Storage Group Fwd Look

STORAGE

(And Controversy?)

Meta Comment

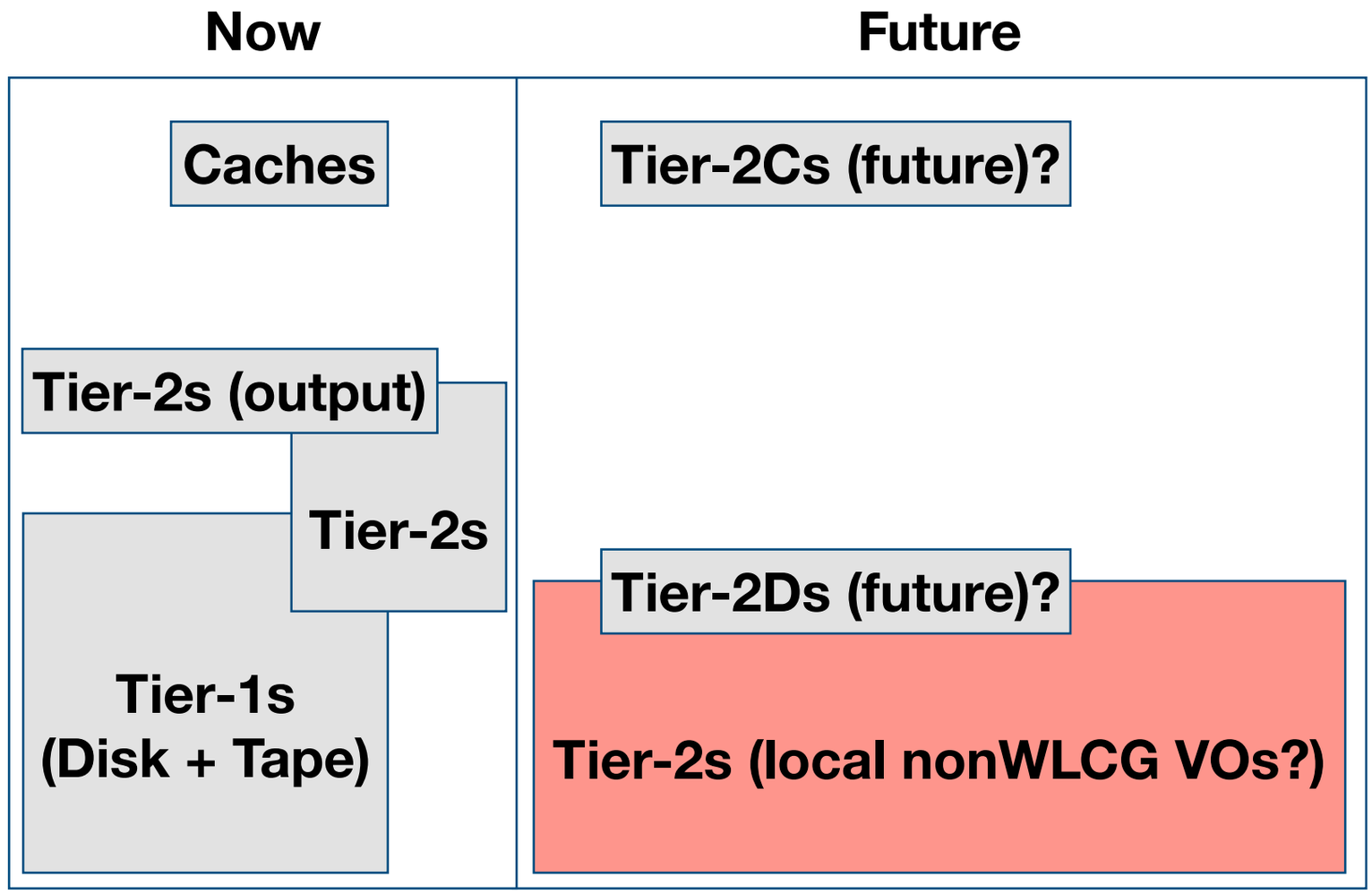
- The Storage Group forward look factorises "Data" into three components:
  - **Data Storage**
  - Data Movement / Access
  - Data Management / Cataloging
- These components are not orthogonal, and hence there are cross-terms.

Data Storage

- "Where you put your data"
- Technologies for:
  - site-level storage of data
  - coordinating/merging servers into storage system
  - site-level metadata
  - local access
  - [interaction with CEs / workflow]

Storage Classes

- Volatile
- Short-term resilient
- Long-term resilient
- Archival



"Grid SEs"

- DPM
- dCache
- StoRM
- EOS
- "plain" XrootD
- Castor/ECHO

Distributed filesystems

- Early grid SEs (DPM and CASTOR, early dCache) managed storage nodes like separate "silos", combining them into a virtual namespace.

- This is done *better* now by a large number of modern standards - Lustre, CEPH(fs), HDFS, GlusterFS, GPFS etc.

- These are also more sophisticated than most GridSEs (can stripe files, erasure code/replicate blocks, provide POSIX interfaces).

- StoRM was/is the "cutdown" grid "shim" solution to this.

Data Lakes

“If you think of a datamart as a store of bottled water – cleansed and packaged and structured for easy consumption – the data lake is a large body of water in a more natural state. The contents of the data lake stream in from a source to fill the lake, and various users of the lake can come to examine, dive in, or take samples.”

James Dixon, 2010

https://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes/

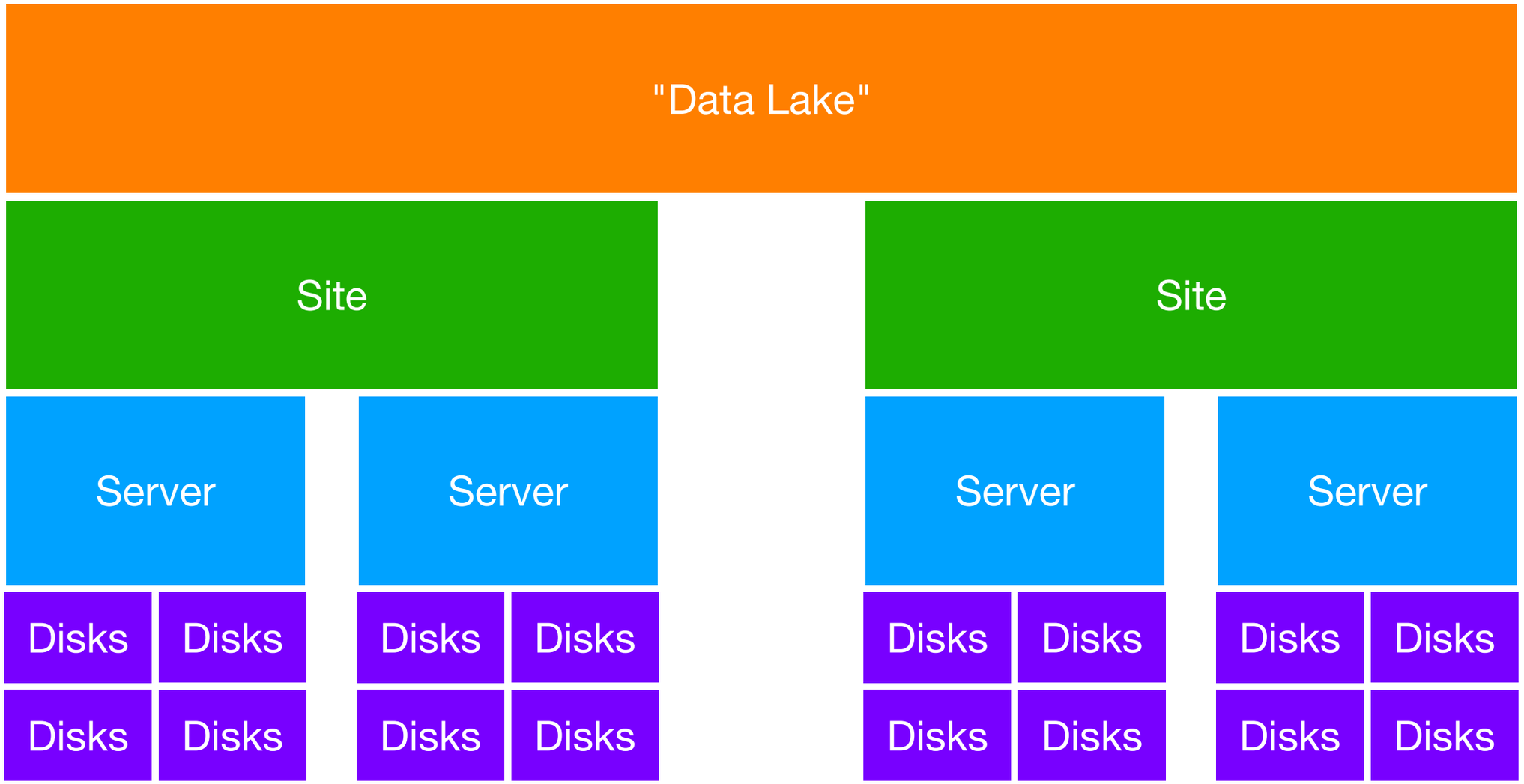
Data Lakes

So… it’s a distributed tape farm?

Data Lakes

- "Data Lake" *now* is an almost pure marketing term.
- In WLCG contexts:
  - a consolidated storage system spanning multiple geographical locations [and presenting one endpoint]
  - *probably* geographically aware, *probably* needs to reconstruct objects on request, *probably* provides different “QoS” levels.
  - eg: distributed EOS; distributed dCache [ ], distributed DPM [Italy, GRIF], Dynafed [UK, Italy], NorduGrid T1…

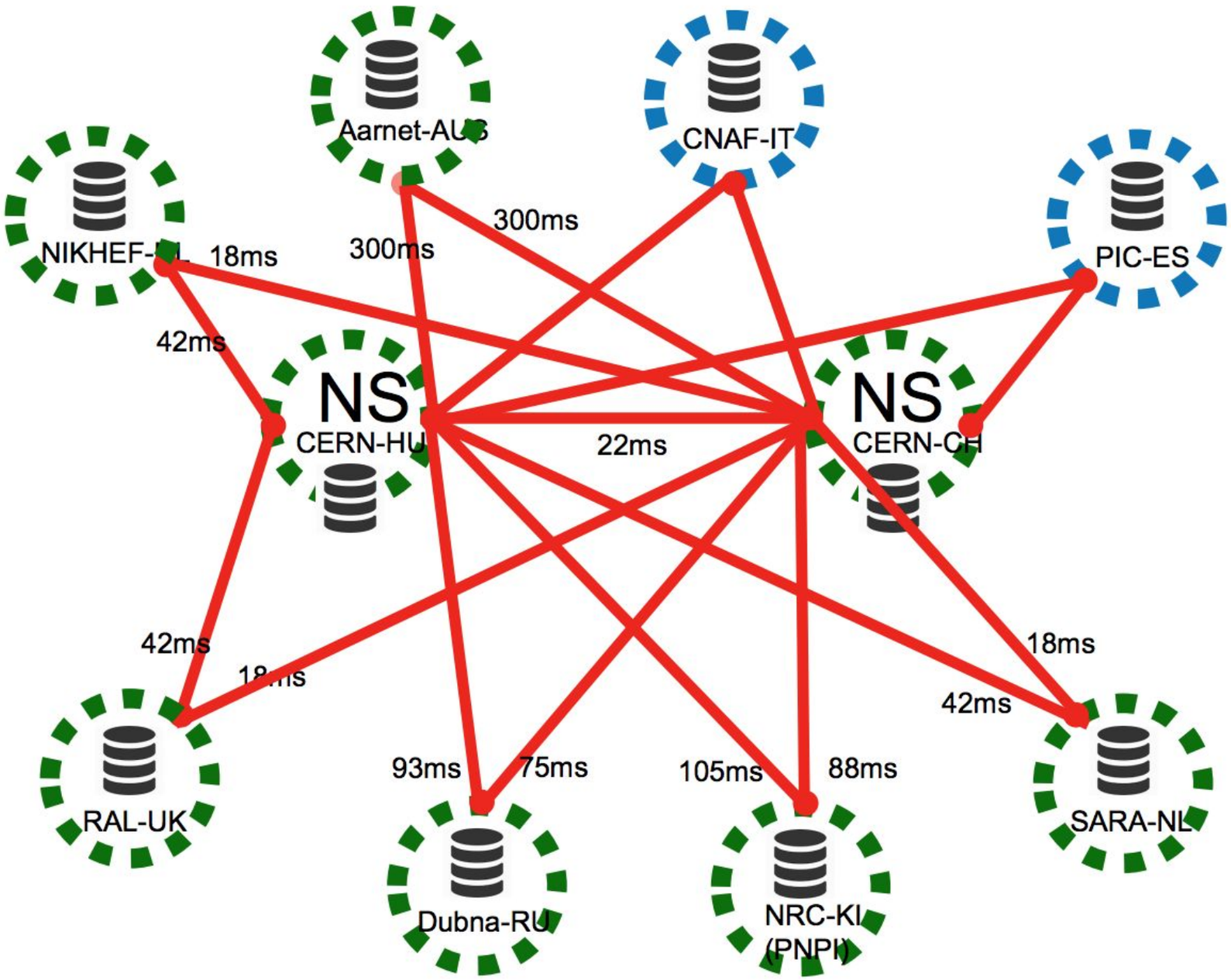
Data Lakes



(Yes, you could mostly replace “Data Lake” with “Federated Storage” here and it would look the same.)

eg: Architecture and prototype of a WLCG data lake for HL-LHC, CHEP 2018
https://indico.cern.ch/event/587955/contributions/2936867/attachments/1680424/2699527/CHEP2018-DataLake.pdf

The eulake prototype



Goal: test and demonstrate some of those ideas

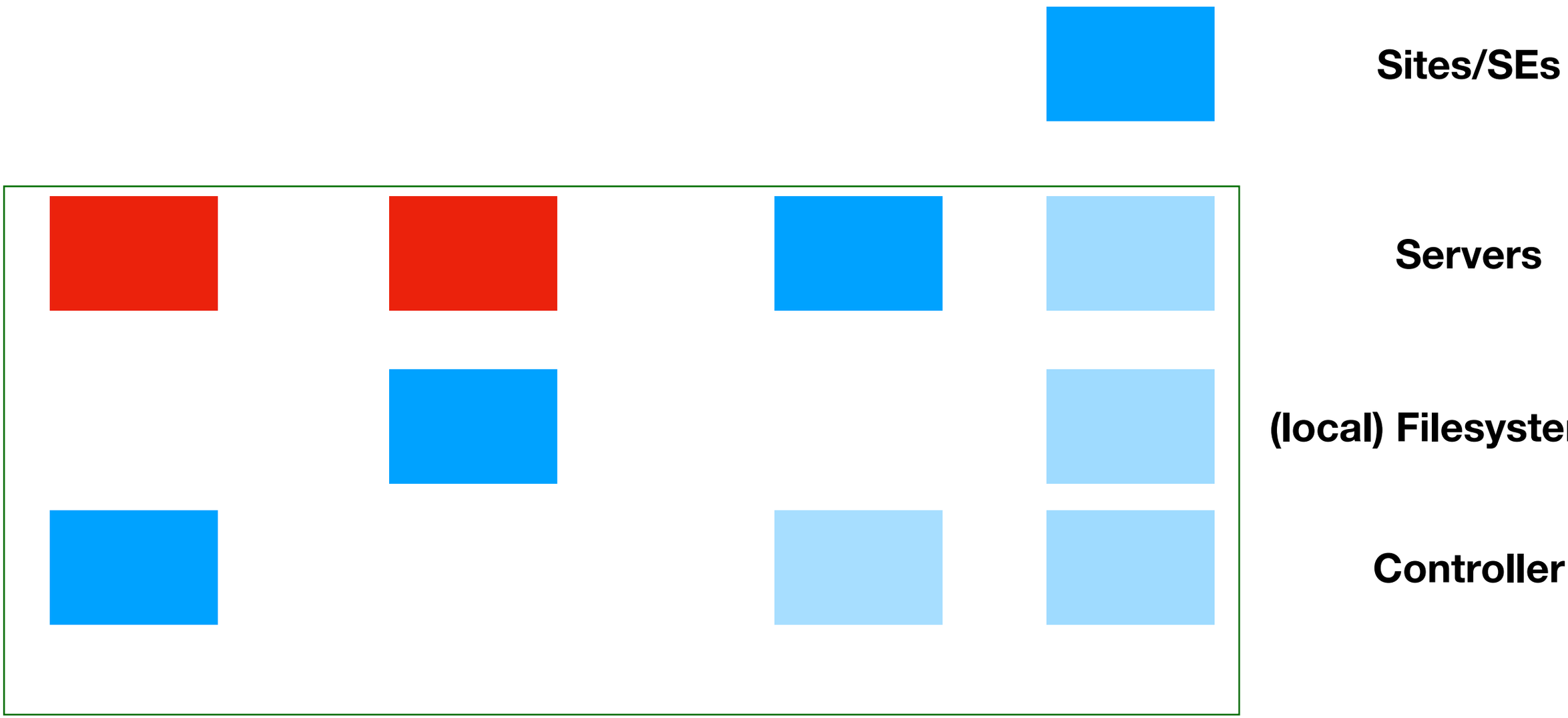
We deployed a Distributed Storage prototype

Based on the EOS technology

Tinkering with resilience

Striping+EC/Replication **Whole file replication only**

DPM w/ RAID **DPM w/ ZFS** **CEPH(say)** **(Data Lake)**



Caches

- "Caches" are any form of volatile storage layer inserted into a path to improve performance.

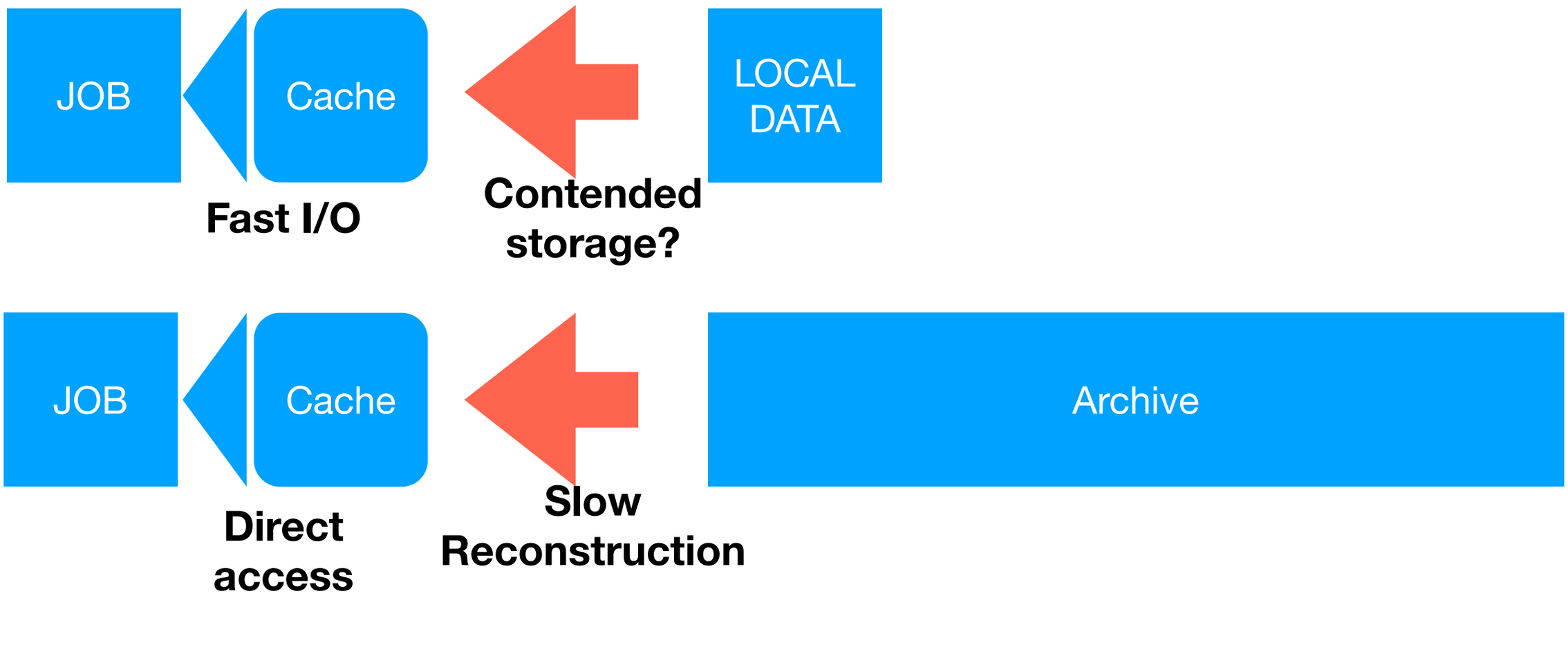
- Caches are useful only if data locality is important for your workflow:
  - imply a *strong coupling* between storage/data and work/jobs

- Broad enough to cover many types, with very different designs and applicability.

- No list will be exhaustive, as a cache simply needs to *improve* on *at least one* performance metric, relative to the layer it interposes.

Some Caches


JOB
Fast local network
Cache
Slow network
Remote DATA

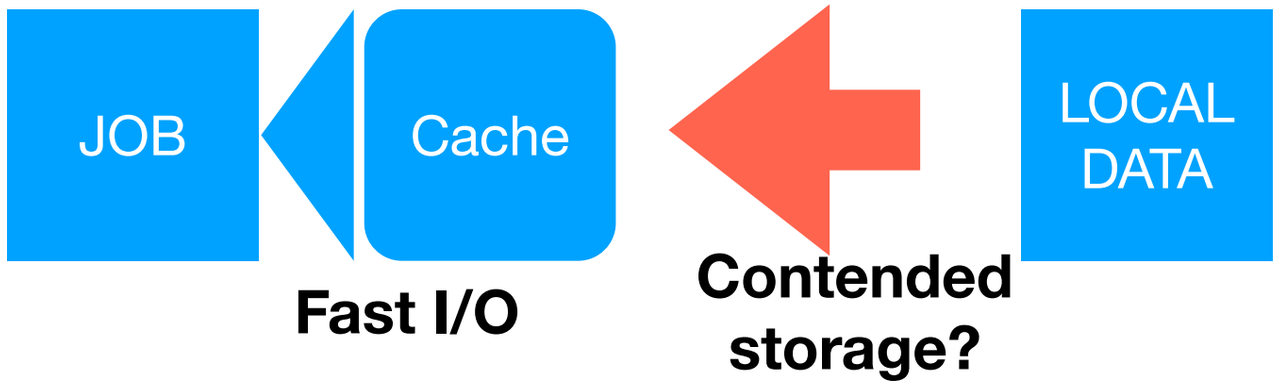

JOB
Cache
Fast I/O
Contended storage?
LOCAL DATA
JOB
Cache
Direct access
Slow Reconstruction
Archive

Some Caches

- "Site local" cache:
  - useful when:
    - data is read more than once [cache opportunistic]
    - data is read once but cache "pre-warmed" with expected data
  - not useful when:
    - data is read only once [cache opportunistic]
      - extra cost incurred from caching + extra network hop
    - jobs are not I/O latency limited at all, network optimised
- Cache performance area:
  - improved *locality / latency*

Some Caches

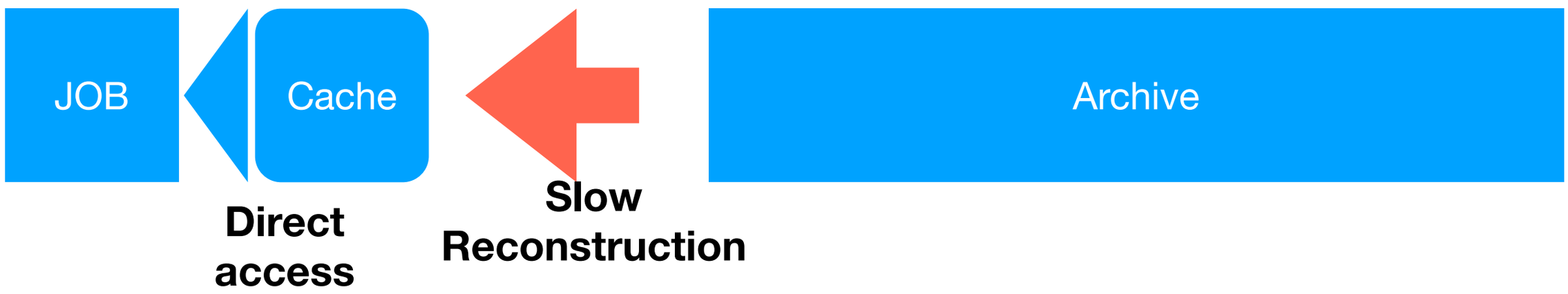
- **IO Performance cache:**
  - **useful when:**
    - **job is I/O bound w/ random I/O heavy workflow**
    - **temp edits needed to local copy**



  - **not useful when:**
    - **job is not I/O bound [esp if job is streaming I/O, reads data once]**
- **Cache performance area:**
  - **improved *performance [expensive SSD/NVMe etc hardware]***

Some Caches

- "Scratch copy" cache
  - useful when:
    - data is stored in slow to retrieve format [tape, EC stripes]
    - data needs to be read repeatedly / access model needs reconstruction on each hit
  - not useful when:
    - data is as easy to retrieve as from local disks
- Cache performance area:
  - improved *latency* / *locality*
  - reduced archive CPU/IO load/tape farm robot activity



Object Stores [and Cloud]

- "Cloud" storage solutions cut out POSIX guarantees for efficiencies.
    - Object Stores recapitulate many of the choices made for Grid storage, for similar reasons (immutability of files makes state consistency etc easy)
- Efficient use of Object storage for naïve grid workflows needs a caching layer [as Objects do not maintain file pointers], or changes to experiment code [less likely].
- Using "Cloud Storage"

Cloud Storage

- Support Cloud APIs, mostly Object Storage [**S3**, Swift?, CDMI?]

- Two use cases:
  - "decoupled Storage" == "archival/resilient copies"
    - economic argument? TCO , flavour of money
  - "coupled Storage" == "making data available to jobs in same Cloud"
    - requires work on job management / knowledge of data requirements.

Cloud / Grid Interoperability

- This is mainly a workload management problem!
- for Storage, we need a translation layer for:
  - protocol [“grid” -> S3] if we don’t natively S3
  - authorisation [X509 -> appropriate capability token]
    - Dynafed can do this now
    - WLCG “Tokens” will be easier to convert, see Security talks.

DOMA

- DOMA are the WLCG project working on Data Organisation/Management/Access.

- DOMA priorities directly contextualise and direct our own policies, via WLCG.

- Most relevant to Brian's talk, but…

- DOMA directions imply:

  - simpler storage [WebDAV might be sufficient?]

  - "Data Lakes"/"geographically distributed storage systems"/ moving resilience/QoS "upstream" from T2s.

  - Token/Capability based authorisation

The Future?

"small" Tier-2s

- "Light" storage: does not need direct access protocols (from outside).
- ~~Can~~ *Should* be POSIX (or Object Stores).
    - Needs to be useful for local users / shared on existing local resource.
- "Grid" access by:
    - "Caching" [volatile, ideally *prewarmed - implies workflow management choices*]
    - Explicit local user data placement.

"large" Tier-2s

- "Heavy" storage: needs (Grid specific) access protocols for managed data.

- Also must support our non-WLCG users: different protocols used outside HEP [mostly S3, http(s)].

- At present, majority of UK "large" Tier-2s are:
  - DPM [non-DOME], many large disk servers (of various ages).
    - resilience by multiple copies / server level EC ("RAID")
  - dCache (majority of *CMS* storage)

"large" Tier-2s

- Can we simplify / improve large Tier-2s?
- Storage costs:
  - Resilience at Server, not Disk level [improves per file performance via striping, improves resilience]
- Grid Layer complexity:
  - No SRM, unified filesystem, so… no need for an extra namespace layer?
- "Sunk Cost" / "Cost of transition"

"large" Tier-2s

- Doesn’t this look like:
  - EOS [with all the distributed storage options turned on]
  - Ceph [with RAL ECHO Xrootd shim on top]
    - As an aside, an obvious name for a RAL Tape system would be NARCISSUS
  - dCache [might not be “*simpler*” than what we have now…]

Tier-1

- [See Alastair's, Rob’s Tier-1 talks]

Wider scaling / consolidation

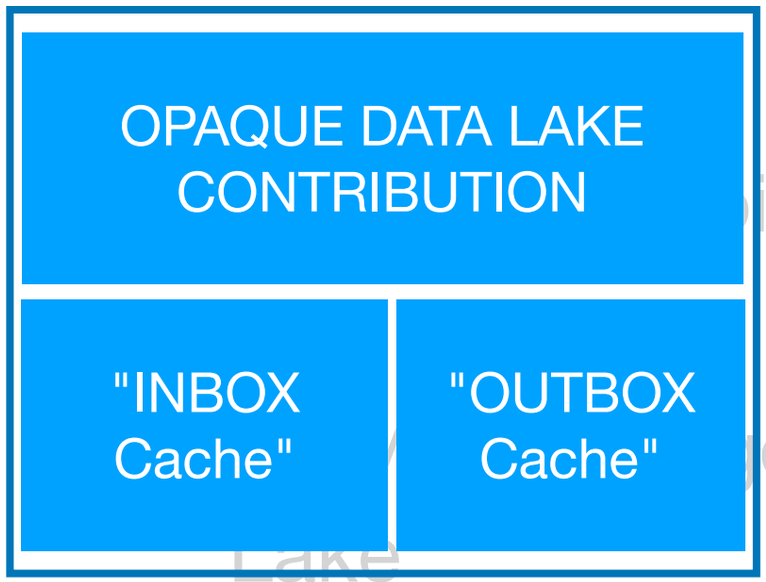
- Storage classes of Tier-2 data are (and will continue) evolving.
- Pressure on Experiment capacity requirements, worldwide, due to cost/scaling issues.
- Now (ATLAS+CMS):
  - (mostly) non-resilient replicas [of central data]
  - temporary copies of local job outputs
- Future?:
  - resilient copies of central data [extending from T1s]

Wider scaling / consolidation

- Future:
  - resilient copies of central data [extending from T1s]
  - DOMA: manage this w/ one endpoint ?EOS?Dynafed? "Data Lake"
    - potentially stripe / erasure code copies *across* Tier-2/Tier-1 "lake" for online storage. [RAIS*ites*]
    - [or *replicate* across "lake"]

Wider scaling / consolidation

- Future:



Reduces resilience needed at *sites*; but also requires *smarter* placement.

Decouples [large] Tier-2 sites from storage requirements (as reconstruction done at client)

All sites, ideally, need (volatile) storage for reconstructed copies.

- potentially stripe / erasure code copies *across* Tier-2/Tier-1 "lake" for online storage. [RAIS*ites*]

- [or *replicate* across "lake"]

Wider scaling / consolidation

- Non-WLCG VOs
  - with *UK*-controlled Data Lake, offer them same access [needs work from GridPP/VOs for data flows to sites]
  - with *WLCG*-controlled Data Lake…?
    - Tier-2s will still need (heavy?) storage at sites to serve these data requirements.

Wider scaling / consolidation

- Tier-2 "heavy" requirements in this context:
  - at least one of WebDav, Xrootd interface to uniform storage.
  - S3 interface?
- Tier-2s look more like object stores / byte store here [something already true for, for example, Tier-2s SEs accessed via Rucio]

Storage Group's Roles

- Tier-1 technology / FTS / Rucio (multiVO or otherwise)
  - "non-WLCG" + IRIS support
  - DIRAC? [DFC catalog / access]
- DOMA/VO liaisons
- Tier-2 technology migrations
  - "non-WLCG" + IRIS support
- “Data Lake” work + management

Conclusions

- small Tier2s -> no grid storage / intelligent pre-filled caches using locally useful POSIX filesystems.
- big Tier2s -> Grid Storage / simplify provision [shims on distributed filesystems]. Object Store interfaces for storage increasingly important.
- wider scale -> work needed, with DOMA, on Data Lake single-endpoint UK-wide solution.
    - also our interface to "Cloud storage"
- Moving away from "Grid-parochial" solutions is always best.