

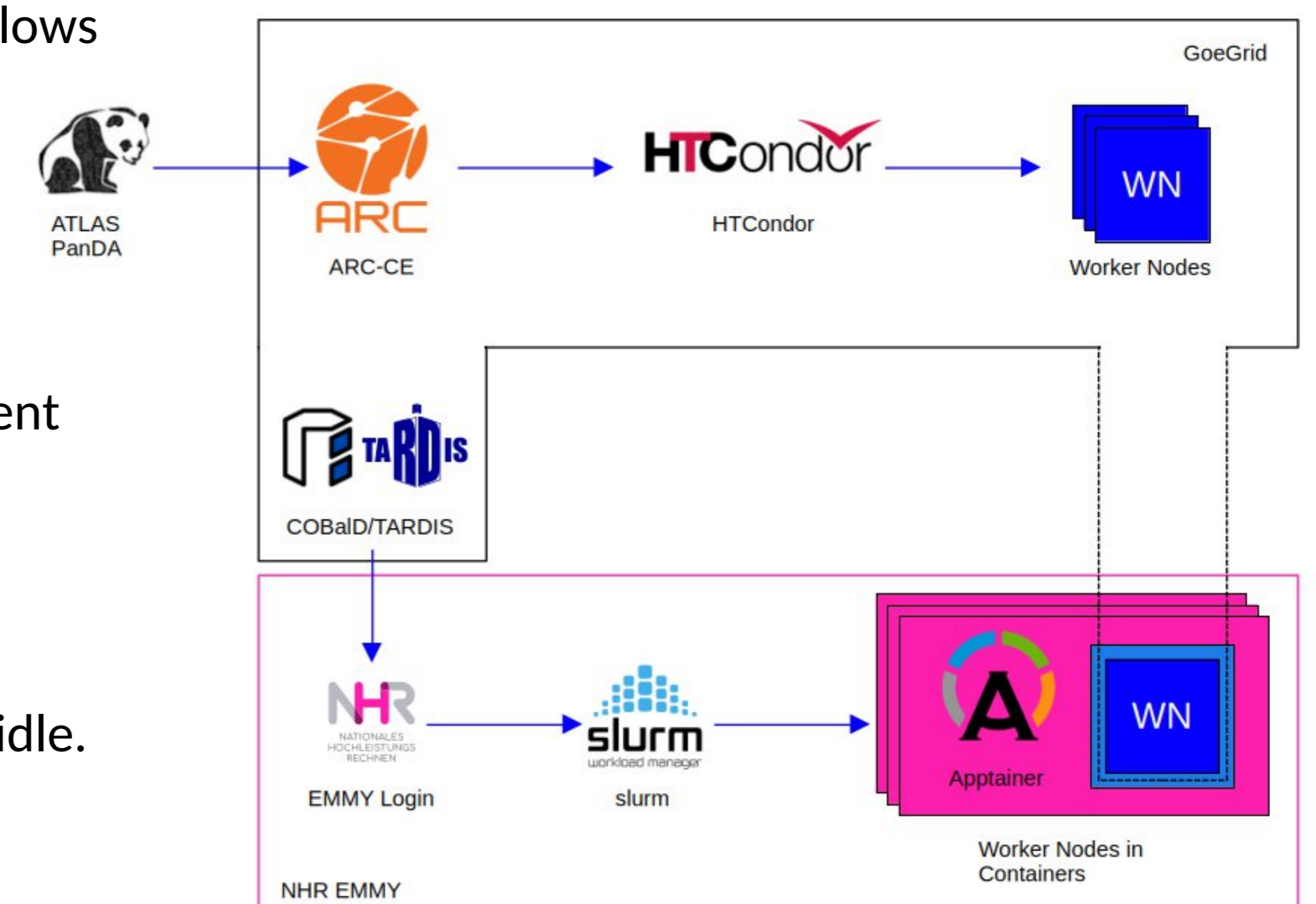
Background jobs for recovering CPU usage during draining phases: ATLAS use case

Maximilian Horzela¹, Inga Lakomic¹, Ughur Mammadzada¹, Arnulf Quadt¹, Rodney Walker², Sebastian Wozniewski¹

1: Georg-August-Universität Göttingen, 2: Ludwig-Maximilians-Universität München

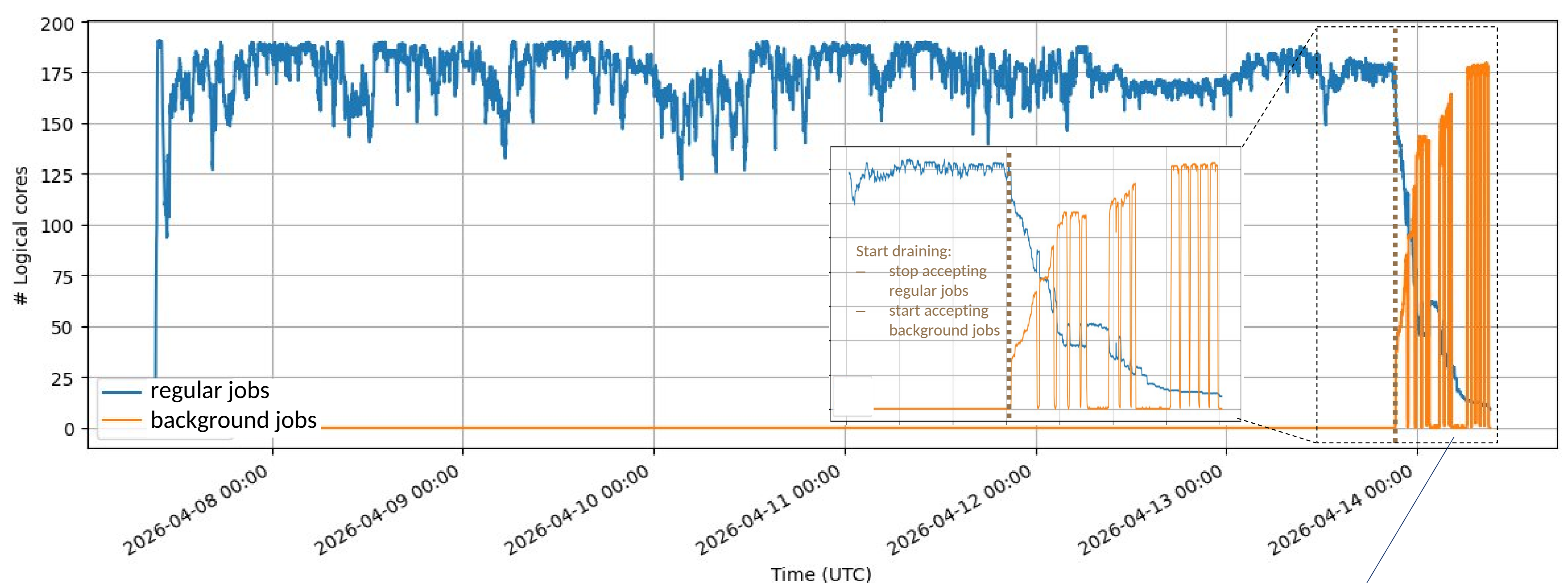
Motivation: Opportunistic HPC resources — NHR-Cluster Emmy Göttingen

- HPC cluster Emmy of the NHR alliance (National High Performance Computing) made accessible to ATLAS workflows via the **overlay batch system approach**.
- Meta scheduler COBalD/TARDIS submits jobs to host batch system, which starts virtual worker nodes (WN) connecting to the batch system of the local WLCG Tier-2 centre GoeGrid (similar to pilot jobs).
- Solves issues related to node partitioning under whole-node-scheduling policies and provides proper environment with cvmfs in user space.
- Cluster policy determines maximum job wall time and hence **maximum life time of virtual WNs** (7 days here).
- Approaching the end of life, **WNs are drained** in order to keep number of aborted jobs low, which leaves cores idle. **Background jobs reduce the amount of lost CPU power.**



Concept of background jobs

- **Highly parallelised jobs** with enough threads to fill an entire worker node (in our case: ATLAS detector simulation configured to 192 threads).
- Priority on CPU given to regular jobs → background jobs **fill remaining cores**.
- The more cores it can use, the **shorter** the background job **runtime** - finally order of minutes.
- **Allows for termination** of virtual worker node almost **any minute**.
- Reduces number of interrupted regular jobs while using free CPU capacities → **Minimising loss of CPU power!**



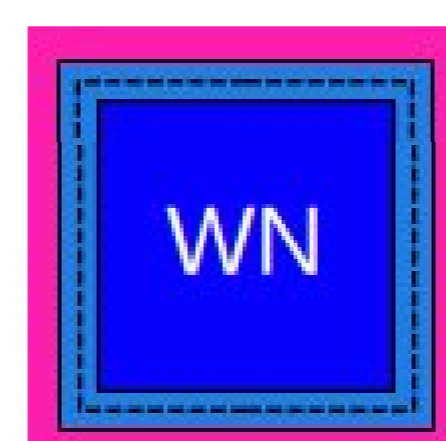
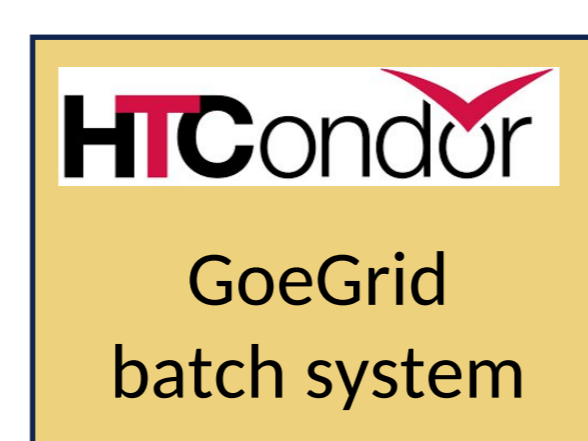
Tuning of number of parallel background jobs needed to fill scheduling gaps while keeping termination time low.

Other use cases possible, e.g. backfilling of slots reserved to prioritised jobs

Technical Implementation at GoeGrid & Emmy

Job submission and CPU management

Dedicated PanDa queue submits 192-core simulation jobs.



Dedicated ARC queue with modified forwarding to HTCondor batch system:

- requesting only 1 core
- requires 'bkg. resource'
- assigns job attribute as bkg. label

HTCondor controls CPU usage of jobs via cgroups by default:

- soft limit
- bkg. job is granted 1 core but can use more if not used by other jobs

Virtual worker nodes require cgroups in user space:

- cgroups v2 for containers
- HTCondor v23.9.6 or later (CREATE_CGROUP_WITHOUT_ROOT)
- Admins delegate cgroup

- Number of bkg. jobs controlled via custom defined resource in HTCondor (like CPU cores), e.g. 1 or 2 bkg. slots (more flexibility than with static slots).
- Acceptance of regular or bkg. jobs on the worker node steered via START condition and job label.

Accounting

- By default, ARC-CE would assume 192 cores and HTCondor 1 core.
- True core usage somewhere in between and is time-dependent → number of used cores can only be estimated a posteriori.

Unknown in advance if background job runs fast on many cores or slow on few cores!

- WLCG / EGI accounting based on wall time and number of cores (CPU time used to calculate CPU efficiency as secondary number).
- Simulation jobs run at CPU efficiency close to 100% (little I/O) → assume this to calculate the number of cores rounded up to a set of allowed values, e.g. 1, 2, 8, 16, 24, 32, ..., 192, and adjust in ARC database.