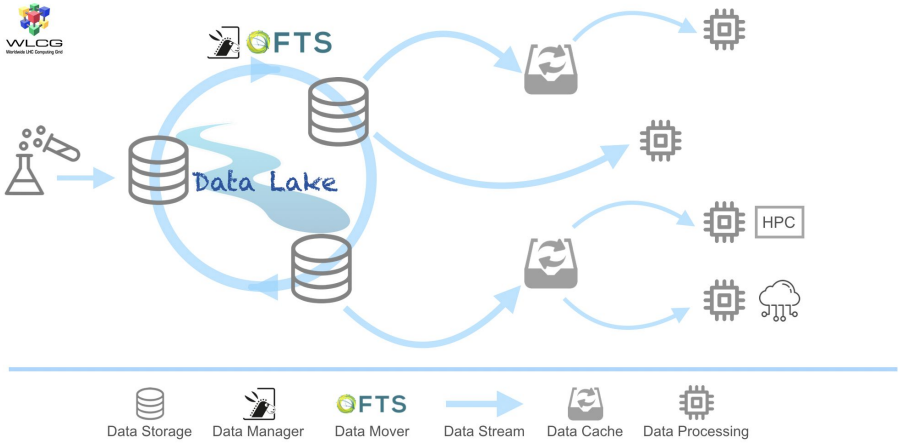
XCache & Virtual Placement

Ilija Vukotic
University of Chicago

2020-03-16

Caches



- Reduce WAN traffic
- Reduce latency / increase CPU eff.
- Cost less to run (in terms of person-power)

Issues:

- They work only if files are accessed multiple times. Cache efficiency expressed as cache-hit-rate. Unlike Netflix, HEP data is not very frequently reused.
- Current job scheduling of jobs “to where the data is” does not work for caches as caches would never get populated.
- We still use multiple protocols to move data around.

Caches - continued

Several ways to deploy them:

- Consider WN local disk as a tiny cache. We had that system (pcache) in use but now had to be reimplemented. A lot of testing needed. Expected cache hit rate 15%.
- Small site (in terms of CPU) without pledged storage, that is “far” from a large storage site (in terms of distance and/or throughput). Relatively easy to set up. Hard to keep running. Making these sites run only EVGEN is simpler solution.
- Large site/HPC without pledged storage. Can not rely on one (closest) site for all of its data. Need a high cache hit rate to reduce WAN need and still have high CPU eff.

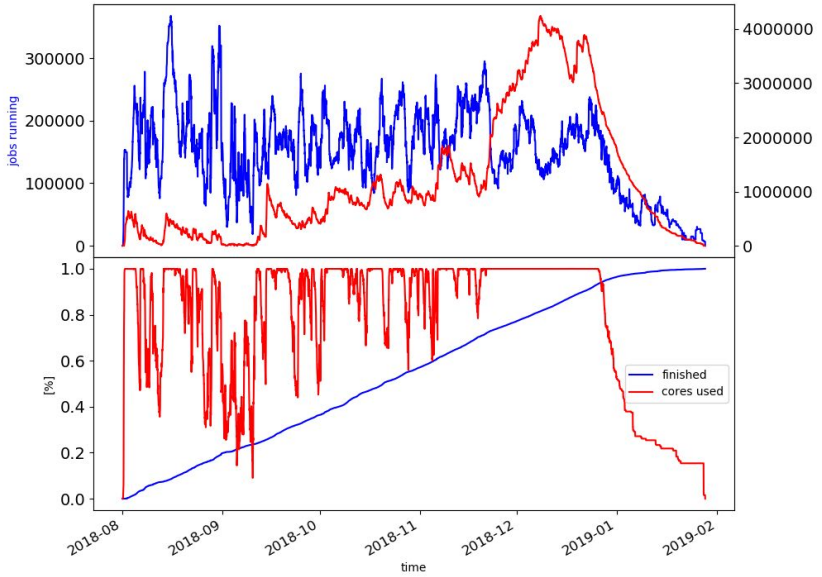
VP - Virtual Placement

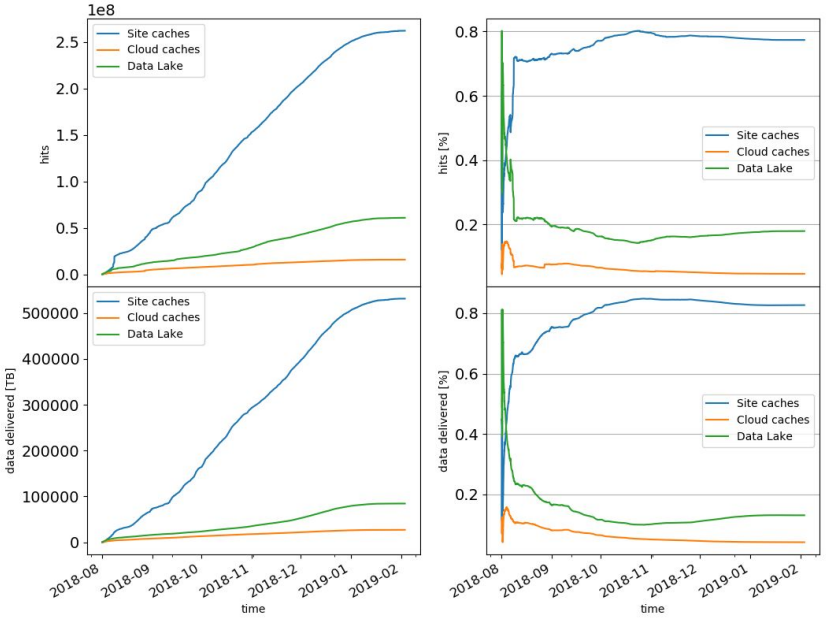
1. On registration in RUCIO every dataset gets assigned to N sites in the same cloud.
2. Assignment is done randomly where each sites probability to get the dataset is proportional to fraction of CPUs that the site contributes to ATLAS.
3. Datasets are not actually copied at all at these 3 sites but only exist in the “lake”.
4. Panda would assign job that needs as an input this dataset to the first site from these 3. In case site is in outage it would get assigned to the second site from the list. Once job is there it would access the data through the cache.

This way we get:

- Very high cache hit rate
- We could use a large fraction of the existing storage as caches
- Reliability
- Adding/removing site would be easily done from a central location
- Less stress on FTS, fewer RUCIO rules (neither needed at xcache-only sites)

VP - expectations





Resources would be fully used. TTC would be the same as now. Cache would deliver 80% of data. Throughput at caches would be reasonable.

VP to two sites of same cloud
One Data Lake (has all the data)
Each cloud has XCache (100TB/2k cores)
Each site has XCache (100TB/1k cores)

IRL Tests - configuration

XCache storage (all except BNL are spinning disks):

- MWT2 16 x 12TB (JBODs)
- AGLT2 8 x 8TB (JBODs)
- Prague 2x44TB, 2x37TB, 2x19TB (6 RAID arrays)
- BNL (NVMe)

VP settings:

- 2k/250k datasets to MWT2 and AGLT2
- 1k/250k to Prague.
- 2k/250k to BNL.

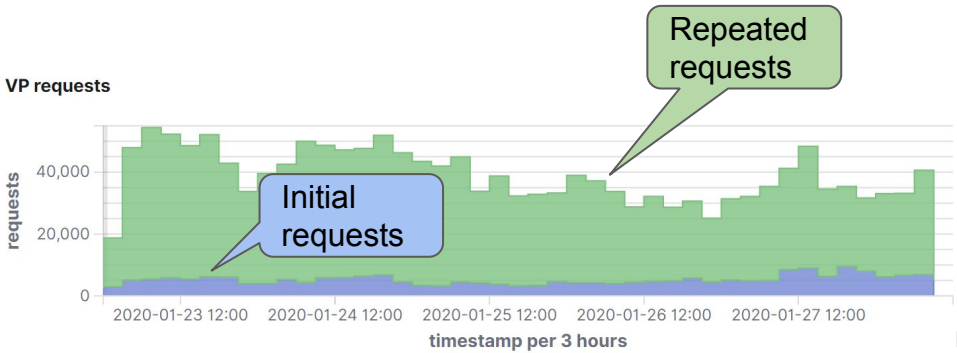
Issues

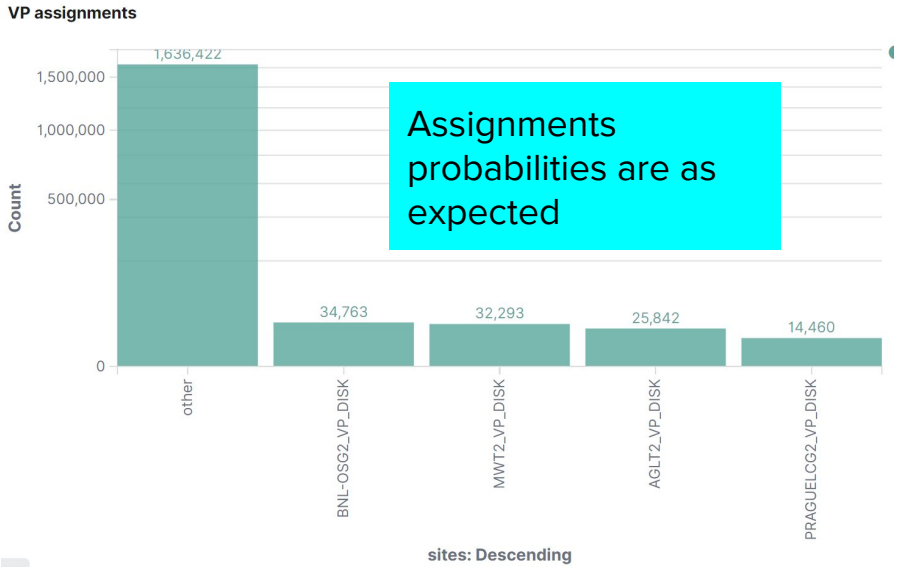
- XCache stability ✔️
- Bad origins ☑
- Which queue can get job that can use VP replicas ✔️
- Copy-to-scratch handling ✔️
- Unavailability of data in origin DDM ☑
- Sites not having xroot as a primary protocol for WAN reads ☑

Does VP service work? Yes!

VP service has been instrumented so it reports all requests and replies to ES@UChicago.

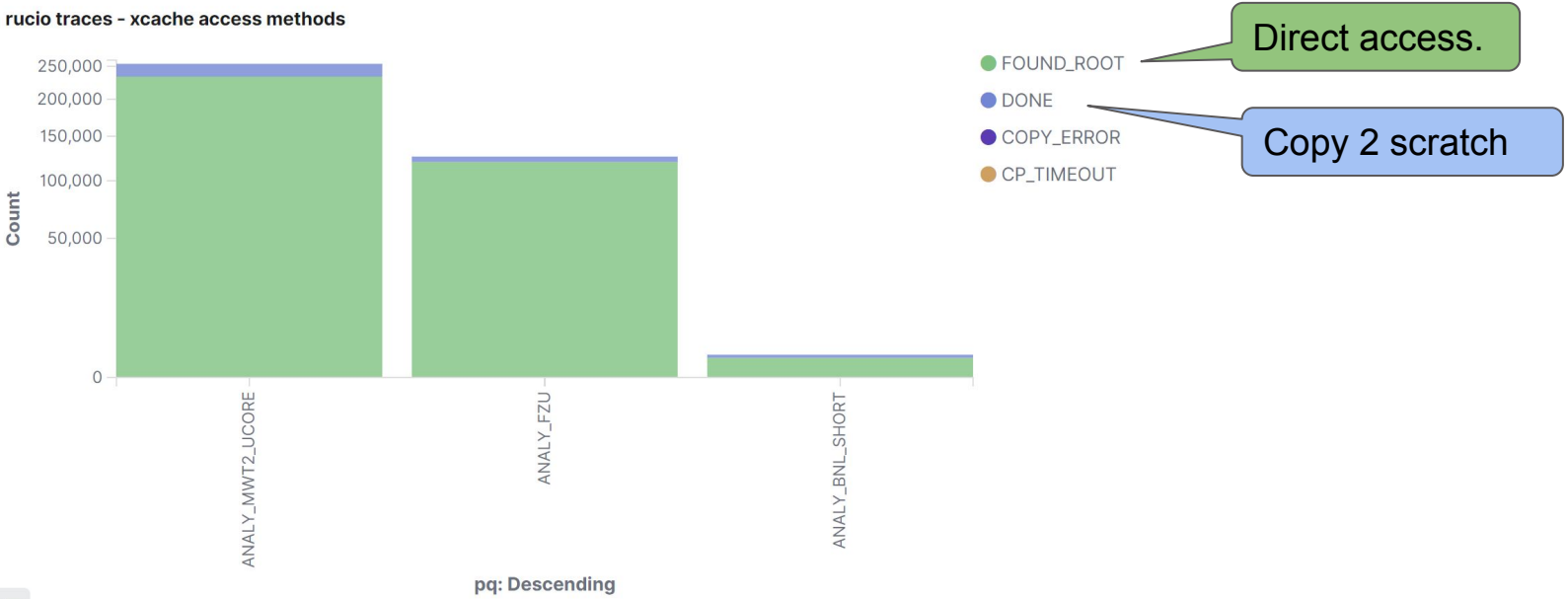
- ~4Hz requests
- A lot of repeated requests in avg. 7.3 times per DS



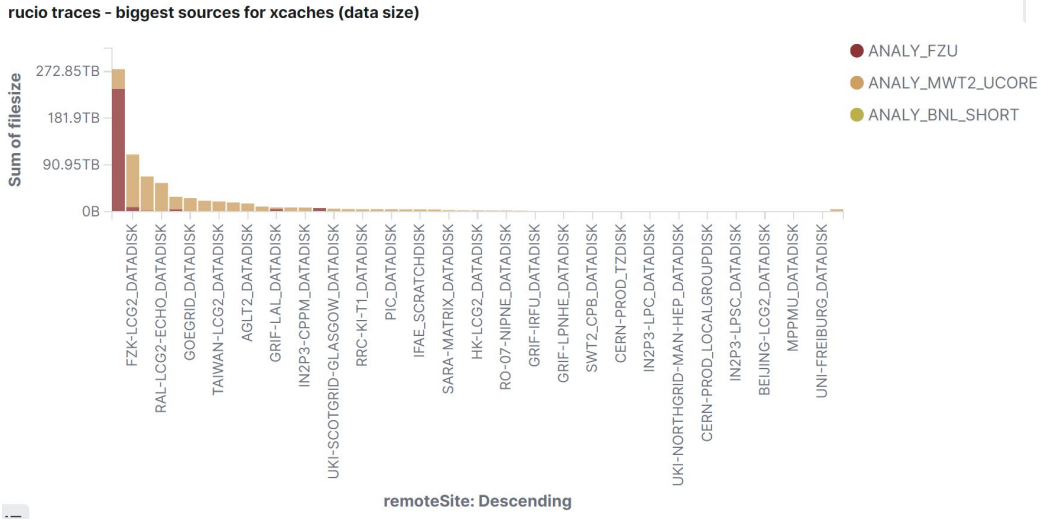


Does scheduling for VP works?

We collect rucio traces. So we can look for reports that had xcache-like paths (url:root*root\:*)



Does scheduling for VP works? Cont.



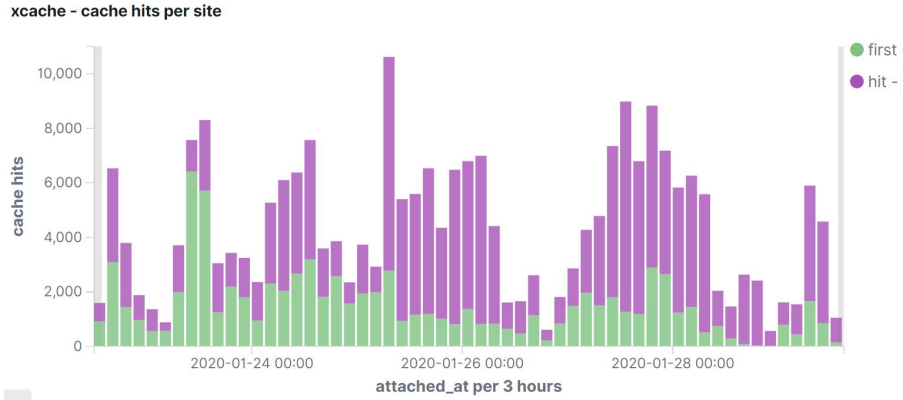
Sources are mainly large sites

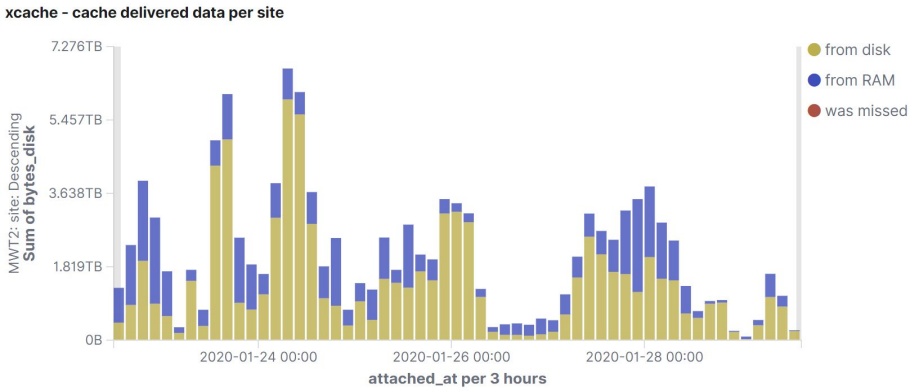
Need serious development to assure best copy (closest) is returned.

pq: Descending	remoteSite: Descending	Count	rate [MB/s]	data
ANALY_MWT2_UCORE	MWT2_UC_SCRATCHDISK	20,196	5.141	746.88GB
ANALY_FZU	PRAGUELCG2_SCRATCHDISK	6,150	5.392	165.04GB
ANALY_BNL_SHORT	BNL-OSG2_SCRATCHDISK	340	16.746	75.89GB

Copy 2 scratch

XCache reports



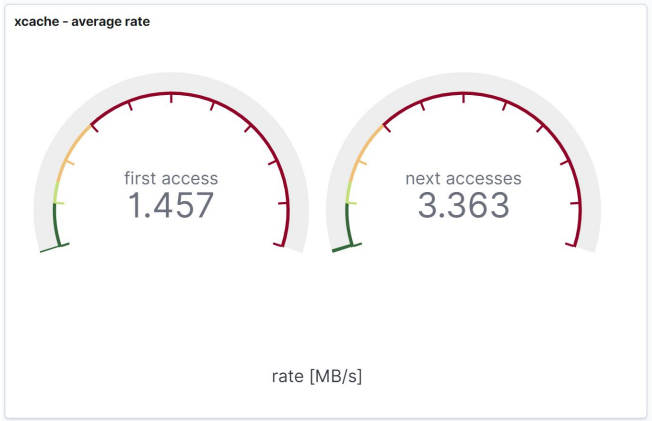


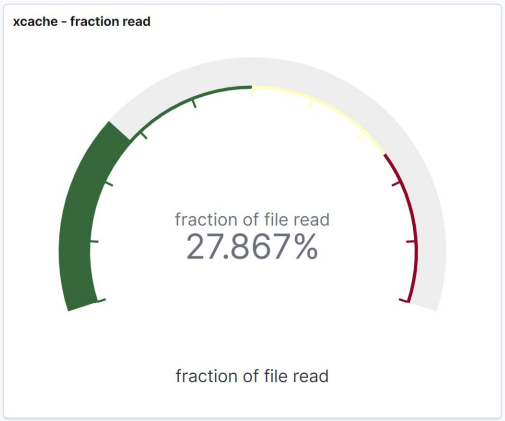
Starting from completely cold cache, one week of data, only MWT2 numbers

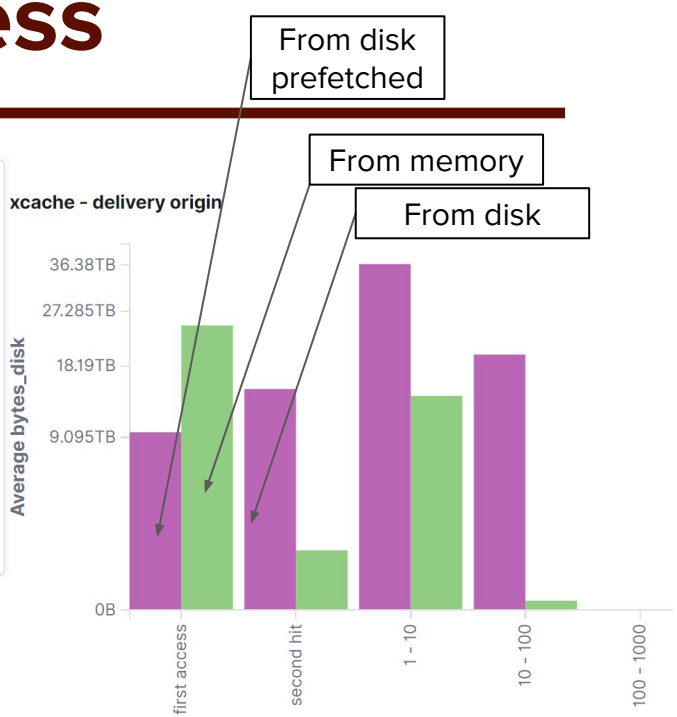
65.5 % cache hit probability.

67% data delivered from xcache disk.

MWT2 - rate and sparseness







In average ~170k files in cache.
Fill factor of the files in cache is ~72%.
Part of the jobs do copy2scratch so these have ~100% fill.

Subsequent accesses add more data to cache.

Non VP caches

first access: filters

site: Descending	files accessed	Delivered from disk	Delivered from RAM
LRZ-LMU	6,104	366.929GB	225.262GB
UKI-SOUTHGRID-BHAM-HEP	997	518.931GB	381.518GB
UKI-SOUTHGRID-CAM-HEP	0	0B	0B
RU-LAKE	0	0B	0B

next accesses: filters

site: Descending	files accessed	Delivered from disk	Delivered from RAM
LRZ-LMU	885	578.996GB	59MB
UKI-SOUTHGRID-BHAM-HEP	2,628	1.124TB	0B
UKI-SOUTHGRID-CAM-HEP	1,204	268.904GB	0B
RU-LAKE	46	92.829GB	0B

Not much usage. Cambridge and RU-LAKE only test files.

Future

- XCache developments
  - Update to version that supports CRC once it’s ready
- Origin fixes
  - NET2 and SWT2, all CA sites don’t have xrootd endpoint or it is not first choice for read_wan.
- VP
  - Find and understand all the things that change load on the caches.
  - Full throttle testing - a large site served only via xcache
  - Deploy in front of an HPC
- Far future
  - Multi node xcache support
  - Moving VPservice into Rucio
  - **Adaptive caching instead of LRU currently used**

Adaptive caching

- By “pinning” datasets to caches (VP) we solve most of the low cache hit rate problem. That can get us to ~80% cache hit rate with the Least Recently Used cleanup model (LRU).
- If we could gain 5% by changing caching model, that would reduce WAN traffic by 25%, which is a lot.
- All kinds of schemes proposed, some even tried.
- Most popular idea is “we know what we are using most”.
  - We don’t really know, up to now all assumptions proved wrong.
  - What is popular changes while hard coded rules tend to stick.
  - Would require continual effort on analysis and re-tunning. Impossible to do for all the sites/panda queues.
- Naive approaches trying to detect “popular” datasets failed.

Why do we need it now?

- For now, we really don’t… first we need to:
  - get xcaches integrated in regular operations
  - gain operational experience
  - characterize performance and effects on job eff, wan throughput etc.
- But we need to start making it now:
  - RL is not something you do in a week
  - Training takes time
  - Integrating it into xcache would take time
- Can be useful for other DDM operations eg. select files to move to tape.

Reinforcement learning

An **actor** gets trained **once** or **online**, by an **environment** that gives a **reward** for “good” **actions**.

Used for everything from Chess, Go, to Hide & seek.

Specially useful for situations where not all info available and multiple actors influence the system simultaneously, thus requiring cooperation (eg. multi-level cache actors, Rod and Ivan).

Plan

- Get data - already in ES, extract it. ✔
- Preprocess data (tokenization of filenames, dataset names too). ✔
- Create environment - basing it on OpenAI gym environment. Two environments:
  - discrete action (cache/not cache) ✔
  - Continuous action (predicting probability that a file is already in cache/ should be cached)
- Train different actors
  - Deep-Q network (DQN) or Dueling DQN for discrete action env. ✔
  - Actor-Critic model for continuous action env
- Compare them with LRU