CernVM Co-Pilot: an Extensible Framework for Building Scalable Computing Infrastructures on the Cloud

A Harutyunyan, J Blomer, P Buncic, I Charalampidis, F Grey, A Karneyeu, D Larsen, D Lombraña González, J Lisec, B Segal, and P Skands
CERN, CH-1211 Genève 23, Switzerland
E-mail: Artem.Harutyunyan@cern.ch

Abstract. CernVM Co-Pilot is a framework for instantiating an ad-hoc computing infrastructure on top of managed or unmanaged computing resources. Co-Pilot can either be used to create a stand-alone computing infrastructure, or to integrate new computing resources into existing infrastructures (such as Grid or batch). Unlike traditional middleware systems, Co-Pilot components communicate using the Extensible Messaging and Presence protocol (XMPP). This allows the system to be easily scaled in case of a high load, and it also simplifies the development of new components.

In this contribution we present the latest developments and the current status of the framework, discuss how it can be extended to suit the needs of a particular community, as well as describe the operational experience of using the framework in the LHC@home 2.0 volunteer computing project.

1. Introduction

CernVM [1] Co-Pilot is a framework for instantiating an ad-hoc computing infrastructure on top of managed and unmanaged distributed computing resources. Such resources include public/commercial computing clouds (e.g. Amazon EC2 [2]), private/academic computing clouds, as well as the machines of users participating in volunteer computing projects (e.g. BOINC [3]). The framework consists of components that communicate using the Extensible Messaging and Presence protocol (XMPP) [4,5], allowing for new components to be developed in virtually any programming language and interfaced to existing Grid and batch computing infrastructures exploited by the High Energy Physics community.

CernVM Co-Pilot is also one of the enabling technologies behind the LHC@home 2.0 volunteer computing project [6], which to the best of our knowledge is the first project of its kind that exploits virtual machine technology. The use of virtual machines eliminates the necessity of modifying existing applications and adapting them to the volunteer computing environment. After the start of the public testing in August 2011, LHC@home 2.0 quickly gained popularity, and as of June 2012 it has about 13000 registered volunteers. Resources provided by volunteers are currently used for running Monte Carlo generator applications that simulate interactions between the colliding proton beams at the Large Hadron Collider at CERN.
In Section 2 we describe components and properties of the Co-Pilot framework. In Section 3 we discuss how the framework can be extended. In Section 4 we describe the operational experience using the LHC@home 2.0 volunteer computing infrastructure, and in Section 5 we present a summary.

2. Components and properties of the Co-Pilot framework

The four main components of the Co-Pilot framework are the Job Manager, the Agent, the Storage Manager, and the Monitor. The Job Manager is responsible for maintaining the job queue, distributing jobs for execution, as well as keeping track of job states. The Agents request jobs for execution from the Job Manager and are responsible for fetching input data, preparing the job execution environment, executing the job, and uploading job execution results. The Storage Manager provides space for uploaded job execution results, and the Monitor collects and stores information about the state of the system (such as length of the job queue, number of executed jobs, space available on Storage Managers, etc.). Agents are running on the machines where jobs are being executed, whereas the rest of the components are deployed centrally.

The architecture of Co-Pilot allows to interconnect it with existing Grid middleware systems by developing so-called Adapters. Adapters one one end communicate with Co-Pilot Agents and on the other end are interfaced to a different middleware system. Such an approach makes it possible to run Grid jobs in untrusted environments by eliminating the need of carrying Grid security credentials along with the job (further details can be found elsewhere [7]).

The architecture of the Co-Pilot system is presented in Figure 1.

Unlike the majority of middleware systems that are used for management of computing resources, Co-Pilot does not follow the traditional client-server model. Instead the framework is built around a messaging network, which operates using the XMPP protocol (Figure 1).

Another distinction of Co-Pilot as compared to traditional Grid middleware systems is that there is no notion of ‘site services’: all the communication between Agent instances and the
central services is direct. This simplifies to a great extent the operational overhead of utilizing volatile computing resources (such as the machines of users participating in volunteer computing projects), because to be able to run a job one just needs to start an Agent instance as opposed to deploying and configuring an ensemble of Grid site services.

Co-Pilot components process XMPP messages asynchronously, which means that in case of a high load the components will not be flooded with requests and possibly even brought down, but instead the requests will be queued by the XMPP server (such as the ejabberd [8]) and then processed at the pace that components can handle.

XMPP also supports so-called offline message delivery, so requests that are addressed to the components that are offline are not lost but are delivered when the component becomes available. This makes it possible to recover gracefully in cases when a component temporarily becomes unavailable (e.g. due to an unstable network connection).

The use of XMPP makes it possible for the Co-Pilot components to communicate without using listening sockets (i.e. without requiring incoming network connectivity). This makes it easier to operate the Co-Pilot components, because there is no need anymore for firewall port openings for the communication channel to work. It is also more preferable from a security point of view.

Co-Pilot components are designed so that in case there are many instances of the same component on the same messaging network (e.g. several instances of the Job Manager connected to the same messaging server) the load between them will be distributed evenly. In other words components can be scaled horizontally and independently from each other.

3. Extending the Co-Pilot framework and reusing existing components

In its current state the Co-Pilot framework provides generic components sufficient for deploying a ready-to-use distributed computing infrastructure. However it can also be used to integrate new computing resources into an existing infrastructure, in which case it is likely that a development of additional components will be required. For example in the case of PanDA [9] (the distributed computing infrastructure of the CERN ATLAS experiment [10]) we needed to develop a special storage manager in order to upload the results of the jobs to the Grid and to communicate with ATLAS Grid services for finalizing the state of executed jobs. Further details can be found elsewhere [7].

To make it easier to reuse Co-Pilot components the functionality of the framework has been split between several independent modules. In addition, the protocol which is used by the components for communication is simple and easy to implement. It consists of 10 commands for job execution and 7 commands for monitoring and status querying [11]. The protocol is based on XML, which makes it possible to implement new components in virtually any programming language. An example message sent from Agent to the Job Manager for reporting job completion is as follows:

```xml
<message to='jm@xmpp.cern.ch' from='agent@xmpp.cern.ch' ack='1a58119b5c0e4f7e883204a502e502e' >
  <info exitCode='_BASE64:MA==' jobID='_BASE64:MzA4MDE3NDE=' state='_BASE64:ZG9uZQ==' command='_BASE64:am9iRG9uZQ==' />
</message>
```

To give an example of interaction between the Co-Pilot components we present below the sequence of steps performed by an Agent for requesting a job for execution (the corresponding sequence diagram is presented in Figure 2). When the Agent has a slot for running a job it broadcasts to the Job Managers the 'want_getJob' request which contains the JDL [12] formatted
Figure 2. The sequence of commands for requesting a job

configuration of the machine where the Agent runs. Upon receiving the message those job managers that have jobs for execution send the ‘have_getJob’ command back to the agent. The agent then makes an actual job request (getJob command) to the job manager from which the first have_getJob command has been received. The job manager replies with the runJob command, which contains the information necessary to execute the job.

4. Operational experience in the LHC@home 2.0 project
The LHC@home 2.0 volunteer computing project uses the CernVM Co-Pilot framework for running jobs on the machines of volunteers. The resources contributed by volunteers are currently being used to run various Monte Carlo generators, and results are used to populate histograms on the MCPLOTS [13] repository. MCPLOTS is a simple browsable repository of Monte Carlo plots comparing High Energy Physics event generators to a wide variety of available experimental data, for generator tuning and reference purposes.

LHC@home 2.0 started its operation in November 2010. It started in a private alpha-testing mode where new users needed an invitation to join. The public beta test started in August 2011, where the project ran with the help of several dozen experienced volunteers for about 9 months.

The public beta test of the project was covered by major technological blogs and magazines (such as ArsTechnica [14] and ZDNet UK [15]) as well as by major news agencies (such as BBC [16]). This created a lot of interest among the members of general public and attracted some 8000 new volunteers within just a couple of days (Figure 3).

The Co-Pilot services, however, sustained the torrent of new users and have proven to be reliable. The architecture of the Co-Pilot system made it possible to sustain a high load by simply introducing several extra Job Manager instances to the messaging network.

As of June 2012 the LHC@home 2.0 project has about 13000 registered volunteers, several thousand of whom are active. Since the beginning of the project volunteers have already contributed about 3000000 CPU-hours. This is an exciting result for us because volunteers have provided a great computing capacity which would not have been available otherwise. The LHC@home 2.0 has become the main source of computing power for the MCPLOTS project.

5. Summary
CernVM Co-Pilot is a generic framework for building an ad-hoc distributed computing infrastructure. It can also be used to transparently integrate new resources into existing computing infrastructures (such as Grids). Individual components of the system can be scaled horizontally so the overall system can sustain a high load. Co-Pilot makes it possible to exploit unmanaged volunteer computing resources so far untapped by LHC experiments. One of the most important features of CernVM Co-Pilot is that it eliminates the need to modify the applications for running on resources contributed by volunteers. It opens new possibilities for outreach, and in case
of LHC@home 2.0 promotes the image of CERN as an open institution. Lastly, it allows the general public to engage in day-to-day research activities and to become a part of something they otherwise see only in the news.

References
[1] Buncic P, Sanchez C A, Blomer J, Franco L, Harutyunyan A, Mato P and Yao Y 2010 Journal of Physics: Conference Series 219 04203
[2] Amazon Elastic Computing Cloud (EC2) URL http://aws.amazon.com/ec2
[3] Anderson D 2004 proceedings of the 5th IEEE/ACM International Workshop on Grid Computing (IEEE Computer Society) pp 4–10 ISBN 0769522564
[4] Saint-Andre P 2004 Extensible Messaging and Presence Protocol (XMPP): Core RFC 3920 (Proposed Standard) URL http://www.ietf.org/rfc/rfc3920.txt
[5] Saint-Andre P 2004 Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence RFC 3921 (Proposed Standard) URL http://www.ietf.org/rfc/rfc3921.txt
[6] Sanchez C A, Blomer J, Buncic P, Chen G, Ellis J, Quintas D G, Harutyunyan A, Grey F, González D L, Marquina M, Mato P, Rantala J, Schulz H, Segal B, Sharma A, Skands P, Weir D, Wu J, Wu W and Yadav R 2011 Journal of Physics: Conference Series 331 062022
[7] Harutyunyan A, Sánchez C A, Blomer J and Buncic P 2011 Journal of Physics: Conference Series 331 062013
[8] The Erlang Jabber/XMPP daemon URL http://www.ejabberd.im
[9] Maeno T 2008 Journal of Physics: Conference Series vol 119 (IOP Publishing) p 062036
[10] Aad G, Abat E, Abdallah J, Abdelalim A, Abdesselam A, Abdinov O, Abi B, Abolins M, Abramowicz H, Acerni E et al. 2008 Journal of Instrumentation 3 S08003
[11] CernVM Co-Pilot Protocol URL http://cernvm.cern.ch/portal/copilot/documentation
[12] Condor ClassAds URL http://www.cs.wisc.edu/condor/classad/
[13] MCPLOTS repository URL http://mplots.cern.ch
[14] Simulate proton smashing with updated LHC@home URL http://arstechnica.com/science/2011/08/virtual-particles-cern-updates-lchome/
[15] Cern taps home PCs for virtual atom smashing URL http://www.zdnet.com/cern-taps-home-pcs-for-virtual-atom-smashing-3040093652/
[16] LHC@home allows public to help hunt for Higgs particle URL http://www.bbc.co.uk/news/science-environment-14488797

Figure 3. Number of registered users and hosts