GlideCAF: A Late Binding Approach to the Grid

Stefano Belforte, INFN Trieste, Italy
Shih-Chieh Hsu, *Elliot Lipeles*, *Matthew Norman*, *Frank Wuŕthwein*, University of California, San Diego, USA
Donatella Lucchesi, INFN Padova, Italy
Subir Sarkar,* INFN-CNAF Bologna, Italy
Igor Sfiligoi, INFN Frascati, Italy & Fermilab, Chicago, USA

Abstract

Higher instantaneous luminosity of the Tevatron Collider requires large increases in computing requirements for the CDF experiment which has to be able to cover future needs of data analysis and MC production. CDF can no longer rely solely on dedicated resources in order to meet all of its computing needs and is therefore moving towards shared, Grid resources. CDF has been running a set of CDF Analysis Farms (CAFs), dedicated pools of commodity nodes managed as Condor pools with a small CDF specific software layer on top of it. We have extended this model by using the Condor glide-in mechanism that can create dynamic Condor pools on top of existing batch systems, without installing any additional software. A GlideCAF is essentially a standard CAF with the addition of the tools needed to keep the dynamic pool alive. All the monitoring tools supported on the dedicated CAFs, including semi-interactive access to the running jobs and detailed monitoring, have been preserved. In this paper, we present the salient features and implementation detail of glide-in based Condor pools. We also show the amount of resources we manage with GlideCAFs and the turn-around of resources we achieved. Finally, we discuss about the limitation of this approach.

INTRODUCTION

CDF [1] is a High Energy Physics experiment at the Tevatron collider located at the Fermi National Accelerator Laboratory in Chicago, USA. The experiment has been collecting Physics data since 2000 during the Tevatron Run II phase. CDF acquires data at a sustained rate of 60 MB/s and has so far collected about 1 PB of physics data. Over the years, improvements in accelerator running condition have resulted in a higher instantaneous luminosity, while that of detector and trigger conditions have steadily increased the data taking efficiency. These mandated an ever increasing amount of computing power for analysis and simulation. The requirement of processing power today is of the order of 5M SPECint2000 which is expected to climb to almost 15M SPECint2000 by the end of 2007.

The current CDF model for event reconstruction, simulation and analysis is based on dedicated Condor [2] pools with a CDF specific submission infrastructure, called CDF Analysis Farms (CAFs) [3]. CDF has so far successfully maintained its own dedicated resource based global computing environment with (a) Central production and analysis farms at Fermilab and (b) 10 decentralised Analysis Farms (dCAFs) as well as a few farms devoted to supervised Monte Carlo production, distributed over three continents. About 50% of the total CDF computing ($\sim$ 2.4M SPECint2000) is carried out at remote sites. CDF priorities for computing outside Fermilab is mainly for MC production although a few large sites have been identified as future Physics Centres. For example, CNAF, the Italian Tier-1 site, will soon become a B Physics centre for CDF.

In order to meet future needs, CDF is reorganizing its computing model to move away from dedicated pools towards the Grid computing environment. This has been dictated mainly by the realization that large increases in computing resources, required due to higher instantaneous luminosities of the Tevatron collider, will be practically impossible with the expansion of dedicated resources. There are several sites with shared resources, both in the LHC Computing Grid (LCG [4]) in Europe as well as the Open Science Grid (OSG [5]) in the USA, that could be available to CDF immediately, had the computing framework been able to take advantage of it. A large fraction of these resources is expected to be available to CDF before the LHC era starts, enabling the experiment to do more physics that it could afford otherwise. The CDF computing model is required to adapt to the Grid world and build dCAFs that can access shared resources, preserving all the CDF specific features such that the impact of the paradigm shift affects the end-users minimally. The new model must also continue to support all the existing applications.

Since the original CAF was Condor based, it was natural to extend the existing model with the Condor glide-in mechanism which is capable of dynamically extending an existing Condor pool onto Grid resources.

CONDOR GLIDE-INS

A regular Condor pool is composed of a set of daemons that manage different parts of the system. The Condor daemons communicate with each other sending UDP packets. The pool is defined by the *collector*, which gathers information about all the other daemons. The worker nodes are controlled by the *starter* daemons, while user jobs are managed by one or more *schedd* daemons. The *negotiator* assigns jobs to the *starters*, which execute them. A *master* daemon overlooks all other daemons and takes care of starting, stopping and restarting them. Figure 1 shows interaction among different Condor daemons in a regular pool. For more information see [3].

Condor glide-ins are regular Condor *starter* daemons,

* Corresponding author, email : subir.sarkar@cnaf.infn.it

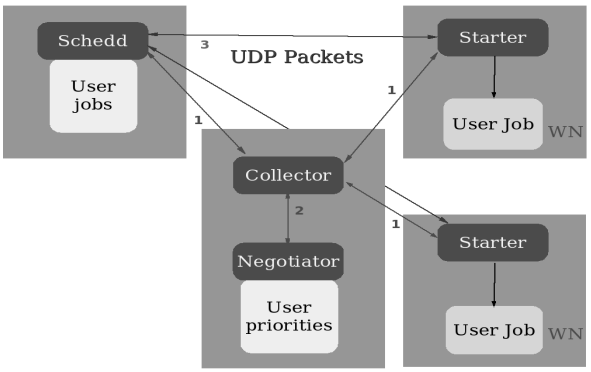

Figure 1: A dedicated Condor pool. A set of daemons manage different parts of the system. The daemons communicate with each other using UDP packets.

properly configured and submitted as jobs to a Grid Computing Element (CE). On successful authentication, the CE delegates the glide-ins to the underlying batch system. Once such a job starts on a worker node, it contacts the designated Condor *collector* and joins the Condor pool as a new virtual machine (VM). From the Condor point of view, such a resource is indistinguishable from a dedicated one, and will be matched in the same way to a user job with the best priority. A user job, sent to that VM, will eventually run on the Grid worker node.

An extension of a regular CAF with the Condor glide-in mechanism is known as a GlideCAF [6]. A GlideCAF brings in a slight modification of the Condor configuration and an addition of a glide-in factory that creates a virtual private Condor pool out of shared resources. Once a pool is established it looks and behaves like a dedicated one, and allows sophisticated monitoring, both interactive and web based, as well as user and group scheduling policies to work in a manner identical to a dedicated Condor farm.

The virtual private pool is managed by the glide-in factory (*glidekeeper*), a process that keeps submitting glide-ins to the Condor *schedd* as new user jobs arrive at the queue. Figure 2 gives a simplified view of how a virtual pool is created using the Condor glide-in mechanism.

The Condor glide-in mechanism has several advantages over direct submission of user jobs to the Grid CEs:

- **Finer grained policy management**: user priorities are managed inside the Condor pool, preserving all the power and flexibility of the Condor fair share policies. Moreover, the policies are managed at two different levels: (a) by virtual organization (VO) at the CE level, and (b) among users at the Condor level.

- **Black holes can be guarded against**: a defective node kills glide-ins before they start. As a result, no user jobs are lost. Additional sanity checks can be performed before a user job is sent, discarding worker nodes that do not meet the needs of the virtual organization. Nodes failing recurrently can be blacklisted as well.

- **Late binding of resources**: in case of multiple Grid sites, user jobs will be pulled only by those sites where glide-ins will have started and resources will be available. This will eliminate the risk of long delay at the CE queues of busy sites.

The glide-in mechanism is independent of the local batch system used at a Grid site; this makes deployment of a GlideCAF rather easy.

Implementation

The first step was to extend the Condor pools at those sites that already had a CAF installed and where CDF enjoys close ties with the local Grid administrators. Several such pools have been deployed at different Grid sites which are mentioned below,

- Production systems at
  - CNAF Tier-1 in Bologna, Italy
  - San Diego, USA
  - Fermilab, USA
  - Lyon Tier-1, France
- Systems in alpha test at
  - Karlsruhe, University College of London (UCL)

The largest production GlideCAF has been running at CNAF, the Italian Tier-1 centre in Bologna. The farm, deployed in September 2005, is one of the largest CDF batch farms. So far it has run 800k jobs from approximately 200 users. The system is robust with virtually no job loss and easily scales up to the full size of the present Tier-1, i.e.

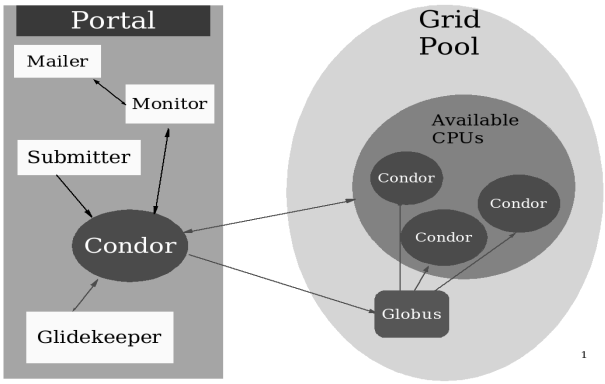

Figure 2: A glide-in based Condor pool. On the CAF portal, various pieces of the CAF framework are shown, that manage user job submission, monitoring etc. *glidekeeper* is the new component that was added in order to extend a standard CAF.

≈2k slots (≈ 1.5M SPECint2000). Figure 3 shows the usage pattern of the farm for the last two months and points out the turn-around with the Condor glide-in approach. The only major complaint from users was related to long directory path names introduced due mainly to the presence of multiple middleware layers.

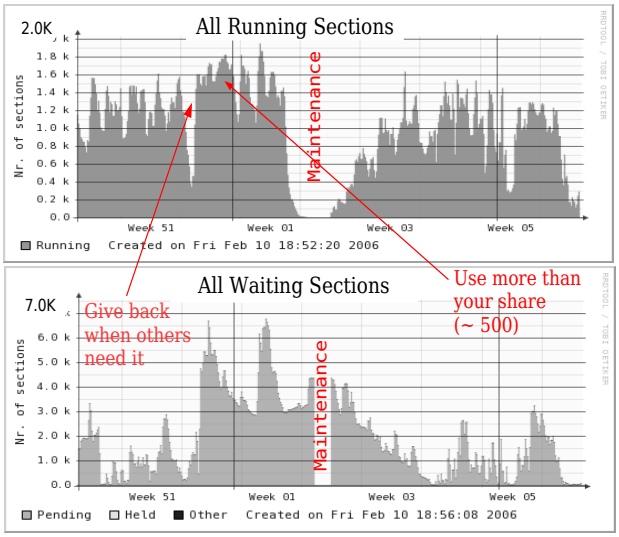

Figure 3: Usage pattern of the GlideCAF at CNAF during the last two months. The upper and lower plots show all the running and waiting sections respectively during that period. The opportunistic use is also indicated in the upper half.

GlideCAF Issues

GlideCAFs internally use GSI based authentication throughout the system. However, it must be noted that only a single CDF service proxy is used for all the glide-in jobs. As a consequence, the site gatekeeper does not know anything about the real users although Condor has a complete knowledge of all the users. All the jobs run under a single VO specific UID (e.g cdf001). This is an outstanding privacy issue GlideCAFs will have to resolve. We are looking at future evolution of *glExec*, a component of the gLite Middleware [7] that can be used to authenticate a real user using the proper GSI proxy and even change the UID to the real user. In order to achieve our goal we need to understand if *glExec* can be used on the worker nodes, much the same way it is designed to work at the level of Computing Elements, where real users will be authorized prior to job execution once the *starter* daemon passes the correct user proxy to *glExec*.

Limitation

The Condor glide-in mechanism was easy to implement, but this solution is not general enough. The UDP based communication, which the Condor daemons use to talk to each other, does not work over the Wide Area Network (WAN). The mechanism also requires bi-directional traffic that fails to work over the firewalls. All the above dictate that a GlideCAF must be installed for each and every Grid site of interest in order to access the site resources. Although this is attainable for major Grid sites, like the Italian Tier-1 centre in Bologna, accessing a large number of small sites poses a severe maintenance overhead. Moreover, for each site, the CDF software must be distributed to the worker nodes by means of a shared file system.

To overcome the above limitations, three tools have been identified, (a) the newly released Generic Connection Brokering (GCB) to bridge firewalls and work over Wide Area Networks, (b) Parrot for software distribution, and (c) HTTP caches to limit the required network bandwidth. Please refer to [8] and [9] and references therein for more details. The final incarnation of the GlideCAF will take advantage of the above tools.

CONCLUSION

Continuous need for more computing power has pushed the CDF experiment to revise its computing model and adapt to the Grid computing paradigm. The first successful step in this direction was to add the Condor glide-in mechanism to a few dedicated Condor pools, which have access to shared resources, and build GlideCAFs. Several GlideCAFs are already in production which ensured a fast turn-around of resources for CDF. The paradigm shift has been completely transparent to end-users and all the high-level CDF specific features like sophisticated monitoring, scheduling policies continue to work natively. We are working on GlideCAF extensions in order to address the remaining issues.

REFERENCES

[1] CDF Collaboration, “The CDF II Technical Design Report”, FERMILAB-Pub-96/390-E (1996)

[2] The Condor project page, http://www.cs.wisc.edu/condor

[3] E. Lipeles, M. Neubauer, I. Sfiligoi, F. Wuŕthwein : The Condor based CDF CAF, CHEP 2004 proceedings, Interleave 2004.

[4] The LCG Homepage, http://lcg.web.cern.ch/LCG/

[5] The OSG Homepage, http://www.opensciencegrid.org

[6] S. Sarkar *et al.* : CDF Way to the Grid, EPS-HEPP 2005 proceedings, Lisbon 2005.

[7] The gLite project page, http://glite.web.cern.ch/glite/

[8] M. Norman *et al.* : OSG-CAF - A single point of submission for CDF to the Open Science Grid, CHEP 2006 proceedings, Mumbai 2006.

[9] C. Moretti, I. Sfiligoi, D. Thain : Transparently Distributing CDF Software with Parrot, CHEP 2006 proceedings, Mumbai 2006.