Using the PBS Pro cluster at ABCC NCI-Frederick

Karol Miaskiewicz (SAIC/NCI-Frederick)

This document provides an introduction to the usage of the PBS Pro cluster on the server ncisgi.ncifcrf.gov, located within the Advanced Biomedical Computing Center at NCI-Frederick. Keep in mind, that our PBS Pro cluster is an evolving configuration. As a result, this document may be subject to frequent modifications and rewritings. Please, check it frequently for up-to-date information.

This version updated Monday, November 22, 2006

Table of contents

Introduction

PBS Pro is a workload and resource management system in use on computational servers. The PBS server that executes on the frontend host ncisgi.ncifcrf.gov, sends batch tasks for execution to the following computational servers:

dexter - SGI Altix with 256 x 1500MHz Itanium2 processors, 1024 GB of memory, 3 TB of scratch disk space

ncip595 - IBM P595 with 64 x 1900MHz Power5 processors (1.92 MB of L2 cache shared between two cpus, 36 MB of L3 cache), 256 GB of memory, 2 TB of scratch disk space

avanti - SGI Origin 3800 with 64 x 600 MHz MIPS R14000 processors (8 MB of secondary cache per processor), 64 GB of memory, 1 TB of scratch disk space

ncies45 - HP ES45 with 4 x 1250MHz Alpha EV68 processors, (16 MB of secondary cache per processor), 32 GB of memory, only 2 processors are enabled for PBS batch use

All computing systems also have access to ten terabytes of shared disk space on an SGI TP9700 disk array. This space is shared as a directly connected CXFS clustered filesystem or through NFS.

The primary PBS commands are: 'qsub' to submit a job, 'qstat' to see the queues or job status, and 'qdel' to delete a job. Your path is set to include PBS commands. Man pages for PBS commands are also available.

Altair provides a User Guide for PBS Pro. This guide is located on ncisgi at:
/usr/local/pbs/doc/PBSproUserGuide.pdf

We highly recommend that you study this document for a more detailed description of PBS Pro and its operations.

Before you start

In order to use PBS on ncisgi, each user's .rhosts file needs to modified, and possibly the .cshrc file. When you are ready to start using PBS, run the script 'prepare_for_pbs'. This script will:

- make the necessary modifications in your .rhosts file
- create /usr/tmp/[yourusername] directories on all computing nodes

It is mandatory that the number of desired processors for your job be specified in your PBS submission script (see the next section). Double check that the specified number of processors is what your job really uses. Jobs without an accurate number of processors defined will be terminated due to the underutilized resources.

Limit your jobs to no more than 8 processors. If you want to run on more processors, seek approval by contacting Bob Lebherz (lebherz@ncifcrf.gov) or Karol Miaskiewicz (miaskiew@ncifcrf.gov).

Note,that there are nodes of different computer architectures available in our cluster (currently SGI Irix, IBM AIX, Linux on SGI Altix, and HP/Compaq Tru64). By default, all submitted jobs are scheduled to run on SGI Irix nodes. To run your job on a different computer architecture, read the section "Running on a specific node or specific architecture" near the end of this document.

Not all applications are available on all computer architectures represented in the cluster. For example, Gaussian03 is installed on the SGI and IBM nodes but not the Alpha nodes (i.e. ncies45). Information about the availability of scientific programs on different platforms can be accessed from our website www.abcc.ncifcrf.gov (go to the "Scientific Applications" section). If the information that you are seeking is not present, please contact our staff (http://support.abcc.ncifcrf.gov or 301-846-5555).

Your home directory is shared among all compute nodes (and ncisgi, too). There are disk quotas enforced on the /users filesystem (this is where the home directories are located), check your quota with the 'quota -v' command. /users is shared via CXFS mechanism on the SGI Altix nodes and the SGI Origin 3800 (hostname avanti) which provides shared filesystem as if it was locally attached storage. However, non CXFS

enabled nodes use much slower NFS mechanism to share /users.

Please, note that NFS access can cause performance degradation during very I/O intensive operations especially when accessing very large files. What it all means for you, is that your job can write output files directly to your home directory if this does not involve very intensive I/O operations (and if they fit into your available disk quota). This is especially true on CXFS nodes. However, if intensive I/O operations are involved, you should read/write to a local filesystem, and specifically /usr/tmp, on the node where the job is executing.

To illustrate this issue with a real-world example, it may be reasonable to write the Gaussian03 output file in your home directory, but all Gaussian scratch files should be written to the local /usr/tmp filesystem. In another example, it is reasonable to write outputs from molecular dynamics runs (such as done with programs CHARMM, AMBER, or Discover) directly into your home directory if you do not not save coordinates and velocities very frequently. However, it is likely that these files will be too large for your disk quota, and for this reason you should consider using the local scratch disk area (i.e. /usr/tmp).

There is another reason that you may want to avoid using NFS filesystems in your PBS jobs. NFS filesystems are very sensitive to any network related problems. If your job tries to write to an NFS mounted filesystem, it may abort when there are intermittent network problems (such as network congestion) and access to an NFS server times out. For this reason, limiting your job to using just local filesystems (such as /usr/tmp) results in increased reliability.

The /usr/tmp filesystem is a dedicated scratch area on each of computing nodes. Keep in mind, that each node has its own /usr/tmp filesystem. As a case in point, if you place files in /usr/tmp/[yourusername] on ncisgi, these files will not be available in /usr/tmp/[yourusername] on other nodes, unless you specifically transfer them there. The /usr/tmp filesystem is configured for the best performace. We use disk striping across multiple disks and controllers, and we also use large block sizes to provide optimal performance.

/usr/tmp is the largest filesystem available to you on the computing nodes - on many of our nodes it is one terabyte or more in size. Therefore, if you need performance and/or a

lot of disk space, your job should be using the /usr/tmp filesystem, and specifically the /usr/tmp/[yoursusername] directory. Keep in mind, that, as a scratch area, /usr/tmp is not backed up. Also, files not accessed for over 30 days are automatically removed. Never store any of your important files under /usr/tmp without saving them elsewhere.

Job Submission

Job submission is performed via the 'qsub' command:
qsub <batch submission script>

A batch submission script contains the usual commands required to execute the application that you are using. It also contains PBS job submission options in the header of the script. These options are placed in the lines that start with the "#PBS" string. These lines must be the first lines of the script.

For example, the following script will perform a Gaussian98 calculation (let's name it g98.job):

```
#PBS -S /bin/csh
#PBS -N h2o_mp2
#PBS -l cput=10:00:00
#PBS -l pvmem=400mb
#PBS -l ncpus=4
#PBS -m e

setenv g98root /usr/local/fbscapp/g98_A11.3
source $g98root/g98/bsd/g98.login
setenv GAUSS_SCRDIR /usr/tmp/miaskiew

cd /users/primgr/miaskiew/Water
g98 < h2o_mp2_opt.in > h2o_mp2_opt.out
```

It can be submitted to PBS with the simple command 'qsub g98.job'. All of the necessary options are provided at the begining of the script (lines starting with #PBS):

`#PBS -S /bin/csh`

The -S flag denotes the shell to be used.

`#PBS -N h2o_mp2`

The -N flag denotes the name with which the job will appear in the batch system (this does not have to be the name of the script, it can be practically any name you deem relevant, but it cannot start with a digit). The name will be truncated to ten characters in the qstat output, if it is longer.

`#PBS -l cput=10:00:00`

The -l flag indicates the limits of resources required for the job. cput is the CPU

time required for the job. It can be provided as an integer of number of seconds or in the notation hours:minutes:seconds (or minutes:seconds).

```
#PBS -l pvmem=400mb
```

Pvmem sets the limit of virtual memory per process. In this case it is 400 megabytes. It should be specified as an integer followed by one of the suffixes: b, kb, mb, gb, w, kw, mw, gw (for bytes, kilobytes, megabytes, gigabytes, words, kilowords, megawords, and gigawords, respectively).

```
#PBS -l ncpus=4
```

Number of cpus required.

```
#PBS -m e
```

The -m flag asks PBS to send an email message to the user. The 'e' argument means to send the message when the job is completed.

Note that it is not necessary to list PBS flags separately in each line. They can be grouped as in the following example:

```
#PBS -S /bin/csh -N h2o_mp2
#PBS -l cput=10:00:00,pvmem=400mb,ncpus=4
#PBS -m e
```

Also note that it is not required and not recommended to submit a job to a specific queue. PBS will make the decision about queue placement based upon your requested resources.

There are more PBS flags that can be used in the script. Man pages on qsub provide information on the available flags/options. At minimum, the following flags must be used in any PBS script to run on our SGI and IBM nodes:

-S (to specify a shell for the job)
-l cput= (to specify the per job time limit)
-l pvmem= (to specify the virtual memory per process)
-l ncpus= (to specify the number of CPUs)

More flags are required to run on the Alpha node (i.e. node ncies45). Contact us for additional details regarding the Alpha.

Please, consult the official PBS User Guide for more information on reserving of

computational resources in PBS - /usr/local/pbs/doc/PBSproUserGuide.pdf.

File Transfer

The example discussed so far does not take into account the file delivery issues. If all of your input and output files reside within your home directory (see the discussion in the section "Before you start") or one of other globally shared filesystems such as /abcc, you do not need to be concerned with file transfer issues. However, if your job will use local(i.e. non-shared) filesystems you must take care of the delivery of your files to and from the execution host.

Using our example of a Gaussian job, if the only output file that is of interest is the Gaussian log file (i.e. h2o_mp2_opt.out) there is no need to transfer any files since the log file is written in your globally shared home directory. However, let's assume that you want to restart the calculation from a Gaussian checkpoint file. Since the checkpoint file is frequently accessed by Gaussian during execution we want to use the local /usr/tmp filesystem for optimal performance. Also, we want the checkpoint file to be saved and returned after the execution of our job.

There are a several methods that can be applied to file delivery in our configuration. To illustrate them with our Gaussian job example：

① Since your home directory is shared between all potential execution hosts, you could simple used the Unix cp command to copy the checkpoint file from your home directory to /usr/tmp/[yourusername] on the execution host:

```
#PBS -S /bin/csh
#PBS -N h2o_mp2
#PBS -l cput=10:00:00
#PBS -l pvmem=400mb
#PBS -l ncpus=4
#PBS -m e

setenv g98root /usr/local/fbscapp/g98_A11.3
source $g98root/g98/bsd/g98.login
setenv GAUSS_SCRDIR /usr/tmp/miaskiew

cd /users/primgr/miaskiew/Water
```

```
cp h2o_mp2_opt.chk /usr/tmp/miaskiew
g98 < h2o_mp2_opt.in > h2o_mp2_opt.out
cp /usr/tmp/miaskiew/h2o_mp2_opt.chk .
```

(Note that all of the commands in the script are executed on an execution host and not on ncisgi where the job is submitted. This is why the cp command in the preceding example does transfer files to and from /usr/tmp/miaskiew on an execution host.)

② The same can be achieved with the Unix rcp command. Note that if you need to transfer the checkpoint file to and from a non-shared filesystem on ncisgi, the rcp command is the one that you must use (you could also use stagein/stageout directives as described below).

In our example, let us assume that we want keep the checkpoint file on the /usr/tmp filesystem on ncisgi. As explained before, this is a different /usr/tmp filesystem than that found on an execution host.

```
#PBS -S /bin/csh
#PBS -N h2o_mp2
#PBS -l cput=10:00:00
#PBS -l pvmem=400mb
#PBS -l ncpus=4
#PBS -m e

setenv g98root /usr/local/fbscapp/g98_A11.3
source $g98root/g98/bsd/g98.login
setenv GAUSS_SCRDIR /usr/tmp/miaskiew

rcp ncisgi:/usr/tmp/miaskiew/h2o_mp2_opt.chk /usr/tmp/miaskiew
cd /users/primgr/miaskiew/Water
g98 < h2o_mp2_opt.in > h2o_mp2_opt.out
rcp /usr/tmp/miaskiew/h2o_mp2_opt.chk ncisgi:/usr/tmp/miaskiew
```

③ The most universal and foolproof way to transfer files is with the PBS Pro stagein and stageout directives.

```
#PBS -S /bin/csh
#PBS -N h2o_mp2
#PBS -l cput=10:00:00
#PBS -l pvmem=400mb
#PBS -l ncpus=4
#PBS -m e
#PBS -W
stagein=/usr/tmp/miaskiew/h2o_mp2_opt.chk@ncisgi:/usr/tmp/miaskiew/h2o_mp2_o
```

```
pt.chk
#PBS -W
stageout=/usr/tmp/miaskiew/h2o_mp2_opt.chk@ncisgi:/usr/tmp/miaskiew/h2o_mp2_
opt.chk

cd /users/primgr/miaskiew/Water
g98 < h2o_mp2_opt.in > h2o_mp2_opt.out
```

(Note the ordering of hostnames in the stagein directive! It appears to be awkward, but this is how it works. The first hostname is always an execution host. In the case of stagein directive it means that the first host listed in the stagein specification is the host where the files are transferred to, and not the host where the files are transferred from.)

As an additional example other than Gaussian, let's consider an AMBER job which we want to run from the /usr/tmp/miaskiew directory. We need to deliver all AMBER input files to /usr/tmp/miaskiew on an execution host, and then return the output files to the home directory.

① Using the cp command:

```
#PBS -S /bin/csh -N dna12mer
#PBS -l cput=100:00:00,pvmem=400mb,ncpus=4
#PBS -m e

setenv AMBERHOME /usr/local/fbscapp/amber6
set path = ($path $AMBERHOME/exe)

cd /usr/tmp/miaskiew

cp /users/primgr/miaskiew/dna12mer/dna_eq.in .
cp /users/primgr/miaskiew/dna12mer/dna_eq.top .
cp /users/primgr/miaskiew/dna12mer/dna_eq.crd .

mpirun -np 4 sander -O -i dna_eq.in -o dna_eq.out \
   -p dna_seq.top -c dna_eq.crd -x dna_eq.traj -r dna_eq.rst

cp dna_eq.rst /users/primgr/miaskiew/dna12mer
cp dna_eq.traj /users/primgr/miaskiew/dna12mer
cp dna_eq.out /users/primgr/miaskiew/dna12mer
```

② With the rcp command:

```
#PBS -S /bin/csh -N dna12mer
#PBS -l cput=100:00:00,pvmem=400mb,ncpus=4
#PBS -m e
```

```
setenv AMBERHOME /usr/local/fbscapp/amber6
set path = ($path $AMBERHOME/exe)

cd /usr/tmp/miaskiew

rcp ncisgi:dna12mer/dna_eq.in dna_eq.in
rcp ncisgi:dna12mer/dna_eq.top dna_eq.top
rcp ncisgi:dna12mer/dna_eq.crd dna_eq.crd

mpirun -np 4 sander -O -i dna_eq.in -o dna_eq.out \
   -p dna_seq.top -c dna_eq.crd -x dna_eq.traj -r dna_eq.rst

rcp dna_eq.rst ncisgi:dna12mer/dna_seq.rst
rcp dna_eq.traj ncisgi:dna12mer/dna_seq.traj
rcp dna_eq.out ncisgi:dna12mer/dna_seq.out
```

③ With the PBS stagein and PBS stageout directives:

```
#PBS -S /bin/csh -N dna12mer
#PBS -l cput=100:00:00,pvmem=400mb,ncpus=4
#PBS -m e
#PBS -W stagein=/usr/tmp/miaskiew/dna_eq.in@ncisgi:dna12mer/dna_eq.in
#PBS -W stagein=/usr/tmp/miaskiew/dna_eq.top@ncisgi:dna12mer/dna_eq.top
#PBS -W stagein=/usr/tmp/miaskiew/dna_eq.crd@ncisgi:dna12mer/dna_eq.crd
#PBS -W stageout=/usr/tmp/miaskiew/dna_eq.rst@ncisgi:dna12mer/dna_eq.rst
#PBS -W stageout=/usr/tmp/miaskiew/dna_eq.traj@ncisgi:dna12mer/dna_eq.traj
#PBS -W stageout=/usr/tmp/miaskiew/dna_eq.out@ncisgi:dna12mer/dna_eq.out

cd /usr/tmp/miaskiew

setenv AMBERHOME /usr/local/fbscapp/amber6
set path = ($path $AMBERHOME/exe)

mpirun -np 4 sander -O -i dna_eq.in -o dna_eq.out \
   -p dna_seq.top -c dna_eq.crd -x dna_eq.traj -r dna_eq.rst
```

All of the above examples will work fine when delivering input files from a submission host (ncisgi) to an execution host. However, they may fail when returning output files to ncisgi if the CPU time limit (as defined with the -l cput=100:00:00) is exhausted.

In our AMBER example, if the job terminates prematurely due to the CPU time limit while in execution of the sander program, no commands in the script after the 'mpirun -np 4 sander' line will be executed, and thus no output files will be ruterned the to home directory when cp and rcp commands are used. To avoid this problem:

- use the PBS stageout directives - they are always executed, even when the CPU time limit is exhausted

- use cp or rcp, but define different CPU time limits for job (cput) and for process (pcput) with the job limit being at least 20 seconds longer. In our AMBER example, we would need to add the pcput limit to the PBS limits definitions:

  #PBS -l cput=100:01:00,pvmem=400mb,ncpus=4,pcput=100:00:00

  If the limit per job is longer than the limit per process, even when the sander process reaches the process CPU time limit, there is still job CPU time available to execute the consecutive commands.

Resource Limits

Currently, the longest cpu time allowed is 999 hours (3596400 seconds). The largest pvmem value allowed is 10000 megabytes, and one can use no more than 8 processors. These limits may change. To verify the current limits, use the 'qstat -q' command to list all of the queues, and then display limits in each queue with the 'qstat -Qf [queuename]' command. If your job needs more than the indicated available limits, we have special queues available. These queue are normally disabled. Contact Karol Miaskiewicz (miaskiew@ncifcrf.gov) or Bob Lebherz (lebherz@ncifcrf.gov) to run in these queues.

Running on a Specific Node or a Specific Architecture

While we normally discourage such use and rely on PBS load balancing, forced execution on a particular node or a particular computer architecture is justified when one is doing benchmarking or when an application is available only on a particular node or a particular architecture.

Running on a particular node can be achieved in a couple of ways:

(1) "-l select=1:host=name

where the "name" is the hostname of the node

(2) "-l select=1:property=true"

where the "property" is the property of the node where you want to run your job.

Thus, to force a job to run on the SGI Origin 3800 (hostname avanti) one would use the following option:

```
#PBS -l select=1:ncpus=1:host=avanti
```

or

```
#PBS -l select=1:ncpus=1:sgi-origin3800-600mhz=true
```

In the second example, the value sgi-origin3800-600mhz is a property of the node avanti.

Note that it is important to define number of processors to be used on the node within the “select” statement as in the above examples.

You can get the hostnames of all of the members of our PBS cluster and their respective properties and resources displayed with the 'pbsnodes -a' command. For example, the pbsnodes output for node avanti, indicates that it is an SGI Origin 3800 system with 600 MHz processors (property sgi-origin3800-600mhz), 64 GB of memory (resources_available.mem = 66060288kb), and sixty four processors (pcpus = 64):

```
avanti
     Host = avanti
     Port = 15002
     ntype = PBS
     state = free
     license = l
     pcpus = 64
     Priority = 12
     jobs =
     resources_available.arch = irix6
     resources_available.gaussian = True
     resources_available.host = avanti
     resources_available.jaguar = True
     resources_available.mem = 66060288kb
     resources_available.ncpus = 62
     resources_available.pvmem = 60000mb
     resources_available.sgi-origin3800-600mhz = True
     resources_available.targetscan = True
     resources_assigned.gcc = 0
     resources_assigned.mem = 0kb
     resources_assigned.ncpus = 0
     comment =
     resv_enable = True
```

You may want to run on a particular computer architecture - for example when your application is unique to the IBM AIX platform. In this case, you do not care which node will execute your job, as long as it is an IBM AIX node. You can specify this with the architecture resource flag:

```
#PBS -l arch=aix4
```

To see the available architecture resources, use the pbsnodes command. The output of pbsnodes will display it for each node in the form:

```
resources_available.arch = irix6
```

or

```
resources_available.arch = digitalunix
```

or

```
resources_available.arch = aix4
```

or

```
resources_available.arch = linux
```

Currently, there are nodes of four different architectures available in our cluster - irix6 (SGI Irix nodes) digitalunix (Alpha based DEC/HP Tru64 node), aix4 (IBM AIX nodes), and linux (SGI Altix node).

Note that irix6 is the default architecture. This means that all jobs are placed on SGI Irix nodes unless otherwise specified.

Customized Environment Settings for Different Operating Systems

Currently, there are nodes with four different operating systems present in our cluster - Irix, Tru64, Linux, and AIX. Note, that your home directory is shared among these three platforms. If you need to have unique environmental settings for each platform:

1. Copy your current .cshrc, .profile, and .login to .cshrc.IRIX64, .profile.IRIX64, and .login.IRIX64, respectively. These files will contain your settings specific to SGI/Irix systems. Use them to customize your settings there.

2. Copy the .cshrc, .profile, and .login from /usr/abcc/skel to your home directory.

Do not edit them!

3. Create .cshrc.OSF1, .profile.OSF1, .login.OSF1 for Alpha/Tru64 specific settings if you wish. They are not required, i.e. you should be able to run jobs on Tru64 nodes without them. Use them only if you need/want to customize your settings there.

4. Create .cshrc.AIX, .profile.AIX, .login.AIX for IBM AIX specific settings. Again they are not required to execute jobs, however, use them if you need to customize your settings on IBM nodes.

5. Create .cshrc.Linux, .profile.Linux, .login.Linux for SGI Altix specific settings, if you need them.

6. If you want to have common aliases/settings for both platforms - place them in cshrc.common or profile.common files (dependent upon which shell you are using).

Note that new accounts at the ABCC have all of the above operating system specific environmental files already created.

Viewing PBS Queues

The qstat command is used to see the PBS queues. Useful options:

qstat -a	lists all of the jobs within the PBS cluster
qstat -an	lists all of the jobs within the PBS cluster and their respective execution hosts (if executing)
qstat -q	lists all of the queues within the PBS cluster (including resource limits)
qstat -s	lists all of the jobs within the PBS with their

	respective status comments
qstat -Qf [queue]	lists all information about a specific queue
qstat -f [jobid]	lists detailed information about a specific job

Additional options are available. Please, read more about qstat in the man pages (man qstat).

Interactive Runs under PBS

PBS can also be used for interactive work. There are some applications that cannot be used in a batch environment. One can use such applications under PBS's resource management control with the interactive flag.

This example illustrates the use of interactive PBS jobs: Suppose that one needs to run a very long MATLAB computation that requires that it be initiated from MATLAB's graphical interface. Also, we want to execute Matlab on an SGI Irix system. This can be achieved with the following simple command:

```
qsub -N myjob -I -l cput=20:00:00,pvmem=2000mb,ncpus=1,arch=irix6
```

As soon as the requested resources are available, you will be presented with a prompt on an SGI node and you can start your interactive MATLAB session. Note that your session will be terminated with no warning when it exceeds the requested resources that you have specified.

In another example, one needs to compile and debug their program on an SGI Altix system. While this can be achieved via a PBS batch script, it may be easier and more convenient to do it interactively. In order to get an interactive shell on an SGI Altix, one could try one of the following:

```
qsub -N compile -I -l cput=20:00:00,pvmem=200mb,ncpus=1,arch=linux
```

One can also specify an interactive batch PBS session on a specific host with:

```
qsub -N compile -I -l cput=20:00:00,pvmem=200mb \
   -l select=1:ncpus=1:host=dexter
```

Occassionally, PBS may not be able to immediately intialize your interactive session. When this occurs, you may simply see the message:

```
qsub: waiting for job 29001.ncisgi to start
```

This occurs when the requested resources are not available. This can be due to the system you want to use is too busy to support any more jobs of any kind, even non-cpu intensive jobs, or due to PBS limits (either user or global limits) already being reached.

PBS Graphical User Interface

If you are interacting with an X11 enabled system (any Unix/Linux box, or a PC running X-server software), you are encouraged to use xpbs, a graphical user interface for pbs. xpbs allows one to view jobs in the PBS queues, resources available in the queues, and other information about the PBS cluster. xpbs will also help you to build and submit a PBS job.

Accessing Your Output Files During Job Execution

This is a very relevant question in our configuration when the execution host may be different than the one where you are logged in (i.e. ncisgi). If your job uses your home directory for output files, you will be able to see them on ncisgi since this is a shared filesystem. However, if your output files are on the /usr/tmp or another local filesystem on an execution host, you do not have direct access to these files while logged into ncisgi. To take a look at your output files, follow these steps:

First, determine which host is executing your job. You can see this information using the 'qstat -an' command.

Next, use the rcp command to copy the file back to ncisgi and view it. For example, if you find out that your job is running on the host named nucleic1, and the output that you would like to view is /usr/tmp/miaskiew/h2o_mp2.out:

```
rcp nucleic1:/usr/tmp/miaskiew/h2o_mp2.out .
```

This command will transfer the file to your current directory. Then use 'vi' or 'more' or whatever viewer/editor you prefer.

If it is sufficient to just see the end of the file to check the progress of your calculations, the 'tail' command can provide that information:

```
rsh nucleic1 tail -40 /usr/tmp/miaskiew/h2o_mp2.out
```

Another approach to viewing output files on a remote execution host is to establish a PBS interactive session with this host – this option may not be available on a fully

utilized server.

Checkpoint / Restart

Some operating systems - currently Irix - support process checkpoint and restart capability. If you are executing your job on SGI Irix node, you should be able to make use of the checkpoint/restart capabilities offered by Irix.

Checkpoint creates an image in disk-based files of a running process. When the process dies, it is possible to restart it from this checkpoint file. If, for some reason, our PBS cluster fails (hardware failure, power outage, etc.), the job can be restarted from the last checkpoint file, and, thus, does not need to be restarted from the begining saving user possibly a lot of CPU time.

Note that some applications offer their own checkpoint capabilities. Gaussian offers checkpointing, and many molecular dynamics programs also create restart files. A system checkpoint is totally different than an application checkpoint. With application checkpointing, such as in Gaussian, a user needs to restart their calculation which usually requires modifications (although often very minor) to a Gaussian input file. Also, some methods in Gaussian are not checkpointable via Gaussian checkpoint file.

System checkpointing allows the job to restart transparently to a user. He/she does not need to adjust input files, and restart jobs. The PBS will take care of all of the details when the job is restarted after a shutdown.

Unfortunately, not every application is checkpointable. Applications that keep open sockets (such as Discover are not checkpointable. However, some such as Gaussian, AMBER, CHARMM often are.

The PBS -c flag controls checkpoint behavior. If you absolutely do not want to have checkpoints performed on your job you should set it to n, i.e. '-c n'. If you want checkpoints performed at regular default intervals set on the PBS server, set it to c, i.e. '-c c', which we strongly recommend. Note, if you do not provide the -c flag, it defaults to '-c s' which means to perform a checkpoint only when PBS shuts down. The problem

with this behavior is that if the system crashes due to a hardware failure, PBS is not shutdown gracefully, and a checkpoint does not execute. This is why we suggest that you use the '-c c' flag.

Currently, our queues are set to perform a checkpoint after every 240 minutes of CPU time.

Note, that non-checkpointable jobs are restarted from the very beginning after PBS restarts on an execution host. Also, if a restart from the system checkpoint fails for any reason, the job is restarted from the very beginning. This usually means that your output files generated so far will be overwritten and you will loose all results from the interrupted run. If you want to avoid such situations, mark your job as non-rerunnable and non-checkpointable with '-r n -c n' options.

Example Scripts

Gaussian98

SGI Irix

```
#PBS -S /bin/csh -N mp2
#PBS -l cput=220:01:00,pcput=220:00:00,pvmem=4000mb,ncpus=8

cd /users/primgr/miaskiew/Work

setenv g98root /usr/local/fbscapp/g98_A11.3
source $g98root/g98/bsd/g98.login
setenv GAUSS_SCRDIR /usr/tmp/miaskiew

g98 < c3endo_mp2.in > c3endo_mp2.out

cp /usr/tmp/miaskiew/c3endo.chk .
```

IBM

The same script as above will run a Gaussian98 job on the IBM servers. One just needs to add arch=aix4 to the #PBS line to force execution on the IBMs, i.e.:

```
#PBS -l cput=220:01:00,pcput=220:00:00,pvmem=4000mb,ncpus=8,arch=aix4
```

Gaussian03

SGI Irix

```
#PBS -S /bin/csh -N mp2
#PBS -l cput=220:01:00,pcput=220:00:00,pvmem=4000mb,ncpus=8

cd /users/primgr/miaskiew/Work

setenv g03root /usr/local/fbscapp/g03_B04
source $g03root/g03/bsd/g03.login
setenv GAUSS_SCRDIR /usr/tmp/miaskiew

g03 < c3endo_mp2.in > c3endo_mp2.out

cp /usr/tmp/miaskiew/c3endo.chk .
```

IBM

The same script as above will run a Gaussian03 job on the IBM servers. One just needs to add arch=aix4 to the #PBS line to force execution on the IBMs, i.e.:

```
#PBS -l cput=220:01:00,pcput=220:00:00,pvmem=4000mb,ncpus=8,arch=aix4
```

SGI Altix

The same basic Gaussian03 script should also work on the SGI Altix server. One needs to specify arch=linux:

```
#PBS -l cput=220:01:00,pcput=220:00:00,pvmem=4000mb,ncpus=8,arch=linux
```

AMBER7

SGI Irix

```
#PBS -S /bin/csh -N amb_bench
#PBS -l cput=80:00:00,ncpus=4,pvmem=600mb

cd /users/primgr/miaskiew/AMBER

set SANDER = "/usr/local/fbscapp/amber7/exe/sander"

mpirun -np 4 $SANDER -O \
            -i 16s80na_eqe.in \
            -o 16s80na_eqe.out \
            -p 16s80nab.top \
            -c 16s80na_eqd.rst \
            -r 16s80na_eqe.rst \
            -x 16s80na_eqe.traj \
            -inf 16s80na_eqe.inf
```

(Note, that the number of processors is controlled via the -np flag to mpirun. Make sure that this number and and the number of processors requested from PBS via ncpus= are the same.)

SGI Altix

AMBER PBS script for the SGI Altix is similar to the one above for the SGI Irix. One needs to change the architecture requested to arch=linux. Additionally MPI environment variable MPI_MEMMAP_OFF needs to be set to control the memory usage.

```
#PBS -S /bin/csh -N amb_bench
```

```
#PBS -l cput=80:00:00,ncpus=4,pvmem=600mb,arch=linux

cd /users/primgr/miaskiew/AMBER
set SANDER = "/usr/local/fbscapp/amber7/exe/sander"

setenv  MPI_MEMMAP_OFF 1

mpirun -np 4 $SANDER -O \
            -i 16s80na_eqe.in \
            -o 16s80na_eqe.out \
            -p 16s80nab.top \
            -c 16s80na_eqd.rst \
            -r 16s80na_eqe.rst \
            -x 16s80na_eqe.traj \
            -inf 16s80na_eqe.inf
```

IBM

```
#PBS -S /bin/csh -N amb_ibm
#PBS -l cput=80:00:00,ncpus=4,pvmem=600mb,arch=aix4

cd /users/primgr/miaskiew/AMBER

set SANDER = "/usr/local/fbscapp/amber7/exe/sander"

setenv MP_PROCS 4
setenv MP_SHARED_MEMORY yes
setenv MP_WAIT_MODE poll

$SANDER -O -i 16s80na_eqe.in \
           -o 16s80na_eqe.out \
           -p 16s80nab.top \
           -c 16s80na_eqd.rst \
           -r 16s80na_eqe.rst \
           -x 16s80na_eqe.traj \
           -inf 16s80na_eqe.inf
```

(Note, that the number of processors is controlled via the MP_PROCS variable. Make sure that this number and and the number of processors requested from PBS via ncpus are the same.)

AMBER8

The scripts for submission of AMBER7 jobs will work for AMBER8. The only modification required is to change the location the of sander executable:

```
   set SANDER = "/usr/local/fbscapp/amber7/exe/sander"
```
to
```
   set SANDER = "/usr/local/fbscapp/amber8/exe/sander"
```

CHARMM

(Please note that our center does not have a license to share a copy of CHARMM with extramural users. You must have your own CHARMM license and your own copy of the program which must be installed in your own user area. If you have problems compiling and installing CHARMM, our personnel may help you with these tasks. The following scripts are examples for using CHARMM - modify them with the proper location of your own copy of CHARMM.)

SGI Irix

```
#PBS -S /bin/csh -N charmm_sgi
#PBS -l cput=02:02:00,pvmem=1000mb,ncpus=8

setenv CHARMM /usr/local/fbscapp/harvard/c27b3/exec/sgi64/charmm

cd /users/primgr/miaskiew/CHARMM

mpirun -np 8 $CHARMM < mbcodyn.inp > mbcodyn.out
```

(Note, that the number of processors is controlled via the -np flag to mpirun. Make sure that this number and and the number of processors requested from PBS via ncpus= are the same.)

IBM

```
#PBS -S /bin/csh -N charmm_ibm
#PBS -l cput=20:00:00,pvmem=1200mb,ncpus=8,arch=aix4

setenv CHARMM /usr/local/fbscapp/harvard/c27b3/exec/ibmsp/charmm

cd /users/primgr/miaskiew/CHARMM

setenv MP_PROCS 8
setenv MP_SHARED_MEMORY yes
setenv MP_WAIT_MODE poll

$CHARMM < mbcodyn.inp > mbcodyn.out
```

(Note, that the number of processors is controlled via the MP_PROCS variable. Make sure that this number and the number of processors requested from PBS via ncpus= are the same.)

GAMESS

SGI Irix

```
#PBS -S /bin/csh -N myjob
#PBS -l ncpus=4,pvmem=500mb,cput=50:00:30,pcput=50:00:00

set GAMESS=/usr/local/fbscapp/gamess/rungms
cd /users/primgr/miaskiew/Gamess

$GAMESS uracil_rohf_ccpVDZ_opt 4 uracil_rohf_ccpVDZ_opt.out
```

(Note, that the number of processors is controlled via the second argument to our rungms script. Make sure that this number and the number of processors requested from PBS via ncpus are the same.)

SGI Altix

The preceding GAMESS PBS script for SGI Irix will also work on our SGI Altix system. The only required change needed is to specify arch=linux, i.e.:

```
#PBS -l ncpus=4,pvmem=500mb,cput=50:00:30,pcput=50:00:00,arch=linux
```

IBM

The same script as above will run GAMESS job on the IBM servers. One needs to add arch=aix4 to force execution on the IBMs, i.e.:

```
#PBS -l ncpus=4,pvmem=500mb,cput=50:00:30,pcput=50:00:00,arch=aix4
```

NAMD

SGI Irix

```
#PBS -S /bin/csh -N namd -l pvmem=600mb,cput=2:00:00,ncpus=4

set path = ($path /usr/local/fbscapp/namd_2.5/bin)
cd /users/primgr/miaskiew/NAMD/er-gre
namd2 +p4 er-gre.namd
```

(Note, that the number of processors is controlled with the +p flag, such as 4 processors in the above example with +p4.)

IBM

```
#PBS -S /bin/csh -N namd -l pvmem=600mb,cput=2:00:00,ncpus=4,arch=aix4

set path = ($path /usr/local/fbscapp/namd_2.5/bin)
cd /users/primgr/miaskiew/NAMD/er-gre
poe namd2 er-gre.namd -nodes 1 -procs 4 -shared_memory yes
```

(NAMD is executed on the IBM systems via IBM's POE - Parallel Operating Environment. The number of processors is controlled with POE's -procs flag, such as '-procs 4' in the above example.)

Limitations, Caveats

If you specify a non-existent file in a stagein directive the job may not fail. Instead it may remain on the PBS server in a "W" (i.e. Waiting) state. Periodically, the PBS will attempt to try transfer the file again. If you create the missing file, your job may start. If it does not, you should delete your job with qdel and resubmit it after fixing the problem.

More Assistance

For a detailed guide to PBS Pro, please, read the document PBSproUserGuide.pdf available on ncisgi at /usr/local/pbs/doc/PBSproUserGuide.pdf.

If you require more assistance, contact our technical personnel on the web at

http://support.abcc.ncircrf.gov or by phone at 301-846-5555. SGI specific questions may also be sent directly to Karol Miaskiewicz (miaskiew@ncifcrf.gov). IBM questions should be addressed to Dennis Foley (foleyd@ncifcrf.gov), while SGI/Altix and HP/Alpha questions to Toni Harbaugh (harbaugh@ncifcrf.gov).