The Grid: A New Infrastructure for 21st Century Science

Grid technologies promise to transform the practice of science and engineering, by enabling large-scale resource sharing and coordinated problem solving within farflung communities

Ian Foster

The history of science is in part a history of its tools, and some of today's most important tools are concerned with computation and communication. Driven by increasingly complex problems and by advances in understanding and technique, and powered by the emergence of the Internet and by Moore's Law increases in computer performance, today's science is as much based on computation, data analysis, and collaboration as on the efforts of individual experimentalists and theorists.

Given such developments, it should not be surprising that despite continued exponential technological improvements, computation, storage, and communication resources fail to keep up with demand. A PC in 2001 is (literally) as fast as a supercomputer of 1990—but while in 1990 biologists were happy to run a single molecular structure computation, now they want to screen thousands of drug candidates. A PC now can have 100 gigabytes (GB) of storage, as much as an entire 1990 supercomputer center—but physicists are engaged in projects that will produce multiple petabytes of data per year by 2005. Wide area networks often now operate at 155 megabits per second (Mb/s), three orders of magnitude faster than the state-of-the-art 56 Kb/s that connected U.S. supercomputer centers in 1985, but scientists are demanding 10s of Gb/s so that they can collaborate with colleagues across the world on the analysis of petabyte datasets.

What many term the "Grid" offers a potential means of addressing these obstacles to progress [3]. This new class of infrastructure and tools layers on today's Internet and Web to enable large-scale sharing of resources within distributed, often loosely coordinated groups—what are sometimes termed *virtual organizations* [5]. By providing scalable, secure, high-performance mechanisms for discovering and negotiating access to remote resources, Grid technologies promise to make it possible for scientific collaborations to share resources on an unprecedented scale, and for geographically distributed groups to work together in ways that were previously impossible [12, 13].

The concept of distributed resource sharing is certainly not new. For example, in 1965, the designers of the Multics operating system envisioned a computer facility operating as a utility, "like a power company or water company" [15], while in 1968, J. C. R. Licklider and Robert Taylor wrote of the computer as a communications device and described Grid-like scenarios [11]. Since then, much work has been devoted to developing distributed systems, with mixed success. Yet I, and my many colleagues working on Grids, believe that a combination of technology trends and research advances make it now feasible to realize the Grid vision and to put in place both a new

international scientific infrastructure and new tools based on that infrastructure that can, together, meet the challenging demands of 21st Century science.

In this article, I review the technology trends that motivate the Grid, explain the types of resource sharing enabled by Grids and their significance for science, and identify key supporting technologies.

Technology Trends

A useful metric for the rate of technological change is the average time period during which speed or capacity doubles—or, more or less equivalently, halves in price. For storage, networks, and computing, these periods are around 12, 9, and 18 months, respectively. The different time constants associated with these three exponentials have significant implications.

The annual doubling of data storage capacity, as measured in bits per unit area, has already reduced the cost of a commodity terabyte (10^{12} bytes) disk farm to less than US$10,000, and can allow major physics experiments to plan multi-petabyte (10^{15} byte) data archives—and allow simulation scientists in such areas as climate and astrophysics to think of archiving petabyte-scale outputs from high-resolution reference simulations. In consequence, large archival datasets are emerging as major community resources in many disciplines.

These data volumes place increased demands on our analysis capabilities. Dramatic improvements in microprocessor performance mean that the lowly desktop or laptop is now a powerful computational engine in its own right. Nevertheless, computational capabilities are falling behind relative to storage, "only" doubling every 18 months and so "only" increasing by a single order of magnitude every 5 years. It is becoming infeasible to assemble at a single location the computational resources needed for large-scale analysis.

The solution to these problems lies in the yet more dramatic changes taking place in networking. Spurred by innovations such as doping [14] and the demands of the Internet economy [9], the performance of wide area networks doubles roughly every 9 months—i.e., increases by *two* orders of magnitude every 5 years. For example, within the U.S., the NSFnet connected the NSF supercomputer centers by a then-unprecedented 56 Kb/s backbone in 1985; in 2002, the centers will be connected by the 40 Gb/s TeraGrid network (www.teragrid.org): six orders of magnitude improvement in 17 years.

This doubling of network performance relative to computer speed every 18 months has already changed how we think about and undertake collaboration. With at least one additional order of magnitude differential expected over the next five years, communication becomes essentially "free," instead of prohibitively expensive, as in the past. We can—indeed must—imagine new ways of working and doing business that are communication intensive: pooling computational resources, streaming large amounts of data from databases or instruments to remote computers, linking sensors with each other and with computers and archives, and connecting people, computing, and storage in

collaborative environments that avoid a need for travel [8]. More succinctly: how can we squander bandwidth to do better science?

What is a Grid?

If communication is unlimited and free, then we are not restricted, when solving problems, to local resources. For example, when running a colleague's simulation code, I need not install it locally but can run it remotely, on my colleague's computer. When applying this code to datasets maintained at other locations, I need not obtain copies of those datasets myself (not so long ago, I would have requested tapes), but can have the remote code access those datasets directly. If I wish to repeat the analysis many hundreds of times, on different datasets, I may call upon the collective computing power of my research collaboration—or purchase cycles from a cycle provider. And when I obtain interesting results, I can review the large output datasets with remote colleagues in rich collaborative environments.

While these scenarios vary considerably in their complexity, there is a common thread: in each case, I use remote resources to do things that I cannot easily do "at home." Now while high-speed networks are often necessary for such remote resource use, they are far from sufficient. Remote resources are typically owned by others, exist within different administrative domains, run different software, and are subject to different security and access control policies. These issues characterize a Grid and historically have made distributed computing difficult. In order to access remote resources, I must first discover that they exist, then negotiate access to them (to be practical, this step cannot involve using the telephone!), then configure my computation to use them effectively—and I must do all these things without compromising my own security or the security of the resources on which I am computing. I may also be called upon to pay for resources. Today's Internet and Web technologies address basic communication requirements, but do not provide uniform mechanisms for such critical tasks as creating and managing services on remote computers, for supporting "single sign on" to distributed resources, for transferring large datasets at high speeds, or for forming large distributed virtual communities and maintaining information about the existence, state, and usage policies of community resources.

In brief, Grid systems and technologies provide the *infrastructure* and *tools* that make large-scale, secure resource sharing possible and straightforward. An *infrastructure* is a technology that we can take for granted when performing our activities. Thus the road system enables us to travel by car, the international banking system allows us to transfer funds across borders, and Internet protocols allow us to communicate with virtually any electronic device. To be useful, an infrastructure technology must typically be broadly deployed, which means in turn that it must be simple—or extraordinarily valuable. For example, the set of protocols that must be implemented within a device to allow Internet access is small: people have constructed matchbox-sized Web servers. A Grid infrastructure needs to provide more functionality than the Internet on which it builds, providing new services that address end-to-end issues of authentication, resource discovery, and resource access—but it must also remain simple. And of course we must not forget the need for the resources that "power the Grid," by supporting, for example,

high-speed data movement, caching of large datasets, and on-demand access to computing.

Tools build on infrastructure services to support new modes of working. Internet and Web tools include browsers for accessing remote Web sites, email programs for handling email, and search engines for locating Web pages. Grid tools are concerned with resource discovery, data management, scheduling of computation, security, and so forth. For example, Figure 1 illustrates tools that support so-called "Data Grid" applications, which harness storage, computing, and networks to support distributed access to, and analysis of, large datasets. Building on Grid infrastructure services, Data Grids provide services for virtual data management (keeping track of where data products are located and, if these data products are generated via analysis routines, whether or not the data products are actually materialized), planning the often complex series of computations and data movements required to satisfy a user request, and managing those computations and data movements.

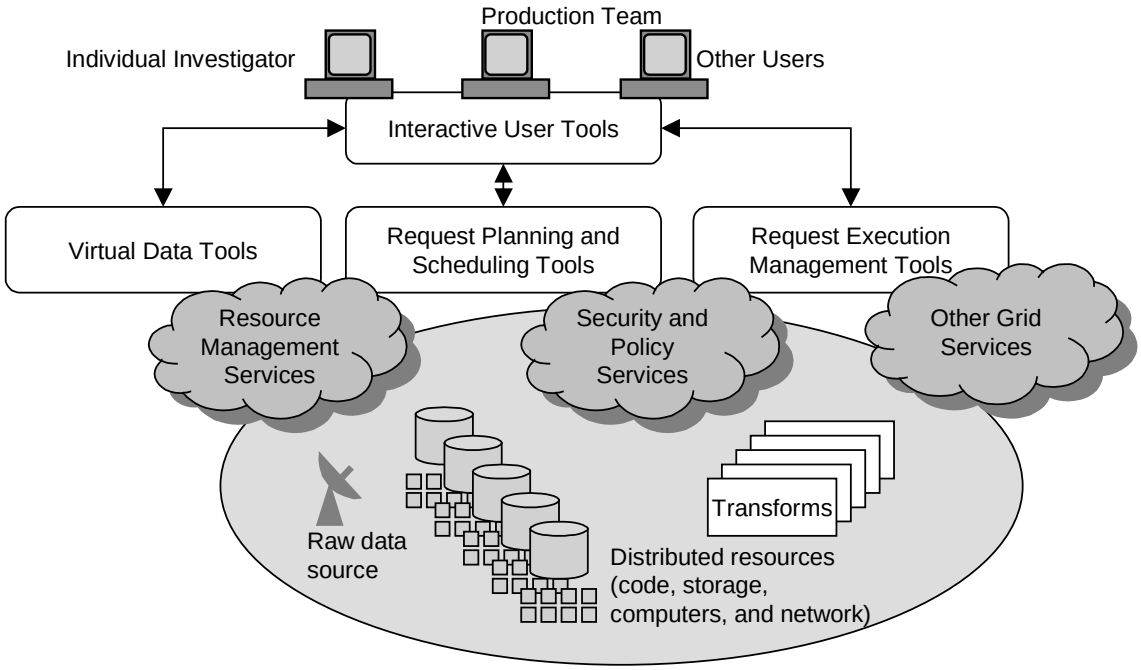

Figure 1: Principal elements of a petascale data grid, as envisioned within the NSF-funded Grid Physics Network project (GriPhyN: www.griphyn.org). GriPhyN R&D is complemented by the work of the EU DataGrid (www.eu-data-grid.org), DOE-funded Particle Physics Data Grid (PPDG: www.ppdg.net), and DOE-funded Earth Systems Grid (www.earthsciencegrid.org) projects.

Resource sharing in scientific communities

The instances of resource sharing provided above give a flavor for what Grid computing is about. However, the implications for the working scientist go far beyond these simple scenarios, as I illustrate with five examples representing different usage modalities.

Science portals. We are accustomed to negotiating a steep learning curve when installing and using a new software package. Science Portals reduce barriers to the use of advanced problem solving methods by allowing sophisticated packages to be invoked remotely, from Web browsers or other simple, easily downloadable "thin clients." Packages can themselves run on suitable computers within a Grid. Such portals are currently being developed in biology, fusion, computational chemistry, and other disciplines.

Distributed computing. High-speed workstations and networks can make the personal computers and clusters of an organization or scientific collaboration, in the aggregate, a substantial computational resource. Such resources are being are being exploited within biology, with Entropia Inc.'s FightAIDSAtHome system harnessing more than 30,000 computers to analyze AIDS drug candidates. In computational mathematics, mathematicians recently solved a long-open problem in numerical optimization, NUG30 by pooling resources across the U.S. and Italy. They were able to bring an average of 630—and a maximum of 1006—computers to bear for a week, delivering a total of 42,000 CPU-days. Future improvements in network performance and Grid technologies will increase the range of problems for which such aggregated computing resources are applicable.

Large-scale data analysis. Many interesting scientific problems require the (often compute-intensive) analysis of large amounts of data. Here, the ability to harness distributed compute and storage resources can be of great value. Furthermore, the naturally parallelism inherent in many data analysis procedures makes it feasible to use distributed resources efficiently. For example, analysis of the many petabytes of data to be produced by future high energy physics experiments will require the harnessing of tens of thousands of processors and hundreds of terabytes of disk space for holding intermediate results. For various technical and political reasons, assembling these resources at a single location appears impractical. Yet the collective institutional and national resources of the hundreds of institutions participating in those experiments can provide these resources. These communities can, furthermore, share more than just computers and storage: by federating databases, they can also share analysis procedures and computational results.

Computer-in-the-loop instrumentation. Scientific instruments such as telescopes, microscopes, X-ray sources, and sensor nets generate raw data streams that are today either interpreted visually or archived for subsequent batch processing. Yet quasi-real-time analysis can greatly enhance an instrument's capabilities. For example, consider an astronomer using a radio telescope array to study solar flares. The deconvolution and analysis algorithms used to process the data and detect flares are computationally demanding, and solar flares vary in seconds. If the astronomer can call upon substantial computing resources (and sophisticated software) in an on-demand fashion, she can use automated detection techniques to "zoom in" on solar flares as they occur.

Collaborative work. Researchers often want also to aggregate human expertise. Collaborative problem formulation, data analysis, and the like become important applications in Grid environments. For example, an astrophysicist interested in

collaborative analysis of large datasets wants to be able to fly through, invoke automatic feature detection algorithms on, annotate, and discuss the results of large (multi-terabyte) simulation datasets—in collaboration with colleagues around the world.

While these examples illustrate selected aspects of the Grid problem, real applications will frequently contain aspects of several of these—and other—scenarios. For example, our radio astronomer might also want to look for similar events in an international archive, discuss results with colleagues during a run, and/or invoke distributed computing runs to evaluate alternative "what if" scenarios.

The architecture of Grid infrastructure

Close to a decade of focused technology R&D and application experimentation (see **Box 1**) has produced considerable consensus on Grid technology requirements and architecture, and in particular the importance of standard *protocols* as a means of achieving interoperability among different Grid resources and services, and hence constructing large-scale Grid systems; and standard *application programming interfaces* (APIs) to facilitate the construction of Grid components and applications. (Protocols define the content and sequence of message exchanges used to request remote operations, and are vital to interoperability; APIs define standard interfaces to code libraries, thus allowing code components to be reused.)

In addition, recent work is revealing close relationships between the concerns and technologies of science and commercial computing: see **Box 2**.

Protocols and APIs can be categorized according to the role that they play in a Grid system (Figure 2). At the lowest level, the Fabric, we have the physical devices or *resources* that Grid participants want to share and access, including computers, storage systems, catalogs, networks, and various forms of sensor.

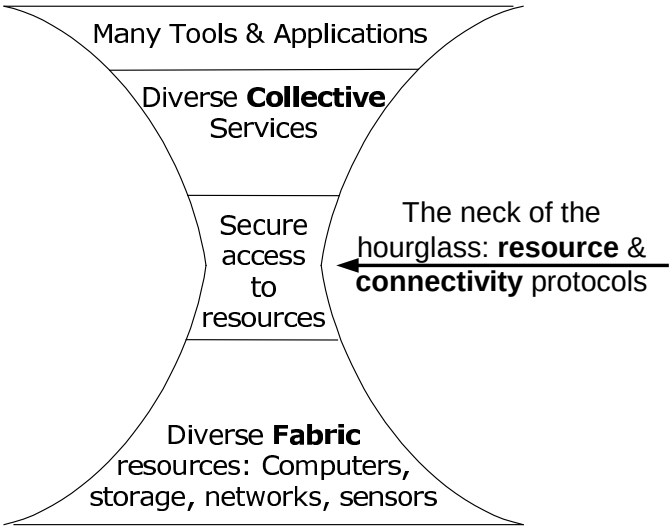

Figure 2: A schematic view of Grid architecture

Above this, the Connectivity and Resource layers define the "neck" in the Grid protocol hourglass: the protocols that (like the Internet Protocol) must be implemented everywhere, and that can be used to implement different application behaviors. The Connectivity layer defines core communication and authentication protocols required for Grid-specific network transactions. Communication protocols enable the exchange of data between resources; authentication protocols build on communication services to provide cryptographically secure mechanisms (discussed below in more detail) for verifying the identity of users and resources. The Resource layer builds on these

communication and authentication protocols to define protocols for the secure initiation, monitoring, and control of sharing operations on individual resources: for example, secure access to remote computers for the purpose of initiating, monitoring, and controlling computations, so that a program can be run on any of a number of different computer systems. The Globus Toolkit™[1] (www.globus.org; and see **Box 3**) is a commonly used source of Connectivity and Resource protocols and APIs.

The Collective layer contains protocols, services, and APIs that implement interactions across collections of resources. Because Collective components build on the narrow Resource and Connectivity layer "neck" in the protocol hourglass, they can implement a variety of sharing behaviors without placing new requirements on resources. Examples of Collective services include directory and brokering services, for resource discovery and allocation; monitoring and diagnostic services; data replication services; and membership and policy services for keeping track of who in a community is allowed to access resources.

Finally, user applications are constructed in terms of, and by calling upon, services defined at any other layer. For example, a high energy physics analysis application that needs to execute several thousands of independent tasks, each taking as input some set of files containing events, might proceed by:

- obtaining necessary authentication credentials (Connectivity layer protocols: see below);
- querying an information system and replica catalog to determine availability of computers, storage systems, and networks, and the location of required input files (Collective services);
- submitting requests to appropriate computers, storage systems, and networks to initiate computations, move data, and so forth (Resource protocols); and, finally
- monitoring the progress of the various computations and data transfers, detecting and responding to failure conditions, and notifying the user when all are completed (Resource protocols).

Many of these capabilities can be encapsulated in tools that automate the more complex tasks—for example, the Condor-G system from the University of Wisconsin (www.cs.wisc.edu/condor).

Authentication, authorization, and policy

Authentication, authorization, and policy are among the most challenging issues in Grids, so I provide some brief technical details to illustrate the types of concern that arise.

"Traditional" security technologies are concerned primarily with securing client-server interactions, in which a client (user) and server need to mutually authenticate (i.e., verify each other's identity) and the server needs to determine whether it wishes to authorize requests issued by the client. Sophisticated technologies have been developed for performing these basic operations and for guarding against and/or detecting various

[1] Globus Project and Globus Toolkit are trademarks of the University of Chicago and University of Southern California.

forms of attack. We use these technologies when we access a remote computer using the popular "Secure Shell" utility or access a secure ecommerce web site.

In Grid environments, the situation is more complex. The distinction between client and server tends to disappear (a Grid is what is sometimes termed a *peer-to-peer* system), because an individual resource can act as a "server" one moment (as it receives a request) and a "client" another (as it issues requests to other resources). For example, when I request that a simulation code be run on a colleague's computer, I am the client and the computer is a server. But a few moments later, that same code and computer act as a client, as they issue requests—on my behalf—to other computers to access input datasets and/or to run subcomputations. A number of interesting requirements arise in this context.

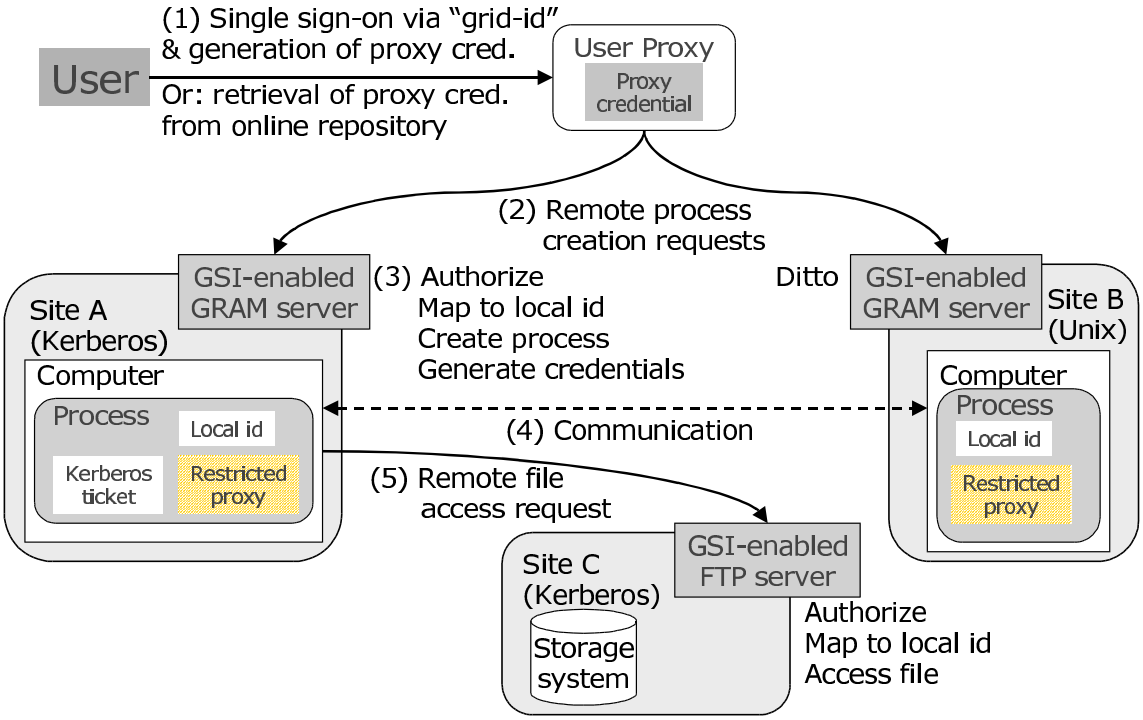

Figure 3: The authentication and authorization steps involved when a user uses Grid services to authenticate (1) and to request that subcomputations (processes) be started at two sites A and B (2), which then communicate with each other (4) and also read files located at a third site, C (5).

Single sign on. A single computation may access many resources and it is generally unacceptable to require that the user re-authenticate (by, for example, typing in a password) on each occasion. Instead, a user should be able to authenticate once and then assign to the computation the right to operate on their behalf, typically for a specified period of time. This capability is achieved via the creation of a *proxy credential* that the computation can use to authenticate, as the user, with remote resources. For example, in Figure 3, the program run by the user (the User Proxy) uses a proxy credential to authenticate twice, with so-called GRAM ("Grid Resource Access and Management") services at sites A and B. These services handle requests to create new processes.

Mapping to local security mechanisms. Different sites may employ different local security solutions (e.g., "Kerberos" and "Unix" in Figure 3). A Grid security infrastructure needs to map to these local solutions at each site, so that local operations performed can proceed with appropriate privileges. Hence, for example, in Figure 3, processes execute under a "local id" and, at site A, are assigned a Kerberos ticket, a credential used by the Kerberos system for authorization.

Delegation. The creation of a proxy credential is a form of *delegation* [6], an operation of fundamental importance in Grid environments. A computation that spans many resources creates subcomputations that may themselves generate requests to other resources and services, perhaps creating additional subcomputations, and so on. For example, in Figure 3, the two subcomputations (processes) created at sites A and B subsequently both communicate with each other and access files at sites C. Authentication operations, and hence further delegated credentials (the "Restricted Proxies"), are involved at each stage, as resources determine whether to grant requests and computations determine whether resources are trustworthy. Yet the further these delegated credentials are disseminated, the greater the risk that they will be acquired and misused by an adversary. These delegation operations and the credentials that enable them must be carefully managed.

Community authorization and policy. In a large community, the policies that govern who can use which resources for what purpose cannot be based directly on individual identity: it is infeasible for each resource to keep track of community membership and privileges. Instead, resources (and users) need to be able to express policies in terms of other criteria, for example group membership, which can be expressed in terms of possession of a cryptographic credential (a capability) issued by a trusted third party. For example, in the scenario depicted in Figure 3, the file server at site C must know explicitly whether or not this particular user is allowed to access a particular file. A community authorization system allows this policy decision to be delegated to a community representative.

Current Status and Future Directions

We now have considerable consensus on Grid architecture principles and the technologies required for basic Grid operations, with in particular the community based, open source Globus Toolkit being applied by most major Grid projects and also seeing significant industrial adoption. Major science communities accept that this technology is important for their future. Numerous government funded R&D projects are variously developing core technologies, deploying production Grids, and/or applying Grid technologies to challenging applications (see http://www.mcs.anl.gov/~foster/grid-projects for a list of major projects). Industrial interest is growing: for example, a total of 12 companies announced support for the Globus Toolkit in late 2001. Progress has been made on organizational fronts: the Global Grid Forum (www.gridforum.org) is a significant force for standards setting and community development, with over 1000 people on its mailing lists. Its thrice-yearly meetings attract hundreds of attendees from some 200 organizations. The International Virtual Data Grid Laboratory is being established as an international Grid system (Figure 4).

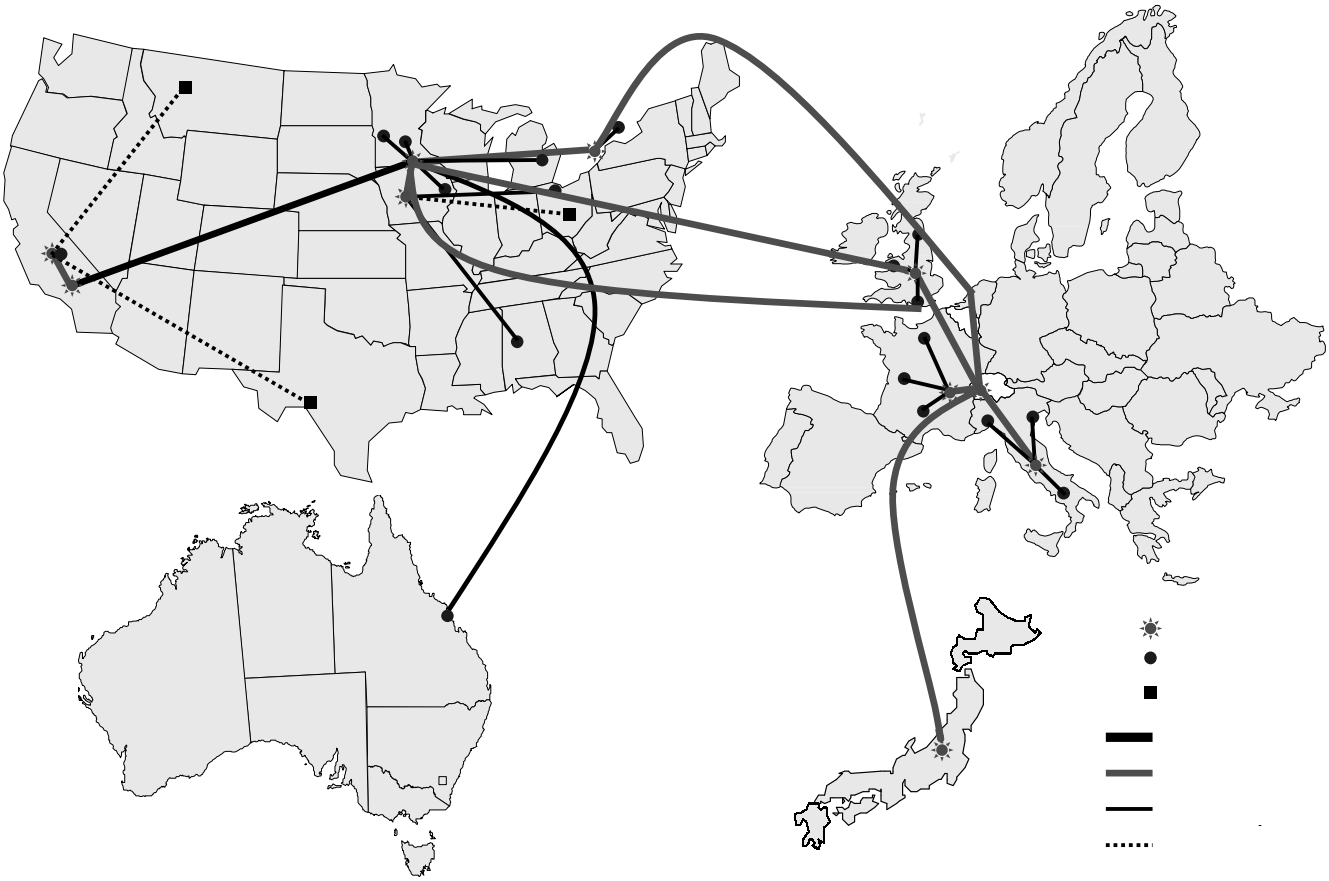

Figure 4: Initial sites involved in the International Virtual Data Grid Laboratory (www.idgl.org)

It is commonly observed that people overestimate the short-term impact of change but underestimate long-term effects [10]. It will surely take longer than some expect before Grid concepts and technologies transform the practice of science, engineering, and commerce, but the combination of exponential technology trends and R&D advances noted in this article are real and will ultimately have dramatic impacts. In a future in which computing, storage, and software are no longer objects that we possess, but utilities to which we subscribe, the most successful scientific communities are likely to be those that succeed in assembling and making effective use of appropriate Grid infrastructures—and thus accelerating the development and adoption of new problem solving methods within their discipline.

BOX 1: Historical Origins

While Grid concepts date to the earliest days of computing, much current Grid research and development owes it genesis to pioneering work conducted on early experimental high-speed networks, such as the U.S. Gigabit testbeds established in the early 1990s [1]. For example, on the Casa network that connected four laboratories in California and New Mexico, Paul Messina and his colleagues developed and demonstrated early "metacomputing" applications, coupling massively parallel and vector supercomputers

for computational chemistry, climate modeling, and other scientific applications. On Blanca, which connected sites in the midwest, Charlie Catlett and his colleagues did early work oriented toward multimedia digital libraries and demonstrated the potential of distance visualization. Two other testbeds investigated remote instrumentation. The gigabit testbeds also experimented with wide area communication libraries and high bandwidth communication protocols. Similar testbeds were created elsewhere, for example in Germany.

Within the U.S. at least, the event that moved Grid concepts out of the network laboratory and into the consciousness of ordinary scientists was the I-WAY experiment [2]. Led by Tom DeFanti and Rick Stevens, this ambitious effort linked 11 experimental networks to create, for a week in November 1995, a national high-speed network infrastructure that connected resources at 17 sites across the U.S. and Canada (Figure 5). Some 60 application demonstrations, spanning the gamut from distributed computing to virtual reality collaboration, showed the potential of high-speed networks. The I-WAY also saw the first attempt to construct a unified software infrastructure for such systems, namely the I-Soft system developed by myself and others that provided single sign, unified scheduling, and other services that allowed the I-WAY to be treated, in some important respects, as an integrated *infrastructure*.

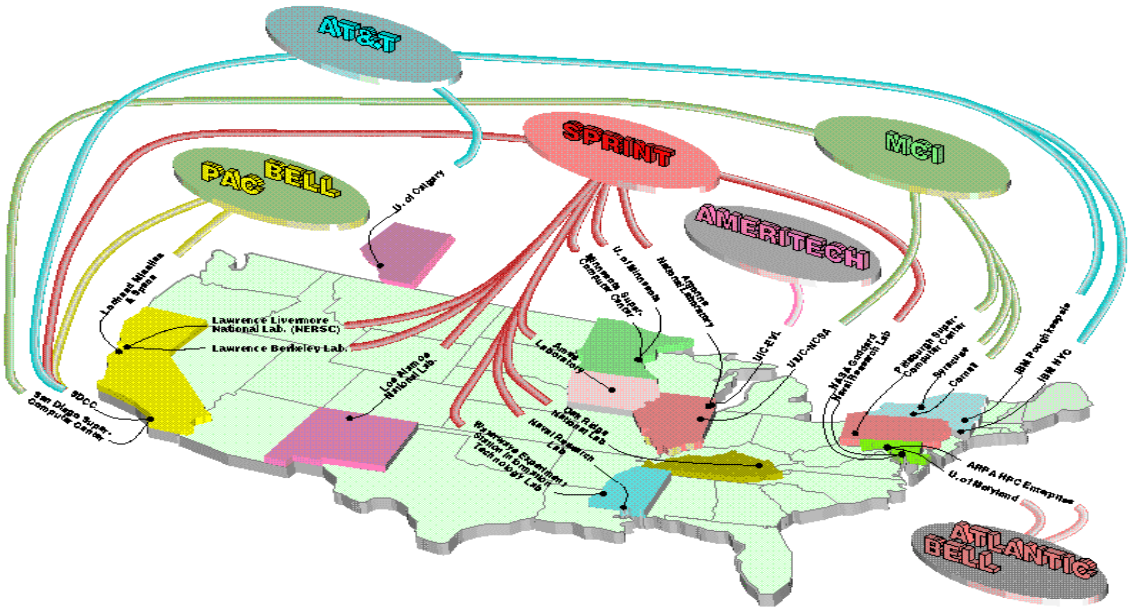

Figure 5: The I-Way network constructed in November 1995, showing the principal networks and sites connected.

BOX 2: Commercial Grids and the Open Grid Services Architecture

Grid concepts are becoming increasingly relevant to commercial information technology (IT). With the rise of ebusiness and IT outsourcing, enterprise applications no longer run exclusively within the friendly confines of an enterprise central computing facility. Instead, they must operate on heterogeneous collections of resources that may span multiple administrative units within a company as well as various external networks and service providers. In commerce as in science and engineering, we face the same need to deliver quality of service within dynamic virtual organizations.

One consequence of this convergence of interests is a growing interest in the integration of previously distinct commercial and Grid technologies. For example, the Open Grid Services Architecture [4] proposed by the Globus Project and IBM's Open Service Architecture group integrates Grid mechanisms into the Web services architecture that is at the core of major industrial distributed computing technologies such as Microsoft's .NET, IBM's Websphere, and Sun's Java 2 Enterprise Edition (J2EE) [7].

Web services define standards for defining, and invoking operations on, remote services: in particular, the Web Service Description Language (WSDL) for describing Web services, the Simple Object Activation Protocol (SOAP), a remote procedure call protocol for invoking services, and UDDI, Universal Description, Discovery, and Integration, for defining directories that can be used to discover specific services. Grid protocols and services address complementary issues relating, for example, to the secure and reliable creation and management of dynamic service instances—as is required for remote computing and resource sharing scenarios such as those described in this article.

The Open Grid Service Architecture leverages these two technologies to define, among other things, standard behaviors (and WSDL interfaces) for a "Grid service": a Web service that can be created dynamically and that supports security, lifetime management, manageability, and other functions required in Grid scenarios. These features are being incorporated into the Globus Toolkit, and will likely also appear in commercial products.

Box 3: The Globus Toolkit

The Globus Toolkit (www.globus.org) is a community-based, open-architecture, open-source set of services and software libraries that support Grids and Grid applications. The Toolkit includes software for security, information infrastructure, resource management, data management, communication, fault detection, and portability. It is packaged as a set of components that can be used either independently or together to develop applications. Components include:

- the Grid Security Infrastructure (GSI), which addresses Grid security concerns reviewed in the text, providing in particular a single-sign-on, run-anywhere authentication service, with support for delegation of credentials to subcomputations, local control over authorization, and mapping from global to local user identities;
- the Grid Resource Access and Management (GRAM) protocol and service, which provides remote resource allocation and process creation, monitoring, and management services;

- the Meta Directory Service (MDS), an extensible Grid information service that provides a uniform framework for discovering and accessing system configuration and status information such as compute server configuration, network status, or the locations of replicated datasets; and
- Data Grid-specific technologies, including a replica catalog, for locating copied of data; GridFTP, a high-speed data movement protocol; and reliable replica management tools.

For each component, the Toolkit both defines protocols and APIs and provides open source reference implementations in C and (for client side APIs) Java. A tremendous variety of higher-level services, tools, and applications have been implemented in terms of these basic components. Some of these services and tools are distributed as part of the Toolkit, while others are available from other sources. The NSF-funded GRIDS Center (www.grids-center.org) maintains a repository of components.

The Globus Toolkit is a product of the Globus Project, led by myself and Carl Kesselman at the University of Southern California's Information Sciences Institute. Steve Tuecke plays an important role as the Chief Architect at Argonne National Laboratory. Many other project participants and collaborators, listed at www.globus.org, have made major contributions. In addition, a number of companies are now starting to offer commercial support for the Toolkit.

Acknowledgements: I am grateful to David Abramson, Paul Avery, Fabrizio Gagliardi, Tony Hey, Satoshi Matsuoka, and Harvey Newman for their comments on a draft of this article. My research is supported, in part, by grants from the U.S. Department of Energy, National Science Foundation, DARPA, NASA, and Microsoft.

Biographical note: Ian Foster is a senior scientist and associate division director in Argonne National Laboratory's mathematics and computer science division, professor in the University of Chicago's department of computer science, and senior fellow in the Argonne-Chicago Computation Institute. He serves on the advisory boards of several companies, including Entropia, Inc.

References

1. Catlett, C. In Search of Gigabit Applications. *IEEE Communications Magazine* (April). 42-51. 1992.
2. DeFanti, T., Foster, I., Papka, M., Stevens, R. and Kuhfuss, T. Overview of the I-WAY: Wide Area Visual Supercomputing. *International Journal of Supercomputer Applications, 10* (2). 123-130. 1996.
3. Foster, I. and Kesselman, C. (eds.). *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1999.
4. Foster, I., Kesselman, C., Nick, J. and Tuecke, S. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Argonne National Laboratory, 2002, www.globus.org/research/papers/physiology.pdf.
5. Foster, I., Kesselman, C. and Tuecke, S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance*

Computing Applications, *15* (3). 200-222. 2001. www.globus.org/research/papers/anatomy.pdf.

6. Gasser, M. and McDermott, E., An Architecture for Practical Delegation in a Distributed System. In *Proc. 1990 IEEE Symposium on Research in Security and Privacy*, (1990), IEEE Press, 20-30
7. Graham, S., Simeonov, S., Boubez, T., Daniels, G., Davis, D., Nakamura, Y. and Neyama, R. *Building Web Services with Java: Making Sense of XML, SOAP, WSDL, and UDDI*. Sams, 2001.
8. Kleinrock, L. The Latency/Bandwidth Tradeoff in Gigabit Networks; Gigabit Networks are Really Different! *IEEE Communications Magazine, 30* (4). 36-40. 1992.
9. Leiner, B.M., Cerf, V.G., Clark, D.D., Kahn, R.E., Kleinrock, L., Daniel C. Lynch, Postel, J., Roberts, L.G. and Wolff, S. A Brief History of the Internet. 2000. http://www.isoc.org/internet-history/brief.html.
10. Licklider, J.C.R. *Libraries of the Future*. MIT Press, 1965.
11. Licklider, J.C.R. and Taylor, R.W. The Computer as a Communication Device. *Science and Technology* (April). 1968. http://memex.org/licklider.pdf.
12. National Research Council *National Collaboratories: Applying Information Technology for Scientific Research*. National Academy Press, Washington, DC, 1993.
13. Teasley, S. and Wolinsky, S. Scientific Collaborations at a Distance. *Science, 292*. 2254-2255. 2001.
14. Thomas, G.A., David A. Ackerman, Prucnal, P.R. and Cooper, S.L. Physics in the Whirlwind of Optical Communications. *Physics Today, 53* (9). 2000.
15. Vyssotsky, V.A., Corbató, F.J. and Graham, R.M., Structure of the Multics Supervisor. In *Fall Joint Computer Conference*, (1965). http://www.multicians.org/fjcc3.html