PROCEEDINGS OF SCIENCE

Contextualization in Practice: The Clemson Experience

Michael Fenn
Clemson University
E-mail: mfenn@clemson.edu

Sebastien Goasguen
Clemson University
E-mail: sebgoa@clemson.edu

Jerome Lauret*
Brookhaven National Laboratory
E-mail: jlauret@bnl.gov

Dynamic virtual organization clusters with user-supplied virtual machines (VMs) have advantages over generic computing environments. These advantages include the ability for the user to have a priori knowledge of the scientific tools and libraries available to programs executing in the virtualized environment as well as the other details of the environment. The user can also perform small-scale testing locally, thus saving time and conserving computational resources. However, user-supplied VMs require contextualization in order to operate properly in a given cluster environment. Two types of contextualization are necessary: image-level and instance-level. Examples of image-level contextualization include one-time configuration tasks such as ensuring availability of ephemeral storage, mounting of a cluster-provided shared filesystem, and integration with the cluster's batch scheduler. Also necessary is instance-level contextualization such as the assignment of MAC and IP addresses. This paper discusses the challenges and techniques used to overcome those challenges in the contextualization of the STAR VM for use with the Clemson University cluster environment that is exposed to OSG. Also included are suggestions to VM authors to allow for efficient contextualization of their VMs and recommendations for future virtualized grids.

13th International Workshop on Advanced Computing and Analysis Techniques in Physics Research
February 22-27, 2010
Jaipur, India

*Speaker.

PoS(ACAT2010)027

1. Introduction

As cloud computing and virtual machines (VMs) become more popular, there is great interest in adapting their use to the many tasks of computational science. However, due to a proliferation of virtualization technologies, a given VM disk image cannot be necessarily used with any given hypervisor. Also, a VM image that must integrate with pre-existing infrastructure must be further modified. This process is known as contextualization [7].

The Virtual Organization Cluster (VOC) model has been developed by the Cyberinfrastructure Research Group at Clemson University in an attempt to ameliorate these concerns. The VOC model provides a framework for developing virtual clusters that seamlessly integrate with existing grid and cloud computing technologies. This integration requires sound principles for contextualization as well as procedures that are guided by those principles. This work seeks to document both.

Central to the VOC model is the idea that a Virtual Organization (VO) is able to provide a customized VM image to the VOC provider. In order to validate this aspect of the VOC model, a real VM image should be deployed on a testbed cluster and a realistic workload run. The STAR VO has long been a proponent of leveraging virtualization in grid computing environments, and since 2007 has worked with the Open Science Grid [1] to investigate the integration of VMs into grid sites. In light of the STAR VO's virtualization expertise and need, it provided the VM image and workload used for this validation experiment [8].

The remainder of this work is organized as follows: the VOC model is discussed in Section 2, contextualization principles and recommendations for packaging VMs according to those principles are presented in Section 3, the procedure followed in contextualizing the STAR VM is described in Section 4, the results of testing performed on that image can be found in Section 5, and conclusions are presented in Section 6.

2. Virtual Organization Clusters

Virtual Organization Clusters (VOCs), first put forward in [9] and illustrated in Figure 1, are a cloud-computing construct that enable the creation of virtual environments. These environments have the properties of being compatible across physical sites, deployable without per-hypervisor node replication of images, transparent to end users, able to be implemented in a non-destructive manner, and customizable by a Virtual Organization. VOCs are made of virtual machines (VMs) spawned from a single image, and thus their worker nodes are trivially homogeneous. If a grid site uses a distributed filesystem such as PVFS [4], VOCs are booted directly from that filesystem, without the need to copy the image to each node. VOCs are transparent because a site appears and is presented as having a normal grid interface to the end user.

A key provision of the VOC model is its separation into two administrative domains, the Virtual Administrative Domain (VAD) and the Physical Administrative Domain (PAD). Each physical site on the grid is a unique PAD containing all hardware, infrastructure, and systems software. Each VOC is a distinct VAD that is managed by the VO. This distinction between PAD and VAD allows each VO to have its own customized, virtualized environment.

Jobs are submitted to the VOC through a dedicated head node. This head node contains a standard installation of the Open Science Grid (OSG) Compute Element software stack. Thus

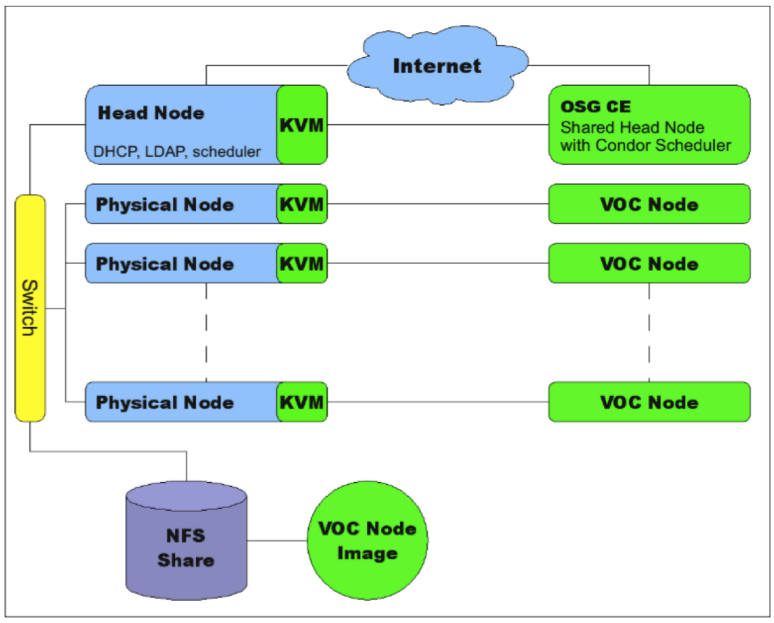

Figure 1: A Virtual Organization Cluster, consisting of a set of virtual machines (VOC nodes) instantiated from a single image file and running on a set of hypervisor-equipped compute nodes.

the VOC appears to the user as a normal site on the grid. A VOC component monitors the job queue on the head node and autonomically starts and stops VMs in response to load. VMs must be contextualized both at boot-time and when the image is received by the VOC via standard grid mechanisms. Boot-time contextualization is defined in the VOC model as the leasing of certain resources from the PAD to the VAD.

As mentioned above, VOCs are autonomically scaled without the user having to make explicit reservation requests [11]. This scaling is accomplished via a daemon that monitors the workload and takes appropriate action. When the daemon detects an increase in job queue length, it interprets this as an increase in the demand for computational resources. The daemon will thus seek to meet the demand by instantiating a new VM from the appropriate VO-specific image. These VMs then join the batch scheduling pool and appear to the user as normal compute nodes. Similar steps are taken when the daemon detects a decrease in demand. The daemon attempts to slowly increase and decrease the number of VMs in order to avoid inefficiencies due to a non-zero VM boot time. It is important to note that each VM utilizes QEMU's *snapshot* mode, in which writes are directed to a local copy-on-write image. These copy-on-write images store the differences from a base image that is itself stored on a network filesystem. The base image does not need to be copied to each hypervisor node before VM instantiation.

3. Contextualization Principles

The topic of VM contextualization merits further discussion. It is a safe assumption that any given VM image will not successfully integrate into a VOC as implemented at any given site. This is due to a variety of factors, including but not limited to: the need to mount any eternal filesystems, the need to acquire an IP address (be it via DHCP or some overlay networking mechanism),

the need to handshake with a batch system, and the need to define any grid-specific environment variables. Thus, the image must be contextualized in two phases: image-level and instance-level. Image-level contextualization occurs once per VM disk image per site. Instance-level contextualization occurs once per VM instance.

3.1 Image-level contextualization

Important considerations for image-level contextualization are image format, image layout, shared filesystem support, and batch scheduler integration. Image format refers to the representation of the disk's data within the image file. Image layout refers to how the various partitions are placed on the disk and to what other disk structures are present.

The simplest image format is that of the raw disk image. A raw image is simply a file containing the exact byte string that would appear on a physical device. This format is highly compatible but is not space efficient because the image file's size must be equal to the capacity of the virtual device being represented. Note that raw images compress very well with gzip compression, so they are fairly easy to distribute. In order to mitigate the in-use size issue, there has been a proliferation of virtual image formats such as VMDK, VDI, VHD, and QCOW2. These formats vary in implementation and hypervisor support, but they all allow the compact representation of a disk image. When utilizing one of these formats, the size of the image is determined by the size of the actual data present on the device, instead of being determined by the capacity of the device. In order to contextualize the VM image format, the image must simply be converted to a format that is compatible with the hypervisor used at a given site. The `qemu-img` [2] tool provides functionality that can convert images between many of the popular formats, thus freeing the user from reliance on any particular hypervisor image format. Hypervisor vendors also generally provide a tool that can convert between their format and the raw format.

The image layout issue can become much more involved. The two main image layouts are the partition image layout and the disk image layout. A partition image contains a representation of a single disk partition. Essentially, this layout could be referred to as a filesystem image, since a partition does not contain any metadata with regard to itself. This layout requires a hypervisor that is able to present individual partitions to a guest OS. Currently, only the Xen hypervisor is capable of this. The disk image layout contains a representation of an entire disk, including the master boot record, boot sector, and partition table. All hypervisors, including KVM, are capable of utilizing this type of image. Since Xen requires the guest kernel and initial ramdisk to be located outside of the VM image, Xen may only boot from disk images when it is used in conjunction with the `pygrub` utility. This utility mounts the disk image and extracts the kernel and initial ramdisk from the image, and as such, can only be utilized with a disk in the raw disk format. There is no set procedure for converting between partition images and disk images. Images will generally need to be converted (at least temporarily) to the raw format in order to allow standard disk tools to be utilized. There are, however, several useful tools and one guiding principle. The principle is: a disk image is the same as a physical disk, and a partition image is the same as a physical partition. Converting between image formats is a matter of getting the correct disk structures into the correct places. Useful tools include:

- `fdisk`, allows the calculation of partition extents and the creation/modification of partition tables,
- `dd`, allows block level copying of defined sections of an image,
- `mount`, when used with the `-o loop` option allows a partition image to be mounted,
- `kpartx`, allows the exposure of the partitions of a disk image as individual devices,
- `chroot`, allows the running of the native tools present in the image if necessary.

These tools, along with the bootloader installer, should be sufficient to assemble a disk image from a set of partition image or decompose a disk image into a set of partition images.

Grid systems have specifications that the compute nodes must adhere to. These specifications generally require that various filesystems be shared among the compute element (CE) and its associated worker nodes. Thus, the image must be contextualized so that it properly mounts those filesystems. In particular, any software libraries needed to mount the site's shared filesystem must be installed and the grid-provided application, user-provided application, and user data shares must the mounted at the locations defined by the CE configuration. One such specification is that of the Open Science Grid which perscribed the definition of the `$OSG_GRID`, `$OSG_APP`, and `$OSG_DATA` environment variables.

There must also be a way to get computational jobs into the VM. Either the site's batch scheduler or a VO-level scheduling system must be installed into the VM image. If the site's batch scheduler is installed, it is prudent to configure the scheduling system in such a way that the VM's scheduling pool may be partitioned off from the site's general scheduling pool in order to satisfy the constraints of the VOC Model. If a VO-level scheduler is installed, some provision must be made for crossing NAT boundaries.

3.2 Instance-level contextualization

Whereas image-level contextualization can be performed manually by a systems administrator, instance-level contextualization occurs once per VM instantiation and as such must be automated. As described in Section 2, certain resources must be leased from the physical site. These resources include network addresses, disk space, and scheduler slots.

Network addresses, including both MAC and IP addresses, should be assigned (leased) to the VMs in such a way as to avoid conflicts. Leasing of MAC addresses must be performed by the hypervisor. Leasing of IP addresses may be performed by the hypervisor if it is capable of passing this information to the guest (e.g. Xen) or may be through the standard DHCP protocol. One such method of assignment is to implement a central leasing server. Before VM instantiation, the hypervisor node would contact a central service and made a lease request for a MAC or IP address. The service would then maintain a lease database in order to avoid duplication. Since MAC and IP addresses will be unique to a hypervisor node, that node may also use a function to map its address to that of the VM. As long as this function will not cause an overlap in addresses, this method satisfies the uniqueness constraint without the requirement of a centralized service.

If the VOC nodes are not spawned from a single image, some allocation of disk space must be made to the hypervisor. This could use hypervisor's local disk, but care must be taken to avoid

exceeding the disk's capacity, especially when dynamically resizing disk image formats are used. Another solution would be to map LUNs of a storage area network to the hypervisor node.

If the scheduling system requires the use of fixed slots for compute nodes, then these must also be assigned [12]. Techniques described for leasing network addresses can be easily extended to provide for such a scheduler.

3.3 Best Practices for VOs

In light of the above discussion, some best practices emerge for VO's that wish to provide a VM. In short, VO's should use disk image layouts in the raw format that either join a global scheduling pool or utilize a common operating system distribution. This advice is expanded upon below.

It is best to provide the VM as a disk image layout in the raw format. The disk image layout is compatible with all hypervisors (although Xen requires the `pygrub` utility) whereas the partition image is only natively compatible with Xen. Substantial administrator effort is required to convert between partition and disk layouts. Similarly, the raw format is preferred due to its broad compatibility. Sites may choose to use the raw image directly, or to convert it to their preferred format. It is however, necessary to compress raw images for transport between sites. It has been observed that raw images compress very well, generally to the size of the actual data.

Two approaches emerge to mitigate the administrator effort needed to install a compatible batch scheduler. First, the VM may be configured to join a global scheduling pool, via mechanisms such as Condor Glide-ins [6], Condor with the IPOP [13] overlay network, or Kestrel [12]. If this is not chosen, the VM should utilize a common distribution of the GNU/Linux operating system such as the various Debian or Red Hat derivatives. The use of such as system maximizes the probability that appropriate packages will exist for the system, thus minimizing administrator effort.

Some VO's may wish to have a measure of certainty that their VM has not been altered from the original image. This guarantee is not easy to enforce in the general case. In fact, it is likely that various sites *must* alter the VM image from its original state due to the factors discussed in this section. It is possible for a VO administrator to sign the VO image with his/her X.509 certificate. This will greatly restrict the VM's usability because the VM will only be able to be deployed at sites which can utilize the provided VM image without modification.

4. Contextualizing the STAR VM

A practical application of the principles and techniques discussed in Section 3 has been performed at Clemson University to enable the contextualization of the STAR experiment's [3, 8] VM. This VM contains the programs and libraries necessary for the simulation and analysis of STAR's experimental data.

The STAR VM image is provided as a Xen partition image named `starworker_part.img` with no bootloader or kernel. Therefore, to use the image with KVM, it is necessary to create a disk device. To do this with the `qemu-img` tool, issue the command `qemu-img create -f raw 10G starworker.img`.

Then, in order to have the appropriate bootloader and kernel installed into the image, a fresh installation of the guest operating system (Scientific Linux in this case) is performed. This instal-

lation should be performed with the target hypervisor. For QEMU/KVM, the invocation command is `qemu-kvm -hda starworker.img -m 512 -net nic -net user`. Note that the contents of a this installation will be completely replaced in a later step, this installation is simply to apply the appropriate partitionioning scheme and to install the bootloader. These two steps may be performed manually if desired.

Now that the kernel and bootloader are installed into the new image, the contents of the root directory must be copied from the provided image to the new image. Do do this, both images must be mounted. However, the new disk image cannot be mounted directly, the partition inside the image must be mounted. There are two ways of doing this: using `kpartx` and calculating the offset manually.

To use kpartx, issue the following commands:

1. `kpartx -l starworker.img` to see which loop devices will be created (some trial and error may be necessary to mount the correct partition),
2. `kpartx -a starworker.img` to actually create the loop devices,
3. `mount /dev/mapper/loop0p# /mnt/loopdisk` where # is the partition number to mount.

To calculate the offset manually issue the commands

1. `fdisk -lu starworker.img` wherein the output will contain a `start` column that contains the offset of each partition as well as a header giving the units of the offset,
2. `mount -o loop,offset=$(( $START * $UNITS )) starworker.img /mnt/loopdisk/` where the `$START` variable has been set with the value of the `start` column and the `$UNITS` variable has been set with the value of the units given by `fdisk`.

Once the appropriate partition in the disk image has been mounted, the partition image is mounted with the command `mount -o loop starworker_part.img /mnt/looppart/`.

Once both images have been mounted, the command `cp -a /mnt/looppart/* /mnt/loopdisk` is used to copy the contents of the partition image into the disk image. Then the images are unmounted with the commands `umount /mnt/looppart` and `umount /mnt/loopdisk`. If `kpartx` was used to mount the partition, the command `kpartx -d starworker.img` is issued to remove the loop devices. At this point, the STAR VM is bootable with KVM and the site-specific batch scheduler and shared filesystems are configured as normal.

Instance-level contextualization also needs to be performed on each instance of the image. Due to the site-local networking and storage environments, the only resource that has to be leased to the VM instance is a MAC address. This is performed via a functional mapping from the hypervisor's hostname to a MAC address, avoiding the need for a central leasing service.

5. STAR VM Results

The STAR VO provided an image that was contextualized for the prototype cluster using the procedures outlined in Section 4. Figure 2 depicts the STAR VM's integration with the prototype

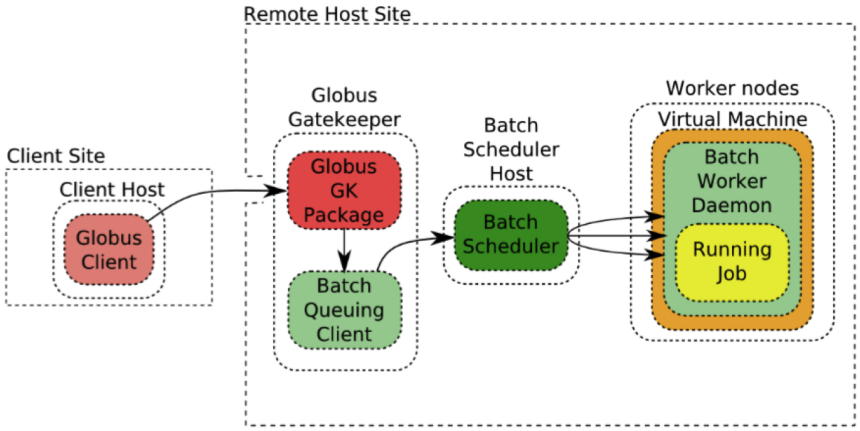

Figure 2: Integration of the STAR VM into the prototype VOC

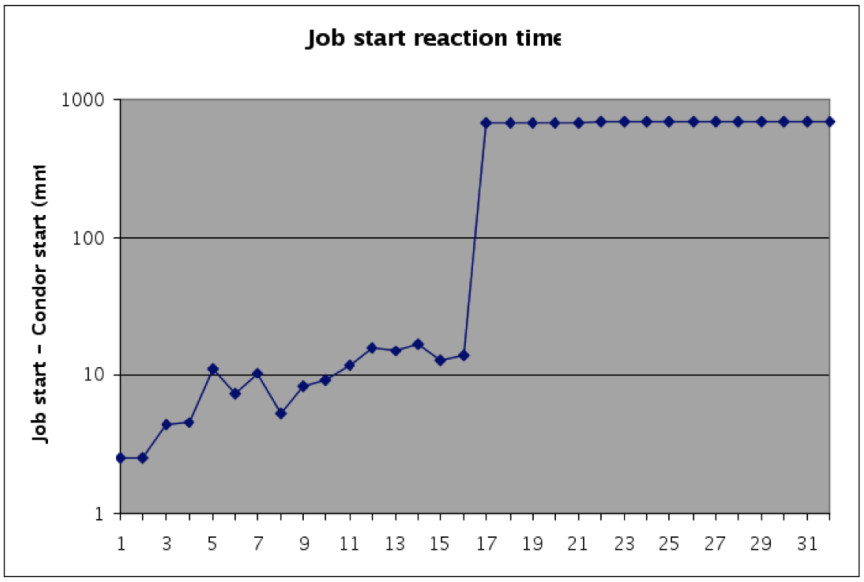

Figure 3: Condor reaction time by Job ID as observed by STAR

VOC. Once the VM has image-level contextualization performed, it appears to the STAR VO to be the same as any other STAR-supporting resource.

STAR utilized the 16 slots available and submitted 32 jobs. The job's 280MB of total output was streamed back to the Brookhaven National Laboratory (BNL) at 6.8MB/s.

The use of network-backed copy-on-write images means that the time required to stage the VM image to the node is negligible. The booting of the VOC nodes added approximately 7 minutes of overhead to the 11 hours of overall walltime required by the workload. Note than an individual VM takes much less than seven minutes to boot [5], but all VMs did not boot in parallel due to behavior of the VOC sizing daemon described in Section 1. The overhead of booting the VMs plus virtualization overhead gives a total VOC overhead of approximately one percent.

As shown in Figure 3, the VOC's Condor scheduler was fast for the first two jobs due to the fact that the watchdog was configured to keep two VMs running at all times. Jobs 2 through 16

started as soon as a VM was started and joined to the Condor pool. Jobs 17-32 were forced to wait in the queue because there were only 16 VOC nodes available. Once the first 16 jobs completed, 16 jobs remained in the queue, causing the VOC sizing daemon to decide against changing the number of VMs. Thus the VMs started in response to the first 16 jobs remained running and Condor was able to schedule the remaining 16 jobs to the VOC nodes without delay. This observation validates prior work that showed that synthetic jobs were well mapped to resources [10].

6. Conclusions

Contextualization of virtual machines is an operational necessity. Of the two stages of contextualization, instance-level contextualization is the most easily automated. Image-level contextualization generally requires systems administrator effort due to the large number of variables present in each specific image, hypervisor, and site configuration. The principles of contextualization that are presented in this work have been shown to be implementable in practice. Unfortunately, due to the complexity of the problem they can only serve as general guides. A worthwhile future goal is to develop a system for automating image-level contextualization.

The Virtual Organization Cluster (VOC) model worked smoothly, with very little (1%) overhead. The model allows a VOC to appear as a normal grid site, thus giving the user confidence in a working and validated software stack. Thus, VOCs show great promise for providing customized environments in a way that is maximally convenient for VOs.

References

[1] Open Science Grid, `http://www.opensciencegrid.org/`

[2] `Qemu-img` man page, `http://linux.die.net/man/1/qemu-img`

[3] The STAR Experiment, `http://www.star.bnl.gov/`.

[4] P.H. Carns, W.B. Ligon, R.B. Ross, R. Thakur, *PVFS: A Parallel File System for Linux Clusters*, in proceedings of *4th annual Linux Showcase and Conference (ALS'00)*.

[5] M. Fenn, M. Murphy, S. Goasguen, *A Study of a KVM-based Cluster for Grid Computing*, in proceedings of *47th ACM Southeast Conference (ACMSE '09)*.

[6] T. Frey, T. Tannenbaum, M. Livny, I. Foster, S. Teucke, *Condor-G: A computation management agent for multi-institutional grids*, *CC* **05** (3) 236-246.

[7] K. Keahey, T. Freeman, *Contextualization: Providing One-Click Virtual Clusters*, in proceedings of *4th IEEE International Conference on e-Science*.

[8] J. Lauret, *Computing for the RHIC Experiments*, in proceedings of *CHEP 2009*.

[9] M. Murphy, M. Fenn, S. Goasguen, *Virtual Organization Clusters*, in proceedings of *17th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP 2009)*.

[10] M. Murphy, B. Kagey, M. Fenn, S. Goasguen, *Dynamic Provisioning of Virtual Organization Clusters*, in proceeding of *9th IEEE International Symposium on Cluster Computing and the Grid (CCGrid '09)*.

[11] M. Murphy, L. Abraham, M. Fenn, and S. Goasguen, *Autonomic Clouds on the Grid, JGC* **08** (1) 1-18.

[12] L. Stout, M. Murphy, S. Goasguen, *Kestrel: An XMPP-based Framework for Many Task Computing Applications*, in proceedings of *2nd Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS 2009)*.

[13] D.I. Wolinsky, Y. Liu, P. St. Juste, G. Venkatasubramanian, R. Figueiredo, *On the Design of Scalable, Self-Configuring Virtual Networks*, in proceedings of *SuperComputing '09*.