Organizing and Submitting HTC Workloads

HTCondor Week 2022
Rachel Lombardi

Objective

Learn how to organize and submit workloads composed of many jobs using HTCondor

Objective

Learn how to organize and submit workloads composed of many jobs using HTCondor

- Know which files to consider when organizing HTC workload submissions
- Plan and implement an organization structure for workload files on the Access Point
- Utilize HTCondor submit file options to accommodate your organization structure and data movement strategy

Organizing HTC Workload Components

High Throughput Computing (HTC)

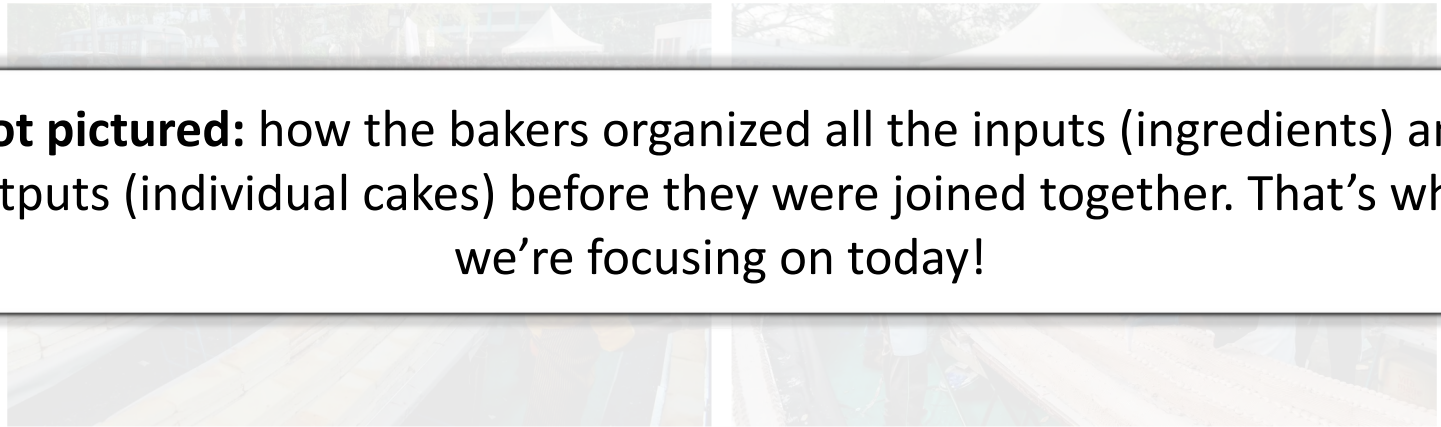
Solving a big problem by executing many small, self-contained tasks and joining them.

Example: baking the world's largest/longest cake

Photos: Arun Sankar via https://www.theguardian.com/world/2020/jan/16/indian-bakers-rise-to-task-of-making-worlds-longest-cake

High Throughput Computing (HTC)

Solving a big problem by executing many small, self-contained tasks and joining them.



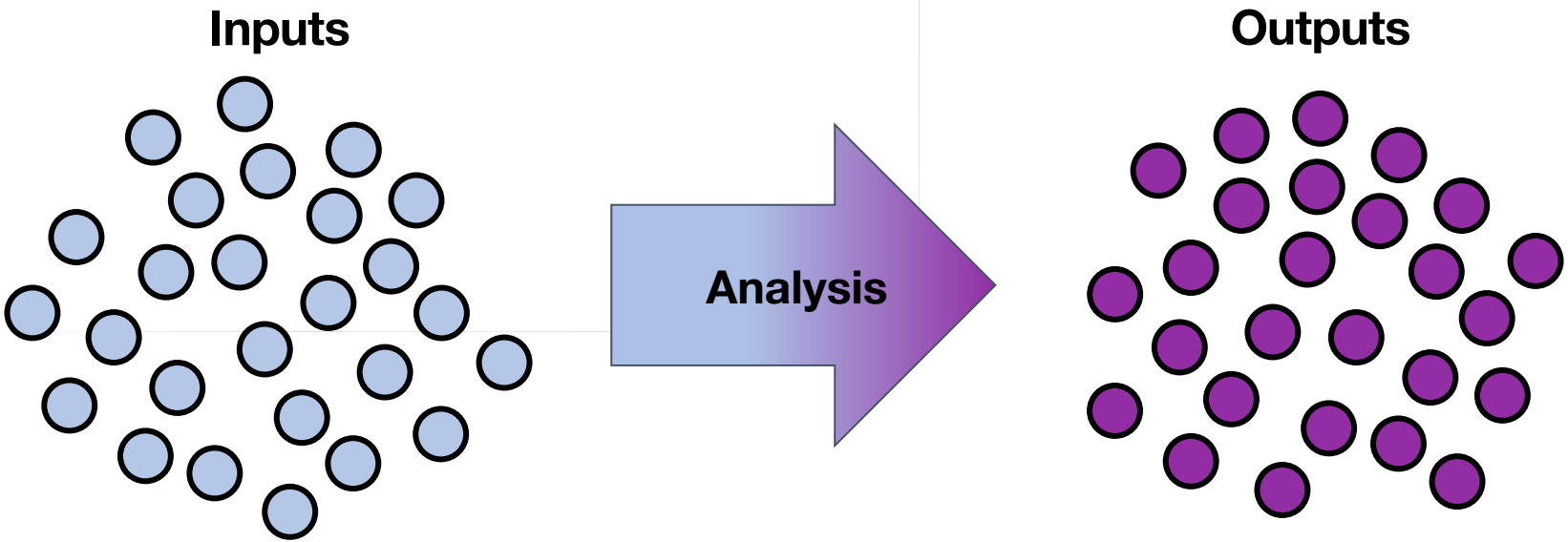
Not pictured: how the bakers organized all the inputs (ingredients) and outputs (individual cakes) before they were joined together. That's what we're focusing on today!

Example: baking the world's largest/longest cake

Photos: Arun Sankar via https://www.theguardian.com/world/2020/jan/16/indian-bakers-rise-to-task-of-making-worlds-longest-cake

HTC Workloads as Input/Output Sets

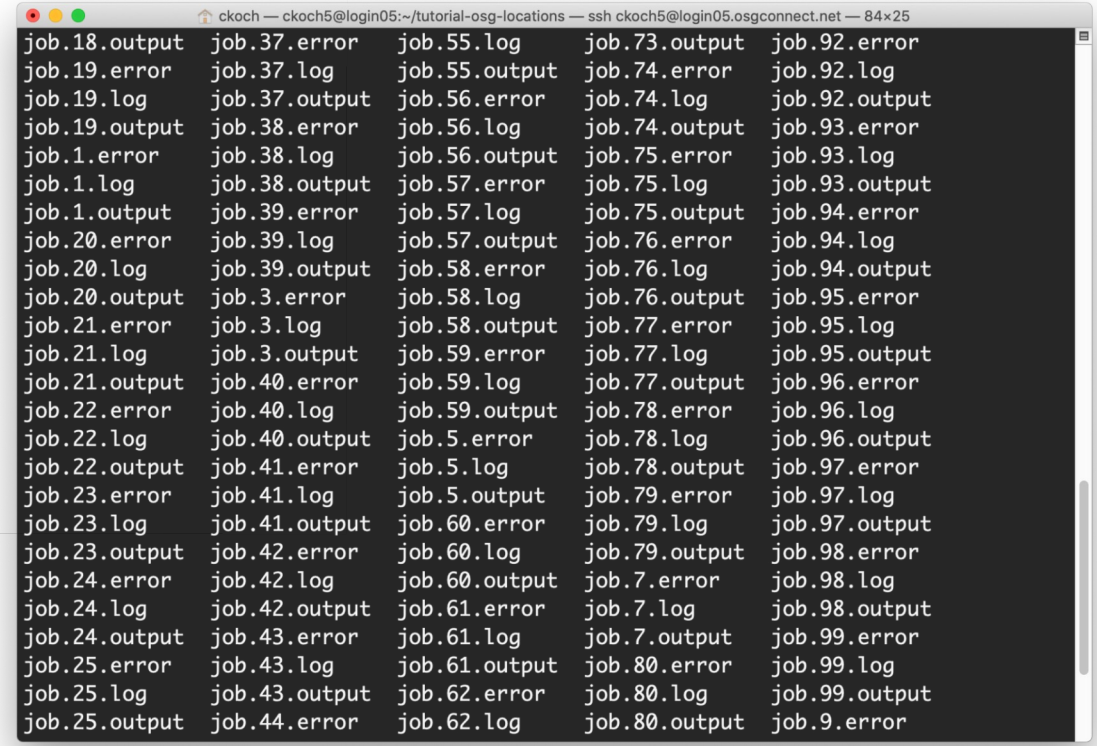
Today, we're mainly going to think about workloads that use many input files to produce many output files.



Why organize?

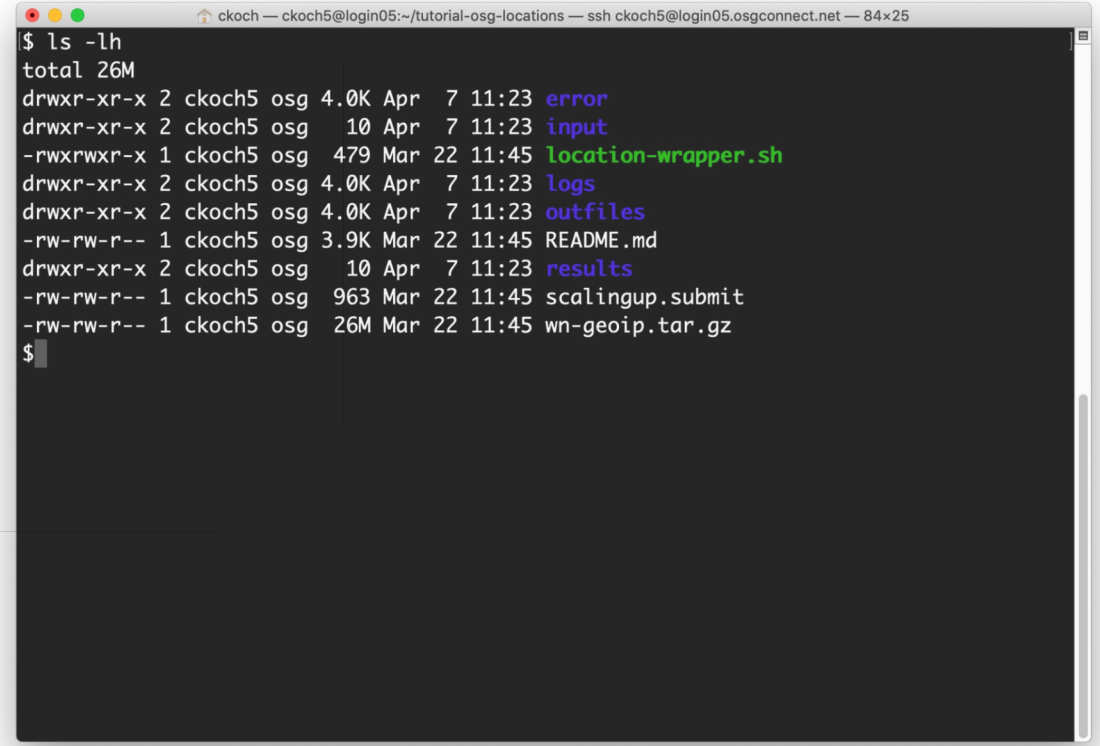
By default, HTCondor writes all job files (input, output, HTCondor logs, etc.) back to the same place, which means your home directory can look something like this:

This makes it hard to find things!



Why organize?

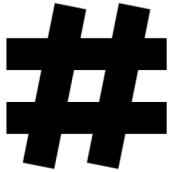
We can improve our workflow by intentionally organizing our input and output files on the Access Point.



Example: Text Analysis

Book text to analyze

Python script that counts the frequency of different words



Output counts of different words in book

```
$ ./wordcount.py Dracula.txt
```

Organizational Plan For Our Files

```
books.submit
wordcount.py
input/
        Dracula.txt
        ...
output/
        count.Dracula.txt
        ...
```

We will assume that we want to put our input files (books) in one folder, and our output files (word counts) in another folder.

Organizational Plan For HTCondor/System Files

```
books.submit
wordcount.py
input/
        Dracula.txt
        ...
output/
        count.Dracula.txt
        ...
log/
        job.0.log
        ...
errout/
        job.0.out
        job.0.err
        ...
```

There are ***additional*** files that will be produced by the job as well that we should consider – the HTCondor log, stdout and stderr. We'll put these into two folders.

Organizing and Submitting One Job

Shell Tools For Organizing Files

Shell commands:

```
mkdir <directory name>
mv <file_to_move> <destination>
```

The wildcard can specify groups of files:

`ls *.txt` - this will match all files that end with .txt

See Creating Files and Directories from Software Carpentry's Shell Lesson for more details.

HTCondor Submit File Options for Organizing Files

Syntax	Purpose	Features

`Transfer_output_remaps =`
`"file1.out=path/to/file1.out;`
`file2.out=path/to/renamedFile2.out"`
	Used to save output files in a specific path and using a certain name	- Used to save output files to a **specific folder**
		- Used to **rename** output files to avoid writing over existing files

`Initialdir =` `path/to/initialDirectory`	Sets the submission directory for each job. When set, this is becomes the base path where output files will be saved.	- Used to submit multiple jobs from **different directories** - Used to avoid having to write some paths in other submit file values

More Information: https://htcondor.readthedocs.io/en/latest/users-manual/file-transfer.html

Let's Practice!

1. Log into an HTCondor Access Point

2. Download the tutorial and navigate inside the folder:

```
$ git clone https://github.com/CHTC/organizing-examples
$ cd organizing-examples
$ ls
```

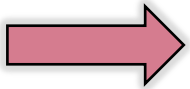
3. Organize input files

```
$ mv *.txt input/
```


Our Project Directory

Before Job Submission

```
books.submit
wordcount.py
input/
        Dracula.txt
        ...
output/
log/
errout/
```



After Job Submission

```
books.submit
wordcount.py
input/
        Dracula.txt
        ...
output/
        count.Dracula.txt
        ...
log/
        job.0.log
        ...
errout/
        job.0.out
        job.0.err
        ...
```


Start With the Job's Executable and Arguments

```
books.submit
wordcount.py
input/
        Dracula.txt
        ...
output/
        counts.Dracula.txt
        ...
log/
        job.0.log
        ...
errout/
        job.0.out
        job.0.err
        ...
```

```
# submit file name: books.submit
executable = wordcount.py
arguments = Dracula.txt




queue
```

Organize, Transfer Inputs

```
books.submit
wordcount.py
input/
        Dracula.txt
        ...
output/
        counts.Dracula.txt
        ...
log/
        job.0.log
        ...
errout/
        job.0.out
        job.0.err
        ...
```

```
# submit file name: books.submit
executable = wordcount.py
arguments = Dracula.txt

transfer_input_files = input/Dracula.txt




queue
```

Transfer, Organize Outputs

```
books.submit
wordcount.py
input/
        Dracula.txt
        ...
output/
        count.Dracula.txt
        ...
log/
        job.0.log
        ...
errout/
        job.0.out
        job.0.err
        ...
```

```
# submit file name: books.submit
executable = wordcount.py
arguments = Dracula.txt

transfer_input_files = input/Dracula.txt

transfer_output_remaps =
  "count.Dracula.txt=output/count.Dracula.txt"




queue
```

Organizing Additional Job Files

```
books.submit
wordcount.py
input/
        Dracula.txt
        ...
output/
        count.Dracula.txt
        ...
log/
        job.0.log
        ...
errout/
        job.0.out
        job.0.err
        ...
```

```
# submit file name: books.submit
executable = wordcount.py
arguments = Dracula.txt

transfer_input_files = input/Dracula.txt

transfer_output_remaps =
  “count.Dracula.txt=output/count.Dracula.txt”

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

queue
```

Queue One Job

Queue one job
to analyze Dracula.txt


```
# submit file name: books.submit
executable = wordcount.py
arguments = Dracula.txt

transfer_input_files = input/Dracula.txt

transfer_output_remaps =
  “count.Dracula.txt=output/count.Dracula.txt”

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

queue
```

Let's Analyze One Book!

Fill out the "books.submit" file in the organizing files tutorial to submit a single element of the workflow (one job).

```
# submit file name: books.submit
executable = wordcount.py
arguments = Dracula.txt

transfer_input_files = input/Dracula.txt

transfer_output_remaps = "count.Dracula.txt=output/count.Dracula.txt"

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

queue
```


Submitting a Full HTC Workload

Submitting the Whole Workload

To submit the whole workload - processing all of our input set, we need to modify this queue statement:

```
# submit file name: books.submit
executable = wordcount.py
arguments = Dracula.txt

transfer_input_files = input/Dracula.txt

transfer_output_remaps =
  “count.Dracula.txt=output/count.Dracula.txt”

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

queue
```

Queue Multiple Jobs

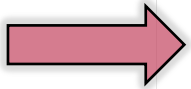
Syntax	List of Values	Variable Name
queue *N*	Integers: 0 through N-1	$(ProcID)
queue *Var* matching *pattern**	List of values that match the wildcard pattern.	**$(*Var*)** If no variable name is provided, default is $(Item)
queue *Var* in (*item1 item2* …)	List of values within parentheses.	
queue *Var*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

First, List the Input Set

Make a file called `list.txt` containing the names of the books we want to analyze:

```
$ pwd
../organizing-examples

$ ls input/ > list.txt
```



```
Alice.txt
PandP.txt
Dracula.txt
…
```

This can be a list of either values (like parameters) or input files.

Add the Input Set to the Submit File

list.txt

```
books.submit
wordcount.py
input/
        Dracula.txt
        ...
output/
        count.Dracula.txt
        ...
log/
        job.0.log
        ...
errout/
        job.0.out
        job.0.err
        ...
```

```
# submit file name: books.submit
executable = wordcount.py
arguments = Dracula.txt

transfer_input_files = input/Dracula.txt

transfer_output_remaps =
  “count.Dracula.txt=output/count.Dracula.txt”

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

queue book from list.txt
```

Replace Changing Values With Variables

list.txt

```
books.submit
wordcount.py
input/
        Dracula.txt
        ...
output/
        counts.Dracula.txt
        ...
logs/
        job.0.log
        ...
errout/
        job.0.out
        job.0.err
        ...
```

```
# submit file name: books.submit
executable = wordcount.py
arguments = Dracula.txt

transfer_input_files = input/Dracula.txt

transfer_output_remaps =
  "count.Dracula.txt=output/count.Dracula.txt"

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

queue book from list.txt
```

Replace Changing Values With Variables

list.txt

```
books.submit
wordcount.py
input/
        Dracula.txt
        ...
output/
        counts.Dracula.txt
        ...
logs/
        job.0.log
        ...
errout/
        job.0.out
        job.0.err
        ...
```

```
# submit file name: books.submit
executable = wordcount.py
arguments = $(book)

transfer_input_files = input/$(book)

transfer_output_remaps =
  "count.$(book)=output/count.$(book)"

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

queue book from list.txt
```

Let’s Analyze Many Books!

Prepare submit file for multi-job (full workload) submission

```
# submit file name: books.submit
executable = wordcount.py
arguments = $(book)

transfer_input_files = input/$(book)

transfer_output_remaps = “count.$(book)=output/count.$(book)”

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

queue book from list.txt
```


Our New Project Directory

Organized Workflow

```
books.submit
wordcount.py
input/
        Dracula.txt Pride_and_Prejudice.txt Huckleberry_Finn.txt
        Alice_in_Wonderland.txt Ulysses.txt
output/
        count.Dracula.txt count.Pride_and_Prejudice.txt count.Huckleberry_Finn.txt
        count.Alice_in_Wonderland.txt count.Ulysses.txt
log/
        job.0.log job.2.log job.4.log
        job.1.log job.3.log

errout/
        job.0.out job.1.out job.2.out job.3.out job.4.out
        job.0.err job.1.err job.2.err job.3.err job.4.err
```


Other Organizational Models

HTCondor Submit File Options for Organizing Files

Syntax	Purpose	Features

`Transfer_output_remaps =`
`"file1.out=path/to/file1.out;`
`file2.out=path/to/renamedFile2.out"`
	Used to save output files in a specific path and using a certain name	- Used to save output files to a **specific folder**
		- Used to **rename** output files to avoid writing over existing files

`Initialdir =` `path/to/initialDirectory`	Sets the submission directory for each job. When set, this is becomes the base path where output files will be saved.	- Used to submit multiple jobs from **different directories** - Used to avoid having to write some paths in other submit file values

More Information: https://htcondor.readthedocs.io/en/latest/users-manual/file-transfer.html

Return Output to Specified Directory with InitialDir

```
submission_dir/
        job.sub
        exec.py
        shared_vars.txt
        results/
                input.txt
                output.txt
                job.err
                job.log
                job.out
```

```
# File name: job.sub
executable = exec.py

initialdir = results
transfer_input_files = input.txt,
  ../shared_vars.txt

log = job.log
out = job.out
error = job.err

queue 1
```


Separate Jobs with InitialDir

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

```
# File name = job.submit

executable = analyze.exe
initialdir = job$(ProcId)

arguments = file.in file.out
transfer_input_files = file.in

log = job.log
error = job.err
output = job.out

queue 3
```

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

Questions to Ask Yourself

How big are the files in my input / output sets?

What organizational strategy makes sense for the next steps in my analysis?

- Do you want inputs in one folder and outputs in another folder? Use `transfer_output_remaps`.
- Do you have many outputs for each job that you'd like to group together, but keep separate from other job outputs? Do you want to keep inputs/outputs for the same job together? Maybe use `initialdir`.

How do you want to organize the HTCondor/system files?

Acknowledgements

This material is based upon work supported by the National Science Foundation under Cooperative Agreement OAC-2030508 as part of the PATh Project. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.