

# An Engineering Computation Oriented Visual Grid Framework

Guiyi Wei<sup>1,2,3</sup>, Yao Zheng<sup>1,2</sup>, Jifa Zhang<sup>1,2</sup>, and Guanghua Song<sup>1,2</sup>

<sup>1</sup> College of Computer Science, Zhejiang University, Hangzhou, 310027, P. R. China

<sup>2</sup> Center for Engineering and Scientific Computation, Zhejiang University, Hangzhou, 310027, P. R. China

<sup>3</sup> Hangzhou Institute of Commerce, Hangzhou, 310035, P. R. China

**Abstract.** Grid computing technology is a focused field in high performance computing. This paper describes an engineering computation oriented visual grid framework VGrid, which is capable to bridge the gap between currently deployed grid services and the computational applications. Based on the Globus toolkit, and coupled with a client component, a services pool and a server component, VGrid visually performs resource discovery, task schedule, and result processing. VGrid improves the efficiency of utilization of resources by introducing a logical resource concept. VGrid applications of numerical simulations in engineering sciences are demonstrated.

## 1 Introduction

Computer simulations become increasingly more important in studying physical systems and engineering designs. However, even today's most powerful supercomputers were utilized, the scope and accuracy of these simulations are still severely limited by the available computational power. When we endeavor to simulate the true complexity of nature and engineering process, we will require much larger scale calculations than those are possible at present. We can break through these limits by simultaneously harnessing multiple networked supercomputers, running a single massively parallel simulation to numerically model the problems of more complexity and high fidelity [1].

In order to solve these engineering and scientific computing problems, grid computing has emerged as a new infrastructure, distinguished from conventional distributed computing since that it focuses on large-scale resource sharing, innovative applications, and, in some cases, high-performance orientation [2]. The sharing that we are concerned with is not primarily file exchange but rather direct access to computers, software, data, and other resources, as is required by a range of collaborative problem-solving and resource brokering strategies emerging in industry, science, and engineering. The shared resources are autonomous, distributed, heterogeneous and dynamic. At the present time, there are at least two available pieces of grid middleware, such as Globus [3] from Argonne National Laboratory, and Legion [4] from University of Virginia. The Globus adopts OGSA [5] architecture. There are many successful

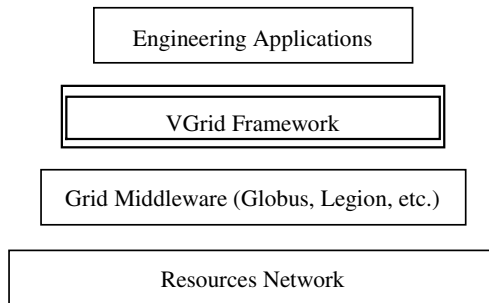
applications based on the Globus, such as Cactus, OVERFLOW-D2, X-ray CMT, SFExpress, MM5, and Nimrod [6]. The Legion uses the object-oriented technology [9]. There exist many successful applications based on the Legion, such as NAS, DSMC, and NPACI [7].

Although grid computing achieved great successes in many fields, it still needs to be improved in some aspects. That is due to the facts:

1. The emergence of standards in grid technologies needs more scientists' efforts in various fields. Their efforts make grid technology meet the requirements in these fields.
2. The gap between currently deployed grid services and the would-be user community has been largely stretched.

## 2 VGrid Framework

VGrid locates between grid middleware (such as Globus and Legion) and engineering applications as shown in Figure 1. It provides an integrated development environment, abstracts basic grid services (such as authorization, authentication, MDS, GRAM, GASS, and etc.) empowered by grid middleware, and provides high level VGrid services oriented to engineering applications. Using VGrid services through their APIs, applications can be developed and deployed more efficiently under grid computing environment. VGrid framework consists of client component, application middleware component and server component, as shown in Figure 2.

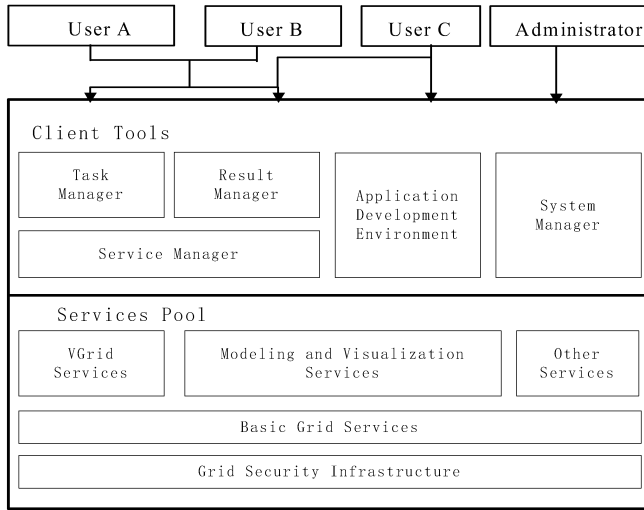


**Fig. 1.** Location of VGrid in the grid environment

The client component is a portal of VGrid and provides a graphical user interface for development and execution of grid-based applications. It comprises a VGrid system manager, a services manager, an application development environment, a task manager and a result manager.

Application middleware component is actually a hierarchical services pool. All services in the pool are based on the Grid Security Infrastructure supplied by the Globus Toolkit. Taking consideration of resources security, the using of each service must base on a success of authentication and authorization.

Server component is composed of distributed resources manager in the virtual organizations. This resources manager communicates with local hosts through the network. They provide resources such as computer power, storage, software, data, and visualization devices.



**Fig. 2.** VGrid architecture

In order to satisfy different requirements, VGrid users are classified into three levels, that is User A, User B, and User C, according to their motivation. User A has some developed application programs, but does not know about grid technologies. He only wants to find out proper resources to execute his programs. User B also has some developed application programs, and has some acquaintance with grid technologies and utilization of the grid-based distributed resources. He wants to choose the proper resources to deploy his programs and to monitor the processes of submission, execution and result processing. User C wants to develop applications in the grid or to upgrade an existing program to make it fit with the grid environment. He is very familiar with grid technologies, and wants to take full advantage of grid services to develop and to deploy his applications in order to utilize use resources more efficiently.

VGrid provides three type services VS (VGrid services), GS (grid services), and MVS (geometry modeling and visualization services):

1. **GS:** Provided by the Globus Toolkit, it includes MDS (Metacomputing Directory Service), GRAM (Globus Resource Allocation Manager), and GridFTP [8].
2. **VS:** It includes virtual organization register, user manager, resource information services, task definition, task import and export, task submission, task scheduling, task monitor, and file transfer. A VGrid user can accomplish resources discovering, information transfer and task running by using these services in a visual fashion.
3. **MVS:** It is oriented to engineering computation and scientific visualization. It includes geometry modeling, geometry meshing, data format conversion, local visualization, remote visualization, and distributed visualization.

### 3 Implementation of VGrid Framework

#### 3.1 Visual Resources Discovery in the VGrid

The execution of a computing task needs a lot of resources. The resources are distributed, dynamic and autonomous in a grid environment. How many resources are available? Which should be allocated to a specified task? These questions must be answered during task scheduling. To answer these two questions, resources information services provide two interfaces, a hierarchical resources tree, and a resources list. The content of this tree is detailed information of the hosts in virtual organizations, to which the local host registered. It also shows the architecture of those virtual organizations. Each item in the list is information of an individual host. The information on the list can be customized for display. Generally it consists of host name, node count, CPU count, speed and free percentage of CPU, free memory size, free secondary storage, network bandwidth and latency. A user can get enough information from this tree and the list for task schedule. The resources a task needs can be allocated through this interface.

In the VGrid, resources are divided into two classes, computing power and visualization devices. The computing power includes personal computers, PC clusters, workstations and supercomputers. The visualization devices are special distributed display devices, and are used to display high fidelity images and sophisticated graphical results of engineering computations.

VGrid adopts object-oriented technology to describe resources. Each attribute item in a resource object is stored by a couple of value as <attribute, value>. All information in a virtual organization is encapsulated as <node\_id, res\_info>, where the “node\_id” is a host name or IP address of the host, and the “res\_info” is an instance of class “MdsResult”, which is used to describe the information of a resource in detail.

#### 3.2 Definition of a VGrid Task and Dynamic Resource Allocation

A session for each user is defined in the VGrid. The session is created when one logs in, and destroyed after one logs out. In the session, a user defines computation task, possesses resources, monitors task execution, collects results, and does post-process for the results. In the VGrid, the session is described as a SCB (session control block), which includes user information, identification of the user for each task.

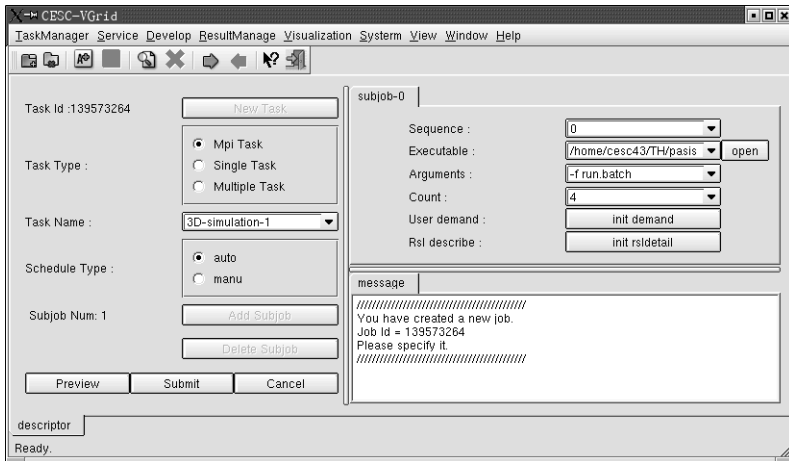
The VGrid provides a graphical user interface as depicted in Figure 3, to input a new task or to update an existing task. The data include executable program, resources requirements, and schedule requirements. When VGrid accepts the data, an instance of task class is initialized and added to logical resources allocation queue. The task’s state is initialized to value “WAITING”. The C++ definition of VGrid task class is defined as follows:

```

class VTask{
public:
    enum TaskType { Single = 0, Mpi = 1,
                  Multiple = 2 };
    enum TaskStyle { CpuBusy = 0, IOBusy = 1};
    enum TaskState { Scheduling = 0, Scheduled= 1,
                   Finished = 2 };
    enum InTaskQueue { ScheduleQueue = 0,
                     SubmittedQueue = 1,
                     FinishedQueue = 2 };

private:
    struct Sid      sessionid;
    QString         taskname;
    QString         jobidstring;
    struct JobRsl   jobrslstring;
    TaskState       taskstate;
    struct TaskQueue taskqueue;
    struct QTime    starttime;
    QString         finishtime;
    int             priority;
    QString         jobcontact;
    ...
};

```



**Fig. 3.** A snapshot of defining a task in the VGrid

The dynamic resource allocation includes two steps, that is logical resources allocation and physical resources allocation. During the logical resources allocation, tasks in the queue possess logical resources by the FIFO algorithm. Logical resource is a type of virtual resource defined by the VGrid. It includes resource name, node count, and CPU count. The physical resources are not allocated until the task is scheduled to execute, because resources in the grid are dynamic. The performance of VGrid can be improved by introducing the concept of logical resource. The next step of this phase is task schedule, which maps logical resources to physical resources. A task is added to ASQ (Auto Schedule Queue) or ISQ (Interactive Schedule Queue), according to its

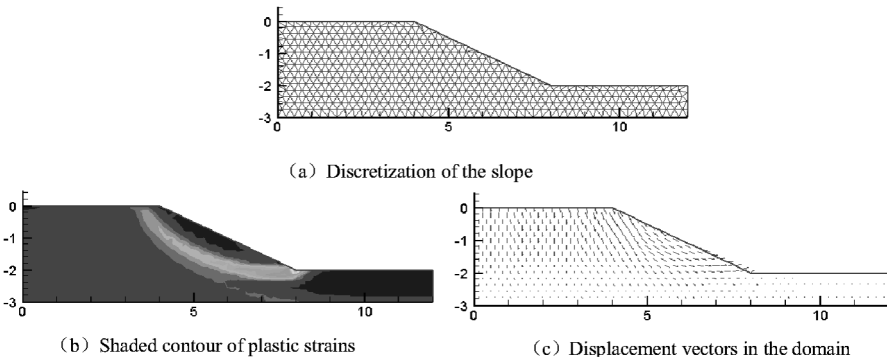
attribute of schedule control after logical resources allocation. Tasks in ASQ will be scheduled automatically. But tasks in ISQ should be scheduled manually, that is, a user can filter and select resources from available list manually as he needs in a graphical interface.

After the task gets enough resources, its state is changed to “READY”, then it will be add to EQ (execute queue). Tasks in the EQ state are actual executable jobs and will be submitted using the FIFO algorithm by VGrid executor, which is a daemon named Vexecutor.

## 4 Engineering Computational Applications with the VGrid

Engineering sciences (such as solid mechanics, fluid mechanics, and thermodynamics) and most physical sciences allow physical systems to be described in a form of Partial Differential Equations (PDE). Due to their complexities, most of these equations could not be solved analytically. With the advances of computer technology, numerical methods become preferred. Finite Element Method (FEM) is a popular approach among these methods. The method, coupled with the development in computer technology, has been successfully applied to the solution of these problems in a nonlinear manner for all spatial domains.

The process of FEM analysis includes three steps: 1) discretization of the domain, which transforms the PDEs into algebraic equations; 2) formulating the algebraic equations to the global equilibrium equations; and 3) solving the algebraic equations. In this section, two use cases of VGrid-based applications are to be presented. The first case is a 2-D numerical simulation for a safe factor of a slope under its self-weight in geotechnical engineering. Four grid computation nodes (single CPU, 2.0GHz) are utilized for the pre-process, computing, and post-process during the simulation. It takes 7.6 seconds in the computing stage. The results of the simulation are shown in Figure 4.



**Fig. 4.** The simulation results of the first case

The second case is a 3-D numerical simulation of the deformation of a mechanical device under a surface pressure. It is more intricate than the first case. To test the speedup and scalability, the VGrid runs the simulation program with the same parameters for two times using different number of computational nodes networked with 100M bandwidth. The types of the computation resources used and the time consumed in this case are shown in Table 1, the results of the simulation are shown in Figure 5.

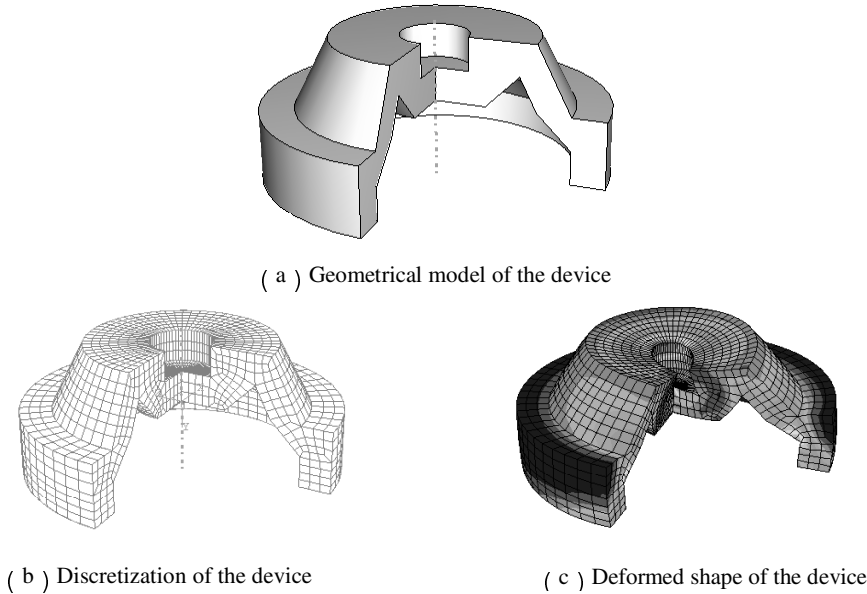


Fig. 5. The simulation results of the second case

Table 1. Used computation resources and consumed time of the second case

Runs	Types of Machines	Nodes	Time Consumed
1	Intel Pentium IV PCs ( P4 2.0G Hz )	3	87.3 sec
	Intel Pentium IV PCs ( P4 1.8G Hz )	1	
2	Intel Pentium IV PCs ( P4 2.0G Hz )	6	61.4 sec
	Intel Pentium IV PCs ( P4 1.8G Hz )	2	

5 Conclusions

An ideal application oriented grid computing framework should have a friendly user interface, and provide a virtual computing environment composed of many heterogeneous resources. Users are enabled to develop and to deploy applications easily with the computing power, provided in the grid environment, just like using a

supercomputer. The VGrid supplies the users with a graphical grid-enabled application development and deployment environment for engineering computation. It hides detailed complex processes of grid middleware. Users are able to use all kind of resources in the virtual organizations transparently. The VGrid improves the efficiency of development and speed up the execution of engineering applications in grid computing environment by using VGrid services. Therefore, it is an efficient environment for engineering computation.

**Acknowledgements.** The authors wish to thank the National Natural Science Foundation of China for the National Science Fund for Distinguished Young Scholars under grant Number 60225009. We would like to thank the Center for Engineering and Scientific Computation, Zhejiang University, for its computational resources, with which the research project has been carried out.

## References

1. Gabrielle Allen, Edward Seidei, John Shalf: Scientific Computing on the Grid: Tomorrow's Dynamic Applications Will Require Computational Grids, *Scientific Computing*, Springer, 2002
2. Ian Foster, Carl Kesselman, Steven Tuecke: The Anatomy of the Grid: Enabling Scalable Virtual Organizations, *Int. J. Supercomputing Applications*, 2001, 15(3)
3. I. Foster, and C. Kesselman (eds.): The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, 1999
4. Anh Nguyen-Tuong, Steve Chapin, Andrew Grimshaw, Charlie Viles: Using Reflection for Flexibility and Extensibility in a Metacomputing Environment, *Technical Report CS-98-33*, Department of Computer Science, University of Virginia, 1998
5. J. Nick, I. Foster, C. Kesselman and S. Tuecke: The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, *Open Grid Service Infrastructure WG*, Global Grid Forum, 2002
6. Globus Alliance, URL: <http://www.globus.org/research/applications/default.asp> (Current September 14, 2003)
7. Legion: A Worldwide Virtual Computer, URL: <http://legion.virginia.edu/overview.html> (Current September 14, 2003)
8. I. Foster, J. Insley, G. von Laszewski, C. Kesselman, and M. Thiebaux: Distance Visualization: Data Exploration on the Grid, *Computer*, 32(12): 36-43, 1999
9. Andrew S. Grimshaw, Michael J. Lewis, Adam J. Ferrari, John F. Karpovich: Architectural Support for Extensibility and Autonomy in Wide-Area Distributed Object Systems1, *Technical Report CS-98-12*, June 3, 1998