Hummingbird: Things you'll need

Information : `https://www.hb.ucsc.edu/`

Things you'll need to be able to work on Hummingbird:

- ✓ Some laptop or desktop on a network to access the machine
- ✓ SSH : access to hummingbird via ssh `<yourcruzid>@hb.soe.ucsc.edu`
- ✓ CruzID and blue password
- ✓ Basic knowledge of Unix command line commands – ls, cd, mkdir, cp, …
- ✓ vi : editor on grape to be able to write programs (emacs?)

Login using your UCSC account

If you are off-campus, you will need to use VPN to log in. If you are unfamiliar, the instructions for downloading the VPN software to your remote machine and for using the VPN are here:

`https://its.ucsc.edu/vpn/`

Hummingbird

Master login (or head) node: "hb.ucsc.edu"

Cluster partitions: Use "Instruction"

Queue	No. of nodes	Node	Chips	Clock	Cores per node	Memory per node	Total cores
Instruction	2	AMD	AMD 6000	?? GHz	48	192GB	96
	3	AMD	AMD 6000	?? GHz	64	256GB	192
256x44	1	Intel	Intel XeonE5-2650v4	2.2 GHz	44	256GB	44
128x24	19	Intel	Intel Xeon E5-2699v4	2.2 GHz	24	128GB	456
96x24gpu4	1	Intel + 4 x Nvidia Tesla P100			24 (+4x 3584 GPU cores)	96GB (+4x 16GB GPU mem)	
Network:		Ethernet Speed: 10Gb/s					
File system:		BeeGFS 750TB					

Hummingbird: Basics

Where am I?: /hb/home/<username>

Modules: Changing the environment that you are working in

```
module avail
module list
module load <module>
module swap <mod_out> <mod_in>
```

e.g. `module load openmpi` (for MPI)

`module load gnu` (for gfortran) ← both done by default on hb!

Fortran compiling:

Regular Fortran: `gfortran <filename.f90>`

Options: -o <executable_file>, -O<1,2,3> (optimization), -g (debug)

MPI-Fortran combo: `mpif90 <filename.f90>` Similar options

OpenMP Fortran: `gfortran –fopenmp <filename.f90>`

Hummingbird: Basics

Batch job submission and management:

BATCH ONLY! You log in to the head node, but must submit jobs to run on the compute nodes through a batch job controller. No jobs can be run on the head node!

On Hummingbird, SLURM is the batch job controller

Comprehensive documentation at : https://slurm.schedmd.com/

(You can technically run in interactive mode, but we will do everything in batch mode here)

Essentials and general process:

Create and compile executable to be run using `mpif90` or `gfortran`

Create a batch job by editing a command file, `jobname.cmd`, with SLURM resource allocation commands and job execution commands

Submit the batch job: `sbatch jobname.cmd`

Monitor the execution of the job: `squeue –u <username>` (shows jobid)

Delete jobs: `scancel <jobid>`

Hummingbird: Basics

STEP 1: Edit source and create Fortran/MPI file: `hello_hb.f90`

MPI: `vi hello_hb.f90`

OpenMP: `vi omp_hello_hb.f90`

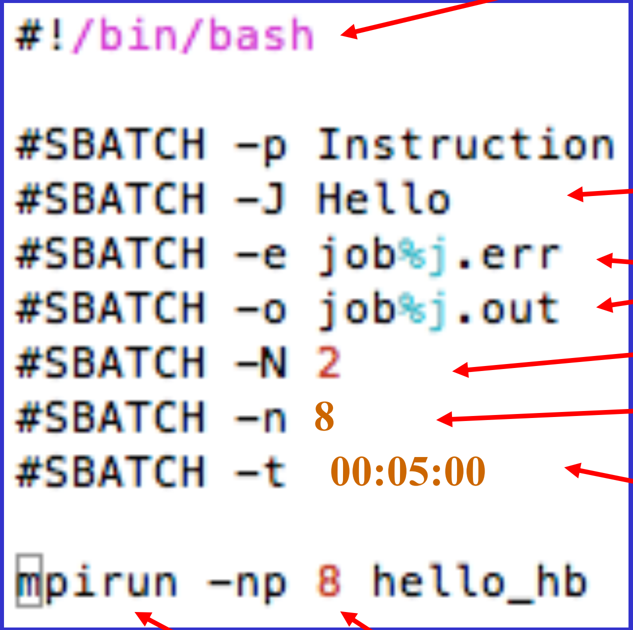
STEP 2: Compile Fortan/MPI source file into an executable:

MPI: `mpif90 –o hello_hb hello_hb.f90`

OpenMP: `gfortran –o omp_hello_hb -fopenmp omp_hello_hb.f90`

Hummingbird: Basics

STEP 3: Edit and create SLURM command file `hello_hb.cmd`



```
#!/bin/bash

#SBATCH -p Instruction
#SBATCH -J Hello
#SBATCH -e job%j.err
#SBATCH -o job%j.out
#SBATCH -N 2
#SBATCH -n 8
#SBATCH -t  00:05:00

mpirun -np 8 hello_hb
```

This #! command says which scripting shell you are using

Other # commands are instructions to SLURM

Send job to partition called "Instruction"

Jobname – shows up in squeue listing

Send error and regular output to "job<jobid>.<err,out>"

= --nodes : Use 2 nodes

= --ntasks : Use a total of 8 tasks = 8 MPI ranks
(note: can specify -- ntasks-per-node)

Allow this much time hh:mm:ss

Shell commands to run job and do other things

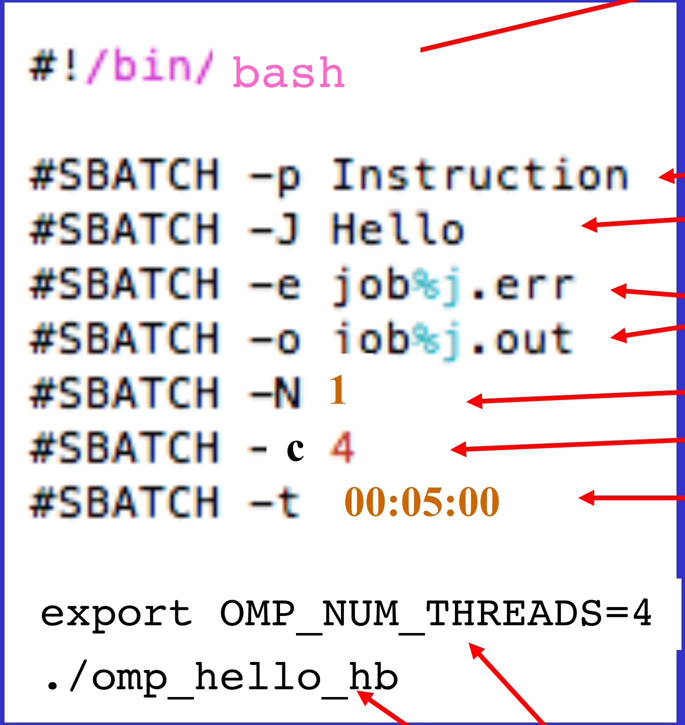
MPI

`np` is total number of MPI tasks = #ntasks = #nodes * #tasks_per_node

`mpirun` runs an MPI compiled executable `hello_hb`

Hummingbird: Basics

STEP 3: Edit and create SLURM command file `hello_hb.cmd`



```
#!/bin/ bash

#SBATCH -p Instruction
#SBATCH -J Hello
#SBATCH -e job%j.err
#SBATCH -o iob%j.out
#SBATCH -N 1
#SBATCH - c 4
#SBATCH -t  00:05:00

export OMP_NUM_THREADS=4
./omp_hello_hb
```

This #! command says which scripting shell you are using

Other # commands are instructions to SLURM

Send job to partition called “Instruction”

Jobname – shows up in squeue listing

Send error and regular output to “job<jobid>.<err,out>”

Use 1 node or else no shared memory!

Use 4 cpus (threads) per node

Allow this much time hh:mm:ss

Shell commands to run job and do other things

OpenMP

`OMP_NUM_THREADS` is total number of tasks = #nodes * #cpus_per_node

Run the OpenMP compiled executable `omp_hello`

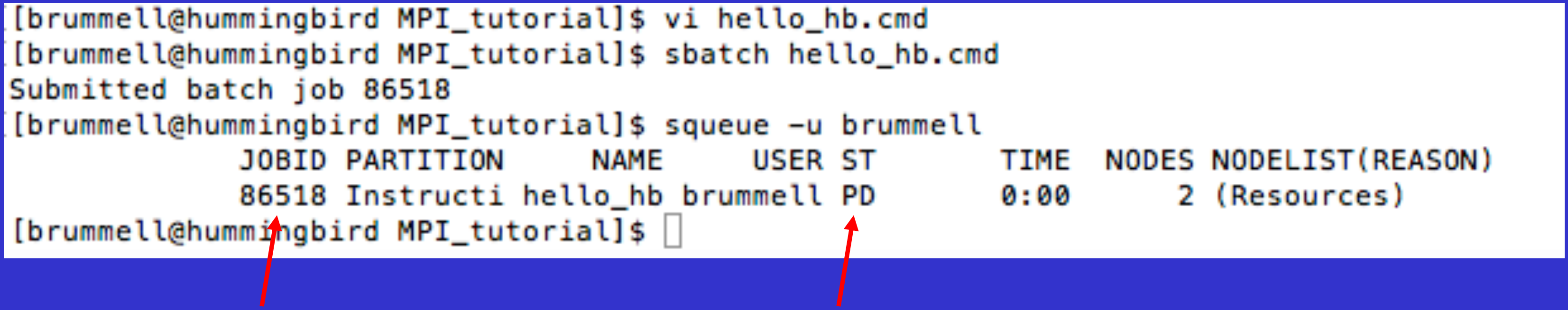
Hummingbird: Basics

STEP 4: Submit job to the partition (queue)

```
sbatch hello_hb.cmd
```

STEP 5: Monitor the job

```
squeue –u brummell
```



```
[brummell@hummingbird MPI_tutorial]$ vi hello_hb.cmd
[brummell@hummingbird MPI_tutorial]$ sbatch hello_hb.cmd
Submitted batch job 86518
[brummell@hummingbird MPI_tutorial]$ squeue -u brummell
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
             86518 Instructi hello_hb brummell PD       0:00      2 (Resources)
[brummell@hummingbird MPI_tutorial]$
```

Job ID:

Use to kill job if necessary

Job state codes:

PD = pending

R = running

CG = completing

CD = completed

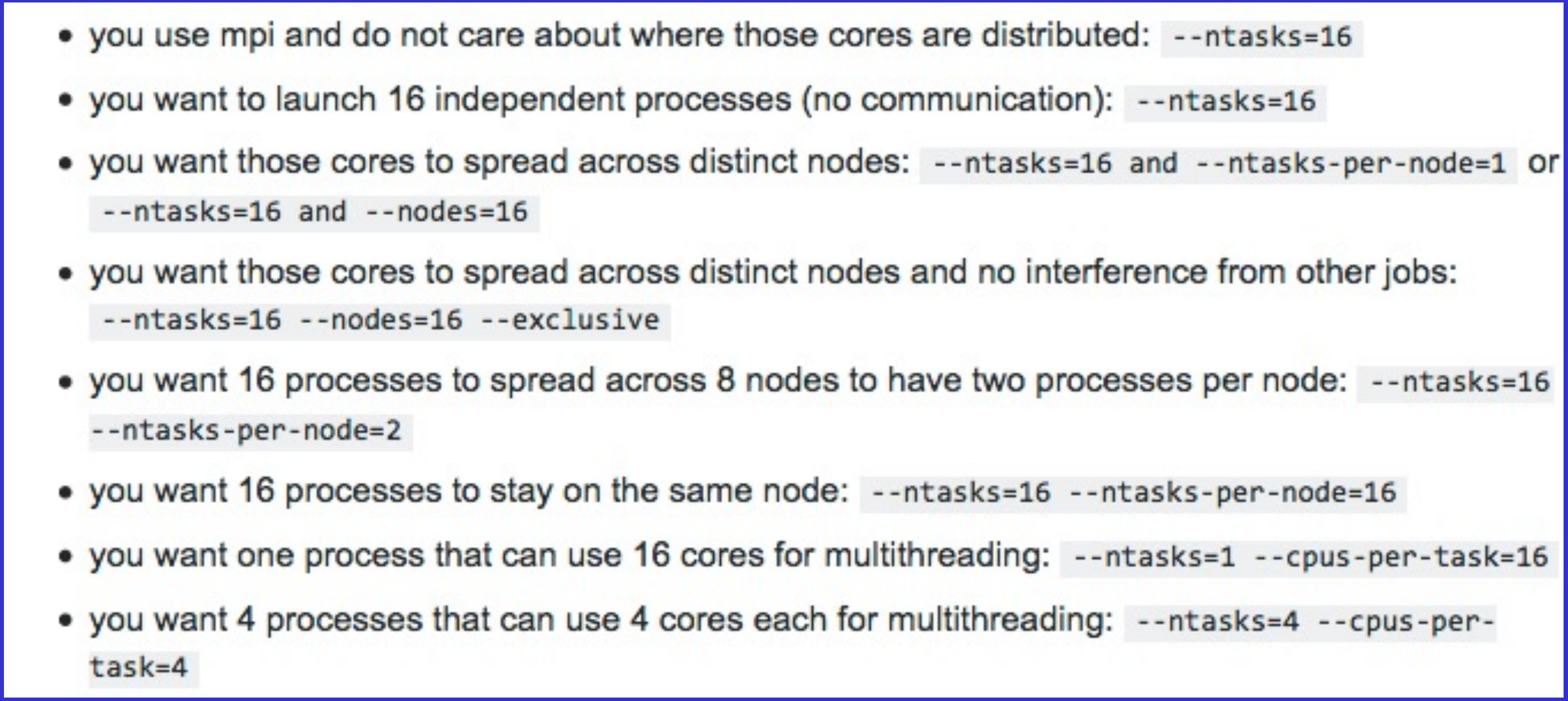
Delete jobs: `scancel <jobid>`

Hummingbird: Basics

INTERACTIVE

You can run interactively on Hummingird but it is more complicated and I really don't recommend it!

SLURM massive overspecification!   Say you want 16 processes:



- you use mpi and do not care about where those cores are distributed: `--ntasks=16`
- you want to launch 16 independent processes (no communication): `--ntasks=16`
- you want those cores to spread across distinct nodes: `--ntasks=16 and --ntasks-per-node=1` or `--ntasks=16 and --nodes=16`
- you want those cores to spread across distinct nodes and no interference from other jobs: `--ntasks=16 --nodes=16 --exclusive`
- you want 16 processes to spread across 8 nodes to have two processes per node: `--ntasks=16 --ntasks-per-node=2`
- you want 16 processes to stay on the same node: `--ntasks=16 --ntasks-per-node=16`
- you want one process that can use 16 cores for multithreading: `--ntasks=1 --cpus-per-task=16`
- you want 4 processes that can use 4 cores each for multithreading: `--ntasks=4 --cpus-per-task=4`

Lux : should be the same!

EXCEPT different queues (partitions): Use "windfall"

Use	Node names	Queue	No. of node s	Node	Chips	Clock	Cores per node	Memory per node	Total cores
Login	lux-0,. Lux-1, lux-0		3						
Compute CPU-only	node001 – node080	Windfall, defq (24 hours, 16 nodes)	80	Dell	2x Intel Xeon Gold 6248 (Cascade Lake)		2x20	192GB	3200
Compute CPU+GPU	Gpu001-gpu028	Windfall, gpuq (24 hours, 16 nodes)	28	Dell	2x Intel Xeon Gold 6248 (Cascade Lake) + 2x NVIDIA 32GB V100		2x20	192GB + 2 x 32GB	
Network:	Mellanox HDR Infiniband Speed: 100Gb/s								
File system:	SSD 2.4TB								

Grape vs Hummingbird/Lux

Note that they are very similar but slightly different!

The different batch job systems make for slighlty different usage.

This is very common on moving between different supercomputers!

Grape: Things you’ll need

Things you’ll need to be able to work on grape:

- ✓ Some laptop or desktop on a network to access the machine
- ✓ SSH : access to grape via ssh grape.soe.ucsc.edu
- ✓ Basic knowledge of Unix command line commands – ls, cd, mkdir, cp, …
- ✓ vi : editor on grape to be able to write programs (emacs?)
- ✓ gfortran : Fortran compiler on grape

  (Intel Fortran: `/opt/intel/Compiler/11.1/073/bin/intel64/ifort`)

Login using your SOE account

Grape: DMP

Master login node "`grape.soe.ucsc.edu`"

Compute nodes: compute-0-i

Queue	No. of nodes	compute-0-i	Node	Sockets	Chips	Clock	Cores per chip	Memory per chip	Cores per node	Memory per node	Total cores
master	1	grape	Dell PowerEdge 2950	2	Intel Xeon 5300	2.66 GHz	4	8GB	8	16GB	8
orig	5	0-4	Dell PowerEdge 1950	2	Intel Xeon 5300	2.33 GHz	4	8GB	8	16GB	40
new	6	5-11	Dell PowerEdge R610	2	Intel Xeon 5500	2.40 GHz	4	8GB	8	16GB	48
newest	8	12-19	Dell PowerEdge R420	2	Intel Xeon Sandy Bridge E5-2470	2.30 GHz	8	16GB	16	32GB	128

Grape: SMP

"Analysis nodes" = "fat nodes"

Machines: jerez, muscat, mencia

NOTE: no MPI on these machines. But OpenMP does work. Use grape for MPI.

No. of nodes	Name	Node	Sockets	Chips	Clock	Cores per chip	Memory per chip	Cores per node	Memory per node	Total cores
1	jerez	Dell PowerEdge R820	4	Intel Xeon Sandy Bridge E5-4640	2.7GHz	8	16GB	32	64GB	32
1	muscat	Dell PowerEdge R820	4	Intel Xeon Sandy Bridge E5-4640	2.7GHz	8	16GB	32	64GB	32
1	mencia	Dell PowerEdge R820	4	Intel Xeon Sandy Bridge E5-4640	2.7GHz	8	16GB	32	64GB	32

Grape: Basics

Where am I?: /soe/<username>

Modules: Changing the environment that you are working in

```
module avail
module list
module load <module>
module swap <mod_out> <mod_in>
```

e.g. `module load rocks-openmpi` (for MPI)

`module load gcc-6.2.0` (for gfortran) ← both done by default!

Fortran compiling:

Regular Fortran: `gfortran <filename.f90>`

Options: -o <executable_file>, -O<1,2,3> (optimization), -g (debug)

MPI-Fortran combo: `mpif90 <filename.f90>` (Similar options)

OpenMP Fortran: `gfortran –fopenmp <filename.f90>`

Grape: Basics

Running non-parallel code

STEP 1: Log in to any single machine

```
ssh jerez.soe.ucsc.edu
ssh muscat.soe.ucsc.edu
ssh mencia.soe.ucsc.edu
```

`ssh grape.soe.ucsc.edu` and then `ssh compute-0-I (for an i=1,19)`

Do not login to the Hummingbird cluster! (It is possible, but it is just a lot more complicated!)

STEP 2: Compile code

```
gfortran –o example example.f90
```

STEP 3: Run code

```
./example
```

Grape: Basics

Running parallel code

Batch job submission and management:

You log in to the head node, but must submit jobs to run on the compute nodes through a batch job controller. No jobs can be run on the head node!

On Grape, the batch job controller is the Maui-Torque Portable Batch System (PBS)

Comprehensive documentation at : https://www.pbsworks.com

You can run in interactive mode (but get used to doing things in batch mode!).

Essentials and general process:

Create and compile executable to be run using `mpif90` or `gfortran –fopenmp`

Either: 1. Run interactively; 2. Run in batch.

To run in batch, create a batch job by editing a command file, `jobname.cmd`, with PBS resource allocation commands and job execution commands

Submit the batch job: `qsub jobname.cmd`

Monitor the execution of the job: `qstat –u <username>` (shows jobid)

Delete jobs: `qdel <jobid>`

Grape: Basics

STEP 1: Edit source and create Fortran/MPI file: `hello.f90`

MPI: `vi hello.f90`

OpenMP: `vi omp_hello.f90`

STEP 2: Compile Fortan/MPI source file into an executable:

MPI: `mpif90 –o hello hello.f90`

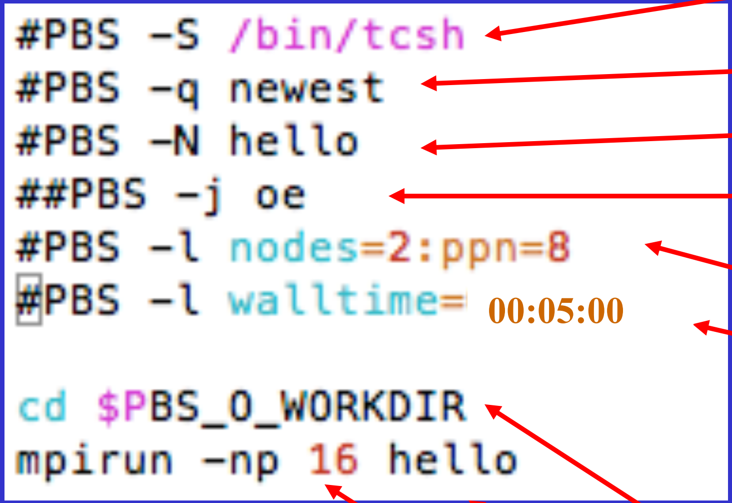
OpenMP: `gfortran –o omp_hello -fopenmp omp_hello.f90`

Grape: Basics

BATCH

STEP 3 (a): Edit and create PBS command file `hello_grape.cmd`

# commands are instructions to PBS



```
#PBS -S /bin/tcsh
#PBS -q newest
#PBS -N hello
##PBS -j oe
#PBS -l nodes=2:ppn=8
#PBS -l walltime= 00:05:00

cd $PBS_O_WORKDIR
mpirun -np 16 hello
```

This optional # command says which scripting shell you are using

Send job to partition/queue called "newest"

Jobname – shows up in squeue listing

Commented out: combine out and error files. `<jobname>.o<jobid>` and `<jobname>.e<jobid>` will be created

Use 2 nodes and 8 processors per node

Allow this much time hh:mm:ss

Shell commands to run job and do other things

Change directory to the one where the job was submitted

`mpirun` runs an MPI compiled executable `hello`

`np` is total number of MPI tasks = #nodes * #procs_per_node

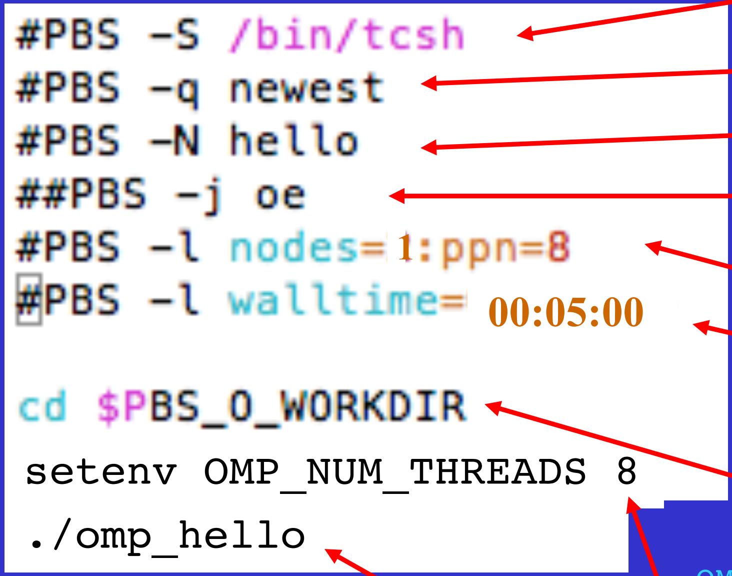
MPI

Grape: Basics

BATCH

STEP 3 (a): Edit and create PBS command file `hello_grape.cmd`

# commands are instructions to PBS



```
#PBS -S /bin/tcsh
#PBS -q newest
#PBS -N hello
##PBS -j oe
#PBS -l nodes=1:ppn=8
#PBS -l walltime= 00:05:00

cd $PBS_O_WORKDIR
setenv OMP_NUM_THREADS 8
./omp_hello
```

This # command says which scripting shell you are using

Send job to partition/queue called “newest”

Jobname – shows up in squeue listing

Commented out: combine out and error files. `<jobname>.o<jobid>` and `<jobname>.e<jobid>` will be created

Use 1 node (shared mem) and 8 processors per node

Allow this much time hh:mm:ss

Shell commands to run job and do other things

Change directory to the one where the job was submitted

`OMP_NUM_THREADS` is total number of tasks = #nodes * #procs_per_node.

Use “setenv” as above for csh; use “export OMP_NUM_THREADS=8” for bash

Run the OpenMP compiled executable `omp_hello`

OpenMP

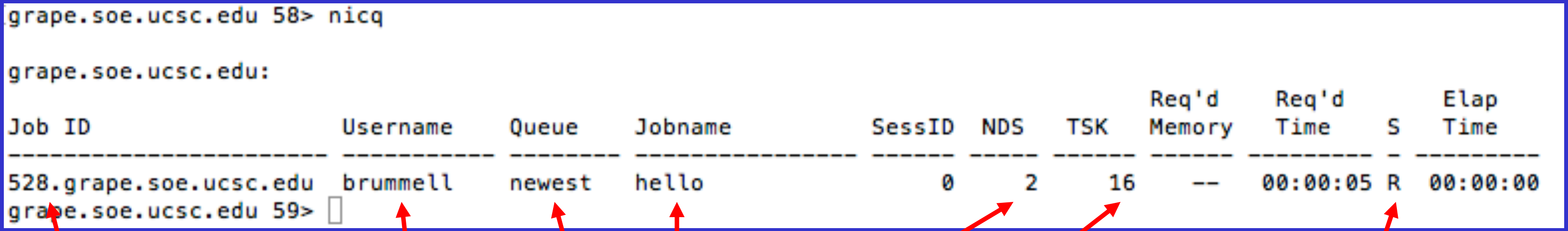
Grape: Basics

STEP 4: Submit job to the partition (queue)

```
qsub hello.cmd
```

STEP 5: Monitor the job

```
qstat –u brummell
```



```
grape.soe.ucsc.edu 58> nicq

grape.soe.ucsc.edu:
                                                                               Req'd    Req'd       Elap
Job ID                  Username    Queue    Jobname          SessID  NDS   TSK   Memory   Time    S   Time
----------------------- ----------- -------- ---------------- ------ ----- ------ ------ --------- - ---------
528.grape.soe.ucsc.edu  brummell    newest   hello                 0     2     16    --   00:00:05 R  00:00:00
grape.soe.ucsc.edu 59> 
```

Job ID:

Use to kill job if necessary

User

Queue/partition

Jobname

Nodes

Tasks (total # procs)

Job state codes:

PD = pending

R = running

CG = completing

CD = completed

Delete jobs: `qdel <jobid>`

Grape: Basics

INTERACTIVE

STEP 3(b): To run interactively, either:

1. Login to compute node and run: NOTE: you can only run up to 16 processes!

```
ssh compute-0-10
cd <working_directory>
```

MPI: `mpirun –np <nprocs> ./hello` (max np = #cores in node)

OpenMP:

```
setenv OMP_NUM_THREADS <nprocs>
./omp_hello
```

or `export OMP_NUM_THREADS=<nprocs>`

2. From master node, create an interactive job:

```
qsub –q newest –l nodes=2:ppn=8 –I
cd <working_directory> (can use cd $PBS_O_WORKDIR)
```

MPI: `mpirun –np 16 ./hello`

OpenMP:

```
setenv OMP_NUM_THREADS <nprocs>
./omp_hello
```

or `export OMP_NUM_THREADS=<nprocs>`

Grape vs Hummingbird/Lux

Note that they are very similar but slightly different!

The different batch job systems make for slightly different usage.

This is very common on moving between different supercomputers!

Homework 3

Run my “hello world” MPI and OpenMP executables on lux (in batch and interactive) and Hummingbird (batch only) on (a) 1 node with 4 processes using MPI, (b) 2 nodes with 4 processes per node using MPI, (c) 1 node 4 threads using OpenMP, (d) 1 node 8 threads using OpenMP. You will have to write SLURM scripts to do the batch jobs! (← key exercise!)

The executables are made for you already and are on both machines in `~brummell/AM250_Exercises/`   Copy these to your own directories.

```
hello_mpi_lux
hello_mpi_hb
hello_omp_lux
hello_omp_hb
```

Submit a tarfile to Canvas containing:

- ✓ the scripts that you write to do the batch jobs
- ✓ output from your jobs
  - screen shots of the commands and the output on the screen for interactive
  - the standard error and output files of the job for batch

Total # of tests = 3 methods (2 batch, 1 interactive) * 2 types (mpi, openmp) * 2 node/procs distributions (a,b or c,d) = 12 jobs (see next page for summary/list)

Homework 3

Summary of jobs:

Job#	Executable	Batch/Interactive	#nodes/procs
Job1:	hello_mpi_lux	batch	(a) 1 node, 4 procs total
Job2:	hello_mpi_lux	batch	(b) 2 nodes, 8 procs total
~~Job3:~~	~~hello_mpi_lux~~	~~interactive~~	~~(a) 1 node, 4 procs total~~
~~Job4:~~	~~hello_mpi_lux~~	~~interactive~~	~~(b) 2 nodes, 8 procs total~~
Job5:	hello_mpi_hb	batch	(a) 1 node, 4 procs total
Job6:	hello_mpi_hb	batch	(b) 2 nodes, 8 procs total
Job7:	hello_omp_lux	batch	(c) 1 node, 4 threads
Job8:	hello_omp_lux	batch	(d) 1 node, 8 threads
~~Job9:~~	~~hello_omp_lux~~	~~interactive~~	~~(c)) 1 node, 4 threads~~
~~Job10:~~	~~hello_omp_lux~~	~~interactive~~	~~(d)) 1 node, 8 threads~~
Job11:	hello_omp_hb	batch	(c)) 1 node, 4 threads
Job12:	hello_omp_hb	batch	(d)) 1 node, 8 threads

The End

Older slides

Homework 3

Try running my “hello world” MPI and OpenMP executables on grape (in batch and interactive) and Hummingbird (batch only) on (a) 1 node with 4 processes for MPI, (b) 2 nodes with 4 processes per node for MPI, (c) 1 node 4 threads for OpenMP, (d) 1 node 8 threads for OpenMP

Executables are made for you already and are on HB:
`~brummell/Classes/AMS250/Exercises`

```
    hello_mpi_grape
    hello_mpi_hb
    hello_omp_grape
    hello_omp_hb
```

You will have to write PBS/SLURM scripts to do the batch jobs!

Submit to Google classroom:

- ✓ the scripts you write for the batch jobs
- ✓ output from your jobs (as screen shots of the commands and the output on the screen for interactive, and the standard error and output files of the job for batch).
- ✓ Total tests = 3 methods (2 batch, 1 interactive) * 2 types (mpi, openmp) * 2 node/procs distributions (a,b or c,d) = 12 jobs (see next page for summary/list)

Homework 3

Summary of jobs:

Job#	Executable	Batch/Interactive	#nodes/procs
Job1:	`hello_mpi_grape`	batch	(a)
Job2:	`hello_mpi_grape`	batch	(b)
Job3:	`hello_mpi_grape`	interactive	(a)
Job4:	`hello_mpi_grape`	interactive	(b)
Job5:	`hello_mpi_hb`	batch	(a)
Job6:	`hello_mpi_hb`	batch	(b)
Job7:	`hello_omp_grape`	batch	(c)
Job8:	`hello_omp_grape`	batch	(d)
Job9:	`hello_omp_grape`	interactive	(c)
Job10:	`hello_omp_grape`	interactive	(d)
Job11:	`hello_omp_hb`	batch	(c)
Job12:	`hello_omp_hb`	batch	(d)