NCI
NATIONAL COMPUTATIONAL INFRASTRUCTURE

Moving 6PB of data 2.5m West

- NCI's migration from LTO-5 to TS1150

Grant Ward
HSM Systems Administrator
National Computational Infrastructure

2 Spectra T950 libraries with many LTO-5 drives

- 5.4PB on massdata (1PB cache, 80% HWM)
- 72 million DUL+OFL files
- occupying ~3300 tapes in each library
- ~100 free tapes, limited empty chambers for more

2 Spectra T950 libraries with TS1150 drives

- 8 frames each so much greater tape count available
- for Lustre HSM project
- JD media - 10TB raw
- JC media - 7TB raw (upformatted from TS1140 4TB)

Large number of tapes marked HOA since April 2015 and must stay like that.

Dilemma

- buy more old tech tapes (and more chamber licences)
  - not many more vacant chambers left
- or share TS library (may require purchase of extra TS1150 drives – very expensive). Has prerequisites :
  - DMF upgrade (currently on SLES 11 SP1 + ISSP 2.5)
  - transfer of library partition licences

Upgraded to SLES 11 SP3 + ISSP 3.3 during NCI's quarterly maintenance on 1 July.

- 2 DMF HA servers, 4 data movers, 5 edge servers

Tested operation of TS1150 library successfully

- Pitfalls - partition numbers can change and therefore SCSI path to library controller, so choose wisely

Configured new massdata migrations to go to TS libraries

- free tape count pressure finally released - phew !!

What about existing data ?

No decision about future of existing data at this stage

- could quite happily run as a hybrid environment

But what would be the best way to migrate the LTO-5 data should we choose/need to ?

Using dmmove seemed to be an essential ingredient.
Could be as simple as

```
find . –type f | dmmove -p n_mass_j,h_mass_j
```

Though lack of `-0` option (to work with `find –print0`) will mean incomplete coverage.

Regardless of migration strategy

- existing DUL files can be migrated without recall

```
dmfind . –state DUL | dmmove –p n_mass_j,h_mass_j
```

- reaching HWM would trigger loss of 150TB of DUL data (and a need to recall later if migration is performed)

HWM approached in mid September and approval was granted to migrate all the DUL files from LTO-5 to TS.

Some files not migrated because of issue raised on previous slide.

NB. Support for `–print0` input has been added to `dmmove` for DMF 6.6 release since this talk was presented. Thanks!

Use previous command with 0FL instead ?

Might work, but

- would *dmmove* cope with 60m pathnames ?
- would it sort the recalls by tape ?
- would it run parallel recalls and writes ?

Perhaps break it down and run separate sequence for each top level directory (project).

- better, but some projects are very big and others almost empty
- also would likely result in each LTO-5 tape being mounted more than once

A more controlled approach was needed, and processing by volume seemed more appropriate.

We can easily obtain list of BFIDs on each volume, and then use dmmove's -b option, avoiding lack of -0 option.

```
for v in $(dmvoladm -q -c “list c_mass_1” | tail +4 | colrm 7) ; do
    dmcatadm -q -c “list vsn=$v” | tail +4 | colrm 25 | \
    dmmove -b n_mass_j,h_mass_j
done
```

But dmmove has drawbacks

- unclear how parallel it would be (testing required)
- our MOVEFS could really only spare 1TB (”-s 1TB”) which would likely slow the process down and reduce total throughput
- could create a new MOVEFS to address this ?
- or use our large cache (which has lots of spindles) with -f option ?

Bypass MOVEFS ?

Large cache

- can likely outperform our MOVEFS
- could allow recall of more OFL files
  - should ensure the backup queue always has work

Could recall OFL files, unmanage them, write them to new libraries releasing blocks to restore capacity in cache.

```
dmfind dir -state OFL -print0 | tee dir.OFL | dmunput -0 -w
cat dir.OFL | dmput -0 -r [-w]
```

Issues with previous method

Would mostly work, but

- multiple mounts of tapes will occur
- some projects are larger than working space available
- doesn't differentiate between VGs the OFL file is stored on (recalling OFLs already on tapes in the new libraries would be a really bad thing)

A mechanism that processes per LTO-5 volume still seems preferable

- efficient on mounts
- reduce seeking time
- only processes LTO-5 based data

Ideally would like to combine the `dmcatadm` and `dmunput`, `dmput` mechanisms. However this isn't possible because `dmunput` can't process BFIDs (only paths and fhandles).

Need a mechanism to convert BFIDs from `dmcatadm` to paths and/or fhandles.

- `dmattr` ? No, only takes paths as input
- data is available in `fhandle2bfid+path` (from `dmscanfs`) but this is a 14GB file so `grep` and `awk` are out of the question

fhandles are a representation of filesystem and inode

- they don't change when a file is released from DMF management (this just invalidates its BFID)
- this makes them preferable than path (eliminating need for NULL terminated lists of paths)

Conceptually this now becomes

```
dmcatadm -q -c "list vsn=$v and chunkoffset = 0"
| tail +4 | colrm 25 | tee $v.bfids |
bfid2fhandle | tee $v.fhandles | dmunput -h –w
cat $v.fhandles | dmput -r -h [-w]
```

This has the elegance and efficiency I was looking for, and looks easy to parallelise to boot !!

And now the migration really starts

Run on 100 input volumes at a time (which equated to 100-150TB of "`dataleft`") fed to "`parallel -j <n>`"

Tuned number of parallel readers

- don't recall too much to cause HWM to be breached
- but need backup queue to always have work

Managed to migrate 120TB per day to 5 TS1150 drives in each library (~300MB/s per drive sustained 24x7).

Never breached HWM (but must have been awfully close a few times).

Adapted a perl script from Jason Ozolins (SGI) to convert targeted BFIDs to fhandles.

Reads a list of BFIDs into a hash, then reads the `fhandle2bfid+path` file. For each entry whose BFID exists in our hash it prints out the corresponding fhandle.

bfid2fhandle.pl

```
# Conceptual version of script.  Boring bits omitted for brevity.

# Mapping file from fhandle to BFID and path.
my $fh2bfid = 'fhandle2bfid+path';
open(M, '<', $fh2bfid) or die "Couldn't open fhandle to BFID file '$fh2bfid': $!";
# BFID hash for targeted BFIDs
my %bh;

while (<>) {
  chomp;
  $bh{$_} = 1;  # create hash of BFIDs
}

while (<M>) {  # scan fhandle file and output it when the line has a bfid we want
  chomp;
  my @F = split(/ /, $_, 3);  # split to get second field (BFID)
# Is this BFID in our list ? If so, print out fhandle.
# BFIDs for soft deleted files will not exist in mapping file so
# no entry will be printed
  printf "%s\n", $F[0] if (exists $bh{$F[1]});
}
```

Checking status of processed volumes, their "`dataleft`" was still non-zero. Why?

- `dmunput` only soft deletes the data so the BFID still exists in the database (cf. `dmmove` which hard deletes)

So once the backup queue had processed the recalled data, a manual hard delete on the migrated BFIDs was needed

- `dmhdelete -o 1m -b $v.bfids`

`dmhdelete` took a long time to run

- best to gather a decent number of BFIDs (but not too many) - a few million seemed to work well

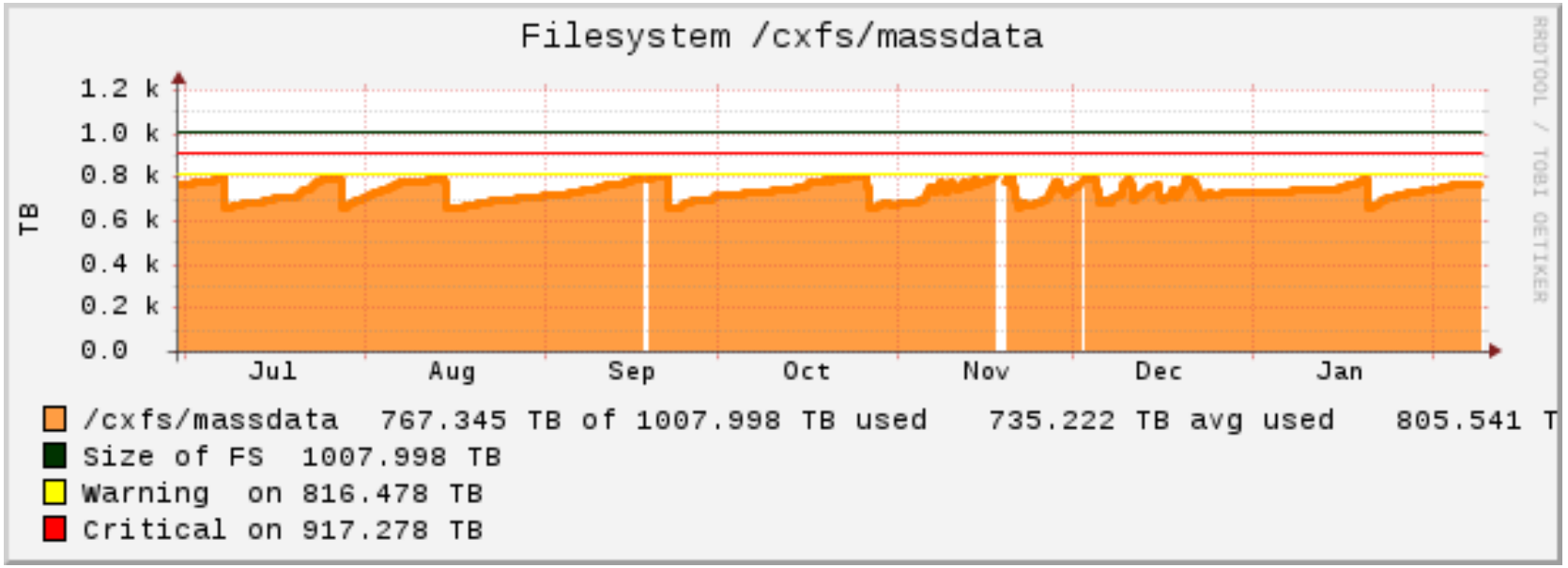
Had to change nightly hard delete task in `dmf.conf` to weekly (weekends) so that I could run my manual ones through the weekdays.

All data migrated in under 8 weeks (includes new data that users were uploading).

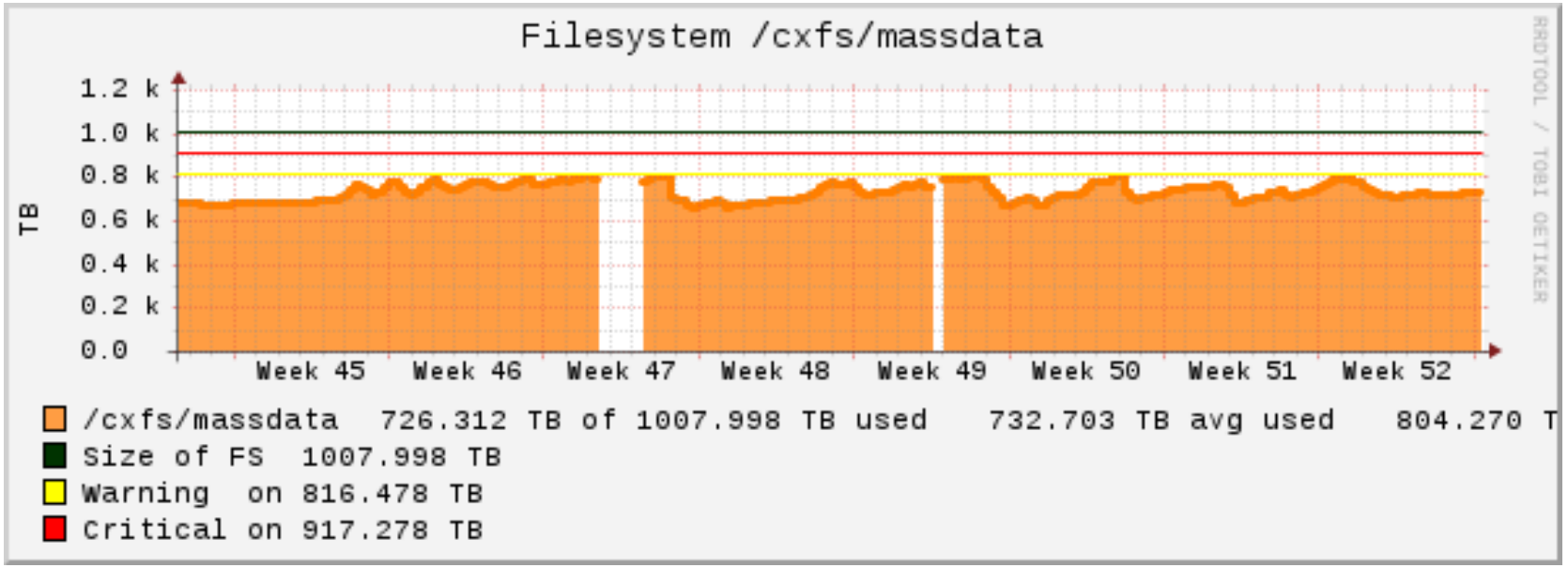
Size had increased to 6.3PB but was now on only 450 JD tapes (50 TeraPacks, or less than the base frame !!).

Issues

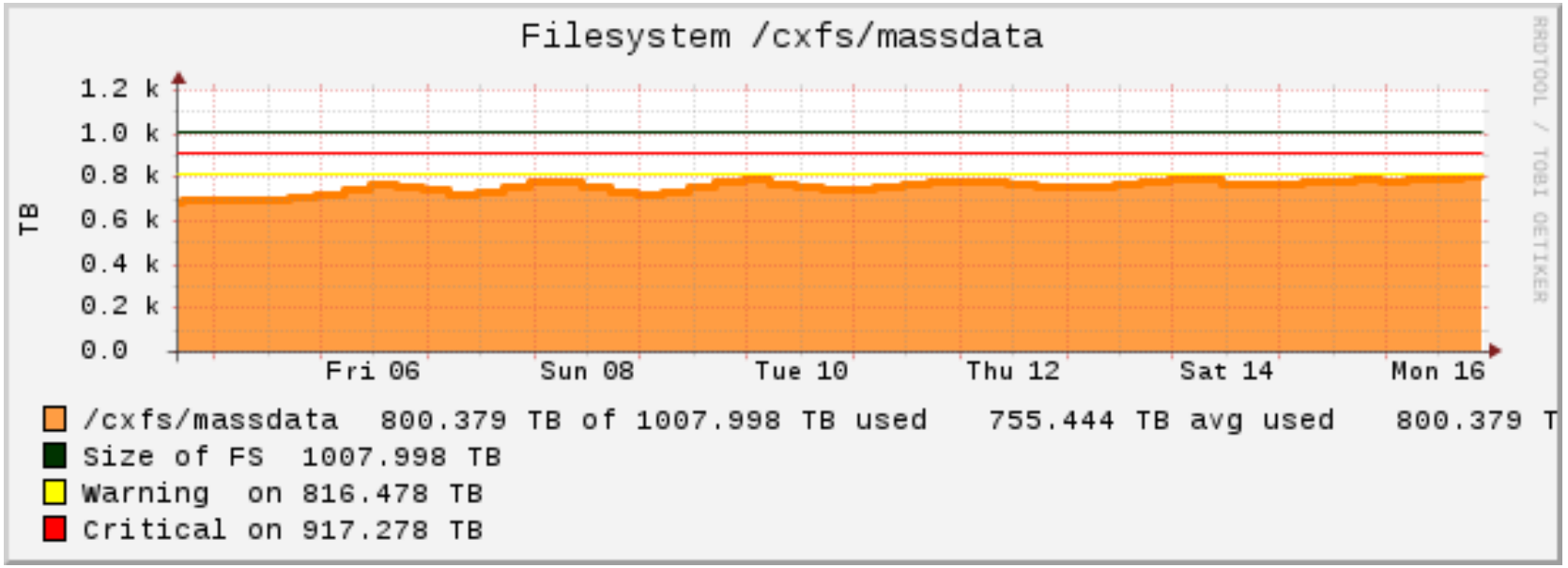
- recalls for files in TS VGs required manual intervention due to *dmatwc*'s not releasing TS drives whilst there was still work in the backup queue
- inefficient recall workloads impact more than our LTO world due to fewer drives being available (Malcolm's monitoring will play a role here)



Cache usage since ISSP upgrade



Cache usage during migration



Cache usage during early stages of migration

Before

After

Questions ?