


Article

BiGeo: A Foundational PaaS Framework for Efficient Storage, Visualization, Management, Analysis, Service, and Migration of Geospatial Big Data—A Case Study of Sichuan Province, China

Xi Liu ^{1,2} , Lina Hao ^{1,*} and Wunian Yang ¹

¹ Faculty of Earth Science, Chengdu University of Technology, Chengdu 610059, China; liuxi.gis@foxmail.com (X.L.); ywn@cdut.edu.cn (W.Y.)

² Sichuan Basic Geographic Information Center, Ministry of Natural Resources, Chengdu 610093, China

* Correspondence: madingludejin@163.com

Received: 16 August 2019; Accepted: 10 October 2019; Published: 12 October 2019



Abstract: With the rapid development of big data, numerous industries have turned their focus from information research and construction to big data technologies. Earth science and geographic information systems industries are highly information-intensive, and thus there is an urgent need to study and integrate big data technologies to improve their level of information. However, there is a large gap between existing big data and traditional geographic information technologies. Owing to certain characteristics, it is difficult to quickly and easily apply big data to geographic information technologies. Through the research, development, and application practices achieved in recent years, we have gradually developed a common geospatial big data solution. Based on the formation of a set of geospatial big data frameworks, a complete geospatial big data platform system called BiGeo was developed. Through the management and analysis of massive amounts of spatial data from Sichuan Province, China, the basic framework of this platform can be better utilized to meet our needs. This paper summarizes the design, implementation, and experimental experience of BiGeo, which provides a new type of solution to the research and construction of geospatial big data.

Keywords: geospatial big data framework; large-scale distributed spatial database; distributed spatial data visualization; distributed spatial data management and analysis; distributed spatial information services; distributed spatial data integration and migration

1. Introduction

Sichuan Province, China has a total area of 486,000 square kilometers with 18 prefecture-level cities and three autonomous prefectures, including 54 municipal districts, 17 county-level cities, 108 counties, and four autonomous counties. As of the end of 2017, the resident population was 83.02 million. At present, the Sichuan Basic Geographic Information Center has a large amount of geographic information data on the province. The storage media include optical disks, hard disks, and disk arrays, whereas the data types include basic mapping data, geographic national data, public platform data, geological disaster data, and the exchange of shared data. In addition, the data formats include raster, vector, 3D, and tile formats. At present, approximately 1.7 PB of geospatial data on the province has been accumulated, and continues to grow at a rate of more than 20%. The maximum single-layer feature data number more than 10 million. How to effectively manage, analyze, and apply so much spatial data at low cost and with high efficiency has become a significant problem.

Big data technologies have been derived from the computer industry [1–3], mainly based on cloud computing and the Internet for solving the problem of the storage and calculation of massive amounts of information [4–6]. Such technologies are currently being widely used in network user behavior analysis [5,7,8], massive log analysis [9,10], advertisement recommendations [11], and other fields [12–14]. After many years of development, geographic information science has accumulated a large number of storage organization models [15–17], visualization schemes [18,19], spatial analysis and statistical algorithms [10,20,21], service publishing technologies [22–24], and data integration and migration standards [25–27] for spatial data. These technologies all face the problem of a distributed transformation in a big data environment. Because the complexity of spatial data and the degree of spatial correlations are much higher than those in general computer information [28,29], distributed transformation of this geographic information technology has become more difficult, and some models and algorithms cannot even be distributed, requiring new spatially distributed methods to replace the original technology. Such a technology convergence evolution is a long-term process.

With a classic four-layer big data structure [30], namely, IaaS, DaaS, PaaS, and SaaS layers, BiGeo is positioned as a general-purpose PaaS platform. BiGeo can be deployed on a virtualized cloud or physical device cluster at the IaaS layer, and through the unified management of massive data resources in the DaaS layer, providing geospatial big data capabilities to end users through platform-service and other functional interfaces. It is also possible to provide a geospatial big data custom application system for the SaaS layer through the basic frameworks and components.

2. Related Studies

At present, there are numerous geospatial big data-related software resources, which can mainly be divided into the following categories:

2.1. GIS Software

ArcGIS: ArcGIS GeoAnalytics Server applies distributed computing based on vector-based feature data [31,32]. It can be used to analyze big data or increase the speed of traditional ArcGIS desktop analytic workflows using ArcGIS Pro and Portal for ArcGIS.

SuperMap: SuperMap iObjects for the Spark spatial big data component extends the Spark spatial data model [33] from the kernel. Existing spatial analysis algorithms are reconstructed based on distributed computing technology. This significantly improves the efficiency of massive spatial data analysis. In addition, a series of new spatial analysis algorithms were developed for big data. SuperMap can be embedded directly into Spark, solving the problem of spatial big data analysis and various applications.

2.2. Geospatial Database

Oracle Spatial All-in-One machine: Oracle Spatial is used to store, manage, and query spatial data [34,35]. It provides a set of SQL schemas and functions for storing, retrieving, updating, and querying a collection of spatial features in a database. Oracle Exadata is an all-in-one hardware platform consisting of a Database Machine and Exadata Storage Server.

MongoDB, MySQL: MongoDB natively supports geo-indexing and can be used directly for location distance calculations and queries [36–39]. It also supports more convenient features such as a range query and distance calculation. MySQL implements a subset of an OGC-recommended SQL environment using the geometry type. The specifications describe a collection of SQL geometry types [40,41] and functions used to create and analyze their values.

2.3. Cloud Solution

AliCloud, Amazon Web Service (AWS): Apsara is an extremely large-scale general-purpose computing operating system independently developed and serviced by Alibaba Cloud. It currently provides services to enterprises, governments, and institutions in more than 200 countries and regions around the world [16,42,43], providing geographic information capabilities through the PostgreSQL/PostGIS service.

GeoCloud: GeoCloud is a cloud geographic information platform based on open-source projects such as PostgreSQL, PostGIS, MapServer, GDAL, TileCache, PHP, and Python [44–48]. GeoCloud is a full-featured geographic information platform that includes the management of spatial geographic data, mapping, and building applications.

Google Earth Engine: Google Earth Engine is a cloud platform provided by Google for online visual computing and analysis of a large amount of global-scale geoscience data (particularly satellite data). The platform has access to data from satellite imagery and other Earth observation databases and provides sufficient computing power to process such data. The Google Earth Engine contains more than 200 public datasets and more than 5 million images, and is increasing by approximately 4000 images per day, with a capacity of more than 5 PB [49–51].

2.4. Distributed Computing Framework

SpatialHadoop, HadoopGIS: SpatialHadoop mainly includes a simple spatial high-level language and a two-level spatial index structure. It contains basic spatial components and range queries, K-NN queries, and spatial links built on MapReduce. HadoopGIS provides spatial data partitioning for task parallelization and an index-driven spatial query engine to handle various types of spatial queries. Its implicit query parallelization generates correct results through MapReduce and boundary processing [52–58].

Geotrellis, GeoSpark: The GeoTrellis software architecture was designed to create scalable, high-performance geospatial web services, creating distributed geographic information processing services. It is used to process massive datasets and complete parallel geographic information processing operations. GeoSpark has a range of creative spatially flexible distributed datasets. With the help of these datasets, users can efficiently load, process, and analyze large-scale spatial data [35,59,60].

2.5. Global and Regional Case

Global Earth Observation System of Systems (GEOSS): GEOSS is a set of coordinated, independent Earth observation, information, and processing systems that interact and provide access to diverse types of information for a broad range of users in both public and private sectors [61,62].

European Copernicus Data and Information Access Service (DIAS): To foster data dissemination and address a strong need for simplification, the European Commission and the European Space Agency (ESA) decided to launch DIAS to offer users the capability to exploit Copernicus data and information without having to manage the transfer and storage on their own computer systems [63].

To summarize, research into the current geospatial big data framework has remained in the realm of isolated basic theoretical studies, and existing distributed spatial data related software is unable to face the rapidly growing spatial data demand in a simple, efficient, and inexpensive manner. Therefore, the research results achieved remain difficult to use quickly in actual situations. Based on the existing research results and software technology, we further propose our own solutions.

3. Design and Architecture

BiGeo is a collection of geospatial big data frameworks that incorporate numerous open-source projects. It provides a software development foundation for the efficient storage, visualization, management, analysis, service, and migration of PB-level spatial data in a distributed environment. Based on this framework, the application and development of various geospatial big data can be further carried out.

3.1. Users and Main Usage Scenarios

The main users of BiGeo are developers and staff working on geospatial big data applications. Such developers include the underlying core developers and secondary application system developers. Those working on applications include spatial data resource managers and spatial analysts.

The main use scenario of BiGeo is a regional organization similar to the Sichuan Basic Geographic Information Center, which is used to support the management and application of regional spatial data similar to those of Sichuan Province of China, as well as the construction of other related systems.

3.2. Functional and Nonfunctional Requirements

BiGeo needs to achieve good scalability. The framework should be able to support massive geospatial big data in a distributed environment. Its efficiency should be better than that of traditional geographic information system (GIS) software, such as ArcGIS and GeoServer [13,64]. The easy extension of BiGeo should be reflected in several aspects: First, it should be easy to expand at the data cluster level. By deploying new hardware devices, the storage and computing power of data resources can be scaled horizontally. Second, in terms of functional modules, it should also be extensible. According to the basic interfaces provided by the framework, various new functional requirements can be easily added. Finally, at the application level, its service capabilities should be scalable. For example, service interfaces can be clustered to improve the concurrency.

BiGeo should also support secondary development and it should be easy to build a large number of dedicated applications on it, including a geospatial data exchange and sharing platform, oil pipeline site management, water rights confirmation management, result directory query system, and service engine system. By building numerous new application systems based on the framework, BiGeo can meet the more professional needs and personalized customization of geospatial big data.

Cost-effectiveness is also an important consideration. The construction, operation, and maintenance costs of geospatial big data are extremely high. For example, the Oracle platform is used in the Sichuan Basic Geographic Information Center and manages the geographic information result data of the province; in addition, its cost is as high as USD \$1,200,000, whereas the cost of the SuperMap spatial distributed analysis platform, using software built by commercial companies, is USD \$420,000. These do not include numerous artificial technology costs, or hardware, software operation, or maintenance costs. Therefore, we hope to reduce these base costs by building our own geospatial big data framework based on open-source software.

3.3. Architectural Choices

As shown in Figure 1, the overall architecture of BiGeo is divided into four layers, namely, the infrastructure, data cluster, basic framework, and application layers. The infrastructure layer is mainly composed of various computing, storage, and network resources managed by the operating system, and is maintained and managed by facility personnel. The data cluster layer and the basic framework layer are the core components of BiGeo, and are composed of a data cluster, GIS engine, toolkit, desktop framework, and server framework. This section is mainly focused on the underlying and secondary developers. The application layer is for end users and consists of various application systems derived from BiGeo.

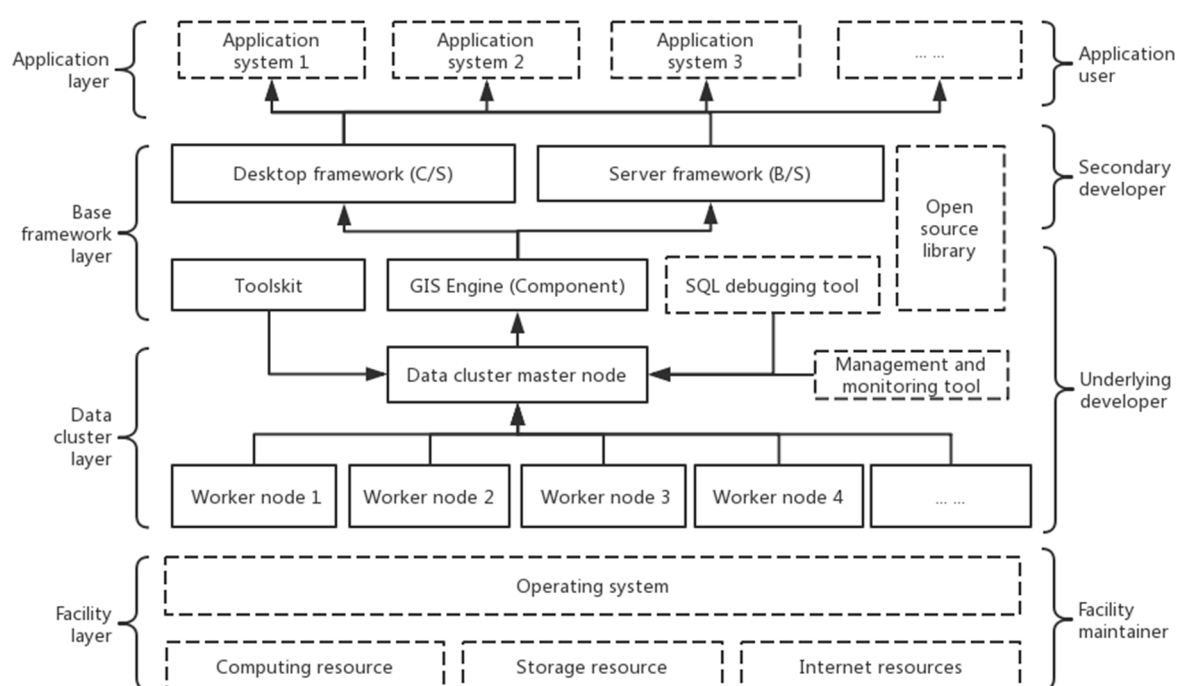


Figure 1. Overall design of BiGeo. BiGeo is a loosely assembled combination of five frames, each with the same technology stack, which can be combined and split at will. Each part of the framework can be used independently or in an extended manner. The solid box in the figure shows the core content of BiGeo, and the dashed box is the relevant support technology or derivative.

BiGeo Geospatial Big Data Platform software system (Figure 1) consists of five parts, as shown in Table 1.

Table 1. BiGeo software.

Software Name	Main Uses
Data Clustering	Basic data cluster software, distributed spatial database based on cheap x86 servers, high-performance distributed storage computing cluster. The function is similar to that of traditional Oracle Exadata with Spatial, Hadoop, and Spark, but has significantly improved efficiency and more spatial characteristics.
GIS Engine	The GIS engine component, namely, the kernel component responsible for map organization and visualization, is the underlying component of other parts and the secondary development interface. The function is similar to that of traditional MapObjects, SharpMap, and Mapnik, but can quickly visualize distributed spatial data and fit with traditional spatial data.
Desktop	Desktop-based basic software, desktop C/S Framework GIS Software. The function is similar to that of traditional ArcMap, QGIS, and Udig [65,66], whereas the distributed spatial data can be managed and analyzed flexibly through the plug-in mechanism.
Server	Server and front and back-end separation of geographic information service publishing basic software under B/S framework. The function is similar to that of traditional ArcServer, GeoServer, MapServer, and Nginx [44], but can quickly publish services for distributed spatial data, exposing various analysis capabilities of distributed spatial data clusters to service interfaces.
Toolkit	Toolset, Support ETL software. Data conversion, migration aids, etc. The function is similar to that of traditional FME, GeoKettle, and ModelBuilder, and can quickly integrate and migrate existing spatial data results into a distributed data cluster environment.

A large amount of result data accumulated by traditional business platforms such as ArcGIS and Oracle can be quickly migrated to BiGeo for management and analysis through the toolkit. The new data generated by BiGeo after editing and analysis can also be reverted to the original platform. BiGeo can make full use of the advantages of the original platform combined with BiGeo's applicable scenarios for hybrid applications.

Various Internet map services such as Google Map, AMap, OpenStreetMap [50,67], and TianDiTu (IMap) can be directly connected to the BiGeo platform as reference data for a basemap service. Downloadable entity data, such as OSM data from OpenStreetMap [68–70], can also be opened, browsed, and queried directly in BiGeo. BiGeo can also publish managed data as a map service that complies with the OGC interface standard [29,71], and can mix and match these Internet map services.

3.4. Technological Choices

BiGeo can be deployed in an operating system in either a physical server or a virtual machine in a cloud environment. In a dense operation scenario with large amounts of data, the software is more suitable for deployment in a physical environment, which can reduce the loss in physical performance caused by virtualization. For large-scale deployments, system maintainability is more important than a performance loss and BiGeo should be deployed in a cloud environment. The experimental environment used in this study is deployed directly on physical devices. The hardware device environment consists of five blade servers, two high-performance workstations, and a notebook (this environment can also be scaled out as needed). The hardware environment used in BiGeo is as follows (Table 2):

Table 2. BiGeo hardware environment.

Data Cluster	Master	OS Centos 7.3.1661, CPU 8 core, Memory 8 GB, IP 10.51.60.30	
	Worker × 4	OS	Centos 7.3.1661
		CPU	E5-2660 Xeon 8 core 16 thread
		Memory	Samsung DDR3 16G RECC1600 × 4 = 64 GB
		Hard disk	Samsung 850 EVO 500 GB × 3
		Network	mainboard Integrated gigabit network card
		IP	10.51.60.21-24
Server	OS Windows Server 2008 R8, CPU Intel i7-7600 8 core, Memory 16 GB, IP 10.51.60.191		
Desktop & Toolkit	OS Windows 7 SP1, CPU Intel i5-2400 4 core, Memory 12 GB, IP 10.51.60.30.150		
Dev & Debug	OS Windows 7 SP1, CPU Intel i7-4500 4 core, Memory 8 GB		

BiGeo uses some open-source/independent libraries as a low-level functional support. They are basically C/C++/.Net libraries. Among them, the C/C++ library uses the .Net interface, leaving the .Net library as a native library or a migration library from Java or C/C++. In the choice of the basic support class library, the main consideration is performance and ease of use, while reducing the complexity brought about by a hybrid multilanguage development. The underlying class library basically uses the C and C++ class libraries to ensure a high performance. At the same time, the framework development level uses the C# interface uniformly, and through its perfect object-oriented features and interoperability with C/C++, a unified development technology stack is formed.

The internal development and operation environment of BiGeo is as follows: The data cluster is developed on the Linux (Centos 6.5+) platform, whereas other systems use Visual Studio 2010 and the

.Net 4.0 environment development on Windows (Windows 7 and Windows server 2008) platforms. Data debugging uses the SQL client Navicat or PgAdmin. In addition, the BiGeo kernel development model applies the C/C++/C# technology stack entirely. The resulting software is a green, stand-alone framework that no longer relies on any third-party commercial software environment.

BiGeo can be extended and developed in a variety of ways, from the outer to the inner layer: (1) For the framework development, an extended development based on the plug-in framework is provided by a desktop and server. This extended development model is suitable for new application system customization for meeting new independent business needs; (2) The component development is based on the GIS engine and relies on open-source libraries for extended development. This type of development is aimed at satisfying high customizable requirements, starting from the basic components provided by the BiGeo system, and building the upper application itself; (3) For the data development, the SQL interface of the data cluster is extended. This development is completely for the underlying statistical data analysis needs, customization, debugging, and optimization at the data level. These three development models are not independent of each other and are free to combine these three secondary development models to meet the diverse customization needs of users.

4. Implementation

4.1. Data Clustering

The geospatial big data distributed cluster solution is built using the open-source projects Greenplum and PostGIS [47]. Greenplum's architecture (Figure 2) uses massively parallel processing (MPP). Based on the x86 architecture and the PostgreSQL database, Greenplum developers use the MPP architecture to organize PostgreSQL instances in conjunction with the PostgreSQL community and application ecosystem, and implement storage and queries through the MPP backend [69]. In a nutshell, Greenplum is a distributed PostgreSQL database with feature compatibility and support for its geographic information extension, PostGIS.

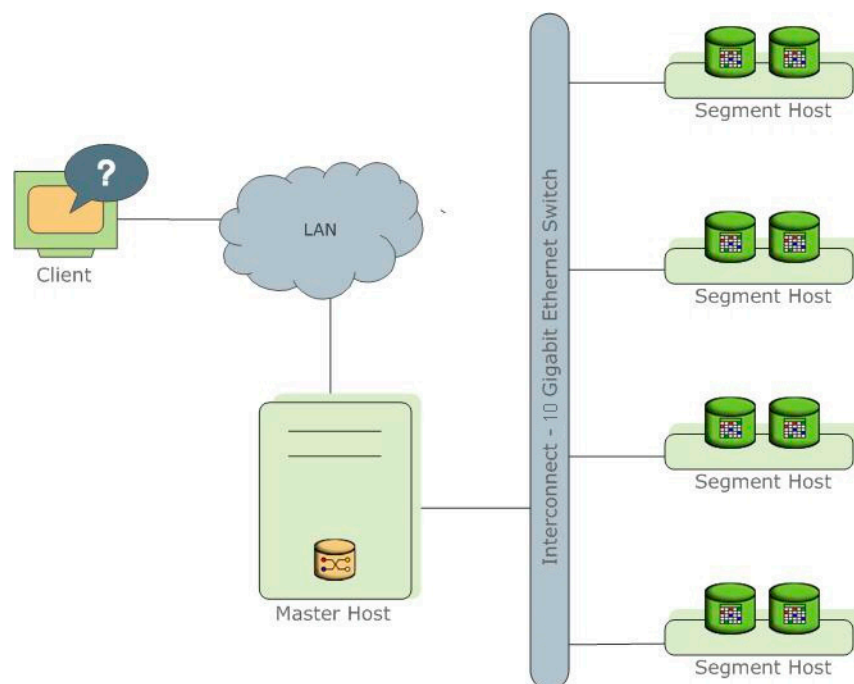


Figure 2. High-level Greenplum database architecture. Greenplum was a successful commercial OLAP product and is now open source; in addition, owing to the PostgreSQL project, it has complete support for spatial data features.

Greenplum supports the storage and processing of 50 PB of data. In the MPP system, each symmetric multiprocessor (SMP) node runs its own operating system and database. The CPU in each node cannot access the memory of another node. The information interaction between nodes is realized through the node interconnection network. This process is generally referred to as a data redistribution. Significantly different from the traditional SMP architecture, the MPP system is generally less efficient than SMP because it needs to transfer information between different processing units. However, this is not absolute, because the MPP system does not share resources, and thus has more resources than SMP. When the transaction to be processed reaches a certain scale, MPP becomes better than SMP based on the proportion of communication time occupied by the calculation time. If the communication time is relatively small, the MPP system can give full play to the advantages of the resources and achieve high efficiency.

The database consists of master servers and segment servers interconnected through an interconnect. The master is responsible for establishing a connection and management with the client, as well as the SQL parsing and formation of an execution plan. The execution plan distributes the task to the segment and collects the execution result of the segment. The master does not store business data, and only stores the data dictionary. The segment is responsible for the storage and access of business data and executes the SQL of the user.

We deployed 13 working segments in each of the four working server nodes in the cluster, plus mirror copies. The cluster has a total of $(13 + 13) \times 4 = 104$ working segments. Each working segment is equivalent to a 1.2 core CPU, and 2.4 GB of memory. In addition to the hard disk storage of the server, we can also use other storage devices to mount the extended cluster space. After the data cluster is set up, a method is used to create a new spatial database extension (we can also build multiple spatial databases on a cluster).

All data clusters have the geospatial capabilities of PostGIS by importing spatial extension functions. These spatial capabilities are provided as SQL functions in accordance with the OGC SQL-MM Part 3 specifications [72,73], including geometry/geography/box types, management functions (AddGeometryColumn, PostGIS_Full_Version, UpdateGeometrySRID, etc.), geometry constructors (ST_GeometryFromText, ST_GeomFromGML, ST_MakeEnvelope, etc.), geometry accessors (ST_Envelope, ST_GeometryType, ST_IsSimple, ST_SRID, ST_XMax, etc.), geometry editors (ST_AddPoint, ST_Force2D, ST_Reverse, ST_Transform, etc.), geometry outputs (ST_AsBinary, ST_AsText, etc.), operators (&&, @...), spatial relationships and measurements (ST_Contains, ST_Disjoint, ST_Relate, etc.), SFCGAL functions, geometry processing (ST_Buffer, ST_Difference, ST_Simplify, etc.), linear referencing, temporal support, long transaction support, miscellaneous functions (ST_EstimatedExtent, ST_Extent, etc.), and exceptional functions. These functions not only cover a full range of spatial characteristics, but also achieve horizontal scalability in a distributed cluster environment, which can be multiplied. The ability to call the geospatial data of a cluster can only be achieved through a simple spatial SQL statement interface.

In addition to directly using the shell tool to directly manage the monitoring data cluster state under Linux, we have also developed a WEB management monitoring program (Figure 3). A cluster state is converted into a rest interface through NodeJS by simply starting and stopping the cluster on the WEB side and viewing the running status of each node in the current cluster. Refer to <https://MasterIP:3000> to access the cluster status. We can also use the GPCC monitoring software that comes with Greenplum.

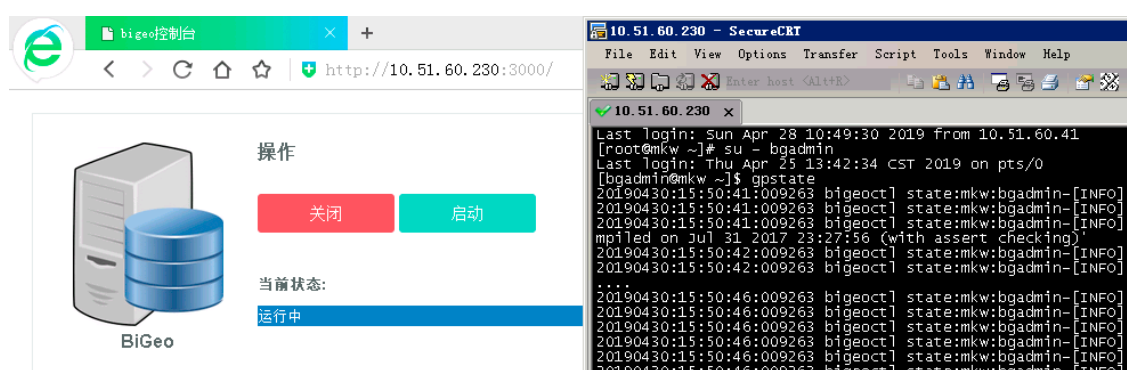


Figure 3. Data cluster management. Through the WEB interface, we can remotely start and stop the data cluster and monitor the status of the data nodes. With shell tools, we can also use the full GP tool for cluster maintenance.

4.2. GIS Engine

As a basic component, the GIS engine internally supports other software such as desktop and server software. The GIS engine is generally composed of three parts (Figure 4), namely, Map and coordinate conversion (transform, unit, pyramid, and MapConfig), layers and data sources (layers and their subclasses), and symbols (symbols and their subclasses). The GIS engine is responsible for the organization and visualization of maps in BiGeo and is a .Net component. The desktop and server access the GIS map capabilities by relying on the GIS engine interface.

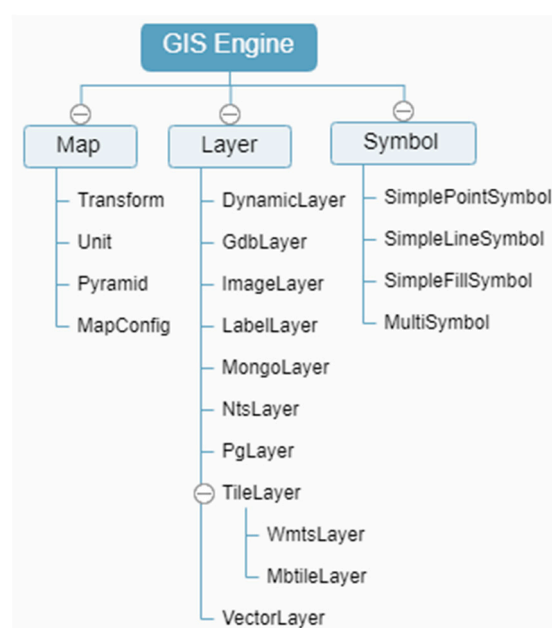


Figure 4. Geographic information system (GIS) engine structure. The name of the GIS engine is imagetile, which was derived from the research results of an actual project. By continuously accumulating support for different data sources and correcting the coordinate and symbol systems, a complete GIS kernel is formed.

As a layer container, Map manages the map data uniformly, and provides interfaces such as drawing, querying, blinking, zooming, and panning, and provides callback events such as drawing and a view transformation. A transform is responsible for the conversion of the map and device coordinates, and for the coordinated conversion of tile fixed grading and vector scales. A Pyramid is responsible for the conversion of geographic coordinates and tile numbers. Unit is responsible for the

conversion of different unit lengths including the latitude and longitude, meters, inches, and points. As the base class interface of the layer, Layer mainly agrees on the virtual interface functions such as drawing, layer range, data source, visibility, and visible scale.

We can use the GIS engine to quickly develop different applications. The following describes a demo example (Figure 5), where the data comes from the data cluster.

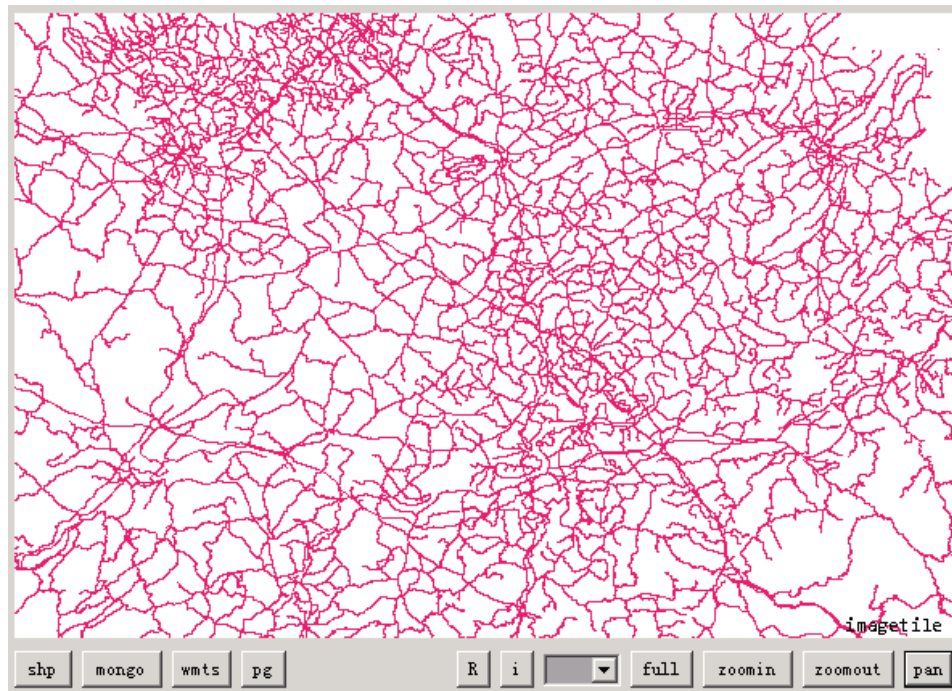


Figure 5. GIS engine sample program. This example embeds the GIS engine in a desktop application to quickly display the province’s water system data in the data cluster, allowing roaming, scaling, and query operations. We can also overlay other data sources, such as mongodb, shapefile, and map service.

Layers are composed of subclasses driven by different data sources, including but not limited to the following (Table 3):

Table 3. Layer type.

Layer Type	Layer Use
DynamicLayer	Implement this layer through inheritance to achieve dynamic feature effects
GdbLayer	Directly retrieve file-type GDB data sources through the FileGDB API
ImageLayer	Image layer, call remote sensing image data sources via Gdal
LabelLayer	Automatically label feature attributes based on automatic avoidance algorithms and annotation symbols
MbtileLayer	Retrieve the SQLite tile dataset that conforms to the mbtiles specification
MongoLayer	MongoDB layer, retrieve the vector data source stored in mongodb
NtsLayer	Nts layer, pure .net read shp file data source
PgLayer	Pg layer, call PostgreSQL/ Greenplum with PostGIS and vector data of similar data source systems via pg driver
TileLayer	Tile file layer, call local tile file data source that conforms to ArcGIS rules
VectorLayer	Vector layer, call shp, mdb, osm vector data source via ogr
WmtsLayer	Tile service layer, call tile service that conforms to the wmts specification

Symbol, as the base class interface of symbols, mainly defines virtual function interfaces such as drawing, color, offset, width, line type, and style (Table 4).

Table 4. Symbol type.

Symbol Type	Symbolic Use
SimplePointSymbol	Simple dot symbol, implemented in three types of parameter symbols: block, circle, and image
SimpleLineSymbol	Simple line symbol, implemented in three types of parameter symbols: solid, template, and image lines
SimpleFillSymbol	Simple area symbol, implemented in three types of parameter symbols: fill color, template pattern, and image pattern
MultiSymbol	Combination symbol, implemented through a combination of various symbols by layer to form complex symbol drawing

MapConfig can save map styles as JSON documents through saveMap. The map style is restored from the file through loadMap by using the map style in the JSON format. MapConfig has a clear structure and is easy to edit and exchange.

When the GIS engine calls the pg data source, it will automatically adopt the most appropriate rendering strategy according to different database server types. When using a normal single-node PostgreSQL database, a fast rendering strategy is used without a complex geometry processing. When Greenplum and BiGeo data clusters are applied, the complex polygons with islands and holes are graphically displayed, hidden, and covered, and the data in the range are batch-cut to obtain better graphic display effects and efficiency. The GIS engine also uses the read-only cursor dynamic read to reduce the memory usage of the massive data of the cluster, and implements a concurrent multithread dynamic rendering mechanism. This will ensure that the rendering data and the cluster call asynchronously and can be viewed and interrupted in real time.

4.3. Desktop

Desktop-side applications include data and functions including clusters for unified integration and expansion (Figure 6). This is suitable for scenarios in which the data are computationally intensive. The software is designed using a plug-in design pattern [74,75]. The functions of the software are stripped out of the framework. This reduces the complexity of the framework and makes the framework and individual modules easier to implement. The function and framework are then combined in a loosely coupled manner, and both can be independently changed and released while keeping the interface unchanged.

The software mainly consists of a plug-in framework, a plug-in contract, and a plug-in component: 1) The plugin framework is used to download, load, combine, instantiate, and destroy system plugins, providing a complete set of operating interfaces to communicate with the back-end framework; 2) The plugin contract exists in the form of an interface. All plug-ins of the system implement a unified interface specification for the framework. The framework effectively organizes and manages plug-in objects; 3) The plugin components are entities that complete the actual function, and implement the required plugin interface, such as the format conversion plug-in, graphics plug-in, and basic statistics plug-in.

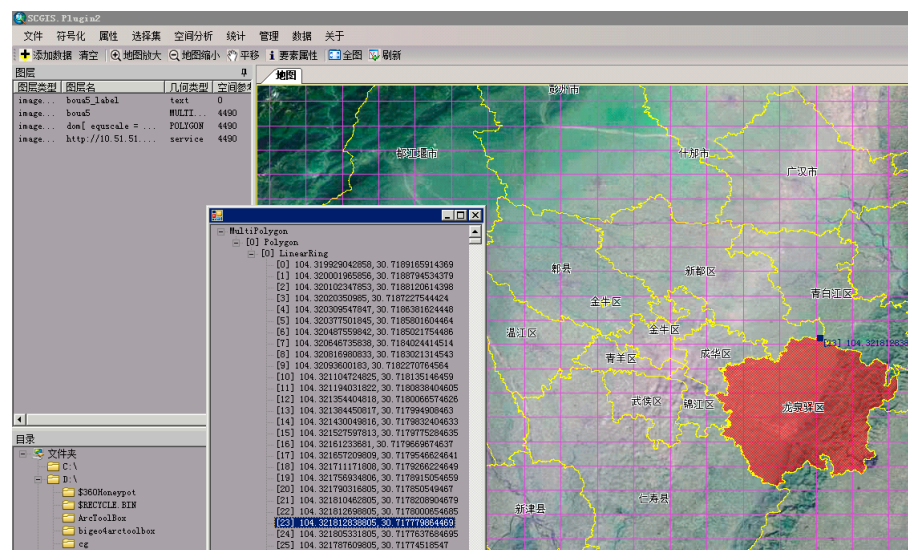


Figure 6. Desktop main interface. The main interface of Desktop includes menus, toolbars, status bars, layer windows, file directory windows, and map views. The current map shows an image of the province's service superimposed over administrative division vector data, and an administrative district selected to display and analyze its geometric structure.

The desktop uses plug-ins to extend the secondary development. All menus/tools are separate plug-in items. The following shows an example of the development of a print menu item after the cluster data map is visualized:

```
public void OnCreate(IApplication hook){
    if (hook != null) this.hk = hook; }
public void OnClick() {
    SaveFileDialog dlg = new SaveFileDialog();
    dlg.AddExtension = true; dlg.DefaultExt = ".jpg";
    dlg.Filter = "jpeg files (*.jpg)|*.jpg|All files (*.*)|*.*";
    if (dlg.ShowDialog() == DialogResult.OK) {
        string path = dlg.FileName;
        Bitmap bmp = new Bitmap(hk.PictureBoxMap.Width, hk.PictureBoxMap.Height);
        Graphics g = Graphics.FromImage(bmp);
        g.Clear(Color.White); hk.MapControl.Draw(g);
        bmp.Save(path); g.Dispose(); bmp.Dispose();
        MessageBox.Show(path); } }
```

As shown in this example, developing a distributed geographic data desktop function under the desktop framework is extremely simple.

The various spatial SQL interfaces of the distributed spatial database are transformed into an easy-to-use graphical interface, and the tasks that the data cluster cannot accomplish are supplemented by integrating various open-source libraries (Figure 7), such as NetTopologySuite and ProjNet (data editing, selection set interaction, etc.).

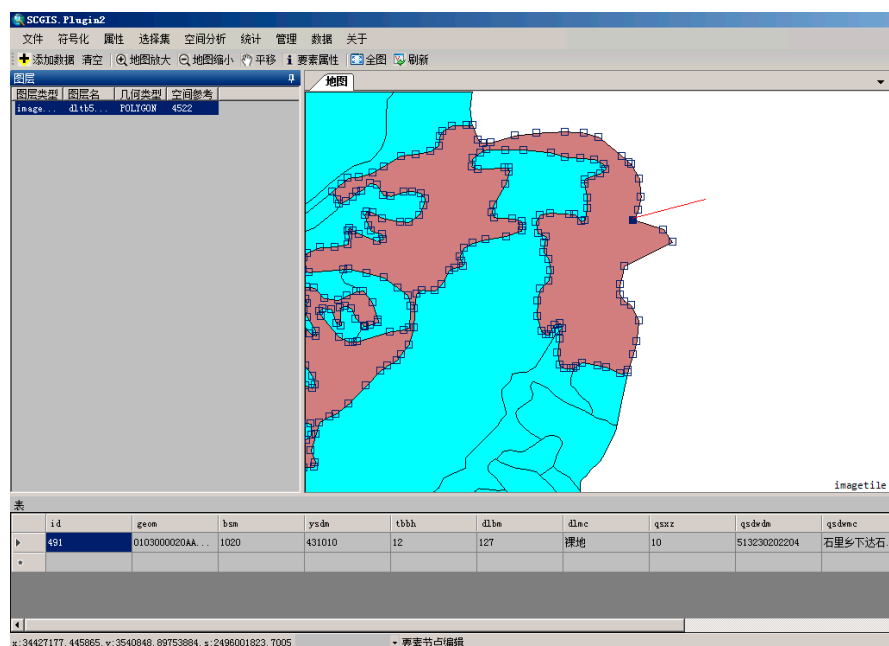


Figure 7. Selection of set interaction and node editing of cluster data. By integrating various GIS open-source libraries, we can further complete more GIS functions on the desktop side in the storage and computing capabilities of spatial data clusters.

4.4. Server

The server can conduct rapid publishing services, exposing back-end distributed spatial data and analysis capabilities, including clusters, to service interfaces. Service processing only needs to implement this minimum interface to access the service framework. This interface mainly allows the service framework to understand two aspects, namely, what tasks are assigned to this processor for processing (uri function), and how this processor handles such processing (process function). This interface also exposes various logic functions (Figure 8) such as conversion, aggregation, publishing, and testing to the client through a consistent interface (such as GET, POST, and WEBSOCKET [76]).

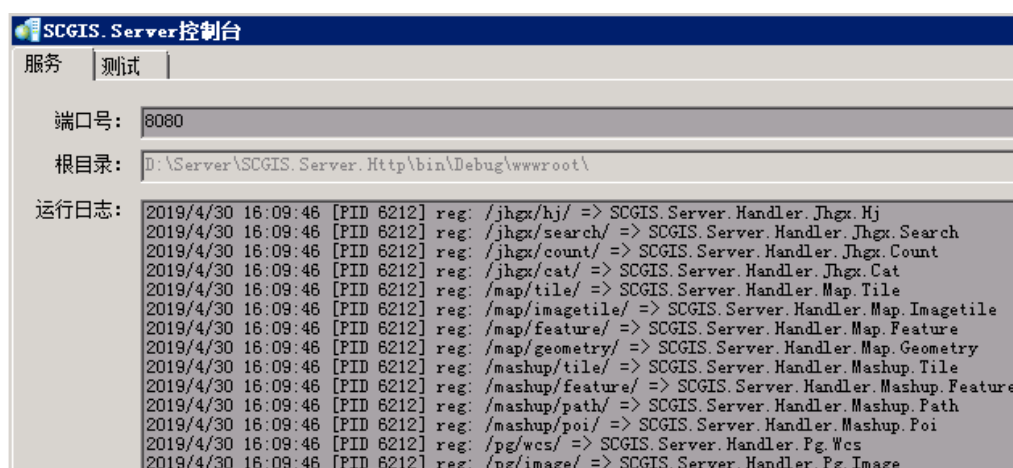


Figure 8. Server console. This is the server host. We can set the IP address, port, and resource root directory of the service; start, stop, browse, and test the service; and monitor the registered handler and its processing exception information. The actual service processing host is completed by another node program, which is an entry shell program.

The server side relies on .net 4.0 and starts the service host with SCGIS.Server.App.exe. The wwwroot folder is placed in the static publishing resource/script library/front end program.

The service host of the server handles different services through the processor handler extension, namely, implementing the interface of the IHandler, starting the server host, and automatically registering the service. This processor logic is automatically called when a corresponding service request comes in.

The map document is placed in the map folder for distributed on-the-fly dynamic map service publishing. The local file-type tile library can also be published as the same OGC WMTS service as the service conversion interface through the service processor “/map/tile/.” The dynamic map service uses the same WMTS interface as the tile service. Its data are not pregenerated, but are based on the data type, request rules, and rendering methods, which are dynamically generated by the server on demand. Support is given to the rapid publishing of numerous data services in the network. The calling interface is as follows:

<http://ServerIP:Port/map/tile/?layer=&tilematrix=&tilerow=&tilecol=&...>

Vector data are published as a query service for various applications. The service processor is “/map/feature/”, and the interface call format is as follows:

<http://ServerIP:Port/map/feature/?layer=&xmin=&xmax=&ymin=&ymax=&onlycount=>

The server provides some general geometric algorithm services to the client (Table 5), which are published by the “/map/geometry/” service processor for general geometry operations on the client side:

<http://ServerIP:Port/map/geometry/?type=&geom=&geom2=&sr=&sr2=&tolerance=>

The access interfaces all interact in the Well-Known Text (WKT) form. The table below shows the different geometry service types supported as well as the required parameters.

Remarks: geom, first geometry; geom2, second geometry; sr, first geometry’s spatial reference srid; sr2, second geometry’s spatial reference srid; tolerance, geometry for computing floating tolerance or double value; $\sqrt{\quad}$, required parameter; \bigcirc , optional parameter; \, no parameters.

A small amount of geometric computing services are provided by GeoAPI, OGR, and GEOS [45,77], and a massive data analysis uses data clusters for distributed computing. This means that we can port all features of the desktop to the server.

The following shows an example of a service processor that obtains access to the data cluster for the user token:

```
public class User2Token : IHandler{
    public void process(object ctx, Dictionary<string, object> param, string strUri, IServer server){
        string nm = param["name"] as string; string psw = param["password"] as string;
        string ip = NetUtil.clientIP(ctx); string token = Token.user2token(nm, psw, ip);
        Dictionary<string, object> dic = new Dictionary<string, object>();
        dic.Add("token", token); ResponseUtil.writeJson(ctx, dic); }
    public string uri() { return "/basic/user2token/"; } }
```

In the same way, we can quickly build a variety of distributed geographic information capabilities into a service interface through this framework. At present, the server implements more than 60+ core-based handler processors. These service interfaces support WEBSOCKET in addition to HTTP. The call interface in WEBSOCKET mode is particularly suitable for a long-term spatial analysis.

Servers can now be started in a cluster, deployed in parallel on multiple servers, or in multinode on a single server, similar to the Nginx mechanism [78]. We use JMeter for the concurrent testing. The surface performance can basically reach the same level as Web servers such as IIS and Nginx. The main impact of the server performance is the processor’s business processing logic and the back-end data cluster analysis and computing capabilities. The server can also serve as a gateway and service bus to support various microservice deployment modes, and the resource consumption is extremely small. As the most important aspect, its geospatial service expansion capability is excellent.

Table 5. Supported geometric spatial analysis types.

Type	Parameters					Output
	Geom	geom2	sr	sr2	Tolerance	
Area	√	\	○	\	\	Double
Boundary	√	\	○	\	\	WKT string
Envelope	√	\	○	\	\	Doubles
Buffer	√	\	○	\	√	WKT string
Centroid	√	\	○	\	\	WKT string
Contains	√	√	○	○	\	Bool
ConvexHull	√	\	○	\	\	WKT string
Crosses	√	√	○	○	\	Bool
Difference	√	√	○	○	\	Bool
Disjoint	√	√	○	○	\	Bool
Distance	√	√	○	○	\	Double
Equal	√	√	○	○	\	Bool
GML	√	\	○	\	\	GML string
JSON	√	\	○	\	\	JSON string
KML	√	\	○	\	\	KML string
Wkb	√	\	○	\	\	Byte
Intersect	√	√	○	○	\	Bool
Intersection	√	√	○	○	\	WKT string
Length	√	\	○	\	\	Double
Overlaps	√	√	○	○	\	Bool
PointOnSurface	√	√	○	○	\	WKT string
Simplify	√	\	○	\	√	WKT string
SymDifference	√	√	○	○	\	WKT string
Touches	√	√	○	○	\	Bool
TransformTo	√	○	√	√	\	WKT string
Union	√	√	○	○	\	WKT string
Within	√	√	○	○	\	Bool

4.5. Toolkit

The toolkit mainly conducts data sorting, cleaning, and migration, among other applications, and it can quickly convert and import a large amount of result data into a data cluster (Figure 9). The toolkit is implemented in a two-tier structure, which is the tool execution function and the tool parameter interface. The tool execution function is a separate minimum data processing unit that can be easily debugged and run through the command line mode. The tool parameter interface integrates the tool execution functions, visualizing the tool parameters and execution status.

The current toolkit implements the following tools: (1) an import tool, (2) an export tool, (3) a mapping conversion tool, (4) a full-text index tool, (5) a compressed tiles tool, (6) a data parallel cache tool, (7) a data automatic edge tool, (8) a data reference classification tool, (9) a data coordinate batch translation tool, (10) a nonspatial data automatic matching tool, (11) a track text conversion tool, (12) a POI data service download tool, (13) a grid statistical calculation tool, (14) a data directory editing tool, (15) a result data volume statistics tool, (16) a data incremental update tool, (17) a Word and Excel

information automatic extraction tool, (18) a symbol conversion tool, and (19) an automatic modeling tool, among others. We can continue to add new tools to the toolkit as needed.

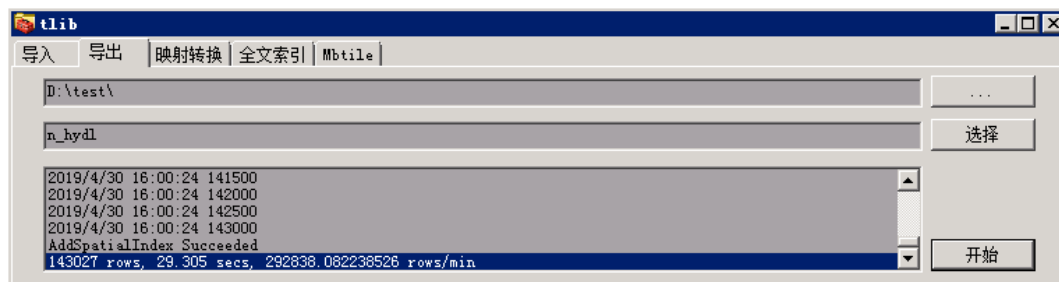


Figure 9. Bulk migration of data into a data cluster. Through this migration method, the historical data in ArcGIS and Oracle can be conveniently imported into BiGeo for analysis, and the analysis results are exported back to the original software for continued application.

5. Case Studies

Because BiGeo mainly provides the basic framework for geospatial big data management applications for regional organizations, we correspond with the main components of BiGeo to verify the storage, visualization, management, analysis, service, and migration effects, as well as efficiency, of geospatial big data. The largest and most complex set of datasets were selected for experimentation in the existing data from the Sichuan Basic Geographic Information Center.

The experimental data are from the Sichuan Province National Geographical Surface Coverage Dataset. The dataset has a total of 21,409,520 rows, and 60 GB of raw data are imported into data clusters of 200 GB. The data are distributed as indicated below (Figure 10).

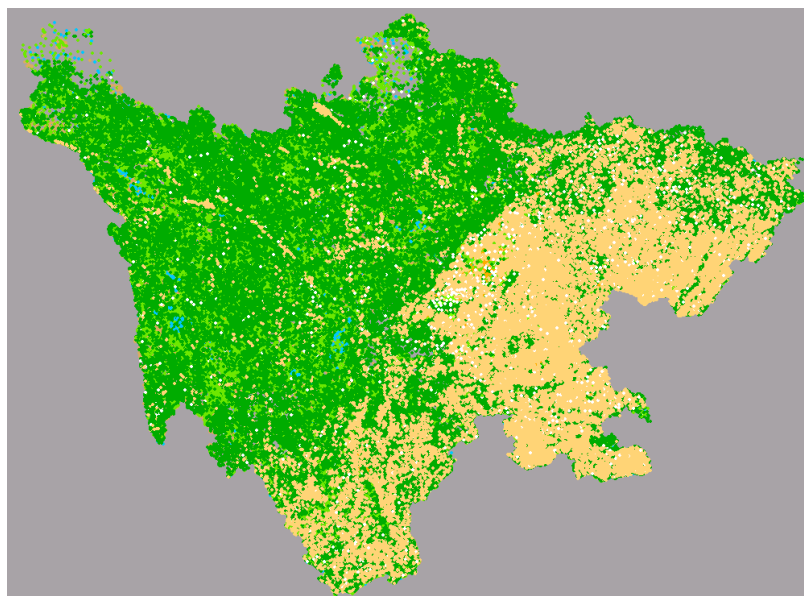


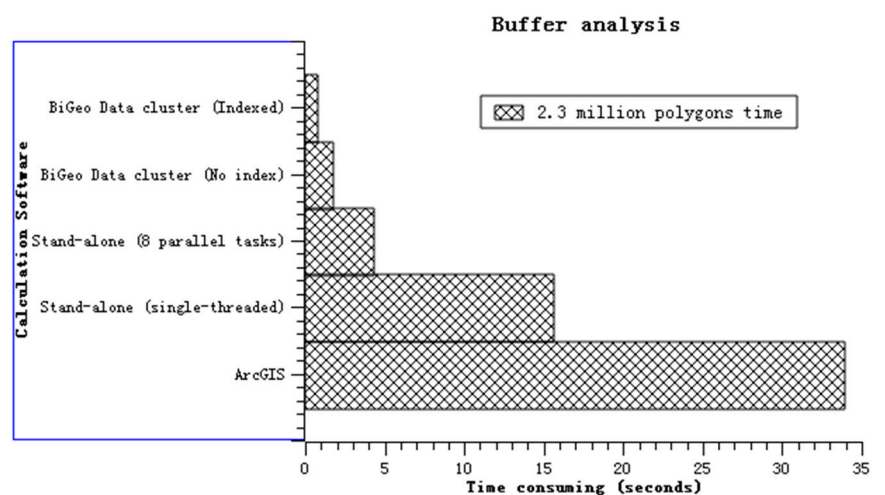
Figure 10. Overview of surface cover data in Sichuan province. The data are full coverage data. The vector data interpreted from the image data include all types of surface features in the province. The geometric complexity of the features exceeds the contour and vegetation features.

5.1. Case Study 1: Conducting Complex Spatial SQL Operations on Data Clusters

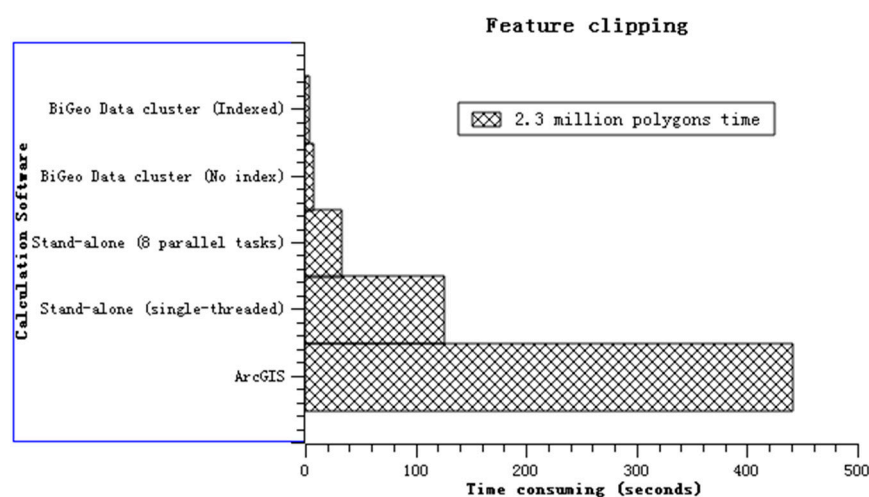
Different SQL queries are applied through the DB admin tools. The experiment results of various spatial analyses and statistics based on experiment data are listed below (Table 6).

We prepared five sets of environments for spatial analysis and comparison experiments (Figure 11). The five experiment environments are as follows: (1) ArcGIS software used for spatial analysis operations; (2) a stand-alone environment in which the data are placed into a singly deployed PostgreSQL database; (3) a stand-alone environment in which the data are placed into a PostgreSQL database deployed in-parallel with eight tasks; (4) BiGeo with spatial data without a spatial index; and (5) BiGeo with spatial data with an index. The five environments were used for buffer analysis (Figure 11a), feature clipping (Figure 11b), and area statistics (Figure 11c) to obtain comparison results of the efficiency.

As shown during the experiment, through the spatial SQL interface of the distributed data cluster, various underlying spatial queries, calculations, and analyses can be quickly conducted. Its ease of use and flexibility are far stronger than those of the current spatial data storage computing model based on Hadoop and Spark. Its computational efficiency is more than ten times that of traditional ArcGIS platforms. It can complete tasks that previously took minutes or even hours to complete on the second level, and even complete some geospatial calculations that could not be completed because of the huge amount of data involved.



(a)



(b)

Figure 11. Cont.

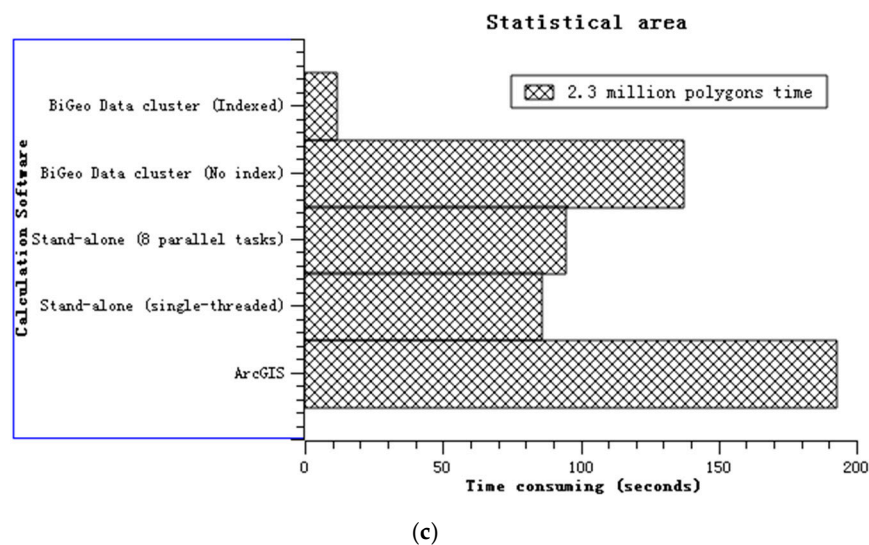


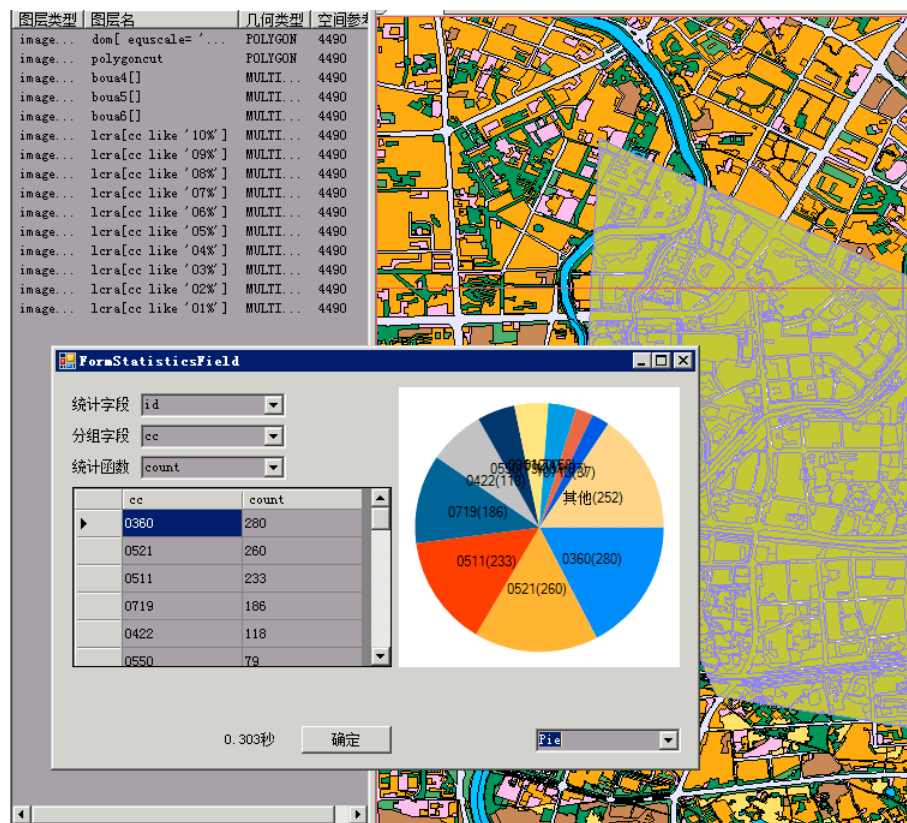
Figure 11. Comparison of different software spatial analysis efficiencies. The amount of data is 2.3 million, with 2,292,765 rows of water systems data for a 3100 km long and 20 km wide buffer area. (a) Buffer analysis (s). The stand-alone (single-threaded) CPU usage of ArcGIS is approximately 35%, the single-machine (eight parallel tasks) CPU usage is 100%. (b) Buffer clipping (s). ArcGIS uses the toolbox clip tool, and a single machine does not optimize within 636.7 s, single machine with eight parallel tasks does not optimize within 637.5 s (in this way, despite the eight parallel settings, single-threaded tasks are actually still conducted), and stand-alone ArcGIS (single thread CPU usage is approximately 35%, and the single-machine (with eight parallel tasks) CPU usage is 100%. (c) Statistical area with real-time calculation. The stand-alone machine (eight parallel tasks) is in fact single-threaded, with no parallelism. The precalculated area is calculated by first calculating the area as an attribute for each feature. The statistical area accumulates the area properties of each feature.

Table 6. Use clustering for SQL spatial analysis.

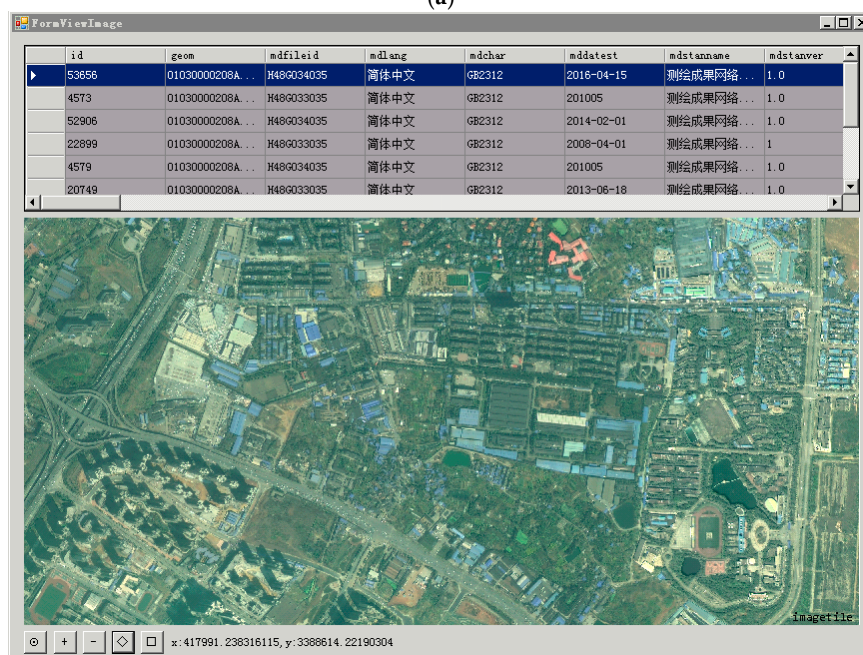
Analysis Type	Time (s)	Return Records	Description
count	0.709	21,409,520	
count by region	1.375	181	count the total number of results for each region by region code
within	0.834	518,137	only returned CC and id
cutting	29.285	521,316	use a polygon to cut and return the geometry
separate areas of all data	47.115	21,409,520	real-time dynamic projection in areas, returning a single area
sum of all data areas	44.054	1	including dynamic projection, statistical area; the result only returns the total
count attribute field	0.784	1	the shape_Leng value of the original data during data entry; the result returns only one total
total area after cutting	5.861	1	including cutting, dynamic projection, statistical area; the result returns the total area
count the area with the largest number of features	1.354	1	the number of records is 373,230
search for a point in which polygon	0.012	1	enter a point coordinate to determine which polygon the point falls in

5.2. Case Study 2: Spatial Analysis and Statistics of Geographic Information Data on the Desktop

The desktop is used for feature clipping and statistical analysis of the cluster (Figure 12a). Similarly, we can use the image map directory in the data cluster to call the massive image raster data in the file directory server (Figure 12b).



(a)



(b)

Figure 12. Desktop used for clips, statistics, and query data on the data cluster. (a) Spatial analysis and statistics of distributed massive vector data. There are many spatial analysis and statistical functions that can be used. These operations can also interact with each other for a more sophisticated analysis. (b) Query and management of distributed massive image data. Through the data cluster and file server, the images can be managed and queried in an integrated manner; in addition, the image entity files are not put into the database, and the original image directory does not need to be modified, and can be used directly.

Commonly used desktop GIS functions are basically covered (Figure 13), and the usage mode and traditional mode are consistent in a distributed environment.

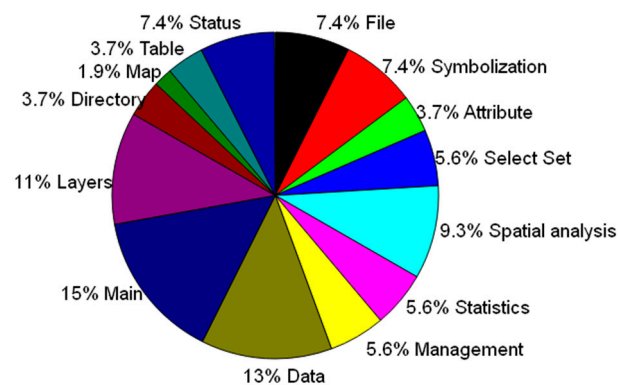
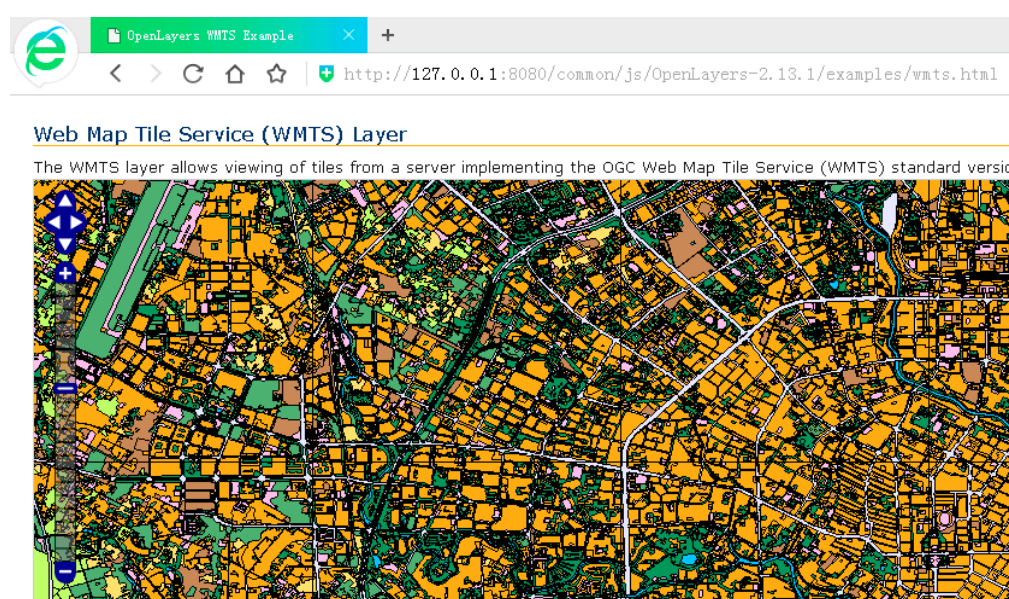


Figure 13. The distribution of various functions of the desktop. At present, the distribution of various functions is relatively balanced, covering almost all aspects of desktop GIS applications. All features are tailored to data clusters and enable a rapid integration of geospatial data in a distributed environment.

The desktop is a basic framework for common management and analysis tasks for spatial data in a distributed environment. It is also a highly customizable development platform that can be used to extend the desktop functionality at any time.

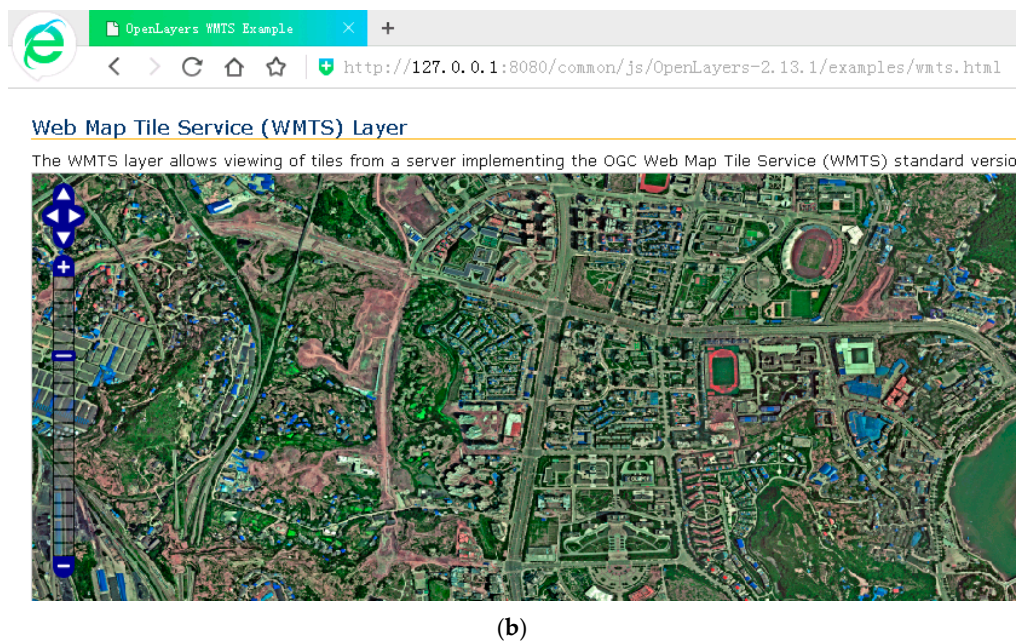
5.3. Case Study 3: Publish a Dynamic Map Service Using the Server

Map services can be quickly published using the above vector and physical image data. As shown in Figure 14, the on-the-fly dynamic rendering of map services can reach the second level (<3 s, with extremely complex features; <500 ms, at 10 GB per large single image file). This speed is a greater improvement over ArcServer and GeoServer. In traditional geographic information service software, this dynamic publishing service technology is basically impractical. With the help of distributed technologies such as data clustering, however, it is feasible to directly implement service publishing technology for physical data through the server. The direct use of physical data in a data cluster can meet the needs of geographical information service applications.



(a)

Figure 14. Cont.



(b)

Figure 14. The server can be used to quickly publish vector and image data on the data cluster (on-the-fly). (a) Viewing vector data in a data cluster published on the server side in real time at a small scale. The viewed data can be automatically cached, and thus the more browsing visits that occur, the more efficient the service response is. The initial server needs to request the data entity in the data cluster. (b) Visiting a global entity image of Dazhou City, Sichuan Province, released by the server. Unlike vector dynamic rendering, the efficiency of accessing physical images depends mainly on the IO speed of the network and the hard disk.

To further optimize the efficiency of this physical data dynamic service technology, we used the BiGeo framework to conduct a comparative experiment of different service delivery methods (Figure 15). The five service publish technologies applied are as follows: (1) dynamic vector rendering for directly and dynamically publishing vector data in a data cluster; (2) a vector cache tile for adding a server-side caching mechanism for dynamically published vector data services; (3) dynamic image rendering for directly and dynamically publishing image entity data services; (4) an image cache tile for adding a server-side caching mechanism to the dynamically published entity image service; (5) compressed tiles for further compressing the vector and image services added using the server-side caching mechanism; and (6) a front-end cache tile, in which the front-end cache response is further increased for services that have previously added a server-side caching mechanism. The service efficiency can be further improved through the mixed use of various combined service optimization technologies.

Based on the distributed capabilities of the data clusters and high-performance computing, the server can achieve a more in-depth display and analysis of the geospatial information.

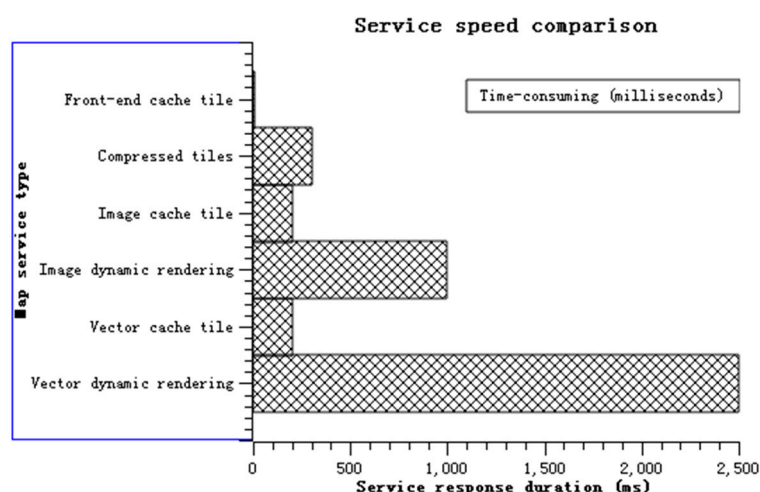


Figure 15. Comparison of response speeds of different map service types. As shown, vector dynamic rendering requires more CPU and memory, which takes the longest amount of time; in addition, image dynamic rendering mainly depends on the disk IO speed, the cached data service efficiency after access can be greatly improved, files can be cached through the client HTTP mechanism and greatly alleviate the server pressure, and the compressed data has better comprehensive space occupation, access speed, and cost performance.

5.4. Case Study 4: Quickly Migrating and Converting Large Amounts of Data Using the Toolkit

It takes about 3.5 h to use this tool to migrate the surface coverage data. The exporting of cluster data is twice as fast as the importing. Different types of data conversion and migration speed analyses are as follows (Table 7 and Figure 16):

Table 7. Import speed analysis.

Complexity	Feature Type	Average Speed	Bulk Packet Size
very complex features	surface coverage, contour lines, vegetation	80,000/min	packet 30 MB/1000
complex features	roads, water systems, administrative districts	150,000/min	packet 15 MB/1000
general features	house, ancillary facilities	300,000/min	packet 5 MB/1000
simple features	points of interest, joint maps, grids, metadata	600,000/min	packet 1 MB/1000

The packet size can also be automatically adjusted based on the data type. For example, each fixed transmission has a size of 2 MB, but with a different number of features. This can control the memory usage during data migration and maintain the stability of the network transmission.

Owing to the accumulation of historical data, the cleaning, integration, and migration of spatial data is a complicated and long process, and more practical tools need to be developed for specific problems. These tools are a prerequisite for ensuring that BiGeo is properly used. However, an application system based on the BiGeo framework has been completely established.

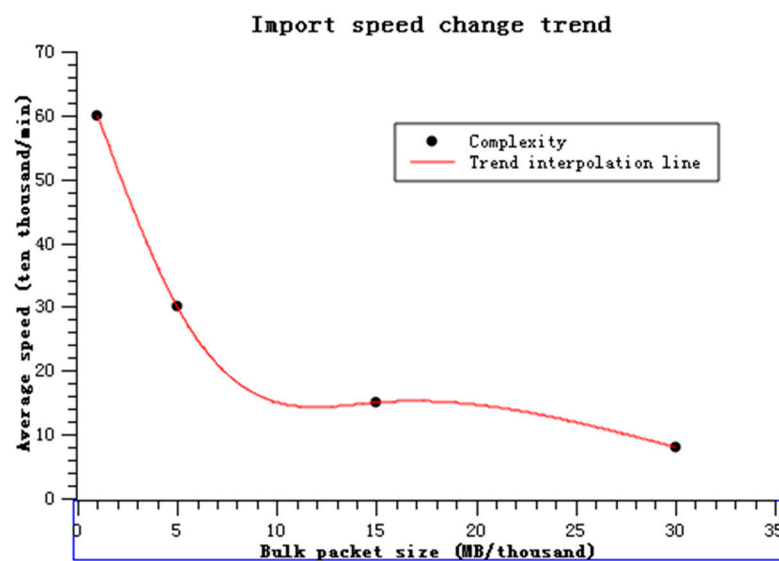


Figure 16. Trend interpolation line. As the size of the packet increases and the complexity of the features increases, the import speed will decrease significantly, and a slow decay rate will be maintained at an inflection point.

6. Conclusions and Future Work

Based on BiGeo's background and existing related technologies, this paper described the design of the overall architecture and framework of BiGeo according to the requirements and user scenarios. Through a concrete implementation and experimental cases, the effectiveness and efficiency of BiGeo were verified. Because BiGeo is a huge framework, after many years of iterative development and practical application, there remain numerous details that are difficult to fully discuss, and owing to limited space, are summarized as follows.

BiGeo has the following advantages: (1) It has a strong expandability. For a data cluster, adding nodes in the architecture can linearly increase the storage capacity and processing power of the system. At the same time, the operation is simple when a node is extended, and the data redistribution can be completed within a short time. Using the loosely coupled design of the GIS engine component, the desktop framework, and the server framework, there is plenty of room for an expansion of the functionality and performance; (2) BiGeo is easier to use, and provides a complete, rich call interface. Users only need to have traditional software development capabilities, and experience in distributed architecture development is not needed. Owing to the large number of simple development frameworks and components provided to developers, the complexity of the underlying distributed system development has been shielded to a certain extent. It is possible to quickly develop a large-scale GIS system based on a distributed architecture for massive spatial information processing, management, query, and service publishing; (3) It achieves highly cost-effective performance. Although the hardware equipment deployed by BiGeo in the experiment environment described herein cost only approximately USD \$110,000 using ordinary x86 servers, the performance under a huge amount of data achieved the expected results. In addition to a reduction in hardware investment, there is no need to charge a license for the database, basic GIS platform, or other software. Through this research, we have developed a BiGeo-based geospatial big data technology system, which will also be helpful for us to further reduce the related costs of technology management, hardware upgrades, and security.

BiGeo is not a substitute for traditional commercial software such as ArcGIS or Oracle but works closely with them to provide full access to their respective advantages. They can collectively form an application circulation of geospatial big data. BiGeo is also not a substitute for all types of open-source geospatial software. On the contrary, BiGeo absorbs a large amount of geospatial information and other computer open-source software projects. Through a unified and coordinated technology stack,

a complete geospatial big data PaaS layer solution can be formed. Therefore, BiGeo and existing technologies, applications, and platforms can be seamlessly connected. In addition, BiGeo is an open platform. In the future, with the development of additional big data technologies, it can be synchronized and rapidly expanded.

Thanks to the open-source community and academic circles, which have provided us with numerous research results, resources, and technologies, we have been able to further improve the use of BiGeo. At present, this technology has better applicability, practicability, and operability for geospatial big data when compared with other methods. However, there are still technical difficulties that need to be resolved.

The next step is to improve the following features: (1) The scale of the existing BiGeo experiment environment is still quite limited. We need to further observe the stability and efficiency of BiGeo after it has been scaled up, allowing it to be verified under more scenarios; (2) BiGeo currently focuses more on the technology implementation used in data management. The next step is to organize the logical organization of the data such that the data organization model can fully utilize the underlying capabilities of the BiGeo framework; (3) The back-end capabilities of BiGeo services lack a front-end performance and need to be further integrated with various front-end technologies [79] to enhance its overall application; (4) The tool side should allow new tools to be added according to the ETL data requirements [80,81]. Finally, (5) the frameworks and components of the call interface should be further optimized based on feedback from application system developers.

Author Contributions: Conceptualization, Methodology, Software and Writing-Original Draft Preparation, Xi Liu; Formal Analysis, Investigation, Resources and Editing, Lina Hao; Supervision and Project Administration, Wunian Yang.

Funding: This research was funded by National Natural Science Foundation of China, grant number 41702358, 41771444 and China Postdoctoral Science Foundation, grant number 2017M622982, Remote Sensing Science and Technology Research Innovation Team of Chengdu University of Technology, grant number KYTD201501.

Acknowledgments: Thanks to Chunlei Ren for his technical assistance and support in this study. Thanks to anonymous reviewers and editors for their contributions to improving this paper.

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

References

1. Siddiqua, A.; Karim, A.; Gani, A. Big data storage technologies: A survey. *Front. Inf. Technol. Electron. Eng.* **2017**, *18*, 1040–1070. [\[CrossRef\]](#)
2. Kamilaris, A.; Kartakoullis, A.; Prenafeta-Boldú, F.X. A review on the practice of big data analysis in agriculture. *Comput. Electron. Agric.* **2017**, *143*, 23–37. [\[CrossRef\]](#)
3. Leidig, M.; Teeuw, R. Free software: A review, in the context of disaster management. *Int. J. Appl. Earth Obs. Geoinf.* **2015**, *42*, 49–56. [\[CrossRef\]](#)
4. Jakimavičius, M.; Palevičius, V.; Antuchevičienė, J.; Karpavičius, T. Internet GIS-Based Multimodal Public Transport Trip Planning Information System for Travelers in Lithuania. *ISPRS Int. J. Geo-Inf.* **2019**, *8*, 319. [\[CrossRef\]](#)
5. Huang, Q.; Cervone, G.; Zhang, G. A cloud-enabled automatic disaster analysis system of multi-sourced data streams: An example synthesizing social media, remote sensing and Wikipedia data. *Comput. Environ. Urban Syst.* **2017**, *66*, 23–37. [\[CrossRef\]](#)
6. Sapountzi, A.; Psannis, K.E. Social networking data analysis tools & challenges. *Future Gener. Comput. Syst.* **2018**, *86*, 893–913.
7. Fritz, S.; McCallum, I.; Schill, C.; Perger, C.; See, L.; Schepaschenko, D.; van der Velde, M.; Kraxner, F.; Obersteiner, M. Geo-Wiki: An online platform for improving global land cover. *Environ. Model. Softw.* **2012**, *31*, 110–123. [\[CrossRef\]](#)
8. Pourebrahim, N.; Sultana, S.; Niakanlahiji, A.; Thill, J.-C. Trip distribution modeling with Twitter data. *Comput. Environ. Urban Syst.* **2019**, *77*, 101354. [\[CrossRef\]](#)

9. Dos Santos, R.F.; Boedihardjo, A.; Shah, S.; Chen, F.; Lu, C.T.; Ramakrishnan, N. The big data of violent events: Algorithms for association analysis using spatio-temporal storytelling. *GeoInformatica* **2016**, *20*, 879–921. [[CrossRef](#)]
10. Jiang, B. Geospatial analysis requires a different way of thinking: The problem of spatial heterogeneity. *GeoJournal* **2015**, *80*, 1–13. [[CrossRef](#)]
11. Chmielewski, S.; Samulowska, M.; Lupa, M.; Lee, D.; Zagajewski, B. Citizen science and WebGIS for outdoor advertisement visual pollution assessment. *Comput. Environ. Urban Syst.* **2018**, *67*, 97–109. [[CrossRef](#)]
12. Repetto, M.P.; Burlando, M.; Solari, G.; De Gaetano, P.; Pizzo, M.; Tizzi, M. A web-based GIS platform for the safe management and risk assessment of complex structural and infrastructural systems exposed to wind. *Adv. Eng. Softw.* **2018**, *117*, 29–45. [[CrossRef](#)]
13. Rafoss, T.; Sælid, K.; Sletten, A.; Gyland, L.F.; Engravslia, L. Open geospatial technology standards and their potential in plant pest risk management-GPS-enabled mobile phones utilising open geospatial technology standards Web Feature Service Transactions support the fighting of fire blight in Norway. *Comput. Electron. Agric.* **2010**, *74*, 336–340. [[CrossRef](#)]
14. Machwitz, M.; Hass, E.; Junk, J.; Udelhoven, T.; Schlerf, M. CropGIS—A web application for the spatial and temporal visualization of past, present and future crop biomass development. *Comput. Electron. Agric.* **2019**, *161*, 185–193. [[CrossRef](#)]
15. Kingdon, A.; Nayembil, M.L.; Richardson, A.E.; Smith, A.G. A geodata warehouse: Using denormalisation techniques as a tool for delivering spatially enabled integrated geological information to geologists. *Comput. Geosci.* **2016**, *96*, 87–97. [[CrossRef](#)]
16. Seo, B.C.; Keem, M.; Hammond, R.; Demir, I.; Krajewski, W.F. A pilot infrastructure for searching rainfall metadata and generating rainfall product using the big data of NEXRAD. *Environ. Model. Softw.* **2019**, *117*, 69–75. [[CrossRef](#)]
17. Ben Brahim, M.; Drira, W.; Filali, F.; Hamdi, N. Spatial data extension for Cassandra NoSQL database. *J. Big Data* **2016**, *3*, 11. [[CrossRef](#)]
18. Kwakkel, J.H.; Carley, S.; Chase, J.; Cunningham, S.W. Visualizing geo-spatial data in science, technology and innovation. *Technol. Forecast. Soc. Chang.* **2014**, *81*, 67–81. [[CrossRef](#)]
19. Zhang, X.; Yue, P.; Chen, Y.; Hu, L. An efficient dynamic volume rendering for large-scale meteorological data in a virtual globe. *Comput. Geosci.* **2019**, *126*, 1–8. [[CrossRef](#)]
20. Hardebol, N.J.; Bertotti, G. DigiFract: A software and data model implementation for flexible acquisition and processing of fracture data from outcrops. *Comput. Geosci.* **2013**, *54*, 326–336. [[CrossRef](#)]
21. Liao, C.; Brown, D.; Fei, D.; Long, X.; Chen, D.; Che, S. Big data-enabled social sensing in spatial analysis: Potentials and pitfalls. *Trans. GIS* **2018**, *22*, 1351–1371. [[CrossRef](#)]
22. Yu, J.J.; Qin, X.S.; Larsen, L.C.; Larsen, O.; Jayasooriya, A.; Shen, X.L. A GIS-based management and publication framework for data handling of numerical model results. *Adv. Eng. Softw.* **2012**, *45*, 360–369. [[CrossRef](#)]
23. Smith, D.A. Online interactive thematic mapping: Applications and techniques for socio-economic research. *Comput. Environ. Urban Syst.* **2016**, *57*, 106–117. [[CrossRef](#)]
24. Giuliani, G.; Nativi, S.; Lehmann, A.; Ray, N. WPS mediation: An approach to process geospatial data on different computing backends. *Comput. Geosci.* **2012**, *47*, 20–33. [[CrossRef](#)]
25. Moncrieff, S.; Turdukulov, U.; Gulland, E.K. Integrating geo web services for a user driven exploratory analysis. *ISPRS J. Photogramm. Remote Sens.* **2016**, *114*, 294–305. [[CrossRef](#)]
26. Zhao, L.; Liu, Z.; Mbachu, J. Highway alignment optimization: An integrated BIM and GIS approach. *ISPRS Int. J. Geo-Inf.* **2019**, *8*, 172. [[CrossRef](#)]
27. Huang, W.; Raza, S.A.; Mirzov, O.; Harrie, L. Assessment and Benchmarking of Spatially Enabled RDF Stores for the Next Generation of Spatial Data Infrastructure. *ISPRS Int. J. Geo-Inf.* **2019**, *8*, 310. [[CrossRef](#)]
28. Chen, P.; Shi, W. Measuring the spatial relationship information of multi-Layered vector data. *ISPRS Int. J. Geo-Inf.* **2018**, *7*, 88. [[CrossRef](#)]
29. Baumann, P. The OGC web coverage processing service (WCPS) standard. *GeoInformatica* **2010**, *14*, 447–479. [[CrossRef](#)]
30. Ludwig, B.; Coetzee, S. Implications of security mechanisms and Service Level Agreements (SLAs) of Platform as a Service (PaaS) clouds for geoprocessing services. *Appl. Geomat.* **2013**, *5*, 25–32. [[CrossRef](#)]

31. Tang, J.; Matyas, C.J. Arc4nix: A cross-platform geospatial analytical library for cluster and cloud computing. *Comput. Geosci.* **2018**, *111*, 159–166. [\[CrossRef\]](#)
32. Qin, R. Development of a GIS-based integrated framework for coastal seiches monitoring and forecasting: A North Jiangsu shoal case study. *Comput. Geosci.* **2017**, *103*, 70–79. [\[CrossRef\]](#)
33. Li, F.; Gui, Z.; Wu, H.; Gong, J.; Wang, Y.; Tian, S.; Zhang, J. Big enterprise registration data imputation: Supporting spatiotemporal analysis of industries in China. *Comput. Environ. Urban Syst.* **2018**, *70*, 9–23. [\[CrossRef\]](#)
34. Bellini, P.; Nesi, P. Performance assessment of RDF graph databases for smart city services. *J. Vis. Lang. Comput.* **2018**, *45*, 24–38. [\[CrossRef\]](#)
35. Huang, Z.; Chen, Y.; Wan, L.; Peng, X. GeoSpark SQL: An effective framework enabling spatial queries on spark. *ISPRS Int. J. Geo-Inf.* **2017**, *6*, 285. [\[CrossRef\]](#)
36. Qian, C.; Yi, C.; Cheng, C.; Pu, G.; Wei, X.; Zhang, H. Geosot-based spatiotemporal index of massive trajectory data. *ISPRS Int. J. Geo-Inf.* **2019**, *8*, 284. [\[CrossRef\]](#)
37. Jun, S.; Lee, S. Prototype system for geospatial data building-sharing developed by utilizing open source web technology. *Spat. Inf. Res.* **2017**, *25*, 725–733. [\[CrossRef\]](#)
38. Hu, F.; Xu, M.; Yang, J.; Liang, Y.; Cui, K.; Little, M.M.; Lynnes, C.S.; Duffy, D.Q.; Yang, C. Evaluating the Open Source Data Containers for Handling Big Geospatial Raster Data. *ISPRS Int. J. Geo-Inf.* **2018**, *7*, 144. [\[CrossRef\]](#)
39. Višnjevac, N.; Mihajlović, R.; Šoškić, M.; Cvijetinić, Ž.; Bajat, B. Prototype of the 3D cadastral system based on a NoSQL database and a Javascript visualization application. *ISPRS Int. J. Geo-Inf.* **2019**, *8*, 227. [\[CrossRef\]](#)
40. Lyu, L.; Xu, Q.; Lan, C.; Shi, Q.; Lu, W.; Zhou, Y.; Zhao, Y. Sino-inspace: A digital simulation platform for virtual space environments. *ISPRS Int. J. Geo-Inf.* **2018**, *7*, 373. [\[CrossRef\]](#)
41. Zaragozí, B.; Belda, A.; Linares, J.; Martínez-Pérez, J.E.; Navarro, J.T.; Esparza, J. A free and open source programming library for landscape metrics calculations. *Environ. Model. Softw.* **2012**, *31*, 131–140. [\[CrossRef\]](#)
42. Saif, S.; Wazir, S. Performance Analysis of Big Data and Cloud Computing Techniques: A Survey. *Procedia Comput. Sci.* **2018**, *132*, 118–127. [\[CrossRef\]](#)
43. Morsy, M.M.; Goodall, J.L.; O’Neil, G.L.; Sadler, J.M.; Voce, D.; Hassan, G.; Huxley, C. A cloud-based flood warning system for forecasting impacts to transportation infrastructure systems. *Environ. Model. Softw.* **2018**, *107*, 231–244. [\[CrossRef\]](#)
44. Blauth, D.A.; Ducati, J.R. A Web-based system for vineyards management, relating inventory data, vectors and images. *Comput. Electron. Agric.* **2010**, *71*, 182–188. [\[CrossRef\]](#)
45. Bunting, P.; Clewley, D.; Lucas, R.M.; Gillingham, S. The Remote Sensing and GIS Software Library (RSGISLib). *Comput. Geosci.* **2014**, *62*, 216–226. [\[CrossRef\]](#)
46. Appel, M.; Lahn, F.; Buytaert, W.; Pebesma, E. Open and scalable analytics of large Earth observation datasets: From scenes to multidimensional arrays using SciDB and GDAL. *ISPRS J. Photogramm. Remote Sens.* **2018**, *138*, 47–56. [\[CrossRef\]](#)
47. Haynes, D.; Manson, S.; Shook, E. Terra Populus’ architecture for integrated big geospatial services. *Trans. GIS* **2017**, *21*, 546–559. [\[CrossRef\]](#)
48. Meyer, D.; Riechert, M. Open source QGIS toolkit for the Advanced Research WRF modelling system. *Environ. Model. Softw.* **2019**, *112*, 166–178. [\[CrossRef\]](#)
49. Singh, S.K. Evaluating two freely available geocoding tools for geographical inconsistencies and geocoding errors. *Open Geospat. Data Softw. Stand.* **2017**, *2*, 11. [\[CrossRef\]](#)
50. Ballagh, L.M.; Raup, B.H.; Duerr, R.E.; Khalsa, S.J.S.; Helm, C.; Fowler, D.; Gupte, A. Representing scientific data sets in KML: Methods and challenges. *Comput. Geosci.* **2011**, *37*, 57–64. [\[CrossRef\]](#)
51. Saah, D.; Johnson, G.; Ashmall, B.; Tondapu, G.; Tenneson, K.; Patterson, M.; Poortinga, A.; Markert, K.; Quyen, N.H.; San Aung, K.; et al. Collect Earth: An online tool for systematic reference data collection in land cover and use applications. *Environ. Model. Softw.* **2019**, *118*, 166–171. [\[CrossRef\]](#)
52. Li, W.; Wu, S.; Song, M.; Zhou, X. A scalable cyberinfrastructure solution to support big data management and multivariate visualization of time-series sensor observation data. *Earth Sci. Inform.* **2016**, *9*, 449–464. [\[CrossRef\]](#)
53. Jo, J.; Lee, K.W. High-performance geospatial big data processing system based on MapReduce. *ISPRS Int. J. Geo-Inf.* **2018**, *7*, 399. [\[CrossRef\]](#)

54. Patterson, M.T.; Anderson, N.; Bennett, C.; Bruggemann, J.; Grossman, R.L.; Handy, M.; Ly, V.; Mandl, D.J.; Pederson, S.; Pivarski, J.; et al. The Matsu Wheel: A reanalysis framework for Earth satellite imagery in data commons. *Int. J. Data Sci. Anal.* **2017**, *4*, 251–264. [[CrossRef](#)]
55. Yu, J.; Zhang, Z.; Sarwat, M. Spatial data management in apache spark: The GeoSpark perspective and beyond. *Geoinformatica* **2019**, *23*, 37–78. [[CrossRef](#)]
56. García-García, F.; Corral, A.; Iribarne, L.; Vassilakopoulos, M.; Manolopoulos, Y. Efficient large-scale distance-based join queries in spatialhadoop. *Geoinformatica* **2018**, *22*, 171–209. [[CrossRef](#)]
57. Aji, A.; Wang, F.; Vo, H.; Lee, R.; Liu, Q.; Zhang, X.; Saltz, J. Hadoop-GIS: A High Performance Spatial Data Warehousing System over MapReduce. *Proc. VLDB Endow.* **2013**, *6*, 1009–1020. [[CrossRef](#)]
58. Alarabi, L.; Mokbel, M.F.; Musleh, M. ST-Hadoop: A MapReduce framework for spatio-temporal data. *Geoinformatica* **2018**, *22*, 785–813. [[CrossRef](#)]
59. Huang, W.; Zhang, W.; Zhang, D.; Meng, L. Elastic Spatial Query Processing in OpenStack Cloud Computing Environment for Time-Constraint Data Analysis. *ISPRS Int. J. Geo-Inf.* **2017**, *6*, 84. [[CrossRef](#)]
60. Nikitopoulos, P.; Vouros, G.A.; Vlachou, A.; Doukeridis, C. Parallel and scalable processing of spatio-temporal RDF queries using Spark. *Geoinformatica* **2019**, 1–31. [[CrossRef](#)]
61. Xia, J.; Yang, C.; Li, Q. Building a spatiotemporal index for Earth Observation Big Data. *Int. J. Appl. Earth Obs. Geoinf.* **2018**, *73*, 245–252. [[CrossRef](#)]
62. Mazzetti, P.; Roncella, R.; Mihon, D.; Bacu, V.; Lacroix, P.; Guigoz, Y.; Ray, N.; Giuliani, G.; Gorgan, D.; Nativi, S. Integration of data and computing infrastructures for earth science: An image mosaicking use-case. *Earth Sci. Inform.* **2016**, *9*, 325–342. [[CrossRef](#)]
63. Teruzzi, A.; Di Cerbo, P.; Cossarini, G.; Pascolo, E.; Salon, S. Parallel implementation of a data assimilation scheme for operational oceanography: The case of the MedBFM model system. *Comput. Geosci.* **2019**, *124*, 103–114. [[CrossRef](#)]
64. Zavala-Romero, O.; Ahmed, A.; Chassignet, E.P.; Zavala-Hidalgo, J.; Fernández Eguiarte, A.; Meyer-Baese, A. An open source Java web application to build self-contained web GIS sites. *Environ. Model. Softw.* **2014**, *62*, 210–220. [[CrossRef](#)]
65. Criollo, R.; Velasco, V.; Nardi, A.; Manuel de Vries, L.; Riera, C.; Scheiber, L.; Jurado, A.; Brouyère, S.; Pujades, E.; Rossetto, R.; et al. AkvaGIS: An open source tool for water quantity and quality management. *Comput. Geosci.* **2019**, *127*, 123–132. [[CrossRef](#)]
66. Rossetto, R.; De Filippis, G.; Borsi, I.; Foglia, L.; Cannata, M.; Criollo, R.; Vázquez-Suñé, E. Integrating free and open source tools and distributed modelling codes in GIS environment for data-based groundwater management. *Environ. Model. Softw.* **2018**, *107*, 210–230. [[CrossRef](#)]
67. Lin, W. Geoforum Volunteered Geographic Information constructions in a contested terrain: A case of OpenStreetMap in China. *Geoforum* **2018**, *89*, 73–82. [[CrossRef](#)]
68. Xie, Z.; Ye, X.; Zheng, Z.; Li, D.; Sun, L.; Li, R.; Benya, S. Modeling polycentric urbanization using multisource big geospatial data. *Remote Sens.* **2019**, *11*, 310. [[CrossRef](#)]
69. Galić, Z.; Mešković, E.; Osmanović, D. Distributed processing of big mobility data as spatio-temporal data streams. *Geoinformatica* **2017**, *21*, 263–291. [[CrossRef](#)]
70. Kulawiak, M.; Dawidowicz, A.; Pacholczyk, M.E. Analysis of server-side and client-side Web-GIS data processing methods on the example of JTS and JSTS using open data from OSM and geoportal. *Comput. Geosci.* **2019**, *129*, 26–37. [[CrossRef](#)]
71. Amirian, P.; Alesheikh, A.A.; Bassiri, A. Standards-based, interoperable services for accessing urban services data for the city of Tehran. *Comput. Environ. Urban Syst.* **2010**, *34*, 309–321. [[CrossRef](#)]
72. Ma, X. Linked Geoscience Data in practice: Where W3C standards meet domain knowledge, data visualization and OGC standards. *Earth Sci. Inform.* **2017**, *10*, 429–441. [[CrossRef](#)]
73. Song, J.; Di, L. Near-real-time OGC catalogue service for geoscience big data. *ISPRS Int. J. Geo-Inf.* **2017**, *6*, 337. [[CrossRef](#)]
74. Horsburgh, J.S.; Reeder, S.L. Data visualization and analysis within a Hydrologic Information System: Integrating with the R statistical computing environment. *Environ. Model. Softw.* **2014**, *52*, 51–61. [[CrossRef](#)]
75. Ames, D.P.; Horsburgh, J.S.; Cao, Y.; Kadlec, J.; Whiteaker, T.; Valentine, D. HydroDesktop: Web services-based software for hydrologic data discovery, download, visualization, and analysis. *Environ. Model. Softw.* **2012**, *37*, 146–156. [[CrossRef](#)]

76. Gao, F.; Yue, P.; Zhang, C.; Wang, M. Coupling components and services for integrated environmental modelling. *Environ. Model. Softw.* **2019**, *118*, 14–22. [[CrossRef](#)]
77. Lucas, G.; Lénárt, C.; Solymosi, J. Development and testing of geo-processing models for the automatic generation of remediation plan and navigation data to use in industrial disaster remediation. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. ISPRS Arch.* **2015**, *40*, 195–201. [[CrossRef](#)]
78. Li, R.; Dong, G.; Jiang, J.; Wu, H.; Yang, N.; Chen, W. Self-adaptive load-balancing strategy based on a time series pattern for concurrent user access on Web map service. *Comput. Geosci.* **2019**, *131*, 60–69. [[CrossRef](#)]
79. Eirinaki, M.; Dhar, S.; Mathur, S.; Kaley, A.; Patel, A.; Joshi, A.; Shah, D. A building permit system for smart cities: A cloud-based framework. *Comput. Environ. Urban Syst.* **2018**, *70*, 175–188. [[CrossRef](#)]
80. Boulekrouche, B.; Jabeur, N.; Alimazighi, Z. Toward integrating grid and cloud-based concepts for an enhanced deployment of spatial data warehouses in cyber-physical system applications. *J. Ambient Intell. Humaniz. Comput.* **2016**, *7*, 475–487. [[CrossRef](#)]
81. Bimonte, S.; Boucelma, O.; Machabert, O.; Sellami, S. A new Spatial OLAP approach for the analysis of Volunteered Geographic Information. *Comput. Environ. Urban Syst.* **2014**, *48*, 111–123. [[CrossRef](#)]



© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).