Dynamic scheduling using CPU oversubscription on the ALICE Grid

Marta Bertran Ferrer
PhD Student - CERN

On behalf of the ALICE Collaboration

Grid resource usage overview

Optimize the use of resources by taking advantage of the different nature of Grid payloads

- Analysis jobs are I/O intensive and they do not fully use the allocated CPU slot
- **Idle portions of CPU** can be gathered and used to execute computing intensive jobs (MonteCarlo)
  - No added pressure in I/O

Our analysis shows that ~75% of the worker nodes have **spare memory resources** not used by the running jobs

- Base allocation of 2GB RAM/core (up to 8GB/core with SWAP)

Whole-node allocation and partitioning

Multicore pilots can run in two configurations

- **Whole node**: JAliEn job wrapper manages resources and splits them into job execution slots
- **Slots of *N* cores**: Jobs are assigned a set of resources by batch
  - This might be further enforced by some resource constraining mechanism
  - These slots can be further partitioned to run finer-grained jobs as in the whole node case

We are particularly interested in the a **whole node** configuration - allows for increased flexibility and optimal resource management

Whole-node allocation and partitioning

In whole node scheduling scenarios we want to use the available resources in the **most efficient way**

- Memory is the biggest constraint during job execution
- Compute **memory per CPU core ratio** and see if they can be oversubscribed
- Node's *Idle* CPU above a threshold → It has room for an extra job
- Assign unused memory to new jobs and oversubscribe CPU cores

Making use of the internal machine and job resource monitoring

Analysis of current Grid worker nodes usage levels

Analysis of the **current usage levels of the Grid worker nodes** during **72 hours**

Goal - to have an accurate picture of the **feasibility of adopting oversubscription policies** for the hosts used for **whole node** submission

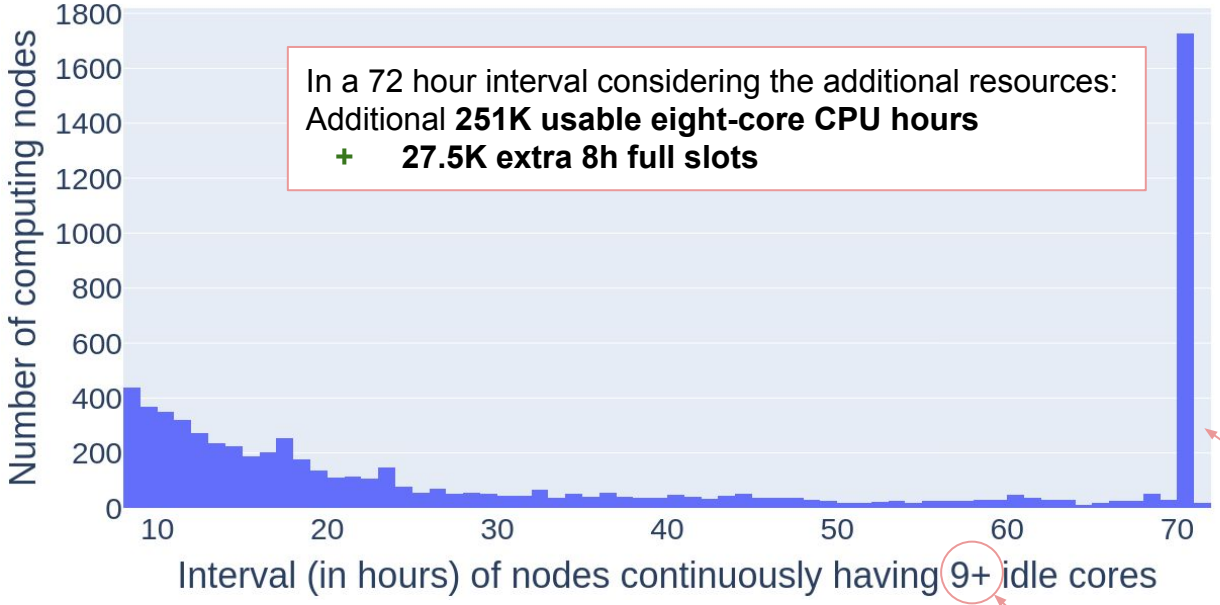
All hosts reporting values to ALICE monitoring system MonaLisa have been included

- To take into account that they might be running other workloads in parallel for which we do not have any knowledge → ALICE workflows are also heterogeneous

Continuous sampling of the **amount of idle cores** to evaluate the potential additional slots

- Set **grace interval** for deciding **when to start** a new payload (waiting for the amount idle cores to stabilize) and for deciding **when to preempt** a payload (when machine becomes saturated)

Eight-core usable resources in oversubscribed mode



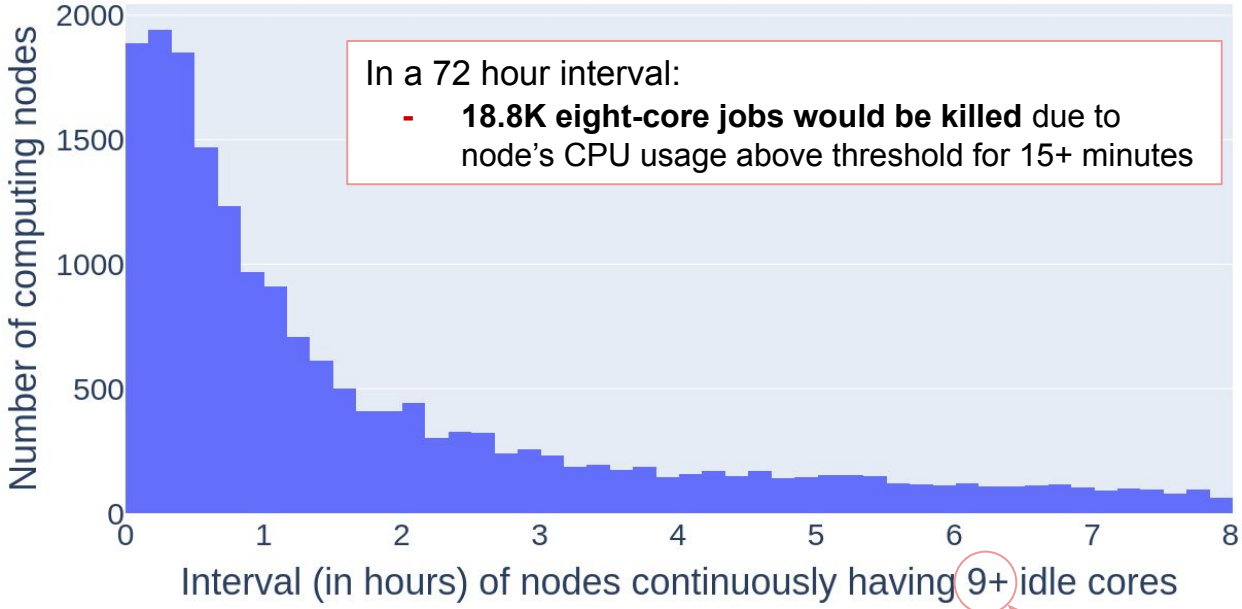
Usable eight-core CPU hours:
Idle time intervals lasting 8+ hours

Total eight-core CPU hours of the surveyed hosts in 72h: 2.5M hours
=> Additional usable CPU hours (from plot on left) are more than a **10% of the total Grid resources**

Large amount of worker nodes with free cores for the entire 72h period

8 core job execution + 1 core left idle for system operations

Caveat: Potential eight-core jobs preemption



In a 72 hour interval:

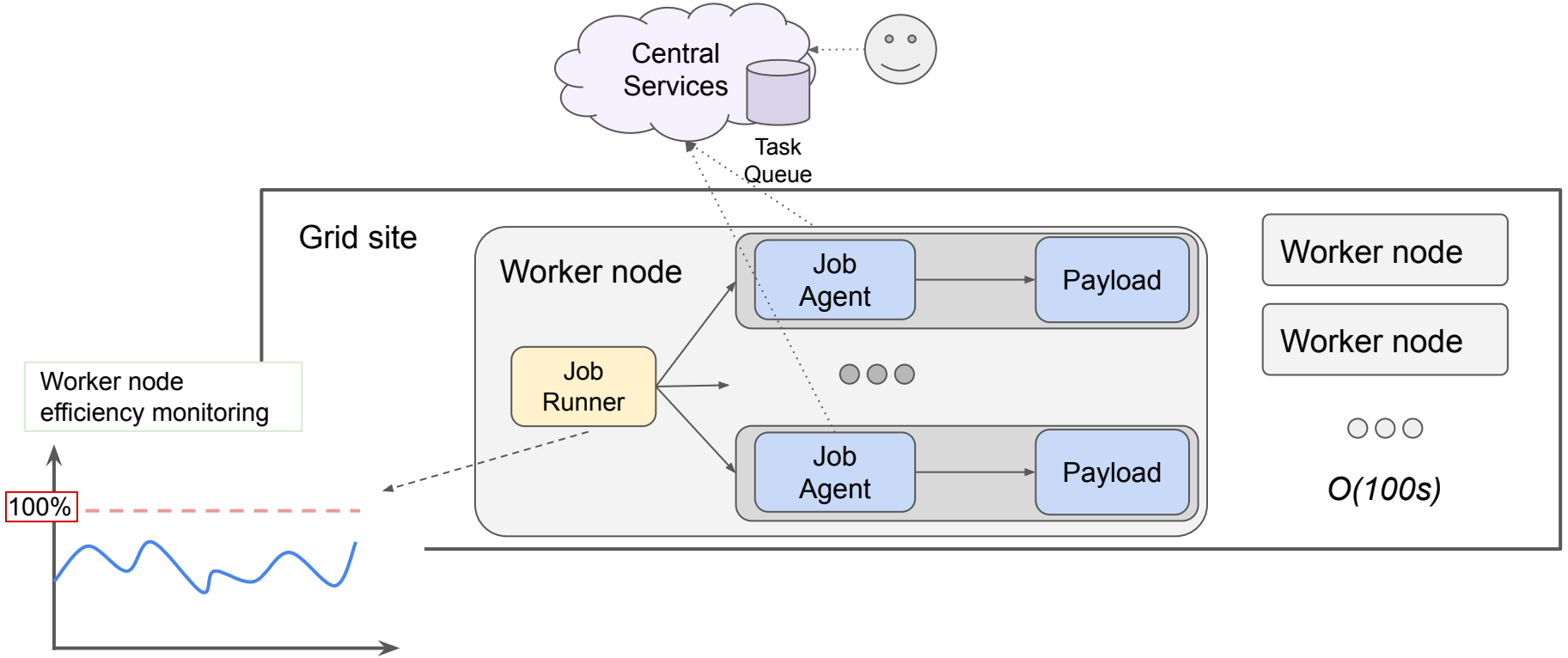
- **18.8K eight-core jobs would be killed** due to node's CPU usage above threshold for 15+ minutes

Monte-Carlo workflows that run in **opportunistic fashion**

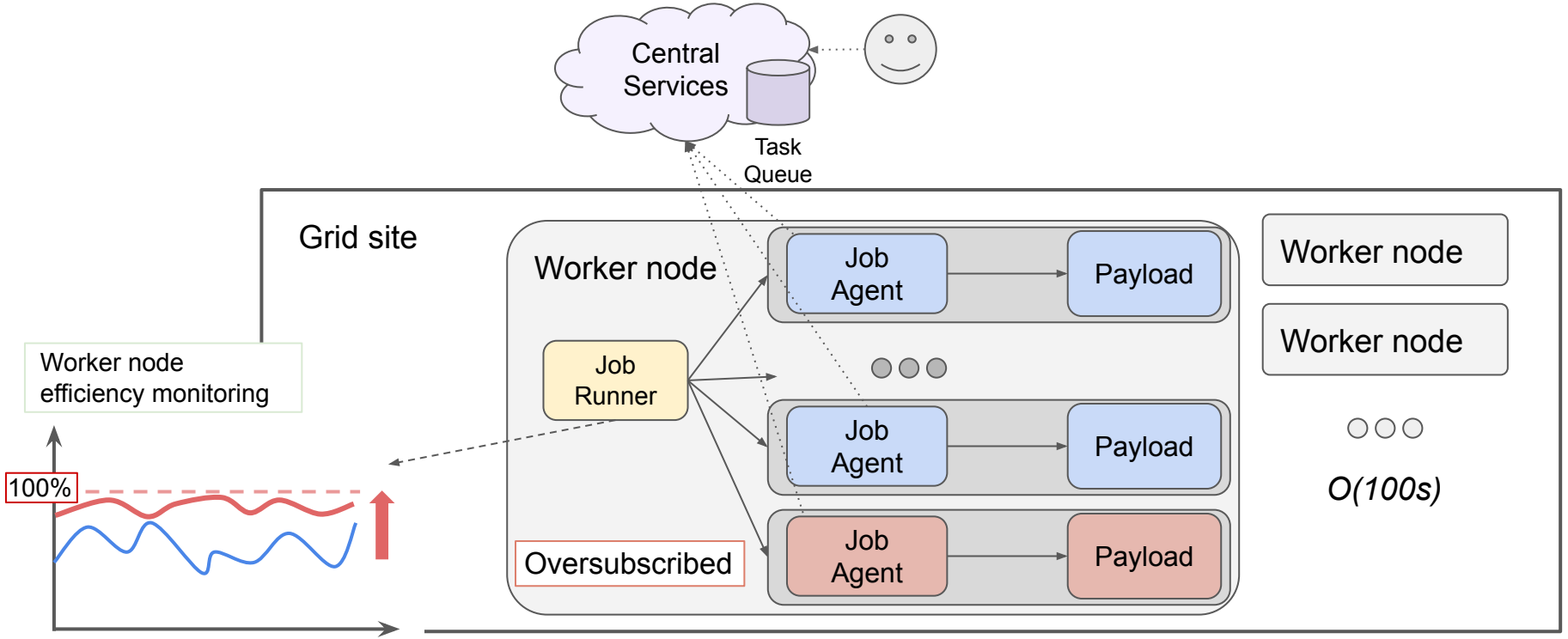
- Find a balance between preemption rate and pressure in database

8 core job execution + 1 core left idle for system operations

JAliEn in oversubscription regime



JAliEn in oversubscription regime



Oversubscription implementation

The amount of CPUs available for oversubscription will depend on the node's memory

- Minimum between [RAM/2, (RAM+SWAP)/8]

Jobs to execute under oversubscription regime are CPU intensive (minimal I/O)

- Target jobs are MonteCarlo with a number of cores that fits the allocated amount of CPUs in the oversubscription pool

Oversubscribed jobs are executed re-niced → **Lower priority**

Preemption of oversubscribed jobs if machine gets saturated

- Jobs are rescheduled without penalty
- Priority to continue executing long-lived jobs

Testing on different environments

- The oversubscription-enabled framework has been deployed on two test machines
  - Oversubscribed cores limited by SWAP in both cases:
    - Machine 1: 32 cores Intel Xeon Gold 6244 at 3.60GHz. 11 oversubscribed cores
    - Machine 2: 64 cores Intel Xeon Gold 6226R at 2.90GHz. 3 oversubscribed cores
- Deployment of 24h slots with fully saturated machines executing equal single-core analysis jobs
  - Oversubscribed jobs are single-core CPU-intensive MonteCarlo simulations

Studied metrics

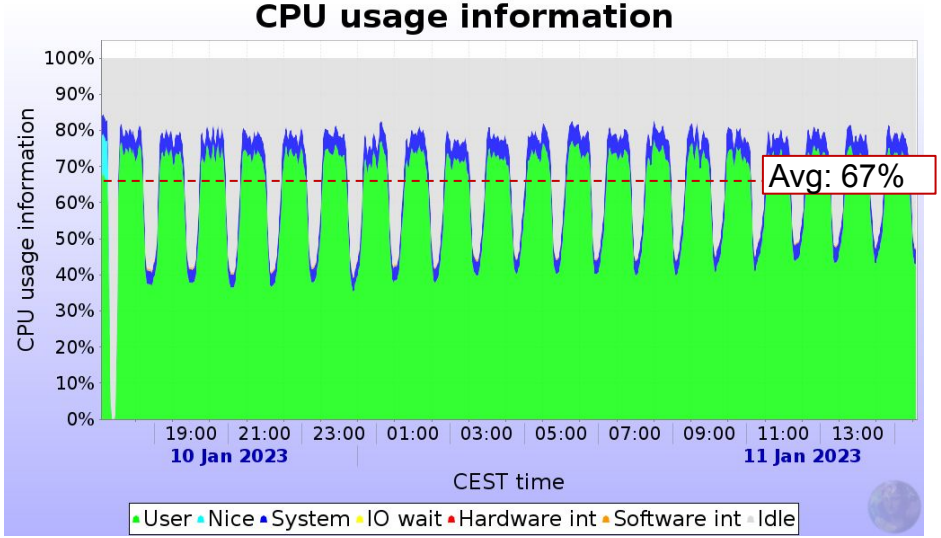
- Job efficiencies
  - Regular job efficiency → $\frac{\Sigma \text{ CPU time regular jobs}}{\Sigma \text{ walltime regular jobs}}$
  - Extra job efficiency (MonteCarlo) → $\frac{\Sigma \text{ CPU time extra jobs}}{\Sigma \text{ walltime extra jobs}}$

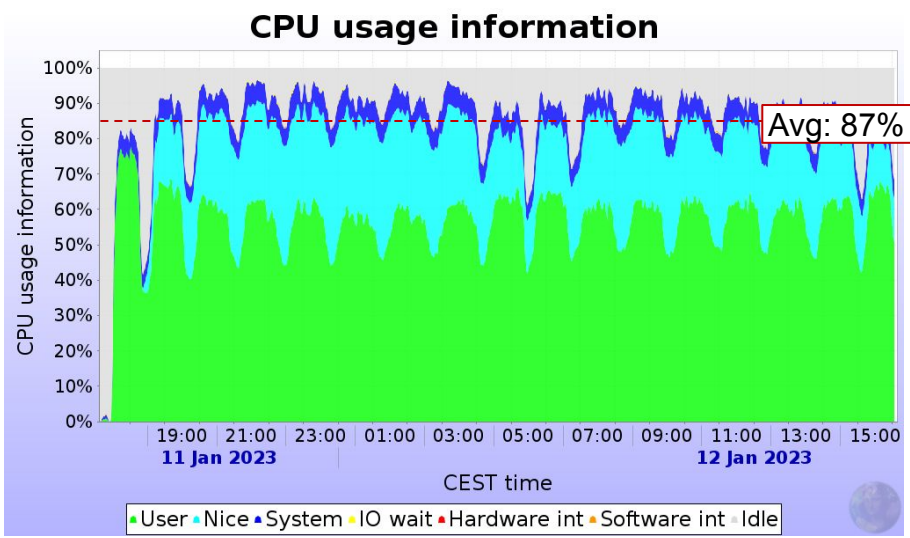
- Machine utilisation efficiency

$$\frac{\Sigma \text{ CPU time all jobs (regular + extra)}}{\text{Slot duration (24h)}}$$

  - This is the **metric we aim to improve**

Oversubscription behaviour - Machine 1





Regular pool: 32 cores

Regular pool: 32 cores
Oversubscribed pool: 11 cores

Evaluation

- Taken for realtime = 24h

- $$\frac{\Sigma \text{ CPU time all jobs (regular + extra)}}{\text{Slot duration (24h)}}$$

$$\frac{\text{Utilization efficiency oversubscribed machine 1}}{\text{Utilization efficiency non-oversubscribed machine 1}} = \mathbf{1.28}$$

$$\frac{\text{Utilization efficiency oversubscribed machine 2}}{\text{Utilization efficiency non-oversubscribed machine 2}} = 1.03$$

Conclusions

- **Whole node scheduling** allows for optimal management of the available resources
  - Currently promoting conversion of sites to whole-node
- The increase in machine utilisation compensates the decrease of efficiency of the execution of individual jobs
- Observed increased utilisation efficiency of the hosts is proportional to the amount of extra executed jobs
- Running additional jobs with **complementary resource usage patterns** on a worker node has a great potential to compensate for inefficiencies due to the different nature of the payloads (I/O intensive vs CPU intensive)