Federating Clouds for High Energy Physics

Ian Gable
University of Victoria

Andre Charbonneau, Martin Conklin, Ronald Desmarais, Colson Driemel, Colin Leavett-Brown, Randall Sobie, Michael Paterson, Ryan Taylor

with significant assistance and support from the ATLAS and Belle II Collaborations, and CERN IT

OpenStack Summit, May 18-22, 2015
Outline

What is experimental High Energy Physics?
What our computing workloads look like?
Components of our Distributed Cloud
  Cloud Scheduler: Batch Job Management
  Glint: VM image distribution
  Shoal: Squid cache discovery
Some results
Large Hadron Collider

27 km ring
ATLAS Detector 2014
40 million collisions per second
Belle II Detector
KEK Laboratory
Scale and other experiments

Each interesting ‘event’ stored on disk

ATLAS experiment roughly 170 PB on disk today, now growing all the time

LHC Experiments and other High Energy Physics experiments sure to grow to exascale in coming years.

Now down to the details.
High Energy Physics Computing workloads

- High Throughput Computing workload composed of mostly embarrassingly parallel tasks (jobs).
- Jobs for HEP are usually 1-24 hours in length and can be done single core, or multi core jobs (memory saving)
- Jobs are either Monte Carlo simulation of collisions or analysis of real collision data from the detector readout
- Most of the workload today is run on ethernet connected Linux clusters from 500 - 10000 cores at Research and Education institutions around the world
- On any given day there is roughly ~300K cores running HEP jobs for the Worldwide LHC Computing Grid (collection of non-cloud federated Linux clusters)
Our IaaS timeline

Can we use Xen? 2005

We discover Nimbus Project 2006

Amazon EC2 2007

Multiple Nimbus Clouds 2008

OpenStack Arrives 2010

Multiple OpenStack Clouds 2012

CERN goes OpenStack 2013

Major Traction in HEP 2013
Today’s Problem and Opportunity

We wish to be able to run across multiple clouds without having any ‘special’ relationship with those cloud providers. In other words we can’t impose any requirements on them.
Components of the Solution

manage Jobs:

- Shoal

manage VM instances:

- Glint
- The VM itself: CernVM +CVMFS
Negotiator
Schedd
Collector
Master
Worker Node 1
Startd
Worker Node 2
Startd
Worker Node 2
Startd
universe = vanilla

### job parameters ===
dir = $(ENV(HOME))/logs/analy
output = $(Dir)/$(Cluster).$(Process).out
error = $(Dir)/$(Cluster).$(Process).err
log = $(Dir)/$(Cluster).$(Process).log
executable = runpilot3-wrapper.sh
arguments = -s ANALY_IAAS -h ANALY_IAAS -p 25443 -w https://pandaserver.cern.ch -u user
environment = "ATLAS_SITE_NAME=IAAS APF_PYTHON26=1 RUCIO_ACCOUNT=pilot"
request_cpus = 1
request_memory = 2000
request_disk = 1000000
requirements = VMType =?= "atlas-worker" && Target.Arch == "x86_64"
x509userproxy = $(ENV(HOME))/atlaspt.proxy

### job behaviour ===
stream_output = False
stream_error = False
notification = Error
should_transfer_files = YES
when_to_transfer_output = ON_EXIT_OR_EVICT

### VM configuration for cloud scheduler ===
+VMName = "PandaCern"
+VMAMI = "ucernvm-prod.1.18-13"
+VMInstanceType = "c8-30gb-430"
+VMKeepAlive = "30"
+VMJobPerCore = "True"
+TargetClouds = "IAAS"
+VMAMIConfig = "/srv/userdata/IAAS.yaml:cloud-config,/srv/userdata/cernvm-data.txt:ucernvm-config"
+VMUseCloudInit = "True"
+VMInjectCA = "False"

$ condor_submit atlas-sub.sub
Cloud Scheduler Define  Resources Available
Define resources available:

/etc/clou sscheduler/cloud_resources.conf

[chameleon]
auth_url: https://proxy.chameleon.tacc.utexas.edu:5000/v2.0
cloud_type: OpenStackNative
regions: regionOne
tenant_name: FG-54
vm_domain_name: .novalocal
key_name: rd_key
networks: FG-54-HEP-NET
security_group: default
username: *******
password: ********
secure_connection: true
enabled: false

[cc-east]
auth_url: https://east.cloud.computecanada.ca:5000/v2.0
cloud_type: OpenStackNative
regions:
tenant_name: Belle
networks: Belle_network
key_name: rd_key
vm_domain_name: .openstacklocal
security_group: default
username: *******
password: ********
secure_connection: true
enabled: false
Email on Friday:

“Hey Mike,

We are taking cloud-x down Tuesday at 9:00 central time can you make sure you aren’t running anything important.

Cheers,

The friendly OpenStack Admins”

State of the System on Monday Morning:

- 1000 Cores of Belle-11 jobs running
- Each job is roughly 12 hours long and each job is in a different state of completion
- There are several thousands jobs waiting in the Condor job queue

Goal:

No users jobs are killed and all VMs are shutdown cleanly before 9:00 Tuesday

Operations on Monday Morning:

Prevent any more VMs from being booted:

```
$ cloud_admin -d cloud-x
```

Stop submitting new jobs to running VM and shutdown the VM once all jobs are complete:

```
$ cloud_admin -o -c cloud-x -a https://github.com/hep-gc/cloud-scheduler
```

https://github.com/hep-gc/cloud-scheduler
Problem: Too many clouds to manage VM images manually

Solution: Glint Image Distribution Service
OpenStack with Glint
Horizon Interface with Glint Pages
Goals for Glint

We have learned a lot this week.

Take advantage of keystone federation. User won’t have to provide creds for multiple clouds

Take Advantage of Glance Tasks

Ultimate goal to have the functionality in Glint available as a part of Keystone and Glance

PyPI:
https://pypi.python.org/pypi/glint-service/

launchpad:
https://launchpad.net/python-glint

Github:
https://github.com/hep-gc/glint-service

more details contact Ron Demarais
<rd@uvic.ca>
The Virtual Machine Image
CernVM and CVMFS

CernVM is RHEL compatible HEP software appliance in only 20 MB

CVMFS is a caching network file system based on HTTP and optimized for software, i.e. millions of small files

CernVM-FS infrastructure used to distribute experiment software can be reused. In comparison, the (already small) CernVM-FS client downloads an additional repository and the repository version is selected.

The amount of data that needs to be loaded in order to boot the virtual machine is very little. The init ramdisk contains the CernVM-FS client and a steering script. The purpose of the early user space is to load the necessary storage device drivers to access the actual root file system. Once the actual root file system in memory provided by the init ramdisk.

First, the scratch hard disk is mounted on /root.rw.

CernVM finds an already prepared scratch space on next boot. The scratch space is used as a persistent writable overlay for local changes to the root file system and as a cache for the CernVM root file system stack.

The initramfs image contains a Linux image and a steering script that can process contextualization information (sometimes called “user data”) from various sources.

Enables an attacker to inject corrupted files somewhere in the file system hierarchy only for lookups; the actual data transfer can take place after the boot process is finished.

Content-addressable storage is compute-intensive. Still, it is worthwhile to use CAS for the “write once read many” pattern of software and experiment data.

CernVM-FS has many advantages, in particular for software repositories.

File de-duplication has been observed to be very useful with CVMFS. From release to release only a fraction of the code has changed, so the amount of data that needs to be fully loaded and decompressed is only a small fraction of the total size.

CVMFS comes with its own CDN! Requires fast and near HTTP cache.

http://cernvm.cern.ch

requires fast and near HTTP cache.

CernVM-FS:

- AUFS R/W Overlay
- initrd: CernVM-FS + μContextualization
- Kernel
- AUFS
- Fuse

User Data (EC2, OpenStack, …)
The caching challenge on IaaS cloud

When booting VMs on different arbitrary clouds they don’t know which squid they should use.

In order to work well, VMs need to able to access a local web cache (squid) to be able to efficiently download all the experiment software and now OS libraries they need to run.

If a VM is statically configured to access a particular cache it can be slow (Geneva-Vancouver for example) and it can get overloaded.
Shoal uses the highly Scalable AMQP protocol to advertise Squid servers to Shoal.

It uses GeoIP information to determine which is the closest to each VM.

Squids advertise every 30 seconds, server verifies if the squid is functional.

https://github.com/hep-gc/shoal
Some Results

- CERN Instance Europe
- UVic Instance (Americas+Australia)

Belle II Experiment

Cloud 500K jobs

Total Number of Jobs by Site

- LCG.CERN.ch
- CERN-IT
- CERN-ITAT

Cumulative ATLAS Jobs
- Cloud
- 500K jobs
Summary

CloudScheduler/HTCondor flexible way to run Batch Jobs on Clouds.

Key enabling technologies for this:
- CVMFS + CernVM
- Shoal: dynamic Squid cache Publishing
- Glint: VM Image Distribution

Current users ATLAS, Belle II, CANFAR, Compute Canada HPC consortium
Acknowledgements