CDF/DOC/COMP_UPG/PUBLIC/7630
Version 1.0
May 6, 2005

GlideCNAF : A Purely Condor Glide-in Based CDF Analysis Farm

Subir Sarkar[1], Igor Sfiligoi[2]

Abstract

A purely Condor Glide-in based CDF Analysis Farm has been built that uses the Grid resources available at the Tier1 Farm at CNAF, Bologna. In this article we shall discuss about the urgent need of CDF for more processing power and argue that as significant upgrade of the dedicated farms is becoming unrealistic, we must start using the shared resources. In this context, we shall show that using Condor-G and Condor Glide-in is the easiest and most efficient way to access Grid resources. This is all the more true for farms that already use Condor as the batch system. The nascent farm, known as GlideCNAF, is about to enter into the production phase.

1 Introduction

The CDF experiment has been collecting Physics data since 2000. Improvements in accelerator running as well as detector and trigger conditions have resulted in steady increase in data taking efficiency over the years. Analysis of an ever growing volume of data and the need for producing yet larger samples of MC events require consistent growth of computing resources over time.

CDF produces $\sim$ 100 TB of data per year. Figure 1 shows the performance of the Tevatron accelerator as well as the data taking efficiency of the CDF experiment during 2005, while Figure 2 extrapolates the present performance of the accelerator for the years to come (until 2009). This gives us an idea about how CDF computing power should evolve to cope up with the growing need. Clearly, computing resource is always at premium in any running experiment like CDF.

[1]INFN-CNAF, Bologna & INFN-Roma1
[2]INFN-LNF, Frascati

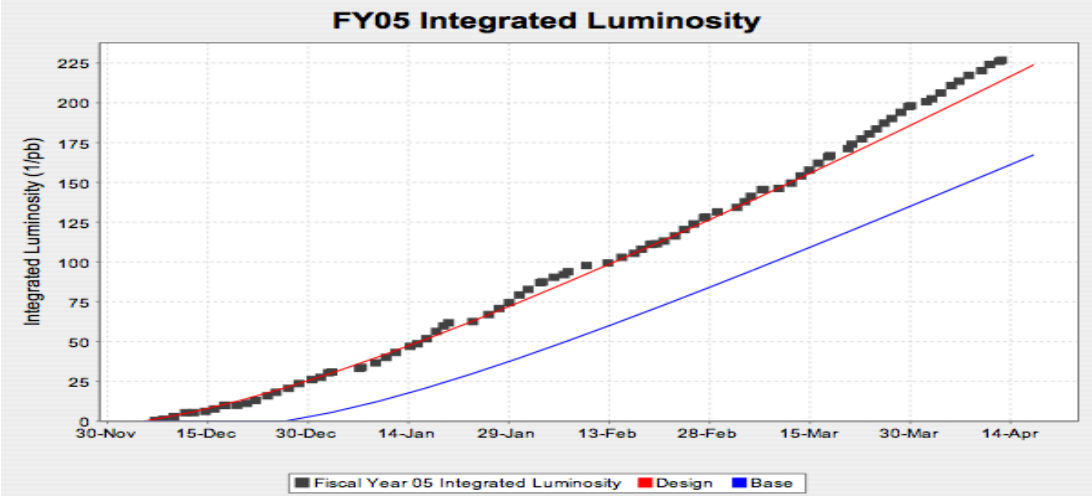

Figure 1: Integrated Luminosity that the Tevatron accelerator has delivered and that collected by the CDF experiment so far in 2005. The average data taking efficiency is 85% with scope of further improvement in future.

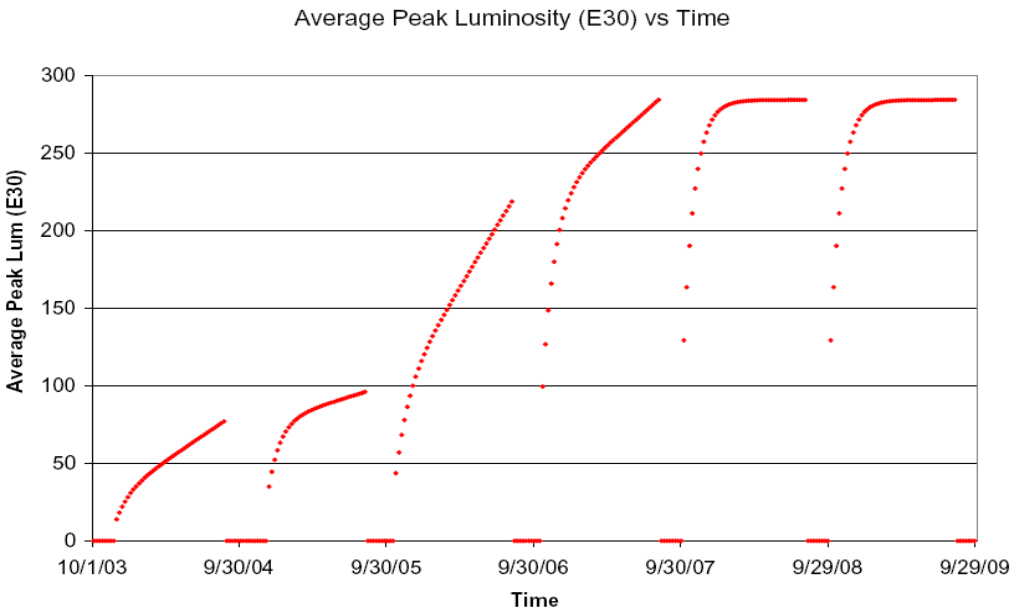

Figure 2: Design luminosity that the Tevatron accelerator is expected to deliver in the coming years during the lifetime of the experiment.

CDF has been successfully maintaining its own global computing environment with 9 Decentralized Analysis Farms (dCAFs [2]) all over the globe and with about 55% of its total computing power ($\sim$ 2.3 THz) at remote sites. The present CDF global computing heavily depends on dedicated resources. However, the "Dedicated Pool" model is no longer favoured. Many sites have expressed increasing desire to move away from dedicated resources following the Global HEP computing strategy of moving towards Grid Computing. Expansion of dedicated resources is increasingly becoming unrealistic and for some dCAFs might simply be impossible due to local policy constraints. For example, CnafCAF is a Condor [1] based dCAF [4] running with the following limited resources,

- CPU: $\sim$490 KSI2K
- Storage: 32 TB
- CDF software and tools are on AFS, the server being at CNAF which guarantees efficient access.

Significant expansion of the dedicated farm at CNAF at par with the increase in data volume and requirement of larger MC samples is simply beyond consideration on longer term. New CPU power will only be added to the global pool, more storage will, however, continue to be available to individual experiments. Tables 1 and 2 show a plausible scenario of evolution of CPU power and storage for CDF at CNAF.

Year	CDF (KSI2K)	T1 (KSI2K)	CDF Share (%)
2004	86		
2005	494.5	1300	38
2006	903	1800	50
2007	1032	2400	43
2008	1161	6300	18
2009	1290	10500	12
2010	1419	18000	8

Table 1: A proposed evolutionary path of CDF share of processing power at CNAF during the lifetime of CDF.

It is clear from the above that integrating CDF within the framework of Grid Computing and building a shared and opportunistic dCAF, hopefully keeping all the CDF/CAF specific advantages is the natural step forward. This will open up a huge potential for availability of new processing power (about 3.5 THz at CNAF itself). We can also take advantage of all the new innovations in global computing this way.

CDF is actively engaged in the following areas of development, ordered in terms of maturity, in order to integrate its computing model into Grid computing smoothly.

- Condor Glide-ins + CAF software (GlideCAF)

Year	CDF (TB)	T1 (TB)	CDF Share (%)
2004	10		
2005	32	200	16
2006	90	850	11
2007	130	1500	9
2008	170	3500	5
2009	210	5600	4
2010	250	9000	3

Table 2: A proposed evolutionary path of CDF Storage at CNAF during the lifetime of CDF.

- CAF interface to the Grid tools (GridCAF)
- SAMGrid
- Direct use of Grid tools

In this article, we shall discuss about the successful effort with Condor Glide-ins and demonstrate that a mature and robust dCAF that uses Grid resources can indeed be built with moderate effort.

2 Condor-G and Glide-ins

The Condor way of job submission to Grid (Globus Universe) is via Condor-G. Condor-G jobs can be configured to bypass the Resource Broker (RB) and directly reach a Grid site Gatekeeper or Computing Element (CE). The Condor Glide-in tool was created to ease the process of preparing Condor-G job submission with arbitrary configuration supported by Condor. When a Condor Headnode submits Glide-ins to a Grid site Gatekeeper, the Gatekeeper distributes those jobs to the underlying batch system (pbs, lsf, condor etc.) with the help of `globus-job-manager` processes. Once the Condor-G jobs start at the local batch system, the Worker Nodes (WN) install the Condor daemons on the fly from a well known location (AFS in case of GlideCNAF) and start them. As the Condor daemons start on the WNs, they communicate back to the submission machine or Headnode and become part of a local Condor pool. The WNs establish direct connection to the Headnode and jobs are pulled from the local Condor queue. Figures 3 and 4 show how Grid resources become part of a Condor pool that has dedicated WNs as well. A Condor pool can also be built purely out of Grid resources, as is the case for GlideCNAF. A Condor based CAF Headnode close to the site Gatekeeper is all that is needed. GlideCNAF uses the shared Tier1 resources at CNAF that has Fair-share policy in place, i.e before the LHC turn-on and except during occasional LHC Data Challenges, CDF can hope to have plenty of resources.

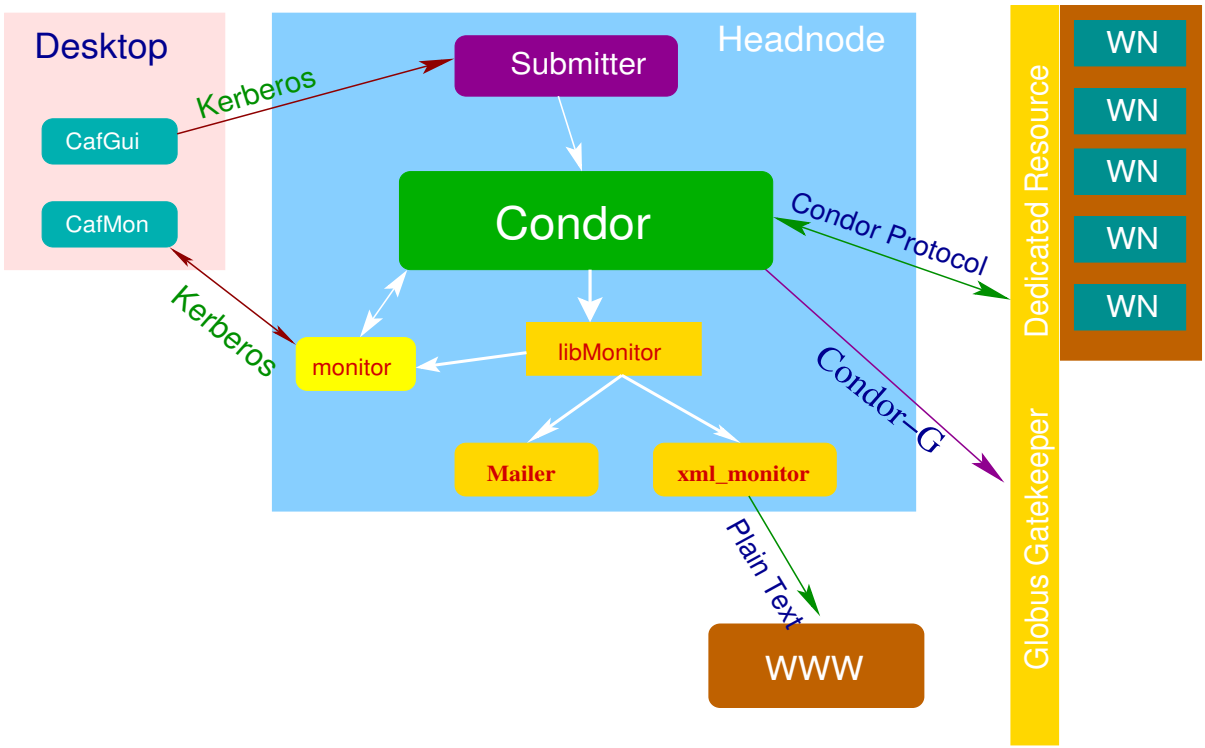

Figure 3: Submit Condor Glide-ins to the Grid site Gatekeeper from the CondorCAF Headnode. The Gatekeeper starts a `globus-job-manager` process for each job submitted, which in turn submits the job to the local batch system, and monitors the status of the job in the batch queue.

2.1 Authentication

Glide-ins are submitted by the CDF Grid user, with only one GSI certificate. As a consequence all user jobs run under the same UID (e.g cdf004).

Although, this is not the way Grid was supposed to work, we have sound technical reasons to support this way of functioning.

- We keep the policy decision in our hands instead of relying on the CDF VO and the local batch system policies on the Grid world. Note that the VO Policy is yet to be implemented. For a running experiment like CDF this poses a problem. Fortunately, a Glide-in based CondorCAF which supports sophisticated user accounting overcomes this short-coming easily.

- Fewer Glide-ins need to be submitted

At present we use the Kerberos CAF service principal (e.g `sarkar/cdf/testcnaf@FNAL.GOV`) to generate a k5→VOMS proxy that requires the following step,

```
> kinit -F -k -t /etc/cdfcaf.keytab sarkar/cdf/testcnaf@FNAL.GOV
```

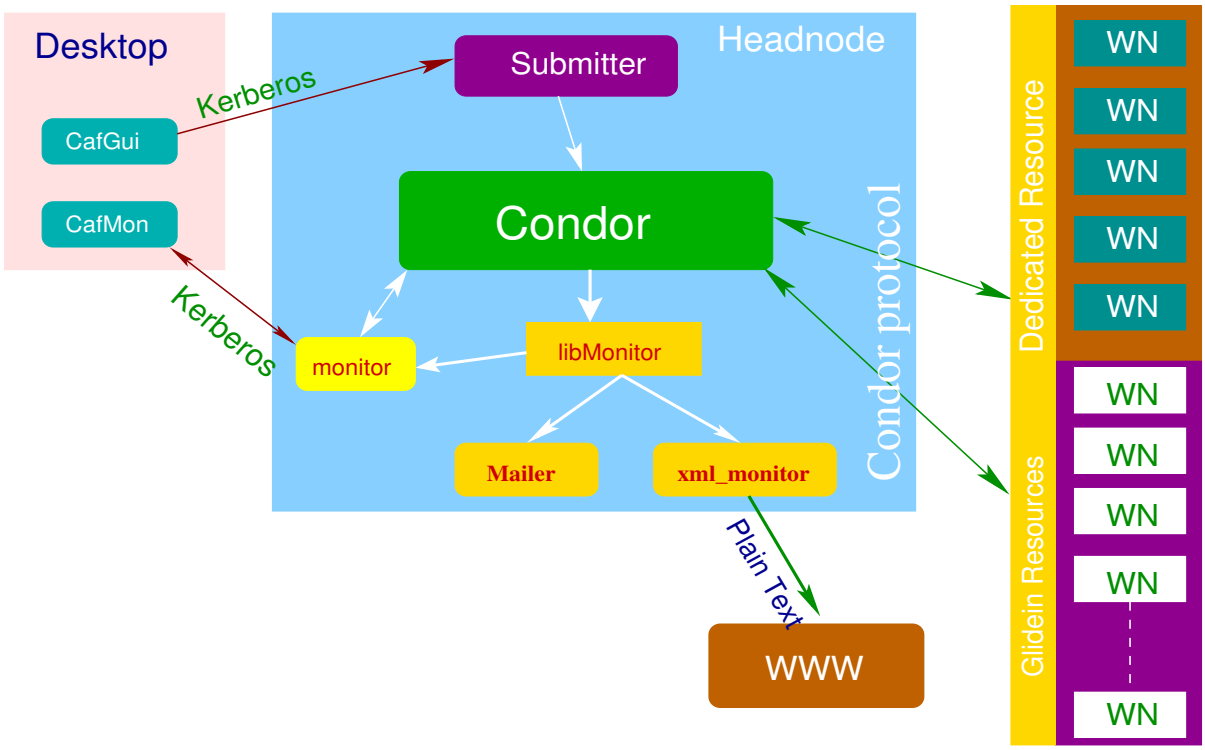

Figure 4: When the Glide-ins start on the WNs, resources become part of the local Condor pool temporarily. The Headnode is notified about availability of new nodes and normal (Vanilla Universe) jobs are subsequently pulled from the queue using usual Condor protocol. All the CAF components continue to work as is.

```
> kx509
> kxlist -p
> voms-proxy-init -noregen -voms cdf
```

For more detail please refer to 5. We are exploring the possibility of using the host certificate for the Headnode which will be more meaningful and Grid compliant and hopefully more maintainable in the future. Note that the Condor Glide-in based farm does not mandate that the Fermilab Robots Certificates of the users derived from the Kerberos CAF Service principals must exist on the CDF VOMS server.

2.2 GlideCNAF

GlideCNAF is a purely Condor Glide-in based dCAF which is into beta test now and is expected to go into production mode soon. GlideCNAF uses Cern Scientific Linux (SLC) based Grid resources at Tier1@CNAF which uses LSF as the local batch system. Only a small change over the Condor based production CAF software was needed for GlideCNAF to function like a native

Condor farm. A dedicated Headnode has been installed at CNAF to guarantee the most efficient I/O between the Headnode and the Gatekeeper dedicated to CDF. The Headnode also acts as a Grid User Interface (UI).

- GlideCNAF uses dedicated storage. We keep CDF user policy in our hands. The farm is robust and all the original CAF specific tools, e.g monitoring, group accounting etc. work without any modification. GlideCNAF is opportunistic and can, in principle, use all the idle CPU cycles of a Grid site.
- GlideCNAF is configured to run 600 jobs simultaneously at present. In the best scenario this means we have about 1.1 THz of CPU power at our disposal.
- GlideCNAF is a dCAF like CnafCAF. Whatever can be run on CnafCAF can be run on GlideCNAF as well.
- GlideCNAF expands its capacity dynamically as and when new user jobs are discovered. Just looking at the current number of VMs in the monitoring page will not tell the real potential of the farm.
- CafMon/CafWWW work the same way as they do for any other dCAF

A detailed description about how to use the farm, in particular how to deal with an environment, where WNs are on the Grid side, unknown to us will be given in Appendix 5.

3 Acknowledgments

Many people from both sides of the Atlantic helped us make GlideCNAF a reality. We thank them all for their support. We acknowledge, in particular, significant help from Luca dell'Agnello, Andrea Chierici and Alessandro Italiano with Tier1 resource usage; Vincenzo Ciaschini and Daniele Cesini regarding VOMS related issues; Dan Bradley for Condor-G and Condor Glide-in related support. We also thank all the beta testers of the farm.

4 Conclusions

A purely Condor Glide-in based CDF Analysis Farm, GlideCNAF, has been built at CNAF that uses the common resources available at the Tier1 Farm. CDF has already started looking beyond dedicated dCAFs for new computing resources, both for MC production and for analysis of ever growing data volume. Condor Glide-in based solution is straight-forward and robust. Moreover, sophisticated job monitoring and user accounting policy work in a native manner for a Glide-in based CAF. These areas are still under-developed in the Grid world. However, a Glide-in based CAF is understood to be best suited for Grid sites that strongly support a particular experiment, as is the case for CDF at CNAF. To exploit generic Grid sites, GridCAF is being developed in CDF.

5 Appendices

In this section all the technical details related to a Glide-in based CAF will be described. Since the CAF software required only trivial modifications, we'll mainly concentrate on the Glide-in related technicalities.

Appendix A: GlideCNAF Operational Issues

- New service principal: <user>/cdf/testcnaf@FNAL.GOV (needed in .k5login)
- To submit jobs, all you will need to do is to copy the cnaf section of your .cafrc (if you do not have one copy the official file ∼cdfsoft/dist/packages/CafUtil/development/cdf_gui/.cafrc), call it glide-cnaf (or invent a better name yourself) and replace

  ```
  host=cdfhead.cnaf.infn.it    by
  host=cdftest.cnaf.infn.it
  ```

  The GlideCNAF section will soon be added to the official .cafrc.

  Group accounting has been implemented in GlideCNAF (groups = common,italy). It is the same as in CondorCNAF.

- New environment variables have been introduced in the CAF Software in order to ease integration with Grid. For example,

  – If you use fcp in your shell script that is executed on the WNs (i.e started by CafExe), do, e.g
  ```
  > fcp -c $KRB5BIN_DIR/rcp -N src dest
  ```
  KRB5BIN_DIR is correctly configured by the dCAF administrator.

  – Similarly, you are encouraged to set the CDF software environment as,
  ```
  > source $CDFSOFT/cdf2.(c)sh
  ```
  instead of specifying the path specific to a dCAF.

  Note that the above environment variables may not yet be available on all the dCAFs.

- GlideCNAF monitor is at http://cdfmon.cr.cnaf.infn.it:8081/glidecaf/
- The following application level problems and the subsequent solutions were found on GlideCNAF:

  – MCProd problem : the full path name of a file to be opened by the Fortran code may become too long due to the fact that on the Grid side a unique directory name must be ensured for each job and that might cause the program to abort. Here is one possible way out,

* In the top dir of `mcProduction` test release, do
  `> ln -sf cesData cesdata`
* Try to use relative path names, namely, set
  `export JOB_OUTPUT_DIR=.`
  `export WORK_DIR=.`
  in mcProduction/scripts/MCProd.

– The C-shell script `submit_BMC` does not run on GlideCNAF and returns with the following error:
`SHELL: Undefined variable.`

The easiest solution is to use `tcsh`. Scientific Linux seems to be stringent on this issue for reasons not yet known. Nevertheless, the recommended solution is to use `bash` shell scripts at GlideCNAF.

Appendix B: Installation of a GlideCAF

A GlideCAF can be easily installed using the following prescription,

- Make sure that you can access the Gatekeeper as well as the local batch system of a grid site of your choice. With a valid proxy (preferably k5→VOMS proxy as discussed in appendix 5) you can try the commands like the following,

```
> globus-job-run ce02-lcg.cr /bin/hostname
ce02-lcg.cr.cnaf.infn.it
> globus-job-run ce02-lcg.cr:2119/jobmanager-lcglsf \
  -queue cdf /bin/hostname
wn-04-04-16-a.cr.cnaf.infn.it
```

  You must ensure the above before proceeding any further.

- Install a CondorCAF Headnode from scratch following instructions given in reference [5]. Note that Condor from version 6.7.7 works correctly for a Glide-in based CAF.

- Adapt the Condor configuration files for Condor-G submission with Kerberos and GSI based authentication

- Create the Condor-G submission file and the Glide-in related configuration files. You might need to slightly manually modify the Condor-G submission and the configuration files.

- Send Glide-ins as Condor-G jobs to the Gatekeeper

- Check the status of the Glide-in jobs,

  - `condor_q -globus` shows status of Glide-in submission, `SUBMITTED`, `PENDING`, `ACTIVE` etc.
  - `condor_status -master` shows the VMs as they are added to the pool
  - Condor-G submission log file is also useful to track down problems

  When the Glide-ins start running on the remote site and the remote VMs become available on the Condor pool, submit test jobs using CAF tools

- Install the standard CondorCAF monitor following the CafCondor installation instruction.

The following sections elaborate on the points noted above.

Appendix C: Glide-ins and Condor Configuration

Glide-ins require authentication related configuration to be introduced in the Condor configuration files. The following was added in `$CONDOR_DIR/dist/etc/condor_config` to switch on authentication.

```
# Switch on authentication
SEC_DEFAULT_AUTHENTICATION = REQUIRED
SEC_DEFAULT_AUTHENTICATION_METHODS = KERBEROS,GSI
SEC_DEFAULT_ENCRYPTION = OPTIONAL
SEC_DEFAULT_INTEGRITY = PREFERRED

SEC_READ_AUTHENTICATION = OPTIONAL
SEC_CLIENT_AUTHENTICATION = OPTIONAL
SEC_READ_ENCRYPTION = OPTIONAL
SEC_CLIENT_ENCRYPTION = OPTIONAL
SEC_READ_INTEGRITY = OPTIONAL
SEC_CLIENT_INTEGRITY = OPTIONAL
```

The configuration file local to the Headnode contains the detail as given below,

```
GSI_DAEMON_DIRECTORY=/afs/infn.it/project/cdf/glidein/dist

# Grid Certificate directory (standard)
GSI_DAEMON_TRUSTED_CA_DIR=/etc/grid-security/certificates/

# Gridmap file that relates the proxy certificate to the user 'condor'
GRIDMAP=$(GSI_DAEMON_DIRECTORY)/grid-mapfile
```

```
SEC_DEFAULT_AUTHENTICATION = REQUIRED
SEC_DEFAULT_AUTHENTICATION_METHODS = KERBEROS,GSI

# k5->VOMS Proxy file
GSI_DAEMON_PROXY = /cdfcaf/tickets/x509_service_proxy

# Use Condor-G GridMonitor
ENABLE_GRID_MONITOR = TRUE

# A single Gatekeeper accepts a maximum of 600 Condor-G jobs
GRIDMANAGER_MAX_SUBMITTED_JOBS_PER_RESOURCE = 600

# Submit Condor-G jobs to the gatekeeper slowly
GRIDMANAGER_MAX_PENDING_SUBMITS_PER_RESOURCE = 2
```

In addition to the above `create_glidein.sh` automatically creates a configuration file for the Glide-ins which has a similar authentication block as above except that the GSI authentication is tried first,q

```
SEC_DEFAULT_AUTHENTICATION_METHODS = GSI,KERBEROS
```

In order for Condor Computing on Demand (Cod) to work properly, the Glide-in side configuration must contain the following line,

```
VALID_COD_USERS = condor,cafmon
```

Appendix D: `create_glidein.sh`

The following script first copies the Glide-in binaries for the correct architecture from the official repository and creates a local distribution for more efficient access from the WNs. The script then creates the configuration that will be used by the Glide-in daemons and finally prepares the Condor-G submission description file. The script is general enough and should at a different site with only trivial. Execute `create_glidein.sh` after starting Condor on the Headnode. Note that, Condor 6.7.7 is the first version where GlideCNAF works fully.

```
#!/bin/sh
#-------------------------------------------------------------
# Prepare Condor-G submission files with condor_glidein
#
# v0.5 19/04/2005 - Igor & Subir
# -------------------------------------------------------------
```

```
DEBUG=0
if [ $# -lt 1 ]; then
  echo Usage: ./create_glidein.sh distDir Condor_version[D=6.7.7]
  exit 1
fi
distDir=$1
if [ $# -lt 2 ]; then
  echo Using default version: 6.7.7
  version="6.7.7"
else
  version=$2
fi

# Check if Condor itself is running
condor_running=‘condor_status -master‘
if [ "$condor_running" == "" ]
then
  echo "Condor must be running! exiting ..."
  if [ "$DEBUG" -eq 1 ]; then
    ps -ef | egrep condor
  fi
  exit 2
fi

# Environment
GLIDEIN_DIR="/afs/infn.it/project/cdf/glidein/${distDir}"
GLIDEIN_LOCAL=.
GLIBC_VERSION=glibc2.3
GLIDEIN_ARCH=${version}-i686-pc-Linux-2.4-${GLIBC_VERSION}
GLIDEIN_GATEKEEPER=ce02-lcg.cr.cnaf.infn.it:2119/jobmanager-lcglsf
SUFFIX=infn
GLIDEIN_USER=condor

# Create the glidein directory, if needed
mkdir -p $GLIDEIN_DIR
cd $GLIDEIN_DIR

# Fetch the tarball and unpack it properly
wget http://cs.wisc.edu/condor/glidein/binaries/$GLIDEIN_ARCH.tar.gz
mkdir -p $GLIDEIN_ARCH
cd $GLIDEIN_ARCH
```

```
tar xvzf ../$GLIDEIN_ARCH.tar.gz
cd $GLIDEIN_DIR
rm -f $GLIDEIN_ARCH.tar.gz

# Create grid-mapfile
echo \"‘grid-proxy-info -issuer‘\" $GLIDEIN_USER > grid-mapfile

# Generate the config file and the startup script
if [ "$DEBUG" -eq 1 ]; then
 echo INFO. Generate the config file and the startup script
fi
condor_glidein -anybody -suffix $SUFFIX -genconfig -genstartup

# Install the daemon startup script.
chmod a+x glidein_startup.$SUFFIX

# It seems condor_glidein script works slightly differently
# for 6.7.3 and for >= 6.7.6

if [ ‘‘$version’’ == ‘‘6.7.3’’ ]; then
var=$(cat <<SETVAR
/bin/sh\n\nexport _CONDOR_GSI_DAEMON_PROXY=\$X509_USER_PROXY
SETVAR)
  perl -pi -e ‘‘s#/bin/sh#$var#’’ glidein_startup.$SUFFIX
fi
mv glidein_startup.infn $GLIDEIN_ARCH/glidein_startup

# Install the condor_config.
mv glidein_condor_config.$SUFFIX glidein_condor_config

cat >> glidein_condor_config <<EOF
VALID_COD_USERS = condor,cafmon

GSI_DAEMON_DIRECTORY=$GLIDEIN_DIR
GSI_DAEMON_TRUSTED_CA_DIR=/etc/grid-security/certificates/
GRIDMAP=\$(GSI_DAEMON_DIRECTORY)/grid-mapfile
SEC_DEFAULT_AUTHENTICATION = REQUIRED
SEC_DEFAULT_AUTHENTICATION_METHODS = GSI
EOF

# Now generate the submit file used to run Glide-ins.
```

```
if [ "$DEBUG" -eq 1 ]; then
  echo INFO. Generate the Condor-G submit file  to run Glide-ins.
fi
condor_glidein \
  -suffix $SUFFIX \
  -basedir $GLIDEIN_DIR \
  -localdir $GLIDEIN_LOCAL \
  -gensubmit \
  -runonly \
  -anybody \
  -arch $GLIDEIN_ARCH  \
  -setup_jobmanager jobmanager-fork \
  -queue cdf
  $GLIDEIN_GATEKEEPER

# Now you have a submit file named glidein_run.submit.$SUFFIX.
# You can submit this to Condor-G and it will launch glidein daemons
# However, we need some customisation

perl -pi \
 -e 's#_condor_NUM_CPUS=1#_condor_NUM_CPUS=2#;' \
 -e 's#_condor_START=True#_condor_START=(VirtualMachineID == 2)#;' \
 -e 's#cdfcaf\@cdftest.#subir\.sarkar\@#' glidein_run.submit.$SUFFIX

# For debugging purposes create
cp glidein_run.submit.$SUFFIX glidein_run.submit.$SUFFIX.dbg

# Again, some customisation required, Condor-G GridMonitor
# won't work without stream_output = False; stream_error  = False

var=$(cat <<SETVAR
output=glidein_$SUFFIX.output
error=glidein_$SUFFIX.error
log=glidein_$SUFFIX.log
should_transfer_files=YES
when_to_transfer_output=ON_EXIT
stream_output = False
stream_error  = False
+Owner = undefined

Queue
```

```
SETVAR)

perl -pi -e "s#Queue#$var#" glidein_run.submit.$SUFFIX.dbg

# Since we are very much in development, we need to change
# the Condor version. Keep a link for easy maintenance
cd $GLIDEIN_DIR/..
if [ -e dist ]; then
  rm dist
fi
ln -s $distDir dist

exit 0
```

Appendix E: Condor-G Submission Script

As already mentioned, the Condor-G submission script is automatically created and adjusted for a particular site by `create_glidein.sh`. The submission script used for GlideCNAF is shown below. When Glide-ins are submitted, the following file works as a template if we submit to a Gatekeeper other than the default one (`ce02-lcg.cr.cnaf.infn.it`) or submit more than one jobs at a time.

```
# For formatting convenience we use
# GLIDEIN_DISTDIR=/afs/infn.it/project/cdf/glidein/v6.7.7
# and break long lines below
Universe = Globus
Executable =
$(GLIDEIN_DISTDIR)/6.7.7-i686-pc-Linux-2.4-glibc2.3/glidein_startup
Arguments = -dyn -f

Environment =
CONDOR_CONFIG=$(GLIDEIN_DISTDIR)/glidein_condor_config;
_condor_CONDOR_HOST=cdftest.cnaf.infn.it;
_condor_GLIDEIN_HOST=cdftest.cnaf.infn.it;
_condor_LOCAL_DIR=.;
_condor_SBIN=$(GLIDEIN_DISTDIR)/6.7.7-i686-pc-Linux-2.4-glibc2.3;
_condor_CONDOR_ADMIN=subir.sarkar@cnaf.infn.it;
_condor_NUM_CPUS=2;
_condor_UID_DOMAIN=cr.cnaf.infn.it;
_condor_FILESYSTEM_DOMAIN=cr.cnaf.infn.it;
_condor_MAIL=/bin/mail;
_condor_STARTD_NOCLAIM_SHUTDOWN=1200;
```

```
_condor_START=(VirtualMachineID == 2)

Transfer_Executable = False
GlobusRSL = (queue=cdf)
GlobusScheduler = ce02-lcg.cr.cnaf.infn.it:2119/jobmanager-lcglsf
output=glidein_infn.output
error=glidein_infn.error
log=glidein_infn.log
should_transfer_files=YES
when_to_transfer_output=ON_EXIT
stream_output = False
stream_error  = False
+Owner = undefined
Queue
```

Appendix F: `submit_glideins.sh`

`submit_glideins.sh` does the following; it,

- accepts the name of the site Gatekeeper and the number of Glide-ins to be submitted as input
- removes those Glide-ins that were put on hold by Condor for some reason (mainly authentication related problems)
- releases any user jobs that were held
- submits the required number of Glide-ins if there is really a need for new submission, i.e if it discovers new user jobs on the Condor queue.

```
#!/bin/sh -f
# ------------------------------------------------------------------
#
# Submit Condor-G jobs to the same Gatekeeper.
# A crontab calls the script every n mins which submits the missing
# glidein jobs when new user jobs are discovered.
#
# Usage: $GLIDEIN_DIR/submit_glideins.sh  gl_min gatekeepr
#
# v0.6 05/05/2005 - Subir
# ------------------------------------------------------------------

GLIDEIN_DIR=/afs/infn.it/project/cdf/glidein/dist
```

```
SUFFIX=infn
DEF_GATEKEEPER=ce02-lcg.cr.cnaf.infn.it:2119/jobmanager-lcglsf
DEF_CDFQUEUE=cdf

QUEUE_OUTPUT_FILE=/tmp/condor_q_$$.output
GLIDEIN_JOB_FILE=/tmp/glidein_run_$$.submit

DEBUG=0

# Input arguments
gatekeeper=$DEF_GATEKEEPER
gl_min=0
gl_max=2000                  # tier 1 LSF farm@CNAF
cdfqueue=$DEF_CDFQUEUE

if [ $# -gt 0 ]; then
  gatekeeper=$1
fi
if [ $# -gt 1 ]; then
  gl_min=$2
fi
if [ $# -gt 2 ]; then
  gl_max=$3
fi
if [ $# -gt 3 ]; then
  cdfqueue=$4
fi

cleanup()
{
  if [ -e $QUEUE_OUTPUT_FILE ]; then
    if [ "$DEBUG" -eq 1 ]; then
      echo INFO. delete $QUEUE_OUTPUT_FILE
    fi
    rm $QUEUE_OUTPUT_FILE
  fi
  if [ -e $GLIDEIN_JOB_FILE ]; then
    if [ "$DEBUG" -eq 1 ]; then
      echo INFO. delete $GLIDEIN_JOB_FILE
    fi
    rm $GLIDEIN_JOB_FILE
```

```
  fi
}

# Check if Condor itself is running
condor_running=‘condor_status -master‘
if [ "$condor_running" == "" ]; then
  echo "ERROR. Either Condor is NOT running or "
  echo "condor_master is not available ! exiting ..."
  if [ "$DEBUG" -eq 1 ]; then
    ps -ef | egrep condor
  fi
  exit 1
fi

echo ----------------------------------------------------
echo === ‘date‘ ====

# Cache condor_q output, do not call it many times within
# a small time interval that may hang condor itself if the
# queue is long

condor_q > $QUEUE_OUTPUT_FILE
status_code=$?
if [ "$status_code" -ne 0 ]; then
  echo "ERROR. condor_q command failed! exiting ...."
  cleanup
  exit $status_code
fi

# Remove glide-in jobs on H/C
for id in ‘cat $QUEUE_OUTPUT_FILE | grep glidein_startup \
 | awk ’{if ($6 == "H" || $6 == "C") print $1}’‘
do
  echo "INFO. removing H/C glideins, id=$id ..."
  condor_rm $id
  sleep 2  # not sure if it is actually needed
done

# Find number of glidein daemons running/queued
gl_inq=‘cat $QUEUE_OUTPUT_FILE | grep glidein_startup \
 | awk ’BEGIN {s=0} {if ($6 == "R" || $6 == "I") s++} \
```

```
        END {print s}’‘

# Find how many use jobs are running
ujob_r=‘cat $QUEUE_OUTPUT_FILE | grep -e CafExe -e sam \
 | awk ’BEGIN {s=0} {if ($6 == "R") s++} \
        END {print s}’‘

# Find how many use jobs are queued
ujob_i=‘cat $QUEUE_OUTPUT_FILE | grep -e CafExe -e sam \
 | awk ’BEGIN {s=0} {if ($6 == "I") s++} \
        END {print s}’‘

# Find how many use jobs are held
ujob_h=‘cat $QUEUE_OUTPUT_FILE | grep -e CafExe -e sam \
 | awk ’BEGIN {s=0} {if ($6 == "H") s++} \
        END {print s}’‘

echo "INFO. User jobs, RUNNING = $ujob_r, "
echo "                   IDLE = $ujob_i, "
echo "                   HELD = $ujob_h ..."

# If held jobs are found release them. However, since
# we do not want to call condor_q again, the released
# jobs will be considered diring the next iteration
if [ "$ujob_h" -gt 0 ]; then
  echo "INFO. Releasing all the jobs ... "
  echo "will be considered during the next cycle"
  condor_release -all
  # Wait for condor_release to finish and the
  # jobs to reappear as I/R
  sleep 10
fi

# Total user jobs
ujob_t=‘expr $ujob_r + $ujob_i‘

# Check if re-submission is needed, first with user job
gl_req=‘expr $ujob_t - $gl_inq‘
if [ "$gl_req" -le 0 ]; then
  echo "INFO. glideins($gl_inq) >= user jobs($ujob_t) ..."
  # Now with minimum glidein requirement
```

```
  gl_req=‘expr $gl_min - $gl_inq‘
  if [ "$gl_req" -le 0 ]; then
    echo "INFO. Nothing to be done! "
    echo "no of glideins to be submitted = $gl_req"
    cleanup
    exit 1
  fi
else
  if [ "$gl_req" -ge "$gl_max" ]; then
    gl_req=‘expr $gl_max - $gl_inq‘
    if [ "$gl_req" -le 0 ]; then
      echo "INFO. Nothing to be done! "
      echo "no of glideins to be submitted = $gl_req"
      cleanup
      exit 2
    fi
  fi
fi
nsub=$gl_req

# Create proper submit script from the template
cp $GLIDEIN_DIR/glidein_run.submit.$SUFFIX.dbg \
   $GLIDEIN_JOB_FILE
perl -pi \
  -e "s#$DEF_GATEKEEPER#$gatekeeper#;" \
  -e "s#queue=$DEF_CDFQUEUE#queue=$cdfqueue#;" \
  -e "s#Queue#Queue $nsub#" $GLIDEIN_JOB_FILE

# Now submit
echo "INFO. Submit $nsub condor_glidein daemons ..."
echo
if [ "$DEBUG" -eq 1 ]; then
  cat $GLIDEIN_JOB_FILE
fi
condor_submit $GLIDEIN_JOB_FILE

# Clean up allocated resources (e.g unlink files)
cleanup

exit 0
```

Appendix G: Crontabs

The Headnode running CondorCAF needs to do the following in order for GlideCNAF to work; it

- looks for new user jobs in the CondorCAF queue and if required, automatically submits Glide-ins,
- refreshes GSI proxy automatically so that the Glide-ins running on the WNs can authenticate themselves

The above is achieved with the help of cron jobs as given below:

```
5 */8 * * * /cdfcaf/bin/signin.sh \
  1>> /tmp/cron_signin.log 2>> /tmp/cron_signin.log
*/5 * * * * cd /cdfcaf/log; /cdfcaf/bin/submit_glideins.sh 1 0 0 \
  1>> /tmp/cron_glidein.log 2>> /tmp/cron_glidein.log
```

Renewal of k5→VOMS Proxy

The CDF Virtual Organisation (VO) [3] is exclusively based on the Virtual Organisation Membership Service (VOMS) and is hosted on a VOMS server at CNAF (`voms.cnaf.infn.it`). CDF users are added to the VOMS server on request using a certificate issued by a trusted authority. The X509 protocol was develop to convert a Fermilab Kerberos proxy into a full-fledged Grid proxy, the so called k5→VOMS Proxy.

We follow the same model of proxy renewal as that used by CAF for Kerberos. `cdfcaf` renews the Kerberos proxy of a CAF service principal first, then converts the Kerberos proxy to a Grid proxy using the X509 protocol and finally creates a new VOMS proxy. The following scripts demonstrates all the steps of proxy renewal.

```
#!/bin/sh

DEBUG=0
PROXY_USER=sarkar/cdf/testcnaf@FNAL.GOV
CONDOR_HOME=/cdfcaf/condor

source $CONDOR_HOME/sh/glidein_setup.sh

k5Proxy()
{
  if [ ‘‘$DEBUG’’ -eq 1 ]; then
    echo kinit -F -k -t /etc/cdfcaf.keytab $PROXY_USER
```

[3] Abstract entity grouping Users, Institutions and Resources (if any) in the same administrative domain [3]

```
  fi
  kinit -F -k -t /etc/cdfcaf.keytab $PROXY_USER
  kx509
  kxlist -p            # Write $X509_USER_PROXY
}

vomsProxy()
{
  if [ ‘‘$DEBUG’’ -eq 1 ]; then
    echo \$X509_USER_PROXY=$X509_USER_PROXY
  fi
  voms-proxy-init -cert $X509_USER_PROXY \
                  -key  $X509_USER_PROXY --voms cdf
                  # Overwrites $X509_USER_PROXY
  voms-proxy-info -all
}

echo == ‘date‘ ==

# Get x509 proxy
k5Proxy

# In the unlikely event
while [ ‘‘$?’’ -ne 0 ]
do
  echo ‘‘WARNING. Problems with k5 proxy generation, ‘‘
  echo ‘‘ wait 1 min and retry ...’’
  sleep 60;
  k5Proxy
done

# Now refresh k5->VOMS Proxy
vomsProxy

# Return status code of VOMS Proxy renewal
exit $?
```

Glide-in Submission Driver Script

The following is an example of a driving script that calls `submit_glidein.sh` with proper input. This is given here just for completeness since `submit_glidein.sh` can also be called

from a cron job passing the input arguments directly.

```
#!/bin/sh
#

GATEKEEPERS[0]=ce02-lcg.cr.cnaf.infn.it:2119/jobmanager-lcglsf
GATEKEEPERS[1]=prod-ce-01.pd.infn.it:2119/jobmanager-lcglsf
GATEKEEPERS[2]=gridba2.ba.infn.it:2119/jobmanager-lcgpbs
GATEKEEPERS[3]=gridit-ce-001.cnaf.infn.it:2119/jobmanager-lcgpbs

gk_index=0
if [ $# -gt 0 ]; then
  gk_index=$1
  if [ ‘‘$gk_index’’ -lt 0 ]; then
    random_number=‘perl -e ’{print int(rand()*10)}’‘
    gk_index=${random_number}
    if [ ‘‘$gk_index’’ -lt 1 ]; then
       gk_index=2
    fi
  fi
fi

n_glideins=0
if [ $# -gt 1 ]; then
  n_glideins=$2
fi

n_glmax=2000
if [ $# -gt 2 ]; then
  n_glmax=$3
fi

gatekeeper=${GATEKEEPERS[$gk_index]}
echo $gatekeeper $n_glideins $n_glmax

CONDOR_HOME=/cdfcaf/condor
GLIDEIN_DIR=/afs/infn.it/project/cdf/glidein

source $CONDOR_HOME/sh/condor_setup.sh
source $CONDOR_HOME/sh/glidein_setup.sh

$GLIDEIN_DIR/submit_glideins.sh $gatekeeper $n_glideins $n_glmax
```

```
exit $?
```

Appendix H: Scalability Issues and Condor-G GridMonitor

GlideCAF software tries to ensure that the Condor-G GridMonitor starts correctly on the Headnode in order to reduce load on the Gatekeeper. If you still find a high load on the Gatekeeper caused by Glide-ins submission, check if the perl script `$CONDOR_HOME/dist/sbin/grid_monitor.sh` supports your jobmanager. For example, GlideCNAF uses a patched version of the script where `lcgpbs` and `lcglsf` job managers were added. Refer to [6] for a discussion about scalability of the Gatekeeper and the role of Condor-G GridMonitor.

References

[1] Condor Project Team. Offical Condor Homepage. *http://www.cs.wisc.edu/condor*. 1

[2] F. Wurthwein *et. al*. CDF CAF Design Document. *CDF Internal Note*, 5961. 1

[3] I. Foster, C. Kesselman and S. Tuecke. The Anatomy of the Grid. *International Journal of High performance Computing Applications*, 15, 3 (2001). 3

[4] I. Sfiligoi *et. al* . CAF based on Condor. *CDF Internal Note*, 7088. 1

[5] I. Sfiligoi *et. al* . Condor CAF Installation Manual. 2005. 5

[6] Alan Roy. Using Condor-G Effectively. *http://www.cs.wisc.edu/ roy/effective_condorg*, Version 4. 3