, USA, 2012; pp. 1–10.
2. Wang, S.; Cao, G.; Zhang, Z. *A CyberGIS Environment for Analysis of Location-Based Social Media Data, in Location-based Computing and Services*; Hassan, A.K., Amin, H.E., Eds.; CRC Press: Boca Raton, FL, USA, 2013.
3. Yang, C.; Huang, Q.; Li, Z.; Liu, K.; Hu, F. Big Data and cloud computing: Innovation opportunities and challenges. *Int. J. Digit. Earth* **2016**, *10*, 13–53. [CrossRef]
4. Lee, J.-G.; Kang, M. Geospatial Big Data: Challenges and Opportunities. *Big Data Res.* **2015**, *2*, 74–81. [CrossRef]
5. Li, S.; Dragicevic, S.; Castro, F.A.; Sester, M.; Winter, S.; Coltekin, A.; Pettit, C.J.; Jiang, B.; Haworth, J.; Stein, A.; et al. Geospatial big data handling theory and methods: A review and research challenges. *ISPRS J. Photogramm. Remote. Sens.* **2016**, *115*, 119–133. [CrossRef]
6. Devogele, T.; Parent, C.; Spaccapietra, S. On spatial database integration. *Int. J. Geogr. Inf. Sci.* **1998**, *12*, 335–352. [CrossRef]
7. Genesereth, M.R.; Keller, A.M.; Duschka, O.M. Infomaster: An information integration system. *Sigmod Rec.* **1997**, *26*, 539–542. [CrossRef]
8. Sheth, A.P. Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics. In *Interoperating Geographic Information Systems*; Springer Science and Business Media LLC: Berlin, Germany, 1999; pp. 5–29.
9. Bishr, Y. Overcoming the semantic and other barriers to GIS interoperability. *Int. J. Geogr. Inf. Sci.* **1998**, *12*, 299–314. [CrossRef]
10. Visser, U.; Stuckenschmidt, H. Interoperability in GIS – Enabling Technologies. In Proceedings of the 5th AGILE Conference on Geographic Information Science, Palma de Mallorca, Spain, 25 April 2002.
11. Fegeas, R.G.; Cascio, J.L.; Lazar, R.A. An Overview of FIPS 173, The Spatial Data Transfer Standard. *Cart. Geogr. Inf. Syst.* **1992**, *19*, 278–293. [CrossRef]
12. Walter, V.; Fritsch, D. Matching spatial data sets: A statistical approach. *Int. J. Geogr. Inf. Sci.* **1999**, *13*, 445–473. [CrossRef]
13. Wang, Y.; Gong, J.; Huang, J.; Deng, Y.I. the data transfer method based on geo-spatial data transfer format. *Acta Geod. Et Cartogr. Sin.* **2000**, *29*, 142–148.
14. Gong, J.; Shi, L.; Du, D.; de By, R.A. Technologies and standards on spatial data sharing. In *Proceedings of 20th ISPRS: Geo-imagery Bridging Continents*; ISPRS: Enschede, The Netherlands, 2004.
15. Stoimenov, L.; Djordjevic-Kajan, S. An architecture for interoperable GIS use in a local community environment. *Comput. Geosci.* **2005**, *31*, 211–220. [CrossRef]
16. Peachavanish, R.; Karimi, H.A.; Akinci, B.; Boukamp, F. An ontological engineering approach for integrating CAD and GIS in support of infrastructure management. *Adv. Eng. Inform.* **2006**, *20*, 71–88. [CrossRef]
17. Butenuth, M.; Gösseln, G.V.; Tiedge, M.; Heipke, C.; Lipeck, U.; Sester, M. Integration of heterogeneous geospatial data in a federated database. *ISPRS J. Photogramm. Remote. Sens.* **2007**, *62*, 328–346. [CrossRef]
18. Safra, E.; Kanza, Y.; Sagiv, Y.; Beeri, C.; Doytsher, Y. Location-based algorithms for finding sets of corresponding objects over several geo-spatial data sets. *Int. J. Geogr. Inf. Sci.* **2010**, *24*, 69–106. [CrossRef]
19. Paul, M.; Ghosh, S.K. A framework for semantic interoperability for distributed geospatial repositories. *Comput. Inf.* **2012**, *27*, 73–92.
20. Safra, E.; Kanza, Y.; Sagiv, Y.; Doytsher, Y. Ad hoc matching of vectorial road networks. *Int. J. Geogr. Inf. Sci.* **2013**, *27*, 114–153. [CrossRef]
21. Gupta, A.; Marciano, R.J.; Zaslavsky, I.; Baru, C. *Integrating GIS and Imagery through XML-Based Information Mediation*; Springer Science and Business Media LLC: Berlin, Germany, 1999; Volume 1737, pp. 211–234.
22. Wong, S.; Swartz, S.; Sarkar, D. A middleware architecture for open and interoperable GISs. *IEEE Multimed.* **2002**, *9*, 62–76. [CrossRef]

23. Hey, T. *The Fourth Paradigm – Data-Intensive Scientific Discovery*; Springer Science and Business Media LLC: Berlin, Germany, 2012; Volume 317.
24. Bryant, R.; Katz, R.H.; Lazowska, E.D. *Big-Data Computing: Creating Revolutionary Breakthroughs in Commerce, Science and Society*; Computing Community Consortium: Washington, DC, USA, 2008.
25. Wang, S.; Anselin, L.; Bhaduri, B.; Crosby, C.; Goodchild, M.F.; Liu, Y.Y.; Nyerges, T.L. CyberGIS software: A synthetic review and integration roadmap. *Int. J. Geogr. Inf. Sci.* **2013**, *27*, 2122–2145. [CrossRef]
26. Zhang, S.; Li, M.; Chen, Z.; Huang, T.; Jiang, X. Parallel geo-raster data conversion engine. *J. Natl. Univ. Def. Technol.* **2015**, *35*, 9–14.
27. Wortman, K. The Spatial Data Transfer Standard (FIPS 173): A Management Perspective. *Cart. Geogr. Inf. Syst.* **1992**, *19*, 294–295. [CrossRef]
28. Akiyama, M.; Takayama, N.; Okuyama, S.; Hisamatu, F.; Kojiroi, R.; Kiuchi, T. Standard procedure and data format for digital mapping. *Geocarto Int.* **1988**, *3*, 67–71. [CrossRef]
29. Herring, J.R. The OpenGIS data model. *Photogramm. Eng. Remote Sens.* **1999**, *65*, 585–588.
30. OpenGIS. *Simple Feature Access - Part 1: Common Architecture*; Open Geospatial Consortium Inc.: Wayland, MA, USA, 2011.
31. Newman, H. I/O Bottlenecks: Biggest Threat to Data Storage. 2009. Available online: http://www.enterprisestorageforum.com/technology/features/article.php/3856121/IO-Bottlenecks-Biggest-Threat-to-Data-Storage.htm (accessed on 27 February 2014).
32. Zhao, L.; Song, Y.; Zhang, C.; Liu, Y.; Wang, P.; Lin, T.; Deng, M.; Li, H. T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction. *IEEE Trans. Intell. Transp. Syst.* **2019**, 1–11. [CrossRef]
33. Tao, C.; Qi, J.; Li, Y.; Wang, H.; Li, H. Spatial information inference net: Road extraction using road-specific contextual information. *ISPRS J. Photogramm. Remote. Sens.* **2019**, *158*, 155–166. [CrossRef]