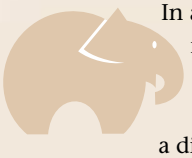
IAN FOSTER, ARGONNE NATIONAL LABORATORY and
UNIVERSITY OF CHICAGO
STEVEN TUECKE, UNIVA

Describing the Elephant:

The Different Faces of IT as Service

Terms such as *grid, on-demand,* and *service-oriented architecture* are mired in confusion, but there is an overarching trend behind them all.



In a well-known fable, a group of blind men are asked to describe an elephant. Each encounters a different part of the animal and, not surprisingly, provides a different description.

We see a similar degree of confusion in the IT industry today, as terms such as *service-oriented architecture, grid, utility computing, on-demand, adaptive enterprise, data center automation*, and *virtualization* are bandied about. As when listening to the blind men, it can be difficult to know what reality lies behind the words, whether and how the different pieces fit together, and what we should be doing about the animal(s) that are being described. (Of course, in the case of the blind men, we did not also have marketing departments in the mix!)

Our goal in this article is to provide, in effect, a description of the elephant. More specifically, we describe what we see as a major technology trend that is driving many related efforts—namely, the transformation from vertically integrated silos to horizontally integrated, service-oriented systems. We explain how various popular terms relate to this overarching trend and describe the technology required to realize this transformation.

THE NEED FOR HORIZONTAL INTEGRATION

Your IT department is asked to develop a new application for computing valuations of customer stock portfolios. They start the process of requirements analysis, design, development, hardware acquisition, deployment, and testing. Some months later, the new application is ready: a finely tuned, vertically integrated stack of specialized application and management software running on a dedicated set of servers.

Describing the Elephant:

The Different Faces of IT as Service

Over time, this application becomes more popular, and major bursts of demand lead to overload. While there is considerable excess server capacity throughout the enterprise, it cannot be brought to bear on these bursts, because other application stacks have those servers locked up. (The servers used by each application run quite different software—perhaps even different operating systems—and are statically configured to perform a single task. And no one wants their servers touched by anyone else, as any change might destabilize their applications.) Thus, the IT department must continue to install additional servers to increase capacity.

This organization of IT resources as a set of more-or-less independent *silos*, each responsible for a distinct enterprise function or application, is commonplace. Indeed, it is a natural consequence of both decentralization and the proprietary resource virtualization, workload management, and application builders used to construct applications today. This isolation is also becoming increasingly untenable, however, as a result of business imperatives both to reduce capital and operation expenses and to respond more rapidly to business demands. It is not uncommon for individual silos to be idle 90 percent of the time because of the need to provide excess capacity for occasional peak loads. Furthermore, each distinct silo needs specialized operations skills.

In seeking solutions to these problems, we need not look far into the past. Not long ago, the mainframe provided both a convenient set of abstractions to which applications could code and powerful resource management functions that allowed many applications to share available resources within that mainframe, all while maintaining appropriate qualities of service. Mainframes would commonly serve dozens of applications effectively while running at close to 100 percent capacity. In effect, centralized IT architectures were *decoupled vertically* and *integrated horizontally*, thus allowing reuse of function within applications and economies of scale in terms of resource usage. Furthermore, all applications were accessible within a uniform environment.

As illustrated in figure 1, the move to distributed, low-cost, and often heterogeneous collections of servers, despite the many benefits, has nonetheless led to this centralized IT architecture disintegrating into isolated silos. The challenge now is to reintegrate, so that the benefits of vertical decoupling and horizontal integration can be achieved in the more complex modern distributed enterprise IT environment. In this context, vertical decoupling means that we standardize interfaces among application components, workload management systems, and physical resources so that different components can be assembled dynamically to meet application needs. Horizontal integration means that we adopt uniform management interfaces so that large numbers of resources, distributed over what used to be distinct silos, can be allocated, used, monitored, and managed in a common and automated manner, improving utilization and reducing operations costs.

To see what these ideas can mean in practice, consider how we would implement the previously described portfolio valuation application in a horizontally integrated world. We would code our application to a workload management interface that allows us to define the

Evolution of Enterprise IT Architecture

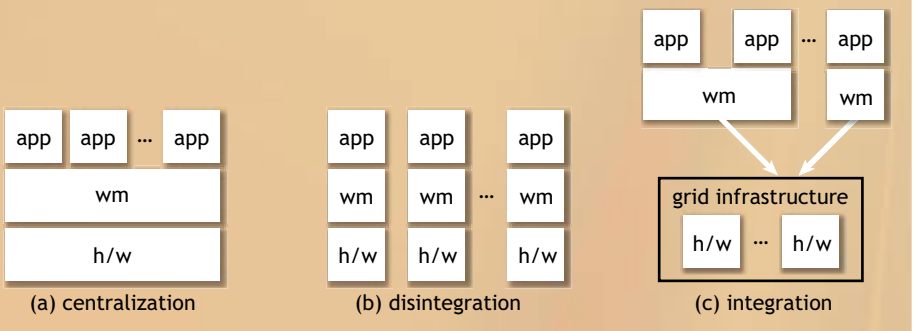


FIG 1

The boxes represent applications (app), workload management (wm) functions, and hardware (h/w).

activities to be performed and our performance requirements for those activities. The workload manager that implements this interface would then use common grid infrastructure mechanisms (deployed on all enterprise hardware) to discover available resources and deploy application components onto those resources. As load increases, new resources could be automatically (and temporarily) allocated to, and used by, the application. Thus, the "application" is totally decoupled from the underlying "hardware."

THE BLIND MEN REPORT

We now turn to the various terms mentioned in the introduction and examine how each relates to the goal of horizontal integration.

The term *grid* is one that we have been particularly involved in promulgating, although it has also caught the fancy of many marketers in recent years. According to one definition, a grid is "a system that uses open, general-purpose protocols to federate distributed resources and to deliver better-than-best-effort qualities of service."[1] This use of the term is inspired by the electric power grid, which as a technology both: (a) implements standards for electric power transmission that allow for the decoupling of consumer and provider; and (b) links diverse providers into a managed utility. By analogy, grid technologies enable: (a) on-demand access to computing capabilities; and (b) the federation of distributed resources and the management of those distributed resources to meet end-user requirements.[2] Thus, *grid* is a big-picture term used to describe solutions relating to the flexible use of distributed resources for a variety of applications—as well as a term that emphasizes the importance of standards to interoperability.

We use the term *grid infrastructure* to refer to a particularly important aspect of the grid space—namely, a horizontal infrastructure integration layer. The term *grid* is also often applied to other layers of the stack: for example, to characterize an application or workload manager that has been structured to make efficient and flexible use of distributed and/or shared resources.

The related term *utility computing* is often used to denote both a separation between service provider and consumer and the ability to negotiate a desired quality of service from the provider—two properties typically associated with utilities such as electricity. The provider may be an organization's IT department or an external utility provider, and the service may be storage, computing, or an application. (For example, Sun offers a pay-as-you-go "computing utility" service, at $1 per CPU-hour, while salesforce.com offers a customer relationship management application as a service.)

On-demand is a broad term used to denote technologies and systems that allow users or applications to acquire additional resources to meet changing requirements. Thus, we may have on-demand computing, storage, bandwidth, and applications. The meaning overlaps significantly with utility computing: *storage utility* and *on-demand storage* have similar meanings.

The terms *utility, on-demand*, and *grid* overlap significantly in meaning. All have a connotation of IT as service, meaning that they involve providing access to physical resources or other services over the network via some standardized protocol. Utility and on-demand can be thought of as specialized use cases for grid (see figure 2)—although certainly many of today's utility and on-demand products do not yet use grid infrastructure. Indeed, an inhibitor to adoption of utility/on-demand computing is the lack of standards for horizontal infrastructure integration. In the absence of standard mechanisms for discovering, negotiating access to, configuring, and managing remote resources, utility/on-demand services often involve complex manual configuration. This discourages software vendors and end users from developing software that depends on the use of those services.

Also related is the term *data center automation*, commonly used to refer to products that enable the coordinated management of resources within an enterprise: for example, to keep a large number of machines up to date with the latest patches. The goal is the automation of operations on applications that are typically not modified for distributed execution. This task typically uses special-

View of the Relationship among Grid, Utility, and On-Demand Computing

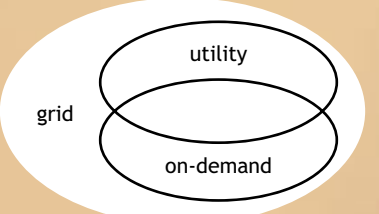


FIG 2

Describing the Elephant:

The Different Faces of IT as Service

ized underlying management infrastructure, but it could be facilitated by a common grid infrastructure.

The term *cluster* (sometimes *Beowulf cluster*, after an early project) denotes a nonshared-memory, multi-CPU system constructed from commodity parts. The relatively low cost of clusters makes them excellent powerplants for grid/utility/on-demand computing systems, particularly as better integration reduces their space, power, and administration costs, and virtual machine technology facilitates more flexible provisioning.

Other terms tend to have both broader and more parochial definitions. Thus, IBM talks about an *on-demand business* as "an enterprise whose business processes—integrated end-to-end across the company and with key partners, suppliers, and customers—can respond with speed to any customer demand, market opportunity or external threat." Hewlett-Packard's term *adaptive enterprise* has a similar connotation, denoting "an organization that adapts to market conditions so that it can respond to and address changes in their market, its environment, and/or its industry to better position itself for survival and profitability."

Also relevant is *outsourcing*, in which a third party offers to run all or part of an enterprise's IT operations. Outsourcing is more a financial strategy than an IT architecture: in many outsourcing deals, IT staff change employers but applications remain hosted on the same silos. If outsourcing firms have the opportunity to modify applications, however, then their focus on efficient operations can make horizontal integration attractive as a means of reducing hardware costs. In addition, successful realization of utility approaches can enable more dynamic outsourcing of resources to meet varying workload demand.

Another recent term is *software as service*. In this approach, the Web is used to provide many customers with access to functions (customer relationship management in the case of salesforce.com) that have been specifically designed for this mode of use. Software as service is thus an approach to writing applications and exposing interfaces to users (e.g., through Web browsers). This approach can make good business sense because the software-as-service vendor can achieve economies of scale. The software-as-service vendor has a high degree of control over its application and is thus a prime candidate for the adoption of horizontal infrastructure integration strategies.

No discussion of big-picture concepts would be complete without introducing *SOA (service-oriented architecture)* and *Web services*. These two terms denote a set of architectural principles and an implementation technology, respectively, that play an important role in realizing IT as service—and, as we discuss later, horizontal integration.

SOA denotes an approach to designing systems that facilitates the realization of the IT-as-service and horizontal integration goals mentioned earlier. A service is a self-contained implementation of some function(s) with a well-defined interface specifying the message exchange patterns used to interact with the function(s). An SOA, then, is a set of services. SOAs thus seek to achieve the clean separation of interface and implementation needed to realize other desirable properties such as interoperability, location transparency, and loose coupling between service and client. (For a discussion of the merits of loose coupling, see Kendall et al.'s classic paper.[3])

Web services are a set of technologies for realizing service-oriented architectures. While not the only technology that can be used for this purpose—for example, CORBA and DCOM have been used in the past—Web services have technical advantages over prior approaches and are being widely adopted.

Web services are defined by a core set of technical specifications that codify mechanisms for describing the set of messages (i.e., interface) that can be used to interact with a service (WSDL) and for encoding messages to/from services (SOAP). Other specifications define how to secure access to services (WS-Security), address services (WS-Addressing), manipulate state (WS-Resource Framework), deliver notifications (WS-Notification), and so forth. The use of XML to describe service interfaces and encode messages facilitates integration of applications and distributed systems from independently defined and loosely connected services.

Technologies for building Web services are by now reasonably mature, with many companies offering good commercial Web services stacks, and Apache providing a solid open source stack. It is important to recognize, however, what these systems do and do not do. They do provide tooling for developing Web services, typically in Java or C#, and containers for hosting those services. They do not, for the most part, address the question of

how to provide a common set of abstractions and interfaces for effective use of IT resources—a key requirement for realizing horizontal integration. The development of these abstractions and interfaces is well under way, but not yet fully completed, as we discuss later.

The term *service-oriented infrastructure* describes the use of SOA approaches to the problem of resource management. Exposing resources as services is an important step forward in that it enables a more uniform treatment of diverse components. As we discuss in the grid infrastructure section later, however, the larger goal of horizontal integration requires more than just the adoption of Web services technologies, which if applied without coordination results in an unstructured mix of partially overlapping, vendor-specific Web services interfaces to resources. We also require commonality of abstractions and interfaces across the broad array of components, so that, for example, a workload manager can allocate, reserve, and manage computing, storage, and network resources from different vendors in common ways.

HORIZONTAL INTEGRATION

We now turn to the nuts and bolts of how service-oriented and horizontally integrated systems are architected and constructed, and the technologies available for building them.

We structure this discussion around figure 3, which expands on figure 1 to show the principal layers involved in a horizontally integrated, service-oriented enterprise IT architecture. In brief, *applications* use *workload managers* to coordinate their access to physical resources. An application and its workload manager are not, as is the case in vertically integrated silos, tightly bound to a single physical resource (or set of physical resources). Instead, they bind dynamically to resources (i.e., are *provisioned*) via a common *grid infrastructure* layer. The resources themselves may implement various *virtualization* approaches to enhance the flexibility with which they serve their users.

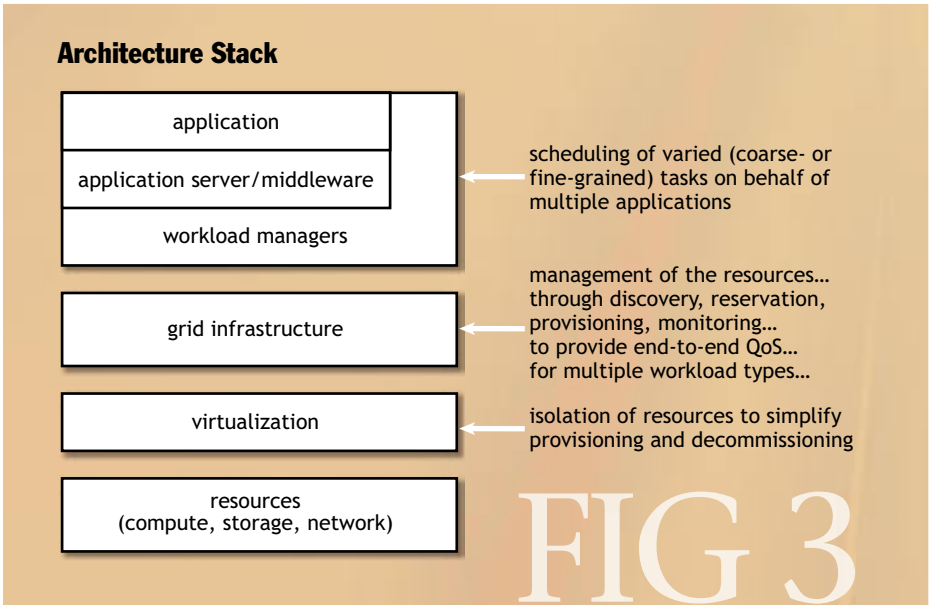
APPLICATION

Enterprise applications span a broad spectrum: they may be coarse- or fine-grained, batch or interactive, compute- or data-intensive, transactional or not. They may be implemented in a variety of ways: indeed, application development and development tools, service-oriented or otherwise, have become a huge industry. The primary concern in this article is not how applications are developed but rather how an application's execution requirements are mapped to physical resources. In the mainframe era, the operating system handled this task. In today's loosely coupled clusters and distributed systems, operating systems no longer provide the necessary support. Thus, application mapping functions must be embedded directly into applications or—a more modular approach—be provided by separate workload managers, as discussed in the next section.

The term *grid application* is often used to refer to applications that have been adapted to use a distributed infrastructure. Such applications have typically been parallelized and written to accommodate the dynamic addition and removal of physical resources. Application programming environments that help programmers write such applications include vendor products from Data Synapse and United Devices; the Message Passing Interface library; workflow systems; and open source systems such as Condor and Nimrod.

WORKLOAD MANAGEMENT

The workload manager has emerged as an important product category in enterprise IT. There are a number of such systems, each specialized to different execution task granularities, transaction characteristics, and performance



Describing the Elephant:

The Different Faces of IT as Service

requirements. For example, batch schedulers such as those from Altair, Platform, Sun, and United Devices—and open source systems such as Condor—run relatively coarse-grained tasks in domains such as digital rendering, engineering analysis, and drug design. Other applications, such as trading applications in financial services, require support for workloads consisting of many fine-grained transactional tasks and may use workload managers from Data Synapse, Gigaspaces, Platform, or others. Yet other applications require support for workflow, data flow, and/or orchestration: via, for example, BPEL (Business Process Execution Language). Oracle's 10G database harnesses clustered computers for database applications.

The term *grid* is often applied to workload managers that target shared and/or distributed resources. Indeed, most existing grid products are concerned with facilitating the writing of parallel or distributed applications and with managing the execution of these decomposed applications on distributed resources. As we discuss in the next section, however, a true grid able to support a range of applications on shared resources needs something more.

GRID INFRASTRUCTURE

Enterprises that want to support a range of applications on shared resources face the problem that different applications and their associated workload managers do not integrate at the infrastructure level. Instead, each relies on a proprietary infrastructure management function. Thus, the enterprise with a variety of application requirements ends up with a collection of silos, each consisting of a distinct application, workload manager, and set of resources—the situation depicted in figure 1.

The solution to this problem is to introduce a common horizontal layer that defines and implements a *consistent set of abstractions and interfaces for access to, and management of, shared resources*. We refer to this horizontal resource integration layer as *grid infrastructure*. This is what enables the horizontal integration across diverse physical resources that we require to decouple application and hardware. This grid infrastructure layer is the focus of Globus software,[5] discussed later.

A grid infrastructure must provide a set of technical capabilities, as follows:

- **Resource modeling.** Describes available resources, their capabilities, and the relationships between them to facilitate discovery, provisioning, and quality of service management.
- **Monitoring and notification.** Provides visibility into the state of resources—and notifies applications and infrastructure management services of changes in state—to enable discovery and maintain quality of service. Logging of significant events and state transitions is also needed to support accounting and auditing functions.
- **Allocation.** Assures quality of service across an entire set of resources for the lifetime of their use by an application. This is enabled by negotiating the required level(s) of service and ensuring the availability of appropriate resources through some form of reservation—essentially, the dynamic creation of a service-level agreement.
- **Provisioning, life-cycle management, and decommissioning.** Enables an allocated resource to be configured automatically for application use, manages the resource for the duration of the task at hand, and restores the resource to its original state for future use.
- **Accounting and auditing.** Tracks the usage of shared resources and provides mechanisms for transferring cost among user communities and for charging for resource use by applications and users.

A grid infrastructure must furthermore be structured so that the interfaces by which it provides access to these capabilities are formulated in terms of equivalent abstractions for different classes of components. For example, a client should be able to use the same authorization and quality-of-service negotiation operations when accessing a storage system, network, and computational resource. Without this uniformity, it becomes difficult for workload managers and other management systems to combine collections of resources effectively and automatically for use by applications.

These considerations make the definition of an effective grid infrastructure a challenging task. Many of the standards and software systems needed to realize this goal, however, are already in place.

VIRTUALIZATION

The resource mapping and management functions defined by grid infrastructure provide convenient and powerful abstractions of underlying physical resources. Implementing those abstractions in an effective, efficient, and uniform manner, however, can be difficult if resources do not provide appropriate isolation and

control functions. Indeed, the lack of such functions in commodity resources has been a significant obstacle to the successful realization of IT as a service.

The broader availability of *virtualization* technologies represents an important step forward in this regard. Such technologies implement a layer on resources that both provides flexible control of the physical resource abstraction (with respect to performance, for example) and, at the same time, supports multiple virtual instances on the same physical resource with good isolation. While such virtualization technologies have been available for a long time on mainframes and other high-end server platforms, they are only now becoming widely available on the commodity hardware and operating systems that have been increasingly adopted by enterprises.

Execution virtualization technologies such as those provided by the open source Xen and the proprietary VMware, Microsoft Virtual Server, and Virtual Iron make it possible for the grid infrastructure to request the creation of a virtualized execution environment with specified operating system image, resource allocations, and isolation properties.[6] Virtual machine–based systems such as J2EE and .NET can play a similar role, although because they deliver an entirely new resource abstraction, they can be applied only to applications written to that abstraction. Communication virtualization technologies (e.g., VLANs, VPNs) and storage virtualization functions (logical volume managers, etc.) provide similar functions for communications and storage, respectively.

To be truly useful within a horizontally integrated infrastructure, virtualization technologies must provide interfaces to their isolation, control functions at appropriate levels, and employ common abstractions. Not all virtualization solutions meet these requirements today. For example, storage is arguably the area that has made the most progress in terms of virtualization standards, particularly at the lower layers of the stack. Many storage vendors, however, still provide proprietary, vertically integrated storage management stacks, and thus there are still substantial silos around storage—both in terms of different vendors' storage silos, as well as storage management interfaces being distinct from the management interfaces of other resource types.

Furthermore, resource allocation control functions must be exposed through the grid infrastructure layer to allow workload managers to manage end-to-end qualities of services across collections of resources. It is not sufficient for virtualization managers to make localized decisions concerning changes in resource allocation, as is done by many current virtualization products.

THE ROLE OF STANDARDS AND OPEN SOURCE

An effective grid infrastructure must implement management capabilities in a uniform manner across diverse resource types—and, if we are to avoid vendor lock-in, it should do so in a manner that does not involve commitment to any proprietary technology.

As was the case with the Internet and Web, both standards and open source software have important roles to play in achieving these goals. Standards enable interoperability among different vendor products, while open source software allows enterprises to proceed with deployments *now*, before all standards are available. In the case of the Internet and Web, standards have been developed over a period of years within bodies such as the IETF and W3C; concurrently with these activities, open source systems such as BSD Unix and the Apache Web server have spurred Internet and Web adoption, respectively, stimulating an explosion of innovation around common standards that has brought amazing results.

We see a similar synergistic relationship between standards and open source software in the case of grid. The standardization of interfaces for resource modeling, monitoring, notification, allocation, provisioning, and accounting functions is proceeding within bodies such as the Global Grid Forum (Open Grid Services Architecture), OASIS (WS-Resource Framework, WS-Notification, and WS Distributed Management), and DMTF (Common Information Model). Concurrently with these efforts, open source software is implementing not only finalized standards but also other interfaces needed to build usable systems. The availability of this software allows enterprises and ISVs to develop grid solutions now—and accelerates the definition and adoption of standards by reducing barriers to their adoption.

At a more fundamental level, the Web services core standards and software are reasonably solid, with such broadly adopted specifications as WSDL, SOAP, and WS-Security available from myriad IT vendors, as well as in open source form from Apache and other sources. Profiles defined by the WS-Interoperability organization promote interoperability among tools and services provided by different vendors.

At higher levels, good-quality open source software is available and playing an important role in the evolution and adoption of standards. For example, the Globus Toolkit that we have been involved in developing implements a wide range of functionality, primarily at the grid infrastructure level, relating to security, execution management, data management, and monitoring and discovery, as well as the core Web services mechanisms referred to

Describing the Elephant:

The Different Faces of IT as Service

earlier. It was originally developed and applied within research and education settings, but the establishment of Univa Corporation as a source of commercial support and services is accelerating enterprise adoption.

SUMMARY

We have argued that SOA, grid, on-demand, utility computing, software as service, and other related terms all represent different perspectives on the same overall goal—namely, the restructuring of enterprise IT as a horizontally integrated, service-oriented architecture. If successfully realized, that goal will see in-house, third-party, and outsourced applications all operating in a uniform environment, with on-demand provisioning of both in-house and outsourced hardware resources—and also, of course, high degrees of security, monitoring, auditing, and management.

This Holy Grail of open, standards-based, autonomically managed software and dynamically provisioned hardware has certainly not yet been achieved. That does not mean, however, that enterprises cannot start today to create horizontally integrated, service-oriented infrastructures. Solid Web services products allow for the creation of service-oriented applications. Mature commercial and open source virtualization and workload management products and open source grid infrastructure software provide what is needed to create horizontally integrated infrastructure to sit behind those applications. Integration remains more of an exercise for the customers (or their services vendors) than is desirable, but that situation should change as independent software vendors start to grid-enable their products. Meanwhile, progress on further standards is accelerating as experience is gained with deployments and pressure builds from end users for interoperable solutions. Q

ACKNOWLEDGMENTS

We are grateful to Greg Astfalk, Stuart Feldman, and Vas Vasiliadis for comments on a draft of this article. Ian Foster's work is supported in part by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract W-31-109-Eng-38, and by the National Science Foundation.

REFERENCES

1. Foster, I. 2002. What is the grid? A three-point checklist; http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf.
2. Foster, I., Kesselman, C., Nick, J.M., and Tuecke, S. Grid services for distributed systems integration. *IEEE Computer 35* (6): 37-46.
3. Kendall, S.C., Waldo, J., Wollrath, A., and Wyant, G. 1994. A note on distributed computing. Sun Microsystems, Technical Report TR-94-29.
4. Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., and Orchard, D. 2003. Web services architecture. W3C, working draft; http://www.w3.org/TR/2003/WD-ws-arch-20030808/.
5. Foster, I. 2005. A Globus primer; www.globus.org/primer.
6. Rosenblum, M., and Garfinkel, T. 2005. Virtual machine monitors: Current technology and future trends. *IEEE Computer* (May): 39-47.

LOVE IT, HATE IT? LET US KNOW

feedback@acmqueue.com or www.acmqueue.com/forums

IAN FOSTER is associate director of the mathematics and computer science division of Argonne National Laboratory and the Arthur Holly Compton Professor of Computer Science at the University of Chicago. He created the Distributed Systems Lab at both institutions, which has pioneered key grid concepts, developed the widely deployed Globus software, and led the development of grid applications across the sciences. He is also cofounder and chief open source strategist at Univa Corporation. Foster graduated with a B.S. in computer science from the University of Canterbury, New Zealand and a Ph.D. in computer science from Imperial College, United Kingdom.

STEVEN TUECKE is CEO of Univa Corporation. He cofounded the Globus Alliance with Ian Foster and Carl Kesselman, where he was responsible for managing the architecture, design, and development of Globus software, as well as the grid and Web services standards that underlie it. He began his career in 1990 as a software engineer for Foster in the mathematics and computer science division at Argonne National Laboratory, where he helped create the Distributed Systems Laboratory. Tuecke graduated with a B.A in mathematics and computer science from St. Olaf College.