

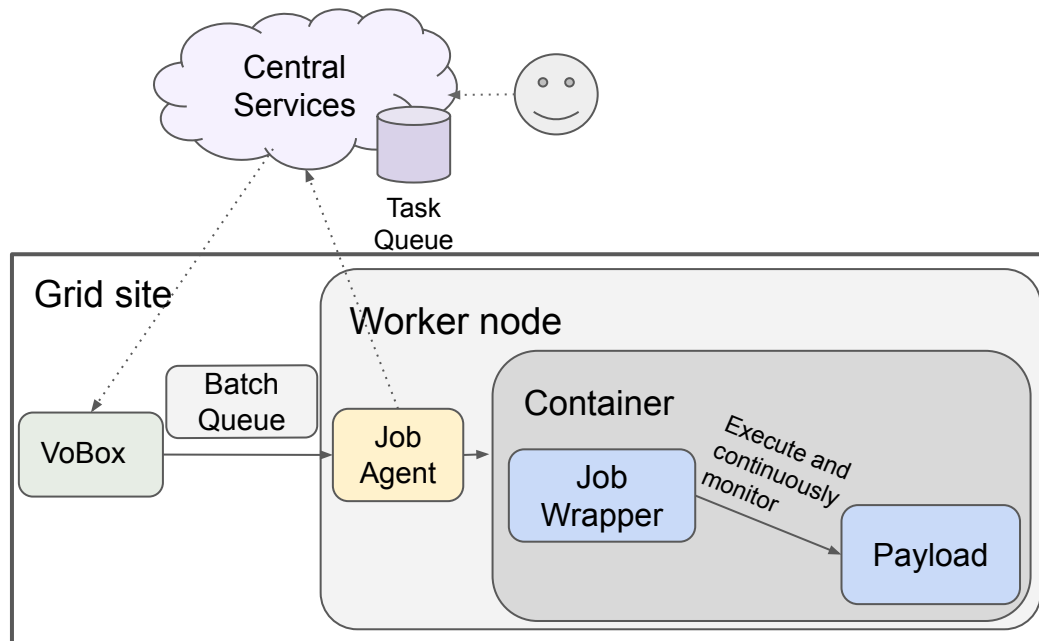
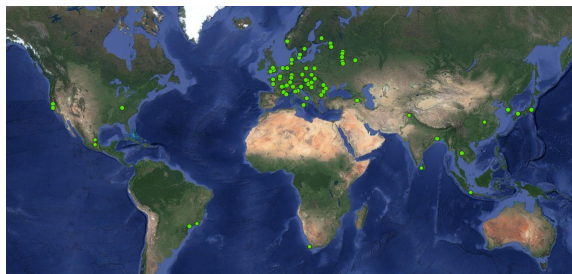


# CPU-level resources allocation for optimal execution of multi-process physics code

Marta Bertran Ferrer  
PhD Student - CERN

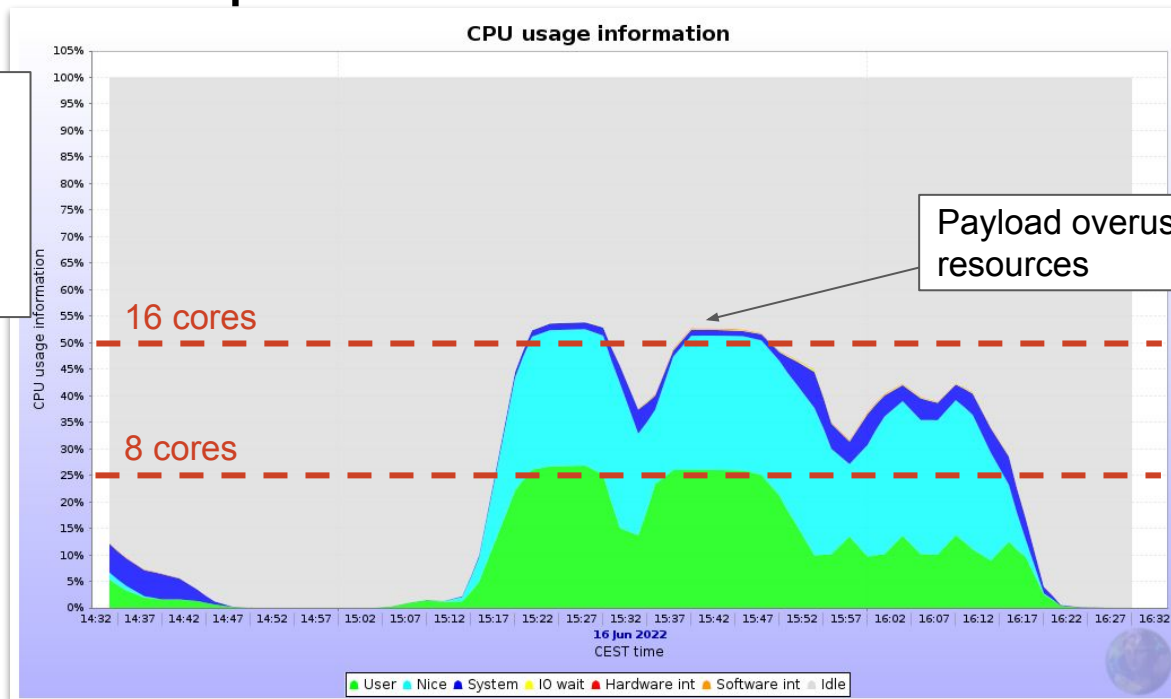
# The ALICE Grid

- Uses WLCG resources
- Used to perform all ALICE computing activities
- JALiEn: Middleware framework



# Payload behavioral problem

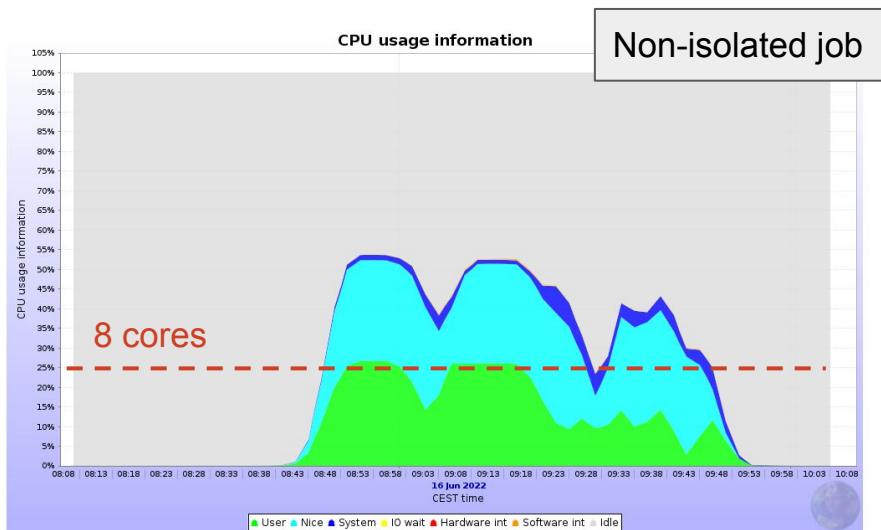
CPU usage of a 32-core idle machine running a simulation payload that *requests 8 CPU cores*.



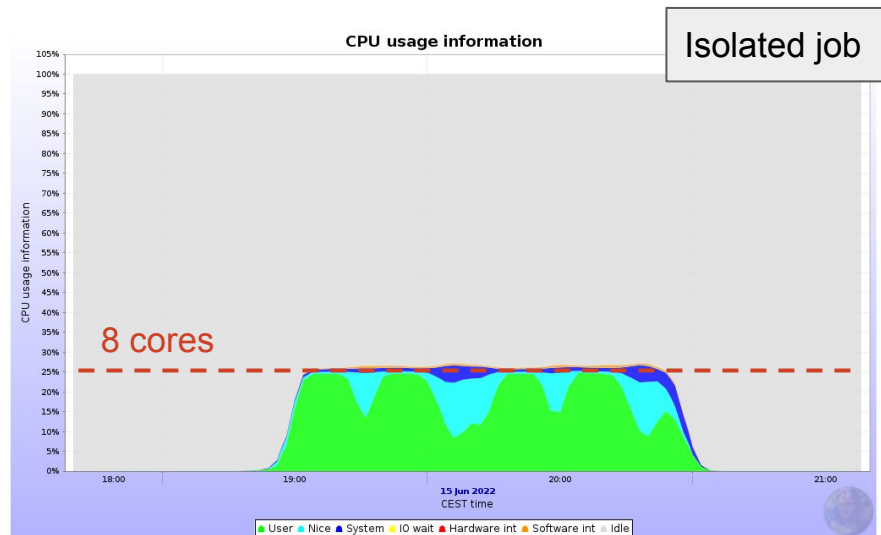
# The ALICE Grid

- Workflows are **overusing resources**
  - Many sites do not constrain job CPU allocation
  - Unpredictable and unfair behaviour towards other concurrently running applications
- Special concern when running in **whole-node scheduling** - we need to make the resource distribution ourselves
- We have implemented a CPU pinning-based mechanism to constrain the jobs' CPU usage using the `taskset` tool
  - CentOS7 is the currently recommended production version by WLCG
  - In the future with CentOS8, `cgroups v2` will be used

# Applying taskset for job isolation



- Total CPU usage goes above the requested 8 cores



- CPU consumption is limited with `taskset`
- Total CPU usage is flat at 8 cores

# Initial implementation from 1 year ago

- To pin specific CPU cores to jobs we use the `taskset` command
- When selecting the specific cores to pin to, we were doing this on a first-come first-serve basis. The first set of available cores were assigned to the job being scheduled
- This implementation has been running in production on the HPCS\_Lr site
  - HPCS\_Lr: HPC general-purpose cluster in the Lawrence Berkeley National Laboratory
- Further issue: potential inefficient use of CPU cache memory due to not adapting the selection to the NUMA topology



# Studying the impact of various core pinning strategies on CPU-bound payloads

# Test environment

## Two test nodes:

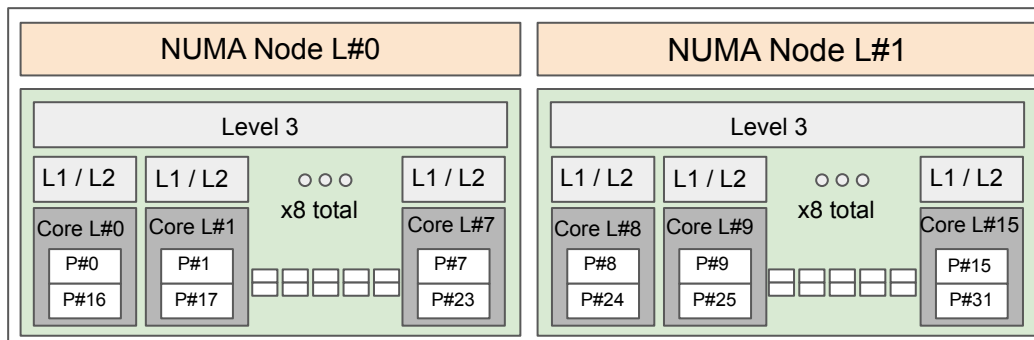
- 2x 16 HT CPU cores - Intel Xeon Gold 6244 @3.6GHz
- 2x 32 HT CPU cores - Intel Xeon Gold 6226R @2.90GHz

## Test results:

- Compatible for both machines, combined in the results slides

## Test conditions:

- Four or eight 8-core ALICE simulation jobs, all cores fully used



*Sample host CPU architecture*

# Tested configurations

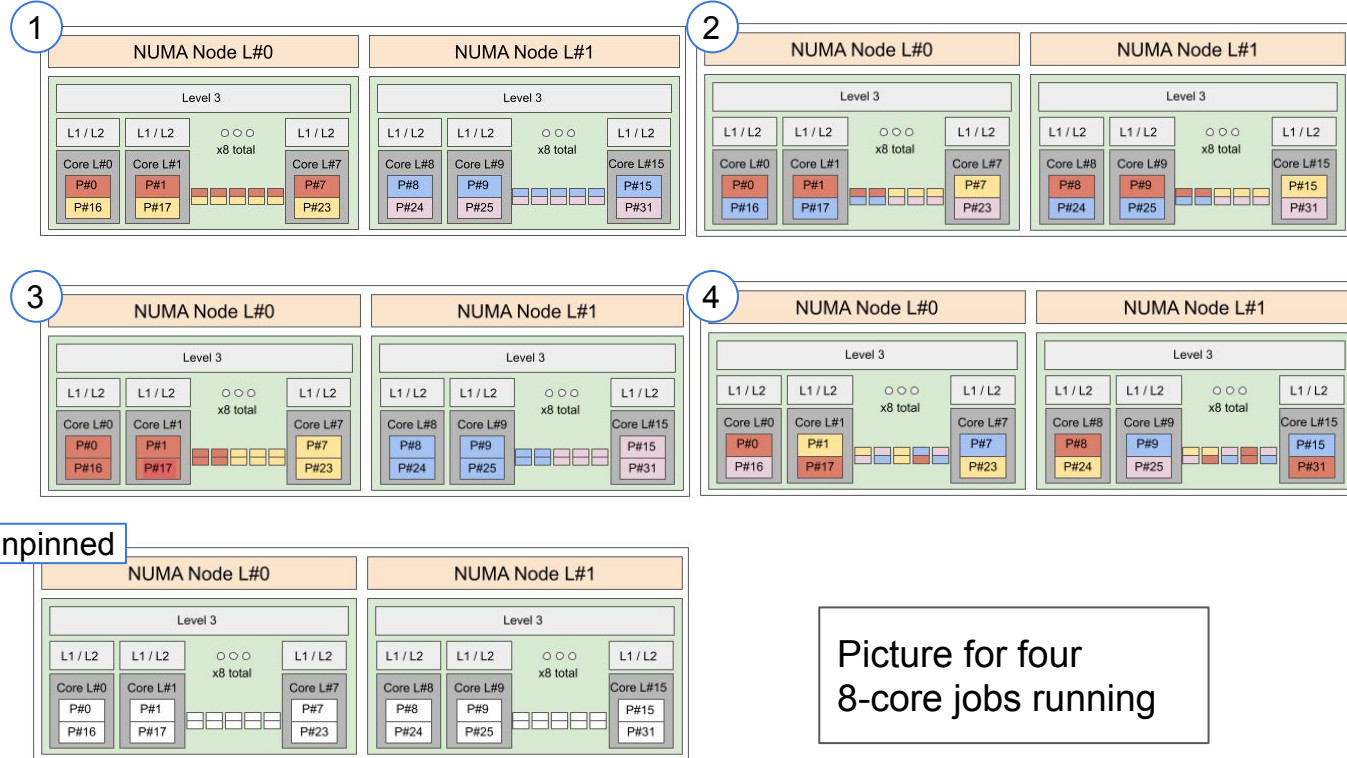
1. Same NUMA Node and independent L1,L2 cache
2. Different NUMA Nodes and independent L1,L2 cache
3. Same NUMA Node and sharing L1,L2 cache
4. Random core assignment
5. Baseline - no CPU pinning



Picture for single job running in idle machine

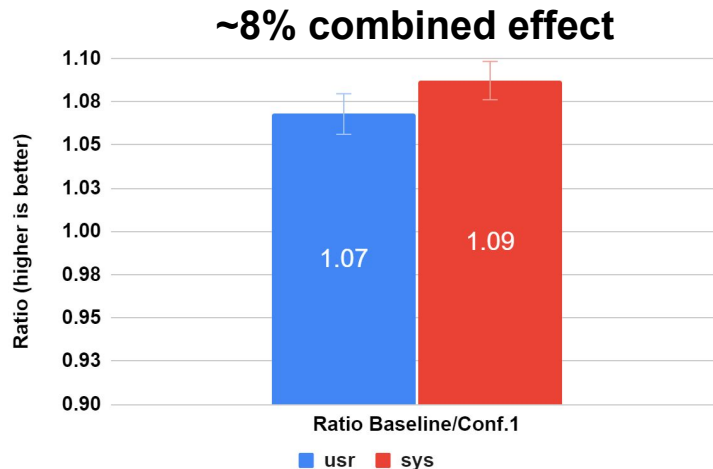
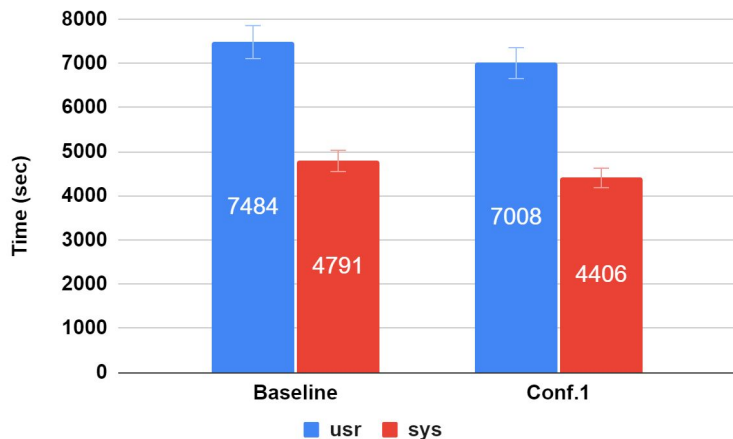
# Tested configurations

1. Same NUMA Node and independent L1,L2 cache
2. Different NUMA Nodes and independent L1,L2 cache
3. Same NUMA Node and sharing L1,L2 cache
4. Random core assignment
5. Baseline - no CPU pinning



# Configuration effects - usr and sys time

- Results shown for baseline (no pinning) and most optimal configuration 1 (same NUMA node, independent L1/L2 cache)
  - Configurations 2, 3 and 4 results fall in between these two



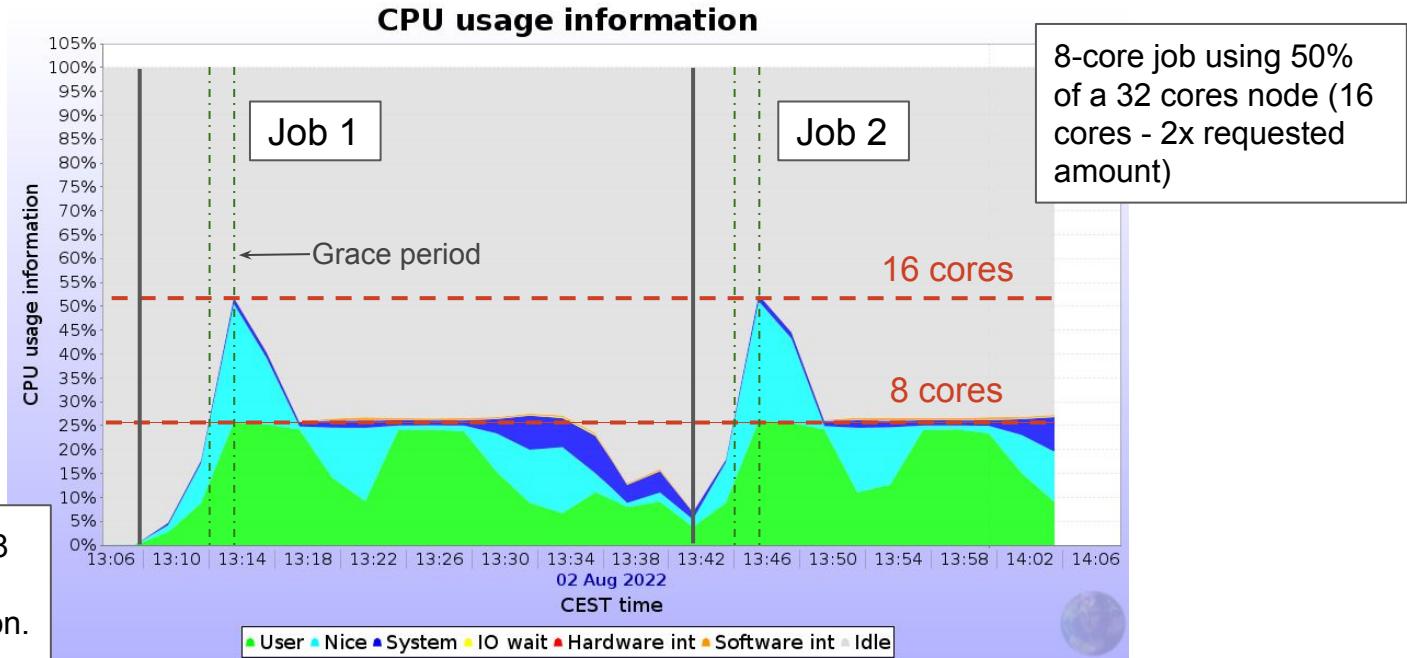


# Payload pinning implementation

# Hybrid strategy in production

- Jobs can expand when there are enough resources
- If overconsumption is detected they are constrained to the originally requested number of cores
  - o Strike a balance between job turnaround and fairness to other jobs
- Used the NUMA pinning in configuration 1 to exploit **data locality of NUMA nodes**
- Random selection of CPU core sets
  - o Start from the batch queue allocated cores, if any were allocated
  - o Defragmentation of NUMA nodes for optimal allocation

# CPU resource consumption on overconsuming job



# Summary

1. Pinning to the same NUMA node and not sharing L1,L2 caches **improves job CPU time - about 8% effect**
2. When running combined load, jobs do **benefit from isolation** - CPU consumption in System and User components is reduced
3. Results are compatible with other machine architectures
4. Tests performed on a realistic production scenario prove that we could **benefit from whole-node allocation combined with the most optimal pinning** applied to all running payloads
5. The CPU pinning implementation is already running in production