Scientific Programming 13 (2005) 265–275
IOS Press

Virtual workspaces: Achieving quality of service and quality of life in the Grid

K. Keahey[a,b], I. Foster[a,b], T. Freeman[b] and X. Zhang[b]
[a]*Math & Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, USA*
[b]*University of Chicago, Chicago, IL 60637, USA*

1. Introduction

By defining standardized protocols for discovering, accessing, monitoring, and managing remote computers, storage systems, networks, and other resources, Grid technologies make it possible – in principle – to allocate resources to applications dynamically, in an on-demand fashion [19]. However, while Grids offer users access to many diverse and powerful resources, they do little to ensure that once a resource is accessed, it fulfills user expectations for *quality of service* (QoS). The problem is that most Grid platforms today do not support performance isolation: activities associated with one user or virtual organization (VO) [17] can influence the performance seen by other processes executing on the same platform in an uncontrolled way. This inability to enforce QoS and provide execution guarantees prevents Grids from being useful in a range of scenarios. Since there is no way to enforce resource usage, we cannot guarantee it and thus we cannot use Grids for reliable future use or time-critical applications.

Another serious issue is that while Grids provide access to many resources with diverse software configurations, a user's application will typically run only in a specific, customized software environment. Variations in operating systems, middleware versions, library environments, and filesystem layouts all pose barriers to application portability. Applications that work on a developer's desktop may function "out of the box" on only a small fraction of the total number of compute resources potentially available to the scientist. Thus, in practice, users are often unable to leverage the available resources without tedious debugging and porting efforts that drain time and energy from their primary objectives – thus compromising their *"quality of life"* in interactions with Grid software.

The ability to provide a perfect execution environment for every set of requirements, with arbitrary enforcement and configuration properties, is an elusive goal. Although achievable in some cases – for example, by using technologies such as virtual machines [23] – it is not universally available and typically involves tradeoffs in one form or another. However, an application can abstract the properties of an environment it requires for execution in terms of memory size, the number of processors, or library requirements and environment variable settings, and a resource can define its offered capabilities in similar terms. Such environment abstractions can be implemented by using a variety of sandboxing technologies and can be mapped onto resources, offering the required degree of isolation, fine-grained enforcement, and configuration.

We have previously introduced [27] the *virtual workspace* (VW) abstraction to describe such environments and showed how this abstraction can be implemented by using virtual machines. Here, we provide a more comprehensive description of virtual workspaces, discuss the different ways in which they can be implemented, and show how they can be used to build layered deployment environments in the Grid. We discuss how a workspace deployed on a physical resource can provide a deployment capability for setting up virtual machines, and we describe how matchmaking, with negotiation and advance reservation protocols [3], can be used to create complex workspaces.

ISSN 1058-9244/05/$17.00

The paper is organized as follows. Section 2 gives an overview of virtual workspaces. Section 3 describes their implementation, Section 4 is devoted to services we developed and adapted to support workspaces in the Grid, and Section 5 describes the agreement-based interactions that the notion of a workspace enables. We discuss related work in Section 6 and conclude in Section 7 with a brief look at future work.

2. Introducing virtual workspaces

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, memory share) as well as its software configuration (e.g., operating system installation, provided services). For example, a physical machine configured as an ATLAS Grid 3 node using the Pacman configuration software [43], a virtual cluster created with a Cluster on Demand (COD) software [9], a cluster of virtual machines configured with the software configuration required by Open Science Grid (OSG) [16], and a set of physical machines configured with Xen hypervisor [7] all represent a workspace.

Different workspace implementations provide different workspace management capabilities [29]. In this section, we discuss various workspace implementation approachs and the extent to which they can be made available dynamically in terms of flexibility of their configuration and efficiency of deployment.

2.1. Workspaces as site-provided installations

Nodes on site installations, often configured to support the needs of specific communities such as Open Science Grid [34] or TeraGrid [8], can provide workspaces to Grid clients. One way of providing workspaces relevant to a specific community is to obtain a priori agreement on required configurations, and then simply deploy those configurations, advertise them, and provide access. This approach is often practiced today. A more flexible way is to allow for automated boot of physical nodes based on a configuration description and boot images (as implemented for example in bcfg [13]), thus allowing a site to manage its nodes in a more flexible and controlled manner. The Cluster on Demand (COD) [9] project pioneered the use of such tools to provide external access to configured physical nodes on-demand, using credential-based authorization. Base configurations can be further refined using configuration management software such as Pacman [43] to deploy pre-defined software configurations automatically. Pacman is able to check the installation against already available software to optimize the process. Thus, one can imagine scheduling the deployment of a Pacman workspace based on the availability of already installed software.

The final step in making a site-provided installation available as a workspace is providing access to Grid clients. Such access is typically provided by the use of dynamic accounts [24,26,30,33,40]: accounts generated on the fly or assigned from a pre-generated pool. The current workspace implementation leveraging site configurations [5] uses this method based on the LCMAPS adaptation [38] of account pool implementation [33]. In addition to workspace creation, this implementation provides a flexible interface for access management and policy inspection: the length of an account lease can be determined and renegotiated based on need. On lease termination the account is cleaned to extent determined by site account configuration policies. The account cleaning process is configurable by the site administrator and, for additional security, includes an optional quarantine: accounts are not returned to the pool until a specific event takes place, such as the expiration of a certain time period or an action by a system administrator.

While it is possible to exercise some control over software configurations of workspaces, enforcing fine-grain, reliable resource allocation and sharing remains problematic. Typically resources are made available in a relatively coarse-grained manner allocating the number of nodes or amount of available disk space but not enforcing fine-grained quotas, such as percentage of CPUs between users of a specific resource. Some control can be provided at the Unix account level by regulating the amount of resource assigned to individual accounts by using additional tools (e.g., "setrlimit," "quota," "chroot") or schedulers such as DSRT [18], PBSPro [35], or Condor [22]. Since these tools are not widely deployed the reliance on them can only be best-effort and rarely provides the required degree of enforcement.

In summary, while providing access to site-configured workspaces is a fast and widely accepted solution it lacks flexibility in that it offers little choice in shaping the workspace's configuration or the resource allocation associated with it. (The physical machine essentially becomes the resource enforcement unit.) Even when allowing full [9] or partial [43] control over the

configuration on a physical resource, the choice is limited to a library of site-configured images. The greater the amount of configuration done at workspace creation time, the greater the flexibility of the solution, but also the greater the time required for workspace deployment. Base image deployment typically takes many minutes; the time then required to deploy and configure application-specific software can be substantial (for example, the time required to install the software used by the Atlas project [6] can take a few hours [44]) Given these considerations, deploying workspaces on physical nodes is cost-effective primarily for widely-applicable and long-lived workspaces that do not require fine-grained enforcement.

2.2. Workspaces as virtual machines

A virtual machine (VM) [23] 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. A VM representation (VM image) is composed of a full image of a VM RAM, disks (or partition) images, and configuration files. Recent exploration of paravirtualization techniques [7] 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. Further, VM state may be serialized into a VM image, allowing the client to pause or shut down VM operation, and resume it at a different time and in a different location, decoupling image preparation from its deployment and enabling migration. In addition, virtual machines offer excellent enforcement of resource usage: typically, a virtual machine is configured with a specific memory and disk size and some, such as [7], allow those qualities to be managed during deployment. Using schedulers, such as [42], a client can assign a percentage of CPU to a given virtual machine effectively regulating the CPU usage of the group of processes encapsulated in it. For these reasons, VMs provide an excellent implementation option for workspaces: the configuration of a VM image can reflect a workspace's software requirements while the hypervisor can ensure the enforcement of hardware properties.

The virtual workspace implementation based on virtual machines uses VM images configured at the time of workspace creation to represent an execution environment. Where VM disk is represented as a set of partition images (as in Xen [7]) a configuration can be assembled by simply putting suitably configured partitions together (while carefully respecting software dependencies). Depending on the installed software, virtual machine images can be large. Thus, assembling an image by partitions is also useful at the time of workspace deployment – partitions containing software installations frequently used at a given site can be stored locally instead of transferred at workspace deployment time. In the future, we plan to explore using the Replica Location Service (RLS) [10] to keep track of and manage copies of VM images associated with specific workspaces.

Overall, VMs have the advantage of both flexibility and speed of deployment. The 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 [27], 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. Virtual cluster workspaces

Workspaces implemented via any of the methods just described can be grouped to create virtual clusters of various topologies. A *virtual cluster workspace* can be constructed via, for example, the Cluster-on-Demand (COD) infrastructure [9], an existing cluster with tools for dynamically enabling access, or a cluster of virtual machines.

When using a virtual machine implementation, a virtual cluster workspace is implemented in terms of multiple VM images that may represent specialized nodes such as worker or head nodes. In representing clusters, we can leverage the fact that some groups of nodes (for example, worker nodes) have the same or similar configuration and leverage this fact to optimize their representation and deployment time. We provide a detailed discussion of these topics elsewhere [45].

3. Describing virtual workspaces

A workspace description should contain sufficient information for a deployment service to create the environment represented by this workspace. This information is of two kinds:

(1) description of packages or other data that need to be obtained from potentially external sources and put together (such as a software installation package or a VM image), and
(2) deployment logistics information, which needs to be interpreted and configured at deployment time (such as network connection configuration for a VM).

The quantity of information that must downloaded and the amount of deployment-time configuration will depend on both workspace implementation (installation-based deployment versus deploying a VM image) and the deployment service implementation (deploying a VM based on a pre-configured image [28] or refining configuration at deployment time [31]). Thus, a wide range of approaches to workspace description are possible, from the simple but inflexible (e.g., a pointer to a VM image and a default deployment configuration) to the complex but powerful (e.g., arbitrary on-the-fly configuration of an image). In this section, we describe the configuration aspects of a workspace definition – workspace meta-data – in the context of the schema we developed for virtual machine representation of workspaces.

We adopt an approach that strikes a balance between definition flexibility and deployment-time configurability and speed. We describe a workspace using the XML schema depicted in block form in Fig. 1. The description associates a workspace *name* with its definition and deployment logistics information. The workspace name is a uniform resource identifier (URI) that can be resolved to obtain more information about the workspace, such as its provenance, creation and modification times, or detailed software catalog. While irrelevant to deployment, this information is valuable to a workspace client and can be made available by services such as the Handle System [11] that additionally allow for policy-controlled release of this information.

We represent a workspace as a collection of modules that can be combined to define the workspace's content. These modules may include a variety of components such as a system module, a community-specific configuration layer, or an application module. These different components can be obtained at different times and from different sources, thus optimizing VM image transfer. Further, such modules can be attested by different parties depending on their provenance, annotated with versioning information, and associated with different operational modes (such as read-only). Thus, the *definition* section of the workspace description contains a definition of workspace modules together with information describing how they should be obtained and put together, as well as *deployment information* describing how the workspace should be configured on deployment. In order to enable verification prior to deployment, the definition section additionally specifies *prerequisites* for workspace deployment, such as CPU architecture, hypervisor, kernel images and versions, and kernel parameters.

With VMs, workspace modules can be implemented by VM partitions or disks that make up a VM image (see Section 2.2). The deployment service must be able to resolve how to acquire these partitions, verify their integrity, and instantiate the VM such that the proper image is loaded by the VM at the expected device. Thus, for each individual module we specify a *module name*, the *location* at which the module can be obtained, the *device binding* to which it should be bound, and modes of operation (*permissions*). Like the workspace metadata, a partition name is a URI that can be used in conjunction with other services to obtain more information about a specific partition including dependencies on other partitions. A partition location can be resolved to locate and transfer an actual image partition. We are currently creating an additional *attestation element* to describe who developed a partition and attested to its content.

To complete VM instantiation, we also need to describe deployment information on how to provide networking and (potentially) additional storage. This content is contained in the *deployment* section of the workspace. To reflect that fact that a VM can have an arbitrary number of network interfaces (NICs) that are mapped to physical hardware in different ways, networking is described as a collection of NIC elements. For each NIC, we describe *naming* (the method used to obtain an IP address), *network binding* describing how VM's network interfaces are bridged and managed outside the VM, and the IP address. The network binding can be bridged directly to a physical NIC, bridged to a VPN, configured behind a NAT of one of the host machine's IP addresses, or configured to use an isolated LAN.

Atomic workspaces, describing one execution environment, can be combined to form aggregate

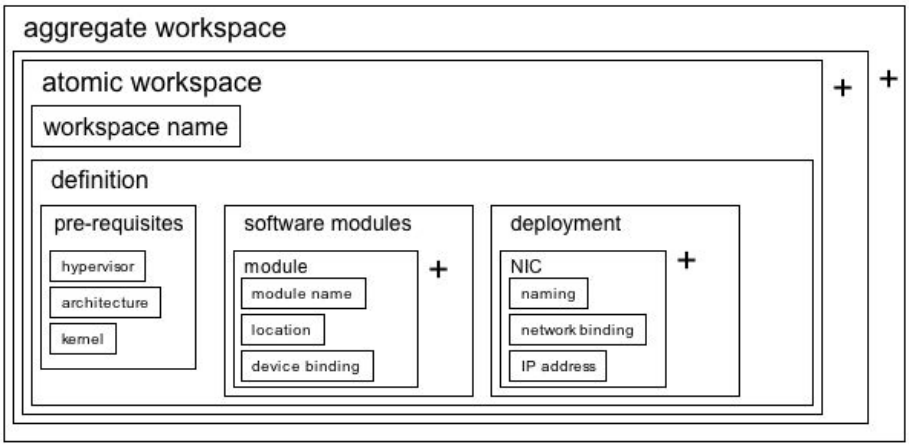

Fig. 1. Graphical representation of the workspace schema. Plus signs denote "one or more elements".

workspaces such as virtual clusters. An aggregate workspace is defined to contain sets of homogeneous atomic workspaces. This approach allows us to define heterogeneous clusters composed for example from a head node and worker nodes (one atomic workspace and a set of homogeneous atomic workspaces). To represent more complex, hierarchical structures the aggregate workspace can easily be made more recursive in future workspace implementations. All information about a cluster workspace is derived from the metadata of the atomic workspaces describing those nodes.

While workspace metadata describes where VM image parts can be obtained, it does not include them. Images as well as workspace descriptions can be put together by VO deployment teams supporting specific groups of applications. Such workspaces can then be shared, copied, and incrementally refined, allowing users to further customize VWs to suit a particular set of needs. A community may provide a configuration service supporting those operations and allowing for more or less workspace customization. VMPlant [31] implements such a service, which however combines the process of VM configuration with its deployment. In our architecture those roles are split: once put together, a workspace description may be deployed many times, each time potentially with a different resource allocation.

4. Creating, deploying, and managing virtual workspaces

The workspace description introduced in the previous section describes a workspace's environment. A workspace defined in this way may be deployed in the context of many different resource allocations. Furthermore, these allocations can be renegotiated during a specific deployment. We use the term Workspace Service to denote the entity responsible for deploying a workspace and associating it with a specific resource allocation. We describe this service here, as well as the context in which a Workspace Service operates and the means by which it interacts with different Grid services to accomplish its goals.

Figure 2 illustrates these services and their function. To obtain a workspace description, a client first works with either workspace configuration services (Section 3) to create a new workspace, or selects from existing workspaces using an information service that associates workspace images with information about those images and the deployment capability (services) they provide.

Workspace deployment comprises two phases: obtaining the workspace data required for deployment and using that data to deploy the workspace onto a requested set of resources. Implementing the first phase (workspace staging) may require orchestrating workspace data transfer from remote sources to the site where the workspace will be deployed. Efficiently scheduling such transfers can require global knowledge and coordination. In the second phase, the workspace data is available at the selected site and the workspace can now be deployed using this site's workspace service. In order to do this, the workspace service must be able to interpret the workspace description and be in control of resources that provide the deployment capability for this kind of workspace (e.g., a hypervisor for virtual machines).

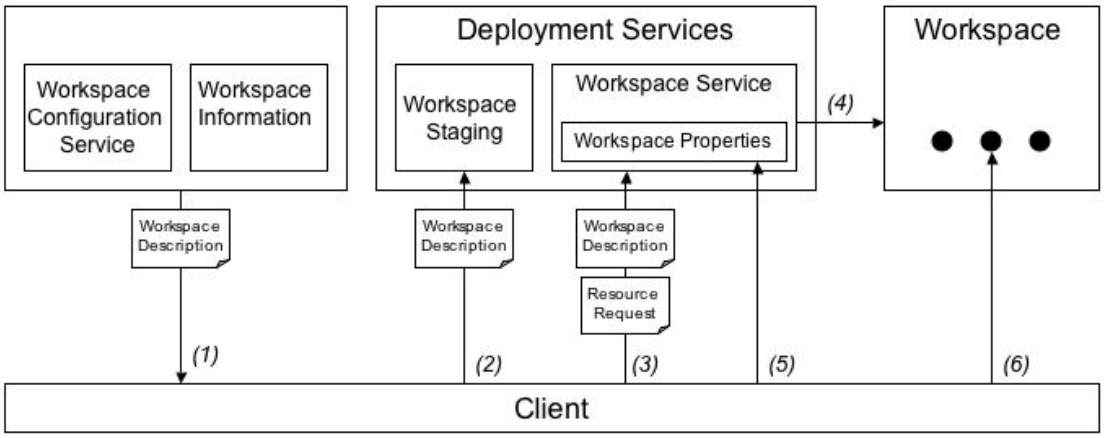

Fig. 2. Creating and deploying workspaces.

At deployment, a workspace is associated with a resource allocation requested by the client. This resource allocation, as well as other deployment-specific properties of the workspace, can be managed (i.e., monitored and/or modified) throughout its deployment. All workspace service operations are subject to authorization; authorization policy may be expressed in terms of both the requesting client and the workspace itself. Once the workspace is deployed, an authorized client can interact with services within the workspace. Note that at this point, the client is authorized by services executing *within* the workspace. By the act of deploying the workspace and associating it with a resource allocation, the workspace service effectively delegates to the workspace the use of this allocation.

Work on other services is pursued elsewhere; here we focus on the Workspace Service. We define Workspace Service protocols based on the Web Services Resource Framework (WSRF) [21], which provides standard methods for the creation and management of manageable state descriptors called "WS resources." A *Workspace Factory Service* uses a "create" operation to create a WS resource representing the new workspace. 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 [21] to provide updates on change. A resource can be destroyed either explicitly or by allowing its lifetime to expire. Figure 3 shows the resource properties we defined for a workspace resource, as well as the operations and arguments that we define for a workspace service in addition to resource management as described above.

The Workspace Factory "create" operation takes as input a workspace description (as described in Section 3) and a description of a resource allocation to be applied to the workspace on startup. The resource allocation request contains a request for association with a resource allocation for a specified length of time. The resource allocation is specified as constrained values that are bound to specific values when the request is accepted. Deploying an aggregate workspace requires describing an aggregate resource allocation reflecting the topology of the specified workspace: the aggregate resource type is specified based on an atomic resource type defined analogously to aggregate workspaces. In addition, the resource allocation request allows a client to specify a state that the workspace deployment should reach on creation of the resource. This element indicates whether the workspace should be started immediately or whether it should be only prepared for startup (for example, images associated with a workspace should be propagated to the actual resource on which they will be deployed). The last option was introduced based on the observation that preparing a workspace to start may take a relatively long time [45].

In addition to the operations used to manage the WS resource, we define operations that allow a client to start a workspace and to shut down a started workspace. Workspace shutdown takes an argument that allows the client to express properties of the shutdown, such as pause, pause and serialize, hard-reboot, and trash (shut down and destroy images). A workspace may be started and shut down many times throughout the lifetime of the WS resource associated with it.

The properties of workspace deployment can be inspected and managed through the WS resource proper-

Types:

AtomicResourceAllocation_Type:
- CPUType
- CPUPercentage
- PhysicalMemory
- NIC []
- DiskSpace

NIC_Type:
- IncomingBandwidth
- OutgoingBandwidth

WorkspaceDeployment_Type:
- DeploymentTime
- ResourceAllocation
- WorkspaceState

Workspace Service Factory

Resource Properties:
1. Workspace pre-requisites

Operations:
- Create(AggregateVirtualWorkspace, AggregateResourceAllocation_Type)

Workspace Service

Resource Properties:
1. AggregateWorkspaceDeployment_Type
2. Workspace deployment information
3. Workspace state

Operations:
- Start (no inputs)
- Shutdown (shutdown inputs)

Fig. 3. Workspace service interfaces.

ties. The resource properties of a workspace resource describe the resource allocation that the workspace is currently bound to, the state of the workspace (propagated/unpropagated – referring to the readiness for execution, running, paused, trashed). A client may use the resource properties to adjust the resource allocation given to a workspace using the WSRF SetResourceProperties operation. Although in the future negotiation could be used for this operation, we currently accept or reject the request based on feasibility.

Our implementation of the protocols described above is based on the Globus Toolkit 4 (GT4) implementation of WSRF [20]. This use of GT4 allows us to leverage many tools available in the Globus Toolkit such as authentication/authorization mechanisms and persistence. The workspace service implements a gateway to a set of resources administered by it: the workspace may be deployed on a different host from the one on which it is installed.

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). However, we expect these interfaces to evolve significantly in the future to support negotiation, more sophisticated agreement structures (such as WS-Agreement [3]), and more refined resource allocation strategies and definitions (potentially based on JSDL [4]). As these definitions evolve, so will the associated resource properties and operation definitions. Another driving force in their evolution will be generalizing these services to support other workspace implementations, which is desirable from the perspective of providing uniform ways of interacting with the same concept, no matter how that concept is implemented. We currently define for example what kinds

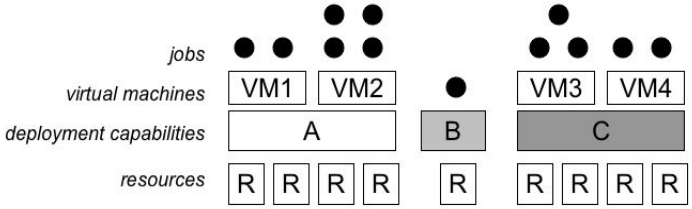

Fig. 4. Environment layers: A and C are hypervisor workspaces, each supporting the deployment of different types of virtual machines, which in turn provide job execution capabilities. Workspace B provides job execution capability.

of workspaces can be deployed by a specific service: more such definitions may be necessary to make the interfaces truly generic.

5. Putting it all together: Workspaces, agreements, and brokers

We have described the different implementations of workspaces and, based on the example of virtual machines, shown how they can be described and deployed. Our motivation in defining a workspace was to provide a required execution environment for an application. However, we note that a workspace itself may constitute an application with well-defined requirements (see workspace pre-requisites as in Section 3). While virtual machines provide a compelling solution for environment deployment, their own deployment, as well as its scope, relies on the presence of compatible hypervisors on a resource. In this section, we discuss how different implementations of workspaces can be put together to implement the full stack of interactions required for workspace deployment.

A workspace is associated with a deployment capability. As the figure below illustrates such capabilities can support the deployment of jobs or other workspaces. The Workspace Service, implementing workspace deployment, can operate at each layer; however, at each layer, workspace deployment is associated with different implementations and ownership.

Base images, deployed on resources owned and maintained by a site, are typically configured by a site owner 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, 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.

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. If, in practice, one dominant capability is required by the site users (such as a specific Linux workspace or a popular hypervisor) this model is sufficient. However, systems such as COD [9] offer a more flexible model that (in effect) implements site-specific workspace services that allow an authorized client to request the deployment of a workspace from a limited set. Providing standardized interfaces to such services would enable a client to flexibly negotiate required workspaces at different sites.

These interfaces are likely to be based on emerging standards such as WS-Agreement [3]. Figure 5 shows how the full workspace deployment could be negotiated using WS-Agreement. A broker negotiates an agreement with a site for the availability of a set of resources and obtains a resource allocation agreement. Workspace deployment requests, defining a deployment capability (for example, a specific hypervisor configuration) and a resource allocation request, can now be made against that resource allocation agreement. If the request is successful, a workspace deployment is bound to the available resource allocation, and the hypervisor configuration is deployed on the requested resources. This agreement in turn can be used as a base for subsequent deployment of virtual machine workspaces. Throughout this interaction the site or

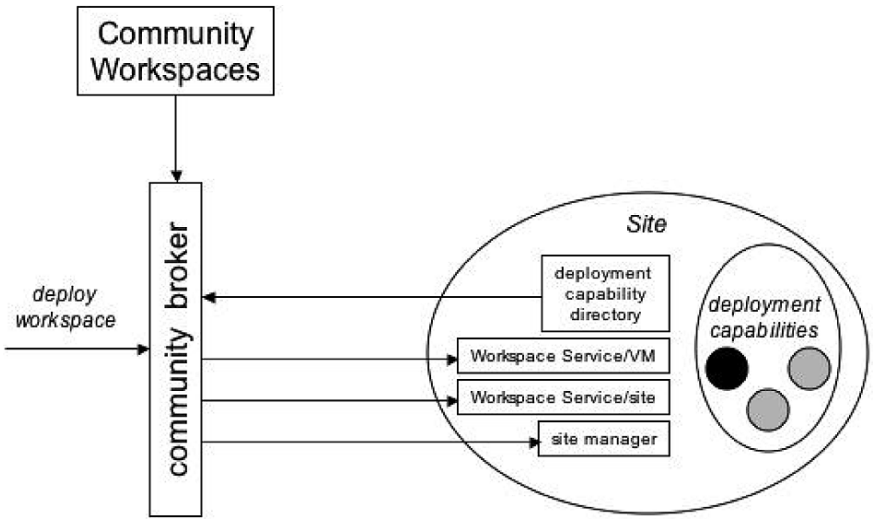

Fig. 5. Negotiating workspaces in the Grid.

community may advertise supported deployment capabilities or even offer them as bids [14] through Grid economic mechanisms.

It should be noted that it is the deployment capability that is advertised by a site: in addition to making resources available we also need to be able to configure them with a software providing the required capability. The power of using virtual machines relies on the fact that while sites can, in principle, maintain configuration images required by many different communities, in practice only a few images can be maintained. However, maintaining a few generic hypervisor images gives the sites a platform that can be used to deploy a large variety of environments.

6. Related work

Because of their superior isolation properties, fine-grained resource management, and ability to instantiate independently configured guest environments on a host resource, virtual machines are attracting increased attention in distributed computing [15,29]. The Xenoserver project [37] is building a distributed infrastructure based on the popular and efficient Xen virtual machine [7]. The In-Vigo project [1,31] proposed a distributed Grid infrastructure based on the use of virtual machines as resources, while the Virtuoso [39] and Violin [25] projects explore networking issues related to using virtual machines in a Grid environment. Likewise, much research has been invested in tools for resource configuration [2,13,14,32] and allocation as well as dynamically providing access for Grid clients to existing workspaces [24,26,30,33,40]. COD [9] enables dynamic workspace deployment for authorized remote clients.

Our contribution differs from these efforts in two ways. First, unlike Krsul *et al*. [31], we seek to separate deployment-independent workspace configuration services on the one hand, and a workspace deployment service that binds a workspace to a specific resource allocation on the other. Second, we believe that the workspace abstraction is not limited to a single virtual machine implementation or site-specific installation mechanisms but can be generalized to encompass a range of approaches. Driven by the requirements of diverse Grid communities working with the Globus Alliance [41], we also seek to define methods for representing clusters of workspaces that form a primary Grid platform. Based on the generalized idea of workspace, we are defining common methods for secure workspace deployment and management that allow us to build nested workspaces flexibly, as described in Section 5. In this sense our work is closest to COD [9], but wider in scope.

As we have shown in Section 5, concepts developed as part of the matchmaker architecture [36], SNAP [12], and WS-Agreement [3] can be applied to workspace management. However, their varied deployment times, varying enforcement capability, and heterogeneous and potentially layered nature raise as many interesting problems in resource brokering and scheduling as they solve in the enforcement and environment definition.

7. Summary and future work

The virtual workspace concept and new implementation approaches, such as virtual machines, enable many scenarios that are hard to achieve with current tools. Among others, we expect that providing reliable quality of service enforcement will lead to more widespread use of agreement-based protocols and commercial interactions in resource use and contributions. In addition, the ability to deploy remote workspaces reliably will provide much needed "quality of life" for Grid users, who will be able to use more resources with greater confidence.

Workspaces also make it easier for resource owners to contribute resources to the Grid and to manage virtual organizations. As we point out in Section 5, workspaces can be used for management at different layers. Virtual organizations can manage workspaces for their applications while still giving a site coarse-grained control over workspace images via attestation. Thus, a virtual organization can evolve workspaces flexibly and independently, while sites manage generic workspaces in site-specific ways, providing platforms for the deployment of workspaces from different communities. This strategy has the potential to lower significantly the entry barrier into the Grid by reducing the administrative cost of maintaining Grid resources.

Much research is still required to realize this vision. In future work, we plan to further refine and experiment with virtual workspaces and to work on ensuring their security, developing networking infrastructure, and devising methods that leverage the migration capability and other aspects. In addition, we plan to explore the notion of *virtual playgrounds*, or virtual Grids abstracting a real physical Grid. A virtual playground is composed of several workspaces, virtual networks, and virtualized storage or data. As a workspace is designed to support a group of jobs sharing similar requirements, a playground supports the interaction of such jobs over a Grid and may involve computations across many platforms. Our research on workspaces paves the way for this work.

References

[1] S. Adabala, V. Chadha, P. Chawla, R. Figueiredo, J. Fortes, I. Krsul, A. Matsunaga, M. Tsugawa, J. Zhang, M. Zhao, L. Zhu and X. Zhu, *From Virtualized Resources to Virtual Computing Grids: The In-VIGO System,* Future Generation Computer Systems, 2004.

[2] P. Anderson and A. Scobie, *Large Scale Linux Configuration with LCFG,* in 4th Annual Linux Showcase and Conference, 2000.

[3] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, J. Pruyne, J. Rofrano, S. Tuecke and M. Xu, Web Services Agreement Specification (WS-Agreement) Draft 20. 2004: https://forge.gridforum.org/projects/graap-wg/.

[4] A. Andrieux, K. Czajkowski, J. Lam, C. Smith and M. Xu, Standard Terms for Specifying Computational Jobs. http: //www.epcc.ed.ac.uk/%7Eali/WORK/GGF/JSDL-WG/DOCS/WS-Agreement job terms for JSDL print.pdf, 2003.

[5] R. Ashkenas, D. Ulrich, T. Jick and S. Kerr, The Boundaryless Organization, San Francisco: Jossey-Bass, 1995.

[6] J. Baldeschwieler, R. Blumofe and E. Brewer, *ATLAS: An Infrastructure for Global Computing,* in Proc. Seventh ACM SIGOPS European Workshop on System Support for Worldwide Applications, 1996.

[7] P. Barham, 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).

[8] C. Catlett, *The TeraGrid: A Primer,* 2002.

[9] J. Chase, 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.

[10] A. Chervenak, E. Deelman, I. Foster, L. Guy, W. Hoschek, A. Iamnitchi, C. Kesselman, P. Kunst, M. Ripenu, B. Schwartzkopf, H. Stockinger, K. Stockinger and B. Tierney, *Giggle: A Framework for Constructing Scalable Replica Location Services,* in SC'02: High Performance Networking and Computing, 2002.

[11] CNRI, *Handle System,* 2005: www.handle.net.

[12] K. Czajkowski, I. Foster, V. Sander, C. Kesselman and S. Tuecke, *SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems,* in 8th Workshop on Job Scheduling Strategies for Parallel Processing, 2002, Edinburgh, Scotland.

[13] N. Desai, 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.

[14] M. Feldman, K. Lai and L. Zhang, *A Price-Anticipating Resource Allocation mechanism for Distributed Shared Clusters,* ACM Conference on Electronic Commerce, 2005.

[15] R. Figueiredo, P. Dinda and J. Fortes, *A Case for Grid Computing on Virtual Machines,* in 23rd International Conference on Distributed Computing Systems, 2003.

[16] I. Foster and others, *The Grid2003 Production Grid: Principles and Practice,* in IEEE International Symposium on High Performance Distributed Computing, 2004: IEEE Computer Science Press.

[17] I. Foster, C. Kesselman and S. Tuecke, The Anatomy of the Grid: Enabling Scalable Virtual Organizations, *International Journal of Supercomputer Applications* **15**(3) (2001), 200–222.

[18] I. Foster, C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt and A. Roy, *A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocation,* in Proc. International Workshop on Quality of Service, 1999.

[19] I. Foster, C. Kesselman, J. Nick and S. Tuecke, *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration,* 2002: Open Grid Service Infrastructure WG, Global Grid Forum.

[20] I. Foster, *Globus Toolkit version 4: Software for Service-Oriented Systems*, IFIP International Conference on Network and Parallel Computing, 2005.
[21] I. Foster, 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.
[22] J. Frey, T. Tannenbaum, I. Foster, M. Livny and S. Tuecke, Condor-G: A Computation Management Agent for Multi-Institutional Grids, *Cluster Computing* **5**(3) (2002), 237–246.
[23] R. Goldberg, Survey of Virtual Machine Research, *IEEE Computer* **7**(6) (1974), 34–45.
[24] T. Hacker and B. Athey, *A Methodology for Account Management in Grid Computing Environments*, Proceedings of the 2nd International Workshop on Grid Computing, 2001.
[25] X. Jiang and D. Xu, *VIOLIN: Virtual Internetworking on OverLay INfrastructure*, Department of Computer Sciences Technical Report CSD TR 03-027, Purdue University, 2003.
[26] N.H. Kapadia, R.J. Figueiredo and J. Fortes, *Enhancing the Scalability and Usability of Computational Grids via Logical User Accounts and Virtual File Systems*, in 10th Heterogeneous Computing Workshop, San Francisco, California, 2001.
[27] K. Keahey, I. Foster, T. Freeman, X. Zhang and D. Galron, *Virtual Workspaces in the Grid*, ANL/MCS-P1231-0205, 2005.
[28] K. Keahey, I. Foster, T. Freeman, X. Zhang and D. Galron, *Virtual Workspaces in the Grid*, in Europar, Lisbon, Portugal, 2005.
[29] K. Keahey, K. Doering and I. Foster, *From Sandbox to Playground: Dynamic Virtual Environments in the Grid*, in 5th International Workshop in Grid Computing, 2004.
[30] K. Keahey, M. Ripeanu and K. Doering, *Dynamic Creation and Management of Runtime Environments in the Grid*, in Workshop on Designing and Building Web Services (to appear), Chicago, IL, 2003.
[31] I. Krsul, A. Ganguly, J. Zhang, J. Fortes and R. Figueiredo, *VMPlants: Providing and Managing Virtual Machine Execution Environments for Grid Computing*, in SC04, Pittsburgh, PA, 2004.
[32] Log4j Logging Services, 2004.
[33] A. McNab, *Grid-Based Access Control for Unix Environments, Filesystems and Web Sites*, Proceeings of the CHEP 2003 conference, 2003.
[34] Open Science Grid (OSG), 2004.
[35] Portable Batch System.
[36] R. Raman, M. Livny and M. Solomon, *Matchmaking: Distributed Resource Management for High Throughput Computing*, in 7th IEEE International Symposium on High Performance Distributed Computing, IEEE Computer Society Press, 1998.
[37] D. Reed, I. Pratt, P. Menage, S. Early and N. Stratford, *Xenoservers: Accountable Execution of Untrusted Programs*, in 7th Workshop on Hot Topics in Operating Systems, Rio Rico, AZ: IEEE Computer Society Press, 1999.
[38] Steenbakkers, Guide to LCMAPS version 0.0.23. 2003: http: //www.dutchgrid.nl/DataGrid/wp4/lcmaps/edg-lcmaps gcc3 2 2-0.0.23/lcmaps.pdf.
[39] A. Sundararaj and P. Dinda, *Towards Virtual Networks for Virtual Machine Grid Computing*, in 3rd USENIX Conference on Virtual Machine Technology, 2004.
[40] V. Talwar, S. Basu and R. Kumar, *An Environment for Enabling Interactive Grids*, in The Twelfth IEEE International Symposium on High Performance Distributed Computing (HPDC-12), Seattle, Washington, 2003.
[41] The Globus Project Web Site.
[42] Xen Scheduler Howto. 2005: http: //xen.terrabox.com/index.php/Sched-HOWTO.
[43] S. Youssef, *Pacman: A Package Manager*, 2004: http://physics.bu.edu/~youssef/pacman/.
[44] S. Youssef, *Personal communication*, 2004.
[45] X. Zhang, K. Keahey, I. Foster and T. Freeman, *Virtual Cluster Workspaces for Grid Applications*, ANL/MCS-P1246-0405, 2005.