Technical Evolutions at CMS Sites

Brian Bockelman,
Thinking Run3 and Beyond

Standard Disclaimer:

This is not a CMS talk.
This is my personal view of ongoing trends I have observed.
I can be wrong. Very wrong.

What is a CMS Site?

- Surprisingly difficult question! Here's my working definition:
  - "A CMS site is an organizational unit that provides CMS with one or more computing services at scale"
- A few things about this definition:
  - "**organizational unit**" - the site boundaries is delineated by how it is administered, not where it is physically located.
  - "**computing services**" - standard batch resource, "special" resource (high-mem, high-IO), archival service, file streaming, file replication.

What is NOT a CMS Site?

(Purposely controversial: discuss!)

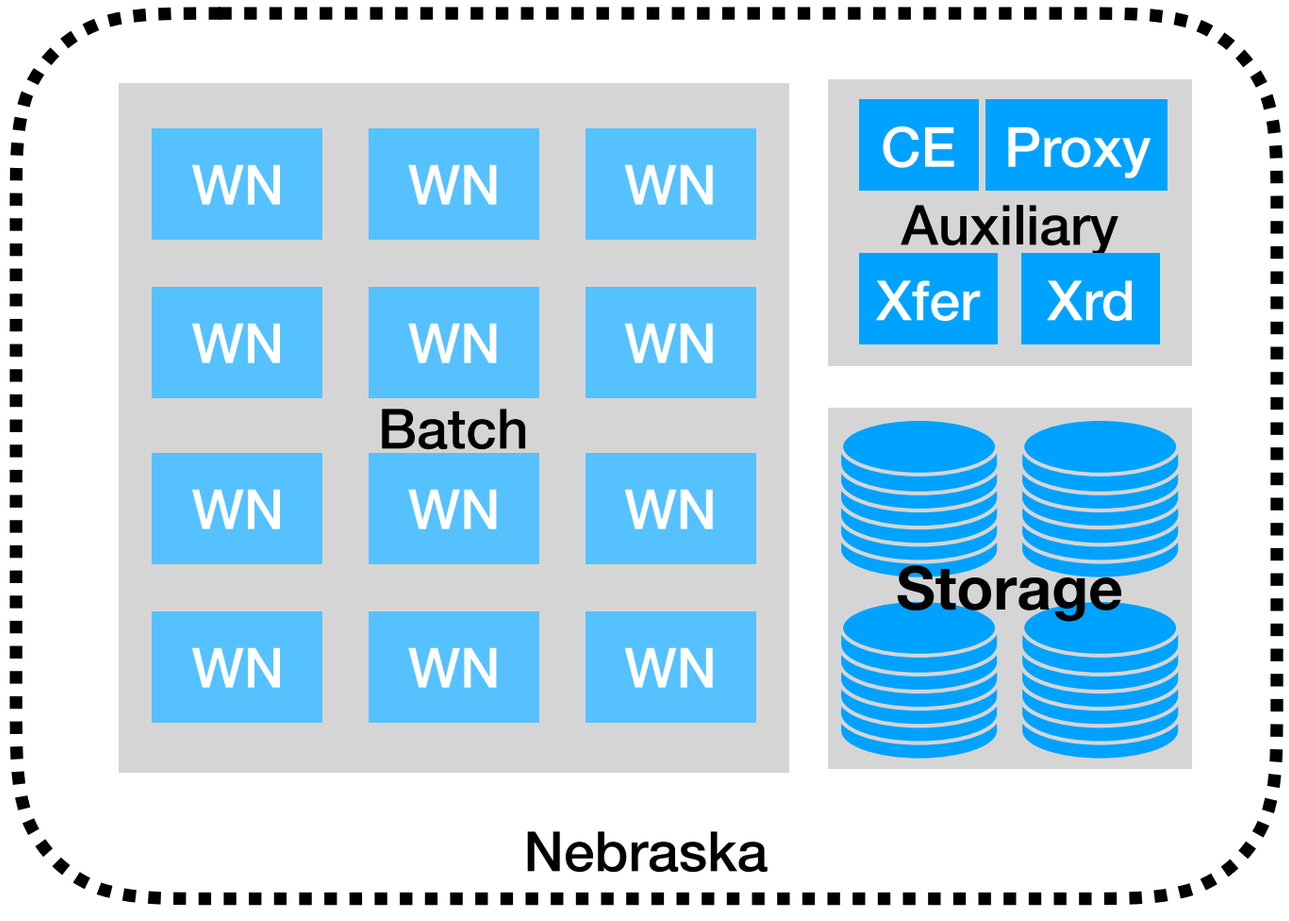
- A CMS Site is not simply a WLCG site in a tiered computing architecture.
    - A single CMS site can span many WLCG sites.
    - A CMS site may not be at *any* WLCG sites.
    - One must work hard to see the distinction of Tier-1 versus Tier-2 versus Tier-3. This is a significant evaluation from Run 1!
    - The only “special” site is the Tier-0. Even there, we have shown it is possible to perform “Tier-0 activities” at other sites.

How I think of CMS Sites

Auxiliary may contain:

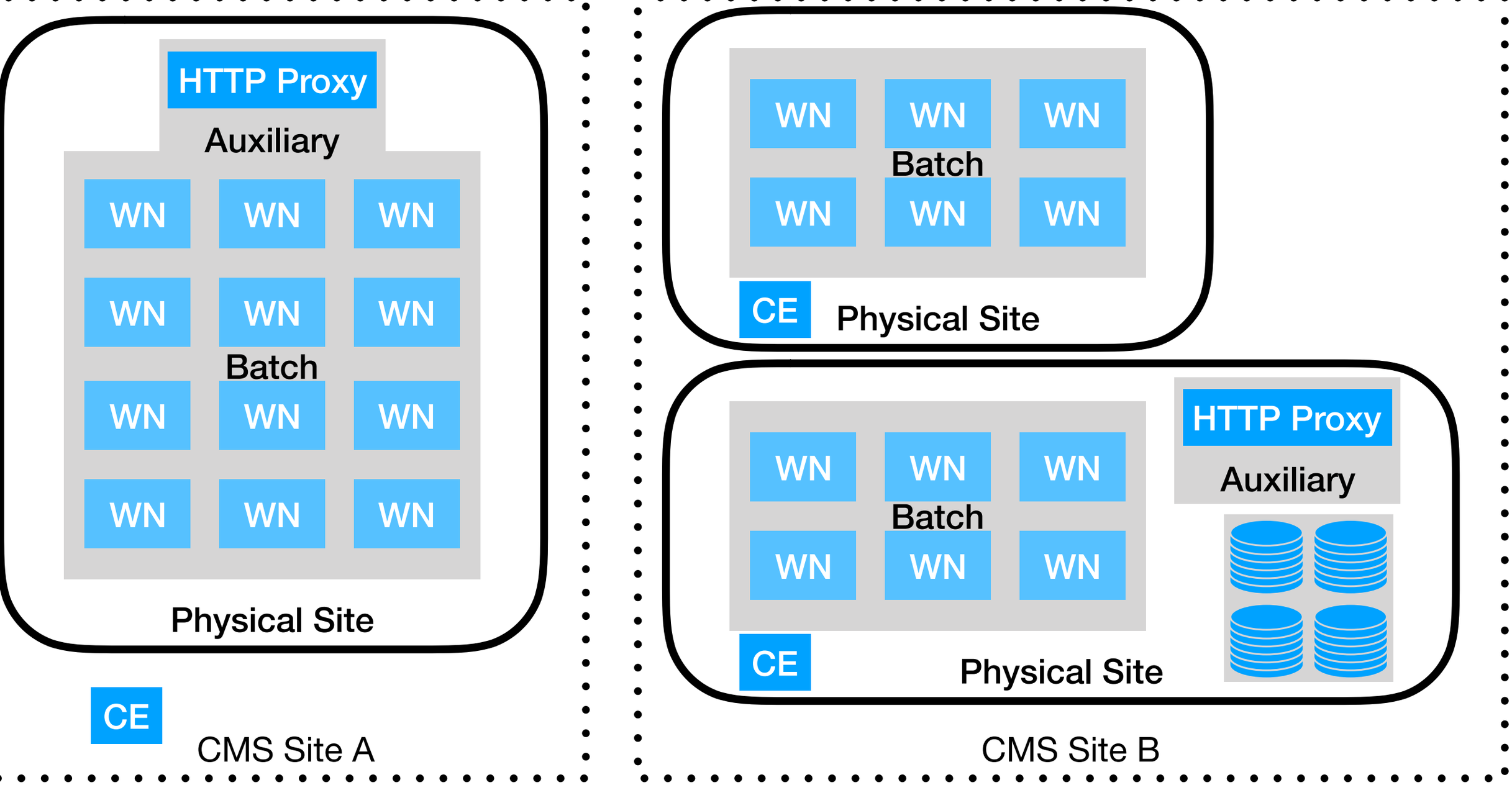
- HTTPS Proxy (Squid)
- CE(s)
- File Transfer: GridFTP
- File Streaming: XRootD

This picture purposely ignores “site fabric” such as running Puppet, monitoring, etc.



Not even trying to draw cloud/HPC sites!

How CMS Sites can look



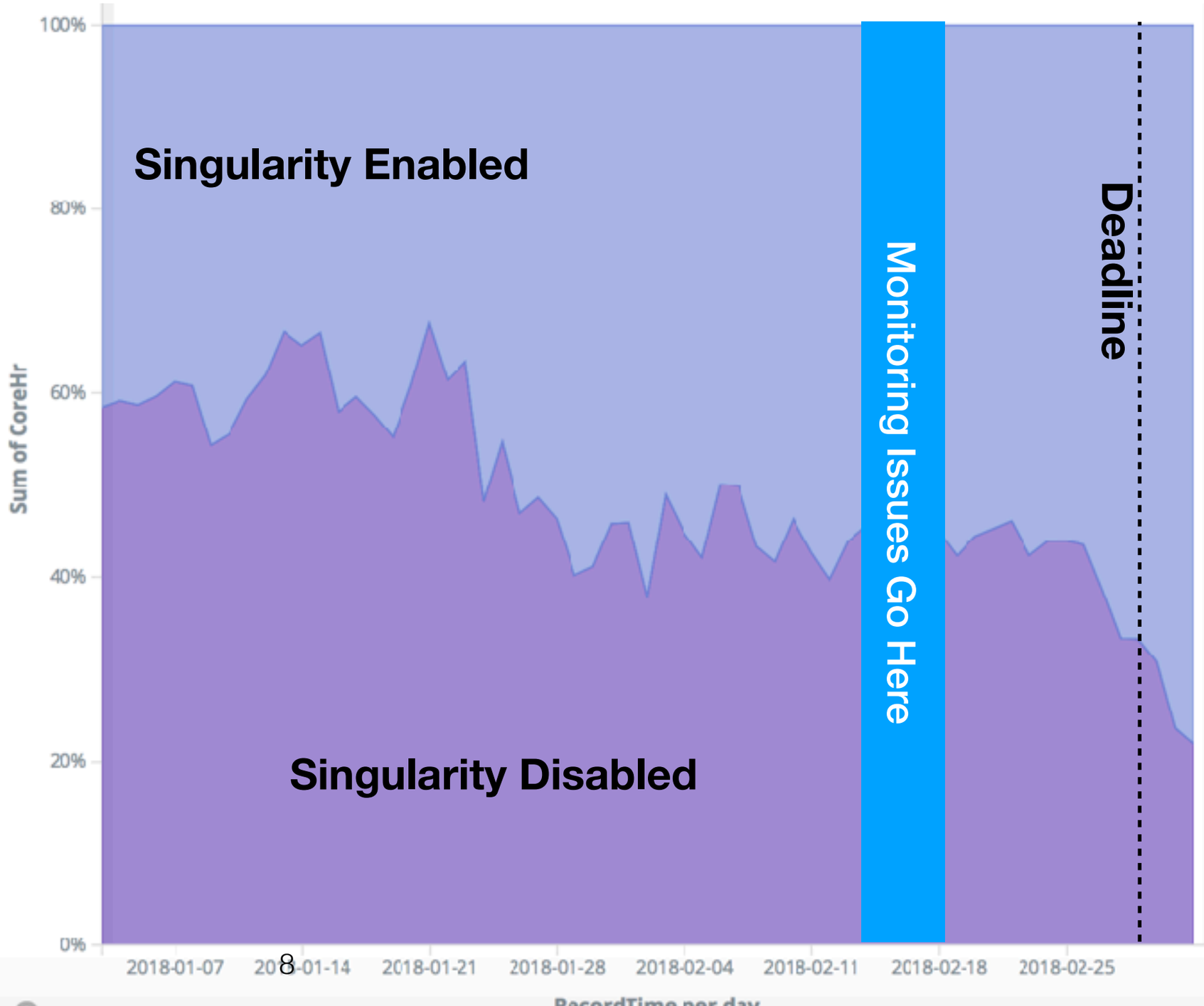
Goal is to allow the logical site to best describe capabilities, not prescribe a certain physical structure!

CMS Site Evolution - Batch

- Thread counts will continue to increase and memory-per-core will continue its decrease. Current baseline requirements for most workflows is 8 cores / <1GB-per-core. An increasing number of workflows will get moved to this category.
    - It is reasonable to believe the “average” workflow will run well with **16 threads in Run 3**.
- The biggest change at the batch layer is essentially done: conversion of payload jobs to containers. CMS required Singularity at sites starting March 1.
    - SAM tests will turn “red” for non-Singularity sites “real soon” (end of month?).
- Singularity is invoked by the pilot; only the payload runs inside the container. Two sites additionally run Singularity inside Docker.
- Singularity containers are distributed as containers via CVMFS.
    - **Potential joint project**: creation and distribution of “fat image” or “single release” containers — particularly at non-CVMFS sites.

Singularity Rollout By The Hours

Unsurprisingly, the biggest transitions happened within 1 week of the deadline!



CMS Site Evolution - Batch

- Containers beget new failure modes: while it isn't visible to sites, the pilot infrastructure continues to evolve to better handle failures.

- Today, about 90% of Singularity use is in "privileged mode" (i.e., setuid).
  - If you aren't worried about a setuid binary at your site, you should be!
  - Unprivileged mode works (FNAL exclusively uses this)!
    Unfortunately, the kernel support has a poor security track record. It currently takes an ambitious site to keep up.

- **Over the next few years, container usage will be normalized and we will switch to unprivileged mode.**

CMS Site Evolution - Batch

- Containers are likely necessary, but insufficient, to better support non-WLCG sites.

- Two other stumbling blocks remain:

  - **Getting pilots to a site**: significant progress here as we can launch pilots via a SSH connection. With proper authorization in place, HTCondor workers can be launched by the site.

  - **Outgoing network**: Even if available payloads don't need this, the pilot requires a live TCP connection. This One Is Hard.

- Handling the "outgoing network issue" is R&D that is barely started. Between now and Run 3, I will be happy if we can solve it at least at high-impact sites.

Data Movement

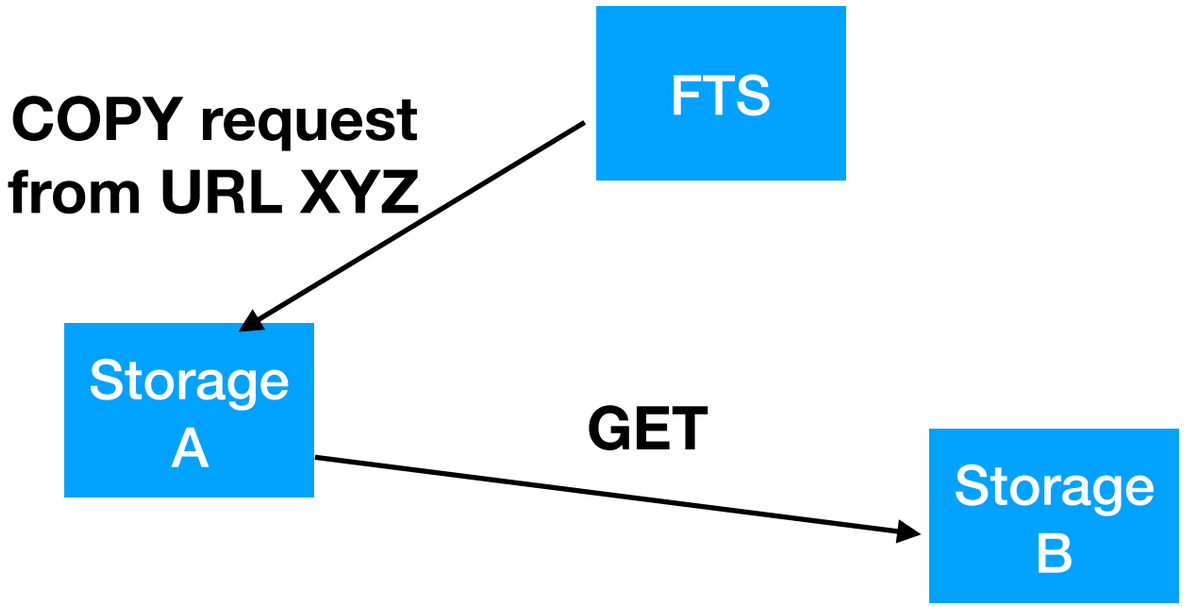
- I mentally organize data movement into two use cases:
  - **Bulk File Transfer**: Move the entire file from storage A to storage B. Responsiveness measured in hours.
  - **Data streaming**: Move a subset of the file from storage A to application X. Responsiveness measured in milliseconds.
- Let's see what's in store here…

Bulk File Movement

- GridFTP is our primary bulk file movement mechanism.
  - Globus GridFTP implementation has been forked and support provided by a community of grid organizations (www.gridcf.org).
    - I see this as providing very important “coverage” for us in the next 3-4 years.
- **Goal for Run 3**: Migrate from Globus to HTTPS/WebDAV for bulk file transfers.
  - In 2018, this is going to mostly involve storage developers and adventurous sites.
  - Bulk of evolution activity will occur in LS2.

WebDAV TPC

- WebDAV TPC is done by FTS contacting one storage endpoint, asking it to COPY to/from a given URL.
    - The active endpoint performs the transfer, typically a HTTP GET or POST.
    - Important: **ANY URL** can be given, including GridFTP or XRootD.
    - “Storage B” needs to know *nothing* about WebDAV TPC; only needs GET/PUT semantics. Allows transfers with S3, for example.
- Already widely implemented, including plugin available for XRootD.
- Tricky part: *authorization* with Storage B. For this, we are working on a concurrent transition away from X509 to bearer-token based.



Authz revolution:

- **Identity-based:** authorization based on mapping who you are.
- **Capability/Token-based**: authorization based on something you are able to present.

Authorization Ecosystem

- In this ecosystem, the VO issuer (Rucio?) would issue a signed token authorizing an action ("open file at Nebraska") to whoever holds the token.

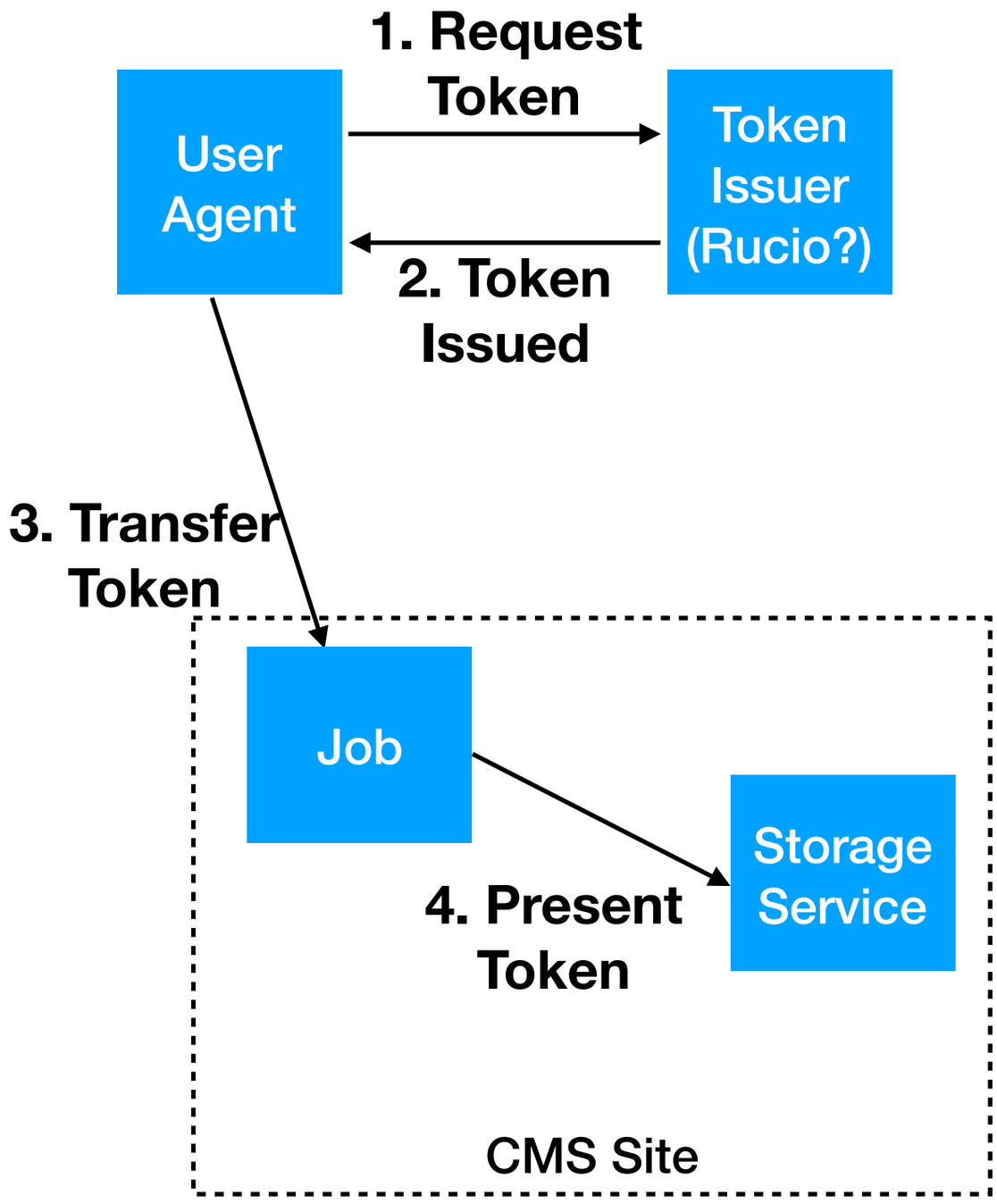
- Token is managed by a user agent (such as HTCondor) and distributed with the job.

- Job presents the token in the storage request (example: using HTTP header).

- Storage service checks signature against VO issuer's public key, then authorizes the action relative to the VO's storage area.

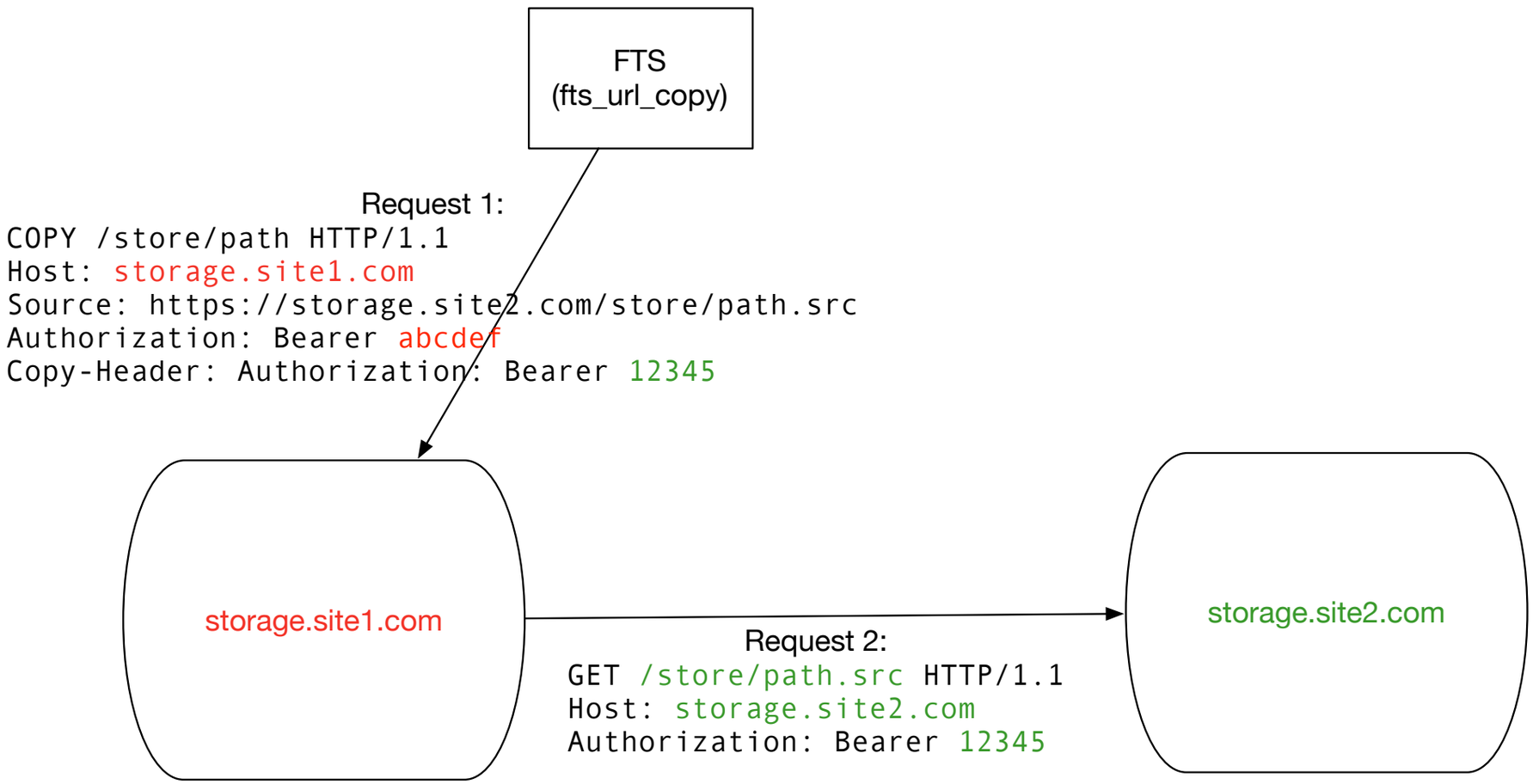
    - Storage service DOES NOT map users or manages access within filesystem.

    - User identifier still present for traceability purposes.



NOTE: User never explicitly manages their own tokens.

Put it together



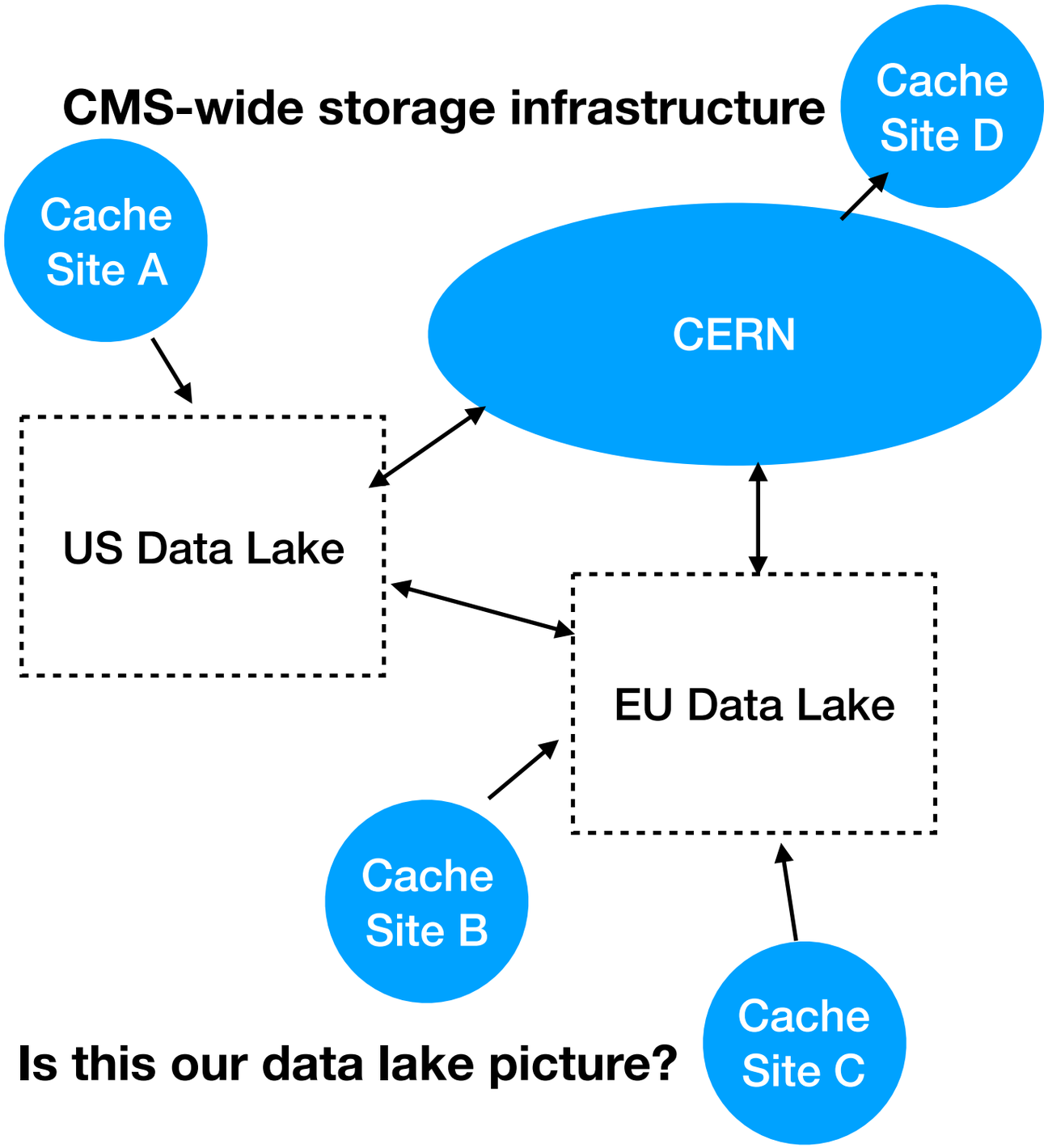
No Globus, no user certificates, fine-grained security.
Curious? Will be presented in detail at WLCG Workshop

Data Streaming (and Caching)

- **Data streaming will continue to be an important use case**, with further improvements made to efficiency.
    - The XRootD protocol (and client) continues to be the best match.
    - I hope to see a quick **transition to XRootD-over-TLS** (once it exists!). This enables use of tokens with XRootD.
- Between now and Run 3, I see the data streaming (AAA) being married with XCache as a simple storage solution for processing-oriented sites.
    - Larger sites are stress-testing XCache. Still exorcising the daemons: progress year-over-year, good shape for Run3.
    - Some technical work to figure out what parts of the namespace to cache-vs-stream.
    - To make this happen: need to find a “cinderella site” that wants to get rid of their current storage system, yet will actively engage with CMS to stamp out bugs.

Data Lakes

- I'll skip conceptual details; see Torre's talk.
- I see the benefit here in shifting the boundaries between data management and storage management — hope is this pushes down operational costs.
- For data lakes, I see Run3 as a bit of dress rehearsal for HL-LHC. My predictions:
  - "Cache sites" will likely exist by beginning of Run3, even if they use the data federation directly.
  - There will be one data lake that looks like "data federation ++".
  - There will be one data lake that is a multi-site storage system (EOS?)
  - Figuring out how to best use these will be a theme throughout the run.



Networking

- **WAN**:
    - Networking requirements should grow in proportion to the total dataset size.
    - Have the ability to use beyond the minimum, but not required. Often a difficult conversation to have with network providers (networks tend to want to upgrade based on peak traffic while CMS tends to think in average traffic).
    - Larger processing sites and data storage sites will continue to survive on 100Gbps. Smaller processing sites *may* be able to get by with 10Gbps.
- **LAN**:
    - The switch to premixing in 2017 reduced LAN requirements for digitization by factor-of-20.
    - High-IO workflows will be isolated to select sites with a higher ratio of IO capability to cores.

CMS Site Evolution - Other

- Various details won't really change:
  - CMS will continue to support CEs in whatever shape and form they come in — provided HTCondor can talk with it.
    - Heavy reliance on HTTPS proxies.
    - CVMFS as a software delivery mechanism.
- Some items I have no insights to share:
  - How commonplace will GPU usage be in Run 3? **Depends**.

Recap

- The big change to containers is already here!
    - But significant work remains to better leverage non-WLCG sites. I worry about series of VO-specific one-offs.
- The development effort for replacing GridFTP & GSI is ramping up. Lots of storage developers onboard for HTTPS and token-based auth.
- Data caching is well on its way - need to find the right onramp to larger scale use - and data lakes are just beginning.
- And this all has to be built on top of the working system!