Volunteer Clouds and Citizen Cyberscience for LHC Physics

2011 J. Phys.: Conf. Ser. 331 062022

(http://iopscience.iop.org/1742-6596/331/6/062022)

Journal of Physics: Conference Series **331** (2011) 062022 doi:10.1088/1742-6596/331/6/062022

Volunteer Clouds and Citizen Cyberscience for LHC Physics

Carlos Aguado Sanchez / CERN
Jakob Blomer / CERN
Predrag Buncic / CERN
Gang Chen / IHEP, Beijing
John Ellis / CERN
David Garcia Quintas / LBNL
Artem Harutyunyan / CERN
Francois Grey / CCC
Daniel Lombrana Gonzalez / CCC
Miguel Marquina / CERN
Pere Mato / CERN
Jarno Rantala / Tampere University of Technology
Holger Schulz / CERN
Ben Segal / CERN
Archana Sharma / CERN
Peter Skands / CERN
David Weir / Imperial College
Jie Wu / IHEP, Beijing
Wenjing Wu / IHEP, Beijing
Rohit Yadav / Banaras Hindu University

CERN: European Organization for Nuclear Research, 1211 Geneva 23, Switzerland

CCC: Citizen Cyberscience Centre, CERN, Switzerland

Tampere University of Technology, Tampere, Finland

Imperial College, London, United Kingdom

LBNL: Lawrence Berkeley National Laboratory, Berkeley, California, USA

Banaras Hindu University, Institute of Technology, Varanasi, India

IHEP: Institute for High Energy Physics, Chinese Academy of Sciences, Beijing, China

Corresponding Author: b.segal@cern.ch / CHEP 2010 Presentation (PS50-4-452)

Abstract

Computing for the LHC, and for HEP more generally, is traditionally viewed as requiring specialized infrastructure and software environments, and therefore not compatible with the recent trend in "volunteer computing", where volunteers supply free processing time on ordinary PCs and laptops via standard Internet connections. In this paper, we demonstrate that with the use of virtual machine technology, at least some standard LHC computing tasks can be tackled with volunteer computing resources. Specifically, by presenting volunteer computing resources to HEP scientists as a "volunteer cloud", essentially identical to a Grid or dedicated cluster from a job submission perspective, LHC simulations can be processed effectively. This article outlines both the technical steps required for such a solution and the implications for LHC computing as well as for LHC public outreach and for participation by scientists from developing regions in LHC research.

1. Citizen Cyberscience

For centuries, citizen scientists have played an important role in many areas of science. As hands-on experimenters with their own equipment, amateur astronomers, archaeologists, botanists and ornithologists have made many original and important discoveries.

1.1 Volunteer Computing

In recent years, citizens have been able to contribute computer processing power from their private PCs, laptops and even game stations. Such ***volunteer computing*** today provides many Teraflops to more than 100 projects in a range of sciences as well as for tasks such as image rendering. Some of the most famous projects, such as Einstein@home (gravitational wave detection), Folding@home (protein folding) and ClimatePrediction.net (large-scale modelling of Earth's climate) routinely harvest the computing resources of tens of thousands of volunteers, many of these providing several computers.

One of the first examples of a large volunteer computing project 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 scientists easily to set up projects which distribute their scientific software for execution by volunteers. It is estimated that more than 2 million volunteers with more than 5 million processors have been active on BOINC projects.

1.2 Volunteer Thinking and m-Science

Apart from simply volunteering CPU power, citizens are also able to volunteer their visual and intellectual skills to deserving projects, for example by helping online to collect, digitize, classify and annotate scientific data. An example of such ***volunteer thinking*** is the project GalaxyZoo, where volunteers classify galaxy images taken by the Sloan Digital Sky Survey or Hubble Telescope, using a simple web interface. A novel type of galactic object, called Hanny's Voorwerp, was discovered by a participant in this project, Dutch school teacher Hanny van Arkel, vividly illustrating that in the Internet age, citizen science still has the potential to produce significant results **[20]**.

Mobile devices provide yet another means for amateurs to contribute to science, by actively contributing scientific data from the field. The project EpiCollect is an example of such ***m-Science*** used for mobile data collection in epidemiology as well as for other scientific tasks **[21]**.

1.3 The Citizen Cyberscience Centre

Volunteer Computing, Volunteer Thinking and m-Science represent three facets of an emerging technological and social trend which we call ***Citizen Cyberscience***. Essentially, this is the pursuit of citizen science with the help of the Internet and the increasingly sophisticated consumer computing devices that are connected to it.

Citizen Cyberscience can provide researchers with unprecedented amounts of computing power as well as an army of online research assistants, at practically no cost. Projects that routinely exploit tens of thousands of volunteer computers would cost tens of millions of dollars to set up if they had to rely on their own hardware. Contributions of volunteer thinkers and data collectors may be even more valuable.

Yet perhaps the biggest opportunity of all is for science in the developing world. Here scientists often have budget constraints that preclude investments in even modest computing clusters. They are also facing the challenge of analyzing large amounts of data with limited human resources. Finally, data collection from the field with mobile devices is a burgeoning opportunity in developing regions, where mobile devices are widespread and often provide the only available connection to the Internet.

Based on two pilot projects to promote citizen cyberscience in developing regions, called Africa@home and Asia@home, the Citizen Cyberscience Centre **[19]** was established in 2009 with CERN, the University of Geneva, and the United Nations Institute for Training and Research (UNITAR) as founding partners, and a number of universities worldwide with activities in citizen cyberscience as associated partners. The Centre is currently sponsored by IBM's World Community Grid initiative, an award from the HP Labs Innovation Research Program, and a Fellowship from the Shuttleworth Foundation in South Africa.

2. Grid vs. Cloud Computing

A conventional means of supplying large amounts of computing power to major research projects has involved 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 their own installation and support of dedicated local computing clusters with all the operational and financial overheads that this entails. Considerable work in this latter area has been carried out by the European Desktop Grid Initiative (EDGI), including the study of BOINC-Grid gateways to permit the use of volunteer computers as part of the combined Grid and desktop grid ensemble.

A recent change to this "Grid" model has been the emergence of ***computing clouds***. These leverage the installed capacity of very 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 benefit from the economies of scale associated with their highly optimized installations, and customers benefit by simply using resources when it suits them and incurring no overheads or wasted cycles when they are unused. Our present approach allows volunteer computers to become part of this paradigm.

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, I/O connectivity, memory and CPU configuration 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 **[12]** of the Amazon Web Services Elastic Compute Cloud (EC2) and Simple Storage Service (S3).

3. Volunteer Computing and BOINC

The BOINC model is based on a project server which sends out jobs on behalf of a BOINC project to all volunteer client nodes which have attached to the project to contribute to 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. This credit is not material or financial, but nevertheless represents a major incentive to some volunteer communities, who form teams and compete actively for the maximum credit amount.

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(s).

3.1 The LHC@home project

An example of such a traditional BOINC project is LHC@home. One of the very first such projects to be set up, it was launched in 2005 as part of CERN's 50th anniversary celebrations and attracted many volunteers and much media attention. The application is a detailed simulation of the colliding beam configuration in the LHC, with the aim of finding stable zones in phase space to avoid beam loss and aid the machine operation. The code ("SIXTRACK") was Fortran-based and was ported to Windows and Linux in the classic way, incorporating calls to the BOINC API library and recompiling and relinking the source code to produce BOINC executables for each client platform. A sophisticated LHC@home screen saver was also developed, but later it was realized that this consumed appreciable client resources and sophisticated volunteers would turn it off, thereby completing their work units faster and gaining more credit. Some 60,000 users with about 100,000 PC's have been active LHC@home volunteers since 2005.

After seeing the interest generated by the SIXTRACK application, the question was raised whether CERN could benefit from BOINC to assist with its LHC physics research programme as well as for accelerator design. The rest of this paper discusses the solutions found for this challenging issue.

4. "Real" LHC physics with BOINC: the problems involved

Previously, we saw the need to port and maintain applications on all popular volunteer platforms, particularly Windows, in order to have a sufficient pool of client machines for a demanding project. But in a discipline like High Energy Physics, almost all code is developed under Linux (in fact for the LHC under one particular brand of Linux, Scientific Linux), and porting to Windows or even to other Linux flavors, is extremely arduous. In fact, the quantity of code involved and the frequency of code changes makes porting impractical and working physicists strongly resist any such suggestion. Each experiment has a huge codebase with a lot of third-party dependencies, and the individual LHC collaborations have different coding environments. Code rebuilds are typically done weekly and sometimes daily.

4.1 Virtualization as a possible solution

By using virtualization, the entire application environment including the 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 applications with the developers' own versions. This solves the basic "porting problem" but leaves us with another problem: the "image size problem". The size of the virtual image produced by encapsulating the whole environment of an LHC physics experiment is of the order of 10 Gigabytes. Even if we accept the overhead of downloading this image initially to a prospective client node, the prospect of rewriting it every time any changes are made in the codebase or libraries becomes impractical.

4.2 BOINC job management

Another problem which arises when using standard BOINC for large scale LHC physics computing is related to the BOINC method of running and managing a job load. Each "job" (i.e. from a physicist's point of view, each physics event or group of events) gets mapped to a BOINC "work-unit", which in

turn is split into several "results" and sent out to independent volunteer clients for solution. When two or more results for a single work-unit are received and "validated", the work-unit is considered complete, the volunteers receive their credits, and only then can the project's job database be updated. Each physics experiment has to write scripts to feed jobs from their job production system into the BOINC project server, and wait until this process is complete for each job. As the actual handling of results is quite uncertain, and as BOINC does not provide in-depth monitoring tools, it is not possible for a physicist to track the progress of any particular set of jobs, change their priority or cancel them, or even estimate precisely when a set of jobs will be completed. This is an unacceptable situation for them.

4.3 LHC experiment job production systems

However, we observed that the LHC experiments have developed their own job submission and scheduling systems (e.g**. [5], [9]**); these send "pilot job agents" into a Grid or Cloud fabric which are used to work out the best scheduling strategies to use at any given time and which then "pull in" the real jobs for execution. These systems also account for failures of jobs or computer 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. These Pilot-job schedulers also deal with job monitoring and management issues, as well as job validation (thus avoiding the need for redundant result handling in BOINC).

So we decided to interface to the LHC experiments' own Pilot-job systems, and chose to use a generic interface called Co-Pilot **[7]** which offers a gateway to these (differing) Pilot-job implementations. Details are given in the following sections of this paper.

5. CernVM

We now describe how detailed solutions to the above problems (of image size and of job interfacing via the CoPilot sytem) have been developed recently at CERN.

In 2008, a CERN R&D project called CernVM **[4]** was launched by Predrag Buncic and collaborators in the PH Department, offering a general solution to the problem of virtual image management for physics computing at the LHC experiments. As three separate papers have been presented at this conference on CernVM itself **[16]** and its associated systems: the CernVM File System **[17]** and the Co-Pilot Cloud-Grid interface **[18]**, we will only give summary details of these systems here, even though they play key roles in the success of our work.

5.1 Image size optimization

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 200 MB is loaded initially, and further image increments are demand-loaded as needed by any given application and choice of LHC experiment. Image updates after code changes are also made incrementally via the CernVM File System and Repository Service which keep up to date versions of all image modules for each supported LHC experiment. The resulting working images are typically under 1 GB in size and are 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 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 they build their normal set of Scientific Linux executables. See Figure 1 for the system architecture.

5.2 Co-Pilot subsystem

Figure 2 shows the Co-Pilot system architecture. Note that the "untrusted" BOINC systems can communicate only with the intermediate Co-Pilot services and not directly with the experiments' Grid services which require Grid certification for full access. On each experiment's side of the gateway, a software package called a "Co-Pilot Adapter" is required. ALICE/AlieN received the first such adapter and so early testing of our system used ALICE jobs, but an adapter for ATLAS/PanDA was also recently written and tested, as well as an adapter for Monte Carlo jobs run for a group of theoreticians

led by Peter Skands in the CERN Physics Department. Adapters will also be produced for the CMS and LHCb experiments in order to complete the system.

6. Connecting BOINC and CernVM

6.1 Basic Virtual Machine support for BOINC

To accommodate Virtual Machine technology into BOINC, we use a method which requires essentially no changes to the standard BOINC client or server infrastructure. This is 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 **[3]** simply forked and executed the binary of a legacy application, then communicated with the BOINC core client code on behalf of the running application process, all running in the volunteer host machine. To begin our work with virtualization under BOINC (starting in 2006), 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 the paper by Daniel Lombrana Gonzalez **[8]** for a detailed report of this work. Two further early prototypes were made independently by David Weir and Kevin Reed of IBM, reported in **[11]**, using modified BOINC Wrapper programs and the VMware Server hypervisor**.**

6.2 Interfacing to VM hypervisors and VM processes

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 the 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 exploit hypervisors such as VMware and VirtualBox which exposed full-function APIs to start and stop virtual machines, load and save running images, and communicate with the guest processes in the VMs.

As a first step, a general-purpose "VMController" service layer was written by David Garcia Quintas **[6]** 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, including VMWare, VirtualBox and others offering a suitable external API. In fact, VMController is not restricted to supporting BOINC clients but allows full control of guest virtual machine(s) by any host process via a convenient XML-RPC interface. The VMController code is written in Python for cross platform compatibility. It is currently being extended and packaged by Rohit Yadav **[14]** as part of a university thesis project.

At the same time, a completely new BOINC wrapper called "VMwrapper" was written by Jarno Rantala **[10]** using the VMController services. VMwrapper is also written in Python, using BOINC API Python bindings written by David Weir **[13]**. 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 the VM and guest process control. (In fact, VMwrapper is also functionally back-compatible with the standard BOINC Wrapper: if provided with a standard BOINC job.xml file it will run the application in the host processor just as before).

6.3 Interfacing to CernVM

Although the above Python-based software offered the most general capabilities for supporting a wide range of VM based systems within BOINC, its use was in fact “overkill” for the CernVM application that we will describe below. We also encountered packaging problems when building installers that would work on the wide variety of BOINC clients we wanted to support (various Windows and Linux flavours as well as MacOSX). So to begin alpha system tests as soon as possible, Jie Wu **[15]** wrote a stripped-down “CernVM-Wrapper” in C++ with the minimum necessary functionality, namely to configure, launch and control a CernVM virtual machine under the VirtualBox hypervisor, and communicate on its behalf with the BOINC core client, including the function of allocating credit for

the volunteer based on resource usage by the virtual machine. Porting this C++ wrapper to Windows, Linux and MacOSX platforms was much simpler than dealing with Python subsystems.

7. The resulting system: BOINC-CernVM

The resulting BOINC-CernVM system is now being maintained at CERN with the support of the Citizen Cyberscience Centre. It works as follows:

- the BOINC volunteer first pre-installs the open source hypervisor VirtualBox on his system (we plan to avoid the need for this step in the future).
- The volunteer then attaches to the BOINC-VM project server in the standard way, using unmodified standard BOINC client software.
- The volunteer receives a single "work unit" containing two files: the CernVM-Wrapper (the executable), and the standard CernVM virtual image containing the Thin Appliance plus the Co-Pilot agent.
- The CernVM-Wrapper is executed: it configures, loads and runs the CernVM image in a guest VM under VirtualBox. From this point on, the Wrapper task simply behaves like a long-running BOINC host application but does no serious computing.
- Inside the guest VM, CernVM runs in a standard way, including a connection to the CernVM file system repository. The Co-Pilot agent, started at CernVM boot time, locates the remote job queue corresponding to the desired LHC experiment's Pilot Job scheduler, receives jobs to be executed in the CernVM environment, and returns the results.
- The CernVM-Wrapper regularly measures the total CPU resources used by the virtual machine, and requests the appropriate credit from the BOINC project server.
- During the first job of a series, the CernVM running image is automatically expanded by the addition of any missing libraries or other modules needed by the specific job environment. Later jobs do not require this phase as all image changes are cached in the guest VM.
- The job flow is entirely under the control of the external Pilot Job scheduler; no BOINC result scheduling is done. In fact the BOINC volunteer PC only sees a very long-running task which grants credit regularly but does not terminate, unless the volunteer suspends or terminates it. (In a later stage, we plan to inform the BOINC volunteer about job activity within the VM, but this is merely for interest and is not an essential feature).
- The Pilot Job scheduler simply sees an executing CernVM node of a standard type which accepts and returns jobs, perhaps less reliably than other nodes, but no different in principle.
- With many BOINC clients attached, the Pilot Scheduler sees a Cloud of CernVM nodes, looking the same as a group of CernVM nodes on Amazon EC2 or other virtual fabrics.

The BOINC-CernVM system architecture is shown in Figure 1 below, and the BOINC-CoPilot Volunteer Cloud architecture is shown in Figure 2.

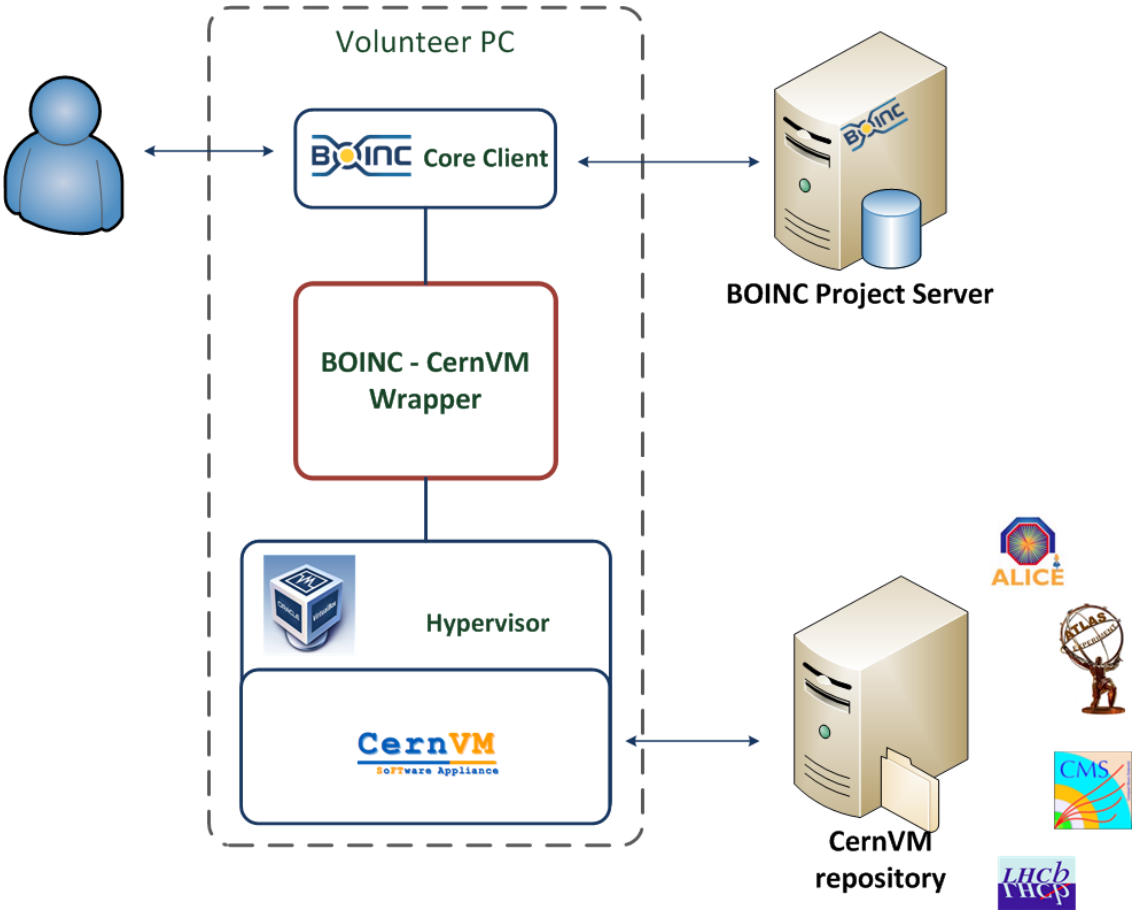

Figure 1: BOINC – CernVM System Architecture

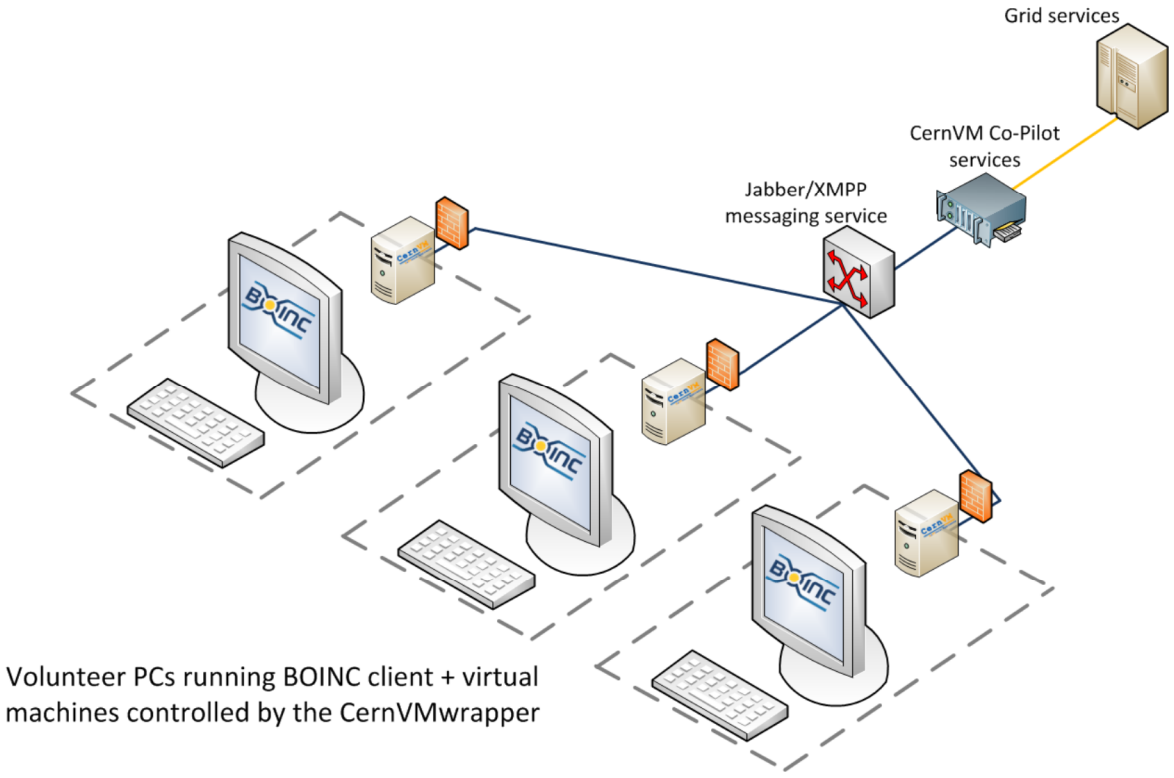

Figure 2: BOINC + CernVM + Co-Pilot Volunteer Cloud Architecture

8. A Volunteer Cloud for LHC physics

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 are needed to BOINC client or server code, or to any LHC experiment code or procedures as long as they use CernVM.

This resulting "volunteer cloud" for LHC computing is about to be beta-tested, initially for the CERN Theoretical Physics Group running Monte Carlo QCD event generation, and then for the LHC experiment collaborations for running simulation, event generation, and perhaps some reconstruction, with an emphasis on CPU intensive rather than data intensive problems.

8.1 Implications for LHC physics

While caution is required when extrapolating from the current results, it is worth noting that the total processing requirements for LHC computing have been estimated at 100k CPU's, and this number will no doubt grow with the increasing sophistication of the simulation and analysis software. The costs involved in running individual data centres for the LHC, including hardware, electricity and manpower, form a significant part of the budgets of CERN and of many of the other major institutional partners contributing to the LHC. Any way to reduce these costs ought to be welcome, especially in the current austere budget climate for HEP.

So farming out a fraction of the LHC physics simulation to volunteer resources can be an attractive option. And not only is it a budget saving for the HEP community, but looked at globally, volunteer computing is an energetically more efficient approach, especially if – as is common these days – volunteers are encouraged to run BOINC only in the background when they are otherwise active on their computers. In this mode, the extra power consumed is often only a few percent of the "ON" power of a regular PC or laptop.

The results we have outlined in this paper demonstrate that, thanks to virtualization, exploiting volunteer computing for meaningful LHC physics is now technically feasible. Just as important for a successful uptake of this technology, though, we have demonstrated that it can be provided to the LHC physicists in a form essentially identical to other computing resources, therefore requiring no extra effort on their part to adapt to it, through the concept of "volunteer clouds".

Finally, it is clear that many other scientific communities could benefit from the same approach we have discussed above in the LHC context. Many of them may have less demanding requirements than those of the LHC physicists, so they will perhaps not need all the power of the CernVM / Co-Pilot approach to enjoy access to volunteer cloud computing.

8.2 Implications for LHC public outreach and international participation

During the six years that LHC@home has been operating, and particularly since the LHC itself has been generating data, a recurring question from volunteers has been whether they could participate more directly in LHC physics computing. Indeed, the volunteer community includes many sophisticated citizen cyberscientists who are very well informed about the LHC and the physics it is attempting to unravel, and are keen to be more closely involved in that process.

The enormous and sustained interest of international media for the LHC should provide a huge base of support for a future expansion of the LHC@home volunteer base. So there is no reason to suppose that such an expansion would saturate volunteer contribution. Of course, as other volunteer computing projects can attest, managing the volunteer community represents a non-negligible amount of effort. But since public awareness and interest in fundamental science is of great value to the HEP community, as it translates into political support for fundamental research, this effort would seem a small price to pay for both the computing and public outreach benefits it generates.

There are many directions that volunteer based LHC computing could go in future. These range from adapting new releases of software from the various LHC experiments for use on volunteer resources such as GPUs and game stations, to exploring possible uses of volunteer thinking in LHC data analysis. These options represent a series of projects that would be well suited for scientists with limited resources wishing to establish themselves in the LHC community.

Through the Citizen Cyberscience Centre, it is our goal to encourage and facilitate such participation in the future expansion of the LHC@home project. In 2011, we are planning a series of workshops for scientists in India, China, Brazil and South Africa, with the support of the Shuttleworth Foundation, introducing Citizen Cyberscience to communities that are unfamiliar with it, and using LHC@home as a prominent illustration of how to get involved.

There are over a billion PCs on the planet now and some 5 billion mobile devices. Most important of all, there are nearly 7 billion people, a rapidly increasing fraction of whom have Internet access. Currently, the sum total of active citizen cyberscientists in any form is probably less than 1 per mil of the global population, and the ones currently involved in LHC@home represent about 1% of that.

The opportunity to engage more citizen cyberscientists in LHC research beckons. And if the HEP community can seize that opportunity, it is our conviction that HEP will acquire a significantly enhanced potential for scientific discovery, through public participation in science.

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.

9. References

[1] **SETI@home** - http://en.wikipedia.org/wiki/SETI@home.
[2] **BOINC**: Berkeley Open Infrastructure for Network Computing - http://boinc.berkeley.edu
[3] **BOINC Wrapper application** - http://boinc.berkeley.edu/trac/wiki/WrapperApp
[4] **Buncic, P. et al.** CernVM - http://cernvm.cern.ch/cernvm/.
[5] **DIRAC**: A Community Grid Solution. International Conference on Computing in High Energy and Nuclear Physics (CHEP'07), 2007.
[6] **Garcia Quintas, D.** A host <--> guest VM communication system for Virtual Box http://boinc.berkeley.edu/trac/wiki/VirtualBox.
[7] **Harutyunyan, A.** https://cernvm.cern.ch/project/trac/cernvm/wiki/CoPilotProtocol
[8] **Lombrana Gonzalez, D., et al.** Customizable Execution Environments with Virtual Desktop Grid Computing. En Parallel and Distributed Computing and Systems Conference (PDCS 2007), Cambridge, Massachusetts, USA, 2007.
[9] **PanDA**: The PanDA Production and Distributed Analysis System - https://twiki.cern.ch/twiki/bin/view/Atlas/Panda.
[10] **Rantala, J.** VMwrapper - http://boinc.berkeley.edu/trac/wiki/VmApps.
[11] **Segal, B., Weir, D., Reed, K., et al.** General considerations for running apps in virtual machines. http://boinc.berkeley.edu/trac/wiki/VmApps.
[12] **Amazon Web Services**: Elastic Compute Cloud EC2 - http://aws.amazon.com/ec2/ and Simple **Storage Service S3** - http://aws.amazon.com/s3/
[13] **Yadav, R.** VMController - http://code.google.com/p/vmcontroller/
[14] **Weir, D.** A Python API for BOINC - http://plato.tp.ph.ic.ac.uk/~djw03/boinc-python/documentation-0.3.1/.
[15] **Wu, J.** BOSS and LHC Computing using CernVM and BOINC - https://openlab-mu-internal.web.cern.ch/openlab-mu-internal/03_Documents/3_Technical_Documents/Technical_Reports/2010/openlab-report-Jie-Wu-2010.pdf (PS06-5-434) - CHEP 2010
[16] **Buncic, P. et al.** CernVM: Minimal Maintenance Approach to the Virtualization (PS29-4-432) - CHEP 2010
[17] **Blomer, J. et al.** Distributing LHC Application Software and Conditions Databases Using CernVM File System (PS06-5-434) - CHEP 2010
[18] **Harutyunyan, A.** CernVM CoPilot: a Framework for Orchestrating Virtual Machines Running Applications of LHC Experiments on the Cloud (PS44-4-435) - CHEP 2010
[19] **Citizen Cyberscience Centre** - http://www.citizencyberscience.net/
[20] **Hanny's Voorwerp** - http://en.wikipedia.org/wiki/Hanny's_Voorwerp
[21] **EpiCollect** - http://www.epicollect.net/