What's new in the HTCondor Software Suite (HTCSS) ? What's coming up?

HTC26 – Madison, WI – June 2026

Todd Tannenbaum
Center for High Throughput Computing
University of Wisconsin-Madison

CENTER FOR HIGH THROUGHPUT COMPUTING

HTCondor Software Suite

PARTNERSHIP to ADVANCE THROUGHPUT COMPUTING

Our Email Lists Are Moving!

Good news: you don't have to lift a finger.

- All current subscribers are imported automatically — no need to re-subscribe.
- Old address will forward to new address
- Move Date: June 22nd 2026

These three lists are moving:

- htcondor-users@cs.wisc.edu → htcondor-users@g-groups.wisc.edu
- htcondor-world@cs.wisc.edu → htcondor-announce@g-groups.wisc.edu
- htcondor-security@cs.wisc.edu → htcondor-security@g-groups.wisc.edu

New defaults v25 might be surprising

- Python API (bindings) v1 being dropped -> long live v2 !

```
import htcondor2 as htcondor
```

- System Swap space (virtual memory) will not be used for jobs on the EP by default
- Dropping support for multiple queue statements in a single submit file
  (Use queue foreach, etc.)
- Partitionable Slots enabled by default (instead of static partitioning)
- The job's executable will no longer be renamed to 'condor_exec.exe'
- GPU discovery is enabled on all Execution Points by default

HTCondor v26.x

- Currently at HTCondor v25.11
- HTCondor v25.14 will likely be the v26 release candidate
- HTCondor v26 expected ~Sept 2026
  - Platforms added to supported list:
    - openSUSE 16.0
    - Ubuntu 24.04 on arm64 *(for the NVIDIA DGX workstation!!)*
    - Ubuntu 26.04 on amd64 and arm64
  - Platforms removed from supported list:
    - openSUSE 15.6 - End of Life: 2026-04-30
    - Debian 12 (Bookworm) - End of Life: 2026-06-10
    - Ubuntu 22.04 (Jammy Jellyfish) - End of life 2027-04-01

Since HTC25…

	Releases	LTS Releases	Feature Releases
number	30	15	15
bugfixes	140	104	36
features	174	13	161

Highlights on the web, full details in the Manual

Highlights:

https://htcondor.org/htcondor/release-highlights/

Details:

https://htcondor.readthedocs.io/en/latest/version-history/index.html

- Documented all the new features / mechanisms that have been added at each version
- Notes about "gotchas" when upgrading from major version X to Y
- condor_upgrade_check : tool to evaluate your current setup for incompatibilities before an HTCondor upgrade

On Tuesday, Christina said "understanding what is happening in a distributed system is hard…."

What is happening? What has happened?

- Job Event Log (e.g. log = file.log)
- ClassAds : key=value pairs
  - Job ClassAds
  - Slot ClassAds
  - Service (Daemon) ClassAds
- Where do I find all these lovely ClassAds live?
  1. Active Job Database (AP schedd) : condor_q, etc
  2. Archived Job Database (AP history files) : condor_history, etc
  3. HTCSS Central Manager and CE Collectors: *condor_status, condor_ce_status, etc*

1. Active Job Database

Jobs still active in the AP – “condor_q”
(could run now or in the future)
(still in core in the schedd)

Active Job Database

New -aaf flag: Append Auto Format

- We encourage custom job/slot attributes
- You can always show with –af
  - condor_q –af // condor_status –af
- If you want all usual + custom, -aaf

Example of the “-aaf” flag … and an Idle job in “Cooldown”

```
$ condor_q -nobatch -aaf MyProject
 ID     OWNER    SUBMITTED    RUN_TIME ST    CMD MyProject
772.0   someone   6/1   01:40 0+04:51:36 R   run ImportantProject
772.1   someone   6/1   01:40 0+01:21:33 Ic  run ImportantProject
772.2   someone   6/1   01:40 0+00:00:00 I   run ImportantProject
```

condor_watch_q improvements

- condor_watch_q improvements for viewing DAGMan jobs
- condor_watch_q now prioritizes displaying rows with jobs still active in the Access Point queue.
  - Options to have tool stop displaying rows of completed jobs after a specified amount of time
- Press any key to exit rather than just CTRL-C

htcondor noun-verb enhancements

htcondor joblist report (incorporating Kashika's work – see talk)

htcondor dag status

htcondor dag resources

htcondor dag histogram

htcondor dag status <dagid>

```
$ htcondor dag status 223
DAGMan Job 223.0 [simple.dag] has been running for 52 days 04:12:46.
DAG has submitted 382 individual job(s), of which:
        45 are running.
        10 are idle.
        0 are held.
        162 have completed successfully
DAG has failed nodes but will continue until all possible work is finished:
        5 nodes failed.
        10 nodes waiting to begin.
        24 nodes running.
[############========----------------] 34% complete.
```

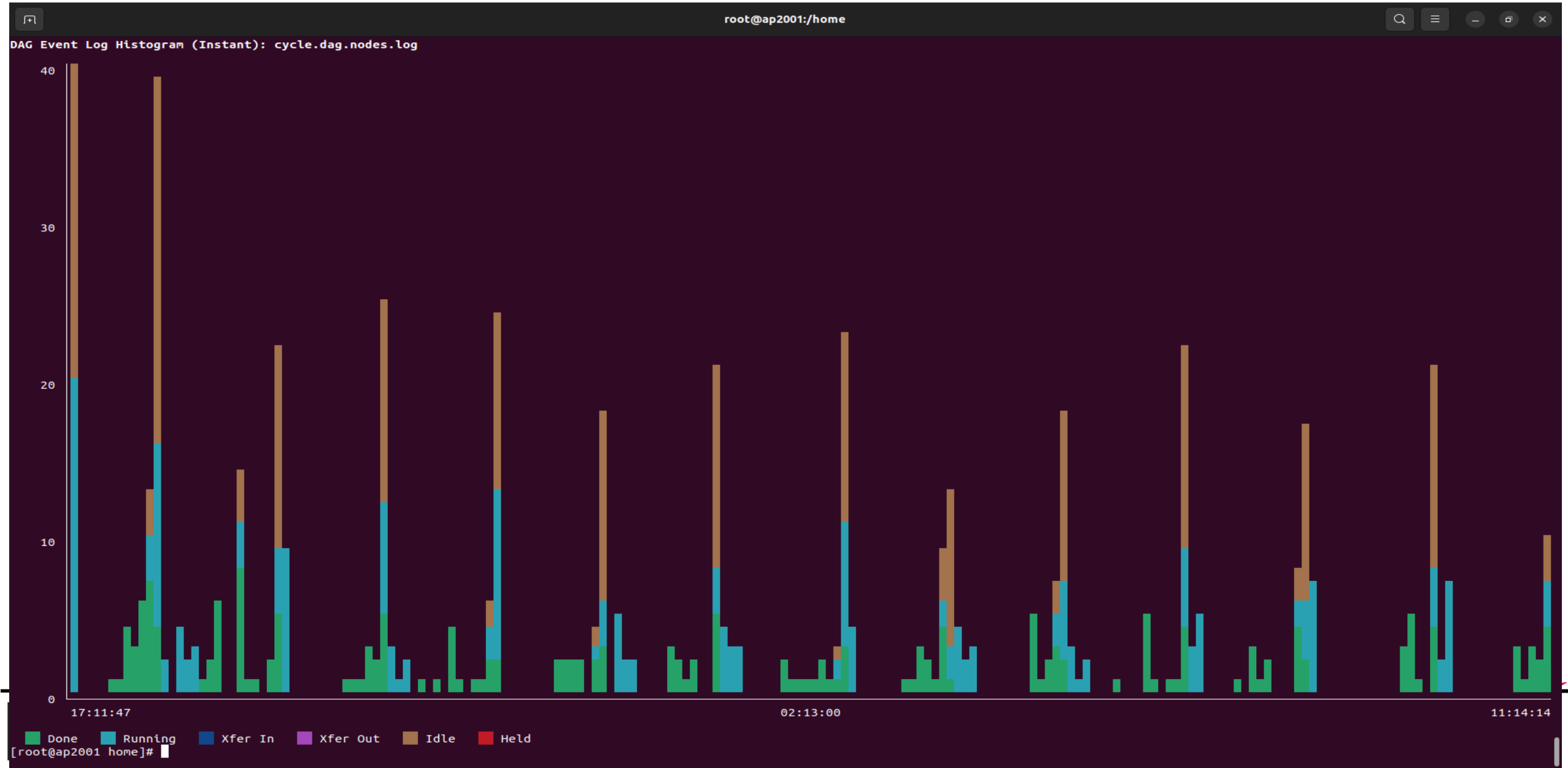
htcondor dag resources <dagid>

```
$ htcondor dag resource 1234.0
NODE_NAME              JOB_ID   STATUS  MEM_USAGE  REQ_MEM  STARTS  REMOTE_HOST
batch_D5AEL7_1         7172.0   Running      1465     4096       2  slot1_59@e4056.chtc.wisc.edu
batch_E6PBW7_1         7118.0   Running      1465     4096       3  slot1_66@e4075.chtc.wisc.edu

Total 2 jobs; 2 running, 0 idle; avg memory
utilization: 35.8%
```

htcondor dag histogram

htcondor dag histogram --instant



condor_rm -transfer

- condor_rm –transfer
  - Removes the job, sends it to the archive
  - But first transfers the sandbox
    - Not to spool, but to the final transfer spot
    - (including non-cedar protocols)

2. Archived Job Database

Snapshot at the AP of job classads for jobs that have left the Running state

Typically use “condor_history” to access

Archived Job Database

Accessing history is difficult because *condor_history* must parse flat files sequentially to find specific job data.

Issues:

- ***Very slow*** for old jobs
- Hard to make any aggregates
- Garbage collection: all info re old jobs is eventially removed
- Hurts effectiveness of other tools meant to improve user experience

Archive Librarian

- New daemon at the Access Point responsible for maintaining an index of archived job records (w/ SQLite).
- Allows tools that access the archive (e.g. *condor_history*) to quickly locate specific job records when given specific job ids rather than sequentially scanning flat archive files.
- Keeps some (small amount) job info around after garbage collection
- Easily enable by setting **use feature:librarian** in configuration of HTCSS Access Point

See Final Report by CHTC Fellow Sandhya Nayar
https://chtc.cs.wisc.edu/fellowships/reports/2025/sandhya-nayar.html

condor_adstash

- Replicates ads in an Archived Job Database to ElasticSearch / OpenSearch
- Improvements
  - New: Option to use a projection (i.e. allow-list) of attributes
  - Upcoming: Option to use an ignore (i.e. deny-) list of attributes
  - Upcoming: More robust index mappings and settings
    - Use existing index mappings' field properties to coerce data types
    - Only update mappings on new, known "CamelCase" fields
    - Options to supply your own custom field properties and dynamic templates
    - Continue populating existing, unknown "lowercase" keyword fields
    - Index unknown fields by default (e.g. machineattr fields)
    - Correctly map nested ClassAds (e.g. TransferInputStats, NumVacatesByReason, etc.)
    - Automatically increase the index field limit as needed (warn if > 5,000 fields)

3. HTCSS Central Manager Collector and/or CE Collectors

Live information about the pool or site
ClassAd for every slot, every daemon, every active user
Typical tool: condor_status

Q: How many slots are running a job?

A: Count slots where State == Claimed (and Activity != Idle)

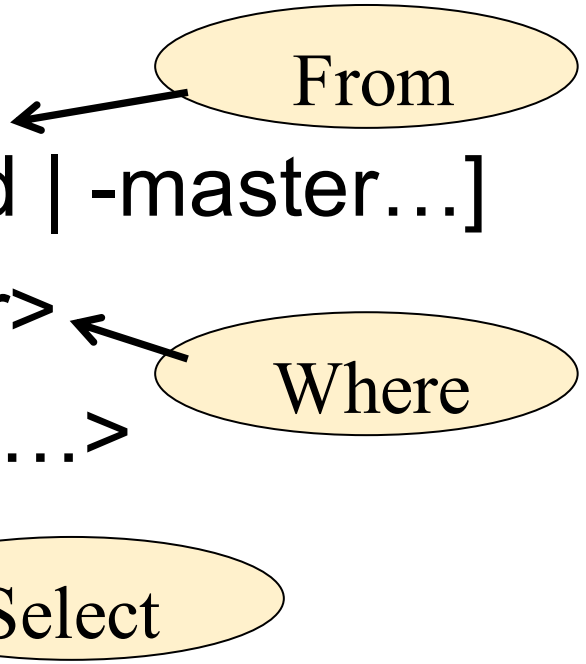
How?

Obvious solutions aren't the best

```
% condor_status
slot7@ale-22.cs.wi LINUX      X86_64 Claimed   Busy      0.990 3002  0+00:28:24
slot8@ale-22.cs.wi LINUX      X86_64 Claimed   Busy      1.000 3002  0+00:14:13
slot1@ale-23.cs.wi LINUX      X86_64 Unclaimed Idle      0.920 3002  0+00:00:04
…
```

- condor_status | grep Claimed | grep –v Idle | wc –l
  - Output subject to change, wrong answers, slow
- condor_status –l | grep Claimed | wc –l
  - Wrong answers, really slow

Use constraints and projections



- condor_status [-startd | -schedd | -master…]
  - –constraint <classad-expr>
  - –autoformat <attr1, attr2, …>

```
condor_status -startd     \
   -cons 'State=="Claimed" && Activity!="Idle"'   \
   -af name  | wc -l
```

Q: How many CPU cores are being utilized?

- Sum the Cpus attribute for each slot that is Claimed and Busy:

```
% condor_status -startd      \
   -cons 'State=="Claimed" && Activity!="Idle"'   \
   -af Cpus | less
1
1
4
4
…
% condor_status -startd      \
   -cons 'State=="Claimed" && Activity!="Idle"'   \
   -af Cpus | st
N        min      max      sum      mean     stddev
9053     1        40       10410    1.1499   1.39239
```

Simple Statistics from command line
https://github.com/nferraz/st

Graph of CPU utilization over time

- Could have a cron job run every minute…

```
#!/bin/sh
echo `date`, ; condor_status \
-cons 'State=="Claimed" && Activity!="Idle"'   \
-af Cpus | st --sum
```

- What if you have hundreds or thousands of metrics?
  - COLLECTOR_QUERY_WORKERS = 5000?
- How about query the collector just once per minute for all attributes needed to compute all metrics?

condor_gangliad

- condor_gangliad queries a (set of) condor_collector(s) once per minute
  - use feature:ganglia
- condor_gangliad has config file to *filter and aggregate* attributes from the ads in the condor_collector in order to form *metrics*
- Forwards these metrics to Ganglia, which stores these values in a database and provides graphs over the web

condor_metricd

- condor_metricd queries a (set of) condor_collector(s) once per minute
  - use feature:metricd
- condor_metricd has config file to *filter and aggregate* attributes from the ads in the condor_collector in order to form *metrics*
- Gives these metrics to Ganglia or Prometheus, which stores these values in a database and provides graphs over the web (should there be other back ends?)

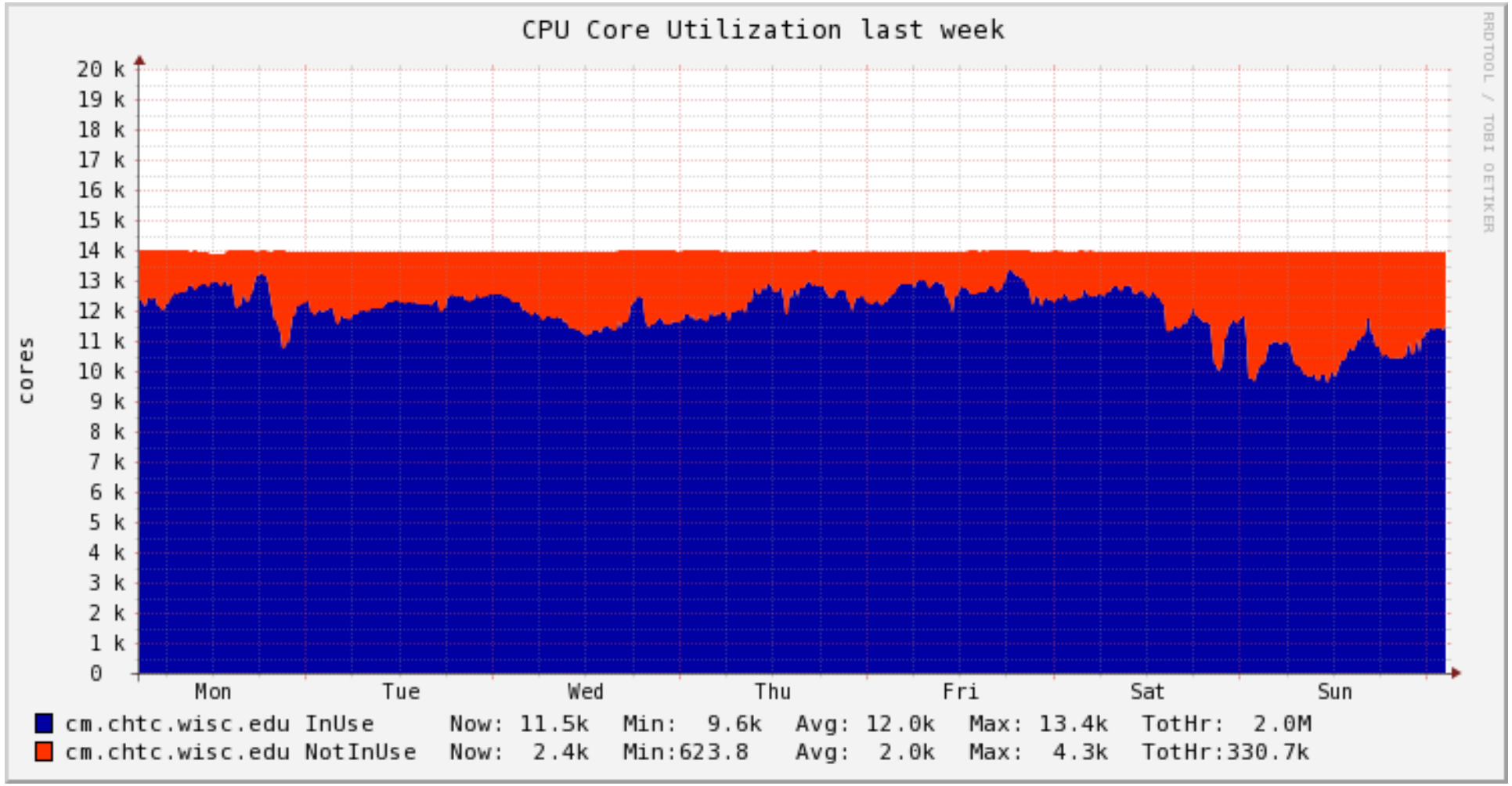
condor_metricd, cont

- Supports
  - Gauges
  - Counters (including aggregates of counters)
  - Labels
- Prometheus exporter on http port 9618 by default
  - Honors ALLOW_READ ip address limiting
  - http basic auth
  - Can specify a different port (and proxy it with NGINX etc)

Example metric definitions

```
[
  Name   = "CpusInUse";
  Aggregate = "SUM";
  Value  = Cpus;
  Requirements = State=="Claimed" && Activity!="Idle";
  TargetType = "Machine";
]
[
  Name   = "CpusNotInUse";
  Aggregate = "SUM";
  Value  = Cpus;
  Requirements = State!="Claimed" || Activity=="Idle";
  TargetType = "Machine";
]
```

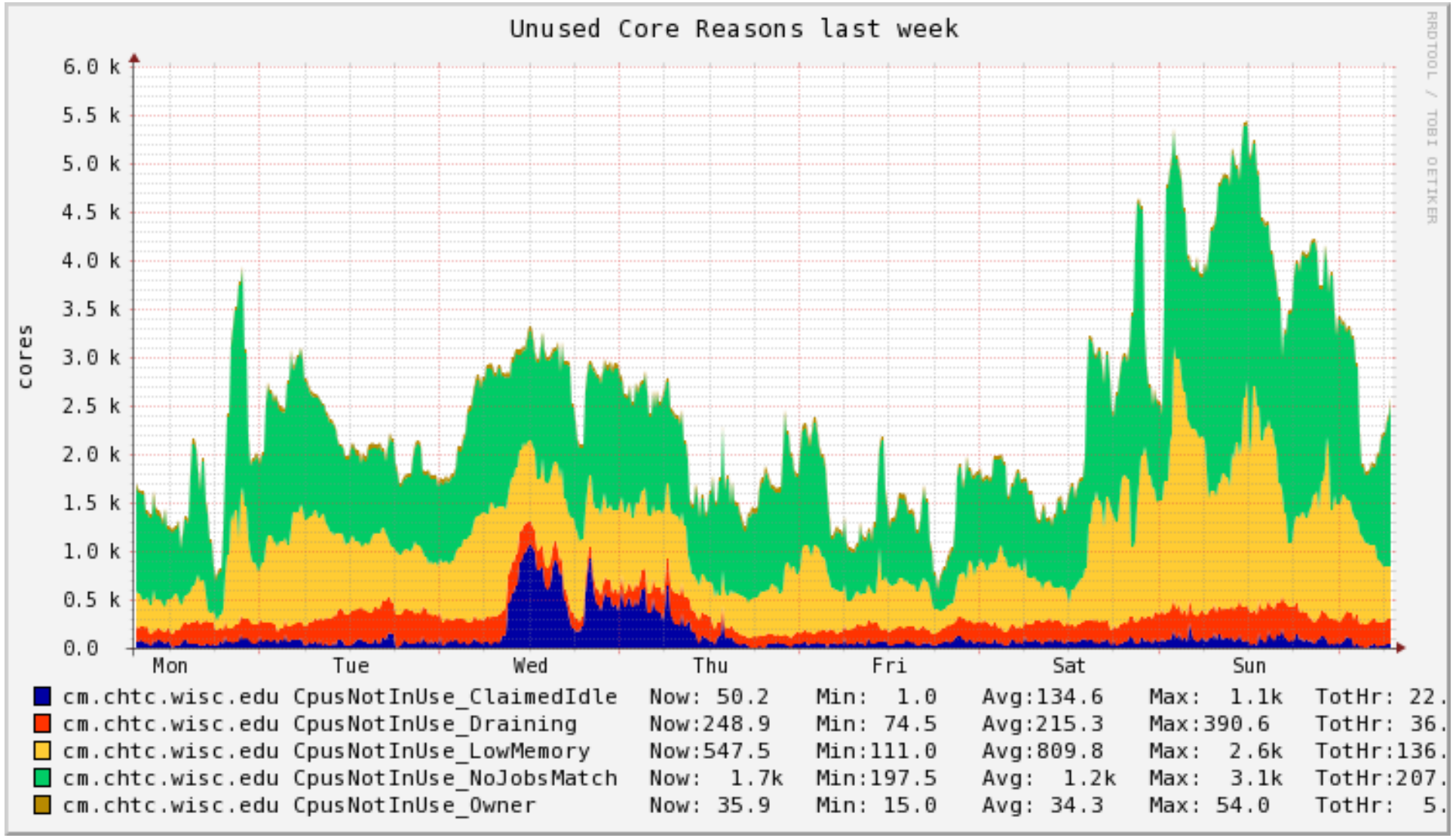
Voila!


CPU Core Utilization last week
RRDTOOL / TOBI OETIKER
cores
20 k
19 k
18 k
17 k
16 k
15 k
14 k
13 k
12 k
11 k
10 k
9 k
8 k
7 k
6 k
5 k
4 k
3 k
2 k
1 k
0
Mon
Tue
Wed
Thu
Fri
Sat
Sun
cm.chtc.wisc.edu InUse Now: 11.5k Min: 9.6k Avg: 12.0k Max: 13.4k TotHr: 2.0M
cm.chtc.wisc.edu NotInUse Now: 2.4k Min:623.8 Avg: 2.0k Max: 4.3k TotHr:330.7k

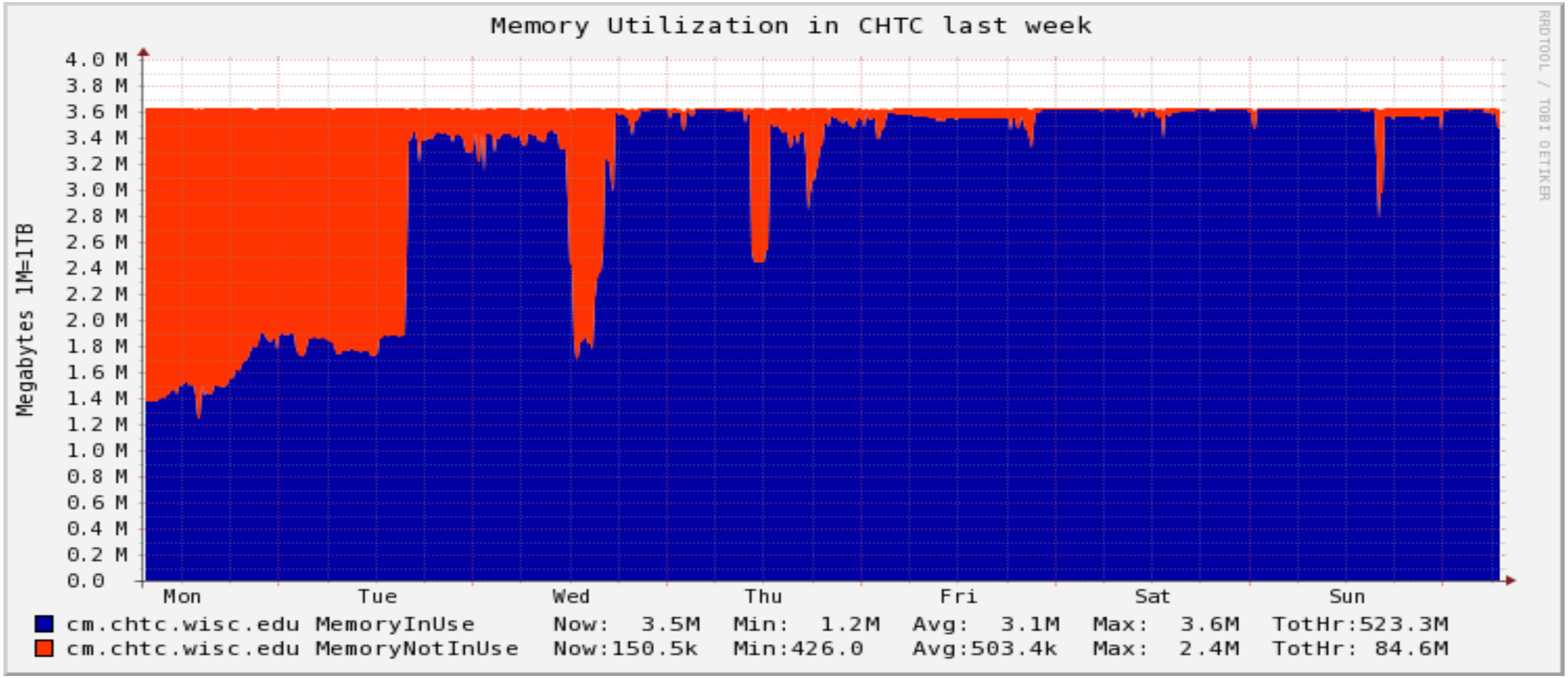
Why are cores not in use?

```
[
  Name   = "CpusNotInUse_LowMemory";
  Aggregate = "SUM";
  Value  = Cpus;
  Requirements = State=="Unclaimed" && Memory < 1024;
  TargetType = "Machine";
]
[
  Name   = "CpusNotInUse_Draining";
  Aggregate = "SUM";
  Value  = Cpus;
  Requirements = State=="Drained";
  TargetType = "Machine";
]
```

Unused Core Reasons



Memory Provisioned



Memory Used vs Provisioned

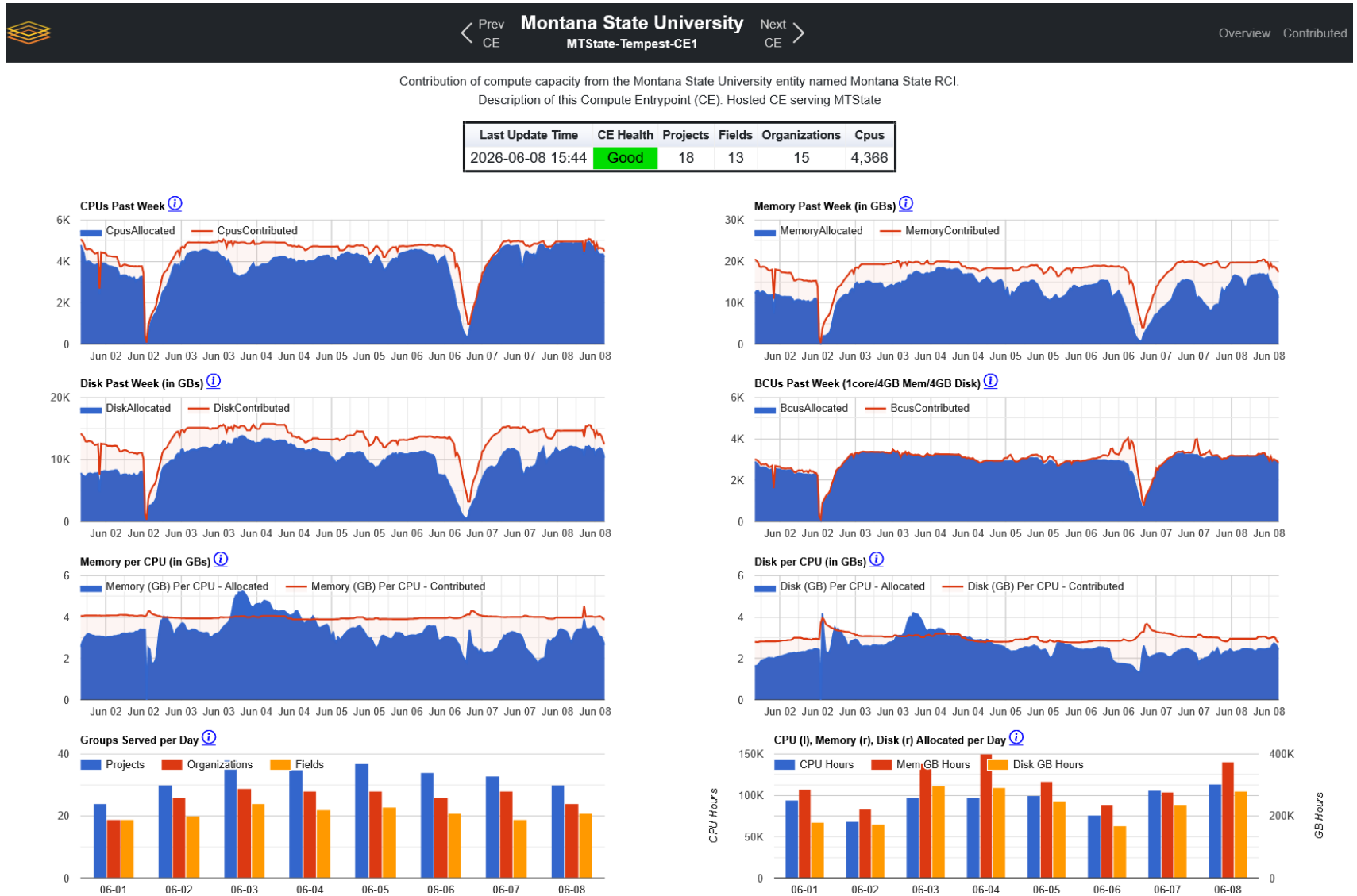
Define MemoryEfficiency metric as:

```
[
  Name   = "MemoryEfficiency";
  Aggregate = "AVG";
  Value  = real(MemoryUsage)/Memory*100;
  Requirements = MemoryUsage > 0.0;
  TargetType = "Machine";
]
```

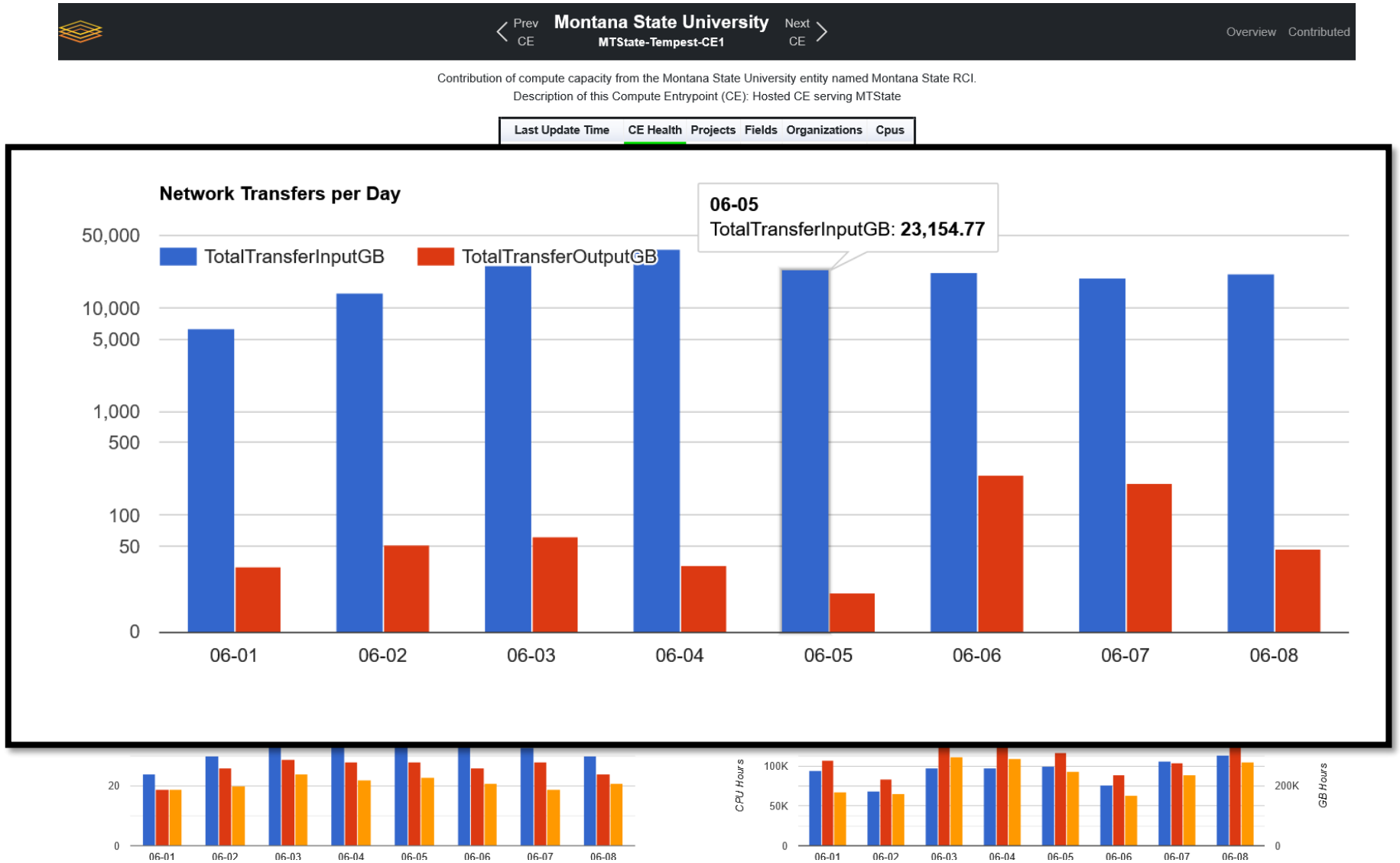
Example: Metrics Per User

```
[
  Name = “UserMemoryEfficiency”
  Labels = {“user”,RemoteUser};
  Ganglia_Name   = strcat(RemoteUser,"-UserMemoryEfficiency");
  Title  = strcat(RemoteUser," Memory Efficiency");
  Aggregate = "AVG";
  Value  = real(MemoryUsage)/Memory*100;
  Requirements = MemoryUsage > 0.0;
  TargetType = "Machine";
]
```

CE Dashboard



CE Dashboard



floor and ceiling leasing

```
$ condor_userprio -setceiling todd  12 \
      -duration 3600
$ condor_userprio -cancelsetceiling todd

$ condor_userprio -setfactor todd 3 \
      -duration 3600
$ condor_userprio -cancelsetfactor todd
```

Owned Capacity Units (OCU) preview

- OCU is an "owned" slot with a defined set of resources (e.g. 1 GPU, 8 CPUs, 32GB Memory, …)
  - Owner can always quickly start using the slot
  - Non-owner can opt in, but will get preempted if owner returns
  - If EP hosting the slot disappears, AP works to replace it
  - Useful for GPUs, etc.

HTCondor Annex Changes: Glidein an EP to any Slurm Cluster

- "htcondor annex create" used to present a menu of NSF HPC Sites
  - NCSA, SDSC, Purdue, …
- Now "htcondor annex create" gives you a tarball file
  - Transfer this file to your Slurm login node
  - Untar it – inside is a README: run a tool, sbatch submit, profit!
    - Jobs placed on the AP tagged with the name of the Annex will run
  - All logs available to the user
- Why the change?

Gliding into DOE sites
Grid Universe Enhancements

- Support for NERSC SuperFacility REST API
  - remote interface to Perlmutter HPC
  - Initial code contributed by Karan Vahi (Pegasus)
- Support for Flux scheduler
  - Used at Lawrence Livermore National Laboratory
  - Initial code contributed by Ian Lumsden

```
# Flux support was added as part of work supported by the US National Science Foundation
# under Grant No. 2530461, 2513101, 2331152, 2223704, 2138811, and 2103845.
#
# Flux support was also added under the auspices of the U.S. Department of Energy by
# Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344
# and was supported by the LLNL-LDRD Program under Project No. 24-SI-005.
```

- Support for IRI (Integrated Research Infra) REST API (upcoming)

U.S. DEPARTMENT of ENERGY | Office of Science

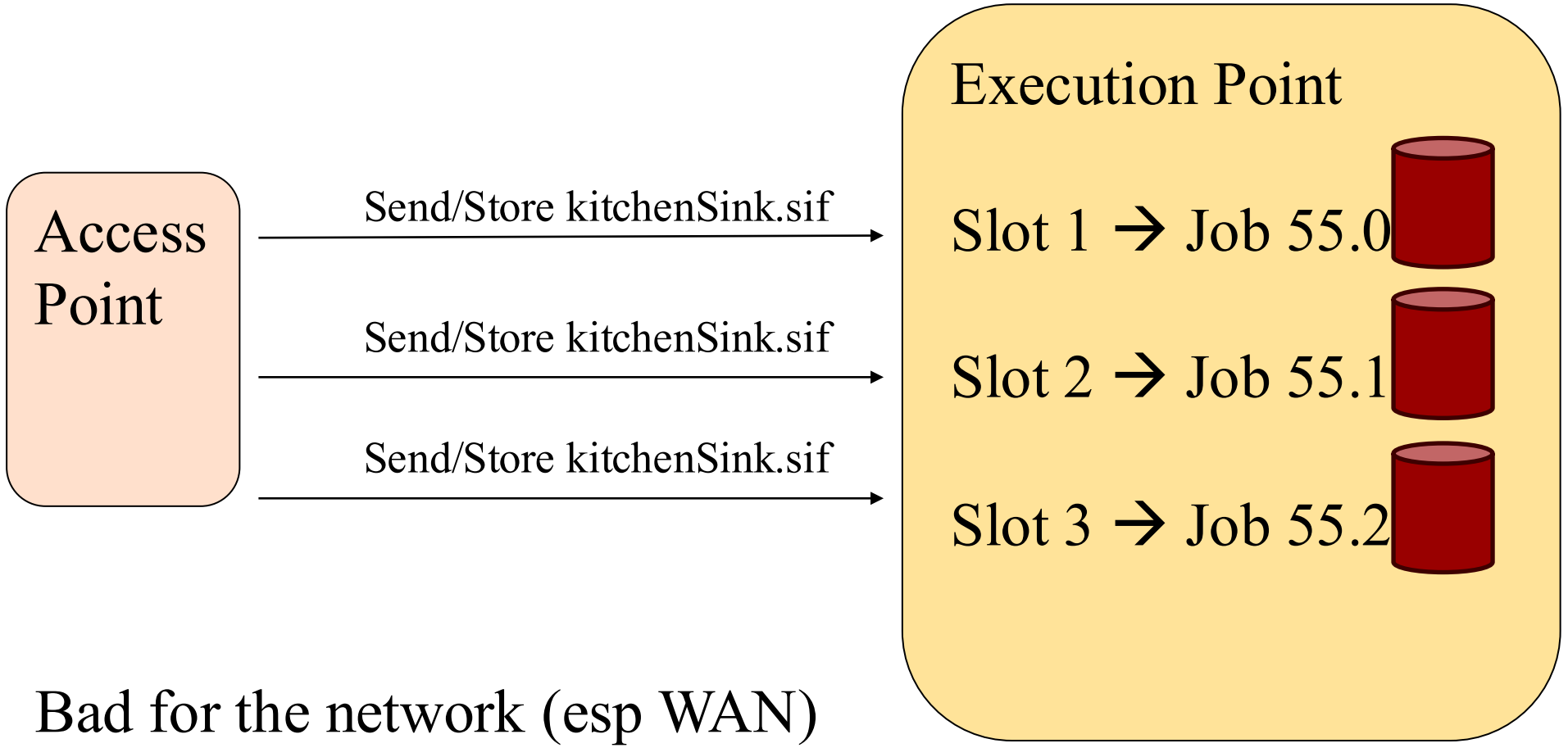
flux

Common Input Data

- Amount of job input is growing
  - Container Images (.SIF files)
  - AI/ML Data
- Often many of the same files occur across all jobs
  - … in a Job List (cluster)
  - … in a DAG workflow

```
executable = myjob.exe
container_image = kitchenSink.sif
output = out.$(Process)
queue 1000
```

Moving Input Today



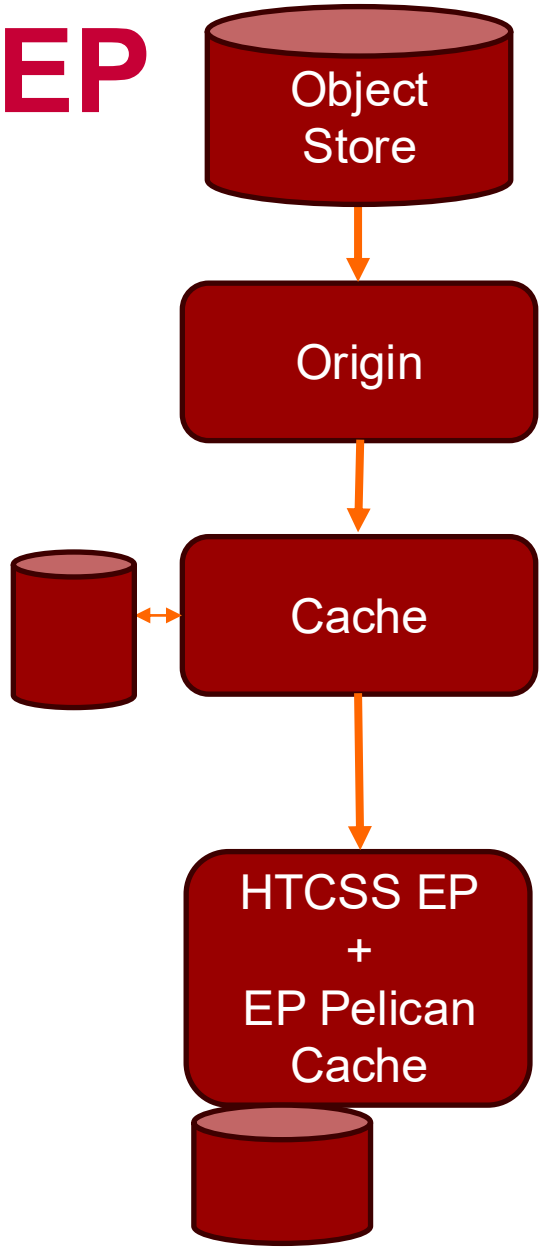
- Bad for the network (esp WAN)
- Bad if EP has more compute than storage

Pelican Local Cache on the EP

- Cache files coming from OSDF (Pelican) on the EP

```
use feature:PelicanCache(size of cache)
```

Works with a glidein EP as well. *Voila!!*

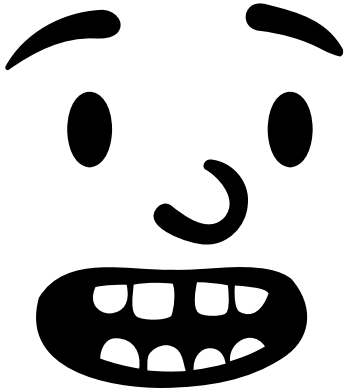


Pelican Local Cache on the EP

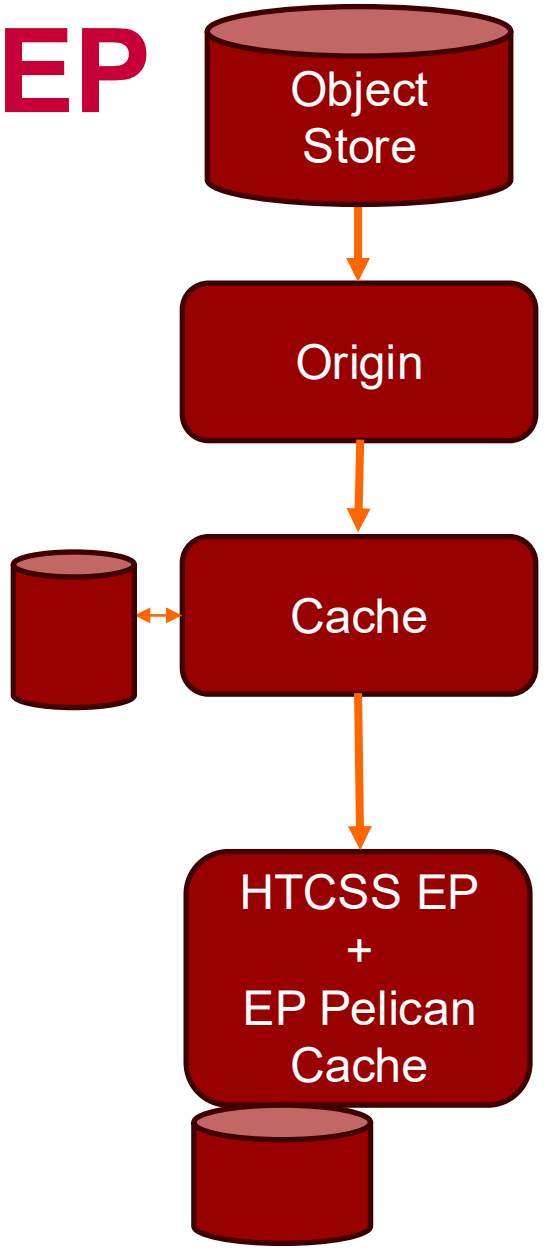
- Cache files coming from OSDF (Pelican) on the EP

  `use feature:PelicanCache(size of cache)`

  Works with a glidein EP as well. *Voila!!*



- What's the Bad News?
  - Admin must fragment EP Disk Space
  - What about data transferred without Pelican?
    - Files from the AP (HTCondor File Transfer / CEDAR)
    - Files from Web Server URLs / HTTP
  - Data Stored twice! Need double the EP disk space
    - One copy in EP Pelican Cache, then copied to job slot scratch dir
  - AP knowledge not captured



HTCSS Support for Common Files

Shares common files between jobs in the same job list (cluster) and/or jobs in the same DAG.

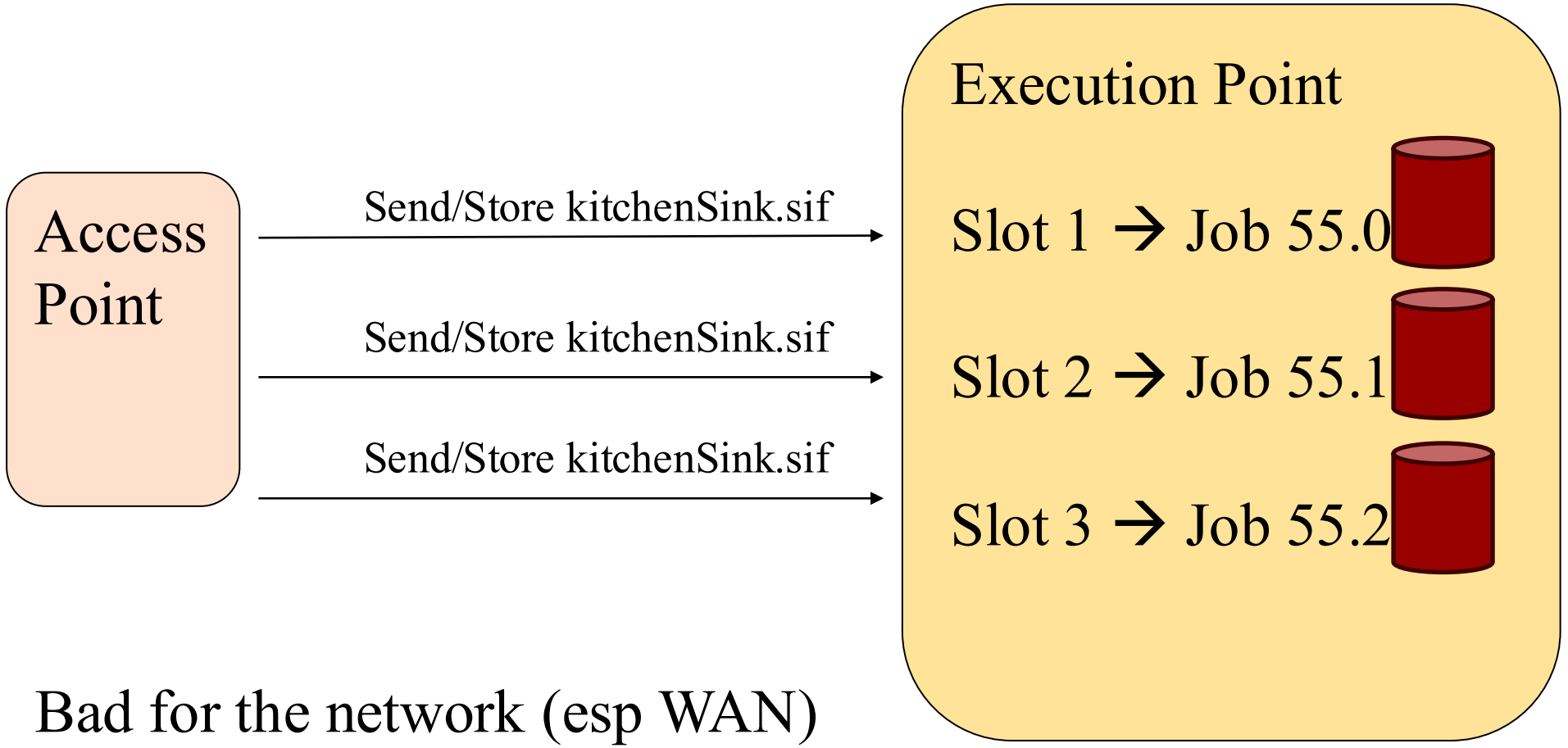
- Transfers them only once regardless of protocol
- Makes only one copy on-disk (if possible…)
- Controlled by the AP.

› What does the user do? In the JDL:

- transfer_common_files = xxx, yyy, …
- container_is_common = True | False

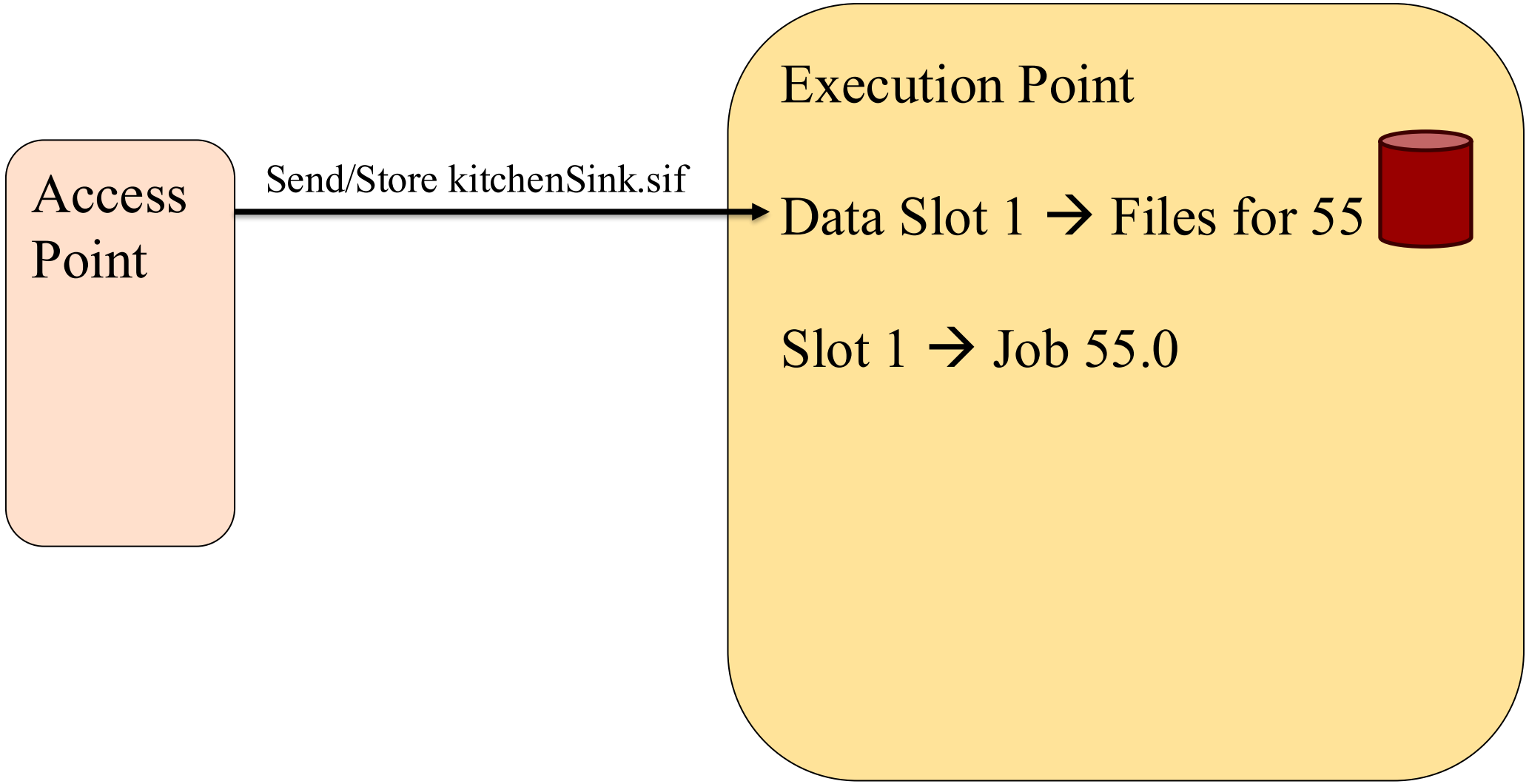
› What does HTCSS do?

Moving Input Today

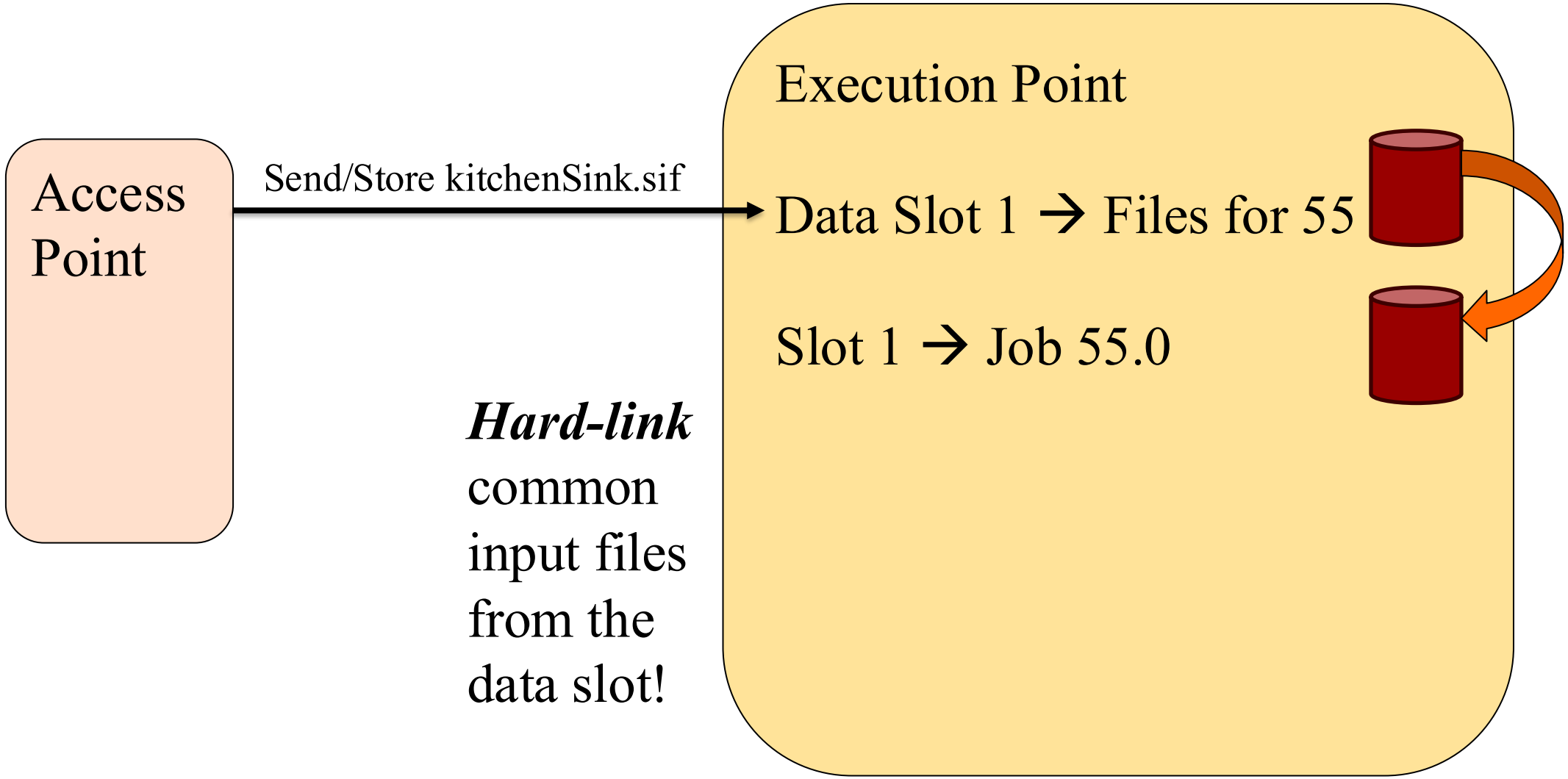


- Bad for the network (esp WAN)
- Bad if EP has more compute than storage

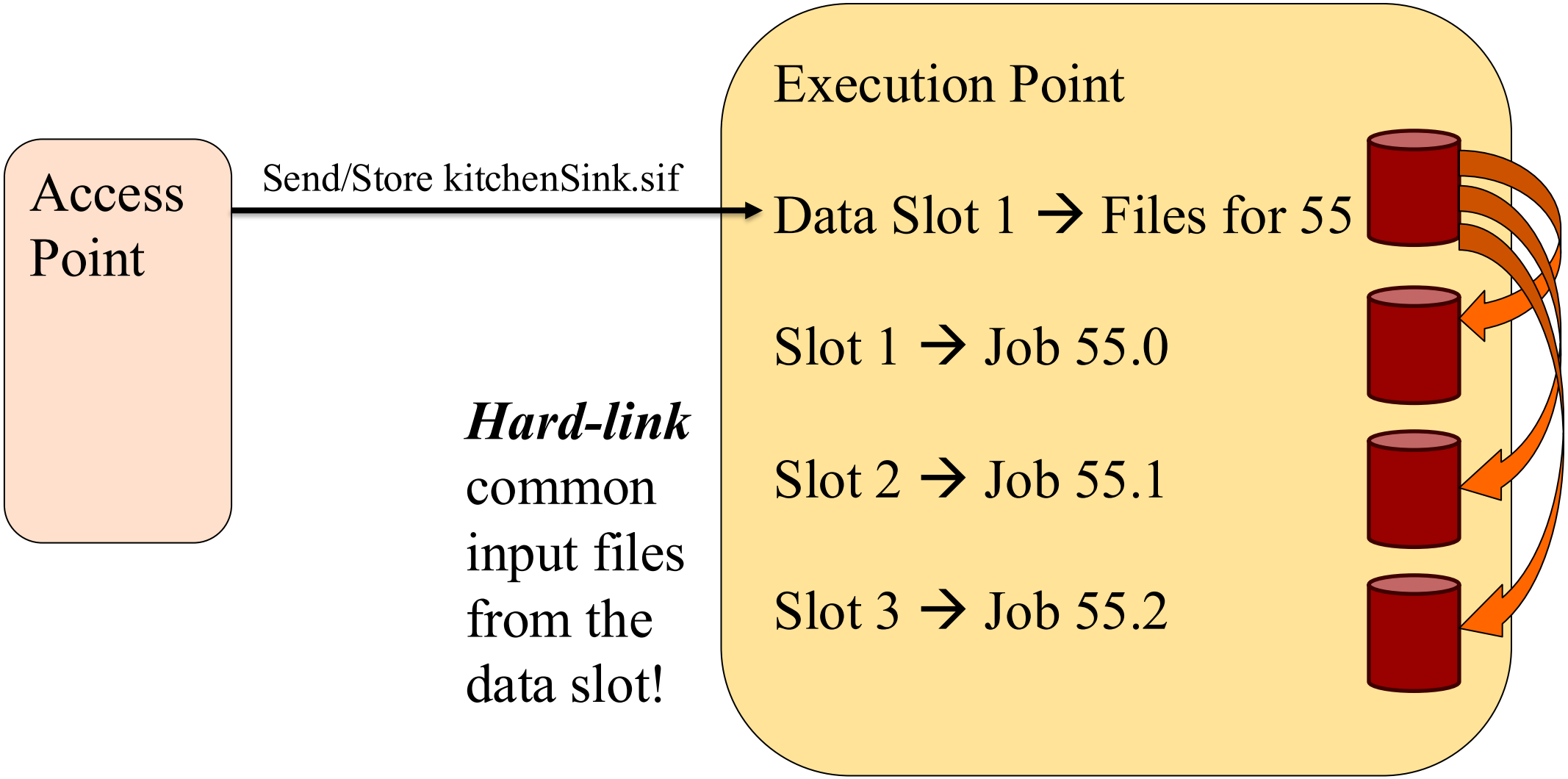
HTCSS Support for Common Files



HTCSS Support for Common Files



HTCSS Support for Common Files



Dealing with Over Provisioning of Resources

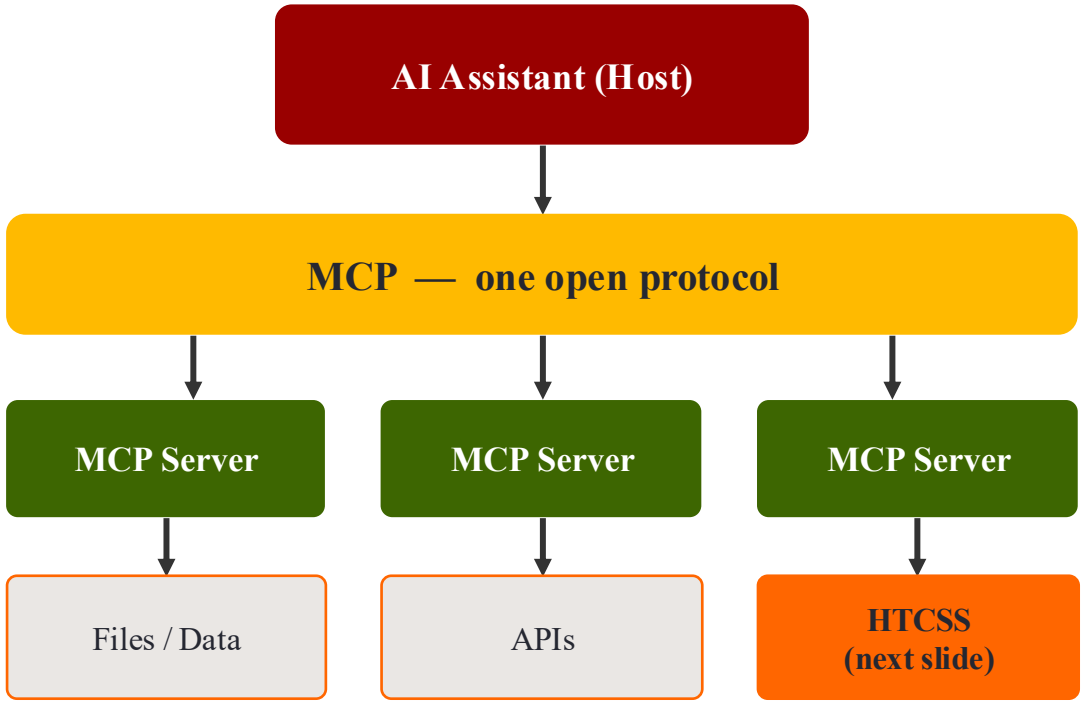
- New Retry Request feature – now for both *disk* and *memory*.
- Try the first, if resource exceeded, re-run with the next value.

```
request_memory = 1GB
retry_request_memory = 4GB
request_disk = 1GB
retry_request_disk = 10GB,40GB
```

What is MCP (Model Context Protocol)?

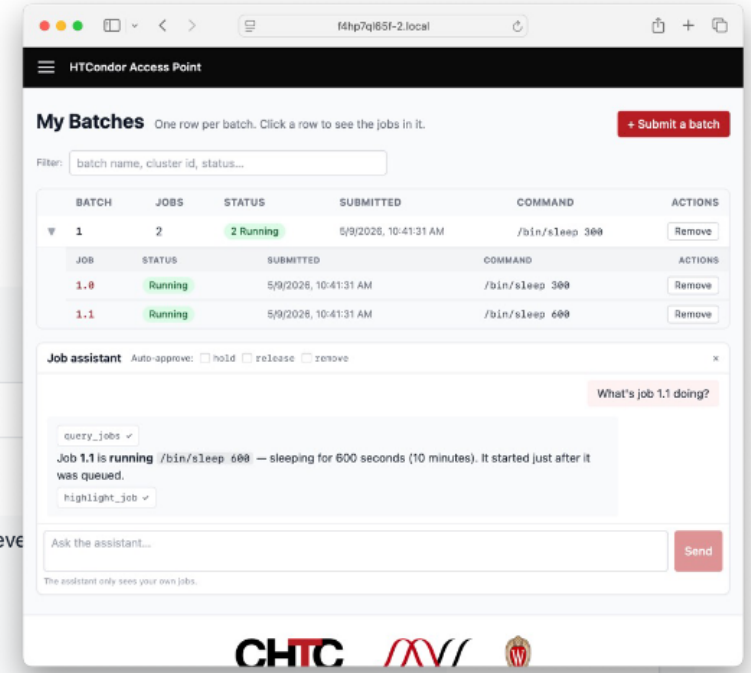
An open standard that lets AI assistants talk to your tools and data.

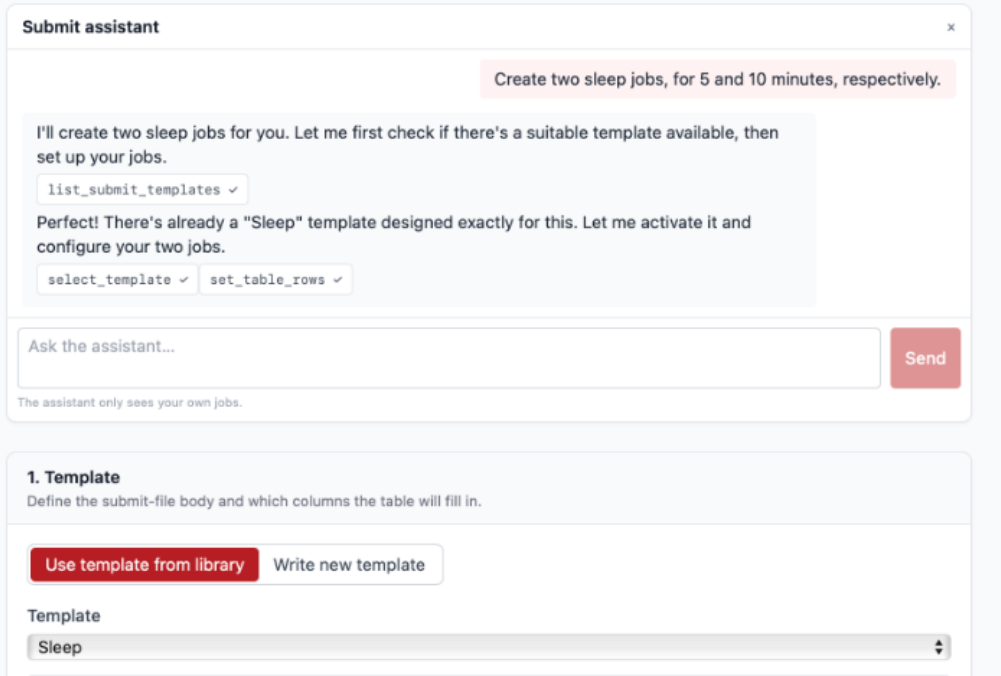
- Think of it as a **“USB-C port for AI”** — one connector, many systems.
- The AI app (the **host**) connects through MCP to lightweight **servers**.
- Each server exposes **tools, data, and prompts** in a common format.
- Build a server once — any MCP-aware assistant can use it.

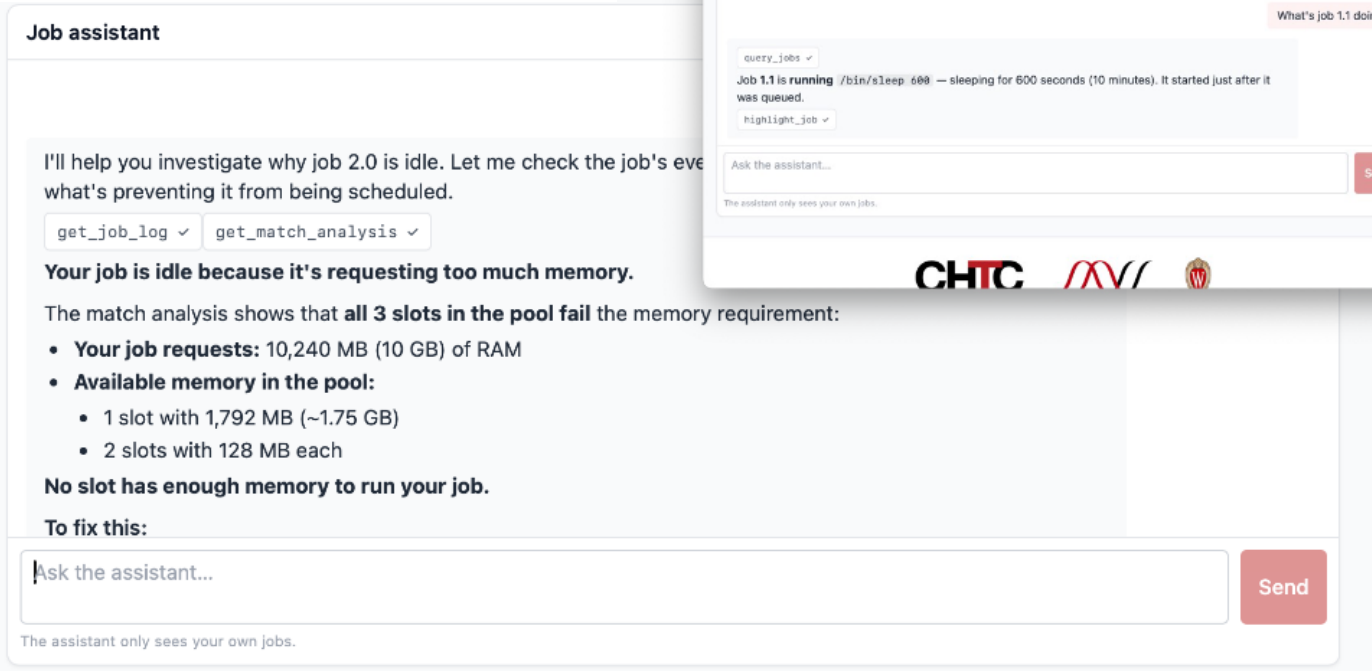


Exploring HTCSS MCP

- Easy to embed an agent into a web page
  - … but UX challenges to go beyond AI “slop”
- See Brian B’s talk from yesterday
- See Girodon S’s talk from yesterday
- See Ilija V’s talk tomorrow







Open OnDemand

- Investigate (and improve?) Open OnDemand integration with HTCondor AP

Reproducible Builds

- Helps against supply chain attacks (e.g. xz)
- Required by Debian (encouraged by EL)
- You *should* be able to make bitwise identical binaries as we ship

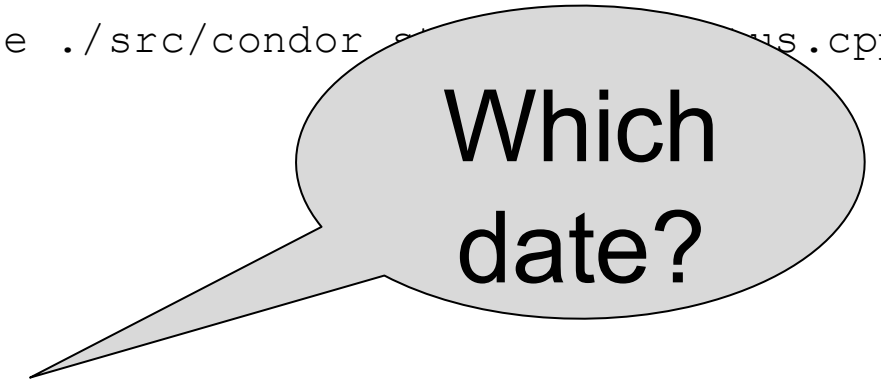
Reproducible builds are hard

- FILE macro in fatal exceptions:

```
ERROR "This should never happen" at line 2624 in file
/home/todd/CONDOR_SRC/src/condor_status.V6/status.cpp
```

- ERROR "This should never happen" at line 2624 in file ./src/condor_[illegible]us.cpp

- Build Date in condor_version
  - $ condor_version
    - $CondorVersion: 27.8.1 **2028-03-31**



- Other headaches

https://reproducible-builds.org/

Container improvements

Local universe uses cgroups by default

- Optionally enforces if Request_Memory set

condor_ssh_to_job works in more places

- with usernamespaces and non-root condor
- Can't work work non-root condor and setuid apptainer

Container universe now supports AMD HIP/ROCM

condor_dag_checker (teased last year)

- Lint DAG files before execution for validity
  - Correct command syntax
  - Check for parse errors
  - Check for references to undefined nodes
  - Check for valid node dependencies
  - Check for cycles in DAG structure
  - Check for infinite recursive DAG file inclusion
  - [Optional] Check valid JDL specified (best effort similar to condor_submit –dry-run)
  - [Optional] Verify Pre/Post scripts exist
- Get useful DAG statistics

Thank You!

PATh PARTNERSHIP to ADVANCE THROUGHPUT COMPUTING

This work is supported by NSF under Cooperative Agreement OAC-2030508 as part of the PATh Project. 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 NSF.