XCache(s) DevOps in OSG

Diego Davila, Derek Weitzel, Marian Zvada

WLCG GDB, May 12th, 2021

XCache

Caching data with XRootD, concept with aim to:

- reduce WAN traffic
- reduce latency and increase CPU efficiency
- cost less to run (based in implementation and operation)

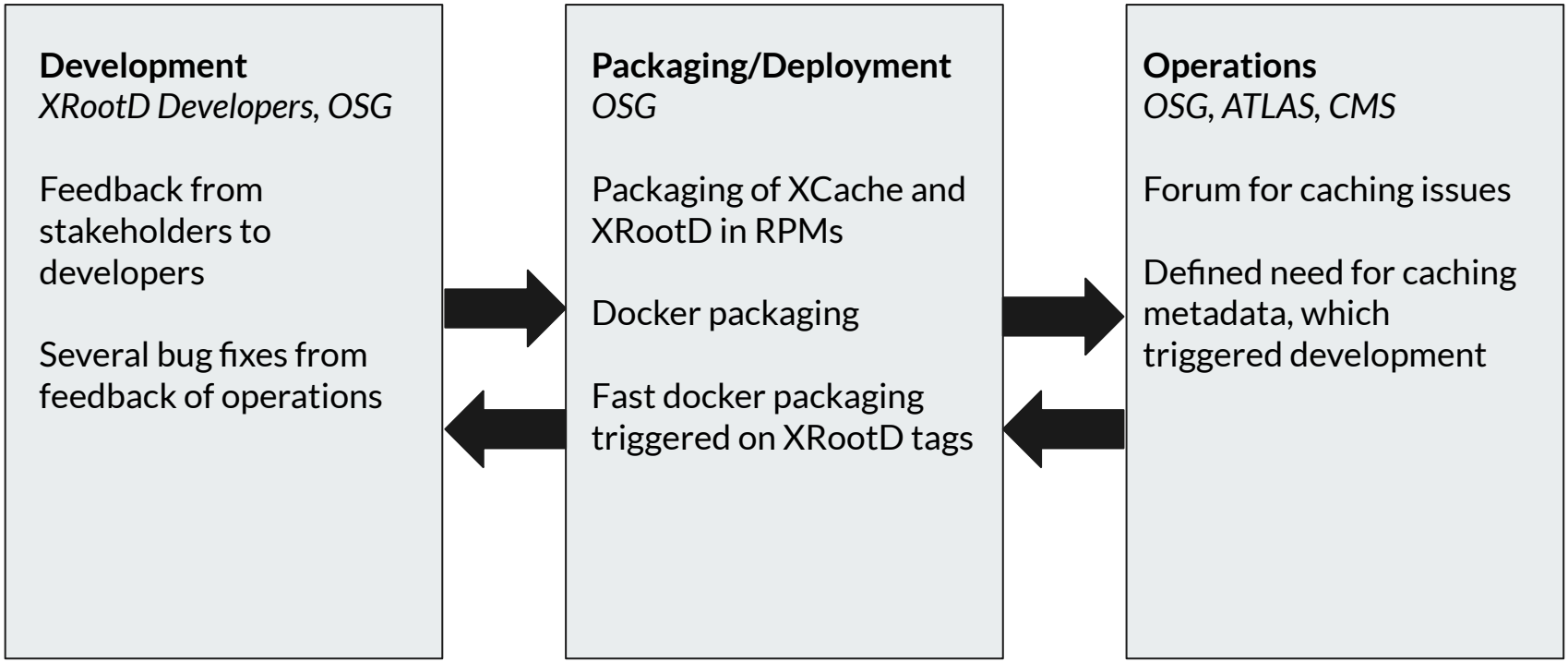
Uses the proxy file cache (**PFC**) plugin to XRootD

First note on StashCache project dates to March 2015, only OSG-centric until 2018

We established weekly meetings with stakeholders (CMS, ATLAS, OSG,…) and the XRootD developers to provide feedback, discuss bugs, and present findings

- Since Dec 2018: xcache@opensciencegrid.org

XCache Members



XCache Implementations

- **ATLAS:**
  - Deployed caches at all US ATLAS sites
  - Individual caches
  - Added logic in Rucio to “mimic” data placement
- **CMS:**
  - Larger pools of cache nodes serving a region
  - Redirect subset of data to use the caches
- **OSG**: Public data for general purpose computing
  - Built for ease of user experience
  - No frameworks to integrate
  - Command line tools

CMS XCache Implementation

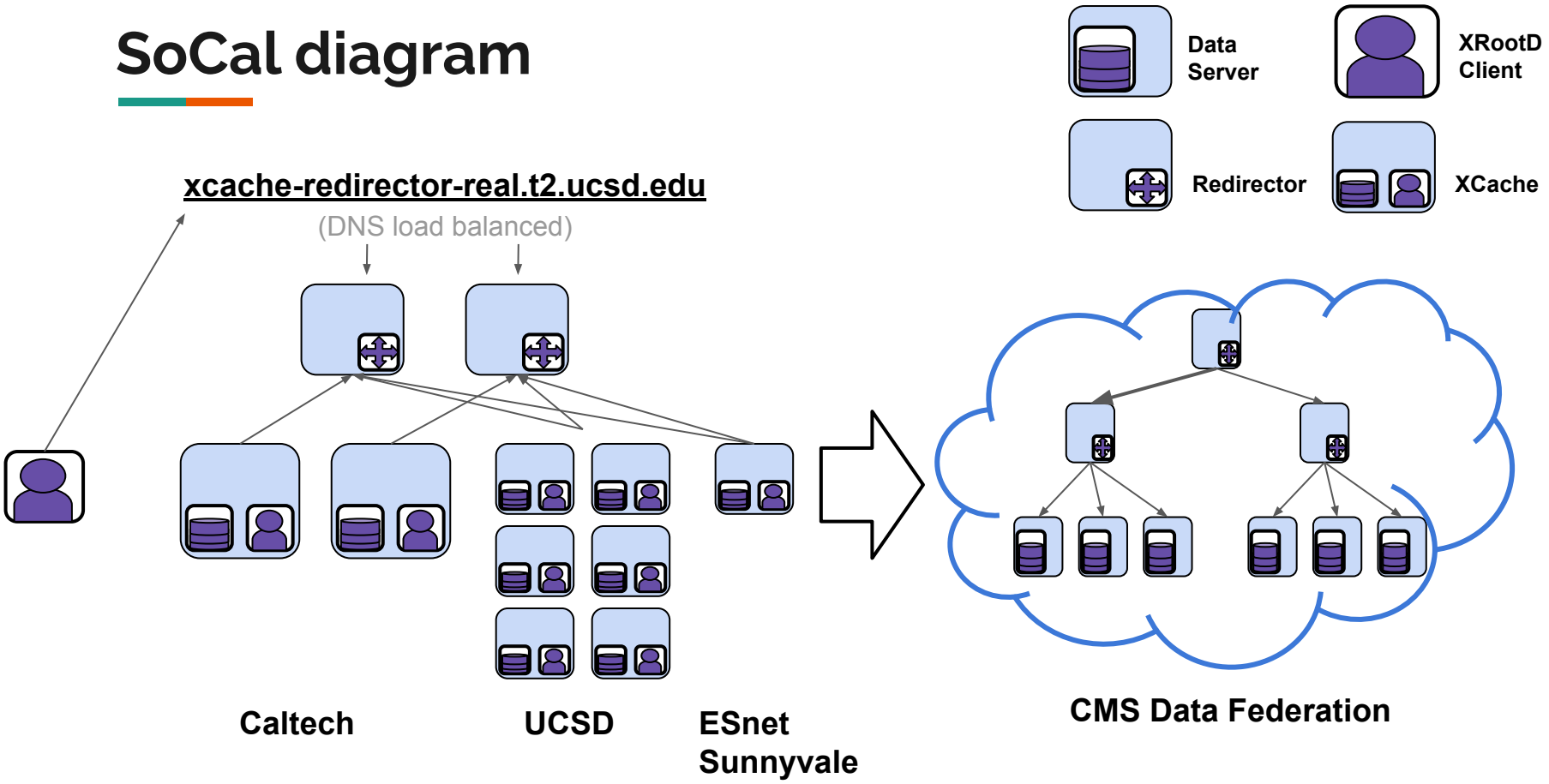
To the best of our knowledge[*] there are 3 main deployments of XCache within CMS:

1. The Southern California (SoCal) cache in the US
2. CIEMAT and PIC caches in Spain
3. A significant deployment in Italy

1. The SoCal cache is a collaboration between the Tier 2s at UCSD and Caltech and ESnet with a total of 665 TB deployed. The jobs are driven into the cache via a special overflow group in the Global Pool Frontend. At the site the jobs are pointed to either the cache or the local storage using special rules in the TFC. The namespace currently cached is MINI*

[*] Apologies if we are missing other deployments. Please contact Diego Davila (didavila@ucsd.edu) if you have an XCache deployment for CMS not mentioned here

SoCal diagram



CMS XCache Implementation

In Spain, 2 caches have been deployed, one in CIEMAT with 22TB, used by only 2 WNs for fallback and one more in PIC with 130TB used by a single WN and no restrictive namespace, i.e. everything is cached

Currently:

- Ongoing effort to **investigate and deploy a monitoring system** for the caches either by using the internal cinfo files of XRootD (Currently used by ATLAS) or by fetching information from monIT at CERN.
- Also using the monIT records of CRAB jobs, **a study in Cpu efficiency** was carried out showing a 6% improvement by reading from the cache.

Next:

- Evaluate the multi-site/regional XCache benefits in the region (interaction between XCaches)

[*] Apologies if we are missing other deployments. Please contact Diego Davila (didavila@ucsd.edu) if you have an XCache deployment for CMS not mentioned here

CMS XCache Implementation

In Italy there is also a significant deployment of XCache for CMS but we do not have updated data to show (Apologies)

All these caches get their data from the CMS Data Federation (via AAA)

[*] Apologies if we are missing other deployments. Please contact Diego Davila (didavila@ucsd.edu) if you have an XCache deployment for CMS not mentioned here

ATLAS XCache Implementation

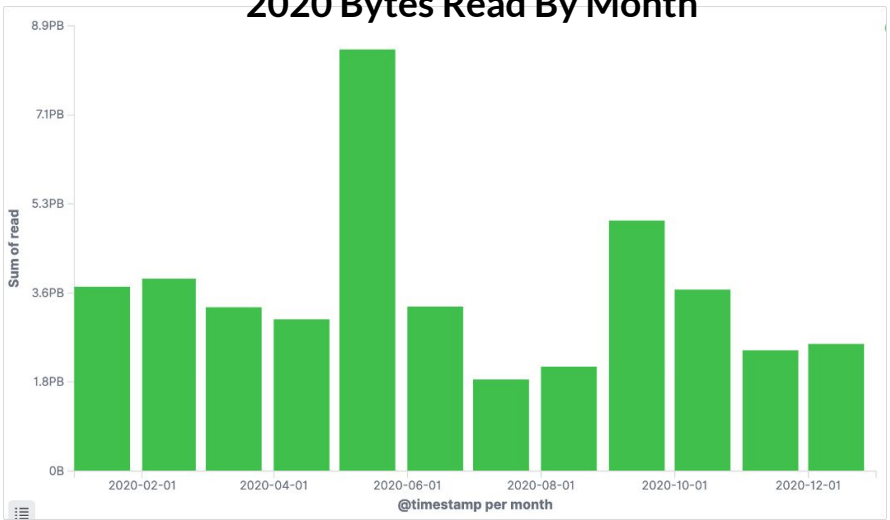
Last 6 months: Data read - **2.7 PB**

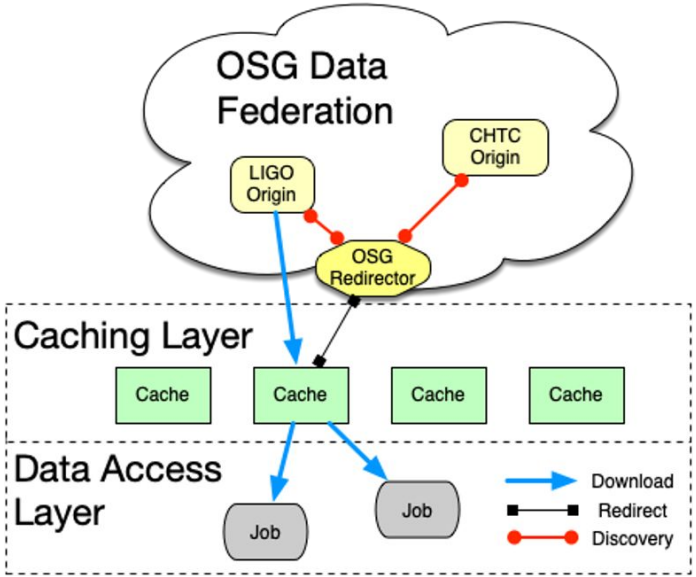
- Deployed an XCache at each USATLAS site using Slate, and in Europe using other methods…
- Use Virtual Placement Service (VPS) to place virtual data at sites within Rucio. Panda will scheduled jobs at sites with the virtual data. Rucio will then prepend XCache to the file requests.
- Lead the development of Cache Metrics Stream. Provided feedback and refined the data made available on the stream.
  - Stream gives information such as what files are cached
  - Developed parser for the stream

OSG XCache - StashCache

2020 - Data read: **22 PB**

General purpose access by end users. Upload files to one of many origins, then available to their jobs everywhere!





OSG XCache - StashCache

- StashCache data is accessed primarily through 2 methods, **CVMFS** and **StashCP**
- CVMFS:
  - Provides a POSIX-like interface to data

```
$ cat /cvmfs/stash.osgstorage.org/osgconnect/public/dweitzel/blast/queries/query1
>Derek's first query!
MPVSDSGFDNSSKTMKDDTIPTEDYEEITKESEMGDATKITSKIDANVIEKKDTDSENNITIAQDDEKVSWLQRVVEFFE
```

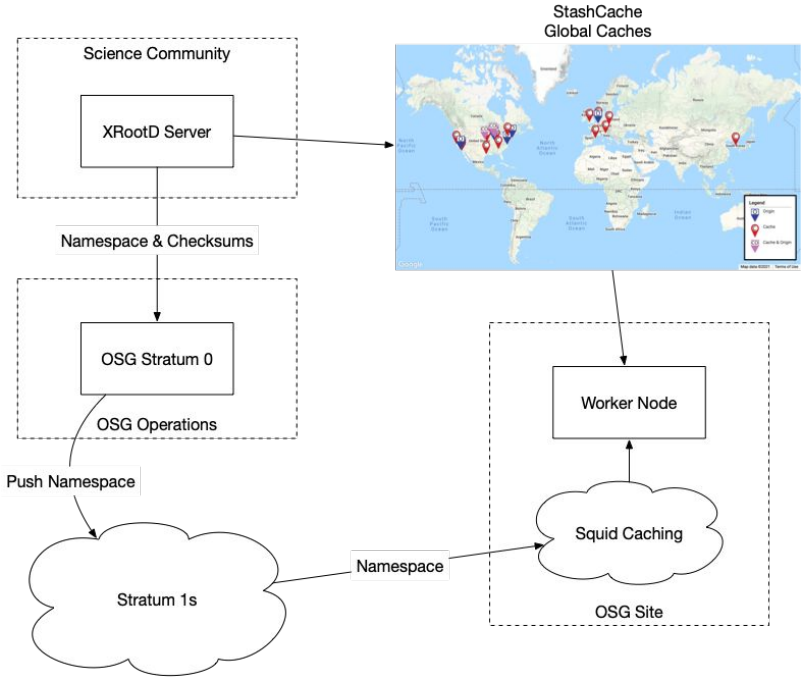
OSG XCache - StashCache

CVMFS stores the namespace separate from the data

Namespace is through cached HTTP

Data is through StashCache Federation

CVMFS can take ~8 hours to scan and update namespace with changes.



OSG XCache - StashCache

StashCP

Custom tool developed for StashCache

Data is immediately available, no need to wait for CVMFS scan

Tries multiple methods to copy data: HTTP, XRootD, and CVMFS (if available)

CVMFS is not required, therefore more available resources

```
$ stashcp /osgconnect/public/dweitzel/blast/queries/query1 ./
```

OSG XCache - StashCache

Both **CVMFS** and **StashCP** use GeoIP to determine nearest cache to use.

Both clients will fall back to other near caches if the selected cache does not respond or is too slow*

Disclamer: Client domain is difficult to capture and will not capture clients from all clusters

Manhattan Cache Feb. 2021

Client Domain	Bytes Read
uconn.edu	37.8TB
amnh.org	6.7TB
syr.edu	1.9TB
org.br	372.1GB
verizon.net	36.9GB
mit.edu	5.3GB
pic.es	5.2GB
rutgers.edu	3.5GB
ac.uk	1.5GB
in2p3.fr	898.2MB

OSG XCache - StashCache

5 Internet2 hosted caches – Kubernetes

7 U.S. OSG Contributor hosted – Mix of Kubernetes and RPM

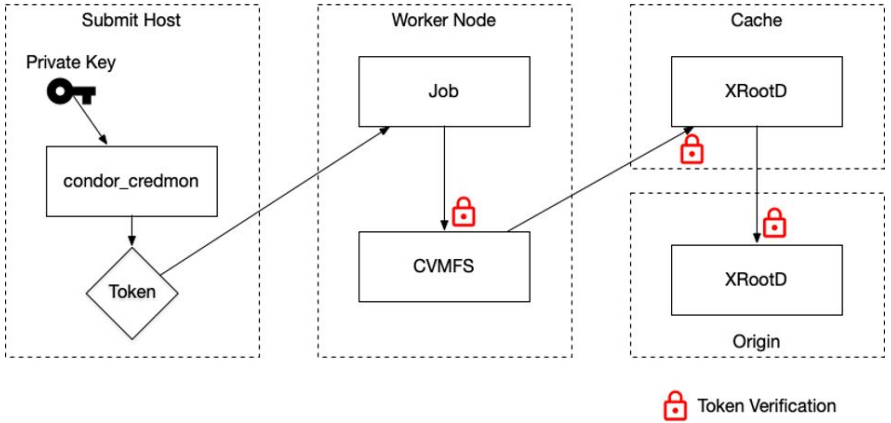
5 Caches in Europe – Kubernetes

OSG XCache - Authentication

LIGO uses authenticated access to StashCache

Used x509, transitioning to SciTokens

1. Token is transferred securely with job
2. CVMFS verifies token on the worker node (cache access)
3. Token is used to request data from cache
4. Cache propagates token to gather data from Origin



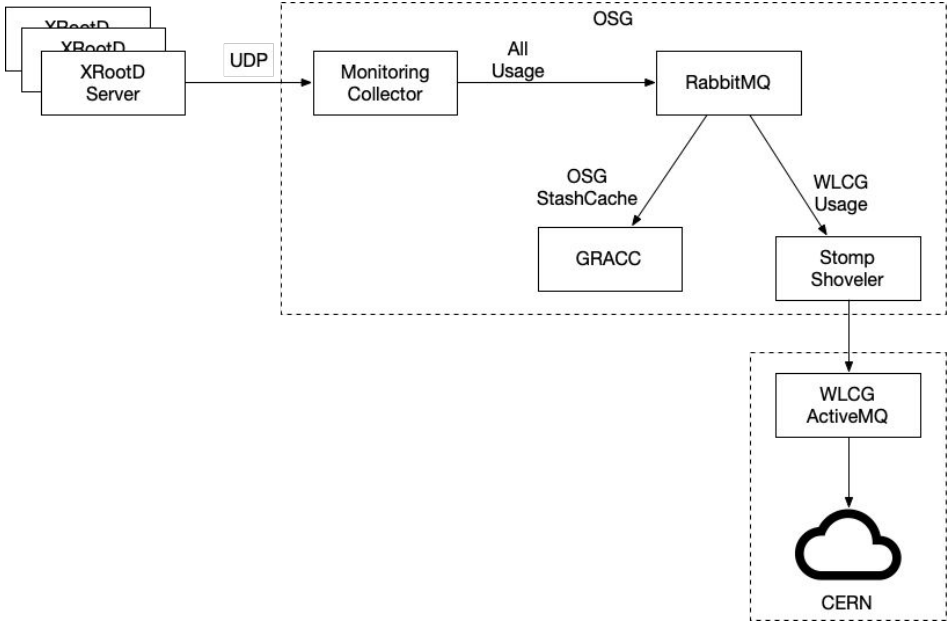
XCache - Transfer Accounting

XRootD servers send transfer stream to central collector which parses and sends to the message bus and transfer databases.

2 validations of this data pipeline:

Correctness:
https://doi.org/10.5281/zenodo.3981359

Scale:
https://doi.org/10.5281/zenodo.4688624



XCache Future

Continue to increase the number of caches

Continue to provide feedback to the developers for features and bugs

Implement more reliable file transfer monitoring

Acknowledgements

This project is supported by the National Science Foundation under Cooperative Agreement OAC-1836650 and OAC-2030508. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.