Enabling opportunistic resources for CMS Computing Operations

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. With the increased pressure on computing brought by the higher energy and luminosity from the LHC in Run 2, CMS Computing Operations expects to require the ability to utilize opportunistic resources, resources not owned by, or a priori configured for CMS to meet peak demands. In addition to our dedicated resources we look to add computing resources from non CMS grids, cloud resources, and national supercomputing centers. CMS uses the HTCondor/glideinWMS job submission infrastructure for all its batch processing, so such resources will need to be transparently integrated into its glideinWMS pool. Bosco and parrot wrappers are used to enable access and bring the CMS environment into these non CMS resources. Here we describe our strategy to supplement our native capabilities with opportunistic resources and our experience so far using them.

1. Introduction
The CMS (Compact Muon Solenoid) experiment is a general purpose particle physics detector at the LHC (Large Hadron Collider) at CERN [1]. The LHC is scheduled to restart for physics in 2015 after two years of maintenance and upgrading. The new Run 2 data taking will bring much higher energy and luminosity than the previous data taking. With that come higher demands on the computing infrastructure, demands that are not fully offset by increases in deployed computing resources. CMS has spent the last two years improving code and physics algorithm performance, but even so, we expect the Run 2 data processing to be a much more resource constrained environment than what we were used to in Run 1.

In this context, CMS has asked itself the question if we can make use of other resources beyond our normal pledged resources from CMS contributors. The next chapters will describe different categories of such resources (collectively called “opportunistic resources”) and our strategy how to make use of them.

2. General strategy for Opportunistic Resources
GlideinWMS [2] and HTCondor [3] [4] form the backbone of the CMS job execution system. Between LHC Run1 and Run2 CMS migrated from using various separate HTCondor pools to using a single common Global pool for job execution. The Tier0 resources currently still use their own separate pool, but this is irrelevant for this discussion here. For more details please refer to [7] and [8].

To first order, any opportunistic resource we want to use should be integrated into the Global Pool via GlideInWMS in a long-term sustainable fashion. The last point is important, we are
looking at integration efforts that easily can be applied to a whole class of resources without a linear scaling in effort with each new resource we want to add.

One caveat here, in this paper I am ignoring certain resources where we attempt a direct integration into the Global Pool without going through GlideInWMS, as that method currently only applies to some special resources and its unclear yet if we want to support this in a general fashion.

3. What are the technical requirements to run a CMS job?
When looking at opportunistic resources, which usually means non-CMS resources, an important question to answer is what the actual technical requirements are to run a CMS job. Once the requirements are known, one can look at any given resource or type of resource and check which requirements are full-filled and which are not. For the latter ones one then has to find solutions or workarounds. In general, the following items are requirements to be able to run CMS jobs on a resource via the usual job submission systems used within CMS:

- 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

We need to be able to run the CMS binaries on the computing resource. For the most part that currently restricts us to AMD64 compatible hardware and a RHEL6/SL6 (Redhat Enterprise Linux 6/Scientific Linux 6) compatible software environment. Fortunately this is not exactly an uncommon hardware/software combination.

The GlideInWMS factory has be be able to contact the computing resource and get it to execute a glideinWMS pilot and that pilot then needs to be able to call back to the central glideinWMS/HTCondor services and pull work (which requires outgoing network from where the pilot runs).

As far as site specific settings are concerned, these can be provided locally or from the Cern Virtual Machine File System CVMFS [5] (via a local contextualization to know where to look on CVMFS). They encompass data access rules, data stage-out rules and settings for proxy servers. On opportunistic resources we usually assume no local storage space, therefore we read data via the CMS xrootd federation and stage-out data remotely to a CMS site that is close enough to provide efficient and reliable stage-out and large enough to be able to handle the additional load on their storage system. Any subsequent processing or handling of the data would then start at that site. Baseline for proxy server settings follows the same guidelines of using proxies at a close and large CMS site, although as a proxy server is somewhat generic, it might be possible to use (or deploy) one also at a non-CMS computing resource.

The access to CMS software is provided via CVMFS, using the previously mentioned proxy servers. Access to conditions is similarly handled via the proxy servers.

4. Types of opportunistic resources
Given a resource we want to use, some of the requirements listed in the previous section will be either met or not depending on the properties of the resource itself. The hardware/software environment and the question of outgoing network from the worker nodes are nothing we can do anything about to first order for instance. What we can do something about is how the resources are integrated into the CMS system. Looking only at these and assuming that all other requirements are met already, we are left with:
• How can we get a glideinWMS pilot onto the resource?
• Can the CVMFS repositories needed by CMS be made available?
• How do we get site specific settings to the job?

If these questions can be answered in a positive way, a resource becomes in principle usable by CMS. Let's look at three generic classes of opportunistic resources CMS is planning to use and how these questions can be answered for them.

4.1. Non-CMS Grid resources
It is trivial to submit a CMS pilot job to a non-CMS grid resource, assuming the resource accepts opportunistic jobs from the CMS VO at all. CVMFS might or might not be available and even if it is, whether or not the CMS specific CVMFS repository is mounted is another question. In general we cannot assume CVMFS availability out of the box. This then also means that site specific settings cannot be assumed to be available from CVMFS and definitely not locally either as the site is not CMS.

The way around these complications is a wrapper script in the glideinWMS pilot that uses parrot [9] to effectively mount (even though the actual mechanism is not really a mount) a CVMFS file system and make it available to the job. The same wrapper script can also provide site specific settings as if they were local.

This modus operandi was tested last year using a handful of non-CMS OSG [6] sites that allowed opportunistic access for CMS jobs. We got to a point where we could sustain operations accessing the handful of sites as a single meta-site for about a week at times, reaching scales of a few hundred jobs on average with peaks going above a thousand jobs. Unfortunately we never really completed commissioning and moving this into production, but we are planning to come back to this.

4.2. Cloud resources
GlideInWMS in principle supports any cloud that provides an interface compatible with Amazon Elastic Compute Cloud EC2 (although in practice it depends on the exact implementation of the EC2 interface as there can be differences between the various cloud middlewares). The way the integration works is very similar to the pilot submission model for grid resources, instead we instantiate a virtual machine on the cloud resource and that VM then has the pilot built-in.

CMS controls the VM image we instantiate, we fully control the software environment and can include CVMFS and local contextualization to point at site specific settings on CVMFS.

Assuming a cloud middleware that is supported by glideinWMS (for instance Amazon AWS, Openstack, OpenNebula) we can easily make use of any cloud resource that is made available for CMS use. Depending on the size of the cloud resource available, the simple site specific settings mentioned above (data reads via the CMS xrootd federation, remote stage-out to a CMS site, remote proxy serves at a CMS site) might not be suitable for stable/efficient operation though.

A large scale example that follows this model are the CMS Tier0 resources at CERN, which are all on the CERN Agile Infrastructure, an Openstack based cloud. Even though this is not an opportunistic resource per se, it follows exactly the same access model, with the major difference that these resources are provided as part of pledged resources to CMS.

For more on the CMS Tier0 and other uses of clouds by CMS see [8] and [10].

4.3. Batch clusters
Batch clusters cover a wide variety of resources from small clusters at the university or even single research group level up to SuperComputer centers. Anything that has purely local batch capacities without exposing them to the outside world via either a grid or cloud interface can be included in this category.
The biggest problem for using these resources is in how to get the pilot running on them. Without a grid CE or cloud controller to connect to, the glideinWMS factory has no way to get a pilot on the resource.

Apart from the pilot submission problem, this type of resources has the same problems regarding CVMFS access and the related access to site specific settings as non-CMS grid sites. Therefore if we can somehow solve the pilot submission problem, CVMFS and site specific settings can be provided via the parrot pilot wrapper script in the same way as for non-CMS grid sites.

CMS at the moment is working on integrating allocations at two different SuperComputers, SDSC and NERSC, into our computing infrastructure. At SDSC we are dealing with a simplified problem, since we could convince SDSC to provide access to the CMS CVMFS repository on the SDSC worker nodes. Therefore we did not have to use the parrot wrapper script, although we still had the problem of getting the glideinWMS pilot on the SDSC worker nodes. At NERSC we have been dealing with the full problem set, no direct access to the worker nodes for pilot submission and no CVMFS on the worker nodes.

5. BOSCO mode glideinWMS pilot submission

This is not the first time CMS has used SDSC. We have run a short ReReco campaign there and at that time we implemented a pilot submission model which is shown in figure 1 and described in more details in [11]. Unfortunately the technical implementation of integrating SDSC into glideinWMS then, while functional and full-filling our needs at the time, turned out to not be maintainable in the long run. This time we re-implemented the same architecture more or less from scratch, with special care taken to end up with a long term maintainable setup.

Figure 1. BOSCO/glideinWMS integration

This architecture is based on BOSCO, an OSG supported software that allows researchers to access their accounts at remote batch clusters from their local desktop or laptop. BOSCO utilizes HTCondor and a feature of HTCondor to submit a job to a remote batch system via an ssh tunnel and a gateway host. Complementing the architecture is a small piece of code that needs to be setup once on the gateway host and which handles the submission to the local batch system.
For glideinWMS integration we only use the piece of code on the gateway host (which has to be setup once). The glideinWMS factory now fully supports the remote pilot submission to such a gateway host out of the box.

6. SDSC status
SDSC is fully integrated into glideinWMS and the CMS computing infrastructure. We have been running real (not just test) workflows there since April 2015 with slowly increasing scales. Figure 2 shows running CMS jobs at SDSC over a 5 day span beginning of May 2015. We will keep pushing the scales further in order to use up our current allocation before it expires.

![Figure 2. CMS jobs at SDSC](image)

We only exercised MC production workflows so far, but will soon switch to also run workflows that read input data, most likely via xrootd from the CMS T2 at UCSD. All these workflows only execute the first stage processing at SDSC, the output of these jobs is then staged out to Fermilab and merge jobs are run at Fermilab.

7. NERSC status
NERSC is also already integrated into glideinWMS and the CMS computing infrastructure, but we are still actively working on making the parrot wrapper fully functional. This is almost working now, but not yet at a point where we are ready to run real production workflows at NERSC. We are using NERSC as a testbed for refining the procedures to use the parrot wrapper
script, once it works at NERSC the plan is to apply the same settings to other resources where we can make use of it.

8. Summary and Outlook
CMS is actively looking at integrating opportunistic resources into its computing infrastructure in order to alleviate some of the resource constraints imposed by the challenging LHC Run2 computing environment.

We can already integrate cloud based opportunistic resources more or less out of the box. Batch cluster are also usable out of the box provided CVMFS is available. There is still some manual setup work to be done there, we are planning to work on further integration to simplify this procedure.

We are also still working on full integration of the parrot wrapper script into the glideinWMS pilot. Once that is done, it opens up access to non-CMS grid sites and batch clusters where we cannot rely on having access to CVMFS.

References
[1] S. Chatrchyan et al. [CMS Collaboration], “The CMS experiment at the CERN LHC”, JINST 3, S08004 (2008).
[2] 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).
[3] HTCondor homepage (accessed on Oct 28, 2015): http://research.cs.wisc.edu/htcondor/
[4] 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.
[5] 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).
[6] R. Pordes et al., “The Open Science Grid”, J. Phys. Conf. Ser. 78, 012057 (2007).
[7] James Letts, “Using the glideinWMS System as a Common Resource Provisioning Layer in CMS”, proceedings of CHEP2015
[8] Dirk Hufnagel, “The CMS Tier0 goes Cloud and Grid for LHC Run 2”, proceedings of CHEP2015
[9] Parrot homepage (accessed on Oct 28, 2015): http://ccl.cse.nd.edu/software/parrot/
[10] David Colling “The diverse use of clouds by CMS”, proceedings of CHEP2015
[11] 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).

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.