

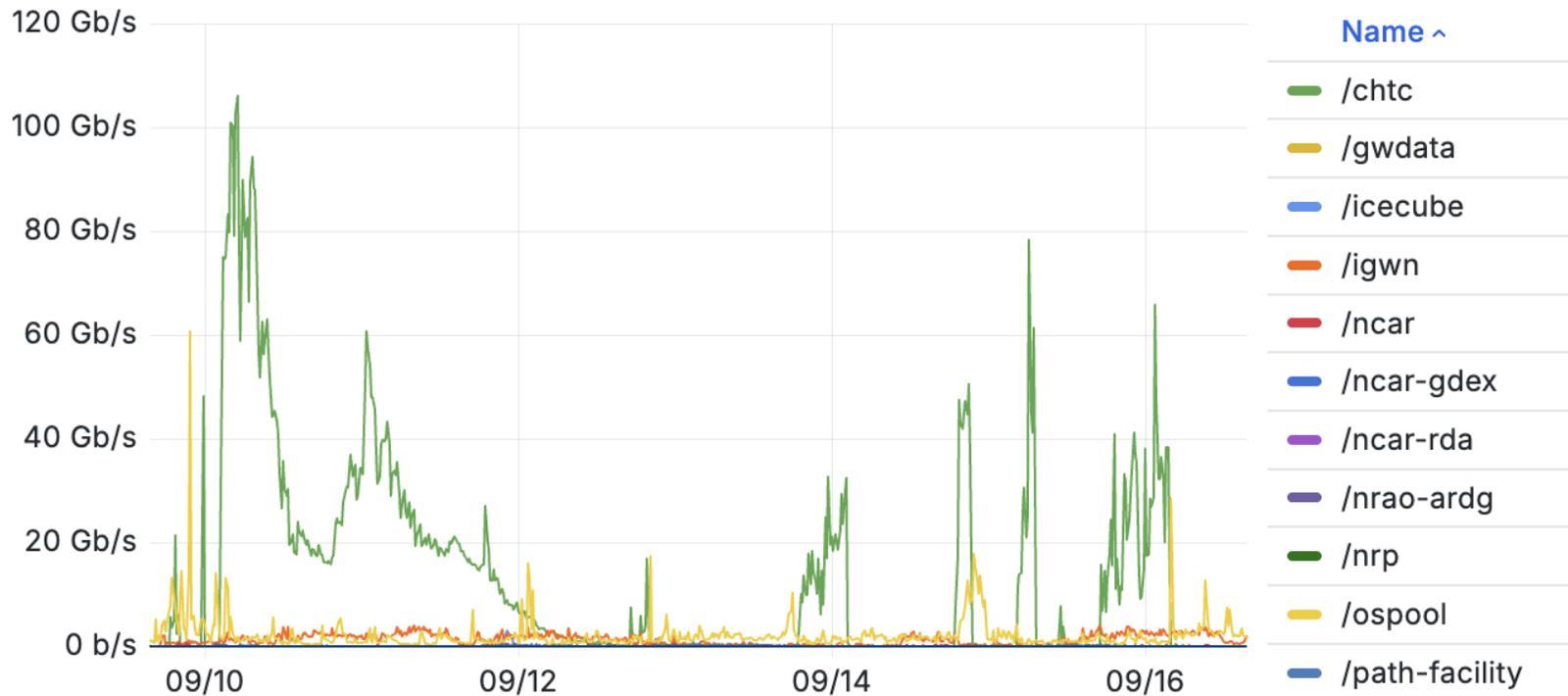
# Files Common Across Jobs and How to Better Handle Them

HTCondor European Workshop 2025

Brian Bockelman

# Check out the neat network graphs!

Client Transfer Rate ⓘ



- ▶ Impressive graphs from a cache at CHTC!
  - ▶ We peak over 120Gbps sustained.
  - ▶ Even more impressive knowing this is fully virtualized Kubernetes networking.
- ▶ ... But it might be less impressive if you know this is often the same SIF container image, over and over, to the same hosts.

# The setup

- ▶ Who has common files?
  - ▶ “I have a great science workflow”
  - ▶ “All my input software is in a container image”
  - ▶ “Oh, and the container is 5GB”
  - ▶ “Oh, and each job is 5 minutes long”
  - ▶ “Oh, and the container image is proprietary”
- ▶ A 5GB container is not a problem ...
  - ▶ ... if the jobs last long enough.
- ▶ A 5-minute job is not a problem ...
  - ▶ ... if the container is small enough!

# Why not a cache?

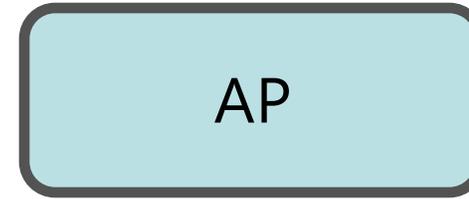
- ▶ A cache sounds like what we need! Why not?
  - ▶ We must to set aside disk space for the cache: **How much?**
  - ▶ What's the “cache key”? How do I know my “container.sif” is the same as your “container.sif”?
    - ▶ **How do we partition the cache?**
  - ▶ Caches work ... until they don't.
    - ▶ **Cache thrashing can make things worse!**

**A really hard problem!**

# First, the mechanism

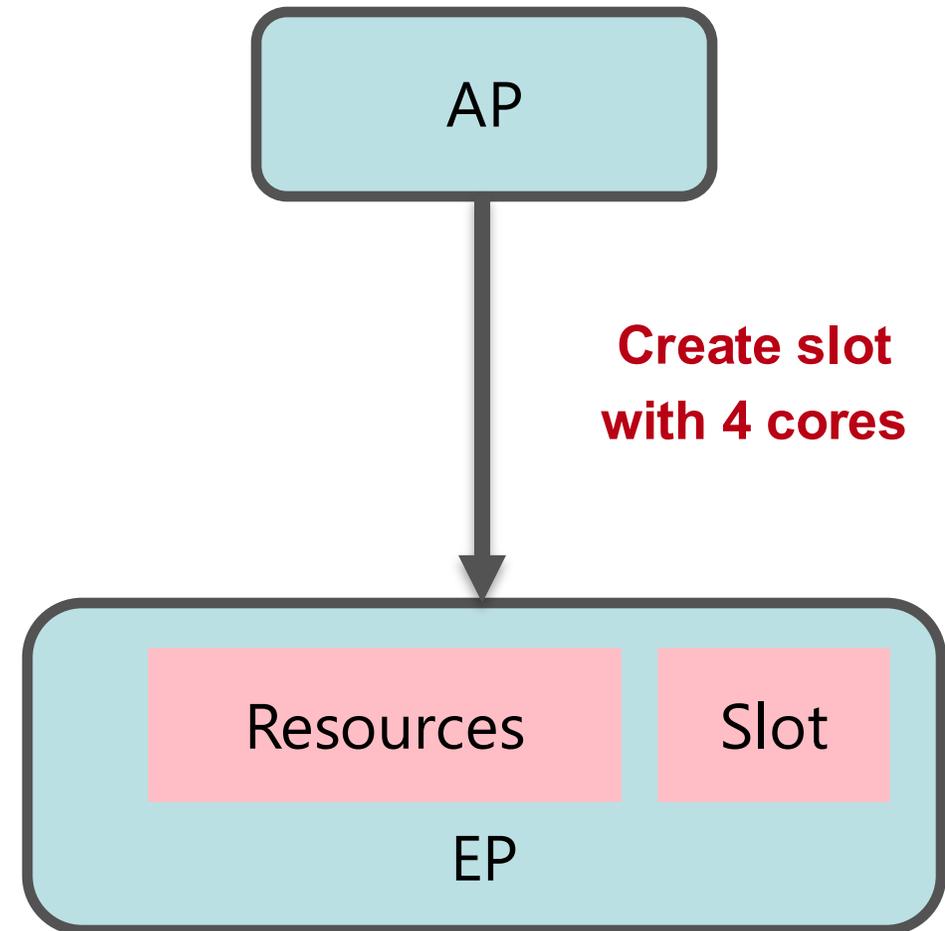
# Concept: AP-managed storage

- ▶ We have a solution already: job slots!
- ▶ That is, the AP already knows how to carve up some resources (CPUs) into a job slot.
  - ▶ Execution is a separate command from the AP.
- ▶ Question: what happens if the AP
  - ▶ Creates a slot containing disk-only resources.
  - ▶ Transfers a bunch of input
  - ▶ ... but doesn't start a job?



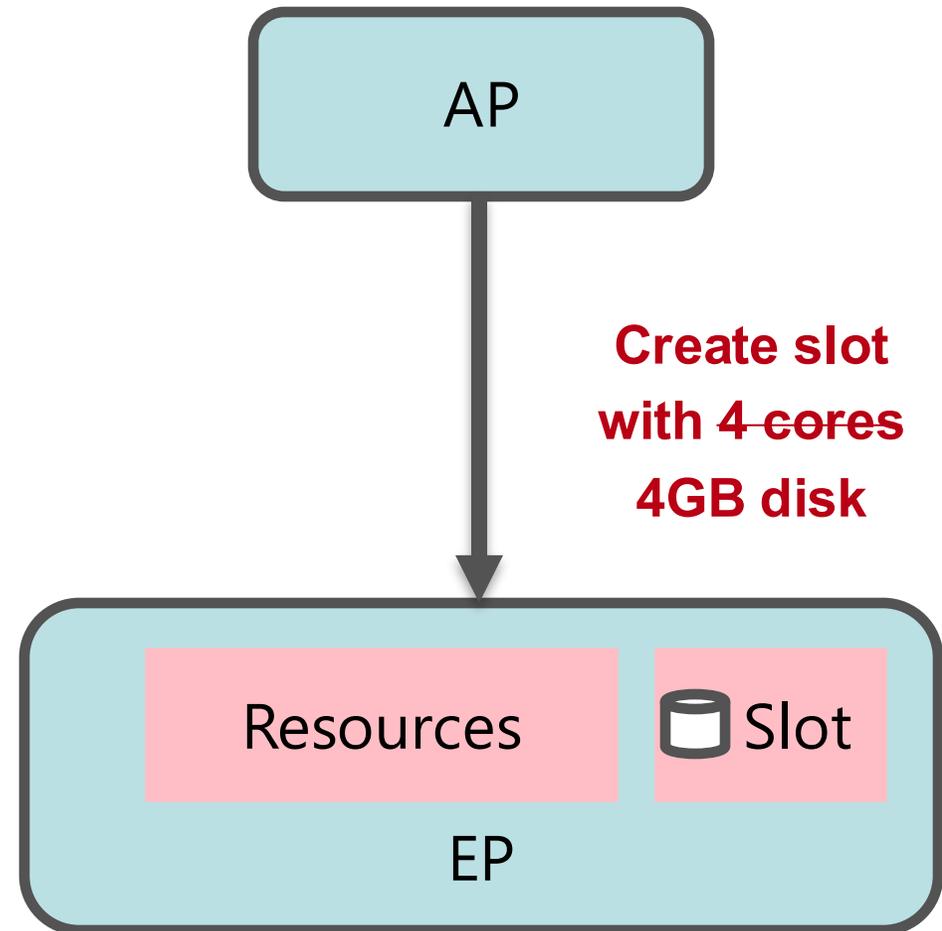
# Concept: AP-managed storage

- ▶ We have a solution already: job slots!
- ▶ That is, the AP already knows how to carve up some resources (CPUs) into a job slot.
  - ▶ Execution is a separate command from the AP.
- ▶ Question: what happens if the AP
  - ▶ Creates a slot containing disk-only resources.
  - ▶ Transfers a bunch of input
  - ▶ ... but doesn't start a job?



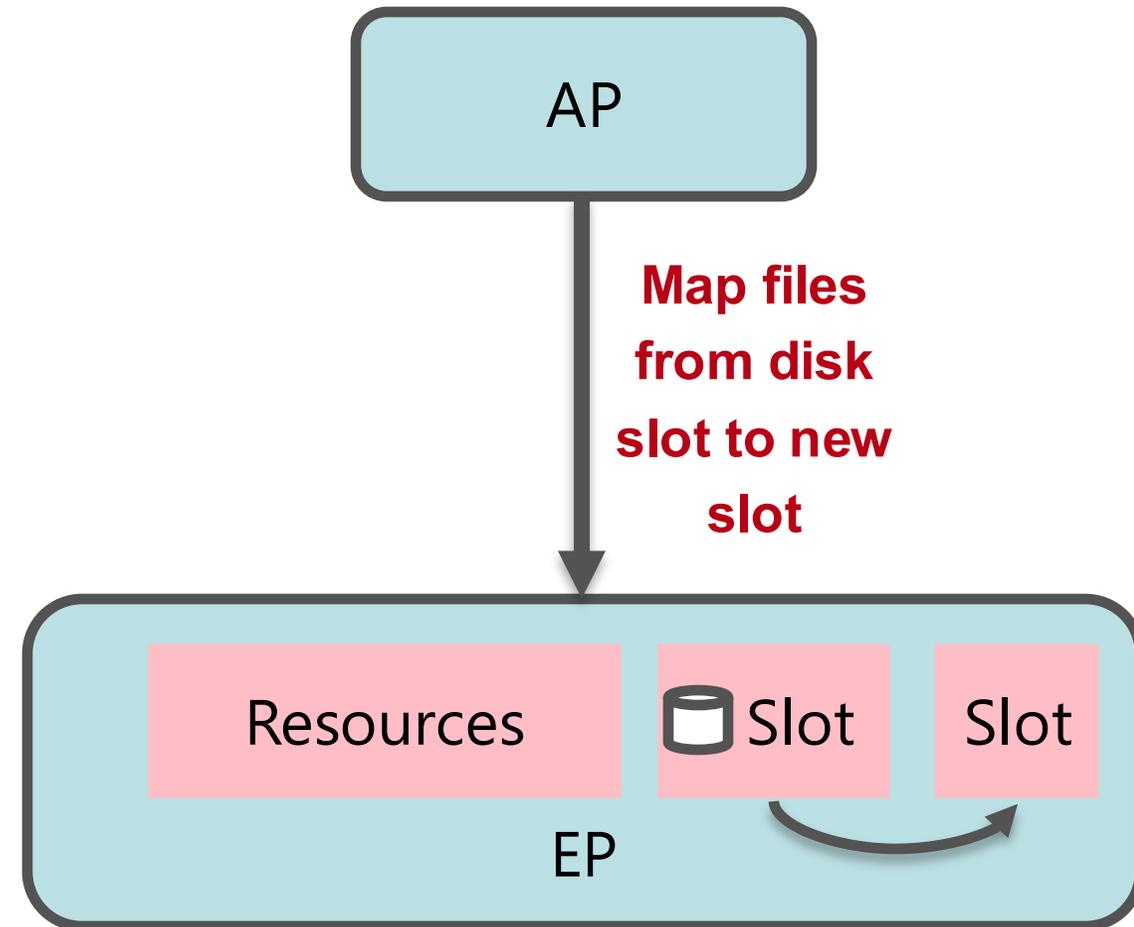
# Concept: AP-managed storage

- ▶ We have a solution already: job slots!
- ▶ That is, the AP already knows how to carve up some resources (CPUs) into a job slot.
  - ▶ Execution is a separate command from the AP.
- ▶ Question: what happens if the AP
  - ▶ Creates a slot containing disk-only resources.
  - ▶ Transfers a bunch of input
  - ▶ ... but doesn't start a job?



# AP understands the workload

- ▶ The AP now controls the end-to-end!
- ▶ It can create a “disk slot” (lovingly referred to as “slot 0” inside the dev team) for what it believes will be useful files at the EP.
- ▶ It can command the EP to “map” the files into the execution environment for a separate slot.
  - ▶ Once the AP is happy with the new slot, it can execute the job.
- ▶ Job sees the common files (read-only).
  - ▶ Using hard links, the files are stored **once** on disk, even if there are 20 jobs running concurrently.





# Now, the User Interface

# How does the AP decide files are “common”

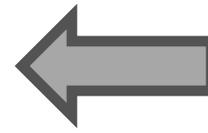
- ▶ When does the AP decide that files are common and can be reused between jobs?

# How does the AP decide files are “common”

- ▶ When does the AP decide that files are common and can be reused between jobs?
  - ▶ **Answer:** It doesn't.
  - ▶ The user specifies the common files in the job.
- ▶ Sadly, there's no “job list” submit file or way to set an attribute for all jobs in a list.
  - ▶ **Ugly:** You need to make sure these are all set the same for each job or Bad Things might happen.
  - ▶ **Ugly:** The files named in each job Had Better be the same (no funny business with lwd / \$PWD!).
  - ▶ **Ugly:** Feature is in rapid development so UI is not well-documented.

# Example submit file

```
universe = vanilla
shell = cat common-file input-file-$(ProcID) \
    > output-file-$(ProcID)
MY.CommonInputFiles = "common-file"
transfer_input_files = input-file-$(ProcID)
transfer_output_files = output-file-$(ProcID)
should_transfer_files = YES
requirements = HasCommonFilesTransfer
request_cpus = 1
request_memory = 1024
log = $(ClusterID).log
queue 10
```



**Beware! This is a ClassAd string; must be quoted.**

# Policies on the AP

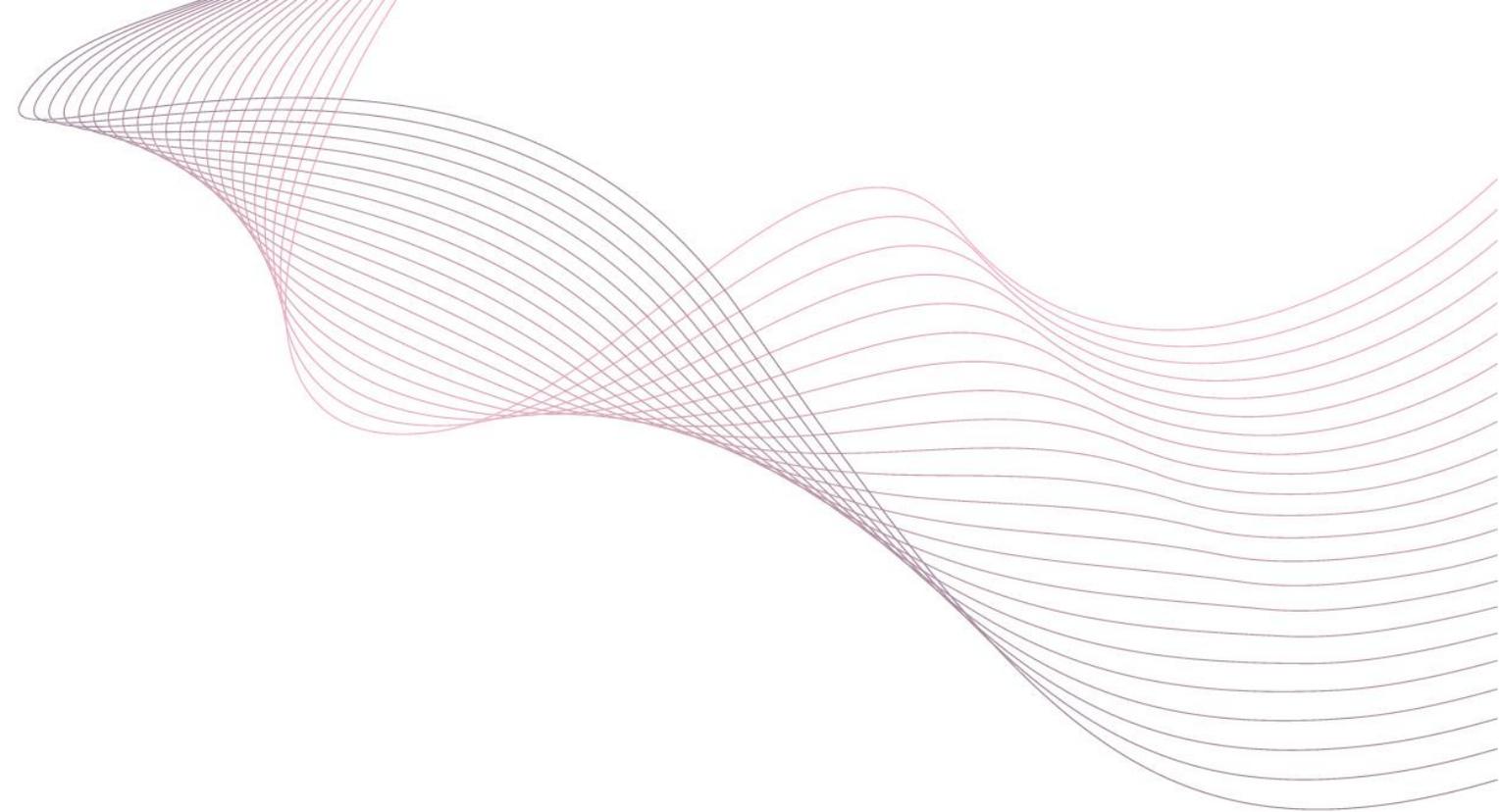
- ▶ The AP does not allow common input shared between jobs in a different job list (“job cluster”).
- ▶ The AP will transfer one set of common inputs **per startd**.
- ▶ The AP does no planning based on file presence.
  - ▶ It does not try to “cluster” jobs on an EP to avoid file transfers.
  - ▶ It will not reuse common inputs between “sequential jobs”

# Policies on the AP

- ▶ The AP does not allow common input shared between jobs in a different job list (“job cluster”).
- ▶ The AP will transfer one set of common inputs **per startd**.
- ▶ The AP does no planning based on file presence.
  - ▶ It does not try to “cluster” jobs on an EP to avoid file transfers.
  - ▶ It will not reuse common inputs between “sequential jobs”
  
- ▶ Implementation scaffolding (here’s the ugly news of things we haven’t finished):
  - ▶ Not actually a separate “slot” on the EP. Not visible in the slot / EP ClassAd.
  - ▶ No disk usage enforcement.
  - ▶ Common input files directory tied to the first job that ran on the host.

# Parting Thoughts

- ▶ Common Input Files is a powerful mechanism to manage input file relationships between jobs.
- ▶ Has potential to greatly reduce total volume moved ...
  - ▶ ... and, importantly, leverage the knowledge in the AP.
- ▶ Currently, lots of “scaffolding” in the implementation:
  - ▶ Missing the conceptual core of a dedicated “input files slot”
  - ▶ UI needs to better ways to specify the “job list” properties.
- ▶ Yet, we have finally crossed the rubicon of shared files – inputs welcome!



# Questions?

**This project is supported by the National Science Foundation under Cooperative Agreements 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.**