PROCEEDINGS OF SCIENCE

CernVM - a virtual appliance for LHC applications

Predrag Buncic[1]
CERN
CH-1211, Geneve 23, Switzerland
E-mail: predrag.buncic@cern.ch

Carlos Aguado Sanchez
CERN
CH-1211, Geneve 23, Switzerland
E-mail: carlos.aguado.sanchez@cern.ch

Jakob Blomer
CERN
CH-1211, Geneve 23, Switzerland
E-mail: jakob.bloomer@cern.ch

Leandro Franco
CERN
CH-1211, Geneve 23, Switzerland
E-mail: lfranco@cern.ch

Steffen Klemer
CERN
CH-1211, Geneve 23, Switzerland
E-mail: steffen.klemer@cern.ch

Pere Mato
CERN
CH-1211, Geneve 23, Switzerland
E-mail: pere.mato@cern.ch

[1] Speaker

© Copyright owned by the author(s) under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike Licence.

PoS(ACAT08)012

PROCEEDINGS OF SCIENCE

CernVM[1] is a Virtual Software Appliance capable of running physics applications from the LHC experiments at CERN. Such a virtual appliance 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). The goal is to remove a need for the installation of the experiment 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. The experiment software is delivered to the appliance just in time by means of a network file system (CVMFS) specifically designed for efficient software distribution and installation. The experiment application software and its dependencies are built independently from CernVM Virtual Machine. The procedures for building, installing and validating software release remains in the hands and under the responsibility of each user community. We provide the tools to synchronize pre-built and configured experiment software releases with our central distribution point. In this paper we present current state of CernVM project and plans for further developments.

XII Advanced Computing and Analysis Techniques in Physics Research
Erice, Italy
3-7 November, 2008

© Copyright owned by the author(s) under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike Licence.

PoS(ACAT08)012

1. Introduction

1.1 LHC software and supported platforms

After more than fifteen years of preparation the LHC experiments are now finally ready to start data taking. Along with detector hardware, by now installed in the experimental pits, the software frameworks are being tested and validated. Being developed for an almost equally long time, the software frameworks have grown to levels of complexity rivalling only the complexity of detectors they aim to describe. The result is a very large code base that has to be maintained on all supported platforms and validated for every release. In the case of LHC experiments we observe a typical release to contain between 2 and 8 GB of build products. Typically, such releases are made every two months with couple of patch releases made in between.

At present, Linux remains the predominant Operating System (OS) for scientific and high throughput computing in High Energy Physics (HEP). Since LHC experiments are bound to utilize a large pool of computing resources by means of Grid[2], an additional constraint imposed by the middleware is to use a specific version of Linux OS which is often a quite outdated one.

Old versions of Linux are usually difficult to install and maintain, therefore some end-users (physicists working on LHC data analysis) choose to run on their desktops and laptops one of the newer and more agile distributions that provide better user interface and support for recent hardware. For the same reasons others choose to stay with default Windows installation or switch to MacOS platform. In all these cases, the net result is that the computing platform (basic OS + compiler), which could be ideally used for software development, analysis and interaction with Grid resources, is different from one on the Grid, rendering the user desktop or laptop useless for anything but e-mail and Web browsing.

1.2 Multicore and Virtualization

Virtualization is a technique that enables the application view of computing resources to be separated from the underlying infrastructure. The virtualization software layer (also referred to as a *hypervisor*) enables the host platform to execute multiple Virtual Machines (VMs) where each VM can run either its own copy of a different operating system, or multiple versions of the same operating system.

The recent development in CPU manufacturing brought multicore CPU architecture [26][27] as well as support for hardware CPU virtualization into commodity computing and triggered a flood of new software solutions allowing for Operating System and server virtualization. While these software solutions used to be costly options in the past, today there are several free or Open Source projects of good quality (such as Xen[4] and KVM[3] on Linux, Sun VirtualBox[5] on Linux, MacOS and Windows) providing very effective hypervisor solutions.

PoS(ACAT08)012

Indeed, the whole software industry seem to be recently embracing virtualization as enabling technology behind various aspects of *Cloud Computing* paradigm ranging from Amazon EC2[7] style utility computing, where end-user is given full control of hosted virtual machine(s), to Software as a Service (SaaS) model of Google Applications[8], where user only retains the control of his application which is hosted, together with data, on service provider infrastructure. Using virtualization technology the service providers can increase security by isolating different users from physical servers and reduce downtime required for planned or unplanned physical system maintenance.
With big companies rushing to build the new data centres around the world in support of their Cloud infrastructure and given that CPU manufacturers are likely to continue fulfilling Moore's Law prophecy by adding multi and many cores on a single chip over several future CPU generations, it looks like the virtualization, seen by many as a key technology that enables server consolidation and better resource utilization in such modern data centres, is here to stay with us for the foreseeable future.

1.3 Virtual Software Appliances

Virtualization can also provide a way to support legacy systems on new hardware and simplify deployment of complex applications. The idea of packaging application software together with a minimal operating system so that it can be easily deployed on virtualized infrastructure is particularly interesting. By adding to such mix an extra management software providing interfaces to simplify configuration and management tasks we get a special kind of Virtual Machine that is called a *Virtual Software Appliance*.

2. CernVM Project

2.1 CernVM Software Appliance

In order to allow end-users to effectively use their desktops and laptops as for analysis and Grid interface, we have created CernVM, a lightweight Virtual Machine intended to run on any OS platform and provide consistent and effortless installation of experiment software as well as interface to Grid.

CernVM Virtual Software Appliance is built using rBuilder tool from rPath[9]. This tool provides a very convenient way to describe, develop, build and distribute such appliances. It supports all image formats required by the major hypervisors and can build them starting from a common package *group recipe*. Such a group recipe is in turn a collection of *package* and *group* recipes where each package is described in a simple python script and built using *conary* package manager. Unlike other popular package managers, conary is capable of analysing the build products and automatically classifying them into sub-packages and detecting all package build and runtime dependencies. The resulting virtual appliance image contains the application as well as just enough of Operating System required to run it. This approach has a clear

PoS(ACAT08)012

advantage since the fewer components we put in the image, the less needs to be maintained and kept up to date.

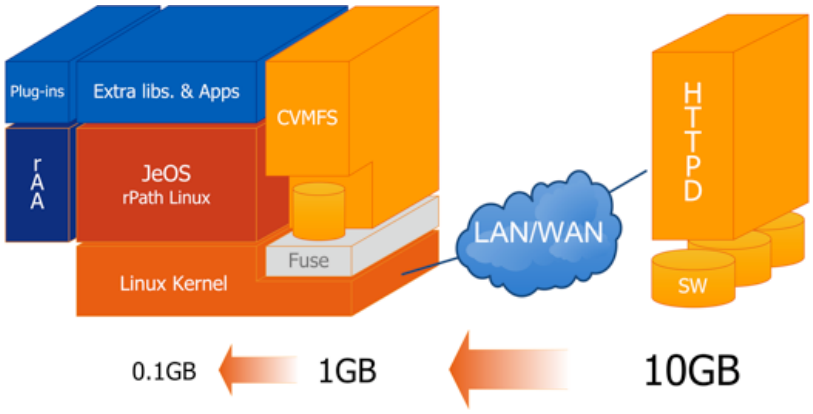

***Figure 1**: Basic building blocks of CernVM Virtual Software Appliance*

Since the underlying rPath Linux is binary compatible with SLC4[10] Linux this allows us to reuse the software packages already built and tested in SLC4 environment and simply run them unmodified under CernVM. In this scenario, the client downloads only the necessary binaries and libraries as they get referenced for the first time. By doing that, the amount of software that has to be downloaded in order to run the typical experiment tasks in the Virtual Machine is reduced by an order of magnitude.

The appliance runs rPath Appliance Agent software that provides two user interfaces, for configuration purposes – a Web based one for easy interaction with the end-user and a XML-RPC one to allow interaction with an instance of CernVM using scripting language interface to facilitate bulk or repetitive operations. The basic components of CernVM appliance are shown in **Figure 1**.

Thanks to the rBuilder software from rPath, we are able to create CernVM Virtual Software Appliance that can run on all hardware (x86 and x86_64) and OS (Windows, Mac, Linux) platforms and supports a spectrum of virtualization backends (VMware, Xen, KVM, VirtualBox). This appliance, which fits in a compressed file of around 130 MB, contains operating system that is sufficient to bootstrap the entire experiment software stack from a dedicated network file system optimized for software delivery (CernVM File System or CVMFS). This approach allows us to develop and maintain only one Virtual Software Appliance as a common platform for all four LHC experiments and allows experiments to customize and feed such appliance with their software releases at runtime and in an efficient way. At the same time our end-users, physicists running the analysis frameworks of LHC experiments, get an easy to install and portable environment, which is always kept up to date.

PoS(ACAT08)012

2.2 CernVM File System

One of the biggest problems for running a large scale data processing in HEP environment is the distribution of code to many thousands of worker nodes around the world[11]. This can be solved by introducing a dedicated file system[12]. A Global Read-Only Web file system (GROW-FS) has been designed to make a directory tree stored on a web server accessible over the wide area network, with aggressive caching and end-to-end integrity checks. It is designed to maximize the cacheability of all components of the file system. Both files and file metadata can be cached on local disk without remote consistency checks, as well as cached on shared proxy servers, allowing the file system to scale to a very large number of clients.

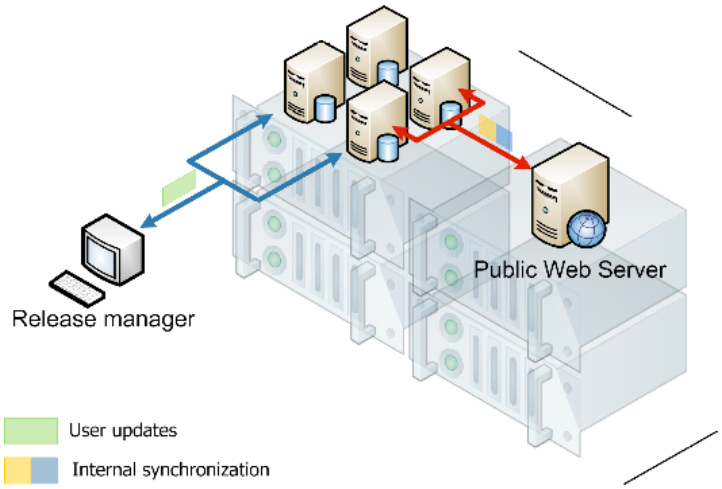

***Figure 2**: Software releases are first uploaded by the experiment's release manager to a dedicated virtual machine, tested and then published on a Web server*

To create a GROW-FS file system, one has to create a catalogue of all files in a directory tree and export it via a web server. This is done by a script that creates the file .catalogue that contains a complete directory listing and checksum of all data. Upon first accessing the file system remotely, GROW-FS loads the catalogue into a tree form in memory. All metadata requests and directory lookups are handled using this data structure. The integrity of the catalogue is ensured by fetching its checksum using HTTPS protocol. If the master checksum and the directory listing are inconsistent, they are reloaded in the same way as files.

The original GROW-FS file system was embedded in Parrot[13] framework and meant to run completely in user space. Since Parrot intercepts the system calls using *strace* interface this introduces considerable performance penalty and occasional problems with certain applications. Since in the Virtual Machine we have full access to the kernel and kernel modules, we have constructed CernVM File System (CVMFS) from Parrot and GROW-FS code base and adapted them to run as a FUSE[14] kernel module adding some new features to improve scalability, performance and simplify deployment in the case when software compiled for different platforms is installed on CVMFS. The principal new features in CVMFS compared to GROW-FS are:

PoS(ACAT08)012

- Using FUSE kernel module allowing for in-kernel caching of file attributes.
- Capability to work in offline mode providing that all required files are cached.
- Possibility to use multiple (hierarchy of) file catalogues on the server side.
- Transparent file compression under given size threshold.
- Dynamical expansion of environment variables embedded in symbolic links.

The procedures for building, installing and validating software release remains in the hand of, and the responsibility of, each user community. We provide the tools to synchronize pre-built and configured experiment software releases with our central distribution point (see **Figure 2).** At present all four LHC experiments are supported, and distribute about 100 GB of their software using this method.

2.3 Supporting Infrastructure

In order to provide scalable central services in support of CVMFS, we have build an infrastructure that can be expanded by adding more front end servers (reverse proxies) operating in DNS load balancing configuration and forwarding the request based on URL content to the backend servers that run again as Virtual Machine on top of the physical servers. Another pair of the servers, running in a highly available configuration, is providing a shared file system for the rest of the infrastructure and allowing for live migration of Virtual Machines between physical nodes.

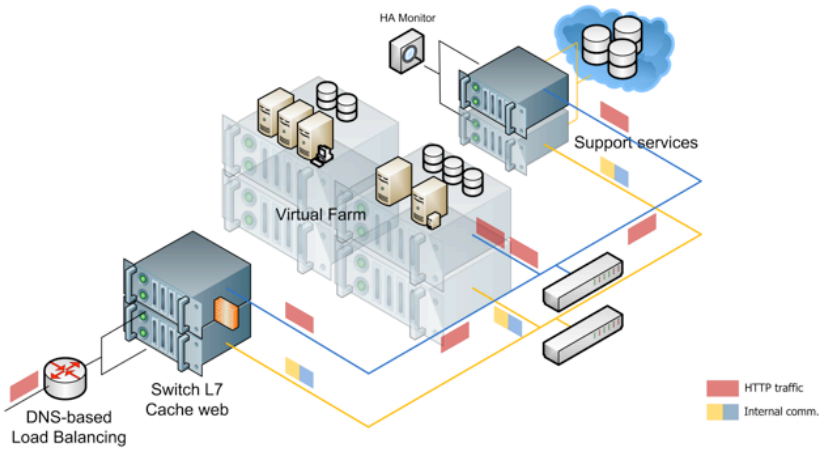

***Figure 3**: Supporting infrastructure for CernVM central services*

The CernVM infrastructure consists of three building blocks: computing, storage and networking services. Computing Services are used to run all VM and CVMFS services. Those services are deployed as virtual machines running on a pool of VMware ESX servers in such a way that CPU time and system memory are aggregated into a common space and can be

PoS(ACAT08)012

allocated for single or multiple virtual machines. This approach enables better resource utilization and simplifies the management for High Availability (HA) and scalability.

The Storage System provides reliable and resilient storage capabilities to the whole infrastructure. The solution is conceived as a multi-tiered storage solution where two multi-headed servers (in HA configuration) provide the bulk storage. It has additional features such as the advance storage management (dynamic allocation, dynamic growth, snapshots and continuous data replication) and capability to implement consistent backup policies to some tertiary MSS.

The CernVM infrastructure makes use of two separate network domains and three different IP segments. Public service provisioning is decoupled from internal service requirements by using physically different channels with different QoS with extensive use of IEEE 802.3ad and 9K Jumbo Frames to improve data I/O and VM live migration.

2.4 Current status

The first public release[15] of CernVM came in October 2008 after a sequence of regular monthly development releases which were given to test users in the LHC experiments over a six month period during which they gave us valuable feedback on general performance as well as their specific requirements which helped us shape CernVM Software Appliance to meet the needs of all four LHC experiments. This release is suitable for its intended use as a development environment, as an interactive data analysis environment and as an interface to Grid resources. In addition, on the ATLAS example we have shown that CernVM can host PANDA[16] jobs while in ALICE case we have demonstrated that it can be used to re-create an entire AliEn[17] site and run all required sites services as well as execute user jobs. As an example of interactive use in a full X environment, **Figure 4** shows CernVM running the LHCb event display, various analysis tasks and the Grid submission tool GANGA [18].

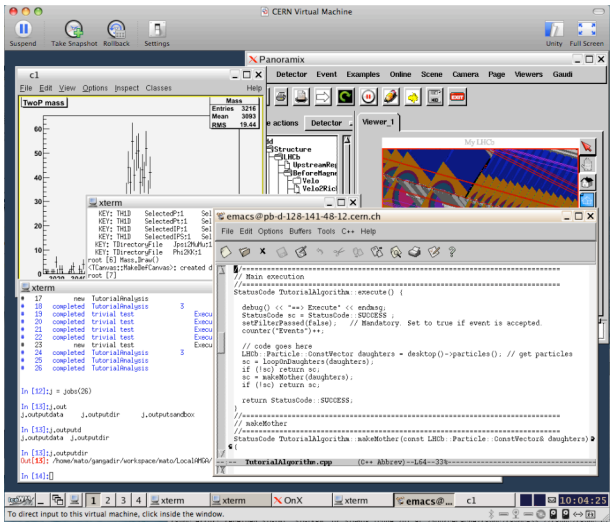

***Figure 4**: CernVM running LHCb event display and analysis (VMware Fusion on MacOS)*

PoS(ACAT08)012

The release is available in three image formats (VMware virtual disk, compressed file system image and hard disk image) compatible with the most popular hypervisors currently on the market (VMware, Xen, Parallels, VirtualBox, KVM). There were more than 800 downloads of first release during past four months with VMware topping the list of most popular downloads (see **Figure 5**). Once downloaded, the appliance can be kept up to date using internal self-update mechanism and normally it is not necessary to download a new image when updates are released.

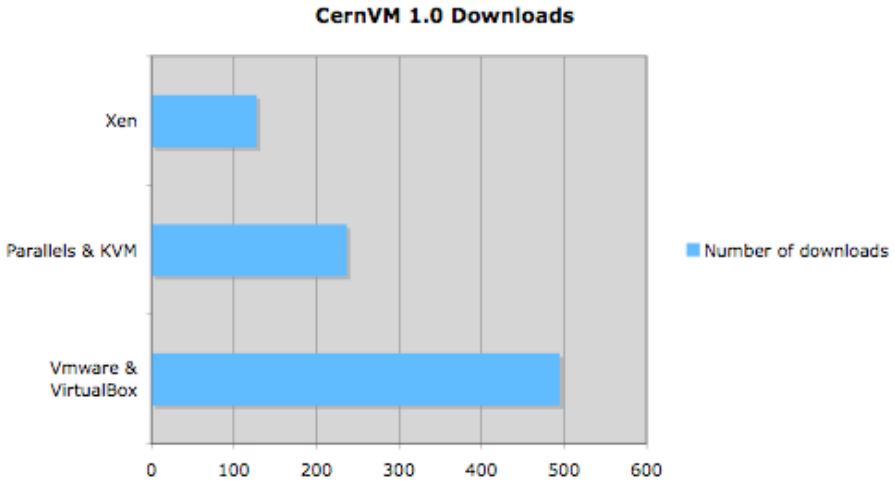

Figure 5: *Total number of downloads of CernVM 1.0 release (864) per virtual machine image type*

3. Future work

There are a couple of areas requiring further work. One is to improve the scalability and remove any single point of failure for the CVMFS server side by using Content Distribution Networks. The other area of work is to extend the initial scope of CernVM to include an environment for end-user job hosting.

3.1 Using Content Distribution Networks with CernVM

A hierarchy of cache servers (**Figure 6**) allows the software distribution network to be organized as a fine-grained distribution tree, having the CernVMs as leaves. Proxy servers may serve CernVMs as well as underlying proxy servers. Each proxy server stores the union set of required data of its subtree. By adding and removing proxy servers, the tree can be adapted to respond to required load.

This approach requires additional manual effort to setup and maintain the proxy hierarchy. Also, losing one of the proxy servers cuts the distribution tree into two unconnected components. At present, such a deployment model is fully supported and CernVM will always use the nearest proxy server as defined by the configuration service at startup time. In spite of

PoS(ACAT08)012

reasonable flexibility, this approach is still static in nature and clearly has a single point of failure.

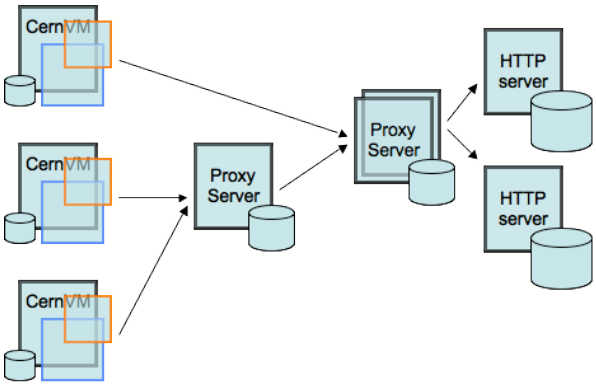

Figure 6: *In the current software version, CernVM clients fetch the data files from central repository via the hierarchy of proxy servers as specified by the configuration file obtained from the configuration server at boot time.*

To improve resilience of the system and remove single point of failure, one can create a couple of central service replicas and place them at strategic locations around the world (**Figure 7**). Such a set of repository mirror servers partition the software distribution network into autonomous areas. The areas are, in best-case, equally sized in terms of number of associated CernVM clients and network distance, thereby equally distributing the load. Moreover, by selecting a close mirror by default, latency is reduced, and by enabling clients to connect to arbitrary mirrors, reliability is improved.

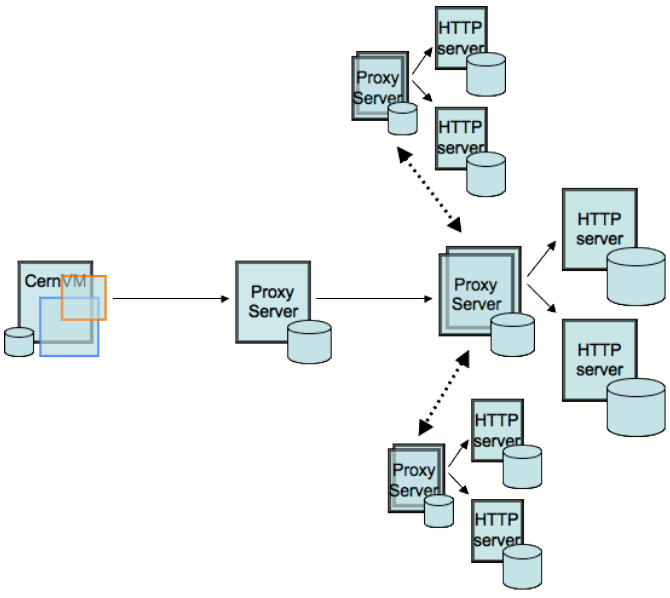

Figure 7: *It is possible to remove a single point of failure and reduce latency on file access by using traditional master-slave configuration with slaves deployed at strategic locations around the world.*

PoS(ACAT08)012

While again possible, this scenario requires extra effort to setup and maintain replicas always synchronized between themselves. Fortunately, we can also use existing commercial infrastructure like Amazon's CloudFront[20] service, to achieve the same functionality. If we mirror our repository to Amazon Simple Storage Service (S3)[21] , CloudFront Service will assure that users access the content using some nearby mirror. This is done by assigning a canonical DNS name to an S3 resource, by which the client transparently connects to the closest mirror. Mirroring is done on demand and expired content will be deleted from mirror servers automatically. An alternative to CloudFront that works on similar principles is the Coral Content Distribution Network[28].

However, in the CernVM case, a single site running many CernVM instances may still run into a local bandwidth bottleneck if every CernVM is fetching data independently. Here it would be desirable to benefit from data already loaded to some instance of CernVMs running on the same local network.

To tackle that problem, we added a capability to CernVM to discover other CernVM instances on a local network using Avahi[29] system to facilitate service discovery. Since every CernVM instance keeps a local cache of files it has already downloaded, it can also share this cache with neighbours acting as an ad-hoc proxy. This way we achieve a self-organizing network on a LAN level in a manner similar to peer-to-peer distribution networks while retaining most of the existing CernVM components and accessing top level repository residing on content distributions networks such as CloudFront or Coral, possibly via a chain of additional proxy caches.

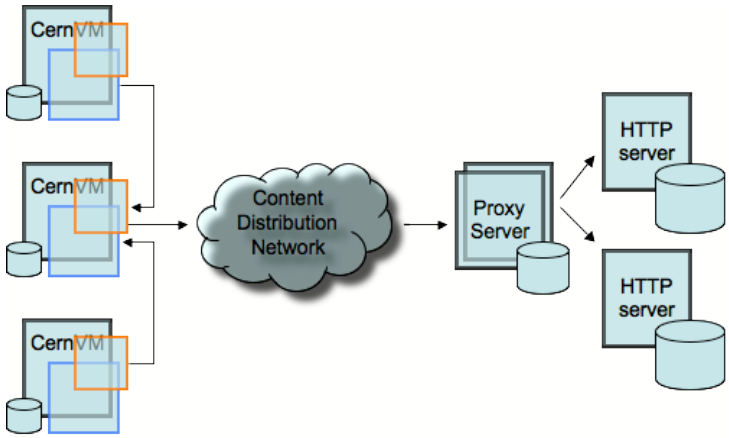

Figure 8: *Removal of single point of failure using content distribution network on WAN and self organizing P2P-like cache sharing on LAN*

This model has all of the ingredients necessary to assure scalability to support thousands of CernVM clients with the reliability expected from a file system, and performance matching that of traditional network file systems (such as NFS or AFS) on a local network.

PoS(ACAT08)012

3.2 CernVM as job hosting environment

Early feedback from users of CernVM indicates to the wish to use CernVM not only as a user interface to test and develop code and submit jobs, but also as a job hosting environment essentially equivalent to the traditional worker node in a batch cluster. While this requirement makes perfect sense from a user point of view, it is unlikely that we could meet it on a larger scale since all available resources are currently pooled together under control of Grid middleware. For this reason we have decided to look for an alternative way, based on BOINC[30] infrastructure for volunteer computing, to deploy large number of CernVM instances that could be used as a job hosting environment.

This requires a small modification of BOINC job wrapper to start a Virtual Machine in place of a real user job payload. Due to small initial image size (<130 MB) we hope that CernVM is still within limits that volunteers (home PC users) would accept to download in order to start contributing CPU cycles the project.

Once such an instance of CernVM is started, it runs an agent that joins a messaging network running on an adapter node that on the other side interfaces to the specific experiment workload management system controlling the flow of Grid jobs (AliEn in case of ALICE, Panda for ATLAS and DIRAC[31] for LHCb). This approach has the advantage that no modification to the user job is required and there are no changes in how the Grid handles jobs from the experiment point of view. In this way the resources made available by means of BOINC infrastructure would simply complement the resources already available to Grid users of a particular experiment, with the caveat that jobs dispatched to BOINC should require little input and produce modest output (which is the case for typical Monte Carlo jobs).

4. Conclusions

In this R&D activity we explored how virtualization technology can be used to provide a portable user environment and interface to Grid. We developed CernVM, Virtual Software Appliance that fulfils the basic needs of all four LHC experiments. This “thin” appliance contains only a basic OS that is required to bootstrap the LHC application software that is delivered to the appliance using CVMFS, a file system specially crafted for software delivery. Using this mechanism, CernVM once configured, does not need to be updated with each experiment software release since updates are automatically propagated. This results in an easy deployment scenario as only one image is required to support all experiments. The same is true if such Virtual Machines would be deployed to serve as job hosting environment on voluntary resources (like BOINC) or managed (like computing clusters and grids).

By leveraging the use of standard protocols (HTTP) for all communication, we can effectively use a proxy server infrastructure where it exists, or create self-organizing proxy networks in LAN environments to minimize the impact on a central software distribution point. Further, by using existing content delivery network infrastructure, public or commercial, we can remove a single point of failure and create a reliable and efficient file system for software delivery to the thousands of CernVM clients with performance close to or better than traditional

PoS(ACAT08)012

network file systems such as AFS or NFS and the additional bonus of capability to work in disconnected mode.

While use of virtualization technology definitively opens many new possibilities and offers some elegant solutions to difficult problems of development and deployment of HEP applications on distributed computing infrastructure such as Grid and emerging Clouds, it is clear that using another layer of indirection will always result in some, hopefully minimal and acceptable, performance penalty. In spite of the tempting prospect, this technology should not be used as a substitute for continuous effort to improve the software quality by porting it to new platforms and using different compilers. Any such improvements, and in particular those resulting in reducing the memory footprint will directly impact the performance of code running in the virtual machine and make use of virtual machines more viable option for serious data processing on large scale.

References

[1] CernVM Project, http://cernvm.cern.ch

[2] LHC Computing Grid, http://lcg.web.cern.ch/LCG/public/default.htm

[3] Kernel Virtual Machine, http://kvm.qumranet.com/kvmwiki

[4] Xen, http://www.xen.org

[5] VirtuaBox, http://www.virtaulbox.org

[6] VMware, http://www.vmware.org

[7] Amazon Elastic Computing Cloud (EC2), http://aws.amazon.com/ec2

[8] Google Applications, http://www.google.com/a

[9] rPath, http://www.rpath.org

[10] Scientific Linux CERN 4, http://linux.web.cern.ch/linux/scientific4/

[11] Pagan Griso, D. Lucchesi, G. Compostella, I. Sfiligoi, D. Cesini, *CDF experience with Monte Carlo*, CHEP'07, IOP Publishing Journal of Physics: Conference Series 119 (2008) 072025

[12] Moretti C, Sfiligoi I, Thain D, *Transparently Distributing CDF Software with Parro*t, Feb 13-17 2006 Presented at CHEP06, Mumbay, India, 26.

[13] Cooperative Computing Tools, http://www.cctools.org

[14] File System in Userspace, http://fuse.sourceforge.net

[15] CernVM downloads, http://rbuilder.cern.ch/project/cernvm/releases

[16] Maeno, T, *PanDA: distributed production and distributed analysis system for ATLAS,* J. Phys.: Conf. Ser. 119 062036 (2008)

[17] Buncic, P. and Peters, A. J. and Saiz, P., and Grosse-Oetringhaus, J.F., *The architecture of the AliEn system*, CHEP 2004, Interlaken, Switzerland (2004) 440

[18] GANGA Project, http://cern.ch/ganga

PoS(ACAT08)012

[19] ROOT Project, http://root.cern.ch

[20] Amazon CloudFront Service, http://aws.amazon.com/cloudfront

[21] Amazon Simple Storage Service, http://aws.amazon.com/s3/

[22] Cohen, B.: *Incentives Build Robustness in BitTorrent* (2003)

[23] Stoica et al., *Chord: a scalable Peer-To-Peer lookup service for internet applications*, Computer Communication Review, Vol. 31, 4, p. 149-160 (2001)

[24] Wuala, http://www.wuala.com

[25] Igor-FS, http://www.igorfs.org

[26] AMD Virtualization, http://multicore.amd.com/

[27] Rich Uhlig et al.: *Intel Virtualization Technology*, Computer 38 5 (2005) 48-56

[28] The Coral Content Distribution Network, http://www.coralcdn.org

[29] Avahi, http://avahi.org

[30] BOINC, an open-source software platform for computing using volunteered resources, http://boinc.berkeley.edu

[31] DIRAC, http://lhcb-comp.web.cern.ch/lhcb-comp/DIRAC/

PoS(ACAT08)012