Solving Small Files Problem in Enstore.

Alexander Moibenko
FNAL

Work supported by the U.S. Department of Energy under contract
No. DE-AC02-07CH11359

Introduction

- Enstore is a tape based Mass Storage System designed to address needs of Run II FNAL experiments.
- Reliable and scalable data archival and delivery solutions.
- Meets diverse requirements of different experiments, such as CMS Tier 1, High Performance Computing, Intensity Frontier, data backups.
- Description can be found in:
  - The Fermilab data storage infrastructure
  - Fermilab's multi-petabyte scalable mass storage system

Problem

- File written to the tape, is followed by a file mark. Writing file mark affects small files the most due to tape drive stops and back-hitching.
- Industry offers File Sync Accelerator (FSA) to increase small file write performance. But it is not available for all types of tape drives we use.
- Tape positioning substantially affects overall read rates. Positioning takes 10 – 25 s. With non sequential access this substantially degrades the performance.
- Users want to aggregate small files transparently.

Currently available solutions

- T10000C has File Sync Accelerator (FSA) to increase small file performance. It maintains about 45 MB/s for small files , which is 5x slower than full drive speed. FSA is a complex feature and has been problematic for us.
- Buffered tape marks are not safe for end users, as the success is reported before data is on the media.
- Read ahead for reads – only good for sequential order.

Problem (Conclusion)

As tape capacity and speeds grow, the minimum desirable file size increases also. Eventually, any file becomes small.

We currently consider < 2GB to be small!

Small file aggregation solution

- Automatically aggregate small files into larger “container” files, with configurable definitions of “small” and “larger.”
- Transparently aggregate user's files through existing enstore interface (encp).
- Assume custodial ownership while staged to disk awaiting aggregation.
- Preserve end-to-end check-summing.
- Per customer "small file" policies.

What files to aggregate?

- Aggregation of files shall account for read access patterns. Only the experiment, or no one, knows what read patterns will be.
- File aggregation policy must be flexible enough to adapt to different patterns without changing code.
- Aggregation of files by file storage group and family is a good starting point. Each experiment or project has a unique storage group to control and balance resources (media, drives). File family is the name which defines a category of data files. Detailed description can be found in "Enstore User Guide".

Policy Entry

- Original (tape) library.
- Resulting (disk) library.
- Storage group
- File family
- File family wrapper.
- Minimum file size – do not aggregate if size is bigger.
- Maximum number of files in a package.
- Maximum waiting time – if exceeded aggregate all files and write to tape.

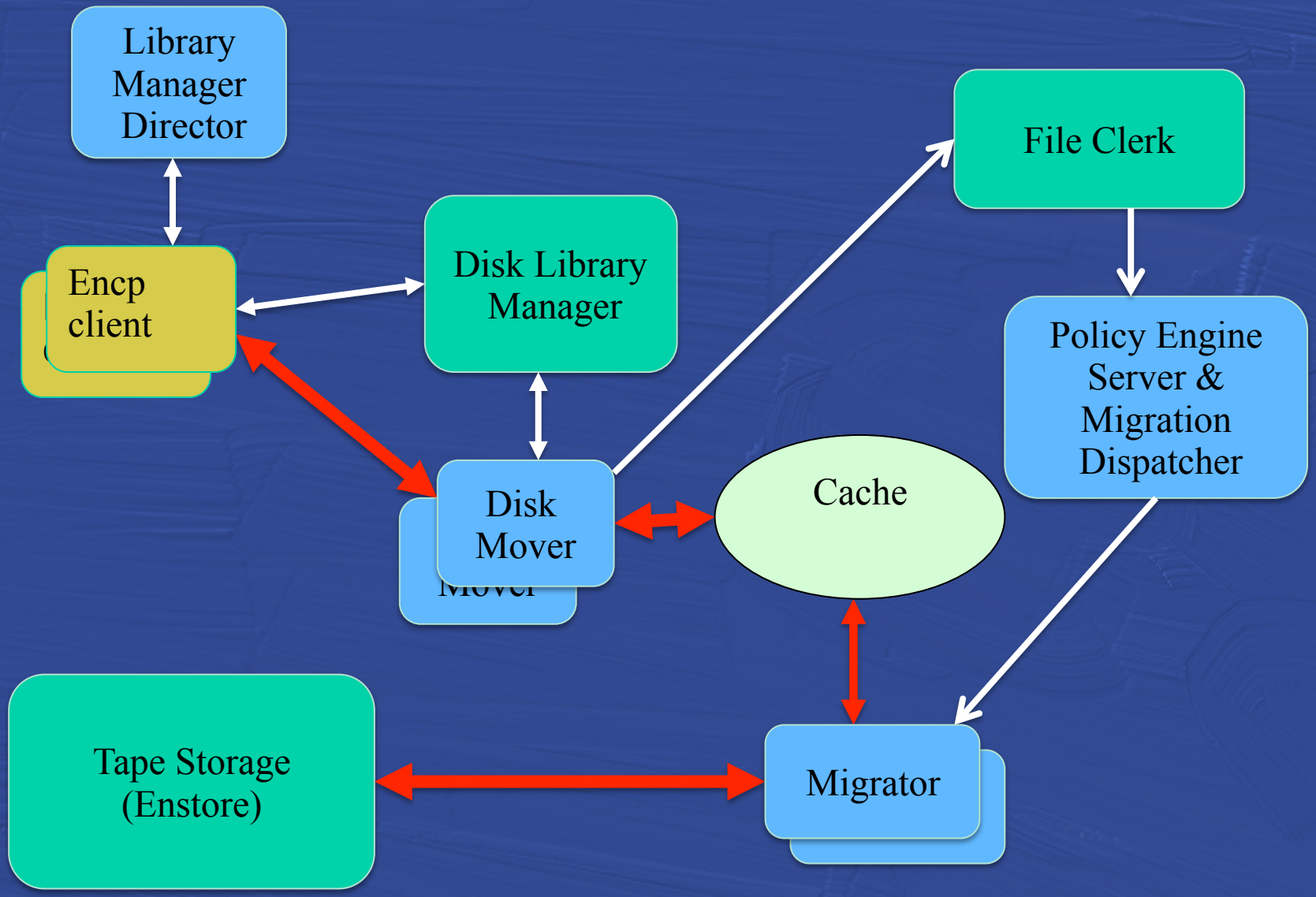
Implementation requirements

- **File caching component** inside Enstore:
  - temporary storage for incoming files, containers, and unpacked files – must be almost as safe as tape.
  - full control over cache disk access to optimize:
    - IO bandwidth to tape.
    - Concurrent Read and Write operations.
  - Scalability of cache by adding new HW and SW.
- Backward compatible with current client tools (encp).
- Must not affect users who do not use aggregation.

HW Cache selection

- Nexenta Appliance , ZFS - RAIDZ2 and checksum protect data on disk (and also some protection against memory errors).
- Perform well enough to aggregate and possibly act as migration vehicle (to make larger packages out of smaller).

Structure of integrated data caching and tape system using encp and disk movers.



Description of components.

- Library Manager Director receives write requests from encp and determines whether to send data to tape directly or to cache for aggregation.
- Disk Library Manager – Enstore library manager configured for disk movers.
- Disk Movers – transfers files between client (encp) and disk cache.
- File Clerk – modified to send events to Policy Engine Server.

Description of components (cont.)

- Policy Engine Server receives event from file clerk specifying that the **new** file arrived into cache or the file written to tape needs to get staged from it. It has 3 types of file lists:
  - Archive – files to be written to tape.
  - Stage – files to be staged from tape(s) to cache.
  - Purge – files to be purged in cache.
- Migration Dispatcher – receives file lists from Policy Engine Server and dispatches them to migrators.

Description of components (cont.)

- Migrators aggregate data in cache and write container to tape. They stage aggregated data and unpack files for read requests. All files in a container read from tape get unpackaged and cached, even if not requested.

Package file.

SFA package is a self described container (tar).

Each package has a special file README.1ST containing information about each data file in container:

- File path in cache.
- File name in the namespace.
- File checksum.

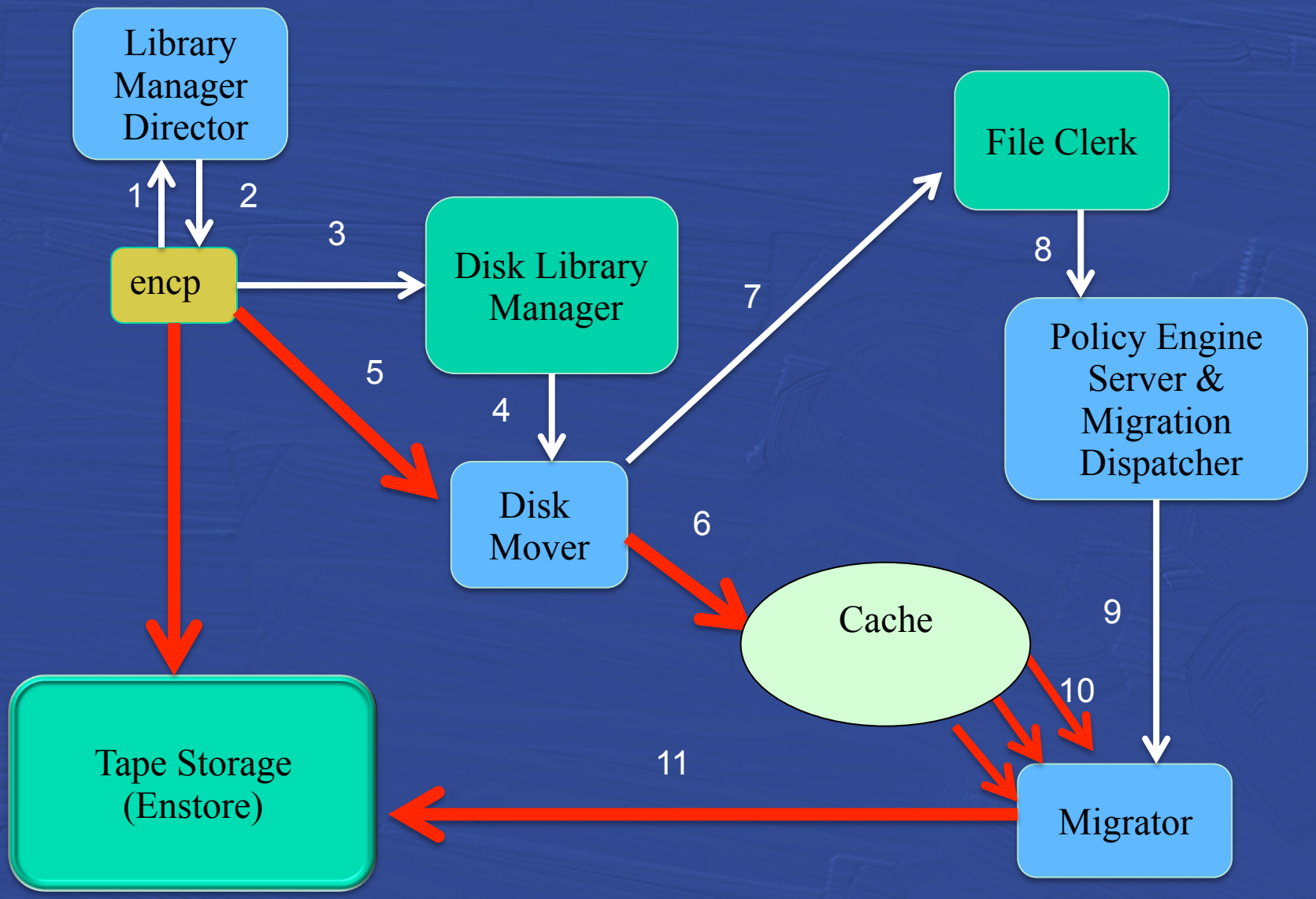
The package file gets written into Enstore and placed in the namespace in a separate directory including the name of the volume it resides on.

New enstore servers

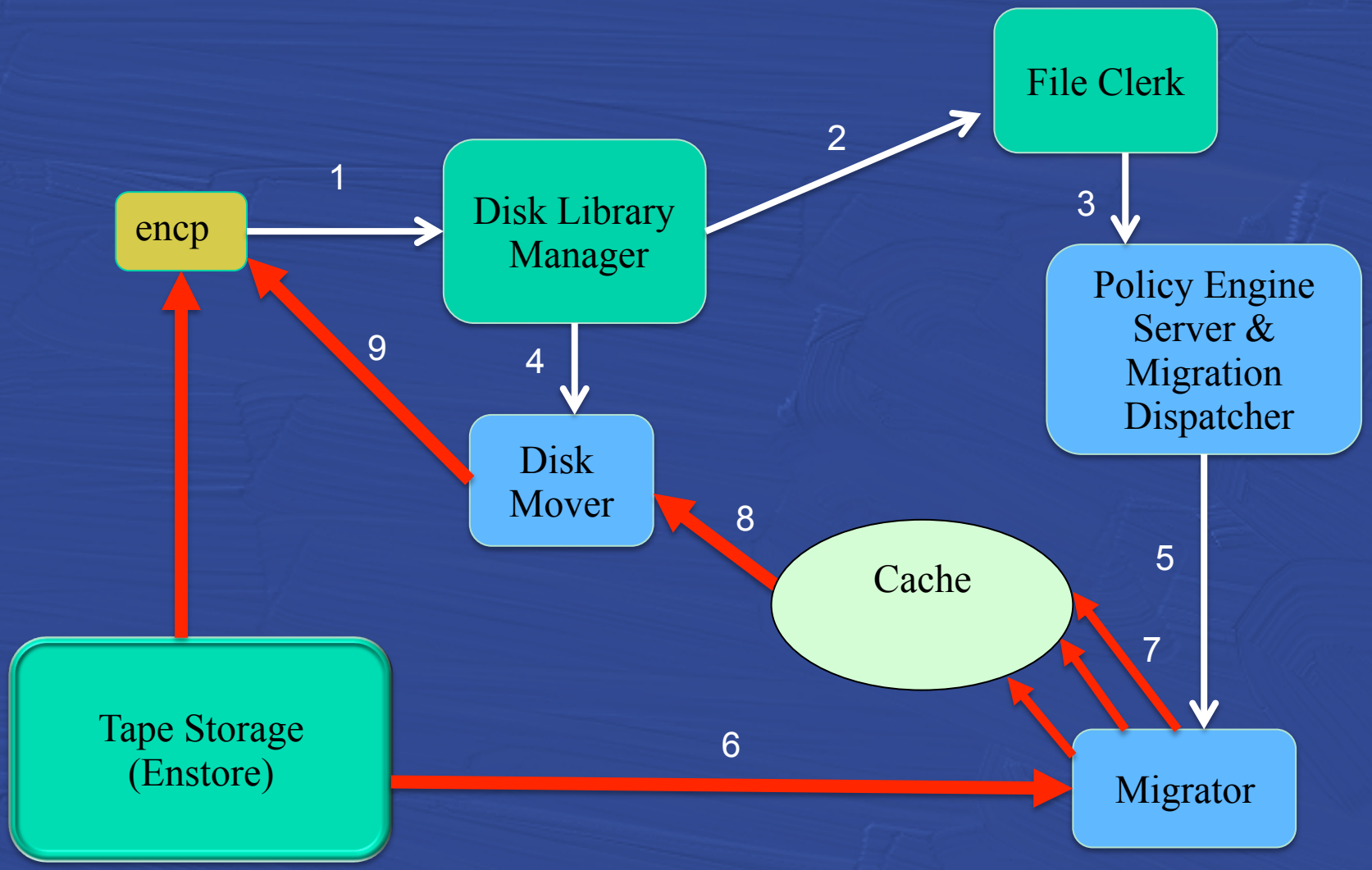
- Library Manager Director.
- Dispatcher (Policy Engine Server & Migration Dispatcher).
- Migrator.

Communications between File Clerk, Dispatcher and Migrators are done using Apache QPID AMQP messaging system.

Writing Files



Reading Files



Cache organization

- One common cache.
- Different cache areas (configured in production):
  - Write cache.
  - Archival cache.
  - Stage cache .
  - Read cache.

Data integrity

- ZFS server is used as Enstore data cache to reliably hold user data while not on tape.
- Verify package member file checksums from sample packages just before writing them to tape.
- Table of files in transition in file database to track and resubmit cached files not yet archived to tape. File clerk periodically checks this table and sends alarm about these files.

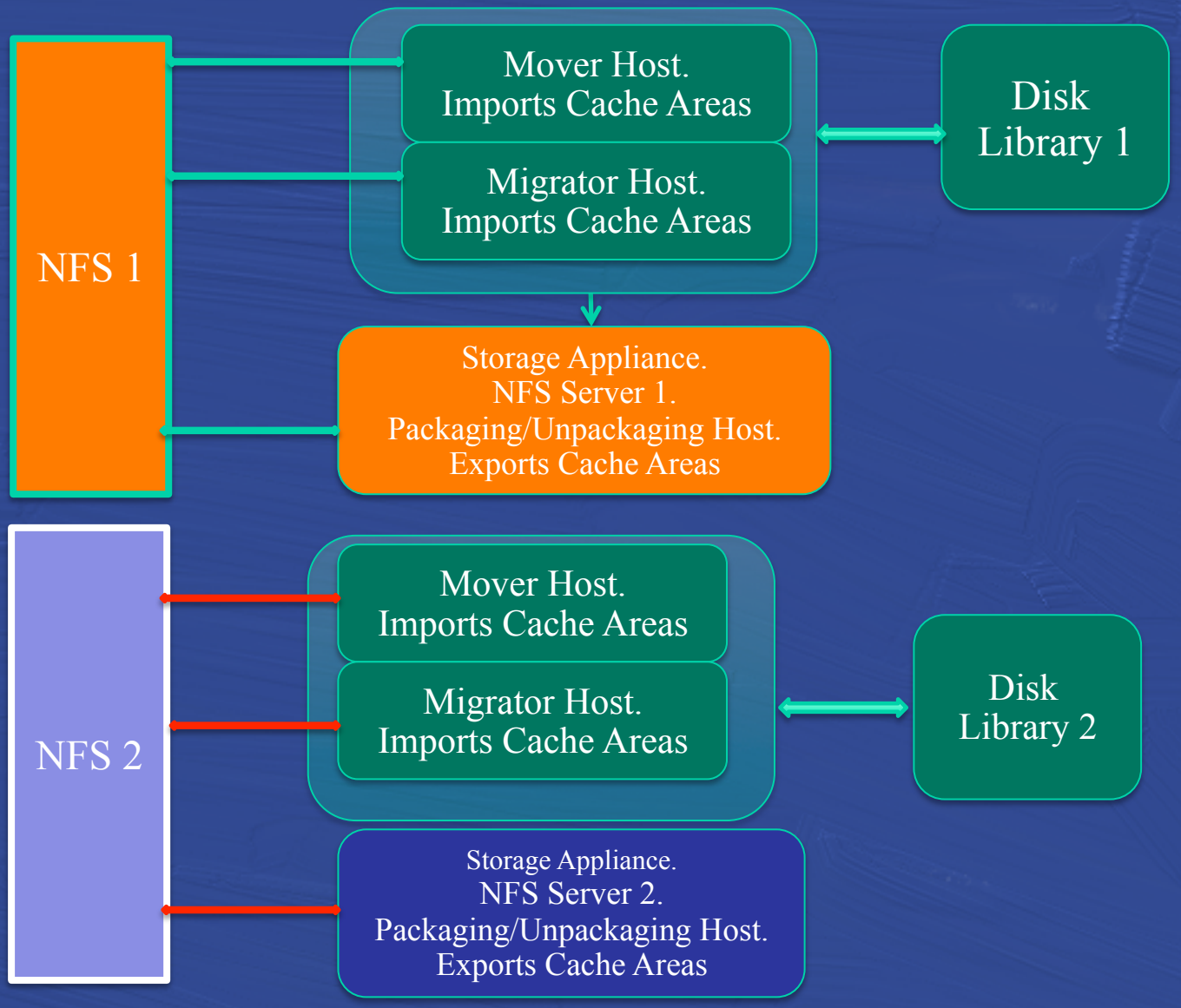
Performance of cache considerations

- Design of the system allows for separate cache areas for reads and writes.
  - Configured for performance to not allow read from write cache. Files are not accessible until written to tape.
- Nexenta Appliance configured so that it has 3 separate groups of disks for:
  - User written files (22TB).
  - Package / Archive (2 TB).
  - Stage / User reads (22TB).

Scaling Cache

- Group disk movers and migrators around selected disk library manager.
- Assign separate disk areas to selected disk libraries.
- Currently 2 cache clusters using 2 separate Nexenta Appliances.

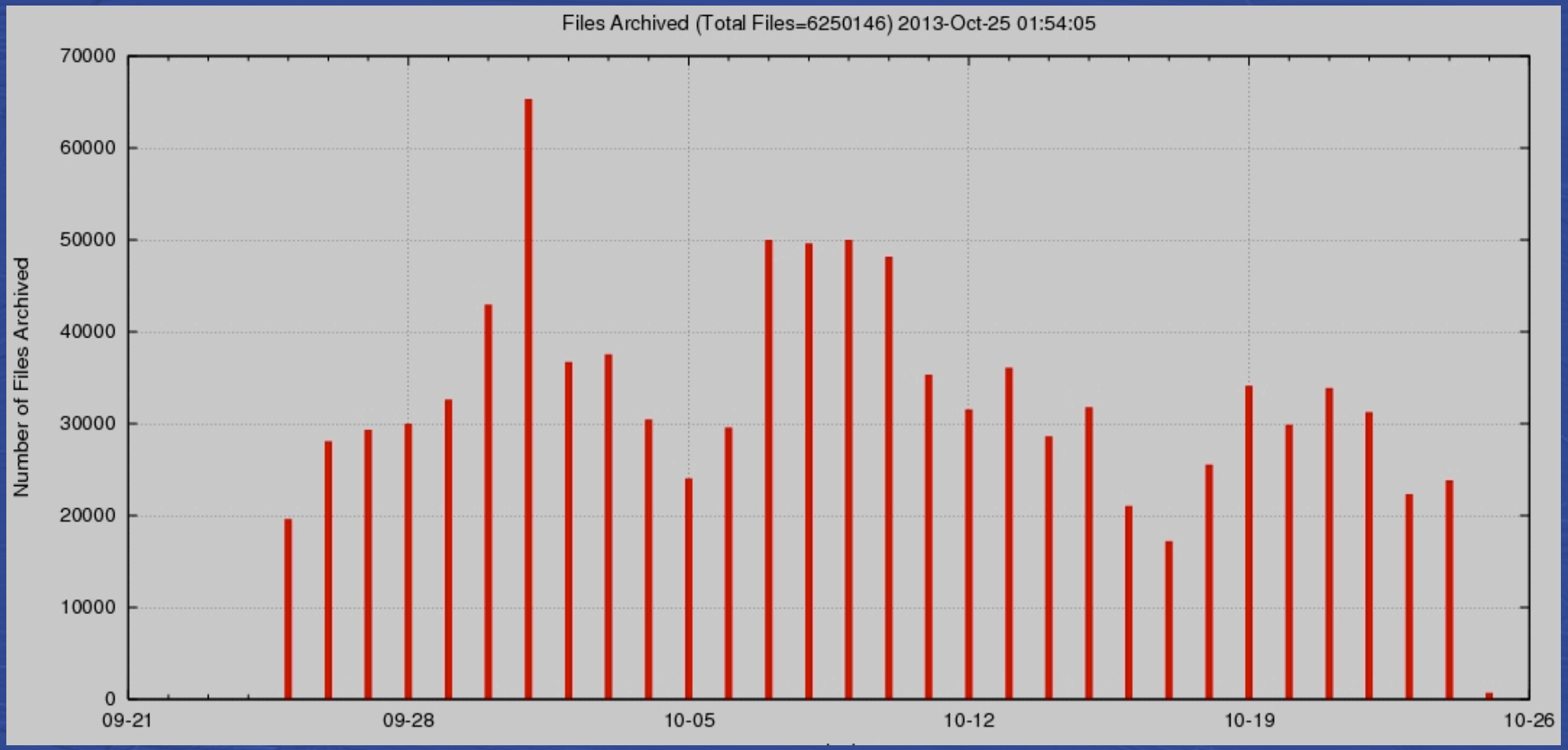
Scaling cache through clusters.



Current state

- In production since April 2012.
- Used by experiments: NOvA, MINERvA, LQCD, ArgoNeuT.
- Configured with 2 clusters beginning September 2013.
- Total number of files > 6,000,000.
- Total data size > 0.5 PB.
- Integrated with Enstore file duplication, command line interfaces, etc.

Files archived for the last month



Plans for the future

Support re-packaging during migration of files from one media to another.