FERMILAB NEWS

FERMILAB A U.S. DEPARTMENT OF ENERGY LABORATORY

Dmitry Litvintsev (Fermilab)

19th International dCache user workshop

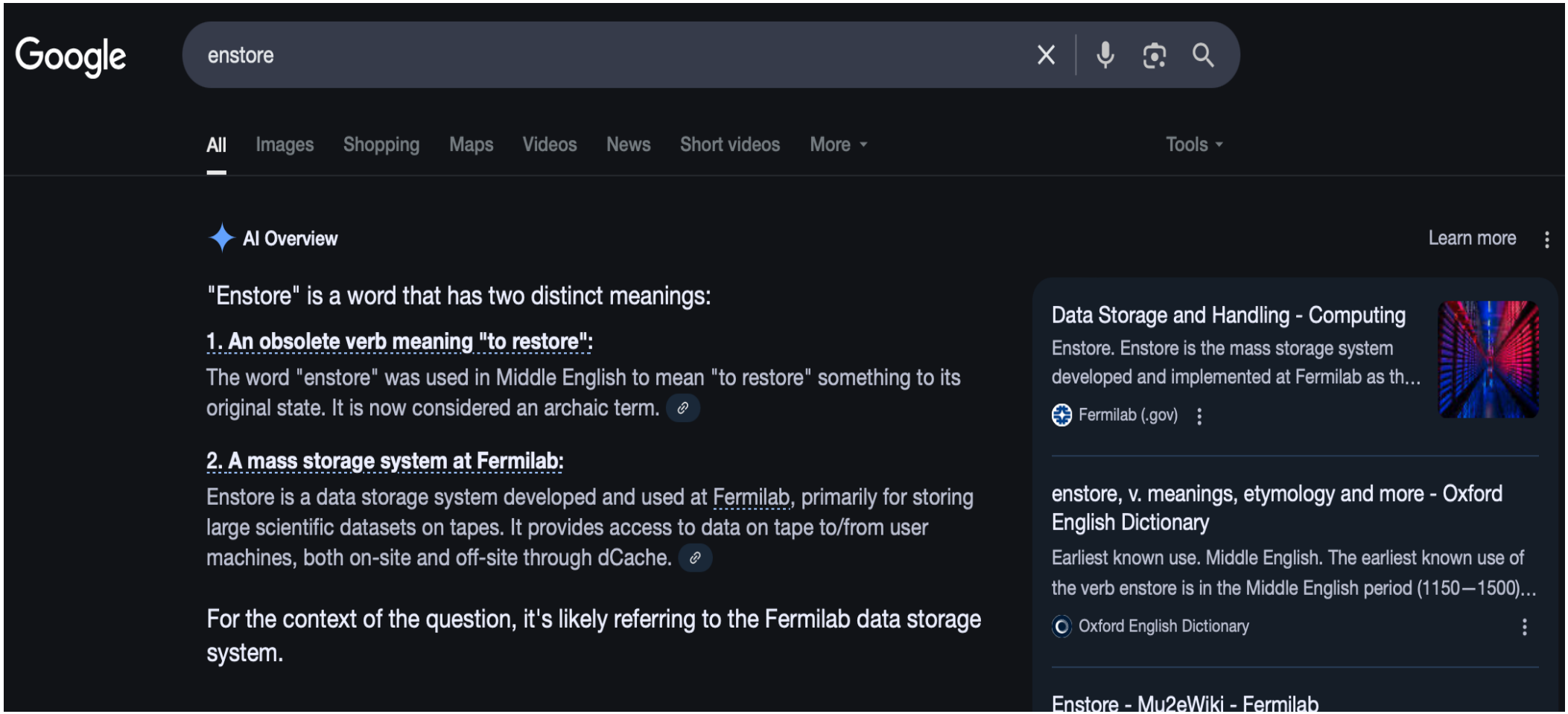
May 20 – 21, 2025

IN2P3 computing centre, Lyon, France

Outline

- Parting with Enstore
- Transition to CTA
- First Experience with CTA
- Experience with Bulk
- Hot File Replication
- Thoughts on containers

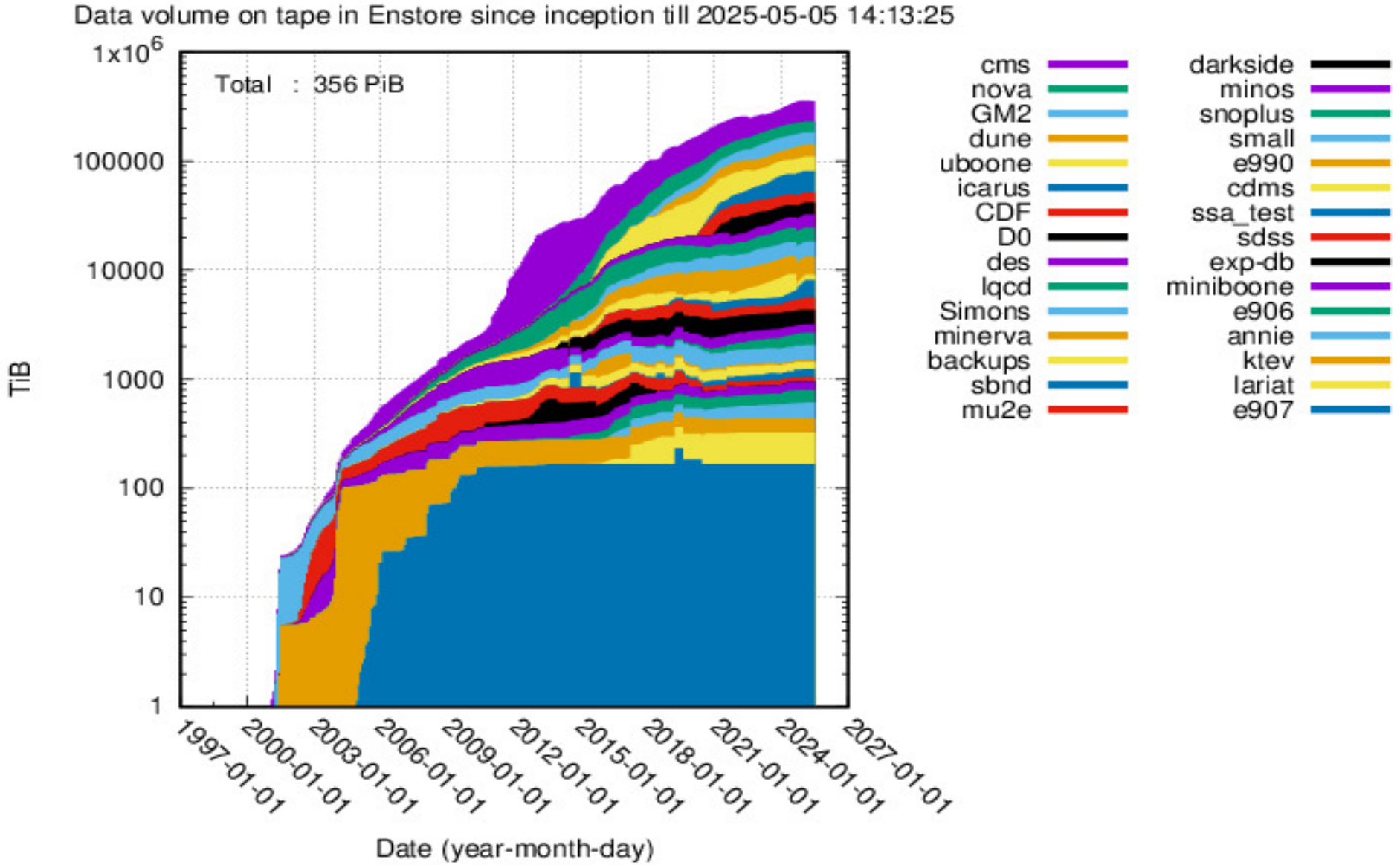
Enstore


Google
enstore
All
Images
Shopping
Maps
Videos
News
Short videos
More
Tools
AI Overview
Learn more
"Enstore" is a word that has two distinct meanings:
1. An obsolete verb meaning "to restore":
The word "enstore" was used in Middle English to mean "to restore" something to its original state. It is now considered an archaic term.
2. A mass storage system at Fermilab:
Enstore is a data storage system developed and used at Fermilab, primarily for storing large scientific datasets on tapes. It provides access to data on tape to/from user machines, both on-site and off-site through dCache.
For the context of the question, it's likely referring to the Fermilab data storage system.
Data Storage and Handling - Computing
Enstore. Enstore is the mass storage system developed and implemented at Fermilab as th...
Fermilab (.gov)
enstore, v. meanings, etymology and more - Oxford English Dictionary
Earliest known use. Middle English. The earliest known use of the verb enstore is in the Middle English period (1150—1500)...
Oxford English Dictionary
Enstore - Mu2eWiki - Fermilab

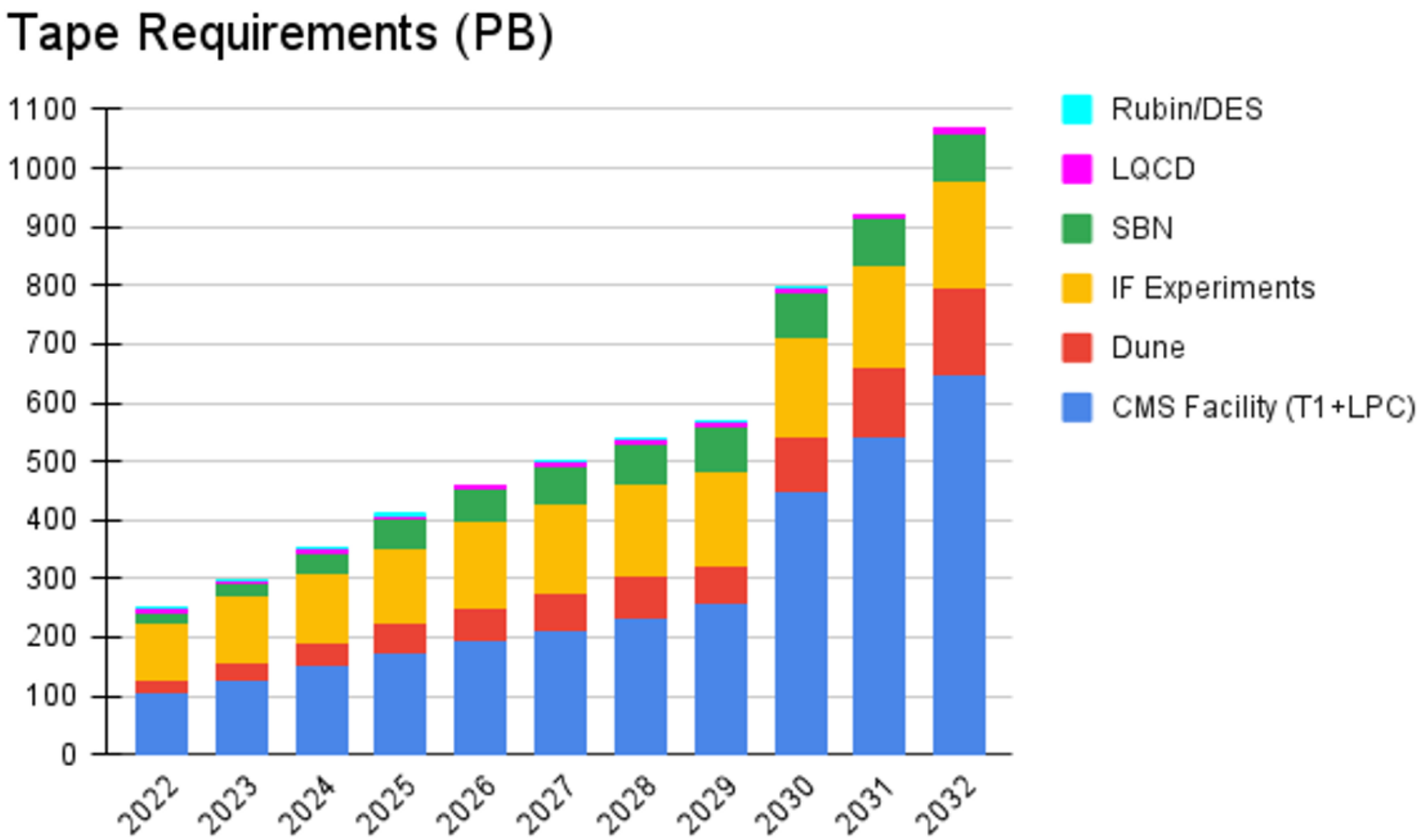
Ode to Enstore

- Prototyped in December 1997 when Don Petravick and Jon Bakken visited DESY to learn about how DESY used MSS. Could not take OSM due to licensing – wrote a python “version” of it.
  - Network attached tape storage with single rooted namespace exposed to users for directory functions over NFS (PNFS from DESY) with a cp-like client for tape I/O
  - Single threaded
  - UDP for control
    - Messages fit in 1 datagram
    - Retries, Unique Ids
    - Clients can't hang servers
  - TCP for data transfers
- Accepted as HSM for Run 2 experiments starting from D0
- Has been used in production for 25 years !

Volume on tape in Enstore vs time



Fermilab data on tape projections



Enstore lessons

- Enstore worked perfectly for Run 2 experiments (under 1PB/year intake each).
  - Around 2011 Support for Small File Aggregation (SFA) added
- Some of the choices made at the outset that allowed the system to be simple, nimble and easy to implement made it somewhat difficult to scale up when rapid expansion of volume, rates and number of files happened when Intensity Frontier Experiments started taking data (around 2011)
  - Multithreading was introduced soon after deployment for Run 2, but multithreading in python is fake and provided only partial relief. Transition to multiprocessing was only partial
  - Choice of UDP protocol for control communication ended up with complicated “TCP over UDP” message delivery check layers
  - Message retries caused by slow response of Enstore components under load would further exacerbate the load leading to even slower response. Enstore database servers suffered from this issue the most, followed by media changer and library managers. Lost messages, messages retried due to timeous result in files written multiple times
  - Each virtual library manager (LM) maintains an in-memory priority queue of store and restore requests. A single library manager (typically corresponding to one physical library) could not handle more than 20K large queue

- Effect of limited LM queue size is profound.
  - The sum of total active stores and restores on dCache pools had to be kept under 20K
    - This means that Enstore always “sees” only fraction of all restores for a given tape resulting in excessive mounts (the most expensive operation in this business)
    - Likewise, Enstore only sees a fraction of all write requests which could be written to a tape in one mount. Instead tape gets dismounted to serve different storage class because requests for “this” storage class are all QUEUED on dCache pools
    - Opening up dCache queues while keeping LM queue to 20K resulted in chaotic LM/encp interaction

Out with the old, in with the new

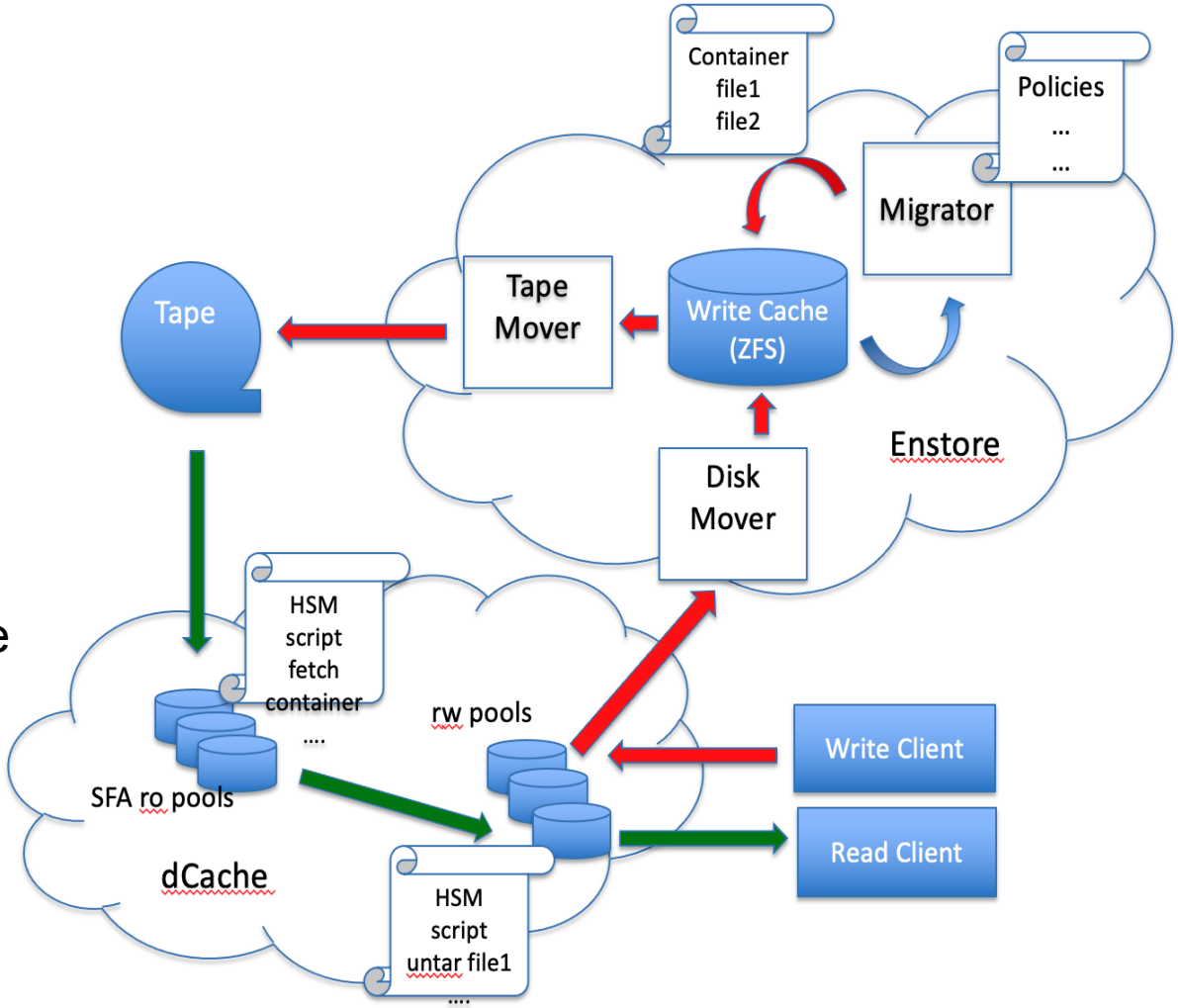
- Faced with:
  - declining budgets
  - need for continuing development effort on Enstore to address scalability in the face of many-fold increase in data volume, request frequency and more stringent security requirements
- Fermilab has decided to adopt CERN Tape Archive (CTA) HSM (following RAL and DESY)

Migration path

- Enstore DB metadata → CTA metadata
  - Plus location info into dCache metadata
- Significant effort went into cleansing/synchronizing/constraining Enstore/dCache metadata
- CTA must be able to read Enstore tapes (“cpio-odc” and “cern” formats). Added two tape types and corresponding read handlers
- Many thanks to PIC who tested reading of Enstore tapes and identified issues that we had missed in testing
- After “10% test” on ITB configured a subset of CMS T1 Tape dCache pools to interact with CTA instance to gain operational experience writing and reading incoming Heavy Ion data
- Mapped out development needed to handle SFA (see below)

Small File Aggregation (SFA)

- Users write small files
- Small files kill tape I/O performance
- Small file problem was solved in Enstore by Enstore SFA (sub)-system that automatically packs (tar) small files into large containers/packages based on storage class, size and time of arrival:
  - Improves write performance (less tape marks to write)
  - Reads/writes run at max drive rate
  - On reads less positioning, backhitching – less tape wear
  - Provides a read-ahead mechanism or “pre-fetching” of the data
  - Improves read rates (wall time based) by factor of 5



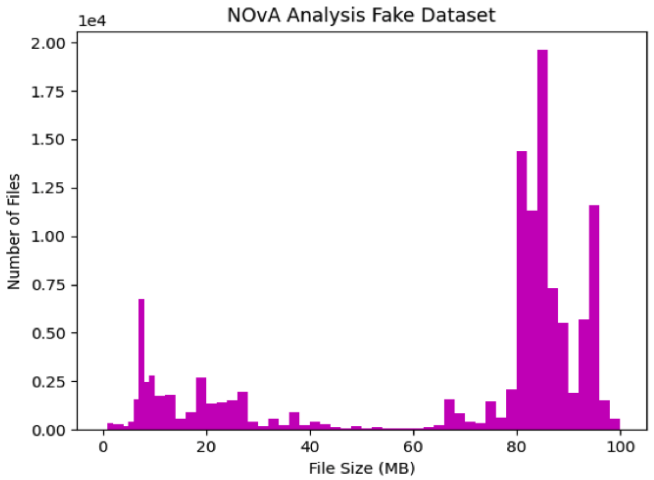
Lost in ~~translation~~ transition

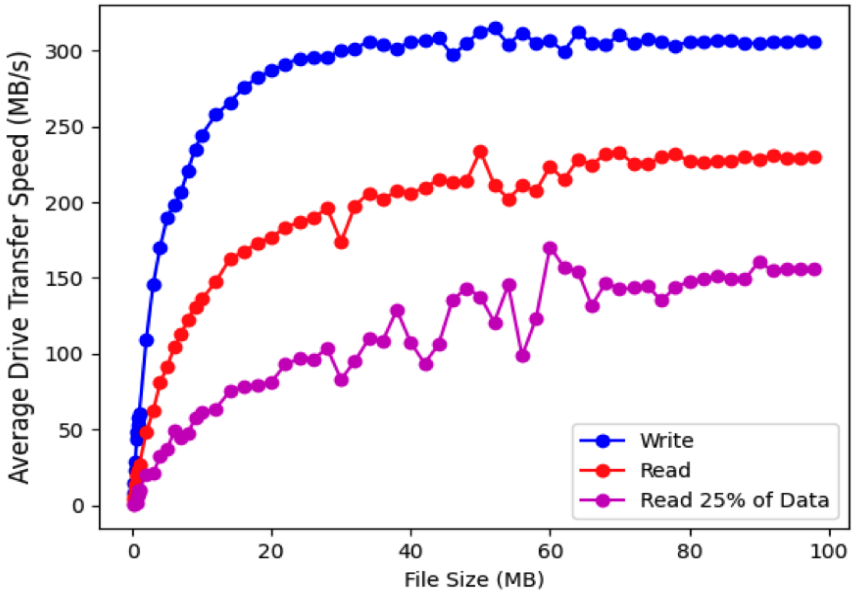
- CTA does not have small file aggregation feature. CERN relies on buffered tape marks when writing small files.
- Evaluated small files CTA performance and provided guidelines for the experiments (basically policing by agreement).
- Implemented existing SFA files reads in dCache

CTA small file write performance

- Generated fake data following re nova dataset file size distribution and wrote it to tape
- Two types of read test:
  - Read tapes end to end
  - Read random 25% of data on tape
- For LTO-8
  - 80% of nominal rate for writin > 20 MB
  - 63% of the nominal rate for reading > 40 MB
- Guidance: keep files > 100 MB

Work done by Tammy Walton





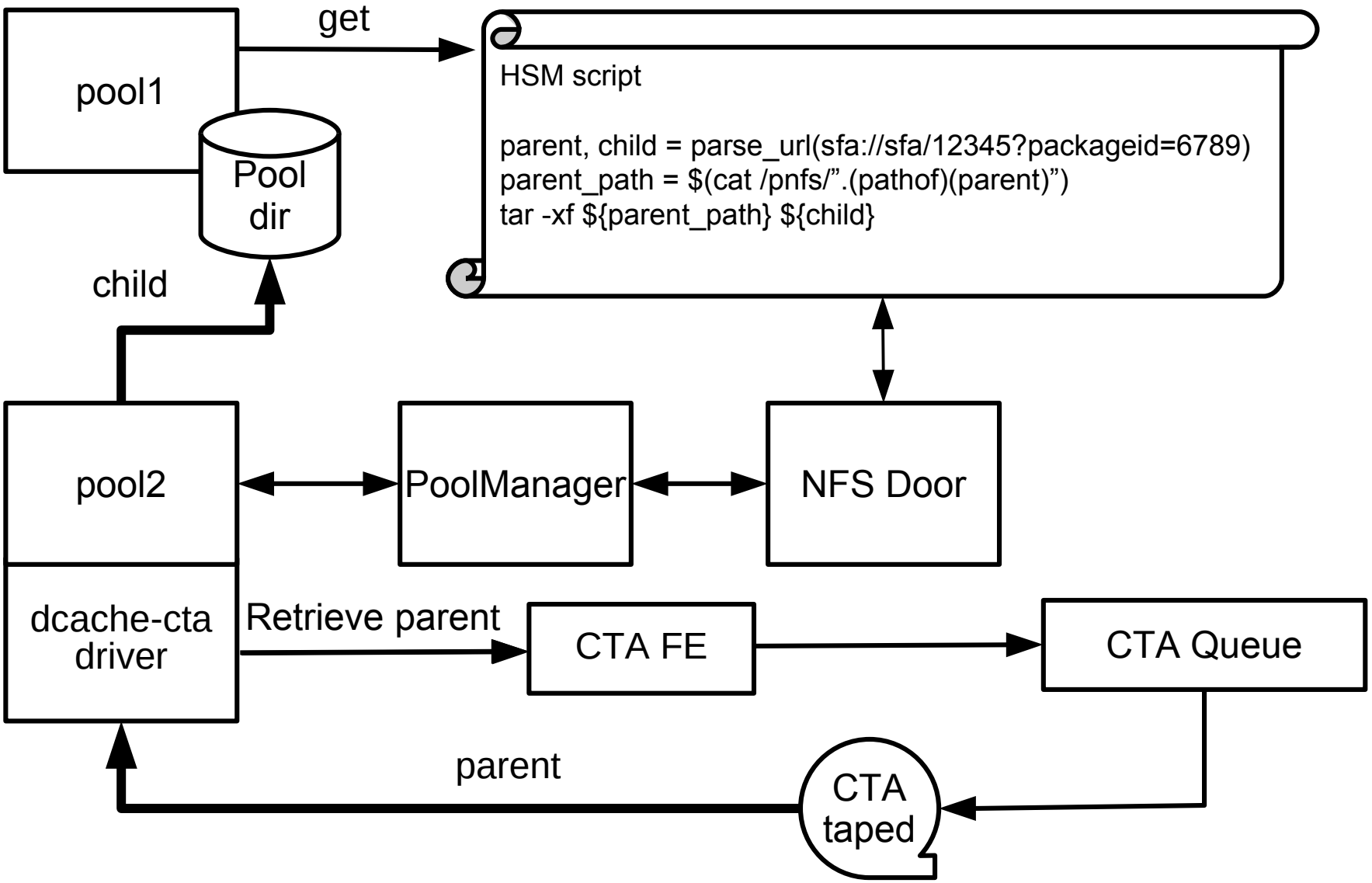
R/O SFA in dCache/CTA

- Translate parent/child relation captured in Enstore metadata on package bfid/child bfid pairs to package pnfsid / child pnfsid and capture it in Chimera `t_locationinfo` table, `ilocation` with value:

  `sfa://sfa/<child pnfsid>?packageid=<parent pnfsid>`

- On pools use HSM script (for now) to untar small file from a package file over NFS mount triggering automatic stage of the package file if its locality NEARLINE

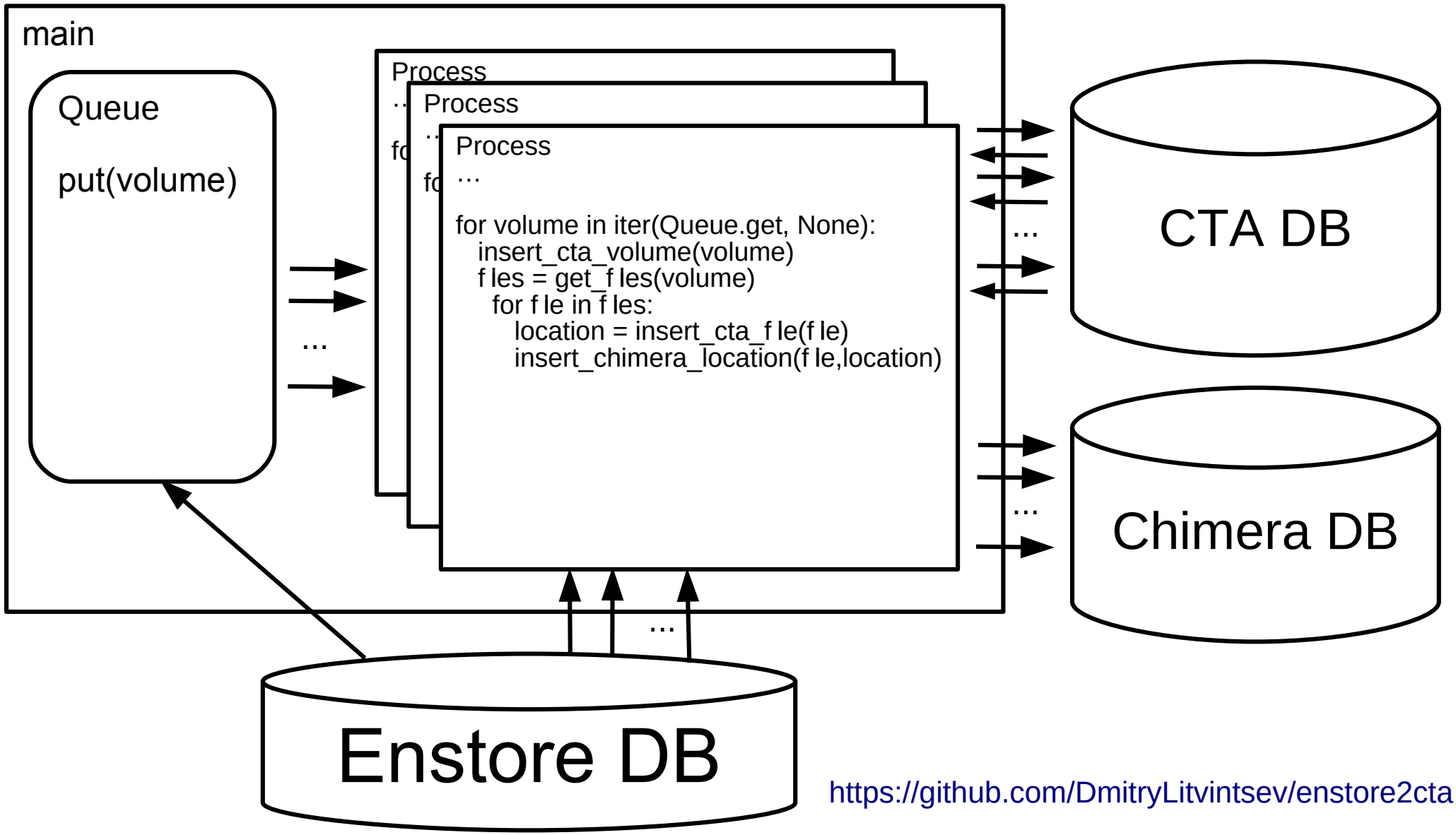
R/O SFA in dCache/CTA



Metadata volumes

- 36K tapes (some tapes have ridiculous file counts in the 300,000s)
- CMS: 40M files
- Public :
  - 183M files on tapes
  - 441M “small” (packaged files. These files will not be known to CTA, but need to have locations added in dCache namespace (aka chimera) DB)

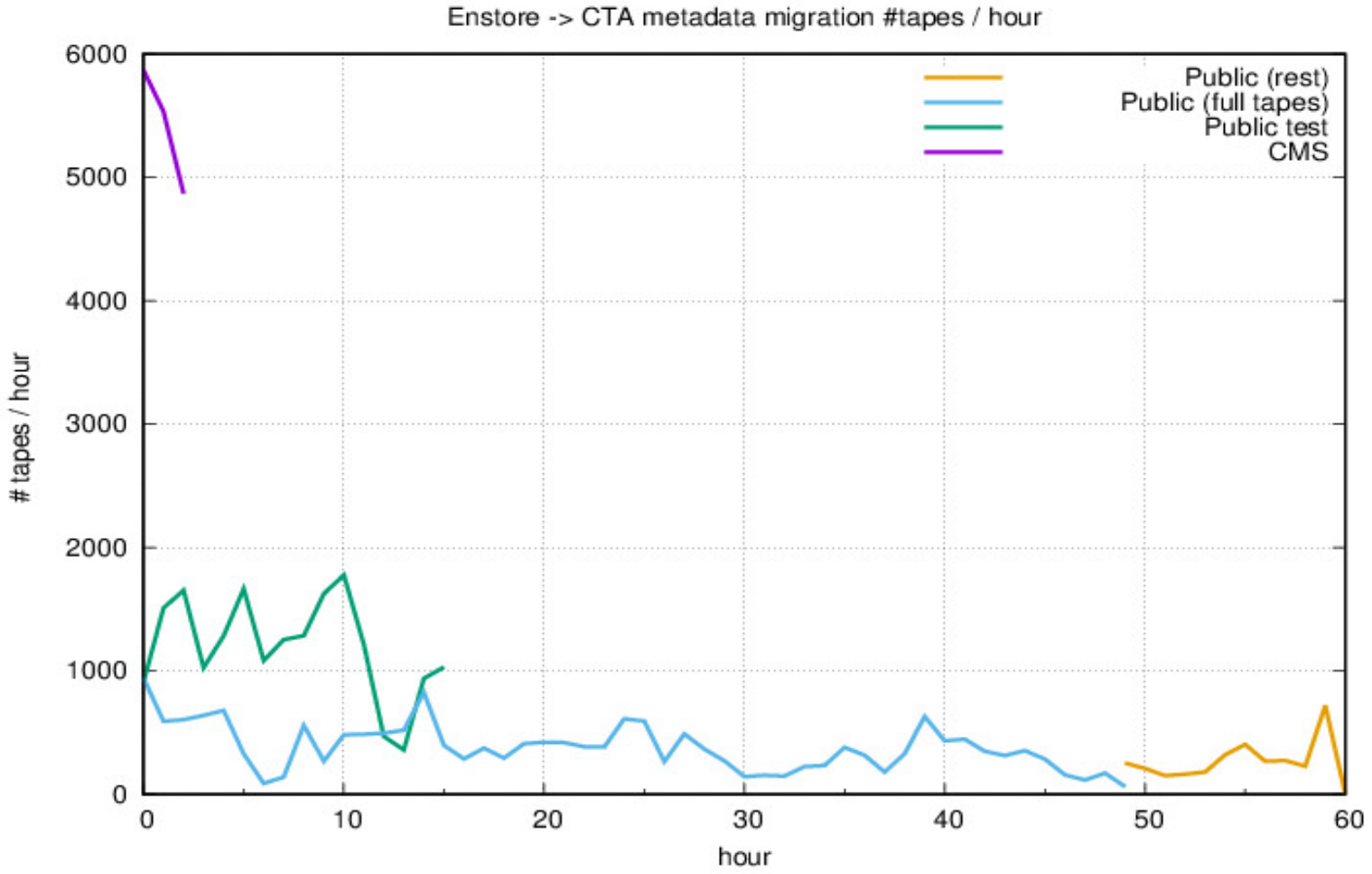
Enstore metadata migration



https://github.com/DmitryLitvintsev/enstore2cta

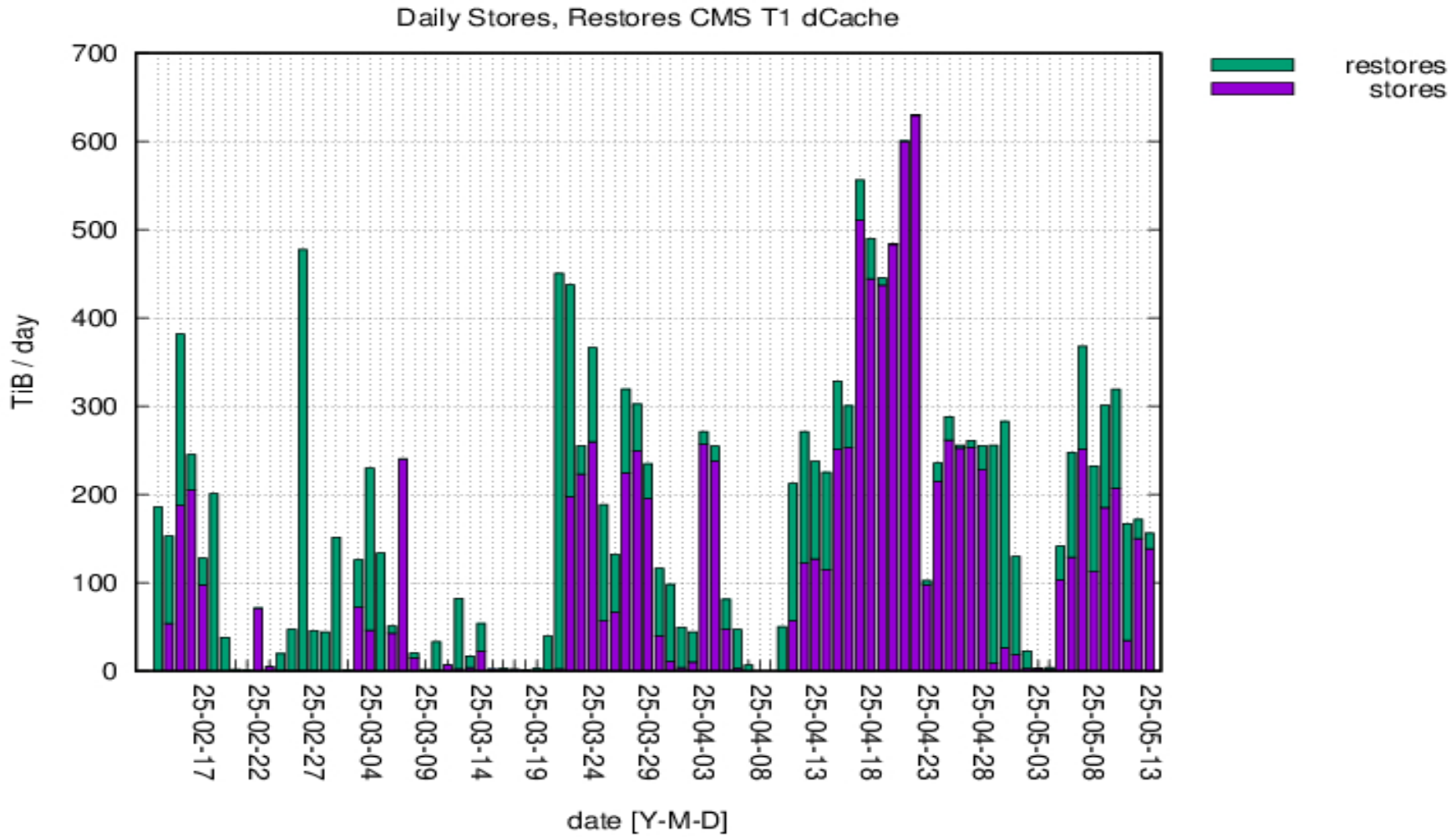
- CMS production was migrated during downtime on 2025-04-07
- Public was migrated in two steps:
  - First, all full tapes were marked (“user_inhibit_1”) and these tapes were migrated in advance on live systems w/o interruptions
  - The rest of the tapes were migrated during downtime on 2025-05-05

Enstore Metadata Migration Timing



Taken by surprise by rather slow performance of production DB when inserting Public Metadata. Glad we did it in two steps and minimized downtime.

Where's ~~Waldo~~ downtime?



Other aspects of transition to CTA

- Setup:
  - 3 front end hosts, 2 in HA setup, one for admin access
  - Tape servers with 2 drives per server connected via FC
  - CMS and public in two different private networks
- All CTA servers are puppetized
- Log events are shipped to ELK stack
- Monitoring development in progress
- Alarms to slack channel
- Procedure for storage class / tape pool definition

Adding storage class / tape pool in CTA

- In Enstore if you did this:

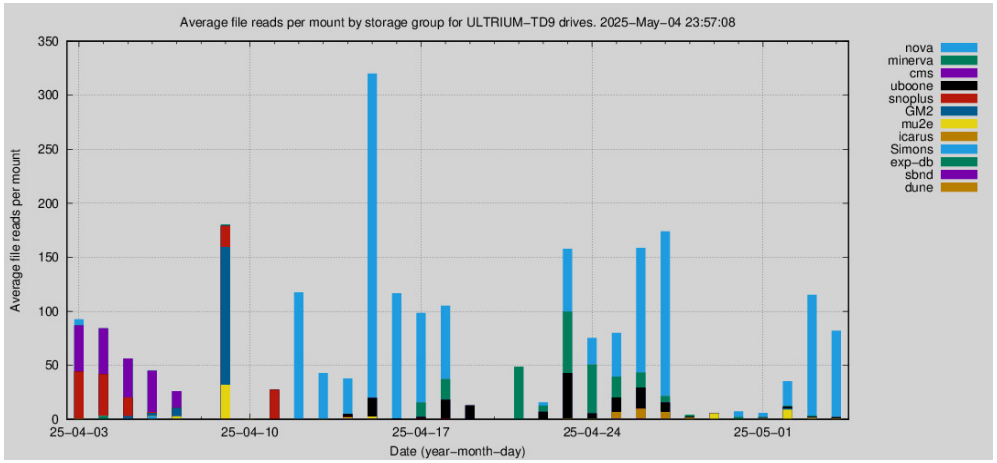
```
cd /pnfs/foo
echo “dune” > “.(tag)(storage_group)”
echo “raw” > “.(tag)(file_family)
echo “10” > “.(tag)(file_family_width)”
```

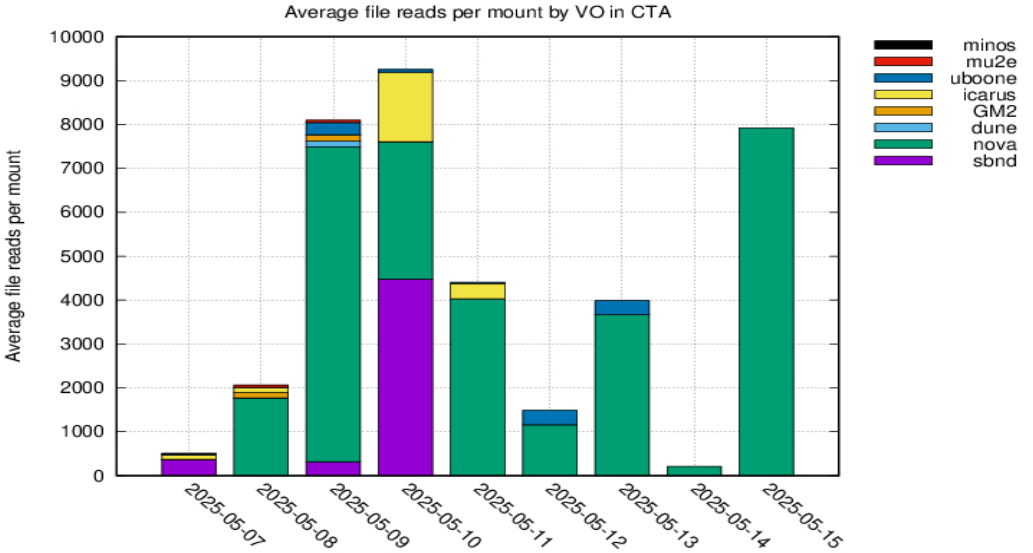
  you have defined all that is needed to store data for VO “dune”, dataset “raw” writing simultaneously to 10 drives. Enstore won't complain as long as there are blank tapes (an no tape quota is used)

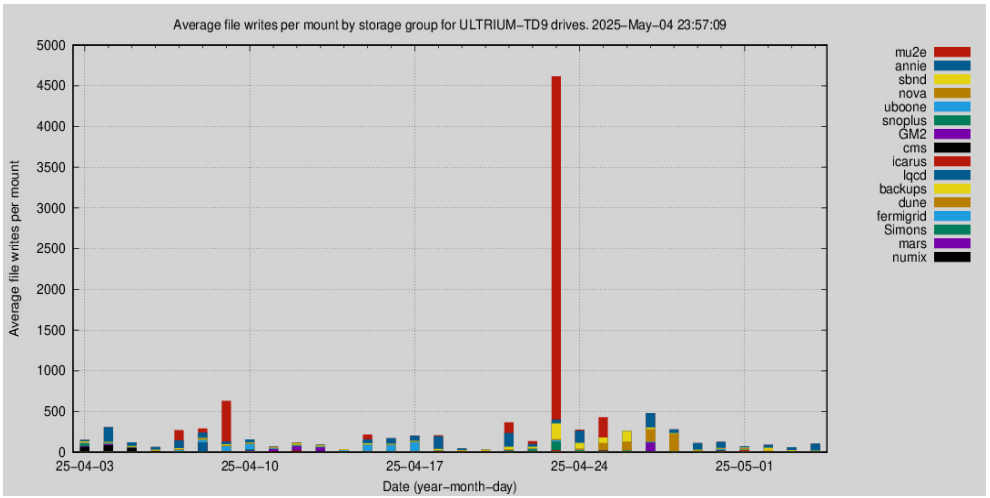
- In CTA things are more constrained and VO “dune”, storage class “dune.raw@cta” and tape pool “dune.raw” and a tape supply have to be defined on CTA end, else no data will go to tape.

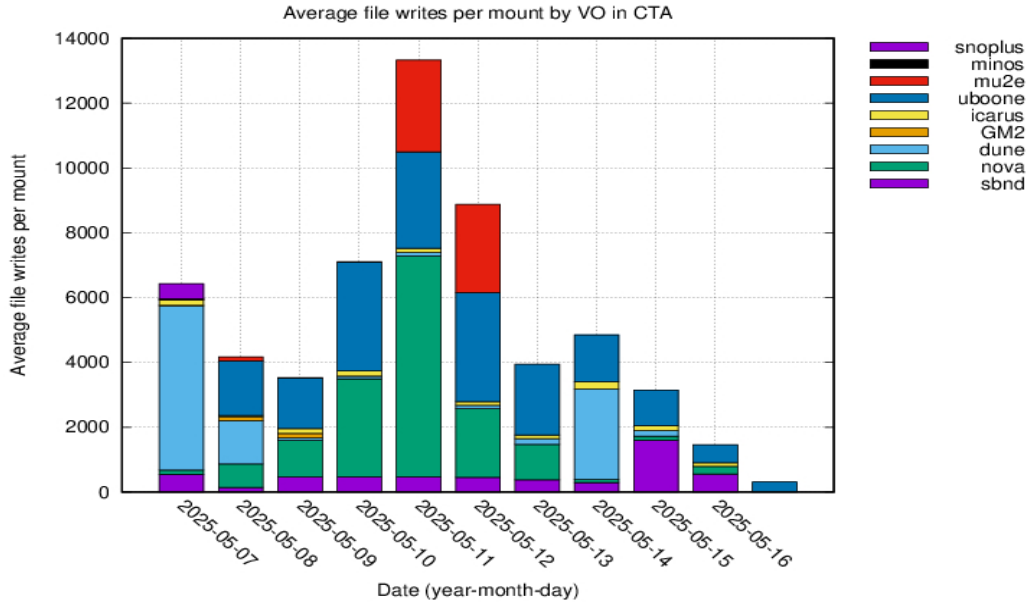
- A GitLab based workflow developed allowing VO data handling representatives to make PRs with new storage class / tape pool definitions. After review and a merge a cron would automatically pull these definitions and create necessary entries in CTA metadata catalogue

Average reads/writes per mount

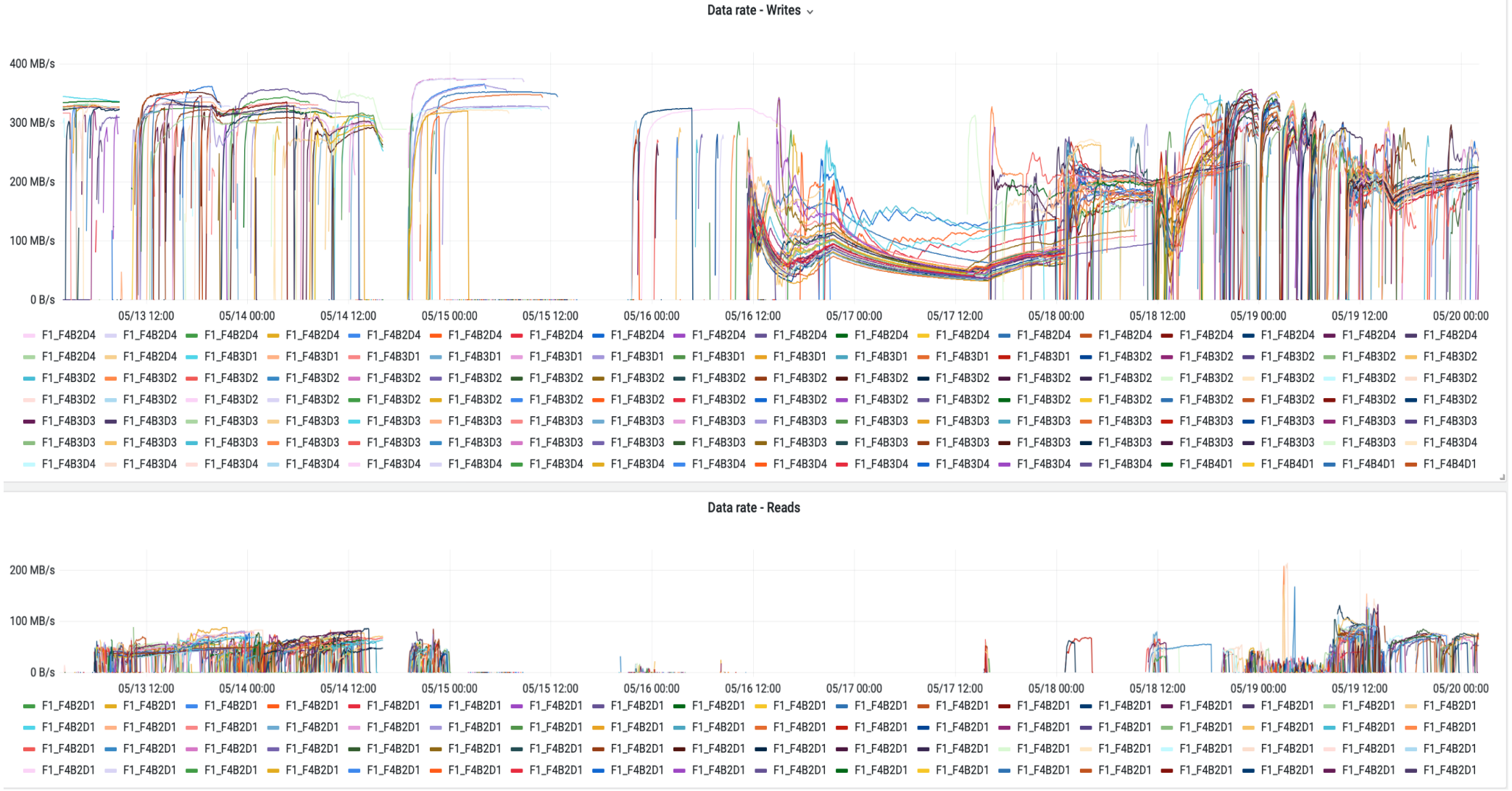








Data rates


Data rate - Writes
400 MB/s
300 MB/s
200 MB/s
100 MB/s
0 B/s
05/13 12:00
05/14 00:00
05/14 12:00
05/15 00:00
05/15 12:00
05/16 00:00
05/16 12:00
05/17 00:00
05/17 12:00
05/18 00:00
05/18 12:00
05/19 00:00
05/19 12:00
05/20 00:00
F1_F4B2D4
F1_F4B3D1
F1_F4B3D2
F1_F4B3D3
F1_F4B3D4
F1_F4B4D1
Data rate - Reads
200 MB/s
100 MB/s
0 B/s
05/13 12:00
05/14 00:00
05/14 12:00
05/15 00:00
05/15 12:00
05/16 00:00
05/16 12:00
05/17 00:00
05/17 12:00
05/18 00:00
05/18 12:00
05/19 00:00
05/19 12:00
05/20 00:00
F1_F4B2D1

- Reads of Enstore tapes proceed at “enstore rates”. Impacted by:
  - No blockid information in metadata (can't utilize tape RAO feature)
  - Sample composition – sparse reads per tape (could me mitigated by fine tuning mount policies per storage class)
- Writes really shine - proceed at nominal drive rate
  - See occasional slowdown. Seem to be attributed to situations when multiple taped server pull data from the same pool host, example:

```
%Cpu(s):  1.3 us,  2.8 sy,  0.0 ni,  0.4 id, 95.0 wa,  0.1
hi,  0.4 si,  0.0 st
```

Breakdown of ESTABLISHED connection to pool by dcache-cta driver ports:

```
17 cmsstor809:1094

10 cmsstor809:1095

15 cmsstor809:1096
```

Perhaps CTA could benef t from *discipline* (Enstore term for allowing only so many transfers per node)

CTA summary

- We bid Enstore a long goodbye
- Transition to CTA was somewhat slow but thorough, particularly in the area of hardware setup and configuration
- We have been running for close to 2 months in production (starting w/ CMS) and already see improvements in write rates and mount efficiency compared to Enstore

In other news

- No more Grid :) (GFTP/SRM/gsi{dcap,xrootd,webdav})
  - Fermilab non-CMS users no longer use x509 {grid,voms} certificates.
  - Completely switched to OIDC tokens
- CMS no longer uses SRM
  - Switched to WLCG Tape REST API
  - So bulk took over bringonline requests

Experience with bulk - CMS

- Stable and smooth running
- Re-discovering some of the stuff that was done for SRM, but was sort of forgotten:
  - Bulk uses single scheduler for all activities (pin, stage, ls, release)
  - If all Bulk threads are waiting for slow activity to complete (stage) then all new requests are queued, even if these are pin request for already ONLINE_AND_NEARLINE files or unpin requests
  - FTS won't transfer files if they are not pinned
  - This creates problems
  - Mitigation is to keep increasing number of threads in bulk thread pool. Eventually you dial that number right (to match FTS request load)
  - A solution would involve putting each activity on separate thread pool just like we did for SRM long time ago

Revisiting Hot Pool Replication

- Current Hot Pool Replication works as follows – if pool cost gets high all new request for data in this pool will trigger p2p transfers
  - Typically this just does a double whammy on that pool
- Chris Green at Fermilab is working on a “Hot File Replication” that would trigger replication of a file if number of requests to this file exceeds a threshold
  - Modified migration module to replicate file to pools belonging to the same pool group without specifying pool group
  - Currently understanding how to manage replication jobs within this new Hot File Replication manager

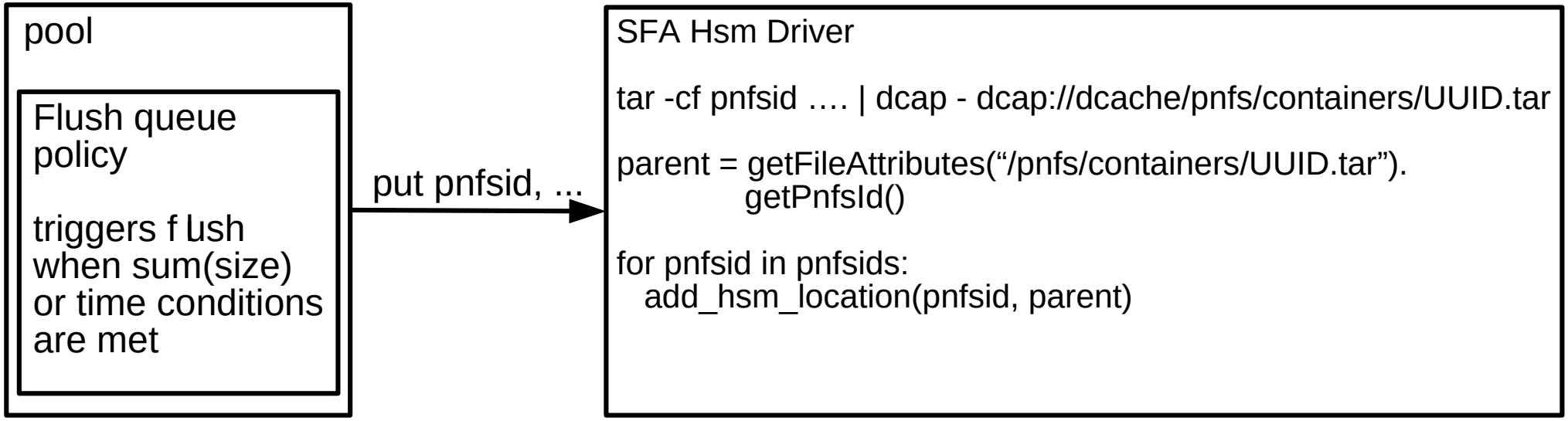
Containers Revolutionized Shipping

We are in similar business – shipping and receiving files

Containers for small files

- Tapes are here to stay
- File based access to tape is not efficient unless files are large
  - As tape capacity increases what is large today is small tomorrow
- In the early 2000s, one of the Run 2 experiments, CDF, was building two tiered (cache/tape) data handling system based on so called filesets, groups of files put together on 10 GB tape partitions
  - Tapes needed to be partitioned before usage
  - Besides great I/O, filesets provided a pre-fetch mechanism which worked nicely as the data was typically colocated on a fileset by time
  - The project was eventually abandoned in favor of … you guessed it - dCache/Enstore, but not because of filesets.
  - I wish we had some of them, filesets tho!
- What if we had filesets built into dCache?

R/W SFA in dCache/CTA



- Regular files have default hsm set to sfa
- Container files have default hsm set to cta and go straight to tape
  - So only containers go to tape for a specific storage class
- I would like to explore this further