Evolution of data management and data access: R+D activities

Xavier Espinal (CERN-IT)

How future looks like for storage?

- High Luminosity LHC and the upcoming Astrophysics, Cosmology and Neutrino experiments face a multi-Exabyte data challenge
- The storage and compute needs will be a factor 10 above what can be achieved by
  - Technology evolution, typically estimated at 20% /year (but actual figures are less optimistic)
  - Funding based on a flat budget scenario
- Need to **rethink** the way data is stored, managed and accessed

What is currently happening?

- Experiments are looking at **reduced data models**. With emphasis on compact datasets for user analysis
  - As an example: mini/nano AODs in CMS and light AODs in ATLAS
- Several sites exploring **storage consolidation** at the level of national initiatives: UK, ES, FR, IT, DE and US
- Studies ongoing to **understand the usage** of the current storage infrastructure: simulations and measurements
  - Impact of read-ahead caches
  - Data access patterns and storage space usage

Where do we start?

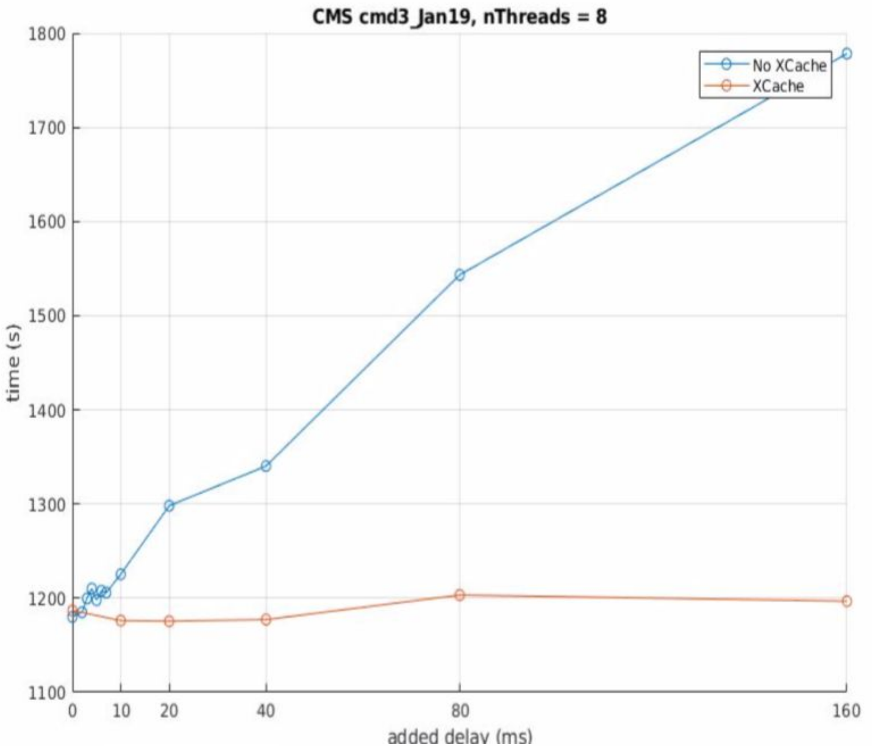
- Studies and observations point us to look into **two main directions**:
  - **Disk caching proxies** (XCache) are able to efficiently hide latency:
    - Are these read-ahead caches good candidate to optimise data access ?
      - Remote IO? no need to replicate data on local batch systems for processing?
      - Provide caching for reusable data, will it be beneficial?
    - Stateless storage is simple to operate and maintain, will this help reducing operational load on sites?
      - Some sites can opt for a lighter storage infrastructure and increase computing resources provisioning, is this reasonable?
  - **Storage consolidation (datalakes)** is a candidate to optimise storage resources, operations and access
    - Storage orchestration and distributed redundancy
    - Definition and implementation of storage Quality of Service (QoS)
    - Efficient data distribution, protocols and networking
- Why these could be good ideas?

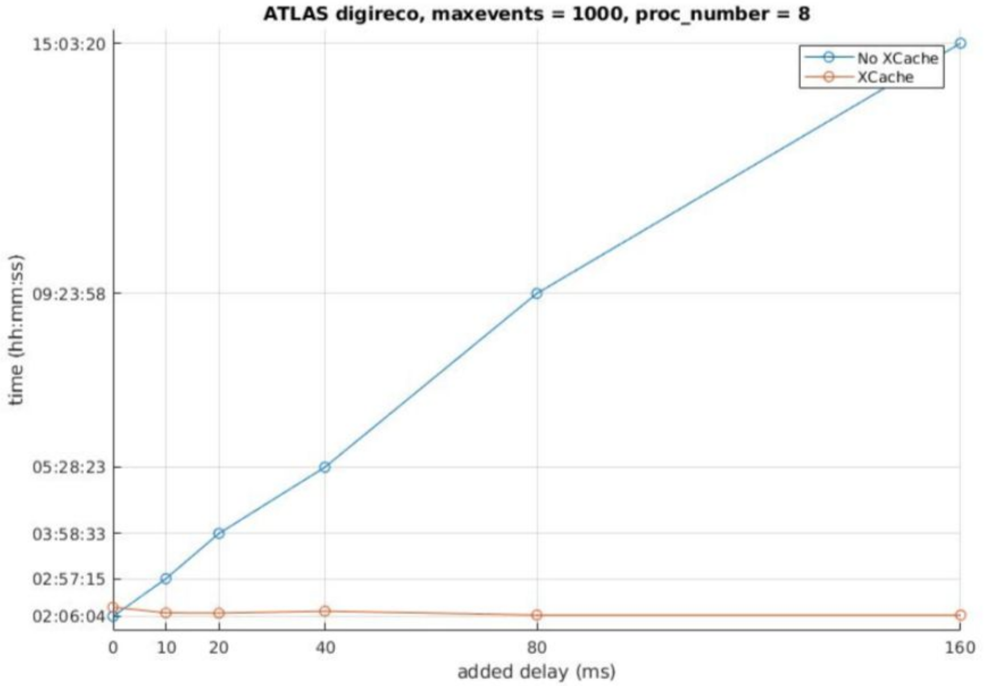
Read-ahead Caches: latency hiding

- Caches with read ahead can efficiently **hide latency** for up to O(100ms)
  - We studied this in:
    - Laboratory environments emulating latency and running experiment workloads
    - Read-ahead cache testing: adding latency, limiting bandwidth, etc.
    - Production environments: topologically close sites, national federations and over long distances
- We focus on XCache technology (readily available and involvement)
  - Deployment for smaller services via containers is straight forward
    - Handful of config parameters governing the read-ahead, ram, blocksize, etc.
  - Good performance demonstrated on modest hardware in a WLCG Tier-2 scale site
  - Deployment on large high performance nodes is more complex and is being studied: SDSC and University of Chicago

Read-ahead Caches: latency hiding

Example: emulation of CMS reco and ATLAS digireco



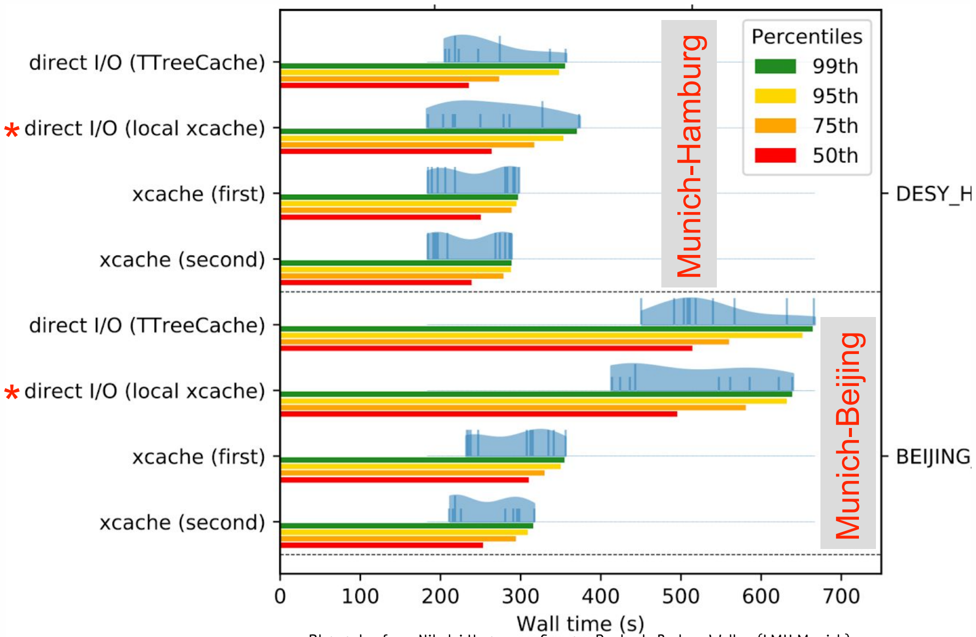


Read-ahead Caches: latency hiding

Example: XCache impact in production

- Realistic analysis job performance evaluated at LRZ-LMU
  - Results show direct access through XCache hides latency as well as ROOT TTC with Asynchronous Prefetch
- Modest hardware: single 2012 old dCache pool
  - Successfully served 3.2k cores through this single disk server with 60TB/Raid-6/3TB disks
  - Jobs running analysis and ATLAS derivations with an average I/O 1MB/s and 3MB/s respectively
- Other production deployments at MWT2, AGLT2 and test deployments at various sites/regions
  - Corner case XCache issues are being addressed
  - Still searching for the best performance configurations (both hardware & software), preliminary base configs ready. WIP.

ATLAS Derivation jobs (500 events, ‹I/O›=3MB/s)



Plot stolen from Nikolai Hartmann, Guenter Duckeck, Rodney Walker (LMU Munich)

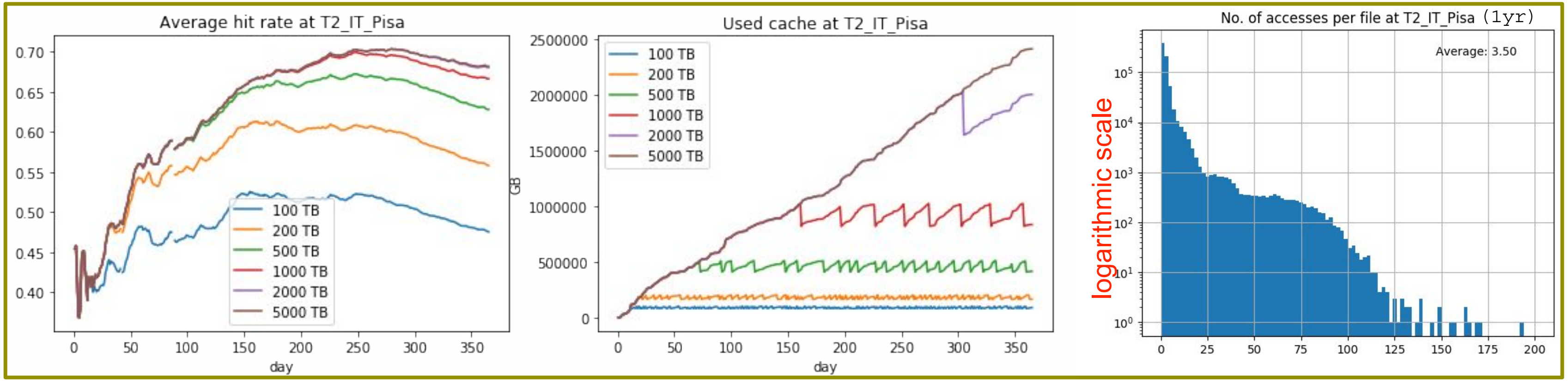
* Read-ahead was disable on XCache by mistake

Read-ahead Caches: storage

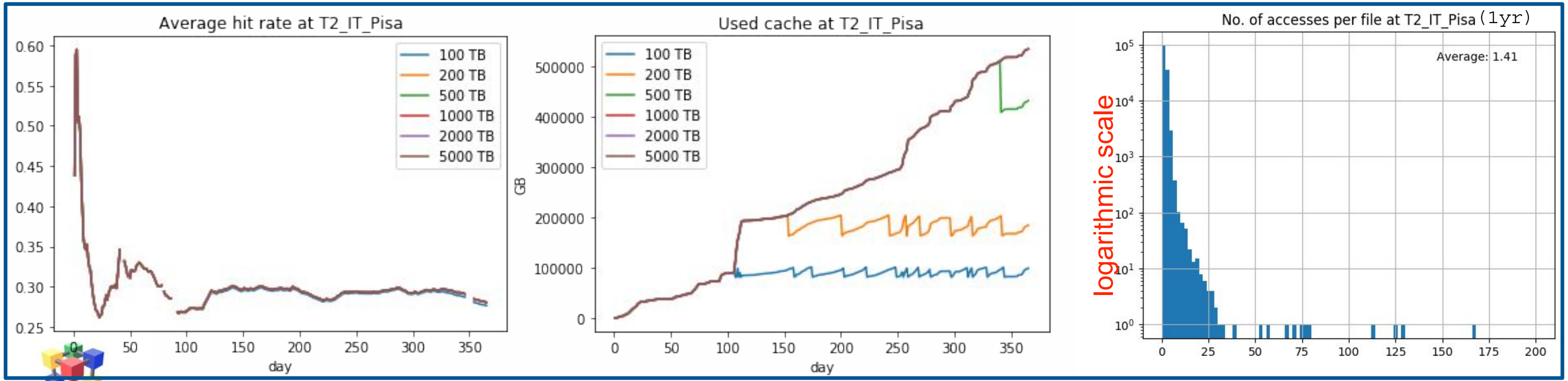
- Cache size and hit rates show typical patterns for the different data types: RAW, AOD, nAOD
  - Hit rates ranging from 40 to 80% for reasonably sized caches (10% of the total storage)
  - Current model of mixing analysis and production data in the same cache is not efficient: read-once prod data pushes analysis data out of the cache
- Experiment managed files are on average read only a few times per year
  - CMS and ATLAS show differences between production and analysis formats: 1.5 to 10 access/yr
- File access rates at T2s are modest on average but can be bursty, as an example (PISA):
  - production: average == 0.01 file/sec and max == 0.08 file/sec (x8 difference)
  - analysis: average == 0.08 file/sec and max == 0.32 file/sec (x4 difference)
- Time between re-reads are relatively short for analysis: read → KO → re-read → OK
- Time between first and last read in the order of 1-2 months globally

Read-ahead Caches: storage

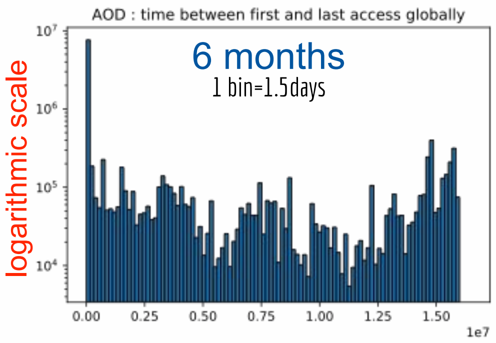
analysis

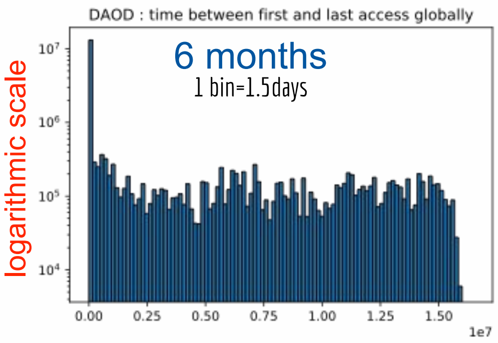


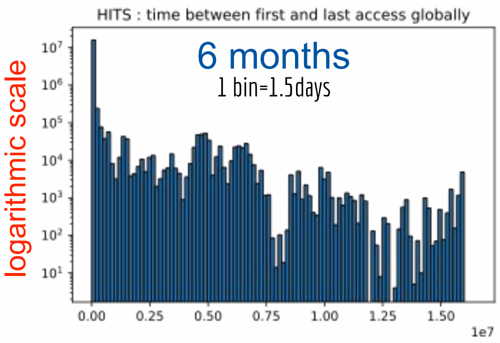
production



Read-ahead Caches: storage







- Data isn't accessed very often, most likely to be re-read within days. Then data gets **cold**.
- These plots provides **hints** but we still miss more substantial analysis to draw conclusions:
  - 6 months cache replay is not long enough, need to enlarge the data set
  - Need to add staging and deletion information
  - Correlate with seasons: conference rushes, holidays, etc.

Current status of the R+D in WLCG DOMA*

- Considering three possible scenarios
  - Evolve the current model
  - Invert the current model for processing workflows
  - Datalake with latency hiding caches

* DOMA stands for Data Organisation, Management and Access

Evolve the current model

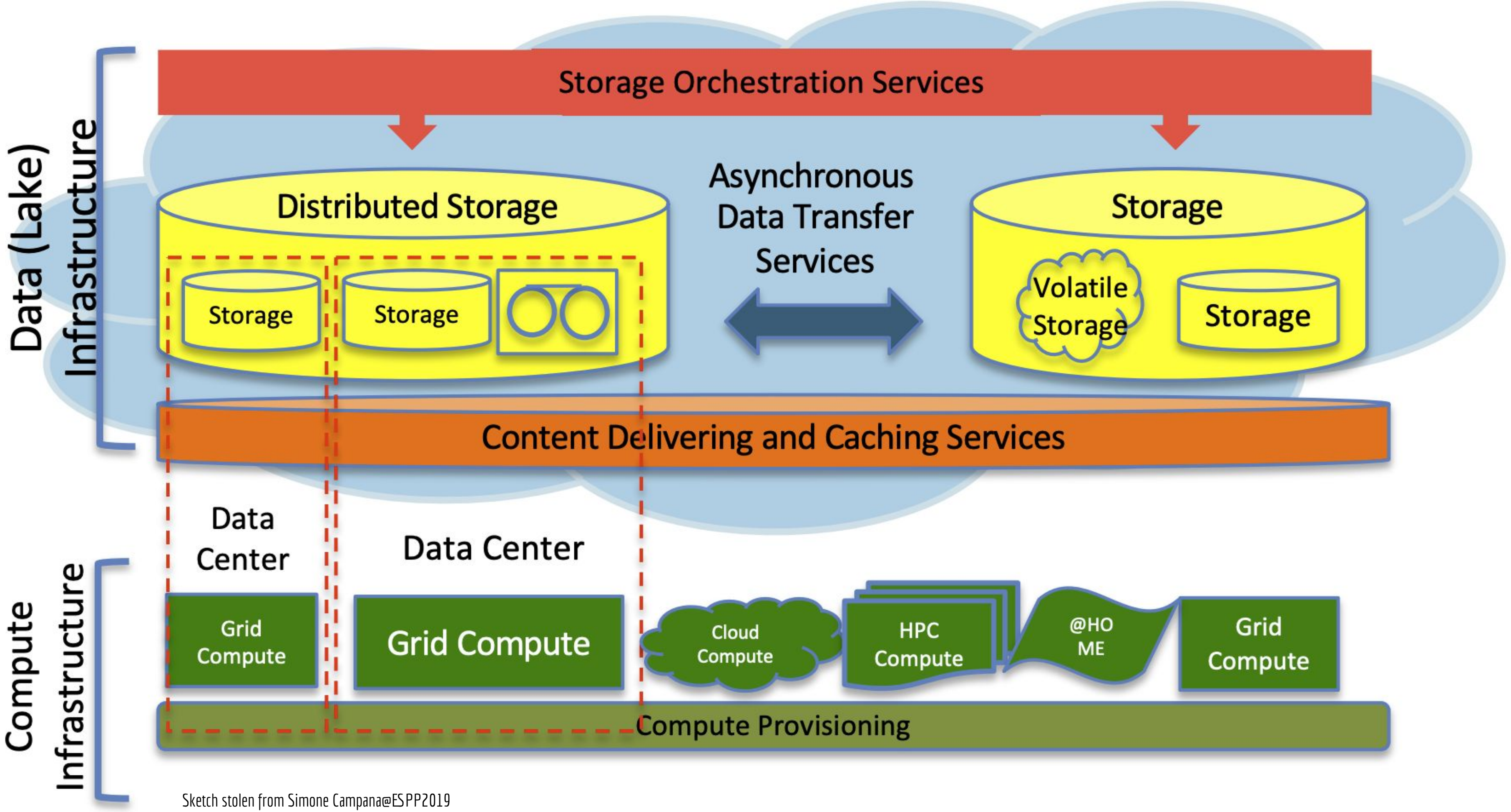
- Continue with the current data processing strategy:
  - Predictable workloads and data intensive processing (ie. reconstruction) at the large centers
  - Unpredictable and user centric data processing (analysis) at the Tier-2s (plus some MC)
- Reducing local redundancy and increase available storage at the same cost
  - Automation of data loss detection and repopulation
  - Adding measures for latency hiding (small cache, staging area, buffers)
- Reducing the time between staging data to a site, the processing and the deletion
  - Bachelor and Master thesis underway to find “optimised” deletion strategy (guess is 40 days)
- Establishing close collaboration at a regional level (small scale federation)
  - Adding caches and sharing some of the storage between sites (US, UK, IT, ES, DE,…)
- No significant reduction in human effort, but automation of recovery might help

Invert the current model

- Revert the current data processing strategy
- The smaller sites with reduced availability requirements handle the production jobs
  - The production is fully automated (retries are cheapish)
  - Data is only accessed “once” → small storage sufficient
- Large sites with high performance storage handle user centric workloads
  - These are difficult to automate, analysis is often time critical
  - User workloads require headroom (bursts of activity)
  - Gaps can be filled with production jobs and effort for different user communities helps to average the load
  - If dedicated analysis facilities are deployed these complex services can be more easily supported
- Potential gains: reduction of scale of storage at smaller sites
  - Less strict requirements on data availability on small sites, especially if a small cache is deployed to handle failover
  - Easy to switch to this model

The datalake

- Data needed by the region is stored at a small number of sites within defined latency domains (as an example: 4 sites within ~20ms RTT)
- Sites providing large scale storage implement QoS endpoints (ie. tape or tape equivalent)
    - All other sites in the region focus on providing processing capabilities
    - Latency hiding read-ahead caches to hold ~1 month analysis needs
    - CVMFS for static data
    - Possibility to increase end user support by providing advanced compute: Machine Learning, GPU, etc.
- The data is replicated between different datalakes
    - Minimal requirements for local redundancy
    - Data accessibility via failover between datalakes
        - Automated replication requires link between storage systems and experiment data management tools
- Some of the benefits:
    - Overall amount of storage can be reduced
    - Operation costs can leverage the economy of scale
    - 5 times larger storage system requires less effort that 5 storage systems



Sketch stolen from Simone Campana@ESPP2019

ESCAPE: Data Infrastructure for Open Science

- The computing centers providing storage resources to the experiments constitute a global storage infrastructure: the datalake
- The datalake, the associated data management tools and the data access models are shared among sciences, scientists and computing centers
- This requires:
  - A global entity that overlays the storage services. A high level data management tool perceived as a single entity: Rucio
  - Common high level characteristics of the storage systems: QoS and protocols
  - Common AAI framework
  - Tool to benchmark and test the infrastructure under real conditions: HammerCloud
- Flexible enough to fit HEP workflows and specially new paradigms in terms of data formats, access patterns and usage from astrophysics and cosmology communities

What is QoS ?

Why storage-QoS?

Eierlegende wollmilchsau

DOMA QoS: the goal

- Implementation of storage Quality of Service (QoS)
  - To **fit data requirements with performance**(=costs)
  - To **allow data "transitions" among different QoS** in time ($$$ to $)

- QoS to leverage:
  - Different **media types**: Tape, "cheap" disks, "enterprise" disks, SSD, …
  - Different **configurations**: JBOD, RAID, RAIN/EC

- This is a complex problem to address. We agreed to describe storage **expectations**, rather than dictate storage **architectures**

DOMA QoS: status

- **Site Survey** has been prepared to identify current QoS-oriented activity and document solutions under evaluation
  - It has been merged with the planned WLCG Ops survey
  - Consists of two parts:
    - Storage: https://forms.gle/mhWPrDfq8n2bDGES9
    - Compute: https://forms.gle/xtyfCM7bDAsJBv4a7

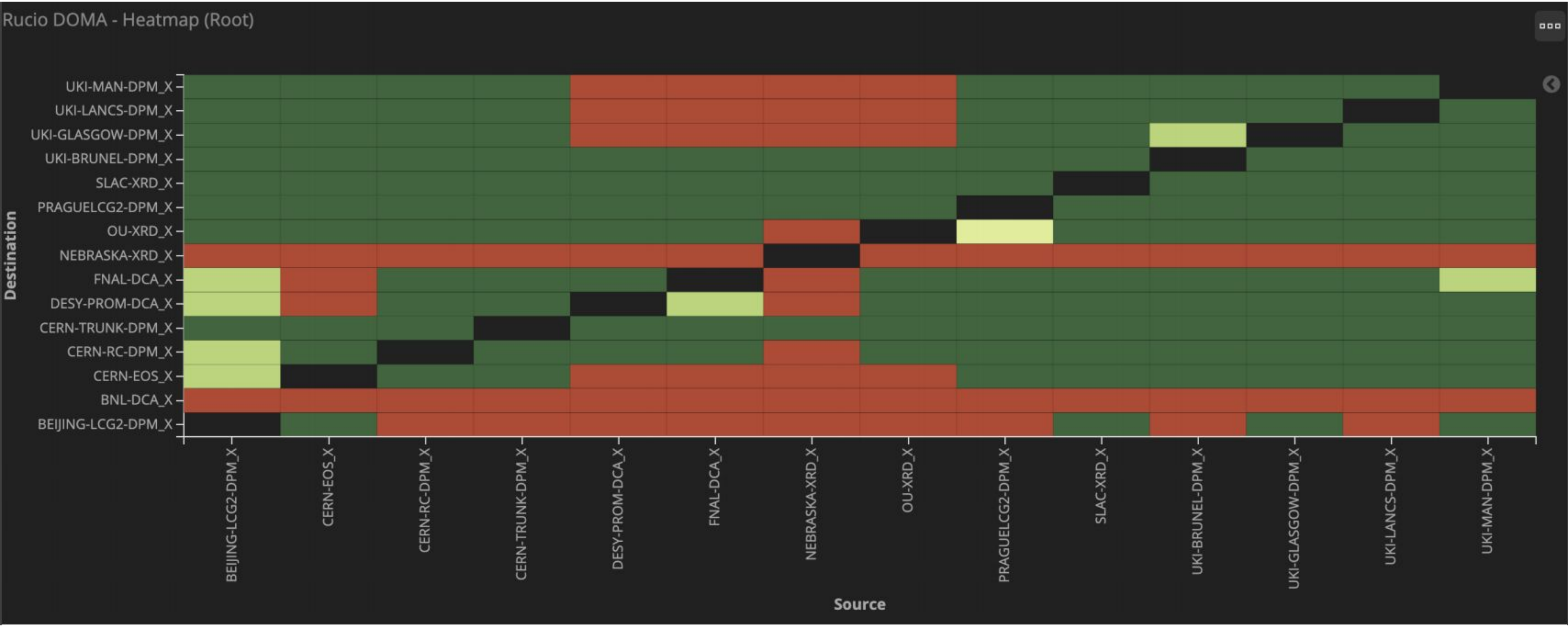
- Discussion on QoS with the **Rucio** team ongoing:
  - How should Rucio interact with storage (interfaces and concepts)
  - What interface/abstraction is seen at the experiment level (e.g. QoS classes)

DOMA TPC: the goal

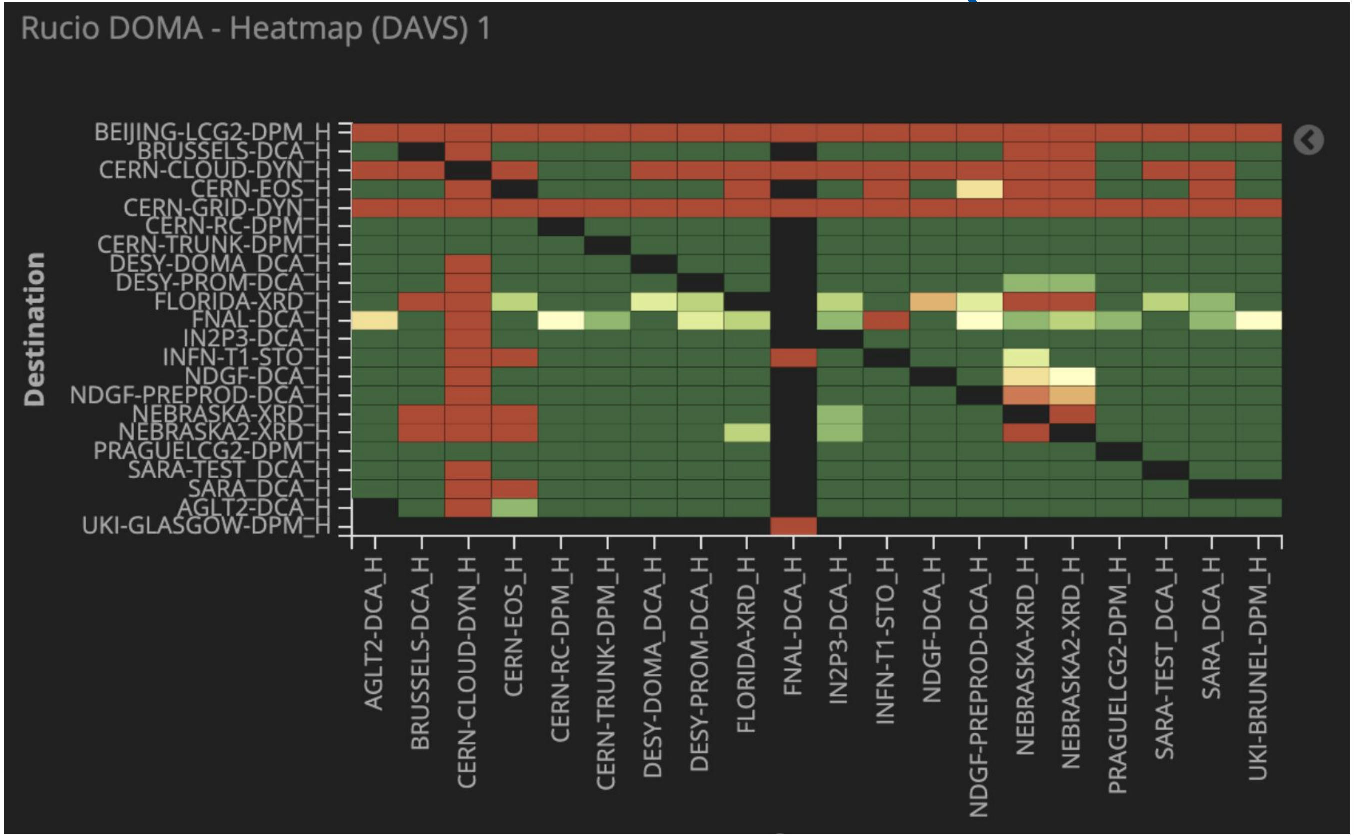
- Biggest data volumes transferred on the grid are third party copy ie. moving data from site A to site B
- **Need to replace** the functionality found in the **Globus Toolkit** (ie. GridFTP and GSI)
- Working group main goal is to find **alternative protocol(s)**, currently focused on **XRootD and http/WebDAV**

- Initial work plan consisting in three phases:
  - **Prototype / implementation**: Demonstrate viability of protocols. Ensure all storage implementations have a valid alternate in production.
  - **Early deployment**: Ensure rollout of alternates at all sites with **>3PB** storage
  - **Widespread deployment**: rollout to remaining WLCG sites

DOMA TPC: results (xrootd)



DOMA TPC: results (webdav)



DOMA TPC: next steps

- Close out “Phase I” and work on “Phase II”

- Asking sites to volunteer to switch over production transfers

Summary and next steps

- We are in **full R+D phase**: exploring new ideas, new concepts and gathering feedback as we advance
- **WLCG DOMA**
    - Access: Exploring the different scenarios through the different prototypes with the goal to produce "cost" estimates (resources, complexity and ops). Test and evaluation of XCache.
    - TPC: Test setup in place. Switch to production transfers with TPC. Go big (sites with >3PB)
    - QoS: Site survey almost ready. Discussion with Rucio about possible implementations of QoS (relation between file metadata and storage endpoints)
- **ESCAPE** - Data Infrastructure for Open Science
    - Building a distributed storage system fitting requirements from the different experiment communities.
    - A datalake composed by several sites with RUCIO as High Level Data Management Tool
    - FTS to orchestrate data replication and http/davix/(xroot) for data access
    - HammerCloud to benchmark the infrastructure