## What Is OUR Grid?

*Henry E. Schaffer, PhD(1), Virinder M. Batra(2), Todd S. Broucksou(3), Phillip A. Emer(3), Alan Galloway(1), J. Charles Kesler(3), Chi-Duen Poon, PhD(4), Bill Rankin, PhD(5), James E. Robinson, III(1), William Setzer(1), Steven C. Woods(3)*

The recent friendly discussions on defining the characteristics of a "grid" have taken on an intensity reminiscent of arguments over details of religious dogma. These discussions are important, and there are serious issues involved. However, our effort is not definitional, but is focused on the needs of the biotechnology research community (bioinformatics, genomics, proteomics) in North Carolina. This is a heavily practical approach, and we hope that our results can help others. Our grid is the NC Bioinformatics Grid, or, for short, the NC BioGrid. http://www.ncbiogrid.org/ Our presentation avoids the arguments about dogma, and deals with the characteristics **we** require to meet **our** needs.

We are also deeply aware that we are dealing with the characteristics we think will be needed to scale our test bed into a robust, efficient production grid serving higher education, not-for-profit research corporations, and the for-profit corporate sector (i.e., our virtual organization, cf. "The Anatomy of the Grid" by Ian Foster, Carl Kesselman and Steven Tuecke - http://www.globus.org/research/papers/anatomy.pdf). Our plans are to include all of these sectors in one grid. At present we have grid nodes operating in organizations in the first two sectors mentioned. This is a working grid, and although it is a limited test bed (see Table I) in the current phase, it is being implemented and being developed into a production grid along the specifications discussed below.
Our planning and implementation has divided the grid effort into three overlapping functional areas of concern as organizing principles:

1. Storage
2. Computation
3. Access

In all of these areas we require that the grid span multiple independently managed domains. Even the test bed that is now operating includes four domains (Duke, NC State, UNC-CH, and MCNC/NCSC) which are under separate administrations, and which generally do not coordinate their choices of equipment, systems and protocols. They individually chose their own security regimes, and use a variety of authentication and authorization methods. Similar considerations apply for enterprise storage, etc. Internetworking is the area in which in NC there is a great deal of cooperation and coordination. As a result, our grid effort has had few problems in internetworking. While not all grid environments require a similar adaptability to span multiple domains, it is a fundamental requirement for our mission. Similarly, we require that the grid include a diversity of platforms, with both hardware and operating system diversity. Our testbed (see Table I) already includes platforms from 2 different manufacturers, 3 different CPU platforms, and with 3 different operating systems. This diversity arises from

- selections made to optimize different criteria, by
- independently managed domains, and using
- selection criteria and purchasing decisions which change over time.

Another practical requirement which flows from our diversity of sites is the requirement of dynamic update in all aspects of grid function. In other words, it must be possible to add, modify and remove grid resources without downtime.

Each of the grid functional areas mentioned above can be broken into subcategories with requirements for each:

## STORAGE

- Integrate multiple distributed, heterogeneous, independently managed data sources. While large, centralized data stores make sense in many situations, we see a continuing need to deal with data sources at many or all of our sites for the indefinite future.
- Implement mechanisms to automatically "stage" input files and the applications to the hosts where the computation will take place, and to copy the resulting output to a specified central location. Note that by having the capability to stage applications, we remove the need for an administrator to maintain separate copies of the applications on all nodes in the grid.
- Provide data caching and/or replication to minimize network traffic and insure that data close to where it is needed for computation. This is required for efficiency, rather than being important in itself.
- Allow the user to find data based on characteristics - the physical location of the data should be transparent to the user. This is often called "data discovery".
- Implement data encryption and integrity checks (e.g. message digesting) to insure that data is transported across the network in a secure fashion.
- Provide the backup/restore mechanisms and policies necessary to prevent data loss and minimize downtime across the grid. This will be critical when we move into the production phase at a later date. For now, backup/restore is provided on a site-by-site basis.

## COMPUTATION

- Allow for independent management of compute resources - we require this for our main sites, but in an academic environment it is common to have further division into sub-sites which are independently managed. (This sub-site division should not cause any additional problem once there is the capability of working with independently managed sites.)
- Provide a "meta-scheduler" that can intelligently and transparently select compute resources capable of running a user's job. This entails understanding the current and predicted loads on grid resources, including the ability to interface with third party queuing systems (e.g. Platform LSF and Sun Grid Engine) that are front ending grid-enabled clusters. This could be called "compute power discovery".

- Provide job checkpointing so that a failed job from the point at which it failed. This is crucial for large jobs that may take many hours or days to run to completion.
- Parallel processing should be available when parallelized code is available for the systems on the grid. The goal is to reduce the job's wall clock time when possible.
- Insure that appropriate security mechanisms (e.g. encryption and integrity checks) are in place to protect the components of a job (the application, parameters and associated data) that are moved across the network and stored on a grid node as part of a computation.

## ACCESS

- Create a uniform name space so that resources can be addressed consistently across the grid. That is, the name by which a resource is referenced should be the same regardless of where it is accessed from on the grid. This is important in a multi-site, multi-domain environment.
- Allow local user IDs, security credentials & grid identity mappings to be independently managed at the "site" level, while centrally managing global grid user IDs.
- Provide fine grained access control mechanisms to restrict usage of grid resources based on individual and/or organizational identities.
- Provide the programming interfaces necessary to build GUIs (preferably browser based) for use by biologists for all of their grid based work - often referred to as a "portal". We consider this to be a necessity for use by our scientific community, but it is not required during our initial development work.

We believe that grid technologies (as advertised) represent a natural solution to the problems summarized by our storage, compute, and access requirements. To date we have tested and/or deployed middleware solutions from Avaki, IBM, Platform Computing, and Sun Microsystems in our testbed. Our testbed includes systems representative of the local operating environments of the respective nodes. For instance, one node supports a local Linux cluster computing environment that uses Platform Computing's LSF to provide local job scheduling functions in an AFS (originally Andrew File System) namespace. Another node supports a local compute cluster that employs Sun Grid Engine (SGE) to provide local job scheduling. We have deployed both Globus and Avaki middleware in our testbed to perform meta-scheduling across all nodes including those with local schedulers. They have different profiles of capabilities, and an intriguing question is whether they can be used simultaneously to better meet our requirements than can either one alone. We have deployed Avaki to provide a uniform name space and associated shared file system across administrative domains. We have created grid user instances and installed datasets and an application into the shared file system - specifically we have intstalled Linux, Solaris, and AIX executables of a single application. With our initial testbed configuration we have successfully launched jobs from grid nodes that host neither the data nor the executable. Simply put, the NC Biogrid testbed is operational.

## Conclusion

Our planning was based on the needs, current and projected, of our genomics, bioinformatics and proteomics research community. This led us to the requirements outlined above, and we are following them in our implementation. We consider that they are needed for the NC BioGrid's success. However we are trying to be pragmatic. We will not insist on a requirement which either cannot be achieved, or which cannot be achieved on our development schedule. We are willing to settle for a partial implementation, and to delay the full implementation until later on our schedule. In this way we expect to maximize the service provided, and minimize the time until service is provided to our research community.

Our grid comprises, and will continue to comprise, a heterogeneous set of compute and storage resources, schedulers, file systems, and security policies integrated with grid middleware into a distributed computing platform. That distributed computing platform allows a cross-domain IT team to manage a shared resource system that abstracts the details of that system from the end user community. And that is the NC BioGrid.

## Table I - NC BioGrid Nodes

Institutional Affiliations
(1) North Carolina State University
(2) IBM
(3) MCNC/NCSC (NC Supercomputing Center)
(4) University of North Carolina-Chapel Hill
(5) Duke University