Journal of Physics: Conference Series

PAPER • OPEN ACCESS

CMS use of allocation based HPC resources

To cite this article: Dirk Hufnagel and CMS Collaboration 2017 *J. Phys.: Conf. Ser.* **898** 092050

View the article online for updates and enhancements.

Related content

- Bringing ATLAS production to HPC resources - A use case with the Hydra supercomputer of the Max Planck Society
  J A Kennedy, S Kluth, L Mazzaferro et al.
- SCEAPI: A unified Restful Web API for High-Performance Computing
  Cao Rongqiang, Xiao Haili, Lu Shasha et al.
- Recent Standard Model results from CMS
  Simon de Visscher and CMS collaboration

This content was downloaded from IP address 188.184.3.52 on 15/12/2017 at 16:18

CMS use of allocation based HPC resources

Dirk Hufnagel (on behalf of the CMS Collaboration)
Fermilab, P.O. Box 500, Batavia, IL 60510-5011, USA

E-mail: Dirk.Hufnagel@cern.ch

Abstract. The higher energy and luminosity from the LHC in Run2 has put increased pressure on CMS computing resources. Extrapolating to even higher luminosities (and thus higher event complexities and trigger rates) in Run3 and beyond, it becomes clear the current model of CMS computing alone will not scale accordingly. High Performance Computing (HPC) facilities, widely used in scientific computing outside of HEP, present a (at least so far) largely untapped computing resource for CMS. Even just being able use a small fraction of HPC facilities computing resources could significantly increase the overall computing available to CMS. Here we describe the CMS strategy for integrating HPC resources into CMS computing, the unique challenges these facilities present, and how we plan on overcoming these challenges. We also present the current status of ongoing CMS efforts at HPC sites such as NERSC (Cori cluster), SDSC (Comet cluster) and TACC (Stampede cluster).

1. Introduction

The CMS (Compact Muon Solenoid) experiment is a general purpose particle physics detector at the LHC (Large Hadron Collider) at CERN [1]. During the ongoing Run2 data taking the LHC delivered record luminosity in 2016 and the outlook for the remainder of Run2 is more of the same. Looking further ahead to the HL-LHC (High Luminosity Large Hadron Collider) [2], the data rates and complexity are expected to increase even more dramatically. All of this results in a lot of pressure on the current CMS computing infrastructure and an outlook for the HL-LHC where our existing computing infrastructure will simply not be sufficient to meet the demands of the experiment.

With this in mind, CMS is investigating all types of additional computing resources. HPC is an area which historically hasn't been used much for HEP (High Energy Physics) computing. As such, HPC resources are an attractive target since the potential gains for our computing capacity are sizable.

2. Current and future HPC resources (United States)

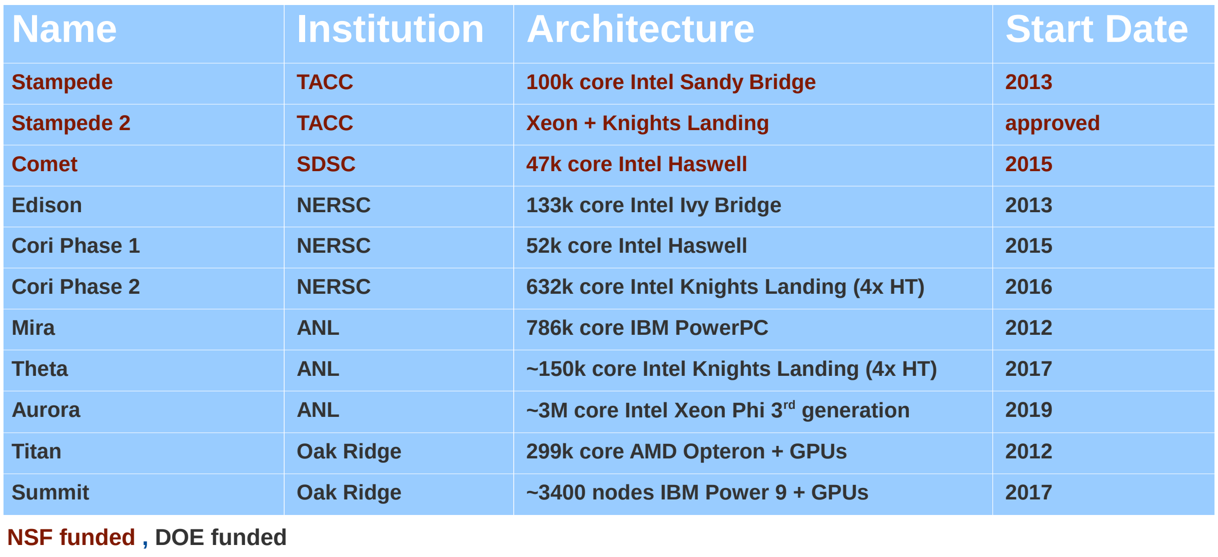
Figure 1 shows a list of current and planned HPC clusters in the US, separated by which agency (NSF or DOE) funds the facility. The list is compiled from publicly available sources, mainly from the HPC facilities web sites, some of which are referenced later in their own subsections. The number of cores are approximations and might only have been correct at some point in the past (Fall 2016 when this material was prepared for the CHEP conference), since at least some of these HPC clusters evolve over time. One thing that immediately jumps out is the dominance of Intel x86-64 in general, and looking ahead a few years, of Intel Xeon Phi in particular. This leads to the conclusion that targeting these architectures and the facilities that have or will have

Content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.
Published under licence by IOP Publishing Ltd

Job type	Core hours
Monte Carlo Generation and Simulation	155M
Data Reconstruction and Monte Carlo Digitization and Reconstruction	175M
User Analysis	190M

Table 1. Accumulated core hours by job type from Feb 2016 to Oct 2016 [9]

clusters based on them promises the most efficient use of peoplepower for CMS due to likely synergy effects. The expectation is that making CMS workflows function well on Cori Phase 2 will reduce the effort needed to then port the same CMS workflows to Theta and Aurora for instance. To give a rough comparison between these HPC clusters and the total amount of grid resources available to CMS, the total deployed and pledged cpu capacity for CMS is equivalent to about order 150,000 cores throughput the year (as of 2016, this is also evolving over time).



Name	Institution	Architecture	Start Date
Stampede	TACC	100k core Intel Sandy Bridge	2013
Stampede 2	TACC	Xeon + Knights Landing	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 Xeon 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

Figure 1. HPC Resources in the United Status

3. General strategy for using HPC

CMS already spends more of its cpu budget on event reconstruction (see Table 1) than on event generation/simulation. That ratio is expected to become even more skewed in favor of reconstruction as pileup in the LHC increases. As such, CMS wants to be able to run its full range of workflows on HPC resources. Commissioning HPC resources just for Monte Carlo Generation and Simulation would be easier to achieve, but would impose restrictions that would severely reduce the usefulness of these resources.

That being said, event reconstruction is much more difficult to make to work efficiently since the workflows are more complex due to large data input and output and consequently larger stress on storage and network.

A side effect of this goal is also that it makes non-x86 resources even less interesting than they already are due to pure scale of available resources. Event generation is relatively easy to port to different architectures since we use external code that doesn't have too many dependencies on experiment specific code. Porting the full reconstruction code of CMS would be very different in terms of needed peoplepower.

Having covered which resources we want to target and which workflows we want to run, lets address our strategy how to integrate these resources into the CMS computing infrastructure. The goal is still the same, it should be easy to run all types of CMS workflows on these HPC resources. As such, an HPC resource should, as much as possible, look just like a regular CMS resource and be usable like a regular CMS resource. Otherwise restrictions will come into play that will reduce the usefulness of HPC resources.

GlideInWMS [3] and HTCondor [4] [5] form the backbone of the CMS computing environment. Any type of extra resource like an HPC cluster we want to use would have to be fully integrated into this environment. As such, we are dealing with two problems here. First, how can the GlideInWMS Factory submit pilot jobs into the HPC resource and secondly, how do we make sure that these pilot jobs find a runtime environment in which they (and the CMS payloads they execute) can function.

Another problem is that HPC resources usually don't provide a lot of storage space to their users. Even if we solve all the above mentioned problems, we still would not have a resource that looks like a normal grid T2 site. The ratio of cpu to local storage would be very different from our normal T2 sites. Since the effort needed to integrate a new storage resource into our data management system is only justified above a certain size threshold, we decided this wouldn't be worth it for HPC resources (at this time). We will use them as cpu-only resources, only using local storage for temporary job scratch space. All job data input will be read remotely from another CMS site and the job output will be staged directly back to a CMS site from the job itself. This then of course puts more stress on the external network connectivity of the HPC resource.

We realized that there were a lot of problems and technical challenges with this strategy, but we wanted to explore if and to what degree we could succeed with it before looking at more complex (and consequentially harder to implement) solutions.

4. What are the technical requirements to run a CMS job ?

There are certain requirements that need to be fulfilled for a resource to be able to run CMS jobs.

- Hardware and software environment can run CMS software
- GlideInWMS factory can get pilots onto resource
- Pilot can call back to get work (outgoing network)
- Access to resource specific settings
- Access to CMS software and conditions

Since we put our priority on x86-64 compatible HPC resources, hardware compatibility is not a big issue. Software environment compatibility can be an issue though since we standardized on Redhat Enterprise compatible Linux versions in HEP and some HPC resource have a very different setup (for instance Cray with their very slimmed down and restricted Compute Linux).

On the next point there are big differences between different HPC clusters. Standard grid sites will usually have a grid CE (Computing Element) providing the interface through which pilot jobs can be submitted into the local batch system. Some HPC clusters provide such a grid CE, some don't or didn't until very recently. Some HPC clusters provide outgoing network from their worker nodes, some don't. Even if we don't have a grid CE for an HPC cluster, there is an alternative access method using BOSCO. BOSCO allows to HTCondor submit jobs directly into a remote batch cluster by using an ssh tunnel to a batch cluster head node and some wrapper code on that head node. More details on how BOSCO is integrated into the GlideInWMS system can be found here [8].

Access to both resource specific settings and to the CMS software is normally provided via the Cern Virtual Machine File System CVMFS [6], but if that is not possible to be mounted on the worker nodes both can also be provided from the local file system.

Access to conditions is via a squid proxy infrastructure [7]. For efficiency reasons local squid proxies are preferred, but we can also use the squid proxies at remote CMS sites if necessary (with some efficiency loss).

5. HPC cluster we are currently trying to commission for CMS use

In this section I want to go through a list of HPC clusters we are currently trying to commission and what the status of these efforts is.

5.1. SDSC Comet

Comet is an Intel Haswell cluster at SDSC (San Diego Supercomputer Center) [10]. We have infrastructure very close (within the same LAN for performance purposes) since SDSC is located on the UCSD (University of California San Diego) campus and UCSD hosts a CMS T2 site. We also have a very good relationship with the SDSC sysadmins.

All of this means that Comet should be easy for CMS to use. The Comet predecessor Gordon was in fact the very first HPC cluster we have used for a large CMS production campaign, using up a few million cpu hours [11].

The runtime environment at Comet is compatible with CMS and we also have CVMFS provided on the Comet worker nodes. At the moment we are working on integration of the new Virtual Cluster interface that Comet provided into the GlideInWMS infrastructure.

5.2. TACC Stampede

Stampede is a Sandy Bridge cluster at TACC (Texas Advanced Computing Center) [12]. The runtime environment is compatible with CMS and access to CVMFS is provided. Stampede is also accessible via a grid CE and we are currently working on integrating it into the CMS workflow management systems. Jobs are run, but we are still debugging problems with remote stageout from the job to the Fermilab T1 site.

An upgrade to Stampede 2 has been approved and will happen in place by replacing the existing Stampede cluster in stages. Stampede 2 will be a mixed Intel Xeon and Intel Knights Landing system.

5.3. NERSC Edison and Cori

Edison and Cori are Ivy Bridge and Haswell clusters at NERSC (National Energy Research Scientific Computing Center) [13]. Currently NERSC is commissioning a Cori Phase 2 cluster based on Intel Knights Landing, which will be usable in parallel to the existing Edison and Cori Phase 1 clusters.

The bulk of our efforts have been spent here, mainly due to the fact that these clusters represent a challenging, but not impossible HPC setup in terms of being usable for CMS. In addition Cori Phase 2 is the first HPC clusters with Intel Knights Landing that is actually available.

All NERSC clusters have outgoing network from the worker nodes and have a cpu architecture which is compatible with CMS software. This makes NERSC in principle usable for CMS.

There are technical problems though. NERSC worker nodes run Compute Node Linux, a stripped down Linux version for Cray system worker nodes. There is no practical way we could get our CMS software to run on this natively. NERSC has a solution for this though, they support a Docker-like Linux container system called shifter [14, 15]. An additional problem is that due to security restrictions we cannot mount CVMFS on the NERSC worker nodes. We work around this restriction by including the relevant parts of the CMS CVMFS repository inside the shifter container. This leads to very big containers (order 100GB to a few 100GB).

When we started testing, we were concerned about the very large shifter containers and how they would impact job startup times and general job performance. For the job startup times we didn't see a noticable effect, mainly because the way we specify the shifter container for a job is via batch system directives. Using these directives the batch system sees what container will be needed by the job before the job runs and can pre-initialize the container. This seems to be working well. We have not done any dedicated measurements on it, but a simple interactive batch job test showed no noticable delay between the batch job starting up and the interactive shell inside the container becoming available. As for general job performance, we also have not done any dedicated measurements. Anectodal evidence from performance tests of reconstruction jobs reading only local input data shows roughly comparable results than similar jobs at other sites. All of which basically says that at the moment we have no evidence that these large shifter containers cause sigificant problems, so this has not been an area of intense study.

For conditions access NERSC provides squid proxies that we can use.

NERSC provides grid CE interfaces for their clusters, but we don't use them. Instead we use BOSCO to submit pilot jobs from the GlideInWMS factories into NERSC. This has both historical and technical reasons. Historical since when we started testing at NERSC there were no grid CE available. Technical since the BOSCO based solution is more flexible and allows more control over the process than the grid CE solution. Testing the grid CE solution is still on our to-do list, but currently not a priority since the BOSCO solution works for us.

All of this taken together leads to NERSC clusters looking from the outside just like a normal CMS site. What happens on the inside

- GlideInWMS factory submits pilot via BOSCO (remote ssh channel).
- BOSCO wrapper code submits pilot into the NERSC batch system (SLURM [16]).
- BOSCO wrapper code configures shifter container.
- Pilot starts up in an SL6 shifter container with the CMS CVMFS repository inside the container.
- Pilot pulls CMS jobs and runs them.
- CMS jobs read data remotely via the CMS xrootd data federation.
- CMS jobs stageout output data remotely to the Fermilab T1 site.

We have tested Monte Carlo Generation and Simulation at NERSC, but also Data Reconstruction and Monte Carlo Digitization and Reconstruction. NERSC was used for the Fall ReReco and DigiReco campaigns. Figure 2 shows the number of cores assigned to CMS jobs over a few days in early October 2016 during one of these campaigns.

In 2016 we are using two allocations at NERSC, one 1.5M cpuhours commissioning allocation and another 5M cpuhours production allocation. We are still in the process of scaling up production use of NERSC and there are still plenty of issues and problems to overcome before NERSC can become a regular CMS production resource.

- There is no automated build procedure for a shifter container with CMS CVMFS repository. NERSC builds them at the moment for us manually (by request).
- Bandwidth limits for reading input data from CMS xrootd data federation can starve cpus on the NERSC worker nodes.
- NERSC clusters are quite busy, working on best strategy how to interface with batch system to get our pilots to run.
- TCP connections are limited per cluster, which causes problems for GlideInWMS.

We are working together with NERSC Technical Support to address these remaining issues.

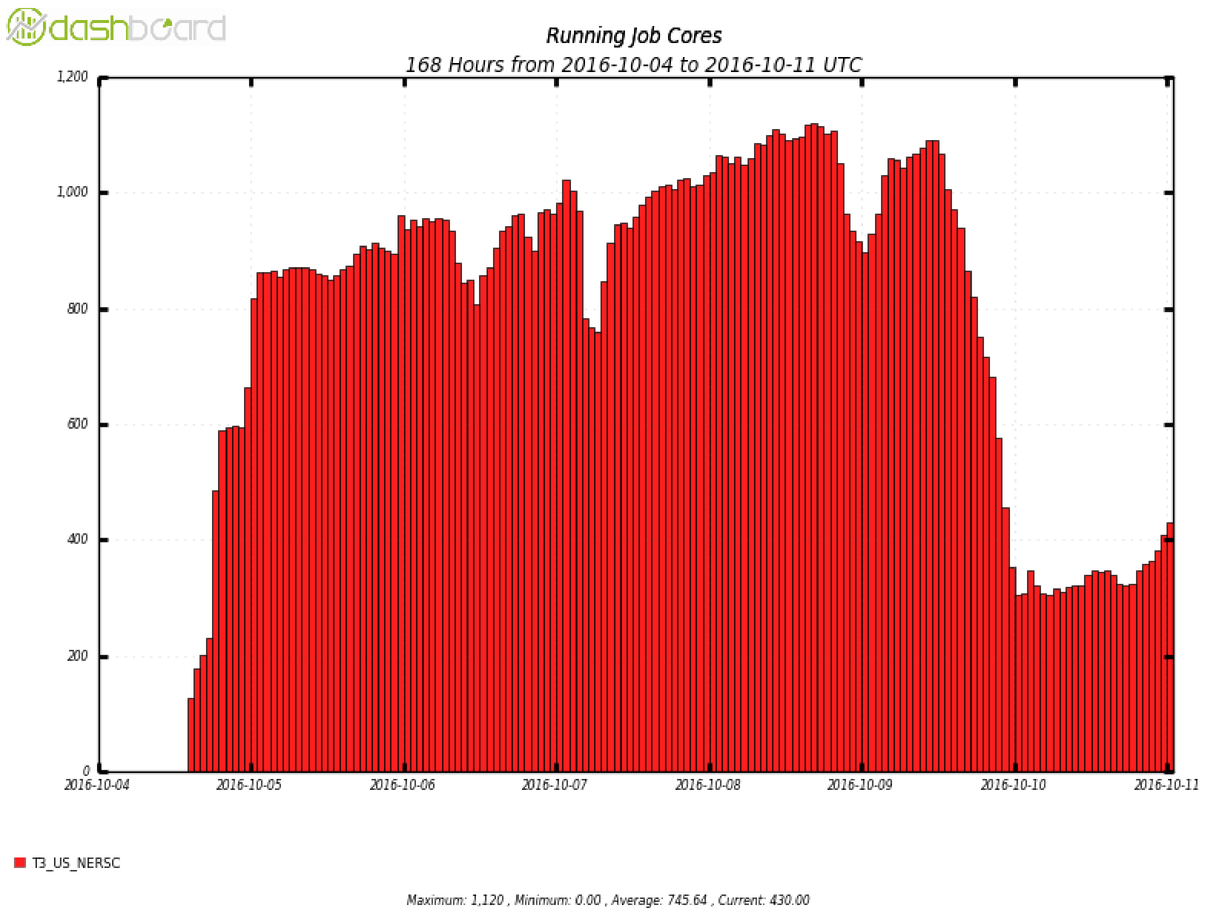


Figure 2. Cores in use at NERSC for CMS jobs

6. Summary and Outlook

CMS is actively looking at integrating HPC clusters into its computing infrastructure in order to alleviate some of the resource constraints imposed by the challenging LHC Run2 computing environment and in order to explore scaling up HPC use for the even more challenging future HL-LHC computing demands.

Both Comet at SDSC and Stampede at TACC are close to being usable, we are working on full integration into the CMS computing infrastructure.

We have commissioned NERSC as “just another CMS site” and used it in production campaigns. We still need to work on efficiency and reliability improvements before we can fully treat it as just another CMS site though.

References

[1] S. Chatrchyan *et al.* [CMS Collaboration], “The CMS experiment at the CERN LHC”, JINST **3**, S08004 (2008).
[2] The HL-LHC Project http://hilumilhc.web.cern.ch/
[3] I. Sfiligoi, D. C. Bradley, B. Holzman, P. Mhashilkar, S. Padhi and F. Wurthwrin, “The pilot way to Grid resources using glideinWMS”, WRI World Congress **2**, 428 (2009).
[4] HTCondor homepage: http://research.cs.wisc.edu/htcondor/
[5] M. Litzkow, M. Livny, and M. Mutka, “Condor A Hunter of Idle Workstations”, Proc. of the 8th Int. Conf. of Dist. Comp. Sys., June, 1988, pp 104-111.
[6] P. Buncic, C. Aguado Sanchez, J. Blomer, L. Franco, A. Harutyunian, P. Mato and Y. Yao, “CernVM: A virtual software appliance for LHC applications” J. Phys. Conf. Ser. **219**, 042003 (2010).

[7] B. Blumenfeld *et al.* [CMS Collaboration], "Operational Experience with the Frontier System in CMS", J. Phys. Conf. Ser. **396**, 052014 (2012).
[8] D. Weitzel, I. Sfiligoi, B. Bockelman, J. Frey, F. Wuerthwein, D. Fraser and D. Swanson, "Accessing opportunistic resources with Bosco", J. Phys. Conf. Ser. **513**, 032105 (2014).
[9] Kibana Elasticsearch CMS Job Monitoring https://es-cms.cern.ch/
[10] SDSC HPC Systems http://www.sdsc.edu/services/hpc/hpc_systems.html
[11] SDSCs Gordon Supercomputer Assists in Crunching Large Hadron Collider Data http://ucsdnews.ucsd.edu/pressrelease/sdscs_gordon_supercomputer_assists_in_crunching_large_hadron_collider_data
[12] Stampede User Guide https://portal.tacc.utexas.edu/user-guides/stampede
[13] NERSC Computational Systems http://www.nersc.gov/users/computational-systems/
[14] Shifter documentation http://www.nersc.gov/research-and-development/user-defined-images/
[15] Shifter github code repository https://github.com/NERSC/shifter
[16] SLURM at NERSC Overview http://www.nersc.gov/users/computational-systems/cori/running-jobs/slurm-at-nersc-overview/

Acknowledgments
Fermilab is operated by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the United States Department of Energy.

This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.