Featured "Single Sign-In" interface enabling Grid, Cloud and local resources for HEP

Max Fischer[1,2] **, Günter Quast**[1] **and Marian Zvada**[2]

[1] *Institut für Experimentelle Kernphysik of the Karlsruher Institut für Technologie*,
Building 30.23, Wolfgang-Gaede-Str. 1, 76131 Karlsruhe
[2] *Steinbuch Centre for Computing of the Karlsruher Institut für Technologie*,
Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen

E-mail: `max.fischer@kit.edu, gunter.quast@cern.ch, marian.zvada@kit.edu`

Abstract. The CMS collaboration is successfully using glideinWMS for managing grid resources within the WLCG project. The glidein mechanism with HTCondor underneath provides a clear separation of responsibilities between administrators operating the service and users utilizing computational resources. German CMS collaborators (dCMS) have explored modern capabilities of the glideinWMS aiming at merging national grid resources, institutional CPU power and cloud resources into the set of pools with common sign-in interface presented towards HEP analysis users. The key goals of service development include ease of use, uniform access, load balancing and automated selection among different resource technologies. The approach integrates experience of dCMS during the development and integration phases as well as production operations and highly encourages other countries to follow. First experience with the production system and an outlook towards ongoing development will be presented.

1. Introduction

With the CMS collaboration[1] fully adopting glideinWMS [2] in 2013, most of the LHC collaborations rely on a *pilot*-based usage model for WLCG resources. The virtual overlay clusters created by this homogenize the fragmented grid environments to increase usability and improve efficiency. For each collaboration, its pilot overlay globally performs scheduling and balancing of resources in line with the needs and goals of the collaboration. The added layer between resources and users helps to standardise interfaces to the underlying infrastructures.

In contrast to resources governed by collaborations on the international scope, those exclusive to national usage are highly fragmented. Sign-in mechanisms and credentials vary on a case-by-case basis. Users must manually synchronize data and programs between sites. Individual resources also exhibit highly diverse architectures and environments.

To cope with this, the German CMS community (dCMS) has decided to launch a pilot service on a national scale. With the service unifying the dCMS resources, users are presented with a single sign-in interface to their common grid, cloud and local resources. This paper briefly describes the current setup and scope of this service, the basics of user access as well as the planned and ongoing expansion.

2. glideinWMS for dCMS

glideinWMS is a fully featured pilot overlay service based on the HTCondor [3] batch system and its own pilot implementation called glidein. It creates a dynamically sized computing pool by on-demand deployment of worker node software, the glideins, on grid resources. The CMS collaboration aims at replacing its previous grid access models entirely with glideinWMS by the end of 2013[4]. This has encouraged a joined development of infrastructure service, user tools and workflows.

Motivated by this, dCMS has adopted glideinWMS for its pilot service as well. In the past year, an instance has been deployed at the Karlsruhe Institute of Technology (KIT) serving as both proof of concept and foundation for a lasting service. The resource backend is formed by the grid share dedicated to German users by the grid sites affiliated to the dCMS members DESY Hamburg, KIT and RWTH Aachen. While dCMS pilots identify themselves with a dedicated DN and role, this has been found not to be required for the core functionality.

2.1. Setup

The glideinWMS service consists of four components (see figure 1): Users submit their jobs and receive output on the *User Portal*. The *Collector* schedules jobs to available resources and is the centre of internal service communication. The *Frontend* monitors the service load relative to service capacity. If needed, it orders the *Factory* to produce more workers by deployment of glideins to resources.

The four components of the service are deployed on three dedicated blade servers integrated in the KIT infrastructure. *User Portal* and *Factory* requirements both scale with the service load; as such, each is hosted on a physical machine. In contrast, the *Collector* and *Frontend* are static and undemanding in their needs; each is deployed on a virtual machine, sharing the same physical machine.

The User Portal does not necessarily have to be part of the same administrative domain, nor is it required to be unique. A second, off-site portal is under consideration at DESY to allow for better user support via local administrators and customized access methods.

2.2. User Authentication

Pilot overlay clusters shift the burden of user authentication from sites to the service. The dCMS service handles authentication via the Portals the users sign in to. In turn, this enables users to access the entire dCMS pool with a single sign-in (see figure 2).

By default, users login via SSH to an institute portal using their institute credentials. In the future, this will be expanded to logins via certificate credentials. The underlying HTCondor framework user management can still be employed in addition. This allows e.g. enforcing usage policies or dealing with malicious users.

In a pilot framework jobs are by default anonymous to resources, presenting only the pilot identity. To ensure site authority over users, identity must be forwarded from the pilot to the host. This can simply be the user identity in the pilot pool, provided via a file containing the pilot state, or the result of a certificate based identification via gLExec [5]. The later allows simple logging of identity or switching job execution to an account associated with the user, similar to the gLite grid job execution.

2.3. User Interfaces

The service exposes the low-level interfaces of the underlying pool to users. These encompass the HTCondor CLI tools, *jdl* job descriptions and HTCondor *ClassAd* queries, providing users with access to the full functionality of a regular HTCondor pool. In fact, the glidein pool is indistinguishable from a standard pool to users.

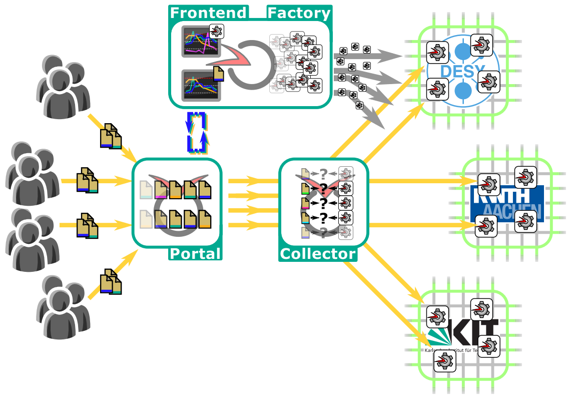

Figure 1. Current setup of the dCMS glideinWMS service: The service consists of the four nodes hosted at KIT - the User Portal, the Collector, the Frontend and Factory. The resource backend is formed by glideins using the dCMS grid share of the three dCMS grid sites at DESY, KIT and RWTH Aachen.

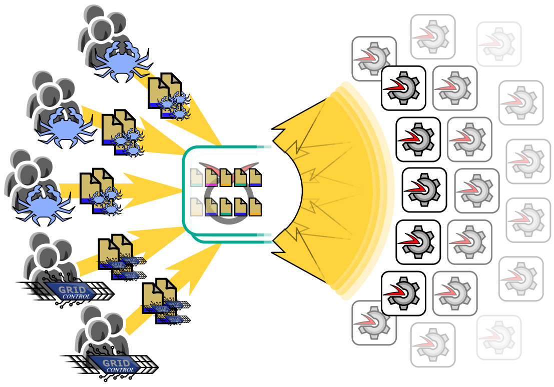

Figure 2. Appearance of the dCMS service to users: The service presents itself as a single HTCondor pool to users. With the sign-in to the User Portal, all acquired resources are directly available. The use of job managers allows users access with familiar tools, making existing grid workflows instantly portable.

However, users are highly encouraged to make use of *Job Managers* for submitting and controlling jobs. These tools feature their own configuration and control languages and transform them to instructions for targeted WMS. This removes the need for users to learn new interfaces for the service and highly simplifies enforcing job restrictions and meta-data conventions. Specifically, the CMS Remote Analysis Builder (CRAB)[6] and the generic job manager Grid Control [7] allow users to instantly switch existing grid workflows to the new service.

3. Service Development Outlook

The existing service has proven the feasibility of operating a glideinWMS service with the limited administration manpower available on a national scale. For the future, this forms the basis to unify the remaining dCMS resources. In addition to providing raw processing power, users also expect a national service to capture, rival or even extend the advantages of existing national computing environments.

3.1. Expansion

In order to unify the dCMS computing environment, the service is required to incorporate a variety of different resource types. Aside from the technical and administrative challenge, each type must be properly integrated into scheduling and provisioning for efficient balancing and usage.

Cloud resources are not part of dCMS resources due to administration overhead of managing and accounting for virtual machines. Yet, it would be highly attractive if properly automatized in a glidein pool. In fact the glideinWMS framework recently introduced support for cloud resources, deploying virtual machines as worker nodes. This is interesting for resource balancing, combining institutional and commercial clouds to dynamically cope with usage-peaks without long-term investments. In addition, users can be provided with access to custom computing setups via the standard submission interface. For example, this facilitates legacy provisioning of deprecated software.

The core of the dCMS computing infrastructure are batch systems affiliated to the institutes.

Most prominently, the National Analysis Facility at DESY provides hundreds of batch slots. The HTCondor based framework BOSCO [8] has proven the feasibility of injecting glideins into production batch systems; implementing a similar functionality directly into glideinWMS is a long-term plan for the dCMS service development.

While many of the batch resources are not operated by the dCMS community, a number is under direct, if shared, administration. It is possible to integrate these resources by converting them to regular, static HTCondor worker nodes. This allows acquiring them with reduced overhead and better control than via glideins, establishing a reliable base capacity. A small portion of institute infrastructure has already been converted to HTCondor nodes and linked to the dCMS service. So far this has been found to seamlessly integrate with the glidein pool with no additional work required compared to the previous batch system administration.

3.2. Evolution

Regarding the desired primary use-case of grid-resources there are two possible, opposed directions of evolution. The obvious path is the integration of dCMS grid pilots into the Central CMS pilot service. This is in line with the initial purpose of the dCMS grid share as an added allowance, not a separate service. The alternative is prioritizing the grid share to process low I/O, low dependency jobs, keeping the the high-performance resources clear. This requires pilot extensions providing interfaces mimicking the primary job resource environment, so that the system has maximum flexibility for resource balancing without explicit adjustments by user.

Introducing standardized information and configurations via pilots is attractive in general. Pilots can provide up-to-date information for Job Managers to optimise resource usage by jobs. Tools and utilities can be wrapped to introduce additional functionality such as caching or to quickly remedy middleware bugs. General utilities can be provided, e.g. offering mounts to access off-site storage for site-independent job execution.

Having access to the volume and versatility of dCMS resources opens the possibility for advanced scheduling. Nodes may be allocated dynamically, offering a number of usage patterns, e.g. for I/O or multi-core usage with the final decision implicitly performed by the matchmaker based on queued jobs. Moreover, non-blocking priority schemes for multi-core nodes using overbooking with different priorities is attractive for handling usage peaks without disrupting the base operations. Generally, per-node balancing schemes combining high-performance and long-running jobs for best utilisation will be a focus to achieve maximum gain with limited resources.

3.3. Monitoring

Keeping track of resource capacity and user demands is integral to the workings of glideinWMS. This suggests the possibility to use the service to easily derive a range of monitoring information.

The glideins automatically map all used resources: availability, utilisation and service quality can be assessed directly. This allows the monitoring of the dCMS resources almost in real-time, allowing for the timely detection of problems and potential bottlenecks. The HappyFace meta-monitoring framework[9] offers the potential to directly integrate this into standard dCMS monitoring operations.

On the national level, the close affiliation between users and service providers lends itself to user focused monitoring. Key figures such as CPU usage, I/O distribution, job performance etc. can be tracked per user for a wide array of resources and thus workflows. Aside from identifying misuse, this information supports users in optimizing their tasks for the dCMS computing environment. In addition, meta information validation will be a key focus, assessing the quality of user provided information such as expected walltime. If proven feasible, matchmaking based on interpreted meta-data should provide optimized scheduling conditions.

4. Summary and Conclusion

The dCMS glideinWMS service is well established since the beginning of operation less than a year ago. The service runs stable and with full functionality for the initial phase. Much administrative experience both for basic operation as well as future approaches has been gained. Most importantly, the feasibility of operating such a service with minimal manpower has been proven.

With this basis firmly established, there are several paths for the evolution of the service. First and foremost, integration of new resource types will enable the complete unification of dCMS resources into one service. Additional features and extensions will help increase usability and capabilities. Advanced monitoring solutions will provide the means to optimize both resource quality and user workflows.

In the long term, the scope of this project may provide solutions that will help to make dedicated grid resources more attractive compared to national resources. Even though our work is far from finished, we hope the example for unifying national resources within a single service will inspire other communities.

Acknowledgments

The authors would like to thank the teams and communities of the HTCondor and glideinWMS projects for their exceptional work and direct as well as indirect support.

References

[1] CMS Collaboration 2008 The CMS experiment at the CERN LHC *JINST* **3** S08004 (doi: 10.1088/1748-0221/3/08/S08004)

[2] Sfiligoi I, Bradley D C, Holzman B, Mhashilkar P, Padhi S and Wurthwein F 2009 The Pilot Way to Grid Resources Using glideinWMS *2009 WRI World Congress on Computer Science and Information Engineering* **2** 428–32 (doi:10.1109/CSIE.2009.950)

[3] Thain D, Tannenbaum T and Livny M 2005 Distributed computing in practice: the Condor experience *Concurrency Computat.: Pract. Exper.* **17** 323–56 (doi: 10.1002/cpe.938)

[4] Bradley D, Gutsche O, Hahn K, Holzman B, Padhi S, Pi H, Spiga D, Sfiligoi I, Vaandering E, Würthwein F and the Cms Offline and Computing Projects 2010 Use of glide-ins in CMS for production and analysis *J. Phys.: Conf. Series* **219** 072013 (doi:10.1088/1742-6596/219/7/072013)

[5] Groep D, Koeroo O, Venekamp G 2009 gLExec: gluing grid computing to the Unix world *J. Phys.: Conf. Series* **119** 062032

[6] Cinquilli M, Fanfani A, Fanzago F, Farina F, Kavka C, Lacaprara S, Miccio V, Spiga D and Vaandering E 2009 CRAB: a CMS application for distributed Analysis *Nuclear Science, IEEE Transactions on Volume 56, Issue 5, Part 2, Oct. 2009* 2850 - 8

[7] The Grid-Control project homepage URL https://ekptrac.physik.uni-karlsruhe.de/trac/grid-control/

[8] Weitzel D, Fraser D, Bockelman B and Swanson D 2012 Campus grids: bringing additional computational resources to HEP researchers *J. Phys.: Conf. Series* **396** 032116 (doi:10.1088/1742-6596/396/3/032116)

[9] Mauch V, Ay C, Birkholz S, Büge V, Burgmeier A, Meyer J, Nowak F, Quadt A, Quast G, Sauerland P, Scheurer A, Schleper P, Stadie H, Tsigenov O and Zvada M 2011 The HappyFace Project *J. Phys.: Conf. Series* **331** 082011 (doi:10.1088/1742-6596/331/8/082011)