System Software for China National Grid

Li Zha[1], Wei Li[1], Haiyan Yu[1], Xianghui Xie[2], Nong Xiao[3], and Zhiwei Xu[1]

[1] Institute of Computing Technology, Chinese Academy of Sciences,
100080 Beijing, China
[2] JiangNan Institute of Computing Technology
[3] National University of Defense Technology

Abstract. The China National Grid project developed and deployed a suite of grid system software called CNGrid Software. This paper presents the features and implementation of the software suite from the viewpoints of grid system deployment, grid application developers, grid resource providers, grid system administrators, and the end users.

1 Introduction

The China National Grid project is a 4-year (2002-2005) R&D project sponsored by China Ministry of Science and Technology. It aims to constructing a wide-area national grid [1] environment to enable resource sharing and collaboration over the Internet, using standards-based components and novel technology developed by this project. Another main goal is to build up human resources in grid computing.

By August 2005, the CNGrid project has built such an environment, consisting of eight grid nodes (computing and data centers) spanning six cities in China. The total computing capability exceeds 20 Tflop/s, provided by domestic HPC systems such as Dawning 4000A (10 Tflop/s) and Lenovo 6800 (5 Tflop/s), as well as HPC systems from multinational vendors. Eleven domain-specific application grids are also running, from fields of scientific research, natural resource and environment, manufacturing, and the service sector.

A key ingredient of CNGrid is the CNGrid Software suite. It connects all the users (end users, grid application developers, grid resource providers, grid administrators) and resources (computing, storage, data, and applications software resources) into a uniform, coherent CNGrid environment.

The CNGrid Software suite employs a service-oriented architecture. It consists of three loosely coupled subsystems: the Vega GOS (grid operating system), the GriShield grid security software, and the GridDaEn data grid software. The current hosting environment is mainly Apache Tomcat/Axis, while providing connections to services running on Microsoft Windows platform.

In what follows, we present the main features of the CNGrid Software suite, from the users' viewpoints. We focus on showing how the CNGrid Software suite supports a loosely coupled architecture and dynamic changing nature of grids, while providing single system image and managed services, in a wide-area distributed environment wherein resource providers desiring autonomous control.

H. Jin, D. Reed, and W. Jiang (Eds.): NPC 2005, LNCS 3779, pp. 14 – 21, 2005.

2 CNGrid Software from Five Viewpoints

This section presents the salient features of the CNGrid Software suite from the viewpoints of grid system deployment, grid resource providers, grid system administrators, grid application developers, and the end users. The CNGrid software suite is meant to provide a tool set for users to build application grid systems, according to the users' business model. Therefore, there could be many alternatives. We will focus on typical scenarios.

2.1 Deployment

Before the CNGrid Software is deployed, CNGrid is the sum of two types of isolated resources: grid nodes and application grids. Grid nodes are HPC centers (e.g., Shanghai Supercomputing Center) or campus grids (e.g., Tsinghua University campus grid). An application grid is usually a distributed enterprise application system, such as the Simulation Grid for aviation/space industry. An application grid could have multiple intra-enterprise grid nodes. Currently, CNGrid has eight grid nodes and eleven application grids.

After the CNGrid Software is deployed, these isolated physical resources can be connected into a virtualized, uniform and coherent national grid environment. The deployment process usually consists of the following activities: install the CNGrid Software on grid nodes (typically one copy per grid node), and configure the initial grid system to create needed grid communities (called *agoras*). The CNGrid Software deployment in CNGrid environment is show in Fig.1. The following is a typical configuration seen by users:

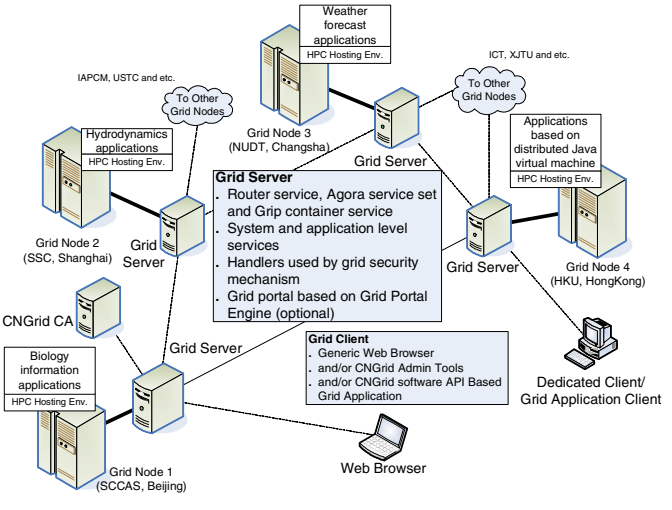

Fig. 1. National wide deployment of CNGrid software

- A single grid system called CNGrid, with its CA system.
- A CNGrid-wide virtual name space, implemented via a set of grid routers.
- One or more agoras. For instance, we could have an agora for each of the grid nodes and another "global" agora for CNGrid, giving nine agoras in total.

2.2 Resource Provider

The Effective-Virtual-Physical space model [2] (show in Fig.2.) implements resource virtualization. This EVP virtualization scheme is compatible with the OGSA 1.0 three-level naming scheme [3]. When a resource provider wants to add a new resource (or connect an existing resource) to CNGrid, he/she sees a grid-wide virtual address space and one or more agoras. A resource provider is responsible for two duties: (1) wrap the resource as a WS-I compliant Web service and connect it into the virtual name space; (2) register the service with one or more agoras, implying that the service can be shared by users in this agora according to the specified policies. The registration process has two aspects: deciding the virtualization mappings and selecting the access control policies. Let us use an example to illustrate virtualization.

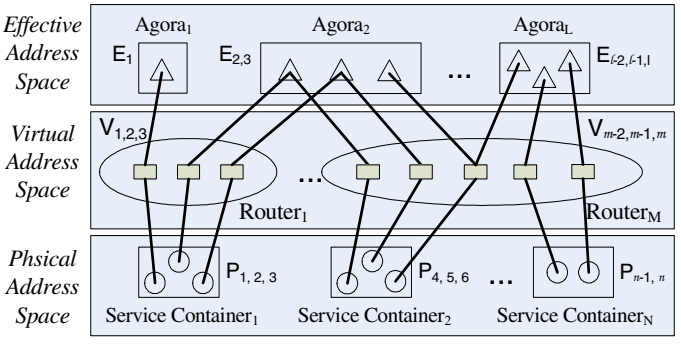

Fig. 2. The Effective-Virtual-Physical virtualization scheme in Vega GOS

The traditional approach to run mpiblast job on a machine is: login a frontend machine by telnet and submit following command to the backend batch system.

mpirun -np *nprocess* mpiblast -p *prog_name* -d *db_file* -i *in_file* -o *out_file*

The above command can be wrapped by general purposed batch service or dedicated one whose interfaces accept this command in a whole or all of the above parameters independently. Multiple such services can be connected and registered into virtual and effective name space, and build up one effective service named as mpiblast service with reduced interfaces. The Parameter Transformer (PT) in Vega GOS that resides on mappings between effective address and virtual address can eliminate the inconsistency at service interface level. In mpiblast case, the *nprocess,*

prog_name, and *db_file* parameters in multiple services can be converted to uniformed ones by separate PT depending on practical situation. As listed below, the code accessing service can be highly reduced at effective layer.

```
...
// "mpiblastEAddr" is effective address of mpiblast
// service. The nprocess, prog_name, and db_file
// parameters are encapsulated behind it.
out_file = mpiblastClient(mpiblastEAddr, in_file);
...
```

2.3 Grid System Administrator

The grid system administrators manage users, resources, and policies via a Web based GUI tool or interfaces to agoras and grid routers. Each administrator can see one agora and its associated grid routers. The management functions include the following:

- Install, configure, and maintain GNGrid Software.
- Add, delete, and change attributes of users.
- Add, delete, and change attributes of resources (especially the EVP mappings of resources).
- Add, delete, and modify policies.

Currently, only two types of policies are supported. They are resource selection policies (e.g., random, FIFO) and access control policies.

2.4 Grid Application Developer

The Vega GOS allows grid application developers to see three levels of details: the effective, the virtual, and the physical levels. Many applications only need to see the effective level, which makes the following information available:

- An agora (with its specific policies and implied mappings to virtual services).
- All the effective services (resources) available in the agora, including system level services (e.g., meta file service and file service) and application level services (e.g., batch service).
- Interfaces to the GPE (Grid Portal Engine), if the developer wants the grid application to provide a presentation layer based on Web technology.
- Five Vega GOS interfaces to services and agora (see Fig. 4).

For example, a weather forecast and visualization grid application is supposed to be developed based on CNGrid Software. As the prerequisite, the weather forecast service and visualization service are developed and registered as effective service in an agora. The only thing that grid application developer needs to do is to integrate application logic flow with these two effective services, and, if the end user wants to submit weather forecast computation and views the graphical results by Web portal, Web pages (.jsp) constructing this grid application logic are needed. The main pseudo code is as below.

```
...
// Create a new grip under GPE's control.
GripClient gc = CreateGripUnderGPE;
// Upload weatherforecast required files to global user
// file space, and get back the global file addresses
// through the grip. The "hotfileEAddr" is the effective
// address of hotfile service.
weather_in_global = Upload(gc, hotfileEAddr,
                                          weather_in_local);
// Compose the weather forecast job description file.
weather_job_xml = weather_job_req + weather_in_global;
// Submit the job to effective weather forecast service
// by grip, and get back global result file addresses.
weather_out_global = JobSubmit(gc, weatherEAddr,
                                          weather_job_xml);
...
// Until weather forecast job finished, compose the
// visualization job description file.
viz_job_xml = viz_job_req + weather_out_global;
// Since the visualization input files are already
// existed in global space, directly submit the
// visualization job to effective visualization service,
// and get back the result.
viz_out_global = JobSubmit(gc, vizEAddr, viz_job_xml);
// Download result files in global space to portal side.
viz_out_local = Download(viz_out_global);
// Display the result at portal side.
Display(viz_out_local);
...
```

2.5 End User

An end user must go through an application/approval process (called user registration) to become a legitimate CNGrid user. Such a user has a certificate and proxy certificate (both GNGrid-wide unique), a home agora, and a user-name/password pair unique within the agora.

Users can log into CNGrid via a common grid portal or a customized client software (e.g., a Matlab client). When a user logs into CNGrid, he/she actually logs into an agora (the home agora by default). There he/she can see and utilize all the effective services (resources) available in the agora, subject to access control policies applied to this particular user. The most common usage scenario for an end user is to look for and utilize a pre-deployed application service. However, CNGrid provides several system level services (utilities) by default:

- A batch job service, which allows jobs to be submitted to the entire CNGrid, instead of a grid node or an application.
- A hotfile service, to allow location transparent access to files.
- GridDaEn data service, a more full-fledged data service than hotfile.

The hotfile services provide a location transparent file space for the users. Each user sees a tree of directories and files under a root "/", with the tree physically distributed across CNGrid.

3 Under the Hood

This section describes some implementation details of the CNGrid Software suite.

3.1 Architecture and Hosting Environment

Learned from computer systems [4], the CNGrid Software can be divided into four layers from bottom up [5]. They are CNGrid hosting environment, core layer, system layer and application layer (as show in Fig.1).

Currently, the CNGrid Software is hosted by J2SE/Tomcat environment, and can be easily migrated to other platforms, such as OMII, WSRF, even the .NET platform.

The core layer is something like OS kernel, provides common functionalities required by grid applications, such as layered service address management, grid user management and grid process (grip) manipulation. Also, the authentication and authorization are included in this layer.

The system layer provides a collection of basic libraries to help programmer developing grid application quickly. The services that shadowed will be gradually appended into this layer.

The application layer is not constructed by services, but by API provided by system layer and core layer. Grid portal developer or integrator can be benefited from Grid Portal Engine by avoiding using system or core layer API directly. GSML (Grid Service Markup Language) software suite is kind of client side service composition and collaboration toolkit which implements the GSML specification 1.0 and provides "on demand" programming environment.

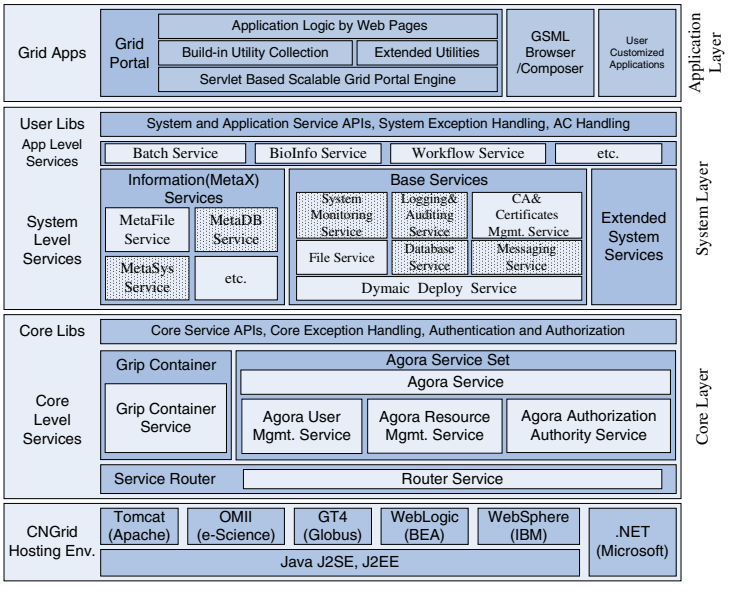

Fig. 3. Hierarchy of CNGrid Software

3.2 Grip, Agora and Router Service

The core layer composed by grip service, agora service set and router service with wrapped client side API; user authentication and service authorization mechanisms implemented by Axis handler chains; and the Vega GOS exception handling extends from the Axis fault which can help the developers accurately locating the service side exceptions and failures.

Aggregated by grip at runtime, agora service sets and router services implement the EVP space model. As show in Fig. 4, the grip client offers only five method calls. Behind these method calls, the grip container service accepts the requests and forwards the them to the agora service set or router service accordingly. When a grip created inside a grip container service, it will retain the information of login user and binding services in grip control block until a close operation is called. During the lifetime of a grip, user can access it at anytime and anywhere. When user is invoking the binding service through a grip, the grip will first resolve the virtual address to physical one, and then invoke the actual service by endpoint. At last, the grip will get back and cache the result of invocation for subsequent retrieval.

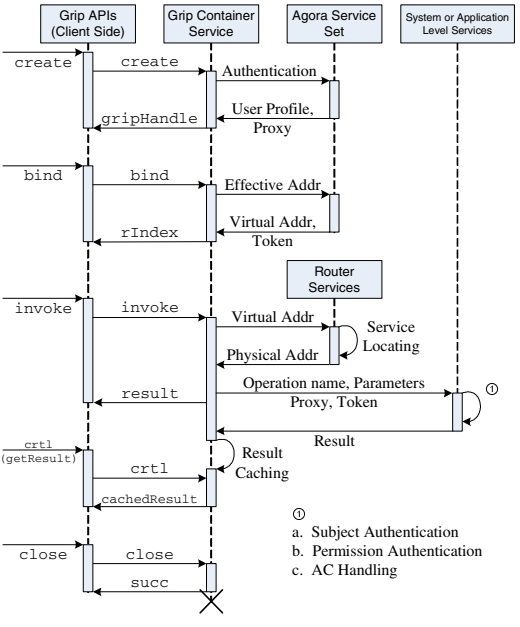

Fig. 4. Sequence diagram of CNGrid Software core

3.3 Security

Inside the CNGrid Software, GriShield is responsible for grid security issues. We have developed a CA service for certificate management, and have implemented WS-Security [6] conformed authentication, authorization, message level secure communication, access control by handler-chains of Axis.

4 Concluding Remarks

A main goal and feature of the CNGrid Software suite is loose coupling, achieved via virtualization and service orientation based on WS-I and OGSA standards. The three main modules, Vega GOS, Grishield security software, and GridDaEn data software are all loosely coupled, not critically depending on each other. The EVP model realizes resource (and service) virtualization, thus separate grid applications from physical resources. It is possible to run an application without changing its code even when the WSDL definition of a referenced physical service changes (location, operations, parameters, etc.).

Another feature of CNGrid is single system image. Once a user logs into CNGrid, he/she can use all available services in a location independent fashion. Furthermore, this capability is realized by the CNGrid system software, not as an application solution.

With the help of the Vega GOS core, agora and grip, the CNGrid API is also virtualized, compared to Web services interface. Application developers only need to understand the five interface calls provided via grip, with many details (including security and policy issues) automatically taken care of by the CNGrid software. Experiences show that it takes about 0.5-2 days to develop/deploy a simple grid application, including presentation, logic and data.

Acknowledgements

We acknowledge contributions by the CNGrid Software team and its users, especially Guangwen Yang, Hao Wang, Peixu Li, Mo Hai, Yanzhe Zhang and Honglei Dai. This work is supported in part by the China Ministry of Science and Technology 863 Program (Grant No. 2002AA104310), the National Natural Science Foundation of China (Grant No. 69925205), and the China National 973 Program (Grant No. 2003CB317000 and No. 2005CB321807), and Chinese Academy of Sciences Distinguished Scholars Fund (Grant No. 20014010).

References

[1] I. Foster, C. Kesselman (Eds.), The Grid 2: Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers, San Francisco, 2004.
[2] W. Li, Z. Xu, A Model of Grid Address Space with Applications, Journal of Computer Research and Development, 2003, Vol. 40, No. 12, pp. 1756-1762.
[3] I. Foster et al (Eds.), The Open Grid Service Architecture 1.0, GGF document, Feb. 2005.
[4] Z. Xu, W. Li, et al., Vega: A Computer Systems Approach to Grid Computing, Journal of Grid Computing, 2004, Vol.2, Issue 2, pp. 109-120.
[5] L. Zha, W. Li, et al., Service oriented Vega grid system software design and evaluation, Chinese Journal of Computers, 2005, Vol. 28, No. 4, pp. 495-504.
[6] OASIS, Web Services Security: SOAP Message Security 1.0, http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf, 2004.3