UNIVERSITY OF NOTRE DAME

CMS connect

Introduction to HTCondor

Kenyi Hurtado

June 17, 2016

Covered in this presentation

- What is HTCondor?
- How to run a job (and multiple ones)
- Monitoring your queue

What is HTCondor?

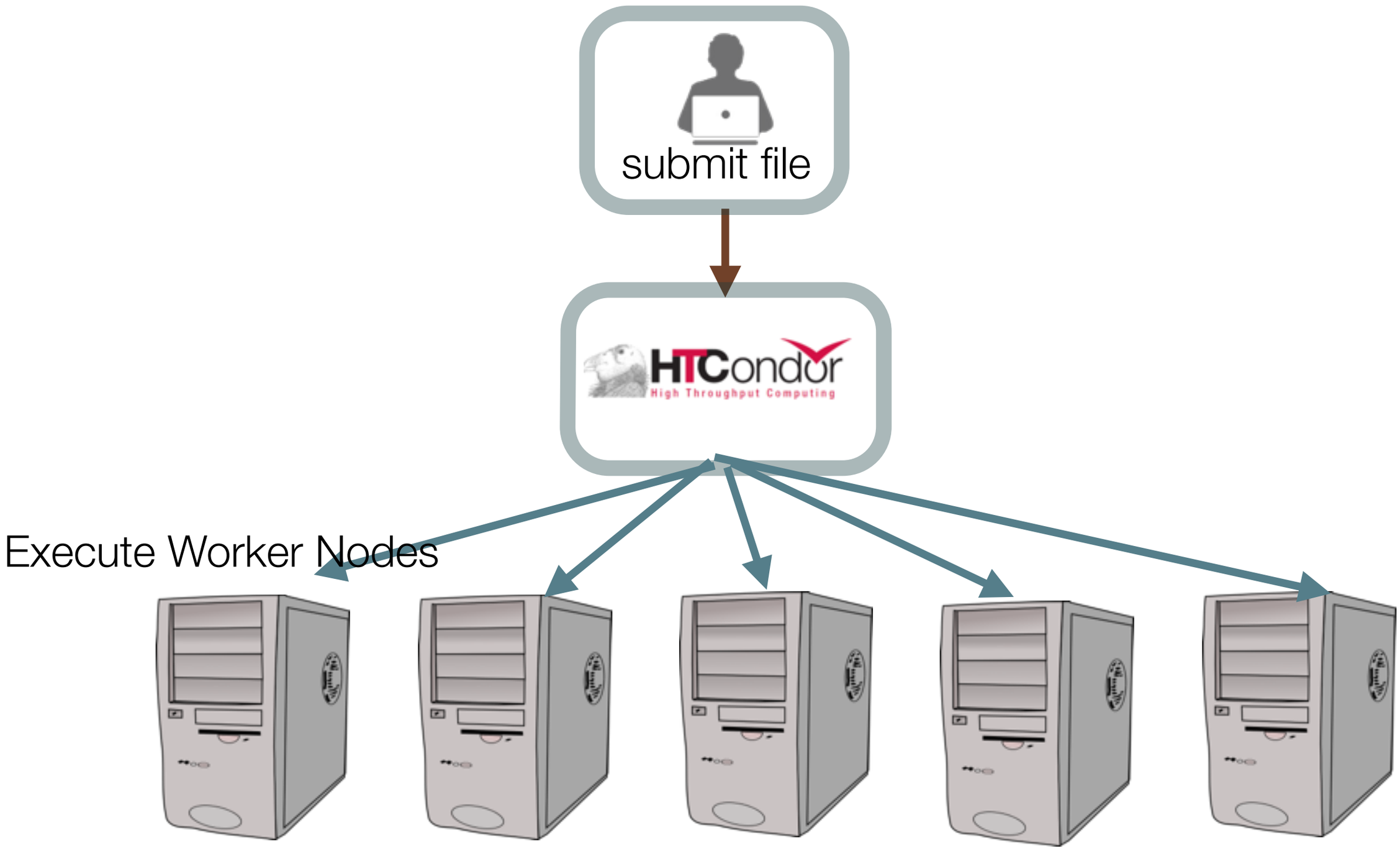
- A specialized workload management system
- Can schedule and run computing tasks on your behalf

Wait, that sounds like CRAB!

- Yes, in a way… CRAB is a tool to submit cmsRun tasks.
- Under the hoods, it creates condor (DAGman) jobs though….

The vary basic idea…

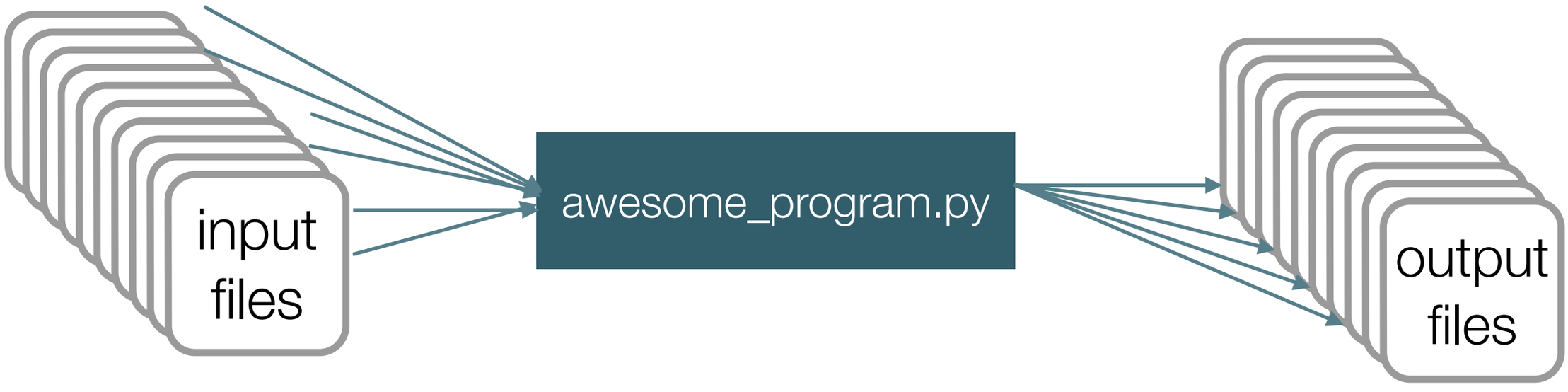
Very simplistic schematic



Running jobs with HTCondor

HTCondor Jobs

- A "job" is a single computing task
  - Let’s assume you have one program you need to run with 100 different inputs, hence producing 100 different outputs. **That would be an example for 100 jobs.**



A simple example

Bash script example to run

```
#!/bin/bash
#program.sh

printf "Start running job at: "; /bin/date
printf "Worker node hostname: "; /bin/
hostname
echo "----------------------------------------"
echo  "Doing hard work..."
sleep ${1-15}
echo "Task completed!"
printf "Finished at: "; /bin/date
```

Take 1 argument or default to 15s if no argument is provided

A simple example

Condor submit file

```
#submit.jdl
Universe = vanilla
Executable = program.sh
Arguments = 30
Log = log/job.log.$(Cluster)
Error = log/job.err.$(Cluster)-$(Process)
Output = log/job.out.$(Cluster)-$(Process)

Queue 1
```

A simple example

Condor submit file

```
#submit.jdl
Universe = vanilla
Executable = program.sh
Arguments = 30
Error = log/job.err.$(Cluster)-$(Process)
Output = log/job.out.$(Cluster)-$(Process)
Log = log/job.log.$(Cluster)

Queue 1
```

There are different “universes” that define the way HTCondor handle these jobs. You will use “vanilla" for CMS Connect.

A simple example

Condor submit file

```
#submit.jdl
Universe = vanilla
Executable = program.sh
Arguments = 30
Log = log/job.log.$(Cluster)
Error = log/job.err.$(Cluster)-$(Process)
Output = log/job.out.$(Cluster)-$(Process)

Queue 1
```

These are the executable and arguments for it

A simple example

Condor submit file

```
#submit.jdl
Universe = vanilla
Executable = program.sh
Arguments = 30
Log = log/job.log.$(Cluster)
Error = log/job.err.$(Cluster)-$(Process)
Output = log/job.out.$(Cluster)-$(Process)

Queue 1
```

These will write a logfile, an errorfile and the output for each job.

$(Cluster), $(Process) are internal condor variables

Cluster = The id of the set of queued jobs you submitted

Process = Each job has its unique process Id within a cluster of jobs.

A simple example

Condor submit file

```
#submit.jdl
Universe = vanilla
Executable = program.sh
Arguments = 30
Error = log/job.err.$(Cluster)-$(Process)
Output = log/job.out.$(Cluster)-$(Process)
Log = log/job.log.$(Cluster)

Queue 1
```

This setting defines how many jobs you want to submit

A simple example

Use **condor_submit** to submit your job

$ condor_submit submit.jdl

```
Start running job at: Thu Jun 16 06:08:12
CDT 2016
Worker node hostname:
g28n01.hep.wisc.edu
----------------------------------------
Doing hard work...
Task completed!
Finished at: Thu Jun 16 06:08:42 CDT 2016
```

Transfering Files


```
Universe = vanilla
Executable = pyROOTscript.py
Arguments = input.root
Error = log/job.err.$(Cluster)-$(Process)
Output = log/job.out.$(Cluster)-$(Process)
Log = log/job.log.$(Cluster)

should_transfer_files = YES
transfer_input_files = input.root
when_to_transfer_output = ON_EXIT
transfer_output_files = output.root

Queue 1
```

Enable HTCondor transfer mechanism

Transfering Files


```
Universe = vanilla
Executable = pyROOTscript.py
Arguments = input.root
Error = log/job.err.$(Cluster)-$(Process)
Output = log/job.out.$(Cluster)-$(Process)
Log = log/job.log.$(Cluster)

should_transfer_files = YES
transfer_input_files = input.root
when_to_transfer_output = ON_EXIT
transfer_output_files = output.root

Queue 1
```

Specify when to transfer output files. In this case after completion of job.

Transfering Files


```
Universe = vanilla
Executable = pyROOTscript.py
Arguments = input.root
Error = log/job.err.$(Cluster)-$(Process)
Output = log/job.out.$(Cluster)-$(Process)
Log = log/job.log.$(Cluster)

should_transfer_files = YES
transfer_input_files = input.root
when_to_transfer_output = ON_EXIT_OR_EVICT
transfer_output_files = output.root

Queue 1
```

Can transfer even if job is evicted too… useful for fault tolerant jobs that can restart where they left

Transfering Files


```
Universe = vanilla
Executable = pyROOTscript.py
Error = log/job.err.$(Cluster)-$(Process)
Output = log/job.out.$(Cluster)-$(Process)
Log = log/job.log.$(Cluster)
should_transfer_files = YES
when_to_transfer_output = ON_EXIT

Arguments = input1.root
transfer_input_files = input1.root
transfer_output_remaps = "output.root = output/output1.root”
Queue

Arguments = input2.root
transfer_input_files = input2.root
transfer_output_remaps = "output.root = output/output2.root”
Queue
```

Makes 2 jobs.
“Queue" without numbers is equivalent to “Queue 1"

Transfering Files


```
Universe = vanilla
Executable = pyROOTscript.py
Error = log/job.err.$(Cluster)-$(Process)
Output = log/job.out.$(Cluster)-$(Process)
Log = log/job.log.$(Cluster)
should_transfer_files = YES
when_to_transfer_output = ON_EXIT

Arguments = input1.root
transfer_input_files = input1.root
transfer_output_remaps = "output.root = output/output1.root”
Queue

Arguments = input1.root
transfer_input_files = input2.root
transfer_output_remaps = "output.root = output/output2.root”
Queue
```

Since our script always produces “output.root”, we can rename it to avoid overwriting

Submiting and monitoring your jobs

```
$ condor_submit submit.jdl
Submitting job(s)..
2 job(s) submitted to cluster 482482.
```

Look at the cluster ID

Submiting and monitoring your jobs

Use **condor_q** to monitor your jobs

```
$ condor_q 482482
-- Schedd: login.uscms.org : <192.170.227.118:9618?…
 ID   OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
482482.0   khurtado        6/16 20:11   0+00:00:00 I  0   0.0  pyROOTscript.py input1.root
482482.1   khurtado        6/16 20:11   0+00:00:00 I  0   0.0  pyROOTscript.py input2.root

2 jobs; 0 completed, 0 removed, 2 idle, 0 running, 0 held, 0 suspended
```

Submit node

job.submit
pyROOTscript.py
input1.root
input2.root
log/
    job.log.482482
    job.{out,err}.482482-{0,1}

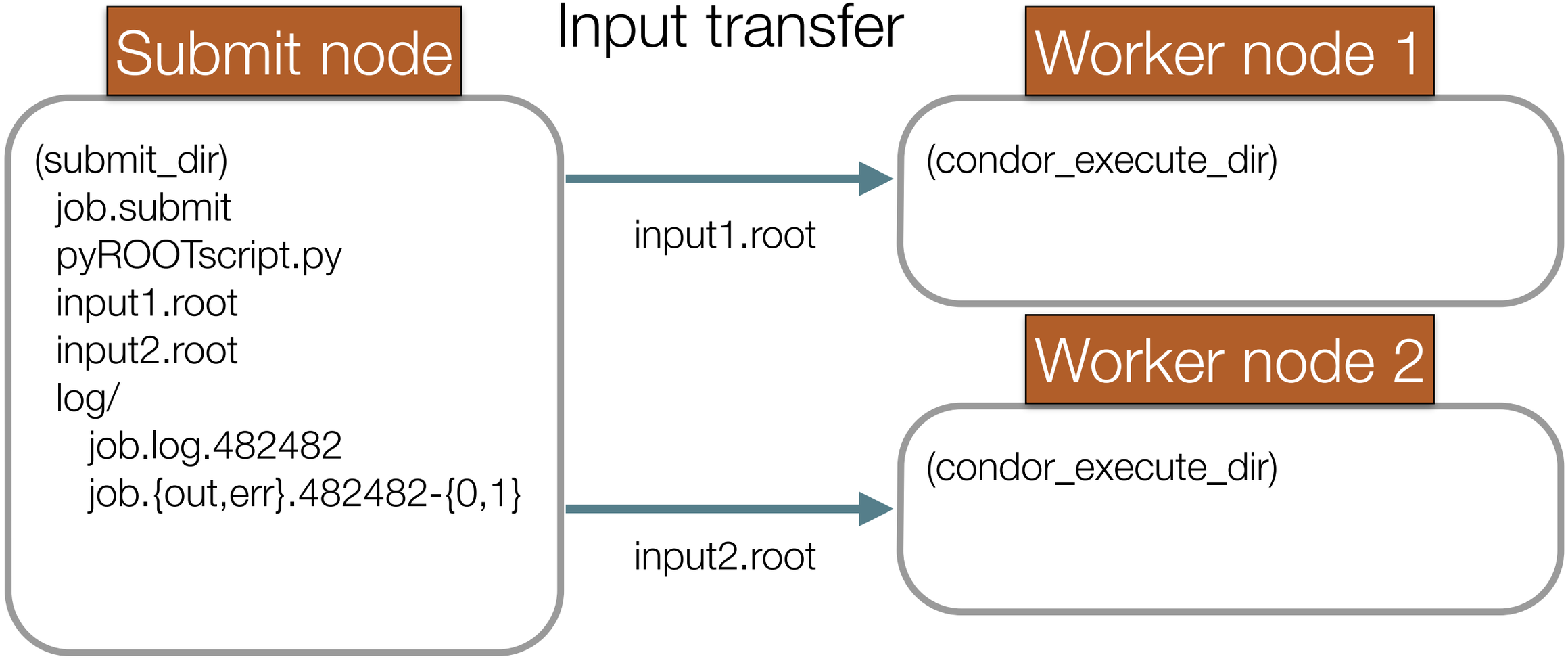
(No worker node assigned yet)

<Your job is waiting in queue>

Submiting and monitoring your jobs

```
$ condor_q 482482
-- Schedd: login.uscms.org : <192.170.227.118:9618?…
 ID    OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
482482.0   khurtado        6/16 20:11   0+00:00:00 <  0   0.0  pyROOTscript.py input1.root
482482.1   khurtado        6/16 20:11   0+00:00:00 <  0   0.0  pyROOTscript.py input2.root

2 jobs; 0 completed, 0 removed, 0 idle, 2 running, 0 held, 0 suspended
```



Submiting and monitoring your jobs

```
$ condor_q 482482
-- Schedd: login.uscms.org : <192.170.227.118:9618?…
 ID    OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
482482.0   khurtado        6/16 20:11   0+00:00:00 R  0   0.0  pyROOTscript.py input1.root
482482.1   khurtado        6/16 20:11   0+00:00:00 R  0   0.0  pyROOTscript.py input2.root

2 jobs; 0 completed, 0 removed, 0 idle, 2 running, 0 held, 0 suspended
```

Running

Submit node

(submit_dir)
- job.submit
- pyROOTscript.py
- input1.root
- input2.root
- log/
  - job.log.482482
  - job.{out,err}.482482-{0,1}

Worker node 1

(condor_execute_dir)
- pyROOTscript.py
- input1.root
- output.root
- stderr
- stdout

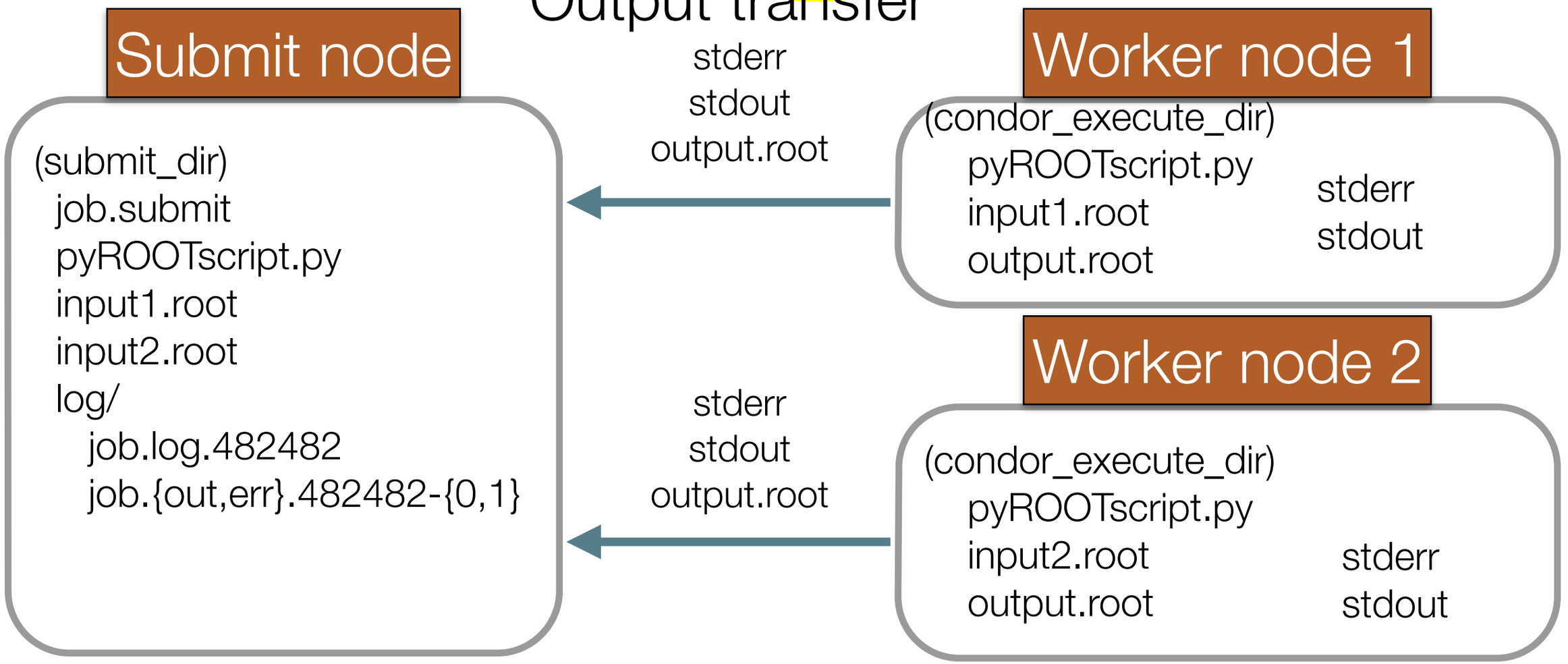
Worker node 2

(condor_execute_dir)
- pyROOTscript.py
- input2.root
- output.root
- stderr
- stdout

Submiting and monitoring your jobs

```
$ condor_q 482482
-- Schedd: login.uscms.org : <192.170.227.118:9618?…
 ID    OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
482482.0   khurtado        6/16 20:11   0+00:00:00 >  0   0.0  pyROOTscript.py input1.root
482482.1   khurtado        6/16 20:11   0+00:00:00 >  0   0.0  pyROOTscript.py input2.root

2 jobs; 0 completed, 0 removed, 0 idle, 2 running, 0 held, 0 suspended
```



Submiting and monitoring your jobs

```
-- Schedd: login.uscms.org : <192.170.227.118:9618?…
 ID   OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD

0 jobs; 0 completed, 0 removed, 0 idle, 0 running, 0 held, 0 suspended
```

All Done.

Submit node

(submit_dir)
- job.submit
- pyROOTscript.py
- input1.root
- input2.root
- log/
  - job.log.482482
  - job.{out,err}.482482-{0,1}
- **output/**
  - **output1.root**
  - **output2.root**

Tips for multiple jobs

- Let's say we want to run multiple jobs using the same submit file.

Tips for multiple jobs

- Multiple jobs without variations (same arguments, input files, executable, etc).

```
Universe = vanilla
Executable = simulation
Arguments = 5000 #events to simulate
Error = log/job.err.$(Cluster)-$(Process)
Output = log/job.out.$(Cluster)-$(Process)
Log = log/job.log.$(Cluster)
transfer_output_remaps = "output.root = output/output_$
(Process).root”
Queue 100
```

Tips for multiple jobs

- Now, imagine we have 100 input files we want to read via XRootD in our program. Assume we provide the LFN as input argument.

```
Executable = simulation
Arguments = $(myfile)
transfer_output_remaps = "output.root = output/output_$(Process).root”
...
```

Tips for multiple jobs

- Now, imagine we have 100 input files we want to read via XRootD in our program. Assume we provide the LFN as input argument.

using multiple Queue statements

```
myfile = /store/mc/RunIIFall15MiniAODv2/TTToSemiLeptonic_13TeV-
powheg/MINIAODSIM/
PU25nsData2015v1_76X_mcRun2_asymptotic_v12_ext1-v1/00001/
E45B4D5A-67C9-E511-BEDB-0CC47A1DF82E.root
Queue 1

myfile = /store/mc/RunIIFall15MiniAODv2/
TTJets_SingleLeptFromT_TuneCUETP8M1_13TeV-madgraphMLM-
pythia8/MINIAODSIM/
PU25nsData2015v1_76X_mcRun2_asymptotic_v12_ext1-
v2/00000/0052E102-10C8-E511-8B8D-003048357A8C.root
Queue 1
…
etc
```

Tips for multiple jobs

- Now, imagine we have 100 input files we want to read via XRootD in our program. Assume we provide the LFN as input argument.

using multiple Queue statements

```
myfile = /store/mc/RunIIFall15MiniAODv2/TTToSemiLeptonic_13TeV-
powheg/MINIAODSIM/
PU25nsData2015v1_76X_mcRun2_asymptotic_v12_ext1-v1/00001/
E45B4D
Queue

myfile =
TTJets
pythia8
PU25n
v2/0000
Queue 1
…
etc
```

Painful to do unless it's just a couple of jobs…

Tips for multiple jobs

- Now, imagine we have 100 input files we want to read via XRootD in our program. Assume we provide the LFN as input argument.

reading it from a file

```
Executable = simulation
Arguments = $(myfile)
transfer_output_remaps = "output.root = output/output_$(Process).root”
queue myfile from file_list.txt
```

Where file_list.txt has the list of LFN files to parse.
Condor will create as many jobs as lines on that list has.

Tips for multiple jobs

- Multiple local input files, same program:

matching pattern

```
Executable = simulation
Arguments = $(myfile)
transfer_input_files = $(myfile)
queue myfile matching *.root
```

Assuming you want to transfer and use all .root files in your submit_dir

Questions?