Pools of Virtual Boxes

Building Campus Grids with Virtual Machines

David J. Herzfeld[1], Lars E. Olson[1], Craig A. Struble[2]
[1]Department of Biomedical Engineering
[2]Department of Mathematics, Statistics, and Computer Science
Marquette University
P.O. Box 1881
Milwaukee, WI 53201-1881
firstname.lastname@marquette.edu

ABSTRACT

A campus grid is a critical component of research cyberinfrastructure. A grid facilitates resource sharing, improving collaboration and increasing the capability to address large scale scientific and engineering problems. Resources comprising a grid include research clusters, centrally managed clusters, opportunistic use of desktop computers, campus clouds, and commercial clouds.

Advances in virtual machine technology and hypervisor availability has made the use of virtual machines an attractive tool for building campus grids. *Pools of Virtual Boxes* (POVB) is an open-source, dedicated virtual machine environment for rapidly deploying a campus grid. POVB is targeted at institutions where financial and administrative constraints prevent large scale changes in computational infrastructure.

The POVB distribution includes: services that manage the virtual machine hypervisor, services that communicate select host information with the Linux guest for debugging and resource utilization policies, a bootstrapping framework for building virtual images, and a third-party package deployment framework that integrates with Condor to advertise available software services. We report on the design and implementation of POVB and examine a deployment of several hundred POVB instances, which has been operational for over a year.

POVB has been released under the GNU Public License Version 3, and is available at `http://poolsofvirtualb.sourceforge.net`.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems; D.2.7 [**Software Engineering**]: Distribution, Maintenance, Enhancement

HPDC'10, June 20–25, 2010, Chicago, Illinois, USA.

General Terms

Virtualization, Distributed Computing

Keywords

Virtual Machines, High Throughput Computing, Deployment, Campus Grid

1. INTRODUCTION

A campus grid is a critical component of research cyberinfrastructure. A grid facilitates resource sharing, improving collaboration and increasing the capability to address large scale scientific and engineering problems. Resources comprising a grid include research clusters, centrally managed clusters, opportunistic use of desktop computers, campus clouds, and commercial clouds. Middleware such Condor [21] and Globus [11] tie these resources together and provide application programming interfaces (APIs) for distributed computation.

Advances in virtual machine technology and hypervisor availability has made the use of virtual machines an attractive tool for building campus grids. *Pools of Virtual Boxes* (POVB) is an open-source, dedicated virtual machine environment for rapidly deploying a campus grid. POVB is targeted at institutions where financial and administrative constraints prevent large scale changes in computational infrastructure. Deployment of POVB requires overhead similar to installing an application in an enterprise environment. Management requires knowledge of administrating Condor, the grid middleware chosen by POVB.

The use of dedicated virtual machines have several specific advantages over direct access to a host system: 1) they decouple the host operating system used for typical academic purposes from the guest operating system configured for typical research needs; 2) they potentially provide a layer of security by sandboxing applications from the host operating system; 3) they do not require substantial changes in existing hardware infrastructure; and 4) host-level monitoring of the virtual machine can be used to identify and recover from problems within the virtual machine. Virtualized machines present some disadvantages: 1) performance overhead due to virtualization; 2) system management challenges due to increased numbers of active machines; and 3) decreased robustness due to flaws or interaction issues with the virtualization software and hardware. Performance overhead is a minor issue, because new compute cycles are made available

even if they are less than optimal. POVB strives to simplify management and increase robustness of deployed systems. Section 3, describes POVB in more detail. Related work is reviewed in Section 2.

POVB systems have comprised a significant portion of the Marquette University Grid (MUGrid) for over a year. Deployment across campus continues, as departments and colleges agree to contribute resources. Over the course of creating MUGrid, we have developed several deployment and maintenance strategies, which are now part of POVB and are described in Section 4. Further discussion of applications running on POVB and observations made about MUGrid resources is provided in Section 5. We conclude the paper in Section 6 and describe future improvements to POVB.

2. RELATED WORK

The inspiration for POVB was the Cooperative Linux (coLinux) based Condor distribution developed at the University of Nebraska—Lincoln and the University of Oklahoma [3, 32], which is predated by an IBM article describing a similar use of coLinux [30]. Cooperative Linux [2] is a user-mode Linux kernel designed to run cooperatively in a Windows® environment as a service. In addition to the standard coLinux distribution were services to monitor host activity and to communicate that information to the coLinux guest. Condor was configured to incorporate the host activity in resource access policies, ensuring jobs run only when the host is idle. The coLinux implementation has extremely low overhead, but does not take advantage of multi-core platforms and cannot run in 64-bit environments.

The Grid Appliance project aims to build ad-hoc pools of resources [33, 34]. It shares many similarities with POVB, including using a virtual machine hypervisor, support for add-on packages in the guest environment, an application for monitoring host activity, and relatively easy deployment. The most novel contribution of Grid Appliance is the use IPOP, a peer-to-peer overlay network infrastructure allowing for the creation of wide area, ad-hoc networks to construct grids across a variety of network infrastructures [13, 14]. The use of overlay networks causes security and network overhead concerns for some system administrators, requiring other networking options. Furthermore, Grid Appliance is not specifically designed to run as a background process, although it could be adapted to do so using approaches similar to those developed in POVB.

A widely successful system using virtualization techniques has been VIOLIN [17, 27]. Initial versions of VIOLIN used User Mode Linux, similar to coLinux, to create virtual overlay networks similar to the IPOP system in Grid Appliance. Much additional related research has been based on VIOLIN, including live adaptation of virtual machines [29], and dynamic allocation of computational domains [28]. VIOLIN has served as the foundation for nanoHUB [19], an environment supporting the nanotechnology community that uses virtual machines to provide TeraGrid resources for simulations.

Condor's virtual machine (VM) universe allows VM images to be run as high-throughput computing jobs [21]. This feature has been used to deploy Linux and other virtual machines allowing for on-demand provisioning of resources and very general applications. Guests in the VM universe do not explicitly know about the hosts they run on, making it difficult to transfer information between hosts and guest. Furthermore, on-demand deployment of VM images can be slow if performed across standard 100Mbit or gigabit network infrastructures commonly found in universities and other locations. Such a deployment requires Condor to run on the host platform, which may not be desirable in some environments.

Finally, cloud computing is gaining popularity and infrastructures such as Amazon's EC2 [1], Eucalyptus [22], and Nimbus [18] are providing both commercial and open source opportunities to provision resources on-demand. Cloud computing provides great flexibility for deliverying cyberinfrastructure and has been used to create virtual clusters and grids similar to what is provided by POVB [12]. Indeed, on-demand configuration and management of resources offers significant advantages over POVB, but at the cost of restructuring an enterprise's infrastructure or outsourcing the resources to a cloud vendor. Such options are not feasible at many institutions with limited resources or declining budgets due to: 1) costs that may be incurred in restructuring their network and storage infrastructure to support cloud solutions; 2) recurring and unbudgeted costs related to commercial solutions (particularly data transfer and storage costs, which often far outweigh the cost of compute hours); 3) training IT staff to support a research cloud environment, which is often beyond their purview; and 4) data security issues that may restrict analysis to institutional machines only. Several of the obstacles to adopted cloud computing have been described previously [5], and a cost comparison between cloud computing and desktop grids concludes that desktop grids remain cost effective given the pricing models of commercial clouds at the time this paper was written [20]. Nevertheless, given appropriate resources and policies, cloud computing platforms can greatly enhance a campus grid environment.

3. SYSTEM DESCRIPTION

The POVB concept is comprised of six layers, depicted in Figure 1. The *host layer* represents the host hardware platform upon which POVB runs. The *host services layer* are POVB services interfacing between the host and virtual machine hypervisor. These services configure the hypervisor, monitor hypervisor activity, and communicate information from the host to the guest system. The *hypervisor layer* is the chosen hypervisor technology used for virtualization. As implied, hypervisors supported by POVB run on a host platform in such a way that they can be managed. The *guest layer* represents the deployed guest operating system. The *guest services layer* are POVB services inside the guest receiving host information provided by the host services layer. This layer also provides additional monitoring and value added capabilities to the guest operating system and grid middleware, such as integration of third party packages. The *grid middleware* is the middleware chosen for providing grid functionality. Figure 1 provides the currently implemented instance of the POVB concept, which builds on Windows as the host operating system, Linux as the guest operating system, VirtualBox as the hypervisor, and Condor for grid middleware. Other configurations for POVB are possible in the future, depending on demand.

Each layer is described in more detail in what follows. Throughout, we include sample configuration lines and commands in fixed-width text to illustrate the POVB implementation.

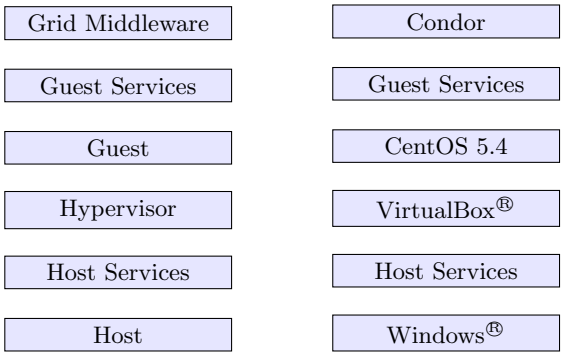

Figure 1: *Left:* The layered POVB concept. *Right:* A specific instance of the POVB concept.

3.1 Host Services

Figure 2 depicts the host services implementation of POVB on a Windows® host. POVB host services consist of three Windows® services:

- The main *POVB Service*, starts by configuring the networking, memory, number of processors, and other aspects of the VirtualBox® hypervisor. The service also removes any disk snapshots to start the virtual machine from its initial installed state. Once configuration is complete, the hypervisor is launched and monitored to verify that it remains running.
- The *Host Monitor* passes the Windows® hostname, system load, memory usage, and other global system information to the VirtualBox® guest through a shared folder. The shared folder is mounted read-only by the guest.
- The *Activity Monitor Daemon* is launched by the *Host Monitor* when a user logs in. The daemon records the last time either keyboard or mouse activity occurred on the machine and is primarily used for defining policies for job initiation and preemption.

3.2 Hypervisor Configuration

The POVB hypervisor configuration for VirtualBox® is shown in Figure 3. Two disk images are attached to the virtual machine, one containing the primary operating system installation, the other containing grid middleware support files. These drives are *immutable* and runtime changes are stored as snapshots against the drives. POVB removes any snapshots upon startup to return the virtual machine to its original installed state when the host machine is rebooted. This makes POVB robust to host shutdowns or poorly written jobs. A simple hypervisor restart brings the system back to a known, stable state.

Any VirtualBox® configuration option can be managed. POVB supports both *bridged* and *NAT* networking, which is chosen at installation time. The chosen networking mode is communicated with the POVB guest services to configure grid middleware appropriately. Both 32-bit and 64-bit guests are supported. The POVB service identifies the availability of hardware virtualization and multi-core architectures and configures the hypervisor to use these underlying capabilities. System administrators can override these

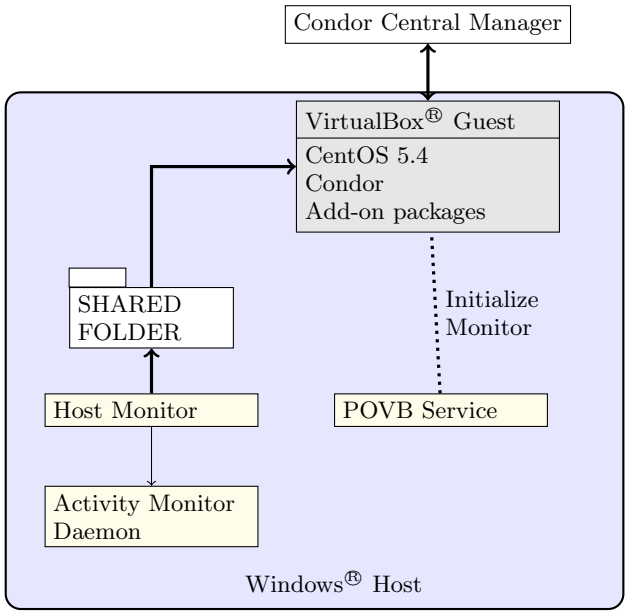

Figure 2: The POVB host-guest architecture. VirtualBox® hypervisor runs a Linux distribution configured with POVB scripts, Condor middleware and additional packages appropriate for the specific institution. CentOS 5.4 is the primary distribution supported.

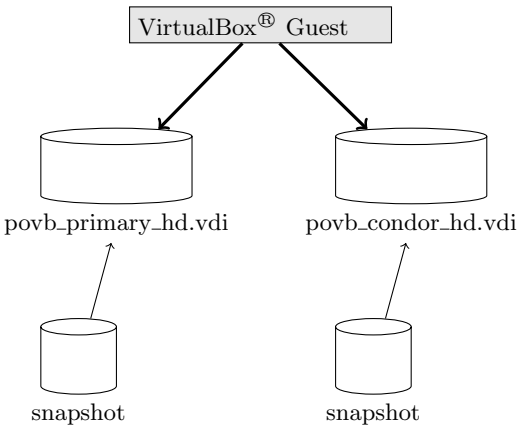

Figure 3: The POVB virtual machine configuration.

choices at installation time, to best support their environment.

3.3 Guest

The VirtualBox® hypervisor allows for the installation of unmodified guest operating systems. POVB virtual machines have been configured for CentOS or Fedora Core Linux distributions. Other Linux distributions are possible, but require additional development for bootstrapping, grid middleware integration, and POVB guest service integration. The bootstrapping process is described in Section 4.2.

3.4 Guest Services

Guest services have been implemented as initialization scripts for Linux running on VirtualBox®, and as *Hawkeye* scripts in the Condor system for grid middleware integration. These services read configuration and information files provided by the host services to the guest through a shared folder, mounted read-only by the guest. Configuration and information files are given an MD5 hash signature by the host services, which is used to verify their integrity since shared folder locking is inadequate and historically simultaneous reading by the guest and writing by the host has been buggy. We have taken this approach to guard against upstream VirtualBox issues, which are beyond our control. We are considering other approaches for sharing information, such a having POVB run a simple web server connected to a network interface dedicated for host/guest communication, or the use of VirtualBox guest properties, which are more robust. Another approach is the use of Chirp, recently introduced in a BOINC/VirtualBox prototype [26].

Configuration files describe how the associated grid middleware should be configured to participate in the campus grid. In addition, resource usage policies are communicated through the shared folder to make it easier to effect policy changes from the host. For instance, configuration files include references to the hostname of the Condor central manager, the guest's assigned domain name, and the location of keys/certificates which allow communication between the Condor central manager and the guest to be encrypted. Using the configuration below, the POVB guest services will modify both the hostname and Condor configuration files appropriately.

```
DOMAIN = myschool.edu
CM_FULLNAME = central-manager.myschool.edu
CONDOR_KEYFILE = condor_ssl.key
```

Information files contain host information, such as activity and CPU load, to incorporate in policy implementation. Additional host identification information may also be provided to assist in identifying problematic hosts and for application debugging. These information files use a semi-structured data model similar to Condor's ClassAds [9]. This form provides enough structure to capture necessary information about the state of the host and allows for advertisement to pool administrators for identification of problems (see Section 3.5). The example below shows the total CPU load of the host system as well as the load associated with the POVB hypervisor. In addition, a value describing the host idle time in seconds is provided. This host information can then be used by pool administrators to define scheduling policies for Condor jobs.

```
HostOsLoad = 0.05
POVBLoad = 0.03
HostOsKeyboardIdle = 1532
MemoryLoad = 0.69
```

In the above sample, `HostOsLoad` represents the load on the host system, `POVBLoad` is the load incurred by POVB host services and the virtual machine hypervisor, `HostOsKeyboardIdle` is the number of seconds since the last keyboard or mouse activity on the host, and `MemoryLoad` is the proportion of memory in use on the host system. This information is read by a Hawkeye job, configured in Condor by the following lines:

```
STARTD_CRON_JOBLIST = UPDATESTATS
STARTD_CRON_UPDATESTATS_PREFIX = HOSTINFO_
STARTD_CRON_AUTOPUBLISH = Never
STARTD_CRON_UPDATESTATS_EXECUTABLE = \
    /home/condor/read_stats.sh
STARTD_CRON_UPDATESTATS_PERIOD = 30s
STARTD_CRON_UPDATESTATS_MODE = WaitForExit
```

The following expression defines the host information that can be used in policy expressions.

```
STARTD_EXPRS = HOSTINFO_HostOsLoad, \
  HOSTINFO_HostOsKeyboardIdle, HOSTINFO_POVBLoad, \
  HOSTINFO_WindowsHostname
```

For the sake of brevity, the `HOSTINFO_` prefix is removed in the text. The `HostOsLoad` expression contains the load of the overall host system. The `HostOsKeyboardIdle` expression contains the length of time since the keyboard has been idle. The `POVBLoad` is the system load incurred by POVB. Finally, the `WindowsHostname` is the host name of the machine.

This information is used to define expressions for policies. The following lines define an expression saying that jobs can start if the host has been idle for more than 15 minutes, and the CPU is either idle or not unclaimed or otherwise in the owner state (the last part of the expression is in Condor's default configuration).

```
START = ( ( HOSTINFO_HostOsKeyboardIdle > 15*60 ) \
        && ( $(CPUIdle) || \
              ( State != "Unclaimed" && \
                State != "Owner" ) ) )
```

Because a guest is typically unaware of the host system load, the additional POVB configuration variables allows Condor policies to have some knowledge of the host they run on.

3.5 Grid Middleware

The grid middleware used by our current POVB implementation is Condor. We chose this middleware because it is specifically designed for opportunistic use of resources, the primary resource available in the typical general purpose lab environment.

Condor is configured by POVB guest services, which receives policies and other relevant configuration information from the host through shared folders. One example is the network configuration. If the hypervisor is configured for bridged networking, Condor is configured to talk directly to a central manager. If NAT networking is used, Condor is configured to use the Condor Connection Broker (CCB) for

communications. The CCB allows for communication between Condor daemons by making all connections go out from NAT boxes rather than Condor's traditional two-way communication approaches. The following lines are included in Condor's local configuration by POVB guest services with NAT networking.

```
CCB_ADDRESS = central-manager.myschool.edu
PRIVATE_NETWORK_NAME = $(FULL_HOSTNAME)
```

Another configuration step is to create dedicated Condor users for job execution. Grid users do not have accounts on POVB virtual machines. By default Condor runs jobs as the untrusted `nobody` user. On multi-core systems, it is possible for jobs in different computational slots to interact as all jobs run as `nobody`. Dedicated users allow the Condor system to segregate jobs running on different Condor slots and to clean up more effectively upon job completion.

```
SLOT1_USER = cnduser1
SLOT2_USER = cnduser2
...
DEDICATED_EXECUTE_ACCOUNT_REGEXP = cndrusr[1-32]+
STARTER_ALLOW_RUNAS_OWNER = False
```

Integration with Condor's ClassAd Mechanism.

Condor's ClassAd mechanism allows both jobs and computing resources to advertise their attributes. Since the guest operating system is known, POVB is able to provide additional information to Condor users for matching purposes. For instance, the existence of common Linux software such as compilers, supported programming languages, and system libraries are advertised to allow matching with user jobs. Additionally, information about the host computer including its Windows domain and host name are provided. These additional ClassAds allow administrators to easily monitor host systems and perform maintenance when appropriate. The following are examples of POVB custom ClassAds:

```
HOSTINFO_WindowsHostname = "MYMACHINE-01"
HOSTINFO_WindowsDomainName = "myschool.edu"
HasPython = TRUE
HasGCC = TRUE
```

Users submitting jobs can now create job requirements to access POVB machines configured with software they need, setting `REQUIREMENTS` in their submission files.

```
REQUIREMENTS = HasPython =?= TRUE
```

Another use of these ClassAds has been to develop system administration tools using constraints to list only POVB systems:

```
condor_status -constraint \
   'HOSTINFO_WindowsHostname =!= UNDEFINED'
```

3.6 Third Party Packages

Since Condor deals with distributed resources, users are usually unaware of the specific software packages available on a particular execution machine. Therefore, unless a shared file system has been previously specified, Condor users must distribute a copy of their required software with their job.

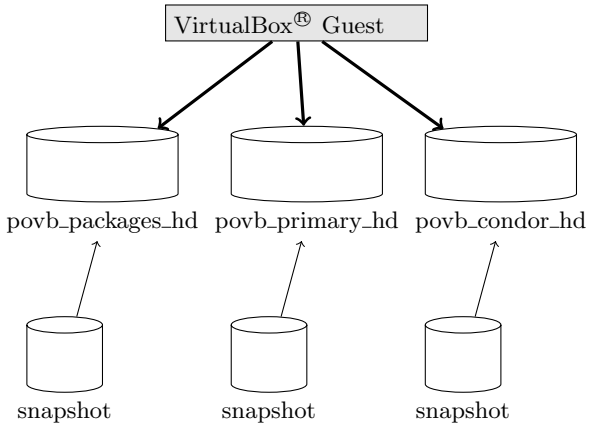

Figure 4: A third hard drive is used to distribute optional packages.

Depending on the size of the software package or required setup procedure, it may be impractical or impossible to distribute certain desired software packages. POVB provides a solution to this problem by providing an additional packages hard drive (see Figure 4). Administrators can quickly set up POVB hard disk images with software specific to current research interests and then attach this image to the POVB virtual machine. Using the Condor ClassAd mechanism, POVB advertises the existence of the attached packages so that user jobs may be matched against them. Each of the packages usually describes both the name of the software package as well as the version, allowing users to be very selective in the choice of machines their jobs run on.

```
HasGate = TRUE
Gate = "5.0.0"
HasMatlab = TRUE
Matlab = "R2009A"
```

POVB also eases access to third party packages using Condor's *job hooks.* A grid user adds lines to their Condor submission file indicating which packages their application uses.

```
+HookKeyword = "POVB_PACKAGES"
+POVB_PACKAGES = "GATE,MATLAB"
```

POVB reads this information and runs setup scripts to create a functional execution environment for the specified packages. At this time, POVB does not check for package conflicts, leaving the grid user responsible for avoiding, detecting, and recovering from such conflicts.

4. DEPLOYMENT AND MAINTENANCE

In academic environments with large numbers of Windows systems, the ability to deploy large pools of POVB workers with minimal configuration overhead is desirable. Therefore, POVB virtual machines are set up to reconfigure themselves upon system start up. In order to ensure that each POVB virtual machine has a unique host name, the host name is reassigned based on a modification of the MAC address of one of the host system's network adapters. Additionally, a modification of a host interface MAC address is also assigned to the virtual machines interface. In the case of bridged

networking, this confirms two things: First, based on their MAC address, virtual machines can be traced back to their Windows host. Second, it ensures that virtual machine interfaces have a unique MAC address. The firewall inside the virtual machine's is also recreated upon boot to provide more secure access from the outside world.

Prior to starting the virtual machine, the master service determines the amount of physical memory and the number of CPUs available on the host system. The memory for the virtual machine is then assigned as a fractional amount of the total memory. Similarly, the virtual machine is reconfigured such that the number of virtual CPUs matches the number of physical CPUs. Therefore, administrators can roll out images containing POVB without worrying about setting the networking, memory, or CPU settings for the virtual machines manually.

4.1 Deployment

POVB workers have been successfully installed using a number of deployment solutions. The current POVB implementation is designed to be deployed as a typical Windows® application. The POVB installer downloads and installs VirtualBox®, and installs the Windows® services. A POVB user is expected to have a Condor central manager (or equivalent) for their campus grid, which POVB is configured to use by editing a configuration file.

System imaging techniques, even for heterogeneous hosts, have been used to successfully deploy POVB workers on multiple systems at a time. Additional active pushing of POVB to machines has been accomplished by applications such as Altiris® and Faronics Deep Freeze®. POVB also contains a batch installation script for installation on smaller pools that do not desire to deploy using commercial solutions. Manual installation has been done from flash drives and network mounts. Typical time for manual installations is a few hours for approximately 100 machines on a gigabit network. Network bandwidth is the most limiting factor for deployment.

4.2 Bootstrapping Framework

In order to provide a useful computational tool to a diverse community of researchers, POVB has a bootstrapping framework that allows individuals to create custom POVB distributions. As its input, bootstrapping takes a Linux operating system installation media and an associated kickstart file. While POVB currently focuses on the CentOS and Fedora operating systems, the framework could be extended to other distributions. Finally, the administrator defines a set of operating system and third-party packages to be included in the installation process. Once initiated, bootstrapping proceeds without manual intervention, taking advantage of VirtualBox's ability to inject keystrokes from the command line to the virtual machine.

The output of a successful bootstrapping process is a set of hard drive images that contain the necessary operating system components, the Condor subsystem, and additional third-party software packages. The hard disk images are bundled with the host service implementations to provide headless Condor execute nodes or, with minor adjustments, these images could be used to provide an interface to users for Condor job submission.

4.3 Maintenance and Debugging

Most maintenance of POVB systems is achieved through reboots of the host system. Because the POVB services remove virtual hard drive snapshots, the virtual machines are returned to their installed state after a reboot. This is clearly a heavy-handed technique for maintenance, but it is also one consistent with how many Windows® deployments are managed.

Updates to POVB installations are done through the deployment strategies in Section 4.1. The POVB installer automatically checks for previous installations, removes them, and then installs the new version.

Online maintenance and debugging is challenging in an environment like POVB. Because POVB hard drives are immutable and snapshots are removed on reboot, remote administration is used primarily for debugging and short-term fixes. Users are provided the ability to install SSH public keys for the root account in POVB systems, which allows remote administrative access if bridged networking is used.

For NAT networking, an OpenVPN [24] SSL-based VPN network can be created for administrative purposes. Users provide the necessary SSL keys and OpenVPN configuration file to establish the network connection. The POVB guest services detects the existence of these files and will attempt to establish the connection. The Condor ClassAd for the POVB machine advertises the success or failure for the OpenVPN connection.

If necessary, an administrator can physically log into a host machine running POVB and launch the virtual manually to gain console access. This is typically used for NAT networked machines without an OpenVPN administrative network, or when the machine has stopped responding to the network (such as when a kernel panic has occurred).

5. EXPERIENCE REPORT

POVB has been in active development at Marquette University since August 2008. The first versions of POVB were based on the coLinux configuration described by the University of Nebraska–Lincoln and the University of Oklahoma [2]. The initial deployment was on approximately 100 machines in the Department of Mathematics, Statistics, and Computer Science (MSCS) and the College of Engineering (COE) at Marquette University. Through this experience we identified several key limitations that needed to be addressed before wide spread adoption at Marquette could take place: *1) coLinux was not capable of running on 64-bit systems; 2) coLinux could not take advantage of multi-core systems; 3) installation and configuration of coLinux by our system administrators was tedious; 4) coLinux development was slow and fragile.*

We reviewed the options for virtual machine hypervisors running on Windows® and decided on VirtualBox® [25] because it is open-source, portable, and has very active development from a commercial entity (Sun, now Oracle) and its open source community. We have been actively running the VirtualBox® version of POVB since January 2009. It is now installed on over 300 machines at Marquette University and has over 500 computational slots available.

POVB machines are currently deployed on Windows XP, Vista, and Windows 7 systems throughout Marquette University. The host machines in this pool are heterogeneous. The pool contains 32 bit and 64 bit AMD and Intel processors. Machines are both single and multicore with memory sizes ranging from 512MB to upwards of 4GB. However, each of these machines appear to Condor users as 32-bit

Intel/Linux machines for a uniform distributed computing environment.

Management of Marquette University's resources is largely decentralized. System administrators managing the host resources are responsible for installing POVB using their Windows® deployment tools. The POVB staff are responsible for configuring and managing the Condor pool, as well as preparing updates to POVB on a regular basis. Installation of POVB is routinely planned 2-3 times a year, coinciding with university breaks to avoid inconvenient lab outages. In some departments, updates are pushed out overnight as needed, when Windows® deployment tools are in place for such updates.

Once deployed, most of the management time for POVB is handled by the POVB staff, and most of that time is spent answering questions about using Condor or debugging users' applications. Very little time is needed monitoring the POVB systems, and most problems are resolved by rebooting the host.

5.1 Details of POVB Deployment

Many of POVB's features are in production at Marquette. In the MSCS pool, bridged networking is used for 90 machines and IP addresses are managed using the department DHCP server. In the COE pool, NAT networking and Condor CCB is used for over 200 machines. No additional service infrastructure is needed to support the COE pool.

Although no formal studies of performance have been done on our POVB resources, we report some anecdotal performance results. POVB machines typically require approximately 20 minutes to fully join a Condor pool. The delay is primarily due to activity data collection on the host and the default policy that a machine must be idle for 15 minutes before being considered in Condor's idle state.

Throughput on our POVB pool has been adequate for users needs. In one evaluation of genomewide association study performance (see below for further details), 3417.2 CPU hours of work were completed in 27.5 wall clock hours for a 124x speed up using 154 job slots. This represents approximately 80% efficiency, defined by speedup/jobslots.

The most frequently executed workflow consists of over 10,000 protein docking simulations (see below for further details). A full run takes approximately 10,000 CPU hours (i.e., one hour per simulation), and is completed in approximated 3.5 days using 420 slots. This is a 119x speedup representing an efficiency of 28.3%. The major difference in this efficiency is in part due to increased data transfer overhead as each job transfers 150MB across a 100Mbit/s link, which is not accounted for in CPU hours, and more frequent preemptions of jobs because of the length of the workflow.

5.2 Applications Using POVB

Our active pool of POVB resources has supported several projects at Marquette University. Below, three of the most developed projects that currently use these resources are described.

Biomedical Image Processing.

Dr. Taly Gilat-Schmidt in the Department of Biomedical Engineering at Marquette University investigates methods for improving image quality and reducing the radiation dose of computed tomography (CT) and single-photon emission computed tomography (SPECT) imaging systems. Monte Carlo methods model the physics of photon transport and track the trajectory of individual photons as they travel through the imaged object. A typical CT simulation may track as many as 50 billion photons, a task that is intractable without large-scale parallelization [7, 31]. POVB machines on MUGrid have contributed approximately 24,000 CPU hours towards this project from August 2009–February 2010.

Protein Docking for Drug Discovery.

The laboratory of Dr. Daniel Sem in the Department of Chemistry is focused on understanding and manipulating protein-ligand interactions, especially those of relevance in drug design and toxicology. In general, computational work is devoted to: (a) semi-empirical (AM1) quantum mechanical calculations of ligand structures, (b) computational docking of these ligands into protein structures, (c) calculation of protein structures, and (d) calculation of protein motion / dynamics [6, 10, 15, 23]. Dr. Craig Struble and Dr. Sem's students have developed a pipeline to dock 10,000 small molecules with several important protein structures to identify potential drug leads. This project has used approximately 31,000 CPU hours on POVB machines from August 2009–February 2010.

Genome-wide Association Studies.

Dr. Craig Struble is developing a scalable framework to support genome-wide association studies (GWAS), a tool for determining associations between genetic variants and phenotypes, most commonly disease [8]. Several aspects of the GWAS process including data cleaning, imputation, and complex analyses, are computational demanding but can be feasibly distributed across multiple machines. During August 2009–February 2010, approximately 20,000 CPU hours on POVB machines have been used to analyze data and develop new approaches for missing data correction.

6. CONCLUSIONS AND FUTURE WORK

POVB is an integrated distribution of host services, hypervisor, guest operating system, guest services, and grid middleware for rapid deployment of virtual machine based campus grids. It is currently designed to bridge the gap between largely idle Windows® workstations commonly available in university labs and the research needs of faculty relying predominantly on Linux/Unix environments. Condor is used as the grid middleware because of its ability to leverage idle resources and flexibility in configuration. POVB integrates third-party packages into the Condor system for easier access and use by grid users. POVB is distributed under the GNU General Public License, version 3, and is available for download at `http://sourceforge.net/projects/poolsofvirtualb`.

The POVB platform has been deployed for over a year at Marquette University and has provided over 100,000 CPU hours of computation time to a variety of projects. At the time of this paper, POVB is deployed on over 300 machines providing over 500 computational slots of opportunistic resources. These slots are integrated into the larger MUGrid, which also has parallel computing resources and host-based opportunistic resources.

While POVB is providing substantial resources for a campus community, there are several additional features planned in the future. As has been demonstrated by cloud computing

and other virtual machine deployments like VIOLIN [27], on-demand deployment and adaptation provides powerful opportunities for distributed computing. The authors plan to investigate the use of the Condor native implementation for on-demand deployment, updates, configuration, and launching of POVB machines.

Jobs supported by POVB run in Condor's vanilla universe, which lacks job checkpointing. Integration of tools such as Berkeley Lab Checkpoint/Restart [16] or DMTCP [4] can provide these capabilities in POVB systems. Another feature under investigation is virtual machine migration. Migration is available in current versions of VirtualBox®, but requires shared access to the hard drives, something that is not typically available in lab environments. Furthermore, on a campus scale the host machine architectures and operating systems vary, complicating migration of virtual machines.

Finally, investigation of better administration tools is underway. The difficulty of online access to POVB machines, particularly in NAT networking, adds some security but also prevents administrators from debugging problems or performing quick, temporary patches. Using overlay networks like IPOP [14] as an administrative channel might help to alleviate these challenges and open up new deployment possibilities.

7. ACKNOWLEDGEMENTS

Funding has been provided by National Science Foundation award OCI-0923037, the Department of Biomedical Engineering, Department of Mathematics, Statistics and Computer Science, and the College of Engineering at Marquette University. The authors would like to thank Information Technology Services and Raynor Memorial Libraries at Marquette University for deploying POVB, particularly Patrick Blume for his feedback on administrating the POVB installations. The authors also thank the Condor Team at the University of Wisconsin–Madison for their feedback, help on configuration, and rapidly addressing bugs in the Condor system that have been identified during POVB development.

8. REFERENCES

[1] Amazon elastic compute cloud (ec2). Available from: `http://aws.amazon.com/ec2/`.

[2] Cooperative linux. Available from: `http://www.colinux.org/`.

[3] Alexander, J., and Franklin, C. Implementing linux-enabled condor in windows computer labs. Available from: `https://twiki.grid.iu.edu/pub/Education/GTGS2008Syllabus/Implementing_Condor_in_Windows.pdf`.

[4] Ansel, J., Arya, K., and Cooperman, G. DMTCP: Transparent checkpointing for cluster computations and the desktop. In *23rd IEEE International Parallel and Distributed Processing Symposium* (Rome, Italy, May 2009).

[5] Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al. Above the clouds: A berkeley view of cloud computing. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28* (2009).

[6] Bazeley, P. S., Prithivi, S., Struble, C. A., Povinelli, R. J., and Sem, D. S. Synergistic use of compound properties and docking scores in neural network modeling of cyp2d6 binding: predicting affinity and conformational sampling. *J Chem Inf Model 46*, 6 (Nov-Dec 2006), 2698–2708.

[7] Bhagtani, R., and Schmidt, T. G. Simulated scatter performance of an inverse-geometry dedicated breast ct system. *Med Phys 36*, 3 (Mar 2009), 788–796.

[8] Bolte, S., Luczkowski, E., Jailwala, P., Serafin, M., Stamm, K., Wang, T., Twigger, S., Ghosh, S., and Struble, C. A scalable information content-based approach for genomewide-association studies. In *Proceedings of American Society of Human Genetics* (2009).

[9] Coleman, N., Raman, R., Livny, M., and Solomon, M. Distributed policy management and comprehension with classified advertisements. Tech. Rep. UW-CS-TR-1481, University of Wisconsin - Madison Computer Sciences Department, April 2003.

[10] Costache, A. D., Trawick, D., Bohl, D., and Sem, D. S. Aminedb: large scale docking of amines with cyp2d6 and scoring for druglike properties–towards defining the scope of the chemical defense against foreign amines in humans. *Xenobiotica 37*, 3 (Mar 2007), 221–245.

[11] Foster, I. Globus toolkit version 4: Software for service-oriented systems. *Journal of Computer Science and Technology 21*, 4 (2006), 513–520.

[12] Foster, I., Freeman, T., Keahey, K., Scheftner, D., Sotomayor, B., and Zhang, X. Virtual clusters for grid communities. In *Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid* (2006), pp. 513–520.

[13] Ganguly, A., Agrawal, A., Boykin, P., and Figueiredo, R. Ip over p2p: Enabling self-configuring virtual ip networks for grid computing. In *Proc. of the IEEE Intl. Parallel and Distributed Processing Symp.(IPDPS)* (2006).

[14] Ganguly, A., Agrawal, A., Boykin, P., and Figueiredo, R. Wow: Self-organizing wide area overlay networks of virtual workstations. *Journal of Grid Computing 5*, 2 (2007), 151–172.

[15] Ge, X., Wakim, B., and Sem, D. S. Chemical proteomics-based drug design: target and antitarget fishing with a catechol-rhodanine privileged scaffold for nad(p)(h) binding proteins. *J Med Chem 51*, 15 (Aug 2008), 4571–4580.

[16] Hargrove, P., and Duell, J. Berkeley lab checkpoint/restart (blcr) for linux clusters. In *Journal of Physics: Conference Series* (2006), vol. 46, Institute of Physics Publishing, pp. 494–499.

[17] Jiang, X., and Xu, D. Violin: Virtual internetworking on overlay infrastructure. In *Proc. of the 2nd Intl. Symp. on Parallel and Distributed Processing and Applications* (2004), Springer.

[18] Keahey, K., Figueiredo, R., Fortes, J., Freeman, T., and Tsugawa, M. Science clouds: Early experiences in cloud computing for scientific applications. *Cloud Computing and Applications 2008* (2008).

[19] Klimeck, G., McLennan, M., Brophy, S., Adams III, G., and Lundstrom, M. nanohub. org: Advancing education and research in nanotechnology.

Computing in Science & Engineering 10, 5 (2008), 17–23.

[20] Kondo, D., Javadi, B., Malecot, P., Cappello, F., and Anderson, D. Cost-benefit analysis of cloud computing versus desktop grids. In *18th International Heterogeneity in Computing Workshop (hcw09)* (2009).

[21] Litzkow, M., Livny, M., and Mutka, M. Condor - a hunter of idle workstations. In *Proceedings of the 8th International Conference of Distributed Computing Systems* (June 1988). Available from: `http://www.cs.wisc.edu/condor/`.

[22] Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., and Zagorodnov, D. The eucalyptus open-source cloud-computing system. In *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid-Volume 00* (2009), IEEE Computer Society, pp. 124–131.

[23] Olson, A. L., Cai, S., Herdendorf, T. J., Miziorko, H. M., and Sem, D. S. Nmr dynamics investigation of ligand-induced changes of main and side-chain arginine n-h's in human phosphomevalonate kinase. *J Am Chem Soc 132*, 7 (Feb 2010), 2102–2103.

[24] OpenVPN Technologies, Inc. OpenVPN. Available from: `http://openvpn.net/`.

[25] Oracle. Virtualbox. Available from: `http://www.virtualbox.org/`.

[26] Quintas, D. Boinc in virtualbox. Available from: `http://boinc.berkeley.edu/trac/wiki/VirtualBox`.

[27] Ruth, P., Jiang, X., Xu, D., and Goasguen, S. Virtual distributed environments in a shared infrastructure. *IEEE Computer 38*, 5 (2005), 63–69.

[28] Ruth, P., McGachey, P., and Xu, D. Viocluster: Virtualization for dynamic computational domains. In *Proceedings of the IEEE International Conference on Cluster Computing (Cluster'05)* (2005).

[29] Ruth, P., Rhee, J., Xu, D., Kennell, R., and Goasguen, S. Autonomic live adaptation of virtual computational environments in a multi-domain infrastructure. In *IEEE International Conference on Autonomic Computing* (2006), vol. 2006.

[30] Santosa, M., and Schaefer, A. Build a heterogeneous cluster with colinux and openmosix. Available from: `http://www.ibm.com/developerworks/linux/library/l-colinux/`.

[31] Schmidt, T. G. Optimal "image-based" weighting for energy-resolved ct. *Med Phys 36*, 7 (Jul 2009), 3018–3027.

[32] Sumanth, J. Running condor in a virtual environment with colinux. Available from: `http://www.cs.wisc.edu/condor/CondorWeek2006/presentations/sumanth_condor_colinux.ppt`.

[33] Wolinsky, D., and Figueiredo, R. Simplifying Resource Sharing in Voluntary Grid Computing with the Grid Appliance. In *Proceedings Desktop Grids and Volunteer Computing Systems (PCGrid)* (April 2008), IPDPS.

[34] Wolinsky, D. I., Agrawal, A., Boykin, P. O., Davis, J. R., Ganguly, A., Paramygin, V., Sheng, Y. P., and Figueiredo, R. J. On the design of virtual machine sandboxes for distributed computing in wide-area overlays of virtual workstations. In *VTDC '06: Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing* (Washington, DC, USA, 2006), IEEE Computer Society, p. 8.