oTN-2010-0x openlab Summer Student Report

BOSS and LHC computing using CernVM and BOINC

Jie Wu (Supervisor: Ben Segal / IT)

1 December
2010
Version 1
Distribution:: **Public**

Abstract

This openlab project provided for a student to work with both IHEP and CERN experts to deploy a multi-platform version of BOSS, the **B**ESIII **O**ffline **S**oftware **S**ystem software suite for BESIII simulation and analysis. BESIII is a general-purpose detector being built to meet requirements of physics studies at BEPCII, which is a multi-bunch e^+ e^- collider at IHEP. The deployment was made using **CernVM**, a general solution to the problem of virtual image management for physics computing, and is intended both for distribution in clusters on the IHEP site, and also as part of a BOINC solution called **BOINC-CernVM**.

The student also helped with testing and demonstrating a BOINC-CernVM prototype system running LHC physics problems for the ATLAS experiment and for CERN's Theory Department. The work involved writing a "C-Wrapper" interface allowing standard BOINC client software to control a CernVM virtual machine running under a VirtualBox hypervisor on both Windows and Linux BOINC client platforms.

Introduction and Goals

Work on combining CernVM **(Ref.1)** and the volunteer computing platform BOINC **(Ref.2)** carried out in part by CERN openlab students over the past two years, is now reaching a point of maturity where it is ready for practical deployment and testing.

At IHEP in Beijing, a project called CAS@home is looking at porting BOSS, the software suite for BESIII simulation and analysis, to various platforms and clusters including BOINC. Since BOSS is very similar in structure to suites like ATHENA, this matches well to the CernVM and BOINC work that has already been done at CERN.

This openlab project therefore provided for a student to work with both IHEP and CERN experts to deploy BOSS using CernVM, both for distribution in clusters on the IHEP site, and also as part of a BOINC solution. The student should also help with final steps of testing and demonstrating BOINC-CernVM applied to LHC physics problems.

The result of this project will be a reference deployment that can be used by other institutions interested in analogous deployment on local or volunteer resources.

1 Background at IHEP

1.1 BOINC and BOSS at IHEP

So far at IHEP, a BOINC server, website framework and a screensaver application have been successfully developed for CAS@home. Work is now ongoing to deploy some applications on CAS@home, such as BOSS, Genome Comparison and so on.

One of the problems IHEP has encountered is that their programs for scientific computing usually run on Linux, but volunteers' operating systems are of a wide variety and most of them are Windows. So being able to run the applications on varied platforms is absolutely necessary. This is particularly true for High Energy Physics applications like BOSS.

There are two ways to solve the problem. One is to adapt the BOSS source code to each platform. It could be handled but means a colossal waste of time and money in normal conditions. The other is virtualization. What is needed is to install a virtual machine and run the appropriate operating system in the virtual machine and there is no need to modify source code. CERN has obtained good results for this using CernVM, so we decided to transfer the necessary know-how from the CernVM team to IHEP.

1.2 Details of the BOSS application

The name "BOSS" stands for the **B**ESIII **O**ffline **S**oftware **S**ystem. BESIII is a general-purpose detector being built to meet requirements of physics studies at BEPCII, which is a multi-bunch e^+ e^- collider that has been providing collisions since 2007.

BOSS is developed using the C++ language and object-oriented techniques, under the operating system Scientific Linux CERN (SLC). CMT is used as the software configuration tool. The BESIII software uses lots of external HEP libraries including CERNLIB, CLHEP, ROOT etc. and also re-uses parts of code from the Belle, BaBar, ATLAS and GLAST experiments. The whole data processing and physics analysis software includes five functional parts: framework, simulation, calibration, reconstruction, and analysis tools.

The framework part is developed based on GAUDI, which provides standard interfaces for the common software components necessary for data processing and analysis. The simulation part is developed based on GEANT4, which simulates the collision processes and the response of the detector. The calibration part is based on the GLAST scheme, which provides reconstruction algorithms in a standard way to obtain the calibration data objects and produces the calibration constants for each sub-detector using an associated calibration algorithm. The reconstruction part takes charge of data reconstruction, which is the central task of the offline software. The analysis tools part includes a set of tools which make the operation of the entire software easier.

2 Background at CERN

We now summarize some of the earlier work already achieved in CERN-IT for BOINC (LHC@home) and by the CernVM team in CERN-PH (CernVM, CVMFS and Co-Pilot).

2.1 BOINC at CERN

One of the very first BOINC projects to be set up, LHC@home 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 standard BOINC way, incorporating calls to the BOINC API library and recompiling and relinking the source code to produce BOINC executables for each client platform. Some 60,000 users with around 100,000 PC's have been active 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. Work began in 2006 to look for the solutions required to achieve this challenge. The relevant references are noted in Jarno Rantala's 2009 CERN openlab project report **(Ref 3)**. The present project follows directly in the line taken by this previous work.

2.2 CernVM and CVMFS

As stated in the Introduction, virtualization permits cross-platform execution of applications developed under each specific LHC software environment. But to support a full set of LHC experiment software packages, a very large size of virtual image is required (8-10 GB); furthermore, some component packages also change frequently, requiring a full image reload in principle. Both these factors would be show-stoppers for the BOINC environment.

The CernVM project was launched as a general solution to the problem of virtual image management for physics computing. Instead of loading each running virtual machine with a full image, only a basic "thin appliance" of about 200MB is loaded initially, and further image increments are downloaded from a CVMFS image repository as needed by any given application and choice of LHC experiment. In this way a virtual image of the order of 1 GB suffices to run all typical LHC physics problems. An image is not only custom-built incrementally for each application but is cached on each execution node and so remains available for succeeding jobs of the same application type, without further need for communication with the repository. Updates to images are made automatically by the CernVM system when required by changes in the package sources in CVMFS. The use of CernVM/CVMFS thus makes it possible to distribute LHC physics jobs efficiently to any virtual machine, including all important platform types which can run those machines under a suitable hypervisor.

2.3 Co-Pilot

Each LHC physics experiment needs to run hundreds of thousands of jobs on the various computing fabrics available to it. This job flow is managed by "job production" facilities of several types. We choose to interface CernVM virtual machines as a "best-effort Cloud", using the same approach as has recently been taken to interface the ALICE experiment's physics job production system to other Cloud facilities such as Amazon EC2, reported in the paper CHEP179 **(Ref. 4).** The information in the rest of this section is based on a part of this paper.

The approach is based on "pilot jobs" sent out by the job production system to explore cloud resources, and "pilot agents" running on the cloud's worker nodes which will request and run received jobs. The approach is general, as all the LHC experiments have pilot job systems, even though they are not identical. To exploit this fact, an "Adapter" is introduced between the pilot job system and the clouds to be interfaced: on each cloud worker node (in our case a CernVM virtual machine) a standard "Co-Pilot Agent" runs and uses the Jabber/XMPP instant messaging protocol to communicate with the "Co-Pilot Adapter", which runs outside the cloud and interfaces to the various job production systems. The use of Jabber/XMPP allows scaling of the system in case of a high load on the Adapter node by just adding new Adapter node(s) to the existing messaging network.

The Co-Pilot Agent code is a standard part of CernVM. In the case of BOINC-CernVM it is thus freely available to BOINC nodes enabled for running with CernVM. The Co-Pilot Adapter system exists already in versions for the ALICE experiment, in this case interfacing to the AliEn job production system, and for ATLAS, interfacing to PanDA. Later versions will also support the CMS and LHCb experiments. During this project, a simple Co-Pilot job queueing interface was also made by the Co-Pilot team to support Monte Carlo applications for the CERN-TH department (**Ref. 5**).

3 BOINC wrapper technology

In order to support guest virtual machines inside a BOINC client host machine, it is necessary to "wrap" the hypervisor and the virtual machines using the same wrapper technology that is used in BOINC to support "legacy applications" – i.e in the cases where one is unable to make modifications or relink the code of these applications with BOINC libraries.

3.1 Original wrapper

The original wrapper **(Ref. 6)** is standard BOINC source code, written in C. This wrapper communicates with the BOINC core client and BOINC server on behalf of the wrapped application, e.g. for starting, stopping, suspending, resuming, aborting and measuring CPU time of the application. The standard BOINC wrapper runs and controls applications as local processes in the host machine.

3.2 Wrappers for virtual machines

An earlier CERN openlab project already referenced **(Ref. 3)** created a new BOINC wrapper called "VMwrapper", (http://boinc.berkeley.edu/trac/wiki/VmApps) which works back-compatibly with the original one: i.e. one can give it an unmodified input configuration file (job.xml) and VMwrapper will do the same as the original wrapper. But in addition, one can now run applications in guest virtual machines instead of in the local host, using new XML tags in the job.xml file. In this case, VMwrapper allows control of the virtualized application (start, stop, suspend, resume, abort, etc.) by the BOINC server or by the volunteer via the BOINC core client.

VMwrapper is written in Python using BOINC API Python bindings written by David Weir. VMwrapper uses the "VM Controller" package written by David Garcia Quintas to communicate with the guest VMs. The VM controller code is also written in Python.

4 Results

We now present the results of the present openlab project:

4.1 Porting of BOSS to CernVM, CVMFS and Co-Pilot

Typically, to deploy BOSS on a physical machine is very tedious. Apart from the fact that a specific OS and many prerequisites must be installed, just the installation of BOSS itself is very complicated: please refer to http://boss.ihep.ac.cn/GuideAndTraining/SetEnv-6.1.0.htm for more details. But if virtualization can be applied to the installation, it will be another situation. What we would need to do then is just to download and install a virtual machine hypervisor, then to download a virtual image which BOSS has been installed in, configure and run it. To install BOSS and apply it like that becomes easy.

The other motivation to virtualize BOSS is that we want to run BOSS on BOINC: the majority of BOINC volunteers are running Windows on their computers and BOSS runs on Linux, and to develop a new version which runs on such different platforms is very hard. With virtualization, we can solve this porting problem. But two challenges make this nearly impractical. The first is that the BOSS structure is very complex and rapidly changing. The other one is that the size of BOSS is as big as 5GB, which is impractical for its distribution to most volunteer computers.

Fortunately, the CernVM system was proposed at CERN, which makes it realistic to virtualize BOSS and deploy it successfully on BOINC. With CernVM we can solve both the image size problem and the problem of managing frequent BOSS code updates.

As shown in Figure 1, this model is based on BOINC, CernVM and Co-pilot. BOINC takes charge of the work to download the CernVM image, deploy, configure and control the virtual machine, and grant credits to volunteers, while Co-Pilot takes charge of the work to fetch jobs from the BOSS job production system, execute them, and upload results via the Co-Pilot Adapter back to the job production system.

Because a managed virtual image is used in this model, it is not necessary to deal with the cross-platform problem. The most important aspect of this model is its use of the shared file system CernVM File System (CVMFS). When the virtual machine in a volunteer computer is started, it will automatically load (and cache locally) all software libraries according to the needs of the current job. As a result, the problem of dealing with a potentially huge source code is solved, as well as the image update problem.

So far, BOSS has been ported to CernVM and successfully run test jobs in CernVM. The next step is to develop a Co-Pilot Adapter for BOSS.

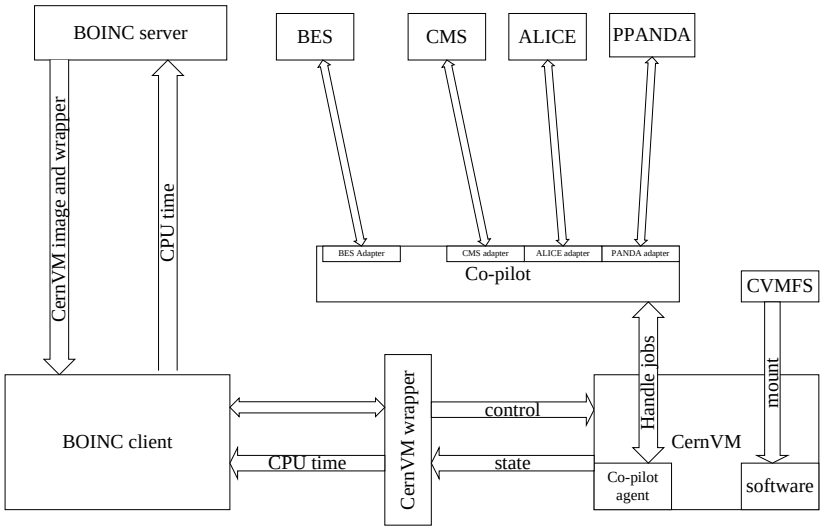

Figure 1: Sketch of the relationship between the BOINC server and client, CERNVM components and the Co-pilot job submission system.

The cookbook to port BOSS to CernVM is as follows:
1. Log in to the server boss-webfs.cern.ch
2. Install the BOSS system on the server:
   - Installation of pacman,
     tar xzvf pacman-latest.tar.gz
     *cd pacman-3.**
     source setup.sh
   - Downloading BOSS,
     mkdir /opt/boss
     cd /opt/boss
     pacman -pretend-platform SLC-4 -get http://bes3.jinr.ru/cache:6.5.3
   - Installation of BOSS,
     source setup.sh
     source scripts/6.5.3/setup.sh
     cd dist/6.5.3/TestRelease//cmt*
     source setup.sh
3. Publish BOSS software
   - Tag the folder repository,
     /usr/bin/cvmfs-update-client -c boss
   - publish the tag in the production environment,
     /usr/bin/cvmfs-update-client -p boss
4. Test whether the installation is successful.
   - Log in to a CernVM instance and run the script http://cern.ch/hartem/boss.sh on it.
   - Run the test job,
     cd /opt/boss/scripts/6.5.3/dist/6.5.3/TestRelease//run*
     boss.exe jobOptions_sim.txt
     boss.exe jobOptions_rec.txt

4.2 Simple C-VMwrapper

As part of the present Openlab project, a minimal-function VMwrapper was written in C, as the full Python packages referred to above were not ready for deployment, and we could manage with a simpler system for prototyping the current system.

Here we use VirtualBox as the virtual machine hypervisor due to some of its characteristics, such as free availability and its support on many platforms, including Windows, Linux and MacOSX. In order to rapidly achieve the stated objectives, we did not apply VirtualBox's advanced COM API, but an earlier-implemented command-line API called VBoxManager, which supplies a set of command-lines controlling a virtual machine. The initial idea was to use functions system() and popen() to directly call VBoxManger commands, but after completing the Linux version and trying to implement the Windows version, we ran into the "black screen problem": when the wrapper was running, a black screen always popped up. The reason is that on Windows, whether it was system() or popen(), when they called a command, they always ran the command via CMD, which is the text interface of Windows. The ultimate solution was to apply pipe technology and write a substitute function vbm_open(). This function didn't use system() or popen(), but the Windows API CreateProcess().

The C-VMwrapper achieved the minimum functionality and the flow chart of this wrapper is shown in Figure 2. On the one hand, it provides the functions to control a virtual machine, including creating, starting, suspending, resuming, removing a virtual machine and getting the state of the virtual machine; on the other hand, it provides the function to measure the VM's CPU time and send this to the BOINC server.

The C-VMwrapper does not need to provide further host<->guest-VM functionality such as command execution or file transfer, due to the CernVM features and the Co-Pilot architecture which together handle all the job flow functions and the virtual image management independently of BOINC.

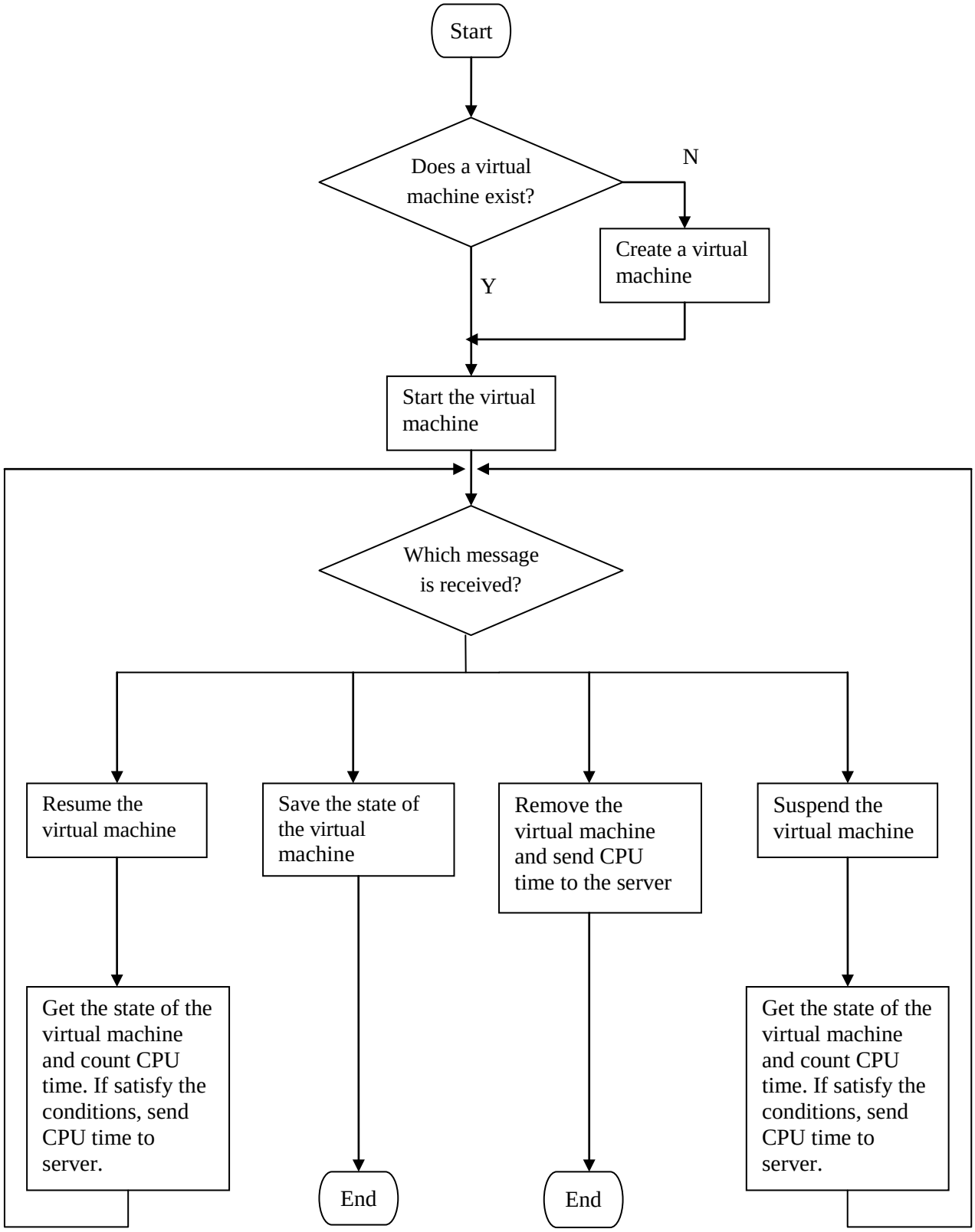

Figure 2: The flow chart of the C-VMwrapper.

4.3 Testing of resulting BOINC-CernVM prototype with ATLAS and Theory jobs

The BOINC-CernVM prototype system was successfully tested at the end of this openlab project, and allowed jobs to be run under BOINC on Windows and Linux client systems from both ATLAS and CERN-TH via the CernVM Co-Pilot interface.

A short film of a typical test session of a BOINC-CernVM Windows client can be seen at:

http://www.youtube.com/watch?v=E96Dd415LMw&hd=1

After a preliminary setup of the VirtualBox execution PATH variable (not needed in later versions), the BOINC Manager is started on the client and attached to the BOINC-CernVM test project server. A new user account is created and then the BOINC messages are displayed between the client and the test server. Soon the download transfer of the CernVM image is seen (a large uncompressed image of about 800MB), and then the BOINC Wrapper task begins - it starts VirtualBox, whose window appears, first displaying the boot process of the CernVM virtual machine. Then the messages from the Co-Pilot agent are displayed in the CernVM window, showing the sequences of job request, job arrival and job execution. The CernVM window is later switched to display the job's standard output as it is produced about 30 seconds later (from a PYTHIA programme). Several of these job sequences are shown.

Then the film shows the BOINC task being **suspended** by the user via the BOINC Manager, and the suspended state is confirmed by the VirtualBox control window (showing the VM "paused") and the CernVM window (frozen). The task is then **resumed** and the VM is seen "running" again and the CernVM window shows resumed activity.

5 Acknowledgements

I want to thank my supervisor Ben Segal for careful guidance when I was at CERN. Thanks also to Francois Grey and my IHEP supervisor Gang Chen for supporting me to study at CERN, and to Predrag Buncic, Artem Harutyunyan, Carlos Aguado Sanchez and Wenjing Wu for selfless help. Finally, thanks to the openlab project for giving me the great chance to learn more at CERN.

6 References

1. **Buncic, P. et al.** CernVM, http://cernvm.cern.ch/cernvm/.
2. **BOINC**: Berkeley Open Infrastructure for Network Computing, http://boinc.berkeley.edu.
3. **J. Rantala:** CERN openlab Summer Student Report, https://openlab-mu-internal.web.cern.ch/openlab-mu-internal/03_Documents/3_Technical_Documents/Technical_Reports/2009/Summer_Students_Reports09_Jarno_Rantala.pdf
4. **A. Harutyunyan, P. Buncic, T. Freeman, and K. Keahey**, "Dynamic Virtual AliEn Grid Sites on Nimbus with CernVM", Computing in High Energy and Nuclear Physics (CHEP), Prague, March 2009.
5. **P. Skands**, "Monte Carlo Development and Tuning", http://home.fnal.gov/~skands/slides/skands-grid-user.pdf
6. **BOINC Wrapper application**: http://boinc.berkeley.edu/trac/wiki/WrapperApp.

7 Appendices

7.1 Simple VMwrapper code in C

The up-to-date C-VMwrapper code can be found in our project repository at:

http://code.google.com/p/boincvm/source/browse/#hg/cernvm-wrapper