WAN Data Access and Caching

Michael Ernst, BNL

LHCOPN/LHCONE Workshop

Washington D.C.

January 31, 2013

Disclaimer

- In talking to ATLAS and CMS Computing Management I found that – at this point – the collaborations are not prepared to quantify their needs for network links/circuits beyond 10 Gbps
- Today the experiments are focusing on the ability of using storage at sites in a federated fashion, meaning running applications at a site accessing transparently (w/o having to replicate complete or partial datasets to the site where the jobs run) data residing at a remote site
  - Latency is (at least) as important as bandwidth
  - Data (re-)ordering in the application (i.e. ROOT layer) to form ~streams/large contiguous data packets is an important prerequisite for reasonable performance
    - I will be using this slot to let you know about progress and ongoing work in this space

WAN Data Access and Caching Motivations and Benefits

- Simplification to data and workflow management; no more choreography in moving data to processing or vice versa
  - Data movement inherent in WAN access/caching, not driven by Distributed Data Management (DDM) system
- More efficient use of storage; reduce replica counts; direct sharing of replicas across sites. Important as storage is an increasingly scarce commodity
- More efficient use of network; fine granularity in moving only the data needed; data moved on demand, only when it is needed
  - Caching avoids redundant copying of same data to same destination, within the cache lifetime
- No replication latency after brokerage decision
- Low support and maintenance load; can be attractive for Tier 3s
- Attractive for cloud utilization; minimal persistent in-cloud storage, minimizes inbound data transfer to the essential
- Escape the protocol/middleware jungle! Data access via standard, efficient, direct protocols

WAN Data Access and Caching Viability

- I/O optimization work in ROOT and experiments in recent years makes WAN data access efficient enough to be viable
  - ATLAS: “TTreeCache shown to be of substantial benefit to direct I/O performance, essential for WAN”
- Caching can further improve efficiency
  - Addressable shared cache at destination reduces latency for jobs sharing/reusing data and reduces load on source
  - Supported by ROOT
  - Requires cache awareness in workflow brokerage to drive re-use
  - **Requires cache support infrastructure at processing sites**
- Asynchronous pre-fetch at the client further improves efficiency
  - Reduce effect of irreducible WAN latencies on processing throughput
  - Supported by ROOT (debugged over the summer, ready for serious testing)

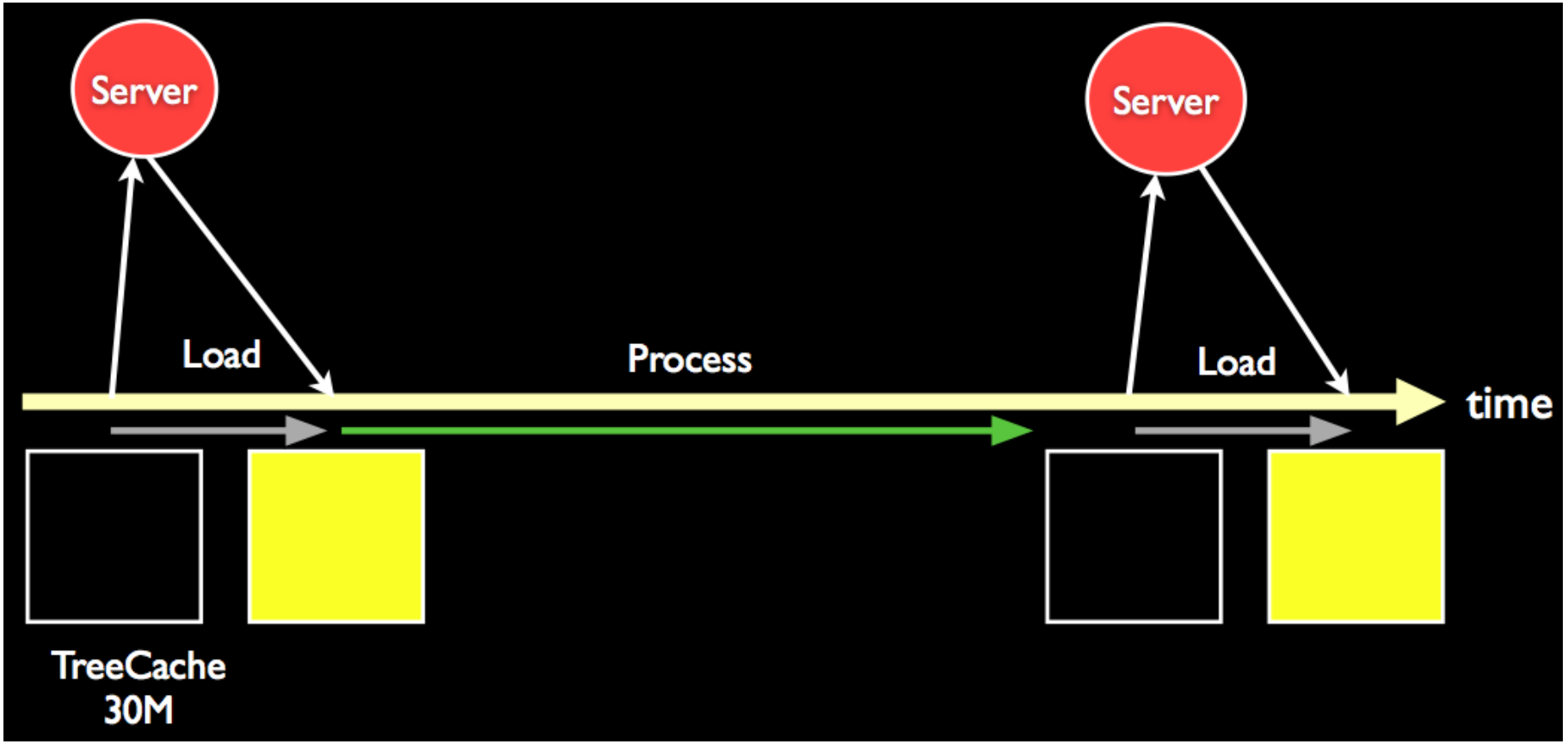
ROOT TTreeCache

- ROOT's TTreeCache optimizes read performance of a TTree by minimizing the number of and ordering of disk reads:
  - In a 'learning phase' it discovers the TBranches that are needed
  - After that, a read on any TBranch will also read the rest of the TBranches in the cache with a single read
- ROOT TTreeCache can have a huge impact on read performance
  - Reduce number of disk-reads by several orders of magnitude
    - Impact depends on system setup and use case
- However, there have been restrictions in the usability of TTreeCache
  - Only one automatic TTreeCache per TFile
    - ATLAS and other experiments use several trees per file (Event data, references, auxiliary transient/persistent converter extensions, …)
  - Slow learning phase, no caching while learning
    - Impediment to activating TTreeCache by default; slow start-up

ROOT Caching and Pre-Fetching Developments

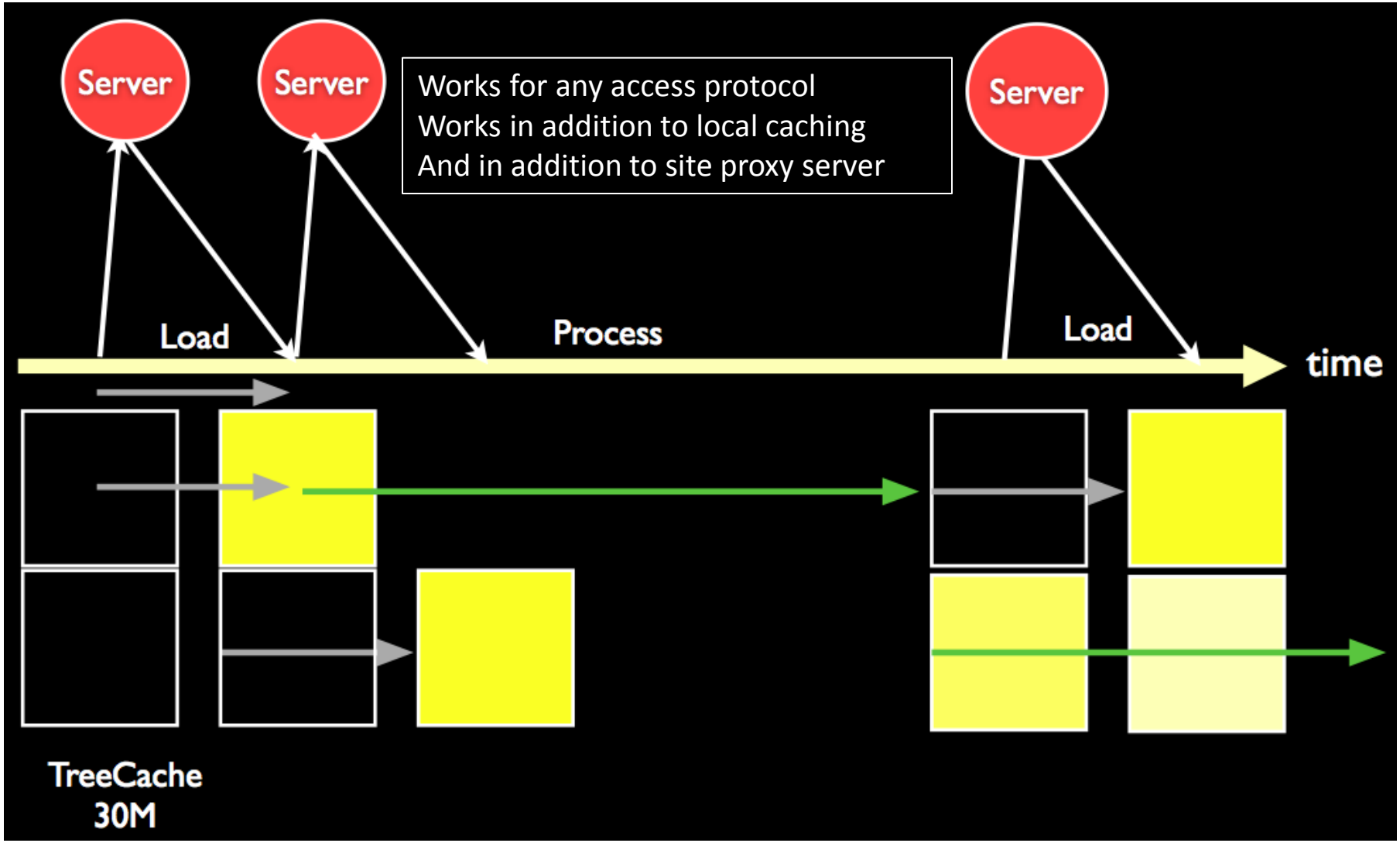
- Support for more than one TTreeCache per file added
  - Needed by and implemented by ATLAS
- Working on turning TTreeCache on by default
  - Ongoing work to improve learning phase by pre-reading all branches (if possible) rather than relying on individual branch reads
- Asynchronous pre-fetching now usable; provides efficient way to improve CPU/runtime efficiency over the WAN
- Caching of TreeCache blocks with reusable block addresses available
  - Enables large latency reductions on reuse (where we can achieve the reuse)

ROOT Synchronous Tree Processing

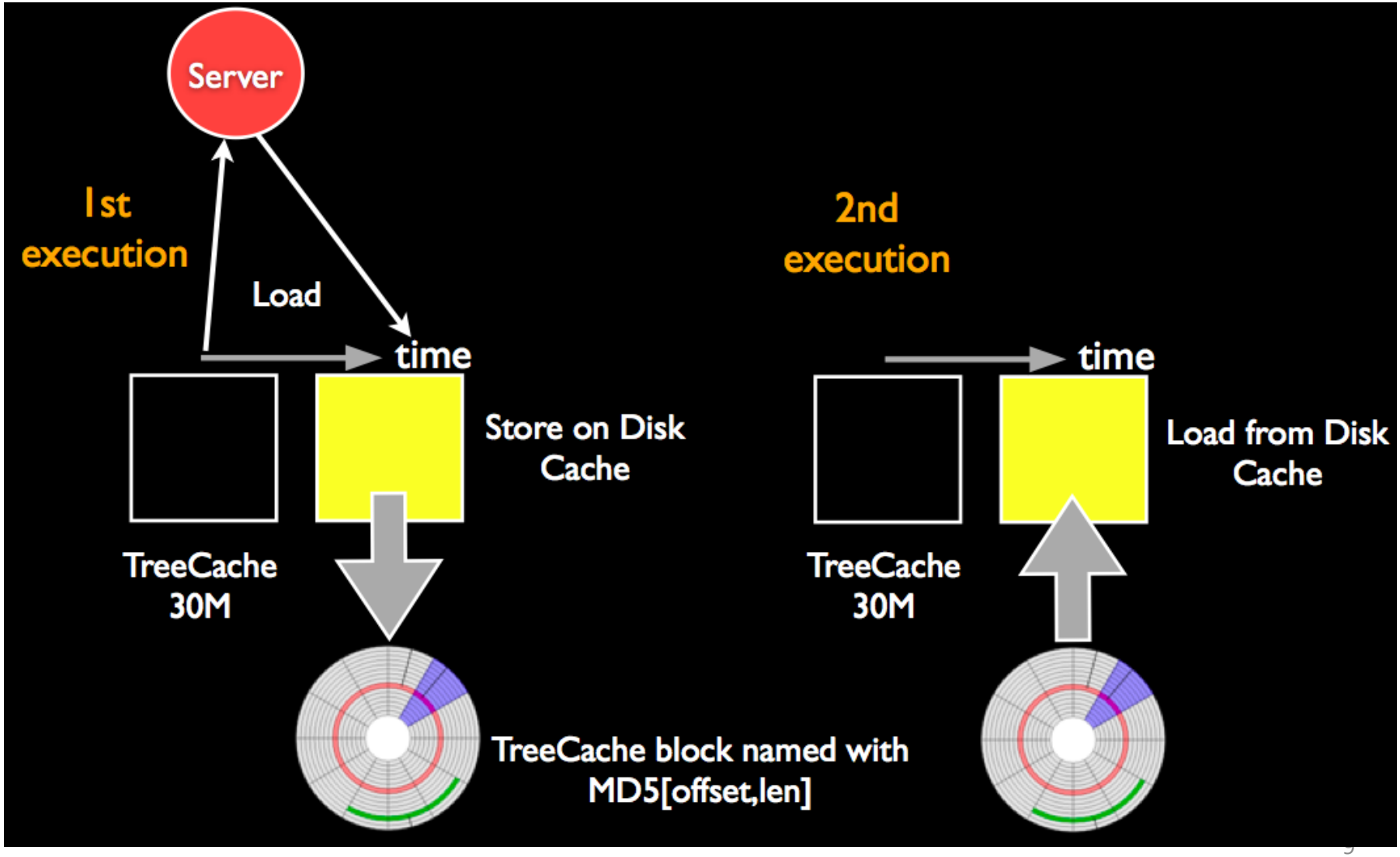


Fons Rademakers

ROOT Asynchronous Pre-Fetch

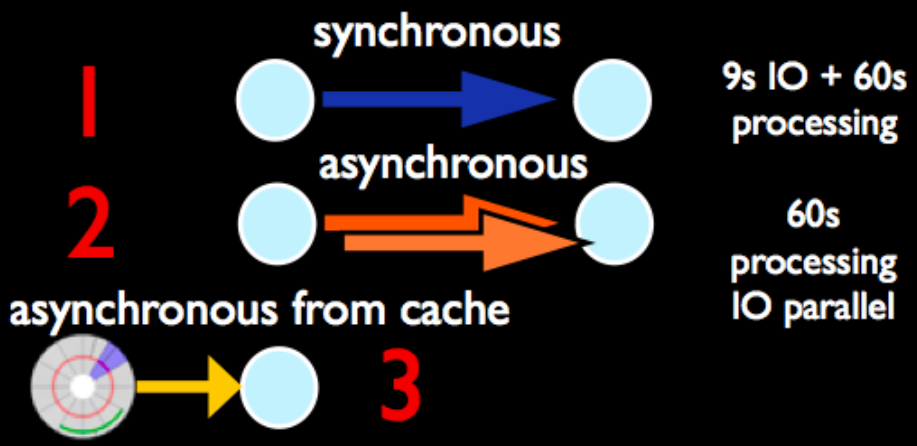


Caching of TreeCache Blocks



Test Results

(reading an 1 GB ATLAS AOD file over http)

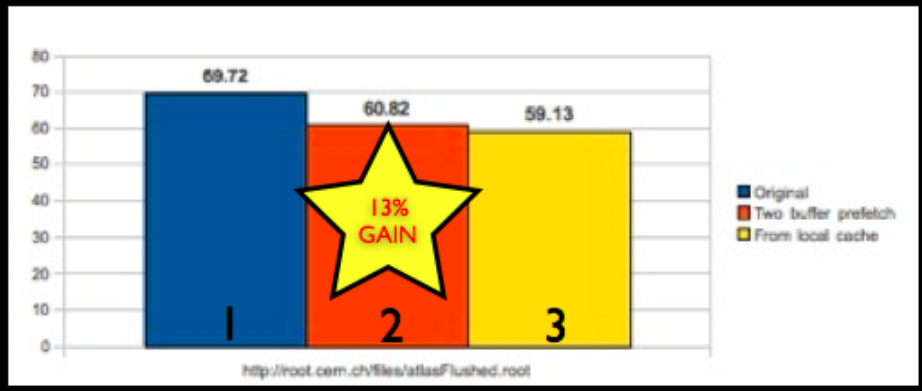


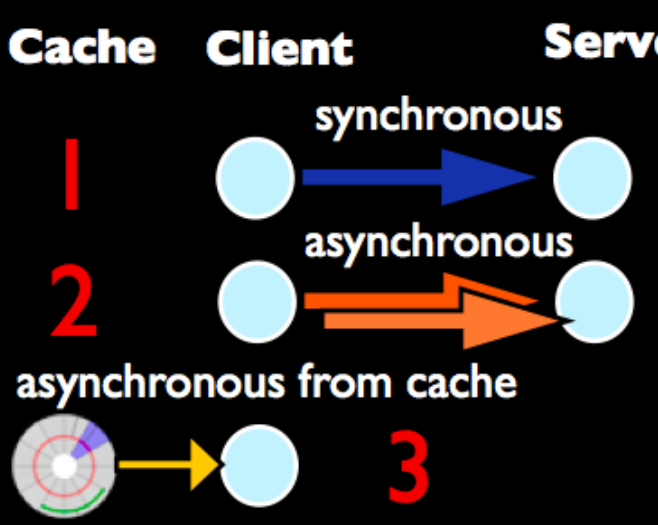
9s IO + 60s processing

60s processing IO parallel

LAN

Server@CERN



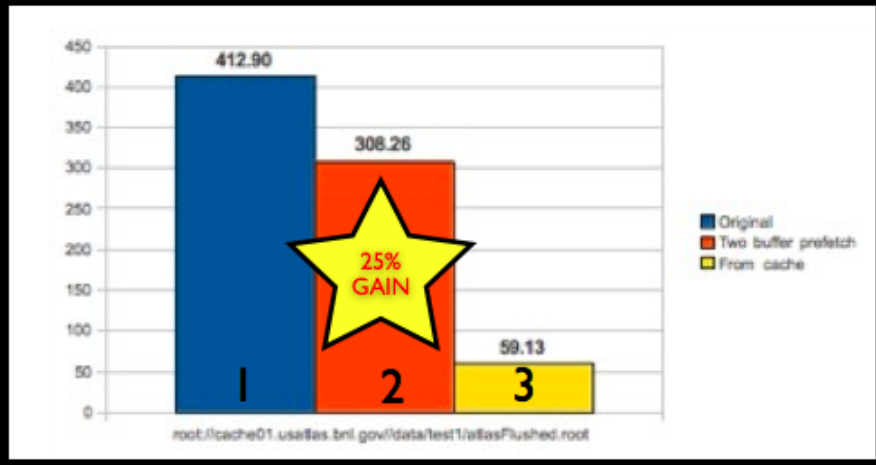


352s IO + 60s processing

258s IO wait* + 60s processing IO parallel

WAN

Server@BNL



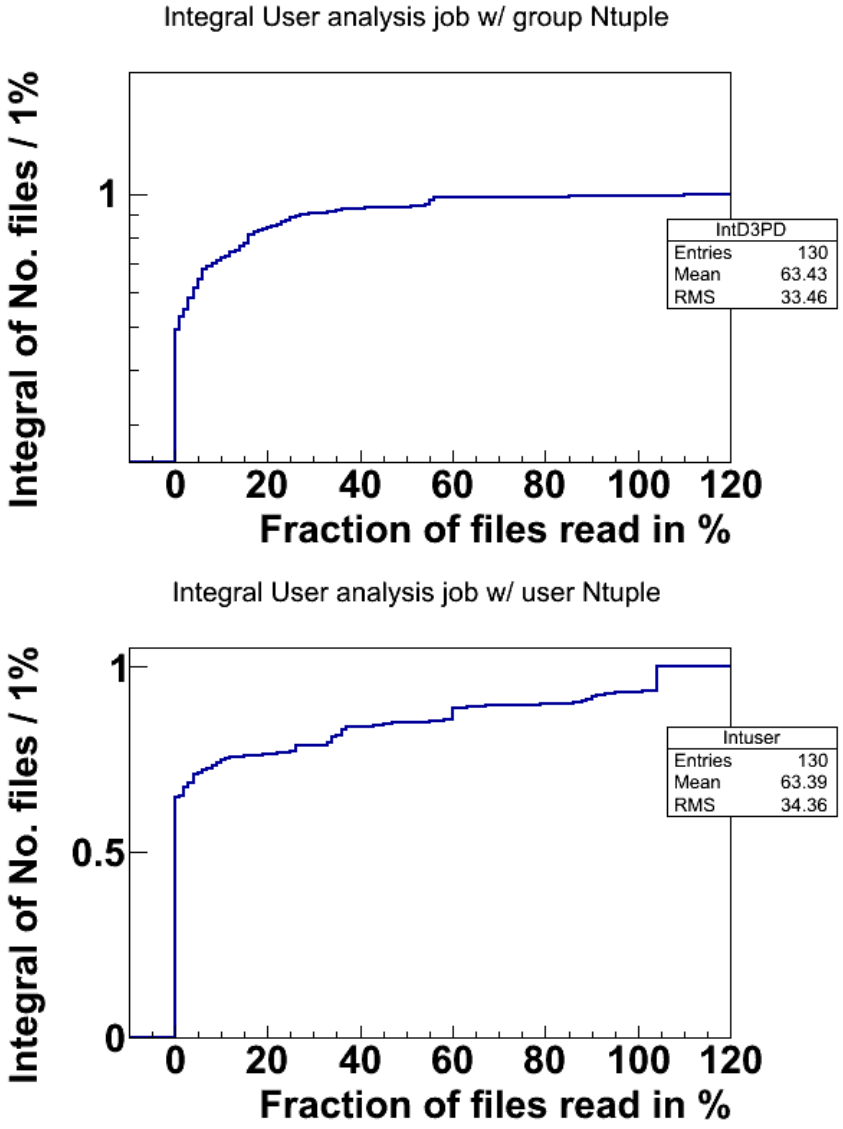
* although the IO is asynchronous we are limited by the available bandwidth

Fons Rademakers, Dirk Duellmann

ATLAS Event I/O

- Key enabler for WAN data access: efficient event data I/O with minimal transactions between application and storage
- ROOT TTreeCache provides the necessary foundation for achieving this
- ATLAS Event I/O optimizations have allowed us to reap the benefits
  - Introduced in release 17 (current production release) newly optimized ROOT storage layout for RDO, ESD, AOD to better match the transient event store and support single event reads more efficiently
  - Events now grouped (up to ~10) contiguously in baskets, minimizing I/O transactions for event retrieval (previously events were split among baskets)
- ATLAS conclusion in working with the new format and TTreeCache: **"TTreeCache shown to be of substantial benefit to direct I/O performance, essential for WAN"**
- Wall time efficiency measures to date show 50-80% efficiency, increasing with increasing TTreeCache buffer size

WAN access vs. Copy: Fraction of Files Read

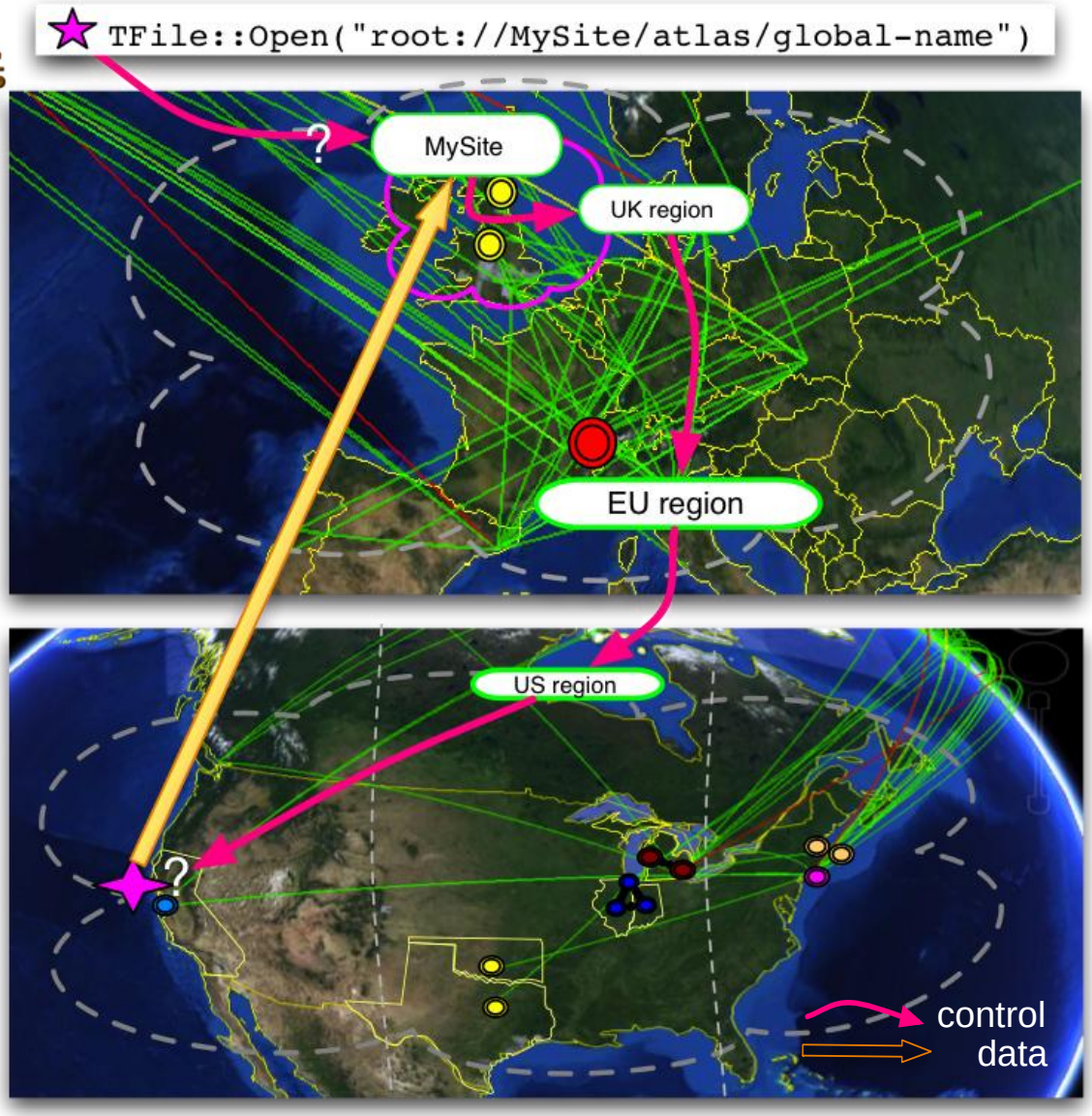


- WAN access vs. copying locally pays for fractional file reads
- What fraction of files are actually read?
- Depends on many details – the analysis, the format – but studies now being made; here is one (from dCache logs)
- 85% of all reads of group D3PD's read less than 20% of file
- 76% of all reads of user files read less than 20% of file
- There is a lot of unread data, not transferred with WAN access

Doug Benjamin

WAN data access and FAX

- WAN data access may be (is probably?) the killer app for the federation
- Xroot is a well optimized, hardened, scalable data access tool optimized for HEP
- Naturally extended to distributed operation through the global redirector layer
- Provides uniformity of data access (global catalog, uniformity of access protocol including direct access) across the distributed store
- For WAN data access across a plethora of sites with different flavors of storage services, the federation encapsulates the heterogeneity and provides the transparency of data location that matches the transparency of access in WAN data access
  - Eases the implementation of support for WAN access at the higher levels (e.g. ATLAS Workflow Management System, PanDA)
- It provides a natural and capable development sandbox for WAN access, and a production deployment context as well
  - Benefits from the active development going on with FAX, its monitoring, its expansion across ATLAS



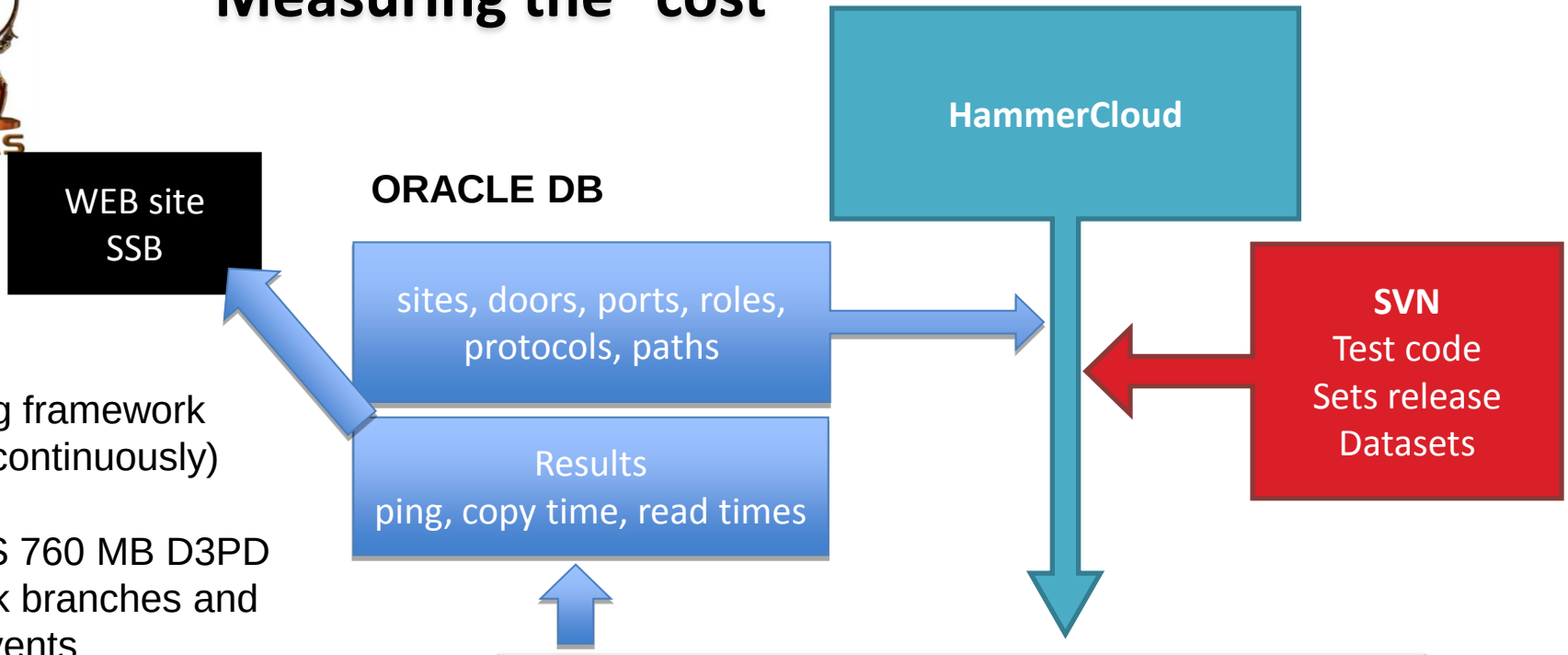
A file is searched locally using the unique global name

If not found at the site the search is expanded to the region using a network of redirectors

File may be copied to the local storage, or read directly over the WAN

Network latency requires intelligent caching by the client if file is directly read

Measuring the “cost”

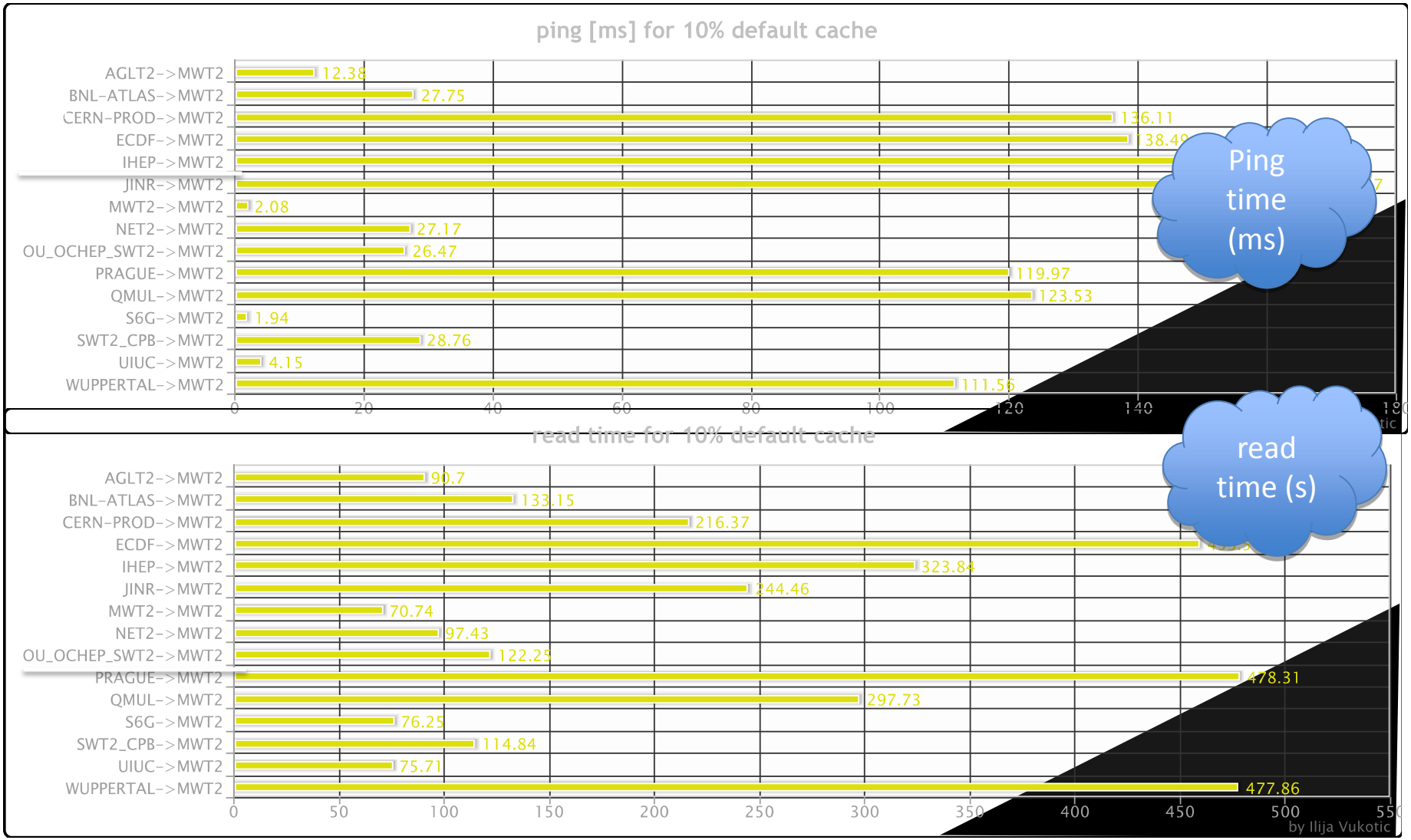


Testing framework
(runs continuously)

ATLAS 760 MB D3PD
with 5k branches and
13k events

- Direct copy
- Read time 10% randomly selected consecutive events using default TTreeCache of 30 MB

Comparing local to wide area performance



xroot federation or http federation?

- WAN data access is a fit with… any federation? which federation(s)?
- Xroot federation draws on long experience with a hardened, scalable, trusted system that supports many storage flavors and is naturally extendible to a distributed federation through the global redirector mechanism
- But what about http, the foundation of the most hardened, scalable distributed computing platform in existence?
- Belated attention now going to http as a data transport protocol for LHC computing
- With http, easy to incorporate hardened, scalable, open source web cache proxies into the architecture, augmenting/complementing other caching approaches
- **At least what we do at the application layer we're aiming at being technology agnostic wherever possible**

A Critical Need: Monitoring

- WAN access in analysis depends crucially on I/O performance, and analysis codes, behaviors are highly variable
- Possible if not probable that users with antique/poorly optimized I/O using WAN access will kill their performance and hold queue slots with very poor CPU/walltime efficiency
- Also, WAN access will be a tricky optimization and tuning problem for some time – important for sites, operations, developers to have clear view into system performance
- Consequently, detailed monitoring of individual and aggregate job performance is critical
- Building blocks mostly in place but must be assembled into effective monitoring enhancements
- Monitoring of underlying infrastructure and site-to-site performance (which feeds into e.g. brokerage decisions) is also vital and in pretty good shape

Event Servers (1)

- With viable WAN data delivery at the event level, we can examine the utility and practicality of going to finer granularity for our processing
- Currently jobs are assigned file(s) to process, whether local or remote
- The real objective is to process tasks: large ensembles of events
- Assigning the work at the file level brings some disadvantages
  - Optimal job partitioning driven by input file size and/or processing time doesn't necessarily (often doesn't) result in optimal file sizes on output
  - Responsibility for file processing can trap a processing thread at a looper event; e.g. one athenaMP process holds up completion and stalls 8 cores
  - Loss of a site or its storage kills a job to the potential loss of a lot of CPU

Event Servers (2)

- Alternative: event servers dispatch events to consumers on WNs
  - Possible event delivery mechanisms: streaming actual events, or (less demanding of new infrastructure and leveraging what we have) sending event addresses/tokens with the WN client doing event retrieval itself
    - Asynchronously retrieving/buffering input events could remove WAN latency
  - On output side, fine grained output handling: send event clusters asynchronously in real time to aggregation site for merging
  - Presents a substantial event-level bookkeeping challenge!
- Event consumers well be suited to time-limited, temporary resources
  - Cheap cloud time on the spot market (factor up to ~10 below baseline cost on EC2)
  - Checkpointable opportunistic resources
  - Borrowed resources requiring exit on short notice
  - Lower latency in liberating prod resources to absorb analysis spikes

Infrastructure Issues

- Monitoring, monitoring, monitoring
  - Including transmitting performance information to end users to drive self-motivated optimization
- Data serving issues with WAN access (ie random read) replacing (some) file replication (ie serial read) activity
  - Random-optimized SSD front end cache to protect the storage from (local and remote) random access
  - Difficult optimization to create an affordable effective cache with significant reuse (SLAC, DESY)
- WAN access is utterly dependent on network performance and latency – PanDA was originally designed specifically to avoid these dependencies
  - Workflows with this dependency are new territory both for facilities and for PanDA, will be a learning experience!
- Wider scope of facilities we can potentially utilize, particularly in longer term if/when we have event serving
  - Opportunistic resources with a light footprint (minimal data flow, flexibly short usage duration)… farms where we have to shrink usage fast, with checkpointing, with variable availability, inherently ephemeral (eg. Boinc), …

Summary

- WAN access can bring greater efficiencies to resource utilization
  - Improved storage efficiency through greater direct data sharing and fewer replicas
  - Improved processing efficiency by broadening the pool of eligible processing sites for a given workload
- ROOT and experiment level I/O improvements and optimizations have demonstrably made WAN data access viable
  - All the more so when augmented by caching schemes if they can be made effective through sufficient reuse
- FAX provides a natural, capable context for WAN access
  - FAX over ROOT I/O provides WAN-capable optimized access
  - Uniformity of access protocol across sites simplifies the implementation at the application and WMS (PanDA) levels