CNGrid Software 2: Service Oriented Approach to Grid Computing

X. Xie[①], N. Xiao[②], **Z. Xu**[③], L. Zha[③], W. Li[③], H. Yu[③]

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

Abstract

In China, CNGrid software is one of the first middleware level grid software aiming at distributed resource sharing and application integration. It is built by SOA technology and composed by Vega GOS, GriShield and GriDaEn, which are responsible for resource sharing mechanism implementation, security and file management separately. In CNGrid software, we have learned from computer systems, view a grid as a distributed computer system. Following this methodology, we proposed several system techniques such as resource router, grid process (grip), grid community (agora), and the GSML software suite. We discussed these techniques and introduced the implementation accordingly.

1. Introduction

Grid related research in China flourished in recent years. We have witnessed a rapidly growing interest in grid technology in China. The CNGrid software project, sponsored by National High Technology Research and Development 863 Program, is being undertaken by Institute of Computing Technology of CAS, JiangNan Institute of Computing Technology and National University of Defense Technology. The goal of CNGrid software is to support efficient management of multiple geographical distributed grid nodes, to provide a secured, uniformed and friendly interface to the grid users, and to provide a convenient accessing approach to the grid resources from anywhere. CNGrid software has been applied to different application domain by now, from computing sensitive such as distributed simulation in manufacturing, underground water and petroleum resources analysing, and large-scale genome sequencing and analysing, to information scope, such as scientific database, education, and e-governance.

This paper is organized as follows. Section 2 gives general situation on CNGrid software 2 architecture and functionalities provided. Section 3 discusses several key approaches that CNGrid software 2 adopted in order to solve resource locating and management, user interaction and grid security issues. Section 4 presents implementation details of major components. They are Vega GOS, GriShield and GriDaEn. Section 5 offers some concluding remarks.

2. CNGrid Software 2.0

In order to support the goals of CNGrid software project, support autonomous and geographically distributed grid node, the SOA (Service Oriented Architecture) concept is fully utilized and embodied in CNGrid software 2.0 architecture. The computing, storage and information resources can be wrapped as different plain Web services[1], and managed by CNGrid software. Thereafter, the developer can access these services via uniformed interfaces provided by CNGrid software.

2.1 Architecture

2.1.1 Hierarchy

The CNGrid software 2.0 can be divided into four layers from bottom up. 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[2], WSRF[3], 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 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 APIs provided by system layer and core layer. Grid portal developer or integrator can benefit from Grid Portal Engine by avoiding using system or core layer APIs directly. GSML[4] (Grid Service Markup Language) software suite is a kind of client side service composition and collaboration toolkit which implements the GSML specification 1.0[4] and provides "on demand" programming environment.

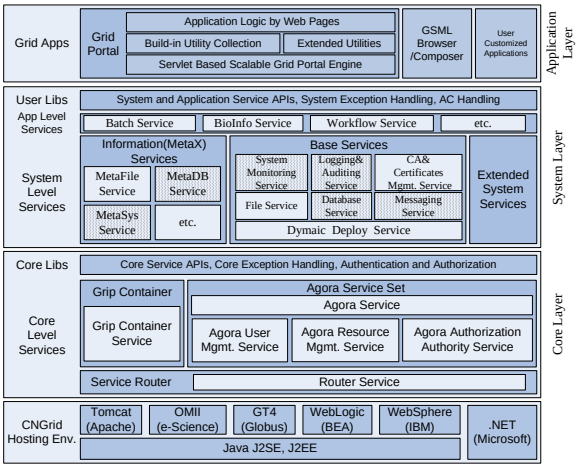

Fig.1. Hierarchy of CNGrid software 2.0

2.1.2 Deployment

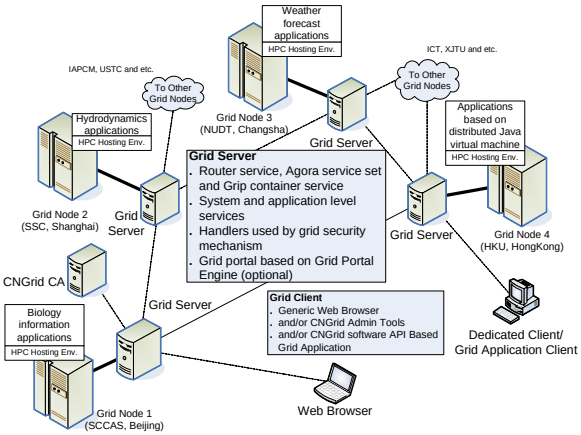

Fig.2. National wide deployment of CNGrid software 2.0

As show in Fig. 2, the CNGrid software 2.0 has been already deployed at 8 major grid nodes. Because of the heterogeneity of mainframes and clusters, each grid node equips a standalone grid server for CNGrid software 2.0 installations, and acts as head server of backend machines. Grid servers can be connected as an overlay network with arbitrary topology by service router which installed in each grid server, so as to manage the services in distributed manner and provide a single system image on service space. The client can be generic Web browser, such as IE and Firefox, or dedicated one based on CNGrid software APIs.

2.2 Features And Functionalities

2.2.1 Virtualized

Virtualization in CNGrid software 2.0 mainly refers to service address (URL) naming and mapping mechanisms, that is to say, services hosted in each grid server will assigned to a readable name when registered to CNGrid software. This name will be mapped to an actual service address dynamically while user accessing virtualized services. During mapping procedure, CNGrid software transparently provides service locating, and provides functionalities such as access controlling, service selection, fault tolerance.

2.2.2 Distributed

Obviously, services managed by CNGrid software are fully distributed. Furthermore, CNGrid software images can be installed in grid servers independently and connected by service routers. Meanwhile, the components, such as application, system and core level services, can be spread around multiple machines for performance consideration. The service oriented and loosely coupled architecture determines that grid environment built by CNGrid software is scalable and adaptable.

2.2.3 Autonomous

Services managed by CNGrid software are kept as autonomous as possible. Services in grid are owned by service provider, and can be managed by grid administrator when authorized to CNGrid software based grid environment. User can access the services via an authorization token assigned by CNGrid software. But the ultimate access control implementation is still operated at service side, which means the service owner will decide who can access their services.

2.2.4 Simple

For grid application developer or integrator, CNGrid software offers a set of basic client side APIs which only include 5 major Java method calls. Through these interfaces, CNGrid software can hide low level technical detail to the developers, such as service binding procedure and security related operations. For Web application integrators, CNGrid software provides a higher level APIs than basic client

side APIs. The interior details of CNGrid software are encapsulated. So the integrator can focus on application logic and UI implementation.

3. Key Approaches

To fulfill the characters listed above, we have proposed several system techniques[5] and applied in CNGrid software 2.0. Fully distributed service routers together with multiple centralized grid communities (agora) implement the layered service virtualization. Grid process (grip) acts as user agent when user wants to assessing the services around grid.

3.1 Layered Service Virtualization

3.1.1 EVP Address Space Model[6]

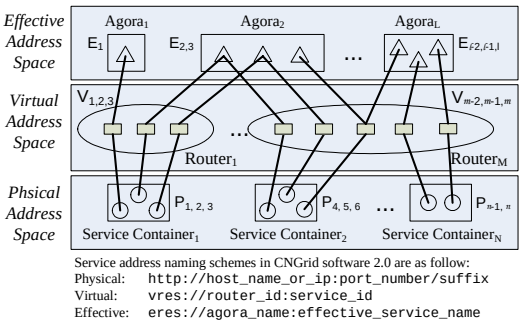

Fig.3. Mappings in the EVP address space model

The EVP (Effective, Virtual and Physical) address space model takes the address space concept in computer system for reference. CNGrid software has implemented this model but with some important changes. As show in Fig.3, from bottom up, *physical address space* consists of Web services called *physical service* hosted in different service containers. The Web service's endpoint called *physical service address* is registered into the service router; the interlinked service routers can form a global *virtual address space* with single system image character. Each registered physical service will get a *guid* from its registering service router as the reference to its attached properties. The *guid* called *virtual service address* will be put into independent agoras manually, and will be assigned a readable name called *effective service address*. An *effective service* is described as a group of customized access control and authorization policies that can be indexed by an *effective service address* in a certain agora. All services in all agoras build up the *effective address space*. Mapping between *virtual address space* and *physical address space* is one to one mapping, while the mapping between *effective address space* and *virtual address space* is *n* to *n* mapping, that is to say, one effective service can be mapped to multiple virtual services and vice versa.

3.1.2 Decentralized Global Service Locating

How to name the physical services is very important when locating physical service globally in grid environment is needed. Unfortunately, location-dependent address makes it harder to handle service location migration. When a physical service wants to change or reassign its access point, the application that binds to this physical service must be rewritten to adapt this change. Therefore, location independent name for physical service is necessary in the grid environment.

The virtual address space depicted above is maintained by decentralized interlinked service routers. Each service router is identified by a global unique id as its name (router_id), which is created automatically only once when the router startups at first time.

We proposed a two-segment naming scheme which represents the virtual service address. When a physical service has been registered to a nearby service router, the router automatically creates a global unique and location-independent id (service_id) for it. A virtual service address is composed by router id and service id which generated by registering router. Hence, Physical services registered in same router have same former segment but different latter segment on its virtual service address.

According to the addressing scheme of virtual service, the physical service locating procedure is divided into two stages. The first stage is responsible for resolving the service router endpoint. After the first stage, service router on which the target physical service registered is known. The second stage is for resolving the service id to the physical service address by directly accessing the target service router.

Due to large scale, distributed and dynamic characters of grid environment, the scalability and adaptability issues must be considered when designing and implementing the service router. Service router is based on the idea that each router stores the information of all routers' access point. The information maintains the one to one mapping between router id and router endpoint, and can help resolving the first part of virtual service address. The information of one to one mappings between service id and physical service address are kept in service router separately, and can help resolving the

second part of virtual service address. So the key issue is how to keep the consistency on global router information among all routers.

Our approach learns from RIP algorithm in IP routing and quite simple. Each router will timely update its global router table from its directly linked routers which called neighbors. When a router is added into or removed from router network, the information changes will propagate to all the routers after a certain time by this updating procedure. Consequently, this approach can ensure that router's entering or leaving will make no affects to other router that still running.

Since the effective address space is built upon virtual address space, it is necessary to keep the virtual address space as stable as possible in order to accommodate changes in physical service address, and provide a fixed view to the upper space, even to the applications. According to our router semantics descript above, when a registered physical service wants to change its registering location to another router, it will get a new virtual service address from the target router. In this case, we have provided a symbol link method that allows physical service changes the registering location but remains the original virtual address.

3.1.3 Centralized Local User, Effective Service And Policy Management

User and effective service information are maintained by centralized *agora* (grid community). We proposed a role based user manage system (UMS for short) in agora. Each registered user will get an external user name and an internal one. The internal ones is the distinguished name (DN for short) that extracted from user certificate, while the external ones along with password is for user login and can be multiple. For security consideration, UMS only stores user proxy certificate[7] that uploaded by registered user. It will be pulled into grip when user login with an external user name and password. Simultaneously, the external user name will be mapped to internal user name for being used inside the CNGrid software.

In agora, several virtual services can be manually grouped into one effective service by administrator. The address mapping between them will be stored permanently. Although the grouped virtual services have same functions, we can not guarantee that they all have the same interfaces. So, we use adaptor approach to eliminate the differences. The adaptor is called Parameter Transformer and applied to the mappings between effective services and virtual services.

Since the mapping between effective address space and virtual address space is n to n, we proposed service selection and authorization policy engine approach to determine which virtual service address will be returned when user is accessing the effective service. The service selection algorithms can be configured and customized. After doing the service selection, the engine will filter out the virtual service that user have the access permission.

By the service grouping and policy mechanism, agora can hide the complexity of service selection and authorization. Centralized management on user, effective service and policy cooperating with service router presents an integrated EVP address space model.

3.2 Dynamic Grid User Agent

We hope that when user is accessing the service in grid, he/she knows as little as possible about the information of physical services. With the EVP address space model, we can hide the information of physical service address to solve the service change problem. Meanwhile, other problems which the user may encounter should be considered. They are as follow: (1) The grid user does not care about the process of address mapping. It is the job of the grid software. There must be exist a module in grid software that can comprehend the interfaces of agora, and call the corresponding interface to complete the address mapping from effective service address to virtual ones. In this process, this module may realize the service selection when an effective service is mapped to multiple virtual services. Then, this module must interact with service router to complete the address mapping from virtual to physical. (2) The user does not care about the variety and detailed information of the physical services, but is aware about the functions that services provide. The grid software needs to get the information of the physical service in order to finish the service accessing. For example, the physical service may be configured as secured which requires getting the user identity to do access control. So, this module should decide whether the user identity should be added or not into the invocation request. (3) The grid software should support convenient services even resources sharing and collaboration. Therefore, the system needs to maintain the user data structure and context in order to use this information implementing the authorization, authentication, and access control dynamically.

As a result, we proposed a runtime construct called *grip* (grid process), which is an abstraction that corresponding to process in a

traditional OS, and runs on CNGrid software, representing a grid subject to access various services in grid. In another words, other components in CNGrid software, such as service router and agora, seem like the brawn which provide many functions; but grip seems like the brain, which makes use of the functions provided by the brawn instructed by the user's requests. At present, the grip presents 5 client side APIs, they are: *create*, *bind*, *invoke*, *control* and *close*. The developer can program the code with the 5 APIs simply and easily.

4. System Implementations

The CNGrid software 2.0 is constituted by Vega GOS, GriShield and GriDaEn. The following is the implementation details about these major modules.

4.1 Vega GOS

The Vega GOS is an open framework that follows the SOA concept[8], and can be downloaded from Vega site for free (http://vega.ict.ac.cn). This framework is divided into layers (as show in Fig. 1) according to different functionality partition. The core layer solves the essential issues that extract from common requirements of grid applications, such as address spaces and grid process mentioned in section 3. More like the system software in traditional OS, the system layer serves as collection of user libraries and toolkit which are developed upon the abstractions provided by core layer. The application layer is the nearest layer away from the end user. The end user can use the applications in this layer accessing the service in grid environment unaware of the things occur in lower layers. All the services in CNGrid software 2.0 are based on Axis 1.2RC2; and all the client side APIs are provided by Java class libraries.

4.1.1 Core Layer

The core layer composed by grip service, agora service set and router service with wrapped client side APIs; 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 address space model that discussed in section 3.1. As show in Fig. 4, the grip client offers only 5 method calls. Behind these method calls, the grip container service accepts the requests and forwards the requests 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 invokes the binding service through a grip, the grip will first resolve the virtual service address to physical one, then invoke the actual service by endpoint. At last, get back and cache the result of invocation waiting for retrieval.

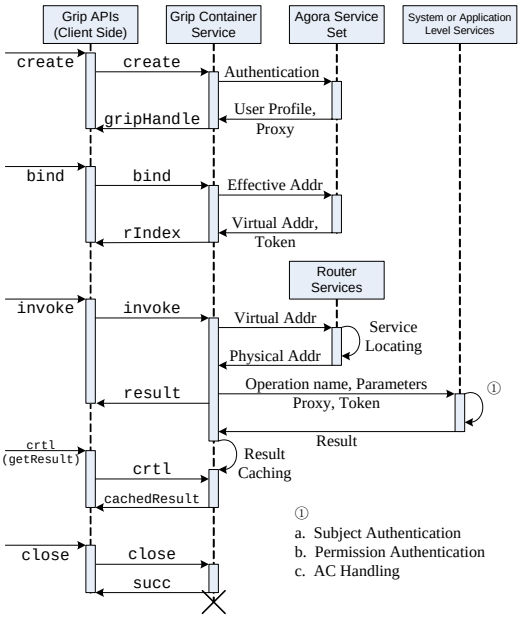

Fig.4. Sequence diagram of CNGrid software 2.0 core

4.1.2 System Layer

The system layer includes two categories of service. One is called MetaX service which aggregates the information that comes from resources wrapped in service, and the other is Base service which provides a bunch of functions needed for grid application developer. We have implemented the dynamic deploy service, grid batch system with accounting, grid file management system and so on.

The dynamic deploy service can remotely deploy a service into its hosted container but need not to stop it. It is done by extending the class loader mechanism of Axis. Supported by dynamic deploy, applications can deploy services on demand at runtime without grid administrator's intervene. For example, a application that does blocked matrix multiplication can dynamic deploy several matrix multiplication service to grid at runtime, Then partitions the matrixes into blocks according to the number of service deployed , and do the computation in parallel.

The grid batch system consists of batch service, accounting service and batch driver for backend batch system (OpenPBS, LSF and so on.) interaction. The batch service has the interface of job submission, job status query, job cancellation or deletion. File stagein and stageout are supported by grid file management system (GriDaEn) which will be described in section 4.3. When a job description document is sent to batch service, the batch service will transfer the request to job script and submit to backend batch system. If the request contains file stagein or stageout requirements, the batch service will download/upload the files from/to user global file space maintained by GriDaEn. When a job finished, the batch service will put a record about computing and storage resource usage into accounting service for later retrieval.

4.1.3 Application Layer

In the application layer, CNGrid software offers two interaction oriented toolkit that helps people transparently accessing the services in grid.

· **Grid Portal Engine**

Grid Portal Engine (GPE for short) provides a mechanism for the integration of servlet/JSP based Grid portal applications. Integration means applications access address spaces using grips managed by GPE.

Grip management, which is the most important function GPE focused on, includes: (1) Lifecycle management, and (2) Invocation management.

Firstly, the creation and close of grips are managed by GPE instead of portal applications. As explained later, GPE chooses which grip to use based on each request and there is no session/lifecycle issues that portal applications have to care about.

Secondly, GPE provides a mechanism called grip-encoded URL, means encoding the name of grip in request URLs, for clients to specify which grip should be used in a request, which frees up GPE and portal applications from maintaining bondage between Grips and HTTP sessions. With grip-encoded URL, GPE can choose which grip to use based on each request, avoiding potential conflicts between session states of GPE and portal applications.

Besides grip management, another important feature provided by GPE is user identifying. GPE maintains the mapping between HTTP sessions and user identities, by which GPE provides user identity information of a request for portal applications. GPE also uses this user identity information to make access control decision internally; for example, one can not make an invocation on other's grip.

Some portal applications may run on behalf of some other user than the one logged in and remembered in the mapping mentioned above; GPE provides a mechanism like setuid in UNIX-like operating systems to fulfill this need.

As far as session is concerned, GPE uses grips as a kind of distributed user session and does not store any data in HTTP session provided by servlet containers. GPE only uses HTTP session as mechanism of user identifying as mentioned above.

GPE does its work by several Servlet filters, which can be reconfigured easily without modifying applications. GPE uses dependency injection containers to provide APIs for portal applications, concealing dependency modification internal GPE in the future.

· **GSML software suite**

Grid Service Markup Language (GSML) is an XML-based Grid programming language, which allows end-users to program grid on demand. Based on reusable components and asynchronous event-driven model, GSML is capable to describe flexible, diverse grid application logic, and provides collaboration frameworks. The goal of GSML is to provide an easy-to-use Grid programming language, which not only can integrate Grid resources flexibly and efficiently, but can also provide infrastructures for collaborations between Grid applications, resources and end-users.

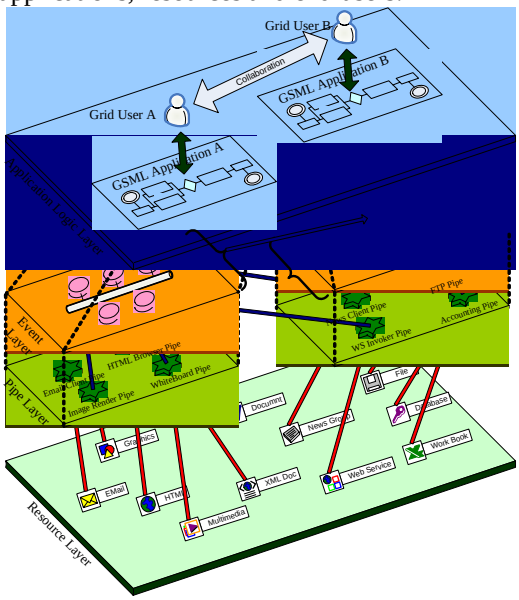

Fig.5. GSML language structure

The core concepts, Pipe, Event, EventSet (as show in Fig.5) is enlightened by pi Calculus. Using these concepts, end-users can organize the interactions between various software components and services uniformly with asynchronous event-driven communication models. GSML software suite includes GSML Composer and GSML Browser. GSML

Composer enables visual and intuitive Grid programming; GSML Browser provides runtime environment for convenient execution of GSML applications.

4.2 GriShield

In grid environment, the population of user and resource (service) is large and dynamic. Participants in grid want to be as flexible as possible. Meanwhile, security must be enabled to protect the user and resource confidential. From user point of view, they do not want to enter password again when change the accessing resource. From resource provider point of view, they want to decide by themselves who can go through. To summarize, the focus of GriShield is to provide a loosely coupled and complete security solution, which can be plugged into CNGrid software and integrated into heterogeneous grid environment. In this solution, the traditional techniques in security will be used, such as PKI, algorithms of encryption and signature, even the matured third party software, such as OpenSSL and xmlsec.

CNGrid software and applications primarily requires functions of authentication, authorization and access control. Especially, the GriShield seeks to (1) provide authentication solutions that allow user and the resources accessed by that user can verify each other's identity; (2) provide agora based multi-grain authorization mechanism, and (3) allow local access control mechanisms to be integrated into GriShield without changing.

Inside the GriShield system, we have developed a CA service that responsible for certificate management, and have implemented WS-Security[9] conformed authentication, authorization, message level secure communication, access control by handler-chains of Axis.

The extensible message processing model is the core of GriShield. This model uses handlers and handler chains of Axis that can enable the functionality to be tailored to satisfy wide variety of situations and requirements. A handler is an atomic component that will operate on a specified part of SOAP message. For example, a handler can be in charge of performing authentication on message sender before allowing it to be processed by the provider. A special handler, the pivot handler (another name for the service's provider), is in charge of executing the service implementation logic. It is called pivot handler because it is where the message's processing cycle changes from request processing to response processing.

As show in Fig.6, we have implemented the following 7 handlers in GriShield:

SignHandler implements the signature procedure of SOAP message and it can operate on both incoming and outgoing messages.

AddHandler can add GOSContext to SOAP message into the SOAP attachments. GOSContext is a common system object storing agora name, certificate or proxy certificate (with key), and token in a structured manner. Agora name is string type, which denotes the name of agora that user registered. The certificate is `byte []` type, which denotes cert or proxy cert of user. Token is `byte []` type too, which denotes a SAML based authorization token or a X.509 based authorization attribute certificate. The token with signature is issued to trusted service invocation request by agora authorization authority service dynamically.

WSSecurityHandler implements signature verification of SOAP message.

GetAttachmentsHandler can get the GOSContext from SOAP message from its attachments.

VerifyCertsHandler verifies certificate or proxy certificate of user stored in GOSContext object.

VerifyTokenHandler verifies token in GOSContext.

ACHandler implements the access control at agora service or physical service side. Service side ACHandler can be replaced by customized ones. The new ACHandler will accommodate with service provider's local security policy.

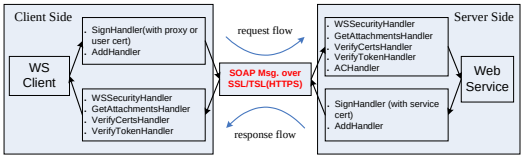

Fig.6. E2E handler-chains and message flow in GriShield

4.3 GriDaEn

Grid file management in CNGrid software called GriDaEn which provides a uniform file resources view and high level interface to access heterogeneous file resources. GriDaEn is composed of two modules: meta service for metadata management and file service for local file system management and file transportation.

By GriDaEn, every agora user possesses a logic, continuous and independent file space. So, different user actions on files will be mapped into different file spaces. A global shared space in an agora is also supported for public accessing. We allow granting and revoking operations to support file sharing among different users. File owners can grant or revoke

the access permission of their files to other user or group in same agora.

Similar to the EVP address space model, we have developed a grid file storage resource space model. It is composed of three layers: *Physical storage space*: the collection of storage resources at grid nodes which can be distributed and heterogeneous; *Virtual storage space*: the abstraction of physical storage resources provides uniform accessing interface; *Effective storage space*: provide a convenient interface for programmers or end users.

On the basis of grid file storage resource space model, files in GriDaEn are divided into layers. *Grid physical file*: the files stored at local file system (e.g. NTFS on Windows or EXT3 on Unix). *Grid virtual file*: global identifier of grid file, which is similar to the file handle in Unix file system or the public file handle in NFS. *Grid effective file*: visible and operable by end user, and exclusive in users' file space.

Effective file name, *virtual file name* and other file information are stored in meta service. Before grid user accessing a file in GriDaEn, meta service should be invoked to do name transform from effective to virtual. Meta service also offers some interfaces on meta information operating such as list and search.

The file service is equipped with two channels to separate the control and data transport protocol. The control channel is implemented by file service interfaces (operations), whereas the transport channel is based on servlet that attached to file service for performance consideration. Therefore, transfer of files can be configured as HTTPS mode when a secured transport is needed. Using HTTP/HTTPS in GriDaEn solved the firewall problem comparing to other widely used grid file management system, such as GridFTP and SRB.

Currently, supported by Vega GOS, GriDaEn allows one meta service and multiple file services combining to construct the "grid file system". Distributed and heterogeneous local file system can enter or leave this big "grid file system" as will. For grid application developers, GriDaEn is an extensible grid file management system that provides virtualized and uniformed file space view, and provides a high level interface to access heterogeneous files systems.

5 Conclusion and Future Works

Guided by computer system approach and SOA concept, CNGrid software implements several system techniques, such as service address spaces, service router, grid process (grip) and grid community (agora). We have also developed grid security mechanism (GriShield) and grid file management system (GriDaEn) based on these system abstractions that implemented in Vega GOS.

The approaches and implementations discussed in this paper reveal several advantages. The CNGrid software achieves virtualization and hides system details from users. It helps applications to seamlessly adapt to resource changes. It helps separate concerns of application logic and interaction, policy and context, and resource naming.

The CNGrid software 1.1 and 2.0 are already deployed and being used in the China National Grid project. After we gain more application experiences and perform extensive testing, some of the code will be released in an open source form to the grid research community.

Reference

1. W3C, *Web Services Architecture*, http://www.w3.org/TR/ws-arch/, 2004.2.11
2. OMII, http://www.omii.ac.uk/, 2005.3
3. Globus Alliance, *The WS-Resource Framework Version 1.0*, http://www.globus.org/wsrf/specs/ws-wsrf.pdf, 2004.3.5
4. LIU Hao-Zhi, YU Hai-Yan et al., *GSML: An Interaction and Collaboration Oriented Grid Programming Language for End-users*, Chinese Journal of Computers, 2005, 28(4): 704~711
5. Zhiwei Xu, Wei Li, et al., *Vega: A Computer Systems Approach to Grid Computing*, Journal of Grid Computing, 2004, Vol.2, Issue 2：109~120
6. LI Wei, XU Zhi-Wei, *A Model of Grid Address Space with Applications*, Journal of Computer Research and Development, 2003, 40(12): 1756~1762.
7. V. Welch, I. Foster, et.al. *X.509 Proxy Certificates for Dynamic Delegation*. In: Proceedings of 3rd Annual PKI R&D Workshop, 2004.
8. ZHA Li, LI Wei, et al., *Service oriented Vega grid system software design and evaluation*, Chinese Journal of Computers, 2005, 28(4): 495~504
9. 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