PROCEEDINGS OF SCIENCE

Virtual Facility at Fermilab: Infrastructure and Services Expand to Public Clouds

Steven Timm
Fermilab
P.O. Box 500, Batavia, IL 60510, USA
E-mail: timm@fnal.gov

Gabriele Garzoglio
Fermilab
P.O. Box 500, Batavia, IL 60510, USA
E-mail: garzogli@fnal.gov

Stuart Fuess
Fermilab
P.O. Box 500, Batavia, IL 60510, USA
E-mail: fuess@fnal.gov

Glenn Cooper
Fermilab
P.O. Box 500, Batavia, IL 60510, USA
E-mail: gcooper@fnal.gov

Abstract:

In preparation for its new Virtual Facility Project, Fermilab has launched a program of work to determine the requirements for running a computation facility on-site, in public clouds, or a combination of both. This program builds on the work we have done to successfully run experimental workflows of 1000-VM scale both on an on-site private cloud and on Amazon AWS. To do this at scale we deployed dynamically launched and discovered caching services on the cloud. We are now testing the deployment of more complicated services on Amazon AWS using native load balancing and auto scaling features they provide.

The Virtual Facility Project will design and develop a facility including infrastructure and services that can live on the site of Fermilab, off-site, or a combination of both. We expect to need this capacity to meet the peak computing requirements in the future. The Virtual Facility is intended to provision resources on the public cloud on behalf of the facility as a whole instead of having each experiment or Virtual Organization do it on their own. We will describe the policy aspects of a distributed Virtual Facility, the requirements, and plans to make a detailed comparison of the relative cost of the public and private clouds. This talk will present the details of the technical mechanisms we have developed to date, and the plans currently taking shape for a Virtual Facility at Fermilab.

PoS(ISGC2015)014

PoS(ISGC2015)014

The International Symposium on Grids and Clouds (ISGC) 2015
March 15-20, 2015
Academia Sinica, Taipei, Taiwan

1. Introduction

The Fermilab scientific program includes several running experiments, both the CMS experiment at the Energy Frontier, and the various neutrino and muon experiments on the Intensity Frontier. The ongoing data analysis and simulation for running experiments, combined with a large simulation load for future facilities and experiments, results in an unprecedented level of computing demand. This paper describes recent progress in the ongoing program of work to expand our computing to the distributed resources of grids and public clouds. We will also describe our plans for the Virtual Facility Project, which will integrate these technologies into our computing facility.

1.1 Definitions

In this paper the Facility refers to all computing resources and storage resources that are provided on behalf of the users. This includes on-site resources, friendly grid sites, opportunistic grid usage, and use of private, community, and commercial clouds. The Virtual Facility consists of the resources that we dynamically provision, plus the services that allow us to provision virtual machines and grid slots, as well as the auxiliary services that are required to make analysis possible at remote sites such as the public cloud. Provisioning is defined as the process of contacting grid and cloud sites on behalf of the users to obtain batch slots for job execution.

1.2. Background

The process of provisioning job slots has historically been done by individual Virtual Organizations. At Fermilab the GlideinWMS system has been used for this purpose. It is designed to request resources from a pre-configured set of hosts, both local and remote. These resources are presented as a unified virtual resource pool to the users, commonly referred to as the "user pool". The combination of unified provisioning and automated credential management has proven a powerful combination to get thousands of users doing grid computation on a regular basis. The developers of GlideinWMS and the HTCondor system on which it relies have worked with various cloud providers over the years to support submission of virtual machines to various public and private clouds including Amazon Web Services, Google Compute Engine, OpenNebula, and OpenStack. The virtual machines submitted by GlideinWMS start up a condor daemon, which calls back to the user pool and is ready to accept a job.

1.3 Business Case for Commercial Clouds

The workloads of our current experiments are stochastic in nature. There is some amount of continuous usage, punctuated by large peaks of usage. This is often related to extra analysis that must be done just before conferences. The peaks of demand can be up to four times the regular usage. We believe it is not financially feasible to size our computing clusters to handle the absolute peak demand. Nor can we rely on opportunistic grid slots to fill the gap because when our site is under high pre-conference demand the rest of them are likely to be

busy as well. It is therefore worthwhile to carefully consider commercial clouds as an option to meet this peak demand. We know it is technically possible to run on the commercial cloud. What we need to demonstrate now is sustainability and efficiency, as well as financial feasibility.

2. Workflows on cloud to date

As part of the joint collaboration between Fermilab and KISTI, we have a program of work building towards distributed federated clouds. In the summer of 2014 the primary goal of this program was to demonstrate a federated cloud running at the 1000 Virtual Machine scale, using our local private cloud nodes and Amazon Web Services EC2. Our application of choice for this is the Cosmic Ray simulation of the NOvA experiment far detector at Ash River. This application requires negligible input and produces about 250MB of output per job, and is quite computationally intensive. The NOvA experimenters spent considerable effort in optimizing their code to run at various sites outside of Fermilab, including loading their code into the OSG OASIS CVMFS server. The NOvA experiment supplied us with a set of files and scripts, which would run one full set of their cosmic ray Monte Carlo, 20000 input files in all, with one job per file.

2.1 Scaling OpenNebula 4 cloud to 1000 Virtual Machines

OpenNebula cloud supports launching virtual machines via its emulation of the EC2 interface. The OpenNebula 3 cloud in production since 2012 supported this function but an OpenNebula upgrade to version 4.8 was necessary to support the bulk launching at the 1000 Virtual Machine level. We also added 140 worker nodes for the purpose of this test. These were 8-core Dell PowerEdge 1950 servers that were formerly part of the CDF clusters at Fermilab. The worker nodes were attached to a routable private network, which could access all network points inside of Fermilab but not outside. We used the Bluearc NAS server as an image store to distribute the operating image, which is stored in the compressed “qcow2” format. A puppet script applied at boot time did configuration management on the image. This was used to install the CVMFS client software and a few other prerequisites including certificate authority files. Since the goal was to test 1000 virtual machines, we launched 1000 virtual machines with one core apiece. Under normal operation we would run virtual machines with four cores apiece.

2.2 Scaling Amazon Web Services to 1000 Virtual Machines

Earlier smaller trials of running jobs on AWS had shown us that the bottleneck was the proxy caching of the code on AWS. We had been using a Squid proxy server at Fermilab to cache the CVMFS code, which is not optimal for security reasons or for bandwidth reasons. So we found it necessary to launch one or more squid services in AWS itself. Since these services are dynamically instantiated there is no a priori way to know what the IP address of the squid server will be at any given time. We used the SHOAL discovery system that was written for this purpose by the team at the University of Victoria. In this system, each squid server launched in the cloud runs a Shoal Agent that contacts a fixed server at Fermilab via the AMQP messaging protocol. Each worker node VM then runs a client which calls the Shoal server at Fermilab which returns a list of possible squid servers, sorted by order of which is closest to the local VM network-wise. In practice we found that a single squid server in the cloud was sufficient to serve 500 virtual machines.

We also required a faster way to upload specialized virtual machines to Amazon Web Services. It was our goal to run a very similar virtual machine image on Amazon as we run on our private cloud, based on Scientific Linux. We leveraged our existing mechanism that manufactures our stock image for the private cloud and added extra steps to it that strip out a few Fermilab-specific configurations and add a few Amazon-specific configurations. Then it is copied to another running image on AWS which has an extra disk mounted. The extra disk is then saved as a snapshot. Once the virtual machine image is stored on Amazon then all copies of the virtual machines are launched from that. We settled on the Amazon m3.large instance type because it had two available cores and enough default scratch space (30GB SSD) to host two jobs at once.

2.3 Results

Figures 1a and 1b show the results of the final and largest trial, in which 1000 jobs ran simultaneously both on our local private cloud and on Amazon Web Services. The completion time of the jobs was relatively the same. The ramp-up time for jobs on our local cloud was more due to the virtual machine launching pattern we were using at the time, which caused all 8 of the virtual machines to launch on the same node simultaneously, significantly stressing the local disk. We have since switched to a scheduling algorithm, which distributes the launch more equally across the cloud.

We ran a total of 3300 jobs on AWS in the largest trial, shown in Figure 1a. Each job generated on average 250MB of output, the total output was 467GB for which we incurred $51 in data transfer charges. We incurred $398 in virtual machine charges, with a peak of 525 virtual machines running.

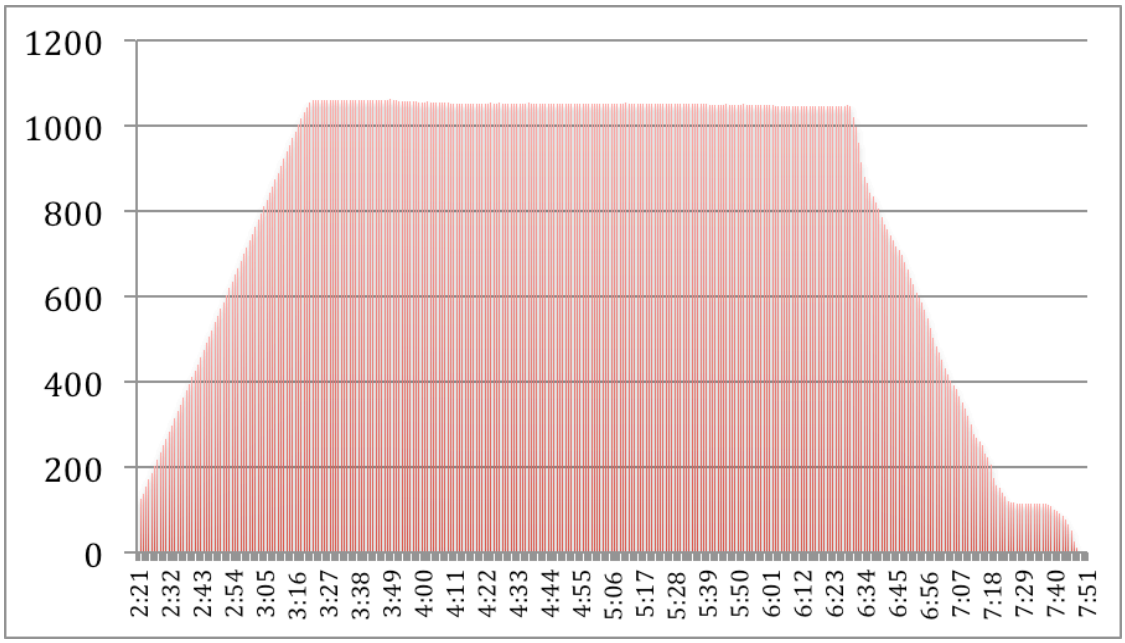

Figure 1a. Number of jobs running as a function of time of day, Amazon AWS

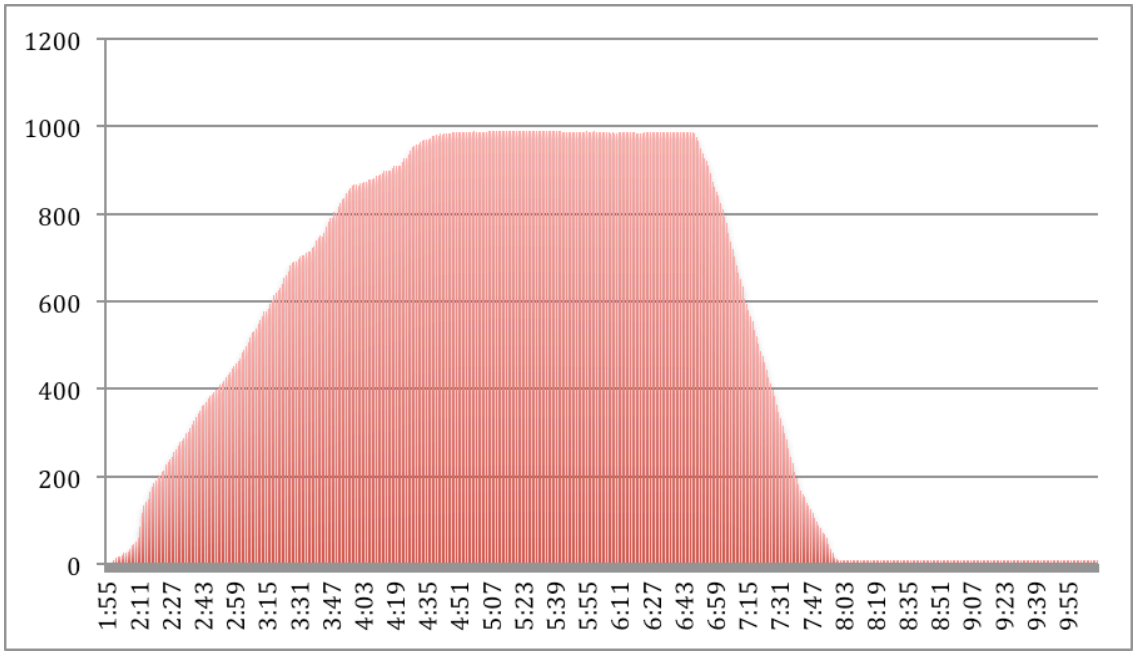

Figure 1b. Number of jobs running as a function of time of day, Fermilab private cloud

3. Virtual Facility—Expanding the facility to the public cloud

It is now the goal to integrate the commercial cloud running of batch jobs into the Fermilab Facility on a transparent basis. We currently have a unified job submission service based on the "Jobsub" client/server system. Users can currently specify usage model of private cloud or paid cloud to run on the private cloud or the paid cloud respectively. It is our goal to shift this paradigm so that the user will specify flags that specify whether their job can run off-site or not, and then the facility decides transparently whether or not to send it to the commercial cloud. We also plan to incorporate the provisioning functions of GlideinWMS into the facility. In this way we will be able to request resources on behalf of the whole facility with a single set of credentials, rather than requesting them on behalf of each of the approximately 20 active experiments we support. We describe below some of the major work items that will be necessary to do in this process.

3.1 Data intensive production

Intensity Frontier data production is typically more data intensive than the cosmic ray example that we used in the summer 2014 demonstration. For the commercial clouds to be a transparent extension of our computing facility, we will have to successfully and regularly do data-intensive data reconstruction and Monte Carlo simulation on the public cloud. For example, a typical data reconstruction campaign for the NOvA experiment is expected to have 10000 input files on average, 250MB in size, and produce one 250MB output file for each, taking 3 hours per job. A typical Monte Carlo simulation campaign of a neutrino beam could take 38000 input files of average 250MB in size and make 250MB of output, taking up to 5 hours per job. We have proposed a program of running six data reconstruction campaigns and ten Monte Carlo simulation campaigns on Amazon over the course of the next year. This will be approximately 2.1 million hours of compute time overall. The proposed program is expected to cost approximately $30,000 for the compute and data transfer, if AWS Spot Pricing is used to purchase all of the compute cycles. For scale, the NOvA experiment used approximately 10.2 million hours of compute time total in 2014, and Fermi Grid as a whole delivered 145 million CPU-hours in calendar year 2014.

Given the increased amount of data that will be produced by these jobs, we believe it will be necessary to do local caching of the output results on the S3 service of Amazon before bringing the data back to Fermilab. In this way we do not have hundreds of virtual machines waiting for 30-45 minutes just to transfer their data back to Fermilab. There will likely be some modifications necessary to our higher-level data handling software to support this.

3.2 Provisioning Algorithms

Currently GlideinWMS operates on a demand-based system. If there are user jobs which request a particular resource type such as off-site cloud, then GlideinWMS will continue trying to contact the off-site resource and to gain more of those resources, until there are a fixed

number of idle jobs waiting and it is apparent that the resource is full. Since commercial clouds have very large capacity and cost us for each machine we start, a different model is required.

We have proposed a call-out architecture to enable the GlideinWMS Frontend to call out to an external routine to determine whether it should request more glideins from the public cloud or not. The goal of this routine will be to optimize the job placement based on expected execution time as well as overall cost. In general if the local resources are full, and expected to remain full for the length of time that it takes to launch virtual machines on AWS, then we would launch VM's on AWS. We may, however, wish to restrict AWS only to certain users.

A number of other small improvements will be needed to the GlideinWMS system. These include better support for AWS spot pricing, capacity to submit simultaneously to multiple AWS regions and availability zones, selection of various resource types and AMI id's.

3.3 On-Demand Services in the Commercial Cloud

A number of auxiliary services may eventually be necessary in the public cloud. We certainly need Squid for code caching via CVMFS. We may well need to have a temporary storage element on the public cloud to aid in data caching inbound and outbound. We might need to set up a CVMFS Stratum 1 server or a node that serves Alien Cache (an extension to CVMFS that allows for serving large files locally that are not easily distributed by Squid).

We demonstrated the Squid service operation in the cloud previously, but it had to be manually launched. It would be preferable to have a known service address that can be activated dynamically when it is needed and scale to the size that is needed. This functionality can be done on Amazon Web Services using a combination of the Elastic Load Balancer, and the Autoscaling group. The Autoscaling group requires one server to be up at all times, however, so to truly launch the service on demand only as needed, an orchestration script such as CloudFormation is necessary to launch the whole group of services. Squid servers are basically stateless and thus are good candidates for scaling up and down.

It is more challenging to scale a stateful service such as a job submission service up and down, since these servers have locally stored log files from the jobs that have to be submitted, that need to be accessible for some length of time. We demonstrated a scalable job submission service on AWS by means of lifecycle hooks and the new "standby" state. Finally, it is possible to create a failover alias that will attempt to contact the public cloud service first and fall back to a service on site if it is not there. By these means it is possible to define a service set that spans both public cloud and local site.

Many if not most of the services mentioned on Amazon Web Services have counterparts in open source cloud software such as OpenStack and OpenNebula as well as other public cloud providers. Any mention of Amazon Web Services in this paper should not be interpreted to exclude future collaboration with other commercial cloud providers.

Following work that has been done at Brookhaven National Labs, we will also configure the virtual machines on Amazon and the network routers on both ends to send network traffic over the ESNET link.

4. Conclusion

The Virtual Facility Project is using many of the long-standing tools of distributed computing such as virtual machines, HTCondor, GlideinWMS, and Amazon Web Services. What we bring is a new emphasis. We will ask the hard policy and architecture questions needed to integrate the commercial cloud into the Facility. We will make full use of cloud features and use the cloud like a cloud, not as an extension of the grid. We will explore strategic partnerships with the commercial cloud vendors. We will do the hardest data-intensive computing on the public cloud. We will unify grid and cloud provisioning activities into the Facility, saving both system overhead and personnel for several Virtual Organizations.

Acknowledgements

This work is supported by the US Department of Energy under contract number DE-AC02- 07CH11359 and by KISTI under a joint Cooperative Research and Development Agreement. CRADA-FRA 2014-0002/ KISTI-C14014

References

[1] J. Blomer et al, Status and future perspectives of CernVM-FS J. Phys.: Conf. Ser. 396052013, doi:10.1088/1742-6596/396/5/052013

[2] I. Gable et al, A batch system for HEP applications on a distributed IaaS cloud J. Phys.: Conf. Ser. 331062010, doi:10.1088/1742-6596/331/6/062010

[3] Gable, Ian, et al. "Dynamic web cache publishing for IaaS clouds using Shoal." *Journal of Physics: Conference Series*. Vol. 513. No. 3. IOP Publishing, 2014.

[4] Wu, Hao, et al. "Automatic cloud bursting under fermicloud." *Parallel and Distributed Systems (ICPADS), 2013 International Conference on*. IEEE, 2013.

[5] Timm, S., et al. "Grids, virtualization, and clouds at Fermilab." *Journal of Physics: Conference Series*. Vol. 513. No. 3. IOP Publishing, 2014.