JAliEn for HPC, whole-node and multicore jobs

Sergiu Weisz
sergiu.weisz@upb.ro
26/09/2022

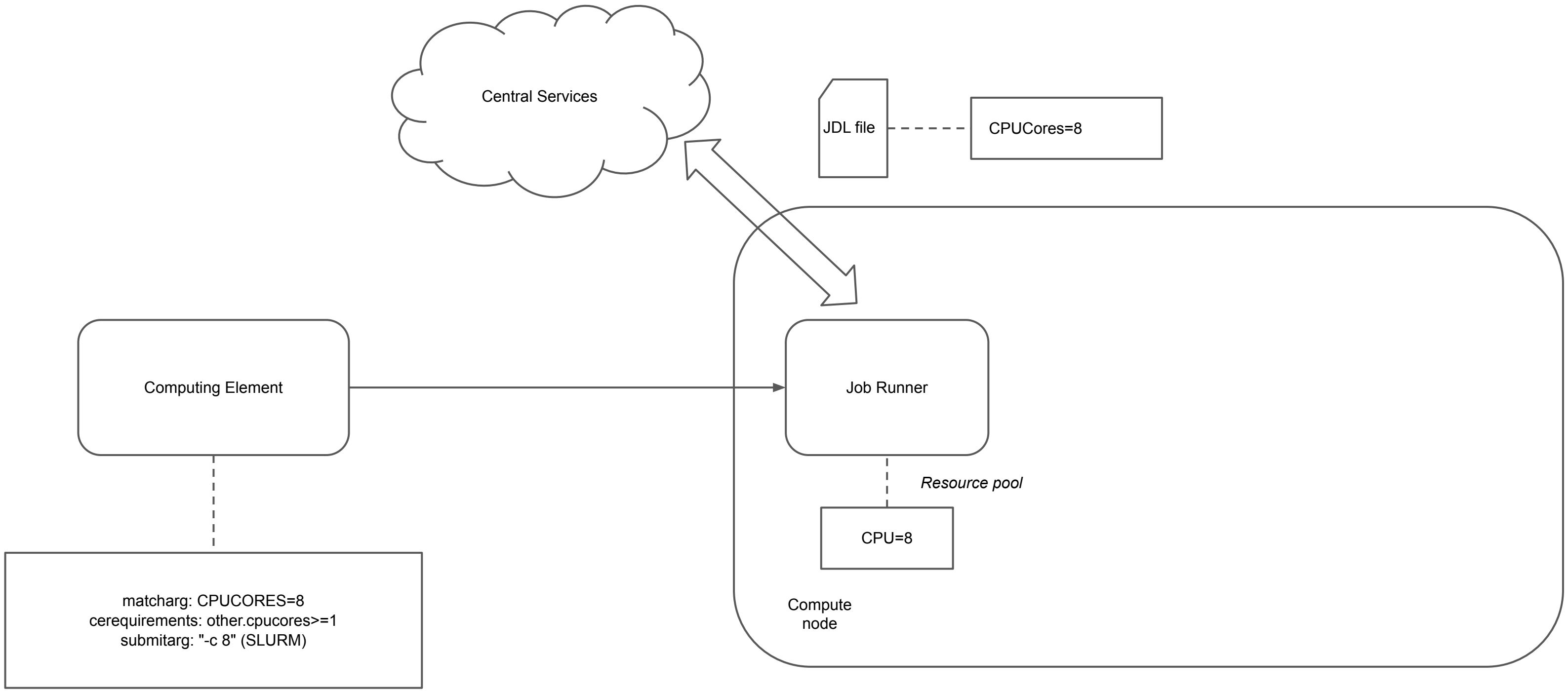
Software Updates for the ALICE Run3 and Beyond

- Updated data acquisition model
  - Increased 100x event rate
  - Generation of 10x more data
- New software framework for the experiment
  - Parallel memory-sharing processes
  - Fit into the 2GB/slot Grid limits
- Grouping of analysis tasks using the same input data
  - Independent processes run in multicore environment - increased I/O efficiency
- New paradigm: **multi-core** scheduling and payload support

Tackling Multi-core Jobs

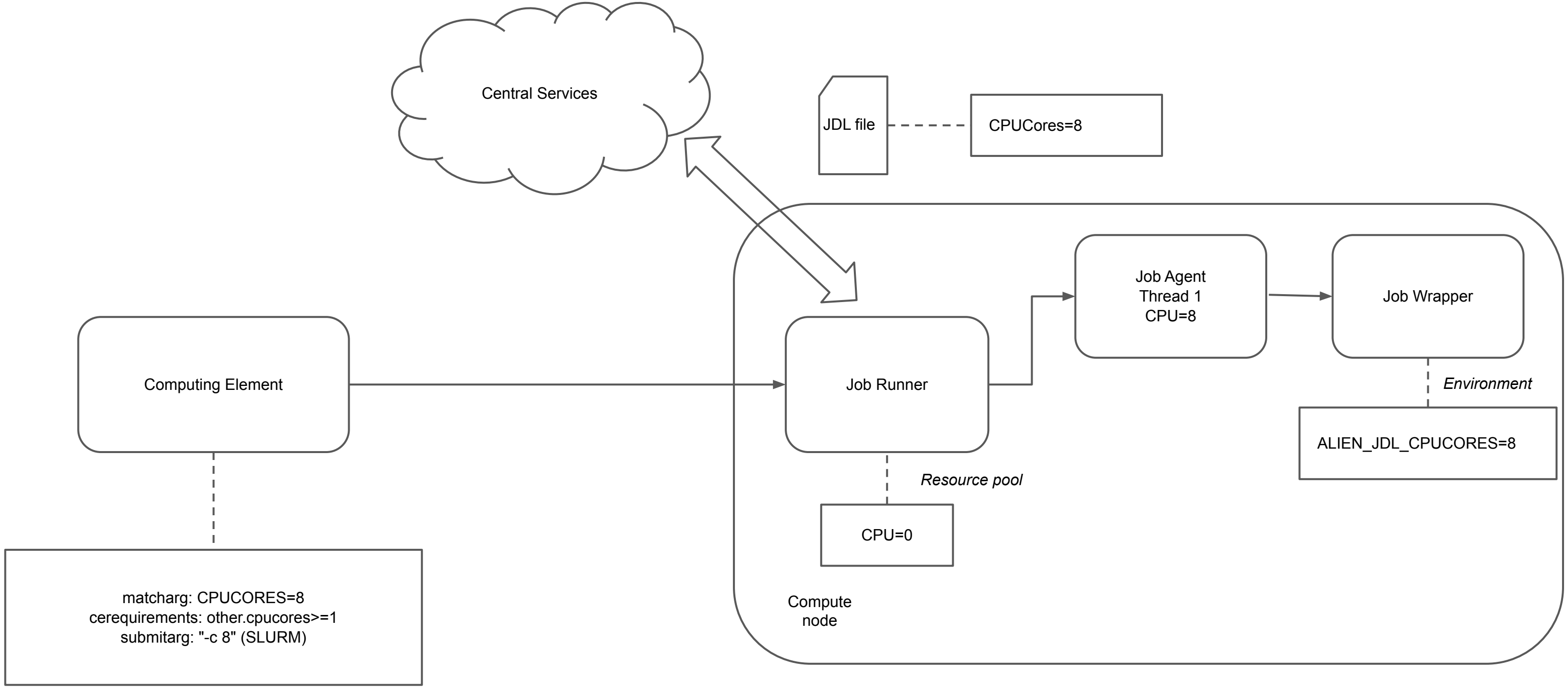
- New JobRunner component starts payload depending on slot configuration
- Single-core slots remain unchanged
- Multi-core slots can scale up or down depending on available jobs

Multi-core Scheduling



matcharg: CPUCORES=8
cerequirements: other.cpucores>=1
submitarg: "-c 8" (SLURM)

Multi-core Scheduling



Central Services

JDL file

CPUCores=8

Job Agent
Thread 1
CPU=8

Job Wrapper

Computing Element

Job Runner

Environment

ALIEN_JDL_CPUCORES=8

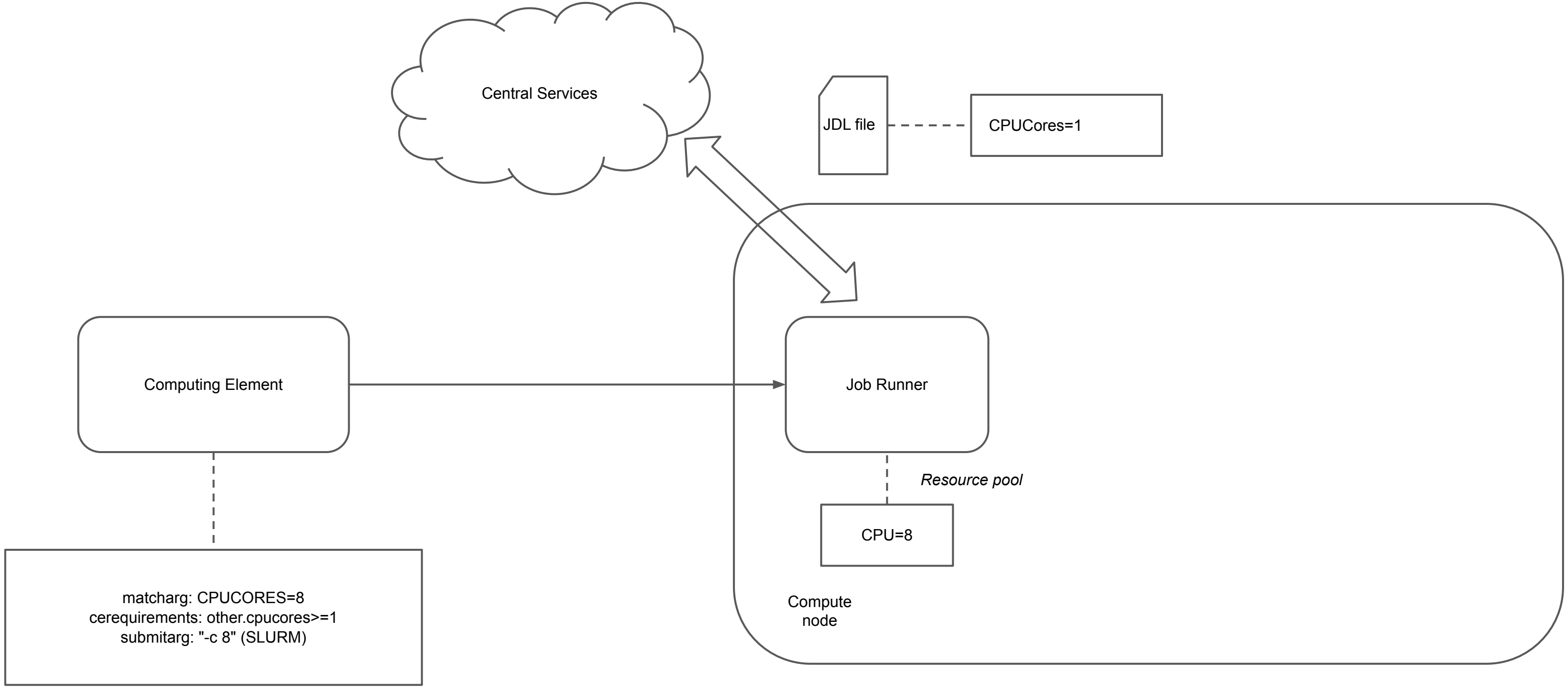
Resource pool

CPU=0

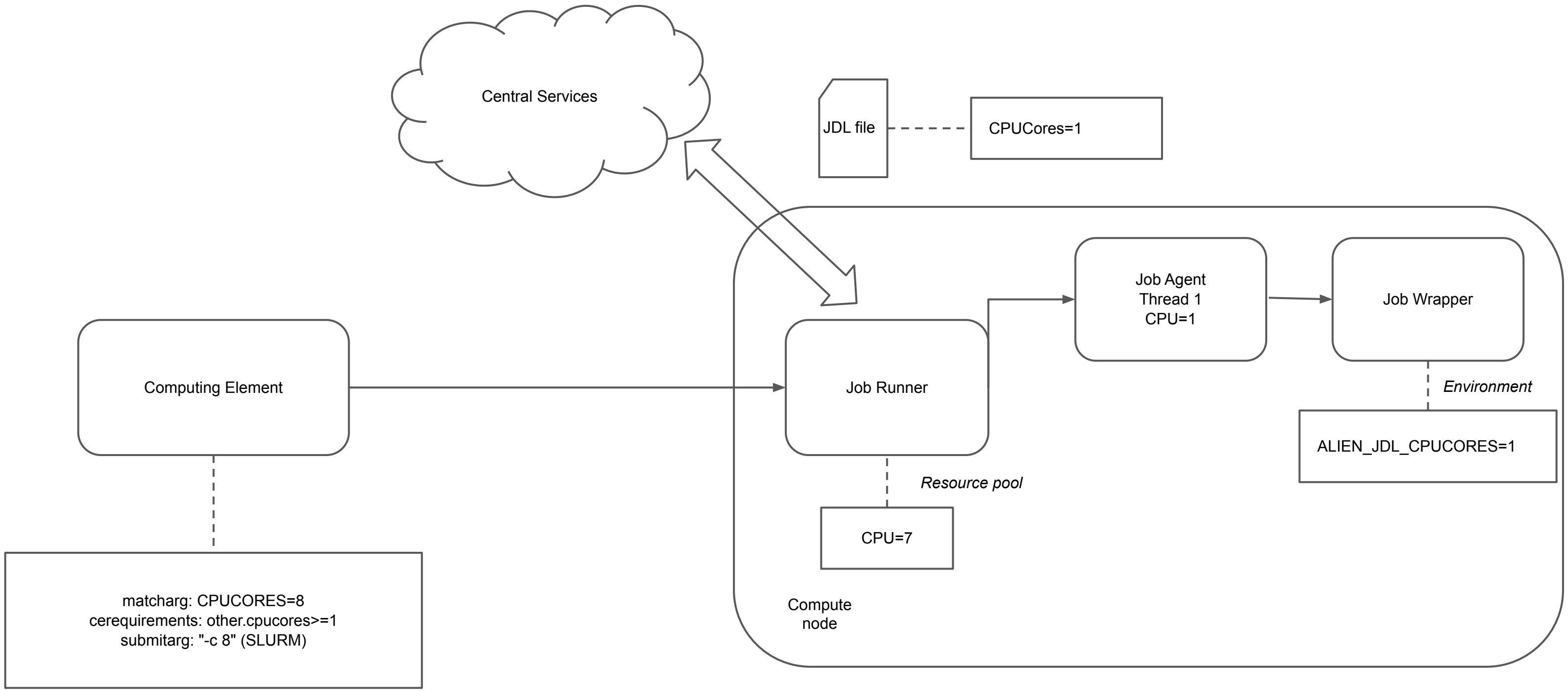
Compute
node

matcharg: CPUCORES=8
cerequirements: other.cpucores>=1
submitarg: "-c 8" (SLURM)

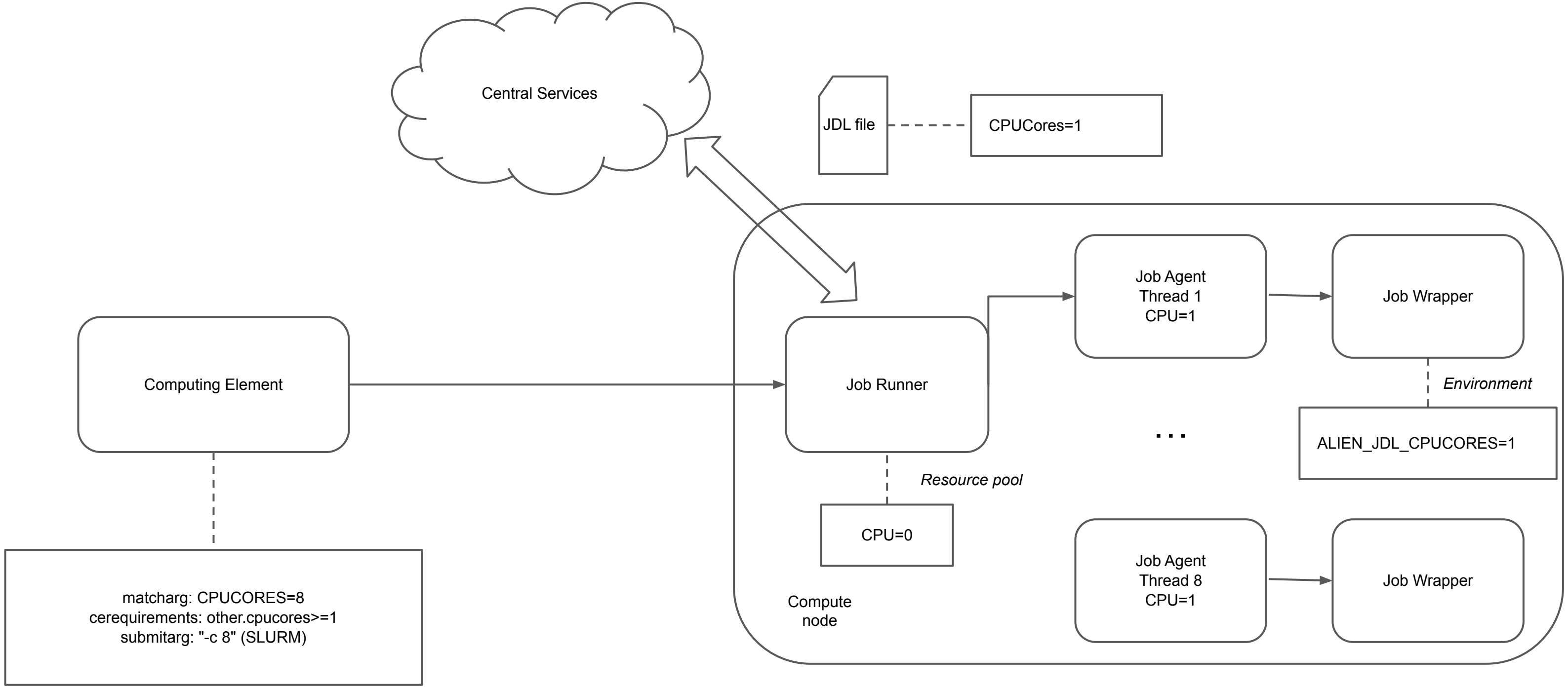
Multi-core Scheduling - 1 Core Jobs



Multi-core Scheduling - 1 Core Jobs



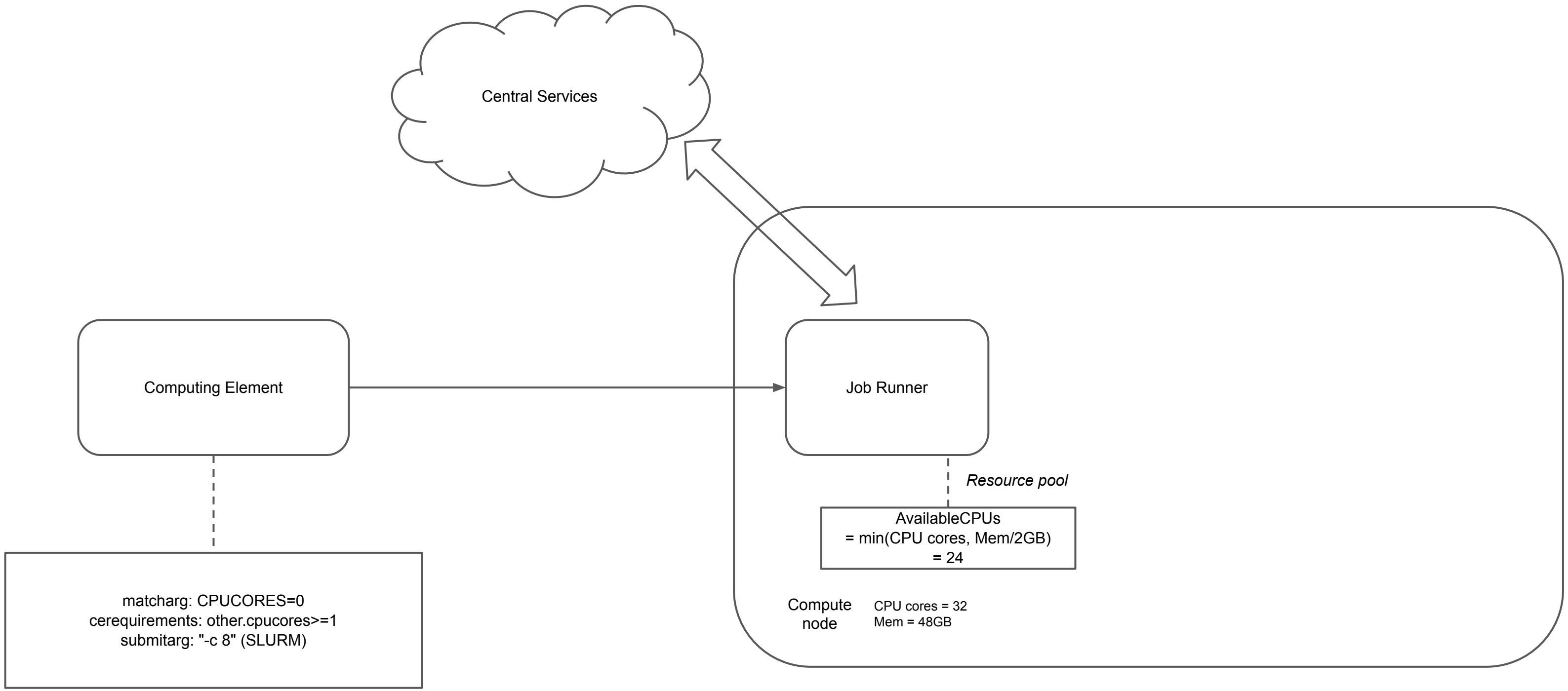
Multi-core Scheduling - 1 Core Jobs



Whole-node Scheduling

- What?
  - Instead of running jobs in slots that will fill a node, one slot is one node
- Why?
  - Better control of job environment
  - Allows for clever scheduling, like oversubscription
- How?
  - Per JobRunner resource management
  - For each node we check the RAM and CPU available and run as many jobs as possible

Whole-node Scheduling Run Through

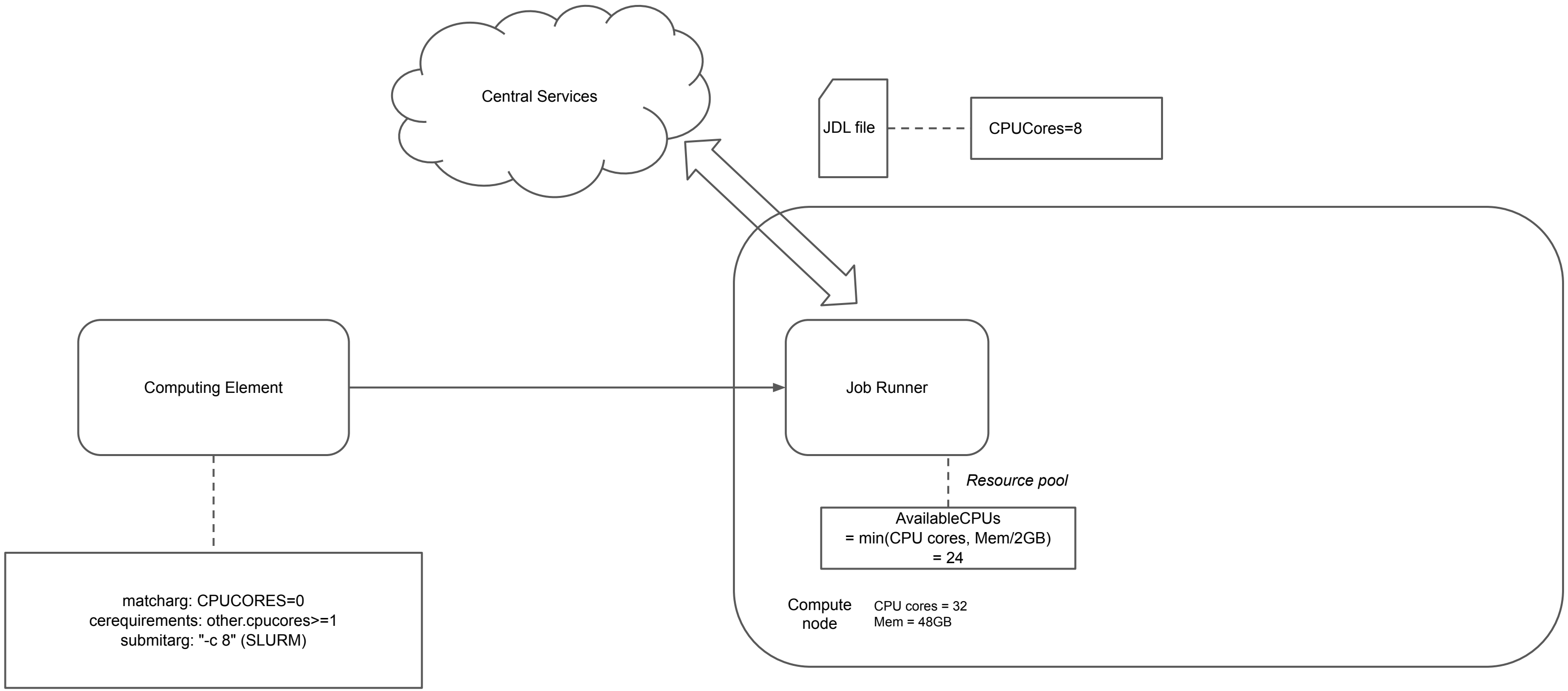


matcharg: CPUCORES=0
cerequirements: other.cpucores>=1
submitarg: "-c 8" (SLURM)

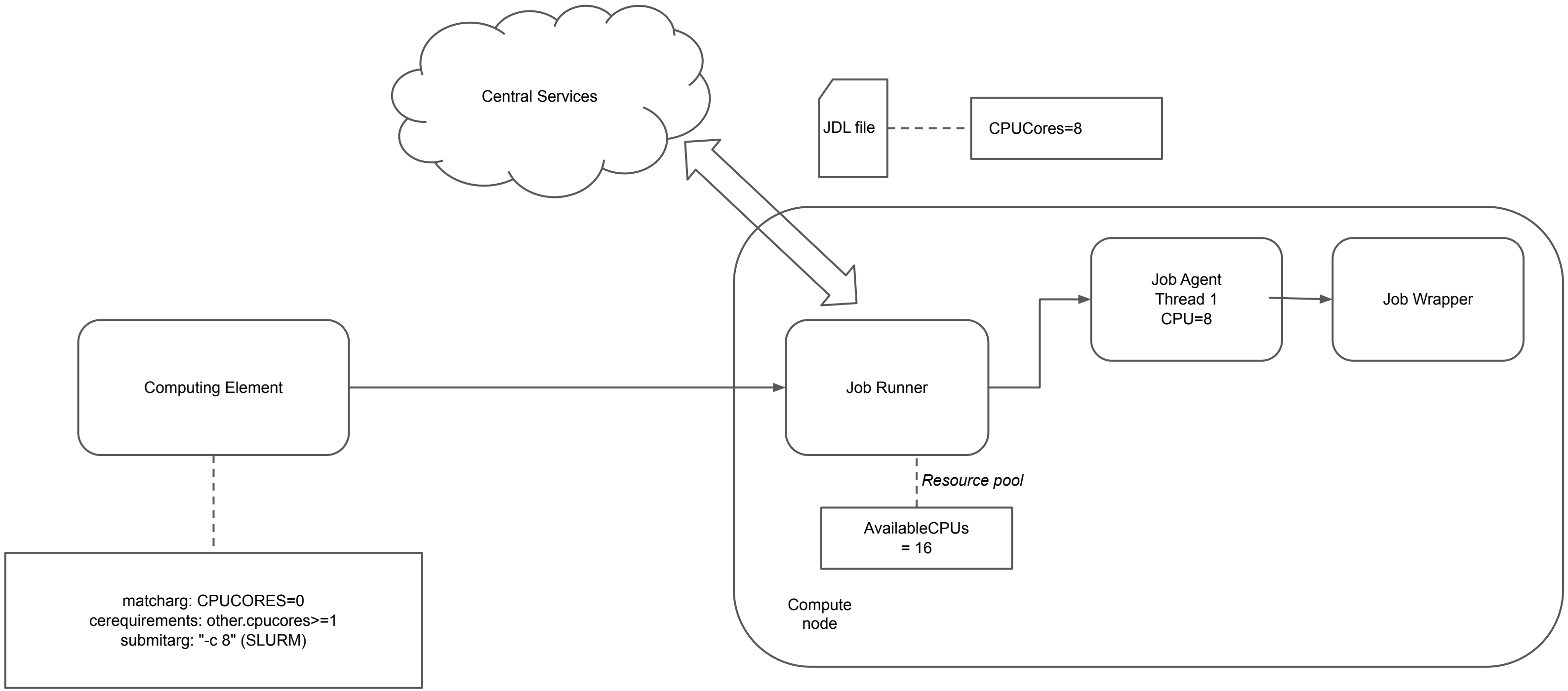
AvailableCPUs
= min(CPU cores, Mem/2GB)
= 24

CPU cores = 32
Mem = 48GB

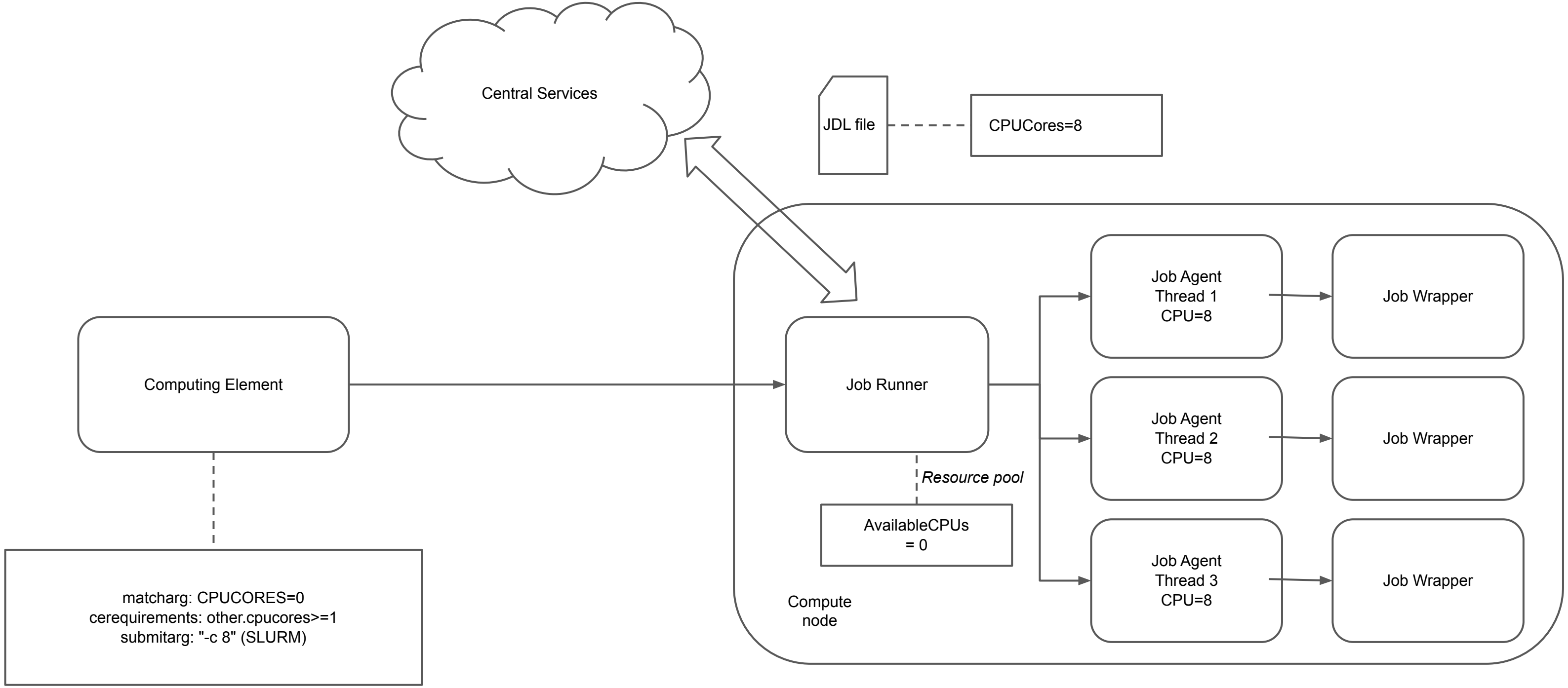
Whole-node Scheduling Run Through



Whole-node Scheduling Run Through



Whole-node Scheduling Run Through



LDAP Entries for Multi-core

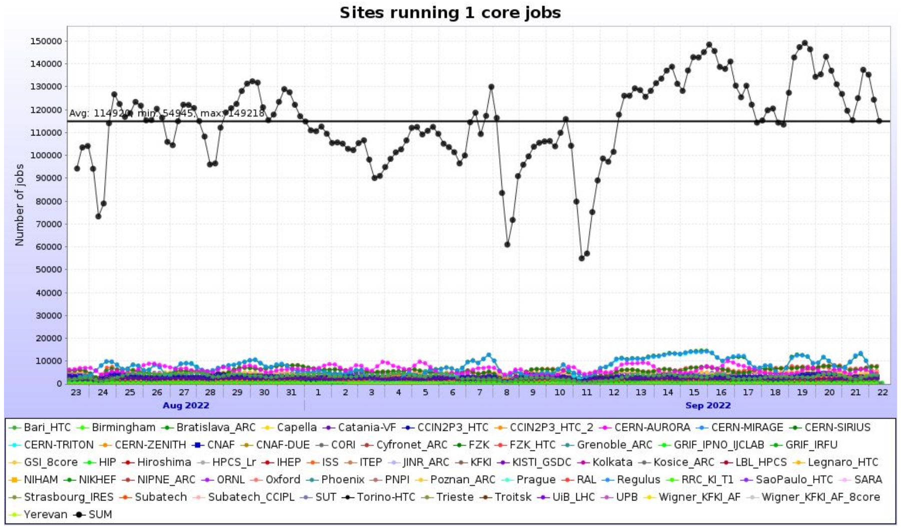
- Assigned cores per slot are set in the `matcharg: CPUCORES` variable
  - Set to 0 if using whole node scheduling

- `cerequirements` accounts for the requirements imposed by the CE to host the execution of jobs
  - The CE might impose constraints on the number of used cores

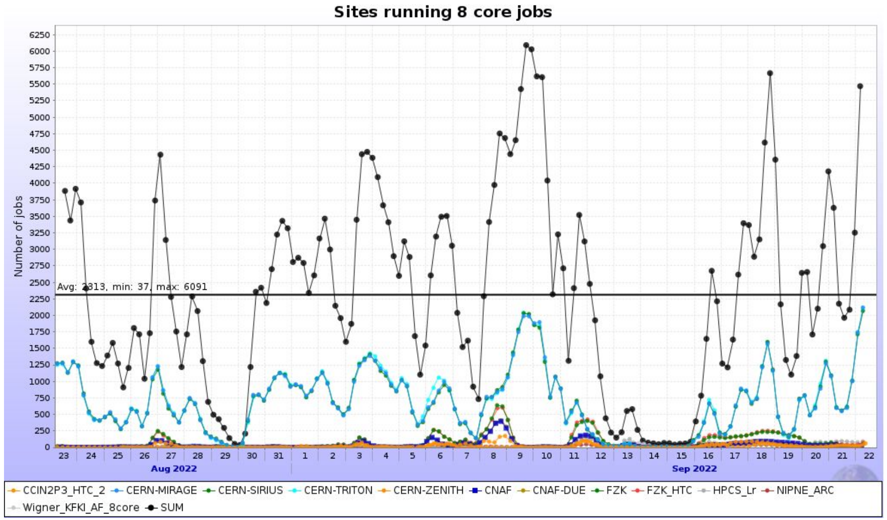
- `submitarg` sets the arguments to be appended to the batch queue submission command

- The site is added to the `multicore_8` partition

Single and Multi-core Job Share



Monthly view of main sites running single-core jobs



Monthly view of main sites running eight-core jobs

Sum avg: 114928

Sum avg: 2313

- 2% of the jobs are multi-core
- 10% of cores used in multi-core jobs

Caveats of Multi-core Jobs

- More file descriptors open => we should double check the open file ulimit
    - The issue is both with CVMFS and the grid process, set both services

- More running processes => we should double check the running process limit

- Higher load on CMVFS => we recommend 20GB per node cache

- Jobs may use too much CPU => we should isolate multi-core slots

HPC resources

Why HPC?

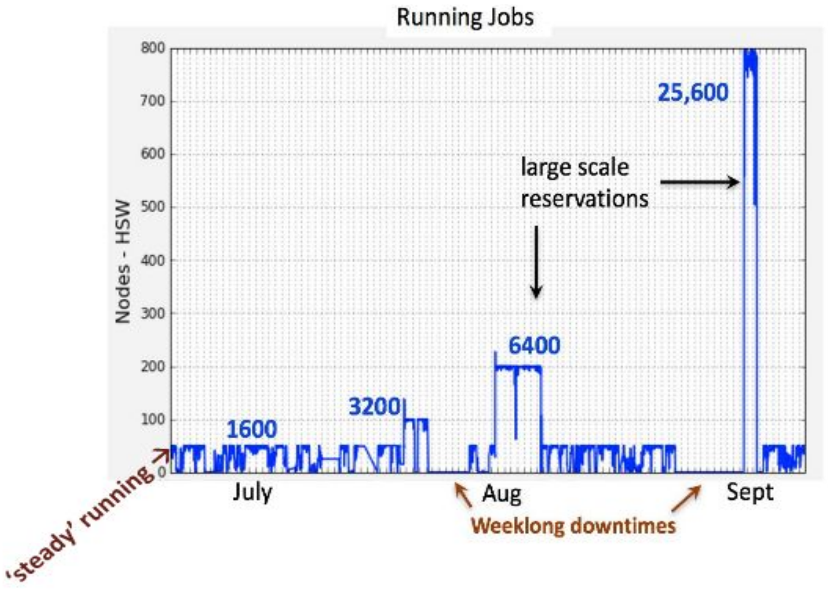


Chart extracted from Cori supercomputer at Lawrence Berkeley National Lab

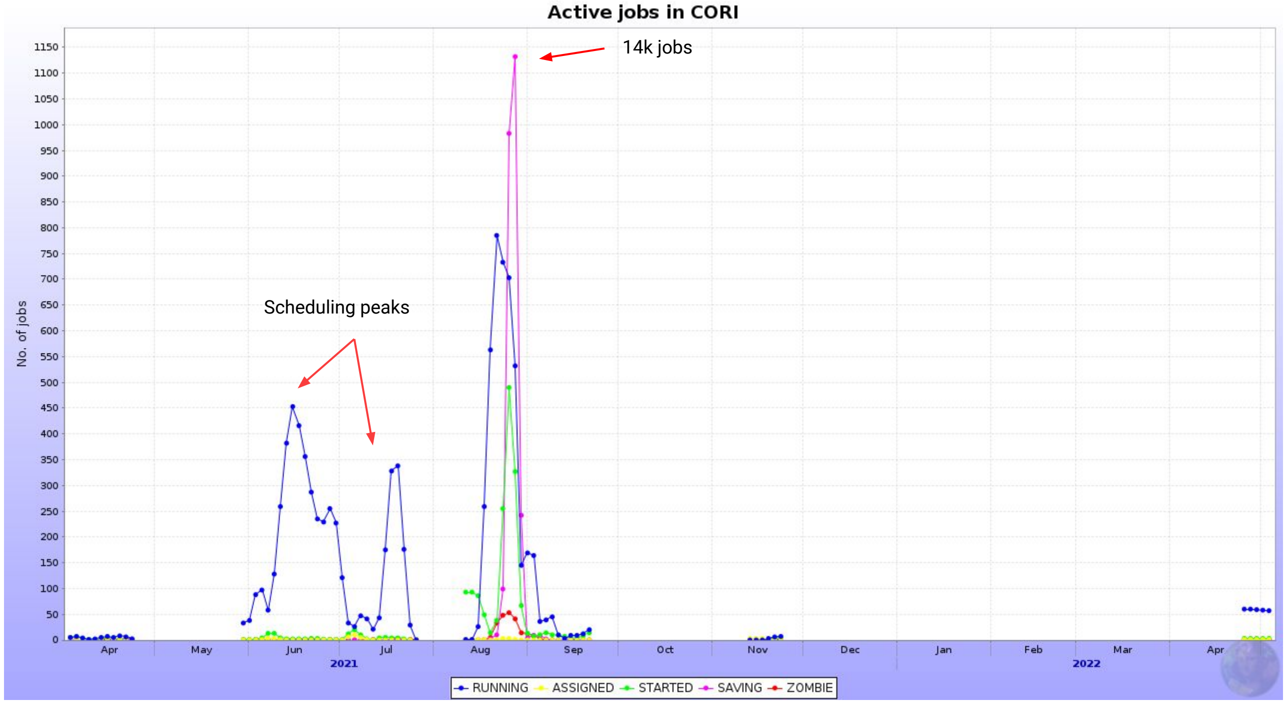
HPC Particularities

- Closed networks
- Heterogeneous architecture
- Peculiar software distributions
  - Some can’t run CVMFS

Meet CORI

- Runs SLES 15
- Only whole-node scheduling
- 2,388 nodes running Intel Xeon Processor E5-2698 v3, 32 core, 128 GB memory
- Jobs run in Shifter containers
- Outside connection from nodes and nodes to CE connection
- Full CVMFS support

Running Jobs On CORI So Far



14k jobs

Scheduling peaks

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

Running Jobs On CORI

- JAliEn runs from the source files

- In order to have a working environment, we have to run the jobs in a Shifter container
    - The only packages installed are *strace* and *HEP_OSlibs* and debugging applications
    - CVMFS is mounted using Shifter

- We still run the old MonaLisa scripts

Issues With Running Jobs On CORI

- VObox scripts don’t work
  - There’s an incompatibility with the CVMFS java version
  - For now we’re running a source files instance with running scripts changed

- We’re working now on demonizing the CE without the vobox scripts

- Low uptime is a big problem
  - Have to always keep an eye out for the screen being killed

Running Analysis Jobs On CORI

- CORI supercomputer is now limited to running simulation jobs

- Analysis tasks are I/O intensive operations
    - Require download bandwidth of at least 10 MB/s per core

- No dedicated storage - all data is accessed remotely
    - Pre-location of the data in the nearest SE - ALICE::LBL_HPCS::EOS.
    - Bandwidth link of 30 Gbps

- Study of site capabilities and behavior in concurrent downloads
    - Usage of crawling tool reporting dimensions and duration of downloads

Running Analysis Jobs On Supercomputers

- Relevant factors:
  - Location of SE from which data is taken - ALICE::LBL_HPCS::EOS
  - File system of the tasks’ working directory
  - Amount of parallel tasks running on working nodes
  - Number of used nodes

- Observed much lower bandwidth than expected
  - Tests on speed of data storage revealed that the problem was on the CORI networking infrastructure

- Not enough details on network topology - handed out problem to site admins
  - Developed set of python scripts that could be used without Grid credentials
  -

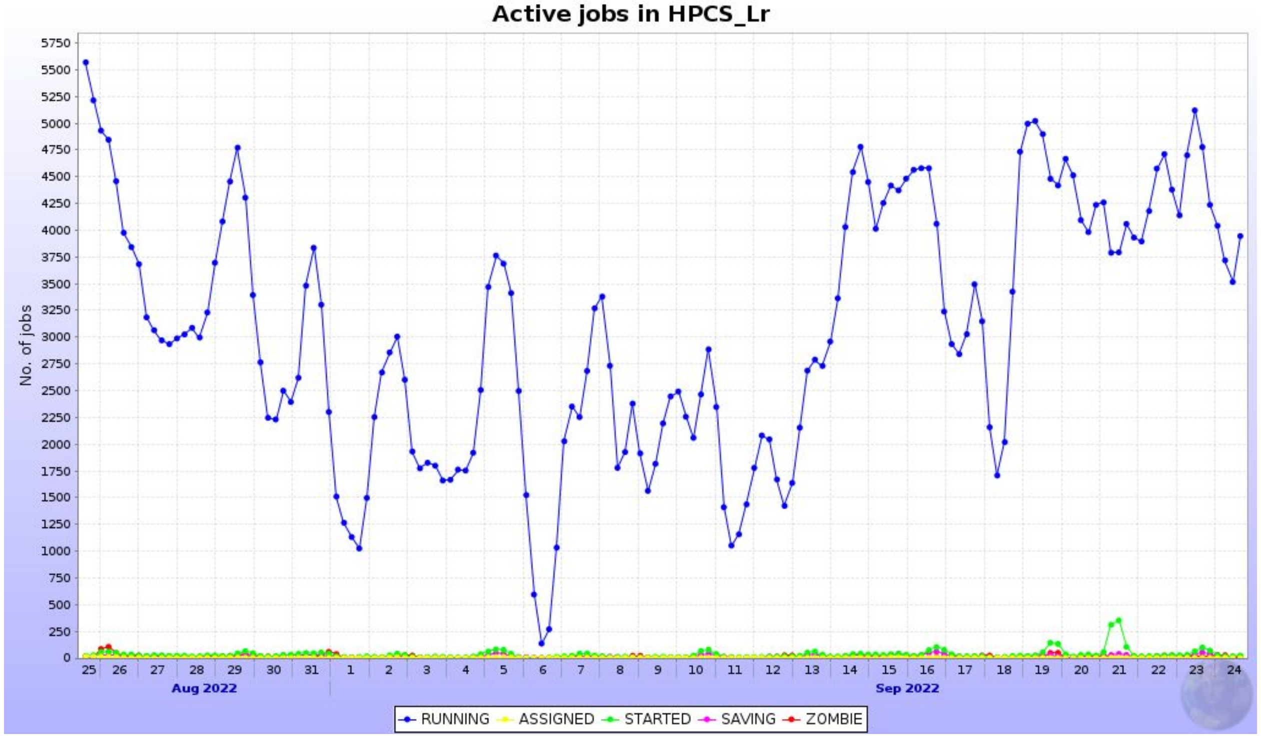
Opportunistic job scheduling

Opportunistic Jobs?

- What?
  - Preemptable job queues

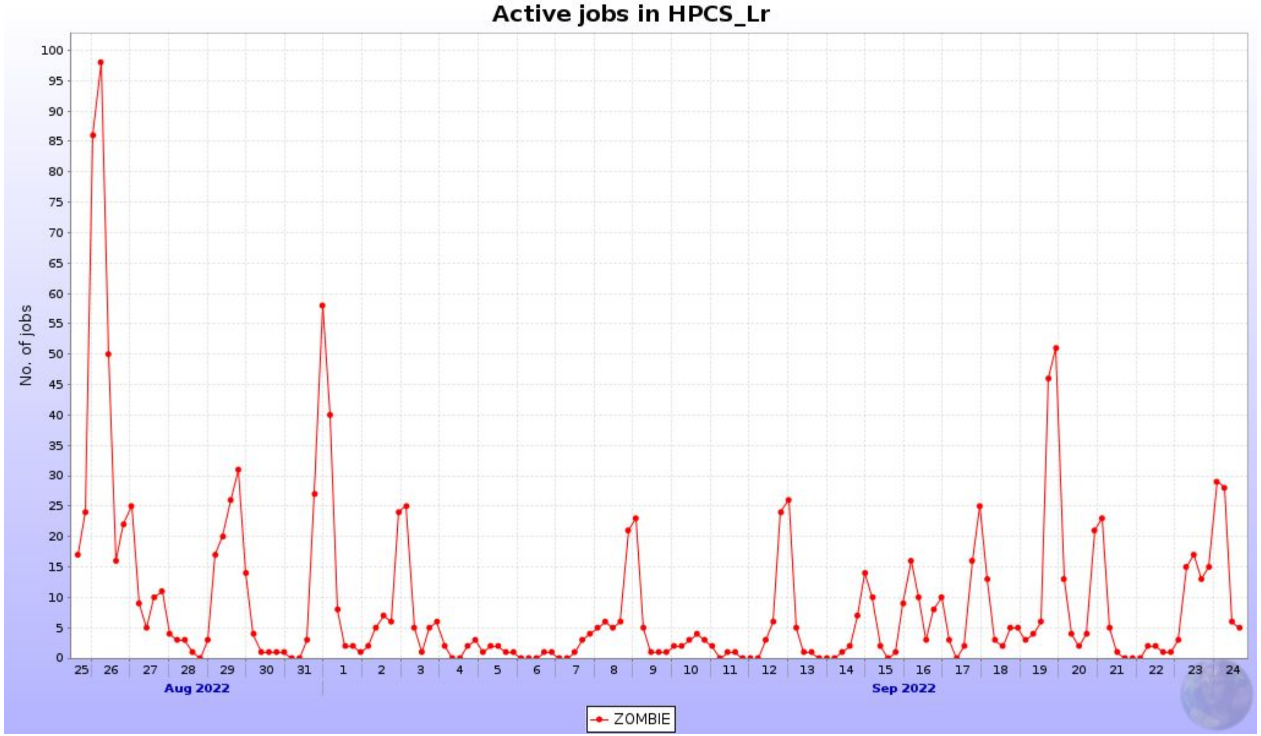
- Why?
  - Cheap resources

Running Jobs On HPCS_Lr So Far



- AKA the second JAliEn install
- Using whole-node slots
- Jobs are being constantly preempted, but their number is lower
- Average running: 3082

Job Fluctuations On HPCS_LR



- Expected job fluctuation
- Average preemption rate: 9.144
- Job fail rate: 0.2%

Running Jobs On Lawrencium

- JAliEn runs using vobox scripts
- Jobs run in a whole-node scheduling fashion
- It has turned out to be a good resource provider, for now

Conclusion

- Multi-core jobs are running on the grid
- Sites should migrate to multi-core slots
- If available, whole-node slots are recommended
- HPC resources have been integrated with mild success
- Scavenging queue have proven to be a great success