University of Victoria
Faculty of Engineering
Fall 2009 Work Term Report

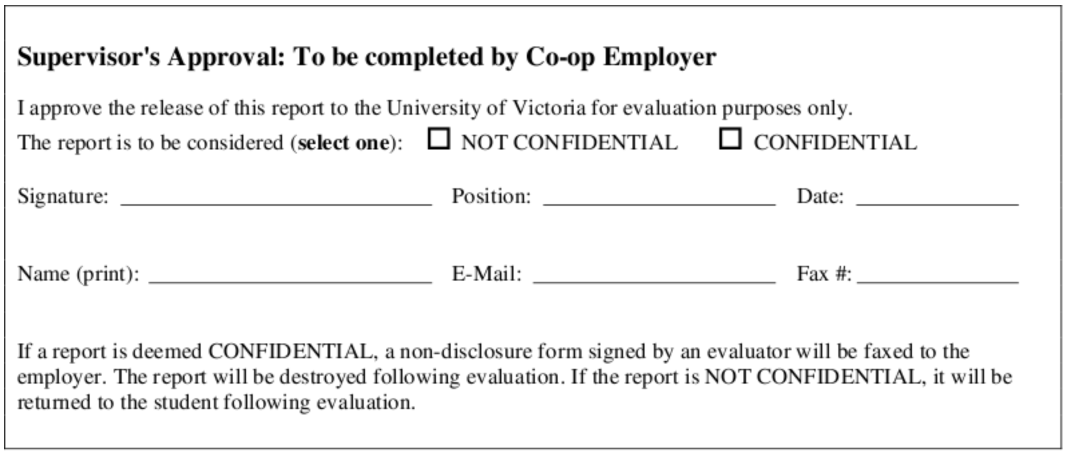
The CERN Virtual Machine and Cloud Computing

Department of Physics
University of Victoria
Victoria, BC

Vikramjit Sandhu
V00194047
Work Term 3
Computer Engineering
vssandhu@uvic.ca

January 29, 2010

In partial fulfillment of the requirements of the
Bachelor of Computer Engineering Degree



Supervisor's Approval: To be completed by Co-op Employer

I approve the release of this report to the University of Victoria for evaluation purposes only.
The report is to be considered (**select one**): ☐ NOT CONFIDENTIAL ☐ CONFIDENTIAL

Signature: ____________________ Position: ____________________ Date: ______________

Name (print): ____________________ E-Mail: ____________________ Fax #: ______________

If a report is deemed CONFIDENTIAL, a non-disclosure form signed by an evaluator will be faxed to the employer. The report will be destroyed following evaluation. If the report is NOT CONFIDENTIAL, it will be returned to the student following evaluation.

Contents

List of Figures

The CERN Virtual Machine and Cloud Computing

Vikramjit Sandhu
vssandhu@uvic.ca

January 29, 2010

Abstract

The department of Physics and Astronomy at the University of Victoria is involved in running a number of High Energy Physics (HEP) applications. These applications are dependent on specific operating system (OS) versions and have a large number of software dependencies. Hence the OS and software present at the resource may not be compatible with the needs of a particular application. This incompatibility results in resources which cannot be utilized. However, the problem can be avoided by the use of virtual machines (VMs) deployed on resources. The CERN VM, which is a virtualization software developed by a team of scientists at Geneva, is very suitable for HEP applications like ATLAS and BaBaR. The CERN VM, Nimbus software for provisioning the VMs, Condor for the execution of jobs and cloud scheduler for automating the process are used to provide a cloud computing environment at the local and remote resources. A cloud computing environment can be developed at local and remote resources. The procedure of running any HEP application, such as ATLAS, on the cloud is discussed and highlighted.

1 Report Specification

1.1 Audience

This report is intended for members of the High Energy Physics (HEP) Grid Computing Group at the University of Victoria as well as other universities and High Performance Computing (HPC) Informatrion Technology (IT) personnel.

1.2 Prerequisites

A general understanding of virtualization, distributed computing and clusters.

1.3 Purpose

Describe the steps required to configure the CERN VM on a desktop as well as a cloud cluster.

2 Introduction

HEP applications have high computational demands as well as a large number of software dependencies on the operating system (OS) version. High computational demands can be met through the use of grid computing technologies. However, the hetrogeneous nature of the resources on the grid coupled with the depnedancies of HEP applications on the OS version can result in a large number of these grid resources being unable to service an HEP application. Virtual machine monitors such as Xen can be used to package HEP applications, with their complete execution environment, to run on the resources that do not meet their OS requirements. Different HEP applications have different Operating System requirements which can be provided on VM's. Each VM runs its own operating system, thus allowing very efficient use of hardware resources. The CERN VM is one such virtual machine which is very suitable for running HEP applications

like ATLAS and BaBaR. Deploying HEP applications using Xen and Globus Virtual Workspaces has been studied and examined by Agarwal et al. [1]
Using the CERN VM, Nimbus software for provisioning the VMs, Condor and the cloud scheduler, a cloud computing environment is developed at local and remote resources. Nimbus is an open source toolkit, developed by the Globus team that allows a cluster to be turned into a virtualized cluster by installing VMs on a physical machine. Nimbus implementaion is based on the Xen hypervisor and can be configured to use different schedulers like PBS, SGE etc. to schedule virtual machines. Nimbus allows ease of lauching self configuring virtual clusters and defines an extensible architecture that allow customization of software according to the needs of an application [2]. Condor is a specialized workload management system for compute-intensive jobs. Like other full-featured batch systems, Condor provides a job queueing mechanism, scheduling policy, priority scheme, and resource execution management. Users submit their serial or parallel jobs to Condor and it places them into a queue, chooses when and where to run the jobs based upon a policy, carefully monitors their progress, and ultimately informs the user upon completion [4]. To automate the provisioning of VMs and their execution on Condor, UVIC team has developed a cloud scheduler which is utilized here. The aim of the present report is to run ATLAS jobs on a desktop CERN VM and compare its results with a conventional machine and on a cloud computing environment mentioned above.

3 The CERN Virtual Machine and Cloud Computing

3.1 What is a Virtual Machine?

VMs allow multiple Operating Systems to co-exist on the same physical machine, but in strong isolation from each other. Each VM provides a computing environment suitable for a certain type of application. The ATLAS experiment also requires a computing environment within which it is able to execute correctly.
The CERN VM is a virtualization software suitable for high energy physics (HEP) applications. It is suitable for various OS's like Windows, Mac or Linux and it is adaptable for any virtualization environment such as Xen, KVM, VMWare, Virtual Box, Qemu for Linux OS. We will only study the Xen virtualization environment in the present report.
Typical scenarios involving scientific calculations, for any one experiment, require executing a very large number of calculations (a typical figure would be 600,000 (FLOPS) calculations). Executing them on a typical workstation would require more than 3600 hours. In order to fully use the distributed resources on a cloud, a batch processing system is required. A batch processing system allows sequential execution of a series of programs run without human interaction.
The CERN VM incorporates a basic virtualization machine (VM) image suitable for desktop/laptop and batch mode environments. More effort has been put towards configuring batch mode images since this type of image provides tools to submit jobs to the grid and cloud computing clusters. Batch mode images do not provide any Graphical User Interface (GUI). CERN VM images are easy to install which makes applications such as BaBaR and ATLAS easy to run.

3.2 XEN CERN VM Batch Mode Images

XEN images are available for various architectures such as Intel x86, x86-64, Itanium and Power PC. It allows several guest operating systems to execute on the same computer hardware concurrently [5]. The CERN VM, Xen image, was first deployed on desktop/physical machine (heplw33.phys.uvic.ca) to ascertain its suitability to execute ATLAS jobs. In order to deploy the VM, the CERN VM image was first downloaded from the website [3]. After untarring and unzipping it, the image was mounted to properly cofigure it. For its proper configuration, the file system was changed into the root environment by using the 'chroot' command. Then, two users, with pseudo powers on the CERN VM, were created. These usernames could then be used for logging into the CERN VM. After making all appropriate changes, the CERN VM image was unmounted. A XEN configuration file was then created, which can be used to boot the CERN VM image on a physical machine. For booting the CERN VM image the xen monitor command or 'xm' can be

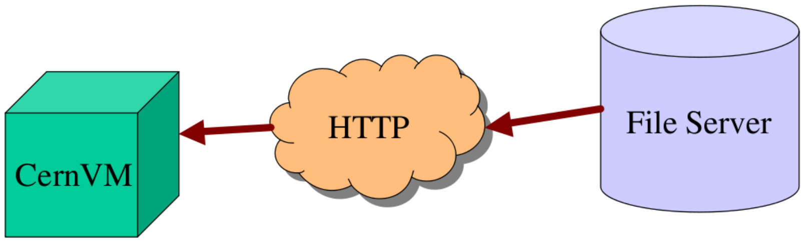
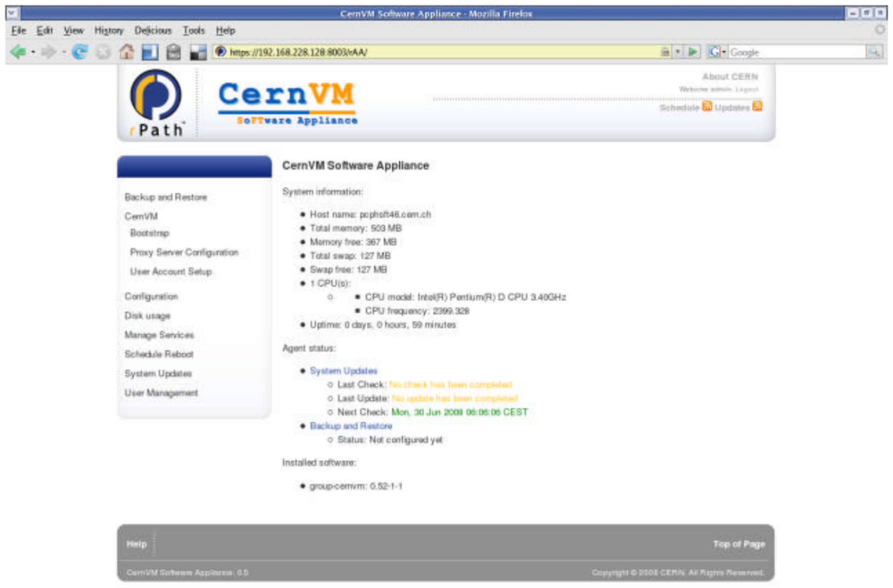


Figure 1: CERN VM File System (CVMFS)

used with the appropriate flags and the XEN configuration file as the input parameter. Once the CERN VM image was successfully booted, it is ready to run ATLAS jobs, any user can log onto the VM via ssh.
Once the machine was booted, root was used to log in. In order to allow the CERN VM to advertise as a resource for the cloud, the Condor package was then installed on it using the 'conory updateall' command. When the CERN VM is deployed on a remote cluster, all required Condor daemons must start automatically. In order to enable this the /etc/rc.local file must be edited and the line '/etc/init.d/condor' must be added. In addition, the parameter DAEMON_LIST in the the condor_config file present in the /opt/condor/etc directory must be edited to include only the Master and the Scheduler daemons. Finally the /etc/profile file must be edited to include the CONDOR_CONFIG variable. This variable must point to the location of the condor_config file i.e. /opt/condor/etc/condor_config. Condor daemons and the need to install Condor on the CERN VM have been explained in later section. Once all the changes are made to the CERN VM image, it is uploaded to the repository at http://vmrepo.phys.uvic.ca/vms/, from where it may be downloaded at any time. This repository is also needed by Nimbus to boot the VM on a physical machine. Using the above procedure, resulted in successful deployment of the CERN VM, and all necessary software required by the CERN VM to execute ATLAS jobs, on the desktop.

3.3 The CERN VM File System

The CERN VM file system (CVMFS) is illustrated in Figure 1.
It provides access to the ATLAS software by mounting the CVMFS on the image through an HTTP server. In case of any change in the software, the image can be easily updated as it is through an HTTP server. All ATLAS releases are saved locally for future access, once they are fetched. The CVMFS also provides grid and dq2 tools which make it easier to get the input files across the grid and save them locally. An ATLAS executable job is called Athena. Athena jobs can be run either on the desktop or through the grid and any output files can be stored either locally or remotely i.e. on the VM or on a machine on the grid.



CernVM Software Appliance

System information:

- Host name: pcphsft46.cern.ch
- Total memory: 503 MB
- Memory free: 367 MB
- Total swap: 127 MB
- Swap free: 127 MB
- 1 CPU(s):
  - CPU model: Intel(R) Pentium(R) D CPU 3.40GHz
  - CPU frequency: 2399.328
- Uptime: 0 days, 0 hours, 59 minutes

Agent status:

- System Updates
  - Last Check: No check has been completed
  - Last Update: No update has been completed
  - Next Check: Mon, 30 Jun 2008 06:06:06 CEST
- Backup and Restore
  - Status: Not configured yet

Installed software:

- group-cernvm: 0.52-1-1

Figure 2: CERN VM non batch mode GUI

3.4 XEN CERN VM non-Batch Mode Images

The CERN VM also has a non-batch mode version. This version provides a Graphical User Interface (GUI) which can be used for administration and configuration purposes. This version is however not suitable for runnin on the remote cluster as a cloud computing environment, having several virtual machines running on a remote cluster, do not provide user interaction. The CERN VM non batch mode GUI is illustrated in Figure 2.

3.5 Running Athena jobs on CERN VM

Once a CERN VM is successfully deployed on a desktop, an Athena job may now be executed on it. In order to run an Athena job on the CERN VM, one can log on as a user into the CERN VM. Then a test directory by the name of 'testarea' was created. The test directory is created so that any changes made during executing the Athena job do not influence any other installed software on the VM. Athena software version 15.4.0 was chosen from the /opt/atlas/software directory and a directory 15.4.0 was created under the 'testarea' directory. A correct environment for that particular release by sourcing the setupAtlasProduction_15.4.0.sh script in the /opt/atlas/software/setupScripts/ directory. Setting up the environment includes setting up the path, get all files required for execution and set up the Code Management Tool Path which sets up libraries that are part of that version of Athena. An input data file needs to be chosen next. For this test, an Aanalysis Object Data(AOD) file names AOD.e344_s479_r541_tid026925/AOD.026925._01108.pool.root.1 was chosen as the input file. A link with the shorter name, AOD.pool.root was created to the the input file name. Next, the job option file, AnalysisSkeleton_topOptions.py was obtained using the 'get_files' command. Finally, in order to figure out the time required to execute the input file chosen, on the CERN VM, the following command
-the 'time athena AnalysisSkeleton_topOptions.py'
was used to run the job. The CERN VM was installed on a machine having an Intel(R) Core(TM)2 Duo CPU running at 2.8 GHz and having 1.5 GB of memory. The desktop on which the ATLAS job was run had a Dual core AMD Opteron running at 2.1 GHz and had 8.3 GB of memeory. The following table contains a

list of the times required to execute the input file for different number of the events.

Events	CERN VM Runtime	Physical Machine Runtime
100	0m27.138s	0m35.499s
1000	0m37.370s	0m42.784s
10000	0m30.798s	0m42.495s
50000	0m31.443s	0m42.485
100000	0m31.115s	042.529ms

Looking at the values in the above table, one can see that the CERN VM actually took less time than the physical machine or desktop. This is attributed to the fact that the physical machine had a processor with a lower clock rat. It also demonstartes that the CERN VM performance has not shown any significant degradation as the execution times are very reasonable with the clock rate.

3.6 Installing VMs on a Cluster through Nimbus

The HEP application requires distributed computing resources deployed in a cloud computing environment. Cloud computing is a general term for anything that involves delivering hosted services over the internet. These services are broadly divided into three categories:

1. Infrastructure-as-a-Service (IaaS).

2. Platform-as-a-service ().

3. Software-as-a-service ().

Nimbus is an open source toolkit, developed by the Globus team, that allows a cluster to be turned into an Infrastructure-as-a-Service cloud. Nimbus allows an easy and stable mechanism for self launching virtual machines on physical machines [6]. In order to deploy a Virtual Machine over the cloud, an XML file with suitable parameters needs to be created. These parameters include the following:

1. Location of the image of the Virtual Machine which generally requires the setup of a VM repository

2. Memory required by the Virtual Machine

3. Time to live. This parameter ensures that the VM will be destroyed after a fixed time and prevents hogging of hardware resources by any one VM.

4. The type of network that the VM will have i.e. private/public. A private netwok cannot be accessed from the outside world whereas all VMS with a public network can be accessed rfom the outside world.

The eXtensible Marklup Language (XML) file, used by the Nimbus client and the shell script to create a workspace can be found in the Appendix. Users can log onto the VM's using ssh to the IP address of the VM, provided by the Nimbus Client. When the VM is not needed, the following workspace command can be used to destroy the image. To shutdown a VM the command
-workspace -e 123456789.eps –shutdown or
-workspace -e 123456789.eps –destroy can be used.

3.7 Setup of Condor Server for Job Execution

Condor is a specialized workload management system for compute-intensive jobs. Like other full-featured batch systems, Condor provides a job queueing mechanism, scheduling policy, priority scheme, resource monitoring, and resource execution management. Users submit their serial or parallel jobs to Condor and Condor places them into a queue, chooses when and where to run the jobs based upon a policy, carefully monitors their progress, and ultimately informs the user upon completion [7]. A Condor pool consists of a number of worker node machines. Each machine determines its own policy for running jobs. In order to

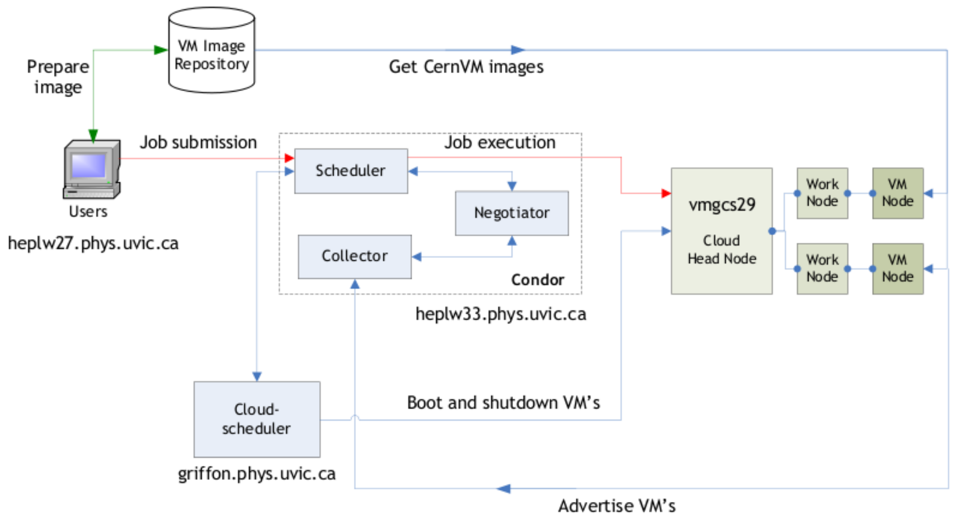


Figure 3: ATLAS Cloud Computing Model

successfully run a job, it needs a machine that matches it. The interaction between Nimbus and Condor is illustrated in Figure 3. Jobs and machines state their requirements and preferences and Condor is responsible for matching a job with a machine.
Typically a job's requirements consist of specifying the operating system, the architecture of the underlying platform like x86/x86_64, the memory required etc. A machine's typical requirements may consist of running jobs only when there is no keyboard activity and not running jobs belonging to a particular user. The above requirements do not specify an exhaustive list, rather this would depend on different jobs/machines.

3.8 Setting up Cloud Computing for ATLAS

In order to set up the framework to execute ATLAS jobs on a cloud cluster, Condor and cloud software are installed on the following physical machines.

1. The Condor Client set up on the desktop (heplw27.phys.uvic.ca)

2. The Condor Server set up on a different machine(heplw33.phys.uvic.ca)

3. The Cloud Scheduler set up on a different machine (griffon.phys.uvic.ca)

The cloud resource is set up on the headnode vmcgs29.phys.uvic.ca which has two worker nodes, cgs03.phys.uvic.ca and cgs04.phys.uvic.ca. The schematic diagram of the ATLAS cloud computing model is illustrated in Figure 3. The Condor client and the Condor server differ only in the daemons running on the machines on which they are installed. These daemons are responsible for passing information between the Condor client, the

Condor Server and the CERN VM. The Condor client is set up so that only master and scheduler daemons are run on it. The Condor server is set up so that it has the master, collector, negotiator and scheduler daemons running on it (see Figure 3). The list of daemons that are to be started in the client or the server can be specified in the file /opt/condor/etc/condor_config. The attribute 'DAEMON_LIST' allows control over daemons are to be started. The Condor Client is a machine on which the user may submit jobs to the Condor Server. The condor_submit command may be used to submit a job. The '-name' option is used to ensure that the job is submitted to the scheduler queue on the server and not to the server queue on the client. The condor-submit command looks as follows:

```
-condor_submit -name heplw33.phys.uvic.ca job_file.jdl
```

The above command allows a job description file to be submitted to the Condor Server installed on the machine with hostname heplw33.phys.uvic.ca. The parameter 'CONDOR_HOST' must point to the machine on which the server is installed. For this test case scenario, it was set to 'heplw33.phys.uvic.ca'.
Once the Job is submitted, it reaches the the condor server where it is scheduled. The job queue is examined regularly by the cloud-scheduler. The job queue can be examined at any time using the command condor_q. When the cloud scheduler notices jobs in the queue, it boots the VMs on the worker nodes which in turn advertise themselves to the condor collector. The Cloud Scheduler is a python script which uses Nimbus to boot the VMs on a remote cluster. The name of the remote cluster and other information like the architecture and memory of the worker nodes is contained in a file which is passed as an argument to the Cloud Scheduler. When a VM is booted on a remore cluster, it automatically starts condor, since the condor start up script was included in the /etc/rc.local file, when the CERN VM was set up. This enables the VM to advertise its resources to the condor server. The condor_config file is central to configuring the condor server as well as the client. In the /etc/profile file on both the client as well as the server machine, variable CONDOR_CONFIG must point to the location of the condor_config file. In this test scenario, on both the server as well as the client, CONDOR_CONFIG was set to /opt/condor/etc/condor_config. Moreover in the condor_config file itself, the parameters HOSTALLOW_WRITE and HOSTALLOW_READ must both be set to accept connections from all virtual machines. This may be done by assigning a '*' value to both of them. The condor negotiator keeps watching the collector for resources. Once the negotiator finds the resources, it matches the job requirements with the resource and instructs the scheduler to execute the job. If the VM is not needed and time to live expires, the cloud scheduler destroys the VMs.

3.9 Analysing a Condor Job Submission File

Condor uses a class-ad mechanism to match jobs with their requirements. Jobs want to find machines upon which they can execute. A job will require a specific platform on which to execute. Machines have specific resources available, such as the platform and the amount of available memory. A separate ClassAd is produced for each job and machine, listing all attributes. The Negotiator acts as a matchmaker between the jobs and the machines by pairing the ClassAd of a job with the ClassAd of a machine [8]. The requirements of the job and the machine are both specified in the job description file. For test purposes, a job description file, 'blue_atlas01.sub', was created and submitted using condor. The job description file contains a list of the requirements of the job i.e. the environment required by the job in order to successfully execute. These requirements contain the name of the virtual machine and the architecture, amongst other things. The same job description file has an 'Executable' tag that specifies the script that represents the actual executable job. In order to run an ATLAS job, the required script containing the steps to run an ATLAS job was created and the 'Executable' tag in the job description file was made to point towards it. The 'Log', 'Output' and 'Error' tags represent the log, output and error files respectively. The job description file used in this scenario has been included in the appendix. Following the above steps an ATLAS job was successfully executed on the remote cluster with head node on machine with hostname vmcgs29.phys.uvic.ca.

4 Conclusion

HEP applications with high computational demands can be serviced through the deployement of virtual machines on hetrogeneous grid resources. The CERN VM is one such virtual machine which has proven to be a useful and stable for running ATLAS jobs. The execution time for ATLAS jobs is found to be comparable with that of a physical machine and does not show any degradation in performance. The CERN VM in conjunction with Nimbus, required to boot VMs on resources, Condor for execution management and the cloud scheduler for automating the process are found to be highly useful to provide a stable cloud computing environment.
Nimbus is extremely efficient and reliable in booting VMs on a loacl or remote resoource; it provides an easy mechanism to create a workspace and later on to kill it, when it is not required any more. Condor has also proven to be very reliable as an execution management system. The class-ad mechanism provided by Condor effectively matches a resource with a job's requirements. The cloud scheduler developed by the UVIC team is robust and useful. Non-batch mode Cern VM images provide development and user analysis on the Desktop/laptop. The CVMFS saves disk space and is much faster than the Andrew File System (AFS).

5 Future Work

Work still needs to be done on the CERN VM running on a remote resources such as the NRC and Mercury clouds. More performance tests should be done to compare the results on different clouds. These tests would then determine the usefullness of running the CERN VM on remote clouds in order to provide a HTC environment.
So far Ganga and Panda jobs submission of ATLAS jobs and storage resource management has not been tried, but must be tested on the cloud resources. The grid tools and glite software available in batch mode images can be used for setting up Tier 3 ATLAS analysis site.

6 Acknowledgments

I would like to thank Dr. Randall Sobie for this work term opportunity. Thanks to Dr. Ashok Agarwal for guidance on this work term. Additional thanks Patrick Armstrong, Ian Gable, Ron Desmarais, Duncan-Penfold Brown, Michael Patterson, Chris Tooley, Colin Brown and all members of the research computing facility

7 Appendix

7.1 XML file used by Nimbus script

```xml
<?xml version="1.0" encoding="UTF-8"?>
<VirtualWorkspace
    xmlns="http://www.globus.org/2008/06/workspace/metadata"
    xmlns:def="http://www.globus.org/2008/06/workspace/metadata/definition"
    xmlns:log="http://www.globus.org/2008/06/workspace/metadata/logistics"
    xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >

    <!-- name is an arbitrary URL -->
    <name>http://example1/hepgc/sl53image</name>

    <log:logistics>
        <log:networking>
            <log:nic>
                <log:name>eth0</log:name>
                <log:ipConfig>
                    <log:acquisitionMethod>AllocateAndConfigure</log:acquisitionMethod>
                </log:ipConfig>

                <!--
                The association string allows you to associate the NIC with
                specific networking contexts (a common example is a string
                which resolves to which network to bridge the virtual NIC
                to; a simple scheme would be 'public' vs. 'private'.
                Another example is VPN).  A site will advertise the
                available assocations in the workspace factory resource
                properties (to query with the sample client, use the
                factoryrp option).
                -->
                <log:association>private</log:association>
            </log:nic>
        </log:networking>
    </log:logistics>

    <def:definition>
        <def:requirements>
            <jsdl:CPUArchitecture>
                <jsdl:CPUArchitectureName>x86</jsdl:CPUArchitectureName>
            </jsdl:CPUArchitecture>
            <def:VMM>
                <def:type>Xen</def:type>
                <def:version>3</def:version>
            </def:VMM>
        </def:requirements>
        <def:diskCollection>
            <def:rootVBD>
                <!--
                Relative path names like in this example will be resolved
                relative to the deployment node's local image repository
                -->
                <def:location>http:/vmrepo.phys.uvic.ca/vms/blue.img.gz</def:location>
                <def:mountAs>sda</def:mountAs>
                <def:permissions>ReadWrite</def:permissions>
            </def:rootVBD>
        </def:diskCollection>
    </def:definition>
</VirtualWorkspace>
```

7.2 shell script used by Nimbus to craete the workspace

```
#!/bin/sh
$GLOBUS_LOCATION/bin/workspace
-z none
–poll-delay 200
–deploy
–file ‘date +%s‘.epr
–metadata ./blue.xml
–trash-at-shutdown
-s https://canfardev.dao.nrc.ca:8443/wsrf/services/WorkspaceFactoryService
–deploy-duration 1400 –deploy-mem 512 –deploy-state Running
–exit-state Running
```

7.3 Job description file used to submit an ATLAS job on a remote cluster

```
# Condor job submission file for Cloud Scheduler Testing

   Universe = vanilla
Executable = test.sh
Arguments = W3-3+3 W3
Log = blue.log
Output = blue.out
Error = blue.error
should_transfer_files = YES
when_to_transfer_output = ON_EXIT
Requirements = (VMType =?= AtlasTest) && (Arch == "INTEL")

   +VMName = "TestVM03-Blue"
+VMNetwork = "public"
+VMCPUArch = "x86"
+VMLoc = "http://vmrepo.phys.uvic.ca/vms/cernvm-devel-1.3.4-x86-root.ext3"
+VMMem = "1624"
```

8 Glossary

VM	Virtual Machine.
CERN	European Organization for Nuclear Research.
HEP	High Energy Physics.
OS	Operating System.
VM	Virtual Machine, an instance of a machine(computer) running in software.
VMM	Virtual Machine Monitor, used for managing virtual machines.
Xen	Open-source VMM used by Nimbus.
XML	eXtensible Markup Language.

References

[1] Agarwal A. et al., Deploying HEP Applications Using Xen and Globus Virtual Workspaces. Proccedings of Computing in High Energy Physics 2007, Victoria, Canada. J.Phys. Conf. Ser. 119062002 (8pp)

[2] Nimbus `http://workspace.globus.org/`

[3] CERN VM download `http://rbuilder.cern.ch/project/cernvm/build?id=841`

[4] Condor `http://www.cs.wisc.edu/condor/description.html`

[5] XEN `http://en.wikipedia.org/wiki/Xen`

[6] Nimbus Cloud Computing `http://workspace.globus.org/`

[7] Condor Cloud Computing `http://www.cs.wisc.edu/condor/description.html`

[8] Condor Overview `http://www.cs.wisc.edu/condor/overview/`