Use of glide-ins in CMS for production and analysis

D. Bradley[1], O. Gutsche[2], K. Hahn[3], B. Holzman[2], S. Padhi[4,†], H. Pi[4], D. Spiga[5], I. Sfiligoi[2], E. Vaandering[2] and F. Würthwein[4]

[1] University of Wisconsin-Madison, Madison, WI, USA
[2] Fermilab, Batavia, IL, USA
[3] Massachusetts Institute of Technology, Cambridge, MA, USA
[4] University of California, San Diego, La Jolla, CA, USA
[5] CERN, CH-1211 Geneva, Switzerland

(On behalf of the CMS Offline and Computing Projects)

Abstract. With the evolution of various grid federations, the Condor glide-ins represent a key feature in providing a homogeneous pool of resources using late-binding technology. The CMS collaboration uses the glide-in based Workload Management System, glideinWMS, for production (ProdAgent) and distributed analysis (CRAB) of the data. The Condor glide-in daemons traverse to the worker nodes, submitted via Condor-G. Once activated, they preserve the Master-Worker relationships, with the worker first validating the execution environment on the worker node before pulling the jobs sequentially until the expiry of their lifetimes. The combination of late-binding and validation significantly reduces the overall failure rate visible to CMS physicists. We discuss the extensive use of the glideinWMS since the computing challenge, CCRC-08, in order to prepare for the forthcoming LHC data-taking period. The key features essential to the success of large-scale production and analysis on CMS resources across major grid federations, including EGEE, OSG and NorduGrid are outlined. Use of glide-ins via the CRAB server mechanism and ProdAgent, as well as first hand experience of using the next generation CREAM computing element within the CMS framework is discussed.

1. Introduction

The CMS collaboration has adopted Grid computing as its base computing model to simplify the deployment and management of the tens of thousand of CPUs needed to accomplish its mission. While the Grid computing paradigm has been shown to be a boon for resource providers, allowing them to keep their administrative autonomy over the resources they manage, the added abstraction layer has introduced several problems for the users, ranging from higher complexity to decreased reliability.

One solution that has proven to significantly reduce user problems is the late-binding, or pilot technology. A late-binding Workload Management System (WMS) hides the complexity of the Grid environment by dynamically creating a virtual private pool of compute resources, thus giving users an environment similar to a dedicated batch cluster. This paper describes the experience of the CMS collaboration with one late-binding WMS implementation, called glideinWMS.

†Corresponding Author, email: sanjay.padhi@cern.ch

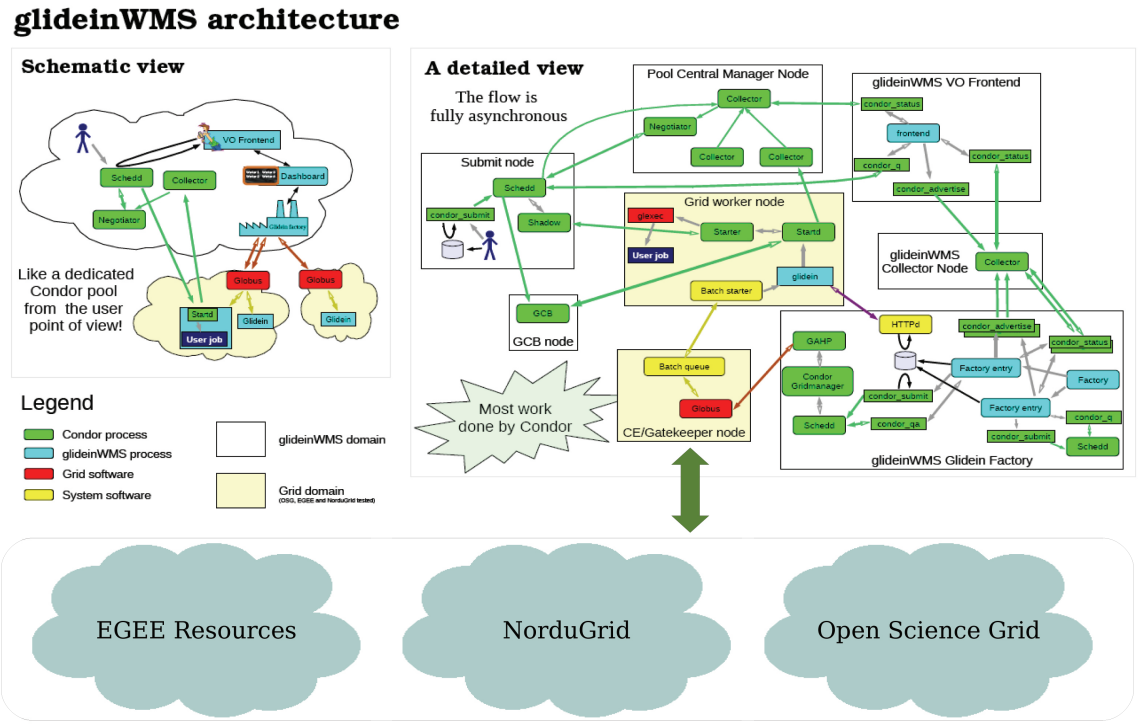


Figure 1. Overview of glideinWMS system.

2. Late-binding based Workload Management System - GlideinWMS

The Grid paradigm calls for the compute resources to be partitioned into multiple independent pools, called Grid sites, with only a thin common layer to provide interoperability. Without some additional tools, this approach makes the life of a Grid user quite difficult.

The four major problems users experience are:

- A user must partition their jobs between the resource pools. Finding the optimal partition is far from an easy task, as explained below.
- At any Grid site, the common layer provides only very limited information about the status and policies of the batch system that handles the local resource pool. This is a necessary evil that allows the common layer to present the information from all the different batch system implementations in a uniform way.
- The common layer provides very limited information about the progress of a job, once it is accepted at a Grid site. Again, this is a necessary evil that allows the common layer to monitor jobs submitted to different batch systems.
- Each Grid site is allowed to configure the worker nodes arbitrarily, within very permissive limits. Users are expected to develop their jobs in such a way to automatically adapt to any condition they encounter.

The approach taken by the late-binding Workload Management Systems (WMS) to ease the user burden is to create a dynamic virtual private pool of compute resources by submitting pilot jobs to the Grid sites. Once a pilot job starts, it joins the virtual private pool and starts a user job from the late-binding WMS job queue, as shown in Fig. 1.

This insulates the users from the Grid layer, giving users the impression of running on a single, local, dedicated pool of compute resources and thus eliminating, or at least reducing the

magnitude of the above mentioned major problems:

- By having a single resource pool, no job partitioning is needed.
- Detailed information about the status and policies of the virtual private pool can be made available, since users can use the tools provided by the specific implementation of the late-binding WMS instance used.
- Detailed information about the progress of a job can be made available, since users can use the tools provided by the specific implementation of the late-binding WMS instance used.
- The late-binding WMS can reduce the heterogeneity of the compute resources, by either only gathering properly configured worker nodes, or by providing wrapper scripts that provide common tools and libraries.

The late-binding WMS used by CMS is called **glideinWMS**. GlideinWMS is based on the Condor batch system, with the addition of a thin layer responsible for the submission of the pilot jobs.

A glideinWMS virtual private pool consists of a regular Condor pool, where worker node Condor daemons, i.e. *condor_startd* and *condor_starter*, have been downloaded, configured and started by a glideinWMS pilot job; such pilot jobs are known as *glide-ins*. Since the different Condor daemons are dispersed around the world and use wide area networking to communicate to each other, the daemons are configured to use strong, X.509 based authentication for authorization and message integrity purposes. The worker node Condor daemons are also configured to have a limited lifetime in order to fit within the wall-clock limits of the batch system of the Grid site they are running on. For all other practical purposes, the resulting Condor pool is indistinguishable from a dedicated Condor pool without a shared file system.

The submission of glide-ins is regulated by two types of daemon processes; one or more **glide-in factories** and one or more **VO frontends**. The two types of processes communicate by means of Condor ClassAds using a dedicated *condor_collector* daemon:

(i) A glide-in factory advertises what Grid sites it knows about, and can submit to, together with the characteristics of the site (for example: what versions of CMS software are installed there).

(ii) The VO frontend queries the user queues, i.e. the *condor_schedds*, matches the found jobs with the factory ClassAds, and then advertises how fast the factory should submit new glide-ins.

(iii) Finally, the factory reads the VO frontend ClassAds and starts submitting the glide-ins at the specified rate.

Together these make the system work as illustrated in Fig. 1.

2.1. Interoperability between EGEE, OSG, and NorduGrid

In principle, the problem of interoperability between different grids is reduced to having a Condor-G client for submission of the glide-ins for the particular grid flavor. All other incompatibilities can be resolved via appropriate configuration of the glide-ins. In practice, CMS deals with many of the differences between EGEE and OSG inside the CRAB or ProdAgent layers of the software stack, thus requiring little special configuration in the glide-ins. As part of our work for the Common Computing Readiness Challenge 08 (CCRC-08), we worked with the Condor team on the details of submitting glide-ins via Condor-G to grids such as the NorduGrid. The method consists of using protocol-specific modules called GAHPs (Grid Ascii Helper Programs) along with logic in the Condor gridmanager for the specific use of it, when submitting to a given grid flavor. This resulted in the first ever use of NorduGrid resources for data analysis within CRAB using glideinWMS. The other grid flavors like EGEE and OSG CEs

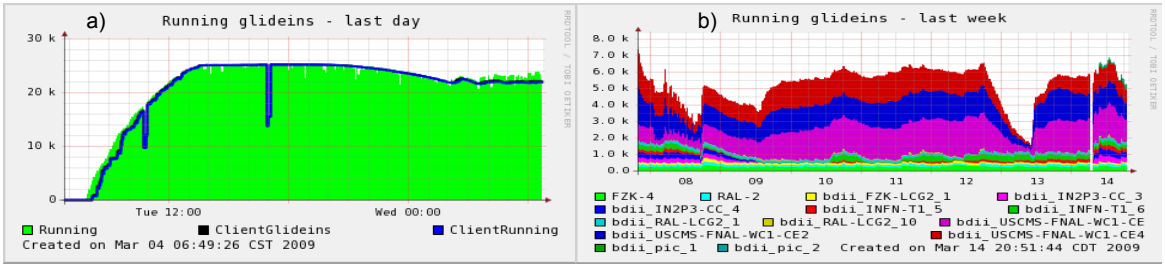


Figure 2. Number of simultaneous running glide-ins for: a) scalability studies and b) CMS data re-processing at Tier-1 centers.

are accessible using GRAM2 protocols via the Condor-G client. This development led to a fully interoperable system across all three grid flavors for the first time.

2.2. Scalability of the system

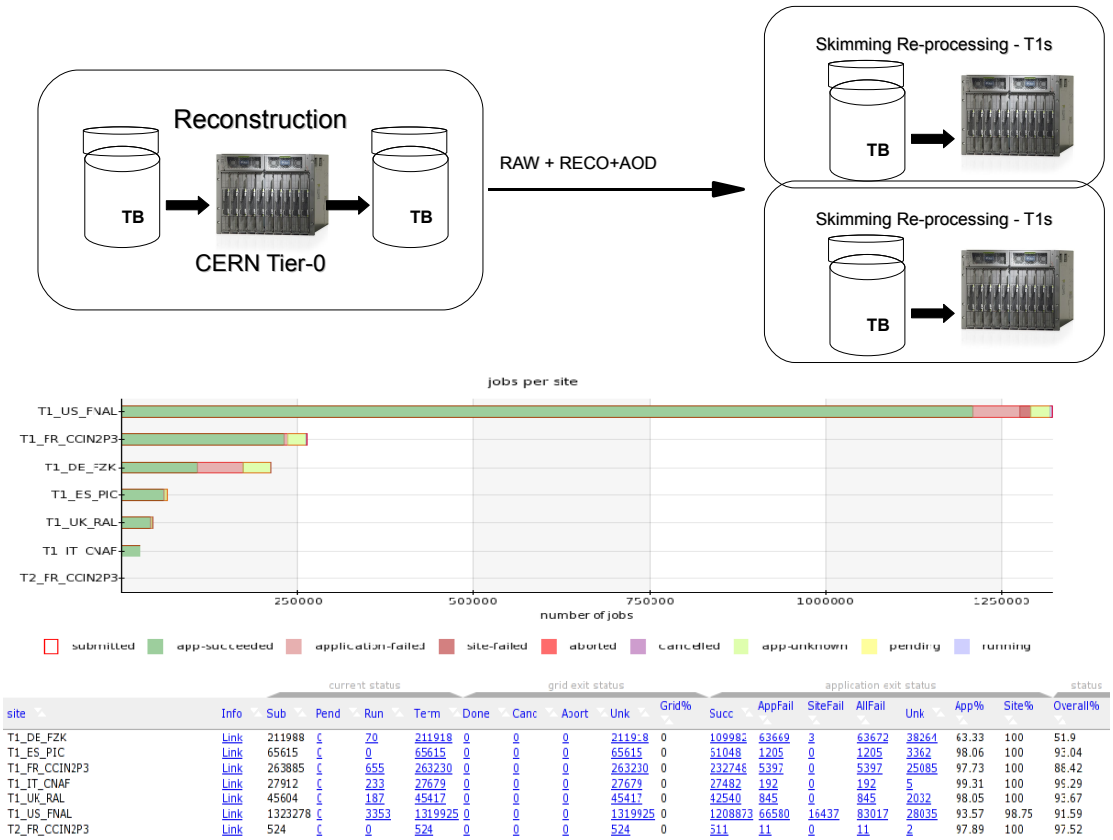
The scalability of the system has been intensively tested over the WAN, especially over large distances such as between US and Europe. Various components were identified which could have an effect on latencies in communications between the glide-in *startds* and the central managers. Many improvements were made both in reducing the cost of authentication, as well as enhancing the speed in matchmaking and the security sessions [1]. The overall improvement from this study led to simultaneous running between 23,000 to 25,000 jobs, as shown in Fig. 2a). More than 500,000 jobs were submitted with an average running period of 3 hours. In excess of 200,000 jobs were queued in addition to the 20,000 running jobs. Successfully harnessing more than 25,000 computing slots gives us confidence that using the system for collaboration-wide data analysis and Monte Carlo productions is feasible.

3. Use of glideinWMS for Production and Data Reprocessing

The CMS computing architecture [2] is based on a tier-organized structure of computing resources, based on a Tier-0 center at CERN and 7 Tier-1 centers for organized mass data processing. The Tier-0 is in charge of storing the data coming from the detector onto mass storage, performing a prompt reconstruction of the data and distributing the data among the Tier-1 centers. The Tier-1 sites archive on mass storage their share of data, run data reprocessing and centrally organized group physics analyses for data selection, and distribute the selected data to Tier-2s for user analysis. Tier-1 centers also have the responsibility of storing Monte Carlo (MC) data produced at the Tier-2 sites.

The workflows and dataflows are conducted using glideinWMS as well as CMS-specific services built on top of them. Data transfers are managed by the CMS data transfer and placement system PhEDEx [3]. Data skimming is conducted at the Tier-1 sites. Skim jobs apply a number of filters which produce the corresponding output files with the selected events. ProdAgent is used to carry out the skimming workflow. It automatically prepares the skim jobs for the sample to be filtered, submits to the glideinWMS and finally launches the corresponding merge jobs. More than 1.9 million jobs were successfully processed at various Tier-1s, with a remarkable efficiency of approximately 96% within last 3 months, using glideinWMS as shown in Fig. 3.

Fig. 2b) shows the number of simultaneous glide-ins running at various Tier-1 centers during the last week of March 2009. More than 1400 million events as shown in Fig. 4b) were reprocessed and merged in last 80 days, which are then used for physics analyses at the Tier-2 centers.



site	Info	Sub	Pend	Run	Term	Done	Canc	Abort	Unk	Grid%	Succ	AppFail	SiteFail	AllFail	Unk	App%	Site%	Overall%
		current status				grid exit status					application exit status							status
T1_DE_FZK	Link	211988	0	70	211918	0	0	0	2119:8	0	109982	63669	3	63672	38264	63.33	100	51.9
T1_ES_PIC	Link	65615	0	0	65615	0	0	0	65615	0	51048	1205	0	1205	3362	98.06	100	93.04
T1_FR_CCIN2P3	Link	263885	0	655	263230	0	0	0	263230	0	232746	5397	0	5397	25085	97.73	100	88.42
T1_IT_CNAF	Link	27912	0	233	27679	0	0	0	27679	0	27482	192	0	192	5	99.31	100	99.29
T1_UK_RAL	Link	45604	0	187	45417	0	0	0	45417	0	42540	845	0	845	2032	98.05	100	93.67
T1_US_FNAL	Link	1323278	0	3353	1319925	0	0	0	1319925	0	1208873	66580	16437	83017	28035	93.57	98.75	91.59
T2_FR_CCIN2P3	Link	524	0	0	524	0	0	0	524	0	511	11	0	11	2	97.89	100	97.52

Figure 3. Overview of CMS data re-processing and resource usage statistics using glideinWMS.

One of the essential features of the glideinWMS is the capability to utilize a large number of resources, with low startup latency. As discussed earlier, most of the reprocessing task workflows involve numerous short jobs. Previous studies have shown that a given CE can schedule jobs at a rate of 0.5Hz. In order to fill 6,000 resources at this rate, it would normally take about 3.33 hours. The glide-in approach, however allows workers (*startds*) to process multiple client jobs sequentially with more than an order of magnitude increase in rate, until the expiration of the glide-in's lifetime [4]. This provides a scalable solution for Tier-1 centers such as Fermilab, where a large number of resources are required to be used efficiently, which otherwise would not be possible to harness effectively using traditional early-binding approach.

4. Use of glideinWMS for data analysis

Data analysis in a distributed environment is a complex computing task. The CMS Collaboration uses CRAB [5], in order to perform user analyses at the Tier-2 centers. CRAB is a CMS-specific software layer layer integrated with a number of WMSes, including glideinWMS. It simplifies the process of data analysis, job submission and retrieval by hiding much of the grid complexity from the end user. The tool splits the analysis task into several jobs based on the requested number of events and the location of the input dataset via the CMS-specific Dataset Bookkeeping System (DBS). Furthermore, it performs various checks along with packaging of the executables/libraries before submitting the jobs to the glideinWMS. Periodic updates are performed regarding job status via glideinWMS to CRAB. Once the job is completed, the WMS provides the log files

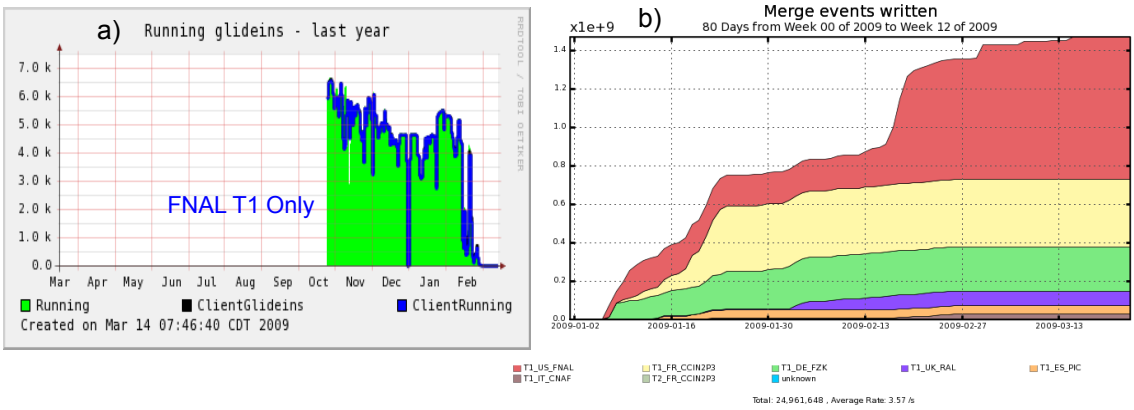


Figure 4. a) Number of simultaneous running glide-ins at Fermilab during Oct.08 - Feb.09. b) Distribution of events merged as a function of time, written at various CMS Tier-1 centers.

(stdout/stderr). The overall process also updates the CMS Dashboard using MonALISA as the transport agent for user job monitoring. In addition. glideinWMS provides a mechanism for on-demand "read" access to the job environment. This allows pseudo-realtime debugging of a running job via commands like ps, top, ls, and tail of various files, including stdout and stderr. At present, CRAB does not yet make use of these features of glideinWMS.

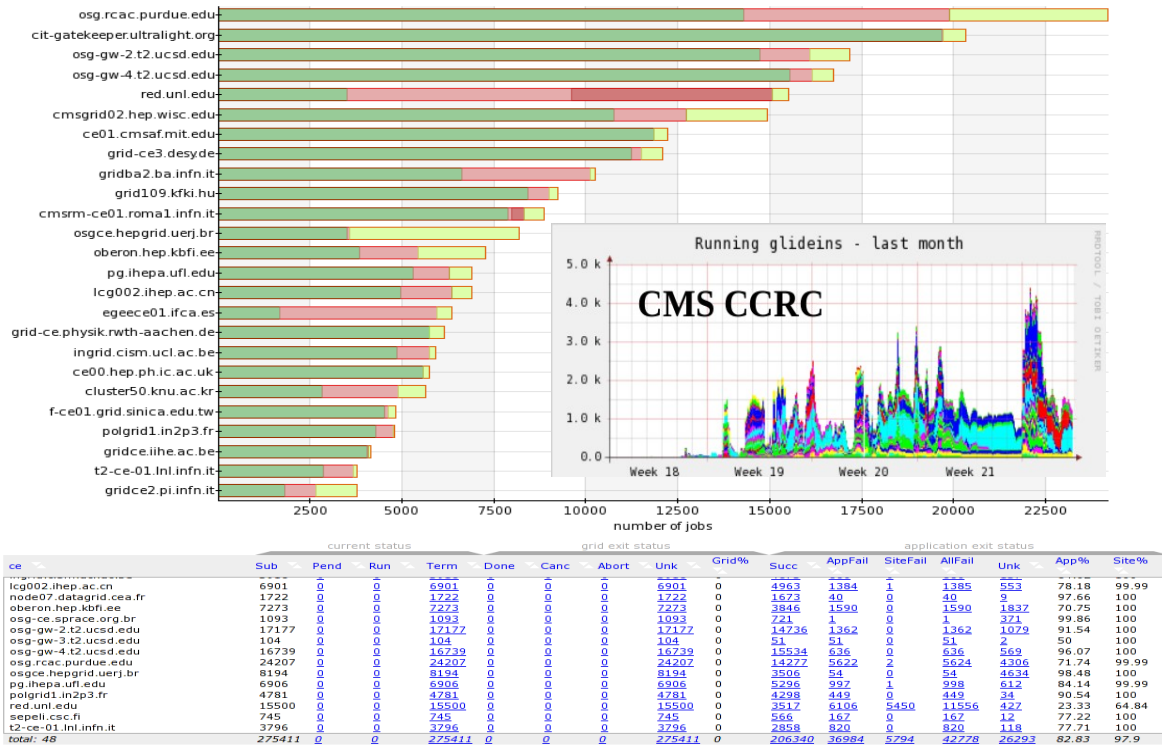
4.1. CMS User Analysis and CCRC-08

The glideinWMS was intensively used during the CCRC-08 computing challenge. The goal was to gain an overall understanding of the performance and readiness of the CMS Tier-2 sites for data analysis. Centrally organized workflows were used for this activity. The site performance was evaluated using different types of jobs with increasing complexity. The long-running CPU intensive jobs with moderate I/O were aimed at understanding the site performance without heavy loads on the storage systems. The short-running jobs with local stage-out were targeted to evaluate the performance of the site storage elements (SE). Intensive I/O with remote stage-out jobs were used not only to mimic the scenario of a real user performing data analyses, but also to understand the site access to the back-end storage.

Over 40 sites across EGEE, OSG and Nordugrid were involved during this exercise. Fig. 5 shows overall results recorded in the CMS dashboard during the month-long exercise. There were a mix of errors at a few sites due to catastrophic storage failures. The overall success rate without SE issues ranged from 92-99%, based on more than 200,000 submitted jobs using glideinWMS. The gliteWMS was also used during this exercise [6].

4.2. Studies using CRABserver and JobRobots

CRABserver is a novel approach towards decoupling the user activities with the stack of software layers responsible for performing the actual workload. The user submission is a simple layer of job submission and retrieval using the CRAB client. On the other hand, the CRABserver handles much of the complex work in a central location. User jobs are submitted using information from the DBS, as well as their credentials. These are sent to the server using gridftp. CRABserver consists of several daemons that are responsible for job tracking, task tracking, managing task lifetimes, etc. These processes essentially ensure all the (re)submission/retrieval activities of a given task are correctly communicated to the WMS, as illustrated in Fig. 6. The glideinWMS



	current status				grid exit status					application exit status						
ce	Sub	Pend	Run	Term	Done	Canc	Abort	Unk	Grid%	Succ	AppFail	SiteFail	AllFail	Unk	App%	Site%
lcg002.ihep.ac.cn	6901	0	0	6901	0	0	0	6901	0	4963	1384	1	1385	553	78.18	99.99
node07.datagrid.cea.fr	1722	0	0	1722	0	0	0	1722	0	1673	40	0	40	9	97.66	100
oberon.hep.kbfi.ee	7273	0	0	7273	0	0	0	7273	0	3846	1590	0	1590	1837	70.75	100
osg-ce.sprace.org.br	1093	0	0	1093	0	0	0	1093	0	721	1	0	1	371	99.86	100
osg-gw-2.t2.ucsd.edu	17177	0	0	17177	0	0	0	17177	0	14736	1362	0	1362	1079	91.54	100
osg-gw-3.t2.ucsd.edu	104	0	0	104	0	0	0	104	0	51	51	0	51	2	50	100
osg-gw-4.t2.ucsd.edu	16739	0	0	16739	0	0	0	16739	0	15534	636	0	636	569	96.07	100
osg.rcac.purdue.edu	24207	0	0	24207	0	0	0	24207	0	14277	5622	2	5624	4306	71.74	99.99
osgce.hepgrid.uerj.br	8194	0	0	8194	0	0	0	8194	0	3506	54	0	54	4634	98.48	100
pg.ihepa.ufl.edu	6906	0	0	6906	0	0	0	6906	0	5296	997	1	998	612	84.14	99.99
polgrid1.in2p3.fr	4781	0	0	4781	0	0	0	4781	0	4298	449	0	449	34	90.54	100
red.unl.edu	15500	0	0	15500	0	0	0	15500	0	3517	6106	5450	11556	427	23.33	64.84
sepeli.csc.fi	745	0	0	745	0	0	0	745	0	566	167	0	167	12	77.22	100
t2-ce-01.lnl.infn.it	3796	0	0	3796	0	0	0	3796	0	2858	820	0	820	118	77.71	100
total: 48	*275411*	*0*	*0*	*275411*	*0*	*0*	*0*	*275411*	*0*	*206340*	*36984*	*5794*	*42778*	*26293*	*82.83*	*97.9*

Figure 5. Resource usage via glideinWMS during CCRC-08 exercise.

then submits them to respective sites by preserving the user identity and privileges. The user credentials and priorities are maintained by the WMS using Condor.

The University of California, San Diego (UCSD) is one of the centers responsible for hosting the CRABserver using glideinWMS for CMS user analyses. UCSD maintains production and development CRABservers that are integrated with glideinWMS. The CRABserver was tested to a large scale using so-called "JobRobots". At regular time intervals, a new analysis task is created for each site to be run on a specific dataset. The task is split into several jobs that are submitted as a collection to the CRABserver, which eventually submits to glideinWMS. Each job performs a trivial data analysis on a fraction of the dataset, and when finished, its output is retrieved. All submitted jobs are classified as either successful, failed at the application level or aborted at the Grid level.

The JobRobot daily statistics are not only used to measure the success rate for each site, but also to evaluate the CRABserver performance. More than 25,000 jobs were submitted within a span of a few days using this technique. Several issues related to monitoring, submission/retrieval, delays, etc. were identified and fixed during this study. Various scalability related issues are currently under study and are expected to be fixed in the forthcoming release version of the server.

5. Coherent monitoring interface for the system

The glideinWMS consists of two sets of jobs: glide-ins and the user jobs. The glide-in jobs, as well as the submission rates, are monitored using the information from the *collector*, *submitter* and the retrieved output after the termination of the glide-ins via Condor-G. Fig. 2 shows the pilot statistics and its client usage.

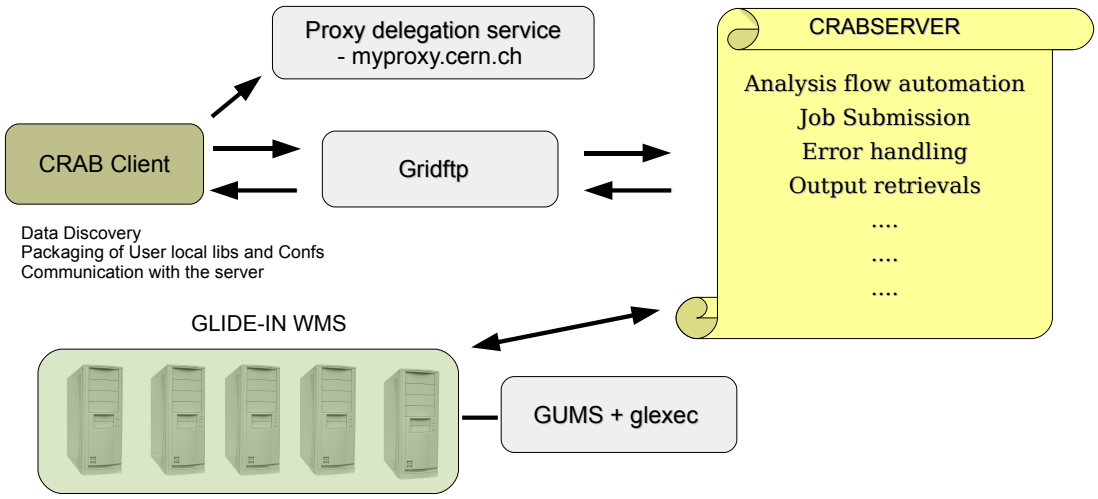


Figure 6. Schematic diagram of CRABserver in association with glideinWMS.

However, it is essential both for the end-users and the glideinWMS administrator to have a secure and transparent access to the real-time job monitoring. This is achieved using "detailed job monitoring" discussed in detail in [7]. The framework enables users to track jobs in substantial detail in quasi-real time as shown in Fig. 7. The monitoring system aims to achieve the following goals:

- For the end users to be able to track jobs using job ID and get
  - the summary information, process list
  - CPU, memory and disk usage as a function of time
  - job and working directory listing
  - access to the log files (e.g stdout, stderr)
  - status of the node that runs a job as well as site information
  - submission/start time
- For the administrators in real time to
  - track jobs using local job ID
  - find faulty worker nodes
  - spot problems quickly with misbehaving jobs (signified by zero load, expiring proxies, etc.)

The framework sensors use the information from the glide-in *collector* and *scheduler* to update the data at a regular interval to a MySQL Database. The database serves the monitoring information to the clients. The interface provides information regarding the CPU load, memory usage, job state and selection in quasi-real time. The database also stores summary information for the already completed jobs for archival purposes. The monitoring system is currently under active development and promises to provide access in real time to the logs of the currently running jobs at remote sites, using pseudo-interactive monitoring [8].

6. Experience using CREAM Compute Element (CE)

The Computing Resource Execution And Management (CREAM) is a next generation web service interface for job management operation at the CE level. Recent developments to Condor-G have focused on adding support for CREAM. This is accomplished for the first time by

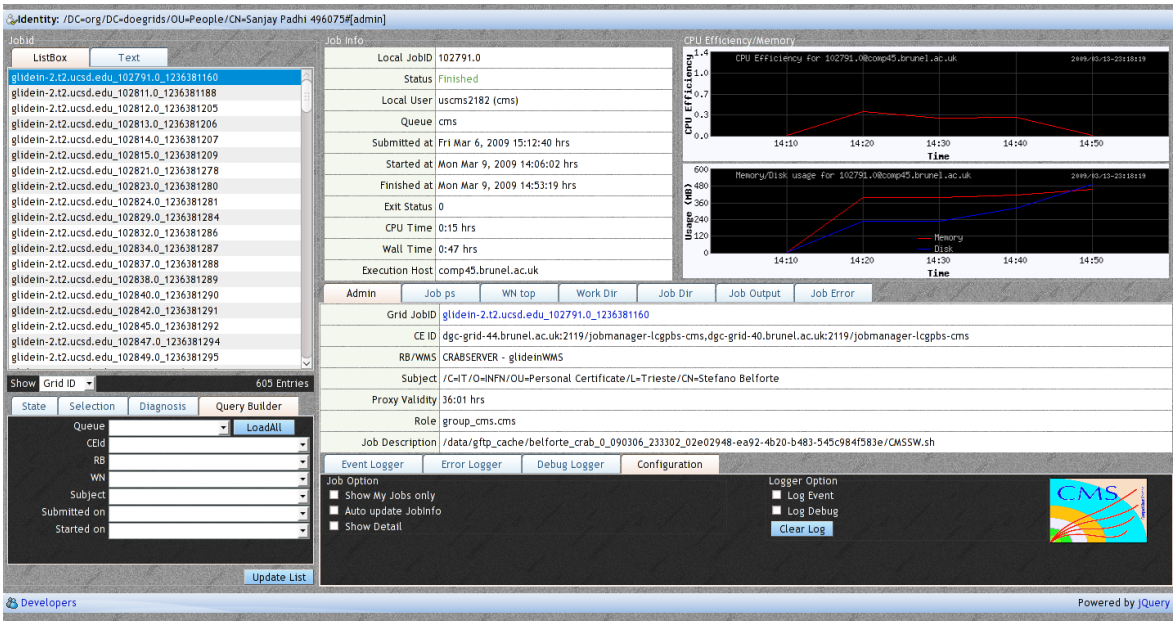


Figure 7. Web-based user job monitoring in quasi-real time

using GAHP daemons tailored to CREAM. It consists of various algorithms responsible for communication with the CE at different stages of job submission procedure. The responses from the CE are sent back to the Condor Gridmanager, which then updates the state of the jobs based on the provided info. The log retrieval is managed by Condor.

More than 10,000 jobs have been submitted to CREAM CEs as shown in Fig. 8. We observed a 25% failure rate, mostly due to proxy renewal/delegation. This problem is under investigation with the CREAM developers. Overall, this approach to interface Condor with CREAM seems to be quite promising. User jobs, submitted via glideinWMS are not affected due to above mentioned issues.

7. Conclusion

GlideinWMS has been extensively used in CMS for central data reprocessing, skimming at the Tier-1 centers, as well as user analysis. It provides a homogeneous pool of resources overlaid on a heterogeneous grid environment. CRAB-based user analysis efficiency benefits significantly from the late-binding approach. Detailed job monitoring associated with the WMS, provides tools for the users to access worldwide running jobs interactively. This is expected to enhance the user participation into the debugging infrastructure of the Grid. An interface to the CREAM CE has been created using Condor. We are in the process of integrating this interface with glideinWMS.

Acknowledgments

This work was partially funded by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Dept. of Energy, under Contract DE-AC02-06CH11357, U.S NSF grants PHY-0427113 (RACE) and PHY-0533280 (DISUN).

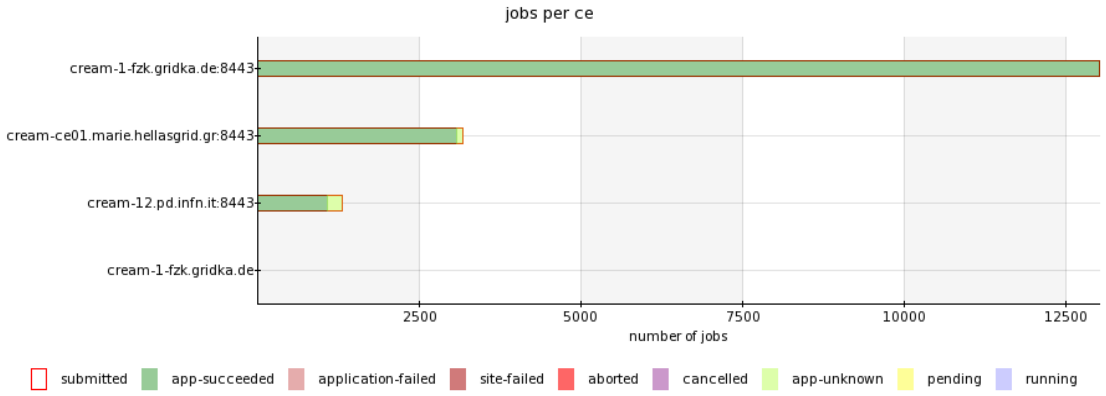


Figure 8. Number of resources used during the CMS tests of the CREAM CE.

References

[1] D. Bradley, I. Sfiligoi, S. Padhi, J. Frey and T. Tannenbaum, "Interoperability and Scalability within glideinWMS", CHEP 09, Prague, Czech Republic.

[2] CMS Collaboration 2005, CMS Computing Technical Design Report, CERN-LHCC-2005-023.

[3] J. Rehn et al., "PhEDEx high-throughput data transfer management system", CHEP 06, Mumbai, India.

[4] B. Holzman and I. Sfiligoi, "A Quantitative Comparison Test of Workload Management Systems", CHEP 07, Victoria, Canada.

[5] D. Spiga, "Automatization of User Analysis Workflow in CMS", CHEP 09, Prague, Czech Republic.

[6] A. Fanfani, "Commissioning Distributed Analysis at the CMS Tier-2 Centers", CHEP 09, Prague, Czech Republic.

[7] C. Dumitrescu, A. Nowack, S. Padhi and S. Sarkar, "A Grid Job Monitoring System", CHEP 09, Prague, Czech Republic.

[8] D. Bradley, M. Livny and I. Sfiligoi, "Pseudo-interactive monitoring in distributed computing", CHEP 09, Prague, Czech Republic.