Journal of Physics: Conference Series **331** (2011) 062010 doi:10.1088/1742-6596/331/6/062010

A batch system for HEP applications on a distributed IaaS cloud

I. Gable, A. Agarwal, M. Anderson, P. Armstrong, K. Fransham, D. Harris C. Leavett-Brown, M. Paterson, D. Penfold-Brown, R.J. Sobie, M. Vliet

Department of Physics and Astronomy
University of Victoria, Victoria, Canada V8W 3P6

A. Charbonneau, R. Impey, W. Podaima

National Research Council Canada
100 Sussex Drive, Ottawa, Canada

E-mail: igable@uvic.ca

Abstract. The emergence of academic and commercial Infrastructure-as-a-Service (IaaS) clouds is opening access to new resources for the HEP community. In this paper we will describe a system we have developed for creating a single dynamic batch environment spanning multiple IaaS clouds of different types (e.g. Nimbus, OpenNebula, Amazon EC2). A HEP user interacting with the system submits a job description file with a pointer to their VM image. VM images can either be created by users directly or provided to the users. We have created a new software component called Cloud Scheduler that detects waiting jobs and boots the user VM required on any one of the available cloud resources. As the user VMs appear, they are attached to the job queues of a central Condor job scheduler, the job scheduler then submits the jobs to the VMs. The number of VMs available to the user is expanded and contracted dynamically depending on the number of user jobs. We present the motivation and design of the system with particular emphasis on Cloud Scheduler. We show that the system provides the ability to exploit academic and commercial cloud sites in a transparent fashion.

1. Introduction

Grid computing, although successful for many High Energy Physics(HEP) projects, has some well understood disadvantages [1] when compared to traditional cluster High Throughput Computing (HTC). On of the more difficult to overcome is the need to deploy applications to remote resources. Applications written for HEP experiments are complex software systems usually requiring an application specialist to make the deployment. Often it can take significant amounts of time to install and configure the software on a new cluster, making it difficult to quickly take advantage of computing resources as they become available.

Virtual Machine software such as Xen [2] and KVM [3] can be used to wrap the application code in a complete software environment [1]. Both Xen and KVM have been shown to present insignificant execution overhead for HEP applications [4]. VMs can be copied and deployed at various remote Infrastructure-as-a-Service (IaaS) cloud computing platforms, such as Nimbus

```
# 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 =
+VMType        = "vm-name"
+VMLoc         = "http://repo.tld/vm.img.gz"
+VMAMI         = "ami-dfasfds"
+VMCPUArch     = "x86"
+VMCPUCores    = "1"
+VMNetwork     = "private"
+VMMem         = "2048"
+VMStorage     = "20"
Queue
```

Figure 1. A typical Condor Job description file used with Cloud Scheduler. There are several custom attributes required which define the type of VM image required to run the job.

[5], OpenNebula [6], Eucalyptus [7]. However, having efficient VMs and the means to deploy them to remote IaaS sites is not enough to make a usable platform for a typical HEP user with many jobs to run. Most HEP users are already familiar with using cluster batch computing as it has been a technology employed for at least the last decade. However, few of these users have experience constructing a batch system as this is typically a task handled by a site system administrator. We have implemented a dynamic multi-cloud site batch system using a simple software component we developed called Cloud Scheduler [8]. Our goal is to produce a system that requires little effort from the users than typical batch job submissions. In this paper we describe this system and show how it can be used effectively with the BaBar application software [9].

2. System Architecture

Cloud resources are typically provided at discrete sites with no mechanism for bridging multiple sites or submitting to a central resource broker. In order to use multiple sites, requests for VMs must be individually submitted to each site. As the number of sites grows this task becomes increasingly onerous for the users. We solve this problem by building Condor [10] resource pools across multiple clouds. The component which allows us to accomplish this is the aforementioned Cloud Scheduler (for a detailed description of Cloud Scheduler's internal design see [8]). Cloud Scheduler is configured with a simple text file that list the available cloud resources and their properties such as available CPU architectures, number of available VM slots, and memory per slot.

Users are provided with a VM with an unmodified installation of an operating system or one configured with project-specific HEP application software. This serves both typical and sophisticated users who wish to substantially modify their application software. Users create a valid X.509 grid proxy [13], then submit a Condor job with a set of custom Condor attributes (see Figure 1). The Condor attributes define the type of VM image which their job requires

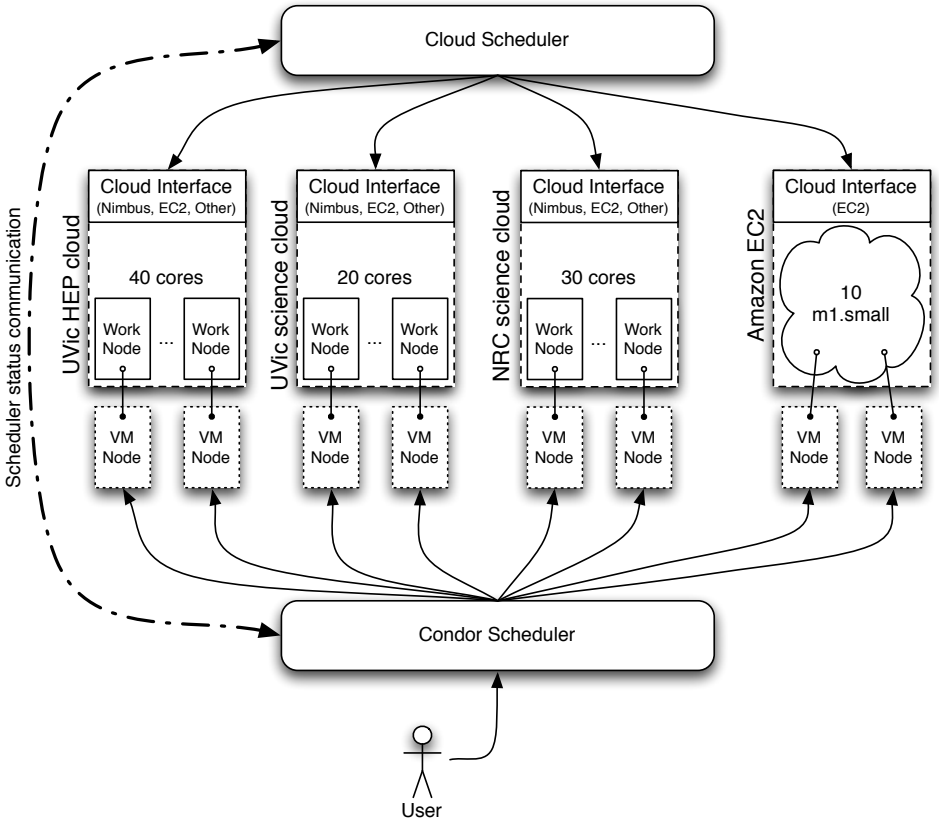


Figure 2. The figure shows the distributed cloud system used in this work. The number of VM slots (or cores) at each site shown on the figure is the number made available to this project.

in addition to other requirements such as memory, scratch space, and CPU architecture. Once submitted to Condor, Cloud Scheduler detects the waiting idle job in the queue and boots a virtual machine on any number of configured IaaS cluster types such as Nimbus, Amazon EC2, and Eucalyptus. Prior to booting, the VM is contextualized (modified by the IaaS software) to connect back to the central Condor Scheduler to begin draining jobs. Users initially see a Condor Scheduler with no available resources and then VMs spring to life that meet the requirements of their jobs. When all jobs are completed the idle VMs are shutdown.

In this work we use a system consisting of four clouds as shown in Figure 2. Both the UVic HEP cloud and UVic science cloud are located in Victoria, Canada. The NRC cloud is located at the National Research Council in Ottawa, Canada, and Amazon EC2 is located in Virginia. All sites are connected with 1 Gbps or better research network connections except Amazon EC2 which is reached via a slower and more congested commodity internet connection. The slower network connection limits the use of EC2 for applications that require access to databases or data sets that are located at remote locations. Each one of the three research cloud sites consists of a Linux cluster with Intel Nahalem Xeon processsors and Gigabit Ethernet or bonded Gigabit Ethernet connections. These clusters are typical of those purchased for HEP HTC clusters. The three research sites use Nimbus and are accessed by the Nimbus IaaS API. Sites provided by Amazon are accessed via its REST API. Each Nimbus site uses X.509 grid proxies for authentication, which allows HEP users to reuse their existing grid credentials. Grid credentials are also well trusted means of authentication at other research sites. Amazon EC2

uses its own access control and secret keys for authentication.

3. Results

We present a analysis run comprised of 255 jobs submitted by a BaBar user who is a member of this project. The submitted jobs run a user analysis over three BaBar data sets: Tau1N-data (total size: 1158 GB), Tau1N-MC (total size: 615 GB) and Tau11-MC (total size: 1386 GB). Tau1N-data and Tau1N-MC have event sizes of 4 KB while Tau11-MC has event size of 3 KB. All the data is stored on a Lustre filesystem hosted at UVic and is accessed by the VMs through the Xrootd [12] protocol.

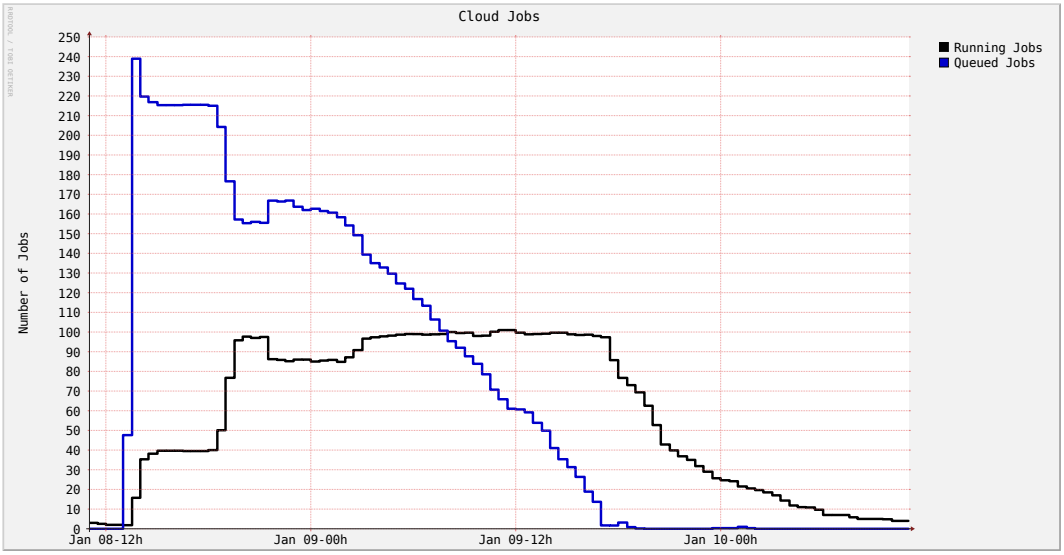
Prior to the submission the user prepared a Xen VM image containing their BaBar application and made it available on an http server at NRC in Ottawa. This server functions as the central virtual machine image repository which can be accessed by any one of the research clouds shown in Figure 2. In addition the VM image must be uploaded to Amazon EC2 as Amazon requires that images preexist on their cloud. Each VM image is roughly 16 GB in size. At present there is no user authentication on the central VM repository, however we have an image repository under development which can use X.509 proxies to authenticate over https.

Figure 3(a) shows the jobs submission occurring at 13:00 on Jan 8 and 255 jobs entering the batch queue. Initially no jobs are able to run because no suitable resources have yet been instantiated in the cloud. Cloud Scheduler detects the waiting jobs and begins to instantiate the necessary VMs to meet the requirements of the jobs as listed in the Condor job description files (see Figure 1). We initially see VMs rapidly starting on both the NRC cloud and EC2 as shown in Figure 3(b). There is a delay of approximately 5 hours at which point we see VMs booting on both clouds at the University of Victoria. Because each 16 GB VM image is transfered over the WAN from Ottawa to Victoria we see a substantial penalty in the boot up times on Victoria based clouds. In order to avoid this penalty VM images could have been preloaded onto those clouds, however this would require extra work on the part of the user and knowledge of the clouds in question. In the future, we intend to address this problem by implementing a caching system such that identical images will only be transfered over the WAN once.

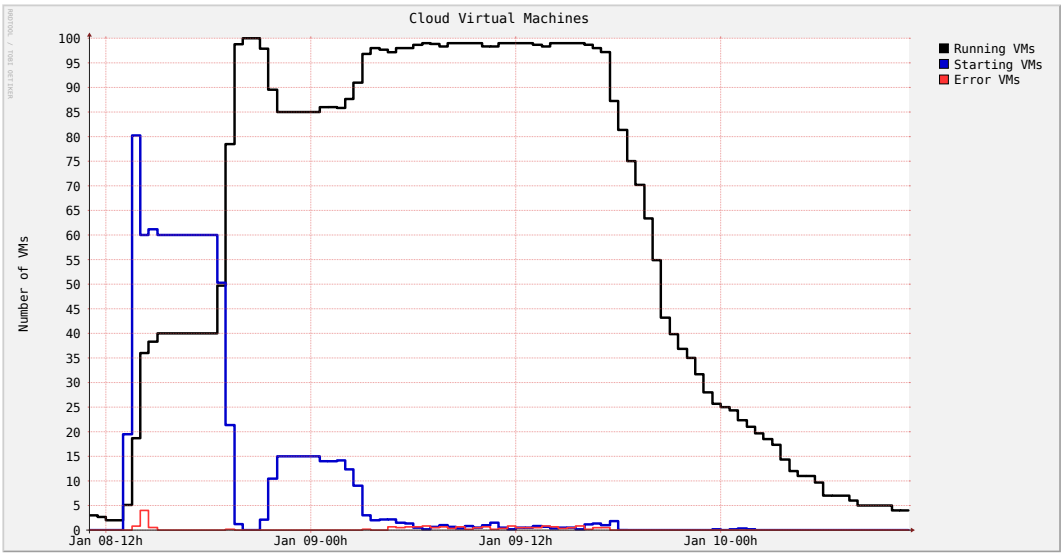
At roughly 20:00 on Jan 8 we see the full distributed cloud reaching its capacity of 100 VMs. After two hours of stable running 2 of the physical worker nodes on the UVic HEP cloud experienced a known intermittently occurring Xen networking bug which caused the VMs on those nodes to drop their network connection to the central Condor Scheduler. Once this occurs those machines drop off the available Condor resources and the jobs are listed as 'evicted' in Condor terms. Cloud scheduler immediately detects that these VMs are booted but not connected to the job queues. It then terminates those VMs via the cluster IaaS interface and attempts to instantiate new VMs to replace them. The cluster soon recovers and returns to full capacity and Condor resubmits the jobs. In this fashion the user is insulated from instabilities in the cloud system. This is particularly important in the case of these new technologies.

Near 15:00 on Jan 9 we see that there are fewer then 100 remaining running jobs and Cloud Scheduler begins to shut down the unused resources as the jobs finish. As the job queue drains we see the number of running VMs simultaneously reduced.

The total network I/O usage of the distributed cloud can be seen in Figure 4. Early in the run the demands on the VM image repository are substantial with half hour average network I/O peaking at 2.2 Gbps when all three research clouds are pulling from the image repository simultaneously. After the NRC cloud, which is co-located with the image repository finishes pulling the VMs we see I/O drop to 400 Mbps as the VM images are transfered across the WAN to Victoria. Amazon EC2 has no impact on the image repository I/O as the EC2 image is already stored at the Amazon site. The Xrootd I/O for data access for the running across all sites peaks at a modest 400 Mbps.



(a)



(b)

Figure 3. (a) Shows the state of 255 jobs executed on the cloud system over a 2 day period in January 2010. The blue line shows the number of queued jobs and the black line show the number of running jobs. (b) Shows the VMs automatically instantiated to meet the needs of those jobs over the same period. The blue curve shows the number of VMs being booted, the black curve shows the running VMs and the red curve shows VMs in an error state. The number of VMs being booted shows a short spike for VMs being booted on the NRC and Amazon EC2 clouds (where the VMs are locally stored). The VMs required at the two UVic clouds needed to be transfered from Ottawa to Victoria and this required 4-5 hours on the research network. The dip in the number of running jobs is explained in the accompanying text.

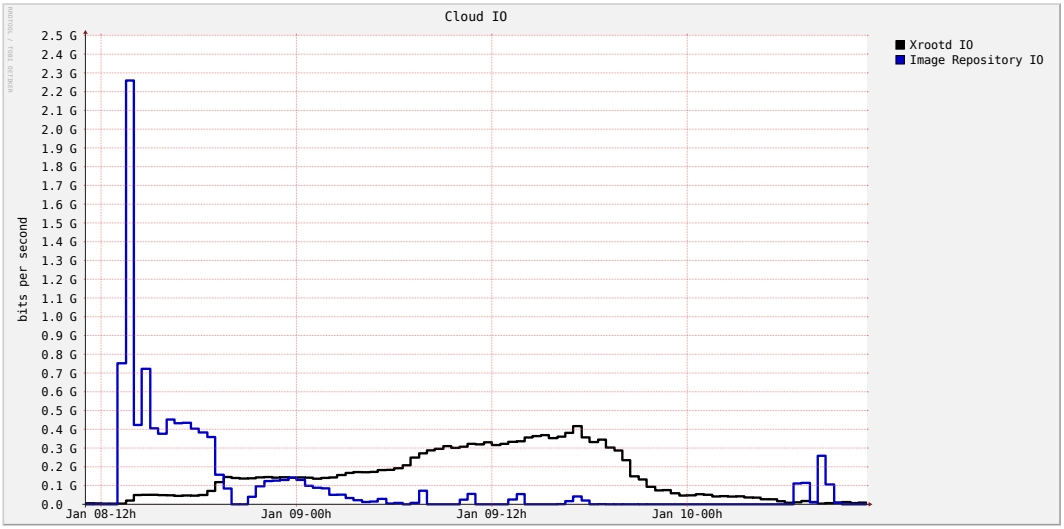


Figure 4. The total network I/O of the distributed cloud system. The blue line shows the network traffic out of the VM image repository as the images are being sent to both the local sites (as seen by the spike at early times) and remote sites (sustained transfer over a few hour period). The black line shows the data I/O from the jobs in the Cloud VMs. The rate grows with the number of jobs but there is also a dependence on the data set used in the analysis job.

4. Conclusion

We have constructed a multi-site distributed cloud using Condor and a new component called Cloud Scheduler. This distributed cloud was able to analyze roughly 5 TB of BaBar data using 100 VMs booted across four cloud sites with no work from the user beyond job submission and VM image preparation. Furthermore, we have shown that the system is resilient to failures in individual cloud resources; a feature essential to any system running on new IaaS technology.

Acknowledgment

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

References

[1] Agarwal A, Charbonneau A, Desmarais R, Enge R, Gable I, Grundy D, Penfold-Brown D, Seuster R, Sobie R and Vanderster D C 2008 Deploying HEP Applications Using Xen and Globus Virtual Workspaces *Proc. of Computing in High Energy and Nuclear Physics 2007 J. Phys.: Conf. Ser.* **119** 062002 doi:10.1088/1742-6596/119/6/062002

[2] Barham P, Dragovic B, Fraser K, Hand S, Harris T, Ho A, Neugebauer R, Pratt I and Warfield A, 2003 Xen and the art of virtualization. *Proc. of the 19th ACM Symp. on Operating Systems Principles* (Bolton Landing, NY, USA) pp 164-177

[3] The Kernel Virtual Machine `http://www.linux-kvm.org/`

[4] Alef M and Gable I HEP specific benchmarks of virtual machines on multi-core CPU architectures. *J. Phys Conf. Ser. 219, 052015 (2009).* doi: 10.1088/1742-6596/219/5/052015

[5] Nimbus Project `http://nimbusproject.org/`

[6] Open Nebula Project `http://www.opennebula.org/`

[7] Eucalyptus Software `http://www.eucalyptus.com/`

[8] Armstrong P, Agarwal A, Bishop A, Charbonneau A, Desmarais R, Fransham K, Hill N, Gable I, Gaudet S, Goliath S, Impey R, Leavett-Brown C, Ouellete J, Paterson M, Pritchet C, Penfold-Brown D, Podaima W, Schade D, and Sobie R J Cloud Scheduler: a resource manager for a distributed compute cloud, June 2010 (arXiv:1007.0050v1 [cs.DC])

[9] Aubert B *et al.* [BABAR Collaboration], The BaBar detector Nucl. Instrum. Meth. A **479**, 1 (2002) [arXiv:hep-ex/0105044].
[10] Thani D, Tannenbaum T and Livny M Distributed computing in practice: the Condor experience. Concurrency and Computation: Practice and Experience Vol. 17 (2005) 323.
[11] The Scientific Linux Distribution: `http://www.scientificlinux.org/`
[12] Dorigo A, Elmer P, Furano F, and Hanushevsky A 2005 XROOTD/TXNetFile: a highly scalable architecture for data access in the ROOT environment *Proceedings of the 4th WSEAS International Conference on Telecommunications and Informatics (TELE-INFO'05)*
[13] Tuecke S, Welch V, Engert D, Pearlman L, Thompson M 2004 Internet X.509 public key infrastructure (PKI) proxy certificate profile *IETF RFC 3820*, June 2004, `http://www.ietf.org/rfc/rfc3820.txt`