CDF GlideinWMS usage in Grid computing of High Energy Physics

Marian Zvada[1,2], Doug Benjamin[3], Igor Sfiligoi[1]

[1]Fermi National Accelerator Laboratory, Batavia, IL 60510, USA
[2]Institute of Experimental Physics, Watsonova 47, 040 01 Kosice, Slovakia
[3]Duke University, Durham, NC 27708, USA

E-mail: zvada@fnal.gov, benjamin@fnal.gov, sfiligoi@fnal.gov

Abstract. Many members of large science collaborations already have specialized grids available to advance their research in the need of getting more computing resources for data analysis. This has forced the Collider Detector at Fermilab (CDF) collaboration to move beyond the usage of dedicated resources and start exploiting Grid resources. Nowadays, CDF experiment is increasingly relying on glidein-based computing pools for data reconstruction. Especially, Monte Carlo production and user data analysis, serving over 400 users by central analysis farm middleware (CAF) on the top of Condor batch system and CDF Grid infrastructure. Condor is designed as distributed architecture and its glidein mechanism of pilot jobs is ideal for abstracting the Grid computing by making a virtual private computing pool. We would like to present the first production use of the generic pilot-based Workload Management System (glideinWMS), which is an implementation of the pilot mechanism based on the Condor distributed infrastructure. CDF Grid computing uses glideinWMS for its data reconstruction on the FNAL campus Grid, user analysis and Monte Carlo production across Open Science Grid (OSG). We review this computing model and setup used including CDF specific configuration within the glideinWMS system which provides powerful scalability and makes Grid computing working like in a local batch environment with ability to handle more than 10000 running jobs at a time.

1. Introduction

CDF Run II started in 2001, and its compute systems have evolved tremendously during the lifetime of the experiment. The experiment collects events from protons-antiprotons collisions at the Tevatron collider at Fermi National Laboratory. In Run II software and analysis development work is done on desktop systems or the interactive login pool, ILP. Getting more computing resources for data analysis forced the CDF collaboration to move beyond the usage of dedicated resources and start exploiting Grid resources. Therefore, the main compute power for data analysis, Monte Carlo and processing comes from large PC farms in Grid installations at Fermilab and also around the world.

The CDF Grid PC farms are accessed through several layers of middleware. An experiment specific package, the CDF Analysis Farm, CAF [1], provides the users with a uniform interface to PC farms on different Grid sites running different middleware and batch systems. The CAF provides tools for submitting, managing and monitoring of batch jobs. In CAF terminology a job consists of multiple sections that all run the same script/executable but over different files of a dataset (or using different random number seeds in case of Monte Carlo).

Currently, CDF Grid infrastructure operates two batch systems, CDFGrid and NAmGrid from Fermilab and three smaller analysis farms, LCGCAF and CNAFCAF from Italy, and PacCAF from Taiwan. Except for LCGCAF all are glidein (pilot) based grid farms driven by glidekeeper mechanism developed as part of CAF package, whereas NAmGrid is using glideinWMS, pilot-based workload management system (WMS), which is generalized version of the glidekeeper developed by USCMS at Fermilab [2].

CDF is using the glidekeeper since a couple of years, but the increasing need of more CPU/RAM resources for data analysis experiment introduced several scalability issues due to its design. With the rising number of computing resources added to the CAF it became more difficult to keep production operations of the system sustainable with the glidekeeper, so CDF decided go for a better workload management solution, glideinWMS.

2. The CAF overview

The CAF consists of a set of daemons that receive requests from the users via kerberized [3] socket connections and coverts them into commands to the underlying batch system [4].

The first batch system used at CDF was FBSNG [5] but since 2001 Condor [6] replaced it. Figure 1 shows an illustrative concept of the CAF using Condor batch system:

- Users can develop, debug and submit jobs from the desktop.
- Authentication is in secure, kerberized, way.
- User job is wrapped in appropriate way by CAF components and propagated to batch system, Condor, which distributes the job on available worker node in the pool.
- Notification and summary of the end of jobs is via email.

Figure 1. The CAF concept overview.

Set of daemons which serve user's requests are as follows:

- Submitter daemon – creates batch system description file based on user input and submits job to the queue
- Monitor daemons – creates interface between the job and user via pseudo-interactive monitoring or web monitoring
- Mailer daemon – monitors the job and once job is done, notifies user about the status with specific summary in the email

Detailed scheme and functionality of these components is described in [1,4], but you can also get a summary view in figure 3.

3. CDF towards to Grid: GlideCAF

The first GlideCAF [7] was an evolution of the dedicated Condor-based CAF, with the inclusion of the Condor glide-in mechanism. See the picture below for schematic overview of the glidein mechanism (figure 2).

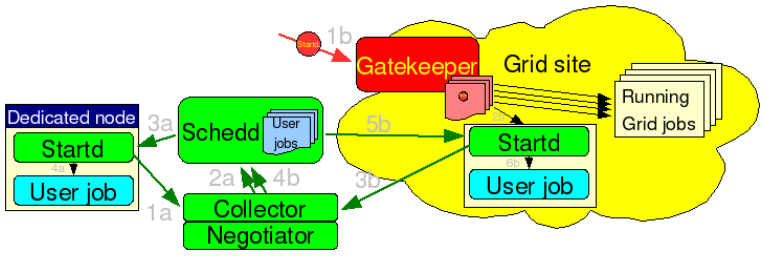

Figure 2. The Glidein mechanism overview.

A dedicated Condor installation has a Condor daemon running on each worker node forming a Condor pool. The Condor glide-ins allow for submission and execution of Condor daemons on Grid resources, making them temporarily part of the CDF Condor pool. The first GlideCAF had a large pool of dedicated worker nodes. Glideins were used to gather additional, mostly opportunistic, resources from the Grid. Recent GlideCAF installations are composed of only Grid resources, both CDF owned and opportunistic.

In order to handle Grid resources a new daemon, the glidekeeper, was added to the CAF code (see figure 3). The glidekeeper submits glidein jobs as needed to all the sites that are in the CDF wish list.

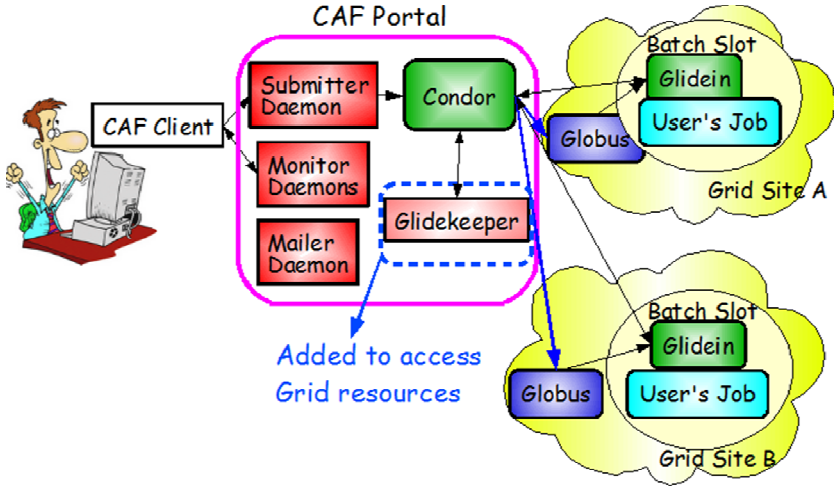

Figure 3. The Condor based GlideCAF layout. The glidekeeper submits glidein jobs to the Grid. When a glidein starts on a worker node, it join the CDF pool and starts asking for users jobs. The glidein job can run many user jobs [4].

3.1. Authentication flow in GlideCAF

The Grids CDF uses, i.e. OSG and EGEE, use x.509 proxy certificates for authentication. Condor-G is used for submission to the Grid resource, and the proxy certificate is delegated down to the worker node. The glideins thus use the available proxy certificate for authentication between the startd, the collector and the schedd.

To keep authorization uniform, CDF decided to use x.509 proxy certificates for all authentication within the GlideCAF; the only exception is communication with the users, where kerberos is used for historical reasons. The proxy certificates are also extended with CDF VOMS (Virtual Organization Management Service) attributes for proper authorization on Grid sites.

3.1.1. User authentication flow

- The user acquires Kerberos credentials on his workstation.
- User Kerberos credentials are used to establish a kerberized connection to the Submitter so that his/her identity can be verified on the head node (see figure 3).
- Once the user has been authorized using his/her interactive kerberos credentials, the Submitter uses the user identity to create a user-specific service kerberos ticket. The Fermilab KCA trusts the CDF portal to behave as an Identity Provider.
- The user-specific service kerberos ticket is converted into a user-specific service x.509 proxy certificate by using kx509. The proxy is then VOMS-extended using voms-proxy-init.
- The user-specific service x.509 proxy certificate is used for submission to Condor.

3.1.2. Refreshing user credentials

User jobs also need a valid VOMS-extended proxy certificate for remote IO when running on a worker node. The user proxy certificate is delegated from the CAF portal to the worker nodes by the Condor daemons. However, since x.509 proxy certificates have a short lifetime, the GlideCAF infrastructure continuously renew a user's proxy certificates on the portal node as long as there are any user jobs in the batch queue.

The renewal is handled by a cron script, running once every 12 hours. The cron script monitors the batch queue, and for every user in the queue issues a renewal command. As in job submission, the renewal command first creates a user-specific kerberos ticket and then converts it into a VOMS-extended x.509 service proxy credential.

There are actually two different scripts, one for user credentials and one for the glidein service proxy.

The renewal command looks for the validity of existing credentials; if the remaining lifetime is above 100 hours, no action is taken. Else, new credentials with a lifetime of 133 hours are created. In all cases, tickets for non-active users are deleted.

3.1.3. Disadvantage

Fermilab Kerberos tickets have a maximum lifetime of 7 days, so the derived x.509 proxy certificates cannot have a longer lifetime either. If execution were perfectly synchronous to submission, the user proxy certificate could be used. But jobs can stay in the queue for days, so proxies need to be refreshed on a regular basis.

The CAF portal obviously cannot refresh neither the user Kerberos tickets nor create user proxy certificates. This forced CDF to use user-specific service proxy certificates instead of the real user proxies for the interaction with the Grid. The drawback is that the CAF portal is now a very sensitive machine that needs to be carefully guarded, as it acts as a Identity Provider.

4. The GlideinWMS

4.1. Overview

The glideinWMS is a WMS based on a virtual private Condor pool. The system is composed of several elements and some of them can be multiplied [2]:

- *Pool Central Manager Node.* This machine runs a collector and a negotiator and holding the state of the virtual private pool.
- *Submit Node.* These machines run the schedds and keep the job queues.
- *glideinWMS Glidein Factory.* Each of these machines run a glidein factory daemon that will submit the pilot jobs to a set of Grid pools.
- *glideinWMS VO Frontend.* Each of these machines run a VO frontend daemon that monitors the schedd queues, matches them to the glidein factories, and decides which glidein factory should submit the pilot jobs, and how many of them.

- *glideinWMS Collector.* This machine runs a collector that is used for communication between the glidein factory daemons and the VO frontend daemons.

Detailed diagram of glideinWMS components and its processes flow is shown at figure 4.

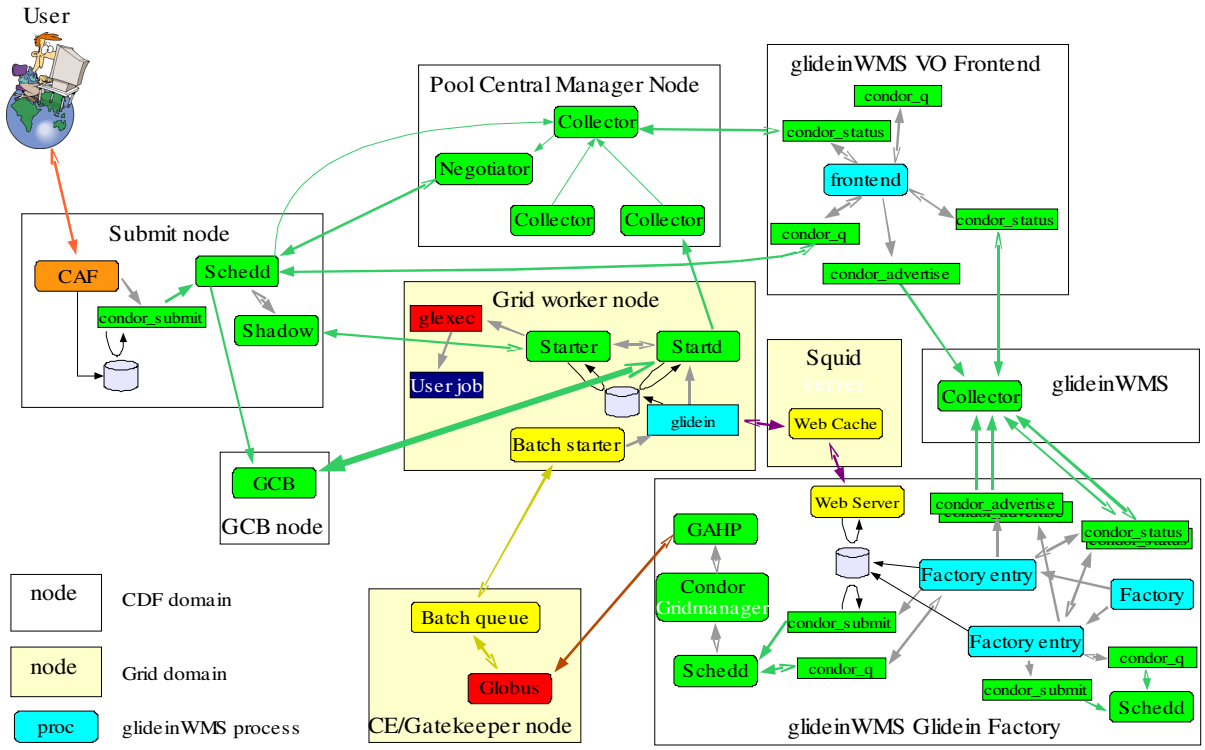

Figure 4. Detailed glideinWMS implementation with CAF.

Operation logic [2] of this system tries to maximize the amount of user jobs, while minimizing the amount of wasted resources. It does this by keeping a steady pressure on the Grid pools; as long as there are jobs in the schedd queues that could potentially run on a Grid pool, a fixed number of pilot jobs is being kept in that pool queue. However, as soon as there are no more suitable jobs waiting in any of the schedds, no more pilot jobs are submitted. If any glideins start after all the suitable user jobs have started, the glidein itself will exit within a few minutes.

The pilot jobs are being submitted by the Glidein Factories, but it is VO Frontends' job to decide how many pilot jobs to keep in each and every Grid pool. This number is calculated by matching the attributes of the user jobs, provided by the schedds, to the attributes of the Grid pools, provided by the Glidein Factories. If the number of matches is higher than the desired pressure, the Glidein Factories are told to keep the pressure. Else, the pressure is reduced as appropriate.

Once a pilot job starts on a Grid resource, it first validates it. This includes looking for the appropriate software libraries, ensuring sufficient disk space is available, and so on. Once these tests successfully complete, it configures the Condor daemons (i.e. the startd and supporting code) and starts them. All control is delegated to them, and the pilot job wrapper just waits for their termination to do the final cleanup.

Once the Condor daemons start, they behave like in a dedicated Condor pool. The startd registers back with the Condor central manager and waits to be matched. Once a suitable job is found, the schedd holding the job will contact the startd and the job starts running; see figure 5 for a schematic view. The startd can run multiple consecutive user job, to keep the pilot wrapper overhead low for very short jobs. However, the pilot needs to end within the Grid back slot lease time, so the startd will stop accepting new jobs if they cannot complete in time.

One important observation is that the negotiator on the Condor central manager machine decides which job(s) will be run on the resource provided by the pilot job. The decision is taken the moment the resource becomes available, so the job started on a given glidein is most probably not the one for which the glidein was submitted. This is especially true when several jobs are run by the same glidein.

The system has been designed to be highly scalable. There are only two pieces that cannot be replicated; the Condor central manager and the WMS collector, and both are relatively lightly loaded. Any other daemon, if overloaded, can be split into two or more daemons of the same kind, and its load distributed among them.

4.1.1. Using GCB over NATs and Firewalls

Condor has been developed in a network friendly environment, where bidirectional network traffic was not a problem. In the present Grid world, a significant fraction of computing resources are behind a NAT or have some sort of firewall in place. As shown in figure 4, the schedd needs to talk to the startd in order for a job to start, so a NAT or a strict firewall at the Grid pool border will not allow such communication to occur.

To work in such environment, Condor recently introduced the Generic Connection Broker (GCB). In this scenario, the startd first establishes a long lived TCP connection with a GCB sitting on a public network node, and GCB associates it with a dedicated TCP port. The startd then communicates to the central manager the obtained GCB port instead of the local TCP port. When anyone needs to talk to the startd, they will contact the GCB server that will relay the network traffic to the startd. Since the long lived TCP connection was initially originated from inside the NAT and/or firewall, all successive traffic, no matter the direction, is permitted from then on.

4.2. The CDF GlideinWMS

As already mentioned, glideinWMS system has been designed to be highly scalable. CDF first implementation of glideinWMS with CAF package ended up in production with 3 physical machines (excluding one machine as GCB while submitting offsite, see chapter 4.1.1.) configured as follows:

- 1 machine as *"Condor Central Manager (collector/negotiator) + VO Frontend"*.
- 1 machine as *"Submit machine with user schedds and CAF package"*.
- 1 machine as *"Glidein Factory with glidein schedds and glideinWMS collector"*.

Note, that for submit machine with the CAF code nothing really changed from the GlideCAF, apart from the fact that we don't use the glidekeeper anymore.

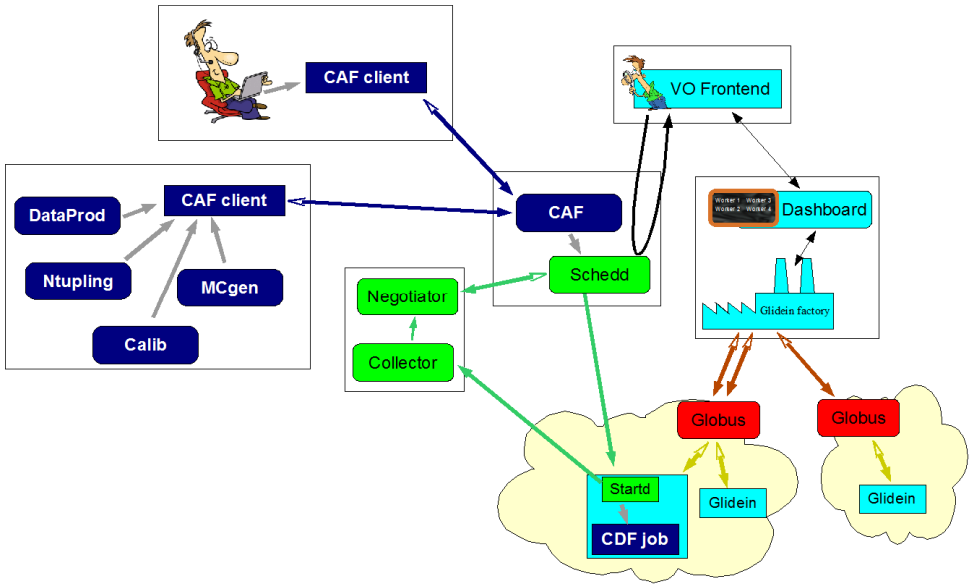

Figure 5. CAF implementation with glideinWMS.

5. CDF GlideinWMS scaling tests

Major goal of glideinWMS implementation for CDF was reduce load of the current glidekeeper system which is serving all processes on one head node. Thanks to glideinWMS, distribution of services across three machines helped much better utilize CPU load per processes while running 10k jobs on submitter head node, whereas glidekeeper system (GlideCAF) consumes significantly more while running just 8k jobs (running >8k jobs was causing instability of the system with glidekeeper). Memory footprint was measured as well, there is unambiguously improvement using glideinWMS too. Tests were done with Condor v7.2.0 and glideinWMS v1_5_2. See figure 6 for results.

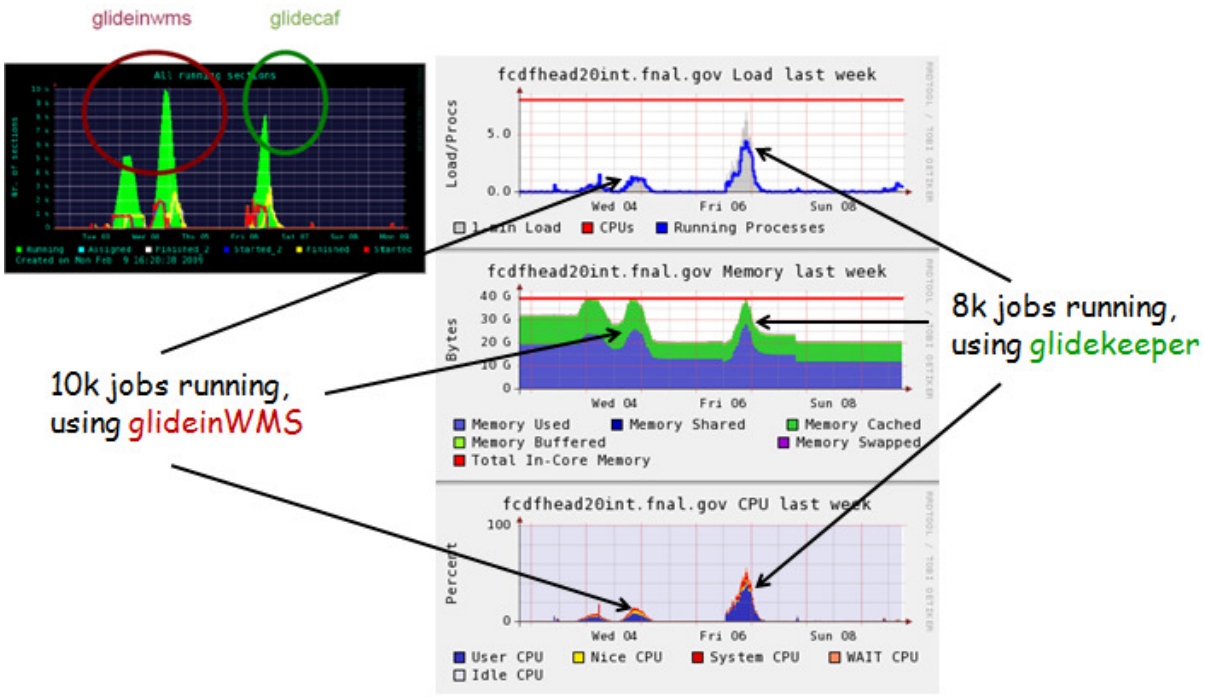

Figure 6. Scaling tests results, glideinWMS vs. glidekeeper.

Scalability of any WMS is an important concern, therefore further effort based on experience is put by glideinWMS developers to understand and extend the limits of scalability in glideinWMS [8].

6. CDF GlideinWMS in production: NAmGrid

NAmGrid is the first production CDF CAF using glideinWMS. After successful scaling tests, CDF introduced NAmGrid cluster to users in late February 2009. NAmGrid is sending glideins to Open Science Grid (OSG) resources, and is currently configured to use 7 different gatekeepers. Figure 7 and 8 represents usage of this cluster over the last month.

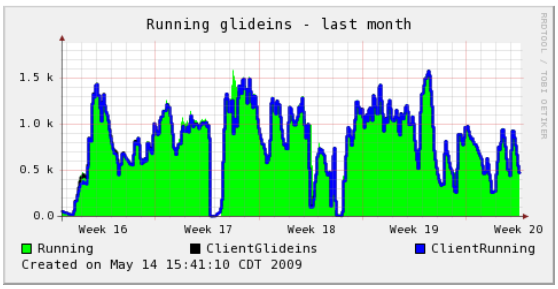

Figure 7. Running jobs/glideins.

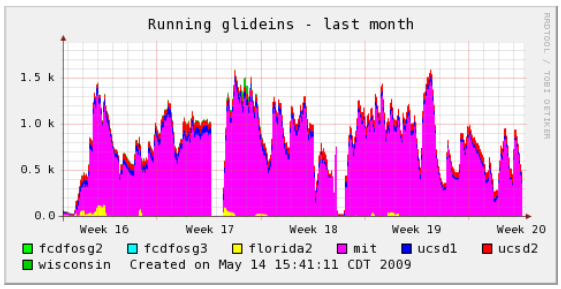

Figure 8. Running jobs/glideins on OSG sites.

7. Conclusions

CDF has been successfully using Grid resources through glideins for the past 4 years, but we reached the scalability limits of the home grown software, the glidekeeper.

USCMS@FNAL has developed a more scalable glidein solution (glideinWMS), very similar to the CDF glidekeeper since glideinWMS borrowed heavily from the CDF experience. Development of glideinWMS has been driven as general purpose project, so we can freely use this product.

Experience up to now is very positive and CDF is in preparation phase deploy glideinWMS also on its CDFGrid cluster used primary for production data analysis from Tevatron. There is continuous plan on to add more available OSG opportunistic resources to NAmGrid soon and keep physics data analyzing moving on across the computing elements out of Fermilab too.

References

[1] M. Casarsa, S. C. Hsu, E. Lipeles, M. Neubauer, S. Sarkar, I. Sfiligoi, F. Wuerthwein "The CDF Analysis Farm" 2005 *AIP Conf. Proc.* **794** 275.

[2] I. Sfiligoi "glideinWMS - A generic pilot-based Workload Management System" *presented at Computing in High Energy and Nuclear Physics, 2-7 September 2007, Victoria, British Columbia, Canada published on Journal of Physics: Conference Series 119 (2008) 062044 http://www.iop.org/EJ/volume/1742-6596/119*, glideinWMS Web site *http://www.uscms.org/SoftwareComputing/Grid/WMS/glideinWMS/doc.html.*

[3] "Kerberos Web site" *http://web.mit.edu/Kerberos/.*

[4] D. Lucchesi "CDF way to Grid" *presented at Computing in High Energy and Nuclear Physics, Prague, Czech Republic, 21-27 March 2009* **414** 2009.

[5] "FBSNG Web site" *Next Generation of FBS http://www-isd.fnal.gov/fbsng/.*

[6] D. Thain, T. Tannenbaum, M. Livny, "Distributed computing in practice: the Condor experience.", *Concurrency - Practice and Experience* **17, 2-4**, 323 (2005).

[7] S. Sarkar, I. Sfiligoi *et al.*, "GlideCAF - A late binding approach to the Grid" *presented at Computing in High Energy and Nuclear Physics, Mumbay, India, Feb. 13-17, 2006* **147** 2006.

[8] D. Bradley, I. Sfiligoi, S. Padhi, J. Frey and T Tannenbaum "Interoperability and scalability within glideinWMS" *presented at Computing in High Energy and Nuclear Physics, Prague, Czech Republic, 21-27 March 2009* **218** 2009.