Brookhaven National Laboratory

U.S. DEPARTMENT of ENERGY

HTCondor operational improvement efforts @ BNL

Tom Smith, BNL Scientific Computing and Data Facility (SCDF)

2026 June 12, HTC26

@BrookhavenLab

Brief Overview of our HTC system

(BNL Site Report in 1 slide)

	USATLAS Tier 1 pool	sPHENIX Tier 0 pool	Shared pool
Worker nodes (EPs)	~320	~1200	~490
Logical cores	32k	133k	41k
HEPscore23	~460k	~2.05M	~516k
Condor CEs	4	0	4
Submit nodes (APs)	0	12	35

(~600 PB of physics data)

ATLAS EXPERIMENT

sPHENIX

STAR

Belle II

DUNE DEEP UNDERGROUND NEUTRINO EXPERIMENT

PHENIX

…and more!

It was a busy year..

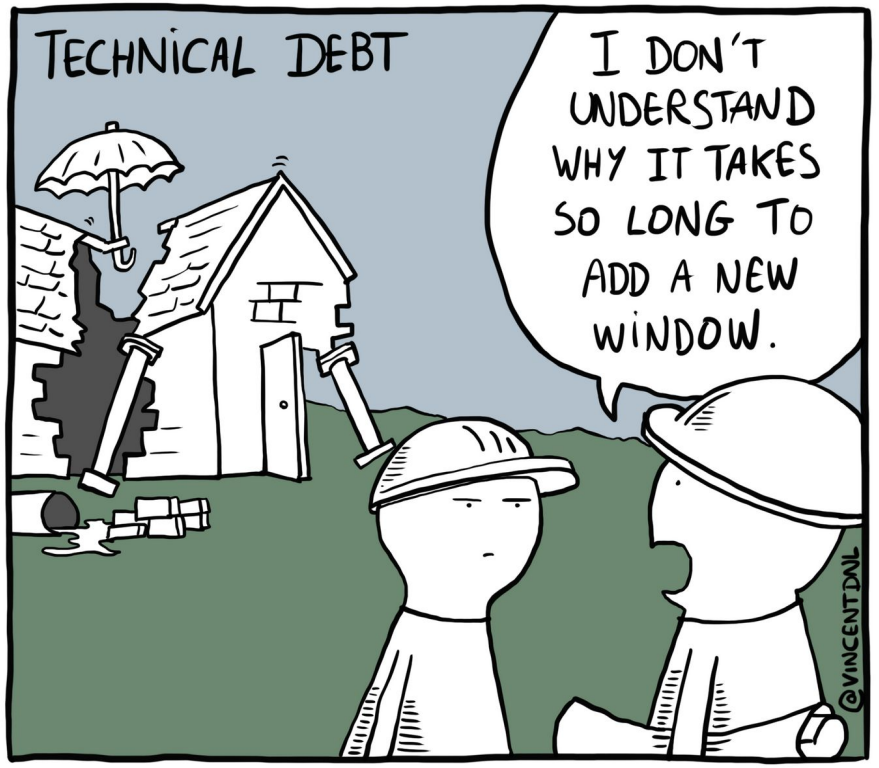
- HTCondor major version upgrades 23.0 -> 24.0 -> 25.0
- el7 to Alma Linux 9 migrations (we are **finally** done!)
- Major refactor of HTCondor related configurations and puppet infrastructure code

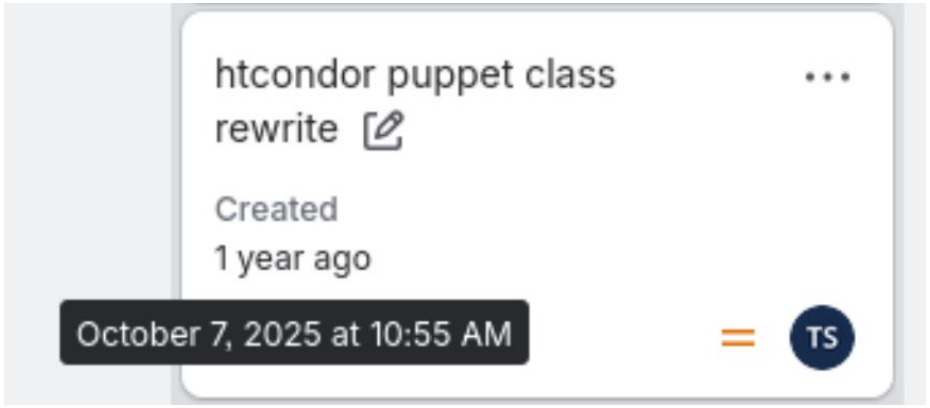
We live in a world of finite resources…

- **Admin time**
  - tackling technical debt
  - better tools
  - useful automation
- **CPU time**
  - less waste, more throughput (number go UP)
- **Memory**
  - quantify inefficiencies, start to address them
  - has anyone else noticed this stuff is expensive?

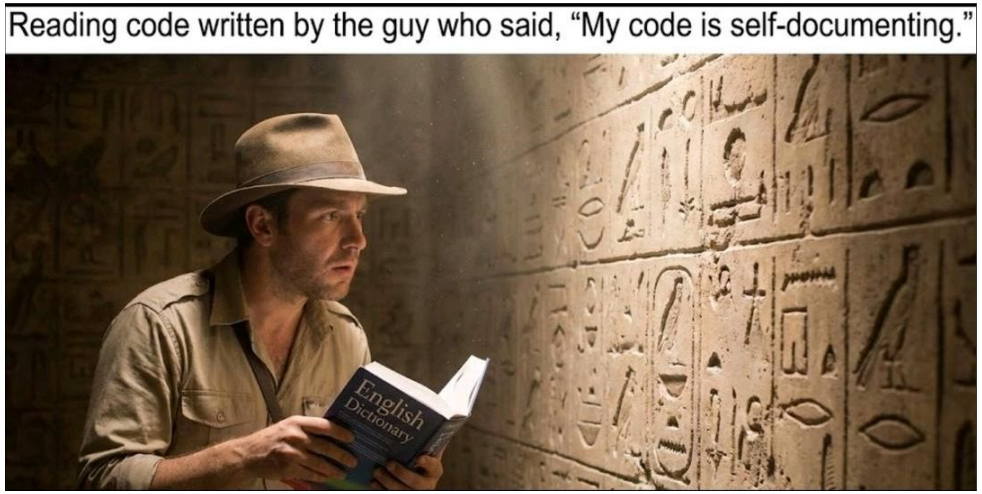
Technical debt costs time

- **Accumulation of technical debt over the course of many years is substantial**
  - Continuous uptime creates a focus on “Just fix it enough to keep on running”
  - Culture of Maintenance over Development
  - Adding any feature to the existing codebase was non-trivial
  - Deprecated features and workarounds everywhere
- **No time like the present**
  - el7 is gone, alma 9 is here
  - Powerful tools at our disposal (Foreman, Jenkins, various monitoring)
  - Need to adapt to a culture of rapid change





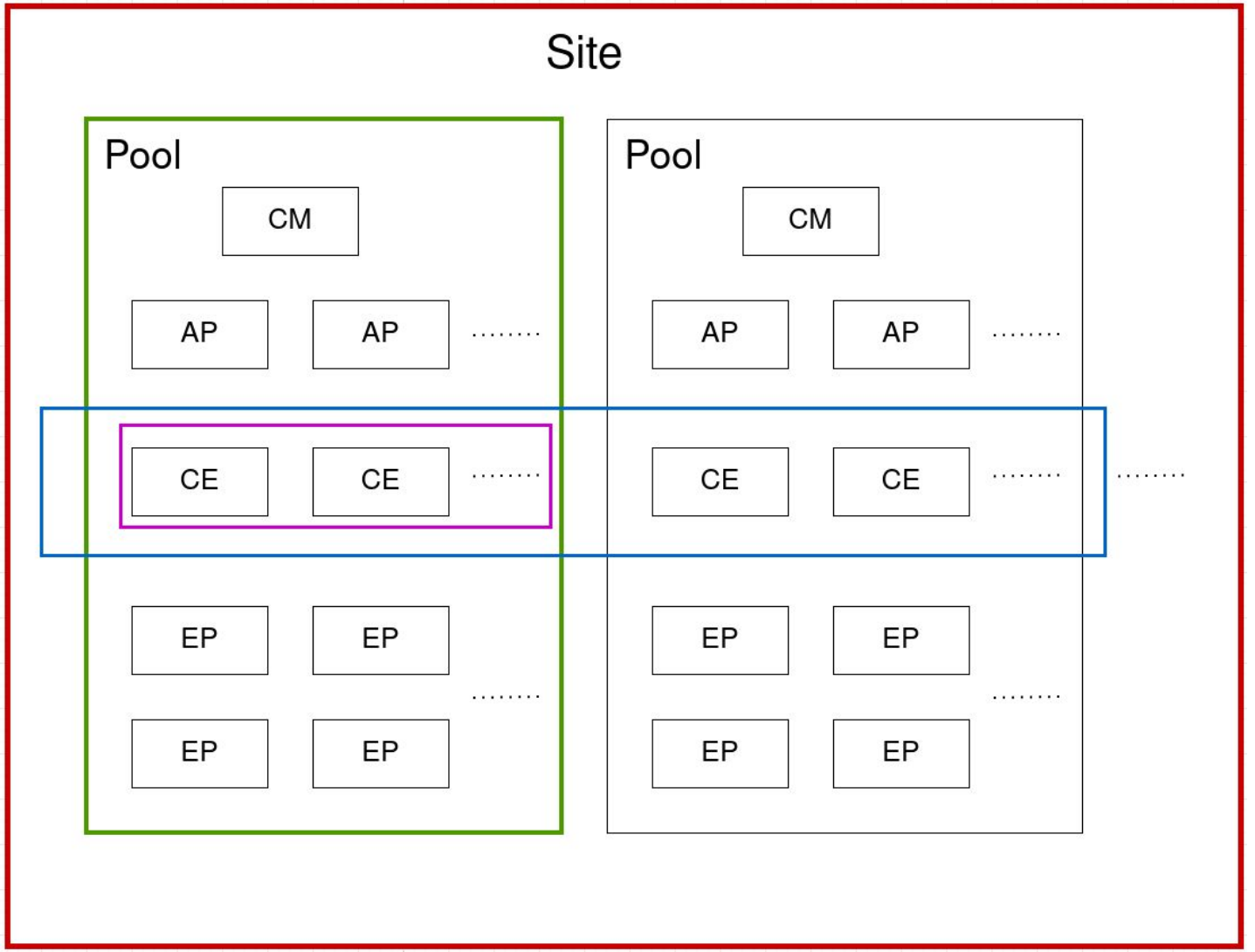
HTCondor config refactor



- All configs gone through line by line
  - some lines were moved to different level
  - deprecated things were removed
  - some (very!) old code existed to provide things that are 1st class features nowadays
  - lines that match the default value were removed
  - values that didn’t match the default were scrutinized
    - comments added to explain values or complex logic
  - # comments added in general #
  - old syntax transforms were transformed <- necessary for HTCondor 26+
- Total size of the configs (by lines) reduced by more than half, including new comment lines added!

HTCondor config refactor

- Condor configs are read in lexicographical order
- We have a hierarchy of files, overriding values where applicable:
  - condor_config (site)
  - config.d/00_pool (pool)
  - config.d/08_common_ce (CE)
  - config.d/10_local_ce (pool CE)
- Configure things at highest level of commonality
  - anything will be exactly where you expect it to be!



Operational improvements

- htctl (HTCondor control)
  - *loosely* inspired by `htcondor noun verb`
  - terminal focused wrapper program
  - written in python
  - provides quality of life / additional functionality for common condor operations
    - context aware (where you are)
    - pre-flight checks for EP starting
    - can disable an EP with message
  - some advanced features not present in normal condor
    - AP backup / restore - rebuild an AP with a running queue! (some restrictions apply)
    - alternative draining options

```
--------- htctl help ----------------------------------------------------------
  htctl is a wrapper program for common condor actions performed on the linux farm

  Subsystem is defined in /etc/htctl.conf, or assumed ep if file not present

  Subsystems: cm = Central Manager,            ce = Compute Entrypoint (gatekeeper)
              ap = Access Point (submit node), ep = Execution Point (worker)

  Usage: htctl <action> <optional sub-action>
  Note - not all actions have sub-actions

--------- Actions for all condor subsystems -----------------------------------
  start        - Starts HTCondor daemons
  stop         - Stops HTCondor daemons
  restart      - Restarts HTCondor daemons
  status       - Shows status of HTCondor daemons

--------- Actions for EP (worker) only ----------------------------------------
  enable       - Enables EP to accept new jobs
  disable <message>
               - Disables EP from accepting jobs with optional message in /NOCONDOR
  fullstart    - On an EP, same as running "htctl start" followed by "htctl enable"
  fs           - Same as "fullstart"
  drain        - Drains EP gracefully and stops daemons (respects MaxJobRetirementTime)
  vacatedrain <hours>
               - Drains EP gracefully and vacates jobs started within last <hours>
  fastdrain    - Drains EP fast and stops daemons (ignores MaxJobRetirementTime)
  canceldrain  - Cancels the drain of an EP and resumes operation

--------- Actions for AP (submit / schedd) only -------------------------------
  backup       - Copies log and spool directories for later restoration after rebuild
  restore      - Retreives aforementioned backups and puts them back
  clear_backup - Clears any backups from the NFS backup directory for this host

-------------------------------------------------------------------------------
```

Condor Wacky draining™

Vacate drain

Observation: In a steady state pool, young jobs are more likely to make a drain take longer

Implementation: perform a graceful drain, but immediately vacate jobs with less than X amount of runtime to go run somewhere else

Good enough drain

Observation: most jobs drain pretty fast- O(hours), but few jobs seem to stick around forever- O(days)

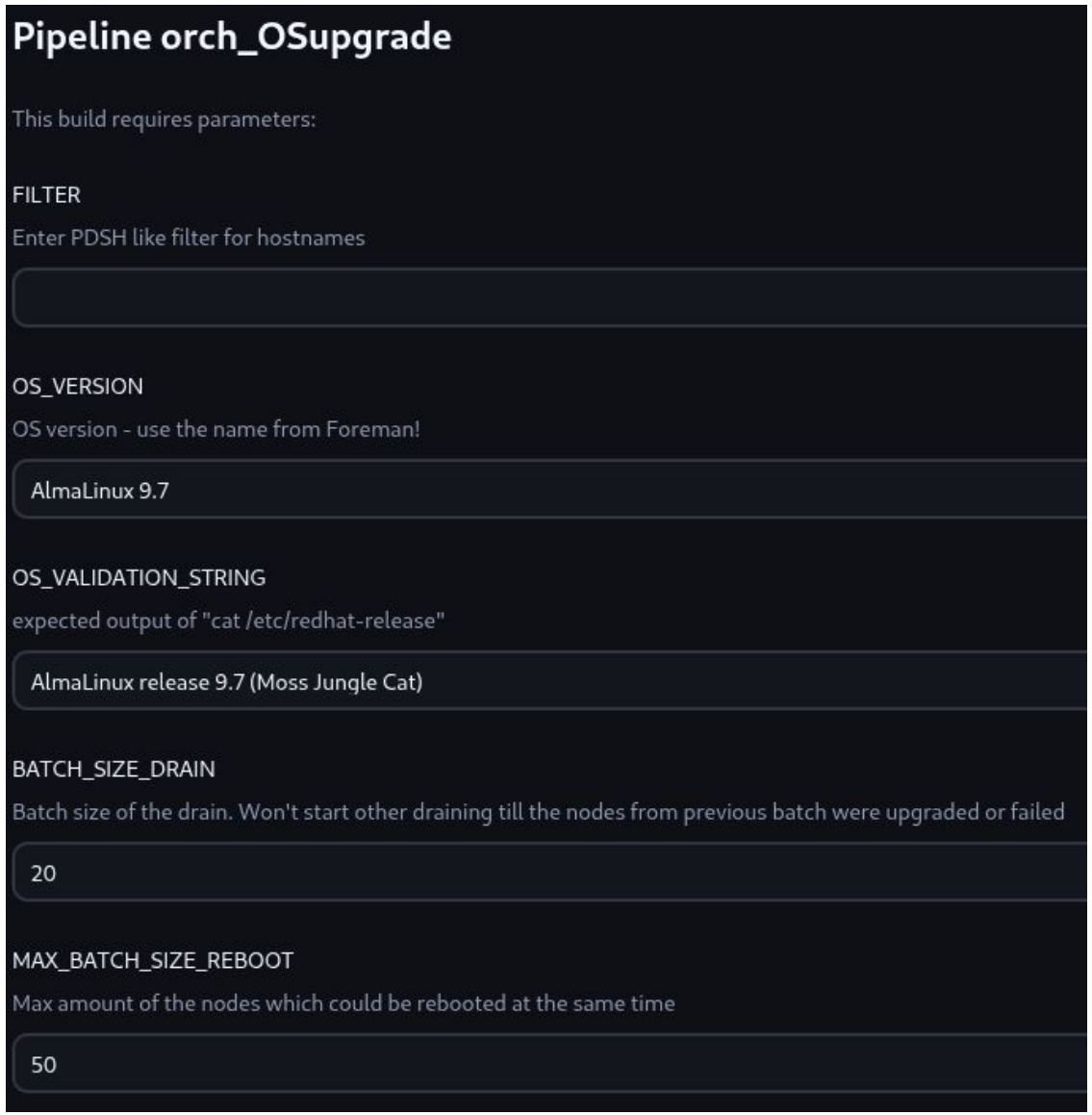
Implementation: perform a graceful drain, but declare a certain percentage “good enough” and kill the rest

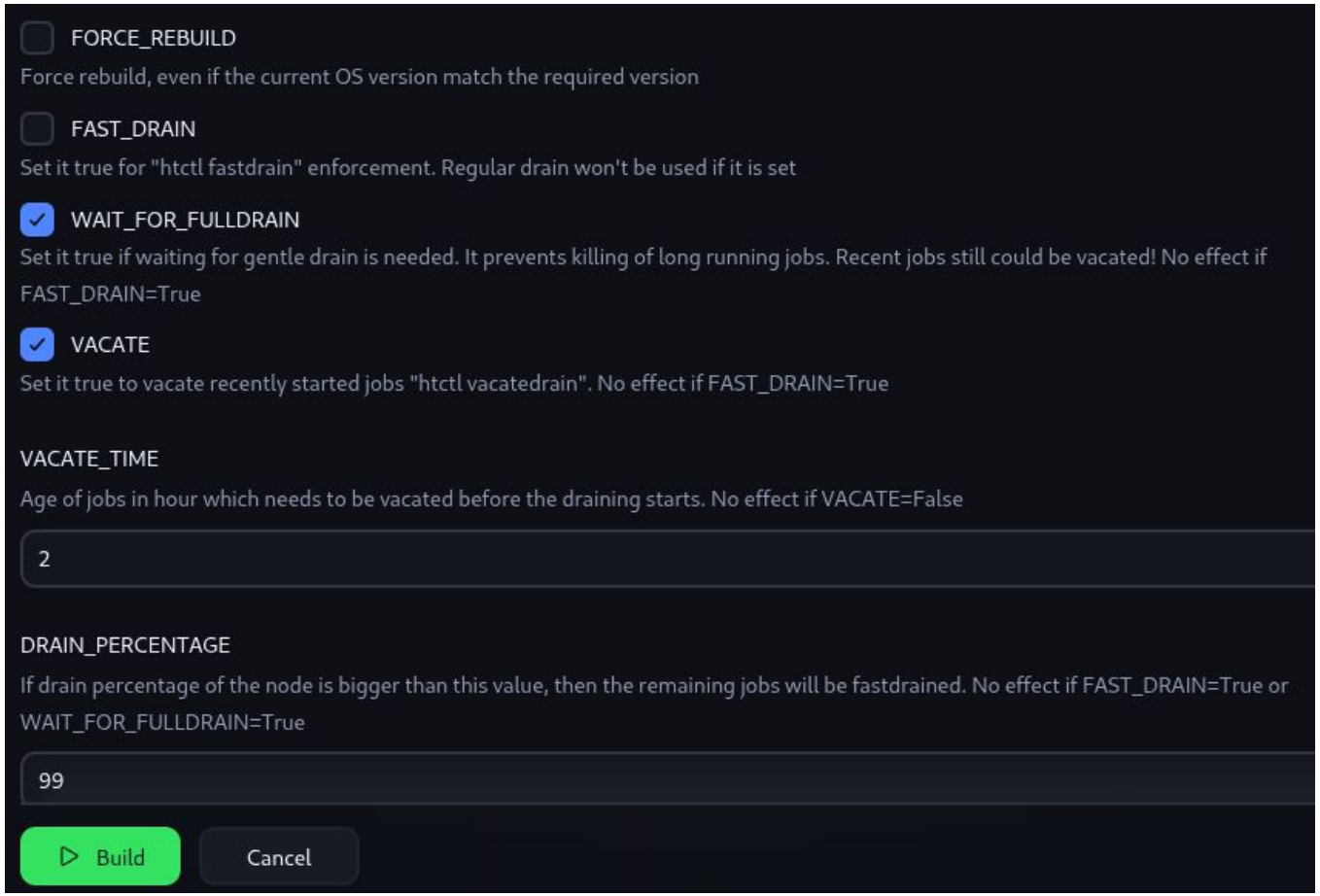
Condor Wacky draining™

"Jobs are money in the bank"

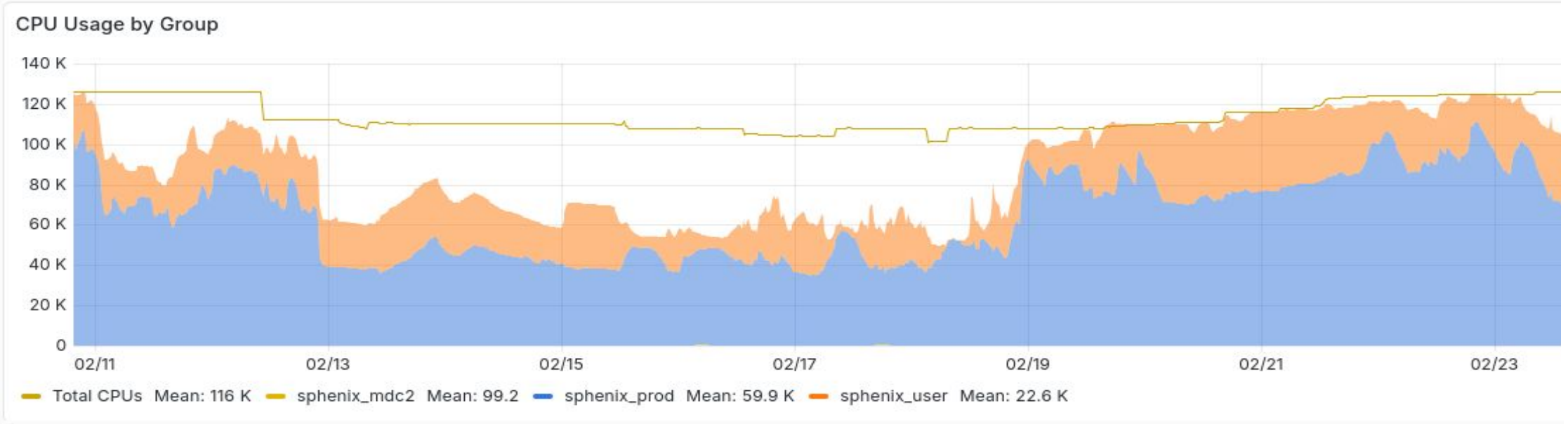
- Wacky draining your EPs is not a one size fits all solution
- Not really for use on shared pools with lots of unrelated users
- Can be very attractive on dedicated pools where the "victim" is the beneficiary
  - "Give me $10 now and I'll give you $100 later today"

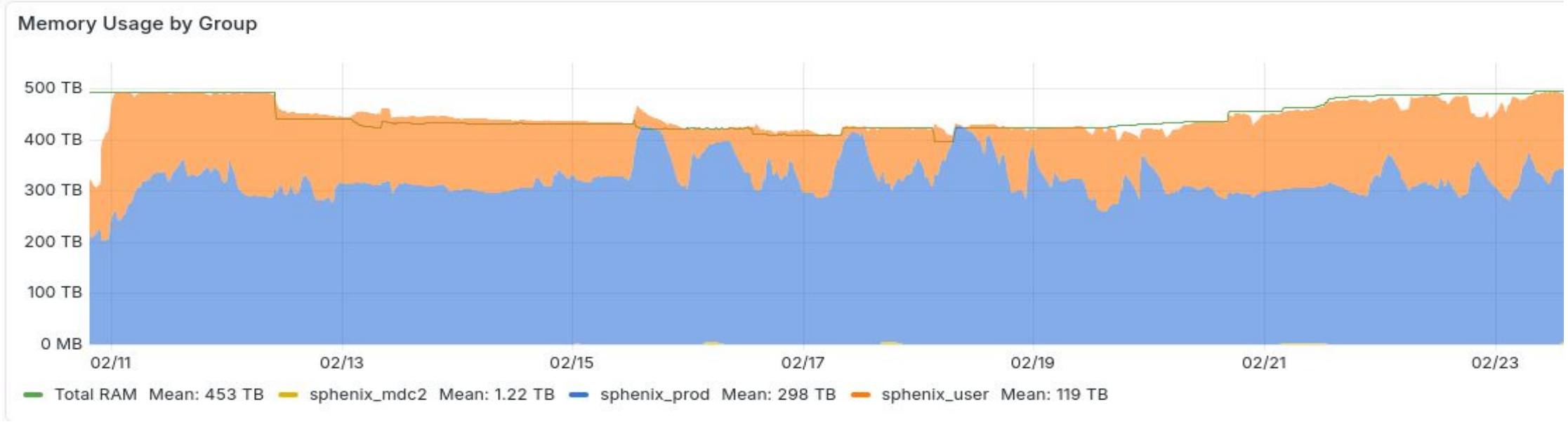
- Jenkins drain / rebuild / start pipeline (Thanks Oszkar Tarjan!)





- Jenkins drain / rebuild / start pipeline (Thanks Oszkar!)



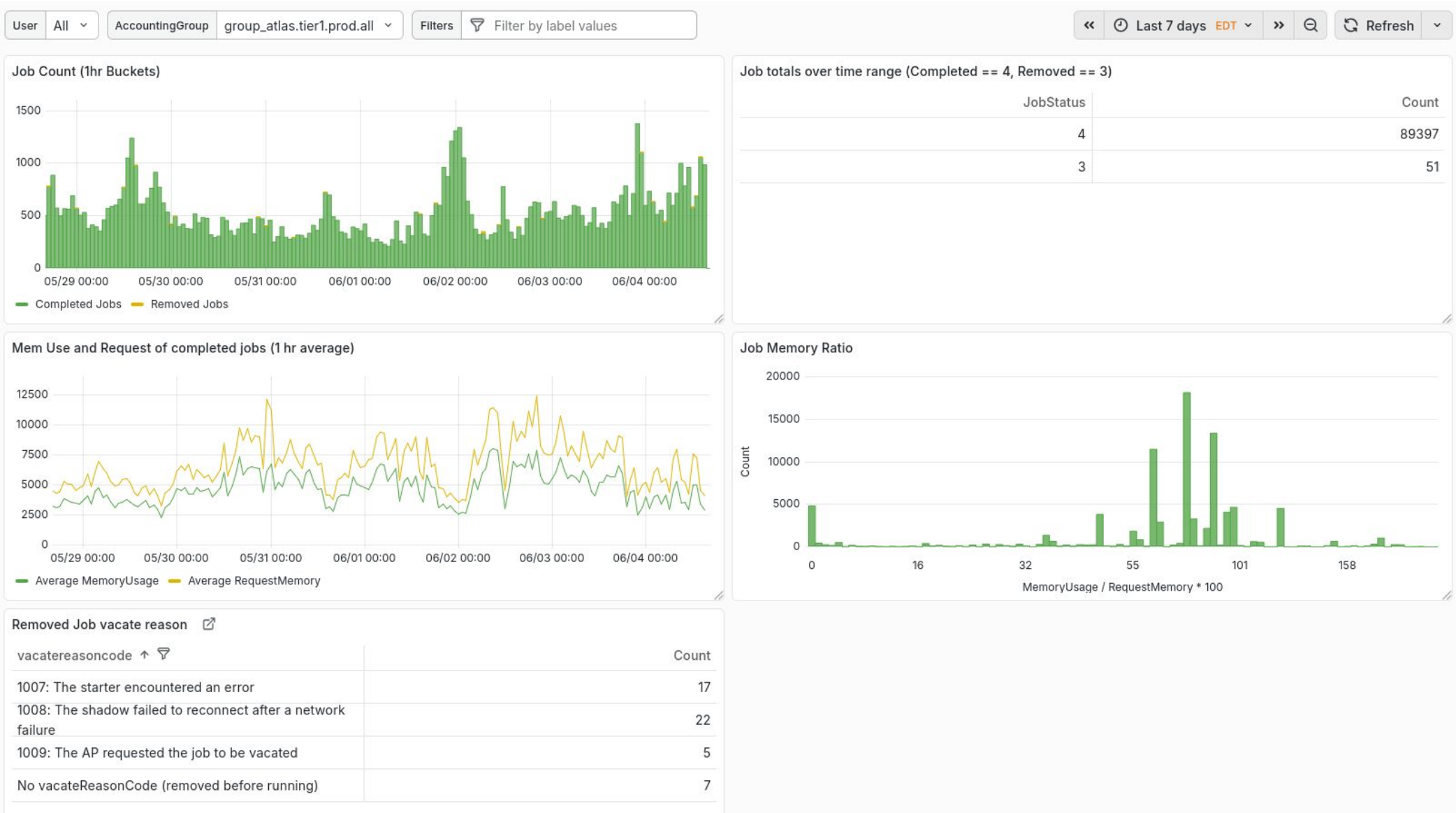


Brookhaven National Laboratory

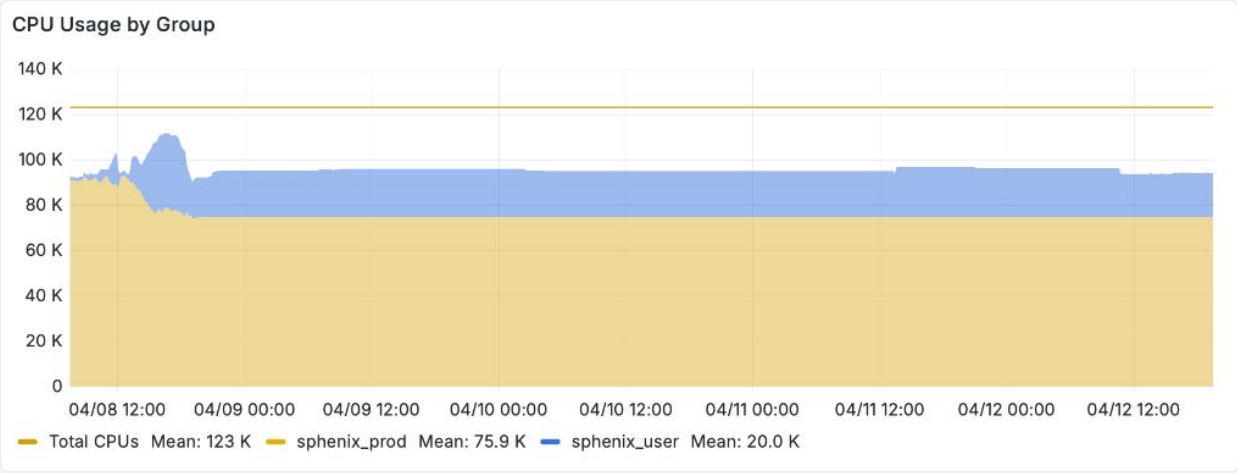
Monitoring (to improve, you must quantify)

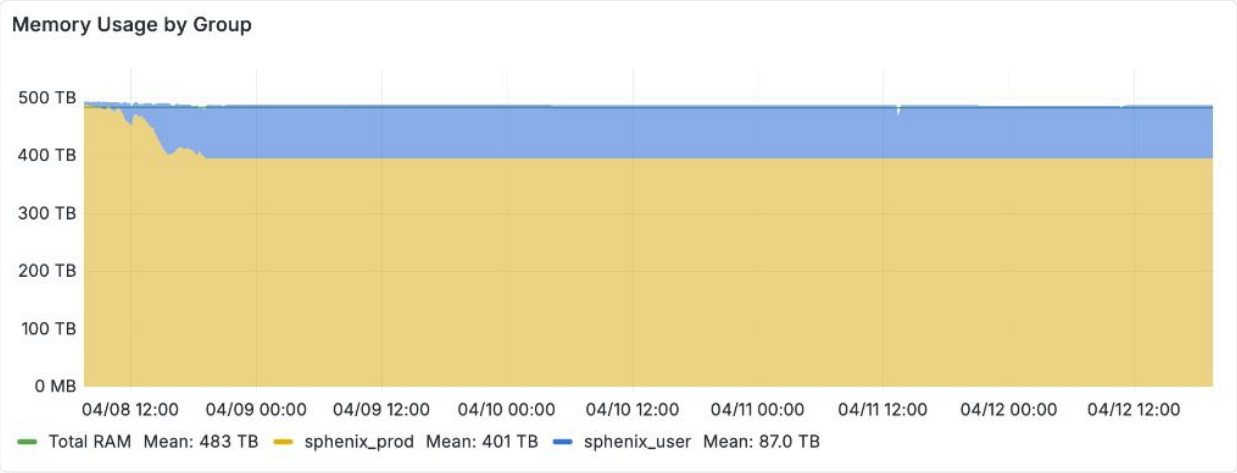
HTCondor adstash -> Opensearch

- **Opensearch -> grafana**
  - Work in progress (grafana not in dark mode)



Memory constrained workloads:





"I have loads of idle CPU, but no memory left"

- Are we making good memory requests? Maybe
- Can we do better? YES!

Submit File Resources

- Use a list instead of a number; try the first, if resource exceeded, re-run with the next value.

```
RequestMemory = 1000, 5000
```


-Todd Tannenbaum, 2025 (colorized)

Lots of ways to do it

```
request_memory = 1 GB, 2GB

request_memory = 1 GB
retry_request_memory = 2 GB

request_memory = 1 GB
retry_request_memory = 2 GB, 3 GB

request_memory = 1 GB
retry_request_memory_increase = RequestMemory * 4
retry_request_memory_max = 16 GB
```

“I don’t have adstash, but I want to explore memory usage anyway”

```
[root@sphnxprod01 ~]# condor_history -constr 'User=="sphnxpro@bnl.gov"' -af:tj MemoryUsage RequestMemory 'MemoryUsage*1.0/RequestMemory*100'
328788.122      2930    4096    71.533203125
328794.36       2930    4096    71.533203125
328775.250      2930    4096    71.533203125
328775.179      3174    4096    77.490234375
328776.1        3418    4096    83.447265625
328823.145      2686    6144    43.71744791666667
328814.32       3174    4096    77.490234375
328816.260      2686    4096    65.576171875
328815.83       2686    4096    65.576171875
```

awk, grep, sed, …….

R&D

- **To hyperthread or not to hyperthread?** (why is this a question?)
  - fact: memory is such a huge fraction of the cost of servers these days
  - based on our workloads, HT on provides at best ~130% throughput compared to the same system with HT off
  - Throughput per dollar can be maximized by buying twice as many systems with half as much memory, and turning HT off (with caveats)
- **cpu oversubscription**
  - we have **some workloads** that spend a decent amount of time idling cpu while waiting for files
  - we can lie to condor and tell it that it has more cpus, and it will get more work
  - gains (if any) not quantified yet, expect it to be marginal at best
- **electrical power based RANK and other things**
  - We already have power data, can add it to machine classad with a startd_cron
  - setting a machine's RANK based on power usage of the server row
  - we can set a maximum power ceiling, and not accept more jobs when exceeded
- **AI stuff (logs monitoring? anomaly detection? Chat, are we cooked?)**

Any questions ?

Thanks to:

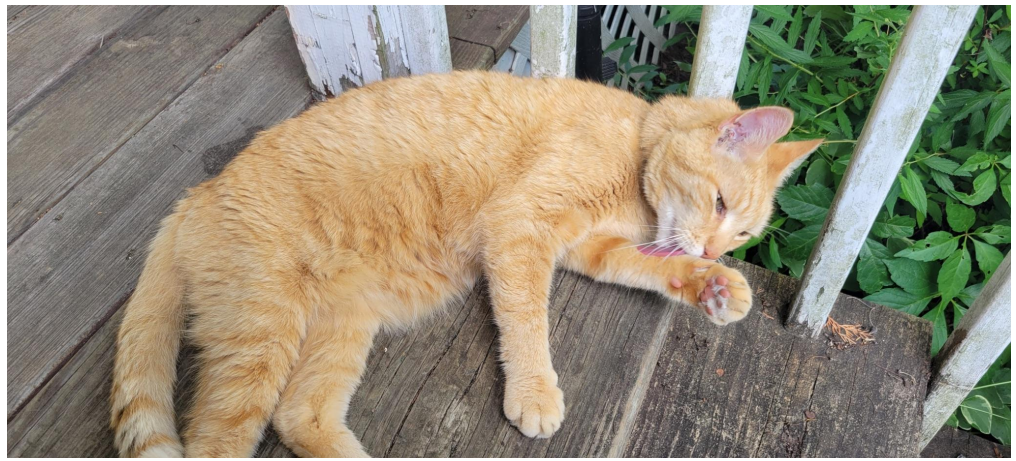
- BNL SCDF Staff
- ATLAS community
- sPHENIX experiment

Special Thanks to:

- SCDF IT Fabric team
  - Matt Cowan, Costin Caramarcu, Kevin Casella, Oszkar Tarjan, Zhihua Dong, Shigeki Misawa
- HTCondor team

Very Special Thanks to:

- TJ (John Knoeller) from HTCondor team



Brookhaven National Laboratory

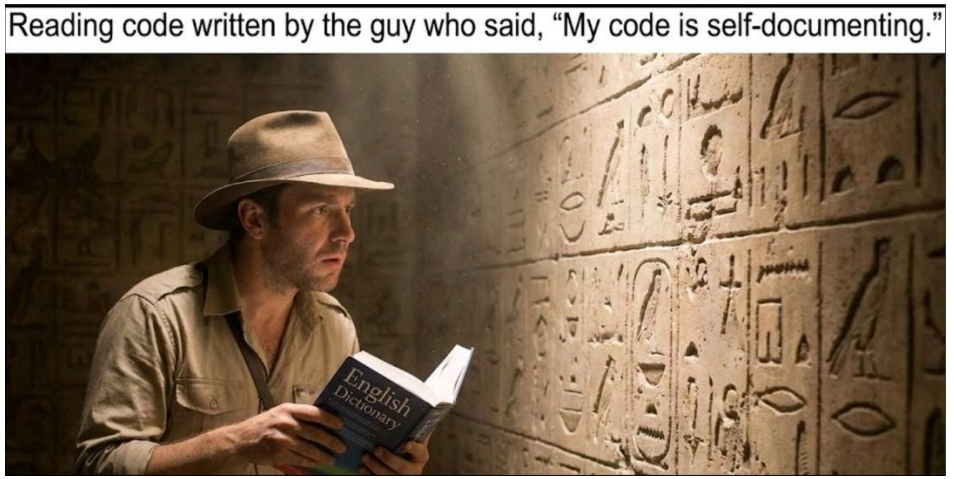
Contact me at: tsmith@bnl.gov

Bonus Slides

Puppet infrastructure code

- Simplify the structure - avoid a web of interconnected classes
- Simple is better than clever - avoid complex and fragile logic (when possible)
- Written with Foreman / Hiera in mind - make use of host group parameters
- Fail safe - default values reflect the test pool
- Useful and plentiful comments
  - parameters fully described (use / purpose / origin / defaults)
  - files created by puppet include a header describing their source
- Written with testing in mind, not an afterthought

```
manifests
├── ap.pp
├── ce.pp
├── cm.pp
├── ep_arm.pp
├── ep_gpu.pp
├── ep.pp
├── init.pp
└── misc
    ├── adstash.pp
    └── condor_autostart.pp
```



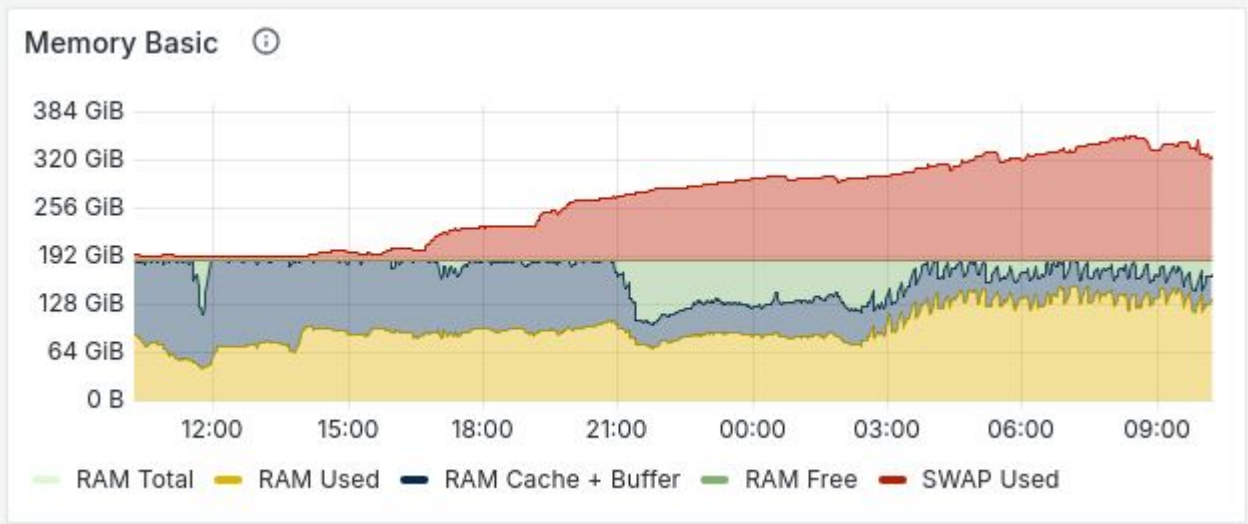

```
htcondor
├── files
│   ├── adstash
│   │   ├── 19_adstash
│   │   └── opensearch.sdcc.bnl.local.pem
│   ├── condor_autostart
│   │   ├── condor_autostart.py
│   │   ├── condor_autostart.service
│   │   └── condor_autostart.timer
│   ├── configs
│   │   ├── ap
│   │   │   ├── common_ap
│   │   │   ├── sphnx_ap
│   │   │   ├── spool_ap
│   │   │   └── test_ap
│   │   ├── ce
│   │   │   ├── atlas_ce
│   │   │   ├── common_ce
│   │   │   └── spool_ce
│   │   ├── cm
│   │   │   ├── atlas_cm
│   │   │   ├── common_cm
│   │   │   ├── jupyter_cm
│   │   │   ├── sphnx_cm
│   │   │   ├── spool_cm
│   │   │   └── test_cm
│   │   ├── common
│   │   │   ├── condor_config
│   │   │   ├── condor_config.old
│   │   │   └── condor_config_source
│   │   ├── ep
│   │   │   ├── atlas_ep
│   │   │   ├── common_ep
│   │   │   ├── sphnx_ep
│   │   │   ├── spool_ep
│   │   │   └── test_ep
│   │   ├── misc
│   │   │   └── 20_aarch
│   │   └── pool
│   │       ├── atlas_pool
│   │       ├── atlas_pool.old
│   │       ├── sphnx_pool
│   │       ├── sphnx_pool.old
│   │       ├── spool_pool
│   │       ├── spool_pool.old
│   │       └── test_pool
│   ├── groups
│   │   ├── sdcc-group-definitions
│   │   └── sphnx-group-definitions
│   ├── htctl
│   └── pool_key
```

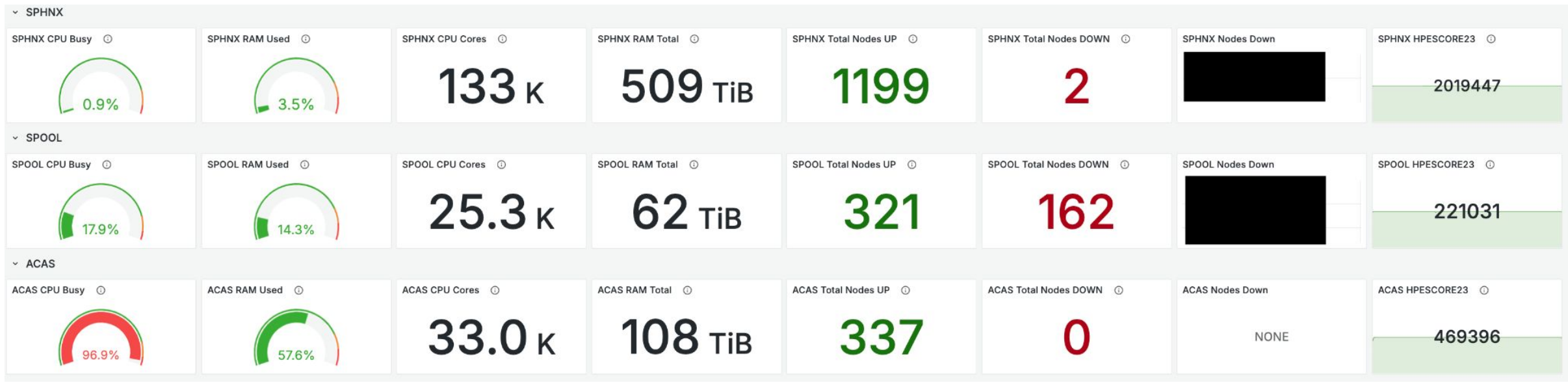
```
### Tom - attempt at a complete rewrite of condor class and all that stuff (here we go...) ###
#
# HTCondor main class
# Top level class, common across all condor nodes regardless of specific function
#
# Parameters :
#   condor_version    : version of the condor package desired.
#                       defaults to 'present' if not set
#
#   condor_pool       : the pool that the node is a member of (atlas,sphnx,spool,jupyter,test)
#                       defaults to 'test' if not set (safe default)
#
#   condor_experiment : the particular experiment this node is affiliated with, for example star,phenix,eic,dune,etc
#                         this parameter is only really relevant on APs, and has an impact on 30_experiment template
#                       defaults to the value of $condor_pool if not set
#
#   condor_test_cm    : optional parameter to point host to a different CM. Useful if you want to build nodes with production-
#                         like config, but want to point them to a different CM, like the testpool
#                       defaults to empty string ''
#
#  condor_pool_key    : optional parameter if you wish to specify a /home/condor/pool_key different from the normal one we use for
#                         our pools. Useful for any pool that has portions external to our site, so as not to leak secrets
#                       defaults to empty string ''
#
#   Parameters can be set in hiera or foreman, but foreman has the highest authority
#   Note: parameters set in hiera or as foreman smart class parameters are class scope ($facts['parameter_name'])
#         parameters set in foreman (global, host group, host level) are considered global scope ($::parameter_name)

class htcondor (  $condor_version    = pick($facts['condor_version'],$::condor_version,'present'),
                  $condor_pool       = pick($facts['condor_pool'],$::condor_pool,'test'),
                  $condor_experiment = pick($facts['condor_experiment'],$::condor_experiment,$condor_pool),
                  $condor_test_cm    = pick_default($facts['condor_test_cm'],$::condor_test_cm),
                  $condor_pool_key   = pick_default($facts['condor_pool_key'],$::condor_pool_key) ) {
```

Prometheus / Node Exporter

Prometheus / Node Exporter cont'd





Honorable mention: Nagios (not pictured)

condor_history (drinking from the firehose)

condor_history -constraint 'your_constraints_here' -af:tj MemoryUsage RequestMemory 'MemoryUsage*1.0/RequestMemory*100'

Use -constraint to pre-filter your output (or omit to get everything)

-af (autoformat) accepts modifiers after the colon, I use t for “tab” and j for “include the jobid as the first column”

Can show any number of attributes, and even do simple math on them!