

# ELECTRONIC SEISMOLOGIST

March/April 2005

**Thomas J. Owens**

**E-mail: [owens@sc.edu](mailto:owens@sc.edu)**

**Department of Geological Sciences**

**University of South Carolina**

**Columbia, SC 29208**

**Phone: +1-803-777-4530**

**Fax: +1-803-777-0906**

## Over the network and to the Grid ...

The folks at the Southern California Earthquake Center are rising rapidly to the top of the Electronic Seismologist's Favorite People List. First, because they are doing interesting work in keeping seismology on the leading edge of the IT world. Second (and far more important in their ESFPL ranking), they are more than willing to share their experiences through contributions to the ES. This issue the subject is "Grid Computing". Although not the most over-hyped IT term at the moment (that would be "Web Services"), Grid Computing is a phrase that gets tossed around in a lot of settings where the ES suspects those involved may not have a firm grasp of what it really means. As luck would have it, the ES knows a few things about grid computing, having pioneered the method in the early 1980's. Over Thanksgiving Break. In today's vernacular, the ES ran his own Virtual Organization back then when traveling from Utah to Lawrence Livermore National Laboratory to access data and computer resources (and to bicycle ... a lot). In addition to being a key financial supporter of the ES's PhD research, LLNL is conveniently located near the Fall AGU venue and Thanksgiving is conveniently placed a couple of weeks before AGU. And, all those lab folks actually take Thanksgiving off! Recognizing this unique opportunity to start a revolution in information technology (and finish his AGU talk), the ES spend a couple of Thanksgivings at LLNL running dozens of Receiver Function inversions (12 Hours each). Grid Security was no problem in the early days, once you got past the people with uniforms and guns. Resource Discovery and Monitoring ... wander the halls and computer rooms looking for free systems! Data Transfer ... 9-track tapes. Job submission ... walk around to as many machines as you could find and run a script. After that, it got a little boring. Fortunately, these were also the early days of microwave dinners, so the ES could settle in for one or more pseudo-turkey dinners while his Grid worked. Now, there have been a few minor innovations since this original seismological foray into grid computing. My friend Al invented the Internet, which quickly replaced the roly chair that the ES used to get from machine to machine after those turkey dinners, and then the SCEC folks came along and made the whole thing purr. The ES apologizes if any other contributors may have been overlooked ...

## **Grid Computing in the SCEC Community Modeling Environment**

**Philip Maechling (1), Vipin Gupta (1), Nitin Gupta (1), Edward H. Field (2), David Okaya (1), and Thomas H. Jordan (1)**

**(1) Southern California Earthquake Center**

**University of Southern California**

**Los Angeles, CA 90089-0742**

**Telephone: +1-213-740-5843**

**Fax: +1-213-740-0011**

**Email: [maechlin@usc.edu](mailto:maechlin@usc.edu), [vgupta@usc.edu](mailto:vgupta@usc.edu), [niting@usc.edu](mailto:niting@usc.edu), [okaya@usc.edu](mailto:okaya@usc.edu), [tjordan@usc.edu](mailto:tjordan@usc.edu)**

**(2) U.S. Geological Survey**

**525 South Wilson Avenue**

**Pasadena, CA 91106-0001**

**Telephone: +1-626-583-7814**

**Email: [field@usgs.gov](mailto:field@usgs.gov)**

**<http://www.scec.org/cme>**

### **Dynamic Communities Sharing Computer Resources**

In our work on the Southern California Earthquake Center Community Modeling Environment (SCEC/CME) Project (Jordan et al, 2003), we are developing computer systems to support dynamic distributed scientific collaborations. Scientists participating in SCEC collaborations are often willing to share their computer resources particularly if, in return, they can gain access to computing capabilities that they do not currently possess. Inter-organizational computer sharing can be difficult to achieve due to the many organizational and technical differences between groups. Recently, however, a new software technology called grid computing (Foster et al, 2001) has emerged which is designed to help dynamic organizations, like SCEC, share heterogeneous collections of computer and storage resources.

## **Grid Computing and Virtual Organizations**

Grid technology enables organizations to share computer resources with other organizations even when the shared computers are administered differently and they have dissimilar hardware and operating systems. Organizations can create a grid environment to provide their users with computer resources such as CPU cycles, disk storage, and software programs that are available outside of their local computer administrative domain. This is done by creating a new computer administrative domain referred to as a Virtual Organization (VO). A VO has its own set of administrative policies that represents a combination of local computer policies, the computer policies of the groups you are sharing with, plus some administrative policies required by the VO itself. When we run a program on the “grid”, we are saying, in a sense, that our program is running outside of our own local administrative domain. Grid middleware is used to facilitate the execution of computer programs in a VO.

In addition to creating multi-organizational administrative domains, grid middleware also strives to hide the heterogeneity of the shared computing environment. Grid software provides a set of commands to perform basic computing operations and these commands are the same regardless of the underlying computer and operating system.

### **Basic Grid Computing Capabilities**

Grid computing is built upon four basic capabilities. These capabilities are; (1) Security (2) Data Transfer (3) Job Submission and (4) Discovery and Monitoring of Computing Resources. Grid computing is based on the premise that these four capabilities are the basic building blocks required to share computer resources in a meaningful way. The following table briefly describes each of these fundamental grid computing capabilities.

<b>Grid Computing Capability</b>	<b>Description of Functionality</b>
<b>Grid Security</b>	Identify computer users and computers in the grid and define what each user is permitted to do.
<b>Data Transfer</b>	Transfer data from one computer to another.
<b>Job Submission</b>	Run a computer program on a local or remote computer.
<b>Resource Discovery</b>	Determine what computers and what data storage devices are in the grid and the status of these resources.

**Table 1: Fundamental Grid Computing Capabilities**

Before we describe these grid functions in detail, let's look at how these capabilities can be combined to share computer resources. Assume a user wants to run a program on a remote, grid-enabled, computer. First, the user runs a grid security program to establish their identity on the grid. Then, the user issues a grid monitoring command to confirm that the remote computer isn't too busy. Then, the user runs a grid data transfer command to move their program, and input files, from their local computer to the remote computer. Now, they issue a grid job submission command to run the program on the remote computer. Finally, they use a grid data transfer command to copy the resulting output files back to their local computer for further analysis.

In our following discussion, we describe grid commands that are provided by our grid software. While these commands are a useful starting place, and they help explain basic grid capabilities, we should point out that users typically don't interact with the grid using these basic grid commands because the commands are quite cumbersome. The real grid computing payoff comes when grid commands are called programmatically from application programs. Then,

complex systems can be built on top of grid services. When this is done, grid-based computer sharing occurs transparently to the user. The grid-based OpenSHA Hazard Map application described in a companion article (Field et al, 2005) is an example of how grid computing can be so well integrated into an application program that the use of the grid becomes transparent to the user.

### **Relationship between Grid Computing and Distributed Computing**

Grid computing is a type of distributed computing. In an earlier ES article (Maechling et al, 2005) we discussed a variety of distributed computing techniques including Java Servlets, Java RMI, CORBA, and Web Services. We are sometimes asked how those distributed computing technologies are related to grid computing. To answer this, we start by characterizing those other technologies as distributed component technologies. Software developers utilize distributed components to execute programs on other people's computers. However, distributed components do not provide general purpose computing capabilities. Organizations offering distributed components are offering fixed solutions. As long as you want to use the distributed component exactly as defined by the organization that deployed it, then the system works. However, if you have your own version of a component, you cannot immediately begin to run it on someone else's computer. You must negotiate the deployment of your version of the component. Grid computing, in contrast, offers a general purpose computing environment on other people's computers. Once the grid VO is established, you can run your own component on someone else's computer. So the grid provides a general-purpose distributed computing environment. This is a more powerful capability than running only existing distributed components.

## Addressing Grid Hype

Within the computer science world, particularly within the high performance computing community, there has been a lot of interest in grid computing over the last few years. For example, there is now a collection of supercomputers in the U.S. called the TeraGrid ([www.teragrid.org](http://www.teragrid.org)) that is configured to support grid computing. The level of interest and activity in grid computing has, in some cases, risen to the level of grid hype. Grid hype can hurt organizations in a couple of ways. For one, it leads to unrealistic expectations that grid computing cannot meet. For another, it can attract organizations into grid computing that are not equipped to handle the additional system administrative burden required to establish and maintain a grid.

One of the more common unrealistic grid computing expectations is that you can plug a grid-enabled computer into a network port and immediately get free, or low cost, computing cycles. While easy sharing of computer resources through grid technology may eventually make this possible, grid technology has not yet reached this level of ease of use.

Another expectation is that grid computing is a replacement for parallel computing technologies such as computational clusters. Grid software works with computational clusters. For example, it can provide easier access to clusters by providing standardized job submission, data transfer, and monitoring commands. But grid computing does not replace clusters computing. Besides technical issues such as the very high speed network connections used by clusters, an important distinction is that computational clusters are typically homogenous collections of computers, which grids are typically heterogeneous collections.

There are additional grid issues. Grid software and administration is currently quite complex and there are very few experienced grid administrators. Our SCEC grid requires a

significant amount of system administrator time. If your collaboration will benefit from sharing computers, grid software may work for you. However, there will be costs associated with rolling out a grid system including training, system administration time, support and maintenance of the grid, and user learning time.

### **Establishing the SCEC Grid**

On the SCEC/CME Project, we built the SCEC grid using the Globus Toolkit ([www.globus.org](http://www.globus.org)) which is the grid software standard in the scientific and academic research world. The Globus Toolkit is an open-source software distribution that is available for download from the Globus web site. The Globus Toolkit is a collection of software programs that, once installed and configured on your computer, provides basic grid functionality.

We installed the Globus Toolkit on several of our SCEC computers and we configured our grid software so that we could access computer resources at collaborating institutions. For example, the SCEC grid computers are configured to share computer and storage resources with the USC High Performance Computing and Communications (HPCC) ([www.usc.edu/hpcc](http://www.usc.edu/hpcc)) group, with collaborating groups at USC's Information Sciences Institute (ISI), and with the TeraGrid network of supercomputers.

As we mentioned previously, the SCEC/CME OpenSHA working group has implemented a Probabilistic Seismic Hazard Analysis (PSHA) Hazard Map program that uses grid software. By using grid software, this PSHA program can be run on a large shared collection of USC workstations called a Condor Pool. The OpenSHA software that performs these hazard map calculations is written primarily in Java. The PSHA calculations performed by this software consist of a series of hazard curve calculations. A hazard map with dimensions of 100km x 100km and a grid spacing of 1km requires 10,000 hazard curve calculations. These hazard curve

calculations are independent and they can be performed in any order. Each hazard curve calculation outputs at least one file.

A second seismological application that we have grid-enabled is an Anelastic Wave Model (AWM) earthquake wave propagation simulation program written by Kim Olsen called AWM-Olsen (Olsen et al, 1997). AWM-Olsen is used by SCEC researchers for a wide variety of geophysical research. For example, it was used to run the TeraShake (Minster et al 2004) simulations in Fall 2004. AWM-Olsen is a 4<sup>th</sup> order finite difference Fortran-90 program that utilizes the Message Passing Interface (MPI) in order to run on computational clusters. We use grid software on SCEC computers to submit this program to both the USC and TeraGrid computational clusters and to transfer the large input and output files between the computational clusters and local SCEC disk storage. As we describe the basic capabilities of grid computing, we'll comment on how grid computing can support these two very different seismological applications.

### **Grid Security**

Globus designers recognized that every grid operation had to be highly secure. If Globus isn't secure, people won't use it to share. Basic Globus grid security is a two step process; Step 1) verify a user's identify. Step 2) verify that the identified user has permission to use the grid resources that they are trying to use. Step 2 is dependent on Step 1.

In a Globus grid, every user is issued a trustworthy identification called a grid certificate. When a user tries to run a grid command, their grid certificate is sent along with the command so the target computer system can determine who has issued the command. Every time a user tries to run a grid command, they must prove their identity with a grid certificate. In security terms, proving one's identify is called authentication. By using robust grid certificates, Globus software

ensures that it can reliably identify all users in the grid. Computers are also issued grid certificates so that the Globus software can reliably identify all computers in the grid.

The other half of Globus grid security is called authorization. Once a user proves their identity using a trustworthy grid certificate, Globus checks to see if that person has permission to run the command they have issued. So when a user issues a grid command, Globus first checks their identification, then it checks whether that person has permission to do what they are asking to do. It is entirely possible for a grid system to accept a user's identification, but to deny that users' grid command due to lack of authorization.

Globus uses Public Key Infrastructure (PKI)-based grid certificates. Grid certificates are issued and signed by a Certificate Authority (CA). A CA is typically the computer system administration department of an organization. Just as with personal identifications, certain CA's are stricter, or more demanding, than others. A strict CA won't issue a grid certificate without substantial verification that you are who you say you are. It is often the case that strict CA's are more widely accepted than less strict CA's. Personal ID's often work the same way. For example, the local fitness center may issue you an ID without asking many questions. However, not many other organizations will trust that ID. The federal government is significantly more demanding. When you apply for a passport, they take your picture and your fingerprints and they check up on you before they issue you a passport. But once you get the passport, it is widely trusted throughout the world.

This leads us to an important practical issue that organizations face as they begin to use grid computing. To start using Globus software, your organization must decide the following security issues: 1) which certificate authority (CA) will issue the grid certificates that your users, and computers, will use? Also, 2) which certificate authorities you will trust?

For a test environment, an organization can act as its own CA and issue its own grid certificates. However, if an organization wishes to interoperate with external grids, they will need to find a CA that all participating organizations trust. In the Internet world, an organization called ICANN ([www.icann.org](http://www.icann.org)) is charged with coordinating the names and numbers used on the Internet. There is no equivalent centralized grid Certificate Authority so organizations usually implement their own CA and coordinate the use of their grid certificates with other organizations. Access to, or operation of, a Certificate Authority is one of the administrative overheads associated with Grid computing.

In the case of our grid-based PSHA program, we want to utilize computers in the USC campus grid. SCEC faculties are on the USC campus, so our users, and computers, are issued grid certificates signed by the USC Certificate Authority. USC accepts its own grid certificates, so SCEC grid computers can interoperate with computers on the USC grid using USC CA-signed grid certificates.

For the AWM-Olsen program, we want to submit jobs the TeraGrid. In order to interoperate with the TeraGrid, USC spent a substantial amount of time working with TeraGrid security groups to agree upon appropriate computer security policies and procedures. After significant review, and some policy updates, USC and the TeraGrid agreed to accept each others grid certificates. So now, when SCEC users issue grid commands to be executed on TeraGrid computers, the SCEC users prove their identity using USC CA-signed grid certificates.

Organizations are understandably cautious about which Certificate Authorities they will trust and therefore, which grid certificates they will accept. In our experience, CA issues (authentication issues) are the most time consuming administrative aspects of setting up a grid.

Table 2 shows an example of a commonly used Globus security command.

<b>Globus Command</b>	<b>Globus Security Infrastructure (GSI)</b>
<b>Example Command:</b>	<pre>% grid-proxy-init -hours 2  Your identity: /C=US/O=USC/OU=SCEC/CN=Philip Maechling/UID=philipm  Enter GRID pass phrase for this identity:  Creating proxy .....  Done  Your proxy is valid until: Mon Mar  7 17:37:16 2005</pre>
<b>Description:</b>	<p>A user enters a pass-phrase to verify their identity. Once the pass-phrase is accepted, grid commands issued from this account will use that identify for the next two hours.</p>

**Table 2: Globus Grid Certificate Initialization Command Example**

### Common Grid Security Issues

Before we leave the topic of grid security, we'd like to mention three specific security issues that organizations are often concerned about. These issues are; 1) Can an organization limit the use of their grid to only approved individuals? 2) Can an organization prevent a trusted user's grid-based program from damaging their computer, or data? 3) Can grid users transfer data across the grid without exposing the data in clear text?

The first issue is addressed by Globus with the two-part authentication and authorization-based grid security system described earlier. To use an organization's system, a user must present a trusted identification, a grid certificate. Once the user is reliably identified, the grid software will then verify that the user has permission to issue grid commands on the specified system. Properly configured, grid software does enable organizations to limit use of their computers to approved individuals only.

The second issue is important once trusted users are allowed to run programs, or perform other grid operations, on an organization's computers. Can an organization protect their shared, grid-enabled systems from accidental, or malicious, activities of trusted users? Typically, this is handled by mapping external users to local computer accounts. When an external user issues a grid command, the grid software maps the external user to a local user account. Then, the external user has all the permissions of the local user, but no more. For example, the local account may have a disk quota, and so the external grid user will be limited to the quota of the local account to which they are mapped. By mapping remote grid users to local accounts, the disk allocation and file access permissions of external grid users can be controlled and remote grid access can be reasonably safe.

There are some grid tools, such as the Condor system that we discuss later, that provide a "sandbox" based approach for running grid programs. In a "sandbox" based system, external programs run in a secure well-controlled region of the computer and they are prevented from accessing anything outside this "sandbox". This technique is helpful if the grid users don't have local accounts on all of the grid-enabled computers.

Globus addresses the third concern, transmission security, by providing the capability to encrypt data during transmission using Secure Socket Layer (SSL) software that is bundled with Globus. Even sensitive data sets like passwords and financial data can be transferred securely using Globus grid tools.

### **Grid Data Transfers**

Once grid software is installed, and the grid security issues are worked out, grid commands, such as data transfers, can be issued. Globus data transfers use a program called GridFTP. GridFTP has been optimized for high performance transfers with capabilities such as

parallel transfers and partial file transfers that are not commonly found in other versions of FTP. When transferring files using GridFTP, the source and destination files are specified using Uniform Resource Locators (URLs) like the URLs that locate web pages. This means that files transferred with GridFTP must be placed in locations that are externally visible as URLs. Table 3 shows an example of a Globus URL Copy command that copies a file from a remote computer to a local file.

<b>Globus Command</b>	<b>Globus Data Transfer (GridFTP)</b>
Example	<pre>% globus-url-copy gsiftp://earth2.usc.edu/tmp/testfile2.txt file:///tmp/testfile1.txt</pre>
Description:	This command will copy a file from the computer earth1 to a file in a local directory called /tmp/testfile2.txt

**Table 3: Globus Data Transfer Command Example.**

The data transfer requirements for our two seismological applications are fairly similar. In order to run either of these programs on a remote computer, we copy the executable to the remote computer, copy the input parameter files (if any), start up the program on the remote computer, and, when the calculations are done, we copy the resulting output files back to our local computer.

### **Grid Job Management**

Next, let us consider Globus Job Management. In the Globus world, job management refers to two main capabilities; 1) job submission, and 2) job monitoring. Job submission refers to the process of starting a program on a computer. Job monitoring refers to determining what happened after the program started. We will focus on job submission here, but Globus also provides job monitoring capabilities.

Those of us that primarily run programs on personal computers or workstations do not commonly work with job submission programs. For the most part, we just double click the program icon, or we type the program name, hit the “ENTER” key, and the program starts to run. For UNIX users, the most common job submission programs are the “&” (ampersand) operator that runs the program in the background, and the “at” and “cron” commands that schedule programs to run at a specific times.

However, when you are submitting your program to run on a collections of computers (e.g. a pool of computers), or if you are submitting your program to run on a computing clusters, job submission is more complex. On these systems, programs are submitted to a job submission manager often using a job submission script. Submitted programs are placed in a job queue by the job submission manager and a program runs when it reaches the front of the queue. Job queues are managed by a job scheduler program that uses some type of scheduling algorithm. From the system operator’s perspective, it is important to keep the system as busy as possible as long as there are programs in the queue. From the user’s perspective, it is important to minimize the wait time before their job runs.

There are a variety of job submission managers, and each job submission manager has its own job submission language. Job submission systems used in the SCEC grid include Condor ([www.cs.wisc.edu/condor](http://www.cs.wisc.edu/condor)) and the Portable Batch System (PBS) ([www.openpbs.org](http://www.openpbs.org)), each of which has its own scripting language.

Globus implements yet another job submission scripting language called Resource Specification Language (RSL). RSL is a scripting language that can be used to submit a job to run on a Globus grid. RSL is designed to be a universal job submission scripting language that can be translated into any other job submission scripting language. Globus takes an RSL

command and translates it into the appropriate underlying job submission language. Because Globus translates RSL into a variety of underlying job submission languages, RSL can be used as a universal job submission language. Table 4 shows an example of an RSL command that submits a job for execution on a remote host.

<b>Globus Command</b>	<b>Globus Resource Allocation Management (GRAM)</b>
Example	% globus-job-run earth1.usc.edu -s myprog
Description:	This command will submit the program called “myprog” to execute on the computer earth1.usc.edu and will copy the executable to the target machine if necessary.

**Table 4: Globus Job Submission Command Example**

Our two example grid-based seismological application programs have significantly different job submission requirements and they illustrate how Globus helps to support a heterogeneous computing environment.

The characteristics of our PSHA hazard map program make it an ideal candidate to run on a collection of independent computers because we run the same program repeatedly and because there are no dependencies between the runs. The USC HPCC group has configured a collection of over 100 campus workstations as a “pool” of computers that is available for general computing when they are not busy. This collection of computers is called a Condor Pool. Programs can be run on computers in the Condor Pool by using a job submission program called Condor. The Condor job manager monitors all the computers in the Condor Pool and runs the job at the front of the queue on the next available computer. USC HPCC has installed a version of Condor, called CondorG, which works with Globus.

To submit our PSHA program to the USC Condor Pool, we create a Globus RSL script and submit the RSL script to the Globus job manager. Globus then converts our RSL script to a Condor script and then submits the Condor script to the Condor job submission manager. The Condor job submission manager then places our PSHA program in the Condor Pool queue, and the Condor job scheduler runs the program on the next available computers.

Running the AWM-Olsen program requires a significantly different type of job submission script. The AWM-Olsen program runs on computational clusters. Clusters often use job submission managers such as the Portable Batch System (PBS) to handle user job submissions. It's worth noting how the queuing approach for clusters reverses the queuing approach for a Condor Pool. Condor establishes a single queue over a large collection of computers. PBS establishes many queues, and each queue refers to portions of one large computer. For example, the job queues on the USC HPC Linux Cluster vary by the number of processors that the job will run on, and by the interconnection (e.g. Ethernet, Myrinet) between nodes accessed by the queue.

To run AWM-Olsen on the USC HPC Cluster, or on the TeraGrid, we create an RSL submission script and submit it to the Globus job submission manager. Globus then translates the RSL script into the appropriate underlying PBS commands, and then submits the PBS commands to the cluster's own job submission manager for execution.

### **Grid Monitoring and Discovery**

Before running a job on a remote computer, it is important to verify that the remote computer meets the minimum system requirements for your program. Globus provides a Monitoring and Discovery Service (MDS) to make this possible. MDS allows users to determine

information about computers in the grid such as the type of CPUs, the operating system, the amount of computer memory, how busy the system is, and file system information such as size and free space. Table 5 shows an example of an MDS command and the type of information that is returned.

Globus Command	Globus Monitoring and Discovery Services (MDS)
Example	<pre>% grid-info-search -h earth1.usc.edu -x dn: Mds-Host-hn=earth1.usc.edu,Mds-Vo-name=local,o=grid Mds-Cpu-model: Intel(R) Xeon(TM) CPU 1 Mds-Cpu-speedMHz: 1394 Mds-Os-name: Linux Mds-Memory-Ram-Total-sizeMB: 4800 Mds-Cpu-Total-Free-15minX100: 385 Mds-Device-name: /usr/local Mds-Fs-sizeMB: 9844</pre>
Description:	<p>This Globus Monitoring and Discovery command returns detailed information about computers in the grid including the operating system, type of CPUs in the system, the amount of RAM memory, the free time of the CPUs, and file system information.</p>

**Table 5: Globus Monitoring and Discovery Command Example**

The Globus MDS system separates system monitoring capabilities from the query and reporting capabilities for performance and convenience reasons. In the background, Globus continuously monitors the grid, and places status information into a cached schema. When a user queries for system status, status information is retrieved out of the cached schema. By using this

caching system, users can query a single system, and Globus can respond quickly with information about all the systems in the organization's grid.

### **Grid Issues and Risk Reduction Strategies:**

No grid computing discussion is complete without comments on limitations with current grid systems. One significant issue regarding grid software is that it is changing rapidly. Globus, in particular, has been changing versions quite frequently. Due to this rapid rate of change, Globus installations around the country have a variety of Globus versions deployed, which leads to compatibility issues. On the SCEC/CME Project, as our baseline, we use the version of Globus that is distributed in the current National Science Foundation Middleware Initiative (NMI) ([www.nsf-middleware.org](http://www.nsf-middleware.org)) software distribution. When NMI releases a software distribution that contains a new version of Globus, we upgrade our systems with the new NMI release. NMI releases tend to be less frequent than new versions of Globus. We believe that the NMI releases are well tested, and that they are widely installed. Interoperability is high. However, the NMI version of Globus lags behind the latest Globus release so you don't get immediate access to new features.

Another issue to consider is that the grid software shakeout is just beginning. The Globus Toolkit is the de-facto grid software standard within scientific communities and therefore we believe it is a good choice for SCEC. However, there are other grid software tools available including versions from commercial vendors such as Sun, Avaki, Data Synapse and others. It is not clear what the grid software standard will be five years from now. The key to selecting grid software is to select standards-based grid software. The leading grid software standards bodies are the Global Grid Forum ([www.ggf.org](http://www.ggf.org)) and the World Wide Web Consortium ([www.w3.org](http://www.w3.org)).

Standards-based grid software from one vendor should interoperate with standards-based grid software from another vendor. We recommend working with grid software that is based on GGF and W3C software standards.

### **Discussion:**

We believe that SCEC and other research groups will benefit by sharing computer resources. Since grid computing provides at least a partial solution to the problem of sharing computer resources, it addresses a real need in the scientific community, and it is likely to persist in some form. We believe that grid software will eventually be integrated into operating system and network software installed on most computers.

As the technical issues related to grid computing are resolved, and as organizations begin to recognize the benefits of sharing computers, the challenges associated with using grids will shift from technical issues to organizational issues. Research organizations will need to develop new processes for defining priority of access, fair use, and compensation for use of shared computer resources. So, for those of us building collaborations, the long term significance of grid computing is likely to be less technical and more social because it challenges us to define, in very unambiguous terms, what sharing means within our collaborations.

### **References**

- Field, E.H., T.H. Jordan, and C.A. Cornell (2003). OpenSHA: A Developing Community-Modeling Environment for Seismic Hazard Analysis, *Seism. Res. Lett.* **74**, 406-419.
- Field, E.H., N. Gupta, V. Gupta, M. Blanpied, P. Maechling, and T.H Jordan (2005). Hazard Calculations for the WGCEP-2002 Forecast Using OpenSHA and Distributed Object Technologies, *Seism. Res. Lett.* **76**, 161-167.
- Field, E.H., V. Gupta, N. Gupta, P. Maechling, and T.H Jordan (2005). Hazard Map Calculations Using GRID Computing, in review; to be submitted to *Seismological Research Letters*.
- Foster I., C. Kesselman, and S. Tuecke (2001), "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *International Journal of Supercomputer Applications* **15**, No. 3.

- Jordan, T. H., P. J Maechling, and the SCEC/CME Collaboration (2003). The SCEC community modeling environment: an information infrastructure for system-level science, *Seism. Res. Lett.* **74**, 324-328.
- Maechling, P., Vipin Gupta, Nitin Gupta, Edward H. Field, David Okaya, and Thomas H. Jordan (2005) *Seismic Hazard Analysis Using Distributed Computing in the SCEC Community Modeling Environment* *Seism. Res. Lett.* **76**, 177-181
- Minster J.B., K. Olsen, R. Moore, S. Day, P. Maechling, T. Jordan, M. Faerman, Y. Cui, G. Ely, Y. Hu, B. Shkoller, C. Marcinkovich, J. Bielak, D. Okaya, R. Archuleta, N. Wilkins-Diehr, S. Cutchin, A. Chourasia, G. Kremenek, A. Jagatheesan, L. Brieger, A. Majumdar, G. Chukkapalli, Q. Xin, R. Moore, B. Banister, D. Thorp, P. Kovatch, L. Diegel, T. Sherwin, C. Jordan, M. Thieboux, J. Lopez, (2004), *The SCEC TeraShake Earthquake Simulation*, *Eos Trans. AGU*, 85(46), Fall Meet. Suppl., Abstract SF31B-05
- Olsen, K., R. Madariaga, and R. Archuleta (1997), *Three dimensional dynamic simulation of the 1992 Landers earthquake*, *Science* 278, 834-838.
- WGCEP - Working Group on California Earthquake Probabilities (1988). Probabilities of large earthquakes occurring in California on the San Andreas Fault, *USGS Open-File Report* **88-393**.