Enstore Status Report

D0:

Enstore working on Igor T's pc with disk IO
Running library manager, mover and media changer
Other servers are being used from our test setup

Added simple mount/load/seek delays for disk IO to help simulate real tape IO: will install when needed

New requirements:

1. Need ability to specify transfer priority: OK
2. Need priority aging mechanism: Extension – OK
3. Need transfer timings: OK

Enstore part on track for Oct 22 prototype system

CDF:

Several meetings trying to decide how proceed
Waiting for plan
CDF will most likely not use a full version of enstore and maybe none of it at all

Prototype working again

- overnight writers fill disk and stop correctly
- readers run over weekend without errors
- possibly some memory leaks – investigating
  - python garbage collection

Redwoods and STK silo:

- connected to 2 pc's and 100 MB/s ethernet: working
- pc's have limited IO rate capacity
  - disk reads limit the real transfer rate to 5 MB/s
  - shared memory tests run at 9.5 MB/s
- No bottlenecks discovered in mover design

Infrastructure:

- User interface is being standardized
- Server–client message passing "ticket" information being
  - cleaned up and standardized
- General cleanup from prototype to supportable code
- Python debugger being enhanced
  - want something like dbx –p ability
- Python trace module begin written
  - based on Ron's trace tool in vxworks
- Standard module build – not standard across platforms
  - change from python 1.5 to 1.51 in make targets

Information Broker: New, in planning stage

- When system is having trouble, we don’t want a deluge of users querying it and causing more trouble
- Most information changes slowly
- Information Broker will be the central place that queries system and has "all" information
- User contacts information broker for all queries
- Not disabling (for now) queries to servers – obscure

Encp binary distribution: Almost working

- Most users will just want to use encp and not want all the baggage that goes with it (python,ftt libraries, libtp...)
- Python freeze feature produces an encp binary
  - Nothing else needed to write/read mss
- This was a lot more work than anticipated
  - all linked shared libraries need to be removed
  - kits release of python can’t be used
  - well worth effort

Database investigations:

- Raima: (Desy uses this): too expensive right now
- evolving to sql interface to allow database changes

New feature: Process a list of input files

adopted unix cp syntax:
at most 1 mover will process your list at one time (use more than 1 encp to get more than 1 mover)

encp file1.file2 ... fileN dir

On reads from MSS, files are sorted by which volume and submitted for processing in chunks

On writes to MSS, files are processed sequentially, but output volume is not dismounted between files
Note that all files will be written to 1 file family

Code is working, not fully optimized

New feature: Volume maps: working

Need to know file placement on tapes to get best thruput

HPSS runs a job once a week and produces a list.
Enstore: produce map user can access immediately

Method:

- Create a duplicate entry in pnfs
- Ordered by file family and volume
- File entry naming conventions are such that file order is readily known

Example:

Let’s say user transferred a file to mass storage and called it
/pnfs/enstore/one
The file was stored on volume named "flop301" with a
byte offset=0 and in a file family named "jon".

In this case, the duplicate entry is
/pnfs/enstore/volmap/jon/flop301/0

In the general case,

- "jon" becomes the file family name,
- "flop301" would become the volume name
- "0" would be the file order on tape.

User’s can do an ls on all files in the flop301 directory to get the list of all the files that are on the volume.

Simple commands are available to allow cross–referencing of the 2 files:

```
$ pcmd info /pnfs/enstore/volmap/jon/flop301/0
    bfid="90104096300000L";
    volume="flop301";
    cookie="(0, 512)";
    file_family="jon";
    orig_name="/pnfs/enstore/one";
    map_file="/pnfs/enstore/volmap/jon/flop301/0";
```

Since the user will not have delete authority on the volmap file, this mechanism gives us an automatic trash can facility

- If the user deletes their entry to the file, we still have the volmap entry and can "easily" restore the original
- Already tested for single file restores

Desy has modified PNFS to provide hard links for us
- don’t have to duplicate data, just links to it

Had initial discussions with Marilyn for using OCS:

1. I was under the impression OCS was central place of drive and volume statistics – not clear any more. Will need a central database for these statistics – not enstore

2. Planning on using xtape to communicate with operator's for the rare mount outside of the robot.