OSG-CAF - A single point of submission for CDF to the Open Science Grid

M. Norman, S-C. Hsu, E. Lipeles, M. Neubauer, F. Würthwein
University of California, San Diego, USA
I. Sfiligoi, INFN Frascati, Italy
S. Sarkar, INFN-CNAF, Bologna, Italy

Abstract

The increasing luminosity of the Tevatron collider is causing the computing requirements for data analysis and MC production to grow rapidly. In order to meet future demands, CDF is moving from dedicated to shared, Grid, resources. Moreover, a significant fraction of opportunistic Grid resources is expected to be available to CDF before the LHC era starts and CDF could greatly benefit from using them. CDF is therefore reorganizing its computing model to be integrated with the new Grid model. In the case of Open Science Grid (OSG), CDF has extended its CDF Analysis Farm (CAF) infrastructure by using Condor glide-in and Generic Connection Brokering (GCB) to produce a CDF portal to the OSG that has an identical user interface to the CAF infrastructure used for submissions to the existing CDF dedicated resources, including its semi-interactive monitoring tools. This talk presents the architecture of the OSG-CAF and its current state-of-the-art implementation. We also present the issues we have found in deploying the system, as well as the solution we adopted to overcome them. Finally, we show our early prototype which harvests the opportunistically schedulable resources on the OSG in ways that are transparent to the CDF user community.

THE CDF COMPUTING MODEL

Development and History

The Collider Detector at Fermilab (CDF) is a detector on the Tevatron, currently the world's highest energy particle collider. The CDF detector was thoroughly upgraded, and began its Run II phase in 2001. At that time, a new computing model was drafted for CDF to meet the challenges of a higher luminosities and a larger collaboration.

The computing plan evolved to include two separate pieces under the title of the CDF Analsysis Farms (CAFs). Computing resources on-site at Fermilab form the heart of the CAF, comprising some 800 nodes, with direct access to CDF data stored in diskpool or on tape [2]. Off-site computing resources, the dCAFs, consist of smaller pools at various institutions available to CDF users, but without the access to the data handling system enjoyed by on-site nodes [3]. The dCAF system proved successful, currently involving over ten sites on three continent, and is used primarily for MC production which frees more resources at FNAL for data analysis.

Motivations to the Grid

With time CDF has had to re-evaluate its computing plan. Increases in Tevatron luminosity and the CDF collaboration have increased its computing requirements. Meanwhile, it becomes increasingly impractical to continue to construct large, dedicated computing facilities for CDF, especially as CDF nears the end of its operational life. Management with more than approximately ten dCAFs also becomes a substantial overhead. While this is happening, many of the original difficulties with running in the grid paradigm have disappeared. Recent developments allow for data transfer to distributed computing sites, and increased organization has lead to greater utility.

As the LHC era nears, the spurt of Grid activity has demonstrated the feasibility of truly distributed computing. It is time for CDF to move to the Grid.

First Steps to the Grid: GlideCAF

The first major step onto the Grid was accomplished by use of the GlideCAF system.

A standard dCAF is a Condor-based [1] dedicated cluster with two major components: a CDF portal consisting of several daemons and a Condor pool. The portal acts as an interface to the Condor pool for CDF users. This requires the dCAF admin to simultaneously manage the headnode and the worker nodes.

In the GlideCAF, only one node is under the control of CDF: the headnode. The Condor pool is dynamically created out of another Grid pool via the submission of Condor glide-ins. More details can be found in [4] and [5].

GlideCAFs have been successfully deployed at several sites, however the sheer number of portals is becoming a problem.

OSG-CAF

The Concept

The OSG-CAF has been visualized as a single point of submission portal for CDF users to access the Grid without ever needing to learn any of the Grid protocols. The basic idea was to make it look and feel like any other CAF portal, where a single headnode would be capable of running jobs on all OSG sites.

Since we did not want to reinvent the wheel, we used the GlideCAF as a starting point. However, the basic GlideCAF inrastructure had several problems:

- The headnode must be located near the worker nodes, since Condor uses two way UDP traffic to communicate between the headnode and the worker nodes, making it impossible to work over firewalls. While this is feasable for major sites, it would effectively prevent us to gain access to smaller sites.
- Every user job comes with a tarball to transfer. Given the large sizes possible for user tarballs, mass transmission over the WAN (as is currently done locally) is impractical
- Most CDF jobs require access to the CDF software distribution. At presently deployed GlideCAFs, a shared file system is used for this.
- The base GlideCAF system has inbuilt scaling problems past a few thousand VMs.

All the above problems have been addressed, and the solutions are presented below.

Firewall Routing: GCB

Site firewalls and security settings have been the primary difficulty for this venture. Without the ability to communicate with a glide-in hosted on an OSG site, the OSG-CAF is worthless. Fortunately, Condor has recently deployed a tool called Generic Connection Brokering (GCB) [6]. OSG-CAF uses a GCB server, placed on a node with incoming network access, to allow cross-firewall communication. Upon initiation, a glide-in will open a persistent TCP connection to the GCB server at the same time it calls-out to register with the Condor Central Manager (CM). Any time a Condor daemon needs to talk to the glide-in, instead of trying to talk directly to the glide-in, it will issue a call to the GCB server. The GCB server will then transmit an instruction to the glide-in, and the glide-in itself will open a connection to the correct daemon to allow communication.

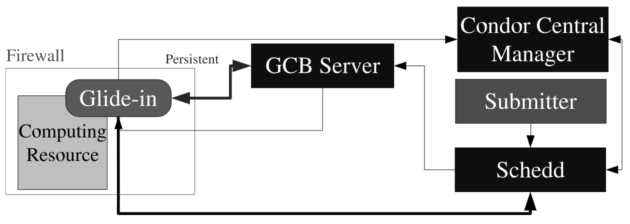

Figure 1: GCB layout

GCB is very flexible-the GCB server can be set up anywhere in the world, as long as it accepts traffic. For scalability reasons, several different GCB servers can be installed on different machines. GCB also supports strong authentication, running right now with GSI. Kerberos support is still a possibility for future upgrades. Additionally, should we encounter a strict site which allows neither calls in nor calls out, a GCB server can be established at each end to allow signal routing, restoring communication.

Reducing WAN Traffic: HTTP Caching

Another large constraint is distributing user tarballs over the WAN to Grid sites. Due to the nature of CDF analysis, some tarballs can approach sizes of 250MB, with one tarball being used by hundreds of parallel sections. In the GlideCAF model, the tarball is copied from the head to the worker node for each and every section, which is obviously unfeasable over the WAN. The solution we adopted is based on cached HTTP transfers. In our view, a Web server is mounted on the headnode,a HTTP cache running near every Grid site and the tarball is pulled to the worker nodes using a standard command line web tool.

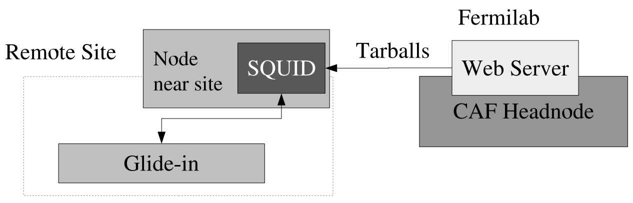

Figure 2: Tarball transfer via HTTP

Since HTTP transfer is a widely used and well developed technology, it has been easy to adapt to our purposes. HTTP's true advantage, however, is that both HTTP servers (Apache) and HTTP caches (Squid) are readily available. By caching tarballs locally to the site, we can dramatically reduce WAN traffice by making the majority of transfers purely local. Additionally, an HTTP cache running on a node near the headnode can decrease traffic to dozens of distributed grid sites.

Installing and maintaining a central Web server and instrumenting the CAF wrapper to pull the tarball via web tool was simple. The real problem lies in getting a HTTP cache near each and every site. The best scenario for us would be if the HTTP cache is installed and maintained by the site admins. To this end, we are working on a proposal to include a SQUID cache as part of the next major OSG release. However, when this is not possible, such a cache can be installed outside the site,but still nearby network-wise. It is also worth mentioning, that for small sites the cache is often not needed in the first place, drastically reducing the problem.

CDF Code Deployment: Parrot

Most CDF jobs require access to the CDF software distribution to work. Although CDF has created self-contained tarballs for the most used applications, like the production simulation jobs, the number of other use cases make this path unfeasable. In order to accomodate those use cases, we have looked at ways to access the CDF software distribution while running on Grid worker nodes.

The most obvious solution would be to install and maintain a copy of the distribution on each and every site. Apart from the fact that we have no guarantee to have enough space to host the whole distribution on each and every site, maintaining such a huge number of copies around the world becomes a maintanance nightmare.

Instead, we decided to use Parrot over HTTP. Parrot is an application that traps user executable I/O calls and reroutes those that meet its specifications. By exporting the CDF software distribution via a Web server, Parrot can route read calls from any executable to those web pages. In other words, Parrot can essentially “mount” files stored on a web server as if they were on a local disk partition, without the need for root priviledges. Paired together with HTTP caching, this makes software distribution an easy task. See [7] for more details.

Scalability: Condor-C

The primary limitation for scalability is the single headnode. Most CAF headnodes run a single schedd, which tracks and manages all job sections. Jobs are submitted as DAGMans, which then submit the individual sections. Tests on our production system reveal that a single schedd, with hundreds of users to track, is not scalable past approximately 2500 running sections. This will not suit our computing needs for the Grid.

One possible solution is to switch to multiple schedds, using one schedd for submitting the DAGMans, while the sections are distributed between the other schedds. This scales much better, but requires high end hardware, both in terms of number of CPUs and amount of available memory.

Since we wanted to use commodity hardware whenever possible, we started looking for an alternative solution. The most obvious step was to distribute the schedds over multiple nodes, reducing the requirements for those nodes. However, in order to maintain sequential non-overlaping job IDs, all DAGMan submissions need to go to the main schedd.

Unfortunatelly, one cannot just plainly submit the DAGMans and make them submit the sections to schedds running on other nodes. So we resolved to use Condor-C. By using Condor-C, we can make our primary schedd, upon reception of a job, immediately reroute the DAGMan to a schedd on another machine. The DAGman will actually start on that other machine and spawn the sections to that same local schedd. All the load of that job is thus confined to that other node, while the load on the headnode is minimal.

Using multiple nodes increases stability, too. Using Condor’s High Availablity features, any node can go down without losing the pool, including the headnode.

OSG-CAF PERFORMANCE

So far the OSG-CAF has been operated on a testing-only basis, as it is too small scale for production jobs. Part of this is due to the effort necessary to obtain enough open slots on OSG sites for full load testing. However, testing has proceeded at the low load level. Jobs can be submitted from the desktop of any CDF user, using kerberos authentication, and then distributed to glide-ins running at various

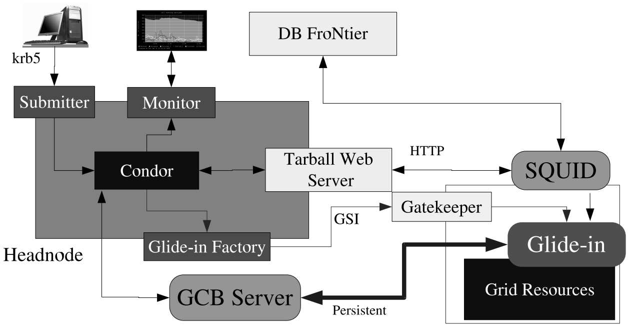

Figure 3: The OSG-CAF

OSG sites from SDSC to Wisconsin. No untoward problems have yet been discovered with this system.

GCB Performance

GCB was tested at Fermilab using a Condor Central Manager on one machine communicated strictly via a GCB server on a separate node. Tests were performed in which Condor glide-ins were submitted to various OSG sites with firewall protections. Although glide-ins could not previously communicate through those firewalls, it was found that they could connect via GCB. This test has been successful at the San Diego Supercomputer Center (SDSC), GPFarms at Fermilab, GLOW, and UIowa. Batch jobs were successfully submitted to these sites.

Further scaling tests for GCB will have to be run as the opportunities present themselves.

Squid deployment

Squid daemons were tested both with test jobs at Fermilab and at production GlideCAFs in San Diego and Bologna. The FNAL test is currently running on 4000 active VMs, while the GlideCAFs have been running with more that 1000 parrallel user sections. No issues have been found on any of them.

Parrot Testing

Parrot has been tested at FNAL with both test and real jobs. In these tests, jobs were able to access the CDF software distribution and run as if it was mounted via NFS. See [8] for more details.

Large scale testing, running on Grid nodes and including HTTP caches, is expected to start soon.

Condor-C Scalability Testing

Condor-C is currently in testing in FNAL. It has not yet entered production due to a slight bug with kerberos authentication (which is required for operations on-site), but it has been tested with up to 4000 VMs with GSI authentication and has performed well. An empty cluster of four thousand VMs fills in less than six hours without undue strain on the Condor Cental Manager. When combined with High Availability it has also proven resilient to headnode crashes and other system errors.

FUTURE CHALLENGES

For the near future, we are proposing the following goals:

- Increase load testing for OSG-CAF by running more than five hundred glide-ins simultaneously on OSG sites
- Begin beta testing by submitting CDF MC production jobs with the self-contained tarball through OSG-CAF
- Set up the web server necessary for code deployment via Parrot
- Load test a local SQUID cache for tarball distribution

For a slightly longer term, our goals are:

- Open OSG-CAF for production
- Merge all the US dCAFs into the OSG-CAF
- Begin testing of data deployment via SAM/SRM (when available)

CONCLUSIONS

CDF is on track to have a single point of submission for the whole Open Science Grid, and is doing it without any change in the habits of the physicists. We are building on the success of the GlideCAF, widely deployed at this time, and are addressing all the limits of the current system. Althought the OSG-CAF is not yet in production, we expect to open it to real users within a month.

ACKNOWLEDGMENTS

We would like to acknowledge the assistance of the OSG management in organizing and documenting the OSG, as well as that of the Condor team for their excellent work on supporting their development product. Much of this work was done with the aid of many OSG admins, especially Terrence Martin at SDSC, and could not have been completed without them.

REFERENCES

[1] http://www.cs.wisc.edu/condor/

[2] I. Sfiligoi, E. Lipeles, M. Neubauer and F. Würthwein, The Condor based CDF CAF, CHEP ’04, Interlaken, Sep. 2004

[3] A. Sill, et. al., Globally Distributed User Analysis Computing at CDF, CHEP ’04, Interlaken, Sep. 2004

[4] S. Sarkar and I. Sfiligoi, GlideCNAF - A Purely Condor Glide-in Based CAF, CDF Note 7630 (2005)

[5] S. Sarkar et. al., GlideCAF - A Late-binding Approach to the Grid, CHEP ’06, Mumbai, Feb. 2006

[6] http://www.cs.wisc.edu/ sschang/firewall/gcb/

[7] D. Thain and M. Livny, Experience with Parrot: User-Level Transparent Middleware for Data-Intensive Computing, in Workshop on Adaptive Grid Middleware, New Orleans, September 2003

[8] D. Thain, C. Moretti, and I. Sfiligoi, Transparently Distributing CDF Software with Parrot, CHEP ’06, Mumbai, February 2006