Simulation and user analysis of BaBar data in a distributed cloud

A. Agarwal,[a] M. Anderson,[a] P. Armstrong,[a] A. Charbonneau,[b] R. Desmarais,[a] K. Fransham,[a] I. Gable,[a] D. Harris,[a] R. Impey,[b] C. Leavett-Brown*,[a] M. Paterson,[a] W. Podaima,[b] R.J. Sobie,[ac] M. Vliet,[a]

[a]*Department of Physics and Astronomy, University of Victoria*
Victoria, BC, Canada V8W 3P6
[b]*National Research Council Canada*
100 Sussex Drive, Ottawa, Ontario, Canada
[c]*Institute of Particle Physics of Canada*
E-mail: crlb@uvic.ca

We present a distributed cloud computing system that is being used for the simulation and analysis of data from the BaBar experiment. The clouds include academic and commercial computing sites across Canada and the United States that are utilized in a unified infrastructure. Users retrieve a virtual machine (VM) with pre-installed application code; they modify the VM for their analysis and store it in a repository. The users prepare their job scripts as they would in a standard batch environment and submit them to a *Condor* job scheduler. The job scripts contain a link to the VM required for the job. A separate component, called *Cloud Scheduler*, reads the job queue and boots the required VM on one of the available compute clouds. The system is able to utilize clouds configured with various cloud Infrastructure-as-a-Service software such as Nimbus, Eucalyptus and Amazon EC2. We find that the analysis jobs are able to run with high efficiency even if the data is located at distant locations. We will show that the distributed cloud system is an effective environment for user analysis and Monte Carlo simulation.

The International Symposium on Grids and Clouds and the Open Grid Forum - ISGC2011,
March 25-30, 2011
Academia Sinica, Taipei, Taiwan

*Speaker.

1. Introduction

For a period of approximately nine years, the BaBar High Energy Physics experiment, based at the SLAC National Linear Accelerator Laboratory in Stanford, California amassed approximately one petabyte of experimental data, another one petabyte of simulated data, and approximately nine million lines of C++ and Fortan programming code. Though the collection of experimental data ended in 2008, the analysis of the data is expected to continue for many years. This scenario poses a problem common to all computationally intensive scientific research: how do we maintain the computational capability for an experiment when its highly complex systems outlive the ageing computing infrastructure on which it depends?

Infrastructure as a Service (IaaS) cloud computing is emerging as a new way to provide computing to the research community and a possible solution for the preservation of scientific data and application operability. The growing interest in clouds can be attributed in part to the ease with which complex research applications can be encapsulated within Virtual Machines (VMs) providing, among others, the following benefits:

1. **Shielding applications from changing technology.** As new servers and devices become available, operating systems are required to incorporate new software drivers. Additionally, operating systems and other software needs to be continually updated to remain free from vulnerability and maintain security. Lack of support for old systems renders them impractical to run on "bare metal", yet support for older applications like BaBar on the latest operating systems will not be provided. Virtualization provides a safe haven by shifting the responsibility for hardware support and security to the hypervisor.

2. **Separation of responsibilities.** Installing and operating a complex scientific application like BaBar requires a high degree of application knowledge and technical competence. This task is usually accomplished by more than one individual, a system administrator and an application specialist, forcing each to know something of the other's responsibilities. The introduction of virtualization into this scenario can bring relief to both parties. For the system administrator, it can allow the provision of a generic infrastructure or "cloud" running virtual machines. For the applications specialist, they can encapsulate their application once and employ the services of many resource providers.

3. **Insulation from the management policies.** 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 applications.

Today, open source virtualization software such as Xen [1] and KVM [2] are incorporated into many Linux operating system distributions, resulting in the use of VMs for a wide variety of applications. Sometimes, special purpose servers, particularly those requiring high availability or redundancy, are built inside VMs making them independent of the underlying hardware and allowing them to be easily moved or replicated. Often, applications running within VMs incur little or no performance degradation [3]. Studies have shown, for example, that particle physics application code runs equally well in a VM as on the native system [4].

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. 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, which provides a batch system on multiple clouds using one type of VM [6].

Even so, the deployment, management, and utilization of many VMs in IaaS clouds can be labour intensive. With the introduction of the *Repoman* repository manager [7] and *Cloud Scheduler* [8] resource manager, we further simplify the management of VMs and use of IaaS clouds by providing new functionality. *Repoman* provides authenticated access to a virtual machine image repository and allows users to create, modify, execute, share, and delete VM images. *Cloud Scheduler* provides virtual resource management on any number of academic and commercial clouds[1], executing and terminating VM images as required by jobs within the *Condor* [9] queues.

In this paper we present the architectures of both the interactive and batch environments created for BaBar analysis and simulation, highlighting the roles of *Repoman* and *Cloud Scheduler*. We will show the interaction of a user to create an image and present results of a significant simulation run.

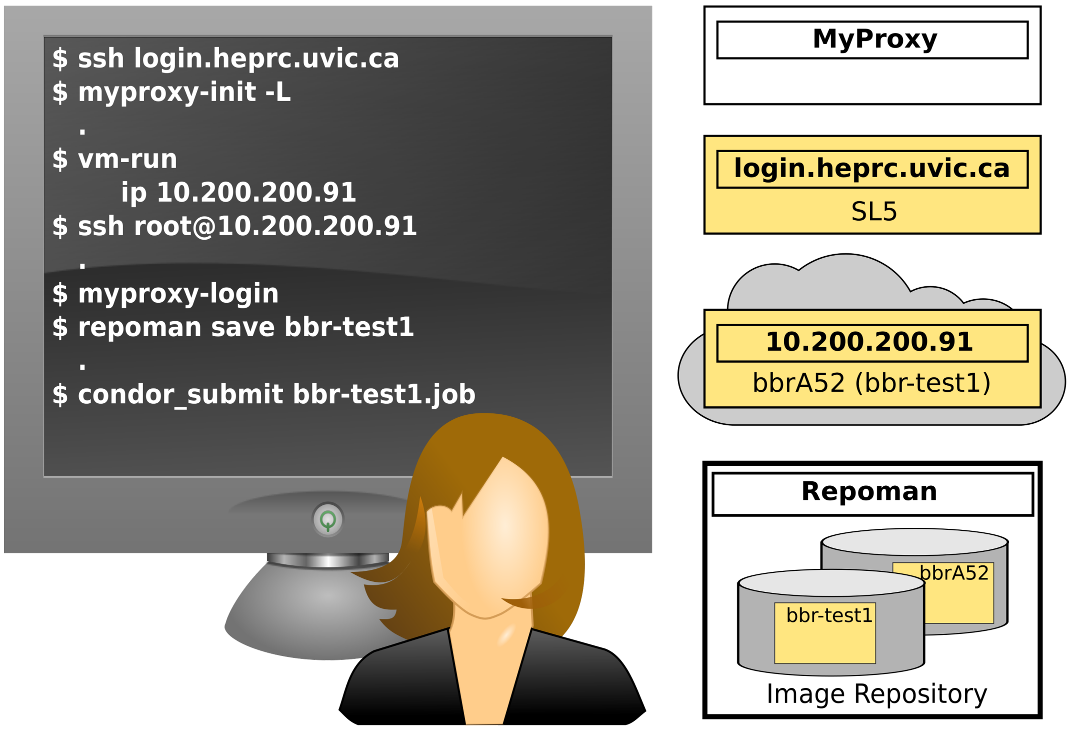
2. System Architecture

This section provides overviews of the interactive and batch systems. Though these systems have common components, for clarity, they are described independently.

2.1 The Interactive System

Fig. 1 illustrates the major components of the interactive system. No particular requirements are placed upon the configuration of the user's workstation beyond the ability to remote shell login in to the head node and interact with a command line interface. However, the user may choose to install an environment similar to the head node (see below) and interact with the rest of the system without the aid of the head node's services. The head node is provided for the user's convenience and is configured with a standard Linux distribution (e.g. Scientific Linux 5.x), GSI (e.g. myproxy-init) and interactive VM (e.g. repoman, vm-run, etc.) tool sets and provides home and X509 certificate directories for each user. The *Repoman* image repository manager is used to control access to the VM image repository based on the user, group, and image meta-data that it

[1]Science clouds use hardware resources funded by governments for research purposes and are located in universities or national laboratories. Commercial cloud providers include Amazon, RackSpace and IBM, etc.



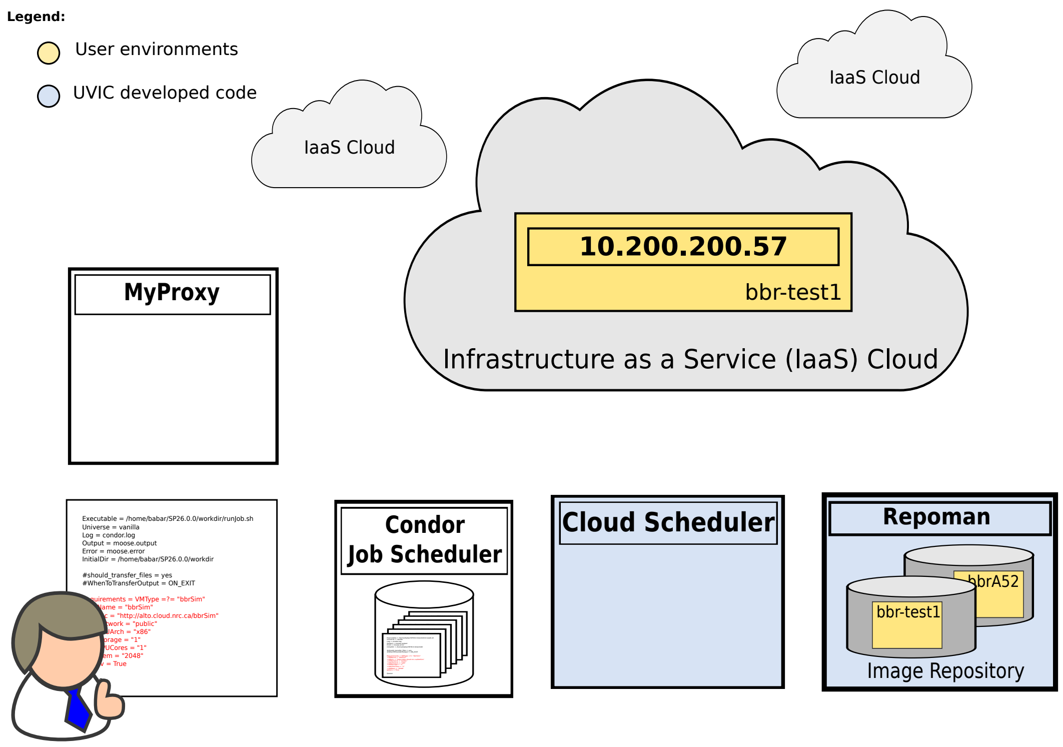
```
$ ssh login.heprc.uvic.ca
$ myproxy-init -L
   .
$ vm-run
      ip 10.200.200.91
$ ssh root@10.200.200.91
   .
$ myproxy-login
$ repoman save bbr-test1
   .
$ condor_submit bbr-test1.job
```

Figure 1: An overview of the Interactive System architecture consisting of the user's workstation, a MyProxy server to manage X509 credentials, a head node providing user access to all required tools, an IaaS cloud capable of running interactive VMs, and a *Repoman* server together with its VM image repository.

implements and maintains. It will support any back-end file system capable of providing storage and retrieval of large (20GB) sequential VM image files. With regard to the IaaS cloud, it is recommended that a dedicated node, or at least some number of VM slots, be reserved to allow a timely response to interactive VM requests.

The following description, with reference to Fig. 1, illustrates how the interactive system operates:

- The user logs in to the head node to access the interactive VM tool set.
- The user establishes an X509 proxy certificate in order to interact with the *Repoman* image repository manager. The preferred method is to use `myproxy-init` and `myproxy-logon` commands (or just `myproxy-init -L` command) to establish proxy credentials, but authentication could also be achieved without a MyProxy [10] server using a `grid-proxy-init` command instead.
- The `vm-run` command, issued from the head node, is used to initiate an interactive VM. It calls the IaaS cloud interface to deploy the VM and, by default, executes the `bbrA52` image, though the user may select another image if they so choose. IaaS propagates the image from



```
Executable = /home/babar/SP26.0.0/workdir/runJob.sh
Universe = vanilla
Log = condor.log
Output = moose.output
Error = moose.error
InitialDir = /home/babar/SP26.0.0/workdir

#should_transfer_files = yes
#WhenToTransferOutput = ON_EXIT

quirements = VMType =?= "bbrSim"
Name = "bbrSim"
c = "http://alto.cloud.nrc.ca/bbrSim"
twork = "public"
UArch = "x86"
orage = "1"
UCores = "1"
em = "2048"
v = True
```

Figure 2: An overview of the Batch System architecture consisting of a MyProxy server to manage X509 credentials, a *Condor* job scheduler, the *Cloud Scheduler* to manage cloud resources, the *Repoman* image repository manager together with its repository, and one or more IaaS clouds.

the repository, assigns an IP address, adds the user's public key to the root user's authorized keys, deploys and boots the new VM. On completion of the image boot, the IP address of the VM is returned to the user so that they may log in.

- The user logs in to the interactive VM as root. With full privileges, the user may modify the environment, add or remove software, develop and install custom code. Normally, changes to the environment are discarded when a VM is shutdown, but in this environment, the user can interact with *Repoman* to save it.
- In order to do this, the user must first establish X509 credentials on the interactive VM, then issue the `repoman save` command to save a copy of the current environment as a new VM image. Each time the `repoman save` command is used the user must specify an image name, allowing the user to create one or more images with different configurations.
- Once images are saved in the image repository, they can be used interactively via the `vm-run` command or referenced from a *Condor* job file for batch processing.

2.2 The Batch System

The following description, with reference to Fig. 2, illustrates how the batch system operates:

- The user will need to establish X509 proxy credential in order to use the batch system. For further discussion regarding X509 credentials, see the Interactive System description above.
- The user prepares and submits *Condor* job files to a *Condor* job scheduler. The job file is a standard *Condor* job file [11] with a number of additional parameters to define the cloud resources required. Jobs requiring cloud resources will remain in a *Condor* queue until a VM of the required type becomes available; an agent is needed to initiate appropriate cloud resources.
- The *Cloud Scheduler* periodically scans the *Condor* queues looking for jobs waiting for cloud resources. It can be configured [12] to utilize one or more geographically distributed clouds. When a job is found, *Cloud Scheduler* selects a cloud, contacts its IaaS interface requesting the deployment of a VM with the required image, processors, memory, network and storage. *Cloud Scheduler* will continue issuing deployment requests until either all jobs have been satisfied or all resources have been utilized.
- In response to a deployment request, the IaaS cloud software selects a compute node within the cloud, directing its worker node to initiate a VM.
- The selected compute node copies the image from the image repository, contextualizes the image by inserting job scheduler location, public keys and X509 credentials, and then directs the node's hypervisor to deploy the VM.
- The node's hypervisor deploys the VM and boots the image.
- At the completion of the boot process, a *Condor* initialization script registers the new VM, together with its attributes, with the *Condor* job scheduler. The *Condor* job scheduler is now able to dispatch matching jobs to the newly registered VM. A single VM may process many jobs. As jobs complete and resources become available on the VM, the job scheduler will dispatch other matching jobs to the VM.
- Eventually, the *Condor* job queue will be drained of jobs matching a particular running VM. During its periodic scanning of the *Condor* queues, *Cloud Scheduler* detects when VMs are no longer required and directs the corresponding cloud to terminate the redundant VM.
- During VM shutdown, a *Condor* termination script deregisters the VM with the *Condor* job scheduler.
- After a short interval to allow the shutdown to complete, the hypervisor is directed to destroy the VM releasing resources for other required VM deployments.

2.3 Data Management

Other services included in the interactive and batch systems are data management services, web caching servers to improve image propagation times, and monitoring services to gather statistics and aid in problem diagnosis.

The data repository and the back-end filesystem for the image repository are both hosted on *Lustre* [13] filesystems. We operate two independent Lustre filesystems, one in Ottawa and one in Victoria. Both filesystems are configured with multiple bonded gigabit ethernet private networks

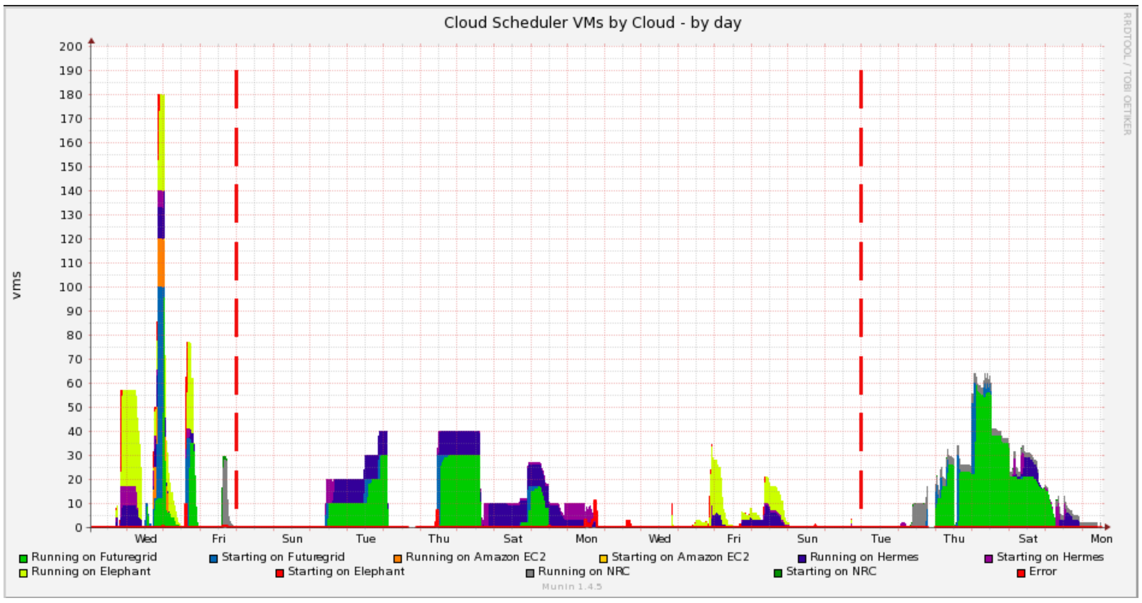


Figure 3: Analysis image development and usage, February 2 through February 28, 2011. The graph shows three distinct periods of activity: Period one, system testing, Wed 2nd through Fri 4th - Period two, image development, Mon 7th through Mon 21st - Period three, analysis production, Tue 22nd through Mon 28th.

and provide the full range of file services to the local servers. Access to the data and image repositories is through single gigabit ethernet public networks and provided by front-end services. For the data repository, read-only access to BaBar data from the wide area network (WAN) is provided by *XRootD* [14]. *XRootD* is an integral component of the BaBar application suite. For the image repository, storage and retrieval of images is through the *Repoman* server using the HTTP/HTTPS protocols.

3. Analysis with User Developed Images

Fig. 3 shows use of the batch system by a BaBar user developing an analysis in February 2011. The colours within the graph indicate the location of starting and running VMs on the various clouds. This was the first attempt to use the interactive and batch systems in concert and was characterized by three distinct periods of activity.

In period one, the system functionality was tested using all available resources. The highest column in this period indicates that we managed to initiate and run up to 180 VMs on five distributed clouds geographically dispersed across North America. The five distributed clouds are operated by five administrative domains and are composed of two distinct types. Nimbus clouds are provided by the UVIC HEP Research Computing and by Compute Canada both located at the University of Victoria, Canada, by the National Research Council Canada in Ottawa, and by FutureGrid [15] in Chicago. The Amazon EC2 cloud that we used is located in North Virginia. For period one, the image repository was located in Ottawa, and the data repository was located in Victoria. Although we were able to boot VMs and process data on all available resources, the period was plagued with lengthy image propagation times and poor I/O performance. A networking issue

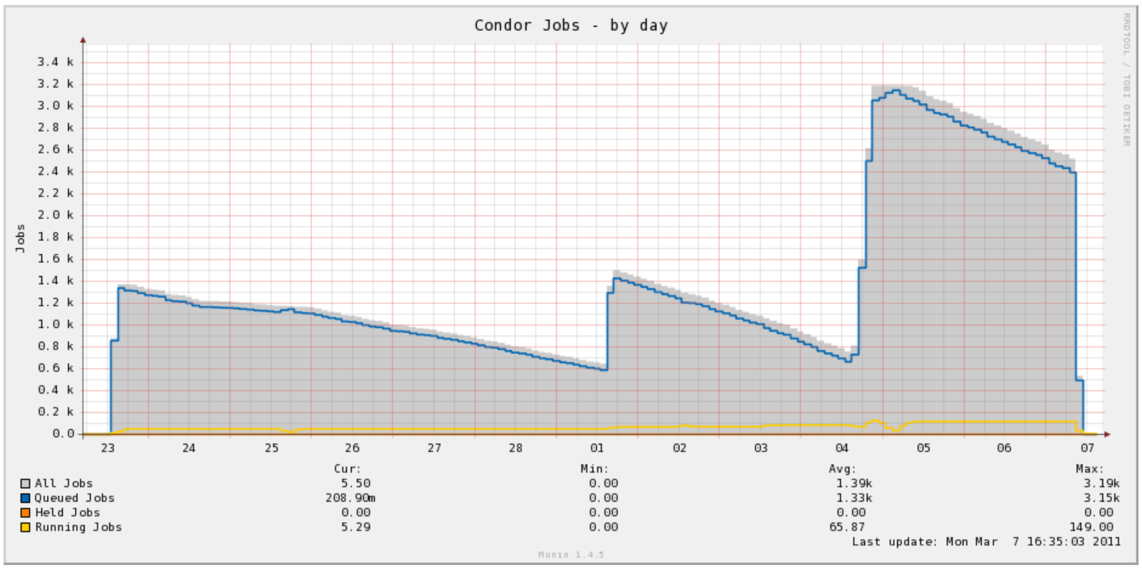


Figure 4: The user's view, simulation production, February 23 through March 7, 2011. The graph shows the total number of jobs in the system (grey), jobs queued for execution (blue), and jobs that are executing (yellow). Nearly 5000 jobs were submitted over the 12 days; more than 1300 on Feb 23, approximately 8500 on Mar 1, and 2500 on March 4. The production run was terminated on March 7 by the removal of the remaining jobs from the *Condor* queue.

on the link between Ottawa and Victoria was the primary cause. With the need to continue our development and the uncertainty of a timely resolution to the networking issue, we decided to re-organize and reconfigure the system to avoid the problem for the following two periods. However, this networking issue was resolved early in March.

Period two is characterized by typical start/stop activity as the user developed their analysis image and the configuration was adjusted to give the best possible user experience. The image repository was moved to Victoria and various combinations of cloud resources were tried. Also, a Squid web caching server was used in Victoria to improve image propagation times. Although this technology looks promising, aggregate propagation times reduced from three hours to less than half an hour, further development and other technologies need to be explored.

Period three encompasses successful analysis achieved with a user developed VM image. A combination of three distributed clouds were employed peaking at 60 VMs to execute over 600 jobs and process many billions of BaBar events.

4. Simulation Production

Monte Carlo Simulation of BaBar data is a centrally controlled process. Sites wishing to contribute simulated data to the collaboration must run a benchmark suite on each node and validate the results before a data segment allocation will be granted. In this case, a single VM image was configured with BaBar and validated at each of the three sites used. Fig. 4 shows the results of the simulation run between February 23 and March 7, and highlights job submissions occurring on the 23rd, 1st and 4th totalling nearly 5000 jobs. The batch system peaked at 149 concurrently running

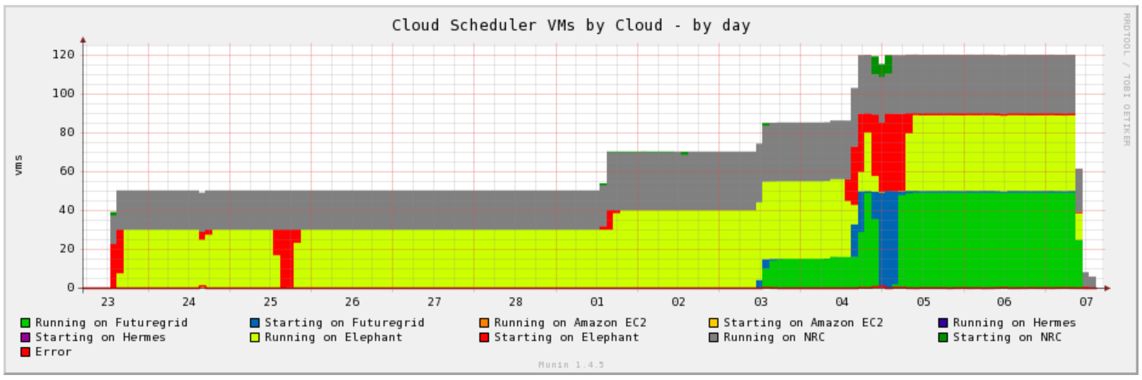


Figure 5: System view, simulation production, February 23 through March 7, 2011. The colours within the graph show the location of starting and running VMs on three distributed clouds. Virtual machine shutdown and restarts occurring on the 24th, 25th, 2nd, 3rd, 4th and 5th demonstrate the fault tolerance of the batch system.

jobs and a maximum queue depth of nearly 3200 jobs. Approximately 2100 jobs of five hours duration were completed before the system had to be halted for scheduled network maintenance. Production was terminated with a simple `condor_rm` command which removed the remaining jobs from the *Condor* queue.

The view for users is simple and straightforward; they prepare job files and use standard *Condor* commands to submit, list, and manage their workload. The system view is a little more complicated. Fig. 5 shows the same run from the system point of view. In this case we see that the workload was spread across three distributed clouds, in Chicago, Ottawa, and Victoria, that resources were added on four different occasions, and that the system took a number of unexpected outages.

These service interruptions can be divided into four categories. On the 24th and the 3rd, nodes had to be rebooted because of hardware problems including one motherboard replacement. On the 25th and the 2nd, IP reallocations and DHCP server restarts caused the shutdown and redeployments of multiple VMs at their respective sites. On the 4th, the resources leases on the Victoria cloud expired. By default, Nimbus leases resources for seven days. This event on the Victoria cloud, occurring exactly seven days after its DHCP server restart, caused all the VMs to be restarted. Finally, on the 5th we see VMs being restarted on all three clouds. This event was caused by the inefficient method by which we submitted the 2500 jobs on that day. Normally, jobs are submitted individually, but *Condor* does allow many jobs to be submitted in a single batch with much less overhead. When *Condor* is processing job submissions it is unable to process other requests. Our submission of 2500 jobs individually took nearly six hours and effectively caused a denial of *Condor* service to *Cloud Scheduler*. With *Cloud Scheduler* unable to inspect the *Condor* queues for more than five hours, it began to free up cloud resources and the problem was only rectified when we stopped submitting jobs to the queue.

In all cases, the batch system responds the same way to all outages and uses the standard *Condor* approach to handling evicted jobs; they are re-queued for execution. In our case, we do not use check-pointing and the job is merely restarted.

5. Conclusion

Acknowledging that the underlying systems are complex and the use cases presented are not flawless, we believe the two systems presented provide excellent utility, producing results for users with ease. Images created with the interactive system were used to analyze real data, and data simulated with the batch system is contributed to the BaBar collaboration. We also believe that these systems could provide excellent utility for other applications.

The interactive and batch systems described in this paper provide effective means for using distributed clouds for both VM image development by end users and for batch production. The systems have proved themselves to be flexible, scalable, reliable, fault-tolerant and easy to use.

Two of the components, *Cloud Scheduler* and *Repoman*, provide capabilities that were not previously available. All the components used, including those written by our group, are freely available as open source projects and could be used to replicate or adapt the two systems for other applications.

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

References

[1] 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. SOSP03: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles.

[2] KVM (Kernel Based Virtual Machine). http://www.linux-kvm.org/page/Main_Page.

[3] 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

[4] 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

[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] P. Armstrong, A. Agarwal, A. Bishop, A. Charbonneau, R. Desmarais, K. Fransham, N. Hill, I. Gable, S. Gaudet, S. Goliath, R. Impey, C. Leavett-Brown, J. Ouellete, M. Paterson, C. Pritchet, D. Penfold-Brown, W. Podaima, D. Schade, R.J. Sobie. Repoman: A Simple RESTful X.509 Virtual Machine Image Repository. The International Symposium on Grids and Clouds and the Open Grid Forum - ISGC2011.

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

[9] D.Thani, T.Tannenbaum and M.Livny. Distributed computing in practice: the Condor experience. Concurrency and Computation: Practice and Experience Vol. 17 (2005) 323.

[10] MyProxy: X.509 Credential Management Service. http://grid.ncsa.illinois.edu/myproxy/

[11] Sample *Condor* job files. https://github.com/hep-gc/cloud-scheduler/tree/master/test-sets/cs0.2.

[12] Sample *Cloud Scheduler* resource configuration file.
https://github.com/hep-gc/cloud-scheduler/blob/master/cloud_resources.conf

[13] Lustre: High Performance and Scalability. Oracle. http://wiki.lustre.org/index.php/Main_Page.

[14] XRootD: software framework for scalable data access. http://xrootd.slac.stanford.edu/.

[15] FutureGrid: A high-performance grid cloud test bed. https://portal.futuregrid.org/.