

**University of Illinois, Urbana-Champaign**

**Survey of Computational Grid Systems and  
Virtualization in the Grid Space**

Submitted to:

Dr. Mehdi Harandi  
CS591 (CS425) – Distributed Systems

July 27, 2008

By

Scott R. Armstrong  
13260 West Eagle Point Road  
Polo, Illinois 61064  
(815) 946-9103  
Scott.Armstrong@acm.org  
sarmstr2@illinois.edu

## Survey of Computational Grid Systems and Virtualization in the Grid Space

### 1. Introduction

Distributed systems, and in particular, computational grid systems and computational cluster systems, are extensions of the traditional concept of a computer system. As processing demands continued to increase with scientific research, data analysis, modeling and other computational problems, computers also grew in size and capability. Different classes of computers were available for different functions. The familiar desktop personal computers served the needs of an individual or a family with word processing, accounting tasks and other applications, while large, self-contained supercomputers designed for high-performance, parallel processing found their homes in research laboratories and universities. The power of the personal computer has already exceeded the computing power of the first supercomputers, and combining the hardware resources of several commercial computers into clusters and grids to function as a single system. The result is supercomputer-class computing power that further blurs the distinction between single-box supercomputers and supercomputer-class systems composed of several personal computers.

By abstracting various parts of the computer system, we have been able to improve the usefulness of a computer system and increase the overall performance of the new abstraction. Disk input-output (I/O) was at one time limited to the system in that box. Networking provided a new I/O abstraction that enabled shared storage systems and shared resources. With networks allowing computers to communicate with each other, sharing memory systems and computational resources were also possible [9]. Sharing the processing resources of multiple computer systems in a managed environment allows us to view a connected system of computational nodes as a single system—a grid or cluster—and using that view, reason about and solve problems that were either computationally unfeasible or financially prohibitive in the past.

By virtualizing various parts of a grid or cluster system, we can add flexibility and scalability to the physical composition of the system as well as to the management of the concurrent task assignments for the applications being run on the system. The flexibility and scalability allow us to squeeze even more productive cycles from each of the available nodes (scheduling), recover more quickly from node failures, and deploy new nodes or clusters in the grid on demand.

I summarize in this paper the timeline of the history of grid computing, several implementations of grid systems, some of the more common middleware systems used to manage grid systems, and examples of virtualization in grids and clusters.

The collection of papers edited by Ian Foster<sup>1</sup> and Carl Kesselman and published in both the original and second editions of *The Grid: Blueprint for a New Computing Infrastructure*

---

<sup>1</sup> Ian Foster, dubbed the “Father of the Grid” [7], was is involved in many of the early grid initiatives and currently works in the Mathematics and Computer Science Division at Argonne National Laboratory.

[19, 20] describe a computational grid as a type of distributed system that shares a collection of computational resources. The name grid was borrowed from the analogy of electrical grids that shared electrical power resources. These two books, published 5 years apart, are an example of the recent chronology of computational grid systems, advancements in the middleware systems used to manage the compute nodes of grids and clusters, and the application domains that are making use of grids.

A computational grid system is typically described as an abstraction of hardware compute nodes that enables concurrent processing on a collection of commodity, consumer-quality computer hardware platforms that can match or exceed the processing capabilities of supercomputers [5, 19, 20]. Grid tasks are assigned to compute nodes by a grid management middleware system [19, 20].

The traditional computer system consists of the processing unit, memory, storage, input-output, operating system, and user applications. Advances in hardware technology have resulted in regular performance improvements, and attempts to boost performance of both hardware and software continue. The demand for increased processing capabilities, however, also continues, with new fields of research analyzing volumes of data, such as the computation of weather patterns, plotting objects in Space, and processing medical images. The compute power, flexibility and scalability of grids and clusters are helping meet the computation demands of today's research.

Though the notion of what exactly constitutes a grid system has been a fluid topic of discussion since the mid-1990s, most current descriptions of a grid include coordinated resource sharing in a virtual organization<sup>2</sup> that spans multiple, physical locations. Though a grid system is often described as an extension of internetworked resources, including the Internet, perhaps the most significant difference then between a grid and a network (and the World Wide Web) is that a grid system (and the eventual Great Global Grid of interconnected grid systems) enables the sharing of computing resources where an interconnected set of networks (and the World Wide Web) enables the sharing of information or files [20].

Grids are regularly compared to supercomputers (and used as an inexpensive alternative to supercomputers), and though the resulting computational power of a grid or cluster can match or exceed the computational power of contemporary supercomputers, the significant difference between the two types of systems is that managers of grid systems don't normally maintain exclusive, physical control over the compute nodes or processing resources. Clusters, similar to grids and also regularly described as supercomputers, employ some of the same technologies and software systems as grids, but compute nodes in clusters are normally under the exclusive control of the cluster manager and are often in the same location or even the same room.

---

<sup>2</sup> A Virtual organization, or VO, is a concept that describes the computer hardware, software, processing resources, data and other resources necessary to collaboratively solve an individual's or organization's computational problem or problems [2].

## 2. History

Though the contemporary grid movement began in earnest in the mid 1990s [19, 20], resource sharing was conceptualized in the 1950s and 1960s [7] to make use of idle processing time and overcome performance limitations. Leonard Kleinrock even suggested what would later become grid computing when he described on-demand access to computing resources as a “computer utility” in a 1969 UCLS press release [17].

Since the mid 1990s, when the term grid was coined to represent a proposed distributed, computational resource sharing infrastructure, researchers have constructed and extended the infrastructure model to support a variety of computational problems, even including systems connected to sensors and devices such as telescopes and microscopes. The early adoption of a sound architecture, coupled with ongoing efforts to provide standardized tools to manage and interact with the shared resources is a testament to the continued success of grid systems [13].

Looking for an affordable alternative to a mainframe supercomputer, Donald Becker brought grid systems to the masses. He designed a computational cluster using commodity computers with Intel DX4 processors as an inexpensive alternative to a supercomputer in 1993, and when funding was secured in 1994, the Beowulf project was born. Beowulf clusters, also referred to as a Beowulf-class computer, a Beowulf cluster and even a Beowulf computer, have been developed by NASA, DOE and several universities [5, 20]. By using commercial, off-the-shelf computers and open-source operating systems, supercomputer-class computational power in the form of a Pile of PCs (POPC) [33] was now available to individuals and institutions at a fraction of the price of a traditional supercomputer.

An early, operational example of a grid system was the I-WAY (Illinois Wide Area Year) system that connected 17 high-speed research networks for a span of two weeks in 1995 [34]. The success of I-WAY led to the funding of the Globus project, and in 1997, the release of the first version of the Globus Toolkit [7].

Early direction for grids was established by Foster and Kesselman as they stated the need for a standardized interface, grid-enabled applications, and the removal of the fragility of grid systems with the goal of shrink-wrapped grid system software [19]. Toward that end, the Globus project, with guidance and influence by Foster and Kesselman, developed a standardized toolkit that provided a standardized architecture and set of software tools to create and manage a computational grid consisting of remotely shared compute nodes. The Globus Took Kit was later handed over to the open source community [18, 20, 21]. This middleware system and others make it possible to view a highly distributed system of shared compute nodes as a single system.

Computing needs continue to evolve, and some on-demand computing solutions are being implemented as grids within an organization with virtualized servers, virtualized storage, and even completely virtualized datacenters. Virtualization of grid system components allows organizations to focus on the problems to be solved or the business to be done

rather than on the hardware infrastructure that is supporting the solution or the business [20].

If the history of the Internet is any indication of the future of grid systems, the future direction of grid systems (and possibly even the Grid, with a capital G, a worldwide system of interconnected grid systems) will be driven and supported by market forces and business needs. The ability to provide utility computing to companies and organizations will open another funding source that can push the development and use of computational grid systems beyond research facilities and into homes and offices. The Grid could do for computation what the World Wide Web has done for data and information.

### 3. Grid implementations

Grid systems are typically implemented to solve a specific problem, but the types of implementations can be categorized very broadly as consisting of compute nodes that are either tightly coupled clusters, loosely coupled clusters and grids, or volunteer grid systems that are loosely coupled by design. Compute nodes that comprise cluster systems are likely to be located in the same facility each other, whereas volunteer grid systems will typically consist of individual computers around the world communicating with the managing system via the Internet. Loosely coupled clusters allow the introduction of new compute nodes, and grid systems typically allow the introduction of new compute nodes and clusters. Grids can be composed of many individual compute nodes, several computational clusters, and a combination of individual nodes and clusters. Cluster systems include the NASA Advanced Supercomputing (NAS) Pleiades and Columbia systems, as well as implementations using Sun Grid by Sun Microsystems and Xgrid by Apple [2, 27, 35]. Volunteer systems include the widely known SETI@home [32] and Folding@home [16] projects.

Software and hardware components in a grid system can vary, but most grid systems will include a number of compute nodes, a middleware manager system that assigns work to the compute nodes, a fabric that includes the compute nodes as well as the network that serves as a means of communication between the manager system and the compute nodes, and an application that can execute in the grid environment [20, 21].

#### 3.1. SETI@home

SETI@home was initially developed in 1995 as part of Berkeley's Search for Extraterrestrial Intelligence (SETI) project as an alternative to deploying supercomputers to the location of radio telescopes to process and analyze radio signals. Launched publicly in 1999, SETI@home asked computer users to run a small piece of software (BOINC) that recognizes when the computer is idle, sends a request to the SETI project system, downloads a data file, analyzes the data based on routines developed by the SETI team, and send the results of that analysis back to the SETI project system. This is a very loosely coupled system that has the potential to exceed the compute power of several supercomputers [32].

#### 3.2. Folding@home

Like SETI@home, Stanford University's Folding@home project uses idle processing power of home computer users to simulate and analyze protein folding, mis-folding, and related diseases. Computer users install a small software application that then enables that computer to become a compute node in a loosely coupled grid system that processes part of a protein folding simulation [16].

### 3.3. Sun Grid at Network.com

Sun Microsystems claims that Sun Grid is the world's only true compute utility. Sun Grid, available at Network.com, offers computing resource services at the cost of \$1 per CPU hour. These services can be high-performance compute, batch processing or other applications, and are made available by running middleware applications from the Sun Grid application catalog. One such application is Blender, a 3D animation software application that was used in conjunction with the Sun Grid at Network.com to render a short 3D animated movie, Big Buck Bunny [28]. Applications can be developed and submitted to the catalog, but the compute nodes can't be added or removed from the Sun Grid system by those paying for the services, making this system appear to users to be a tightly coupled system [28, 35].

### 3.4. OurGrid

OurGrid is a peer-to-peer grid middleware system that minimizes the importance of an extensive or substantial computational infrastructure. The OurGrid Community uses the OurGrid volunteer, free-to-join grid system. It incorporates a "Network of Favors" approach to job scheduling by rewarding more computing resources to those who donate more idle computer cycles to the grid. Applications best suited for processing on an OurGrid system are "Bag of Tasks" or easily parallelized applications that require significant computational power, such as simulations, medical image processing and data mining [6, 8].

### 3.5. Apple Xgrid

Apple released Xgrid in 1994 and incorporated the grid functionality in the workstation and server versions of OS X since then, allowing very simple configuration of compute nodes and grid management for a cluster of OS X systems. It is considered a loosely coupled system and consists of the grid controller module, the client module, and an agent module that advertises its availability to the grid. Other agents exist to allow Linux Xgrid nodes in an Xgrid controlled system, and a Java agent to provide true, heterogeneous clients in an Xgrid [2, 24].

### 3.6. NASA Advanced Supercomputing (NAS) facility

NASA has been involved with supercomputing-class research since the 1980s, including projects using Beowulf clusters and mass storage systems. Current running projects include the RTJones, Schirra, Columbia, and the recent Pleiades clusters. The scale and performance of the NAS systems continue to grow, from the 4.8 Teraflop/s of the Schirra system, the 43.5 Teraflop/s of the RTJones system, the 88.88 Teraflop/s of the Columbia system, to the 245 Teraflop/s of the Pleiades cluster. These are tightly coupled, non-

volunteer systems that are dedicated to NASA mission research and support [27]. It is interesting, though, that each QuadCore processor at approximately 48 Gigaflop/s in the Pleiades system is itself about 300 times more powerful than the Cray-1 supercomputer's capability of 160 Megaflop/s in 1976.

### 3.7. Open Science Grid

Founded by the National Science Foundation and the Department of Education, the Open Science Grid (OSG) supports the computing and data management requirements of the scientific and educational communities. This is a loosely coupled, volunteer grid system that requires that each remote compute node be registered as a resource with the OSG. Researchers gain access to the OSG grid resources by registering with one or more of the OSG VOs [29].

### 3.8. TeraGrid

With 750 Teraflop/s capabilities, TeraGrid is the world's largest grid system supporting open scientific research. Coordinated through the Grid Infrastructure Group at the University of Chicago, the TeraGrid is a loosely coupled grid system that consists of clusters and systems at Indiana University, Purdue University, Louisiana Optical Network Initiative, National Center for Atmospheric Research, National Center for Supercomputing Applications at the University of Illinois, the National Institute for Computational Sciences, Oak Ridge National Laboratory, the Pittsburgh Supercomputing Center, San Diego Supercomputing Center, Texas Advanced Supercomputing Center, and Argonne National Laboratory [36].

### 3.9. Access Grid

Access Grid is a distributed system that includes resources traditionally shared in a grid system, but it also shares interactive environments and multimedia resources to support real-time collaboration through teleimmersion. More than sharing computational resources in pursuit of supercomputer computational capabilities, this is a highly interactive grid system built around the Globus Toolkit that enables communication with, and control of, remote systems including multimedia camera systems and groups of service robots [1].

### 3.10. UC Grid

The University of California Grid (UC Grid) is a loosely coupled grid system of independently owned and managed resources. Major components of the UC Grid include primarily university-specific grids that are themselves composed of multiple computational clusters. Some clusters outside of the University of California system are also on the UC Grid, including some clusters on the TeraGrid. Nodes and clusters on the UC Grid can be accessed via the UC Grid Portal—a Web interface [38].

### 3.11. FusionGrid

The FusionGrid supports the National Fusion Collaboratory Project by providing software tools and computational resources for the magnetic fusion research community. Participants include the Argonne National Laboratory (Ian Foster and others), General Atomics, Princeton University, Princeton Plasma Physics Laboratory, Lawrence Berkeley National Laboratory, University of Utah, and the MIT Plasma Science and Fusion Center. FusionGrid features include efforts to simplify access instead of portability, to make resources available to a large audience without having to support multiple installations or configurations. Networked compute node resources on the FusionGrid are viewed as network services. Users of FusionGrid also make use of the grid-enabled interactive systems provided by Access Grid [41].

#### 4. Grid middleware systems

Simply connecting compute nodes together on a network won't provide grid-like computational resources to scientists and researchers. Grids and clusters typically include software middleware systems that handle job scheduling, task assignments, resource monitoring, process communications, and other needs common in distributed systems. Foster and Kesselman advocated the need for a standard, open interface, and as part of the Globus project, developed the Globus Toolkit, often referenced in grid-related articles as GTK [19, 20, 21].

Grid middleware is described as requiring four component layers: 1) the communications fabric; 2) the core middleware; 3) the user-level middleware; and 4) the applications and portals layer. The grid fabric consists of the shared hardware resources, networks, storage, and any other hardware that interacts with the system. Core middleware manages remote processes, allocation of resources, storage, node registration and discovery, and security. User-level middleware makes use of the abstraction provided by the core middleware system to present application development environments and tools as well as tools for resource management and task scheduling to the user. The applications and portals layer provides grid-enabled languages, libraries and utilities, and Web services to submit, schedule and retrieve the results of jobs that were executed on remote resources [21].

Below are brief overviews of several of the more common grid middleware systems. There are many tools and toolkits available that provide specific functionality and also work in conjunction with other grid middleware systems—the Java Grid Application Toolkit [39], for example. Middleware systems can also interact with each other as clusters and grids are combined into VOs to create larger grids.

##### 4.1. Globus Project

Globus provides an open-source toolkit to build grids and grid-enabled applications. The Globus architecture includes three main services: 1) resource management; 2) data management; and 3) information services. A Grid Security Infrastructure layer provides grid user authentication and secure communications between the services and the underlying grid resources [3, 21].

The resource management service includes two main components. The Globus Resource Allocation Manager that provides remote execution capability and monitoring, job

submission and scheduling, as well as load sharing and load leveling. The Globus Access to Secondary Storage is a remote file access mechanism to support the need to access data or executable files, pre-fetch data for scheduled jobs and retrieve output data from completed jobs [3].

The data management service provides utilities and libraries to access, transmit, manage and store massive amounts of data across the shared storage resources [3].

The information services provided by Globus include the Monitoring and Discovery Service that provides information on the static and dynamic properties of the nodes on the grid [3].

#### 4.2. Condor

The University of Wisconsin's Condor project is a specialized workload-management system that is designed to support high-throughput computer as apposed to the efforts by many other systems to support high-performance computing. Features include policy-based job scheduling and prioritization, as well as resource monitoring and management of dedicated compute nodes. Condor is able to run applications that were not compiled specifically for the Condor system in a plain-vanilla "universe" but users can also re-link their application with a set of Condor-specific libraries and submit those jobs to the standard Condor "universe" for execution [12].

#### 4.3. Unicore

Unicore is a middleware system that lets end users or grid managers easily migrate jobs to different hardware platform. It is a three-tiered system that includes a Java-based client, a gateway, and multiple network job supervisors with target system interfaces. The user view of Unicore is a client application on the workstation, a system that provides an interface to the grid sites, and a system to allow access to the resources [3].

#### 4.4. Gridbus

Gridbus, an open-source grid middleware system managed at the University of Melbourne, was designed and developed as service-oriented cluster and grid middleware technologies to support eScience and eBusiness. The design included efforts to abstract the distributed systems features of cluster and grid systems to simplify the development process for grid-enabled applications. The Gridbus architecture also integrates Globus and Unicore middleware systems, providing the developer a single abstraction of the development environment. Gridbus includes distinct systems to provide scheduling services based on fair market computational power needs, resource allocation and monitoring, accounting services, resource brokers and a Web portal [3, 22].

#### 4.5. Legion

The University of Virginia's Legion project is a middleware system designed to manage a large number of heterogeneous, loosely coupled nodes and storage systems. It structures the nodes as distributed objects, resulting in an object-based metacomputer that offers both flexibility and site autonomy. All resources in a Legion grid system are represented as

Legion objects and are described using a Legion-specific interface description language [3, 26].

## 5. Virtualizing the grid space

Virtualization, in particular the virtual machine monitor (VMM) and the virtual machine (VM), has been a useful abstraction since the 1960s, when it was originally used to multiplex mainframe environments to share the resources between multiple applications [31]. The reasons for employing VMs in the 1960s differ from the current driving forces, but the solutions had similar challenges and benefits.

Though hardware and software are logically equivalent, hardware is generally faster than software while at the same time less flexible than software. Software solutions, though slower than hardware, can be modified as often as needed and even replaced with completely different software. Virtualization, a software abstraction of a hardware platform, was the solution then to share scarce computational resources.

Costs associated with virtualization are generally in the form of additional processing overhead, but para-virtualization techniques [4] and on-chip support for virtualization [25] continue to minimize the costs of virtualization. The most common VMM properties can be generally categorized as isolation, duplication, and abstraction, while VMs enjoy the benefits of encapsulation, increased security, and portability. These properties enable quicker backup, recovery and initialization of entire system environments than would be possible with non-virtualized systems [11, 31].

As a software layer of abstraction, VMMs provide additional functionality not normally available at the hardware layer, such as managing multiple VMs using the resources of a single hardware platform and migrating VMs between VMMs. It is this functionality that makes virtualization a useful option for deploying and managing computational grid and cluster systems. In fact, virtualization has already been incorporated into a computational grid system in a variety of ways.

### 5.1. Shirako

Virtualization introduced into the grid domain using the Shirako system can provide “... on-demand, adaptive, and reliable allocation of networked computing resources from a common pool.” [23] The Shirako toolkit uses Cluster on Demand to deploy and manage Xen VMs at cluster sites, resulting in “... secure, adaptive, on-demand resource sharing in federated clusters.” [23] VM migration can satisfy load balancing needs, and using Shirako with the Xen hypervisor, VMs can be migrated either via stop-and-copy or live migration. Established policies broker migrations to allocate and balance resources, including both provisioning and placement, with provider sites having some control over the placement of virtualized nodes [23].

### 5.2. Sandboxing grids with Xen

Virtual Machines as grid sandboxes can enhance the deployment of compute nodes in a Wide-area Overlay of virtual Workstations. By virtualizing not just the hardware nodes,

but also the networking environment, Wolinski, Agrawal and others created an entire grid sandbox that is isolated from other systems. As with any virtualization effort, processing overhead exists. The processing overhead in this implementation, though, is only about 4% when using Xen and about 10% when used with VMWare. Virtualized network and disk I/O incur typical overhead costs as well. Implementing IP over Peer-to-Peer networking adds quite a bit of overhead, but it allows the network to exist completely in user space, enabling the deployment of preconfigured, homogeneous VM nodes on heterogeneous hosts. The target usage for this system is a PlanetLab system, such as PlanetLab Central and Open Science Grid. Though the nodes in these systems are virtualized, the networks aren't virtualized. This sandbox approach virtualizes all components of a grid system, including the network, and packages them in a generic, baseline image—a preconfigured VM appliance. The VM appliance contains all of the necessary software and configurations as well as the settings necessary to self-configure its virtual IP address and initiate peer discovery when it is instantiated on an overlay network. Operating system updates on the appliances are simplified as well, and the use of the UnionFS, a distributed file system, enhances the overall effectiveness of the system of virtualized appliances once they are merged with the file system environment [40].

### 5.3. OurGrid SWAN

OurGrid SWAN is a free-to-join grid that employs virtualization to provide security to the host from the unknown applications that may be running on a node, but also provides sabotage tolerance to protect the applications from any attacks by the resources. The sandbox exists only during the execution of the grid task and is mounted as a read-only image. It also uses a shared file system—the Oracle Cluster File System, version 2—to avoid the need for a virtualized network that could access the host network. This isolates the computing environment by sharing storage resources instead of network resources. Again, virtualization isn't free. OurGrid SWAN suffers a 10% performance penalty in processing costs and a 24-38% performance penalty from the use of the shared storage solution [8].

### 5.4. PLXD

Chris Edwards and Aaron Harwood used Xen, a para-virtualization system, to federate the domains running on PlanetLab by separating some of the management functionality into different, virtualized entities, called Xlices. These slice authorities provide users the ability to create and assign virtualized compute nodes, while the overall system management authority remains separate from user influences. PlanetLab in a Xen Domain (PLXD) makes it possible to run multiple PlanetLab nodes with their related management authorities on the same hardware. This virtualized system also allows recursive federation, just as virtualized environments can in turn host other virtualized environments [14].

### 5.5. XGE

Niels Fallenbeck and others describe the use of virtualization to support fair scheduling techniques for a mix of parallel and serial execution tasks in a computational cluster.

Scheduling jobs normally requires either partitioning a cluster (or grid resources) or balancing and back-filling jobs, and the method used will normally depend on whether the majority of the computational tasks that are run on the system are parallel or serial tasks. Used separately, both methods result in idle (wasted) resources. By extending the Sun Grid Engine with the Xen Hypervisor Monitor, the resulting system, XGE, separates each physical host into two nodes—one to support serial and one to support parallel task execution. Additionally, serial jobs running on virtualized nodes can be suspended in favor of a higher priority parallel job and then be restarted transparently with little overhead. Xen was chosen as the virtualization system because of the lower performance penalty when compared to the full virtualization provided by VMWare. The XGE system provides better fair use job scheduling than partitioning or balancing alone [15].

## 6. Summary

A computational grid is a system of coordinated resource sharing in a virtual organization that spans multiple, physical locations. As distributed systems, computational grid systems and cluster systems have the same resource-sharing challenges of smaller distributed systems, including scheduling and synchronization, communication failures, node failures, and in some cases, challenges introduced by heterogeneous hardware and operating systems. Grids and clusters, however, can also provide a system view that allows computational resource sharing on a grand scale, resulting in computational capability that is easily scalable, relatively fault tolerant, and far exceeds that of a single supercomputer.

The advent of distributed computational resource sharing in the form of Beowulf clusters also brought supercomputer-class power to organizations an affordable level. Grids and clusters have been deployed to support a variety of requirements, from academic and scientific research, to teleimmersion and remote collaboration, and utility computing as a purchased service.

Nodes in a computational grid can be either loosely or tightly coupled, each with suitability for different types of requirements and functions. The idle processor cycles of the loosely coupled nodes in the SETI@home and Folding@home grids are used to concurrently process segmented data analysis tasks with little concern for scheduling, while the tightly coupled compute nodes of the NAS facility can provide computational resources to NASA scientists that can satisfy strict scheduling and timing constraints of the various high-performance, highly parallel computing jobs.

Middleware systems are a key feature of computational grid systems, providing a standardized interface to the shared resources, tools to develop grid-enabled applications, and portals for job submission and scheduling. These software systems also connect the resources as a virtual organization, load balance the jobs assigned to the compute nodes, and manage node registration, failure and recovery. The physical components in a grid system—the compute nodes, storage systems and networking systems—at the level below the middleware system make up the grid fabric.

Virtualization affords flexibility to computational grid systems by replacing the physical components of a grid fabric with virtualized components. Compute nodes can be replaced

with appliance-like VMs to facilitate rapid grid and node deployment, rapid node failure recovery and restart, simple scalability, and overall system simplification by using preconfigured, homogeneous VMs as nodes hosted by heterogeneous physical systems. Networks, storage systems and even memory systems can also be virtualized, abstracting and simplifying the user view of the grid system as a single shared resource.

Though virtualization of hardware resources typically incurs a performance penalty, the performance gains realized through more efficient job scheduling in the case of XGE, for example, could exceed the performance penalty of the virtualized compute nodes. Virtualization makes it possible to reason about, and implement new solutions to the traditional distributed systems challenges in high-performance computing, supercomputer applications, large-scale distributed, concurrent processing jobs, and large volume data analysis.

Virtualization brings exciting abstraction possibilities to the grid space.

References:

- [1] Access Grid. <http://www.accessgrid.org/>
- [2] Apple Xgrid. <http://www.apple.com/server/macosx/technology/xgrid.html>
- [3] Global Grids and Software Toolkits: A Study of Four Grid Middleware Technologies (2004). Asadzadeh, P., Buyya, R., Kei, C., Nayar, D., Venugopal, S. <http://www.gridbus.org/papers/gmchapter.pdf>
- [4] Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A. 2003. Xen and the art of virtualization. *SIGOPS Oper. Syst. Rev.* 37, 5 (Dec. 2003), 164-177. DOI= <http://doi.acm.org/10.1145/1165389.945462>
- [5] Beowulf Project: <http://www.beowulf.org/>
- [6] Brasileiro, F. Inexpensive and Scalable High Performance Computing: the OurGrid Approach. [http://www.ieeetcsc.org/tcsc\\_files/TCSC\\_OG\\_ShortArticle%20\\_2\\_.pdf](http://www.ieeetcsc.org/tcsc_files/TCSC_OG_ShortArticle%20_2_.pdf)
- [7] Braverman, A. Father of the Grid, an interview with Ian Foster (April 2004). <http://magazine.uchicago.edu/0404/features/index.shtml>
- [8] Cavalcanti, E., Assis, L., Gaudencio, M., Cirne, W., and Brasileiro, F. 2006. Sandboxing for a free-to-join grid with support for secure site-wide storage area. In *Proceedings of the 2nd international Workshop on Virtualization Technology in Distributed Computing* (November 17 - 17, 2006). Virtualization Technology in Distributed Computing. IEEE Computer Society, Washington, DC, 11. DOI= <http://dx.doi.org/10.1109/VTDC.2006.11>
- [9] Chase, J. S., Levy, H. M., Feeley, M. J., and Lazowska, E. D. 1994. Sharing and protection in a single-address-space operating system. *ACM Trans. Comput. Syst.* 12, 4 (Nov. 1994), 271-307. DOI= <http://doi.acm.org/10.1145/195792.195795>
- [10] Chen, L., Zhu, Q., and Agrawal, G. 2006. Supporting dynamic migration in tightly coupled grid applications. In *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing* (Tampa, Florida, November 11 - 17, 2006). SC '06. ACM, New York, NY, 117. DOI= <http://doi.acm.org/10.1145/1188455.1188577>
- [11] Chen, P. M. and Noble, B. D. 2001. When Virtual Is Better Than Real. In *Proceedings of the Eighth Workshop on Hot Topics in Operating Systems* (May 20 - 22, 2001). HOTOS. IEEE Computer Society, Washington, DC, 133.
- [12] Condor. <http://www.cs.wisc.edu/condor/>
- [13] Douglass, F. and Foster, I. 2003. Guest Editors' Introduction: The Grid Grows Up. *IEEE Internet Computing* 7, 4 (Jul. 2003), 24-26. DOI= <http://dx.doi.org/10.1109/MIC.2003.1215656>
- [14] Edwards, C. and Harwood, A. 2006. Using Para-Virtualization as the Basis for a Federated PlanetLab Architecture. In *Proceedings of the 2nd international Workshop on Virtualization Technology in Distributed Computing* (November 17 - 17, 2006). Virtualization Technology

- in Distributed Computing. IEEE Computer Society, Washington, DC, 13. DOI=  
<http://dx.doi.org/10.1109/VTDC.2006.16>
- [15] Fallenbeck, N., Picht, H., Smith, M., and Freisleben, B. 2006. Xen and the Art of Cluster Scheduling. In *Proceedings of the 2nd international Workshop on Virtualization Technology in Distributed Computing* (November 17 - 17, 2006). Virtualization Technology in Distributed Computing. IEEE Computer Society, Washington, DC, 4. DOI=  
<http://dx.doi.org/10.1109/VTDC.2006.18>
- [16] Folding@home project. <http://folding.stanford.edu/>
- [17] Foster, I. What is the Grid. In *The Grid Today* (<http://www.gridtoday.com/>). <http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf>
- [18] Foster, I. and Kesselman, C. Globus: A Metacomputing Infrastructure Toolkit. <ftp://ftp.globus.org/pub/globus/papers/globus.pdf>
- [19] Foster, I. and Kesselman, C. (eds.). *The Grid: Blueprint for a New Computing Infrastructure (Original Edition)*. Morgan Kaufmann, San Francisco, 1999.
- [20] Foster, I. and Kesselman, C. (eds.). *The Grid: Blueprint for a New Computing Infrastructure (2nd Edition)*. Morgan Kaufmann, San Francisco, 2004.
- [21] Globus Alliance. <http://www.globus.org/>
- [22] Gridbus. <http://www.gridbus.org/>
- [23] Grit, L., Irwin, D., Yumerefendi, A., and Chase, J. 2006. Virtual Machine Hosting for Networked Clusters: Building the Foundations for "Autonomic" Orchestration. In *Proceedings of the 2nd international Workshop on Virtualization Technology in Distributed Computing* (November 17 - 17, 2006). Virtualization Technology in Distributed Computing. IEEE Computer Society, Washington, DC, 7. DOI= <http://dx.doi.org/10.1109/VTDC.2006.17>
- [24] Hughes, B. 2006. Building computational grids with apple's Xgrid middleware. In *Proceedings of the 2006 Australasian Workshops on Grid Computing and E-Research - Volume 54* (Hobart, Tasmania, Australia, January 16 - 19, 2006). R. Buyya, T. Ma, R. Safavi-Naini, C. Steketee, and W. Susilo, Eds. ACM International Conference Proceeding Series, vol. 167. Australian Computer Society, Darlinghurst, Australia, 47-54.
- [25] Intel. <http://www.intel.com/>
- [26] Legion project. <http://www.legion.virginia.edu/>
- [27] NASA Advanced Supercomputing facility. <http://www.nas.nasa.gov/>
- [28] Network.com, a Sun Grid Compute Utility. <http://www.network.com/>
- [29] Open Science Grid. <http://www.opensciencegrid.org/>

- [30] Rosenblum, M. 2004. The Reincarnation of Virtual Machines. *Queue* 2, 5 (Jul. 2004), 34-40. DOI= <http://doi.acm.org/10.1145/1016998.1017000>
- [31] Rosenblum, M. and Garfinkel, T. 2005. Virtual Machine Monitors: Current Technology and Future Trends. *Computer* 38, 5 (May. 2005), 39-47. DOI=<http://dx.doi.org/10.1109/MC.2005.176>
- [32] SETI@home project: <http://setiathome.berkeley.edu/>
- [33] Sterling, T. 1996. The scientific workstation of the future may be a pile of PCs. *Commun. ACM* 39, 9 (Sep. 1996), 11-12. DOI= <http://doi.acm.org/10.1145/234215.234461>
- [34] Stevens, R., Woodward, P., DeFanti, T., and Catlett, C. 1997. From the I-WAY to the National Technology Grid. *Commun. ACM* 40, 11 (Nov. 1997), 50-60. DOI= <http://doi.acm.org/10.1145/265684.265692>
- [35] Sun Grid Utility Computing. <http://www.sun.com/service/sungrid/>
- [36] TeraGrid. <http://www.teragrid.org/>
- [37] UCLA Press Release, July 3, 1969. <http://www.lk.cs.ucla.edu/LK/Bib/REPORT/press.html>
- [38] University of California Grid. <http://www.ucgrid.org/>
- [39] van Nieuwpoort, R. V., Kielmann, T., and Bal, H. E. 2007. User-friendly and reliable grid computing based on imperfect middleware. In *Proceedings of the 2007 ACM/IEEE Conference on Supercomputing* (Reno, Nevada, November 10 - 16, 2007). SC '07. ACM, New York, NY, 1-11. DOI= <http://doi.acm.org/10.1145/1362622.1362668>
- [40] Wolinsky, D. I., Agrawal, A., Boykin, P. O., Davis, J. R., Ganguly, A., Paramygin, V., Sheng, Y. P., and Figueiredo, R. J. 2006. On the Design of Virtual Machine Sandboxes for Distributed Computing in Wide-area Overlays of Virtual Workstations. In *Proceedings of the 2nd international Workshop on Virtualization Technology in Distributed Computing* (November 17 - 17, 2006). Virtualization Technology in Distributed Computing. IEEE Computer Society, Washington, DC, 8. DOI= <http://dx.doi.org/10.1109/VTDC.2006.8>
- [41] Fusiongrid. <http://www.fusiongrid.org/>