Throughput Computing 2023

OSG All-Hands Meeting   HTCondor Week

Getting Your Data with HTCondor

July 12, 2023

Mátyás ("Mat") Selmeci

Justin Hiemstra

PATh PARTNERSHIP to ADVANCE THROUGHPUT COMPUTING

Getting Your Data *With* HTCondor

- This talk is about why you should let HTCondor manage jobs' data transfer, and how you can do so
- Todd Tannenbaum gave this talk four years ago: https://indico.cern.ch/event/817927/contributions/3570472/
- The principles are the same, but we've done some new work since then, and have plans for the future...

If you just wanted to use your data in a job...

You could use a shared file system...

- If you're just running locally, you could read/write in your jobs to a shared file system
- Shared file systems are nice and convenient when everything works
- Every file is (as far as you can tell) already there

You could download stuff right in the job...

- Put some curl or wget calls in your job script

- Works on your laptop, works on most sites most of the time

... so why shouldn't you?

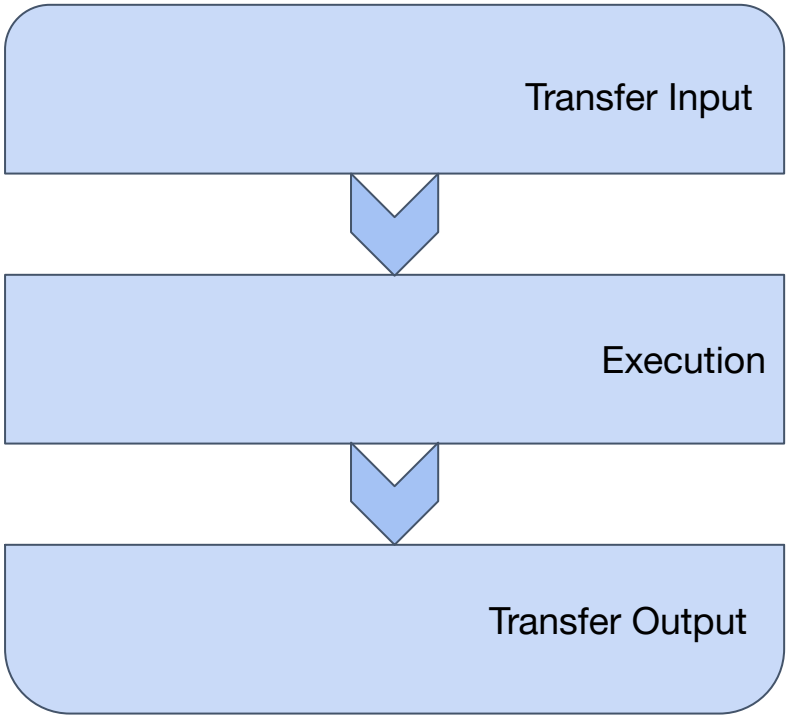
- Shared file systems are nice and convenient **when everything works**
- curl or wget work on **most** sites **most** of the time
- Failures happen -- how do you spot them?

How do you debug failures?

- If everything is being done in the middle of the job, all failures look like job failures

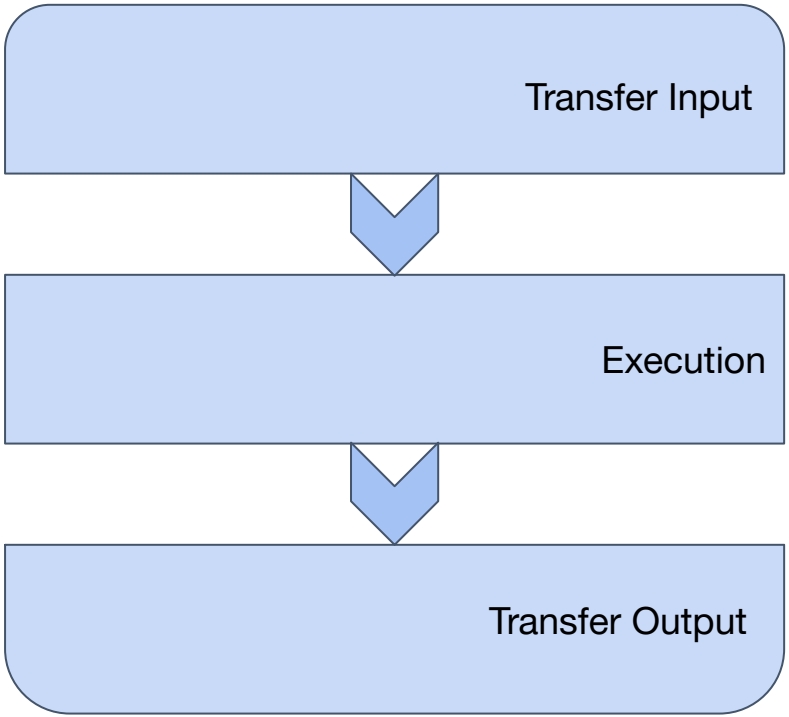
- Couldn't read input data? Job terminates early, go dig through the logfiles to figure out why
- Program crash? Job terminates early, go dig through the logfiles to figure out why
- Couldn't write output data? Job terminates without output data, go dig through the logfiles to figure out why

Jobs are in stages



If one stage fails, do not go to on to the next.

Then, HTCondor can manage the stages



Transfer_Input_Files = input1.dat,input2.dat

Executable = process_all_the_things.py

Transfer_Output_Files = output.csv

What input failures look like

- A failure in the Transfer Input stage results in a job going on hold. condor_q -hold will show something like:

1.0 submituser 7/11 06:16 **Transfer input files failure** at execution point slot1@mini using protocol https. Details: The requested URL returned error: **404 Not Found** (URL file = **https://pages.cs.wisc.edu/~matyas/nonexistant-input**)|

2.0 submituser 7/11 06:17 **Transfer input files failure** at access point mini while sending files to execution point slot1@mini. Details: reading from file **/home/submituser/nonexistant-input**: (errno 2) **No such file or directory**

What output failures look like

- A failure in the Transfer Output stage results in a job going on hold. condor_q -hold will show something like:

3.0 submituser 7/11 06:19 **Transfer output files failure** at execution point slot1@mini while sending files to access point mini. Details: reading from file **/var/lib/condor/execute/dir_568/my-nonexistent-output**: (errno 2) **No such file or directory**

How to do have HTCondor do it

- File transfer from the AP (via HTCondor built-in file transfer)
- File transfer from outside the AP (via URL-based file transfer)
- OSDF

- Each with pros and cons
- Can mix and match

HTCondor's built-in file transfer

- transfer_input_files=file1,file2,directory1
- Transferred from/to the Access Point's file systems via HTCondor's own network protocols
- Pros:
  - Nothing for the admin to set up
  - Straightforward to use - user lists file paths from the AP's disk
- Cons:
  - Puts load on the Access Point
  - No caching

HTCondor's built-in file transfer is good for:

- Files that frequently change
  - Job scripts, programs
  - Lack of caching is an advantage, not a drawback
- Smaller file sets
  - "Smaller" defined between you and your AP admin
    - Data point: Current OSPool APs have 20 Gbps uplink; we tell OSPool AP users: 'max 1 GB per job via built-in file transfer'

URL/Plugin based file transfer

- transfer_input_files=https://host.com/file1,https://host.com/file2
- Self-hosted or third-party-hosted
  - HTTP(S), WebDAV, FTP, S3 (protocol)
- Cloud
  - OneDrive, Google Drive, Amazon S3, Box.com
- This is how you replace curl/wget

URL/Plugin based file transfer

- Pros:
  - Standardized protocols - features like HTTP caching may be available
  - Can leverage existing infrastructure provided by your institution or a cloud vendor
  - Data does not need to live on the Access Point
- Cons:
  - AP admin needs to set up auth for private data or for writing output
  - AP admin needs to obtain API keys for cloud services -- different instructions for each service

OSDF

- transfer_input_files=osdf:///chtc/PUBLIC/matyas/input1.dat
- See Brian and Fabio's talk https://agenda.hep.wisc.edu/event/2014/contributions/28482/
- Widely deployed caching infrastructure for Open Science
- Origins often deployed along with Access Points
- Pros:
  - Handles large reusable files (containers, common data sets) well
- Cons:
  - Write-once: if you want to change a file, also change the name
  - Auth needs to be set up for writing and reading private data

Those were the basics...
What's new? What's changed?

New Since HTCondor 9.0.X

- **Google Cloud support:** We've added support for `gs://`-style Google Cloud Storage URLs, with the corresponding `gs_access_key_id_file` and `gs_secret_access_key_file` aliases. Available in 9.1.3
- **Improved error messages:** When jobs experience a file transfer error and are placed on hold, the `HoldReason` is now set to indicate whether the error was an error transferring input or output (`TransferInputError/TransferOutputError`). Available in 9.11.1
- **Introduction of new file transfer plugins:** HTCondor now supports file transfers using `stash://` & `osdf://` URLs. Plus, these files are managed by HTCondor! Available in 10.0.1
- **Introduction of `MAX_FILE_TRANSFER_PLUGIN_LIFETIME`:** File-transfer plug-ins may no longer take as long as they like to finish. Available in 10.2.0
- **Behavior changed for submit file's `output_destination`:** Logs generated through HTCondor's stdout and stderr are no longer transferred to `output_destination` – instead they now return to submit directory. Available in 10.3.0

Gearing Up for What's Next…

- **Integration with LotMan library:** Admins will have more control over jobs that spool input.
  - Jobs requesting unavailable amounts of disk will be held before files are transferred to spool
  - Additional metrics supplied for admins curious about how spool is being used (or maybe why it's filling up)

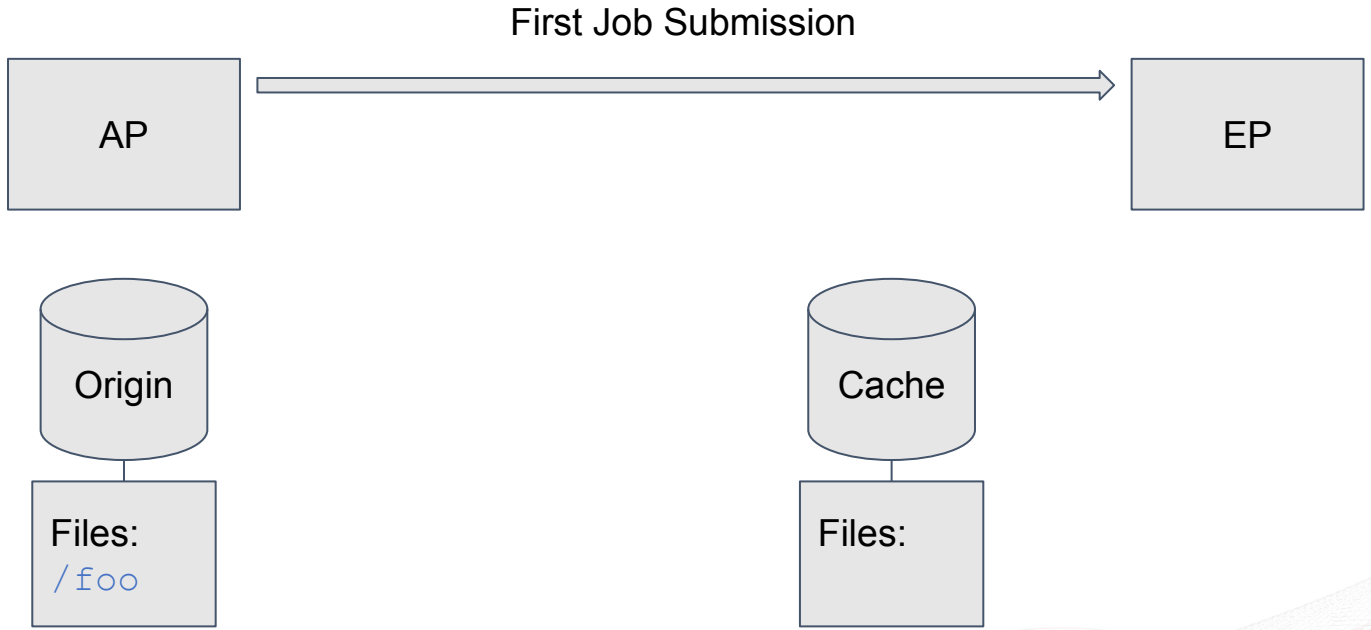
Gearing Up for What's Next…

- More & better transfers with OSDF plug-in:
  - Addressing issues of object immutability

Gearing Up for What's Next…

- More & better transfers with OSDF plug-in:
  - Addressing issues of object immutability

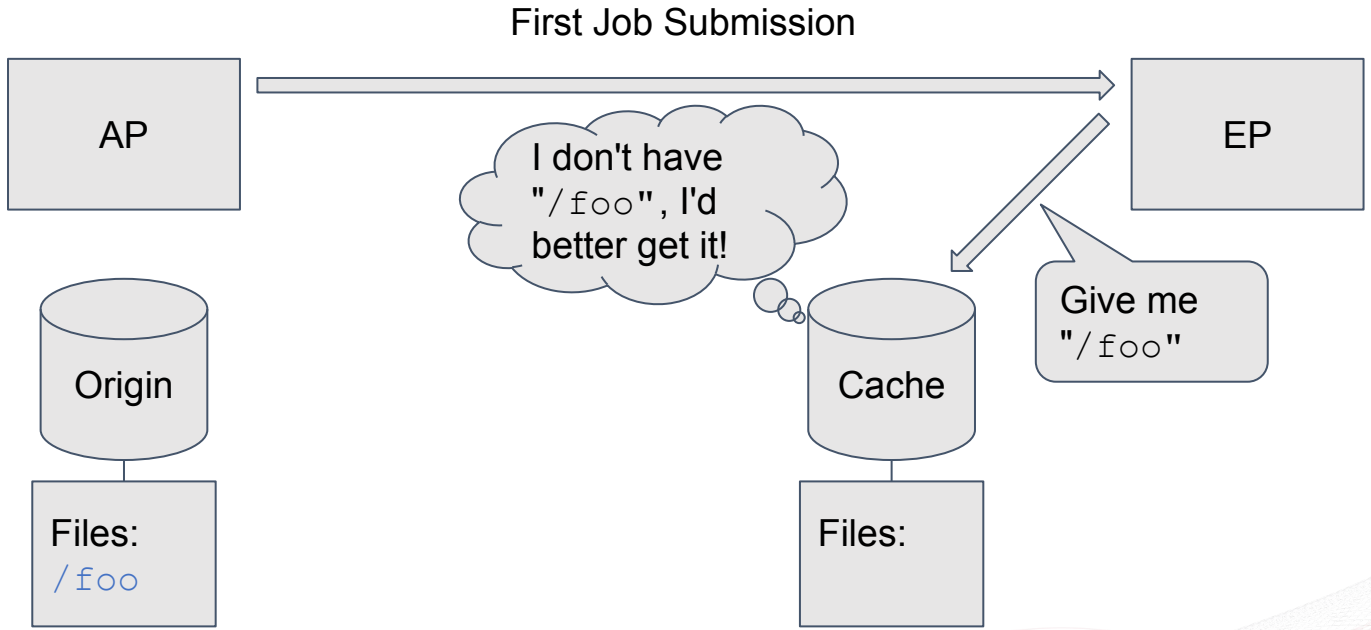
1. User submits a job, specifying `/foo` as an OSDF input



Gearing Up for What's Next…

- More & better transfers with OSDF plug-in:
  - Addressing issues of object immutability

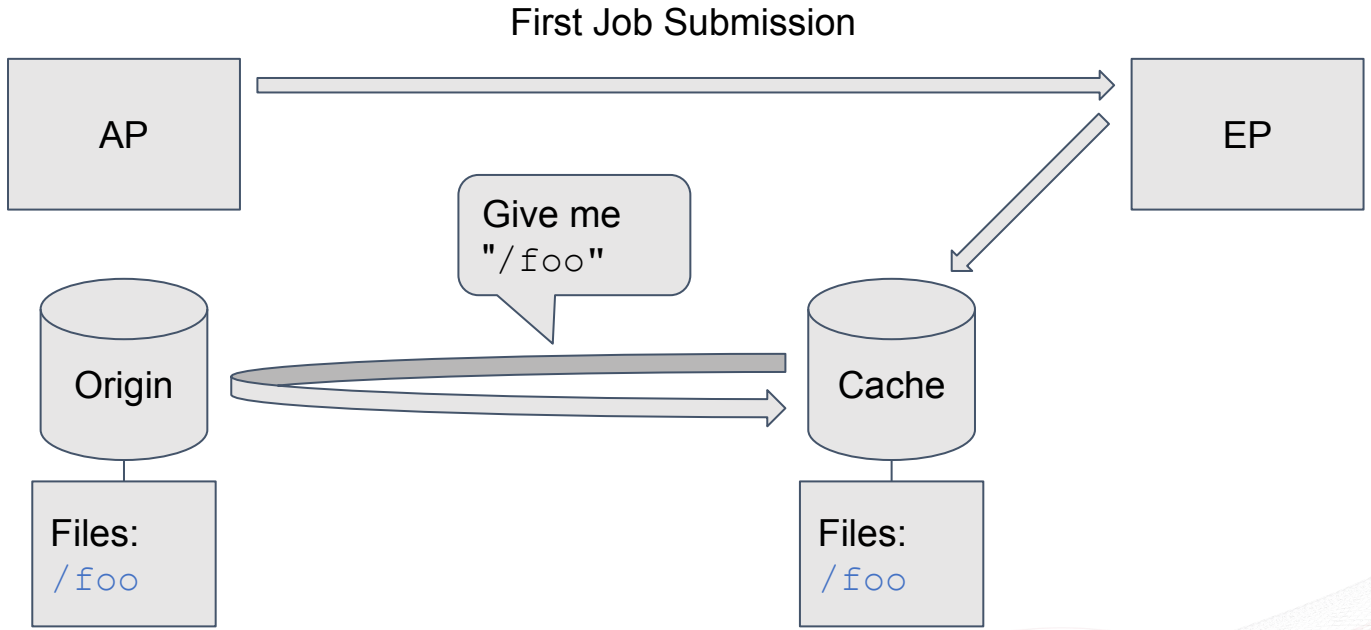
1. User submits a job, specifying `/foo` as an OSDF input
2. EP requests "`/foo`" from nearest cache



Gearing Up for What's Next…

- More & better transfers with OSDF plug-in:
  - Addressing issues of object immutability

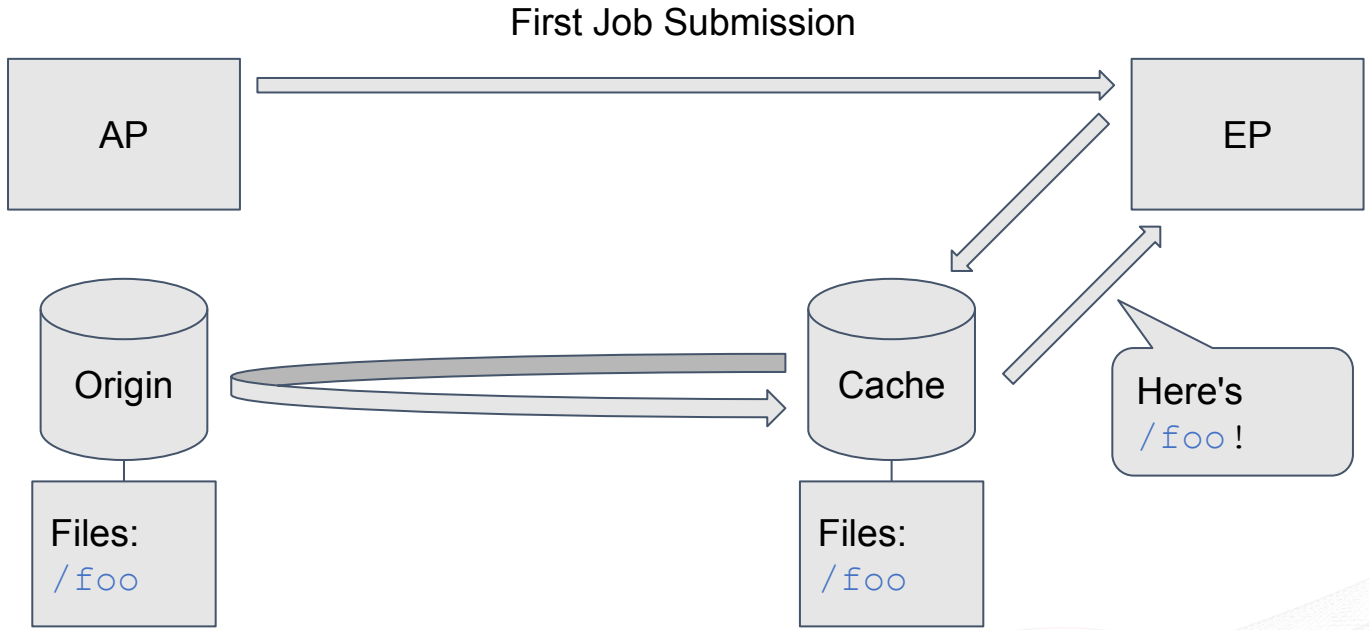
1. User submits a job, specifying `/foo` as an OSDF input
2. EP requests "`/foo`" from nearest cache
3. The cache doesn't have `/foo`, so it requests the object from the origin



Gearing Up for What's Next…

- More & better transfers with OSDF plug-in:
  - Addressing issues of object immutability

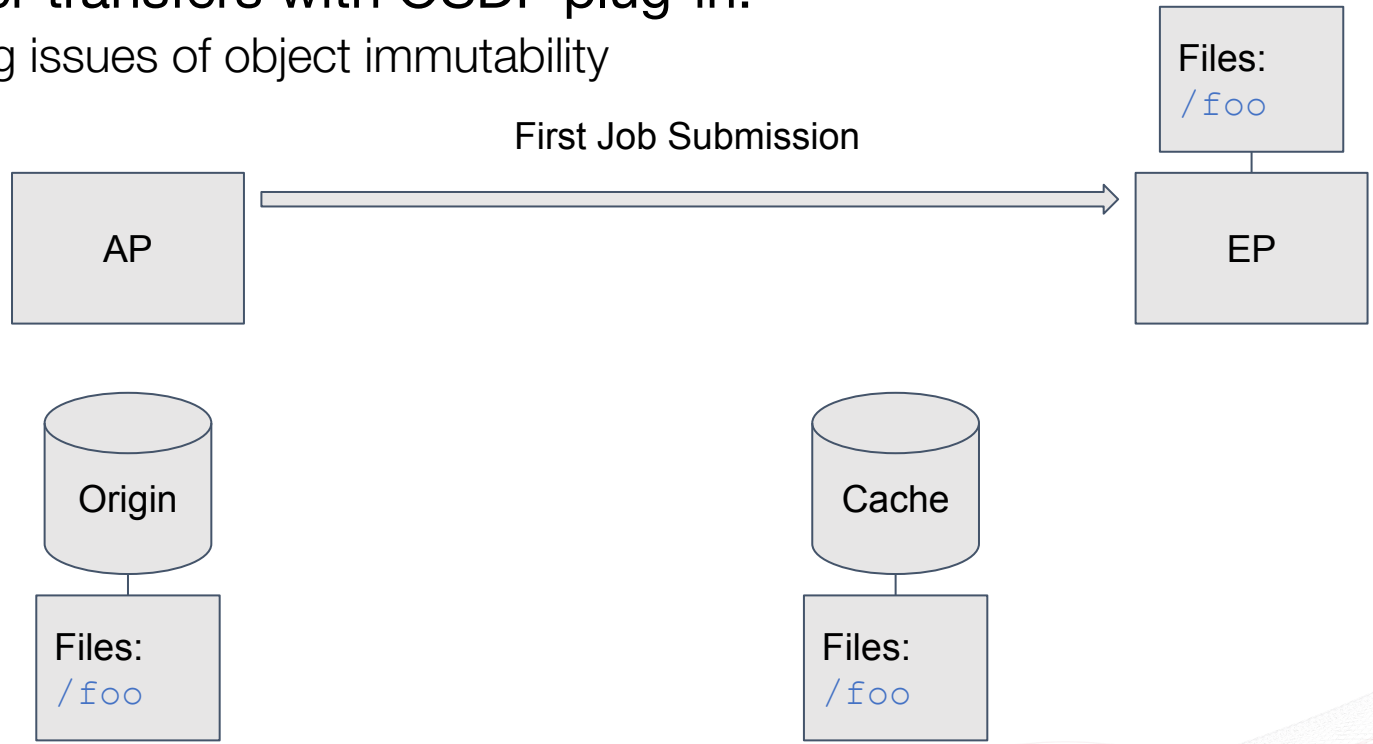
1. User submits a job, specifying `/foo` as an OSDF input
2. EP requests "`/foo`" from nearest cache
3. The cache doesn't have `/foo`, so it requests the object from the origin
4. The cache now delivers `/foo` to the EP



Gearing Up for What's Next…

- More & better transfers with OSDF plug-in:
  - Addressing issues of object immutability

1. User submits a job, specifying `/foo` as an OSDF input
2. EP requests "`/foo`" from nearest cache
3. The cache doesn't have `/foo`, so it requests the object from the origin
4. The cache now delivers `/foo` to the EP
5. The job continues on its merry way

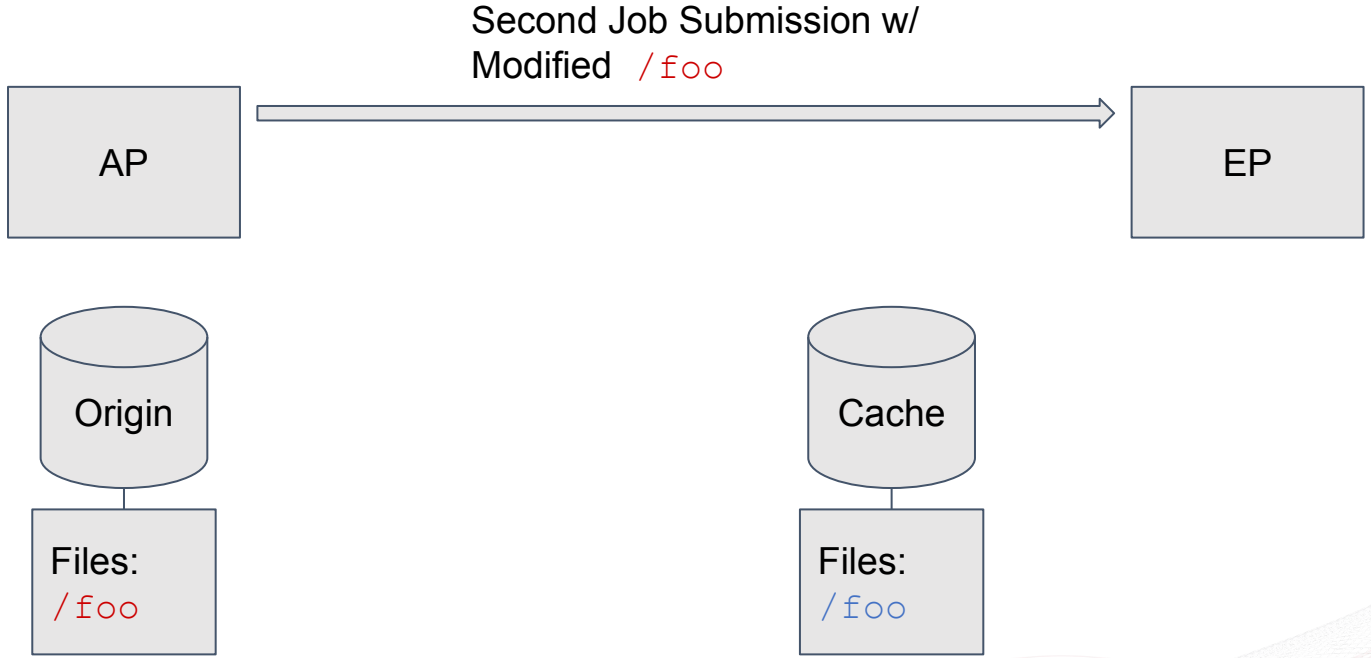


The user realises there was a mistake in `/foo`, so they modify its contents, producing `/foo`…

Gearing Up for What's Next…

- More & better transfers with OSDF plug-in:
  - Addressing issues of object immutability

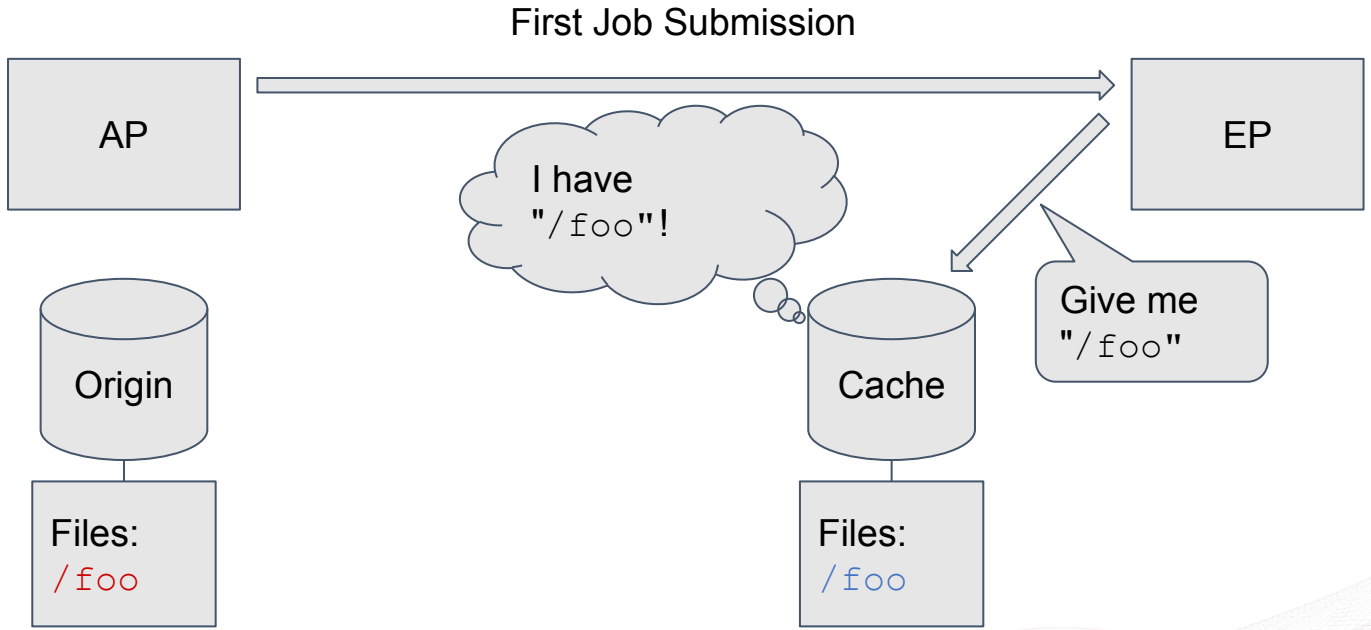
1. User submits a job, specifying `/foo` as an OSDF input



Gearing Up for What's Next…

- More & better transfers with OSDF plug-in:
  - Addressing issues of object immutability

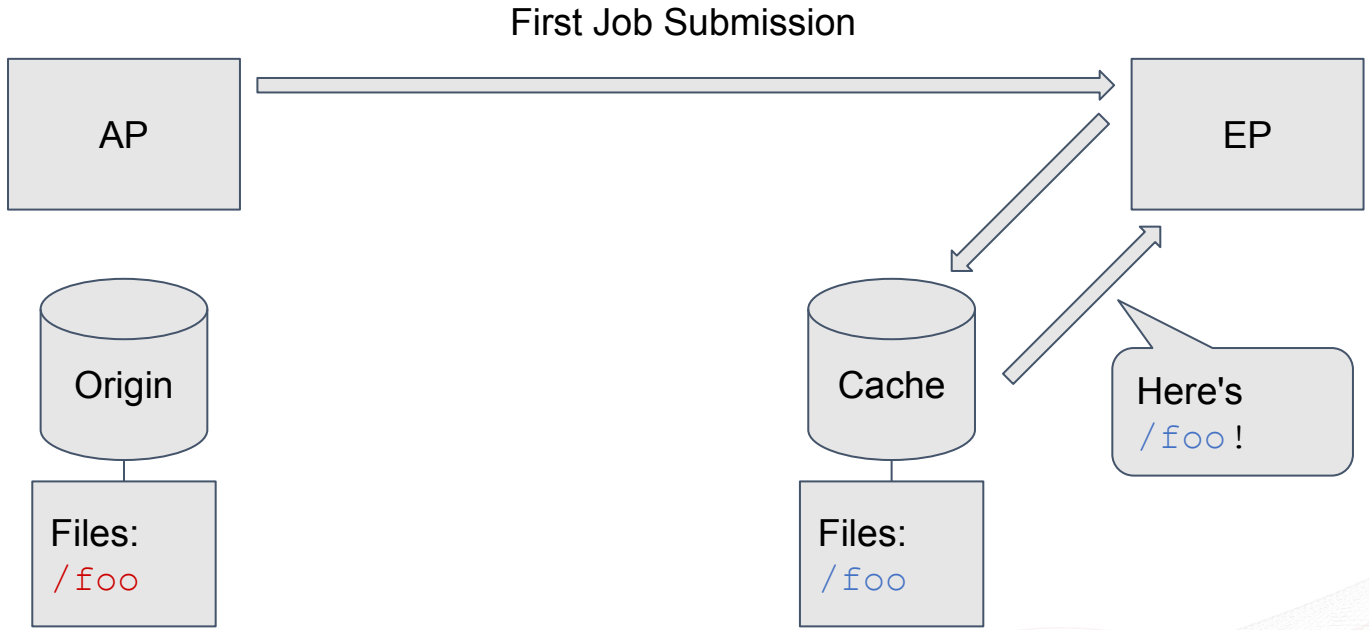
1. User submits a job, specifying `/foo` as an OSDF input
2. EP requests "`/foo`" from nearest cache



Gearing Up for What's Next…

- More & better transfers with OSDF plug-in:
  - Addressing issues of object immutability

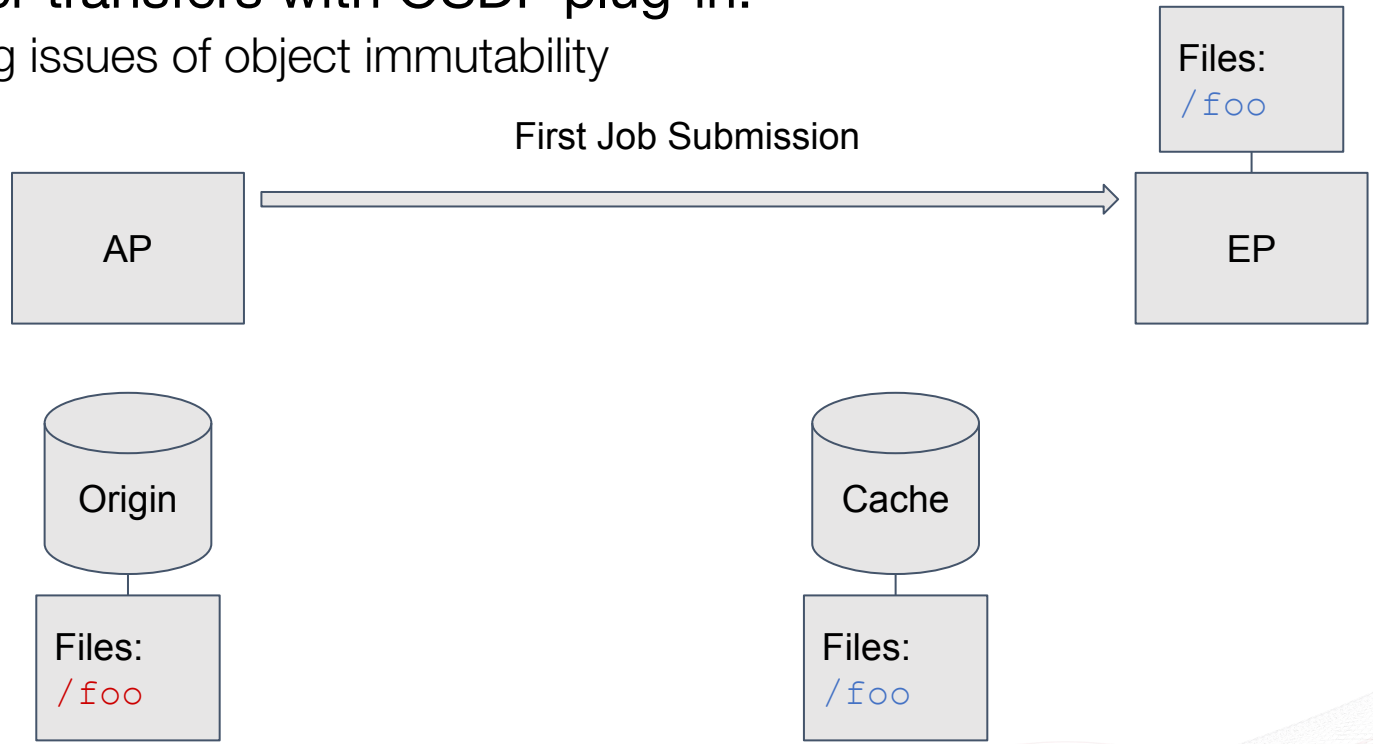
1. User submits a job, specifying `/foo` as an OSDF input
2. EP requests "`/foo`" from nearest cache
3. The cache, unaware of the difference between `/foo` and `/foo`, delivers `/foo` to the EP



Gearing Up for What's Next…

- More & better transfers with OSDF plug-in:
  - Addressing issues of object immutability

1. User submits a job, specifying `/foo` as an OSDF input
2. EP requests "`/foo`" from nearest cache
3. The cache, unaware of the difference between `/foo` and `/foo`, delivers `/foo` to the EP
4. The job continues, but with the WRONG version of "`/foo`"!



Why not just modify the cache?

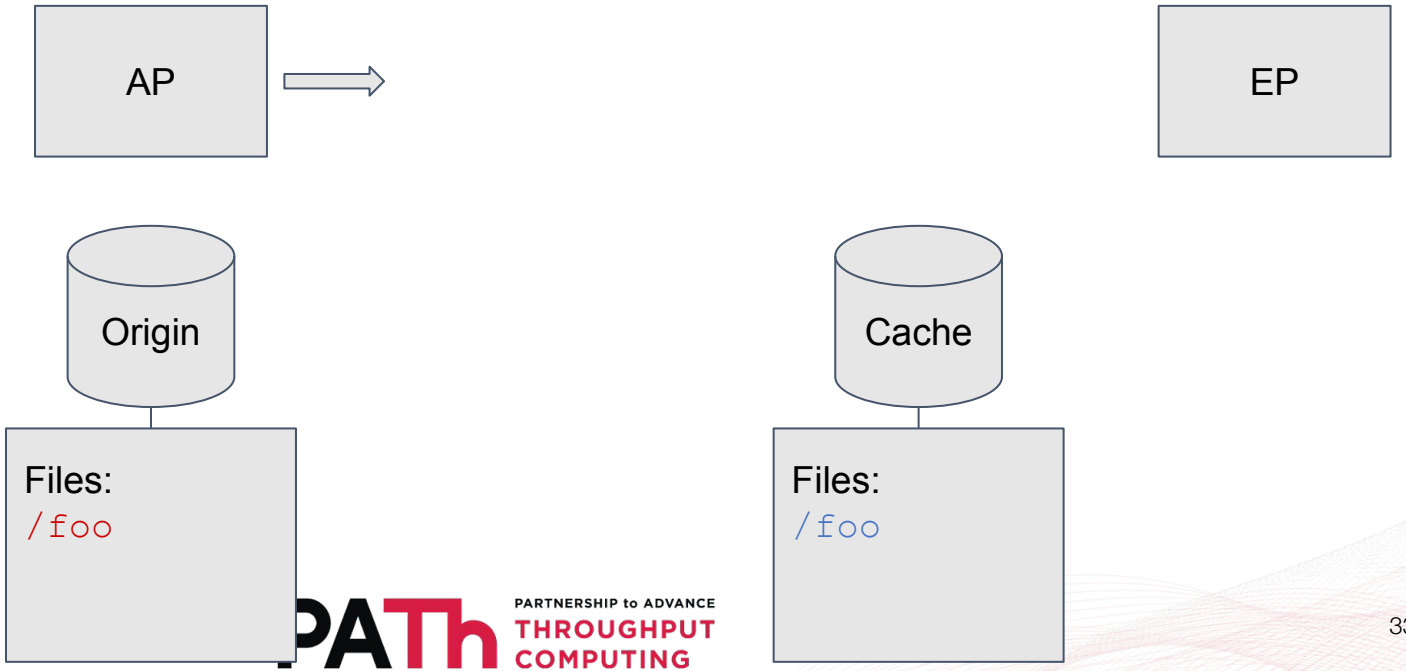
- The cache is built on XRootD, and there's no built-in mechanism to achieve this.
- OSDF was originally built to serve datasets for Big Science
  - Your datasets shouldn't be changing!
- Calculating checksums from file contents for large files is expensive – we don't want to do this

Our upcoming solution

- Introduction of Shadow Job Hooks

Second Job Submission w/ Modified `/foo`

1. User submits a job, specifying `/foo` as an OSDF input

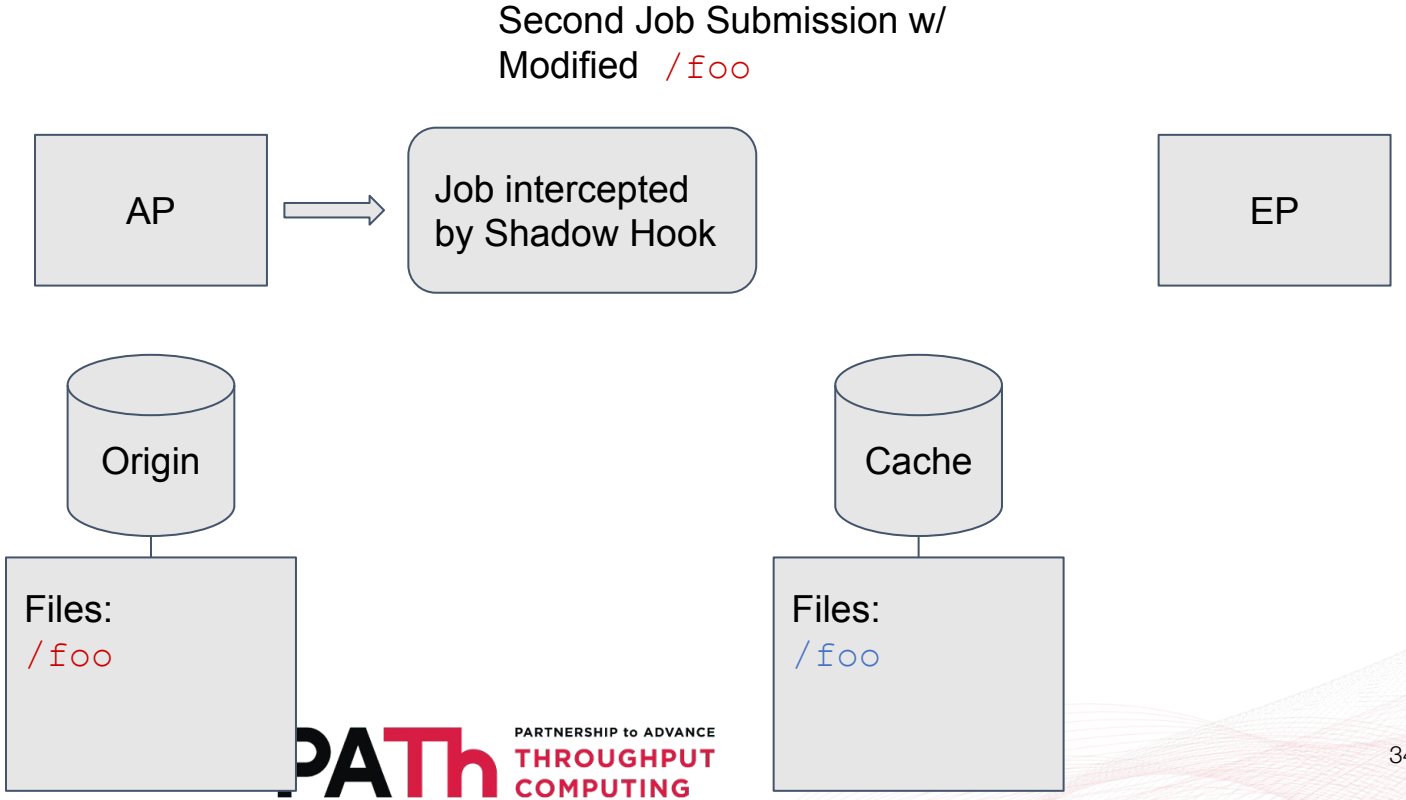


PARTNERSHIP to ADVANCE THROUGHPUT COMPUTING

Our upcoming solution

- Introduction of Shadow Job Hooks

1. User submits a job, specifying `/foo` as an OSDF input
2. Job intercepted by shadow hook, which does the following:

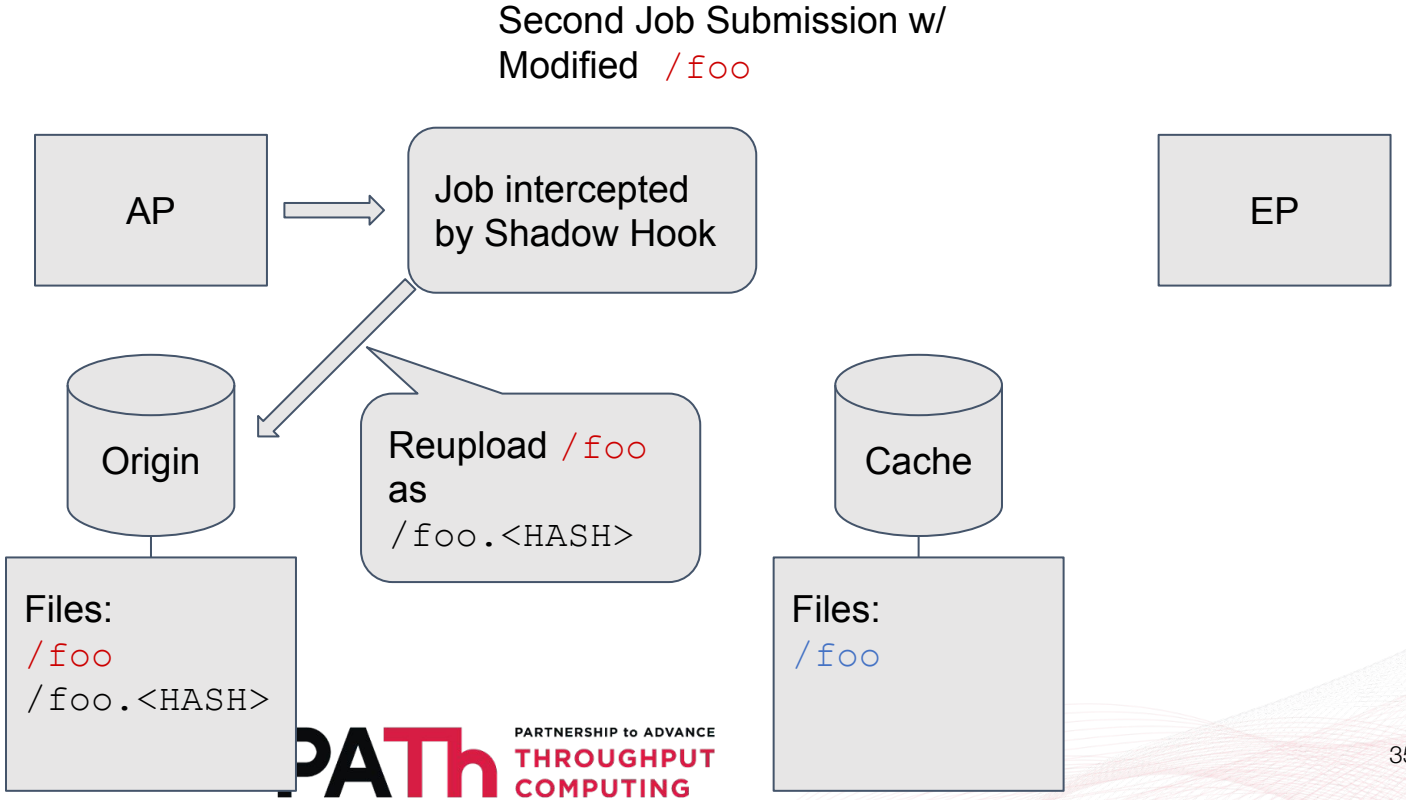


PARTNERSHIP to ADVANCE THROUGHPUT COMPUTING

Our upcoming solution

- Introduction of Shadow Job Hooks

1. User submits a job, specifying `/foo` as an OSDF input
2. Job intercepted by shadow hook, which does the following:
   a. Creates a "hash" of `/foo` using ctime, mtime, filesize, and day
   b. Reuploads `/foo` to Origin with new name of `/foo.<HASH>`

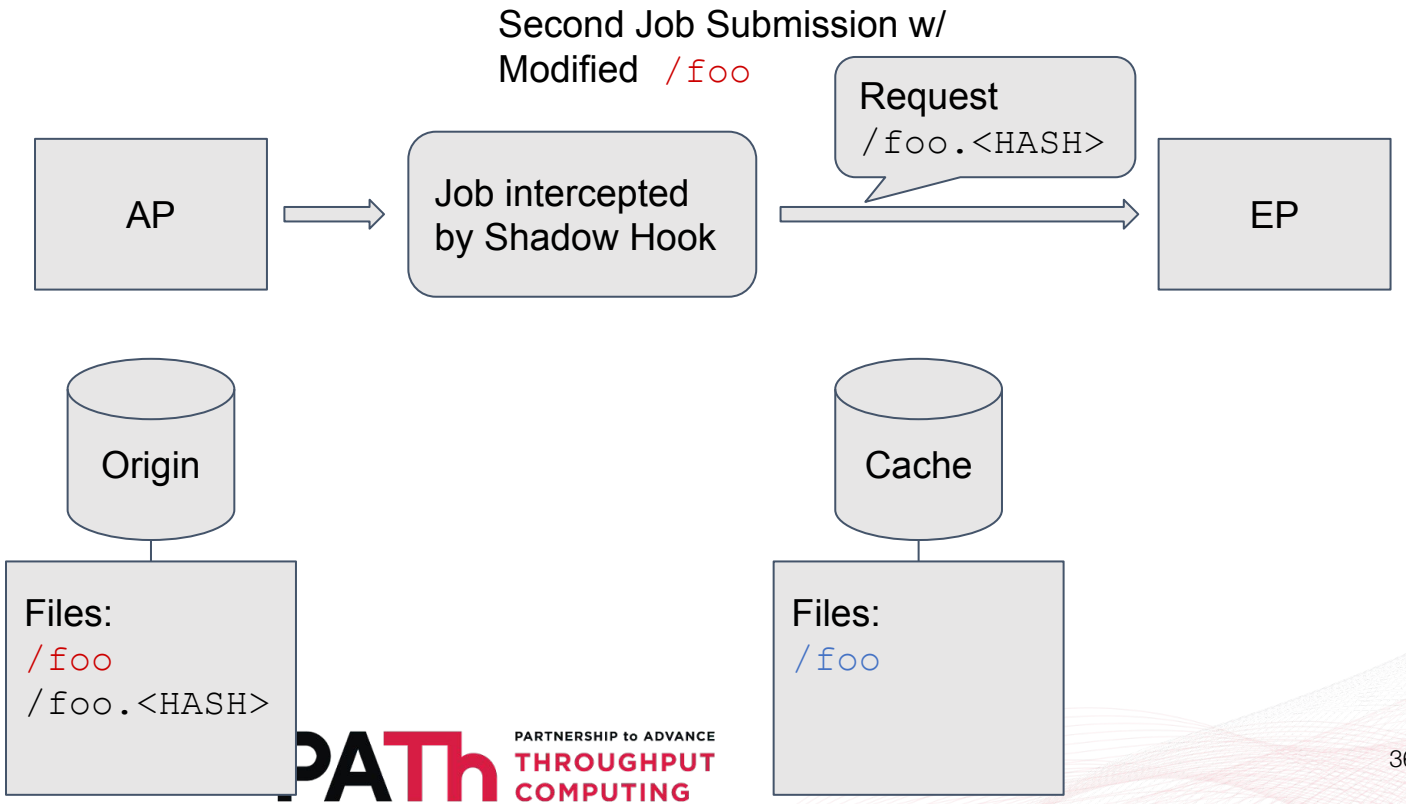


PARTNERSHIP to ADVANCE THROUGHPUT COMPUTING

Our upcoming solution

- Introduction of Shadow Job Hooks

1. User submits a job, specifying `/foo` as an OSDF input
2. Job intercepted by shadow hook, which does the following:
   a. Creates a "hash" of `/foo` using ctime, mtime, filesize, and day
   b. Reuploads `/foo` to Origin with new name of `/foo.<HASH>`
   c. Modifies job's ClassAd to request `/foo.<HASH>`

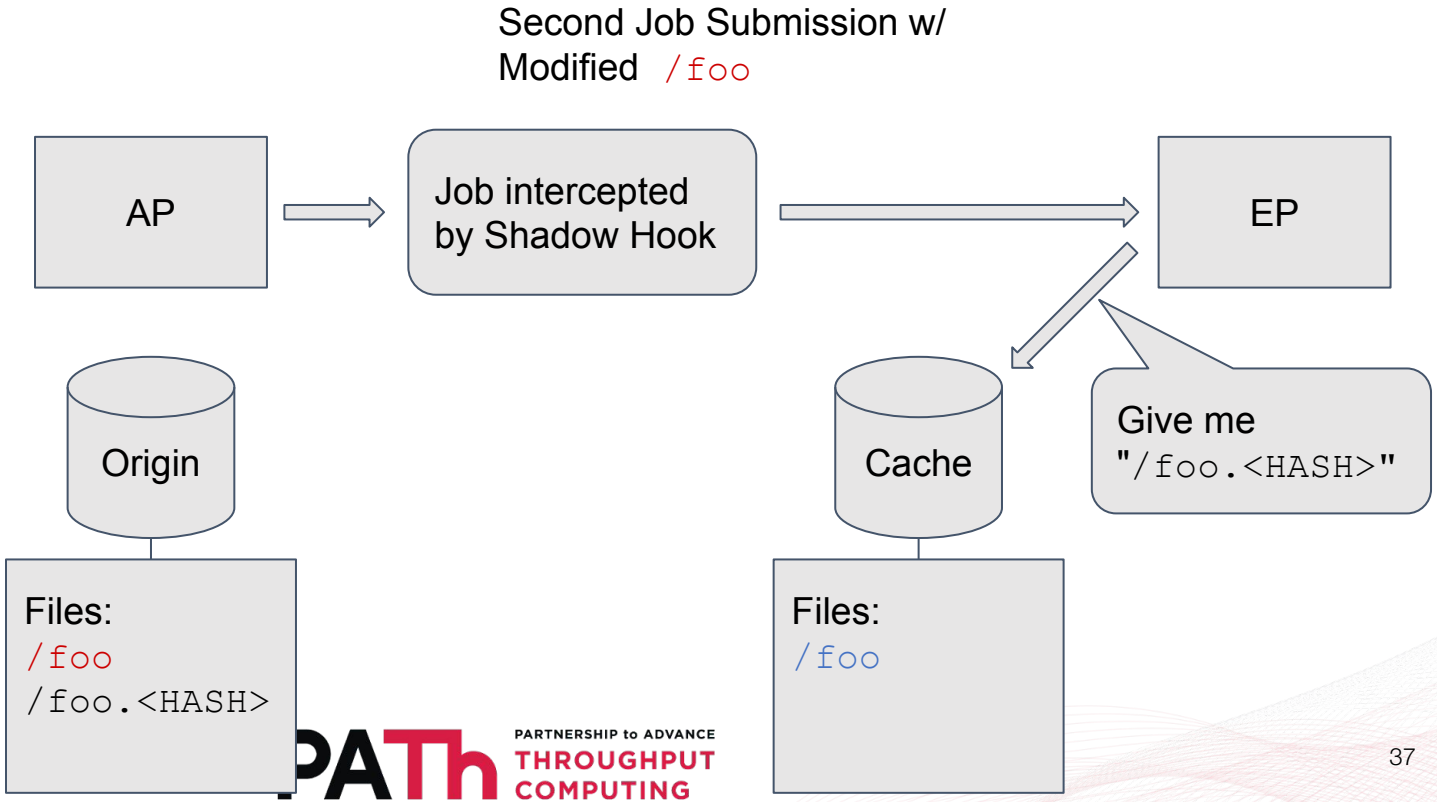


PATh PARTNERSHIP to ADVANCE THROUGHPUT COMPUTING

Our upcoming solution

- Introduction of Shadow Job Hooks

1. User submits a job, specifying `/foo` as an OSDF input
2. Job intercepted by shadow hook, which does the following:
   a. Creates a "hash" of `/foo` using ctime, mtime, filesize, and day
   b. Reuploads `/foo` to Origin with new name of `/foo.<HASH>`
   c. Modifies job's ClassAd to request `/foo.<HASH>`

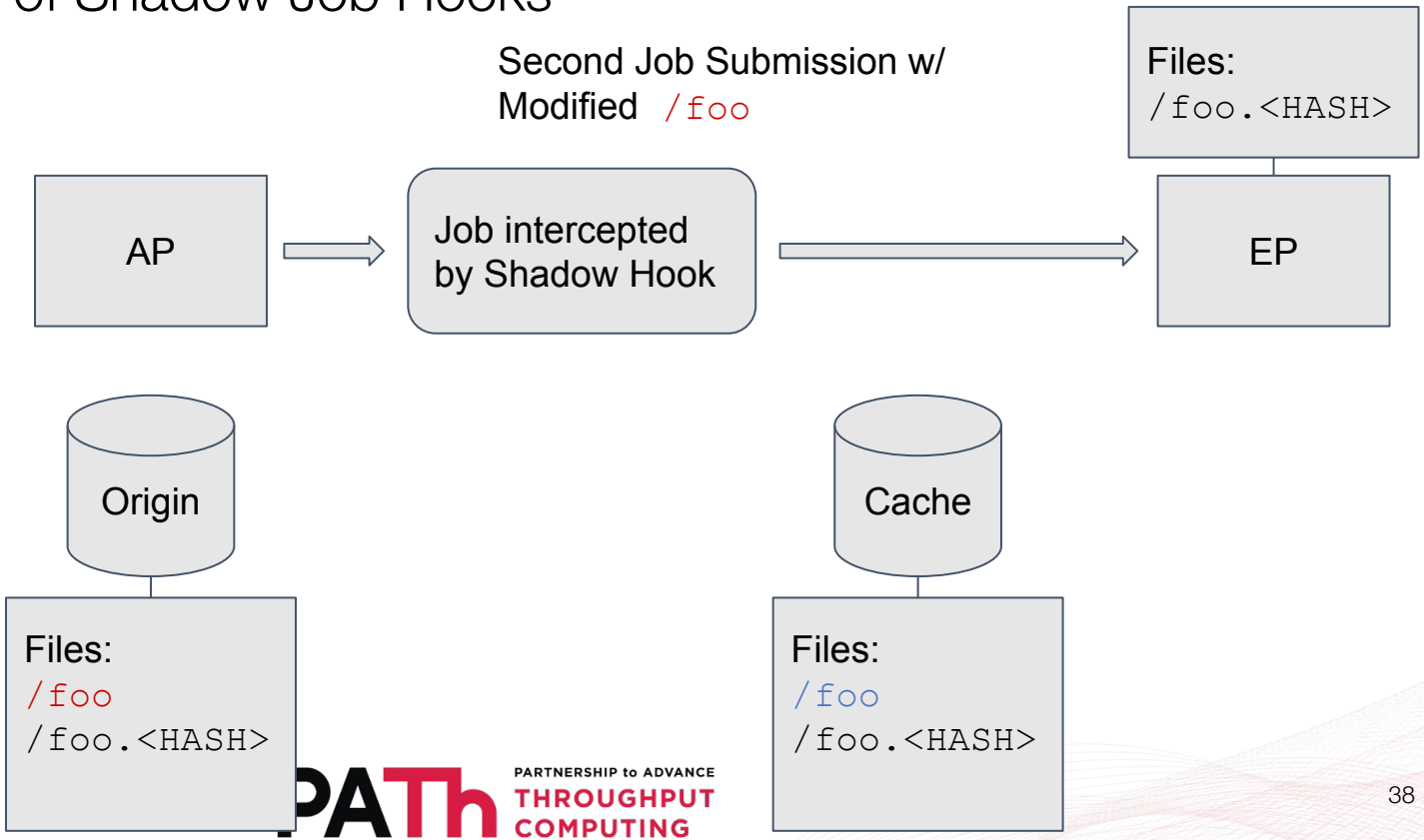


PATh PARTNERSHIP to ADVANCE THROUGHPUT COMPUTING

Our upcoming solution

- Introduction of Shadow Job Hooks

1. User submits a job, specifying `/foo` as an OSDF input
2. Job intercepted by shadow hook, which does the following:
   a. Creates a "hash" of `/foo` using ctime, mtime, filesize, and day
   b. Reuploads `/foo` to Origin with new name of `/foo.<HASH>`
   c. Modifies job's ClassAd to request `/foo.<HASH>`



PATh PARTNERSHIP to ADVANCE THROUGHPUT COMPUTING

Questions?

Have an interesting use case? Unconvinced that HTCondor can transfer your particular set of files?

We'd love to hear about it!

This material is based upon work supported by the National Science Foundation under Grant No. 2030508. 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.