Virtual Workspaces for Scientific Applications

Kate Keahey, Tim Freeman: Argonne National Laboratory

Jerome Lauret: Brookhaven National Laboratory

Doug Olson: Lawrence Berkeley National Laboratory

Corresponding Author: keahey@mcs.anl.gov

Abstract. One of the primary obstacles users face in Grid computing is that Grids are typically composed of many diverse resources, while applications require a very specific, customized environment to run in. Many applications are dependency-rich and complex, making it hard to run them on anything but a dedicated platform. Worse, even if the applications do run there, the results they produce may not be consistent across different runs.

As part of the Center for Enabling Distributed Petascale Science (CEDPS) project we have been developing the Workspace service which allows authorized Grid clients to dynamically provision environments in the Grid. Virtual machines provide an excellent implementation of a portable environment as they allow users to configure an environment and then deploy it on a variety of platforms. This paper describes a proof-of-concept of this strategy developed for the the High-Energy Physics STAR application. We are currently building on this work to enable production STAR runs in virtual machines.

1. Introduction

A naïve view holds that most applications can be rendered as relatively simple codes that can be easily ported to a large set of platforms, can be then further staged to an arbitrary platform from that set based on availability, where they will reliably execute and produce consistent results. In practice however, significant application codes are complex: they are likely to be written by a diverse team of people, over multiple years of changing software fashions and contain many dependencies not all of which are well understood or easy to describe. Such applications are hard to port on anything but a dedicated platform as variations in operating system, middleware versions, and library environments all pose barriers to their portability. Thus, applications that work on a developer's desktop may only function "out of the box" on a small fraction of the total number of compute resources potentially available to the scientist. To make matters worse, while some applications can in principle work in a variety of environments, in practice the results they produce may be affected by how those environments are configured.

The applications of the high-energy physics STAR experiment [1] are an example of such demanding applications. Comprising over one million lines of C++ and Fortran code, the STAR applications rely heavily on the right combination of compiler versions and available libraries to work. Even when the application compiles, validating a new platform is a very complex task.

Although such applications tend to have large computational resource demands, unsurprisingly, they find it hard to acquire resources in the Grid: a matching environment is hard to find. The existing information on supported environment features is either non-existent or too coarse-grained to provide

a sufficient representation of the environment. The problem is exacerbated by the fact that policies on many resources prohibit users from simply logging in and verifying the environment features.

In this paper, we describe our work on solving this problem by developing methods allowing scientists to use virtual machines to deploy customized environments on remote resources. We report on the experiences of an experiment consisting of running the STAR application inside VM images, describe the insights we gained, as well as our plans for future work.

2. Resource Provisioning for Scientific Applications

To address the issues described above we developed a service that allows an authorized client to dynamically map environments – which we call workspaces -- onto a set of resources [2, 3]. Thus, the workspace abstraction captures both the software configuration (e.g., operating system installation, provided services) as well as the resource quota assigned to the execution environment (e.g., CPU, network share, or memory). While workspaces may be implemented in many different ways [4] – e.g., as physical machines configured using an automated configuration management system such as Bcfg2 [5] or the Pacman configuration software [6] – the current workspace service implementation [3] relies on the use Xen virtual machines (VMs) [7].

A VM provides a virtualized abstraction of a physical machine. Software running on a host supporting VM deployment, 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 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, disk images, and configuration files. Thus, a VM can be paused, its state serialized, and later resumed at a different time and in a different location, thus decoupling image preparation from its deployment and enabling migration. 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.

The Workspace service provides interfaces, based on the Web Service Resource Framework (WSRF) protocol, that allow an authorized Grid client to deploy, shutdown, pause, and reactivate VMs. The authorization is based on attributes (e.g., contained in the VOMS [8] credential) as well as site policies as to what request can be accepted. A workspace is deployed based on two types of information: (1) a pointer to an image and meta-data describing deployment-specific information and (2) a resource allocation (memory, CPU share, etc.) to be assigned to the VM. Once deployed, a client can discover relevant information about the image (e.g., the assigned IP address), manage the image (e.g. increase the time-to-live of a VM), adjust its resource allocation (e.g., the memory allocated to the VM), or terminate the image.

The workspace meta-data provided on deployment allows a client to request network configuration for the VM accommodating several flexible options (allocating new network address, bridging existing address, requesting multiple NICs, etc.). The resource allocation currently specifies memory allocation and CPU share and can be managed during deployment. To implement workspace deployment and management we provide a resource manager that can manage a pool of nodes on which workspaces are deployed as well as a back-end for the popular Amazon's EC2 service []. We are also currently working on providing extensions to existing resource managers and schedulers to enable non-invasive deployment of workspaces.

3. Running STAR in a Virtualized Setting

To verify that the concept of flexibly provisioning resources using VMs, we produced a proof-of-concept using the workspace service to dynamically deploy images configured to support the STAR application. The images were dynamically deployed and managed on the TeraPort cluster at the University of Chicago. This work was demonstrated at SC06 and generated insights that we are working to respond to, discussed in section 4.

The platform used for the proof-of-concept was provided by the TeraPort cluster [9] at the University of Chicago. The workspace service version TP1.2 was managing 4 nodes of the cluster, each equipped with 4GB memory and two 2GHz Opterons (the workspace service itself was deployed on another node). The nodes were configured with Xen 3.0. To support the experiment, a VM representing an OSG compute element (CE) was developed and statically deployed on one of the nodes (the same node as the workspace service). The virtual CE was configured with GRAM2 (to allow remote users to submit jobs) and Condor (to administer the virtual worker nodes).

The worker node images were developed using the rBuilder configuration manager [10] and consisted of a base configuration (operating configuration, basic OSG software stack: about 1GB), STAR application (about 3GB), and a data partition (about 300MB). Altogether, the worker node image was 4-5GB. In addition, the nodes required about 3GB writable disk space. Each worker node image was pre-configured with the IP address of the statically deployed CE node so that the worker nodes could register with the Condor headnode on deployment.

In the proof-of-concept scenario (shown in Figure 1), the worker node deployment was requested on-demand by an authorized off-site Grid client (which in a full scenario could represent a provisioning broker). The typical resource allocation request would ask for roughly 2GB memory and the full use of a CPU for each virtual node allowing us to deploy up to 8 STAR virtual worker nodes at a time. On deployment, as part of the boot sequence each node would report to the Condor headnode and thus join the Condor pool. A web application would display current virtual cluster node information based on condor pool properties. A client (the same or different) could then start jobs on the deployed VM using GRAM2 deployed on the static CE.

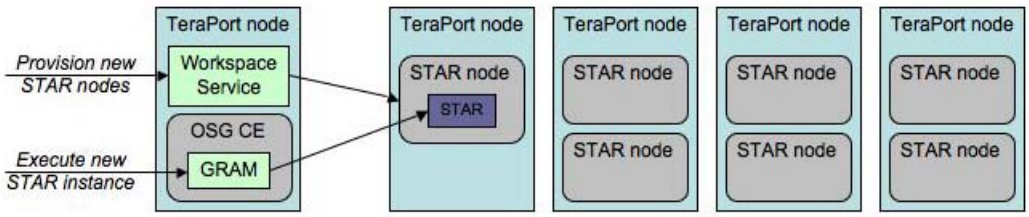

Figure 1: On-demand provisioning of STAR nodes

The deployment worked well and provided a proof-of-concept showing that by using virtualization a complex application can be deployed on-demand on a site not previously configured to support it. However, the experiment also identified new requirements for middleware supporting VM deployment. In particular, the size of the STAR image proved to be a significant factor: moving the image between the workspace image node (co-located with the workspace node) and the worker nodes over TeraPort's network (constrained by disk I/O) took as long as 8 minutes. Those issues can be overcome by managing images more effectively: i.e. caching frequently used images or their parts on the worker nodes (as we ended up doing in our experiment) and generating and mounting "blank" partitions to the image. Additional concerns were raised by resource providers as well as image users and are summarized in the next section.

4. Lessons Learned

The STAR proof-of-concept experiment identified several areas where progress is necessary to achieve scalable and truly dynamic deployment of virtual platforms complex enough to be of practical use to large scientific communities. Below, we summarize the most direct conclusions generated by the experiment:

- *Image management on deployment.* As the STAR example illustrates, in practice VM images can be large, leading to long deployment times. In order to reduce this time we first note that VM disks, like disks of physical machines, are composed of partitions. If those partitions are empty they can be simply generated and mounted at destination. When the partitions are read-only, they could potentially be cached and used by multiple VMs [11]. These simple strategies have the

potential to save disk space, network bandwidth, and deployment time. Implementing them however, requires viewing a VM as a set of partitions rather than an opaque image. The implementation required to support it is non-trivial: the partitions need to be well described, properly mounted, require a sound security model (i.e., we need to determine whose VMs can access whose partitions), we need to structure indexing and reuse (and trashing), and integrate their transfer into a caching scheme.

- *Virtual clusters*. While in the proof of concept we were deploying individual worker nodes, to support a production scenario we need to provision an entire virtual cluster, complete with a CE and potentially other services, and then manage the resources of this cluster. To do this requires the ability to describe and manage collections of virtual nodes and their dependencies. A description of preliminary strategies to achieve that can be found in [12]; we are currently working on integrating them into the Workspace service.
- *Contextualization*. On deployment a VM needs to be made aware of its deployment context: its IP address may need to be assigned, or it may have to be pointed at site services. This is particularly important when VMs make up more complex constructs such as a virtual cluster: the cluster nodes may need to be configured to network with each other, share storage and other resources, or recognize a headnode (or all of these things). In the proof-of-concept, the worker nodes were statically configured with an IP address of a pre-deployed headnode – to do this dynamically, we need to provide ways for the node to consume this information dynamically at deployment time. While ad hoc tools can be easily developed to deal with a special case, to address this problem in a scalable way contextualization tools synchronized with configuration management tools need to be developed [13].

The broader conclusions from the experiment and the work leading up to it point to a general need for more scalable methods of VM image generation and management. Methods allowing users to easily and reliably procure VM images from trusted sources are urgently needed. Without them, the users are prevented from taking advantage of virtualization by the need to configure an image, a significant barrier for many users (if not because of the initial configuration, because of the ongoing maintenance issues). Furthermore, the resource providers are understandably reluctant to deploy images unless their provenance from a trusted source can be demonstrated. To enable this functionality, methods of image signing and validation [14] need to be developed, as well as other tools allowing providers to verify that the deployed images conform to site policies as well as reliably manage their deployment. In general, based on the discussions surrounding the proof-of-concept work, we see a need for the emergence of VO-based administrators that could produce images trusted by resource providers and the formulation of policies required for image acceptance, contextualization and management.

5. Ongoing Work

Our proof-of-concept shows that virtualization allows application users to run on platforms that were not specifically prepared for the application by deploying VM images on those platforms. This has the potential to allow users to run their applications on more platforms. However, by the same argument, virtualization also has the potential to allow more resource providers to reach more users by making their platforms suitable for more applications. This principle has been embraced by the industry and resulted in the creation of a number of services selling compute cycles via VMs. The Amazon Elastic Compute Cloud (EC2) is the best known example of such services [15].

Since virtualization adoption in the scientific community is proceeding at a slow pace and only limited virtualized resources are available (our proof-of-concept ran on only 5 nodes), in order to provide a production execution platform for users motivated to use virtualization, we decided to build a workspace gateway to the EC2 service [3]. This will allow us to buy virtualized cycles for communities motivated to run in VMs by using attribute-based authorization to map users to EC2

accounts. Further, it will allow those user communities later to use the same services and images to take advantage of other virtualized platforms.

Using EC2 allows us to provision resources adequate for a STAR production run. In addition, middleware developed in answer to the proof-of-concept experiences described in the previous section will allow us to deploy and manage virtual clusters, making more scalable deployment possible. We are thus positioned to try another experiment, this time running the STAR applications in a production setting. We are currently developing images and finalizing middleware development to support such a deployment. This new thrust will test the not only the ability to dynamically provision new platforms via virtualization, but also provide insights into the viability of outsourcing the provisioning of computation cycles.

Acknowledgments

We acknowledge the role of the rPath company in developing the STAR image used in these experiments. This work was supported by the SciDAC Program and the Mathematical, Information, and Computational Sciences Division Subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contracts DE-AC02-06CH11357 and DE-AC03-76SF00098.

References

1. *The STAR Experiment*. 2007: www.star.bnl.gov.
2. Keahey, K., I. Foster, T. Freeman, X. Zhang, and D. Galron. *Virtual Workspaces in the Grid*. in *Europar*. 2005. Lisbon, Portugal.
3. *Virtual Workspaces*: http://workspace.globus.org.
4. Keahey, K., I. Foster, T. Freeman, and X. Zhang, *Virtual Workspaces: Achieving Quality of Service and Quality of Life in the Grid*. Scientific Programming Journal, 2005.
5. 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.
6. Youssef, S., *Pacman: A Package Manager*. 2004: http://physics.bu.edu/~youssef/pacman/.
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. *The Virtual Organization Management System*: http://infnforge.cnaf.infn.it/projects/voms.
9. *The TeraPort Cluster*: http://www.ci.uchicago.edu/research/detail_teraport.php.
10. *rPath*: www.rPath.com.
11. Sotomayor, B., K. Keahey, and I. Foster. `Overhead Matters: A Model for Virtual Resource Management`. in *The 1st IEEE/ACM International Workshop on Virtualization Technology in Distributed Computing (VTDC 2006)*. 2006.
12. Freeman, T., K. Keahey, B. Sotomayor, X. Zhang, I. Foster, and D. Scheftner, *Virtual Clusters for Grid Communities.* CCGrid, 2006.
13. Bradshaw, R., N. Desai, T. Freeman, and K. Keahey. *A Scalable Approach to Deploying and Managing Virtual Appliances*. in *TeraGrid 2007 Conference*. 2007. Madison, WI.
14. Lu, W., T. Freeman, K. Keahey, and F. Siebenlist, *Making your workspace secure: establishing trust with VMs in the Grid.* SC05 Poster Presentation, 2005.
15. *Amazon Elastic Compute Cloud (Amazon EC2)*: http://www.amazon.com/gp/browse.html?node=201590011.