Enstore site installation and configuration.

Alexander Moibenko

Table of Contents

History

Action	Performed by	Date and Time	Comment
Publish	Alexander Moibenko	06/23/2011 09:40:00	Modified dcache part
Retract	Alexander Moibenko	2009-11-11 10:14	No comments.
Publish	Alexander Moibenko	2007-10-05 08:44	No comments.

1. Product description

Enstore is a multi-Petabyte scale tape based Mass Storage System (MSS) for High Energy Physics (HEP) Experiments and other scientific endeavors. It has been designed to permit us to scale to multiple petabytes of storage capacity, manage tens of terabytes per day in data transfers, support hundreds of users, and maintain data integrity. Enstore can be used for data storage needs of any scale, for different kinds of enterprises. The Enstore architecture allows easy addition and replacement of hardware and software components. Enstore has the following major components

- Servers
  - Configuration server - maintain system configuration information and present it to the rest of the system components.
  - Volume clerk - maintain volume part of the enstore database.
  - File clerk - maintain file part of the enstore database.
  - Info Server - provides read only functionality of File and Volume Clerks
  - Multiple, distributed library managers - provide queuing, optimization and distribution of user requests to assigned movers.
  - Multiple, distributed movers - transfer data between user computers and storage devices (tape drive, disk).
  - Media changers - mount/dismount requested media in the tape devices.
  - Log server - log messages from the Enstore components.
  - Alarm Server - generate alarms upon requests from Enstore components.
  - Accounting Server - provides accounting of data transfers. Uses accounting database to store accounting information.
  - Drivestat server - records tape drives usage statistics into drivestat database.
  - Event relay - relays events between enstore components.
  - Inquisitor – monitors enstore work and presents information (in particular on web pages)
  - Ratekeeper – collects information about data transfer rates

- namespace - implemented by the PNFS package from DESY.
- encp - a program used to copy files to and from tape libraries.
- monitoring system (includes the Inquisitor)
- administration tools

The description of the system and other related information can be found here or in the doc subdirecrtory of enstore product. If you are new to enstore please read this document to start with.

2. Licensing

Enstore (BSD like):

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the FERMI NATIONAL ACCELERATOR LABORATORY, nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

PNFS:

Enstore uses a PNFS virtual file system package that implements the Enstore namespace. It was written at DESY. The licensing information and code can be obtained at the dcache site. Installation instructions can be found here: http://www-pnfs.desy.de/gettingStarted.html

PNFS must be installed only on the node that will run pnfs server. PNFS mount point(s) must be exported

to any node transferring data to or from enstore. Some instructions can be found here: http://www-isd.fnal.gov/ISA/pnfs.html

3. Hardware requirements

3.1 Hosts

There are no strict requirements for hosts. The general requirements are:

1. dual CPU Intel processor (3.0GHZ or better),
2. 1GB (or more, better 2GB) of RAM
3. 120MB (or more) system disk
4. 1Gb (or more) network adapter for data transfer
5. 100 Mb network adapter - not necessary but it comes with the system anyway and can be used for private LAN connection with robotic library controller
6. Tape drive adapter (whichever is appropriate (SCSI, Fiber Channel) for **mover** node

3.2 Additional storage

3.3 Network switch

No strict requirements

4. System configuration

We recommend to have the following configuration:
For the small system (one robotic library with one or 2 tape drives and few thousands tapes).
Minimal configuration:

1. host1: pnfs server, apache web server, configuration_server, log_server, alarm_server, inquisitor, event_relay, ratekeeper, postgresQL DB server, file_clerk, volume_clerk, info_server, accounting_server, drivestat_server
2. host2: media_changer(s), library_manager(s), mover

*** Note that this configuration may have problems as the number of accesses and their rates increase. It is always better to run one mover on a separate host

*** Note! For enstore demo only one host is required.

Recommended configuration:

1. host1 : pnfs server
2. host2 (head node): apache web server, configuration_server, log_server, alarm_server, inquisitor, event_relay, ratekeeper
3. host3: media_changer(s), library_manager(s)
4. host4: postgresQL DB server, file_clerk, volume_clerk, info_server, accounting_server, drivestat_server
5. host5: backups, plots, migration work, etc. This can be done on one of existing hosts but may interfere with operations.
6. one host per mover.

Additional requirements:

Disable firewalls. Enstore is internal to the site product and is supposed to be protected by the firewall at the organization network level.[1]

5. Enstore installation procedure.

Enstore is written in python and currently requires a slightly modified python language interpreter. Enstore uses a Fermi Tape Tool (FTT) library and set of tools providing access to different types of tape drives. It is included into enstore rpm for the revisions later than 2.0.0-0. These installation procedures describe installation on Linux OS. For installation on other systems please contact enstore developers.

Enstore is a distributed system which components can run on hundreds of nodes. It has a special tools for the product installation, distribution, and synchronization. Before starting the installation please define at least the minimal configuration of the nodes on which the system will run. If you do not want to this just follow instructions for Fast automatic installation.
The installation and configuration of enstore can be done in many different ways. Here we offer one of them. You are welcome to develop your own procedures. We will appreciate if you share with us you experience. Please send your suggestions and feedback to enstore mailing list.
Described here installation procedure is based on the assumption that the installation starts on the node where **enstore configuration server** will run – enstore head node.

5.1 Remote access

Before installing enstore you need to set up all its nodes for remote access. This step is not needed for enstore demo.
remote access and product distribution is provided by the following scripts in $ENSTORE_DIR/sbin

1 Firewall still can be set, but with very “liberal” rules. Ask for details.

coming with enstore rpm:

1. enrsh - analog of rsh
2. enrcp - analog of rcp
3. enrsync - analog of rsync

2 options are currently provided for secured access: kerberized rsh and rcp, and ssh.
For using either of them you need to setup kerberized or (and) ssh access to all enstore nodes for 'enstore' and 'root' accounts.

The mentioned scripts first check for the presence of the directory containing kerberized utilities. If it is found they assume that krb5 utilities will be used. If $ENSSH or $ENSCP are defined the ssh and scp will be used. As the last resort they try to use non kerberized rsh, rcp, and rsync.

5.2 Rpm installation.

The enstore rpm and additional rpms are located here ftp://ssasrv1.fnal.gov/en/. Currently there are rpms for SL4 and SL5. They are located in lts44 and slf5x sub directories correspondingly.
For other OS versions please check enstore download page or contact enstore mailing list.
Installation of enstore always begins on enstore head node - the node where configuration server will be running.
You can try automatic or manual installation. Automatic installation may not always go through all steps as there may be unpredictable differences in particular OS configurations and preinstalled packages. We recommend to follow steps in the automatic installation script for the manual installation and fix problems as they may appear. The installation script can be found here: install_enstore_rpm.sh
Copy it into your working directory for automatic installation or for the reference.
You need to work from "root" account.

5.3 Fast automatic installation.

Installs enstore and a required products on a single node, starts enstore, entv, and completes few demo transfers. You need to install this on system supporting X-windows, Check that X-windows applications can run from root account on your terminal. Copy script install_enstore_demo.sh from repository to your work directory, set execute permissions, and run it as user “root”. If this script runs successfully it will create complete enstore instance on a single node according to configuration, described in in /opt/enstore/etc/ minimal_enstore.conf. Then entv will start to show data transfers and few transfers will be done automatically. The demo installation can not be considered as production enstore configuration. Different sites use different repositories and ways of software distribution. To make install_enstore_demo.sh succeed you may want to consider installing the following rpms: tk, tcl, httpd, ncompress. We also recommend to copy all rpms from corresponding (depending on OS version and

processor architecture) enstore rpm repository to your local installation directory – install_dir. Create install_dir and 2 sub directories:

<install_dir>/<arch> - where arch is i386 or x_86_64 depending on the architecture of your machine.

<install_dir>/noarch

Copy files from

ftp://ssasrv1.fnal.gov/en/<OS_TYPE>/<arch> to <install_dir>/<arch>

ftp://ssasrv1.fnal.gov/en/<OS_TYPE>/<arch> to <install_dir>/noarch

where OS_TYPE is currently lts44 or slf5x

Run the following command

 # /root/install_enstore_demo.sh install_dir

5.4 Automatic installation.

This installation uses install_enstore_rpm.sh script and installs enstore rpm and required products.

Begin installation on the node dedicated to run as enstore head node first. The installation script has following arguments and options:

./install_enstore_rpm.sh [-c config_server] [-hqx] [force] [server] [fnal] [url]

-c config_server – specify configuration server to complete installation on satellite nodes

-h – print help

-q – quiet mode

-x – verbose output

force – force re-installation of already installed products

server – install server. Without this parameter it will install mover, which is a subset of server.

Fnal – fnal specific installation

url – ftp path to repository

All options and arguments are redundant.

To make install_enstore_rpm.sh succeed you may want to consider installing the following rpms: tk, tcl, httpd, ncompress. We also recommend to copy all rpms from corresponding (depending on OS version and processor architecture) enstore rpm repository to your local installation directory – install_dir and run install_enstore_rpm.sh as:

On enstore head node copy install_enstore_rpm.sh into your work directory and run it as (here and later bold – keyboard entries, italic - comments):

 # ./install_enstore_rpm.sh server install_dir

On the rest of enstore nodes On enstore head node copy install_enstore_rpm.sh into your work directory and run it for enstore server as:

```
# ./install_enstore_rpm.sh -c <head_node> server  install_dir
```

and for enstore mover as:

```
# ./install_enstore_rpm.sh -c <head_node> install_dir
```

6. Configuring enstore.

6.1 Enstore configuration file.

Enstore system configuration is described in the enstore configuration file that is downloaded by enstore configuration server. All servers
that need configuration information get it from configuration server. The configuration file is defined by $ENSTORE_CONFIG_FILE in $ENSTORE_HOME/config/setup-enstore file and is downloaded from $ENSTORE_CONFIG_HOST. We recommend to source setup-enstore before proceeding with creation of the configuration file (**reminder: do this on enstore configuration node**).

```
source /usr/local/etc/setups.sh
```

Enstore configuration file is a python script. It can be created and modify it using /opt/enstore/etc/enstore_configuration_template. It is recommended to put enstore configuration file into $ENSTORE_HOME/site_specific/config directory. Description of configuration file keys can be found in /opt/enstore/doc/config_params.html.
You can also use one of the real enstore configuration files, for instance $ENSTORE_DIR/etc/stk.conf
If you installed enstore using install_enstore_demo.sh your enstore configuration file is $ENSTORE_HOME/site_specific/config/enstore_system.conf on enstore head node. You can modify this file to create a production enstore instance, move servers to different hosts, add or remove movers change existing server settings and so on.

6.2 Farmlets.

Farmlets are the files containing node names for distributed operations on enstore nodes. These files are used by rgang command (multinode analog of rsh). You need to create initial farmlets and **then add and modify them when adding or removing an enstore node**.
Initially farmlets can be created by /opt/enstore/external_distr/make_farmlets.sh script or manually. You have to create enstore configuration file before running make_farmlets.sh. Below is the example of farmlets directory on one of the enstore production system.

cdfen cdfen-all cdfen-alla cdfen-lto3 cdfenmvr
cdfenmvr101a cdfenmvr102a cdfenmvr104a cdfenmvr105a cdfenmvr110a

cdfenmvr11a cdfenmvr12a cdfenmvr13a cdfenmvr14a cdfenmvr15a
cdfenmvr16a cdfenmvr17a cdfenmvr18a cdfenmvr19a cdfenmvr1a
cdfenmvr20a cdfenmvr21a cdfenmvr22a cdfenmvr23a cdfenmvr24a
cdfenmvr25a cdfenmvr26a cdfenmvr27a cdfenmvr28a cdfenmvr29a
cdfenmvr2a cdfenmvr3a cdfenmvr-a
cdfensrv cdfensrv0 cdfensrv1 cdfensrv2 cdfensrv3
cdfensrv4 cdfensrv5 cdfensrv6
enstore enstore-down

Here you can see a separate file for each host containing a host name.
There also are additional files grouping all movers (cdfenmvr) and all servers (cdfensrv). These files are not required, but very useful.
The mandatory files are **enstore** and **enstore-down,** that are used for starting enstore components on all enstore hosts and stopping them correspondingly. The content of the files reflect the order of nodes to which commands are sent. In "enstore" first node must be the one that runs enstore configuration server, then the rest of the server nodes and movers. "enstore-down" contains the list of enstore nodes in the order reverse to the order in "enstore".
In the current installation farmlets must be in in /usr/local/farmlets directory on each enstore node.

6.3 Data bases

Enstore has 3 databases: file/volume, accounting, and drivestat.

1. file/volume database contains information about all files written to enstore and all volumes (tapes) defined in enstore
2. accounting database contains information about all successful and failed data transfers
3. drivestat database contains information about tape drives configured in the system

The database management system is PostgresQL. The version of posgresQL must be not lower than 8.
The database server can be installed from rpm. The latest rpms can be found here.
All these 3 databases can be served by separate DB servers or just by one. Install PosrgresQL on the server defined in the enstore configuration file. The script that configures and creates databases is oriented on a single DB server ($ENSTORE_DIR/external_distr/install_database.sh).
This script requires corresponding entries in the enstore configuration file. If it detects some discrepancies it asks to correct them. After installation pg_hba.conf file may need modifications to allow access to databases from different hosts and accounts. This script can be used to make different DB configurations. You can also consult description of DB tables and script in the corresponding subdirectories in ($ENSTORE_DIR/databases).

6.4 PNFS

Before staring writing data into enstore you need to nfs mount pnfs, create subdirectories, and set pnfs

tags for enstore.
Below is some information regarding pnfs taken from "Enstore User Guide" (Ch 4).

6.4.1 PNFS Directory Limitations

It is recommended to keep the number of files in any given PNFS directory under 2000. This is recommended for any NFS-based file system.

6.5 About PNFS Tags

Before files can be written to tape, Enstore needs to know where and how to write them. Pnfs uses tag files (usually just called tags) in the /pnfs namespace to specify this type of configuration information, and **encp** transfers this information to Enstore. Tags are associated with directories in the /pnfs namespace, not with any specific file, and thus apply to all files within a given directory (with the exception noted below). When a new directory in the /pnfs namespace is created, it inherits references to the tags of its parent directory. It is a feature of PNFS that a change to a parent directory will also affect its existing subdirectorys' tag references. Manually setting a directory's tags will destroy references to its parent directory's tags. This may be what you want to do, but be aware.

A file gets the tag references of its directory as they exist when the file is written to Enstore, and these are what **encp** uses to access it. Subsequent changes to a directory's tag references do not affect pre-existing files, therefore it is possible to have files in a directory to which the current directory tags do not apply.

Allowable characters within tags are: alphanumeric characters, underscore (_), dash (-), and slash (/).

6.6 Tag Listing

The tags include:

file_family
: This tag determines the file family associated with all files in this directory. See section 2.2.1 *File Family* for information on file families.

file_family_width
: This tag determines the file family width associated with all files in this directory. See section 2.2.2 *File Family Width* for information on file family width.

file_family_wrapper
: This tag determines the file family wrapper associated with all files in this directory. See section 2.2.3 *File Family Wrapper* for information on file family wrappers. The default is cpio_odc.

library

This tag determines the virtual library (and thus the library manager) associated with all files in this directory. See section 8.3 *Library Manager* for information about the library.

storage_group
: This tag determines the storage group associated with all files in this directory, and shows up as your experiment's top level directory under /pnfs. Typically, one storage group is associated with an entire experiment. A storage group is assigned to each experiment by the Enstore administrators. Users never change this tag.

6.7 How to View Tags

Off-site users cannot mount pnfs, and therefore cannot see tags. On-site users: to see the values of the tags for a given directory, first setup **encp** (with qualifier, see section 6.1 *Setup encp*) then cd to the /pnfs subdirectory of interest (or enter the directory as an argument to --tags) and enter the command:

```
% enstore pnfs --tags

.(tag)(file_family) = dcache
.(tag)(file_family_width) = 1
.(tag)(file_family_wrapper) = cpio_odc
.(tag)(library) = eagle
.(tag)(storage_group) = test
-rw-rw-r--   11 xyz     sys            6 Jul 26 10:22 .(tag)(file_family)
-rw-rw-r--   11 xyz     sys            1 May  5  2000 .(tag)(file_family_width)
-rw-rw-r--   11 xyz     sys            8 May  5  2000 .(tag)(file_family_wrapper)
-rw-rw-r--   11 xyz     sys            5 May  5  2000 .(tag)(library)
-rw-r--r--   11 xyz     sys            4 Jul 26 10:20 .(tag)(storage_group)
```

The output first lists the tags and their values, then the tags again in long format to show the owners and protection modes.

6.8 Setting tags

Setting tags

We recommend root 644 permissions for library and storage_group tags. Permissions for other tags can be 666 to allow everyone to

change file_family, file_family_width, or file_family_wrapper

root owned tags can be changed only on the machine where pnfs server runs.

The pnfs help command describes what options can be used to get pnfs information or set certain values:

```
###
[enstore@d0ensrv2n ~]$ enstore pnfs --help
Usage:
       pnfs [OPTIONS]...
       --bfid <FILENAME>     lists the bit file id for file
       --cat <FILENAME> [LAYER]  see --layer
       --const <FILENAME>
       --counters <FILENAME>
       --countersN <DBNUM>   (must have cwd in pnfs)
       --cp <UNIXFILE> <FILENAME> <LAYER>  echos text to named layer of the
                             file
       --cursor <FILENAME>
       --database <FILENAME>
       --databaseN <DBNUM>   (must have cwd in pnfs)
       --down <REASON>       creates enstore system-down wormhole to prevent
                             transfers
       --dump                dumps info
       --duplicate [FILENAME] [DUPLICATE_FILENAME]  gets/sets duplicate file
                             values
       --echo <TEXT> <FILENAME> <LAYER>  sets text to named layer of the file
       --file-family [FILE_FAMILY]  gets file family tag, default; sets file
                             family tag, optional
       --file-family-width [FILE_FAMILY_WIDTH]  gets file family width tag,
                             default; sets file family width tag, optional
       --file-family-wrapper [FILE_FAMILY_WRAPPER]  gets file family wrapper
                             tag, default; sets file family wrapper tag,
                             optional
       --filesize <FILE>     print out real filesize
   -h, --help                prints this messge
       --id <FILENAME>       prints the pnfs id
       --info <FILENAME>     see --xref
       --io <FILENAME>       sets io mode (can't clear it easily)
       --layer <FILENAME> [LAYER]  lists the layer of the file
       --library [LIBRARY]   gets library tag, default; sets library tag,
                             optional
       --ls <FILENAME> [LAYER]  does an ls on the named layer in the file
       --mount-point <FILENAME>  prints the mount point of the pnfs file or
                             directory
       --nameof <PNFS_ID>    prints the filename of the pnfs id (CWD must be
                             under /pnfs)
       --parent <PNFS_ID>    prints the pnfs id of the parent directory (CWD
                             must be under /pnfs)
```

```
--path <PNFS_ID>      prints the file path of the pnfs id (CWD must be
                      under /pnfs)
--position <FILENAME>
--rm <FILENAME> <LAYER>  deletes (clears) named layer of the file
--showid <PNFS_ID>    prints the pnfs id information
--size <FILENAME> <FILESIZE>  sets the size of the file
--storage-group [STORAGE_GROUP]  gets storage group tag, default; sets
                      storage group tag, optional
--tag <TAG> [DIRECTORY]  lists the tag of the directory
--tagchmod <PERMISSIONS> <TAG>  changes the permissions for the tag; use
                      UNIX chmod style permissions
--tagchown <OWNER> <TAG>  changes the ownership for the tag; OWNER can
                      be 'owner' or 'owner.group'
--tagecho <TEXT> <TAG>  echos text to named tag
--tagrm <TAG>         removes the tag (tricky, see DESY documentation)
--tags [DIRECTORY]    lists tag values and permissions
--up                  removes enstore system-down wormhole
--usage               prints short help message
--volume <VOLUMENAME>  lists all the volmap-tape for the specified
                      volume
--xref <FILENAME>     lists the cross reference data for file
```

Please notice that pnfs tags are inherited in child directories. The inheritance breaks it a tag is changed in the child directory.

6.9 PNFS Trash Bins.

Trash bins must be set for pnfs levels level 1 and 4. Smply make subdirectories 1 and 4 in a directory referred as directories "trash_bin" in /usr/etc/pnfsSetup configuration file. The information gets written into these directories when files get deleted from pnfs. Later a special cron job (delfile) goes through the files in these directories, cleans then and marks corresponding files on tapes as deleted (but does not actually delete them). When all files on tape are marked "deleted" the tape can be recycled using a special command. A great deal of precaution is taken about deleting files on tapes, so even if tape is deleted it gets renamed into ORIG_NAME.deleted, and can be recovered. If tape was recycled, but no new files have been written, it can still be recovered.

7. Installing web server and configuring web site.

Enstore uses Apache web server. This server can be installed directly from http://httpd.apache.org/ Installation instructions can also be found there. For configuration of enstore web site install the following rpm: enstore_html . Before installing rpm you can configure areas where web site will be located. It is again done in $ENSTORE_CONFIG_FILE. Description of web server keys can be found in $ENSTORE_DIR/etc/config_params.html.

Note! All configuration and installation is done automatically in install_enstore_demo.sh

8. Starting enstore

After enstore is installed and configured it can be started in several ways.

1. On boot of the individual host.

All enstore components will start on the individual host on boot, but only if configuration server is running.

2. By command on individual host.

Needs to be done as user "enstore":

```
[enstore@ens07 ~]$ id
uid=6209(enstore) gid=6209(enstore) groups=6209(enstore)
# check what enstore processes are running in this host
[enstore@ens07 ~]$ EPS
# start enstore servers configured to run on this host in $ENSTORE_CONFIG_FILE
[enstore@ens07 ~]$ enstore start
Checking null1.library_manager.
Starting null1.library_manager: 193.146.197.219:7511
Checking 9940B.library_manager.
Starting 9940B.library_manager: 193.146.197.219:7522
Checking null1.media_changer.
Starting null1.media_changer: 193.146.197.219:7520
Checking stk.media_changer.
Starting stk.media_changer: 193.146.197.219:7523
# check what enstore processes are running in this host
[enstore@ens07 ~]$ EPS
enstore  29495  0.6  0.4 17492 8568 pts/2    -    23:41   0:00 python
/opt/enstore/sbin/library_manager null1.library_manager
enstore  29497  0.8  0.4 17448 8568 pts/2    -    23:41   0:00 python
/opt/enstore/sbin/library_manager 9940B.library_manager
enstore  29499  0.9  0.3 17272 7424 pts/2    -    23:41   0:00 python
/opt/enstore/sbin/media_changer null1.media_changer
enstore  29501  2.0  0.3 16208 7424 pts/2    -    23:41   0:00 python
/opt/enstore/sbin/media_changer stk.media_changer
```

3. By command on all nodes in configuration

Needs to be done as user "enstore"

```
# enstore Estart
# this will start enstore servers on all enstore nodes described in
FARMLETS_DIR/enstore
# the very first host in this file must be the host on which enstore configuration
server runs.
# for more information on farmlets see section above.
```

9. Running enstore.

Note! If you change enstore configuration file you must reload configuration by the following command:

 # enstore config –config $ENSTORE_CONFIG_FILE –load

Note!! All commands must be done as user enstore if not specified differently.

Note!!! All enstore commands have –help option. If you do not know what can be done just type **enstore** and investigate what options are available. Also refer to “Enstore Admin Guide” (can be found in /opt/enstore/doc/guides/Enstore_Administrator_Guide.odm – Openoffice document)

To run enstore you need to declare volumes in robotic library to enstore. This is done with the following command on enstore Volume Clerk host:

 # enstore vol – add <VOLUME_NAME> <LIBRARY> <STORAGE_GROUP> <FILE_FAMILY> <WRAPPER> <MEDIA_TYPE> <VOLUME_BYTE_CAPACITY>

where:

- VOLUME_NAME – name of tape volume as known to the robotic library
- LIBRARY – name of enstore library (same as library pnfs tag)
- STORAGE_GROUP – storage group to which this tape is assigned (or none)
- FILE_FAMILY – file family to which this tape is assigned (or none)
- WRAPPER - “cpio_odc” or “cern” (or none)
- MEDIA_TYPE – type of media as known to the robotic library
- VOLUME_BYTE_CAPACITY – capacity of tape in bytes

After volumes are added to enstore you can start writing to them. Please refer to "Enstore User Guide" and “Enstore Admin Guide”

10. Setting up and configuring dcache and dchache pool nodes to use enstore.

This section describes how to set up and configure dcache pool node to to use enstore as HSM backend.

It is assumed that dcache is already installed on the pool node. The complete and comprehensive dchache documentation can be found in dcache book.

10.1 Installing enstore on dcache pools nodes.

Enstore must be installed on all dcache pool nodes, which use enstore as a tape backup system.

1. Install enstore as described above. One of the recommended approaches is to run the following command (use root):

   # ./install_enstore_rpm.sh -c <head_node> server install_dir –where head_node is the node name where configuration server is running.

   !!! Note that scp access to head_node must be enabled.

10.2 Configuring dcache.

Here we assume that dcache is installed into /opt/d-cache

1. Most dCache distributions do not expect to run together with an backend system. To make this work, make sure that the the option ***lfs=precious*** is NOT specified in the following configuration files:

   /opt/d-cache/config/pool.batch

   /opt/d-cache/config/<pool>.poollist

2. Edit pool setup file: <PoolLocation>/<PoolName>/setup – where <PoolLocation> - is location of pool on the disk and <PoolName> is its name:

   # where is the pnfs mountpoint

   hsm set enstore -pnfs=/pnfs/fs

   # where is the external hsm copy script.

   hsm set enstore -command=<ENSTORE_DIR>/dcache-deploy/scripts//real-encp.sh # - replace <ENSTORE_DIR> with the value of $ENSTORE_DIR. $ENSTORE_DIR gets created by running:

   # source /usr/local/etc/setups.sh

3. * Restart pools. You may want to do this after checking that real-encp works in 10.3

10.3 Checking real-encp.

1. Login to dcache enabled node as root
2. Setup environment:

   # source /usr/local/etc/setups.sh
3. Select or create a test pnfs directory.

   # mkdir /pnfs/fs/usr/tape/test_dir
4. Make sure that all pnfs tags pertaining to enstore are set correctly.

   # cd /pnfs/fs/usr/tape/test_dir

   # enstore pnfs –tags

   The sample output:

   .(tag)(library) = CD-LTO4

   .(tag)(file_family) = fd_data

   .(tag)(file_family_wrapper) = cpio_odc

   .(tag)(storage_group) = demos

   .(tag)(file_family_width) = 1

   -rw-r--r-- 11 root root 9 Feb 1 14:05 /pnfs/fs/usr/tape/test_dir.(tag)(library)

   -rw-rw-r-- 11 1019 5111 11 Jan 1 12:02 /pnfs/fs/usr/tape/test_dir.(tag)(file_family)

   -rw-rw-r-- 11 1019 5111 8 Sep 6 2000 /pnfs/fs/usr/tape/test_dir.(tag)(file_family_wrapper)

   -rw-r--r-- 11 root root 5 Sep 6 2000 /pnfs/fs/usr/tape/test_dir.(tag)(storage_group)

   -rw-rw-r-- 11 1019 5111 1 Jul 2 2001 /pnfs/fs/usr/tape/test_dir.(tag)(file_family_width)
5. Write a file into selected dcache pnfs directory:

   # /opt/d-cache/dcap/bin/dccp <file> /pnfs/fs/usr/tape/test_dir/f1
6. Find what pnfsid was assigned to the written file:

   # cd /pnfs/fs/usr/tape/test_dir

   # enstore pnfs –id f1

   Let's sat the output was 000200000000000000015C78

7. Check in what pool it has been written. Lets assume that it has been written to poll node **poolnode001** and the file pnfsid is 000200000000000000015C78.
8. login to this node as root.
9. Locate the data file corresponding to pnfsid 000200000000000000015C78. Assume it is in dcPools/pool1/pool/data/000200000000000000015C78 and its size is 1048576
10. Copy file into enstore (on tape):

    **# /opt/enstore/dcache-deploy/scripts/real-encp.sh put 000200000000000000015C78 **
    **/dcPools/pool1/pool/data/000200000000000000015C78 /usr/local/bin/real-encp.sh **

    **'-si=size=1048576;new=true;stored=false;sClass=test.enstoretest;cClass=-; **

    **hsm=enstore;path=/pnfs/pic.es/tape/paco/test_v20; **
    **;path=<Unknown>;group=test;family=enstoretest;bfid=<Unknown>; **

    **volume=<unknown>;location=<unknown>; **
    ' -pnfs=/pnfs/fs/usr -command=/opt/enstore/dcache-deploy/scripts/real-encp.sh
11. Check that file was written.

    11.1 When file is written into dcache it has a non-empty layer 2 in pnfs

    11.2 When file is written into enstore it has non-empty layers 1 and 4:

    # enstore pnfs --layer 2G_1 1

    GCMS128949904300000 # bit file id

    # enstore pnfs --layer 2G_1 4

    TST075 # tape volume

    0000_000000000_0000001 # location on tape

    2000000000 # file size

    gcc2 # file family

    /pnfs/data2/test/moibenko/LTO4/2G_1 # file path

    000200000000000000B5AF0 # pnfsid

    GCMS128949904300000 # bit file id

    stkenmvr141a:/dev/rmt/tps2d0n:9310091462 # mover/drive the file was written to

    1803076910 # checksum

11. NOTES