Status update
Goethe Universität

Themenbereich II

20.10.2022

V. Lindenstruth, A. Redelbach

GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

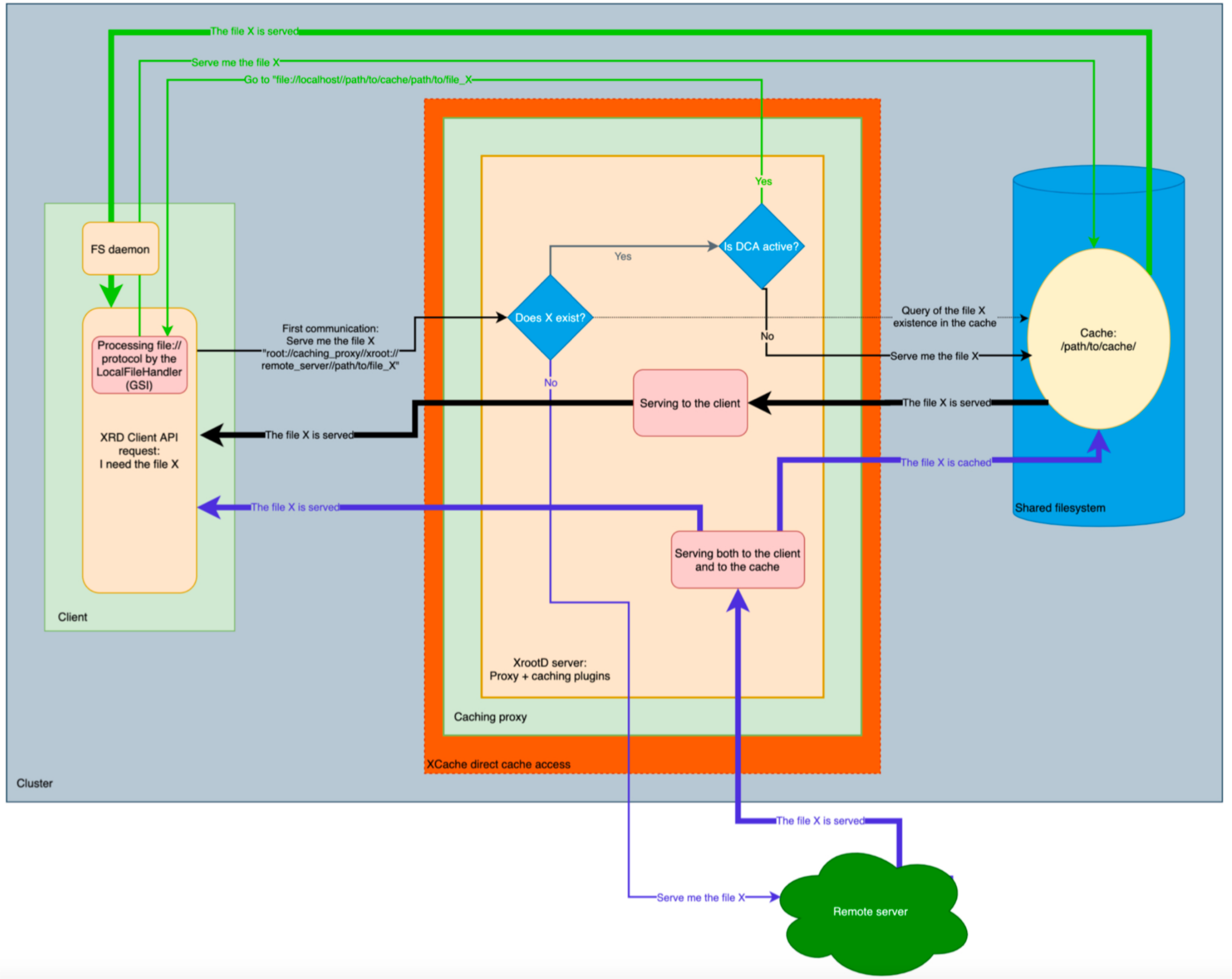
Overview

- Planned developments shall contribute to an efficient integration of caches also into new heterogenuous resources (e. g. HPC- oder Cloud- Systems).
- Furthermore these integrated data caches shall be integrated into the production systems in a transparent way.
- Significant work in comparisons between existing methods for dynamic caching in pilot phase (ErUM-IDT)
- Previous results and resources as basis for plans in context of FIDIUM
- Focus of work on first milestone: Erstellen einer vollständigen Liste der relevanten Unterschiede von XCache und “Disk Caching on the fly” und eines daraus abgeleiteten Entwicklungsplans

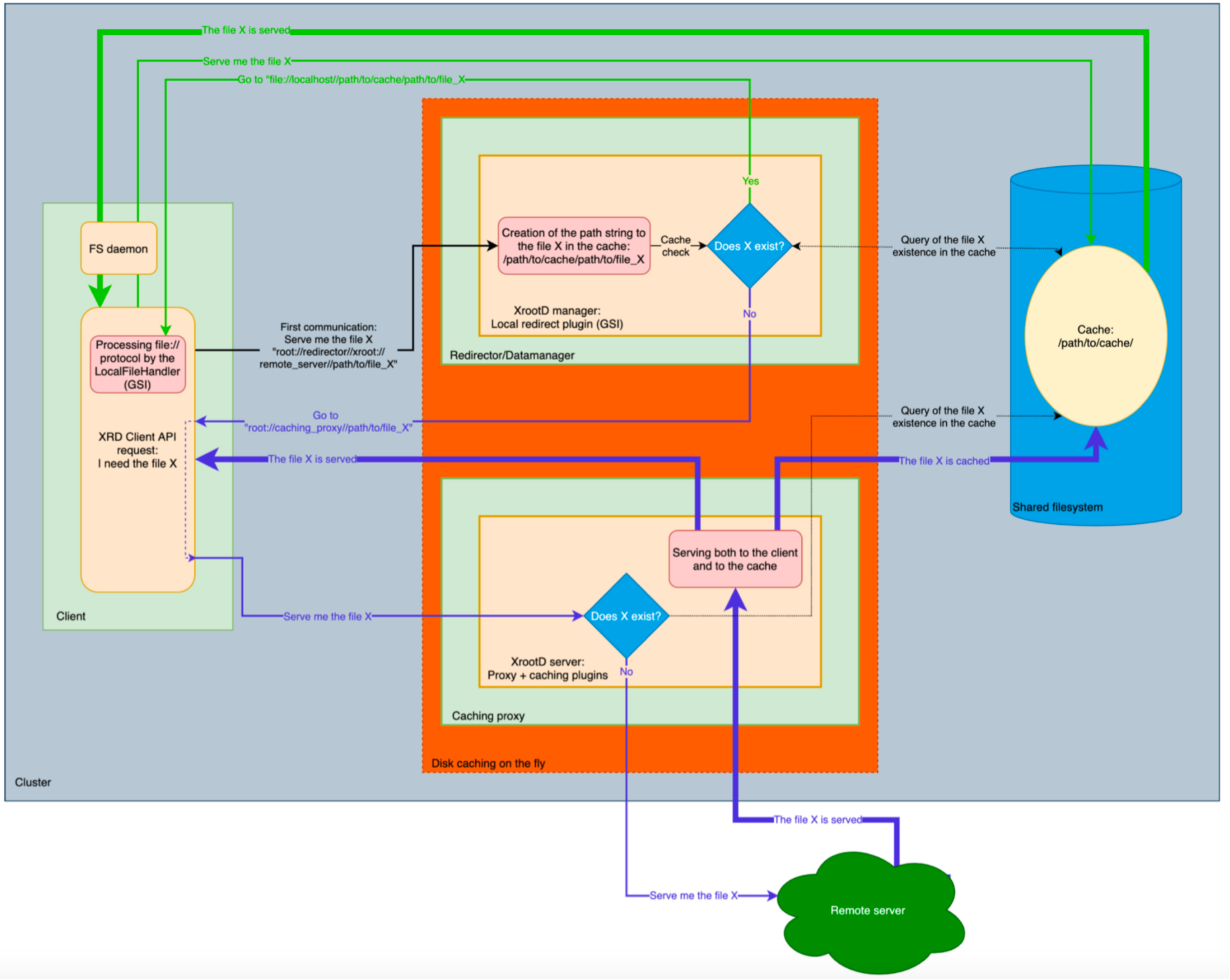
Setup for comparison of Disk Caching on the Fly – XCache (with direct cache access)

- Performance studies utilizing high-bandwidth WAN between Goethe HLR and GSI
- Access to ALICE AOD data sets (stored at GSI)
- Singularity containers used with caching system
- Measurement of runtimes and data rates for processing these data sets
- Different numbers of clients investigated (at Goethe HLR)

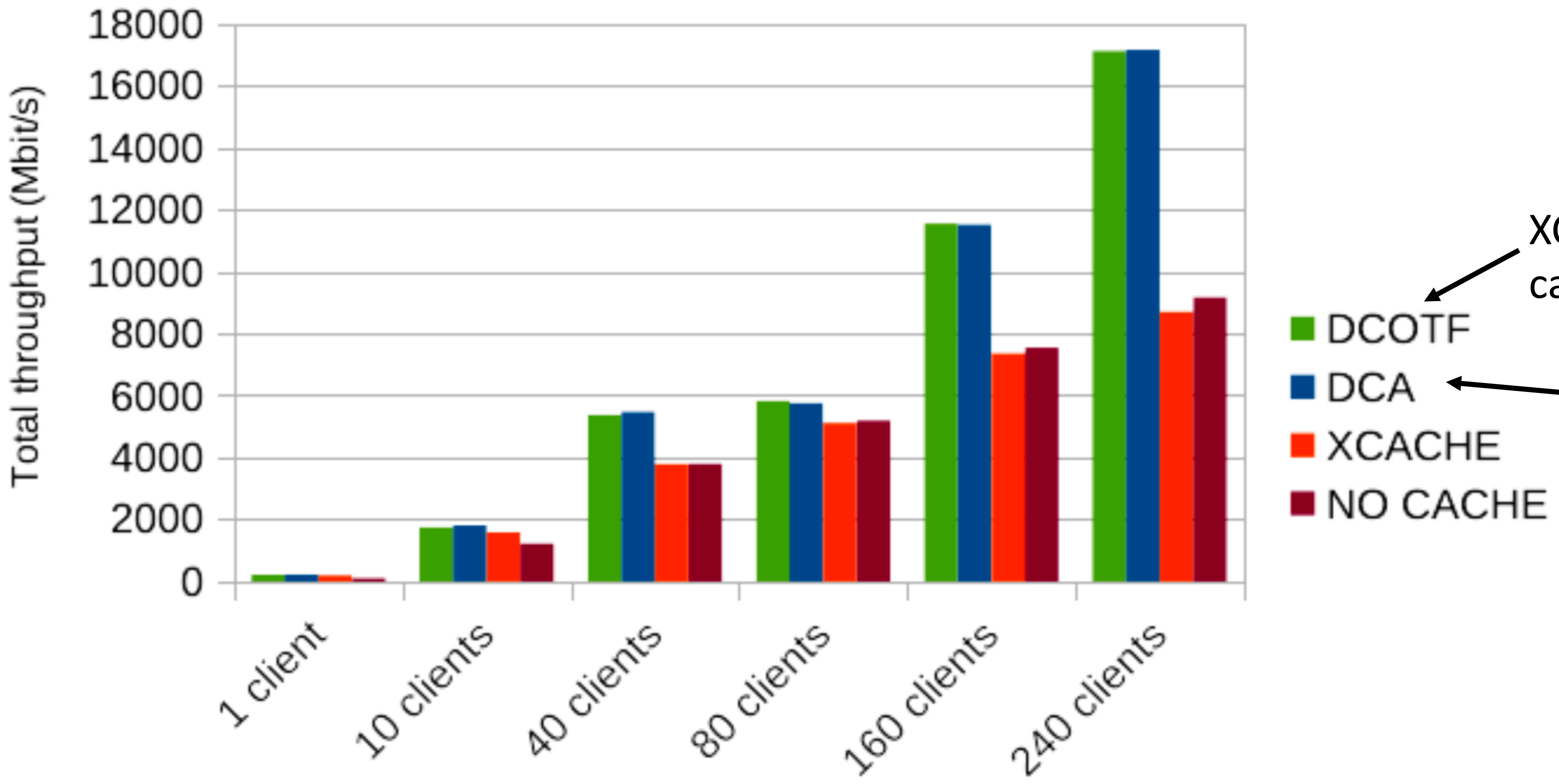
Workflows for XCache (with direct cache access)



Workflows for Disk Caching on the Fly


The file X is served
Serve me the file X
Go to "file://localhost//path/to/cache/path/to/file_X
FS daemon
Processing file:// protocol by the LocalFileHandler (GSI)
XRD Client API request: I need the file X
Client
First communication: Serve me the file X "root://redirector//xroot:// remote_server//path/to/file_X"
Creation of the path string to the file X in the cache: /path/to/cache/path/to/file_X
Cache check
Yes
Does X exist?
No
XrootD manager: Local redirect plugin (GSI)
Redirector/Datamanager
Query of the file X existence in the cache
Cache: /path/to/cache/
Query of the file X existence in the cache
Go to "root://caching_proxy//path/to/file_X"
The file X is served
The file X is cached
Shared filesystem
Serving both to the client and to the cache
Serve me the file X
Does X exist?
XrootD server: Proxy + caching plugins
No
Caching proxy
Disk caching on the fly
Cluster
The file X is served
Serve me the file X
Remote server

Performances: XCache_dca versus DCOTF



XCache with direct cache access

Disk Caching on the Fly

→ Same performances of XCache_dca and DCOTF observed in many benchmarks.

Features of DCOTF

- **Redirector** is lightweight and can run on a separate machine. In such cases each request will be handled immediately and be redirected to either the caching server or the cache.
- **UNIX permission of files**: Each file is owned and only be accessed (UNIX permission 600) by the user that runs the caching proxy. No other user can read data from the file system in case of direct access. This could be solved by editing the XRootD caching plug-in which writes files to the cache.
- Necessity of **special central management service** (cms) plug-in: One needs a plug-in developed for the project which is not natively included in XRootD.
- **No marking of an access record** when a file is directly accessed. This poses an issue while purging file.
- Redirector is **not aware of the status of the file**. It only checks if the file exists. As a consequence, there is no coherence of the cache.

Features of XCache with direct cache access

- Everything happens in a single machine. If a **caching proxy** is **overloaded** with a long queue of requests, then the client requesting data from the cache may be delayed until the server becomes available again.
- **UNIX permission of files**: There is an option to set in XRootD configuration file to enable group or world read access. Thus there is no need for editing plug-ins or a new plug-in.
- Everything is native, **only XRootD installation** is **sufficient**. Necessary changes (LocalFileHandler) have been already merged into upstream XRootD package.
- XRootD **marks** an **access record** for all kind of **file access**. Purging of files occurs according to the access records.
- Everything is managed by a **single XRootD service** which is **aware of the status of every file** and locks the files if necessary to avoid deletion of the file during transfer.

Further strategic considerations

- Close coordination with GSI group
- Earlier concepts of DCOTF developed at GSI have been integrated into XCache_dca
- Existing group of developers working on XRootD/XCache
- No strong requirements for extra future developments for DCOTF in addition to XCache_dca

Summary and next steps

Decision for further investigation and developments based on XCache with direct cache access

Investigation/optimisation of possible delays when caching proxy is overloaded with a long queue of requests (solutions to circumvent/reduce overload of redirector)

Towards realistic workflows (also from other areas of FIDIUM)

Extended set of use cases for efficient deployment of dynamic caching in terms of most relevant parameters:

- Network bandwidth
- Caching resources
- Data sets for analysis
- Numbers of clients/nodes for analysis

Consideration of results in area III (see also presentation tomorrow)

Coordination with other FIDIUM partners and XRootD/XCache developers