Building data infrastructure for geo-computation by spatial information grid

ZhenChun Huang\(^1\), GuoQing Li\(^2\), Yi Zeng\(^2\)

\(^1\) Department of Computer Science and Engineering, Tsinghua University  
Beijing, China  
Email: huangzc@tsinghua.edu.cn

\(^2\) China Remote Sensing Satellite Ground Station  
Beijing, China  
Email: \{gqli, yzeng\}@ne.rsgs.ac.cn

Abstract: The continuing research on geo-computation purchases more orders to the data infrastructure of geo-computation. It should be distributed, platform-independent, extensible, robust, autonomic, and can on-stop serve the users and applications. In this paper, based on spatial information grid (shortly SIG), a 4-layered data infrastructure which is built up by data nodes, data sources, data agencies, support libraries, and other components is proposed. It adopts an XML based extensible data accessing language as the operating protocol for the data sources and data agencies. Furthermore, some primary data sources and data agencies are designed and implemented; and based on them, a prototype of the data infrastructure is constructed and tested. The test shows that the SIG based data infrastructure can provide different sources, types and formats of spatial information data and make it easier to build up application for geo-computation.

Keywords: Data infrastructure, Spatial Information Grid.

1. Introduction

In the last decades, the continuing research on geo-computation gives the world a lot of advantages in economy and society. Today, the research interest of geo-computation such as data mining and knowledge discovery purchases more orders to the data infrastructure of geo-computation. For instance, data with different format, different source, and different goal may be processed and analyzed together so that the knowledge can be discovered. Because of some connatural limit, it is too difficult for the traditional data infrastructure based on database technology to response the orders. Based on the presentation-oriented HTML language, the conventional data infrastructure such as GCMD Mapserver [1] or CEOP [2] is friendlier to human. Without a well-defined accessing language, it is not suitable for automatic geo-computation applications. Although the WCS[3]/WFS[4]/WMS[5] purposed by OGC try to make it possible for geo-computation applications to access heterogeneous data through a universal interface, it is still hard to be used because of the absent of data searching capability and difficulty of complex programming. A new data infrastructure which is distributed, extensible and platform-independent is needed to provide more powerful and flexible data services for the geo-computation research and applications.

Grid computing is a new research agenda which evolves from the distributed computing and meta-computing. It tries to provide virtual computation resources by strip the power of resources from the computer hardware and software. When the grid computing technology is adopt in the research domain of spatial information and geo-computation, Spatial Information Grid [6] (shortly SIG) is proposed and studied. In SIG, the computing power, data, model, arithmetic, and other resources are shared and assembled as abstract resource through a series of middleware, toolkits and infrastructure. It will be a powerful and easy-to-use infrastructure for spatial information applications.

Based on SIG, a new data infrastructure for geo-computation can be designed and implemented. It is powerful and flexible, and can response the orders of geo-computation. It is:

- Distributed: the SIG based data infrastructure is distributed in geography and management. The data infrastructure has a lot of service nodes distributed geographically, and the nodes are managed distributed by their owners instead of a centralized organization.
- Platform-independent: the SIG based data infrastructure are well designed by XML schema for cooperation and implemented in java language, so that it can run on almost any platform and supports any type of data sources.
- Extensible: the SIG based data infrastructure can be extended in function and data set by adding new data service nodes and data agencies. When the new nodes are connected and registered, it can serve for the users immediately. Furthermore, new data service nodes and data agencies are easy to be developed by toolkits provided by SIG.
- Robust: because it is distributed and non-centralized, the SIG based data infrastructure can keep serving with some functional and performance decrease only when one or more of its nodes are failed.
- Autonomic: by the support of SIG runtime environment, the SIG based data infrastructure can self-manage, self-heal, and self-optimize naturally. It can manage the users and data through SIG management functions; it can ignore the failed nodes to keep the infrastructure from...
crashing; and it can adjust the data access policies of agencies to optimize the performance for users, too.

• On-stop serving: the SIG based data infrastructure is not a gallimaufry of pieces, but an organic whole. All resources, including data, services and agencies, are assembled well and a one-stop data service is provided based on them. User can access the infrastructure through any entrance (it may be a user interface, a programming interface, or other entrance) and get all resources on his finger.

The new data infrastructure based on SIG can organize the distributed data nodes and data agencies dynamically; build an extensible, robust, and autonomic data infrastructure; and serve the users on-stop as an organic whole. In this paper, the architecture of the infrastructure is proposed, the framework and co-operation protocols are designed, some important parts of the infrastructure are implemented by java language, and the implemented data infrastructure is tested through a web-based user interface.

2. Architecture

In the last years, by the announcement of OGSA [7] and WSRF [8], researchers and constructors of grid tried to combine grid technology with web service and adopt the service oriented architecture (SOA) which is very popular as its basic architecture. So, the SIG based data infrastructure also adopts the SOA as its footstone, and regards all components including data sources and data agencies as web services. In logic, it takes a 4-layered architecture show as figure 1.

The data infrastructure shares spatial data in different types, with different formats, and for different goals in a uniform infrastructure. Because the data are often stored on different platforms, the data sources have to be invoked by a platform-independent protocol such as which has a uniform service accessing point. The service accessing point is a web service which is described by a WSDL [9] document and invoked through SOAP [10] protocol. By invoking the web service in a well-defined XML based protocol; data stored in the data node can be searched and accessed.

Although users and applications can search and download data from the data sources directly, it is too deficient for the SIG based data infrastructure which is built up by data nodes only to be an ideal data infrastructure for geo-computation. The third layer – agency layer is the most important layer to make the data infrastructure flexible, extensible, autonomic, and powerful. In this layer, there are a series of data agencies with different goal and different functions serving for the users of the data infrastructure. For instance, a “catalogue agency” may collects information from data sources, generates a “catalogue” of data in the infrastructure, and provides a service for users to find data with some given features in the “catalogue”. It is a “search engine” like “google” in the data infrastructure, and provides more powerful functions for the data infrastructure.

In order to make the design of data infrastructure simple and neat, the data agencies are required to adopt the same protocol as the data sources. It is called “eXtensible Data Accessing Language”, shortly XDAL, and will be proposed detailedly in the next section.

The top layer of the data infrastructure is “Application layer”, where there are a lot of applications who are using the data infrastructure and solving real-life problems. One of the applications is the user interface of data distribution. Users can search, browse, and download data from the nodes in data infrastructure by the user interface application.

For the sake making development of applications easier and faster, some client support libraries are designed and implemented in the application layer. These libraries provide some powerful functions based on data sources and data agencies for the application developers to develop applications based on the data infrastructure easier and faster.

In the SIG based data infrastructure, nodes, data sources, agencies, and other components are registered into the registry service [11] provided by SIG so that they can be found by their users. The registry service is a part of SIG run-time environment and does not belong to any layer. If there is no registry service provided, a UDDI [12] registry center can play the role instead.

3. the eXtensible Data Accessing Language

The data infrastructure shares spatial data in different types, with different formats, and for different goals in a uniform infrastructure. Because the data are often stored on different platforms, the data sources have to be invoked by a platform-independent protocol such as
SOAP. Furthermore, a well-defined extensible data accessing language which suits any data source and any data type is needed for the data source accessing based on SOAP. In order to make it platform-independent, XML should be adopted as its format.

There are several frequently used operations on a data source or data agency: searching data, downloading data, and querying its capability description. The grammars and usage rules of request and response for these operations are standardized by the well-defined XDAL in XML schema. Users can accomplish the operation by invoking the web service provided by data source or data agency, passing the request to it, and analyzing the response for the result.

In the XDAL, a REQUEST should have a root element named <query>, <access>, <getCapability>, <getStatus>, or <getResult>, for functions of searching data, downloading data, getting capability description of data source, and for commands of getting operation status, getting operation result. A RESPONSE should have a root element named <response>, <status>, or <result>, for responses of starting an operation, getting operation status, and getting operation result. An access to the data source (for searching data, downloading data, or getting its capability description) should be accomplished in three steps: starting the operation by sending an operation request to the data source, monitoring the operation status for its finish, and getting the operation result. First, the client generates a request with root element <query>, <access> or <getCapability>, and pass it to the data source as a string parameter of SOAP interface. When the request is received, the data source analyzes it, starts the operation contained by the request (such as searching data, preparing the data being downloaded), assigns a unique identification to create a response and sends the response to the client for getting operation status or getting result. Then, the client monitors the operation status by request with root element <getStatus> until the operation finishes. At last, the result is required by request with root element <getResult> as result sorting, maximum result count, etc. And the result got by the last step will be a list of found image, with some key features such as satellite ID, sensor type, region covered, and data acquisition time, so that the client can select the image to be used among them further.

The requests for downloading data and for getting capability description of data source are almost the same as them for searching data. The only differences are the request sent in the first step and the result returned in the last step. In order to downloading data, a request with a root element <access>, which contains sub-elements such as ID list of data to be downloaded and detail of accessing operation, should be sent to the data source. The detail of accessing operation will be described as a “process chain”, which can be executed for processing and packing the data, and returning to the user a series of URLs can be accessed by user as the <dataURL> sub-elements of <result> root-element in the last step. Now, the processes in the process chain can be: all data, data of some wave bands, quick look and meta-data, data of a sub-region, and the legal combination of these processes. In order to get the capability description of a data source, the request sent in the first step should have a root element named <getCapability> with sub-elements named <parameterName> which means name of the feature required. And a list of data

Fig. 2. the timing sequence of the three steps for accessing a data source

Fig. 3. sample of searching data

1. client sends the request of searching an image

   <query>
   <conditions relation="AND">
     <condition op="EQ">
       <param>satellite</param>
       <value>landset7</value>
     </condition>
   </conditions>
   <sortBy order="ASC">
     <param>date</param>
     <param>satellite</param>
     <param>time</param>
   </sortBy>
   </query>

2. data source starts the operation, and returns a operation ID

   <response>
   <operationID>001235864631</operationID>
   </response>

3. client gets the operation status

   <getStatus>
   <operationID>001235864631</operationID>
   </getStatus>

4. data source returns the operation status

   <status>
   <operationID>001235864631</operationID>
   <currentStatus>processing</currentStatus>
   </status>
   <!-- processing -->

5. client gets the operation result

   <getResult>
   <operationID>001235864631</operationID>
   <resultSet>
   <item>
     <id>0013256782</id>
     <spacecraft>landset7</spacecraft>
     <sensor>ETM+</sensor>
   </item>
   <item>
     <id>0020165784</id>
   </item>
   </resultSet>
   </getResult>

6. data source returns the operation result

   <result>
   <operationID>001235864631</operationID>
   <resultSet>
   <item>
     <id>0013256782</id>
     <spacecraft>landset7</spacecraft>
     <sensor>ETM+</sensor>
   </item>
   <item>
     <id>0020165784</id>
   </item>
   </resultSet>
   </result>
source features will be returned in the last step.

Considering the extensibility of the system, the XDAL is designed as a XML based extensible language. Both the request of <query>, <access>, <getCapability>, and the response of <result> can be extended a lot. By extending the request and response, the XDAL will be suit for almost all kinds of data sources and geo-computation applications.

4. Design and implementation of data nodes

Because the registry service of Spatial Information Grid is referenced directly and need not be implemented again, there are four parts to be designed and implemented for the prototype of data infrastructure: the data sources/nodes, the data agencies, the support libraries, and the web-based user interface. In this section, some data nodes will be designed and implemented; and the other parts will be designed and implemented in the later sections.

Java language is often adopted by the grid implementation because of its platform-independent, acceptable performance and mass of open-source support. So the prototype of SIG based data infrastructure is also implemented by Java language. And the test bed will be built on the well-known Servlet/JSP server Apache Tomcat [13] with axis 1.2 [14].

In the SIG based data infrastructure, a data node is such a computer system which stores spatial data, provides a service of data source, and shares the data through the data source service accessing point. So, if the data source is designed and implemented, the data node is accomplished successfully.

Because the spatial data is far different in type, format, and organization, the implementation of data source must be very flexible. The framework of “plug-in with configuration” is a good choice. In the data sources proposed by this paper, a plug-in should be a Java class and configured by the configuration file for the class name and invoking parameter of plug-in java class. Figure 4 is the implementation of data source.

In the implementation of data source, a “operation” is implemented as a java class “Operation”, which includes the operation ID, the process chain for the operation with processes should be done (sub-class of java class ProcessAdapter), current status of the operation, current data being processed (sub-class of java class AbstractData), and a java thread who is executing the process chain. All sub-class of ProcessAdapter must overload the thread named process() to process on the current data saved in class Operation. Data sources with different types and goals are different at their process chains. The sub-classes of ProcessAdapter used by them should be configured to assemble a process chain, which will be executed in the java thread stored in Operation class to process the data in class Operation and handle the status. The class DataSourceEngine plays the role of the data source manager, converts the client request provided by class DataSourceService to instance of Operation, creates, destroys, and managers the instance of Operation, and returns the response of Operation to client through class DataSourceService, so that the sequence shown in Figure 2 and Figure 3 can be accomplished.

When the data source is implemented, deployed, and registered into the resource registry service, it can serve for the users. Now there are eight data sources implemented and deployed. These data sources share data with format MrSid, GEOTIFF, HDF, DEM, and JP2; from satellite Landset, Envisat, TERRA, AQUA, etc.; and for remote sensing, GIS and GPS applications. The continuing joined data sources provide powerful data supports for the data infrastructure. Because they are far different in goals and functions, the data sources should be implemented, deployed, and registered to raise the capability of system step by step.

5. Design and implementation of data agencies

Besides the data clients and the user interfaces, the data agencies will access the data sources, too. Data agencies will enlarge the capability of system, raise the productivity of system, and make the system more easy-to-use. A data agency named Spatial Data Catalogue Agency which can collect, organize and index the information of data is designed and implemented now. It provides function of data searching for users by collecting information from a series of data sources and re-organizing, storing and indexing the information. Users can search data from the Spatial Data Catalogue Agency to get its information, and access the data from data source.

As figure 5 shows, the data catalogue agency is separated into two parts: the back end to collect, re-organize, index the information from data sources and store them to database, and the front end to respond the request of users and searching data from database. They share a database to transfer information from the back end to the front end.

The implementation of the agency is simple. It is built up by several java classes such as DCService for service...
interface, DataSearcher for searching data from data base, DataCollector for data collecting and DataAnalyzer for analyzing data and storing them into database. The database stores the information collected and bridges the back end and front end.

The data catalogue agency is only a simplest data agency in the SIG based data infrastructure for geo-computation. Agencies with more powerful functions can be developed and deployed into the infrastructure. For example, a “merging agency” may collect some remote sensing image data with different resolutions and different wave bands and merge them to one image, process the image for some target features, add some GIS layers on the image, and generate a thematic map for the user. The continuing joined data agencies will make the data infrastructure more powerful. For instance, a well-designed agency for indexing data in the infrastructure can indicate which data source is faster and more useful for user, ignores the failed data nodes and find available nodes instead, filter out the nodes who provides false data or damaged data, and make the infrastructure more health, high-performance and autonomic. Although it will be a long way to make the infrastructure more powerful and autonomic, it will be reached, sooner or later.

6. User interface and test

When the data sources and data brokers are implemented, the support libraries for accessing them are implemented at the same time. By appending some additional function for users, the support libraries of data infrastructure is built up. Furthermore, a web based user interface is implemented based on the support libraries and JSP technology. Users can search data by conditions such as time, satellite ID, sensor type, and region covered; and operation on the results such as download all data, view the quick look, get the meta-data, and download data of some wave bands. The pictures in figure 6a-6d are samples of the user interface deployed in Apache Tomcat when being tested.

The test shows that user can search data, download data, get meta-data, view quick look, etc. on a mass of data sources through a uniform user interface. The test also shows that the data infrastructure can distribute spatial data with different types, different sources and different goals through a uniform programming interface for the geo-computation applications.
7. Conclusions

The research of geo-computation purchases the order of distributed, platform-independent, extensible, robust, and autonomic data infrastructure to serve the geo-computation applications in one-stop. Based on the fashionable grid computing research productions, a SIG based data infrastructure for geo-computation is proposed. Figure 7 shows the different between it and the ordinary data infrastructure for geo-computation.

<table>
<thead>
<tr>
<th></th>
<th>Ordinary data infrastructure</th>
<th>SIG based data infrastructure</th>
</tr>
</thead>
<tbody>
<tr>
<td>distributed</td>
<td>maybe</td>
<td>yes</td>
</tr>
<tr>
<td>platform-independent</td>
<td>difficult</td>
<td>easy</td>
</tr>
<tr>
<td>dynamic</td>
<td>low</td>
<td>high</td>
</tr>
<tr>
<td>robust</td>
<td>low</td>
<td>high</td>
</tr>
<tr>
<td>autonomic</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>application development</td>
<td>difficult</td>
<td>easy</td>
</tr>
<tr>
<td>one-stop service</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>

Fig. 7. Different between the ordinary data infrastructure and the SIG based data infrastructure

To response the requirement of geo-computation, a 4-layered architecture based on SOA is proposed for the data infrastructure for geo-computation, and some useful data sources, data agencies and support libraries are designed and implemented in this paper. A test on the user interface of the data infrastructure shows that it can organize the distributed data nodes and data agencies dynamically; build an extensible, robust, and autonomic data infrastructure; and serve the users on-stop as an organic whole.

References

[9] Web Services Description Language (WSDL) 1.1: http://www.w3.org/TR/wsd1

Author Bios

ZhenChun Huang was born in 1975 and graduated with the Ph-D degree at Tsinghua Univ. in 2002. Since 2002 he is research assistant at the Dept. of Computer, Tsinghua Univ. His main research interest is Grid Computing and High-performance Computing.

GuoQing Li was born in 1968 and graduated with the Ph-D degree at China Academy of Science. He is Associated Researcher at the China Remote-Sensing Satellite Ground Station. His main research interest is High-performance Geo-Computing and Spatial Information Grid.