



Submitting Multiple Jobs With HTCondor

Rachel Lombardi



Agenda

- Motivation for submitting many jobs using a single submit file
- HTCondor submit file options
 - Using variables
 - Modifying the queue statement
- Organizational tips for handling many input/output files
 - Submit file options for handling different job structures

Why multiple jobs?

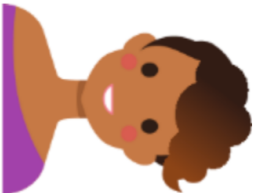
Mei Monte Carlo

Needs to run many random simulations to model particles in a detector

Why multiple jobs?

Mei Monte Carlo


Needs to run many random simulations to model particles in a detector.

Tamara Trials

Testing different design parameters for designing clinical trials.

Why multiple jobs?

Mei Monte Carlo

Needs to run many random simulations to model particles in a detector.

Tamara Trials

Testing different design parameters for designing clinical trials.

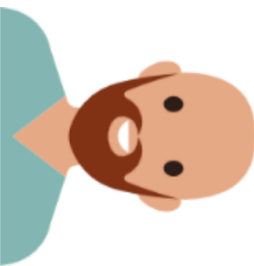

Ben Bioinformatics

Applying a quality control / processing pipeline to 20 RNA samples.

Image credit: [The Carpentries Instructor Training](#)

Why multiple jobs?

Mei Monte Carlo

Needs to run many random simulations to model particles in a detector.

Tamara Trials

Testing different design parameters for designing clinical trials.

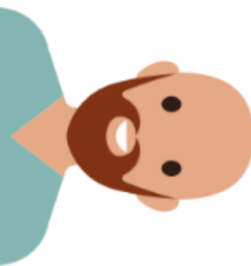
Ben Bioinformatics

Applying a quality control / processing pipeline to 20 RNA samples .

Image credit: [The Carpentries Instructor Training](#)

Why multiple jobs?

Mei Monte Carlo	Tamara Trials	Ben Bioinformatics
Needs to run many random simulations to model particles in a detector	Testing different design parameters for designing clinical trials.	Applying a quality control / processing pipeline to 20 RNA samples.

When running many jobs we want to avoid:

- starting each job manually
- creating separate submit files for each job

Image credit: [The Carpentries Instructor Training](#)

Many jobs, one submit file



HTCondor has several built-in ways to submit many independent jobs from one submit file

Let's review: one job

```
executable = analyze.sh
arguments  = file.in file.out
transfer_input_files = file.in

log      = job.log
output  = job.stdout
error   = job.stder
queue
```

This is the command we want HTCondor to run.

Let's review: one job

```
executable = analyze.sh
arguments = file.in file.out
transfer_input_files = file.in
log = job.log
output = job.stdout
error = job.stder
queue
```

These are the files we need for the job to run.

Let's review: one job

```
executable = analyze.sh  
arguments = file.in file.out  
transfer_input_files = file.in
```

```
Log      = job.log  
output  = job.stdout  
error   = job.stderr  
queue
```

These files track
information about the job.

Let's review: one job

```
executable = analyze.sh
arguments  = file.in file.out
transfer_input_files = file.in
```

```
log      = job.log
output  = job.stdout
error   = job.stderr
```

queue

The queue term tells HTCondor how many jobs to run.



Submitting Multiple Jobs

When submitting multiple jobs using one submit file, it is helpful to start by thinking about:

1. What is ***constant*** across all jobs?
2. What is ***changing*** from job to job?



Submitting Multiple Jobs

When submitting multiple jobs using one submit file, it is helpful to start by thinking about:

1. What is ***constant*** across all jobs?
2. What is ***changing*** from job to job?

When editing the submit file, it is helpful to start by editing the **queue** statement.



Variable and queue options

Syntax	List of Values	Variable Name
queue N	Integers: 0 through N-1	\$(Procid)
queue Var matching pattern *	List of values that match the wildcard pattern.	\$(<i>Var</i>)
queue Var in (<i>item1 item2 ...</i>)	List of values within parentheses.	If no variable name is provided, default is \$(Item)
queue Var from <i>list</i>	List of values from <i>list</i> , where each value is on its own line.	

Variable and queue options

Syntax	List of Values	Variable Name
<p>queue N</p> <p>queue <i>Var</i> matching pattern*</p>	<p>Integers: 0 through N-1</p> <p>List of values that match the wildcard pattern.</p>	<p>\$(Procid)</p>
<p>queue <i>Var</i> in (<i>item1 item2 ...</i>)</p>	<p>List of values within parentheses.</p>	<p>\$(<i>Var</i>)</p>
<p>queue <i>Var</i> from <i>list</i></p>	<p>List of values from <i>list</i>, where each value is on its own line.</p>	<p>If no variable name is provided, default is \$(Item)</p>

Example 1:

Queue *variable* from *List*

Example 1:

Scenario: Use an executable to analyze Wisconsin population data

```
$ ./compare_states state.wi.dat out.state.wi.dat
```



```
executable = compare_states  
arguments  = state.wi.dat out.state.wi.dat  
transfer_input_files = state.wi.dat  
queue
```



Example 1:

Scenario: Use an executable to analyze Wisconsin population data

Suppose we have data for all 50 states: `state.wi.dat`,
`state.mn.dat`, `state.il.dat`, ...

Let's use HTCondor to automatically queue a job to
analyze each state's data file!

```
e
arguments = state.wi.dat out.state.wi.dat
transfer_input_files = state.wi.dat
queue
```

Provide a list of values with queue ... **from**

One option is to create another file with the list of input files and use the **queue variable from List** syntax.

```
executable = compare_states
arguments  = state.wi.dat out.state.wi.dat
transfer_input_files = state.wi.dat
queue state from state_list.txt
```

File name: state_list.txt

```
state.wi.dat
state.mn.dat
state.il.dat
state.ia.dat
state.mi.dat
```



Which job components vary?

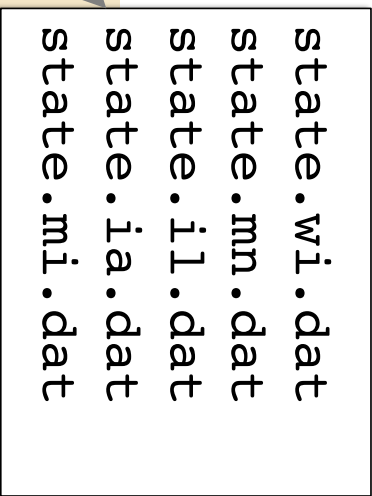
- Now, what parts of our submit file vary depending on the input?
- We want to vary the job's **arguments** and one **input file**.

```
executable = compare_states
arguments  = state.wi.dat out.state.wi.dat
transfer_input_files = state.wi.dat
queue state from state_list.txt
```

Use a custom variable

Replace all our varying components in the submit file with a variable.

```
executable = compare_states
arguments  = $(state) out.$(state)
transfer_input_files = $(state)
queue state from state_list.txt
```



```
state.wi.dat
state.mn.dat
state.il.dat
state.ia.dat
state.mi.dat
```

Use multiple variables with queue ... from

- The queue from syntax can also support multiple values per job.
- Suppose our command was like this:

```
$ ./compare_states -i [input file] -y [year]
```

File name: state_list.txt

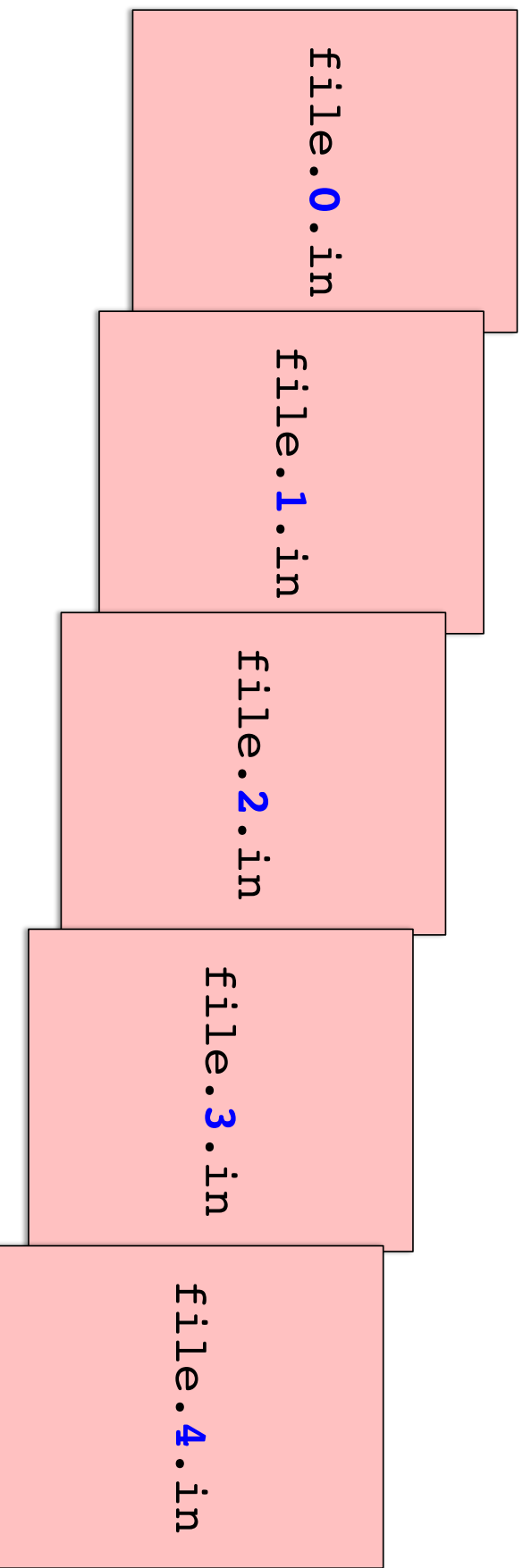
```
state.wi.dat, 2010  
state.wi.dat, 2015  
state.mn.dat, 2010  
state.mn.dat, 2015
```

```
executable = compare_states  
arguments  = -i $(state) -y $(year)  
  
transfer_input_files = $(state), country.us.dat  
queue state,year from state_list.txt
```

Example 2: Queue N

List of numerical input values

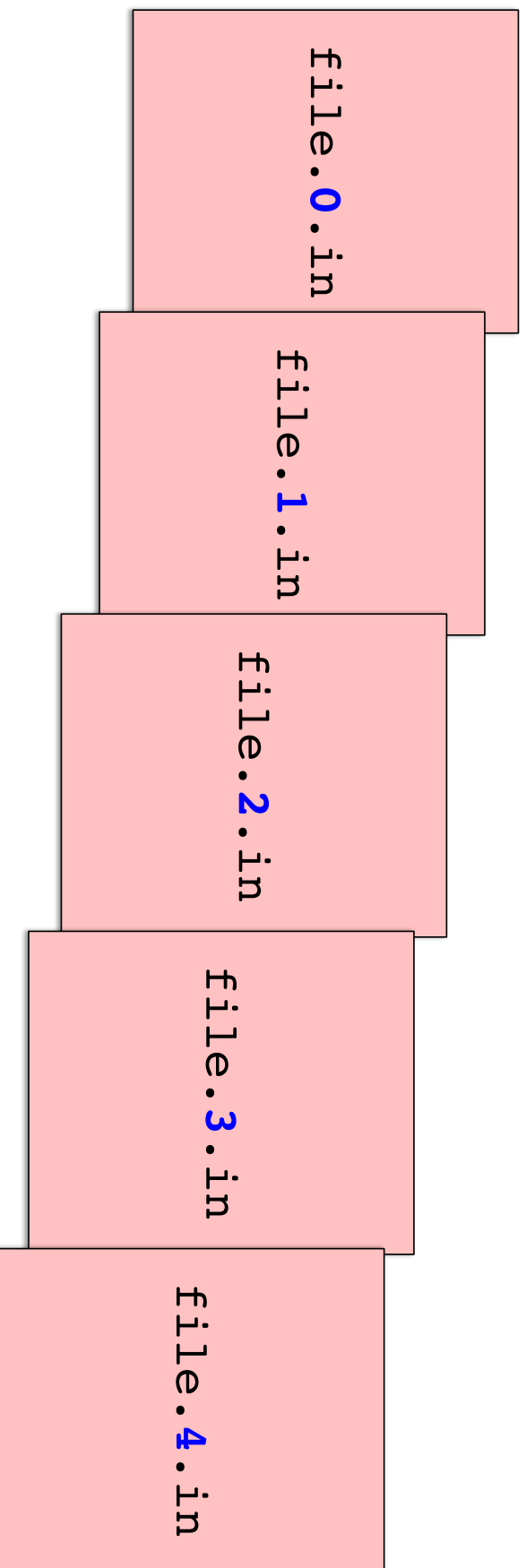
Suppose we have many input files and we want to run one job per input file.



List of numerical input values

Suppose we have many input files and we want to run one job per input file.

We can capture this set of inputs using a **list of integers**.





Provide a list of integer values with queue N

```
executable = analyze.sh
arguments  = file.in file.out
transfer_input_files = file.in

log      = job.log
output  = job.stdout
error   = job.stder

queue 5
```

This queue statement will generate a list of integers, 0 - 4

Provide a list of integer values with queue N

```
executable = analyze.sh  
arguments = file.in file.out  
transfer_input_files = file.in
```

```
log = job.log  
output = job.stdout  
error = job.stderr
```

queue **5**

If we *only* change our queue statement to queue N, HTCondor will queue N *identical* jobs.

This queue statement will generate a list of integers, 0 - 4

Which job components vary?

```
executable = analyze.sh
arguments = file.in file.out
transfer_input_files = file.in

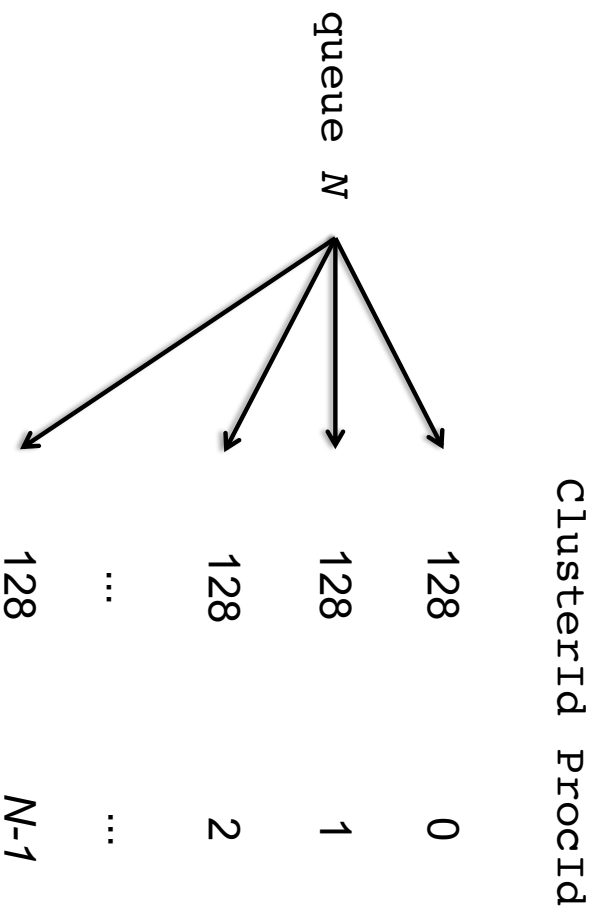
Log = job.Log
output = job.stdout
error = job.stderr

queue 5
```

The arguments for our command and the input files would be different for each job.

We might also want to differentiate these job files.

HTCondor Automatic Variables



Each job's
 ClusterId and
 ProcId can be
 accessed inside the
 submit file using:

`$(ClusterId)`
`$(ProcId)`

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



Use `$(Procid)` as the variable

```
executable = analyze.sh
arguments  = file.$(Procid).in file.$(Procid).out
transfer_input_files = file$(Procid).in

Log      = job.$(Procid).Log
output  = job.$(Procid).stdout
error   = job.$(Procid).stderr
queue   5
```

The default variable representing the changing numbers in our list is `$(Procid)`

Submitting Jobs

Jobs in the queue will be grouped in batches
(default: cluster number)

```
$ condor_submit job.submit
Submitting job(s).
5 job(s) submitted to cluster 128.
```

```
$ condor_q
-- Schedd: submit-1.chtc.wisc.edu : <128.104.101.92:9618?... @ 05/09/19 10:35:54
OWNER  BATCH_NAME  SUBMITTED  DONE  RUN  IDLE  TOTAL  JOB_IDS
alice  ID: 128        5/9  11:03  -    -    5      5      128.0-4
5 jobs; 0 completed, 0 removed, 5 idle, 0 running, 0 held, 0 suspended
```

To see individual jobs, use:

condor_q -nobatch

Other options: queue N

Can I start from 1 instead of 0?

- Yes! These two lines increment the \$(Procid) variable

```
tempProc = $(Procid) + 1  
newProc = $INT(tempProc)
```

- You would use the second variable name \$(newProc) in your submit file

Can I create a certain number of digits (i.e. 000, 001 instead of 0,1)?

- Yes, this syntax will make \$(Procid) have a certain number of digits

```
$INT(Procid,%03)
```



Other Options for Submitting Multiple Jobs

Variable and queue options

Syntax	List of Values	Variable Name
queue <i>N</i>	Integers: 0 through N-1	\$(Procid)
queue <i>Var</i> matching <i>pattern</i> *	List of values that match the wildcard pattern.	\$(<i>Var</i>)
queue <i>Var</i> in (<i>item1 item2 ...</i>)	List of values within parentheses.	If no variable name is provided, default is \$(<i>Item</i>)
queue <i>Var</i> from <i>list.txt</i>	List of values from <i>list.txt</i> , where each value is on its own line.	



Other options: `queue ... matching`

Queue matching has options to select only files or directories

```
queue infile matching files *.dat
```

```
queue indir matching dirs job*
```

If you have questions about which queue statement would work best for *your* workflow, don't hesitate to reach out to OSG staff this week!

Queue options, pros and cons

queue N	<ul style="list-style-type: none">- Simple, good for multiple jobs that only require a numerical index.
queue matching pattern*	<ul style="list-style-type: none">- Natural nested looping, minimal programming, use optional “files” and “dirs” keywords to only match files or directories- Requires good naming conventions.
queue in (List)	<ul style="list-style-type: none">- All information contained in a single file, reproducible- Harder to automate submit file creation
queue from file	<ul style="list-style-type: none">- Supports multiple variables, highly modular (easy to use one submit file for many job batches), reproducible- Additional file needed



Organization

(more on this later!)



Organization

```
12181445_0.err 16058473_0.err 17381628_0.err 18159900_0.err 5175744_0.err 7266263_0.err
12181445_0.log 16058473_0.log 17381628_0.log 18159900_0.log 5175744_0.log 7266263_0.log
12181445_0.out 16058473_0.out 17381628_0.out 18159900_0.out 5175744_0.out 7266263_0.out
13609567_0.err 16060330_0.err 17381640_0.err 3446080_0.err 5176204_0.err 7266267_0.err
13609567_0.log 16060330_0.log 17381640_0.log 3446080_0.log 5176204_0.log 7266267_0.log
13609567_0.out 16060330_0.out 17381640_0.out 3446080_0.out 5176204_0.out 7266267_0.out
13612268_0.err 16254074_0.err 17381665_0.err 3446306_0.err 5295132_0.err 7937420_0.err
13612268_0.log 16254074_0.log 17381665_0.log 3446306_0.log 5295132_0.log 7937420_0.log
13612268_0.out 16254074_0.out 17381665_0.out 3446306_0.out 5295132_0.out 7937420_0.out
13630381_0.err 17134215_0.err 17381676_0.err 4347054_0.err 5318339_0.err 8779997_0.err
13630381_0.log 17134215_0.log 17381676_0.log 4347054_0.log 5318339_0.log 8779997_0.log
13630381_0.out 17134215_0.out 17381676_0.out 4347054_0.out 5318339_0.out 8779997_0.out
```

Many jobs means many files.

We will have a talk on how to be organized while scaling out your workflow later this week



Questions?