Karlsruhe Institute of Technology

Opportunistic data locality for end user data analysis

Max Fischer, Eileen Kühn, Christoph Heidecker, Matthias Schnepf, Manuel Giffels

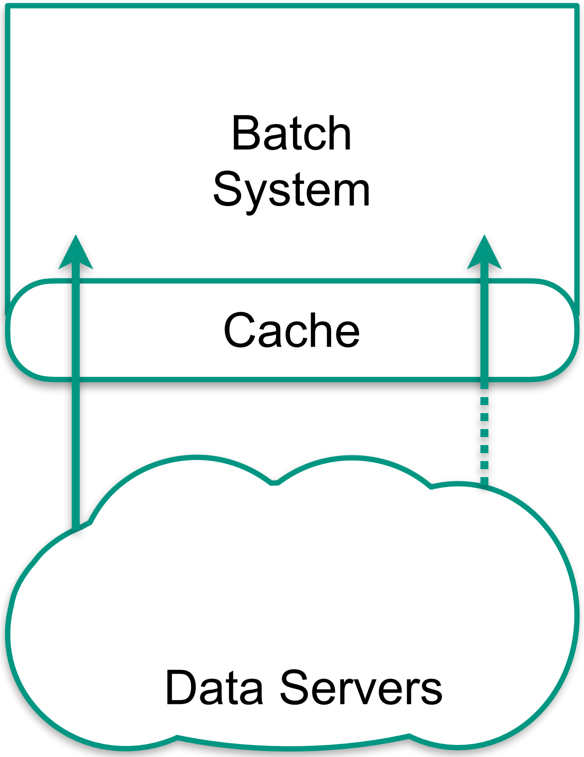
CHEP 2016

Steinbuch Centre for Computing & Institute for Experimental Particle Physics

Image By Matt Howard [CC BY-SA 2.0 (http://creativecommons.org/licenses/by-sa/2.0)], via Wikimedia Commons

Coordinated Caching: Overview

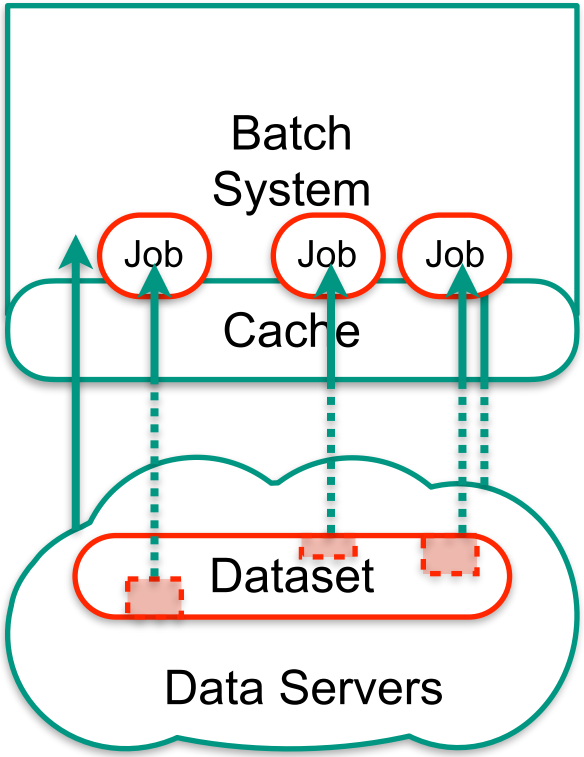
- Cache between batch system and data sources
  - Consumer focused caching
  - Provides partial data locality



Coordinated Caching: Overview

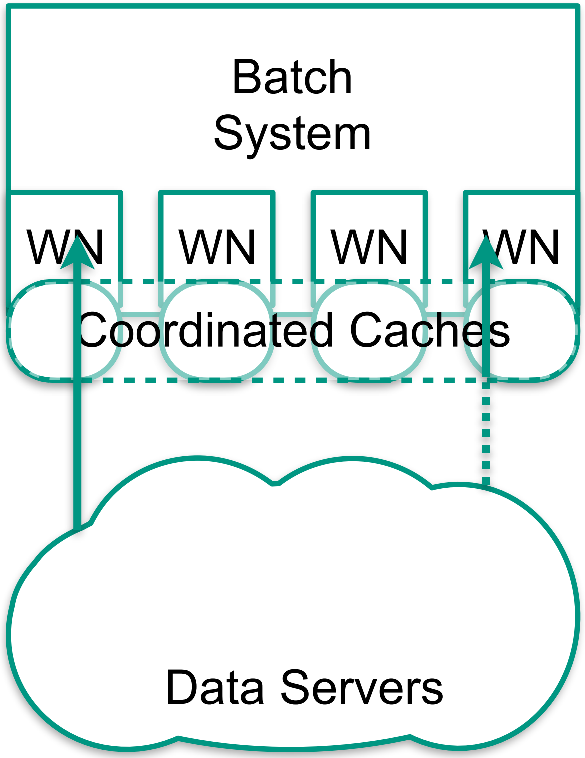
- Cache between batch system and data sources
  - Consumer focused caching
  - Provides partial data locality

- Abstracts cache to batch system scale
  - Utilize meta-data of entire user workflows
  - Works on files used by jobs
  - Aware of hosts and locality



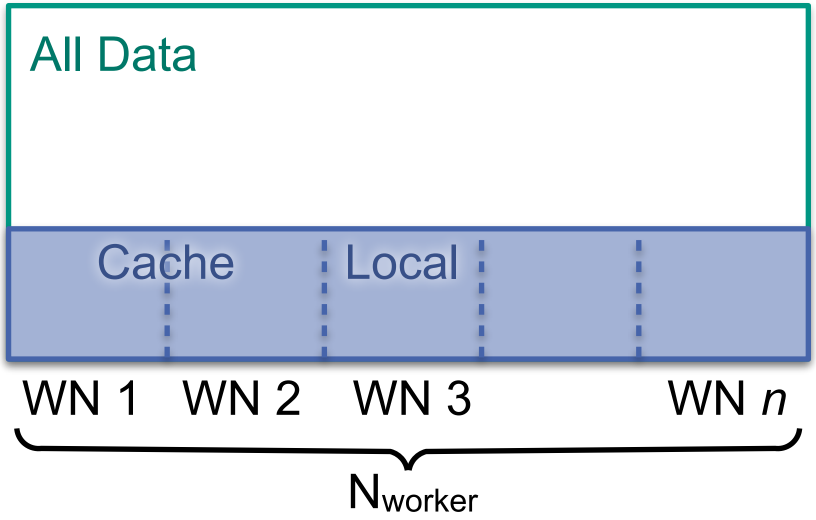
Coordinated Caching: Overview

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



Coordinated Caching: Data Scheduling

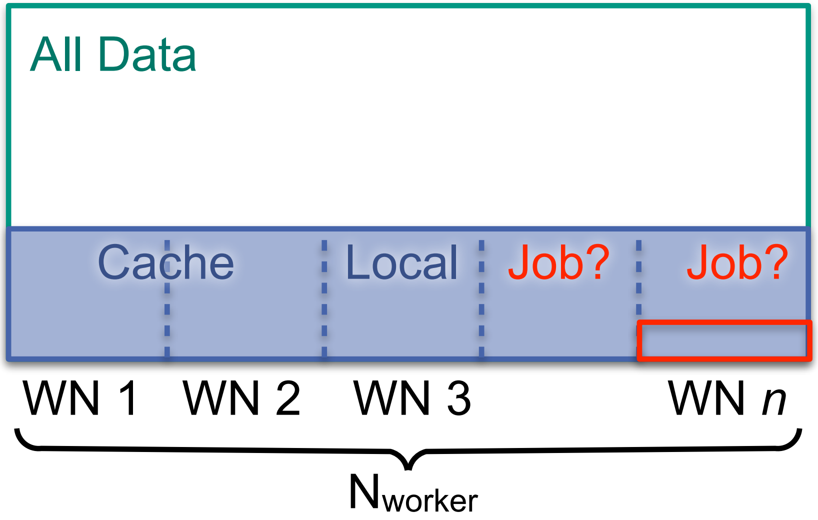
- Distributed caching complicates cache access
  - Data cached anywhere (cache hit rate)
  - Data local on job host (local hit rate)



Coordinated Caching: Data Scheduling

- Distributed caching complicates cache access
  - Data cached anywhere (cache hit rate)
  - Data local on job host (local hit rate)

- Schedule Jobs to input data location
  - Unscheduled hit rate limited to ~1/N_{worker}
  - Data location published to batch system

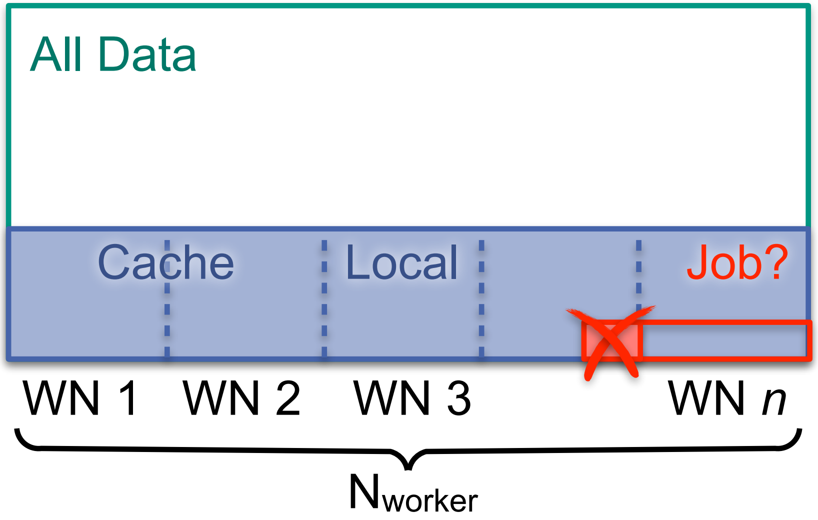


Coordinated Caching: Data Scheduling

- Distributed caching complicates cache access
  - Data cached anywhere (cache hit rate)
  - Data local on job host (local hit rate)

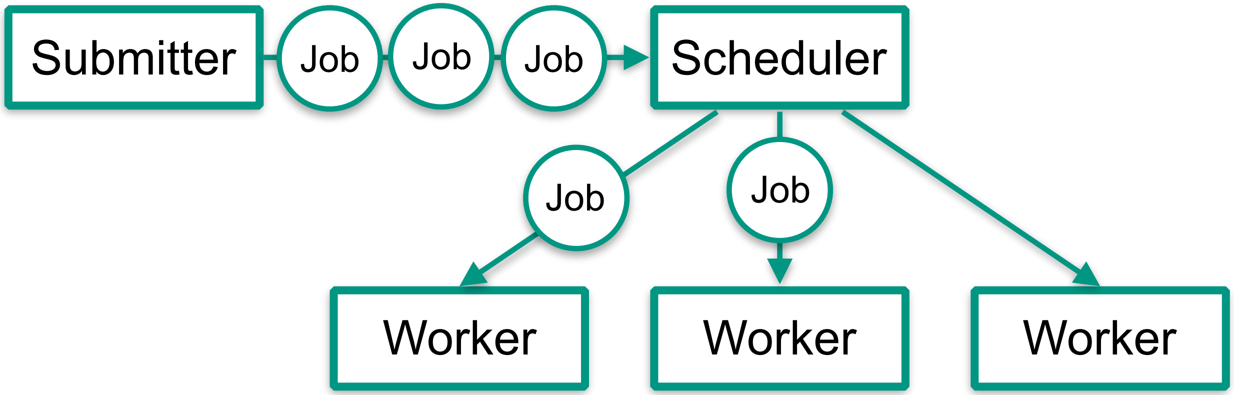
- Schedule Jobs to input data location
  - Unscheduled hit rate limited to ~1/N_{worker}
  - Data location published to batch system

- Place data to match workflows
  - Jobs require groups of files
  - Data placement replicates observed file groups



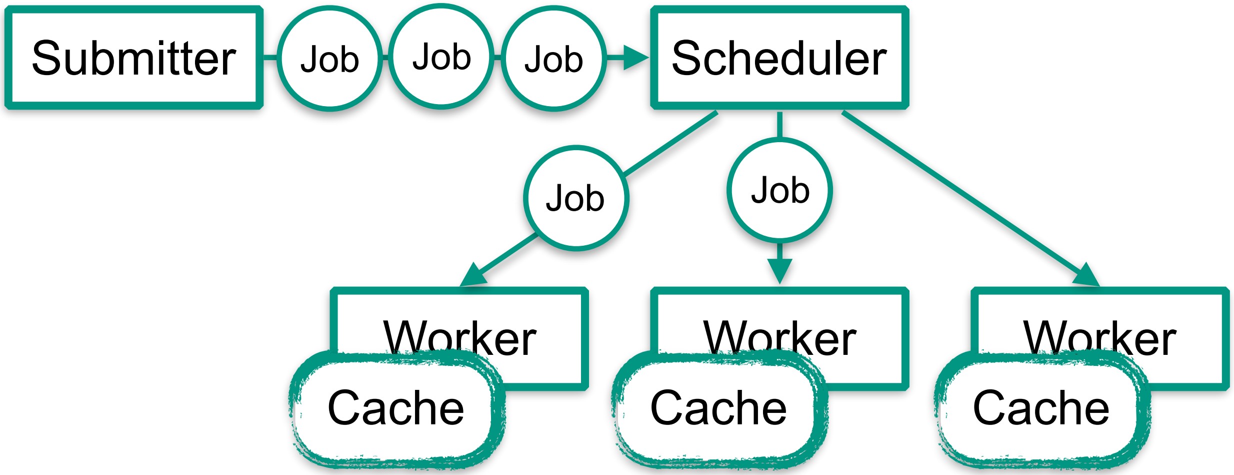
HTDA Batch System Extension

High Throughput Data Analysis



HTDA Batch System Extension

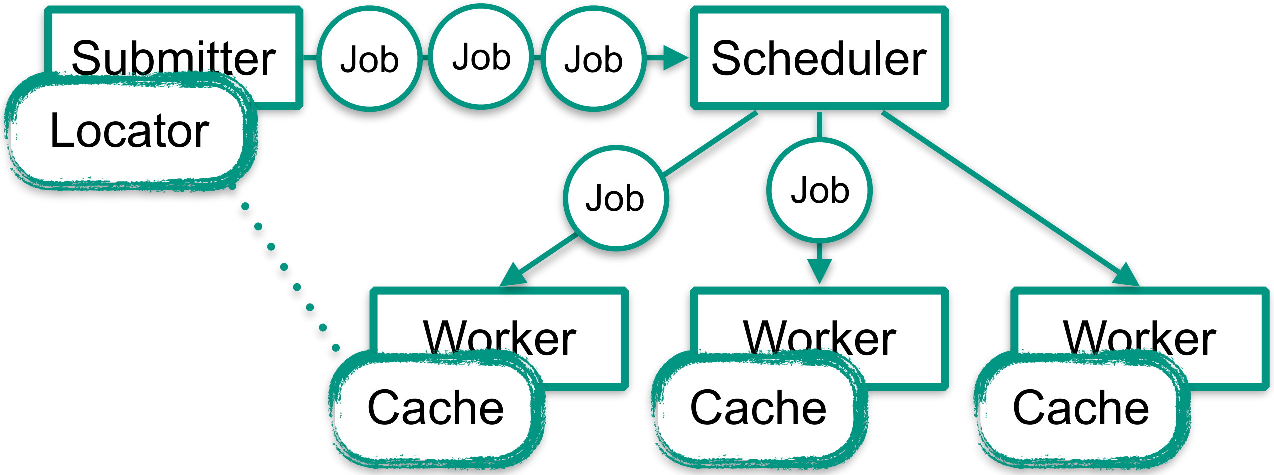
High Throughput Data Analysis



- Caches maintain data copies on worker nodes

HTDA Batch System Extension

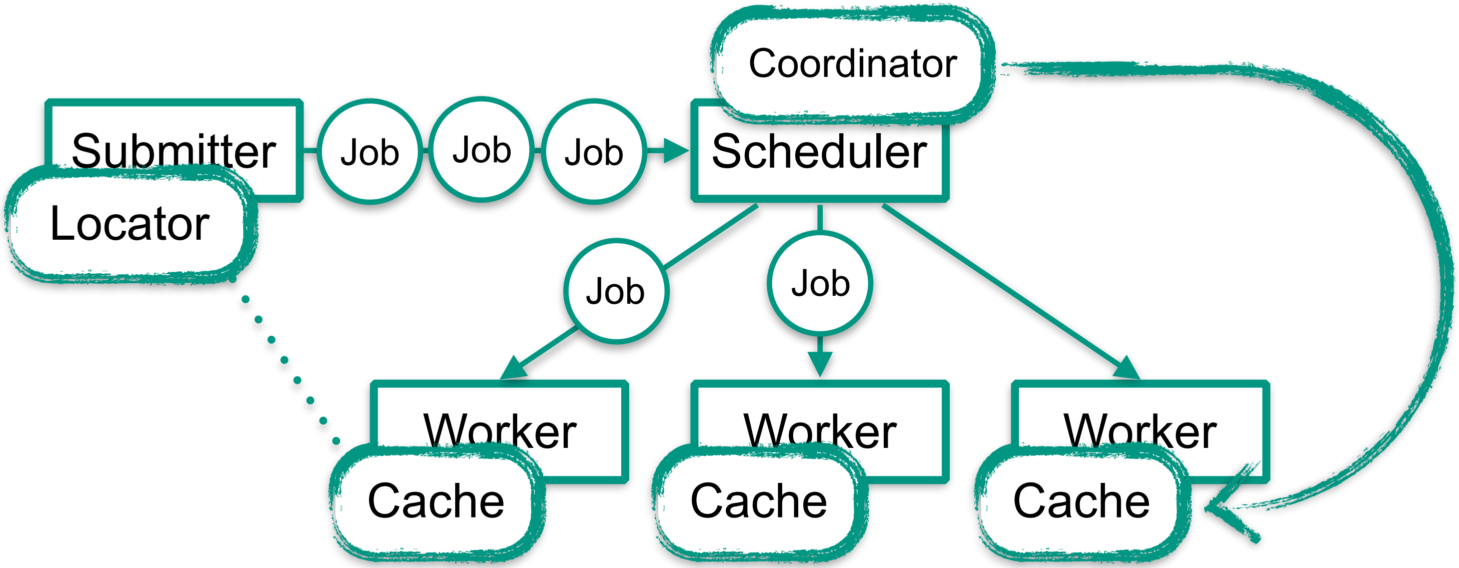
High Throughput Data Analysis



- Caches maintain data copies on worker nodes
- Locator provides locality information for jobs

HTDA Batch System Extension

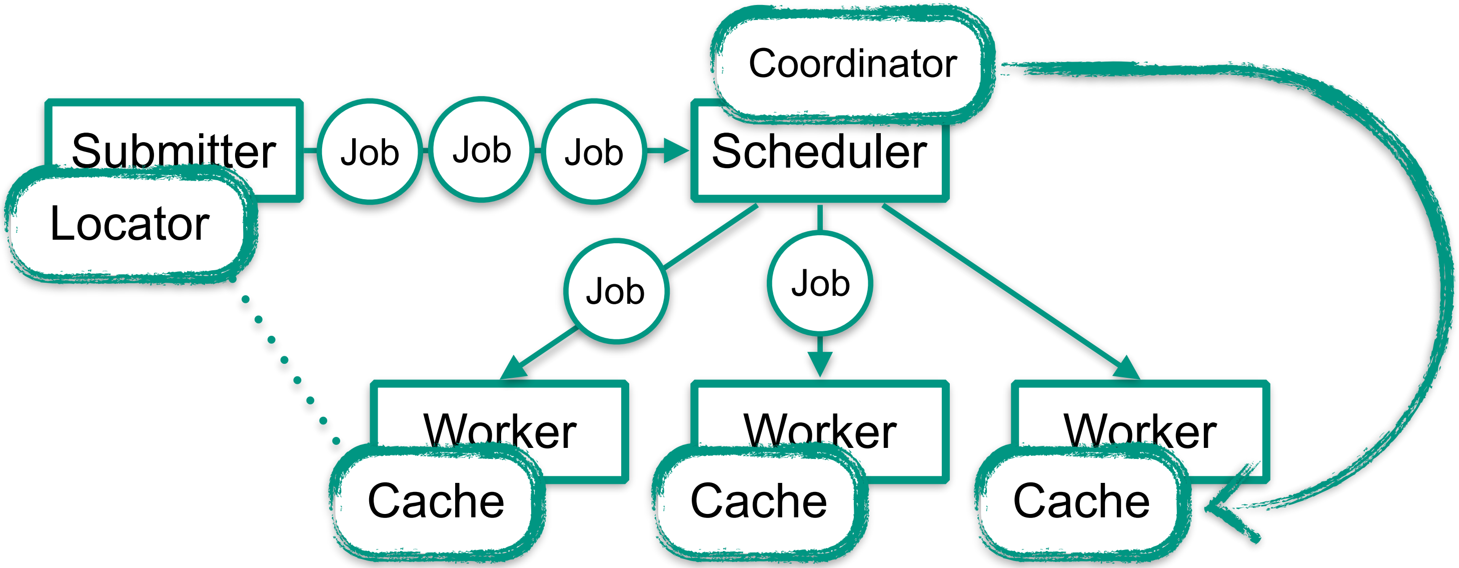
High Throughput Data Analysis



- Caches maintain data copies on worker nodes
- Locator provides locality information for jobs
- Coordinator schedules files for caching on nodes

HTDA Batch System Extension

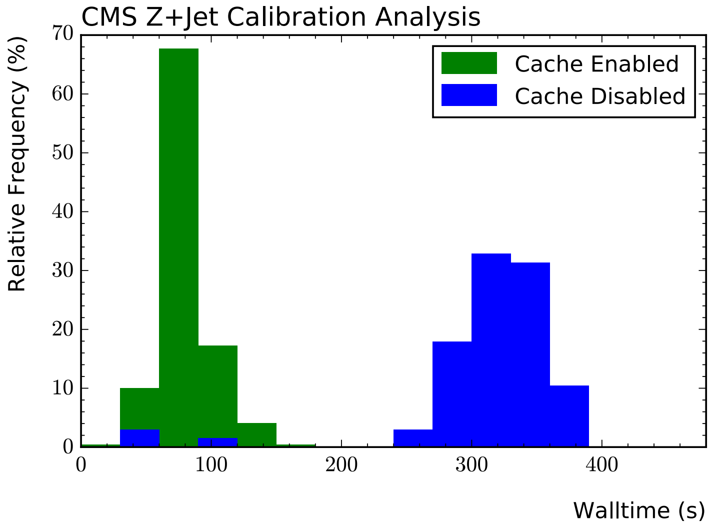
High Throughput Data Analysis



- Caches maintain data copies on worker nodes
- Locator provides locality information for jobs
- Coordinator schedules files for caching on nodes
- Repository: https://bitbucket.org/kitcmscomputing/hpda

Things that worked well

- Responsibilities distributed in cluster
  - File selection, job scheduling, data provisioning
  - Metadata exposed, exchanged between components
  - Locality available for scheduling

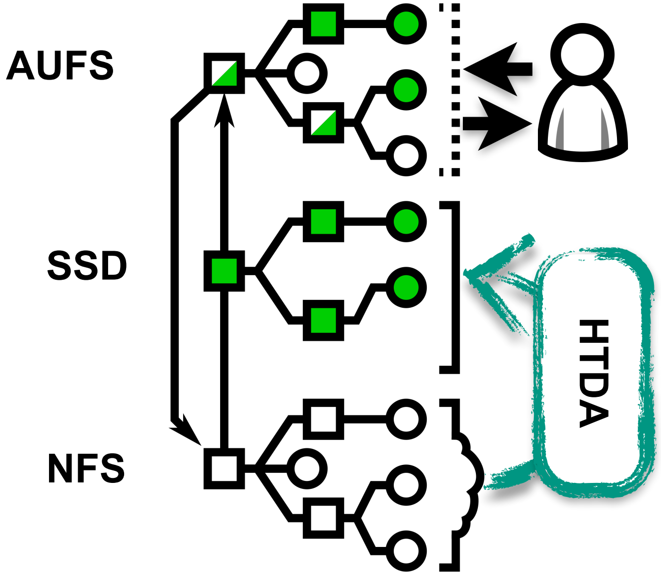


- Simple, heuristic scheduling
  - Jobs preferably scheduled to data
  - Cache-Misses acceptable:
notable throughput via network

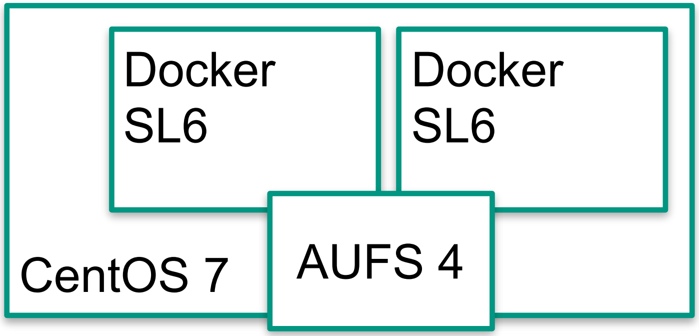
- Core service handles file placement only
  - Job routing/tracking via batch system plugin
  - Data access at protocol level

Caching as a Service via Docker

- POSIX access for local Batch System
  - Local SSDs in front of NFS servers
  - AUFS merges cache onto storage



- Major stability issues on SL5/6
  - AUFS 2.X only (~2010)
  - Kernel deadlock after ~30min
- Dependencies handled via Docker/HTCondor
  - Host system with kernel 4.X
  - Cache services at recent versions
  - Jobs run in SL6 Container, access CentOS7 services



Caching in the Cloud

- Caching enables Cloud data analysis
  - Coordinated caching provides data automatically
  - Data access with no/little external traffic
  - Volatility of caches matches volatility of VMs

- Prototype setup using xrootd access
  - Proxy xrootd server shadows remote server
  - Jobs read data via VM- or Site-local proxy
  - Data selected with knowledge from non-Cloud WNs

- Room for optimizations
  - Overlap of xrootd & HTDA capabilities
  - Heuristic scheduling unsuitable for Clouds

Summary

- Coordinated Caches for Batch Systems: HTDA
    - Array of caches on worker nodes
    - Coordination by global service
    - Targets input files of user workflows

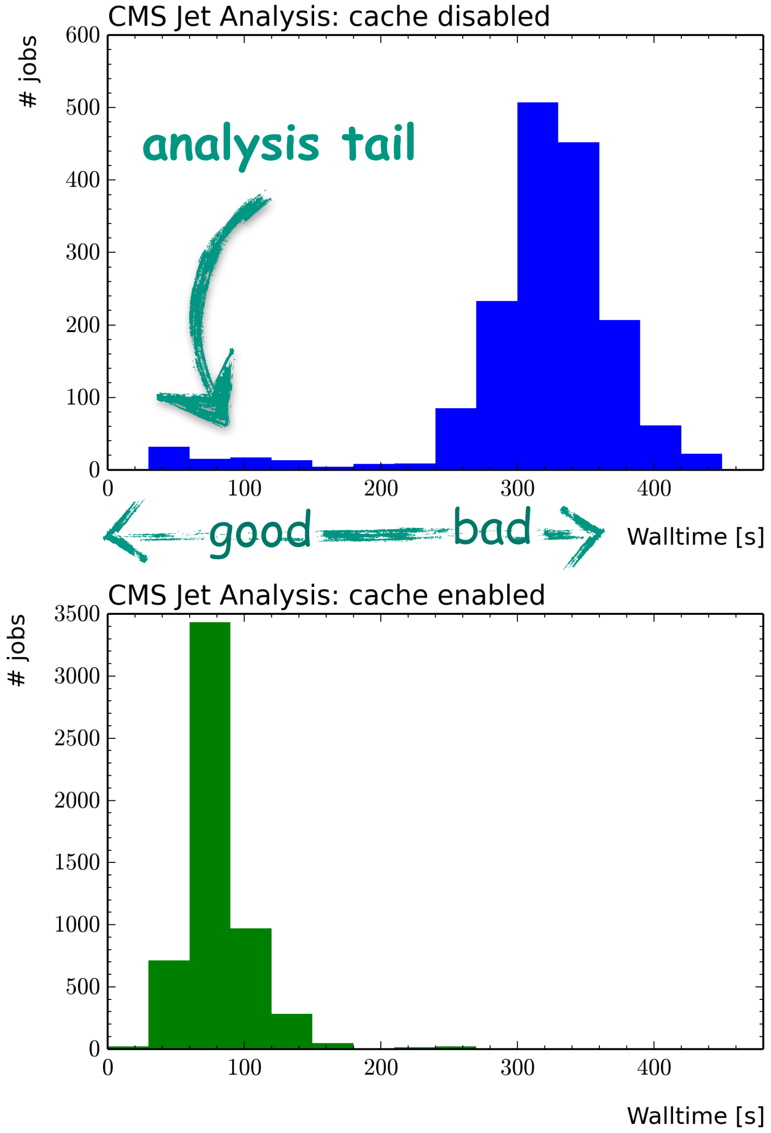
- Caching service provided via Docker
    - Service running on recent kernel
    - Jobs running on SL6 via docker
    - Enables modern technology for HEP

- Extending data analysis to Clouds
    - Data provisioned via caching
    - Data access via Cloud-local xrootd proxies

BACKUP

Experience: User Workflows

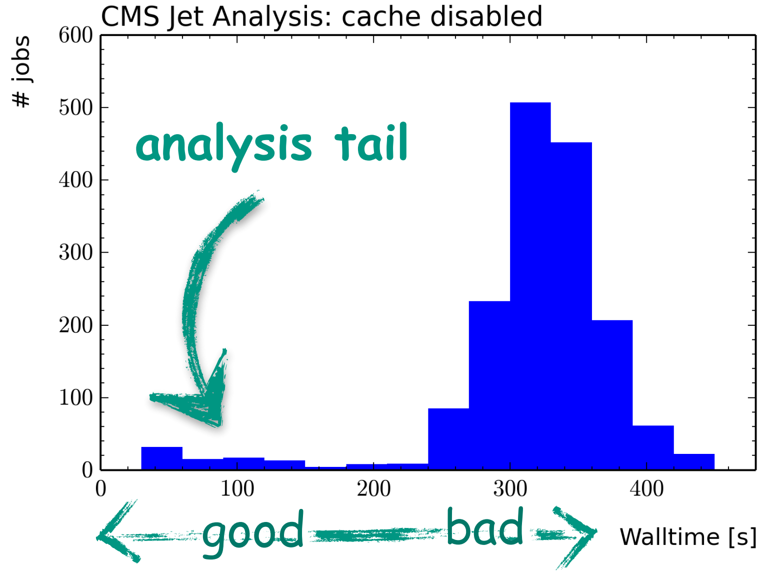
- Benchmark workflow: CMS calibration
  - ROOT n-tuple analysis
  - 400 GB LHC run1 input data
  - Notable improvement

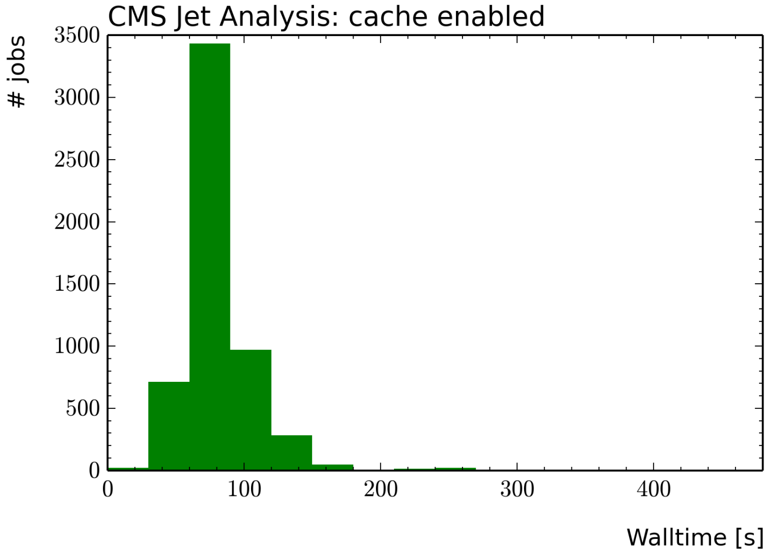


Experience: User Workflows

- Benchmark workflow: CMS calibration
  - ROOT n-tuple analysis
  - 400 GB LHC run1 input data
  - Notable improvement

- Used for LHC run2 user analyses
  - Single patch to submission tool
  - Fully transparent in regular cluster
  - Non-intrusive to regular operation


CMS Jet Analysis: cache disabled


CMS Jet Analysis: cache enabled

Experience: HTDA Middleware Performance

- Mature prototype implementation
  - Stable operation for 6+ months
  - Worker CPU/RSS overhead negligible

	CPU	RSS
Cache	3,5 %	120 MB
Locator	1,0 %	60 MB
Coordinator	14,1 %	1 GB

Experience: HTDA Middleware Performance

- Mature prototype implementation
  - Stable operation for 6+ months
  - Worker CPU/RSS overhead negligible

	CPU	RSS
Cache	3,5 %	120 MB
Locator	1,0 %	60 MB
Coordinator	14,1 %	1 GB

- Mixed experiences with SL6 (2.6 kernel)
  - Similar analysis (ROOT) performance as on 3.X kernel systems
  - Availability reduced by unstable AUFS 2.X (for cache access)

Experience: HTDA Middleware Performance

- Mature prototype implementation
  - Stable operation for 6+ months
  - Worker CPU/RSS overhead negligible

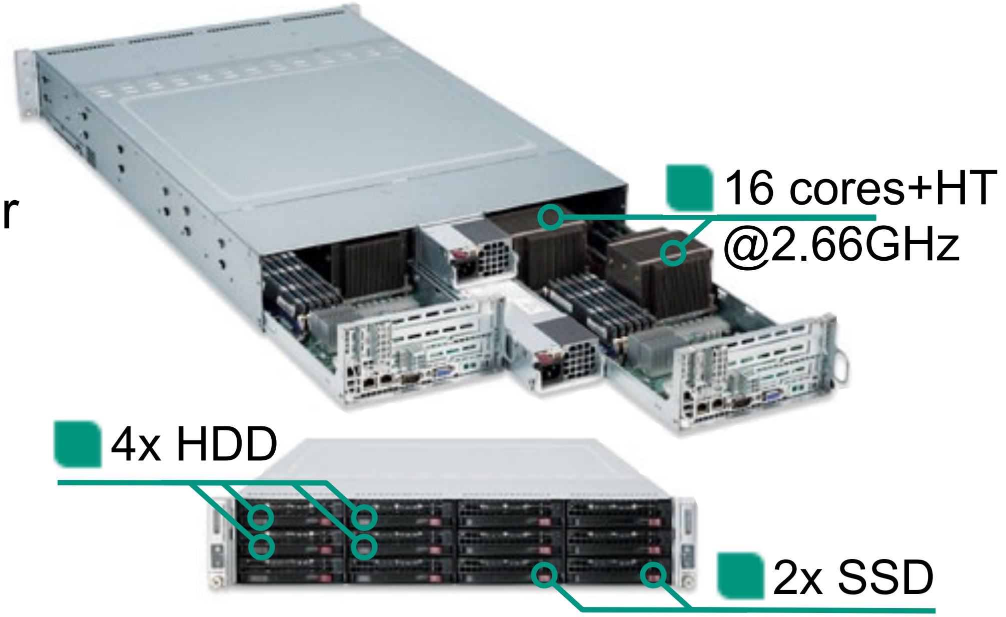
	CPU	RSS
Cache	3,5 %	120 MB
Locator	1,0 %	60 MB
Coordinator	14,1 %	1 GB

- Mixed experiences with SL6 (2.6 kernel)
  - Similar analysis (ROOT) performance as on 3.X kernel systems
  - Availability reduced by unstable AUFS 2.X (for cache access)

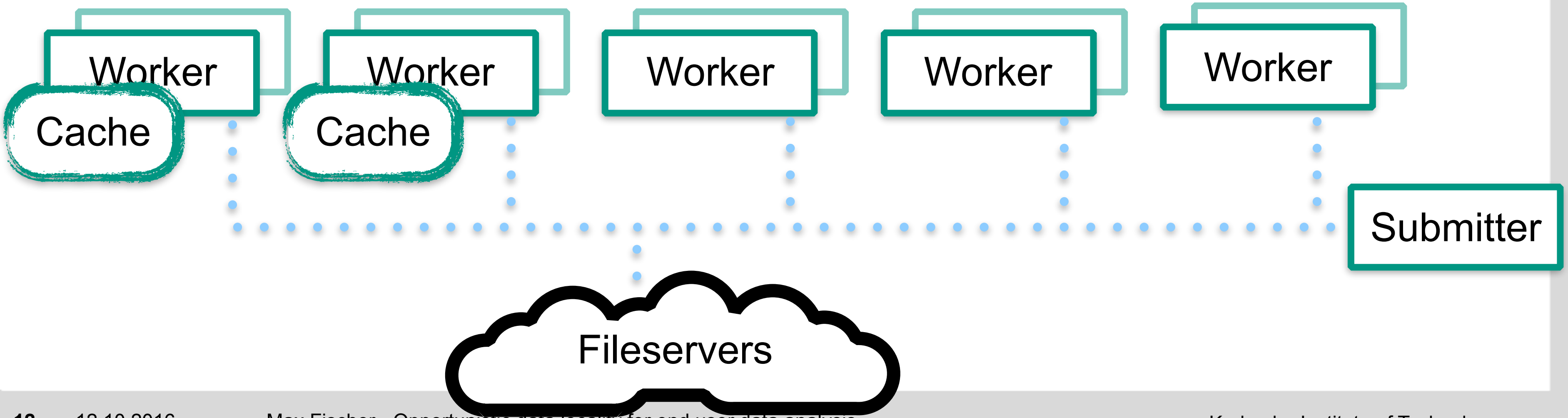
- Open issues: no showstoppers
  - Deliberate cleanup of meta-data and file reallocation
  - Tweaks and optimizations

Prototype Batch System

- Extends HTCondor setup
  - Static, opportunistic and HTDA nodes in same cluster
  - 5 HTDA worker nodes á 500 GB SSD cache
  - 6 fileservers
    - POSIX + XRootD



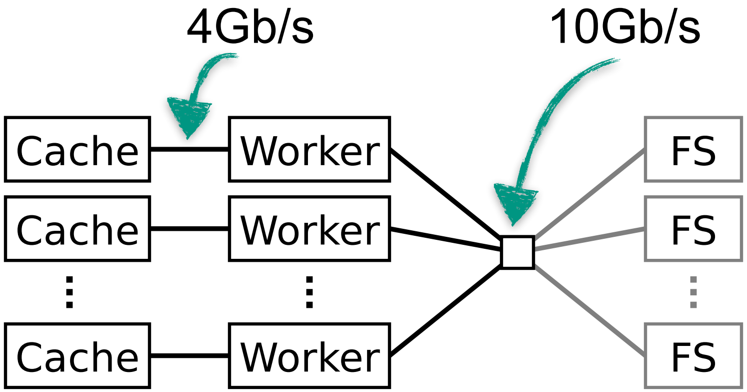
- 16 cores+HT @2.66GHz
- 4x HDD
- 2x SSD



Worker
Cache
Worker
Cache
Worker
Worker
Worker
Submitter
Fileservers

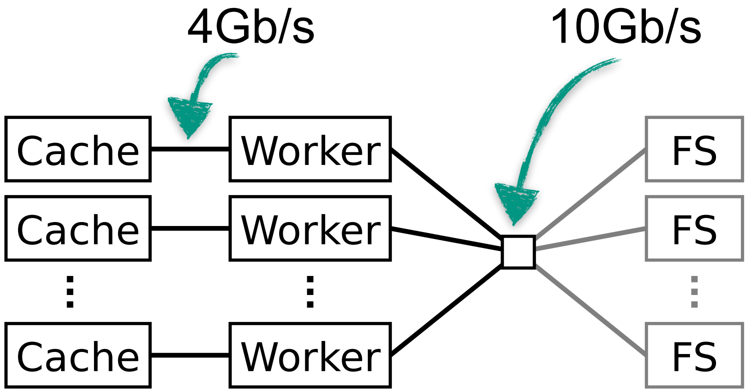
Coordinated Caching: Throughput Simulation

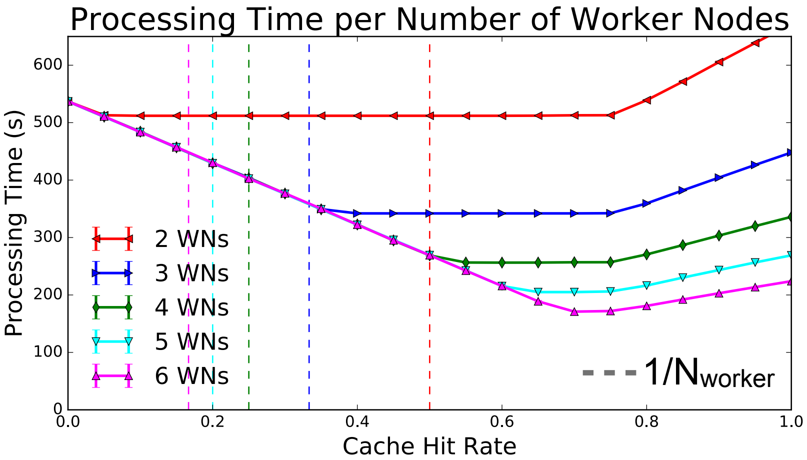
- Batch system throughput simulation
  - Setup of KIT Tier3
  - Parameters: local hit rate, N_{worker}



Coordinated Caching: Throughput Simulation

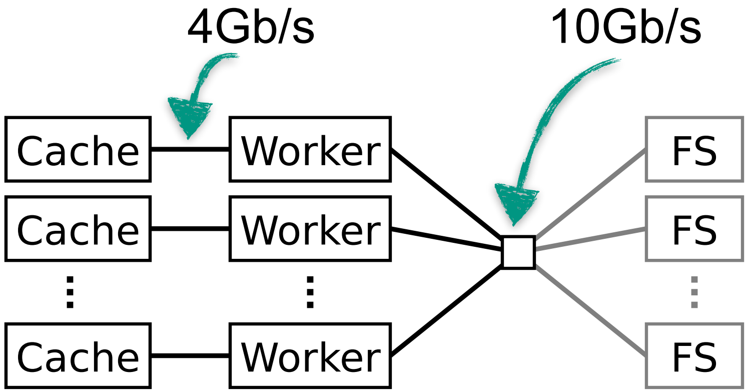
- Batch system throughput simulation
  - Setup of KIT Tier3
  - Parameters: local hit rate, N_{worker}
- Caching allows horizontal scaling
  - Throughput scales with workers…
  - …if jobs are scheduled to data





Coordinated Caching: Throughput Simulation

- 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
  - Leverage remote I/O
  - Potential to…
    - Use simple algorithms
    - Increase effective cache size



4Gb/s

10Gb/s

Cache — Worker

Cache — Worker

⋮

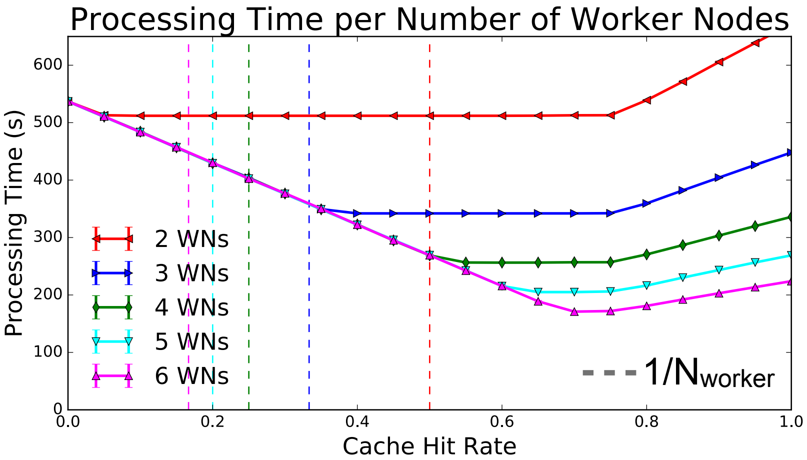
Cache — Worker

FS

FS

⋮

FS



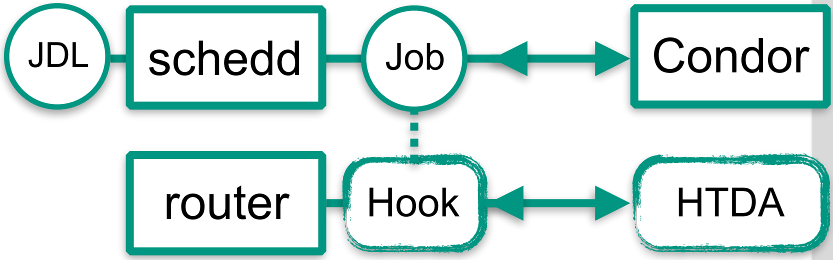
Experience: Batch System Integration

- Hooks on submission hosts via *job_router*
  - Integrates directly into batch system
  - Efficient push instead of pull behavior
  - Constraint of 1 active route (service) per job

Experience: Batch System Integration

- Hooks on submission hosts via *job_router*
  - Integrates directly into batch system
  - Efficient push instead of pull behavior
  - Constraint of 1 active route (service) per job

- Only job meta-data exchanged
  - Job features from HTCondor
  - Placement information from HTDA

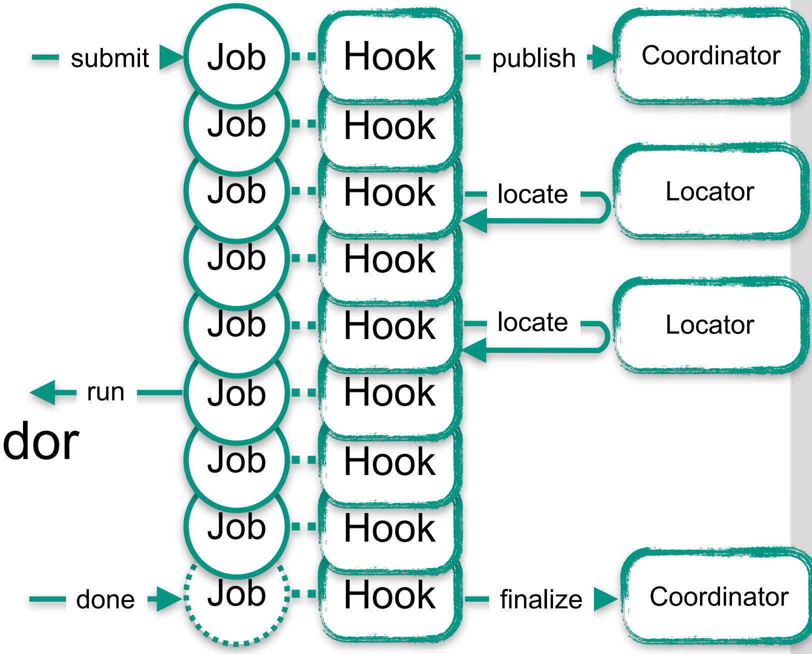


Experience: Batch System Integration

- Hooks on submission hosts via *job_router*
  - Integrates directly into batch system
  - Efficient push instead of pull behavior
  - Constraint of 1 active route (service) per job

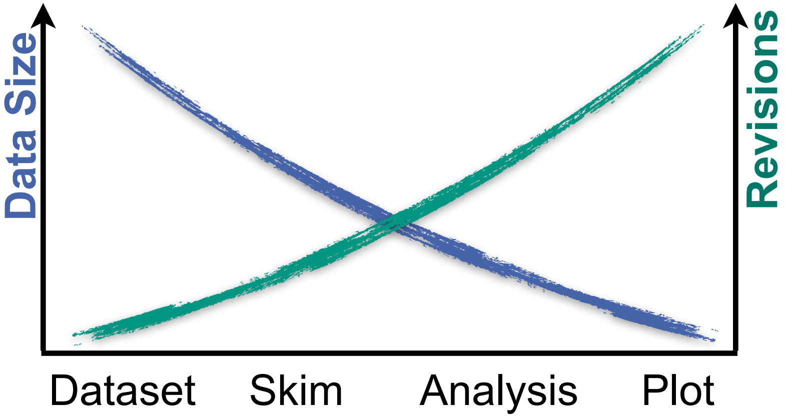
- Only job meta-data exchanged
  - Job features from HTCondor
  - Placement information from HTDA

- Efficient interface to HTCondor
  - Selection/tracking handled by HTCondor
  - Hook skips any meaningless updates
  - Arbitrary number of untracked jobs



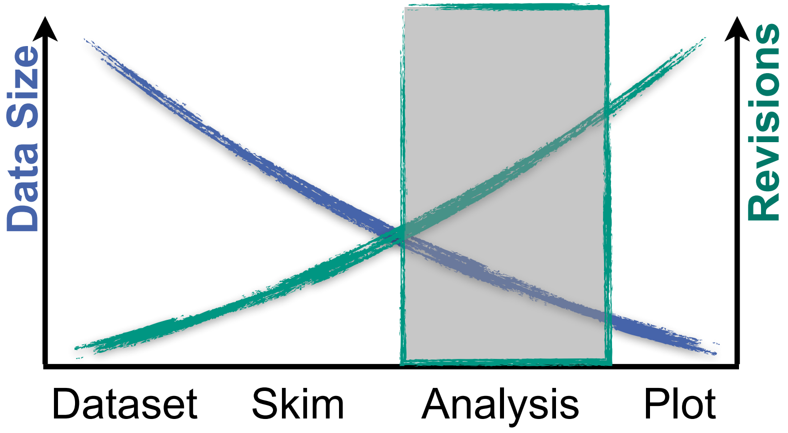
Context: HEP End User Data Analysis

- Hierarchical, iterative workflows
  - Reduction of data size
  - Increase of iterations
  - Dedicated processing environments



Context: HEP End User Data Analysis

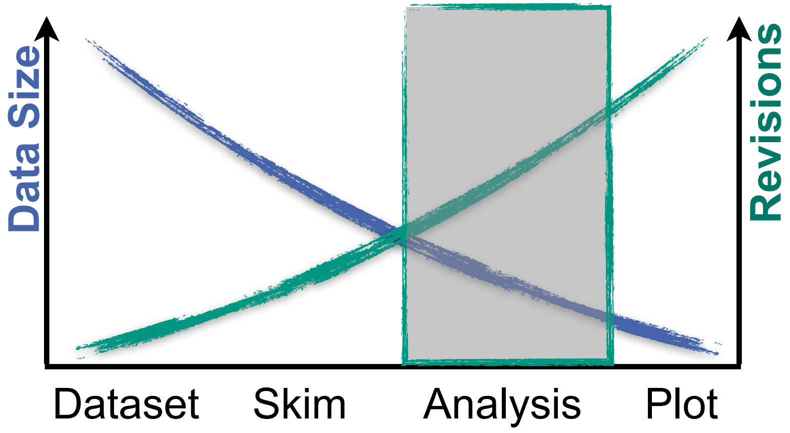
- Hierarchical, iterative workflows
  - Reduction of data size
  - Increase of iterations
  - Dedicated processing environments

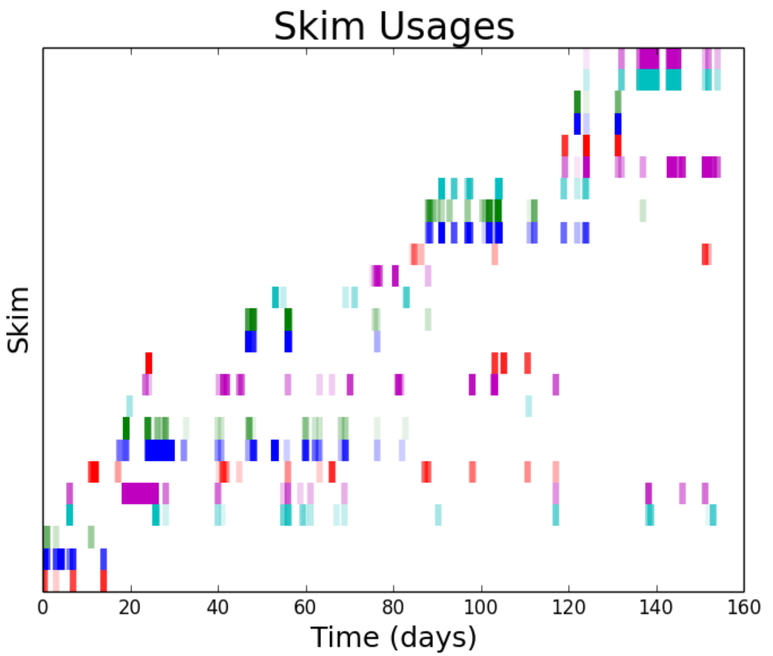


- Data intense analyses on Tier 3
  - Standard batch systems and fileservers
  - Extraction of observables from optimized data sets/formats

Context: HEP End User Data Analysis

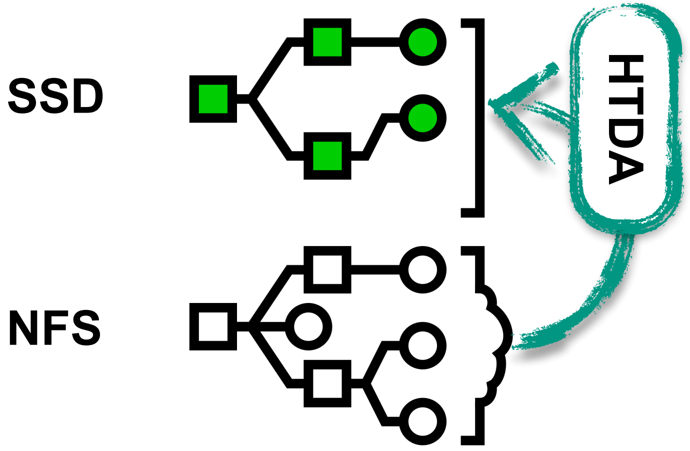
- Hierarchical, iterative workflows
  - Reduction of data size
  - Increase of iterations
  - Dedicated processing environments
- Data intense analyses on Tier 3
  - Standard batch systems and fileservers
  - Extraction of observables from optimized data sets/formats
- Usage suitable for caching
  - Repeated processing of same input
  - Strongly dependent on input rate





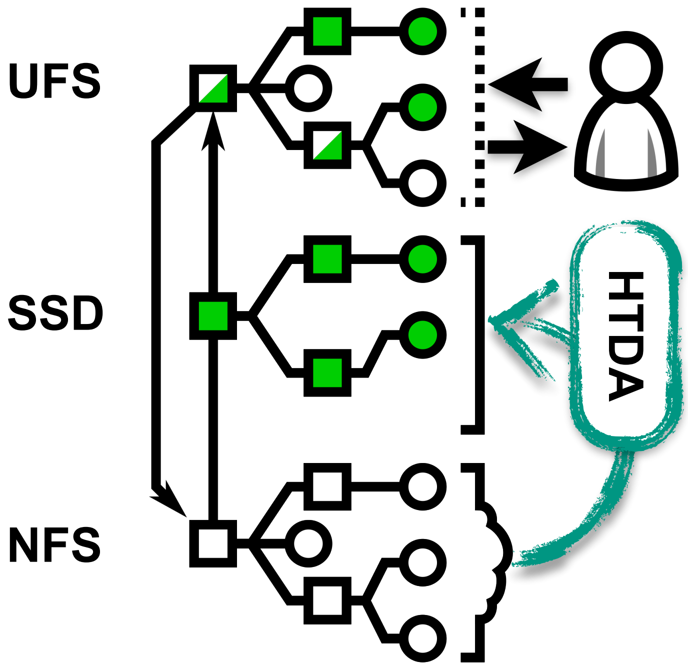
Cache Content Access

- Cache node stages/unstages files according to coordinator request



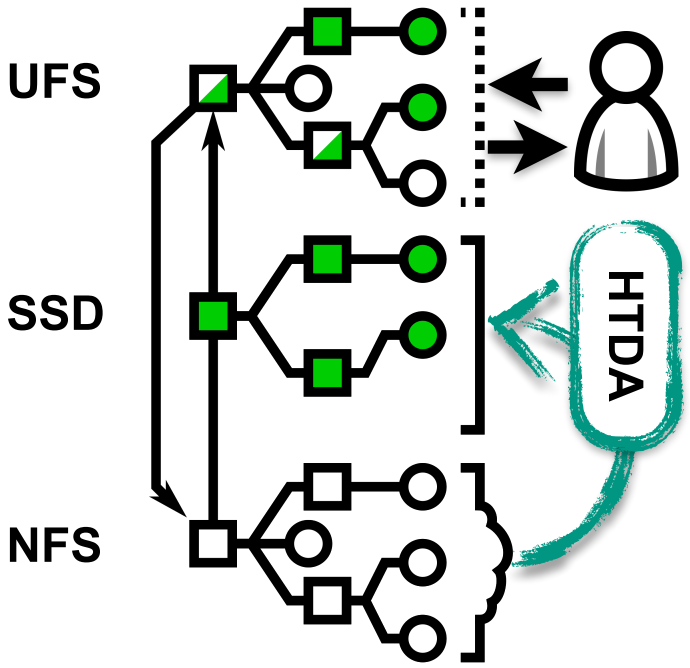
Cache Content Access

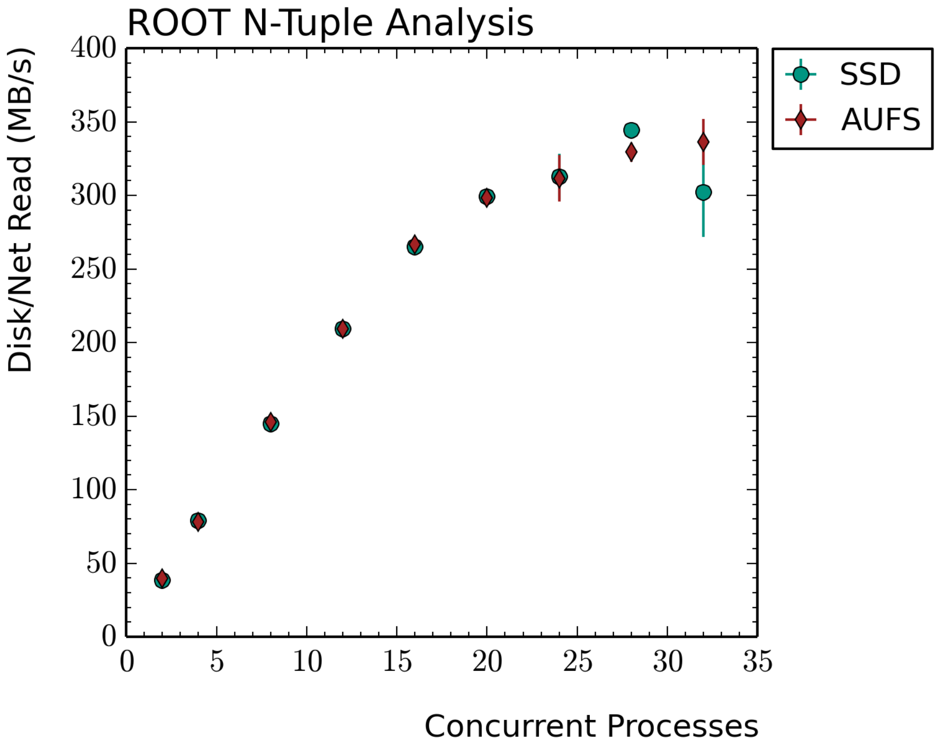
- Cache node stages/unstages files according to coordinator request
- Union File System provides transparent cache access for users



Cache Content Access

- Cache node stages/unstages files according to coordinator request
- Union File System provides transparent cache access for users

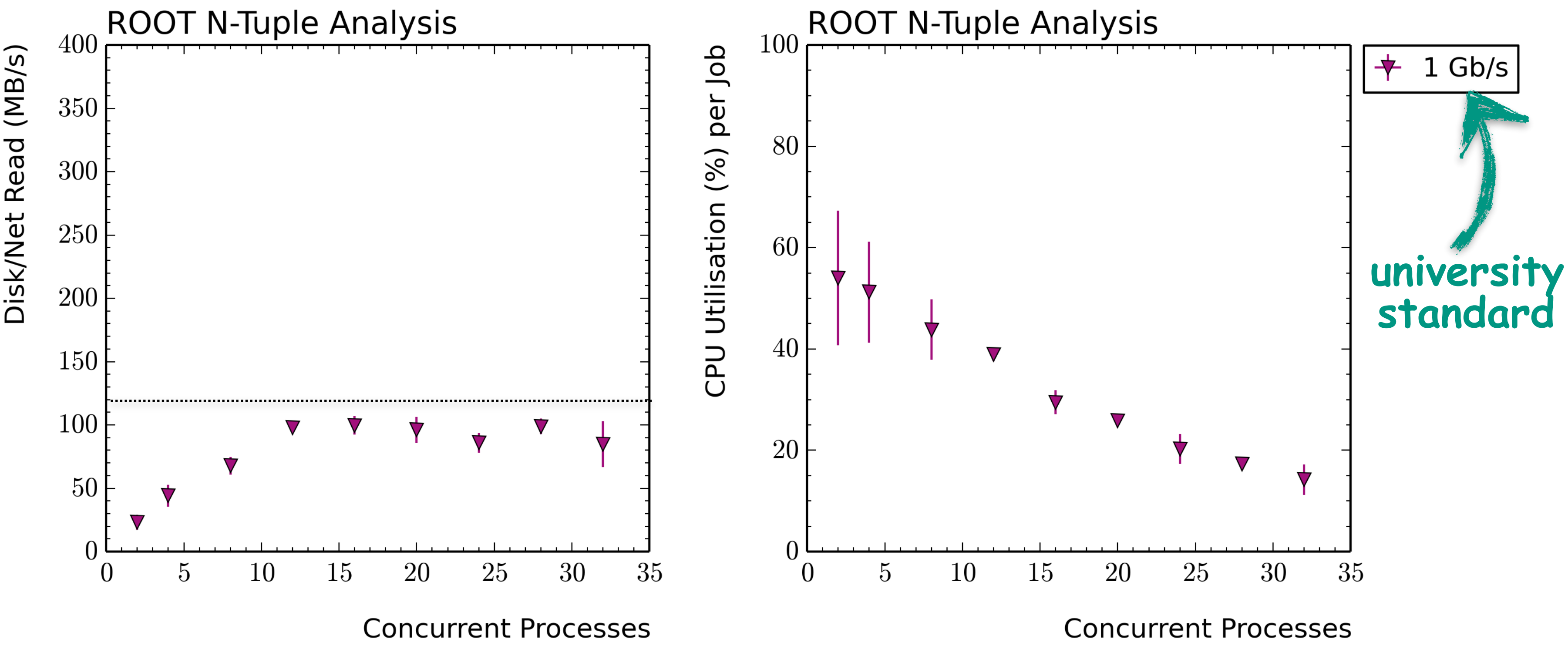




- Lightweight cache access ensures optimal performance

I/O Performance Evaluation

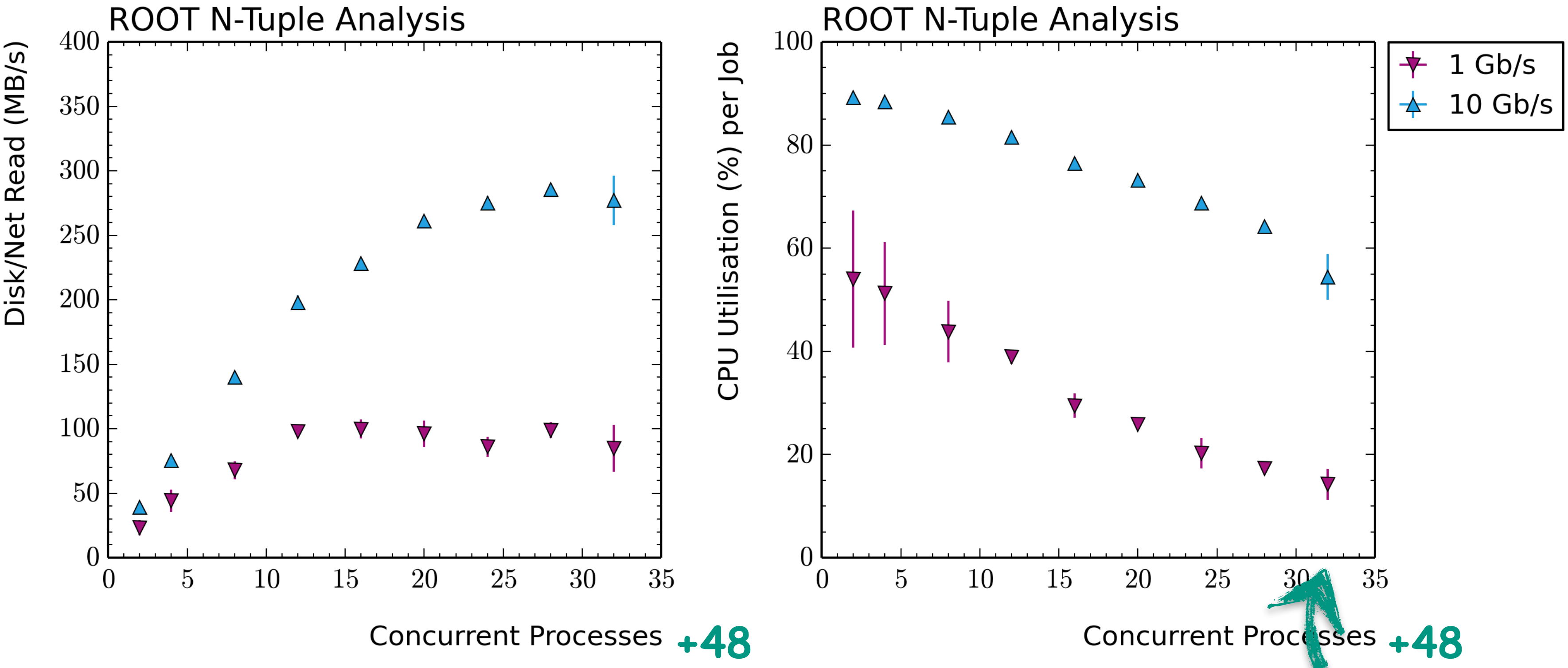
- CMS jet calibration analysis (ROOT n-tuple)



university standard

I/O Performance Evaluation

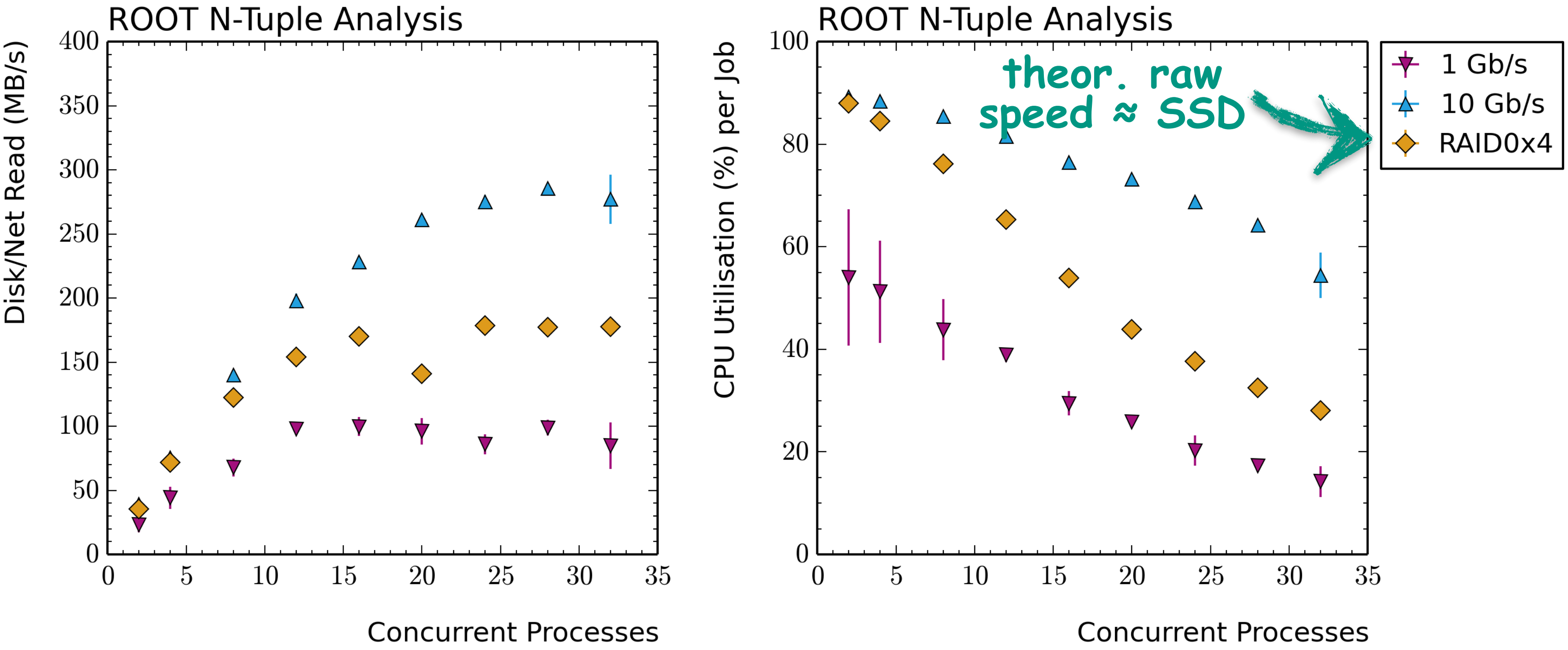
- CMS jet calibration analysis (ROOT n-tuple)
- Additional 48 concurrent reads from other workers for 10 Gb/s test



2006 Tier2
CPU capacity

I/O Performance Evaluation

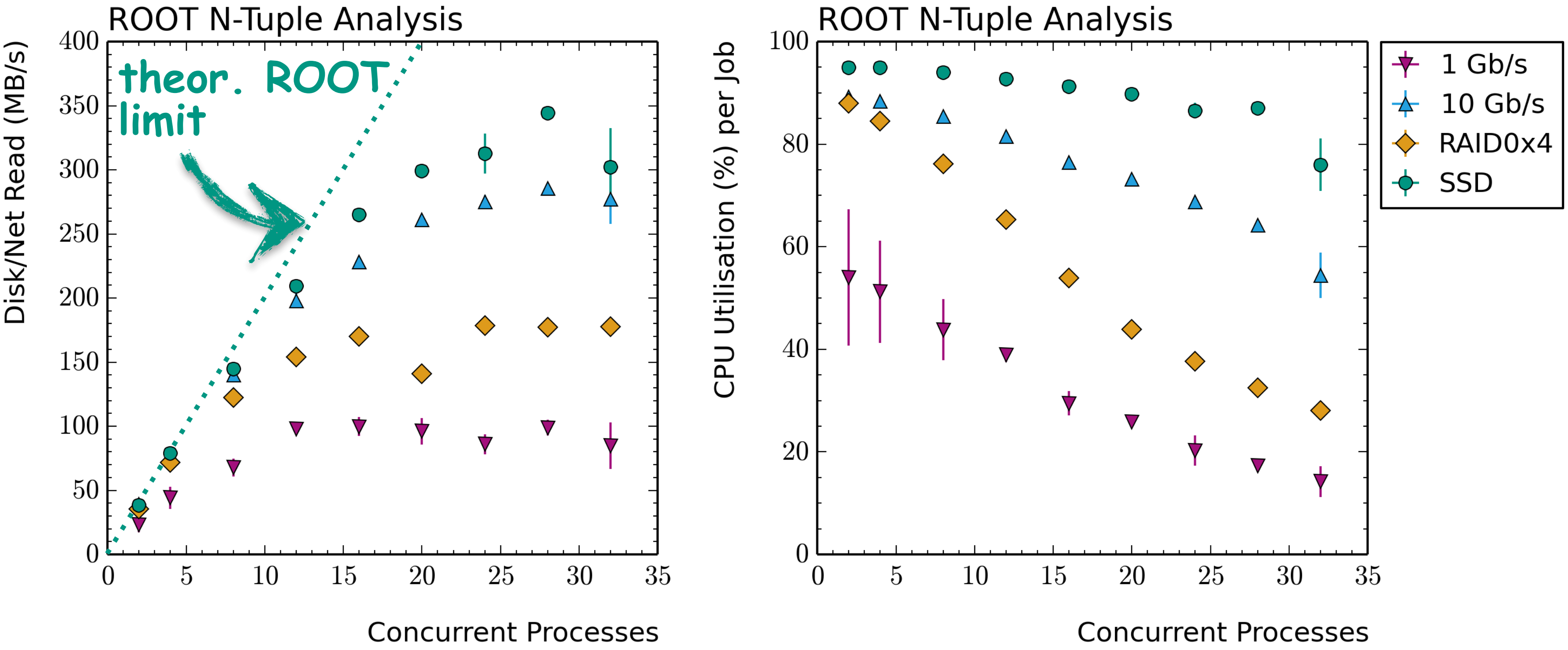
- CMS jet calibration analysis (ROOT n-tuple)
- Additional 48 concurrent reads from other workers for 10 Gb/s test



- 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