Data Management: Post-CWP

Brian Bockelman

Standard Disclaimer

- This talk aims to motivate discussion in the DOMA area and highlight some activities I’m personally involved in.
    - It is *NOT* a survey talk.
    - I include my biases and how I interpret things that are in the CWP.
    - Please interrupt to delve deeper on topics.

Post-CWP? Let's review CWP first!

- CWP: https://arxiv.org/pdf/1712.06982.pdf. Data Organization, Management, and Access (DOMA) on pages 36-41.

- Definitions:

  - **Organization**: How data is structured in storage. Prominent example: ROOT file format.

  - **Management**: Overall handling of data location policy and execution (replication), associated metadata, archival, and lifecycle.

  - **Access**: Protocols used to access data for computation. Can also include conceptual approach (file-based or database).

CWP Review

- Expected challenges going to HL-LHC:
  - Experiments will increase data rates & volume. Significant **cost** challenge.
  - Increasing computational costs are expected to lead to new computing resources. Data will need to be accessed from **new** and more **dynamic** locations.
  - New computational techniques (machine learning; *high event rate analysis facilities*) will require different **access paradigms**.

Research Directions

- **Event-level access granularity**: Aims to support new access mechanisms (database-like queries) and reduce cost (shorter jobs provide more flexibility, which allows utilization of more resources).
- **Adopt external, non-HEP tools** such as Spark (data management) or Ceph (innovation in storage layer). Reduce costs, support new use cases.
- **Exploit varied QoS from storage**: Study potential role of opportunistic storage or different archival storage solutions.
- **Data placement & latency optimizations**: More differentiation between sites - **“data lakes” and caches**.
    - Discussed further in this presentation.

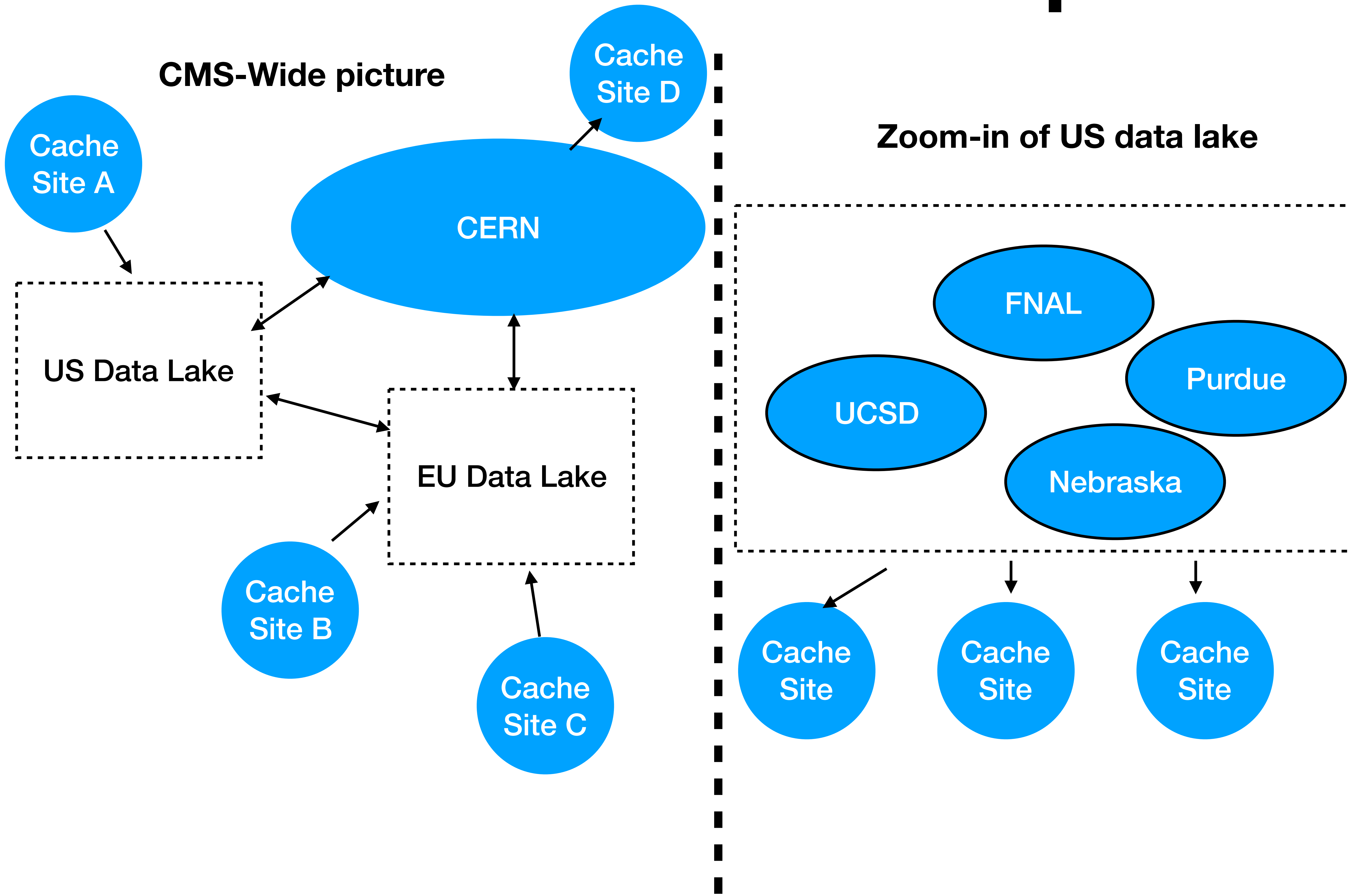
Data Lake Concept

- Instead of one-SE-per-site, have a single **logical SE** that encompasses a significant amount of high-performance storage.
- Sites outside a data lake are dynamic: no persistent experiment storage.
    - E.g., all is cached or streamed.
- Contrast “experiment storage” with **user outputs**: goes directly to destination site user is associated with.
    - Regardless of whether the site is in-or-out of the lake.

Why Data Lakes?

- Motivation for change: **current DOMA model has high cost**.
  - The "storage element centric" model is a complex, manpower-intensive way to manage data. Requires significant sysadmin attention and central ops teams.
  - Current HEP implementations have little overlap between experiments.
  - Hence, total cost **upper bound is**:
    $(BIG_NUMBER) x $(# of SITES) x $(# of experiments)
- How to reduce this bound?
  - Reduce number of sites that have their own storage element. ← **Insert Data Lakes Here**
  - Reduce number of (unique) VO data management systems. While we have a poor track record, there is some interesting work around Rucio (potential future topic?).
  - Alternate data management models. ← **Sub-event access? Database-like access? Not explored here…**

Data Lake Example



Data Lake - Highlights

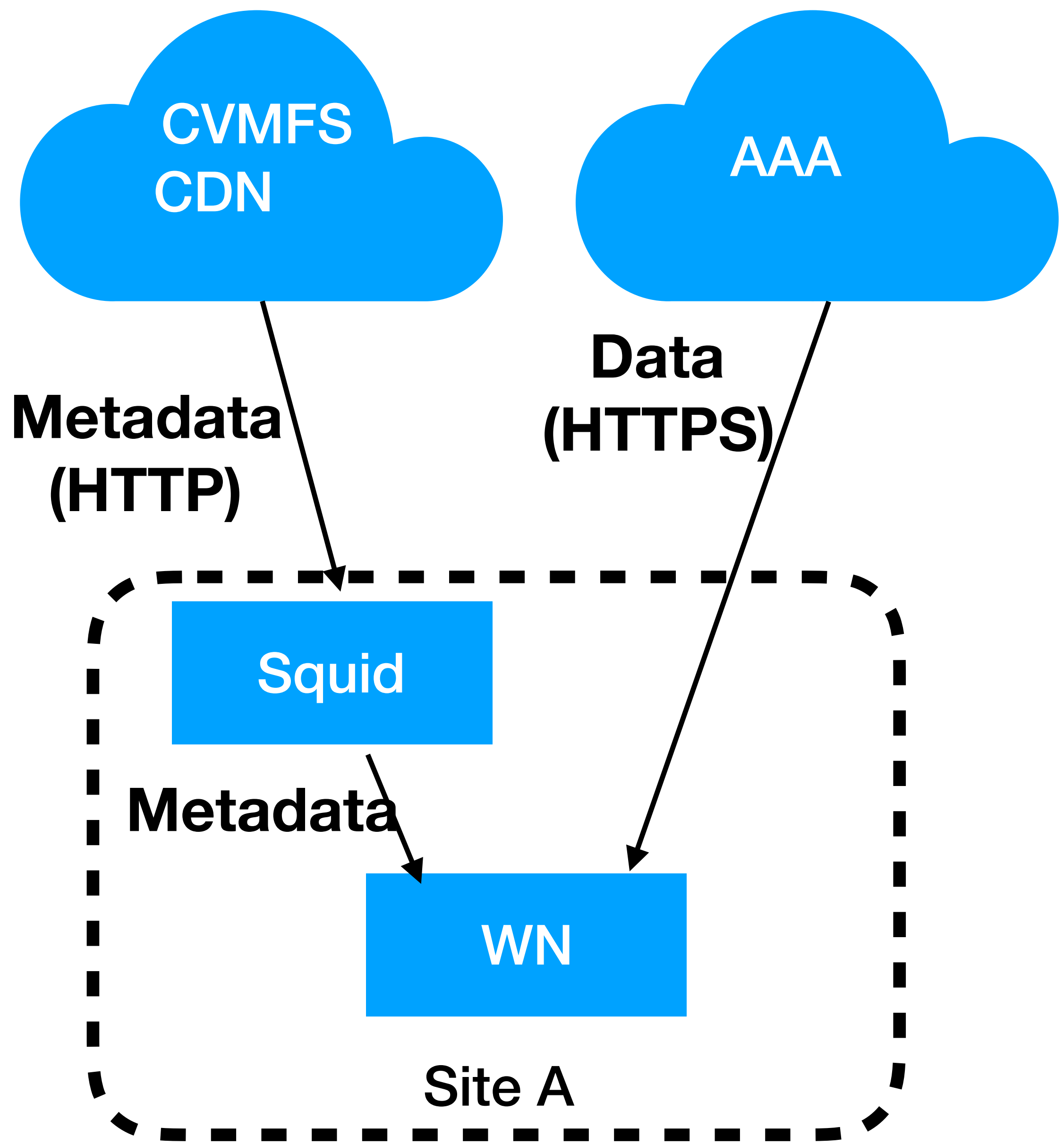
- Data lake is a *logical SE*, which can be either a single site or multiple.
  - If applicable, intra-site transfer is responsibility of the lake.
  - VO is only responsible for inter-lake transfers.
- Data lakes are assumed to be well-connected with each other.
  - Non-data-lake ("cache") sites may be small or **HUGE**. Not a reflection on site quality, but overall storage investment for a VO.
  - Sites are not tiered!
- Cache contents are not managed by the VO.
  - Caches optimize connectivity to a given lake.
  - Cache may store transient outputs — not necessarily streamed.
- **Drawback**: trade-off between storage management and data management. Data lakes explicitly assume more out of the storage layer.

Current Demo - Big CVMFS

- Motivation:
  - With AAA, we have already shown the ability to scale inter-site data access.
  - CVMFS has demonstrated how to provide an extraordinarily scalable namespace with a modest tradeoff.
  - Why not combine the two? Deliver data via AAA but provide namespace with CVMFS.
- Focusing on a subset of well-connected sites with good AAA performance: US Tier-2s. All sites interconnected with 100Gbps and storage similar in volume (~3-4PB usable).
- Right now, Nebraska publishes all its CMS files in a CVMFS repo — cms.osgstorage.org.
  - Significant subset of UCSD is also included.

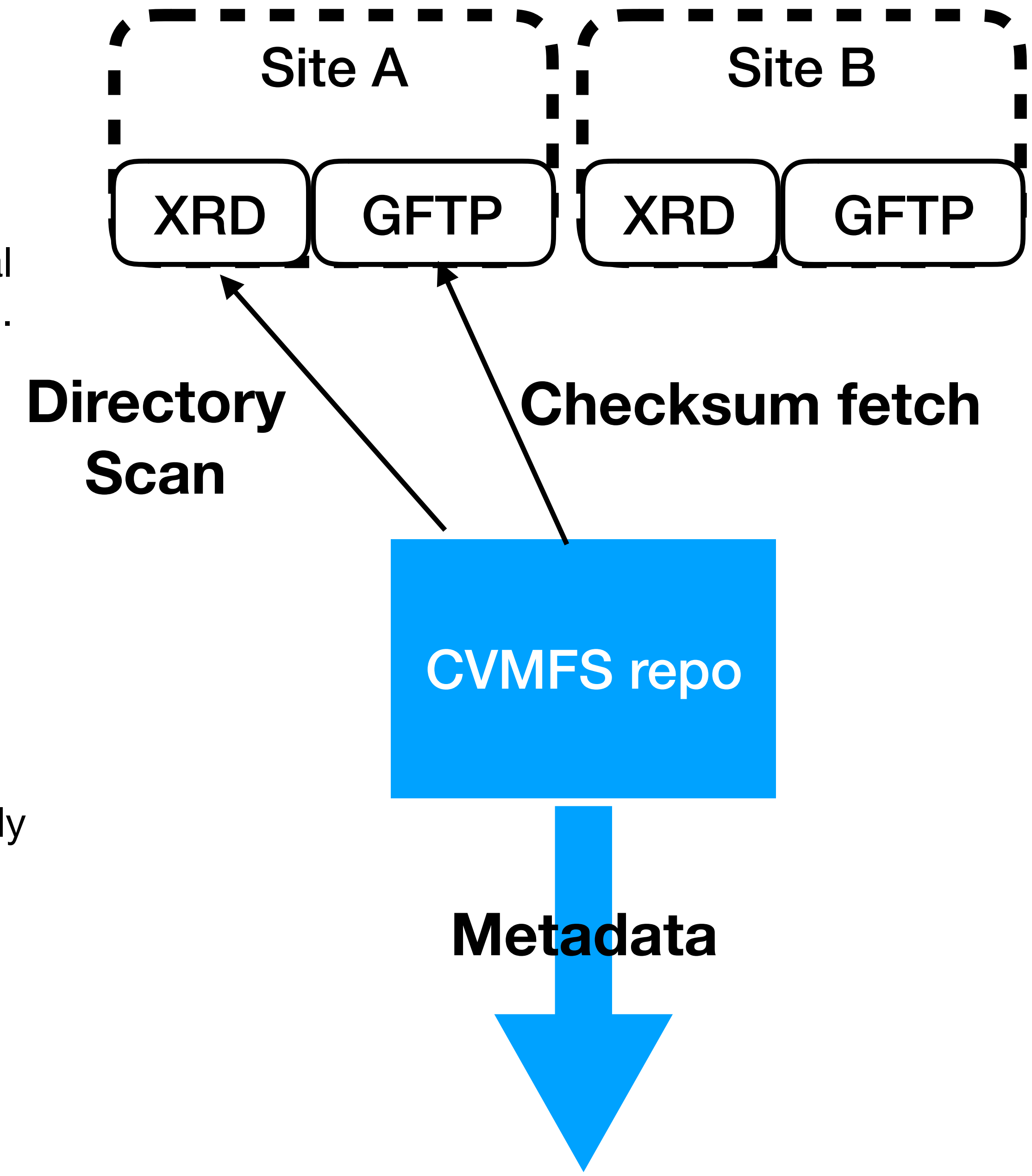
Big CVMFS - Details

- Starting in CVMFS 2.3, we added the ability to:
  - Have the CVMFS / FUSE client download data from files *not* in the existing CDN. (e.g., use AAA).
  - Utilize a separate authorization callout to retrieve credentials from the user environment. In this case, we get the GSI proxy from the user.
    - Note: when HTTPS is used, we can't proxy.
  - Enforce ACLs at the repository level.
- CVMFS provides an extraordinarily scalable namespace. Solves AAA problems:
  - Record of what is *supposed to be* accessible via AAA!
  - CVMFS client can be updated independently of CMSSW version.
  - **POSIX!**
- Overall, behaves more like a distributed storage element than a data federation.



Publication Process

- The CVMFS repository server contains a systemd service that periodically crawls each participating site.
    - Files in CVMFS but no longer at the site are removed.
    - Files *not* in CVMFS are checksummed using the special CVMFS checksum format (one SHA-1 per 24MB block).
    - Either the GridFTP server responds with this special checksum format or data is streamed.
        - The latter is only relevant for non-CMS cases.
    - When scan is complete, namespace transaction is committed.
- Repository server serves the namespace via HTTP in a highly cache-friendly, read-only manner.
    - Completely analogous to “normal” CVMFS.
- **Files written into the data lake may take O(1 hour) to appear in CVMFS.**



Big CVMFS: Outlook and Other Notes

- BigCVMFS has been a partnership with LIGO: overlapping use cases, but different scales.
- Intra-site data balancing still needs to be investigated: currently looking at doing this with Rucio.
- Challenges exist in continuing to scale the directory scans.
- Expecting to scale up in 2018.
  - Need to fully-enable HTTPS for all sites in the redirector.
- Looking forward to HL-LHC, scaling challenges mostly in terms of number of files to scan.
- Need a forum to interact with other data lake demonstrators, such as multi-site EOS.

Why Caches?

- Cache contents are not explicitly managed by VO.
  - VO can provide hints to pre-populate the cache (or what should be evicted). Makes no assumption about data availability.
  - Cache can include outputs from the site, but jobs aren't considered "complete" until output is copied to a lake.
- Caches are meant to "feed" computational resources.
  - **Cost reduction**. Ideally, less VO-specific activity at such locations, allowing effective utilization of smaller resources.
  - **Dynamic or new resources**: Cache may be an "edge service" at a large HPC resource, or provide ability to "move in and move out" of a site.

Active work within CMS

- **Streaming**: Essentially, a zero-sized cache. We did a round of optimizing CMSSW for high-latency access before Run 2. Some signs that a new round is needed in the context of other new Run 2 techniques (multi-threading, premixing).
- **Technical demos**: Large-scale (>100TB) XRootD caches at UCSD and Caltech. Using LRU for cache eviction and whitelists certain latest versions of MINIAOD.
    - Trying to have one set of MINIAOD between the two sites.
    - Observing cache eviction & thrashing rates.
    - Improving technical implementation of the XRootD proxy cache ("XCache"). Original implementation of proxy cache was part of the AAA grant - 5+ years ago!
- **Working set size estimation**: Based on data popularity information, simulate the working set size of CMS analysis and analyze different cache replacement policies.
    - Try to understand how the system would behave if a significant portion was cache-based.

Conclusions

- Caches have been used for awhile within CMS: usage and understanding scaling up.
- Data lakes are a popular idea, but implementations are just beginning.
- CWP goal is to have large-scale prototypes by 2020.
- Significant DOMA challenges remain, particularly around what a “data analysis facility” may look like:
    - If we have an analysis facility, what does ingest / egress look like?