# CMS use of HPC resources

Dirk Hufnagel (FNAL) for CMS Offline&Computing

OSG AHM 2017 San Diego

6th March 2017

# Overview

- Motivation
- HPC resources (US only)
- Strategy
- Progress

# Motivation

HPC resources traditionally not used much by HEP.

To prepare for the future we want to be ready to use a multitude of HPC resources.

Most HPC resources are made available based on allocations, not on the usual HEP model which is pledged. This potentially allows more elastic use of resources (peak use vs. steady state use for pledged resources).

# HPC Resources (US only)

| Name | Institution | Architecture | Start Date |
|---|---|---|---|
| **Stampede** | **TACC** | **100k core Intel Sandy Bridge** | **2013** |
| **Stampede 2** | **TACC** | **Xeon+Phi** | **approved** |
| **Comet** | **SDSC** | **47k core Intel Haswell** | **2015** |
| **Edison** | **NERSC** | **133k core Intel Ivy Bridge** | **2013** |
| **Cori Phase 1** | **NERSC** | **52k core Intel Haswell** | **2015** |
| **Cori Phase 2** | **NERSC** | **632k core Intel Knights Landing (4x HT)** | **2016** |
| **Mira** | **ANL** | **786k core IBM PowerPC** | **2012** |
| **Theta** | **ANL** | **~150k core Intel Knights Landing (4x HT)** | **2017** |
| **Aurora** | **ANL** | **~3M core Intel Phi 3rd generation** | **2019** |
| **Titan** | **Oak Ridge** | **299k core AMD Opteron + GPUs** | **2012** |
| **Summit** | **Oak Ridge** | **~3400 nodes IBM Power 9 + GPUs** | **2017** |

**NSF funded , DOE funded** (from public information like facility websites etc)



Fermilab

# Strategy

CMS already spends more of its cpu budget on event reconstruction than on event generation and simulation and the ratio will become even more skewed with increased pileup. Accumulated core hours from ~Feb to ~Oct 2016:

- ~155M GenSim
- ~175M Data Reco + MC DigiReco
- ~190M User Analysis

Want to be able to run our full range of workflows, including Data Reco and MC DigiReco, on HPC resources.

- **large data input/output**
  - more stress on storage and network
    and in general more difficult to run efficiently

# Strategy

First order goal was to make HPC resources look just like any other CMS/OSG site. That means integration into the glideInWMS system and a CMS supported/validated runtime environment with the CMS software available (via cvmfs preferably).

Limited amount of effort in CMS CompOps, the more these new resources can work like grid resources, the more different resources we will be able to easily support.

Don't have much local storage, so plan to not use any of it except for jobs temporary scratch space.

# Two categories of HPC sites: NSF or DOE funded

NSF funded HPC clusters (Comet, Stampede)

- Allocations through XSEDE
- Working with OSG on this (FactoryOps and others)
- Access via CE
- CMS compatible runtime environment (CentOS6 with cvmfs mounted)

DOE funded HPC clusters (Edison, Cori)

- Commissioning in a Fermilab HEPCloud context
- Access via BOSCO
- Have to provide our own runtime environment

# Progress – SDSC Comet and TACC Stampede - Status

Factory connects to cluster via CE.

Commissioning “complete”,  we can run workflows. They read input data via the AAA data federation and stage out to the FNAL CMS T1. We are using a local proxy server (at UCSD) for Comet and remote proxy servers at FNAL for Stampede.

Concerns are about scalability of AAA data reads and stageout to FNAL. Additional concerns about impact on job efficiency from using remote proxy servers at Stampede.

# Progress – SDCS Comet and TACC Stampede - Problems

Due to the low limit for total number of jobs (which are normal for HPC), the only way to get large amounts of resources is to request jobs that use multiple nodes.

Factory entry can specify the pilot job to be run on multiple nodes, this was used for LIGO production on Stampede via OSG. At the moment this has problems though:

- Factory entry needs to specify all the cpus (on all nodes)
- Need override in pilot to only use it's own node cpus
- Pilot logs of all these pilots go into single file

Supporting multi-node pilots properly is a feature request for GlideInWMS.

# Progress – SDCS Comet and TACC Stampede - Outlook

For both Comet and Stampede we only had startup allocations (50k SU, roughly equivalent to 100k cpuhours). Now that we can use these resources, what do we want to use them for ?

Scaling issues are a concern but we have learned a lot about massive remote cpu-only resources from the Amazon and Google Cloud tests and know the problems (network, remote data reads, remote job stageout, squid proxy setups) and how to solve/avoid them.

We are better positioned here already at Comet with the close UCSD CMS site (and especially the local proxy server setup). Ramping up CMS use of Stampede would require a similarly close proxy server setup in order to not pay significant job efficiency penalties.

# Progress – NERSC Edison/Cori

Bulk of our efforts have been spend here. NERSC is the primary commissioning use case for the Fermilab HEPCloud HPC integration.

NERSC presents a class of HPC resource that is challenging, but not “impossible” (Cray SuperComputer that is very different from standard batch cluster, but: x86-64, nodes have outbound network, in principle all we need).

Cori Phase 2 one of the first HPC resources with Intel Knights Landing.

Two challenges:

- Submission (how to get GlideInWMS pilots onto worker nodes)
- Runtime (how to make jobs run)

# Progress – NERSC Edison/Cori - Submission

There are grid CE interfaces for Edison/Cori, but we don’t use them at this time.

When we started first tests at NERSC they were not available, so we looked at BOSCO. Since this setup is working and is in principle more flexible (exposes all the options that are available via direct batch submission), we decided to stay with it for now.

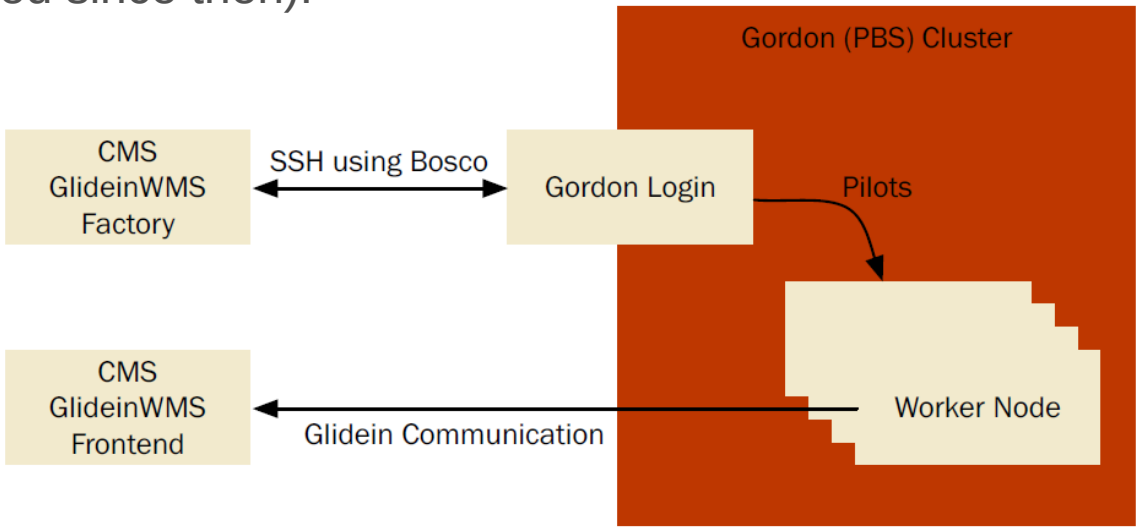
Testing the grid CE provided for Edison/Cori is on the to-do list as well.

# Progress – NERSC Edison/Cori - Submission

Bosco style (remote ssh) pilot submission with local wrapper code to interface with site batch system.

Complication: NERSC migrated Cori/Edison to SLURM in January 2016, Bosco did not support native SLURM (~fixed since then).

**Example diagram for the SDSC Gordon cluster where we first used this**






Fermilab

# Progress – NERSC Edison/Cori - Runtime

Worker nodes run Compute Node Linux, a stripped down Linux version for Cray worker nodes. Even if we could make it work somehow, we don’t really want to run our software directly on that (concerns about comparable output without validation).

- support shifter (NERSC developed container system)

Cannot mount cvmfs on worker nodes (software in container).

NERSC worker nodes have outgoing network.

NERSC provides local squid proxies (for conditions data).

# Progress – NERSC Edison/Cori - Status

From the outside NERSC looks like a normal CMS site. On the inside:

- glideInWMS factory submits pilot via remote ssh
- BOSCO wrapper code submits pilot to SLURM
- BOSCO wrapper code configures shifter container
- pilots starts up in an SL6 shifter container
- pilot runs jobs
- jobs read data remotely via AAA xrootd federation
- jobs stages out remotely to FNAL

NERSC has been used for a ReReco campaign, a MC DigiReco campaign and a MC GenSimDigiReco campaign (same workflow as was run at Google Cloud).

# Progress – NERSC Edison/Cori - Status

In 2016 we had two NERSC allocations:

- 1.5M cpuhours integration/commissioning
- 5M (later cut to 2M) cpuhours production

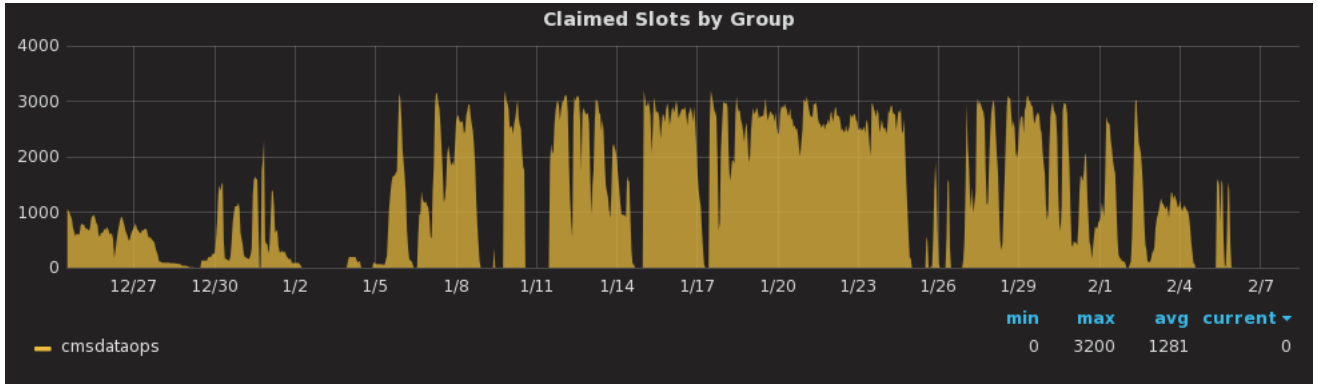
It took us a long time to ramp up usability of NERSC resources, but we used up about 1M cpuhours of the first allocation and 2M of the second. Most of this use came in the last month of 2016.

In 2017 we have one NERSC allocation:

- 12M cpuhours production

During the first 30 days of the allocation we already used 1.5M cpuhours.

# Progress – NERSC Edison/Cori - Status




3200 slots represent:

50 pilot jobs x 1 node per pilot job x 64 logical cores per node

50 jobs is the limit for jobs requesting one full node on Cori.



Fermilab

# Progress – NERSC Edison/Cori - Shifter

We use shifter images that contain SL6 plus the CMS cvmfs repo. NERSC doesn't build them for us (anymore). What we are doing instead:

- Call uncvmfs on FNAL machine to build squashfs image
  - Clones CMS cvmfs repo
  - Adds CMS cvms repo to image under prefix
  - Adds SL6 OS files to image
- Copy resulting squashfs image to Cori scratch space
- Notify NERSC so they create the updated shifter image

All of this is so far manually triggered, automating it and optimizing it up are the next steps. Fully automating this means we don't need to worry about availability of CMS software releases, they will always be there with a day delay at most.

# Progress – NERSC Edison/Cori - Network

CMS has spend a lot of effort optimizing our framework and data formats to support efficient remote input data reads. Even so, if the available bandwidth is too low, we will starve the cpus and they will site idle waiting for input data to be read from a remote source.

Certain steps in CMS workflows are at the moment only ~50% cpu efficient when run at NERSC due to this effect. We can mitigate this by careful workflow selection, but this stands against the goal of being able to use HPC resources for the full range of workflows.

We are working closely together with NERSC support to address these issues.

# Progress – NERSC Edison/Cori - Outlook

We will be looking into scaling up peak use of NERSC. For this we will start using Edison and Cori and both full node and shared queues in parallel. We attempted this once before in December and failed. We now have better options in how GlideInWMS/BOSCO can be configured to try this again (different factory entries can point at different Bosco installations). This should bring us to roughly 10k cores peak use.

The next step after that is submitting multi-node pilots, limits in this mode are only in what NERSC is willing to give us.

What the sustainable peaks over longer time windows actually are remains to be seen. Can we use HPC to address truly urgent demands or will it be more about replacing base (slow and steady) capacity ?

# General Outlook

Looking forward, USCMS wants to access HPC resources in the US via Fermilab HEPCloud.

This makes sense since the way we run workflows at the HPC sites, the workflows will read their input data from the Fermilab CMS T1 (and also some CMS T2 sites) and stage their job output back to the Fermilab CMS T1.