Virtual Playgrounds: Managing Virtual Resources in the Grid

K. Keahey[1,2], J. Chase[3], I. Foster[1,2]
[1]*University of Chicago*
[2]*Argonne National Laboratory*
[3]*Duke University*
keahey@mcs.anl.gov
chase@cs.duke.edu
foster@mcs.anl.gov

Abstract

Large Grid deployments increasingly require abstractions and methods decoupling the work of resource providers and resource consumers to implement scalable management methods. We proposed the abstraction of a Virtual Workspace (VW) describing a virtual execution environment that can be made dynamically available to authorized Grid clients by using well-defined protocols. Virtual workspaces provide resources in controllable ways that are independent of how a resource is consumed. A Virtual Playground may combine many such workspaces, as well as other aspects of virtual environments, such as networking and storage, to form virtual Grids. In this paper, we report on the goals and progress of the Virtual Playground Project and put in context the research to date.

1. Introduction

Large and successful Grid deployments such as Grid3 [1], Open Science Grid (OSG) [2], and TeraGrid [3], increasingly structure their operations as interactions between two classes of participant in a distributed system with distinct roles and goals: resource providers and resource consumers [4, 5]. *Resource providers* own and operate physical resources at some site, and may contribute them to satisfy consumer demand. Providers require incentives to participate, low participation costs, protection from activities performed by the resource consumer, and the ability to monitor and control the resources. *Resource consumers* want on-demand access to computational resources at modest cost, and the ability to configure them to meet the needs of the hosted environment. A resource consumer may be an individual user or some entity, such as a Virtual Organization (VO) [6] that operates or manages a distributed service on behalf of a user community.

Thus requirements of both providers and consumers converge on the following aspects of interaction:

1) *Flexibility in environment configuration*: it is essential to the consumers that they can adapt a leased environment to their needs while the provider needs to be able to provide this ability at a low cost and such that it will scale in the number of consumers.
2) *Environment isolation*: the provider needs to be able to delegate resource usage to consumers in such a way that their activities cannot impact the provider -- and therefore don't require fine-grained, consumer infrastructure dependent, and thus costly, monitoring.
3) *Strict enforcement*: a provider needs to be able to grant, constrain, enforce, and account for consumer resource usage in a way that is independent of how the resource is consumed thus avoiding adapting the system of every individual consumer. Such strict enforcement capabilities also underpin the feasibility of incentives.
4) *Dynamic creation and management*: it is essential that the creation of such environments, or environment leases be negotiated between the provider and consumer dynamically in order to support on-demand relationships.

We argue that the key to providing such synergistic interactions between resource consumers and resource providers is the ability to create virtual execution environments: virtual computers whose properties are negotiated between consumer and provider. The term

"virtual execution environment" can cover a range of architectural concepts and structures, from a single environment to a dynamic and distributed collection of virtual resources in the wide-area. Implementations of the concept can range from automating methods of hardware configuration to fulfill the conditions described above, to using diverse implementations of the virtual machines concept as well as virtual networking and storage. Further, mapping such virtual environments onto multi-dimensional resources federated across many administrative domains involves complex strategies in monitoring and resource brokering.

A key objective of the Virtual Playgrounds project is to define the abstractions of such virtual execution environments and develop methods for their deployment and management that meet the needs of producers and consumers in the Grid setting. In this paper, we describe virtual workspaces (representing environments within one administrative domain) and virtual playgrounds (collections of workspaces potentially spanning multiple administrative domains). Our objective is to develop methods to (a) dynamically provide isolated execution environments to Grid clients, (b) enable a fluid assignment of resources to such environments, and (c) enable environments to adapt to changing resource assignments transparently to hosted applications. We report on progress in modeling and development of virtual workspaces and their management capabilities, and discuss ongoing and future efforts in the areas of resource management, networking and security.

In our ongoing research we actively interact with scientists from OSG and TeraGrid to both understand their requirements as well as obtain feedback on the developed methods. With support from deployment-oriented projects, we have already been able to see prototype deployment of the ideas developed on this project and were given a positive evaluation of the infrastructure proposed to date.

2. The Virtual Workspace Abstraction

A *virtual workspace* is an abstraction of an execution environment that can be made dynamically available to authorized clients by using well-defined protocols. The abstraction captures the resource quota assigned to the execution environment (e.g., CPU, network share, or memory) as well as its software configuration (e.g., operating system installation, provided services). Workspaces may be implemented in many different ways; examples include a physical machine configured as an ATLAS Grid 3 [1] node using the Pacman configuration software [7], a physical cluster configured with the Xen hypervisor [8], or a set of virtual machines representing a service node for an OSG resource.

2.1. Workspaces on Physical Resources

One way of providing on-demand access to customized workspaces is the to allow for automated boot of physical nodes based on a configuration description and boot images (as implemented for example in bcfg [9]), thus allowing a site to manage its nodes in a more flexible and controlled manner. The Cluster on Demand (COD) project [10] was one of the first examples of this approach: it allowed a remote client, identified by a token, to provision and deploy an isolated virtual cluster based on database-driven network booting.

Many Grid deployments today prefer to provide a base configuration but allow clients to refine it using configuration management software (such as Pacman [7]) to deploy pre-defined software configurations automatically. One can thus treat the availability of base images as a precondition and schedule the deployment of environments building on that base.

However, deploying workspaces as physical resources offers little flexibility in determining the workspace's configuration (it is still largely provider-dependent and likely to remain so for security reasons) or shaping the resources that can be allocated to it at the required granularity (the physical machine is essentially the unit of granularity in this case). Further, the more configuration can be done at workspace creation time the greater the flexibility but also the longer the deployment time. In the case of real-world complex configuration this difference can be significant and add up to a few hours [11].

2.2. Workspaces on Virtual Resources

Virtual machines (VMs) [12] provide a promising implementation option for workspaces. A VM provides a virtualization of a physical host machine. Software running on the host, typically called a virtual machine monitor (VMM) or hypervisor, is responsible for supporting this abstraction by intercepting and emulating instructions issued by the guest machines. A hypervisor also provides an interface allowing a client to start, pause, serialize, and shut down multiple guests. In addition, many modern hypervisors offer fine-grained enforcement of resource usage. A VM representation (VM image) is composed of a full image of a VM RAM, disk images, and configuration

files. Thus, a VMs can be paused, its state serialized, and later resumed at a different time and in a different location, decoupling image preparation from its deployment and enabling migration. Recent exploration of paravirtualization techniques [8] has led to substantial performance improvements in virtualization technologies, making virtual machines an attractive option for high-performance applications.

Virtual machines allow a client to create a custom execution environment configured with a required operating system, software stack, and access policies and then deploy it on any resource running a hypervisor. Since VMs provide strong isolation, the risk to provider is small. The deployment flexibility stems from the VM concept, which provides an abstract representation of state that can be deployed anywhere a hypervisor is present. In modern hypervisors such deployment is quick: we show that deploying a VM can take less than a second [13], which is comparable to the overhead induced by the Grid tools. In addition to this, hypervisor's ability to provide fine-grain enforcement makes virtual machines an ideal solution for short-term deployment of uniquely configured workspaces requiring controlled resource usage. We note however that VM deployment relies strongly on the availability of a hypervisor of a compatible type; in effect on an underlying deployment of another workspace.

2.3. Nesting Workspaces

As the last section outlines, a workspace requires a specific *deployment capability* (e.g., in the case of a VM: a compatible hypervisor). The deployment of a specific workspace implementation may therefore depend on the deployment of another workspace providing such deployment capability. This enables providers and consumers to find a mutually satisfactory solution to hosting workspaces: provider provides base images configured with hypervisors; then consumers can use deployed hypervisor workspaces to deploy a nesting workspace customized to support the consumer's application. Base images, would still be owned and maintained by a site to reflect site policies. Virtual machines, on the other hand, whose deployment and shutdown can easily be controlled by a site without impairing the availability of its resources, can be configured by communities wishing to support specific applications. A site can still impose restrictions on their configuration, for example by deploying only images attested and versioned by trusted sources (as we explain in Section 4.1), but it is no longer responsible for providing and maintaining complex community-specific configurations.

This strategy can make a significant difference in the variety of environments that a site can support: while it is difficult for a site to maintain a community-specific workspace for every community wishing to work with that site, it is much easier to maintain several generic hypervisor images. Thus, we argue that it is advantageous for sites to support deployment capability platforms for the deployment of VM workspaces rather than end-user capabilities. This also enables a site to react to supply and demand in a more dynamic fashion: while today a site typically simply advertises the supported deployment capability, such advertisements are static and do not allow much flexibility in the support of deployment capabilities.

In general, we can envision supporting many layers of nested workspaces. Such workspaces can be deployed as a result of n-tiered scheduling where the deployment of one layer has a strong dependency on the other. We are working on developing general scheduling algorithms enabling this capability.

3. Virtual Workspace Management

In this section, we describe two different approaches to implementing virtual workspaces: the Workspace Project at the University of Chicago [14] and the Cluster on Demand (COD) Project at Duke University [10]. Both projects provide management of virtual workspaces but emphasize its different aspects and thus develop different interfaces and tools. The COD Project initially focused on physical node imaging; this was recently generalized to virtual machines. The Workspace Project implements management of individual and clustered workspaces using the WSRF protocols while the COD project uses a protocol based directly on SOAP [15]. One of the goals of the Virtual Playgrounds project is to explore the affinities of the interfaces and methods developed by the two projects and establish a common base.

3.1. Workspace Service

We describe virtual workspaces and the supporting architecture in [16]; here, we summarize the key concepts of this technology. At its core is the workspace service, whose protocols are based on the Web Services Resource Framework (WSRF) [17], which provides standard methods for the creation and management of state descriptors, called "WS resources." A *Workspace Factory Service* exposes a "create" operation that allows Grid clients to dynamically request deployment of a customized and isolated Grid execution environment. The create

operation also results in the creation of a WS resource representing the newly deployed environment. Associated with each WS Resource are a limited lifetime and other resource properties that can be inspected, queried, and managed in standard ways, and that can be used in conjunction with WS-Notifications [18] to provide updates on change. A WS resource (and thus also the workspace it describes) can be destroyed either explicitly, or by allowing its lifetime to expire.

Workspaces are deployed based on *workspace meta-data*, provided by the client, which contains all the information necessary for enacting workspace deployment (i.e., in the VM case, VM image and configuration information). Along with the workspace meta-data, the client sends a request for a *resource allocation* that describes resources that should be bound to the workspace at deployment time. Resource allocations describe values such as percentage of CPU assigned, memory and disk size or available bandwidth; their definition and implementation is discussed in detail in [5]. This requested resource allocation is typically expressed in terms of constraints and can be concretized by the service based on available resources. Once a deployment request is accepted, the client can verify the assigned resource allocation (as well as other as well as deployment-time details such as the assigned IP address) by inspection of the WS resource corresponding to the deployed workspace. Further, the client can renegotiate the workspace resource allocation by requesting changes to the workspace resource property.

All workspace service operations are subject to authorization. Since our implementation of the WSRF protocols described above is based on the Globus Toolkit 4 [19] (GT4, we were able to leverage Globus security mechanisms including attribute-based credentials (VOMS [20] as well as Shibboleth-based [21]) and authorization callouts. However, in general workspace deployment requires more sophisticated security methods including vetting and verification of images as well as providing credentials to deployed workspaces. We discuss them in Section 4.1.

The current Workspace Service is implemented as a gateway to a set of resources located within a Trusted Computing Base (TCB) and serving as deployment pool for workspaces. At present we assume that this resource pool has already been configured with the required deployment capability (in our current implementation: the Xen hypervisor) and we use hypervisor-dependent methods on workspace service back-end to implement workspace deployment. We are currently working on providing effective scheduling, monitoring and management tools to map workspaces onto this set of resources. In general, as discussed in Section 2, the deployment of a workspace may take the form of deploying a nested workspace, i.e. first deploying a hypervisor-based workspace and then stacking a VM-based workspace on top of it.

The protocol outlined above fulfills many of our basic design requirements: it provides the control necessary to deploy and shutdown workspaces, request and manage the resource allocation assigned to them, and support desirable behaviors, such as migration (which can be accomplished by pausing and serializing a workspace and restarting it in a different location). Our main current thrust is work on developing techniques to schedule workspaces onto the resource backend effectively, as well as facilitating flexible resource configuration. Using different implementations of workspaces will allow us to implement "tiered scheduling" referred to above.

3.2. Virtual Workspace Clusters

Atomic workspaces can be combined to form virtual clusters [22]. To describe them we extended workspace description to include *aggregate workspaces:* sets of sets composed of atomic workspaces with the same configuration. A combination of such sets can be used to define complex heterogeneous clusters – for example, a typical OSG cluster is composed of two workspace sets: a set containing one or more of service nodes (with service node configuration) and a set of worker nodes (all with the same worker node configuration). To match the resource needs of aggregate workspace, we have defined a corresponding type for resource allocation allowing the user to specify resource allocation for groups of workspaces. An aggregate resource allocation is a set of homogeneous sets of atomic resource allocations (CPU, memory, etc.). For example, an aggregate resource allocation for an OSG cluster might be a set of identical resource allocations (if the requirements for the service node and all the worker nodes are the same), reflect a different resource allocation for the service node and identical ones for the worker nodes, or yet another configuration. The structure of the aggregate workspace type and the aggregate resource allocation need not be the same – differently configured workspaces may require the same type of resource allocation, and vice versa. For the purpose of matching workspaces to resource allocations an ordering has been imposed on both sets.

Virtual Cluster Workspace creation uses the same mechanisms as atomic workspace creation described in the previous section. A workspace is deployed through submission of workspace back-end scripts to local

schedulers (our current implementation works with SLURM [23] and PBS [24]). The first step of workspace deployment involves propagating the images to target deployment: workspace scripts executing on each node download the images from a specified location. To deploy a workspace, the back-end scripts work with the Xen hypervisor and complete the configuration of the workspace. Configuration information that can't be processed by Xen (such as networking) is set up by calling an OS startup script preinstalled in the VM images. After a workspace is deployed, it can be managed through invoking start and stop operations with different parameters to pause/unpause or shut down a workspace. These operations are simply broadcast to all participating nodes.

In [22] we described the results of our initial experiments with virtual clusters. Our experiments with virtual cluster management highlight the importance of treating staging as an important operation that, depending on the granularity of considered deployments, should preferably be independently scheduled. Our experiments with OSG applications are promising: we find that the performance impact of executing in VMs is negligible even for the data-parallel applications and thus virtual clusters are an acceptable platform for this community. Further, we plan to use virtual clusters to provide scientific gateways for TeraGrid applications.

3.3. Cluster on Demand

Cluster-on-Demand is a cluster management service for mixed-use clusters [10]. A COD server partitions cluster resources into isolated virtual clusters, each comprising some set of computing nodes with attached storage resources. COD provides basic services for booting and imaging, naming and addressing, and binding storage volumes and user accounts on a per-virtual cluster basis. It also provides external SOAP interfaces to deploy and control virtual workspaces on virtual clusters.

COD was initially designed to control physical machines with database-driven network booting (PXE/DHCP), in a manner similar to Emulab, Oceano, Rocks, or *bcfg*. The physical booting machinery is now familiar: in addition to controlling the IP address bindings assigned by PXE/DHCP, a COD server for the cluster site (called the *site authority*) controls boot images and options by generating configuration files served via TFTP to standard bootloaders (e.g., *grub*).

As part of the Virtual Playgrounds project, we extended COD to manage virtual machines using the Xen hypervisor [8], as well as physical machines. In addition, we have added facilities to manage resources across multiple cluster sites, to allow composition of virtual workspaces into distributed virtual playgrounds. A COD site administrator may delegate control over specified portions of the cluster resources for a specified period of time to an external management authority (a *broker*). The broker has power to allocate resources across multiple sites in a coordinated way, and is an essential element for extending virtual workspaces to virtual playgrounds, as discussed in the next section. We are currently exploring policy interfaces for brokers, and common resource management APIs that allow resource consumers to negotiate with brokers to obtain resource to adapt to dynamic resource demands [25].

Resource consumers invoke SOAP [15] interfaces on each COD authority to configure the workspaces at each site, in the style previously described. The COD site authority accepts meta-data attributes passed with the create request for the virtual workspaces, and interprets them to drive configuration. An important goal of the Virtual Playgrounds project is to construct a common configuration formalism that is sufficiently rich to express a full range of resource configuration requirements, and amenable to static checking for internal consistency and conformance to site policies and security policies.

To enable flexible control over configuration, we are experimenting with a scriptable back-end virtual workspace manager in COD. As resources are added or removed from a virtual workspace, a resource-specific *setup* or *teardown* handler is invoked within the authority server. To represent the wide range of configuration actions that may be needed in production clusters, the handlers are scripted using Ant, an open-source OS-independent XML scripting package. Ant scripts invoke a library of packaged tasks for remote command execution and network management, and to build, configure, deploy, and launch software packages on various operating systems and Web application servers. Ant is in wide use, and new plug-in tasks continue to become available. The scripting architecture may also be used by the resource consumer to specify actions that take place when a new resource (e.g., a virtual machine) *joins* or *leaves* a workspace.

Like the virtual workspace servers, the Ant tasks and the Ant interpreter are written in Java, so the *setup/teardown* drivers invoke the Ant interpreter directly. Java exception handling is a good basis for error detection, reporting, attribution, and logging of configuration actions scripted with Ant. Our continuing work is investigating methods to select appropriate response, notification, and repair actions

for failures during configuration, or in operating virtual workspaces.

The scripted post-configuration architecture will allow COD to interoperate with other widely used cluster management tools. For example, as an initial step, we have instantiated automatically PBS and SGE batch schedulers and an operating Globus grid—with a GRAM/PBS head node with GridFTP staging space exported to the PBS worker nodes via NFS—in a Xen virtual cluster.

We emphasize that COD is intended to facilitate the use of such middleware where desired, and not to replace it. COD enables flexible, safe affiliation with external grids by encapsulating grid services in virtual clusters whose sizes and access to local resources are constrained by site policies, to allow a suitable degree of isolation and control over how local resources are used.

The combination of support for both physical and virtual machines offers useful flexibility: it is possible to assign blocks of physical machines dynamically to boot Xen hypervisors, and then add them to a resource pool for dynamic instantiation of virtual machines. Another area of continuing work is to extend the external interfaces to allow broker policies greater control the resources assigned to each virtual machine.

A COD site authority drives cluster reconfiguration in part by manipulating data stored in a back-end directory server with the Lightweight Directory Access Protocol (LDAP) [26]. The COD LDAP schema extends the RFC 2307 standard for an LDAP-based Network Information Service. Standard open-source services exist to administer networks from an LDAP directory server. The DNS server for the site is an LDAP-enabled version of the standard BIND9, and for physical booting we use an LDAP-enabled DHCP server from the Internet Systems Consortium (ISC). In addition, guest nodes in a virtual workspace have read access to an LDAP subtree describing the containing virtual cluster. Guest nodes configured to run Linux use an LDAP-enabled version of AutoFS to mount NFS file systems, and a PAM/NSS module that retrieves user logins from LDAP. This facilitates control over user account sets and storage configurations at the granularity of virtual workspaces or clusters, or for different sets of nodes within a virtual cluster.

COD should be comfortable for cluster site operators to adopt, especially if they already use RFC 2307/LDAP for administration. The directory server is authoritative: if the COD site authority fails, the disposition of the cluster is unaffected until it recovers. Operators may override the COD server with tools that access the LDAP configuration database directly.

4. From Workspace to Playground

A *Virtual Playground* (VP) may be obtained by composing multiple workspaces along with other virtual resources, such as networks or storage, relating to the VP as a whole. In this sense, Virtual Playgrounds are a generalization of the workspace concept but are not necessarily confined to one TCB and thus must implement the complex trust relationships as well as consider the performance implications of executing in a wide-area network. Furthermore, unlike workspaces, which focus on deployment, VPs take a holistic view of the system including the "hidden costs" of virtualization such as image distribution and replication.

In this Section, we present a brief outline of research directions arising in this context.

4.1. Security

The problem of division of labor between providers and consumers that workspaces are expected to solve determines the trust relationships they must enable: a workspace must not maliciously misuse the hosting resource and the resource must not jeopardize data or computations taking place inside the workspace. From the provider's point of view it is thus important that their resource be used consistently with site policies. This can be obtained by vetting a workspace (either by the resource provider himself or a party it trusts) and verifying the image on deployment. From the consumer's point of view, it is necessary that its trust relationship with a deployed workspace be rooted both in the workspace itself and the provider who deployed it.

In order to satisfy both consumers and providers constraints the Workspace Project developed an image management model and a workspace certificate authority that, in conjunction, cover those constraints [16]. We focus our efforts on virtual machines. Thus we developed a model whereby each virtual machine is represented as a set of partitions describing different aspects of its configuration that can be combined to define its image. These partitions may include a variety of components such as a system module, a community-specific configuration partition, or an application or data partitions. These partitions can be attested by different parties depending on their provenance, annotated with versioning information, and associated with different confidentiality levels (for example, a system partition may need to be only validated while a

data partition containing sensitive data may require encryption).

We modified workspace meta-data to include information sufficient to reconstruct the image from such partitions and equipped the workspace service with mechanisms allowing for partition validation and decryption. These operations can be costly, but may not be significant in practice since it is reasonable to assume that specific, often used, partitions will be pre-fetched into the TCB of a site.

In order to resolve the second issue – providing a basis of trust establishment between a deployed VM and its clients – we equipped the workspace service with the ability to generate proxy certificates based on its own host certificate. Each workspace has been assigned a name that is a part of its (attested) meta-data. After validating workspace information the workspace service generates a proxy certificate attesting to its name and deployment site. The advantage of this method is that it results in information rich certificates and allows for fine-grain trust relationships. An alternative is to pre-generate a host certificate for the workspace, place it on an encrypted partition, and create policies on where the workspaces may or may not be deployed. We are in the process of investigating the trade-offs between the two methods.

4.2. Networking

Shared campus clusters and grids often force users to adapt the way they work to use computing resources in another administrative domain. A key issue is how leased resources are integrated into the network IP address space. Domains often manage their own LAN segments and IP subnet space; these are frequently firewalled at the departmental level, and private IP address spaces are often used to reduce consumption of scarce public IP space.

Naming issues often interfere with resource access and sharing. If nodes in the workspace are assigned IP addresses from the resource provider's space they may be unable to access internal resources such as network storage in the consumer's home domain, since these are often exported only to the internal IP space. Access to licensed software is sometimes controlled by lower-level MAC addresses. Regardless of the access control policy, use of private IP name spaces in either the provider domain or consumer domain, or both, makes communication impossible without NAT bridging.

We are working to address these issues with reconfigurable virtual networks based on a hybrid of dynamically configured VLANs and SSL tunneling. The current COD prototype addresses these problems in a very limited way: it assigns private IP subnets to slices, which share a common LAN segment, with a limited number of public IP addresses available (e.g., for ``head nodes'') in a common externally visible subnet. One goal of our ongoing work is to integrate remote virtual workspaces into the home addressing domain of the consumer on a temporary and dynamic basis, to allow for data and resource sharing within the domain, without compromising isolation from external attack or data snooping.

4.3. Resource Management

Resource management, understood as combining resource from both different domains and playground aspects (computational resources, storage, networking, etc.) is one of the main thrusts of the project. Our ongoing thrusts are in fine-grained resource management and scheduling [5], multi-tiered provisioning and scheduling (see Section 3), investigation of methods resource assignment and scheduling developed in the context of incentive-based systems such as Tycoon [27, 28], and co-scheduling multiple aspects (e.g., combining network reservations with resource reservations).

5. Impact

We are actively working with several communities and projects to see our ideas tried out in practice and to bring current requirements and feedback to bear on the direction of our research. We were fortunate to establish collaborations with the TeraGrid and OSG projects; many of our collaborators have deployed or are in the process of deploying the infrastructure developed as a result of our research.

The most progress so far has been made by OSG. Specifically, the Workspace Service has been deployed by the Edge Services Framework (ESF) developed by the OSG in order to decouple the process of configuring and managing service nodes for Virtual Organization from providing resources. Current ESF deployment sites include ANL, FNAL, University of Chicago, UCSD and SDSC. While the deployment is still in preliminary stages, the experiences to date are encouraging and the community response has been very positive.

7. Summary

We have described the abstraction of a virtual workspace, a customizable execution environment for Grid environments that can be deployed dynamically

and consume resources in controllable ways. The Virtual Playgrouds project develops methods to deploy and manage such workspaces and aims to establish a reliable framework for their management in the Grid. This paper surveys our research efforts to that end.

8. References

1. Foster, I. and others. *The Grid2003 Production Grid: Principles and Practice*. in *IEEE International Symposium on High Performance Distributed Computing*. 2004: IEEE Computer Science Press.
2. *Open Science Grid (OSG)*. 2004: www.opensciencegrid.org.
3. *The TeraGrid Project*.
4. Foster, I., K. Keahey, C. Kesselman, E. Laure, M. Livny, S. Martin, M. Rynge, and G. Singh, *Embedding Community-Specific Resource Managers in General-Purpose Grid Infrastructure.* White Paper, 2005.
5. Keahey, K., I. Foster, R.D. Freeman, A. Rana, B. Sotomayor, and F. Wuerthwein, *Division of Labor: Tools for Growth and Scalability of the Grids.* White Paper, 2006.
6. Foster, I., C. Kesselman, and S. Tuecke, *The Anatomy of the Grid: Enabling Scalable Virtual Organizations.* International Journal of Supercomputer Applications, 2001. **15**(3): p. 200-222.
7. Youssef, S., *Pacman: A Package Manager*. 2004: http://physics.bu.edu/~youssef/pacman/.
8. Barham, P., B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebar, I. Pratt, and A. Warfield. *Xen and the Art of Virtualization*. in *ACM Symposium on Operating Systems Principles (SOSP)*.
9. Desai, N., A. Lusk, R. Bradshaw, and R. Evrard. *BCFG: A Configuration Management Tool for Heterogeneous Environments*. in *IEEE International Conference on Cluster Computing (CLUSTER'03)*. 2003.
10. Chase, J., L. Grit, D. Irwin, J. Moore, and S. Sprenkle, *Dynamic Virtual Clusters in a Grid Site Manager.* accepted to the 12th International Symposium on High Performance Distributed Computing (HPDC-12), 2003.
11. Youssef, S., *Personal communication*. 2004.
12. Goldberg, R., *Survey of Virtual Machine Research.* IEEE Computer, 1974. **7**(6): p. 34-45.
13. Keahey, K., I. Foster, T. Freeman, X. Zhang, and D. Galron, *Virtual Workspaces in the Grid.* ANL/MCS-P1231-0205, 2005.
14. *Virtual Workspaces*: http://workspace.globus.org.
15. W3C, *SOAP*. 2002: http://www.w3.org/TR/SOAP.
16. Lu, W., T. Freeman, K. Keahey, and F. Siebenlist, *Making your workspace secure: establishing trust with VMs in the Grid.* SC05 Poster Presentation, 2005.
17. Czajkowski, K., D. Ferguson, I. Foster, J. Frey, S. Graham, I. Sedukhin, D. Snelling, S. Tuecke, and W. Vambenepe, *The WS-Resource Framework*. 2004: www.globus.org/wsrf.
18. Foster, I., J. Frey, S. Graham, S. Tuecke, K. Czajkowski, D. Ferguson, F. Leymann, M. Nally, T. Storey, and S. Weerawaranna, *Modeling Stateful Resources with Web Services*. 2004, Globus Alliance.
19. Foster, I., *Globus Toolkit version 4: Software for Service-Oriented Systems.* IFIP International Conference on Network and Parallel Computing, 2005.
20. *The Virtual Organization Management System*: http://infnforge.cnaf.infn.it/projects/voms.
21. Welch, V., T. Barton, K. Keahey, and F. Siebenlist. *Attributes, Anonymity, and Access: Shibboleth and Globus Integration to Facilitate Grid Collaboration*. in *4th Annual PKI Research and Development Workshop*. 2004.
22. Zhang, X., K. Keahey, I. Foster, and T. Freeman, *Virtual Cluster Workspaces for Grid Applications.* ANL/MCS-P1246-0405, 2005.
23. Yoo, A.B., M.A. Jette, and M. Grondona, *SLURM: Simple Linux Utility for Resource Management*, in *Job Scheduling Strategies for Parallel Processing*, L. Rudolph and U. Schwiegelshohn, Editors. 2003, SpringerVerlag. p. 44-60.
24. *Portable Batch System*. 2003: http://www.openpbs.org.
25. Irwin, D., J. Chase, L. Grit, A. Yunerefendi, D. Decker, and K. Yocum, *Sharing Networked Resources with Brokered Leases*. 2006: in submission, available at http://issg.cs.duke.edu/publications/sisyphus.pdf.
26. Howes, T.A., *The Lightweight Directory Access Protocol: X.500 Lite.* 95.

27. Lai, K., L. Rasmusson, E. Adar, S. Sorkin, L. Zhang, and B. Huberman, *Tycoon: an Implementation of a Distributed Market-Based Resource Allocation System.* Technical Report arXiv:cs.DC/0412038, 2004.
28. Irwin, D., J. Chase, L. Grit, and A. Yumerefendi, *Self-Recharging Virtual Currency.* Proceedings of the Third Workshop on Economics of Peer-to-Peer Systems (P2P-ECON), 2005.