SIGRE – An Autonomic Spatial Information Grid Runtime Environment for Geo-computation

ZhenChun Huang[1], GuoQing Li[2], Bin Du[1], Yi Zeng[2], and Lei Gu[1]

[1] Department of Computer Science and Engineering, Tsinghua University, Beijing 100084
[2] China Remote Sensing Satellite Ground Station, Beijing 100086
huangzc@tsinghua.edu.cn, gqli@ne.rsgs.ac.cn,
dubin@mails.tsinghua.edu.cn, yzeng@ne.rsgs.ac.cn,
jackflit98@mails.tsinghua.edu.cn

Abstract. Spatial Information Grid is a kind of application grid which tries to connect resources such as computer, data sources, and processing algorithms, and builds a distributed, robust, flexible and powerful infrastructure for geo-computation. It needs a powerful and easy-to-use running environment. In this paper, an autonomic runtime environment for geo-computation is proposed and named SIGRE — the Spatial Information Grid Runtime Environment. Based on it, SIG resources can be distributed, discovered, and matched autonomically. And a distributed, flexible and powerful data infrastructure which can distribute data with different types, different sources, and different goals in a uniform interface easily and flexibly is founded. Based on the implementation of SIGRE by java language, a SIG testbed is constructed, and the test on it shows that SIGRE can provide a powerful, easy-to-use, robust and autonomic runtime environment for SIG, and developers can develop SIG resources and SIG applications on SIGRE easily and quickly.

Keywords: geo-computation, spatial information grid, runtime environment.

1 Introduction

Geo-computation can be regarded as the application of the computational science paradigm to study a wide range of problems in geographical and Earth science context. It is concerned with new computational techniques, algorithms, and paradigms that are dependent upon and can take advantage of techniques such as Grid computing, high performance computing and high throughput computer. Based on grid computing technologies, the Spatial Information Grid (shortly SIG) tries to connect resources such as high performance computers, high throughput data storages and spatial data processing algorithms; enable the sharing, co-operation and collaboration among the resources and their consumers; and build a distributed, robust, flexible, and powerful infrastructure for geo-computation. Based on SIG, resources such as computing powers, spatial information data, and processing models and algorithms can be shared; and distributed geo-computation applications can be built easily and quickly.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 322–329, 2007.

In brief, SIG can be regarded as a kind of application grid for the geo-computation researchers. It is based on the general grid infrastructure; and built by the general grid infrastructure and dozens of SIG extensions and improvements, such as the SIG job framework, SIG data infrastructure, SIG registry meta-service extension, and so on. In SIG, the general grid infrastructure makes it possible to share resources and co-operating among resources. On the other hand, the SIG extensions and improvements make the infrastructure more powerful, friendly, and adaptive for the geo-computation users, and make it easier and quicker to develop SIG applications on the infrastructure.

As viewed from the physical deployment, SIG is constructed by a lot of distributed SIG nodes which are connected by network (usually Internet). All resources in SIG are attached to the SIG nodes, and accessed through the nodes. At the same time, most of the web-based SIG applications are deployed on the SIG nodes and provide web UI for their users, too. Whether the SIG resources or the SIG applications need a basic support running environment, which will be proposed later and named SIGRE — the Spatial Information Grid Runtime Environment, an autonomic runtime environment for geo-computation.

Most traditionally, the "runtime environment" is implemented as a set of support libraries on the target platform for the basic functions of the applications. These libraries are invoked by the application codes through a well-defined API (Application Programming Interface) and serve applications the most frequently used functions. One of the most famous samples is the Unix lib5 runtime environment for C applications, which carries the basic functions such as file accessing and process management. And there are different platform-dependent lib5 libraries on different platforms, and they support the same functions. The Win32 runtime environment is another sample for this set.

By the growing up of virtual machine techniques, virtual machine is adopted as a part of the runtime environment, and provides a platform-independent running core for applications; so that the applications may move from one platform to another smoothly and quickly without modification. It can be called "VM based runtime environment". One of the most famous VM based runtime environment is java runtime environment which is often called JRE. In JRE, a virtual machine named JVM (Java Virtual Machine) is included, and interprets the java virtual codes for the java applications. And at the same time, a set of java runtime libraries are included to provide frequent functions such as iostream or url supports. The JRE supports different platforms, and makes it possible to "build one, run anywhere". The .Net framework is also another sample of this kind of runtime environments.

Though the VM based runtime environment provides a platform-independent environment for applications by a well-defined virtual machine and a set of support libraries, it is still not enough for the applications today, especially for the resources and applications in grid such as SIG. To develop, deploy and manage resources and applications based on a VM based runtime environment is still a hard work. So, an autonomic runtime environment which is able to self-configure, self-manage, self-heal and self-optimize is needed for better development and execution of the resources and applications.

In this paper, based on the techniques including web service, grid computing, and spatial information grid, an autonomic runtime environment named SIGRE (Spatial

Information Grid Runtime Environment) is proposed and implemented for geo-computation. Each SIG node carries out an instance of SIGRE, which provides the basic running environment including the virtual machine, runtime libraries, management and optimization components, monitoring and healing components, etc. All SIGRE instances on the SIG nodes makeup an environment for the execution and management of resources and applications in SIG, and provide the key functions including resource discovering, data managing, service quality supporting, etc.

2 Architecture

There are two connotations of SIGRE: the SIGRE instance deployed on a SIG node, which is called SIGRE instance; and the runtime environment built by all the SIGRE instances on SIG nodes, which is called SIGRE platform. As viewed from the logical architecture, SIGRE is built up by the following components: Java virtual machine and runtime environment; general web server and servlet container; SIG resource and job framework; SIG data infrastructure support; SIG resource registry and discovery; SIG service and node monitor; SIG service quality control and management; and so on. It is shown in Figure. 1.

SIGRE is founded based on the following components: JVM and JRE [1], web server, and servlet container [2]; and they should be downloaded and installed separately for the sake of the user's free choice on them. SIG resource and job framework [3] provides a basic job framework and make it possible to create, run, monitor, and manage an SIG job through web service interface. Most of the SIG resources are implemented and provided as SIG jobs. With the toolkits provided by SIGRE, it is very simple and easy to develop and deploy an SIG job so that almost each user who knows a little of Java can share his resources by develops and deploys new SIG jobs.

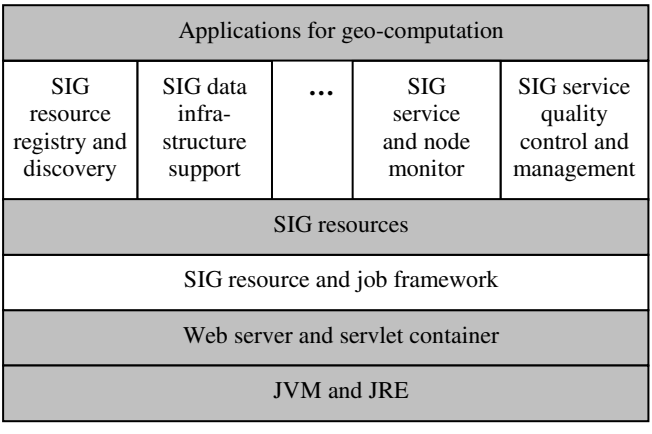


Fig. 1. Architecture of SIGRE

Besides the job framework, SIGRE provides some more components to make the runtime environment more useful and autonomic. They are: SIG resource registry and discovery service, SIG data infrastructure support, SIG service and node monitor framework, SIG service quality control and management, etc. The components provides a lot of basic functions for resources and applications, such as to collect and distribute the information about services ad nodes, monitor the execution of resources

and nodes, index and discover SIG resources, and control and manage service quality which can make SIG more friendly and autonomic.

3 The Autonomic Resource Distribution, Discovery and Matching

It is very important for users and applications to find and assemble the most suitable resources. It depends on several functions provided by SIGRE: resource discovery, service and node monitor, service quality control and management, etc. The figure 2 shows information flows for the autonomic resource distribution, discovery and matching.

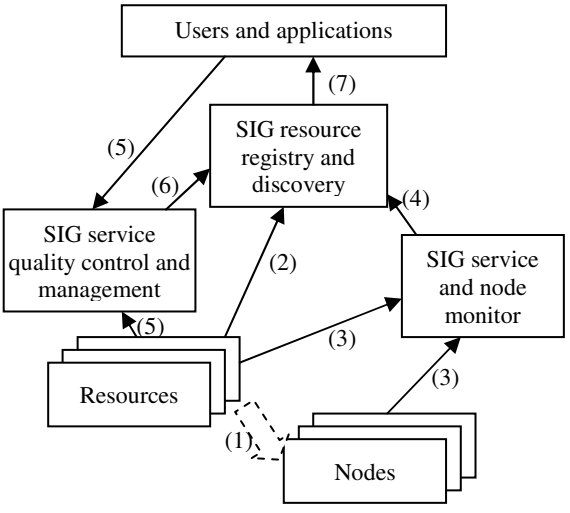


Fig. 2. Information flows for the autonomic resource distribution, discovery and matching

First of all, when the resource are deployed on nodes, (1) the static information about the resources such as its name, usage and function will be registered to the resource registry and discovery service. (2) Then, when the resources and services are invoked executed, the performance and status information about them will be probed and collected by the SIG service and node monitor. (3) And the information organized and analyzed by the SIG service and node monitor will be provided to the SIG resource registry and discovery service for service discovery and matching. (4) Furthermore, the quality and cost of SIG service may be different far away, users may care about them. So, the SIG service quality control and management will collect the service quality information from resources, users and applications, analyze them to get the score, (5) and provide the score to the SIG resource registry and discovery component so that the users and applications can find the most suitable and high quality resources by the resource matching. (6) Finally, integrating the static and dynamic information from SIG resources, service quality control and management component, service and node monitor, and other parts of SIGRE, the SIG resource registry and discovery service can discover and match the most suitable SIG resources for users and applications request. (7)

Around the SIG resource registry and discovery service which is extended based on the Distributed Grid Resource Registry Meta-Service [4] for geo-computation users, SIG service and node monitor, SIG service quality control and management component, and all the deployed SIG resources make up a closed loop. By the closed loop, the static, history and real-time information about all SIG resources, node status, service status, service quality, and so on, is collected, analyzed, and proceeded for the

resource matching. Then, the users and applications can find the resources needed: the resource with the highest performance, the best quality, the lowest cost, the most robust node, or the widest network bandwidth. With the distributed architecture of SIG, the closed loop enables the SIGRE to self-configure, self-manage, self-heal, and self-optimize in the resource discovery and matching.

For example, an application can find a data processing resource based on a super computer which has at least 16 processors and 8GB memory, a data source which provides Landset7 image through network with bandwidth at least 10Mbps, and a WMS service to visualize the processing result. SIGRE and the resource discovery service will search in the SIG resources by a well-defined p2p protocol and find the available and most suitable resources for the application. What the application need do only is to access the resources found and resolve problems for the end users. SIGRE provides an autonomic run time environment for the SIG applications.

In the other hand, the development and distribution of resources is very simple, too. Most of the SIG computation resources are implemented as SIG jobs which is deployed in the SIG job framework and managed by the SIGRE. As soon as a java class or a script file which implement the job and a job description file are deployed in the job framework, a SIG job is deployed and can serve the users and applications. And all information about the job will be collected, analyzed and used by the registry and discovery service, the service quality control and management component and other parts of SIGRE. SIGRE can configure and manage itself and resources based on it. It provides an autonomic runtime environment in resource distribution, discovery and matching for SIG resources and applications.

4 The Autonomic Data Infrastructure Based on SIGRE

Data sharing and distribution is another important usage for Spatial Information Grid. And it plays the role of bridge between the data consumer and data supplier, and brings spatial data on the finger of users. It is required for almost all the SIG applications to search and access spatial data through a powerful and easy-to-use data infrastructure. For the construction of SIG data infrastructure, a data sharing framework for the support of data infrastructure is included in the SIGRE. Based on it, a data distribution system is designed and implemented. [5] The data distribution system is built up by data sources, data agencies, registry services in SIGRE, client support libraries, and other components. Applications can query and access stored data by a well-defined XML based extensible RSI-data source accessing language through SOAP protocol which is platform-independent.

In the data infrastructure, each data provider is organized as a data source which has a uniform service accessing point. The service accessing point is a web service which is described by a WSDL [6] document and invoked through SOAP [7] 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. Users can accomplish the operation by invoking the web service provided by data source or data agency, passing the request in XDAL format to it, and analyzing the response in XDAL format for the result. Figure 3 is a sample of searching data from satellite "Landset-7" with a given acquirement date.

Comparing with the ordinary data distribution system, the SIG based data infrastructure can distribute data with different types, different sources, and different goals in a uniform interface easily and flexibly. Furthermore, based on the data infrastructure support of SIGRE, a lot of "data agencies" will be continuously deployed in the data infrastructure and make the infrastructure more flexible, extensible, autonomic, and powerful.

1、client sends the request of searching an image

```
<query>
  <conditions relation="AND">
    <condition op="EQ">
       <param>satellite</param>
      <value>landset7</value>
    </condition>
    <condition op="EQ">
       <param>date</param>
       <value>2006-01-01</value>
    </condition>
  </conditions>
  <orders>
    <sortBy order="ASC">ID</sortBy>
  </orders>
</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 -->
<status>
  <operationID>001235864631</operationID>
  <currentStatus>finished</currentStatus>
</status> <!-- finished -->
```

5、client gets the operation result

```
<getResult>
  <operationID>001235864631</operationID>
</getResult>
```

6. data source returns the operation result

```
<result>
  <operationID>001235864631</operationID>
  <resultSet>
    <item>
      <id>0013256782</id>
      <spacecraft>Landset7</spacecraft>
      <senser>ETM+</senser>
       ……
    </item>
      ……
    <item>
      <id>0020165784</id>
       ……
    </item>
  </resultSet>
</result>
```

Fig. 3. Sample of searching data

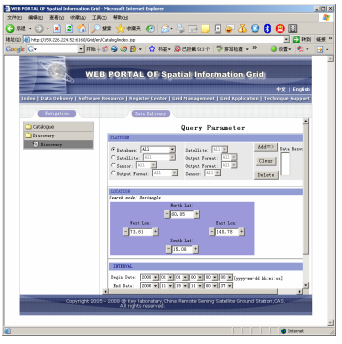
5 Implementation, Test and Future Works

Java language is often adopted by grid implementation because of its platform-independent, acceptable performance and mass of open-source support. So, the SIGRE is implemented based on the famous SOAP support library Axis [8] by java language, too. For the sake of easy deployment, the SIGRE implementation is distributed as a WAR package. It can be deployed on most of the popular servlet container such as Apache Tomcat [9] and

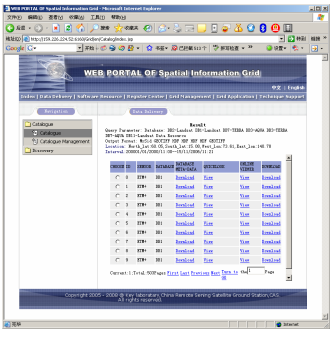
provide an autonomic runtime environment for not only SIG applications but also SIG resources. Based on the SIGRE, a SIG testbed is constructed and deployed on Apache Tomcat servlet container. The testbed integrates three kinds of computing nodes (high performance cluster, condor computing pool, and traditional high-end server, totally more than 100 processors) for its computing infrastructure, about ten data sources with more than 10TB data for its data infrastructure, more than forty algorithms for data processing models, several WMS servers for data visualization, a web based management and monitor user interface for management, and so on. Figure 4 shows some of the SIG testbed user interfaces.

The test on SIG testbed shows that SIGRE can provide a powerful, easy-to-use, robust and autonomic runtime environment for SIG, developers can develop SIG resources and SIG applications on SIGRE easily and quickly. For example, based on the SIGRE data infrastructure support component, a new data source can be developed and deployed successfully in one day at most, and to create a new application based on SIGRE may take only hours or less.

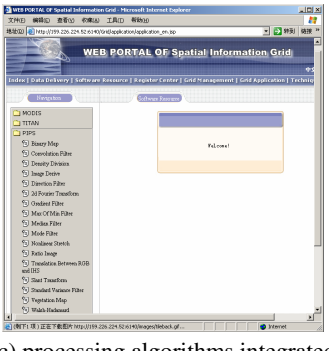
As a kind of application grid for geo-computation context, Spatial Information Grid needs a powerful and easy-to-use runtime environment for its users and developers. In this paper, based on the techniques including web service, grid computing, and spatial information grid, an autonomic runtime environment named SIGRE is proposed and implemented for geo-computation. Each SIG node carries out an instance of SIGRE, which provides the basic running environment including the java virtual machine, runtime libraries, management and optimization components, monitoring and healing components, etc. All SIGRE instances on the SIG nodes makeup an environment for the execution and management of resources and applications in SIG, and provide the key functions including resource discovering, data managing, service quality supporting, etc. based on SIGRE, a SIG testbed is constructed and tested. The test shows that SIGRE can provide a powerful, easy-to-use, robust and autonomic runtime environment for SIG, and developers can develop SIG resources and SIG applications on SIGRE easily and quickly.



(a) input data searching arguments



(b) data searching result



(c) processing algorithms integrated

Fig. 4. User interfaces of SIG testbed

But, the SIGRE can not make the SIG popular immediately and automatically. A lot of useful and powerful applications are needed to respond the requirement of SIG end users. It is very important for the survival and growth of SIG to provide mass of applications for the users to resolve their problem.

The requirements of the end users are so various that it is difficult to respond them by limited developed applications. The application development becomes the bottleneck of SIG popularization. One way to ease the bottleneck is to develop and deploy more and more powerful applications as soon as possible and as many as possible so that the requirements of end users can be responded. It is the traditional way and seems failed. There are two main reasons at least. First, if the application is more powerful, it is more complex, more expensive, and more difficult to develop. Second, the requirement of end user is always uncertain and unstable. It is too difficult to develop a powerful application for the uncertain and unstable requirement of a mass of end users with a low cost.

Another possible way to ease the bottleneck is to make the application development easier so that more people, even all end users can develop applications for themselves. Situational application will be a good choice. In SIG, situational application is such an application that it is quickly created and deployed based on the SIG infrastructure and SIG resources for a situational requirement by the end user. It is often developed and deployed by a series of visual development toolkits. To make the SIGRE support the development and deployment of situational applications will be one of our most important future works to make the SIG more useful and popular. Based on the SIGRE situational application support, users can create and deploy situational applications for their requirements themselves. It will ease the application development bottleneck of SIG and make SIG more powerful and popular.

References

1. Java Technology, http://java.sun.com
2. Java Servlet - Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/Java_Servlet
3. Huang, Z.C., Li, G.: An SOA based On-Demand Computation Framework for Spatial Information Processing. In: GCCW 2006. Fifth International Conference on Grid and Cooperative Computing Workshops, Hunan, China, October 21-23, 2006, pp. 487–490 (2006)
4. Huang, Z.C., Du, B., Gu, L., He, C., Li, S.: Distributed Grid Resource Registry Meta-Service: Design and Implementation. In: The 7th International Symposium on Autonomous Decentralized Systems, Chengdu, China, April 4-6, 2005, pp. 531–535 (2005)
5. Huang, Z.C., Li, G.: SIG-DDS: A Grid-based Remote Sensing Data Distribution System. In: SKG 2006. Second International Conference on Semantics, Knowledge, and Grid, GuiLin, China, November 1-3, 2006, pp. 93–94 (2006)
6. Web Services Description Language (WSDL) 1.1: http://www.w3.org/TR/wsdl
7. SOAP: The fundamental message enveloping mechanism in Web services, http://www.w3.org/TR/SOAP
8. The Apache Software Foundation: Web Services - Axis, http://ws.apache.org/axis/
9. The Apache Software Foundation: Apache Tomcat, http://tomcat.apache.org/