Building a Volunteer Cloud

Ben Segal, Predrag Buncic, David Garcia Quintas / CERN
Daniel Lombrana Gonzalez / University of Extremadura
Artem Harutyunyan / Yerevan Physics Institute
Jarno Rantala / Tampere University of Technology
David Weir / Imperial College, London

CLCAR, Mérida, Venezuela
September, 2009

1. Volunteer Computing and BOINC

Volunteer computing provides today many teraflops of contributed processor power to a wide range of scientific and technical projects. The first example of a large volunteer computing system was SETI@home **[1]**, at the Berkeley Space Sciences Laboratory. This attracted so much volunteer interest and CPU power that its inventors, led by David Anderson, went on to develop open source middleware (the Berkeley Open Infrastructure for Network Computing: BOINC **[2]**) which enables hundreds of institutes or individual researchers to access large amounts of computing power otherwise unavailable to them.

The BOINC model is based on a project server which sends out jobs on behalf of a BOINC project to any volunteer client nodes which have attached to the project to assist it with its computing tasks. The project server receives back any results for the submitted jobs, validates them and deals with issues of client down-time or unreliability. There is also a comprehensive system of project message boards to allow communication between volunteers and project staff, and credit is awarded to volunteers to encourage their collaboration.

However, until recently, each BOINC project required considerable effort to set up, first to port its computing application to a wide variety of BOINC client platforms (typically Windows, but with some Linux and MacOSX systems), and then to develop suitable job submission scripts to manage the flow of work between the project scientists and their BOINC server. The present work offers solutions to both of these problems as part of a BOINC system supporting the LHC experiments at CERN.

2. Grids and Clouds

A traditional means of supplying computing power to research projects has been the development of "research Grids". These are essentially distributed federations of large computing clusters belonging to research institutes and operated by them. An alternative for smaller research groups or institutes has been the installation and support of dedicated local computing clusters, with all the operational and financial overheads that this entails.

A recent change to this traditional model has seen the emergence of "computing clouds". This revolutionary development allows efficient leverage of the installed capacity of large computing centers by offering attractive rented chunks of processor power and storage to consumers over the Internet. Providers like Amazon, Google, IBM and Microsoft can benefit from the economies of scale associated with their highly optimized installations, and customers can benefit by simply using resources when it suits them and incurring no overheads or wasted cycles when they are unused.

3. Virtualization

A key technology that has enabled such cloud development is virtualization. This permits the logical separation of an underlying physical computing fabric, installed in a computing centre, from the users' view of the computing resources provided by this fabric. In particular, underlying platform characteristics such as operating system, memory and CPU configuration, and I/O connectivity can be abstracted into virtual machines of chosen standard type(s) which can then be further custom-configured for (or by) the end-users for the period of their resource occupation. For an example of such a service, see details **[3]** of the Amazon Web Services Elastic Compute Cloud (EC2) and Simple Storage Service (S3).

But virtualization can also be extremely useful in its own right for volunteer computing, by enabling cross-platform support of project applications across the wide range of volunteer clients which are encountered on the Internet. Previously, it was necessary to port and maintain such an application on all popular volunteer platforms, particularly Windows, in order to have a sufficient pool of client machines for a demanding project. In important disciplines like High Energy Physics, almost all code is developed under Linux (in fact under a particular brand of Linux, Red Hat Scientific Linux SL4.x or SL5.x), so porting to Windows or even to other Linux flavours, is extremely undesirable. Both the quantity of code involved and the frequency of code changes makes porting impractical and working physicists resist strongly any such suggestion. However, using virtualization, an entire application environment including operating system, code, libraries and support utilities can be incorporated into a virtual image, which can then be executed under a suitable virtual hypervisor installed on the client machines, ensuring complete compatibility of the application with the developers' own version.

4. BOINC and Virtualization

This work, as well as most of that which followed and is described in this paper, was carried out by CERN Summer Students and Research Fellows in the summers of 2006-2009. Starting in 2006, we started to investigate the use of either emulation or virtualization for BOINC, driven by the porting considerations just described. We quickly showed that emulation was too slow but that virtualization entailed only a few percent of CPU overhead and looked very promising, given the scale of the potential volunteer base.

We looked for a method which would require little or no changes to the standard BOINC client or server infrastructure. The method chosen was based on the "Wrapper" technique used for porting "legacy applications" to BOINC (i.e. those whose source is not available and which can therefore not be ported in the usual way using the BOINC API). The standard BOINC Wrapper **[4]** simply forks and execs the binary of a legacy application and then communicates on behalf of the running application process with the BOINC core client code in the volunteer machine. To begin our work with virtualization under BOINC, we simply made minimal modifications to the standard Wrapper code so that it would boot a virtual machine under a preloaded hypervisor and then run a virtual image instead of a host executable. This yielded only primitive BOINC client functionality but was enough to show that we could run virtualized applications in this way. See reference **[5]** for a detailed report of this work.

Summer 2007 brought further progress **[6]**. We attacked the problem of running a typical LHC physics application and succeeded with a full ATLAS ATHENA environment running under VMware and executing realistic test runs of the simulation package ATLFast under BOINC. However we rapidly encountered the problem of excessive image size. The size of a virtual image incorporating an entire HEP physics environment can be very large, of the order of 10 GByte. One is also confronted with the problem of incorporating frequent code changes, requiring an image replacement at each code or library update.

We investigated a possible solution to this problem. The image size was reduced by mounting most of the ATLAS libraries via the distributed file system AFS. But the system was not as robust as desired and it was felt that AFS, even read-only, may not be accessible to many volunteer clients out on the Internet.

5. The CernVM Project

In 2008, a CERN R&D project called CernVM **[7]** was launched by Predrag Buncic, offering a general solution to the problem of virtual image management for physics computing at the LHC experiments. Instead of loading each running virtual machine with a full image containing all the code and libraries for an experiment's applications, only a basic "thin appliance" of about 100MB is loaded initially, and further image increments are demand-loaded as needed by any given application and choice of LHC experiment. Updates to images after code changes are made automatically by the CernVM system, which keeps up to date versions of all image modules in a repository service for each supported LHC experiment. The resulting working images are typically under 1 GByte in size and are also cached by the virtual machines, minimizing access to the CernVM repository until changes appear in the physics code or a new type of application needs to be executed.

Not only has CernVM solved the problems of virtual image size and of image updating, but it also satisfies the physicists' requirement of requiring absolutely minimal changes to their working habits. In effect, with only trivial extensions to their existing code building scripts, they obtain a complete set of virtual image modules in the CernVM repository at the same time as their normal set of Red Hat Linux executables.

6. CernVM and Clouds

The virtual image formats that can be produced by CernVM are extremely varied. Support exists for basically every known hypervisor. In particular, physicists appreciate its support for running images under VMware, VirtualBox or Parallels, allowing them to test and develop large physics packages on their Windows or MacOSX laptops. Another very fruitful option opened to them by CernVM is that, with no further changes, they can run their applications on a production scale on computing clouds such as the Amazon EC2. Even though such commercial cloud offerings must be paid for, this has already affected the outlook of the physics communities and has put pressure on the LHC Grid community who are trying to serve physicists' needs using an older and sometimes less convenient infrastructure. The actual price of an EC2-like solution is not so much more today than a Grid solution, if all the overheads are fully accounted for. Of course, not all LHC physics computing is suitable for cloud operation, particularly that which is I/O intensive or needs widely distributed data sets; but a significant proportion can be, depending on market prices of the respective service offerings.

7. Combining BOINC and CernVM

With the advent of CernVM in 2008, we realized that a solution for our remaining problems now existed, and work was begun to prepare for more general support of virtual hypervisors in BOINC, capable of running CernVM or other virtual images as guest processes under control of the BOINC core client in the host machine. To achieve this, we decided to use hypervisors such as VMware and VirtualBox which had started to expose full-function API's to start and stop virtual machines, load and save running images, and communicate with the guest processes in the VM's. Two early prototypes were made independently by David Weir and Kevin Reed of IBM, reported in **[8]**, using modified BOINC Wrapper programs and the VMware Server hypervisor**.**

As a further step, a general-purpose "VM controller" layer was written by David Garcia Quintas **[9]** which allows asynchronous communication to occur among host and guest entities, and files and other process information to be exchanged between the host and guest layers. Generic support for various hypervisors was incorporated in this layer and VirtualBox was chosen for the intensive testing which followed.

Next, a completely new wrapper called "VMwrapper" was written by Jarno Rantala **[10],** using the VM controller services. VMwrapper is written in Python using BOINC API Python bindings written by David Weir **[11]**. The VM controller code is also written in Python. To configure the new BOINC-VM applications, VMwrapper supports XML files with formats based on standard BOINC job.xml files but with additional tags to support the new functions associated with VM and guest process control. VMwrapper is also functionally back-compatible with the standard BOINC Wrapper, and will run a host application as before if provided with a standard job.xml file.

8. Last Steps to a Volunteer Cloud

In order to provide additional computing resources to the LHC experiments, it is important not to expect the physicists who run job production to make changes to their existing scripts or procedures. To use BOINC in a classic way would violate this principle, and likely result in a system without many users. But we observed that a large amount of LHC job production is being done using a "pilot job" approach: rather than submit jobs to the Grid or to in-house clusters using existing schedulers, the LHC experiments have developed their own job submission and scheduling systems (e.g. **[12]**,**[13]**), which send pilot jobs into a fabric and use them to work out the best scheduling strategies to use at any given time. These systems also account for failures of jobs or compute nodes, considering the fabric as an unreliable resource. This corresponds perfectly to the situation with a collection of BOINC resources which may appear, disappear or run intermittently.

So we decided to interface to the Pilot-job systems, and chose to use a generic interface called Co-Pilot **[14]** which offers a gateway to the differing Pilot-job implementations of the LHC experiments. On each experiment's side of the gateway, a software package called a "Co-Pilot Adapter" is required: currently only ALICE has such an adapter and so initial testing has used ALICE jobs. Adapters for ATLAS, CMS and LHCb will be produced in order to complete the system.

In effect we have been able to set up a BOINC computing configuration which simply appears as an additional cloud resource for the LHC experiments, in exactly the same way as EC2 and other cloud resources have been interfaced to them. All the code to support the Co-Pilot agents, and thus to communicate with the LHC pilot job schedulers via the Co-Pilot adapters, is included in the CernVM images that we use. Thus no changes to BOINC or to any LHC experiment code or procedures are needed.

This resulting "volunteer cloud" for LHC computing is about to be beta-tested, initially for the ALICE collaboration. It is expected to be suitable for running simulation, event generation, and perhaps some analysis work, with an emphasis on CPU intensive rather than data intensive problems.

9. Acknowledgements

The authors would like to thank the ATLAS experiment, the Tampere University of Technology, the University of Extremadura, the CERN Summer Student and CERN openlab student programs, David Anderson / Berkeley-SSL and others in the BOINC community, and many other colleagues (including Markus Schulz / CERN and Ignacio Reguero / CERN) who have supported and encouraged this work since 2006.

10. References

[1]. **SETI@home**: (http://en.wikipedia.org/wiki/SETI@home)

[2]. **BOINC**: (http://boinc.berkeley.edu)

[3]. Amazon Web Services **Elastic Compute Cloud EC2**: (http://aws.amazon.com/ec2/)
and: **Simple Storage Service S3**: (http://aws.amazon.com/s3/)

[4]. BOINC Wrapper application: (http://boinc.berkeley.edu/trac/wiki/WrapperApp)

[5]. Daniel Lombrana Gonzalez et al: **"Customizable Execution Environments with Virtual Desktop Grid Computing"**,
Parallel and Distributed Computing and Systems Conference (PDCS 2007), Cambridge, Massachusetts, USA
(November 19–21, 2007).

[6]. David Weir: (https://twiki.cern.ch/twiki/bin/view/LHCAtHome/BOINCAndParavirtualization)
and: (https://twiki.cern.ch/twiki/bin/view/LHCAtHome/BOINCAndAtlas)
and: (https://twiki.cern.ch/twiki/bin/view/LHCAtHome/BOINCAndAtlasJobtransform)

[7]. Predrag Buncic et al. **CernVM**: (http://cernvm.cern.ch/cernvm/)

[8]. Ben Segal, David Weir, Kevin Reed et al: General considerations for **"Running apps in virtual machines":** (http://boinc.berkeley.edu/trac/wiki/VmApps)

[9]. David Garcia Quintas: **A host <-> guest VM communication system for Virtual Box:**
(http://boinc.berkeley.edu/trac/wiki/VirtualBox)

[10]. Jarno Rantala: **“VMwrapper”**: (http://boinc.berkeley.edu/trac/wiki/VmApps)

[11]. David Weir: **A Python API for BOINC:**
(http://plato.tp.ph.ic.ac.uk/~djw03/boinc-python/documentation-0.3.1/)

[12]. **PANDA**: (https://twiki.cern.ch/twiki/bin/view/Atlas/Panda)

[13]. **DIRAC**: **A Community Grid Solution**: International Conference on Computing in High Energy and Nuclear Physics (CHEP’07)

[14]. Artem Harutyunyan: (https://cernvm.cern.ch/project/trac/cernvm/wiki/CoPilotProtocol)