Xrootd HPSS and Archiving

Links

http://confluence.slac.stanford.edu/display/ds/Xrootd+Discussion+-+January+2008

Xrootd and HPSS Access

Xrootd is able to retrieve files from HPSS. If a users requests a file that is not on disk Xrootd will check if this file exists in HPSS and if so it will be staged to disk. All of this is transparent to the client and the only difference it sees is that it takes a little bit longer to access the file.

In order to have HPSS work efficiently it is important to have large files be put into HPSS. The time to mount a tape and to position it in order to read the file are significant (ten's of seconds). Typically files sizes of 1GB and larger are desirable.

Migrating files to HPSS and purging from disk

Xrootd itself does not put new files by itself into HPSS and neither does it remove files if disk space runs short. It, however, setups new files in such a way that the migrate daemon is able to identify new files.

An xrootd data server runs a migrate and a purge daemon. Both of them run periodically (typically every 10 min for the migration and every 30 min for the purging).
The migrate daemon looks for new files, files that were transfered into xrootd (with xrdcp for example).   If it finds new files it will transfer them to HPSS and mark them as old (in HPSS). Files that are staged from HPSS are marked as old.

The purge daemon will check the disk space periodically. If the free space is less then a certain percentage (configurable) it will search the directories for old files. These files are sorted by age and the oldest ones will be removed until the free space reaches a certain limit.
For example: Lets assume the required free space is al least 5% and the purge threshold is 10% free space. If the disk fills up more then 95% the purge daemon will remove files until the disk usage is 90%.

Archiving file to HPSS (Archiving)

As mentioned earlier putting small files into HPSS is inefficient:

- The overhead mounting and positioning a tape is large compared to the tape I/O.
- Both reading and writing will be inefficient.
- Access to the HPSS meta-data catalog might be slowed down for large number of files (but it is not clear were the limit is).

A possible solution is to tar small files and then only copy the tar file into HPSS:

- tar file is create on a data server
- need to maintain extra catalog that contains the tar file listing.
- If a file is not on disk xrootd has to check HPSS and tar files in HPSS.
- Xrootd doesn't know anything about tar files when placing files on a data-server.
- Try to prevent that tar files are staged to multiple data servers.
- policy if files should be tared depending on:
    - file path
    - file name (regex)
    - size

Migrate files to HPSS

For migration:

1. Only large files are directly migrated to HPSS
2. Small files are somehow sorted and copied into a tar file
3. A record is kept which files are in a tar file
4. Tar files are migrated to HPSS (irrespective of its size)

For Xrootd (assuming a file is not on disk):

1. Check if a file is in HPSS. If so just stage it to disk
2. Check if file is in a tar file
3. Stage the tar file to disk
4. untar the file and put the content into the xrootd directories

In addition it is needed that Xrootd is:

1. able to create a directory listing from what is in HPSS and what is in the tar files
2. able to provide file stat (size, last modification time) irrespective if the file is on disk, in HPSS or tarred

Tarring Files

- Files on a data server are tarred independently from the other data servers.
- Using the file size to decide if a file should go into a tar. Size 500MB or 1GB? Are there files that should go directly into HPSS even if smaller?
- Files to be tarred should be sorted in order to increase the probability that a client might use more then one file from a tar.
    1. first sort by stem:
       e.g.: /glast/mc/DC2/ChickenLittle-GR-v7r3p24-2
    2. sort by type: DIGI.root, MC.root, RECON.root, merit.root
       Should types be mixed in a tar?
- Create tar-file info (eventually this info should go into a database). Other values needed for a stat command (usser, permissions, file type) can be created on the fly.
    1. data file name
    2. size
    3. time put into tar (Could be the same for all files in a tar-file)
- Tar file size 2GB, 4GB ?
- How often should it run? Depends on how files should be sorted.
- Tar files are written to the local xrootd disks.

Retrieving Tar Files

A client doesn't know anything about tar files. It will ask xrootd to open a fail and xrootd has to
determine if the file is: on disk, in HPSS, or in HPSS as a tar file.

A client that wants to open a file connects to a redirector (RDR) and the redirect asks its data-server if they have this file on disk. If none of the data-servers replies the RDR assumes that the file is not on disk and it will pick a server that allows staging from HPSS and tell the client to go to that server.
The client will then ask this data-server tp open the file which will trigger the copy from HPSS to disk. If a second client asks for the same file
it will be redirected to the same server the first client was redirected to (even if the file is not yet on disk).
This makes sure that a file is only staged to a single server.

The data server could be setup so that upon an open request it would:

1. check if the file is on disk
2. check if it is in HPSS and if so stage the file
3. check if the file is in a tar file and if so get the file from the tar file

The naive approach to stage the tar file and untar'ing would not work:
If two clients ask for different files which are in the same tar file, the redirector might redirect the two clients to different data servers (the RDR doesn't know anything about tar files) and subsequently the tar would be staged twice.
As it is likely that clients open many files from the same tar this would lead to unwanted data duplication.

Solution??