Caching Servers for ATLAS*

CHEP 2016 San Francisco
October 10-14, 2016

(*) Work in collaboration with Matevz and Alja Tadel (UC San Diego, CMS)

Rob Gardner
University of Chicago

Ilija Vukotic
University of Chicago

Andy Hanushevsky
SLAC

Wei Yang
SLAC

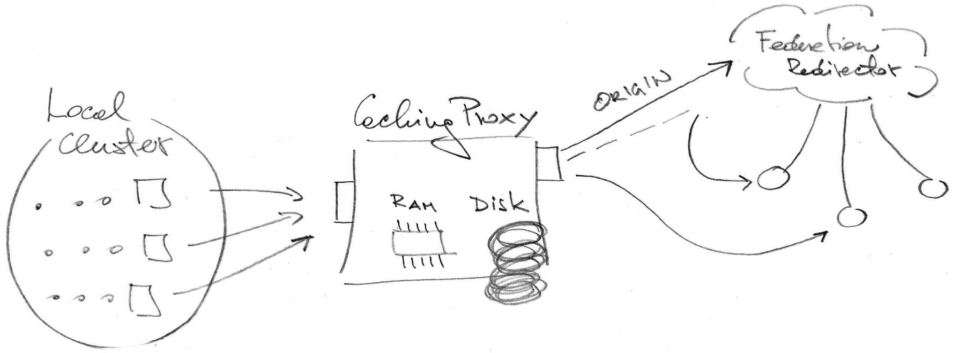
ATLAS EXPERIMENT

Why Caching?

- Many small sites have limited storage capacity and spend a great deal of time managing datasets downloaded from the grid
- Meanwhile the capacity of the Tier1, Tier2 sites has increased significantly and already host the most popular datasets and are directly accessible via FAX
- Most analysis datasets are read multiple times
- Can also use proxy caches in front of opportunistic computing sites which do not have managed storage

Xrootd Caching Proxy Concept

- Provides disk-based cache for WAN access:
  - performs read-ahead to reduce latency
  - avoids WAN access on subsequent access.
- Use fixed-size blocks between *32kB* and *4MB*



Operating Modes

- Opening files
  - Cold cache: remote open through FAX
  - Warm cache: open the file from local disk
    - Remote read only happens if not block not local
- Read, Vector Read
  - If in local RAM/disk => serve from RAM/disk
  - Otherwise request from remote and:
    - Serve to client and write to disk cache via write queue (remains in RAM until written to disk)
    - All read requests padded to align with cache blocks

Authentication

- Local
  - Any method supported by XRootd
- Remote
  - Proxy uses GSI to authenticate itself to FAX

Some details about the service

- ***cinfo file:*** stores details about downloaded blocks and local accesses.
- ***Prefetching:*** the proxy can issue advance read requests to reduce RTT latency (optional).
- ***Decision plugin:*** allows users to configure which parts of namespace are to be cached.
- ***Cache purging:*** high/low watermark algorithm used to start/stop purging.

Stress Testing at SLAC Tier3 Cluster

- Caching server (Dell R730xd)
  - Single Intel Xeon CPU E5-2643 v4 @ 3.40GHz (6 physical cores)
  - 128GB memory
  - Intel X540 10Gbps NIC
  - H730 raid controller
  - 12x 2TB 7200rpm HDD in raid-0
- Configured block size of 1MB
- Caching server memory limit set to 30 GB
- 750 concurrent jobs; 35 Gbps to WAN

Stress testing, cont.

We prepared a list of ~5000 files, all reside at BNL-OSG2_LOCALGROUPDISK.
Each job had the following characteristics:

1. Randomly select a file from the list
2. Randomly determinate the number of reads from the file (0 - 1024), offset (starting position) of each read, and the length of each read (0 - 128KB).
3. Data read from network are discarded and the next read follows immediately.
4. Repeat 1-3 for each file
5. All jobs start at about the same time, with run limit of 2.5 hours.

In the stress test, the offset of each read is completely random. In a realistic use case, though they generally move from the start to end of the file.

Stress testing, cont.

We conduct three sets of batch jobs:

1. Read directly from BNL over the WAN.
2. Read from the SLAC cache when it is completely cold (this warms up the cache)
3. Read from a "warm" cache.

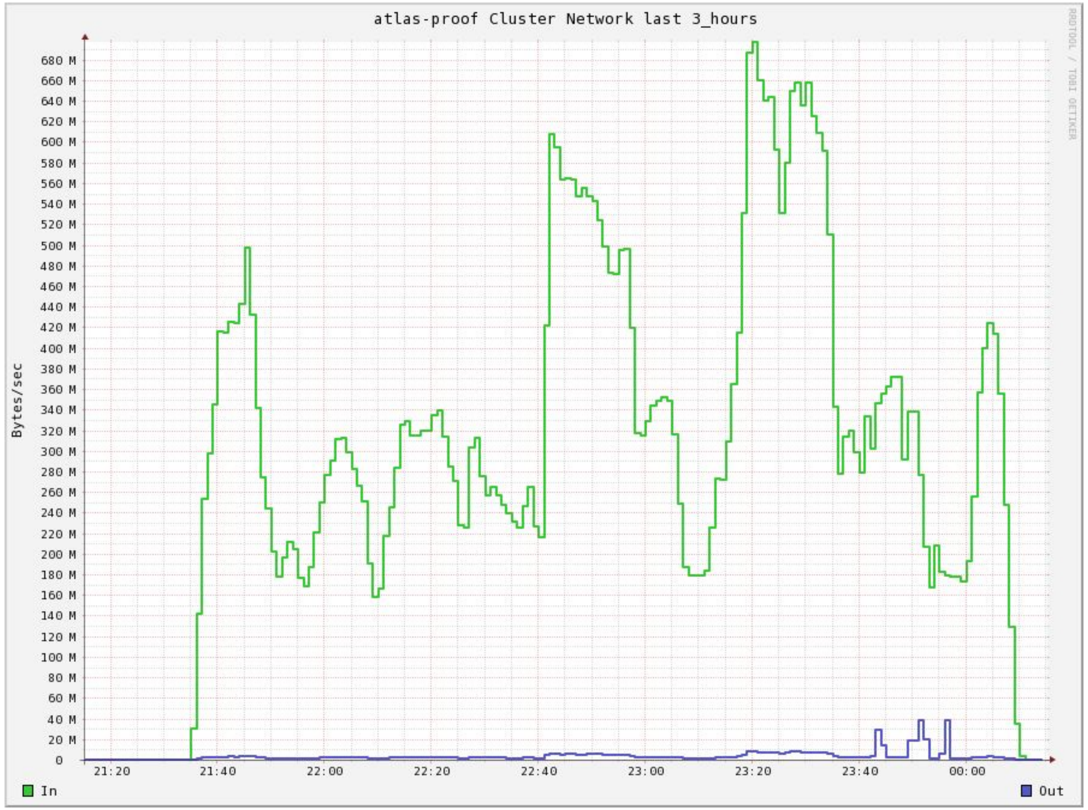
Note:

- We are not necessarily looking to see if we can outperform reading from directly from remote source (BNL storage has thousand of disks)
- We look the functionality of the cache, and whether we can reach the full potential of the cache's hardware (thus what we need to outperform reading directly from the remote data source).

Stress test 1: direct WAN

The following is a networking traffic plot of the dedicated cluster. The green line shows the data coming to the cluster **directly over the WAN**.

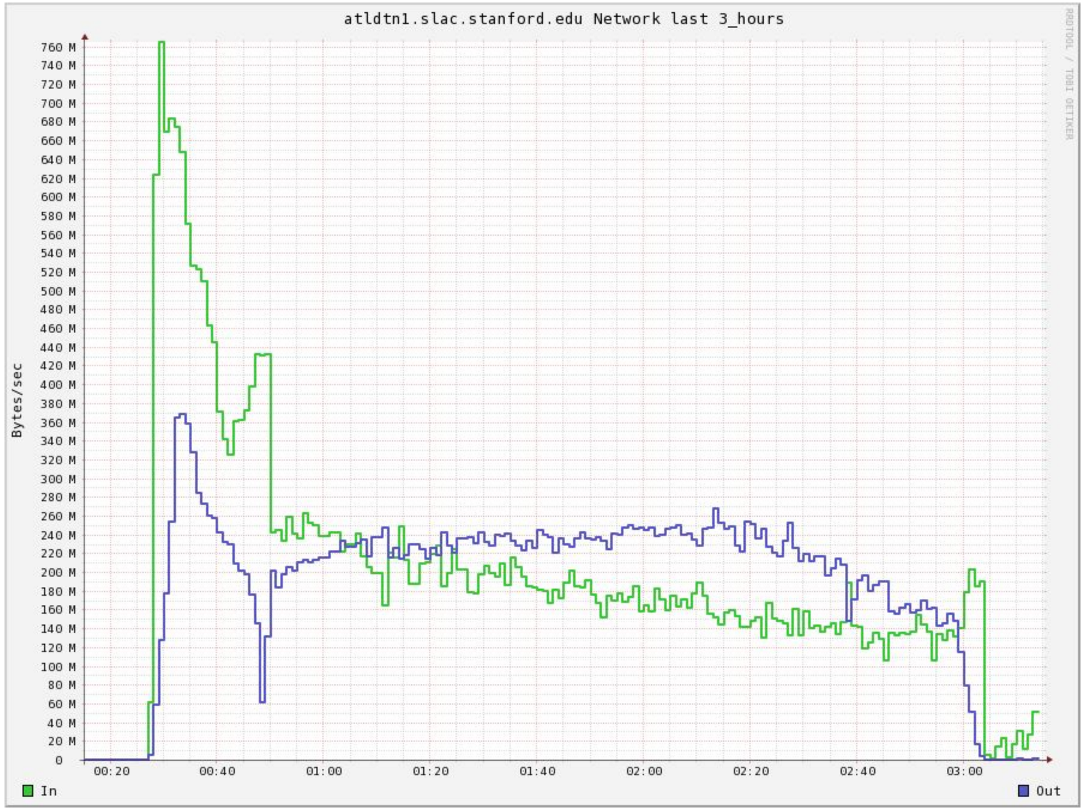
The output also shows that on average, each job was able to "complete" about 100 files during the 2.5 hour run time.



Stress test 2: cold cache

The following is a networking traffic plot of the **proxy cache**.
The green line shows the data coming from the remote data source.

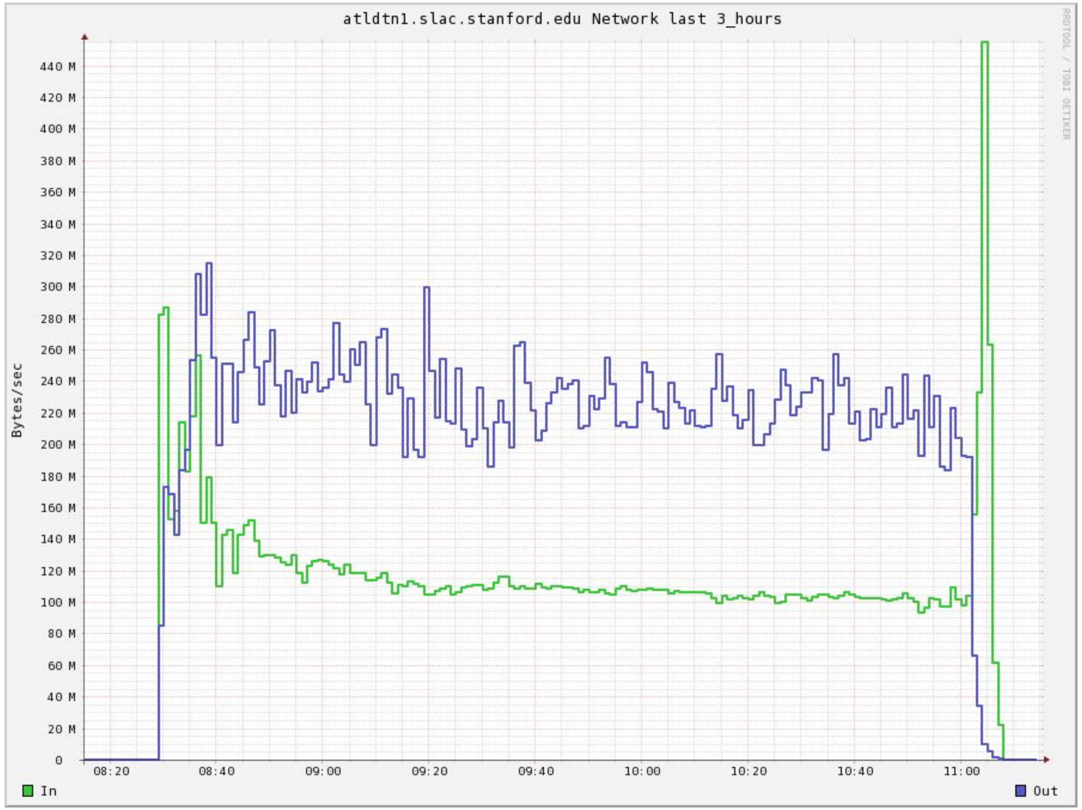
The blue line shows the data sent to the batch jobs. On average, each job completes ~80 files in the 2.5 hour run time (delivering ~ 250 MB/s)



Stress test 3: warm cache

As the cache warms, some data are read from the cache's disk storage rather than reading from the remote data source.

While performance is limited to disk I/O, the result is a reduction of needed WAN bandwidth from the site



Testing with Realistic Analysis Jobs

To get a flavor of cache performance with actual ATLAS analysis jobs, we built a caching server and placed it in front of ANALY_MWT2 and directed "overflow" brokered jobs through it (i.e. jobs originally scheduled to another site, rebrokered to MWT2 due to long queue times, with input data left in place at the original site).

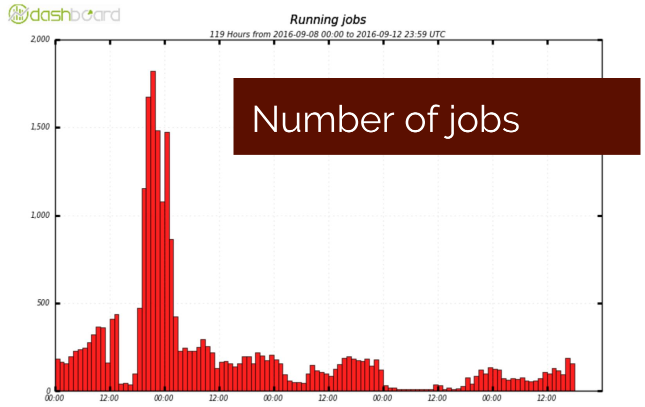
Tier2 Analysis Cache Setup

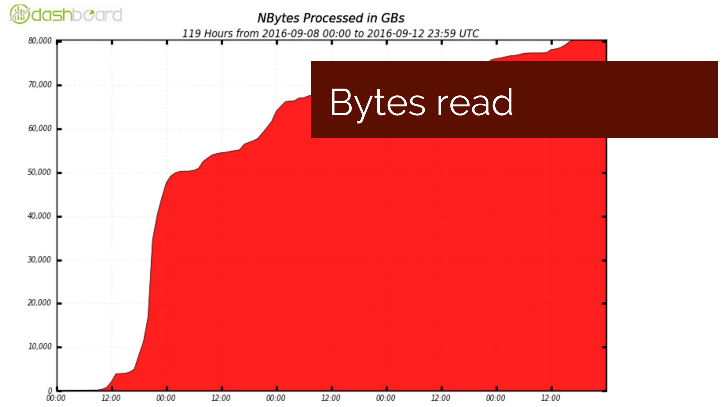
The test caching server is a Dell R730xd E5-2650 v3 @ 2.30GHz, 40 core, 96GB RAM, 10 Gbps NIC, connected to the WAN via a Juniper EX9206 router at 80 Gbps. The storage setup is as follows:

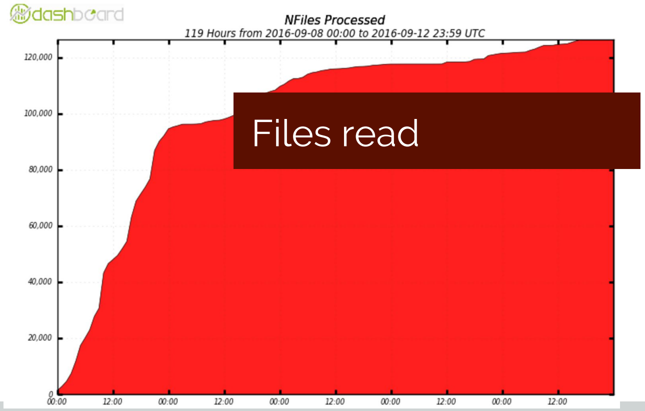
- (10) 8TB disks in RAID-0
- Deadline scheduler selected for the kernel
- nr_requests block tunable set to 1024 (I/O request queue size, default 128)
- 5120 KB read-ahead
- Disk cache policy enabled, using 2GB NVRAM on RAID controller
- Controller set to read ahead & write back modes.

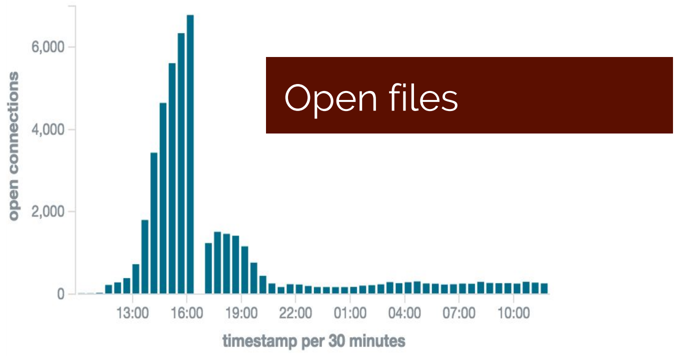
The caching software is configured to run in file block caching mode with block size of 256kB. We allow up to 48GB of memory to be used by the cache software as buffer. When a requested data block is not found in the cache, it reads from FAX.

Tier2 Analysis Cache Testing





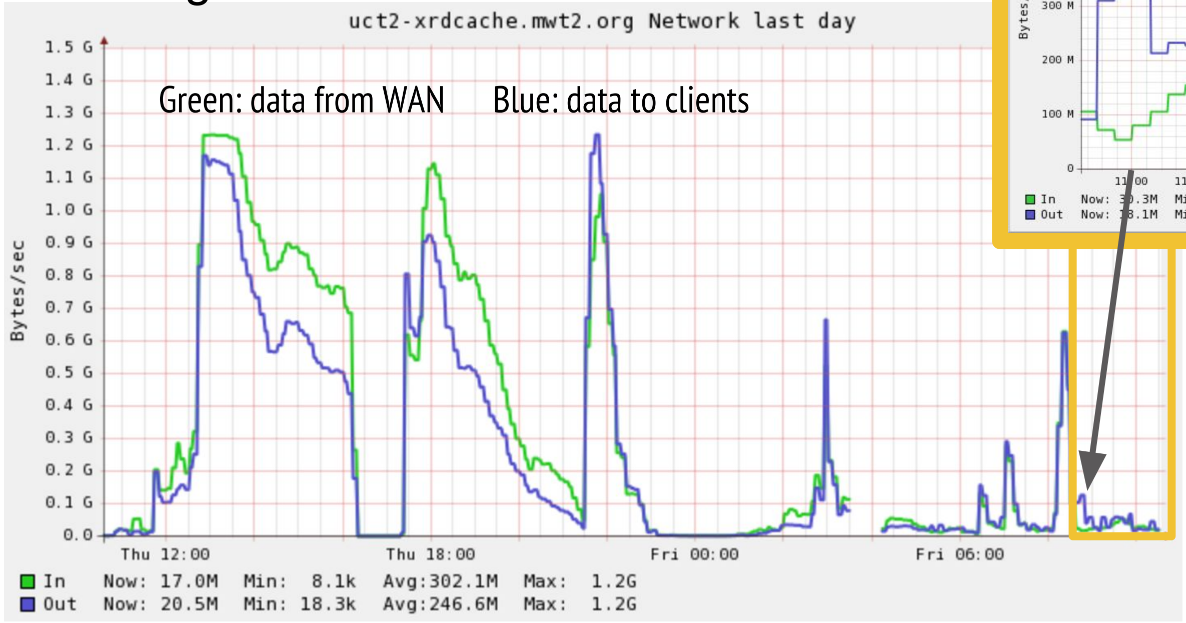


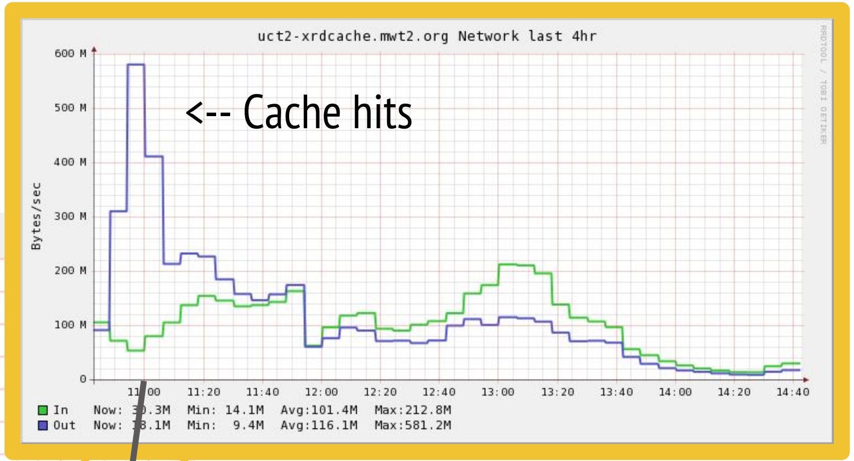


WAN data accesses from re-brokered analysis jobs cached by the XRootd proxy cache

Tier2 Analysis Cache Testing Results

Initial cold cache filling while delivering data to worker nodes:





Later: data re-use results in cache hits and less traffic on the WAN

Conclusions

- Disk caches offer a flexible and efficient data delivery mechanism for ATLAS distributed processing
- Development of XRootd caching software in the past year is becoming sufficiently robust for use in production analysis environments
- Will be useful for Tier3, Tier2-WAN, opportunistic, and cloud processing
- These will be highly performant caches and can scale to available bandwidth by clustering

Thank you!