Journal of Physics: Conference Series **513** (2014) 032066 doi:10.1088/1742-6596/513/3/032066

Commissioning the CERN IT Agile Infrastructure with experiment workloads

Ramón Medrano Llamas[α], Fernando Harald Barreiro Megino[α], Katarzyna Kucharczyk[β], Marek Kamil Denis[β], Mattia Cinquilli[α]

α: CERN, Geneva, Switzerland
β: Warsaw University of Technology, Warsaw, Poland

E-mail: ramon.medrano@cern.ch

Abstract. In order to ease the management of their infrastructure, most of the WLCG sites are adopting cloud based strategies. In the case of CERN, the Tier 0 of the WLCG, is completely restructuring the resource and configuration management of their computing center under the codename Agile Infrastructure. Its goal is to manage 15,000 Virtual Machines by means of an OpenStack middleware in order to unify all the resources in CERN's two datacenters: the one placed in Meyrin and the new on in Wigner, Hungary.

During the commissioning of this infrastructure, CERN IT is offering an attractive amount of computing resources to the experiments (800 cores for ATLAS and CMS) through a private cloud interface. ATLAS and CMS have joined forces to exploit them by running stress tests and simulation workloads since November 2012.

This work will describe the experience of the first deployments of the current experiment workloads on the CERN private cloud testbed. The paper is organized as follows: the first section will explain the integration of the experiment workload management systems (WMS) with the cloud resources. The second section will revisit the performance and stress testing performed with HammerCloud in order to evaluate and compare the suitability for the experiment workloads. The third section will go deeper into the dynamic provisioning techniques, such as the use of the cloud APIs directly by the WMS. The paper finishes with a review of the conclusions and the challenges ahead.

1. Introduction

During the commissioning of the new private cloud infrastructure at CERN, the IT Department has offered ATLAS and CMS some non-trivial amount of resources in order to stress test the infrastructure prior to the stable release.

ATLAS and CMS have joined forces in order to benefit from the infrastructure to run simulation tasks and to optimize their workload management systems (WMS) to make efficient use of the cloud resources before there is a generalized move towards this model. This paper presents the work performed since November 2012 in order to integrate and optimize the cloud resources at CERN, based on an OpenStack cloud middleware, within the current experiment workloads. This work is the first approach of a cloud based grid execution at CERN and has inspired other work, such as the usage of the high level trigger (HLT) farms both in ATLAS and CMS [1,2].

The paper is structured as follows: the first section outlines the different WMS used in ATLAS and CMS and gives an introduction on how the integration with cloud resources can be performed, the following section summarizes the performance and stress testing performed with HammerCloud; after,

the new possibilities of dynamic provisioning and automatic resource management are investigated. The paper finishes with the conclusions and an outlook for future improvements.

2. Integration with experiment workloads

The main requirement of this work is to integrate cloud resources, in their infrastructure as a service (IaaS) incarnation within the current tool set for grid submission used by the users. The existence of the cloud backend should be transparent for the users, while the WMS has to make optimal use of the resources.

The CERN IT Department provided resources enough to create clusters of up 800 cores using the new private cloud, codenamed Agile Infrastructure. This cloud service is based on OpenStack (we worked with Essex, Folsom and Grizzly over time) and is expected to manage more than 15,000 servers by 2015, when the new data centre in Hungary is fully operational [3].

2.1. ATLAS PanDA

All the ATLAS collaboration submits analysis and simulation jobs by means of PanDA. This is a pilot job based framework with two important components: the PanDA server, which holds all the job information and the pilot factories. Those are the most important part of the infrastructure for this work, since they are responsible of making the pilot jobs arrive to the worker nodes, which then will fetch actual jobs (payloads) from the PanDA server [4].

In the classic grid deployment of the pilot factories, they are based on an HTCondor-G submission mechanism, which allows sending the pilot jobs as grid jobs. The deployment made for the private cloud inherits the experience gained from other smaller scale tests made with commercial providers or inside the Helix Nebula consortium by setting up the virtual machines as worker nodes of an HTCondor cluster.

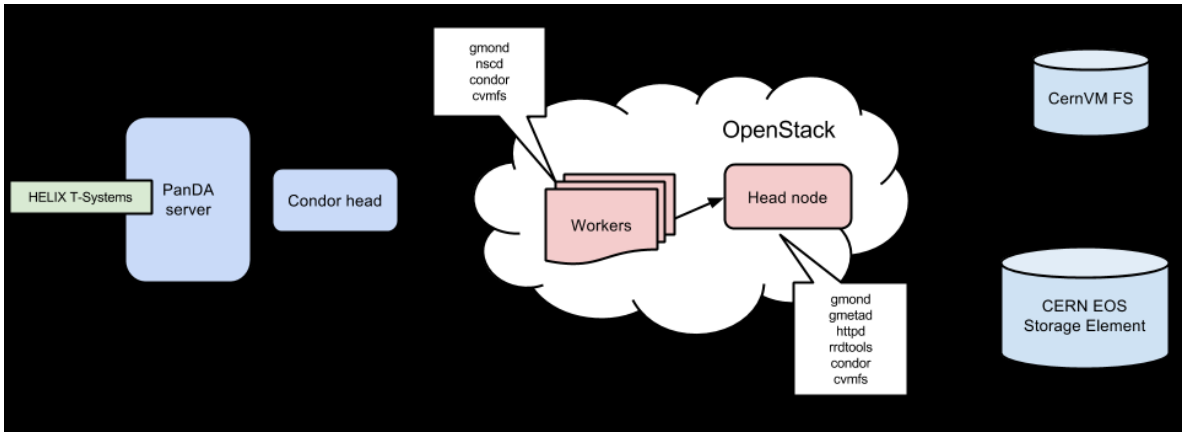

Figure 1: Schematic of the ATLAS deployment for cloud backed infrastructures

All the master machines for the cluster are deployed at CERN in a *vobox* schema, which allows an easy way to deploy the worker node machines either inside the local data centre or any other place in the world, as they will join an HTCondor pool that is already been feed by PanDA with pilot jobs.

All the data access is performed through the EOS storage element, sitting at CERN and the CVMFS distributed file system, which is used for the software access.

The provisioning of the virtual machines is done statically, requesting the cluster beforehand, which will then join the PanDA infrastructure automatically.

2.2. CMS glideinWMS

The CMS Computing Operations team has been testing and deploying the glideinWMS (glidein) for some time with a more than reasonable amount of success. One of the best features of this WMS is the ability to perform dynamic orchestration of cloud resources automatically.

From the user point of view, the submission of a job is just a regular HTCondor submission, while behind the scenes, provisioning of resources is performed and a pilot based framework makes the virtual machine become a worker node by getting the HTCondor binaries from the frontend machine.

glidein requires that the cloud middleware offers an EC2 compatible interface in order to access the resources, on the other hand, this is the only special requirement of the system.

As on the ATLAS deployment, access to the data and to the software is performed via EOS and CVMFS respectively. As on the ATLAS deployment, all the HTCondor head machines are sitting locally at CERN.

3. Functional and stress testing

The testing of the cluster was made by means of the HammerCloud [5,6] testing as a service framework, both from a functional perspective and from a performance perspective, by running stress tests on the clusters.

Stress testing this infrastructure is always a challenge since the tests have a very fast turnover. They are very short simulation jobs that are intended to test the underlying infrastructure and the grid scheduler, in this sense, there is no interest in running a job during 10-12 hours of computing, so they are configured to process a small amount of events during 15-20 minutes. As a result, HammerCloud will schedule a big number of jobs per day, stressing the grid dispatcher, the scheduling, caches and cloud orchestration.

3.1. Functional testing

The functional testing was performed through the regular framework of HammerCloud monitoring that the experiments have in place, namely, the production functional tests (PFT) for ATLAS and the functional set of tests for CMS.

In this testing, HammerCloud was submitting a stream of different types of simulation jobs constantly, exactly the same jobs that are used to monitor other grid sites.

The results do not defer much of those obtained from other grid sites, such as the CERN bare metal site. There are issues with regard to actual infrastructure that might be useful to distinguish whether the OpenStack cloud is failing or the WMS is failing.

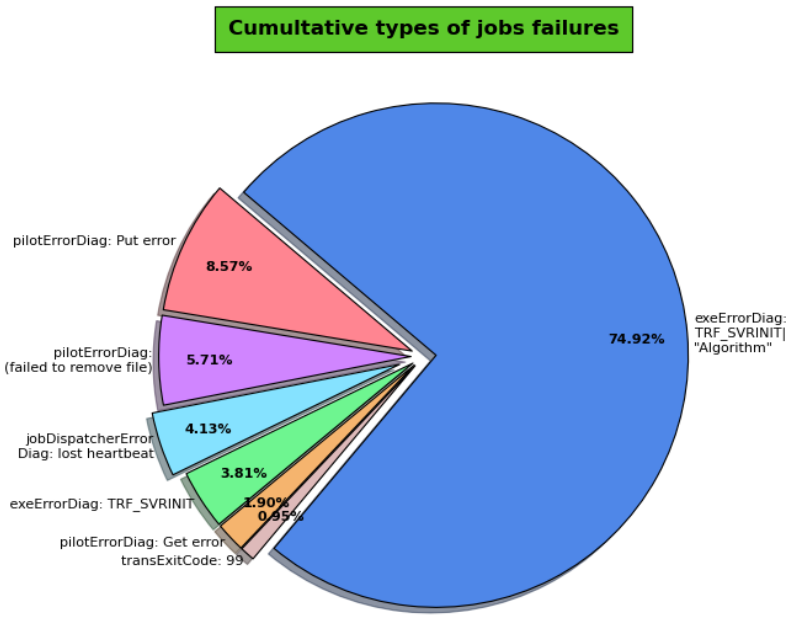

Figure 2: Job failure analysis for the ATLAS jobs on the Agile Infrastructure

The three most common errors are clearly related with the ATLAS infrastructure, for instance the initialization of the transformation of an error on credentials with accessing the storage element

The PanDA lost heartbeat error is one of these, however, giving its generality it's very difficult to use as an indicator of an infrastructure failure. The PanDA pilot job sends a heartbeat to the PanDA server every 30 minutes and if it does fail, the server will kill the job after some timeout. A failure on this heartbeat by many different reasons on the stack and could be unacceptable to account these errors as an infrastructure failure.

On the CMS side, glidein is responsible for the lifecycle management of the VMs and will have much more detailed information about infrastructure errors, such as failures for the machine requests, turn over timings and several statistics. This information is collected by the frontend.

3.2. Stress testing

At the same time that we were using HammerCloud for the functional monitoring, we used the tools to submit stress tests that could help us understand how the cloud infrastructure was performing, in comparison with other clouds and the bare metal batch system. We needed these tests to further optimize the virtual machine configuration and the scheduler setup.

In grand total, more than one million test jobs were submitted over two periods of two months (the rest of the time, regular simulation tasks where executed). The success ratio and a summary of the performance metrics are following:

ATLAS	
Scheduler	PanDA
Cluster management	Static
Cluster size	770 cores
Jobs submitted	694,698
Failure rate	9.95%
Job type	Simulation
Typical job duration	31 min.
Duration variance	17.8 min.
Most common error	Failed to read LFC

CMS	
Scheduler	gideinWMS
Cluster management	Dynamic
Cluster size	200 cores
Jobs submitted	337,080
Failure rate	0.31%
Job type	Simulation
Typical job duration	9 min.
Duration variance	4.8 min.
Most common error	App. Error 8020

Table 1: Stress test parameters for ATLAS and CMS

In grand total, more than one million test jobs were submitted over two periods of two months (the rest of the period, general processing was performed).

3.3. Other infrastructure testing

Giving the ease to port this schema to other providers, we performed some other tests on several providers, such as Rackspace on its London datacentre.

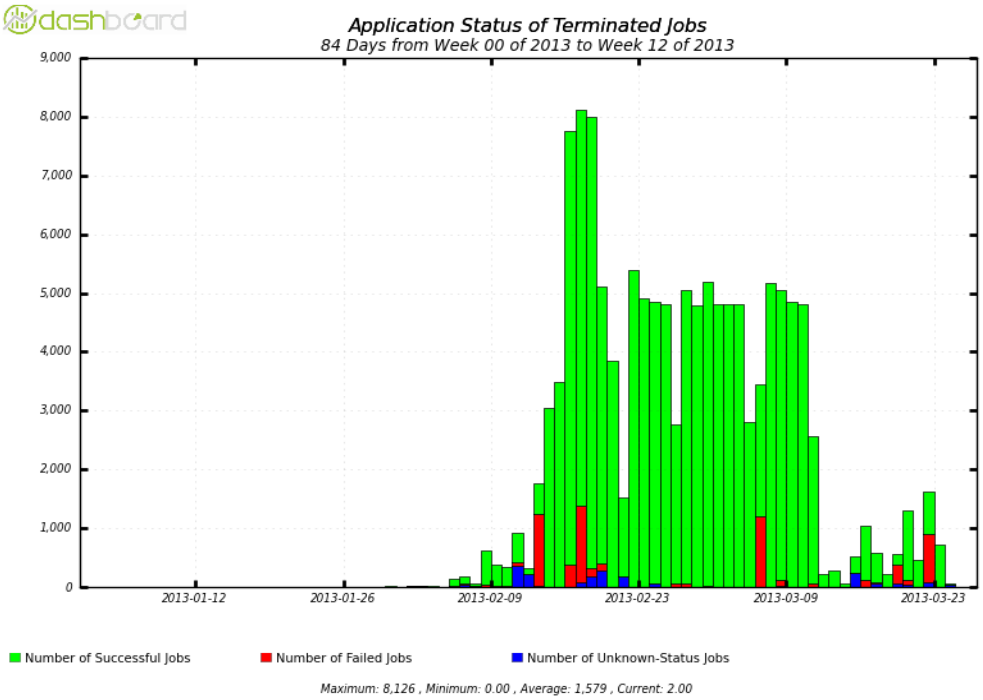

Figure 3: Job submission profile during CMS tests

The results are engaging, since with exactly the same deployment schema, we are able to exploit resources on private providers with the same level of job completion efficiency. This can eventually be used as a bursting out strategy or even to set up small T2 equivalent sites in a transparent way.

4. Dynamic provisioning

The CMS infrastructure uses glidein, built on top of HTCondor, which in turn it tries to correlate available virtual resources with the current requested workload. This is extremely useful, especially when one has to pay for resources utilization. When a new VM is booted up it parses the data transferred from the requesting server (condor master in this case). The data includes most crucial information like trusted certificates and poll master endpoints. The glidein client discovers the resource (CPUs, RAM and storage available) and connects to the master, awaiting for new jobs to come.

When the reasonable amount of jobs is computed or the timeout is raised the VM kills itself. Thanks to that, the cluster may scale down when the workflow is not too big. It also has significant security advantages (no malware or viruses will persist probably doing harm on the VM). In case of some software problems (memory leaks, not cleaning storage space) deleting and later booting up fresh VM may be very useful too. But also might hide these problems on the long term.

Currently, the glidein scheduling algorithm tries to adjust the cluster size to the workload, which might not always be correct. Starting with a fresh cluster and uploading many short lasting jobs (5-10 minutes) may result in glidein trying to boot up many VMs (as is the case of a HammerCloud stress test). Taking as propagation time the elapsed time between the booting request and VM creation, most of those short jobs can be already computed by that time. This may result in wasting of resources, especially since the timeout values are usually set to hours if not days.

On the other hand it is very hard to predict the optimal number of cores in the pool, especially where job duration is one of the key parameter. A possible workaround is either fixing the scheduling algorithm, where jobs' computing duration would be taken into consideration or manual operating the workload size, so it scales up steadily.

5. Conclusions and future work

Although the work done shows the potential of cloud backed resources, being very efficient and flexible to use via the WMS of the experiments, there is still work to do in several areas. Namely, the creation and configuration of the images must be streamlined in order to save effort and to make the worker node prototype portable to any other cloud in a simple way [7].

Also, given that both WMS used are based on a pilot framework, which involves the use of a certificate for the pilot job (which has many privileges, for instance, getting user certificates proxies), the analysis of the security concerns of using contextualization and user data transferal mechanisms of third party clouds must be carried out.

References

[1] The CMS High Level Trigger. Trocino, Daniel. Computing in High Energy and Nuclear Physics (CHEP). 2013.

[2] Usage of the CMS Higher Level Trigger Farm as a Cloud Resource. Cinquilli, Mattia and Colling, David and Grandi, Claudio. Computing in High Energy and Nuclear Physics (CHEP). 2013.

[3] Production Large Scale Cloud Infrastructure Experiences at CERN. Moreira, Belmiro and van Eldik, Jan and Schwickerath, Ulrich and Castro Leon, Jose. Computing in High Energy and Nuclear Physics (CHEP). 2013.

[4] Next Generation PanDA Pilot for ATLAS and Other Experiments. Barreiro Megino, Fernando and Caballero, Jose and De, Kaushik and Hover, John and Love, Peter and Maeno, Tadashi and Medrano Llamas, Ramon and Walker, Rodney and Wenaus, Torre. Computing in High Energy and Nuclear Physics (CHEP). 2013.

[5] Grid Site Testing for ATLAS with HammerCloud. Medrano Llamas, Ramon and Legger, Federica and van der Ster, Daniel and Sciacca, Giovanni. Computing in High Energy and Nuclear Physics (CHEP). 2013.

[6] Testing as a Service with HammerCloud. Medrano Llamas, Ramon and Elmsheuser, Johannes and Legger, Federica and van der Ster, Daniel and Sciacca, Giovanni and Barrand, Quentin. Computing in High Energy and Nuclear Physics (CHEP). 2013.

[7] ATLAS Cloud Computing R&D. Panitkin, S and Barreiro Megino, F and Caballero Bejar, J and Benjamin, D and DiGirolamo, A and Gable, I and Hendrix, V and Hover, J and Kucharczuk, K and Medrano LLamas, R and Love, P and Ohman, H and Paterson, M and Sobie, R and Taylor, R and Walker, R and Zaytsev, A. Computing in High Energy and Nuclear Physics (CHEP). 2013.