Enstore Small File Aggregation User Guide

CS-doc-4698
Data Movement and Storage Department
1/9/2013

This document describes the Enstore Small File Aggregation feature from the user perspective. A description of the feature is presented, its operation is described, user interfaces are detailed, and recommendations for use practices are made.

Motivation

Tape media are optimal for large files where the per file overhead is not significant compared to the time to write or read the file. In general, tape is not very efficient for small files. The major overhead for writing a file to tape occurs when writing a file mark after the file. Normally, writing a file mark is a sync operation. It flushes any data in the tape drive's on-board write buffer to tape then writes the file mark before returning control to the requesting program. Since the on-board write buffer is then empty, the tape drive must stop and back up so it can be in position to speed up to write the next file's data when available. This "back-hitching" can take several seconds – typically 5-8 seconds. This means that the effective write speed for a 5 MB file is less than 1 MB/s. Writing 54,000 100 MB files to a T10000C tape can take over 75 hours just to write file marks, over 10x the optimum rate. Excessive back-hitching will also wear out tape drive parts at a faster rate.

It is possible to buffer tape marks so that writing a tape mark is no longer a sync operation. This is inherently unsafe however, since the drive returns success to the calling program even though the data is in the drive's volatile write buffer and not on tape. A power failure or other glitch can result in undetected data loss. This mode can be useful in certain circumstances – in particular copying data where verification (read back with checksum verification) of the copied data is performed, but is unsuitable for general use.

Some tape drive manufacturers provide enhanced small file performance features that are vendor and model specific. For example, the T10000C tape drives have a File Sync Accelerator feature (FSA, not to be confused with the enstore SFA feature). With this feature enabled, small files (< 250MB) and tape marks are streamed to the tape in a non-consecutive order in parallel with writing the data to the internal 2GB buffer. When the buffer is full, the tape is repositioned where the FSA began, and the files and file marks that are in the buffer are written sequentially to the tape. This means there isone sync/back-hitch operation around every 2 GB of file data. This differs from buffering

tape marks in that the data is actually on tape when control is returned to the user – it is just not in the normal consecutive order (until the buffer is flushed to tape) - FSA files are always synced to tape. Throughput for small files with this feature is about 45 MB/s, which is 20% of the drives potential rate of 240 MB/s. This feature is not available for LTO.

The performance goal of this feature is to provide aggregation of small files at a faster rate than File Sync Accelerator can provide (i.e. a throughput exceeding 100 MB/s).

Back-hitching is not a problem for reads since tape drives read ahead, so as long as the read accesses are in sequential order, the drive can stream reads of small files. However, with enstore, often even though read requests are queued sequentially, non-sequential access can occur because of "discipline". Discipline is an enstore feature that throttles access to/from user computers so that networking bandwidth is not overcommitted. This throttle can result in reads being processed out of sequence. SFA read caching can help with this for small files.

Overview

Files are selected for aggregation based on their size and volume-family. SFA internally stores selected "small" files until a packaging threshold criteria is met. The threshold can be based on the number of files collected, their aggregate size, or the time since the receipt of the first file. The collection of these selection and packaging criteria makes up an SFA policy. Any number of policies can be active on an enstore system that has the SFA feature.

File aggregations are stored on tape, transparent to the user, in self-describing tar archives called packages. When a request is received for a file in a package, the tar file is fetched from tape, all files in the tar archive are unwound to a read-cache, and the specific file is returned to the user from the read cache. Enstore keeps track of what files are in the read cache, and if a request comes in for a file that is already in the read cache, it returns it to the user from the cache.

The read cache is not intended to be a general-purpose file cache for the end user. It assumes locality of reference in reads and its sole purpose is to prevent multiple tape reads of the same package to get several, or all files that are contained in a package.

SFA consists of several new enstore components and modifications to enstore servers. The new components are:

- The Write Cache contains files written by disk movers.
- The Read Cache contains files unpacked in response to a read request
- The Package Staging Area is where packages are staged from tapes for the consequent unpacking. The files then get moved to read cache.

- The Package Archive Area where files get packed for the consequent transfers to a tape.
- Disk Movers move files between read/write caches and end users
- Migrators make packages and writes them to enstore. For reads, they also extract files from packages to the read-cache. They are also responsible for enforcing cache purging policies.
- The Library Manager Director makes selection decisions on whether a file should be aggregated or go directly to tape and tells encp where to send the request lis
- The Policy Engine Server and Migration Dispatcher are responsible for maintaining the metadata for policies and their state, for making file selection decisions, maintaining aggregation lists, and making decisions as to when packages should be created, and for initiating and dispatching creation of packages

The read cache, write cache, and package staging areas are all implemented with a ZFS appliance. This appliance currently has 60TB of capacity. Each of these areas are implemented as separate ZFS pools and are NFS exported to Disk Movers and Migrators. ZFS was chosen for its data integrity features - in particular for its block level checksumming. Integrity is important since the file can remain in the caches for a long time, depending on the availability of tape drives and the caching policy. With encp write transfers that end up in the write cache, the usual enstore end-to-end file checksumming stops when the file is written to the cache. The integrity of a file in the write cache, until it is written to tape, is maintained by the ZFS file system (more on this later).

Reads of packaged files are transparent to the end user using encp or dCache clients. Writes require a command line switch to encp to enable it to talk to the library manager director instead of the regular library manager. Details on how to use the SFA feature are described below.

Enabling and Using Small Files

To write aggregated files you must have a policy defined for you by a Storage Services Administrator. In addition you must use a special switch in encp to tell it to talk to the library manager director and must use an encp release of v3_11 or higher. To read back aggregated files you need to do nothing special. Enstore will read the file from the disk cache; if the file is not in the disk cache, the package containing the file will first be read from tape and all of the files it contains will be extracted and placed into the read cache.

When you request a policy, you will need to specify several pieces of information:

1. The volume family (file family, storage group) for which the policy is to apply

2. The threshold file size for aggregation and for packages. Files smaller than the threshold for the volume family will be selected for aggregation. This file size is also used as the package size threshold.
3. The maximum number of collected files. If this threshold is crossed the files get packaged even if their total size is lower than the threshold file size.
4. A time threshold measured from the arrival of the first file. If this threshold is crossed the files get aggregated to a package.

If any of these three thresholds are crossed the files will be aggregated into a package and written to tape.

The specification of thresholds and other policy parameters should be determined on a case-by-case basis in consultation with DMS. Generally though, a time threshold should be less than 24 hours, the file threshold should be large enough such that a tape contains no more than 1000 files. Around 5GB is good for T10000C tape technology. The file count threshold is experiment specific.

Once the policy is in place, you need to start using the encp switch that enables aggregation on file writes to tape:

encp –enable-redirection <other encp command line fields>

For reads no special encp switch is necessary.

You can read your aggregated files through dCache. The Public dCache is also configured to write aggregated files (the encp the Public dCache uses to write files to tape uses the enable-redirection switch).

Monitoring

Enstore has a set of web based monitoring pages that provide information on the state of files that are being aggregated and the state and health of the aggregation feature. The following web pages are provided:

1. A Heads Up Display (HUD) page. This is linked in off of the Enstore “System Summary” page for the particular instance of enstore. This page has information on the health of the feature, including usage information about the disk caches and a link to the Files in Transition Web page.
2. Files in transition page. This page displays, per policy, all of the files that are being aggregated but have not been written to tape yet. A web page is provided that contains information, for each policy, about the parameters of the policy and the state of the policy, that is, how close it is to packaging up the file to write to tape. Each file that is awaiting packaging is listed along with its size and the timestamp

of when it was written into the write-cache. This page can be navigated to from the HUD page.

3. A listing, by storage group, of completed write transfers to tape in the last 72 hours. This page is a new page that is not specific to this feature and is generally useful.

HUD Page

The HUD displays the following SFA health indicators:

1. For the write cache volume the total number of KB of files in transition, the total number of files in transition, and the total size of the disk volume in KB
2. For the package staging disk volume: the total number KB used on the volume, the total number of files on the volume and the total size of the volume in KB
3. For the read cache disk volume, the total number of KB used on the volume, the total number of files on the volume, and the total size of volume in KB
4. A link to the Files in Transition page

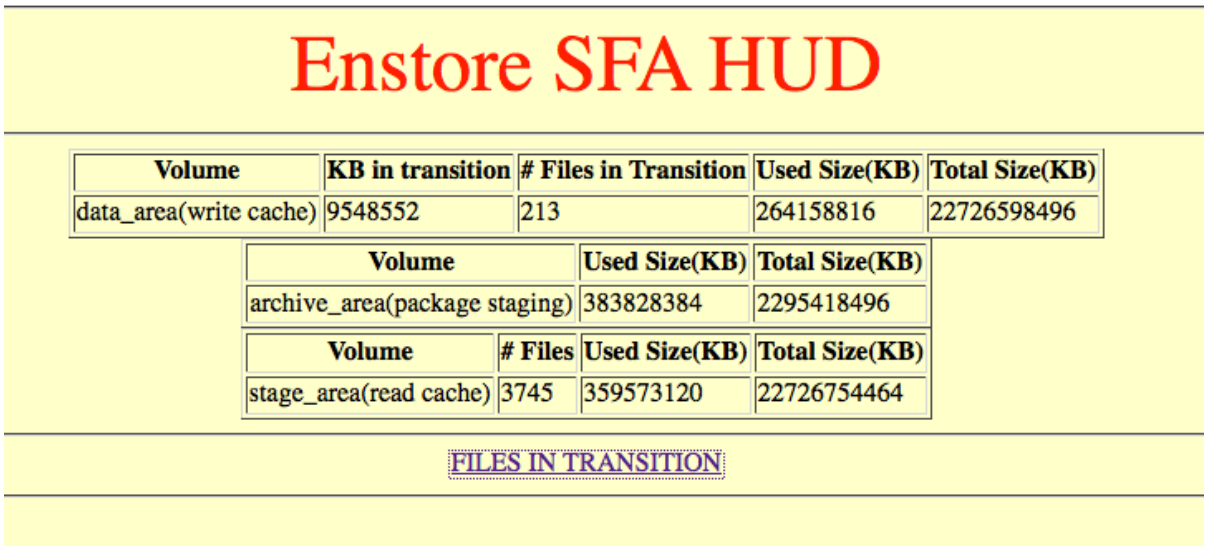
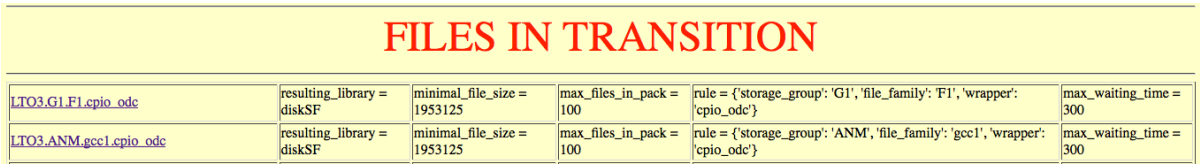


Figure 1. Small Files Aggregation Heads Up Display

Files In Transition Page

This page provides on-demand information on files in transition (not yet packaged and written to tape).



FILES IN TRANSITION

LTO3.G1.F1.cpio_odc	resulting_library = diskSF	minimal_file_size = 1953125	max_files_in_pack = 100	rule = {'storage_group': 'G1', 'file_family': 'F1', 'wrapper': 'cpio_odc'}	max_waiting_time = 300
LTO3.ANM.gcc1.cpio_odc	resulting_library = diskSF	minimal_file_size = 1953125	max_files_in_pack = 100	rule = {'storage_group': 'ANM', 'file_family': 'gcc1', 'wrapper': 'cpio_odc'}	max_waiting_time = 300

Figure 2. Files in Transition page policies in effect

Each policy lists the rule (volume family) and the thresholds for the policy. The link is an anchor to a list on the same page with files in transition, if any.

In the example below, the policy for LTO3GS.ANM.FF1.cpio_odc has two file lists. The Format for each list is

- List descriptor
- List metadata
- List entry descriptor (names of fields)
- List entries (omitted here as they are too long)

pool=migration_pool, total=39, size=1974964 (kB), #of lists=1

list id=22faa586-7336-4670-bbe4-c189b3c88127, total=39,size=1974964 (kB), time_qd=Thu Apr 12 11:55:20 2012

```
cache_mod_time storage_group file_family volume location_cookie bfid size
crc pnfs_id pnfs_path archive_status
```

and

pool=cache_written, total=36, size=1410180 (kB), #of lists=1

list id=e571634b-a9f8-41f0-ba6c-a3e462576192, total=36,size=1410180 (kB), time_qd=2012-04-12 11:55:20

```
cache_mod_time storage_group file_family volume location_cookie bfid size
crc pnfs_id pnfs_path archive_status
```

The first contains one full list that is awaiting packaging (pool=migraton_pool). It has 39 files and is 1.97 GB.

The second list contains one list of files that are being collected for packaging (pool=cache_written), but which haven't yet exceeded a packaging threshold.

Metadata about the lists precedes the list of files. Metadata includes:

- List ID
- Number of files
- Total list size in KB
- Time the list was started (time_qd)

This metadata, together with the thresholds for the policy, can be used to determine how a list is progressing.

72 Hour Completed Write Transfer Listings Page

This page is identical in format to the complete file listings except that it will only contain files archived onto tape in the last 72 hours. This page is linked off of the enstore Tape Inventory Summary page from the "RECENT FILES ON TAPE" link:

Figure 3. Location of Recent Files On Tape

Clicking on the link will bring you to a page organized by Storage group from which you can view the lists of files written to tape in the last 72 hours.

Storage Group	Time	File Listing	Size
sg1	was generated: Mon Mar 26 15:30:09 2012	RECENT_FILES_ON_TAPE_sg1	313
litvinse	was generated: Mon Mar 26 15:30:08 2012	RECENT_FILES_ON_TAPE_litvinse	318
nova	was generated: Mon Mar 26 15:30:09 2012	RECENT_FILES_ON_TAPE_nova	314
ALEX	was generated: Mon Mar 26 15:30:08 2012	RECENT_FILES_ON_TAPE_ALEX	314
TEST1	was generated: Mon Mar 26 15:30:09 2012	RECENT_FILES_ON_TAPE_TEST1	1012839
none	was generated: Mon Mar 26 15:30:09 2012	RECENT_FILES_ON_TAPE_none	314
ANM	was generated: Mon Mar 26 15:30:08 2012	RECENT_FILES_ON_TAPE_ANM	313

Figure 4. Recent Files on Tape Page: link to per storage group lists

Each line in the listing contains the following fields:

- Archive modification time
- Storage group
- File-family
- Tape Volume label
- Location cookie (position on tape)
- Bit file ID (BFID)
- File size in bytes
- Adler32 checksum for the file
- Pnfs/Chimera ID
- Full file path name
- Archive status

Useful Enstore Commands and Tools

New and changed Enstore Commands and formats

There are new enstore commands in addition to new switches for existing enstore commands, and new formats for the output of some commands involving packaged files and packages.
The enstore info –list <volume> will list all files that are on the tape volume regardless of whether they are in a package or not, and packages are not listed. This format is the same as previous versions and makes the use of packages transparent.

To see all physical files written on a volume (packages and files written directly without a package) you can use the --package switch:

enstore info –package –list <volume>

The output format with this switch is:

<volume label> <file id> <file size> <location cookie> <del/active flag> <file/package name>

Where file can either be a package or a file written directly to tape.

To list all user files on a tape (files contained within a package and files written directly), including information about their packaging, you can use the –pkginfo switch:

enstore info –pkginfo –list <volume>

The format of the line with this switch is:

<volume label> <file id> <file size> <location cookie> <del/active flag> <package-id> <archive status> <cache status> <original file name>

The additional information this switch provides is the package-id and the archive and cache statuses. If the file was written directly to tape, then the <package-id> field is "None".

A package file can be retrieved using either its name, or its ID, just as regular files can. However there is an extra step to determine the package name or ID for a given regular file. One can use the commands above or the following steps:

First get information on the regular file by either the file ID or filename:

enstore info –bfid <file ID> or
enstore info –file <file name or file ID>

If the file is part of a package, the new field "package_id" will contain the package ID from which the package can be obtained. The output will also have the new fields:

archive_status
archive_mod_time
cache_status
cache_mod_time
cache_location
package_files_count
tape_label

For a file that is written directly to tape and is not associated with a package, package_files_count is “0” and package_id is “None”.

There are also fields that have changed format and meaning: external_label and drive.

The external_label will be of the form like:

“common::<volume family><time stamp>”

for a file that is part of a package. There is now a tape_label field that holds the physical tape volume label. For the output of a package file, the external_label = tape_label.

The drive field is similarly different for a file that is part of a package:

“common:<path>”

For a package, the drive field contains the tape drive the package was written to.

Note that if the location cookie (location on tape) for a file, as reported by enstore info –list <volume>, is the same as another file’s on the same volume, then they are packaged, belong to the same package on tape, and the location cookie refers to the location of the package.

Checking Package File Consistency

Small files are packaged with tar. The first file in the tar file is README.1st, which is followed by the individual files. The filenames in tar are in a hashed format that bears no resemblance to the original user supplied file name. The Readme.1st file has the mapping between the original and hashed filename. In addition, it contains the Adler32 checksum for each file. The format of this text file is:

<hashed (tar) filename > <Normal (user defined) filename > <Adler32 checksum>

With this information and the ecrc executable, which calculates an Adler32 checksum for a file and is part of the encp product, it is simple to verify the integrity of all files in a package. Below is an example script that does this:

```
#!/bin/bash
# usage verify_package <package-file>
file=$1
function do_verify() {
 nm=0; skip=0; bytes=0; files=0
 while read line; do
    [ $skip -eq 0 ] && { skip=1; continue; } #skip README.1st header
    hfn=$(echo $line | awk '{ print $1}')
```

```
    spfn="./_vtemp_${hfn}"
    ufn=$(echo $line | awk '{print $2}')
    ocrc=$(echo $line | awk '{print $3}')
    ccrc=$(ecrc $spfn | awk '{print $2}')
    sz=$(ls -l $spfn | awk '{print $5}')
    bytes=$[bytes + sz ]
    files=$[files+1]
    [ "$ccrc" !=  "$ocrc" ] && nm=$[nm + 1]
    echo "README_CRC,Calc_CRC,size,filename,namespace=$ocrc,$ccrc,$sz,$hfn,$ufn"
  done < <( cat ./_vtemp_/README.1st )

  echo "Summary: $nm CRC mismatches, files=$files, summed file sizes=$bytes"
}

mkdir ./_vtemp_
tar --force-local -C ./_vtemp_ -xf $file >& /dev/null
do_verify
rm -R ./_vtemp_
```

Limitations

Currently, because of the architecture of enstore, a single stream of files from encp can result in significant per file overheads. This is because the disk mover polls the library manager for work at a regular interval. If the library manager queue is empty, the disk mover sleeps. This will always happen with a single encp stream. Specifying a list of files to write with encp does not help either, since these are also sent serially to the library manager. The only way to overcome this currently is to have multiple encps running in parallel so that the library manager always has something in its queue. We plan to address this limitation in a future release.

This in general should not be an issue if the files are written through dCache, since dCache can start up several simultaneous writes to enstore.

Usage Guidelines

There are two performance concerns that are dependent on the policy definition and the usage pattern for that policy:

1. Tape mounts/dismounts should be minimized. A rough guideline is that filling a tape with files should require less than 1000 mounts
2. Throughput should be kept as high as possible.

There are several factors that affect the first concern above.

1. Package sizes. If the package is large enough, even if a mount/dismount is required for each package written, the number of mounts can be kept at a tolerable level.
2. Package build rate. If a single stream of files takes longer to package than the dismount wait time of a drive[1], then the tape will be dismounted before the file can be written. We can currently configure the dismount wait time globally in the migrator computers, but too large a value will waste tape drive resources that could be used by somebody else.
3. Input file rate. This affects the packaging rate, and if files are coming in too slowly, the timeout for the policy may occur with the resulting package having a less than optimal size. Enstore overheads can contribute significantly to small file transfers (see the limitations section).

We make the following recommendations, but there should be a consultation between the experiment and Data Movement and Storage to determine an optimal policy and usage pattern.

1. The package size threshold should be set such that it takes less than 1000 file to fill a tape. For instance, for T10000C, whose capacity is 5400 GB, the recommended threshold is around 5 GB.
2. The experiment should, if possible, locally accumulate enough small files belonging to a policy to be able to stream them through enstore and write multiple packages in each mount/dismount cycle.
3. Pursuant to the current limitations on enstore per file overhead, for a stream of files for a given policy, multiple encps should be running in parallel in order to keep requests queued up in the disk library manager (dCache does this automatically).
4. We recommend that the time threshold for packaging files that have not met their total files or total size threshold be set to no more than 24 hours.

In addition, we request that the experiments keep a local copy of a file on their local disk until they have verified that it is on tape. The recommended way of verifying the file is on tape is to scrape the enstore web page that lists files written to tape in the last 72 hours, which is updated every hour, before deleting the local copy. If more than 72 hours has elapsed, then the check can be done by scraping the much larger complete files listing page. DMS can also provide further guidance and consultation for verification.

---

[1] Dismount-wait: When there is no more work for an enstore mover/tape drive with a mounted tape, the mover will wait for a configurable amount of time (usually on the order of 30s) anticipating more requests for the tape will be forthcoming. This reduces unnecessary mount/dismount cycles.