Karlsruhe Institute of Technology

New GridKa Tape Storage System
-from design to production deployment -

Dorin-Daniel Lobontu – on behalf of GridKa Team

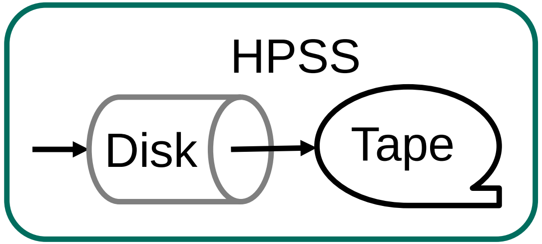
Agenda

- Requirements of the new tape storage system
- HPSS considerations
- HPSS setup and cache issue
- Productive – write
- Productive – stage
- Future plans

Requirements for the new Tape System

- Main goals:
  - Efficiently recall a large bunch of files O(100K) from tape
  - Maximize the transfer rates of the tape drives
- Write together the files that belong together
  - Recall optimization starts at the time of writing
- Large queues of read requests → tests with the former system CHEP2020
  - Read as much data as possible in one mount
- Tape file namespace decoupled from dCache/XRootD
- Impose upper limits on used resources to Vos
- Keep the initial layout of tapes
- Save the files alongside with their checksums
- Monitor every single component of the chain

HPSS considerations



- Data flows through the disk cache
  - can be bypassed for reading but not if FAR is used
- Aggregates written as standalone entities on tape
  - Recalling isolated files from an aggregate not recommended
  - Repositioning always from the first block of an aggregate, bad performance
    - Mitigated by “fast positioning” of the drives → not tested yet
  - Full aggregate recall – FAR
- FAR useful only if aggregates has been built cleverly
  - Directory vs. Create time

HPSS considerations

- GridKA max aggregate size 300GB
  - TS1160 400MB/s => 13 minutes to write/read an aggregate
- Caveat: be sufficient number of files on disk before migrating to tape
- Continue migration if grater than: 60 minutes of I/O
- Aggregates together only files from the same directory: **same dataset**
- File families used to group sets of files on sets of cartridges
  - All files from the **same dataset** assigned to the **same file family**
- Write to tape with only one stream/drive per file family
  - Files from the **same dataset** written on **the same cartridge.** Apply to small datasets

> - LFN structure in dCache might reflect how the files belong together
> - Alice: LFN directory structure no dataset meaning→ time based datasets

HPSS considerations

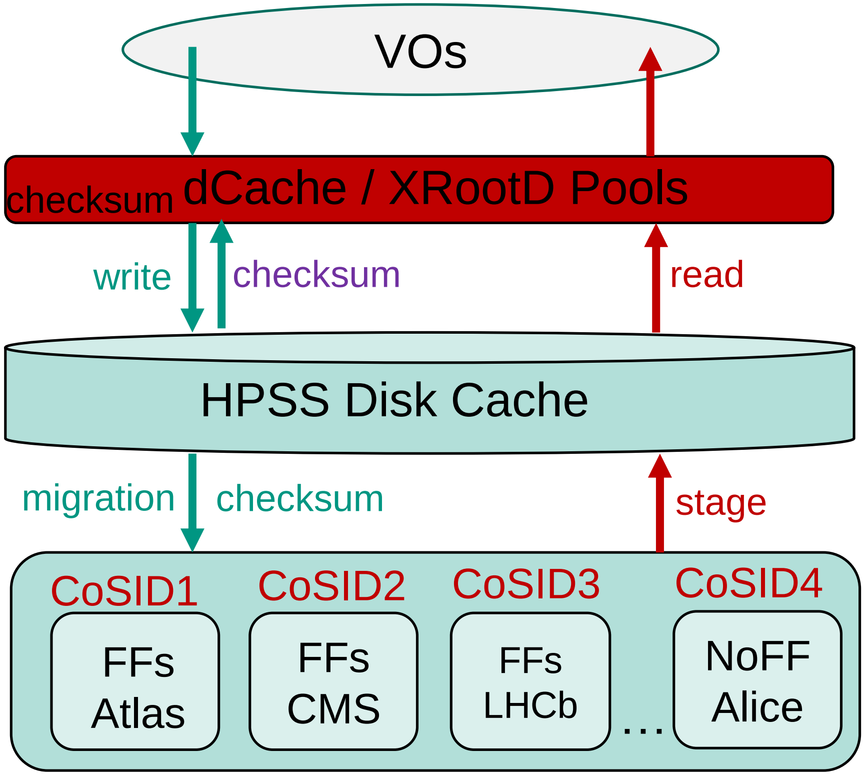
- Large datasets treated differently
  - One drive IBM TS1160 max. 400MB/s → HPSS cache might get filled up
  - Define several dedicated file families for big datasets
  - Dataset size hint from experiments needed → by dCache extended attributes
  - Provided only by Atlas for datasets > 40TB: use 8 dedicated file families
  - Up to 8 drives used to write big datasets to tape → up to 3.2 GB/s

Example:

- /mc/winter23/ztautau/0/output_1.file → dataset: /mc/winter23/ztautau/ → file family: 91 (mc: 91-94)
  ...
- /mc/winter23/ztautau/1/output_n.file → dataset: /mc/winter23/ztautau/ → file family: 91 (mc: 91-94)
- /data/run3/tau/output_1.file → dataset: /data/run3/tau/ → file family: 99 (data: 95-98)
  ...
  → Files from /mc/winter23/ztautau/{0,1} aggregated up to 300 GB within each directory individually

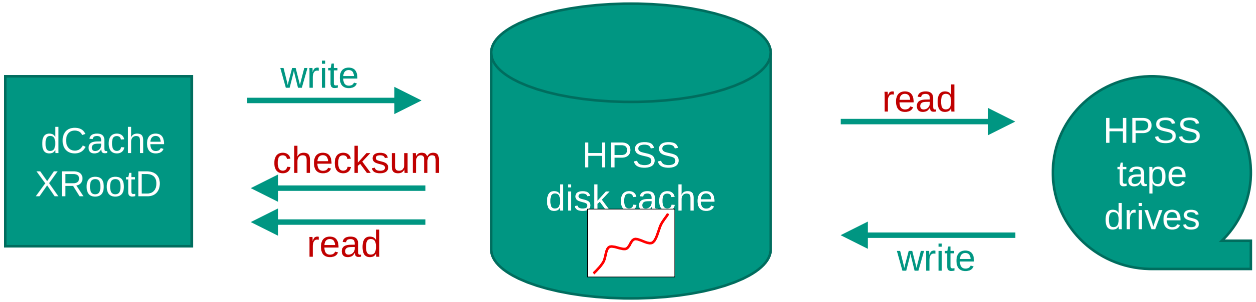
HPSS setup

- HPSS Cache shared by all VOs ~650TB
  - Only one migration policy → same aggregate size, same aggregation option: e.g. number of streams per FF for all VOs
- Predefined File Families (FF) for each VO
  - Alice no FFs → uses all migration streams permitted by the migration policy
  - Maximize the size of aggregates → minimize the number of used drives



HPSS Disk Cache Load

Details: [Andreas Petzold@HEPIX2023]



- Tier-1 writing to tape
  - Write from client + read from client for checksum
  - Writing to tape: read ~same as write from client

→ 2:1 read:write

- Tier-1 reading from tape
  - Read from tape: write on cache one stream per drive
  - Read from client: read from cache

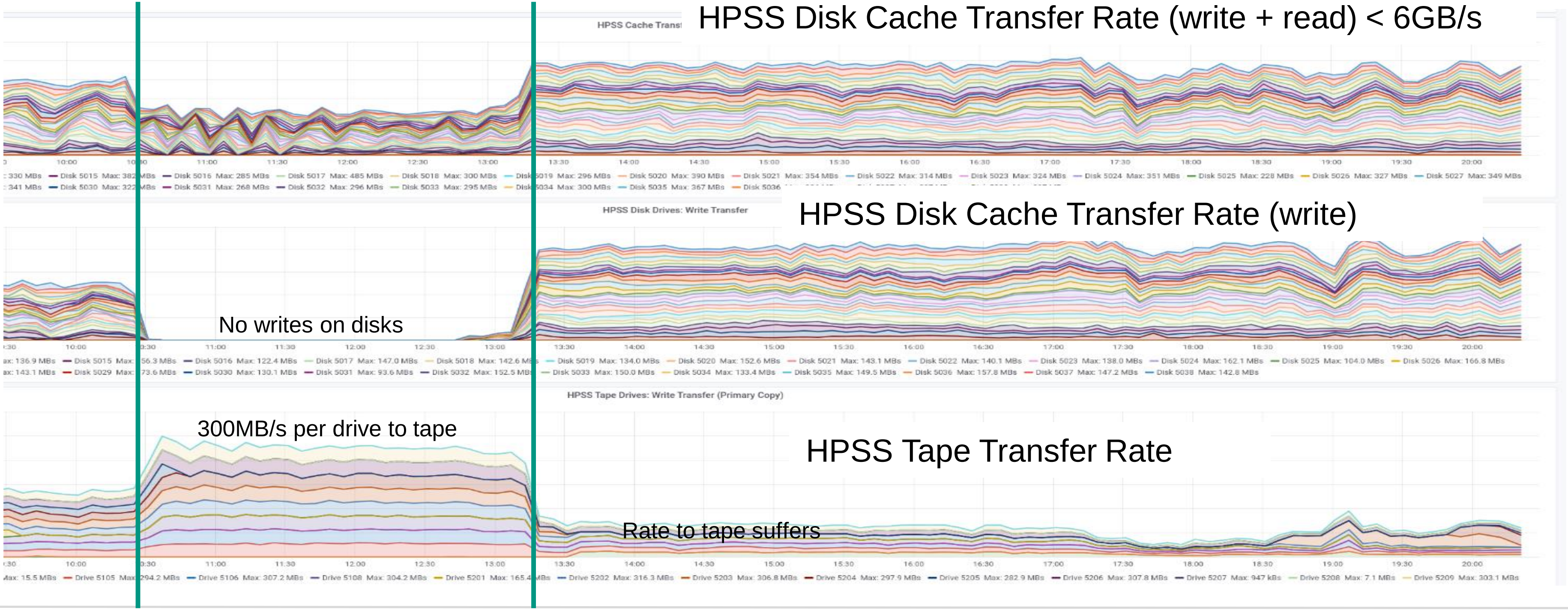
→ 1:1 read:write

- Streams to tape drives need to be stable

→ more IOPS

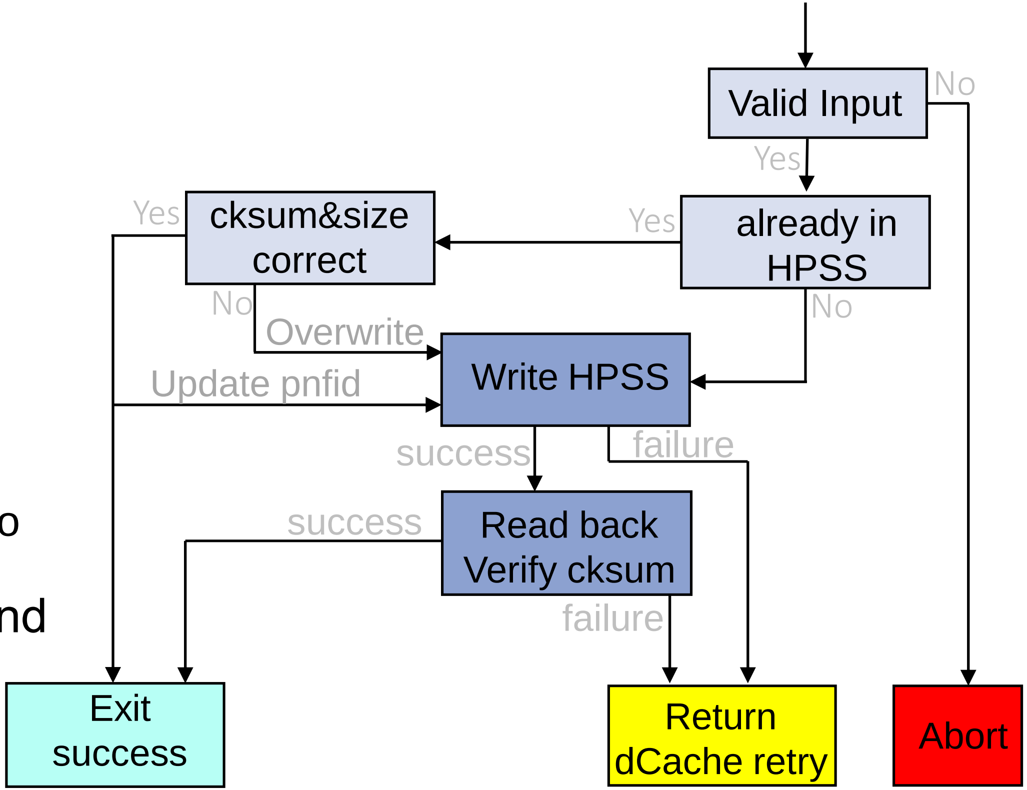
HPSS Disk Cache Load

Solution: NVMe SSDs and XiRAID



dCache HPSS Interface - write

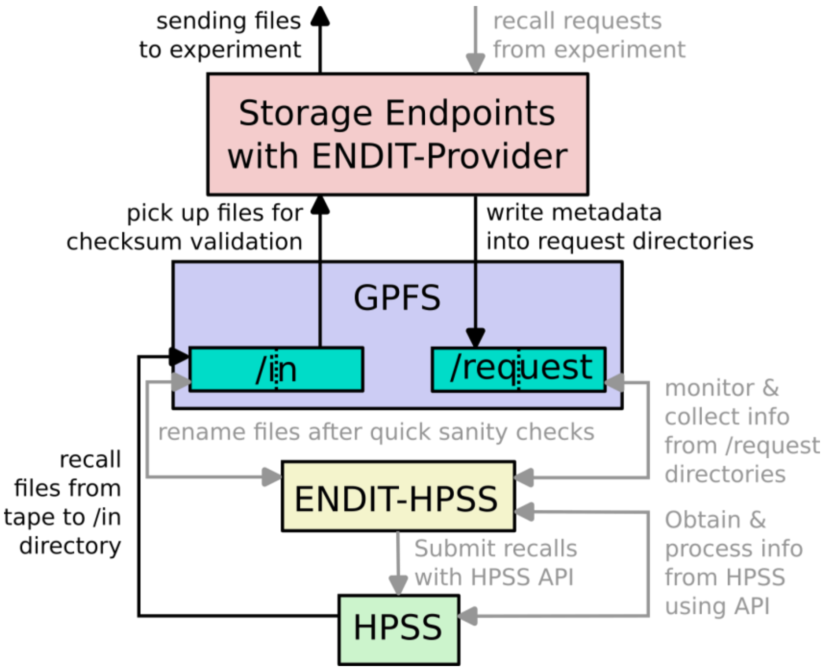
- dCache calls a script dc2hpss.py to write a file into HPSS
- Check if the file already in HPS
  - Request checksum and size from HPSS for validation
  - If correct only update pnfid
- Write a new file into HPSS
  - Compute dataset name → set file family
  - Compute HPSS path and transfer it to HPSS
- Successfully written → read back and re-compute checksum
- Return URI to dCache



Recall workflow of ENDIT

1. Collect metadata from recall requests
2. Use provided URI to obtain info from HPSS
3. Group requests by tape and aggregate
4. Put tapes in a processing queue
5. Process multiple tapes concurrently
   → number of used drives
6. For each tape, submit to HPSS a recall for **one file** per aggregate
   → triggers FAR for these aggregates
   → HPSS uses RAO for efficiency

1. Once submitted file recalled: iterate through remaining files from the same aggregate to recall them quickly
2. Once no aggregates left for a tape, pick new one from processing queue

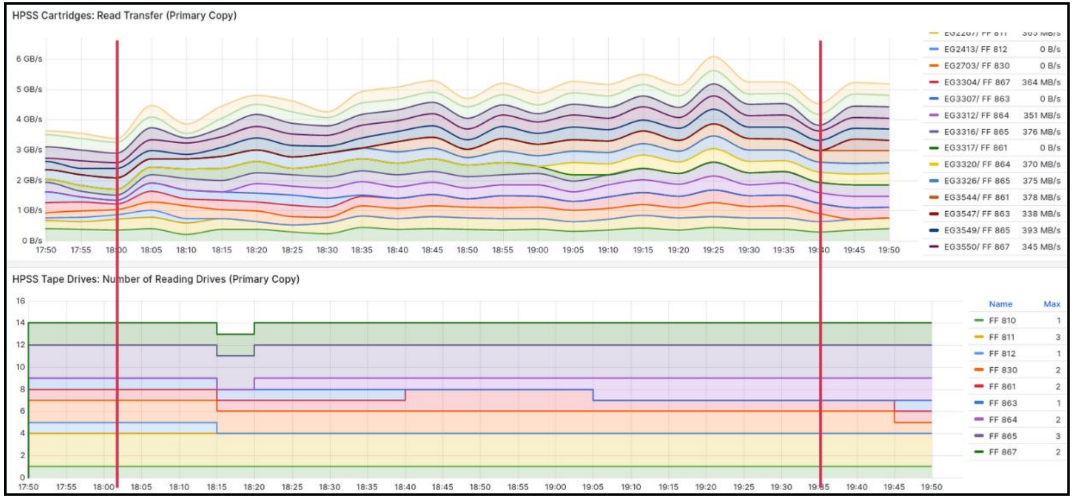


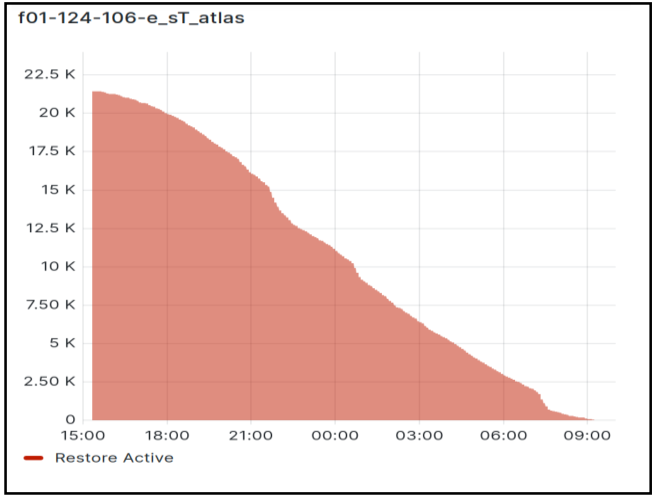
Details: Haykuhi@CHEP2023

Recall experience: ATLAS tests

Performance tests ATLAS, October 2023:

- Good performance in handling several 10k requests of ~ 100 TB volume
- 320 - 340 MB/s per drive on average
- More details presented by [Xin Zhao]



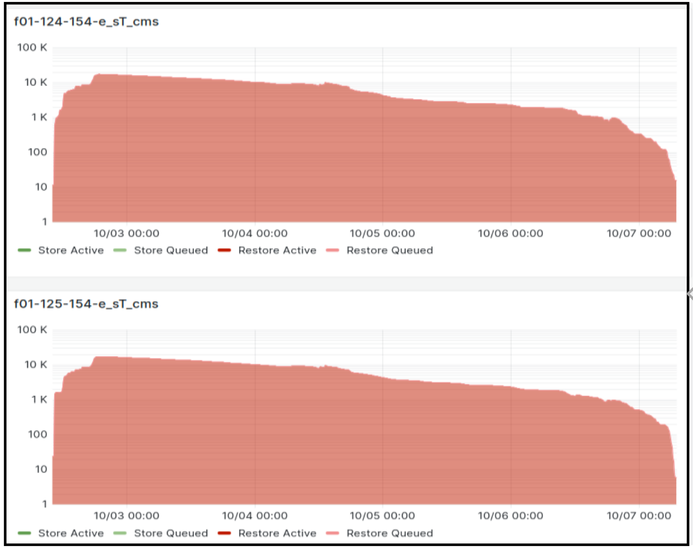


Proportion of requested vs. recalled files

We expect a dataset to be recalled entirely but this is not always the case

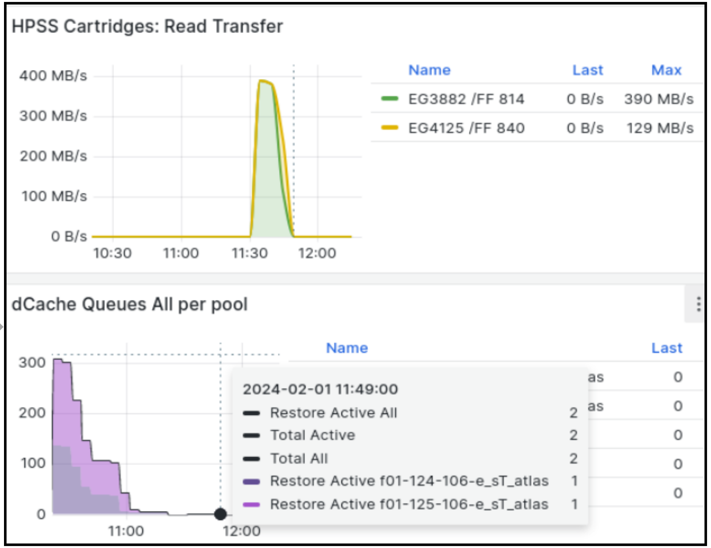
→worst-case just a **single file requested**

Main reason: a “disk replica” preferred over a “tape replica”



CMS recall campaign:
~35k requested files
Due to FAR ~102k files recalled → **~35%** files read

Atlas normal production
→ one file per aggregate
→ max 600GB staged



Future plans

- Look how to make use of “Read Queueing” of HPSS
  - A new set of APIs introduced lately
- Create a Read Queue (RQ) on HPSS and add reads to it
- More clients can add reads to the same RQ
- Clients ask HPSS which reads are ready to run
- Data ready to run means:
  - Data is on hpss cache
  - Data is on tape but the read can be immediately served
- Read Queues are persistent over HPSS restart

> - Read requests management bears by HPSS core alone
> - No need to keep read threads alive waiting for tapes to be mounted

Who made it to success

■ Department/tape group leader:	Andreas Petzold Doris Ressmann
■ HPSS Team:	Preslav Konstantinov, Karin Schaefer, Dorin-Daniel Lobontu
■ dCache Team:	Samuel Ambroj, Xavier Mol
■ GridKa Client – read:	Haykuhi Musheghyan
■ GridKa migration and client – write:	Artur Gottmann
■ Alice:	Max Kuehn
■ Tape libraries and drives:	Martin Beitzinger

Contact Email: name.surname@kit.edu