

Integrating the Perlmutter HPC system in the ALICE Grid

Sergiu Weisz
On behalf of the ALICE collaboration

sergiu.weisz@upb.ro
24/10/2024



Why HPC?

Running Jobs

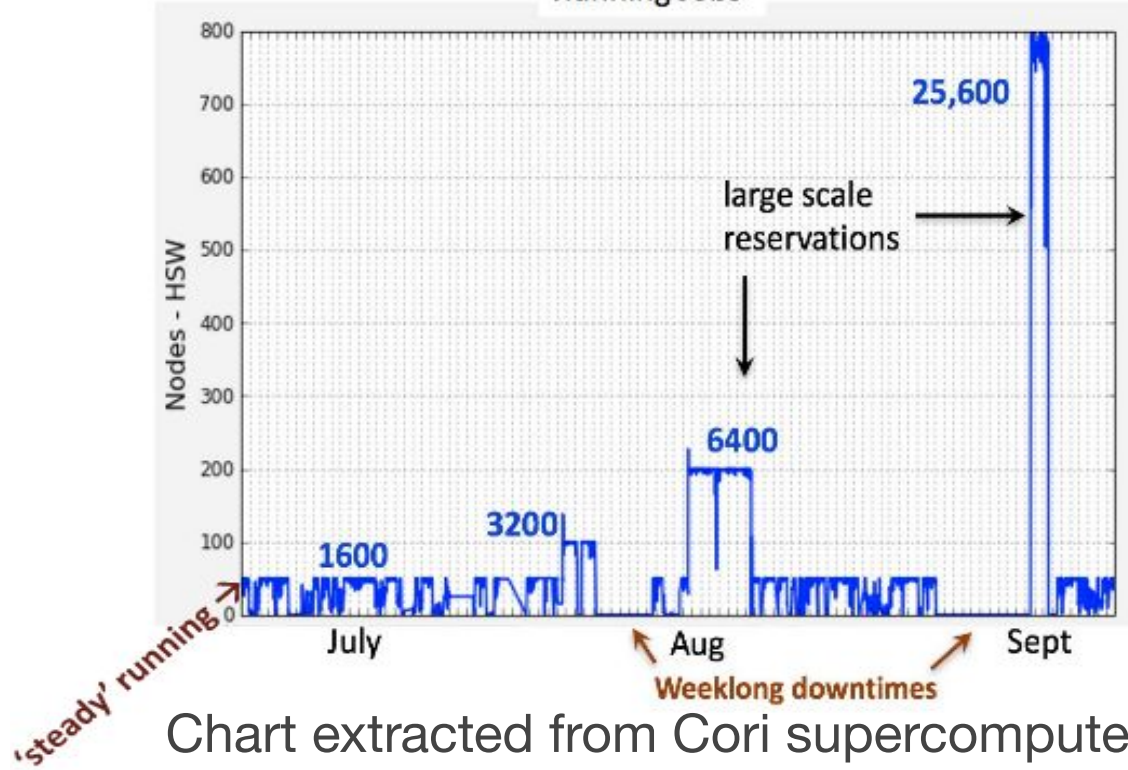
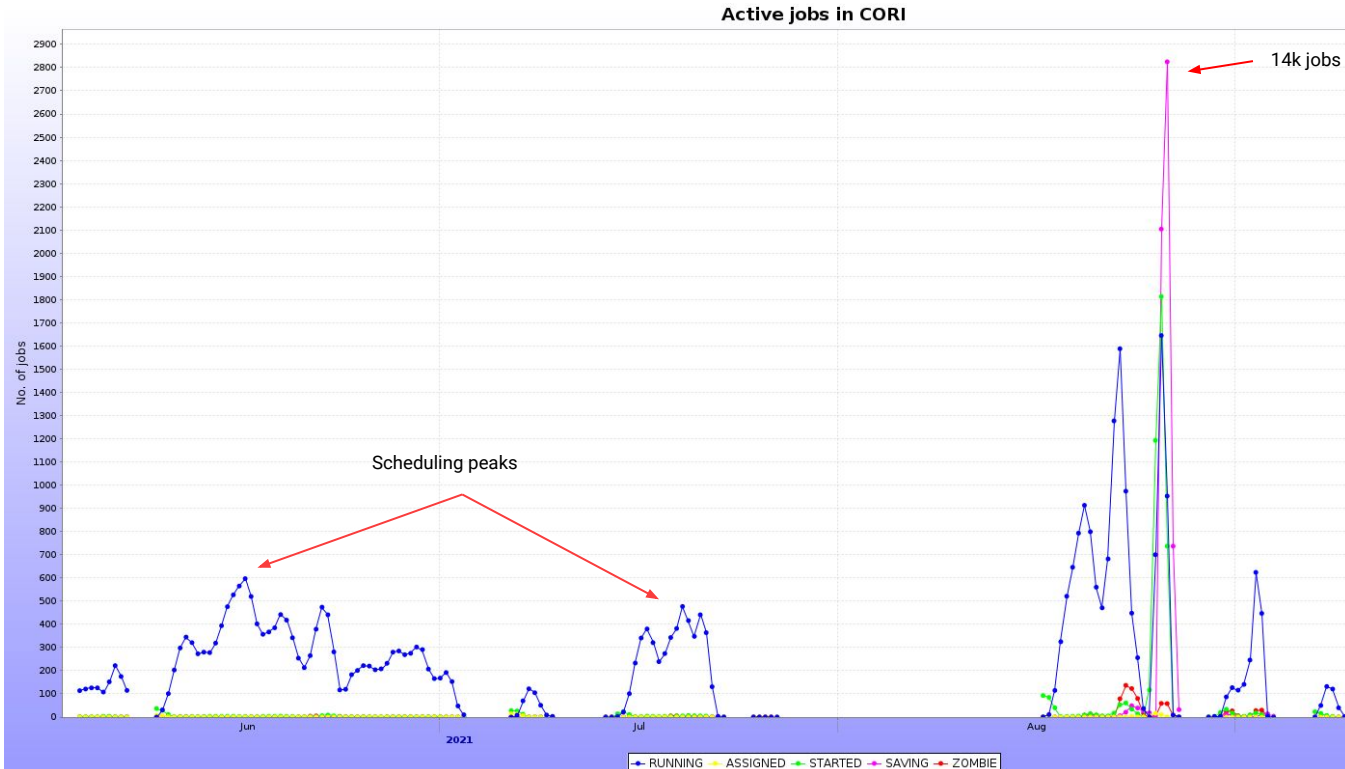


Chart extracted from Cori supercomputer at Lawrence Berkeley National Lab

Running Jobs On Cori



- Most of the time running 100-200 jobs
- Peaks in scheduling, as expected
- Long shutdown while testing networking and job behaviour



High Performance Computing Limitations

- Closed networks
- Heterogeneous architecture meant for data center-wide jobs
- Non-compatible grid authentication
- Limited software distributions
 - Some can't run CVMFS
- No on-node scratch

Cori Supercomputer

- Can communicate to the outside
- Limited bandwidth
 - We require 2MB/s per job
 - Could not ramp up I/O jobs
- Used shifter for containers
- The VObox deployed on “service node” with low uptime
- Used custom JAliEn setup
 - First JAliEn deployment outside of CERN



Meet Perlmutter

- Runs HPE Cray OS
- Only whole-node scheduling
- 3072 nodes running 2 x AMD EPYC 7763, 64 core, 512 GB memory
- Jobs run in Shifter containers
- Outside connection from nodes
- Full CVMFS support





Objectives

- Turn the HPC site into a production system
- Low maintenance
- Steady scheduling
- Burn as many resources as possible



Perlmutter Points of Contention

- Sending jobs to the supercomputer
- Storage bandwidth for I/O bound jobs
- Node time allocation

How do we send jobs to Perlmutter?

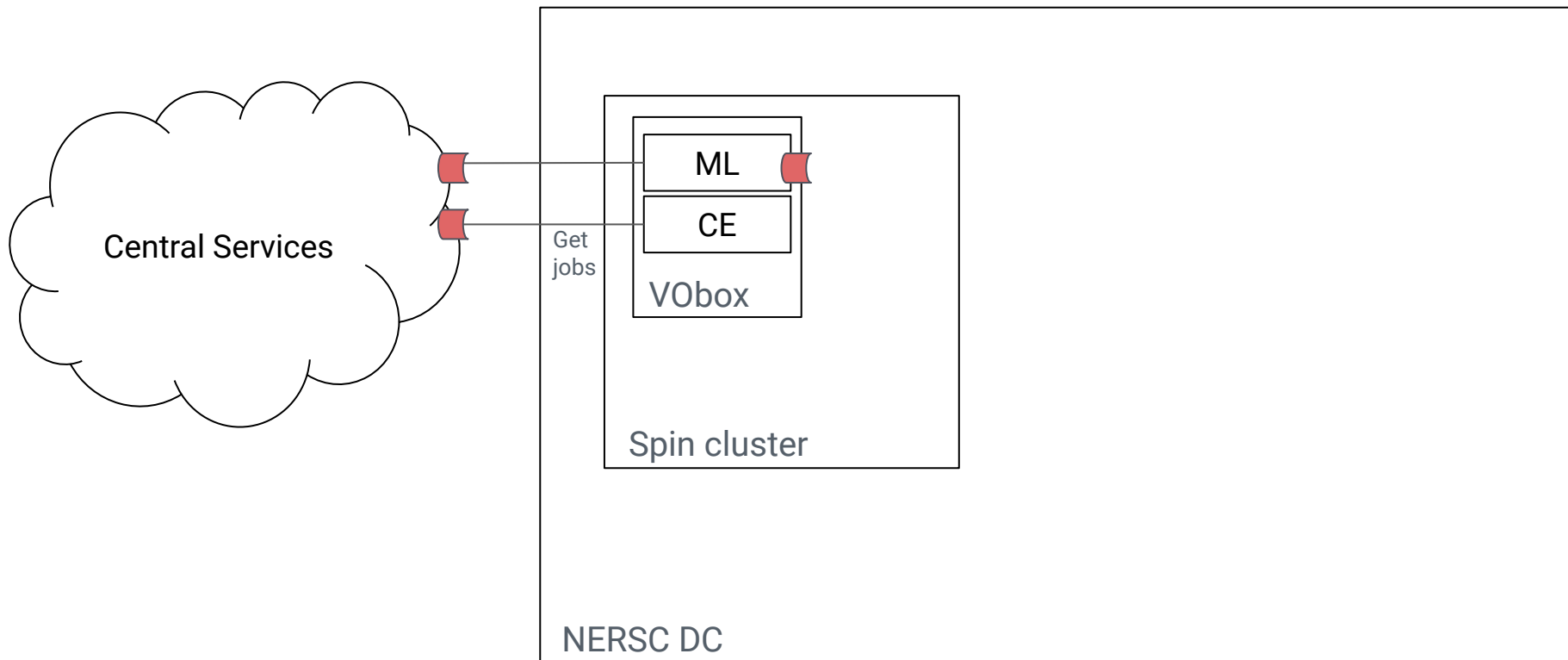
- Internally runs SLURM for job scheduling
- Simple, just run CE on login nodes like Cori!
 - Login nodes don't guarantee uptime
 - Admins recommend running a cron to keep the CE up
- NERSC suggested a new submission API and hosting CE on their Kubernetes cluster

The SuperFacility API

- HTTP based API to manage jobs
- Uses JWT to authenticate
- Initially we were promised an easy security review process for getting an auth token
 - Policy is to receive 30 days job submission privileges
- SFAPI receives SLURM job specifications and sends them to the HPC system

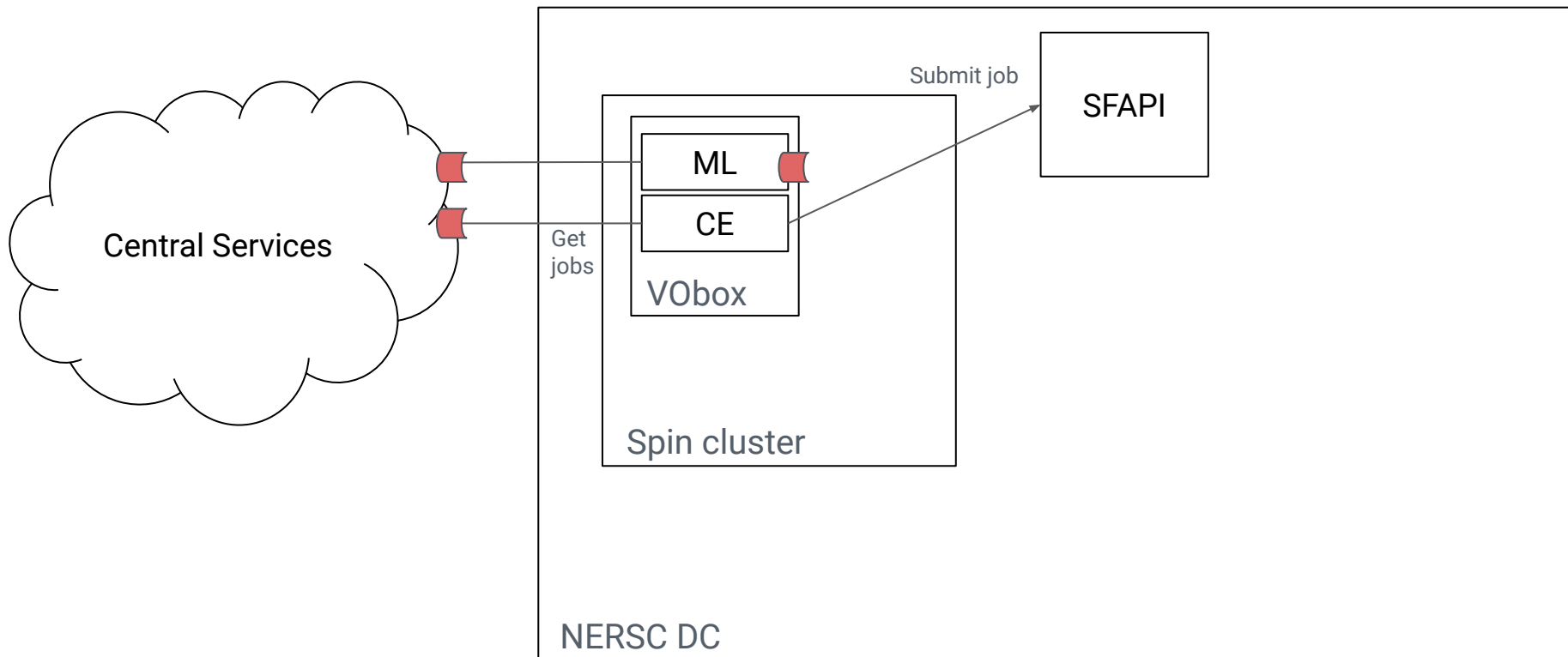


Let's use Kubernetes



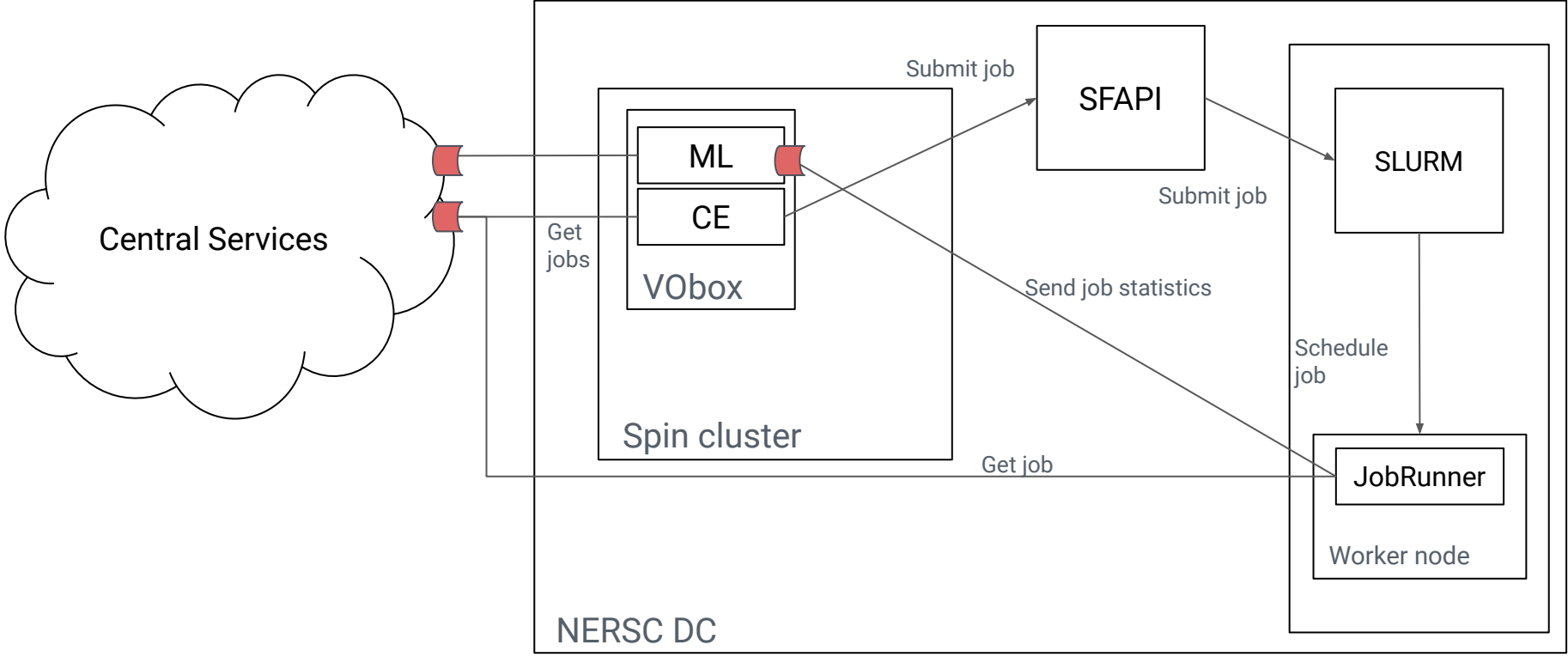


Let's use Kubernetes





Let's use Kubernetes



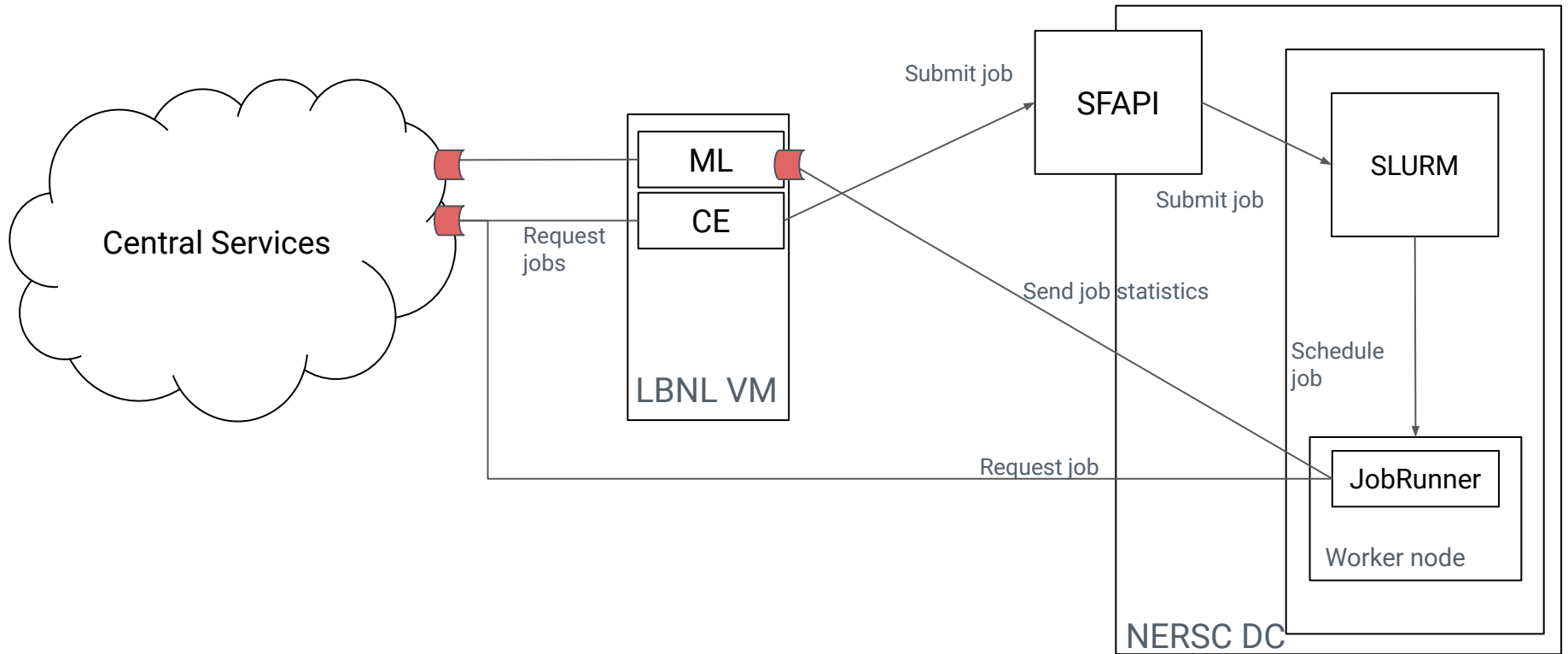


Running CE in unprivileged container

- Adapted the workflow to run inside of container based on CERN solution
- We couldn't get ingress to route our traffic
 - Only certain ports are allowed
- The environment was proving too restrictive for no real benefits



VMs are still better





Move to an LBNL VM

- Requested a VM in the LBNL cluster to run the CE
- Technically hosted in the same institution, different organizations
- Root-less access to the VM
- Software initially running custom JAliEn with SFAPI support



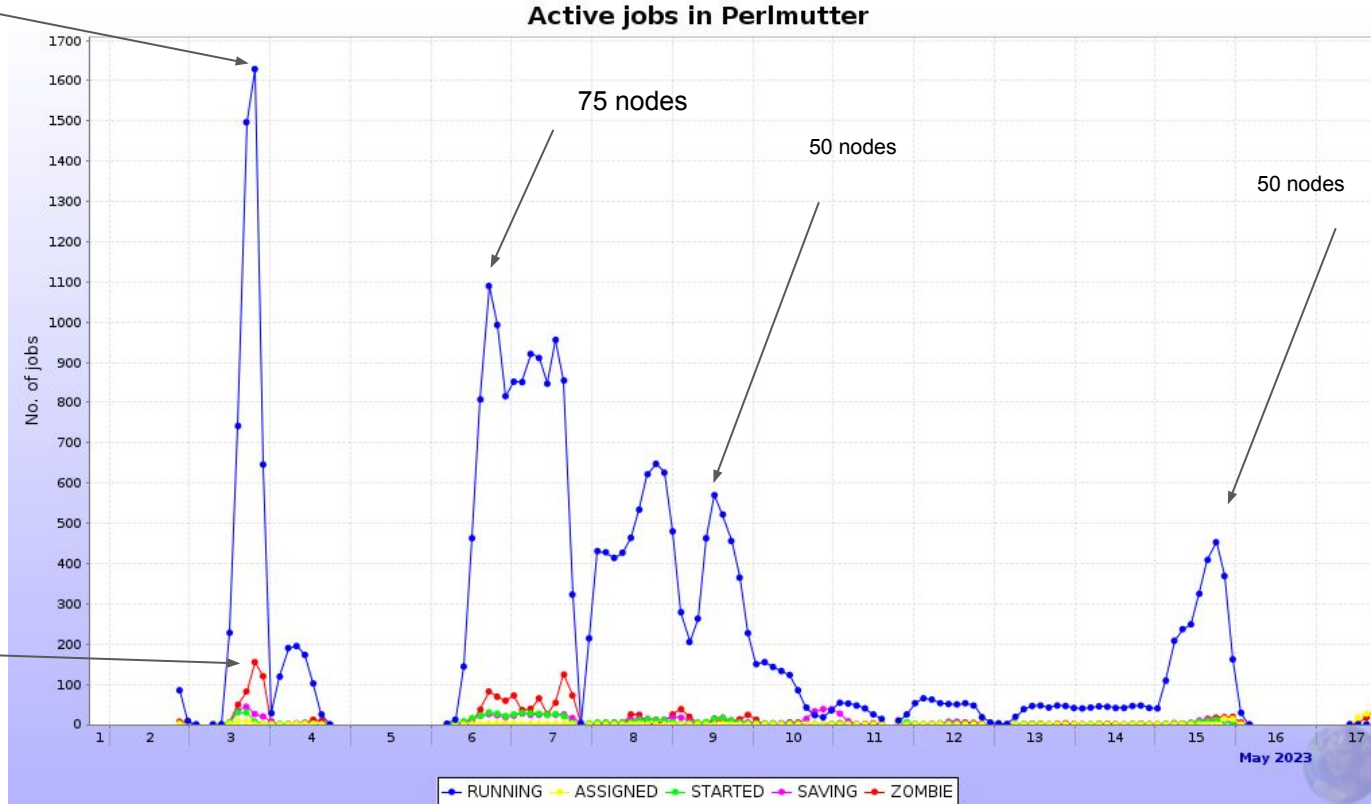
Implementing the SFAPI

- Authentication is done before doing any request using a private key and client ID
 - To be configured every time the key expires
- GET active jobs
- GET queued jobs
- POST submit a job to a queue
- Implemented as a module the same as other BQ implementations in JALiEn

Early Perlmutter Operation

Tried running on scavenging queue

All were preempted





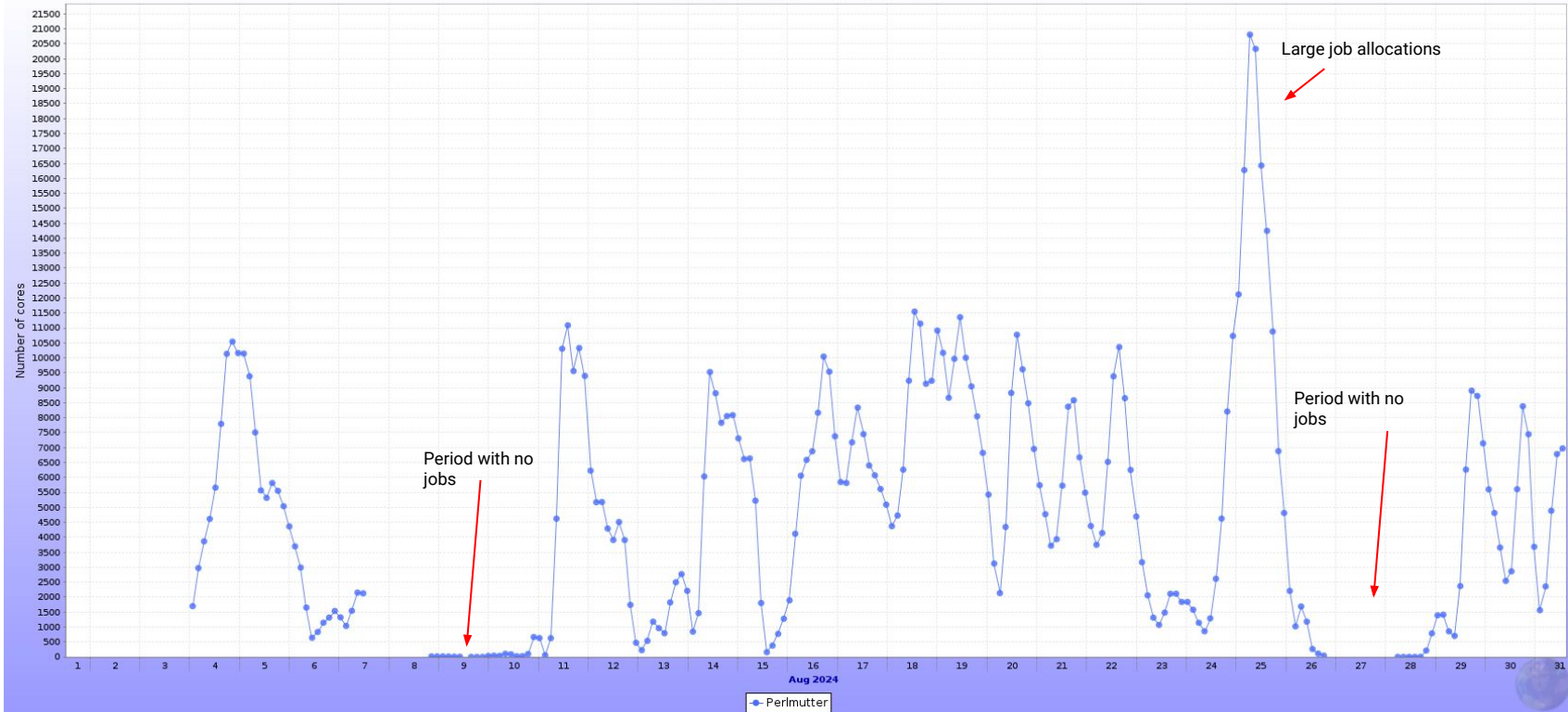
First Issues

- ulimits
 - Open files were running at $\frac{1}{4}$ than required
- CVMFS open files
 - Common issue for many high density sites
- Initial allocation was too small
 - Got an extension at the moment, but we scaled down job numbers



Getting Perlmutter to Run Steadily

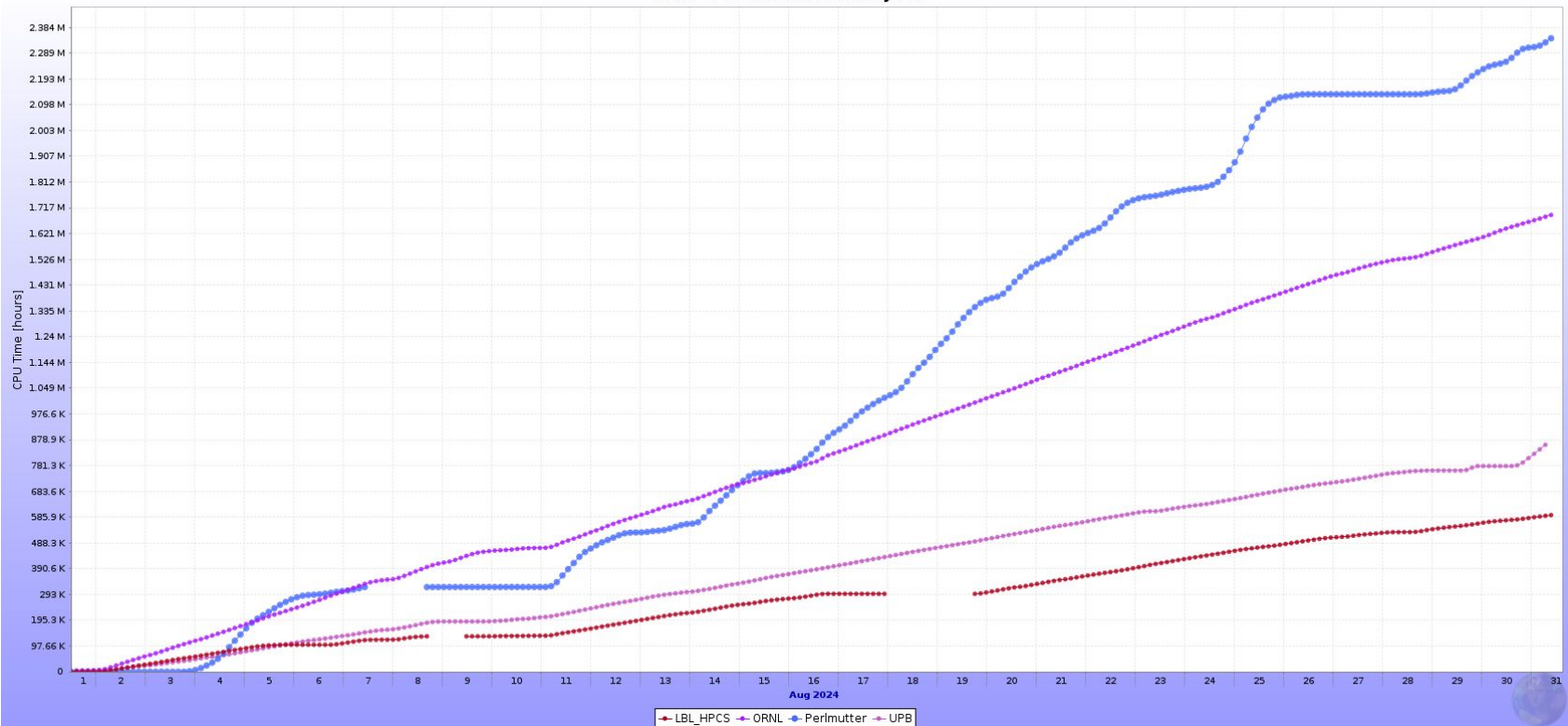
Total allocated cores per site





Perlmutter CPUtime

Total CPU time for ALICE jobs





What are we running into now?

- Tail of the slot is hard to fit jobs in
- HPC testing doesn't keep up to fast pace of JAliEn development
- Key management is still done by hand
- Sometimes queue is full for long times



Conclusion

- Integrated a new HPC to the grid
- Added a new job submission mechanism done through SFAPI

Future work

- Look into running on Perlmutter GPU nodes