Multi-core resources management in ALICE

L. Betev, M. Litmaath

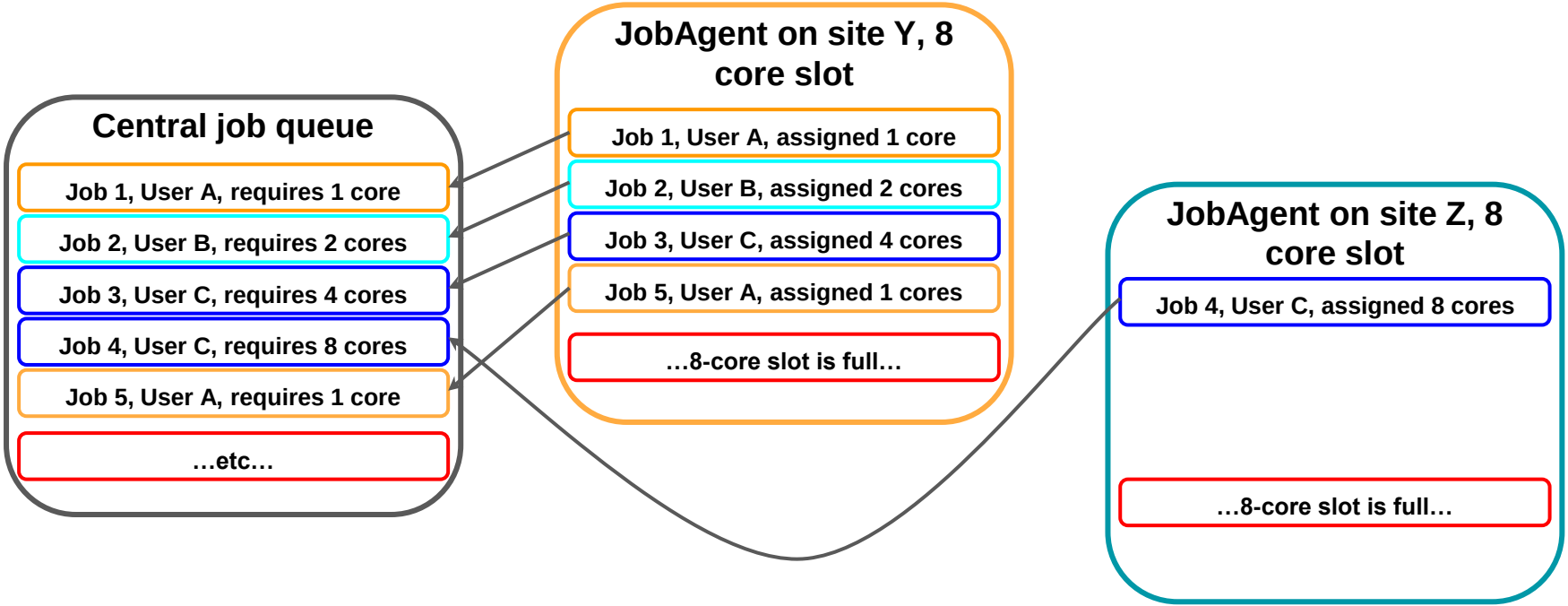
Single core - multi core

- **ALICE upgraded the detector and software for Run3/4**
  - Full rewrite of the online and offline software - multi core + GPUs
  - Full rewrite of the top-level Grid software stack - from AliEn to JAliEn
- Main challenges from Grid perspective
  - Assure smooth transition from single to multi core payloads, while keeping full support for legacy software
  - Make the transition as simple as possible for both the ALICE users and Grid sites - keep the interfaces to the Grid and on the sites unchanged

Update of the Job Agent model

- Simple life pre-Run3
  - One JA, one job on a single core, one user per JA
- Now
  - Anything from single to 8 core jobs in the queue for standard workflows + higher number of cores for GPU-enabled workflows (CTF reconstruction)
  - Multiple users could be combined in the same batch slot
  - The above must work in a standard multicore queue (8-cores) on the Grid
  - The mixed-cores-job functionality should be achieved on central task queue and JA level

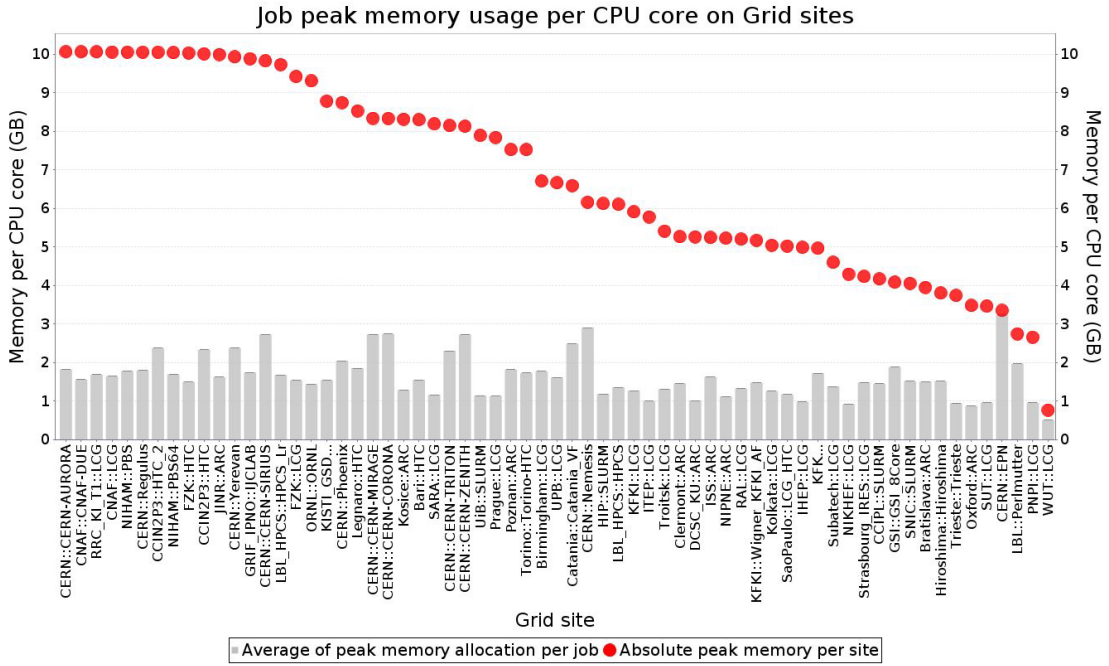
Very simplified JA functionality - core matching



- Not in the picture - matching on priorities, expected run time, memory requirements, data location, software versions

Additional considerations

- Different job requirements - every job is running in its own container
  - EL7, EL8, EL9 + custom containers with GPU enabled
- Non-uniformity of site resources, especially memory
  - Memory and CPU control of the payload - use of *cgroups v2, taskset* in production (and wherever available)



Why 8-core is not the best solution

- 8-core queue is ideal *only* for 8-core jobs
  - For any other job mix - there is not much real estate to play with
  - 24 hour max proxy could leave unused cores and reduce the overall efficiency
  - Not all jobs can use efficiently 8 cores, even if they are nominally 8-core
- The 8-core JA functionality is trivially extensible to any higher number of cores
- … to a whole node submission
  - Especially relevant for HPCs, where ‘whole node’ allocation is typical
  - Allows for more intricate and precise allocation of resources, for example
    - Mix of high I/O with high CPU consumption jobs (better balance)
    - Mix of high memory with lower memory requirement jobs (still fitting the overall envelope)
    - Full control of the CPU pinning - use of same NUMA domain*
    - Possibility to oversubscribe, in case of idle cores **
  - Makes the sysadmins jobs easy - no more queue management

** this pinning technique improves efficiency by ~5%*

*** oversubscription can provide up to ~15% additional resources*

Experience with multi-core and whole node

- On HPCs and ALICE only sites
  - We have asked the site to configure the batch system as whole node
- The JA detects the number cores and submits as many jobs as it could fit
  - Plus uses the other refinements described in the previous slide
- Still, the majority of resources are on 8-core queues

Service	Stat	Cores ▲
54. Perlmutter		256
23. EPN_MI100		96
22. EPN		64
78. UPB		16

Service	Stat	Cores ▼
26. FZK_KIT		0
33. HPCS_Lr		0
39. KISTI_GSDC		0
43. LBL_AFP		0
44. LBL_HPCS		0
48. NIHAM		0
51. ORNL		0
53. Pandora		0

ALICE

What is missing for wider adoption of whole node or larger than 8-core queues

- Compatibility of OS on the host
  - *cgroups v2* are only available and fully configurable on EL9
- Compatibility of batch system
  - Only HTCondor (versions >23) and Slurm (we developed a plug-in) support rootless encapsulation in *cgroups v2*
  - Considerable inertia of the Grid resources - not only OS updates, but configuration updates must be done at all sites
- Biggest issue - how to make all this work on shared sites where the job slots must be freed after a reasonable time
  - Whole node or >8 core queues will work better with >24 hour proxy (slot booked for several days)

Conclusions

- 8-core queues are nice, but have limitations with mixed workflow
- 16-core queues would make sense if the slot can be used for >>longer than 24-hour period
- The variability of workflows and resources require flexible allocation and workload management
  - Cores, memory, runtime, software packages, I/O…
  - More HPCs than ever in production
- The tools to manage arbitrary number of cores are ~there and can be incorporated in the VO Grid software
  - Payload isolation, CPU pinning, self-imposed memory limits, use of accelerators
- It would make sense to re-examine the relatively rigid (and old) resource fragmentation model and to move to a more open system
  - For example offer whole-node queues at large sites
  - Allow for long-lived (>>24h) proxy on whole node or even on 8-/16-core queues