Sept. 2006, Vol.21, No.5, pp.– *J. Comput. Sci. & Technol.*

Usability Issues of Grid System Software

Zhi-Wei Xu[1] (徐志伟), Hao-Jie Zhou[1,2] (周浩杰), and Guo-Jie Li[1] (李国杰)

[1] *Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, P.R. China*

[2] *Graduate University of the Chinese Academy of Sciences, Beijing 100039, P.R. China*

E-mail: zxu@ict.ac.cn; zhouhaojie@software.ict.ac.cn; lig@ict.ac.cn

Received April 28, 2006; revised July 18, 2006.

Abstract This paper addresses the problem why grid technology has not spread as fast as the Web technology of the 1990's. In the past 10 years, considerable efforts have been put into grid computing. Much progress has been made and more importantly, fundamental challenges and essential issues of this field are emerging. This paper focuses on the area of grid system software research, and argues that usability of grid system software must be enhanced. It identifies four usability issues, drawing from international grid research experiences. It also presents advances by the Vega Grid team in addressing these challenges.

Keywords grid computing, system software, usability, distributed system architecture

1 Introduction

The idea of grid computing can be traced back to the utility computing concept of the 1960's. It was revived in the 1990's with the introduction of metacomputing[1], and later the concept of computational grids[2], analogous to electrical power grids. In the early 2000's, grid computing research flourished worldwide, with a lot of efforts put into building wide-area, decentralized IT infrastructure, or cyberinfrastructure[3], whereby various resources and services can be provisioned on demand. Viewed as "the next phase of distributed computing"[4], grid computing has grown into a rich field, embracing distributed application solutions, business services, distributed middleware, distributed system software, and even servers and storage hardware.

Grid system software is an important research area in the grid computing arena. Its current landscape can be summarized as follows: 1) the architecture of grid system software has gone through a number of transitions, and is converging towards into a style of high-performance service-oriented architecture, as exemplified by the Open Grid Service Architecture (OGSA)[5] proposed by the Global Grid Forum (GGF); 2) with grid computing embraced by both academia and the industry, many products and services are available, either as open source or as proprietary offerings; 3) many grid applications and services are available now, utilizing various grid system software technologies, ranging from scientific research to financial applications and online games.

However, grid technology spreads much slower than Web technology of the 1990's. There are many reasons for this. Coordinated resource sharing and collaboration, in a wide area and crossing multiple administrative domains, is a far more complex problem than sharing static Web pages, thus must overcome more technical hurdles. There are more standards efforts needed by grids than the HTML, URL, HTTP needed by Web. Grid system software is still a young field, lacking both theoretical understanding and mature, stable implementation.

Another main reason is that it is still difficult to use grid technology, from grid construction, resource deployment, application development, system management, to end user experiences. Much research efforts in the past were devoted to the basic services of OGSA. With these services gradually in place, there is a growing need to address the usability issues.

In this paper, we examine the status and trends of grid system software, from the user's viewpoint. The emphasis is on identifying the usability gap. Different grid environments, usage modalities and usability issues are analyzed, from representative international grid research projects. We also discuss research advances from the Vega Grid team in addressing usability issues.

The rest of this paper is organized as follows. In Section 2, we analyze grid system environment and summarize usage modalities from representative grid system software efforts in the international community (e.g., OGSA, TeraGrid, Globus, NAREGI, and China National Grid). Section 3 discusses usability issues. Section 4 presents research advances in the Vega grid software suite. Section 5 draws conclusion remarks.

2 Usage Modalities in Grid Environments

In this paper, usability issues refer to those requirements of grid system software which affect ease of use. To better understand these issues, we need to carefully examine the grid environments, the users involved, and the usage modalities.

2.1 Environment for Grid System Software

Fig.1 illustrates a conceptual grid environment for the discussions in this paper. A grid consists of four types of elements: users, applications, system software, and sites.

Survey

This work is supported in part by the National Natural Science Foundation of China (Grant Nos. 60573102, 90412010) and the National Grand Fundamental Research 973 Program of China (Grant Nos. 2003CB317000, 2005CB321800).

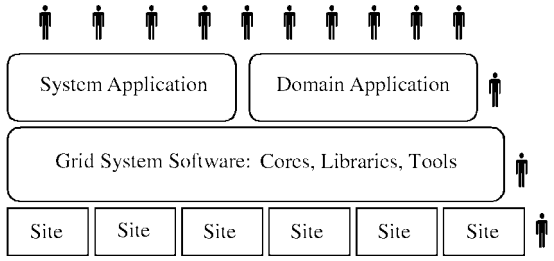


Fig.1. Four types of elements in a grid: users, applications, system software, and sites.

Physically, a grid consists of multiple sites, which are distributed in a wide area (in different cities or even different continents) and connected by a public network (e.g., the Internet) or a private network (e.g., a VPN). Each site is a computing center or data center, providing physical resources such as computing/storage hardware, local operating systems, application software packages, licenses, data, instruments and sensors. The sites in a grid are usually owned and managed by multiple administrative domains. This paper makes the simple assumption that each site corresponds to a domain. In reality, a domain could manage multiple sites, and a site could host multiple domains.

On top of the interconnected sites runs a layer of grid system software, which is responsible for integrating the sites into a cohesive grid system, thus eliminating silos. The grid system software consists of three types of components: one or more always-on runtime subsystems (cores), libraries and services, and user operable utility tools.

The grid system software supports two types of applications. Domain applications are those that provide domain-specific application functions, such as a bioinformatics application, an enterprise information dashboard, a manufacturing logistics application, an online games application, or an integrated e-government application. System applications provide values common to all application domains, such as grid resource accounting, security/trust management, searching and mining of grid resources information. The distinction is not absolute between system applications and utility tools of the system software.

In real world usage patterns, there are many types of grid users, and many ways to classify them. Fig.1 illustrates one scheme to classify users into four basic types:

• *End users* use a domain (or system) application to attain some business (common) value from the grid. They normally are aware of the grid system software only through the utility tools provided, but are not necessarily concerned with API or other details.

• *Developers* (application developers) use the grid system software to develop domain or system applications. They may be fully aware of all grid system software interfaces.

• *Grid administrators* manage the grid as a whole. They are most concerned with the management tools of the grid system software, from user management to security policies, service provisioning, configuration management, monitoring and accounting, and management of virtual organizations.

• *Site administrators* each manage a site. They need to interact with the grid system software by specifying physical resources/services and the mapping between grid and site users.

2.2 Classification of Grids

There are hundreds of on-going grid projects worldwide (see [6] for a sample of operational and planned grids). From the users' viewpoint, these grids can be projected into a two-dimensional space. We select a few representative grid systems to illustrate the classification scheme in Fig.2.

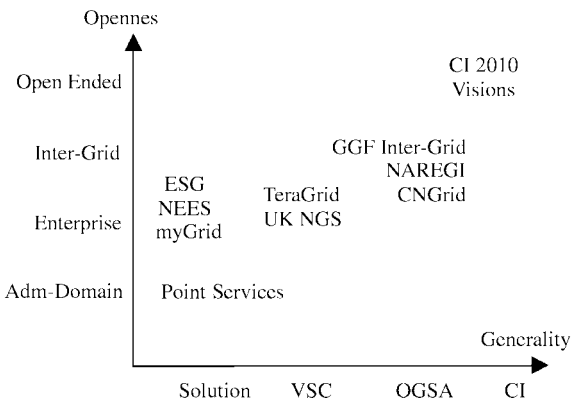


Fig.2. Classification of grids by openness and generality.

The vertical axis in Fig.2 shows the openness of administration and controlled accesses for a grid. The most closed grids have one administrative domain, and all administration policies and rules are dictated by the domain manager.

An *enterprise grid*, having multiple administrative domains, is often governed by a committee of stakeholders, who establish a cohesive set of policies and rules for user management, resource/service accesses, security, scheduling, accounting, etc. However, such a grid does allow access and sharing of resources previously owned and controlled by individual domains.

The next level of openness comes with *inter-grid systems*, where multiple interoperable grids must open their resources and policies for a wider scope of sharing. Users of one grid can construct, deploy, or utilize resources in another grid, although the two grids may have different sets of policies and rules. GGF is now advocating a cooperation project called Grid Interoperation (GIN) to interconnect main national grids.

Finally, a long term goal is to have *open ended grid systems*, similar to the Web today. Any individual, being a scientist or a school kid, can become a legitimate user in a matter of a few seconds, without explicit committee approval. The same individual can also contribute to the grids as easily as putting an HTML page on the Web. He could autonomously create and deploy a resource, a service, even a grid, without going through a lengthy apply-approve process.

The horizontal axis of Fig.2 shows the scope of generality visible to the users. The most focused (thus least general-purpose) grids are those that provide a solution for a specific application domain. Three examples are shown in Fig.2. The Earth System Grid (ESG)[7] aims to providing a grid solution for modeling and simulation of Earth's climate. NEES[8] is an application grid for a Network for Earthquake Engineering and Simulation. The myGrid[9] provides a grid solution for bioinformatics applications.

Such *solution grids* are the most numerous today, especially in the USA and Europe. They directly provide domain-specific application values to the users, and encourage cross-discipline cooperation between business domain specialists and computer professionals. Users mostly see business-level interfaces, not the grid system software, as the latter is often hidden in the application solution, even encapsulated in the application codes (see Fig.3).

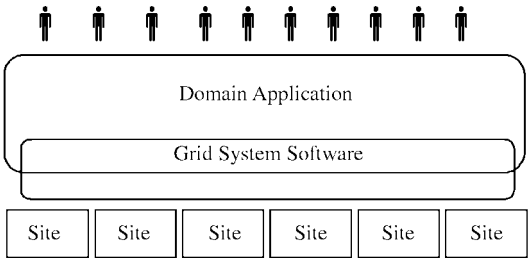


Fig.3. Solution grid.

A *virtual supercomputer* (VSC) grid provides a general-purpose platform, not specific to any application domain. Two typical examples are the US TeraGrid[10] and UK National Grid Service (NGS)[11]. Like a traditional supercomputer, such a grid provides uniform user management (such as identification and authentication), a home directory for each user, and scratch spaces for temporary files. A user can upload his own application codes and data, and execute interactive commands as well as batch jobs.

Unlike a single supercomputer, a VSC grid integrates several sites offering high-performance computing, storage, and visualization functions. The grid system software provides a uniform environment, such that users can expect, by default, the availability of the same interoperable grid software tools and services on every site. This is illustrated in Fig.4.

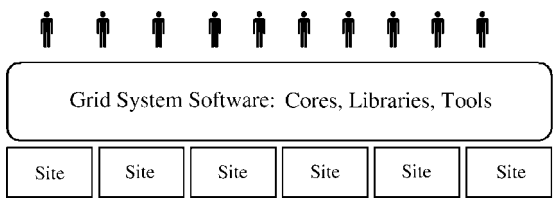


Fig.4. Virtual supercomputer (VSC) grid.

For instance, on TeraGrid, each site provides a set of grid software tools known as the Common TeraGrid Software Stack (CTSS). The current CTSS includes the Globus 2.4 software for common grid runtime, GridFTP for cross-site data transfer, MyProxy for user credentials, GSI for grid security, Condor-G for job submission, and MPICH-G2 for grid-enabled message passing. Additional values are added by a Storage Resource Broker (SRB) service, accounting information service, visualization tools, etc.

The OGSA mode is more general than the VSC mode in the following three senses. First, it employs a service-oriented architecture, and attempts to provide grid system software functions uniformly as services. This is different from the previous approach of a bag of ad hoc tools, and could alleviate the problem of users needing to know the tedious details of each tool and worrying about the tools interoperability. Second, OGSA aims to providing more than the bare essentials of a supercomputer environment such as user management, files storage and transfer, interactive and batch jobs management. The on-going OGSA standards efforts have made advances in sophisticated data management, information utilization, workflow, and semantic grid services. It enables grids to transition from a scientific computing platform to a more general IT platform, delivering services for computing, data management, information management, and knowledge processing. Third, an OGSA grid supports the concept of virtual organization (VO), a dynamic partition of grid resources/services and associated policies. *OGSA grids* are illustrated in Fig.5.

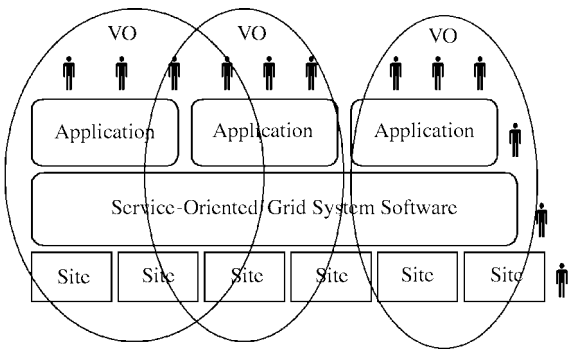


Fig.5. OGSA grid.

Some recent grid projects adopted the OGSA mode, such as China National Grid (CNGrid)[12], the Japanese National Research Grid Initiative (NAREGI)[13], and several semantic grid efforts[14]. The next phase of TeraGrid will incorporate the Globus Toolkit 4 grid system software[15], moving towards the OGSA mode. In this regard, the NAREGI grid software is the most aggressive. At its release in late 2007, it will implement dozens of OGSA related GGF standards or drafts.

Cyberinfrastructure (CI), as illustrated in Fig.6, is still a set of ongoing research goals[3]. Cyberinfrastructure differs from the previous three modes in several important ways.

Multiple grids. In addition to multiple VOs, multiple grids are supported on the same infrastructure, including solution grids, VSC grids, and OGSA grids. This im-

plies that 1) the grids can share the same infrastructure platform, and 2) the grids can access one another, thus sharing and collaboration can cross boundaries of grids. These grids use and become part of the infrastructure.

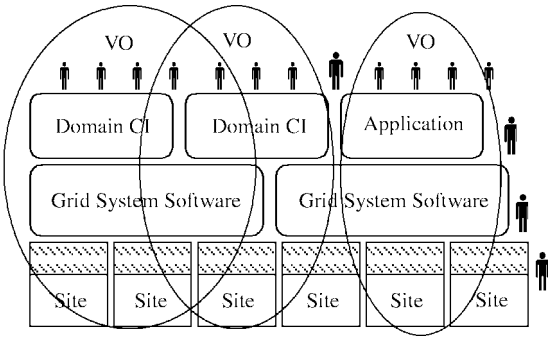


Fig.6. Illustration of cyberinfrastructure.

Shared infrastructure. Cyberinfrastructure emphasizes an interoperable infrastructure shared by all application domains, on top of which domain-specific CIs can be built. This shared infrastructure offers common supports for collaboration and sharing of "HEC (high-end computing) resources, massive data stores, broad and deep databases, high-bandwidth networks, middleware, shared software libraries and tools, services, people, and domain-specific software"[3].

Virtualization. To support the dynamic, diverse environment of multiple grids, multiple application domains, and multiple VOs, the sites should be "grid-oriented" or "grid enabling". A common approach is to utilize virtualization technology. For instance, IBM has selected virtualization as the main theme for its next generation servers, to support its On Demand initiative. In the open source community, various virtual machines are used, such as Java virtual machines, XEN[16], grid virtual machine[13], and service virtual machine[17]. In ideal cases, each sites can configure a suitable software stack and environment, including local operating system image, application software, libraries, and provide proper bindings and policy mappings for the grid in need. These resources plus the bare metal hardware are properly configured and provisioned on demand.

Empowering individuals. Cyberinfrastructure is not geared towards big organizations only, where all resources, services, policies are well defined/deployed and managed by committees and IT professionals. Instead, cyberinfrastructure, by providing all necessary plumbing, could drastically lower the barriers. It enables individuals, illustrated by the bigger human figures in Fig.6 who do not belong to any team, to access and consume resources as easily as big teams. Furthermore, it also empowers individuals to *produce*: to construct a grid, initiate a VO, deploy an application service, or contribute a resource. Peter Freeman *et al.*[3] believe that: "in the long run, the empowerment of individuals and small teams may be where the most significant changes in S&E (science and engineering) take place."

2.3 Usage Modalities

From a single user's viewpoint, we can project the usage modalities into a two-dimensional space depicted in Fig.7, where a sample of such modalities are shown. The vertical axis of Fig.7 measures the giving factor, which also indirectly indicates how much the user is aware of other users. While a lot of users just consume grid resources, contribution-based usage cannot be ignored. Many users, including end users, are contributing to the grid, by creating, producing, and deploying resources, services, applications, and VO's. Collaboration among users can be viewed as containing both consumption and production.

The horizontal axis of Fig.7 measures the interactivity factor, which also indirectly indicates how much the user is aware of the responsiveness of grid system.

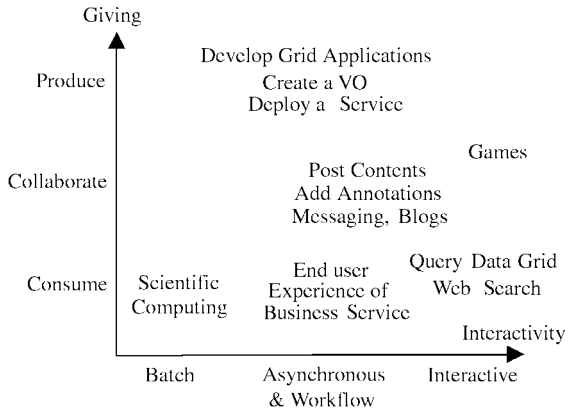


Fig.7. Single user usage modalities in a grid.

In an online game usage mode, a user sends the game grid interactive commands or events, and expects immediate responses from the game grid and from other players. Studies for interactive user experience show that "crisp" responses within 150 milliseconds by the system are preferred[18].

Users have bigger latency tolerance with other usage modalities. In scientific computing on a virtual supercomputer grid, a user submits a job to the grid and only expects the batch scheduler timely responses that the submission is indeed successfully accepted by a batch queue. In a business process workflow scenario, the user can also tolerate a reasonable delay after a user input.

3 Grid Values and Usability Issues

From an end-user's viewpoint, a grid system must provide both functional and nonfunctional values, for the grid technology to spread. There are two types of functional values:

- *Business value*: values derived from the realized grid application functions that satisfy the application domain needs, such as drug screening or information dashboard;
- *Grid value*: integration, resource sharing, and collaboration in a decentralized, dynamic, wide area environment.

There are three types of nonfunctional values:

• *Security*: authentication, authorization, availability (and reliability), trust, privacy, etc;

• *Efficiency*: low cost, low overhead, high utilization, and high quality of services;

• *Usability*: ease of use issues, such as the four issues of single system image, portability, agility, and managed code discussed below.

These values must all present for the grid technology to become widely deployed. In the past, not enough attention is paid to usability issues. Usability is more than just designing a graphic user interface, grid portal or problem solving environment, it must be considered when designing the architecture and standards of grids.

In addition to supporting various usage modalities, a deeper requirement of grid usability is to liberate users. A grid system, with its system software core, libraries, configuration files, environment variables, tools, and system applications, should take care of tedious, routine details, and prevent these overheads from appearing in user actions or application codes.

Furthermore, the grid system software should strive to satisfy the usability requirements without too much overhead. Currently, calling a trivial grid service (e.g., an Apache Tomcat+Axis service or a WSRF service) needs a few dozens milliseconds. Adding security features increases this overhead to hundreds of milliseconds[19]. Calling a GRAM service on a grid based on Globus Toolkit 4 could take 0.4∼2 seconds. In the current implementations of the Japanese NAREGI grid[13] and the China National Grid[12], with their virtualized services and other features to support usability and security, calling an end-user level service need to call several Axis or WSRF type of services. Consequently, calling a NAREGI service needs tens of seconds, and calling a CNGrid service could exert an overhead of 2∼7 seconds.

There already exist grids that are fast and have good usability. However, these are mostly solution grids, where the efficiency and usability issues are resolved by the application developer and embedded in the design and codes of the application software. The challenge for the next few years is to develop general-purpose grid system software that is both efficient and easy to use.

Four usability requirements are identified below that the grid system software should support. By "support", we mean that the grid system software should enable the user to construct a grid exhibiting these features. While grids and service-oriented architecture have the potential to enhance usability, the current implementations still have a large gap from the requirements. Grids today are far behind traditional computer systems and the Web technology, with respect to the four usability issues of single system image, portability, agility, and managed code.

Single System Image. The grid system software should offer simple, uniform rules for the user to design and use the proper level of single system image. One analogy is the Web technology, at the heart of which is a single space of URL links, which theoretically is the entire Web. Such simplicity and clarity of single system image is one reason why the Web spreads so fast.

This is not the case with current grids. The usual case is that single system image issue is not explicitly discussed. A bag of techniques are implicitly offered, such as URL, XML namespace, WSDL, Web addressing endpoints, file directories, without a coherent set of guidelines for constructing and using a grid. In an application grid, a user sees application level services. In TeraGrid, a user sees his home directory, scratch spaces, and multiple supercomputers. In CNGrid, a user sees a number of VO's called agoras, where user-level services crossing multiple sites can be accessed.

Portability. The grid system software should itself be portable. But more importantly, it should support application portability and service portability among different VO's and grids. An application program, written in some high-level language, should be portable on different grids, just as it is portable on different computers, with the support of compilers. A service should be portable on different grids, just as a hardware device is portable on different computers, with the same system software and hardware platforms.

We do not yet have any system software like a grid compiler. It is not easy to port an application grid, for instance myGrid, to the TeraGrid or the UK NGS platforms. We still cannot do it in the way we port a scientific application program, for instance NASTRAN, from a Unix platform to a Linux or Windows platform.

Agility. The grid system software should support changes in application requirements, security and resource use policies, grid services, grid system software, and the hosting environment, without unnecessary human involvement. One of the key advantages of grids and SOA is loose coupling and support for agility. Changes in one part do not require an overhaul of the entire system.

While some solution grids already provide for business agility, the current implementations of grid system software still lack good support for agility. Often the application codes need to be modified to accommodate changes in resources or policies. Many grid applications need to access a service via its URL. When the URL changes, often the application codes have to be modified.

Managed Code. The grid system software should support the "do no harm" philosophy, by enforcing complexity boundaries, lest one user's grid application should affect other users or the grid system. Again, this is not well supported in today's grid system software implementations.

4 Vega Suite of Grid Software

The Vega Suite of grid software aims to supporting system software requirements for the cyberinfrastructure of Fig.6. Usability issues are addressed through a loosely coupled architecture and a set of systems abstractions.

4.1 Architecture of the Vega Suite

Fig.8 depicts the Vega Suite architecture. The suite consists of four components. The grid operating system core and utilities (Vega GOS) provide common supports. For those CI's that focus on information sharing, integration, and management, the Vega information grid software (VIG) provides additional functions. The VINCA software provides functions for those CI's that require supports for workflows and business processes. GSML is a new XML-based language that supports application integration and end-user interaction. Users who prefer not to learn a new language can use traditional Java plus APIs from Vega GOS, VIG, and VINCA.

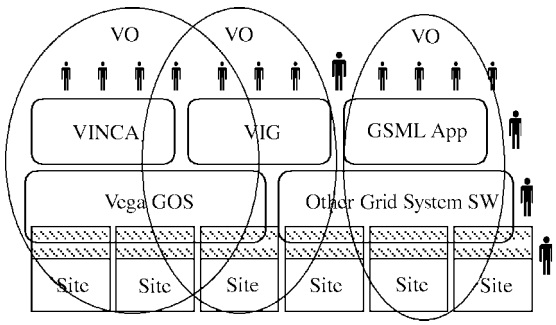


Fig.8. Illustration of the Vega Suite of grid software.

These software packages are loosely coupled, meaning they can be used separately and evolved independently. This is analogous to the LAMP suite of open source software (Linux, Apache, MySQL, PHP/Perl/Python). GSML, VINCA, and VIG run on top of the Vega GOS, but they can also run on other grid system software platforms, such as Apache Tomcat/Axis or GT4. In addition, the Vega Suite is designed to support interoperability with other grid system software. A grid built with the Vega suite can access resources in another grid built on top of different grid system software.

4.2 System Abstractions

The Vega Suite employ four abstractions called *resource space*, *grip*, *agora*, and *funnel*. The last one is to aid grid programming and discussed elsewhere[20]. Below we discuss how the first three abstractions help address the usability issues.

To support the cyberinfrastructure with multiple VO's and multiple grids, the application developer and the site manager face the challenge of dealing with multiple interfaces. The Vega strategy is to separate the development of applications and resources from grid system software.

When a resource (e.g., a BLAST program for gene sequence) is developed and then deployed as a *physical service*, it is done following the WSDL standard, independent of any grid system software concerns. Then the system software can link the physical service (e.g., named grid.net/blast2006) to a *virtual service* in a grid.

When a domain application such as a bioinformatics workflow is developed, it is done again independent of any grid system software concerns. The application codes only use *effective services*, which are business (application-level) services, instead of physical services. When and after the application is deployed on a grid, these business services are bound to virtual services, which in turn are bound to physical services.

This service virtualization scheme, utilizing the abstraction of effective-virtual-physical resource spaces, help achieve portability and agility. For instance, when the physical service blast2006 is changed to blast2007, or its URL is moved, there is no need to modify applications using this service. When an application is ported to another grid, we only need to establish the necessary mappings among the resource spaces.

To support a proper level of single system image and managed code, the Vega Suite employs the abstractions of agora and grip, and separate application logic from context and policies. An agora is a realization of the VO concept. It maintains two sets of members, the members of users and the members of services. In addition, it maintains context and policies shared by the members. Current agora implementation supports context and policies related to security (authentication, authorization, access control) and resource use (e.g., load balancing, accounting). The important point is that the application developer does not have to write codes to take care of such concerns.

When a grid application is evoked to run, a grip is created for the lifecycle of the application. The codes of the grip include two parts. One part corresponds to the application logic and associated data structure, which is provided by the application developer. The other part corresponds to the common support for virtualization, context, policies, accessing physical services, and other low-level details (e.g., XML documents translation, SOAP messaging, exception handling), which is provided via the Vega Suite interface.

A grip is to grid what a process is to a traditional operating system. However, unlike a process, a grip can run on multiple nodes of a site, or even on multiple sites, while still maintain a single system image of coordinated resources and services.

The grip and the agora are system abstractions, and can provide automatic checking against bad effects such as invalid or unauthorized access to grid resources and services. However, the current Vega Suite design cannot realize fully managed codes. If a user writes a malicious code in the application logic without using the Vega Suite interface, the Vega grid system software cannot catch it. This exists in other grid and SOA systems, too. For instance, a grid allowing a user to submit a batch job has limited protection if the job contains malicious codes. An approach offering better supports of managed code is proposed in [17], which uses a service virtual machine to dynamically check all application codes.

4.3 Implementation and Evaluation

Part of the work in Vega GOS presented above have been implemented and deployed in China National Grid[12]. The application experience with CNGrid has been invaluable to the new design of the Vega Suite. Most ideas presented above have been implemented in Vega Suite version 1 (alpha). Preliminary tests show that the Vega approach is feasible in enhancing supports for grid usability.

For instance, although the current Vega Suite implementation is based on J2SE and Apache Tomcat/Axis, we have demonstrated that the Vega Suite can work with international grid system software platforms such as GT4[15]. The Vega Suite can use GT4 as its kernel. Alternatively, a grid application based on the Vega Suite can access a GT4 grid service via a simple gateway, utilizing the grip and the agora constructs.

The Vega Suite also has improved performance than the China National Grid system software. With all features supporting service virtualization, security and context enabled, the overhead for calling a business service has been reduced to around 0.4 seconds. This improvement is mainly due to a more efficient implementation of the grip construct. There is still much room for efficiency improvement, as we have not utilized locality well yet. In addition to the obvious case of *service locality*, we are also exploring the cases of *context locality* and *policy locality*.

5 Conclusions

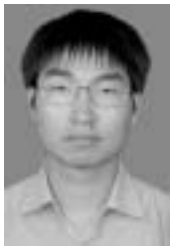
Good usability is a must for grid technology to be widely used. By reviewing international grid research experiences and usage modalities, we have identified four usability issues: single system image, portability, agility, and managed code.

The Vega Suite addresses the usability issues with a loosely coupled architecture and four system abstractions. Preliminary evaluations show that this is a feasible approach. It can enhance usability, while at the same time maintain security and improve efficiency.

References

[1] Catlett C, Smarr L. Metacomputing. *Comm. ACM*, 1992, 36(6): 44–52.
[2] Foster I, Kesselman C (eds.). The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, 1998.
[3] Freeman P A, Crawford D L, Kim S T *et al.* Cyberinfrastructure for science and engineering: Promises and challenges. In *Proc. the IEEE*, 2005, 93(3): 682–691.
[4] Global grid forum website. http://www.ggf.org.
[5] Foster I *et al.* The open grid services architecture. http://ggf.org/documents/GFD.30, 2005.
[6] Parashar M, Lee C A (eds.). Special issue on grid computing. In *Proc. the IEEE*, 2005, 93(3): 479–714.
[7] Bernholdt D *et al.* The earth system grid: Supporting the next generation of climate modeling research. In *Proc. the IEEE*, 2005, 93(3): 485–495.
[8] http://www.nees.org/.
[9] http://www.mygrid.org.uk/.
[10] http://www.teragrid.org/.
[11] http://www.ngs.ac.uk/.
[12] Zha L *et al.* System software for China National Grid. In *Proc. the 2nd IFIP Int. Conf. Network and Parallel Computing*, Beijing, China, Springer, LNCS 3779, 2005, pp.14–21.
[13] Matsuoka S *et al.* Design and implementation of NAREGI super-scheduler based on the OGSA architecture. *Journal of Computer Science and Technology*, July 2006, 21(4): 521–528.
[14] D de Roure, N R Jennings, N R Shadbolt. The semantic grid: Past, present, and future. In *Proc. the IEEE*, 2005, 93(3): pp.669–681.
[15] Foster I. Globus Toolkit version 4: Software for service-oriented systems. *Journal of Computer Science and Technology*, July 2006, 21(4): 513–520.
[16] www.cl.cam.ac.uk/Research/SRG/netos/xen.
[17] Wang X *et al.* Abacus: A service-oriented programming language for grid applications. In *Proc. 2nd IEEE Int. Conf. Services Computing*, Floarida, USA, 2005, pp.225–232.
[18] Tolia N, Anderson D G, Satyanarayanan M. Quantifying interactive user experience on thin clients. *IEEE Computer*, 2006, 39(3): 46–52.
[19] Humphrey M *et al.* State and events for Web services: A comparison of five WS-resource framework and WS-notification implementations. In *Proc. 14th IEEE Int. Symp. High Performance Distributed Computing*, Virginia, USA, 2005, pp.24–27.
[20] Shu C *et al.* BEAP: An agile end-user programming paradigm for business applications. *Journal of Computer Science and Technology*, July 2006, 21(4): 609–619.

Zhi-Wei Xu received his Ph.D. degree from University of Southern California in 1987. He is a professor and deputy director at Institute of Computing Technology, Chinese Academy of Sciences. His research areas include distributed computing, high-performance computing architecture and net-centric operating system, He serves on the editorial boards of Journal of Grid Computing, Journal of Parallel and Distributed Computing, and IEEE Trans. Computers.



Hao-Jie Zhou is a Ph.D. candidate at Institute of Computing Technology, Chinese Academy of Sciences. His research interests include grid file management and grid security.

Guo-Jie Li received his Ph.D. degree from Purdue University in 1985. He is a professor and director of Institute of Computing Technology, CAS. His research areas include CPU micro-architecture, social computing, grid computing, and high-performance computing architecture. He serves as President of China Computer Federation and Editor-in-Chief of Journal of Computer Science and Technology.