AN INTRODUCTION TO HIGH-THROUGHTPUT COMPUTING AT ISYE

CREATING THE NEXT®

WHAT IS COVERED IN THIS TUTORIAL

- How to get access to the ISYE Cluster
- How to get your files to the ISYE Cluster
- How to access the Cluster
- What is HTCondor ?
- What the different between HPC and High-Throughtput Computing
- How to Run a Job with HTCondor
- How to Monitor a HTCondor job
- Submitting a simple jobs with HTCondor
- Submitting Multiple Jobs with HTCondor
- Testing and Troubleshooting
- Cluster Documentation

HPC RESOURCES IN ISYE

- ISyE
  - HTCondor Cluster – Group of Linux servers with standard set of software and networked home drives scheduled via HTCondor and maintained by ISyE IT staff and housed within Groseclose. **Open to <u>all</u> members of the ISyE community** (faculty, undergraduate, graduate, & postdocs)

GETTING ACCESS TO THE ISYE CLUSTER

- First, sign up for an ISyE account: https://www.isye.gatech.edu/about/school/computing
- Get a terminal app:
  - Windows:
    - PuTTY: https://www.chiark.greenend.org.uk/~sgtatham/putty/ (careful if you Google!)
    - SecureCRT: https://software.oit.gatech.edu
    - Windows Subsystem for Linux: https://docs.microsoft.com/en-us/windows/wsl/install-win10
  - Mac:
    - Terminal.app (built-in)
    - iTerm2: https://www.iterm2.com
  - Linux: It's built in!
  - iOS: Prompt: https://panic.com/prompt
  - Android: lots of options https://www.dunebook.com/10-best-android-terminal-emulator/

GETTING YOUR FILES TO THE CLUSTER

File access via SCP/SFTP:

- Windows:
  - WinSCP: https://winscp.net
  - FileZilla: https://filezilla-project.org/
- Mac:
  - Cyberduck: https://cyberduck.io
  - FileZilla: https://filezilla-project.org
  - Transmit: https://panic.com/transmit (not free)
- Linux:
  - scp at the command line, GUI varies by distro
- GitHub

HOW TO ACCESS THE ISYE CLUSTER

Off campus, or on GTwireless:

- Connect to VPN (https://faq.oit.gatech.edu/content/how-do-i-get-started-campus-vpn)

OR

- SSH first to **castle.isye.gatech.edu**

Once connected to VPN or castle, or if you're on the wired ISyE network:

- SSH to **wren.isye.gatech.edu**
- SCP/SFTP via **castle.isye.gatech.edu** (does not require VPN or an SSH connection to castle first)

HTCONDOR DOCUMENTATION

- Documentation:
  - ISyEwebsite (better & more documentation coming soon!)
    - https://www.isye.gatech.edu/about/school/computing
    - https://www2.isye.gatech.edu/apps/helpdesk/kb/browse/866-Computational-Resources
  - HTCondor website (make sure you select the 'stable' release)
    - https://research.cs.wisc.edu/htcondor/manual/
    - https://research.cs.wisc.edu/htcondor/manual/quickstart.html
    - http://chtc.cs.wisc.edu/multiple-jobs.shtml

WHAT IS HTCONDOR

Open source software for distributed High-Throughput computing. Developed at the University of Wisconsin in the 1980's. HTCondor has three components

- A job scheduler
- A resource manager
- A workflow management system

High-Throughput Computing - Allows for many computational tasks to be completed over a long period of time. Useful for researchers and other users who are more concerned with the number of computations they can do over long spans of time than they are with short-burst computations

WHY HTCONDOR ?

HTCondor manages and runs work on your behalf
Schedule tasks on a single computer to not overwhelm the computer

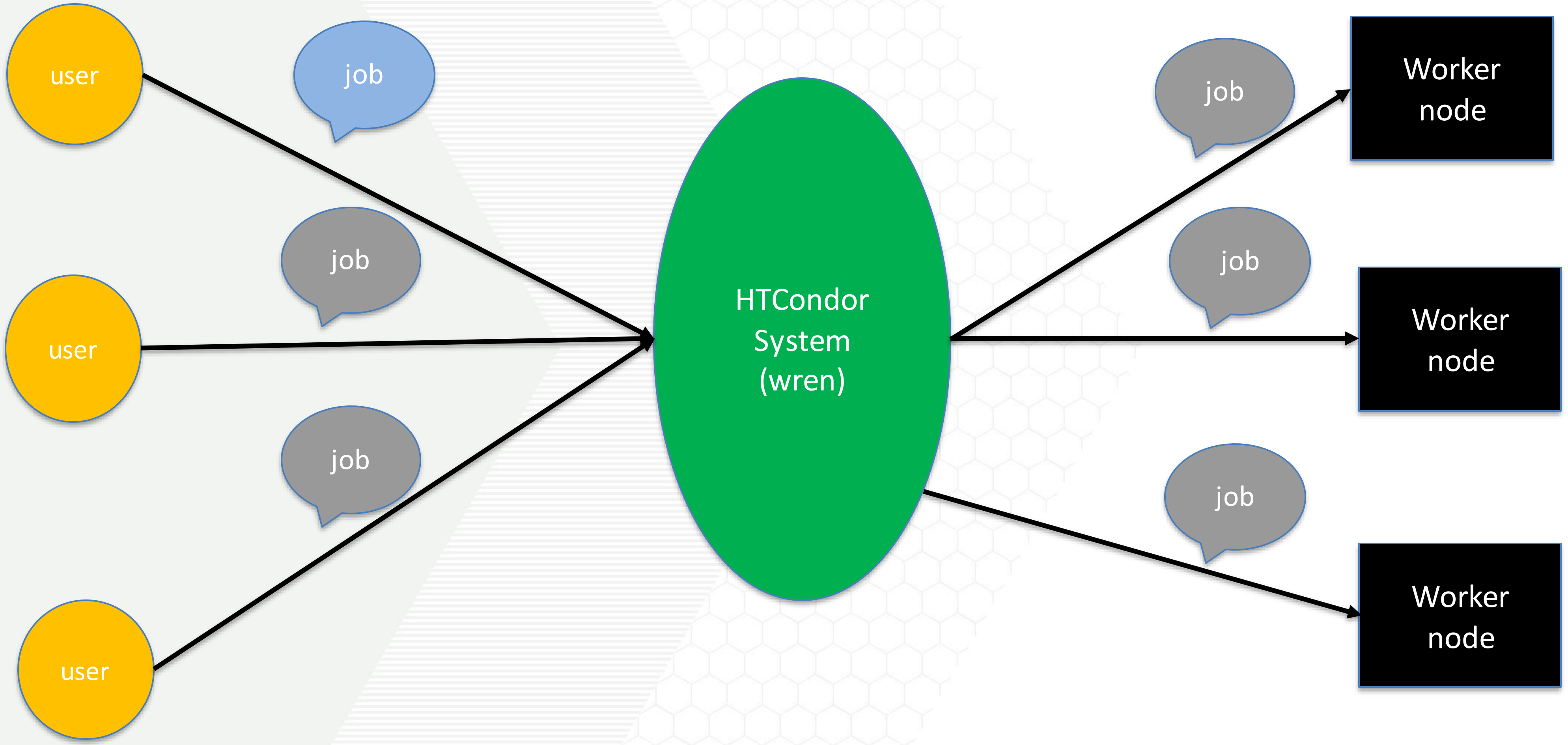
Schedule tasks on a group* of computers (which may/may not be directly accessible to the user)

Schedule tasks submitted by multiple users on one or more computers

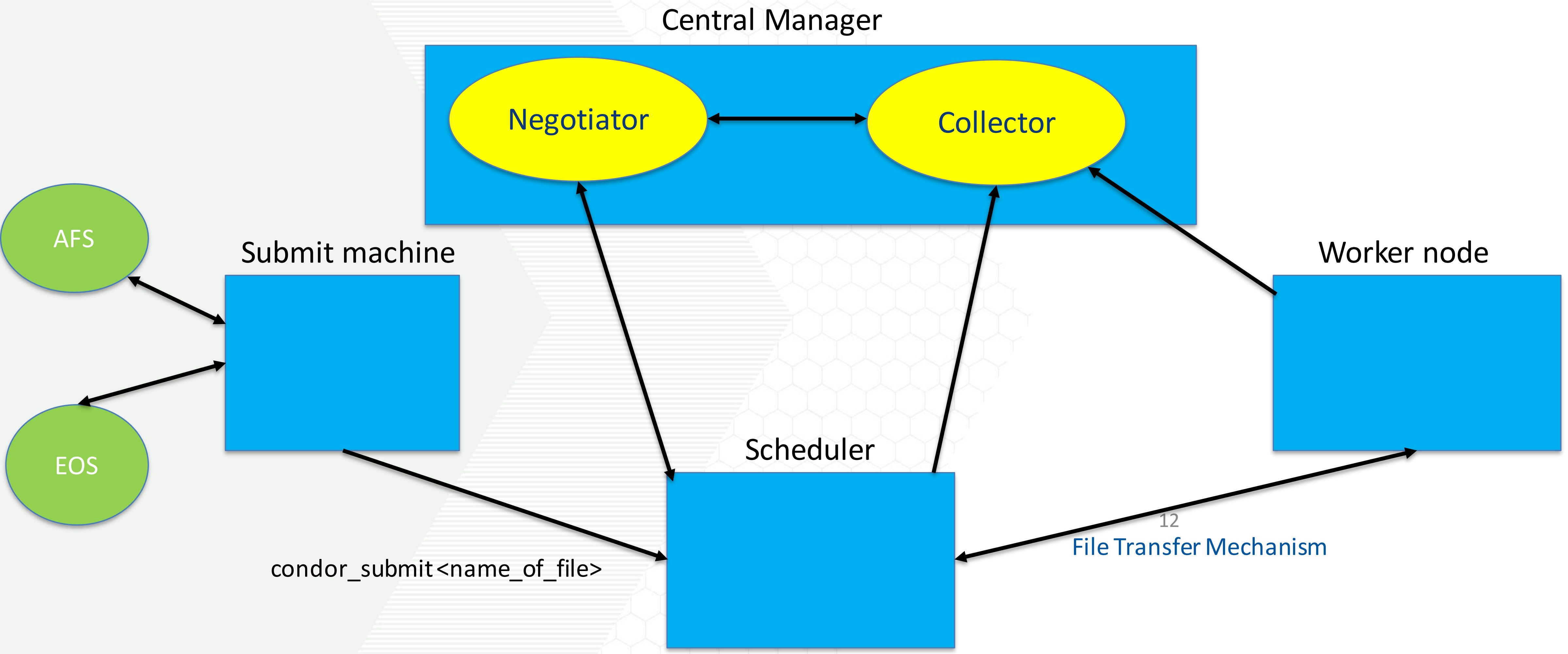
HTCONDOR CAPABILITIES

- HTCondor has built-in ways to submit multiple independent jobs with one submit file.
  - Analyze multiple data files
  - Test parameter or various input combination
- ...without having to:
  - Start each job individually
  - create separate submit files for each job

HTCONDOR WORKFLOW



HTCONDOR SERVICE COMPONENTS



HTCONDOR JOBS

- A single computing task is called a “job”
- Three main pieces of a job are the input, executable (program) and output
- Executable must be runnable from the command line without any interactive input

RESOURCE REQUEST

- Jobs are nearly always using a part of a computer, not the whole thing

- Very important to request appropriate resources
  - memory,
  - cpus
  - gpus
  - disk

- Even if your system has default CPU, memory and disk requests, these may be too small!

- Important to run test jobs and use the log file to request the right amount of resources

- Requesting too little: causes problems for your and other jobs; jobs might by held by HTCondor and requesting too much: jobs will match to fewer “slots”

Simple Submit File

```
universe = vanilla

getenv = true

executable = lp.py

arguments = test0.lp

Log = test.log

output = test.out

error = test.error

notification = error

notification = complete

request_memory = 1024

request_cpus = 1

queue
```

HOW TO SUBMITTING AND MONITOR YOUR JOB

To submit a job/jobs:

condor_submit *submit_file_name*

To monitor submitted jobs, use:

condor_q

condor_q –nobatch

- Show the user's job only

- Jobs summarized in "batches"

- **–nobatch to see extra detail about what is happening with a job.**

WHY IS MY JOB NOT RUNNING

- Jobs can go wrong “internally”:
  - something happens after the executable begins to run
- Has a badly formatted executable
- Uses too much memory
- Has a badly formatted command file
- Asked for too many resources.

TROUBLESHOOTING COMMANDS

```
condor_q -better <jobid>
condor_q -better <jobid> -machine <name>
condor_q -better -reverse -machine <machine>

condor_q -hold
```

SUBMITTING AND MONITORING

- To submit a job/jobs:

  condor_submit *submit_file_name*

```
$ condor_submit mul.cmd
Submitting job(s).
1 job(s) submitted to cluster 102329.
```

- To monitor submitted jobs, use:

  condor_q

```
$ condor_q

-- Schedd: isye-hps0005.isye.gatech.edu : <10.1.112.2:9618?... @ 03/04/20 13:11:06
OWNER   BATCH_NAME    SUBMITTED   DONE   RUN    IDLE  TOTAL JOB_IDS
kross48 ID: 102329   3/4  13:10      _      1      _      1 102329.0

Total for query: 1 jobs; 0 completed, 0 removed, 0 idle, 1 running, 0 held, 0 suspended
Total for kross48: 1 jobs; 0 completed, 0 removed, 0 idle, 1 running, 0 held, 0 suspended
Total for all users: 22 jobs; 0 completed, 0 removed, 1 idle, 21 running, 0 held, 0 suspended
```

HTCONDOR STATES

Idle (I)	• The job is waiting in the queue to be executed
Running (R)	• The job is running
Completed (C)	• The job is exited
Held (H)	• Something is wrong with the submission file • The user executed condor_hold <job_id>
Suspended (S)	• The user executed condor_suspend
Removed (X)	• The job has been removed by the user.

OTHER USEFUL COMMANDS

- condor_status -master – View all compute nodes and there resources
- condor_q -ana:sum (user)
- condor_rm <jobid> - Remove a job
- condor_qedit <jobid> RequestMemory 3078 – Edit an existing job
- condor_q –all – See the whole queue (All users, All Jobs)
- condor_rm -a - Removes all of your jobs.
- `condor_q -af HoldReason`
- condor_q -analyze 107.5
- condor_status -compact -constraint 'TotalGpus > 0' - View available GPU systems
- condor_status -compact -constraint 'TotalGpus > 0' -af Machine TotalGpus CUDADeviceName CUDACapability – Display GPU hardware information

RESOURCE REQUEST

- Jobs are nearly always using a part of a computer, not the whole thing
- Very important to request appropriate resources (memory, cpus, disk) for a job

Submitting Multiple jobs Using $(ProcId)

```
universe = vanilla

getenv = true

executable = $ENV(HOME)/gurobi/lp.py

arguments = test$(ProcID).lp

Log = $ENV(HOME)/gurobi/$(Cluster).log

output = $ENV(HOME)/gurobi/$(Cluster).$(ProcId).out

error = $ENV(HOME)/gurobi/$(cluster).$(ProcId).error

notification = error

notification = complete

request_memory = 1048

request_cpus = 1

queue 3
```

Use the `$(ClusterId)`, `$(ProcId)` variables to provide unique values to jobs.*

Submitting Multiple jobs Using Queue Statements

```
universe = vanilla

getenv = true

executable = lp.py

arguments = $(file)

Log = $ENV(HOME)/gurobi/$(Cluster).log

output = $ENV(HOME)/gurobi/$(Cluster).$(process).out

error = $ENV(HOME)/gurobi/$(cluster).$(Process).error

notification = error

notification = complete

request_memory = 1048

request_cpus = 1

queue file from testlp.txt
```

Submitting Multiple jobs Using InitialDir

```
universe = vanilla

getenv = true

initialdir = job$(process)

executable = lp.py

arguments = test.lp

Log = job.log

output = job.out

error = job.error

notification = error

notification = complete

request_memory = 1048

queue 3
```

CREATING THE NEXT®

APPLICATIONS AVAIABLE ON THE CLUSTER

- Ampl
- Mathematica
- Matlab
- Openmpi
- Cplex
- Python3
- Gams
- RH
- Gurobi
- xpressmp
- Julia
- Lua

Additional Batch Commands

```
periodic_remove = JobStatus == 2 &&  (CurrentTime – EnteredCurrentStatus > 24*60*60)
```

```
periodic_remove = JobStatus == 2 &&  (CurrentTime – EnteredCurrentStatus > 24*60*60)
```

HTCONDOR OUTPUT AND LOGS

These files are defined in the submit file
- Output
  - The STDOUT of the command or script
- Error
  - The STDERR of the command or script
- Log
  - Information about job's execution
    - execution host
    - the number of times this job has been submitted
    - the exit value, etc

Can use relative or absolute paths for all of them

HTCondor will search for this directory
- So it should be already created

VIEW MACHINES ATTRIBUTES

```
[kross48@isye-hps0005 examples]$ condor_status –compact

Machine                         Platform     Slots Cpus Gpus  TotalGb FreCpu  FreeGb  CpuLoad ST Jobs/Min MaxSlotGb

isye-abel.isye.gatech.edu       x64/RedHat7    0   32         377.71        32  377.71     0.00 Ui   0.00 *
isye-ames.isye.gatech.edu       x64/RedHat7    0   24         125.68        24  125.68     0.00 Ui   0.00 *
isye-boyle1.isye.gatech.edu     x64/RedHat7    1   16          47.00        15   15.75     0.06 **   0.00         31.25
isye-boyle10.isye.gatech.edu    x64/RedHat7    0   16          47.00        16   47.00     0.00 Ui   0.00 *
isye-bulldozer.isye.gatech.edu  x64/RedHat7    1   32         125.73        31  121.73     0.03 **   0.00          4.00
isye-dollar.isye.gatech.edu     x64/RedHat7    1   12          39.12        11    7.87     0.08 **   0.00         31.25
isye-fisher1.isye.gatech.edu    x64/RedHat7    0   24         125.73        24  125.73     0.00 Ui   0.00 *
isye-fisher2.isye.gatech.edu    x64/RedHat7    0   24         125.73        24  125.73     0.00 Ui   0.00 *
isye-fisher3.isye.gatech.edu    x64/RedHat7    0   24         125.73        24  125.73     0.00 Ui   0.00 *
isye-fisher4.isye.gatech.edu    x64/RedHat7    0   24         125.73        24  125.73     0.00 Ui   0.00 *
isye-goddard1.isye.gatech.edu   x64/RedHat7    1   12          70.62        11    6.62     0.13 **   0.00         64.00
isye-goddard2.isye.gatech.edu   x64/RedHat7    1   12          70.62        11   66.62     0.08 **   0.00          4.00
isye-gpu1001.isye.gatech.edu    x64/RedHat7    1   72    8    503.10        71  483.10     0.00 *i   2.00         20.00
isye-hpc0009.isye.gatech.edu    x64/RedHat7    1   10          57.44         9   55.44     0.85 **   0.00          2.00
*****
```

RETRIES, CONT.

- Can also combine with

  success_exit_code = < Integer >

  retry_until = < Integer | Expression >

```
executable = foo.exe
max_retries = 5
retry_untl = ExitCode >= 0
queue
```

HOLDING OR REMOVING JOBS

- If you know your job has a problem and it hasn’t yet completed, you can:
  - Place it on hold yourself, with **condor_hold [U/C/J]**

```
$ condor_hold bob
All jobs of user ”bob" have been held
```

```
$ condor_hold 128
All jobs in cluster 128 have been held
```

```
$ condor_hold 128.0
Job 128.0 held
```

  - Remove it from the queue, using **condor_rm [U/C/J]**

MULTI-CPU AND GPU COMPUTING

- Jobs that use multiple cores on a single computer can be run in the vanilla universe (parallel universe not needed):

```
request_cpus = 16
```

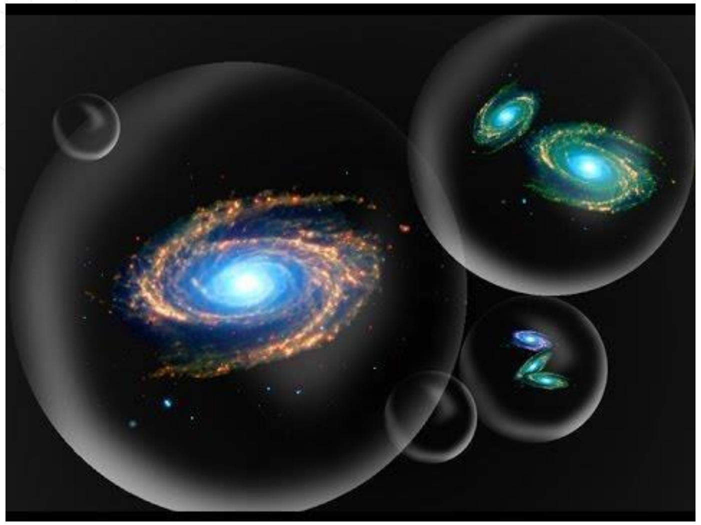
- If there are computers with GPUs, request them with:

```
request_gpus = 1
```

JOB UNIVERSES

- HTCondor has different “universes” for running specialized job types
- Vanilla (default)
  - – good for most software
- Set in the submit file using:

```
universe = vanilla
```



REVIEWING FAILED JOBS

- A job’s log, output and error files can provide valuable information for troubleshooting

Log	Output	Error
• When jobs were submitted, started, and stopped • Resources used • Exit status • Where job ran • Interruption reasons	Any “print” or “display” information from your program	Captured by the operating system

RETRIES

- Problem: a small number of jobs fail with a known error code; if they run again, they complete successfully.
- Solution: If the job exits with the error code, leave it in the queue to run again

```
max_retries = 3
```

USING $(PROCId)

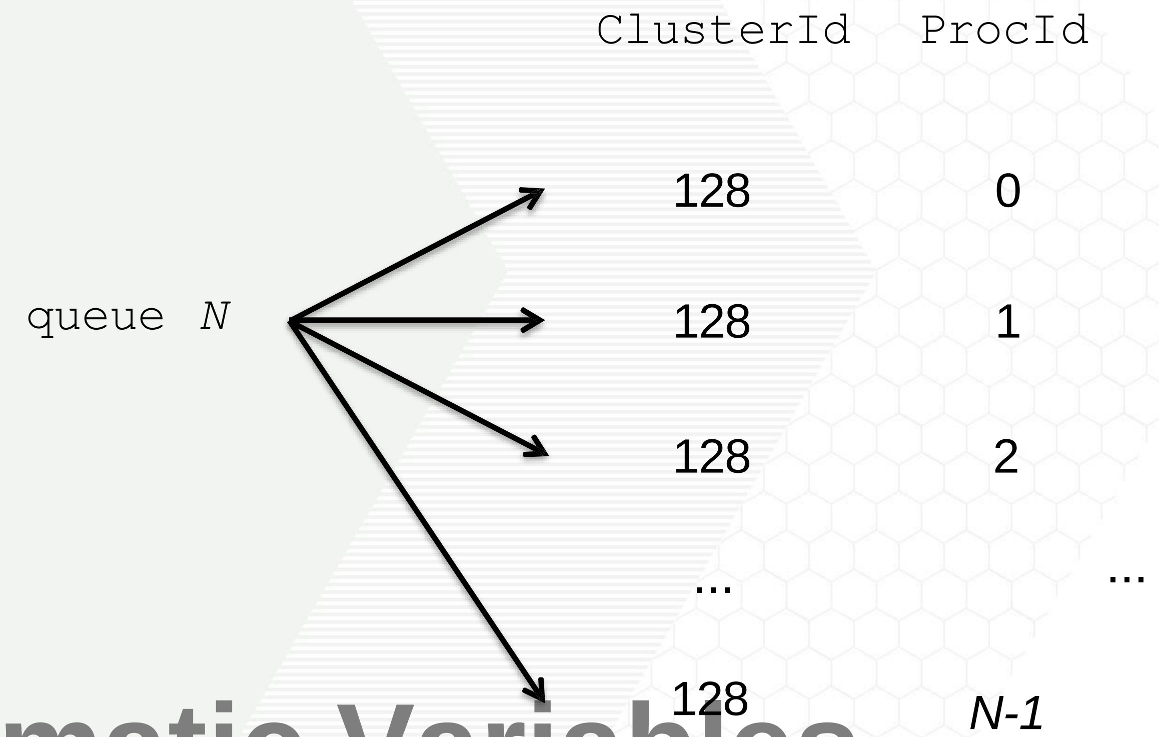
job.submit

```
executable = analyze.exe
arguments = file$(ProcId).in file$(ProcId).out
should_transfer_files = YES
transfer_input_files = file$(ProcId).in
when_to_transfer_output = ON_EXIT

log = job_$(ClusterId).log
output = job_$(ClusterId)_$(ProcId).out
error = job_$(ClusterId)_$(ProcId).err

queue 3
```

- Use the `$(ClusterId)`, `$(ProcId)` variables to provide unique values to jobs.*

* May also see `$(Cluster)`, `$(Process)` in documentation

AUTOMATIC VARIABLES

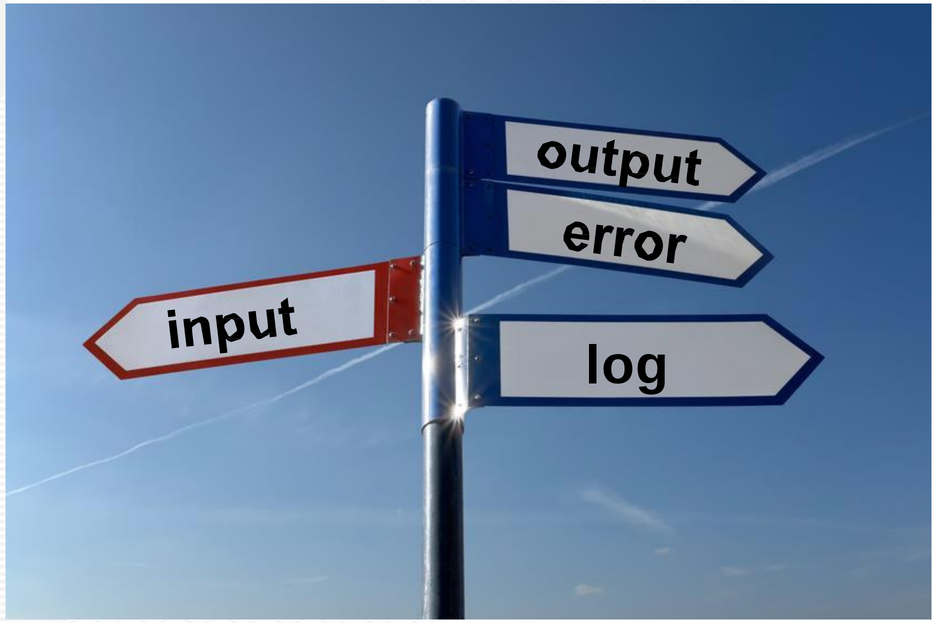


- Each job’s `ClusterId` and `ProcId` numbers are saved as job attributes
- They can be accessed inside the submit file using:
  – `$(ClusterId)`
  – `$(ProcId)`

Automatic Variables

ORGANIZE FILES IN SUB-DIRECTORIES

- Create sub-directories* and use paths in the submit file to separate input, error, log, and output files.



* must be created before the job is submitted

InitialDIR

- Change the submission directory for each job using `initialdir`
- Allows the user to organize job files into separate directories.
- Use the same name for all input/output files
- Useful for jobs with lots of output files

SEPARATE JOBS WITH initialdir

```
(submit_dir)/

job.submit          job0/          job1/          job2/
analyze.exe           file.in        file.in        file.in
                      job.log        job.log        job.log
                      job.err        job.err        job.err
                      file.out       file.out       file.out
```

job.submit

```
executable = analyze.exe
initialdir = job$(ProcId)
arguments = file.in file.out
transfer_input_files = file.in

log = job.log
error = job.err

queue 3
```

Executable should be in the directory with the submit file, *not* in the individual job directories

USE PATHS FOR FILE TYPE

```
(submit_dir)/
```

```
job.submit      file0.out      input/          log/            err/
analyze.exe     file1.out        file0.in        job0.log        job0.err
                file2.out        file1.in        job1.log        job1.err
                                 file2.in        job2.log        job2.err
```

job.submit

```
executable = lp.py
arguments = file$(Process).in file$(ProcId).out
transfer_input_files = input/file$(ProcId).in

log = log/job$(ProcId).log
error = err/job$(ProcId).err

queue 3
```

USING MULTIPLE VARIABLES

- Both the “`from`” and “`in`” syntax support using multiple variables from a list.

`job.submit`

```
executable = compare_states
arguments = -year $(option) -input
$(file)

should_transfer_files = YES
when_to_transfer_output = ON_EXIT
transfer_input_files = $(file)

```

```
queue file,option from job_list
```

`job_list.txt`

```
wi.dat, 2010
wi.dat, 2015
ca.dat, 2010
ca.dat, 2015
ia.dat, 2010
ia.dat, 2015
```

SELF-CHECKPOINTING HOT-TO

- Edit executable:
  - Atomically save intermediate states to a checkpoint file
  - Always check for a checkpoint file when starting
- Add HTCondor option that a) saves all intermediate/output files from the interrupted job and b) transfers them to the job when HTCondor runs it again

```
when_to_transfer_output = ON_EXIT_OR_EVICT
```

SELF-CHECKPOINTING

- By default, a job that is interrupted will start from the beginning if it is restarted.
- It is possible to implement self-checkpointing, which will allow a job to restart from a saved state if interrupted.
- Self-checkpointing is useful for very long jobs, and being able to run on opportunistic resources.

HOW TO INSTALL PYTHON PACKAGES LOCALLY

Procedures to install Python Packages not provided by Default

- pip install <module> -- user
- pip install virtualenv --user

WRAPPING UP

- Remember to use full paths for all files!
  - Including executable directive and all scripts, data files, log files, and output
  - In condor submit file, $ENV(HOME)=/home/username
  - In your code ~/ =/home/username
  - Not just in condor submission file, but in your code, too!

GETTING HELP

- In Person: Groseclose 302
- Email: helpdesk@isye.gatech.edu

  (please don’t email a specific person directly, use the helpdesk email to keep from getting lost!)

ISyE HPC Announcements Email List

https://lists.isye.gatech.edu/mailman/listinfo/isye-hpc-announce