Virtual Clusters for Grid Communities

Zhang, X., T. Freeman, K. Keahey, I. Foster, and D. Scheftner

Abstract— **One of the most challenging issues facing Grid communities today is that while Grids provide access to many heterogeneous resources, the available resources often do not match the needs of a specific application or service. In an environment where both resource availability and software requirements evolve rapidly, this leads to resource underutilization, user frustration, and much wasted effort spent on bridging the gap between applications and resources. These issues could be overcome by allowing authorized Grid clients to deploy clusters made up of virtual machines configured to suit the client's software environment and hardware allocation requirements. In this paper, we develop descriptions and methods allowing us to deploy flexibly configured virtual cluster workspaces in the Grid. We describe their configuration, implementation, and evaluate them in the context of a virtual cluster representing the environment currently in production use by the Open Science Grid. Our performance evaluation results show that virtual clusters representing current Grid production environments can be efficiently deployed and managed and provide an acceptable platform for Grid applications.**

Key Words—**Grid computing, virtualization, clusters, virtual machines**

I. Introduction

Most significant Grid deployments today, such as Open Science Grid [1, 2] or TeraGrid [3], rely on clusters to provide a powerful computation platform for their user communities. Sharing such clusters between different virtual organizations (VOs) [4] is not always easy. Problems arise when requirements of different VOs using the same resources conflict or are incompatible with site policies. Furthermore, the software available on clusters today does not provide the capabilities needed to implement isolation of different communities and guarantee resource availability while also ensuring good utilization of site resources. Both of these issues result in user dissatisfaction (as resources with suitable hardware and software configuration cannot be guaranteed or even obtained on a best-effort basis), hardware under-utilization (as available resources do not match the current demand), and significant overhead to ensure software compliance and portability.

To overcome these problems, the use of virtual machines (VMs) in Grid computing has been proposed [5, 6]. Virtual machines offer the ability to instantiate a new, independently configured guest environment on a host resource; multiple, different such guest environments can be deployed on one resource at the same time. In addition to providing convenience to the users of a resource, this arrangement can also increase resource utilization as more flexible, but strongly enforceable, sharing mechanisms can be put in place. Furthermore, the ability to serialize the state of a VM and migrate it opens new opportunities for better load balancing and improved reliability that are not possible with traditional resources. Modern VM implementations, such as Xen [7] and VMware [8] also provide outstanding isolation and enforcement properties, as well as excellent performance making the use of virtual machines cost-effective.

While the issue of combining Grid and virtualization technology has generated much interest, relatively little attention has been paid to evaluating to what extent the VM technology can be applied to the most common Grid fabric – clusters. Deploying generic virtual clusters of diverse topologies requires the ability to deploy many VMs in a coordinated fashion so that sharing of infrastructure, such as disks and networking, can be properly orchestrated. Due to the need for deployment and configuration of many VMs, cluster deployments can be more costly than the deployment of single VMs [9]. Furthermore, many Grid applications are composed of interdependent tasks requiring frequent I/O calls – typically hard for virtual machines. In this paper, we first describe how to efficiently represent and implement a generic infrastructure for virtual cluster deployment. This infrastructure extends the Workspace Service described in [10]. We then evaluate our implementation in the context of a virtual cluster modeled on Grid clusters currently in production use by one of the leading Grid communities: the Open Science Grid (OSG) [2]. We complete our investigation by evaluating the performance of an OSG application on the virtual cluster.

We first provide our work context by describing the related work. We proceed to describe how generic virtual clusters of diverse topologies can be represented to ensure deployment with the required characteristics. We follow with a description of extensions to the Workspace Service required to deploy such clusters, and the implementation that goes with it. We then describe the structure and operation of a typical OSG cluster. We use the described infrastructure to model an OSG *virtual* cluster and evaluate its deployment and management using the Workspace Service. Finally, we evaluate OSG application performance on the virtual cluster and conclude.

II. Related Work

With superior isolation properties, fine-grained resource management, and the ability to instantiate independently configured guest environments on a host resource, virtual machines provide a good platform for Grid computing [5, 6]. The Xenoserver project [11] is building a distributed infrastructure as an extension of the Xen virtual machine effort [7]. The In-Vigo project [12, 13] proposed a distributed Grid infrastructure based on the use of virtual machines as resources, while the Virtuoso [14] and Violin [15] projects explore networking issues related to using virtual machines in a Grid environment. Our approach differs in that it focuses on the workspace abstraction [10] treating virtual machines as one potential implementation. Thus, we focus on developing workspace descriptions and interfaces, that could fit a variety of approaches, and that clearly define the required resource allocation abstractions and enable their secure provisioning.

In addition, driven by community requirements, we also focus on clusters, rather than single resources, as a primary Grid platform. The idea of providing customized clusters "on demand" (albeit not based on virtual machines) based on some form of Grid credential has been pioneered by the Cluster on Demand (COD) project [16]. This effort is complementary to our work: workspaces provisioned by COD could be used to deploy virtual clusters as described here.

III. Virtual Clusters

In [10] we describe virtual workspaces (VWs) which provide an abstraction of an execution environment that can be made dynamically available to authorized clients by using well-defined protocols. We briefly summarize this abstraction here.

Workspaces are described by *workspace meta-data*, which contains all the information needed to deploy a workspace in various contexts. An atomic workspace, representing a single execution environment, specifies what data (such as VM images) needs to be obtained, and what deployment information (such as networking setup) needs to be configured on deployment. At deployment time, a workspace is associated with a *resource allocation*, which describes how much resource (CPU, memory, etc.) is assigned to the workspace. We implemented such workspace deployment in the *Workspace Service*, a Globus Toolkit 4 (GT4) [22] based service used to deploy workspaces. The Workspace Service implements a WSRF-based protocol allowing remote clients to start, stop, manage and inspect workspaces. All client actions are authorized using the Grid Security Infrastructure (GSI).

In this section, we describe in detail extensions made to the workspace meta-data schema to support a virtual cluster and explain how the schema was used to support its deployment. We also describe changes to Workspace Service interfaces and implementation. For a comprehensive description of the meta-data schema as well as interfaces refer to [10].

A. Virtual Cluster Representation

Atomic workspaces, first described in [9], can be combined to form virtual clusters. The key extension proposed in this work is therefore an *aggregate workspace* which contains one or more *workspace sets* – sets 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, described in Section IV, is composed of two workspace sets: a set containing one service node (with service node configuration) and a set of worker nodes (all with the same worker node configuration). This cluster specification can be easily redefined to for example include multiple service nodes of the same or different configurations or different sets of worker nodes. All information about a cluster workspace is composed from the metadata of the atomic workspaces describing those nodes.

The networking of the cluster is described to reflect the potentially different configurations of its atomic components: each atomic workspace can have a number of differently configured network connections [10]. Continuing with our example of an OSG cluster, each worker node has one NIC element configured to obtain an IP on a private network – this could be done by DHCP, pre-arranged, obtained from an external service, or (as in our experiments) allocated by the Workspace Service. The service node is described by two NIC elements, one of them sharing the subnet of the worker nodes, the other using the pre-arranged option, giving the service node a static, public IP address.

Further configuration options, such as information about shared disk partitions are also recorded in the definition sections of the workspace meta-data for all atomic descriptions. It is extracted by the Workspace Service and passed as a kernel option when the corresponding node is propagated. When a node is booted, an OS boot script is executed to customize the NFS sharing configuration and update the necessary configuration files.

B. Workspace Service Interfaces for Virtual Cluster

Performing operations on the virtual cluster requires allocating resources to a group of VMs as well as deploying and managing them as a group. The Workspace Service operations have thus been extended to handle aggregate workspaces. 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. The resource properties reflect the aggregate structure of the workspace and allow the client to operate on it.

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. In addition, the current implementation assumes that the resource allocation will match the workspace exactly; in the future, we plan to extend this functionality to work with bulk allocations (such as represented by WS-Agreement [17]).

C. *Workspace Service Implementation*

In our implementation we assume that the Workspace Service executes on a service node of a physical cluster and provides a secure gateway to a set of resources that can support the deployment of virtual machines. We further assume that all data necessary for deployment (such as VM images) has already been staged to a node in the trusted computing base (TCB). The service node of the physical cluster runs a GT4 container and the Workspace Service. The hosts are configured with the Xen hypervisor, Workspace Service back-end scripts and a means to invoke them such as SLURM [18], as well as some means of transferring image files and other data relevant to the workspace from within the TCB.

The current implementation accepts workspace creation requests based on resource availability. The Workspace Service maintains a database of information about physical hosts available for workspace deployment. For each physical host it records availability, CPU type, total/available memory size, total/available disk size, and system information. When the Workspace Service receives the cluster workspace creation request, it searches the database for a set of resources matching the resource allocation request, defines a matching set, marks it as reserved, and maps the resource allocation onto it. When the workspace is terminated, the resources are reclaimed and the database is modified accordingly. The implementation discussed in this paper does not yet support advance reservations although the database can support the time dimension. In addition to allocating resources, the Workspace Service also handles local IP address allocation.

Workspace creation results in the creation of a workspace WSRF resource as well as workspace deployment up to a specified state [10]. A workspace is deployed through invocation of workspace back-end scripts via local schedulers (the current implementation works with SLURM [18] and PBS [19]). 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. This step is separate to create an opportunity for pre-staging of the images without actually starting the VMs. 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 boot 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.

IV. OSG Clusters and Infrastructure

The Open Science Grid (OSG) [2, 20] is a national production-quality Grid for large-scale science, enabling scientists to access shared resources using common Grid infrastructure tools. The OSG software stack is based on the NSF Middleware Initiative distribution, which includes Condor [21] and Globus [22] technologies, as well as additional utilities provided by the Virtual Data System (VDS) [23].

The OSG Grid infrastructure is structured so that users with work to do contact a *submit host* which organizes the work as a set of inter-dependent tasks and orchestrates their execution on OSG resources. To do this, the submit host uses tools provided by VDS to express task workflow, planners such as Pegasus [24] to translate it into an executable form, and Condor DAGMan [25] as well as the Condor-G scheduler [21] to schedule them on OSG resources.

OSG resource are typically organized as clusters, consisting of one or more *service nodes*, providing secure access to a group of *worker nodes* where application jobs are executed. The worker nodes are typically configured to be accessible on a site's private network only, while service nodes can be accessed both on private as well as on public network. Services executing on the service nodes thus execute on the edge of the public and private network and are often called *Edge Services*. They can implement various management functions – for example, *compute* elements (CEs) provide access to the cluster's compute resources, while *storage elements* (SEs) provide access to its storage resources. In this paper, we will focus on providing access to a cluster's computational resources.

To provide compute power for their communities, a typical OSG cluster today manages worker nodes through at least one service node and supports the following configuration:

- A local batch scheduler, such as Condor-C [26], PBS [19], LSF [27] or SGE [28], is installed on the cluster and used to manage it.
- The worker nodes and the service nodes have access to a shared file system, such as NFS; this ensures that application executables, data and other files placed on the service nodes are available to the worker nodes.
- Grid infrastructure, in current deployments typically the Globus GRAM [29] and GridFTP [30], is running on the service node to authenticate and authorize job requests, stage required files, and submit jobs to the local batch scheduler.

To use OSG resources, a user first authenticates to an OSG submit host. If the submission is authorized, executables and other data are staged to a selected OSG cluster followed by a

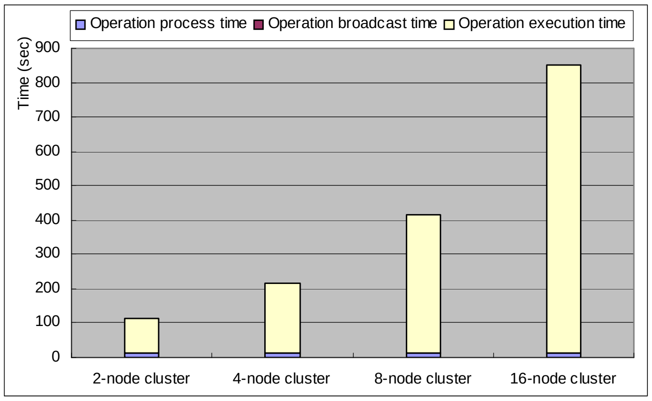

Figure 1: Propagation operation timing results on four clusters.

Condor-G submission of the requested tasks to the cluster. The submission is orchestrated and managed through the cluster's service nodes.

V. The Experimental Results

In this section, we describe two kinds of experiments we made to evaluate the performance of OSG virtual cluster: deploying and managing the cluster itself, and estimating the efficiency of the cluster for OSG applications of different profiles.

We ran our experiments on a testbed constituting of a partition of the Chiba City cluster at Argonne National Laboratory (ANL) [31]. Each Chiba node is equipped with two 500 MHz Intel PIII CPUs (with a 512 KB cache per CPU), 512 MB main memory, and 9GB of local disk. The nodes are connected by a 100 Mbps LAN.

To host the virtual cluster, Chiba City nodes were configured with Xen 2.0 testing distribution (both domain 0 and user domain run port of Linux 2.6.11) and rebooted with XenLinux. We chose domain 0 memory size to allow the best possible performance: neither too large (so that as much memory as possible can be given to the VM) nor too small (so that it does not interfere with the VM performance by skimping on buffering space). We found that for I/O intensive application the optimal memory size is 96 MB and used this size in the experiments described here. In the experiments described here, SLURM was used to schedule VMs on Chiba City nodes.

The configuration of the OSG virtual cluster replicates in detail the configuration of the OSG cluster described in section IV (our experiences show that even seemingly irrelevant configuration details may have significant impact on performance). Since the images represent inactive VMs, the primary constituent of the image is VM's disks. The head node's file system is made up of three disk partitions mounted from three disk images, including a 600MB root file system image containing a Debian Sarge Linux installation, a 750MB image required for the Virtual Data Toolkit (VDT) v1.3.6 OSG middleware stack, and a 1GB image containing OSG application executables and data. The latter two partitions are exported as an NFS volume so that they can be shared by worker nodes. The worker node's root file system is built from a 600MB root file system image containing Debian Sarge Linux installation, similar to the head node. Worker nodes are configured to share the OSG middleware partition exported by the head node when it is booted.

A. Evaluation of Virtual Cluster Deployment and Management

We first evaluated the time it takes to deploy and manage the virtual cluster. In our timings we assume that all the data to deploy the cluster has already been staged to a known place within the TCB. We split VM deployment into two major components: (1) virtual cluster scheduling and image propagation phase, in which physical nodes are selected and VM images are copied to those nodes, and (2) customizing and deploying the virtual machines. The first phase takes by far the most time and can be executed ahead of actual VM deployment.

Figure 1 shows timing results for the invocation of the *create* operation with the "propagate only" option [10] measured on the server from the moment of receiving the request to the request's completion. The graph shows the timing of the request as a sum of three components: (1) operation processing time (which includes finding physical resources available for deployment), (2) operation broadcast time, and (3) image propagation time where the Workspace Service backend is used to pull images to hosts of worker nodes. Note that while the broadcast time is negligible, the operation processing time is relatively high (on the order of seconds versus ~100ms for operations shown in Figure 2) as it includes all of the deployment overhead (i.e., mapping virtual machines to resources).

The deployment time is dominated by image propagation. Although the workspace service backend can be configured to support a variety of transport mechanisms to be used for image propagation, the data we show here uses an NFScopy technique (using the Linux *cp* command to copy files from a locally mounted NFS directory to a directory on a local disk) on the shared file system (NFS v3) available on the physical cluster (using GridFTP in the same scenario yielded similar results). As expected, the propagation time increases with the size of physical resource allocation that the cluster is assigned to. This is because the 100Mbps network link is easily saturated by the 600 MB image transfer and each connection adds a fixed amount of time to the transfer. Using Chiba City's default NFS server, equipped with a Gigabit Ethernet connection to the switch, the same operation resulted in a significant speedup and flat transfer times for up to 10 transfers when the incoming connections to target nodes get saturated.

Figure 2 show the timing results of start and stop operations in different contexts. Virtual cluster operation requests are broadcast using cluster job submission tools. All of the request information is broadcast to all the nodes; each node then selects information relevant to it from the broadcast data. This implementation results in flat broadcast time as shown in the graph. The time spent in each operation is dominated by the time it takes to perform requested actions and varies little with the cluster size as the operations on each node are independent from each other. The start/running operation is slightly slower than start/unpause since in the latter case the image is already in memory. Note that operation processing time for all the start and shutdown operations (100ms) is significantly less than that for the propagation operation (a few seconds). This is because the propagation operation timing includes the *create* overhead which includes matching to resources as well as resource setup.

B. *OSG Applications in Virtual Clusters*

To assess the practicality and trade-offs involved in using a virtual cluster, we ran experiments evaluating the impact of running on a virtual cluster for various OSG applications.

In our tests we compared the execution time of an application instance on cluster made up of physical Chiba nodes to the execution time of the same application instance on a virtual cluster running on physical Chiba nodes. Unlike [7] we did not replicate the exact hardware conditions for virtual and physical nodes. Thus, the virtual machines deployed on the cluster had access to fewer resources (memory and CPU) as some resources had to be assigned to hypervisor functions (domain 0). We believe that these tests give a better effective measure of how well an application running in a VM can perform in practice on a given resource (figuring in the overhead of running a virtual machine).

For our evaluation we chose the FOAM [32] application. FOAM is a climate science application that uses coupled climate modeling techniques to address climate questions that require many simulated years of interaction. FOAM is an MPI-based data-parallel application organized into three components: atmosphere, land/sea-ice, and ocean. The first two are collocated data-parallel programs that use MPI for component computation but communicate between themselves via shared memory. The third one executes on a distinct set of processors and uses MPI to communicate with the other two. Altogether, the communication makes up about 10% of run

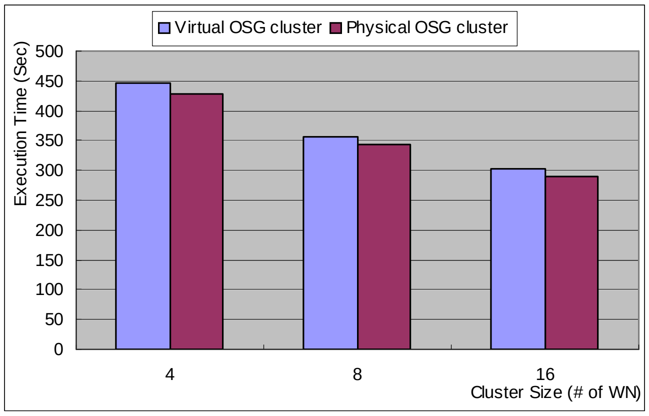

Figure 3: Comparison of FOAM execution time on virtual and physical clusters.

TABLE 1
FOAM PERFORMANCE LOSS ON VIRTUAL CLUSTER RELATIVE TO PHYSICAL CLUSTER.

4 Nodes	8 Nodes	16 Nodes
4.55%	4.05%	4.10%

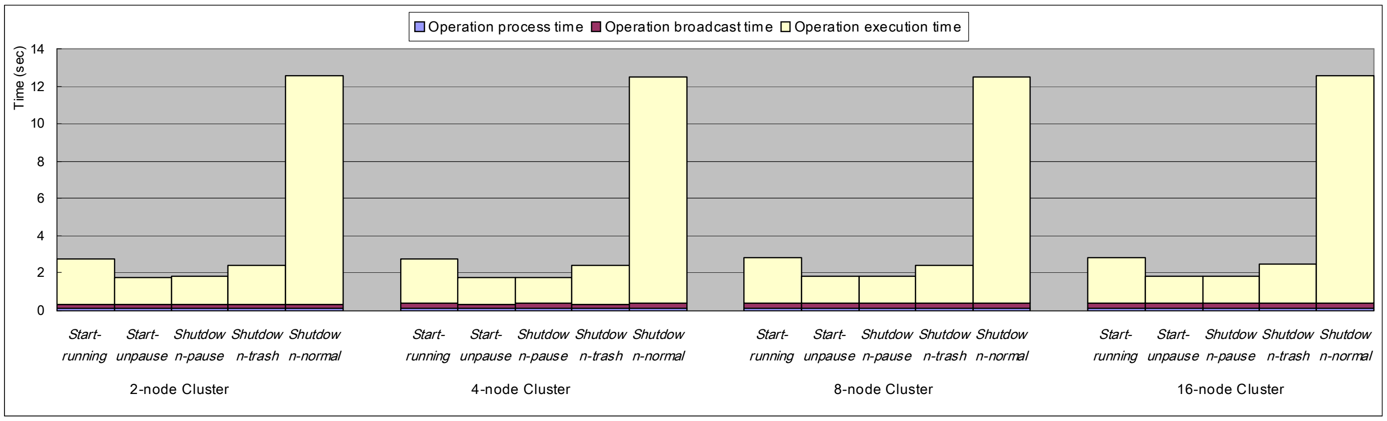

Figure 2: Start and shutdown operations timing results on four clusters.

time. Other IO requests issued by the application consist of using GridFTP to transfer the input data files from a storage node (in our experiment, located on the same LAN as the execution cluster) at the beginning of the computation. The size of the data files is less than 22MB.

Figure 3 presents the execution time of the FOAM 1.5 workflow running on both physical and virtual clusters. The time measured is from the submission of the *mpirun* command to its completion. Although our benchmark study of MPI behavior under Xen [33] indicates that certain patterns of MPI communication over Ethernet may be potentially expensive in Xen, we observe the workflow has close execution time performance on both clusters. Table 1 shows the performance difference between them to be overall less than 5%. This is due to the fact that the overall proportion of time spent on communication within the application is small. In practice, applications that are heavily communication-bound tend to scale poorly and are not a good match for clusters thus making it likely that applications of this type will respond favorably to virtual clusters.

C. Analysis

While much faster than node re-imaging, virtual cluster deployment can be costly – on the order of minutes – due to the necessity of copying images to target nodes. Where NFS is present in principle nodes could be loaded directly from the sources – however our practical experience indicates that the NFS infrastructure is not always capable of supporting the network traffic that results as some images need to be frequently written to. A more practical optimization is to ensure that images are staged to advantageous locations (e.g., NFS server with high speed location). We are also investigating schemes which would optimize propagation by pre-staging the most commonly used images, or their parts, to the worker nodes or containing less used read-only OS libraries to separate partitions that could be mounted via NFS with acceptable performance.

The potentially relatively high cost of image propagation makes it imperative that a client, or a scheduler, be given the opportunity to execute this operation before the virtual machine is started. We incorporated this observation into our interface design by allowing for the workspace to progress through the deployment states only up to a specific state – this gives a client the opportunity to halt the workspace deployment at well-defined points, image propagation being one of them. Even given this flexibility, when using medium to large sized images virtual cluster deployment is potentially expensive; it is therefore most suitable for scenarios that can offset this relatively high deployment cost, such as either hosting several applications or hosting a long-running application.

OSG application results are promising. For our evaluation we chose FOAM because its data-parallel nature makes it potentially hard for virtual machines. The 5% performance degradation shows that in practice where communication constitutes a small part of an application the performance issues observed in [34] are not significant. In fact, evaluating other OSG applications we observed that some tasks appear to be in fact executed faster on a virtual than on a physical machine which is likely due to double caching of I/O operations. We are currently investigating these effects and their trade-offs in realistic settings.

VI. Conclusion

We have described virtual clusters, groups of virtual machines designed to execute, and share infrastructure, within a trusted computing base. We showed how to define cluster descriptions so that atomic workspaces can be flexibly composed into more complex constructs while organizing infrastructure sharing between the virtual nodes. Cluster deployment allows us to specify different resource allocations for different members of aggregates defined in this way. Further, we described how such clusters can be deployed, evaluated their deployment, and integrated its results into our design.

Our application execution results are promising – the slowdown suffered by the FOAM application from virtual machine impact on execution as well as from the resource overhead of using virtual machines was less than expected -- within 5%. Considering that virtual machines offer unprecedented flexibility in terms of matching clients to available resources, this performance impact seems to be an acceptable trade-off. Preliminary results from investigating other OSG applications of more complex dependency patterns are equally promising and lead us to believe that virtual clusters have the potential to be a popular solution in production settings.

Acknowledgment

We acknowledge the help of Jens Voelker, Mike Wilde, and Robert Jacob with explanation of the configuration of an OSG cluster as well as application setup. We also thank Rick Bradshaw for assistance with the Chiba City cluster testbed at Argonne. This work was supported in part by the Mathematical, Information, and Computational Sciences Division.

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. *TeraGrid*: www.teragrid.org.
4. 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.
5. Figueiredo, R., P. Dinda, and J. Fortes. *A Case for Grid Computing on Virtual Machines*. in *23rd International Conference on Distributed Computing Systems*. 2003.

6. Keahey, K., K. Doering, and I. Foster. *From Sandbox to Playground: Dynamic Virtual Environments in the Grid*. in *5th International Workshop in Grid Computing*. 2004.
7. 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)*.
8. *VMware*: www.vmware.com.
9. Keahey, K., I. Foster, T. Freeman, X. Zhang, and D. Galron. *Virtual Workspaces in the Grid*. in *Europar*. 2005. Lisbon, Portugal.
10. Keahey, K., I. Foster, T. Freeman, and X. Zhang, *Virtual Workspaces: Achieving Quality of Service and Quality of Life in the Grid.* accepted for publication in the Scientific Progamming Journal, 2005.
11. Reed, D., I. Pratt, P. Menage, S. Early, and N. Stratford. *Xenoservers: Accountable Execution of Untrusted Programs*. in *7th Workshop on Hot Topics in Operating Systems*. 1999. Rio Rico, AZ: IEEE Computer Society Press.
12. Adabala, S., 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.
13. Krsul, I., A. Ganguly, J. Zhang, J. Fortes, and R. Figueiredo. *VMPlants: Providing and Managing Virtual Machine Execution Environments for Grid Computing*. in *SC04*. 2004. Pittsburgh, PA.
14. Sundararaj, A. and P. Dinda. *Towards Virtual Networks for Virtual Machine Grid Computing*. in *3rd USENIX Conference on Virtual Machine Technology*. 2004.
15. Ruth, P., X. Jiang, D. Xu, and S. Goasguen, *Towards Virtual Distributed Environments in a Shared Infrastructure.* IEEE Computer, Special Issue on Virtualization Technologies, 2005.
16. 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.
17. Andrieux, A., 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/.
18. 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.
19. *Portable Batch System*. 2003: www.openpbs.org.
20. Pordes, R., *The Open Science Grid.* Proceedings of the CHEP'04 Conference, 2004.
21. Frey, J., T. Tannenbaum, I. Foster, M. Livny, and S. Tuecke. *Condor-G: A Computation Management Agent for Multi-Institutional Grids*. in *10th IEEE International Symposium on High Performance Distributed Computing*. 2001: IEEE Computer Society Press.
22. Foster, I., *Globus Toolkit version 4: Software for Service-Oriented Systems.* IFIP International Conference on Network and Parallel Computing, 2005.
23. Foster, I., J. Voeckler, M. Wilde, and Y. Zhao. *Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation*. in *14th Intl. Conf. on Scientific and Statistical Database Management*. 2002. Edinburgh, Scotland.
24. Deelman, E., J. Blythe, Y. Gil, C. Kesselman, G. Metha, S. Patil, M.-H. Su, K. Vahi, and M. Livny, *Pegasus: Mapping Scientific Workflows onto the Grid.* Proceedings of the 2nd Europena Across Grids Conference, 2004.
25. Frey, J., *Condor DAGMan: Handling Inter-Job Dependencies*: www.cs.wisc.edu/condor/dagman.
26. Litzkow, M.J., M. Livny, and M.W. Mutka, *Condor - A Hunter of Idle Workstations*, in *8th International Conference on Distributed Computing Systems*. 1988. p. 104-111.
27. *LSF Web Site*: www.platform.com/products/wm/LSF/index.asp.
28. Gentzsch, W., *Sun Grid Engine: Towards Creating a Compute Power Grid.* Proceedings of 1st International Symposion on Cluster Computing and the Grid, 2001.
29. Czajkowski, K., I. Foster, and C. Kesselman, *Resource and Service Management*, in *The Grid: Blueprint for a New Computing Infrastructure (2nd Edition)*. 2004.
30. Allcock, W., *GridFTP: Protocol Extensions to FTP for the Grid*. 2003, Global Grid Forum.
31. *Chiba City Homepage*: www.mcs.anl.gov/chiba.
32. Jacob, R., C. Schafer, I. Foster, M. Tobis, and J. Anderson. *Computational Design and Performance of the Fast Ocean Atmosphere Model, Version One*. in *International Conference on Computational Science*. 2001: Springer-Verlag.
33. Zhang, X., *The Effect of DomO Memory Size on the Performance of DomU Applications*: people.cs.uchicago.edu/~hai/vcluster/find-dom0-sweetpoint.pdf.
34. Zhang, X., *Evaluation of a Virtual Xen Cluster Using the Pallas MPI Benchmarks Suite*: people.cs.uchicago.edu/~hai/vcluster/PMB/.