Grid middleware in China

Yongwei Wu*

Department of Computer Science and Technology
Tsinghua University
Beijing, 100084, China
E-mail: wuyw@tsinghua.edu.cn
*Corresponding author

Chunming Hu

School of Computer Science and Engineering
Beihang University
Beijing, 100083, China
E-mail: hucm@buaa.edu.cn

Li Zha

Institute of Computing Technology
Chinese Academy of Sciences
Beijing, 100080, China
E-mail: char@ict.ac.cn

Song Wu

School of Computer
Huazhong University of Science and Technology
Wuhan, 430074, China
E-mail: wusong@hust.edu.cn

Abstract: Grids aim at constructing a virtual single image of heterogeneous resources and provide uniform interface for distributed internet applications. China also devotes much passion and endeavour to the evolution of grid projects. Based on the experience in building and enhancing Chinese grids in collaboration with colleagues from around the globe, it is important to choose grid technologies that support and work on a wide variety of resources.

Grid middleware research and development in China is described in this paper. First we give the overview of the three government-sponsored grid programmes, namely, China National Grid, ChinaGrid and NSFCGrid. Then three representative grid middleware, ChinaGrid Support Platform (CGSP), China National Grid Operation System (GOS), China Research and Development Environment Over Wide-area Network (CROWN) are introduced in detail from six aspects: design motivation, architecture, function modules, main features, interoperability, current status and applications. Finally, we abstract the main characteristics of these three middleware systems and put forward a comparison among them.

Keywords: grid; grid computing; grid projects; grid middleware.

Reference to this paper should be made as follows: Wu, Y., Hu, C., Zha, L. and Wu, S. (2007) 'Grid middleware in China', *Int. J. Web and Grid Services*, Vol. 3, No. 4, pp.371–402.

Biographical notes: Yongwei Wu received his PhD degree from the Chinese Academy of Sciences in 2002. Then, he worked in the Department of Computer Science and Technology, Tsinghua University. His research interests include grid computing and distributed and parallel computing. From 2004, he has been involved in the research and development of the ChinaGrid Support Platform.

Chunming Hu is part of the research staff in the Institute of Advanced Computing Technology at the Beihang University, China. He received his PhD degree from the School of Computer Science and Engineering of the Beihang University in late 2005. His research interests include peer-to-peer and grid computing, distributed systems and software architecture.

Li Zha graduated with a PhD degree in the Department of Computer Science and Technology at the Beijing Institute of Technology in 2003. From 2003, he has worked as the leader of the VEGA grid system software project. His research focused on large-scale distributed resource management, which includes naming, organisation and policy mechanisms. His interest also includes other classic issues in distributed computing and the grid computing field.

Song Wu got his PhD degree in Computer Engineering from the Huazhong University of Science and Technology in 2003. He is now an Associate Professor in the same university and the Vice Head of the Computer Science and Engineering Department. He has been a member of the ChinaGrid team from 2003. His research interests include grid computing and distributed storage system.

1 Introduction

Grid computing (Foster and Kesselman, 1999), which focuses on distributed large-scale resource sharing, provides a new solution for professional and contemporary users who want to effectively share and collaborate among themselves in the distributed and self-maintained Virtual Organisations (VOs) (Foster *et al*., 2001).

In the past decade, both the research and industry fields showed great interest and passion to push forward the quick growth of grid computing. Many governments and organisations devote huge efforts to grid-related projects. The most famous ones from research institutions are: Globus,[1] TeraGrid,[2] GIG,[3] and OSG[4] in the USA; e-Science Program[5] and Open Middleware Infrastructure Institute (OMII)[6] in the UK; GridCOMP,[7] EuroGrid[8] and Enabling Grids for E-science (EGEE)[9] in the European Union; National Research Grid Initiative (NAREGI)[10] in Japan and K*Grid in Korea. In addition to these projects, there are some projects supported by other strengths: Unicore[11] Program by open source and GPE[12] by Intel, for example.

It is important to choose grid technologies that support and work on a wide variety of resources, which are heterogeneous in terms of various factors including architecture, instruction set, configuration, node operating system and local resource managers. Grid middleware provides a series of cooperating programmes, protocols and

agents designed to help users access the distributed resources transparently. There are various grid middleware developed by the grid project introduced above. The most influential middleware are described below.

Globus Toolkit by Globus Alliance includes software services and libraries for many components, such as distributed security, resource management, monitoring and discovery, and data management (Ghalem and Yahya, 2007). Coordinated TeraGrid Software and Services (CTSS) by TeraGrid creates an integrated, persistent computational resource for open scientific research. Storage Resource Broker (SRB) by SDSC presents the user with a single file hierarchy for data distributed across multiple storage systems, which has features to support the management, collaboration, controlled sharing, publication, replication, transfer and preservation of distributed data.

GLite by EGEE provides a bleeding-edge, best-of-breed framework for building grid applications tapping into the power of distributed computing and storage resources across the internet. MyGrid by OMII-UK is building high-level services for data and application resource integration such as resource discovery, workflow enactment and distributed query processing. XtreemOS supported by the European Commission's IST programme is an open source grid operating system with native support for VO and capable of running on a wide range of underlying platforms, from clusters to mobiles. NAREGI middleware makes the various computational resources (heterogeneous high-performance computers, high-end servers) appear as one huge virtual computing resource to users. The Grid Component Model (GCM) implemented by GridCOMP makes it possible to seamlessly compose applications and services deployed on large-scale infrastructures, *e.g.*, several thousand machines all over the world.

Unicore is now developing its latest version of software: Unicore 6.0. At the same time, GPE has developed several implementations for other famous grid middleware, for example: GPE4GTK. Other famous middleware products by companies include WebSphere[13] by IBM, Grid Engine[14] by Sun, EGO/Symphony[15] by Platform and WebLogic[16] by BEA.

In this paper, we try to roughly depict the whole picture of the grid middleware research and development in China, and briefly introduce several key grid projects in addition to grid middleware systems and give a glimpse of each of them. Then the remainder of this paper is organised as follows: Section 2 presents the research status of the Chinese main grid middleware. The three key Chinese grid middleware, CGSP, GOS and CROWN are described in detail in Sections 3, 4 and 5. Section 6 abstracts their main characteristics. Finally we conclude this paper in Section 7.

2 Main Chinese middleware

Just like the rest of the world, China also devotes much passion and endeavour to the evolution of its grid projects and grid middleware. There are three nationwide projects in China. They are China National Grid (CNGrid),[17] China Education and Research Grid[18] and Natural Science Foundation Grid (NSFCGrid). In this section, we will take a glimpse at these related research projects in China.

2.1 Grid projects in China

2.1.1 CNGrid

The China National Grid, which is supported by the major project of National High-Tech R&D Program (863 programme) of the Ministry of Science and Technology of China, is a test bed which integrates high-performance computing and the transaction processing capability of a new generation of information infrastructure. Through resource sharing and technological innovation, CNGrid can effectively support scientific research, resources and environment sharing, advanced manufacturing, information services applications and promote the country's development of the information industry and related industries.

China National Grid is equipped with independently developed grid-oriented high-performance computers (Legend deep Tang 6800, Dawn 4000A). These eight nodes, including one in Hong Kong, constitute an open grid environment through self-development of grid software. CNGrid is also responsible for the run of grid-operating environment, grid middleware development and the construction of the application grid.

2.1.2 ChinaGrid

ChinaGrid (Jin, 2004), an important 211 project in the Tenth Five-Year Plan Period of the Chinese Ministry of Education, aims at constructing a public service system for higher education. Through developing corresponding grid middleware, ChinaGrid aims to integrate heterogeneous mass resources distributed in the China Education and Research Network (CERNET), shares those resources in the CERNET environment effectively and avoids the resource islands, provides useful services, and forms the public platform for research and education in China.

The first phase of ChinaGrid is from 2003 to 2006, and the second phase is launched in 2007. The ChinaGrid member universities also expanded from an initial 12 to 20 in the first phase. To this day, the computing power gathered by ChinaGrid has reacted 170 teraflops and the storage capacity has exceeded 170 terabytes. In ChinaGrid, there are five typical applications: Bioinformatics Grid, Computational Fluid Dynamics Grid, Massive Information Processing Grid, Realcourse Grid (remote high education courses online) and Image Processing Grid.

2.1.3 NSFCGrid

Network-based e-Science Environment (NSFCGrid) is a research programme funded by the Natural Science Foundation Committee of China. The programme started in 2004, and will end by the end of 2007. The main goal of this programme is to build a virtual science and experiment environment to enable a wide-area research corporation such as large-scale computing and distributed data processing. The research projects are organised into three layers: basic theory and principles, general test bed and pilot applications. Different from CNGrid and ChinaGrid, NSFGrid pays more attention on the basic research of grid-related technologies.

The NSFCGrid (CROWN test bed) integrates 41 high-performance servers or clusters distributed among 11 institutes in five cities (by April 2007). They are logically arranged in 16 domains of five regions by using the CROWN middleware. The testing

environment is growing continuously and is becoming much similar to the real production environment. The NSFCGrid (CROWN test bed) will eventually evolve into a wide-area grid environment both for research and production.

In addition to the three projects mentioned above, there are still some projects deserving mention here such as the ShangHai Grid, and Tsinghua Grid (TG). The Shanghai Grid project, supported by the Science & Technology department of Shanghai, aims to study the key technique of information grid, develop information grid system software and application supporting environment, and establish an information grid to support 'city traffic service system' that supplies the best traffic route. TG, as one representative of Chinese campus grids, is one of the earliest efforts in grid computing in China. It aims at aggregating various computing resources on the Tsinghua University campus into a powerful computational environment and providing unified and transparent interface for users.

2.2 Chinese key grid middleware

With powerful support of these projects, there are numerous grid middleware showing up. We will describe them briefly in this section then go deep into the details of three of them from Section 3 to Section 5.

2.2.1 CGSP

ChinaGrid Support Platform (CGSP)[19] (Wu *et al.*, 2005) is a grid middleware developed for building the ChinaGrid. It integrates all sorts of heterogeneous resources, especially education and research resources distributed over the CERNET, to provide transparent and convenient grid services for science research and high education. In moving towards this end, CGSP is developed based on the following motivations:

- Provide a platform for grid construction from the top portal to the integration of bottom resources of the grid. Not only does CGSP support the uniform management of heterogeneous resources, but it also supplies the portal building, job defining, service packaging and grid monitoring.
- Support secondary development of grid service and improve the flexibility of the system. Parallel programming interface and its running environment supply the complicated application development based on deployed services in the grid.
- Follow the latest grid standard and integrate existing advanced technology to avoid reduplicated works.
- Provide an extensible and reconfigurable grid framework, in order to fit the purpose of ChinaGrid to cover the top 100 universities of China in the near future, and satisfy the autonomy of each ChinaGrid application or unit.
- Avoid unnecessary data delivery over the grid. Data required by the computing job not in stream with the job description file is delivered to the personal data space in data manager first and real computing nodes could get it directly when the job begins to be executed.

2.2.2 GOS

Grid Operating System (GOS)[20] (Zha *et al*., 2004; Xu *et al*., 2005) serves as the underlying system software of the CNGrid. It supports the loosely coupled, dynamic and autonomous nature of the grid, while providing single system image and managed services in wide-area distributed environment, GOS implements a novel resource virtualisation mechanism to cope with computing, data, software and combined resource-sharing issues, and provides secured, uniformed and friendly interfaces when accessing these scientific computing and information integration resources, such as high-performance computer centres (called grid nodes), file storage and databases. By satisfying common requirements, different domain-specific applications are running on CNGrid now.

Rather than taking the middleware or network approach, GOS takes the computer systems approach (VEGA) to guide its design. Similar to traditional operating system, it focuses on four key issues: naming, process/states, VO (something like file system), and programming. Accordingly, GOS puts more emphasis on key abstractions and software architecture, but not on some specified protocol or service development.

2.2.3 CROWN

CROWN[21] is short for *C*hina *R*esearch and *D*evelopment *E*nvironment *O*ver *W*ide-area *N*etwork. Generally, it includes a grid middleware suite to build a service-oriented grid system, and a distributed CROWN test bed to enable the evaluation and verification of grid-related technologies.

CROWN service grid middleware is the kernel on which to build an application service grid. Basic features of CROWN are listed as follows. First, it adopts an OGSA/WSRF compatible architecture; second, considering the application requirements and the limitation of security architecture of OGSA/WSRF, more focus is put on the grid resource management and dynamic management mechanism in the design stage. A new security architecture with distributed access control mechanism and trust management mechanism, which are proposed to support the sharing and collaborating of resources in a loosely coupled environment is proposed in CROWN.

There are also many other efforts for grid middleware in China. Campus grid, one independent grid software developed by the Tsinghua University, was one the earliest efforts in grid middleware in China. GridDAEN (Xiao *et al.*, 2003) is a data grid middleware that can integrate various kinds of file systems and provide uniform seamless access to distributed datasets. HowU, one network computing platform developed by the Huazhong University of Science and Technology, is a global computing paradigm based on a volunteer computing model. DartGrid (Huang *et al*., 2003; Wu *et al*., 2003), built upon semantic web standards and grid technologies, aims at providing a semantic heterogeneous solution that addresses the challenge of web-scale cross-enterprise database integration.

3 CGSP

3.1 Design motivation

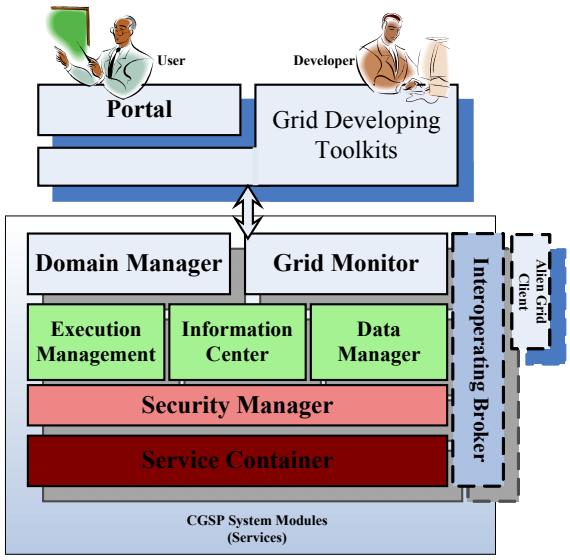
China Education and Scientific Research Grid Project – ChinaGrid aims at constructing a public service system for higher education. Through developing corresponding grid middleware, ChinaGrid integrates heterogeneous mass resources distributed in the CERNET, shares those resources in the CERNET environment effectively and avoids the resource islands, provides useful services, and finally forms the public platform for research and education in China. CGSP is a grid middleware developed to build the ChinaGrid. It integrates all sorts of heterogeneous resources, especially education and research resources distributed over CERNET, to provide transparent and convenient grid services for scientific research and education. More specifically, CGSP is developed based on the following motivations:

- Providing a platform for grid construction from the top portal to the integration of bottom resources
- Providing an extensible and reconfigurable grid framework, in order to fit the purpose of covering the top 100 universities of China in the near future while satisfying the autonomy of each unit
- Following the latest grid standard and integrating existing advanced technology
- Supporting independent development of grid service and improving the flexibility of the system.

3.2 Architecture

You can see the overall structure of CGSP in Figure 1.

Figure 1 CGSP architecture



The CGSP software platform meets the OGSA and the WSRF criteria. All of the resources such as software, hardware, storage and network are abstract to the form of service. Through the difference of service, it supports the foundation and premise for the sharing and cooperation of the resources.

The Container produces a basic environment for the installation, deployment, running and monitoring of services, especially CGSP kernel services. The Security Manager focuses on the user identity authentication, identity mapping, service and resource authorisation, and secure message passing between CGSP nodes. The Information Centre provides service registration, publishing, metadata management, service querying, service matching and resources status collection for the CGSP services in a uniform way. The Data Manager's function is to shield users from the heterogeneousness of underlying storage resources to provide a uniform storage resource access mode. Execution Management accepts users' job execution requests, and invokes relevant service according to job description, and manages the job in its lifetime. The Heterogeneous Database (HDB) aims to enable users in the grid environment to access various heterogeneous database in a uniform way. Domain manager is in charge of the user management, log and accounting, and user identity mapping between different CGSP domains. The grid monitor, named ChinaGrid Super Vision (CGSV), mainly focuses on the monitoring of CGSP resources load, quality of services, user actions, job status and network, to ensure system running normally, and to enhance grid performance as well. Portal is a webpage for visiting current CGSP service and also for providing some support for new grid applications. GridPPI is a set of APIs that enable end users, especially developers, to write grid-enabled parallel programs.

3.3 Function modules

3.3.1 Container

Container is the core component of CGSP that will be put on every node. As a basic runtime environment of the services and supply 'service-oriented facilities for computing', container is put on every node which supports the services.

According to the requirements of ChinaGrid, the Container should supply not only the core functions of the services management, such as remote deployment, runtime management, and service status monitor, but also hot deployment and dynamic deployment of service. Hot deployment means that after the service being deployed, there is no need to restart the Container to enable the service to be invoked by users; as hot deployment is in the case, we need to employ a lock mechanism to ensure entirety of service transaction.

3.3.2 Information Centre

Information Centre (IC) is responsible for providing information of all resources in ChinaGrid, which have the nature of considerable diversity, dynamic behaviour, great distribution and large number. The difficulties of design and implementation of the IC come from the following goals:

- To support all kinds of resources
- To reflect the nearly recent state of resources
- To support query both within a domain and across domains
- To provide acceptable performance in large-scale use cases.

Information Centre consists of the following components: Domain Information Service (DIS), Domain Registry Service (DRS), Node Registry Service (NRS) and Schema Repository Service (SRS).

DIS provides information about domains in ChinaGrid. There are two kinds of information about domains: the topology of domains, which indicates how the domains are connected, and the configuration and deployment information of each domain, which primarily concerns the addresses of CGSP modules, such as Job Manager, Data Manager and so on.

DRS provides information about all kinds of resources. Based on DIS, DRS provides a global view of all resources, including computing elements, services, databases, devices, *etc.*, within all domains in ChinaGrid. The schema used to describe resources in DRS is provided by SRS.

NRS is responsible for maintaining information of resources and services in a local node, and registers this information to DRS. To make the resource or service information visible in a domain, it must be registered to the IC. NRS, deployed on each computing node, automatically collects the service information in the CGSP container, resources and services and periodically reports the collected and registered node information to the IC's DRS.

SRS stores all the schemas utilised to describe resources, each of which defines how to describe a category of resources. Thus, a schema is associated with one or more categories. The resources registered to Information Centre should follow a schema in SRS according to its category, and users can make a query to DRS according to this schema.

3.3.3 Security Management

Security Management in CGSP is designed for guaranteeing that the grid resource can be registered and deployed by those who are authorised to do so. It offers protection for the container, services and resources in CGSP, and also the security of message transmission during all the communication between the different entities of CGSP.

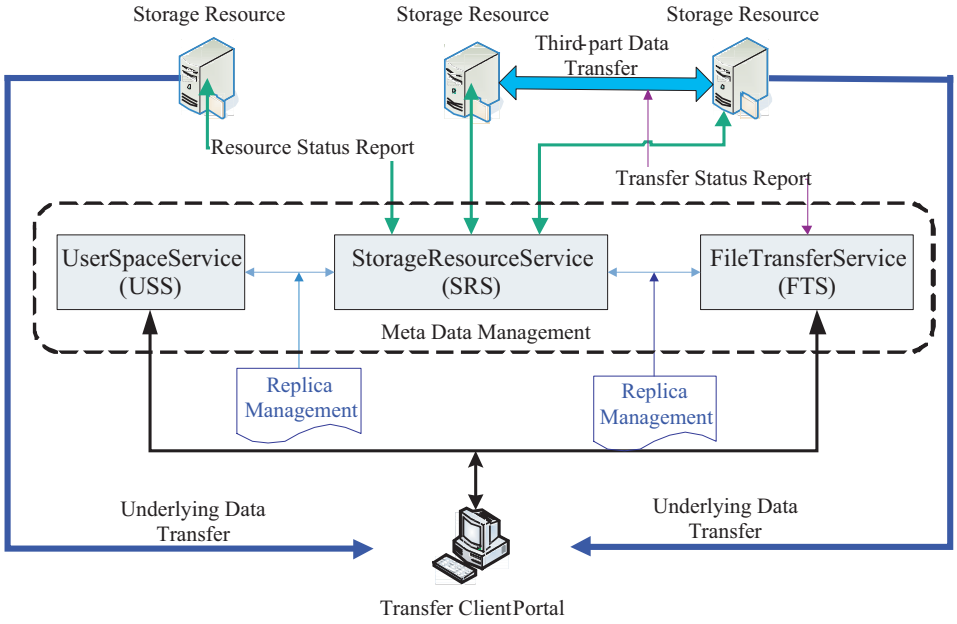
There are four main modules of security management: user identity setting, authentication in the container level, authentication of user token and authorisation management in the service level. User identity setting module maps user id to group id and adds these user identity information to the SOAP message. Authentication in container level mainly uses XML encryption technology to take encryption, decryption, signature and authentication in the SOAP message level. Authentication of the user token module contrast user's token and user information in the client side to authenticate user's identity. Authorisation management in the service-level module takes charge of taking users authorisation according to the authorisation rules in service security descriptor.

3.3.4 Data Management

Data Management (DM) aims to provide virtual storage space and reliable data transfer mechanism with high performance. The purpose of the storage virtualisation is to integrate the heterogeneous storage resources distributed in the grid environment. In the perspective of a grid user, DM provides a virtual file system. One can execute operations on the virtual file system as he/she can do on local file systems. The requirements of the data transfer in CGSP are efficiency, manageability and reliability. By managing each

process of data transfer as a WSRF resource, the creator of the resource is able to start, suspend, restart, stop the transfer process and get its up-to-date status. The architecture of the DM in CGSP is shown in Figure 2.

Figure 2 Architecture of data service in CGSP



DM consists of the WSRF service: UserSpaceService, which is responsible for user space management, StorageResourceService, which is responsible for management of the storage resources, and FileTransferService, which carries the physical data transfers between storage resources.

The GUI of the virtual storage pool is a grid file system, which is shown in the CGSP Portal. A legal user of CGSP Data Management will be allocated a grid file system. In the system, a user can perform several basic file system operations: make a directory, delete files/directories, search files/directories, upload files from users' local file systems to the grid file system, download files from the grid file system to users' local file systems and browse the grid file system.

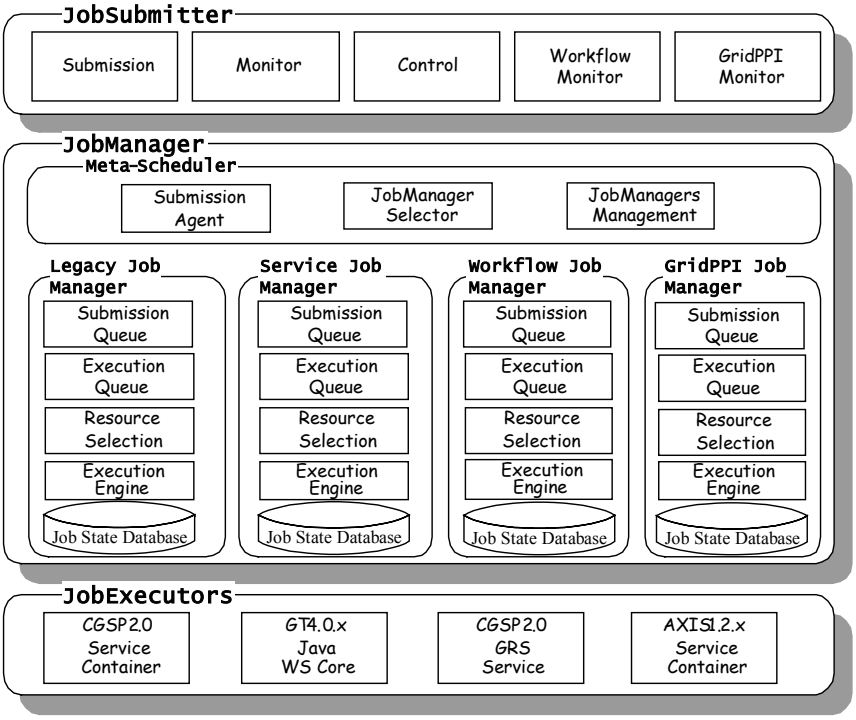
3.3.5 Execution management

Execution management is a very important function module in CGSP, which consists of many software components. Execution management, which is responsible for submitting, scheduling, managing and monitoring jobs launched by end-users enables applications to have coordinated access to underlying resources. It provides uniform interface of job creation and monitoring (such as WS, WSRF, JSDL, Composite Service, *etc.*), legacy binary program execution management, dynamic WSRF/WS service invocation, distributed workflow engine management and distributed workflow balancing.

Execution management mainly consists of three types of software module roles, which are supply side, broker side and demand side. On the supply side are services that manage and supply resources such as CPU, disk, data, memory and services. On the demand side are applications that consume the corresponding capabilities of resources. The broker side is responsible for building the binding between them. As Figure 3 illustrates, job executors, job manager and job submitter, respectively, serve these roles in CGSP.

- *Job submitter* – facilitates end users to submit jobs. Job submitter includes Java jars, GUI applications, command line applications and JSP pages. It can complete job submission, job control, job state monitor, workflow deployment and undeployment, and GridPPI job launch and monitor.
- *Job Manager (JM)* – receives job request submitted by users and is responsible for finding job execution location, initiating the execution, managing and monitoring the execution. JM's main task is to do job scheduling. A schedule is a mapping (relation) between services and resources. A schedule will typically attempt to optimise some objective function such as execution time, cost, reliability, *etc*. When resource selection generates the schedule, JM is responsible for enacting the schedule. According to different types of job, JM will use different types of concrete job managers to invoke job executors to complete the job.
- *Job executors* – are containers that carry out the execution of job. The containers could be GRS, CGSP service container, Java WS Core in GT4.0.x, and Axis1.2.x. GRS is for running legacy programs in Linux host. The CGSP service container supplies the WSRF-compliant service to encapsulate applications.

Figure 3 Architecture of execution manager in CGSP



3.3.6 Heterogeneous Database

HDB aims to enable users in the grid environment to access the services provided by various heterogeneous database more conveniently. HDB intends to provide users a uniform, flexible and standard interface to access and integrate databases. To make this standard interface, it has several unified accessing methods, which makes the data integration possible and brings heterogeneity transparency, naming transparency and distribution transparency. Hence, HDB is capable of being a supporting platform for accessing the very large and basic heterogeneous databases.

Execution Engine, one of the most important modules in HDB, is dedicated to parsing Perform Document, interpreting SQL statements, optimising conditions and schedule jobs in order to undertake most of workload. It is made up of SQL Interpreter and SQL Dispatch. When a user submits a request performed on a virtual table, SQL Interpreter receives and parses the SQL statement contained in the Perform Document, then partitions the statement into query plans whilst converting the operation referred to virtual table to that associated with real physical table; finally if conditions exist in the SQL statement, optimisation would be carried out. When the SQL Interpreter is completed, SQL Dispatch dispenses query plans to target data services, and controls the sequence of jobs execution, then constitutes retrieval data returned by distributed and disparate databases to form the final results.

3.3.7 Portal

CGSP Portal provides an easy way to use the service of ChinaGrid. Portal is a webpage not only for visiting current CGSP service but also for providing some support for new grid applications. As shown in the function list in Portal, there are six main functions: user and group management, grid services management, job management, data management, heterogeneous database management and system module management. From Portal, the users can manage ones' user information and group information, browse information centre, view ones' data space, upload and download files with Http or GridFtp, deploy ones' own applications to the grid which is managed by CGSP GRS service, submit jobs to applications and services, and can also view virtual tables in CGSP environment.

3.3.8 GridPPI

As its name implies, GridPPI is a set of APIs that enable end users, especially developers, to write grid-enabled parallel programs. Moreover, GridPPI is bundled with a behind-the-scenes runtime framework that saves developers all the effort in finding a way to run their programs in a parallel mode. Just follow several rules and instructions; a program that fully exploits the services provided by CGSP is right at hand.

3.4 Main features

3.4.1 High-quality, low-cost computing services platform

CGSP is a platform rather than a toolkit. It tries to cover from the top user interface to the bottom of integration of heterogeneous resource. It brings the user face to face with the grid developers but also the field expert, who are familiar with specific fields but not familiar with grid, and end users, who are ignorant of both specific field and grid but who need to use the computing power.

3.4.2 Grid workflow with WSRF support

Since activities of grid workflows are often related to stateful services (WSRF-compliant services), we enhance the support of WSA and add a new service invoking engine (for orchestrating WSRF service) in ActiveBPEL engine, which can select proper service invoking engine to complete the corresponding service invocation according to the designation of the service type of partner service in the Process Deployment Description file (PDD file).

3.4.3 JSDL job submission and legacy job execution using General Running Service (GRS)

By providing support to JSDL, CGSP enables user and program to submit jobs using a standard submission interface. CGSP uses GRS to perform real job execution. GRS is a WSRF service designed and implemented according to specific CGSP requirements and is able to take care of an instance of legacy job in its whole lifetime from application deployment to job completion.

3.4.4 MPI-like Grid Parallel Program Interface (GridPPI)

GridPPI is a grid-enabled parallel programming environment. It is also a framework designed for thread-level parallelism in Java. It provides a set of MPI-like Java APIs for communication and collaboration among threads. It is mainly designed for advanced Java/Grid application programmers to write fine-grained parallel programs.

3.4.5 Heterogeneous database integration

Based on the OGSA-DAI, heterogeneous database integration module is implemented for accessing multiple heterogeneous databases and enables users in the grid environment to use services provided by various databases through a uniform, flexible and standard interface and integrate contents of them.

3.5 Interoperability

Interoperability can combine the unique capability of heterogeneous grid platforms and enable grid systems to possess more resources in their own infrastructure. However, because all grid platforms have their own applications, it is a nightmare to migrate these applications to a new platform, and reprogramming the grid middlewares also takes much time. Hence, to achieve interoperability between heterogeneous grids, the following issues must be addressed: security, information service, job management and data management.

Based on the concept of mediated bridge mechanism, we adopt virtualisation and plug-in technologies to achieve interoperation among heterogeneous grids. In view of compatibility with other grids and enabling integration of more grid platform in the future, our proposed interoperability mechanism exploits current widely accepted standards such as Job Submission Description Language (JSDL) for job description and GLUE for resource description.

Our proposed interoperability mechanism adopts three layers of architecture, and security, information service, job management, and data management are considered. Heterogeneous grid platform layer refers to a set of distinct grid platforms built with diverse grid middleware, each of which usually is autonomous with its own management policy, its organisation architecture and its own implementation mechanism. Plug-in layer serves as an intermediary, which bridges the underlying grid platforms and upper virtual management layer, performing bidirectional information format translations, job parameters-type translation, data staging, and information integration and dissemination, *etc*. In the virtual management layer, Consistent Information Repository (CIR) and service modules constitute a virtual grid management centre, which hides the heterogeneity of underlying grid systems from users.

The main contributions of the proposed interoperability mechanism lie in the following respects: Firstly, it implements the interoperation among heterogeneous grid systems without any changes to them, keeping the independence of original grid systems. Secondly, it periodically translates and caches the information and data retrieved from grid platforms, helping to lower information querying latency, improve information querying accuracy and speed up service access response. Finally, the interoperability system that scales easily for the number of plug-ins only grows linearly in O(n).

3.6 Current status and applications

There are now five main grid applications with the support of CGSP: Image processing grid, Bioinformatics grid, Course online grid, Computational fluid dynamic grid and Large-scale information processing grid.

Image processing grid now can offer three types of image processing services: three-dimension reconstruction of digital virtual human being, medical image diagnosis and remote sensing image processing.

Bioinformatics grid develops computing grid-aided drug design systems based on the integration of the techniques of assembled biology, assembled chemistry, and computer-aided filter, sequence analysis and structure, to optimise drug design.

Course online grid has already provided about 300 courses and 3500-hour video from 12 universities for VOD and download. Users can get the courses VOD by IE using Real Player.

Computational fluid dynamics grid integrates computational mathematics, computer science, hydromechanics and computer visualisation.

Large-scale information processing grid includes three information-intensive grid applications: university digital museum, which aims at sharing the cultural relics effectively and to provide appropriate information service to users; high-energy physics computing and alpha magnetic spectrometer experiment.

3.7 Future work

In the near future, we intend to further study the dynamic maintenance approach of grid system at different levels (node, container and service), and implement a set of dynamic maintenance schemes in CGSP.

4 GOS

4.1 Design motivation

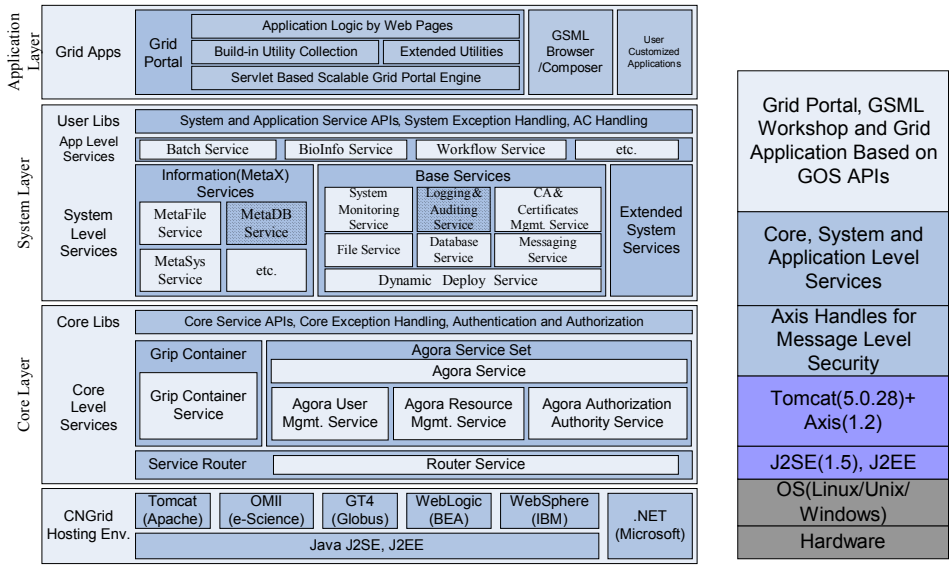
As internet technologies advance, there are more and more distributed resources to be shared and collaborated, which impose a great need for internet-based grid system software to manage resources. A system software is needed to manage large-scale distributed resource effectively, to provide a uniform approach accessing the heterogeneous resources in grid and support internet-based resource sharing and collaborating across administrative domains.

There is also a great need for easy-to-use grid. A grid system software is needed to hide interior details for developers and administrators, to help developers lower their cost for developing grid applications. The grid should be more convenient for end users by multiple access modes, such as client/server mode, browser/server and others. In high-performance computing field, batch mode and interactive mode are both needed to develop, debug and submit jobs.

4.2 Architecture

GOS architecture (GOS) is illustrated in the left side of Figure 4, which is mainly based on the GOS version 2 and version 2.1. GOS is totally four layers, including underlying hosting environment as the bottom layer.

Figure 4 GOS hierarchical architecture and runtime stack



The hosting environment mainly consists of the operating system, j2se, application server and SOAP engine. GOS runs on the hosting environment and has its interface of support portals to other hosting environments.

The core layer is something like the OS kernel, and 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 authorisation are included in this layer. The core layer is composed of grip service, agora service set and router service with wrapped client side API; user authentication and service authorisation mechanisms implemented by Axis handler chains; and the GOS exception handling extends from the Axis fault, which can help the developers in accurately locating the service side exceptions and failures.

The system layer provides a collection of basic libraries to help programmers develop grid applications quickly. Batch service accepts batch job submission, status query and cancel request; forwards batch jobs to backend batch system and gets back results. File service is based on local file system, organises plain files for user and provides HTTP-based file transfer. Dynamic deploy service dynamically deploys a service implementation (.jar file) into its hosted service container; updates and removes an existing service. To support message subscribe and notification, messaging service provides reliable messaging between peers by message queue and topic. Some services are undergoing development, such as general metainfo service, CA service, Database service listed in Figure 4.

The application layer is not constructed by services, but by API provided by system layer and core layer. The grid portal developer or integrator can benefit from the Grid Portal Engine by avoiding using the system or core layer API directly. Grid Service Markup Language (GSML) workshop, which is composed of GSML browser and GSML composer, is an agile integration toolset for grid applications development. There are user-customised applications based on GOS API existing in the top layer also.

The right side of Figure 4 shows the GOS runtime stack. The axis handles, core system and application level services are in the scope of the system software. Applications are built on services provided by GOS. At runtime, applications will invoke the core-, system- and application-level services to do authentication, services discovery and policy decision. After the corresponding policy enforcement is done by the axis handlers at the service side, the target service is invoked and the result is returned.

4.3 Function modules

4.3.1 Resource and user management service

In GOS, resources are jointly managed by the router service and agora service. The address (URI) of service is registered and virtualised by the router service. Each grid router service can manage the meta information of domain-scoped services. By incrementally propagating the difference between neighbour routers, each router keeps the router ids (the virtualised router service access point which is created automatically while the router starts for the first time) globally synchronised, that is to say, the decentralised interlinked grid routers can provide a unique global (SSIed) meta information space. The elements in this space are meta information of registered services, which are identified by a combination of router id and service id.

According to some classifying rules, some virtualised services maintained in different router services can be associated with certain users, therefore they generate a set of specified permission policies. The classified service, user information and policies are all

stored in one agora service. Agora service is the implementation of the VO. An agora service user is identified by a Distinguish Name (DN) in his/her certificate, and granted different permission by his/her role.

4.3.2 Batch job execution service

GOS provides a job management system to aggregate local cluster job management systems. It can support heterogeneous backend batch job systems such as OpenPBS, LSF and NQS. The job management system is encapsulated as batch service for better sharing. There is a metaschedule service to dispatch grid jobs to these batch services in CNGrid. It will monitor and collect the load information of each node and make the dispatch decision based on some default or user-defined custom policies.

The users can describe the job in standard JSDL language, submit it to the grid by portal or the API, monitor the job status, and wait for the result. The job management system will select a grid node, may be a cluster, to run the job. These candidate nodes must have the corresponding software to execute the job and the grid user must have local user accounts in that grid. The installed software and the user mapping information are provided by two services maintained by local cluster administrator. Then the job system will transfer the input files to an appropriate node, execute it and transfer the output files to the predefined position. The files are transferred by a grid file management system. After job submission, the accounting information of each node and each grid user will be gathered. GOS allows customised accounting policies.

4.3.3 Distributed file management system

Composed of metafile service and file storage service, a distributed file management system integrates local file systems and forms a logical global file system. Except for remote file reading/writing function, it can provide remote file management interfaces which include file upload/download, file/directory creation/deletion, file authorisation and so on. The metafile service records the mapping between logical file name and physical file name to maintain an independent space for each grid user. Physical file is available at the file storage service side, and physical file name is the absolute location of this file (/path/to/file or driver:\path\to\file). Virtual file name is the id of the virtualised file storage service plus the file owner's DN and relative path of the file in local file system. The virtual file name can uniquely determine the location of the file and owner in the grid. The file storage service handles the file transfer requests and organises the local file system. Files are transferred by HTTP protocol to avoid firewall problems.

4.3.4 Grid system monitoring service

GOS provides a grid monitoring system based on SOA to monitor the CPU, memory or disk usage of PC, cluster or supercomputer. The monitor system follows a hierarchy architecture, including local monitor server – LMS and global monitor server – GMS. LMS is responsible for collecting and storing local operating system-level status data, such as loadavg, mem and CPU utilisation, *etc.*; while GMS is for gathering necessary data from multiple LMS and supporting grid scope information retrieval. The LMS can live alone and provide information of one grid node before being registered to a GMS. Both LMS and GMS are equipped with web service front-end interface.

4.3.5 Security system

Especially, the security in a grid environment seeks to:

- provide authentication solutions that allow user and the resources accessed by that user to verify each other's identity
- provide agora-based authorisation mechanism
- allow local access control mechanisms at service side to be integrated into grid security without changing the original source code.

Inside the security system, we have developed a CA service that is responsible for certificate management, and have implemented WS security-compliant authentication, authorisation, message-level secure communication, access control by handler chains of axis.

The extensible message processing model is the key to the security system. This model uses handlers and handler chains of axis that can enable the functionality to be tailored to satisfy a wide variety of situations and requirements. A handler is an atomic component that will operate on a specified part of a SOAP message. For example, a handler can be in charge of performing authentication on a 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.

4.4 Main features

GOS is featured by its design principles and key technologies. Some principles are formed to guide its designs, including satisfying minimal common requirements, using a computer system approach and adopting matured standards and technologies.

4.4.1 GOS components and resources are all encapsulated by service

The components of GOS and resources are encapsulated by service to provide uniformed interface to a variety of resources and hide underlying resource heterogeneity. Services are easy to integrate and expand owing to their loosely coupled features. These services are WS-related and standards-compliant to allow better interoperability.

4.4.2 Layered resource spaces for virtualisation

GOS introduced a layered resource space called Effective-Virtual-Physical Service Address Space to tackle the naming issue of grid resources. It is constructed by physical, virtual and effective layer, and every upper layer is built on a lower layer. The physical address layer is composed of the URL of the services. When a service is registered, it will generate a unique global identifier, which is the virtual address. Multiple virtual addresses having the same functions can be added to the agora and form a user-friendly effective address, which is used by developers. Service address naming schemes in GOS are as follows:

Physical: `http://host_name_or_ip:port_number/suffix`

Virtual: `vres://router_id:service_id`

Effective: `eres://agora_name:service_name`

The address space can separate resources and applications so that the application can keep still while resources are changing. Furthermore, it can provide functions such as resource selection, fault tolerance, authorisation and access control, and all these functions can be made transparent to users.

4.4.3 Utilising several abstractions to simplify accessing grid

GOS proposed some key abstractions including agora and grip. 'Agora' is a type of concrete implementation of 'VO'. It is defined as a set of users, resources and policies. It is used to aggregate and organise users and resources locally, establishing corresponding relationship across them, such as roles, service category, service selection and authorisation policy. Agora provides a standard way of sharing resources across real administrative domains in grid. The administrator of a grid can just organise needed users and resources in the same agora and define the appropriate access control polices. If the resources trust the agora, then the users can share these resources in the same agora.

'Grip' is something like 'Grid process'. It is used to maintain the conversation between users and services at runtime. It holds necessary information about users and services, such as user proxy, resource addresses in use during the grip's lifetime and so on. It provides a set of uniformed interfaces for accessing different services. Most importantly, grip is used to manage the resources during an application's running. It can log the resource usage for audit or billing. It can revoke all the resources used when the grip is killed.

4.4.4 Grid security mechanism that supports web services

GOS provides PKI-based authentication, which is WS security-compliant. GOS separates authorisation decision and enforcement using SAML token. The authorisation policies are defined in agora. When an application tries to bind a resource, the grip will ask the agora to get corresponding policies about the resources. These policies will be formed as a SAML token, signed by the agora. That is the process of decision. When invoking services, the token will be passed on to the service side, then the service enforces the access control polices according to both the SAML token and its local policies.

GOS supports flexible security features such as multiple access control enforcement and SOAP message integrity verification by axis handler-chain mechanism. These security features can be combined and configured during service deployment and are independent with the service implementation.

4.4.5 Rich user environment

The most powerful interface is the API generated from the core, system-level services. It can be used by developers to build their own grid applications, including some applications already in the GOS itself. Based on these APIs, GOS provides a portal to manage the grid. Grid administrators can use the portal to manage users, resources, agoras, define policies and manage running grip, killing it for example.

The GSML Workshop, which includes composer and browser, is an agile integration toolset for developing and running grid applications. It features reusable components, event-driven programming model and WYSIWYG programming environment. The GSML Workshop enables flexible application logic and collaboration mechanisms by reusable components with an internal event-based model. These components provide uniformed event interfaces to hide low-level technical details of resources. When these components are together composed by connecting their event interfaces, loosely coupled applications are produced that are represented by individual GSML documents. The visual IDE helps the developers to edit, debug and run the applications intuitively.

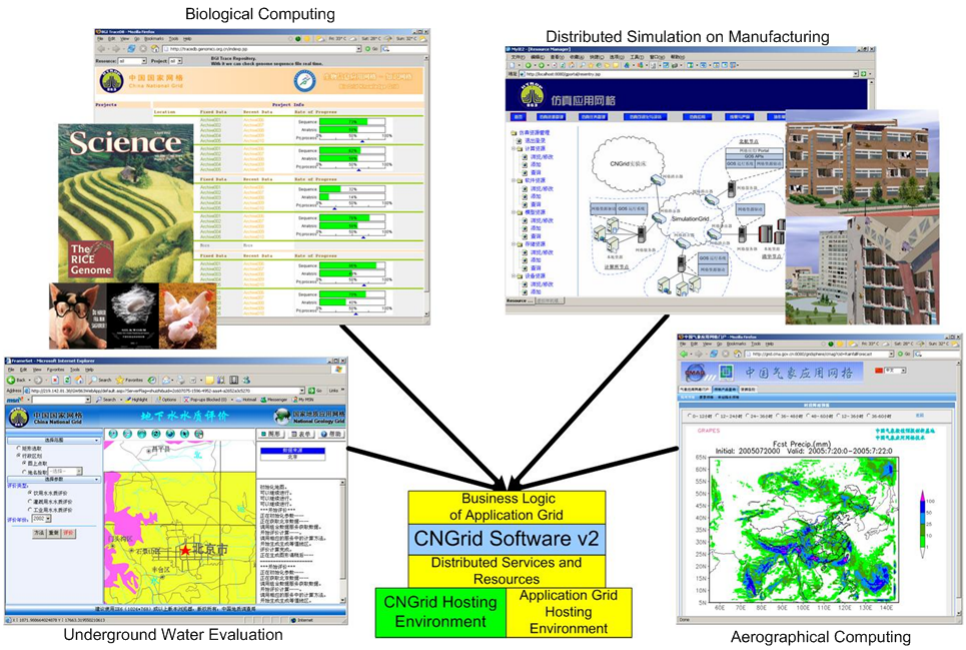
Also, a grid SSH tool is built in the GSML workshop, which can help grid users transparently log into multiple connected grid nodes. Once a grid users logs into one node, he/she can freely login to other nodes in the grid and execute local command or copy files between nodes.

4.5 Current status and applications

GOS version 2.1 has been released and deployed on the CNGrid platform in November of 2006. In this version, the core abstraction has been refined and the performance and robustness of the software have been improved. The software is easier to use. Also, the stability of the batch system has been improved.

The application scope contains science research, manufacturing and resources and environment. Some example applications are shown in Figure 5.

Figure 5 Application scope of GOS



Biological research is mainly focused on genome computing, genome sequence tracing and computing. The manufacturing application in Figure 5 is aviation and space simulation computing. There are some geological researches, such as underground water evaluation. Figure 5 also shows some daily running weather forecast application.

4.6 Future work

The future work of GOS will include the following aspects. The first one is GOS key abstraction and core-level service refinement, including 'address space', 'grip' and 'agora'. The second is system-level service and functionality expanding, especially database service, CA service, metainfo service, grid data management and workflow. The third is application scope enlarging, from scientific computing area to general service computing area.

5 CROWN

5.1 Design motivation

Since many researchers are focusing on technologies and applications of grid, the interoperability and loosely coupled integration problem between different grid systems are now becoming a hot topic. At the same time, the application and standardisation of web services technology are developed rapidly, and Service-Oriented Architecture (SOA) becomes an important trend in building a distributed computing environment for wide area network, which helps the merging of grid and web services. Recently, Open Grid Service Architecture (Foster, 2002) and Web Service Resource Framework[22] were proposed and have become two of the fundamental technologies in grid computing. SOA and related standardisation work provide an important methodology to the research and application of grid technology. First, the resources are encapsulated into services with standardised interfaces, supporting the unified service management protocol, which helps to solve the problem caused by the heterogeneity of resources. Second, the resources are utilised through a service discovery and dynamic binding procedure, which helps to set up a loosely coupled computing environment. However, the current resource management mechanism is not enough for all the grid application scenarios because the distributed and autonomic resource environment, and the existing security mechanism cannot provide features such as privacy protection and dynamic trust relationship establishment, which limit the further application of grid technology.

Actually, not only grid computing, but also peer-to-peer computing and ubiquitous computing try to explore the internet-oriented distributed computing paradigm. The common issue in these computing paradigms is how to use the capability of resources efficiently in a trustworthy and coordinated way in an open and dynamic network environment. As we know, the internet (especially the wireless mobile network) is growing rapidly, while it lacks effective and secure mechanisms to manage resources, especially when the resource environment and relationship between different autonomic systems are changing constantly. At this point, three basic problems, namely, cooperability, manageability and trustworthiness are proposed. The cooperability problem is how to make the resources in different domains work in a coordinated way to solve one big user's problem. The manageability problem is how to manage

heterogeneous resources and integrate the resources on demand in a huge network environment, which is a basic condition for building an internet-oriented distributed computing environment. The trustworthiness problem is how to set up a reliable trust relationship between cross-domain resources when they are sharing and collaborating.

From 2002, based on our previous work on web service supporting environment (Shu *et al.*, 2004), and OGSA/OGSI-compatible service grid middleware WebSASE4G (Hu *et al.*, 2004), a WSRF-compatible CROWN grid middleware (Huai *et al.*, 2006; Hu *et al.*, 2005; Sun *et al.*, 2005) is proposed. On the basis of the three basic problems, several key issues have been explored, such as resource management of service grid, distributed access control, cross-domain trust management, grid service workflow and service orchestration-based software development.

5.2 Architecture

Grid computing started from metacomputing in the 1990s. In recent years, the grid has changed from metacomputing to computing grid and service grid, but the basic architecture of such a computing paradigm has not changed much. In 2001, the five-layer sandglass architecture was proposed and accepted generally. With the application and standardisation of the grid, the architecture and its supporting technology has become an important research issue. OGSA is a service-oriented architecture, which adopts the service as the unified resource encapsulation format to provide better extensibility and interoperability between grid resources. WSRF refines the service interface and interoperating protocols of OGSA, and makes the OGSA a web service-compatible implementation framework, which helps the merging of grid and web service technology more smoothly.

Actually, the five-layer sandglass architecture just proposed an abstract functionality structure for the service grid. OGSA/WSRF provided an implementation framework based on service concept, and a set of grid service interfaces, neither of which discussed the design principles, middleware component definitions, and the detailed solutions for access control and trust management in the service grid. In this paper, we analyse the requirements from typical grid applications, and provide a detailed introduction of CROWN middleware and its architecture, design principles and kernel technologies based on the OGSA/WSRF service grid architecture.

Generally, there are three kinds of services in an OGSA/WSRF service grid: general services, resource encapsulating services and application-specific services. General services, such as grid job broker service, metascheduling service, and grid information services (GISs) are an important part of a service grid middleware. In a typical application grid scenario (see Figure 6), a user first submits a job to the metascheduling service, then gets the job status and result from the job broker service; metascheduling service retrieves the resource requirements from the job description language, queries GIS to discover the necessary service, submits the job to the service and traces the status of the job execution.

On the basis of the above analysis, we provide a layered architecture for CROWN and compare it with the five-layer sandglass architecture (see Figure 7). In Figure 7, service grid middleware covers resource layer, collective layer and application layer. In our system design, services in resource layer (*e.g.*, resource encapsulation service), collective layer (*e.g.*, grid information service, metascheduling service and job broker service) and part of services in application layer are OGSA-compatible grid services,

using information service to obtain the capability of registration and dynamic discovery. Grid application support tools are used to enable the interoperation between the collective layer and application layer (for example, the application-specific developing framework, routines, and web-based grid application portals, *etc.*).

Figure 6 CROWN-based service grid application pattern

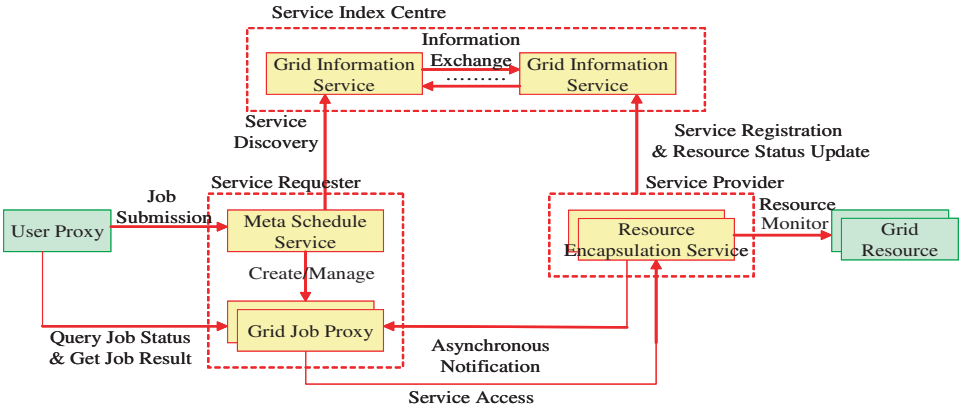
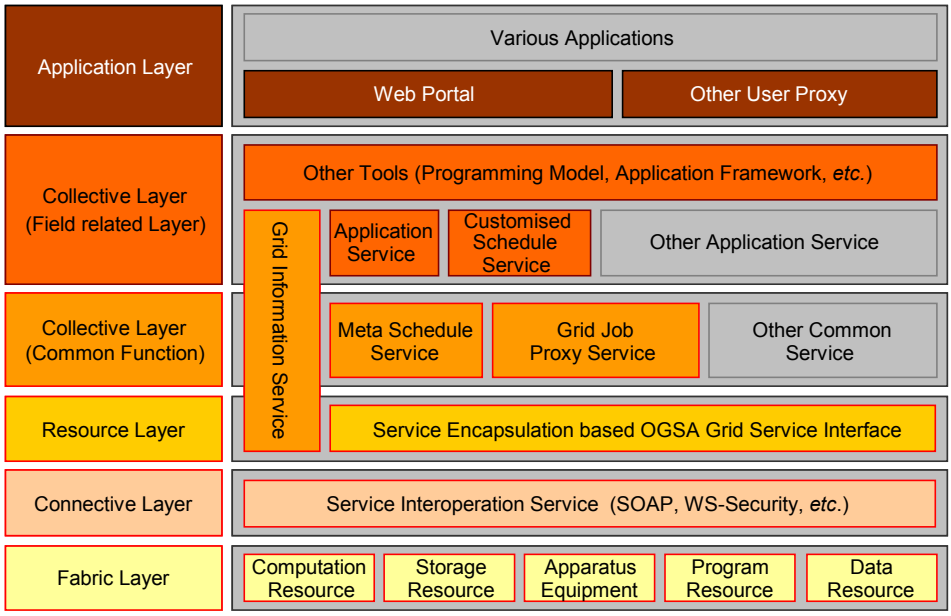


Figure 7 CROWN-based service grid application



5.3 Function modules

There are 11 components in total in CROWN service grid middleware analysed as follows:

1 Node server

It provides a basic runtime environment for grid service. Using node server, the underlying resources can be encapsulated into grid services. Node server provides all the generic functionalities when running a service instance, such as SOAP message processing, service instance management; instance/invoke life cycle management and notification mechanism. Based on Globus Toolkit 4.0, the node server adds lots of features such as remote and hot service deploying, resource status monitoring and reporting, logging and remote control and management. By adding security modules, the node server can provide features such as PKI/Kerberos-based authentication, fine-grained authorisation, trust management and Automatic Trust Negotiation (ATN), which could guarantee the security and privacy effectively when resources are used by remote users or cross-domain resources.

2 Resource locating and description service

It is a distributed information service system for service registration and discovery. Multiple Resource Locating and Description Service (RLDS) instances use information exchange and topology maintenance protocol to build the hierarchical architecture and the overlay network at runtime as needed to get better performance of resource management and service discovery.

3 CROWN Scheduler

It is a metascheduling service in CROWN that queues and schedules user's jobs according to a set of predefined strategies, interoperates with RLDS to get current service deployment information and job status, uses predefined scheduling policy (random policy, load balancing policy, *etc.*) to do the matchmaking, and performs the service invocation. CROWN scheduler supports two types of job, POSIX application invocation and grid service invocation. Job Submission Description Language (JSDL) is used to describe the QoS requirements and security demands of the jobs.

4 CROWN CommSec

It is a plug-in for the node server and a generic web service to provide the basic security communication feature such as building and verifying of certificate chains. Administrators can edit the predefined policy file according to complex security requirements to provide independent, extensible and feasible security solutions.

5 CROWN Authz Service

It is a generic service using an XACML-based authorisation policy description language and provides the capability of authorisation decision and policy management. It supports the multigranularity access control policy and domain access control policy.

6 CROWN CredMan

It is used to manage user credentials. Through the agent certificate issue, the identified subjects can be managed especially when the job is submitted from the web portal or in the mobile networks.

7 CROWN CredFed

It contains a plug-in for the node server and a credential mapping service. It can be used to map credentials from different security infrastructures (such as PKI and Kerberos) to enable the identity mapping between two security domains. Administrators can modify the mapping policy to control the behaviour of CredFed.

8 CROWN ATN

It contains a plug-in for the node server and a set of generic services. The ATN establishes the trust relationship between strangers on the internet, protecting the privacy (for example, the information on attributes-based certificates and the negotiation policies) of both sides. It provides a security decision and trust management mechanism for open network environment.

9 CROWN Portal and Rich Client Framework

The two tools provide a unified user interface to service grid to support the job parameter configuration, job submitting, JSDL generating and result demonstration. CROWN Portal provides a web-based interaction model for the applications, and Rich Client Framework provides a Java-based application framework for applications, which has extensive visualisation and interaction demands (such as complex visualisation). The framework can be customised according to the application scenario to speed up the application development.

10 CROWN Designer

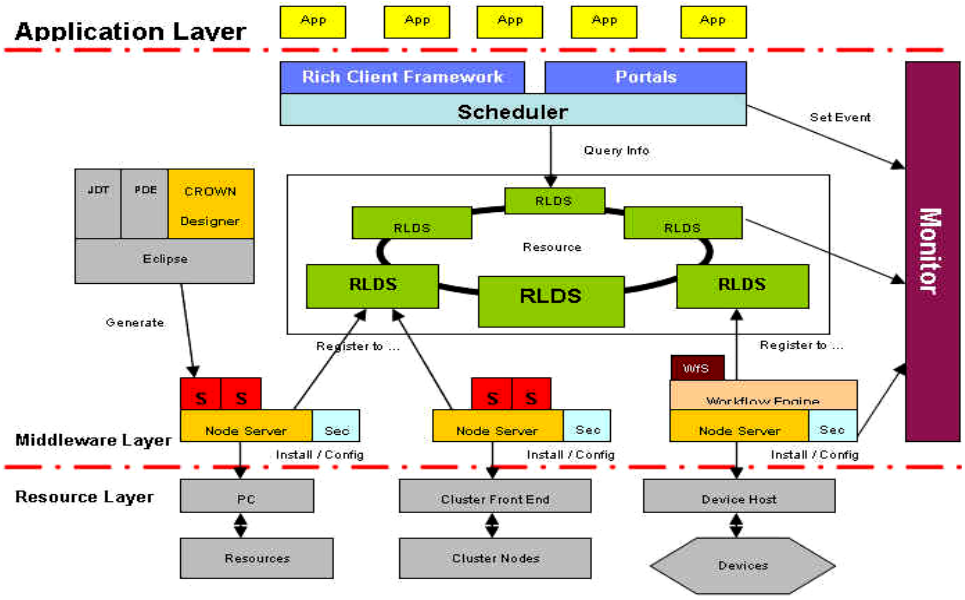
It is a grid service developing and deploying tool based on the Eclipse platform. A set of wizard and dialogues provided make the development and deployment of grid service much easier. By using the remote and hot deploy feature of the node server, the designer provides drag-and-drop features to deploy the GAR file. In the near future, more service orchestration tools will be integrated into the CROWN Designer.

11 CROWN Monitor

It is an Eclipse RCP-based client tools written in Java. It is used to retrieve, store and analyse events/information from different grid entities, and to show current runtime information using map and chart. We can also adjust parameters of the tool to change the monitor behaviour to the target service grid systems.

Based on these middleware modules, CROWN is designed as shown in Figure 8. There are three layers in a service grid. CROWN middleware connects resources in a resource layer. The application grid uses a web portal or other customised client interfaces to submit jobs and solve the user's problem. First, the node server should be deployed on each resource to support service deploy and runtime management. Second, all the grid resources are divided into multiple domains; at least one RLDS instance should be deployed into each domain, and all the RLDS instances have to be configured into a predefined architecture to form a distributed information systems. Third, CROWN scheduler will be deployed into the grid, to get the job request from the user and to find the proper services for each job; finally monitoring and developing tools simplify the building procedure of a service grid and its applications.

Figure 8 Service grid design principle based on CROWN middleware



5.4 *Main features*

CROWN adopts an OGSA/WSRF-compatible architecture, with the following features.

5.4.1 *Overlay-based distributed resource management*

Overlay technique is an effective way to support new applications as well as protocols without any changes in the underlying network layer. The basic idea of the overlay network is to build a new network over the existing physical network nodes according to some selected logical rules. In CROWN, resources are managed by an information service overlay consisting of a set of RLDS services. These RLDS instances are linked with each other according to a tree-based topology with carefully selected short cuts, exchanging resource and request information with their logical neighbours. Such a fully decentralised structure can provide better performance with the avoidance of single point of failure at information systems.

5.4.2 *Remote and Hot Deploy with Trust (ROST)*

Traditionally, the remote service deployment is supported in a cold fashion, which means deploying a new service, and the service runtime environment needs to be restarted. Therefore, the hot service deployment has become increasingly important, which does not need to restart the runtime environment while deploying services. To achieve this feature, an archive format called GAR file (Grid Archive) is proposed to encapsulate all the necessary files and configurations for a grid service. The GAR file can be moved to the target service container through SOAP/HTTP protocols. The target service container receives the GAR file and uncompresses it to update the container information without stopping the container.

Security issues are guaranteed through the trust negotiation using the ATN technique. ATN is a new approach to access control in an open environment, which, in particular, successfully protects sensitive information while negotiating a trust relationship. With ATN, any individual can be fully autonomous. Two individuals, which are not in different security domains, try to set up a trust relationship by exchanging credentials according to respective policies.

With the availability of remote and hot service deployment, many applications will benefit, such as load balancing, job migration and so on.

5.4.3 JSDL-based job submission and BES-based job scheduling

Job Submission Description Language (JSDL) and Basic Execution Service (BES) are adopted in the CROWN scheduler, with extension to web service-based job submission. Jobs can be submitted to the CROWN scheduler via any JSDL-compatible clients, such as GridSAM, and gLite using a BES interface. Interoperability demonstrations are proposed in AHM 2006 and SC 2006, organised by High Performance Computing Profile (HPCP) working group in OGF.

5.4.4 Security architecture supporting domain interoperability

CROWN uses a federate construction to form the virtual organisation. We use the term 'region' to denote the area with homogenous security infrastructure such as PKI or Kerberos, and the term 'domain' to denote the area of autonomous organisation. When grid services are deployed in different domains, each domain may have their own security concerns about the services. CROWN provides a fine-grained and extensible architecture that maximises the separation of service administrators and service developers.

Besides this, CROWN allows that the same implemented service be deployed into those PKI domains as well as Kerberos domains without having to modify the source code of the service. Furthermore, CROWN-ST also supports users from domains with heterogeneous security infrastructures to access the resources from other domains.

5.5 Interoperability

The interoperability of CROWN middleware can be described in several aspects. From the architecture layer, CROWN follows the service-oriented architecture with OGSA/WSRF-compatible design principles. Grid services developed for CROWN can be reused in other WSRF-compatible service containers. From the message layer, CROWN middleware adopts many widely used standards and specifications, such as SOAP, WSDL, and WS-Security specifications, which helps CROWN to interoperate with other web service-based grid system (such as OMII and GoS) at the message layer. From the implementation layer, CROWN Node Server is an enhanced Globus Toolkit 4.0 core, which brings better interoperability with other GT-based systems.

5.6 Current status and applications

CROWN is now becoming one of the important e-science infrastructures in China.

We have developed and deployed a series of applications from different disciplines, which include Advanced Regional Eta-coordinate numerical prediction Model (AREM), Massive Multimedia Data Processing Platform (MDP), gViz for visualising the temperature field of blood flow, Scientific Data Grid (SDG) and Digital Sky Survey Retrieval (DSSR) for virtual observatory. These applications are used as test cases to verify the technologies in CROWN.

AREM uses the grid as a tool to study and refine the numerical prediction model of weather and climate. Several numerical models are worked out by meteorologists during their research and prediction work. Typically these models use the raw weather data from national meteorology authority as input, and simulate the weather transformation based on the laws of atmospheric physics and fluid dynamics. The output can be used as a prediction result of future weather. The simulations are all based on complex numerical calculations and need large quantities of computing power and storage capacities. By using the resource organisation, job scheduling technologies provided by CROWN, we successfully developed the AREM research system. We encapsulated the Fortran compiler, visualisation tools (GrADS) and the simulation framework of AREM as services; a unified raw weather data centre is also deployed. Meteorologists can submit simulation jobs to the system and refine the numerical models according to the results. Since the jobs are executed by using the resources provided by the CROWN test bed, the execution procedure can be parallel, the execution time can be much reduced and the efficiency of weather system research and prediction model refinement is improved.

Large amounts of storage capability and computing power is needed when performing multimedia data processing, such as content recognition of voice or video. Traditionally a central processing model is applied and pieces of data are collected and processed in a single point. When the input data increases, this method provides little scalability especially for the real-time applications. We combine the service grid technologies with the massive data processing and implement the MDP platform for multimedia data processing. MDP has been deployed into CROWN and provides service since 2005. We encapsulate the related algorithms into services and deploy then on many grid nodes. Users can provide many ways of multimedia data and submit jobs to the grid scheduling system. After analysing the workload of grid nodes, available resources can be found automatically and data can be processed by invoking corresponding services. Since the platform is deployed in a wide-area environment, we also introduced the trust management and negotiation mechanisms. These technologies protect the user data and make the processing trustworthy. By using MDP, the resources that can be used to process multimedia data increase apparently, and the throughput and dependability of processing can be much improved.

CROWN interoperates with other grid middleware through specifications. The test bed also links to some famous grid test beds. For example, gViz application is deployed both in CROWN and White Rose Grid (WRG), which is a part of the UK National Grid Service (NGS). We demonstrated the application on UK e-Science All Hands Meeting 2005 to show the interoperability of heterogeneous and autonomic grid systems.

The experience of system development and deployment mentioned previously shows that CROWN provides the capability of resource management, distributed access control and trust management and negotiation. It can be used to support applications that are computation-intensive and/or data-intensive. Eleven applications had been deployed into CROWN by April 2006 and more than 25 000 requests has been processed.

5.7 Future work

We are currently focusing on the fundamental problems of resource collaboration, system management and trust to establish a trustworthy virtual service computing environment. A further step of research includes the overlay-based and dynamic resource organisation and allocation model, management of QoS and protocol computing-based interdomain trust management and access control mechanisms. We are also developing a fast application development method and tool that is based on service composition technologies. More applications from more disciplines will also be deployed into CROWN.

6 Comparison

From details discussed previously, we can compare these three projects of main middleware in China as shown in Table 1.

Table 1 Comparison among three middleware in China

Item	*CGSP*	*GOS*	*CROWN*
Intent	Enable 100 Chinese universities to collaborate on scientific research, and high education	Construct a distributed high-performance environment to support various applications	Build a virtual environment for research and scientific experiments
Dates	1st phase: 2003–2006	1st phase: 2002–2006	1st phase: 2003–2007
	2nd phase: 2007–	2nd phase: 2006–	
Developers	42 developers from 20 ChinaGrid member universities	More than 20 universities, institutions and enterprises in mainland China and Hong Kong	More than 12 universities and institutions in mainland China
Container	GT WS Java Core	Tomcat 5.0.28+Axis 1.2	Extension of GT 4 WS Java core
	Hot deploy		
	Remote deployment		
Information service	ChinaGrid topology maintainer	Grid Router with distributed meta information management	RLDS network with topology management
	Service and resource information manager		SQL-like resource query language (RIQL)
Data manager	Storage broker	Grid File Management System which can individual file systems	Meta Data Service and Local Data Service, with extended P-FTP implementation, support the file sharing and transfer
	File transfer		
	Heterogeneous databases integrator based on OGSA-DAI		

Table 1 Comparison among three middleware in China (continued)

	CGSP	*GOS*	*CROWN*
Execution Manager	JSDL Supporting Legacy and MPI program execution WSRF and WS invoking Job define tools Job manager	Batch Job Service and batch system drivers that can accept JSDL jobs and communicated with backend batch system	CROWN Scheduler and node server extension, enable the execution of legacy code and web services
Workflow	BPEL for WSRF Distributed Workflow Engine	N/A	BPEL-based service composition and distributed execution
Job Types	JSDL Job WSRF Service GRS Application (Legacy Programs)	JSDL Job Web service implementation (.jar) for dynamic deployment	POSIX Job (legacy code and patch program), pure WS, WSRF services
Programming Interface	GridPPI	GSML and GOS APIs	Java API to interact with CROWN modules
Security Support	Identity mapping Access control	WS-Security+SAML	Identity mapping, distributed access control and trust negotiation at runtime
Interoperability	GPE ProActive GT4	GT4; OMII	Support BES/JSDL spec, Interop demo with Globus, gLite, Unicore, OMII in SC'06 and OGF 20

7 Conclusion

In recent years, grid computing has become a rapidly growing research area. With the pace of the whole world, we need grids for greater computing capacity, resource sharing and collaborative work. China views the grid as an important opportunity for many fields and contributes huge effort to the development of grid middleware as well as grid computing technology. The rapid development and growth of grid projects inspire more concern on grid middleware.

In this paper, we mainly focus on the introduction of Chinese main grid middleware. Three representatives, CGSP, GOS and CROWN, are discussed in detail from design motivation, system architecture, main functions, key features, interoperation with others and related applications.

Grid technologies have been attracting more and more attention from various distributed applications. China views the grid as the national information infrastructure. Many national plans and industrial projects have been launched for building grid middlewares to push practical applications.

Acknowledgements

This work is supported by the ChinaGrid project of the Ministry of Education of China, Natural Science Foundation of China (90412006, 90412011, 60573110, 90612016, 60673152, 60673174), National Key Basic Research Project of China (2004CB318000, 2003CB317007), and National High Technology Development Program of China (2006AA01A108, 2006AA01A111, 2006AA01A101, 2006AA01A106, 2006AA01A115).

References

Foster, I. (2002) 'The physiology of the grid – an open grid service architecture for distributed systems integration', *Open Grid Service Infrastructure WG, Global Grid Forum.*

Foster, I. and Kesselman, C. (1999) 'The grid: blueprint for a new computing infrastructure', San Francisco, CA: Morgan Kaufmann Publishers, Inc.

Foster, I., Kesselman, C. and Tuecke, S. (2001) 'The anatomy of the grid: enable scaleable virtual organizations', *Journal of High Performance Computing*.

Ghalem, B. and Yahya, S. (2007) 'A hybrid approach to replica management in data grids', *International Journal of Web and Grid Services (IJWGS)*, Vol. 3, No. 1.

Hu, C., Zhu, Y., Huai, J., Liu, Y. and Ni, L.M. (2005) 'Efficient information service management using service club in CROWN grid', *Proceedings of the 2005 IEEE International Conference on Service Computing (SCC 2005)*, in conjunction with the *2005 IEEE International Conference on Web Services (ICWS 2005)*, pp.5–12.

Hu, C.M., Huai, J.P. and Sun, H.L. (2004) 'Web service-based grid architecture and its supporting environment' [in Chinese], *J of Software*, Vol. 15, No. 7, pp.1064–1073.

Huai, J., Hu, C., Sun, H., Li, J., *et al*. (2006) 'CROWN: a service grid middleware with trust management mechanism', *Science in China F: Information Science*, English ed., Vol. 49, No. 6, pp.731–758.

Huang, C., Wu, Z., *et al*. (2003) 'Dart: a framework for database resource access and discovery', *Proceedings of the Second International Workshop on Grid and Cooperative Computing, Lecture Notes in Comput. Sci. 3032/3033*, Shanghai, December,.

Jin, H. (2004) 'ChinaGrid: making grid computing a reality', *International Collaboration and Cross-Fertilization*.

Shu, J., Hu, C.M., Ge, S., *et al*. (2004) 'Research and implementation of web service runtime platform' [in Chinese], *J Computer Research Development*, Vol. 41, No. 3, pp.442–450.

Sun, H., Hu, C., Zhu, Y., Huai, J., Liu, Y., Li, Y. and Li, X. (2005) 'Early experience of remote & hot service deployment with trustworthiness in CROWN grid', *Proceedings of 6th ACM International Workshop on Advanced Parallel Processing Technologies (APPT'05)*, Hong Kong, October, pp.301–312.

Xiao, N., Li, D., Fu, W., Huang, B. and Lu, X. (2003) 'GridDaen: a data grid engine', *Proceedings of the Second International Workshop on Grid and Cooperative Computing, Lecture Notes in Comput. Sci. 3032/3033*, Shanghai, December.

Wu, Y., Wu, S., Yu, H. and Hu, C. (2005) 'Introduction to ChinaGrid support platform', *Lecture Notes in Computer Science*, (ISPA2005), Vol. 3759, pp.232–240.

Wu, Z., *et al*. (2003) 'Knowledge base grid: a generic grid architecture for semantic web', *Journal of Computer Science and Technology*, July.

Xu, Z., Li, W., *et al*. (2004) 'Vega: a computer systems approach to grid computing', *Journal of Grid Computing*, Vol. 2, No. 2, pp.109–120.

Zha, L., Li, W., *et al*. (2005) 'System software for China national grid', *IFIP International Conference on Network and Parallel Computing (NPC 2005)*, LNCS 3779, Beijing, China, pp.14–21.

Bibliography

Graham, F.S., Kesselman, C., Maguire, T., Sandholm, T., Snelling, D. and Vanderbilt, P. (2004) 'Open grid services infrastructure', *OGSI V.1 Specification: GFD.15*.

Li, M., Liu, H., Jiang, C., *et al*. (2003) 'ShanghaiGrid in action: the first stage projects towards Digital City and City Grid', *Proceedings of the Second International Workshop on Grid and Cooperative Computing*, Lecture Notes in Comput. Sci. 3032/3033, Shanghai, December.

OGSA, http://www.globus.org/ogsa/.

Tang, F., Li, M. and Cao, J. (2003) 'A transaction coordination mechanism and its compensation technology in grid environment' [in Chinese], *Journal of Computer Research and Development*, December.

Tang, F., Li, M. and Cao, J. (2003) 'A transaction model for grid computing', *Proceedings of the Fifth International Workshop on Advanced Parallel Processing Technologies, Lecture Notes in Comput. Sci. 2834*, September.

Tang, F., Li, M., *et al*. (2003) 'Coordinating business transactions for grid service', *Proceedings of the Second International Workshop on Grid and Cooperative Computing, Lecture Notes in Comput. Sci. 3032/3033*, Shanghai, December.

Wu, Y., Wu, S., Yu, H. and Hu, C. (2005) 'CGSP: an extensible and reconfigurable grid framework', *Lecture Notes in Computer Science*, (APPT2005), Vol. 3756, pp.292–300.

Notes

1 Globus, http://www.globus.org.

2 Teragrid, http://www.teragrid.org.

3 GIG, http://www.nsa.gov/ia/industry/gig.cfm.

4 OSG, http://www.opensciencegrid.org.

5 E-Science, http://www.rcuk.ac.uk/escience/.

6 OMII, http://www.omii.ac.uk.

7 GridComp, http://gridcomp.ercim.org/.

8 EuroGrid, http://www.eurogrid.org.

9 EGEE, http://www.eu-egee.org/.

10 NAREGI, http://www.naregi.org.

11 Unicore, http://www.unicore.eu/.

12 GPE, http://gpe4gtk.sourceforge.net/.

13 Websphere, http://www.ibm.com/software/websphere.

14 GridEngine, http://gridengine.sunsource.net.

15 Platform, http://www.platform.com.cn/Products/Platform.Symphony/Home.html.

16 WebLogic, http://www.bea.com.cn/products/beawebLogic/index.jsp.

17 CNGrid, http://www.cngrid.org/.

18 ChinaGrid, http://www.chinagrid.edu.cn.

19 CGSP, http://www.chinagrid.edu.cn/cgsp/.

20 GOS, http://www.cngrid.org/en_resource.htm.

21 CROWN, http://www.crown.org.cn.

22 WSRF, http://www.globus.org/wsrf/.