Study on Implementation of High-Performance GIServices in Spatial Information Grid

Fang Huang[1,2,4] , Dingsheng Liu[1], Guoqing Li[1], Yi Zeng[1,3,4], and Yunxuan Yan[1,4]

[1] Center for Earth Observation and Digital Earth, Chinese Academy of Sciences, Beijing 100086, P.R. China
[2] Institute of Remote Sensing Applications, Chinese Academy of Sciences, Beijing 100101, P.R. China
[3] Institute of Electronics, Chinese Academy of Sciences, Beijing 100090, P.R. China
[4] Graduate University of Chinese Academy of Sciences, Beijing 100049, P.R. China
{fhuang, dsliu, gqli, yzeng, yxyan}@ceode.ac.cn

Abstract. Providing geo-spatial data services (GDS) and processing functionality services (PFS) are the key issues in spatial information grid (SIG). Especially, it's crucial for SIG to offer PFS related to Geographic Information Science (GIS), instead of just focused on Remote Sensing (RS) field. Furthermore, implementing high-performance GIServices is the main task of SIG to offer PFS for GIS. Lacking of high-performance GIServices mainly resulted from the limitations of architecture as well as the complexity for services implementation and encapsulation. Based on existing SIG platform, we propose the new architecture of SIG, upon which the constituted GIS nodes can provide GIServices. Within the improved architecture, some parallel GRASS GIS algorithms programs, which are built by different parallelization patterns and can run in cluster with better efficiency, are encapsulated to high-performance GIServices guiding by certain generic mode. Lastly, the analyses of the test demonstrate that the approach can reach our aims.

Keywords: Spatial Information Grid (SIG); GIServices; Cluster; GRASS GIS.

1 Introduction

Generally, spatial information grid (SIG) is a fundamental infrastructure that can collect and share all types of geospatial information rapidly and effectively, with powerful capabilities for service on demand, geospatial data management and information processing. In addition, SIG is a distributed environment that combines resources such as geospatial data and computing, story and processing tools to supply services to geospatial applications [1].

The current SIG platform was supported by the Hi-Tech Research and Development Program of China [2]. It is comprised several grid nodes such as data grid node, computing grid node, controlling and management node, which are built based on basic grid container. Moreover, the platform integrated such software as Titan (one type of commercial software for RS image processing) [3], PIPS (one parallel RS image processing software in cluster developed by CEODE, CAS) [4]. In all, the platform can provide both geo-spatial data services (GDS) and processing functionality services (PFS). The

M. Bubak et al. (Eds.): ICCS 2008, Part II, LNCS 5102, pp. 605 – 613, 2008.

former can handle terabytes of data; the latter not only provides normal RS services (came from sequential RS algorithms), but also offers some high-performance RS services (encapsulated from parallel RS algorithms in cluster with high performance and better efficiency). However, current PFS in SIG have not extended to GIS field yet. Meanwhile, to the extent that it's crucial and full of challenges to provide some high-performance GIServices in SIG, which mainly resulted from: (1) The existing architecture can not flexibly suitable for extending GIS node to provide GIServices, because the node construction and services procedure for GIS is much more different than that of RS; (2) It's difficult to get some parallel GIS programs running in Linux cluster utilizing common commercial GIS packages; and (3) Lacking of some instructions to guide the GIServices implementation.

Owing to the limitations, most geospatial end users cannot access the services related to GIS available through SIG. Thus, the challenges that arise are how to overcome the architecture limitation and how to implement some GIServices, especially some algorithms processing functionality with one easy and convenient approach. Fortunately, the paper puts forward new layout of SIG, which can support GIS nodes and provide GIServices. Within the improved architecture, some parallel GRASS GIS (Geographic Resources Analysis Support System)[5] algorithms programs, which are reconstructed by different parallelization patterns with better speed-up and efficiency, are encapsulated to high-performance GIServices in cluster with assistance of SIG tools.

The paper is organized as follows. Section 2 gives a brief introduction to SIG's improved architecture. Based on this, in Section 3, various parallelization patterns for extracting some GIS parallel programs from GRASS GIS package are proposed. Subsequent section mainly concentrates on the encapsulation mode for those parallel programs, and explains the service invoking flow. In Section 5, one practical way is put forward to evaluate the efficiency of those high-performance GIServices based on one test example. Finally, Section 6 gives some conclusions.

2 Improved Architecture and Analyses for Implementation

2.1 Improved Architecture

As vector, one data structure type of GIS, has different characteristics from RS, on which make the algorithms based become relative complicated. Thus, the difference makes it much difficult to extract some parallel programs from the GIS package in Linux cluster, and to wrap into GIServices, especially into the high-performance GIServices. Meanwhile, the existing architecture of SIG considered litter to the related aspects, which makes it difficult to provide GIServices in SIG platform directly. Thus, the overall layout of SIG need be improved when considering these factors. The new architecture is just illustrated as Fig.1.

In Fig.1, SIG container is the fundamental part of SIG, which is the middleware combination of several grid tools that is quite suitable for using grid technology in geospatial field. Through container, the different grid nodes can be easily to communicate and achieve one task collaboratively in the distributed heterogeneous environment.

In the overall arrangement, there are 4 types of nodes in SIG: SIG management & controlling node (SIG MC Node), geo-spatial data Grid Service node (GDS Node), processing functionality service node (PFS Node), and SIG Web portal.

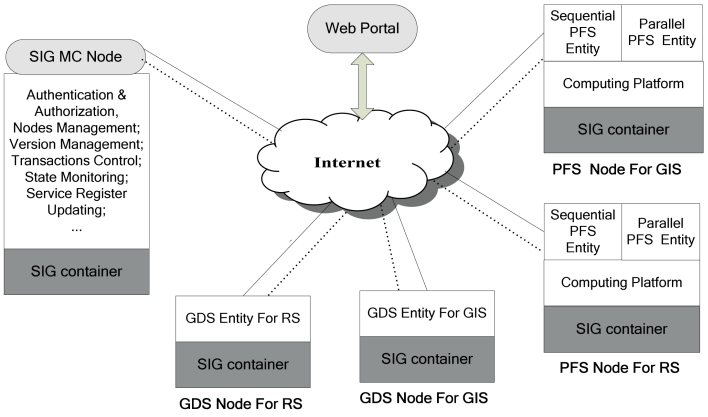

Fig. 1. Improved architecture of SIG. In the initial stage, there are only some RS nodes. In the new arrangement, some GIS nodes are added and can provide some GDS and PFS of GIS. Especially, the new architecture facilitates providing high-performance GIServices.

SIG Web portal is the only entrance of SIG. Through the authentication and authorization, the user can select the appropriate services.

SIG MC Node is the controlling and management centre of SIG. It not only manages all kinds of the operations as authentication and authorization, transaction controlling in SIG Web portal, but also takes responsibilities for Grid services node, comprising updating of services registry information, version management, node maintenance, state controlling, resources scheduling and services controlling.

GDS Node can publish the storied RS/GIS data with the form of services through SIG. Those services called GDS for RS and GDS for GIS, respectively. The users can share or download them through SIG Web portal.

Meanwhile, PFS Node can serve PFS related to RS and GIS, respectively called PFS for RS and PFS for GIS. Among those 2 types of PFS, they are respectively divided into normal PFS (sequential programs before encapsulation) and high-performance PFS (parallel programs in cluster before encapsulation) according to the computing environment and other factors. Thus, we can provide 4 kinds of PFSs, namely, S-PFS for RS/GIS and HP-PFS for RS/GIS.

2.2 Analyses for High-Performance GIServices Implementation

From the above, we know that high-performance GIServices here ascribes to HP-PFS for GIS. Similarly, the procedure for high-performance GIServices can be deduced from that of HP-PFS for RS, and mainly includes:

Step 1: It's critical to get some parallel GIS programs with high speed-up and better efficiency in Linux cluster; and

Step 2: Guiding by some encapsulation mode, those programs can be encapsulated into SIG services;

Relatively, step 1 should be paid more attention for implementation high-performance GIServices, because: (1) Utilizing commercial GIS packages, we can not get some parallel programs with certain GIS algorithms according with our demands,

for we can not get any source codes of them; and (2) Those commercial packages mostly run in Windows, while our computing platform is Linux system. From the point of performance and convenience, it also adds extra difficulties to the first step.

Just because those factors, the subsequent step cannot work due to lacking of some parallel GIS algorithms programs produced in step 1.

2.3 Our Approach

With carefully studies, we select one open source GIS package in Linux, GRASS GIS, as our research object, which can overcome the difficulties mentioned above. The following sector will discus several parallelization patterns to it, through which some parallel GRASS GIS modules with better speed-up can be easily reconstructed.

3 Reconstructed Parallel GIS Programs Based on GRASS GIS

Our cluster is Linux based system established by commodity PCs, and belongs to shared disk architecture. In general, it needs some special libraries such as MPI (Message Passing Interface) [6] to develop parallel programs on it.

3.1 Several Parallel Patterns for GRASS GIS

Parallel programming involves developing a single computer program in such a way that it can be executed by more than one processor simultaneously. Data partitioning and function partitioning are effective parallel programming techniques for most applications [7, 8]. Taken account of the characteristics of cluster, the database and other factors of GRASS GIS, we tentatively put forward several parallel patterns for GRASS GIS. Those patterns mainly comprise multi-user data paralleling pattern (MUDPP), GRASS GIS algorithm parallel pattern (GGAPP) and duple parallel pattern (DPP).

MUDPP is the method of data partitioning based on the multi-user runtime environment (MURE) and geo-database of GRASS [9]. In fact, MUDPP is the development mode of SPMD (single program multi data). GGAPP dedicates to develop some parallel programs with function partitioning technique. Those independent modules concentrated on v.path and v.buffer. DPP is the integration of MUDPP and GGAPP.

For the limitation of paper space, we don't further to specify the last 2 patterns, which deserve further study and will be introduced in certain special articles. Namely, here we only focus on MUDPP.

3.2 Working Principle, Generic Mode of MUDPP

In MUDPP, some GRASS mapsets belong to one LOCATION are established both in master and slave nodes of cluster. The working principle of MUDPP is described as follows. Firstly, the mapset belongs to master node will partition the input map into litter subparts and send instructions to the corresponding mapsets located in slave nodes. When the slave mapsets finished their own subtasks by receiving and invoking the same processing instructions concurrently, the master will merge the sub-results as the entire output map. Of course, the output must be identical to the processing result when the same task processed in sequential.

Thus, the problems of data partitioning and merging in the course of input and output should be attached sufficient importance to. As GRASS has only 2 data type, raster and vector [10], if we can solve their partitioning and merging problems, some of the GRASS modules can be paralleled in cluster without many changes in their codes. The deception is just what our generic model [9] dedicates to achieve.

3.3 Realization of MUDPP

In order to accomplish MUDPP, it is requirement to establish some modules firstly, which includes 2 kinds: partitioning & merging modules, and universal module. The former can partition and stitch the input/output dataset of raster/vector. The latter can parallel several GRASS GIS functionalities with one universal program by invoking the other modules. The functionalities of those modules are listed in Table 1.

Table 1. In MUDPP, p.universal can obtain the parallelization GRASS modules by invoking the remainders

Module name	Functionality
RunModuelI-nOneUser.sh	Start GRASS GIS in one mapset either in master or slave, thus the functionality modules can run on the active mapset.
p.universal	The implementation of MUDPP. Within the pattern, the parallel GRASS GIS modules are reconstructed with the method of SPMD.
p.r.in.partition	Finished the partitioning processing for raster map.
p.r.out.merge	Accomplished the merging procedure for raster map.
p.v.in.partition	Finished the partitioning processing for vector map.
p.v.out.merge	Accomplished the merging procedure for vector map.

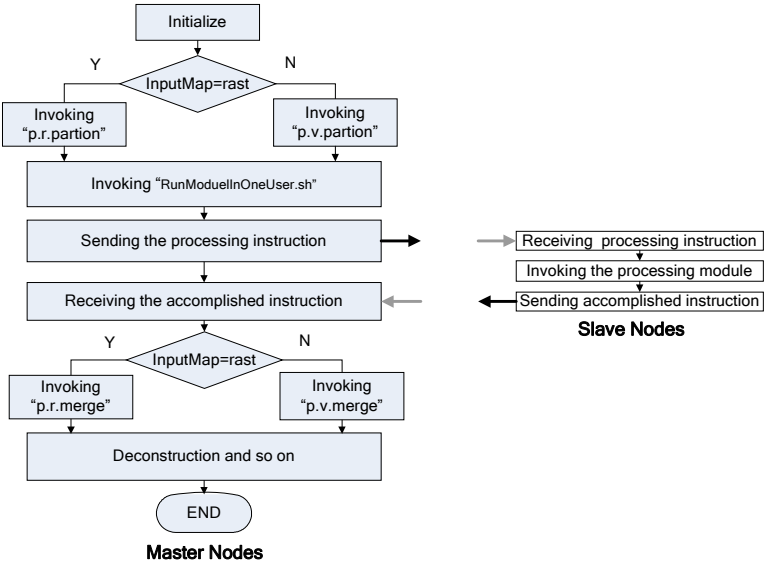

Fig. 2. Flow chart of MUDPP development. The universal module invokes the partitioning and merging modules with MPI under the fundamental GRASS GIS environment.

Fig.2 illustrates the development of MUDPP, which invokes the fundamental modules of GRASS GIS, and MPI.

4 GIServices Encapsulation Mode and Invoking Flow in SIG

Through these patterns, the parallel executing programs of GRASS modules are available in cluster. Utilizing relevant tools, we can wrap the parallel modules into high-performance GIServices under the encapsulation mode.

4.1 High-Performance GIServices Encapsulation Mode

As those parallel programs are running in Linux cluster, and still need the support of GRASS fundamental environment, all of those make its encapsulation become more different than that of RS PFS. Integrating with the existing SIG platform, 4 steps are summed up (Fig. 3):

Step 1. Extracted some executed parallel program (C programs) from GIS package. Those programs can reconstructed by the patterns mentioned above;

Step 2. Those executed programs need be encapsulated to .class files with help of Java JNI (Java Native Interface). When the java program can be run successfully in local, it indicates the service entity is implemented successfully;

Step 3. Publish the service entities (Java class files) to SIG services with Tomcat. As the result, the WSDL (Web Services Description Language) file will be produced;

Step 4. The published high-performance GIServices should be registered to the SIG MC Node with corresponding tools.

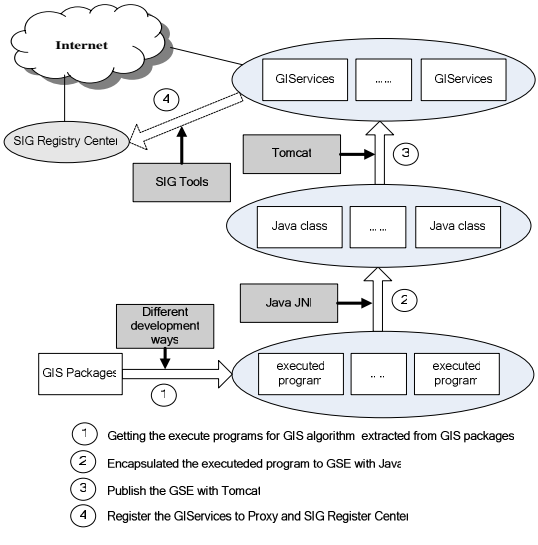

Fig. 3. The high-performance GIServices encapsulation mode. The details for each step are illustrated under the figure.

4.2 High-Performance GIServices Invoking Flow in SIG

When the published GIServices is needed, the user should select the corresponding data for processing located in SIG. After the processing accomplished, the results can be either viewed online or downloaded to the user's computer. Fig. 4 instantiates the whole work flow of the services invoked in SIG platform in detail.

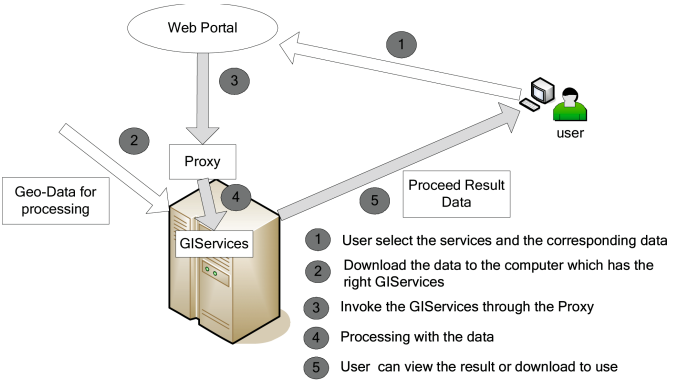

Fig. 4. Invoking workflow of the published high-performance GIServices in SIG. Follow the figure, some explanations for the invoked steps are listed.

5 Analyzing of the High-Performance GIServices Efficiency

There is also requirement to use one suitable way to validate the efficiency of the high-performance GIServices through those parallelization patterns. It seems to be the best method to contract the high-performance GIServices with the corresponding normal GIServices directly under identical conditions. In fact, the sequential GIServices may be developed from some commercial GIS packages, which has different computing efficiency from GRASS GIS. Moreover, some uncertain factors such as the state of the network etc. may exist in the respective services procedure. Those factors must be considered to evaluate the services efficiency. Thus, we propose the following formula to ascertain the services efficiency.

$$T_{GIServices} = T_{Data\_Acquision} + T_{Data\_Processing} + T_{Communication} + T_{Result\_Dowload} + T_{Others} \quad (1)$$

Here, $T_{GIServices}$ represents the sum elapsed time of the whole services; $T_{Data\_Acquision}$ is the consuming time of the data acquisition, by means of downloading or sharing; $T_{Data\_Processing}$ means the processing time of the program with the same datasets in one some computing platform; $T_{Communication}$ is the communication time related to network; $T_{Result\_Dowload}$ indicates the time for user to download the results;

while the last part T_{Others} is the elapsed time for the remainder except the parts mentioned. The equation illustrates the consuming time of the GIServices in the dynamic environment, which can be used to represent the efficiency of GIServices indirectly.

When we suppose those 2 kinds of GIServices are in same conditions, namely, all of items expect $T_{Data\_Processing}$ have the identical values in (1). Therefore, the whole services efficiency depends on $T_{Data\_Processing}$, i.e., we can use it to represent the corresponding GIServices efficiency.

In order to illustrate the excellent efficiency of high-performance GIServices, we select the sequential and parallel programs developed both from GRASS GIS, which can avoid the computing capability differences result from different GIS packages. Table 2 shows the value of $T_{Data\_Processing}$ in r.example and r.contour in the form of sequential and parallel respectively.

Table 2. The approximate consuming time(s) of r.example and r.contour in sequential and parallel forms under different processors in the same computing platform

Module name	Number of the processors							
	1	2	4	6	8	10	12	20
r.example	158	/	/	/	/	/	/	/
p.universal/r.example	/	129	119	91	101	105	106	115
r.contour	148	/	/	/	/	/	/	/
p.universal/ r.contour	/	157	63	45	38	37	34	39

From the contrast results, we know that the values of $T_{Data\_Processing}$ have much difference. When they in the right numbers of processors (>2), the parallel modules has a better efficiency than the common modules. Therefore, we can deduce that under the same conditions, including the same dataset, network environment, computing platform and so on, the high performance GIServies has a better efficiency than that of the common GIServies, especially for the big size of data, whose processing are full of computation intensive.

6 Conclusions

Much work is still needed to explore efficient approaches to make GRASS GIS algorithms parallel in cluster except the mentioned 3 parallelization patterns. Moreover, there is also a requirement to construct more high-performance GIServices based on the new architecture with GRASS GIS.

However, the test examples and analyses to the experimental GIServices have led to some useful conclusions: (1) The new architecture is practicable for constructing GIServices; (2) The parallelization patterns, especially MUDPP, are suitable for presenting some parallel GIS algorithms in cluster; and (3) The constructed high-performance GIServices have a better efficiency than the opposite normal GIServices.

References

1. Jin, J.J.: The applications of grids in geosciences [in Chinese], http://support.iap.ac.cn/bbs/viewthread.php?tid=176&extra=page%3D1
2. http://www.863.org.cn
3. http://www.otitan.com/index.shtml
4. http://159.226.224.52:8021/showdetail.asp?id=2
5. http://grass.itc.it/
6. http://www-unix.mcs.anl.gov/mpi/mpich
7. Brawer, S.: Introduction to parallel programming. Academic Press, San Diego (1989)
8. Wang, F.J.: A Parallel GIS-Remote Sensing System for Environmental Modeling. In: IGARSS 1992, pp. 15–17 (1992)
9. Huang, F., Liu, D.S., Liu, P., et al.: Research On Cluster-Based Parallel GIS with the Example of Parallelization on GRASS GIS. In: GCC 2007, pp. 642–649 (2007)
10. Blazek, R., Neteler, M., Micarelli, R.: The new GRASS 5.1 vector architecture. In: Proceedings of the Open Source GIS–GRASS user conference 2002 (2002)