You have been set up with temporary accounts from which we can run some Condor jobs.

To get started, ssh to `chopin.cs.wisc.edu`, and login using the username and passwords distributed earlier.

If you don’t have ssh and are running Windows, you can get it from

```
http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
```

Now run the following commands:

```
cd /scratch
mkdir your_name  (choose some unique identifier)
cd your_name
cp -r /scratch/COPTA/* .
source ./setit
```
Mmmmmmmmmmmmmmmmm. Pie

- Our first computational task will be to estimate $\pi$ by numerical integration.
- Everyone knows...

$$\int_0^1 \frac{1}{1 + x^2} \, dx = \arctan(x)\big|_{x=0}^{x=1} = \arctan(1) = \frac{\pi}{4}.$$
The Rectangle Rule

\[ \frac{4}{1+x^2} \]
A Program to Estimate $\pi$

- We’ve written a $\pi$-calculator for you.

```
cd pi
gcc pi1.c -lm -o pi1
./pi1 1000
```

- This is not a parallel program. Just a simple (one process) program.
Condor Universes

- Condor jobs run in a specific Condor Universe

  - Standard—Has cool features like checkpointing and migration of jobs
    - Requires special linking of your program
  - Vanilla—No cool condor features (regular)
  - MPI/PVM/Java/Grid
    - Not mentioned here today, but they exist.
Compiling for Condor

- **Standard Universe**
  - Put the command `condor_compile` in front of your normal link line.
  - `[jtl3@fire1 condor]$ condor_compile gcc pi1.c -o pi1-standard -lm`

- **Vanilla Universe**
  - Do nothing

- **Condor submission is like other resource management software**
  - Describe your job in a *job submission file*
  - Submit and monitor your job with command line programs
Sample Condor Submission Files

universe = standard
executable = pi1-standard
arguments = 1000000000
output = pi1.out
error = pi1.err
notification = Complete
notify_user = me@gmail.com
getenv = True
rank = kflops
queue

universe = vanilla
executable = pi1
arguments = 666
output = pi1.out
requirements = (OpSys != WINNT51)
error = pi1.err
getenv = True
rank = Memory
queue

- man condor_submit
The Big Four

- `condor_submit <job.condor>`
  - Submit a job to the Condor scheduler
- `condor_q`
  - Check the status of the queue of Condor jobs
- `condor_status`
  - Check the status of the Condor pool
- `condor_rm <jobid>`
  - Delete a Condor job
Let’s Do It!

[jtl3@fire1 condor]$ condor_submit run.condor
Submitting job(s).
1 job(s) submitted to cluster 16.

[jtl3@fire1 condor]$ condor_q

-- Submitter: fire1.cluster : <192.168.0.1:32777> : fire1.cluster

<table>
<thead>
<tr>
<th>ID</th>
<th>OWNER</th>
<th>SUBMITTED</th>
<th>RUN_TIME</th>
<th>ST</th>
<th>PRI</th>
<th>SIZE</th>
<th>CMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>16.0</td>
<td>jtl3</td>
<td>8/4 11:22</td>
<td>0+00:00:16</td>
<td>R</td>
<td>0</td>
<td>3.4</td>
<td>pi1-standard</td>
</tr>
</tbody>
</table>

[jtl3@fire1 condor]$ cat pi1.out
pi is about 3.1415926555921398488635532
Error is 2.0023467328655897e-09

- I could do condor_rm 16.0
- Any Condor questions?
Condor Parallel Example: Statistical Bootstrapping

\{z_1, z_2, z_3, z_4, z_5, ... \} Distribution \rightarrow \{z_2, z_2, z_5, ... \} Sample

\text{Resamp } \{z_2, z_5, z_7, ... \} \quad \text{Resamp } \{z_5, z_7, z_9, ... \} \quad \text{Resamp } \{z_7, z_7, z_9, ... \}

\text{Analyze} \quad \text{Analyze} \quad \text{Analyze}

\text{Coalesce}
Statistical Bootstrapping

Driver Creates distribution.

```
# driver.m

dist_size = 100000;
d = rand(dist_size, 1) .* 500;
subset = d(floor(rand(1000,1) .* 100000));
save ”subset” subset;
```

```
# submit

universe = vanilla
executable = worker.m
transfer_files = true
when_to_transfer_output = on_exit
transfer_input_files = subset
output = mean.$(PROCESS)
log = log
queue 5
```

Linderoth (UW-Madison)  An Introduction to the Computational Grid  ICCOPT II 12 / 15
Parallel Job on Condor: Statistical Bootstrapping

Running the example

Shell prompt

# ./driver.m
Submitting job(s).....
Logging submit event(s).....
5 job(s) submitted to cluster 565262.

5 minutes later...

All jobs done.
mean of mean is 161.014978
Let’s run it!

Go to your directories scratch/your_name/octave on chopin.cs.wisc.edu, and look at the files submit, driver.m, and worker.m

You’ll see that they contain the material from the previous slides. The two .m files have a line at the start indicating that they are to be run using Octave (a free Matlab-like language available from www.octave.org.

Now run it by typing “driver.m” on the command line!

- driver.m creates the file “subset” and then invokes submit.
- Five instances of worker.m are submitted into the condor pool. You can check on the status of these by typing condor_q (in another window).
- When the five jobs are finished, driver.m does the final computation and prints a message to the screen.
Any Questions?

- What optimization problem would you like to solve on the Grid?
- We’re here to help!