Exploiting Virtualization and Cloud Computing in ATLAS

Fernando Harald Barreiro Megino$^{\alpha}$, Doug Benjamin$^{\beta}$, Kaushik De$^{\gamma}$, Ian Gable$^{\delta}$, Val Hendrix$^{\epsilon}$, Sergey Panitkin$^{\zeta}$, Michael Paterson$^{\delta}$, Asoka De Silva$^{\eta}$, Daniel van der Ster$^{\alpha}$, Ryan Taylor$^{\delta}$, Roberto A. Vitillo$^{\epsilon}$, Rod Walker$^{\theta}$ on behalf of the ATLAS Collaboration

α: CERN: European Organization for Nuclear Research, Geneva, Switzerland
β: Duke University Durham, NC 27708
γ: The University of Texas at Arlington, TX, USA
δ: University of Victoria, BC, Canada
ϵ: Lawrence Berkeley National Lab, 1 Cyclotron Road, Berkeley, CA, USA 94720
ζ: Brookhaven National Laboratory, Upton, NY, USA
η: TRIUMF, Vancouver, BC, Canada
θ: Ludwig-Maximilians-Universitaet Munich

Abstract. The ATLAS Computing Model was designed around the concept of grid computing; since the start of data-taking, this model has proven very successful in the federated operation of more than one hundred Worldwide LHC Computing Grid (WLCG) sites for offline data distribution, storage, processing and analysis. However, new paradigms in computing, namely virtualization and cloud computing, present improved strategies for managing and provisioning IT resources that could allow ATLAS to more flexibly adapt and scale its storage and processing workloads on varied underlying resources. In particular, ATLAS is developing a "grid-of-clouds" infrastructure in order to utilize WLCG sites that make resources available via a cloud API.

This work will present the current status of the Virtualization and Cloud Computing R&D project in ATLAS Distributed Computing. First, strategies for deploying PanDA queues on cloud sites will be discussed, including the introduction of a "cloud factory" for managing cloud VM instances. Next, performance results when running on virtualized/cloud resources at CERN LxCloud, StratusLab, and elsewhere will be presented. Finally, we will present the ATLAS strategies for exploiting cloud-based storage, including remote XROOTD access to input data, management of EC2-based files, and the deployment of cloud-resident LCG storage elements.

1. Introduction

Emerging standards and software of the cloud computing paradigm bring attractive features to improve the operations and elasticity of scientific distributed computing. At the same time, the existing European Grid Infrastructure and Worldwide LHC Computing Grid (WLCG) have been highly customized over the past decade or more to the needs of the Virtual Organisations, and are operating with remarkable success. It is therefore of interest not to replace "The Grid" with "The Cloud", but rather to consider strategies to integrate cloud resources, both commercial and private (academic), into the existing grid infrastructures, thereby forming a *grid-of-clouds*. This work will present the efforts underway in the ATLAS Experiment [1] to adapt existing grid workload and storage management services to cloud computing technologies.

In mid-2011 the ATLAS experiment formed a Virtualization and Cloud Computing R&D project to evaluate the new capabilities offered by these software and standards (e.g. Xen, KVM, EC2, S3, OCCI) and to evaluate which of the existing grid workflows can best make use of them. In parallel, many existing grid sites have begun internal evaluations of cloud technologies (such as Open Nebula or OpenStack) to reorganize the internal management of their computing resources. In both cases, the usage of standards in common with commercial resource providers (e.g. Amazon, RackSpace) would enable an elastic adaptation of the amount of computing resources provided in relation to the varying user demand.

The remainder of this paper is organized as follows. Section 2 presents a list of cloud computing use-cases foreseen by ATLAS. The work in data processing in the cloud is presented in sections 3, 4, and 5. Next, section 6 presents the work towards using cloud-resident storage for caching and archival. Finally, we conclude and present future plans of the R&D project in section 7.

2. Cloud Computing Use-Cases

We have enumerated a number of use-cases which highlight the various ways in which cloud computing technologies can be exploited by ATLAS. They are generally grouped into Data Processing and Data Storage.

2.1. Data Processing Use-Cases

The first set of use-cases describes how ATLAS could generate or process data in the cloud. Due to the relative difficulties of exploiting cloud storage with the existing grid middlewares, most of these use-cases rely on the existing grid storage elements.

2.1.1. PanDA Queues in the Cloud: The primary goal of the *grid-of-clouds* approach is to deploy PanDA [2] processing queues for Monte Carlo (MC) production or distributed analysis. In this case, cloud resources are added to a new or existing PanDA queue and pilot jobs [3] are used to retrieve the payloads; however, the cloud site would not need a grid computing element.

These queues would allow jobs to run on either commercial or private (academic) cloud resources. In the first examples of this use-case, data stage-in and stage-out could use existing grid storage elements, but the use of cloud object storage services is not excluded. Data access to remote grid storage elements may be more or less practical based on varying I/O demands for different job types; for example, distributed analysis jobs are generally more I/O intensive and may present performance challenges, which could be ameliorated with data caching or other methods. Monte Carlo simulation jobs, having lighter I/O requirements, are ideal as a first use-case.

These queues are intended to be managed centrally and used for grid capacity bursting when more resources are needed and available. In fact, if the MC production work can be successfully offloaded to the cloud, then more of the existing grid resources could be dedicated to analysis.

2.1.2. Cloud Analysis Clusters: Users or sites should be able to easily deploy a new analysis cluster (e.g. a generic batch farm or PROOF cluster [4]) in commercial or private cloud resources. The goal of this activity would be to provide a "one-click" solution which would deploy all of the necessary virtual machines and software components needed to run ATLAS analysis. The success of such clusters will depend on efficient I/O with the local or remote storage.

2.1.3. Personal PanDA Analysis Queues: In situations where the conventional grid resources are fully occupied and an individual physicist has exhausted all of his or her fair share of the resources, it would be desirable to have a mechanism for the user to exploit private cloud resources or purchase extra resources from commercial cloud providers. Such a tool should hide the complexities of cloud computing and transparently run that user's own PanDA jobs on these new resources. For example, a user could submit an analysis job with the PanDA clients as usual and then execute another command to instantiate the necessary cloud resources to process those jobs.

These queues are intended to be non-centrally managed, with the underlying infrastructure being instantiated on demand by the users.

2.2. Data Storage Use-Cases

ATLAS should be able to make use of cloud storage for short and long term storage. Note that cloud storage can refer to both object stores such as Amazon S3 and OpenStack Swift, and also the disk provided with cloud IaaS instances (e.g. the disk associated with an EC2 instance that can optionally persist independently from the life of the instance).

2.2.1. Data Caching in the Cloud The aforementioned data processing use-cases would have improved performance if the I/O operations had a local caching layer. For instance, reads over the wide area network could be reduced with an Xrootd [5] local caching policy, or technologies such as HDFS could be used to build short-term storage clusters with the ephemeral disks.

2.2.2. Object Storage and Archival It may be beneficial for ATLAS to store DQ2 [6] datasets on cloud resources, thereby federating cloud storage with the conventional grid storage elements (which are accessible via the SRM protocol). In addition, HTTP-based data transfer protocols such as WebDAV and S3 may have advantages over gsiFTP (e.g. session reuse). The implications for third party transfers and the File Transfer Service [7] should also be studied.

3. Building PanDA Queues in the Cloud

The primary goal of the *grid-of-clouds* work is to deliver PanDA queues which transparently add cloud resources to the ATLAS grid infrastructure.

3.1. Condor and the Helix Nebula Science Cloud

The Helix Nebula Science Cloud is a European initiative between three laboratories (CERN, EMBL and ESA) to evaluate cloud computing resources from different European commercial IT providers. The ultimate goal is to set up a cloud computing infrastructure for the European Research Area and to identify and adopt policies for trust, security and privacy at the pan-European level. The general timeline of the project is shown in Figure 1. ATLAS is participating in the pilot phase and is committed to run real Monte Carlo production jobs on each of around five commercial providers.

The first provider to be tested was Cloud Sigma AG in Zurich. Given the short turn-around time per provider (around 1 month each), it was necessary to develop a lightweight approach to use these cloud resources. The basic architecture chosen was to build a worker node VM image with a Condor `startd` that is configured to connect back to a `schedd` at CERN. The VM images were based on CernVM [8] for convenience and the ATLAS software was accessed via CVMFS [9]. The Condor `schedd` at CERN already existed as one of the PanDA pilot factories. To maintain data simplicity, files were read and written over the WAN via both the CERN SRM interface and the native `root://` protocol of CERN's EOS [10] storage service.

These worker node servers at CloudSigma were cloned to a final deployment of 100 dual-core nodes (thus 200 job slots in total). To integrate this new Condor cluster with PanDA, a new queue was created (*HELIX*) and a stream of PanDA pilots were submitted to the cluster.

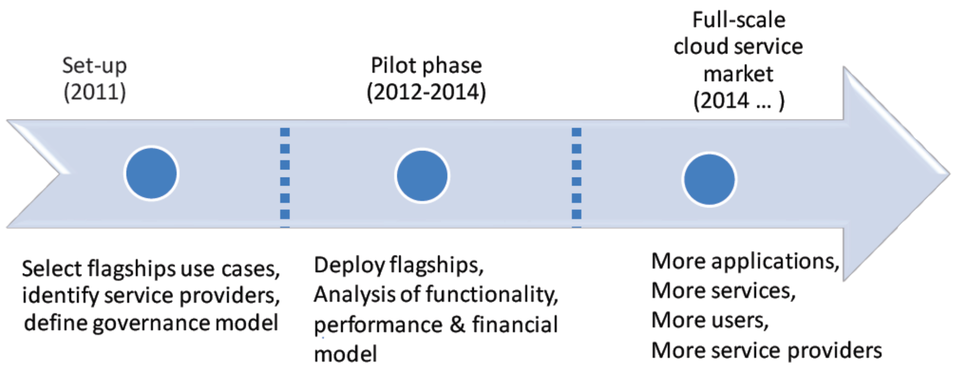

Figure 1. Timeline of the Helix Nebula Science Cloud project

Table 1. Helix Nebula Summary Statistics

	HELIX	**CERN**
Success Rates	265 failed, 6000 succeeded	36 failed, 1000 succeeded
Mean Running Times	$16267s \pm 7038s$	$8136.6s \pm 765.5s$

Following some short validations of the CPUs, a series of six MC production tasks (of 1000 jobs each) was assigned to HELIX and they were left to run for a period of almost 2 weeks.

The results of the CloudSigma HELIX jobs in comparison with a single similar task executed at CERN are shown in table 1. The success rates of HELIX and CERN are similar. However, it is clear that there is a high variance in the job running time at HELIX. This could be caused by resource contention with other CloudSigma users that have instances co-scheduled on the same bare-metal nodes. Also note that we show the mean running times for completeness, but it is not valid to compare the running times at HELIX and CERN since the hardware is very different. To get an ideal throughput comparison one should estimate the cost per event, though unfortunately the costs are not presently known at either site. The observed results need to be further investigated and fully understood.

In summary, the ATLAS tests of CloudSigma can be seen as a success; these were the first demonstration of real MC tasks running in the cloud for ATLAS. Work with CloudSigma and the other commercial providers related to Helix Nebula is ongoing.

3.2. The Cloud Scheduler Project

A second approach to running PanDA jobs in the cloud uses a software component called Cloud Scheduler [11]. Cloud Scheduler is a simple python software package designed to monitor the state of a Condor queue and boot VMs in response to waiting condor jobs. Cloud Scheduler can boot VMs on multiple IaaS cloud sites by interacting with cloud APIs at each site. This approach allows the user to run jobs on IaaS resources by simply submitting to a standard Condor queue using familiar commands. This same approach has been used successfully for BaBar jobs [12].

In order to integrate Cloud Scheduler with the existing PanDA grid infrastructure for ATLAS, two PanDA queues were established: *IAAS* for production jobs and *ANALY_VICTORIA-WG1-CLOUD_TEST* for analysis jobs. Since Cloud Scheduler can interact with multiple IaaS cloud sites, many cloud resources can be aggregated and served to these two queues. A new pilot factory was established at TRIUMF and is being configured to submit PanDA pilots to Cloud

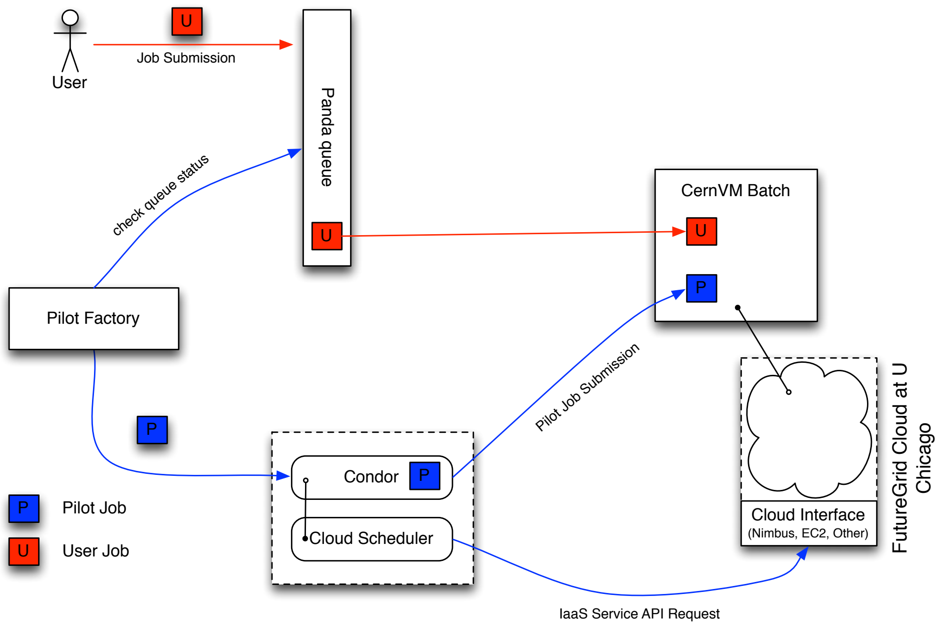

Figure 2. Integration of Cloud Scheduler with existing PanDA workload management system.

Scheduler as shown in Figure 2. The VMs booted were CernVM images configured in a similar fashion to those described in Section 3.1. The principal motivation for this design was to minimize changes to the existing ATLAS workload management system, and to be able to add new IaaS cloud resources quickly.

Cloud Scheduler depends on custom Condor job description attributes to identify the VM requirements of a waiting job, the most important of which are the name and URL of the VM images to boot, e.g. `+VMType = "vm-name" +VMLoc = "http://repo.tld/vm.img.gz"`. These extra attributes are used by Cloud Scheduler to determine what VM should be booted for each job, and are added to the standard pilot job attributes by the pilot factory.

A HammerCloud [13] benchmark was conducted on FutureGrid, a cloud provider in Chicago, U.S., to evaluate the performance and feasibility of running analysis-style jobs in the cloud using remote data access. Initial results were promising, with 692 successful jobs out of a total of 702. HammerCloud benchmark jobs are I/O intensive, and depend on significant wide-area network (WAN) bandwidth capacity between the sites. After a focused effort to improve the bandwidth between the two locations, data was transferred at several Gbps over the WAN, from a dCache storage element in Victoria to the booted VMs in Chicago. Two HammerCloud runs were conducted: one using WebDAV to stage in data and one using GridFTP. Each HammerCloud run consisted of 351 jobs, that downloaded 15GB in files sized between 0.8 and 1.6 GB. The WebDAV copy achieved better performance levels and download times (mean: 2007s, deviation: 1451s to download 15GB) than the GridFTP copy (mean: 2329s, deviation: 1783s to download 15GB), but more tests need to be completed to guarantee the statistical significance. Overall, the tests demonstrated the I/O capability of the WAN and FutureGrid infrastructure. Figure 3 shows the event rate and time to stage in data for the run which used WebDAV.

The production Panda queue (*IAAS*) has been put into continuous operation for running simulation jobs. Unlike analysis and HammerCloud jobs, these Monte Carlo simulation jobs have low I/O requirements, so they do not stress the network. Several thousand jobs have been executed, with over 200 VMs running simultaneously at FutureGrid and the Synnefo cloud at

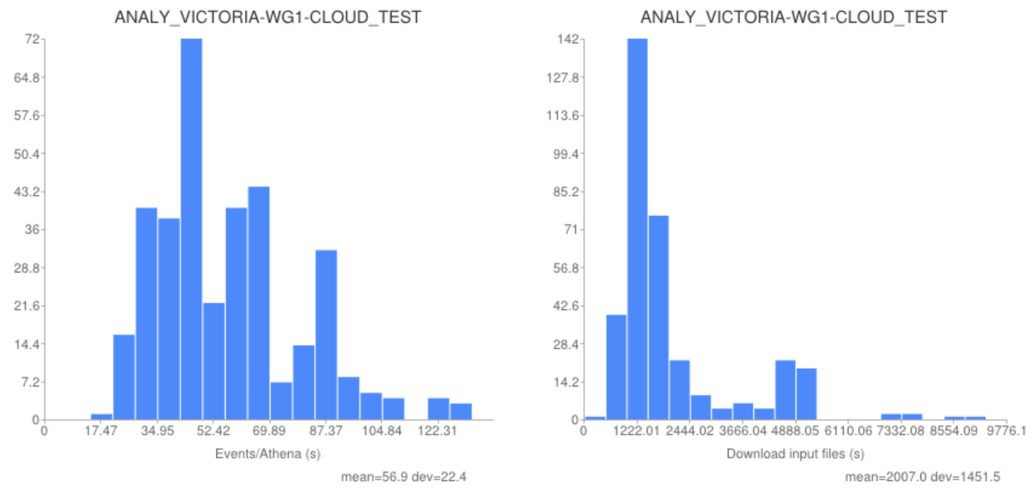

Figure 3. Event rate on the FutureGrid cloud at U Chicago (left). Time to download input files via WebDAV from the University of Victoria (right).

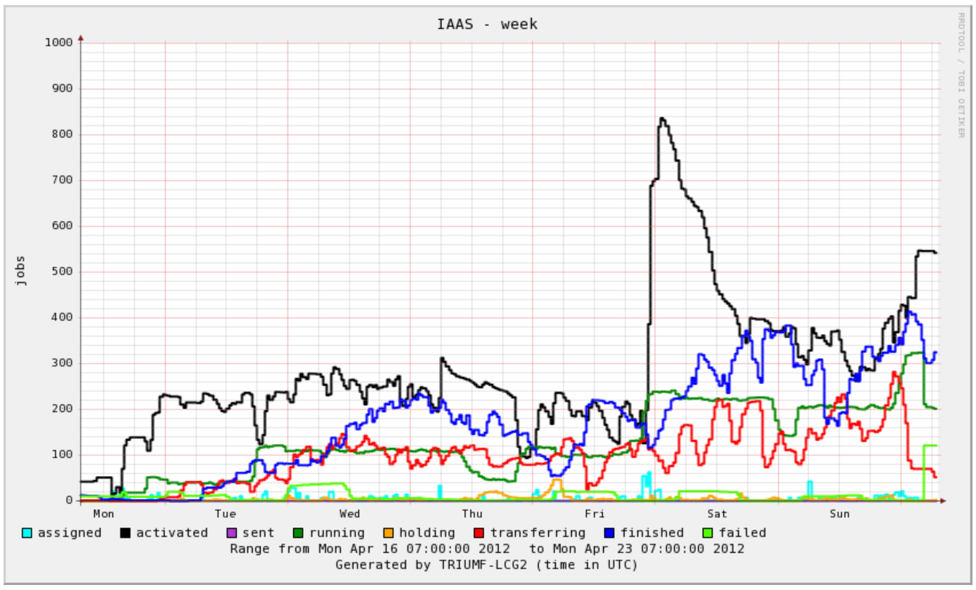

Figure 4. Production operation of *IAAS* PanDA queue managed by Cloud Scheduler, April 16 - 22, 2012.

the University of Victoria, as shown in Figure 4. We plan to scale up the number of running jobs as additional cloud resources become available.

4. Analysis Clusters in the Cloud

As previously mentioned, a site should be able to easily deploy new analysis clusters in a commercial or private cloud resource. It doesn't matter who runs a job in the cloud, which cloud they run on or who pays for it. The fact is that an easy and user-transparent way to scale out jobs in the cloud is needed so that scientists may spend time on data analysis, not system administration. This work [14] is an effort to meet these scientists' needs.

We have investigated tools for contextualizing cluster nodes and managing clusters. Our goal is to migrate all the functionality of a physical data analysis cluster (DAC) into the cloud. A physical DAC has services that support job execution and submission (condor), user management (ldap), ATLAS software distribution (cvmfs), web caching (squid), and distributed filesystems (nfs) to name a few. It is our plan to support these services in the cloud. Therefore, we need to find a cloud management tool that will enable us to define these DACs in all their complexity without burdening the scientist with its details.

4.1. Cloud Management Tools

CloudCRV [15], Starcluster [16] and Scalr [17] cloud management tools were chosen for our investigation. All of these tools are open source projects with varied levels of community support and features. We investigated their strengths and weaknesses for managing DACs.

4.1.1. CloudCRV , developed at Lawrence Berkeley National Lab (LBNL), uses a built-in web server to host a website that deploys clusters in one click. It doesn't require any external support tools and allows clusters to be managed (started and stopped) with a web interface. A thin client runs on each VM for communication and control purposes. On the CloudCRV server, an asynchronous event controller manages the lifecycle of the clusters, roles and virtual machines (VM). The clusters are written in python using a highly extensible API [18]. The roles are defined with the Puppet [19] configuration management engine which is a high-level language for defining the target state of a machine. These roles are applied to virtual machines so that they may work as a cluster of interdependent services.

CloudCRV is designed to be configuration and cloud provider agnostic. At this point in time, it only supports Puppet and cloud providers with an EC2 interface, but it could be extended to support others. Unfortunately, this software is in alpha stage and does not have a rich community of support.

4.1.2. Starcluster is an open source cluster-computing toolkit for Amazon's EC2 [20] written in python. It is a wrapper around Amazon EC2 command line tools (CLI) and provides its own CLI for configuration, management and scaling out of multiple clusters. There is a flexible plugin architecture that provides an easy way to extend the platform. Elastic load balancing is supported by shrinking and expanding the cluster based on a Sun Grid Engine queue.

Starcluster is a mature project with a very rich community. Its downside is that the management of the cluster must happen from the machine that has the configuration files. It also does not have monitoring and dynamic scaling of nodes.

4.1.3. Scalr is a commercial Software as a Service (SaaS) that allows you to launch, auto-scale and configure clusters in the cloud. The commercial product is actively developed and released by Scalr Inc. under an open source license [21]. Scalr provides a robust graphical user interface as well as a command line interface for managing multiple clusters on multiple cloud providers. It has integration with Chef [22], a configuration management tool by Opscode, and provides many features in customization, management and failover of clusters. Multi-cloud deployments are offered for Amazon EC2, Eucalyptus [23], Nimbus [24], Cloudstack [25], Rackspace [26] and soon Openstack [27]. Customization is allowed through its scripting engine, role importer, and API control.

Our comparison of these three open source cloud management tools has highlighted a clear leader. Scalr has the most robust feature set as well as an active community [28]. Unlike CloudCRV and Starcluster, Scalr provides customizable scaling algorithms, monitoring of load statistics for single nodes or entire clusters and has both a CLI and a web interface.

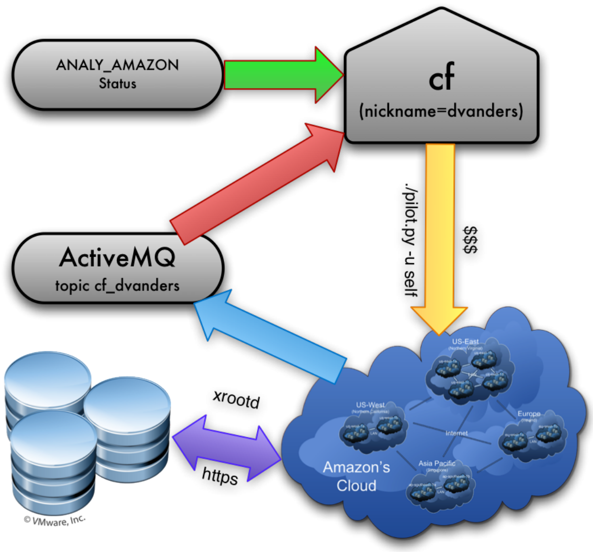

Figure 5. Workflow of the Cloud Factory Tool

5. Personal PanDA Analysis Queues in the Cloud

The availability of personal PanDA analysis queues would enable users to access extra computing resources on-demand; these extra resources would be sourced from private clouds or purchased from commercial cloud providers. One goal of personal PanDA queues is that they should look identical to the normal grid sites; users should be able to transparently access new cloud resources with zero or minimal changes to their data analysis workflow. For example, a personal analysis queue could be accessible by the user submitting their job with the PanDA clients and specifying the site `ANALY_AMAZON`. Note that in this case there would be no pilot factory associated with the virtual cloud site `ANALY_AMAZON`. Instead, this workflow would rely on the user taking some extra action to instantiate PanDA pilot jobs which would retrieve and run their jobs waiting in the `ANALY_AMAZON` queue. The PanDA pilot already has a *personal pilots* feature which will retrieve only the invoking users job from the PanDA server, ignoring any other user jobs that may be in the queue.

What is needed to enable the above workflow is a simple tool which senses the queued jobs in the personal queue, instantiates sufficient cloud VMs, runs the personal pilots, monitors the VMs, and destroys them after they are no longer needed. We call such a tool the *cloud factory*.

5.1. Cloud Factory

The requirements for a cloud factory tool are as follows:

- Manage the physical cloud VM instances needed to process jobs in a given logical queue (i.e. PanDA queue)
- Discover demand by querying the logical queue status
- Create instances via a cloud API according to demand
- Monitor instance states (alive/zombie, busy/idle, X509 proxy status)
- Destroy instances (zombies, expired proxy, unneeded)

ATLAS has developed a proof-of-concept cloud factory tool named `cf` (see Figure 5). This tool wraps around `cvm`, which is an EC2-compatible cloud instance management utility provided

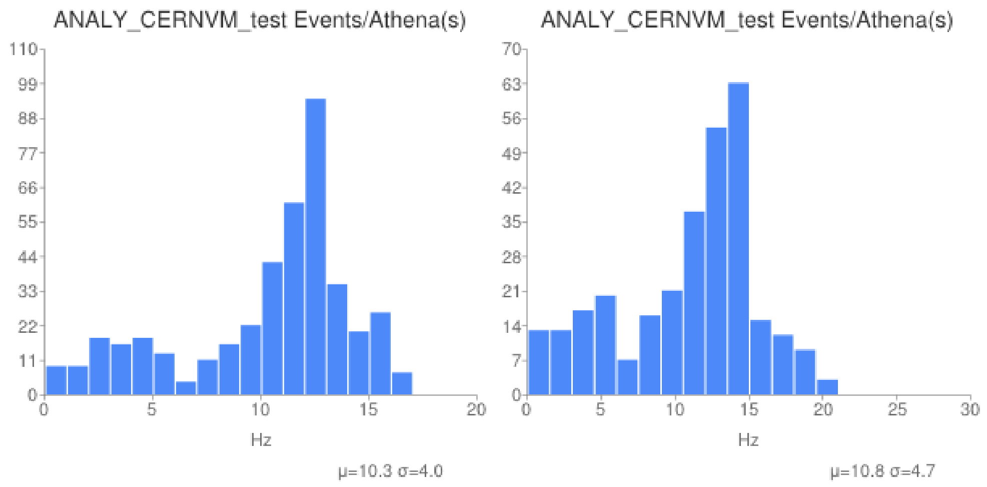

Figure 6. Performance comparison between LxCloud (left) and bare-metal (right).

with CernVM; `cvm` is used to instantiate, contextualize, and terminate instances of CernVM running in an EC2 cloud. `cf` includes three components:

cf: the tool itself, which queries the PanDA queue status and invokes `cvm` to instantiate instances.

cf-wrapper: a wrapper script which does some environment initialization and runs PanDA pilot on the cloud VM

cf-heartbeat: a script which runs in parallel with `cf-wrapper` on the cloud VM and monitors the status of the running instance

The heartbeat messages are communicated back to a message queue and report if the node is active or idle. When `cf` is periodically invoked, it retrieves the messages from the queue and by comparing with the activity in the PanDA queue it decides whether it should create new instances or terminate existing ones.

5.2. Cloud Factory Testing at LxCloud

LxCloud is an OpenNebula testing instance at CERN. Because resources are made available via the EC2 API, LxCloud is a convenient platform to evaluate `cf` and the personal PanDA queues workflow. We tested `cf` by first creating a queue `ANALY_CERNVM_test` and configuring it to get ATLAS releases from CVMFS and read data from the CERN storage facility. We then configured a series of HammerCloud [13] stress tests in order to measure the performance of the jobs running in the cloud. `cf` was used to run these test jobs with the same series of jobs run on both LxCloud and on identical bare metal via the CERN batch facility LSF; some results are shown in figure 6. No difference was observed in the reliability of virtualized versus bare metal hardware, while there was a small decrease in performance (from a mean of 10.8Hz in LSF to 10.3Hz in LxCloud).

6. Storage and Data Management

As mentioned in section 2, ATLAS sees two main use-cases for cloud storage: cloud-resident caches to accelerate data processing, and persistent object stores in the cloud for data archival.

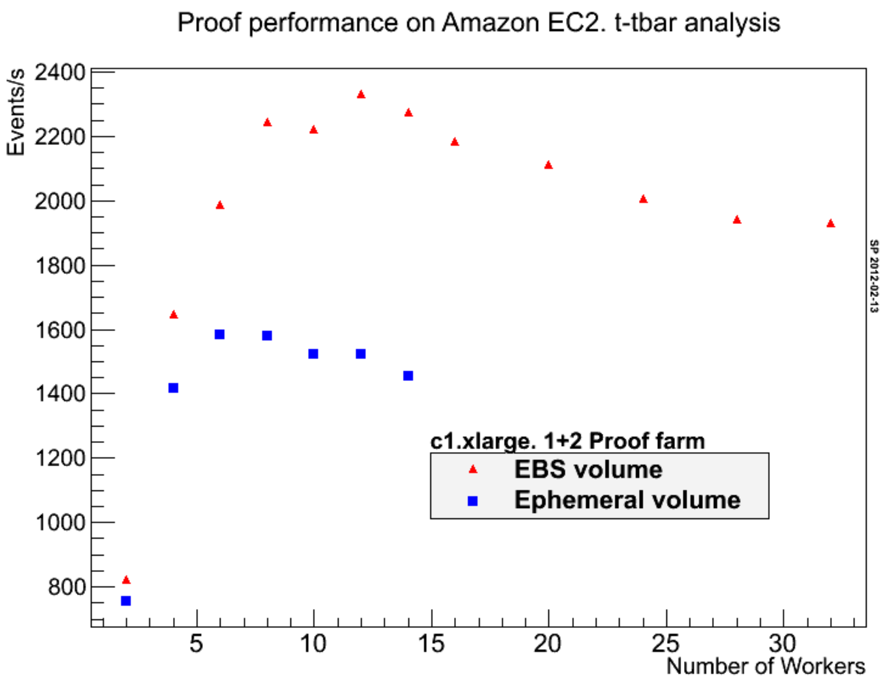

Figure 7. Performance comparison between EBS and ephemeral storage in ATLAS data analysis.

6.1. Data Access and Caching

Some of the most important issues associated with use of cloud computing for physics data analysis are data storage and data management. Different cloud platforms provide different storage abstraction implementations, and cloud storage is often multi-tiered in terms of performance, persistency and related costs. Amazon EC2 [20] provides a good example of this model. It offers at least three storage options: Simple Storage Service (S3), Elastic Block Store (EBS) and ephemeral store associated with a VM instance. When planning an analysis cluster model one needs to take into account all these factors. It is also clear that any evaluation and testing of cloud storage needs to be done with ATLAS codes and data.

In the course of our tests of Proof/Xrootd clusters on Amazon EC2, we studied some aspects of the cloud storage performance. For example, Figure 7 shows a performance comparison between EBS and ephemeral volumes. For these tests we utilized a three-node Proof farm running an ATLAS D3PD data analysis. One can see from Figure 7 that the EBS volume performed better than the ephemeral one. It is worth mentioning that ephemeral storage comes free with EC2 instances and also allows for scaling of storage space and performance with the size of the analysis farm. This example emphasizes the point that in planning for storage layouts in clouds, one needs to carefully analyze the cost-performance benefits of various storage layouts in connection to the particular analysis use-case.

We also tested direct read of ATLAS data located on ATLAS Grid sites from EC2 based analysis clusters. The main motivations for this were the free out-of-cloud data reads and greatly simplified in-cloud data management. For these tests we utilized both individual grid sites as well as the ATLAS federated Xrootd store as data sources. We found out that analysis performance in many cases was quite acceptable, if variable, and depended on the quality of the network connection from EC2 to a given storage site. This mode may also require tuning of the

Root WAN I/O parameters and is a subject of current active studies in ATLAS.

As part of these data caching tests, an Xrootd storage cluster was setup in the cloud. The storage cluster was composed of three data servers and a redirector node. Tests were performed on Amazon EC2 large instances. Both the Amazon ephemeral storage and Amazon elastic block storage (EBS) were used. The goal was to test the write performance of such a cluster where the data comes from outside of the cloud and is pulled into the cloud automatically.

As part of the automation of the data transfer, the XRoot FRM (file residency manager) daemon [29] was used to fetch the data files and copy them automatically to Xrootd storage partitions. The files were transferred using the Xrootd native copy program with the setting to fetch files from multiple sources. The copies were done in this way to ensure maximum transport speed.

In the initial tests, both ephemeral storage partitions provided with the EC2 large instance were used. Xrootd managed the volumes. This configuration proved to be unstable and resulted in many failures with files often not created. The most useable configuration was determined to be two ephemeral storage partitions joined together by Linux LVM into one partition. In this configuration, an average transfer rate of 16 MB/s to one data server was achieved (around 45 MB/s for the three servers). In this configuration, the startup time of the virtual machine would be increased due to the added time of assembling, formatting and configuring the storage.

In order to address the issue of data persistence, the storage cluster used Amazon Elastic Block Storage for the data volumes. In this way, when the data was not needed right away the partitions could remain on the storage to be available when the Xrootd storage cluster was reconstituted. A series of tests was performed using either one or two Elastic Block Storage partitions. The average transfer rate into the storage was less than 12 MB/s, with a large number of very slow transfers (less than 1 MB/s). In this configuration, storage costs would be an issue (the average amount of storage in an ATLAS local analysis site in the US is 50 TB). At current Amazon rates, the yearly costs would be around four times larger than the one-time cost of the existing storage which should be operational for at least 5 years. It should be noted that many universities provide power, cooling and space without additional cost to research groups, thus lowering the total cost of ownership for the storage.

6.2. Future Evaluation of Object Stores

In respect to persistent object stores, a new project has been started to investigate the possibility of integrating Amazon S3 with Rucio, the future version of the ATLAS Distributed Data Management (DDM) system. The DDM team will demonstrate the compatibility of their future system with the S3 API implementations of different storage providers (e.g. Huawei, OpenStack Swift and Amazon). The main questions that need to be answered are:

- how to store, retrieve and delete data from an S3 store and how to combine the S3 (bucket/object) and ATLAS (dataset/file) data organization models
- how to integrate cloud storage with the existing grid middleware
- how to integrate the various authentication and authorization mechanisms

The project will try simulating various ATLAS use-cases and will measure the storage performance over a timeline of one year.

7. Conclusion

This work has presented the past year of activities of the ATLAS Virtualization and Cloud Computing R&D project. The majority of the work has focused on data processing in the cloud. In this area, ATLAS has used Condor to rapidly deploy a production facility at commercial providers for the Helix Nebula Science Cloud project. The UVic work has delivered a Panda queue in permanent operation under which multiple cloud resources can be automatically aggregated and managed by Cloud Scheduler. Finally, the development of a personal pilot factory to allow individual users to access extra resources as needed is ongoing.

In the area of data caching and storage in the cloud, ATLAS is still working actively but has yet to achieve production-ready results. The studies into EC2 I/O performance in a PROOF cluster are promising, while the work to integrate cloud object stores into the distributed data management system is just beginning.

References

[1] The ATLAS Collaboration, The ATLAS Experiment at the CERN Large Hadron Collider, 2008 JINST 3 S08003 doi:10.1088/1748-0221/3/08/S08003

[2] T. Maeno, Overview of ATLAS PanDA Workload Management, Proc. of the 18th Int. Conf. on Computing in High Energy and Nuclear Physics (CHEP2010)

[3] P. Nilsson, The ATLAS PanDA Pilot in Operation, in Proc. of the 18th Int. Conf. on Computing in High Energy and Nuclear Physics (CHEP2010)

[4] http://www2.lns.mit.edu/proof/

[5] Alvise Dorigo et al.; XROOTD- A highly scalable architecture for data access. WSEAS Transactions on Computers, 1(4.3), 2005.

[6] M Branco et al. Managing ATLAS data on a petabyte-scale with DQ2. 2008 J. Phys.: Conf. Ser. 119 062017 doi:10.1088/1742-6596/119/6/062017.

[7] https://svnweb.cern.ch/trac/glitefts/wiki/FTSUserGuide

[8] P Buncic et al.; The European Physical Journal Plus 126(1) 1-8 10.1140/epjp/i2011-11013-1, "A practical approach to virtualization in HEP".

[9] J Blomer et al. 2011 J. Phys.: Conf. Ser. 331 042003, "Distributing LHC application software and conditions databases using the CernVM file system".

[10] Andreas J Peters and Lukasz Janyst. Exabyte Scale Storage at CERN. 11 J. Phys.: Conf. Ser. 331 052015 doi:10.1088/1742-6596/331/5/052015.

[11] P Armstrong et al. Cloud Scheduler: a resource manager for distributed compute clouds. arXiv:1007.0050v1

[12] I Gable et al. A batch system for HEP applications on a distributed IaaS cloud. 2011 J. Phys.: Conf. Ser. 331 062010 doi:10.1088/1742-6596/331/6/062010

[13] D. van der Ster et al. HammerCloud: A Stress Testing System for Distributed Analysis. 2011 J. Phys.: Conf. Ser. 331 072036 doi:10.1088/1742-6596/331/7/072036.

[14] https://indico.cern.ch/getFile.py/access?contribId=3&resId=1&materialId=slides&confId=180053

[15] https://indico.cern.ch/getFile.py/access?contribId=26&sessionId=5&resId=0&materialId=poster&confId=92498

[16] http://web.mit.edu/star/cluster/

[17] http://scalr.net

[18] http://code.google.com/p/cloudcrv/

[19] http://puppetlabs.com/

[20] http://aws.amazon.com/ec2/

[21] http://code.google.com/p/scalr/

[22] http://www.opscode.com/chef/

[23] http://www.eucalyptus.com/

[24] http://www.nimbusproject.org/

[25] http://www.cloudstack.org/

[26] http://www.rackspace.com/

[27] http://openstack.org/

[28] https://groups.google.com/forum/?fromgroups#!forum/scalr-discuss

[29] http://www.xrootd.org/doc/dev/frm_config.htm