

# Enstore with Chimera namespace provider

**Dmitry Litvintsev, Alexander Moibenko, Gene Oleynik, Michael Zalokar**

Fermi National Accelerator Laboratory, P.O. Box 500, Batavia, IL 60510-5011, USA

E-mail: {litvinse,moibenko,oleynik,zalokar}@fnal.gov

**Abstract.** Enstore is a mass storage system developed by Fermilab that provides distributed access and management of the data stored on tapes. It uses namespace service, pnfs, developed by DESY to provide filesystem-like view of the stored data. Pnfs is a legacy product and is being replaced by a new implementation, called Chimera, which is also developed by DESY. The Chimera namespace offers multiple advantages over the pnfs in terms of performance and functionality. Enstore client component, encp, has been modified to work with Chimera or any other namespace provider. We performed high load end-to-end acceptance test of Enstore with Chimera namespace. This paper describes modifications to Enstore, test procedure and results of the acceptance testing.

## 1. Introduction

Enstore is a Mass Storage System developed and operated by Fermilab. It provides seamless access to the data stored in permanent media by client applications distributed across IP networks [1].

The system provides hierarchical view of stored files via NFS mounted file system, served by namespace provider, the PNFS (a Perfectly Normal File System), developed by DESY [2]. The PNFS has been designed and implemented in the end of 90s and has reached its limitations. In the run up to LHC data challenge DESY has developed next generation namespace provider called Chimera [3] that meets scalability and performance requirements of the LHC experiments.

This article describes recent migration of Enstore system from PNFS to Chimera namespace provider.

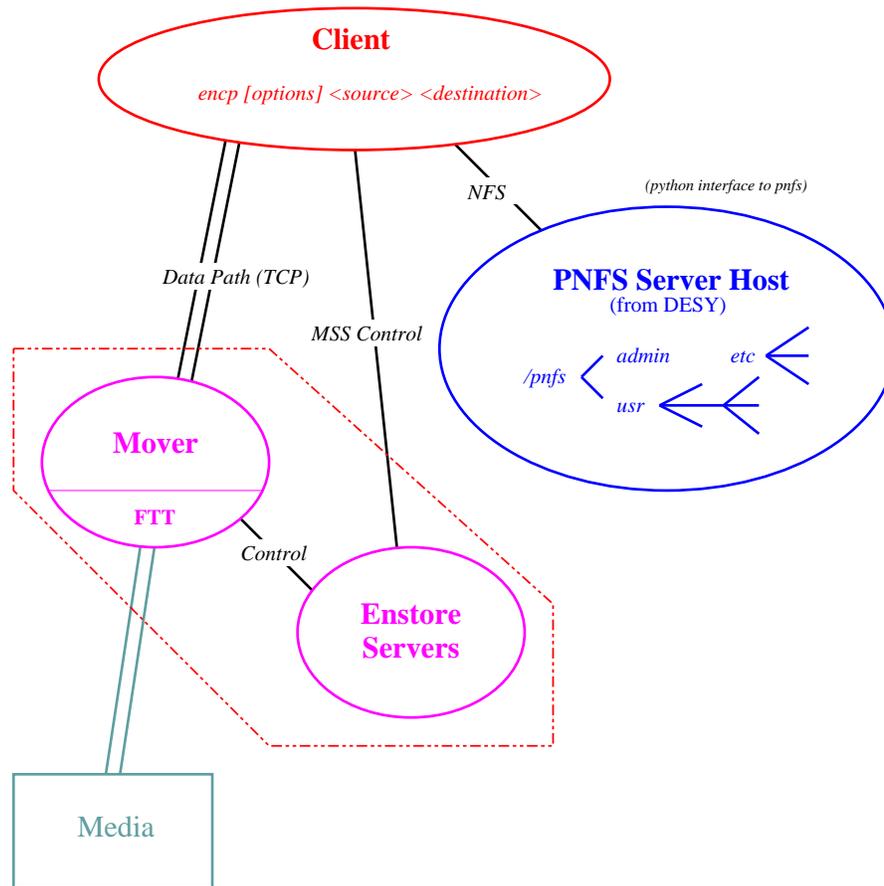
## 2. Enstore

The Enstore project was started in preparation for Tevatron collider Run II data taking in 1998. It has delivered highly reliable, scalable, fault tolerant, and flexible storage system that satisfied the needs of two major Run II experiments, D0 and CDF, as well as host of smaller experiments. It is also used for general mass storage needs like system and user data backups. Currently Enstore is the primary storage system at Fermilab. It holds about 40 TiBs of data, 50 million files. It can be operated stand-alone or can be used with dCache [4] or SAM [5] systems, as a disk caching front-ends for optimal tape I/O.

Enstore is a client-server application implemented mostly in Python with inclusion of C codes for modules where high performance is required.

The server is a multicomponent structure of distributed servers that provides:

## Enstore at Fermilab



**Figure 1.** Schematics of encp client interaction with Enstore system.

- hierarchical meta-data view of user files stored on tape, presented to client as if it were a Unix file system;
- management of user files (e.g. renaming, removal or recovery);
- distributed access to tape drives;
- interface to robotic tape libraries;
- resource management of available tape drives;
- flow control based on meta-data associated with files.
- tape allocation accounting per storage group, file family, media type;
- self-monitoring, error-reporting and alarm services;
- periodic data integrity checks;
- web-interface;
- file migration from older media type to new media type (e.g. LTO3 → T10KC);
- small file aggregation.

Enstore client, `encp`, provides “cp”-like functionality to retrieve/store files from/to tapes, end-to-end data integrity verification and meta-data manipulation.

Enstore components run on distributed commodity hardware connected via IP. They communicate using inter-process communications based on UDP. Great care has been taken to ensure that the system will function well under extreme load conditions. By design, there is no preset limit on the number of concurrent user computers or on the number of physical media libraries or drives. The system is only limited by the availability of physical resources.

The system is designed for distributed and peer-to-peer reliability. Each request originating from the encp is branded with a unique ID. Encp retries under well-defined circumstances, issuing an equivalent request with a new unique ID. The system can instruct encp to retry if it needs to back out of an operation.

A simplified interaction between encp and Enstore system is presented in figure 1. The file hierarchy is presented to the client via NFS mounted filesystem served by namespace provider, PNFS, a component developed by DESY [2]. A client command to write a file into Enstore would look like

```
encp [options] <source> <destination>
```

where <source> is file name and <destination> is full path name or a directory name in PNFS. Encp checks destination for write permissions using PNFS interface, extracts flow control meta-data stored in special tag files located in the destination directory. These meta-data contain name of the storage group, library name, file family, file family width and file family wrapper. Then it communicates with configuration server to discover address of corresponding library manager (LM). Encp then sends information about source file, and flow control information to LM. The latter queries volume clerk (VC) for available tape that satisfies selection based on file size, storage group name, file family name and file family wrapper name. If tape is found, the LM schedules the request in the request queue and will process the request as soon as there is mover available for the tape. Once the mover is found, LM sends mover IP address to encp. Encp established TCP connection for data transfer from client host directly to the mover. Mover then writes data to tape. After transfer is finished, encp updates file meta-data in the destination, and sets additional meta-data in special layer files. These data contain unique file identifier (BFID), volume label, location of file on the tape, size and crc values, file id (pnfsid), file name and tape drive information.

Reading of files from Enstore involves checking PNFS for read permissions, extracting file layer information and passing this information to the corresponding LM. LM schedules request in its queue. Then request is scheduled to run, the available mover mounts the tape, if not already mounted, encp gets mover IP address and data is transferred via TCP to the client host.

### 3. Namespace provider

Namespace provider is software component that provides association between file names and data distributed across multiple locations and on the variety of media.

It supplies:

- Unique file ID independent of file name.
- Path to ID and reverse ID to path mapping.
- File meta-data storage. Arbitrary meta-data may be associated with the file, in particular storage system specific information like tape volume name, offset etc.
- Directory tags inherited by sub directories to store extra meta-data associated with storage system like storage group, file family and so on.
- Callbacks on filesystem events.
- Worm holes. A convenient feature: files that are not shown in directory listing, but are available in all directories. Useful for distributing configuration files.
- Additional channel for the client to access meta-data.

From the outset of Enstore project the PNFS was chosen as namespace provider service. PNFS is a NFS server on top of a database. PNFS allows all NFSv2 operations except actual data I/O which is performed by storage system specific utilities. It is implemented as NFS daemon which runs in user space and communicates with db-server through shared memory block. The db-server simulates a filesystem on top of a database (PostgreSQL or GDBM back-ends are supported).

While PNFS has been used successfully during Tevatron Collider Run II serving about 10M files each for D0 and CDF, the following limitations have been identified:

- Maximum file size is 2 GB due to NFSv2 specification.
- Meta-data access only through NFS server making it a bottleneck. Heavy access to meta-data by storage system adversely impacts NFS operations.
- Meta-data are stored as (key,value) pair with key being unique file id and value being BLOB. This impacts performance (e.g. for ls operations) and does not allow for selective meta-data querying.
- No ACL support.
- No security.

As LHC experiment anticipated manifold increase in data rates, requiring higher throughput rates and better scalability in terms of performance and capacity of storage systems with additional requirements of ACL support and authentication it became clear that a new namespace provider was needed.

In 2004 DESY has developed next generation name space provider called Chimera. Chimera is a filesystem simulator on top of a RDBMS. It has been designed for performance and provides well defined API for namespace operations, meta-data manipulations and administration. Storage system accesses the filesystem directly via the API bypassing the NFS interface.

Chimera is written in Java and therefore is highly portable. It uses JDBC interface for database access without data base specific bindings. This allows to deploy Chimera on top of any database that has JDBC driver. A relational data base, depending on implementation, provides benefits of query language, consistency checks, triggers and stored procedures and backup and recovery tools.

The data base schema is designed to allow meta-data query isolation by providing a table for each type of meta-data, so that queries for one type of meta-data do not impact queries for others.

Chimera supports ACLs via pluggable permission handlers with standard Unix permission handler as default and GSS authentication.

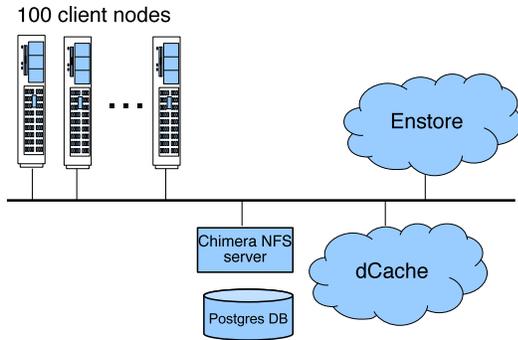
Most importantly Chimera implements NFSv4.1 protocol for namespace and data file access which allows parallel POSIX I/O on distributed data. NFSv4.1 is production ready starting from dCache version 1.9.12.

#### 4. Chimera and Enstore

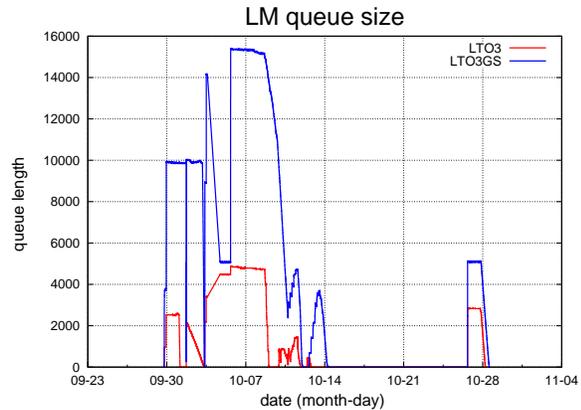
Improvements in performance and scalability demonstrated by Chimera, its extended feature set and its NFSV4.1 support prompted Enstore developers to evaluate Chimera namespace provider for Enstore system. Technically, the only namespace aware component of Enstore is Enstore client, encp. Enstore used extended filesystem interface to handle file meta-data in PNFS. This interface has been made into a base class with two concrete subclasses `Pnfs` or `Chimera`. A specific interface is instantiated at the run time based on system specific tags available in NFS served filesystems, using factory method pattern. Chimera support is available starting with encp version v3\_10e.

## 5. Acceptance Test

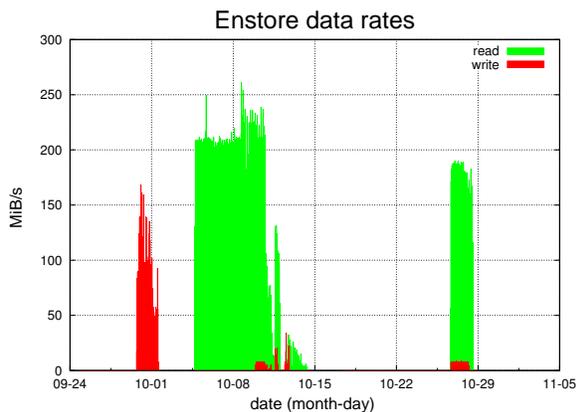
The goal of the acceptance test was to determine that Enstore with Chimera namespace provider was stable under heavy load in close to production environment. Tests involved performing encp transfers directly to/from Enstore as well as dcp transfers via dCache for end-to-end testing.



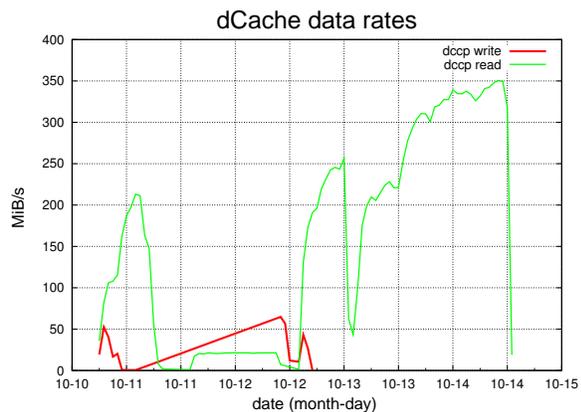
**Figure 2.** Enstore with Chimera acceptance test setup.



**Figure 3.** Library Manager queues during acceptance test. Two LMs were involved in the test.



**Figure 4.** Enstore data rates.



**Figure 5.** dCache data rates.

The acceptance test setup is shown in figure 2, it consisted of:

- 100 client nodes.
- Fully functional Enstore instance attached to SL8500 robotic library. The system had:
  - Separate namespace node to run Chimera, underlying PostgreSQL data base and dCache PnfsManager.
  - Separate node for Enstore internal databases and database servers.
  - Two LTO3 library managers (LMs) running on separated nodes.
  - 5 LTO3 movers.
- dCache system attached to Enstore :
  - 10 pool nodes. One dcap door per node.
  - Separate head node.

- dCache version 1.9.5-28 (support for Enstore and Chimera is available in dCache starting version 1.9.5-24).

Each client node ran a torture test harness in a loop submitting about 130 simultaneous encp/dccp file transfers. Files were generated to contain random binary data with the average size of 2 GiB. First, write tests were performed until all tapes were used up. Then the read tests commenced. A portion of the tapes was later recycled and dccp write tests were performed for smaller file sizes.

The load, created on the system, is expressed in terms of LM queue sizes as shown in plot in figure 3. As can be seen LM sustained up to 15K requests in the queue without request drops. This far exceeds typical queue sizes on for instance CMS T1 production system where the peak load is about 5K-10K requests.

**Table 1.** Enstore Chimera tests summary

transfers	TiBs	Enstore/dCache	r/w
83198	161.5	Enstore	read
23749	25.4	Enstore	write
1215	2.3	dCache	read
9869	1.4	dCache	write

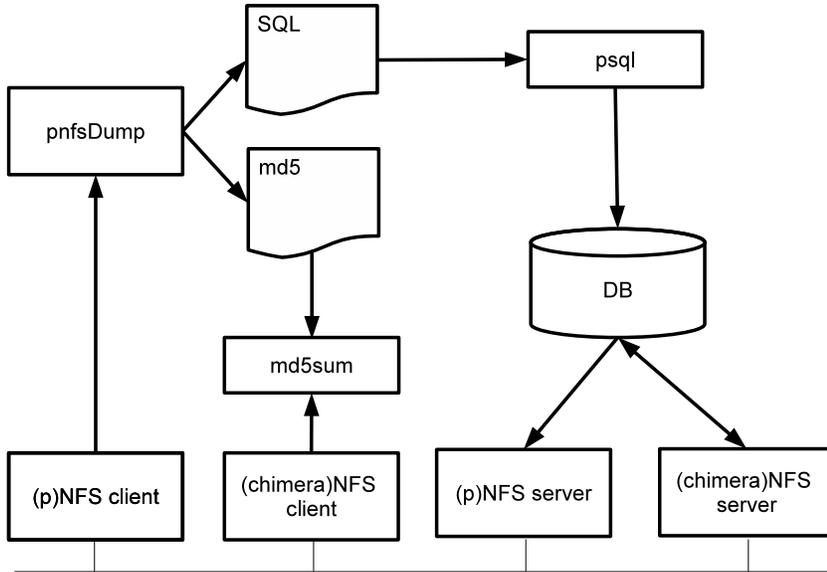
The data transfer rates in MiB/s seen by Enstore and dCache system are shown in figures 4 and 5 correspondingly. Enstore rates were limited by the number of available movers while dCache rates were limited by network and aggregate disk I/O. Table 1 gives summary of transfers. No errors have been seen. Chimera performed flawlessly.

## 6. PNFS → Chimera migration

Fermilab operates three Enstore instances – public Enstore, D0 Enstore and CDF Enstore. CMS T1 uses public Enstore, but employs separate PNFS namespace. Therefore there are four PNFS systems.

PNFS to Chimera migration procedure is outlined in figure 6. It involves several steps that have been wrapped into all-or-nothing python script and migration procedure has been tested several times on copies of PNFS databases taken from all three production Enstore systems.

The migration starts with setting up chimera PostgreSQL data base, stopping PNFS, starting chimera server, creating top level directories, and mapping their “chimera” IDs to corresponding PNFS IDs. Then chimera server is stopped, PNFS server is started, and `pnfsDump` script is run. After `pnfsDump` completes it produces three sets of files for each top directory: list of SQL statements to be injected into Chimera data base to create the directory tree structure underneath, list of all files in the directory tree and the file containing list of md5 checksums for each file. At the next step the migration script takes SQL statement lists and injects them into Chimera data base using `psql` client. After this stage is completed, the file meta-data stored in layer 4 is converted into Chimera location information and companion information, which contains list of dCache pools where the files are located, is also converted to Chimera location information. Then PNFS server is stopped and Chimera server is started. At this point the system is available. In the background the script executes `md5sum` on all files in Chimera filesystem and comparing results against md5 checksum lists produced by `pnfsDump`. The time consuming part is `pnfsDump` and SQL injection. To minimize effects of the downtime the PNFS



**Figure 6.** Diagram of PNFS to Chimera migration procedure.

server remains open for users in read-only mode (by setting PNFS data base connection user to read-only role).

We performed migration of public Enstore system PNFS to Chimera on 02/23/12 and simultaneously migrated PNFS to Chimera on D0 and CDF systems on 05/01/12. No issues were encountered during migration or in subsequent running of production Enstore instances with Chimera back-end. The migration has been smooth and transition was absolutely transparent to the users. No changes on these user ends were necessary beyond upgrading to encp version v3\_10e which has been done well ahead of actual migration owing to the fact that encp v3\_10e supports both PNFS and Chimera simultaneously.

**Table 2.** PNFS to Chimera migration timing

action	public Enstore	CDF	D0
pnfsDump	6h52m	5h32m	12h53m
SQL Import	9h47m	5h38m	8h28m
layer4 to location info	1h08m	0h26m	2h00m
companion import	0h17m	0h20m	N/A
Total	18h4m	11h56m	23h21m
number of files	15317841	8523362	14001375
number of dirs	323053	686947	19468

Timing of various stages of PNFS to Chimera migration is summarized in table 2. CDF Enstore system was the fastest to migrate while D0 was the slowest. Comparing number of files and number of directories presented in table 2, it can be concluded that it was due to larger

number of files per directories in D0 system. Companion import was not done for D0 because D0 uses SAM front-end rather than dCache.

## 7. Conclusion

After almost a decade of using PNFS namespace provider Fermilab production Enstore systems have switched to a new Chimera namespace provider. Migration from PNFS to Chimera went smoothly and we are gaining operational experience with the new system. So far no problems have been observed. In the future we plan to fully explore capabilities of NFSv4.1 interface which is supported by Chimera to provide POSIX I/O on stored data to the Fermilab user community.

## References

- [1] Bakken J, Berman E, Huang C, Moibenko A, Petravick D, Rechenmacher R, Ruthmansdorfer K “Enstore Technical Design Document” URL <http://www.dcache.org>
- [2] Fuhrmann P, “A Perfectly Normal File System PNFS” <http://www-pnfs.desy.de/>
- [3] Gasthuber M, Mkrtchyan T, Fuhrmann P 2005 “Chimera - a new, fast, extensible and Grid enabled namespace service” *Proc. Conf. Computing in high energy physics and nuclear physics September 27-October 1 2004 (Interlaken)* p 1180
- [4] dCache URL <http://www.dcache.org>
- [5] Lueking L *et al.* 1998 “The Sequential access model for Run II data management” *Proc. Conf. Computing in high energy physics and nuclear physics September 1-4 1998 (Chicago) (FERMILAB-CONF-98-382-E)*