The Global Bio Grid at Virginia
GBG@Virginia

Andrew Grimshaw, Gabrial Robins, Marty Humphrey, Mark Morgan

Department of Computer Science, University of Virginia

William Knaus

Department of Health Evaluation Sciences, University of Virginia

Abstract:

The future of the life sciences will depend on distributed, large-scale, integrated, computational infrastructures, called Grids. Indeed, as observed recently by the National Research Council, "biology is becoming an information science". An interdisciplinary group of biologists, computer scientists, and clinicians at the University of Virginia have teamed up to implement this exciting vision and to make it a reality in the near future.

The University of Virginia Center for Grid Research (CGR) was founded with the belief that the life sciences are the "killer application" for Grid computing, and with the commitment to transform life sciences research by providing the tools to Grid-enable the life sciences. The center has **four specific objectives**; we will:

(1) Develop and deploy the Global Bio Grid (GBG), a data, computational, and algorithmic infrastructure for the life sciences; The GBG will provide a shared, secure infrastructure for collaboration and research across academic, government, and industrial institutions – and across the spectrum of life sciences applications.

(2) Perform Research on dependable Grid and Service Oriented Architectures (SOA). Grid computing technology is still in its' infancy and many problems remain. Particular areas that we will focus on are data access, security (confidentiality, data integrity, access control, policy negotiation), dependability (availability, SLA's, policy languagues), and grid standards.

(3) Develop new algorithms, tools, and techniques for hierarchical data integration and analysis.

(4) Outreach to the life sciences community to both accelerate their science and to better understand their requirements and special needs.

This technical report gives an overview of Grid computing, life sciences use cases for Grid, and the Global Bio Grid technical plan and status as of September, 2004.

1 Our Vision: Biology as an Information Science

The future of the life sciences will depend on distributed, large-scale, integrated, computational infrastructures, called Grids. Indeed, as observed recently by the National Research Council, "biology is becoming an information science"[1]. An interdisciplinary group of biologists, computer scientists, and clinicians at the University of Virginia have teamed up to implement this exciting vision and to make it a reality in the near future.

The life sciences are experiencing a growing tidal wave of biomedical data from DNA sequencers, gene array chips, mass-spectrometry systems, clinical databases, longitudinal studies, and even embedded inside-the-patient monitoring systems. Like radio astronomy before it, the life science community now realizes that only by leveraging large-scale computational resources can these mountains of generated data be converted into information and knowledge. But unlike radio astronomy, where only a handful of radio telescopes generated data in a single domain, the life sciences contain millions of potential data sources from divergent disciplines containing different, and often incomparable abstraction levels, notations, formats, and space / time scales. This challenge is enormous along all conceivable dimensions: intellectual, algorithmic, computational, and data management perspectives.

The University of Virginia Center for Grid Research (CGR) was founded with the belief that the life sciences are the “killer application” for Grid computing, and with the commitment to transform life sciences research by providing the tools to Grid-enable the life sciences. The center has **four specific objectives**; we will:

(1) Develop and deploy the Global Bio Grid (GBG), a data, computational, and algorithmic infrastructure for the life sciences; The GBG will provide a shared, secure infrastructure for collaboration and research across academic, government, and industrial institutions – and across the spectrum of life sciences applications.

(2) Perform Research on dependable Grid and Service Oriented Architectures (SOA). Grid computing technology is still in its’ infancy and many problems remain. Particular areas that we will focus on are data access, security (confidentiality, data integrity, access control, policy negotiation), dependability (availability, SLA’s, policy languagues), and grid standards.

(3) Develop new algorithms, tools, and techniques for hierarchical data integration and analysis.

(4) Outreach to the life sciences community to both accelerate their science and to better understand their requirements and special needs.

Our cross-disciplinary team of researchers at the University of Virginia shares a common collaborative philosophy that "it is better to create bridges than walls", and we envision the Center for Grid Research as a "call to service". The Global Bio Grid will not only bring new computational, data integration, and analysis capabilities to the proposed projects, but it will also serve the international life sciences community in the coming decades.

We envision that the future of the life sciences will be fundamentally intertwined with grid computation. The notion of performing biological research a decade from now without grid computing will be as limited as studying evolution a century ago without knowledge of the

genome. In order to build bridges between biology and computing, the Department of Computer Science has partnered with the departments of Health Evaluation Sciences, Biochemistry & Molecular Genetics, Neurology, the Curry School of Education, the NIH-funded Diabetes and Endocrine Research Center, as well as institutions around the world in order to create a large-scale, integrated Grid infrastructure. We have identified the challenges facing the successful completion of a Global Bio Grid, and established a series of projects that will test its capabilities. More importantly, we have committed our time and resources to a joint and fully collaborative effort.

Inventing the Future

A decade from now, the world will be a very different place. Consider that ten years ago, there was no World Wide Web; the commercial aspects of the Internet were still in their infancy; wide-spread Internet connectivity was rare; broadband service to homes did not exist; a gigahertz machine was a multi-million dollar supercomputer; the first gigabyte disks became available; the human genome project was still a "grand challenge"; and bioinformatics was in its infancy. At that time, most biologists lacked access to high-performance computing, and indeed many would not have even thought it relevant to their work.

Today all this has changed dramatically. The Web is prevalent and iconic; billions of dollars are transacted through e-commerce (itself a new word) every week; broadband home access to the Web is ubiquitous; wireless Web access is exploding; multi-gigaflop processors sit on most desktops; a terabyte disk costs about $1,200; the human genome has been sequenced; new organisms sequencing is completed almost daily; and exponentially-growing torrents of new data are available to biologists.

What can we expect a decade from now? Terabit networks will span the globe; Web-services and grid computing technology will knit the world together into vast interconnected systems; secure sharing of information and applications will be commonplace; information technology will transform the way people live and how science is conducted; the cost of sequencing an individual human will be only a few thousands dollars; some countries (Singapore, for example) may sequence their entire population; clinical data generating smart "motes" (micro sensors) will be embedded inside patients; sequence data, expression data, clinical data, lifestyle data, and other information sources will be integrated together, analyzed, and manipulated; new insights into multi-factorial disease will be gained.

One of our driving goals is to exploit the inevitable advances in technological capability and accelerate knowledge and insight into multi-factorial disease by integrating a wide range of resources, data streams, computation infrastructure, and application tools.

How do we know that all of this will happen? As Alan Kay, inventor of object-oriented programming and the graphical user interface was fond of saying, "the best way to predict the future is to invent it". We intend to follow Alan Kay's advice: our cross-disciplinary team of researchers will build a Global Bio Grid that will embody new algorithms, tools, and applications, empower life science researchers worldwide, and enable the biology to mature into a true information science.

Integration

The term “integration” can mean many things to different people. We begin by defining two different aspects of integration, namely resource integration and semantic integration.

Resource integration: Life science research and clinical delivery are highly fragmented domains. The data, applications, providers, and users typically inhabit separate organizations, at different sites, with distinct access control policies. This has led to a model where most research and clinical delivery takes place in a relatively isolated “island” with very limited (particularly in terms of bandwidth) input from, and access to, resources on other “islands”. Resources in this case can be “soft” resources such as people, data sets, processes and applications, as well as “hard” resources such as instruments, processing power, data storage, or other specialized devices. The lack of access to these resources can cause redundant work, but more importantly, slower and less efficient progress on important challenges. Resource integration constitutes the “plumbing” that makes access to these diverse resource sets, including data resources, transparent to the application as well as to the end user. In effect we would like to make it seem from a user or an application viewpoint that all data is local data – removing boundaries that impede productivity, and providing access to compute and application resources that otherwise might be inaccessible to end users.

Semantic integration, in the case of data, is making sense of all of the data once it is directly accessible by applications and end users. This is a significant challenge. Data exists at numerous levels of abstraction, in many different formats, annotated in distinct styles, with different schemas.

Our overall approach is illustrated in Figure 1-1. We begin with a three layer conceptual view of the problem of gaining insight. At the bottom of the stack is resource integration. The basic problem is to provide secure, transparent access resources (compute, data, applications) to users and applications – both within the Virginia Center and external users. We believe that grid technology is the right approach to solving the access problem. Grid is all about virtualization and sharing resources across organizational boundaries. Grids eliminate boundaries between research groups, users of data and producers of data, and users and producers of cycles and applications. By managing data updates and data coherence we free researchers from worrying about the “freshness” of the data – and let them spend their time on their work – not chasing down files. By giving researchers access to a wider resource pool, e.g., larger and statistically more meaningful data sets, we improve the quality of the results. By giving researchers access to more compute capability we allow them to solve more difficult problems more quickly. And by giving them access to a wider range of applications we allow them to focus on their research, not chasing down the right tool or “re-inventing the wheel”.

The semantic layer focuses on the difficult problem of making the semantic match between different data schemas. This revolves around different data sets, DNA and protein data, cell data, organ data, clinical outcomes data, demographic data, and epidemiological data.

The knowledge discovery layer focuses on making sense of the data – of making new discoveries and generating new, testable, hypotheses. Algorithms and tools will to be developed to integrate, find patterns in, and draw inferences from the data.

The GBG is an open grid infrastructure to support the needs of life science researchers in academia, government, and industry. As indicated by its name, the GBG is intended to be global

in scope: accessible by users both in the US and abroad (subject to authorization). The GBG will provide: secure and transparent access to data resources in the grid (both public and protected resources) for authorized users using the latest data grid technology; access to selected compute resources in the grid for authorized users using the latest in compute grid technology; and access to a range of applications.

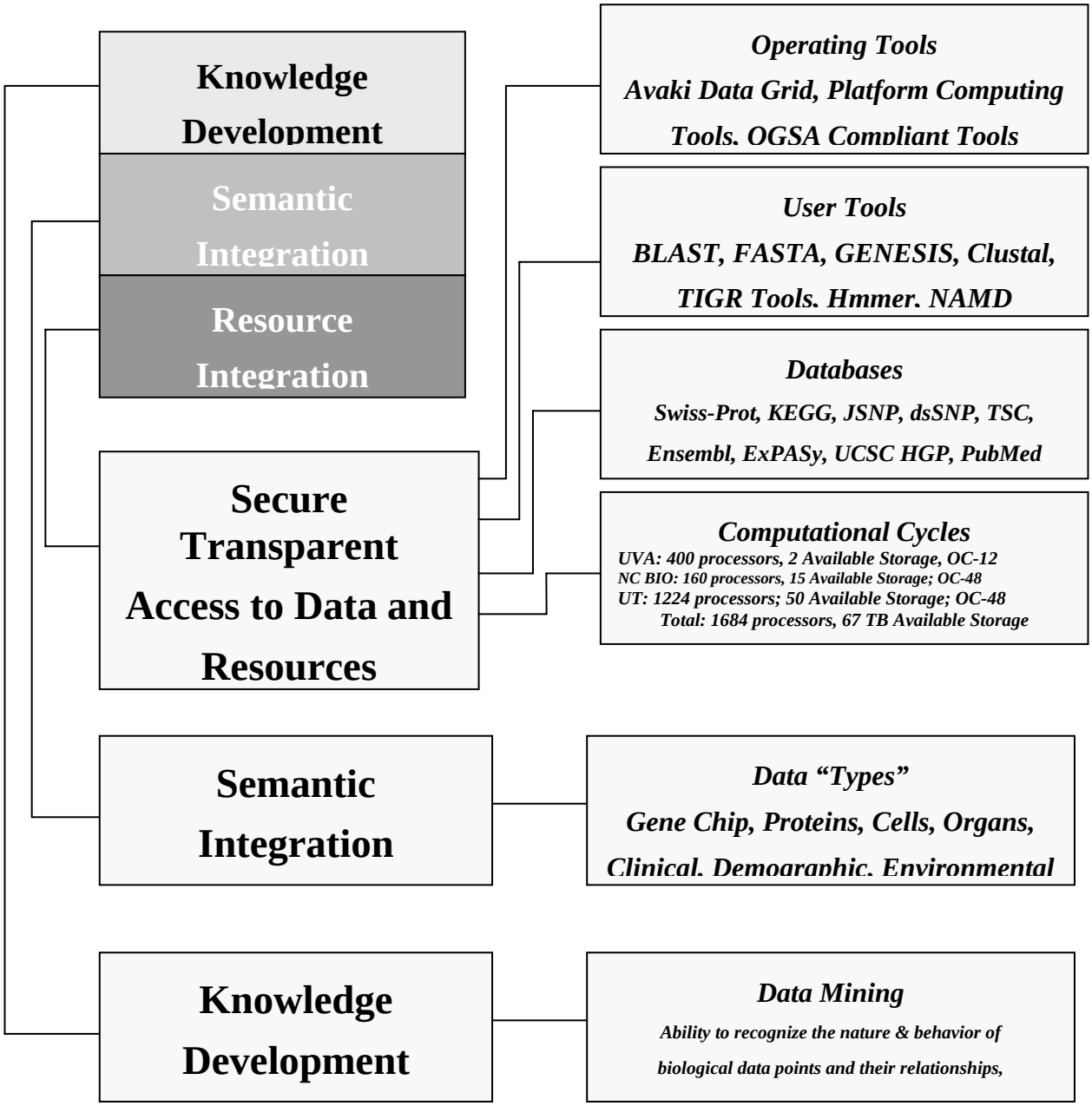


Figure 1-1 Knowledge development is built on insights gained by integrating multiple data types. The data used is from multiple groups, institutions, and administrative domains.

There are three main thrusts or "cores" to the Global Bio Grid:

1. Computer Science research in

a. Grids and Service Oriented Architectures (SOA's)

b. Security. One of the most important challenges in the creation of the GBG is to create a secure environment for users and data. The security infrastructure of the GBG must be designed to support a wide variety of user needs and anticipate the advances of the rapidly-developing field of security mechanisms, policies, and legislative requirements from around the world. To meet these diverse requirements, there are a number of basic questions that need to be answered, including:

   Authentication. How do users, machines, software components, etc., prove that they are who they claim to be? How does the user manage all of the potential "authentication points" in the Grid?

   Authorization. Who is allowed to access (read, write, delete, etc.) which information? How is this enforced and monitored?

   Integrity. Is the data protected from malicious or inadvertent attempts that might alter it?

   Confidentiality. Is the data protected from unauthorized attempts to read it?

   Privacy. Does a user have adequate mechanisms by which to control how much information about himself/herself can be shared with others?

c. Dependable and trustworthy computing. Building dependable life science grids is, at least in part, an engineering challenge. Developers and users of distributed systems are typically primarily concerned with availability and reliability. Informally, scientists or business users want to know that a given service is "available" to perform useful work, e.g., access, query and retrieve data from a protein database; fetch a document on the web; or view radiography images from a remote site. They also expect services to be "reliable," meaning that services will function correctly. This is not always the case. Why do grids fail and what can we do about it? Some of the primary cause for the brittleness of current grid systems are environmental:

   Grids mostly consist of prototypes and proof-of-concept projects — there are very few production grids in use today.

   Grids have been used mostly in research environments by early pioneers who were willing to tolerate unreliability.

   In commercial settings, product development is driven by the push to be first to market. Dependability is at best a secondary concern.

   These problems can be addressed through adoption of state-of-the-art engineering practices and industrial-strength solutions [2] [3]in designing and building the basic services provided by a grid system, so as to avoid mistakes and "bugs" in both the design and implementation of a grid infrastructure.

   d. Algorithms and tools for optimization, semantic integration, databases, and data mining.

2. Life sciences research in algorithms and tools, e.g., knowledge discovery in the context of SIDS and *diabetes mellitus*.
3. Grid Operations. Operational aspects cannot be ignored. The infrastructure for both the life sciences and computer sciences research must be brought on-line, maintained, and enhanced over time.

2 Grid Computing

There's more to Grids than just CPU cycle sharing. Grids enable resource virtualization and sharing. In the field of life sciences, data is *the* crucial resource. UVA is a pioneer in grid computing – and has been doing grid computing for over a decade [4-52].[citations] The following section will define a grid, explain how grids can help the life sciences, and addresses the various challenges and issues in deploying and using grids.

2.1 What is a Grid?

In 1994, we outlined our vision for wide-area distributed computing:

"For over thirty years science fiction writers have spun yarns featuring worldwide networks of interconnected computers that behave as a single entity. Until recently such science fiction fantasies have been just that. Technological changes are now occurring which may expand computational power in the same way that the invention of desk top calculators and personal computers did. In the near future, computationally demanding applications will no longer be executed primarily on supercomputers and single workstations using local data sources. Instead enterprise-wide systems, and someday nationwide systems, will be used that consist of workstations, vector supercomputers, and parallel supercomputers connected by local and wide area networks. Users will be presented the illusion of a single, very powerful computer, rather than a collection of disparate machines. The system will schedule application components on processors, manage data transfer, and provide communication and synchronization in such a manner as to dramatically improve application performance. Further, boundaries between computers will be invisible, as will the location of data and the failure of processors." [6]

The future is now. After almost a decade of research and development by the grid community, we see grids (then called metasystems [53]) being deployed around the world in both academic settings and, more tellingly, commercial use.

Grids are collections of interconnected resources harnessed together in order to satisfy various needs of users. The resources may be administered by different organizations and may be distributed, heterogeneous and fault-prone. The manner in which users can interact with these resources as well as the usage policies for those resources may vary widely. However the grid's infrastructure must manage these complexities so that users interact with resources as easily and smoothly as possible.

A popular definition of grids is that a grid system is a collection of distributed resources connected by a network. A grid system, also called a grid, gathers resources – desktop and hand-held hosts, devices with embedded processing resources such as digital cameras and phones or tera-scale supercomputers – and makes them accessible to users and applications. This reduces overhead and accelerates projects. A grid application can be defined as an application that operates in a grid environment or in other words is "on" a grid system. Grid system software (also called middleware) is software that facilitates writing grid applications and manages the underlying grid infrastructure. The resources in a grid typically share at least some of the following characteristics:

They are numerous.

They are owned and managed by different, potentially mutually-distrustful organizations and individuals.

They are unreliable.

They have different security requirements and policies.

They are heterogeneous. E.g., they have different CPU architectures, run different operating systems, and have different amounts of memory and disk.

They are connected by heterogeneous, multi-level networks.

They have different resource management policies.

They are likely to be geographically separated (perhaps by a campus, an enterprise, or a continent).

However these definitions are ambiguous. What constitutes a "resource" can be a difficult question, and the actions performed by a user on a resource can vary widely. For example, a traditional definition of a resource has been "machine," or more specifically "CPU cycles on a machine." The actions users perform on such a resource can be running a job, checking availability in terms of load, and so on. These definitions and actions are legitimate, but limiting. Today, resources can range from biotechnology applications, stock market databases, to wide-angle telescopes. The actions that might be performed on a grid system could be "run if license is available," "join with user profiles," and "procure data from specified sector." A grid can encompass a wide variety of such resources and user actions, so its infrastructure must be designed to seamlessly incorporate them without compromising the basic principles such as ease of use, security, autonomy, etc.

A grid enables users to share resources, applications and data across systems in order to facilitate collaboration, make applications execute faster and simplify access to the data. More concretely, this means being able to:

Find and share data. When users need access to data on other systems or networks, they should simply be able to access it like data on their own system. System boundaries that are not useful should be invisible to users who have been granted legitimate access to the information.

Find and share applications. The leading edge of development, engineering and research efforts consist of custom applications – permanent or experimental, new or legacy, public-domain or proprietary. Each application has its own requirements. Why should users be made to jump through hoops to get applications together with the data sets needed for analysis?

Share computing resources. The basic premise here is very fundamental: one group has computing cycles that it doesn't need, while colleagues in another group don't have enough cycles. The first group should be able to grant access to the other group to its own computing power without compromising the rest of the network.

Grid computing is in many ways a novel way to construct and deploy applications. It has received a significant amount of recent press attention and been heralded as the next wave in computing. However, under the guises of "peer-to-peer systems", "Grids" and "distributed systems," grid computing essentials (enablers) have been under development for decades. Grid computing requirements address the issues that frequently confront a developer trying to

construct applications for a grid. The novelty is that these requirements are addressed by the grid infrastructure, in order to reduce the burden on the application developer.

Clearly, the baseline requirement needed to develop grid applications is the ability to transmit bits from one machine to another. Everything else can be built from that. However, there are several challenges that frequently confront a developer constructing applications for a grid. These challenges lead us to a number of requirements which we believe any complete grid system must address. They are:

Security. Security basically implies that trust can be incorporated into the system by its users. This covers a gamut of issues, including authentication, data integrity, authorization (access control), and auditing. If grids are to be accepted by corporate and government IT departments, a wide range of security concerns must be addressed. Security mechanisms must be integral to applications and capable of supporting diverse policies. Furthermore, they must be incorporated from the start. Trying to patch security measures in as an afterthought (as some systems are attempting today) is a fundamentally flawed approach. We do not believe that a single security policy can be perfect for all users and organizations, so there must be mechanisms that allow users and resource owners to select policies that fit their particular security and performance needs as well as local administrative requirements. In medical computing, security has a particular set of externally imposed requirements (such as HIPAA) which may vary from country to country.

Global namespace. The lack of a global namespace for accessing data and resources is one of the most significant obstacles to wide-area distributed and parallel processing. The current multitude of disjoint namespaces greatly impedes developing applications that span sites. All grid objects must be able to access (subject to security constraints) any other grid object *transparently* without regard to location or replication.

Fault tolerance. Failure in large-scale grid systems is and will be a fact of life. Hosts, networks, disks and applications frequently fail, restart, disappear, and otherwise behave unexpectedly. Forcing the programmer to predict and handle all of these failures significantly increases the difficulty of writing reliable applications. Fault-tolerant computing is a major research issue in itself. Nonetheless the issue must be addressed or users will not entrust the grid.

Accommodating heterogeneity. A grid system must support interoperability between heterogeneous hardware and software platforms. Ideally, a running application should be able to migrate from platform to platform as necessary. At a bare minimum, components running on different platforms must be able to communicate transparently.

Binary management. The underlying system should be able to keep track of executables and libraries and know which ones are current, which ones are used with which persistent states, where they have been installed, and where upgrades should be installed. These tasks reduce the burden on the user of the grid.

Multi-language support. In 1970s, the joke was “I don’t know what language they’ll be using in the year 2000, but it’ll be called Fortran.” Fortran has lasted over 40 years, and C almost 30. Diverse languages will always be used and legacy applications will need support.

Scalability. There are over 400 million computers in the world today and over 100 million network-attached devices (including computers). Scalability is clearly a critical necessity. Any architecture relying on centralized resources is doomed to failure. A successful grid

architecture must strictly adhere to the distributed systems principle: the service demanded of any given component must be independent of the number of components in the system. In other words, the service load on any given component must not increase as the number of components increases.

Persistence. It is critical to have I/O and the ability to read and write persistent data in order to communicate between applications and to save data. However, the current files/file libraries paradigm should be supported, since it is familiar to programmers.

Extensibility. Grid systems must be flexible enough to satisfy current user demands and unanticipated future needs. Therefore, we feel that mechanism and policy must be realized by replaceable and extensible components, including (and especially) core system components. This model facilitates developments of improved implementations that provide value-added services or site-specific policies while enabling the system to adapt over time to a changing hardware and user environment.

Site autonomy. Grid systems will be composed of resources owned by many organizations, each of which will want to retain control over its own resources. For each resource, its owner must be able to limit or deny use by particular users, specify when it can be used, and so forth. Sites must also be able to choose or rewrite an implementation of each grid component as best suits their needs. A site may trust the security mechanisms of one particular implementation over those of another and it should be able to freely choose that implementation.

Complexity management. Finally, but importantly, complexity management is one of the biggest challenges in large-scale grid systems. In the absence of system support, the application programmer is faced with a confusing array of decisions. Complexity exists in multiple dimensions: heterogeneity in policies for resource usage and security, a range of different failure modes and different availability requirements, disjoint namespaces and identity spaces, and the sheer number of components. For example, professionals who are not IT experts should not have to remember the details of five or six different file systems and directory hierarchies (not to mention multiple user names and passwords) in order to access files they use on a regular basis. Thus, providing the programmer and system administrator with clean abstractions is critical to reducing the cognitive burden.

These requirements are high-level and also are independent of implementation. But they all must be addressed by the grid infrastructure in order to reduce the burden on the application developer. If the system does not address these issues with an architecture based on well-thought principles, the programmer will spend valuable time on basic grid functions, needlessly increasing development time and costs.

3 Problems in Life Science Research

3.1 Grids are the hammer, what is the nail?

Life science researchers face a myriad of problems in effectively and efficiently carrying out their research mission. Significant human effort is spent by research organizations and sometimes by the researchers themselves in manually managing their data; setting up and maintaining their local computational resources; and deploying, managing, and fine-tuning the performance of the applications required to perform their research. The effort required is exacerbated if the resources or participants in a research effort are geographically or organizationally separate. Day-to-day management becomes more complicated, more time consuming, and more prone to error. All of these factors slow down the discovery process and raise costs as human, data, computational and other equipment are not utilized as efficiently as they might be.

3.2 Access and Management of Data

Researchers in the biological and behavioral sciences require access to a broad array of existing data resources. Also since these are experimental sciences they also generate large amounts of new data through their efforts. As computational and data storage capacities continue to increase, researchers are producing larger volumes and types of data. Also researchers are creating ever-more complex models of biological processes and require access to and integration of data of different types from multiple sources. At the same time, researchers are using more sophisticated data mining and data integration techniques to detect significant patterns across biological data, while the speed and volume at which data are consumed is also rapidly increasing. The trend in biological sciences is towards a spiraling increase of volume and type of data being produced and digested.

At the same time, the data required by these new research techniques are physically stored in different locations by different organizations. Simply finding it is a non trivial problem. Data sets, even of similar semantic meaning, are stored in different formats with different levels of accuracy or reliability and different storage technologies. Using data that is copied or cached in multiple places raises problems with consistency and coherence, especially when running multiple distributed jobs against the data. Each data set may also have access or usage restrictions, whether because of proprietary ownership and licenses, patient privacy rights (such as HIPPA), security reasons, or some other factor.

All of these obstacles add up to make data management and access a difficult task that consumes a significant amount of research team effort. Currently, there are a variety of ways to acquire data sets, usually involving some level of manual intervention. The proper data sets need to be identified and found. This is still an ad-hoc process with only limited support for searching. The source organization must then grant access to the data. Except for truly public data sources, negotiating access to data is currently done almost entirely by direct bilateral discussions. Then, a method for accessing data must be chosen. This might involve accessing the data directly from the source or transferring a copy of the data to a site local to the research effort. In either case a human must make the transfer and manage the copy. For dynamic data sources, the copy may rapidly go out of date. For a research effort that spans multiple sites or computing resources,

multiple copies are required, each needing to be kept up to date. For large data sets, it may be a significant strain to copy and store data locally at all. Accessing a large data set from across the country each time it is needed can also create a considerable performance problem. Once the data is finally available, the content may need to be automatically or manually manipulated in some fashion: transformed to a new format, cleaned up to recalibrate or eliminate errors, randomized to provide privacy for patient records, etc.

How can grid computing help this situation? Grid software can be thought of as lower- and middle-level software with respect to data. At the lowest level, grid software is designed to handle storing the bits, keeping track of where they reside, and providing basic access and security to them. At a middle level, grid software may add functionality such as caching and replication to improve performance and availability. Current grid software does not provide good tools and abstractions to describe and understand the semantics of data, carry out data mining operations, and integrate data across multiple data sets. These shortcomings are not limited to grid systems, since most other data management systems do not handle these issues well either.

3.3 Computational Resources

With the increased reliance on computational biology and data and statistical mining techniques, biological and behavioral science research efforts require ever more computational resources. A common paradigm in bioinformatics involves scanning a database and applying some function to each entry to generate a score or some other value. More formally, this procedure can be described as the set of results:

$$f(x,t) \mid x \quad X$$

or, alternatively:

For all x an element of X, compute f(x, t), where f(x, t) is some function such as BLAST, a docking code, or some other application.

In certain cases the function needs to be evaluated for the cross product of all values in one database across all of the values in another database:

$$f(x,y) \mid x \quad X, y \quad Y$$

or, alternatively :

For all x an element of X, For all y an element of Y, compute f(x, y), where f(x, y) is once again some application.

Typical times to evaluate the function for a single pair of entries range from a few seconds to several hours of CPU time and the number of evaluations can easily take up thousands of CPU hours. To make such tasks tractable, the task is parallelized, breaking the total number of evaluations into a single, or a small group, of evaluations per compute job and the jobs are run across multiple processors. This has become common practice within labs using clusters.

Another common paradigm involves larger numbers of ensemble computations with either stochastic behavior or slightly different parameters. Either way, the result is a large number of essentially independent, non-communicating computationally complex jobs which in aggregate require more power than one machine can provide. When spread across a large collection of machine, however, they are manageable.

The need to accommodate increasing demands for computational power causes research efforts to spend significant effort on managing the required computational resources. As computational biology becomes more ubiquitous, it is likely that a single research effort will require more computational power than the local institution can provide. Thus, it is reasonable to expect that local research efforts will need access to resources that are physically separated from the research team and controlled by a different organization.

Currently, gaining access to computational resources is done either by purchasing the required resources with the project budget or individually negotiating for use the time of existing resources. Buying new computational resources for a project can potentially prove an inefficient use of both time and money. New resources require expertise and effort to configure and may end up under-utilized or useless outside the project's specialized demands. For borrowed/rented resources, there is often a significant effort up front to provide the research team with the proper accounts and infrastructure software (such as queuing systems or databases) and train research team members on unfamiliar infrastructure tools.

Grids address these challenges by bringing a potentially much larger resource base to the problem: rather than spreading a set of jobs out over a small cluster or group of workstations, they can be distributed throughout the grid [32, 35] [35] [54] [46].

3.4 Applications

Researchers employ various computational applications to further their studies, including simulations of base biological processes, protein and genome comparisons, statistical data mining and correlation, and many others. Some of these applications are home-grown while others are standard vendor products or freeware packages, such as BLAST, FASTA, NAMD, and Amber. Research teams have to arrange for their applications to be deployed wherever they will be used, make sure that different machines have the proper version of the software, and check that the software is configured in a compatible manner to other machines used in the research effort. For applications being developed and tested by the research team, the redeployment of newer application versions will be frequent and can be a substantial drain on time. As research increasingly spans the computational resources of many smaller desktops or clusters or multiple sites and organizations, the problem of deploying applications and maintaining their consistency becomes a greater strain on system administrators associated with each project.

In a grid one does not deploy common applications many times – they are installed once and the underlying grid middleware is responsible for provisioning them to various resources, managing licenses, etc.

3.5 Dependability

Dependability—the ability to justifiably rely on the grid infrastructure for day-to-day operations—is a paramount property for a usable grid. Incorporating strategies and techniques for achieving dependability in grid applications is a known, difficult problem. It must nevertheless be addressed, or businesses and researchers will not adopt grids as a technological foundation on which to entrust their data and applications. Grid components such as hosts, networks, disk and applications frequently fail, restart, disappear and otherwise behave unexpectedly. Furthermore, the evolving fault and threat models that grid designers face exacerbate the task of achieving dependability; not only will grids be disrupted by attacks that target the Internet community as a

whole, e.g., viruses and worms, but also they will also be subject to attacks that specifically target life-science grids, as the technology matures and becomes widely employed in production systems. Requiring programmers or life science domain experts to anticipate and handle all these possible faults and threats would significantly increase the difficulty of writing dependable applications. Instead, a grid system should automatically and proactively manage and cope with faults as an essential service to its user applications.

4 The Global Bio Grid

Given the above definition of grids, and our experience in the life sciences both in academic projects such as the NPACI grid activities, and our industrial experiences with biotech's and pharmaceuticals, we believe that grids will have a tremendous impact on research in the life sciences.

To achieve that impact, several institutions — the University of Virginia (UVa), North Carolina BioGrid (NCBio), the University of Texas at Austin (UT), The Center for Advanced Genomics (TCAG), and Texas Tech University (TTU) — have agreed to form the nucleus of a hardware and software infrastructure called the Global Bio Grid (GBG) to further biological, behavioral, and life sciences research. Creation of this grid has already begun at several institutions and several others are committed to their participation in the GBG if this proposal is funded. Furthermore, several other national and international partners have been identified for future expansion of the GBG and agreements in principle have been reached with them.

We envision the GBG as a dynamic system, where users, projects, data sets, computational and other resources, applications, and organizations are added and removed as the capabilities of the system are enhanced, as resources are purchased or decommissioned, and as research needs evolve.

4.1 Global Bio Grid Principles and Design Philosophy

The Global Bio Grid uses the design philosophy and principles exposed by designers of Legion, one of the first grid systems [20] [55] [29] [27] [32] [35] [37] [56] [9] [57] [7] [33] [31] [34] [18]. Many of them have been incorporated in the Global Grid Forum's Open Grid Services Architecture. These principles are:

Provide a single-system view. With today's operating systems we can maintain the illusion that our local area network is a single computing resource. But once we move beyond the local network or cluster to a geographically-dispersed group of sites, perhaps consisting of several different types of platforms, the illusion breaks down. Researchers, engineers and product development specialists (most of whom do not want to be experts in computer technology) are forced to request access through the appropriate gatekeepers, manage multiple passwords, remember multiple protocols for interaction, keep track of where everything is located, and be aware of specific platform-dependent limitations (e.g., this file is too big to copy or to transfer to that system; that application runs only on a certain type of computer, etc.). Re-creating the illusion of single computing resource for heterogeneous, distributed resources reduces the complexity of the overall system and provides a single namespace.

Provide transparency as a means of hiding detail. Grid systems should support the traditional distributed system transparencies: access, location, heterogeneity, failure, migration, replication, scaling, concurrency and behavior [27]. For example, users and programmers should not have to know an object's location in order to use it (access, location and migration transparency), nor should they need to know that a component across the country failed. They want the system to recover automatically and complete the desired task (failure transparency). This behavior is the traditional way to mask details of the underlying system. Transparency also addresses fault-tolerance and complexity.

Provide flexible semantics. Our overall objective is a grid architecture that is suitable to as many users and purposes as possible. A rigid system design in which the policies are limited, trade-off decisions are pre-selected, or all semantics are pre-determined and hard-coded would not achieve this goal. Indeed, if we dictate a single system-wide solution to almost any of the technical objectives outlined above, we would preclude large classes of potential users and uses. Therefore, we will allow users and programmers as much flexibility as possible in their applications' semantics, resisting the temptation to dictate solutions. Whenever possible, users can select both the *kind* and the *level* of functionality and choose their own trade-offs between function and cost. This philosophy is manifested in the system architecture. The object model specifies the functionality but not the implementation of the system's core objects; the core system therefore consists of extensible, replaceable components. We will provide default implementations of the core objects, although users will not be obligated to use them. Instead, we encourage users to select or construct object implementations that answer their specific needs.

Reduce user effort. In general, there are four classes of grid users who are trying to accomplish some mission with the available resources: end-users of applications, applications developers, system administrators and managers. We believe that users want to focus on their jobs (i.e., their applications), and not on the underlying grid plumbing and infrastructure. Thus, for example, to run an application in Legion a user could type

```
legion_run my_application my_data
```

at the command shell. The grid should then take care of messy details such as finding an appropriate host on which to execute the application and moving data and executables around. Of course, the user may need to specify or override certain behaviors, perhaps specify an OS on which to run the job, or name a specific machine or set of machines, or even replace the default scheduler. This level of control should also be supported.

Reduce "activation energy." One of the typical problems in technology adoption is getting users to actually use it. If it is too difficult to shift to a new technology, users will tend to avoid trying it out unless their need is immediate and extremely compelling. This is not a problem unique to grids: it is human nature. Therefore, one of our most important goals is to make it easy to use the technology. Using an analogy from chemistry, we kept the activation energy of adoption as low as possible. Thus, users can easily and readily realize the benefit of using grids – get the reaction going – creating a self-sustaining spread of grid usage throughout the organization. This principle manifests itself in features such as "no recompilation" for applications port to a grid and support for mapping a grid to a local operating system's file system. Another variant of this concept is the motto "no play, no pay." The basic idea is that if you do not need encrypted data streams, fault resilient files or strong access control, you should not have to pay the overhead of using it.

Do no harm. To protect their objects and resources, grid users and sites will require grid software to run with the lowest possible privileges.

Do not change host operating system. Organizations will not permit their machines to be used if their operating systems must be replaced. Our experience with Mentat [58] indicates, though, that building a grid on top of host operating systems is a viable approach. However, it must be able to run as a user level process, and not require root access.

Overall, the application of these design principles *at every level* provides a unique, consistent, and extensible framework upon which to create grid applications.

4.2 Technical Overview of GBG

Since the GBG is part production system and part research system, it is impossible and frankly undesirable, to characterize all of the important features the system will have in five or even three years. Many aspects of the system, such as policy negotiation and confidentiality enforcement, will be influenced by feedback from the needs of biological researchers as they deploy into the new grid environment and develop schemes to match the new possibilities open to them. However, our long experience with distributed and grid computing systems dictates that the system must have certain features and follow certain guiding principles in order to succeed in delivering its promise to users (Section 4.1). These features and principles are summarized below.

4.2.1 Single Global Namespace

One of the key problems in managing projects and resources and in writing application software that spans distributed sites is that there is no common way to identify many of important entities, such as users, applications, computers, data sets, files and directories, databases, policies, accounting records, etc. The vast majority of institutions maintain isolated local naming schemes for certain important entities, such as the local user account names, local file systems (which may or may not be locally shared), and local machine names and addresses. Since the naming schemes at each site are independent, there is no way to identify whether a name used at one institution identifies the same entity at another institution. For example, there is no way to determine if the person Joe Smith has an account at two different institutions, and if so what the local names are. The situation is similar for a common application, for common data sets and databases.

The lack of a common and global namespace for important entities is a significant barrier to managing a distributed environment, since it prohibits the creation of software to automate many processes and inhibits the development of applications that can be easily moved from one machine or organization to another. Therefore, one of our fundamental principles is that GBG will provide a single global namespace for certain key entities. At the minimum, this will initially include users, directories, files, access to database views, several common applications, and the compute resources incorporated into the GBG.

4.2.2 Federated Sites with Common Access

The GBG will be organized as a federation of cooperating distributed sites, departments, and individual research labs. The decision to organize the system as a federation reflects the practical situation of the real world: equipment available for GBG is physically housed at different sites; the data sets are physically located at different sites; personnel are already in place to maintain these resources at certain sites; researchers are in geographically separate areas from which they must be able to access the resources of the GBG; and it may be politically, technically, or contractually difficult or even impossible to move or copy many of the resources. In addition, a centralized approach is undesirable from a technical standpoint, since a centralized structure will ultimately not scale as the system grows.

Grid software is the key to making such a complex system manageable for administrators and reasonable to use for researchers. The grid software will be the glue that ties together the users, data, and computational and other resources at the various sites.

Specifically the GBG will provide access to the following types of resources:

Data sources. The GBG will incorporate relevant public databases, such as the Protein Database (PDB), PubMed, and Swis-Prot. By mapping public databases into the GBG, each user can access them without having to make local copies or otherwise expend effort to procure or maintain them. Each database can be administered and maintained by one site — the one which is likely already doing so. The grid software can automatically manage local caching and cache consistency to improve performance for researchers.

Relevant and available commercial databases will also be incorporated into the GBG. The goals and issues are the same as for public databases, except that the grid software will have to manage access to the data differently. Only select users may be able to access the data or restrictions on where the data can be moved may be imposed by the licensing agreement. The grid software will need to deal with such restrictions.

Finally, the GBG will incorporate relevant in-house databases, data sets, annotations and the like. The GBG will provide mechanisms to easily distribute and provide access to locally produced data, saving the data owner the hassles of developing and maintaining a dissemination mechanism himself, while allowing him to retain the proper level of control if necessary.

Application suites. Many biological research projects require running one or more biological, statistical, or data mining application to drive discovery. Just like data, applications may be publicly available freeware solutions, vendor licensed software, or home-grown and possibly proprietary programs, and the grid will need to handle access and license enforcement accordingly. Legacy applications will be written in various languages, may run on one or more operating systems, and may be stand-alone programs, parallel programs, or entire processes or work flows. Again, the grid environment must be able to run the types of legacy applications that researchers need to use. Applications will may need very little or very large amounts of compute time, and may need to be run a small number of times or a very large number of times, as is the case for Monte Carlo and other probability based simulations, studies involving non linear systems and parameter space studies.

The tools provided to grid users must make deploying and running such applications, work flows or suites of applications easy and efficient. Since the computational environment of the GBG is distributed and federated in nature it is important that the grid software help manage the physical disbursal of the appropriate versions of programs to the various computers in the system. This will help to alleviate the inefficiencies and potential errors that researchers and administrators face in trying to use distributed computing resources. Similarly, the grid must provide useful tools that help end users run applications, work flows, or suites of applications in an efficient manner that best uses the researcher's time and the computing resource's capacity.

The GBG will provide tools for end users or administrators to deploy applications into the GBG when necessary. This approach will not only allow power users to control rolling out their applications, which is especially important if the applications are to be frequently updated, but will also provide a quick way to disseminate new programs or program versions at least within the GBG community. Certain relevant applications will also be deployed directly into the

GBG for public use by participating GBG sites. For certain heavily used or important applications the GBG may deploy portals for easing the use of these popular applications.

Heterogeneous Compute Resources. The complement to deploying applications and data is deploying and providing controlled access to the necessary computing environments on which to run applications. The GBG must pull together an array of computing resources from its various member institutions which provides the computation backbone. These resources will be geographically separate, across site and administrative domain boundaries, heterogeneous in machine type, capabilities, and operating system. Computers will span the gamut from PCs and workstations to clusters and supercomputers.

Since the resources will be a federated collection, the resources will potentially have different access and usage policies. This is simply a fact of life when disparate organizations are involved, but by dealing with it explicitly, it vastly increases the GBG's ability to attract and incorporate new partner sites into the grid.

4.2.3 Virtual Organizations.

A virtual organization is a way to use grid technology and its management of distributed resources and access control to create the illusion for a group of users that they are part of one organization. As an example, a project team could set up a virtual organization within the GBG that contains directories for all of the data, applications and computers to be used in the project, and set access control and security and resource usage policies so that to users they are part of a self contained world. Users of such a virtual organization have the experience of working within a local environment, including the ability to share data, applications, computation resources and policies on the usage thereof, even though the actual resources are physically separated. The grid infrastructure takes care of managing the stated access control policies of the organization and masking the fact that resources are distributed. Since the GBG will be a platform shared among various projects, and aims to scale to a national biological computing and data resource, providing virtual organizations to manage and keep separate the policies and private resources of various separate partnerships is essential to its ultimate usefulness.

4.2.4 Fine-grain Modular Security with Work grain modular security with work towards HIPPA and CFR 21 part 11

Security is clearly one of the hallmark problems encountered when resources are distributed across sites and administrative boundaries. The GBG must handle the usual security issues: authentication of identity, access control, data privacy and data integrity in order to be a workable solution for its users and resource providers.

Since the GBG is designed to support work specifically in the biological and behavioral sciences, it is certain that some data that researchers collect or require access to will be subject to HIPPA and CFR 21 part 11 rules. It is important that over time the GBG can incorporate such data, so the GBG must therefore be able to support enforcement of relevant HIPAA and CFR 21 rules. To comply with CFR 21 and HIPAA rules, we will initially only deploy data that has been de-indentified.

4.2.5 Build GBG Using Off-Off the- Shelf Technology

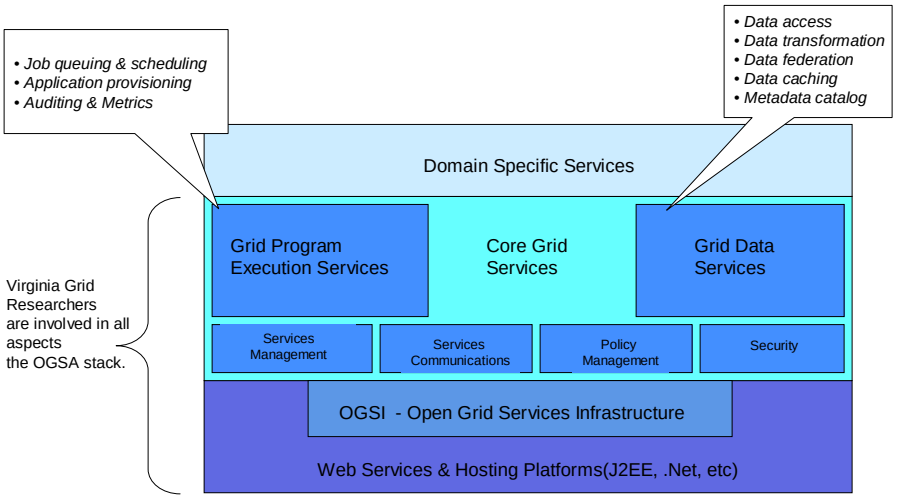
Our experience in developing Legion taught us many valuable lessons. Among them is that the development of grid software is an arduous and time-consuming task. Many of the basic grid issues have been solved reasonably well and several off-the-shelf commercial and freeware software have emerged that incorporate the functionality we need. Partial solutions can be obtained from commercial vendors such as Avaki, Platform Computing [59], Sun [60], and United Devices and from academic efforts such as Legion [20] [55] [29] [27] [32] [35] [37] [8] [9] [13] [7] [33] [31] [34] [18], Nimrod [61], Condor[62], or Globus [63]. None of these solutions provides all of the necessary functionality, even for the initial phase of the project. But each has features that may be woven into a cohesive and robust initial platform which can rapidly start delivering significant benefits to the research of the other core efforts.

4.2.6 Build GBG on Evolving Standards

In constructing and deploying the GBG we are committed to deploying solutions that follow the grid and bioinformatics standards being developed in the Global Grid Forum (GGF) and I3C. Deploying system components that adhere to standards is the only way that we can ensure that the GBG will work within the widest possible set of future sites, will integrate with the widest possible set of outside software components, and will minimize the effort required to develop new functionality. The goal of the GBG is to pave the way towards biological grid systems that provide vast collections of resources to the researchers of the entire country. In order to approach this goal, the GBG must minimize solutions that will be incompatible with other technology or which will be rejected by the community.

In the grid community, the major standards bodies are the GGF and the various Web Services standards groups (W3C, DMTF, and OASIS). Several of the members of the GBG team have been actively engaged in the GGF since its inception. Particular standards efforts of note include the WSRF and the work of several groups under the umbrella of the Open Grid Service Architecture (OGSA) initiative as well as the work in security, WS-Security, and policy negotiation, WS-Agreement within the Web Services community.

The OGSA basic architecture is shown below in Figure 1-2. At the bottom of the stack are the Web Services standards and WSRF on which the rest of the architecture is built. WSRF provides basic grid service naming, discovery, lifetime management, and communications capabilities.



Source: A Visual Tour of Open Grid Services Architecture. IBM, 8/03

Figure 1-2 The OGSA services stack as currently envisioned. The Virginia Grid team has worked on software at all layers in this stack, and is actively involved in the OGSA working group, the Grid Program Execution Services group, and the Data services group.

Higher level, "core services" include security services, policy management services, manageability services, data services, service (e.g., application) provisioning services, scheduling and brokering services, and so on. The architecture and philosophy of the OGSA are very reminiscent of the Legion architecture.

The pace of standards work in both groups very rapid at the moment and the GBG will participate in the relevant efforts and will participate in driving important areas if necessary. We believe that OGSA, in conjunction with relevant Web services standards, represents the best choice for the life sciences community.

We are not alone in choosing to base the GBG on OGSA. All of the major vendors, IBM, HP, Sun, Fujitsu, and NEC have all embraced OGSA and are actively participating in the OGSA working group that is defining the standards. Further, IBM is basing its "on-demand computing" and "autonomic" computing efforts on OGSA. Thus we are confident that in the future there will be a large number of OGSA compliant hardware and service offerings.

In addition to the grid world, there are standards for within the biology community for data naming and search and for data formats. One recent standard that we intend to monitor closely is the Life Science Identifier (LSID) standard [64] finalized by the Interoperable Informatics Infrastructure Consortium (I3C). This standard has been proposed to provide a common naming scheme and name binding process for all types of biological and life sciences data. The GBG architects will monitor the progress of LSID based solutions and are committed to deploying compatible solutions as they become available.

5 GBG Deployment

The GBG will be deployed in three broad phases over the next five years although we expect that the GBG will be evolving in smaller ways almost continuously over its lifetime to incorporate the best technology available as it develops. Since it will ultimately be a production environment for the biological scientists that use it, GBG administrators will plan upgrades and new software roll outs carefully, with an emphasis on keeping user disruption to a minimum. The first phase, which is already partially under way, is designed to provide researchers with state-of-the-art tools as soon as possible in order to rapidly improve their access to and management of data and computational resources and to increase their overall ability to get related work done. The two later phases will be to deploy the new state of the art software, including software both developed as part of this proposal as well as developed by vendors and others in the grid and biological sciences community. In addition later phases will add new data sources and participating organizations into the expanding GBG community.

5.1 Phase I

Phase I focuses on getting technology into the hands of researchers and providing access to data and computational resources otherwise unavailable to them or difficult to maintain by them. Phase I will begin with the charter member sites of the GBG, as shown in Figure 1-3, which are UVa, NC BioGrid, UT, TCAG, ETH Zurich, and TTU.

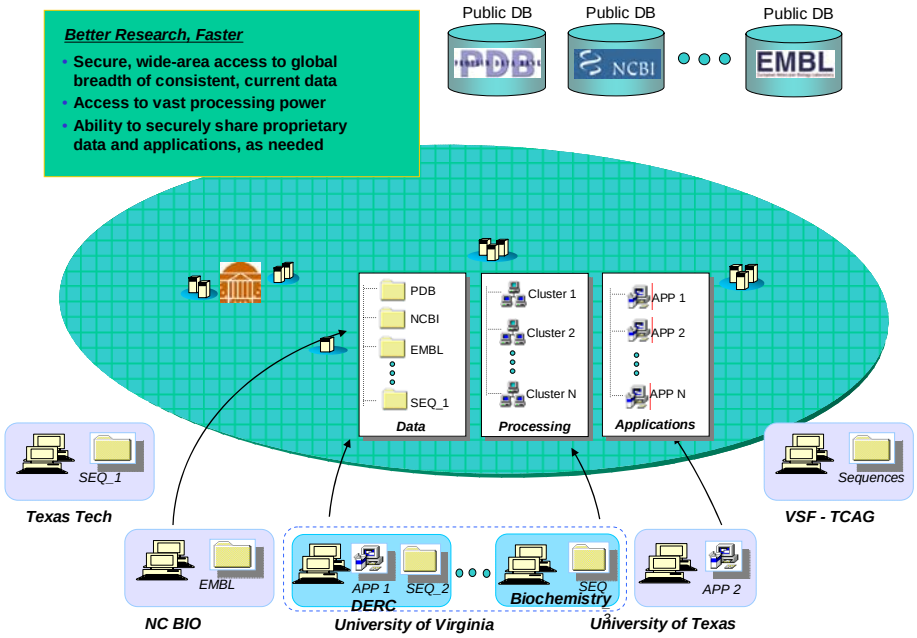


Figure 1-3 The GBG initial sites are shown. As we move from Phase I to Phase II additional sites around the world will be added. Preliminary discussions have already been held with the e-Science centers in Cambridge and Oxford, UK.

5.1.1 Phase I Goals

To accomplish this mission, we have set eight goals for Phase I.

Goal #1: Construct GBG rapidly. Getting the GBG operational, even if in a somewhat limited way, is crucial to supporting the research efforts of both the biological and bioinformatics researchers as well as computer scientists. Until the system is operational, researchers will not be able to truly learn to use it or be able to realize any of the benefits it offers. To accomplish this quick deployment, Phase I of the GBG will employ primarily currently available off-the-shelf software and existing computational hardware.

Goal #2: Deploy real data sets. Several public databases are readily available and widely used by the biological research community. By deploying these databases into the GBG during Phase I we can quickly gain real users from the research community and begin to collect early feedback from them. The users themselves will benefit from easier access to the data, possibly improved access performance and decreased maintenance requirements at each user site**.** We will also deploy data sets stored locally at member institutions as needed, .e.g., PDB or PubMed. These data sets will then be available to all collaborating partners, regardless of their location, subject to access control policies.

Goal #3: Deploy a strong base of computational hardware into the GBG.To fulfill the computational needs of the biological researchers as well as to provide a platform for the computer science researchers, the GBG will incorporate a substantial set of computational assets. In particular, during Phase I the GBG will initially incorporate machines from 4 academic departments within Uva, including the Centurion 300+ processor cluster and over 200 PCs in the public computing laboratories and classrooms, 160 processors from the NCBio, and over 1200 processors from UT's Texas Advanced Computing Center. These machines collectively represent a capability that no single institution could provide. As Phase I evolves, additional machines will be incorporated into the GBG if they become available at the existing sites or additional sites join the GBG.

Goal #4: Deploy relevant applications into the GBG. The GBG will deploy several widely used, freely available applications, including BLAST, FASTA, NAMD, and Genesis. As with goal #2, deploying common applications will provide a carrot to recruit early phase users from the biological research community and will provide real world application instances with which researchers and GBG administrators can gain experience.

Goal #5: Provide easy access to data for researchers. It is not sufficient to simply have data available within the GBG, we must also provide easy mechanisms for researchers to access and update their existing data and to deploy new data sets. The GBG will deploy the Avaki Data Grid (ADG) to the necessary GBG sites. ADG software provides functionality to manage distributed directories and files and for accessing relational databases via canned queries. It also includes client-side software that accesses data as though it is local to the user's machine, providing access to data via NFS and CIFS.

Goal #6: Provide software to run applications on the grid. Data access is only one piece of the puzzle. Researchers also need tools to make the job of harnessing the GBG's computational power relatively easy and as reliable as possible for the main types of bioinformatics applications, including parallel programs, parameter space studies, and large scale comparisons. Platform Computing's MultiCluster and related other components of their LSF suite will provide

the application management and monitoring capability for GBG Phase I. MultiCluster is a reliable and proven product for running applications across resources that span networks and sites.

Goal #7: Create a stable platform to support work on research efforts. The biologists and bioinformaticians are the driving force behind the construction of the GBG. However, the CS researchers 1 also need a platform on which to build, debug, and test software. The Phase I deployment is designed to be more than adequate to support early prototype work. As prototypes develop and solidify, they will be deployed into the GBG.

Goal #8: Initiate early feedback from researchers to drive future directions. Part of the process of using technology as new as grid software and working towards the long-term goal of a nation-wide bioinformatics grid system is to find out what works and what doesn't and to determine what users really need. By deploying a usable and state-of-the-art grid system as early as possible, the GBG will jump-start the feedback process from a real user base. This is critical for making the most possible progress in the shortest timeframe, since determining user requirements happens most easily when users actually see the possibilities and use the latest technology. The feedback gained in Phase I will drive improvements to the deployed GBG grid software as well as to drive research.

5.1.2 Phase I Grid Software

To quickly deploy the GBG and to provide the best and most stable possible platform, we have purposely chosen to use the best existing, proven grid solutions to construct the GBG's software infrastructure. In the sections below, we briefly introduce the major grid software components.

Platform LSF MultiCluster™

Platform LSF is software for managing and accelerating batch workload processing for compute-and data-intensive applications. Platform LSF enables users to intelligently schedule and guarantee completion of batch workloads across a distributed, virtualized IT environment. Platform LSF fully utilizes all IT resources regardless of operating system, including desktops, servers and mainframes to ensure policy-driven, prioritized service levels for always-on access to resources. Platform LSF is based on the production-proven, open, grid-enabling, Virtual Execution Machine (VEM)™ architecture that sets the benchmark for performance and scalability across heterogeneous environments. With more than 1,600 customers and the industry's most extensive library of third-party application integrations, Platform LSF is the leading commercial solution for production-quality workload management.

The Platform LSF MultiCluster extends the basic LSF suite to enable control of resources which span multiple clusters and geographical locations. With Platform LSF MultiCluster, local ownership and control is maintained ensuring priority access to any local cluster while providing global access across an enterprise grid. The features of Platform LSF will enable GBG users to complete workload processing faster with increased computing power, enhancing productivity and speeding time to results.

Avaki Data Grid™

Avaki Data Grid 5.0 (ADG) provides a single stream-lined mechanism for making various types of data available to developers and users in any location on demand. ADG

supports relational database data (results of SQL statements and stored procedures); URL-based data sources such as servlets, CGI scripts or web services, XML documents, XSLT style sheets and files in direct- or network-attached or SAN storage. Database results can be automatically formatted into XML, and XML data can be transformed via stored XSLT style sheets into any structure or format, providing a powerful range of conversion capabilities.

When using ADG 5.0, users and applications access data as if it were local. One unified data catalog provides one view for accessing all available data. Distributed caches ensure performance across a wide area. Users access data transparently through standard file system protocols, while applications access data using standard methods: ODBC, JDBC, Web Services/SOAP, file read, and JSP/Tag library.

In order to make the selection of ADG as our grid data engine, Avaki has agreed to license the necessary software for all of the GBG academic sites at no cost. With this arrangement, the GBG will acquire the proven best-of-class data grid software to handle a good portion of grid software needs for no cost.

Contingency Plans

In the event that we cannot acquire the commercial software for whatever reason (budget, availability, etc.), there are reasonable solutions that we can obtain for free to which we can fall back. UVa has a license agreement to use the Legion 1.8 grid system for the deployment on projects like the GBG. Legion 1.8 provides remote data management and access, application staging, and a wide range of support for running legacy and MPI programs in a distributed environment. Several members of the team members are responsible for the development of the Legion system and are confident that it will serve as a reasonable replacement for either the Platform MultiCluster or ADG components or both if necessary.

5.2 Phase I Status

As of September, 2004 the Global Bio Grid is beginning to come on-line. Specifically the Data Grid is up and installed at five sites: the University of Virginia, the North Carolina BioGrid, the University of Texas, Austin, Texas Tech University, and ETH Zurich. Early performance results are good [65]. Below results from a local area and wide area test are shown. We expect to begin compute grid integration in the Fall of 2004.

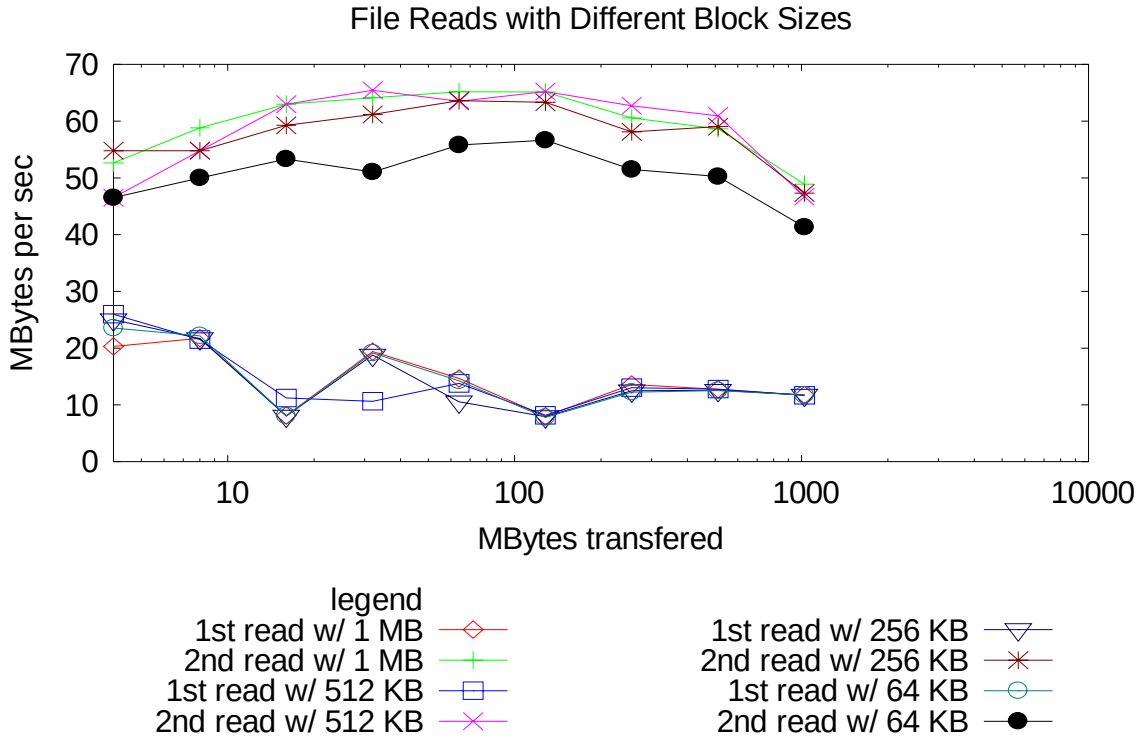


Figure 1-4 Performance results with a single reader when the source data, Avaki DGAS, and client are all in the same, Gigabit Ethernet connected, environment. The lower numbers are for the first read – before Avaki caches the data. The higher numbrs are once the data has been cached by Avaki. Note: All non-Avaki (e.g., File System) caches are cleared between runs.

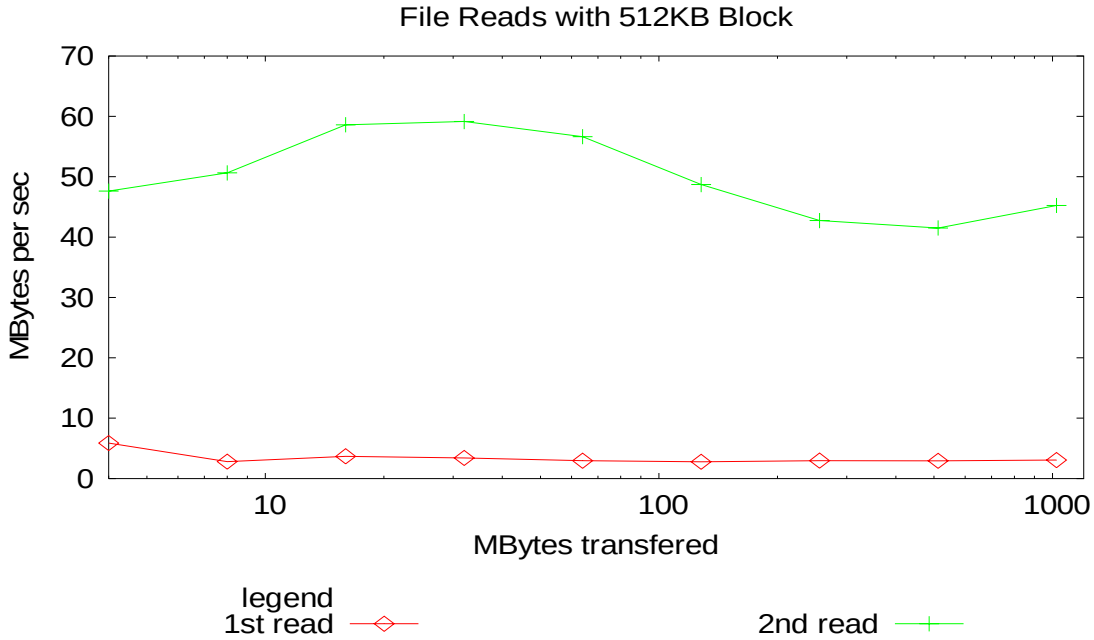


Figure 1-5 Wide-area test similar to above between UVA and TTU. Note effect of caching.

6 Technology Integration Plan

The Global Bio Grid will not suddenly appear in full form. Like any complex technology-based project it will come into being in phases as described above. Further, unknown events, changes in requirements, and technological changes can change the road-map. That said, we have an overall plan based on what we know today.

The plan consists of three components, starting with robust commercial implementations of critical software, incorporating center and partner constructed software and components, and incorporating and dealing with changes in the external environment, e.g., new standards, new threat characteristics, etc.

As stated above our intention is to begin with robust commercial software, and to use that whenever possible and cost-efficient. The reasons are simple: (1) the development costs to re-engineer what has already done are prohibitive, and (2) as a general rule the open source implementations of the same functionality are not yet as robust and mature as the commercial versions, and (3) the commercial versions are supported by an existing support infrastructure.

The GBG deployment plan is covered in the GBG Infrastructure Plan <cite>. Here we address the technology roadmap and how we expect to acquire and integrate the required pieces in the face of a constantly changing external technology environment. Figure 1-6 below presents a very high-level view, low-detail view of the roadmap.

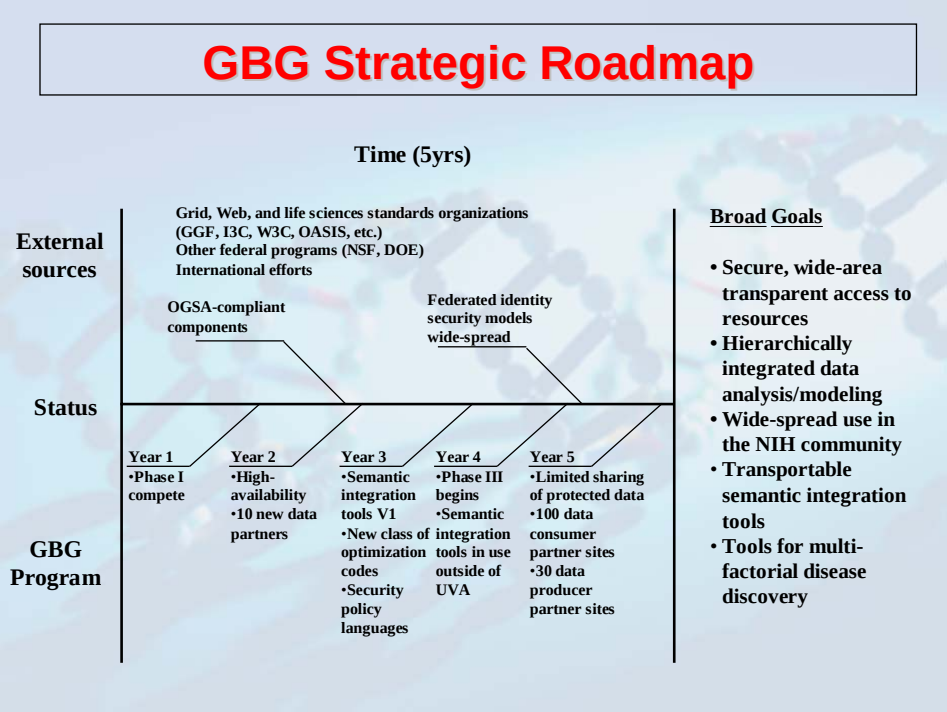


Figure 1-6 The GBG initial conditions are dominated by a nucleus set of sites and off-the-shelf commercial software. By year five the grid software base will have moved to web and grid-service based software components developed by a mixture of GBG staff, the community at large, and vendors.

The challenge presented by the rapidly changing technological landscape for the GBG is how to smoothly incorporate new software without disrupting the user base. This is complicated by the fact that testing grid software is more complex than testing single CPU, single administrative domain software. The number of ways that things can go wrong is large – and not always well understood.

The transition from our initial, commercial based grid, to an OGSA-based grid will be our single largest obstacle. Our approach will be to deploy two grids alongside one another for a long test period. And when the "new", OGSA-based grid is stable begin a rapid transition.

As to software developed in the context of the Center. We will use a model developed at NPACI for integrating and rating the stages that in-house-developed go through. The system involves a process of an external committee applying the following scheme.

Stage	Characterized by
1 - Demonstration	Identify target audience for the software At least one application and one site
2 - Early Deployment	Hardening and testing Multiple applications and one GBG resource site
3 - Pre-production	Wide deployment within GBG. Compatibility with GBG infrastructure for transition
4 - Production	Software of wide interest to the GBG research community Installed at all GBG resource sites
5 - Technology Transfer	Transition to "vendor" supported system

References

1. Institute of Medicine, N.R.C., *Large Scale Biomedical Science: Exploring Strategies for future research.* 2003.
2. Brewer, E., *Lessons from Giant-Scale Services.* IEEE Internet Computing, 2001.
3. Oppenheimer, A.G. and D.A. Patterson. *Why do Internet services fail, and what can be done about it?* in *Fourth USENIX Symposium on Internet Techologies and Systems (USITS '03)*. 2003.
4. Grimshaw, A. *Meta-Systems: An Approach Combining Parallel Processing and Heterogeneous Distributed Computing Systems*. in *Sixth International Parallel Processing Symposium Workshop on Heterogeneous Processing*. 1992. Beverly Hills, CA.
5. Grimshaw, A.N.-T.a.A.S., *Using Reflection for Incorporating Fault-Tolerance Techniques into Distributed Applications.* Parallel Processing Letters, 1999. **9**(2): p. 291-301.
6. Grimshaw, A.S., *Enterprise-Wide Computing.* Science, 1994. **256**: p. 892-894.
7. Grimshaw, A.S., *The Legion Vision of a Worldwide Virtual Computer*. Communications of the ACM, 1997. **40**(1): p. 39-45.
8. Grimshaw, A.S., et al., *Wide-Area Computing: Resource Sharing on a Large Scale.* IEEE Computer, 1999. **32**(5): p. 29-37.
9. Grimshaw, A.S., et al., *Metasystems.* Communications of the ACM, 1998. **41**(11): p. 486-555.
10. Grimshaw, A.S., M.A. Humphrey, and A. Natrajan, *A philosophical and technical comparison of Legion and Globus.* IBM Journal of Research & Development, 2004. **48**(2): p. 233-254.
11. Grimshaw, A.S., et al. *Architectural Support for Extensibility and Autonomy in Wide-Area Distributed Object Systems*. in *Proceedings of the 2000 Network and Distributed Systems Security Conference (NDSS'00)*. 2000. San Diego, California.
12. Grimshaw, A.S., et al., *From Legion to Avaki: The Persistence of Vision*, in *Grid Computing: Making the Global Infrastructure a Reality*, Fran Berman, Geoffrey Fox, and T. Hey, Editors. 2003.
13. Grimshaw, A.S., et al., *Campus-Wide Computing: Early Results Using Legion at the University of Virginia.* International Journal of Supercomputing Applications, 1997. **11**(2): p. 129-143.
14. Grimshaw, A.S., et al., *Metasystems: An Approach Combining Parallel Processing And Heterogeneous Distributed Computing Systems.* Journal of Parallel and Distributed Computing, 1994. **21**(3): p. 257-270.
15. Grimshaw, A.S. and W.A. Wulf. *Legion - A View from 50,000 Feet*. in *Proceedings of the 5th IEEE International Symposium on High Performance Distributed Computing*. 1996.
16. Grimshaw, A.S. and W.A. Wulf. *Legion flexible support for wide-area computing*. in *The Seventh ACM SIGOPS European Workshop*. 1996. Connemara, Ireland.
17. Grimshaw, A.S. and W.A. Wulf, *The Legion Vision of a Worldwide Virtual Computer.* Communications of the ACM, 1997. **40**(1): p. 39-45.
18. Chapin, S.J., et al., *Resource Management in Legion.* Journal of Future Generation Computing Systems, 1999. **15**: p. 583-594.
19. Chapin, S.J., et al. *The Legion Resource Managemen System*. in *5th Workshop on Job Scheduling Strategies for Parallel Processing in conjunction with the International Parallel and Distributed Processing Symposium*. 1999.
20. Chapin, S.J., et al., *A New Model of Security for Metasystems.* Journal of Future Generation Computing Systems, 1999. **15**: p. 713-722.

21. Ferrari, A. and A. Grimshaw, *Basic Fortran Support in Legion*. 1998, Department of Computer Science, University of Virginia.

22. Ferrari, A.J., S.J. Chapin, and A.S. Grimshaw, *Heterogeneous Process State Capture and Recovery Through Process Introspection.* Cluster Computing, 2000. **3**(2): p. 63-73.

23. Ferrari, A.J., et al. *A Flexible Security System for Metacomputing Environments*. in *7th International Conference on High-Performance Computing and Networking Europe (HPCN'99)*. 1999. Amsterdam.

24. Humphrey, M., et al. *Legion MPI: High Performance in Secure, Cross-MSRC, Cross-Architecture MPI Applications*. in *2001 DoD HPC Users Group Conference*. 2001. Biloxi, Mississippi.

25. Humphrey, M., et al. *Accountability and Control of Process Creation in Metasystems*. in *Proceedings of the 2000 Network and Distributed Systems Security Conference (NDSS'00)*. 2000. San Diego, CA.

26. Jin, L. and A. Grimshaw. *From MetaComputing to Metabusiness Processing*. in *IEEE International Conference on Cluster Computing - Cluster 2000*. 2000. Saxony, Germany.

27. Lewis, M.J., et al., *Support for Extensibility and Site Autonomy in the Legion Grid System Object Model.* Journal of Parallel and Distributed Computing, 2003. **Volume 63**: p. pp. 525-38.

28. Lewis, M.J. and A.S. Grimshaw. *The Core Legion Object Model*. in *Symposium on High Performance Distributed Computing (HPDC-5)*. 1996. Syracuse, NY.

29. Natrajan, A., et al., *Studying Protein Folding on the Grid: Experiences using CHARMM on NPACI Resources under Legion.* Grid Computing Environments 2003, Concurrency and Computation: Practice and Experience, 2003.

30. Natrajan, A., et al. *Protein Folding on the Grid: Experiences using CHARMM under Legion on NPACI Resources*. in *International Symposium on High Performance Distributed Computing (HPDC)*. 2001. San Francisco, California.

31. Natrajan, A., M. Humphrey, and A. Grimshaw. *Capacity and Capability Computing using Legion*. in *Proceedings of the 2001 International Conference on Computational Science*. 2001. San Francisco, CA.

32. Natrajan, A., M. Humphrey, and A.S. Grimshaw, *The Legion support for advanced parameter-space studies on a grid.* Future Generation Computing Systems, 2002. **18**: p. 1033-1052.

33. Natrajan, A., M.A. Humphrey, and A.S. Grimshaw, *Grids: Harnessing Geographically-Separated Resources in a Multi-Organisational Context.* High Performance Computing Systems, 2001.

34. Natrajan, A., M.A. Humphrey, and A.S. Grimshaw, *Grid Resource Management in Legion*, in *Resource Management for Grid Computing*, J. Schopf and J. Nabrzyski, Editors. 2003.

35. Natrajan, A., et al., *The Legion Grid Portal.* Grid Computing Environments, Concurrency and Computation: Practice and Experience, 2001.

36. Nguyen-Tuong, A., *Integrating Fault-Tolerance Techniques into Grid Applications*, in *Department of Computer Science*. 2000, University of Virginia.

37. Nguyen-Tuong, A. and A.S. Grimshaw, *Using Reflection for Incorporating Fault-Tolerance Techniques into Distributed Applications.* Parallel Processing Letters, 1999. **vol. 9**(No. 2): p. pp. 291-301.

38. Nguyen-Tuong, A., A.S. Grimshaw, and M. Hyett. *Exploiting Data-Flow for Fault-Tolerance in a Wide-Area Parallel System*. in *15th International Symposium on Reliable and Distributed Systems*. 1996.

39. Nguyen-Tuong, A., et al., *Towards Dependable Grids.* 2004, University of Virginia, Computer Science.

40. Viles, C.L., et al. *Enabling Flexibility in the Legion Run-Time Library*. in *the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'97)*. 1997. Las Vegas, NV.

41. Wasson, G., et al. *OGSI.NET: OGSI-compliance on the.NET Framework*. in *Proceedings of the 2004 IEEE International Symposium on Cluster Computing and the Grid*. 2004. Chicago, Illinois.

42. Wasson, G. and M. Humphrey. *Attribute-based Programming for Grid Services*. in *The Ninth Global Grid Forum, Workshop on Designing and Building Grid Services*. 2003. Chicago, IL.

43. Wasson, G. and M. Humphrey. *Policy and Enforcement in Virtual Organizations*. in *4th International Workshop on Grid Computing (Grid2003)*. 2003. Phoenix, AZ.

44. Wasson, G. and M. Humphrey. *Toward Explicit Policy Management for Virtual Organizations*. in *IEEE Workshop on Policies for Distributed Systems and Networks (POLICY'03)*. 2003.

45. Wasson, K.S., et al. *Tools Supporting the Communication of Critical Application Domain Knowledge in High Consequence Systems Development.* in *SAFECOMP 2003, The 22nd International Conference on Computer Safety, Reliability and Security*. 2003. Edinburgh, Scotland.

46. Waugh, A., G.A. Williams, L. Wie, & R. B. Altman. *Using Metacomputing Tools to Facilitiate Large Scale Analyses of Biological Databases*. in *Pacific Symposium on biocomputing 2001*. 2001. Mauna Lani.

47. Weissman, J.B. and A.S. Grimshaw. *Network Partitioning of Data Parallel Computations*. in *Symposium on High-Performance Distributed Computing (HPDC-3)*. 1994. San Francisco, CA.

48. Weissman, J.B. and A.S. Grimshaw, *A Framework for Partitioning Parallel Computations in Heterogeneous Environments.* Concurrency: Practice and Experience, 1995. **7**(5): p. 455-478.

49. Weissman, J.B. and A.S. Grimshaw. *A Federated Model for Scheduling in Wide-Area Systems*. in *Fifth International Symposium on High Performance Distributed Computing (HPDC-5)*. 1996.

50. White, B., A. Grimshaw, and A. Nguyen-Tuong. *Grid Based File Access: The Legion I/ O Model*. in *Proceedings of the Symposium on High Performance Distributed Computing (HPDC-9)*. 2000. Pittsburgh, PA.

51. White, B., et al. *LegionFS: A Secure and Scalable File System Supporting Cross-Domain High-Performance Applications*. in *SC 01*. 2001. Denver, CO.

52. Wulf, W., C. Wang, and D. Kienzle, *A New Model of Security for Distributed Systems.* 1995, Department of Computer Science, University of Virginia.

53. Smarr, L. and C.E. Catlett, *Metacomputing.* Communications of the ACM, 1992. **35**(6): p. 44-52.

54. Katramatos, D., et al. *JobQueue: A Computational Grid-wide Queuing System*. in *International Workshop on Grid Computing*. 2001.

55. Grimshaw, A.S., A. Natrajan, and M.A. Humphrey, *Legion: An Integrated Architecture for Grid Computing.* IBM Systems Journal.

56. Grimshaw, A.S., et al., *Wide-Area Computing: Resource Sharing on a Large Scale.* IEEE Computer, 1999. **32**(5): p. 29-37.

57. Grimshaw, A.S., A.J. Ferrari, and E.A. West, *Mentat*, in *Parallel Programming Using C++*, G. Wilson, Editor. 1996, MIT Press.

58. Grimshaw, A.S., J.B. Weissman, and W.T. Strayer, *Portable Run-Time Support for Dynamic Object-Oriented Parallel Processing.* ACM Transactions on Computer Systems, 1996. **14**(2): p. 139-170.

59. Zhou, S. *LSF: Load Sharing in Large-scale Heterogeneous Distributed Systems*. in *Work. on Cluster Computing*. 1992.

60. Ferstl, F., *CODINE Technical Overview.* Genias, 1993.

61. Abramson, D.e.a. *Nimrod: A Tool for Performing Parameterised Simulations using Distributed Workstations*. in *4th IEEE Intl. Symp. on High Performance Dist. Computing*. 1995.

62. Litzkow, M.J., M. Livny, and M.W. Mutka. *Condor-A Hunter of Idle Workstations*. in *The Eighth International Conference on Distributed Computing Systems*. 1988.

63. Foster, I. and C. Kesselman, *Globus: A Metacomputing Infrastructure Toolkit.* International Journal of Supercomputing Applications, 1997. **11**(2): p. 115-128.

64. Consortium, I.I.I.

65. Huang, H. and A. Grimshaw, *Grid-Based File Access: The Avaki I/O Model Performance Profile*. 2004, Department of Computer Science, University of Virginia: Charlottesville, VA.