Karlsruhe Institute of Technology

GridKa

Lightweight dynamic integration of opportunistic resources

Computing in High Energy Physics 2019, Adelaide

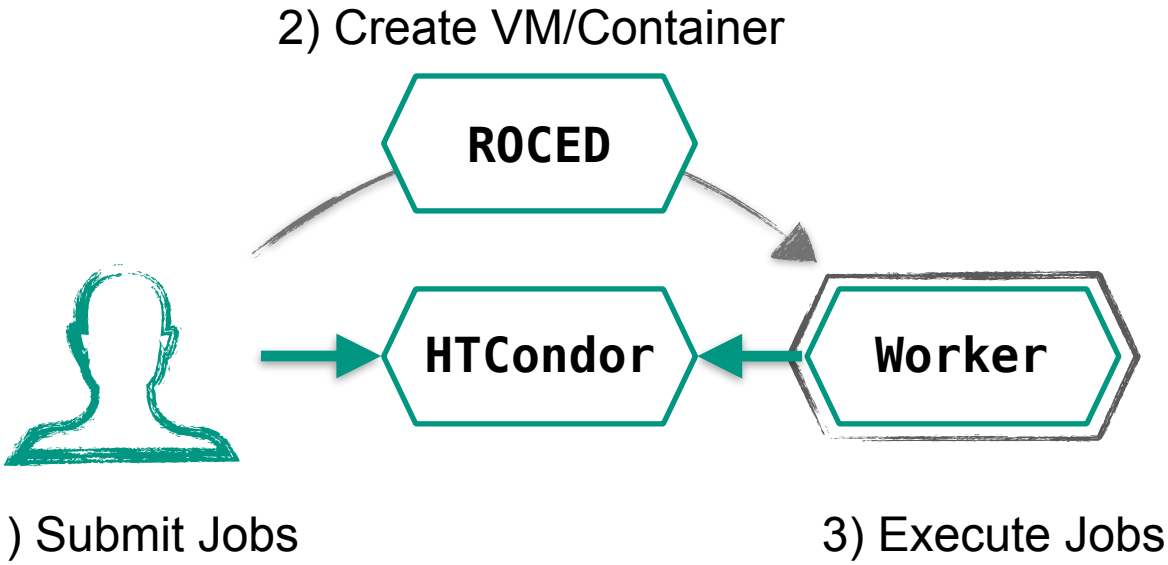
Max Fischer, Eileen Kühn, Manuel Giffels, Matthias Schnepf, Andreas Petzold, Andreas Heiss

Helmholtz Research Field Matter

Case Study: ROCED (2010-2018)

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

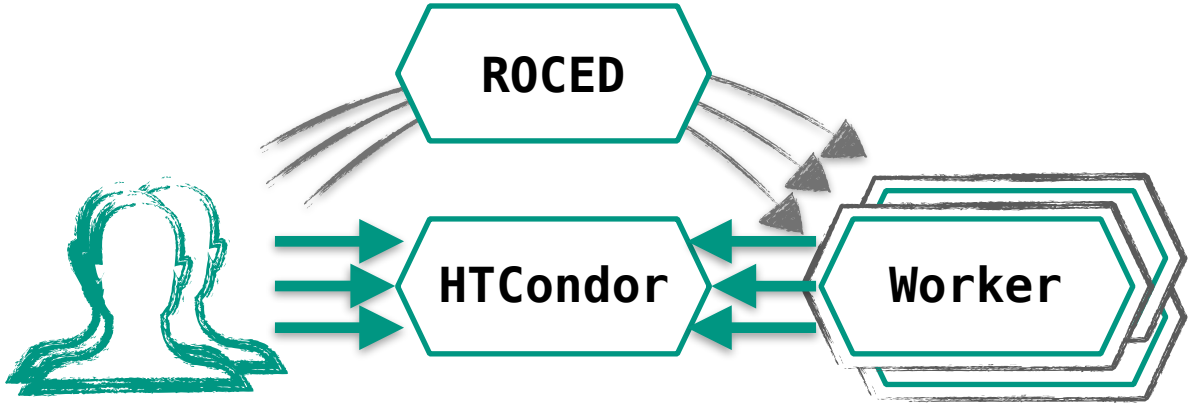
- Classical Job to Resource to Job Meta-Scheduler



Case Study: ROCED (2010-2018)

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

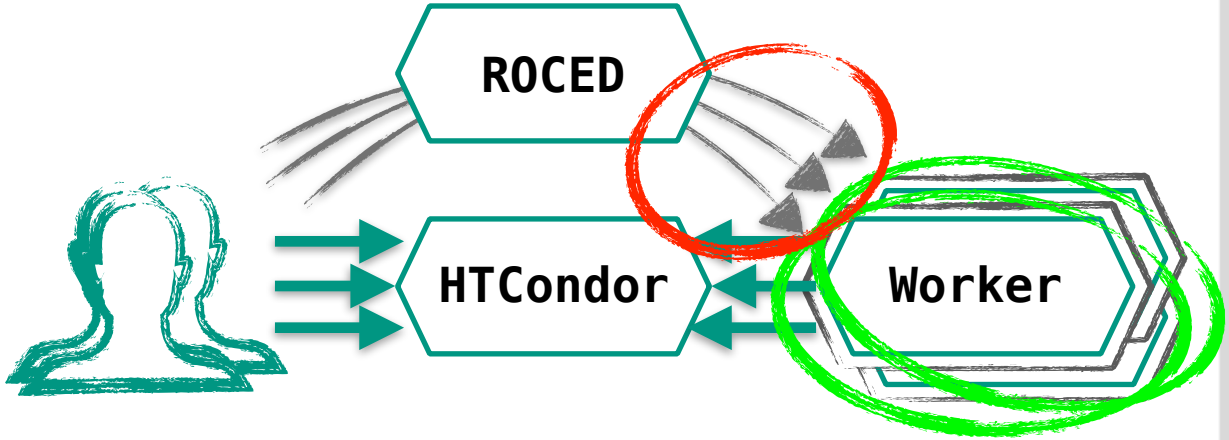
- Classical Job to Resource to Job Meta-Scheduler



Case Study: ROCED (2010-2018)

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

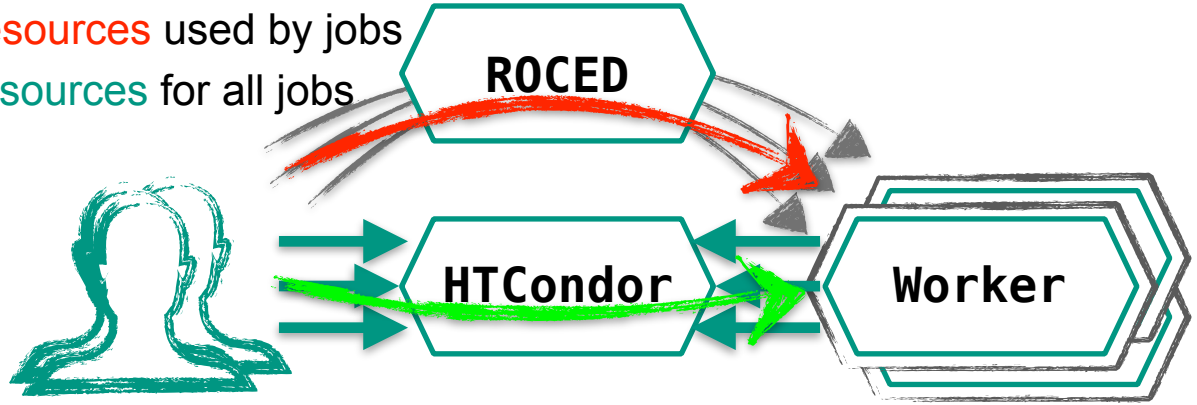
- Classical Job to Resource to Job Meta-Scheduler
- Dynamic resource acquisition matching user demand
  - Trivial to support new providers for many users
  - Difficult to manage several providers for many users



Case Study: ROCED (2010-2018)

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

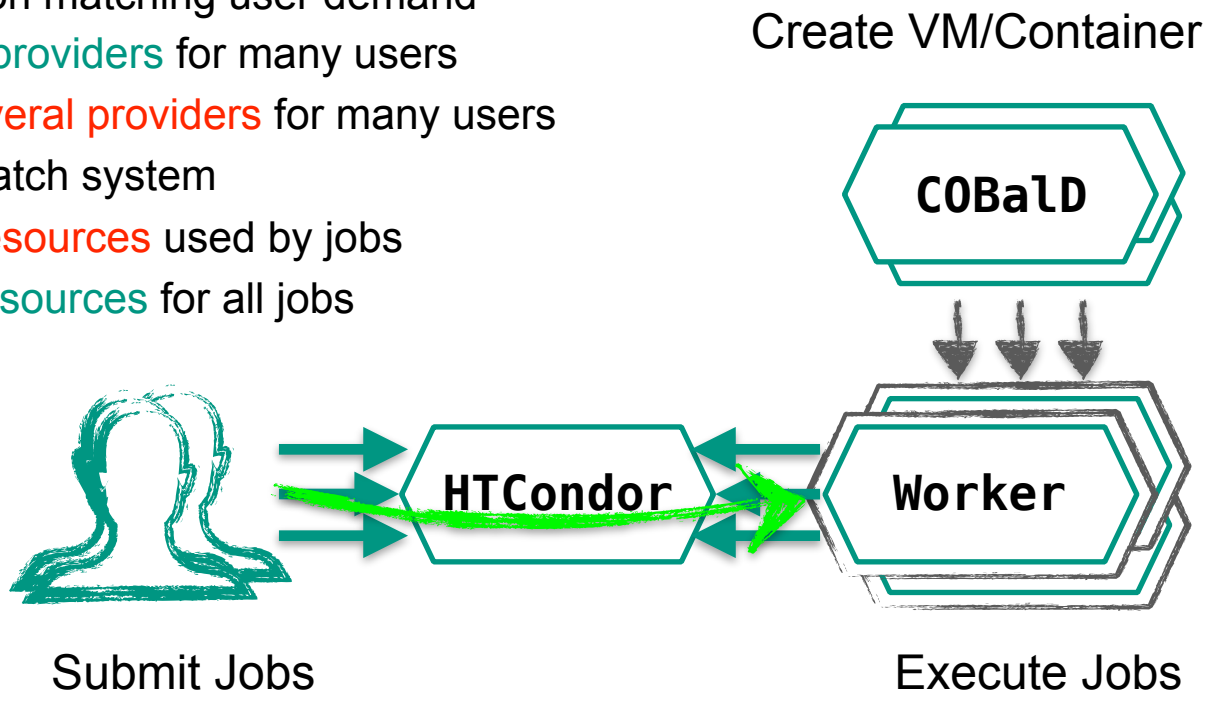
- Classical Job to Resource to Job Meta-Scheduler
- Dynamic resource acquisition matching user demand
  - Trivial to support new providers for many users
  - Difficult to manage several providers for many users
- Job scheduling in overlay batch system
  - Unreliable to predict resources used by jobs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



Case Study: ROCED (2010-2018)

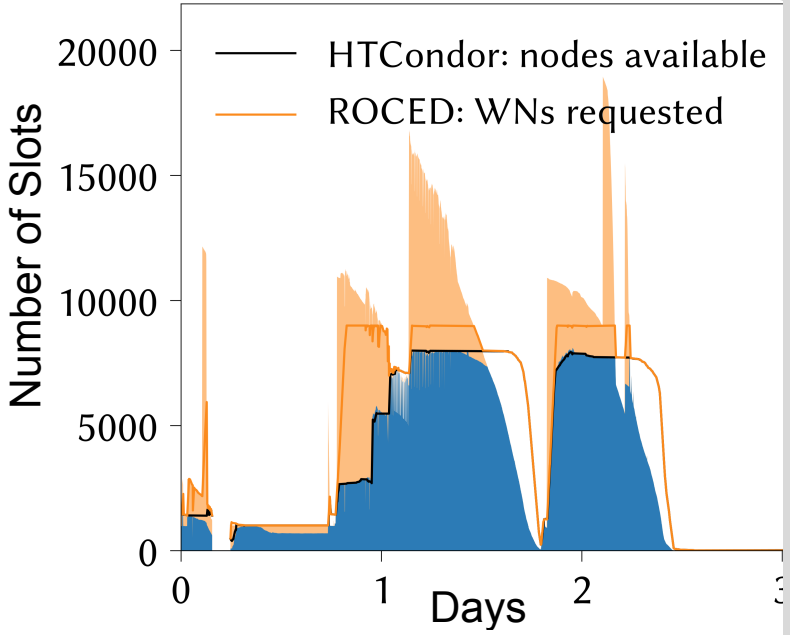
[**R**esponsive **O**n-**D**emand **C**loud **E**nabled **D**eployment]

- 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)

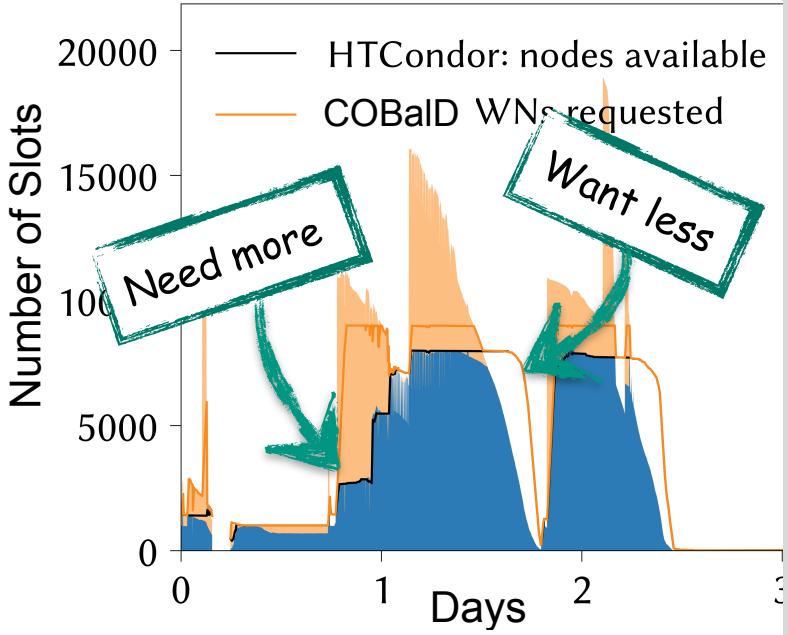
[**C**OBalD - the **O**pportunistic **Bal**ancing **D**aemon]



Simpler Approach: COBalD (2018-present)

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

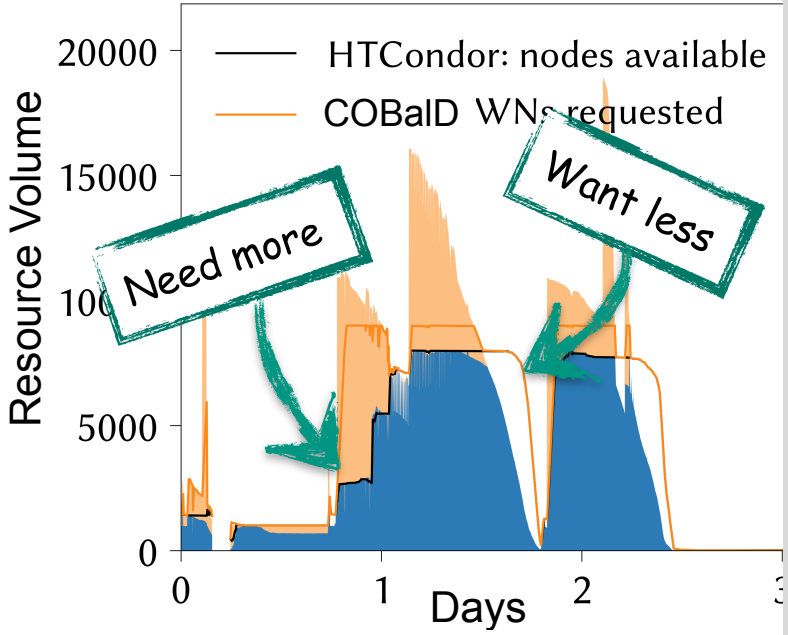
- Look at what is used, not what is requested
  - Simple logic: more used, less unused resources
  - Only batch system scheduler handles jobs
  - COBalD only acquires/releases resources



Simpler Approach: COBalD (2018-present)

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

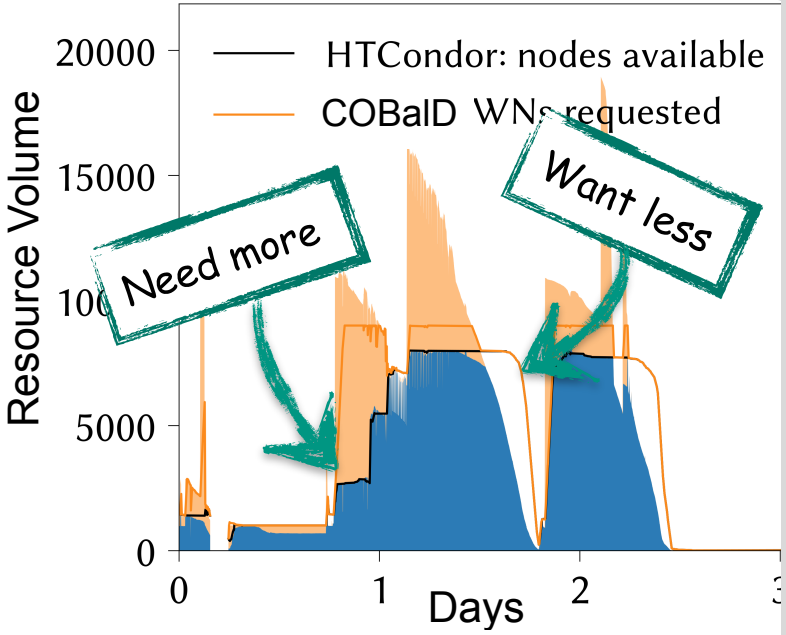
- Look at what is used, not what is requested
  - Simple logic: more used, less unused resources
  - Only batch system scheduler handles jobs
  - COBalD only acquires/releases resources
- Generic design for any resources
  - COBalD just knows (un-)used
  - CPU, CPU+RAM, CPU+DISK, CPU+DISK+RAM, …
  -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



Simpler Approach: COBalD (2018-present)

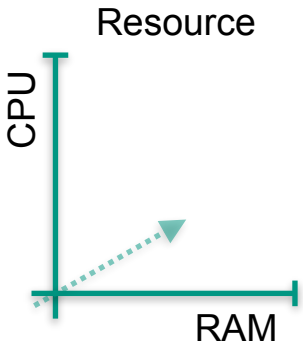
[**C**OBalD - the **O**pportunistic **Bal**ancing **D**aemon]

- 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
- HEP integration via COBalD/TARDIS
  - Define VM/Container/Job as resource
  - See **Talk by M. Giffels at 14:30**



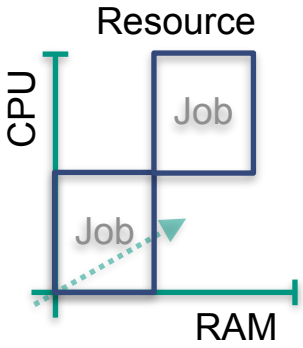
The COBalD Pool Model

- COBalD tracks basic resource features
  - Just allocation, fitness, supply
  - Defined by resource implementation
  - Usually obvious and configurable



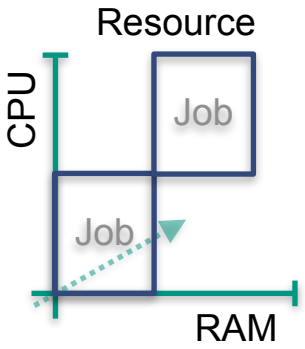
The COBalD Pool Model

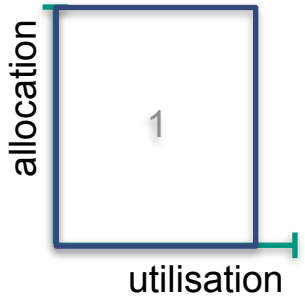
- COBalD tracks basic resource features
  - Just allocation, fitness, supply
  - Defined by resource implementation
  - Usually obvious and configurable



The COBalD Pool Model

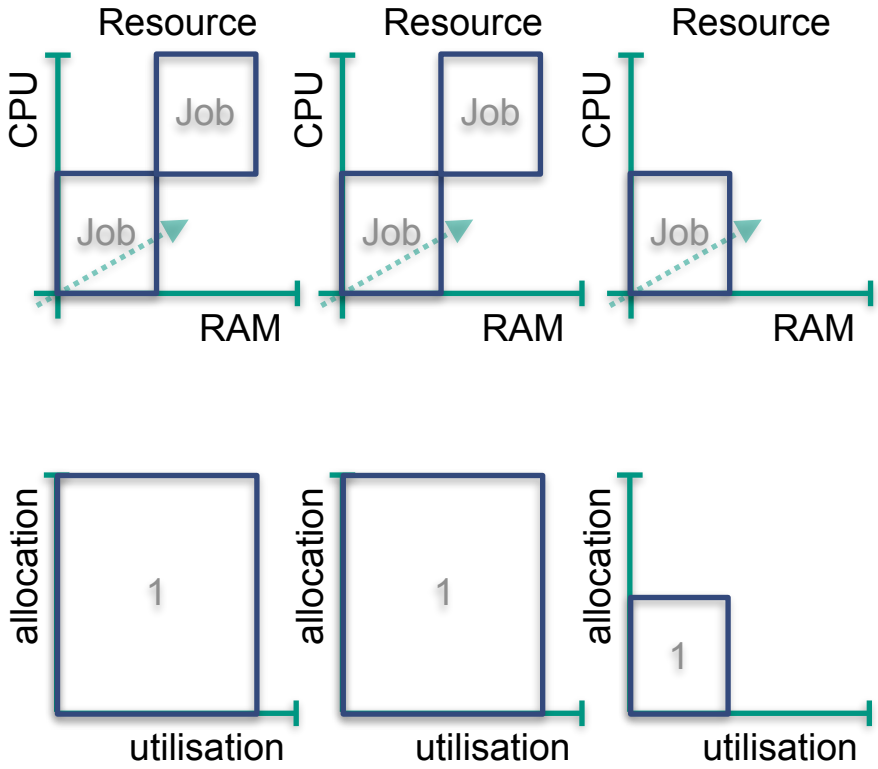
- COBalD tracks basic resource features
  - Just allocation, fitness, supply
  - Defined by resource implementation
  - Usually obvious and configurable





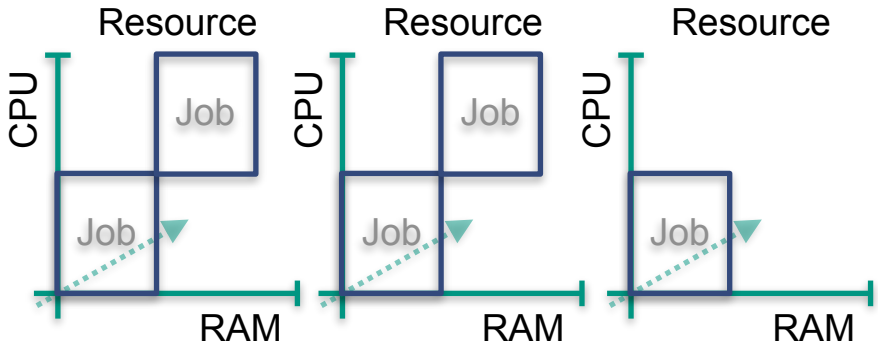
The COBalD Pool Model

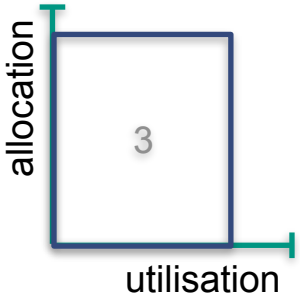
- COBalD tracks basic resource features
  - Just allocation, fitness, supply
  - Defined by resource implementation
  - Usually obvious and configurable



The COBalD Pool Model

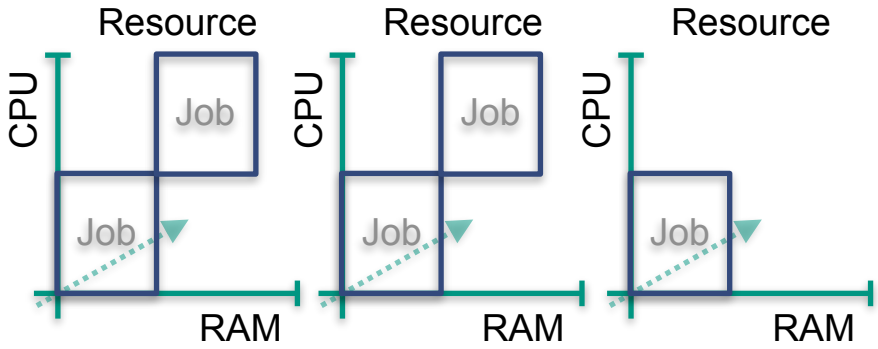
- COBalD tracks basic resource features
  - Just allocation, fitness, supply
  - Defined by resource implementation
  - Usually obvious and configurable
- Aggregate resources to uniform Pools
  - Pools are just another resource
  - Simple sum/avg transformation
  - Elements become indistinguishable

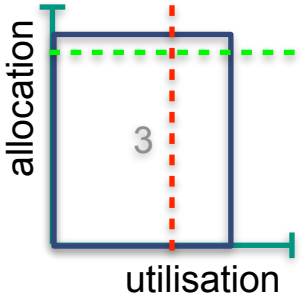




The COBalD Pool Model

- COBalD tracks basic resource features
  - Just allocation, fitness, supply
  - Defined by resource implementation
  - Usually obvious and configurable
- Aggregate resources to uniform Pools
  - Pools are just another resource
  - Simple sum/avg transformation
  - Elements become indistinguishable
- Manageable via simple rules
  - Positive and negative assertion
  - Thresholds and Ratios
  - “Ensure X free above Y% fitness”

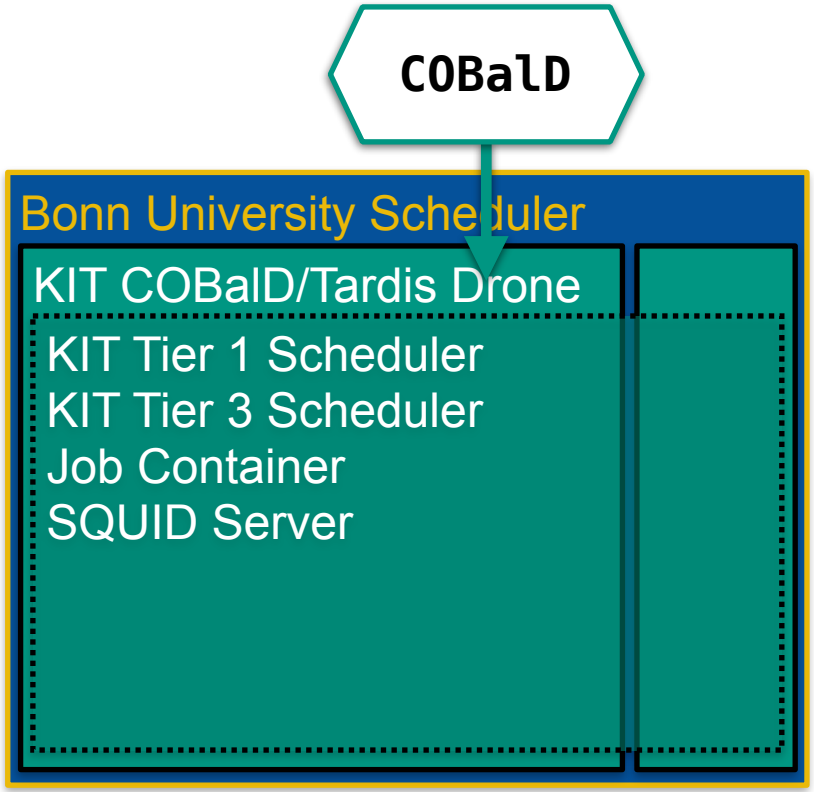




Decoupling Job and Meta-Scheduler

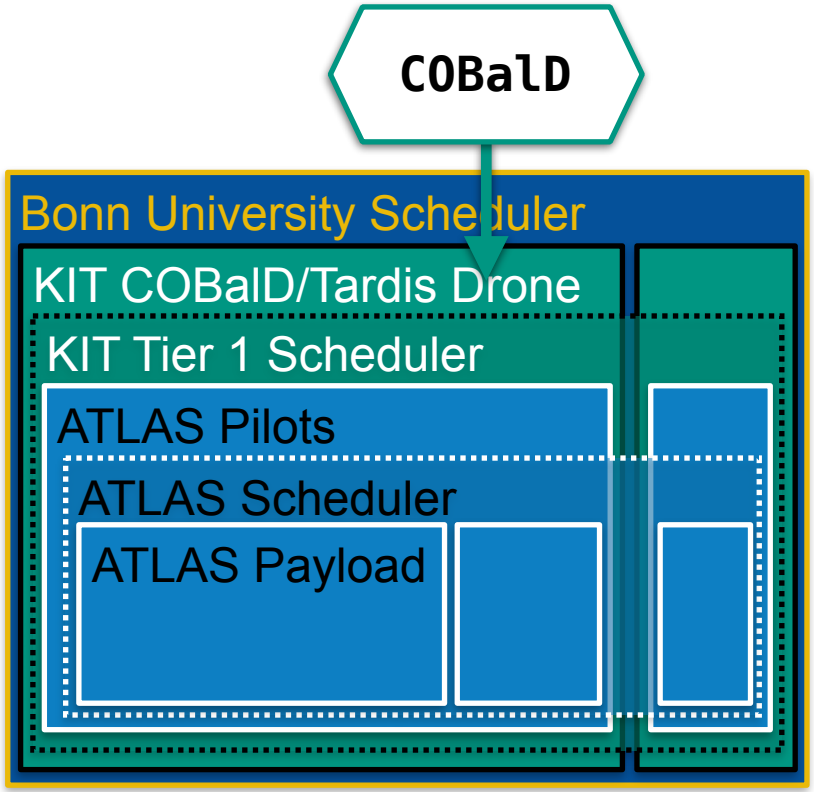
Decoupling Job and Meta-Scheduler

- No prediction of nested scheduler decisions
  - Payload-Scheduler operates independently
  - COBalD only observes decisions
  - Simplifies policies and configuration



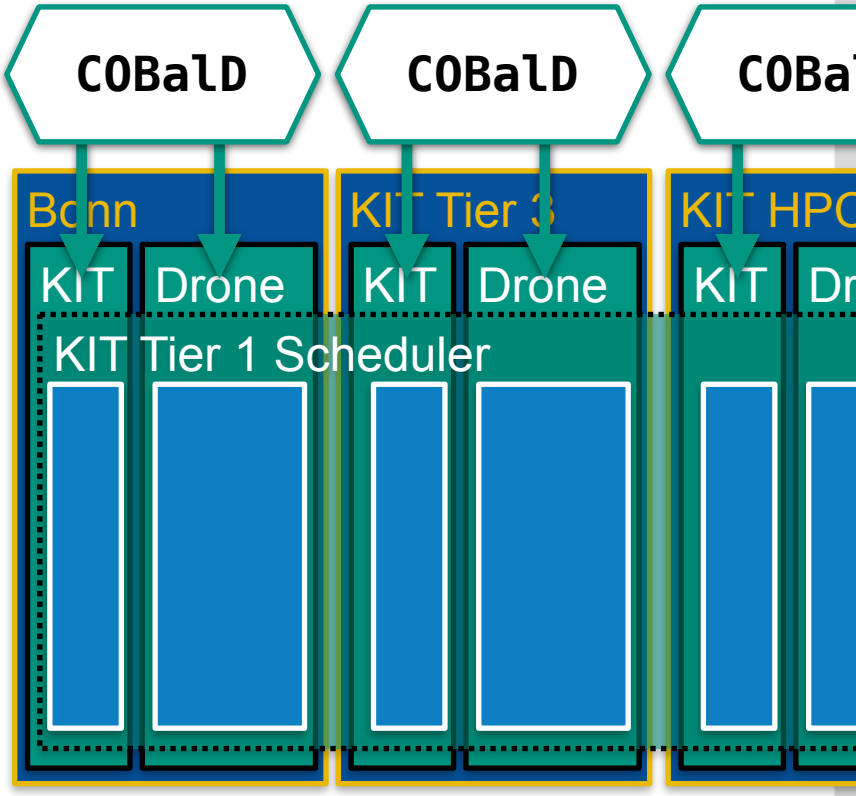
Decoupling Job and Meta-Scheduler

- No prediction of nested scheduler decisions
  - Payload-Scheduler operates independently
  - COBalD only observes decisions
  - Simplifies policies and configuration
- Support arbitrarily nested schedulers
  - Resources for other meta-scheduler
  - “Supports” nested preemption, fairshare, …
  - COBalD tends to conservative latency



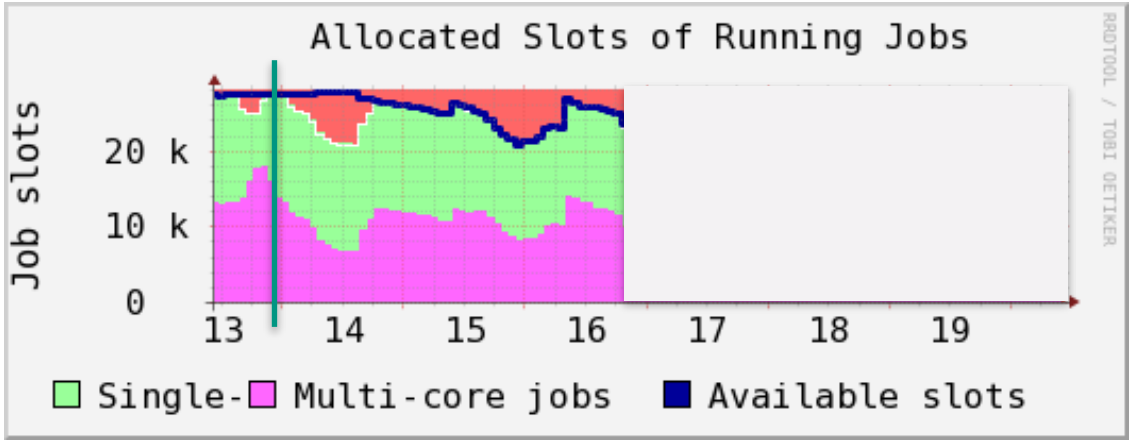
Decoupling Job and Meta-Scheduler

- No prediction of nested scheduler decisions
  - Payload-Scheduler operates independently
  - COBalD only observes decisions
  - Simplifies policies and configuration
- Support arbitrarily nested schedulers
  - Resources for other meta-scheduler
  - “Supports” nested preemption, fairshare, …
  - COBalD tends to conservative latency
- Multiple instances can service same overlay
  - Each daemon acts independently
  - Overlay handles heterogeneous resources
  - Survival of the fittest resources



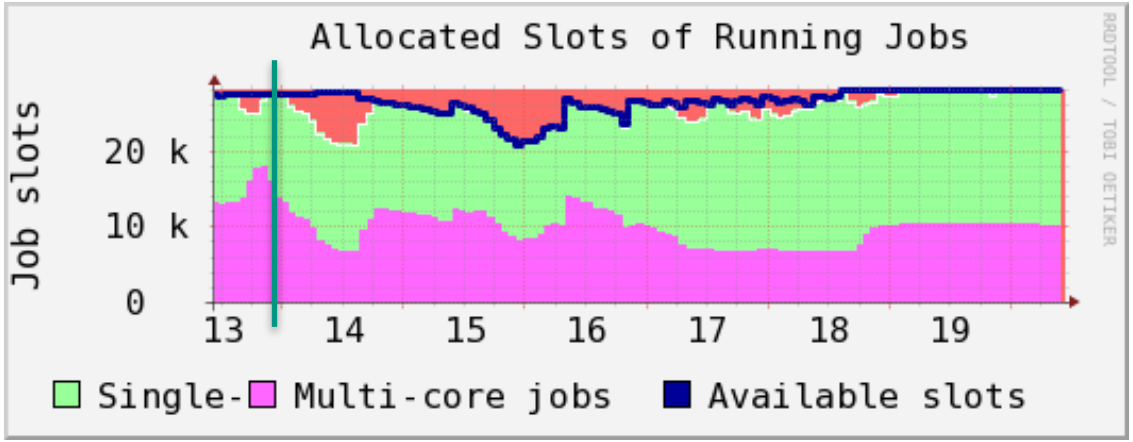
Flexibility Study: Concurrency Limits

- Use HTCondor ConcurrencyLimits to limit active Single Core jobs
  - Limits Multi Core slot fragmentation by Single Core jobs
  - Effectively creates a protected partition in Batch System



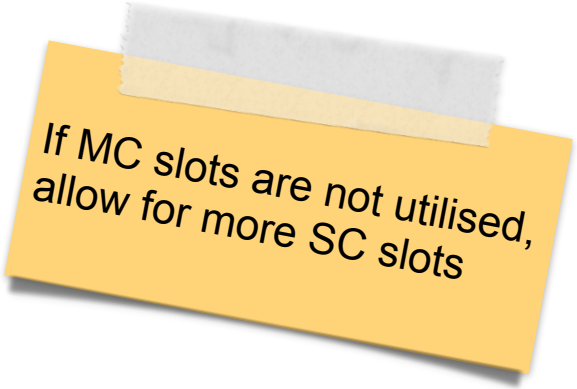
Flexibility Study: Concurrency Limits

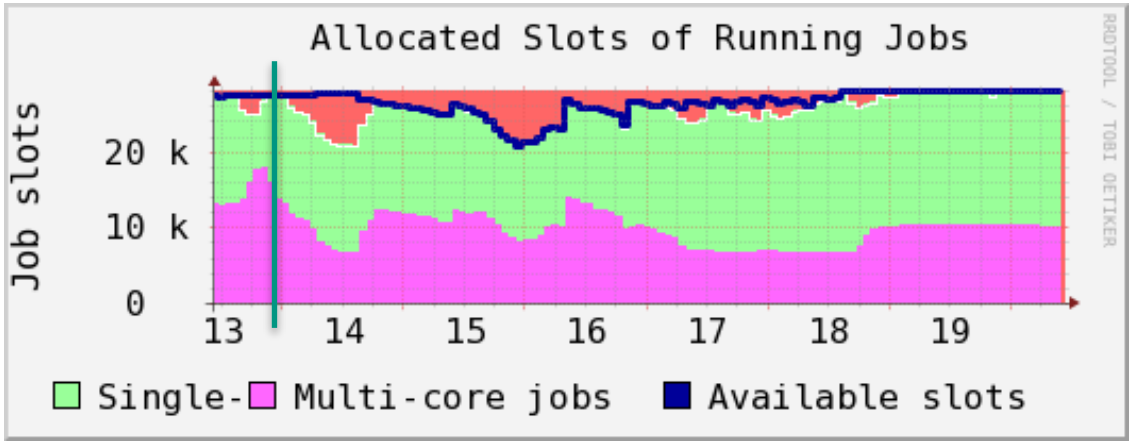
- Use HTCondor ConcurrencyLimits to limit active Single Core jobs
  - Limits Multi Core slot fragmentation by Single Core jobs
  - Effectively creates a protected partition in Batch System
- COBalD manages protected partition as Opportunistic Resource Pool
  - Only access ConcurrencyLimits, no underlying Pilot/VM/…
  - Pool represents absence of Single Core slots



Flexibility Study: Concurrency Limits

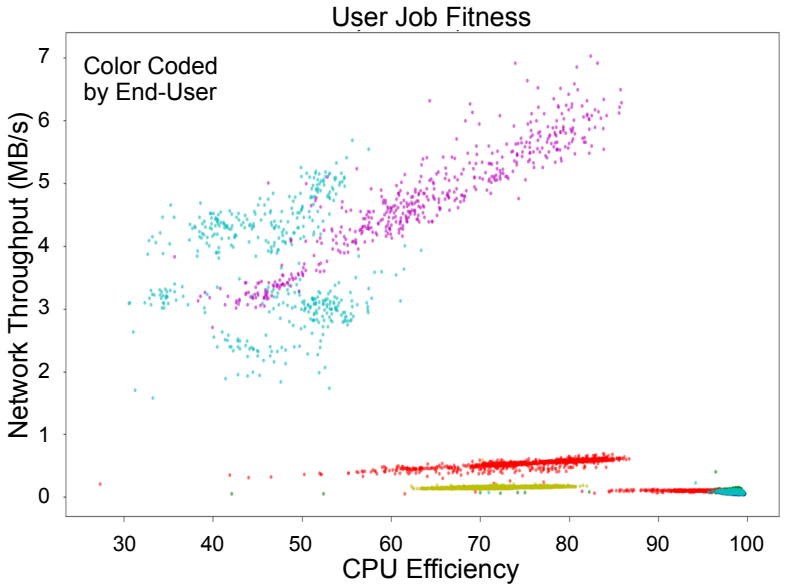
- Use HTCondor ConcurrencyLimits to limit active Single Core jobs
  - Limits Multi Core slot fragmentation by Single Core jobs
  - Effectively creates a protected partition in Batch System
- COBalD manages protected partition as Opportunistic Resource Pool
  - Only access ConcurrencyLimits, no underlying Pilot/VM/…
  - Pool represents absence of Single Core slots





Towards Implicit Network Scheduling

- Respect network availability and congestion for provisioning
  - Congested network is the bottleneck for opportunistic resources
  - Non-linear interference and noticeable measurement overhead

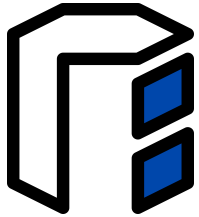


Towards Implicit Network Scheduling

- Respect network availability and congestion for provisioning
  - Congested network is the bottleneck for opportunistic resources
  - Non-linear interference and noticeable measurement overhead
- Research: Implicitly schedule network capacity via side-effects
  - Cheap CPU efficiency query as boundary for network efficiency (and other resources)
    - CPU efficiency implies general fitness
  - Safeguard to push the maximum possible data analysis jobs to opportunistic resources

Conclusions

- Meta-scheduling via prediction is difficult for complex cases
  - Implies duplicated responsibilities and high coupling
  - Prediction unfeasible with pilots + opportunistic resources
- COBalD: Lightweight feedback based meta-scheduler
  - Reacts to observed resource allocation and fitness
  - Generic resource model covering many use-cases
- Exploit simple resource model to face HEP challenges
  - Manage diverse resource types with separate agents
  - Limit network congestion on opportunistic resources
  - COBALD/TARDIS: Talk by M. Giffels at 14:30

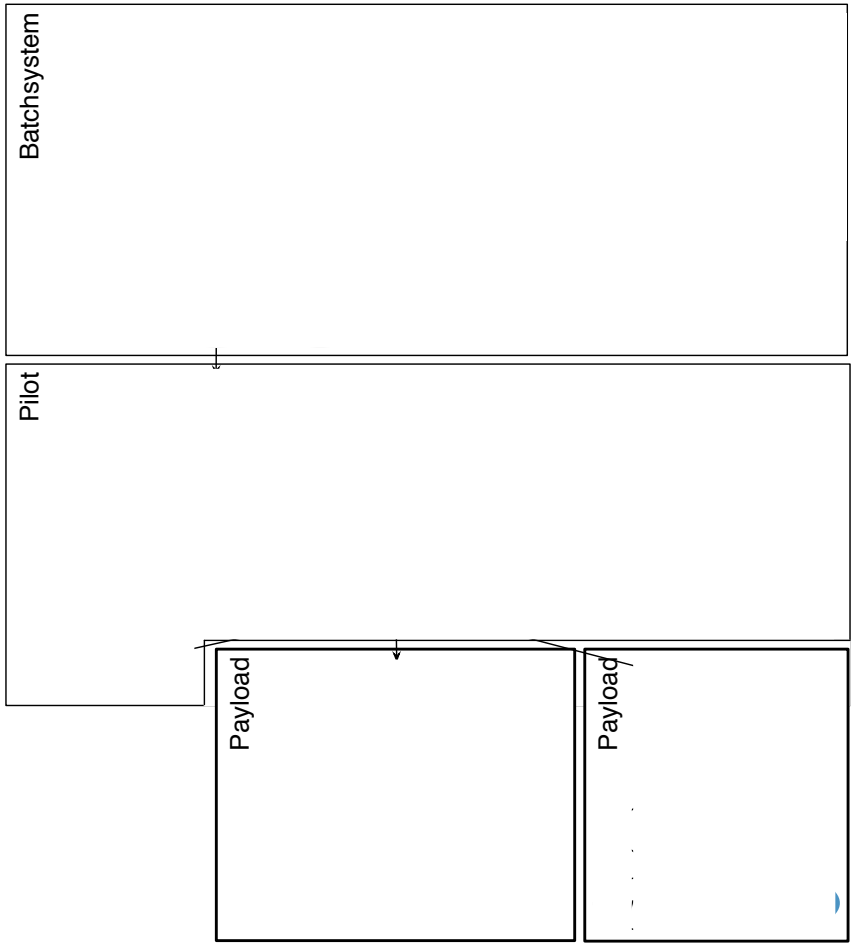


COBalD

Pilots: Classification

[Finalised/PhD E. Kühn]

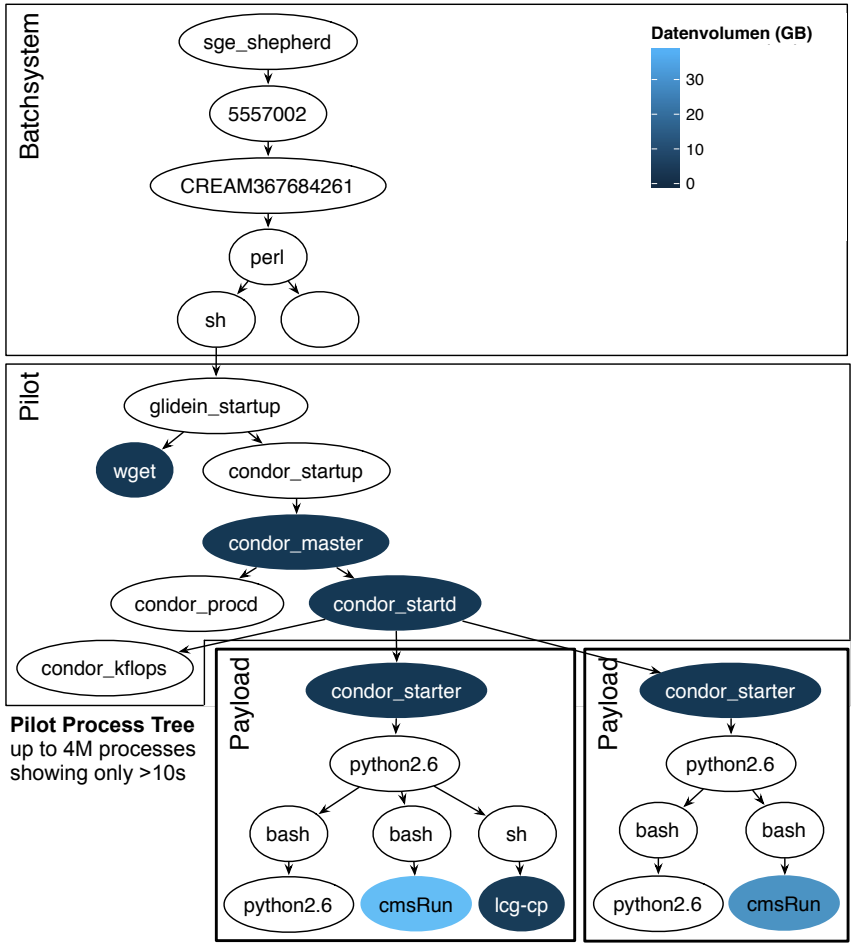
- Schedule Pilots not Jobs
  - Only know estimate for wrapping Pilot
  - Jobs are late-bound as Payloads



Pilots: Classification

[Finalised/PhD E. Kühn]

- Schedule Pilots not Jobs
  - Only know estimate for wrapping Pilot
  - Jobs are late-bound as Payloads
- Runtime Clustering
  - Cluster processes as dynamic tree stream
  - Exceeds knowledge of owning VOs

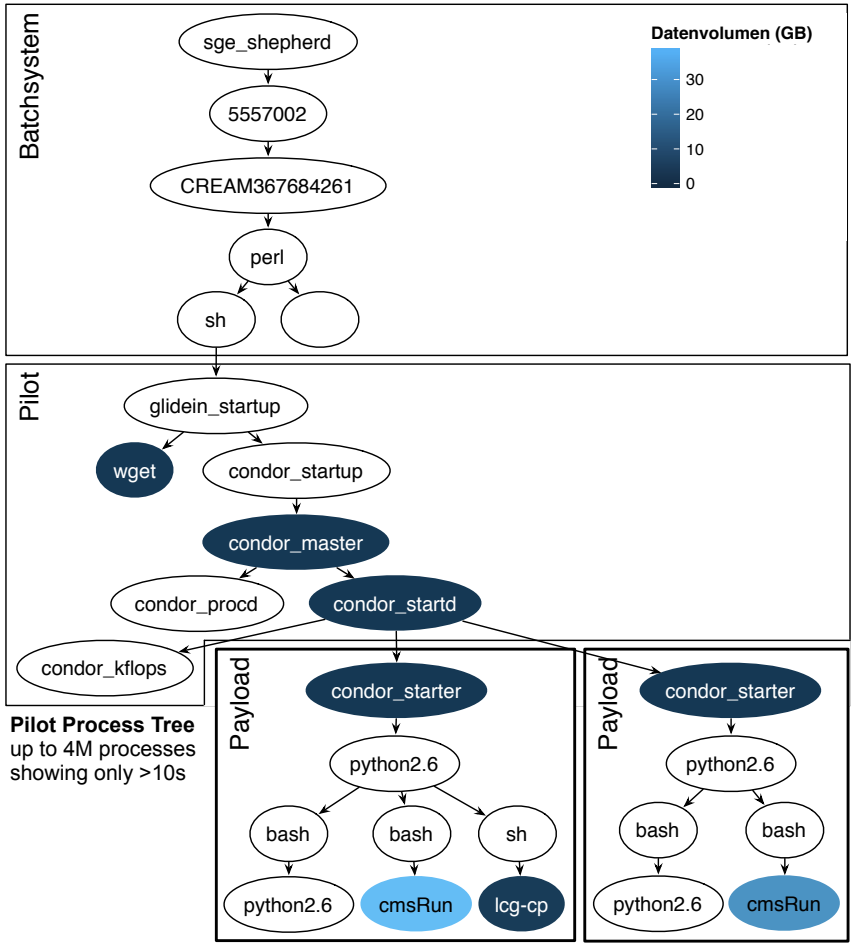


Pilot Process Tree
up to 4M processes
showing only >10s

Pilots: Classification

[Finalised/PhD E. Kühn]

- Schedule Pilots not Jobs
  - Only know estimate for wrapping Pilot
  - Jobs are late-bound as Payloads
- Runtime Clustering
  - Cluster processes as dynamic tree stream
  - Exceeds knowledge of owning VOs
- Practical information limit
  - Apply observations to successive pilots
  - Only able to observe current, partial state

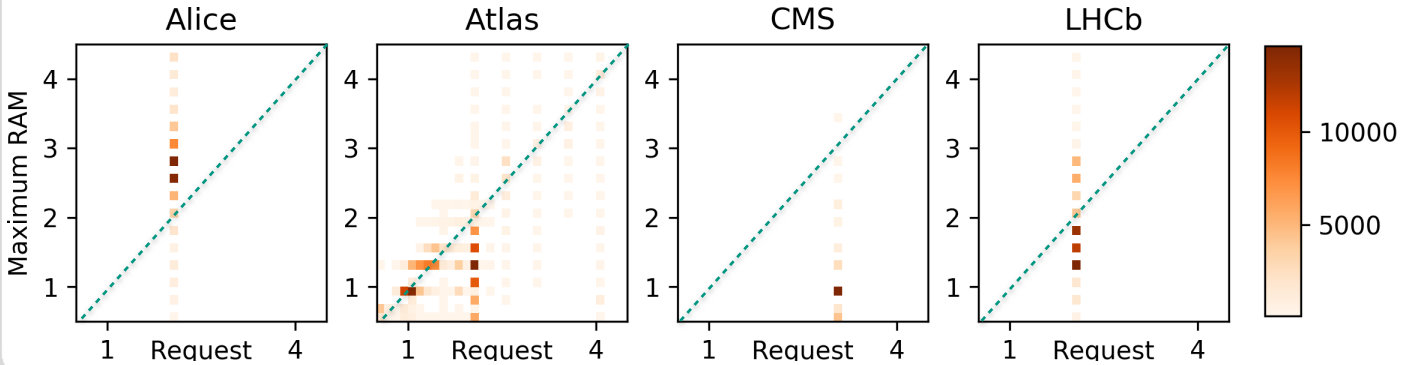


Pilot Process Tree
up to 4M processes
showing only >10s

Pilots: Requirement Estimations

[Proof of Concept/MSc S. Lange]

- Attempt to predict precise pilot requirements for scheduling
  - Transfer monitored features from finished to queued pilots
  - Investigate possible gains from heuristic approaches
- Viable to use statistics and machine learning for short-term prediction
  - Improve average resource request accuracy from 75% to 90%
  - Only reliable up to 10 min before pilot is scheduled!



ROCED -> COBalD

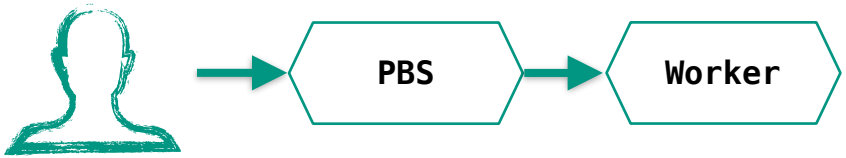
- ROCED [2010-present]
  - Central management service
  - Monolithic decision logic
  - Tightly coupled to HTCondor
- COBalD [2018-present]
  - Distributed micro services
  - Cooperating individual agents
  - Loosely coupled to HTCondor

Site/Provider	Capacity
~~1 und 1~~	~~400 CPU~~
~~OpenTelekomCloud~~	~~580 CPU~~
GridKaSchool Cloud	600 CPU
NEMO (Freiburg)	8000 CPU
ForHLR2	60 CPU

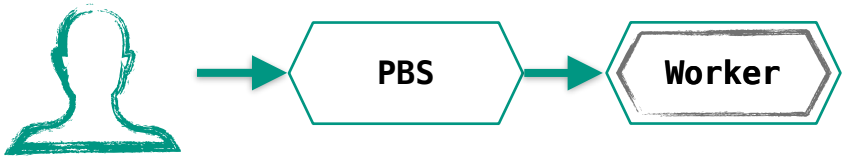
Site/Provider	Capacity
~~Exoscale~~	~~300 CPU~~
~~OpenTelekomCloud~~	~~580 CPU~~
GridKaSchool Cloud	800 CPU
NEMO (Freiburg)	8000 CPU
ForHLR2	60 CPU
GridKa Multicore	30000 CPU pseudo-anti-share-stuff

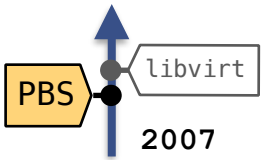
We don't even have a word for these thingies

A Tale of Opportunistic Resources (2007-present)



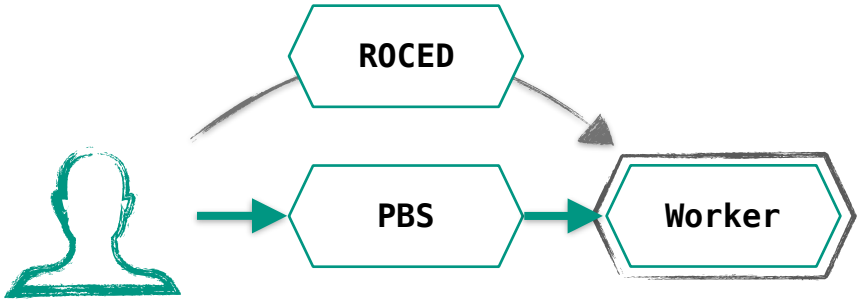
A Tale of Opportunistic Resources (2007-present)

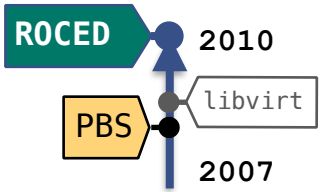




A Tale of Opportunistic Resources (2007-present)

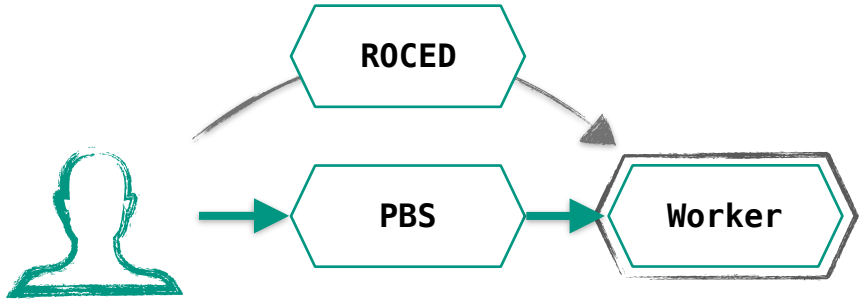
ROCED

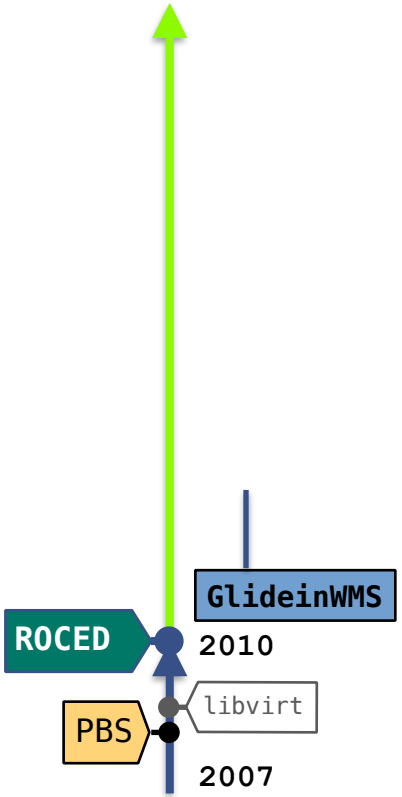




A Tale of Opportunistic Resources (2007-present)

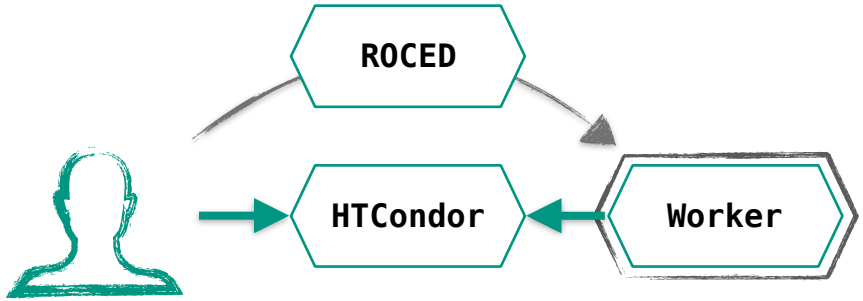
ROCED

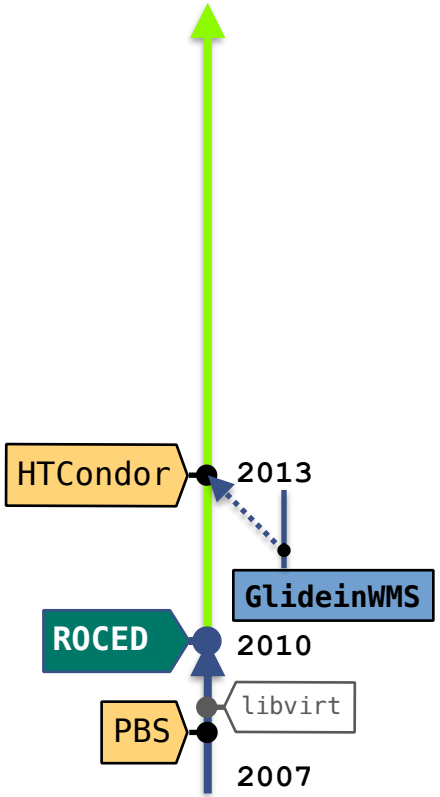




A Tale of Opportunistic Resources (2007-present)

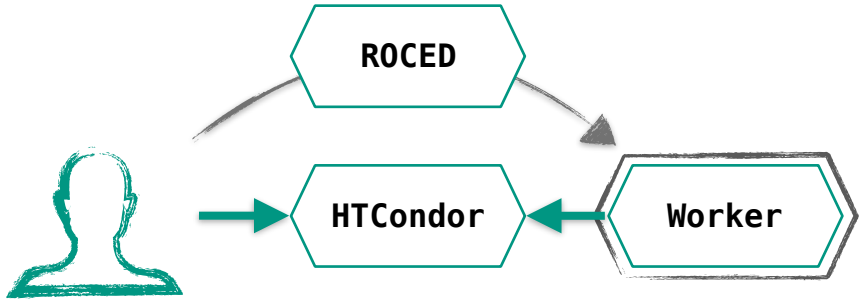
ROCED

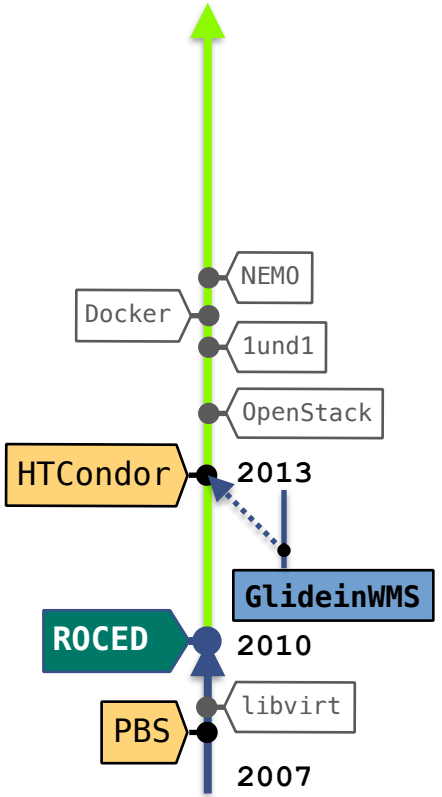




A Tale of Opportunistic Resources (2007-present)

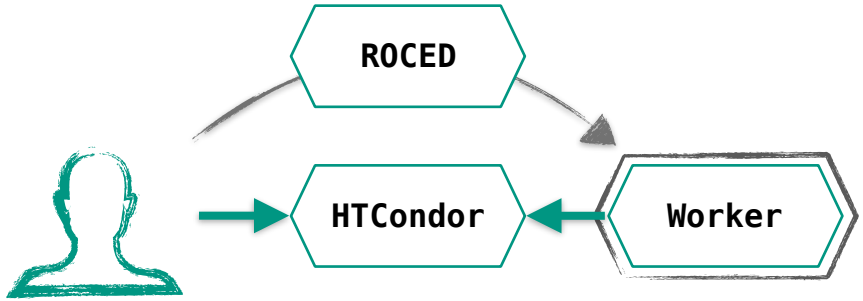
ROCED

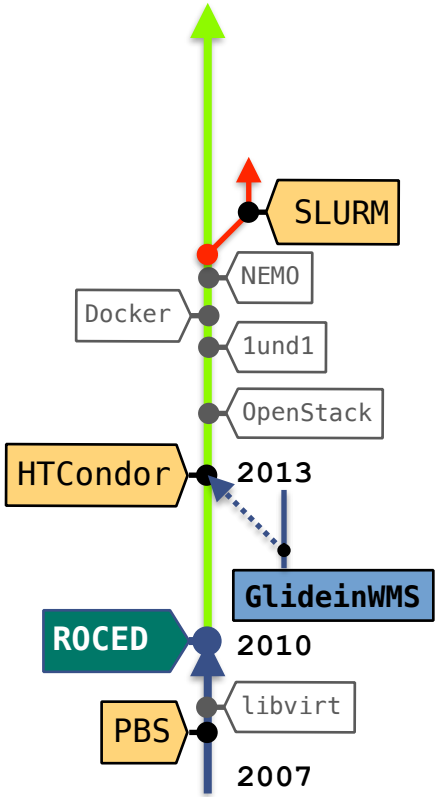




A Tale of Opportunistic Resources (2007-present)

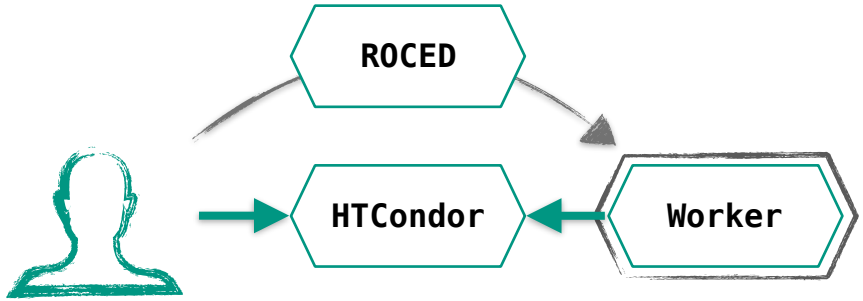
ROCED

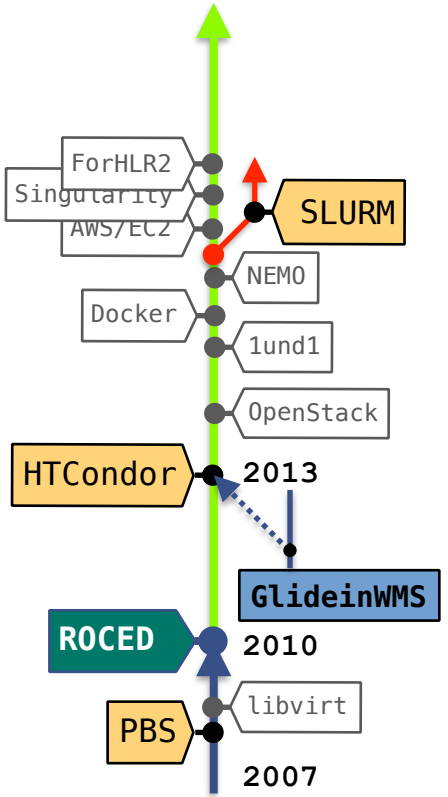


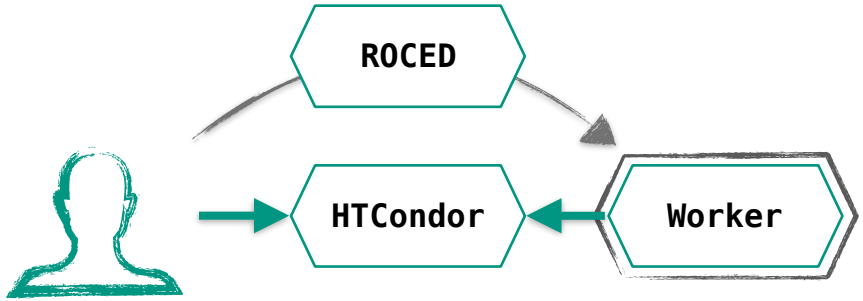
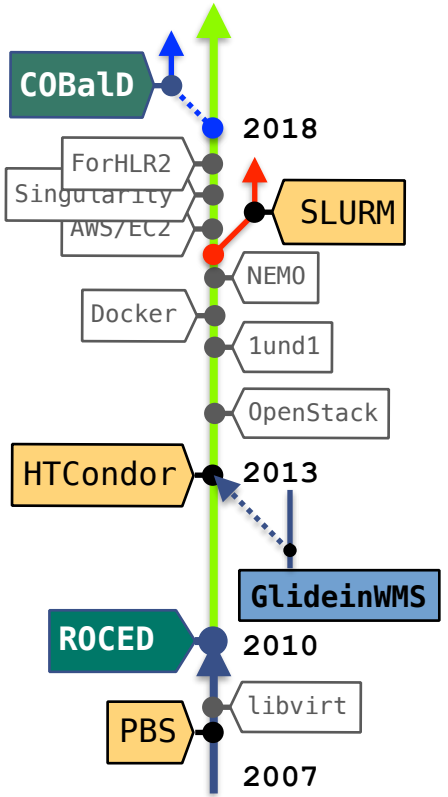


A Tale of Opportunistic Resources (2007-present)

ROCED





A Tale of Opportunistic Resources (2007-present)

ROCED