SAM-FS to DMF Exodus:
tales of data movement

Jason Ozolins

NCI National Facility

ANU Supercomputer Facility

these awful slides: bit.ly/UmUYJm

Word Association Football

- **<u>TERABYTES</u>**
- **Bandwidth**
- *iops*
- latency???

If you want to work effectively with all those terabytes, you need to keep enough parallel work going to use bandwidth effectively, given latency and IOPS of underlying system

- You can hug terabytes
- You can admire a mighty data pipe
- IOPs and latency not so easy to love

Backstory

- Needed to fully restore a ~3M inode SAM filesystem ASAP
- OK, **samfsrestore**, then **stage -r** to bring it all online...
- But the remote silo with the only archive copies had only a couple of 9940B drives; seem to recall we could only use one??
- Small # drives worked fine for writing out archive copies, but...
  - **stage -r** works in recursive filesystem traversal order, relies on SAM to schedule stage requests
  - SAM maximum file staging window for scheduling tape requests is 50,000 files
  - in worst case, could need as many as 60 passes over all the tapes to get three million files online.

A hack seemed necessary

```
[jao900@dcmds0-acsls samfs-examples]$ ./statcopy1 ~/parse_dbload.pl
VSNp09     0x00000000000024a8.0009939c                5601
/home/900/jao900/parse_dbload.pl
```

- Modified Sun-provided example sam_stat.c code to emit one line per file with VSN, tarfile location, offset, in fixed format for sorting
- **sfind /home -type f -print0 | gxargs -0 statcopy > /tmp/copies**
- **sort -k 1,2 /tmp/copies > /tmp/worklist**
- **(perl -ne '@F = split " ",$_,4; print "$F[3]\0";' /tmp/worklist | gxargs -0 stage) > /tmp/log 2>&1**
- brought 3M files back online in one evening
- perl is not verbatim, but was pretty much that level of complexity
- didn't care about filenames with embedded NL characters
- **stage -r** at end to handle any silly filenames

Moving into the present...

- politics, tender timing meant entire StorageTek Silo dismantled before new T950 library arrived
- so all SAM-FS files rearchived to disk VSNs
- all ~40M files supposed to be online as well (1.3PB)
- after dmcapture of SAM /massdata filesystem metadata and restore onto DMF, just skip through dmscanfs.out and do **dmget**s for everything
- hooray, high five!
- but...
- big 1.3PB online filesystem developed errors, couldn't **samfsck**
- 120TB conventional HSM cache replaced it
- so we found out how SAM 5.2 stages stuff from disk VSNs
  - answer: with blithe optimism
  - disk VSNs assumed not subject to physical limits like seek times, rotational latency
  - starts any # of concurrent copies from one disk VSN if requests present
  - reads in small chunks (1MB)
  - no documented ways to change these behaviours in SAM 5.2

DMF differences from SAM-FS caused a few surprises

- SAM-FS inode had timestamps for data residency
  - age calculation uses residence timestamp
  - can enforce LRU release policy pretty easily
- DMF sits on CXFS metadata + xattrs
  - age calculation uses most recent of atime,mtime
  - recall from SAM does not touch atime (rightly so)
  - file with old atime just recalled from SAM is candidate for dmfsfree as soon as it satisfies tape migration policy
  - so you don't want to do redundant dmgets as you may cause tape recalls

More hacks seemed necessary

- Saw a lot of bad staging happening
  - 16 * 500MB stages from single 14TB 8+1 RAID 5 disk VSN
  - horrible throughput (<50MB/sec for that VSN) from head thrashing; guessing only 128kB read per disk per seek.
  - poor HSM throughput affecting Vayu copy queue jobs
- Read the SAM source
  - found undocumented way to restrict # of concurrent disk VSN copies, so we could limit that
  - couldn't recompile **sam_stagerd_copy** to change hardwired disk VSN read/write sizes because build depends on closed, unreleased source
  - disassembled **sam_stagerd_copy**, found 1MB buffer size constant offset in file (no deep significance; they just reused old MO disk buffer size!), patched hex dump, turn it back to binary
  - finally noticed that requests to same archive tar file (analogous: DMF zone) always get handled by one sam_stagerd_copy process.

A plan emerges

```
[jao900@dcmds0-acsls bin]$ ./statcopies ~/parse_dbload.pl
- 0100755        1   1471108               5601 /home/900/jao900/parse_dbload.pl
        0 dk   VSNp09     0x00000000000024a8.0009939c
        1 dk   VSNp20     0x00000000000070f9.01374506
```

•enhanced old **statcopy1** to display all copies and their media types
•traverse entire massdata filesystem, build lists of files by disk VSN
•sort lists by tarfile location, offset to get work order lists by VSN
•parse dmdump of SAM FTP MSP daemon records to build big Perl tied BDB hash, **live_ftp_msp_bfids**:
- original SAM path to files => BFID
- BFID => current path (handling renames)

•tmpfs is great for building such a DB if it can fit sensibly in mem.
•thanks SGI for providing 96GB on the DMF MDS nodes...
•significant space saving in this DB by storing each directory path once and referring to it as prefix for filenames in that directory.

How it fits together

- **stage-dryrun** preprocesses SAM VSN work lists against tmpfs DB to quickly skip already migrated files
- **stage-vsn-stream** worker issues dmgets for files from one SAM VSN work list:
  - for each file, checks if still needs recall from SAM using DB (because DB is updated more often than **stage-dryrun** is run)
  - check (using another, static DB) that VSN tarfile for this file is available on SAM server (some tarfiles never got written; presumably during a SAM crash?); warn and skip if tarfile not present
  - add file to dmget list if satisfies other criteria (min/max size)
  - once list full, issue dmget for that list (as file paths, see NOTE)
    - fullness criteria are total data size in list, number files in list, whether crossing over to a new tarfile on source VSN
  - and so on...

Did it help??

- hacky backgrounded shell worker loop collects VSN workfile paths from SysV message queue and starts **stage-vsn-stream** against that workfile
- start one worker loop, observe throughput
- start another, observe...
- repeat while not yet scared of inviting DMF HA STONITH
- can suspend workers with job control :-)
- got to 600MB/sec throughput, but DMF got STONITHd
- DMF seemed OK with about 300-400MB/sec on our hardware
- SAM staging behaviour well controlled, no big slowdowns.
- small file throughput still too low (tens/second)
- re-used **stage-dryrun** to prepare lists of files to prestage and tar up on SAM MDS; untar into staging area on DMF cache, fix atimes and **mv** over the original offline files using the rename tracking in the **live_ftpsam_bfids** DB. This soft deletes the corresponding FTP MSP BFID daemon records without causing a recall.

A 'oncer' that was used twice

- The 800TB SAM /projects online filesystem died during /massdata migration, and **samfsck** couldn't fix it.
- Large impact on users as this data was supposed always to be online – the archive copies were for disaster recovery rather than HSM as such.
- Same approach was used to bring it over to DMF in order determined by triage of user projects.
- Total data moved with these scripts was probably on order of 1.5PB.

Future directions

- **mdss** command enhanced with daemon rec/chunk/path database to deliver similar function as CSIRO have with their **dmget** wrapper. Aim is to improve fairness w.r.t:
  - # concurrent tape mounts for one **mdss get**
  - latency on other accesses to VSNs satisfying an **mdss get**.

Thanks for listening

Questions?