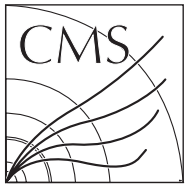
Available on CMS information server

CMS CR -2013/369



The Compact Muon Solenoid Experiment

Conference Report

Mailing address: CMS CERN, CH-1211 GENEVA 23, Switzerland

29 October 2013

CMS experience of running glideinWMS in High Availability mode

Igor Sfiligoi, J Letts, S Belforte, A McCrea, K Larson, M Zvada, B Holzman, PMhashilkar, D C Bradley, M D Saiz Santos, F Fanzago, O Gutsche, TMartin and F Wrthwein

Abstract

The CMS experiment at the Large Hadron Collider is relying on the HTCondor-based glideinWMS batch system to handle most of its distributed computing needs. In order to minimize the risk of disruptions due to software and hardware problems, and also to simplify the maintenance procedures, CMS has set up its glideinWMS instance to use most of the attainable High Availability (HA) features. The setup involves running services distributed over multiple nodes, which in turn are located in several physical locations, including Geneva, Switzerland, Chicago, Illinois and San Diego, California. This paper describes the setup used by CMS, the HA limits of this setup, as well as a description of the actual operational experience spanning many months.

Presented at *CHEP2013 Computing in High Energy Physics 2013*

CMS experience of running glideinWMS in High Availability mode

I Sfiligoi[1], J Letts[1], S Belforte[2], A McCrea[1], K Larson[3], M Zvada[4], B Holzman[3], P Mhashilkar[3], D C Bradley[5], M D Saiz Santos[1], F Fanzago[6], O Gutsche[3], T Martin[1] and F Würthwein[1]

[1]University of California San Diego, 9500 Gilman Dr, La Jolla, CA 92093, USA
[2]INFN Sezione di Trieste, Via Valerio 2, 34127 Trieste, Italy
[3]Fermi National Accelerator Laboratory, P.O. Box 500, Batavia, IL 60510, USA
[4]Karlsruhe Institute of Technology, Wolfgang-Gaede-Str. 1, 76131 Karlsruhe, Germany
[5]University of Wisconsin - Madison, 1150 University Ave, Madison, WI 53706, USA
[6]INFN Sezione di Padova, Via Marzolo 8, 35131 Padova, Italy

isfiligoi@ucsd.edu

Abstract. The CMS experiment at the Large Hadron Collider is relying on the HTCondor-based glideinWMS batch system to handle most of its distributed computing needs. In order to minimize the risk of disruptions due to software and hardware problems, and also to simplify the maintenance procedures, CMS has set up its glideinWMS instance to use most of the attainable High Availability (HA) features. The setup involves running services distributed over multiple nodes, which in turn are located in several physical locations, including Geneva, Switzerland, Chicago, Illinois and San Diego, California. This paper describes the setup used by CMS, the HA limits of this setup, as well as a description of the actual operational experience spanning many months.

1. Introduction

In recent years, the pilot paradigm has become the dominant way of using widely distributed computing resources for the scientific communities. One pilot product is the glideinWMS[1], which is being used by the CMS experiment at the Large Hadron Collider[2], among others, to manage compute resources deployed on the Open Science Grid (OSG)[3][4] and the European Grid Initiative (EGI)[5].

A pilot system creates a global, dynamic, private overlay batch system on top of leased resources obtained from possibly many sources. As such, it becomes a critical component of the computing model for any scientific community relying on it. It is thus important that the pilot system does not have any single point of failure, and that it also allows for maintenance activities without drastically disrupting the global computing activity. The methods employed to avoid single point of failure are typically called High Availability (HA) features.

This paper presents the HA features of the glideinWMS product, alongside the CMS experience of using it. The glideinWMS product and its HA capabilities are described in section 2, while the setup used by CMS and their operational experience is described in section 3.

2. The glideinWMS and its High Availability capabilities

The glideinWMS pilot system has a clear separation between the provisioning and the scheduling layer, as shown in figure 1. Only the provisioning layer is really glideinWMS specific; the scheduling layer is handled by the HTCondor product (formally known as Condor)[6], without any glideinWMS-specific modifications to its code. This section provides an analysis of the High Availability (HA) capabilities of both layers, since both are needed for the successful operation of the system.

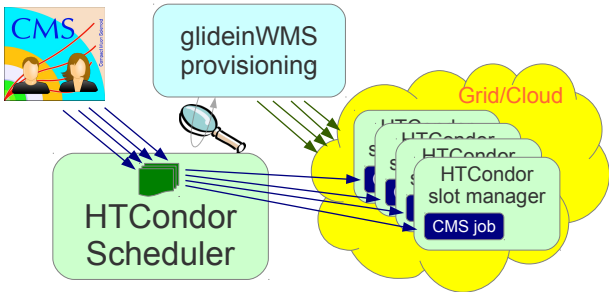


Figure 1. The glideinWMS layers

Each of the two layers is composed of several services, as shown in figure 2. Please note that the management of the compute resource, i.e. the slot manager of the scheduling system, does not have to be done in HA mode. This service only handles a single compute resource, so loosing it in the event of the compute resource problem is acceptable, since no user job could use this resource anyhow.

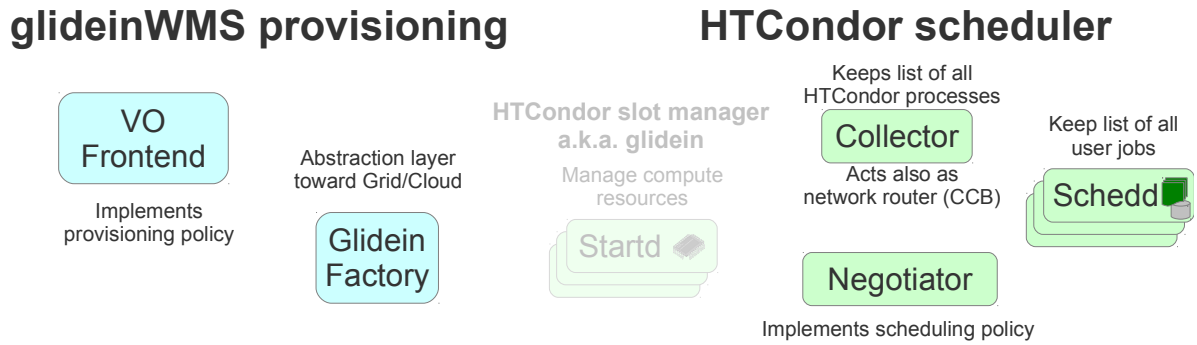


Figure 2. The glideinWMS services

The rest of this section provides an analysis of the capabilities of each of the above involved services.

2.1. Glidein Factory

A Glidein Factory acts as an abstraction layer toward the Grid and Cloud resource providers. It contains no decision logic, performing provisioning activities based on requests from VO Frontends. This allows for several Glidein Factories to serve a single VO Frontend, both for High Availability and Load Balancing reasons.

Moreover, once a resource is provisioned, the Glidein Factory service is not involved in the management of that resource anymore, so the loss of the Glidein Factory is of no consequence. And as long as there are others that can pick up the increased load, it is completely transparent to the users.

2.2. VO Frontend

A VO Frontend implements the provisioning logic for the glideinWMS overlay batch system. It monitors the HTCondor scheduling services looking for imbalances between available resources and

the needs of the user jobs, and issues provisioning requests to one of more Glidein Factories, as appropriate. As a consequence, a VO Frontend service has no persistent state.

In theory, one could thus run two VO Frontends that regulate the same glideinWMS overlay system, and achieve a HA setup of this service. Unfortunately, the two VO Frontends would be unaware of each other, possibly resulting in over-provisioning of the resources, although the effect should be limited, due to the logic of constant pressure used[7].

It is important to note that the VO Frontend does not need to be truly Highly Available. A downtime of the order of an hour is acceptable and would only impact the growth of the overlay system. The already provisioned resources will not be affected. Therefore the recommended setup is to have two VO Frontends configured, one which is running and a second cold spare.

2.3. Collector

The Collector is the reference point of the HTCondor scheduling layer. It contains the list of all other HTCondor services, including the slot managers. It also acts as a network router for bridging firewalls, which is usually referred to as Condor Connection Brokering (CCB)[8].

The Collector does not need to store any persistent state. The other HTCondor daemons send their own information to the Collector on a regular basis, letting it know that they are still alive. As a consequence, one can achieve HA setup by deploying two independent Collector instances, and instructing all the other HTCondor services to talk to both of them. If one instance goes down, the other still has all the information. It should be noted that, even when both instances are operating, they both handle all the data, so there is no load balancing involved. This functionality is fully functional only since HTCondor 8.0.1.

2.4. Negotiator

The Negotiator implements the scheduling policy of queued user jobs to the available resources. One of the scheduling parameters is the current user priority, which is based on past usage of resources by that user's jobs. This information cannot be derived from any other source, so the Negotiator does have a persistent state.

As a consequence, one cannot have two independent Negotiators running. In order to achieve High Availability, HTCondor provides mechanisms to have several Negotiators in hot spare setup, where only one is active at any given time and its persistent state is continuously being synced to the other participating Negotiators. If the primary Negotiator goes down, one of the spares will be automatically started, using the persistent state that it has available locally. This process is managed by the High Availability Daemon (HAD), which comes standard with HTCondor.

2.5. Schedd

The *Scheduling Daemon*, or Schedd service, is responsible for accepting and managing user jobs, including any input sandboxes then may have. As such, it does have a persistent state, and this persistent state has typically a large footprint.

This large persistent state makes it impractical to put in place replication strategies between Schedd instances which are not closely co-located. HTCondor does provide HA options for when a shared file system is available, based again on HAD, but this was not considered viable for CMS, so we will not discuss it further in this paper.

3. The CMS computing and its glideinWMS setup

The CMS computing model[2] has three tiers of computing facilities connected by high-speed networks up to 10 Gbps. Data flows within and between these tiers. These include the Tier-0 at CERN, used for data export from CMS and archival to tape as well as prompt reconstruction of data, and 7 Tier-1 centers used for the tape backup and large-scale reprocessing of CMS data and the distribution of data products to the Tier-2 centers. There are about 50 Tier-2 facilities, typically located at

universities, where physics data analysis and Monte Carlo production are carried out. Approximately 50,000 job slots are available to be shared equally between production and analysis at the Tier-2 sites. Recently analysis and Monte Carlo production activities have been expanded to include certain Tier-1 sites when job slots are available, as well as smaller so-called “Tier-3” centers, usually processor farms hosted at universities which may or may not provide storage capacity.

Over the past few years, CMS has come to rely on the glideinWMS system to handle most of its distributed computing needs for both physics analysis and service computing across all tiers. At this time CMS physics analysis and the CMS service computing groups have two separate infrastructures for scheduling, with each having its own VO Frontend at the provisioning layer as well. The two, however, do share a common subset of Glidein Factories, since they lease resources from the same set of resource providers. Due to space constraints, the rest of this paper only presents the details of the physics analysis infrastructure. The differences between the two are however relatively minor, and the implications of any relevant feature that is specific to only one of the two is noted in the text. Moreover, in the near future it is foreseen to merge the two infrastructures, thus forming a single logical global queue of jobs for CMS[9].

3.1. Glidein Factories

The Glidein Factories used by CMS are not necessarily part of the CMS infrastructure. They may serve many different experiments and organizations. Currently there are four which CMS uses; three are in three different availability zones in the USA, and one is in Europe, at CERN.

Over the past few years we had several prolonged downtimes of one of the factories either due to regular software maintenance or hardware problems. None of these events had any significant impact on CMS computing.

3.2. VO Frontend

CMS operates a single instance of the VO Frontend. Software maintenance is regularly performed on it, and the relatively short downtimes have never been a significant problem for CMS computing.

The VO Frontend configuration is regularly being backed up, and that is enough to promptly re-create a new instance on different hardware if needed.

3.3. Collector and Negotiator

CMS runs a Collector and a Negotiator service on the same hardware, as it is typical for most HTCondor setups. Since summer of 2013, two instances of the service pair are deployed, using the methods described in sections 2.3 and 2.4. It is worth noting that CMS moved from a non-HA to an HA setup on the live system, with O(30k) resources being managed at that time.

The two nodes are in two different availability zones, one in the UCSD Physics Department and one in the San Diego Supercomputer Center. They are however still tied geographically, so CMS may seek to deploy a third collector in Europe or somewhere else in the world for increased availability.

Nevertheless, the current HA setup has served CMS well. CMS has so far not experienced any loss of availability since this deployment. Previously CMS would experience an outage a few times a year due to power failures, cooling failures or scheduled maintenance of the machine room where the non-HA collector was hosted.

3.4. Schedd

As explained in section 2.5, HTCondor does not provide useful a means for achieving HA of the Schedd on the submission node. CMS thus deployed several submission nodes in several different availability zones; three in three different availability zones in the USA, and two in a single availability zone in Europe.

CMS physics analysis uses CRAB, the analysis job framework of CMS, over gsissh[11] to submit jobs to the Schedd. CRAB is instrumented to spread the jobs among many submission nodes. This limits the damage in case one of the submission nodes becomes temporarily unresponsive; the jobs

running on that node may be lost, but the provisioned resources may be used by jobs from the other submission nodes. Moreover, CRAB will use the remaining submission nodes for new submissions, avoiding further impact to the users.

And even in case of permanent loss of a submission node, the damage is still manageable, since the length of time of any one task sitting in the queue is typically of the order of a day. The lack of full HA capability thus does not present a tremendous gap in functionality at this time.

The CMS organized production workflow is however slightly different. The tasks in the organized queues persist over weeks or months, so losses are potentially more troublesome there. CMS is currently still assessing what should be the appropriate solution there.

4. Conclusions

CMS has come to rely on the glideinWMS system to handle most of its distributed computing needs. To minimize damage from both scheduled maintenance, temporary electrical outages and outright failures of the hardware, CMS has deployed the system over many availability zones and is relying on the High Availability feature of glideinWMS to gracefully handle any single node failures.

The experience so far has been very positive, with no major problems encountered so far. We are however a little worried about the lack of proper HA functionality of the Schedd, and will continue to investigate what a proper solution should be.

Acknowledgements

This work was partially sponsored by the US National Science Foundation Grants No. PHY-1148698 and PHY-1120138.

References (to be written)

[1] Sfiligoi I, Bradley D C, Holzman B, Mhashilkar P, Padhi S and Würthwein F 2009 The pilot way to grid resources using glideinWMS *Comp. Sci. and Info. Eng., 2009 WRI World Cong. on* **2** 428-432 doi:10.1109/CSIE.2009.950

[2] Chatrchyan S et al. 2008 *J. of Instr.* **3** S08004 doi:10.1088/1748-0221/3/08/S08004

[3] Pordes R et al. 2007 *J. Phys.: Conf. Ser.* **78** 012057 doi:10.1088/1742-6596/78/1/012057

[4] Sfiligoi I, Würthwein F, Dost J M, MacNeill I, Holzman B and Mhashilkar P 2011 Reducing the human cost of grid computing with glideinWMS *Proc. Cloud Computing 2011* (Rome, Italy) 217-221 ISBN 978-1-61208-153-3 http://www.thinkmind.org/index.php?view=article&articleid=cloud_computing_2011_8_40_20068

[5] Kranzlmüller D, Marco de Lucas J and Öster P 2010 *Remote Instr. and Virt. Lab.* 61-66 doi:10.1007/978-1-4419-5597-5_6

[6] Thain D, Tannenbaum T and Livny M 2005 Distributed computing in practice: the Condor experience Concurrency and Computation: Practice and Experience **17 2-4** 323-356 doi:10.1002/cpe.938

[7] Sfiligoi I, Hass B, Würthwein F and Holzman B 2011 Adapting to the unknown with a few simple rules: the glideinWMS experience *Proc. ADAPTIVE 2011* (Rome, Italy) 25-28 ISBN 978-1-61208-156-4 http://www.thinkmind.org/index.php?view=article&articleid=adaptive_2011_2_20_50040

[8] Bradley D, Sfiligoi I, Padhi S, Frey J and Tannenbaum T 2010 *J. Phys.: Conf. Ser.* **219** 062036 doi:10.1088/1742-6596/219/6/062036

[9] Gutsche O et al. TBD Evolution of the pilot infrastructure of CMS: towards a single glideinWMS pool [10] *Proc. CHEP 2013* (Amsterdam, NL)

[10] Belforte S et al. TBD Using ssh as portal, the CMS CRAB over glideinWMS experience *Proc. CHEP 2013* (Amsterdam, NL)