EPJ Web of Conferences **245**, 03023 (2020) https://doi.org/10.1051/epjconf/202024503023
CHEP 2019

Exploiting CRIC to streamline the configuration management of GlideinWMS factories for CMS support

Jeffrey Dost[1,*], *Marco* Mascheroni[1,**], *Julia* Andreeva[2], *Alexey* Anisenkov[3,4]*Dennis* Box[5], *Alessandro* Di Girolamo[2], *Saqib* Haleem[6], *Edita* Kizinevič[7], *Krista* Majewski[5], *James* Letts[1], *Lorena* Lobato Pardavila[5], *Bruno* Moreira Coimbra[5], *Antonio* Pérez-Calero Yzquierdo[8,9], *Marco* Mambelli[5], *Panos* Paparrigopoulos[2], and *Marian* Zvada[10]

[1]University of California San Diego, La Jolla, CA, USA
[2]European Organization for Nuclear Research, Meyrin, Switzerland
[3]Novosibirsk State University, Novosibirsk, Russia
[4]Budker Institute of Nuclear Physics, Novosibirsk, Russia
[5]Fermi National Accelerator Laboratory, Batavia, IL, USA
[6]National Centre for Physics Rd, Islamabad, Pakistan
[7]Vilnius University, Vilnius, Lithuania
[8]Centro de Investigaciones Energéticas Medioambientales y Tecnológicas, Madrid, Spain
[9]Port d'Informació Científica, Barcelona, Spain
[10]University of Nebraska-Lincoln, Lincoln, NE, USA

Abstract. GlideinWMS is a workload management and provisioning system that allows sharing computing resources distributed over independent sites. Based on the requests made by GlideinWMS frontends, a dynamically sized pool of resources is created by GlideinWMS pilot factories via pilot job submission to resource sites' CEs. More than 400 CEs are currently serving more than ten virtual organizations through GlideinWMS, with CMS being the biggest user with 230 CEs. The complex configurations of the parameters defining resource requests, as submitted to those CEs, have been historically managed by manually editing a set of different XML files. New possibilities arise with CMS adopting the CRIC, an information system that collects, aggregates, stores, and exposes, among other things, computing resource data coming from various data providers. The paper will describe the challenges faced when CMS started to use CRIC to automatically generate the GlideinWMS factory configurations. The architecture of the prototype, and the ancillary tools developed to ease this transition, will be discussed. Finally, future plans and milestones will be outlined.

1 Introduction

The Compact Muon Solenoid (CMS) experiment is one of two multipurpose experiments that collect and analyze data from proton-proton and heavy ion collisions at the Large Hadron Collider (LHC), in Geneva, Switzerland. CMS, as well as other High Energy Physics (HEP) experiments, uses Grid, Cloud, and high performance computing (HPC) centers to satisfy its

*e-mail: jdost@ucsd.edu
**e-mail: marco.mascheroni@cern.ch

computing needs, while GlideinWMS [1] is the workflow management system used by CMS to access these compute resources.

A user HTCondor [2] pool, that shrinks and grows based on user requests, is *dynamically* created by sending *pilot* jobs [3] to many resources located all around the world. Pilot jobs are responsible for validating the resources they are accessing, and then they join the user HTCondor pool in order to run user jobs. As shown in figure 1, two components form the GlideinWMS system: a frontend and a factory.

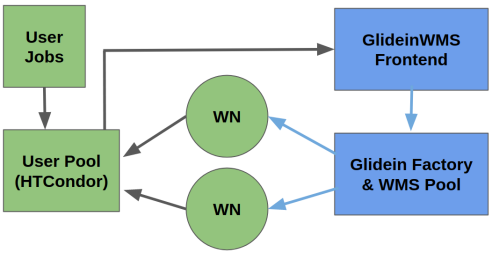

Figure 1. The components of the GlideinWMS system. In order to create a dynamic HTCondor pool, a frontend and a factory work together determining the pressure from users' jobs and sending pilot jobs to Grid, Cloud, and HPC resources. Pilot jobs running on the worker nodes then start the condor daemons that are necessary to join the central user pool, and finally start user jobs

The factory is responsible for sending the pilot jobs to the different resources. Resources belonging to the same institute form a *site*. Compute Elements (CEs), the access points of Grid and HPC resources, are a set of services that provide an interface for pilot jobs to access a local resource management system, and also take care of authentication, authorization and delegation of jobs. A site can have multiple CEs, which, in turn, can have multiple *queues*. Queues give the site the ability to expose different machine types, and restrict access to machines of a given type for specific user communities. In the GlideinWMS factory, *entries* represent the set of parameters required to properly configure the factory to submit and run a pilot on a particular queue on a CE. The frontend looks at user and central production job pressure, and then decides how many pilot jobs the factory should send to each entry, based on the user's job requirements and the current load on each entry. Pilot jobs will first execute a set of tests to make sure the resources they are accessing are valid, checking for example that all the software necessary to execute the user jobs is installed on the worker nodes, and then they will connect to the HTCondor user pool in order to start executing user jobs. In GlideinWMS, different experiments, which are called Virtual Organizations (VOs), run one or more different frontends, which are in turn connected to one or more factories. Various multi-experiment redundant factories are used for high availability.

Factory configurations are a set of XML files that contain all of the parameters for each entry. Currently, there are more than 400 entries in those XML files which are manually maintained and correspond to about 16000 lines of XML. In this paper we discuss a new solution that CMS adopted in order to reduce the operational effort needed to maintain the factory configurations. This solution exploits the CMS Computing Resource Information Catalogue (CRIC) [4] to automatically generate a subset of the CMS factory configurations.

The rest of this paper is organized as follows. Section 2 explains more in detail how factory operations works. Section 3 introduces CRIC, while Section 4 discusses how CRIC has been used to build a prototype that can automatically generate factory configurations. Section 5 gives more detail about how the prototype has been used. Sections 6 and 7 outline future directions and draw some conclusions, respectively.

2 CMS Factory Operations

Factories are operated by a dedicated team of 2.5 full-time equivalent workers. The team is responsible for ensuring that the GlideinWMS system is optimally delivering useful resources to users when these are requested, while trying to minimize the waste of resources in doing so.

One of the major time-consuming activities for the factory operations team is maintaining all of the XML files containing the entries. Those configuration files have been populated over several years with site resources. A site administrator who wants to connect their cluster to the factory and start receiving jobs from CMS opens a ticket with all of the CE queue information. Factory operators check if the information is complete and request clarifications if needed. VO-specific information is also added to the entry, and factory operators may require input from other CMS experts in order to set them. When the information is complete, the operators add the entry to a test factory. Test pilots are then sent to the site to verify all the validation scripts are executed correctly. If everything is correct, the entry can finally be added to the production factories.

The entire process of configuring, testing, and validating a new factory entry or a change to an existing entry is very time consuming and requires many interactions between the factory operations team and site administrators. It usually takes one week from the opening of the ticket to the addition of the entry to the production factory. Since plenty of room for optimization is available, the factory operations team started to evaluate possibilities to automate parts of this process.

The entire factory configuration was analyzed, and we determined there are four different categories of entry parameters:

1. Resource description parameters that are published in Grid-infrastructure owned information systems: CE hostname, CE type (ARC, CREAM, HTCondor, etc.), queue name, resource name, supported VOs, etc.

2. Unpublished resource description parameters: working directory, maximum number of jobs, operating system, etc.

3. Internal GlideinWMS parameters: Grid submission rates, entry names, etc.

4. VO configuration parameters: pilot CPU / memory configuration, etc.

Category 1 includes attributes needed in order to submit to a given CE queue at a site. They can be found in various information systems (e.g. Open Science Grid (OSG) Topology [5], EGI Grid Configuration Database (GOCDB) [6]), with each information system containing information about a specific set of attributes. A given site can have multiple CEs, each accessible by a physical machine hostname. There are a few implementations of CE software. In order to submit to a given CE, one must know the hostname, the implementation type and the corresponding port, and various fields may need to be set to select a queue depending on site configuration. As mentioned in Section 1, a particular CE can expose various resource types described by queues. A queue may represent a subset of machines a VO is allowed to run on, and enforce constraints on a VO pilot that lands on the machine, such as maximum allowed memory, walltime, and number of CPUs. Depending on the CE implementation, the submission parameters might require only a logical queue name, or one may need to configure the constraints directly.

Unfortunately, the parameters available on each information systems differ and some parameters are never published, so there are parameters that currently a site administrator must

communicate directly to factory operators to completely configure an entry. These parameters fall under Category 2. Category 3 contains parameters that are internal to GlideinWMS, which need to be set according to factory operations experience, for example in order to regulate the pilot submission thresholds and rates by the factory for the specific resource represented in the entry. Finally, Category 4 parameters in an entry are used to configure the pilot based on VO policy. For example, a targeted resource might have up to thirty-two core machines, but CMS would prefer to configure eight core pilots to run at that site.

As discussed in Section 3, some of the information in these categories were already available in CRIC.

3 The Computing Resource Information Catalogue (CRIC)

In 2015 the OSG [7] announced plans to stop using BDII [8] to publish their computing resources. This triggered a review of information systems in Worldwide LHC Computing Grid (WLCG). As part of a trend in CMS (and HEP at large) to leverage community software projects, CMS began to evaluate the adoption of CRIC in 2016 as its information service. CRIC is a community project based on a refactorization of AGIS [9], the ATLAS Grid Information System.

CRIC gathers and allows access to information about physical and logical computing resources in CMS. CRIC is made up of two components, CRIC Core, and CRIC CMS. CRIC Core provides descriptions of resources (Category 1) and gathering information from GOCDB for EGI sites, and OSG Topology for OSG sites, respectively. CRIC CMS contains experiment-specific information on how CMS organizes these resources (Category 4). For example, CRIC CMS contains the list of CMS sites available on the Grid. APIs are available to retrieve this information from both CRIC Core and CRIC CMS in JSON format. In 2018, it was decided to create interfaces for factory operators in order to leverage CRIC and automate the factory configuration generation.

There are multiple advantages in this approach. CRIC represents a central and public place where all the site configurations are available via the web for everybody. The factory operations workflow can be streamlined, since less back and forth with site administrators is required to update and validate information, which in turn reduces the workload on the factory operations team. Web forms are easier to fill in than big XML files; therefore, a site administrator might be able to insert information directly into CRIC instead of factory operators doing it on their behalf. CRIC can also be used as a starting point for implementing more automated tools, such as a method to automatically send test jobs once an entry is created in CRIC.

4 Architecture

4.1 CRIC CMS User Interface

While CRIC Core provides a mostly complete list of all available CEs and queues per CE for a site, currently it does not provide all the details necessary to completely configure a GlideinWMS entry. In order to remedy this, the CRIC CMS component was used to bridge the missing information. In the initial integration of GlideinWMS factories with CRIC, i.e. using factories as a data source, the CMS API contains the rest of the fields necessary to complete a factory entry. CMS uses this information for other functions outside the scope of this paper, such as the SiteDB [10]. In order to invert the logic so that factory entries can be generated from CRIC and at the same time not disrupt consumers of CRIC data when the factories are removed as a data source, the CRIC developers worked with factory operations to

create a web-based site administrator User Interface (UI). This UI allows a site administrator or a factory operator to directly configure any fields required beyond what is already exposed in the CRIC Core API in order to complete the missing data needed to generate an entry (Categories 2-4). The site administrator UI then acts as the new input for the CMS API, which ultimately replaces the factory itself as a CRIC data source.

4.2 Auto Entry Generation

Two Python scripts have been deployed to run on the factory to maintain submit point entries. The first script periodically parses the data from the CRIC JSON files and keeps an up-to-date local copy of the data on disk. The second script is responsible for merging the CRIC data with operator-provided user data, which gives the operator the ability to whitelist the desired entries to be propagated to the final factory configuration, as well as the opportunity to define new attributes, or override any attribute associated with the entry that was already retrieved from CRIC.

YAML was chosen as the file format for the intermediate files stored on disk. The reason for this is it needs to be a human-friendly readable format since operators need to look at the auto-generated files and make decisions about what to override. Factory operations experience has proven working with XML files to be cumbersome as a configuration file format that humans read and modify.

The ability to whitelist entries and manually add or override attributes are important because the factory operations team usually will perform an audit of the entry configuration and need a quick turnaround in case the generated entry is incorrect. In fact, auto-generated entries might contain errors because the data contained in Grid information not being always accurate, and fixing the errors might require long iterations with the responsible of the resource. Moreover, as discussed in Section 2, information systems typically only contain resource attributes from Category 1, but frequently operators must fill in custom fields to properly configure the pilot for a special resource.

5 Proof of Concept

The eight CMS Tier 2 sites in the US (UCSD, Purdue, Caltech, Florida, MIT, Nebraska, Wisconsin, and Vanderbilt) were selected as candidates to test the auto-generation of entries from CRIC data. YAML whitelist files were manually edited, and the new scripts were used to automatically generate the factory XML configuration in a production environment. These sites were ideal because they are representative of most of the entries in the factory configuration. Figure 2 illustrates the immediate gains due to integrating the improved setup. Before having the ability to auto-generate the entries, most of the configuration that would have been put in place by hand would have contained lots of repetitive template code that is prone to error. This boiler plate code is now pulled from the default YAML file instead. Resource-specific fields are almost completely taken from a combination of CRIC Core and site administrator UI. The manual override YAML file that the factory operator must maintain is reduced to simply whitelisting the entry, and overriding a handful of fields as needed. Besides the great simplification of the configuration file, there is a reduction in the number of interactions between site administrator and factory operator, since the site administrator has direct control of the fields describing their site by virtue of the CRIC UI, and consequently a reduction in time needed to deploy an entry.

While running the auto-generation in a production environment has proven successful, there is still some room for improvement in the current implementation. The first issue is

Figure 2. Old manual XML entry compared to the new YAML override entry. The reduction in the number of lines to maintain is clearly visible since most of the XML parameters are now in CRIC or the default YAML file

that the site administrator UI is still predominantly designed from the perspective of factory operations team. For example, the list of accepted VO has specific naming schemes that are case-sensitive and are internal to factory operations. A site administrator would not know to put "CMS instead of "cms", "OSGVO" instead of "osg", and so on. Similar conventions exist for Operating System (OS) flavor. For example, knowing to spell out "RHEL7" instead of "sl7", and worse, knowing to use the special key word "any" if the site supports containers. Finally, a site administrator would have to know that factory operations uses specific units for numeric fields, like seconds for maximum walltime, or megabytes for maximum memory. These are only a few examples of why configuring a factory entry is not very user-friendly for a site administrator.

Another limitation of the current implementation is that initially the factory operations team focused only on the auto-generation of CMS entries. However, the scope of factory operations is beyond CMS and WLCG. For example, there are some exotic configurations specific to non-CMS VOs that do not need to be considered in the CMS CRIC case and would only over-complicate the implementation. The longer term solution for auto-generation of entries for factory operations as a whole should work for all sites. CRIC gives factory operations the extra help in auto-generating CMS and WLCG entries, but other tools will need to be developed to interface other external information systems and databases to handle the non-WLCG, non-CMS sites that the factory also supports.

6 Future Plans

In order to address the issues discussed in Section 5, we have begun developing a new implementation focusing on OSG CEs that are not in CRIC. The OSG CE Collector is the OSG replacement of BDII. It is a simple HTCondor collector that contains descriptions of all of the CEs in the OSG. Coincidentally, it is one of the data sources CRIC Core consumes for

WLCG sites in the OSG. The idea behind the new implementation is to use CRIC for all sites where it is appropriate, then for non-WLCG OSG sites, directly parse the OSG CE Collector for the data required to configure the entries. One downside is for these sites, site administrators do not get the nice CRIC UI. However for OSG it is not an issue because administrators are used to using the OSG Configure tools to propagate the resource descriptions into the OSG CE Collector. Learning how OSG Configure works has also given the factory operations team a better understanding what a site administrator-centric interface looks like. So the operations team can take this and work with the CRIC developers to decouple the factory specific peculiarities of the current entry creation interface, and improve the user experience for site administrators.

7 Conclusions

The CMS GlideinWMS factory is an important piece of the computing infrastructure of CMS. It has been historically operated by managing a set of XML files which contains more than 400 entries, corresponding to about 16000 lines of XML. In order to reduce the effort needed to maintain the factory, the factory and the CRIC teams worked together on a new solution that allows CMS factory operators to leverage the CRIC interface to insert and manage entries.

The solution has been validated in a production environment, and the factory operations team is now evaluating how to address a wider variety of use cases in addition to CMS VO and WLCG Sites.

This work was partially supported by the U.S. Department of Energy, the National Science Foundation, and by Spain's Ministry of Economy and Competitiveness grant FPA2016-80994. CMS thanks our partners in the GlideinWMS, HTCondor, and CRIC development teams, the OSG, and our colleagues at CERN, all of whom make the shared computing infrastructure a success.

References

[1] P. Mhashilkar, M. Mambelli, I. Sfiligoi, B. Holzman, K. Majewski, J.M. Dost, D. Box, M. Mascheroni, J. Weigand, L. Lobato et al., *glideinwms/glideinwms: v3.4* (2018), https://doi.org/10.5281/zenodo.1309679

[2] *HTCondor public web site*, https://research.cs.wisc.edu/htcondor/

[3] I. Sfiligoi, D.C. Bradley, B. Holzman, P. Mhashilkar, S. Padhi, F. Wurthwein, *The Pilot Way to Grid Resources Using glideinWMS*, in *Proceedings of the 2009 WRI World Congress on Computer Science and Information Engineering - Volume 02* (IEEE Computer Society, Washington, DC, USA, 2009), CSIE '09, pp. 428–432, ISBN 978-0-7695-3507-4, http://dx.doi.org/10.1109/CSIE.2009.950

[4] A. Anisenkov, J. Andreeva, A. Di Girolamo, P. Paparrigopoulos, A. Vedaee, EPJ Web Conf. **214**, 03003 (2019)

[5] *OSG Topology web site*, https://topology.opensciencegrid.org/

[6] *GOCDB web site*, https://goc.egi.eu/

[7] *OSG public web site*, https://opensciencegrid.org/

[8] A. Osman, A. Anjum, N. Batool, R. McClatchey, *A Fault Tolerant, Dynamic and Low Latency BDII Architecture for Grids* (2012), Vol. abs/1202.5512, 1202.5512, http://arxiv.org/abs/1202.5512

[9] A. Anisenkov, A. Di Girolamo, M. Alandes Pradillo, Journal of Physics: Conference Series **898**, 092023 (2017)

[10] S. Metson, D. Bonacorsi, M. Dias Ferreira, R. Egeland, Journal of Physics: Conference Series **219**, 072044 (2010)