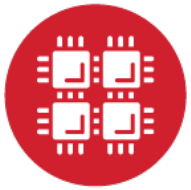


Ohio Supercomputer Center

An OH·TECH Consortium Member

Efficient Use of OSC Resources: Strategies for Bundling Jobs

Kate Cahill
Education & Training Specialist

Previous Presentations

Week 1

- What is OSC?
- HPC Concepts
- Hardware Overview
- Data Storage Systems
- Batch Processing
- Accessing Available Software
- OnDemand Web Portal Demonstration

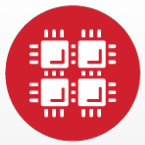
Week 2

- Jobs suited to OSC resources
- Condos at OSC
- How to get an allocation
- Other HPC resources to consider

All presentation slides will be available at CCAPP condo page:
https://www.osc.edu/supercomputing/computing/ruby/ccapp_condo

Overview

- Batch System Overview
- 2 Strategies for Grouping Jobs
  - Bash Loops
  - Parallel Command Processor

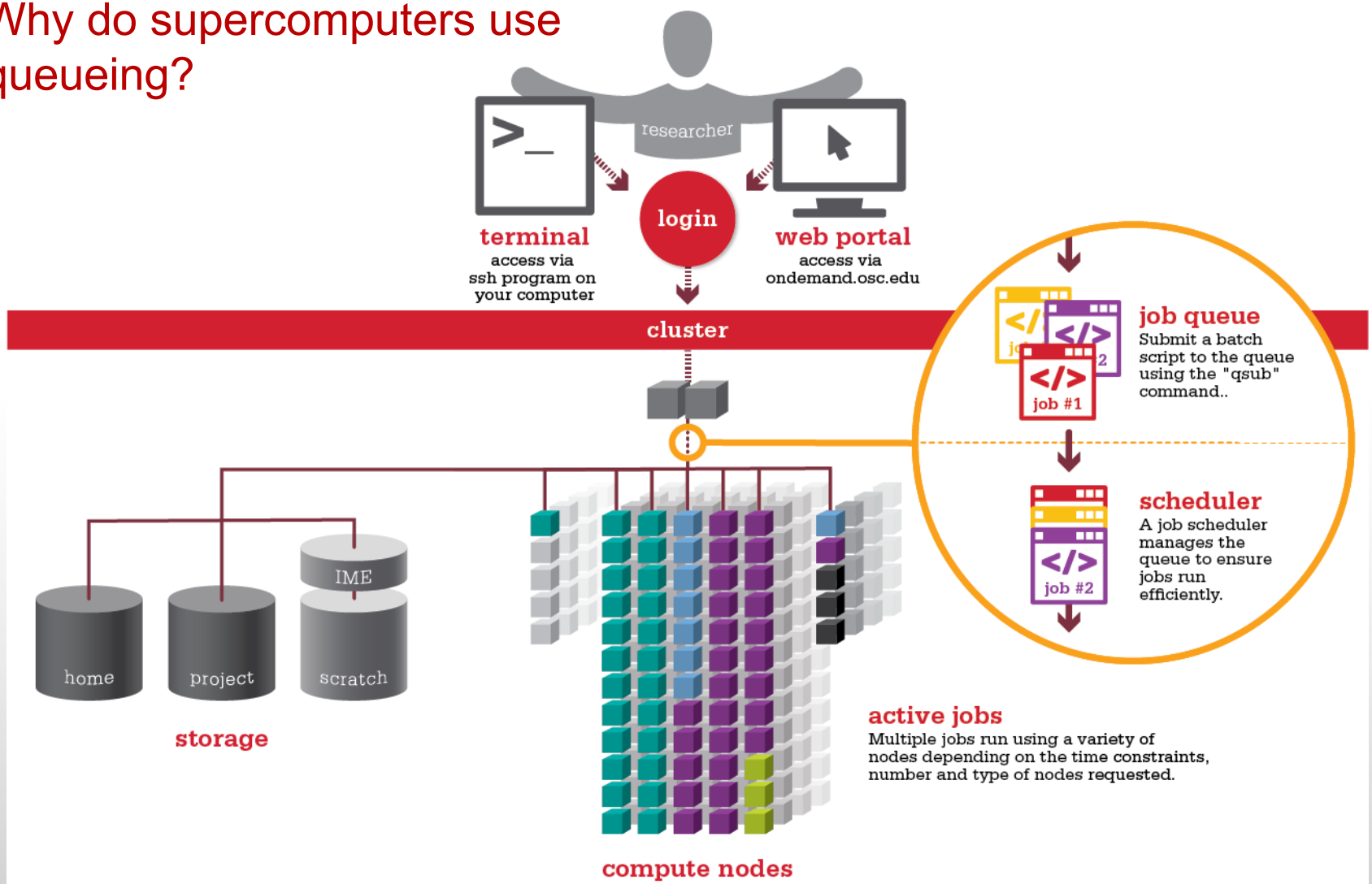


Ohio Supercomputer Center

An OH·TECH Consortium Member

Batch Processing

Why do supercomputers use queueing?



Batch System at OSC

- Compute nodes are allocated through the batch system
  - PBS – Portable Batch System
  - Torque – resource manager
  - Moab – scheduler
- Documentation at

  www.osc.edu/supercomputing/batch-processing-at-osc

Idea Behind Batch Processing

- Whatever you would normally type at the command prompt goes into your batch script
- Output that would normally go to the screen goes into a log file (or files)
- The system runs your job when resources become available
- Very efficient in terms of resource utilization
- Ruby uses whole node scheduling

Sample Batch Script

```
#PBS -N serial_fluent
#PBS -l walltime=1:00:00
#PBS -l nodes=1:ppn=28
#PBS -j oe
#PBS -l software=fluent+1
# Set up the FLUENT environment
module load fluent
# Move to directory job was submitted from
cd $PBS_O_WORKDIR
# Copy input files to compute node
cp run.input $TMPDIR
cd $TMPDIR
# Run fluent and copy results back to home
fluent 3d -g < run.input
cp 'results*' $PBS_O_WORKDIR
```

Job setup information for PBS

This is a comment

Commands to be run

Put all this into a text file!

Submitting a Job and Checking Status

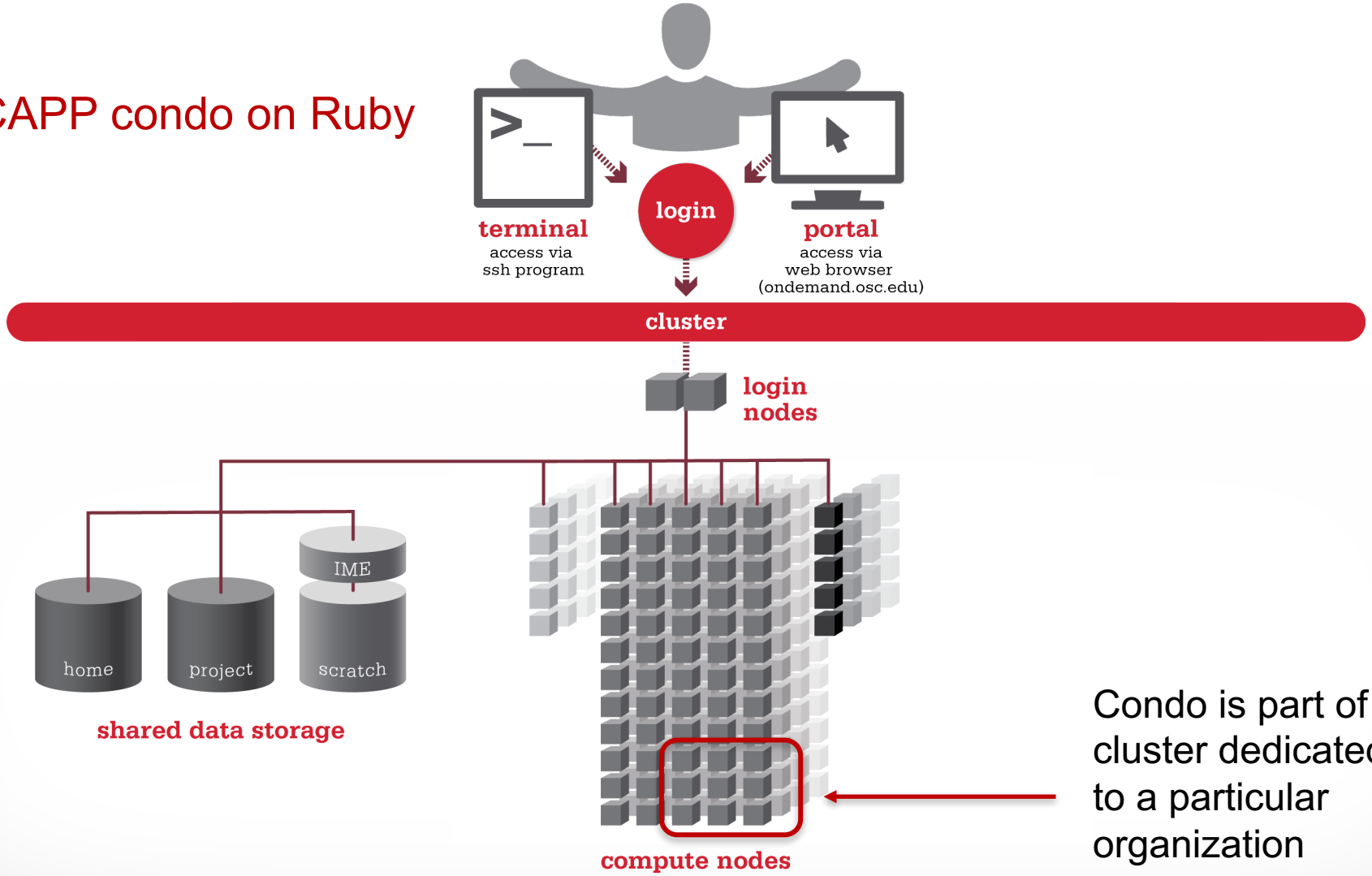
- Command to submit a job
  - **qsub *script_file***
- Response from PBS (example)
  - `123456.oak-batch.osc.edu or 123456`
- Show status of batch jobs
  - **qstat -a *jobid***
  - **qstat –u *username***
  - **qstat -f *jobid***

List of Batch commands on osc.edu

Job Output

- Screen output ends up in file ***job_name.ojobid***
  - Copied to your working directory when job ends
  - Example: **`testjob.o1234567`**
- To see screen output while job is running
  - **`qpeek`** ***`jobid`***
  - Example: **`qpeek 1234567`**
- To delete a job
  - **`qdel`** ***`jobid`***

CCAPP condo on Ruby



Condo is part of cluster dedicated to a particular organization

Parallel Computing

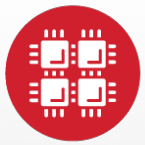
- Each processor is fast, but real speed comes from using multiple processors
- Multithreading
  - Use multiple cores on a single node
  - Shared memory
- Message passing (MPI)
  - Use one or multiple nodes
  - Distributed memory
  - Some good tutorials on MPI: http://www.mcs.anl.gov/research/projects/mpi/tutorial/mpibasics/index.htm
  - http://hpcuniversity.org/trainingMaterials/182/

Python Matrix example with Lapack

https://www.osc.edu/~kmanalo/multithreadedsubmission or search 'multithreaded' at osc.edu

- Python script using multithreading
- A*x=b
- A random matrix is generated for square matrix A, also for solution vector b
- Solve for x
- Several jobs will be run on multiple array sizes

- All input files are accessible here:
  `/users/appl/kcahill/workshop/lapack`



Ohio Supercomputer Center

An OH·TECH Consortium Member

Strategy 1: Bash Loop

Bash loop

```
for f in file1 file2 file3 file5
do
echo "Processing $f" # do something on $f
done
```

```
for i in $(seq 1 10000)
do
```

Advantages

- Easy to use, flexible

Disadvantages

- Less control over how jobs run, sequence, timing

Bash Loop Job Script - Set up

```
#!/bin/bash#
PBS -N bash_loop
#PBS -l walltime=00:10:00
#PBS -l nodes=1:ppn=20
#PBS -j oe   # Append stderr to stdout
#PBS -A PZS0557 #Change for your project
```

Bash Loop Job Script

```
cd $PBS_O_WORKDIR
cp example_lapack.py $TMPDIR
cd $TMPDIR
module load python/2.7.latest
# alternatively: for size in 4 2 3 1; do
for size in $(seq 1000 1000 10000); do
echo
echo python example_lapack.py $size
python example_lapack.py $size
Done
cp *.* $PBS_O_WORKDIR
exit
```

Bash loop gives input to job;
The calculation is run 10 times

Submit job **`qsub bash_loop.sh`**

OnDemand Queue view – Active Jobs

Running bash_loop
1295804

Cluster	Ruby
PBS Id	1295804
Job Name	bash_loop
User	kcahill
Account	PZS0557
Walltime	00:10:00
Walltime Used	
Node Count	1
PPN	20
Total CPUs	20
CPU Time	0
Memory	0 b
Virtual Memory	0 b

OnDemand Queue view – Active Jobs

Bash loop output *bash_loop.sh.o${PBS_JOBID}*

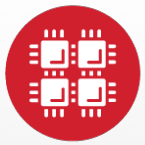
```
python example_lapack.py 1000
Solved a matrix of size: 1000 using 20 threads.
Relative Error: 1.18480195883e-12
--- Random Matrix Generation Time: 0.09836602211 seconds ---
---                 Solution Time: 0.344985961914 seconds ---

python example_lapack.py 2000
Solved a matrix of size: 2000 using 20 threads.
Relative Error: 1.10688620104e-11
--- Random Matrix Generation Time: 0.139647960663 seconds ---
---                 Solution Time: 0.203233957291 seconds ---

python example_lapack.py 3000
Solved a matrix of size: 3000 using 20 threads.
Relative Error: 5.85581800884e-12
--- Random Matrix Generation Time: 0.229151964188 seconds ---
---                 Solution Time: 0.387782096863 seconds ---

python example_lapack.py 4000
Solved a matrix of size: 4000 using 20 threads.
Relative Error: 1.72465518722e-11
--- Random Matrix Generation Time: 0.361100912094 seconds ---
---                 Solution Time: 0.583263874054 seconds ---
```

Bash loop output *bash_loop.sh.o${PBS_JOBID}*

```
python example_lapack.py 5000
Solved a matrix of size: 5000 using 20 threads.
Relative Error: 7.50426653623e-11
--- Random Matrix Generation Time: 0.479506969452 seconds ---
---                 Solution Time: 0.75616312027 seconds ---

python example_lapack.py 6000
Solved a matrix of size: 6000 using 20 threads.
Relative Error: 1.26792777047e-11
--- Random Matrix Generation Time: 0.702255964279 seconds ---
---                 Solution Time: 1.13342285156 seconds ---

python example_lapack.py 7000
Solved a matrix of size: 7000 using 20 threads.
Relative Error: 3.70321066796e-10
--- Random Matrix Generation Time: 0.721174955368 seconds ---
---                 Solution Time: 1.49235486984 seconds ---

python example_lapack.py 8000
Solved a matrix of size: 8000 using 20 threads.
Relative Error: 4.96462623023e-11
--- Random Matrix Generation Time: 0.925333023071 seconds ---
---                 Solution Time: 2.38715410233 seconds ---

python example_lapack.py 9000
Solved a matrix of size: 9000 using 20 threads.
Relative Error: 9.15271498894e-11
--- Random Matrix Generation Time: 1.15859508514 seconds ---
---                 Solution Time: 3.08035898209 seconds ---
```

Bash loop output *bash_loop.sh.o${PBS_JOBID}*

```
python example_lapack.py 10000
Solved a matrix of size: 10000 using 20 threads.
Relative Error: 1.19903571601e-10
--- Random Matrix Generation Time: 1.38230586052 seconds ---
---                 Solution Time: 3.50143814087 seconds ---


------------------------
Resources requested:
nodes=1:ppn=20
------------------------
Resources used:
cput=00:03:43
walltime=00:00:28
mem=1.776 GB
vmem=0.260 GB
------------------------
Resource units charged (estimate):
0.015 RUs
------------------------
```



Ohio Supercomputer Center

An OH·TECH Consortium Member

Strategy 2: OSC tool – Parallel Command Processor

Parallel Command Processor

- OSC tool (not used elsewhere)
- Designed for running multiple serial jobs in a parallel environment (N processors can run N-1 jobs simultaneously)
- Can read a list of jobs from separate file or a loop or a list in the job script

- See examples here: https://www.osc.edu/resources/available_software/software_list/parallel_command_processor
  - (or search osc.edu pcp)

PCP example script set up

```
#PBS -l nodes=13:ppn=4 #PBS -l walltime=1:00:00 #PBS -S
/bin/bash #PBS -N blast-PCP #PBS -j oe
date
module load biosoftw
module load blast
set -x
cd $PBS_O_WORKDIR
pbsdcp query/query.fsa.* $TMPDIR
pbsdcp db/rice.* $TMPDIR
cd $TMPDIR
```

PCP example script loop

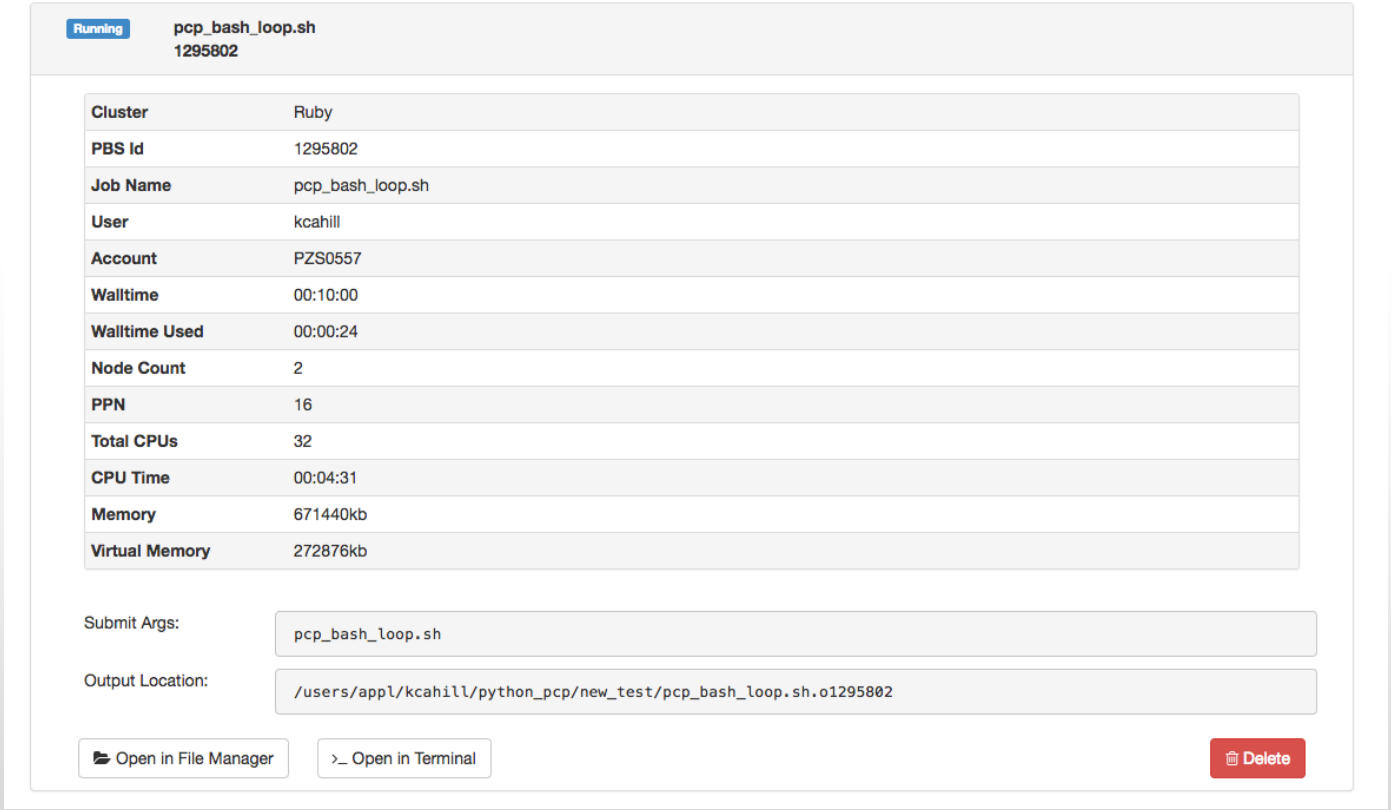
```
for i in $(seq 1 49)
do
   cmd="blastall -p blastn -d rice -i query.fsa.$i -o
out.$i"
   echo ${cmd} >> runblast
done
mpiexec parallel-command-processor runblast
mkdir $PBS_O_WORKDIR/output
pbsdcp -g out.* $PBS_O_WORKDIR/output
exit
```

Lapack example (uses bash loop)

```
procs=32
ppn=16
max_nodes=$((procs/ppn))
count=0
base=0
array=($(seq 1000 1000 10000))
while [ $base -lt ${#array[@]} ]; do
    for task in $(seq 1 $procs); do
        case=${array[$((task / ppn + base))]}
        command="python example_lapack.py $case"
        echo "if [ $((task%$ppn)) -eq 1 ] ; then $command;
fi"
    done | mpiexec parallel-command-processor
    base=$((base+max_nodes))
Done
```

Submit job **`qsub pcp_bash_loop.sh`**

OnDemand Queue View – Active Jobs



OnDemand Queue View – Active Jobs

Lapack example output

```
Solved a matrix of size: 1000 using 16 threads.
Relative Error: 3.0462875541e-12
--- Random Matrix Generation Time: 0.0418329238892 seconds ---
---                 Solution Time: 0.670638084412 seconds ---
Solved a matrix of size: 2000 using 16 threads.
Relative Error: 3.65857426525e-12
--- Random Matrix Generation Time: 0.0640590190887 seconds ---
---                 Solution Time: 1.81462192535 seconds ---
Executed 33 commands in 4.349066 seconds on 31 minions
Solved a matrix of size: 3000 using 16 threads.
Relative Error: 9.91562029658e-12
--- Random Matrix Generation Time: 0.162482023239 seconds ---
---                 Solution Time: 2.11311793327 seconds ---
Solved a matrix of size: 4000 using 16 threads.
Relative Error: 1.2483129645e-11
--- Random Matrix Generation Time: 0.287737131119 seconds ---
---                 Solution Time: 3.14574193954 seconds ---
Executed 33 commands in 3.846388 seconds on 31 minions
Solved a matrix of size: 5000 using 16 threads.
Relative Error: 1.4514650146e-11
--- Random Matrix Generation Time: 0.445693016052 seconds ---
---                 Solution Time: 5.47839593887 seconds ---
Solved a matrix of size: 6000 using 16 threads.
Relative Error: 2.62104316572e-10
--- Random Matrix Generation Time: 0.619028091431 seconds ---
---                 Solution Time: 8.79232096672 seconds ---
```

Lapack example output

```
Solved a matrix of size: 7000 using 16 threads.
Relative Error: 3.93675746254e-11
--- Random Matrix Generation Time: 0.818657875061 seconds ---
---                 Solution Time: 12.9895291328 seconds ---
Solved a matrix of size: 8000 using 16 threads.
Relative Error: 5.61979357726e-11
--- Random Matrix Generation Time: 1.04007816315 seconds ---
---                 Solution Time: 19.1941049099 seconds ---
Executed 33 commands in 20.514196 seconds on 31 minions
Solved a matrix of size: 9000 using 16 threads.
Relative Error: 8.9107077528e-11
--- Random Matrix Generation Time: 1.31298208237 seconds ---
---                 Solution Time: 26.6784529686 seconds ---
Solved a matrix of size: 10000 using 16 threads.
Relative Error: 3.18338344307e-11
--- Random Matrix Generation Time: 1.61283707619 seconds ---
---                 Solution Time: 36.3206989765 seconds ---
Executed 33 commands in 38.219204 seconds on 31 minions
```

Lapack example output

```
-----------------------
Resources requested:
nodes=2:ppn=16
-----------------------
Resources used:
cput=00:41:46
walltime=00:01:30
mem=3.520 GB
vmem=11.452 GB
-----------------------
Resource units charged (estimate):
0.080 RUs
-----------------------
```

Important Points to Remember

- Group jobs using bash loop or pcp to run multiple jobs on 1 node
- The CCAPP condo on Ruby is 21 nodes (420 total processors)
- Ruby uses whole node scheduling (20 processors per node)
- Users on the condo cannot access other OSC clusters
- If your project requires lots of jobs or many large jobs, consider getting a separate allocation

See these references:
https://www.osc.edu/~kmanalo/multithreadedsubmission
https://www.osc.edu/resources/available_software/software_list/parallel_command_processor
https://www.osc.edu/supercomputing/batch-processing-at-osc

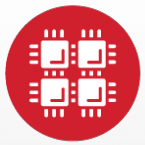
Upcoming Training Events

- Big Data workshop **October 26** 1-4pm at OSC
- XSEDE OpenACC workshop, live webinar at OSC: **November 7** 11am-5pm
- XSEDE Big Data Workshop, live webinar at OSC: **December 5 & 6**, 11am-5pm

You can sign up for one-on-one consultations
go.osu.edu/rc-osc

Questions?

Kate Cahill
Education & Training Specialist
Ohio Supercomputer Center
kcahill@osc.edu



Ohio Supercomputer Center

An OH·TECH Consortium Member

Strategy 3: Task Array

Task array

- Another type of grouping, it uses the scheduler to run multiple jobs
- It is less useful for grouping jobs on Ruby unless each job should use one whole node

Task array example script

```
#!/bin/bash#PBS -l walltime=00:10:00#PBS -l
nodes=1:ppn=20# Append stderr to stdout#PBS -j oe#PBS -A
PZS0557
module load python/2.7.latest
cd $PBS_O_WORKDIR
#array=(1 3 4 50000)
array=($(seq 1000 1000 10000))
if [ -z ${PBS_ARRAYID+x} ] ; then # read
http://stackoverflow.com/a/13864829/992834
   echo PBS_ARRAYID is not set
   echo submit this script using "qsub -t 1-10 $0”
else
   echo PBS_JOBID: $PBS_JOBID
   echo python example_lapack.py ${array[$PBS_ARRAYID]}
   python example_lapack.py ${array[$PBS_ARRAYID]}
fi
```