Why we are here!

• You have a computational task of some sort

• Most people who come to CARC fall into these categories
  • ... for a publication you are working on
  • ... an upcoming class assignment
  • ... analysis of data for a government agency

• The common feature is that they require more compute/memory/storage resources than is commonly available.
Why we are here!

• To use the resources at CARC effectively there are things you have to learn! Some are commonplace but some are very specific to HPC and CARC.

By the end of the day you will know how to:
• Request time compute time on the clusters (PBS scripting)
• Interact with contained environments (Anaconda, Modules, Singularity)
• Get output from your compute jobs
• Use storage appropriately (scratch vs user storage)
• Write scripts to run embarrassingly parallel tasks
• Write and run a simple MPI program for tightly coupled tasks

• Ultimately get a huge increase in the computing power you can apply.
Introduction to HPC at the UNM Center for Advanced Research Computing

Matthew Fricke (mfricke@unm.edu)
Research Assistant Professor
http://unm.edu/~mfricke

Please send corrections to me at mfricke@unm.edu.
Version 1.0.
What is High Performance Computing?

Scaling up

- NVIDIA DGX-2H ($400,000 each, 81k CUDA cores, 10240 tensor cores)

Scaling out

- Stampede 2
  https://www.tacc.utexas.edu/systems/stampede2
- $30,000,000, 285,000 CPUs
Why it Matters to You

• Grant and publication reviewers know about these systems so there are no excuses for small sample sizes.

• Machine Learning is showing up everywhere from cosmology to firefighting. Machine Learning requires enormous resources to process huge datasets.
The Center for Advanced Research Computing’s Mission

The UNM Center for Advanced Research Computing is the hub of computational research at UNM and one of the largest computing centers in the State of New Mexico. It is an interdisciplinary community that uses computational resources to create new research insights. The goal is to lead and grow the computational research community at UNM.

CARC provides not just the computing resources but also the expertise and support to help the university’s researchers. This service is available to faculty, staff, and student researchers free of charge through support from the UNM Office of the Vice President for Research.
http://carc.unm.edu
Big Data and Machine Learning

• Machine learning needs lots of everything.

• The current revolution in convolutional neural networks (Think Google, Self-driving cars, etc) is due to algorithms rooted in the 1960s being given huge training sets.

• Most of machine learning comes down to floating point matrix and vector operations. GPUs excel at those operations and are orders of magnitude fast at them than CPUs.

• Xena has dual Nvidia Tesla K40M GPUs for this purpose.
Biology

- Computational biology memory usage increases with input sizes. Rapid genotyping tools generate sequences faster and faster.
- Hundreds of GB of RAM are becoming a normal requirement to complete these calculations.
- The Taos cluster is dedicated to computational biology and has 440 CPUs and 300 GB per node.
- Xena has 3 TB RAM nodes.

- Pandemic flu modelling
- Tuberculosis antibiotic resistance
- Pacific island bird genetics
- Intra-species viral spread
- NM Tree species mapping from LASER scans
Physics

• Wheeler is a general purpose scale-out machine used by biophysicists, cosmologists, and many others.

• Gibbs is primarily used by computational chemists.

Molecular simulation of new photovoltaic materials

Long Wavelength Array Radio Telescope Data Processing on Wheeler
When users need a lot of network bandwidth or storage

- Lots of data comes in the form of large images
- Largest online pathology database (15k people, 15k image each)
- MRI and FMRI image databases

- Specialized Network Access
  - Mapping the Great Firewall
2016101  Finding approximate symmetries in graphs
2016100  Simulation of impact on armor plate using Velodyne
2016099  Cyber-infrastructure Performance Modeling
2016098  Effective refinement of protein structure
2016097  LANL_HIGRAD
2016096  Coarse-grained modeling of biomolecules
2016095  Resilient Composites
2016093  Computational Investigation of Ru Photochromes

2016092  Reinforcement learning neuropathologies underlying psychiatric sequelae in Traumatic Brain Injury
2016086  Polyurethane Foam Modeling
2016084  An Interval Multi-level Monte Carlo Method for Reliability Analysis of Imprecise Probabilistic Systems
2016083  Modeling Immune System Cell Search Processes
2016081  RNA transcriptome sequencing of T-cells exposed to Uranium and Arsenic
2016080  Predicting Progression to Alzheimer’s Disease
2016079  Relating plant traits to biomass dynamics in New Mexico aridlands
2016078  Statistical methods for investigating large scale gene environment interaction
2016077  TB Genomic Analysis
2016076  LiDAR-based tree identification in Northern, New Mexico
<table>
<thead>
<tr>
<th>Date</th>
<th>Title</th>
<th>Authors</th>
</tr>
</thead>
<tbody>
<tr>
<td>2016074</td>
<td>Implicit Monte Carlo Simulation of Thermal Radiation Transport</td>
<td>Prinja, Anil K.</td>
</tr>
<tr>
<td>2016073</td>
<td>Implicit Monte Carlo Simulation of Transport Phenomena</td>
<td>Prinja, Anil K.</td>
</tr>
<tr>
<td>2016072</td>
<td>simulations for LEGEND neutrino-less double beta decay experiment</td>
<td>Gold, Michael S</td>
</tr>
<tr>
<td>2016070</td>
<td>Transcriptomic Analysis of Drosophila Neuroblasts</td>
<td>Johnston, Christopher</td>
</tr>
<tr>
<td>2016068</td>
<td>Simulation of quantum many-body systems</td>
<td>Miyake, Akimasa</td>
</tr>
<tr>
<td>2016066</td>
<td>microRNA and epigenetic control on gene expression</td>
<td>Liang, Fu-Sen</td>
</tr>
<tr>
<td>2016063</td>
<td>Genomic Data Analysis</td>
<td>Guo, Yan</td>
</tr>
<tr>
<td>2016062</td>
<td>Simulation of terawatt x-ray free electron lasers</td>
<td>Freund, Henry</td>
</tr>
<tr>
<td>2016061</td>
<td>Evolutionary convergence in mainland and island lizards</td>
<td>Poe, Steve</td>
</tr>
<tr>
<td>2016059</td>
<td>Curcumin modulation of amyloid-beta peptide interaction with lipid membranes</td>
<td>Chi, Eva Y</td>
</tr>
<tr>
<td>2016058</td>
<td>Atlantic salmon gill microbiome</td>
<td>Salinas, Irene</td>
</tr>
<tr>
<td>2016057</td>
<td>High-fidelity Model for Wind Farms</td>
<td>Lee, Sang</td>
</tr>
<tr>
<td>2016056</td>
<td>Monte Carlo Simulation of Stochastic Multiplying Systems</td>
<td>O'Rourke, Patrick F</td>
</tr>
<tr>
<td>2016053</td>
<td>Nanophotonic metasurfaces</td>
<td>Acosta, Victor</td>
</tr>
<tr>
<td>2016052</td>
<td>Genomic analyses of cellular quiescence</td>
<td>Osley, Mary Ann</td>
</tr>
<tr>
<td>2016051</td>
<td>Performance Optimization of LANL Multi-Physics Applications</td>
<td>Bridges, Patrick G</td>
</tr>
<tr>
<td>Key</td>
<td>Title</td>
<td>Author</td>
</tr>
<tr>
<td>--------</td>
<td>----------------------------------------------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>2016050</td>
<td>Investigating the impact of metal contaminants in environmental microbial populations</td>
<td>Cerrato, Jose M.</td>
</tr>
<tr>
<td>2016049</td>
<td>COMPUTATIONAL INVESTIGATION OF PARAMETER SPACE APPLICABLE TO NUCLEAR FACILITY SAFEGUARDS USING THE UNM CENTER FOR ADVANCED RESEARCH COMPUTING</td>
<td>Arthur, Edward D</td>
</tr>
<tr>
<td>2016047</td>
<td>Multiscale Mechanistic Model to Study Nanotherapy Delivery in Tumors</td>
<td>Bearer, Elaine L</td>
</tr>
<tr>
<td>2016046</td>
<td>Designing of Fuel Performance Experiments to be Performed Using the Annular Research Reactor (ACRR)</td>
<td>Lee, Youho</td>
</tr>
<tr>
<td>2016045</td>
<td>Model building the HELP physical unclonable function</td>
<td>Plusquellic, Jim F</td>
</tr>
<tr>
<td>2016044</td>
<td>Genomic Comparisons of Multipartite Symbiosis: Understanding the metabolic basis of parasitism</td>
<td>Kamel, Bishoy S</td>
</tr>
<tr>
<td>2016042</td>
<td>QIIME analyses of microbial sequences acquired from saliva samples</td>
<td>Carroll-Portillo, Amanda</td>
</tr>
<tr>
<td>2016041</td>
<td>Deterministic and Bayesian Seismic source inversion involves</td>
<td>Appelo, Daniel EA</td>
</tr>
<tr>
<td>2016040</td>
<td>Consequence-based Impact Rating using ANSYS</td>
<td>Moreu, Fernando</td>
</tr>
<tr>
<td>2016039</td>
<td>Machine-Learning Design of Novel Photovoltaic Materials Based on Conceptual Understanding of Electronic Structure Calculations</td>
<td>Talipov, Marat R</td>
</tr>
<tr>
<td>2016036</td>
<td>Melt migration in continental interiors</td>
<td>Roy, Mousumi</td>
</tr>
<tr>
<td>2016035</td>
<td>Monte Carlo Simulation for Weak Neutron Sources</td>
<td>Goss, Vanessa</td>
</tr>
<tr>
<td>2016034</td>
<td>Programmable Nanowalkers: Models and Simulations</td>
<td>Stefanovic, Darko</td>
</tr>
<tr>
<td>2016033</td>
<td>Deep Learning and Differential Geometry</td>
<td>Huang, Hongnian</td>
</tr>
<tr>
<td>2016032</td>
<td>Analyzing Neuronal Coordination during a task of Behavioral Flexibility in a Model of Fetal Alcohol Spectrum Disorder</td>
<td>Brigman, Jonathan L</td>
</tr>
<tr>
<td>2016031</td>
<td>“Mountain Lions on the Edge: Integrating Conservation into Urban Planning through Predictive Modeling”</td>
<td>Milne, Bruce T</td>
</tr>
<tr>
<td>2016029</td>
<td>Discrete Element Modeling of Drilled Shafts in Granular Materials</td>
<td>Ng, Tang-Tat</td>
</tr>
<tr>
<td>2016028</td>
<td>Exploration of optical rogue wave phenomena in dielectrics as a function of the intrinsic randomness or disorder</td>
<td>Mafi, Arash</td>
</tr>
<tr>
<td>2016026</td>
<td>Differential Splicing by Sex in DNA Repair Genes</td>
<td>Berwick, Marianne</td>
</tr>
<tr>
<td>2016021</td>
<td>Atlantic salmon microbiome</td>
<td>Salinas, Irene</td>
</tr>
<tr>
<td>2016019</td>
<td>Small Area Population Estimates</td>
<td>Rhatigan, Robert</td>
</tr>
<tr>
<td>2016018</td>
<td>Differential Gene Expression in Cancer</td>
<td>Trujillo, Kristina</td>
</tr>
</tbody>
</table>
Investigating the impact of metal contaminants in environmental microbial populations
Cerrato, Jose M.

Computational investigation of parameter space applicable to nuclear facility safeguards using the UNM Center for Advanced Research Computing
Arthur, Edward D.

Multiscale mechanistic model to study nanotherapy delivery in tumors
Bearer, Elaine L.

Designing of fuel performance experiments to be performed using the annular research reactor (ACRR)
Lee, Youho

Model building the HELP physical unclonable function
Plusquellic, Jim F.

Genomic comparisons of multipartite symbiosis: understanding the metabolic basis of parasitism
Kamel, Bishoy S.

QIIME analyses of microbial sequences acquired from saliva samples
Carroll-Portillo, Amanda

Deterministic and Bayesian seismic source inversion involves
Appelo, Daniel EA

Consequence-based impact rating using ANSYS
Moreu, Fernando

Machine-learning design of novel photovoltaic materials based on conceptual understanding of electronic structure calculations
Talipov, Marat R

Melt migration in continental interiors
Roy, Mousumi

Monte Carlo simulation for weak neutron sources
Goss, Vanessa

Programmable nanowalkers: models and simulations
Stefanovic, Darko

Deep learning and differential geometry
Huang, Hongnian

Analyzing neuronal coordination during a task of behavioral flexibility in a model of fetal alcohol spectrum disorder
Brigman, Jonathan L

“Mountain lions on the edge: integrating conservation into urban planning through predictive modeling”
Milne, Bruce T

Discrete element modeling of drilled shafts in granular materials
Ng, Tang-Tat

Exploration of optical rogue wave phenomena in dielectrics as a function of the intrinsic randomness or disorder
Mafi, Arash

Differential splicing by sex in DNA repair genes
Berwick, Marianne

Atlantic salmon microbiome
Salinas, Irene

Small area population estimates
Rhatigan, Robert

Differential gene expression in cancer
Trujillo, Kristina

And a couple of hundred more.
(TLDR)
Basics of HPC Systems

• Parallelism within CPUs
• Parallelism within GPUs
• Parallelism of CPUs and GPUs
• Parallelism of whole computers
Some useful Linux commands

- `ssh username@wheeler.alliance.unm.edu`
- `scp myfile.txt username@wheeler.unm.edu:~`
- `scp username@wheeler.unm.edu:~/myfile.txt`
- `CyberDuck, or WinScp`
- `CARC supports secure file transfer (SFTP), so choose that protocol if you use a graphical transfer program.`
- `Rsync`
- `ls -lah`
- `find . -name "*.txt"`
Compartmentalization

- A challenge running large multi-user systems is supporting all the different software required for hundreds of projects.

- Compartmentalization keeps user software isolated.

- We use 3 general methods at CARC: Environment Modules, Anaconda, and Singularity (these are the current standards so what you learn here will translate to other HPC centers)
Environment Modules

• Open a secure shell on wheeler is you haven’t already

$ ssh username@wheeler.alliance.unm.edu

• Display your environment variables
$ env

• All the modules do is set the environment variables for different software
Environment Modules

- Let’s load the R module so we can use it.

```bash
$ module list

$ module load module load r-3.5.0-intel-18.0.2-python2-mkl-r6lx6yy
(use tab complete !!)

$ R
```
Environment Modules

- Let’s load the R module so we can use it.

$ module list (Again)

Will take a while!

$ module avail (to show all available modules)
$ module spider <software name> (to find a software package)

Conda

- CARC staff have to install the software and create the environment modules you just saw.
- Anaconda provides an environment manager called *conda* that allows you to install the software you need into your home directory.
- Conda works with python, perl, R, and theoretically any language
Conda – Hands On

• Let’s setup a local install of numpy

$ module load anaconda
$ conda –V
$ conda create -n numpy_py3 python=3.4 numpy

Wait a while – introduce yourselves to your neighbor... believe there is a reason we are doing this...
Coda – Hands On

• Let’s setup a local install of numpy

$ module load anaconda

$ conda –V

$ conda create -n numpy_py3 python=3.4 numpy

Now we have defined a conda environment called numpy_py3 and installed numpy in our home directories. At CARC we pay for the commercial intel version of mkl (math kernel library). Installing numpy now means you will get a performance boost for later installs. Sometimes other packages will select the non-intel mkl so lets get the right version.
Conda – Hands On

• Let’s setup a local install of numpy
  $ module load anaconda
  $ conda -V
  $ conda create -n numpy_py3 python=3.4 numpy

• Now we can load the environment
  $ source activate numpy_py3
Conda – Hands On

• Let’s see if numpy is available
• Create a file called test_numpy.py with the following contents:

```python
try:
    import numpy
    print("imported numpy")
except ImportError:
    print("numpy is not installed")
```
Conda – Hands On

• Let’s see if numpy is available
• Create a file called test_numpy.py with the following contents:

```python
try:
    import numpy
    print("imported numpy")
except ImportError:
    print("numpy is not installed")
```

$ python test_numpy.py
Conda – Hands On

Conda just installs the software under ~/.conda

- $ conda list

$ source deactivate numpy_py3
Docker and Singularity

- Singularity allows you to load converted Docker images on HPC systems.
- Docker is not secure so singularity locks down access to the host machine.
- Docker containers allow you to configure a whole virtual operating system environment (e.g. your software needs Ubuntu but Wheeler runs CentOS).
- Convert your docker image to singularity and you can run the container. We will post the singularity QuickByte soon.
15 mins Break

- Snacks and Drinks in the Seminar room
Submitting Jobs and Embarrassingly Parallel Problems
Hands On

• Write a very short program that takes as its first argument a file path
• Does something with the contents, can be as simple as printing the file’s contents to standard out.
• If you are not a programmer you can use the python program we have provided (it reads matrix sizes from the input file, creates a random matrix, and outputs the inverted matrix.)
Multiuser Systems and Batch Scheduling

• TORQUE (PBS)
• MAUI
Interactive Mode

• ONLY FOR DEBUGGING!

$ ssh <username>@galles.alliance.unm.edu
$ qsub -l -l nodes=2:ppn=2 -q Core2

Capital I for interactive

Request 2 nodes and 2 cores per node

Lower case l followed by options

Submit the job to the Core2 queue
PBS Variables Provide Information

• In interactive mode try:

  $ echo $PBS_O_WORKDIR
  
  $ echo $PBS_NODEFILE
  
  $ cat $PBS_NODEFILE
PBS Variables

• There are lots:

$PBS_ENVIRONMENT  $PBS_JOBID  $PBS_MOMPORT  $PBS_NP  $PBS_O_HOME
$PBS_O_LOGNAME  $PBS_O_QUEUE  $PBS_O_WORKDIR  $PBS_VERSION  $PBS_GPUFILE
$PBS_JOBNAME  $PBS_NODEFILE  $PBS_NUM_NODES  $PBS_O_HOST  $PBS_O_MAIL
$PBS_O_SERVER  $PBS_QUEUE  $PBS_VNODENUM  $PBS_JOBCOOKIE  $PBS_MICFILE
$PBS_NODENUM  $PBS_NUM_PPN  $PBS_O_LANG  $PBS_OPATH  $PBS_O_SHELL
$PBS_TASKNUM  $PBS_WALLTIME
Workflow

Head Node
- User 1
  - Program A
  - PBS Script A
- User 2
  - Program B
  - PBS Script B

Compute Node 01
Compute Node 02
Compute Node 03
Compute Node 04
Compute Node 05

Shared filesystems – All nodes can access the same programs and write output
Workflow

- Head Node
  - User 1
    - Program A
    - PBS Script A
  - User 2
    - Program B
    - PBS Script B
- Scheduler (MAUI)
- Compute Node 01
- Compute Node 02
- Compute Node 03
- Compute Node 04
- Compute Node 05

Shared filesystems – All nodes can access the same programs and write output.
Shared filesystems – All nodes can access the same programs and write output
Workflow

- **Head Node**
  - User 1
    - Program A
    - PBS Script A
  - User 2
    - Program B
    - PBS Script B

- **Scheduler (MAUI)**
  - Compute Node 01: PBS Script A
  - Compute Node 02
  - Compute Node 03
  - Compute Node 04: PBS Script B
  - Compute Node 05

- **Shared filesystems** – All nodes can access the same programs and write output
Workflow

Shared filesystems – All nodes can access the same programs and write output
The scheduler looks at all the currently queued and running jobs and runs a backfill algorithm.

Smaller jobs in terms of number of CPUs and requested time are easier to schedule since there is more likely to be space for them.

However the longer a job is in the queue the more priority it gets. This way every job runs eventually.

To see how long it will be before a job starts and completes, enter:

```
$ /usr/local/maui/bin/showstart <job ID>
```
Writing a Torque Batch Script

- Make a directory called ~/workshop
- Edit a new file in the workshop directory, name it workshop_example.pbs
- This REQUESTS time on the Core2 queue. We are asking for 2 nodes, and 2 cores on each node. We promise our job won’t take more than 10 minutes. Email me when the begins, aborts, and ends (bae). Combine standard out and standard error into one file.

```bash
#!/bin/bash
#PBS -q Core2
#PBS -l nodes=2:ppn=2
#PBS -l walltime=00:10:00
#PBS -N ws_example
#PBS -S /bin/bash
#PBS -j oe
#PBS -m bae
#PBS -M my_email@unm.edu
#PBS -V
```
Writing a Torque Batch Script

#!/bin/bash

#PBS -q Core2
#PBS -l nodes=2:ppn=2
#PBS -l walltime=00:10:00
#PBS -N ws_example
#PBS -S /bin/bash
#PBS -j oe
#PBS -m bae
#PBS -M my_email@unm.edu
#PBS -V

echo $HOSTNAME
Managing your jobs

To submit your batch job:

```
$ qsub workshop_example.pbs
```
Writing a Torque Batch Script

Whatever is written to standard out and standard error is saved to <Job Name>.o<Job ID> When the job ends.

If you want to see the output live, start your job with:

\$ qsub -k oe <script_name.pbs>
Writing a Torque Batch Script

```
#!/bin/bash

#PBS -q Core2
#PBS -l nodes=2:ppn=2
#PBS -l walltime=00:10:00
#PBS -N ws_example
#PBS -S /bin/bash
#PBS -j oe
#PBS -m bae
#PBS -M my_email@unm.edu
#PBS -V

cat $PBS_NODEFILE
```

Everything that comes after the PBS preamble is executed on the first node you were allocated.
Writing a Torque Batch Script

```bash
#!/bin/bash

#PBS -q Core2
#PBS -l nodes=2:ppn=2
#PBS -l walltime=00:10:00
#PBS -N ws_example
#PBS -S /bin/bash
#PBS -j oe
#PBS -m bae
#PBS –M my_email@unm.edu
#PBS –V

$PBS_O_WORKDIR/workshop_example1
```

Everything that comes after the PBS preamble is executed on the first node you were allocated.
# Job Arrays

```
#!/bin/bash

#PBS -q Core2
#PBS -l nodes=1:ppn=2
#PBS -l walltime=00:10:00
#PBS -N ws_example
#PBS -S /bin/bash
#PBS -j oe
#PBS -m bae
#PBS -M my_email@unm.edu
#PBS -V
#PBS -t 1-12%3

echo "$HOSTNAME - $PBS_ARRAYID"
```

-t allows you to schedule many jobs at once. -t 1-12%3 means run 12 jobs but only schedule 3 at a time.
Writing a Torque Batch Script

#!/bin/bash

#PBS -q Core2
#PBS -l nodes=1:ppn=2
#PBS -l walltime=00:10:00
#PBS -N ws_example
#PBS -S /bin/bash
#PBS -j oe
#PBS -m bae
#PBS –M my_email@unm.edu
#PBS –V
#PBS –t 1-12%3

$PBS_O_WORKDIR/ws_example1

Everything that comes after the PBS preamble is executed on the first node you were allocated.
Writing a Torque Batch Script

```bash
#!/bin/bash

#PBS -q Core2
#PBS -l nodes=1:ppn=2
#PBS -l walltime=00:10:00
#PBS -N ws_example
#PBS -S /bin/bash
#PBS -j oe
#PBS -m bae
#PBS -M my_email@unm.edu
#PBS -V
#PBS -t 1-12

$PBS_O_WORKDIR/ws_example2 "my_inputfile$PBS_ARRAYID"
```

You can use this to execute your program with many different input files. Here the program will run on files my_inputfile1, my_inputfile2, ... my_inputfile12
Hands on

Write some text files that will be input to the program you wrote earlier. Give them all the same file extension: say .ws

If you want to use our canned matrix inverted program let us know.

Use the Linux find command to send the paths to all the ws files to ws_example2

Play with the –joblog option to get some statistics on your job
Embarrassingly Parallel Problems

• Perfect case!
• All computation is independent so speedup is equal to the number of additional computers you throw at the problem.
• Especially good for generating lots of samples when you have a stochastic algorithm, simulation, or want to benchmark performance.
GNU Parallels – Input Driven

$ module load parallel-20170322-intel-18.0.2-4pa2ap6

$ find . -name "*.txt"

$ parallel echo ::: A B C ::: D E F

$ find . -name "*.txt" | parallel echo {}

$ find . -name "*.txt" | parallel echo {.}

$ find . -name "*.txt" | parallel echo {/./}
GNU Parallels – Environments

Recall that software may require particular environments. GNU Parallel by itself loses the environment in which it was called.

Use `env_parallel` to keep the environment. Need to tell it what shell you are using.

```
source `which env_parallel.bash`
```

Then you can use `env_parallel` exactly like `parallel`.
GNU Parallels

# Create a temporary unique directory in which to store the summary output for each job
TEMP_DIR=$(mktemp -d -p $PBS_O_WORKDIR)

# Setup Gurobi solver environment
module load parallel-20170322-intel-18.0.2-4pa2ap6
module load gurobi
module load anaconda
source activate gurobi

source `which env_parallel.bash`

# Use find to make a list of all the .graph files to pass to the integer program solver.
# Divide the work up among compute nodes using the GNU parallel tool. Use a local /tmp work directory.
# "::: -" reads from stdin (find ... *.graph) to {1}, "::: $EPSILON_VALUES_PATH" reads from the epsilon parameter file to {2}, {1/} fetches the input base filename only
find $GRAPH_INPUT_DIR -name '*.graph' | env_parallel --jobs 8 --joblog $PBS_O_WORKDIR/ip_progress.joblog --resume --sshloginfile $PBS_NODEFILE -workdir $(mktemp -d) "python $PBS_O_WORKDIR/ip/solve_ip.py {1} $IP_METHOD {2} $TEMP_DIR/{1/}.txt $TIME_LIMIT $SOLUTION_OUTPUT_DIR" ::: - $EPSILON_VALUES_PATH
GNU Parallels

find $GRAPH_INPUT_DIR -name '*.graph' | env_parallel --jobs 8 $PBS_O_WORKDIR/ip_progress.joblog --resume --sshloginfile $PBS_NODEFILE --workdir $(mktemp -d) "python $PBS_O_WORKDIR/ip/solve_ip.py {1} $IP_METHOD {2} $TEMP_DIR/{1/.}.txt $TIME_LIMIT $SOLUTION_OUTPUT_DIR" :::: - $EPSILON_VALUES_PATH

Let’s try to parse this command together...

Some syntax I haven’t explained yet
GNU Parallels – Monitoring Progress

• A logfile of the jobs completed so far can be generated with --joblog:

```
$ parallel --joblog $PBS_O_WORKDIR/job.log exit :::: 1 2 3 0
$ cat $PBS_O_WORKDIR/job.log
```

• The log contains the job sequence, which host the job was run on, the start time and run time, how much data was transferred, the exit value, the signal that killed the job, and finally the command being run.

We are running “exit” so we can fake jobs that succeed and fail.
GNU Parallels – Resuming Jobs

• With a joblog GNU parallel can be stopped and later pickup where it left off. It it important that the input of the completed jobs is unchanged.
• Why would you want to do this…???

```
$ parallel --joblog $PBS_O_WORKDIR/job.log exit ::: 1 2 3 0
$ cat $PBS_O_WORKDIR/job.log
$ parallel --resume --joblog $PBS_O_WORKDIR/job.log exit ::: 1 2 3 0 0 0
$ cat $PBS_O_WORKDIR/job.log
```
GNU Parallels – Resuming Jobs

• The previous command just ran the jobs that didn’t finish
• This command reruns jobs that has a failing exit code

```
$ parallel --joblog $PBS_O_WORKDIR/job.log exit ::: 1 2 3 0
$ cat $PBS_O_WORKDIR/job.log
$ parallel -resume-failed --joblog $PBS_O_WORKDIR/job.log exit ::: 1 2 3 0 0 0
$ cat $PBS_O_WORKDIR/job.log
```

We are running “exit” so we can fake jobs that succeed and fail.
GNU Parallels

- GNU Parallel is what you should be using to run many experiments, solve many independent instances of a problem.
- If you have 1000 input files and 100 CPUs allocated parallel will do all the scheduling for you to process those files.
GNU Parallels

- If you have 1000 input files and 100 CPUs allocated parallel will do all the scheduling for you to process those files.
- Remember though: Torque assigns you resources, parallels makes use of them. You have to use
  --sshloginfile $PBS_NODEFILE
- To be sure parallels is using resources you were actually allocated
MPI

Some parallel computations need the programs running on each core to communicate with each other.

The Message Passing Interface (MPI) supports this kind of communication.

We will look at a simple example using MPI.
Example Problem: Calculate $\pi$
Serial Calculation of $\pi$

$$\int_{0}^{1} \frac{4}{1+x^2} \, dx = \pi$$

And can be numerically approximated with:

$$\sum_{i=0}^{N} \frac{4}{1+x_i^2} \Delta x \approx \pi$$

As $\Delta x$ gets smaller and $N$ gets larger the approximation converges on $\pi$.
Serial Program to Calculate $\pi$

![Graph showing the calculation of $\pi$](image)

$\pi \approx \frac{4}{1 + x^2}$

1$^\text{Sung Bae, Ph.D, New Zealand eScience Infrastructure}$
Serial Program to Calculate $\pi$

calcPiSerial.py

```python
import time

def Pi(num_steps):  # Function to calculate pi
    step = 1.0/num_steps
    sum = 0
    for i in xrange(num_steps):
        x = (i+0.5)*step
        sum = sum + 4.0/(1.0+x*x)
    pi = step * sum
    return pi

if __name__ == '__main__':  # Main function
    start = time.time()  # Start timing
    num_steps = 10000000
    pi=Pi(num_steps)
    end = time.time()  # Stop timing
    # If this is the root process print the result
    print "Pi=%f (calculated in %f secs)" % (pi, end-start)
```
Serial Program to Calculate $\pi$
Parallel Program to Calculate $\pi$
As before we will need to load an environment module to get access to the MPI libraries we need.

To load the MPI environment enter:

```
module load openmpi-3.1.1-intel-18.0.2-vde2j7x
```

Other useful commands:

Find modules: `module spider <search string>`
List all modules on the system: `module avail`
List loaded modules: `module list`
Unload a module: `module unload <module name>`
Anaconda Environment

We will create a conda environment that gives us access to mpi4py.

To create an environment called “wheeler_mpi_py2” that includes the python packages for numerical computing, scientific computing, and MPI enter:

```
module load anaconda to load the anaconda module
conda create --name wheeler_mpi_py2 python=2 mpi4py numpy scipy
```

Once the python packages have finished installing enter:

```
source activate wheeler_mpi_py2
```
calcPiMPI.py

```python
from mpi4py import MPI
import time

# Get MPI variables
comm = MPI.COMM_WORLD # Communication framework
root = 0 # Root process
rank = comm.Get_rank() # Rank of this process
num_procs = comm.Get_size() # Total number of processes

# Distributed function to calculate pi
def Pi(num_steps):
    step = 1.0/num_steps
    sum = 0
    for i in xrange(rank, num_steps, num_procs):
        x = (i+0.5)*step
        sum = sum + 4.0/(1.0+x*x)
        mypi = step * sum
        pi = comm.reduce(mypi, MPI.SUM, root)
    return pi

# Main function
if __name__ == '__main__':
    start = time.time() # Start timing
    num_steps = 10000000
    # Broadcast number of steps to use to the other processes
    comm.bcast(num_steps, root);
    pi=Pi(num_steps)
    end = time.time() #Stop timing
    # If this is the root process print the result
    if (rank==root): print "Pi=%f(calculated_%iin_%fsecs)" %(pi, end-start)
```
Write and compile your code on the wheeler head node. Run your programs on the compute nodes.

We use the **torque** system to schedule jobs on the compute nodes.

Some useful scheduler commands:

Show current jobs: `qstat -a`

Queue information: `qstat -q` or `qgrok`

Show jobs of a particular user: `qstat -u <username>`

Submit a job: `qsub <pbs² script>`

Delete a job: `qdel <job ID>`

---

²portable batch system
calc_pi.pbs

#!/bin/bash

#PBS -l nodes=2:ppn=8
#PBS -l walltime=00:05:00
#PBS -N calc_pi
#PBS -S /bin/bash
#PBS -j oe
#PBS -M youremailaddress@unm.edu
#PBS -V

module load openmpi-3.1.1-intel-18.0.2-vde2j7x
module load anaconda
source activate wheeler_mpi_py2

mpirun -n $PBS_NP python calcPiMPI.py
File systems

Home directory. Limited space. Limited speed. Store your code here. Backed up. Path: ~

Scratch. Fast. Lot’s of space. Not-backed up. Store data that you can regenerate here. Path: ~/wheeler_scratch

Temp (on Wheeler these are RAM drives). Very fast but decreases available RAM. Path: /tmp
Final Thoughts

During this tutorial you have learned about:
- Environment modules
- Writing PBS scripts and submission of jobs to the TORQUE/MAUI system
- Embarrassingly parallel tasks and GNU Parallel
- More tightly coupled parallelism with the Message Passing Interface (MPI)
- Scratch vs user space

For assistance send an email to help@carc.unm.edu