

# File Aggregation in Enstore (Small Files)

<https://cdcvs.fnal.gov/redmine/projects/show/fileaggregation>

**Alexander Moibenko**

# Problem

- Writing or reading a tape mark (EOF) at the end of a file takes about 3 seconds. Writing or reading a full tape of continuous data takes just under 2 hours at top speed.
  - Thus, a tape full of 360 MB files would take twice as long —4 hours.
  - So files ought to be much larger; a few GB is good.
- And as tape capacity and speeds grow, the minimum desirable file size increases also. “Eventually, any file becomes small.”

# Project Goals

- 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

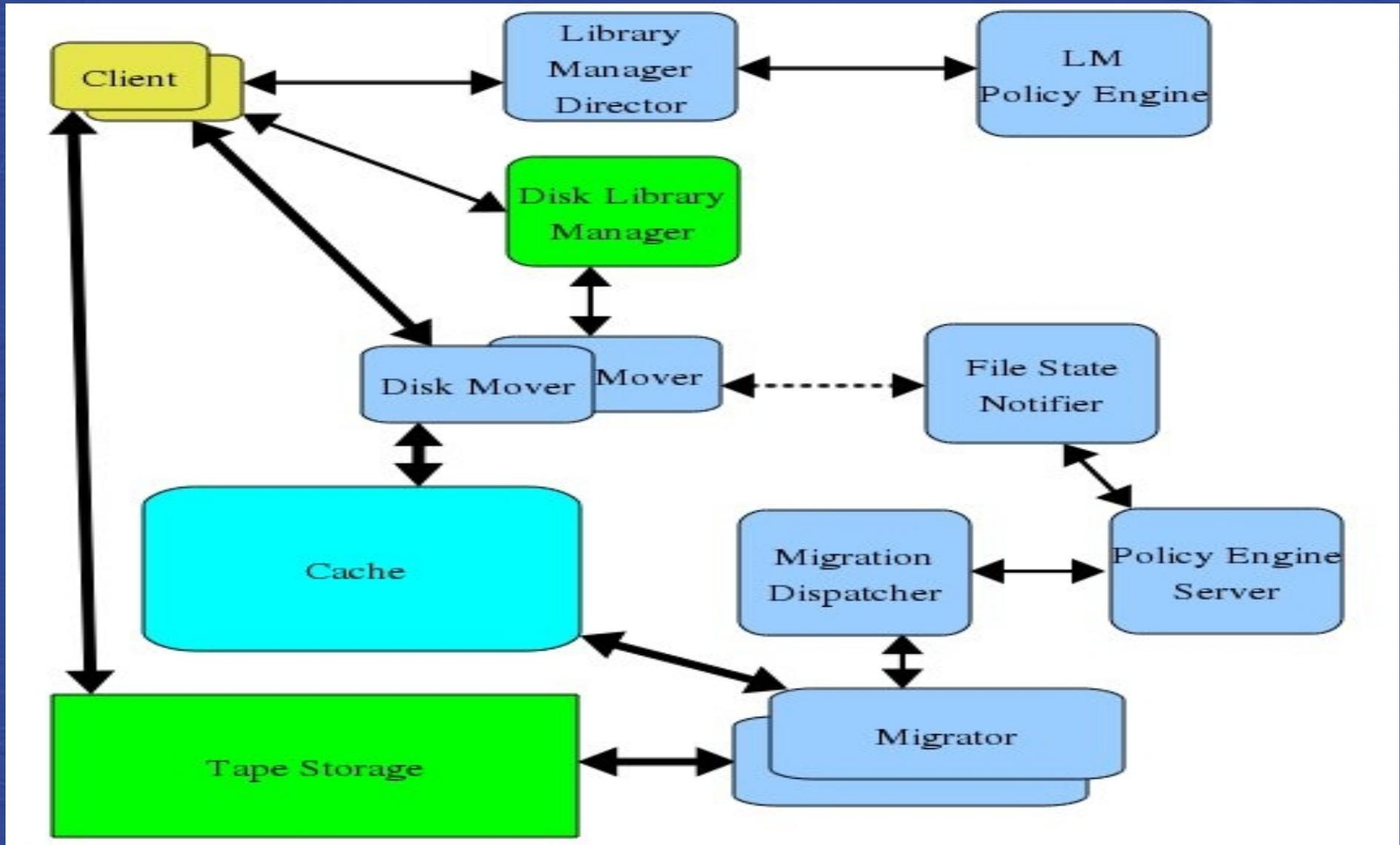
# What files to aggregate ?

- 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 family and directory trees is good to start with.

# 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
  - **enstore disk movers can access any unpacked file in cache; tape movers can access any container.**
- Compatibility with name-space (pnfs now, chimera later) and current client tools (encp).

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



# Description of components

- Library Manager Director – determines whether to send data to tape directly or to cache for aggregation.
- Library Manager Policy Engine – policy engine for selection of the LM to send encp request to.
- Disk Library Manager – standard enstore library manager configured for disk movers
- Disk Movers – modified enstore disk movers
- File State Notifier – notifies Policy Engine Server about changed state of file (created, written to cache, etc)
- Manager Policy Engine – policy engine for selection of an LM to direct encp request to.

# Description of components (cont.)

- Policy Engine Server receives event from Notifier specifying that the **new** file arrived into cache or the file written to tape needs to get staged from it. PE Server 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.
- Migrators aggregate data in cache and write containers to tape. They also 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.

# Policy Engine

- Sun's **Intelligent Event Processing** engine (IEP), part of the Glassfish Application Server. It has GUI to set and edit rules. Open Source - Event Driven Application Server
  - operate on events in time window
  - process, split, combine event streams
  - store data in DB or call execution of the process at the end of processing chain
  - API to provide input data feed and output sink
  - Recovery mechanism can be provided outside of EDAS if not provided internally
- Communication Provider: AMQP. Has interface to all used languages (Python – enstore, Java, C, C++)

# File Aggregation

- When PE has collected “enough” files it triggers aggregation of files into container.
- Ultimate disaster recovery: tape in hand but name-space is completely lost. Information will be included in each container to enable restoration of files with the names as originally stored by the user.
  - Similar capability already exists for non-aggregated files

# Scope of The Project

- Enstore Cache
  - integrate at least one multi-client file system.
- Policy Engine
  - Develop event feed, sink, rules.
- Migration/Purge Dispatcher
- Select and integrate reliable messaging system (AMQP)
- Migration Modules integrated with Enstore

# Status of The Project

- HLD for the whole system – in progress
- HLD for components – in progress
- Enstore Cache
  - Still selecting (consider ZFS as base)
- Policy Engine
  - IEP installed, tested, tested proof of concept
- AMQP installed, tested, tested proof of concept

# Status of The Project (cont)

- Event Notifier – in HLD
- Library Manager Director – in HLD
- Event Notifier – in HLD
- Migration/Purge Dispatcher - in HLD
- Migrator – in HLD
- Disk Mover – 80% done