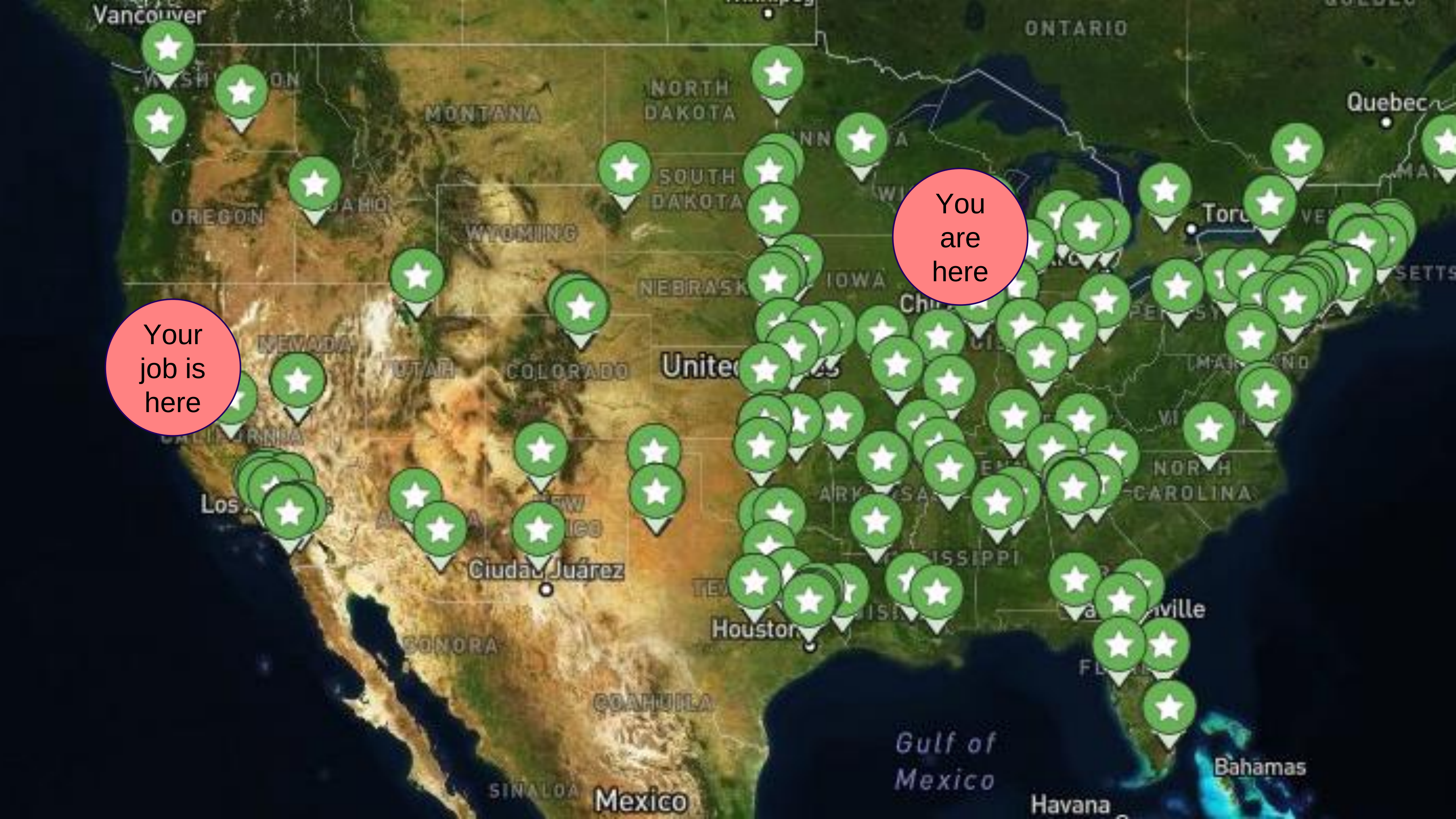
Moving Data on the OSPool

Wednesday, August 7

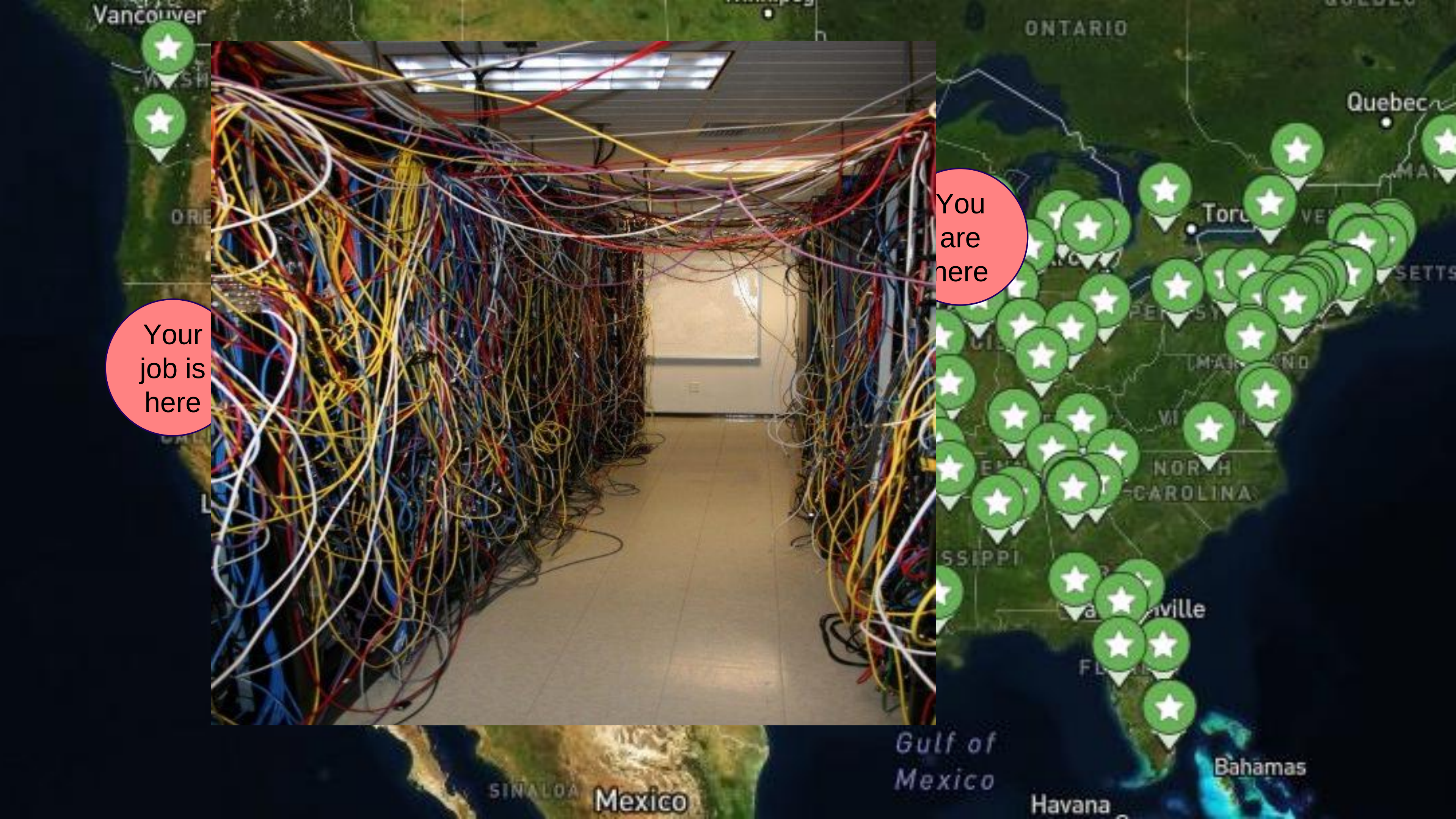
Andrew Owen

Slides adapted from Mats Rynge

This work was supported by NSF grants MPS-1148698, OAC-1836650, and OAC-2030508



You are here
Your job is here
Vancouver
ONTARIO
Quebec
NORTH DAKOTA
MONTANA
WASHINGTON
SOUTH DAKOTA
OREGON
IDAHO
WYOMING
NEBRASKA
IOWA
Chicago
Toronto
NEVADA
UTAH
COLORADO
United States
MARYLAND
CALIFORNIA
NORTH CAROLINA
Los Angeles
ARKANSAS
NEW MEXICO
MISSISSIPPI
Ciudad Juárez
Houston
SONORA
COAHUILA
Gulf of Mexico
Bahamas
SINALOA
Mexico
Havana



Your job is here

You are here

From yesterday…

```
container_image = py-cowsay.sif
```

From yesterday…

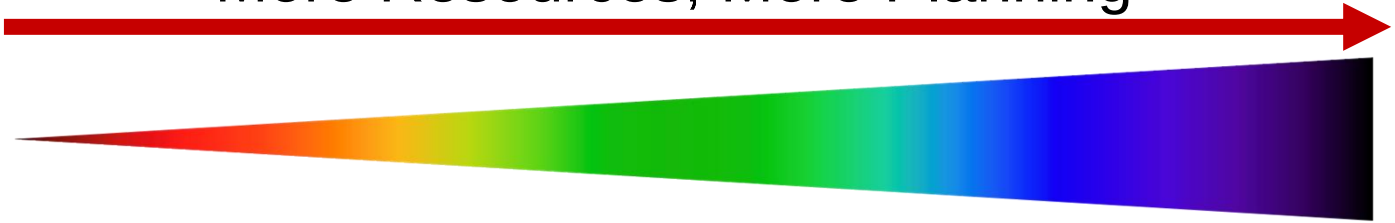
```
container_image = py-cowsay.sif

queue 10000
```

Like all things

We like to think of HTC/OSPool usage as a spectrum:

More Resources, More Planning



Laptop

Cluster

OSPool

Moving Data on the OSPool

- Overview / Things to Consider
- HTCondor File Transfer
- OSDF
- Other Considerations

What is ~~big~~ large data?

- In reality, “big data” is relative
  - What is ‘big’ for *you*? Why?

What is ~~big~~ large data?

- In reality, “big data” is relative
  - What is ‘big’ for *you*? Why?

- Volume, velocity, variety!
  - think: a million 1-KB files, versus one 1-TB file

Determining In-Job Needs

- "**Input**" includes *any* files needed for the job to run
  - `executable`
  - `transfer_input_files`
  - data ***and*** <u>software</u>

- "**Output**" includes any files produced for the job that *need to come back*
  - `output, error`

Data Management Tips

1. Determine your per-job needs
   a. minimize per-job data needs

2. Determine your batch needs

3. Leverage HTCondor and OSPool data handling features!

First! Try to minimize your data

- split large input for better throughput
- eliminate unnecessary data
- file compression and consolidation
  - job input: prior to job submission
  - job output: prior to end of job
  - moving data between your laptop and the submit server

‘Large’ data: The collaborator analogy

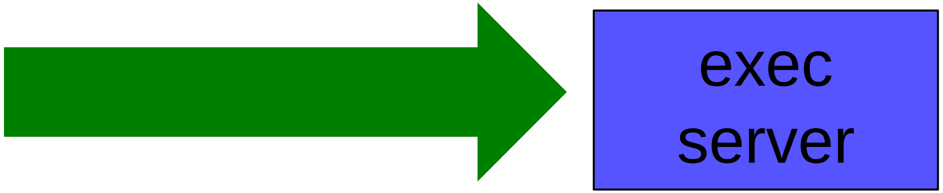
What method would you use to send data to a collaborator?

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

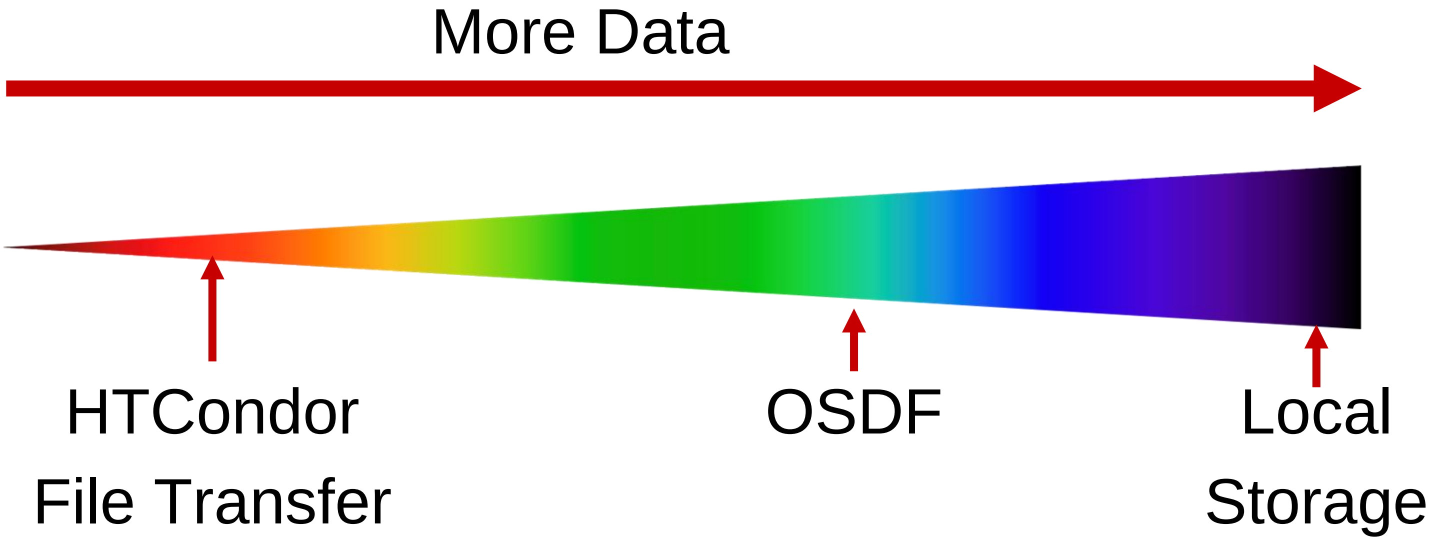
Large *input* in HTC and OSPool



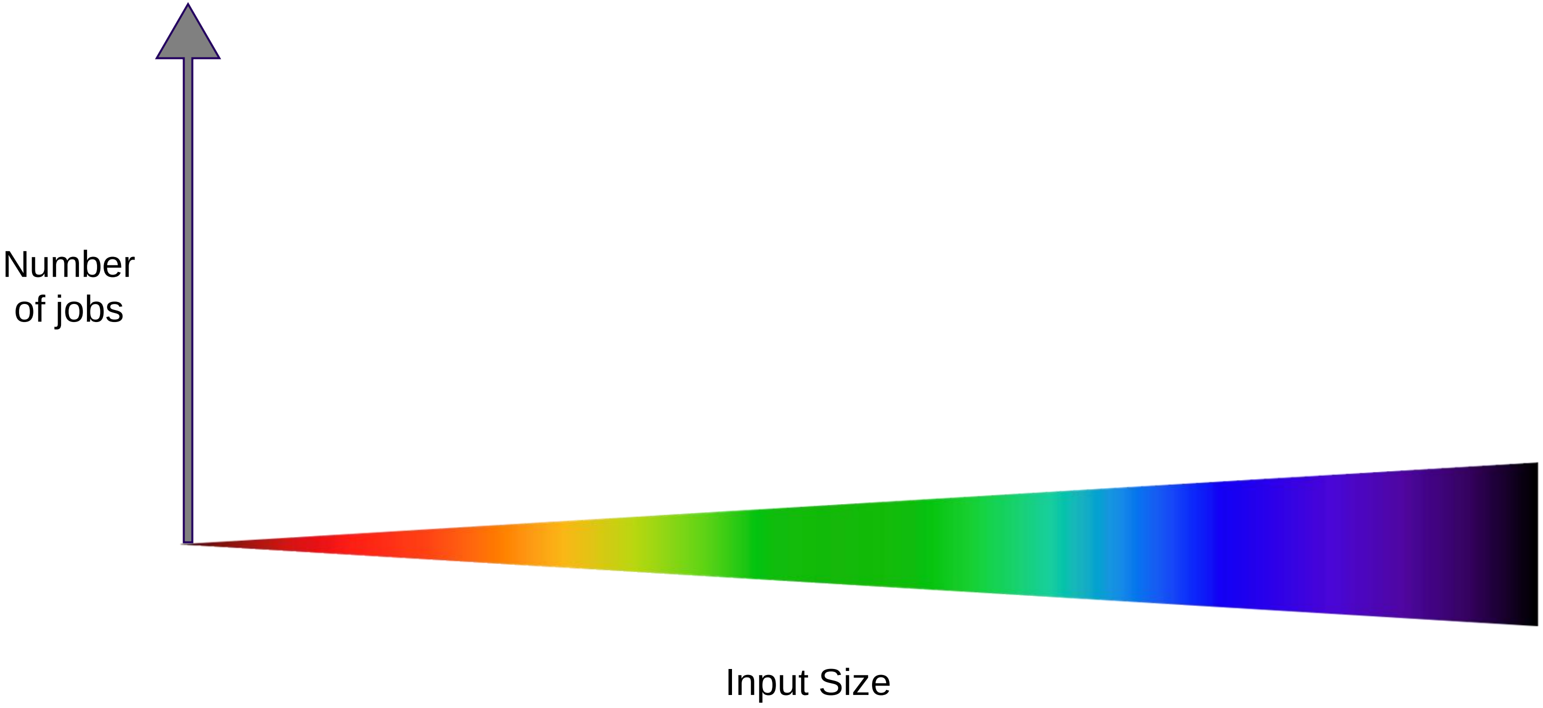
exec server

file size	method of delivery
words	within executable or arguments?
tiny – 1GB per file	HTCondor file transfer (up to 1GB total per job)
1GB – 20GB	OSDF (regional replication)
20 GB – TBs	shared file system (local copy, local execute servers)

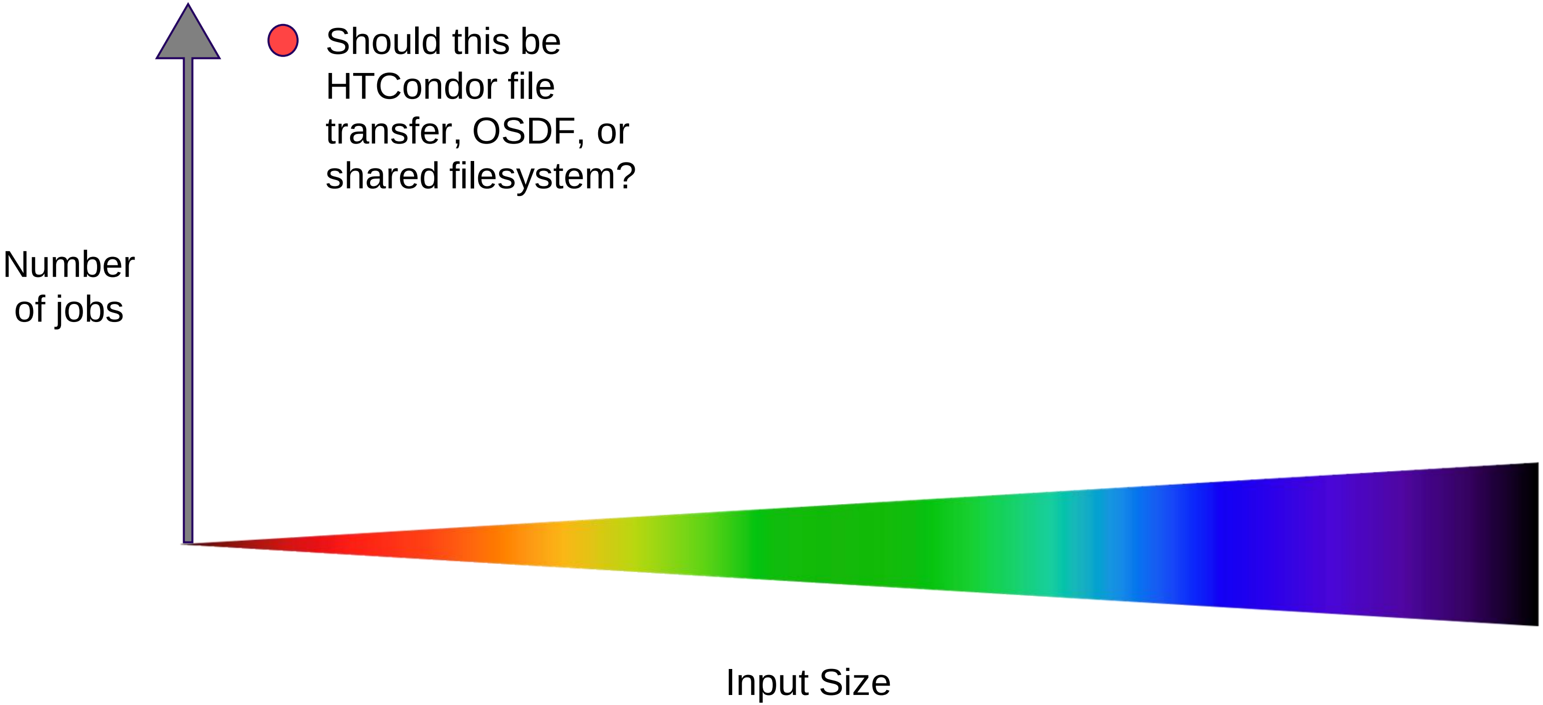
Transfers



Rule of thumb - many dimensions



Rule of thumb - many dimensions

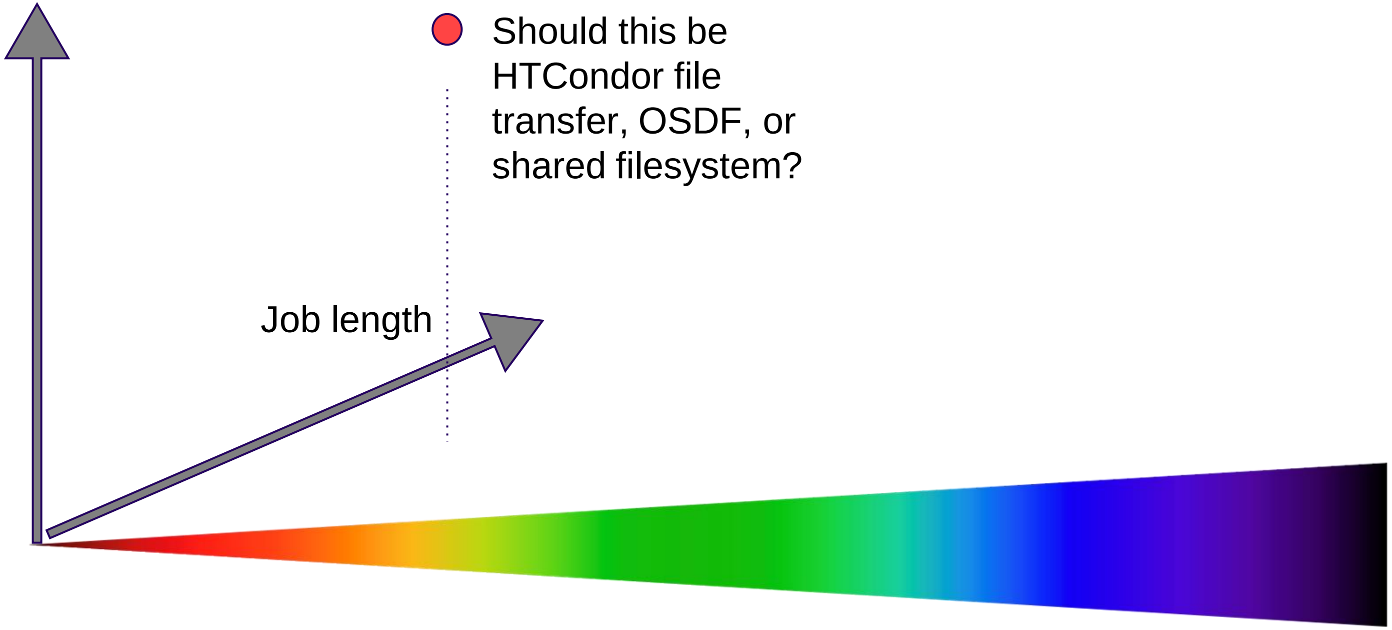


● Should this be HTCondor file transfer, OSDF, or shared filesystem?

Number of jobs

Input Size

Rule of thumb - many dimensions



Should this be HTCondor file transfer, OSDF, or shared filesystem?

Number of jobs

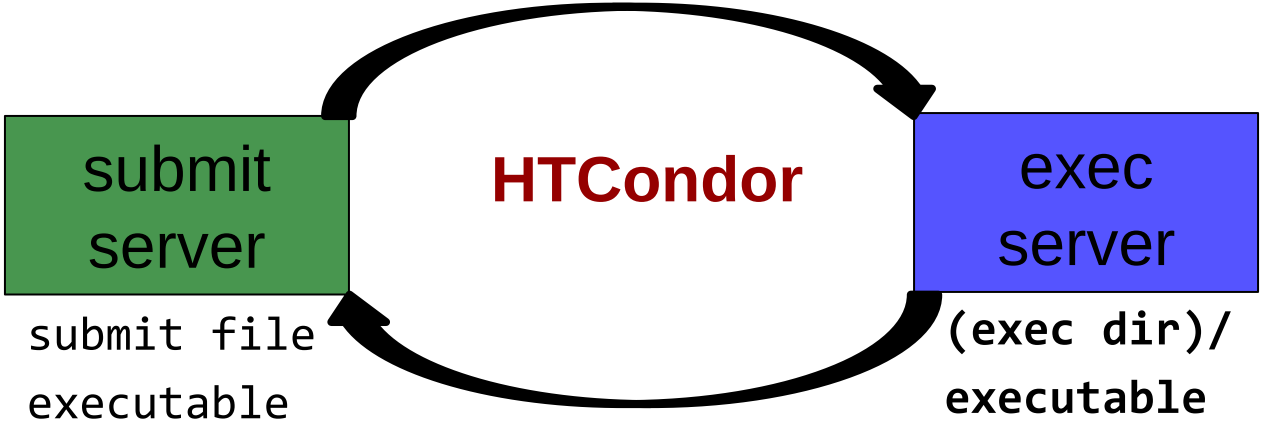
Job length

Input Size

Moving Data on the OSPool

- ~~Overview / Things to Consider~~
- **HTCondor File Transfer**
- OSDF
- Other Considerations

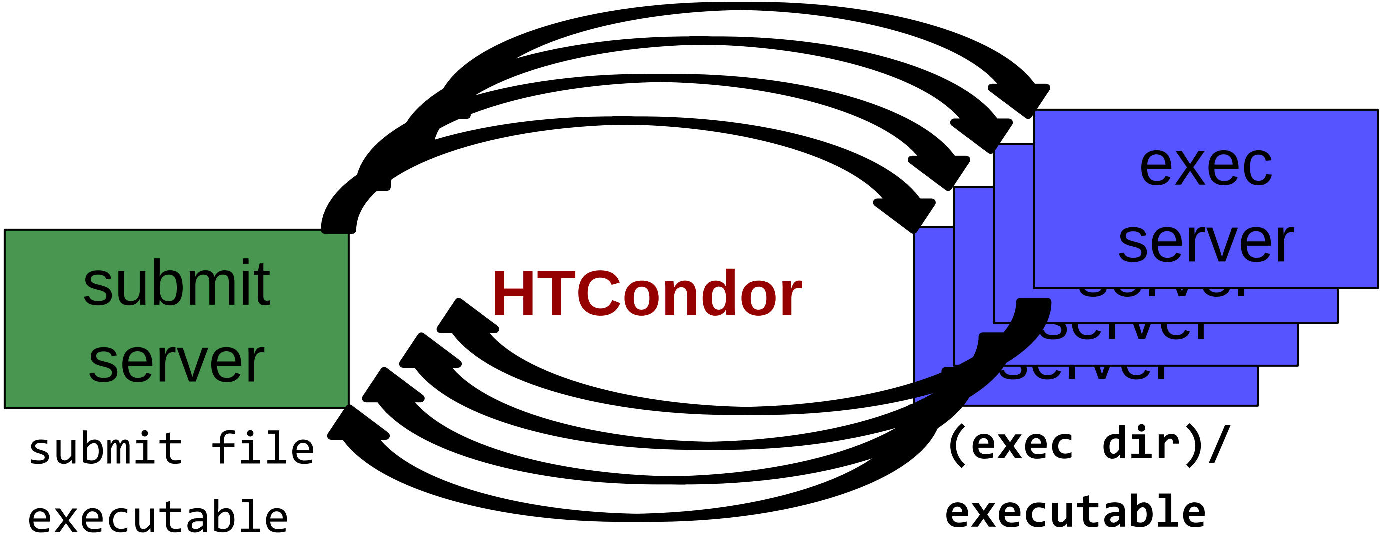
Review: HTCondor Data Handling



submit file
executable
dir/ input
output

(exec dir)/
executable
input
output

Network bottleneck: the submit server



submit file

executable

dir/ input

output

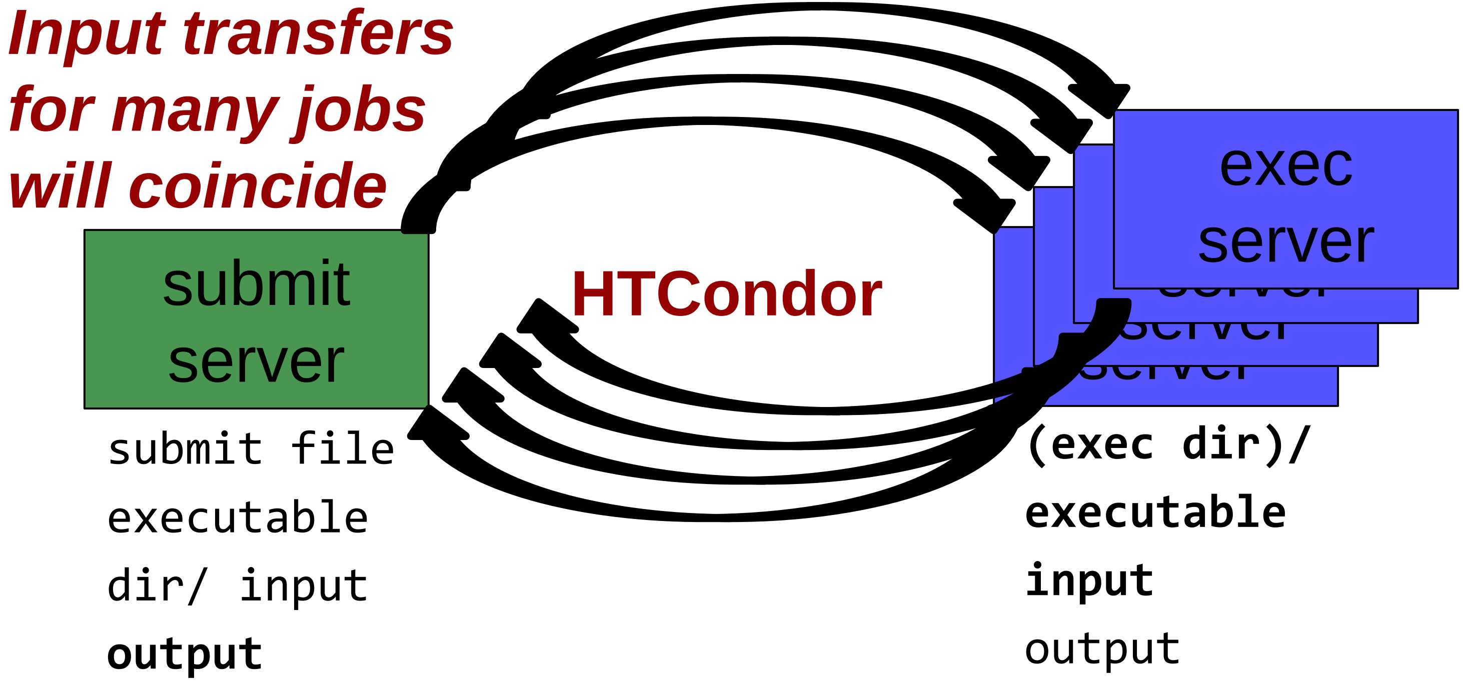
(exec dir)/

executable

input

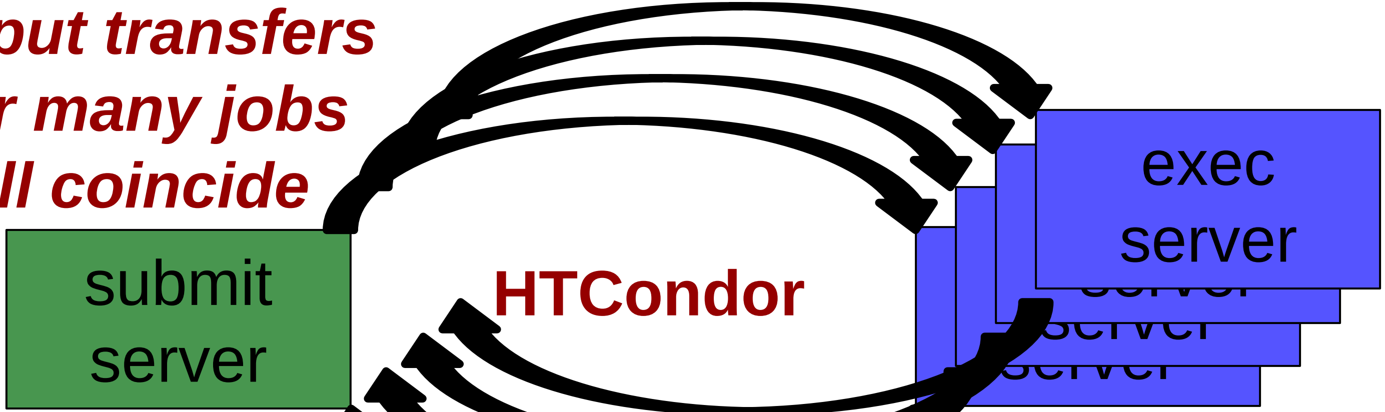
output

Network bottleneck: the submit server



Network bottleneck: the submit server

Input transfers for many jobs will coincide

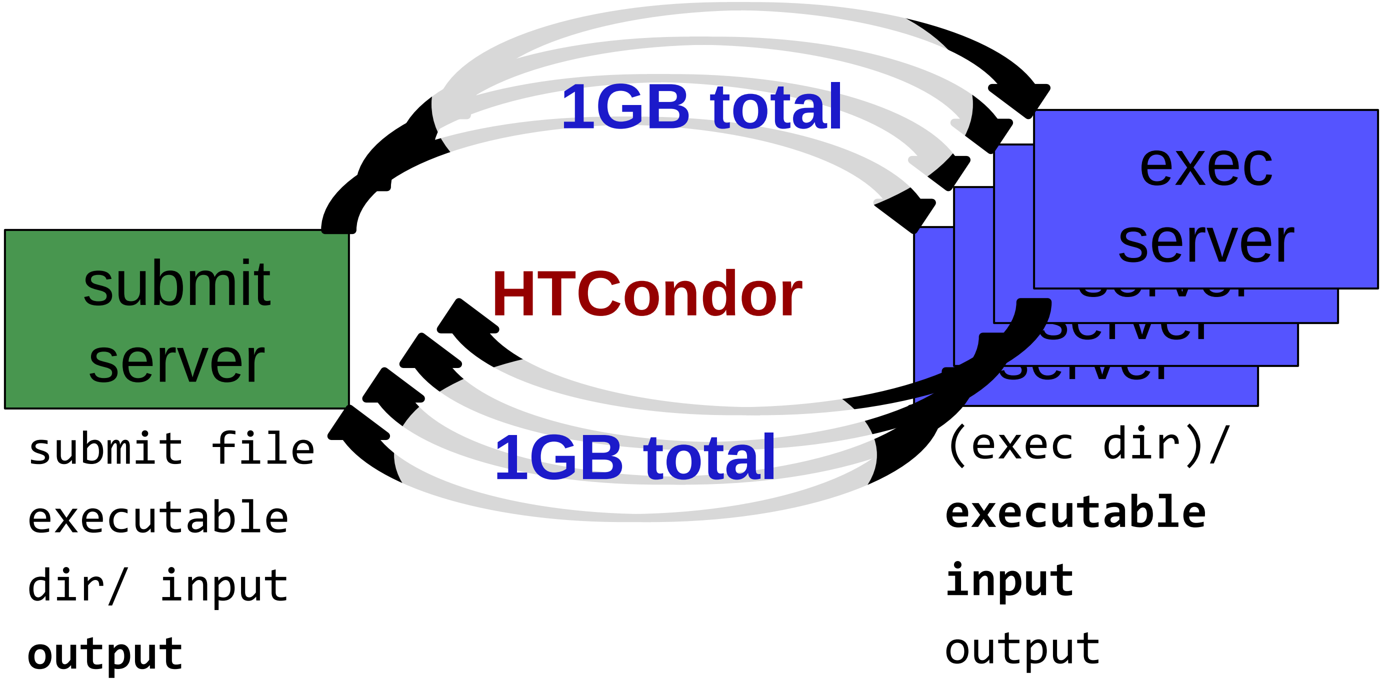


Output transfers are staggered

```
submit file
executable
dir/ input
output
```

```
(exec dir)/
executable
input
output
```

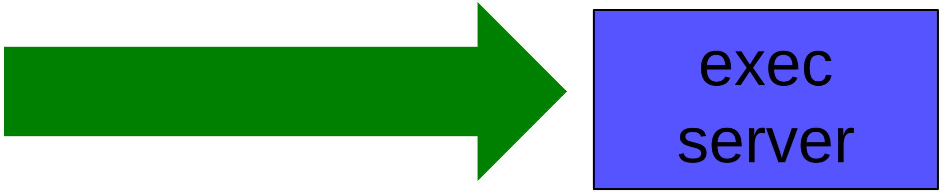
Hardware transfer limits



Moving Data on the OSPool

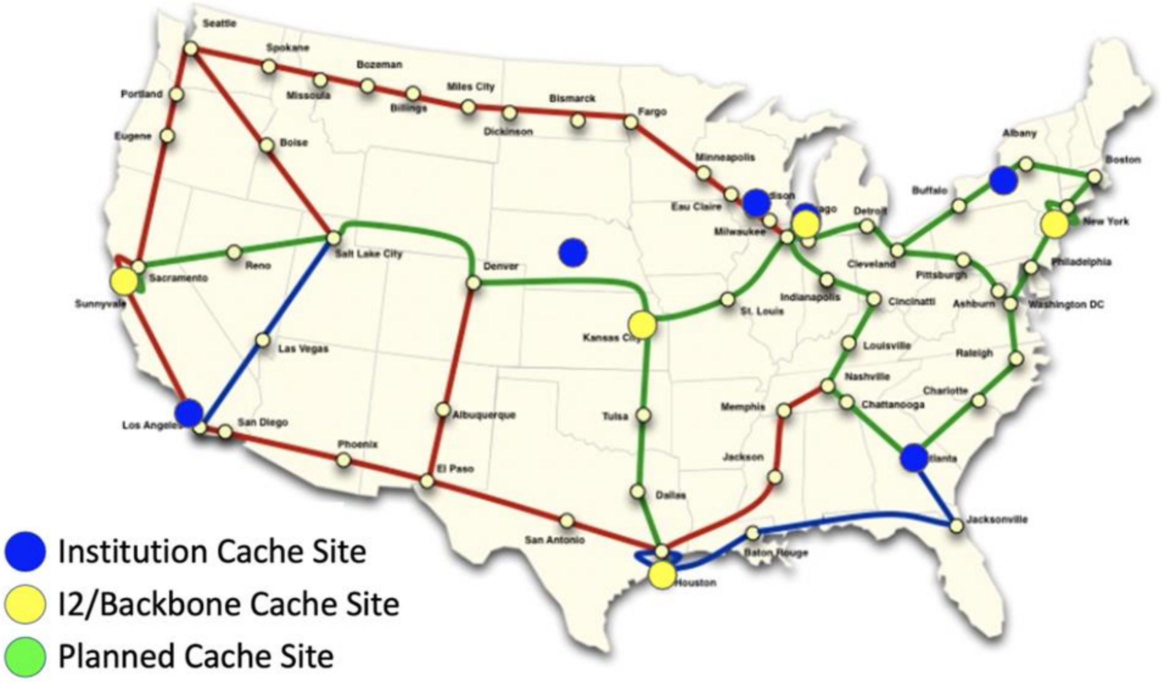
- ~~Overview / Things to Consider~~
- ~~HTCondor File Transfer~~
- **OSDF**
- Other Considerations

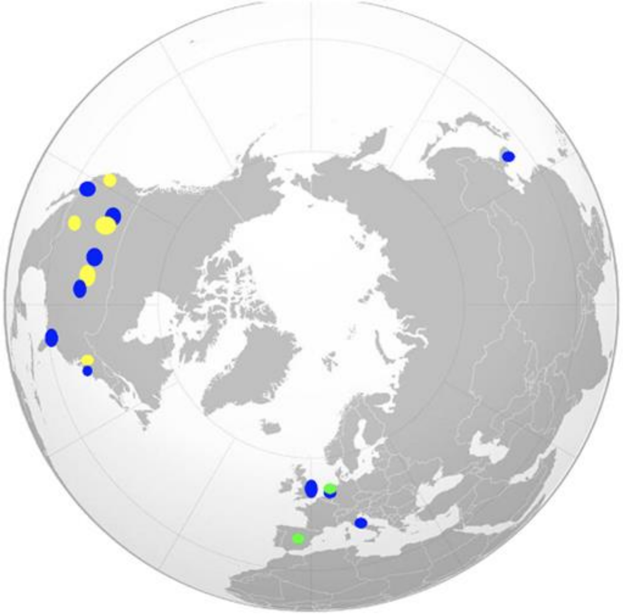
Large input in HTC and OSPool



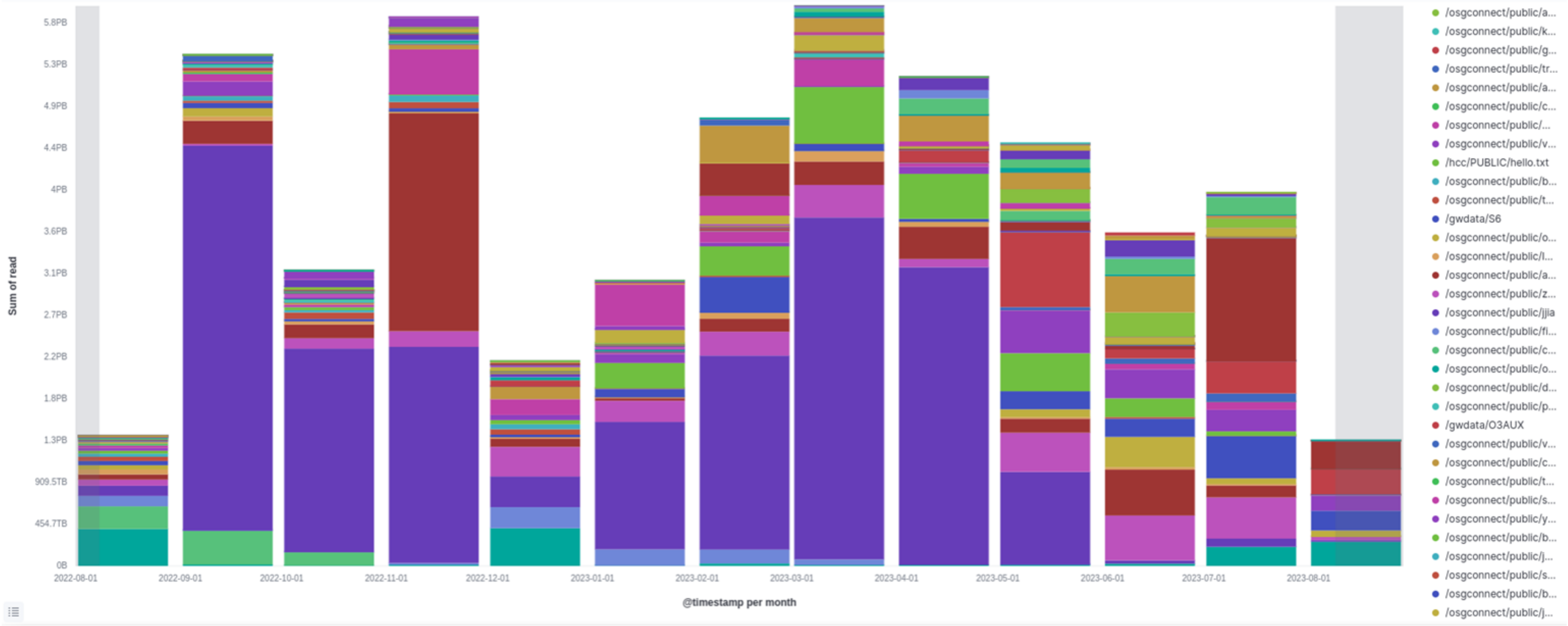
file size	method of delivery
words	within executable or arguments?
tiny – 100MB per file	HTCondor file transfer (up to 1GB total per-job)
100MB – 1GB, shared	download from web server (local caching)
1GB – 20GB, unique or shared	OSDF (regional replication)
10 GB - TBs	shared file system (local copy, local execute servers)

Open Science Data Federation (OSDF)





OSDF Usage on OSG



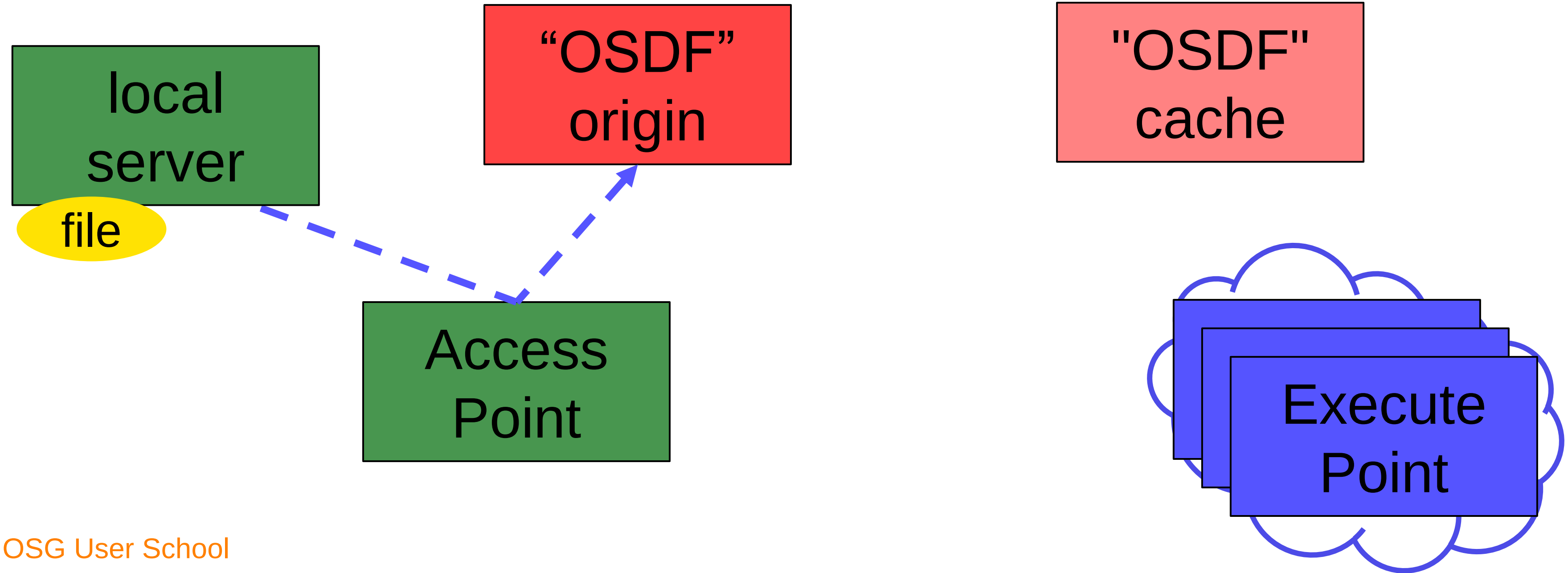
OSDF Considerations

- Available at ~95% of OSG sites
- Regional caches on *very fast* networks
  - **Recommended max file size: 20 GB**
- Can copy multiple files totaling >10GB
- Change name when update files

Placing Files in OSDF

- Place files in `/ospool/apXX/data/username/`

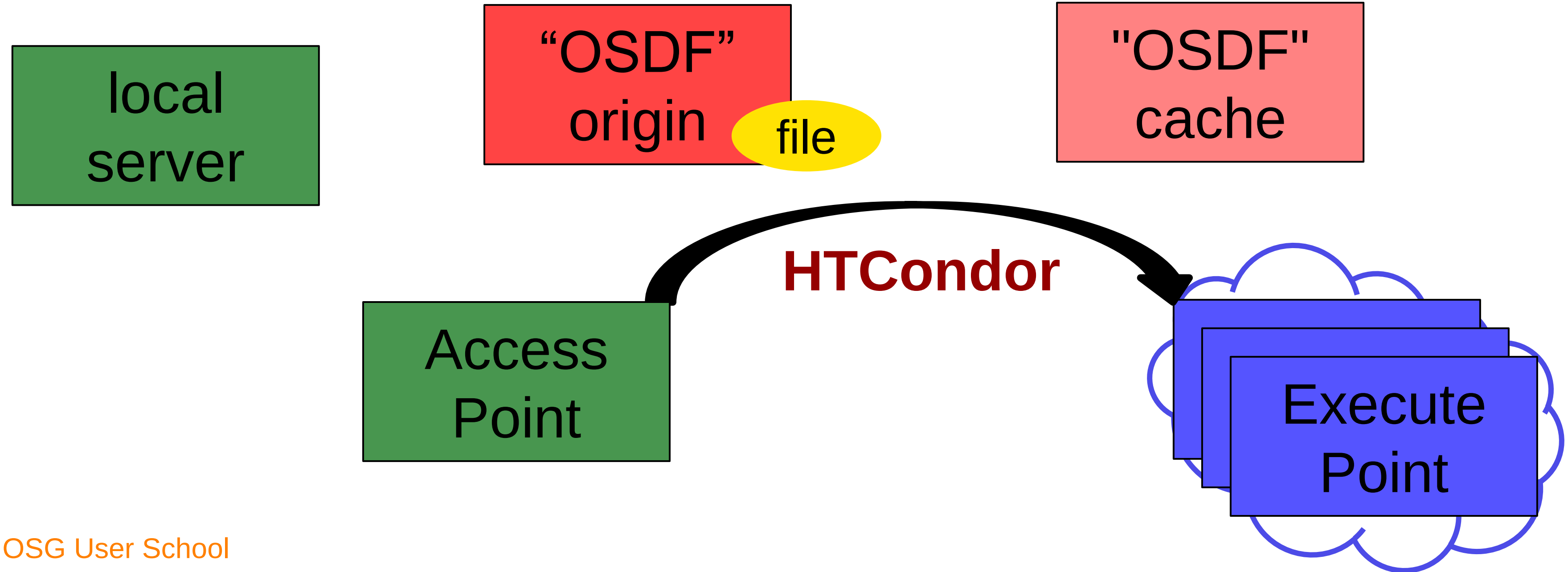
`/ospool/apXX/data/username/`



Obtaining Files in OSDF

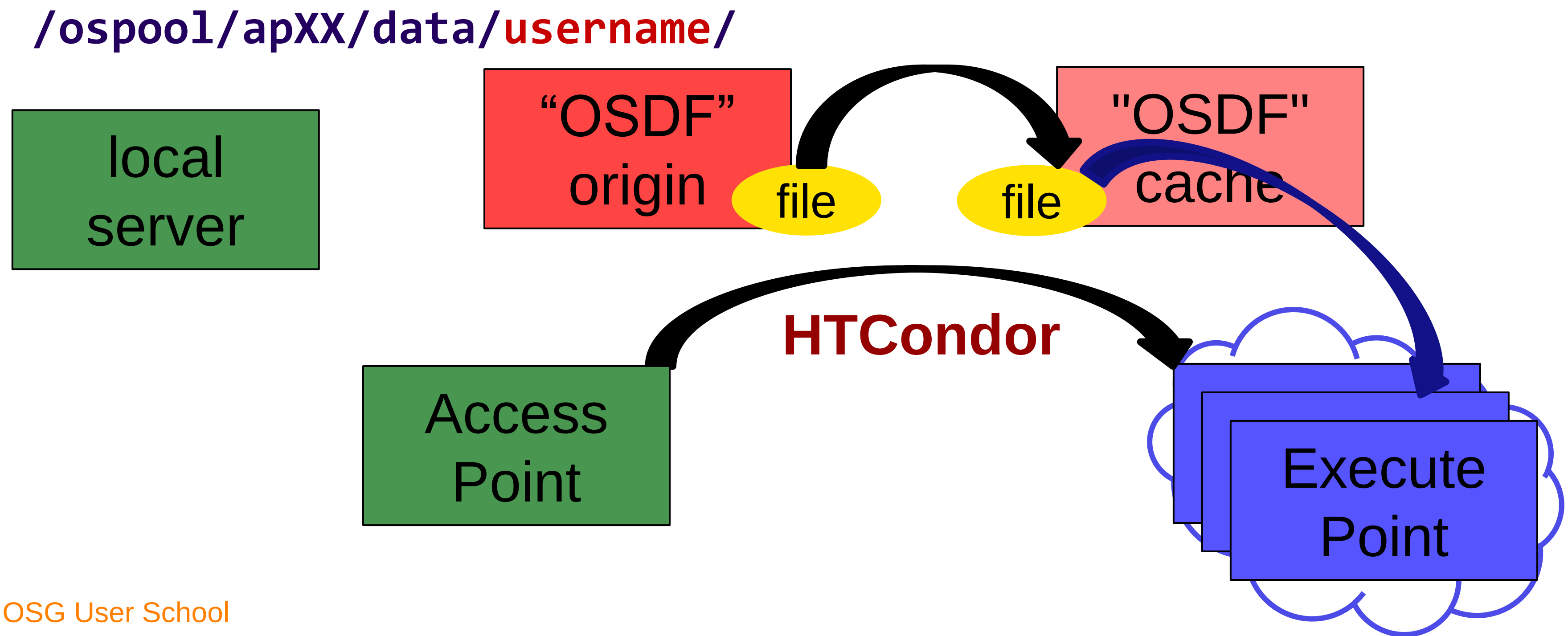
- Use HTCondor transfer for other files

`/ospool/apXX/data/username/`

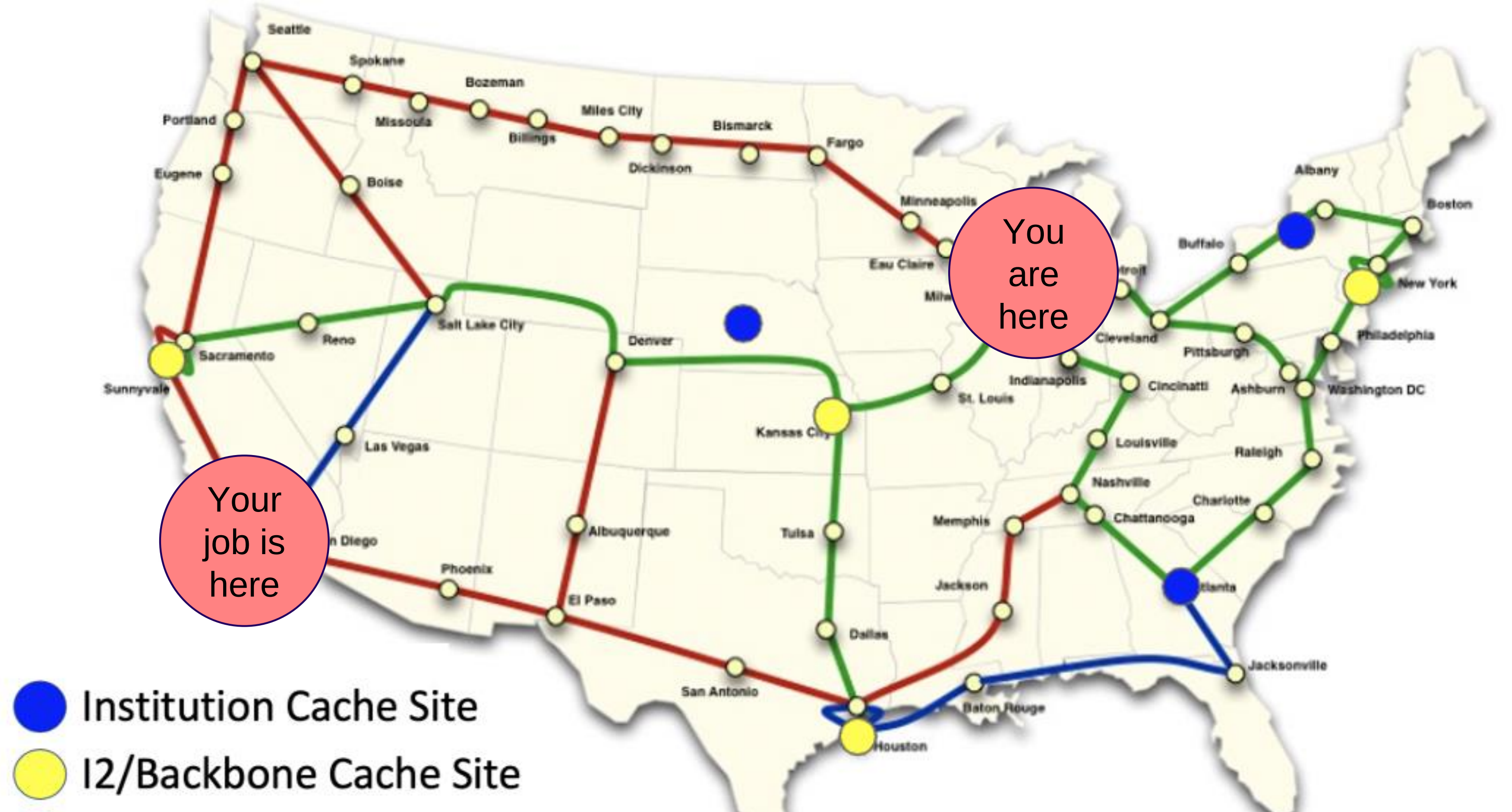


Obtaining Files in OSDF

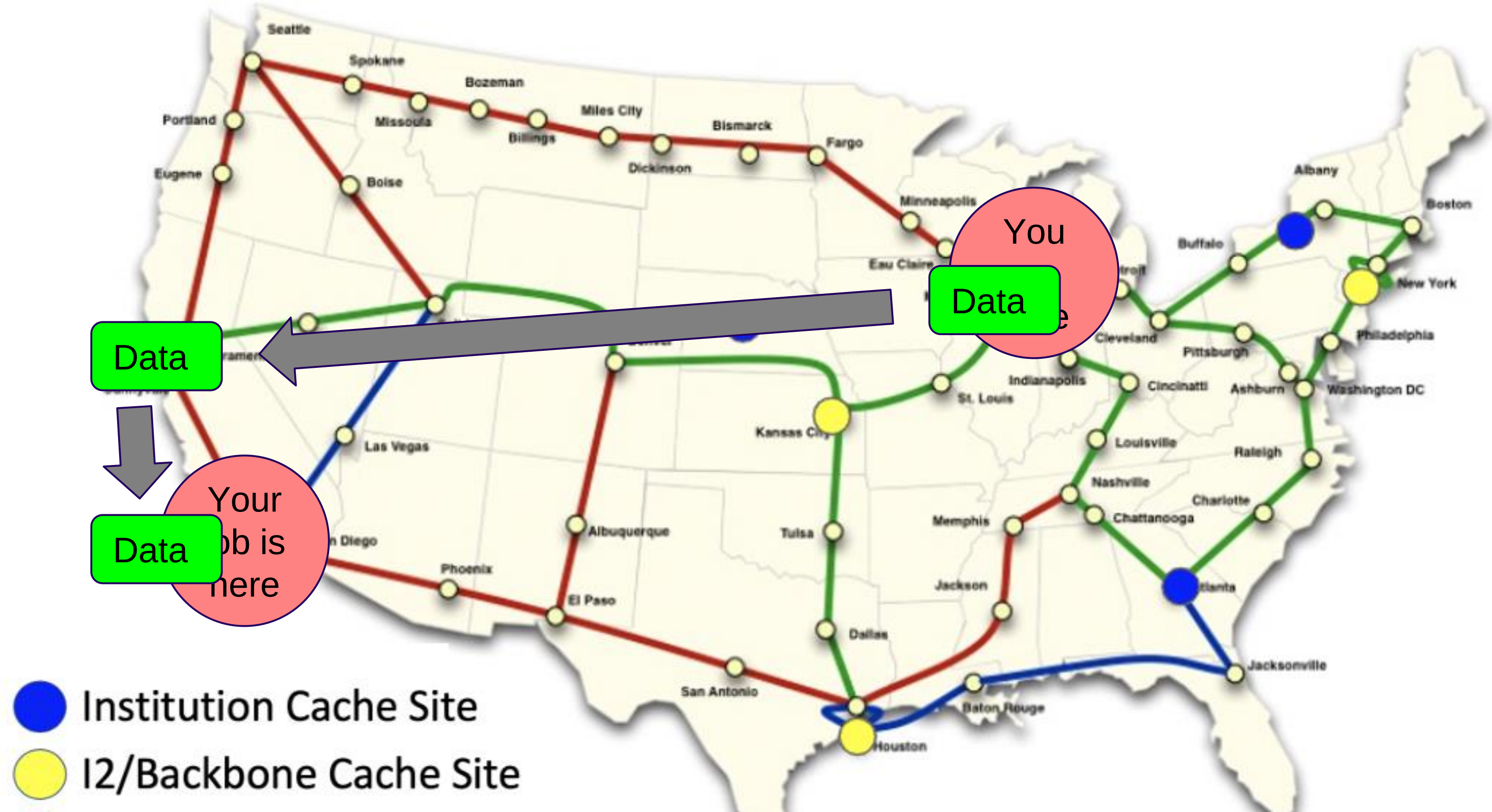
- Execute point downloads the large file through the cache



Open Science Data Federation (OSDF)



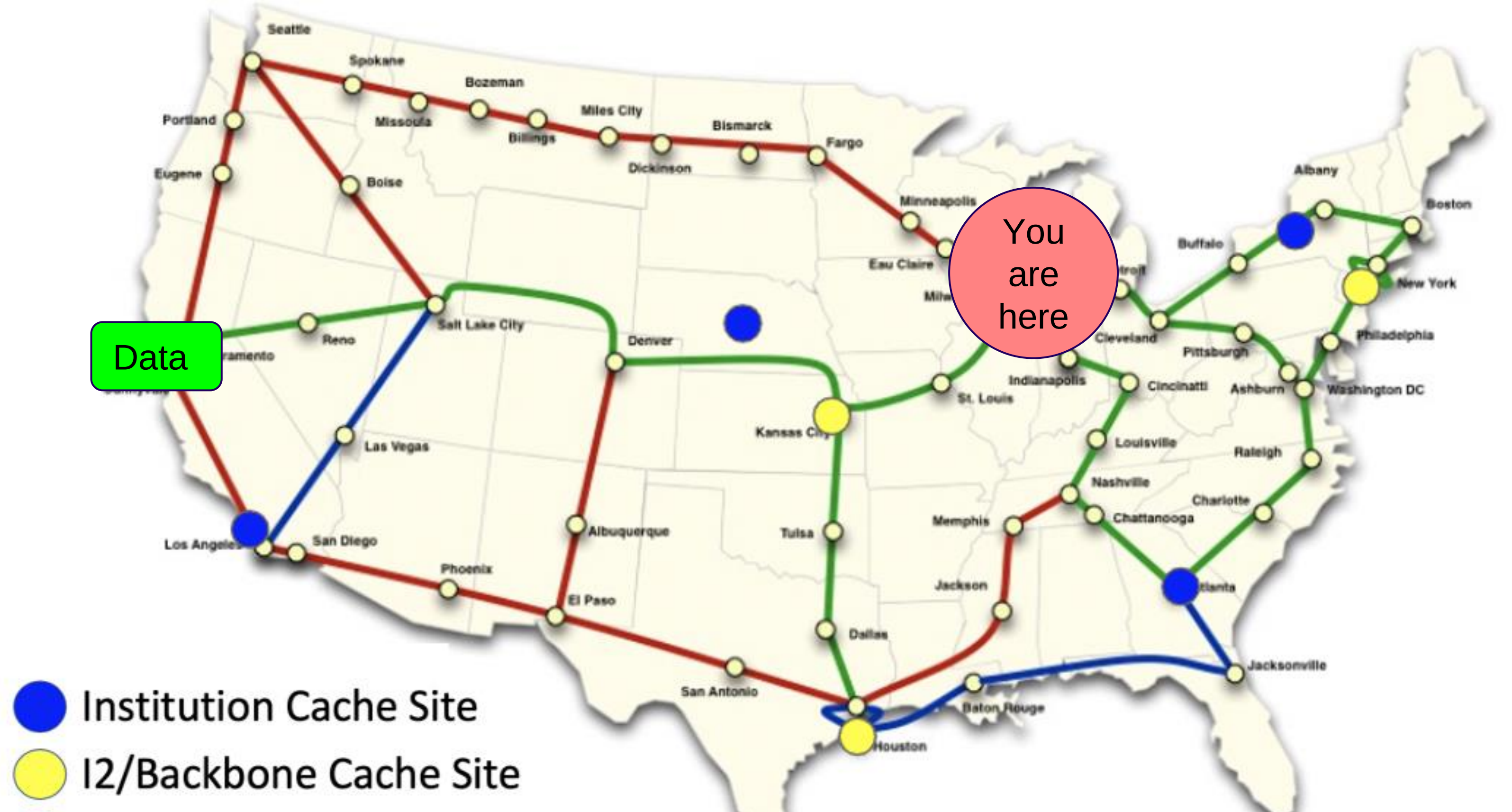
Open Science Data Federation (OSDF)



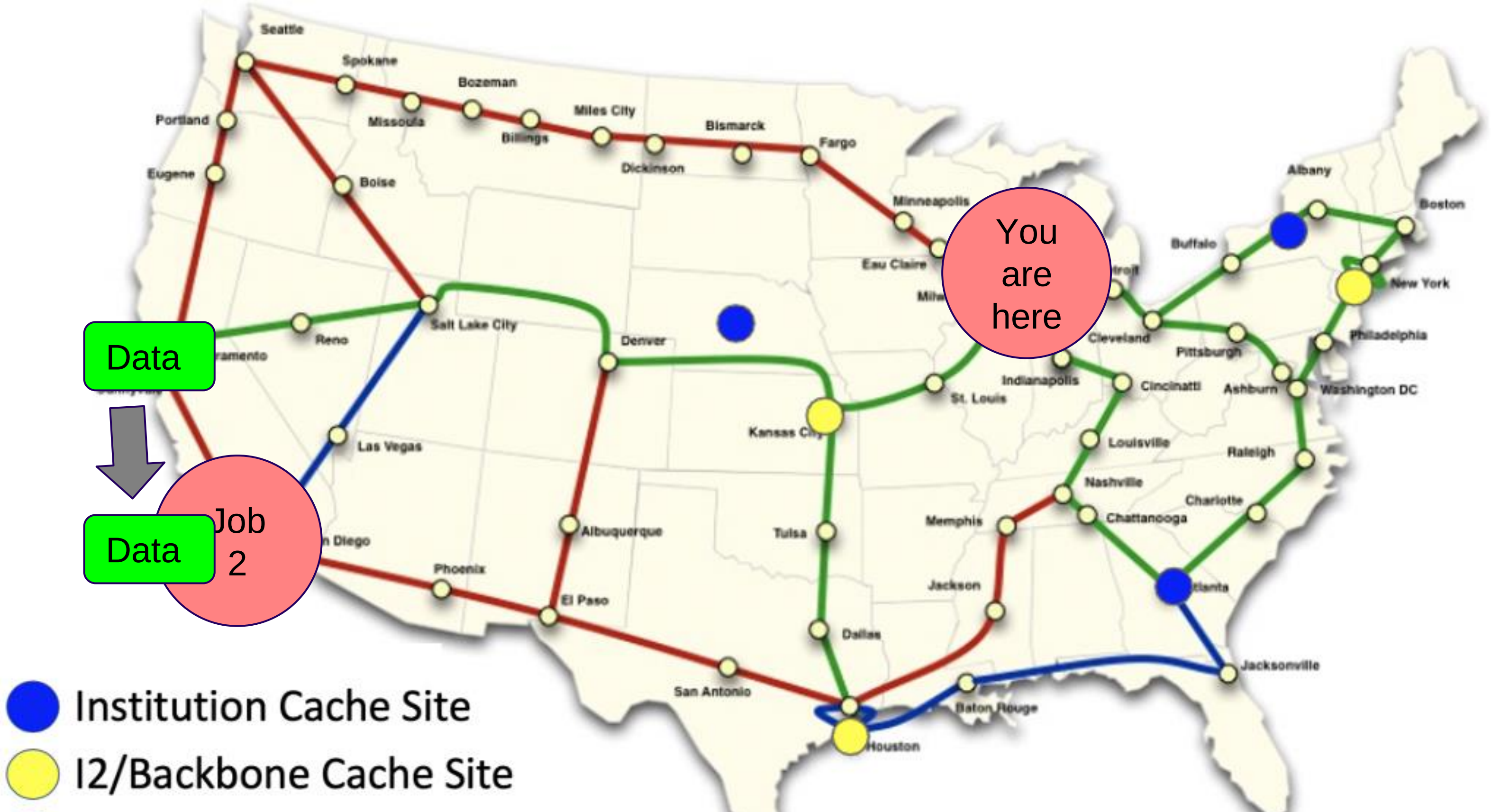
● Institution Cache Site

● I2/Backbone Cache Site

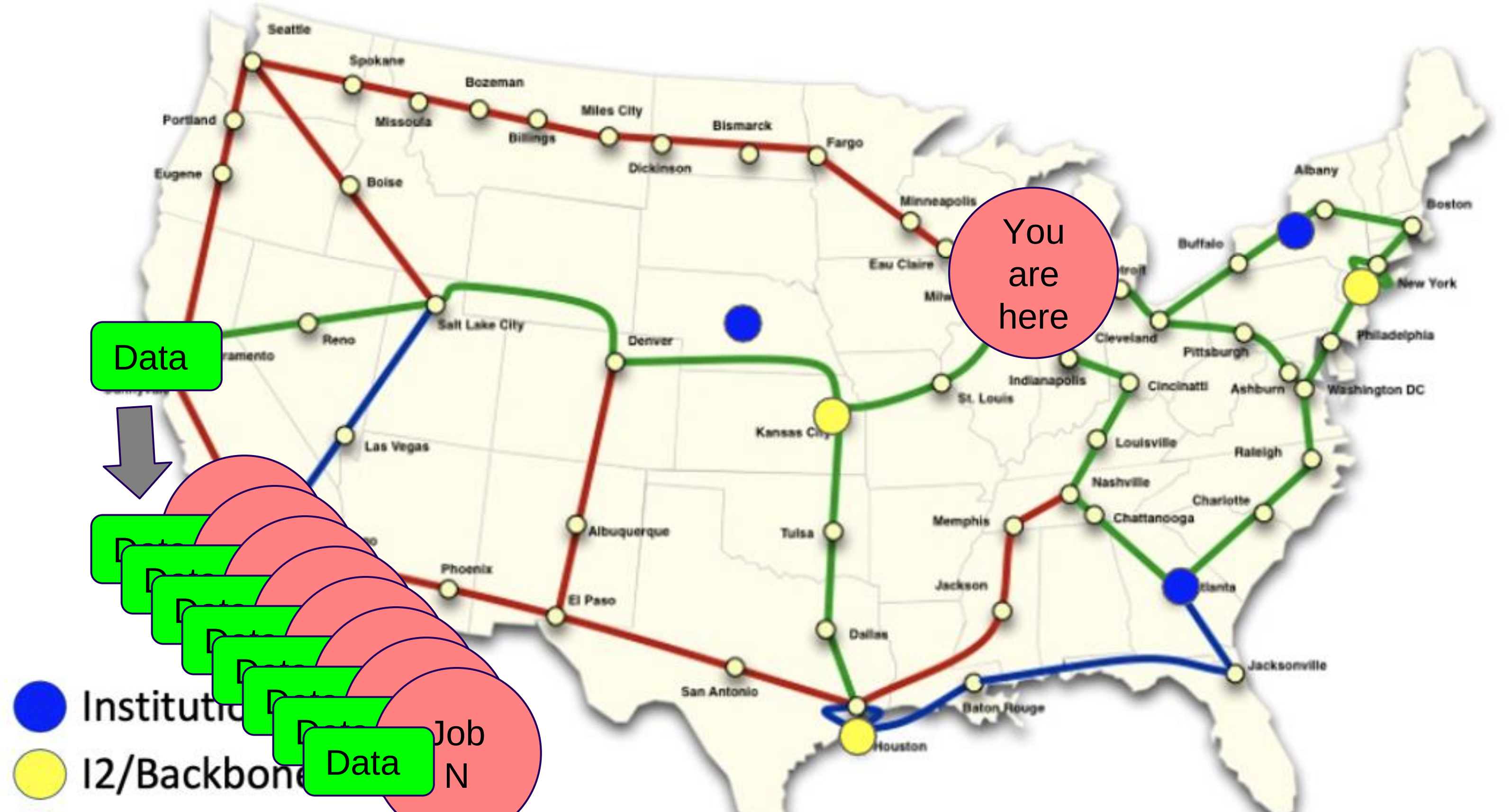
Open Science Data Federation (OSDF)



Open Science Data Federation (OSDF)



Open Science Data Federation (OSDF)



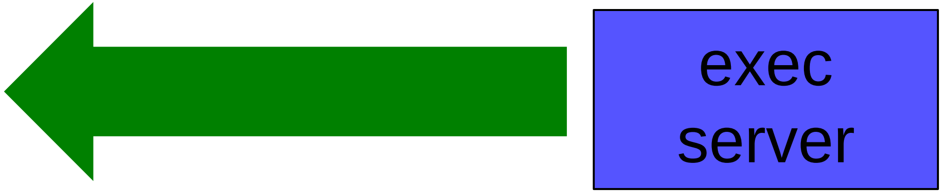
In the Submit File

```
transfer_input_files = osdf:///ospool/apXX/data/username/…
```

3 slashes, not 2!

How about output?

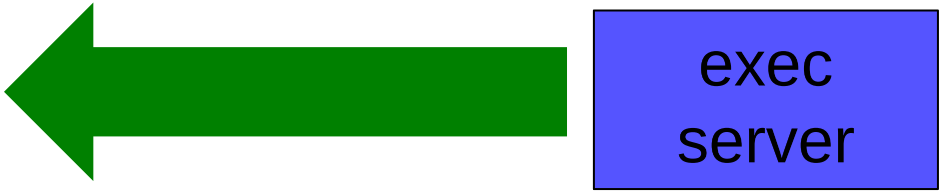
Output for HTC and OSPool



exec server

amount	method of delivery
~~words~~	~~within executable or arguments?~~
tiny – **1GB, total**	HTCondor file transfer
1GB - 20GB, unique or shared	OSDF
20GB+, total	shared file system (local copy, local execute servers)

Output for HTC and OSPool



exec
server

amount	method of delivery
~~words~~	~~within executable or arguments?~~
tiny – **1GB, total**	HTCondor file transfer
1GB – 20GB, unique or shared	OSDF
20GB+, total	shared file system (local copy, local execute servers)

Writing to OSDF

```
transfer_output_remaps = "Output.txt =
osdf:///ospool/apXX/data/username/Output.txt"
```

**Use semicolons (;) to separate multiple entries*

Moving Data on the OSPool

- ~~Overview / Things to Consider~~
- ~~HTCondor File Transfer~~
- ~~OSDF~~
- **Other Considerations**

Working with Even Larger Data

- Only use these options if you MUST!!
  - Comes with limitations on site accessibility and/or job performance, and extra data management concerns

file size	method of delivery
words	within executable or arguments?
tiny – 10MB per file	HTCondor file transfer (up to 1GB total per-job)
10MB – 1GB, shared	download from web server (local caching)
1GB - 10GB, unique or shared	OSDF (regional replication)
10 GB - TBs	shared file system (local copy, local execute servers)

Cleaning Up Old Data

Make sure to delete data when you no longer need it in the origin!!!

- Servers do NOT have unlimited space!
  - Some may regularly clean old data for you. Check with local support.

Quick Reference

Option	Input or Output?	File size limits	Placing files	In-job file movement	Accessibility?
HTCondor file transfer	Both	100 MB/file (in), 1 GB/file (out); 1 GB/tot (either)	via HTCondor access point	via HTCondor submit file	anywhere HTCondor jobs can run
OSDF	Both	20 GB/file	via HTCondor access point or Pelican origin	transfer_*_file	OSG-wide (most sites), by anyone
Shared filesystem	Input, likely output	TBs (may vary)	via mount location (may vary)	use directly, or copy into/out of execute dir	local cluster, only by YOU (usually)

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 is Pelican?

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

Questions?

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/

Additional Slides

Shared Filesystem Details

(Local) Shared Filesystems

- data stored on file servers, but network-mounted to local submit and execute servers
- use local user accounts for file permissions
  - Jobs run as YOU!
  - readable (input) and writable (output, most of the time)
- *MOST* perform better with fewer large files (versus many small files of typical HTC)

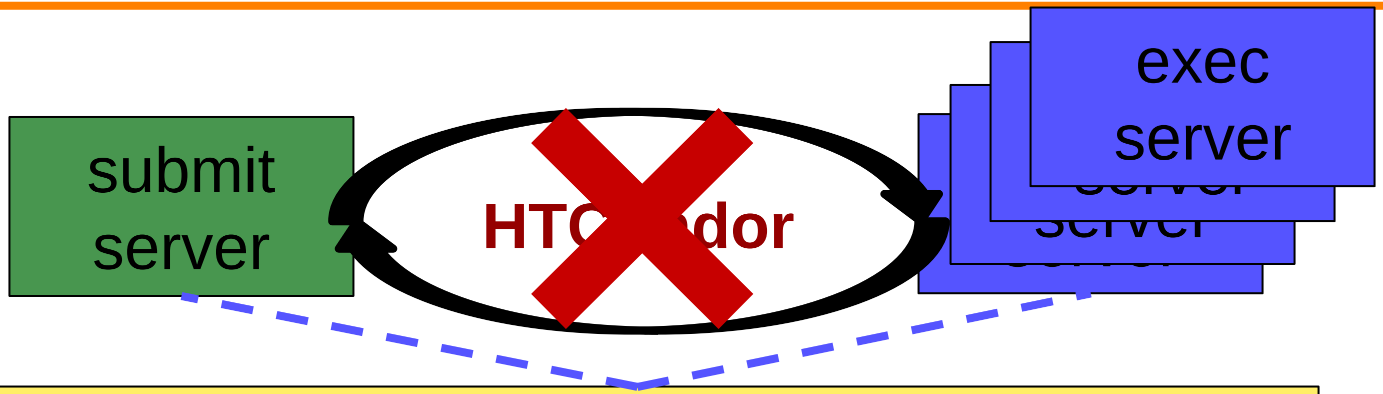
Shared FS Technologies

- *via network mount*
  - NFS
  - AFS
  - Lustre
  - Isilon (may use NSF mount)
- *distributed file systems (data on many exec servers)*
  - HDFS (Hadoop)
  - CEPH

Shared FS Configurations

1. Submit directories *WITHIN* the shared filesystem
   - most campus clusters
   - limits HTC capabilities!!
2. Shared filesystem separate from local submission directories
   - supplement local HTC systems
   - treated more as a repository for VERY large data (>GBs)
3. Read-only (input-only) shared filesystem
   - Treated as a repository for VERY large input, only

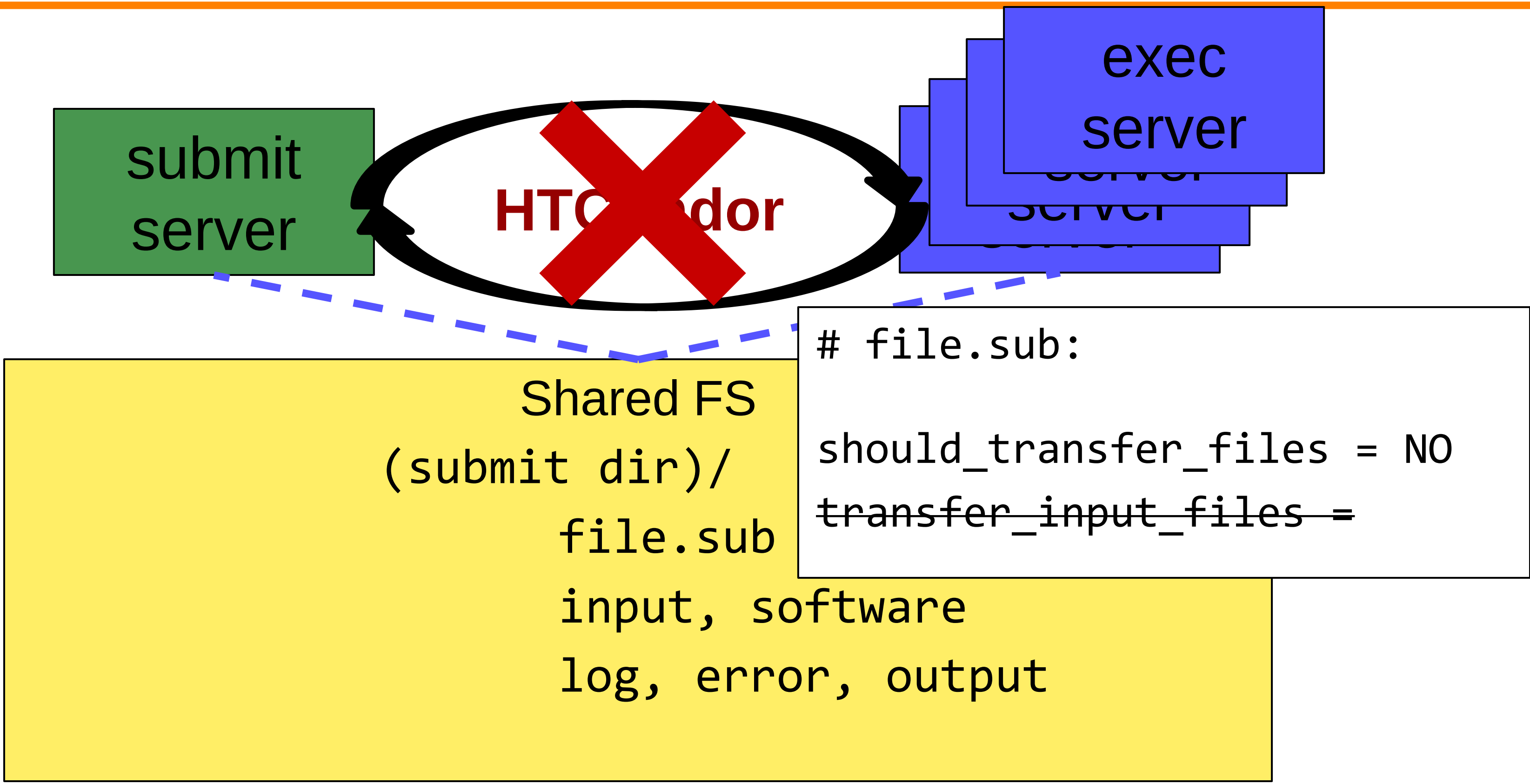
Submit dir within shared FS



Shared FS

```
(submit dir)/
          file.sub
          input, software
          log, error, output
```

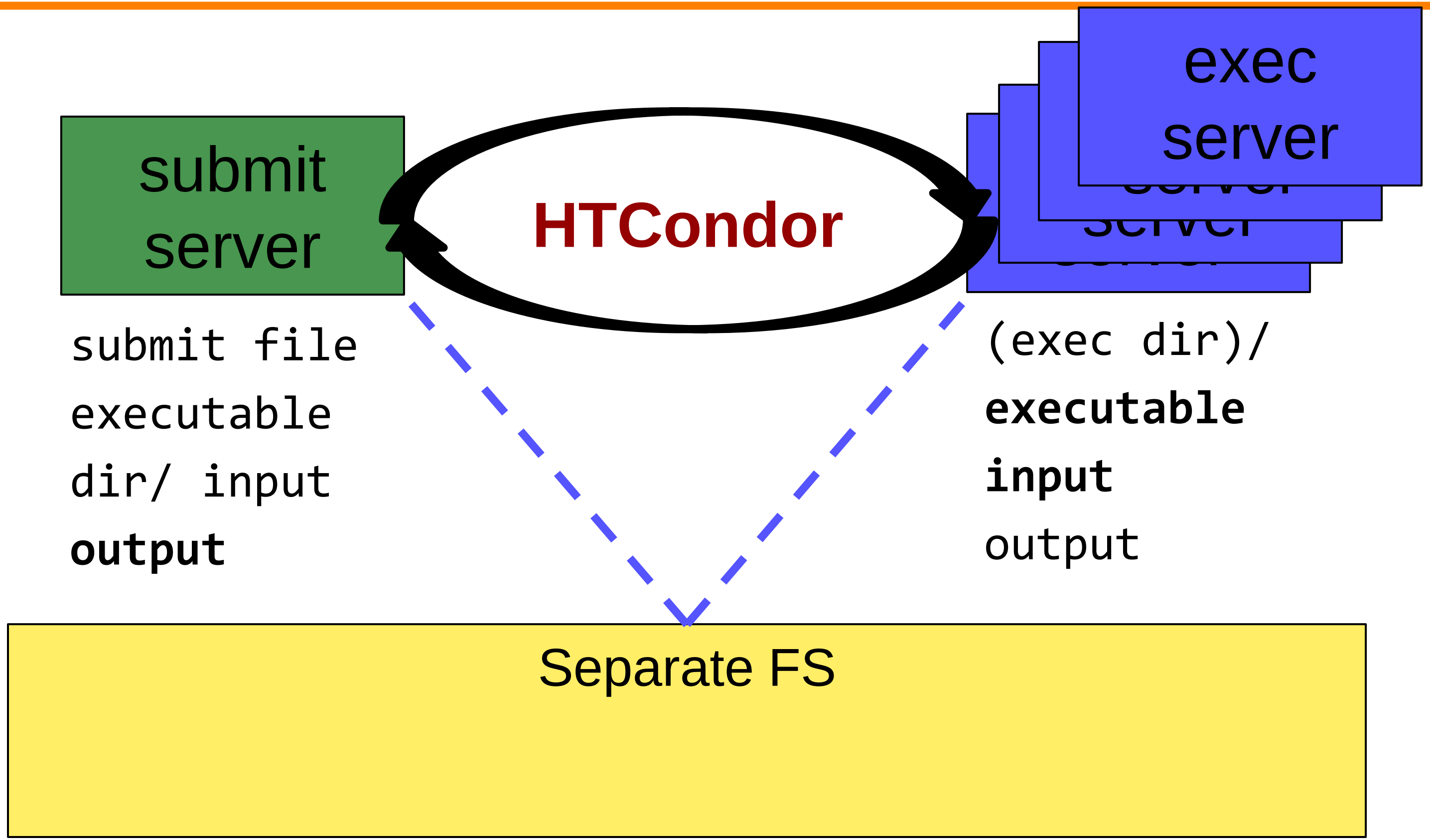
Submit dir within shared FS



```
# file.sub:

should_transfer_files = NO
```
~~transfer_input_files =~~

Separate shared FS



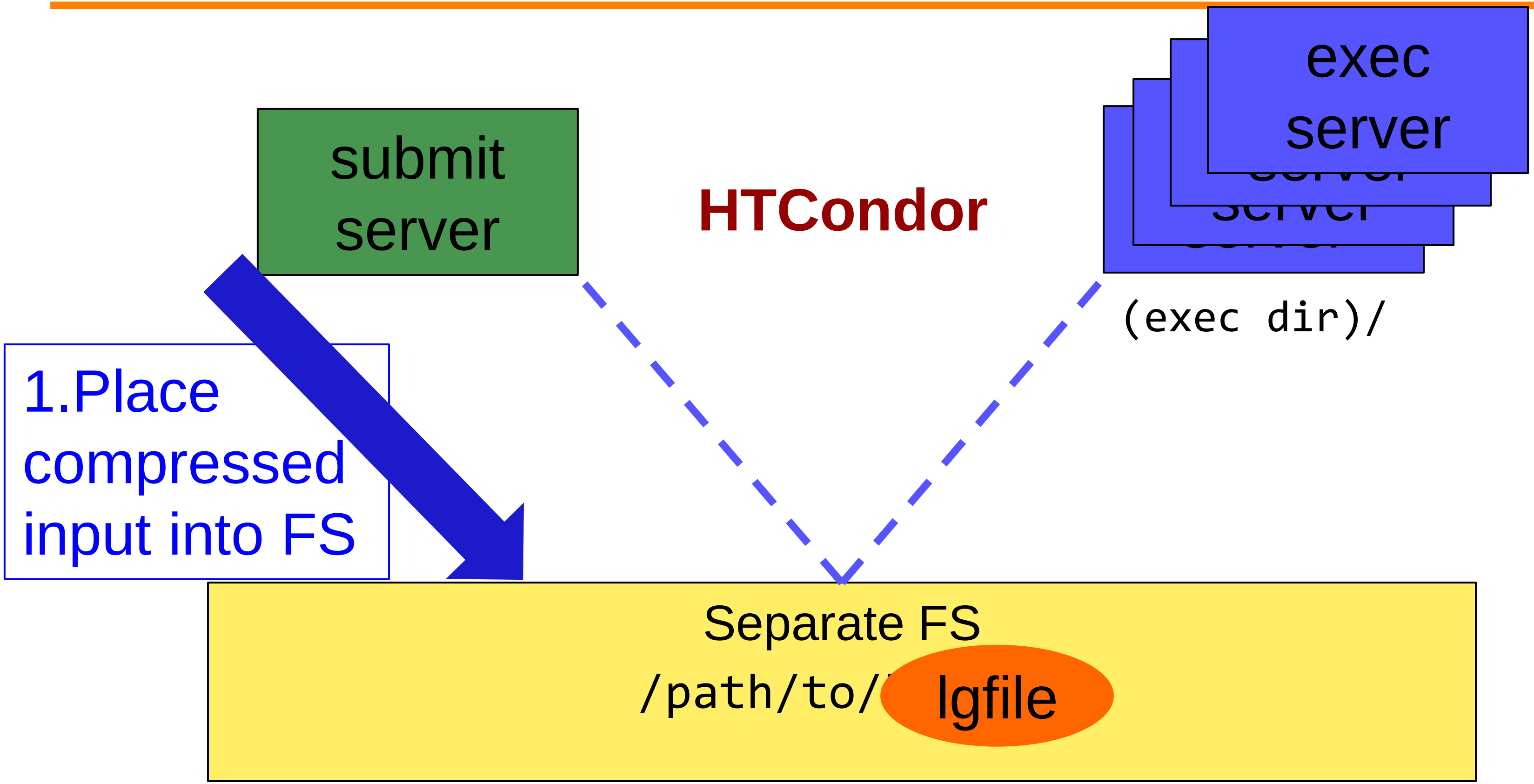
```
submit file
executable
dir/ input
output
```

output

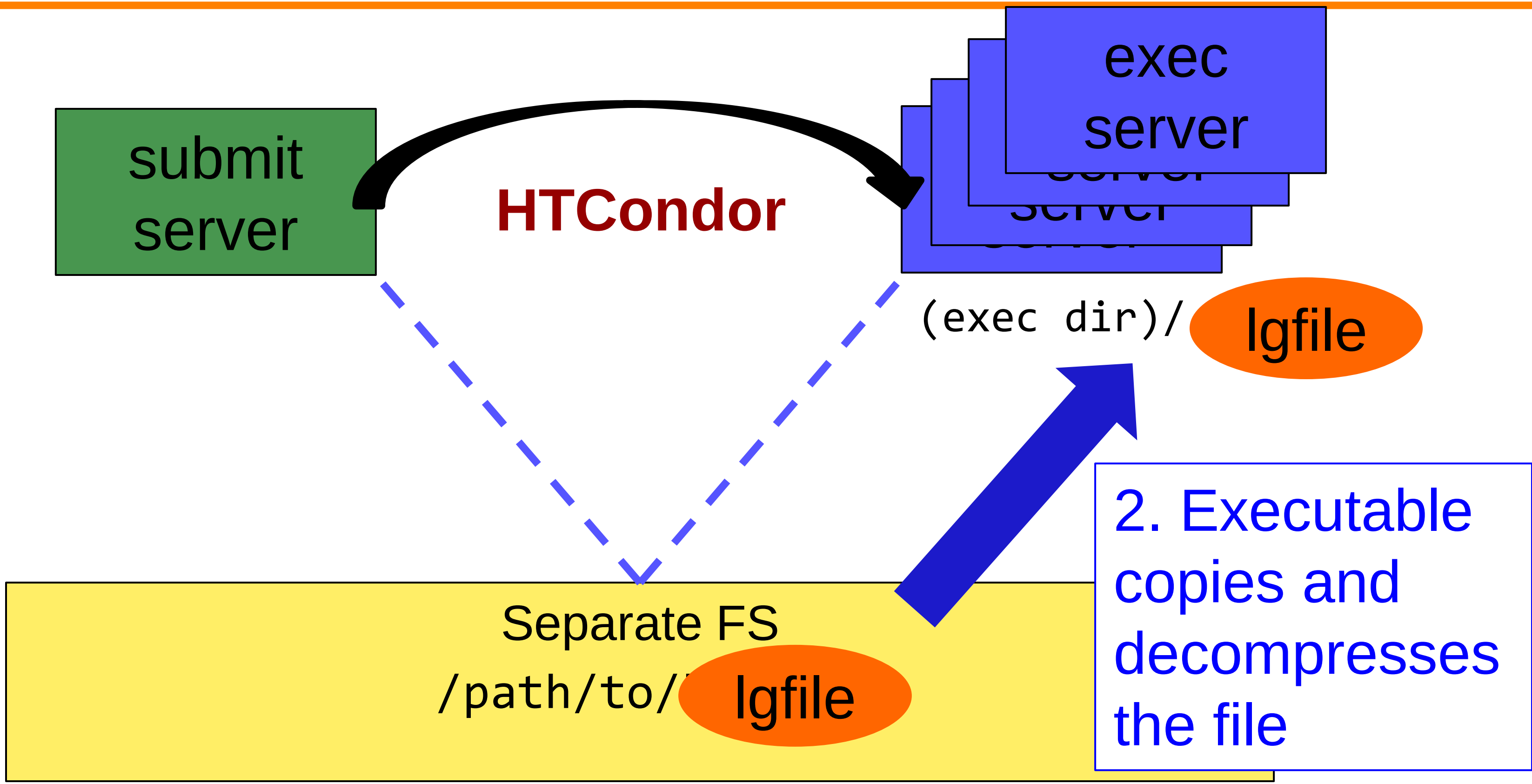
```
(exec dir)/
executable
input
output
```

executable, **input**

Separate shared FS - Input

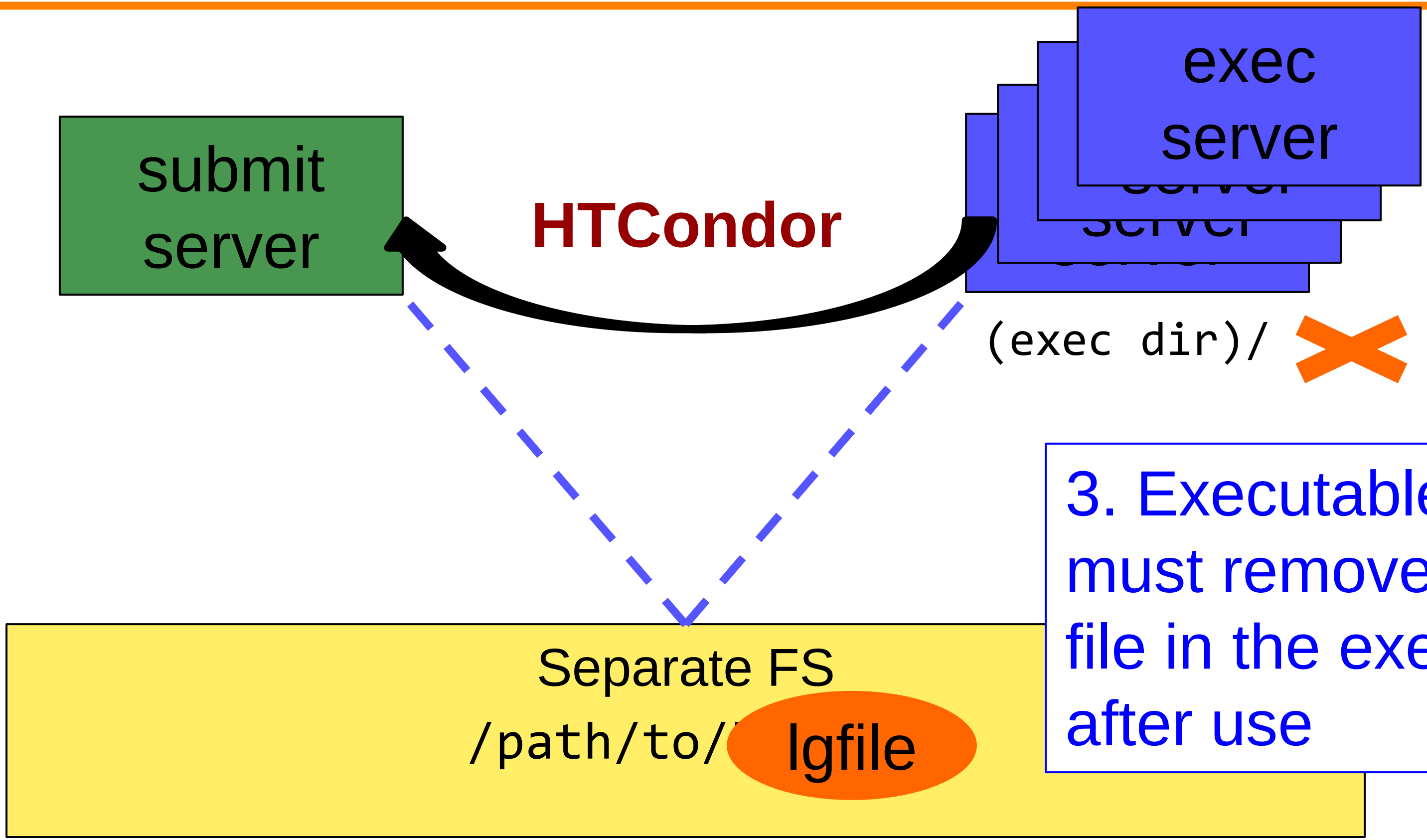


Separate shared FS - Input



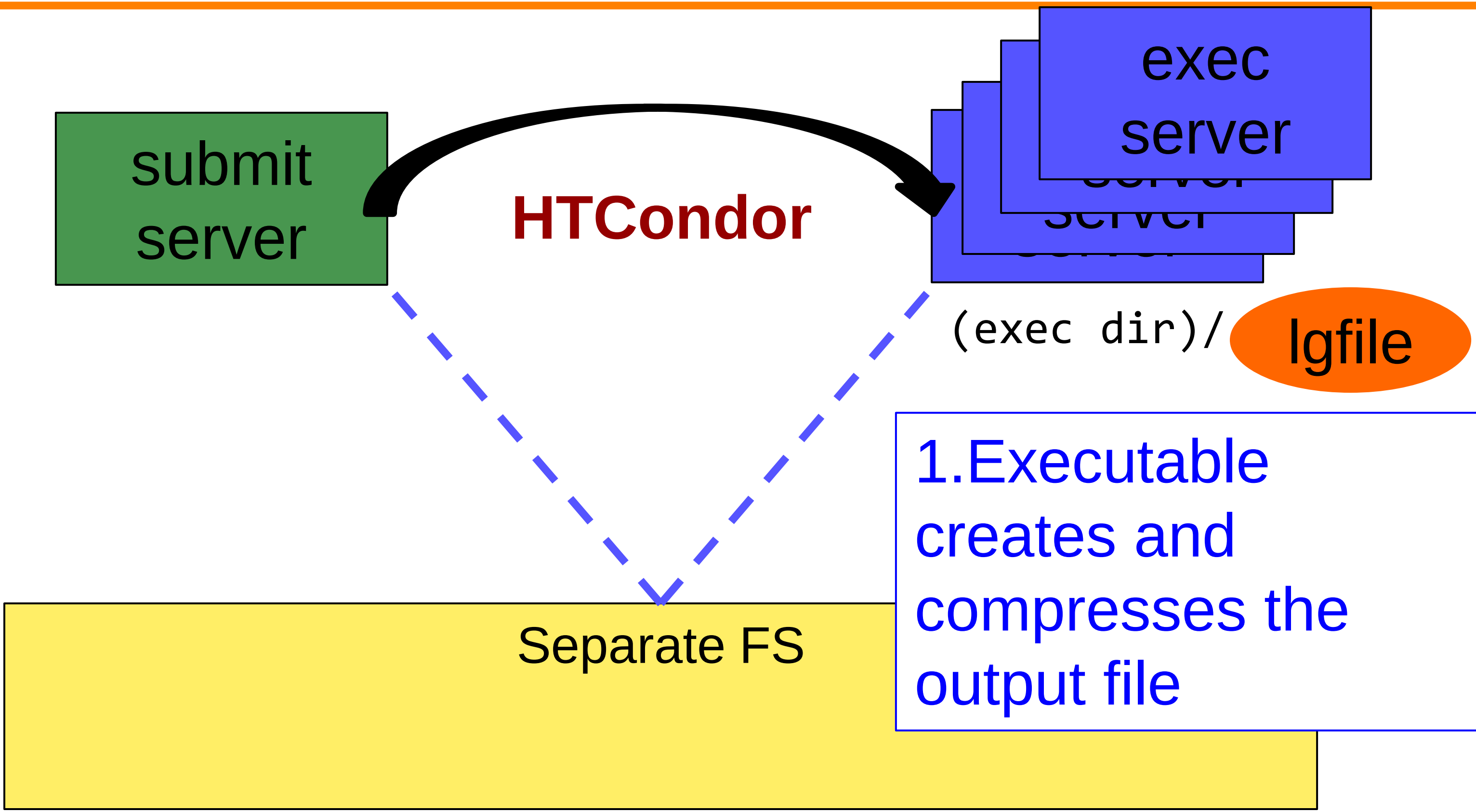
2. Executable copies and decompresses the file

Separate shared FS - Input



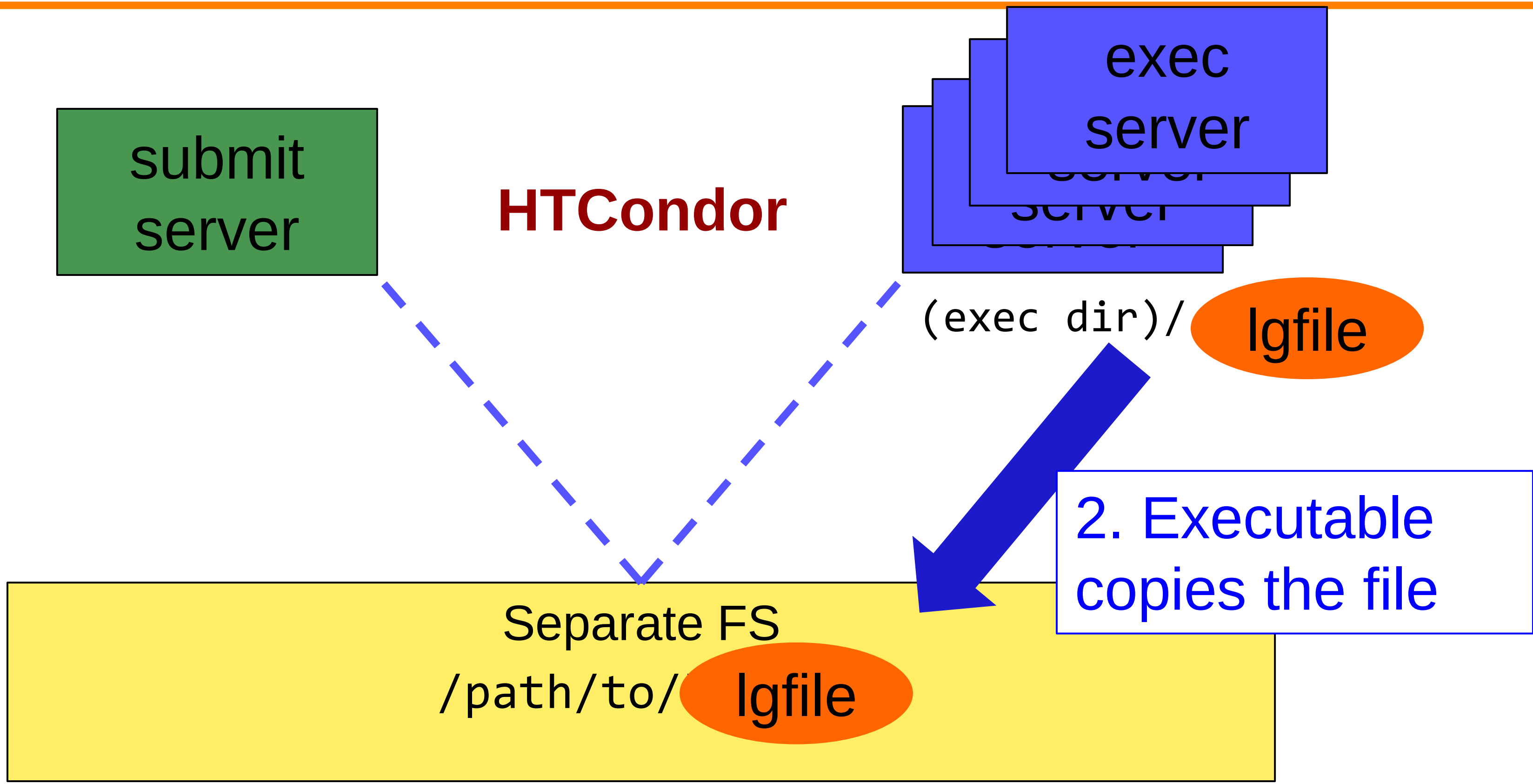
3. Executable must remove the file in the exec dir after use

Separate shared FS - Output



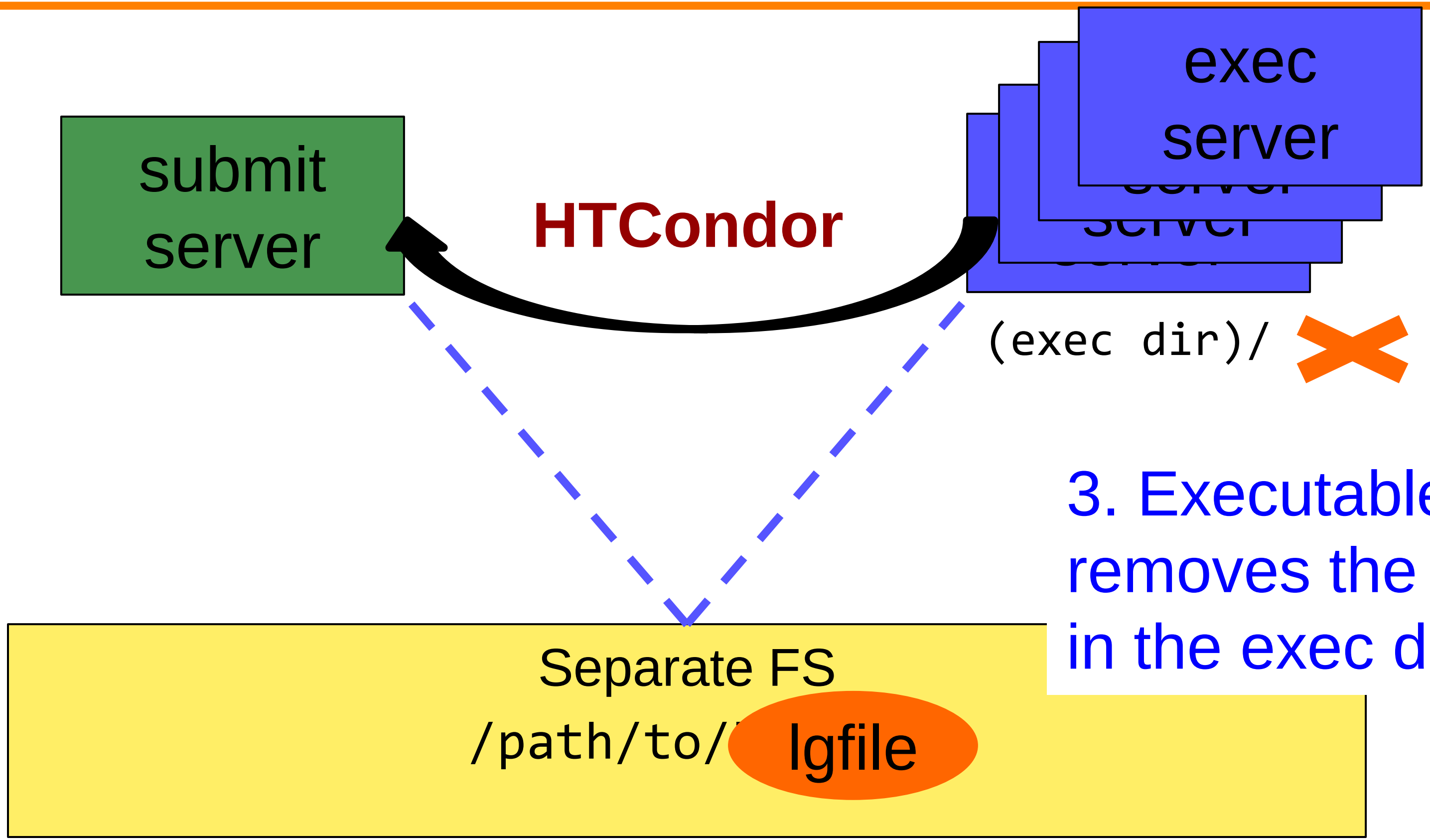
1.Executable creates and compresses the output file

Separate shared FS - Output



2. Executable copies the file

Separate shared FS - Output



3. Executable removes the file in the exec dir