Xcache Essentials

US ATLAS Computing Facility Meeting

ANL

December 3 - 5, 2018

Andrew Hanushevsky, SLAC

http://xrootd.org

What is Xcache?

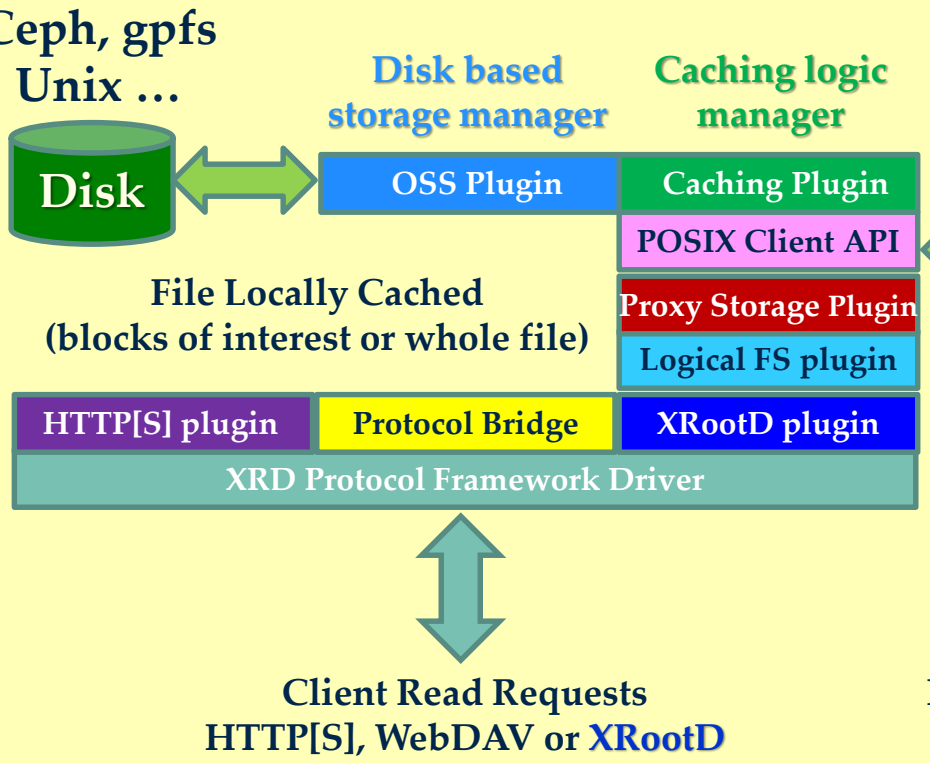
- A file or file block caching service
  - Scalable via horizontal clustering
  - Configurable for different workflows
  - Self managing in terms of disk utilization
  - Architected for local or regional use
  - Accessible via HTTP[S] or xroot protocols
- I will attempt some demystification

Xcache is highly configurable

- Allows adaption it to your environment
  - Simple single cache
  - Multiple VO cache
  - Clustered cache (i.e. multiple servers)
  - Multi-protocol cache (**XRootD** & HTTP[S])
  - CERNVMFS cache
- How can it be so flexible?

Xcache architecture allows it!

We compose the various plugins to construct a canonical caching proxy server!



XRootD *has a Lego™ type of architecture making it trivial to customize.*

Downside: It may be a bit daunting to get started.

Read() on miss read remotely

Logical FS thinks it's accessing local storage

There are configuration options for each plug-in

But Xcache can be simple

- The minimum directives to get going
  - all.export *path*
    - Path to make visible (any number of these)
  - ofs.osslib *path*/libXrdPss.so
    - Use proxy plug-in
  - pss.cachelib *path*/libXrdFileCache.so
    - Use caching plug-in
  - pss.origin *host*:*port*
    - Location of the data source

But that's usually too simple

- Where do the complications arise?
  - Security
    - Authentication and authorization
  - Resource limits
    - How much resource to use
  - Rucio
    - Original files may exist in multiple locations

The security conundrum

- Normally, need these directives
  - sec.protocol *authprotocol*
    - Use this authentication mechanism
      - Complicated by the need to config authentication
  - ofs.authorize
    - Authorize all accesses
  - ofs.authdb *path*
    - The *path* to the file containing authorization rules
- Fortunately, it's all boilerplate
  - For any particular experiment

The resource conundrum

- Normally, want these directives
  - oss.localroot *path*
    - Path where the file system is mounted
  - pfc.diskusage *parameters*
    - How much disk to use before purging files
  - pfc.ram *bytes*[**m**|**g**]
    - How much memory to use
- Unfortunately, these are site specific
  - Determined by the hardware being used

The Rucio conundrum

- Rucio usage would want these directives
  - pss.namelib -lfncache -lfn2pfn *path*/XrdName2NameDCP4RUCIO.so
    - Rucio specific plug-in to handle multiple sources
    - The plug-in is an ATLAS add-on
      - Not part of the **XRootD** distribution
  - pss.origin localfile:1094
    - This replaces the previous origin directive
- For ATLAS it's boilerplate but still…

Performance considerations

- Simple doesn't always mean performing
  - Additional directives may be warranted
    - Disk layout optimization
    - Caching optimization
      - Pre-fetching, blocksize, metadata location, cache bypass
    - Multiple caches for scalability
  - Some may be site specific
    - Dependent on the hardware being used

Feature considerations

- Simple might not have the desired features
  - Additional directives may be warranted
    - Checksum support
    - Networking
      - Pre-fetching, blocksize, metadata location, cache bypass
    - Multiple caches for scalability
  - Some may be site specific
    - Dependent on the hardware being used

Simple is squishy

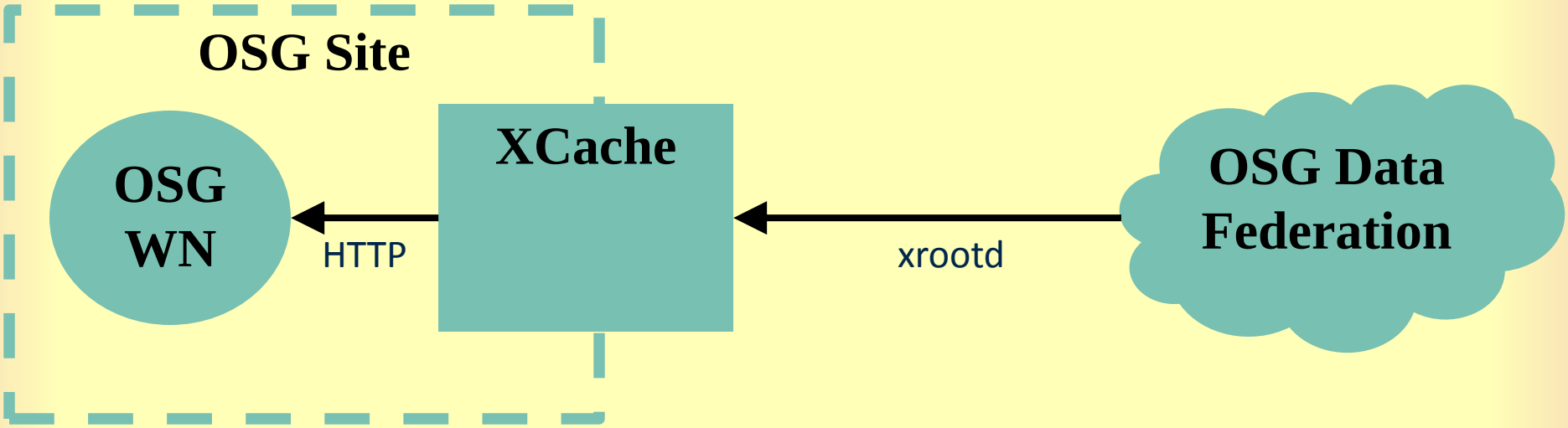
- Containerization implies standardization
  - We can boilerplate **Xcache** directives but…
  - Specific directives now tied to the container definition
- Tiered caches
  - Allows simplification at the edges, but…
  - Gets more complex as you go up the tree
- This is a hard but not unsolvable problem
  - Let's discuss!

XCache, StashCache, and LHC

- What's the difference between all this jargon?
  - **XRootD**: Flexible software framework we know and love.
  - **XCache**: XRootD configured to proxy and cache data.
  - XCache has three different deployments:
    - **ATLAS XCache.**
    - **StashCache**: OSG caching infrastructure.
    - **CMS XCache.**

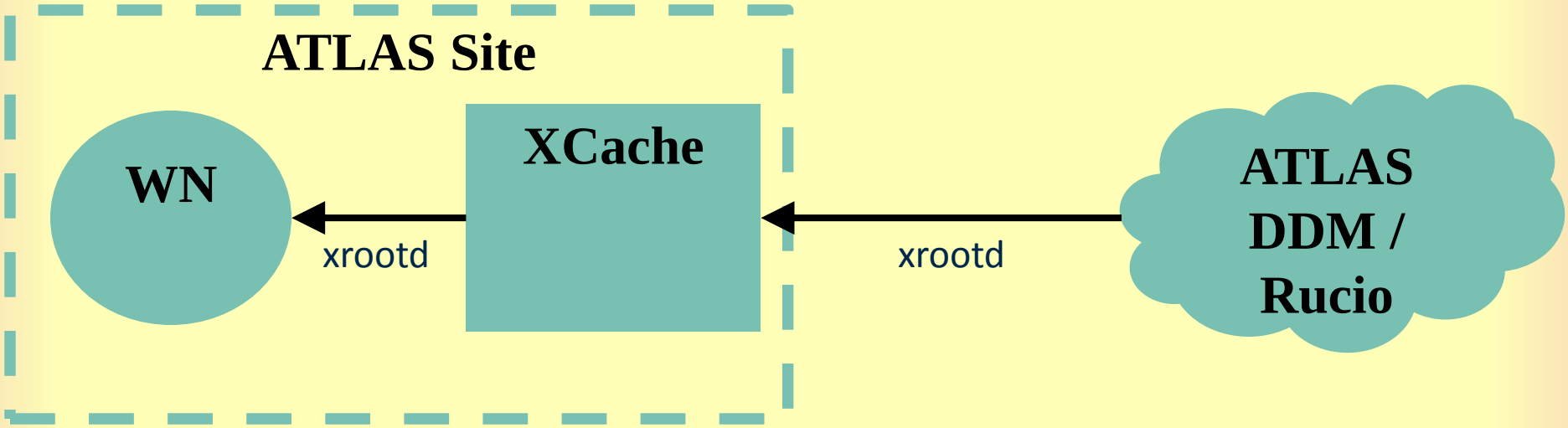
Frontends, backends, and more.

- At a high level, the difference between deployments is the frontend and backends.
- StashCache:



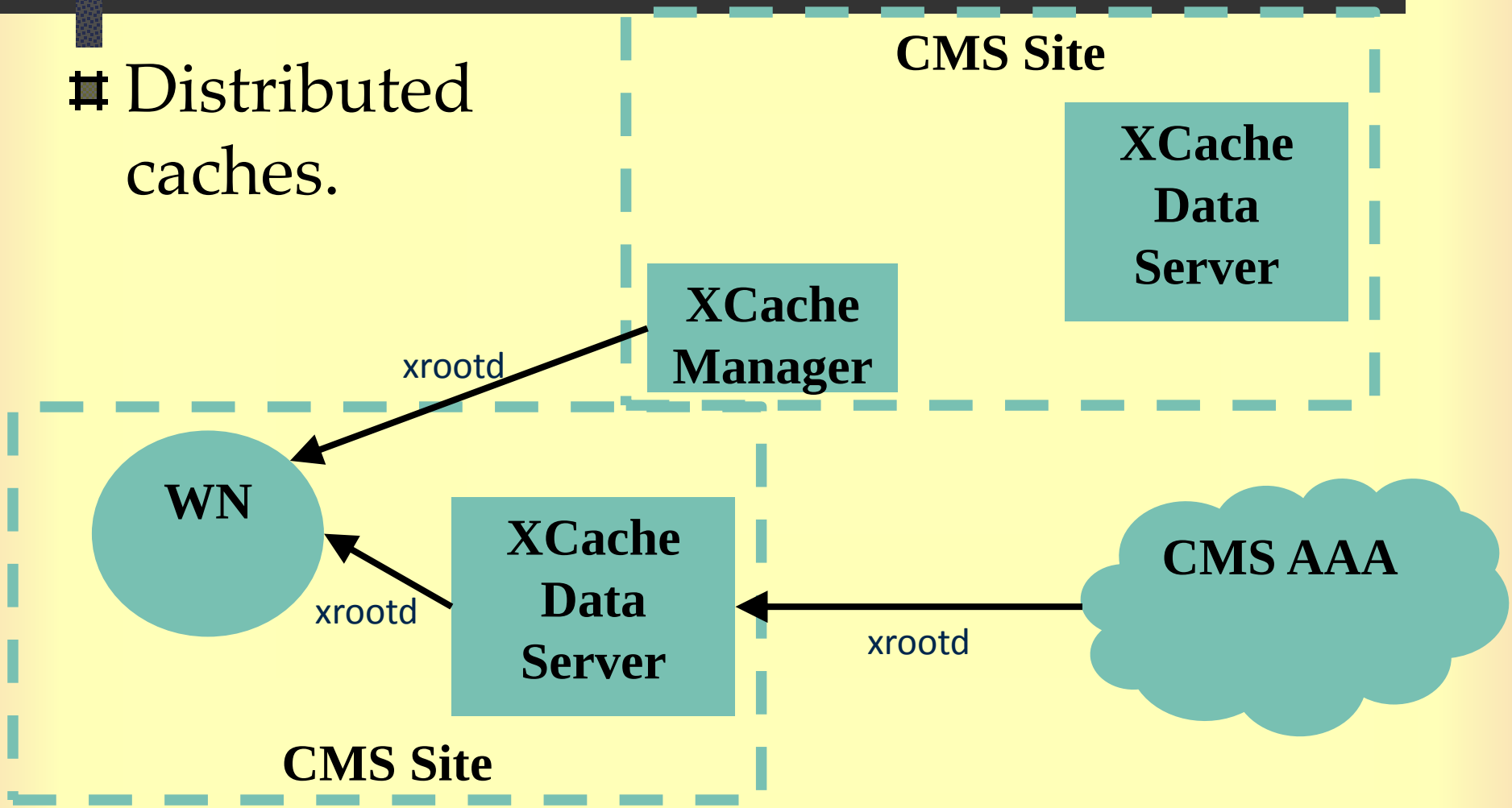
Frontends, backends, and more.

- There are also differences in authentication, monitoring infrastructure, and authorization.
- ATLAS XCache:



CMS XCache

- Distributed caches.



Where should we be headed?

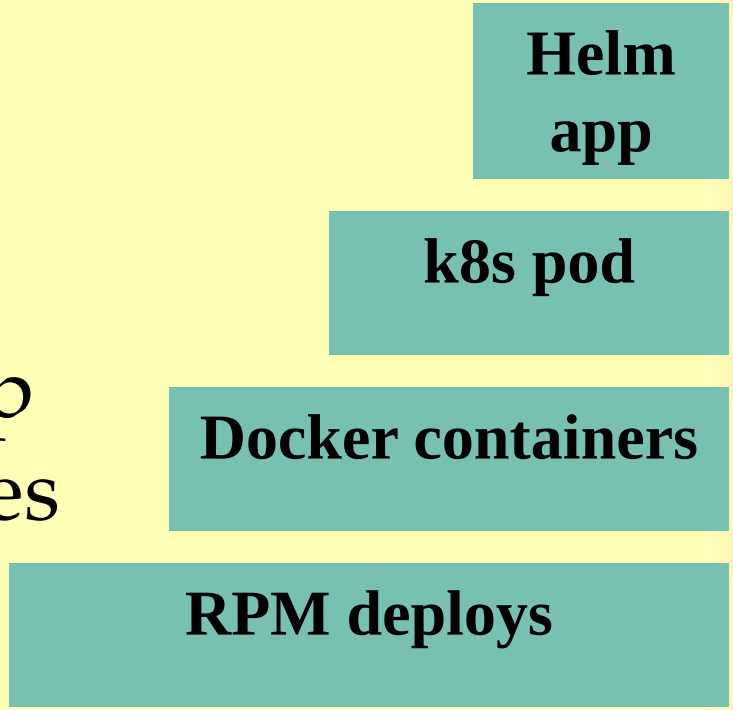
- I believe it is possible to have a single configuration of XRootD that can participate in all three XCache deployments:
  - The namespaces are all distinct.
  - Plugins shouldn't interfere with each other – Rucio integration is simply a N2N.
  - No conflicting authorization technologies.

How do we get there?

- **Documentation**: Write down how the different architectures work, how these are configured, and how to deploy.
  - Make sure someone could go to https://opensciencegrid.org/docs/ and end up with a working XCache.
- **Packaging**: Capture configurations into the packages themselves.
- **Software delivery / operations**: Start working on shortening the delivery pipelines.

Deployment Strategies

- I think the different packaging strategies as building on top of each other.
- Need to keep all layers functional. As you go up the stack, release becomes more targeted.



Deployment Strategies - Details

- XRootD 4.9.x introduces the ability to "include" config.d-style directories, meaning we can more easily layer configurations.
  - I.e., separate "XRootD base," "XCache base", and "ATLAS XCache" into three distinct layers.
  - RC1 RPM is now available in OSG repos.
- The SLATE team and PRP team have both been developing XCache pods

Places we could use some help

- I'd like to see a shortened pipeline between XRootD tag and OSG release.
  - It might also be useful to think about release cadence – last feature release was Dec. 2017.
- Currently no clear information flow for how new configs from VO into OSG.
- How should the monitoring deployment work?
- First step of this is evolving the StashCache meeting is evolving into a more generic XCache meeting.
  - Thursdays @ 1PM central.

Take-Home Message

- Over the course of 2018, I think we've done the R&D and integration on:
  - How each use case should be deploying XCache.
  - Battle-hardening the XCache code itself.
  - Determining how to fit XCache into new deployment scenarios.
- With the upcoming 4.9.x, it's now take this into production!