CERN

Scaling HTCondor at CERN

Ben Jones

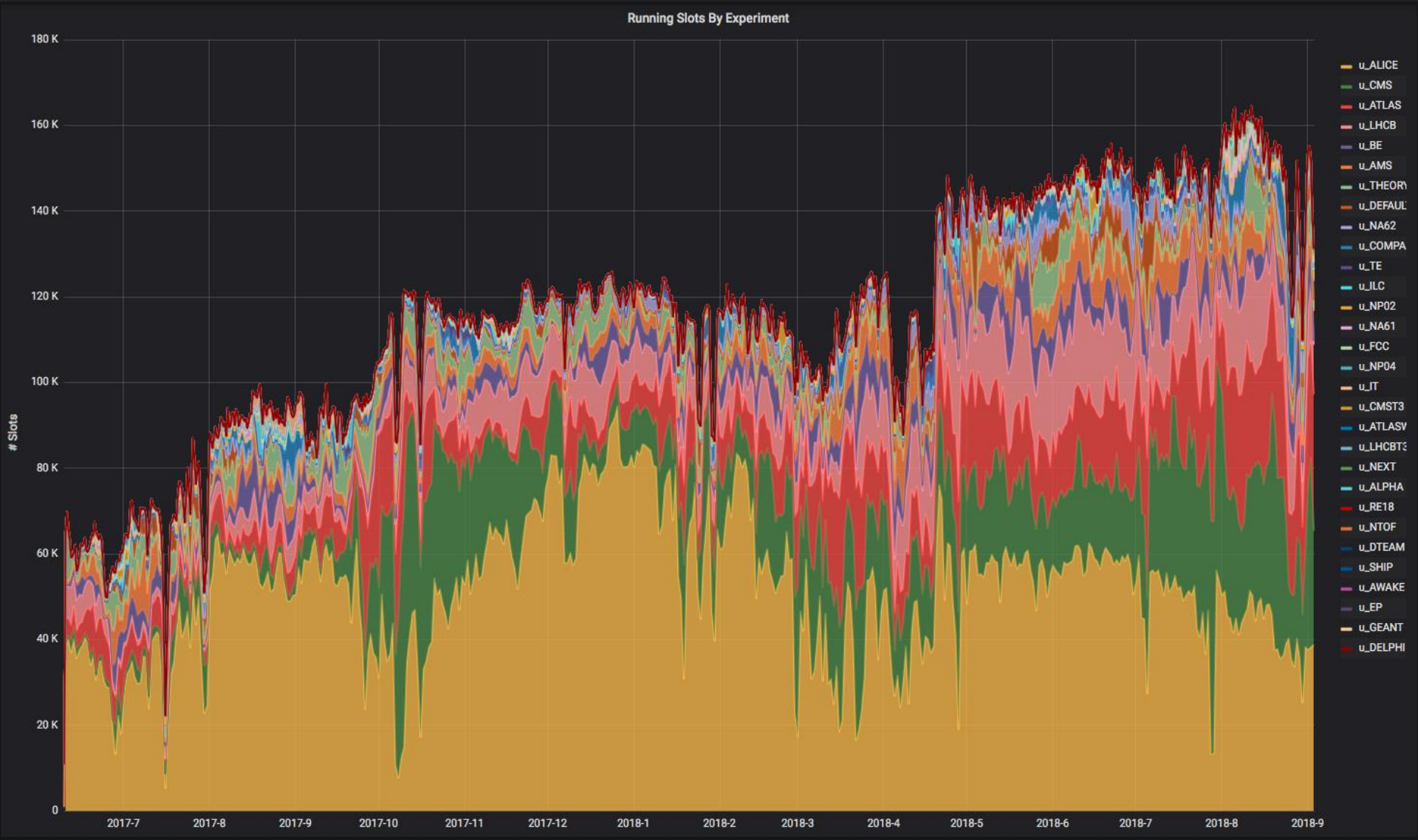
HTCondor at CERN

- Batch service used by LHC & related experiments via grid, and local users mainly via shell
- Were using Platform/IBM LSF, decided to migrate away due to scale (+proprietary) concerns, eventually to HTCondor
- In production since 2015 Q4, first for grid with ~10K cores, then local in 2016.

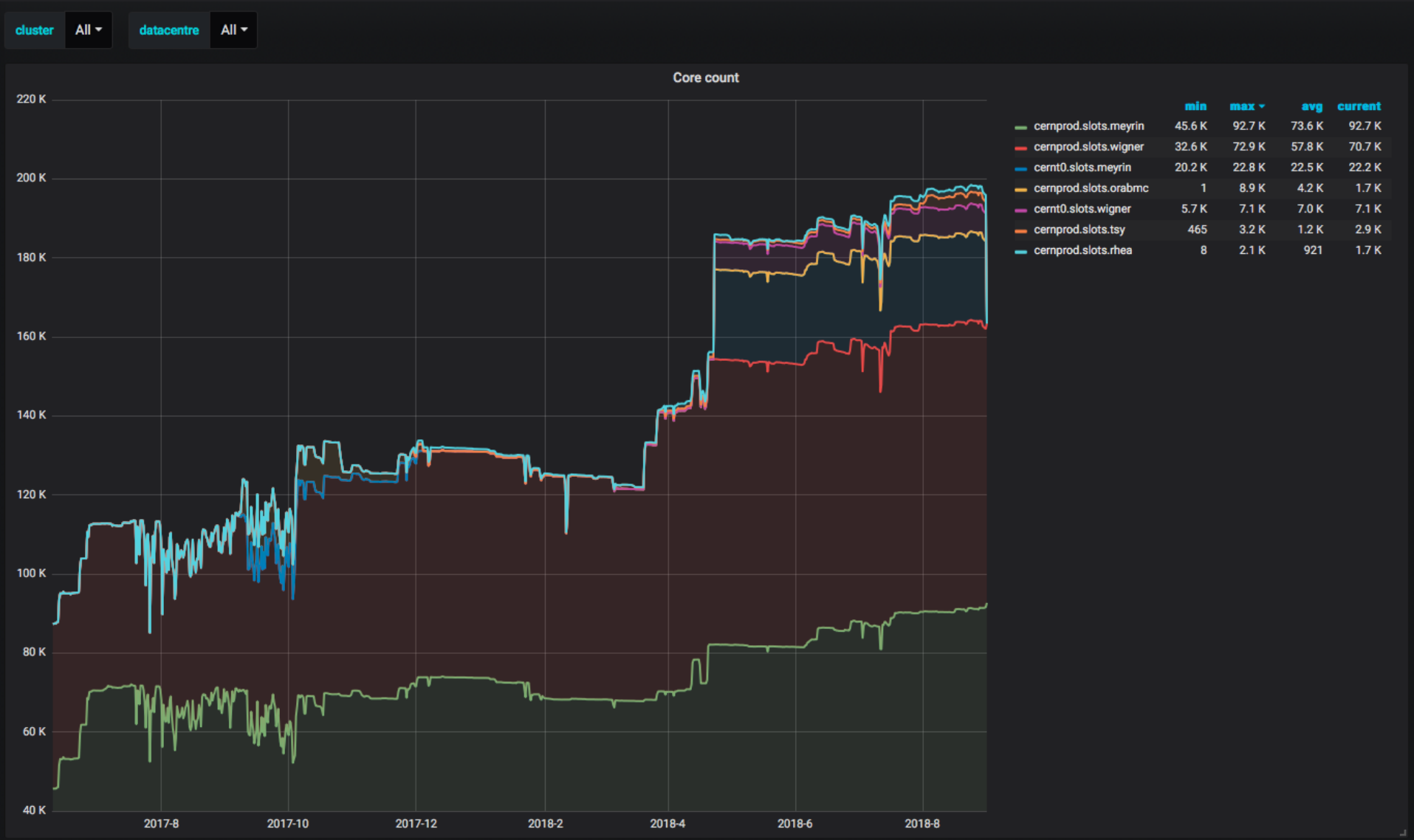
What have we had to scale?

- Increase in the number of nodes
- Additional pools
- Scaling the central infrastructure
- The number of users and groups now using the system
- The different use cases HTCondor is being asked to manage

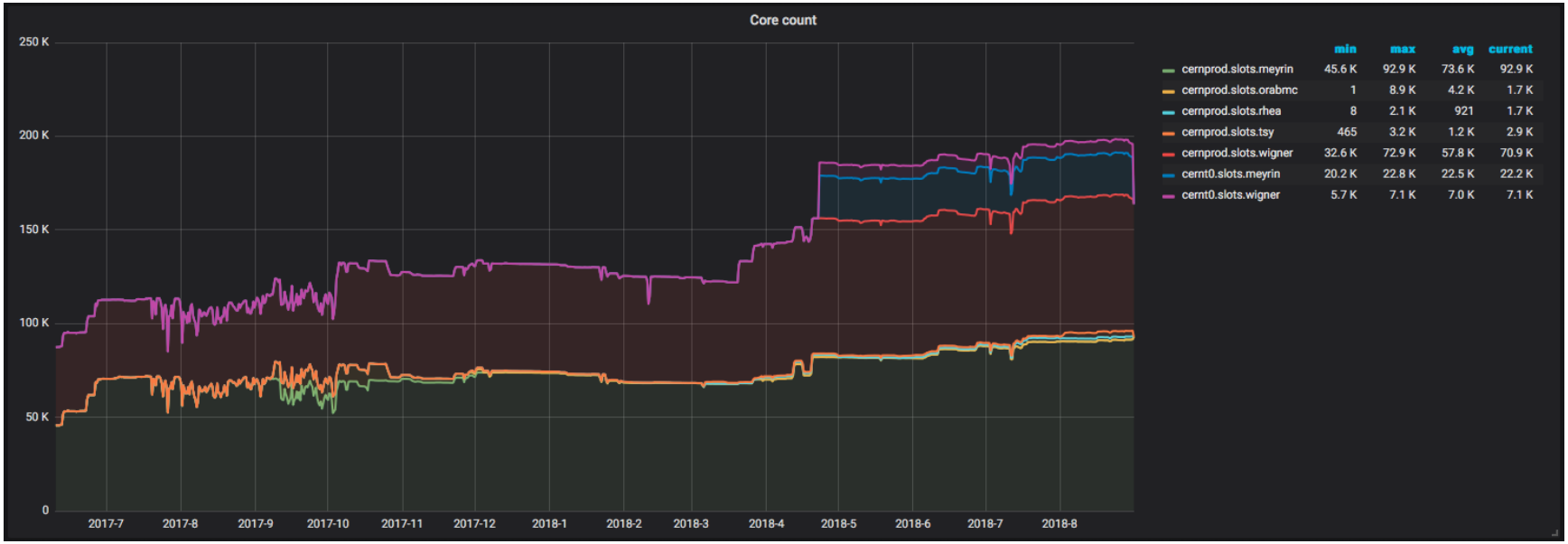
Growth since last workshop (jobs in share)



Growth since last workshop (slots)

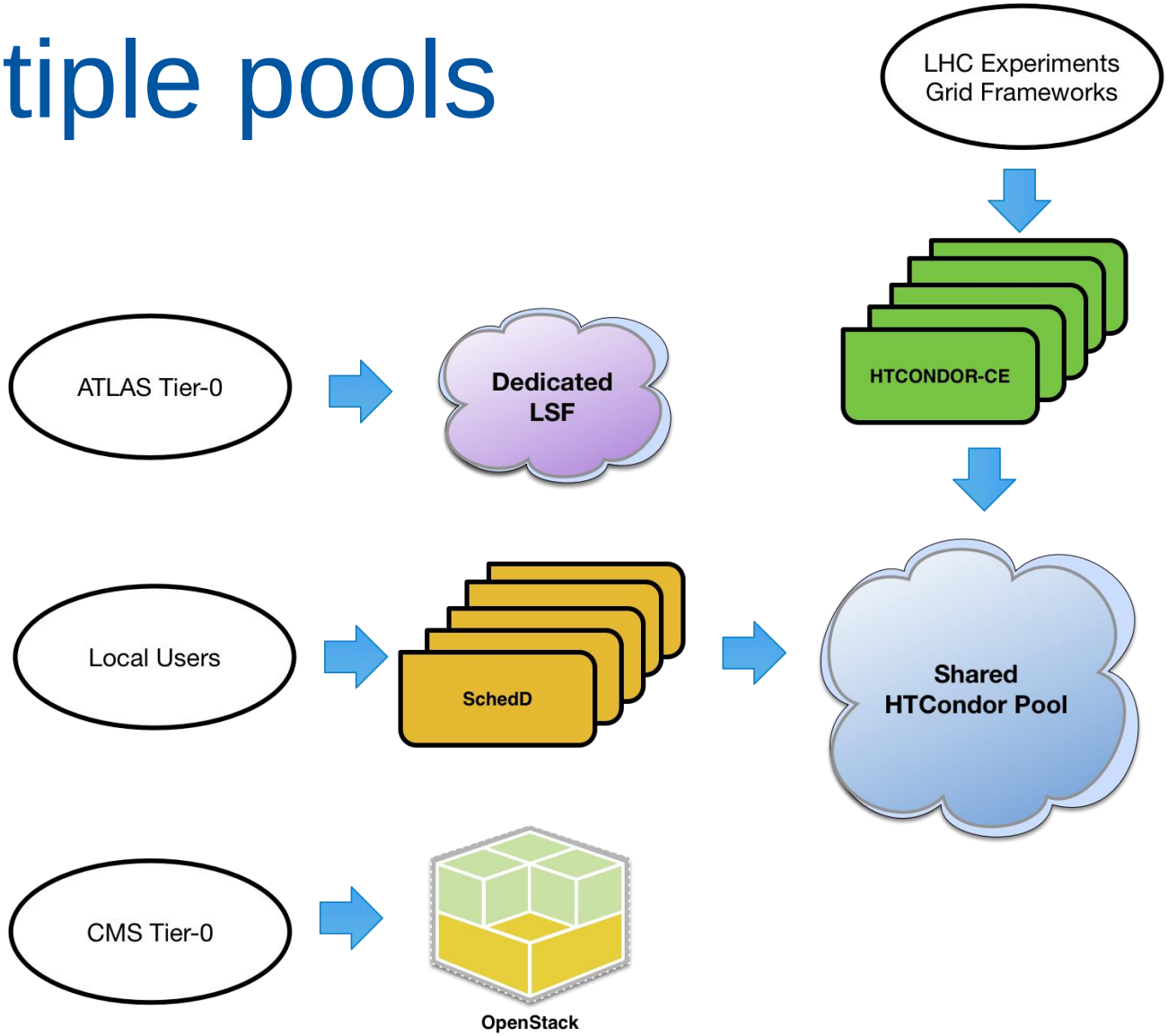


Pools & Datacentres

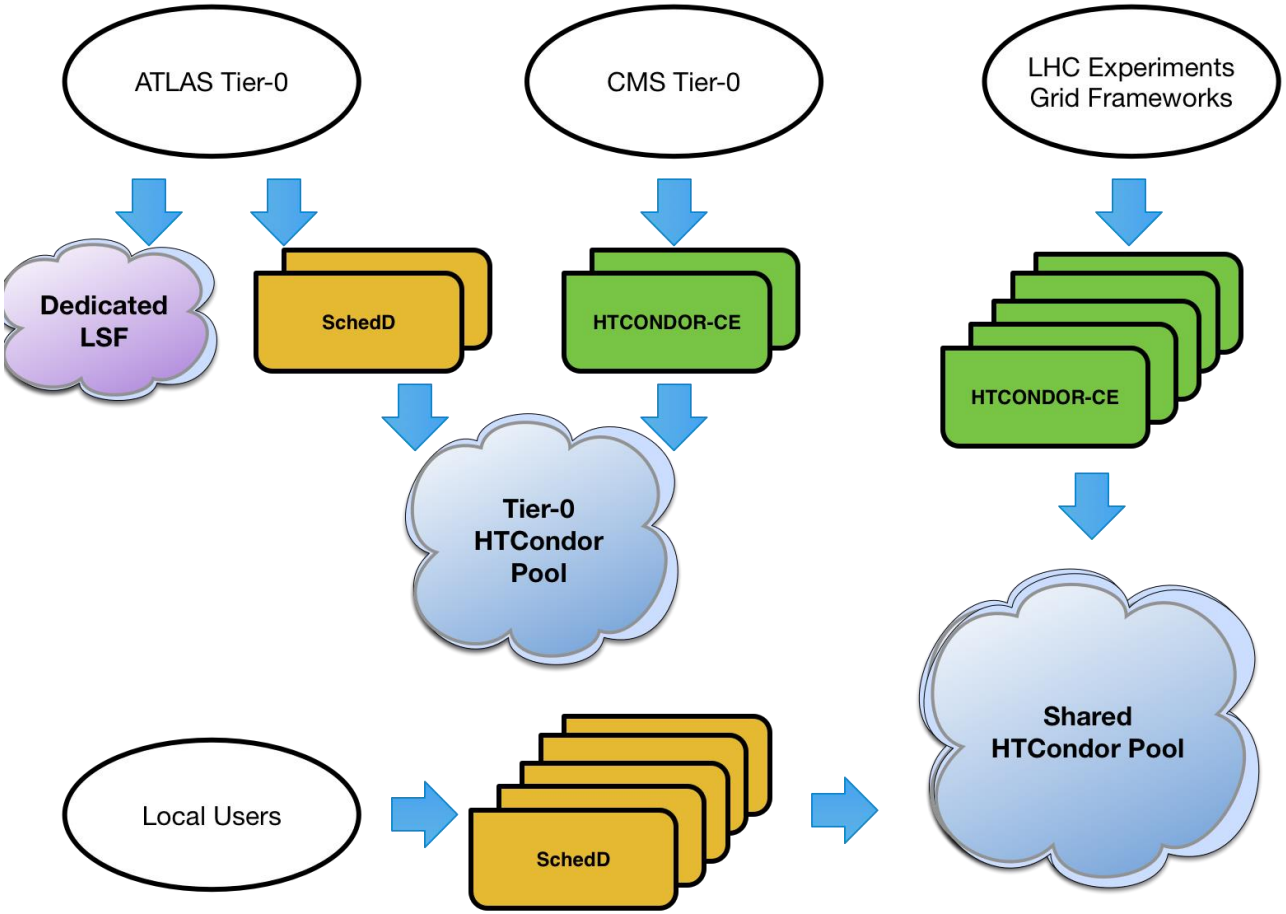


- Almost at 200K Cores
- Multiple Pools
  - ”cernprod” general multipurpose pool
  - “cernt0” for ATLAS & CMS Tier-0 function
- Multiple Datacentres
  - Not just CERN sites of Meyrin & Wigner
  - Cloud sites: tsy, rhea, orabmc

Multiple pools



Tier-0 pool


ATLAS Tier-0
CMS Tier-0
LHC Experiments
Grid Frameworks
Dedicated
LSF
SchedD
HTCONDOR-CE
HTCONDOR-CE
Tier-0
HTCondor
Pool
Shared
HTCondor Pool
Local Users
SchedD

Infrastructure by numbers

- **Share**
- Managers 2x 40 core 128GB
  - 32 collectors
  - 3 negotiators
- 10 user schedds, 15 CEs (8 core, 16GB) 10K running jobs
- 17.5K StartDs, 171K cores

- **Tier-0**
- Managers 2x 16 core 32GB
  - 1 collector
  - 1 negotiator
- 1 user Schedd (16 core 32GB) 20K running jobs
- 2 CEs (8 core, 16GB) 10K running jobs
- 2.25K StartDs, 28K cores

Scaling Collector

- Following the CMS recipe (thanks!)
- The “Brian ratio” of about 5K cores per collector
  - Tier-0 almost entirely full-node, so far 1 collector enough
- CLASSAD_LIFETIME set to 45 minutes (our startds are long lived)
- COLLECTOR_FORWARD_FILTERING
  - execptions for State, Cpus, Memory, IdleJobs, CERN_CREDMON_VER
  - we do still filter ”activity” as again: currently long lived startds

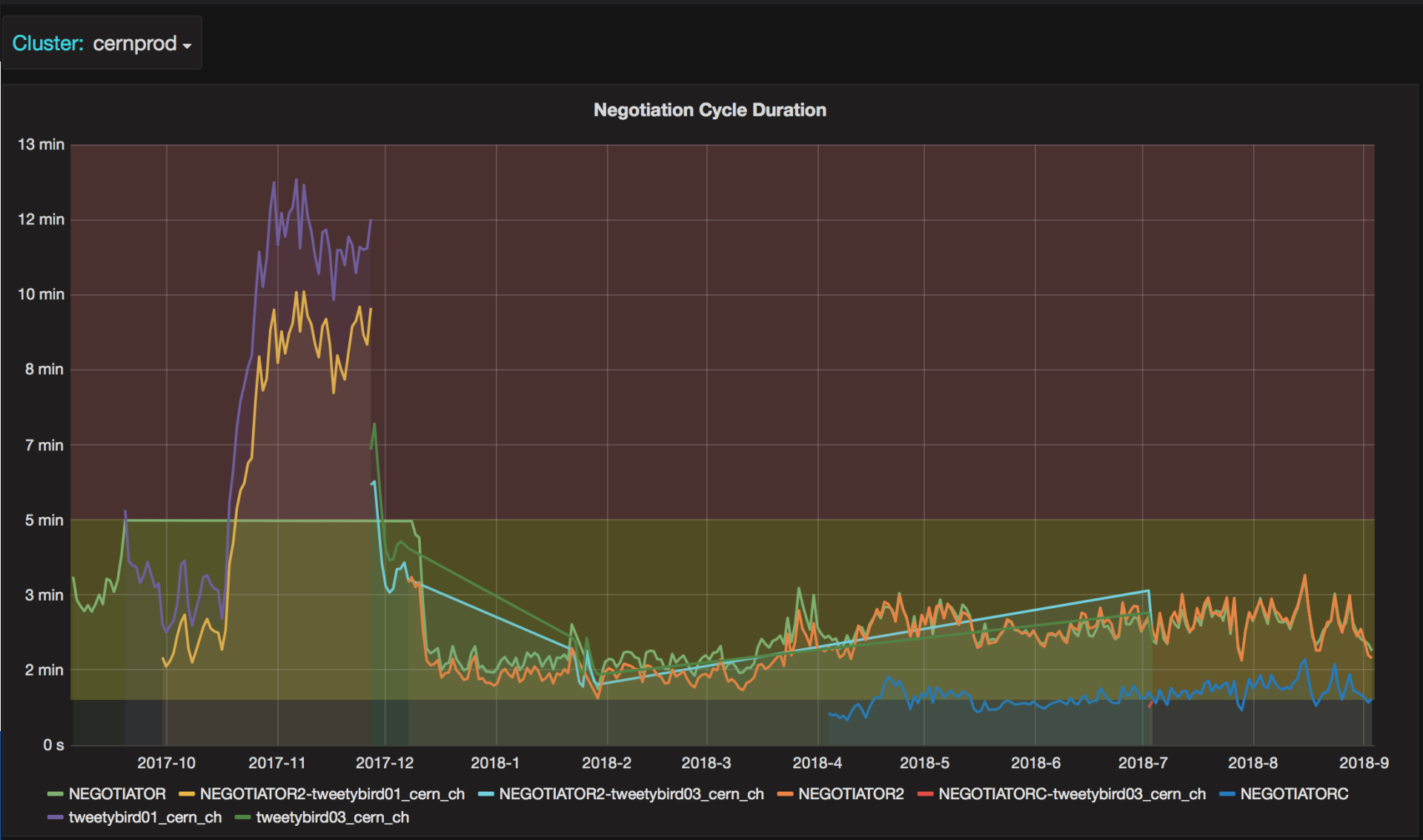
Negotiator

- Biggest impact on the Negotiator was by reducing the load on the top level Collector
- Split the negotiator in two for the main share pool
  - CERN uses puppet/foreman “hostgroups” to group machines
  - Use puppet hiera in the hostgroup to populate ClassAd to assign to a Negotiator

```
NEGOTIATOR_SLOT_CONSTRAINT=CernNegotiator =!= "2" && HNSci =!= True &&
BeerMachine =!= True
NEGOTIATOR2.NEGOTIATOR_SLOT_CONSTRAINT=CernNegotiator =?= "2" && HNSci
=!= True && BeerMachine =!= True
NEGOTIATORC.NEGOTIATOR_SLOT_CONSTRAINT=HNSci =?= True || BeerMachine =?=
True
```

- Watch Negotiator cycle to keep < 5 minutes

Guess where we made changes…



Users to Schedds

- LXPLUS is the interactive logon service for CERN
- We don’t run schedds on LXPLUS
  - Pool of LXPLUS servers with round robin DNS
  - LXPLUS are deleted / rebuilt frequently
- Users need to be mapped to **remote** schedds
  - LOCAL_CONFIG_FILE with call to a map
  - Map currently a call to zookeeper

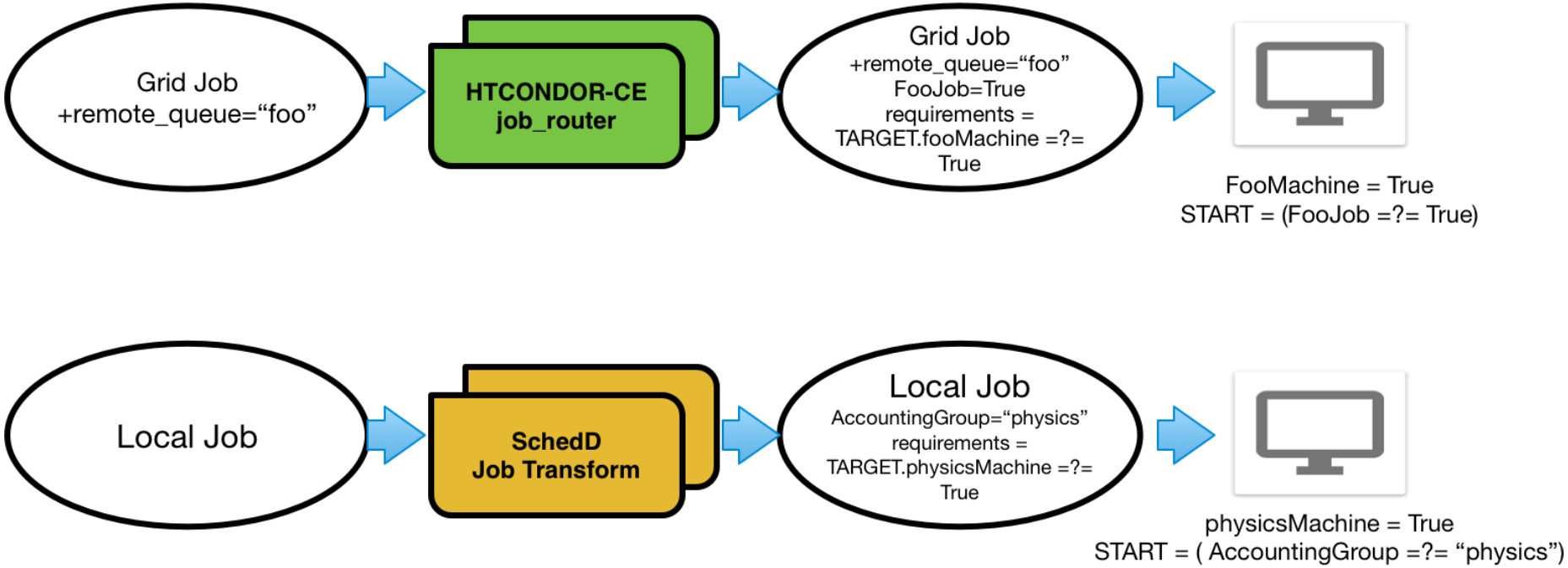
Issues with Users to Schedds

- Even if we had huge schedds rather than VMs, we'd still need to scale horizontally due to # of jobs
  - (we also occasionally see issues with individual users briefly DOSing schedds)
- Users submission rate is uneven and bursty, so we need to rebalance
  - This is a manual process and it requires user involvement if there are still jobs on the previous schedd

Special Resources (CERN anti-pattern?)

- The majority of jobs can run on the majority of the pool(s).
- Why are there exceptions?
  - Special sets of resources bought or promised to specific groups
  - Special projects, ie external cloud or running on disk servers
- We manage special resources by decorating job and machine Ad with attribute that the other side requires

Anti-pattern?



Non-general use cases

- Cloud – adding resources as a separate site/queue/jdl behind CERN CE
  - Now running a ‘WLCG pool’ so that commonly procured Cloud resources can be provided behind a single entry point for the experiments
- BEER
  - “Batch on Eos Extra Resources” – just use spare resources from disk servers to provide slots, using systemd / CGroups to partition resources & Docker to abstract host
- Cloud & BEER require different Negotiator & ability to treat as separate site

Future…

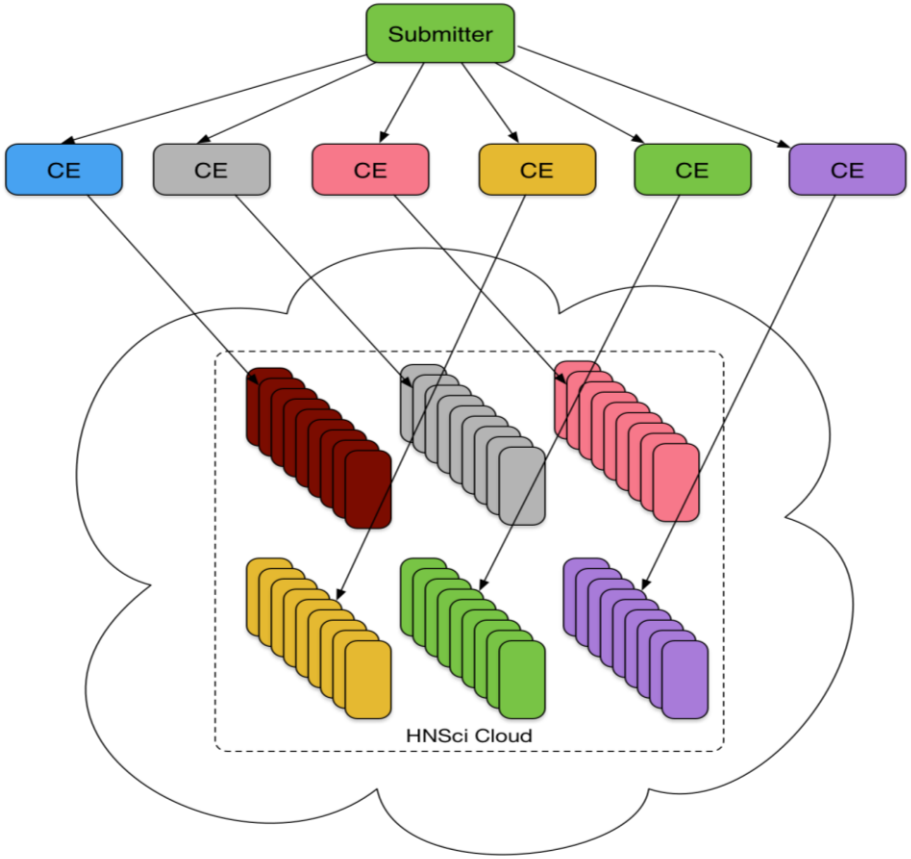
- Ideal: the convenience of cloud APIs, the homogeneity of owning resources, the fairshare, scheduling & use case focus of HTCondor
  - Best bet so far: provisioning layer with kubernetes, including cloud federation, with HTCondor doing the job federation
  - Kubernetes & kube-fed look like they will help fill ”service” areas of datacenter & hetrereogenous clouds easily
  - Internally, kubernetes might help us eliminate virtualization layer whilst keeping cloud management layer

Questions?

backup slides

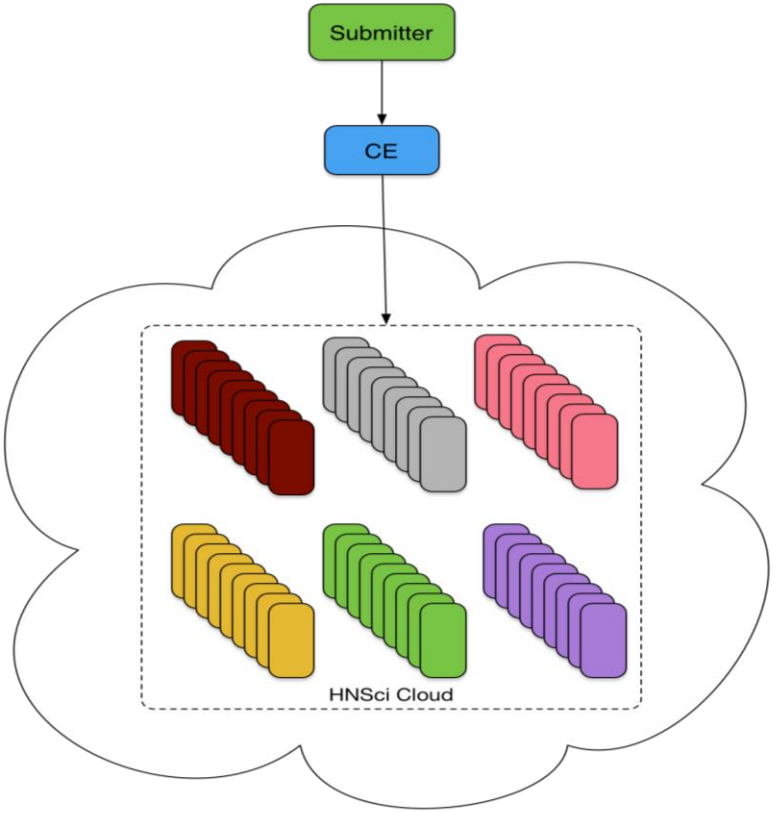
Unconsolidated

- In this setup, an experiment could have to define 8 additional sites to send jobs to the same resources



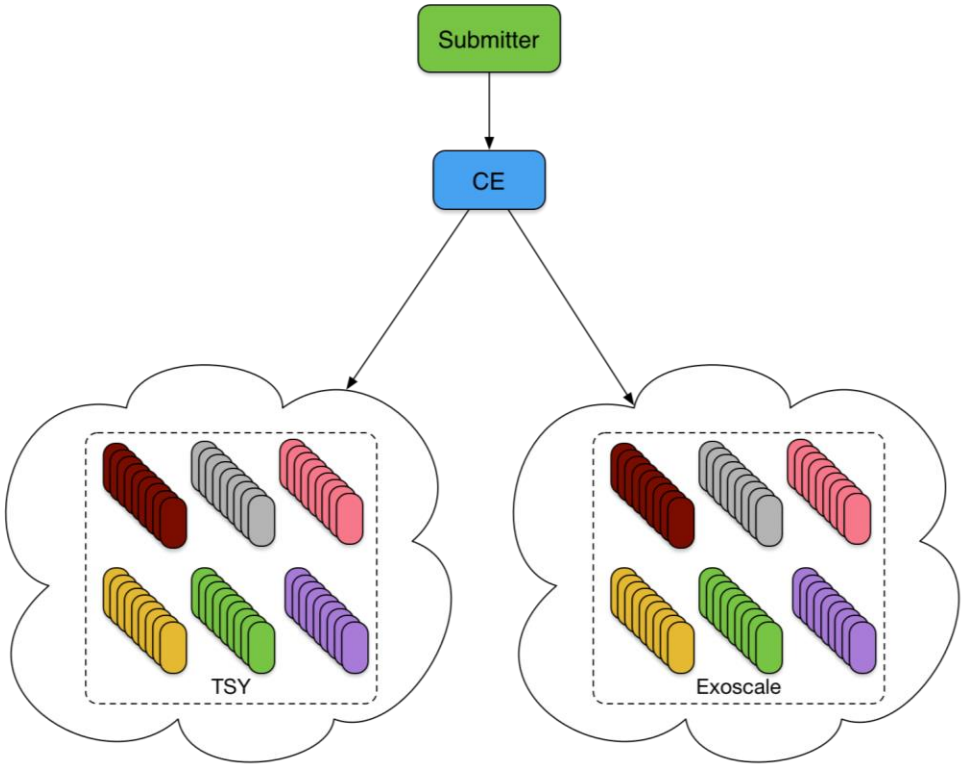
Shared Tenant

- With a shared tenant and single entry point (CE), only one site needs to be setup

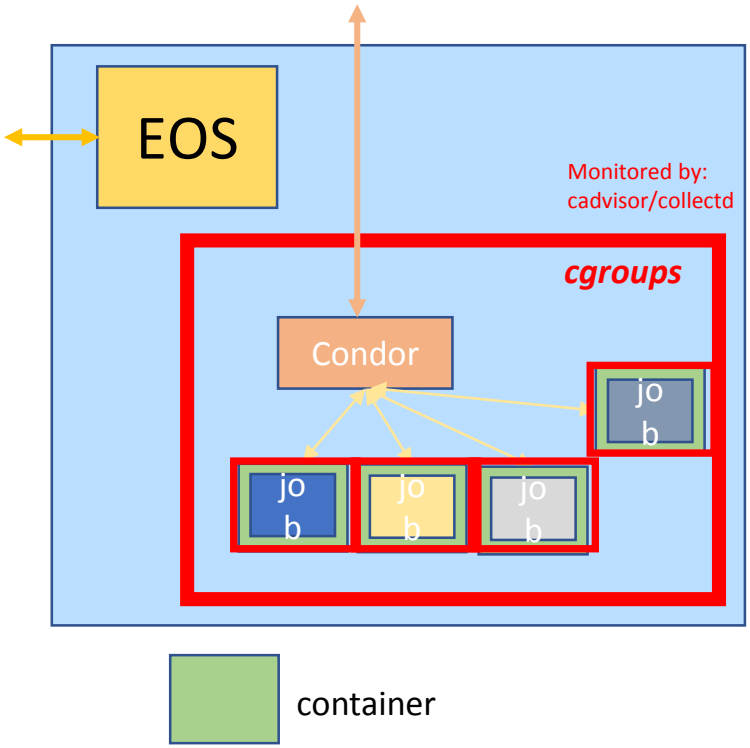


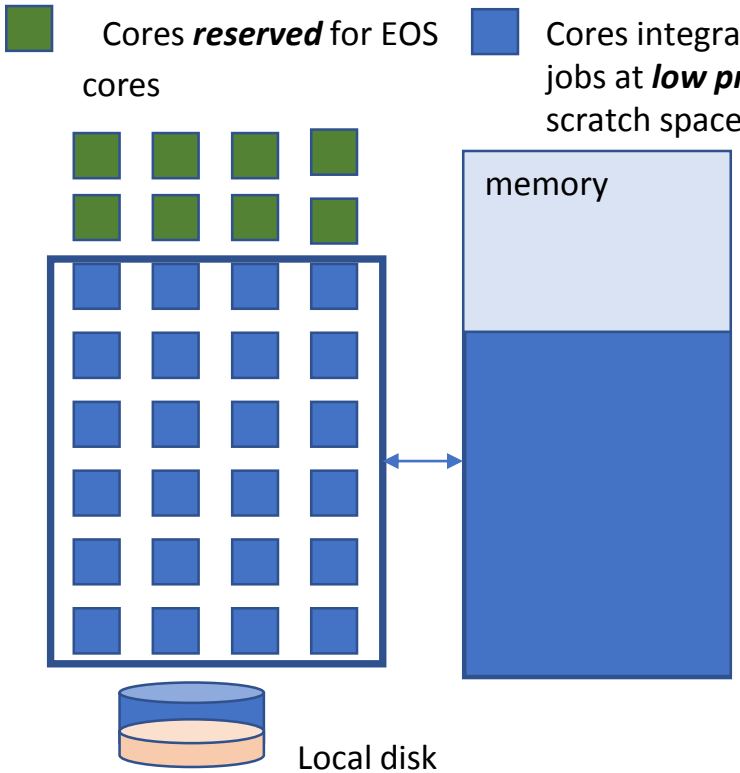
Multiple Clouds

- Current setup being explored at HNSciCloud with a single entry point



Condor + Containers





Proof of Concept III