Journal of Physics: Conference Series **219** (2010) 042028 doi:10.1088/1742-6596/219/4/042028

Virtual Machine Logbook - Enabling Virtualization for ATLAS

Yushu Yao[1], Paolo Calafiura[1], Julien Poffet[2], Andrea Cavalli[2], Charles Leggett[1], and Bapst Frédéric[2]

[1] Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA, USA, 94720
[2] Ecole d'Ingénieurs et d'Architectes de Fribourg, Switzerland

Abstract. ATLAS software has been developed mostly on CERN linux cluster lxplus or on similar facilities at the experiment Tier 1 centers. The fast rise of virtualization technology has the potential to change this model, turning every laptop or desktop into an ATLAS analysis platform. In the context of the CernVM project we are developing a suite of tools and CernVM plug-in extensions to promote the use of virtualization for ATLAS analysis and software development.

The Virtual Machine Logbook (VML), in particular, is an application to organize work of physicists on multiple projects, logging their progress, and speeding up "context switches" from one project to another. An important feature of VML is the ability to share with a single "click" the status of a given project with other colleagues. VML builds upon the save and restore capabilities of mainstream virtualization software like VMware, and provides a technology-independent client interface to them. A lot of emphasis in the design and implementation has gone into optimizing the save and restore process to makepractical to store many VML entries on a typical laptop disk or to share a VML entry over the network.

At the same time, taking advantage of CernVM's plugin capabilities, we are extending the CernVM platform to help increase the usability of ATLAS software. For example, we added the ability to start the ATLAS event display on any computer running CernVM simply by clicking a button in a web browser.

We want to integrate seamlessly VML with CernVM unique file system design to distribute efficiently ATLAS software on every physicist computer. The CernVM File System (CVMFS) download files on-demand via HTTP, and cache it locally for future use. This reduces by one order of magnitude the download sizes, making practical for a developer to work with multiple software releases on a virtual machine.

1. Introduction

The offline software of the ATLAS experiment[1] is a complex system. It consists of millions lines of code from hundreds of developers around the world.

Currently there are two major approaches to do development for ATLAS software:

- Remote login to interactive clusters and access ATLAS software via AFS. For example, users can use the LXPLUS[2] interactive cluster. However, its performance depends on the speed of network and the available resource of the cluster. It is hard to run long time test jobs or utilize X11 based applications like the ATLAS event display.
- Downloading ATLAS software as distribution kits into a local Linux Computer. The ATLAS software is distributed as a distribution kits which normally have sizes around 7-8GB. The

release cycle of the ATLAS software is roughly one month, downloading a kit for each release update is a waste of network resource and put extra workload onto institutions' system managers.

CernVM[3] is a Virtual Software Appliance. A virtual appliance can run on any hardware provided the necessary virtualization software (also called hypervisor). CernVM aims to provide a complete and portable environment for developing and running LHC data analysis on any end-user computer (laptop, desktop) and on the Grid, independently of Operating System, software and hardware platform (Linux, Windows, MacOS).

In CernVM, the experiment software is delivered to the appliance by a network file system (CVMFS). CVMFS provides a read-only file system on Linux based which only fetch a file when it is needed. For day to day development or analysis work, only a fraction of the distribution kit is required (e.g. only 800MB out of the 8GB kit is needed to run a full ATLAS reconstruction job). This approach significantly reduces the amount of data to be transfered over the network from order of multiple gigabyte to the order of several hundred megabyte. Also, since the fetching of a file is automatically handled by CVMFS.

By adapting CernVM to run ATLAS software, we can minimize the work for installation and updating ATLAS software and to minimize the number of platforms (compiler-OS combinations) on which experiment software needs to be supported and tested, thus reducing the cost of software maintenance.

Virtual Machine Logbook is a open source tool developed to let users easily organize, switch between projects and keep track of the project history.

This paper divides into two parts, part one introduce the Virtual Machine Logbook, and part two summarizes how we customized CernVM to meet the needs of ATLAS software.

2. Virtual Machine Logbook

Some of the virtualization software (e.g. VMWare) let users save their work state by pausing the VM or VM snapshots. However, there is no existing open source software that let the user save multiple states of multiple projects in a flexible way, nor to easily share work states between different users.

The Virtual Machine Logbook (VML) is an application to organize physicistsÕ work on multiple projects, logging their progress, and speeding up "context switches" from one project to another. It is a versioning system (like CVS) adapted for virtual machines. An important feature of VML is the ability to share with a single "click" the status of a given project with other colleagues. VML builds upon the save and restore capabilities of mainstream virtualization software like VMware, and provides a technology-independent client interface to them. A lot of emphasis in the design and implementation has gone into optimizing the save and restore process to makepractical to store many VML entries on a typical laptop disk or to share a VML entry over the network.

2.1. Design of VML

Figure.1 shows the general architecture of VML with the main components - working areas, entries, and repositories.

2.1.1. Working Area A working area is a folder on the host machine which contains the files necessary to start the virtual machine. For a user, each working area corresponds one project that he is working on. VML can build an entry based on the state of the current working area, or restore the state of a working area from a VML entry. VML can manage multiple working areas on the same host in case the user need to work on multiple projects.

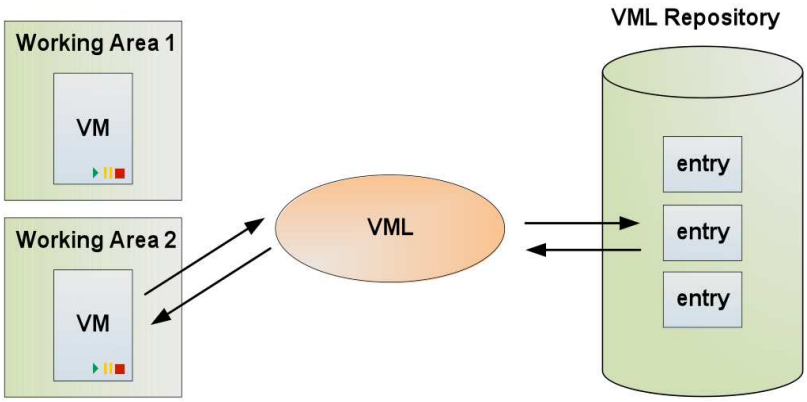

Figure 1. Main Components of VML

2.1.2. Entry An entry contains the descriptions of virtual machines at a given state. An entry should contain enough information to rebuild a VM with the same state, but should be as small as possible for the sharing of work to be possible.

2.1.3. Repository A repository is a pool of entries that can be accessed by VML. For the use case of running VML on a local machine, a repository is a directory on the local file system which contains the entries and some metadata. However, the idea of a repository can be generalized to a repository service, which runs on a specific computer system, and allows a VML client to add, remove or read out entries.

2.2. Technology Used for VML Development
VML is developed primarily with the Python[4] programing language. It utilizes XML to store description of work areas, repositories or entries.

Currently the hypervisor-related functionalities (e.g. start, stop, suspend or snapshot) are based on a python implementation of the VMware-VIX[5] program interface. This enables VML to operate on any system that is equipped with the VMware Free Server Edition or higher.

We are also investigating possibilities to port VML to other hypervisors like XEN[8] or KVM[9]. For example, we can utilize the Libvirt[6] library to perform the hypervisor-related functionalities on multiple hypervisors, or use OpenOVF[7] as a unified format to store or transport virtual machines.

2.3. Saving VM State
There are two fundamental processes VML performs:

- Save a VM to an Entry: Based on the state of the VM in the current working area, VML creates a new entry and push it into the repository;
- Build a VM from an Entry: From and existing entry in the repository, VML builds a virtual machine inside a working area.

We investigated two approaches to save a VM's state into a VML entry.

2.3.1. Based on VMWare's Snapshot Functionality Some hypervisors (e.g. VMware) can make incremental snapshots of a VM. These incremental snapshots can be saved into a VML entry.

However, snapshot function is not available on many popular hypervisors (e.g. Xen, KVM) which limits the portability of VML.

2.3.2. Comparing to a Base CernVM A CernVM user normally only have constrained activity when working on a development project. Only a limited number of files are changed inside the VM's filesystem. In other words, if a user started with a specific version of CernVM, only a small number of changes are made to the filesystem in order to develop the project. This difference can be saved into a VML entry. By sharing this entry, any user can rebuild a VM at the same state by combining the difference with the same version of CernVM.

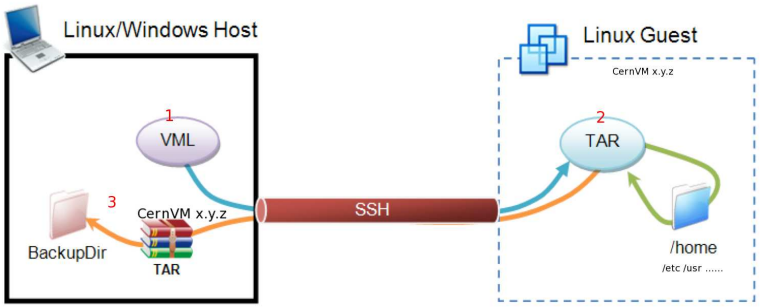

Figure 2. Saving the state of CernVM via TAR and SSH.

Figure.2 illustrates an approach to save a CernVM's state by running TAR over SSH tunnel. Upon receiving the command to save the VM's state, VML first open an SSH tunnel between the host and the guest (step 1), then execute a TAR command inside the VM as a privileged user. The TAR process (step 2) scans through a pre-defined list of directories (normally /home, but can also include system directories like /etc or /usr), save all files that are different from the base CernVM (version x.y.z). The output of the TAR process is streamed back to the host computer via the SSH tunnel. VML then save the TAR output together with the version of the CernVM (x.y.z) into a VML entry.

2.4. Sharing Work with VML

In the current development model of LHC especially ATLAS, if two users would like to share work in a specific project, the most frequently used way is to exchange the project source code with necessary instructions. However, due to the difference between the software environment, a project that compiles and runs at place A does not necessarily compile or run at place B.

The effort to transplant the project from place A to place B can be eliminated by exchanging a virtual machine, which provides an software environment that will not change during the exchange process. One problem with this approach is, however, a VM is normally at the size of hundreds of MB, and this does not include the ATLAS software distributions which are 8GB each. Exchanging gigabyte of data for a VM is time consuming and is a waste of network resources since most of those data are redundant and can be obtained through other means (e.g. the ATLAS software).

With the help of CernVM's file system CVMFS, the ATLAS software can be distributed to each CernVM client separately so that it does not need to be exchanged between users. The size of data to be exchanged for the VM is reduced from 8GB to order of several hundred MB.

Furthermore, VML can construct a VML entry based on the difference between an existing VM with its base CernVM of a specific version. Depending the size of the project, this will further reduce the size of the data to exchange during the share.

Figure.3 shows the process of sharing a work state with VML.

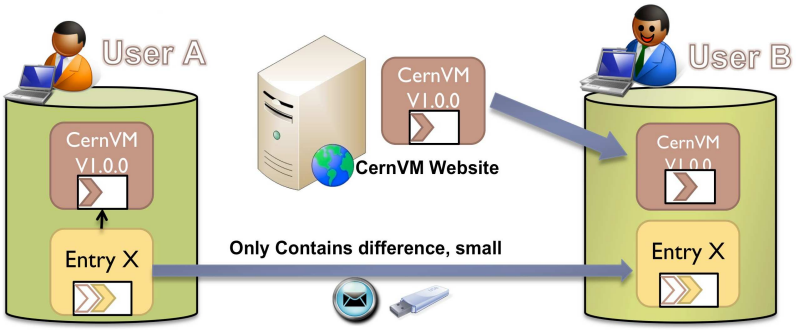

Figure 3. Sharing Work with VML.

(i) VML produce a VML entry and put it in its local repository. The entry only contains difference of the current VM from the base CernVM;

(ii) Since the entry is small, the user who would like to share can easily transfer the entry (as a file) to a second user (via email, FTP, or a central VML repository);

(iii) The second user receives the entry and ask VML to build a VM from the entry;

(iv) Based on the version of CernVM of the entry, VML will first obtain the proper version of CernVM. Then VML patch the base CernVM with the data inside the entry and rebuild the same state as the first user.

3. CernVM and ATLAS extenstion

Users of CernVM are categorized into two classes: developer and data analyzer. We extended CernVM to to meet the needs of both kinds of users.

3.1. Setup and Testing of Proxy Servers

CernVM distributes Software releases from the central HTTP server located at CERN. To speed up the file access of users out of CERN (e.g. in the US or Canada), we setup proxy servers. The proxy servers are SQUID[10] servers with cache size 10-100GB. Our tests shows that with a proxy server in the local area network, the speed of compilation of ATLAS packages has a factor of five (5) increase.

By the time of this paper, we have setup three (3) proxy servers in the following places: Lawrence Berkeley National Lab (LBL) in California, University of Washington in Washington and TRIUMF in British Columbia. All of these servers have the CERN central proxy server as their parent node. When a CernVM starts up, it obtains a list of proxy servers that have the shortest geographical distance. E.g. a CernVM in Canada is likely to access the file repository via the TRIUMF server, while a CernVM at Berkeley will access the file repository via the LBL proxy server.

We have put online a step-by-step instruction[11] to setup a proxy server, and we are seeking volunteers to setup more proxy servers to speed up the distribution of ATLAS software to CernVM users around the world.

3.2. Plugins

CernVM includes the "rPath Appliance Platform Agent (rAPA)"[12] software package which allows managing the Virtual Machine through a web interface or Remote Procedure Calls (XML-RPC)[13]. rAPA provides an easily-extendable framework which enables the developers to add functionalities to the web interface to meet their specific needs. We have developed several plugins to meet specific requirements of ATLAS users.

3.2.1. Cache Preload The CVMFS fetch files only when they are accessed via the open() system call of UNIX. Upon opening an un-cached file, CVMFS will connect to the remote HTTP server (or PROXY server, as specified) to obtain the file. To access a locally cached file, no internet connection is needed. If a user caches a sufficient set of files in CVMFS, the user is able to continue working without a network connection.

3.2.2. Nightly Builds ATLAS software is a large scale development project. It has a Nightly COntrol System (NICOS)[14] that manages the multi-platform nightly builds based on the recent versions of software packages. It helps to identify possible problems, and makes the developers spread over different institutions and countries do able to work with the latest software immediately.

The nightly builds are mounted as different CVMFS directories and the files are served from separate file serves. Figure.4 shows the screen shot of the release manager plugin.

Figure 4. Screen shot of the release manager plugin to CernVM

3.2.3. VNC-X windows Many development software (e.g. xEmacs or Eclipse) or analysis software (e.g. ROOT) require X-Windows to display the graphical interface. One way to use these applications is to execute them via the X11 forwarding functionality of SSH. However, the user have to have 1) an SSH client and 2) a X-Windows server running on the user's workstation. In case of a non-Linux workstation (e.g. Windows on a Laptop), these in many cases are not readily availably and might not be easy to obtain due to the need of a privileged user to install these software.

To solve this problem, we developed a plugin to CernVM which can start multiple Virtual Network Computing (VNC) servers inside the CernVM. VNC is a graphical desktop sharing system that can be used to remotely control another computer. It transmits the keyboard and mouse events from one computer to another, while at the same time receiving the graphical screen updates, over a network. A CernVM user can connect to the VNC server by either using a java-enabled web-browser or a VNC client that needs no privileged user to run or install.

Figure.5 shows a screen shot of the Xvnc manager plugin to CernVM.

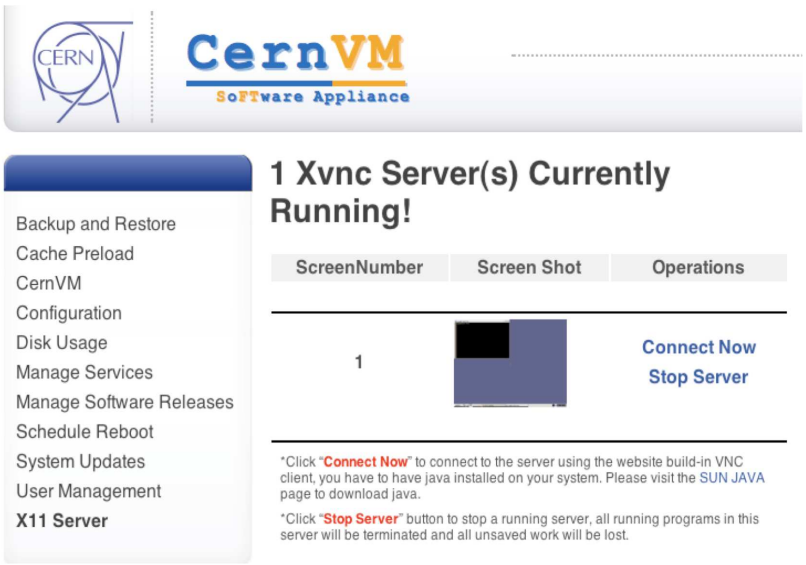

Figure 5. Screen shot of the Xvnc manager plugin to CernVM. It shows one running VNC server inside the VM and it's screen snapshot at the time.

3.2.4. One Click Launching ATLAS Live Event Display ATLAS have a sophisticated event display system ATLAS Virtual Point 1 (VP1), which provides a modern 3D enabled display for point 1 visualizing the sub-detectors, tracks and hits. It aims to provide a useful tool for the understanding of physics events and help with debugging of software and analysis. Figure.6 shows the VP1 visualizing the ATLAS detector.

4. Conclusion

In this work we extended CernVM to be able to run ATLAS softwares. This enables ATLAS users to do development or data analysis on any computer regardless of software or hardware setup. The Virtual Machine Logbook provides an efficient way for users to save their work states on CernVM, and to share work state with each other. It could potentially change the ways how users in a large collaboration do their work individually or cooperating with each other, while at the same time reduce the maintaince and development cost of the ATLAS software.

Acknowledgment

Thanks to Asoka De Silva for integrating and testing GRID tools in CernVM, and for setting up the first proxy server in Canada. Thanks to Gordon Watts for setting up the first proxy server in the US, and for useful discussions. Thanks to the CernVM team lead by Predrag Buncic for

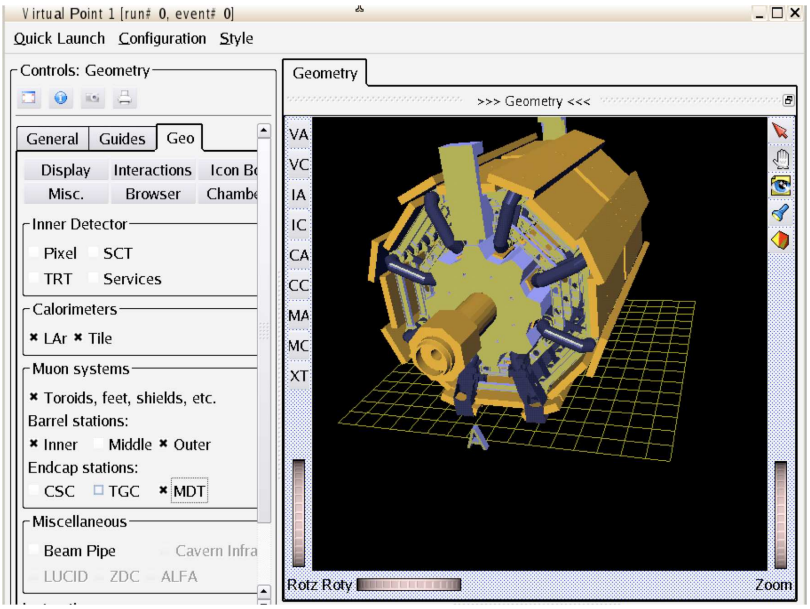

Figure 6. Running the ATLAS Event Display via Xvnc inside CernVM

their cooperation in enabling CernVM to run ATLAS software. Thanks to Birger Koblitz for providing web space to serve ATLAS nightly builds to CernVM clients.

References

[1] http://atlas.web.cern.ch/Atlas/index.html
[2] http://plus.web.cern.ch/plus/
[3] http://cernvm.cern.ch/cernvm/
[4] http://www.python.org/
[5] http://www.vmware.com/support/developer/vix-api/
[6] http://libvirt.org/
[7] http://open-ovf.wiki.sourceforge.net/
[8] http://www.xen.org/
[9] http://www.linux-kvm.org/page/Main_Page
[10] http://www.squid-cache.org/
[11] https://twiki.cern.ch/twiki/bin/view/Atlas/CernVMGuidePROXY
[12] http://wiki.rpath.com/wiki/rPath_Appliance_Platform_Agent
[13] http://www.xmlrpc.com/
[14] https://twiki.cern.ch/twiki/bin/view/Atlas/NIghtlyCOntrolSystem