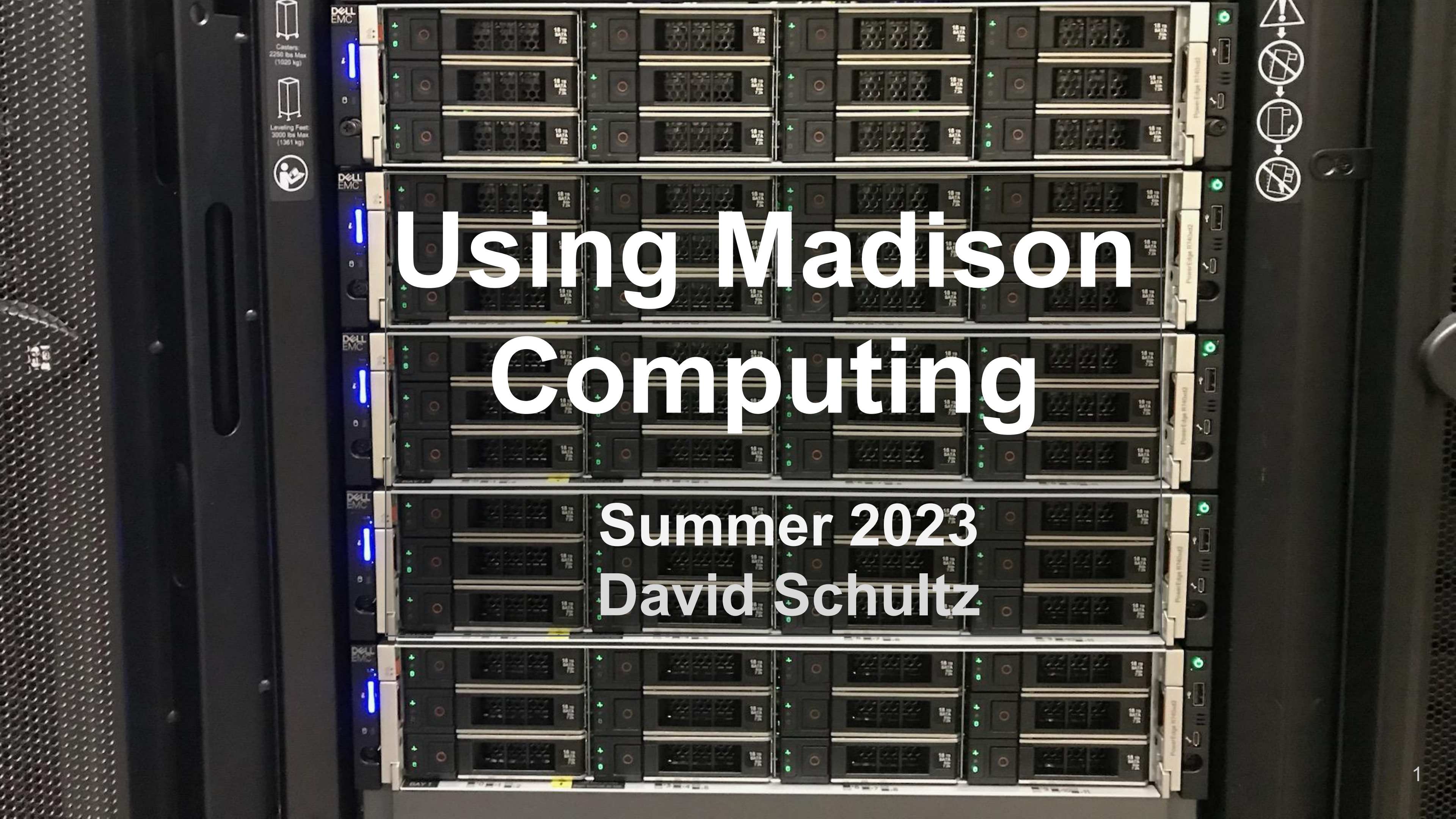


Using Madison Computing

Summer 2023
David Schultz

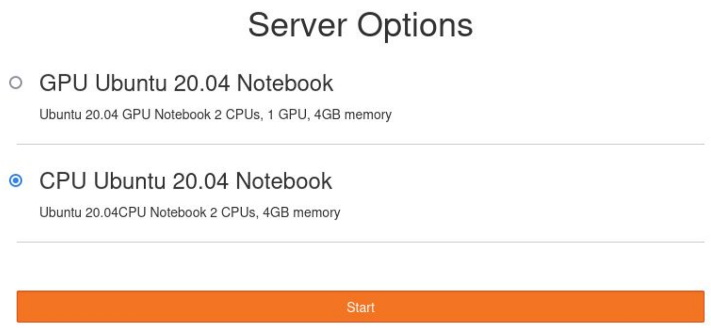
This session:
- Basic computing at Madison
- NPX / using HTCondor locally
  - Example exercise

Afternoon session:
- Grid computing
- Other advanced topics

JupyterHub

Python scripting in the browser, and more!

- jupyterhub.icecube.wisc.edu
  - fully browser based
  - no need to setup ssh tunnels
- CPU and GPU notebooks available
  - select CVMFS environment at runtime
  - all py3-v4 releases available
  - can create your own kernelspec with custom software
- NPX/Cobalt network filesystems available
  - /home, /data, /cvmfs
- More info on the wiki

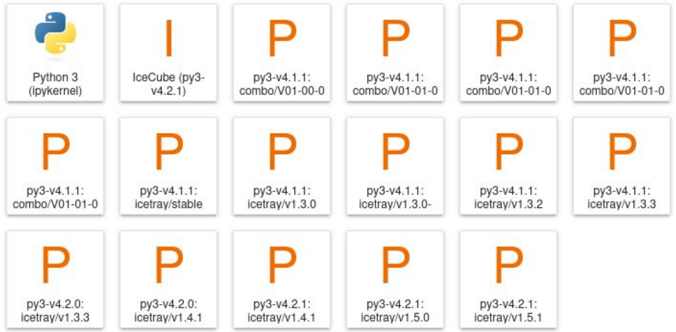


Server Options

○ GPU Ubuntu 20.04 Notebook
Ubuntu 20.04 GPU Notebook 2 CPUs, 1 GPU, 4GB memory

[✓ CPU Ubuntu 20.04 Notebook]
Ubuntu 20.04CPU Notebook 2 CPUs, 4GB memory

Start


Python 3 (ipykernel)
IceCube (py3-v4.2.1)
py3-v4.1.1: combo/V01-00-0
py3-v4.1.1: combo/V01-01-0
py3-v4.1.1: combo/V01-01-0
py3-v4.1.1: combo/V01-01-0
py3-v4.1.1: combo/V01-01-0
py3-v4.1.1: icetray/stable
py3-v4.1.1: icetray/v1.3.0
py3-v4.1.1: icetray/v1.3.0-
py3-v4.1.1: icetray/v1.3.2
py3-v4.1.1: icetray/v1.3.3
py3-v4.2.0: icetray/v1.3.3
py3-v4.2.0: icetray/v1.4.1
py3-v4.2.1: icetray/v1.4.1
py3-v4.2.1: icetray/v1.5.0
py3-v4.2.1: icetray/v1.5.1

SSH Access

While we will be using jupyterhub, many people prefer ssh via the terminal

Primary access via the command line goes through the pub gateway machines:

```
ssh pub.icecube.wisc.edu
```

If you want to directly connect to machines inside the network, do

```
ssh -Jpub.icecube.wisc.edu cobalt
```

This is a common enough pattern that many people add this to their .ssh/config file:

```
Host pub.icecube.wisc.edu data.icecube.wisc.edu
    ProxyCommand none
Host *.icecube.wisc.edu
    ProxyJump pub.icecube.wisc.edu
```

SSH Keys

Entering your password for every login gets tedious, but there is a better way.

1. Create a key: `ssh-keygen`
2. Copy the key: `ssh-copy-id pub.icecube.wisc.edu`
3. (optional) Add to ssh agent: `ssh-add`
4. (optional) Use key forwarding: in your .ssh/config, set

```
Host *.icecube.wisc.edu
    ForwardAgent yes
```

Cobalts

The cobalts are for interactive usage

- This does **not** mean run 20 processes in parallel - that is what HTCondor is for
- Original intent for cobalts was development, plots, light ad hoc computing

They are accessed via a common name:

```
ssh cobalt      (round robin for a pool of machines)
```

If you want an individual machine, you need to access it by number: `ssh cobalt01`

- This practice is generally not needed, except for accessing a specific /scratch
- Can cause issues if a specific machine is offline or has failed

Scratch space is unique to each machine, under /scratch

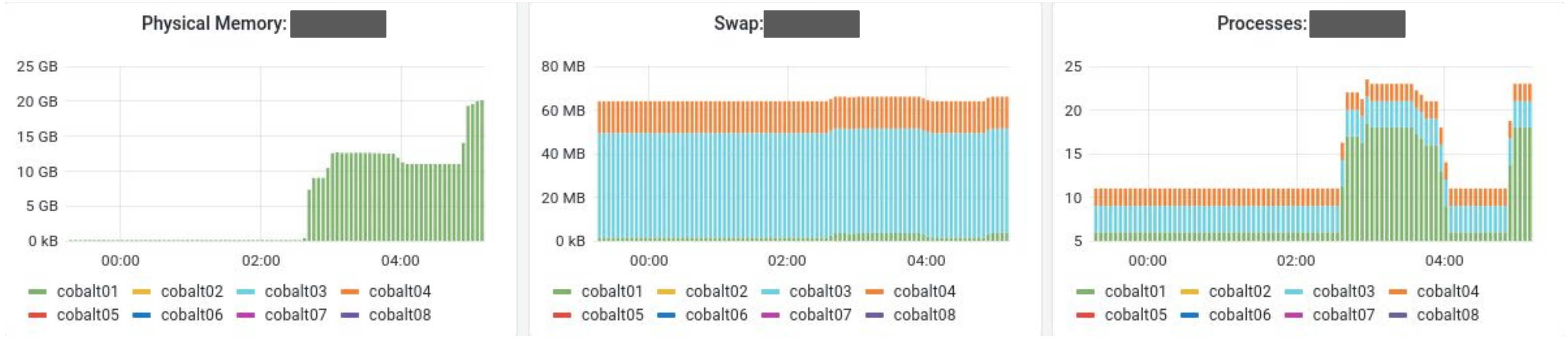
- Create a new directory /scratch/$USER to use
- It is meant for temporary (~30 days) files

Monitoring Cobalt Resource Use

You can monitor your resource usage on the cobalts via these techniques

- Command line tools for interactive use
  - top/htop, ps
  - valgrind, free
  - netstat, lsof

- Dashboard
  https://grafana.icecube.wisc.edu/grafana/d/RTSEgBumk/cobalt-resource-usage



Software Environments

By default, only basic system packages are installed

The common IceCube and ARA software environment is in CVMFS:

```
eval $(/cvmfs/icecube.opensciencegrid.org/py3-v4.2.1/setup.sh)
```

- This gives python 3.10, gcc 9.3, and many other software dependencies
- IceTray versions are available under `$SROOT/metaproject/*`

If you need to install custom Python packages:

- use a virtualenv
- install in your home directory with `pip install --user`

Getting Help

There are several ways to get help, based on the type of help desired:

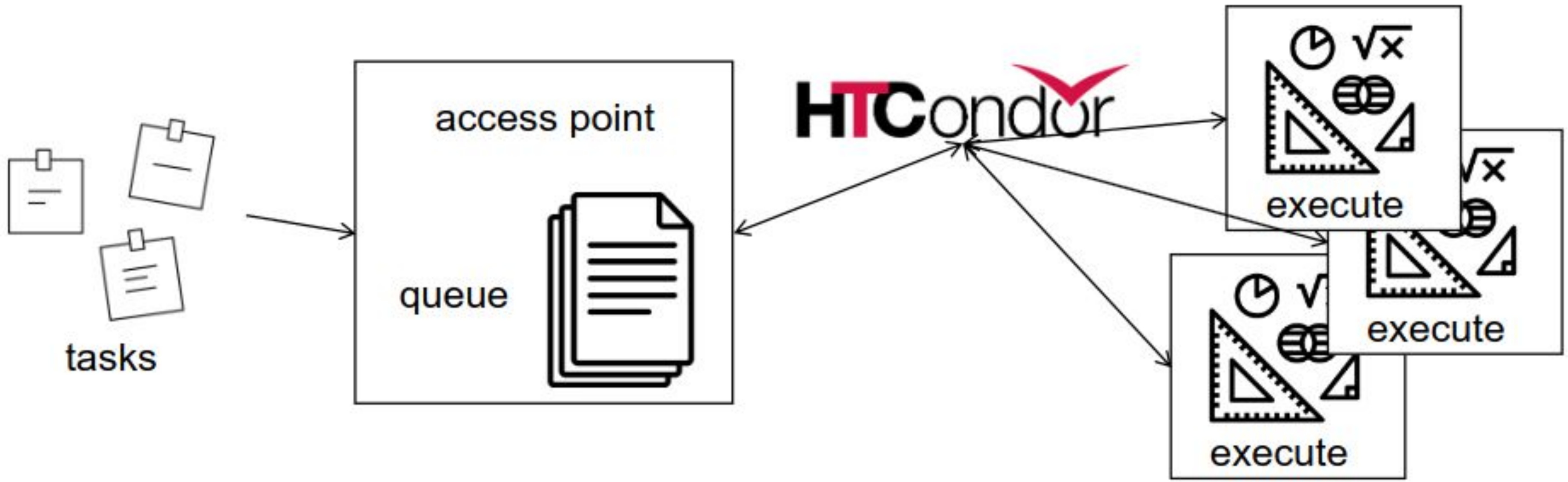
- Self-help:
  - For user account password resets
  - Account requests?
- Slack:
  - #software - help with any IceTray-related issue
  - #icecube-it - HTCondor or machine issues
  - #idontgetit - random questions (can be anonymous: /abot #idontgetit <msg>)
- Email:
  - help@icecube.wisc.edu - more complex issues, change requests

Batch / Parallel Computing

HTCondor is available for batch computing on multiple clusters

- NPX - our internal cluster at Madison
  - older CPUs and GPUs, direct access to /home and /data
  - “standard” resources are 1 CPU, 4 GB memory
  - can get an interactive “slot” with condor_submit -interactive
- “The grid”
  - sub-1 - combination of IceCube sites using pyglidein (deprecated)
  - sub-2 - many OSG sites, national computing allocations
  - many CPUs and GPUs available
  - “standard” resources are 1 CPU, 2 GB memory
    - asking for more than that will reduce the number of matching nodes

HTCondor Basics



HTCondor is all about running tasks in a distributed manner

Credit: Post It by Arianna Galimberti; papers by jabbar; Math by Turkkub from the Noun Project

HTCondor Basics

Basic HTCondor submit file:

- executable: a script or wrapper to run
- arguments: any cmdline arguments
- log: where to write the HTCondor job log
- output: where to write the job stdout
- error: where to write the job stderr
- notification: whether to send status emails
- transfer_input_files: send scripts to the job
- request_*: job resources
- queue N: number of jobs to run

```
executable = job.sh
arguments = physics.py

log = job.log
output = job.out
error = job.err
notification = never

transfer_input_files = physics.py

request_cpus = 1
request_memory = 100MB
request_disk = 1GB
#request_gpus = 1

queue 1
```

HTCondor Basics

Many jobs use a wrapper shell script to

- print some logging
- set the environment

```
submitter ~ $ cat job.sh
#!/bin/bash
set -e
printf "Start time: "; /bin/date
printf "Job is running on node: "; /bin/hostname
printf "Job is running in directory: "; /bin/pwd

eval $(/cvmfs/icecube.opensciencegrid.org/py3-v4.2.1/setup.sh)
$SROOT/metaprojects/icetray/v1.5.1/env-shell.sh python $@
echo "Job complete!"

submitter ~ $ chmod +x job.sh
```

```
executable = job.sh
arguments = physics.py

log = job.log
output = job.out
error = job.err
notification = never

transfer_input_files = physics.py

request_cpus = 1
request_memory = 100MB
request_disk = 1GB
#request_gpus = 1

queue 1
```

HTCondor Resource Limits

While batch jobs are a little flexible on resources (CPU, memory), they do enforce limits on usage. Be proactive and specify what your jobs need.

- CPU: in your submit file, use request_cpus=4
- Memory: in your submit file, use request_memory=8G
- GPUs: in your submit file, use request_gpus=2
- Time on “the grid”: in your submit file, use +OriginalTime=7200 (in seconds)

Estimating resource usage ahead of time:

- On a cobalt, run /usr/bin/time <my_cmd>, which will output:

```
14.86user 2.15system 0:17.17elapsed 99%CPU (0avgtext+0avgdata 4016196maxresident)k
177928inputs+0outputs (496major+1252999minor)pagefaults 0swaps
```

memory in KB

time

#/100 = CPUs
(this is 1 CPU)

Basic monitoring

After submitting a job

```
submitter ~ $ condor_submit job.sub

Submitting job(s).

1 job(s) submitted to cluster 12898721.
```

You can monitor it with

```
submitter ~ $ condor_q

-- Schedd: submit-1.icecube.wisc.edu : <128.104.255.232:9618?... @ 06/07/18 11:43:06

To monitor submitted jobs: condor_q

OWNER   BATCH_NAME       SUBMITTED   DONE   RUN    IDLE  TOTAL JOB_IDS

gmerino ID: 101524801   6/7  11:43       _      _       1      1 101524801.0

Total for query: 1 jobs; 0 completed, 0 removed, 1 idle, 0 running, 0 held, 0 suspended

Total for gmerino: 1 jobs; 0 completed, 0 removed, 1 idle, 0 running, 0 held, 0 suspended

Total for all users: 1736 jobs; 0 completed, 0 removed, 1548 idle, 188 running, 0 held, 0
suspended
```

Basic monitoring

```
submitter ~ $ condor_q -nobatch

-- Schedd: submit-1.icecube.wisc.edu : <128.104.255.232:9618?... @ 06/07/18 11:50:11

 ID             OWNER                SUBMITTED      RUN_TIME ST PRI SIZE CMD

101524801.0    gmerino              6/7  11:49    0+00:00:00 I  0     0.0 job.sh 10
```

ST = status

Most Common Job status:

- Idle “I”: Job has not started yet… waiting in queue
- Running “R”: job is currently running
- Completed “C”: If the job has completed, it will not appear in condor_q
- Held “H”: Stalled jobs. Something you need to fix

A job that goes on hold is interrupted (all progress is lost) and kept from running again.
It remains in the queue in the “H” state.

Basic monitoring

Jobs on hold look like this:

```
submitter ~ $ condor_q -hold

189123084.0   USER            6/7  14:58 Error from slot1_6@n-43.icecube.wisc.edu:
Policy violation. Execution time limit exceeded: 2+00:00:03 > 2+00:00:00.

189251539.0   USER            6/7  09:13 Error from slot1_24@c9-16.icecube.wisc.edu:
Policy violation. Memory limit exceeded: requested 4096MB, used 5151MB.
```

Other reasons are:

- CPU limit exceeded
- Local storage limit exceeded
- File transfer errors

Basic monitoring - Log file

```
submitter ~ $ cat job.log
000 (12898721.000.000) 06/13 21:50:14 Job submitted from host:
<128.104.255.232:58276?addrs=128.104.255.232-58276>
...
001 (12898721.000.000) 06/13 21:52:33 Job executing on host:
<144.92.166.137:27680?addrs=144.92.166.137-27680>
...
006 (12898721.000.000) 06/13 21:52:33 Image size of job updated: 1
      0  -  MemoryUsage of job (MB)
      0  -  ResidentSetSize of job (KB)
...
005 (12898721.000.000) 06/13 21:52:43 Job terminated.
      (1) Normal termination (return value 0)
            Usr 0 00:00:00, Sys 0 00:00:00  -  Run Remote Usage
            Usr 0 00:00:00, Sys 0 00:00:00  -  Run Local Usage
            Usr 0 00:00:00, Sys 0 00:00:00  -  Total Remote Usage
            Usr 0 00:00:00, Sys 0 00:00:00  -  Total Local Usage
      314  -  Run Bytes Sent By Job
      281  -  Run Bytes Received By Job
      314  -  Total Bytes Sent By Job
      281  -  Total Bytes Received By Job
      Partitionable Resources :    Usage  Request Allocated
          Cpus :                                1         1
          Disk (KB) :                 12   102400    940361
          Memory (MB) :                0      100       100
```

Debugging - Finding Job Attributes

```
submitter ~ $ condor_q -long 12887688.0
JobStatus = 2
LastJobStatus = 1
User = “gmerino@icecube.wisc.edu”
Err = "/scratch/gmerino/Data/2013/logs/12887688.err"
Out = "/scratch/gmerino/Data/2013/logs/12887688.log" NumJobStarts = 1
Args = "-g
/data/exp/IceCube/2013/filtered/level2/0505/Run00122300/Level2_IC86.2013_data_Run00122300_0505_0_9_GCD.i3.gz -i
/data/exp/IceCube/2013/filtered/level2/0505/Run00122300/Level2_IC86.2013_data_Run00122300_Subrun00000070.i3.bz2 -o
/data/ana/Cscd/StartingEvents/exp/IC86_2013/burnsample/l3/00122300/Level2_IC86.2013_data_Run00122300_Part00000070.i3.bz2”
RemoteHost = “slot1@e281.chtc.wisc.edu"
ResidentSetSize_RAW = 1200308
DiskUsage_RAW = 891690
RemoteUserCpu = 7669.0
```

Debugging - Useful Job Attributes

JobStatus: number indicating Idle (1), Running (2), Held (5), etc

RemoteHost: where the job is running

ResidentSetSize_RAW: Maximum observed physical memory in use by the job in KiB while running.

DiskUsage_RAW: Maximum observed physical memory in use by the job in KiB while running.

RemoteUserCpu: The total number of seconds of user CPU time the job has used.

EnteredCurrentStatus: time of last status change

NumJobStarts: number of times the job started executing

https://htcondor.readthedocs.io/en/latest/classad-attributes/job-classad-attributes.html

Debugging - Displaying Job Attributes

Use the “-autoformat” option for condor_q

```
submitter $ condor_q -af JobStatus ClusterID ProcId RemoteHost ResidentSetSize_RAW
2 12892531 0 slot1@glidein_235900_283164684@cabinet-0-0-7.t2.ucsd.edu 1272208
2 12892986 0 glidein_10345_623188368@jux7c.zeuthen.desy.de 1290364
2 12893002 0 slot1@e137.chtc.wisc.edu 1181296
```

The “-constraint” option can also be handy

```
submitter $ condor_q --constraint 'JobStatus =?= 2' -af JobStatus ClusterID ProcId
RemoteHost ResidentSetSize_RAW
2 12892531 0 slot1@glidein_235900_283164684@cabinet-0-0-7.t2.ucsd.edu 1272208
2 12892986 0 glidein_10345_623188368@jux7c.zeuthen.desy.de 1290364
2 12893002 0 slot1@e137.chtc.wisc.edu 1181296
```

Debugging - Displaying Machine Attributes

Sometimes you want to know more information about an execution machine

```
dschultz@submit-1 ~ $ condor_status -constraint 'GPUs && PartitionableSlot' -af Machine
CPUs GPUs Memory Disk CUDADeviceName
gtx-7.icecube.wisc.edu 18 2 12674 702621821 NVIDIA GeForce GTX 1080
gtx-13.icecube.wisc.edu 16 2 12545 542442949 NVIDIA GeForce GTX 980
gtx-16.icecube.wisc.edu 21 1 7041 698444797 NVIDIA GeForce GTX 980
gtx-17.icecube.wisc.edu 13 1 12545 537691665 NVIDIA GeForce GTX 980
gtx-24.icecube.wisc.edu 12 3 4017 684190814 NVIDIA GeForce GTX 980
gtx-29.icecube.wisc.edu 26 2 27520 704518999 NVIDIA GeForce GTX 980
gtx-31.icecube.wisc.edu 22 4 35712 698451190 NVIDIA GeForce GTX 980
gtx-32.icecube.wisc.edu 21 2 12800 709486790 NVIDIA GeForce GTX 980
gtx-36.icecube.wisc.edu 17 1 1968 690525880 NVIDIA GeForce GTX 980
gtx-41.icecube.wisc.edu 5 1 4096 225299905 NVIDIA GeForce GTX 980
```

NPX Tips

- Use the provided scratch space
- Logfiles in network filesystems (/home, /data/user) can generate instability and are prohibited
  - Keep your job logfiles on local disk (typically /scratch/$USER)
- If you want to use file transfer, set the FileSystemDomain
- Job duration is limited to 48 hours
  - If you want more, specify the 1_week or 2_week accounting groups
- If you want to run interactively, use `condor_submit -interactive`

```
executable = job.sh
arguments = physics.py

log = /scratch/$ENV(USER)/job.log
output = job.out
error = job.err
notification = never

+FileSystemDomain = "foo"
transfer_input_files = physics.py

request_cpus = 1
request_memory = 100MB
request_disk = 1GB
#request_gpus = 1+
AccountingGroup="1_week.$ENV(USER)"

queue 1
```

NPX Exercise

Tell me the number of events in the first 10 regular i3 files in this directory:
/data/exp/IceCube/2018/filtered/level2/0608/Run00131134

- Here are some hints to get you started:

```
submitter ~ $ cat job.sh
#!/bin/bash
set -e
printf "Start time: "; /bin/date
printf "Job is running on node: "; /bin/hostname
printf "Job is running in directory: "; /bin/pwd

eval $(/cvmfs/icecube.opensciencegrid.org/py3-v4.2.1/setup.sh)
$SROOT/metaprojects/icetray/v1.5.1/env-shell.sh python $@
echo "Job complete!"

submitter ~ $ chmod +x job.sh
```

```
executable = job.sh
arguments = physics.py $(ProcId)

log = job.log
output = job.out.$(ProcId)
error = job.err.$(ProcId)
notification = never

+FileSystemDomain = "foo"
transfer_input_files = physics.py

request_cpus = 1
request_memory = 1GB
request_disk = 1GB

queue 10
```

NPX Exercise

Tell me the number of events in the first 10 regular i3 files in this directory:
/data/exp/IceCube/2018/filtered/level2/0608/Run00131134

If you want bonus points (or are bored)

- How many P frames?
- How many Q frames pass the MuonFilter_13
- How many Q frames have a charge greater than 1000 PE

Afternoon session:

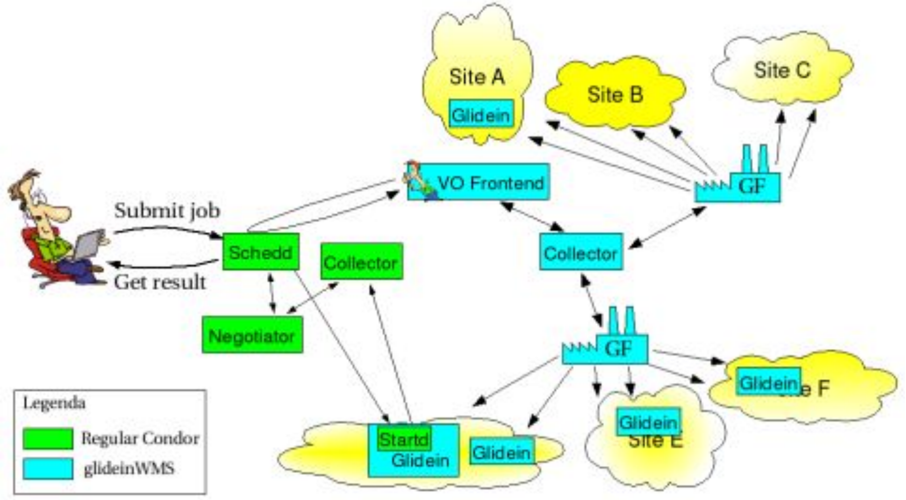
- Grid computing
- Other advanced topics

Grid Computing

IceCube has access to many external resources

- CHTC - large UW campus cluster
- IceCube sites:
  - Aachen, Canada, DESY, Dortmund, Harvard, KIT, Maryland, Michigan, NBI, …
- Supercomputer allocations:
  - Anvil, Bridges, Delta, Expanse
- Open Science Grid, National Research Platform

Easily 10k+ CPUs, 500+ GPUs



Grid Computing

Access is provided at sub-2.icecube.wisc.edu

- sub-1 is deprecated, and will be removed “soonish”

The Grid also uses HTCondor, but does not have access to the /data directories (sim, exp, ana, user), so files must be transferred

- /home is mounted on sub-2, but not accessible from your jobs

https://wiki.icecube.wisc.edu/index.php/Condor/Grid

Grid Computing - Data Transfer

1. Built-in file transfer - for scripts and small files

```
transfer_input_files = physics.py,tables.dat,/home/$ENV(USER)/my_scripts
transfer_output_files = out.hdf5
transfer_output_remaps = "out.hdf5 = out.$(ClusterId).$(ProcId).hdf5  "
```

Here we transfer two input files and a whole input directory

Then we transfer out a small hdf5 file with our results. This gets saved to the working directory we submit from, usually on /scratch on sub-2

We also rename the output file with the id number of this job, in case we submitted multiple jobs and job sets

- Note: when using remaps, it is an error for the file to not exist

Grid Computing - Data Transfer

2. GridFTP file transfer - the old way of handling large data

```
#!/bin/bash
set -e
eval $(/cvmfs/icecube.opensciencegrid.org/py3-v4.2.1/setup.sh)
infile = $1
shift;
outfile = $1
shift;
echo “transferring input $infile”
globus-url-copy gsiftp://gridftp.icecube.wisc.edu/$infile file:/$PWD/infile

echo “running program”
$SROOT/metaprojects/icetray/v1.5.1/env-shell.sh python physics.py
infile outfile $@

echo “transferring output $outfile”
globus-url-copy file:/$PWD/outfile
gsiftp://gridftp.icecube.wisc.edu/$outfile

echo "Job complete!"
```

```
transfer_input_files = physics.py, x509
transfer_output_files =
arguments = /data/user/me/my_file.i3
/data/user/me/out_file.i3 --prog args
```

```
dschultz@sub-2 $ grid-proxy-init -valid 24:0 -out x509
Your identity: /DC=org/DC=cilogon/C=US/O=University of
Wisconsin-Madison/CN=David Schultz B47305562
Enter GRID pass phrase for this identity:
Creating proxy
..................................................................................
..................................................................................
....... Done
Your proxy is valid until: Thu Jun  8 21:22:03 2023

dschultz@sub-2 $ condor_submit job.sub
```

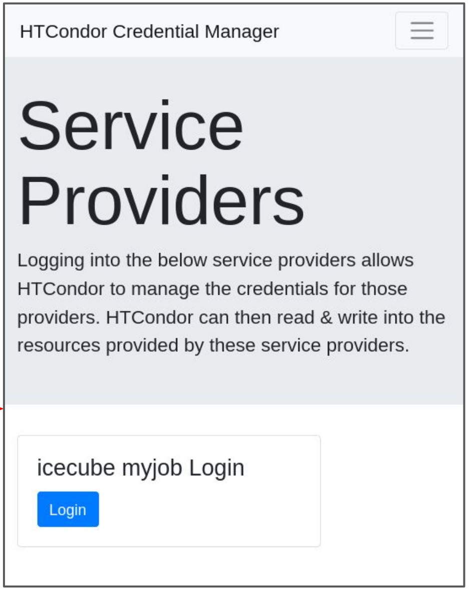
Grid Computing - Data Transfer

3. HTTP file transfer - the **new** way of handling large data

```
# required lines for http transfers
use_oauth_services = icecube
icecube_oauth_permissions_myjobs = offline_access

transfer_input_files =
physics.py,icecube.myjobs+https://data.icecube.aq/data/user/me/my_input.i3
transfer_output_files = out.hdf5
transfer_output_remaps = "out.hdf5 =
icecube.myjobs+https://data.icecube.aq/data/user/me/out.$(ClusterId).$(ProcId).hdf5"
```

```
dschultz@sub-2:~$ condor_submit job.sub
Submitting job(s)
Hello, dschultz.
Please visit:
http://localhost:22280/key/5b2dfca80ec4b5ebce55c40b114c40ab57290
3171e37efba065de29a2789999e
```


HTCondor Credential Manager

Service Providers

Logging into the below service providers allows HTCondor to manage the credentials for those providers. HTCondor can then read & write into the resources provided by these service providers.

icecube myjob Login

After logging in, `condor_submit` will work

Grid Computing - Pitfalls

The grid is very dynamic - machines can leave at any time and “evict” your job

- The job should restart from the beginning elsewhere

Resource limits can be stricter

- Other resource owners may have a hard cutoff in terms of memory, and kill your job if it exceeds the requested amount for even a moment

Time limits are generally smaller

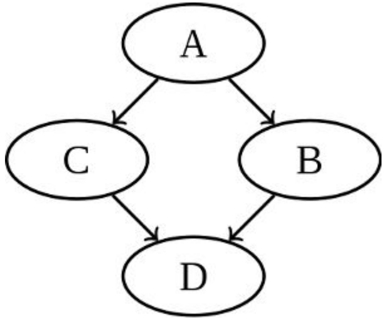
- The ideal runtime for a job is 2-4 hours, and anything greater than a day will only run on a handful of machines
- Request runtime by specifying `+OriginalTime=7200` (a value in seconds)
- The default is 1 hour

DAGMan

DAGMan is a tool that comes bundled with HTCondor. It can do several useful things:

- Control the number of queued and running jobs
- Handle inter-job dependencies
- Save the state of a “run” - which jobs completed and failed - so you can restart it

DAGMan = Directed Acyclic Graph Manager.



https://htcondor.readthedocs.io/en/latest/users-manual/dagman-workflows.html

DAGMan

Let's make a basic DAG submit file and a regular submit file:

```
# file name: dagman.submit
JOB job1 job.condor
VARS job1 Filenum="001"
JOB job2 job.condor
VARS job2 Filenum="002"
JOB job3 job.condor
VARS job3 Filenum="003"
JOB job4 job.condor
VARS job4 Filenum="004"
```

```
# file name: job.condor
# special variables:
#  Filenum = Filenum var defined in dagman.submit
Executable = job.sh
Arguments = physics.py $(Filenum)

output = job.$(ProcId).out
error = job.$(ProcId).err
log = job.log

notification = never

queue
```

DAGMan

We can submit this, limiting it to 2 active (idle + running) jobs:

```
dschultz@submitter ~/test/dagman $ condor_submit_dag -maxjobs 2 dagman.submit


-----------------------------------------------------------------------
File for submitting this DAG to Condor           : dagman.submit.condor.sub
Log of DAGMan debugging messages                 : dagman.submit.dagman.out
Log of Condor library output                     : dagman.submit.lib.out
Log of Condor library error messages             : dagman.submit.lib.err
Log of the life of condor_dagman itself          : dagman.submit.dagman.log

Submitting job(s).
1 job(s) submitted to cluster 21135967.
-----------------------------------------------------------------------
```

DAGMan

One nice feature is the dagman output file, which updates regularly with the current status of the run:

```
...
06/02/14 14:40:49 Of 4 nodes total:
06/02/14 14:40:49  Done   Pre    Queued    Post    Ready    Un-Ready   Failed
06/02/14 14:40:49   ===   ===        ===   ===   ===        ===        ===
06/02/14 14:40:49   0     0          2     0     2          0          0
06/02/14 14:40:49 0 job proc(s) currently held
06/02/14 14:40:49 Note: 2 total job deferrals because of -MaxJobs limit (2)
...
```

DAGMan - Job Dependencies

Now let us look at an example with dependencies, where one job must run before another one.

Let's make a dag with 3 parents and one child (maybe processing and cleanup?):

```
# file name: dagman.submit
JOB job1 job.condor
VARS job1 Filenum="001"
JOB job2 job.condor
VARS job2 Filenum="002"
JOB job3 job.condor
VARS job3 Filenum="003"
JOB job4 job.condor
VARS job3 Filenum="004"
# define the DAG relationship
Parent job1 job2 job3 Child job4
```

DAGMan - Job Dependencies

If we submit this, we can see the child job is un-ready until the three parents have finished:

```
...
06/02/14 15:45:31 Of 4 nodes total:
06/02/14 15:45:31  Done   Pre   Queued   Post   Ready   Un-Ready   Failed
06/02/14 15:45:31   ===   ===       ===   ===   ===       ===        ===
06/02/14 15:45:31   0     0         2     0     1         1          0
06/02/14 15:45:31 0 job proc(s) currently held
06/02/14 15:45:31 Note: 1 total job deferrals because of -MaxJobs limit (2)
...
06/02/14 15:45:56 Of 4 nodes total:
06/02/14 15:45:56  Done   Pre   Queued   Post   Ready   Un-Ready   Failed
06/02/14 15:45:56   ===   ===       ===   ===   ===       ===        ===
06/02/14 15:45:56   3     0         0     0     1         0          0
06/02/14 15:45:56 0 job proc(s) currently held
...
```

DAGMan - Restarting

If a dag encounters errors, it will finish the jobs it can, then write out a rescue file next to the log files

To rerun the dag, just submit it again. It will auto-detect the rescue file and continue from where it left off

This is useful if you need to update one of your scripts to fix a bug

Containers

It is possible to run jobs inside containers with HTCondor

```
# file name: job.condor
# special variables:
#  Filenum = Filenum var defined in dagman.submit
Executable = job.sh
Arguments = physics.py $(Filenum)

output = job.$(ProcId).out
error = job.$(ProcId).err
log = job.log

# use an OSG EL7 image for consistency
+SingularityImage="/cvmfs/singularity.opensciencegrid.org/opensciencegrid/osgvo-el7:latest"

notification = never

queue
```

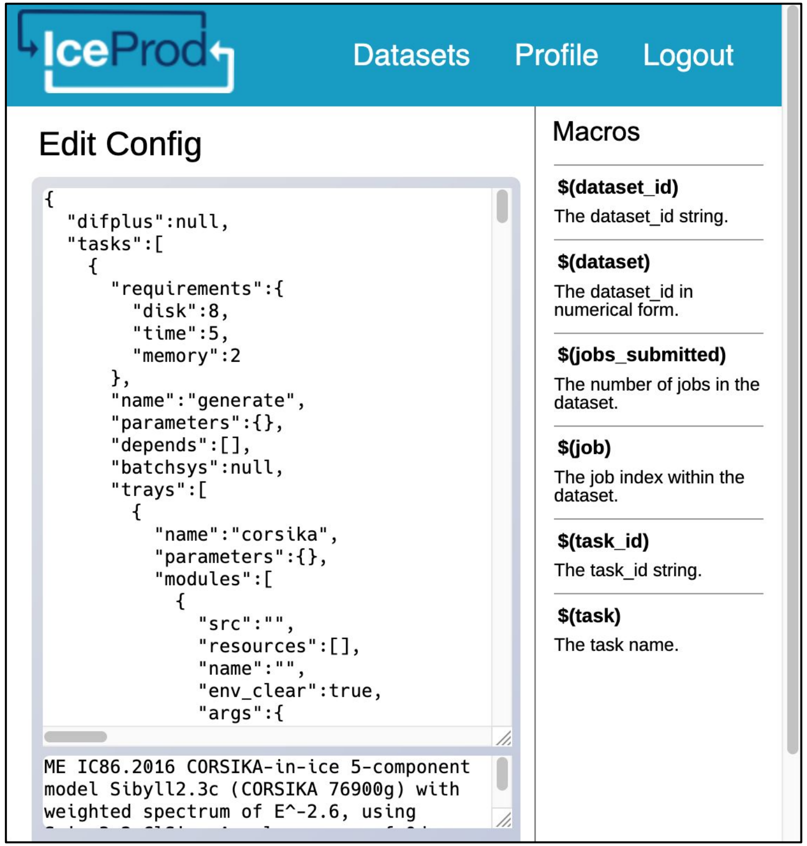
IceProd

IceCube-specific scheduler for the grid

- Used by simulation production and working groups to create official datasets
- Describe jobs to run using json
- Handles file transfers to data warehouse
- Auto retries for site errors
- Uses web interface

Available for any IceCube user

Talk to me if you want to use it



Edit Config

```
{
  "difplus":null,
  "tasks":[
    {
      "requirements":{
        "disk":8,
        "time":5,
        "memory":2
      },
      "name":"generate",
      "parameters":{},
      "depends":[],
      "batchsys":null,
      "trays":[
        {
          "name":"corsika",
          "parameters":{},
          "modules":[
            {
              "src":"",
              "resources":[],
              "name":"",
              "env_clear":true,
              "args":{
```

```
ME IC86.2016 CORSIKA-in-ice 5-component
model Sibyll2.3c (CORSIKA 76900g) with
weighted spectrum of E^-2.6, using
```

Macros

$(dataset_id)
The dataset_id string.

$(dataset)
The dataset_id in numerical form.

$(jobs_submitted)
The number of jobs in the dataset.

$(job)
The job index within the dataset.

$(task_id)
The task_id string.

$(task)
The task name.

Random Questions

My job hasn't started running yet. Why not?

If the queue is full you may need to wait up to an hour for your job to start. Also, if you have been running lots of other jobs your priority may be lower than other users

You can check your priority with `condor_userprio`

If you think your job should be running and it isn't, then debugging can start. First, find the ID of the job. Then run `condor_q -better-analyze` on that ID

Random Questions

condor_q fails

condor_q or other condor commands are failing with an error like:

```
-- Failed to fetch ads from: <10.128.12.110:40381> : submitter.icecube.wisc.edu
CEDAR:6001:Failed to connect to <10.128.12.110:40381>
```

HTCondor is likely overloaded, so stop trying to ask it things. Wait 5 minutes and try again

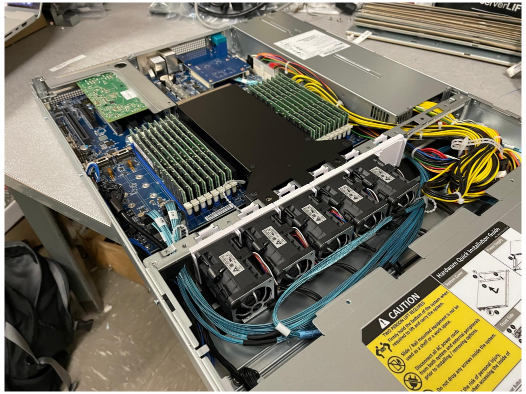
If it fails to work for 30 minutes, then there might be a real problem. Ask on slack at #icecube-it or email help@icecube.wisc.edu with the error message

More Tutorials

Last year’s HTCondor conference had several tutorial talks (+ recordings):

- Basic submission in more detail
- DAGMan details
- Organizing and submitting large workloads
- Pybindings for HTCondor

https://htcondor.org/event_summary/htcondor_week_2022



Other questions?