CernVM-FS for Data

Brian Bockelman, Derek Weitzel

The Vision

Ideally, CMS and other VOs could utilize any CPU on the plan with zero host-level requirements.

Lacking that, what if we just required CVMFS:

- CVMFS delivers Software. **DONE**
- CVMFS delivers OS environment (container images). **DONE**
- CVMFS delivers container runtime (non-setuid Singularity). *In testing … hopefully production by Spring!*
- CVMFS delivers the experiment's data. **BIG CVMFS**

Big CVMFS

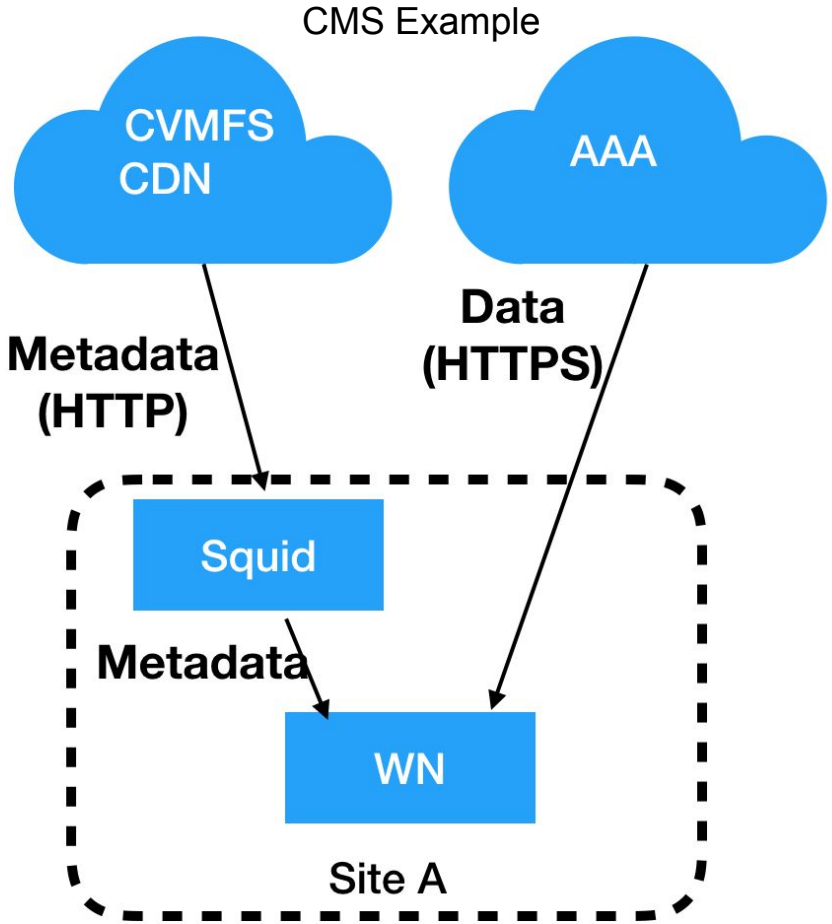
- Big CVMFS is not a data management technology.
  - It is a set of features added to the CVMFS client and infrastructure working toward the bigger vision.
  - It is the next evolution in our data access work.
- Big CVMFS provides the ability to publish petascale data repositories into CVMFS by linking the CVMFS namespace with an external CDN.
- Big CVMFS provides the ability to secure data access to a CVMFS repository.

What's Wrong With Streaming access?

- Streaming data access is available in most experiments.
  - Widely done via XRootD.
  - Making this efficient in ROOT remains a dark art...
- But, it doesn't provide a **namespace**!
  - What's in a namespace?
    i. Ability to list and navigate files (perhaps a directory hierarchy). Preferably via POSIX!
    ii. Able to know definitively whether a file is in or out of the namespace. Knowing the correct contents (metadata like checksums).

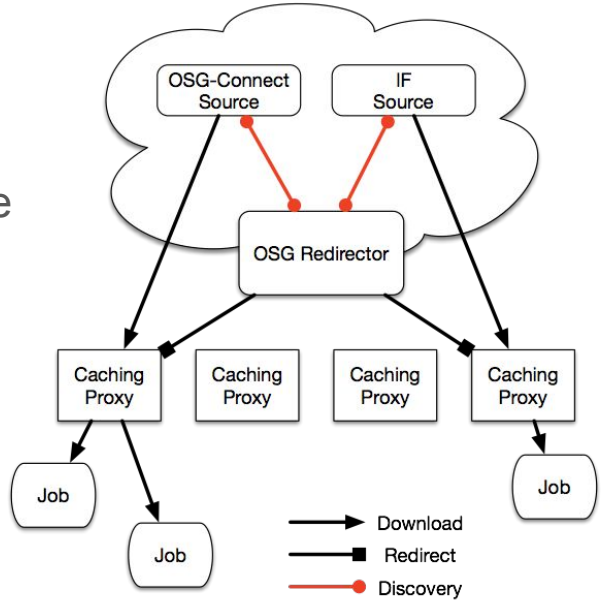
Big CVMFS

- 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.
- **POSIX!** Always the first request from many users.



Example - Public Data

- OSG runs an XRootD-based federation for different VO origin servers.
- OSG also runs a series of world-readable caches (also based on XRootD). Prevents clients from overloading the origin.
- OSG then periodically scans the data federation to synchronize current state to CVMFS. Examples:
  - **nova.osgstorage.org**: Serve (large) background distributions for the NoVA experiment.
  - **stash.osgstorage.org**: Serve data OSG VO users put in $HOME/public on the login nodes.
  - Proposal for a “LIGO open data” repo that contains the LIGO open data releases from 2016 and before.
- Any users can read the data from these repos. Considered public!



CVMFS Synchronization

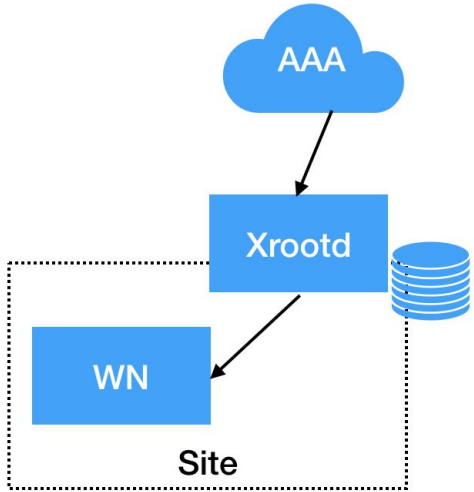
- Synchronization is a relatively straightforward-sounding script that must:
  - Start a CVMFS transaction.
  - Iterate in a source directory given an arbitrary root:// URL. Multithread to speed up indexing.
  - Check if the corresponding files exist in the CVMFS repo.
  - If not, then re-synchronize the file data. Stream and calculate the checksums.
  - Write out graft files for missing data; delete data that is no longer upstream.
  - Publish!
- However, the script is actually fairly tricky: must handle error conditions, make sure things are multithreaded and asynchronous, and verify that the entire filesystem isn't held in memory.
  - See: https://github.com/bbockelm/cvmfs-sync
- Recent improvements:
  - If available, can query remote server for necessary CVMFS checksum(s). Avoids need to stream data centrally (key for petabyte scale repos!). OSG ships necessary code in the GridFTP-HDFS integration.
  - Cleaned up script to have systemd-based times and proper RPM.
  - All our *.osgstorage.org repos now use the exact same code -- just a different configuration file.

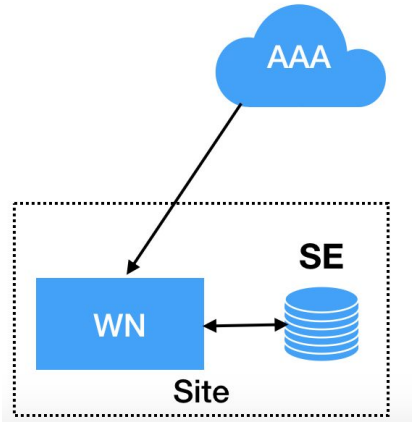
Current State - Secure CVMFS

- Secured CVMFS introduced last year. The CVMFS FUSE client can run a helper process to acquire user's credentials. Helper process can be externally implemented -- even in python!
- Example: secure with X509 certificates
  - Credentials are acquired by looking at accessing process's $X509_USER_PROXY environment variable (or from /tmp/x509up_uXXXX).
  - **cms.osgstorage.org** - CMS private data, including both official files and user files.
  - **ligo.osgstorage.org** - All non-public LIGO data
- Namespace is public, but data is only accessible with proper X509 certificate.
  - CVMFS still has ability to use non-public namespaces. This hasn't been strongly requested by any experiments. Public namespace is a "win" in terms of scalability (can reuse the CVMFS CDN).

Cache Strategies

- There are two valid caching strategies we have explored:
- XRootD-based caches:
  - Caches are placed strategically across computing infrastructure.
  - Caches serve data to one or more sites.
  - Caches must be authenticated and operated.
- CVMFS-based caches:
  - CVMFS FUSE process implements a multi-layered cache which writes into local WN disk and site-level shared storage. Serves only one site.
  - FUSE client runs as a privileged process: no need for additional authentication. No need for additional operational components.
- Not clear one can completely eliminate either?!?
  - XRootD-based would be best for regional caches.
  - CVMFS-based would be based for edge services or very small sites.
- At Nebraska, we run both. CVMFS cache is implemented on HDFS.
  - CVMFS-based cache might be slightly easier?





CMS Example

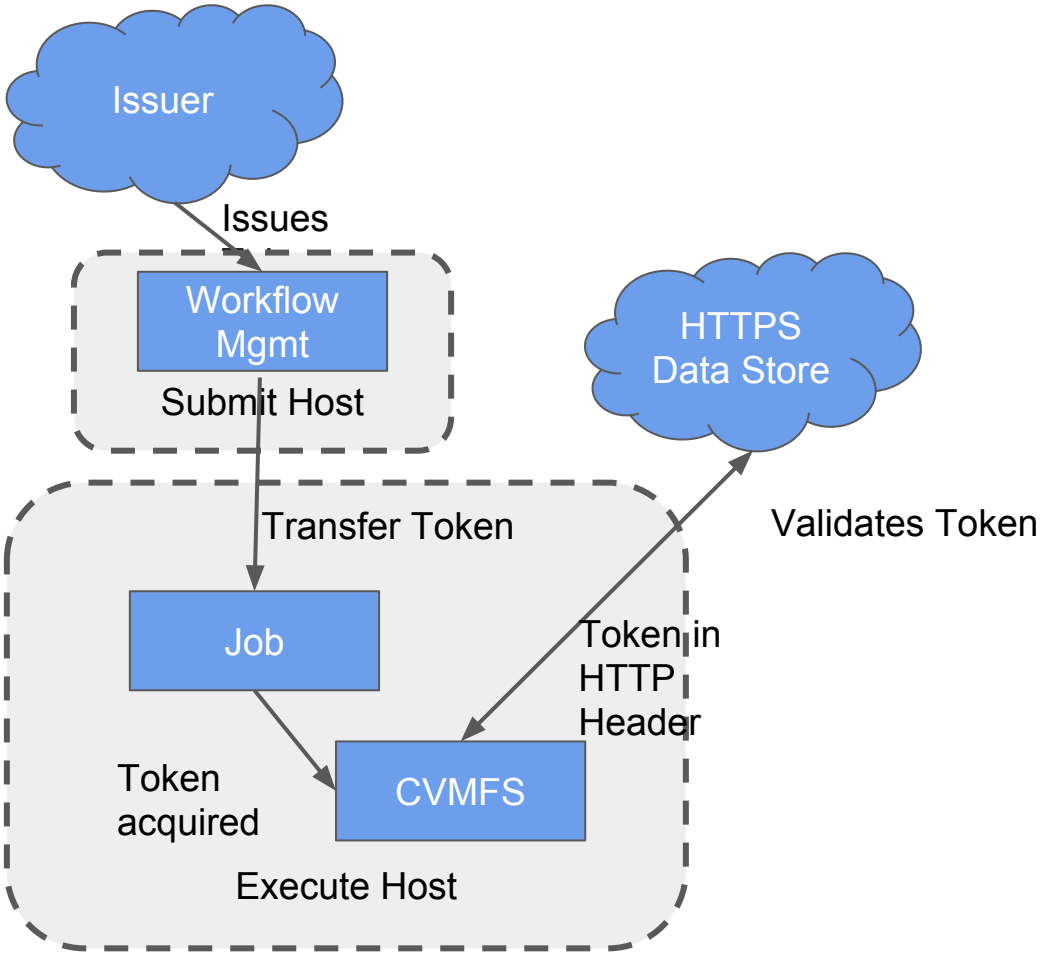
- For cms.osgstorage.org, we currently have two Tier-2s indexed.
  - A few rounds of XRootD patches were needed to make indexing of sites fast enough.
  - Synchronization relies on remote GridFTP server to provide the CVMFS-style “checksum”.
    - Since there is one checksum per 24MB, the “checksum” value may be in the megabytes!
  - This approach is lightweight in terms of operational cost - single systemd unit on one host.
    - Much better than prior attempts, which relied on HTCondor to distribute “checksumming jobs”.
- Access to the repo is through AAA. For a small subset of sites, HTTPS-based access is enabled at the XRootD daemons.
  - HTTPS-based access complements parallel investigations into third-party-copy.
- Access requires a CMS VOMS proxy.
- All enabled by default if you use the OSG configuration repository!
- Currently, about 3PB published.

New Developments - non-X509 auth

- X509 user certificates say *who you are*. It's up to the system to map your identity to some access permission. VOMS augments this with groups/role-based mapping.
  - Alternate: *capability-based* schemes. Capability provides a specific authorization for the bearer, not the identity.
  - You use this all the time: Google Docs sharing URLs are a capability-based scheme.
- We want to enable capability-based schemes based on HTTP bearer tokens.
  - Unlike X509, capabilities are transmitted as a HTTP header ("Authorization") not as part of the TLS connection. Makes connection reuse simpler!
- We used SciTokens as an example (https://scitokens.org/). More on next slide.
  - Pull request is submitted and being reviewed #1980.
  - SciToken is discovered by a helper process from job environment - just like X509.

SciTokens CVMFS

- Remote issuer generates the token; workflow system brings it along with the job.
  - Token contents are signed by a known public key.
- Token is validated first at the WN to access the cache and mounted namespace.
- Token is forwarded as header on HTTP request by CVMFS in order to access data from Data Store.
  - Just like with X509, this can't be mixed with X509 auth.
- The public key is posted at a well-known URL and cached locally. *No distribution of CAs or VOMS info*, provided the service running the well-known URL uses a system CA for its hostcert.



Conclusions

- CVMFS is getting close to being able to power all aspects of computing, top-to-bottom.
  - This work shows this can even include large amounts of experiment data!
- We've shown that these large-scale data repositories are flexible:
  - Allows cache-based or direct streaming.
  - Public or non-public data.
  - Plugin mechanism for securing data access.
  - CVMFS metadata can be created by streaming the input data to repository host - or simply by computing it remotely.
  - OSG runs a half dozen of these repos all with the same script!
- The current widely-used authorization is based on X509: have demonstrated that this can be switched to more industry-standard bearer tokens (such as SciTokens!).