Resource Sharing for Pivot
Derek Weitzel - HCC
Connecting your PIvOT cluster

• Pivot clusters are very small

• If the pivot cluster is full, where can researchers run?
Connecting your PIvOT cluster

• This is the same problem we had in the national infrastructure
• Solved it with a grid, the Open Science Grid
Connecting your PIvOT cluster

• Use the same methods as the OSG

• Resource Sharing
  – My jobs run on your computers!

• Independent resource control
  – Set policies for your own cluster
Connecting your PIVOT cluster

• There are different ways to share the PIVOT cluster:
  – PIVOT ➔ HCC – Jobs submitted on the PIVOT cluster can run on the supercomputers here at the Holland Computing Center
  – PIVOT ➔ PIVOT – Jobs submitted on PIVOT clusters will run on other PIVOT clusters.
  – HCC ➔ PIVOT – Jobs submitted at HCC can run on PIVOT clusters.
PIvOT ➔ HCC

• The PIvOT to HCC connection is probably the most useful.

• Jobs submitted to Condor on your pivot cluster can run on HCC Computers.

• Very useful if the cluster is full, consider this an overflow.
PivOT ➔ HCC: What it looks like

• Normal Condor Submission

• Jobs will first look at your own cluster, if it can’t run there, the jobs will move to other clusters

• List of other clusters configured by admin
PIvOT ➔ HCC: What it looks like

- Submit Locally, Compute Remotely
PIvOT ➡️ HCC: What it looks like

• Submit File and descriptions can be found on hcc documentation: http://go.unl.edu/pivotgriduser

• Important to list out the input files.
PlvOT ➔ HCC: What it looks like

• Submit File:

```plaintext
universe = vanilla
Executable = myjob.exe
arguments = 10
output = job.output
error = job.error
log = job.log
transfer_input_files = input1, input2
should_transfer_files = YES
when_to_transfer_output = ON_EXIT
queue 1
```

• Submit Job:

```bash
$ condor_submit example.submit
```
PIvOT ➔ HCC: What it looks like

- A Condor Submit file describes the job to the system

```plaintext
universe = vanilla
Executable = myjob.exe
arguments = 10
output = job.output
error = job.error
log = job.log
transfer_input_files = input1, input2
should_transfer_files = YES
when_to_transfer_output = ON_EXIT
queue 1
```
PIvOT ➔ HCC: What it looks like

• Executable is the (compiled?) executable that will run on the remote host

```plaintext
universe = vanilla
Executable = myjob.exe
arguments = 10
output = job.output
error = job.error
log = job.log
transfer_input_files = input1, input2
should_transfer_files = YES
when_to_transfer_output = ON_EXIT
queue 1
```
PIvOT ➔ HCC: What it looks like

- Arguments are used for the application when it runs on the worker node.

```plaintext
universe = vanilla
Executable = myjob.exe
arguments = 10
output = job.output
error = job.error
log = job.log
transfer_input_files = input1, input2
should_transfer_files = YES
when_to_transfer_output = ON_EXIT
queue 1
```
PIvOT ➔ HCC: What it looks like

- Output and Error is the file locations that the executable’s stdout and stderr should go.

```plaintext
universe = vanilla
Executable = myjob.exe
arguments = 10
output = job.output
error = job.error
log = job.log
transfer_input_files = input1, input2
should_transfer_files = YES
when_to_transfer_output = ON_EXIT
queue 1
```
PIvOT ➔ HCC: What it looks like

- Log shows the job’s transitions (idle -> running)

```plaintext
universe = vanilla
Executable = myjob.exe
arguments = 10
output = job.output
error = job.error
log = job.log
transfer_input_files = input1, input2
should_transfer_files = YES
when_to_transfer_output = ON_EXIT
queue 1
```
PIvOT ➔ HCC: What it looks like

• Log shows the job’s transitions (idle -> running)

```plaintext
universe = vanilla
Executable = myjob.exe
arguments = 10
output = job.output
error = job.error
log = job.log
transfer_input_files = input1, input2
should_transfer_files = YES
when_to_transfer_output = ON_EXIT
queue 1
```
PIvOT ➔ HCC: What it looks like

- Lists the files to be transferred with the job.
- Job is not running in your home directory.

```
universe = vanilla
Executable = myjob.exe
arguments = 10
output = job.output
error = job.error
log = job.log
transfer_input_files = input1, input2
should_transfer_files = YES
when_to_transfer_output = ON_EXIT
queue 1
```
PIvOT ➔ HCC: What it looks like

• Tell Condor when to transfer the files

```
universe = vanilla
Executable = myjob.exe
arguments = 10
output = job.output
error = job.error
log = job.log
transfer_input_files = input1, input2
should_transfer_files = YES
when_to_transfer_output = ON_EXIT
queue 1
```
PIvOT ➔ HCC: What it looks like

• Submit 1 of the jobs described above

```
universe = vanilla
Executable = myjob.exe
arguments = 10
output = job.output
error = job.error
log = job.log
transfer_input_files = input1, input2
should_transfer_files = YES
when_to_transfer_output = ON_EXIT
queue 1
```
PIvOT ➔ HCC: What it looks like

- Submit 1 of the jobs described above

**Interactive Submission**

- error = job.error
- log = job.log
- transfer_input_files = input1, input2
- should_transfer_files = YES
- when_to_transfer_output = ON_EXIT
- queue 1
So what just happened?

1. Job was submitted to cluster
2. Job was scheduled onto a worker node
3. Input files and executable was transferred to the worker node
4. Executable is started on the worker node
5. Executable completes
6. Stdout and Stderr, as well as files that were changed are transferred back to submitter
PIvOT ➔ HCC: What it looks like

• Job Status (when flocking to HCC):

```
$ condor_q

-- Submitter: login.sandhills.hcc.unl.edu : <129.93.229.137:53691> :
login.sandhills.hcc.unl.edu
ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
9544.0  dweitzel        11/6 22:17   0+00:00:00 I  0   0.1 ls -l

1 jobs; 0 completed, 0 removed, 1 idle, 0 running, 0 held, 0 suspended
```
PIvOT ➔ HCC: What it looks like

- Condor Configuration

```bash
# Remote Clusters to flock to
FLOCK_TO = $(FLOCK_TO) hcc-sandman.unl.edu

# Who to trust? Include the <VO> DN to trust the submitter
GSI_DAEMON_NAME = $(GSI_DAEMON_NAME), /DC=org/DC=doegrids/OU=Services/CN=hcc-sandman.unl.edu

########################
### May need to edit ###
########################
# This host's certificates
# These are needed if you are flocking to a HCC Host
GSI_DAEMON_CERT = /etc/grid-security/hostcert.pem
GSI_DAEMON_KEY = /etc/grid-security/hostkey.pem

########################
### Safe to leave alone ###
########################
# Enable authentication from the Negotiator
SEC_ENABLE_MATCH_PASSWORD_AUTHENTICATION = TRUE
# Enable gsi authentication and CLAIMTOBE
SEC_DEFAULT_AUTHENTICATION_METHODS = FS,GSI,CLAIMTOBE
```

docs at: http://go.unl.edu/pivotgrid
We are looking for beta testers

• We want users for the Pivot Grid

• If you want your cluster to overflow to HCC’s, let us know.

• We can help with setup, administration, and getting user’s jobs to run