Information Power Grid: The new frontier in parallel computing?

William Leinberger
Vipin Kumar

University of Minnesota

NASA's Information Power Grid is an example of an emerging, exciting concept that can potentially make high-performance computing power accessible to general users as easily and seamlessly as electricity from an electrical power grid. In the IPG system, high-performance computers located at geographically distributed sites will be connected over a high-speed interconnection network. Users will be able to submit computational jobs at any site, and the system will seek the best available computational resources, transfer the user's input data sets to that system, access other needed data sets from remote sites, perform the specified computations and analysis, and then return the resulting data sets to the user. Systems such as the IPG will be able to support larger applications than ever before.

New types of applications will also be enabled, such as multidisciplinary collaboration environments that couple geographically dispersed compute, data, scientific instruments, and people resources together using a suite of grid-wide services. IPG's fundamental technology comes from current research results in the area of large-scale computational grids. Figure 1 provides an intuitive view of a wide-area computational grid.

Recently, a panel discussion was engaged at IPPS/SPDP '99 to discuss the technical challenges of computational grids and the research underway to solve these challenges (see "Panel members" sidebar). This report summarizes that discussion.

The University of Minnesota's Vipin Kumar moderated the panel. Its members came from various backgrounds, each with his or her own perspective on computational grids. Subhash Saini from NASA Ames represented grid users. Dennis Gannon from Indiana University, currently visiting NASA Ames to support the IPG's development, provided an implementation perspective. Three other panel members are currently researching various aspects of computational grids. Kento Aida from the Tokyo Institute of Technology is working with performance evaluation and scheduling techniques for large-scale grids. Ian Foster of Argonne National Laboratory and the University of Chicago co-leads the Globus project (with Carl Kesselman), which investigates grid-enabling services (Foster's presentation was given by Saini). Andrew Grimshaw from

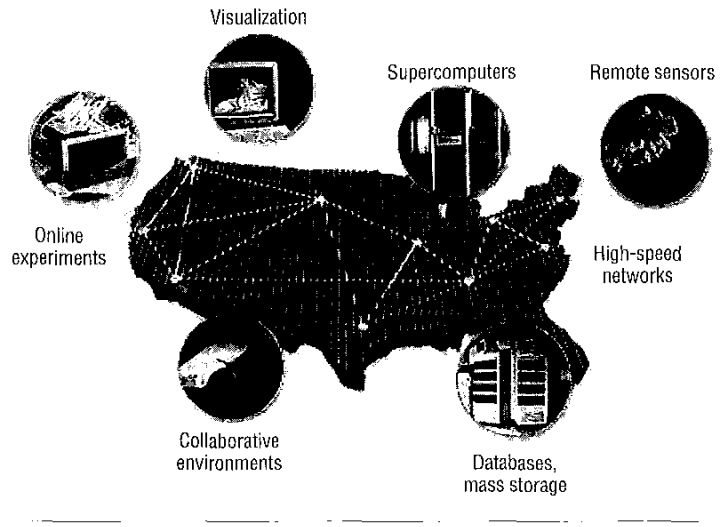


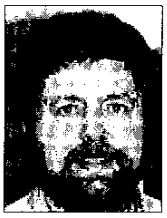
Figure 1. Conceptual view of a computational grid. (Figure courtesy of William Johnston, www.nas.nasa.gov/IPG, 1999.)

Panel members

Kento Aida is a lecturer in the Department of Computational Intelligence and Systems Science at the Tokyo Institute of Technology. His research interests include scheduling techniques on parallel and distributed systems, global computing systems, parallelizing compilers, and multiprocessor operating systems. He received his BS, MS and PhD degrees in electrical engineering from Waseda University, Tokyo. Contact him at aida@cs.dis.titech.ac.jp; www.cs.dis.titech. ac.jp/~aida.

Frederica Darema is the senior science and technology advisor at the National Science Foundation's Experimental and Integrative Activities (EIA) and the Directorate for Computer & Information Science and Engineering (CISE) , and director of the Next Generation Software Program. Her interests and technical contributions span the development of parallel applications, parallel algorithms, programming models, environments, and performance methods and tools for the design of applications and of software for parallel and distributed systems. She received a BS from the University of Athens, an MS in theoretical nuclear physics from the Illinois Institute of Technology, and her PhD in theoretical nuclear physics from the University of California at Davis. Contact her at darema@nsf.gov.

Ian Foster is senior scientist and associate director of the Mathematics and Computer Science Division at Argonne National Laboratory. He also holds a position as associate professor of Computer Science at the University of Chicago. His research interests are in high-performance parallel and distributed computing. He co-leads with Carl Kesselman of USC/ISI the Globus project, a multi-institutional effort developing basic infrastructure for high-performance grids. Contact him at foster@mcs.anl.gov.



Dennis Gannon is a professor in the Department of Computer Science at Indiana University, which he also chairs. His research interests involve the construction of distributed applications based on software component technology, the integration of parallel and distributed programming systems, the design of problem-solving "workbench portals," and distributed grid services. In addition, he is a partner in the NSF Computational Cosmology Grand Challenge project, the DOE 2000 Common Component Architecture software tools group, and the NCSA Alliance. Contact him at gannon@cs.indiana.edu.

the University of Virginia leads the Legion project, which is pursuing a component-based object model for grid architecture.

In any research community, the funding agencies play an important role in accelerating technical progress. The National Science Foundation's Frederica Darema has developed programs fostering research into new software approaches for grid application design environments and runtime systems. Finally, Jamshed Mirza from IBM's Server Group provided industry's perspective in the development of grid technology.

Even though the panel members represented a diverse set of backgrounds, all acknowledged the importance of computational grids. Although the overall discussion was aimed toward educating the audience about computational grids, there was also a significant discussion about the challenges in constructing them. Even with this consensus, computational grids encompass a broad spectrum of research topics. Given this diversity, any attempt to summarize each panel member's position statement would lead to a fragmented list detailing each panelist's current work. Therefore, this report tries to summarize and synthesize the collective thoughts of the panelists.

In particular, we will try to answer the following basic questions about computational grids:

- What is grid computing and why do we need it?
- What are the technical challenges in building large-scale grids?
- What are the enabling technologies required to solve these challenges?
- What is the role of industry in grid technology development?

What is grid computing and why do we need it?

While the power grid analogy provides some insight into how a computational grid should behave, grids are best described by their physical construction, target applications, and why these complex systems are needed.

Physically, high-speed LANs, WANs, and Long-Haul networks serve to connect heterogeneous computers, data repositories, user-interface facilities, and real-time scientific instruments. In time, these resources will be pooled together from geographically dispersed sites with different administrative domains. A grid-wide operating system bridges the heterogeneous systems together by providing a diverse set of services, creating the seamless computational engine, which supports new and diverse applications. Alternatively, Darema stated that future high-end petaflops platforms will be composed of heterogeneous processing nodes interconnected with multiple levels of networks, involve deep memory hierarchies, and interface to a variety of I/O devices. We might look at these platform architectures as "grids in a box" or GiBs (see Figure 2).

Saini, Foster, and Darema provided examples of grid-scale applications. These applications fall into the following categories:

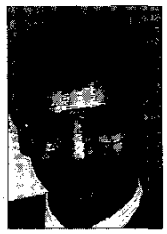
- *Just-in-time*, or *on-demand computing*. On-demand computing provides access to specialized resources that are only needed on a short-term or infrequent basis. When such a resource is requested, it is usually needed immediately. However, because access needs are sparse, there might not be sufficient justification to own the resources or even collocate users with the resources. In a grid environment, many users might share such resources cost-effectively. On-demand computing must dynamically support a potentially large user population and a number of diverse resources. Technology issues with on-demand computing include resource location, scheduling and coscheduling with local resources, code con-

figuration management, security, and payment mechanisms in an open environment.

- *Data-intensive computing.* With the current explosion in information, data-intensive computing becomes an important application area that must be supported by grids, which involves the synthesis of new information from geographically distributed data sources. A key issue that the grid must solve is the scheduling and configuration of high-volume data flows through the multiple levels of network hierarchy.
- *Collaborative computing.* Perhaps the newest application enabled by computational grids is collaborative computing, or the creation of laboratories without walls. The basic goal is to enhance the interactions between humans and computing resources that might be geographically distributed. There are many issues involved in dealing with human perceptual capabilities and the rich variety of interactions that might take place with the computational resources.
- *High-throughput computing.* High-throughput computing involves executing as many tasks as possible within a given time frame. Often, the tasks are loosely coupled or totally independent. The large number of resources available in a grid provides many spare cycles for throughput computing. Key to the success of high-throughput computing is accurate resource load information and efficient scheduling mechanisms.
- *Distributed supercomputing.* A driving goal of computational grids is to solve problems that cannot currently be solved on a single system by aggregating computational resources from many sites together. Fundamental issues include parallel algorithm scalability and tight coscheduling of resources from multiple computing sites. Implicit in distributed supercomputing is that the resources are typically very expensive and therefore very limited. Distributed supercomputing might be a component of other grid applications such as on-demand computing or collaborative computing.

Foster noted that a computational grid is not an alternative to parallel computing. Grids are loosely coupled by higher-latency and lower-bandwidth WAN interconnections. Additionally, only one of the five grid-scale application classes concentrate on traditional supercomputing applications. Many envisioned grid applications will span several of the above classes. For example, a tele-immersive engineering application might use on-demand distributed supercomputing to perform real-time computations but also require data-intensive methods to move the visualization data between the computational sites and the user interfaces. Another such example integrates dynamically collected data with simulations. Such applications extend the notion of computational grids to include embedded systems and remote sensors for field-data collection. These applications use data-intensive methods to move the data (collected in real time) to distributed supercomputers, which perform simulations based on the collected data in a just-in-time fashion. Successful grid applications couple resources that cannot be replicated at a single site (even with the expected growth of single systems). This is the driving force behind grids. Grids let users solve larger or new problems by pooling resources that could not be coupled before. According to Saini, grids enable a fully networked research community in which application scientists can interact with computer scientists. In the end, such multidiscipli-

Andrew S. Grimshaw is an associate professor of Computer Science and director of the Institute of Parallel Computation at the University of Virginia. His research interests include high-performance parallel computing, heterogeneous parallel computing, compilers for parallel systems, operating systems, and high-performance parallel I/O. He is the chief designer and architect of Mentat and Legion. He received his MS and PhD from the University of Illinois, Urbana-Champaign. Contact him at grimshaw@Virginia.edu.



Vipin Kumar is the director of the Army High Performance Computing Research Center and a professor of computer science at the University of Minnesota. His research interests include high-performance computing, parallel algorithms for scientific computing problems, and data mining. His research has resulted in the development of the isoefficiency metric for evaluating the scalability of parallel algorithms, as well as highly efficient parallel algorithms and software for sparse matrix factorization (PSPACES), graph partitioning (METIS, ParMetis), VLSI circuit partitioning (hMetis), and dense hierarchical solvers. Contact him at kumar@ cs.umn.edu; www.cs.umn.edu/~kumar.

Jamshed H. Mirza is an IBM Distinguished Engineer and a member of the IBM Academy of Technology. He works in the Server Architecture group, and has been a part of the team that designs and develops the RISC System/6000 SP system. His current responsibilities include systems architecture and technology strategy for future SP systems. His area of expertise is computer systems architecture with a focus on high-performance computing, superscalar designs, and parallel processing. Contact him at mirza@us.ibm.com.

Subhash Saini manages a department at NASA Ames Research Center that includes several groups such as Information Systems Performance Modeling; Algorithms, Architectures and Application; Information Power Grid Architectures; Legacy Codes Modernization; Higher Level Languages; Nanotechnology; and Device Modeling. His research interests involve performance evaluation and modeling of a new generation of CMOS-based processors and highly parallel computers. He received his PhD from the University of Southern California. Contact him at ssaini@mail.arc.nasa.gov.

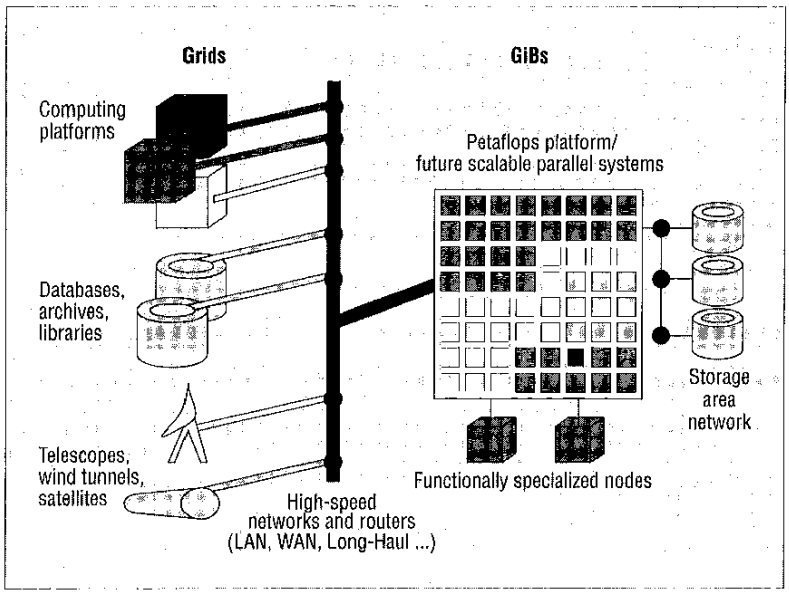


Figure 2. Physical grids and GiBs (grids in a box).

nary collaboration leads to highly optimized, lower-cost products.

Mirza pointed out that grid-like structures that can support the above application classes can also be used to solve a variety of commercial computing problems. In commercial computing, computers are being used not just for running a business—online transaction processing (OLTP), enterprise requirements planning (ERP), and document preparation, for example—but also increasingly to gain strategic competitive advantage by mining the transaction data to make business decisions. However, support costs dominate the cost of ownership of computing systems. In many business environments, the demand for computing cycles varies with business and product cycles as well, and is often difficult to predict. These trends make outsourcing of information technology services a very attractive proposition. Outsourcing of applications by big and small businesses will go beyond e-mail and Web-site hosting to database management, ERP, e-commerce, business intelligence, and design services. In effect, IT service providers will become application service providers. That, in essence, is a trend toward grid computing.

Mirza noted that grid-like structures are also emerging within large corporations and between smaller corporations. In this age of global corporations and global commerce, employees, customers, suppliers, contractors, corporate resources, and corporate data are all geographically dispersed. Intranets provide access to all corporate assets and knowledge resources to anyone in the corporation from anywhere. Internet and intranet grids enable collaboration and e-business by connecting corporate-wide business processes and by connecting the corporation to its customers, suppliers, and contractors. These e-business solutions rely on heterogeneous computing resources and are enabled by object-based technologies that make these applications function across diverse platform and operating systems (Enterprise Java Beans and Component Broker are examples). This is also a grid-based computing environment; only the specific applications are different. Instead of compute- or data-intensive applications, these are interactive, transaction-based applications.

Technical challenges for grid computing

The panelists highlighted three key characteristics of computational grids that make their implementation very difficult; grids are (potentially) very large, grid resources are heterogeneous at multiple levels, and grids are very dynamic.

By definition, computational grids are physically scalable. Networking standards such as Gbit Ethernet, ATM, and SONET provide common interconnect links, switches, and hubs for interconnecting parallel processors, storage devices, and user interfaces together across wide geographical distances. Bridge technology is available to link in other devices such as scientific sensors. Interprocess communication and control standards such as TCP/IP, MPI, and MAP provide the necessary software connectivity between heterogeneous resources. Estimates for the number of resources in a grid range from single digits to millions of devices.

The core hardware resources of computational grids—the computers, data-storage devices, networks, and so forth—are heterogeneous. At the software level, different programming languages or models, operating system versions, or availability of libraries or compiler licenses make similar hardware systems essentially different. Information resources such as databases also add heterogeneity at an application level. Finally, the ownership of these resources is often distributed across multiple administrative domains, giving rise to heterogeneity in the policies governing the use of particular resources.

Grids are highly dynamic, due to the dynamic nature of resource availability as well as the requirements of the envisioned applications. The existence of multiple administrative domains also contributes to this characteristic. This is in sharp contrast to traditional systems, which are generally considered static. Resource availability changes on a continual basis as resources are added, deleted, upgraded, modified, and moved within the grid. For example, consider the impact of upgrading the operating system on a parallel processor. This essentially deletes the old resource (the parallel processor with the old operating system version) and creates a new and different resource (the parallel processor with the new operating system version). Applications that could use the original resource might not be compatible with the new resource. Alternatively, the new resource might enable a larger set of applications. The state of any particular resource might change rapidly. For example, the load on a particular parallel-processing system changes as new jobs are selected for execution or existing jobs complete. Additionally, communication latencies between resources are both high and variable as traffic from other applications asynchronously compete for shared network links. Finally, resource failure is the rule, not the exception. With so

many resources in a grid, the probability of some resource failing is naturally high. Note that resource failure is somewhat different than resource availability in that availability affects resource allocation, which occurs before the application (or subtask of the application) begins execution. Resource failure might occur after resource allocation and impact the execution of an application (or subtask).

These three characteristics make computational grids far more complex than can be managed by current methods for heterogeneous- or distributed-computing platforms, and are the basis for numerous technical challenges in implementing grids. Each panel member described the technical challenges of building a large-scale computational grid from his or her own perspective. The following list condenses the discussion into the top five technical challenges.

Complexity management

Above all, a grid must be easy to use. This is implicit in using "seamless" to describe how grid resources are to be integrated. Grid hardware and software resources are heterogeneous in nature and might span multiple administrative domains across wide geographical distances. The underlying complexity must be hidden from the user. This is especially important given the multidisciplinary nature of the target user bases. The grid is a tool to be used to find a solution, not the solution itself. It must be easy for a user community to compose, debug, and execute or interact with an application. Users thus must understand, on some abstract level, both the capabilities and limitations of the grid. This level of understanding is also necessary so that users can extend or alter applications to enhance their solutions. Grids are inherently evolving as new resources are added while others are scrapped. The two to three years currently required to develop and performance-tune a large application for current parallel systems must fall significantly. To make use of newly acquired resources, or replace obsolete resources, applications must be developed or ported quickly, and at an acceptable level of performance.

Grid performance

How do we measure the effectiveness of a grid? Performance has traditionally been the driving measure of success for any computational platform. For a single user, performance is defined as measured runtime or other classical measures such as wait time or turnaround time. In contrast, from a system perspective, a high utilization of system resources is considered the goal. Both performance perspectives are also relevant in a grid environment, especially in the context of the distributed supercomputing and high-throughput computing application classes. Other application classes might require a different definition of performance, however.

A major goal of grids is to solve new problems, ones that could not be solved before, such as applications from the data-intensive, collaborative computing, or on-demand computing classes. For example, suppose a grid can compute a detailed weather model that can predict the movements and severity of a hurricane in accelerated time by combining online weather sensors, data on past hurricanes from archives, and interactive visualization stations at different weather analysis centers. However, the execution of this application results in low system utilization (of some resources) or severe degradation in average wait time as seen by other users sharing the same grid.

In this case, we would evaluate performance by the capability of completing the computation in a timely manner and less by traditional measures. Therefore, grid performance might need to be redefined as a success rate in providing new capabilities. In general, measuring the performance of grids as the value of the applications that they enable might be difficult. Furthermore, supporting multiple (heterogeneous) performance objectives in a common grid is a major technical challenge.

Grid scalability

How large can a grid grow and still be useful? The loose coupling between the grid resources results in performance degradation as the grid's size increases. Although any single device might have a few high-bandwidth links to the grid network, the latencies in using these links might be high and variable under heavy sharing. This loose external coupling makes it difficult to efficiently support the grid-scale interprocess communication, synchronization, and coscheduling functions required for scalable applications.

Applications that require a large number of geographically located resources must be designed to be extremely latency tolerant. As the size of the grid grows, it might be impossible to achieve a single, controllable system image across even a small subset of grid resources, due to interference from activity in other parts of the grid. Finally, many high-performance computing resources (such as parallel processors) and I/O servers are tightly coupled internally but have limited connectivity to the outside world. These devices are designed for operations that execute entirely within their physical limits. Typically, they are not designed to be tightly coupled with other geographically dispersed devices. This characteristic further reduces the size of a usable grid.

Designing grid-aware system software and applications

Given the dynamic nature of a computational grid, applications must be able to adapt at startup and runtime to best use the currently available resources. Applications must also be able to dynamically request new resources. The application, resource management, and runtime services in the grid typically share responsibility for adapting to the grid's current state. The application must be designed to be flexible in what resources it uses, have mechanisms for querying the system state, and have some decision support for how to reconfigure based on the current system state. Current programming environments typically support application development only for statically defined systems. In addition to being able to express adaptivity in the applications programming model, services for system state queries and runtime support for reconfiguration

are required. Adaptively managing for resource availability or failure becomes more complex as more resources are required by the application.

Administration

Computational grids might contain resources from many different administrative domains with different resource-usage policies. Therefore, no longer does any single entity own or control responsibility for functions such as system administration, security, resource ownership, maintenance, accounting, and user support. Guidelines for participation in a grid must be developed that are both amenable to productivity yet enforceable. Two of the most important functions that must be resolved are site autonomy and security.

Grids let users construct new and larger applications by pooling more resources together than are available at any single site. However, each site still maintains responsibility for its own resources and needs to control the policies for how other grid users might use them. This is especially true in the near future when the resources of a given site are primarily intended for supporting efforts other than the grid computations. These local functions must be supported deterministically, with excess capacities provided to the grid users. Usage policies at one site might directly conflict with the policies at another site, making it impossible for a single application to simultaneously use resources from both sites. In future grid economies, sites might cooperate in obtaining complementary resources intended for the collective benefit of enabling emerging grid applications.

Grid participants must trust the grid system to enforce proper resource usage as well as to protect their proprietary data or they will not use it.

Related to site autonomy is grid security. In the grid environment, this becomes very important as different communities (industry, academia, government labs, the US Department of Defense, and so forth) will wish to share grid resources. Each community might have a different set of security requirements, at different levels of implementation costs, but all must be supported. Grid participants must trust the grid system to enforce proper resource usage as well as to protect their proprietary data or they will not use it.

Enabling technologies

Given the technical challenges we've just discussed, you might naturally ask, "What are we doing to meet these challenges?" While the scope of the current research related to computational grids is enormous, the panel members pinpointed five key technology areas aimed specifically at meeting these challenges. An emerging grid architecture provides the framework for a generic computational grid. The use of performance-engineering methods and the creation of grid services deal with the dynamic nature and scalability issues of grids. A coherent, object-oriented framework reduces the complexity in building grid components from low-level communications to high-level applications. Finally, large-scale, persistent testbeds serve to discover the realities of implementing, using, sharing, and maintaining computational grids.

The emerging grid architecture

While supercomputing is a key challenge for grids, they also enable a variety of new problem solutions. Grid applications are conceptually more complex than current parallel and distributed applications. Many are interactive and require robust performance guarantees from the grid infrastructure. Additionally, these applications require a diverse set of services to use the grid resources effectively. These requirements have led to the emerging architecture for computational grids described later, as provided in Foster's presentation.

Figure 3 depicts the Integrated Grid Architecture (Ian Foster, *An Integrated Grid Architecture*, tech. report, Argonne National Laboratory, 1999). At the highest level is the applications layer. Applications from the classes we described just now are designed and developed using a set of services from the application toolkit layer. This toolkit layer provides a set of services to application developers or end users tailored to their specific application domain. Included in the toolkit layer might be a problem-solving environment for a particular domain. The primary goal of the application toolkit layer is to abstract the computational grid to a level, which the user might understand and exploit. The key assumption here is the fact that the user very likely is not a computer scientist!

The application toolkit layer makes use of a general set of services provided by the grid services layer. This layer provides a set of functions, analogous to the runtime of a single system, which are required for building the grid-scale applications. An example is an information service, which might be used for locating a particular type of resource or for querying the current load on a grid resource. Note that similar services are currently used in distributed-computing applications. However, in a grid environment, these services must scale to handle a larger number of resources.

The lowest level of the grid architecture is the grid fabric layer. This layer is analogous to the operating system of a single system and provides a seamless fabric across the various grid resources. The fabric layer provides a stable programming interface to the dynamic resources in the grid. The grid fabric is built on the actual grid resources and associated commercial computing standards.

Foster noted that the architecture depicted in Figure 3 provides a set of orthogonal services which may be used to construct grid-scale applications. This approach has been used for other distributed systems such as CORBA and the Internet and is being adopted by most of the current grid research projects. However, in a post-panel discussion, Grimshaw argued

that this sum-of-services approach is basically flawed in that it doesn't actively support grid-specific issues such as complexity management and fault tolerance.

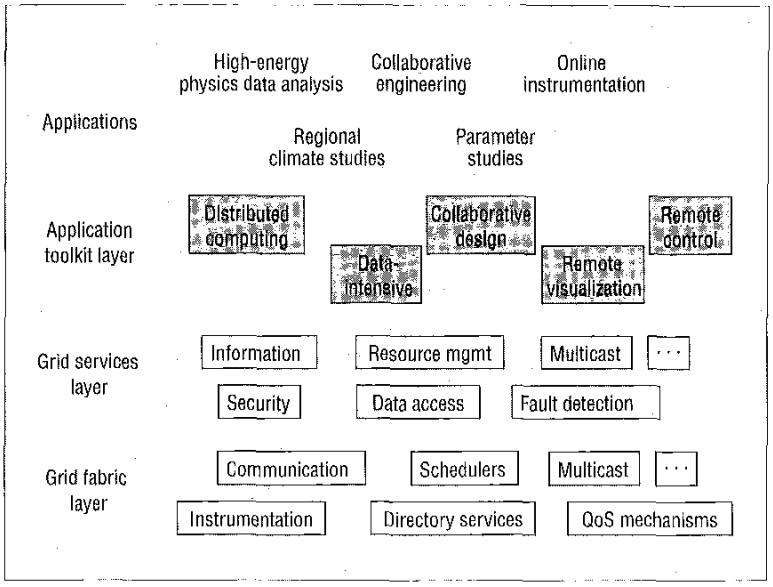


Figure 3. Integrated grid architecture.

Performance engineering for grid-aware applications

A primary research challenge involves dealing with the dynamic nature of computational grids. The new types of applications require coupling multiple devices across geographical locations as well as new grid services and possible new features in local operating systems. The grid resources themselves are quite heterogeneous and dynamic. Therefore, the application-programming models, runtime systems, and grid services need to be grid-aware so that adaptive applications can be composed quickly and executed efficiently in the dynamic grid environment. The ad hoc design methods used in the past are not sufficient. New methods, which provide systematic design tools and performance tools, are needed so that performance considerations are designed into an application from the outset. Furthermore, enhancements to the application and runtime environments are needed to deal with the dynamic grid system. Darema provided an overview of the current NSF New Generation Software program. NGS has two components: performance engineering and compiler and application-composition technologies.

The NGS program takes a strong view that performance-engineering technology plays a key role in supporting adaptive applications. Performance models of grid components at all levels are constructed, validated, and combined into a performance framework. Models, simulators, and measurements for computing resources, networks, and other devices as well as for operating system services, grid services, and application components must be developed to create an accurate system model. Furthermore, to make such analysis tractable, these multimodal methods need to be able to describe the system at multiple levels of abstraction. The performance framework then assists in the composition of applications by providing performance analysis on a system-level model, which is composed from component models at the appropriate level of detail required to support the target performance objective. Design trades might be made between types of resources to use or even algorithmic approaches. The framework includes the complex behavior of all the complete system's components so that users can perform an accurate performance analysis.

Aida advocated a similar thrust. Aida has been working the Ninf project, which models network performance as well as server performance at a detailed level. This supports an accurate evaluation of scheduling systems when applied to grid-scale systems. Experimental results verify that this approach provides a more accurate performance prediction than current methods, which use simplified models.

While application composition is in itself a hard problem, giving the applications the ability to adapt to the dynamic grid during execution is even harder. The second portion of the NGS focuses on incorporating performance-engineering methods into the application programming model and runtime system. Essentially, the application can query the state-of-the-grid resources on the fly (with the runtime system) and adapt to the changes. The performance models and measurements can aid the decision-making process. Grid-resource-management services might use similar methods to aid in resource allocation and scheduling.

The end goal is to greatly reduce the time to create or port a large application in a grid environment, and to produce applications that execute with a deterministic Quality of Service (QoS).

Scalable grid services

The performance-engineering framework research provides a view of the grid resources at multiple levels. In the integrated grid architecture of Figure 3, this has the greatest impact at the grid-services layer where the performance-prediction information can serve to support resource management and runtime-application adaptation, as well as the application-toolkit layer where it supports application composition from components. The performance-modeling framework represents a grid service, which provides a functionality useful to many applications. In general, the grid-services and application-toolkit layers in the integrated architecture provide a set of common functions, which might be used by many applications.

Research in grid services aims to provide a seamless virtual image of the computational grid, much like modern operating systems provide a virtual machine image to compilers in a uni- or parallel-processing system. However, the services required in a grid are quite different because of the scale and dynamic properties of computational grids, as well as other properties related to the types of envisioned applications, separate administrative domains, and so forth. Gannon summarized his current

research on the IPG project, which has identified two sublayers of grid services: gridwide core services and higher-level application toolkit services. The work leverages heavily off the Globus project, which is researching grid-scale services.

Core services must be supported throughout the entire grid. The user might never directly see many of these. In fact, they are part of the infrastructure that makes "seamless" possible. Some examples have user authentication and other security mechanisms, uniform resource naming, resource discovery and brokering, QoS mechanisms for multiresource coscheduling, and event management, logging, and notification. These services might be integrated directly into the application-programming model or be part of a global shell used for resource access and job control.

In addition to the gridwide core services, users need a number of additional services to put an application up on the grid. These higher-level application toolkit services include visualization tools, scripting tools, object models and component composition tools, and publishing tools. The problem-solving environments use this services layer, which interacts heavily with the lower-level, core grid services.

Object-oriented frameworks for reducing complexity

The grid services functionally provide a seamless interface to the complex heterogeneous grid resources. However, to be of any use, these services must be completely interoperable. In this light, Grimshaw proposed that a computational grid requires a single coherent, object-based model as its foundation. A single coherent model should provide three key advantages (or the three Cs):

The grid services functionally provide seamless interface to the complex heterogeneous grid resources. To be of any use, these services must be completely interoperable.

- *Composition.* Application construction is vastly simpler if components can be easily composed into larger components. As an example, consider the power of Unix processes and pipes.
- *Communication.* Intercommunication standards between components must be standardized. The interfaces to a component must be well-defined and typed. Some minimum functionality should exist in all interfaces, to facilitate extracting the complete interface. The MAC clipboard is an example.
- *Clarity.* A single coherent model is required to "bring order to the Babel" in the complex grid environment.

In addition to using a single coherent model, Grimshaw proposed that the model be object-based. The "everything is an object" notion implies that all grid components (application software, grid services, and hardware resources) are encapsulated as objects with well-defined functions and interfaces. This approach supports legacy applications (using wrappers) and simplifies the composition of new applications and grid services. Object methods provide a simple way to describe resources and services, for example, in terms of their capabilities and interfaces. Object-based system models also reduce complexity by encapsulating complex behaviors within a component object such as fault tolerance, parallelism, and other implementation details. Objects also provide a clean mechanism for extension through inheritance or hierarchical composition. Object composition might be used to construct objects with new behaviors. Finally, object boundaries are a natural point to exercise access control for enforcing security policies.

Large-scale persistent testbeds

Concept validation is an important part of research. Transferring the technology to a real user community and letting them "kick the tires" is one way of accomplishing this. The computational grid concept's complex nature leads to one conclusion: that large-scale testbeds are required to test, refine, and validate the concept. Aida used a set of distributed compute servers, connected by the Internet, as a large testbed for validating the performance-modeling approach that he designed to evaluate grid-scale scheduling algorithms. Aida's work is part of a larger ongoing global computing infrastructure project (Ninf) performed in collaboration with the Electrotechnical Laboratory, the Real World Computing Partnership, and the Tokyo Institute of Technology in Japan. Both the Globus project (led by Foster and Kesselman) and the Legion project (led by Grimshaw) are using large-scale testbeds to test and refine various grid technologies.

Another large testbed is NASA's IPG, presented by panel members Saini and Gannon. The IPG is focused on improving the Aeronautics and Space Transportation Technology design process. The goal is to create a computational grid that links the computational, data, and human resources throughout the various NASA sites. This grid will provide an increased computational capability that can serve to decrease the design cycle time of air and space vehicles as well as improve the quality by supporting innovative designs derived from early multidiscipline trade studies. Essentially, the IPG will enable larger design simulations than ever before. Additionally, the IPG will support the integration of multiple data sources (experimental, computational, theoretical, low-to-high fidelity) and support multidisciplinary design teams in a collaborative environment. Application-design environments will support the use of the IPG by "nonprogrammers."

Building the IPG involves identifying and establishing collaborations with application communities that will use, validate, and participate in refining the grid infrastructure and middleware, and that have direct relevance to the NASA mission. Essentially, the IPG grid project is evolutionary, build-

ing on the work of grid-research projects such as Globus and Legion. Its objective is to transfer the grid-research technology and tailor it to build to the next-generation information applications that are important to NASA's mission and customers. Along the way, it will discover the realities of implementing, administering, maintaining, and evolving a large-scale computational grid.

What is the role of industry in grid-technology development?

Computational grids appear to be the current hot topic in the research environment. Even NASA's IPG might be considered an infant research project. Until now, most of the research has been funded by government-related agencies. In the commercial world, the growth in the use of high-performance computing as well as Internet and intranet usage will ultimately benefit from grid-like structures as well. In the near future, IT providers will use grid-like structures to supply high-performance computing services to the commercial world, much like grids will be used to support the HPC world. Additionally, grid-like structures will be used to solve the interactive, transaction-based applications required by the growing use of e-business-related solutions. So the final question to ask is, "What is the role of industry in the development of grid technology?" Mirza provided his personal view on the dynamics of industry with respect to computational grids and their prognosis for the near future.

Information grids provide a natural mechanism to solve applications where distribution is part of the problem.

According to Mirza, grid technology and applications fall into two categories. In one case, distribution is part of the solution. These are heroic compute-intensive applications that need prodigious amounts of computing resources, which a single supercomputer cannot satisfy; aggregating distributed supercomputers and distributing parts of application across these resources is the solution the grid provides. In the other case, distribution is part of the problem. There are corporations and organizations with distributed resources and assets, and people who need access to these assets; you need a solution that connects them together so they can function together. In both cases, a grid-like environment is the answer.

Information grids provide a natural mechanism to solve applications where distribution is part of the problem. These are the applications that are emerging as a result of the global economy and the trend toward leveraging the Internet's standards, simplicity, and connectivity to transform key business processes of a global corporation or organization. Increasingly, corporations will use grid-based solutions for running their business—to communicate with their partners and customers, to connect with their back-end data systems and knowledge assets, and to transact commerce. It includes

- customer relationship management (providing quality customer self-service, giving customers controlled access to data they need, analyzing past customer behavior to personalize offerings, and anticipate the customers' wants and needs),
- supply-chain management (transforming the way the supply chain is managed by using Internet technology to integrate the complex system of suppliers, partners, employees, and customers), and
- e-commerce (transforming the way the enterprise conducts business through online transactions, streamlining electronic billing and payment systems, and fundamentally revamping revenue and cost structures).

These are the applications of most interest to the industry at large, and these are the ones industry will focus on first. While many of the general problem areas might be common (issues of heterogeneity, connectivity, security, and QoS, for example) the specific issues to be addressed are very different because the nature of the applications are very different.

Industry by itself is unlikely to make significant advances in middleware and solutions for grid applications where distribution is not the problem but the solution. Cooperative effort by government, industry, and the research community is required to make progress there. These are problems, many of them of national importance, that require compute resources that are beyond what a single system can deliver. Partitioning and distributing the computation across multiple systems is the solution. But this requires a heroic programming effort and major breakthroughs, much of which is targeted in the grid research plans for programming model languages, compilers, tools, libraries, schedulers, resource managers, and so forth. Initiatives such as the IPG are therefore every important. Although these research and technology advances might not be directly applicable today to problems of interest to the general industry, they are nonetheless important. This is because historically, whatever has been good for the HPC community has ultimately been assimilated in the mainstream-computing arena. In fact, many of the advances here will likely accelerate technologies that are very relevant to future high-end servers, which can be expected to have many grid-like characteristics. Like the grid, future scalable parallel systems will have heterogeneous nodes, multilevel hierarchy of connection mechanisms, and a need to dynamically partition and allocate the resources within these systems. These systems will have many of the same problems being tackled by the grid research today and will benefit from the advances that this effort produces.

The IPG's basic concept has considerable merit. Panelists agreed that it is a worthy goal to pursue but that overcoming the technical challenges needed for achieving an effec-

tive grid will require considerable effort. Computational grids provide the ability to solve larger computational problems than ever before, as well as enabling the solution to new problems. However, building such a grid is difficult due to the technical challenges that result from working with a large number of geographically distributed, heterogeneous, and dynamic resources. Users of current-generation parallel and distributed systems also face many of these challenges. The grid environment simply extends the challenges. Therefore, research in computational grids will also advance the state of the art in these smaller high-performance computing platforms.

Several efforts are already underway in developing enabling technologies that can address these challenges. A great deal will be learned from the early testbeds, such as NASA's IPG. At the moment, such efforts are largely based in the HPC community, which has historically been the trailblazer for many key technologies that were eventually adopted for mainstream computing. The use of successively more sophisticated supercomputer and parallel architectures in mainstream business and the wild embrace of the Internet are but two examples. However, given the importance of the grid concept to other non-HPC domains, steps must be taken to ensure that the mechanisms designed for constructing HPC-based grids will also support commercially viable grid applications so that this research truly benefits society as a whole.

Acknowledgments

We would like to acknowledge the support provided by the Army High Performance Computing Research Center (cooperative agreement number DAAH04-95-2-0003/contract number DAAH04-95-C-0008), NASA (grant NCC2-5268), and the Minnesota Supercomputing Institute in the preparation of this article.

William Leinberger is a PhD student and Research Fellow in the Department of Computer and Information Sciences at the University of Minnesota. He received a BS in computer and electrical engineering from Purdue University. His current research interests include high-performance computer architectures, parallel and distributed processing systems, and resource management for computational grids. He is currently on an educational leave from General Dynamics Information Systems, Bloomington, Minnesota, where he has held positions as a hardware engineer, systems engineer, and systems architect in the general area of special-purpose processing systems. Contact him at leinberg@cs.umn.edu.

Vipin Kumar's biography appears on page 77.