Notes:Parallel Computing
Topic: Running Jobs on a Parallel Computer

Mary Thomas

Department of Computer Science
Computational Science Research Center (CSRC)
San Diego State University (SDSU)

Posted: 01/24/17
Last Update: 03/02/17

Table of Contents

Overview of an HPC Cluster

A Cluster has multiple, separate nodes, each has multiple cores

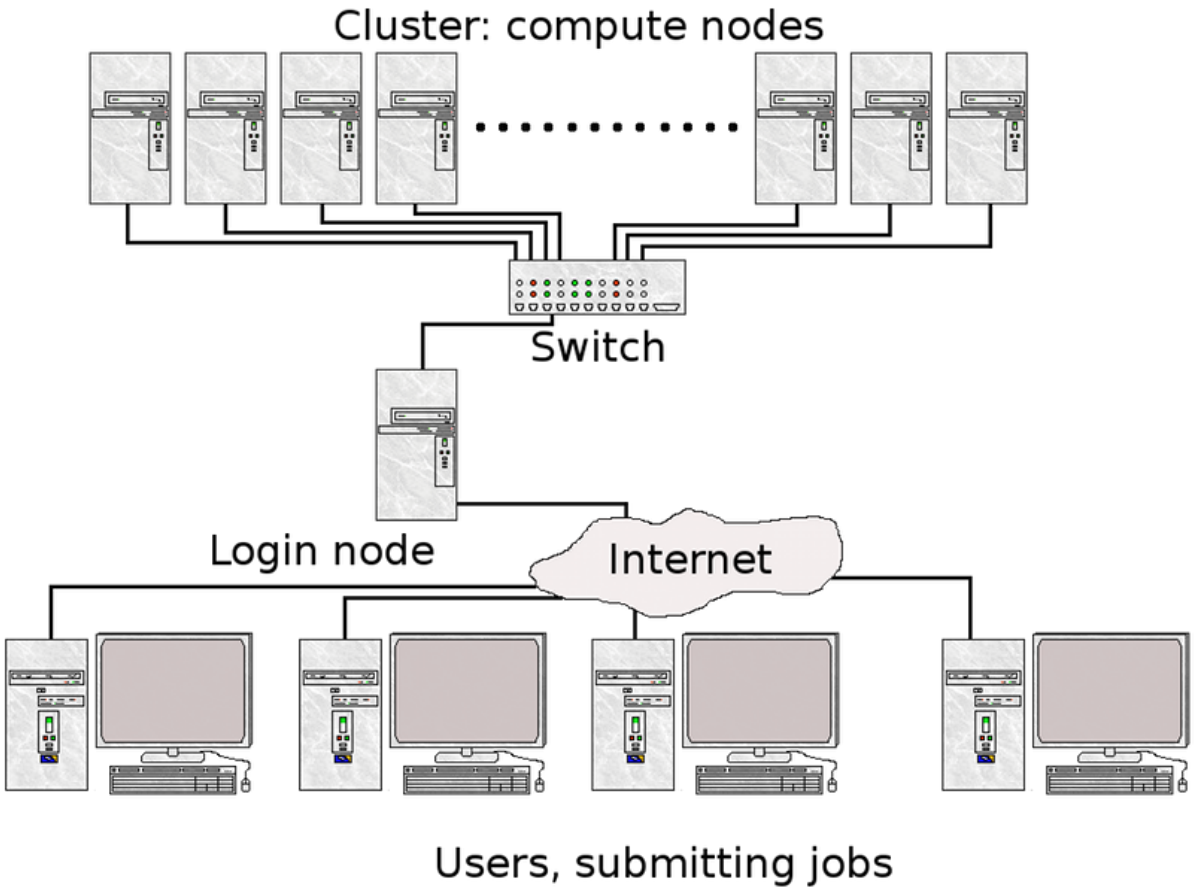
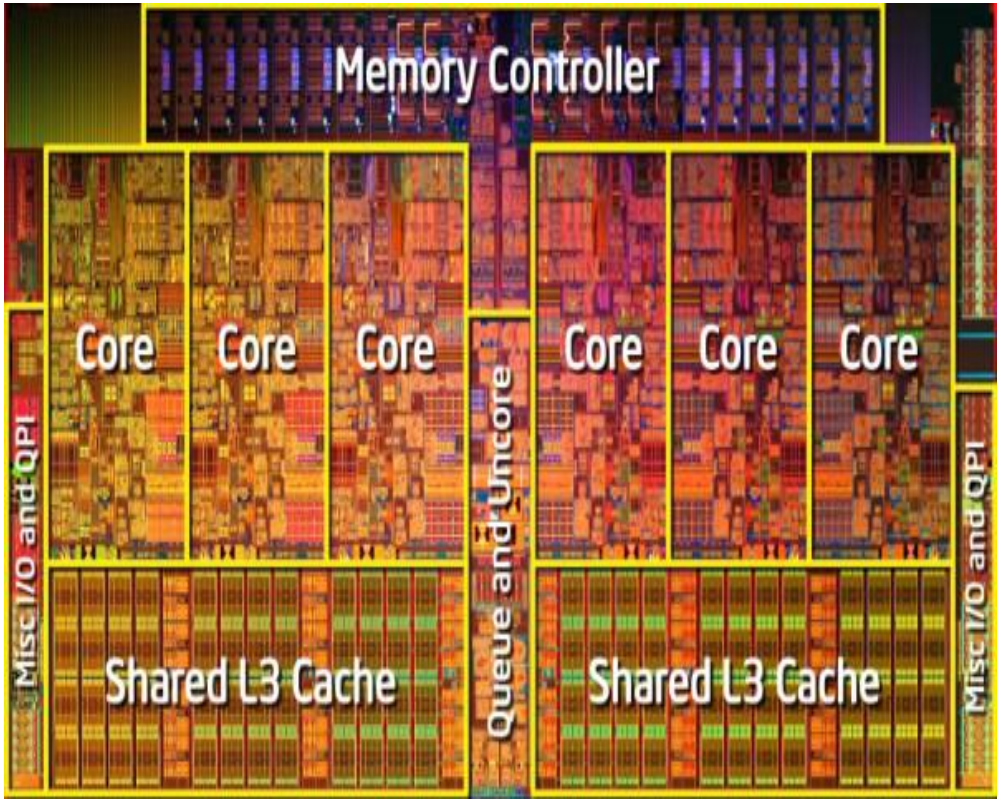


Figure: Diagram of a cluster

Source: https://www.hpc2n.umu.se/support/beginners_guide

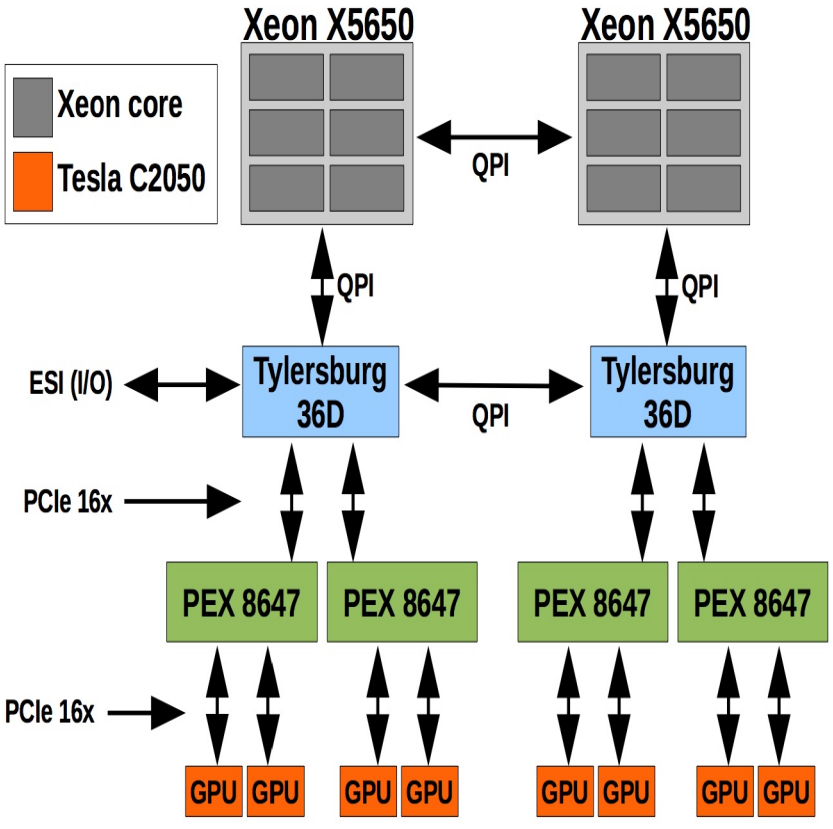
Intel Xeon X5650 system with six cores

- Intel Xeon X5650 system contains six CPUs (Xeon 5650)
- QPI-PCIe bridge.
- PCI-e switch for GPUs. (Peripheral Component Interconnect)



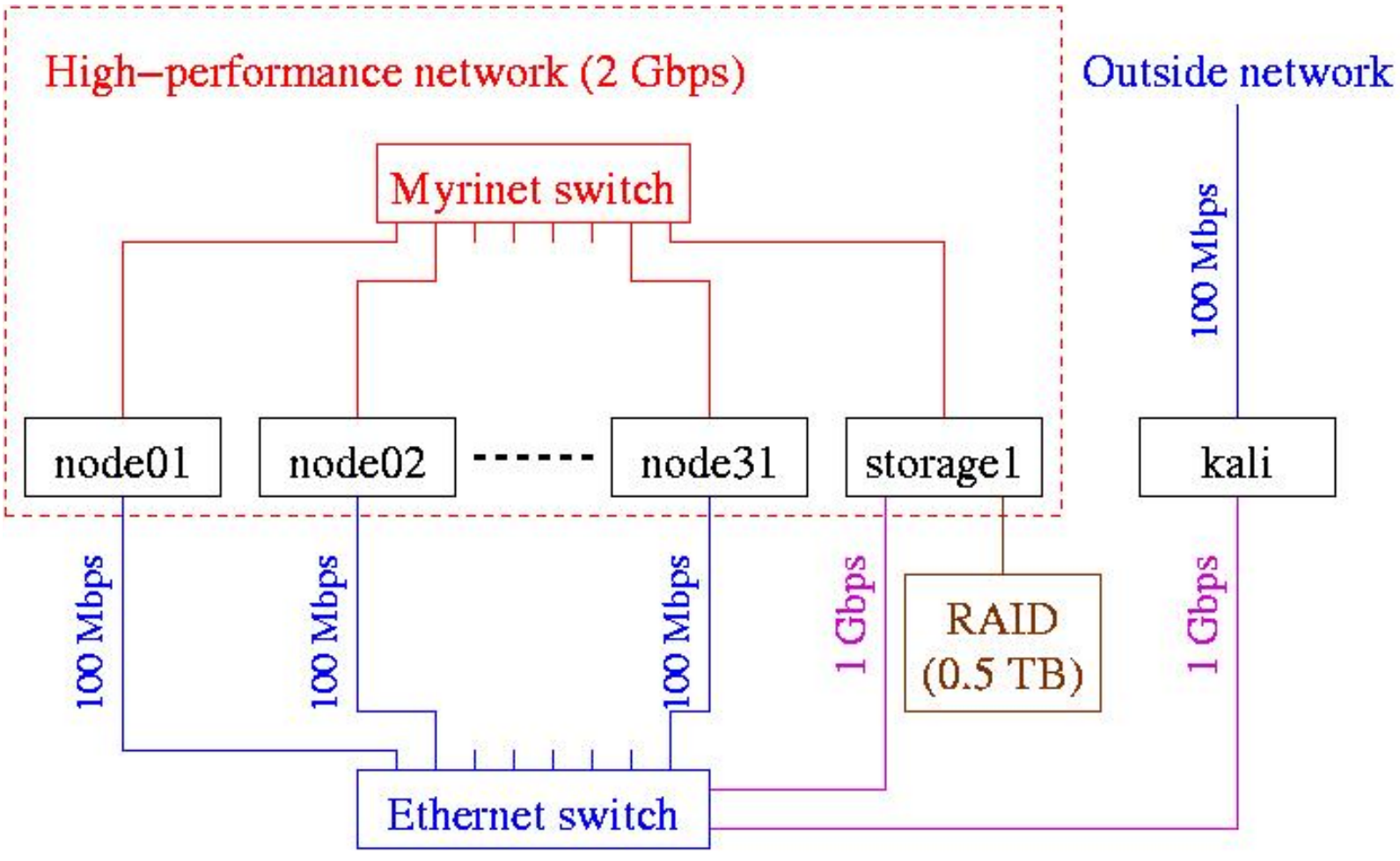
CPU/GPU System: 2 Intel Xeon X5650 and 8 Nvidia GPU Teslas

- Intel Xeon X5650 system
- "Core12": two six-core CPUs (Xeon 5650)
- eight GPUs
- Tylersburg-36D, QPI-PCIe bridge
- PXE8647 PCI-e switch for GPU pairs.



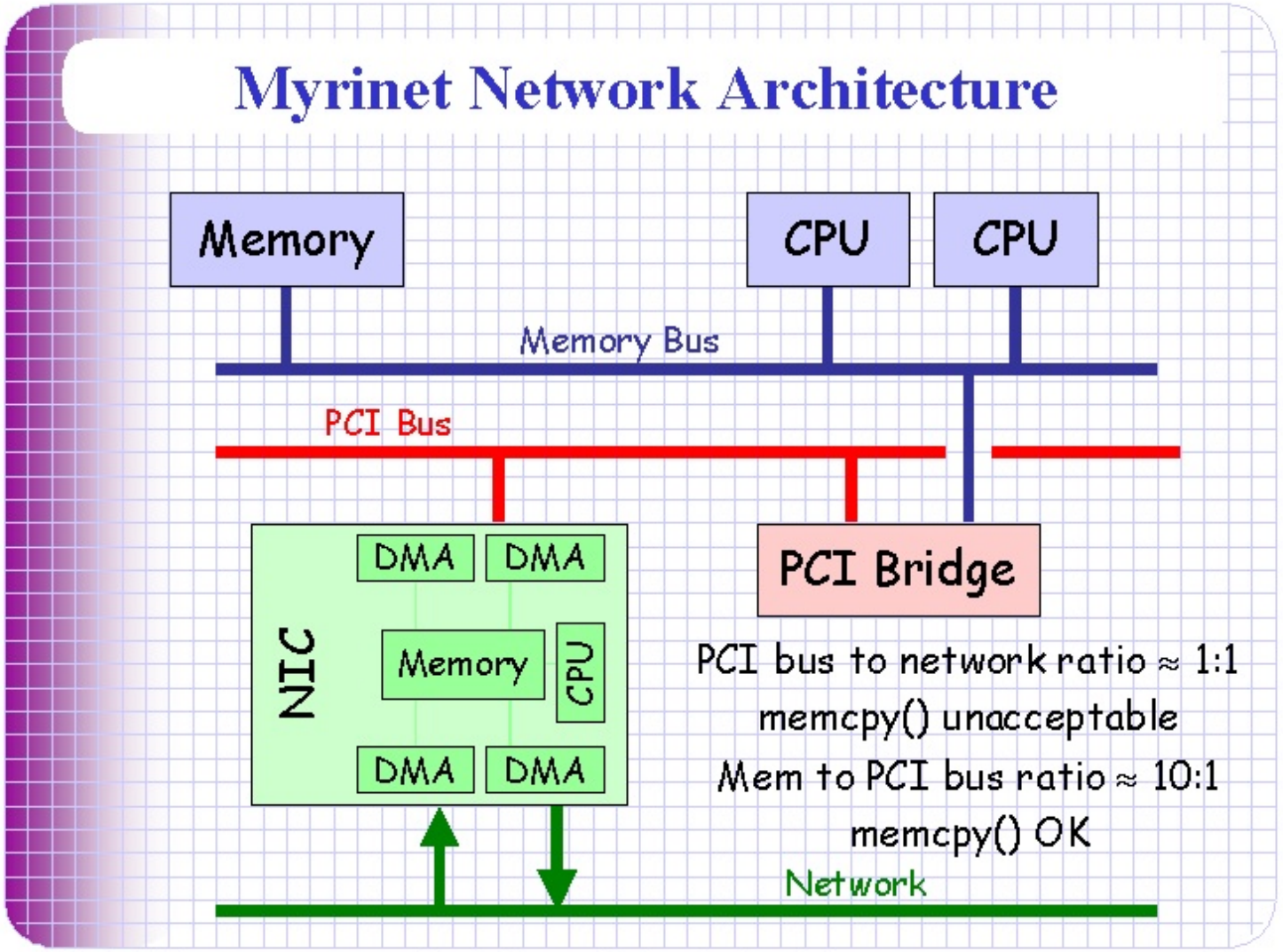
Source: http://hothardware.com/Articles/NVIDIA-GF100-Architecture-and-Feature-Preview

High Performance Network Architecture



Source:
http://www.math.umbc.edu/~gobbert/kali/binaries/networkXY.jpg

Myrinet Networks



Source:

HPC Software Stacks Provide Environment to Run & Operate the Cluster System

Essential Software and Management Tools Needed to Build a Powerful, Flexible, and Highly Available Supercomputer.

HPC Programming Tools

Performance Monitoring	HPCC	Perfctr	IOR	PAPI/IPM	netperf
Development Tools	Cray® Compiler Environment (CCE)	Intel® Cluster Studio	PGI (PGI CDK)	GNU	
Application Libraries	Cray® LibSci, LibSci_ACC	MVAPICH2	OpenMPI	Intel® MPI- (Cluster Studio)	

Middleware Applications and Management

Resource Management / Job Scheduling	SLURM	Grid Engine	MOAB	Altair PBS Pro	IBM Platform LSF	Torque/Maui
File System	NFS	Local FS (ext3, ext4, XFS)	PanFS	Lustre		
Provisioning	Cray® Advanced Cluster Engine (ACE) management software					
Cluster Monitoring	Cray ACE (iSCB and OpenIPMI)					
Remote Power Mgmt	Cray ACE					
Remote Console Mgmt	Cray ACE					

Operating Systems

Operating System	Linux (Red Hat, CentOS, SUSE)

CRAY HPC Software Stack

Source: `http://www.hpcwire.com/2014/02/24/comprehensive-flexible-software-stack-hpc-clusters/`

Distributed Resource Management Systems (D-RMS)

- Primary Function:
  - To control the usage of HPC hard resources: CPU cycles, memory, disk space and network bandwidth.
  - To optimize utilization of resources, maximize system throughput.
  - To orchestrate the process of assigning hard resources to user jobs: Users request resources by submitting jobs (serial or parallel).
- To gain access to one or more nodes in the cluster
- Main components
  - Job Management subsystem (JMS)
  - Physical Resource Manager
  - Scheduler and Queuing Systems

Source: Yan & Chapman, Comparative Study of Distributed Resource Management Systems (2005)
`http://web.cs.uh.edu/~hpctools/Pub/D-RMS-study.pdf`

Queing Systems Control Who is Using the Resources

```
top - 16:38:31 up 5 days,  8:05,  7 users,  load average: 0.28, 0.31, 0.20
Tasks: 176 total,   2 running, 174 sleeping,   0 stopped,   0 zombie
Cpu(s): 24.2%us,  0.8%sy,  0.0%ni, 73.5%id,  1.4%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  12188132k total,  4513528k used,  7674604k free,    29736k buffers
Swap: 33409020k total,    21692k used, 33387328k free,  1665928k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
16744 [redacted]  20   0 4664m 2.4g  896 R 99.7 20.9   0:11.82 histogram_mod
15234 [redacted]  20   0  105m 1812 1440 S  0.3  0.0   0:00.15 bash
16721 mthomas   20   0 15040 1292  940 R  0.3  0.0   0:00.05 top
    1 root      20   0 19364  696  480 S  0.0  0.0   0:02.61 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.01 kthreadd
```

D-RMS: Some Commonly Used Systems

- Single parallel systems:
  - Portable Batch Systems (PBS)/TORQUE:
    `http://www.adaptivecomputing.com`
  - Simple Linux Utility for Resource Management (SLURM,
    `https://computing.llnl.gov/linux/slurm/`
  - IBM Load Lever
  - Platform Load Sharing Facility (LSF)
- Multiple, distributed, parallel systems:
  - Sun Grid Engine (SGE):
    `http://star.mit.edu/cluster/docs/0.93.3/guides/sge.html`
  - HTCondor: High Throughput computing
    `http://research.cs.wisc.edu/htcondor/`

Source: Yan & Chapman, Comparative Study of Distributed Resource Management Systems (2005)
`http://web.cs.uh.edu/~hpctools/Pub/D-RMS-study.pdf`

D-RMS: Job Management subsystem (JMS)

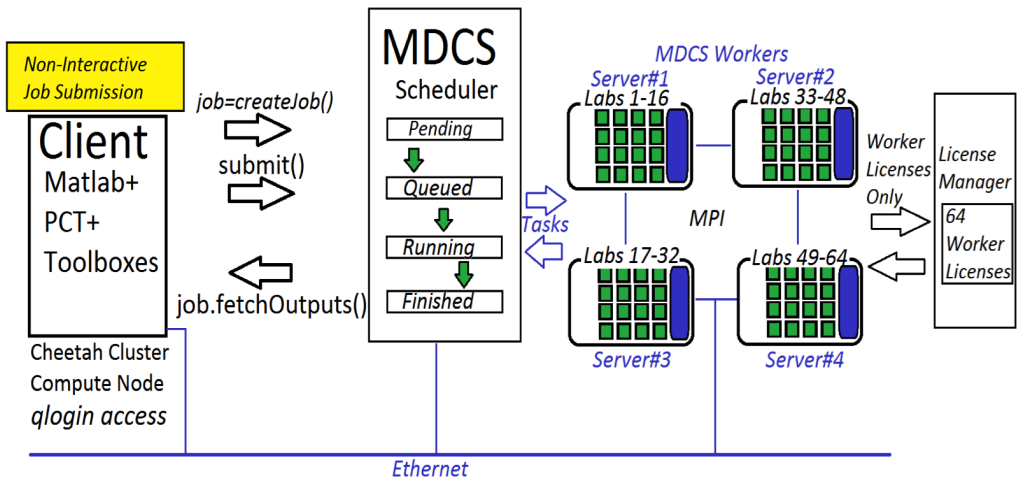
- Interface between users and RMS
- Different types of jobs
  - simple scripts
  - command line/interactive
  - Apps using MPI, OpenMP, CUDA, parallel libraries
  - job arrays/multi-task
  - workflow/complex dependent jobs
- Manages jobs:
  - Submission: name, type, I/O, parallel environment
  - Resource requirements: #cores, wall-clock time, memory, disk space, network
  - Control: queuing/scheduling, deleting, status/checking, suspension/resume, checkpointing
  - Monitoring, History
  - Accounting

D-RMS: Physical Resource Manager

- Static Resource Info:
  - Used to control the use of hardware (CPU cycles, memory, swap, disk, network)
  - Applies resource usage constrains and/or local usage policies.
  - Architecture: #nodes, #cores, OS
  - Memory: amount and architecture (shared, distributed)
  - Network: topology, bandwidth, latency
  - Software: libraries, utilities
- Dynamic Resource Info:
  - Resource Load information and thresholding.
  - Memory: percent used.
  - Network: available bandwidth.
- Accounting/Usage:
  - Account status and usage of resources.
  - Track job ID, user, history

D-RMS: Scheduler and Queue Systems

- Utilizes information about local resources:
  - high-performance
  - networking, network topology
  - multi-core node configuration
  - cache and memory hierarchy
  - parallel file systems
  - parallel libraries
- Applies scheduling algorithms to organize and optimized jobs
- Enforces policies for usage and charging, etc.



Parallel Matlab Distributed Computing Server.
Note: MOM == machine oriented mini-server.

Portable Batch Scheduler (PBS/TORQUE)

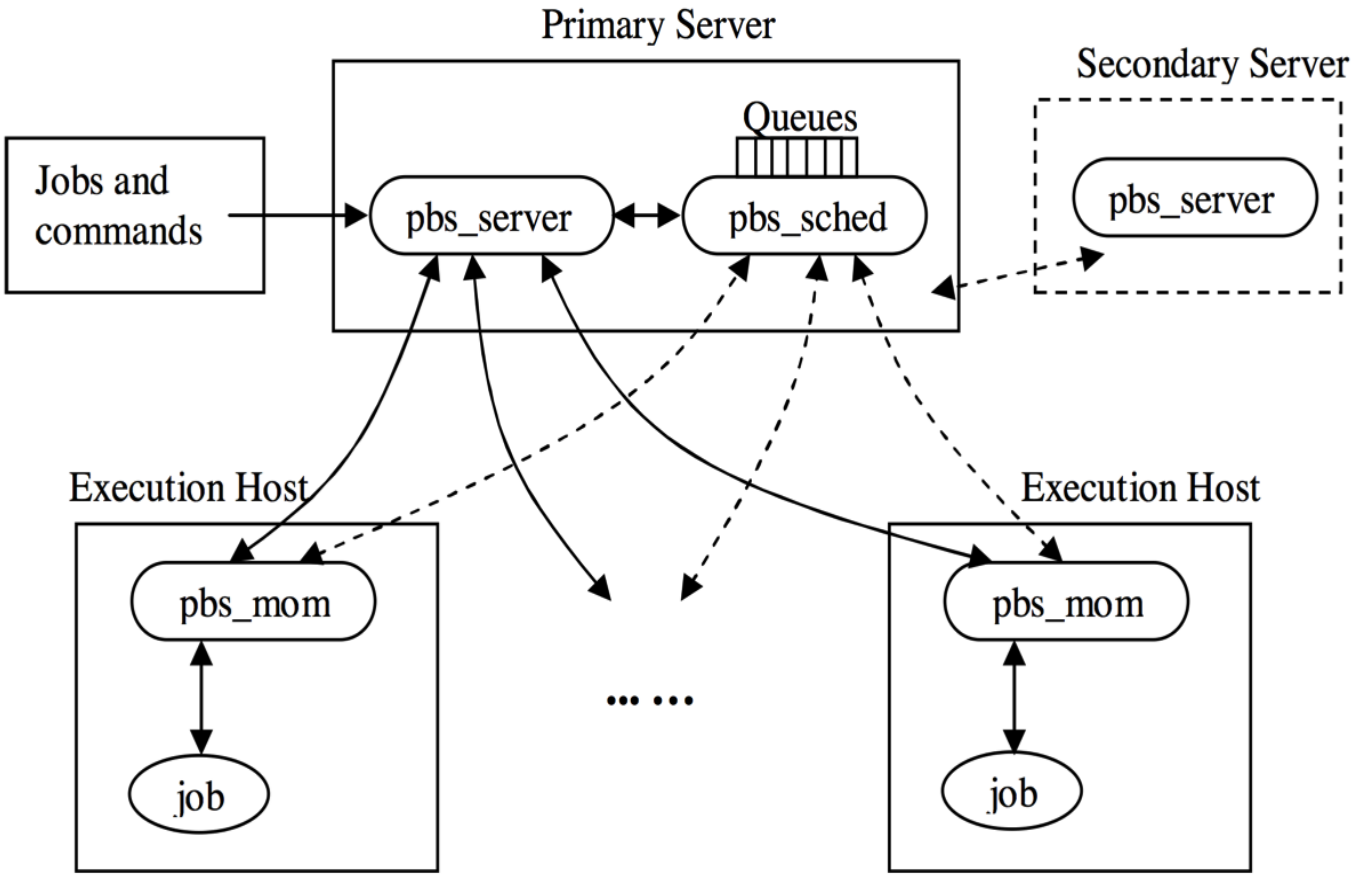


Figure: TORQUE/PBS Pro Installation Architecture (tuckoo.sdsu.edu)

PBS Batch Queuing System.

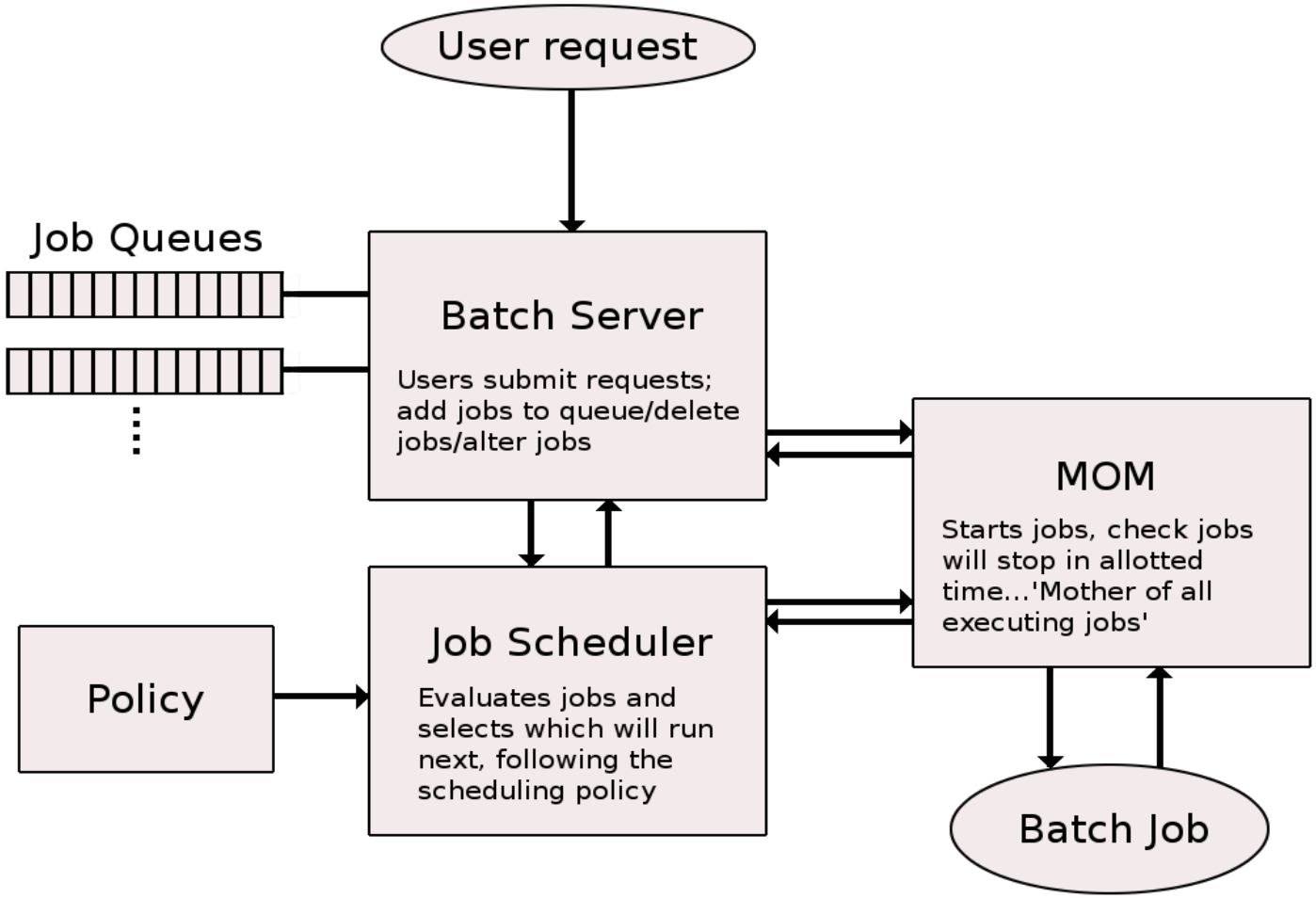
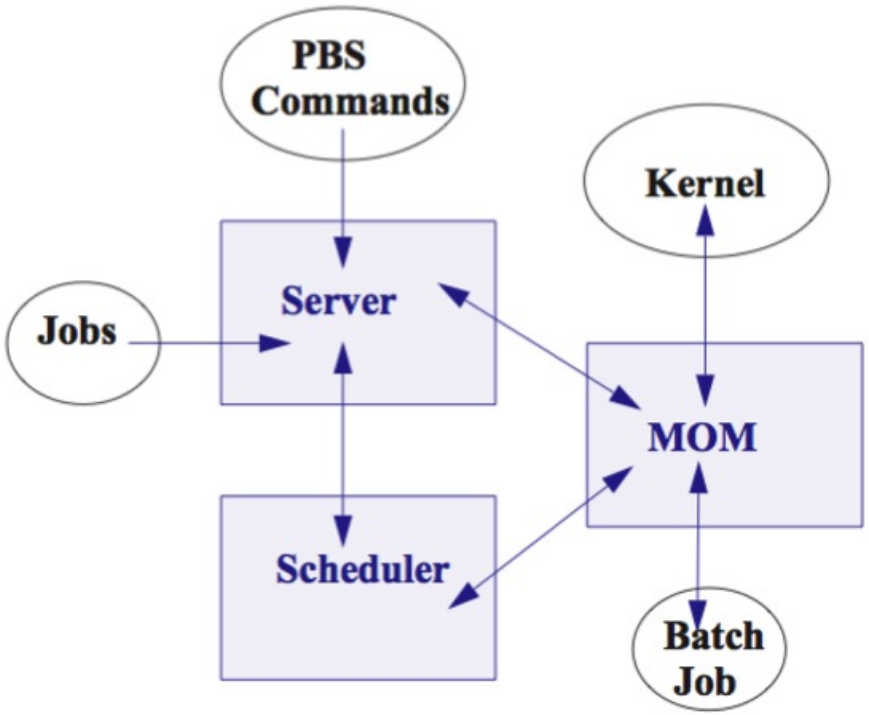


Figure: TORQUE/PBS batch queuing system

Source: `https://www.hpc2n.umu.se/support/beginners_guide`

Portable Batch Scheduler (PBS) Components

- PBS Job Server
  - commands/daemons communicate with Server
  - batch job services: receiving/creating, running, modifying, protecting against system crashes
- PBS Job Scheduler
  - control when/where jobs run
  - communicate with machine oriented mini-server (MOM)
- PBS MOM Processes
  - Machine Oriented Miniserver
  - Starts job, makes sure it completes in specified time
  - user login session
- PBS Client Programs
  - command line or GUI, user, operator, manager
  - submit, monitor, modify, delete



PBS/TORQUE Job life cycle

Stage	Description

Creation	Typically, a submit script is written to hold all of the parameters of a job. These parameters could include how long a job should run (**walltime**), what resources are necessary to run, and what to execute. The following is an example submit file:
	`#PBS -N localBlast`
	`#PBS -S /bin/sh`
	`#PBS -l nodes=1:ppn=2,walltime=240:00:00`
	`#PBS -M user@my.organization.com`
	`#PBS -m ea`
	`source ~/.bashrc`
	`cd $HOME/work/dir`
	`sh myBlast.sh -i -v`
	This submit script specifies the name of the job (localBlast), what environment to use (`/bin/sh`), that it needs both processors on a single node (**nodes=1:ppn=2**), that it will run for at most 10 days, and that TORQUE should email "user@my.organization.com" when the job exits or aborts. Additionally, the user specifies where and what to execute.

Submission	A job is submitted with the qsub command. Once submitted, the policies set by the administration and technical staff of the site dictate the priority of the job and therefore, when it will start executing.
Execution	Jobs often spend most of their lifecycle executing. While a job is running, its status can be queried with qstat.
Finalilzation	When a job completes, by default, the `stdout` and `stderr` files are copied to the directory where the job was submitted.

PBS: Common Commands

- Job control
  - qsub: submit a job
  - qdel: delete a batch job
  - qsig: send a signal to a batch job
  - qhold: hold a batch job
  - qrerun: rerun a batch job
  - qmove: move a batch job to another queue
- Job monitoring
  - qstat show the status of batch jobs
  - qselect select a specific subset of jobs
- Node status
  - pbsnodes list the status and attributes of all nodes in the cluster.
  - `http://linuxinfo.physik.hu-berlin.de/pbs.html`
- Others
  - qalter alter a batch job
  - qmsg send a message to a batch job

Example of User Guide: `http://rcc.its.psu.edu/user_guides/system_utilities/pbs/`

PBS: Batch Script Example

```
#!/bin/sh
#########
### request number of cores and the nodes on which to run the job
#PBS -l nodes=2:ppn=8:core4

########
### Define the job name
#PBS -N myParallelJob

#########
### Define where the output and error messages should go
#PBS -j oe

#########
### Define the name of the queue
#PBS -q batch

#########
### change to the directory from where this script was submitted
cd $PBS\_O\_WORKDIR
echo ------------------------------------------------------
echo PBS: job identifier is $PBS_JOBID
echo PBS: job name is $PBS_JOBNAME
echo PBS: current home directory is $PBS_O_HOME
echo ------------------------------------------------------
NCORES=‘wc -w < $PBS_NODEFILE‘
echo "many-test using $NCORES cores..."
mpirun -np 10 -hostfile $PBS_NODEFILE  ./myParallelJob
```

PBS Batch Script: Environment Variables - Submission Machine

Variable Name	Description
PBS_O_HOST	The host machine on which the qsub command was run.
PBS_O_LOGNAME	The login name on the machine on which the qsub was run.
PBS_O_HOME	The home directory from which the qsub was run.
PBS_O_WORKDIR	The working directory from which the qsub was run.

(1) PBS User Guide Example:
https://wiki.hpcc.msu.edu/display/hpccdocs/Advanced+Scripting+Using+PBS+Environment+Variables (2)
http://www.ep.ph.bham.ac.uk/general/support/torquepbsdsh.html

PBS Batch Script: Environment Variables - Execution Machine

Variable Name	Description
PBS_O_QUEUE	Queue where job was submitted to.
PBS_QUEUE	Queue job is running in (same as PBS_O_QUEUE)
PBS_JOBID	Job ID – used by qstat, showstart, and dque.
PBS_JOBNAME	Name of the job. This can be set using -N in script
PBS_NODEFILE	Name of file that contains list of HOSTS for job
PBS_VNODENUM	Determine the task number of each processor.

(1) PBS User Guide Example:
`https://wiki.hpcc.msu.edu/display/hpccdocs/Advanced+Scripting+Using+PBS+Environment+Variables` (2)
`http://www.ep.ph.bham.ac.uk/general/support/torquepbsdsh.html`

Running Parallel Jobs: typically use *mpirun*

- A script/wrapper controls several aspects of program execution in Open MPI.
- Uses the Open Run-Time Environment (ORTE) to launch jobs.
- Used to hide differences in starting jobs on different machines.
- You need to list the hosts on which the jobs will run (created by the PBS resource manager)

- You can run serial or parallel *executables*; language does not matter (C, F90, Python, etc).
- From the command line, our cluster will launch multiple Pthreads (OS/compiler dependent).
- From the batch queue, it will use the PBS resource manager
- See online manpage: `http://www.linuxcommand.org/man_pages/mpirun1.html`

Specifying Attributes: #PBS -l nodes=2:ppn=3

Create a batch script and set batch queue attributes. There are a large number of attributes for selecting nodes and cores.

#PBS -l nodes=2:ppn=3

- Requesting 2 nodefiles
- Asking for 3 processors per node (ppn)
- Asking for 6 total processors
- $NPROCS = nodes * ppn$
  $= 2 * 3 = 6$

```
[mthomas@tuckoo hello]$ pbsnodes node1
node1
     state = free
     np = 4
     properties = core4,type1
     ntype = cluster
     status = rectime=1423772995,...
[mthomas@tuckoo hello]$
[mthomas@tuckoo hello]$ cat batch.hello
#!/bin/sh
#PBS -V
#PBS -l nodes=2:ppn=3
#PBS -N hello
#PBS -j oe
#PBS -q batch
cd $PBS_O_WORKDIR

echo "PBS: job name is $PBS_JOBNAME "
mpirun -np 6 -hostfile $PBS_NODEFILE ./hello
```

Tutorial:
http://ktchu.serendipityresearch.org/research/CSE/PBS.html

Setting Attributes: Request a specific number of cores #PBS -l nodes=1:ppn=6

In this case, the system picks one node, which has 8 cores, because we requested 6 ppn.

```
[mthomas@tuckoo hello]$ cat batch.hello
#!/bin/sh
# "man pbs_resources"
#PBS -V
#PBS -l nodes=1:ppn=6
#PBS -N hello
#PBS -j oe
#PBS -q batch
cd $PBS_O_WORKDIR
mpirun -np 6 -hostfile $PBS_NODEFILE ./hello

[mthomas@tuckoo hello]$ qsub batch.hello
197.tuckoo.sdsu.edu
[mthomas@tuckoo hello]$ qstat -a
tuckoo.sdsu.edu:
                                                                  Req'd  Req'd   Elap
Job ID               Username Queue    Jobname          SessID NDS   TSK Memory Time  S Time
-------------------- -------- -------- ---------------- ------ ----- --- ------ ----- - -----
197.tuckoo.sdsu.     mthomas  batch    hello              6487     1   6    --    --  C 00:00
[mthomas@tuckoo hello]$ cat hello.o197
hello-test using 6 cores...
hello, world from node8
hello, world from node8
hello, world from node8
hello, world from node8
hello, world from node8
hello, world from node8
```

Setting Attributes: Request a Specific Node Type #PBS -l nodes=1:ppn=6:core6

In this case, the system picks one node, which has 8 cores, because we requested 6 ppn.

```
[mthomas@tuckoo hello]$ cat batch.hello
#!/bin/sh
# "man pbs_resources"
#PBS -V
#PBS -l nodes=1:ppn=6:core6
#PBS -N hello
#PBS -j oe
#PBS -q batch
cd $PBS_O_WORKDIR
mpirun -np 6 -hostfile $PBS_NODEFILE ./hello

[mthomas@tuckoo hello]$ qsub batch.hello
2019.tuckoo.sdsu.edu
[mthomas@tuckoo hello]$ qstat -a
tuckoo.sdsu.edu:
                                                                  Req'd  Req'd   Elap
Job ID               Username Queue    Jobname          SessID NDS   TSK Memory Time  S Time
-------------------- -------- -------- ---------------- ------ ----- --- ------ ----- - -----
2019.tuckoo                 ...h.hello-cpuid mthomas         00:00:00 C batch
[mthomas@tuckoo hello]$ cat batch.hello-cpuid.o2019
PBS: job name is batch.hello-cpuid
hello, world from node: node5, core: 2
hello, world from node: node5, core: 6
hello, world from node: node5, core: 4
hello, world from node: node5, core: 1
hello, world from node: node5, core: 0
hello, world from node: node5, core: 6
```

Specifying Attributes: Request Node by Name (v1)
#PBS -l nodes=1:ppn=8:node8

place node name after 'ppn' attribute

```
[mthomas@tuckoo hello]$ cat batch.hello
#!/bin/sh
# this example batch script requests hello processors...
# "man pbs_resources"
#PBS -V
#PBS -l nodes=1:ppn=8:node8
#PBS -N hello
#PBS -j oe
#PBS -q batch
cd $PBS_O_WORKDIR
NCORES=‘wc -w < $PBS_NODEFILE‘
echo "hello-test using $NCORES cores..."
mpirun -np 8 -hostfile $PBS_NODEFILE ./hello

[mthomas@tuckoo hello]$ qsub batch.hello
197.tuckoo.sdsu.edu
[mthomas@tuckoo hello]$ qstat -a
tuckoo.sdsu.edu:
                                                                  Req'd  Req'd   Elap
Job ID               Username Queue    Jobname          SessID NDS   TSK Memory Time  S Time
-------------------- -------- -------- ---------------- ------ ----- --- ------ ----- - -----
197.tuckoo.sdsu.     mthomas  batch    hello              6487     1   8    --    --  C 00:00
[mthomas@tuckoo hello]$ cat hello.o197
hello-test using 4 cores...
hello, world from node8 hello, world from node8
hello, world from node8 hello, world from node8
hello, world from node8 hello, world from node8
hello, world from node8 hello, world from node8
```

Specifying Attributes: Request Node by Name (v2) #PBS -l nodes=node8:ppn=8

place node name before 'ppn' attribute

```
[mthomas@tuckoo hello]$ cat batch.hello
#!/bin/sh
# this example batch script requests hello processors...
# "man pbs_resources"
#PBS -V
#PBS -l nodes=node8:ppn=8
#PBS -N hello
#PBS -j oe
#PBS -q batch
cd $PBS_O_WORKDIR
NCORES=`wc -w < $PBS_NODEFILE`
echo "hello-test using $NCORES cores..."
mpirun -np 8 -hostfile $PBS_NODEFILE ./hello

[mthomas@tuckoo hello]$ qsub batch.hello
197.tuckoo.sdsu.edu
[mthomas@tuckoo hello]$ qstat -a
tuckoo.sdsu.edu:
                                                                  Req'd  Req'd   Elap
Job ID               Username Queue    Jobname          SessID NDS   TSK Memory Time  S Time
-------------------- -------- -------- ---------------- ------ ----- --- ------ ----- - -----
197.tuckoo.sdsu.     mthomas  batch    hello              6487     1   8    --    --  C 00:00
[mthomas@tuckoo hello]$ cat hello.o197
hello-test using 4 cores...
hello, world from node8 hello, world from node8
hello, world from node8 hello, world from node8
hello, world from node8 hello, world from node8
hello, world from node8 hello, world from node8
```

Setting Attributes: Using Two Random Nodes #PBS -l nodes=2:ppn=3

In this case, the system will choose the nodes

```
[mthomas@tuckoo hello]$ qsub batch.hello-cpuid
221.tuckoo.sdsu.edu
[mthomas@tuckoo hello]$ qstat -a

tuckoo.sdsu.edu:
                                                                  Req'd  Req'd   Elap
Job ID               Username Queue    Jobname          SessID NDS   TSK Memory Time  S Time
-------------------- -------- -------- ---------------- ------ ----- --- ------ ----- - -----
2011.tuckoo                 ...h.hello-cpuid mthomas         00:00:00 C batch

[mthomas@tuckoo hello]$ cat batch.hello-cpuid.o2011
PBS: job name is batch.hello-cpuid
hello, world from node: node2, core: 1
hello, world from node: node2, core: 3
hello, world from node: node2, core: 0
hello, world from node: node1, core: 3
hello, world from node: node1, core: 2
hello, world from node: node1, core: 2
```

Setting Attributes: Select Two Nodes by Name #PBS -l nodes="node1:ppn=3+node4:ppn=3"

In this case, the system will only used the requested nodes.
The plus (+) sign combines the requested resources.
Note: must use "xxxx" for combination to work

```
[mthomas@tuckoo hello]$ qsub batch.hello-cpuid
221.tuckoo.sdsu.edu
[mthomas@tuckoo hello]$ qstat -a

tuckoo.sdsu.edu:
                                                                  Req'd  Req'd   Elap
Job ID               Username Queue    Jobname          SessID NDS   TSK Memory Time  S Time
-------------------- -------- -------- ---------------- ------ ----- --- ------ ----- - -----
2011.tuckoo                 ...h.hello-cpuid mthomas         00:00:00 C batch

[mthomas@tuckoo hello]$ cat batch.hello-cpuid.o2011
PBS: job name is batch.hello-cpuid
hello, world from node: node1, core: 0
hello, world from node: node1, core: 3
hello, world from node: node1, core: 1
hello, world from node: node4, core: 3
hello, world from node: node4, core: 2
hello, world from node: node4, core: 2
```

Other Ways to Specify Combinations of Nodes

```
### one node, 8 processors per node.
#PBS -l nodes=1:ppn=8

#PBS -l nodes="node1:ppn=4+node9:ppn=4+node11:ppn=16"

#PBS -l nodes="1:ppn=4:core4+1:ppn=4:core6+1:ppn=16:core16"

#PBS -l nodes="2:ppn=4:core4+2:ppn=6:core6"

#PBS -l nodes="2:ppn=4:core4"
```

Oversubscribing Nodes

- Scheduling more processes to run than there are available slots
- Oversubscribing can be useful for applications where multiple threads do not contend for CPU power.
- Oversubscribing can result in performance degradation.
- *mpif90* on tuckoo is *openmpi*.
- Open MPI schedules processes to nodes by asking two questions from each application on the *mpirun* command line:
  - How many processes should be launched? *mpirun -np X*
  - Where should those processes be launched? depends on three factors:
    - The final node list
    - The scheduling policy
    - The default and maximum number of slots on each host/node (slots are #processors on host)
    - "max_slots" value set to be the same as the number of "slots" value for each node
- use **–nooversubscribe** option

Source: `http://rpm.pbone.net/index.php3/stat/45/idpl/7942490/numer/1/nazwa/mpirun`

Setting Attributes: asking for the right number of cores #PBS -l nodes=1:ppn=8:node8

In this case, the job completes because we requested enough cores, even though we did not use them all

```
[mthomas@tuckoo hello]$ cat batch.hello
#!/bin/sh
# this example batch script requests hello processors...
# "man pbs_resources"
#PBS -V
#PBS -l nodes=1:ppn=8:node8
#PBS -N hello
#PBS -j oe
#PBS -q batch
cd $PBS_O_WORKDIR
NCORES=‘wc -w < $PBS_NODEFILE‘
echo "hello-test using $NCORES cores..."
mpirun -np 7 -hostfile $PBS_NODEFILE ./hello

[mthomas@tuckoo hello]$ qsub batch.hello
213.tuckoo.sdsu.edu
[mthomas@tuckoo hello]$ qstat -a
tuckoo.sdsu.edu:

                                                                  Req’d  Req’d   Elap
Job ID               Username Queue    Jobname          SessID NDS   TSK Memory Time  S Time
-------------------- -------- -------- ---------------- ------ ----- --- ------ ----- - -----
213.tuckoo.sdsu.     mthomas  batch    hello              6487     1   8    --    --  C 00:00
[mthomas@tuckoo hello]$ cat hello.o216
PBS: job name is hello
hello-test using 8 cores.
hello, world from node8 hello, world from node8
hello, world from node8 hello, world from node8
hello, world from node8 hello, world from node8
hello, world from node8 hello, world from node8
```

Setting Attributes: Oversubscribing #PBS -l nodes=1:ppn=4:core4

In this example below, we ask for 8 cores, which is more than were requested and are available on the core4 node. The default setting for the OpenMPI library is to allow cores to be oversubscribed:multiple threads will be launched on the core.

```
[mthomas@tuckoo hello]$ cat batch.hello
[mthomas@tuckoo hello]$ cat batch.hello
#!/bin/sh
#PBS -V
#PBS -l nodes=1:ppn=4:core4
#PBS -N hello
#PBS -j oe
#PBS -q batch
cd $PBS_O_WORKDIR
mpirun -np 8 -hostfile $PBS_NODEFILE  ./hello
[mthomas@tuckoo hello]$ qsub batch.hello
2586.tuckoo.sdsu.edu
[mthomas@tuckoo hello]$ qstat -a
tuckoo.sdsu.edu:
                                                                  Req'd  Req'd   Elap
Job ID               Username Queue    Jobname          SessID NDS   TSK Memory Time  S Time
-------------------- -------- -------- ---------------- ------ ----- --- ------ ----- - -----
2586.tuckoo.sdsu     mthomas  batch    hello              16079     1   4    --    --  C 00:00
[mthomas@tuckoo hello]$ cat hello.o2586
Running: hello, using 4 cores...
hello, world from node1     hello, world from node1
hello, world from node1     hello, world from node1
hello, world from node1     hello, world from node1
hello, world from node1     hello, world from node1
```

Setting Attributes: Using *-nooversubscribe*

#PBS -l nodes=1:ppn=4:core4

In this case, the job fails because we used the mpirun argument –nooversubscribe .

```
[mthomas@tuckoo hello]$ cat batch.hello
#!/bin/sh
#PBS -V
#PBS -l nodes=1:ppn=4:core4
#PBS -N hello
#PBS -j oe
#PBS -q batch
cd $PBS_O_WORKDIR
NCORES=‘wc -w < $PBS_NODEFILE‘
echo "hello-test using $NCORES cores..."
mpirun -np 7 -hostfile $PBS_NODEFILE -nooversubscribe ./hello

[mthomas@tuckoo hello]$ qsub batch.hello
qsub batch.hello
2605.tuckoo.sdsu.edu
[mthomas@tuckoo hello]$ qstat -a
tuckoo.sdsu.edu:
                                                                  Req’d  Req’d   Elap
Job ID               Username Queue    Jobname          SessID NDS   TSK Memory Time  S Time
-------------------- -------- -------- ---------------- ------ ----- --- ------ ----- - -----
2605.tuckoo.sdsu     mthomas  batch    hello              6254     1   4    --     --  C 00:00
[mthomas@tuckoo hello]$
[mthomas@tuckoo hello]$ cat hello.o2605
Running: hello, using 4 cores...
--------------------------------------------------------------------------
There are not enough slots available in the system to satisfy the 8 slots
that were requested by the application:
  ./hello

Either request fewer slots for your application, or make more slots available
for use.
```

Cluster Configuration Information Sources & Commands

Information File	**Description**
cat /etc/motd	Message of the day, typically printed out on login.
cat /proc/cpuinfo	Detailed information about the CPU & GPU's.

Commands	**Description**
hostname	Prints out local hostname and IP address
uname -a	Prints out OS information
ifconfig	getting network configuration
pbsnodes	PBS/TORQUE Command
ifconfig	displays the status of the currently active interfaces (ethernet, infiniba
ip	show / manipulate routing, devices, policy routing and tunnels

/etc/motd file

Typically printed out when you log in

```
[mthomas@tuckoo]$ cat /etc/motd

  the cluster system has 10 compute nodes with various CPUs:
    Node name               #Avail Cores   Node Properties**    Got GPUs?
-------------------------------------------------------------------------------
    node1,node2,node3,node4  4ea.          core4, type1            no
    node8                    8              --    type1            yes
-------------------------------------------------------------------------------
    node5                    8             core6, type2            no
    node6                    6             core6, type2            no
    node9                    6             core6, type2            yes
    node10                   16            core16,type2            no
    node11                   16            core16,type2            yes
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
     **see the output from "pbsnodes -a".

   CPUs & RAM
   ----------------------------------------------
   node1 thru node4, Xeon X3360   @ 2.83GHz, 8GB ea.
   node5             Xeon E5420   @ 2.50GHz, 20GB
   node6             Xeon E5-1650 @ 3.20GHz, 64GB
   node8             Xeon E5620   @ 2.40GHz, 48GB
   node9             Xeon E5-1660 @ 3.30GHz, 32GB
   node10            Xeon E5-2650 @ 2.60GHz, 64GB
   node11            Xeon E5-2650 @ 2.60GHz, 64GB

   GPUs
   ----------------------------------------------
   node9  has 2     GTX 480 gpu cards (1.6GB dev ram ea.)
   node8  has 2     C2075   gpu cards (  6GB dev ram ea.)
   node11 has 1     K40     gpu card  (             )
```

/proc/cpuinfo

```
[mthomas@tuckoo ~]$ cat /proc/cpuinfo
processor : 0
vendor_id : GenuineIntel
cpu family : 6
model : 26
model name : Intel(R) Xeon(R) CPU           E5504  @ 2.00GHz
stepping : 5
cpu MHz : 1600.000
cache size : 4096 KB
physical id : 1
siblings : 4
core id : 0
cpu cores : 4
apicid : 16
initial apicid : 16
fpu : yes
fpu_exception : yes
cpuid level : 11
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe sys
bogomips : 3999.68
clflush size : 64
cache_alignment : 64
address sizes : 40 bits physical, 48 bits virtual
power management:

processor : 1
vendor_id : GenuineIntel
cpu family : 6
model : 26
...
```

Table: Table of tuckoo CPU/GPU configurations (Spring, 2014)

Property	**csrc-gpu**	**csrc-gpu2**	**csrc-gpu3**
node ID	10	11	12
CPU Type	2 Xeon X5650	2 Xeon X5650	2 Xeon E5620
#CPU cores	6*2=12	4*2=8	3*2=6
GPU Type	**2 Tesla C1060**	**2 Tesla C2075**	**2 Tesla C2075**
Multiprocessor (MP)	30	14	14
#SP/Cores	240	448	448
#GigaFLOPS		515	
Max Thd/Block	512	1024	1024
Max Grd Dim	$64k \times 64k \times 1$	$64k \times 64k \times 64$	$64k \times 64k \times 64$

Obtaining information on all nodes: *pbsnodes -a*

```
[mthomas@tuckoo hello]$ pbsnodes -a
node1
     state = free
     np = 4
     properties = core4,type1
     ntype = cluster
     status = rectime=1423769834,varattr=,jobs=,state=free,netload=11658237823,gres=,loadave=0.99,ncpus=4,
                physmem=7929488kb,availmem=33728156kb,totmem=34011004kb,idletime=148949,nusers=1,
                nsessions=1,sessions=1811,uname=Linux node1 2.6.32-220.17.1.el6.x86_64 #1
                SMP Wed May 16 00:01:37 BST 2012 x86_64,opsys=linux   gpus = 0

node2
     state = free
     np = 4
     properties = core4,type1
     ntype = cluster
     status = rectime=1423769848,varattr=,jobs=,state=free,...
     gpus = 0

node3
     state = free
     np = 4
     properties = core4,type1
     ntype = cluster
     status = rectime=1423769846,varattr=,jobs=,state=free,...
     gpus = 0

node4
     state = free
     np = 4
     properties = core4,type1
     ntype = cluster
     status = rectime=1423769849,varattr=,jobs=,state=free,...
     gpus = 0
...
```

Obtaining information on one node: *pbsnodes < nodeName >*

```
[mthomas@tuckoo hello]$ pbsnodes node11
node11
     state = free
     np = 16
     properties = core16,type2
     ntype = cluster
     status = rectime=1423800490,varattr=,jobs=,state=free,netload=2257827798,gres=,loadave=0.00,ncpus=16,
     physmem=65941452kb,availmem=131961592kb,totmem=133050308kb,idletime=106538,
     nusers=1,nsessions=6,sessions=11134 11138 11139 11147 11149 11171,uname=Linux node11
     2.6.32-431.el6.x86_64 #1 SMP Fri Nov 22 03:15:09 UTC 2013 x86_64,opsys=linux
     gpus = 0

[mthomas@tuckoo hello]$
[mthomas@tuckoo hello]$
[mthomas@tuckoo hello]$
[mthomas@tuckoo hello]$ pbsnodes node8
node8
     state = free
     np = 8
     properties = type1
     ntype = cluster
     status = rectime=1423800494,varattr=,jobs=,state=free,netload=1866286069,gres=,loadave=0.52,ncpus=8,
     physmem=49414804kb,availmem=95577436kb,totmem=96300496kb,idletime=106508,
     nusers=1,nsessions=1,sessions=1884,uname=Linux node8
     2.6.32-220.17.1.el6.x86_64 #1 SMP Wed May 16 00:01:37 BST 2012 x86_64,opsys=linux
     gpus = 0

[mthomas@tuckoo hello]$
```

More condensed summary: *pbsnodes -a | grep properties*

tuckoo naming convention:

- core4 node has 4 cores
- core6 node has 6 cores
- core16 node has 16 cores

```
[mthomas@tuckoo]$ pbsnodes -a  | grep properties
     properties = core4,type1
     properties = core4,type1
     properties = core4,type1
     properties = core4,type1
     properties = type1
     properties = core6,type2
     properties = core6,type2
     properties = core6,type2
     properties = core16,type2
     properties = core16,type2
```

Network information

```
[mthomas@tuckoo ~]$ hostname
tuckoo
[mthomas@tuckoo ~]$
[mthomas@tuckoo ~]$ cat /proc/net/dev
Inter-|   Receive                                                |  Transmit
 face |bytes    packets errs drop fifo frame compressed multicast|bytes    packets errs dr
    lo: 665015398  704989    0    0    0     0          0         0 665015398  704989    0
  eth0: 22597919775 19820450    0    0    0     0          0         0 2245854104 8501040
  eth1:       0       0    0    0    0     0          0         0        0       0    0
  eth2: 1995666451 18784458    0    0    0     0          0     36884 21099375732 18785051
[mthomas@tuckoo ~]$

[mthomas@tuckoo ~]$ ssh node1 "cat /proc/net/dev"
mthomas@node1's password:
Inter-|   Receive                                                |  Transmit
 face |bytes    packets errs drop fifo frame compressed multicast|bytes    packets errs dr
    lo: 3039772   10402    0    0    0     0          0         0  3039772   10402    0
  eth0:6276061155 5974043    0    0    0     0          0       102 8119134972 7258708
   ib0:       0       0    0    0    0     0          0         0        0       0    0

[mthomas@tuckoo ~]$ ssh node11 "cat /proc/net/dev"
mthomas@node11's password:
Inter-|   Receive                                                |  Transmit
 face |bytes    packets errs drop fifo frame compressed multicast|bytes    packets errs dr
    lo: 1838675    7213    0    0    0     0          0         0  1838675    7213    0
  eth0:2067241887 2195727    0    0    0     0          0     37114 414102996  744255    0
  eth1:       0       0    0    0    0     0          0         0        0       0    0
```

Network information

```
[mthomas@tuckoo ~]$ ssh node1 "/sbin/ifconfig eth0"
mthomas@node1's password:
eth0      Link encap:Ethernet  HWaddr 00:24:81:B2:06:F7
          inet addr:10.0.1.1  Bcast:10.0.1.255  Mask:255.255.255.0
          inet6 addr: fe80::224:81ff:feb2:6f7/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:5976844 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7260934 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6276774881 (5.8 GiB)  TX bytes:8119389701 (7.5 GiB)
          Interrupt:17



[mthomas@tuckoo ~]$ ssh node11 "/sbin/ifconfig ib0"
mthomas@node11's password:
ib0       Link encap:InfiniBand  HWaddr A0:04:02:20:FE:80:00:00:00:00:00:00:00:00:00:00
          BROADCAST MULTICAST  MTU:4092  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1024
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

Ifconfig uses the ioctl access method to get the full address information, which limits ha
Because Infiniband address has 20 bytes, only the first 8 bytes are displayed correctly.
Ifconfig is obsolete! For replacement check ip.
```

Job Example: Serial Hello World

```
#include <stdio.h>
#include <unistd.h>

int main(void)
{
   char cptr[100];

   gethostname(cptr,100);
   printf("hello, world from %s\n", cptr);

   return 0;
}
```

Compiling & Running Serial Code – Command Line

```
[mthomas@tuckoo hello]$ ls hello*
-rw-r--r-- 1 mthomas mthomas 263 Feb 12 10:00 hello.c
-rwx------ 1 mthomas mthomas 791 Feb 12 09:54 hello.f90

==================================
COMPILE USING   mpicc
==================================

[mthomas@tuckoo hello]$ mpicc -o hello hello.c
[mthomas@tuckoo hello]$
[mthomas@tuckoo hello]$ ls hello*
-rwxrwxr-x 1 mthomas mthomas 7276 Feb 12 10:03 hello
-rw-r--r-- 1 mthomas mthomas  263 Feb 12 10:00 hello.c
-rwx------ 1 mthomas mthomas  791 Feb 12 09:54 hello.f90
[mthomas@tuckoo hello]$

==================================
RUN SERIAL CODE FROM COMMAND LINE
==================================

[mthomas@tuckoo hello]$ ./hello
hello, world from tuckoo
[mthomas@tuckoo hello]$
```

```
==================================
RUN SERIAL CODE FROM COMMAND LINE
USING A PARALLEL ENVIRONMENT (mpirun)
==================================

[mthomas@tuckoo hello]$ mpirun -np 4 ./hello
hello, world from tuckoo
hello, world from tuckoo
hello, world from tuckoo
hello, world from tuckoo
```

How did the serial code run 4 times? *mpirun*.

Using Batch Queue to run Serial Code

```
=====================================
= You don't need to recompile code
=====================================
[mthomas@tuckoo hello]$ cat batch.hello.1
#!/bin/sh
# this example batch script requests hello processors...
# "man pbs_resources"
#PBS -V
#####  SPECIFY NODE ATTRIBUTES  #######
#PBS -l nodes=1:ppn=1:node1
#####
#PBS -N hello.1
#PBS -j oe
#PBS -q batch
cd $PBS_O_WORKDIR
NCORES=`wc -w < $PBS_NODEFILE`
echo "Running $PBS_JOBNAME, using $NCORES cores..."
mpirun -np 1 -hostfile $PBS_NODEFILE  ./hello
[mthomas@tuckoo hello]$
[mthomas@tuckoo hello]$ qsub batch.hello.1
177.tuckoo.sdsu.edu
[mthomas@tuckoo hello]$ qstat
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
177.tuckoo                 hello.1          mthomas         00:00:00 C batch
[mthomas@tuckoo hello]$
[mthomas@tuckoo hello]$ cat hello.1.o177
hello.1 using 1 cores...
hello, world from node11
```

Getting Node and Core ID

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
/* This code is adapted from an example at: http://brokestream.com/procstat.html*/
int get_cpu_id()    {
    /* Get the the current process' stat file from the proc filesystem */
    FILE* procfile = fopen("/proc/self/stat", "r");
    long to_read = 8192;
    char buffer[to_read];
    int read = fread(buffer, sizeof(char), to_read, procfile);
    fclose(procfile);

    // Field with index 38 (zero-based counting) is the one we want
    char* line = strtok(buffer, " ");
    int i;
    for (i = 1; i < 38; i++) {
        line = strtok(NULL, " ");
    }
    line = strtok(NULL, " ");
    int cpu_id = atoi(line);
    return cpu_id;
}

int main(void)   {
  char cptr[100];
  int cpuid;

  gethostname(cptr,100);
  cpuid = get_cpu_id();
  printf("hello, world from node: %s, core: %d\n", cptr, cpuid);

  return 0;
}
```

Showing Node & Core ID Using One Node

```
[mthomas@tuckoo hello]$ qsub batch.hello-cpuid
198.tuckoo.sdsu.edu
[mthomas@tuckoo hello]$ qstat -a
tuckoo.sdsu.edu:
                                                                  Req'd  Req'd   Elap
Job ID               Username Queue    Jobname          SessID NDS   TSK Memory Time  S Time
-------------------- -------- -------- ---------------- ------ ----- --- ------ ----- - -----
220.tuckoo.sdsu.     mthomas  batch    hello-cpuid         --      1   8    --    --  Q   --
[mthomas@tuckoo hello]$ cat hello-cpuid.o197
cat: hello-cpuid.o197: No such file or directory
[mthomas@tuckoo hello]$ cat hello-cpuid.o198
-bash: BASH_FUNC_module(): line 0: syntax error near unexpected token `)'
-bash: BASH_FUNC_module(): line 0: `BASH_FUNC_module() () {  eval `/usr/bin/modulecmd bash $*`'
-bash: error importing function definition for `BASH_FUNC_module'
hello-cpuid-test using 16 cores...
[mthomas@tuckoo hello]$ cat hello-cpuid.o220
PBS: job name is hello-cpuid
hello, world from node: node8, core: 7
hello, world from node: node8, core: 3
hello, world from node: node8, core: 4
hello, world from node: node8, core: 5
hello, world from node: node8, core: 2
hello, world from node: node8, core: 1
hello, world from node: node8, core: 5
hello, world from node: node8, core: 6
```

Set Attributes: #PBS -l nodes=1:ppn=8

Showing Node & Core ID Using Two Nodes

```
[mthomas@tuckoo hello]$ qsub batch.hello-cpuid
221.tuckoo.sdsu.edu
[mthomas@tuckoo hello]$ qstat -a

tuckoo.sdsu.edu:
                                                                  Req'd  Req'd   Elap
Job ID               Username Queue    Jobname          SessID NDS   TSK Memory Time  S Time
-------------------- -------- -------- ---------------- ------ ----- --- ------ ----- - -----
217.tuckoo.sdsu.     mthomas  batch    hello-cpuid       25132     2   8    --    --  C 00:00
218.tuckoo.sdsu.     mthomas  batch    hello-cpuid       25164     2   8    --    --  C 00:00
219.tuckoo.sdsu.     mthomas  batch    hello-cpuid        6775     1   8    --    --  C 00:00
220.tuckoo.sdsu.     mthomas  batch    hello-cpuid        6803     1   8    --    --  C 00:00
221.tuckoo.sdsu.     mthomas  batch    hello-cpuid       25198     2   8    --    --  C 00:00
[mthomas@tuckoo hello]$ cat hello-cpuid.o221
PBS: job name is hello-cpuid
hello, world from node: node2, core: 1
hello, world from node: node2, core: 3
hello, world from node: node2, core: 2
hello, world from node: node2, core: 0
hello, world from node: node1, core: 2
hello, world from node: node1, core: 3
hello, world from node: node1, core: 3
hello, world from node: node1, core: 2
```

Set Attributes: #PBS -l nodes=2:ppn=2:core4

Printing Out Detailed PBS Information

```
[mthomas@tuckoo hello]$ cat batch.details
#!/bin/sh
#PBS -V
#PBS -l nodes=1:ppn=4
#PBS -N batch.details
#PBS -j oe
#PBS -q batch
cd $PBS_O_WORKDIR
echo ----------------------------------------------------
echo "    date: " `date`
echo "hostname: "  `hostname`
echo "  whoami: "  `whoami`
echo "     pwd: "  `pwd`
echo ----------------------------------------------------
echo -n 'Job is running on node '; cat $PBS_NODEFILE
echo ----------------------------------------------------
echo PBS: qsub is running on $PBS_O_HOST
echo PBS: originating queue is $PBS_O_QUEUE
echo PBS: executing queue is $PBS_QUEUE
echo PBS: working directory is $PBS_O_WORKDIR
echo PBS: execution mode is $PBS_ENVIRONMENT
echo PBS: job identifier is $PBS_JOBID
echo PBS: job name is $PBS_JOBNAME
echo PBS: node file is $PBS_NODEFILE
echo PBS: current home directory is $PBS_O_HOME
echo PBS: PATH = $PBS_O_PATH
echo ----------------------------------------------------
NCORES=`wc -w < $PBS_NODEFILE`
echo "Running: $PBS_JOBNAME, using $NCORES cores..."
echo ----------------------------------------------------
mpirun -np 4 -hostfile $PBS_NODEFILE   ./hello-cpuid
```

Printing Out Detailed PBS Information

```
[mthomas@tuckoo hello]$ cat batch.details.o2070
-------------------------------------------------------
    date:  Mon Feb 23 13:59:19 PST 2015
hostname:  node2
  whoami:  mthomas
     pwd:  /home/mthomas/pardev/hello
-------------------------------------------------------
Job is running on node
node2
node2
node2
node2
-------------------------------------------------------
PBS: qsub is running on tuckoo.sdsu.edu
PBS: originating queue is batch
PBS: executing queue is batch
PBS: working directory is /home/mthomas/pardev/hello
PBS: execution mode is PBS_BATCH
PBS: job identifier is 2070.tuckoo.sdsu.edu
PBS: job name is batch.details
PBS: node file is /var/spool/torque/aux//2070.tuckoo.sdsu.edu
PBS: current home directory is /home/mthomas
PBS: PATH = /usr/lib64/qt-3.3/bin:/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/usr/local/openmpi/bin:
/usr/local/torque/bin:/usr/local/torque/sbin:/usr/local/cuda/bin:/usr/local/tau/x86_64/bin:/usr/local/vampirtrace/bin:
/opt/pgi/linux86-64/11.0/bin
-------------------------------------------------------
Running: batch.details, using 4 cores...
-------------------------------------------------------
hello, world from node: node2, core: 3
hello, world from node: node2, core: 1
hello, world from node: node2, core: 0
hello, world from node: node2, core: 2
```

Passing Command Line Arguments

```
/*
 * File: hello-arg.c by Mary Thomas, 2/12/15
 */
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>

/*
* This code is adapted from an example at:  http://brokestream.com/procstat.html
*/
int get_cpu_id()  {
   ..
}

int main (int argc, char* argv[])
{
  long cmdarg;
  char cptr[100];
  int cpuid;

  /* read long integer  from the command line */
  if (argc != 2) {
      printf("usage: hello-arg <integer number>\n");
      return 0;
  }
  cmdarg = strtol(argv[1], NULL, 10);

  gethostname(cptr,100);
  cpuid = get_cpu_id();
  printf("hello, world from node: %s, core: %d, cmdarg= %ld\n", cptr, cpuid, cmdarg);

  return 0;
}
```

Passing Command Line Arguments: Batch Script

Example #1: Hard code the value in the script

```
[mthomas@tuckoo hello]$ cat batch.hello-arg
#!/bin/sh
#PBS -V
#PBS -l nodes=2:ppn=4:core4
#PBS -N hello-arg
#PBS -j oe
#PBS -q batch
cd $PBS_O_WORKDIR
echo "PBS: job name is $PBS_JOBNAME "
NCORES=`wc -w < $PBS_NODEFILE`
echo "$PBS_JOBNAME running using $NCORES cores..."
mpirun -np 8 -hostfile $PBS_NODEFILE --nooversubscribe ./hello-arg 12345
[mthomas@tuckoo hello]$
[mthomas@tuckoo hello]$ qsub batch.hello-arg
229.tuckoo.sdsu.edu
[mthomas@tuckoo hello]$ qstat
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
229.tuckoo                 hello-arg        mthomas         00:00:00 C batch
[mthomas@tuckoo hello]$
[mthomas@tuckoo hello]$ cat hello-arg.o229
PBS: job name is hello-arg
hello-arg running using 8 cores...
hello, world from node: node2, core: 3, cmdarg= 12345
hello, world from node: node2, core: 1, cmdarg= 12345
hello, world from node: node2, core: 2, cmdarg= 12345
hello, world from node: node2, core: 0, cmdarg= 12345
hello, world from node: node1, core: 3, cmdarg= 12345
hello, world from node: node1, core: 2, cmdarg= 12345
hello, world from node: node1, core: 3, cmdarg= 12345
hello, world from node: node1, core: 0, cmdarg= 12345
```

Passing Command Line Arguments: Batch Script

Example #1: Pass the argument to the *qsub* command

```
[mthomas@tuckoo hello]$ cat batch.hello-arg-qsubv
#!/bin/sh
# this batch script takes variables passed to the qsub command
#    and passes them to the hello-arg executable as command line arguments. Usage:
#  %qsub -v CMDARG=7654  batch.hello-arg-qsubv
#
#PBS -V
#PBS -l nodes=1:ppn=4:core4
#PBS -N hello-arg-qsubvars
#PBS -j oe
#PBS -q batch
cd $PBS_O_WORKDIR
echo "PBS: job name is $PBS_JOBNAME "
NCORES=‘wc -w < $PBS_NODEFILE‘
echo "$PBS_JOBNAME running using $NCORES cores..."
mpirun -np 4 -hostfile $PBS_NODEFILE --nooversubscribe ./hello-arg $CMDARG

[mthomas@tuckoo hello]$ qsub batch.hello-arg-qsubv
230.tuckoo.sdsu.edu
[mthomas@tuckoo hello]$ qsub -v CMDARG=9876 batch.hello-arg-qsubv
231.tuckoo.sdsu.edu
[mthomas@tuckoo hello]$
[mthomas@tuckoo hello]$ qstat
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
231.tuckoo                 ...-arg-qsubvars mthomas         00:00:00 C batch
[mthomas@tuckoo hello]$
[mthomas@tuckoo hello]$ cat hello-arg-qsubvars.o231
PBS: job name is hello-arg-qsubvars
hello-arg-qsubvars running using 8 cores...
hello, world from node: node1, core: 0, cmdarg= 9876
hello, world from node: node1, core: 1, cmdarg= 9876
hello, world from node: node1, core: 2, cmdarg= 9876
hello, world from node: node1, core: 3, cmdarg= 9876
```

Code Ex: two.c (J. Otto)

```
int main (int argc, char* argv[])
{
  int rank, num_nodes, ierr, other;
  int tkn;
  char cptr[100];
  MPI_Status status;
  gethostname(cptr,100);
  ierr = MPI_Init(&argc, &argv);
  if (ierr != MPI_SUCCESS) {
    printf("MPI initialization error\n"); return 0;
  }
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Comm_size(MPI_COMM_WORLD, &num_nodes);
  if (num_nodes != 2) {
    if (rank == ROOT) {
      printf("program error: runs on 2 processes!\n");
    }
    MPI_Finalize();
    return 0;
  }
    if (rank == 0) { /* get things started */
    token = 0;    other = OTHER;
    MPI_Send(&tkn,1,MPI_INT,other,0,MPI_COMM_WORLD);
    MPI_Recv(&tkn,1,MPI_INT,other,0,MPI_COMM_WORLD,
                  &status);
    printf("process %d: (%s) received the token...\n",
                  rank, cptr);
    if (token == 1) {
      printf("process %d: token= %d, matches (things
              look ok).\n",   rank, tkn);
    }
    else {
      printf("process %d: ERR -- val of token (%d)
              does not match expected (%d)!\n", rank, token, 1)
    }
  }
  else {
    other = ROOT;
    MPI_Recv(&token, 1, MPI_INT, other, 0,
                          MPI_COMM_WORLD, &status);
    token++;
    MPI_Send(&token, 1, MPI_INT, other, 0,
                          MPI_COMM_WORLD);
  }
  MPI_Finalize();
  return 0;
}
```

```
------------------------------------------------
-  Copy examples
------------------------------------------------
[mthomas@tuckoo ]$ cp -r /examples/ ~/ex.2016
[mthomas@tuckoo pardev]$ cd ex.2016

------------------------------------------------
-  make
------------------------------------------------
[mthomas@tuckoo ex.2016]$ make two
mpicc -o two two.c

------------------------------------------------
-   BATCHSCRIPT:  batch.twoF
------------------------------------------------
[mthomas@tuckoo hw1]$ cat batch.twoF
#!/bin/sh
# for more info on requesting specific
# nodes see "man pbs_resources"
#PBS -V
#PBS -l nodes=2:ppn=1:core4
#PBS -N twoF
#PBS -j oe
#PBS -q batch
echo NODEFILE: $PBS_NODEFILE
cd $PBS_O_WORKDIR
NCORES=`wc -w < $PBS_NODEFILE`
echo "twoF test using $NCORES cores..."
mpirun -np 2 -host node6,node7 ./two

### use nooversubsrcibe to run 1 job/pe or core
###mpirun -np 4 -hostfile $PBS_NODEFILE  --nooversubscribe ./looptstp
```

```
------------------------------------------------
-  RUN JOB
------------------------------------------------

[mthomas@tuckoo ex.2016]$ qsub batch.two
16442.tuckoo.sdsu.edu

------------------------------------------------
-  OUTPUT
------------------------------------------------

[mthomas@tuckoo ex.2016]$ cat two.o16442
NODEFILE: /var/spool/torque/aux//16442.tuckoo.sdsu.edu
twoF test using 8 cores...
process 0 of 2 (on node6)
process 1 of 2 (on node7)
success (process 0)
```

MPI Code: mpi-hello.c Code

```
% program mpi-hello.c, by Mary Thomas
% reads a command line argument and prints out the value
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include "mpi.h"

int main (int argc, char* argv[])
{
  long myarg;
  int rank, nprocs, ierr, i, error=0;
  MPI_Status status;

  ierr = MPI_Init(&argc, &argv);
  if (ierr != MPI_SUCCESS) {
    printf("MPI initialization error\n");
  }

  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Comm_size(MPI_COMM_WORLD, &nprocs);

  /* get a number from the command line */
  if (argc != 2) {
      printf("usage: mpi-hello <integer number>\n");
      MPI_Finalize();
      return 0;
  }
  myarg = strtol(argv[1], NULL, 10);
  printf("Hello From Processor: rank of  %d nprocs: %d;  CMDARG= %ld\n", rank, nprocs,myarg);

  MPI_Finalize();
  return 0;
}
```

Passing cmd args

```
[mthomas@tuckoo hello]$ qsub batch.mpi-hello
40.tuckoo.sdsu.edu
[mthomas@tuckoo hello]$
[mthomas@tuckoo hello]$ qstat -a

tuckoo.sdsu.edu:
                                                                  Req'd  Req'd   Elap
Job ID               Username Queue    Jobname          SessID NDS   TSK Memory Time  S Time
-------------------- -------- -------- ---------------- ------ ----- --- ------ ----- - -----
37.tuckoo.sdsu.e     mthomas  batch    hello             22972     1   4    --    --  C 00:00
38.tuckoo.sdsu.e     mthomas  batch    mpi-hello         22560     2   8    --    --  C 00:00
39.tuckoo.sdsu.e     mthomas  batch    mpi-hello         22594     2   8    --    --  C 00:00
40.tuckoo.sdsu.e     mthomas  batch    mpi-hello         22630     2   8    --    --  C 00:00
[mthomas@tuckoo hello]$
[mthomas@tuckoo hello]$ cat mpi-hello.o40
PBS: job name is mpi-hello
Hello From Processor: rank of  0 nprocs: 8;  CMDARG= 12345
Hello From Processor: rank of  3 nprocs: 8;  CMDARG= 12345
Hello From Processor: rank of  5 nprocs: 8;  CMDARG= 12345
Hello From Processor: rank of  2 nprocs: 8;  CMDARG= 12345
Hello From Processor: rank of  6 nprocs: 8;  CMDARG= 12345
Hello From Processor: rank of  7 nprocs: 8;  CMDARG= 12345
Hello From Processor: rank of  1 nprocs: 8;  CMDARG= 12345
Hello From Processor: rank of  4 nprocs: 8;  CMDARG= 12345
```

A comment about nondeterministic outputs

mpi_hello.c: sends local message to master, where it is printed to STDIO.

```
if (my_rank != 0) {    /* Create message */
    printf(greeting, "Greetings from Slave: %d of %d!",
           my_rank, comm_sz);
    /* Send message to process 0 */
   MPI\_Send(greeting, strlen(greeting)+1, MPI_CHAR, 0, 0,
            MPI_COMM_WORLD);
} else {
    /* Print my message */
    printf("Greetings from Master: %d of %d!\n", my_rank, c
    for (int q = 1; q < comm_sz; q++) {
        /* Receive message from process q */
        MPI_Recv(greeting, MAX_STRING, MPI_CHAR, q,
            0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
        /* Print message from process q */
        printf("%s\n", greeting);
    }   }
```

What happens when we ask for more nodes than we put into request or are available on the nodes? Set np=16

```
 [mthomas@tuckoo hello]$ cat batch.mpi-hello
#!/bin/sh
# this example batch script requests mpi-hello processors...
# for more info on requesting specific nodes see
# PBS -V is no longer recommended.
#PBS -V
#PBS -l nodes=2:ppn=4:core4
#PBS -N mpi-hello
#PBS -j oe
#PBS -q batch
cd $PBS_O_WORKDIR

echo "PBS: job name is $PBS_JOBNAME "
###mpirun -np 8 -hostfile $PBS_NODEFILE --nooversubscribe ./mpi-hello  12345
mpirun -np 13 -hostfile $PBS_NODEFILE  ./mpi-hello  12345
[mthomas@tuckoo hello]$
[mthomas@tuckoo hello]$ cat mpi-hello.o41
PBS: job name is mpi-hello
Hello From Processor: rank of  0 nprocs: 13;  CMDARG= 12345
Hello From Processor: rank of  2 nprocs: 13;  CMDARG= 12345
Hello From Processor: rank of  3 nprocs: 13;  CMDARG= 12345
Hello From Processor: rank of  5 nprocs: 13;  CMDARG= 12345
Hello From Processor: rank of  4 nprocs: 13;  CMDARG= 12345
Hello From Processor: rank of  7 nprocs: 13;  CMDARG= 12345
Hello From Processor: rank of  1 nprocs: 13;  CMDARG= 12345
Hello From Processor: rank of  8 nprocs: 13;  CMDARG= 12345
Hello From Processor: rank of  6 nprocs: 13;  CMDARG= 12345
Hello From Processor: rank of  12 nprocs: 13;  CMDARG= 12345
Hello From Processor: rank of  9 nprocs: 13;  CMDARG= 12345
Hello From Processor: rank of  11 nprocs: 13;  CMDARG= 12345
Hello From Processor: rank of  10 nprocs: 13;  CMDARG= 12345
```

Even though we asked for only 8 cores, we seem to have gottne 13!

Simple Hello World Batch script - Fix

```
#!/bin/sh
# this example batch script requests hello processors...
# for more info on requesting specific nodes see "man pbs_resources"
#PBS -V
#PBS -l nodes=2:ppn=2:core4
#PBS -N hello
#PBS -j oe
#PBS -q batch
cd $PBS_O_WORKDIR
echo ------------------------------------------------------
echo -n 'Job is running on node '; cat $PBS_NODEFILE
echo ------------------------------------------------------
echo PBS: qsub is running on $PBS_O_HOST
echo PBS: originating queue is $PBS_O_QUEUE
echo PBS: executing queue is $PBS_QUEUE
echo PBS: working directory is $PBS_O_WORKDIR
echo PBS: execution mode is $PBS_ENVIRONMENT
echo PBS: job identifier is $PBS_JOBID
echo PBS: job name is $PBS_JOBNAME
echo PBS: node file is $PBS_NODEFILE
echo PBS: current home directory is $PBS_O_HOME
echo PBS: PATH = $PBS_O_PATH
echo ------------------------------------------------------
NCORES=`wc -w < $PBS_NODEFILE`
echo "hello-test requesting $NCORES cores..."
```

mpirun -np 16 -hostfile $PBS_NODEFILE **–nooversubscribe** ./hello

Simple Hello World Batch script - now oversubscribing fails

```
-------------------------------------------------------------------------
[mthomas@tuckoo hello]$ cat batch.mpi-hello
#!/bin/sh
# this example batch script requests mpi-hello processors...
# for more info on requesting specific nodes see
# PBS -V is no longer recommended.
#PBS -V
#PBS -l nodes=2:ppn=4:core4
#PBS -N mpi-hello
#PBS -j oe
#PBS -q batch
cd $PBS_O_WORKDIR

echo "PBS: job name is $PBS_JOBNAME "
mpirun -np 18 -hostfile $PBS_NODEFILE --nooversubscribe ./mpi-hello  12345

[mthomas@tuckoo hello]$ !qsub
qsub batch.mpi-hello
43.tuckoo.sdsu.edu

[mthomas@tuckoo hello]$
[mthomas@tuckoo hello]$ cat mpi-hello.o43
PBS: job name is mpi-hello
-------------------------------------------------------------------------
There are not enough slots available in the system to satisfy the 18 slots
that were requested by the application:
  ./mpi-hello

Either request fewer slots for your application, or make more slots available
for use.
-------------------------------------------------------------------------
[mthomas@tuckoo hello]$

-------------------------------------------------------------------------
```