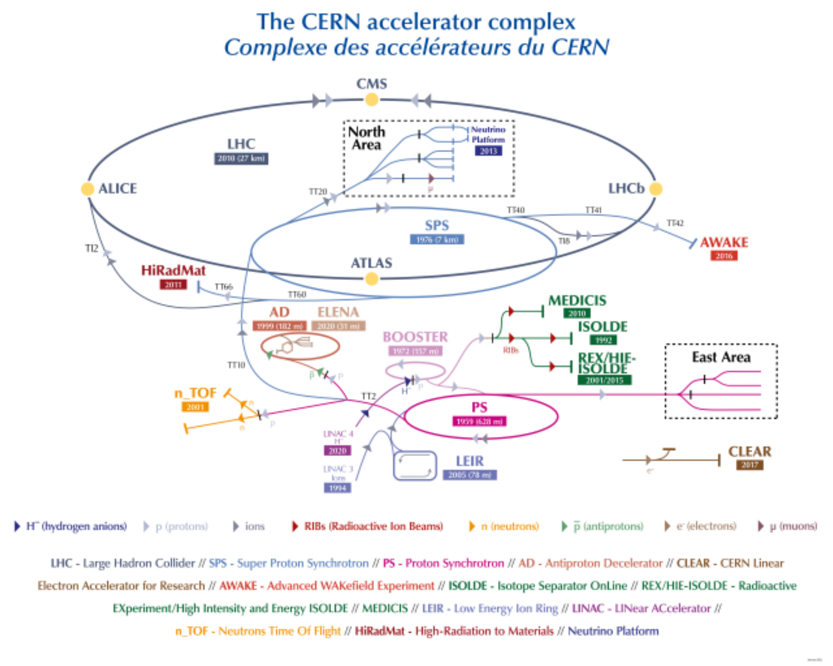
CERN

Tier-0 Batch Computing

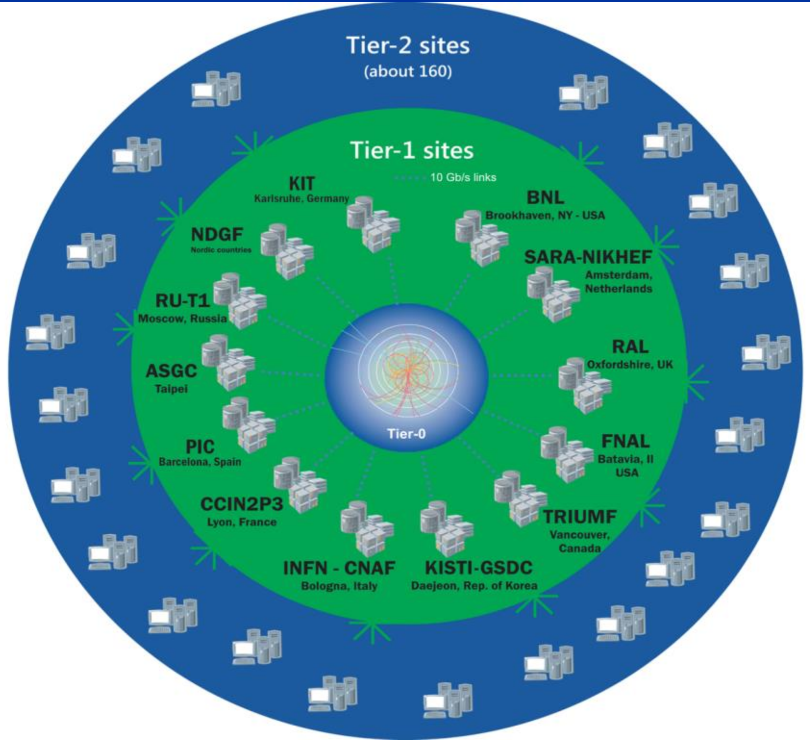
Ben Jones IT-CD-CC

CERN Batch System


The CERN accelerator complex
Complexe des accélérateurs du CERN
CMS
LHC
2010 (27 km)
North Area
Neutrino Platform
2013
ALICE
LHCb
TT20
TT40
TT41
TT42
SPS
1976 (7 km)
AWAKE
2016
TI2
TI8
ATLAS
HiRadMat
2011
TT66
TT60
AD
1999 (182 m)
ELENA
2020 (31 m)
BOOSTER
1972 (157 m)
MEDICIS
2010
ISOLDE
1992
RIBs
REX/HIE-ISOLDE
2001/2015
East Area
TT10
n_TOF
2001
TT2
PS
1959 (628 m)
LINAC 4
H⁻
2020
LINAC 3
Ions
1994
LEIR
2005 (78 m)
CLEAR
2017
H⁻ (hydrogen anions)
p (protons)
ions
RIBs (Radioactive Ion Beams)
n (neutrons)
p̄ (antiprotons)
e⁻ (electrons)
μ (muons)
LHC - Large Hadron Collider // SPS - Super Proton Synchrotron // PS - Proton Synchrotron // AD - Antiproton Decelerator // CLEAR - CERN Linear Electron Accelerator for Research // AWAKE - Advanced WAKefield Experiment // ISOLDE - Isotope Separator OnLine // REX/HIE-ISOLDE - Radioactive EXperiment/High Intensity and Energy ISOLDE // MEDICIS // LEIR - Low Energy Ion Ring // LINAC - LINear ACcelerator //
n_TOF - Neutrons Time Of Flight // HiRadMat - High-Radiation to Materials // Neutrino Platform

Local Production

The WLCG



User analysis

What does it consist of?

Access Points / Submit Side / Control plane | Execution Side

- **”Jobs” are submitted to APs, aka “schedds” or “CEs”**
  - What is a job? Can be some code to execute with some input / output expectations. Can also be a “pilot” or “glidein” – essentially an agent for another task submission service
  - A ”schedd” runs a shadow process for every “job” running on the execute side. A “CE” is merely “a schedd that can talk to the grid”.
  - Scale point: each shadow requires 0.5->1mb of memory. Horizontally scalable
- **Collector / Negotiator**
  - Stateless machines that “collect” all the info about machines and jobs and match them
  - Ultimate scale point of a “pool”: collector update time

- **Batch worker are (now) physical machines**
  - Intel machines at ~10HS/core or AMD at ~16
  - Around 2.5 -> 3Gb RAM / core
- **What is a ”slot”?**
  - Vague term for what in WLCG counts as a normal unit of compute: 1 core + 2-3Gb memory + 20GiB scratch disk
  - Importantly: we give 100 cpu_shares (ie cgroup share equiv to 1cpu)
- **“mcore”?**
  - To ease use on WLCG “mcore” or “multicore slots” usually means 8 cores (or core equivalents)

What is a job at CERN?

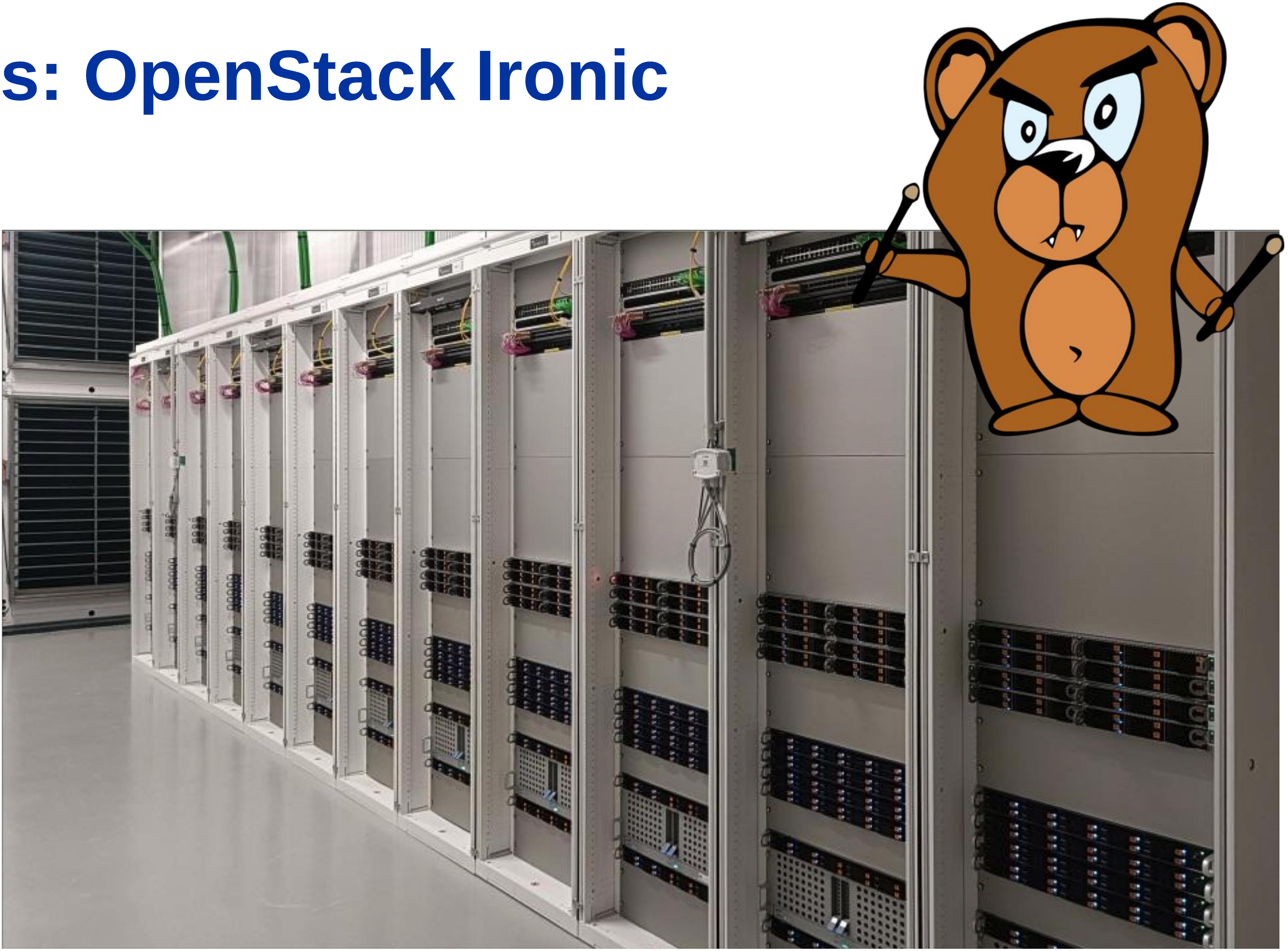
- Physicist with a submit file, sending a job via an AP (all our submission is remote…)
- An experiment's production jobs, sent by "submission framework" via an AP
- ATLAS sending jobs (real jobs) via Grid to CEs
- ALICE / LHCb sending non-condor pilots via Grid to CEs
- CMS sending glideins via Grid to CEs
- A physicist with a metascheduler sending workers via an AP

Building Blocks: OpenStack Ironic

- Bare-metal batch worker nodes provisioned by OpenStack Ironic

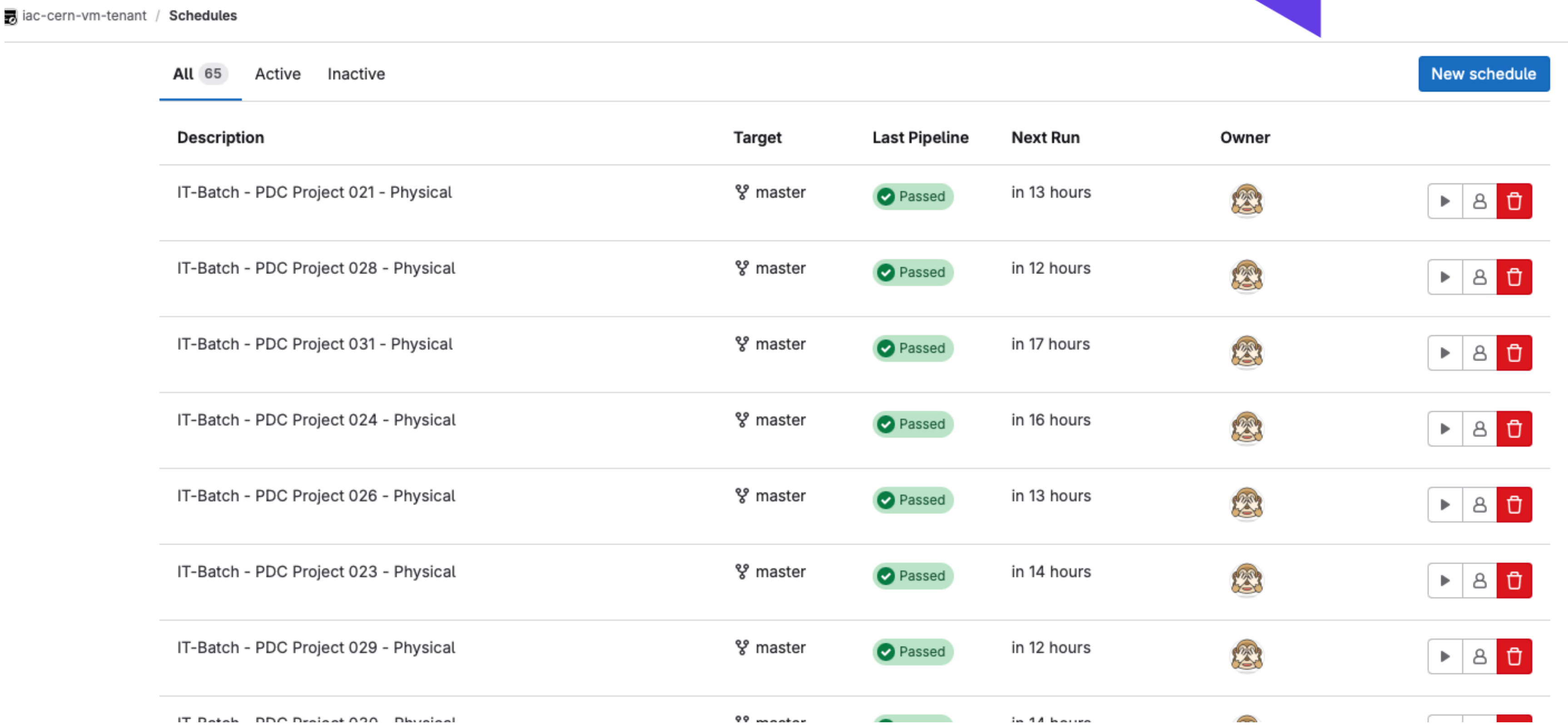
- Having cloud APIs to build machines is helpful for us to manage scale

- Separated into distinct “projects” or “tenants” of similar machines, with an IP service (often around ~200)



Building Blocks: terraform

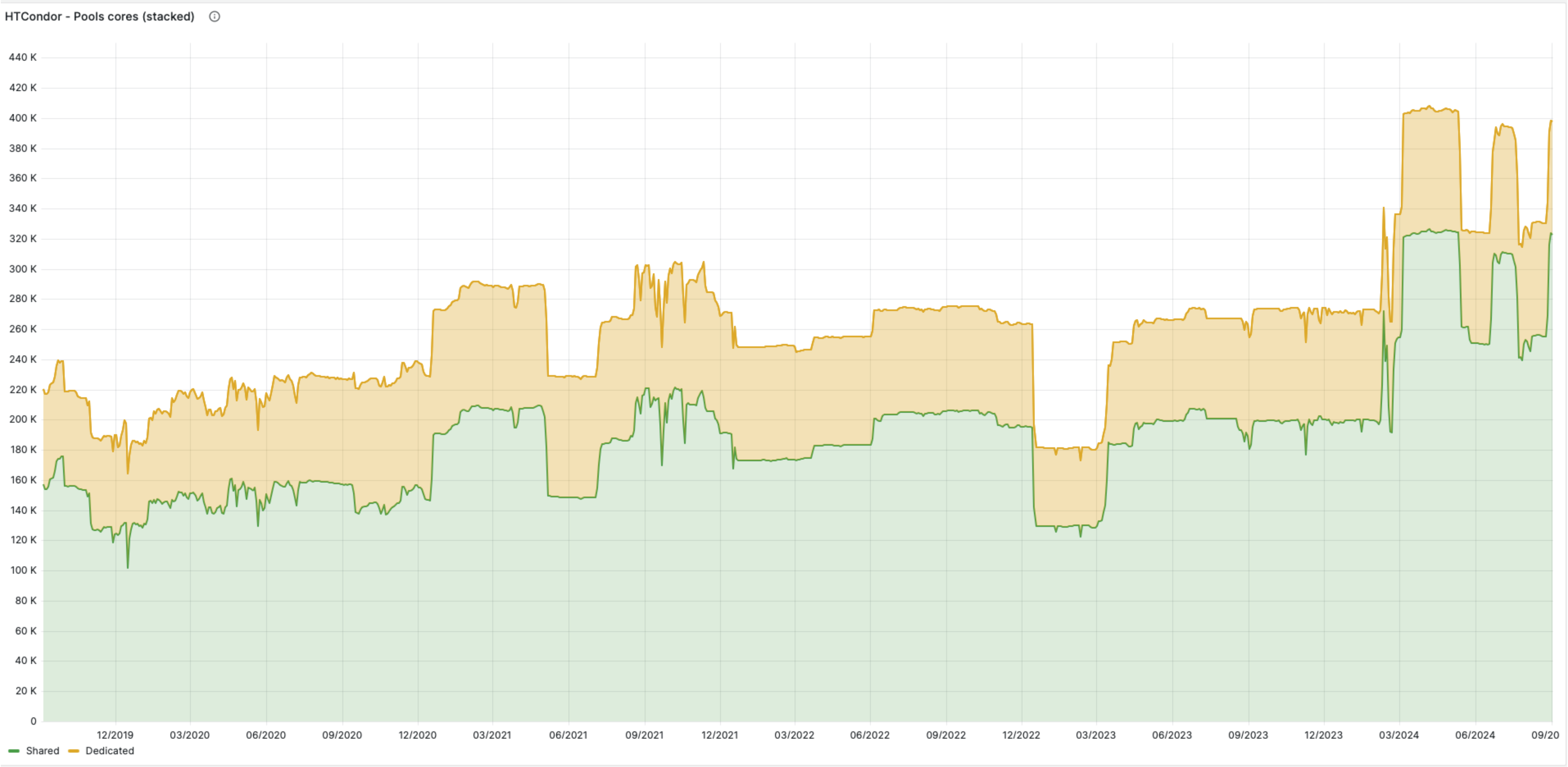
- Each OpenStack project is built out by a Gitlab-CI job running terraform
- terraform builds out to fill the quota of the project
- Machines are rebuilt every night if they have been repaired



iac-cern-vm-tenant / **Schedules**

Description	Target	Last Pipeline	Next Run	Owner
IT-Batch - PDC Project 021 - Physical	master	✓ Passed	in 13 hours	🙈
IT-Batch - PDC Project 028 - Physical	master	✓ Passed	in 12 hours	🙈
IT-Batch - PDC Project 031 - Physical	master	✓ Passed	in 17 hours	🙈
IT-Batch - PDC Project 024 - Physical	master	✓ Passed	in 16 hours	🙈
IT-Batch - PDC Project 026 - Physical	master	✓ Passed	in 13 hours	🙈
IT-Batch - PDC Project 023 - Physical	master	✓ Passed	in 14 hours	🙈
IT-Batch - PDC Project 029 - Physical	master	✓ Passed	in 12 hours	🙈
IT-Batch - PDC Project 030 - Physical	master		in 14 hours	🙈

LxBatch cores ~5y

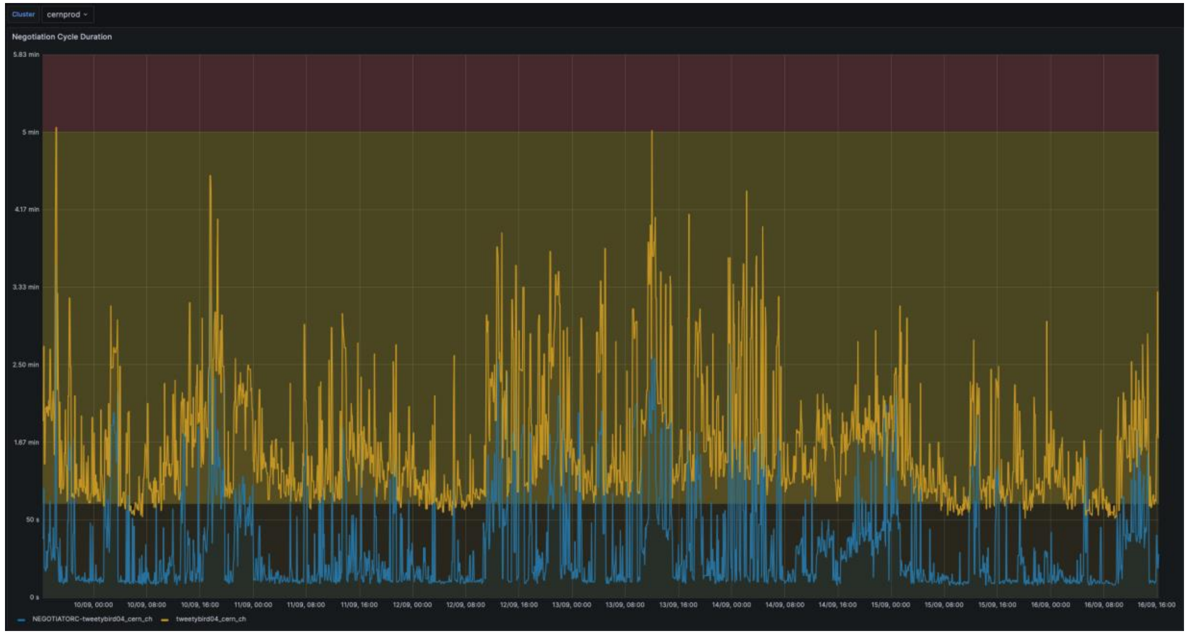


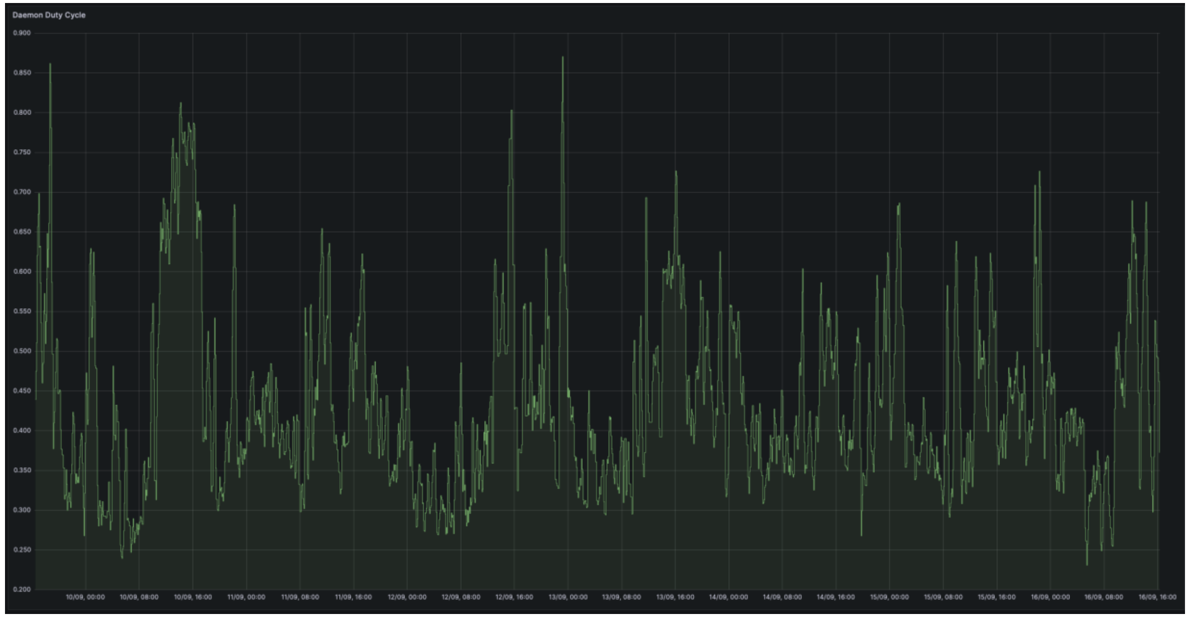
HTCondor @ CERN

- **HTCondor used at CERN since 2016**
- **Used for “high throughput” workload. For HPC workload (read: code that runs on multiple computers ie MPI) we use SLURM**
- **We use htcondor CE as the “Access Point” or “Compute Element” for grid jobs**
  - https://htcondor.org/htcondor-ce/overview/
  - For us: easier to have same middleware provider for both (though others in WLCG do use ARC)
- **European community very HEP focused**
  - Workshop next week: https://indico.cern.ch/event/1386170/
- **Upstream & user mailing list responsive**
  - https://lists.cs.wisc.edu/mailman/listinfo/htcondor-users

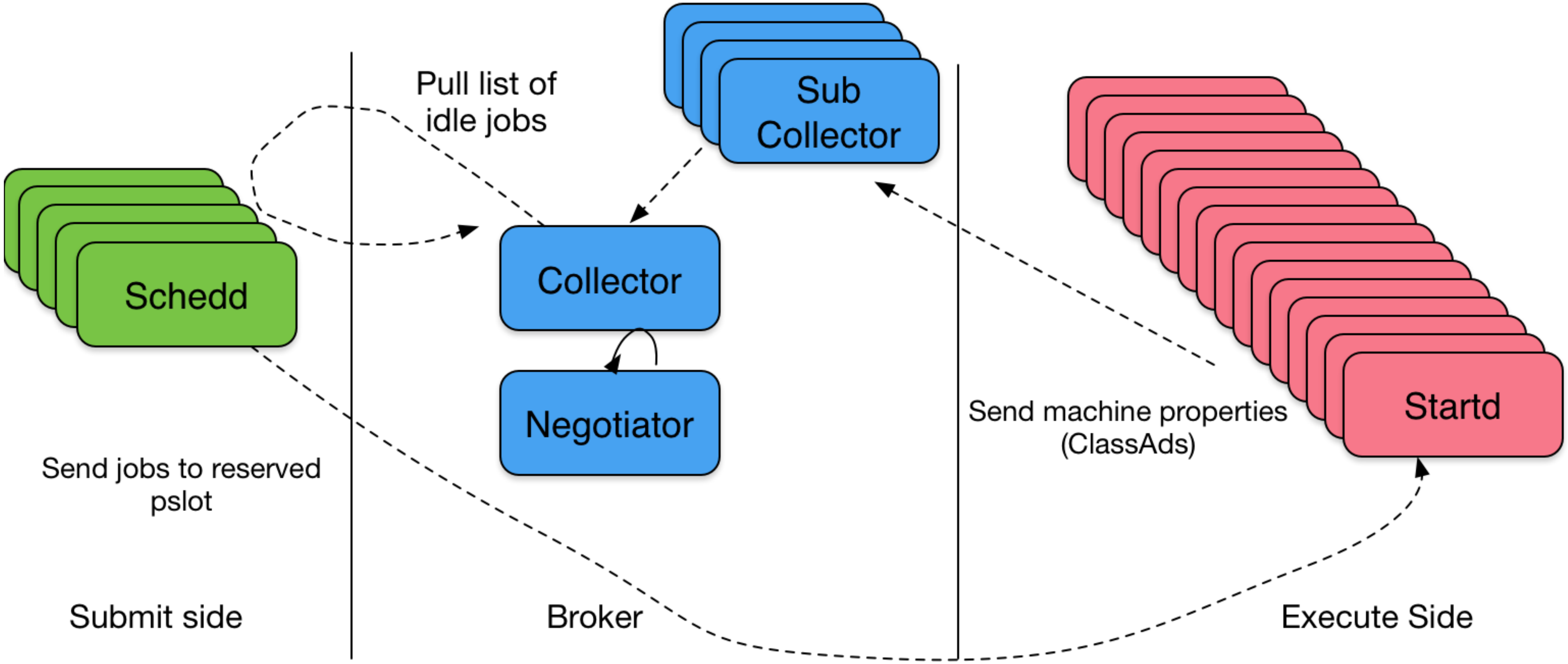
Central Managers

- Lots of effort spent scaling, unlikely to be a problem for smaller pools
    - CMS global pool & (to lesser extent) CERN encounter most of the issues
- Key metrics:
    - Negotiation cycle time: how long it takes for each cycle to match jobs with open slots.
        - aim 3-5 mins, now v easy with threads, we run `NEGOTIATOR_NUM_THREADS = 8`
    - DutyCycle of collectors, if it hits 1, the collector is missing updates
        - We have “sub collectors” reporting into a top level colllectors, but this is probably only necessary after 1000 startds reporting
- No real state to worrry about

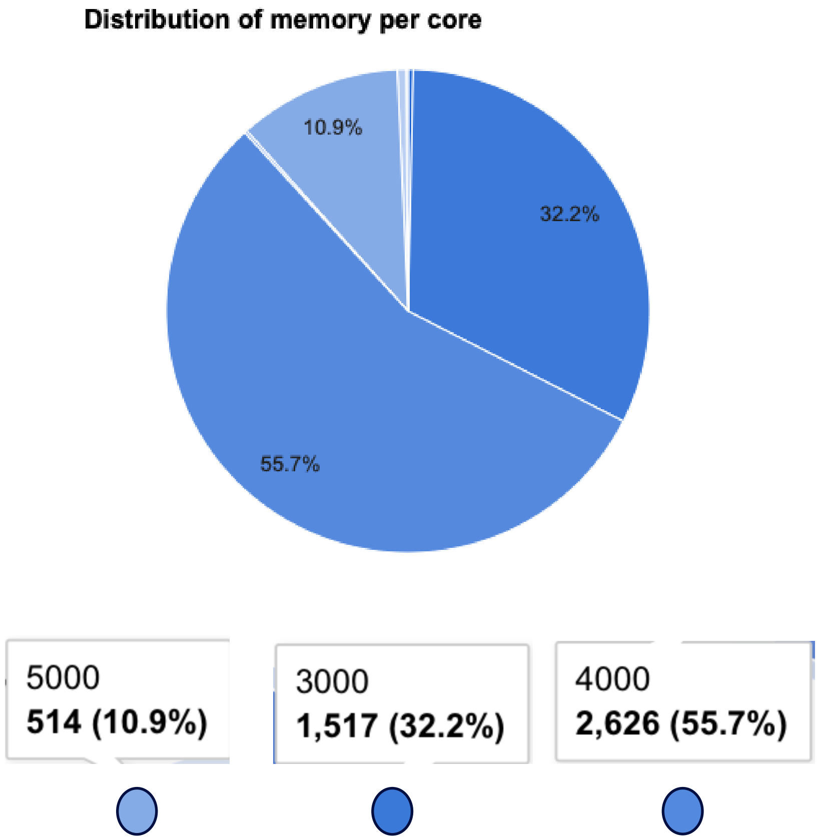




Sub Collectors



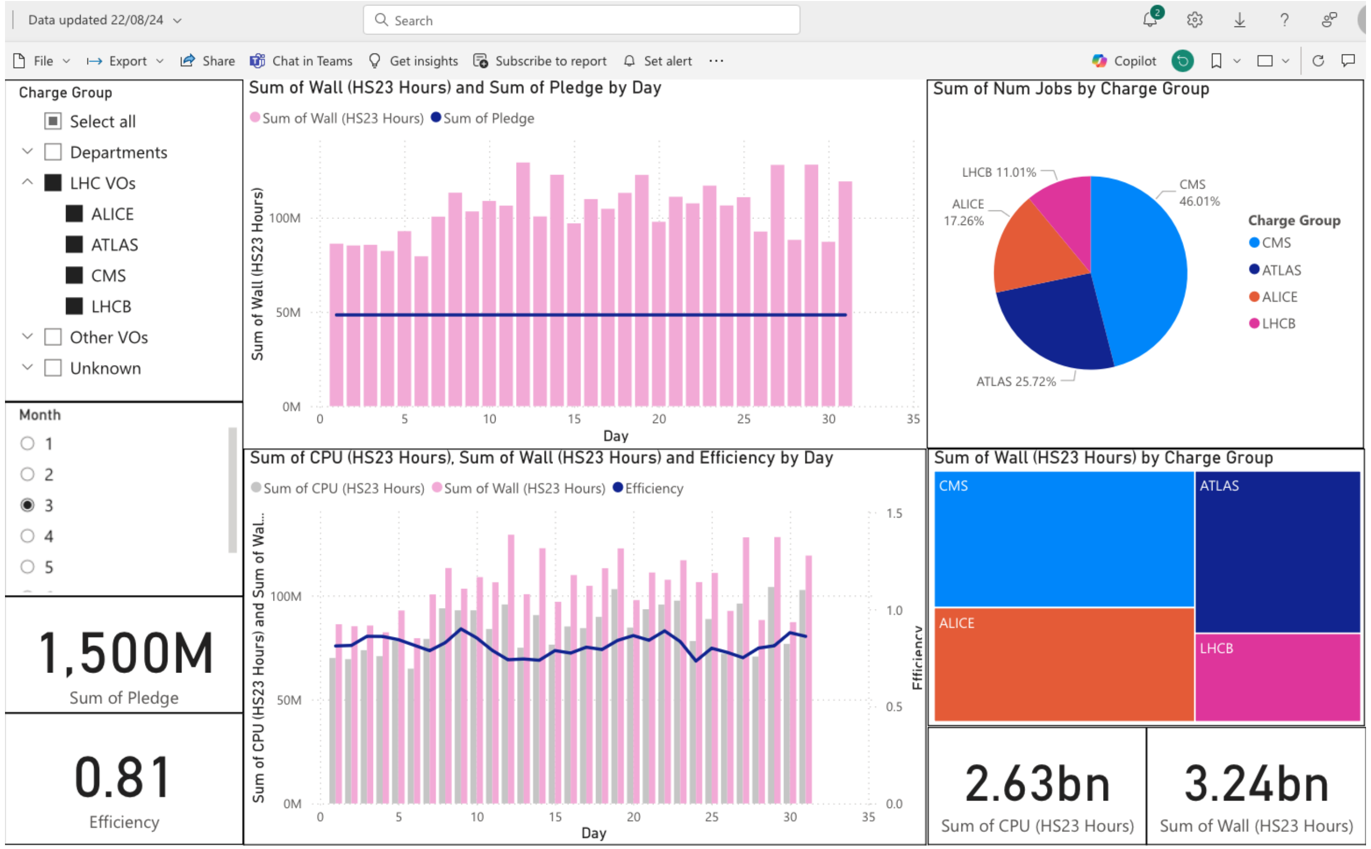
Batch workers



- The WLCG standard is still (I believe):
  - 2Gb memory per core + 20GiB disk per core
- We are at (at least) 3Gb per core + 30GiB of disk
- Mix of older Intel (around 11.5 HS / core) and newer AMD EPYC v3 (16 HS / core)
- We have some aarch64 (around 2k cores) but only ATLAS ready for production at this point
- HTCondor uses CGroups, cpu shares/weights v easy
  - Cgroupv2 for memory (currently) more of a challenge
    - Though easier the more homogenous workflow

Cluster health – efficiency

- **CPU Efficiency (cpu / wall)**
  - Job Type is expected to have different efficiency (ie Simulation > Reco > analysis)
  - Opaque to sites with Pilot Jobs
  - Efficiency could point at other scale / capacity issues
    - Network? Has been “free” till now, do we have easy way to correlate?
  - Efficiency is calculated from accounting records so can be affected by reporting issues from htcondor version
    - We check accounting data for unusual efficiencies
    - Cross checks with efficiency from monitoring



Data updated 22/08/24

Charge Group
- ▣ Select all
- □ Departments
- ■ LHC VOs
  - ■ ALICE
  - ■ ATLAS
  - ■ CMS
  - ■ LHCB
- □ Other VOs
- □ Unknown

Month
- ○ 1
- ○ 2
- ● 3
- ○ 4
- ○ 5

1,500M
Sum of Pledge

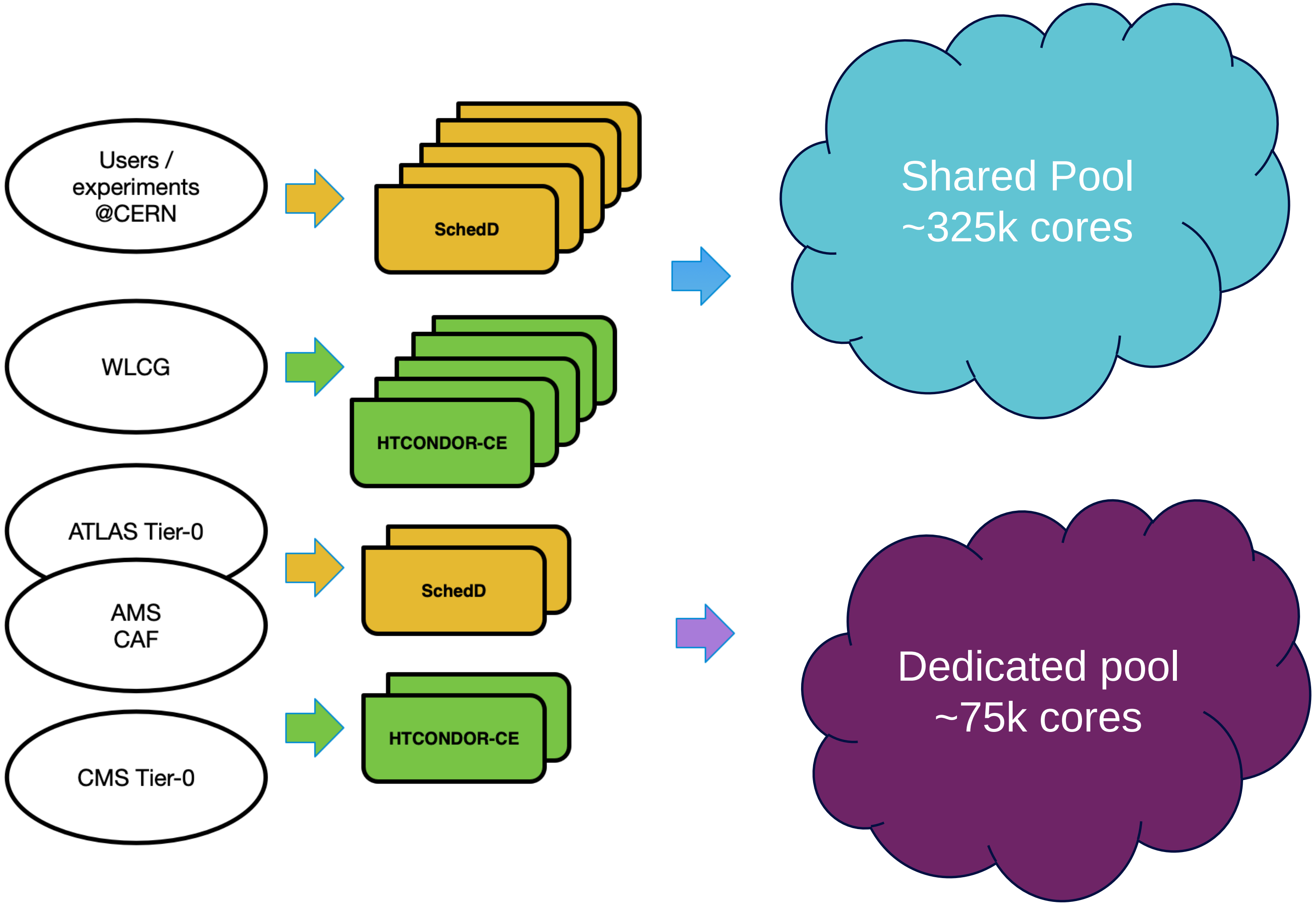
0.81
Efficiency

2.63bn
Sum of CPU (HS23 Hours)

3.24bn
Sum of Wall (HS23 Hours)

CEs

- We have 2 separate HTCondor pools
- “Share”:
  - All jobs can run on all machines
  - Quotas fairshared
- “Dedicated”
  - Machines are dedicated to specific experiments
  - CMS machines only take CMS jobs for example
- Other than scale, some beneits
  - On share we need to have standards for multicore (ie 8 core jobs) to ensure we can defrag appropriately
  - On dedidated, CMS use “whole node” pilots, which reduces # of jobs for us, more flexible for CMS



Scaling CEs (or any other schedd)

- **In HTCondor, the condor_schedd process manages the job queue**
- **CEs or schedds are horizontally scalable**
  - Increasing does increase load on collector / negotiator
  - We have 18 CEs and 20 schedds
- **Scale is down to the "condor_shadow" size**
  - Every running job has its shadow on the submitting schedd/CE
  - Roughly 500kb for a shadow (or closer to 1mb for a shadow with Kerberos)
  - We use VMs and more or less aim for 10k running jobs
  - Could use fewer, bigger machines, it's more about manageability than anything else
- **Token authentication for the Ces**
  - Mapping for token IDs via /etc/condor-ce/mapfiles.d/10-scitokens.conf

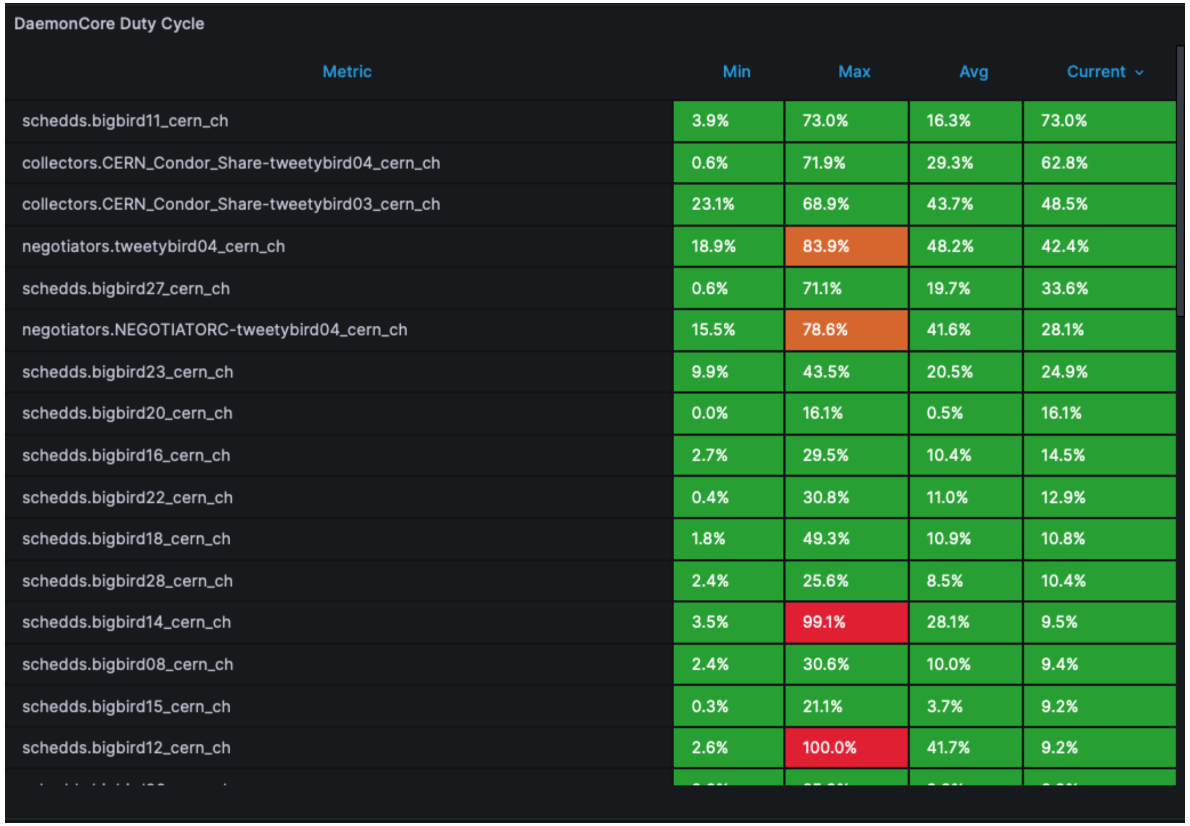
Pool authentication

- **We use kerberos for pool auth**

```
KERBEROS "host/b9g00p4763.cern.ch@CERN.CH" worker-node@cern.ch
```

- **Probably not what I’d do unless I had a pre existing Active Directory / kerberos setup**
  - HTCondor IDTokens are probably what similar sites to yours would do
- **We have previously used GSI (ie SSL based)**
  - Again, based on our pre-existing infrastructure, in this case “grid certificates”
- **Password authentication easiest, but IDToken an enhancement**
- **CERN not the best examplar as we have lots of pre-existing infra**

Monitoring



DaemonCore Duty Cycle

Metric	Min	Max	Avg	Current ⌄
schedds.bigbird11_cern_ch	3.9%	73.0%	16.3%	73.0%
collectors.CERN_Condor_Share-tweetybird04_cern_ch	0.6%	71.9%	29.3%	62.8%
collectors.CERN_Condor_Share-tweetybird03_cern_ch	23.1%	68.9%	43.7%	48.5%
negotiators.tweetybird04_cern_ch	18.9%	83.9%	48.2%	42.4%
schedds.bigbird27_cern_ch	0.6%	71.1%	19.7%	33.6%
negotiators.NEGOTIATORC-tweetybird04_cern_ch	15.5%	78.6%	41.6%	28.1%
schedds.bigbird23_cern_ch	9.9%	43.5%	20.5%	24.9%
schedds.bigbird20_cern_ch	0.0%	16.1%	0.5%	16.1%
schedds.bigbird16_cern_ch	2.7%	29.5%	10.4%	14.5%
schedds.bigbird22_cern_ch	0.4%	30.8%	11.0%	12.9%
schedds.bigbird18_cern_ch	1.8%	49.3%	10.9%	10.8%
schedds.bigbird28_cern_ch	2.4%	25.6%	8.5%	10.4%
schedds.bigbird14_cern_ch	3.5%	99.1%	28.1%	9.5%
schedds.bigbird08_cern_ch	2.4%	30.6%	10.0%	9.4%
schedds.bigbird15_cern_ch	0.3%	21.1%	3.7%	9.2%
schedds.bigbird12_cern_ch	2.6%	100.0%	41.7%	9.2%
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

- **HTCondor expose lots of metrics from various daemons**
  - This is the famous “DutyCycle” which is the most obvious metric for a busy daemon
- **HTCondor has python bindings, lots of monitoring (including ours) use python to push metrics to be displayed in Grafana**
  - https://htcondor.readthedocs.io/en/lts/apis/python-bindings/index.html
- **For more ”plug & play” there’s condor_gangliad**
  - https://htcondor.readthedocs.io/en/lts/admin-manual/monitoring.html
  - We don’t use ganglia (at least not for this)

Less standard use of htcondor

- **HTCondor is very flexible, can take advantage of opportunistic resources**
- **Other things we do (or have done) with HTCondor:**
  - Backfill SLURM slots via condor_gridmanager
  - Run on public cloud resources
    - Both with VPNs and also across firewalls
    - condor_annexe also exists, but is not our usecase
  - Run on short term preemptible resources
  - Run workers on kubernetes
  - Use DASK metascheduler to run DASK workers as HTCondor ”jobs”
  - Run htcondor jobs in containers on fileservers
  - [I must be missing other examples]

home.cern