Brookhaven National Laboratory

U.S. DEPARTMENT of ENERGY

Managing, Maintaining, and Monitoring a large HTC system

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

2025 June 4, HTC25

@BrookhavenLab

Brief Overview of our HTC system

(BNL Site Report in 1 slide)

	ATLAS Tier 1 pool	sPHENIX pool (new!)	Shared pool
Worker nodes (EPs)	~340	~1200	~480
Logical cores	33k	133k	40k
HEPscore23	~490k	~2.03M	~485k
Condor CEs	8	0	2
Submit nodes (APs)	0	12	40+

ATLAS EXPERIMENT sPHENIX STAR Belle II DUNE DEEP UNDERGROUND NEUTRINO EXPERIMENT PHENIX …and more!

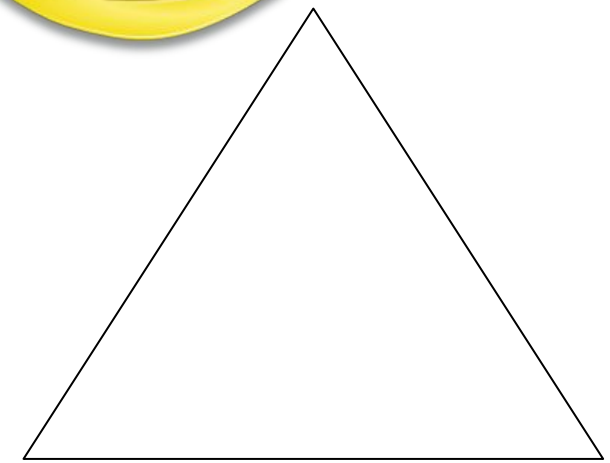
It was a busy year..

- EOL for el7 (Migrate EVERYTHING to Alma Linux 9)
- New provisioning system (Foreman)
- Puppet code updated
- Addition of 23 racks of compute (~100k logical cores)
- Retired some truly ancient servers

- New ARM arch queue
  - Ampere Altra, Altra Max
  - Nvidia Grace/Grace
  - See bonus slide for benchmark
- Monitoring improvements

Brookhaven National Laboratory

Managing the linux farm



Provisioning with Foreman:

- Handles the OS management, PXE booting, DHCP, TFTP
  - Kickstart/preseed, partitioning
- Serves as Puppet ENC, handles puppet parameters / environment switching

Orchestration with Puppet:

- Handles post-install configuration
- Runs on a schedule to deploy changes / ensure compliance

Version Control with Git (Gitea):

- Pretty self explanatory. Use git
- Git branches correspond to puppet environments (deployed with r10k)

Maintaining the HTC system

The end of el7 brought on many challenges (and opportunity!)

- Updating the infrastructure (OS upgrades, RHEV -> ~320 Openshift VMs)
- Updating the HTCondor configurations
  - Clean up (a lot) of outdated / redundant configs
  - Use defaults when possible
  - Capture *everything* in configuration management
    - If it isn't in git, it didn't happen
    - one-off manual changes not allowed in production

Within the last 2 years we upgraded HTCondor a bit

9.0.X -> 10.0.X -> 23.0.X -> 23.9.X* -> 24.0.X

*needed to jump off LTS for a bit to grab some critical cgroups fixes included in later LTS versions

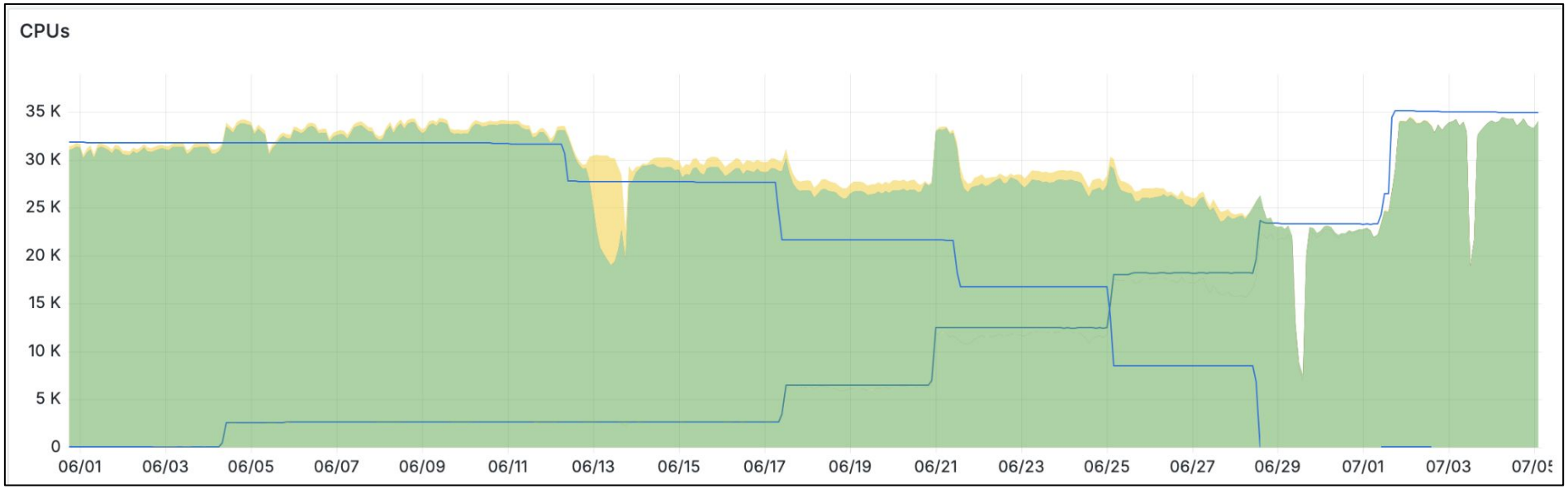
HTCondor upgrade strategies

Strategy 1: Rolling upgrade / mixed major version pool

- The following steps can be performed in no particular order
  - Drain EPs in batches, upgrade, then add back to the same pool
  - Upgrade APs and CEs (respecting redundancy), add back to same pool
  - Upgrade Central Manager <- (surprisingly the easiest part)

- **Pros:**
  - 100% uptime (very nice!)
  - *mostly* transparent to users
  - Pool is unified during the upgrade

- **Cons:**
  - Mixed version pool (across Major versions)
  - configuration of old/new need to play nice
  - *Feels* like testing in production



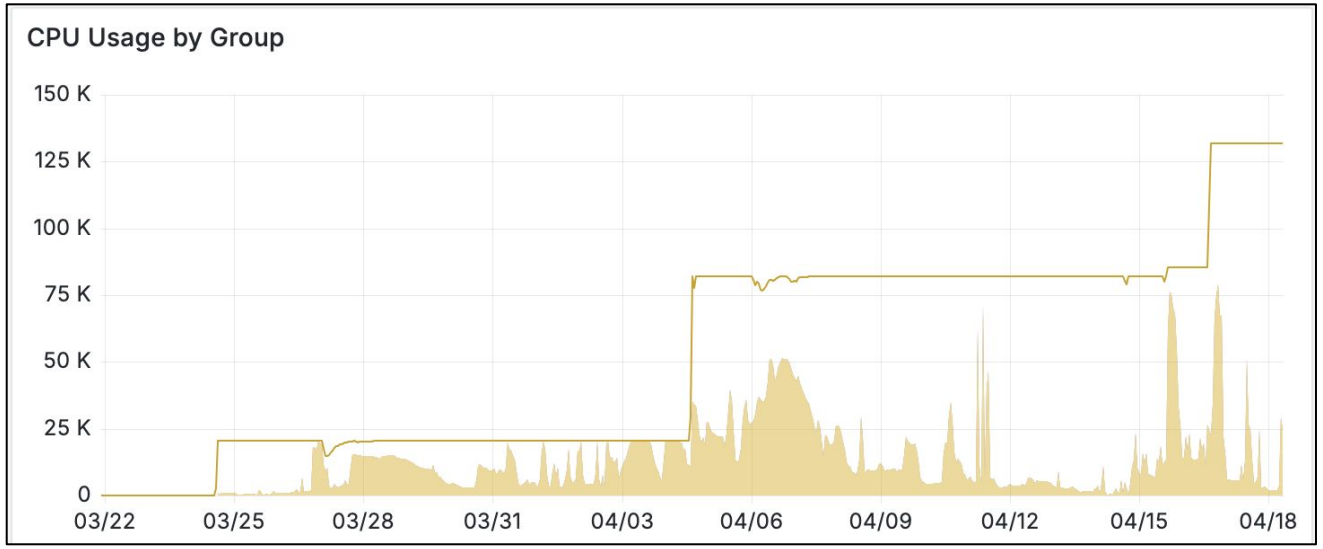
Brookhaven National Laboratory

HTCondor upgrade strategies (part 2)

Strategy 2: Create a new separate pool (aka- new pool, who dis?)

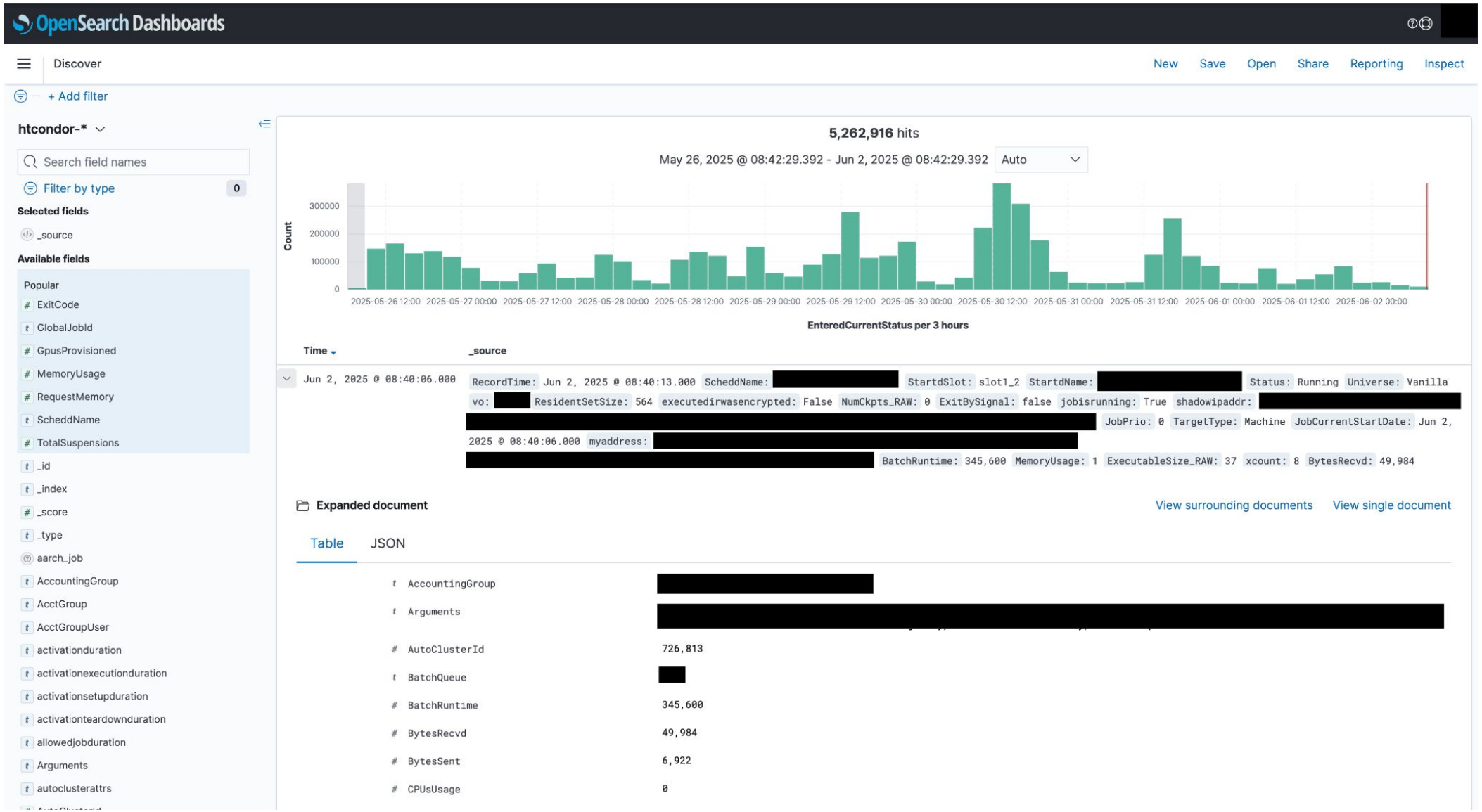
- Create the “vertical slice”
  - Brand new Central manager
  - New (or stolen from old pool) EPs
  - New (or stolen from old pool) APs/CEs
- Test the new pool under production like circumstances (get brave users to run real jobs, etc)
  - repeat until satisfied
- Drain and migrate / rebuild resources from old pool until finished

- **Pros:**
  - New pool configuration can be different
  - Homogenous versions (very nice!)
  - No version compatibility testing required
- **Cons:**
  - Pool is split during upgrade
  - Not so transparent for users



Monitoring - all the things!

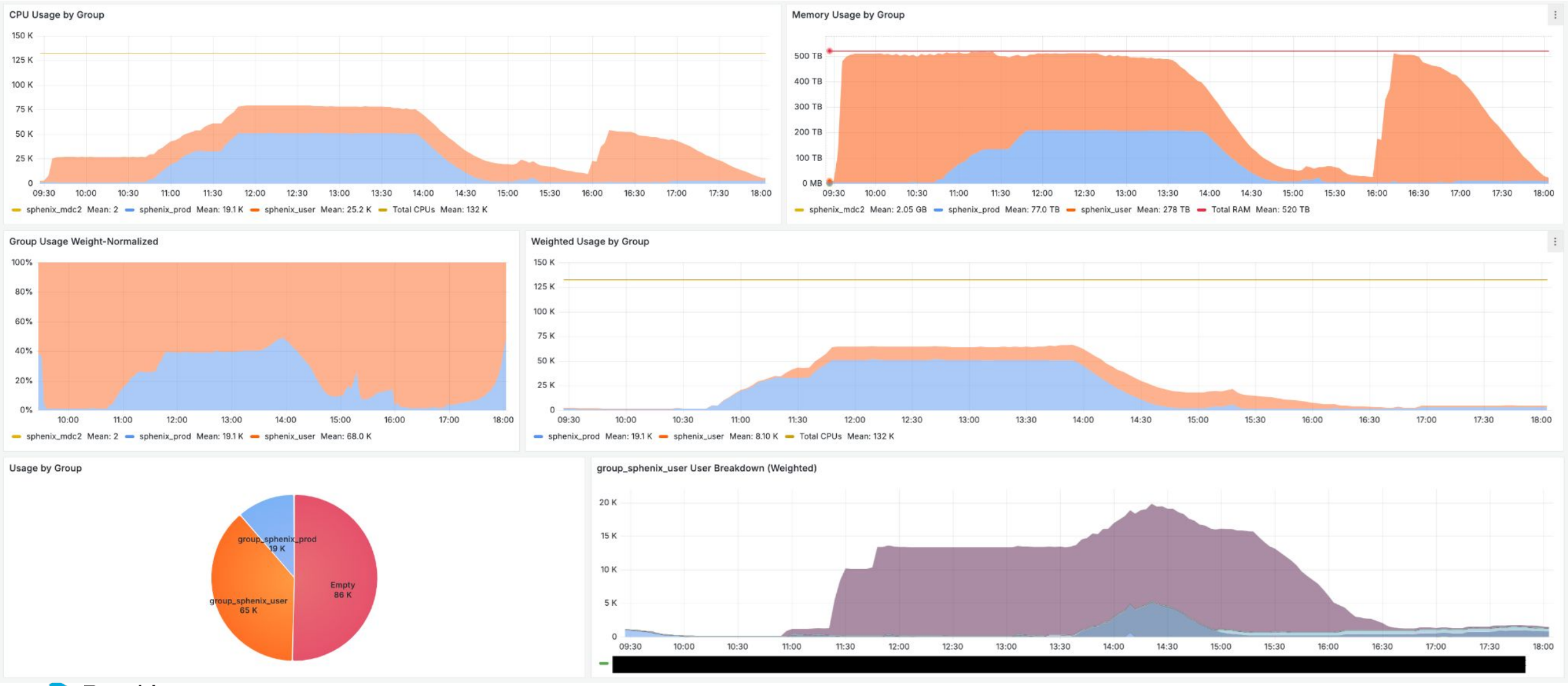
Condor adstash / Opensearch



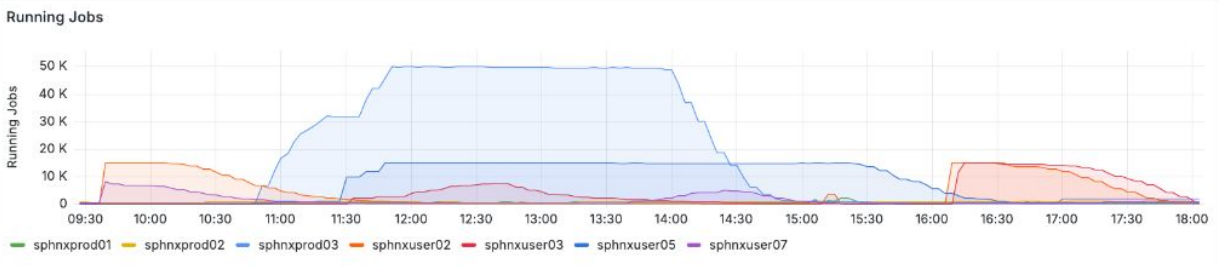
Brookhaven National Laboratory

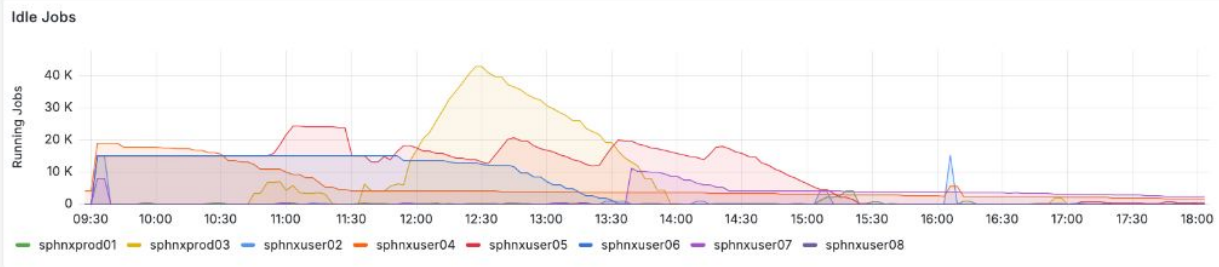
Grafana / Fifemon

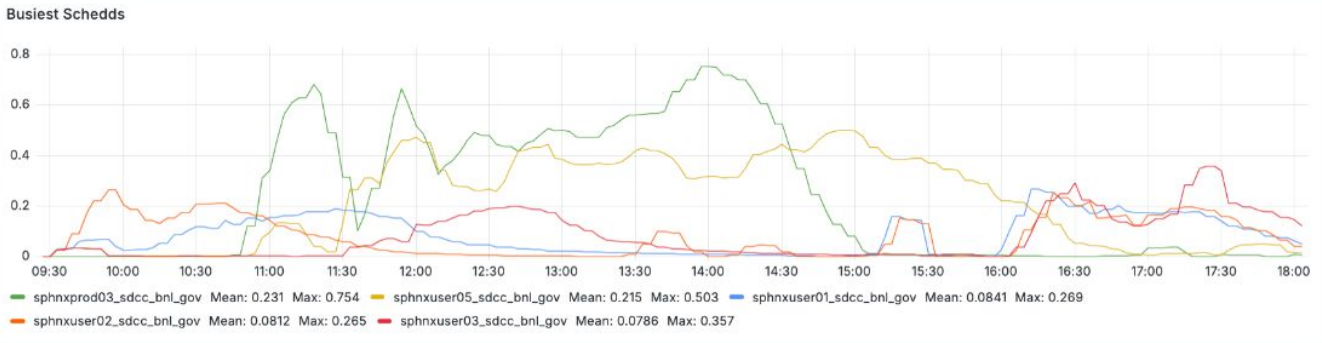
https://research.cs.wisc.edu/htcondor/HTCondorWeek2017/presentations/ThuStrecker-Kellogg_Monitoring.pdf
https://research.cs.wisc.edu/htcondor/HTCondorWeek2016/presentations/ThuRetzke_Fifemon.pdf

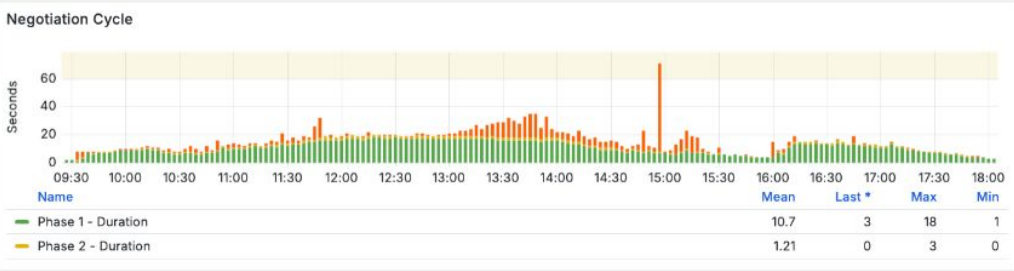


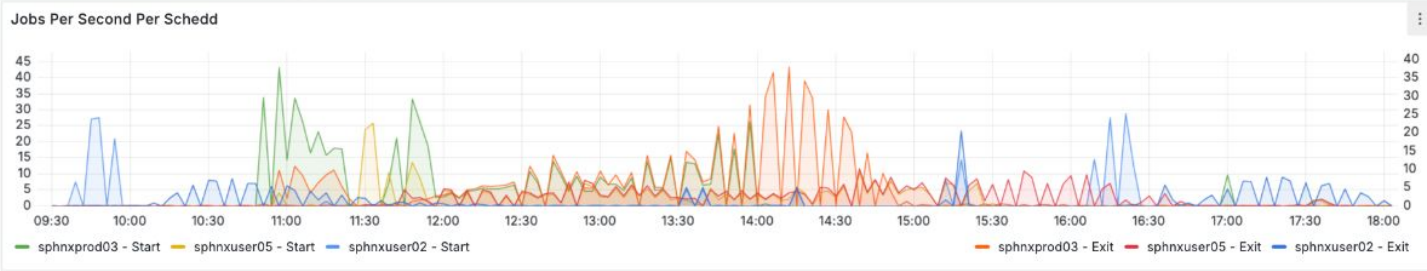
Brookhaven National Laboratory

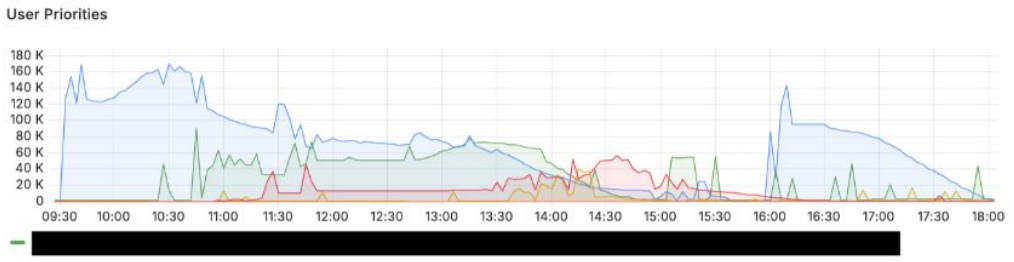


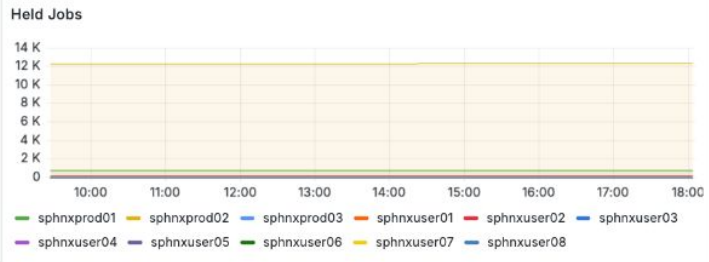


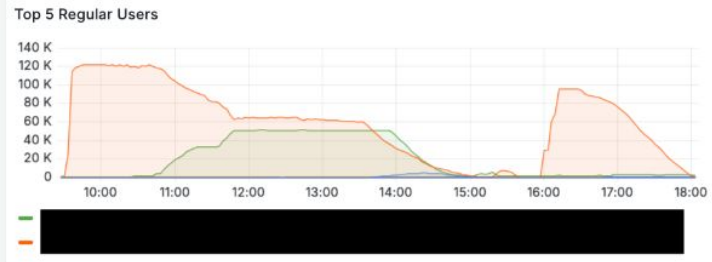








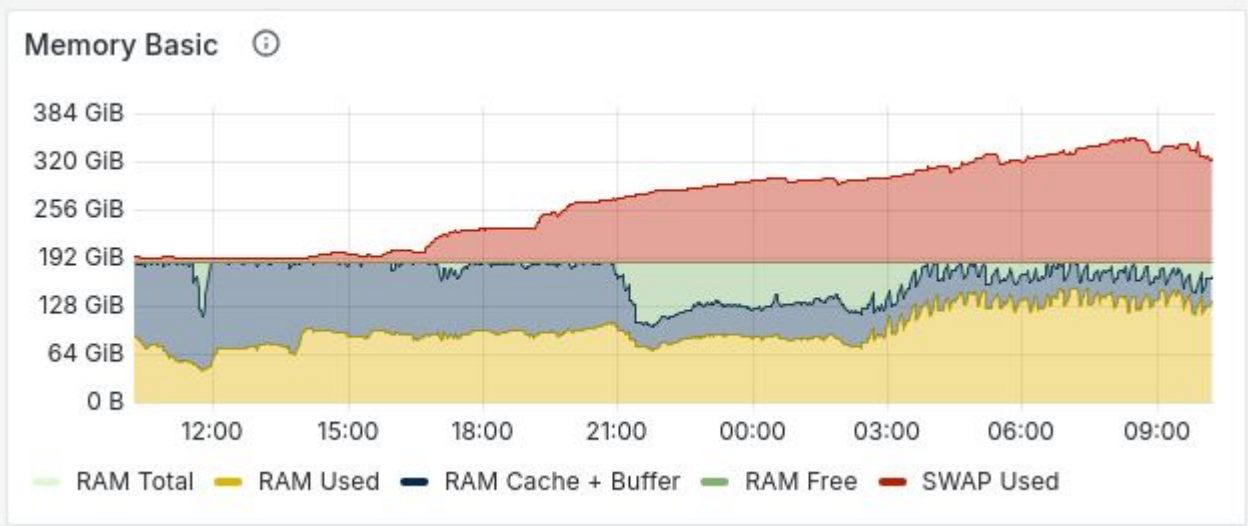


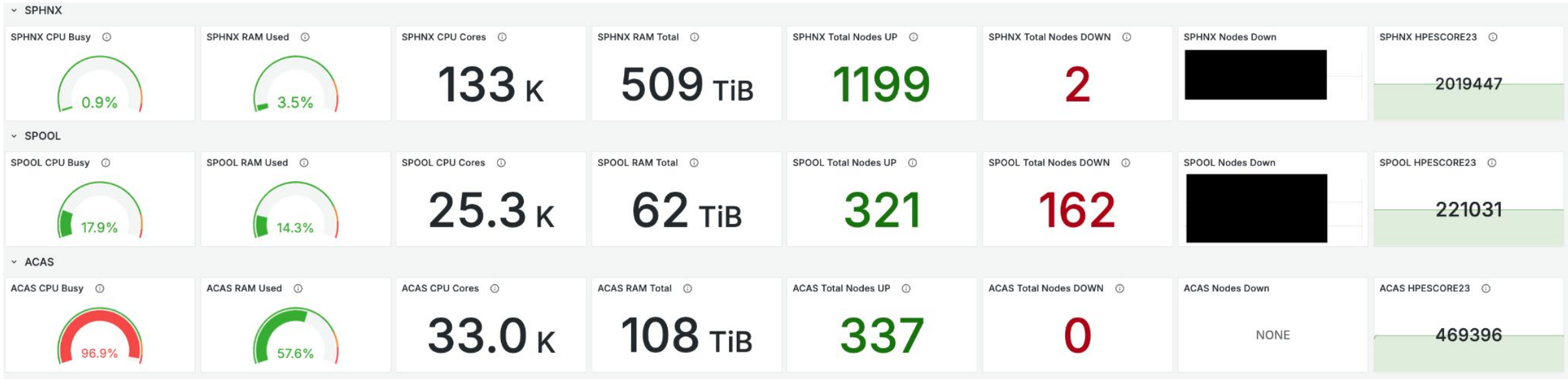


Brookhaven National Laboratory

Prometheus / Node Exporter

Prometheus / Node Exporter cont'd





Honorable mention: Nagios (not pictured)

HTCondor Wish~~list~~

Problem: I want to upgrade my pool(s) more often, but every time I do, it wastes (literally) millions of cpu hours of compute

Facts:

- LTS releases are pretty often, and contain a lot of fixes that we want
- Upgrading our pool(s) is a costly endeavor, so we must pick and choose / prioritize the fixes and features we need vs want

(Impossible?) Request: A way to upgrade the condor version on EPs that doesn't require fully draining it. (Can already running jobs finish under the old version, and new ones start under the new version?)

Acknowledgment / 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

Bonus Slides

Most common hold reason nowadays…

```
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8027 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8063 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 7941 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8019 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8160 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 7999 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8048 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8026 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8053 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 7988 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8018 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8049 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8041 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8105 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8158 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 7986 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8018 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 7980 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8127 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8134 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8094 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8164 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8140 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8091 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8019 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8010 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8132 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8053 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8043 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8063 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8033 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8029 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8085 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8014 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8118 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8138 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8148 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8131 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8112 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8123 megabytes.  Consider resubmitting with a higher request_memory.
Job has gone over cgroup memory limit of 8192 megabytes. Last measured usage: 8159 megabytes.  Consider resubmitting with a higher request_memory.
```

Jobs get Held the moment they go over what they request. Working as intended!
Prevents machines from swapping / exhausting memory

Solution:
Simply re-submit with a larger request

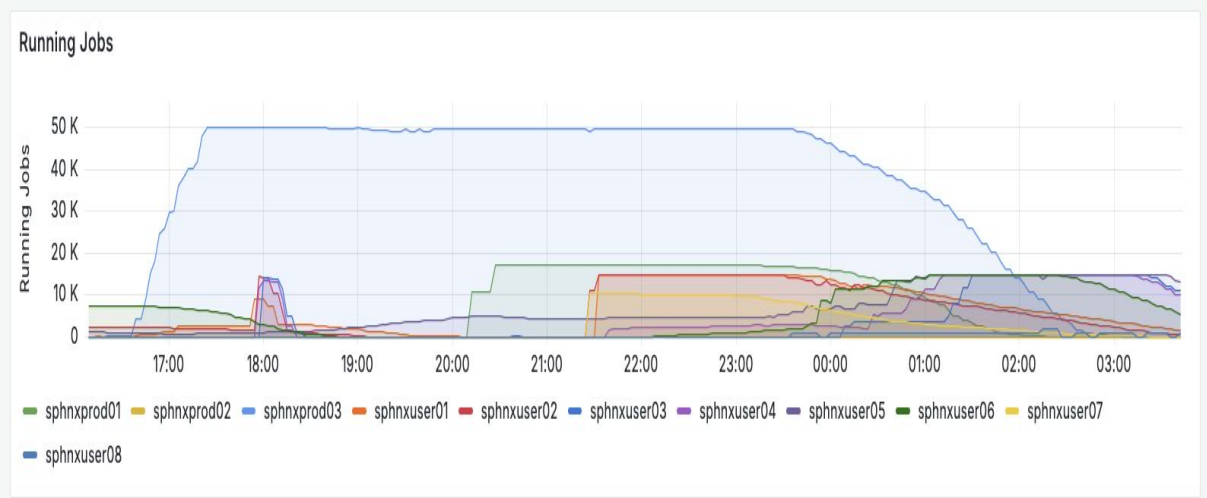
Some favorite condor commands

condor_who	What's running on the EP (nice replacement for `ps -ef | grep starter`)
condor_status -compact	Overview of startds (EPs)
condor_status -schedd	Overview of schedds (~APs, CEs)
condor_status -master	Overview of versions, condor daemon uptime
condor_q -global -limit X	Show X number of jobs in every queue in the pool
condor_config_val -summary	Shows configs you changed (and where!)
condor_config_val -dump | grep -i <something>	Looking for <something> in your config?
condor_status -direct `condor_who -daemons | grep Startd | awk '{print $6}'`	Run on an EP, give me the output of condor_status of myself, but don't query the CM (This idea is useful for certain types of cron jobs so you don't kill the CM with requests)
condor_status -compact -af:t Machine <Attribute1> <Attribute2> … condor_q -af:jt <JobAttribute1> <JobAttribute2> …	Nice format for feeding into grep / sed / awk. You can cook some crazy bash one liners to do just about anything. Autoformat is great

Condor AP tuning for absurd number of running jobs

We wanted to be able to fill our sPHENIX pool (130k logical core) with the fewest number of APs

- Worst case: all single core jobs
- Settled on 3 APs with up to 50k running jobs each, a 4th for redundancy (soon)
- We allot ~2MB of memory per running job (>100GB of memory total)
- needed to increase linux ephemeral port range (Theoretical max of ~65K)
  Default was 32768-61000
  Changed to 10000-65000
- Some other config changes:
  - MAX_JOBS_PER_SUBMISSION=1000000000
  - MAX_JOBS_RUNNING=50000
  - MAX_JOBS_PER_OWNER = 1000000



ARM Arch (and x86) benchmarking results

	6336Y	6448Y+	6538Y+	6548Y+	6766E	9754	6766E	ARM Ampere Altra	ARM Ampere AltraMax	ARM NVIDIA Grace *
HS	1285	2238	2255	2496	2401	3570	4580	1041	2029	4435
HS/thread	13.39	17.48	17.62	19.5	16.67	13.95	15.9	16.27	15.85	30.8
Cores	2 x24	2 x32	2 x32	2 x32	1 x144	1 x128	2 x144	1 x64	1 x128	2 x72
Threads	96	128	128	128	144	256	288	64	128	144
Power (W)	601	654	692	695	-	-	-	252	370	817
HS/W	2.14	3.42	3.26	3.59	-	-	-	4.13	5.49	5.43