The HTCSS Data Story

MORGRIDGE
INSTITUTE FOR RESEARCH
RESEARCH COMPUTING

FEARLESS SCIENCE

Management is the Key to Success

In order to effectively share capacity, it is essential to manage finite computing resources. What does this mean?

1. HTCondor must understand/**model the capacity** of the resources.
2. **Jobs** should describe their resource needs.
3. HTCondor needs enforcement **mechanisms and policy**.

This is natural for some resource types (cores, memory):

- "I have 48 cores on my computer"
- "My job needs 8 cores to function"
- "If the job uses more cores than requested, preempt it!"

CPU cores are easy to enumerate, the OS provides an obvious way to measure & several enforcement mechanisms, and many users understand their needs.

HTCondor Wants Your Data!

An oft-overlooked finite resource is ‘data movement’:

- Storage services have a finite capacity for moving files in parallel (concurrent xfers).
  - Similarly, they have a limit on the number of IO Operations Per Second (IOPS)
- Networks have a finite capacity (Mbps).
- The EP has finite space to hold sandbox data for the running job.
- The user (probably) has a finite space for output or checkpoint files.

Any one of these items can ruin your day if they go unmanaged!

If you don’t tell HTCondor about its data, then it can’t manage it!

You Want HTCondor To Manage Your Data!

If you don’t have HTCondor watch your CPU usage, you end up with a mess.

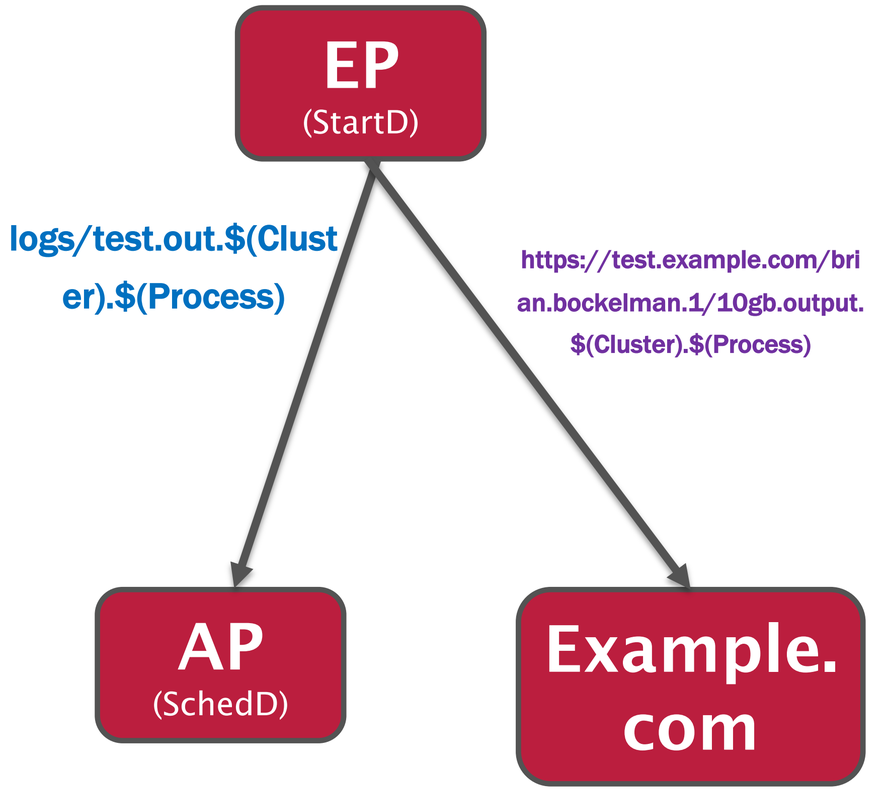
If you don’t have HTCondor watch your data, you end up with a mess!

Let's Make A Deal

What services and guarantees does HTCondor make?

- If you tell HTCondor about your data movement needs (input and output), THEN:
  1. HTCondor will trigger the data movement:
     - It will move files and directories between the AP and the EP.
     - Given a (supported) URL, it will download/upload files from the URL.
     - HTCondor won't start your job until the inputs are node-local. **No shared filesystem failures mid-job**.
  2. If a data movement failure occurs, HTCondor can evaluate policy to decide what to do next.
  3. It will manage the data movement, avoiding overscheduling resources.

Triggering Data Movement




```
universe = vanilla
request_cpus = 1
request_memory = 1000
log = logs/test.$(Cluster).log
output = logs/test.out.$(Cluster).$(Process)
error = logs/test.err.$(Cluster).$(Process)
executable = test.sh
```

Transferred by HTCondor between AP and EP

Triggers transfer to a third-party service

```
transfer_input_files = https://test.example.com/brian.bockelman.1/10gb
transfer_output_files = test.output
transfer_output_remaps =
"test.output=https://test.example.com/brian.bockelman.1/10gb.output.$(
Cluster).$(Process);"
queue 100
```

Data Transfer Policies

If the AP is aware of the data transfers performed, it can manage policies on failure.

Examples:

- Consider “file not found” errors fatal until a user intervenes.
- Retry certain transfers up to 5 times.
- Change the job to avoid problematic sites.
- Consider any jobs with more than 1GB output to be mistakes.

No visibility to transfers = no policy mechanisms!

Data Movement Management within HTCondor

The most mature piece of data movement management is around file transfer concurrency:

- The AP maintains a "file transfer queue" and a **limit on the maximum** number of file transfers permitted in parallel.
- The AP monitors the storage and network performance, continuously adjusting the file transfer concurrency.
  - Actual algorithm is somewhat "magic," but the goal is to maximize throughput, reducing concurrency if the I/O load appears too high.
- "Transfer jobs" in the queue are scheduled in a round-robin fashion to avoid a single user from dominating the AP's capacity.

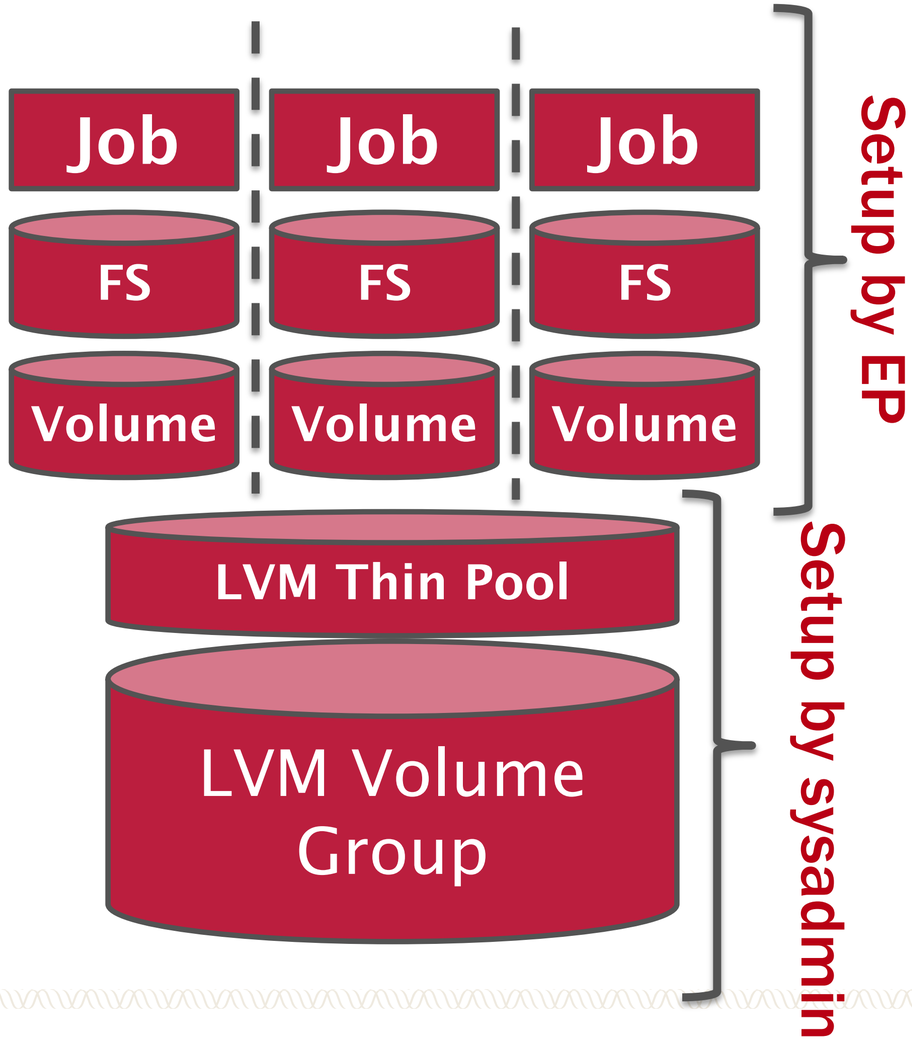
Coming Soon: Improved Sandbox Management

HTCondor 9.9.0 (**finally!**) includes the capability to carefully manage the storage used at the EP sandbox:

- The administrator must hand over a raw block device to the EP (via LVM2)
  - Alternately: the EP can create an empty loopback file in an existing filesystem - at a sizeable performance penalty.
- For a match, the EP will allocate a thinly-provisioned volume from the storage pool.
  - This can be encrypted if desired.
  - Will be at least the size of the requested disk space.

Benefits?

- Monitoring resource usage becomes a cheap operation ("df") and can be done frequently.
- An enforcement mechanism to prevents users from causing problems for other people

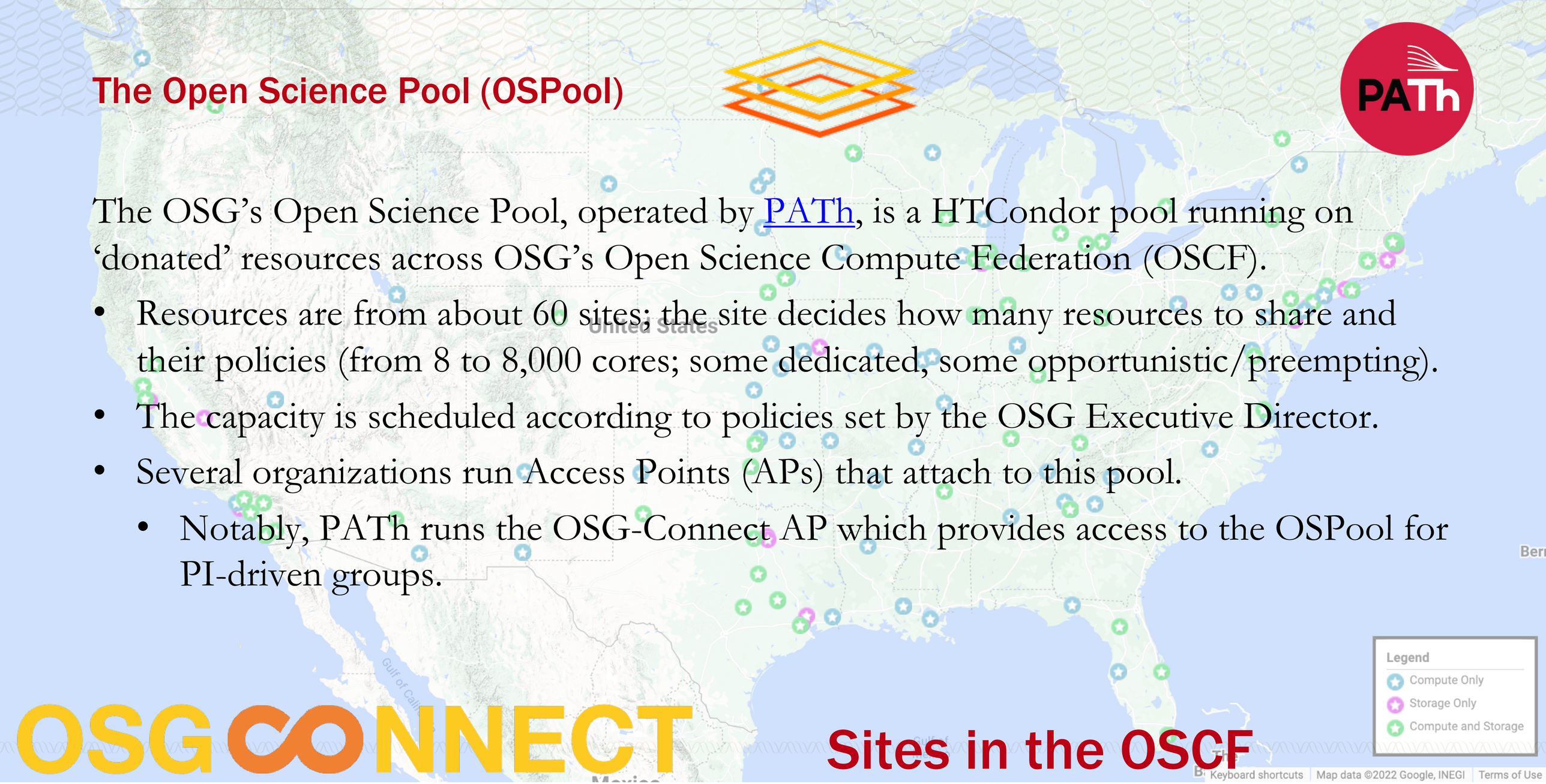


Managing Data Resources on the Open Science Pool

The Open Science Pool (OSPool)

The OSG's Open Science Pool, operated by PATh, is a HTCondor pool running on 'donated' resources across OSG's Open Science Compute Federation (OSCF).

- Resources are from about 60 sites; the site decides how many resources to share and their policies (from 8 to 8,000 cores; some dedicated, some opportunistic/preempting).
- The capacity is scheduled according to policies set by the OSG Executive Director.
- Several organizations run Access Points (APs) that attach to this pool.
  - Notably, PATh runs the OSG-Connect AP which provides access to the OSPool for PI-driven groups.



Sites in the OSCF

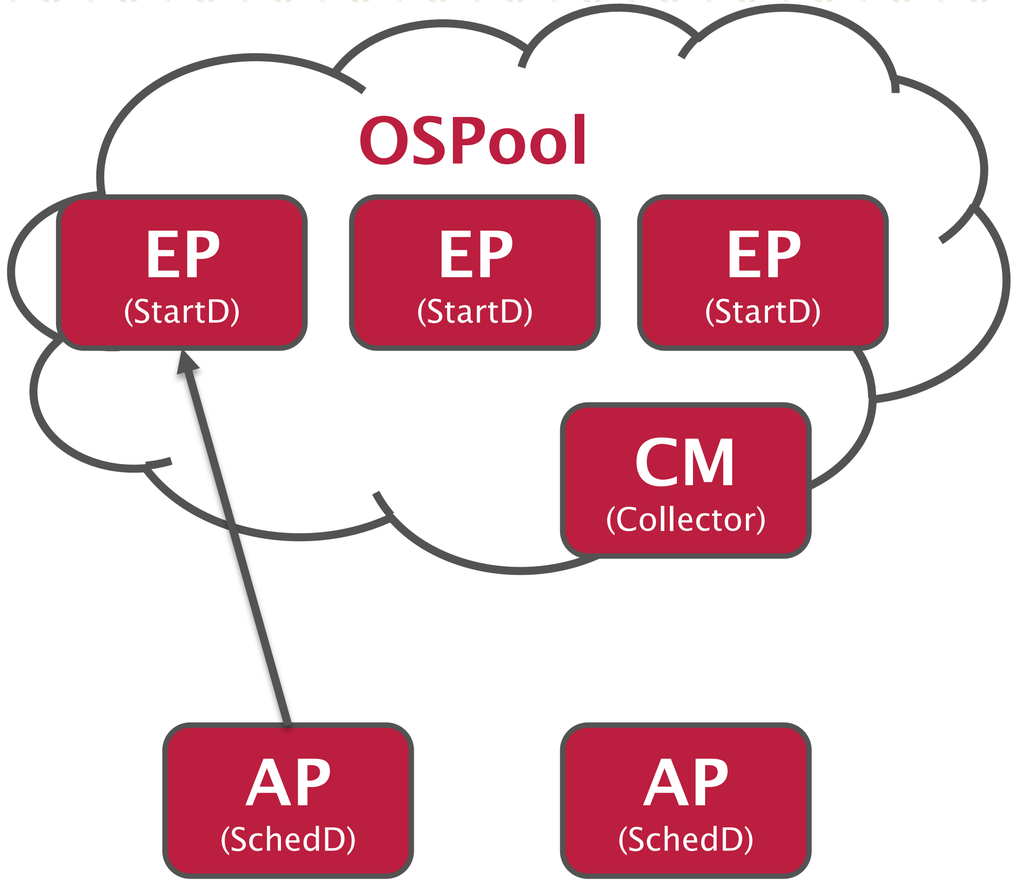
The OSPool

To the user, the OSPool looks like a single, 'cluster-like' pool.

- In reality, the pool is very heterogeneous; on a typical day, the pool is spread across 40 sites.
- The connectivity between any given AP and EP may be 100Mbps or 100,000Mbps!

Hence, it's important to let HTCondor know about your data movement.

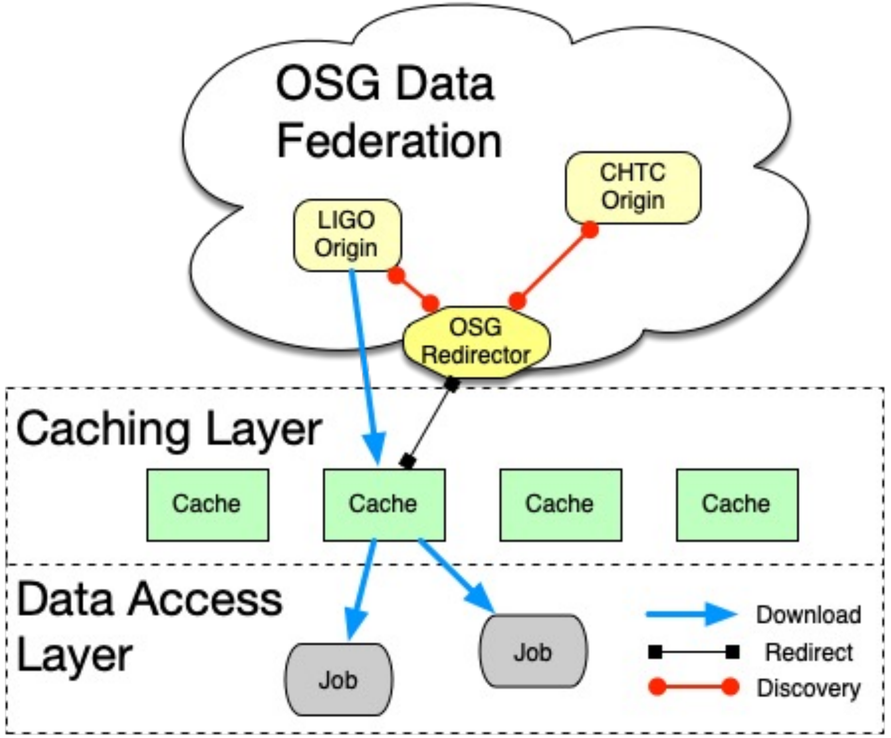
- Otherwise, Murphy's Law guarantees your job needing 1,000,000MB of input will land on a host with a dialup connection.



The OSPool and the Open Science Data Federation

The OSPool pairs nicely with another OSG Service, the Open Science Data Federation (OSDF):

- Among other things, the OSDF (the topic of the next talk!) provides a **file access capability for the OSPool**.
- Organizations can run an origin services that exports part of a filesystem into a **global (filesystem-like) namespace**.
  - PATh operates a series of caches at or nearby sites, reducing the access load on the origin.
- **READING**: Jobs can provide a URL to the AP which causes the job to go through the nearest OSDF **cache** instead of the AP.
- **WRITING**: Jobs can provide a URL to push outputs back to the origin.

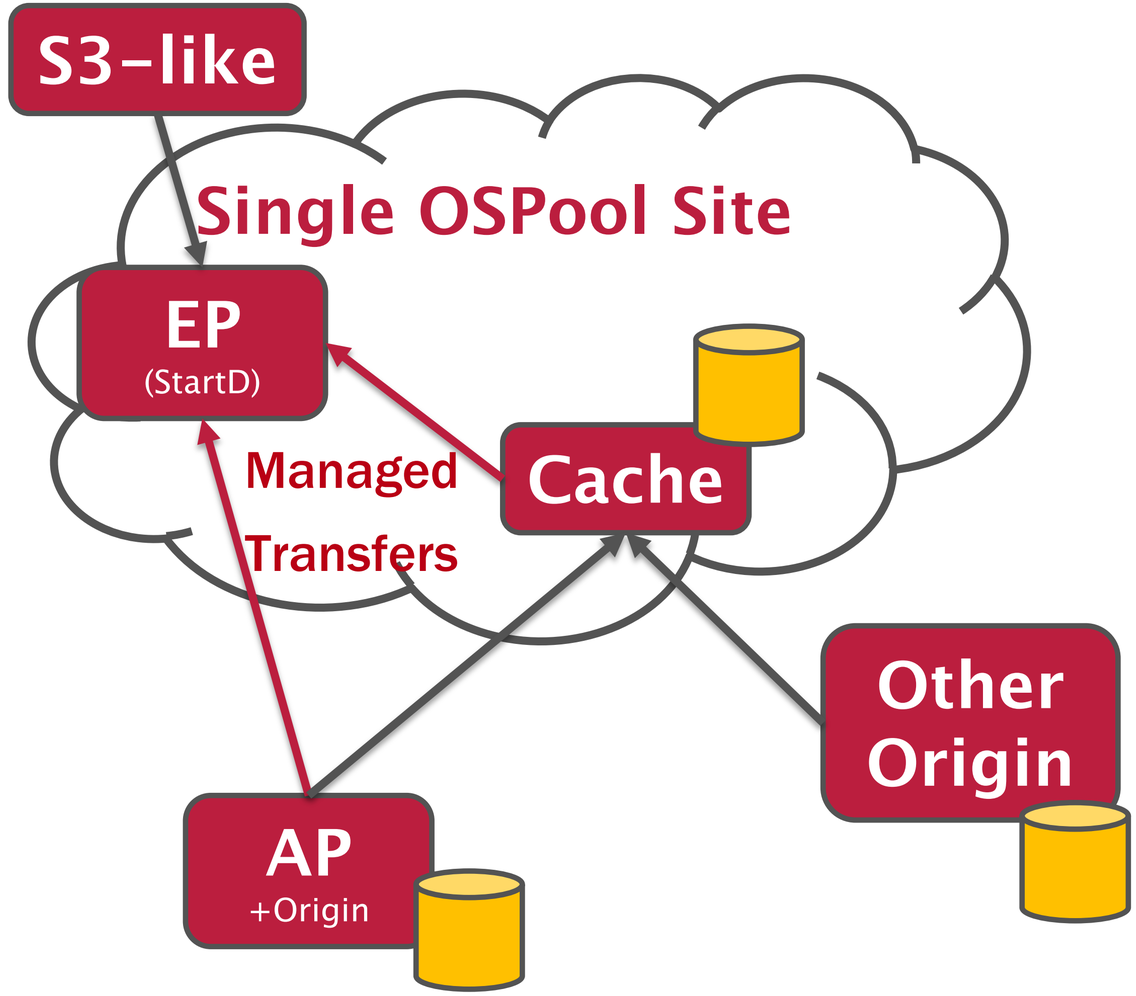


Data Transfer options: OSDF & AP from the OSPool

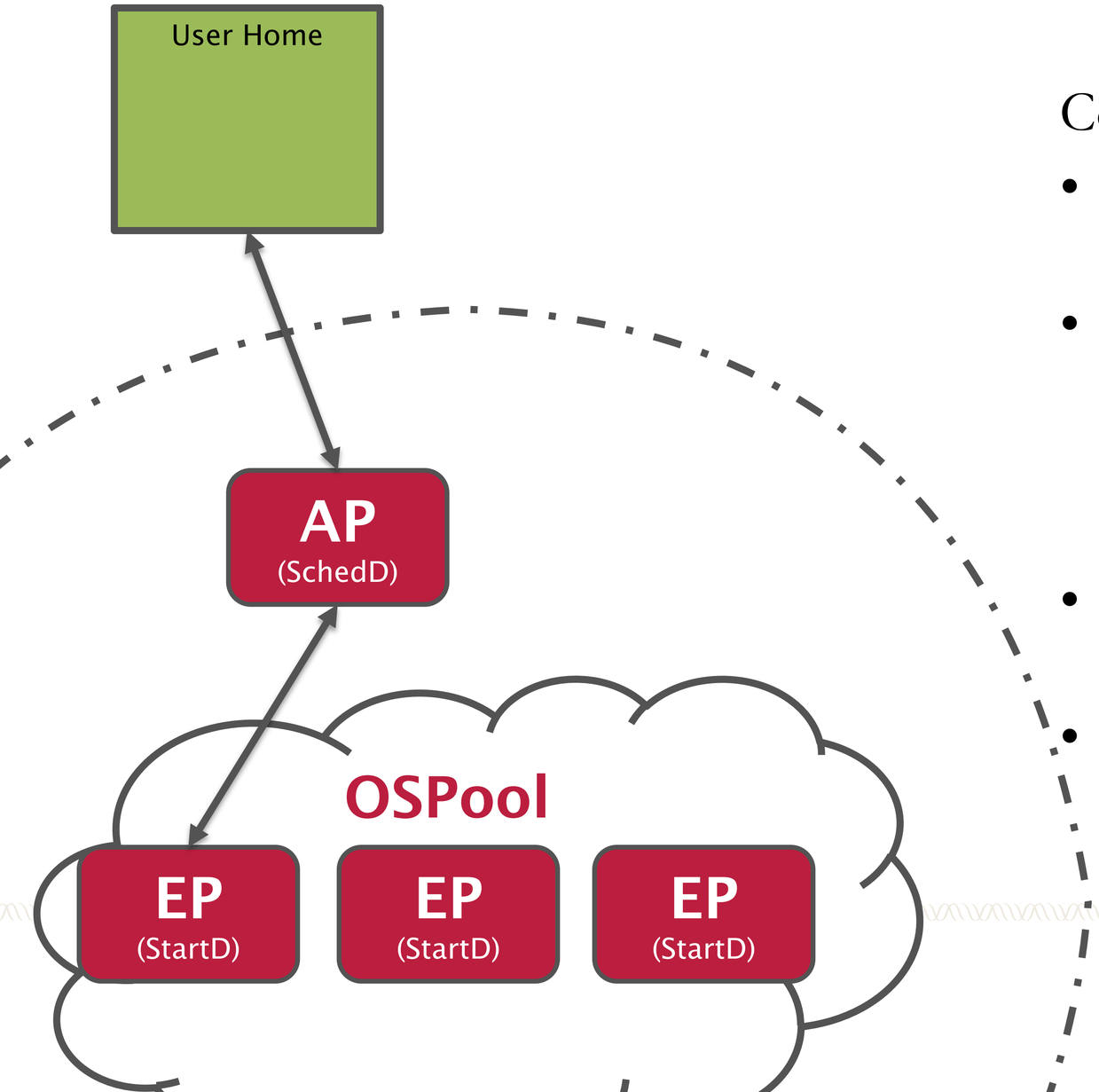
On the OSPool, users can:

- Move data to/from the AP, using HTCondor's built-in concurrency limits.
- From the local cache, using the cache as a way to 'throttle' data transfers at a site.
  - The origin for the cache can be at the AP or a third-party service.
- To/From S3 (or S3-like endpoints).

The AP can manage the authorization for all the data movement to/from the EP, reducing the burden on the user.



Data Source <-> AP <-> EP



Conceptually, there's two important transfers:

- Moving data 'in-processing' between the AP and EP. **This is what we tend to talk about**.
- Moving from the AP to/from the user's permanent "home" / archive.
  - **We usually forget this important piece of the workflow**.
- HTCondor <u>can</u> manage this data transfer but has few / no automation or workflow tools.
- On the OSPool, a variety of mechanisms exist – including scp, rsync, or Globus.

What Comes Next?

HTCondor As A File Transfer Tool

HTCondor excels at moving data for jobs.

1. AP has elaborate mechanisms for managing authorization and trust between the two sides.
2. The internal transfer protocol ("CEDAR") is relatively efficient, moving millions of files per day.
3. Jobs have flexible policy and can be organized into workflows.

However, for moving data to an AP from the outside world, one can do "condor_submit –spool" and "condor_transfer_data" – and you're on your own!

Goal: Bring the authorization, data management, and workloads available to AP<->EP transfers to the user.

- We have been doing this with HTPheno project, which simplifies attaching a user's EP to an AP and generating a DAG to synchronize directories. **Can we do this on the fly?**

Storage Management

Surprisingly, HTCSS has few storage management mechanisms:

- APs do not track storage usage per account.
  - Jobs can potentially produce more output than the AP's ability to ingest – leading to lots of badput.
- Checkpoints can easily fill the available spool space.

Our axioms:

- All writes should go to explicitly allocated space.
- All space should have an explicit owner responsible for the data. The owner can partition and share their space as desired.
- All space should have an explicit reclamation / lifetime policy.

We are excited to share that NSF OAC #2209645 has just been funded to start building a systematic framework for managing storage in HTC.

Take Home Message

HTCSS wants to manage data movement and storage just like any other resource!

- The OSG's OSPool has special integration with the OSDF, enabling users to utilize a diverse set of data origins for inputs and outputs of HTC workflows and using a data access layer of caches.
  - Provides a broad range of tools to fit the workload!
- Allowing HTCondor to manage data movement enables it to help you:
  - Managing is key to effective use of shared resources – improving throughput for all.
  - The AP can execute your policies when movement fails.
- Challenges remain!
  - Storage Management is an open frontier.
  - File transfers to/from the HTCondor System is still being prototyped.

MORGRIDGE
INSTITUTE FOR RESEARCH
RESEARCH COMPUTING

morgridge.org

This project is supported by National Science Foundation under Cooperative Agreement OAC-2030508. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

FEARLESS SCIENCE