Karlsruhe Institute of Technology

Dynamic Compute Resource Integration at/by GridKa Tier 1

MU Days November 2021, virtual

Max Fischer, on behalf of Matterminers, GridKa Team and Collaborators

Helmholtz Research Field Matter

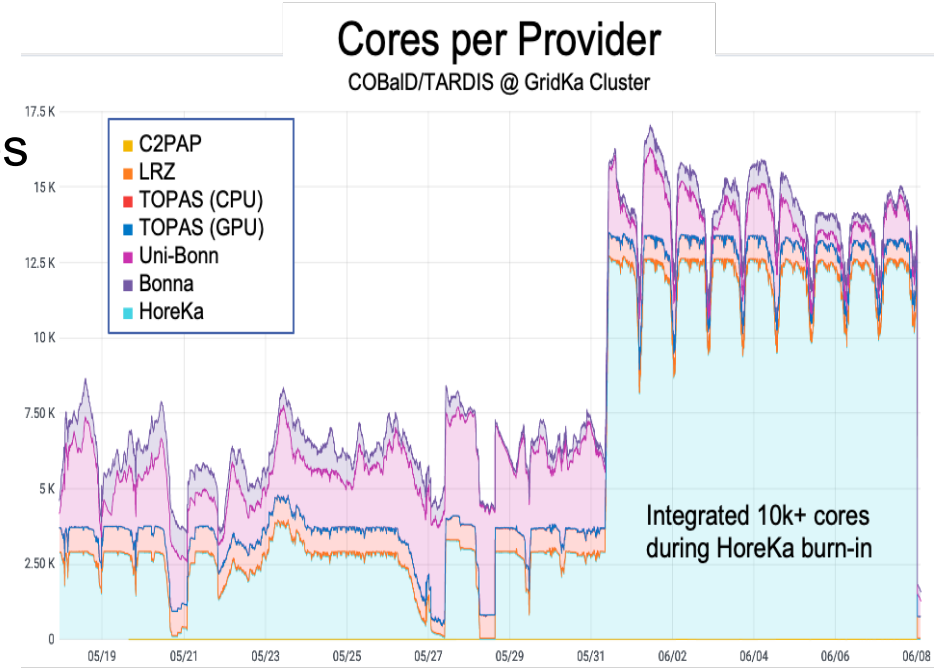
GridKa: Compute and Collaborate

- WLCG Tier 1 Center
  - Compute Center with large scale batch, storage and archival infrastructure
  - Grid Services to connect to hundreds of other grid sites
- Collaboration Contacts
  - Directly work with members and working groups of scientific collaborations
  - Active part in technology development and research

combined expertise for
large scale systems,
reliable distributed services, and
specific needs of scientific collaborations

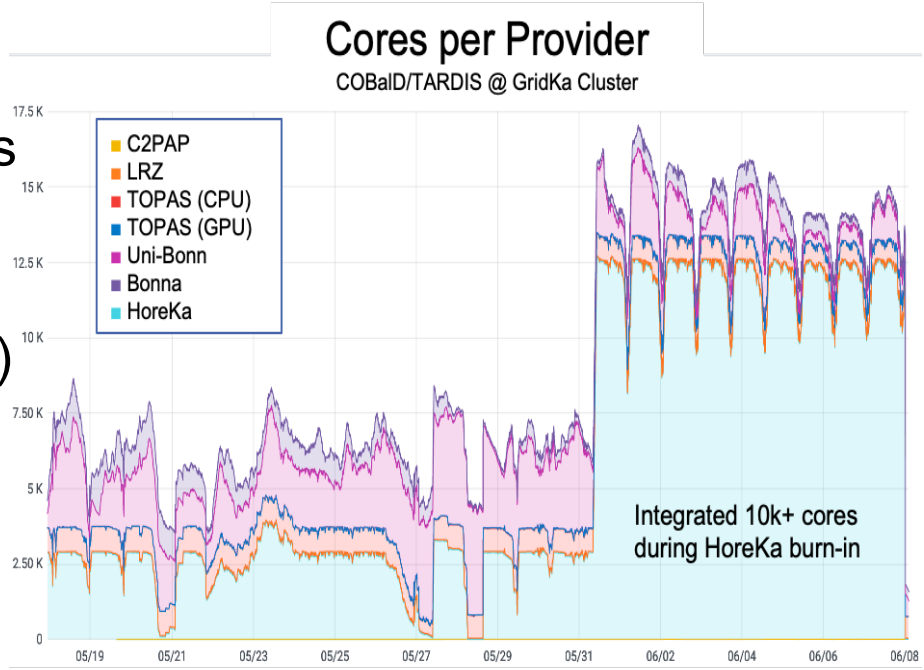
Synergies in Practice: Opportunistic Resources

- Opp. Resource Pool @ GridKa
  - Separate batch system with volunteered external resources
  - Presented as regular WCLG compute element to users



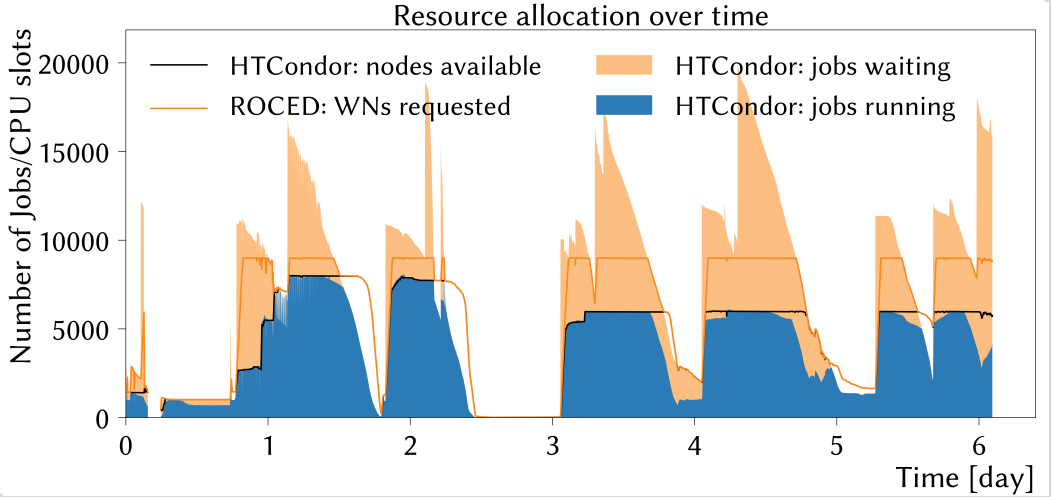
Synergies in Practice: Opportunistic Resources

- Opp. Resource Pool @ GridKa
  - Separate batch system with volunteered external resources
  - Presented as regular WCLG compute element to users
- Powered by COBalD/TARDIS (C/T)
  - Open Source software for dynamic resource acquisition
  - Integrate compute resources from HPC, Cloud, …



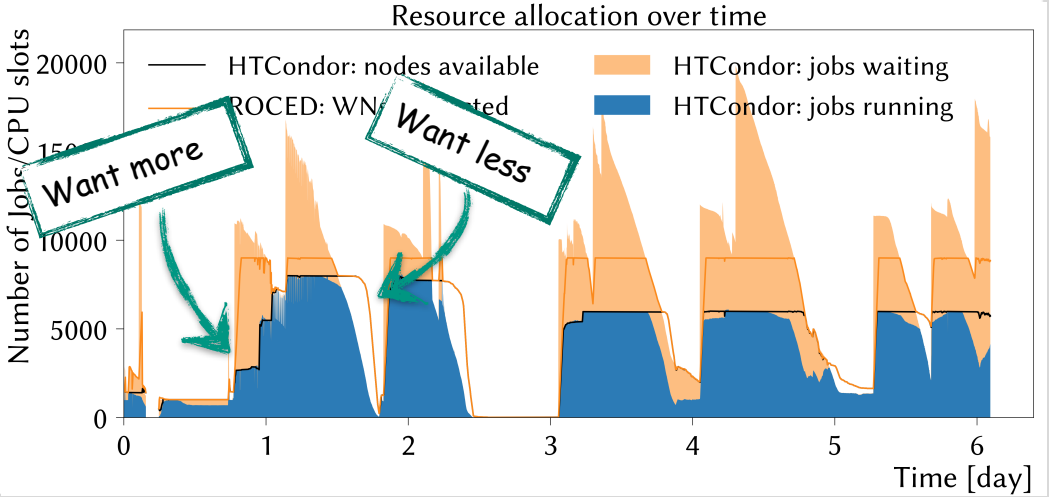
COBalD/TARDIS approach to Opp. Resources

- Resource Meta-Scheduler for Job Scheduler
  - C/T acquires resources from HTC, Cloud, … via pilot jobs
  - Challenge to match resources to need of actual job scheduler
  - Many pitfalls in practice: fairshares, latencies, black hole nodes, …



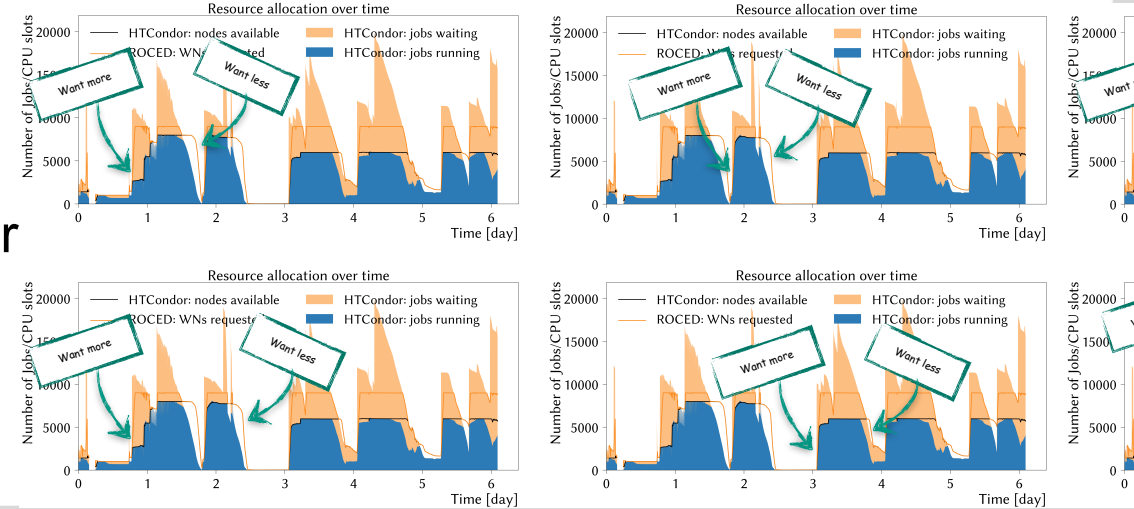
COBaID/TARDIS approach to Opp. Resources

- Resource Meta-Scheduler for Job Scheduler
  - C/T acquires resources from HTC, Cloud, … via pilot jobs
  - Challenge to match resources to need of actual job scheduler
  - Many pitfalls in practice: fairshares, latencies, black hole nodes, …
- In short: Feedback Loop
  - Seamlessly adjusts to existing schedulers



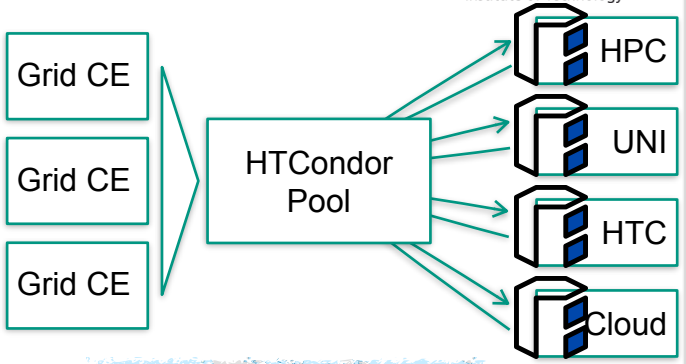
COBalD/TARDIS approach to Opp. Resources

- Resource Meta-Scheduler for Job Scheduler
  - C/T acquires resources from HTC, Cloud, … via pilot jobs
  - Challenge to match resources to need of actual job scheduler
  - Many pitfalls in practice: fairshares, latencies, black hole nodes, …
- In short: Feedback Loop
  - Seamlessly adjusts to existing schedulers
  - Separate instances per provider contribute to single resource pool



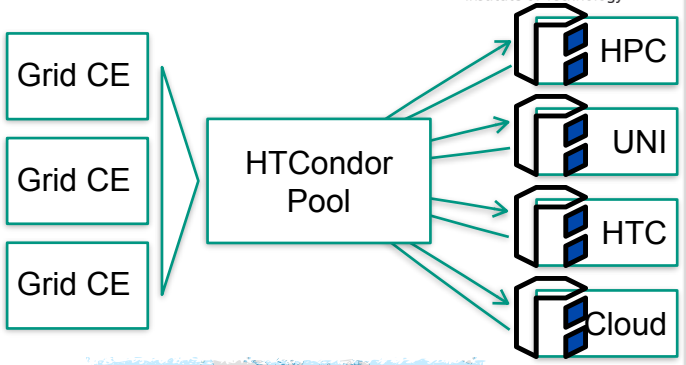
National Resource Pool via GridKa

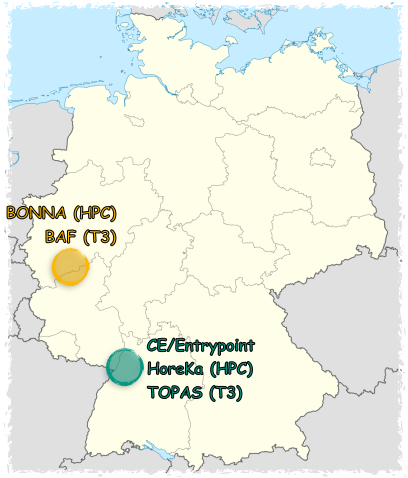
- Overlay Batch System hosted at GridKa
  - Batch and Grid Services as for Tier 1
  - Frontend, authentication, scheduling, …



National Resource Pool via GridKa

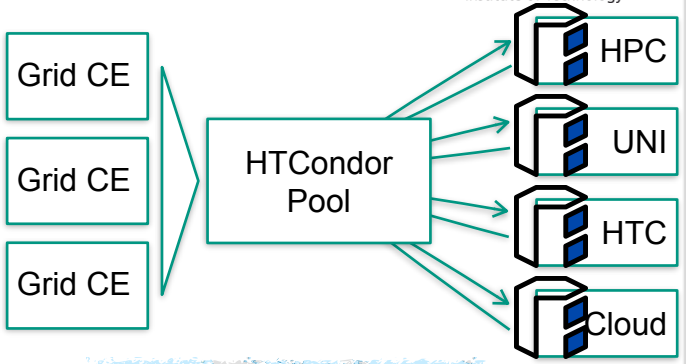
- Overlay Batch System hosted at GridKa
  - Batch and Grid Services as for Tier 1
  - Frontend, authentication, scheduling, …
- HoreKa (KIT) – Backfilling
  - Slurm+Singularity, cvmfsexec on DFS
  - Reaping idle compute resources
- Bonn T3 & BONNA – ATLAS/Belle2 Share
  - Managed entirely by Bonn contacts
  - Expose HPC & Tier3 to collaboration

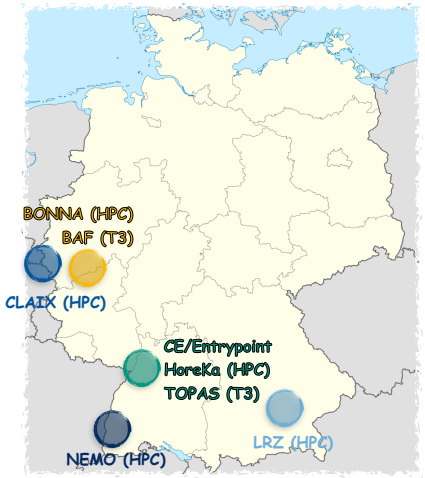




National Resource Pool via GridKa

- Overlay Batch System hosted at GridKa
  - Batch and Grid Services as for Tier 1
  - Frontend, authentication, scheduling, …
- HoreKa (KIT) – Backfilling
  - Slurm+Singularity, cvmfsexec on DFS
  - Reaping idle compute resources
- Bonn T3 & BONNA – ATLAS/Belle2 Share
  - Managed entirely by Bonn contacts
  - Expose HPC & Tier3 to collaboration
- Simplifying the compute landscape by collaboration and expertise

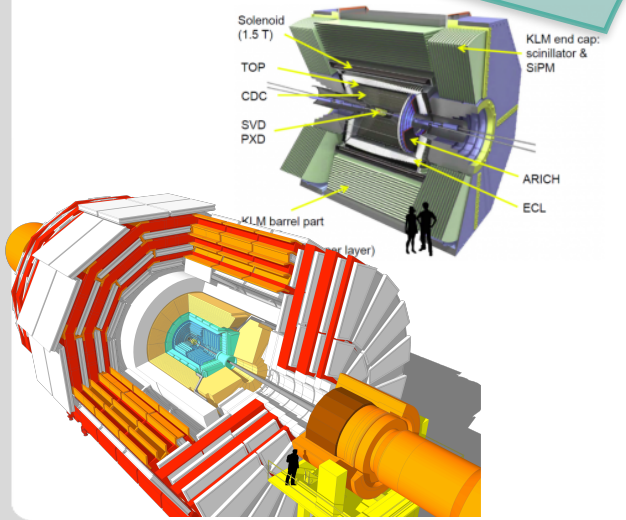




Backup

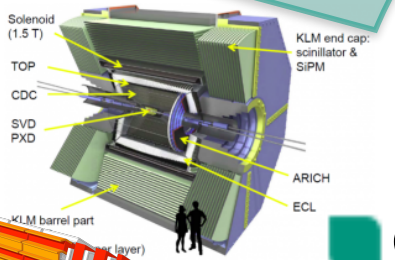
The High Energy Physics / WLCG use-case

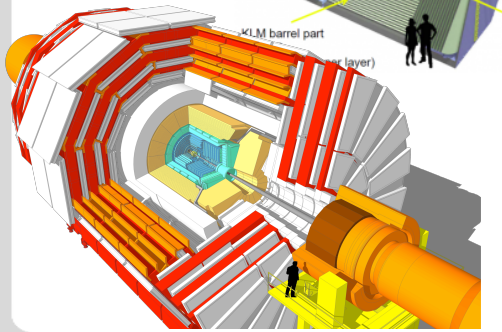
- Particle detectors record physics event data
- Each detector used by a collaboration of scientists

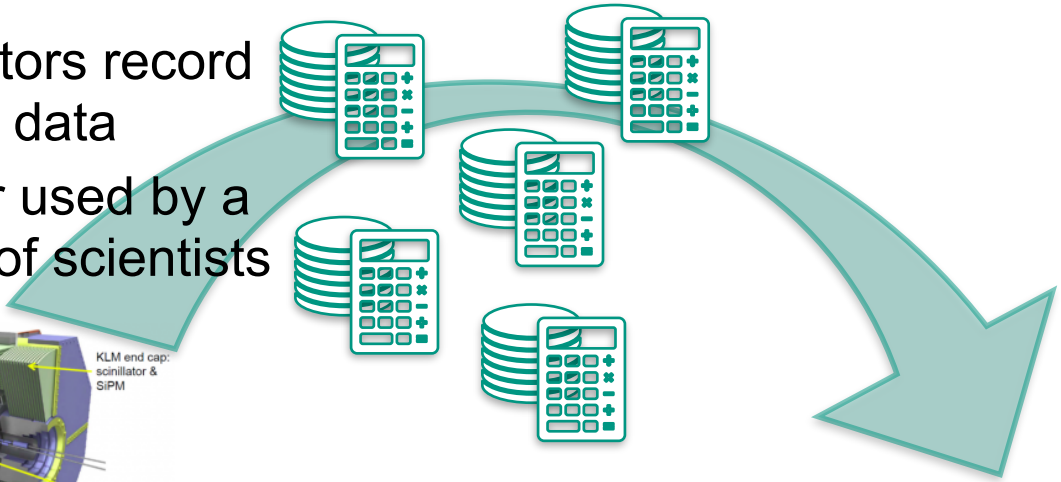


The High Energy Physics / WLCG use-case

- Particle detectors record physics event data
- Each detector used by a collaboration of scientists



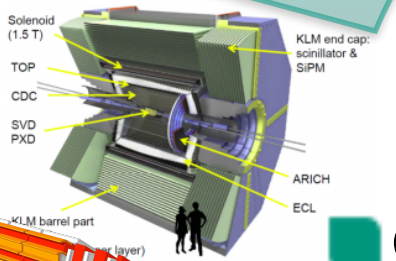


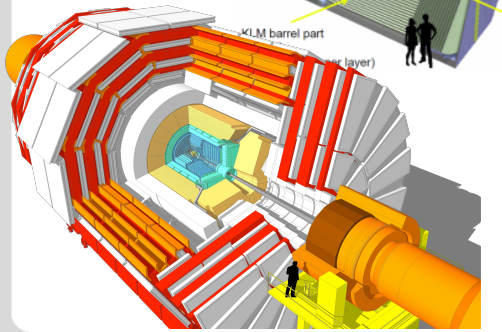


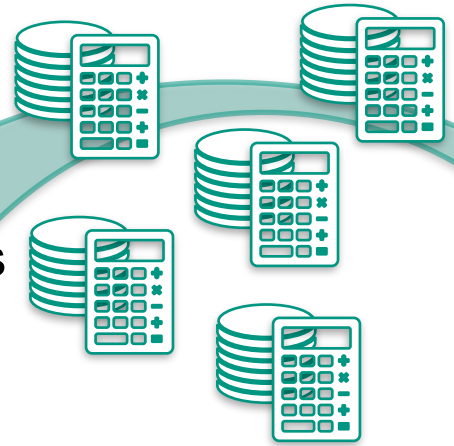
- Collaborators provide storage/compute centres
- Resources shared via worldwide computing Grid

The High Energy Physics / WLCG use-case

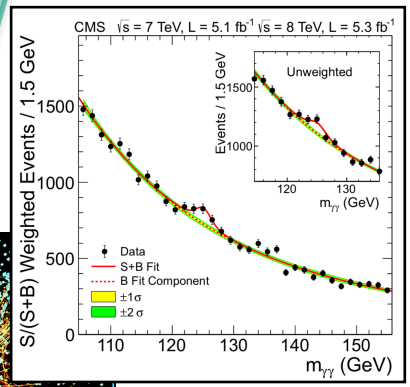
- Particle detectors record physics event data
- Each detector used by a collaboration of scientists

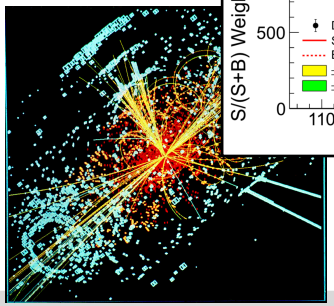






- Collaborations automate common pre-processing
- Scientists run individual end-user analyses

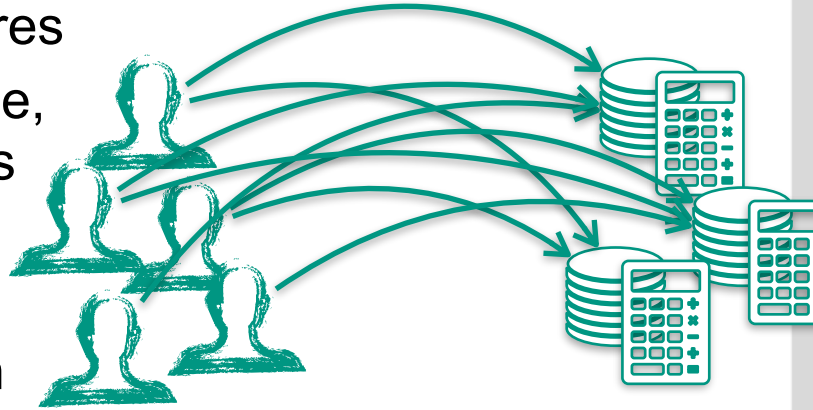




- Collaborators provide storage/compute centres
- Resources shared via worldwide computing Grid

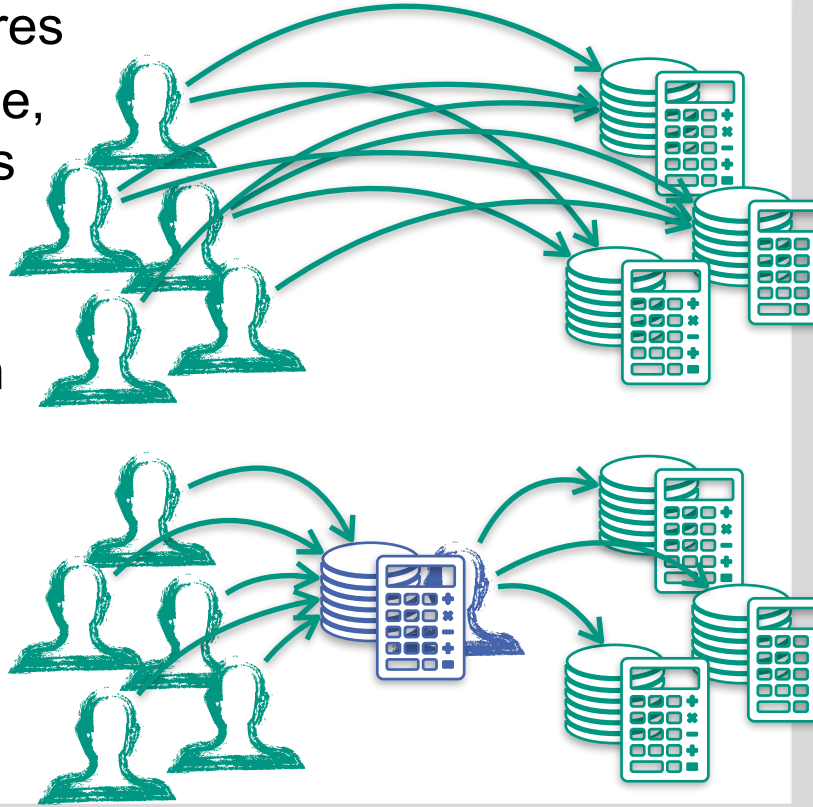
Distributed Computing: The Pilot Model

- Challenge: computing across many centres
  - Each centre with separate wait queue, policies, constraints to use resources
  - Direct, manual submission requires expertise and incurs heavy latencies
  - => Waste of resources and time with ~100+ centres and 10k+ scientists



Distributed Computing: The Pilot Model

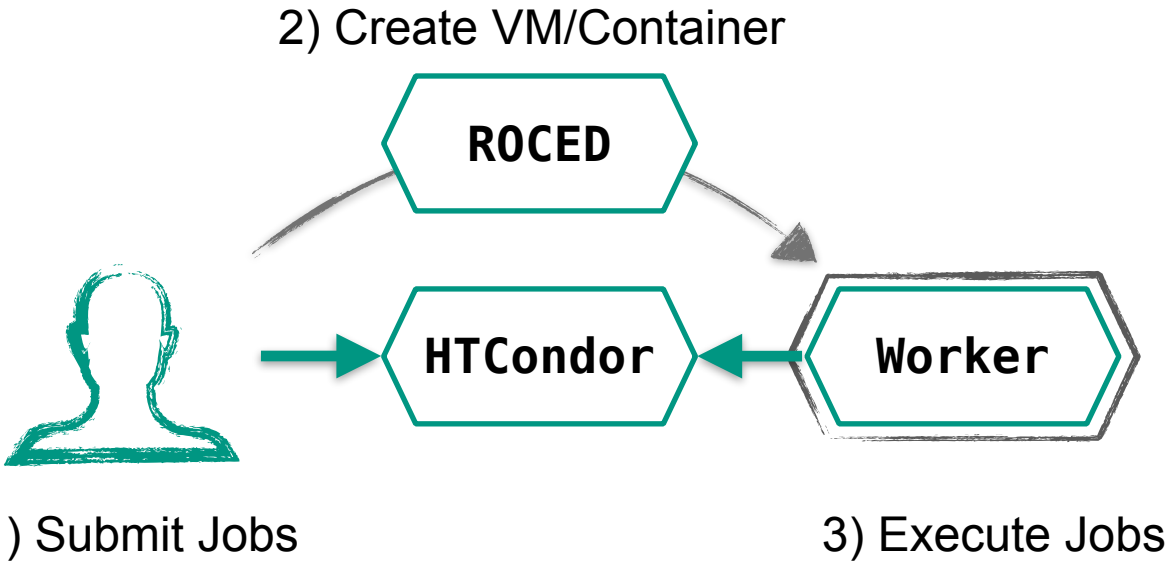
- Challenge: computing across many centres
  - Each centre with separate wait queue, policies, constraints to use resources
  - Direct, manual submission requires expertise and incurs heavy latencies
  - => Waste of resources and time with ~100+ centres and 10k+ scientists
- Solution: “Pilot jobs” reduce complexity
  - One batch system across centres
  - One group-user across scientists
  - Abstraction of resources and people



Case Study: ROCED (2010-2018)

[**R**esponsive **O**n-**D**emand **C**loud **E**nabled **D**eployment]

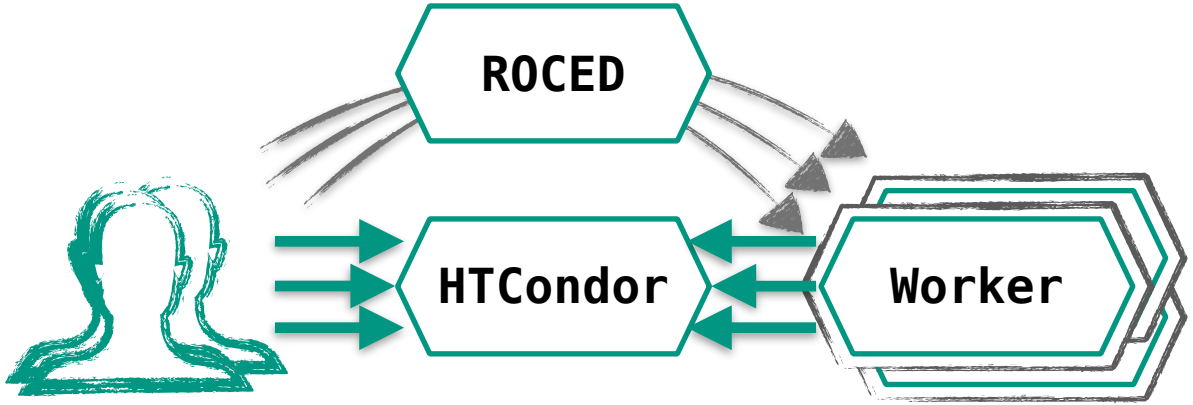
- Classical Job to Resource to Job Meta-Scheduler



Case Study: ROCED (2010-2018)

[**R**esponsive **O**n-**D**emand **C**loud **E**nabled **D**eployment]

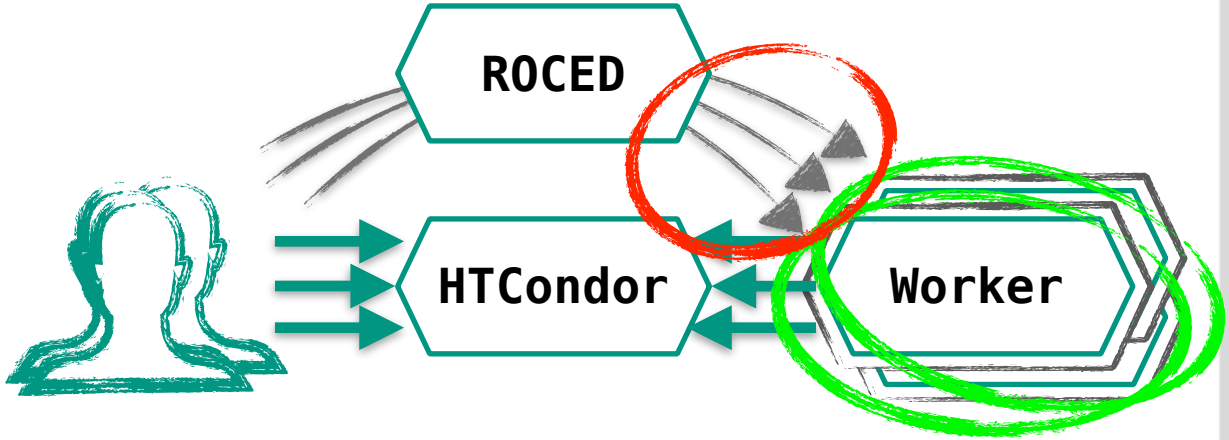
- Classical Job to Resource to Job Meta-Scheduler



Case Study: ROCED (2010-2018)

[**R**esponsive **O**n-**D**emand **C**loud **E**nabled **D**eployment]

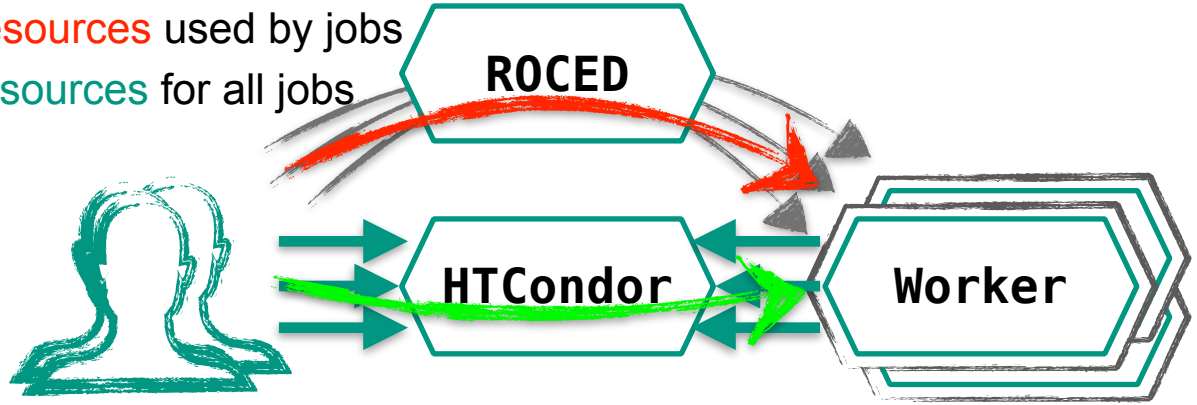
- Classical Job to Resource to Job Meta-Scheduler
- Dynamic resource acquisition matching user demand
  - Trivial to support new providers for many users
  - Difficult to manage several providers for many users



Case Study: ROCED (2010-2018)

[**R**esponsive **O**n-**D**emand **C**loud **E**nabled **D**eployment]

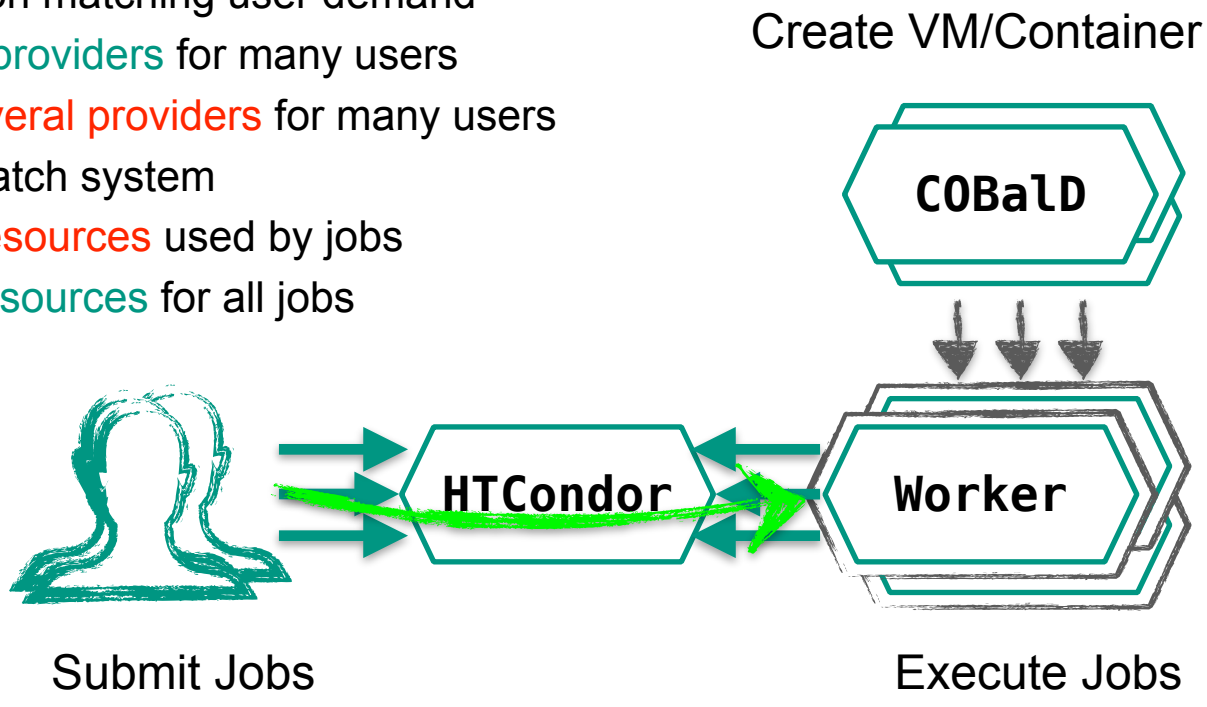
- Classical Job to Resource to Job Meta-Scheduler
- Dynamic resource acquisition matching user demand
  - Trivial to support new providers for many users
  - Difficult to manage several providers for many users
- Job scheduling in overlay batch system
  - Unreliable to predict resources used by jobs
  - Efficient to integrate resources for all jobs



Case Study: ROCED (2010-2018)

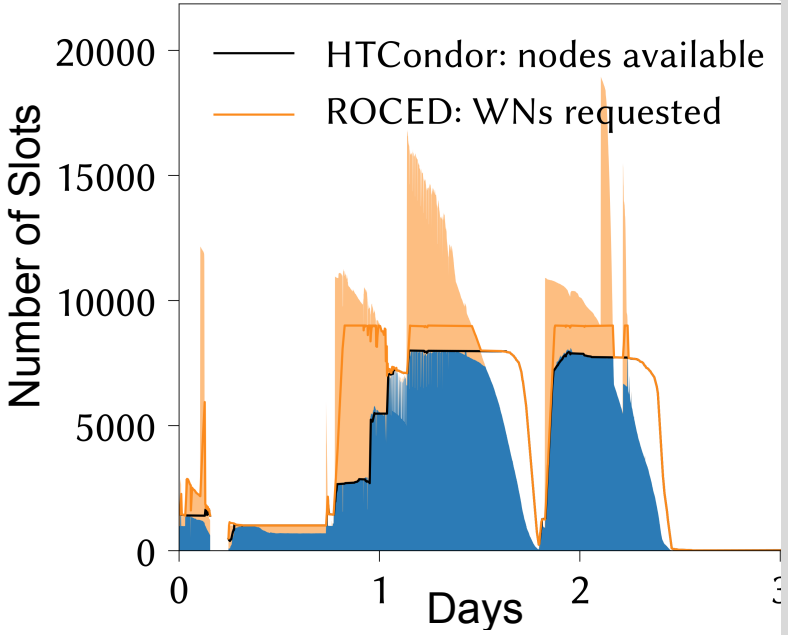
[**R**esponsive **O**n-**D**emand **C**loud **E**nabled **D**eployment]

- Classical Job to Resource to Job Meta-Scheduler
- Dynamic resource acquisition matching user demand
  - Trivial to support new providers for many users
  - Difficult to manage several providers for many users
- Job scheduling in overlay batch system
  - Unreliable to predict resources used by jobs
  - Efficient to integrate resources for all jobs



Simpler Approach: COBalD (2018-present)

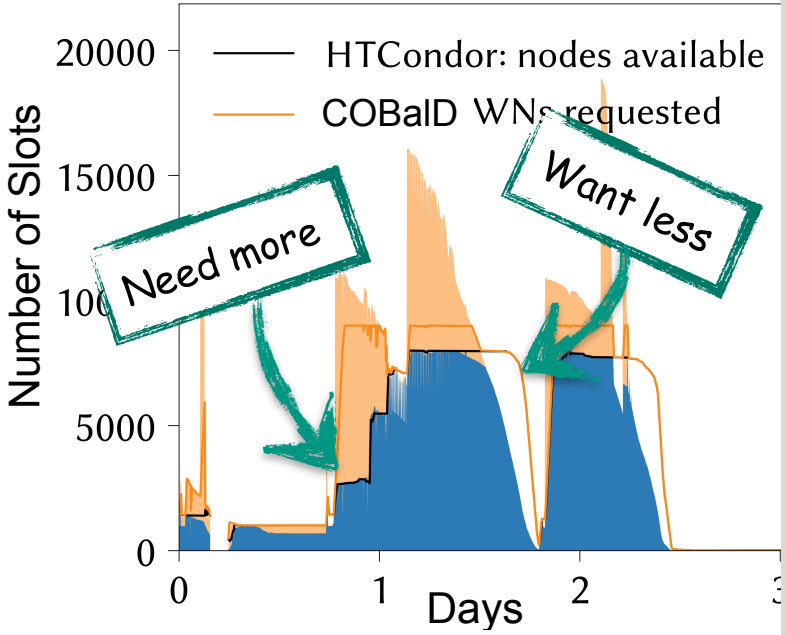
[**C**OBalD - the **O**pportunistic **Bal**ancing **D**aemon]



Simpler Approach: COBalD (2018-present)

[**C**OBalD - the **O**pportunistic **Bal**ancing **D**aemon]

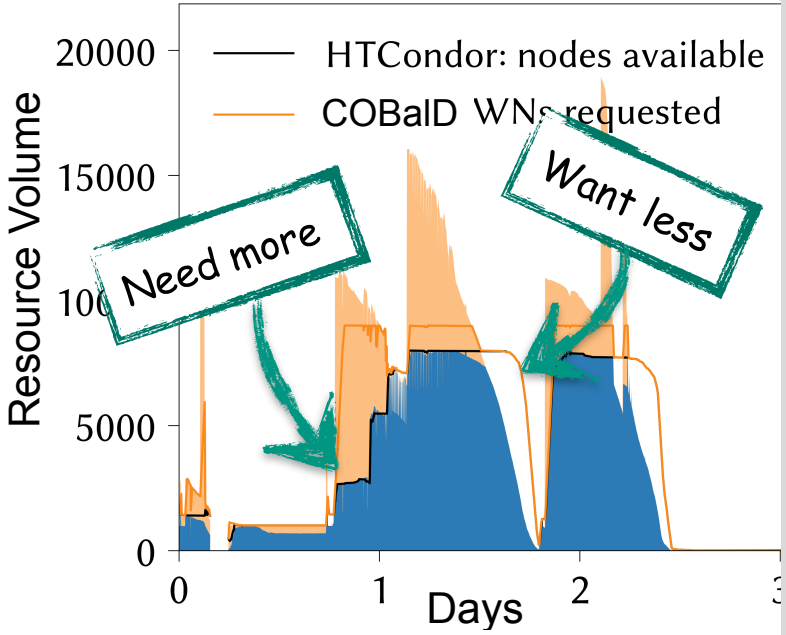
- Look at what is used, not what is requested
  - Simple logic: more used, less unused resources
  - Only batch system scheduler handles jobs
  - COBalD only acquires/releases resources



Simpler Approach: COBalD (2018-present)

[**C**OBalD - the **O**pportunistic **Bal**ancing **D**aemon]

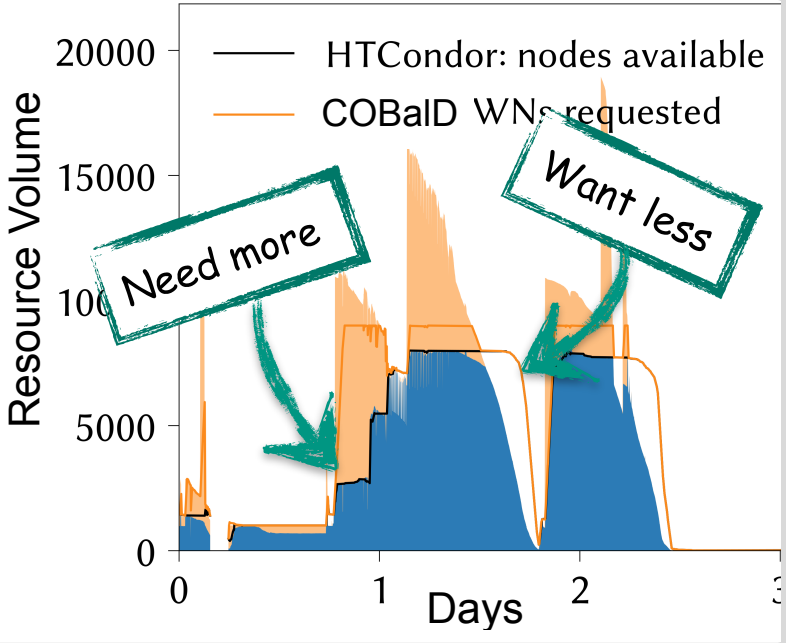
- Look at what is used, not what is requested
  - Simple logic: more used, less unused resources
  - Only batch system scheduler handles jobs
  - COBalD only acquires/releases resources
- Generic design for any resources
  - COBalD just knows (un-)used
  - CPU, CPU+RAM, CPU+DISK, CPU+DISK+RAM, …
  - Virtual multi-core slot partition



Simpler Approach: COBalD (2018-present)

[**C**OBalD - the **O**pportunistic **Bal**ancing **D**aemon]

- Look at what is used, not what is requested
  - Simple logic: more used, less unused resources
  - Only batch system scheduler handles jobs
  - COBalD only acquires/releases resources
- Generic design for any resources
  - COBalD just knows (un-)used
  - CPU, CPU+RAM, CPU+DISK, CPU+DISK+RAM, …
  - Virtual multi-core slot partition
- HTC integration via COBalD/TARDIS
  - Define VM/Container/Job as resource
  - Supports any use-case that can be put into a VM/container/script!

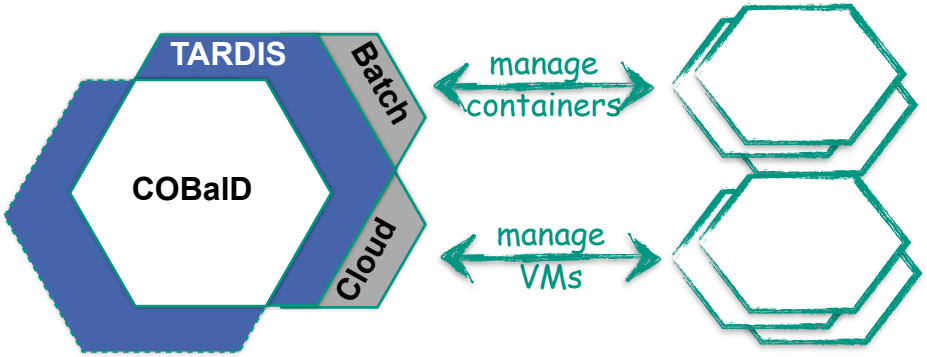


TARDIS: Out-of-the-Box Resource Adapters

- Combine resource provider APIs with COBalD
  - Request, monitor, decommission individual resources
  - Automatically match demand via COBalD approach
  - Basically a “use-case agnostic autonomous Pilot factory”

TARDIS

OpenStack
CloudStack
HTCondor
Moab
Slurm
“other”



TARDIS: Out-of-the-Box Resource Adapters

- Combine resource provider APIs with COBalD
  - Request, monitor, decommission individual resources
  - Automatically match demand via COBalD approach
  - Basically a “use-case agnostic autonomous Pilot factory”
- Support for common HPC batch systems, Cloud APIs, …
  - Behave like “regular users” as much as possible
  - Customisable payload for each centre’s peculiarities
  - HEP: insert HTCondor+CVMFS as available

TARDIS

OpenStack
CloudStack
HTCondor
Moab
Slurm
“other”