Computing on Mio & RA
Advanced Scripts and Commands

Timothy H. Kaiser, Ph.D.
tkaiser@mines.edu
Director - CSM High Performance Computing
Director - Golden Energy Computing Organization

Let's do some examples

```
[tkaiser@mio tests]$ [tkaiser@mio tests]$ cd oneday

[tkaiser@mio cwp]$ make
...
mpicc -o c_ex00 c_ex00.c
/opt/intel/Compiler/11.1/069/lib/intel64/libimf.so: warning: warning: feupdateenv is not implemented and will always
fail
mpif90 -o f_ex00 f_ex00.f
/opt/intel/Compiler/11.1/069/lib/intel64/libimf.so: warning: warning: feupdateenv is not implemented and will always
fail
rm -rf fmpi.mod
icc info.c -o info_c
ifort info.f90 -o info_f
cp info.py info_p
chmod 700 info_p
...
...
[tkaiser@mio cwp]$ cd scripts1
[tkaiser@mio cwp]$ ls
batch1  batch2  c_ex00  c_ex00.c  cwp.tar  f_ex00  f_ex00.f
info_c  info.c  info_f  info.f90  info_p  info.py  makefile
```

What we have

- [c_ex00.c, c_ex00]
  - hello world in C and MPI
- [f_ex00.f, f_ex00]
  - hello world in Fortran and MPI
- [info.c, info_c] [info.f90, info_f] [info.py]
  - Serial programs in C, Fortran and python that print the node name and process id. Creates a node name process id
- batch1
  - Batch file to run the 1 of the examples in parallel using mpiexec
- batch1b
  - Runs all of the examples

info.c

```
#include <unistd.h>
#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>
main() {
 char name[128],fname[128];
 pid_t mypid;
 FILE *f;

/* get the process id */
mypid=getpid();
/* get the host name */
gethostname(name, 128);
/* make a file name based on these two */
sprintf(fname,"%s_%8.8d",name,(int)mypid);
/* open and write to the file */
f=fopen(fname,"w");
fprintf(f,"C says hello from %d on %s\n",(int)mypid,name);
}
```

info.f90

```
program info
  USE IFPOSIX ! needed by PXFGETPID
  implicit none
  integer ierr,mypid
  character(len=128) :: name,fname
! get the process id
  CALL PXFGETPID (mypid, ierr)
! get the node name
  call mynode(name)
! make a filename based on the two
  write(fname,'(a,"_",i8.8)')trim(name),mypid
! open and write to the file
  open(12,file=fname)
  write(12,*)"Fortran says hello from",mypid," on ",trim(name)
end program
```

```
subroutine mynode(name)
! Intel Fortran subroutine to return
! the name of a node on which you are
! running
  USE IFPOSIX
  implicit none
  integer jhandle
  integer ierr,len
  character(len=128) :: name
  CALL PXFSTRUCTCREATE ("utsname", jhandle, ierr)
  CALL PXFUNAME (jhandle, ierr)
  call PXFSTRGET(jhandle,"nodename",name,len,ierr)
end subroutine
```

info.py

```
#!/opt/development/python/2.6.5/bin/python
import os
# get the process id
mypid=os.getpid()
# get the node name
name=os.uname()[1]
# make a filename based on the two
fname="%s_%8.8d" % (name,mypid)
# open and write to the file
f=open(fname,"w")
f.write("Python says hello from %d on %s\n" %(mypid,name))
```

Example Output From the Serial Programs

```
[tkaiser@mio cwp]$ ./info_c
[tkaiser@mio cwp]$ ls -lt mio*
-rw-rw-r-- 1 tkaiser tkaiser 41 Jan 11 13:47
mio.mines.edu_00050205
[tkaiser@mio cwp]$ cat mio.mines.edu_00050205
C says hello from 50205 on mio.mines.edu
[tkaiser@mio cwp]$
```

C MPI example

```
#include <stdio.h>
#include <stdlib.h>
#include <mpi.h>
#include <math.h>

/*****************************************************
This is a simple hello world program. Each processor prints out
it's rank and the size of the current MPI run (Total number of
processors).
*****************************************************/
int main(argc,argv)
int argc;
char *argv[];
{
    int myid, numprocs,mylen;
    char myname[MPI_MAX_PROCESSOR_NAME];

    MPI_Init(&argc,&argv);
    MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD,&myid);
    MPI_Get_processor_name(myname,&mylen);

/* print out my rank and this run's PE size*/
    printf("Hello from %d of %d on %s\n",myid,numprocs,myname);

    MPI_Finalize();
}
```

Fortran MPI example

```
 !**************************************
!  This is a simple hello world program. Each processor
!  prints out its rank and total number of processors
!  in the current MPI run.
!**************************************************************
      program hello
      include "mpif.h"
      character (len=MPI_MAX_PROCESSOR_NAME):: myname
      call MPI_INIT( ierr )
      call MPI_COMM_RANK( MPI_COMM_WORLD, myid, ierr )
      call MPI_COMM_SIZE( MPI_COMM_WORLD, numprocs, ierr )
      call MPI_Get_processor_name(myname,mylen,ierr)
      write(*,*)"Hello from ",myid," of ",numprocs," on ",trim(myname)
      call MPI_FINALIZE(ierr)
      stop
      end
```

Our Batch file, batch I

```
#!/bin/bash
#PBS -l nodes=2:ppn=8
#PBS -l walltime=08:00:00
#PBS -N test_1
#PBS -o outx.$PBS_JOBID
#PBS -e errx.$PBS_JOBID
#PBS -r n
#PBS -V
##PBS -m abe
##PBS -M tkaiser@mines.edu
#-----------------------------------------------
cd $PBS_O_WORKDIR

# get a short and full list of all of my nodes
sort -u  $PBS_NODEFILE > mynodes.$PBS_JOBID
sort   $PBS_NODEFILE > allmynodes.$PBS_JOBID

# select one of these programs to run
# the rest of the lines are commented
# out
export MYPROGRAM=c_ex00
#export MYPROGRAM=f_ex00
#export MYPROGRAM=info_p
#export MYPROGRAM=info_c
#export MYPROGRAM=info_f

echo "running" $PBS_O_WORKDIR/$MYPROGRAM
mpiexec -np 16 $PBS_O_WORKDIR/$MYPROGRAM
```

Running the job

```
[tkaiser@mio cwp]$ qsub batch1
3589.mio.mines.edu
[tkaiser@mio cwp]$ ls -lt | head
total 1724
-rw------- 1 tkaiser tkaiser    466 Jan 11 14:08 outx.3589.mio.mines.edu
-rw------- 1 tkaiser tkaiser     64 Jan 11 14:08 allmynodes.3589.mio.mines.edu
-rw------- 1 tkaiser tkaiser      0 Jan 11 14:08 errx.3589.mio.mines.edu
-rw------- 1 tkaiser tkaiser      8 Jan 11 14:08 mynodes.3589.mio.mines.edu

[tkaiser@mio cwp]$ cat mynodes.3589.mio.mines.edu
n45
n46
```

```
[tkaiser@mio cwp]$ cat allmynodes.3589.mio.mines.edu
n45
n45
n45
n45
n45
n45
n45
n45
n45
n46
n46
n46
n46
n46
n46
n46
n46
```

The “real” output

```
[tkaiser@mio cwp]$ cat outx.3589.mio.mines.edu
running /home/tkaiser/data/tests/cwp/c_ex00
Hello from 0 of 16 on n46
Hello from 6 of 16 on n46
Hello from 1 of 16 on n46
Hello from 5 of 16 on n46
Hello from 12 of 16 on n45
Hello from 2 of 16 on n46
Hello from 10 of 16 on n45
Hello from 4 of 16 on n46
Hello from 8 of 16 on n45
Hello from 9 of 16 on n45
Hello from 3 of 16 on n46
Hello from 11 of 16 on n45
Hello from 7 of 16 on n46
Hello from 13 of 16 on n45
Hello from 15 of 16 on n45
Hello from 14 of 16 on n45
[tkaiser@mio cwp]$
```

Running a serial program with mpiexec

export MYPROGRAM=info_p

```
[tkaiser@mio cwp]$ qsub batch1
3590.mio.mines.edu
[tkaiser@mio cwp]$
```

```
[tkaiser@mio cwp]$ ls -lt
total 1792
-rw------- 1 tkaiser tkaiser      0 Jan 11 14:13 errx.3590.mio.mines.edu
-rw------- 1 tkaiser tkaiser     44 Jan 11 14:13 outx.3590.mio.mines.edu
-rw------- 1 tkaiser tkaiser     64 Jan 11 14:13 allmynodes.3590.mio.mines.edu
-rw------- 1 tkaiser tkaiser      8 Jan 11 14:13 mynodes.3590.mio.mines.edu
-rw------- 1 tkaiser tkaiser     36 Jan 11 14:13 n46_00051792
-rw------- 1 tkaiser tkaiser     36 Jan 11 14:13 n46_00051793
-rw------- 1 tkaiser tkaiser     36 Jan 11 14:13 n46_00051794
-rw------- 1 tkaiser tkaiser     36 Jan 11 14:13 n46_00051795
-rw------- 1 tkaiser tkaiser     36 Jan 11 14:13 n46_00051796
-rw------- 1 tkaiser tkaiser     36 Jan 11 14:13 n46_00051797
-rw------- 1 tkaiser tkaiser     36 Jan 11 14:13 n46_00051798
-rw------- 1 tkaiser tkaiser     36 Jan 11 14:13 n46_00051799
-rw------- 1 tkaiser tkaiser     36 Jan 11 14:13 n47_00051784
-rw------- 1 tkaiser tkaiser     36 Jan 11 14:13 n47_00051785
-rw------- 1 tkaiser tkaiser     36 Jan 11 14:13 n47_00051786
-rw------- 1 tkaiser tkaiser     36 Jan 11 14:13 n47_00051787
-rw------- 1 tkaiser tkaiser     36 Jan 11 14:13 n47_00051788
-rw------- 1 tkaiser tkaiser     36 Jan 11 14:13 n47_00051789
-rw------- 1 tkaiser tkaiser     36 Jan 11 14:13 n47_00051790
-rw------- 1 tkaiser tkaiser     36 Jan 11 14:13 n47_00051791
```

```
[tkaiser@mio cwp]$ cat n46_00051792
Python says hello from 51792 on n46
```

Our Batch file, batch.pbs

```
#!/bin/bash
#PBS -l nodes=2:ppn=8
#PBS -l walltime=08:00:00
#PBS -N test_1
#PBS -o outx.$PBS_JOBID
#PBS -e errx.$PBS_JOBID
#PBS -r n
#PBS -V
##PBS -m abe
##PBS -M tkaiser@mines.edu
#-----------------------------------------------
cd $PBS_O_WORKDIR

# get a short and full list of all of my nodes
sort -u  $PBS_NODEFILE > mynodes.$PBS_JOBID
sort   $PBS_NODEFILE > allmynodes.$PBS_JOBID

# run each program, one after the other

for P in c_ex00 f_ex00 info_p info_c info_f ; do
  export MYPROGRAM=$P
  echo "running" $PBS_O_WORKDIR/$MYPROGRAM
  mpiexec -np 16 $PBS_O_WORKDIR/$MYPROGRAM
done
```

Another Parallel Run Command

Mio has a second parallel run command not connected to MPI: beorun

```
BEORUN(1)                                                    BEORUN(1)

NAME
       beorun - Run a job on a Scyld cluster using dynamically selected nodes.

SYNOPSIS
       beorun   [  -h,  --help  ] [ -V, --version ] command [ command-args... ] [ --map node list ] [ --all-cpus ] [
       --all-nodes ]
           [ --np processes ] [ --all-local ] [ --no-local ] [ --exclude node list ]

DESCRIPTION
       The beorun program runs the specified program on a dynamically selected set of cluster nodes. It generates  a
       Beowulf  job map from the currently installed beomap scheduler, and starts the program on each node specified
       in the map.
```

beorun --no-local --map 21:22:24 ./info_c

beorun

- Works on the head node (bad idea)
- Can also work inside a batch script
  - Use the batch system to allocate a set of nodes
  - Create a node list for the command from your set of nodes (tricky)
  - Run the command

One way to create a list of nodes

```
sort   $PBS_NODEFILE > allmynodes.$PBS_JOBID
export NLIST=`awk '{ sum=sum":"substr($1,2);  }; END { print substr(sum,2) }' \
allmynodes.$PBS_JOBID`
```

```
[tkaiser@mio cwp]$ cat allmynodes.3590.mio.mines.edu
n46
n46
n46
n46
n46
n46
n46
n46
n46
n46
n47
n47
n47
n47
n47
n47
n47
n47
n47
```

46:46:46:46:46:46:46:46:47:47:47:47:47:47:47:47

beorun in a batch file

batch2

```
#!/bin/bash
#PBS -l nodes=2:ppn=8
#PBS -l walltime=08:00:00
#PBS -N test_2
#PBS -o outx.$PBS_JOBID
#PBS -e errx.$PBS_JOBID
#PBS -r n
#PBS -V
##PBS -m abe
##PBS -M tkaiser@mines.edu
#-----------------------------------------------
cd $PBS_O_WORKDIR

# get a short and full list of all of my nodes
sort -u  $PBS_NODEFILE > mynodes.$PBS_JOBID
sort   $PBS_NODEFILE > allmynodes.$PBS_JOBID

# select one of these programs to run
# the rest of the lines are commented
# out
#export MYPROGRAM=info_p
#export MYPROGRAM=info_c
export MYPROGRAM=info_f

# create a list of nodes that can be used by the
# Mio command beorun.  This next line takes the
# names form the file and strips off the "n" to
# just give us an node number.  It then puts a
# ":" between the numbers
export NLIST=`awk  '{ sum=sum":"substr($1,2);  }; END { print substr(sum,2) }' allmynodes.$PBS_JOBID`
echo $NLIST
echo "running" $PBS_O_WORKDIR/$MYPROGRAM
beorun --no-local --map $NLIST  $PBS_O_WORKDIR/$MYPROGRAM
```

Results

```
[tkaiser@mio cwp]$ ls -lt n*_* *mio.mines.edu
-rw------- 1 tkaiser tkaiser   0 Jan 11 15:02 errx.3599.mio.mines.edu
-rw------- 1 tkaiser tkaiser  45 Jan 11 15:02 n44_00055776
-rw------- 1 tkaiser tkaiser  45 Jan 11 15:02 n44_00055777
-rw------- 1 tkaiser tkaiser  45 Jan 11 15:02 n44_00055778
-rw------- 1 tkaiser tkaiser  45 Jan 11 15:02 n44_00055779
-rw------- 1 tkaiser tkaiser  45 Jan 11 15:02 n44_00055780
-rw------- 1 tkaiser tkaiser  45 Jan 11 15:02 n44_00055781
-rw------- 1 tkaiser tkaiser  45 Jan 11 15:02 n44_00055782
-rw------- 1 tkaiser tkaiser  45 Jan 11 15:02 n44_00055783
-rw------- 1 tkaiser tkaiser  45 Jan 11 15:02 n45_00055784
-rw------- 1 tkaiser tkaiser  45 Jan 11 15:02 n45_00055785
-rw------- 1 tkaiser tkaiser  45 Jan 11 15:02 n45_00055786
-rw------- 1 tkaiser tkaiser  45 Jan 11 15:02 n45_00055787
-rw------- 1 tkaiser tkaiser  45 Jan 11 15:02 n45_00055788
-rw------- 1 tkaiser tkaiser  45 Jan 11 15:02 n45_00055789
-rw------- 1 tkaiser tkaiser  45 Jan 11 15:02 n45_00055790
-rw------- 1 tkaiser tkaiser  45 Jan 11 15:02 n45_00055791
-rw------- 1 tkaiser tkaiser  92 Jan 11 15:02 outx.3599.mio.mines.edu
-rw------- 1 tkaiser tkaiser  64 Jan 11 15:02 allmynodes.3599.mio.mines.edu
-rw------- 1 tkaiser tkaiser   8 Jan 11 15:02 mynodes.3599.mio.mines.edu


[tkaiser@mio cwp]$ cat n44_00055776
 Fortran says hello from       55776  on n44
[tkaiser@mio cwp]$
```

Selecting Particular nodes

- Replace the "#PBS -l nodes=2:ppn=8" line in your script with...
  - #PBS -l nodes=2:ppn=12
    - requests 2 12 core nodes
  - #PBS -l nodes=n4:ppn=8
    - requests node n4
  - Can also be done on the qsub command line
    - qsub batch1 -l nodes="n1:ppn=8+n2:ppn=8"
- Listed nodes are separated by a "+"

Running Interactive

Interactive Runs

- It is possible to grab a collection of nodes and run parallel interactively
- Nodes are yours until you quit
- Might be useful during development
- Running a number of small parallel jobs
- Required (almost) for running parallel debuggers

Multistep Process

- Ask for the nodes using qsub
  - `qsub -I -V -l nodes=1:ppn=8`
  - `qsub -I -V -l nodes=1:ppn=12`
  - `qsub -I -V -l nodes=n4:ppn=8`
- You are automatically Logged on to the nodes you are given
- cat $PBS_NODEFILE to see list
- Go to your working directory
- Run your job(s)

Don’t abuse it, lest you be shot.

Example

```
[tkaiser@ra ~/guide]$qsub -I -V -l nodes=1 -l naccesspolicy=singlejob -l walltime=00:15:00
qsub: waiting for job 1280.ra.mines.edu to start
qsub: job 1280.ra.mines.edu ready

[tkaiser@compute-9-8 ~]$cd guide
[tkaiser@compute-9-8 ~/guide]$mpiexec -n 4 ./c_ex00
Hello from 0 of 4 on compute-9-8.local
Hello from 1 of 4 on compute-9-8.local
Hello from 2 of 4 on compute-9-8.local
Hello from 3 of 4 on compute-9-8.local
[tkaiser@compute-9-8 ~/guide]$
[tkaiser@compute-9-8 ~/guide]$mpiexec -n 16 ./c_ex00
Hello from 1 of 16 on compute-9-8.local
Hello from 0 of 16 on compute-9-8.local
Hello from 3 of 16 on compute-9-8.local
Hello from 4 of 16 on compute-9-8.local
...
...
Hello from 15 of 16 on compute-9-8.local
Hello from 2 of 16 on compute-9-8.local
Hello from 6 of 16 on compute-9-8.local
[tkaiser@compute-9-8 ~/guide]$
[tkaiser@compute-9-8 ~/guide]$exit
logout

qsub: job 1280.ra.mines.edu completed
[tkaiser@ra ~/guide]$
```

Ensures exclusive access

More Scripting

Examples in oneday/scripts2

What the ??? How did you do that?

- Normal scripts review (script01)
- Getting output before the job finishes (script02)
- N < 8 tasks per node and other mappings (script03)
- Different executables on different nodes (script04)
- Notifications (script05)
- Keeping a record of what you did (script05)
- Multiple mpiexec commands per script (script06)
- Using local disk space (script07)

All of this came from users questions “How can I????”

Our C Example #1

```
/**********************************************************
!  This is a simple hello world program. Each processor
!  prints out its rank, number of tasks, and processor name.
**********************************************************/


 #include <stdio.h>
 #include <stdlib.h>
 #include <mpi.h>

 int main(int argc,char *argv[])
 {
     int myid, numprocs;
     FILE *f1;
     int i,resultlen;
     char myname[MPI_MAX_PROCESSOR_NAME];
     MPI_Init(&argc,&argv);
     MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
     MPI_Comm_rank(MPI_COMM_WORLD,&myid);
     MPI_Get_processor_name(myname,&resultlen);
     printf("C %4d of %4d says Hello from %s\n",myid, numprocs ,myname);
     MPI_Finalize();
 }
```

Our Fortran Example #1

```
!*****************************************************
!  This is a simple hello world program. Each processor
!  prints out its rank, number of tasks, and processor name.
!*****************************************************

      program hello
      include "mpif.h"
      character (len=MPI_MAX_PROCESSOR_NAME):: name
      call MPI_INIT( ierr )
      call MPI_COMM_RANK( MPI_COMM_WORLD, myid, ierr )
      call MPI_COMM_SIZE( MPI_COMM_WORLD, numprocs, ierr )
      call MPI_Get_processor_name(name,nlen,ierr)
      write(*,10)myid, numprocs ,trim(name)
10    format("Fort says Hello from",i4," of ",i4 " on ",a)
      call MPI_FINALIZE(ierr)
      stop
      end
```

A Simple Script for a MPI job

```
!/bin/bash
#PBS -l nodes=1:ppn=8
#PBS -l walltime=00:02:00
#PBS -N testIO
#PBS -o stdout
#PBS -e stderr
#PBS -V
#PBS -m abe
#PBS -M tkaiser@mines.edu
#-----------------------
cd $PBS_O_WORKDIR
sort -u $PBS_NODEFILE > shortlist
mpiexec -n 8 ./c01
```

Scripts contain comments designated with a # that are interpreted by PBS and normal shell commands

A Simple Script for a MPI job

Command	Comment
!/bin/bash	We will run this job using the "bash" shell
#PBS -l nodes=1:ppn=8	We want 1 node with 8 processors for a total of 8 processors ppn should always be equal to 8 or 12
#PBS -l walltime=00:02:00	We will run for up to 2 minutes
#PBS -N testIO	The name of our job is testIO
#PBS -o stdout	Standard output from our program will go to a file stdout
#PBS -e stderr	Error output from our program will go to a file stderr
#PBS -V	Very important! Exports all environment variables from the submitting shell into the batch shell.
#PBS -m abe	Send mail on abort, begin, end use n to suppress
#PBS -M tkaiser@mines.edu	Where to send email to me (cell phone?)
#----------------------	Not important. Just a separator line.
cd $PBS_O_WORKDIR	Very important! Go to directory $PBS_O_WORKDIR which is the directory which is where our script resides
sort -u $PBS_NODEFILE > shortlist	Save a sorted unique list of nodes
mpiexec -n 8 ./c01	Run the MPI program c01 on 8 computing cores.

Where's my Output?

- Standard Error and Standard Out don't show up until after your job is completed
- Output is temporarily stored in /tmp on your nodes
  - You can ssh to your nodes and see your output
  - Looking into changing this behavior, or at least make it optional
- You can pipe output to a file:
  - mpiexec -n 8 ./c01 Becomes
  - mpiexec -n 8 ./c01 > myjob.$PBS_JOBID

Here we use the variable $PBS_JOBID to tag our output with the job number

Using $PBS_JOBID

script02

```
#!/bin/bash
#PBS -l nodes=1:ppn=8
#PBS -l walltime=00:01:00
#PBS -N testIO
#PBS -o out.$PBS_JOBID
#PBS -e err.$PBS_JOBID
#PBS -V
#PBS -m abe
##PBS -M tkaiser@mines.edu
#-----------------------
cd $PBS_O_WORKDIR
sort -u $PBS_NODEFILE > shortNodeList
mpiexec -n 8 ./c01 > myjob.$PBS_JOBID
```

Assume we want 2 nodes but only want 4 tasks per node

- Node request line becomes
  - #PBS -l nodes=2:ppn=8
- We can create a file that has a mapping of programs to nodes
- The file that contains the mapping is used on the mpiexec line instead of the program name
- We have created a script "match" to help with this kind of thing
- Lots more details at
  - http://geco.mines.edu/guide/mpmd.shtml

4 copies of "c00" on 2 nodes

script03

```
#!/bin/bash
#PBS -l nodes=2:ppn=8
#PBS -l walltime=00:01:00
#PBS -N testIO
#PBS -o out.$PBS_JOBID
#PBS -e err.$PBS_JOBID
#PBS -V
##PBS -m abe
##PBS -M tkaiser@mines.edu
#-----------------------
cd $PBS_O_WORKDIR
sort -u $PBS_NODEFILE > shortlist

/opt/utility/match  shortlist -p"./c00" 4 > appfile
mpiexec  --app appfile
```

4 copies of "c00" on 2 nodes

```
C   0 of   8 says Hello from compute-9-29.local
C   1 of   8 says Hello from compute-9-29.local
C   2 of   8 says Hello from compute-9-29.local
C   3 of   8 says Hello from compute-9-29.local
C   4 of   8 says Hello from compute-9-30.local
C   5 of   8 says Hello from compute-9-30.local
C   6 of   8 says Hello from compute-9-30.local
C   7 of   8 says Hello from compute-9-30.local
```

Different apps and/or #s on nodes

script04

```
#!/bin/bash
#PBS -l nodes=2:ppn=8
#PBS -l walltime=00:01:00
#PBS -N testIO
#PBS -o out.$PBS_JOBID
#PBS -e err.$PBS_JOBID
#PBS -V
##PBS -m abe
##PBS -M tkaiser@mines.edu
#----------------------
cd $PBS_O_WORKDIR
sort -u $PBS_NODEFILE > shortlist

/opt/utility/match  shortlist -p"c01 f01" 4 8 > appfile.$PBS_JOBID
mpiexec  --app appfile.$PBS_JOBID
```

Different apps and/or #s on nodes

```
C    0 of   12 says Hello from compute-9-29.local
C    1 of   12 says Hello from compute-9-29.local
C    2 of   12 says Hello from compute-9-29.local
C    3 of   12 says Hello from compute-9-29.local

Fort says Hello from   4 of   12 on compute-9-30.local
Fort says Hello from   5 of   12 on compute-9-30.local
Fort says Hello from   6 of   12 on compute-9-30.local
Fort says Hello from   7 of   12 on compute-9-30.local
Fort says Hello from   8 of   12 on compute-9-30.local
Fort says Hello from   9 of   12 on compute-9-30.local
Fort says Hello from  10 of   12 on compute-9-30.local
Fort says Hello from  11 of   12 on compute-9-30.local


/opt/utility/match  shortlist -p"c01 f01" 4 8 > appfile.$PBS_JOBID
mpiexec  --app appfile.$PBS_JOBID
```

Notifications and Records

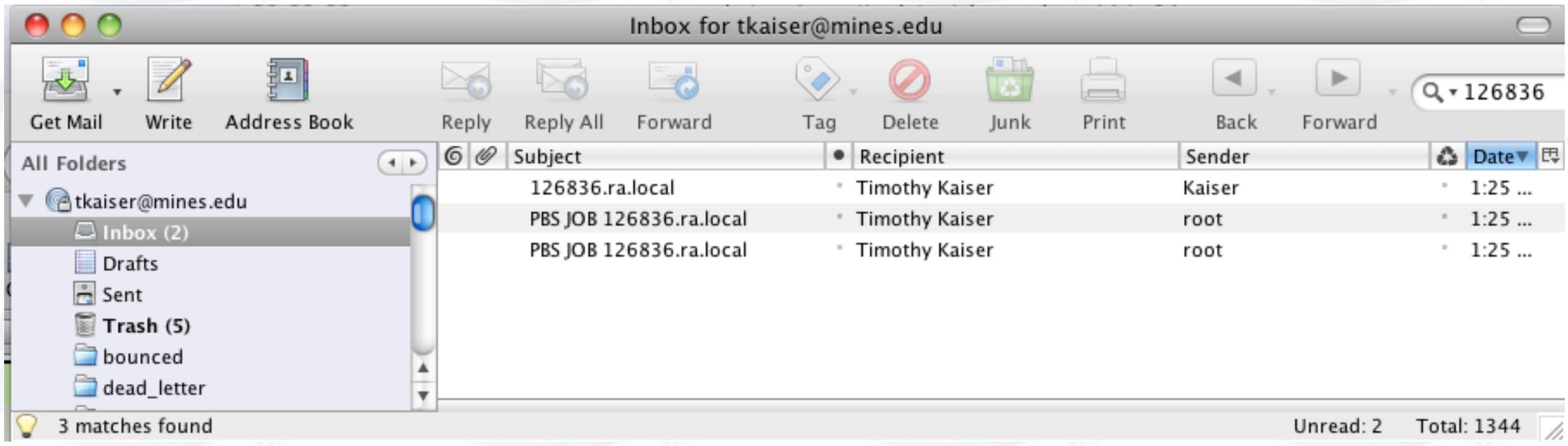
script05

```
#!/bin/bash
#PBS -l nodes=2:ppn=8
#PBS -l naccesspolicy=singleuser
#PBS -l walltime=00:05:00
#PBS -N testIO
#PBS -o out.$PBS_JOBID
#PBS -e err.$PBS_JOBID
#PBS -r n
#PBS -V
#PBS -m abe
#PBS -M joeminer@mines.edu
#------------------------------------------------
cd $PBS_O_WORKDIR


#our list of nodes
sort -u  $PBS_NODEFILE > mynodes.$PBS_JOBID

#save a copy of this sciprt (may strip comments)
cat $0 > runscript.$PBS_JOBID

#save our PBS environment, the path to our mpiexec command, and executable
printenv | grep PBS > env.$PBS_JOBID
which mpiexec >> env.$PBS_JOBID
which ./c01 >> env.$PBS_JOBID

#mail us the environment and other "stuff"
####  mail <  env.$PBS_JOBID -s $PBS_JOBID  $USER@mines.edu
ssh mio “mail <  $PBS_O_WORKDIR/env.$PBS_JOBID -s $PBS_JOBID  $USER@mines.edu”


mpiexec  -np 16 ./c01
```

How can I know when a particular script starts and exactly what is running?

Lots of records...

```
[tkaiser@ra quick]$ ls -l *126833*
-rw-rw-r-- 1 tkaiser tkaiser 1688 Mar 4 13:18 env.126833.ra.local
-rw------- 1 tkaiser tkaiser    0 Mar 4 13:18 err.126833.ra.local
-rw-rw-r-- 1 tkaiser tkaiser   38 Mar 4 13:18 mynodes.126833.ra.local
-rw------- 1 tkaiser tkaiser 1018 Mar 4 13:18 out.126833.ra.local
-rw-rw-r-- 1 tkaiser tkaiser  716 Mar 4 13:18 runscript.126833.ra.local
[tkaiser@ra quick]$
```



Running multiple mpi jobs in a script

- If you want the jobs to run after each other than just put in multiple mpiexec lines
- If you want to run two jobs at the same time on the same node things can be tricky
- Add a "&" to the end of the mpiexec command line. This allows the second program to start
- Add a guard "program" to check if your real programs are done

c02.c and f02.f90

- Hello world but takes two optional command line arguments
  - # of seconds to sleep for task 0
  - Name for file for task 0 to create
- Programs also print out start times and time the file is created

Our C Example #2

```
#include <stdio.h>
#include <stdlib.h>
#include <mpi.h>
#include <time.h>
void dostuff(int myid,int argc,char *argv[]);
struct tm *ptr;
time_t tm;
int main(int argc,char *argv[])
{
    int myid, numprocs;
    FILE *f1;
    int i,resultlen;
    char myname[MPI_MAX_PROCESSOR_NAME];
    MPI_Init(&argc,&argv);
    MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD,&myid);
    MPI_Get_processor_name(myname,&resultlen);
    tm = time(NULL); ptr = localtime(&tm);
    printf("C %4d of %4d says Hello from %s %s\n",
          myid, numprocs ,myname,asctime(ptr));
    dostuff(myid,argc,argv);
    MPI_Finalize();
}
```

```
/*************************************************************
Hello world but here we add a subroutine that can take
command line arguments and optionally sleep for some number
of seconds and create a file.
*************************************************************/
#include <unistd.h>
#include <string.h>
void dostuff(int myid,int argc,char *argv[]) {
    int isleep;
    char aname[20];
    FILE *f;
    if(myid == 0) {
        if(argc > 1) {
            sscanf(argv[1],"%d",&isleep);
            sleep((unsigned int) isleep);
        }
        if(argc > 2) {
            strncpy(aname,argv[2],(size_t)19);
            f=fopen(aname,"w");
            tm = time(NULL); ptr = localtime(&tm);
            fprintf(f,"hello %s \n",asctime(ptr));
            fclose(f);
        }
    }
    MPI_Barrier(MPI_COMM_WORLD);
}
```

Our Fortran Example #2

```
      program hello
      include "mpif.h"
      character (len=MPI_MAX_PROCESSOR_NAME):: name
      character (len=8) :: datestr
      character (len=10) :: timestr
      call MPI_INIT( ierr )
      call MPI_COMM_RANK( MPI_COMM_WORLD, myid, ierr )
      call MPI_COMM_SIZE( MPI_COMM_WORLD, numprocs, ierr )
      call MPI_Get_processor_name(name,nlen,ierr)
      call date_and_time(datestr,timestr)
      write(*,10)myid, numprocs ,trim(name),datestr,timestr
10    format("Fort says Hello from",i4," of ",i4 " on ",a,a15,a15)
      call dostuff(myid)
      call MPI_FINALIZE(ierr)
      stop
      end
      subroutine dostuff(myid)
      include "mpif.h"
      character(len=20) :: aline
      character (len=8) :: datestr
      character (len=10) :: timestr
      integer len,stat,isleep
      if(myid .eq. 0)then
        call get_command_argument(1, aline, len, stat)
        if(stat .eq. 0)then
          read(aline,*)isleep
          call sleep(isleep)
        endif
        call get_command_argument(2, aline, len, stat)
        if(stat .eq. 0)then
          call date_and_time(datestr,timestr)
          open(file=trim(aline),unit=17)
          write(17,"(3a12)")"hello",datestr,timestr
          close(17)
        endif
      endif
      call MPI_Barrier(MPI_COMM_WORLD, ierr )
      end subroutine
```

```
!*************************************************************
!  Hello world but here we add a subroutine that can take
!  command line arguments and optionally sleep for some number
!  of seconds and create a file.
!*************************************************************
```

Our Waiting Script

script06

```
[tkaiser@ra quick]$ cat script06
#!/bin/bash
#PBS -l nodes=1:ppn=8
#PBS -l walltime=00:05:00
#PBS -N testIO
#PBS -o stdout
#PBS -e stderr
#PBS -V
#PBS -m abe
##PBS -M tkaiser@mines.edu
#----------------------
cd $PBS_O_WORKDIR
sort -u $PBS_NODEFILE > shortlist

rm cent1 cent2

mpiexec -n 4 ./c02 30 cent1 &

mpiexec -n 4 ./f02 40 cent2 &
```

```
while [ ! -f cent1 ]
do
echo "looking for cent1" >> guard
sleep 10
done

while [ ! -f cent2 ]
do
echo "looking for cent2" >> guard
sleep 10
done
```

Output

```
C    0 of    4 says Hello from compute-9-30.local Thu Mar  4 14:36:20 2010
C    1 of    4 says Hello from compute-9-30.local Thu Mar  4 14:36:20 2010
C    2 of    4 says Hello from compute-9-30.local Thu Mar  4 14:36:20 2010
C    3 of    4 says Hello from compute-9-30.local Thu Mar  4 14:36:20 2010
Fort says Hello from   0 of    4 on compute-9-30.local       20100304     143620.610
Fort says Hello from   1 of    4 on compute-9-30.local       20100304     143620.613
Fort says Hello from   2 of    4 on compute-9-30.local       20100304     143620.613
Fort says Hello from   3 of    4 on compute-9-30.local       20100304     143620.611
```

```
[tkaiser@ra quick]$ cat cent1
hello Thu Mar  4 14:36:50 2010

[tkaiser@ra quick]$ cat cent2
       hello    20100304  143650.618
[tkaiser@ra quick]$
```

Using Local Disk Space

- Each node on Mio has a local space that you can use for temporary files: `/scratch`.
- Don't use /tmp
- Size depends on type of nodes:

Node	Cores	DISK Free (/scratch) GB
0 to 31	8	128
32 to 47	12	849

- We will talk about moving data from a script and staging of data before a run

Using local space

- Create your subdirectories in /scratch
- Run your program, writing to your directory
- If you want the files copy them to your working directory
- Delete the files and directories

Fortran Example # 3

```
!*************************************************************
!  This is a simple hello world program but each processor
!  writes out its rank and total number of processors
!  in the current MPI run to a file in /scratch
!*************************************************************

      program hello
      use ifport  ! ## needed for GETENV called below ##
      include "mpif.h"
      character (len=MPI_MAX_PROCESSOR_NAME):: nodename
      character (len=20) :: myname
      character (len=40) :: basedir,fname
      call MPI_INIT( ierr )
      call MPI_COMM_RANK( MPI_COMM_WORLD, myid, ierr )
      call MPI_COMM_SIZE( MPI_COMM_WORLD, numprocs, ierr )
! get the name of the node on which i am running
      call MPI_Get_processor_name(nodename,nlen,ierr)
! get my username
      CALL GETENV ("LOGNAME",myname)
!  for every mpi task...
!  create a unique file name based on a base directory
!  name, a username, "f_", and the mpi process id
      basedir="/scratch/"
      write(fname,'(a,a,"/f_",i5.5)'),trim(basedir),trim(myname),myid
! echo the full path to the file for each mpi task
      write(*,*)"opening file ",trim(fname)
! open the file
      open(10,file=fname)
! write the mpi process id to the file along with the node name
      write(10,'("Hello from",i4," on ",a)')myid,trim(nodename)
      close(10)
      call MPI_FINALIZE(ierr)
      stop
      end
```

Part 1: Creating Files

script07

```
#!/bin/bash
#PBS -l nodes=2:ppn=8
#PBS -l naccesspolicy=singleuser
#PBS -l walltime=00:05:00
#PBS -N testIO
#PBS -o testIOo.$PBS_JOBID
#PBS -e testIOe.$PBS_JOBID
#PBS -r n
#PBS -V
#-----------------------------------------------
cd $PBS_O_WORKDIR

# Save a nicely sorted list of nodes with no repeats
sort -u  $PBS_NODEFILE > shortlist

# Get our list of nodes
export nlist=`cat shortlist`

# For each node...
for i in $nlist
do
# Create my temporary directory in /scratch on each node
  ssh $i mkdir /scratch/$USER
done


# Run the program that writes to the temporary directory
mpiexec -np 16 ./f03
```

Part 2: Moving the files and clean up

```
#for each node...
for i in $nlist
do
# Copy files from /scratch on each node back to
# my working directory with a subdirectory for each node.
  mkdir -p $PBS_O_WORKDIR/$i
  scp -r $USER@$i:/scratch/$USER $USER@ra.mines.edu:$PBS_O_WORKDIR/$i
##### or #####
# ssh -r $USER@$i cp -r /scratch/$USER $PBS_O_WORKDIR/$i


# Remove the temporary directory
  ssh $i rm -r /scratch/$USER
done
```

Our Final output...

```
-rw-------  1 tkaiser tkaiser    990 Mar  4 15:59 testIOo.126879.ra.local
drwxrwxr-x  3 tkaiser tkaiser   4096 Mar  4 15:59 compute-9-30.local
drwxrwxr-x  3 tkaiser tkaiser   4096 Mar  4 15:59 fatcompute-10-11.local
-rw-rw-r--  1 tkaiser tkaiser     42 Mar  4 15:59 shortlist
-rw-------  1 tkaiser tkaiser      0 Mar  4 15:59 testIOe.126879.ra.local

[tkaiser@ra quick]$ ls -R compute-9-30.local
compute-9-30.local:
tkaiser

compute-9-30.local/tkaiser:
f_00000  f_00001  f_00002  f_00003  f_00004  f_00005  f_00006  f_00007

[tkaiser@ra quick]$ ls -R fatcompute-10-11.local
fatcompute-10-11.local:
tkaiser

fatcompute-10-11.local/tkaiser:
f_00008  f_00009  f_00010  f_00011  f_00012  f_00013  f_00014  f_00015


[tkaiser@ra quick]$ cat fatcompute-10-11.local/tkaiser/f_00015
Hello from  15 on fatcompute-10-11.local
[tkaiser@ra quick]$
```

Getting data to local directories

- Command run on the compute node:
  - cp sourceFile /scratch/mydir
  - scp mio:/full/path/to/sourceFile /scratch/mydir
- Compute node to compute node “n36”
  - scp /scratch/mydir/myfile n36: /scratch/mydir/myfile
- Command run on Mio to compute node “n32”
  - scp /full/path/to/file n36:/scratch/mydir

File Systems on Mio

- Four partitions
  - $HOME - should be kept very small, having only start up scripts and other simple scripts
  - $DATA - Should contain programs you have built for use in your research, small data sets and run scripts.
  - $SCRATCH - The main area for running applications. Output from parallel runs should be done to this directory. NOT BACKED UP
  - $SETS - This area is for large data sets, mainly read only, that will be use over a number of months. NOT BACKED UP

Data - Best Practices

- Home directory needs to be small, less than 100 Mbytes

```
[tkaiser@mio ~]$ du -sh
70M .
```

- $DATA can contain programs you build, scripts but not large data sets
- $SCRATCH can contain programs you build, scripts and should be used for program output

Running in $SCRATCH

script08

```
#!/bin/bash
#PBS -l nodes=2:ppn=8
#PBS -l naccesspolicy=singleuser
#PBS -l walltime=00:05:00
#PBS -N testIO
#PBS -o testIOo.$PBS_JOBID
#PBS -e testIOe.$PBS_JOBID
#PBS -r n
#PBS -V
#-----------------------------------------------
cd $PBS_O_WORKDIR

# Create a directory in $SCRATCH for your run
# Here we create a directory "tagged" with our
# job id
mkdir $SCRATCH/$PBS_JOBID
echo created $SCRATCH/$PBS_JOBID

# Next you might want to copy data to the
# $SCRATCH/$PBS_JOBID directory
cp $0 *data $SCRATCH/$PBS_JOBID

# go to the directory
cd $SCRATCH/$PBS_JOBID

# Run a program that "lives" in your starting
# directory but the output goes into our
# $SCRATCH/$PBS_JOBID directory
mpiexec -np 8 $PBS_O_WORKDIR/mpiio > out.$PBS_JOBID
```

Results

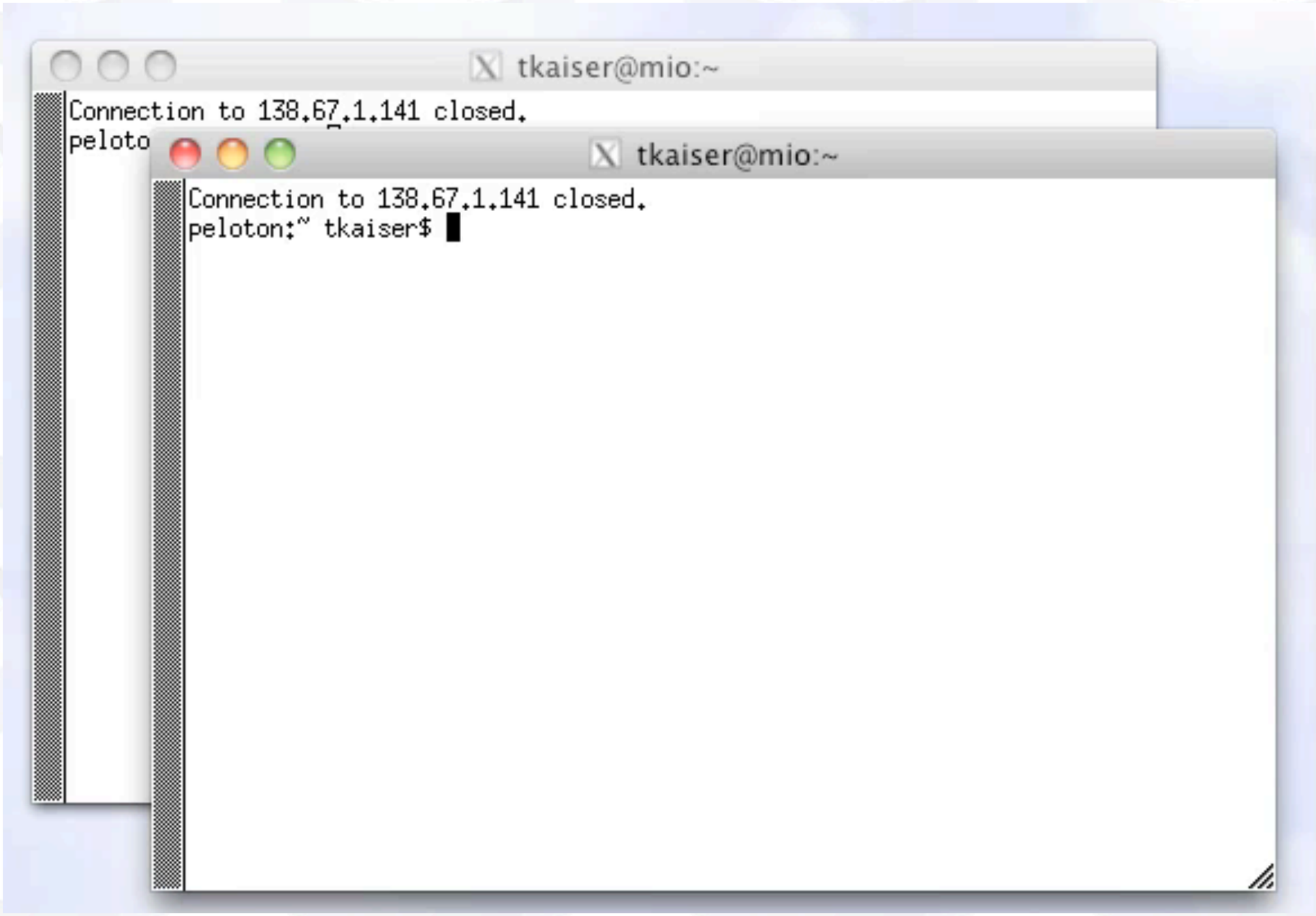
```
[tkaiser@mio data]$ qsub script08
4407.mio.mines.edu

[tkaiser@mio data]$ ls -lt testIO*
-rw------- 1 tkaiser tkaiser  0 Jan 21 10:35 testIOe.4407.mio.mines.edu
-rw------- 1 tkaiser tkaiser 58 Jan 21 10:35 testIOo.4407.mio.mines.edu
[tkaiser@mio data]$

[tkaiser@mio data]$ cat testIO*
created /panfs/storage/scratch/tkaiser/4407.mio.mines.edu

[tkaiser@mio data]$ cd /panfs/storage/scratch/tkaiser/4407.mio.mines.edu


[tkaiser@mio 4407.mio.mines.edu]$ ls -lt
total 1748
-rw------- 1 tkaiser tkaiser    1303 Jan 21 10:30 out.4407.mio.mines.edu
-rw------- 1 tkaiser tkaiser 1500012 Jan 21 10:30 mpiio_dat_008_0100_0050_0075
-rw------- 1 tkaiser tkaiser      25 Jan 21 10:30 dummy.data
-rwx------ 1 tkaiser tkaiser     758 Jan 21 10:30 4407.mio.mines.edu.SC
[tkaiser@mio 4407.mio.mines.edu]$
```

You may want to copy the testIO* files to the new directory after the job finishes

Graphics Sessions

- You may want to get X-windows output from a node in an interactive job
- Multi step process
- Start with two local windows
- In both ssh -Y mio
- In first window start your interactive session
- You are logged on to a node but can't get a display from that window
- Find out what node you are on - hostname
- In the second window ssh -Y "hostname". This puts you on the compute node.
- You will be able to get X-window output from the second window

Graphics Sessions



Useful commands

- showq = http://mio.mines.edu/jobs
- showstart = http://mio.mines.edu/jobs (click on #)
- /opt/utility/inuse = http://mio.mines.edu/inuse
- /opt/utility/nps
- /opt/utility/scpath
- /opt/utility/backup
- http://mio.mines.edu/ganglia/

showq

- like qstat but more details
- showq -h gives options

```
[tkaiser@mio cwp]$ showq
ACTIVE JOBS--------------------
JOBNAME            USERNAME      STATE  PROC   REMAINING            STARTTIME

3598                mdvorak    Running     8    12:03:14  Tue Jan 11 14:56:16
3600                mdvorak    Running    16    12:15:34  Tue Jan 11 15:08:36
3528                jgodwin    Running    64  1:15:00:53  Mon Jan 10 15:53:55
3540                jgodwin    Running    64  3:01:46:41  Wed Jan 12 02:39:43
3541                jgodwin    Running    64  3:02:32:29  Wed Jan 12 03:25:31
3542                jgodwin    Running    64  3:02:57:32  Wed Jan 12 03:50:34
3543                jgodwin    Running    64  3:02:58:53  Wed Jan 12 03:51:55
3584               cmmaupin    Running     8  5:03:37:01  Tue Jan 11 12:30:03

     8 Active Jobs     352 of  432 Processors Active (81.48%)
                        44 of   46 Nodes Active      (95.65%)

IDLE JOBS----------------------
JOBNAME            USERNAME      STATE  PROC     WCLIMIT            QUEUETIME

3618                jusander      Idle    40  2:00:00:00  Tue Jan 11 17:57:12
3619                jusander      Idle    40  2:00:00:00  Tue Jan 11 17:57:48

2 Idle Jobs

BLOCKED JOBS----------------
JOBNAME            USERNAME      STATE  PROC     WCLIMIT            QUEUETIME

3544                jgodwin  BatchHold    64  3:08:00:00  Mon Jan 10 10:11:24
3545                jgodwin  BatchHold    64  3:08:00:00  Mon Jan 10 10:11:25
3546                jgodwin  BatchHold    40  3:08:00:00  Mon Jan 10 10:11:25

Total Jobs: 13   Active Jobs: 8   Idle Jobs: 2   Blocked Jobs: 3
[tkaiser@mio cwp]$
```

http://mio.mines.edu

Job Status

http://mio.mines.edu/jobs/

Google

Job Status at: 2011 01 12 08:57

ACTIVE JOBS----------

JOBNAME	USERNAME	STATE	PROC	REMAINING	STARTTIME
3598	mdvorak	Running	8	11:58:35	Tue Jan 11 14:56:16
3600	mdvorak	Running	16	12:10:55	Tue Jan 11 15:08:36
3528	jgodwin	Running	64	1:14:56:14	Mon Jan 10 15:53:55
3540	jgodwin	Running	64	3:01:42:02	Wed Jan 12 02:39:43
3541	jgodwin	Running	64	3:02:27:50	Wed Jan 12 03:25:31
3542	jgodwin	Running	64	3:02:52:53	Wed Jan 12 03:50:34
3543	jgodwin	Running	64	3:02:54:14	Wed Jan 12 03:51:55
3584	cmmaupin	Running	8	5:03:32:22	Tue Jan 11 12:30:03

8 Active Jobs 352 of 432 Processors Active (81.48%)
44 of 46 Nodes Active (95.65%)

IDLE JOBS----------

JOBNAME	USERNAME	STATE	PROC	WCLIMIT	QUEUETIME
3618	jusander	Idle	40	2:00:00:00	Tue Jan 11 17:57:12
3619	jusander	Idle	40	2:00:00:00	Tue Jan 11 17:57:48

2 Idle Jobs

BLOCKED JOBS----------

JOBNAME	USERNAME	STATE	PROC	WCLIMIT	QUEUETIME
3544	jgodwin	BatchHold	64	3:08:00:00	Mon Jan 10 10:11:24
3545	jgodwin	BatchHold	64	3:08:00:00	Mon Jan 10 10:11:25
3546	jgodwin	BatchHold	40	3:08:00:00	Mon Jan 10 10:11:25

Total Jobs: 13 Active Jobs: 8 Idle Jobs: 2 Blocked Jobs: 3

LIST ALL JOBS

- Web based version of showq
- Clicking on user gives just their jobs
- Clicking on Active job # gives details
- Clicking on Idle job # also gives "starttime"

Job Status

http://mio.mines.edu/jobs/index.cgi?job=3618

Google

Job Status at: 2011 01 12 09:02

LIST ALL JOBS

```
Job Id: 3618.mio.mines.edu
Job_Name = Std_Corr_Ints
Job_Owner = jusander@mio.mines.edu
job_state = Q
queue = batch
server = mio.mines.edu
Checkpoint = u
ctime = Tue Jan 11 17:57:12 2011
Error_Path = mio.mines.edu:/data/work/jusander/011011/correlation/E.$PBS_JOBID
Hold_Types = n
Join_Path = n
Keep_Files = n
Mail_Points = e
Mail_Users = jusander@mines.edu
mtime = Tue Jan 11 17:57:12 2011
Output_Path = mio.mines.edu:/data/work/jusander/011011/correlation/O.$PBS_JOBID
Priority = 0
qtime = Tue Jan 11 17:57:12 2011
Rerunable = False
Resource_List.nodect = 5
Resource_List.nodes = 5:ppn=8
Resource_List.walltime = 48:00:00
etime = Tue Jan 11 17:57:12 2011
submit_args = run
Walltime.Remaining = -129467533
fault_tolerant = False

**********
job 3618 requires 40 procs for 2:00:00:00
Earliest start in         12:06:36 on Wed Jan 12 21:08:36
Earliest completion in  2:12:06:36 on Fri Jan 14 21:08:36
Best Partition: DEFAULT

**********
```

LIST ALL JOBS

Information for Idle Job 3618

Estimated start time

Job Status

http://mio.mines.edu/jobs/index.cgi?job=3600

Google

Job Status at: 2011 01 12 09:06

LIST ALL JOBS

```
Job Id: 3600.mio.mines.edu
Job_Name = si_BSE_test2
Job_Owner = mdvorak@mio.mines.edu
resources_used.cput = 269:23:35
resources_used.mem = 1526984kb
resources_used.vmem = 5888760kb
resources_used.walltime = 17:56:55
job_state = R
queue = batch
server = mio.mines.edu
Checkpoint = u
ctime = Tue Jan 11 15:08:35 2011
Error_Path = mio.mines.edu:/panfs/storage/scratch/mdvorak/exciton/si/si_BSE_test2_error
exec_host = {n45: 8, n44: 8}
Hold_Types = n
Join_Path = n
Keep_Files = n
Mail_Points = abe
Mail_Users = mdvorak@mines.edu
mtime = Tue Jan 11 15:08:36 2011
Output_Path = mio.mines.edu:/panfs/storage/scratch/mdvorak/exciton/si/si_BSE_test2_output
Priority = 0
qtime = Tue Jan 11 15:08:35 2011
Rerunable = True
Resource_List.nodect = 2
Resource_List.nodes = 2:ppn=8
Resource_List.walltime = 30:00:00
session_id = 55998
etime = Tue Jan 11 15:08:35 2011
submit_args = si_submit2
start_time = Tue Jan 11 15:08:36 2011
Walltime.Remaining = 4333
start_count = 1
fault_tolerant = False
```

LIST ALL JOBS

63

Information for active Job 3600

showstart

Estimates when a particular job will start

```
[tkaiser@mio cwp]$ showstart 3619
job 3619 requires 40 procs for 2:00:00:00
Earliest start in       1:14:45:12 on Thu Jan 13 23:53:55
Earliest completion in  3:14:45:12 on Sat Jan 15 23:53:55
Best Partition: DEFAULT

[tkaiser@mio cwp]$
```

```
[tkaiser@mio inuse]$ /opt/utility/inuse
mio.mines.edu  --  2011 01 12 09:20
nodes in use: 44
             node  load    users
               n3     8   jgodwin
...
              n20     8   jgodwin
              n21     8   cmmaupin
...
              n46     8   mdvorak
              n47     8   jgodwin

open nodes= ['n1', 'n2']

active users:
cmmaupin jobs 1 on  1 nodes
    n21

jgodwin jobs 5 on  40 nodes
     n3      n4      n5      n6
     n8      n9     n10     n11
    n12     n13     n14     n15
    n16     n17     n18     n19
...
...
    n41     n42     n43     n47

mdvorak jobs 2 on  3 nodes
    n44     n45     n46
```

/opt/utility/inuse
or
http://mio.mines.edu/inuse

Shows usage by node and users along with open nodes

/opt/utility/nps

"ps" show what you have running and where

```
[joeuser@mio ~]$ /opt/utility/nps
  NODE UID        PID  PPID  C    SZ   RSS PSR STIME TTY          TIME CMD
master joeuser  8431  8275  0 09:39 pts/2    00:00:00 -bash
master joeuser  8506  8431  0 09:40 pts/2    00:00:00 ssh mio
master joeuser  8509  8507  0 09:40 ?        00:00:00 sshd: joeuser@pts/8
master joeuser  8510  8509  0 09:40 pts/8    00:00:00 -bash
master joeuser  8575  8574  0 09:40 pts/8    00:00:00 sh -c ps -Af | grep joeuser
master joeuser  8576  8575  0 09:40 pts/8    00:00:00 ps -Af
master joeuser  8577  8575  0 09:40 pts/8    00:00:00 grep joeuser
   n21 joeuser 45364 20699  0 Jan11 ?        00:00:00 [csh]
   n21 joeuser 45435 45364  0 Jan11 ?        00:00:00 [3584.mio.mines.]
   n21 joeuser 45631 45435 99 Jan11 ?        20:58:57 [dftb+]
   n21 joeuser 45638 45631  0 Jan11 ?        00:00:00 [dftb+]
   n21 joeuser 45639 45631 92 Jan11 ?        19:32:08 [dftb+]
   n21 joeuser 45640 45631 92 Jan11 ?        19:32:31 [dftb+]
   n21 joeuser 45641 45631 92 Jan11 ?        19:33:04 [dftb+]
   n21 joeuser 45642 45631 92 Jan11 ?        19:33:15 [dftb+]
   n21 joeuser 45643 45631 92 Jan11 ?        19:32:28 [dftb+]
   n21 joeuser 45644 45631 92 Jan11 ?        19:32:54 [dftb+]
   n21 joeuser 45645 45631 92 Jan11 ?        19:33:09 [dftb+]
[joeuser@mio ~]$
```

/opt/utility/scpath

- Gives a fully qualified path (URL) for use with scp
  - `UserName@MachineName:/full/path/to/file`
- Recommend you copy this script to other machines
- scp becomes "copy/paste"

```
[tkaiser@mio cwp]$ ls
batch1  batch2  c_ex00.c  cwp.tar  doit  f_ex00.f  info.c  info.f90  info.py  inuse  makefile
[tkaiser@mio cwp]$
[tkaiser@mio cwp]$ scpath
tkaiser@mio.mines.edu:/home/tkaiser/data/tests/cwp
[tkaiser@mio cwp]$
[tkaiser@mio cwp]$ scpath cwp.tar
tkaiser@mio.mines.edu:/home/tkaiser/data/tests/cwp/cwp.tar
[tkaiser@mio cwp]$
[tkaiser@mio cwp]$ scpath i*
tkaiser@mio.mines.edu:/home/tkaiser/data/tests/cwp/info.c
tkaiser@mio.mines.edu:/home/tkaiser/data/tests/cwp/info.f90
tkaiser@mio.mines.edu:/home/tkaiser/data/tests/cwp/info.py
tkaiser@mio.mines.edu:/home/tkaiser/data/tests/cwp/inuse


[tkaiser@mio cwp]$ scp cwp.tar tkaiser@peloton.mines.edu:/Users/tkaiser
cwp.tar                                                                    100%
20KB  20.0KB/s   00:00
[tkaiser@mio cwp]$
```

/opt/utility/backup

- Nothing fancy but I use it often
- Short hand for creating a machine name and time stamped archive of a directory
- Lots of options

```
[tkaiser@mio cwp]$ ls
batch1  batch2  c_ex00.c  cwp.tar  doit  f_ex00.f  info.c  info.f90  info.py  inuse  makefile
[tkaiser@mio cwp]$
[tkaiser@mio cwp]$ backup
tar --exclude .svn --exclude CVS --exclude CVSROOT --exclude .DS_Store -czf ./
0112_103058mio.tgz *

[tkaiser@mio cwp]$ ls *tgz
0112_103058mio.tgz
[tkaiser@mio cwp]$
```

The following options are available:

```
-h              show this help and quit
-s              work silently
-n              don't time stamp the file name
-svn            include svn and CVS directories
-b directory    directory in which to place the archive
-w              put the archive in your scratch space (over rides -b)
-i file         file or directory to archive
-o name         replace the machine name in the archive with this name
-io file_name   combination of -i and -o using the same value for each
```