Schedule

12:30	**Lunch**
13:00	**Part I** • [15 min] Placing lists of jobs • [20 min] What happened to my job? • [15 min] Data placement • [30 min] Troubleshooting strategies • [20 min] GPU jobs
14:40	**Break**
14:50	**Part II** • [40 min] Principles of DAGMan • [40 min] Hands-on: DAGMan • [20 min] Python bindings • [20 min] Hands-on Python bindings • [10 min] Computing at Nikhef • [25 min] Philosophy & architecture
17:25	**Social**

Data placement

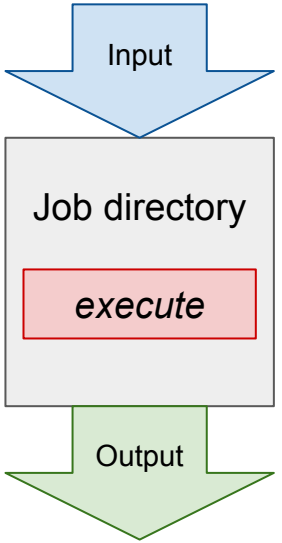
What counts as "data"?

"**Input**" - *any object needed for the job to run*

- `executable`
- `transfer_input_files`
- software environment (`.tar.gz` or container)

"**Output**" - *any object created/modified that you want to have*

- standard output and error (`.out`, `.err`)
- program log files
- generated/processed data or dataset



Where is the data and where is it going?

Wherever the **Input** data is, it will need to be transferred to the **job directory**.

Once the **Output** data is generated, it will need to be transferred from the **job directory** to wherever it needs to go.

	Original Location	*Destination*
Input data	**Many places**	Job directory
Output data	Job directory	**Many places**

How the data is transferred depends on a couple of factors

Considerations for data movement

Where you place the data and how you move it depends on:

- Size/number of objects per job
  - Are you transferring many small files? Or one big file? Or a mix?
- Number of jobs
  - How many times will you need to repeat the transfer process?
- Job runtime
  - Is the job runtime short (couple minutes) or long (hours - days)?
- Uniqueness
  - Are you using the exact same files for each job? Or does each job have its own unique file?

The collaborator analogy

What method would you use to send data to a collaborator?

amount	method of delivery
words	email body
tiny – 100MB	email attachment (managed transfer)
100MB – GBs	download from Google Drive, Drop/Box, other web-accessible repository
TBs	ship an external drive (local copy needed)

Never underestimate the bandwidth of a station wagon full of tapes hurtling down the highway.

Andrew S. Tanenbaum (1981) – Professor Emeritus, Vrije Universiteit Amsterdam

Data transfer on HTC

How can we move data on a HTC system?

amount	method of delivery
words	Executable or submit; print to `.out`
tiny – 1GB	Default HTCondor file transfer (up to 1GB total per job)*
1GB – 20GBs	OSDF (regional distribution), cloud storage
20 GB – TBs	shared file system (local copy, local execute servers)

**Best to have < ~10 total objects; use `tar`/`zip` to your advantage!*

Data transfer on HTC

How can we move data on a HTC system?

amount	method of delivery
words	Executable or submit; print to `.out`
tiny – 1GB	Default HTCondor file transfer (up to 1GB total per job)*
1GB – 20GBs	OSDF (regional distribution), cloud storage
20 GB – TBs	shared file system (local copy, local execute servers)

**Best to have < ~10 total objects; use `tar`/`zip` to your advantage!*

Data transfer mechanisms - default file transfer

HTCondor has a built-in mechanism for handling file transfers to/from the Access Point (AP)

	Original Location	*Destination*
Input data	**Access Point**	Job directory
Output data	Job directory	**Access Point**

The transfer of these files is managed by HTCondor on the AP

Avoid overwhelming the access point with file transfers!
→ keep the total size under 1 GB and total number of objects < ~10, per job per direction

Data transfer mechanisms - default file transfer

HTCondor has a built-in mechanism for handling file transfers to/from the AP

In your submit file, use the following to declare your **Input** files:

```
transfer_input_files = input1, input2
```

Use the following to declare your **Output** files:

```
transfer_output_files = output1, output2
```

*If you do NOT declare any **Output** files, then by default HTCondor transfers *any new or changed file (but not directories) from the job*

Data transfer on HTC

How can we move data on a HTC system?

amount	method of delivery
words	Executable or submit; print to `.out`
tiny – 1GB	Default HTCondor file transfer (up to 1GB total per job)*
1GB – 20GBs	OSDF (regional distribution), cloud storage
20 GB – TBs	shared file system (local copy, local execute servers)

**Best to have < ~10 total objects; use `tar`/`zip` to your advantage!*

Data transfer mechanisms - OSDF

Open Science Data Federation (OSDF)

↓

a system of data caches that can stage large, repeatedly used files closer to the actual compute resources

From https://osg-htc.org/services/osdf.html, numbers listed are worldwide totals

Data transfer mechanisms - OSDF

Let's say you are submitting jobs in Amsterdam, but they are running in Geneva

From https://osg-htc.org/services/osdf.html, numbers listed are worldwide totals

Data transfer mechanisms - OSDF

If submitting a lot of jobs, can turn into a lot of network traffic

10,000 jobs
x
10 GB input file
x
1 transfer / job
=
100,000 GB
network transfer

From https://osg-htc.org/services/osdf.html, numbers listed are worldwide totals

Data transfer mechanisms - OSDF

Using the OSDF, a copy of the file is placed in a nearby cache, reducing the overall network traffic

10,000 jobs
x
10 GB input file
x
*1 transfer **total***
=
10 GB
*network transfer**

From https://osg-htc.org/services/osdf.html, numbers listed are worldwide totals

Data transfer mechanisms - OSDF

HTCondor has a file transfer plugin for moving objects via the OSDF

	Original Location	*Destination*
Input data	**OSDF**	Job directory
Output data	Job directory	**OSDF**

The transfer of these files is managed by the plugin at the Execution Point (EP)

Data transfer mechanisms - OSDF

HTCondor has a file transfer plugin for moving objects via the OSDF

For **Input** objects:

1) Place your data in an OSDF "origin"
2) In your submit file, declare that you are using the OSDF (the `osdf://` prefix) and provide the OSDF object address:

```
transfer_input_files = osdf:///origin_namespace/object
```

*for authenticating access, may need to generate and include a token

Data transfer mechanisms - OSDF

HTCondor has an OSDF plugin to handle transfers

For **Input** objects

a) The OSDF plugin (client) looks for a nearby cache that already has a copy of the object
b) If a nearby cache does not have the object, gets a copy from the origin

Data transfer mechanisms - OSDF

HTCondor has an OSDF plugin to handle transfers

For **Output** objects:

1) In your submit file, declare the output files that need to be transferred:

```
transfer_output_files = my_object
```

2) In your submit file, also declare the specific output files that need to be "remapped" to the OSDF location:

```
transfer_output_remaps = "my_object = osdf:///origin_namespace/my_object"
```

*for authenticating access, may need to generate and include a token
**multiple items in the "remaps" are separated by semicolons (;)

Data transfer on HTC

How can we move data on a HTC system?

amount	method of delivery
words	Executable or submit; print to `.out`
tiny – 1GB	Default HTCondor file transfer (up to 1GB total per job)*
1GB – 20GBs	OSDF (regional distribution), cloud storage
20 GB – TBs	shared file system (local copy, local execute servers)

**Best to have < ~10 total objects; use `tar`/`zip` to your advantage!*

Data transfer mechanisms - shared file system

HTCondor can use a file transfer plugin to transfer files within a shared file system

	Original Location	*Destination*
Input data	**Shared File System**	Job directory
Output data	Job directory	**Shared File System**

You can either

(a) declare the file movements in the submit file, or

(b) copy/read/write directly to the shared file system

Depends on your system and your admin's policies

Data transfer mechanisms - shared file system (a)

(a) declare the file movements in the submit file

For **Input** objects:

1) Place your data in the shared file system (follow admin instructions)
2) In your submit file, declare the path in the shared file system using the `file://` plugin:

```
transfer_input_files = file:///shared_file_system/path/to/file
```

HTCondor (on the EP) will copy the file to the job directory

Data transfer mechanisms - shared file system (a)

(a) declare the file movements in the submit file

For **Output** objects:

1) In your submit file, declare the output files that need to be transferred:

```
transfer_output_files = my_object
```

2) In your submit file, also declare the specific output files that need to be "remapped" to the shared file system:

```
transfer_output_remaps = \
     "my_object = file:///shared_file_system/path/to/my_object"
```

Data transfer mechanisms - shared file system (b)

(b) copy/read/write directly to the shared file system

In your executable script, have explicit copy/read/write commands

```
cp /shared_file_system/path/to/file ./
./my_command --input_dir=./ --output_dir=./
cp ./large_out.tar.gz /shared_file_system/path/to/large_out.tar.gz
# or maybe...
./my_command \
    --input_dir=/shared_file_system/path/ \
    --output_dir=/shared_file_system/path/
```

Whether you should copy, or should read/write directly, depends on your system's policies (and maybe size of data)!

Data transfer on HTC

How can we move data on a HTC system?

amount	method of delivery
words	Executable or submit; print to `.out`
tiny – 1GB	Default HTCondor file transfer (up to 1GB total per job)*
1GB – 20GBs	OSDF (regional distribution), cloud storage
20 GB – TBs	shared file system (local copy, local execute servers)

HTCondor can manage the transfer of data in almost all cases!

Data transfer failures

An error in the data transfer means an error in the job submission process.

That means most errors are captured by HTCondor in some fashion* as **hold messages**.

Two main types of errors:

1) `Transfer input file failure`
2) `Transfer output file failure`

Most common of both of these is "No such file or directory"

*whether the error is nicely explained by the plugin is a different story

Data transfer failures

Use `condor_q -hold` to see the reason:

```
$ condor_q -hold

ap40.uw.osg-htc.org : <128.105.68.62:9618?... @ 08/01/24 15:01:50
 ID          OWNER           HELD_SINCE  HOLD_REASON
130.0        alice            8/1  14:56 Transfer input files failure at …
```

Questions?

What powers the OSDF?

Pelican Platform

www.pelicanplatform.org

Just like how OSG uses

HTCondor as the software that runs the *OSPool,*

OSG is transitioning to use

Pelican as the software that runs the *OSDF.*

What powers the OSDF?

Like HTCSS, the **Pelican Platform** is an open-source software being developed at CHTC (Center for High Throughput Computing) at University of Wisconsin – Madison

Overall goal for Pelican includes:

- Make it easy to deploy and manage systems like the OSDF
- Provide a single protocol for users to access data (regardless of storage location)
- Make it easy for data owners to share their data

Want to learn more? Please talk to Andrew for more info

More info about Pelican: HTC24 talks

- "Deployment Scale and Use of OSDF" session: https://agenda.hep.wisc.edu/event/2175/contributions/30968/
- "Introducing Pelican: Powering the OSDF" https://agenda.hep.wisc.edu/event/2175/contributions/30967/
- "Pelican Under the Hood: How the Data Federation Works" https://agenda.hep.wisc.edu/event/2175/contributions/31334/
- "Connecting Pelican to your data" https://agenda.hep.wisc.edu/event/2175/contributions/31335/
- "Data in Flight: Delivering Data with Pelican – Tutorial" https://agenda.hep.wisc.edu/event/2175/contributions/31337/

Schedule

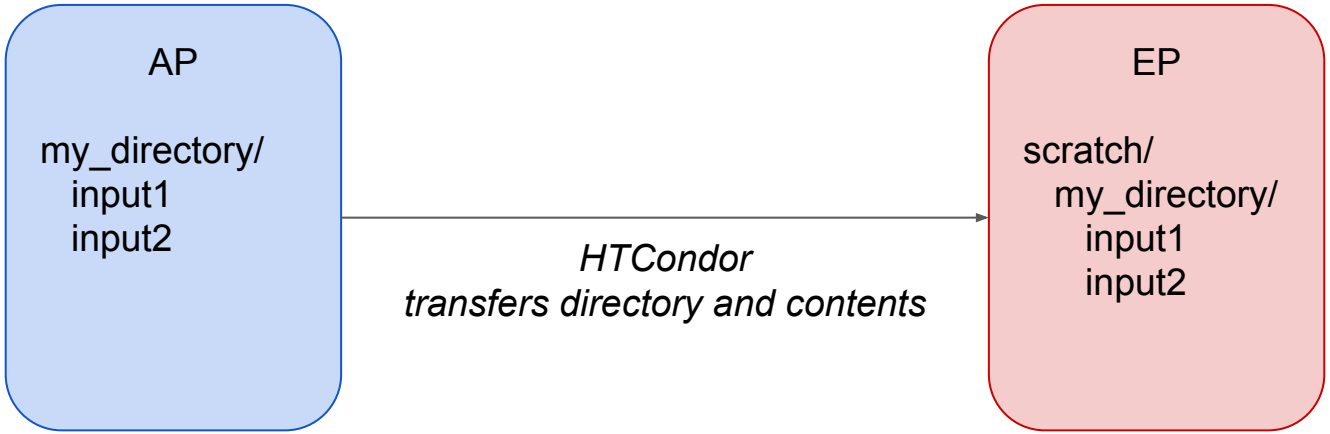
12:30	**Lunch**
13:00	**Part I** • [15 min] Placing lists of jobs • [20 min] What happened to my job? • [15 min] Data placement • [30 min] Troubleshooting strategies • [20 min] GPU jobs
14:40	**Break**
14:50	**Part II** • [40 min] Principles of DAGMan • [40 min] Hands-on: DAGMan • [20 min] Python bindings • [20 min] Hands-on Python bindings • [10 min] Computing at Nikhef • [25 min] Philosophy & architecture
17:25	**Social**

Data transfer mechanisms - default file transfer

If transferring a directory (let's call it `my_directory`):

Providing just the directory name will transfer the **directory *and* its contents**.

```
transfer_input_files = my_directory
```

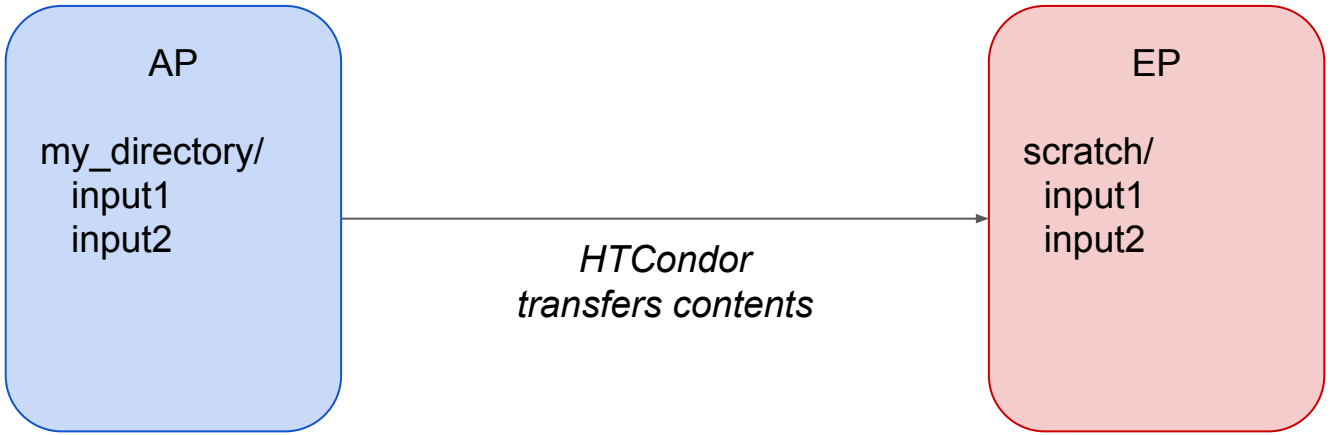


Data transfer mechanisms - default file transfer

If transferring a directory (let's call it `my_directory`):

Providing the directory name with a trailing slash will transfer ***only* the directory's *contents***.

```
transfer_input_files = my_directory/
```



Data transfer mechanisms - cloud

HTCondor has a variety of file transfer plugins for moving objects via the cloud

	Original Location	*Destination*
Input data	**Cloud Storage**	Job directory
Output data	Job directory	**Cloud Storage**

The transfer of these files is managed by the plugin at the EP

Data transfer mechanisms - cloud

HTCondor has a variety of file transfer plugins for moving objects via the cloud

Built in plugins:

- `http://`
- `https://`
- `ftp://`
- `box://`
- `gdrive://` (Google Drive)
- `stash://`
- `onedrive://` (Microsoft OneDrive)

Mechanism is similar to use of the `osdf://` plugin, although authentication method varies.

Data transfer mechanisms - cloud

HTCondor has a variety of file transfer plugins for moving objects via the cloud

… but you can use Pelican and create an Origin to contribute to the OSDF

Advantages of setting up a Pelican Origin:

- using the cache system in the OSDF reduces the load on the data source
- users just need to learn one plugin
- can configure authentication at the Pelican Origin
    - Makes it easier for the users to set up their submit files

Data transfer failures - input example

Full message:

> Transfer input files failure at access point ap40 while sending files to the execution point. Details: reading from file /home/andrew.owen/test/hold_this_job.txt: (errno 2) No such file or directory

Data transfer failures - input example

Full message:

> **Transfer input files failure at access point ap40** while sending files to the execution point. Details: reading from file **/home/andrew.owen/test/hold_this_job.txt**: (errno 2) **No such file or directory**

The file at that path doesn't exist at the AP. Could be

(a) file doesn't exist anywhere on the AP

(b) file is in a different directory on the AP

(c) the submit file has a typo in the file path in `transfer_input_files` line

(a) In this case, I deliberately referenced a file `hold_this_job.txt` that I knew didn't exist

Data transfer failures - output example

Full message:

> **Transfer output files failure** **at the execution point** while sending files to access point ap40. Details: reading from file **/var/lib/condor/execute/hold_this_job.txt**: (errno 2) **No such file or directory**

The file at that path doesn't exist at the EP. Could be

(a) file doesn't exist anywhere on the EP

(b) file is in a different directory on the EP

(c) the submit file has a typo in the file path in `transfer_output_files` line

(a) In this case, I deliberately referenced a file `hold_this_job.txt` that I knew didn't exist

Data transfer failures

Most common error is "**No such file or directory**"

Other errors include

- Permission denied (file exists but your user doesn't have permission to touch)
- Quota exceeded / no room left on device
- Transfer timed out/transfer too slow
- Network error
- Plugin error
  - For example, OSDF client error

hopefully "try again later" errors - if persistent, reach out to admin/facilitator

What are the many places data can live?