Data Locality via Coordinated Caching for Distributed Processing

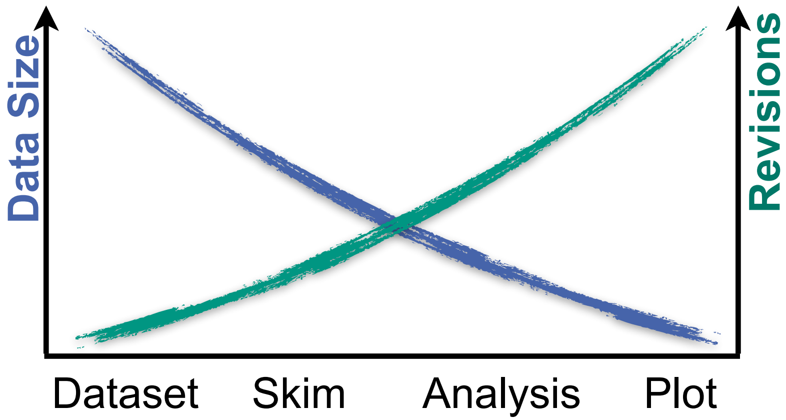
Max Fischer, Eileen Kühn, Manuel Giffels, Christopher Jung

ACAT 2016

Karlsruhe Institute of Technology

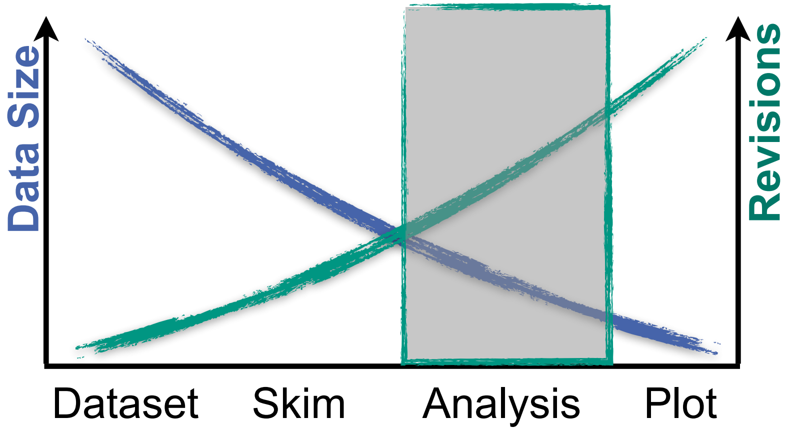
Context: HEP End User Data Analysis

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



Context: HEP End User Data Analysis

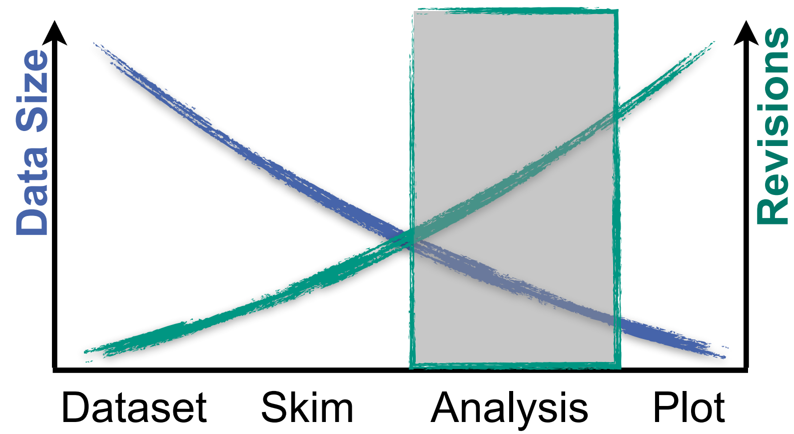
- Hierarchical, iterative workflows
  - Reduction of data size
  - Increase of iterations
  - Dedicated processing environments

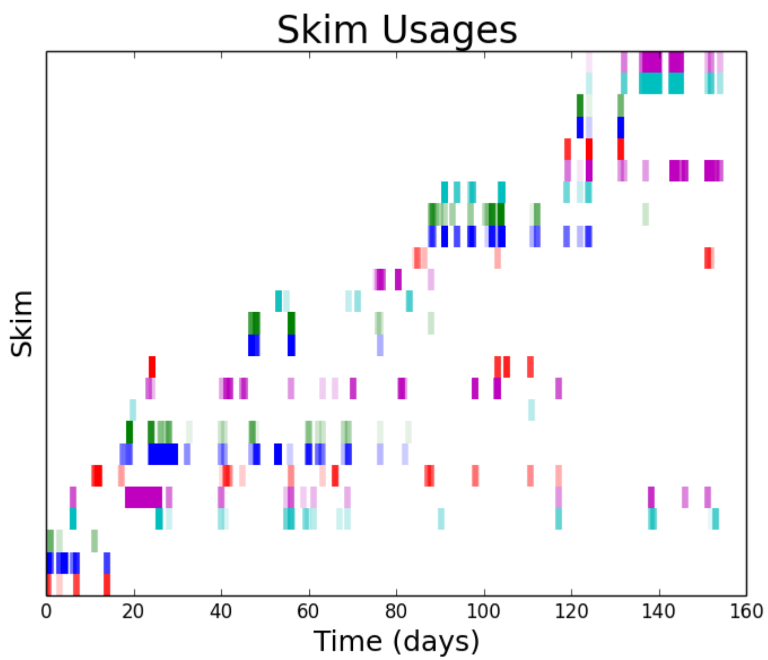


- Data intense analyses on Tier 3
  - Standard batch systems and fileservers
  - Extraction of observables from optimized data sets/formats

Context: HEP End User Data Analysis

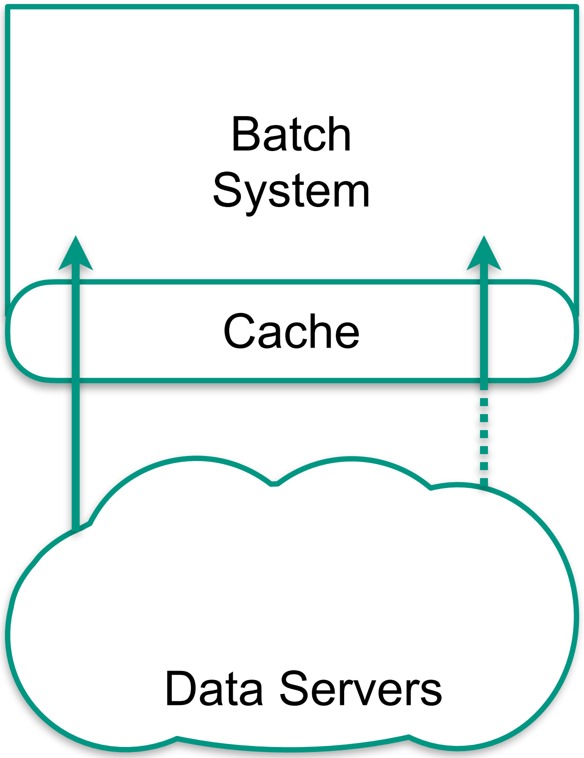
- Hierarchical, iterative workflows
  - Reduction of data size
  - Increase of iterations
  - Dedicated processing environments
- Data intense analyses on Tier 3
  - Standard batch systems and fileservers
  - Extraction of observables from optimized data sets/formats
- Usage suitable for caching
  -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





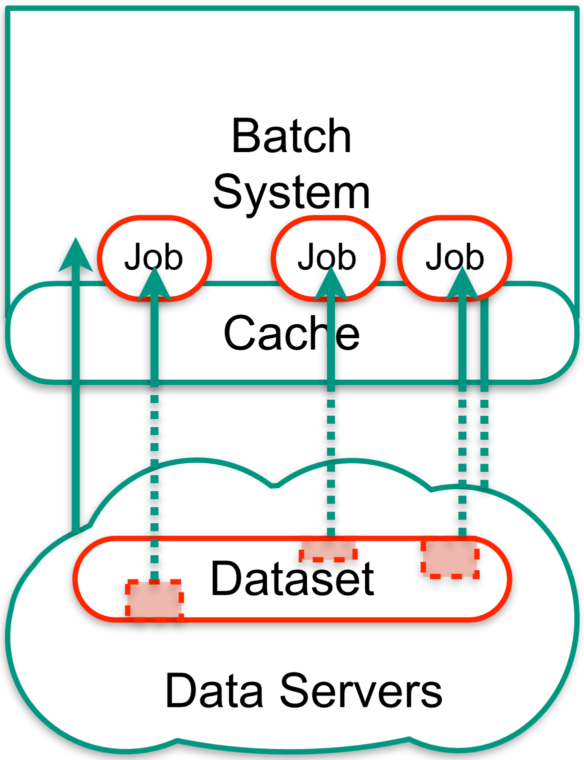
Coordinated Caching: Overview

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



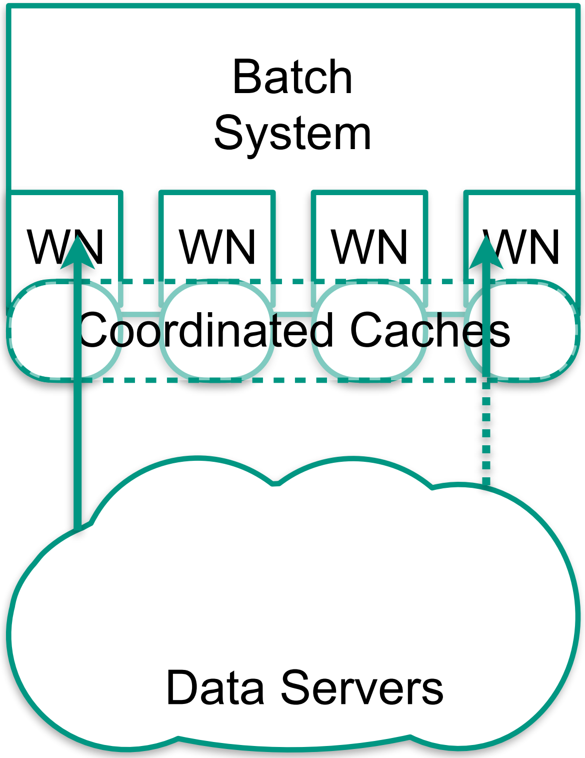
Coordinated Caching: Overview

- Caching 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



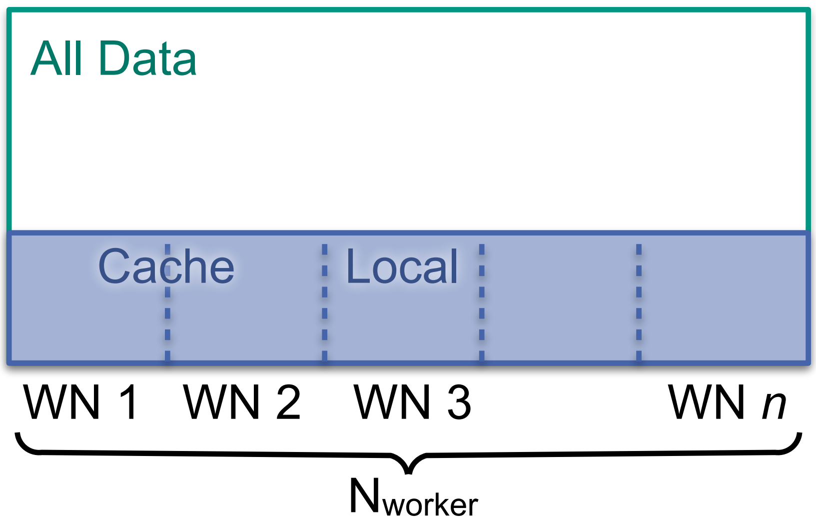
Coordinated Caching: Overview

- Caching 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
- Implementation at host granularity
  - Array of individual caches on worker nodes
  - Caches coordinated by global service
  - Some glue for data locality…



Coordinated Caching: Data Availability

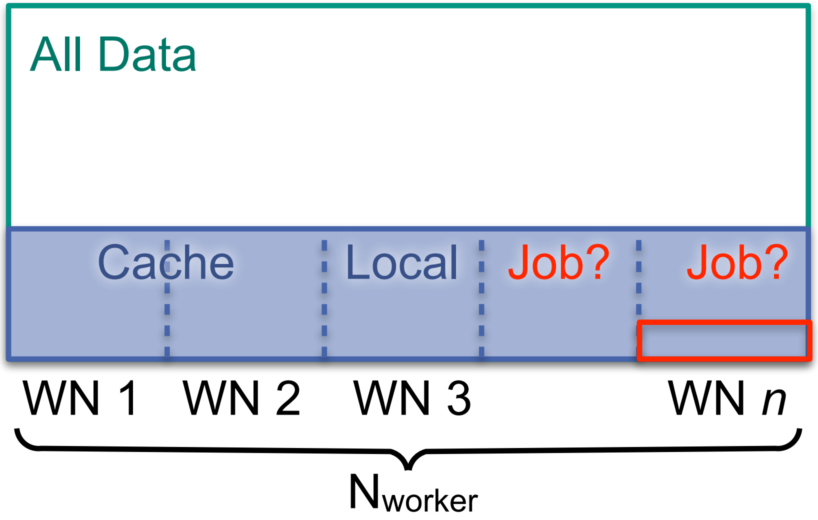
- Distributed caching complicates cache access
  - Data cached anywhere (cache hit rate)
  - Data local on job host (local hit rate)



Coordinated Caching: Data Availability

- 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

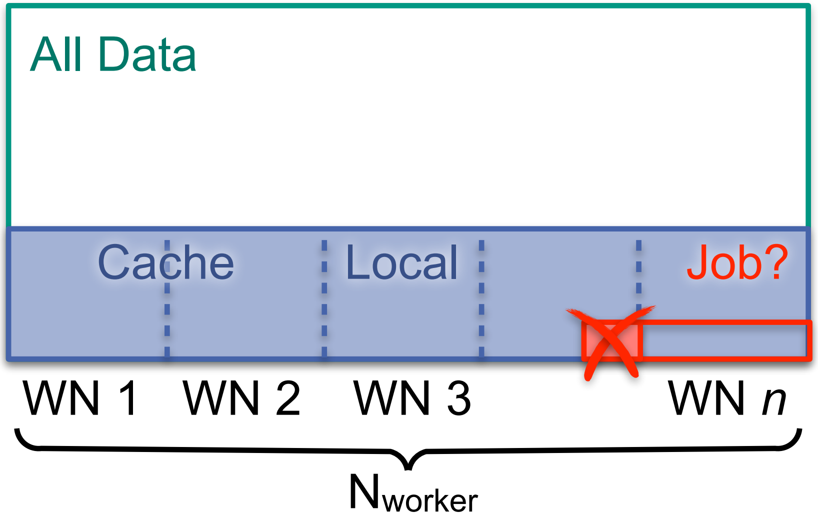


Coordinated Caching: Data Availability

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

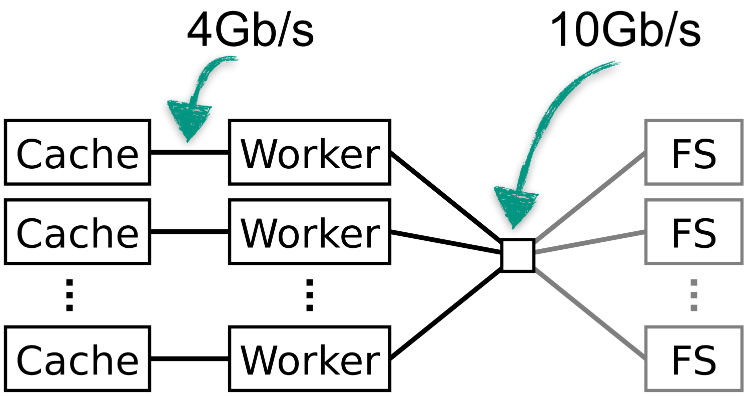
- Schedule Jobs to input data location
  - Unscheduled hit rate limited to ~$1/N_{worker}$
  - Data location published to batch system

- Place data to match workflows
  - Jobs require groups of files
  - Placement uses observed data splitting



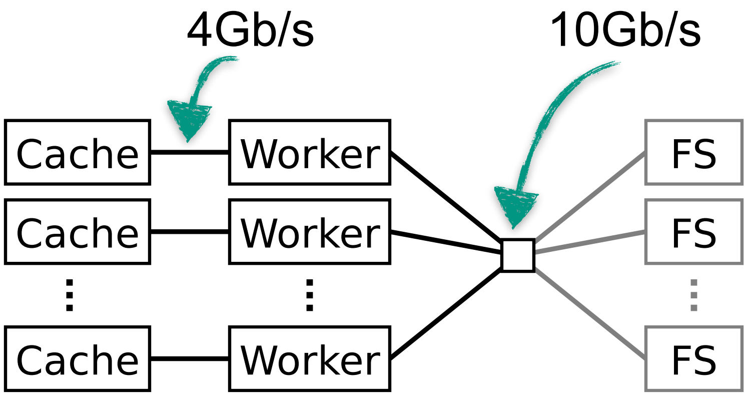
Coordinated Caching: Throughput Simulation

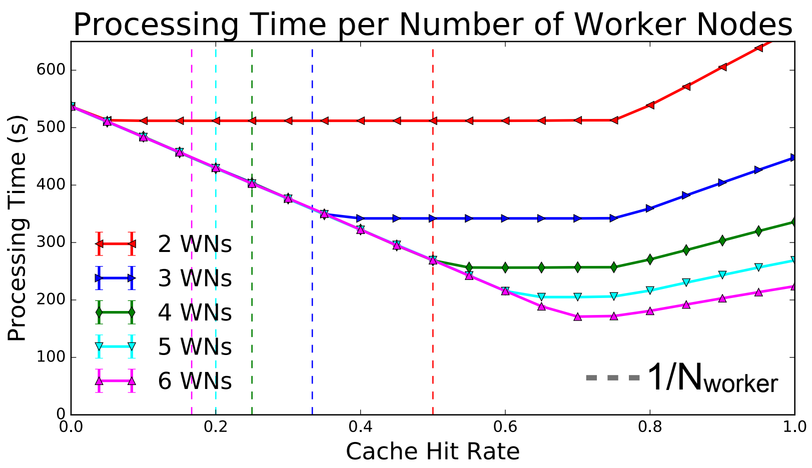
- Batch system throughput simulation
  - Setup of KIT Tier3
  - Parameters: local hit rate, N_{worker}



Coordinated Caching: Throughput Simulation

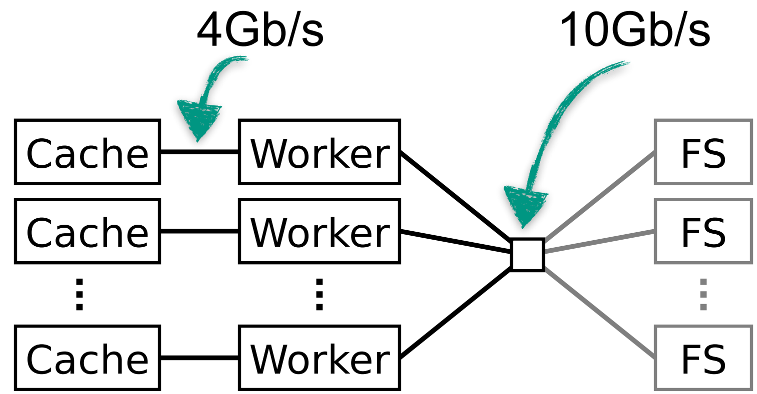
- Batch system throughput simulation
  - Setup of KIT Tier3
  - Parameters: local hit rate, N_{worker}
- Caching allows horizontal scaling
  - Throughput scales with workers…
  - …if jobs are scheduled to data

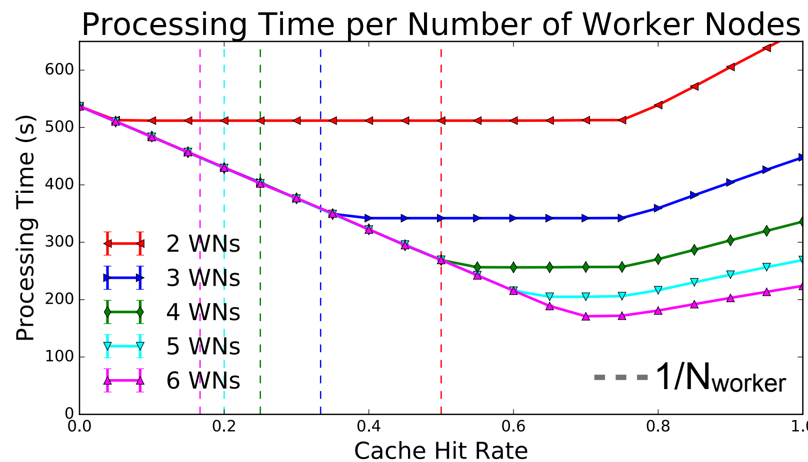




Coordinated Caching: Throughput Simulation

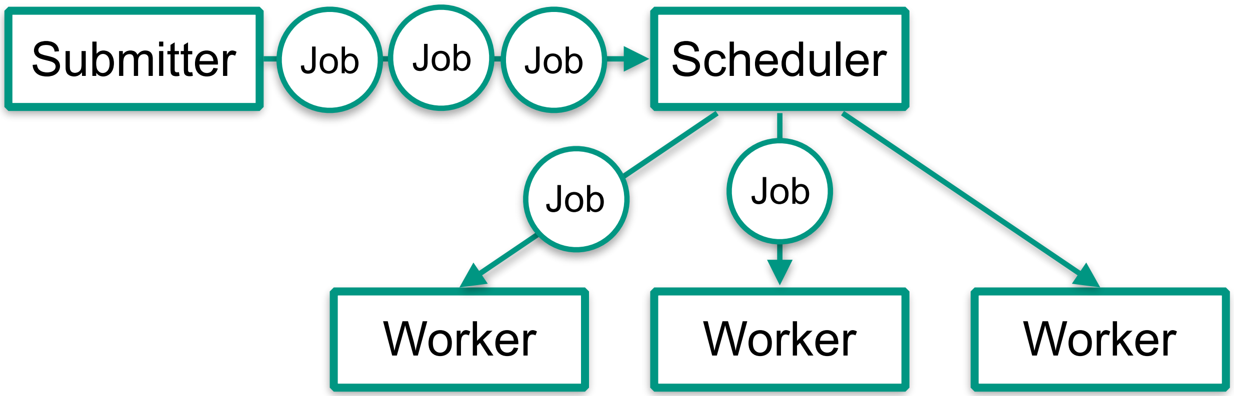
- Batch system throughput simulation
  - Setup of KIT Tier3
  - Parameters: local hit rate, N_{worker}
- Caching allows horizontal scaling
  - Throughput scales with workers…
  - …if jobs are scheduled to data
- Perfect hit rate not ideal
  - Leverage remote I/O
  - Potential to…
    - Use simple algorithms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





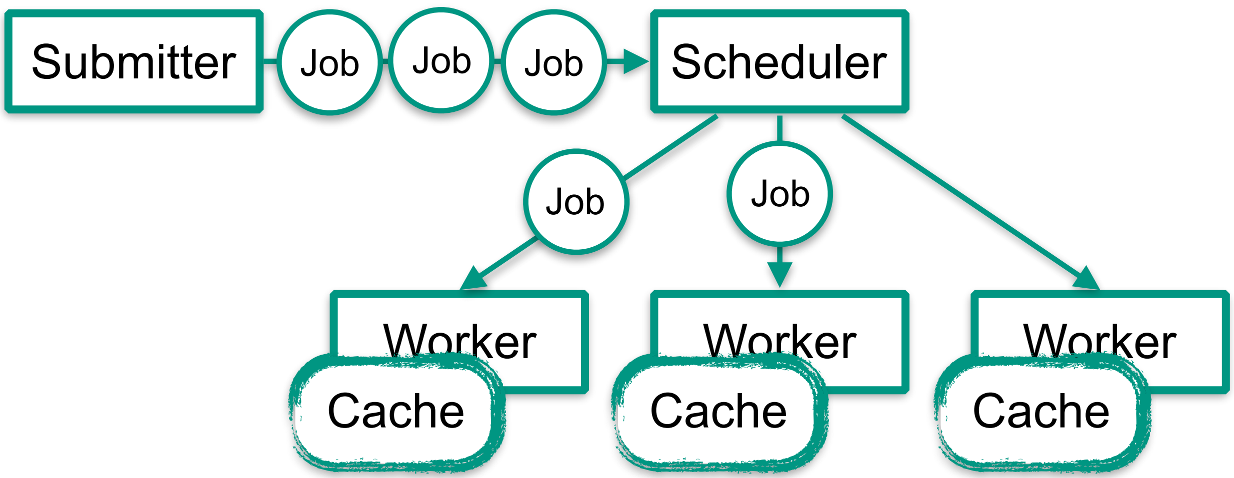
HTDA Batch System Extension

High Throughput Data Analysis



HTDA Batch System Extension

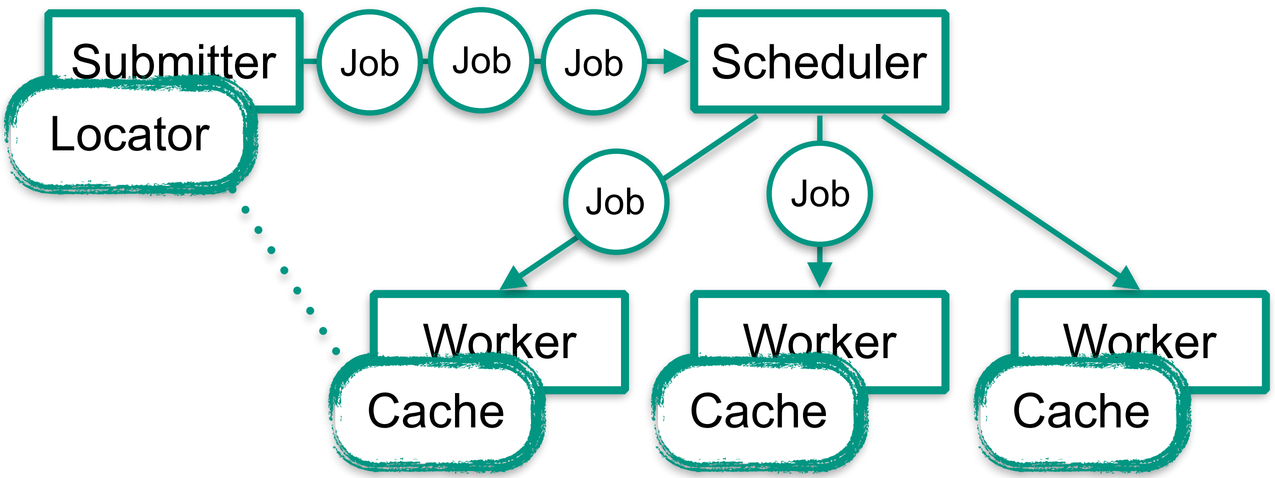
High Throughput Data Analysis



- Caches maintain data copies on worker nodes

HTDA Batch System Extension

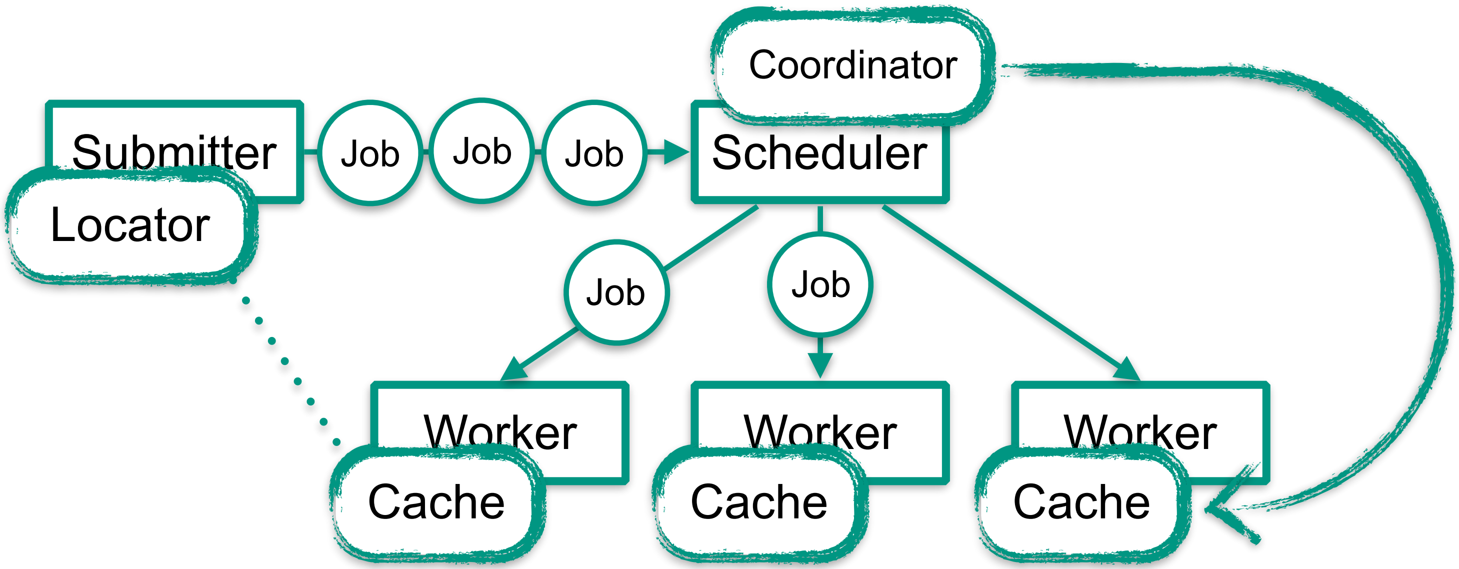
High Throughput Data Analysis



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

HTDA Batch System Extension

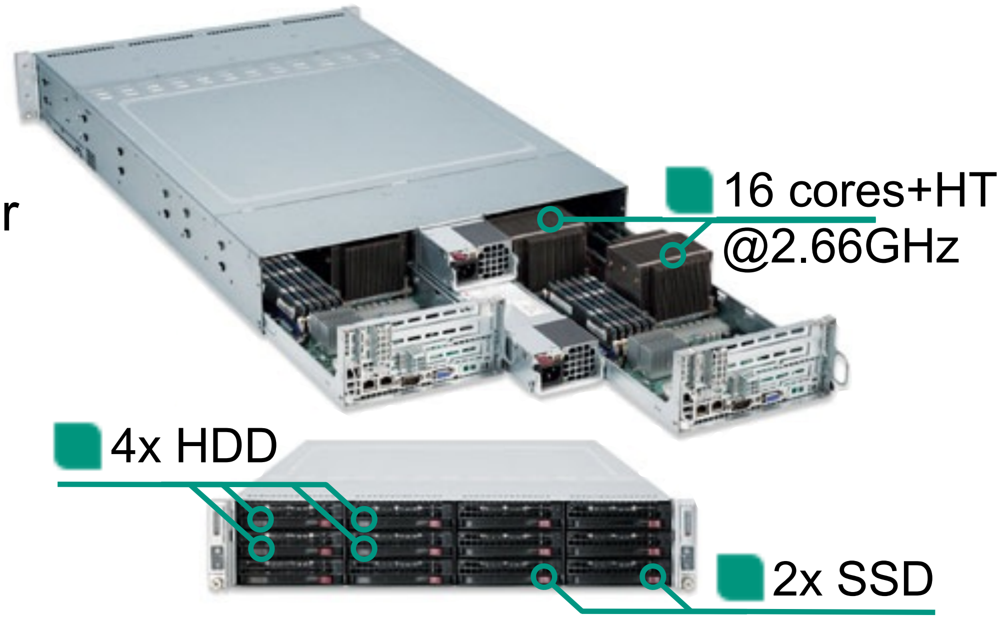
High Throughput Data Analysis

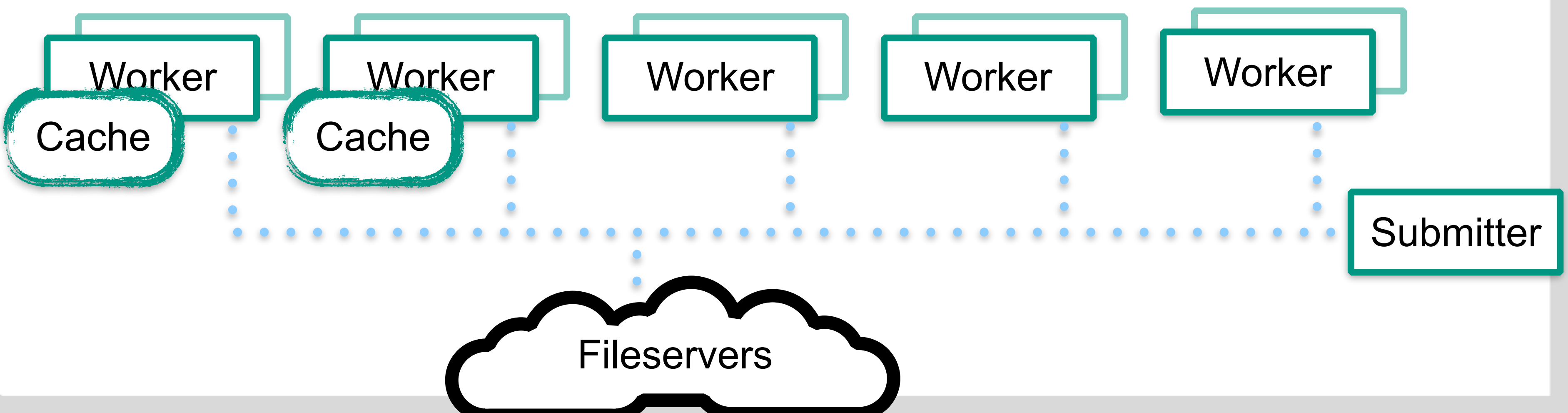


- Caches maintain data copies on worker nodes
- Locator provides locality information for jobs
- Coordinator schedules files for caching on nodes

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





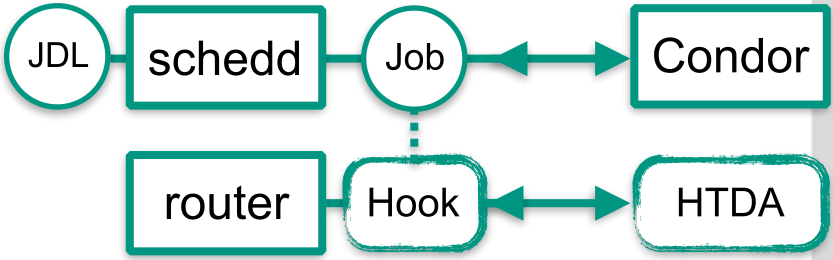
Experience: Batch System Integration

- 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

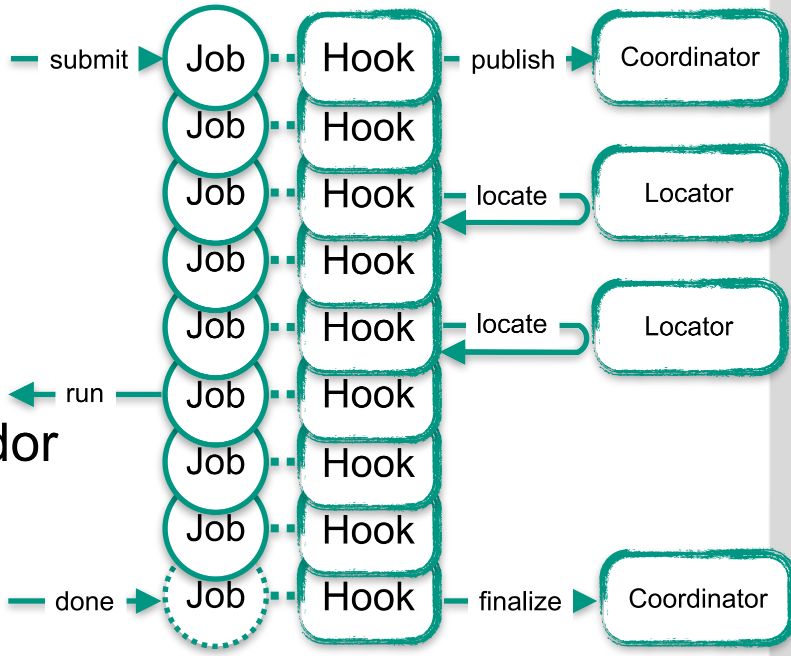
- Hooks on submission hosts via *job_router*
  - Integrates directly into batch system
  - Efficient push instead of pull behavior
  - Constraint of 1 active route (service) per job

- 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

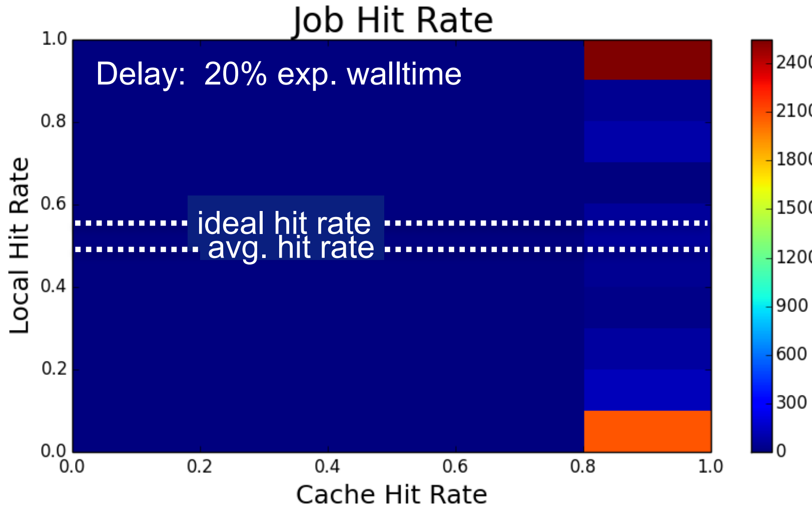


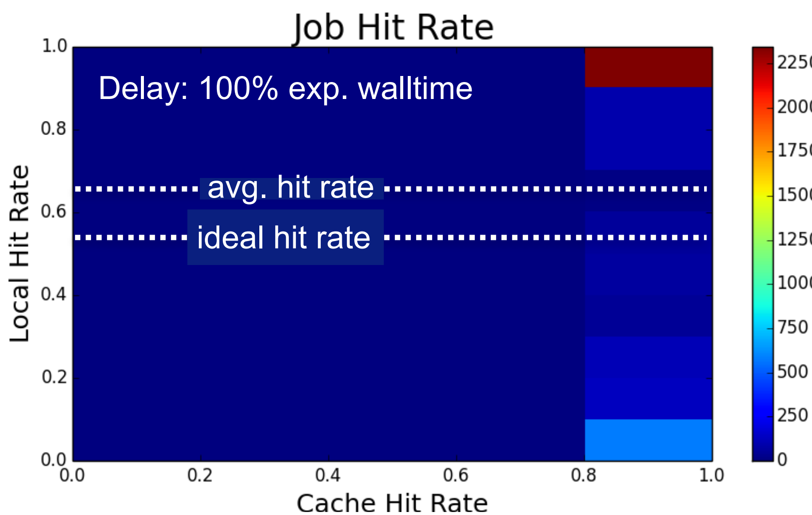
Experience: Scheduling and Cache Hit Rate

- Data locality added to job scheduling
  - Nodes ranked by local data
  - Fixed delay for hosts w/o local data

Experience: Scheduling and Cache Hit Rate

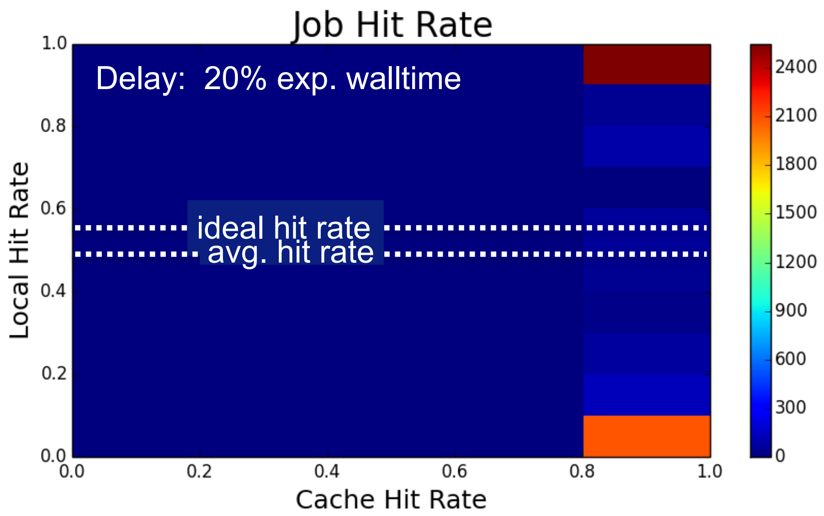
- Data locality added to job scheduling
  - Nodes ranked by local data
  - Fixed delay for hosts w/o local data
- Tradeoff efficiency vs responsiveness
  - Wait for perfect vs use worse now
  - Large delay only efficient for high cluster utilization

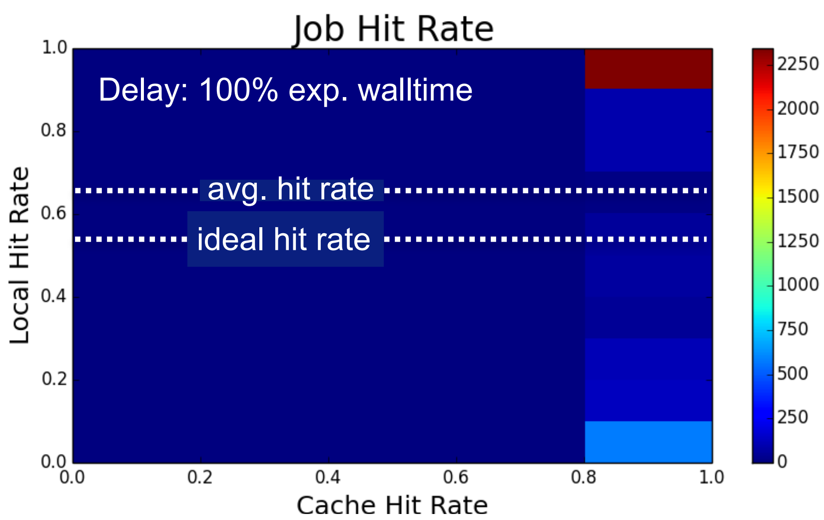




Experience: Scheduling and Cache Hit Rate

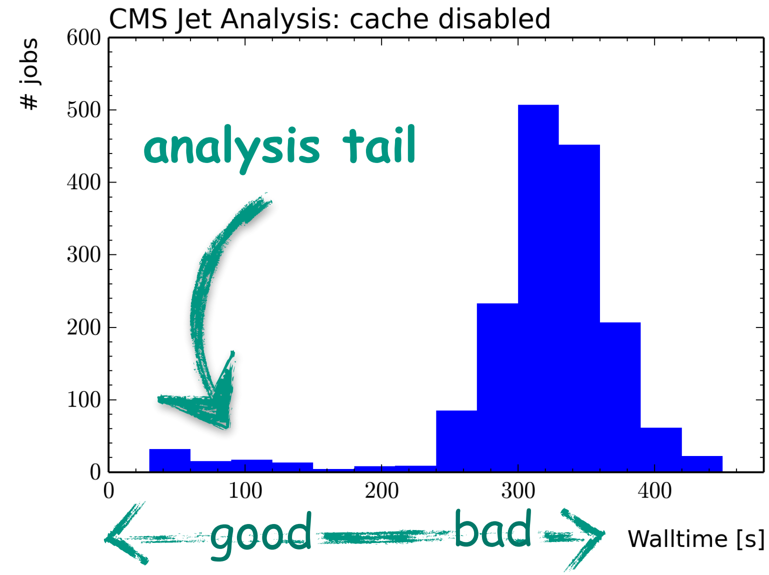
- Data locality added to job scheduling
  - Nodes ranked by local data
  - Fixed delay for hosts w/o local data
- Tradeoff efficiency vs responsiveness
  - Wait for perfect vs use worse now
  - Large delay only efficient for high cluster utilization
- Simple approach mostly good enough
  - Fixed delay, tuned for user demand
  - Possible „pile-up" of jobs scheduled to inefficient hosts
  - Investigating suspension of inefficient jobs to clear pile-up

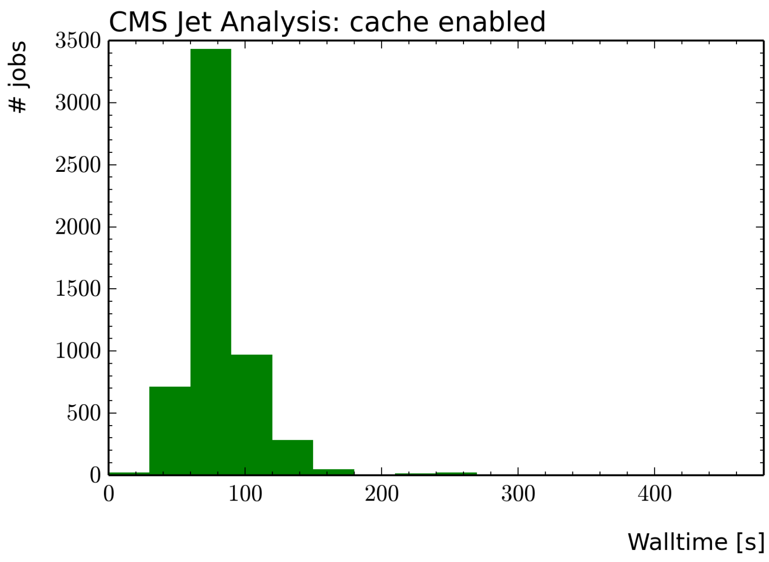




Experience: User Workflows

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

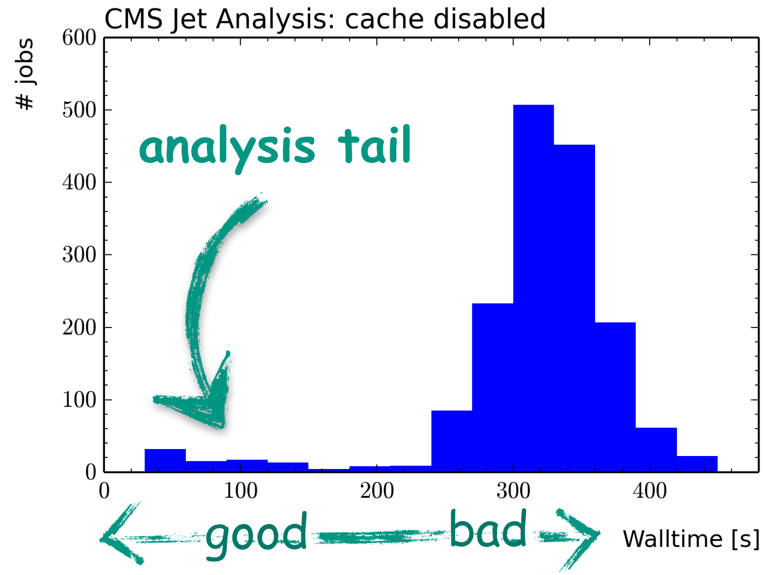


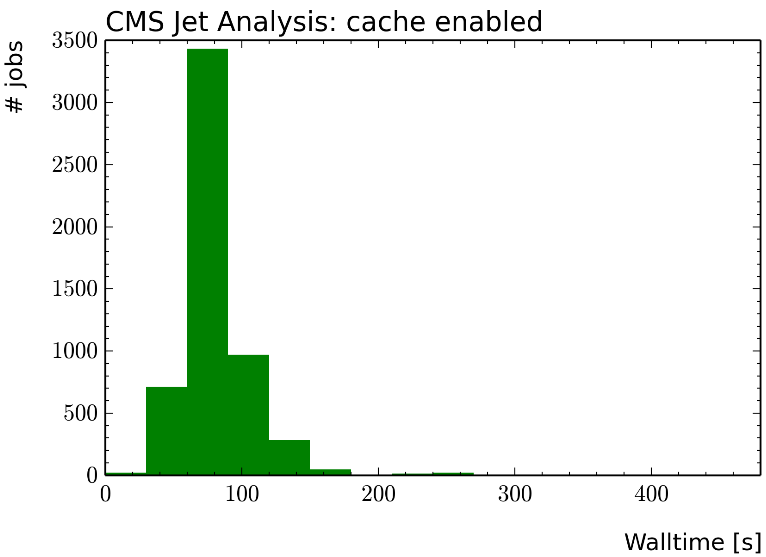


Experience: User Workflows

- 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





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

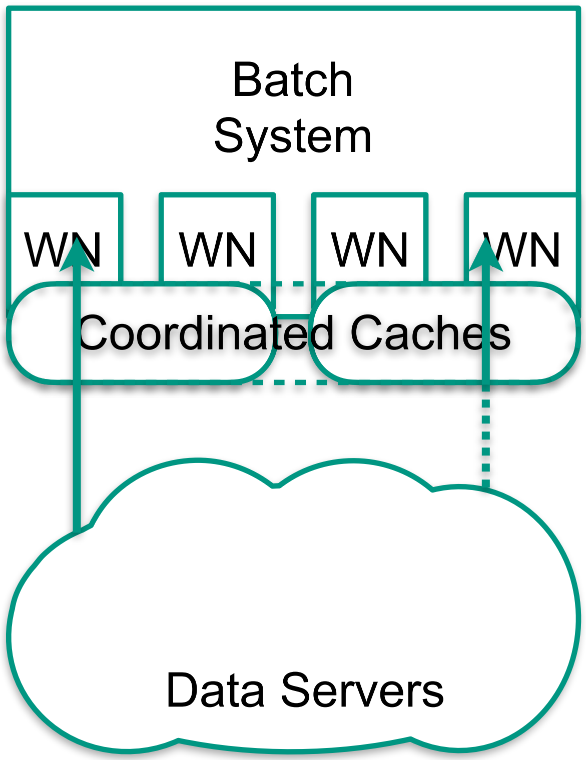
	CPU	RSS
Cache	3,5 %	120 MB
Locator	1,0 %	60 MB
Coordinator	14,1 %	1 GB

- 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

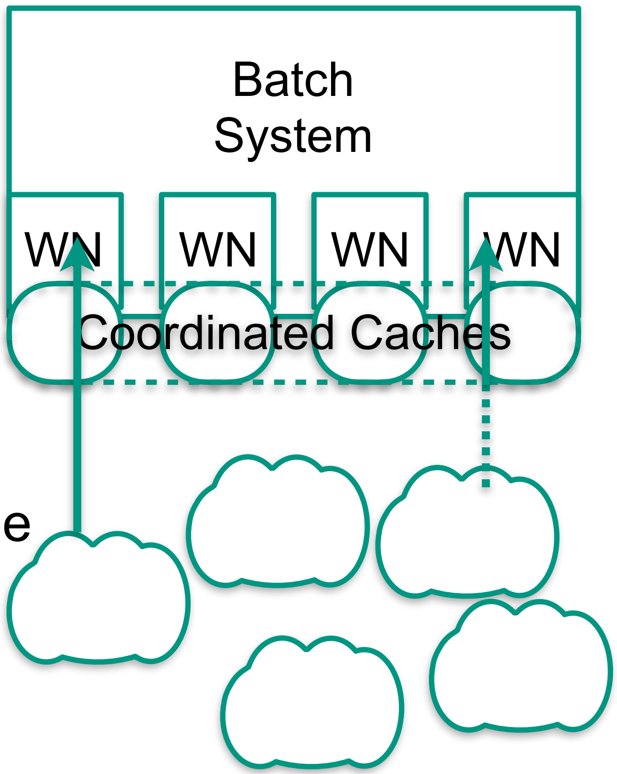
Outlook: Applicability to other setups

- Shared cache for multiple workers
  - Cache volume shared across hosts
  - Shared filesystem support via POSIX
  - Tweaks to location meta-data format



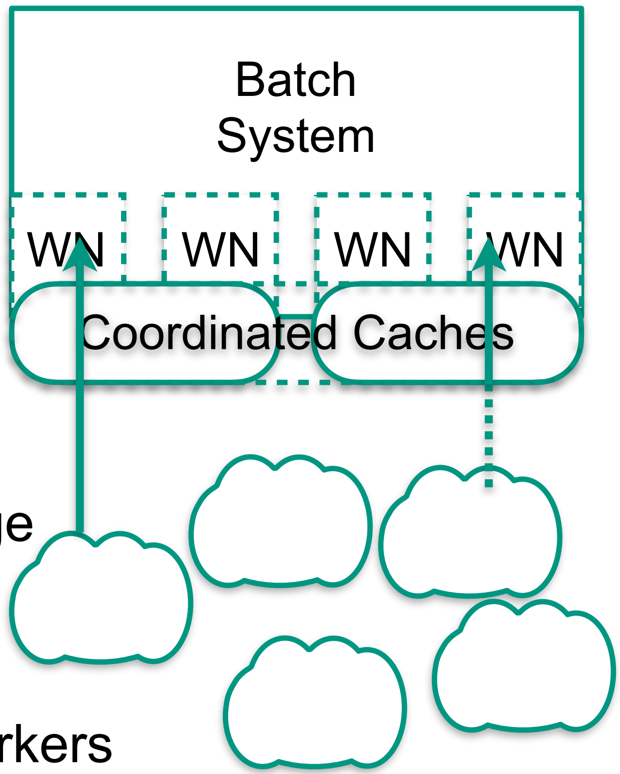
Outlook: Applicability to other setups

- Shared cache for multiple workers
  - Cache volume shared across hosts
  - Shared filesystem support via POSIX
  - Tweaks to location meta-data format
- Diskless Tier3 with attached cache
  - Same logical setup, other protocols
  - Pluggable backends could support xRootD
  - Tune to optimize local/remote resource usage



Outlook: Applicability to other setups

- Shared cache for multiple workers
  - Cache volume shared across hosts
  - Shared filesystem support via POSIX
  - Tweaks to location meta-data format
- Diskless Tier3 with attached cache
  - Same logical setup, other protocols
  - Pluggable backends could support xRootD
  - Tune to optimize local/remote resource usage
- Opportunistic Data Analysis
  - Semi-persistent cache shared by volatile workers
  - Support for volatile nodes using persistent meta-data
  - Combine shared cache with diskless setup



Summary

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

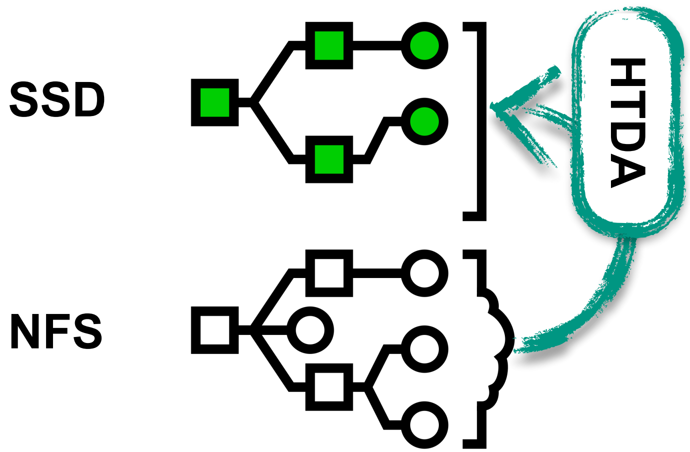
- Prototype Implementation: HTDA
  - Proof of principle, all major features covered
  - Room for improvements and extensions
  - Already considerable performance improvements

- Applicable to other setups
  - Shared caches via parallel filesystems
  - Cache-only Tier3 without dedicated storage
  - Persistent cache for opportunistic resources

BACKUP

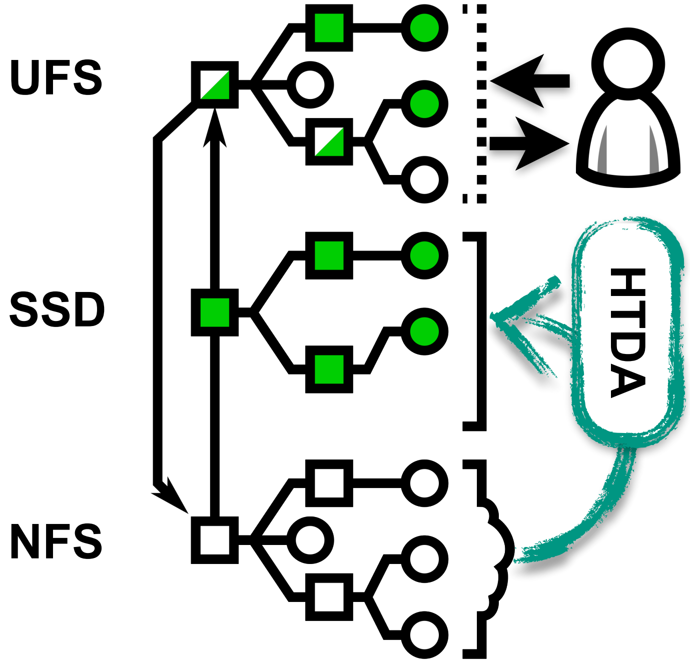
Cache Content Access

- Cache node stages/unstages files according to coordinator request



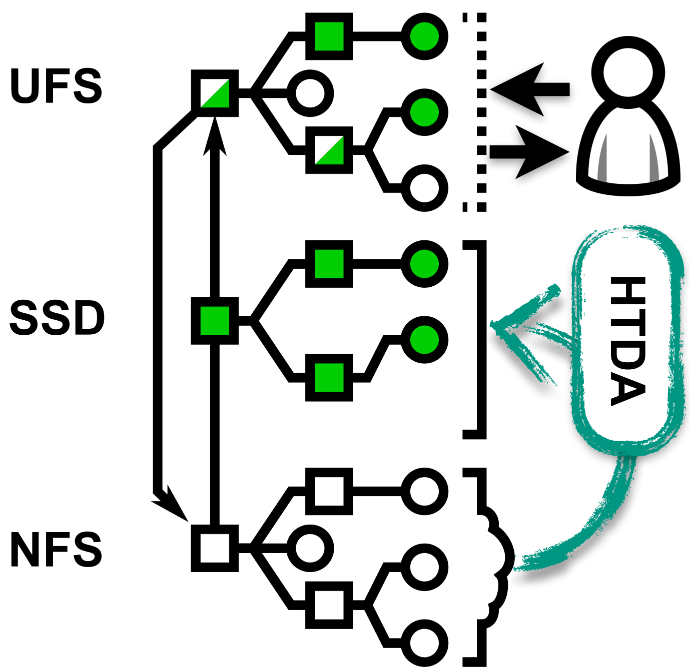
Cache Content Access

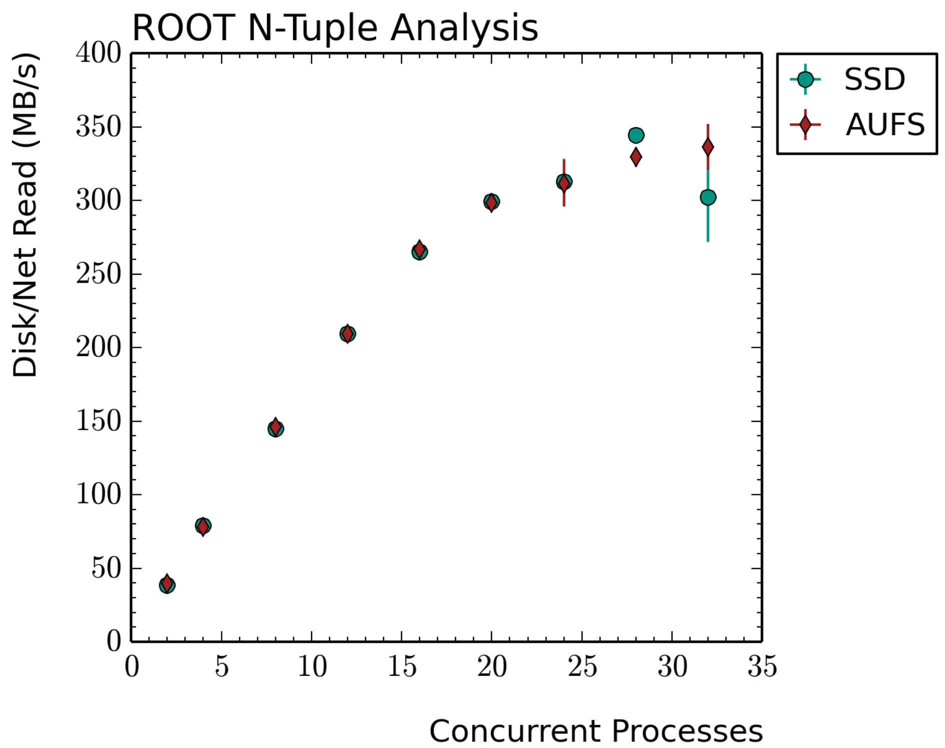
- Cache node stages/unstages files according to coordinator request
- Union File System provides transparent cache access for users



Cache Content Access

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

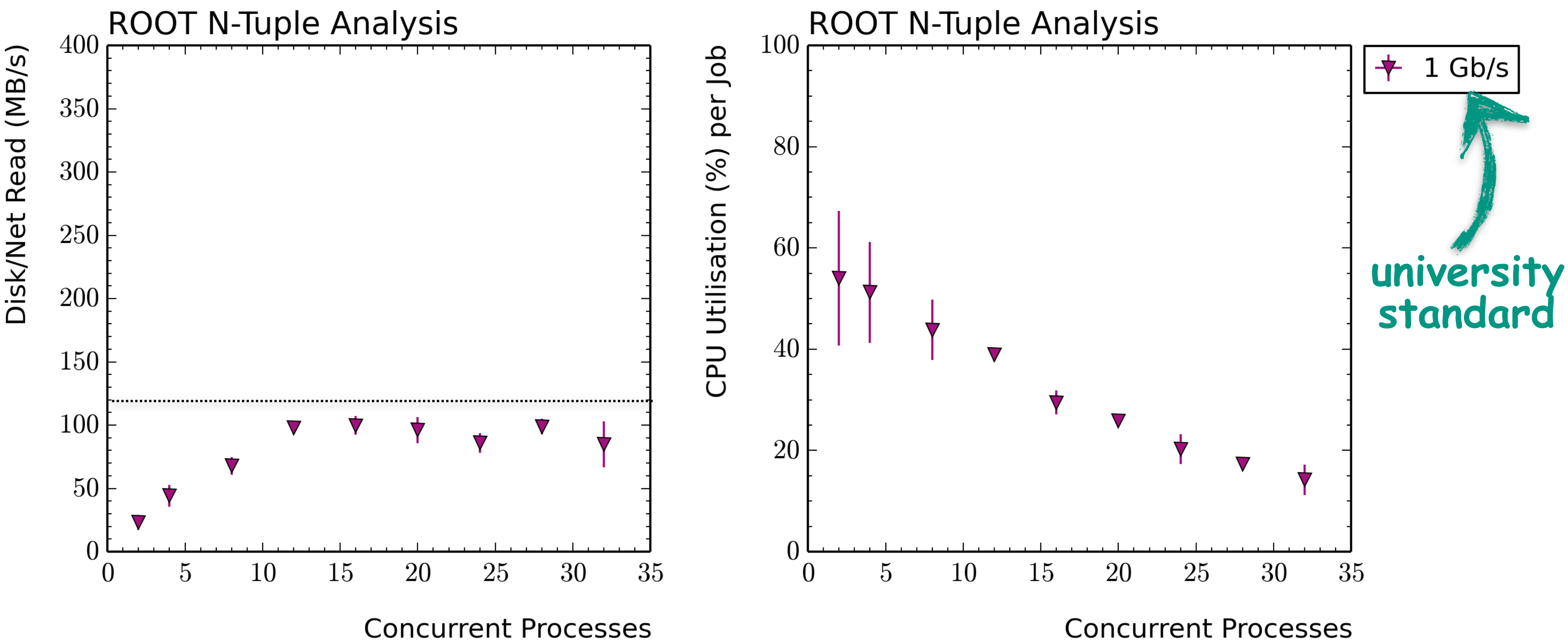




- Lightweight cache access ensures optimal performance

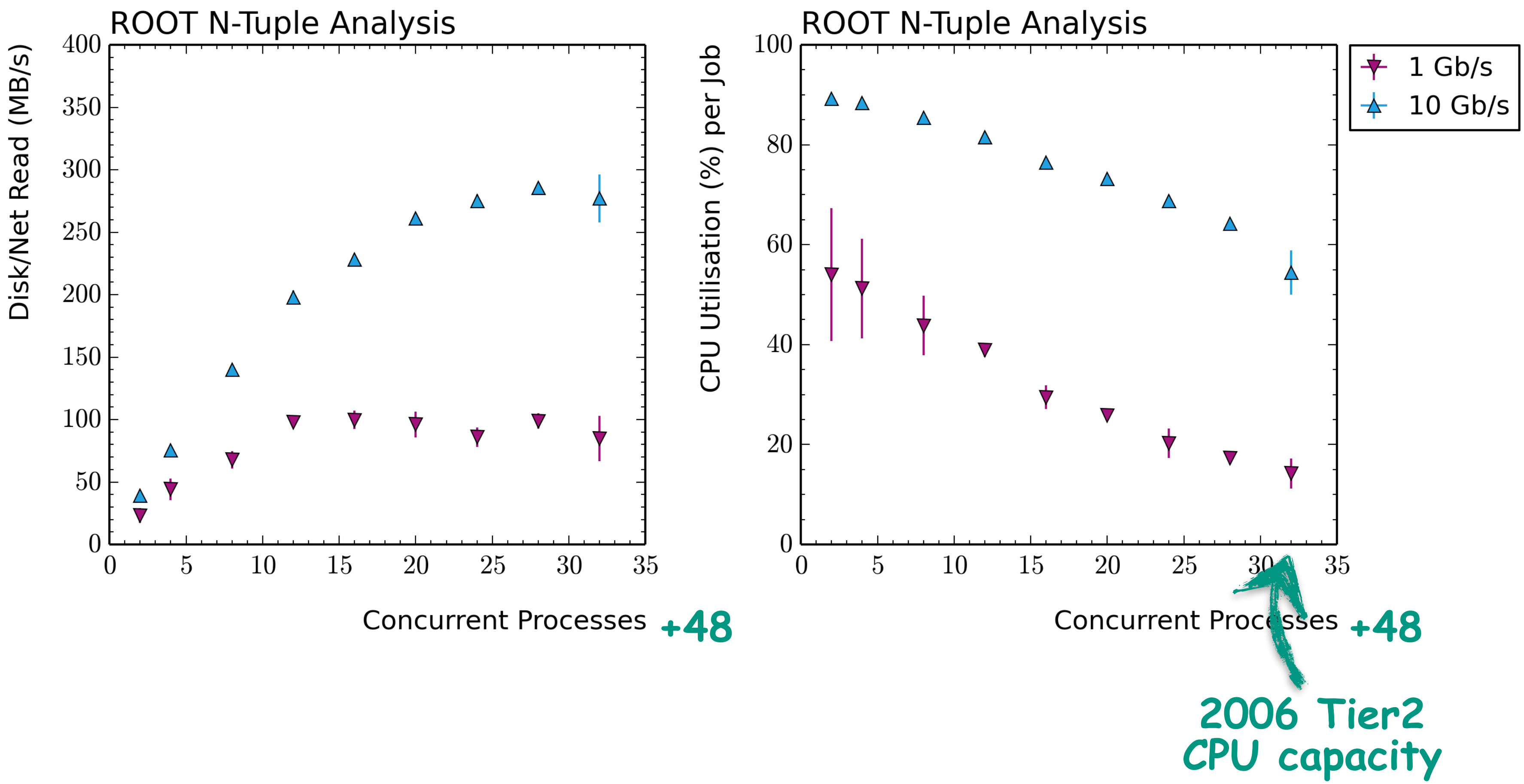
I/O Performance Evaluation

- CMS jet calibration analysis (ROOT n-tuple)



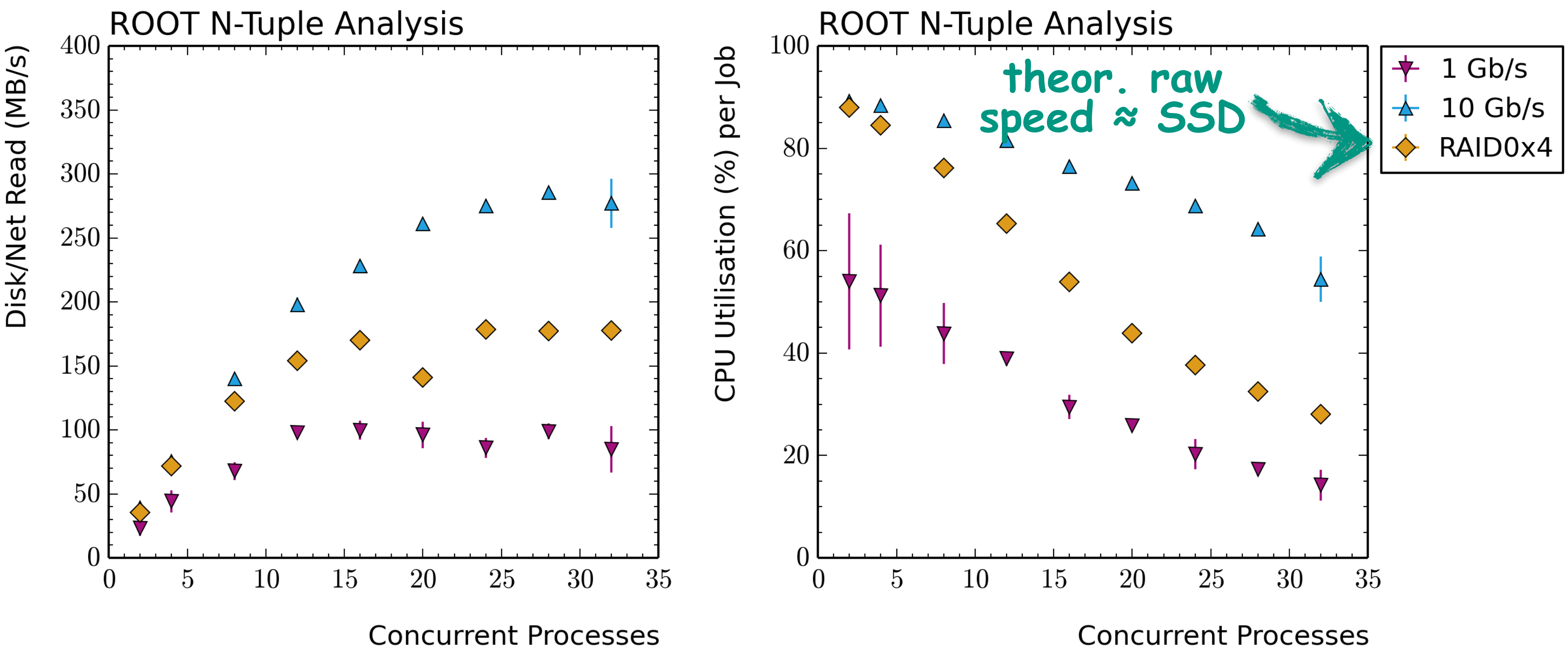
I/O Performance Evaluation

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



I/O Performance Evaluation

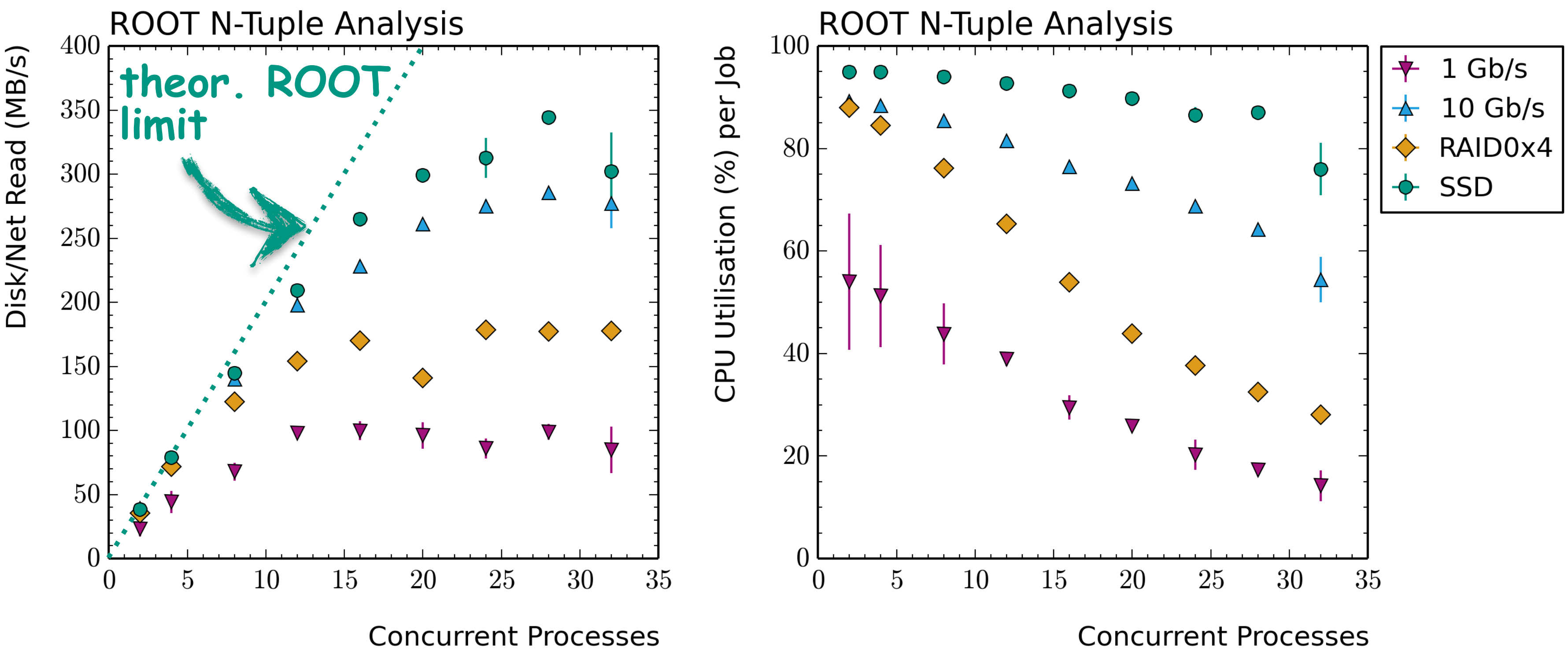
- CMS jet calibration analysis (ROOT n-tuple)
- Additional 48 concurrent reads from other workers for 10 Gb/s test



- 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