Sources for more information

Do you have questions about Condor-G?

There is a free mailing list for Condor users (not just Condor-G users) that you can join. To subscribe to condor-users send a message to majordomo@cs.wisc.edu with the body of:

```
subscribe condor-users
```

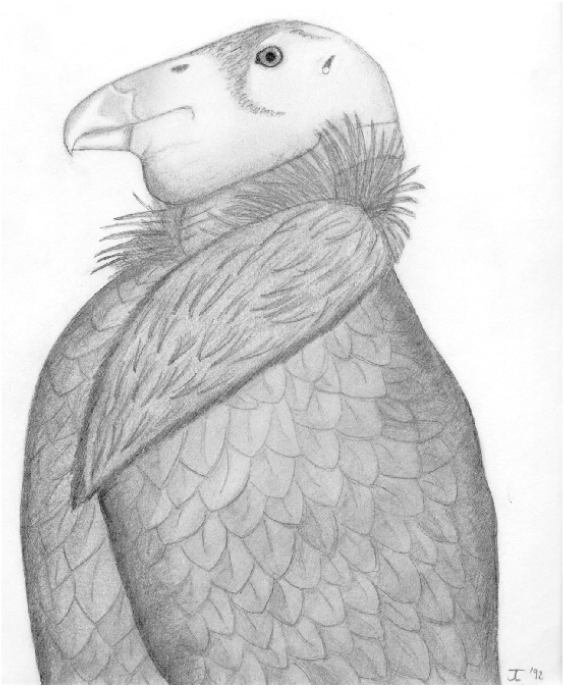
To unsubscribe from condor-users send a message to majordomo@cs.wisc.edu with the body of:

```
unsubscribe condor-users
```

You can send email to the Condor developers to ask for help. Send your email to condor-admin@cs.wisc.edu. We will do the best we can to answer your question, but we do not guarantee an answer. We are pretty good though.

Using Condor-G Effectively

A discussion by Alain Roy
Version 4
15-Apr-2004



This document may be found online:
http://www.cs.wisc.edu/~roy/effective_condorg
Send comments about this document to:
roy@cs.wisc.edu

Table of Contents

Miscellaneous Hints

Use large jobs
Think carefully about what kind of jobs you submit to a grid. Some people submit thousands of jobs that each take 30 seconds to run, then wonder why they see poor performance. Each job submission has a scheduling overhead, and you need to submit jobs that are sufficiently long to make this overhead worthwhile. Instead of submitting 1,000 jobs of 30 seconds each, try submitting 10 jobs of 3,000 seconds each by combining the work into larger portions. You will see significantly better throughput.

Synchronized clocks
Keep your clocks synchronized to standard time, and ensure the time zone is correct. If the clock on your submission machine is not close to the time on the gatekeeper host, you are likely to have security failures, because GSI expects synchronized clocks.

Be prepared for failure
We have worked very hard to make Condor-G as bulletproof as possible: it should deal well in the face of power outages, computer crashes, network congestion, disappearing file systems and more. But other components will fail because grid jobs live in a hostile environment, and these problems will occur in greater frequency than you would find in a local environment. When you build grid applications, accept that failures will happen and be flexible.

Forcing jobs to be removed
Sometimes, jobs cannot be removed with condor_rm. Condor-G puts them into the X state while trying to tell the remote grid site to clean up the job. This may take a very long time, and it may hang in some conditions, beyond Condor-G's control. Use "`condor_rm -forcex`" to forcibly remove the jobs. Note that this may leave orphaned jobs on the remote grid site.

Pack your bags: Bring your software with you
Many people find it simplest to pre-stage their software on a grid site, then use grid jobs to start up that software. This is fragile, prone to error, and will limit the number of grid sites that you can use.

Instead, submit a grid job that installs your software, or transfer it with every job. You will be sure that you have the correct version and that it is properly installed. If someone offers you a new grid site, you can take advantage of it with minimal effort. This democratization of computing is the greatest potential for grids: if you do not have access to lots of local computing power but you are prepared to use any grid sites that may be available, you can benefit.

Use the GridMonitor to increase scalability

The GridMonitor helps to make Condor-G jobs more scalable. In our experience, submitting large numbers of Globus jobs to a single gatekeeper can fail due to the load on the gatekeeper. In Globus 2.x, each job submission creates a process—whether the job is actively running or not—and that process, called the jobmanager, queries the underlying batch system every ten seconds. When submitting 300 simultaneous jobs, we have noticed a system load in excess of 400.

The Condor-G GridMonitor decreases this load in some circumstances by forcing Globus to not run the jobmanager when a job is not actively being run by the batch system. If you submit 300 jobs that the batch system can execute simultaneously, the GridMonitor will not help. If you submit 300 jobs that run for a few seconds each, the GridMonitor will not help. However, if you submit 300 jobs to a system that can run tens of jobs and they are long-running, you may see significant improvements in job throughput.

The GridMonitor does not help while jobs are running because Condor-G relies on the Globus jobmanager to stream standard output and error back. You can disable streaming by adding the following commands to your submit file:

1
```
stream_output = false
stream_error  = false
```

The files will be returned when your job finishes, and now the GridMonitor can replace the jobmanager for the entire time the job runs, except for a brief time at the end of the job's execution.

To use the GridMonitor, you must add two variables to your Condor configuration, then do a condor_reconfig:

2
```
GRID_MONITOR = $(SBIN)/grid_monitor.sh
ENABLE_GRID_MONITOR = TRUE
```

The first of these is probably already in your condor_config file, but the second one probably is not. When you make this change and submit a single job, you will see no effect, but only when you submit many jobs.

Introduction

We hope that you can use the information in this booklet to use Condor-G effectively. Although the basics of using Condor-G are not hard, there are a lot of tips and tricks that can help you to use Condor-G better.

This booklet assumes that you already know the basics of using Condor-G. If you do not, please review the Condor manual, particularly Chapter 5, *Grid Computing*. The Condor manual can be found online at the Condor web site:

http://www.cs.wisc.edu/condor

If you have comments on this booklet, or suggestions for how to make it better, please let us know. Send email to the Condor team at:

condor-admin@cs.wisc.edu

Always specify a log file in your submit file.

(With a note about idle jobs)

Condor-G tells you important information in log files. For example, look at this submit file:

1
```
Universe        = Globus
Globusscheduler = beak.cs.wisc.edu
Executable      = analysis
Arguments       = "beta 10"
Input           = analysis.in
Output          = analysis.out
Log             = analysis.log
Queue
```

When your job is submitted, Condor-G will create a file called analysis.log that shows you what happened to your job, and when it happened. Sometimes your job looks idle but there is a problem listed in the submit file. For example, I submitted a job, but spelled the hostname "beak" incorrectly as "beakk". The job appeared idle, but Condor-G gave extra information in the log file:

(Note that the comment lines beginning with a # are comments by me—they are not in the log file.)

2
```
000 (018.000.000) 10/13 15:15:31 Job submitted from host:
<128.105.121.21:55506>
...
# Aha—this is our problem!
020 (018.000.000) 10/13 15:15:35 Detected Down Globus Resource
    RM-Contact: beakk.cs.wisc.edu
```

When this happens, Condor-G cannot tell the difference between a misspelled host and a host that is actually down. (There may be a transient DNS failure that looks like a misspelled host error.) Condor-G will try to submit my job again later, but it remains idle for now. By looking in the log file, you can see what happened. To fix the problem, I used condor_rm to remove the job, edited the submit file to have the correct name, and resubmitted the job. Now in the log file I saw:

3
```
# This was the above failed job.
009 (018.000.000) 10/13 15:22:26 Job was aborted by the user.
        via condor_rm (by user roy)
...
# I resubmitted the job 9 seconds later
000 (019.000.000) 10/13 15:22:37 Job submitted from host:
<128.105.121.21:55506>
...
```


The Globus Jobmanager Log File (gram_job_mgr_<number>.log)

- Useful for job execution problems

- Located on the remote system

After the gatekeeper runs your job, it creates a jobmanager. The jobmanager submits your job to the local batch system and monitors the job while it is in the queue. While the jobmanager runs, it produces a log file which is either located in the user's home directory or in /tmp. This log file often has interesting error messages in it. For instance, when the jobmanager fails to create a scratch directory for a job to store its files in, the log file contains:

3
```
11/10 12:15:10 Job Manager State Machine (entering):
GLOBUS_GRAM_JOB_MANAGER_STATE_MAKE_SCRATCHDIR
11/10 12:15:10 Failed to create scratch dir
```

It is hard to document the wide variety of errors you may see in this log file, but it is often a treasure trove for problem solving.

The Condor-G GridManager log (GridmanagerLog.username)

- Useful for Globus interaction problems

- Located on the submit system

Condor-G records problems with its interaction with Globus in the Gridmanager log files. These are normally either located in /tmp or the GridLogs subdirectory of Condor's log directory. If you do not have access to the remote site's log files, this may help you debug more problems. If you are having problems and the log is not showing you enough information, you can increase the amount of debugging. Edit your Condor configuration file to set:

4
```
GRIDMANAGER_DEBUG = D_FULLDEBUG
```

Then execute "condor_reconfig" to tell Condor to pay attention to this change in the configuration. This will give you quite a bit more debugging information in the log file.

Problem Solving & Log Files

As much as we would like to tell you that you will never encounter problems, you will. Condor-G does a great deal to deal with problems. But how do you solve the problems that cause jobs to be put on hold, or other more serious problems?

There are three log files that you should know about, in addition to the log file for your job. Each of these has information that will help you debug problems.

The Globus Gatekeeper Log File (globus-gatekeeper.log)
- Useful for authentication, authorization, and problems starting jobmanagers
- Located on the remote sysytem
Every time you submit a job to a Globus site, the Globus gatekeeper approves your job's request and creates a *jobmanager* to handle your job's request. The gatekeeper creates a log file of everything that it does, and it will often help you. For instance, in your job log file, you might see:

(1)
```
018 (091.000.000) 11/10 18:06:26 Globus job submission failed!
    Reason: 7 authentication with the remote server failed
```

On the remote server, you can look at the gatekeeper log. It will be in $GLOBUS_LOCATION/var/globus-gatekeeper.log. In this case, I can see:

(2)
```
Notice: 5: Authenticated globus user:
/DC=org/DC=doegrids/OU=People/CN=Alain Roy 424511
Failure: globus_gss_assist_gridmap() failed authorization. rc=1
```

This tells us two things: it recognized my user certificate, but it failed to authorize me. The fix for this is to add me to the grid-mapfile. In this case, looking in the log file did not add much to our understanding over the error message that Condor-G gave us, but in some cases the error messages can be rather enlightening. Error messages that you find can help you debug problems with authentication, authorization, and creation of the job manager, but will rarely help you with problems that occur while your job is running. If you do not find any record in the gatekeeper log, then the gatekeeper may not have started. If you have access, look in the system's log files to see if you can see why it did not start.


```
# Here we see that the job has been accepted by Globus.
# You will probably never need the JM-Contact.
017 (019.000.000) 10/13 15:22:50 Job submitted to Globus
    RM-Contact: beak.cs.wisc.edu
    JM-Contact:
https://beak.cs.wisc.edu:47818/22589/1066076561/
    Can-Restart-JM: 1
...
001 (019.000.000) 10/13 15:22:50 Job executing on host:
beak.cs.wisc.edu
...
# The job has finished. Unfortunately, Condor-G does not
# know the details of the job, so they are all zeros.
# More information will be available in future versions of
# Condor-G
005 (019.000.000) 10/13 15:22:55 Job terminated.
     (1) Normal termination (return value 0)
         Usr 0 00:00:00, Sys 0 00:00:00  -  Run Remote Usage
         Usr 0 00:00:00, Sys 0 00:00:00  -  Run Local Usage
         Usr 0 00:00:00, Sys 0 00:00:00  -  Total Remote Usage
         Usr 0 00:00:00, Sys 0 00:00:00  -  Total Local Usage
     0  -  Run Bytes Sent By Job
     0  -  Run Bytes Received By Job
     0  -  Total Bytes Sent By Job
     0  -  Total Bytes Received By Job
...
```

Without the log file, I would have not learned any of this information. You should always have a log file for every job you submit to Condor-G.

Note that you can have multiple jobs using the same log file. You can tell which job is associated with each message by examining the numbers in parentheses. For example, the last message above has *(019.000.000)* in it. That means *cluster 19, process 0, sub-process 0*. In the condor_q display, you would see it listed as 19.0.

Advanced condor_q usage

When you submit a Condor-G job, you probably know that you can get information about the job using condor_q:

1
```
% condor_q
ID      OWNER  SUBMITTED     RUN_TIME  ST PRI SIZE CMD
26.0   roy     10/13 20:08 0+00:00:00   I  0   0.0  sleep 120

1 jobs; 1 idle, 0 running, 0 held
```

Many people are unaware that they can get Globus specific information from condor_q:

2
```
% condor_q -globus
ID     OWNER     STATUS  MANAGER  HOST             EXECUTABLE
 26.0  roy   UNSUBMITTED fork  beak.cs.wisc.edu    /bin/sleep
```

This tells you useful information that condor_q without the –globus argument does not tell you:

- The job has not yet been submitted to Globus.
- The job is going to the beak.cs.wisc.edu Globus gatekeeper.
- The job is using the Globus fork job manager. (Instead of a batch system job manager, such as Condor or PBS.) Note that this is a bit of a guess because Condor-G has to guess from the name of the jobmanager, and it may be wrong. For example, it guesses the a plain "jobmanager" is a fork jobmanager. This is usually correct, but different sites could change that default.

Many people quite reasonably expect "condor_q –analyze" to return meaningful results. Unfortunately, it does not tell you anything useful for Condor-G jobs unless your Condor-G jobs use matchmaking. This is a new feature in Condor-G, so most of you are not using it yet. Improving the "-analyze" option is something the Condor team hopes to improve in future releases.

You can find out all sorts of details if you ask condor_q nicely:

3
```
% condor_q -l

MyType = "Job"
TargetType = "Machine"
ClusterId = 26
QDate = 1066093717
...
```


Another way to increase the limit is to become root, increase the limit, switch back the user, and start the program you want it increased for:

4
```
su - root
ulimit -n 16384         # if root uses sh
limit descriptors 16384 # if root uses csh
su - your_user_name
program_to_run, like condor_master
```

Local port range

When you submit lots of jobs, you might run out of ports. Condor-G uses randomly assigned ports from a limited range. By default that range is 1024 to 4999, allowing 3975 simultaneous outgoing connections. We have found it important to increase this for Condor-G submit nodes. You can find the current range with:

5
```
cat /proc/sys/net/ipv4/ip_local_port_range
```

You can set the range with:

6
```
echo 1024 65535 > /proc/sys/net/ipv4/ip_local_port_range
```

Scalability: Linux configuration

(This is extracted from http://www.cs.wisc.edu/condor/condorg/linux_scalability.html)

Doing large scale grid work, we regularly press various limits of Linux and other operating systems. If you submit lots of jobs with Condor-G you will also bump into these limits. We will discuss some of these limits for Linux.

At several points we suggest making changes to the Linux kernel's configuration by echoing data into the /proc filesystem. These changes are transient and the system will reset to the default values on a reboot. As a result, you will want to place these changes somewhere where they will be automatically reapplied on reboot. On many Linux systems, you can use the /etc/rc.d/rc.local script to do this. Depending on your particular configuration, you might also be able to edit /etc/sysctl.conf.

System-wide file descriptor limit

Linux has a limit on the concurrently open file descriptors throughout the system. It defaults to 8192. We have found it important to increase this for Globus gatekeeper nodes and Condor-G submit nodes.

To see the current limit:

1
```
% cat /proc/sys/fs/file-max
```

To set the limit to 32768, as root do:

2
```
% echo 32768 > /proc/sys/fs/file-max
```

Per-process file descriptor limit

Each user has per-process file descriptor limits. It defaults to 1024, but can be increased to the system's hard limit. Unfortunately the hard limit is also 1024. We have found it important to increase this when starting the condor_master for Condor-G.

You can check the limit with:

3
```
ulimit -n         # sh
limit descriptors # csh
```

You may be able to give each user a larger limit in the /etc/security/limits.conf file. This will only apply to Condor daemons started as the user in question. At the moment (October 2003) Condor will ignore these limits when run as root.


```
x509userproxysubject =
"/DC=org/DC=doegrids/OU=People/CN=Alain_Roy_424511/CN=proxy"
x509userproxy = "/tmp/x509up_u8471"
GlobusResource = "beak.cs.wisc.edu"
GlobusStatus = 32
WantClaiming = FALSE
NumGlobusSubmits = 0
GlobusResubmit = FALSE
GlobusContactString =
"https://beak.cs.wisc.edu:53044/30086/1066093721/"
```

The information you see here is a ClassAd. ClassAds are essentially lists of name-value pairs. Details about ClassAds are in the manual. The ClassAd listed here is incomplete, to simplify this discussion. The information in the ClassAd can be surprisingly useful, because you can customize the output that condor_q reports. For instance, suppose you submitted jobs to two different grid sites. Condor_q would show you:

4
```
% condor_q -globus
ID      OWNER STATUS  MANAGER   HOST                EXECUTABLE
  27.0   roy  ACTIVE      fork  beak.cs.wisc.edu    /bin/sleep
  28.0   roy  UNSUBMITTED fork  omega.cs.wisc.edu   /bin/sleep
```

You can constrain condor_q to show you just the jobs submitted to beak: (This command is all on one line)

5
```
% condor_q -globus -constraint 'GlobusResource ==
"beak.cs.wisc.edu"'

chopin.cs.wisc.edu
 ID      OWNER STATUS MANAGER  HOST                EXECUTABLE
  27.0   roy   ACTIVE fork     beak.cs.wisc.edu    /bin/sleep
```

You can constrain based on any attribute in the ClassAd. See the condor_status manual page for more information about using constraints.

Understanding jobs on hold

Condor-G will put jobs on hold when it encounters Globus errors. At first, this may seem surprising, but it is useful to you and your jobs. In earlier versions, when errors were encountered Condor-G would terminate the job and record it in the log file. Condor-G now places your job on hold so that you can attempt to fix the error, then tell Condor-G to restart the job with the condor_release command.

For example, if you submit a job that requires an executable pre-staged on a remote grid site, but the executable does not exist, your job will be placed on hold. (Normally, executables are always transferred, so we do not rely on them being on the remote grid site, but in this example we specified "transfer_executable = false" to override that behavior.)

1
```
# Submit file
executable = /bin/doesnt_exist
transfer_executable = false
arguments = 120
globusscheduler = beak.cs.wisc.edu
universe = globus
output = noexec.out
log = noexec.log
queue
```

2
```
% condor_q
ID   OWNER SUBMITTED  RUN_TIME  ST PRI SIZE CMD
29.0 roy  10/13 20:28 0+00:00:00 H  0  0.0 doesnt_exist 120

1 jobs; 0 idle, 0 running, 1 held
```

Notice that the job is held. This means that it will not run or exit, unless you intervene. But why is it held? Ask condor_q with the -hold option:

3
```
% condor_q -hold
ID   OWNER HELD_SINCE  HOLD_REASON
29.0 roy   10/13 20:28 Globus error 5: the executable does
not exi

1 jobs; 0 idle, 0 running, 1 held
```

Condor_q is mostly helpful here: we can see most of the reason that the job is on hold. The executable does not "exi"—Condor-G chopped off the text. You can probably figure out the problem, but your knowledge of condor_q that you acquired earlier will aid you, if it is unclear:

4
```
% condor_q -l | grep HoldReason
HoldReason = "Globus error 5: the executable does not ex-
ist"
HoldReasonCode = 2
HoldReasonSubCode = 5
```

You can now fix the problem (put the executable into place) and use condor_release to restart the job. If you cannot fix the problem, you can use condor_rm to remove the job.

Condor-G will put jobs on holds for several reasons, including:

- Globus GRAM errors, including a missing executable.
- Requesting transfer of an output file that is not created.
- Use of the periodic_hold expression. This lets you automatically put a job on hold when something happens, and you can define the something. See the condor_submit manual page for more information.