



Search for: _____ within _____

[Search help](#)

[IBM home](#) | [Products & services](#) | [Support & downloads](#) | [My account](#)

developerWorks > Grid computing

developerWorks



Perspectives on grid: Grid computing -- next-generation distributed computing

What's different between grid computing and P2P, CORBA, cluster computing, and DCE?

[Matt Haynos](#) (mph@us.ibm.com)

Program Director, Grid Marketing and Strategy, IBM
27 January 2004

Two major needs have dramatically increased the value of the concept of grid computing in the last few years. A lean economy has forced those with a limited IT budget to more fully utilize their existing computing assets and to become more flexible to respond to rapidly evolving markets by being able to intelligently allocate finite resources to the appropriate business applications. In this first of a series of articles, Matt Haynos provides a cursory analysis of the similarities and differences between grid computing and such distributed computing systems as P2P, CORBA, cluster computing, and DCE.

Grid computing has recently enjoyed an increase in popularity as a distributed computing architecture that is becoming highly suitable for corporate computing. Grid computing solutions are being employed in many areas to address critical business requirements, such as:

- Financial services firms tapping into grid computing to address risk management and compliance
- Automotive manufactures using grid solutions to accelerate product development and increase collaboration
- Oil companies harnessing grid technology to hasten the discovery of oil and increase the odds of successful mining

As grid computing matures, the application of the technology in additional areas will increase.

Grid computing can be differentiated from almost all distributed computing paradigms by this defining characteristic: *The essence of grid computing lies in the efficient and optimal utilization of a wide range of heterogeneous, loosely coupled resources in an organization tied to sophisticated workload management capabilities or information virtualization.* (Note that an organization can span multiple departments, physical locations, and so on. We use term "organization" here in the abstract sense.)

How does the characteristic mentioned in the previous paragraph separate grid computing from other distributed models? That's what we hope to answer in this article by, instead of looking forward, exploring where grids came from, determining how they've matured, and specifically illustrating how grid computing differs from alternative distributed computing solutions such as P2P and CORBA. We'll do this by comparing and contrasting the grid concept with the most popular distributed computing solutions. First, let's understand the value of grid computing.

Why grid computing?

As companies have re-examined their investment in information technology during the last few years, many have come to the conclusion that it is essential to more fully leverage the computing assets they already have. Therefore, the importance of utilization has increased; it has become necessary to squeeze more functionality from a limited IT budget.

Contents:

[Why grid computing?](#)

[The origins of grid computing](#)

[Grid fills a crucial gap](#)

[How grid differs from cluster computing](#)

[Grid or CORBA?](#)

[What about DCE?](#)

[Finally, there's P2P](#)

[Leveraging a world of data](#)

[Resources](#)

[About the author](#)

[Rate this article](#)

Related content:

[A visual tour of Open Grid Services Architecture](#)

[Is Web services the reincarnation of CORBA?](#)

Subscriptions:

[dW newsletters](#)

[dW Subscription
\(CDs and downloads\)](#)

Also, there is a particular appeal to being able to *intelligently* allocate finite resources to the appropriate business applications in a distributed enterprise. This technique offers the corporation a certain flexibility, whether it is in the form of being able to redistribute resources to address new market segments, or to enable business applications to better serve rapidly evolving, existing clientele.

Taking a cue from the manufacturing line (which dedicates most of its resources to the production of the most profitable products), workload management aims to allocate computing resources to the most important applications. We refer to this as *workload optimization*. It's an attractive notion, although it can present quite a few business transformation challenges. For example, how do you decide what constitutes the most important work across the various components, or organizations, in an enterprise?

Still, the potential productivity gains and business benefits associated with the trend toward workload optimization are too great for the concept to be abandoned. The idea behind grid computing is aimed squarely at addressing the pressing need to leverage and reallocate existing IT resources. Next, we'll look at the origins of these ideas and concepts.

The origins of grid computing

Just like with the Internet, academic institutions were at the forefront when it came to developing the first-generation technologies and architectures that formed the basis of grid computing. Institutions such as the Globus Alliance, the China Grid, and the UK e-Science Grid core program were some of the first to incubate and grow grid solutions to maturity, preparing them for commercial adoption.

Grids were borne out of the research and academic communities' very real need to collaborate. A crucial component of research is the ability to disseminate knowledge -- the more efficiently you can share *not only* vast amounts of information *but also* the computational resources that help you create this data -- the more refined and informative a level of quality in collaboration you can achieve.

A counterpart to this need to disseminate knowledge is in the commercial world. Grid computing can also address this need, because the integration of business processes and transactions, facilitated by Web services standards, continues to grow in importance. As the adoption of commercial grid computing continues, standards (such as those proposed by organizations like the Global Grid Forum, or GGF) will benefit from the real-world, hardened, and practical requirements to which commercial applications will subject them.

Currently, grid computing benefits from the early identification and development of standards-based technologies in the academic world that is matched with more practical and robust implementations that commercial businesses require. There is no reason to imagine that this synergy will not continue as grid computing matures.

Grid fills a crucial gap

In the last few years, a crucial gap has developed between the advance of networking capability (the bits per second a network can handle) and microprocessor speed (based on the number of transistors per integrated circuit). This is illustrated in Figure 1.

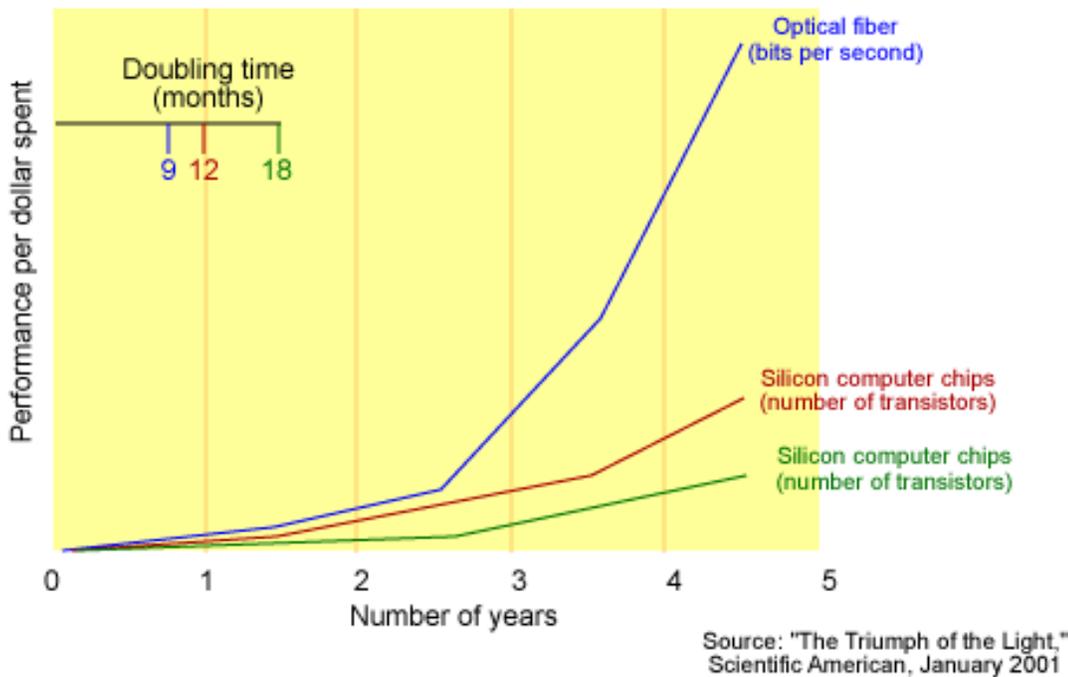
Figure 1. Moore's Law vs. storage improvements vs. optical improvements

Khosla on metacomputing

From "The Triumph of the Light," *Scientific American*, January 2001:

Vinod Khosla, a venture capitalist with Kleiner Perkins Caufield and Byers, talks of the promise of projects that pool together computers, which can be either side by side or distributed across the globe.

*Metacomputing [which Khosla defines as grid computing] can download Britney Spears and Fatboy Slim, or it can comb through radio telescope data in search of extraterrestrial life. Khosla sees **immense benefit** in using this model of networked computing for business, tying together machines to work on, say, the computational fluid dynamics of a 1,000-passenger jumbo jet.*



As the graph demonstrates, networking capability essentially doubles every nine months today, although historically this growth was much slower. And Moore's Law dictates that the number of transistors per integrated circuit still doubles every 18 months. Therein lies the problem. Moore's Law is slow compared with the advancement in network capability.

If you accept as a given that core networking technology now accelerates at a much faster rate than advances in microprocessor speeds, then it becomes apparent that in order to take advantage of the advances in networking, a more efficient way of harnessing microprocessor capacity is required. This new point of view changes the historical trade-off between networking and processing costs. Similar arguments apply to bulk storage.

Grid computing is the means to address this gap, this change in the traditional trade-offs, by tying together distributed resources to form a single virtual computer. Combining this resource-rich virtual computer with the benefits provided by application acceleration (going from weeks to days, days to hours, hours to minutes, and so on) offers businesses an enticing promise (although this will likely require a significant change in telco business practices, most notably in pricing).

Now that I've examined the origins of grid computing and made a case for its importance, I'll help strengthen your grid knowledge base by comparing it with other distributed computing concepts -- cluster computing, CORBA, DCE, and P2P.

How grid differs from cluster computing

Cluster computing can't truly be characterized as a distributed computing solution; however, it's useful to understand the relationship of grid computing to cluster computing. Often, people confuse grid computing with cluster-based computing, but there are important differences.

Grids consist of heterogeneous resources. Cluster computing is primarily concerned with computational resources; grid computing integrates storage, networking, and computation resources. Clusters usually contain a single type of processor and operating system; grids can contain machines from different vendors running various operating systems. (Grid workload-management software from IBM, Platform Computing, DataSynapse, and United Devices are able to distribute workload to a multitude of machine types and configurations.)

Grids are dynamic by their nature. Clusters typically contain a static number of processors and resources; resources come and go on the grid. Resources are provisioned onto and removed from the grid on an ongoing basis.

Grids are inherently distributed over a local, metropolitan, or wide-area network. Usually, clusters are physically contained in the same complex in a single location; grids can be (and are) located everywhere. Cluster interconnect technology delivers extremely low network latency, which can cause problems if clusters are not close together.

Grids offer increased scalability. Physical proximity and network latency limit the ability of clusters to scale out; due to their dynamic nature, grids offer the promise of high scalability.

For example, recently, IBM, United Devices, and multiple life-science partners completed a grid project designed to identify promising drug compounds to treat smallpox. The grid consisted of approximately two million personal computers. Using conventional means, the project most probably would have taken several years -- on the grid it took six months. Imagine what could have happened if there had been 20 million PCs on the grid. Taken to the extreme, the smallpox project could have been completed in minutes.

Cluster and grid computing are completely complementary; many grids incorporate clusters among the resources they manage. Indeed, a grid user may be unaware that his workload is in fact being executed on a remote cluster. And while there are differences between grids and clusters, these differences afford them an important relationship because there will always be a place for clusters -- certain problems will always require a tight coupling of processors.

However, as networking capability and bandwidth advances, problems that were previously the exclusive domain of cluster computing will be solvable by grid computing. It is vital to comprehend the balance between the inherent scalability of grids and the performance advantages of tightly coupled interconnections that clusters offer.

Grid or CORBA?

Of all distributed computing environments, CORBA probably shares more surface-level similarities with grid computing than the others. This is due to the strategic relationship between grid computing and Web services in the Open Grid Services Architecture (OGSA). Both are based on the concept of service-oriented architecture (SOA). CORBA is the backbone of multiple mission-critical applications and has continued to mature since its creation in 1991. In many respects, CORBA was a precursor to the Web (grid) services world we live in today. It laid an important foundation, as did the Java Remote Method Invocation (RMI) some years later.

For example, Boeing uses a CORBA-based solution in its DCAC/MRM application (just a fancy abbreviation for Define and Control Airplane Configuration/Manufacturing Resource Management), essentially the application that manages the parts configuration and inventory for commercial airplanes (and jetliners have *lots* of parts!). And Peter Coffee, *e-Week's* technology editor, recently reported that all operations on the new Cunard *Queen Mary 2* ocean liner are powered by CORBA.

A key distinction between CORBA and grid computing is that CORBA assumes object orientation (after all, it is part of the name), but grid computing does not. In CORBA, every entity is an object and it supports mechanisms such as inheritance and polymorphism. In OGSA, there are similarities to some object concepts, but there isn't a presumption of object-oriented implementation in the architecture. The architecture is message oriented; object orientation is an implementation concept. However, the use of a formal definition language (such as WSDL, Web Services Definition Language) in WSRF (Web Services Resource Framework) means that interfaces and interactions are just as precisely defined as in CORBA, sharing one of the major software engineering benefits also exhibited by object-oriented design.

Another distinction is that grid computing (OGSA) is built on a Web services foundation. CORBA integrates with and interoperates with Web services. One of the problems with CORBA was that it assumed too much of the "endpoints," which are basically all the machines (clients and servers) participating in a CORBA environment. There are also issues of interoperability between vendors' CORBA implementations, how CORBA nodes are able to interoperate on the Internet, and how endpoints are named. This means that all of the machines in the cohort had to conform to certain rules and to a certain way of doing things (all assuming the same protocols like IDL, IOR, and IIOP) for CORBA to work. This is an appropriate approach when you're building high-reliability, tightly coupled, pre-compiled systems.

However, synergy between the way CORBA performed its job and the Internet methodology was absent. CORBA did inspire the creation of Web services standards -- people liked what they saw in the underpinnings of CORBA and began creating such standards as XML, WSDL, SOAP, and others. They improved on the interoperability and flexibility issues that CORBA had by building Web services on an open Internet foundation with loose coupling and late binding between service requestors and services. To this improvement, OGSA added a "soft state" approach to fault tolerance. These were design goals.

The Web services architecture is a service-oriented architecture and so too is CORBA. CORBA had a different target though -- it was designed for building integrated, relatively closed, systems.

What about DCE?

As its name suggests, DCE (Distributed Computing Environment) is not so much an architecture as it is an environment, and therein lies the important distinction. DCE can be defined as a tightly integrated collection of technologies designed to facilitate distributed computing; grid computing (in the form of OGSA) is more of an end-to-end architecture designed to encapsulate many of the intricacies of the mechanics of distributed computing.

As we saw in examining CORBA, in DCE we see a distinction between tightly coupled and loosely coupled approaches. DCE technologies include security technology (DCE ACLs or Access Control Lists), object and component technologies (DCE distributed objects), a file system (DFS or Distributed File System), and a directory specification (DCE registry) -- in fact, OGSA can be made to work atop many DCE technologies.

For example, grid security protocols, either in the form of GSI (Grid Security Infrastructure) or in the form of appropriate Web services standards, could be made to interoperate with DCE ACLs. Many grid applications can be made to work with the underlying DFS (or its predecessor, AFS, the Andrew File System). Core grid registry services can work with the DCE registry.

While most of these technologies can be considered services, DCE isn't so much a service-oriented architecture as it is a collection of technologies. Its support for building applications in an SOA environment is limited because DCE primarily delivers the blocks to build distributed applications, but not necessarily to construct distributed service-oriented applications.

Another important distinction between grid computing and DCE, which is also relevant to CORBA, is that OGSA grid computing defines the following three categories of services:

- Grid core services
- Grid data services
- Grid program-execution services

CORBA, DCE, and Java RMI do not pay specific attention to data (beyond the DFS) or program-execution services, because these technologies are all essentially remote procedure call (RPC) systems. (An RPC is a protocol that one program can use to request a service from a program located in another computer in a network without having to understand network details. It is a *synchronous* operation that requires the requesting program to be suspended until the remote procedure returns results unless you use *lightweight processes* that share the same address space.) Many of the services specified and implemented in grid core services (as well as the WSRF) are similar to foundational services found in both DCE and CORBA. But data and program-execution services are unique to grid computing.

Finally, the distinction we made between grid computing and CORBA regarding their relationship to Web service standards is also true of DCE. Again, many of the improvements we see in Web services are owed to experience working with and fine-tuning such distributed systems as DCE and CORBA.

Finally, there's P2P

Applications such as KaZaA -- which invariably show up in headlines due to some problematic copyright issue -- are the primary source that has recently showered P2P (peer-to-peer) computing with a lot of attention. The technology itself, though, demonstrates interesting distributed features, many of which would be beneficial if used in a grid environment.

First, the hallmark of a P2P system is that it lacks a central point of management; this makes it ideal for providing anonymity and offers some protection from being traced. Grid environments, on the other hand, usually have some form of centralized management and security (for instance, in resource management or workload scheduling).

This lack of centralization in P2P environments carries two important consequences:

- P2P systems are generally far more scalable than grid computing systems. Even when you strike a balance between control and distribution of responsibilities, grid computing systems are inherently not as scalable as P2P systems.
- P2P systems are generally more tolerant of single-point failures than grid computing systems. Although grids are much more resilient than tightly coupled distributed systems, a grid inevitably includes some key elements that can become single points of failure.

This means that the key to building grid computing systems is finding a balance between decentralization and manageability -- not an easy chore.

Also, while an important characteristic of grid computing is that resources are dynamic, in P2P systems the resources are much more dynamic in nature and generally are more fleeting than resources on a grid. For both P2P and grid computing systems, utilization of the distributed resources is a primary objective. Given a wealth of computing resources, both of these systems will try to use them as much as possible.

A final distinction between the two systems is standards -- the general lack of standards in the P2P world contrasts with the host of standards in the grid universe. And, thanks to entities like the Global Grid Forum, the grid universe has a mechanism for refining existing standards and creating new ones.

Based on the mutual benefits that grid and P2P systems seem to offer to each other, we can expect that the two approaches will eventually converge, especially when grids reach the "inter-grid" stage of development in which they essentially become public utilities.

Leveraging a world of data

We've examined the components of grid computing, looked at its origins, explained its importance in enterprise-level, Web services-based applications, and provided a cursory analysis of the similarities and differences between grid computing and four major distributed computing systems.

Almost every organization is sitting atop enormous, widely distributed, unused computing capacity. Virtualization -- the driving force behind grid computing -- can help reach unused capacity, and IBM has a long history of involvement in the development of virtual memory, virtual storage, and virtual processor technology. But not just in the creation of this technology for customers.

IBM's intraGrid, based on Globus, is a research and development grid that lets IBM leverage its own global assets for research purposes, plus it gives grid developers within the company the opportunity to understand the real-world issues and complexities of managing a grid infrastructure on an enterprise scale. IBM also uses multiple organizational grids throughout the company, including a grid to tie together its Design Centers for e-business on demand, letting IBM manage the Centers as a single entity.

In the next article, we'll explain strategic technology directions for many important components of a grid system. Grid computing is growing into on demand and adaptive computing environments by integrating complementary technologies, mainly in the automation space, to realize enterprise class and business computing.

Resources

- ["The Triumph of the Light"](#) describes how extensions to fiber optics will supply network capacity to meet ever-growing bandwidth needs. (This article is the source of Figure 1, the original designed by Cleo Vilett with data supplied by Vinod Khosla, Kleiner, Caufield, and Perkins.)
- The ["History of CORBA"](#) provides a detailed look at the revisions to the specification starting with version 1.0 in October 1991.
- The whitepaper ["Reinventing the Wheel? CORBA vs. Web Services"](#) explains how CORBA is a services-oriented architecture designed for building relatively close integrated systems.
- ["On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing"](#) compares and contrasts P2P and grid computing, drawing a roadmap to their eventual convergence.
- ["A visual tour of Open Grid Services Architecture"](#) (*developerWorks*, October 2003) explains the objectives and components of the Open Grid Services Architecture (OGSA), the backbone behind grid computing.
- ["Is Web services the reincarnation of CORBA?"](#) (*developerWorks*, July 2001) details the differences and makes a case for the value of Web services within the distributed computing world.
- The [CORBA site](#) offers a list of success stories from a lengthy list of industries.

- The [Globus Alliance](#) is the purveyor of the widely used Globus Toolkit and a leader in setting the grid agenda.
- The [China Grid](#) is a joint project between IBM and China's Ministry of Education that uses grid technology to enable universities across the country to collaborate on research, scientific, and education projects; it is one of the world's largest implementations of grid computing. (The China Grid site is in Chinese; this [news item](#) explains the project in English.)
- The [UK e-Science GRID core program](#), with information on grid technology, promotes grid computing as the natural successor to the Web.
- The [Global Grid Forum](#) is a community-based forum of more than 5000 researchers and practitioners developing and furthering grid technologies and best practices.
- This [press release](#) explains grid's role in the IBM, United Devices, and Accelrys project to aid a global research effort focused on the development of new drugs that could potentially combat the smallpox virus post infection.

About the author



Matt Haynos is a Program Director on the IBM Grid Marketing and Strategy team, based in Somers, NY. He has various responsibilities on the team covering a broad range of initiatives related to building the IBM grid computing business. He has held a variety of technical and managerial positions within IBM in the application development, program direction, and business development areas. He holds a BA in Computer Science/Applied Mathematics and Cognitive Science from the University of Rochester and an MS in Computer Science from the University of Vermont. He lives with his wife and two sons in Connecticut.



What do you think of this document?

Killer! (5)

Good stuff (4)

So-so; not bad (3)

Needs work (2)

Lame! (1)

Comments?