

HTC From a Users Perspective

By: Cole Bollig

Software Developer for CHTC

European HTCCondor Workshop 2024

CHTC

HTCCondor
Software Suite

PATH

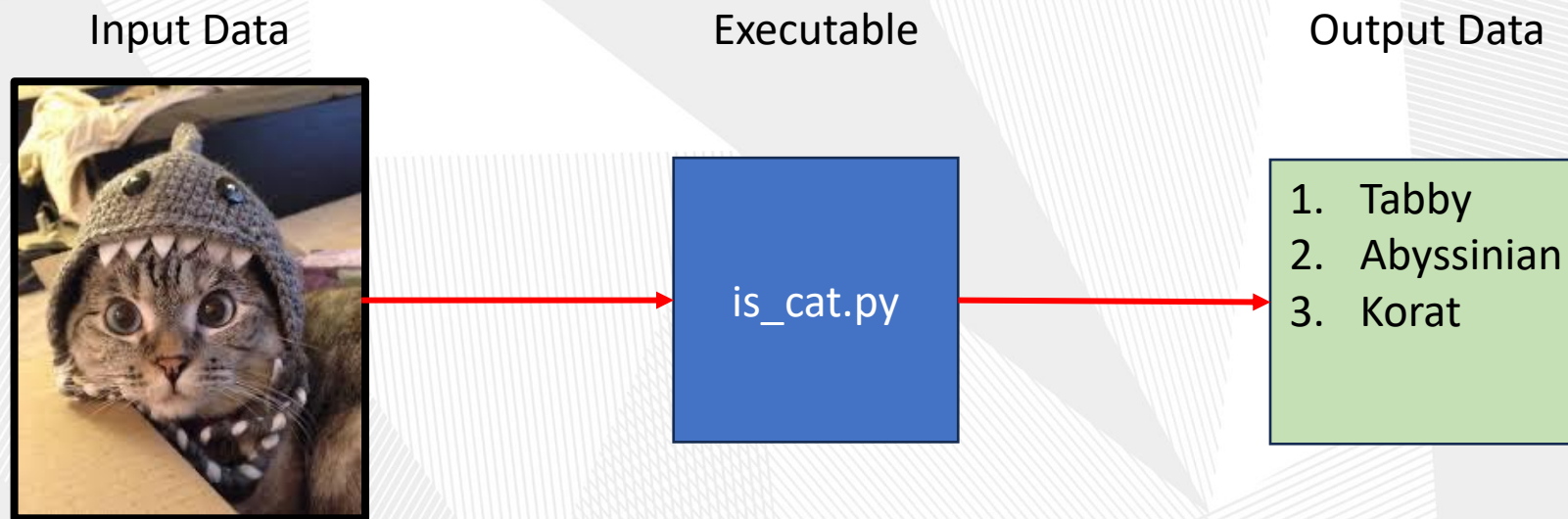
HTCondor Jobs

- HTCondor schedules and executes Jobs
- What is a Job?
 - A 'Job' is a single computational task
 1. Input Data
 2. Executable (program)
 3. Output Data
- Executable must be runnable from the command line without any interactive input



Example Job

- I wrote a program to check if a picture is of a cat. If the picture is a cat, then the program will create a list of possible breeds. Otherwise, the program fails.



Describing an HTCondor Job

```
executable = is_cat.py
arguments = cat.img

transfer_input_files = cat.img
transfer_output_files = potential-breeds.txt
should_transfer_files = Yes
when_to_transfer_output = ON_SUCCESS

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

request_cpus = 1
request_disk = 50MB
request_memory = 20MB

queue 1
```

Describe an HTCondor job
with the Job Description
Language (JDL)

CAT-Analysis.sub

Describing an HTCondor Job

```
executable = is_cat.py  
arguments = cat.img
```

```
transfer_input_files = cat.img  
transfer_output_files = potential-breeds.txt  
should_transfer_files = Yes  
when_to_transfer_output = ON_SUCCESS
```

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

```
request_cpus = 1  
request_disk = 50MB  
request_memory = 20MB
```

```
queue 1
```

List the executable and arguments for HTCondor to run at the EP like how you would run the executable by hand

```
$ is_cat.py cat.img
```

CAT-Analysis.sub

Describing an HTCondor Job

```
executable = is_cat.py
arguments = cat.img

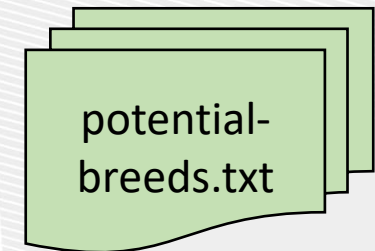
transfer_input_files = cat.img
transfer_output_files = potential-breeds.txt
should_transfer_files = Yes
when_to_transfer_output = ON_SUCCESS

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

request_cpus = 1
request_disk = 50MB
request_memory = 20MB

queue 1
```

Specify input data that the job requires and output data we are expecting to be created (and transferred back)



CAT-Analysis.sub

Describing an HTCondor Job

```
executable = is_cat.py
arguments = cat.img

transfer_input_files = cat.img
transfer_output_files = potential-breeds.txt
should_transfer_files = Yes
when_to_transfer_output = ON_SUCCESS
```

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

```
request_cpus = 1
request_disk = 50MB
request_memory = 20MB
```

```
queue 1
```

- **log** - Generated by HTCondor to track job progress from key events in the jobs lifetime
- **output** - File that captures the jobs STDOUT
- **error** - File that captures the jobs STDERR

CAT-Analysis.sub

Describing an HTCondor Job

```
executable = is_cat.py
arguments = cat.img

transfer_input_files = cat.img
transfer_output_files = potential-breeds.txt
should_transfer_files = Yes
when_to_transfer_output = ON_SUCCESS

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

request_cpus = 1
request_disk = 50MB
request_memory = 20MB

queue 1
```

- Request the appropriate resources for your job to run.
- **queue** - keyword indicating “create a job.”

CAT-Analysis.sub

Resource Requirements

- Jobs are nearly always using a part of a computer, not the whole thing. EP divides worker node into execute "Slots".
- Very important to request appropriate resources (memory, cpus, disk) for a job
- Job will likely be enforced to only use what it requests

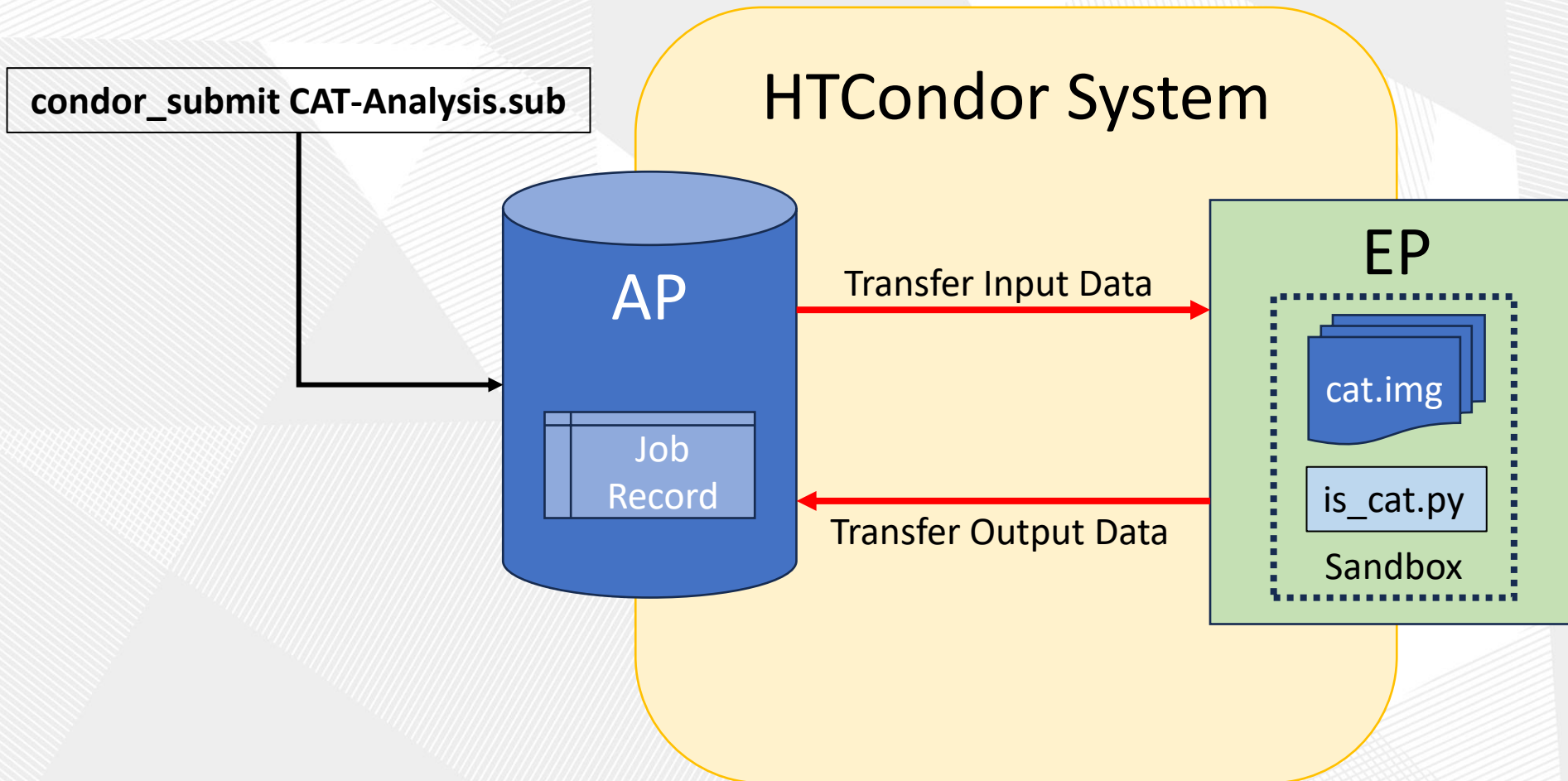


Place the Job to an Access Point

- Simply run *condor_submit <Submit File>*

```
colebollig@Coles-MacBook-Pro sleep % condor_submit CAT-Analysis.sub
Submitting job(s).
1 job(s) submitted to cluster 470.
colebollig@Coles-MacBook-Pro sleep %
```

What happens?



Monitor Jobs

- Use *condor_q* to get job record information from the AP
- Use *condor_watch_q* to actively track a jobs state

```
colebollig@Coles-MacBook-Pro sleep % condor_q
-- Schedd: COLES_AP@ : <127.0.0.1:49876?... @ 09/25/24 00:51:57
OWNER   BATCH_NAME   SUBMITTED   DONE   RUN   IDLE   TOTAL JOB_IDS
colebollig ID: 470   9/25 00:51   _   1   _   1 470.0
```

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

```
colebollig@Coles-MacBook-Pro sleep % condor_watch_q
BATCH   IDLE RUN DONE TOTAL JOB_IDS
ID: 470   - 1 - 1 37.0 [=====]
```

```
[=====]
```

```
Total: 6 jobs; 5 completed, 1 running
```

```
Updated at 2024-09-25 00:54:15
Input ^C to exit
```

Monitor Jobs

- Get human readable job status via *htcondor job status*

```
$ htcondor job status 123.45
```

```
Job 123.45 is currently running on host exec221.chtc.wisc.edu.  
It started running again 2.1 hours ago.  
It was submitted 3.6 hours ago.  
Its current memory usage is 2.5 GB out of 4.0 GB requested.  
Its current disk usage is 3.8 GB out of 5.5 GB requested.  
It has restarted 2 times.  
Goodput is 80% (0.5 hours badput, 2.1 hours goodput).
```

```
[cabollig@ap2002 ~]$ htcondor ap status
```

<u>Access Point</u>	<u>Health</u>
ap2002.chtc.wisc.edu	Good
ap2004.chtc.wisc.edu	Good

Also, checkout AP health via *htcondor ap status*

Monitor Jobs

- Use *condor_history* to view completed job records from the AP

```
colebollig@Coles-MacBook-Pro sleep % condor_history -limit 10
ID OWNER SUBMITTED RUN_TIME ST COMPLETED CMD
37.0 colebollig 9/25 00:54 0+00:01:00 C 9/25 00:55 /bin/sleep 60
34.0 colebollig 9/18 19:25 0+00:01:38 X /Users/colebollig/Desktop/...
36.0 colebollig 9/18 19:25 0+00:01:16 X /bin/sleep 1000
35.3 colebollig 9/18 19:25 X /bin/sleep 1000
35.2 colebollig 9/18 19:25 X /bin/sleep 1000
35.1 colebollig 9/18 19:25 X /bin/sleep 1000
35.0 colebollig 9/18 19:25 X /bin/sleep 1000
31.0 colebollig 9/18 19:23 0+00:00:56 X /Users/colebollig/Desktop/...
33.0 colebollig 9/18 19:24 0+00:00:13 X /bin/sleep 1000
32.1 colebollig 9/18 19:23 0+00:00:15 X /bin/sleep 1000
```

Other Useful Tools

- condor_status
- condor_q
- condor_q -analyze
- condor_ssh_to_job
- condor_submit -i
- condor_hold / release
- condor_run
- condor_rm
- condor_prio
- condor_history
- condor_submit_dag
- condor_chirp

View Ads in the Collector (e.g. EP Slots)

View Jobs at an AP

Why job/machines fail to match?

Create ssh session to active job

Submit interactive job

Hold a job, or release a held job

Submit and block

Remove Jobs

Intra-User Job Prios

Completed Job Info

Submit new DAG workflow

Access files/ad from active job

Details of the Job Event Log

```
000 (128.000.000) 05/09 11:09:08 Job submitted from host:
<128.104.101.92&sock=6423_b881_3>
...
001 (128.000.000) 05/09 11:10:46 Job executing on host:
<128.104.101.128:9618&sock=5053_3126_3>
...
006 (128.000.000) 05/09 11:10:54 Image size of job updated: 220
1 - MemoryUsage of job (MB)
220 - ResidentSetSize of job (KB)
...
005 (128.000.000) 05/09 11:12:48 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
0 - Run Bytes Sent By Job
33 - Run Bytes Received By Job
0 - Total Bytes Sent By Job
33 - Total Bytes Received By Job
Partitionable Resources : Usage Request Allocated
Cpus : 1 1 1
Disk (KB) : 14 20480 17203728
Memory (MB) : 1 20 20
```


Submitting Lists of Jobs

```
executable = is_cat.py
arguments = $(image)

transfer_input_files = $(image)
transfer_output_files = $(image).breeds
should_transfer_files = Yes
when_to_transfer_output = ON_SUCCESS
```

```
log = job.log
output = out/job-$(Process).out
error = error/job-$(Process).err
```

```
request_cpus = 1
request_disk = 50MB
request_memory = 20MB
```

```
queue image matching images/*.img
```

- "Nobody is in the business of submitting one job" – Miron

Working Directory/

CAT-Analysis.sub	images/	out/	error/
is_cat.py	cat.img	job-0.out	job-0.err
job.log	dog.img	job-1.out	job-1.err
cat.img.breeds	zoo.img	job-2.out	job-2.err
	...	job-N.out	job-N.err

CAT-Analysis.sub

Possible Queue Statements

matching ... pattern	<pre>queue image matching images/*.img</pre>
in ... list	<pre>queue image in (cat.img dog.img zoo.img)</pre>
from ... file	<pre>queue image from picture_list.txt</pre> <pre>cat.img dog.img zoo.img</pre> <pre>picture_list.txt</pre>

Late Materialization

- Submit millions of jobs without taking down the AP
- AP creates a job factory to materialize jobs as needed

```
executable = is_cat.py
arguments = $(image)

...

max_materialize = 500
max_idle = 100

queue 1000000
```

CAT-Analysis.sub

condor_submit -factory <Submit File>

Python API

- HTCondor has a built in Python API
 - Simply import the **htcondor** python module
- Can do what most of the tools can do from the comfort of Python
 - Place jobs
 - Query an AP
 - Query for available capacity (Slot Ads)
 - And more...
- [HTCondor Python API](#)

```
hist.py — executables
1  #!/usr/bin/env python3~
2  ~
3  import htcondor~
4  import htcondor2~
5  ~
6  ~
7  print("Schedd Job History:")~
8  for ad in htcondor.Schedd().history(None, ["ClusterId", "ProcId"], match=1):~
9  > print(ad)~
10 ~
11 print("Epoch History:")~
12 for ad in htcondor.Schedd().jobEpochHistory(None, ["ClusterId", "ProcId"], match=1):~
13 > print(ad)~
14 ~
15 print("Epoch history w/ type:")~
16 for ad in htcondor.Schedd().jobEpochHistory(None, [], match=1, ad_type="transfer"):~
17 > print(ad)~
18 ~
19 print("Startd history:")~
20 for ad in htcondor.Startd().history(None, ["ClusterId", "ProcId"], match=1):~
21 > print(ad)~
22 ~
23 print("====htcondor2====")~
24 ~
25 print("Schedd Job History:")~
26 for ad in htcondor2.Schedd().history(None, ["ClusterId", "ProcId"], match=1):~
27 > print(ad)~
28 ~
29 print("Epoch History:")~
30 for ad in htcondor2.Schedd().jobEpochHistory(None, ["ClusterId", "ProcId"], match=1):~
31 > print(ad)~
32 ~
33 print("Epoch history w/ type:")~
34 for ad in htcondor2.Schedd().jobEpochHistory(None, [], match=1, ad_type="transfer"):~
35 > print(ad)~
36 ~
37 print("Epoch history w/ list[type]:")~
38 for ad in htcondor2.Schedd().jobEpochHistory(None, [], match=1, ad_type=["output"]):~
39 > print(ad)~
40 ~
41 print("Startd history:")~
42 for ad in htcondor2.Startd().history(None, ["ClusterId", "ProcId"], match=1):~
43 > print(ad)~
44 ~
45 print("Epoch None")~
46 for ad in htcondor2.Schedd().jobEpochHistory():~
47 > print(ad)~
48 > break~
49
```

Documentation

<https://htcondor.readthedocs.io/en/latest/>

The screenshot shows a web browser displaying the HTCondor Version 23.9.6 Manual page. The browser's address bar shows the URL <https://htcondor.readthedocs.io/en/latest/>. The page title is "HTCondor Version 23.9.6 Manual". The main content area contains the following text:

HTCondor Version 23.9.6 Manual

The HTCondor Software Suite (HTCSS) is a software system that creates a High-Throughput Computing (HTC) environment. This environment might be a single cluster, a set of related clusters on a campus, cloud resources, or national or international federations of computers.

If you are a user of HTCondor, and have been given a login or credentials to use a batch scheduler on an Access Point (sometimes called a scheduler or login node), you may want to read our Quick Start guide here: [Users' Quick Start Guide](#)

If you are a beginning administrator of HTCondor, or want to install it for the first time, please look at our installation guide here: [Downloading and Installing](#)

Otherwise, for users of HTCondor who want more information, a complete user's reference manual is here: [Users' Manual](#), and a similar complete reference for administrators of HTCondor can be found here: [Administrators' Manual](#)

HTCondor contains many command line tools, each with a traditional Unix "man-page". These may be found here: [Commands Reference \(man pages\)](#)

Finally, for users writing Python interfaces to HTCondor, our Python API documentation is here: [Python Bindings](#)

A complete table of contents follows.

Manual built on August 8, 2024

Quick start guides

- [Users' Quick Start Guide](#)
 - [What is a Job?](#)
 - [A First HTCondor Job](#)
 - [The science Job Example](#)
 - [Expanding the science Job and the Organization of Files](#)
 - [Where to Go from Here](#)
- [Downloading and Installing](#)
- [Overview](#)
 - [High-Throughput Computing \(HTC\) and its Requirements](#)
 - [HTCondor's Power](#)

The left sidebar contains a search bar and navigation links for "QUICK START GUIDES", "REFERENCE MANUALS", "ADDITIONAL DOCS", and "REFERENCE, GLOSSARY AND INDEX". The bottom of the sidebar shows "Read the Docs" and "v: latest".

Documentation Navigation

<https://htcondor.readthedocs.io/en/latest/>

REFERENCE MANUALS

Users' Manual

Administrators' Manual

ClassAds

DAGMan Workflows

Python Bindings

ADDITIONAL DOCS

Cloud Computing

Grid Computing

Platform-Specific Information

Recipes, Examples, and Other Answers

Version History and Release Notes

REFERENCE, GLOSSARY AND INDEX

Commands Reference (man pages)

ClassAd Attributes

Codes and Other Needed Values

Glossary

Index

Questions?