Research computing in a distributed cloud environment

K. Fransham¹, A. Agarwal¹, P. Armstrong¹, A. Bishop¹, A. Charbonneau², R. Desmarais¹, N. Hill³, I. Gable¹, S. Gaudet³, S. Goliath³, R. Impey², C. Leavett-Brown¹, J. Ouellet³, M. Paterson¹, C. Pritchett¹, D. Penfold-Brown¹, W. Podaima², D. Schade³, R.J. Sobie¹, ⁴

¹ Department of Physics and Astronomy, University of Victoria, Victoria, Canada V8W 3P6
² National Research Council Canada, 100 Sussex Drive, Ottawa, Canada
³ National Research Council Herzberg Institute of Astrophysics, 5071 West Saanich Road, Victoria, BC, Canada, V9E 3E7
⁴ Institute of Particle Physics of Canada.

E-mail: fransham@uvic.ca

Abstract. The recent increase in availability of Infrastructure-as-a-Service (IaaS) computing clouds provides a new way for researchers to run complex scientific applications. However, using cloud resources for a large number of research jobs requires significant effort and expertise. Furthermore, running jobs on many different clouds presents even more difficulty. In order to make it easy for researchers to deploy scientific applications across many cloud resources, we have developed a virtual machine resource manager (Cloud Scheduler) for distributed compute clouds. In response to a user’s job submission to a batch system, the Cloud Scheduler manages the distribution and deployment of user-customized virtual machines across multiple clouds. We describe the motivation for and implementation of a distributed cloud using the Cloud Scheduler that is spread across both commercial and dedicated private sites, and present some early results of scientific data analysis using the system.

1. Introduction

Infrastructure as a Service (IaaS) cloud computing is emerging as a new and efficient way to provide computing to the research community. Clouds are considered to be a solution to some of the problems encountered with early adaptations of grid computing where the site retains control over the resources and the user must adapt their application to the local operating system, software and policies. This often leads to difficulties, especially when a single resource provider must meet the demands of multiple projects or when projects cannot conform to the configuration of the resource provider. IaaS clouds offer a solution to these challenges by delivering computing resources using virtualization technologies. Users lease the resources from the provider and install their application software within a virtual environment. This frees the providers from having to adapt their systems to specific application requirements and removes the software constraints on the user applications. In most cases, it is easy for a user or a small project to create their virtual machine (VM) images and run them on IaaS clouds. However, the complexity rapidly increases for projects with large user communities and significant computing
requirements. In this paper we describe a system that simplifies and combines multiple IaaS clouds into a single distributed cloud for High Throughput Computing (HTC) workloads.

The growing interest in clouds can be attributed in part to the ease in encapsulating complex research applications in Virtual Machines (VMs), often with little or no performance degradation [1]. Studies have shown, for example, that particle physics application code runs equally well in a VM as on the native system [2]. Today, open source virtualization software such as Xen [3] and KVM [4] are incorporated into many Linux operating system distributions, resulting in the use of VMs for a wide variety of applications. Often, special purpose servers, particularly those requiring high availability or redundancy, are built inside a VM making them independent of the underlying hardware and allowing them to be easily moved or replicated. It is also not uncommon to find an old operating system lacking the drivers for new hardware, a problem which may be resolved by running the old software in a virtual environment.

The deployment and management of many VMs in an IaaS cloud is labour intensive. This can be simplified by attaching the VMs to a job scheduler and utilizing the VMs in a batch environment. The Nimbus Project [5] has developed the one-click cluster solution. This provides a batch system on multiple clouds using one type of VM [6].

We further simplify the management of VMs in an IaaS cloud and provide new functionality with the introduction of Cloud Scheduler. Cloud Scheduler provides a means of managing user-customized VMs on any number of science and commercial clouds1.

In the following sections we present the architecture of our system and highlight the role of Cloud Scheduler to manage VMs on IaaS clouds in a batch HTC environment. We present early results on the operation, and highlight the successes of the system.

2. System architecture

The architecture of the HTC environment running user-customized VMs on distributed IaaS clouds is shown in fig. 1. A user creates their VM and stores it in the VM image repository. They write a job script that includes information about their VM and submits it to the Condor Job Scheduler. The Cloud Scheduler reads the queues of the Condor Job Scheduler, requests that one of the available cloud resources boot the user VM, the VM advertises itself to the Condor Job Scheduler which then dispatches the user job to that VM. In this section, we describe the components of the system and the method used to manage VMs.

2.1. System components

In this section we describe the following components: VM image repository, the cloud resources, the Condor Job Scheduler and the Cloud Scheduler.

VM image repository

The user builds their VM by first retrieving a base image from a VM image repository. The base images are simple Linux VM images or images that include project or application based code. Once the user has modified their image, they will store it back in the VM image repository. The repository may reside at a single site or be distributed, however, it must be accessible to the cloud resources.

Cloud resources

The system currently supports Amazon EC2 and IaaS clouds using Nimbus [5]. Support for IaaS clouds using OpenNebula [7] and Eucalyptus [8] is under development.

Job Scheduler

The job scheduler used in the system is the Condor HTC job scheduler [9]. Condor was

1 Commercial cloud providers include Amazon, RackSpace and IBM, to name a few. Science clouds use hardware resources funded by governments for research purposes and are located in universities or national laboratories.
Figure 1. An overview of the architecture used for the system. A user prepares their VM image and a job script. The job script is submitted to the Condor Job Scheduler. The Cloud Scheduler reads the job queue and makes a request to boot the user VM on one of the available clouds. Once there are no more user jobs requiring that VM type, the Cloud Scheduler makes a request to the proper cloud to shutdown the user VM.

chosen because it was designed to utilize heterogeneous idle workstations which makes it ideal to use as a job scheduler for a dynamic VM environment. Condor has a central manager which matches user jobs to resources based on job and resource attributes. The Condor startd daemon must be installed and started when a VM image is booted. The VM then advertises its existence to the Condor central manager. Users submit jobs by issuing the condor_submit command. The user must add a number of additional parameters specifying the location and properties of the VM. The description of the parameters is found in section 4.

Cloud Scheduler

The Cloud Scheduler is an object oriented python-based package designed to manage VMs for jobs based on the available cloud resources and job requirements. Users submit jobs to the Condor Job Scheduler after they have been authenticated using X.509 Proxy Certificates [10]. The certificates are also used to authenticate starting, shutting down, or polling VMs.

2 We use Condor Connection Brokering (CCB) to allow VM worker nodes that use Network Address Translation (NAT) to connect to the Condor Central Manager.
with Nimbus clusters. Authentication with EC2 is done by using a standard shared access key and secret key. The Cloud Scheduler then communicates with the IaaS clouds available to it, and manages the deployment and shutdown of VMs to run the jobs in the Condor Job Scheduler queue.

3. VM management
When Cloud Scheduler is started it reads configuration files that describe the available cloud resources. To schedule user jobs across the available clouds, the Cloud Scheduler starts the following threads that are run on a periodic basis.

(i) The JobPoller thread maintains the state and metadata of the jobs that are queued and running on the Condor Job Scheduler. It effectively maps the Condor Job Scheduler queue into the JobPool.

(ii) The Scheduler thread starts and stops VMs based on the information in the JobPool, satisfying the resource demands of the workload. The design goal of Cloud Scheduler is to leave prioritization and scheduling decisions in the domain of the Condor Job Scheduler. However, the order in which the Scheduler thread provides resources can impact the scheduling algorithms of the Condor Job Scheduler. The Scheduler thread also monitors the VMs and, if necessary, updates the state of VMs in the JobPool. It will shut down VMs that are in an error state; if there are jobs that still require this VM, then the Scheduler will start a new instance of the VM to replace the one it has shut down.

(iii) The CleanUp thread stops VMs that are no longer required. It can correct the state of the job in the JobPool. If a VM is shut down due to an error, then the CleanUp thread changes the state of the job in the JobPool from “scheduled” to “new” so that a new VM can be created for that job.

When Cloud Scheduler is shut down, it can either shut down all the VMs that is has started, or it can persist its state. In the latter case, the VMs continue to run the jobs. Cloud Scheduler reloads the state when it is restarted and resumes managing the jobs and resources.

4. User Interaction
The users of the distributed cloud are typically researchers who have extensive knowledge of job submission to traditional HTC batch systems. The goal of the system developed around the Cloud Scheduler is to maintain as much of the user’s existing workflow as possible. To that end, the user’s job submission script follows the traditional Condor format, with some extra parameters required to configure the user’s VM. A sample job script is shown below, and a description of the additional Cloud Scheduler attributes follows in table 1.
Regular Condor Attributes
Universe = vanilla
Executable = script.sh
Arguments = one two three
Log = script.log
Output = script.out
Error = script.error
should_transfer_files = YES
when_to_transfer_output = ON_EXIT
#
# Cloud Scheduler Attributes
Requirements =
+VType = "vm-name"
+VMLoc = "http://repository.tld/your.vm.img.gz"
+VMAMI = "ami-dfasdfs"
+VMPA = "x86"
+VMCPUCores = "1"
+VMNetwork = "private"
+VMMem = "512"
+VMStorage = "20"
Queue

Table 1. Description of customizable attributes required by the Cloud Scheduler

| Attribute     | Description                     |
|---------------|---------------------------------|
| VMTyp         | Unique name of required VM      |
| VMLoc         | URL (Nimbus) of the image       |
| VMAMI         | AMI (EC2-like) of the image     |
| VMPA          | CPU architecture (x86 or x86_64)|
| VMCPUCores    | Number of CPU cores             |
| VMStorage     | Required storage space          |
| VMMem         | RAM required                    |
| VMNetwork     | Network required (public/private)|

5. Results
Currently, a distributed cloud employing the architecture described above is being used to generate simulated data for the BaBar particle physics experiment. This distributed cloud is managed by the Cloud Scheduler, and can boot up to 84 simultaneous virtual machines on three different cloud sites: the system has been configured to have access to 24 CPU cores on a Nimbus-based at the National Research Council (NRC) in Ottawa, Canada; 40 CPU cores on a Nimbus-based cloud at the University of Victoria (UVic) in Victoria, Canada; and 20 cores on Amazon EC2. The Cloud Scheduler, Condor Job Scheduler, and VM Image Repository are all located at the NRC.

3 Based at the SLAC National Accelerator Laboratory in Stanford, California, the BaBar experiment collides high-energy electrons and positrons, and studies the results of the collision in order to gain insight into the properties of exotic sub-atomic particles.
A VM image containing the BaBar simulation application has been created, and is stored in the VM Image Repository. The application itself contains C++ and FORTRAN code, requires 1GB or more of RAM, and is completely contained in a VM approximately 16 GB in size. The BaBar application contains scripts to interface the production of simulated data with a local batch scheduler; in order to run on the distributed cloud, the only change to the software that was required was to point the simulation production manager to the cloud’s Condor Job Scheduler. As described in section 4, a stated goal of the distributed cloud system was to avoid altering the cloud user’s ordinary workflow as much as possible. In the case of Babar simulation production, this goal was certainly achieved, since any user familiar with running the simulation production application is able to use the same commands to run it on the distributed cloud.

When the system is functioning correctly, the Cloud Scheduler boots VMs and jobs are run on all of the available clouds sites available. Figure 2 shows the total number of completed jobs on the three different cloud sites over the course of a week. The increase in completed jobs from each of the three sites over the course of the week shows that the system is functioning as expected: the Cloud Scheduler has turned the three unique IaaS clouds into a single distributed cloud. The flat sections in the curves correspond to times when there were no jobs in the queue.

Furthermore, the BaBar collaboration places strict checks on the quality of any produced simulated data. The data that continues to be produced on the distributed cloud has passed all quality checks, further demonstrating the robustness and usefulness of the system.

The Cloud Scheduler is also a central component of a different distributed cloud used by CANFAR [14], an astronomy project led by researchers at the University of Victoria and the

Figure 2. A comparison of the number of jobs that ran to completion at all three cloud sites in the distributed cloud. UVic, having the largest number of CPUs available, ran the most jobs over the course of the week.
National Research Council of Canada Herzberg Institute of Astrophysics. The goal is to provide researchers the ability to create custom environments for analyzing data from survey projects. Currently the system is available to early adopters, using two clusters: one at the WestGrid facility located at the University of Victoria (25 machines, 200 cores) and one at the Herzberg Institute of Astrophysics (6 machines, 32 cores). The system has been tested successfully with over 9,000 jobs utilizing more than 33,000 core hours. Work is currently underway to streamline the process of creating and sharing virtual machines with researchers, to allow them to easily make VM images that suit their needs and ready for deployment on cloud resources.

6. Summary
We have presented a new method for running large scale complex research applications across multiple IaaS computing clouds. The development of the Cloud Scheduler is central to the distributed cloud, as it simplifies the management of virtual machine resources across multiple science and commercial clouds [15] by hiding the complexity of VM management. We have demonstrated that the system works for both astronomy and particle physics applications, using multiple cloud resources.

The support of CANARIE, the Natural Sciences and Engineering Research Council, the National Research Council of Canada and Amazon are acknowledged.

References
[1] M. Alef and I. Gable. HEP specific benchmarks of virtual machines on multi-core CPU architectures. J. Phys Conf. Ser. 219, 052015 (2008). doi: 10.1088/1742-6596/219/5/052015
[2] A. Agarwal, A. Charbonneau, R. Desmarais, R. Enge, I. Gable, D. Grundy, D. Penfold-Brown, R. Seuster, R.J. Sobie, and D.C. Vanderster. Deploying HEP Applications Using Xen and Globus Virtual Workspaces. J. Phys.: Conf. Ser. 119, 062002 (2008). doi: 10.1088/1742-6596/119/6/062002
[3] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt and A. Warfield. Xen and the art of virtualization. SOSPO3: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles.
[4] KVM (Kernel Based Virtual Machine). http://www.linux-kvm.org/page/Main_Page.
[5] K.Keahey, I.Foster, T.Freeman, and X. Zhang. Virtual workspaces: Achieving quality of service and quality of life in the Grid. Sci. Program. Vol. 13 (2005) 265.
[6] K.Keahey and T.Freeman. Contextualization: Providing One-Click Virtual Clusters. SCIENCE08: Proceedings of the 2008 Fourth IEEE International Conference on eScience.
[7] Open Nebula (Open Source Toolkit for Cloud Computing). http://www.opennebula.org/.
[8] D. Nuri, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. The Eucalyptus Open-Source Cloud-Computing System. CCGRID ’09: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid. doi:10.1109/CCGRID.2009.93
[9] D.Thani, T.Tannebaum and M.Livny. Distributed computing in practice: the Condor experience. Concurrency and Computation: Practice and Experience Vol. 17 (2005) 323.
[10] S.Tuecke, V.Welch, D.Engert, L.Pearlman, and M.Thompson. Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile. http://www.ietf.org/rfc/rfc3820.txt.
[11] A. Agarwal, M. Ahmed, A. Berman, B.L. Caron, A. Charbonneau, D. Deatrich, R. Desmarais, A. Dimopoulos, I. Gable, L.S. Groer, R. Haria, R. Impye, L. Klektau, C. Lindsay, G. Mateescu, Q. Matthews, A. Norton, W. Podaima, D. Quesnel, R. Simmonds, R.J. Sobie, B.St. Arnaud, C. Usher, D.C. Vanderster, M. Vetterli, R. Walker, M. Yuen. GridX1: A Canadian computational grid. Future Generation Computer Systems 23 (2007) 680. doi:10.1016/j.future.2006.12.006
[12] A. Agarwal, P. Armstrong, R. Desmarais, I. Gable, S. Popov, S. Ramage, S. Schaffer, C. Sobie, R.J. Sobie, T. Sullivan, D.C. Vanderster, G. Mateescu, W. Podaima, A. Charbonneau, R. Impye, M. Viswanathan, D. Quesnel, BaBar MC Production on the Canadian Grid using a Web Services Approach. J. Phys.: Conf. Ser. 119 (2007) 072002. doi: 10.1088/1742-6596/119/7/072002
[13] CANARIE. Canada’s Advanced Research and Innovation Network. http://www.canarie.ca.
[14] CANFAR. Canadian Advanced Network For Astronomical Research. http://astrowww.phys.uvic.ca/canfar/
[15] The Cloud Scheduler code is available at http://cloudscheduler.org.