Whole-node scheduling in the ALICE Grid: Initial experiences and evolution opportunities

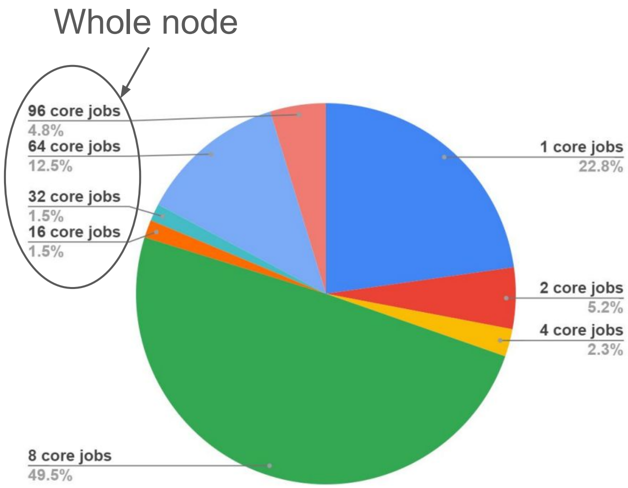
Marta Bertran Ferrer
marta.bertran.ferrer@cern.ch

On behalf of the ALICE Collaboration

CHEP 2024
October 21st-25th, 2024

ALICE Grid workload overview

- ALICE jobs with different resource requirements
  - Including CPU cores, memory, disk and accelerators
- All Grid sites are running **multicore queues**
  - 12 whole-node queues
  - All others 8+ CPU core queues
- JAliEn adapts to the different allocations
  - Automatically discover and scale to whole node resources where available
  - If fixed-size slot, limits inferred from cgroups v2 or batch queue settings
    - Bounding CPU and memory usage
  - Further partition slot resources within the job mix



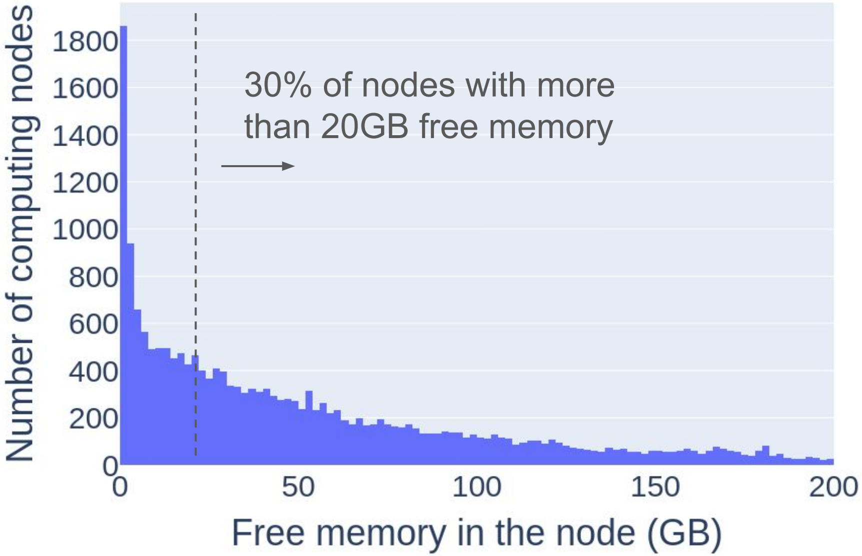
8-core vs whole-node queues

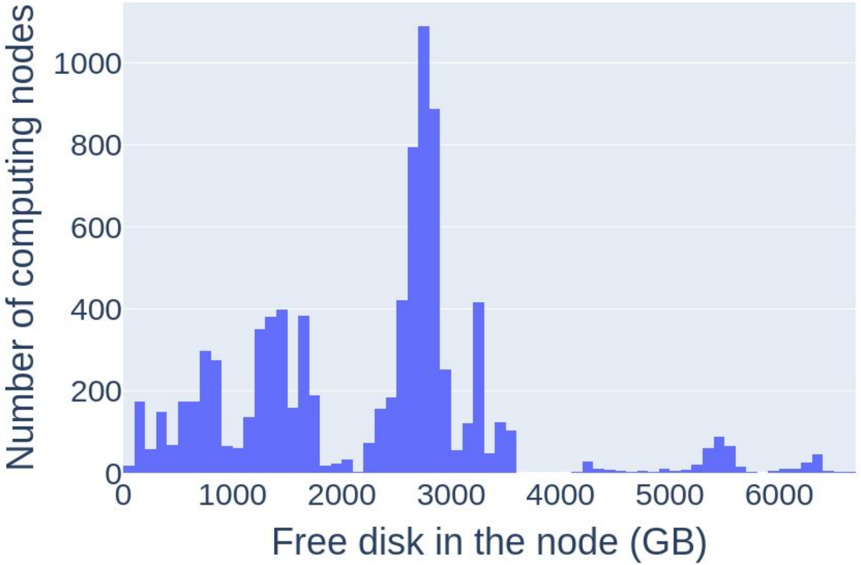
- 8-core queues are **current WLCG standard**
  - Uniform setting to support the multiple VOs sharing sites
  - Site-specific policies to partition and isolate jobs to their allocated slots
  - Jobs running on parallel slots are not aware of the impact among them
- **Whole-node** as our preferred strategy to accommodate varying computing and memory needs from job mixes
  - All ALICE-only sites are operating in whole-node submission
  - Flexibility and freedom to adapt to the running job mix
  - Optimising job scheduling within a node based on workflow type (CPU, IO or memory intensive)

Limitations of current scenario

- In the traditional accounting model, utilisation efficiency considers only CPU
  - Users are encouraged to fine-tune the CPU requests
  - While memory and disk are granted proportionally to cores
- On fixed-size slots, going over the default memory and disk requirements may lead to **job terminations**
  - Users should be aware of the limits and adapt their workflow
  - If more memory is needed, more cores have to be requested → lowering CPU efficiency
- Many Grid nodes with **more memory** than the default 2 GB/core
  - As workflows are confined to the tight bounds, nodes may be left with unused memory

Limitations of current scenario





Free resources on the 8-core slot hosts when we start a Job Runner (pilot)

Proposal for job brokering in whole-node

- On whole-node slot, **common pool** for resource management
  - Keeping track of allocated and free CPU cores, memory and disk space
  - Advertising available resources to late-binding job matching requests
  - Subtracted from the pool on job start and refilled upon completion
- **JDL tags** for each resource type
  - `CPUCores` (default 1)
  - `RequiredMemory` (default 2 GB * CPUCores)
  - `WorkDirectorySize` (default 10 GB * CPUCores)
- Allowing users to provide **accurate estimates**

ALICE

Proposal for job brokering in whole-node

Example requests of main physics workloads

Simulation	Analysis	Reconstruction
CPUCores=8	CPUCores=1	CPUCores=32
RequiredMemory=15GB	RequiredMemory=4GB	RequiredMemory=220 GB
WorkDirectorySize=50GB	WorkDirectorySize=1GB	WorkDirectorySize=50GB

- **Balance workloads** with complementary requirements
- The resources in the pool will vary with job mix

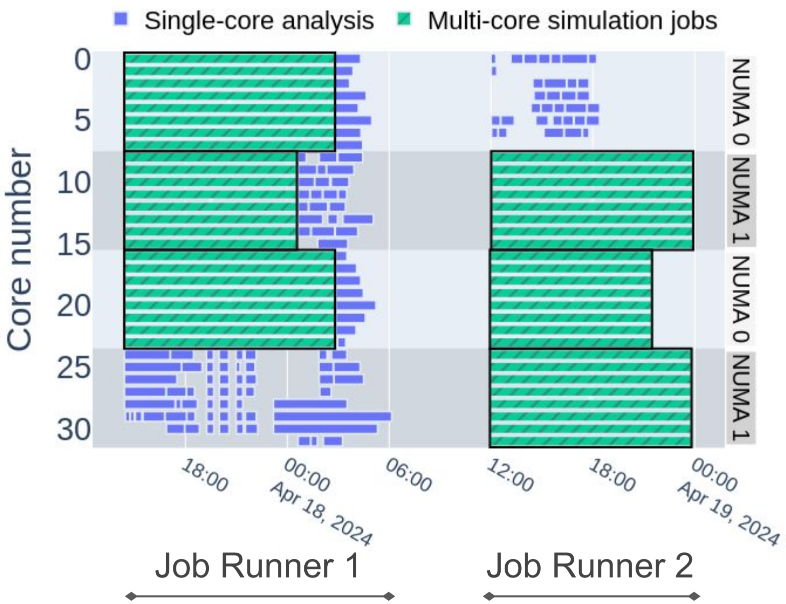
Optimising job placement

- Explicitly **bind jobs to CPU cores** to constrain executions
  - Going one level further on resource allocations
- Core selection based on node **NUMA architecture**
  - Optimal job to core mapping re-assessed at every job start and end
- Batch slots are **internally partitioned** to run optimal job mix
  - Mixing jobs with regards to CPU, memory and I/O requirements

ALICE

Optimising job placement

- **Job-to-core mapping** algorithm starts by jobs with larger allocations
  - Simulation jobs (8-core) are placed first and analysis (1,2,4-core) as backfill
  - Transparently balancing resource loads
- NUMA-aware pinning leads to improved **execution efficiency** [1]
- Predictable execution time fostering better **scheduling decisions**



[1] *CPU-level resource allocation for optimal execution of multi-process physics code*

Whole-node slot TTL extension

- Slots started with a **pre-defined TTL** (Time To Live) of the Job Agent
  - TTL included in job matching requests
  - Less jobs will match as TTL decreases
- Slot might be **partially used** when TTL is not enough to match any job
  - Low slot utilisation efficiency during draining period
- If whole node is dedicated to ALICE → Extending the slot TTL would **minimise draining effects**
  - Tokens need to remain valid during the whole slot duration
- Have experimented with 48 and 72h whole node slots
  - Our middleware adapts to the slot length and is ready to exploit extended TTL

CPU oversubscription in whole-node sites

- CPU oversubscription **already running** on some whole-node sites
  - Using idle CPU cycles for running additional jobs
  - Preempting jobs if CPU gets heavily overloaded
  - Additional pressure in CPU limited in time
- Enabled when enough [*idle CPU && free memory && free disk*]
- On most sites, **disk** is the main limiting factor
  - 10 GB of disk are reserved per CPU core (default reservation)
  - Can be larger if explicitly requested by jobs
  - VMs in particular seem created with this default value in mind
    - Limiting how many jobs can be executed in parallel
- Consider provisioning **>>10 GB * CPU cores** for new machines

ALICE

CPU oversubscription a whole-node site - UiB

- In peak Grid utilisation, **11%** of executed jobs are started by the oversubscription workflow
  - 5200 allocated CPU cores in 16-core whole-node slots
  - Over a 10-day study period : 12500 oversubscribed jobs out of 118000 total executions
- 13 minutes spent in oversubscription regime on average, then moved to regular resource pool
- **94.5%** of the oversubscribed jobs succeed
  - The remainder are preempted due to excessive pressure on the CPU usage

Conclusions

- Whole-node scheduling is our **preferred option** for resource allocation
  - Flexibility to accommodate heterogeneous resource demands from jobs
  - Converting sites across the Grid, 12 sites already moved to whole-node
- Brokering decisions now also consider **memory and disk**
  - With new syntax to express job requirements
- Improved job **placement and orchestration**
  - Exploiting complementary job resource usage patterns
- **Extended TTL** with robust resource management
  - Minimising the inefficiencies derived from slot draining
- **CPU oversubscription** on whole-node sites proving to have a significant impact
  - Worker-node disk space as a limiting factor