Introduction to DPCC

Patrick McGuigan
mcguigan@cse.uta.edu
Agenda

- Who, What, When, Where, and Why
- DPCC Resources
- Accessing System
- PBS Introduction
- MPI and PBS together
- PBS scheduling issues
Distributed and Parallel Computing Cluster

- Collaboration between CSE, Physics, and UTSW Medical Center
- Cluster on-line since 10/03
- SMP systems now in procurement
- Housed in 101 GACB
- http://www-hep.uta.edu/dpcc
DPCC Cluster Resources

• 97 Computers
  – 3 head nodes
  – 78 worker nodes
  – 10 RAID systems
  – SAN components
• Dual Intel Xeon Processors 2.4 or 2.6GHz
• 2GB RAM / machine
• 50+TB available in centralized storage
DPCC Resources (cont)

- Linux (RH 7.3)
- gcc (2.96)
- MPI-CH (1.2.5)
- OpenPBS (2.3)
- Other

Will install other packages if needed.
DPCC Accounts

• Usernames and Passwords available
• Web-site (http://www-hep.uta.edu/dpcc)
  – Operations Mailing list
• Head nodes:
  – master.dpcc.uta.edu
  – grid.dpcc.uta.edu
  – atlas.dpcc.uta.edu
• SSH (v2) – terminal services
• SFTP/SCP – file transfers
• Connections must be from on campus
DPCC Account details

• Use `yppasswd` to change password
• No quotas right now on `/home` (be sensible)
• Default shell is bash
  – Change with `ypchsh`
  – Edit `.bashrc` and `.bash_profile` to add directories to default search paths
    • `PATH=$PATH:/usr/local/mpich-1.2.5/bin`
    • `export $PATH`

• MPI installed at `/usr/local/mpich-1.2.5`
Running jobs in PBS

- PBS is batch scheduling system
- Jobs are scheduled on worker nodes
- Jobs are really shell scripts
- Shell script starts executables (compiled programs)
Minimal PBS job

$ cat helloworld.sh
  echo Hello World from $HOSTNAME

$ qsub helloworld.sh
  147819.master.dpcc.uta.edu

$ cat helloworld.sh.o147819
  Hello World from node38.cluster
Compiled program

```
$ cat helloworld.c
#include <stdio.h>
#include <unistd.h>
#include <string.h>

int main( argc, argv )
  int argc;
  char **argv;
{
  char host[256];
  int val;

  val = gethostname(host,255);
  if ( val != 0 ){
    strcpy(host,"UNKNOWN");
  }
  printf( "Hello world from node %s\n", host );
  return 0;
}

$ gcc –o helloworld.exe helloworld.c
```
Compiled program (WRONG)

$ qsub helloworld.exe
147822.master.dpcc.uta.edu

$ cat helloworld.exe.o147822

$ cat helloworld.exe.exe.e147822
  -bash: /var/spool/pbs/mom_priv/jobs/147822.mast.SC: cannot execute binary file
Compiled Program (cont.)

$ cat helloworld.sh
echo Hello World from $HOSTNAME

$ qsub helloworld.sh
147819.master.dpcc.uta.edu

$ cat helloworld.sh.o147819
Hello World from node38.cluster
Minimal PBS job (Right)

$ cat run_helloworld.sh

cd $PBS_O_WORKDIR
./helloworld.exe

$qsub run_helloworld.sh
147826.master.dpcc.uta.edu

$ cat run_helloworld.sh..o147826
Hello World from node38.cluster
Useful Information

• Use `set` command or `/usr/bin/env` in job
  – Helps debug path and other problems
  – Interesting variables:
    • `$PATH`
    • `$PWD`
    • `$HOSTNAME`
    • `$PBS_O_xxx`
PBS Commands

- **qsub**
  - Used to submit a job to PBS
- **Qstat**
  - Show status of jobs, queues, PBS system
- **Qdel**
  - Remove an executing or queued job
- **Qalter**
  - Alter attributes of a queued job
- **Pbsusers (no man pages)**
  - Provides summary information
qsub options

• -l switch used to specify needed resources
  – Number of nodes
    • nodes = x
    • Nodes = x:ppn=y
  – CPU time
    • cput=hh:mm:ss
  – Walltime
    • walltime=hh:mm:ss
  – Example:
    • -l walltime=100:00:00 –l cput=01:00:00 –l nodes=1

• See man pages for qsub, pbs_resources
qsub options (cont.)

• Output streams:
  – -e (error output path)
  – -o (standard output path)
  – -j (join error + output as either output or error)

• Mail options
  – -m [aben] when to mail (abort, begin, end, none)
  – -M who to mail

• Name of job
  – -N (15 printable characters MAX first is alphabetical)

• Which queue to submit job to
  – -q [name] Unimportant for now

• Environment variables
  – -v pass specific variables
  – -V pass all environment variables of qsub to job

• Additional attributes
  -w specify dependencies
Qsub options (cont.)

• Previous examples could have been specified as:
  
  
  qsub helloworld.sh -l walltime=100:00:00 –l cput=01:00:00 –l nodes=1
  
• Can embed option in PBS job with comments
  
  – $ cat helloworld_embed.sh

  #PBS -l nodes=1
  #PBS -m n
  #PBS -l cput=01:00:00
  #PBS -l walltime=100:00:00

  echo Hello World from $HOSTNAME
Using MPI and PBS together

• PBS can allocate multiple CPU’s for a job
  – $nodes=x$
    • (x processors on x different nodes)
  – $nodes=x:ppn=2$
    • (2x processors on x different nodes)
  – $nodes=x:ppn=2+y$
    • (2x+y processors on x+y nodes)

• PBS creates node file $PBS\_NODEFILE$.

• Job is responsible for launching parallel tasks.
```
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include "mpi.h"

int main ( argc, argv )
{
    int    argc;
    char **argv;

    int rank, size;
    char host[256];
    int val;

    val = gethostname(host,255);
    if ( val != 0 ){
        strcpy(host,"UNKNOWN");
    }

    MPI_Init( &argc, &argv );
    MPI_Comm_size( MPI_COMM_WORLD, &size );
    MPI_Comm_rank( MPI_COMM_WORLD, &rank );
    printf( "Hello world from node %s: process %d of %d
", host, rank, size );
    MPI_Finalize();
    return 0;
}
```
PBS script for MPI job

$ cat mpi_run.sh
#PBS -l nodes=6
#PBS -l walltime=00:30:00
#PBS -j oe
#PBS -o helloworld.out
#PBS -N helloword_mpi

NN=`cat $PBS_NODEFILE | wc -l`
echo "Processors received = "$NN

echo "script running on host `hostname`"
cd $PBS_O_WORKDIR
echo

echo "PBS NODE FILE"
cat $PBS_NODEFILE
echo

/usr/local/mpich-1.2.5/bin/mpirun -machinefile $PBS_NODEFILE \ -np $NN ~/pbs_examples/helloworld_mpi.exe
nodes=6 output

Processors received = 6
script running on host node47.cluster

PBS NODE FILE
node47.cluster
node46.cluster
node45.cluster
node44.cluster
node43.cluster
node42.cluster

Hello world from node node46.cluster: process 1 of 6
Hello world from node node45.cluster: process 2 of 6
Hello world from node node43.cluster: process 4 of 6
Hello world from node node44.cluster: process 3 of 6
Hello world from node node42.cluster: process 5 of 6
Hello world from node node47.cluster: process 0 of 6
nodes=3:ppn=2 (output)

Processors received = 6
script running on host node46.cluster

PBS NODE FILE
node46.cluster
node46.cluster
node46.cluster
node44.cluster
node44.cluster
node43.cluster
node43.cluster

Hello world from node node46.cluster: process 1 of 6
Hello world from node node43.cluster: process 4 of 6
Hello world from node node44.cluster: process 3 of 6
Hello world from node node44.cluster: process 2 of 6
Hello world from node node43.cluster: process 5 of 6
Hello world from node node46.cluster: process 0 of 6
PBS setup for class

• Dedicated queue (cse5351)
  – Max 100 hours run time
  – Default run time 1 hour
  – Max number of nodes 8
  – May limit simultaneous executions by user or by queue
PBS Scheduling

• There are no nodes dedicated for parallel processing.

• Scheduling decisions
  – What can run (are enough resources available)
  – What is the user’s priority
    • Recent CPU usage
    • Weighting factor
  – Starvation limit
Scheduling Consequences

• Dedicated node requests can be hard to satisfy
  – Jobs will starve
  – After starvation, jobs will wait for resources to come free

• Wait times can be 2 or 3 days