HEP Computing in a Context-Aware Cloud Environment

Frank Berghaus, Ron Desmarais
Ian Gable, Colin Leavett-Brown
Michael Paterson, Randall J. Sobie
Ryan Taylor

Department of Physics and Astronomy
Institute for Particle Physics
University of Victoria, Victoria, Canada
frank@uvic,ca, rd@uvic.ca,
igable@uvic.ca, crlb@uvic.ca,
mhp@uvic.ca, rsobie@uvic.ca,
rptaylor@uvic.ca

Andre Charbonneau
Research Computing Support
Shared Services Canada, Ottawa, Canada
Andre.Charbonneau@SSC-SPC.gc.ca

ABSTRACT

This paper describes the use of a distributed cloud computing system for high energy physics (HEP) applications. The system is composed of IaaS clouds integrated into a unified infrastructure that has been in production for over two years. It continues to expand in scale and sites, encompassing more than twenty clouds on three continents. We are prototyping a new context-aware architecture that enables the virtual machines to make connections to both software and data repositories based on geolocation information. The new design will significantly enhance the ability of the system to scale to higher workloads and run data-intensive applications. We review the operation of the production system and describe our work towards a context-aware cloud system.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed Applications

Keywords

cloud computing, grid of clouds, scientific applications

ScienceCloud 2014, June 23, 2014, Vancouver, BC, Canada.

http://dx.doi.org/10.1145/2608029.2608035 .

1. INTRODUCTION

In a previous paper, we showed that a distributed cloud computing system can be effectively used for high-throughput computing workloads for high energy physics (HEP) applications [12]. Since 2012, we have operated a distributed cloud in a production mode, running over two million jobs, with peak workloads of 1500 concurrent jobs. This system is composed of a varying number of IaaS clouds, owned and administered by academic institutions and national laboratories, and distributed across three continents. The exact size of this unified infrastructure varies according to availability, maintenance and development cycles, but as many as twenty five clouds have been employed at one time.

We have focused on running applications with no input data requirements such as the simulation of the particle collisions that are required for the development of analysis algorithms and understanding the interaction of particles with the detectors. The output from these applications is modest and is written to a single site on each continent. The applications running in the virtual machine (VM) instances retrieve the HEP-specific software and calibration databases from a few locations. The distributed cloud has been very successful, as highlighted in our previous work [12], without any scalability limits observed.

In order to make the system more versatile and to run applications requiring large input data sets, we need to address a number of issues. First, we want the VM instance to access the closest software cache to reduce the demand on the single site (eliminating a single point of failure), and minimize the network latency.

Second, we wish to run applications that require moderately large input data sets (20-100 GB for a single job over a 12-hour period). The input data needs to be read in at a high rate by the application. Hence we have selected protocols that cache or copy the data to the local disk rather than streaming the data directly over the network. In addition, the job will query a federation storage system for the nearest location of the input data and stage the data, using high-speed transfer protocols, to the local disk.

We believe that these issues can be addressed if the distributed cloud computing system becomes context-aware. Context-aware systems, that can sense and react to their

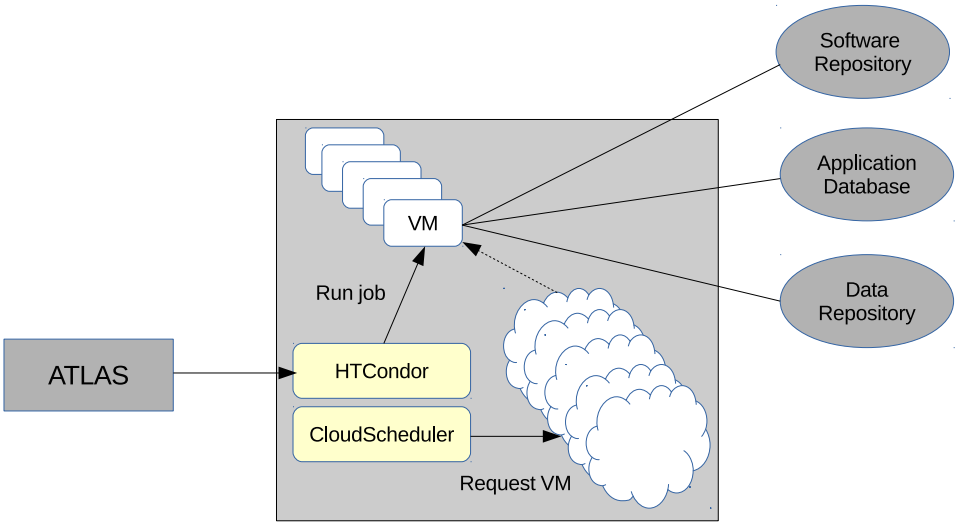

Figure 1: A high-level view of the distributed cloud computing system as used by the ATLAS experiment since 2012. The cloud batch system is contained in the outlined box where jobs are submitted by the ATLAS user to HTCondor. CloudScheduler uses the list of jobs to request the start of VM instances on the available IaaS clouds. The VM instances connect to the software repository, the application database and the data repositories. The application database contains calibration constants associated with experimental apparatus whereas the software repository has the application source code.

environment, are not new and are used extensively in the mobile phone industry. For example, context-aware applications use location to adapt interfaces, locate data, and discover services for the user. In our case, a context-aware cloud would dynamically locate the optimal locations for retrieving the software and data. More speculatively, we imagine the job or VM instance would establish a network connection to those repositories using the emerging Software Defined Network technologies.

Section 2 of this paper describes the distributed cloud computing system as it exists today, and highlights the major components and operation. Section 3 describes the changes needed to transition to a context-aware distributed cloud. Section 4 provides a summary of this work.

2. DISTRIBUTED CLOUD SYSTEM

A full description of the distributed cloud computing system can be found in [12]. The design is based on the original concept of a 'grid of clouds' or 'sky computing' as a solution for combining separate IaaS clouds into a unified infrastructure [6].

The distributed cloud is used for the ATLAS experiment at the CERN Laboratory in Geneva by utilizing IaaS academic clouds in Canada, the United States, Europe and Australia (see [12]). The majority of participating clouds are configured with the OpenStack [9] software suite, though some are still running Nimbus [7]. The use of the Nimbus software on the IaaS clouds is expected to end in 2014 with most clouds migrating to OpenStack.

At the heart of the distributed cloud computing system is an *HTCondor* and *CloudScheduler* instance operating at the University of Victoria. We have recently commissioned a new, independent instance at CERN with its purpose being to shift operations to the central location so that we can take advantage of round-the-clock production support staff.

The applicability of the distributed cloud computing model is not limited to the ATLAS experiment. The Canadian CANFAR project has created an independent system using different resources for the processing and analysis of astronomical data [1]. In addition, we recently created a new system for the Belle-II HEP experiment at the KEK Laboratory in Japan. Each of the systems has their own *HTCondor/CloudScheduler* instance and utilize different IaaS clouds (though there are some shared clouds).

The goal is to run embarrassingly-parallel HEP applications on IaaS clouds. The application environment is completely provided by the VM image and distributed cloud supporting services. Typically, jobs run on a single core for approximately $6 - 12$ hours.

Very recently the ATLAS experiment has developed an application that uses 8-cores for a single job. Running multi-core jobs in an 8-core VM is expected to have many benefits (reduced memory requirements, shared caches, reduced in-

put file staging space, and fewer jobs) and we expect to run these jobs in the coming months.

In fig. 1, we present a high-level view of the distributed cloud. The scheduling of batch jobs is performed by *HTCondor* [13] and the deployment of VM images is done by *Cloud Scheduler* [2]. *HTCondor* was designed as a cycle scavenger, making it an ideal job scheduler for a dynamic cloud environment where VM instances appear and disappear based on demand. *Cloud Scheduler* periodically reviews the requirements of the jobs in the *HTCondor* job queue and makes requests to boot user-specific VM images on one of the IaaS clouds. Once the VM image is booted, it attaches itself to the *HTCondor* resource pool and is ready to accept jobs. The instances are shut down when there are no jobs in the *HTCondor* queue.

The VM images are manually uploaded and stored in each cloud, as in the case of OpenStack clouds, or retrieved from a central image repository for Nimbus clouds. A collaboration involving researchers on ATLAS at CERN and members of this group have developed a set of Puppet scripts [10] that provide site-specific contextualization [8]. In addition, the Puppet scripts can configure a VM instance with the desired application software suite, starting from a standard release of Scientific or RedHat Linux. This is required for the clouds that do not allow users to upload their own customized images. In this situation, an image provided by the cloud site is booted and then configured at run-time for the task at hand using Puppet.

The application software is stored in an HTTP read-only file system called the CERN Virtual Machine File System (CVMFS)[3]. CVMFS presents a remote directory as a local file system, in which the client has read access to the project application files. On the client's first access request, a file is downloaded and cached locally, and all subsequent requests for that file will use the cached copy. The ATLAS project uses Squid [17] as an HTTP web cache for CVMFS; we will describe the use of Squid caches in more detail in the next section.

In addition, a Frontier database, developed for the HEP community [4], is used for calibration data required during the running of the application software by the ATLAS experiment. All output data is written to the Victoria Storage Element (SE) for both the ATLAS and Belle-II experiments.

3. CONTEXT-AWARE CLOUD DESIGN

The distributed cloud is performing well with the current type of HEP applications and workload conditions. The new design will include a system (currently under development) to manage and distribute VM images to the distributed clouds. Further, it will include a more efficient system for distributing software that is starting to be used in a production environment. We plan to use a federated storage repository for input data sets and currently evaluating it in a test bed. In the following sections, we describe our efforts in these key areas.

3.1 Image distribution system

In working with IaaS clouds that can instantiate images stored remotely (such as Nimbus clouds), we have developed a VM image repository, called *Repoman* [18]. *Repoman* operates as a standalone system that exposes a RESTful web service allowing users to upload their VM images to the repository and configure various image properties such as

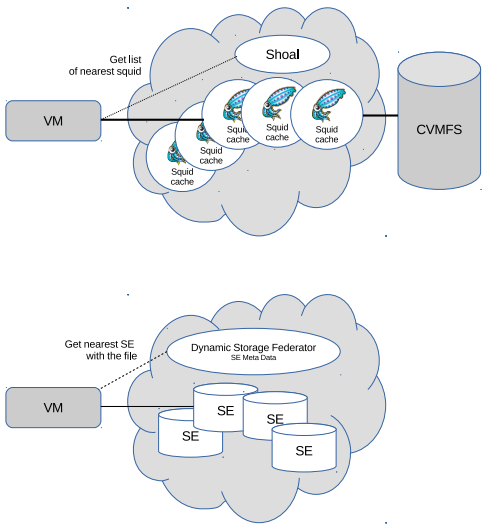

Figure 2: The upper plot shows the new software distribution system. The VM periodically retrieves a list of Squid caches from Shoal that is ordered by location and load of the Squid. Each Squid cache connects to the CVMFS repository at CERN. The lower plot shows the federated storage system that will be used for the input data samples. The VM queries the Dynamic Storage Federator for the physical path to the nearest Storage Element (SE) with the requested file. The Dynamic Storage Federator has a complete metadata catalog of the Storage Elements (SE).

image metadata and access controls (the software can be obtained from `https://github.com/hep-gc/repoman`). Using *Repoman* client tools, image snapshots can be saved to the *Repoman* server from within the running image, which greatly simplifies the workflow for the end users. In addition to being accessible via the *Repoman* client tools, each VM image stored in *Repoman* is also accessible via HTTP(S) and is easily retrieved using command-line tools such as wget and curl. *Repoman* implements the concept of dual-hypervisor VM images, allowing images containing older Linux kernels to boot under both the KVM and Xen hypervisors. We are unaware of another VM image repository that operates independently of an IaaS cloud and also provides dual-hypervisor capability.

With the emergence of clouds that require images to be stored locally before instantiation (such as OpenStack), an image propagation component is needed to automatically deploy images to remote clouds. We are developing a service, called *glint*, for managing images stored on multiple clouds, concentrating on OpenStack clouds and the Glance repository, but with a plugable architecture to allow support for different cloud types. Through a web browser interface, the user will identify clouds and be able to control the distribution of images.

Glint provides site management, image registration, credential management and image deployment
(see `https://github.com/hep-gc/glint`). By design, *glint*

has no pre-established list of clouds. Instead the user must specify and provide their credentials for each cloud. The images to be distributed are uploaded by the user to *glint* who then selects the target clouds for each image. Image transfers are multithreaded to reduce upload times and the user is notified once the transfers are completed.

The software is modular, making it simple to integrate new cloud types such as Amazon EC2 or Google Compute Engine. Extra functionality such as web-based image consistency checks and improved propagation monitoring is planned for future releases of *glint*.

3.2 Software repository

The CernVM file system (CVMFS) [3], supported with Squid HTTP web caches, is an ideal solution for distributing application software to VM instances. The software suite for HEP projects, such as ATLAS or Belle-II, can be many gigabytes in size but often only a small subset of the software is used in a particular job. CVMFS significantly reduces the size of the HEP VM images by eliminating the need to store the application file tree in the image. Our use of 8-core VM instances, also allows the applications to share the CVMFS cache within the VM instance further reducing the memory requirements.

The master copy of the ATLAS and Belle-II software is stored in the CVMFS repository at CERN. The HEP community uses Squid caches to distribute the workload of delivering the application software, minimizing the demand on single, distant repositories.

For most of the recent operation, we have used the Squid cache in Victoria for the cloud sites. The Victoria Squid cache supports approximately 1000 cores of cloud and traditional computing resources. The number of client requests to this Squid ranges from 6000 requests per minute on average to a peak of 130,000 requests per minute with the rate of data transfer ranging from 2 MB/s to 16 MB/s, respectively. The Squid server fetches information from the CVMFS server at CERN at between 100 (average) and 1000 (peak) times per minute with the rate of 100 (average) to 4000 (peak) KB/s.

We view our reliance on a single Squid cache for the cloud resources as a potential bottleneck as the system expands to more sites. There are many Squid caches for the HEP community distributed around the globe. We can distribute the load and minimize the long distance transfers by having the VM instances use the nearest Squid cache. There are no services that provide this functionality, and, as a result, our group has developed *Shoal*, which is a dynamic web cache publishing service [16].

Shoal provides a reliable, scalable solution for a distributed cloud environment and can manage large numbers of requests (see https://github.com/hep-gc/shoal). *Shoal* builds and maintains a list of Squid caches that advertise their existence to the central *shoal-server* using a *shoal-agent*. The clients (VM instances) contact the server for a list of Squids that is ordered on the relative location to the client and the load of the Squid server (see Fig. 2). The frequency at which the clients contact the server is configurable and we currently have it set to once every 30 minutes. If a Squid cache is no longer available, then the client fails over to the next one in the list.

Recently, we added Squid caches in Geneva and the TRIUMF Laboratory in Vancouver to *shoal* and initial operation has been successful. We are planning to expand the pool of Squid caches as we gain further operational experience.

3.3 Data federation

Currently, the distributed cloud is used for ATLAS production jobs that use little or no input data. The jobs write their modestly-sized output to the ephemeral storage on the VM, which is then transferred to the Victoria SE after the job is completed.

We are evaluating various options for running jobs with large input data sets. We observe that streaming the input data from a remote location results in a large number of failed jobs because of latency issues. Hence, we are only considering solutions where the input data is staged to the ephemeral disk of the VM instance.

In general, the input data used by each HEP job is independent (typically 20-100 GB) and not shared between jobs. The data staged by the VM is stored on the ephemeral disk of the instance, which is physically attached to the node. For completeness, we mention that the input/output data are written in a HEP-specific format, called *Root* [11] and transferred over the network with protocols such as *XRootD* [19] or gridftp. However, the operation of the distributed cloud does not have any dependence on the file format or network protocol.

We have adopted WebDAV [14], a widely-used open standard using HTTP protocols, to present a file system to the VM instances. DAVfs, a WebDAV client, presents a view of the entire file system tree but only stages the file content as it is accessed. Once the file is local, there are no longer any latency issues. Like CVMFS, the WebDAV cache is shared by all applications within the VM instance.

Currently, the VM instances retrieve the input data from the Victoria SE using a WebDAV client. Using a single SE for the entire system is a limiting element, especially for distant clouds. Further, the Victoria SE only hosts a subset of the ATLAS data. Other SEs in Canada, for example, allow access to their storage by WebDAV and using these SEs would increase the pool of available data as well as distribute the file transfer workload.

A federated storage system is an ideal solution for our distributed cloud. The federated storage system would provide, like CVMFS, a common view of the data file tree to the application (see Fig. 2). The federated storage system would direct the job to the nearest SE with the desired input data. If multiple copies of the data reside on different SEs, then the selected SE would be determined by the geographical location of the client with respect to the storage servers.

We are testing a new federated WebDAV system that meets our design requirements [5]. There is an alternative solution being developed internally by the ATLAS project called FAX [15]. We favour the federated WebDAV solution as FAX is designed for a HEP environment. The use of federated storage systems within HEP is relatively new and both WebDAV and FAX are under evaluation. FAX is being used in a pre-production deployment with a limited number of sites [15]; currently there are no HEP federated WebDAV deployments.

We have established a prototype federated WebDAV system with three storage resources (including the Victoria SE) and the basic functionality tests have been successful. Currently, we are testing its operation and performance. Our next step is to evaluate the federated storage system in a

more realistic environment using real ATLAS applications. We plan to establish a small distributed cloud test system with resources at both ends of the continent. We will measure the performance and reliability of the WebDAV federator, and present the results at the workshop.

4. SUMMARY

We have described our activities and plans to evolve a production distributed cloud computing system to a context-aware design. The goal is for the VM instances to locate and retrieve the software and input data based on their geographic location, network and system loads. Many of the elements of the software distribution system are in place and being integrated into the production system. The use of federated data repositories will require significant testing and evaluation; however, the outlook for this solution is very positive.

Acknowledgements

The support of CANARIE, the Natural Sciences and Engineering Research Council, and FutureGrid are acknowledged.

5. REFERENCES

[1] CANFAR: Canadian Advanced Network for Astronomical Research. `https://wiki.cadc-ccda.hia-iha.nrc-cnrc.gc.ca/canfar/`

[2] 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, and R. Sobie. Cloud Scheduler: a resource manager for a distributed compute cloud. arXiv:1007.0050v1 [cs.DC].

[3] J. Blomer, P. Buncic, and T. Fuhrmann. CernVM-FS: delivering scientific software to globally distributed computing resources. In *Proceedings of the first international workshop on Network-aware data management*, NDM '11, pages 49–56, New York, NY, USA, 2011. ACM.

[4] Frontier distributed database caching system. `http://frontier.cern.ch/`.

[5] F. Furano *et al.*. Dynamic federations: storage aggregation using open tools and protocols, Journal of Physics: Conference Series 396 (2012) 032042. doi:10.1088/1742-6596/396/3/032042

[6] Katarzyna Keahey, Mauricio Tsugawa, Andrea Matsunaga, Jose Fortes, Sky Computing, IEEE Internet Computing, vol. 13, no. 5, pp. 43-51, 2009.

[7] Nimbus: an open-source EC2/S3 compatible IaaS implementation. `http://www.nimbusproject.org/`.

[8] H. Ohman *et al.*. Using Puppet to contextualize computing resources for ATLAS analysis on Google Compute Engine. Proceedings of the CHEP 2013 Conference, Amsterdam 2013.

[9] OpenStack: Open source software for building clouds. `http://www.openstack.org`.

[10] Puppet Open Source. `http://puppetlabs.com/puppet/puppet-open-source`.

[11] Root: a framework for the analysis of HEP data. `http://root.cern.ch`.

[12] Randall Sobie, Ashok Agarwal, Ian Gable, Colin Leavett-Brown, Michael Paterson, Ryan Taylor, Andre Charbonneau, Roger Impey, and Wayne Podiama. 2013. HTC scientific computing in a distributed cloud environment. In Proceedings of the 4th ACM workshop on Scientific cloud computing (Science Cloud '13). ACM, New York, NY, USA, 45-52. DOI=10.1145/2465848.2465850 `http://doi.acm.org/10.1145/2465848.2465850`

[13] D. Thain, T. Tannenbaum, and M. Livny. Distributed computing in practice: the HTCondor experience. *Concurr. Comput. : Pract. Exper.*, 17(2-4):323–356, Feb. 2005.

[14] Web Distributed Authoring and Versioning (WebDAV). `http://tools.ietf.org/html/rfc4918`.

[15] I. Vukotic *et al.*. Data federation strategies for ATLAS using XRootD. Proceedings of the CHEP 2013 Conference, Amsterdam 2013.

[16] I Gable *et al.*. Dynamic web cache publishing for IaaS clouds using Shoal, Proceedings of the CHEP 2013 Conference, Amsterdam 2013. arXiv:1311.0058 [cs.DC]

[17] Squids: a caching proxy for the Web. `http://www.squid-cache.org`.

[18] M Vliet *et al.*. Repoman: A Simple RESTful X.509 Virtual Machine Image Repository. Proceedings of the International Symposium on Grids and Clouds , Taipei, March 2011

[19] XRootD: a high-performance protocol for accessing HEP data. `http://xrootd.org`.