PROCEEDINGS OF SCIENCE

"Single Sign-In" User Centered Computing (SUCC) for HEP

Marian Zvada*
Karlsruhe Institute of Technology
E-mail: marian.zvada@cern.ch

Max Fischer
Karlsruhe Institute of Technology
E-mail: max.fischer@cern.ch

Oliver Oberst
E-mail: Oliver.Oberst@cern.ch

Günter Quast
Karlsruhe Institute of Technology
E-mail: Gunter.Quast@cern.ch

Aiming at simplifying the grid usage for German CMS collaborators (dCMS), a GlideInWMS service is currently being set up at KIT-EKP. Current user-focused development is geared towards easy access and the goal is that users can submit, check and control jobs in the GlideIn service from any authorized device, be it a work station, a development resource or a mobile device, without requiring any permanent infrastructure. The GlideIn mechanism with HTCondor underneath provides clear separation of responsibilities to admins who operate the service and users who utilize computing resources encapsulated in the grid, cloud or within an institute network. A major part is the extension of the job management tool, Grid-Control, which is a very popular interface to dCMS for physics analysis, serving as a plugin to the GlideInWMS resources. We introduce the system where literally within seconds, users can switch existing tasks to the new centralized glideIn back-end. With minimal effort setup your submission end-point and SUCC let you use dCMS computing on demand.

The International Symposium on Grids and Clouds (ISGC) 2013
17 - 22 March 2013
Academia Sinica, Taipei, Taiwan

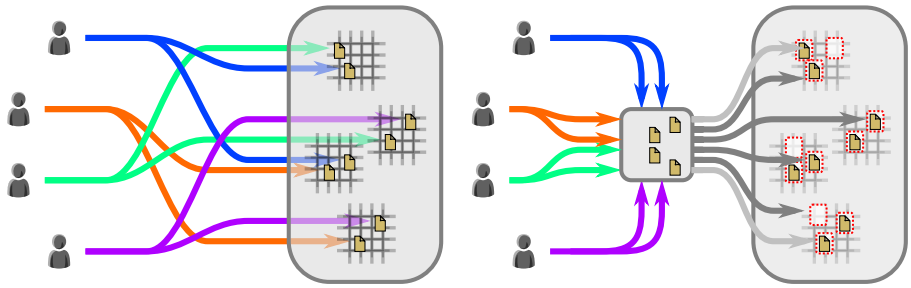
*Speaker.

PoS(ISGC 2013)016

1. Introduction

Modern day scientific work is increasingly reliant on large scale computing capabilities. Individual groups from different scientific disciplines use resources ranging from a few hundred up to several thousand CPUs. On an international scale, providing resources shared between several collaborations via the Grid computing model has proven very successful. However, the Grid is a complex environment and using it efficiently can be a challenging task to individual users.

To simplify and streamline the usage of Grid resources, scientific collaborations have begun employing the pilot paradigm: a service sends meta-jobs, so called *pilots*, to acquire computing resources and then forwards user jobs to them for execution. Thus a virtual cluster is created on top of existing resources, with the pilots acting as computing nodes of the cluster. This improves the load balancing between individual resources, simplifies the computing environment and provides a more robust computing environment.



(a) Direct Submission (b) Pilot Submission

Figure 1: Direct submission and pilot overlay working principle: Classically, user jobs are sent directly to resources. With a pilot service, jobs are sent to an entry node. The service then executes the jobs on resources it acquired using pilots.

The CMS collaboration is using GlideInWMS [1], a pilot framework funded by Open Science Grid[2] and developed at FNAL, to consolidate and manage its Grid resources. GlideInWMS builds on the HTCondor[3] framework to provide users with a seemingly local computing pool. For the future, the service is planned to be the sole method for using Grid resources for CMS.

The German CMS community (dCMS) has decided to employ its own instance of GlideInWMS to likewise consolidate resources exclusively or preferentially available to German CMS members. Constructed at the Institute of Experimental nuclear Physics[4] (IEKP) of the Karlsruhe Institute of Technology[5] (KIT), the instance serves to prove the viability of operating GlideInWMS on a national level and to provide the development environment for integrating the diverse, heterogeneous resources available to dCMS.

This paper describes the general working principle of GlideInWMS, the setup of the dCMS GlideInWMS at the IEKP and its extension for the entire dCMS community, future development plans for supporting cloud resources and local clusters as well as monitoring for both administrators and users.

2. Motivation

In addition to international resources available to an entire collaboration it is common for subgroups to have access to dedicated national and institutional resources based on their affiliation. This introduces a high level of complexity to the pool of resources available to each individual collaboration member. The distribution over different administrative domains and implementations results in major differences in configuration and access management. In effect, scientists have to manually deal with a highly fractured and heterogeneous computing environment. Subsequently, taking full advantage of their available resources is unattractive for scientists. With each user individually selecting resources, the possibilities for an overarching resource balancing by the collaboration are limited.

For example, the members of the KIT CMS group have access to a total of seven resources consisting of a dedicated share for German CMS members on the three Grid sites at DESY, KIT and RWTH Aachen, the National Analysis Facility[6] at DESY, two institute clusters and a local cloud service hosted at KIT. This is in addition to the international resources of the collaboration[7]. Future development plans include a further increase in the number of services as well as changes or outright replacements of existing services. The situation at other German CMS institutes is comparable.

Based on this situation, a consolidation of resources appears to be the most prudent solution. With the different affiliations, administrative concerns and funding of sites, both a physical merge as well as extensive changes to the setup of sites are out of the question. As such, the minimal invasive nature of pilot based virtual clusters is a promising approach. GlideInWMS is already successfully in use by the CMS collaboration for managing the project's WLCG resources. As such, utilizing it as the basis for the development of the dCMS pilot service brings the added benefit of offering a similar interface to both CMS and dCMS resource.

The key goals of the service development include:

- **Ease of use** to ensure widespread acceptance of the service.
- **Uniform Access** to all resources to allow spreading user jobs over the entire pool.
- **Load Balancing** between individual resources to improve resource usage.
- **Minimal dependencies** to require little work from resource administrators.

3. Architecture and Implementation

3.1 GlideInWMS Architecture

As the name implies, GlideInWMS is based on the usage of glideins, the native pilot job of the HTCondor High-Throughput Computing framework. In the following, „resource" or „computing resource" refers to the underlying infrastructure, i.e. Grid Sites, Cluster, Clouds etc, while 'glidein' refers to the pilot jobs (potentially) running on the infrastructure.

The basic structure of GlideInWMS is best understood by dividing it into two levels:

- **User Job Management** houses queues for user jobs and assigns jobs to existing glidein resources for execution.

- **glidein Production** monitors the load on the queues and produces new glideins to match demand.

A major advantage of GlideInWMS is the appearance to users as a local HTCondor pool. The User Job Management level is a regular HTCondor pool configured to make use of the specific setup of the glideins but no external tools. Access points for users are the User Pools where users can add their jobs to a queue. The centerpiece is the Collector service which matches jobs from the queues to available glideins according to their requirements and features. Actual execution of jobs is handled by glideins that have started on the available underlying resources. While the Collector is a unique service in every GlideInWMS instance, an arbitrary amount of User Pools can be in use. The maximum number of glideins is dictated by the capacity of the resources used.

The glidein Production level is an added layer on top of the user HTCondor pool. It automates and controls HTCondor's glidein production features. The Factory is configured with a list of available resources which also defines features of potential glideins for each resource; it has the capability to produce requested and track existing glideins. The VO Frontend monitors the number and features of jobs waiting in the User Pools. It queries the Factory on the number of existing glideins and what types of glideins are available for production. Based on this, it will compile the number and type of glideins needed and delegate production to the factory. glideins produced by the Factory will at first queue at their assigned resource like regular jobs. When being executed, they report back to the Factory for tracking and to the Collector for assignment of jobs. If a glidein doesn't receive a valid job for some time, it terminate itself automatically. The VO Frontend is unique per GlideInWMS instance but can make use of multiple Factories.

3.2 Implementation at IEKP

The IEKP GlideInWMS service is running on three dedicated blades of which one is running two virtual machines for a total of four hosts. User Pool and Factory are deployed on the baremetal blades while the less demanding Collector and Frontend use the virtual machines.

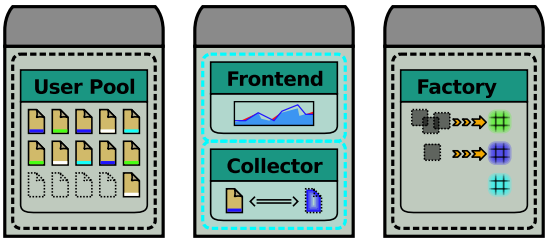


Figure 2: Hardware partitioning for GlideInWMS at KIT: the service nodes are partitioned among the three identical blades to best match their demands.

Notable details of each component include:

The **Factory** was configured to target the CEs of the dedicated dCMS Grid resources. Due to glideins being actual Grid jobs, basic operation was possible without requiring the targeted sites to make any modifications specific to our service.

The **Frontend** was provided with its own GSI[8] VOMS identity certificate. This is used in to generate proxies for glidein production delegation. The dCMS collaboration created the specific group role of *dCMS pilot* to allow identification and privileging of glideins by sites.

The **Collector** did not require any notable configuration. In fact, it was only specifically modified when debugging the matching and assignment of user jobs to glideins.

The **User Pool** was the main focus of the current initial service development. Configuration was refined to reduce the input required by users to an absolute minimum. Several access methods for users were investigated; the preferred method is now remote delegation via ssh[9]. The user database for this is imported from the LDAP service provided by IEKP.

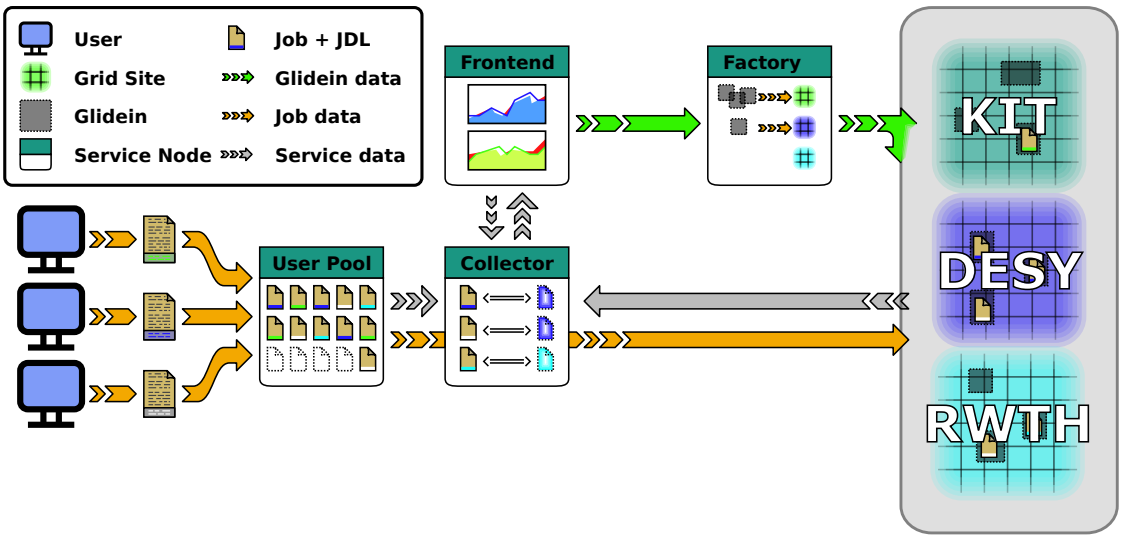


Figure 3: Basic setup of GlideInWMS at KIT: all service nodes are unique and glideins are sent only to dCMS grid sites.

3.3 Interfacing: Job Managers

GlideInWMS greatly simplifies the degree of detail end users have to deal with. However, it is still a complex system with many peculiarities. Manual interfacing with the system can be a challenging task for new users. To make the system as accessible as possible, users are implicitly expected to interface with the service via the existing job managers commonly in use by the collaboration.

Job managers are user controlled programs that allow defining jobs, assigning them to bulk resources, tracking their progress and controlling them. They are advanced wrappers for resource interfacing with their own configuration language. As such, users only have to know how to use the job manager in order to access all supported types of resources.

The official CMS analysis job manager CRAB[10] already supports the official CMS GlideInWMS instance. As the dCMS GlideInWMS instance is configured to mimic this setup, CRAB

supports it natively. An alternative Job manager developed at IEKP called Grid-Control[11] was extended by a highly generic HTCondor backend during the development of the dCMS GlideInWMS service. This allows Grid-Control users to easily target different HTCondor-based systems, including instances of GlideInWMS.

As such, users employing either job manager can switch their existing jobs and templates to the new system with ease. Modifying a single line of configuration is sufficient. Users are not constrained in their preferences for managing their jobs.

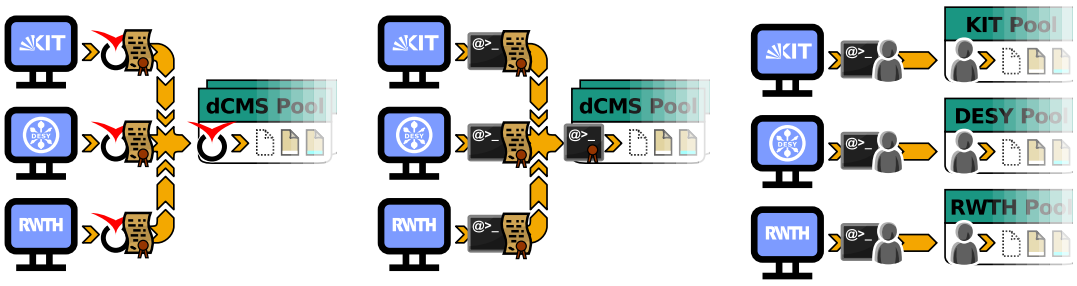
3.4 Scope: dCMS Service

Several options have been investigated on how to best expand the service from an institute specific one to a dCMS wide application.

HTCondor Delegation: With the user level of GlideInWMS only running with basic HTCondor components, it is possible to make use of the vast number of authorization and delegation methods already implemented in the framework. Specifically, HTCondor allows for sending jobs from one pool to another. Authorization is then handled internally on a user basis by means of their GSI certificate. This requires users to have only a minimal HTCondor package available[1]. The actual User Pool can then authorize users based solely on their dCMS affiliation; no user database is required.

Institute Pools: The most straightforward approach is cloning the existing IEKP User Pool at other institutes. Only the user database needs to adapted for the corresponding institute. This gives institutes maximum freedom in managing their users' privileges and authorization methods as well as enforcing usage rules.

Collaboration Pool: A single collaboration pool can be used for all users if access is properly ensured. Using information from the collaboration's VOMS system, a user database can be automatically created. Alternatively, a pool can dynamically map users identified by their certificate to generic pool accounts. The official CMS analysis GlideInWMS service for example generates a static database from VOMS information. Users then log in using GSI enabled ssh[12].



(a) HTCondor + X509 (b) GSISSH + X509 (c) SSH + Account

Figure 4: Multi-institute access scenarios: the various authentication scenarios require different user pool infrastructure.

[1] Only the bare executables for queue management are required. No constant installation or deployment of daemons is necessary.

However the pool implementations are limited by Job Managers in use. Grid-Control supports all access models on account of having been used for internal testing. The official CMS job manager CRAB currently only supports gsissh[2] based access.

The favored architecture is a hybrid of collaboration and institute pool setup. For redundancy and scalability, every institute operates a collaboration pool. This allows administrators to assign temporary accounts to users from their institute who do not own a certificate.

4. Future Plans

GlideInWMS for dCMS already successfully consolidates Grid resources, improving reliability, accessibility and efficiency. The project as a whole is far from being complete, though. Having proven its basic functionality, several extensions are planned for the service in the future. The range of different resource types will be extended step by step to allow consolidating as many different resources as possible. Centralization of resource allocation will be used to improve the capabilities for monitoring by both users and administrators.

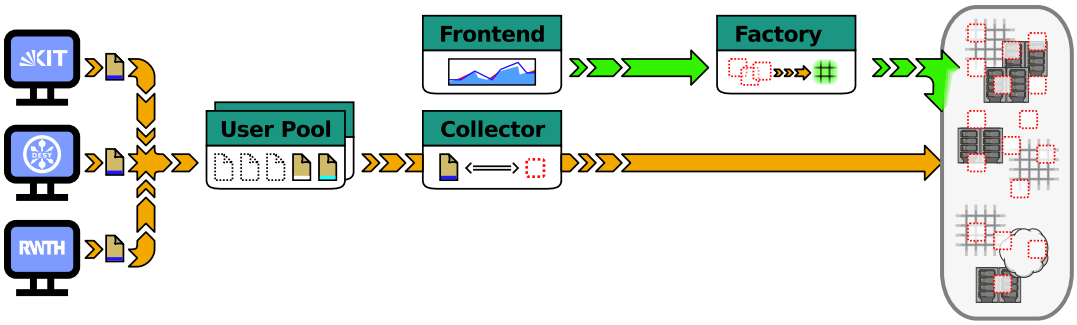


Figure 5: Planned dCMS GlideInWMS service: members from all dCMS institutes can submit to the service. Glideins combine all accessible dCMS grid, cloud and local resources into a single pool.

4.1 Cloud Resources

The idea of providing Grid and Cloud resources to end users, although based on technically different levels of abstraction, is fundamentally very similar. From a large pool of resources, a chunk is allocated to a specific user. Pre-partitioning and re-using of resource slots minimizes losses from initialization and waiting times for individual job scheduling in either system. While Cloud usage comes with a greater overhead for supplying resources, it can provide arbitrary software architectures and hardware setups.

At present, CMS is already expanding the GlideInWMS service for using Cloud type resources, using HTCondor's inbuilt Cloud submission service[13]. Instead of dispatching just a glidein to an existing computing resource, the glidein Factory requests an entirely new virtual machine in the Cloud on which a glidein is then automatically launched after successful startup. The

[2]As gsissh is an extension of openSSH, regular ssh based access is implicitly supported as well

freedom of simulating arbitrary machines in the Cloud is reflected by having a variety of possible glidein virtual host machines pre-configured. These are made to match the most probable use cases of users.

The dCMS GlideInWMS service will make use of the CMS GlideInWMS Cloud development. The first target for testing and assessment of virtual host requirements will be the Open Nebula Cloud hosted at Campus North of KIT.

4.2 Local Resources

Even though the current glidein service implementations in GlideInWMS make use of native HTCondor submission mechanics, the basic glidein architecture does not impose a constraint on the submission type. A glidein can be submitted to any computing resource that satisfies the basic requirements. Thus, regular batch systems are also potential targets for glidein production. Other workload management systems such as BOSCO[14] already demonstrated the compatibility of glideins with PBS, SGE and other batch systems.

In order to extend the dCMS GliedeinWMS service to local resources, the glidein production mechanism must be extended. Development of a custom modification for the Factory component is planned by the dCMS GlideInWMS team. Ensuring independence from local configuration is a high priority. Thus the goal is to enable submission of glideins using the same methods users currently employ for submitting jobs. Since the resource types of interest are already supported by Grid-Control modules, the development will build heavily on these proven components.

Primary focus will be on deploying glideins on SGE/OGE type resources. The computing cluster at KIT Campus South will serve as the environment for development and testing of SGE submission. The primary goal is to make use of the vast resources available at the NAF's SGE batch system.

4.3 Monitoring

Combining all resources into a single service also automatically aggregates information about the usage of said resources centrally. This presents the opportunity to extend the service by an advanced monitoring service for users.

As the functionality is not an integral part of the service, research has so far only been on the fundamental possibilities for future developments. Potential features include life monitoring for users focused on job status and efficiency. Basic management features such as canceling or multiplying jobs are planned as well. Being a proven part of the RedHat MRG package, the monitoring and management framework Cumin[15] is under investigation for implementing this functionality. Following the positive test of a smartphone app for the HappyFace[16] metamonitor, a similar system is under consideration for end users as well.

5. Conclusion and Perspective

The pilot mechanism is a promising method for pooling together diverse and fractured computing resources. Thanks to the GlideInWMS framework, creating a pilot service for the dCMS community has proven both realistic and worthwhile. With the positive experience from consolidating the German grid resources, the concept of including dedicated, national dCMS resources

into a single service has been successfully substantiated. Users interact with a simpler computing environment and administration can efficiently handle resource access, usage and balancing. Since its deployment in summer 2012, the service is now reliable, efficient and easy to use.

With the infrastructure in place and interfacing as well as user access being well developed, the service offers a solid foundation for future expansions. The plan for including Cloud and local batch resources can be progressively implemented with users automatically benefiting from the additional services. Cloud development can build on the expertise and experience of the CMS collaboration. The challenges of integrating the diverse local resources are well met by the year-long expertise of in-house development of innovative job submission solutions. In the future, the implicit aggregation of usage information in a single service can be used as the basis for advanced monitoring and administration mechanisms.

References

[1] http://www.uscms.org/SoftwareComputing/Grid/WMS/glideinWMS/doc.prd/index.html

[2] https://www.opensciencegrid.org

[3] http://research.cs.wisc.edu/htcondor/index.html

[4] http://www.ekp.kit.edu/english/index.php

[5] http://www.kit.edu/english/index.php

[6] http://naf.desy.de/

[7] http://wlcg.web.cern.ch/

[8] http://www.globus.org/security/overview.html

[9] https://htcondor-wiki.cs.wisc.edu/index.cgi/wiki?p=RemoteCondor

[10] Spiga, D. et al., *CRAB: A CMS Application for Distributed Analysis*, Published in: *Nuclear Science, IEEE Transactions on Volume 56, Issue 5, Part 2, Oct. 2009 pp:2850 - 2858*

[11] https://ekptrac.physik.uni-karlsruhe.de/trac/grid-control/wiki

[12] http://grid.ncsa.illinois.edu/ssh/

[13] http://research.cs.wisc.edu/htcondor/CondorWeek2012/presentations/tiradani-glideinwms.pdf

[14] http://bosco.opensciencegrid.org

[15] https://fedorahosted.org/grid/wiki/Cumin

[16] https://ekptrac.physik.uni-karlsruhe.de/trac/HappyFace