GridPP
UK Computing for Particle Physics

THE UNIVERSITY of EDINBURGH

A data caching model for Tier-2 WLCG computing centres using XCache

Teng LI, Rob Currie, Andy Washbrook

July, 2018

Outline

General Introduction

XCache as a transparent cache service

Performance evaluation

Next to do and Summary

Transparent cache: What & Why?

What is a transparent cache service?

- A data caching server sits closely to the computing nodes
- Invisible to the high level DM and low level apps.
- An optimization method for remote data access

Why is transparent cache needed?

- The trend
  - More lightweight sites

    ATLAS is consolidating T2s/T3s
  - More storage under common namespaces to exploit network

    AAA, FAX, Data lake
- Cache is cheap and effective

More remote data access

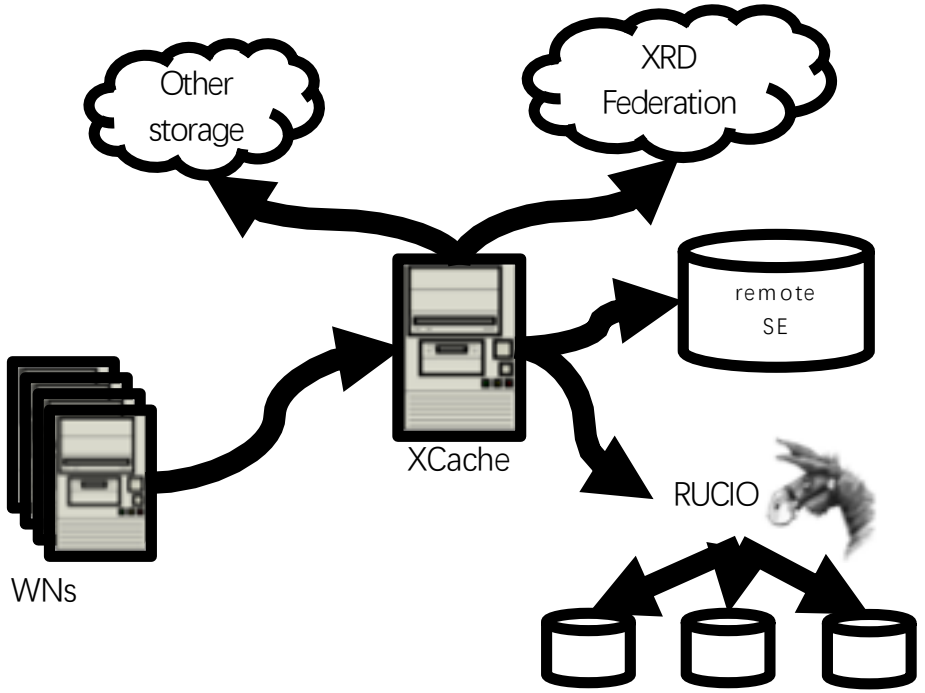
XCache

XCache = XRootD Proxy Cache

- XRootD proxy server with file cache
- Squid-like cache using root protocol

Advantages

- xroot protocol is widely used
- Both whole file/block-based partial file cache are supported
- Clusterable to scale up
- Based on a plugin architecture
  - Easy to to fancy stuff
  - Decision lib, Name lib, disk/memory cache
  - Apply different implementations for different VOs



Motivation

What do we do

- Build a test transparent XCache service with the computing/storage at our Tier-2 site.
- Integrate the test instance to ATLAS (largest consumer) production environment
- Evaluate performance

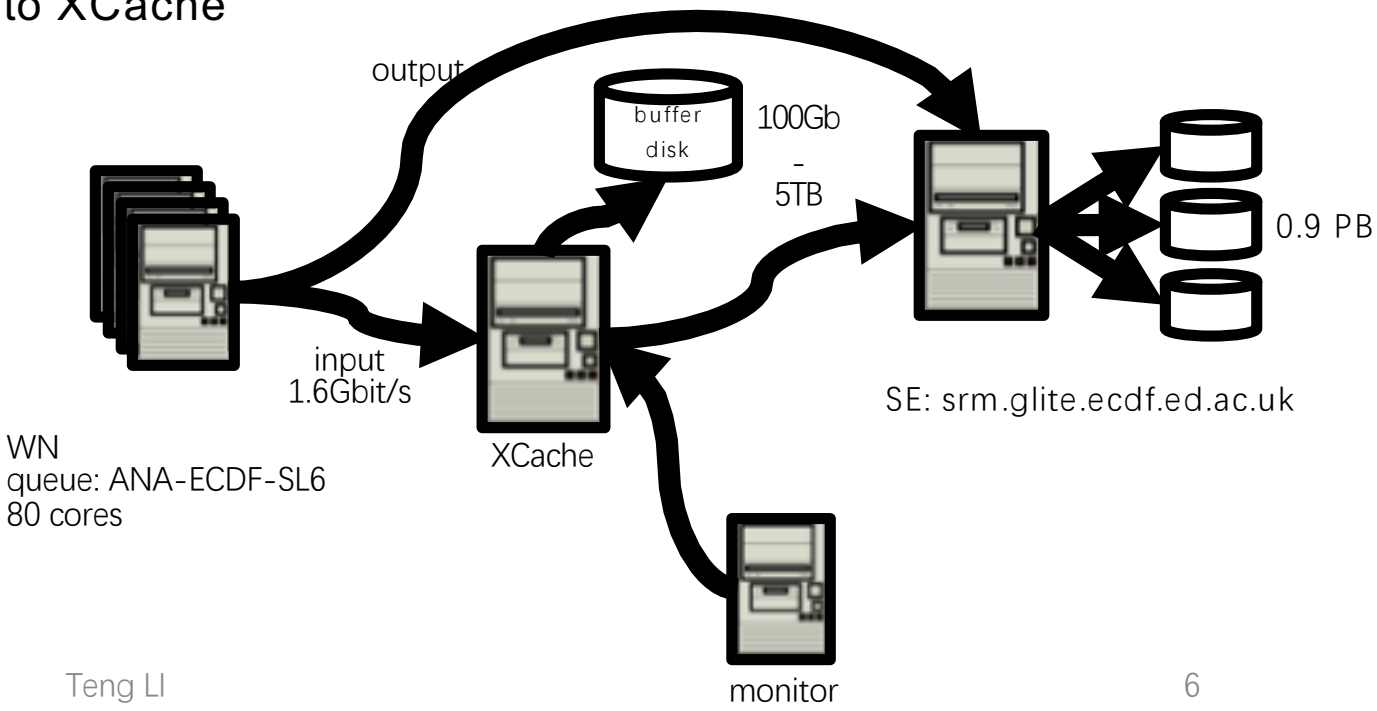
Performance evaluation

- Judge whether XCache is feasible and stable as a transparent cache
- Evaluate cache effectiveness (cache hit rate, cached file information, etc.)
- Identify issues to provide reference for future R&D work

Transparent Cache Service

Overview

- Use an ATLAS analysis queue for testing
- Simulating a CE attached to a remote SE (diskless site)
- Workflow
  - Input network traffic of WNs is redirected to XCache
  - Input files are cached in the buffer disk
  - Output network remains unchanged
  - Whole file pre-fetch mode is used
- Metrics of XCache are collected and analysed
  - Mainly for cache hit/miss info



Transparent Cache Service

XCache system Info.

- OpenStack virtual machine, 2*2.4GHz Intel Core Processor (Haswell, no TSX)
- XCache configuration

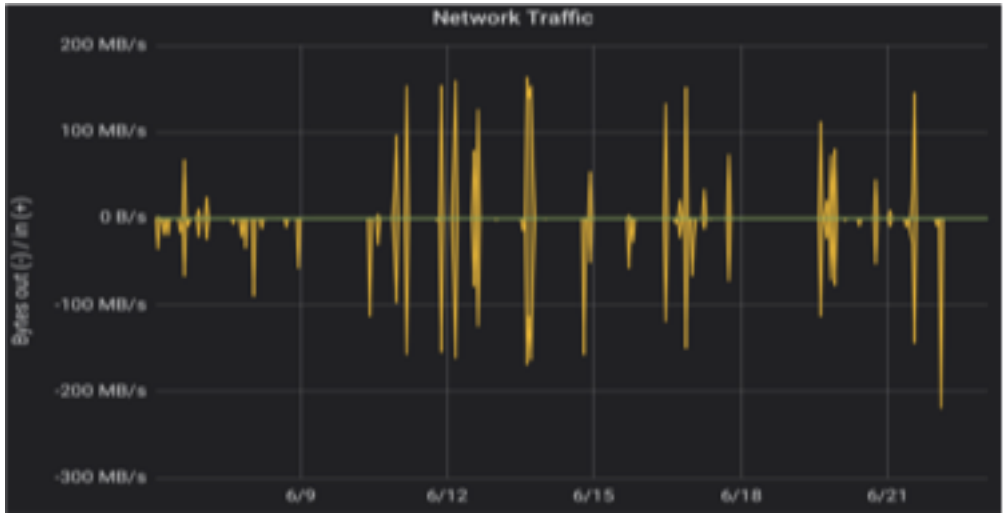
config	value	Desc.
Disk usage watermark	85% -- 95%	High/low water mark for disk purging
File block size	512 Kb	I/O is blocked in this size
Number of pre-fetching blocks	8	Number of file blocks of data fetching ahead of serving

ATLAS integration

- Since XCache is read only, a special XRD client plugin is developed and deployed on the WNs, as the only trick.
- I/O activities are separated by prefixing input url

  root://srm.glite.ecdf.ed.ac.uk/file → root://xcache.url//root:// srm.glite.ecdf.ed.ac.uk/file

General system metrics

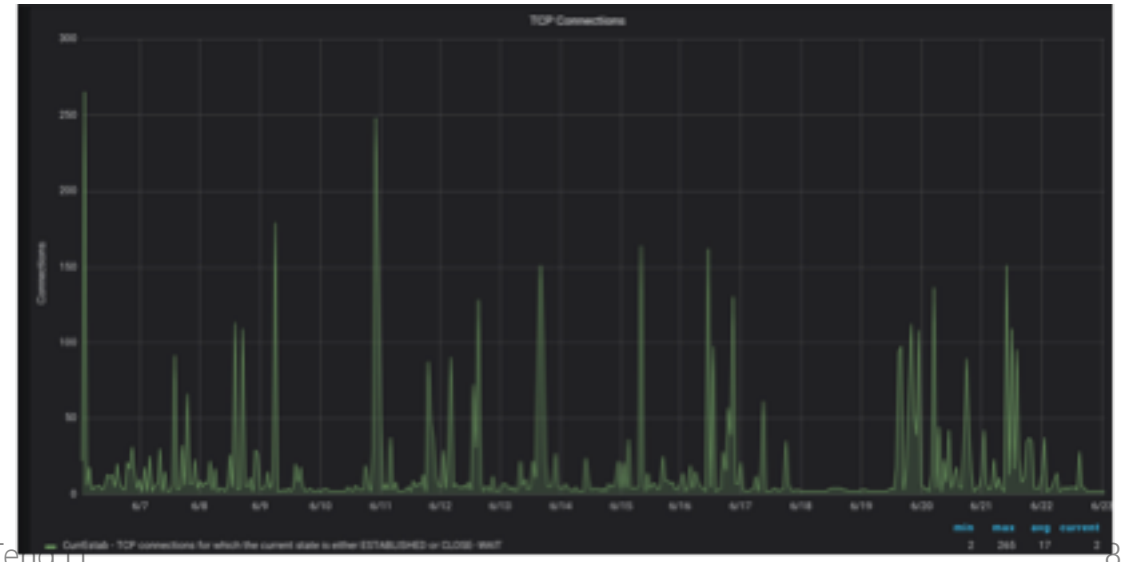


With 80 cores of frontend:

- Activities are loosely distributed over time. Network and I/O peak value could reach a couple of hundreds of Mb/s

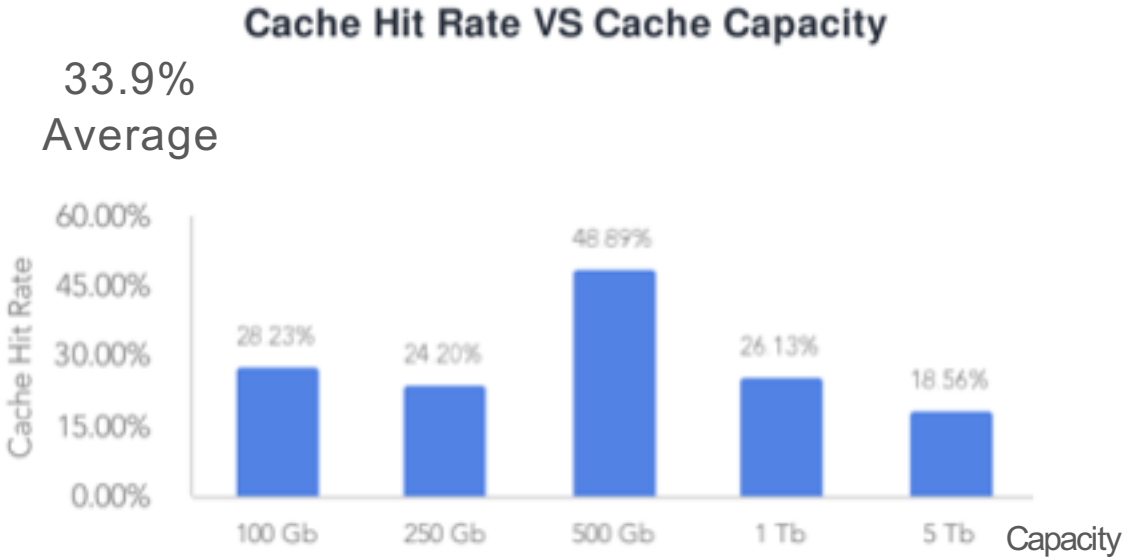
System is quite stable

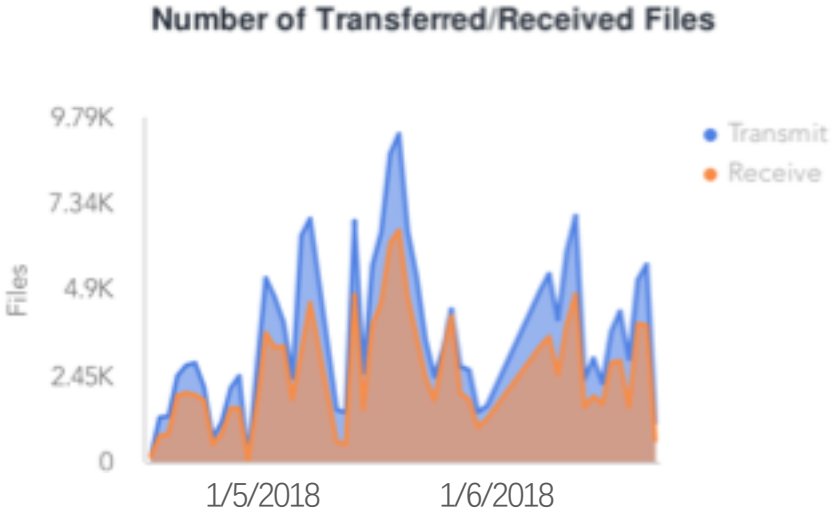
- Crashed 2 times in 4 months. Recovered with restarting

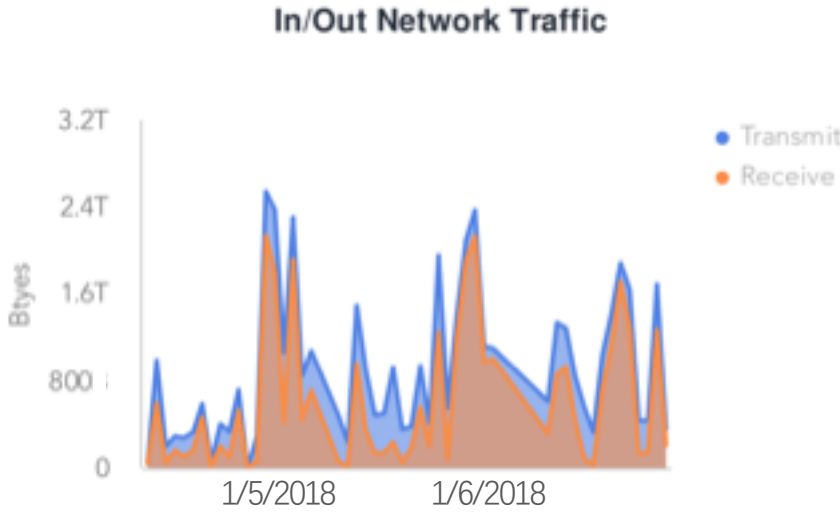


Cache Hit/Miss

- 4 months of data is taken to measure the cache performance
- Average cache hit rate is 33.9%.
- As XCache tends to purge the coldest file on disk, larger capacity does not provide better performance. 500Gb (6.25Gb per CPU core of the queue) is found to have the best performance in our case



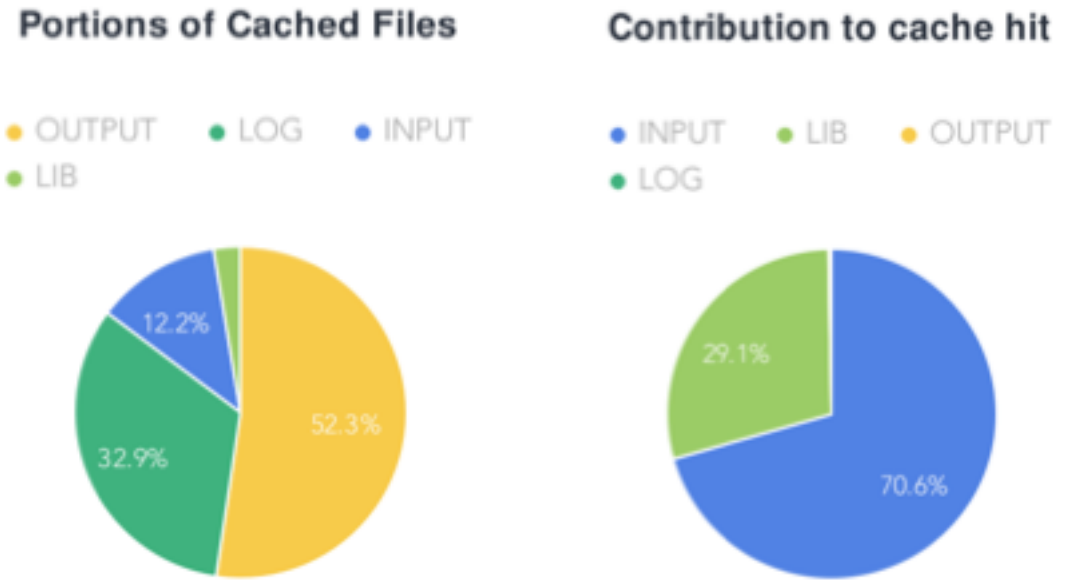




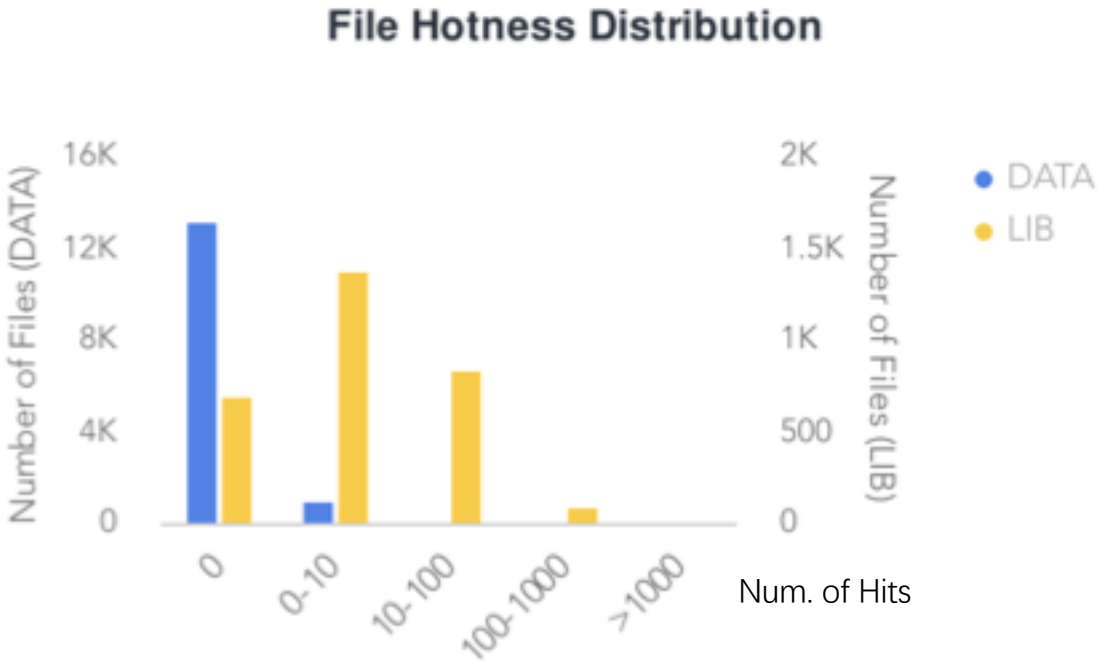
Let's take a closer look

4 kinds of files are cached:

- **input**: input data files (AOD, DAOD, …)
- **output**: user output
- **library**: user library files (dispatched by panda)
- **log**: job log files



Libraries and input data files make almost all the contributions

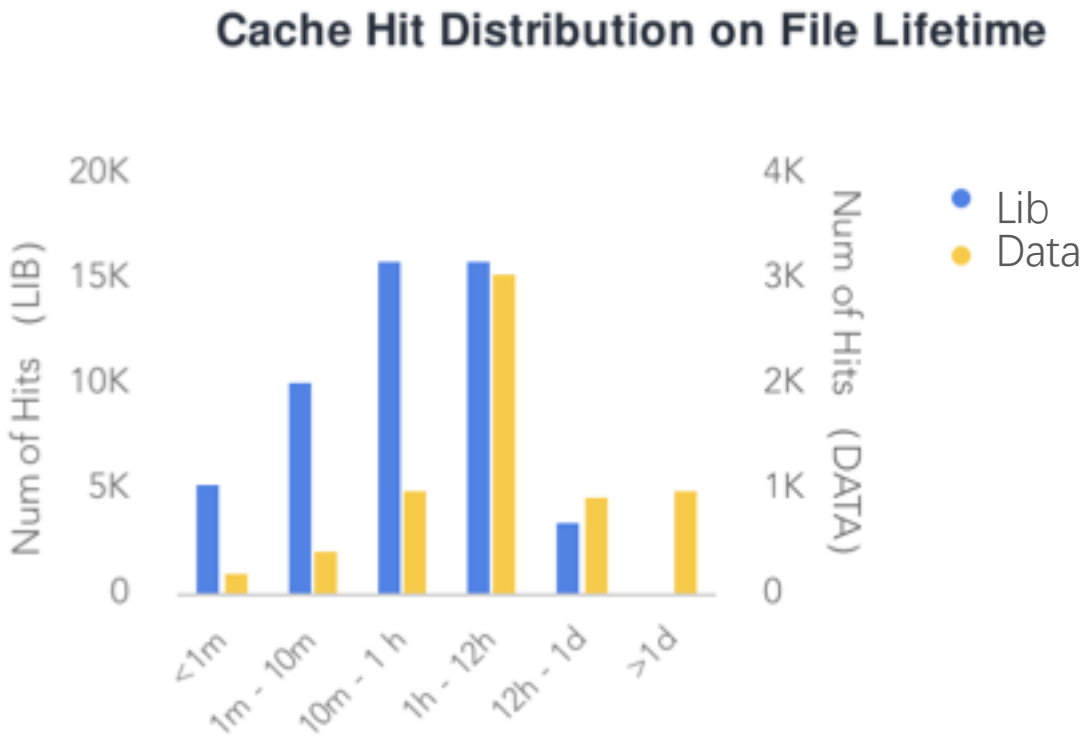


Library and data files behave very differently.

Library files are usually hot

Most data files are relatively cold, but the minority still makes 2/3 of the contribution.

Let's take a closer look



Files (especially libraries) are usually hot in the first 12 hours of their lifetimes.

Summary:

- System is fairly stable during the test
- Average cache hit rate is 34% for ATLAS analysis jobs (could be better with optimized configuration and well-defined decision library)
- As hot files are usually hot during the first 12 hours, and XCache tends to remove the cold files, relatively smaller cache capacity and more aggressive purging policy are recommended.
- A decision library is recommended to filter cold files (logs, user output)
- Memory-based cache is recommended for libraries, since they take very small space and are usually very hot during a short time period

Looking forward

Identified R&D work of XCache

- Proxy forwarding

  Use clients' credentials to authenticate with the backend storage
- Multiple-VO support

  Different implementations for different VOs
- Writing-support

  Make it easier to integrate with current workflow

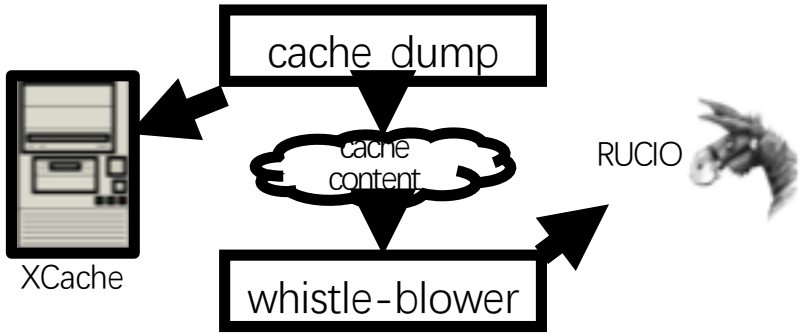
Utilization of partial file cache

- For analysis jobs, usually only very small fragments of input files are needed
- Need to corporate with Athena devs.

Looking forward

Integration with RUCIO

- XCache can act like a volatile Rucio Storage Endpoint
- Cached content is listed and reported to RUCIO
- RUCIO and job-broker is aware of the cached content
- Could be a solution of future cache-only site



Some tests are ongoing at Edinburgh

- XCache server & whistle-blower are deployed
- Volatile RSE is defined (UKI-SCOTGRID-ECDF-XCACHE). RUCIO is ready to listen

Some policies need to be defined

- Cache publication
- Job brokering

Summary

Cache for ATLAS analysis queue is tested

- To simulate CE attached to a remote SE
- XCache is stable, easy to deploy and maintain
- Network amplification is measured

  A very modest cache capacity is able to provide decent network amplification
- Cached file content is analyzed

  Several recommendations are made for better performance

Some desired R&D work is spotted

XCache & RUCIO integration is being tested

Questions?