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. Scientists often face the need for more computing power than is available locally, but are constrained by the fact that even if the required resources were available remotely, their complex software stack would not be easy to port to those resources. 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 on another system, 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 clients to dynamically provision execution environments, using virtual machine technology, on remote computers. 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 High-Energy and Nuclear Physics (HENP) applications such as STAR's. We are currently building on this work to enable production STAR runs in virtual machines.

1. Introduction

Scientists often require compute power beyond the locally available resources to accommodate more users, more data, and more jobs. However, in order to use a remote platform, they first need to port their applications to run on this platform. While this may be relatively easy to do for a HENP simulation or Monte Carlo event generation process, it is far from the easy when it comes to the production of Physic's usable quantities based on the mining of detector's signal data. These tasks require significant and complex application codes that are likely to have been developed by a diverse team of scientists over multiple years of changing software fashions. The architecture of those codes relies on dynamically loading external libraries depending on the task to be performed – while flexible and efficient from the end user's perspective, this approach creates many dependencies that are not easily predictable within the workflow of application execution. Thus, configuring an environment for such an application is complex, and their deployment on non-dedicated platform effort consuming.

The applications of the nuclear physics STAR experiment [1] are an example of such demanding applications. Comprising over one million lines of C++ and half a million of Fortran code, developed over ten years by more than 100 scientists, and involving hundreds of dependencies, the STAR applications rely heavily on the right combination of compiler versions and available libraries to work. Even when the application compiles on a new platform, validating it is a controlled process subject to

quality assurance and regression testing to ensure Physics reproducibility and result uniformity across environments.

Considering its heavy reliance on dependencies deeply embedded in the environment, porting an application would be easiest if we could take the full software stack from operating system up, and simply install that environment on remote resources. The virtual machine technology allows us to do just that. A virtual machine (VM) provides a software-based virtualization of a physical host machine; it can be configured with a full software stack and, once configured, deployed on remote resources in a matter of milliseconds. This makes resource provisioning via VMs very attractive: a scientist can develop his or her application within a familiar environment, and then port this environment between local and remote resources as the need arises. This facilitates provisioning resources for an application: the virtual machine can be run as easily on local resources as on remote community resources or resources outsourced commercially.

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 a 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 execution environments—what we call workspaces—onto a set of resources [2, 3]. 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] uses 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 certain 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] led to performance improvements, making VMs 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.

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 [9]. 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 system that dynamically deploys and executes images configured to support the STAR application. This work was demonstrated at SC06.

The platform used for the proof-of-concept was the TeraPort cluster [10] at the University of Chicago. Workspace service version TP1.2 managed four nodes of this cluster, each equipped with 4GB memory and two 2GHz Opterons and configured with Xen 3.0. (The workspace service itself was deployed on another node.) A VM representing an Open Science Grid (OSG) [11] compute element (CE) was statically deployed on the same node as the workspace service. This virtual CE was configured with GRAM2 (to allow users to submit jobs) and Condor (to administer the worker nodes).

The worker node images were developed using the rBuilder configuration manager [12] and consisted of a base configuration (operating configuration, basic OSG software stack: about 1 GB), STAR application (about 3 GB), and a data partition (about 300 MB). Altogether, the worker node image was 4-5 GB. In addition, the nodes required about 3 GB 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. (In a full scenario, this client might be a user or alternatively a resource broker.) A typical resource allocation request asked for roughly 2 GB memory and the full use of a CPU for each virtual node, allowing us to deploy up to eight 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 displayed 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.

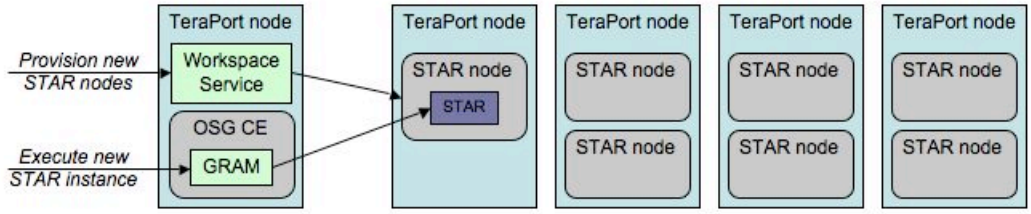

Figure 1: On-demand provisioning of STAR nodes

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. Furthermore, once a STAR image has been configured, it can be run many times without further effort; in contrast, installing and validating STAR on a new platform may take weeks. The experiment also identified new requirements for further refinements in middleware supporting VM deployment. The size of the STAR image is a significant factor: moving it 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. This 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 that can be improved to achieve scalable deployment of virtual platforms. Below, we summarize the conclusions of 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 [13]. 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 [14]; 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). The X509 security infrastructure will need to handle dynamic network environments. 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 [15].

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 [16] 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 and site agreements required for image acceptance, contextualization and management.

5. Ongoing Work

Virtualization allows application users to run on platforms that were not specifically prepared for the application by deploying VM images on those platforms. Thus, users can rapidly scale the resources available to even complex applications. Virtualization can also allow resource providers to reach more users by making their platforms suitable for more applications. This principle has been embraced by industry and resulted in the creation of a number of services selling compute cycles via VMs, of which Amazon Elastic Compute Cloud (EC2) is the best known example [17].

The adoption of virtualization in the scientific community is proceeding at a slow pace, due to lack of experience with the technology. As a result, only limited virtualized resources are available (our proof-of-concept ran on only five nodes). Thus, to provide a production execution platform for users motivated to use virtualization, we have built a workspace gateway to the EC2 service, which uses attribute-based authorization to map users to EC2 accounts [3]. This implementation allows communities motivated to run in VMs to buy virtualized cycles, while using consistent interfaces and images that can later be moved to other virtualized platforms.

Using EC2 allows us to provision resources adequate for a STAR production run run on non-STAR dedicated resources (no initial STAR software support). This is a significant step toward opportunistic use of resources. 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 provision new platforms dynamically 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 in part by 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 contract DE-AC03-76SF00098 and W-31-109-Eng-38, and the SciDAC program under Contract DE-AC02-06CH11357.

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. Vaniachine, A., *DASH: Database Access for Secure Hyperinfrastructure*: OSG document 307. http://osg-docdb.opensciencegrid.org/cgi-bin/ShowDocument?docid=307.
10. *The TeraPort Cluster*: http://www.ci.uchicago.edu/research/detail_teraport.php.
11. *Open Science Grid (OSG)*. 2004: www.opensciencegrid.org.
12. *rPath*: www.rPath.com.
13. 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.
14. Freeman, T., K. Keahey, B. Sotomayor, X. Zhang, I. Foster, and D. Scheftner, *Virtual Clusters for Grid Communities*. CCGrid, 2006.
15. 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.
16. Lu, W., T. Freeman, K. Keahey, and F. Siebenlist, *Making your workspace secure: establishing trust with VMs in the Grid*. SC05 Poster Presentation, 2005.
17. *Amazon Elastic Compute Cloud (Amazon EC2)*: http://www.amazon.com/gp/browse.html?node=201590011.