Introduction to Enstore

10th international dCache users workshop

April 11-12, 2016, Hosted by PIC

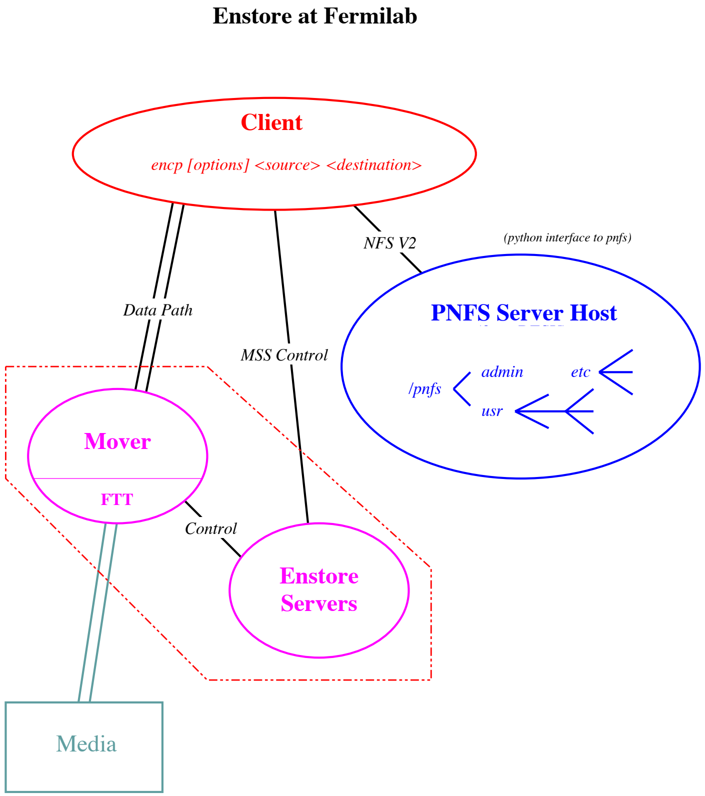
Barcelona, Catalunya

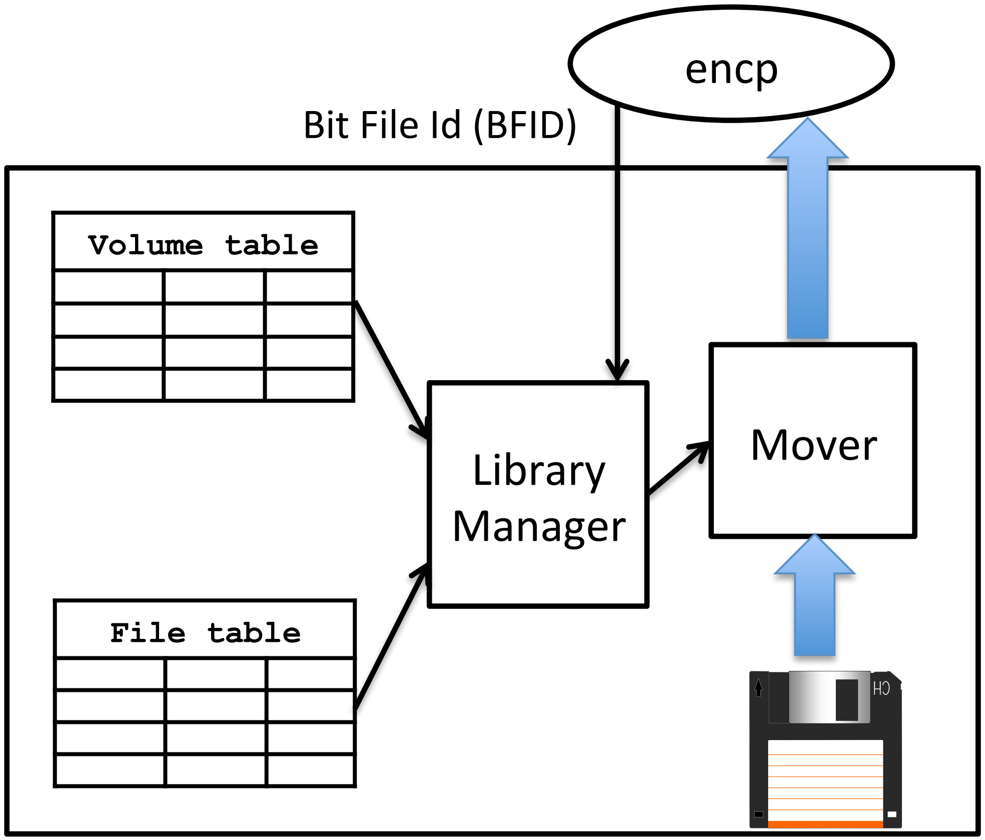
Brought to you by Enstore team: Dmitry Litvintsev and Alexander Moibenko

Some (ancient) History

- 1997
  - April, CHEP’97: Patrick Fuhrmann presents A Perfectly Normal Namespace for the DESY Open Storage Manager (name obviously inspired by THGTTG)
  - Fermilab is looking for alternatives to HPSS for mass storage needs of Run 2 Tevatron experiments (D0 and CDF)
  - Don Petravick learns about PNFS and OSM
  - December: Don Petravick visits DESY and develops Enstore prototype in python

Christmas prototype





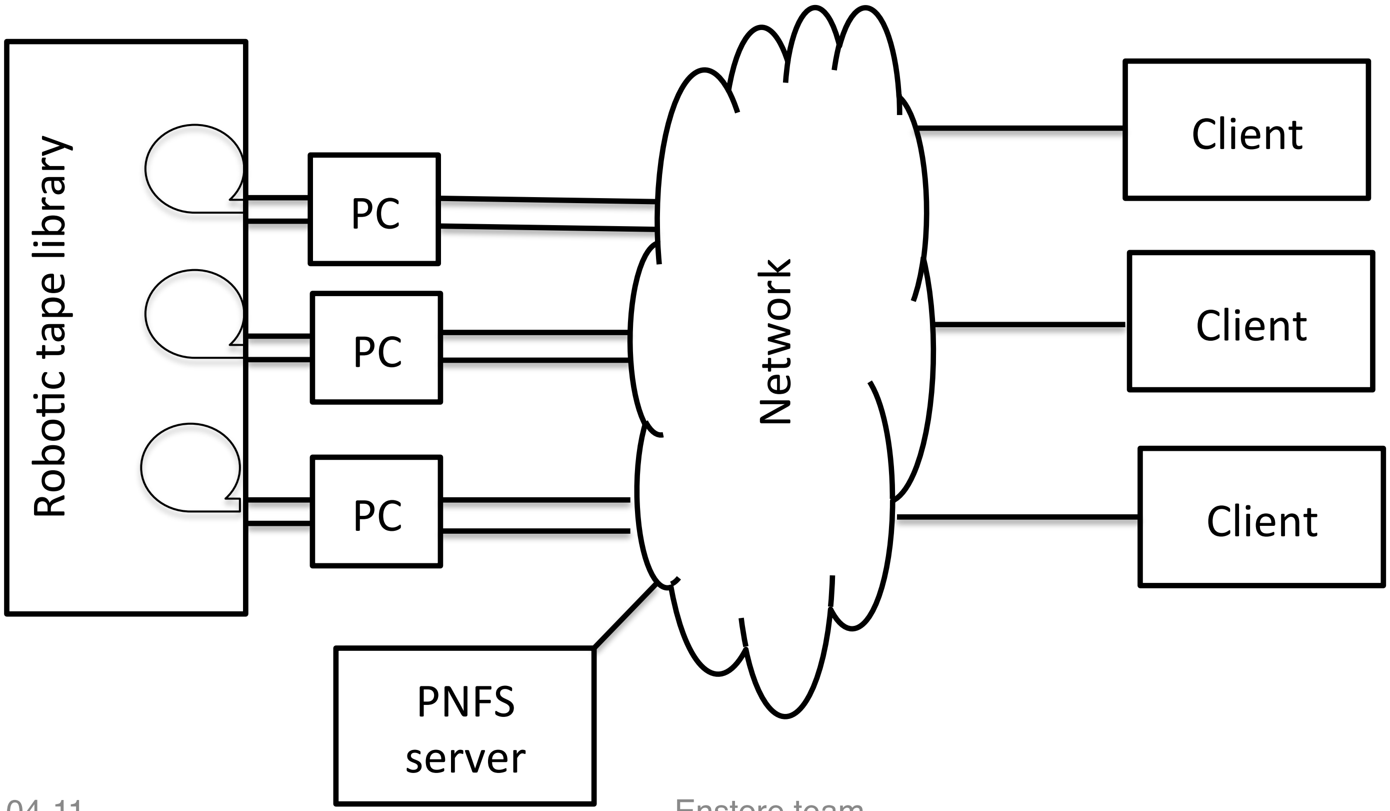
Concept

- s/OSM/Enstore/g
- Use FTT library (Fermi Tape Tools) for tape I/O

Prototype:

- 133 MHz Pentium w/ 16MB RAM
- Floppy emulating tape drive
- Distributed design (multiple movers)

- Continued large scale testing back at Fermilab showed that the basic principles were solid:
  - Python is usable language for large scale system development
  - Actual data transfers in C
  - Network attached drives
  - Distributed server components



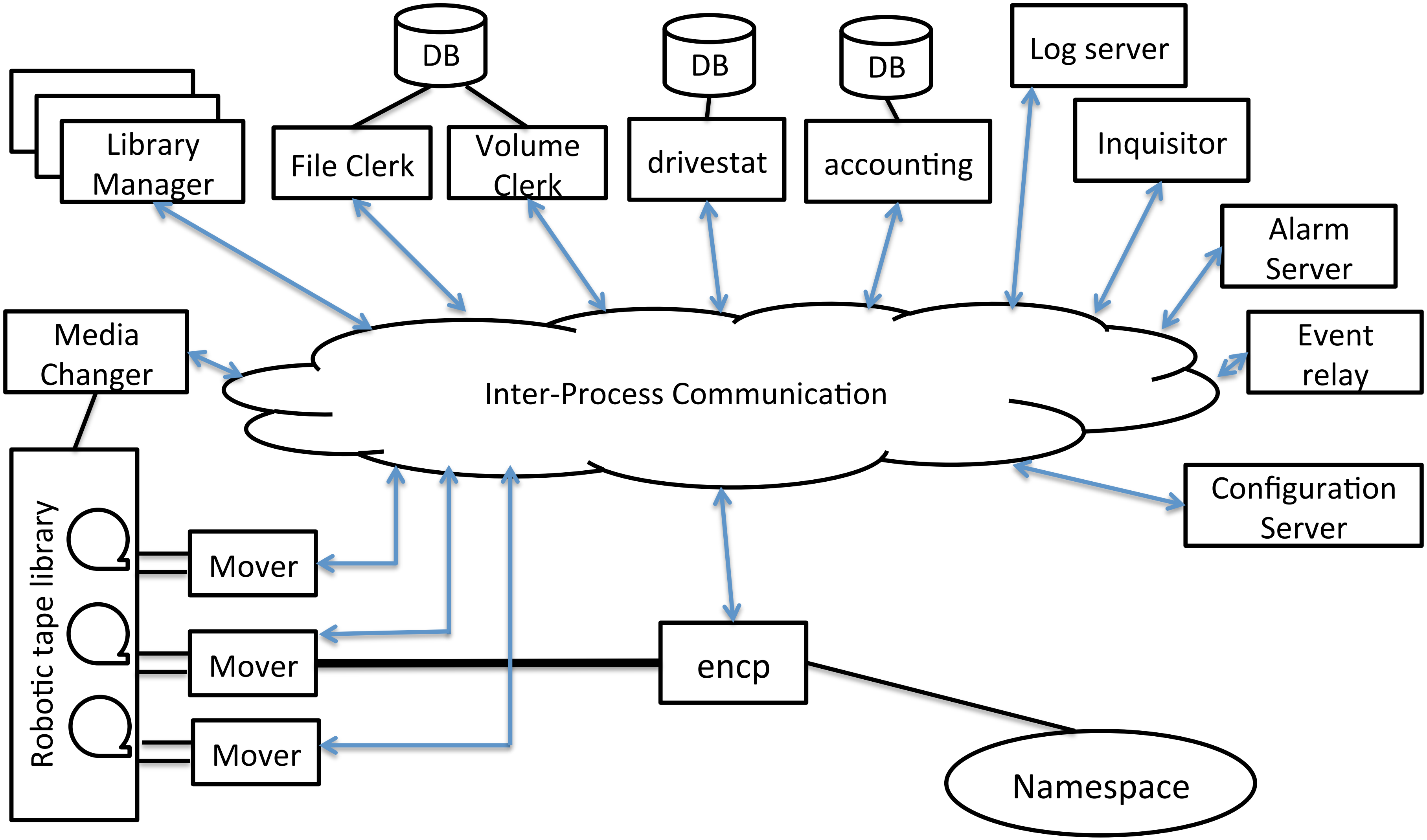
Purpose

- Enstore system was designed to meet and exceed requirements of the Run 2 Tevatron experiments (~ 20 PB, 0.5GB/s aggregate throughput)
- It has evolved into feature rich, primary data storage solution that underpins Fermilab scientific program that includes CMS T1 with total data capacity exceeding 100 PB and aggregate throughput approaching 5GB/s

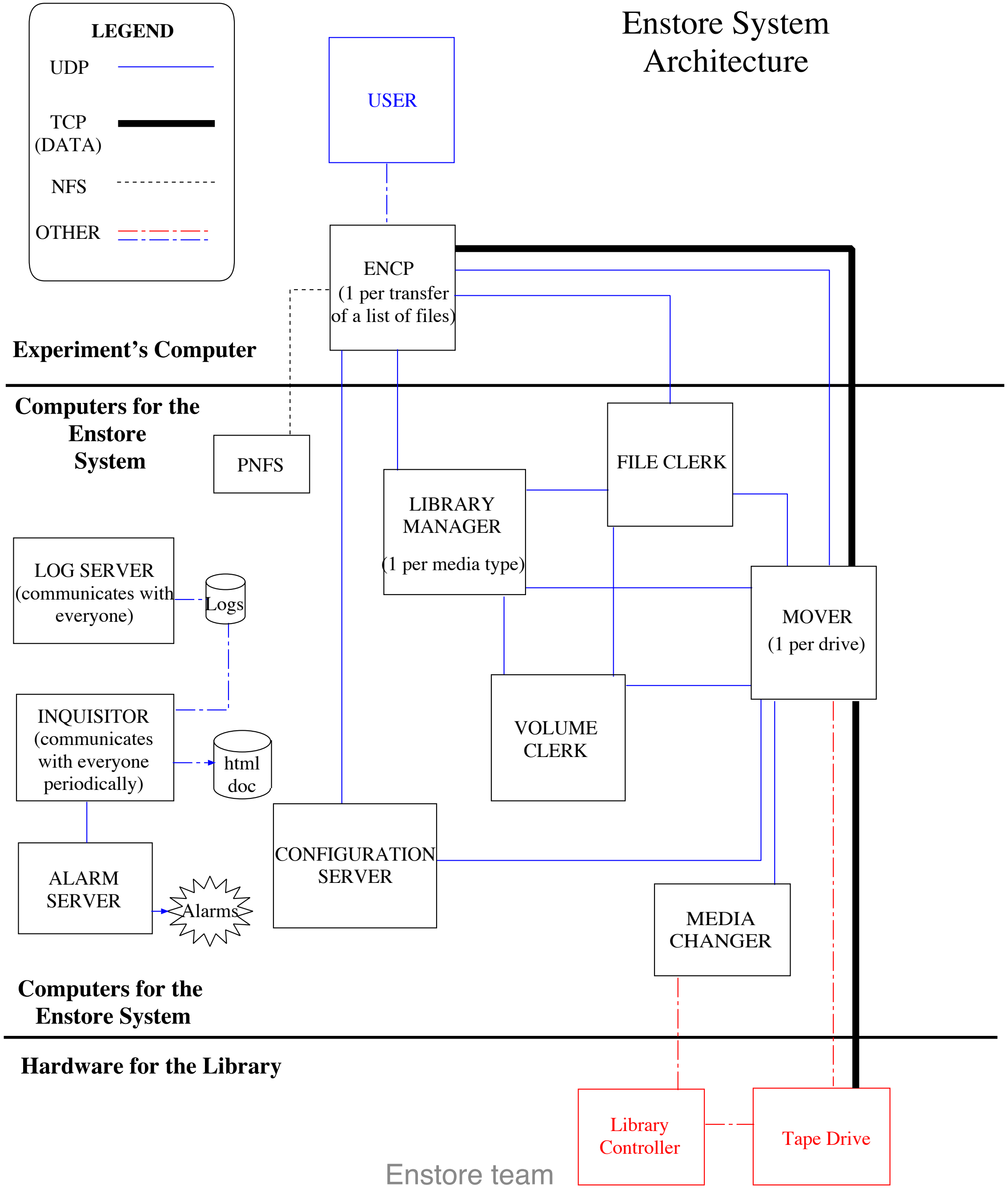
Design considerations

- Client/Server Architecture
  - Reuse python server framework
- Networked distributed drive access (crucial for meeting scalability requirements)
- Portability
  - Python
  - Time critical code in C
- Communication Protocols
  - UDP for control request/response
    - Messages fit in 1 datagram
    - Retries, unique ID
    - Clients can not hang servers
  - TCP for data transfers
- Products reuse
  - Fermilab FTT – portable hooks for handling tape drives
  - PNFS (later chimera) namespace
  - Apache web server for web based monitoring and admin interface
  - PostgreSQL DB
  - Gnuplot utility for monitoring and performance metrics
  - crond for scheduling various ancillary tasks

Enstore Zoo: a bird’s eye view



Enstore System Architecture


LEGEND
UDP
TCP (DATA)
NFS
OTHER
USER
ENCP (1 per transfer of a list of files)
Experiment's Computer
Computers for the Enstore System
PNFS
FILE CLERK
LIBRARY MANAGER (1 per media type)
LOG SERVER (communicates with everyone)
Logs
MOVER (1 per drive)
INQUISITOR (communicates with everyone periodically)
html doc
VOLUME CLERK
ALARM SERVER
Alarms
CONFIGURATION SERVER
MEDIA CHANGER
Computers for the Enstore System
Hardware for the Library
Library Controller
Tape Drive

Namespace

- PNFS/Chimera namespace is used to:
  - Present user data as familiar hierarchical filesystem
  - Store MSS related information associated with each file in layers. Enstore client, encp, uses:
    - Layer1 to store Bit File ID (BFID)
    - Layer4 to store additional (partially redundant) information that potentially would allow to reconstruct the entire MSS data catalog in case of loss
  - Store per-directory data steering and resource management information by utilizing PNFS tags. The following tags are used:
    - storage_group (usually name of an experiment)
    - library (name of the virtual tape library)
    - file_family (name of a dataset to be grouped on the same tape/set of tapes)
    - file_family_width (how many movers can be used simultaneously)
    - file_family_wrapper (file wrapper type)

Layer examples

- Layer 1:

```
cat ".(use)(1)(ep047d08.0042dila)"
CDMS139575977795761
```

- Layer 4:

```
cat ".(use)(4)(ep047d08.0042dila)"
VOO534
0000_000000000_0002378
1359778439
dcache
/pnfs/fnal.gov/usr/test/litvinse/world_readable/ep047d08.0042dila

0000B08FD2359604479283B589689FC8892B

CDMS139575977795761
stkenmvr218a:/dev/rmt/tps6d0n:1310297132
2886985789
```

Tags examples

```
 grep "" $(cat ".(tags)()")

.(tag)(file_family):nova_production
.(tag)(file_family_width):6
.(tag)(file_family_wrapper):cpio_odc
.(tag)(library):CD-10KCF1
.(tag)(OSMTemplate):StoreName sql
.(tag)(sGroup):chimera
.(tag)(storage_group):nova
```

Enstore client: encp

- encp copies data between media and user disk.
- Mimics Unix cp command.

```
encp --help
Usage:
       encp [OPTIONS]... <source file> <destination file>
       encp [OPTIONS]... <source file> [source file [...]] <destination directory>
       encp [OPTIONS]... --get-bfid <bfid> <destination file>
       encp [OPTIONS]... --get-cache <pnfs|chimera id> <destination file>
       encp [OPTIONS]... --put-cache <pnfs|chimera id> <source file>
```

- Writes: destination file/directory is located in PNFS/Chimera namespace.
- Reads: source file is located in PNFS/Chimera namespace.
- Distributed as statically linked executable produced with Python freeze tool => Requires no dependencies.
- Control communication uses UDP => cannot hang shared Enstore servers.
- Data transfer to/from Mover use TCP.
- Provides end-to-end checksum.

encp

- Write:
  - – Extracts steering information from destination directory tags.
  - – Sends info to library manager.
  - – Enstore sets up file transfer.
  - – Reads data from disk and writes to mover on TCP socket.
- Read:
  - – Reads file BFID from namespace
  - – Sends BFID to file clerk
  - – Enstore sets up file transfer
    Reads data from mover on TCP socket & write to disk

Configuration Server

- Configuration Server maintains and distributes all information about system configuration such as host, port and other parameters of each server.
- On startup, each Enstore server queries the Configuration Server for :
  - Information on how to setup itself.
  - Locations of other servers it needs to communicate.
- A new configuration can be loaded into the Configuration Server from a file without disrupting the running system.
- Configuration is stored in a file in a form of python dictionary
- The only well known port in the system.

```
export ENSTORE_CONFIG_HOST=example.com

export ENSTORE_CONFIG_PORT=7500
```

Library Manager

- Library Manager (LM) queues up and dispatches work for a virtual library. There is one LM for each virtual library.
- Virtual library is a collection of tape volumes of the same media type in a physical tape library and movers that use these tapes.
- Main job of LM is to submit transfer requests to its movers:
  - Waits for mover to contact it and gives it request if mover in state IDLE or HAVE_BOUND
  - Tells movers which volume to mount (draws volume from family or blank on writes, or volume containing file on reads)
- Serializes independent requests on same volume and sorts them by file location on tape for efficient access.
- Transfers can prioritized based on flexible criteria
- Implements fair-share and discipline (controls how many simultaneous transfers can be made to/from the same host)
- Checks if the "width" of volumes already active
- Maintains two queues:
  - Pending requests
  - Work at mover requests

File Clerk and Info server

- File Clerk tracks all files in the system.
- There is one record for each file keyed on BFID.
- File records are persisted to DB (PosgtreSQL is used for DB backend)
- File Clerk serves as DB frontend and provides:
  - Object to relation mapping of file records
  - Unique BFID generation
  - DB connection handling
  - Request processing thread pool and request queuing
- Info server is essentially read-only version of File Clerk and is used to query file records by command line tools and by other Enstore servers

Volume Clerk

- Volume clerk tracks tapes in the system
- There is one record for each volume keyed on unique volume label
- Volume records are persisted to DB (PosgtreSQL is used for DB backend)
- There is very simple DB structure one to many relation between file and volume table with foreign key in file table on volume id.
- Volume Clerk is architected similar to File Clerk
- Volume Clerk responsibility is to:
  - Assign new volumes
  - Draw volumes for write on request from Library Manager
  - Provide interface to query and modify volume information
  - Maintain tape quotas by storage_group and library

Media Changer

- Media changer represents a physical device performing mounts/dismounts of volumes per request from Movers on drives by talking to library specific control micro
- May serve multiple drives and Library Managers
- The Media Changer issues multiple simultaneous commands by forking processes that do the work. There is a certain preset limit (MAXWORK) of forked processes.
- If Media Changer receives mount/dismount requests while there are MAXWORK unfinished operations then new these requests are discarded, Mover requests time out and retried.

Log server

- Receives messages from other processes and logs them into formatted log files.
- Log files are labeled by dates.
- Files are rotated at midnight.
- Clients do not retry on UDP messages
- System can not hang because of full logfile partition.
- Format: timestamp clienthost UID Username ClientName message:

```
16:46:43 enmvr064.fnal.gov 006676 root I 10KC_064MV  Updating stats Thread media_thread
16:46:43 enmvr064.fnal.gov 006676 root I 10KC_064MV  Ejecting tape Thread media_thread
16:46:43 cmsstor271.fnal.gov 001553 root I ENCP  using request cmsstor271.fnal.gov-1460225802-1553-0 instead for error processing
16:46:43 cmsstor271.fnal.gov 001553 root W ENCP  ('RESUBMITTING', 'cmsstor271.fnal.gov-1460225802-1553-0')
16:46:43 cmsstor271.fnal.gov 001553 root I ENCP  Sending /pnfs/.(access)(000001B0E8F6E19048A9A9CDE95E611FC772) read request to
LM: unique_id: cmsstor271.fnal.gov-1460225802-1553-0 inputfile: /pnfs/.(access)(00007CFBDE0E9BCB44A9BFEAE8B9D6728FB2)/.
(access)(000001B0E8F6E19048A9A9CDE95E611FC772) outputfile: /storage/data2/write-pool/data/
000001B0E8F6E19048A9A9CDE95E611FC772
```

Mover

- Reads files from tapes and sends data to user.
- Reads data from user and writes file to tape.
- TCP for data transfers.
- Each tape drive has its own mover process
  - One computer can run many movers
  - One mover can belong to many library managers.
- Steps in transfer:
  - When idle, asks library manager for work
  - Library manager tells mover to mount volume x
  - Mover calls media changer to mount volume
  - Library manager gives mover transfer requests
  - Mover contacts encp
  - Mover transfers requested data to user

Mover

- Reads:
  - Using the file location_cookie, position tape to the beginning of data.
  - Read wrappering information that precedes the actual data.
  - Fork a process that reads and calculates crc on the data from the volume and placing the data into a shared memory buffer.
  - Write data from the shared memory to the encp process over TCP socket.
  - Read any wrappering information that comes after the data.
  - Close the data port.
  - Tell the user done and all is well.
  - Close the control port

Mover

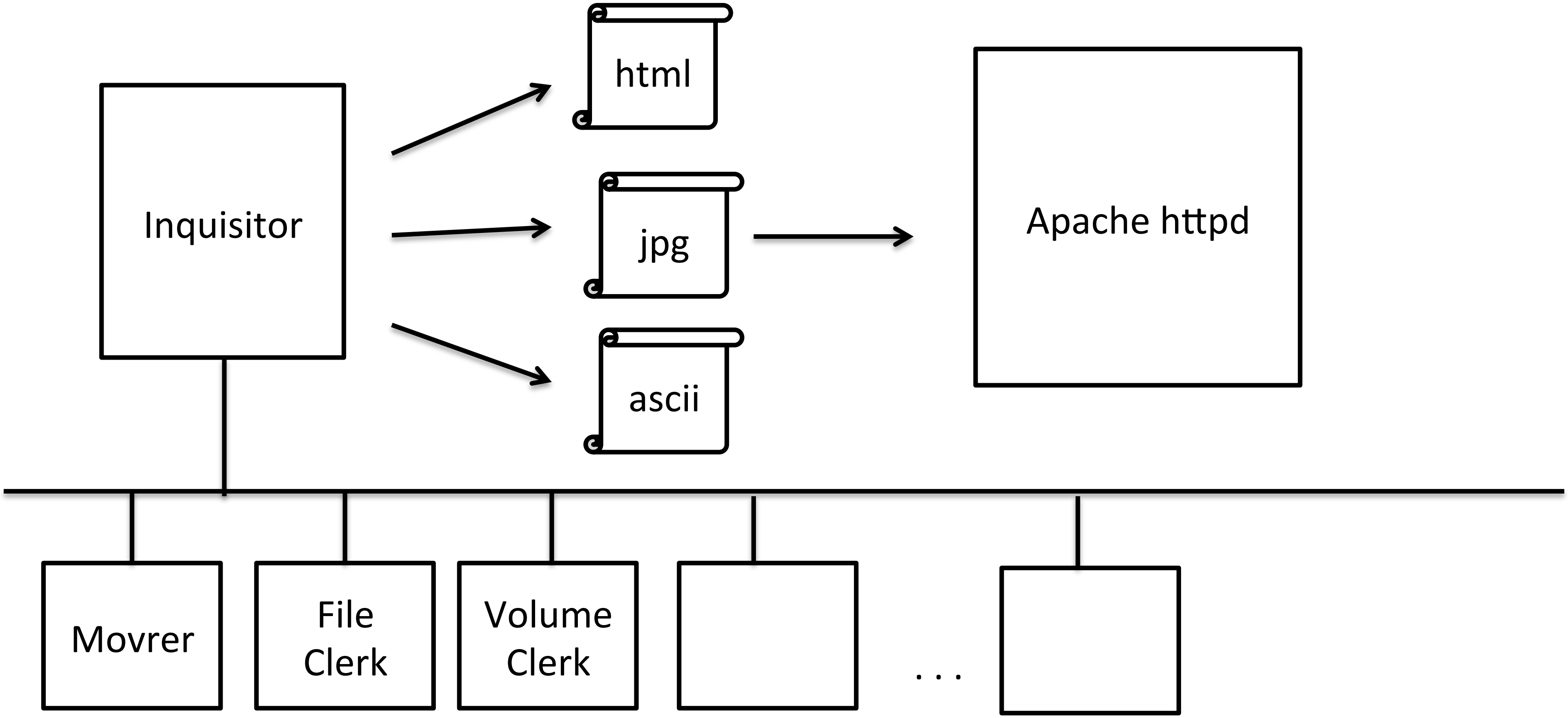
- Writes:
  - Using the volume eod_cookie, fast forward to end of volume. Try to verify that we are actually at the end of volume.
  - Write any wrappering information that precedes the data.
  - Fork a process that reads and crc's data from the encp and placing the data into a shared memory buffer.
  - Write data from the shared memory to the tape device.
  - Close the data port.
  - Write any wrappering information after the data.
  - Compute new eod_cookie and tell Volume Clerk that the volume is writable. Update remaining bytes as well.
  - Compute the file location cookie, and tell the bit File Clerk about the new file. Get a bit file ID in return.
  - Give the bit file ID to encp. Done.

FTT

- Fermi Tape Tools (ftt)
  - Table driven method to add new tape drives
  - Provides mt and raw SCSI access to tapes
  - Supports multiple types of serial media
  - Portable implementation
- Parts of FTT that Enstore uses:
  - Position media to correct file
    - By filemarks
    - By partitions
    - Read/write data
  - Write filemarks, both buffered and unbuffered
  - Get remaining capacity of tape
  - Drive statistics

Inquisitor

- Monitors system status and activity.



Alarm Server

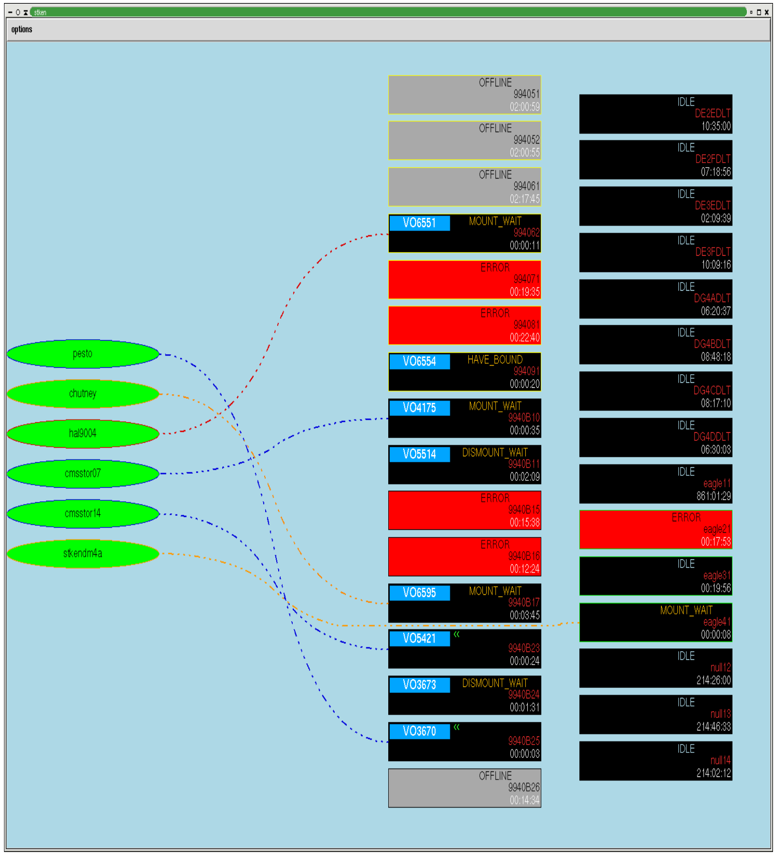
- Receives and maintains alarm messages send by Enstore components
- Interfaces to ServiceNow to generate incident tickets and pages so support personnel.

Inquisitor

http://www-stken.fnal.gov/enstore/status_enstore_system.html

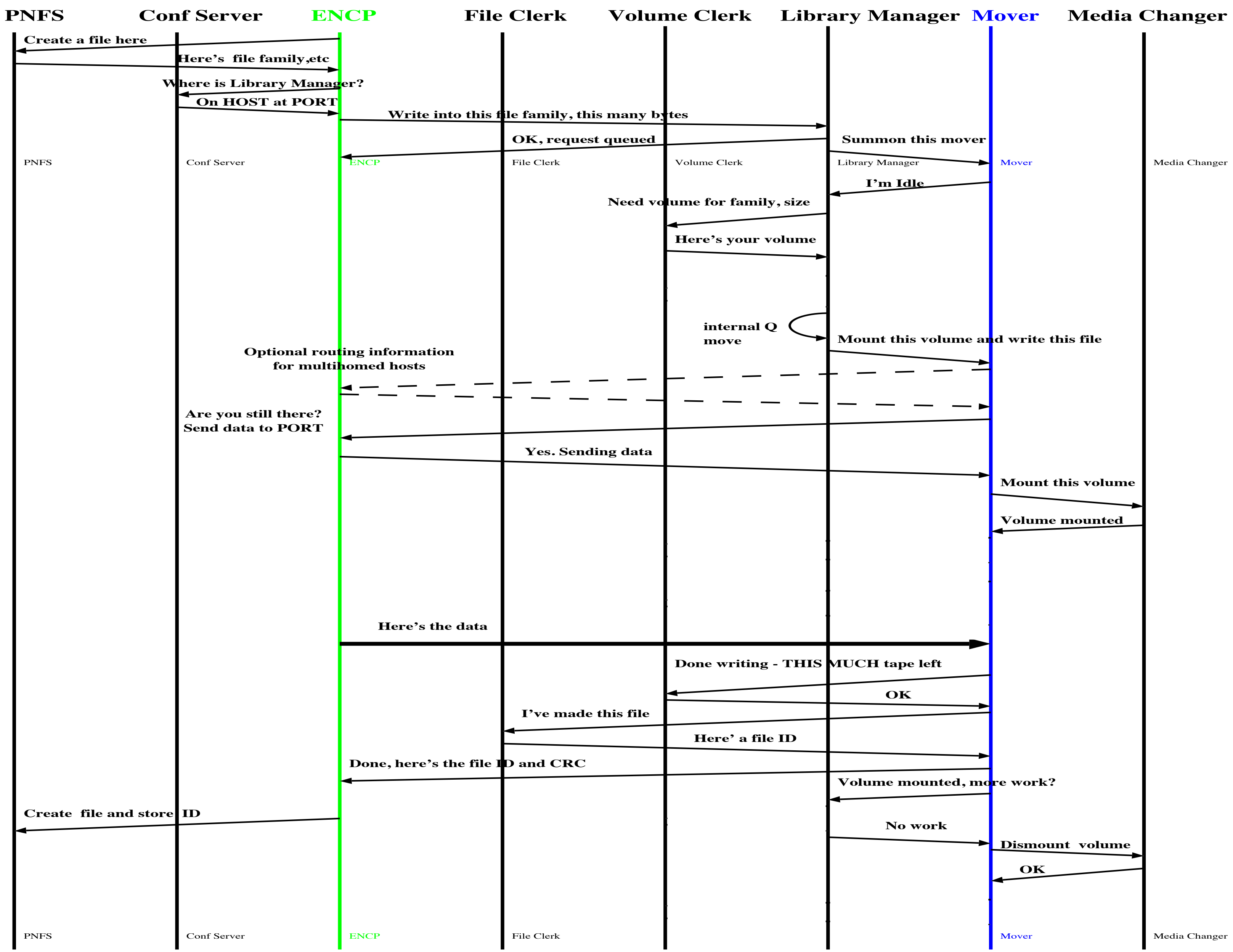
Monitoring

- Server States
- Resources (tape quota, drive utilization, drive hours)
- Data movement rates
- Volume usage (fill factor)
- Alarms

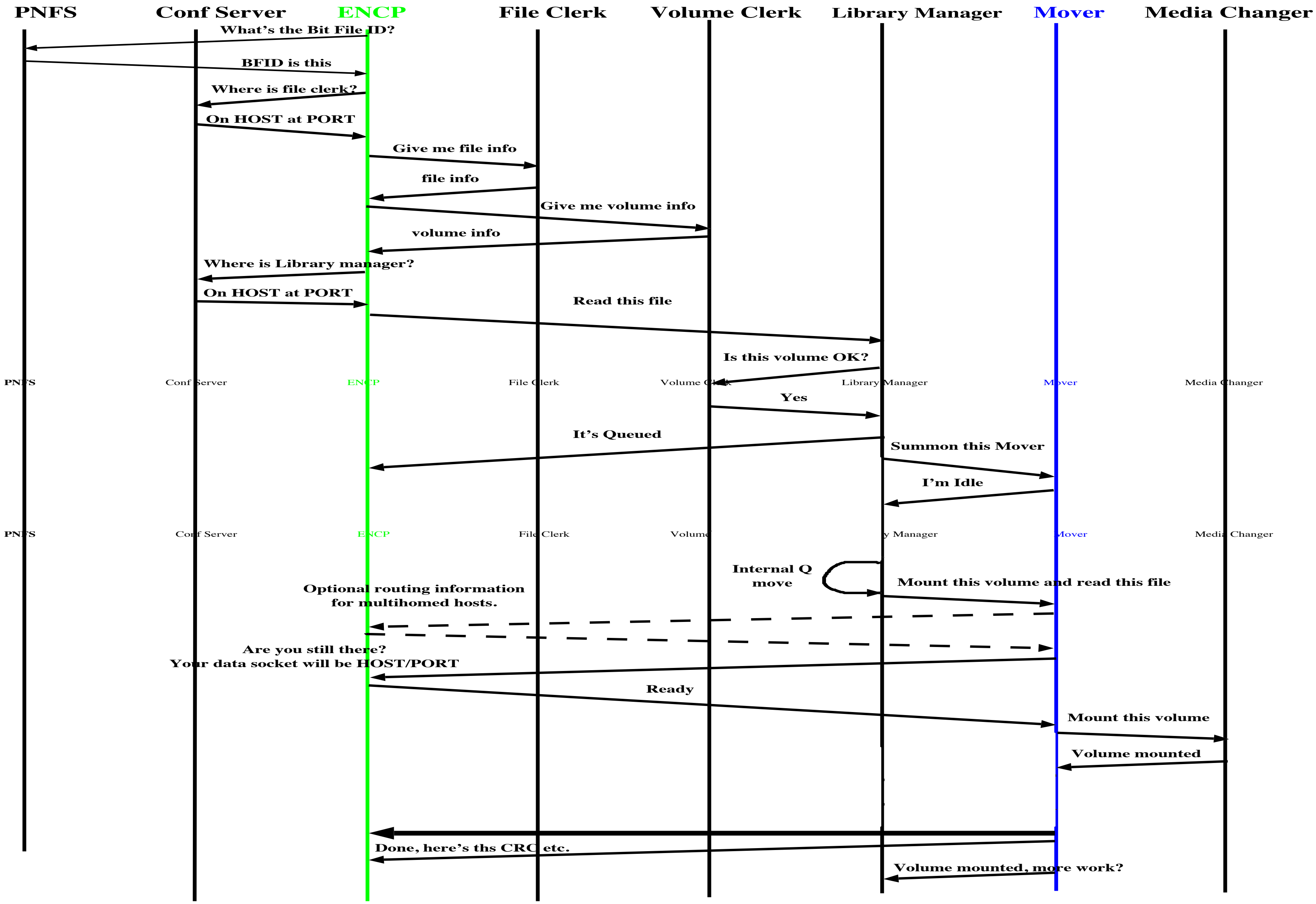


Event Relay

- Enstore services subscribe to receive notifications (events) via event relay
- Typically configuration reloads are relayed to Enstore servers and they implement logic of how to react to these changes.



File Write Communications



PNFS Conf Server ENCP File Clerk Volume Clerk Library Manager Mover Media Changer

What's the Bit File ID?

BFID is this

Where is file clerk?

On HOST at PORT

Give me file info

file info

Give me volume info

volume info

Where is Library manager?

On HOST at PORT

Read this file

Is this volume OK?

PNFS Conf Server ENCP File Clerk Volume Clerk Library Manager Mover Media Changer

Yes

It's Queued

Summon this Mover

I'm Idle

PNFS Conf Server ENCP File Clerk Volume y Manager Mover Media Changer

Internal Q
move

Mount this volume and read this file

Optional routing information
for multihomed hosts.

Are you still there?
Your data socket will be HOST/PORT

Ready

Mount this volume

Volume mounted

Done, here's ths CRC etc.

Volume mounted, more work?

File Read Communications

General Features

- End-to-end checksums of data transfers.
- Periodic checks of random volumes to detect data/ media corruption.
- Optimized access to user data by utilizing steering information stored in PNFS directory tags:
  - Enstore puts files on tape in the order the files were submitted.
  - Files are grouped on tape using file family and file family width scheme.
- Utilities to query tape content.
- Filesystem-like view of user stored data (thanks to PNFS/Chimera).
- Policy driven small file aggregation.

Resource Management

- Tape quotas based on storage group tags.
- Movers (and hence drives) can by dynamically assigned to match the conditions and priorities.
- Mount/dismount minimization: once tape is mounted, the queue is checked & all requests for volume are done before dismounting.
- Priority based request handling.
- Fair-share.
- Discipline (allow only certain number of clients from the same host to use a movers at the same time because of bandwidth considerations).

Features

- Tape import/export:
  - Volumes in Enstore are self describing allowing for easy tape export.
  - Conversely, provided metadata, it is relatively easy to import tape in Enstore
- Read-only and other flags
  - An CLI exist to set various flags on a tape (read-only, NOACCESS etc.)
- Open format
  - Tapes are self–describing with each file wrappered (cpio or cern wrapper)
  - Unix utilities exist to read wrappered files
- Data migration:
  - Enstore provides semi-automated procedure to migrate data to new media

Availability

- Unattended operation
- Automatic error reporting and alarming on serious errors or when available resources drop below threshold (number of available movers, tape quota approaching etc.)
- Robust against mover/drive failures because system is distributed.
- Single point of failure servers are simple and very robust.
- Extensive end–to–end error control and retry.

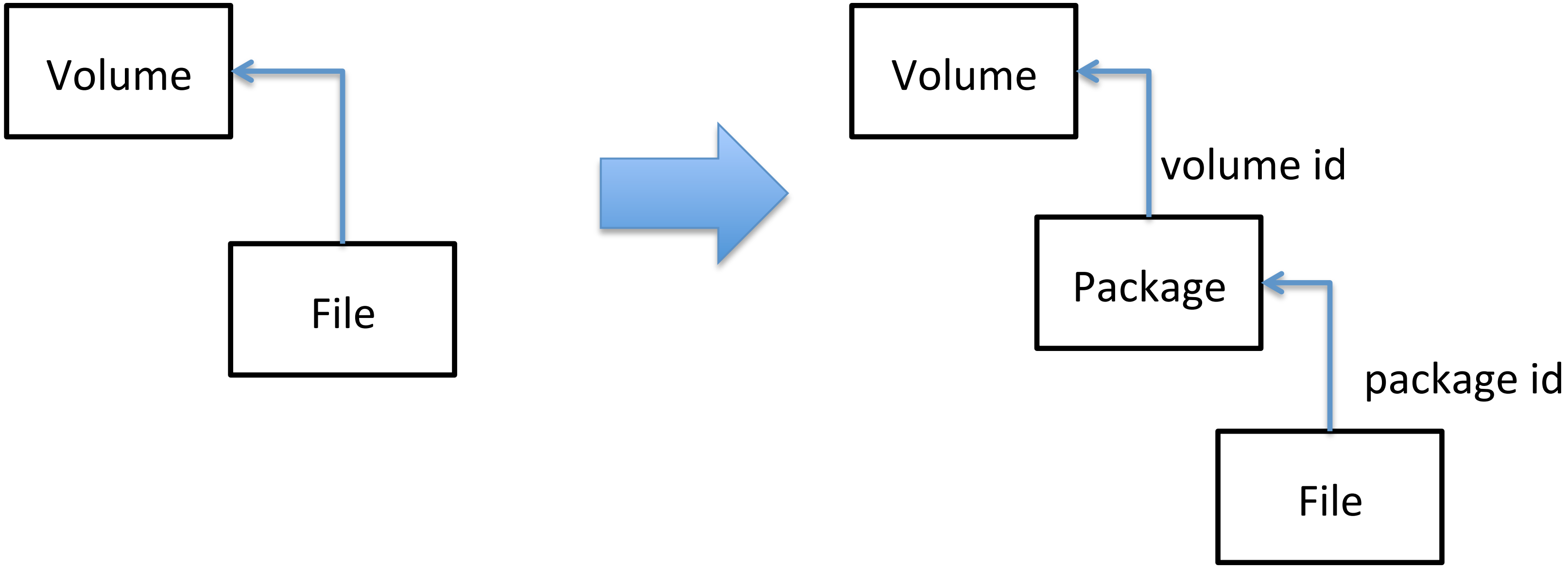
Small Files Problem

- Issues with small files:
  - Per file overhead to write a file mark (on writes)
  - Tape back-hitching : when data streaming to tape is interrupted (e.g. for the next file in queue) (on writes)
  - Mount latency (including load time) (on writes and reads)
  - Unload (rewind time) (on writes and reads)
  - Seek time (on writes and reads)

SFA: Small File Aggregation

- Enstore automatically aggregates small files into larger containers (using tar utility)
  - Implemented by utilizing so called disk mover
- Transparent to user - packing and un-packing at server side, users sees only small files
- Preserve end-to-end checksums
- Assume custodial ownership of files in SFA disk cache
- Per customer small files policies (that define what small file is and how many files per package)

Small File Aggregation



Policy driven aggregation

- Enstore aggregates files by storage group and file family.
- Policy is expressed using:
  - Original library
  - Resulting disk library
  - Storage group
  - File family
  - File family wrapper
  - Minimum file size
  - Maximum # of files per package
  - Maximum time to wait before files are packaged and written to tape

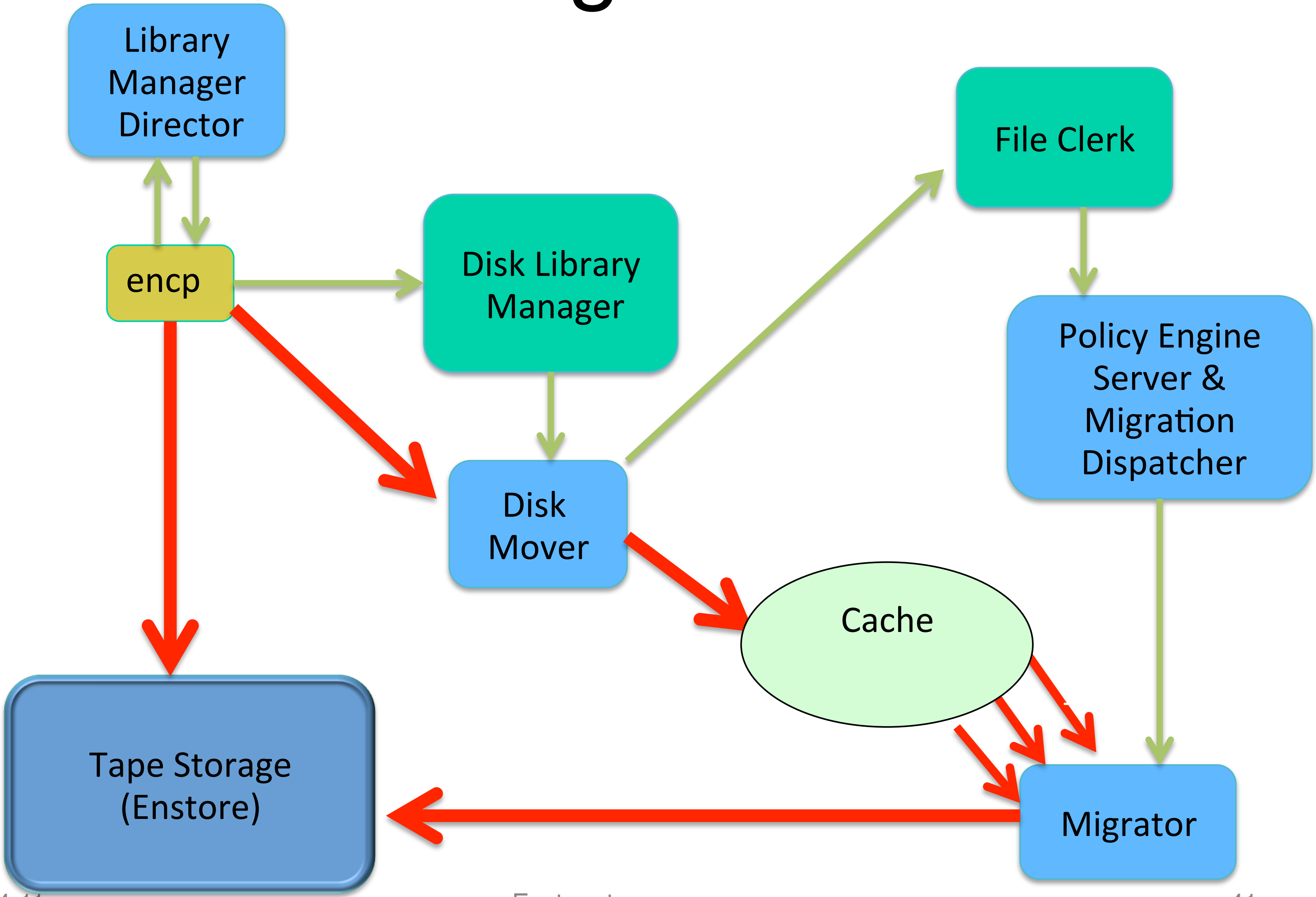
SFA Components

- Policy engine: stores and appies file aggregation policies:
  - receives event from File Clerk that file has arrived to disk cache or needs to be staged into disk cache
  - Maintains 3 lists:
    - Archive – files to be written to tape.
    - Stage – files to be staged from tape
    - Purge – files to be purged form disk cache
- Migration Dispatcher : receives file lists from Policy Engine and dispatches them migrators.
- 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.
- Library Manager re-Director:
  - receives write request from encp and determines if data needs to be send to disk mover instead.
- Disk movers: transfer files to SFA disk
- Communication layer between File Clerk, Dispatcher and Migrators are implemented using Apache QPID AMQP messaging system.

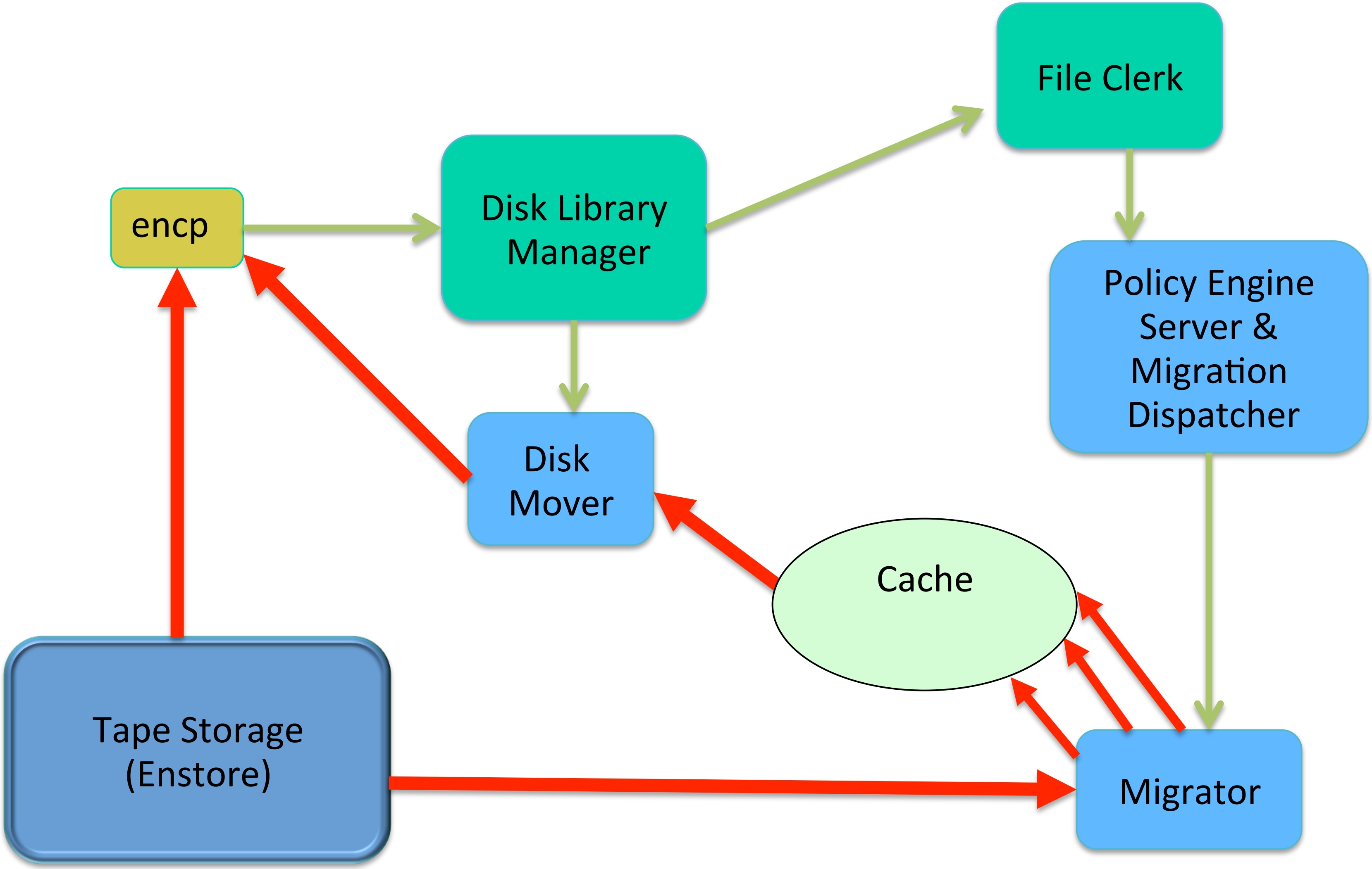
SFA package

- SFA package is self described container (tar file)
- Each SFA package contains a file manifest that includes:
  - file path in cache
  - file name in namespace
  - file checksum
- Package files are written into Enstore and places into a separate (user invisible) directory.

Writing Files



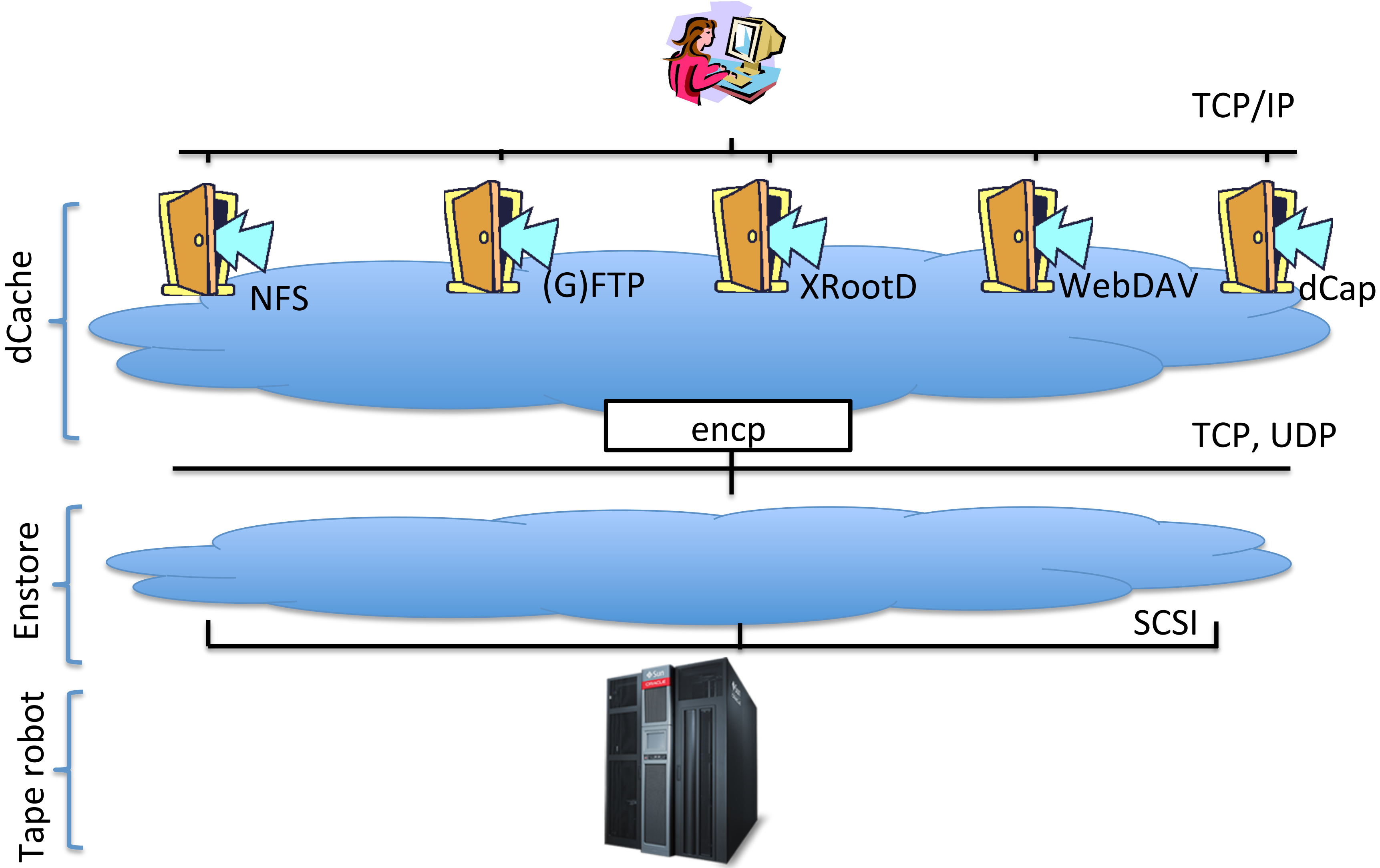
Reading Files.



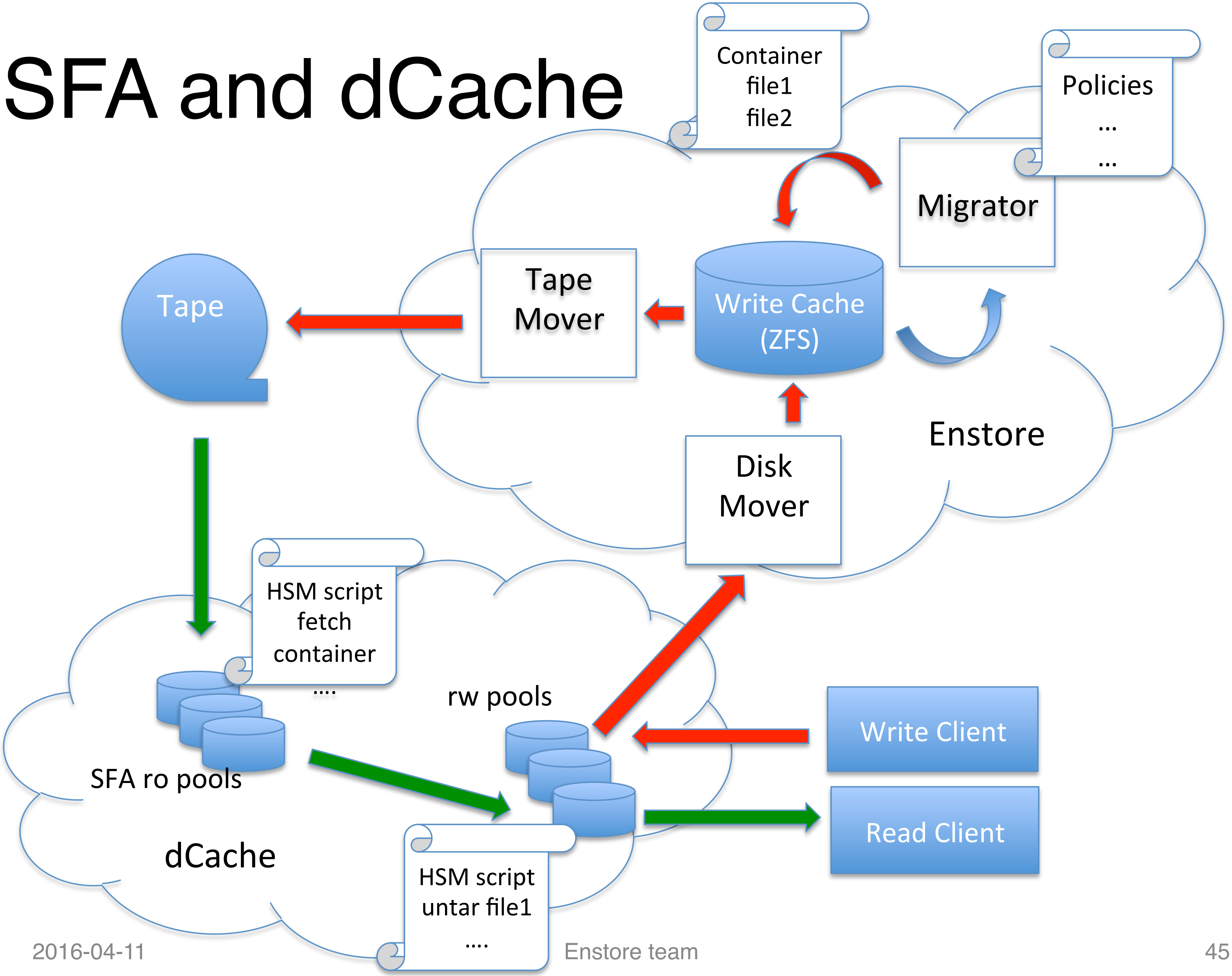
Data Integrity in Cache

- ZFS is used as Enstore fata cache for reliability (Nexenta Appliance, RAIDZ2).
- Selective checksum verification before writing package files to tape.
- Track files in transition in database. Alarms if there are stale, left over files.

dCache and Enstore



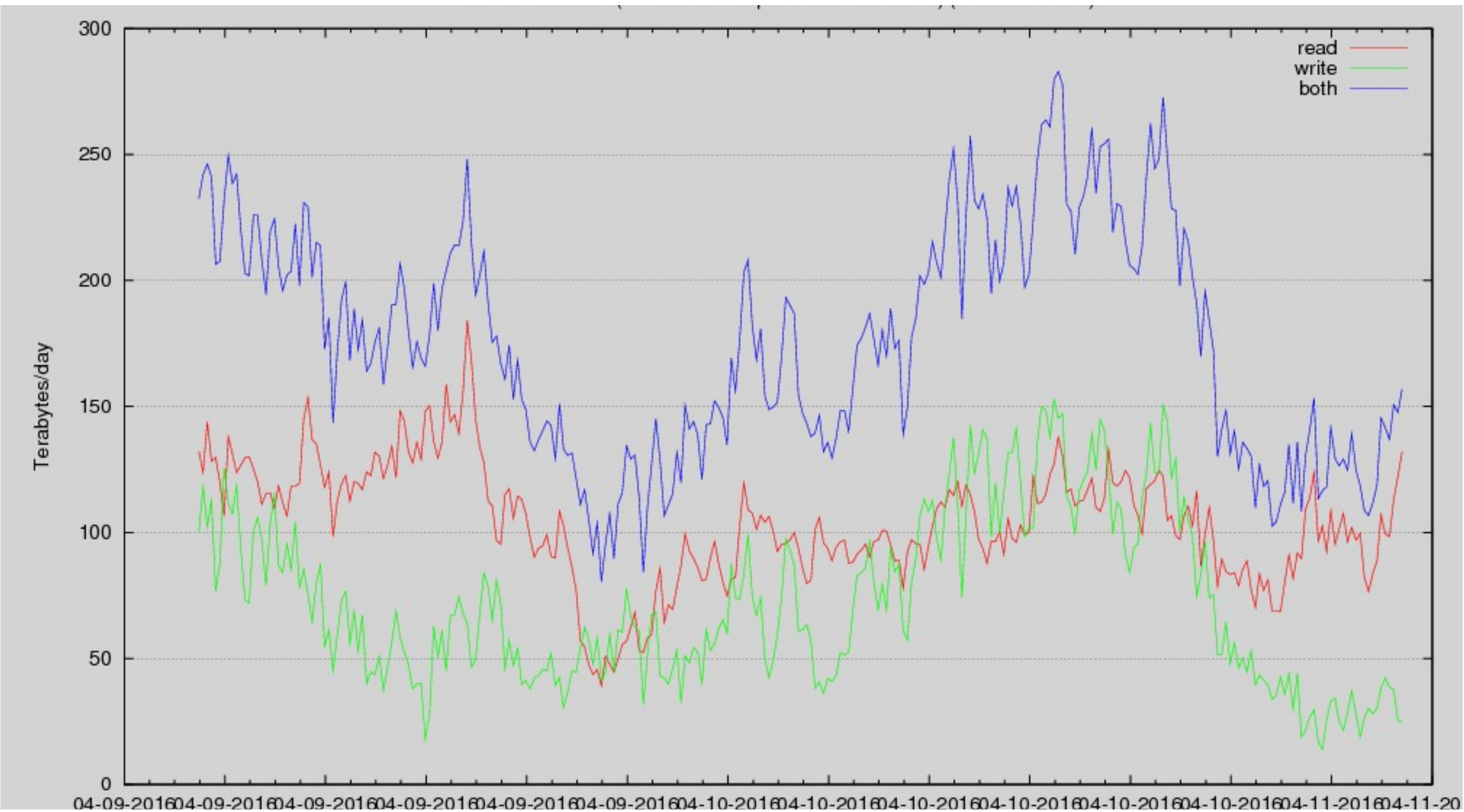
SFA and dCache



Enstore Deployments

- Enstore is currently used:
  - Fermilab (3 instances, total capacity 100 PB)
  - PIC
  - 2 Russian Tier 1 sites

A performance plot (Fermilab)



Concluding remarks

- Enstore meets and exceeds data storage requirements of Fermilab hosted (and collaborated) experiments.
- Scales easily to higher throughput and capacity
- Robust and fault tolerant
- Uses commodity hardware and software (besides robotic livbrary)
- Check current status and details here:
  - http://www-ccf.fnal.gov/enstore/