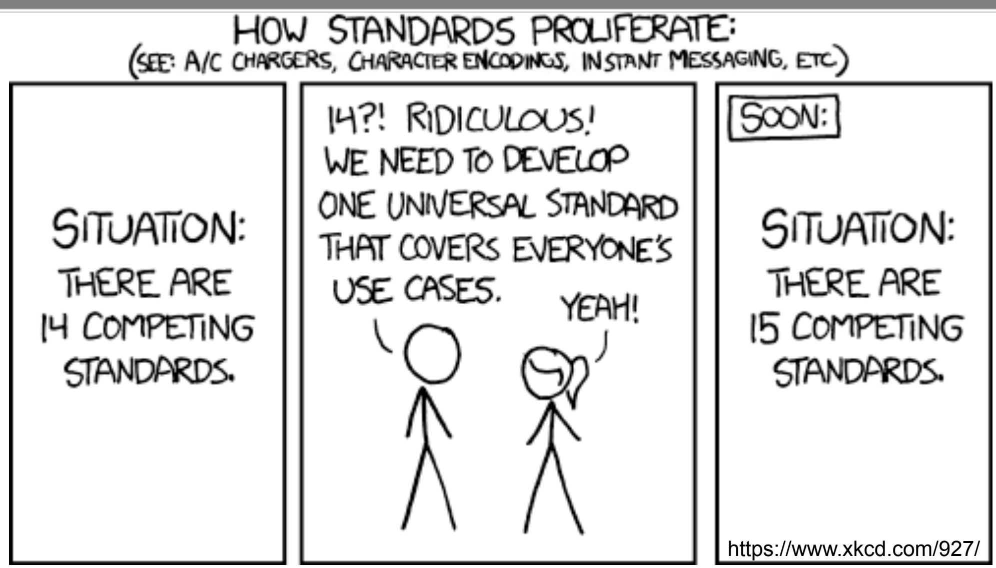
COBalD – Boosting Scientific Workflows with Opportunistic Resources

How I Learned to Stop Worrying and Love the Cloud

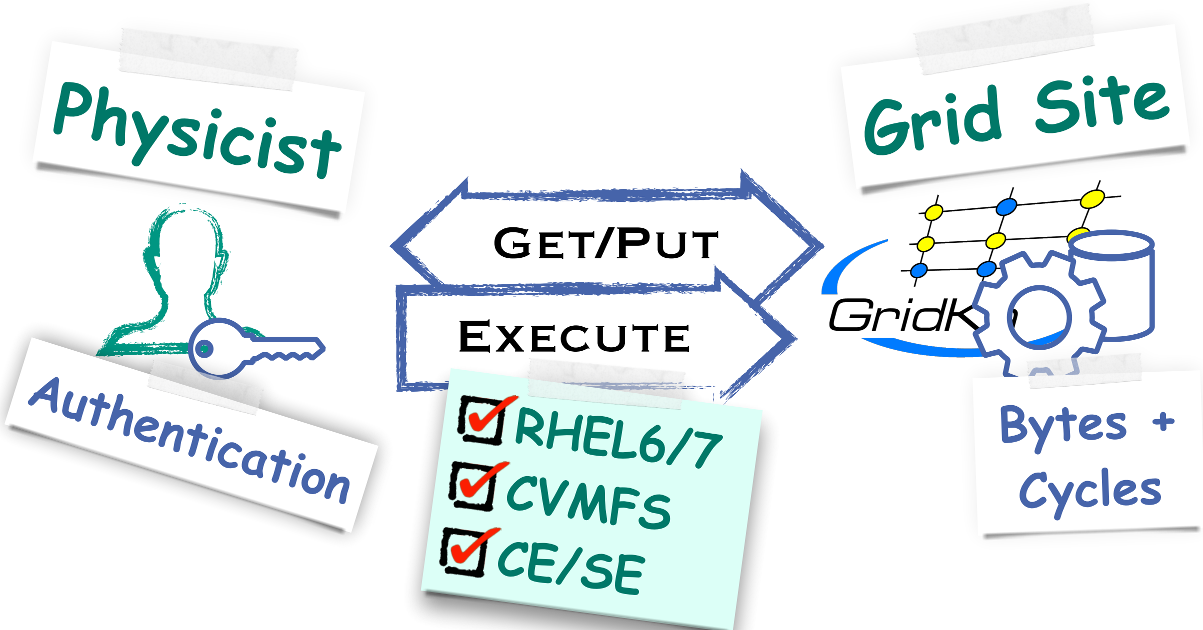
HNSciCloud at GridKa School 2018

Max Fischer, Eileen Kühn, Manuel Giffels

Steinbuch Centre for Computing / Institute for Experimental Particle Physics

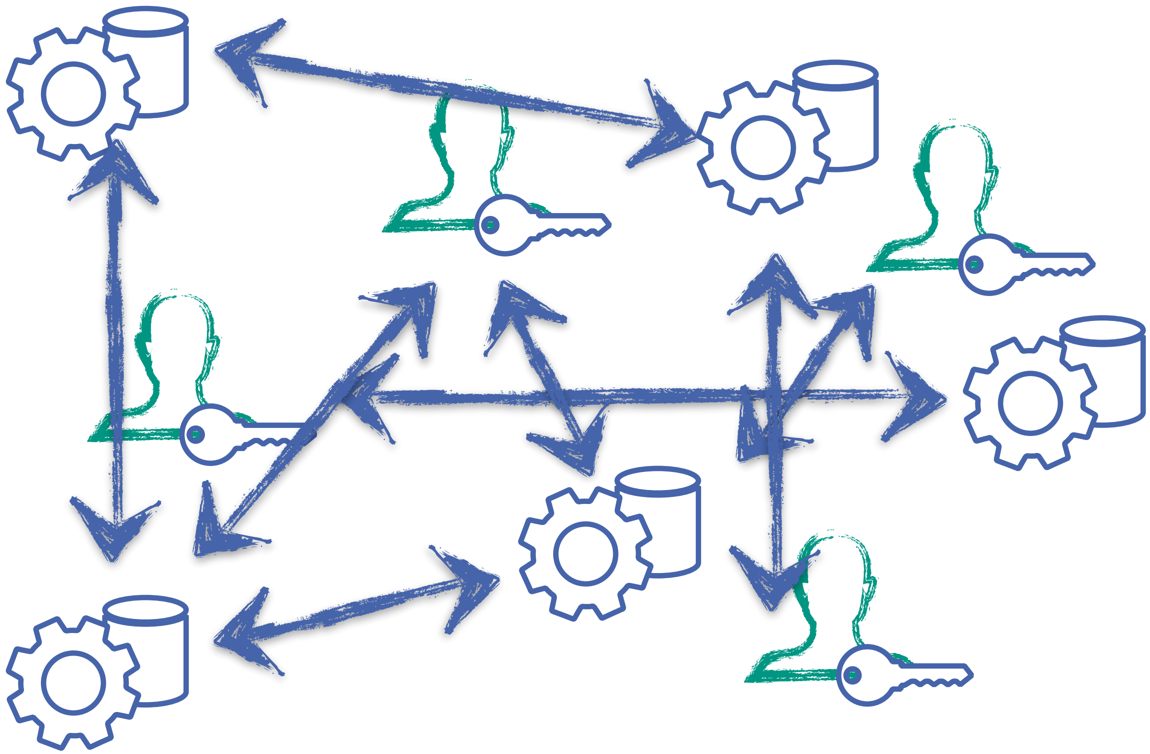


The Grid

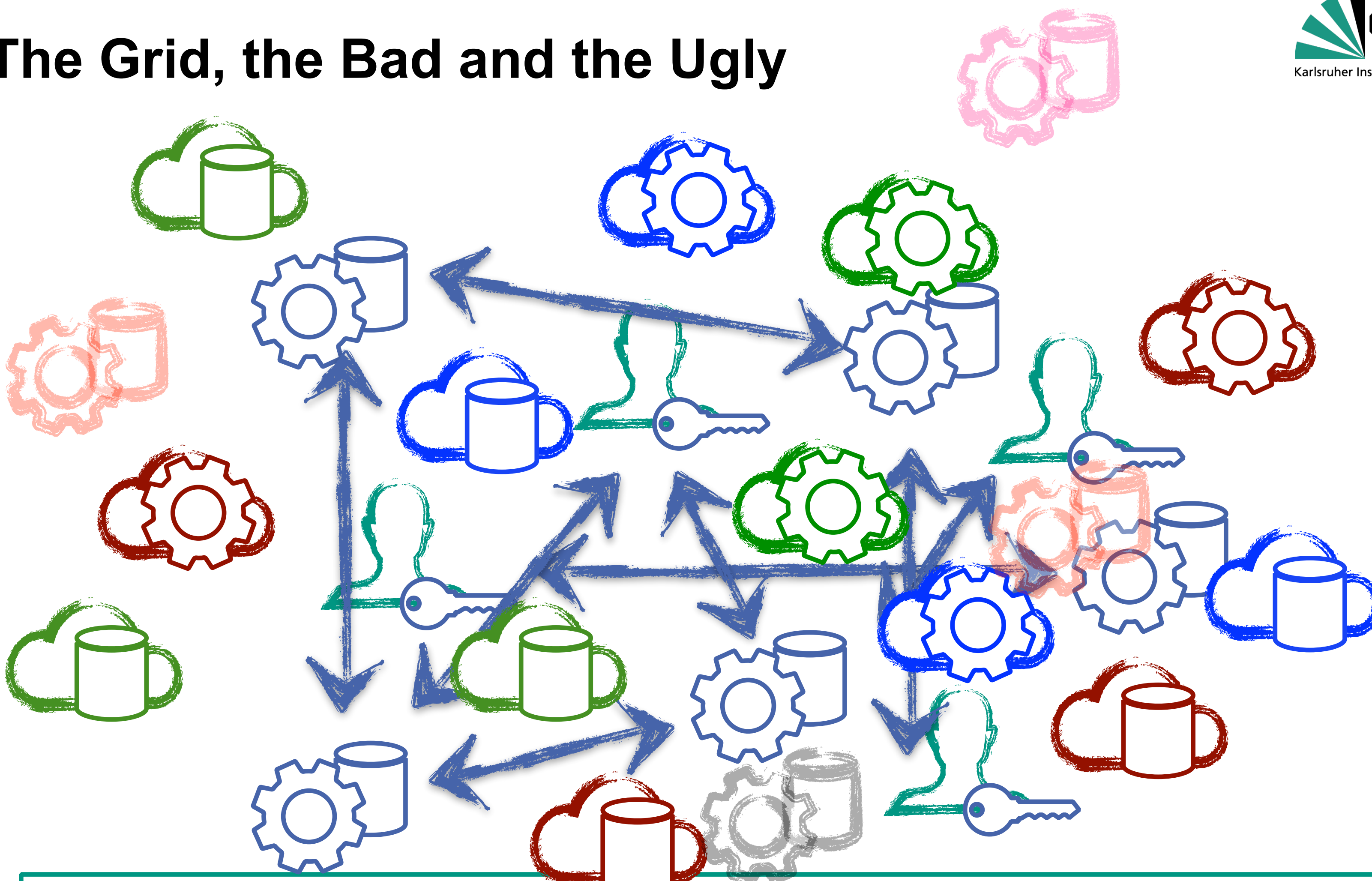


Physicist
Authentication
GET/PUT
EXECUTE
RHEL6/7
CVMFS
CE/SE
Grid Site
Bytes +
Cycles

The Grid



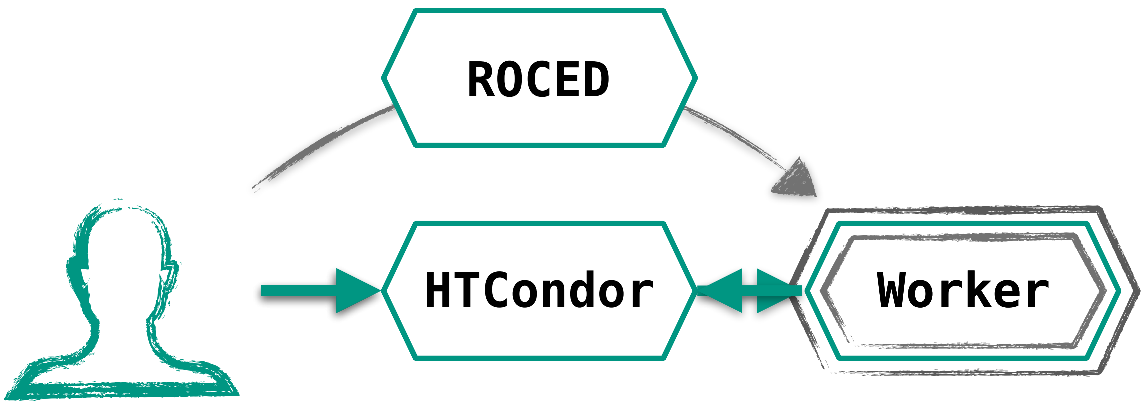
The Grid, the Bad and the Ugly

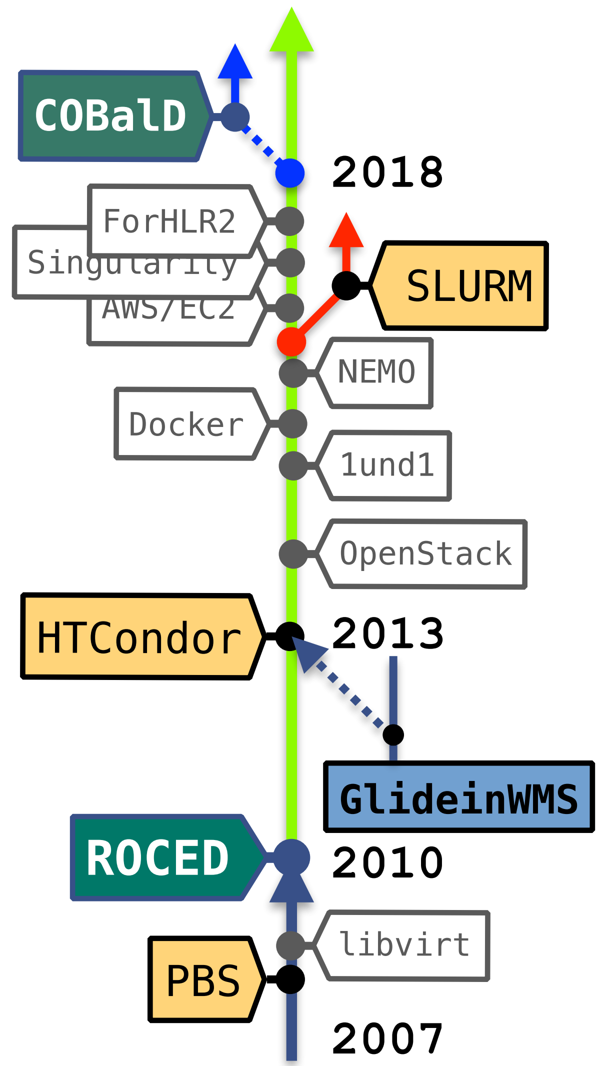


Opportunistic Resource

Any resources *not permanently dedicated to* but *temporarily available for* a specific task, user or group.

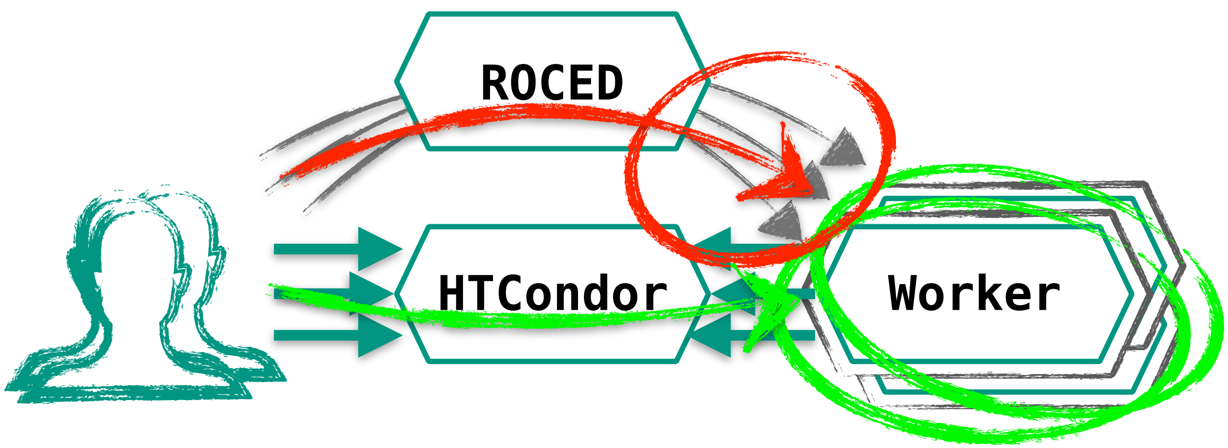
A Tale of Opportunistic Resources (2007-present)





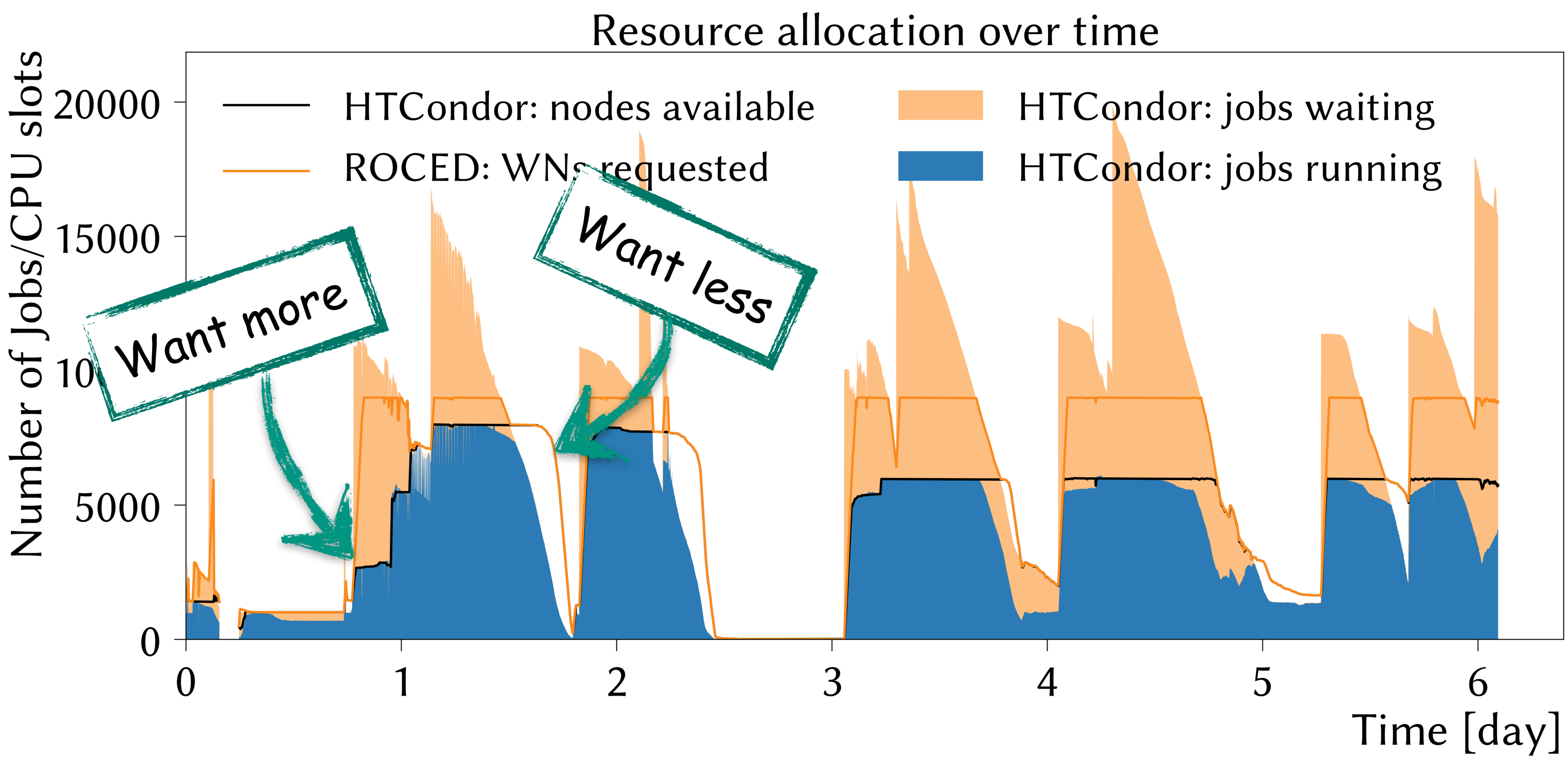
ROCED Resumé

- Dynamic resources matching user demand
  - Trivial to support new providers for many users
  - Difficult to manage several providers for many users
- Resource aggregation in overlay batch system
  - Unreliable to predict resources required for jobs
  - Efficient to integrate resources, then match jobs
- Yet it really works!

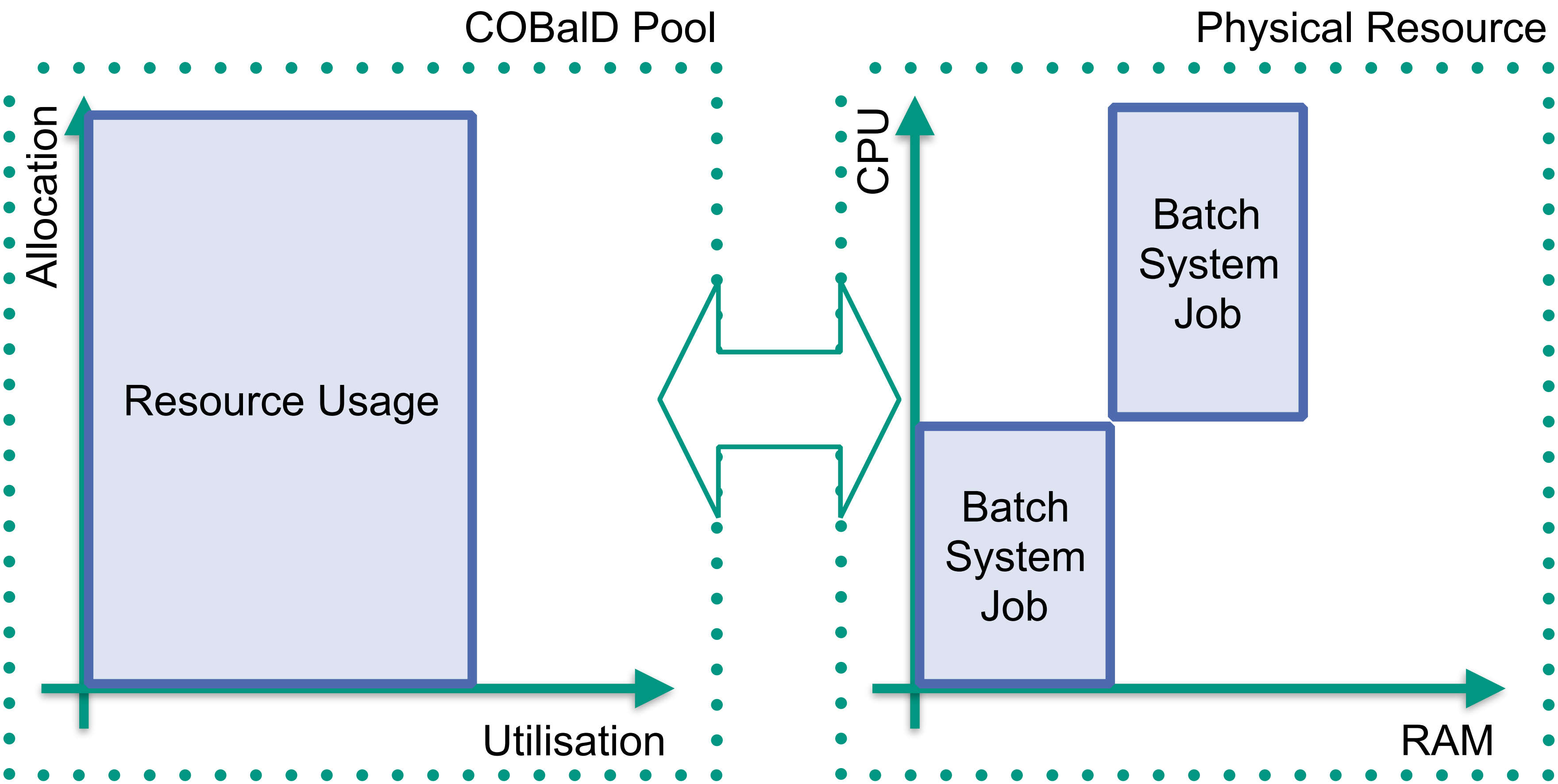


Pragmatic View to Resource Management

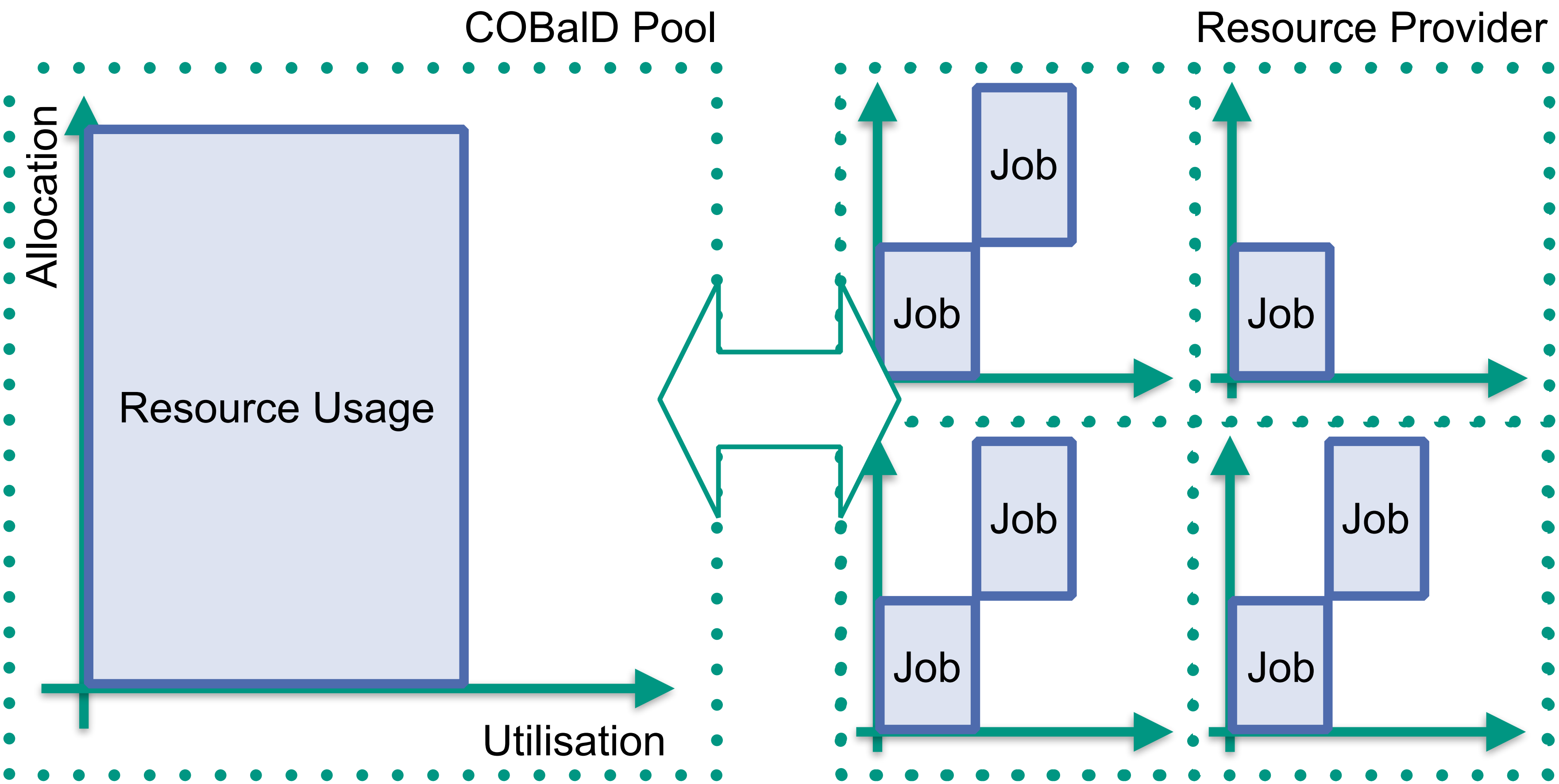
- New development for scalability and maintainability in HNSciCloud
- Simple logic: more used resources, less unused resources
  - COBalD only watches, creates and disables resources
  - Batch system scheduler selects appropriate resources



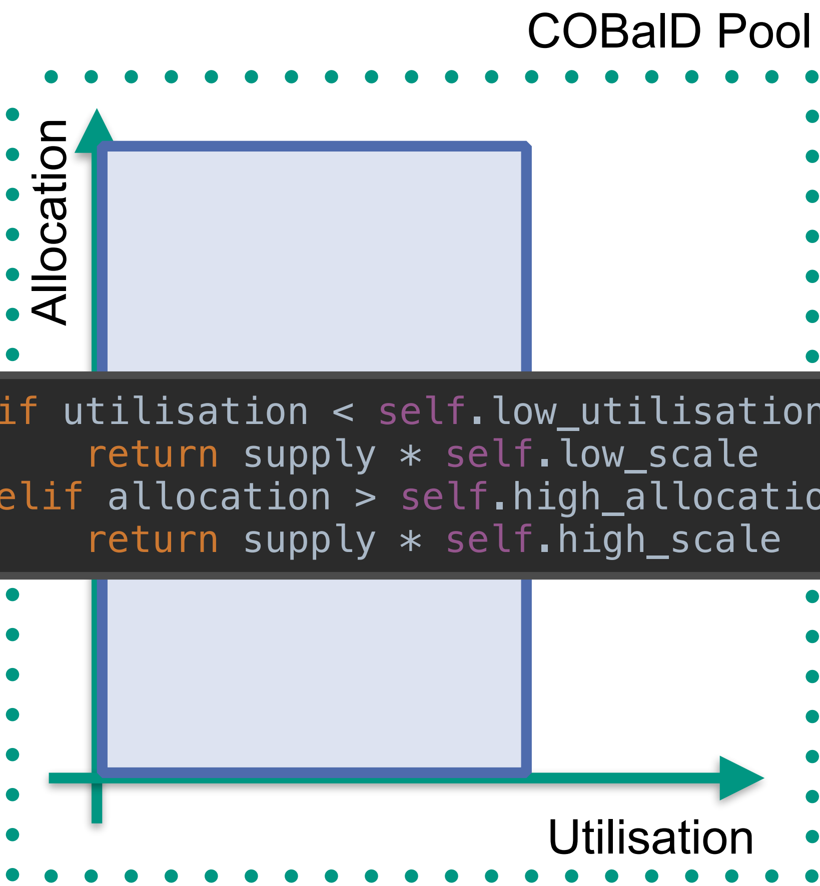
COBalD Resource Pool Model



COBalD Resource Pool Model

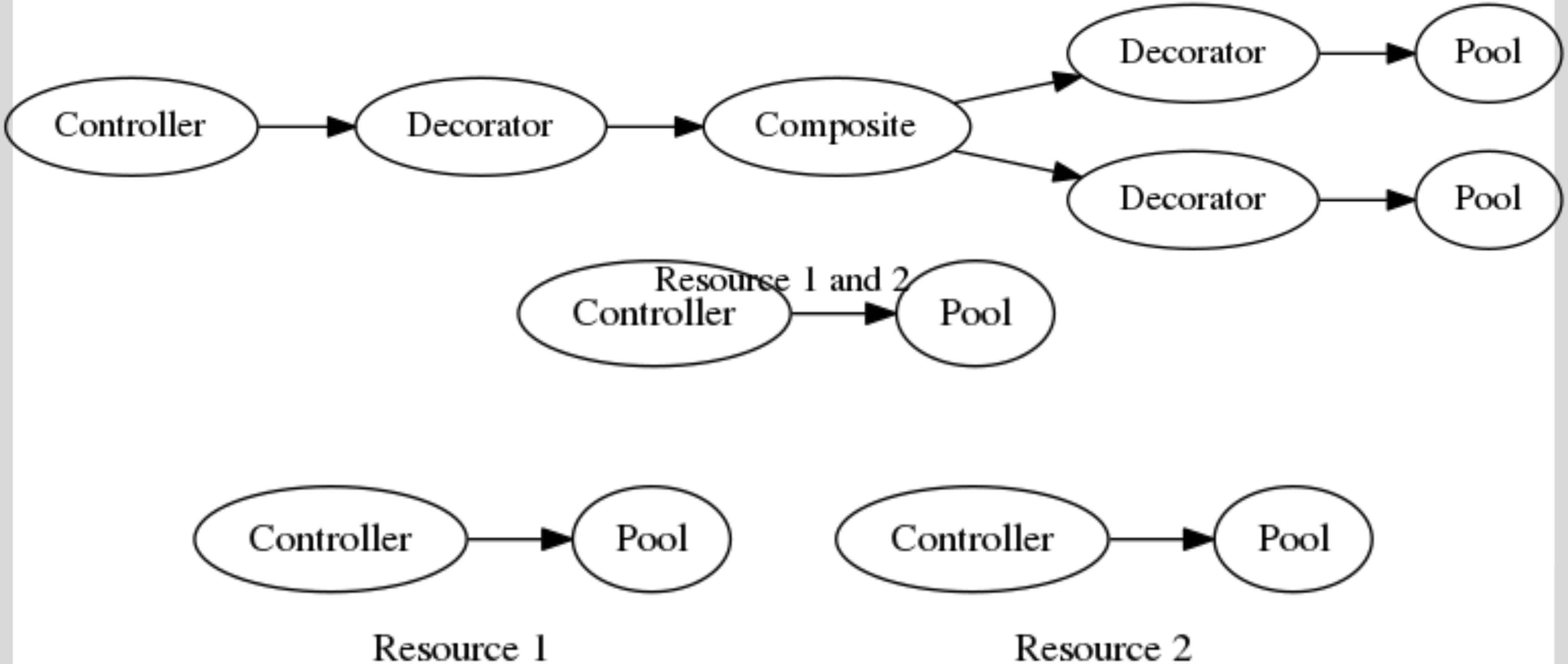


COBalD Resource Pool Model

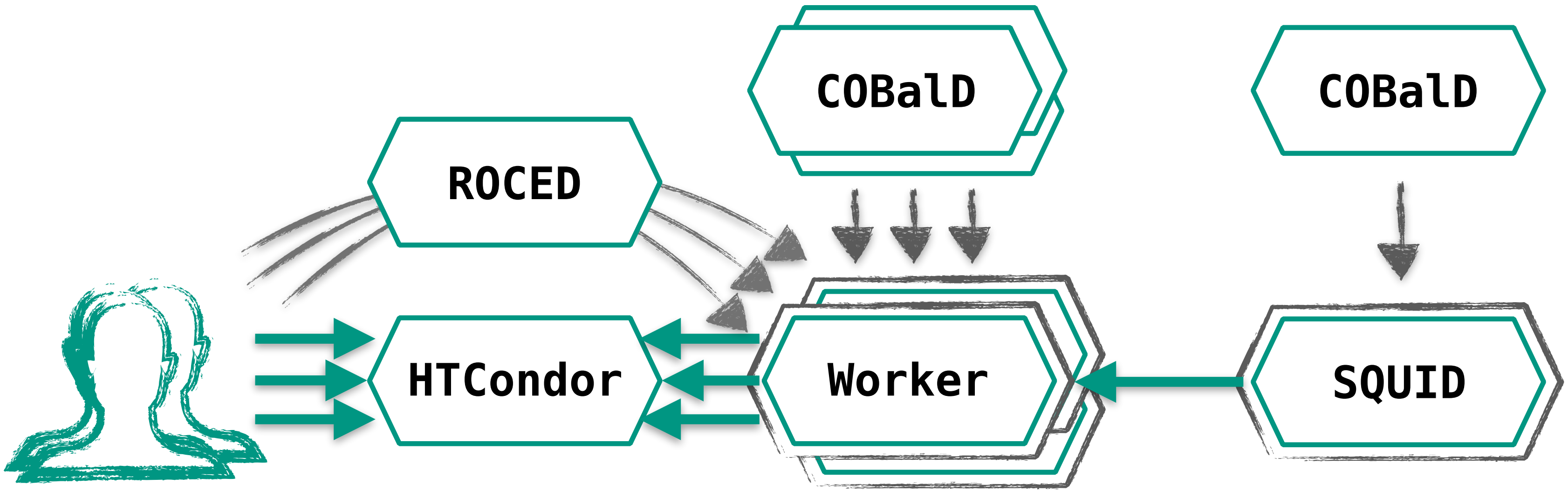



```
if utilisation < self.low_utilisation:
    return supply * self.low_scale
elif allocation > self.high_allocation:
    return supply * self.high_scale
```

COBalD Resource Pool Model

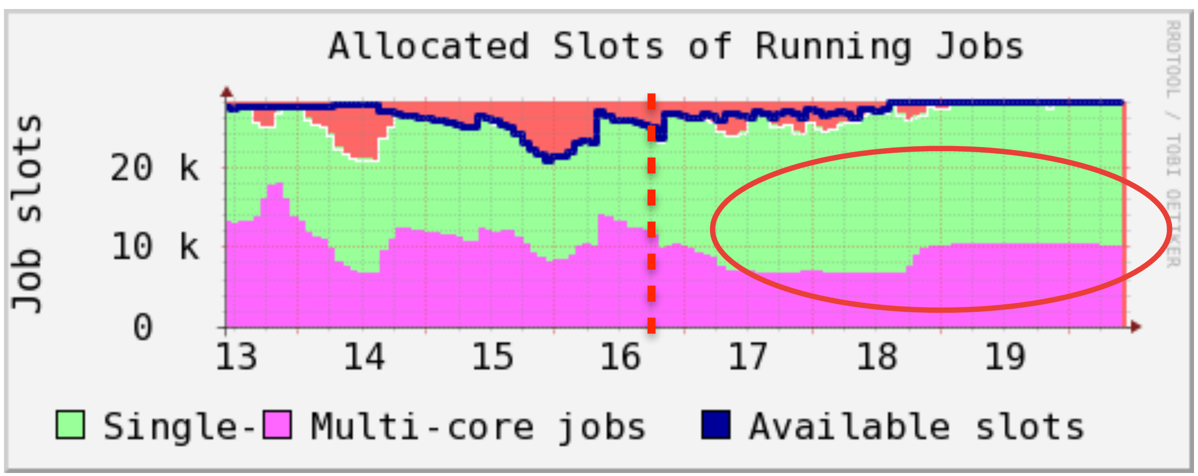


Side by Side



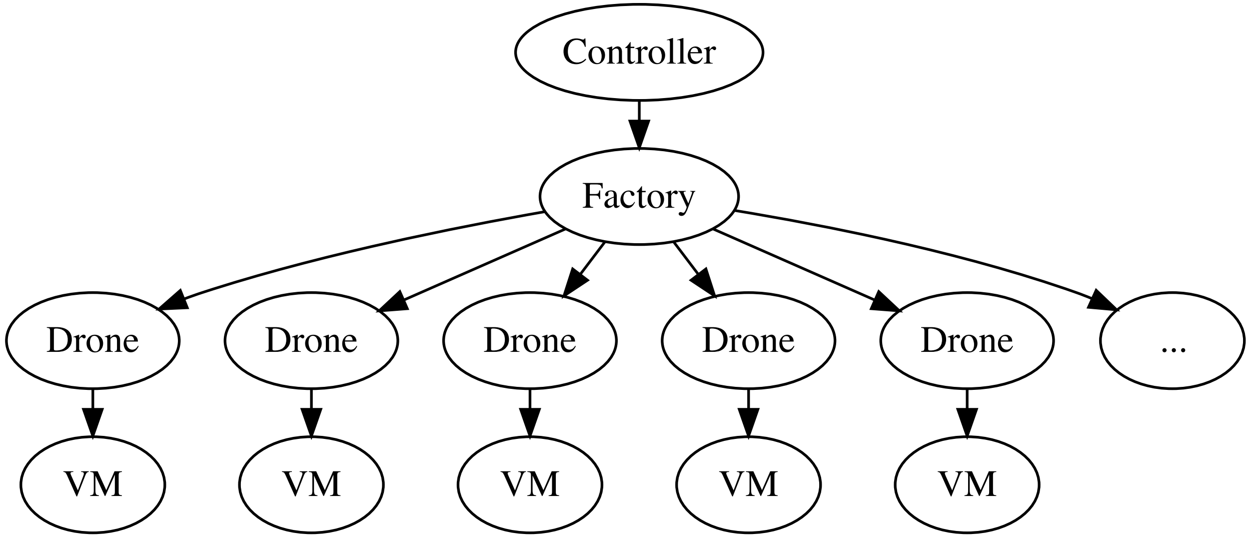
Flexibility Study: Condor ConcurrencyLimits

- GridKa limits small jobs via ConcurrencyLimits
  - Prevents fragmentation of large resource blocks
  - Effectively partitions in Batch System
- COBalD manages this as single Resource Pool
  - Only access ConcurrencyLimits, no underlying Pilot/VM/…
  - Pool represents absence of SC slots
  - "If MC slots are not utilised, allow for more SC slots“



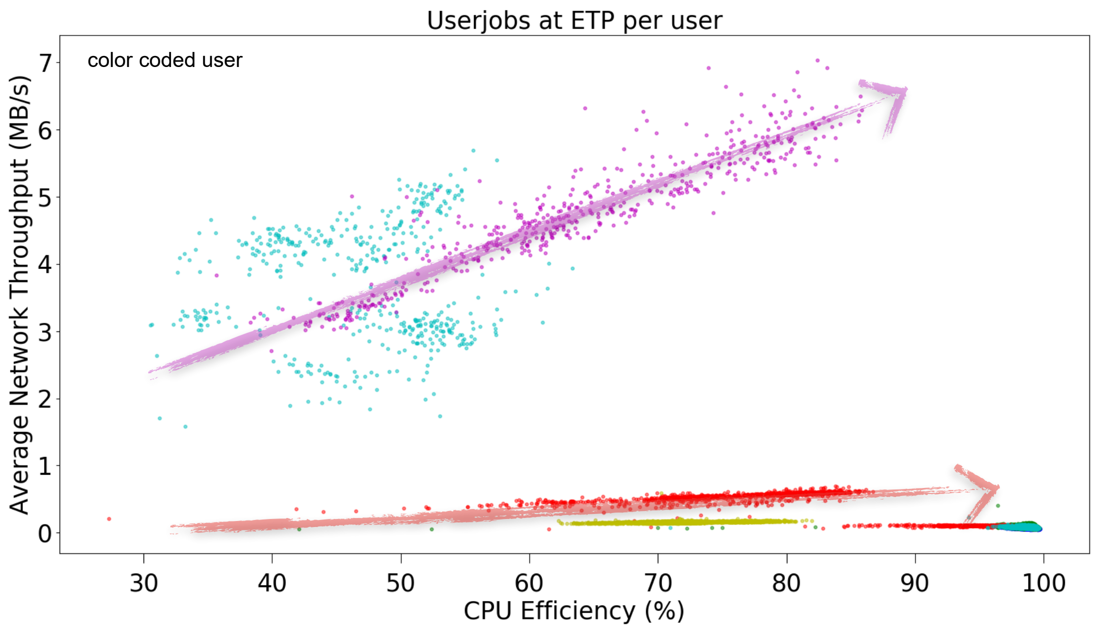
Current Work: HNSciCloud Integration

- Integration of HNSciCloud resources with GridKa Batch System
  - Separate HTCondor instance for Exoscale/OTC VMs
  - Use Cloud resources *if* adequately utilised by Pilots
- TARDIS: Drone Pools aggregated in dynamically sized Composite
  - Each drone represents and manages single VM
  - Scalability from asynchronicity and composition



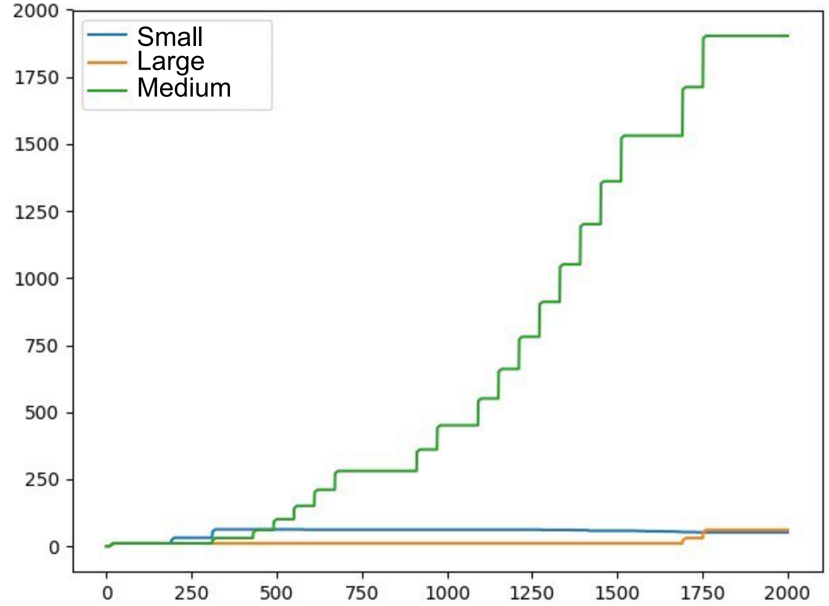
Current Research: Implicit Network Scheduling

- Integrate Network availability and congestion into provisioning
  - Congested network as bottleneck for opportunistic resources
  - Non-linear interference and noticeable measurement overhead
- Implicitly respect network congestion via side-effects
  - Prove CPU efficiency as utilisation to reflect congestion
  - Show benefit/limitation of optimising for CPU utilisation



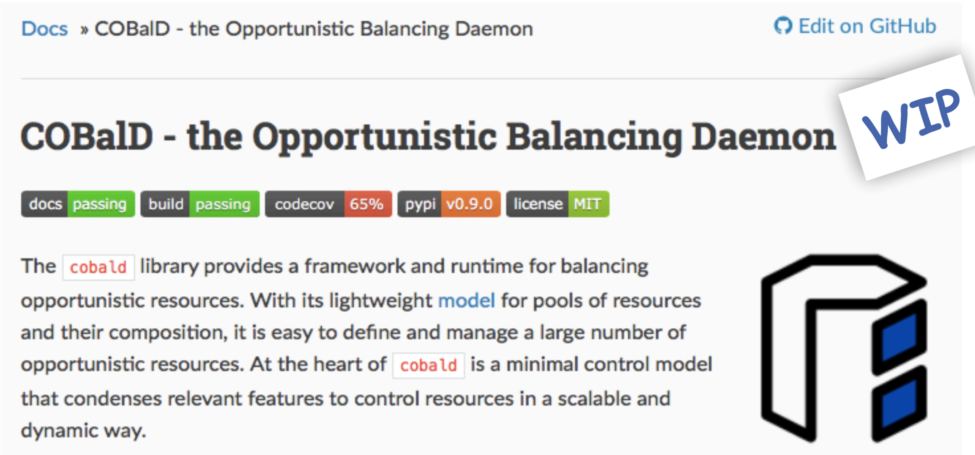
Current Research: Simulation and Strategies

- Modular design allows to replace active components
  - Simulated resource acquisition and utilisation with real Controllers
  - Replicated most of HTCondor queue and scheduling functionality
- Investigating optimal control strategies for various situations
  - Responsiveness to fast increase/decrease in job pressure
  - Multiple pools providing different resources for job sizes



Conclusion

- Long History of using Opportunistic Resources for Physics
  - Easy to get them, hard to use them
  - We are now beyond being able to micromanage
- COBalD: lightweight approach to Opportunistic Resources
  - Rely on underlying resource broker for resource selection
  - Request resources based on tracking actual usage
  - Applicable beyond classical batch system scenario


Docs » COBalD - the Opportunistic Balancing Daemon

Edit on GitHub

COBalD - the Opportunistic Balancing Daemon

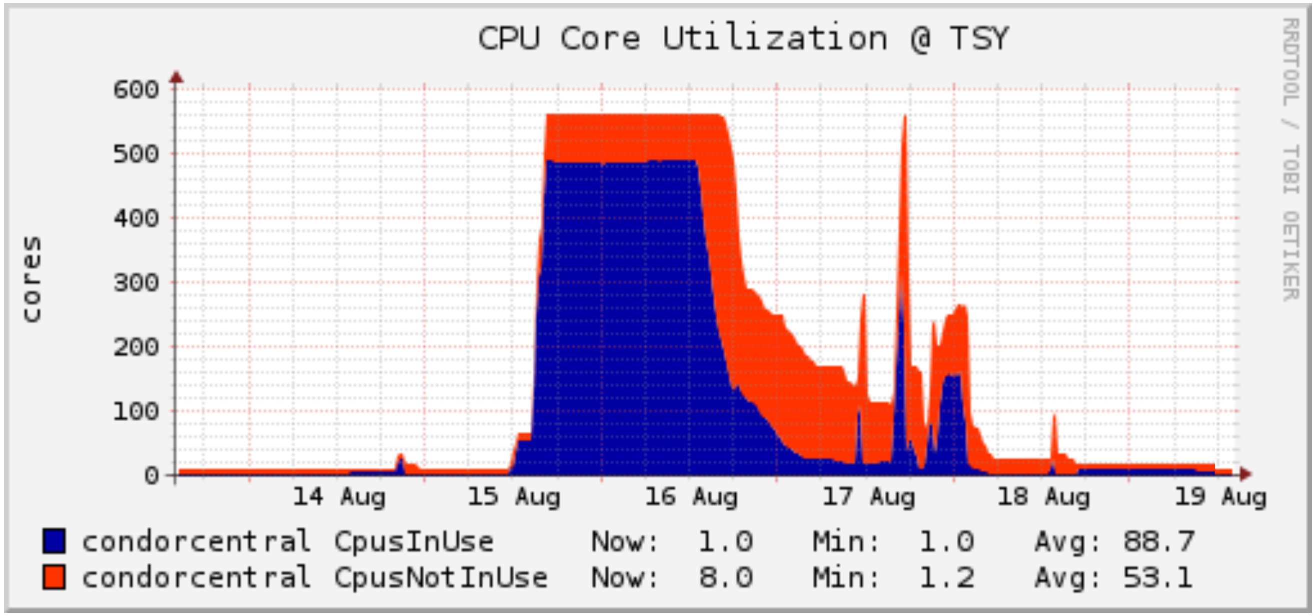
The `cobald` library provides a framework and runtime for balancing opportunistic resources. With its lightweight model for pools of resources and their composition, it is easy to define and manage a large number of opportunistic resources. At the heart of `cobald` is a minimal control model that condenses relevant features to control resources in a scalable and dynamic way.

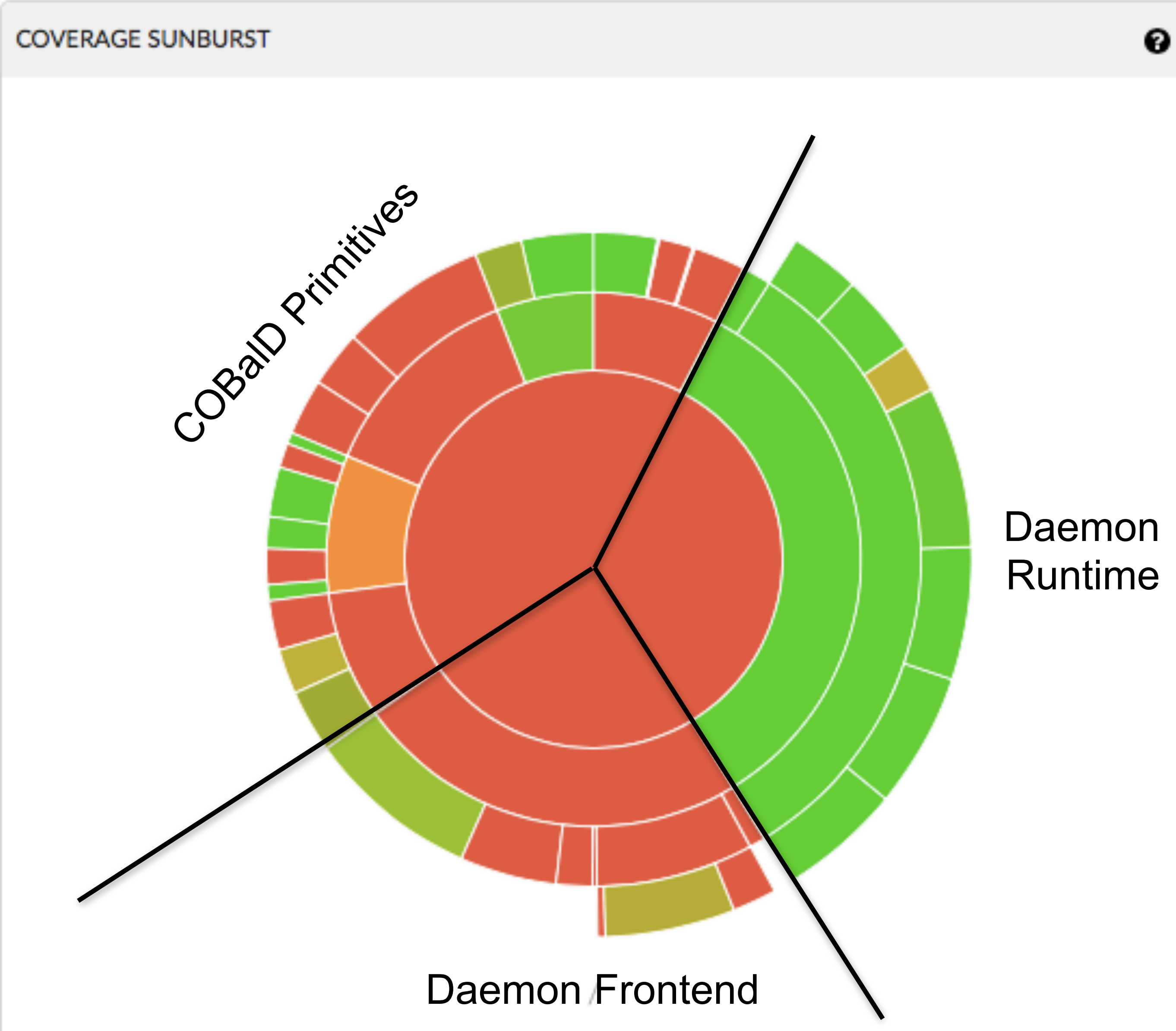
Backup

Resources

- COBalD: http://cobald.readthedocs.io/
- ROCED: https://github.com/roced-scheduler/ROCED
- TARDIS: https://github.com/giffels/tardis
- COBalD Simulation: https://git.scc.kit.edu/fq8360/cobalt_sim
- COBalD Demo: https://github.com/MaineKuehn/cobald_demo

ROCED Efficiency in HNSciCloud



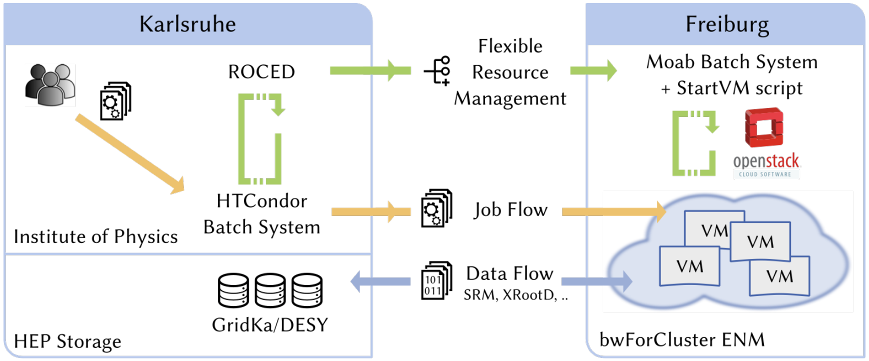

COVERAGE SUNBURST
COBalD Primitives
Daemon Runtime
Daemon Frontend

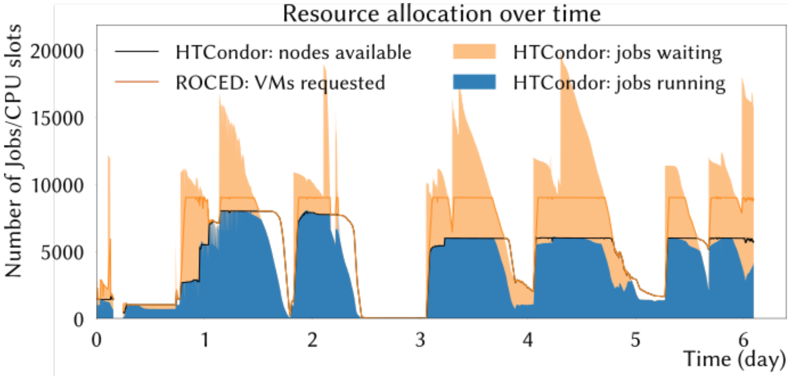
Opportunistic Resources

- ForHLR2 (KIT) - HPC
  - Backfilling and cycle stealing
  - Docker/Singularity container and permanent, shared cvmfs cache
- NEMO (Freiburg) - HPC
  - Fairshare in batch system
  - Virtual machines and semi-permanent Squids for cvmfs
- 1und1, Amazon, … - Public Cloud
  - Bought/donated resources
  - Virtual machines and temporary Squids for cvmfs
- ETP Desktop (KIT) - Desktops
  - Cycle stealing (day), temporary worker nodes (night)
  - Docker Container and permanent Squids for cvmfs
- GridKa - Grid Site
  - Fairshare in batch system
  - Pilot jobs and existing cvmfs

Success Story - Opportunistic “Tier 1“ for a Day

- Dynamically shared HPC Centre at Freiburg (three diverse communities)
- Virtualization is key component to:
  - Allow dynamic resource partitioning
  - Meet OS & software requirements
- ROCED cloud scheduler developed at KIT
  - On-demand resource provisioning
  - Transparent resource integration
- Suitable for CPU-intense workflows

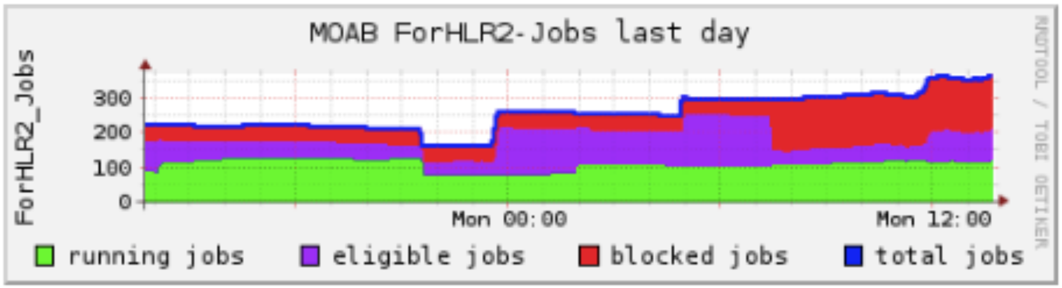


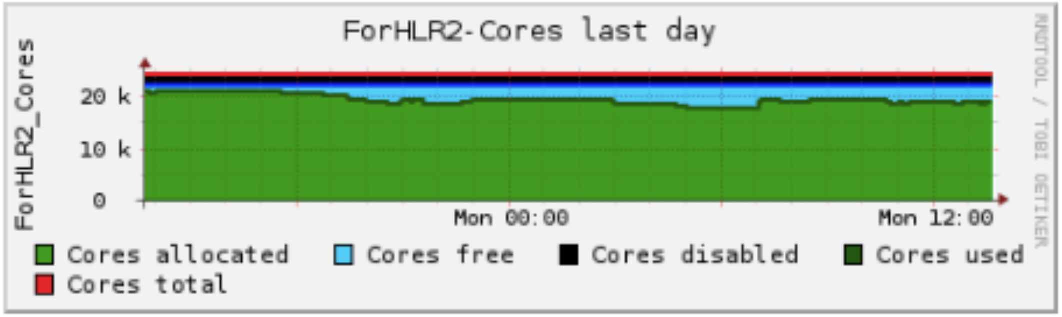


[ACAT17MS, JOP898TH, JOP762TH, JOP664TH]

Typical situation at HPC clusters

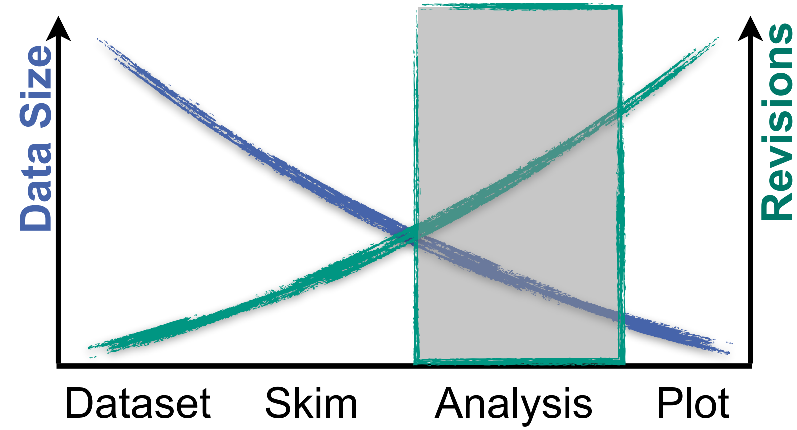
- small number of big multi node jobs
- unused cores / nodes due to scheduling of big jobs
- could back filled with short running single core jobs (e.g. HEP jobs)



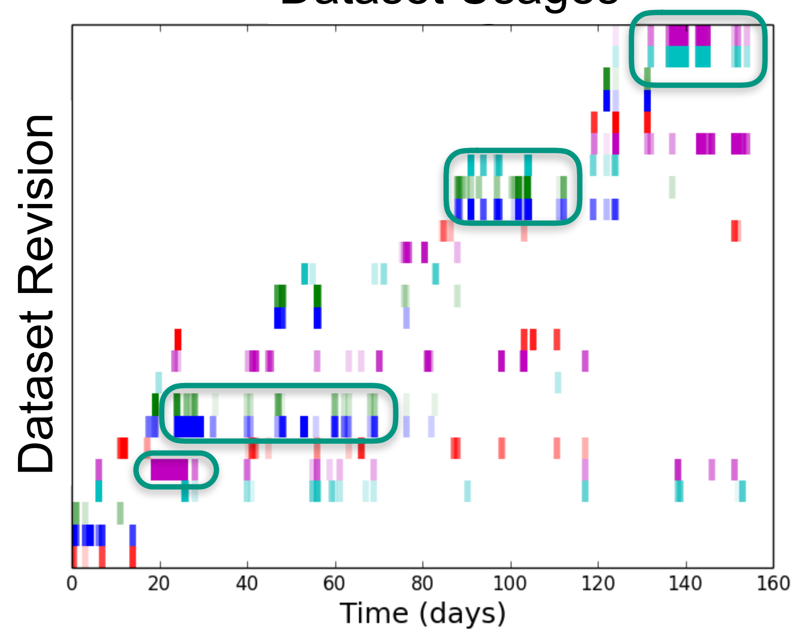


Context: HEP End User Data Analysis

- Hierarchical, iterative workflows
  - Reduction of data size
  - Increase of iterations
  - Dedicated processing environments
- Data intense analyses on Tier 3
  - Local batch system
  - Network accessed fileservers
  - Optimized input data sets
- Usage suitable for caching
  - Repeated processing of same input
  - Highly dependent on input rate
  - Limited by network bandwidth

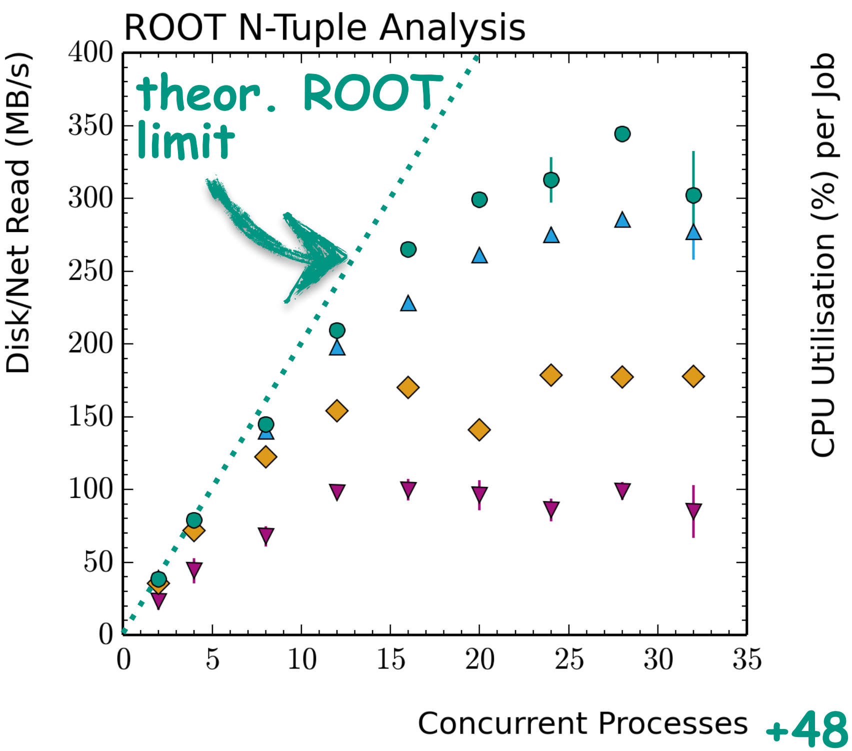


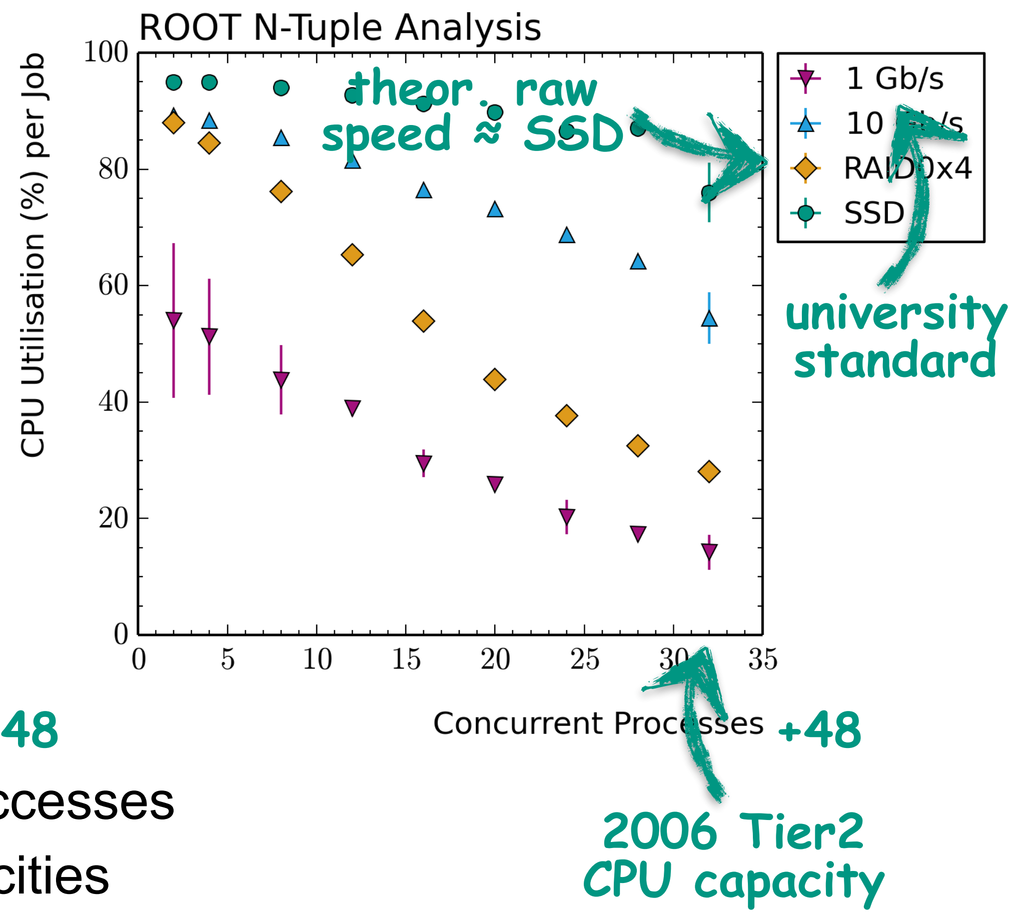
Dataset Usages



I/O Performance Evaluation

- CMS jet calibration analysis (ROOT n-tuple)
- Additional 48 concurrent reads from other workers for 10 Gb/s test

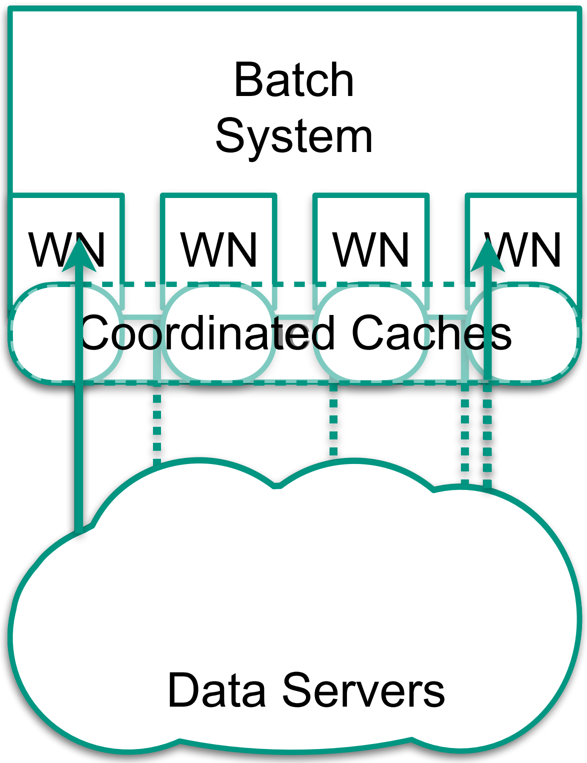




- HDDs limited on concurrent accesses
- SSDs exploit full system capacities

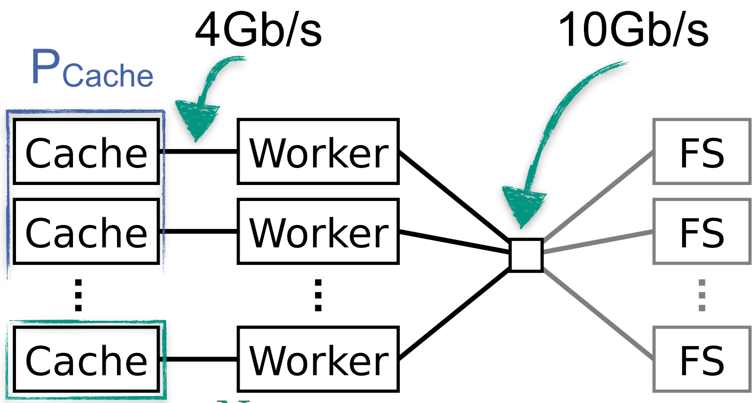
Coordinated Caching: Overview

- Caching between batch system and data sources
  - Consumer focused caching
  - Partial data locality for remote files
- Abstracts cache to batch system scale
  - Utilize meta-data of entire user workflows
  - Works on files used by jobs
- Implementation at host granularity
  - Array of individual caches on worker nodes
  - Caches coordinated by global service
  - Some glue for data locality…

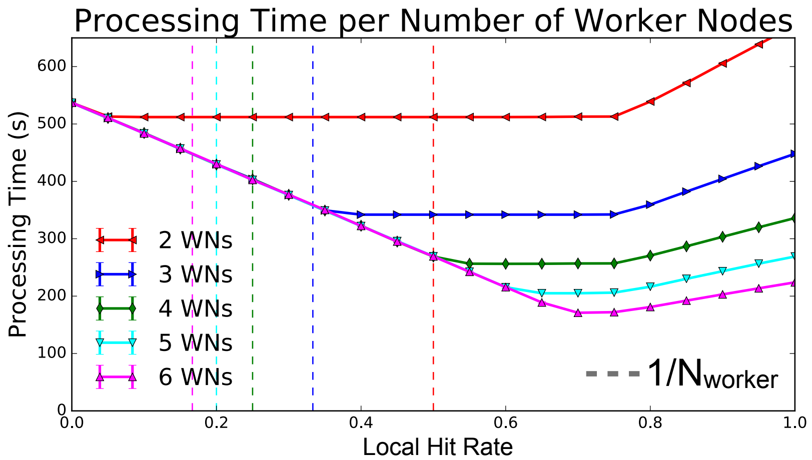


Coordinated Caching: Throughput

- Batch system throughput simulation
  - Setup of KIT Tier3
  - Parameters: local hit rate, N_{worker}
- Caching allows horizontal scaling
  - Throughput scales with workers…
  - …if jobs are scheduled to data
- Perfect hit rate not ideal
  - Balance remote and cache I/O
  - Potential to…
    - Use simple heuristics
    - Increase effective cache size

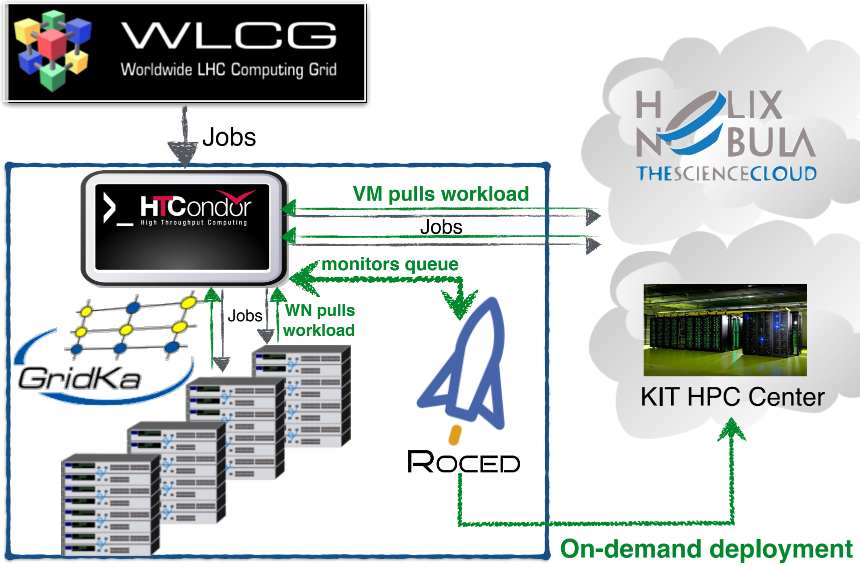


$$P_{Local} = \frac{N_{local}}{N_{total}}$$



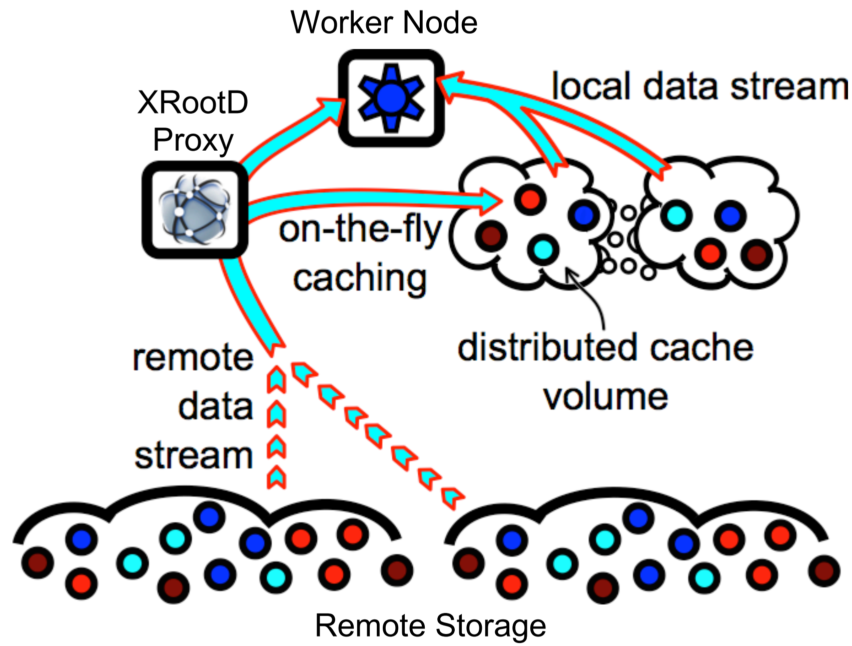
Dynamic Compute Expansion of GridKa Tier 1

- Transparent on-demand integration of opportunistic resources using ROCED
  - Helix Nebula Science Cloud (based on traditional virtualization)
  - KIT HPC Center (FORHLR II) (based on container technology)
- Automated detection and redirection of suitable CPU-intense workflows
- Evaluate ML for scheduling optimizations [JSSPP18MS]



Caching Concepts on Opportunistic Sites

- Opportunistic Resources usually well suited for CPU-intense workflows
- Many opportunistic sites offering fast cloud storage or distributed storage
- Benefit from caching R&D and bring recurrent I/O-intense workflows to the cloud
- Transparent data access also a hot topic in Helix Nebula Science Cloud



Collaboration in developing a xrootd based caching proxy between KIT and GSI

[ACAT17CH]

Resources at KIT

- small local institute cluster with HTCondor
- grid cluster and storage at GridKa
- opportunistic computing resources dynamic included via ROCED in HTCondor pool
  - HPC center at KIT
  - HPC center at Freiburg
  - HNSciCloud
    - OpenTelekomCloud

Scientific Cloud

HPC center at Freiburg

HPC center at KIT

Grid storage at GridKa

Commercial Cloud

ETP