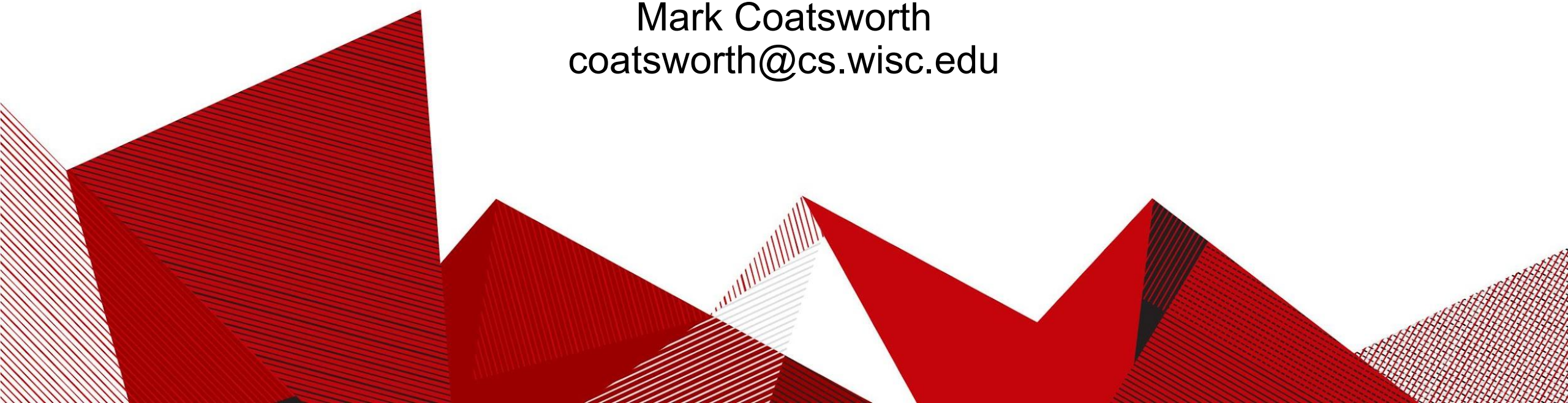
File Transfer Plugins

Why You Want Them and How To Write Them

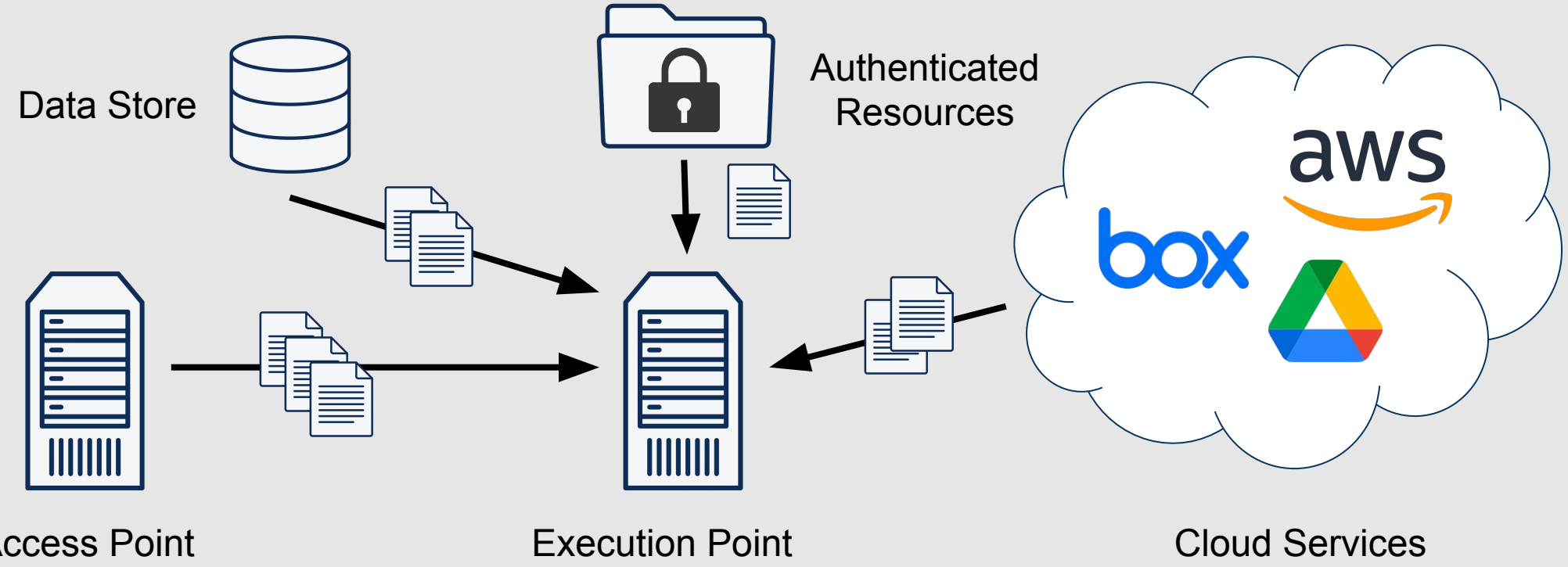
European HTCondor Workshop 2021

Mark Coatsworth
coatsworth@cs.wisc.edu



File Transfer in HTCondor

- Interesting problem in HTCondor: how to move your files to the execution points where your jobs run



File Transfer in HTCondor

- Several different mechanisms to move files around:
  - Condor file transfer
  - Shared file systems
  - Job wrapper scripts
- **Some of these are really bad!**

File Transfer in HTCondor

- Several different mechanisms to move files around:
  - Condor file transfer
  - **Shared file systems**
  - **Job wrapper scripts**
- **Some of these are really bad!**
- In this talk I'm going to go over some of the problems with these other methods
- Make a case for why you should be using custom file transfer plugins

Shared File Systems

- One widely used file transfer approach: shared file systems
- These are **convenient**: your files are available at same location from the access point and every execution point
- They are also **dangerous:** lots of things can go wrong at any time, with no indication of what or why
  - Errors can happen after your job has been running for 100 hours
  - Writes are not idempotent, can break job restarts
- They are **non-portable:** if your execution point doesn't have access to shared FS, your jobs cannot run

Shared File Systems

- Worst of all, shared file systems are **unmanaged:** HTCondor has no control over what they are doing
- Unable to measure file transfers, report statistics, or take action if your files are not transferring correctly
- No concept of "I'm too busy" or "maybe you could get that file faster somewhere else"
- Input files get misidentified as generated and transferred

Shared File Systems

- **Errors difficult to identify and explain!**
- Difficult to propagate back to the user
- Condor cannot manage file transfer errors (by restarting job)

HTCondor File Transfer

- File transfers that are **managed** by HTCondor
- Used by specifying a list of files in your submit description that should get moved from the access point

```
transfer_input_files = input1.dat, input2.dat, input3.dat
should_transfer_files = yes
```

- This is a good approach! HTCondor is able to identify errors, throttle overloaded access points, report useful statistics

HTCondor File Transfer

- Unfortunately, there are still limitations with HTCondor file transfer
- All files come from the access point, which is a choke point
- If transferring 100 GB of files to 10,000 jobs, every single byte of those files gets sent to every single job
- No support for any special mechanisms, like authentication or transferring files out of Amazon S3

Job Wrapper Scripts

- Do not do this! It's a terrible idea!
- Unfortunately some jobs are already doing this
  - I'm here to show you a better way
- You can wrap your job executable in a script which also transfers any input files
- **Good news:** A user has more control to transfer files however they want
- **Bad news:** A user has more control to transfer files however they want

Job Wrapper Scripts

- Needless to say, job wrapper scripts are **unmanaged** by HTCondor.
- ~~If~~ When things go terribly wrong, HTCondor will be unable to identify what or why.
- No clear knowledge of input/output files
- Do not do this! It's a terrible idea!
- **Fortunately, there is a better way....**

File Transfer Plugins

- File transfer plugins are typically small scripts that programmatically move files however you want
- Allows you to include input files with a URL syntax:

```
transfer_input_files = myplugin://pages.cs.wisc.edu/~coatsworth/input1.dat,
myplugin://pages.cs.wisc.edu/~coatsworth/input2.dat
```

- When Condor sees the URL syntax (**method** followed by **://**) it realizes that you are asking for a file transfer plugin
- It then invokes that plugin and passes in the full URL

HTCondor

File Transfer Plugins

- File transfer plugins are **managed** by HTCondor
  - If anything goes wrong, the job will go on hold
  - Strong mechanisms to describe error in the hold message
- By default, HTCondor provides file transfer plugins for HTTP transfers (http://), Box (box://), Google Drive (gdrive://) and OneDrive (onedrive://)
- **However, you can also provide your own!**

File Transfer Plugins: How They Work

- How exactly do they work?
- Let's use an example: HTTP transfers

```
transfer_input_files = http://pages.cs.wisc.edu/~coatsworth/input1.dat,
http://pages.cs.wisc.edu/~coatsworth/input2.dat
```

- HTCondor sees the **http://** prefix and knows it is looking for a file transfer plugin.
- Looks at the FILETRANSFER_PLUGINS configuration:

```
FILETRANSFER_PLUGINS = /usr/libexec/condor/curl_plugin,
/usr/libexec/condor/data_plugin
```

File Transfer Plugins: How They Work

- Asks these file transfer plugins which methods they support:

```
$ /usr/libexec/condor/curl_plugin -classad
MultipleFileSupport = true
PluginVersion = "0.2"
PluginType = "FileTransfer"
SupportedMethods = "http,https,ftp,file"
```

File Transfer Plugins: How They Work

- Next, HTCondor wraps up your request in a list of classads
- Each transfer described by two attributes:
  - **Url**
  - **LocalFileName:** this is how HTCondor tells your plugin where to put the file on the local system

```
[ LocalFileName = "/var/lib/condor/execute/slot1/dir_11121/input1.dat"; Url =
"http://pages.cs.wisc.edu/~coatsworth/input1.dat" ][ LocalFileName =
"/var/lib/condor/execute/slot1/dir_11121/input2.dat"; Url =
"http://pages.cs.wisc.edu/~coatsworth/input2.dat" ]
```

File Transfer Plugins: How They Work

- Next, HTCondor hands the list of classads to the curl_plugin that matched your http:// method

```
$ /usr/libexec/condor/curl_plugin -infile .curl_plugin.in -outfile .curl_plugin.out
```

- At this point, the plugin can do whatever it wants to transfer these files
  - In the case of http transfers, we simply use the curl library to access the file
  - But you can get creative!

File Transfer Plugins: How They Work

- Lastly, HTCondor expects your plugin to produce an output file with the results
- Each transfer must be described with a classad containing the following attributes:
  - **TransferUrl**
  - **TransferSuccess**
  - **TransferError** (this is how error message gets propagated)

```
$ cat .curl_plugin.out
[ TransferUrl = "http://pages.cs.wisc.edu/~coatsworth/input1.dat"; TransferSuccess =
true; ][ TransferUrl = "http://pages.cs.wisc.edu/~coatsworth/input2.dat";
TransferSuccess = false; TransferError = "Error: The requested URL returned error: 404
File Not Found"]
```

File Transfer Plugins: How They Work

- Error handling is the responsibility of the plugin, not HTCondor
  - Any timeouts, failure to transfer files, other issues must be handled by the plugin
  - Error/hold messages also responsibility of the plugin

File Transfer Plugins: How to Write Them

- How do you write a custom plugin?
- First, choose a programming language.
  - You'll need to use ClassAds, so ideally Python or C++.
  - Think if you need third party libraries to access external services
- Start with our Python example template: https://github.com/htcondor/htcondor/blob/master/src/condor_examples/filetransfer_example_plugin.py
  - Provides the structure needed to integrate with HTCondor
  - All you need to do is add your own custom transfer logic

File Transfer Plugins: How to Write Them

- Some other examples available at https://github.com/htcondor/htcondor/tree/master/src/condor_scripts
- How to transfer credentials?
  - The Google Drive and OneDrive plugins are useful references for how to use authentication tokens along with your plugin
- Usage of these plugins is described in the manual: https://htcondor.readthedocs.io/en/latest/users-manual/file-transfer.html#file-transfer-using-a-url

File Transfer Plugins

- How do you get your custom plugin onto the execution points in your pool?
- **Option 1: Call your friendly local system administrator**
  - Ask them to put the plugin script on every execution point in your pool, then add it to the FILETRANSFER_PLUGINS list
- However this makes some big assumptions:
  - Your system administrator likes you (and has time to do this)
  - You have access to the execution points where your jobs will be running. This doesn't work in many places.

File Transfer Plugins

- **Option 2: Bring Your Own Plugin (BYOP) method**
- You can ship a transfer plugin along with your job using the following in your submit file:

```
transfer_plugins = myexample=example_plugin.py
transfer_input_files = myexample://path/to/file1, myexample://path/to/file2
```

The End

- File transfer plugins are a great improvement to using shared file systems or wrapper scripts.
- The manual provides more detailed information about how plugins work: https://htcondor.readthedocs.io/en/latest/admin-manual/setting-up-special-environments.html#enabling-the-transfer-of-files-specified-by-a-url
- Questions?