Karlsruher Institut für Technologie

GridKa

High Performance Opportunistic Data Analysis for HEP/HTC computing

ErUM Data Kickoff Meeting

M. Fischer, M. Giffels, C. Heidecker, F. Cube, et al

Steinbuch Centre for Computing / Institute for Experimental Particle Physics

R&D Environment and Background

- Two collaborating HEP Computing Groups at KIT
  - SCC: GridKa Tier 1, focus on throughput and production systems
  - ETP: Institute Tier 3, focus on responsiveness and prototypes

R&D Environment and Background

- Two collaborating HEP Computing Groups at KIT
  - SCC: GridKa Tier 1, focus on throughput and production systems
  - ETP: Institute Tier 3, focus on responsiveness and prototypes

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

R&D Environment and Background

- Two collaborating HEP Computing Groups at KIT
  - SCC: GridKa Tier 1, focus on throughput and production systems
  - ETP: Institute Tier 3, focus on responsiveness and prototypes
- Prior work on opportunistic data processing
  - On-demand processing resources via VMs/container („ROCED“)
    T. Hauth et al., On-demand provisioning of HEP Compute Resources on Clouds Sites and Shared HPC Centers, Journal of Physics **898**, 5 (2017)
  - Adaptive placement of input data via Caches („HPDA“)
    M. Fischer et al., Opportunistic Data Locality for End User Data Analysis, Journal of Physics **898**, 5 (2017)
  - Runtime classification of payloads in overlay batch systems
    E. Kühn et al., A scalable architecture for online anomaly detection of WLCG batch jobs, Journal of Physics **762**, 1 (2016)

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

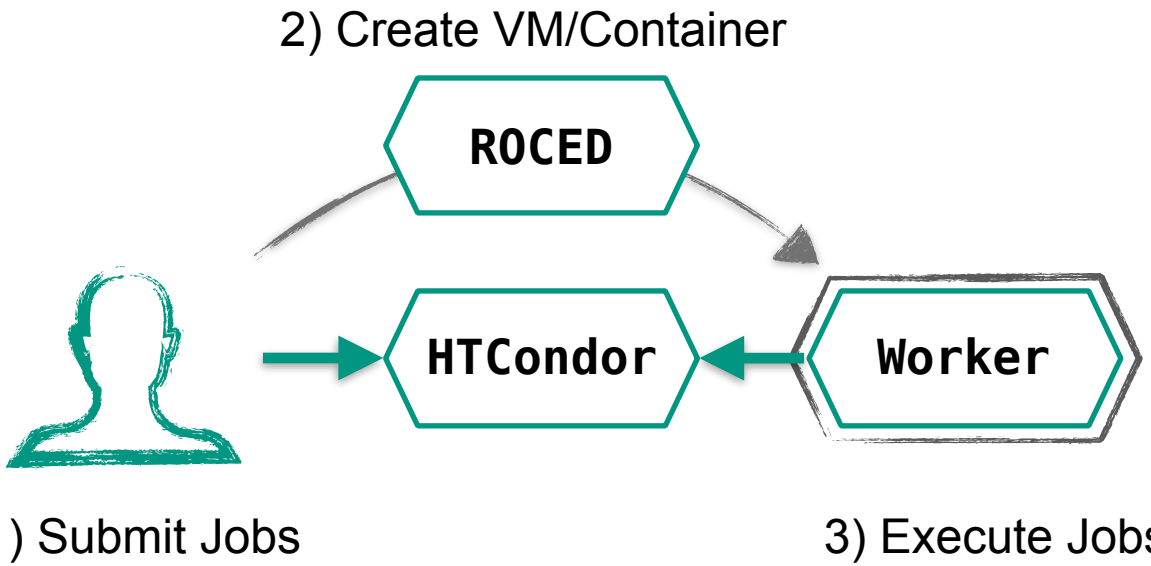
R&D Environment and Background

- Two collaborating HEP Computing Groups at KIT
  - SCC: GridKa Tier 1, focus on throughput and production systems
  - ETP: Institute Tier 3, focus on responsiveness and prototypes
- Prior work on opportunistic data processing
  - On-demand processing resources via VMs/container („ROCED“)
    T. Hauth et al., On-demand provisioning of HEP Compute Resources on Clouds Sites and Shared HPC Centers, Journal of Physics **898**, 5 (2017)
  - Adaptive placement of input data via Caches („HPDA“)
    M. Fischer et al., Opportunistic Data Locality for End User Data Analysis, Journal of Physics **898**, 5 (2017)
  - Runtime classification of payloads in overlay batch systems
    E. Kühn et al., A scalable architecture for online anomaly detection of WLCG batch jobs, Journal of Physics **762**, 1 (2016)
- 2 PhD Students for ErUM Topic A
  - Christoph Heidecker (Caching), Florian von Cube (Cloud)

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

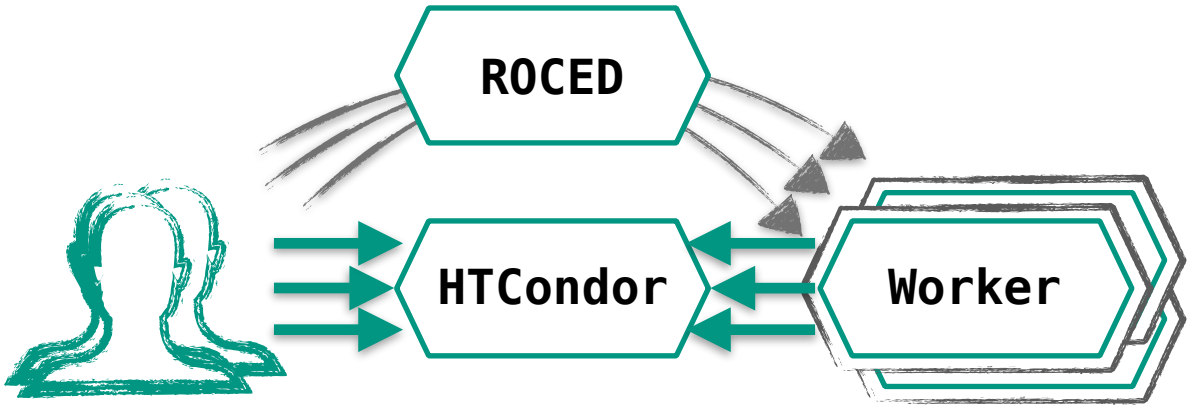
Opportunistic Processing: ROCED (2010-present)

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



Opportunistic Processing: ROCED (2010-present)

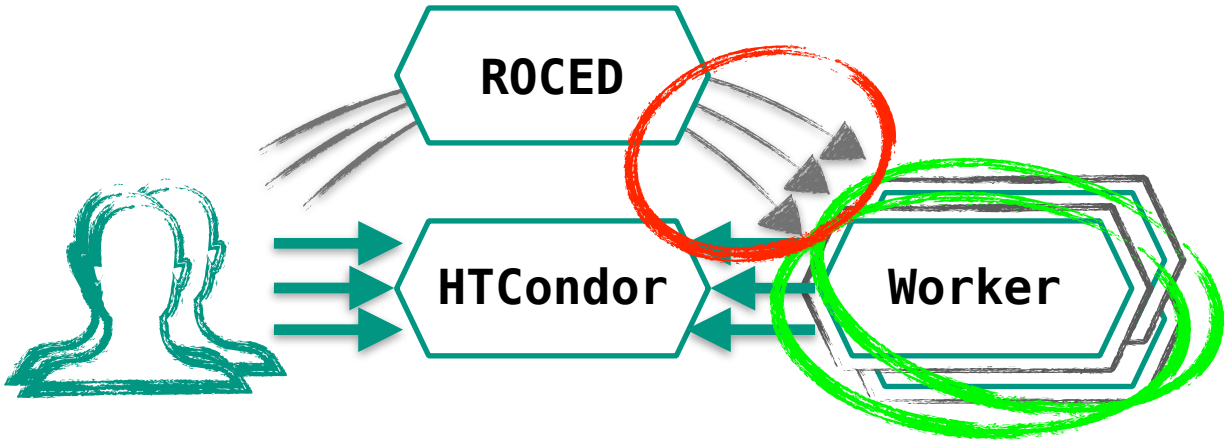
[**R**esponsive **O**n-**D**emand **C**loud **E**nabled **D**eployment]



Opportunistic Processing: ROCED (2010-present)

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

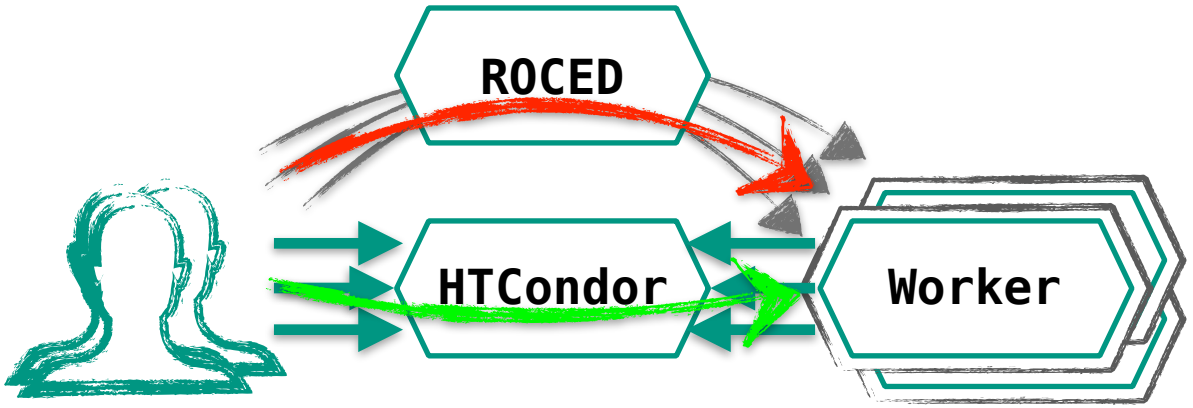
- Dynamic resources matching user demand
  - Trivial to support new providers for many users
  - Difficult to manage several providers for many users



Opportunistic Processing: ROCED (2010-present)

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

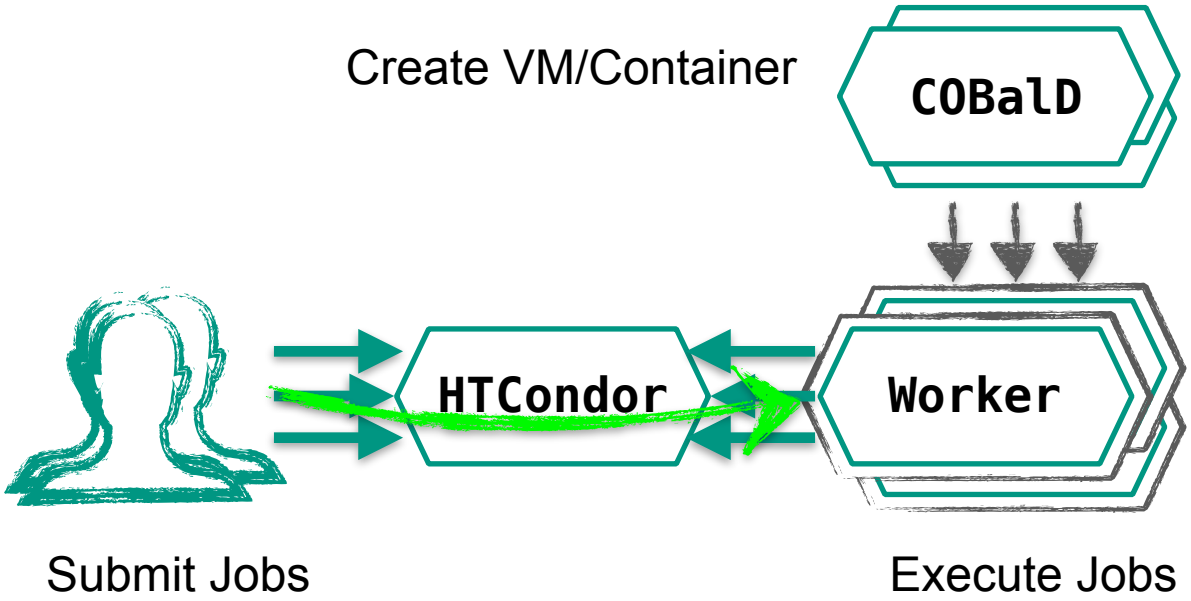
- Dynamic resources matching user demand
  - Trivial to support new providers for many users
  - Difficult to manage several providers for many users
- Resource aggregation in overlay batch system
  - Unreliable to predict resources required for jobs
  - Efficient to integrate resources, then match jobs



Opportunistic Processing: ROCED (2010-present)

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

- 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

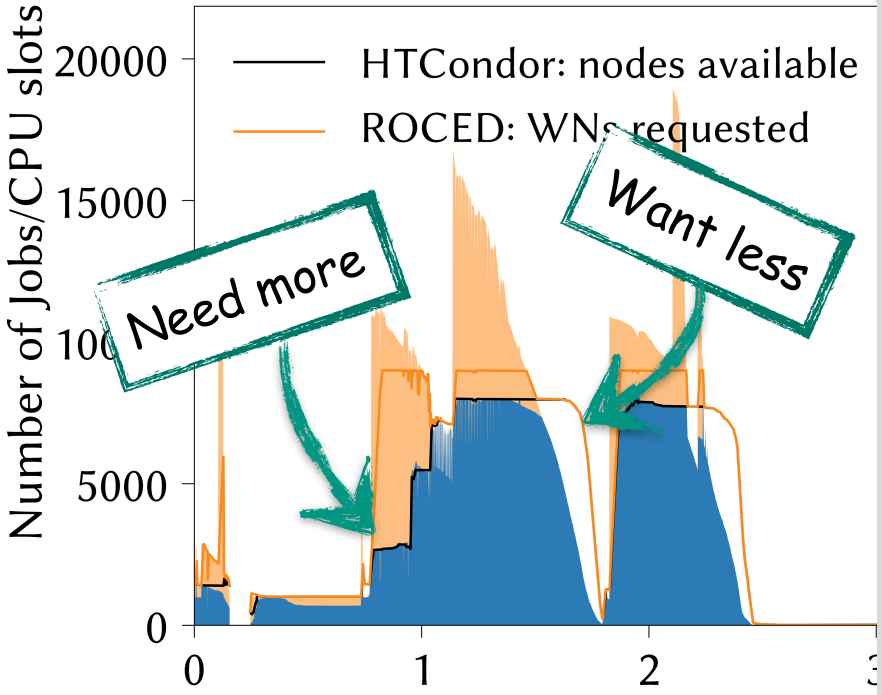


Simpler approach: COBalD (2018-present)

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

Simpler approach: COBalD (2018-present)

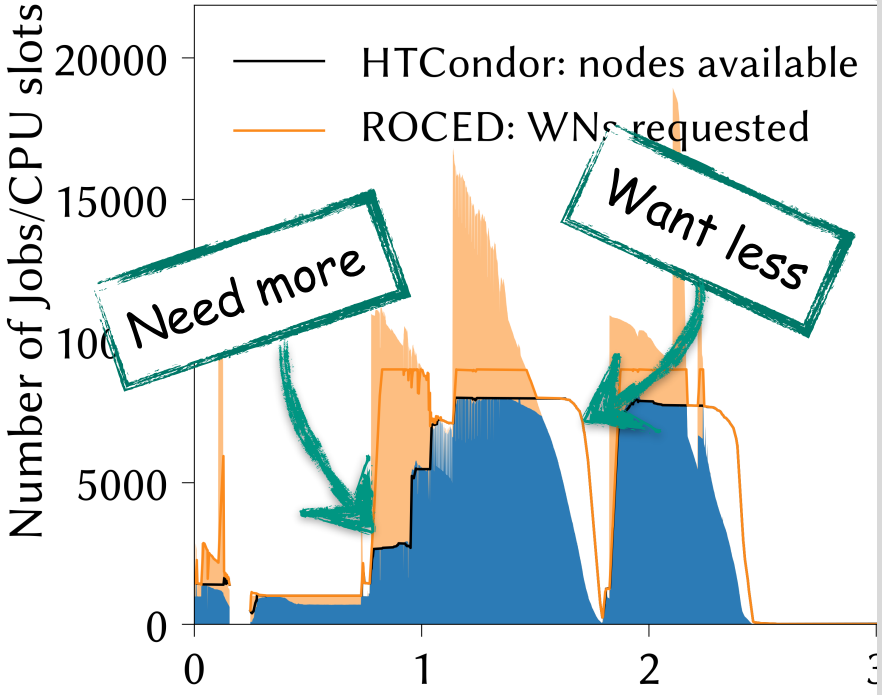
[**C**OBalD - the **O**pportunistic **Bal**ancing **D**aemon]



Simpler approach: COBalD (2018-present)

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

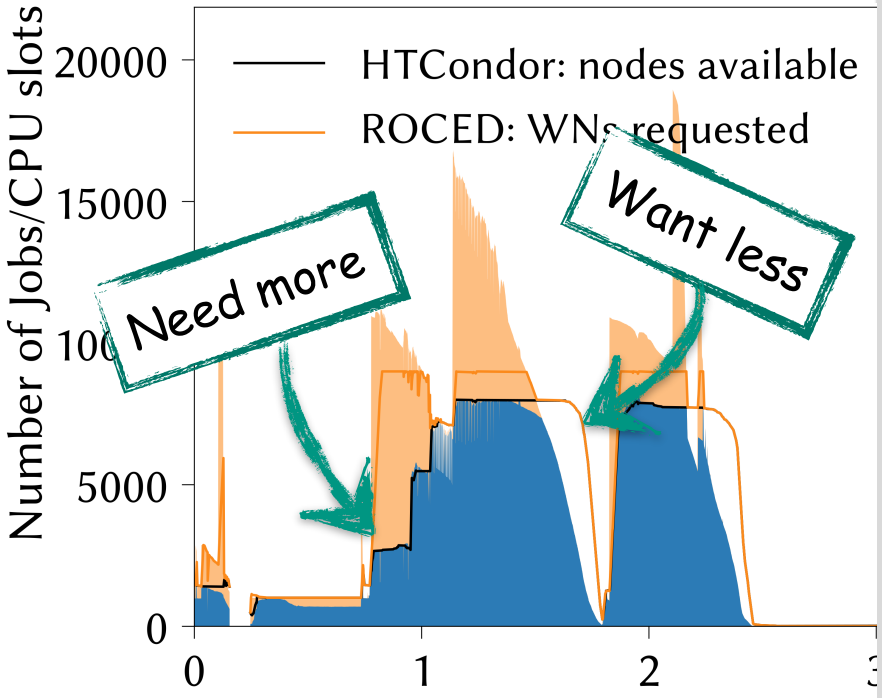
- Look at what is used, not what is requested
  - Simple logic: more used resources, less unused resources
  - Regular batch system scheduler decides what to use
  - COBalD only creates/destroys resources



Simpler approach: COBalD (2018-present)

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

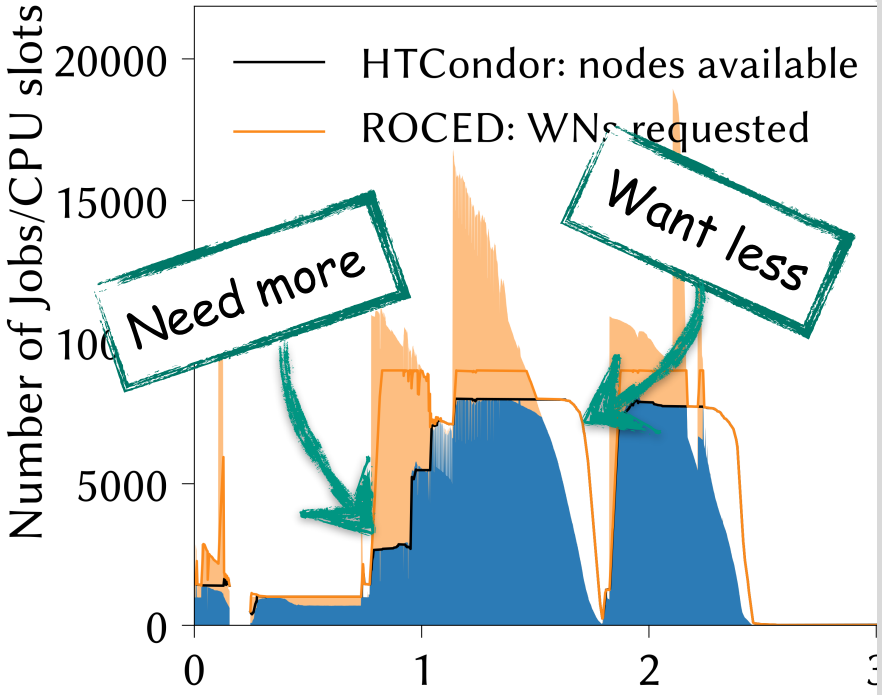
- Look at what is used, not what is requested
  - Simple logic: more used resources, less unused resources
  - Regular batch system scheduler decides what to use
  - COBalD only creates/destroys resources
- Generic design for any resources
  - COBalD just knows (un-)used
  - CPU, CPU+RAM, CPU+DISK, CPU+DISK+RAM, …
  - Partitioned multi-core slots



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 resources, less unused resources
  - Regular batch system scheduler decides what to use
  - COBalD only creates/destroys resources
- Generic design for any resources
  - COBalD just knows (un-)used
  - CPU, CPU+RAM, CPU+DISK, CPU+DISK+RAM, …
  - Partitioned multi-core slots
- Native composition / concurrency
  - Aggregate same resources to entire pools managed as one
  - Several COBalDs manage pools for same overlay batch system



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

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

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

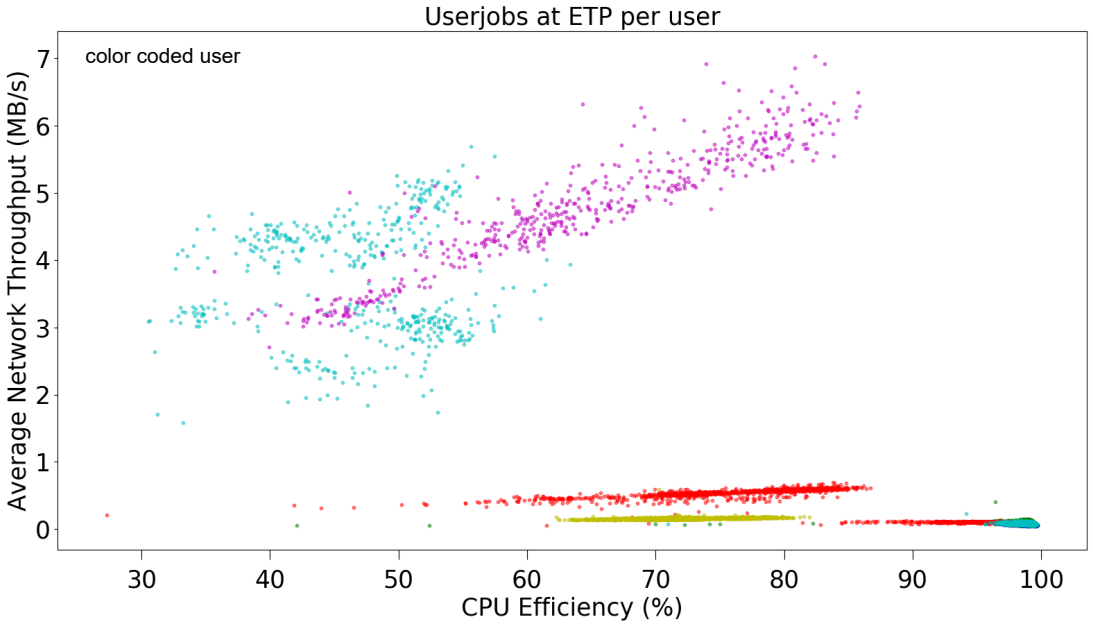
We don't even have a word for these thingies

Current Research: 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

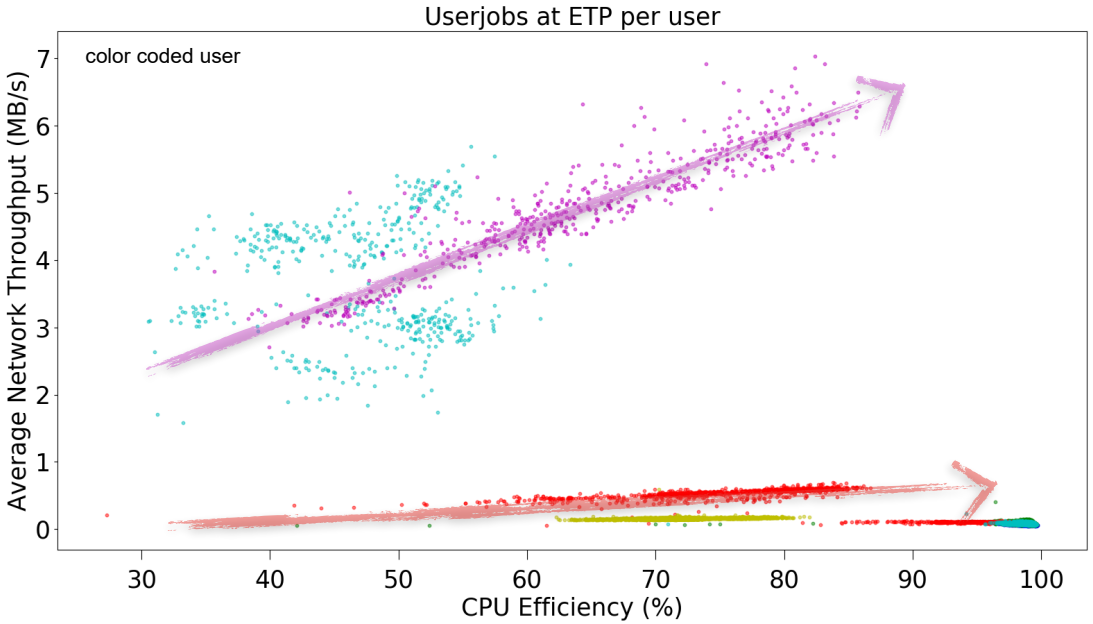
Current Research: Implicit Network Scheduling

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



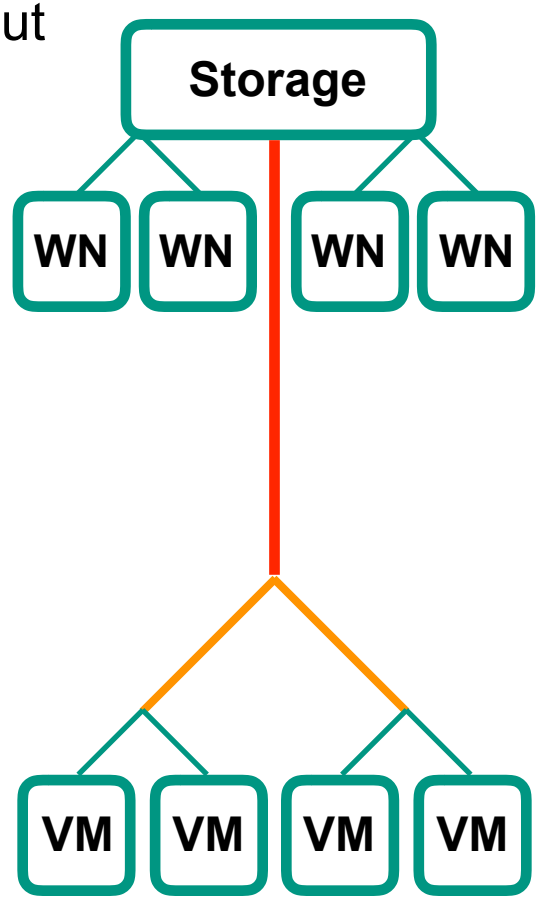
Current Research: Implicit Network Scheduling

- 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
  - Prove CPU usage as indication for network congestion
  - Show benefit/limitation of optimising for CPU utilisation



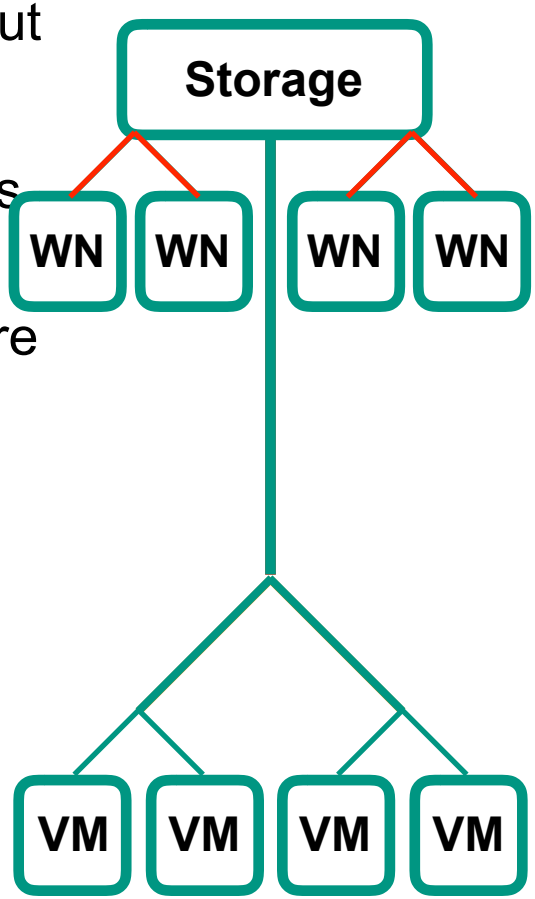
Opportunistic *Data* Processing

- Opportunistic resources unsuited for data processing
  - Jobs read data from remote grid storage
  - Congestion of connections limits total throughput
  - Non-linear limits from multiple SEs and sites



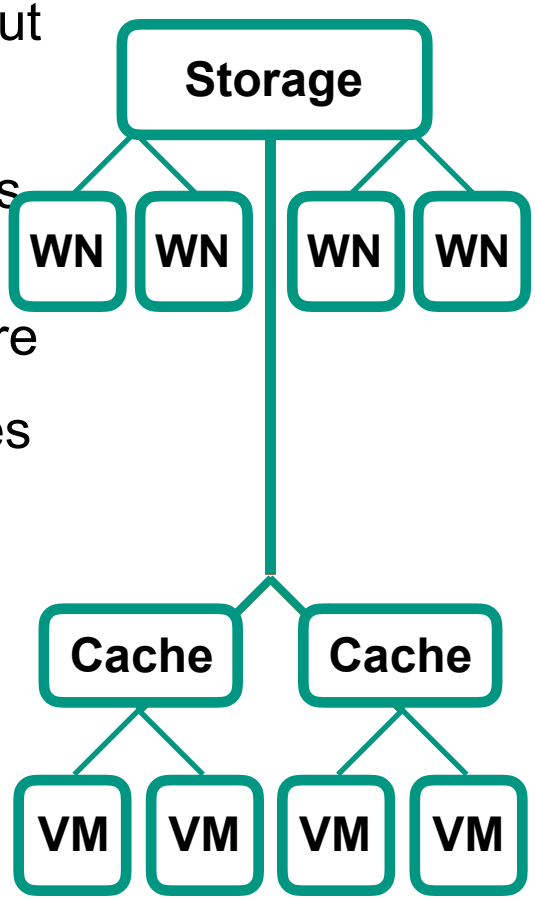
Opportunistic *Data* Processing

- Opportunistic resources unsuited for data processing
  - Jobs read data from remote grid storage
  - Congestion of connections limits total throughput
  - Non-linear limits from multiple SEs and sites
- A) Avoid data processing on opportunistic resources
  - Keep data intensive jobs in grid sites
  - Changes job mix on dedicated grid infrastructure



Opportunistic *Data* Processing

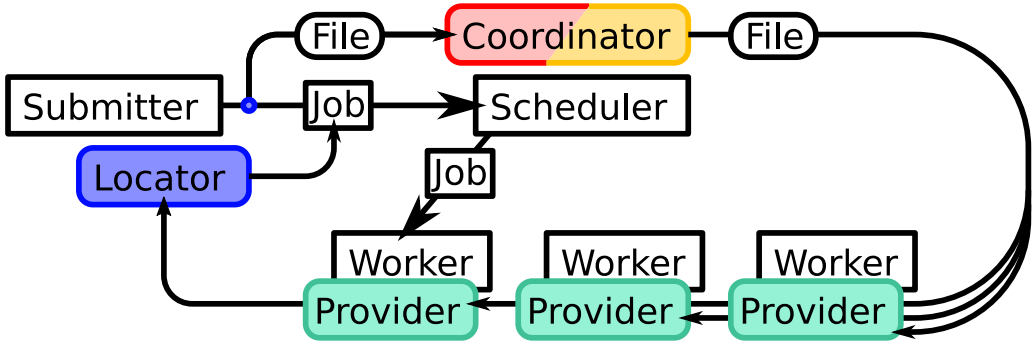
- Opportunistic resources unsuited for data processing
  - Jobs read data from remote grid storage
  - Congestion of connections limits total throughput
  - Non-linear limits from multiple SEs and sites
- A) Avoid data processing on opportunistic resources
  - Keep data intensive jobs in grid sites
  - Changes job mix on dedicated grid infrastructure
- B) Reduce data access from opportunistic resources
  - Reuse transfered input data via caching
  - Only for recurring, high performance jobs



Challenges for Distributed Opportunistic Caching

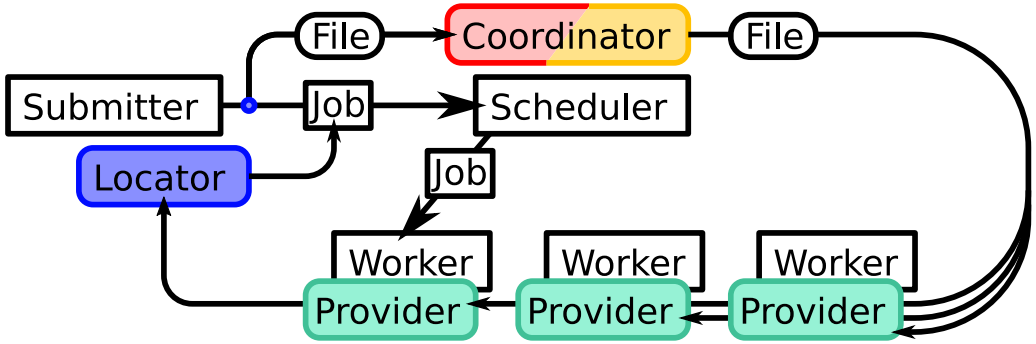
Challenges for Distributed Opportunistic Caching

- Distributed caches require coordination to stay consistent
  - Degradation from random access and duplicates (loss$\propto N_{Hosts}$)
  - Individual nodes see only fraction of metadata



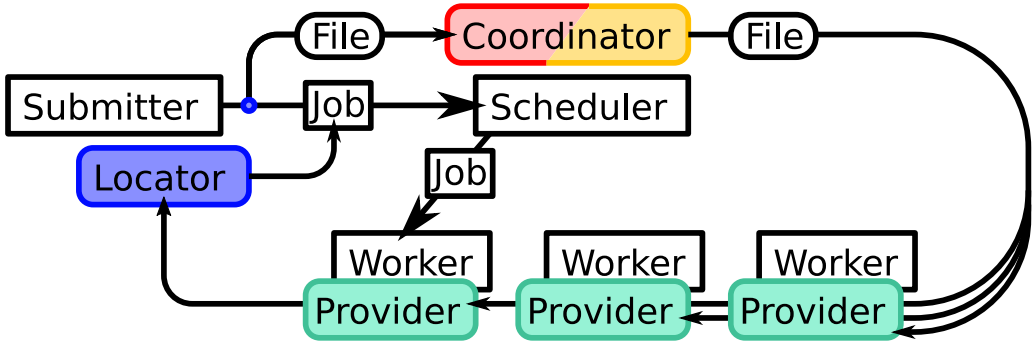
Challenges for Distributed Opportunistic Caching

- Distributed caches require coordination to stay consistent
  - Degradation from random access and duplicates (loss$\propto N_{Hosts}$)
  - Individual nodes see only fraction of metadata
- Perfect cache hit rates are undesirable
  - Infrastructure still offers considerable network bandwidth
  - Caching more data than needed wastes cache space

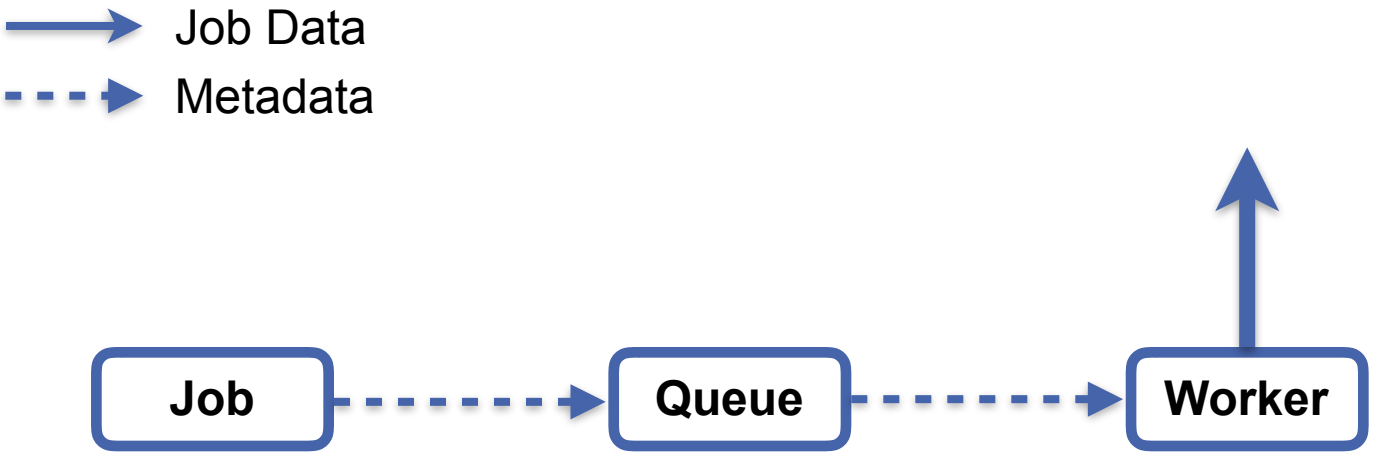


Challenges for Distributed Opportunistic Caching

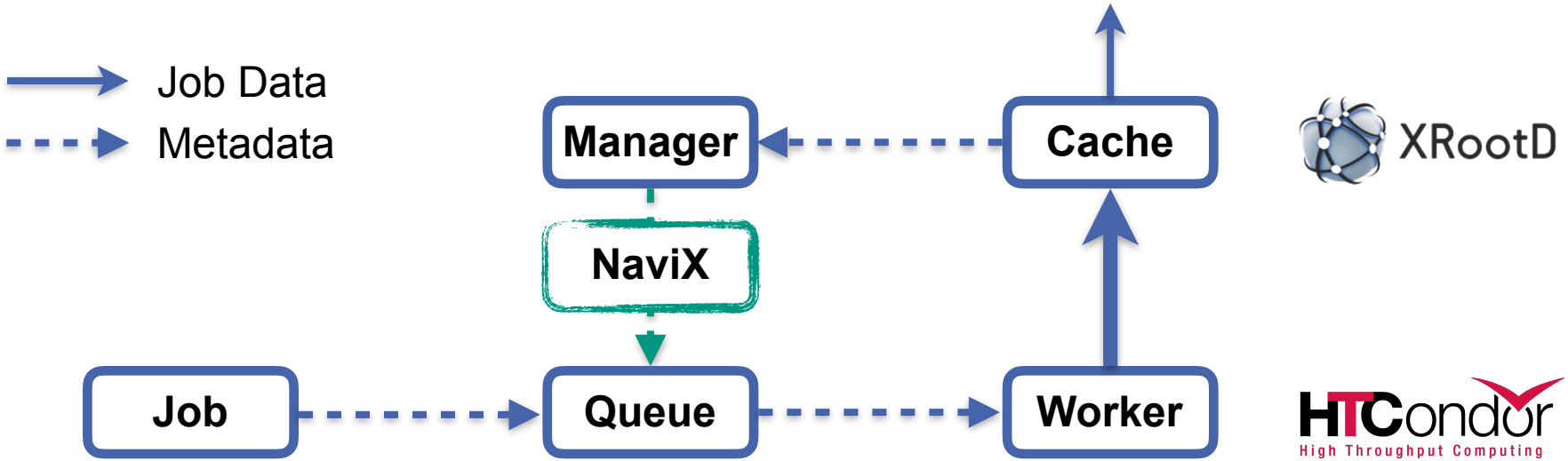
- Distributed caches require coordination to stay consistent
  - Degradation from random access and duplicates (loss$\propto N_{Hosts}$)
  - Individual nodes see only fraction of metadata
- Perfect cache hit rates are undesirable
  - Infrastructure still offers considerable network bandwidth
  - Caching more data than needed wastes cache space
- Data features requires specialised caching algorithms
  - Avoid trashing from access cycles of days to weeks
  - Only a fraction of workflows benefits from caching



Minimalist approach: NaviX (2017-present)

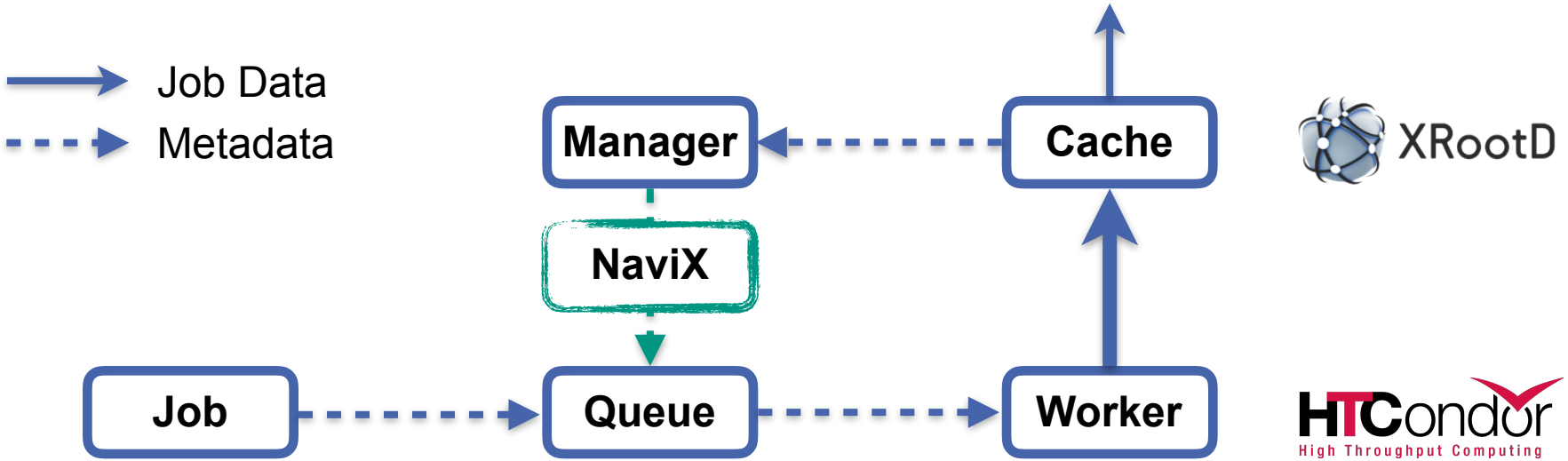


Minimalist approach: NaviX (2017-present)



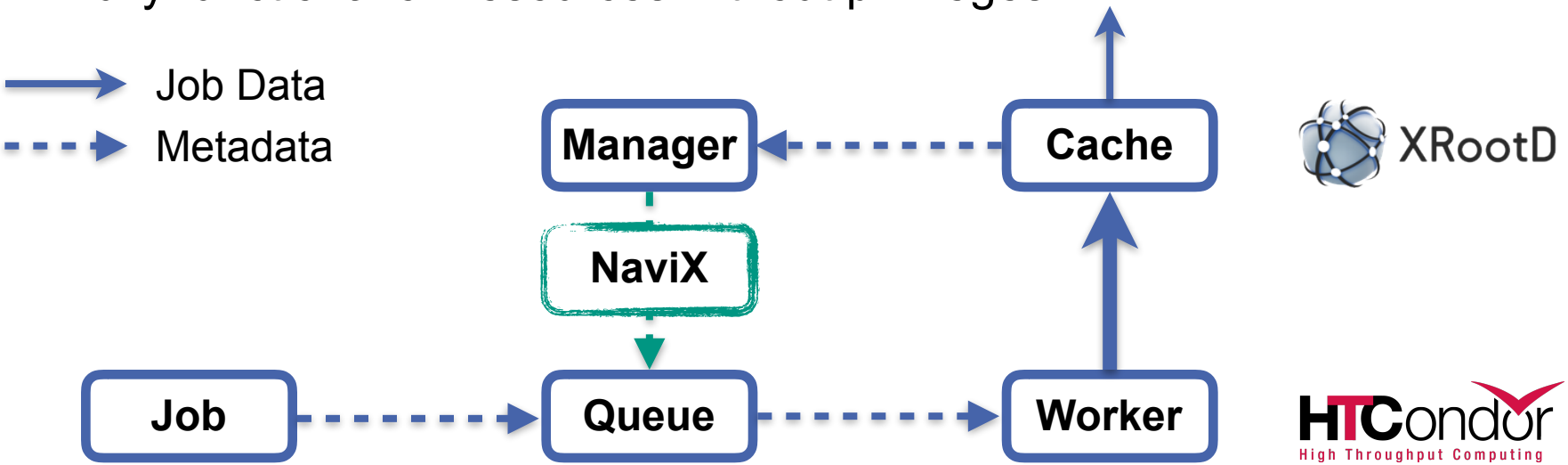
Minimalist approach: NaviX (2017-present)

- Orchestrate scalable, established middleware for core functionality
  - HTCondor for scheduling and coordination
  - XRootD for caching and localising



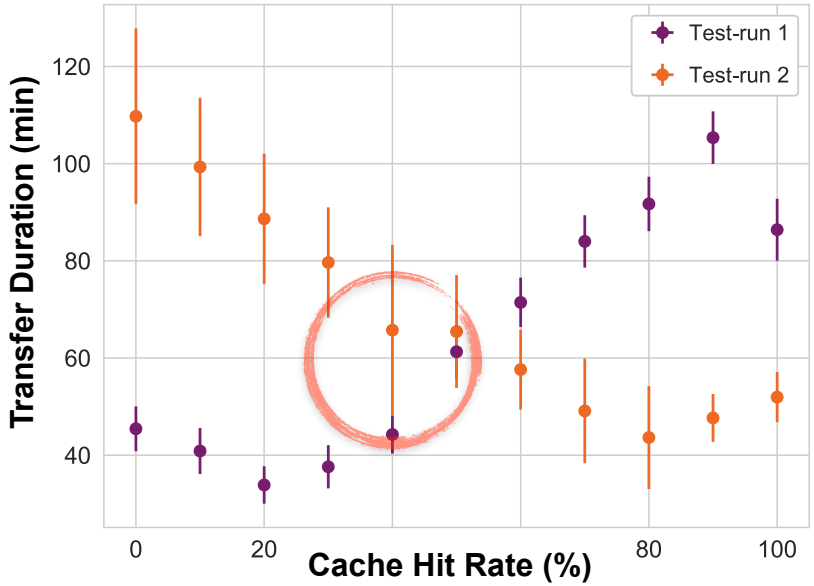
Minimalist approach: NaviX (2017-present)

- Orchestrate scalable, established middleware for core functionality
  - HTCondor for scheduling and coordination
  - XRootD for caching and localising
- Implicitly coordinate caches by pinning jobs
  - Prefer scheduling jobs with similar input to same caches
  - Reduces or eliminates need for coordination between caches
- Fully functional on resources without privileges



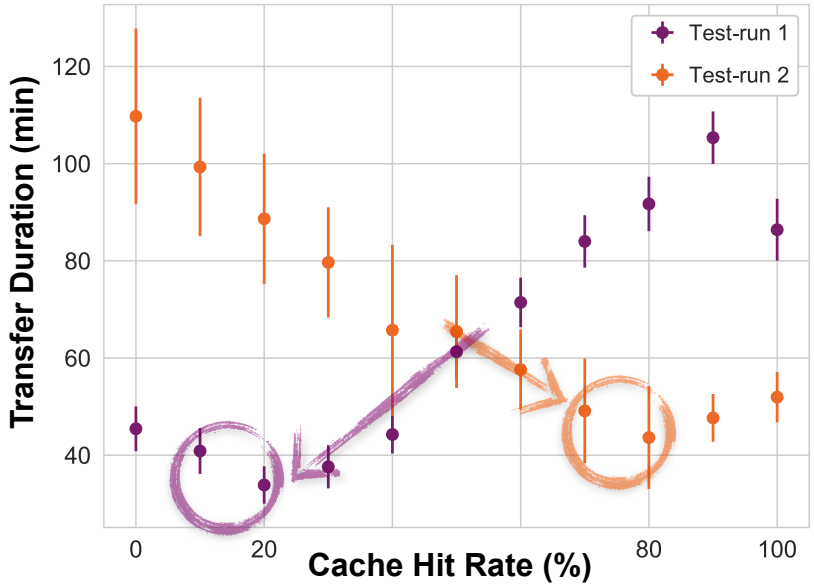
Current Research: Cooperative Data Delivery

- Cache versus remote is not a clear-cut decision at our scale
  - Network sufficient for most tasks, but overloads affect every task
  - Caches limited in capacity / throughput, but pinpoint accuracy



Current Research: Cooperative Data Delivery

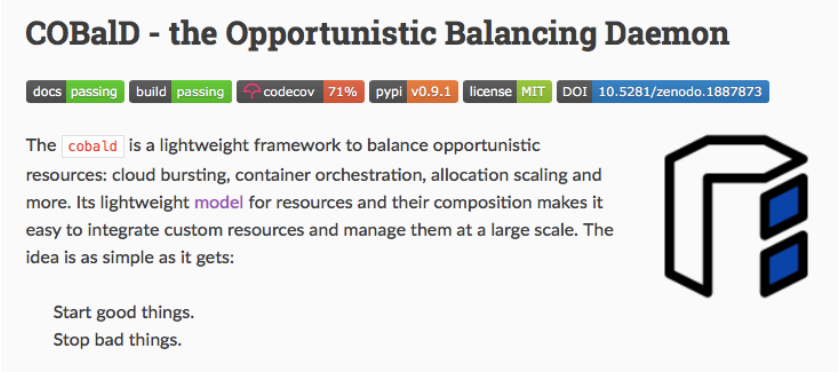
- Cache versus remote is not a clear-cut decision at our scale
  - Network sufficient for most tasks, but overloads affect every task
  - Caches limited in capacity / throughput, but pinpoint accuracy
- Research: Pinpoint cache tuning for throughput shortages
  - Separate distinct job classes and I/O environments
  - Dynamically derive locally optimal cache support



Conclusion

- Long history of using opportunistic processing resources
  - Dynamically scale out to handle peak job loads
  - Combine range of resource types and providers
  - Opportunistically manage data via caching
- Ongoing work to enable opportunistic data analysis
  - Implicitly include unaccountable infrastructure constraints
  - Unified caching for dedicated and opportunistic resources
  - Reformulate problems to simplify finding optimal conditions

NAVIX: https://gitlab.ekp.kit.edu/ETP-HTC/NaviX
COBalD: https://cobald.readthedocs.io
TARDIS: https://git.io/fhF8R



Backup

Resources

- COBalD: http://cobald.readthedocs.io/en/latest/
- NaviX: https://gitlab.ekp.kit.edu/ETP-XRootD/NaviX
- ROCED: https://github.com/roced-scheduler/ROCED
- HTDA: https://bitbucket.org/kitcmscomputing/hpda/src/master/

[ACAT17MS] M. Schnepf et al., Mastering Opportunistic Computing Resources for HEP, to be published (2017)
[ACAT17CH] C. Heidecker et al., Opportunistic Data Locality for HEP Analysis Workflows, to be published (2017)
[JOP898TH] T. Hauth et al., On-demand provisioning of HEP Compute Resources on Clouds Sites and Shared HPC Centers, Journal of Physics **898**, 5 (2017)
[KITPhDEK17] E. Kühn, Online Analysis of Dynamic Streaming Data, KIT Dissertation (2017)
[JOP898MF] M. Fischer et al., Opportunistic Data Locality for End User Data Analysis, Journal of Physics **898**, 5 (2017)
[JOP762TH] T. Hauth et al., Dynamic provisioning of a HEP computing infrastructure on a shared hybrid HPC system, Journal of Physics **762**, 1 (2016)
[JOP762EK] E. Kühn et al., A scalable architecture for online anomaly detection of WLCG batch jobs, Journal of Physics **762**, 1 (2016)
[IEEE16EK] E. Kühn et al., Online Distance Measurement for Tree Data Event Streams, IEEE 681-688, (2016)
[IARIA16MF] M. Fischer et al., Data Locality via Coordinated Caching for Distributed Processing, Cloud Computing 2016, 113-118 (2016)
[KITPhD16] M. Fischer, Coordinated Caching for High Performance Calibration using Z→μμ Events of the CMS Experiment, KIT Dissertation (2016) [JOP762MF] M. Fischer et al., Data Locality via Coordinated Caching for Distributed Processing, Journal of Physics **762**, 1 (2016)
[JOP664EK] E. Kühn et al., Active Job Monitoring in Pilots, Journal of Physics **664**, 5 (2015)
[JOP608EK] E. Kühn et al., Analyzing data flows of WLCG jobs at batch job level, Journal of Physics **608**, 1 (2015)
[ICDM15EK] E. Kühn et al., Clustering Evolving Batch System Jobs for Online Anomaly Detection, IEEE 1534-1535 (2015)
[JOP664MF] M. Fischer et al., High Performance Data Analysis via Coordinated Caches, Journal of Physics **664**, 9 (2015)
[JOP608MF] M. Fischer et al., Tier 3 batch system data locality via managed caches, Journal of Physics **608**,1 (2015)
[JOP664TH] T. Hauth et al., Dynamic provisioning of local and remote compute resources with OpenStack, Journal of Physics **664**, 2 (2015)

Karlsruher Institut für Technologie

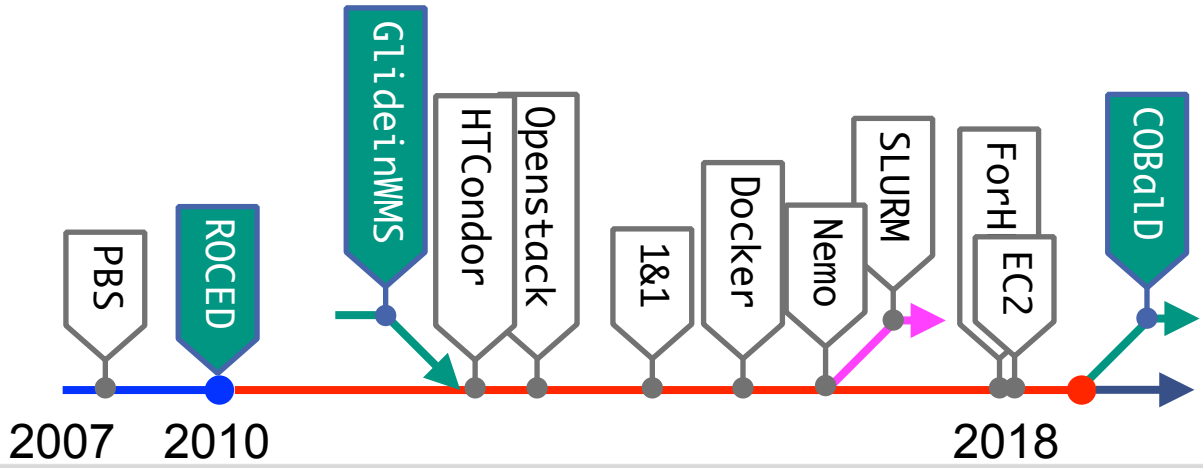
Backup Cloud

Steinbuch Centre for Computing / Institute for Experimental Particle Physics

Opportunistic Resources

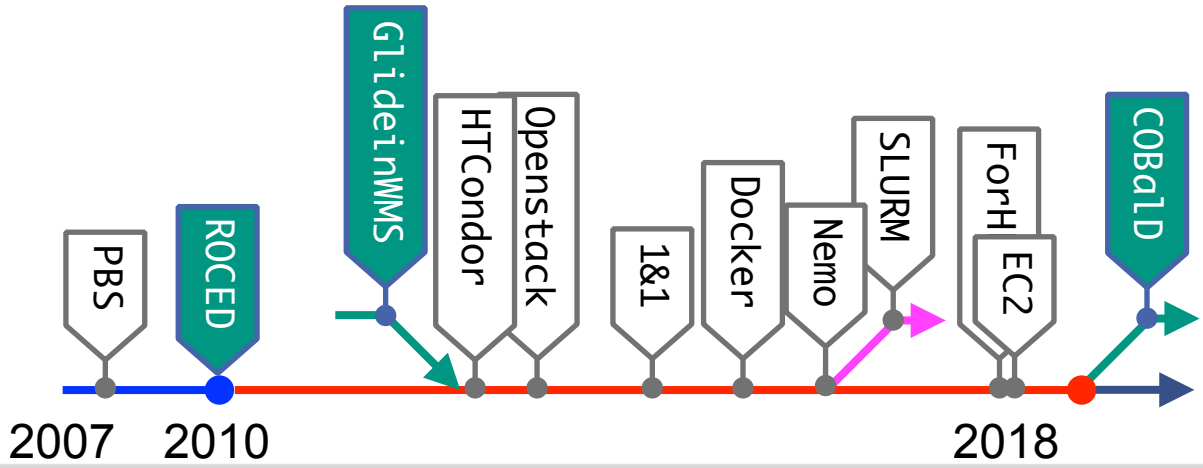
- ForHLR2 (KIT) - HPC
  - Backfilling and cycle stealing
  - Docker/Singularity container and permanent, shared cvmfs cache
- NEMO (Freiburg) - HPC
  - Fairshare in batch system
  - Virtual machines and semi-permanent Squids for cvmfs
- 1und1, Amazon, … - Public Cloud
  - Bought/donated resources
  - 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

Our Experience in a Nutshell



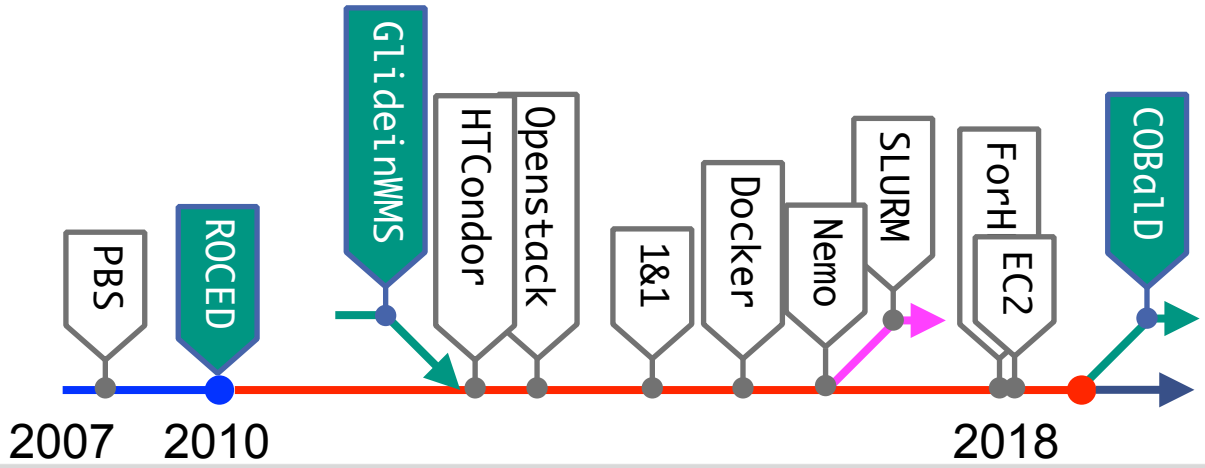
Our Experience in a Nutshell

- Need to look beyond fetching resource A, B, … for job 1, 2, …
  - Integrating resources is a solved problem
  - Micromanaging does not work/scale well



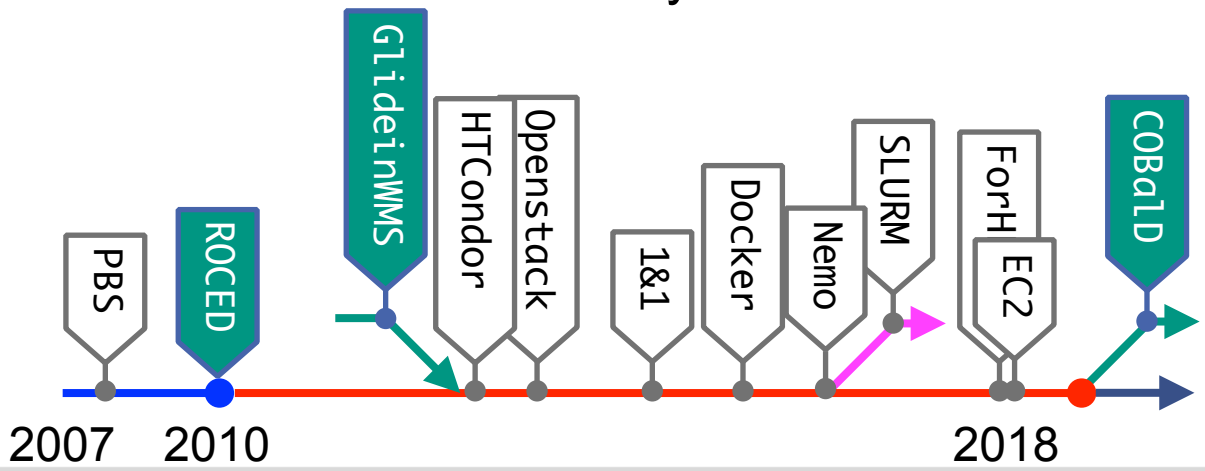
Our Experience in a Nutshell

- Need to look beyond fetching resource A, B, … for job 1, 2, …
  - Integrating resources is a solved problem
  - Micromanaging does not work/scale well
- Opportunistic resources mean generalisation of Pilot model
  - Volatile resources (pilot, container, VM) from providers (Grid, HPC, cloud, …)
  - Overlay resource broker (HTCondor, U/SGE, Torque, Slurm, …) for users



Our Experience in a Nutshell

- Need to look beyond fetching resource A, B, … for job 1, 2, …
  - Integrating resources is a solved problem
  - Micromanaging does not work/scale well
- Opportunistic resources mean generalisation of Pilot model
  - Volatile resources (pilot, container, VM) from providers (Grid, HPC, cloud, …)
  - Overlay resource broker (HTCondor, U/SGE, Torque, Slurm, …) for users
- No silver bullet for opportunistic resources yet (that includes ROCED)
  - Demand exists – ROCED used by other institutes as well



Opportunistic Processing: ROCED (2010-present)

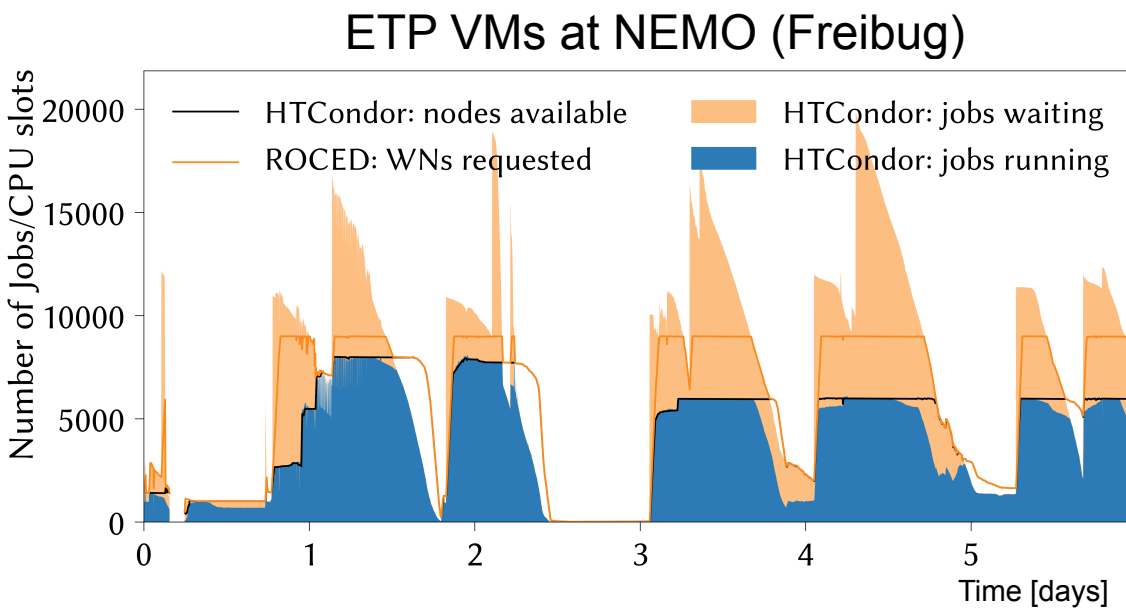
[**R**esponsive **O**n-**D**emand **C**loud **E**nabled **D**eployment]

- Dynamically extend batch systems with cloud resources
  - Monitor queued user jobs to predict required resources
  - Boot VMs as worker nodes to match predicted resources

Opportunistic Processing: ROCED (2010-present)

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

- Dynamically extend batch systems with cloud resources
  - Monitor queued user jobs to predict required resources
  - Boot VMs as worker nodes to match predicted resources

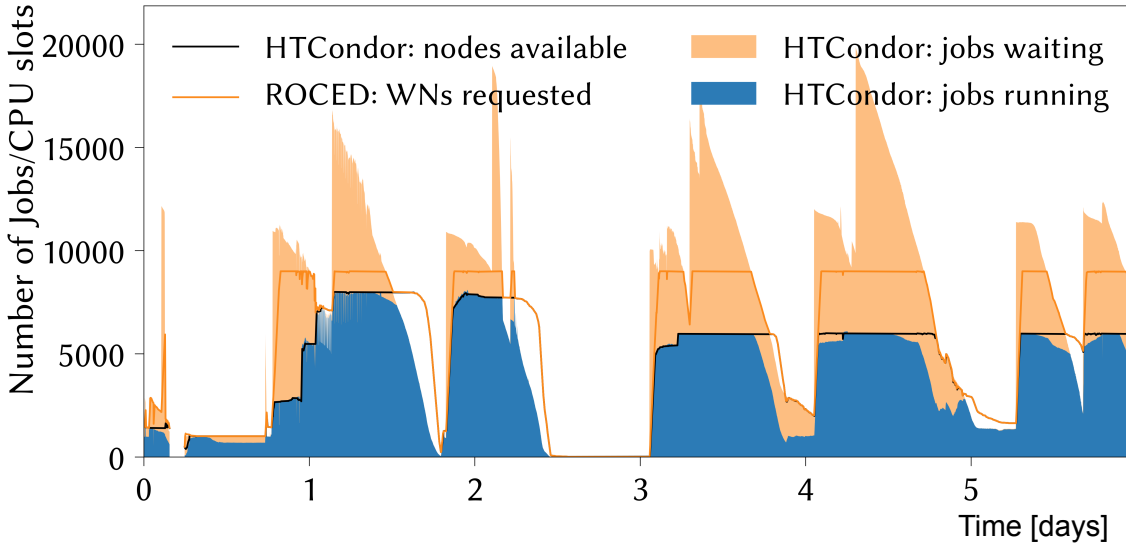


Opportunistic Processing: ROCED (2010-present)

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

- Dynamically extend batch systems with cloud resources
  - Monitor queued user jobs to predict required resources
  - Boot VMs as worker nodes to match predicted resources
- Integrating many diverse resource is feasible
  - Reusable / generic core logic with high software quality
  - Adding resource types is a matter of manpower and time

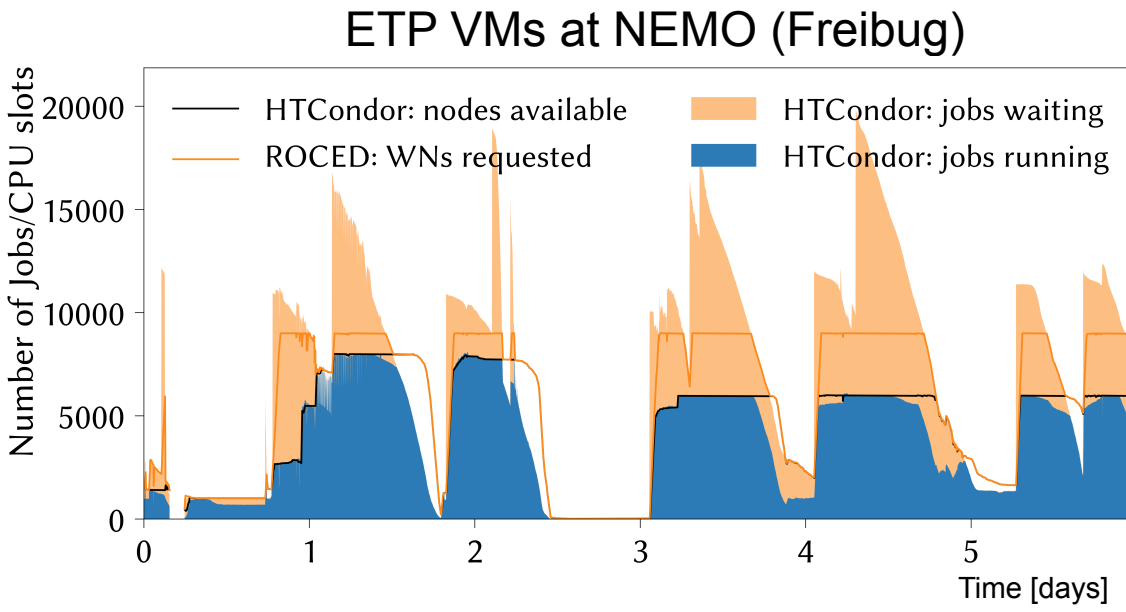
ETP VMs at NEMO (Freibug)



Opportunistic Processing: ROCED (2010-present)

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

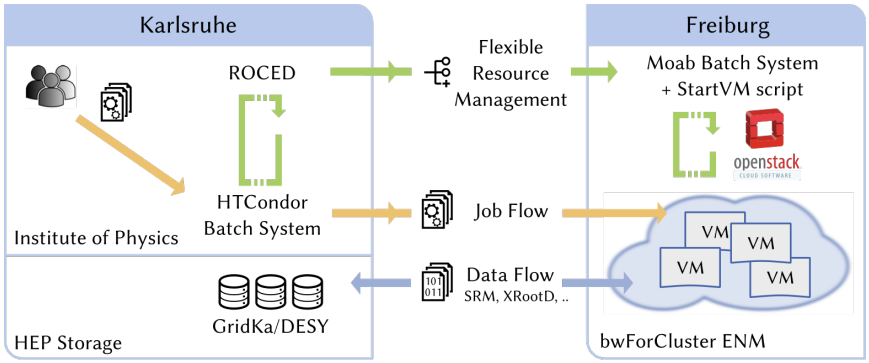
- Dynamically extend batch systems with cloud resources
  - Monitor queued user jobs to predict required resources
  - Boot VMs as worker nodes to match predicted resources
- Integrating many diverse resource is feasible
  - Reusable / generic core logic with high software quality
  - Adding resource types is a matter of manpower and time
- Managing many diverse resource types is hard!
  - Problems of scale, latency, non-linearity
  - Micromanagement does not work!

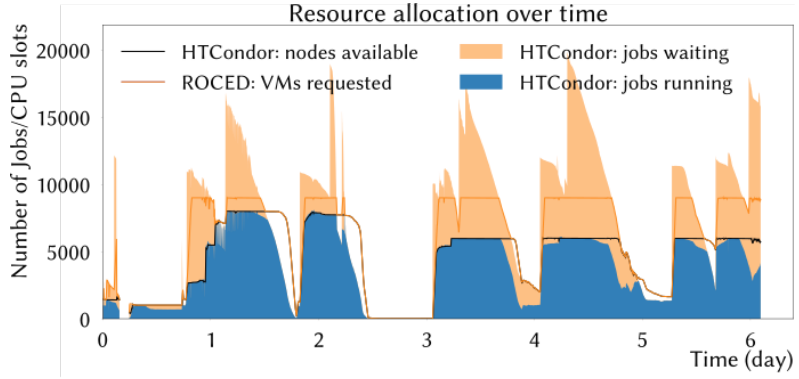


Success Story - Opportunistic “Tier 1“ for a Day

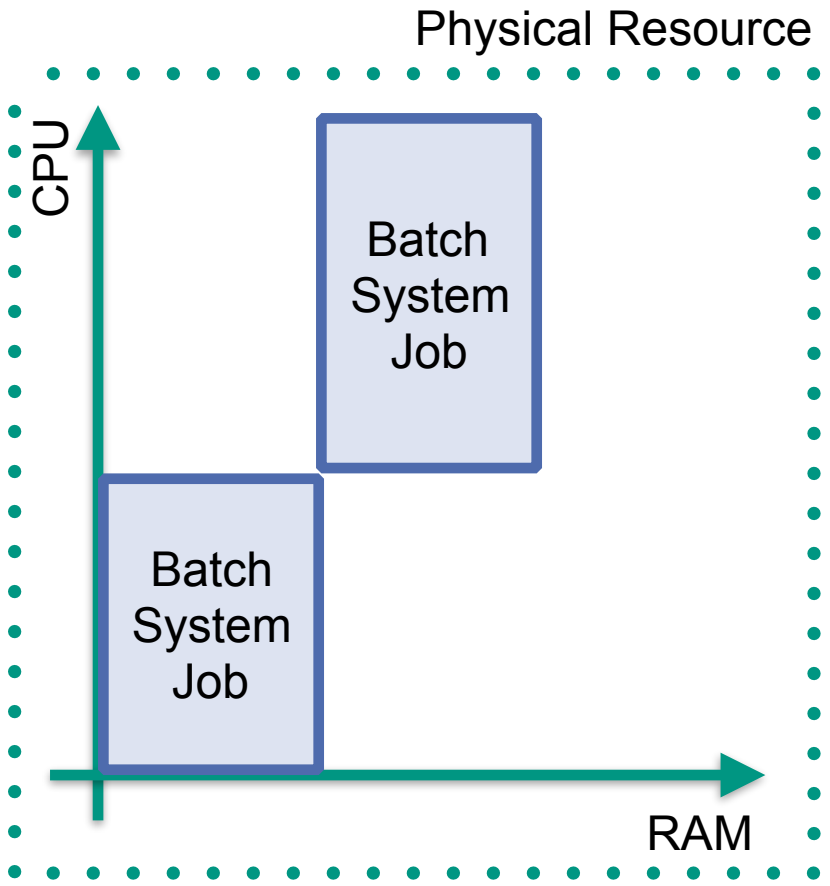
- Dynamically shared HPC Centre at Freiburg (three diverse communities)
- Virtualization is key component to:
  - Allow dynamic resource partitioning
  - Meet OS & software requirements
- ROCED cloud scheduler developed at KIT
  - On-demand resource provisioning
  - Transparent resource integration
-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

[ACAT17MS, JOP898TH, JOP762TH, JOP664TH]

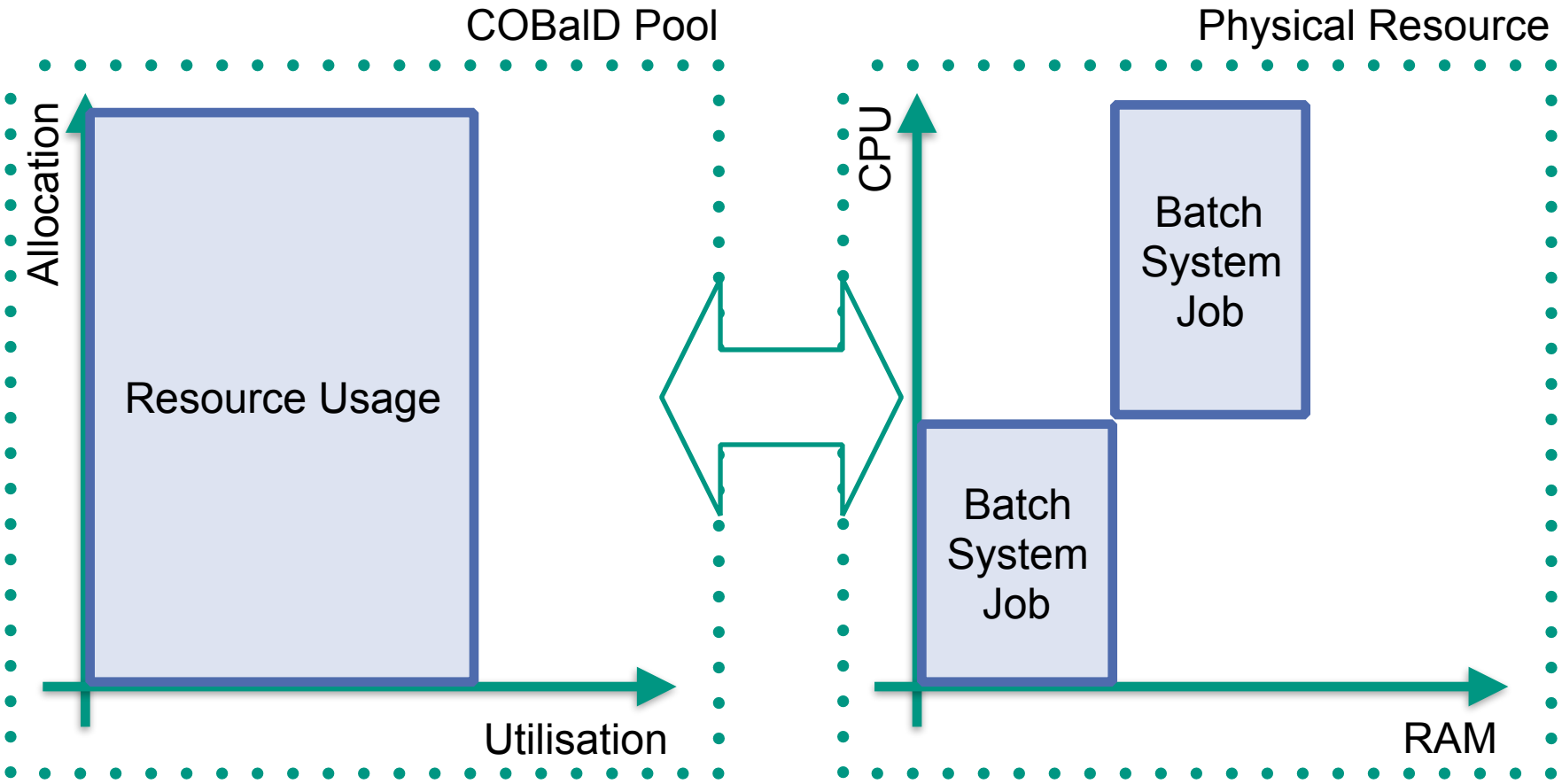




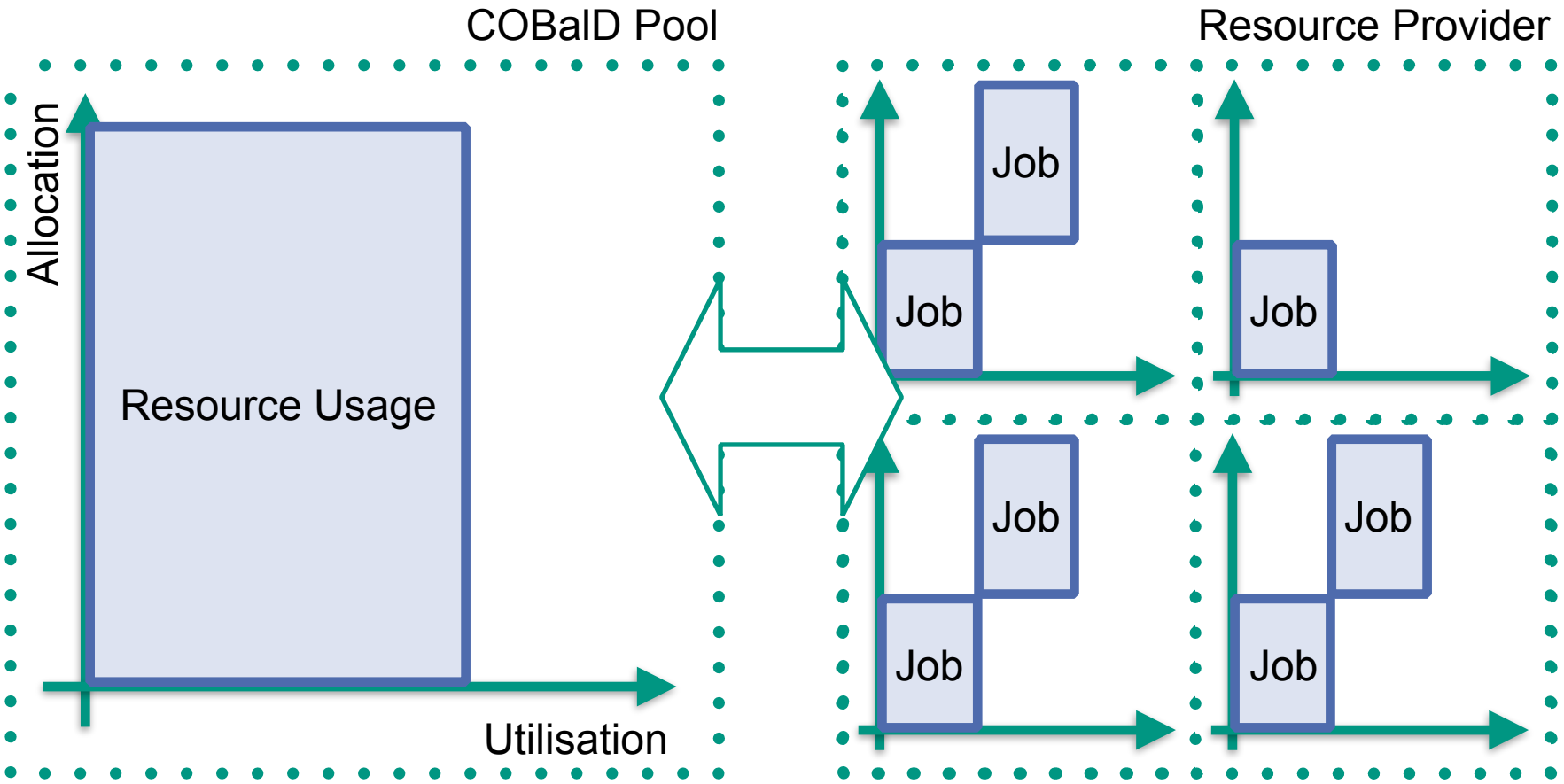
COBalD Resource Pool Model



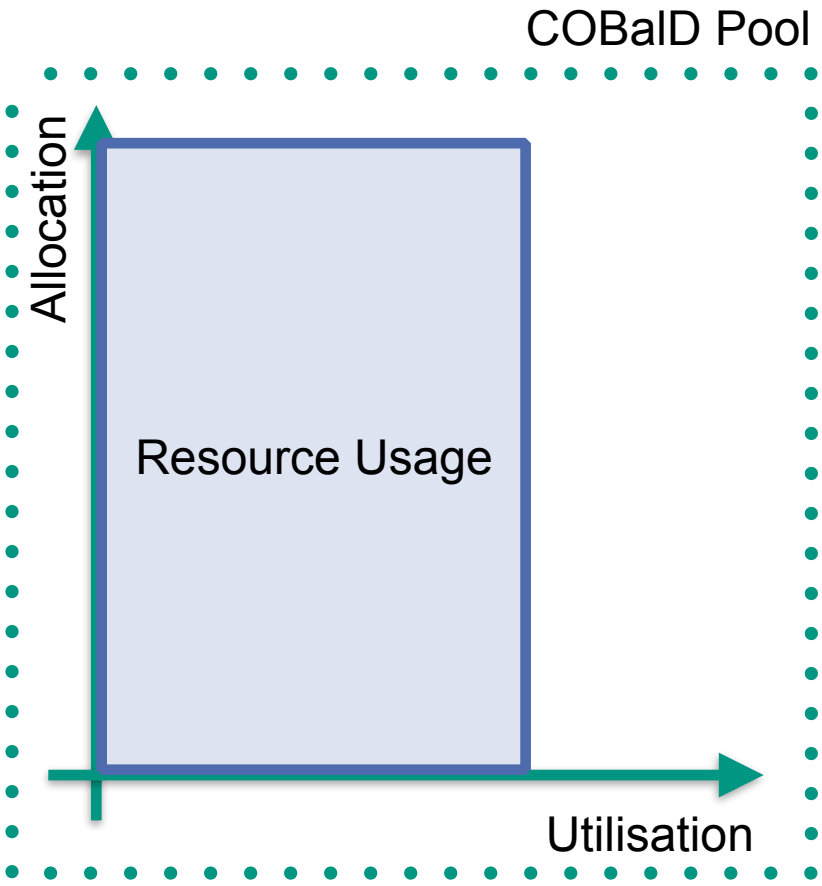
COBalD Resource Pool Model



COBalD Resource Pool Model

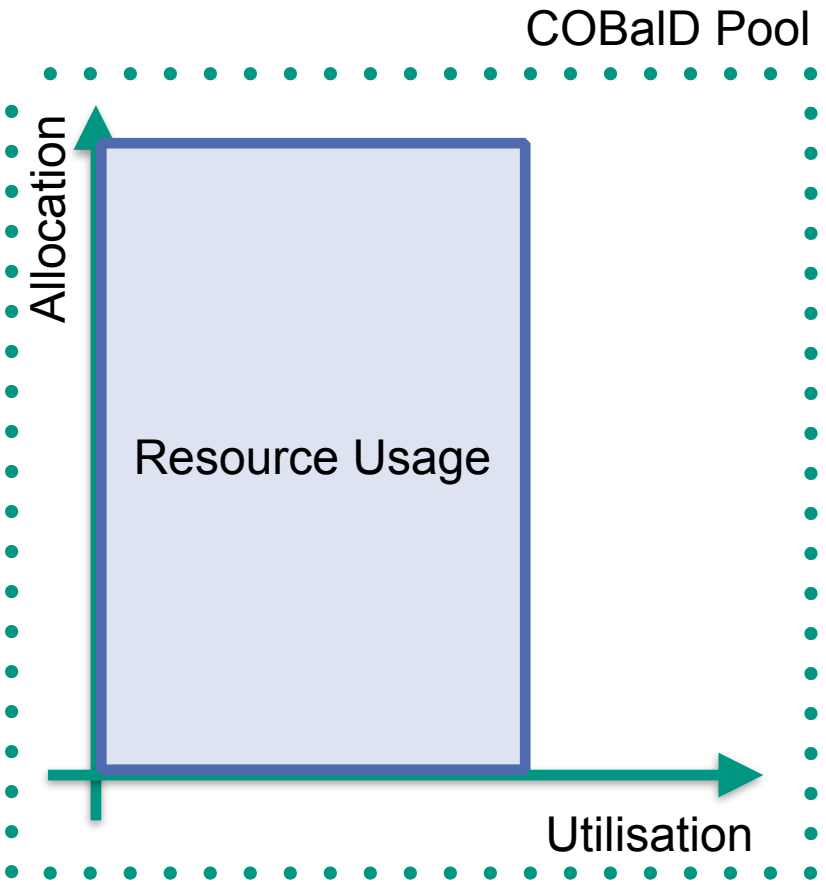


COBalD Resource Pool Model



COBalD Resource Pool Model

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



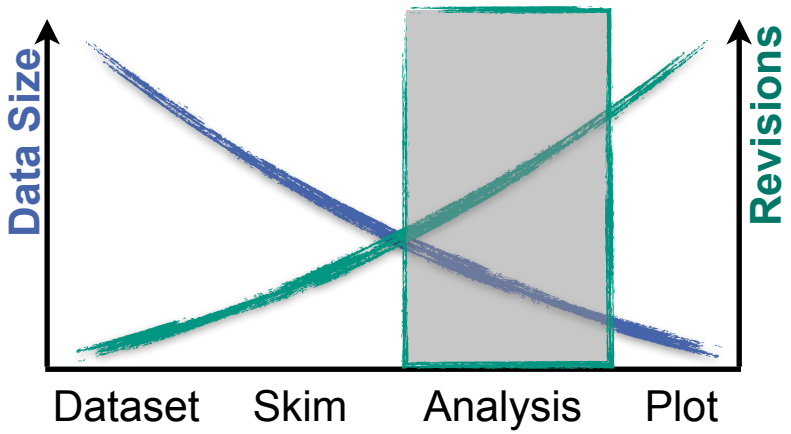
Karlsruher Institut für Technologie

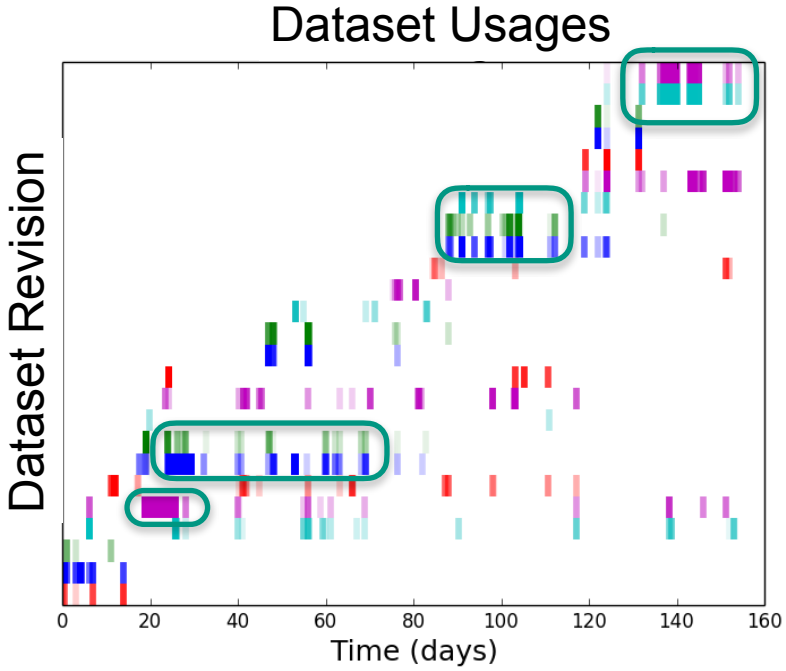
Backup Caching

Steinbuch Centre for Computing / Institute for Experimental Particle Physics

Context: HEP End User Data Analysis

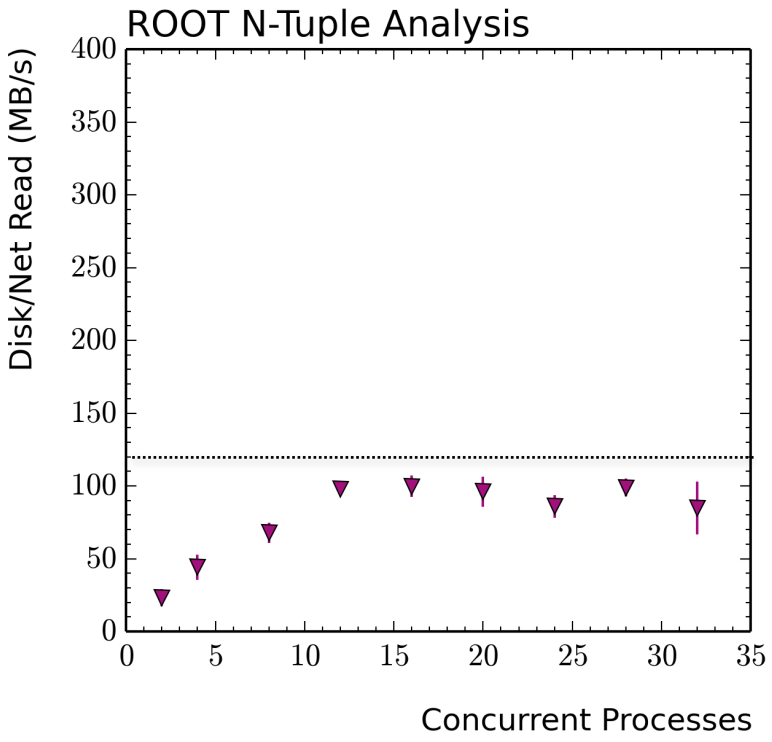
- Hierarchical, iterative workflows
  - Reduction of data size
  - Increase of iterations
  - Dedicated processing environments
- Data intense analyses on Tier 3
  - Local batch system
  - Network accessed fileservers
  - Optimized input data sets
- Usage suitable for caching
  - Repeated processing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

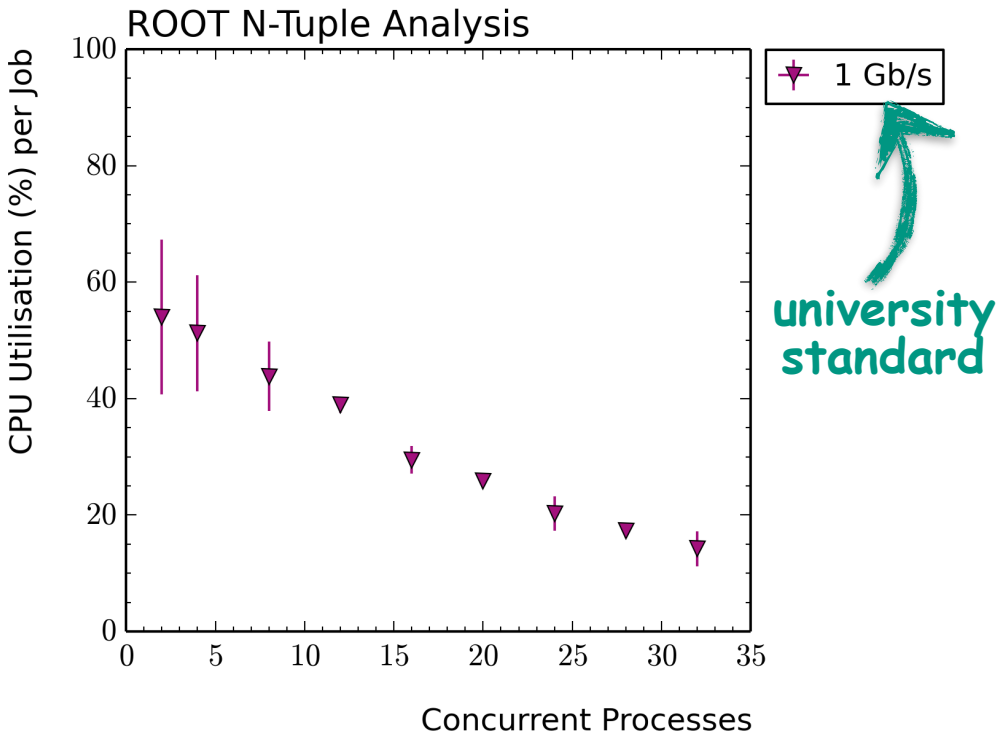




I/O Performance Evaluation

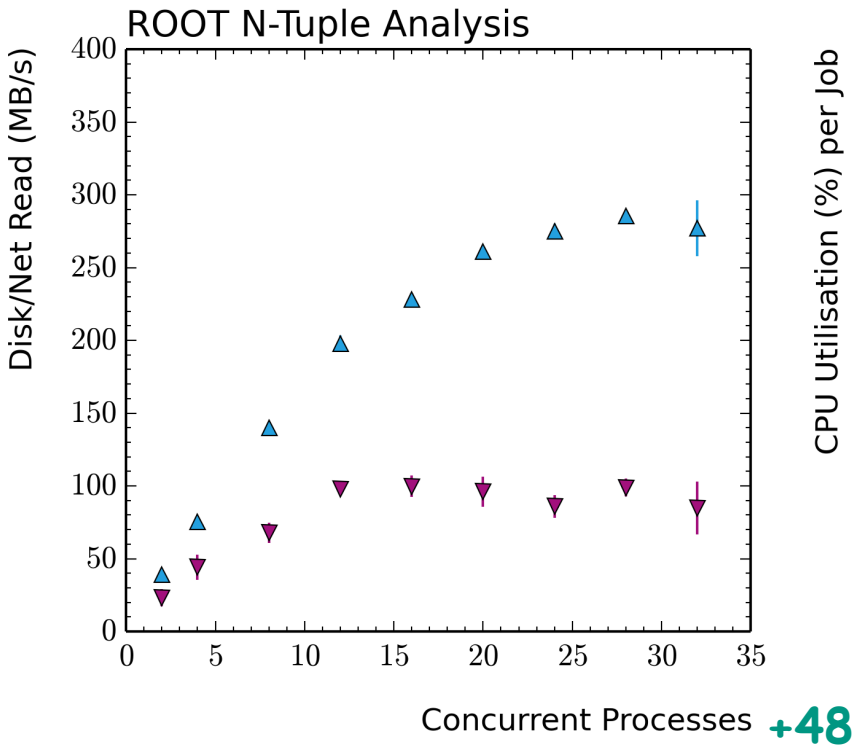
- CMS jet calibration analysis (ROOT n-tuple)

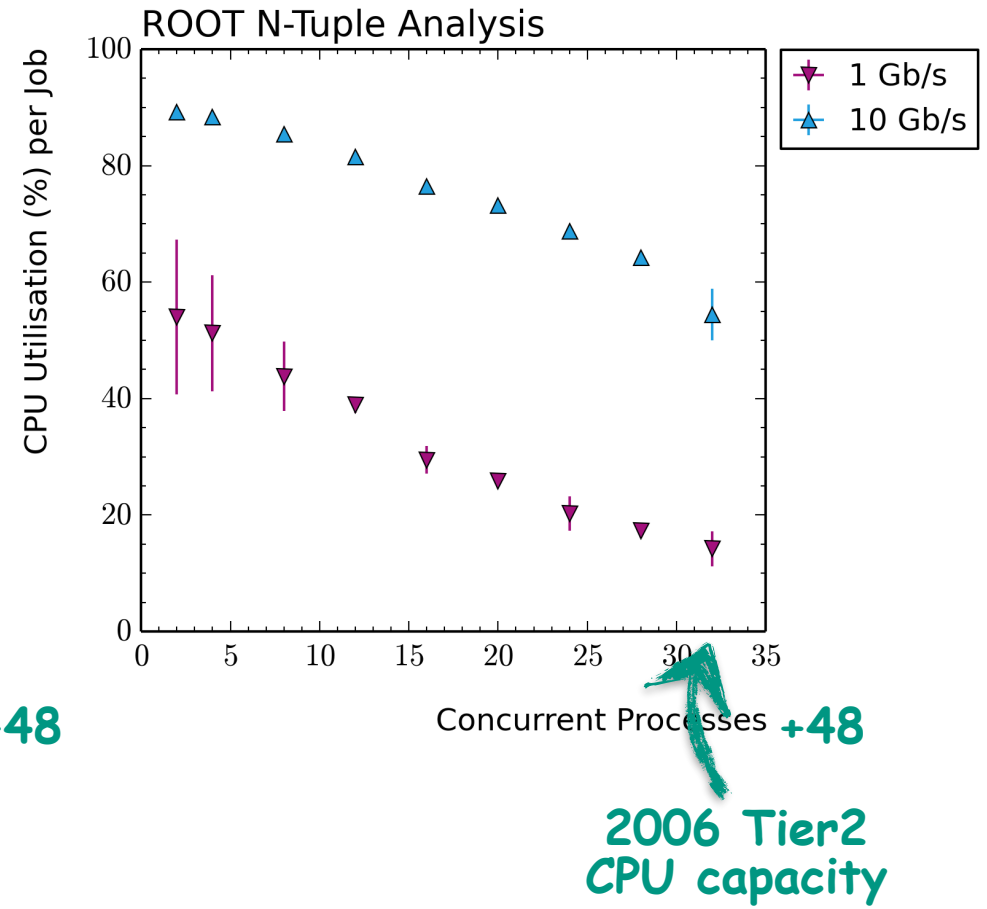




I/O Performance Evaluation

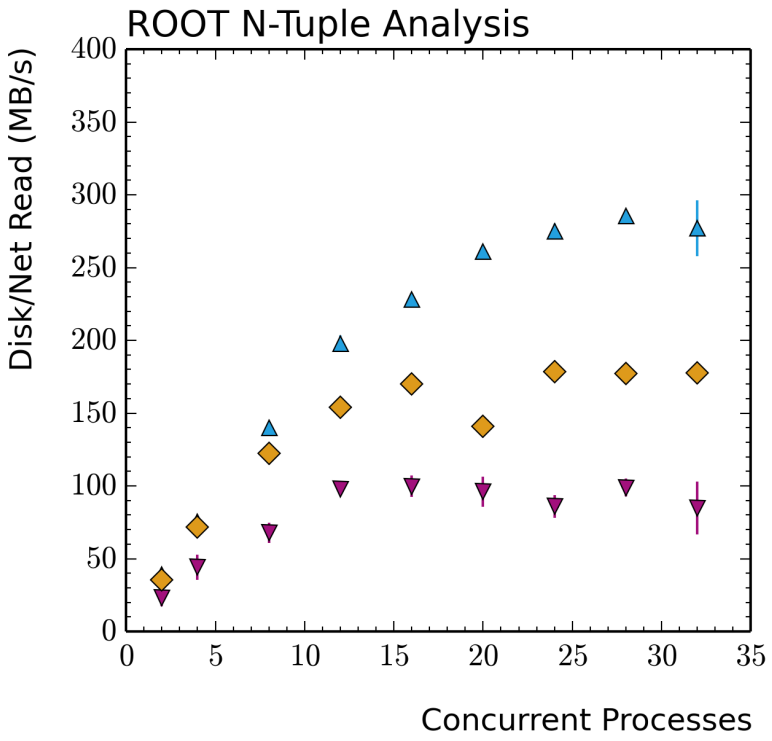
- CMS jet calibration analysis (ROOT n-tuple)
- Additional 48 concurrent reads from other workers for 10 Gb/s test

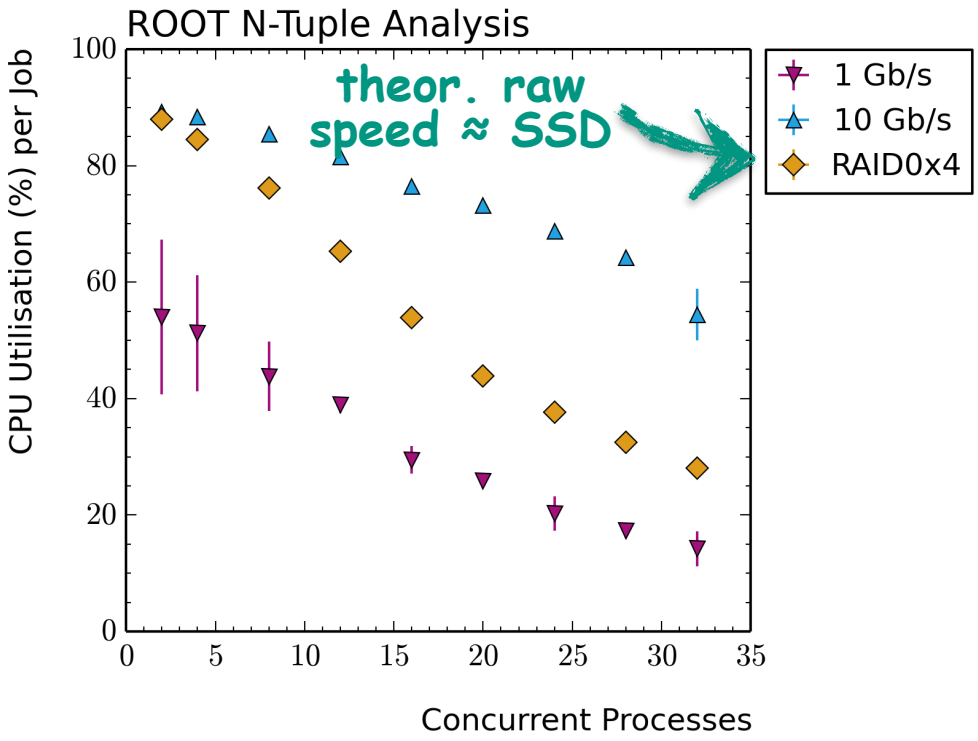




I/O Performance Evaluation

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

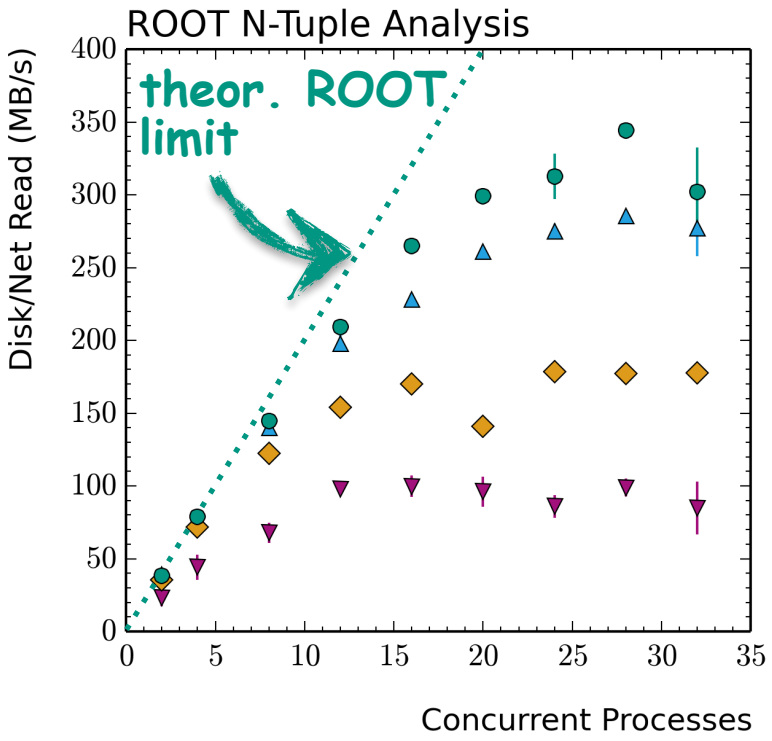


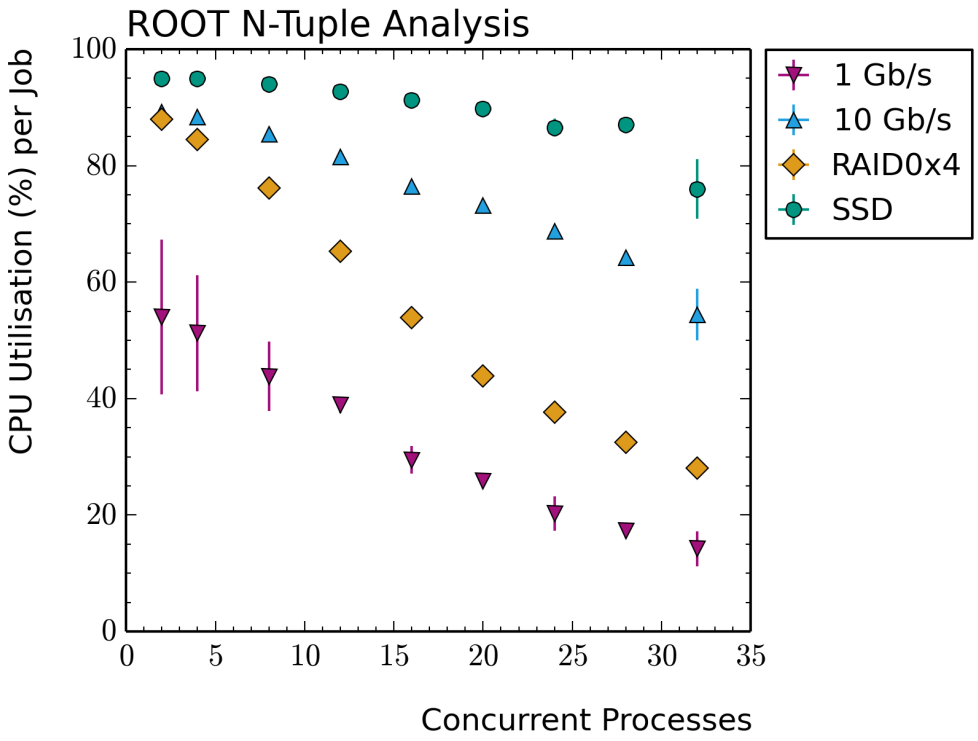


- HDDs limited on concurrent accesses

I/O Performance Evaluation

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

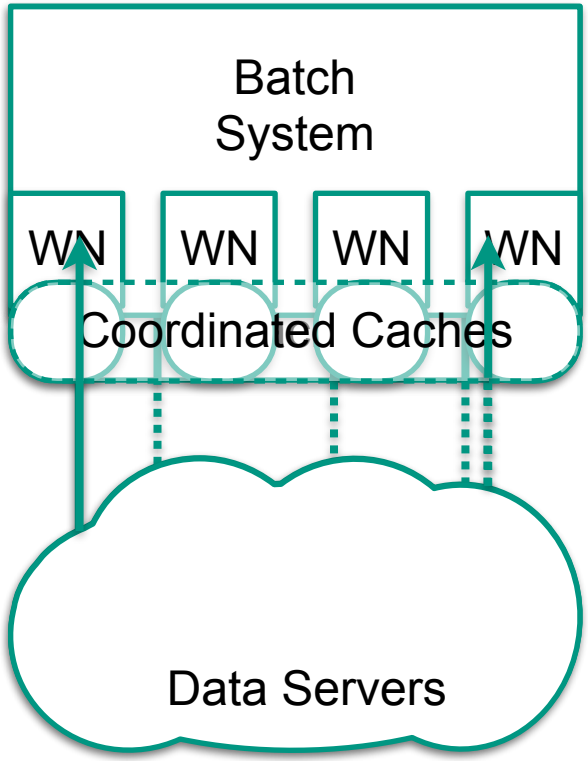




- HDDs limited on concurrent accesses
- SSDs exploit full system capacities

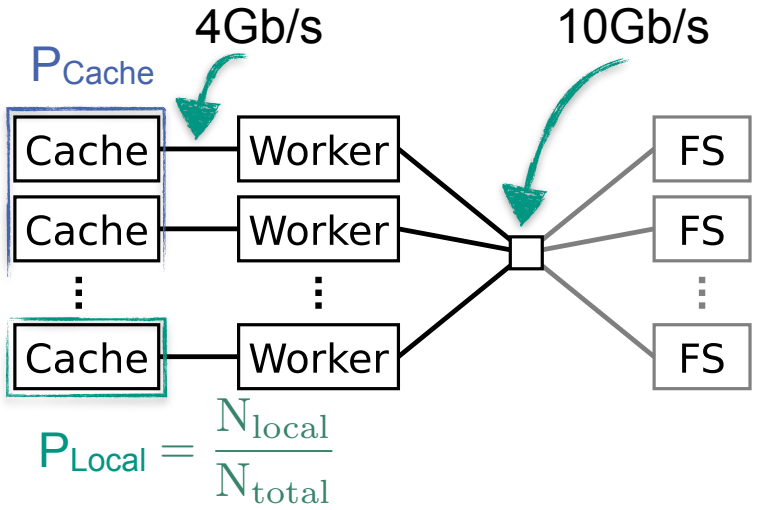
Coordinated Caching: Overview

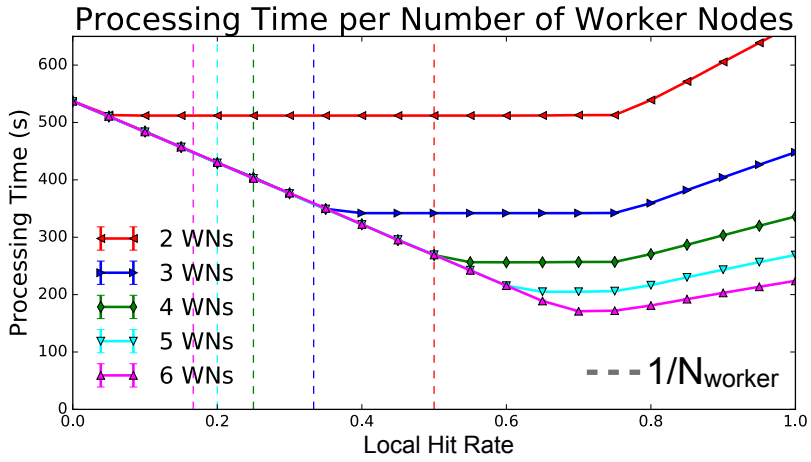
- Caching between batch system and data sources
  - Consumer focused caching
  - Partial data locality for remote files
- Abstracts cache to batch system scale
  - Utilize meta-data of entire user workflows
  - Works on files used by jobs
- Implementation at host granularity
  - Array of individual caches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





Opportunistic Data Locality: Coordinated Caches

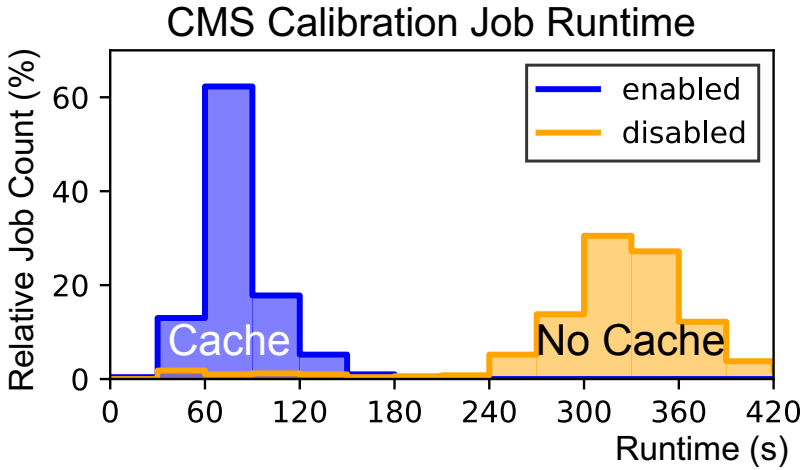
- Challenge: match cache scope to workflow scope
  - Workflows are distributed, but jobs are local
  - Isolated caches lack scope, distributed caches lack locality

Opportunistic Data Locality: Coordinated Caches

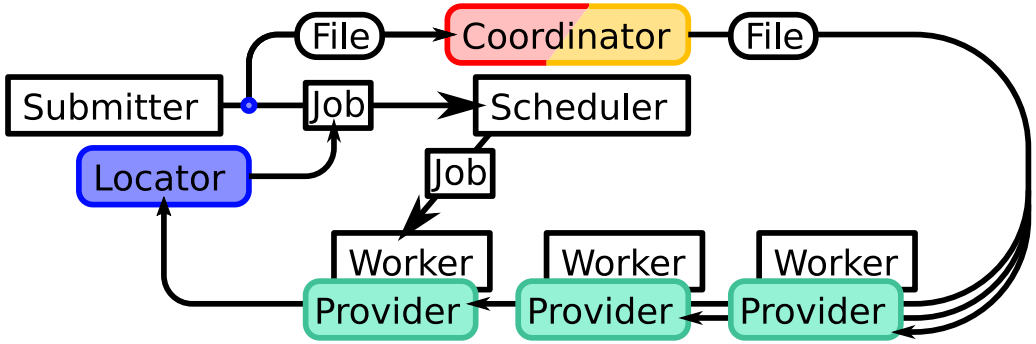
- Challenge: match cache scope to workflow scope
  - Workflows are distributed, but jobs are local
  - Isolated caches lack scope, distributed caches lack locality
- Coordinated Caching: Distributed management of local caches
  - Analyse usage metadata at batch system scope
  - Provide used data at worker node scope
  - Expose locality information for scheduling

Opportunistic Data Locality: Coordinated Caches

- Challenge: match cache scope to workflow scope
  - Workflows are distributed, but jobs are local
  - Isolated caches lack scope, distributed caches lack locality
- Coordinated Caching: Distributed management of local caches
  - Analyse usage metadata at batch system scope
  - Provide used data at worker node scope
  - Expose locality information for scheduling
- Data locality is optional
  - Cache only repeated workflows with high throughput
  - Uncached workflows benefit from uncontested network

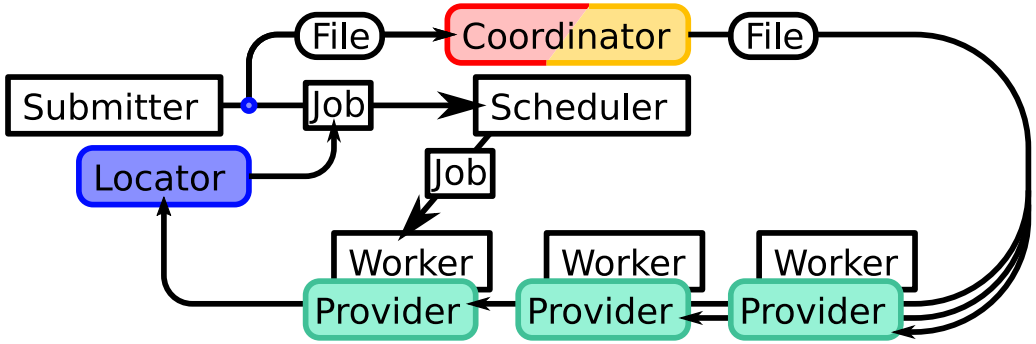


Useful notes: Opportunistic Caching



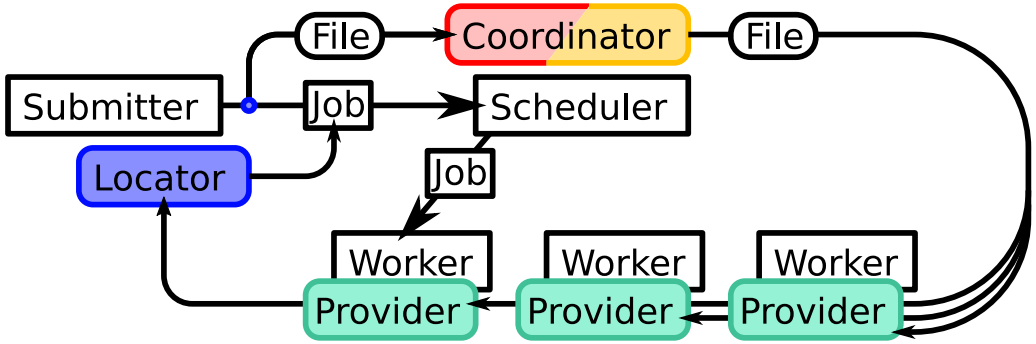
Useful notes: Opportunistic Caching

- Coordinated caches required to match distributed infrastructure
  - Degradation from random access and duplicates (loss$\propto N_{Hosts}$)
  - Individual nodes see only fraction of metadata



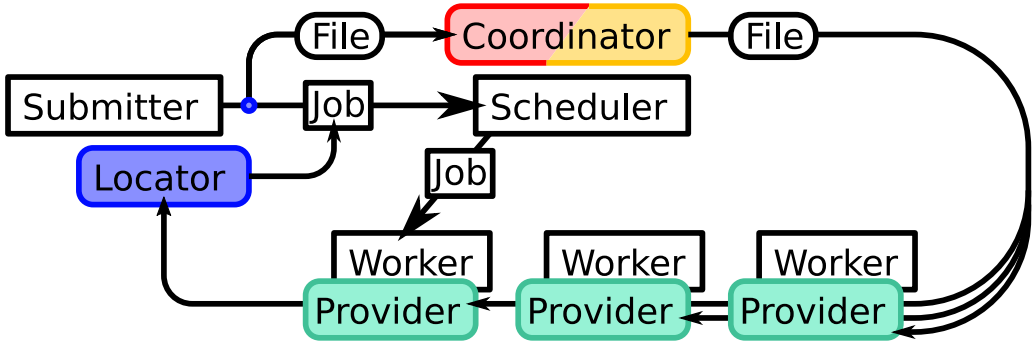
Useful notes: Opportunistic Caching

- Coordinated caches required to match distributed infrastructure
  - Degradation from random access and duplicates (loss$\propto N_{Hosts}$)
  - Individual nodes see only fraction of metadata
- Perfect cache hit rates are undesirable
  - Network still offers considerable bandwidth
  - Caching more data than needed wastes cache space



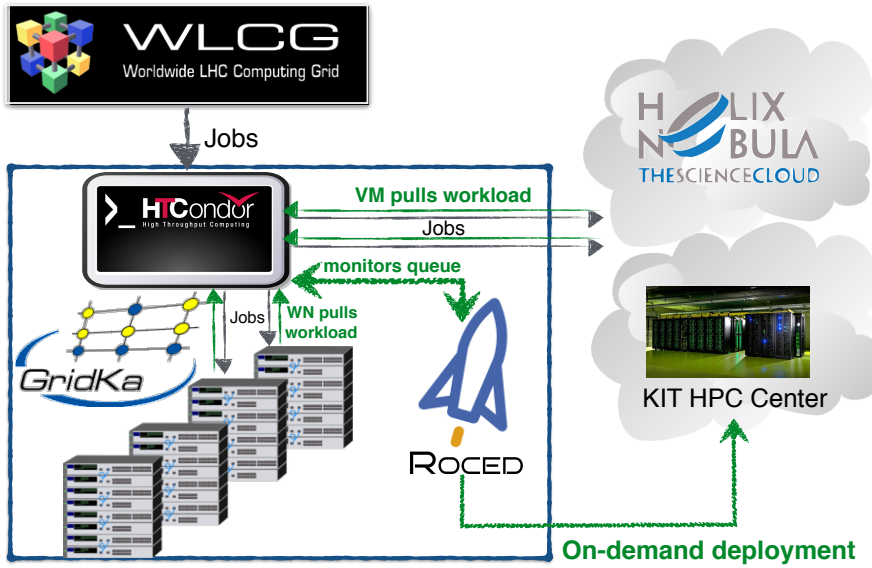
Useful notes: Opportunistic Caching

- Coordinated caches required to match distributed infrastructure
  - Degradation from random access and duplicates (loss$\propto N_{Hosts}$)
  - Individual nodes see only fraction of metadata
- Perfect cache hit rates are undesirable
  - Network still offers considerable bandwidth
  - Caching more data than needed wastes cache space
- Data size requires specialised caching algorithms
  - Avoid trashing from access cycles of days to weeks
  - Only a fraction of workflows benefits from caching



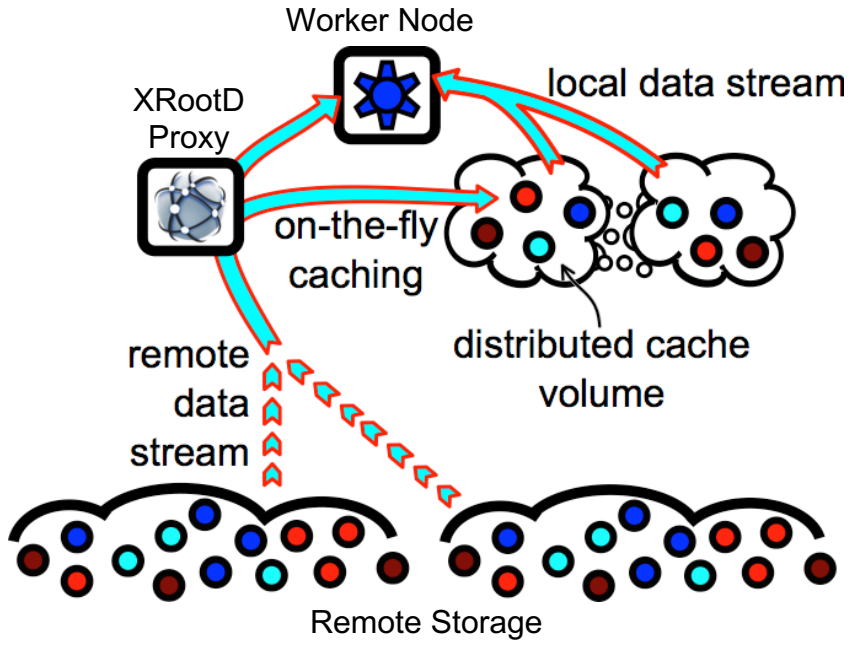
Dynamic Compute Expansion of GridKa Tier 1

- 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]