Spectrum Scale Data Migration With AFM

A Cautionary Tale

Bob Oesterlin
Sr Principal Storage Engineer
Nuance Communications

Nuance - Global technology reach

Established global technology leadership across multiple domains and languages

2,500
voice and language scientists, developers, and engineers

~4000
patents and applications

80
languages across voice, NLU, and text

10,000
employees worldwide

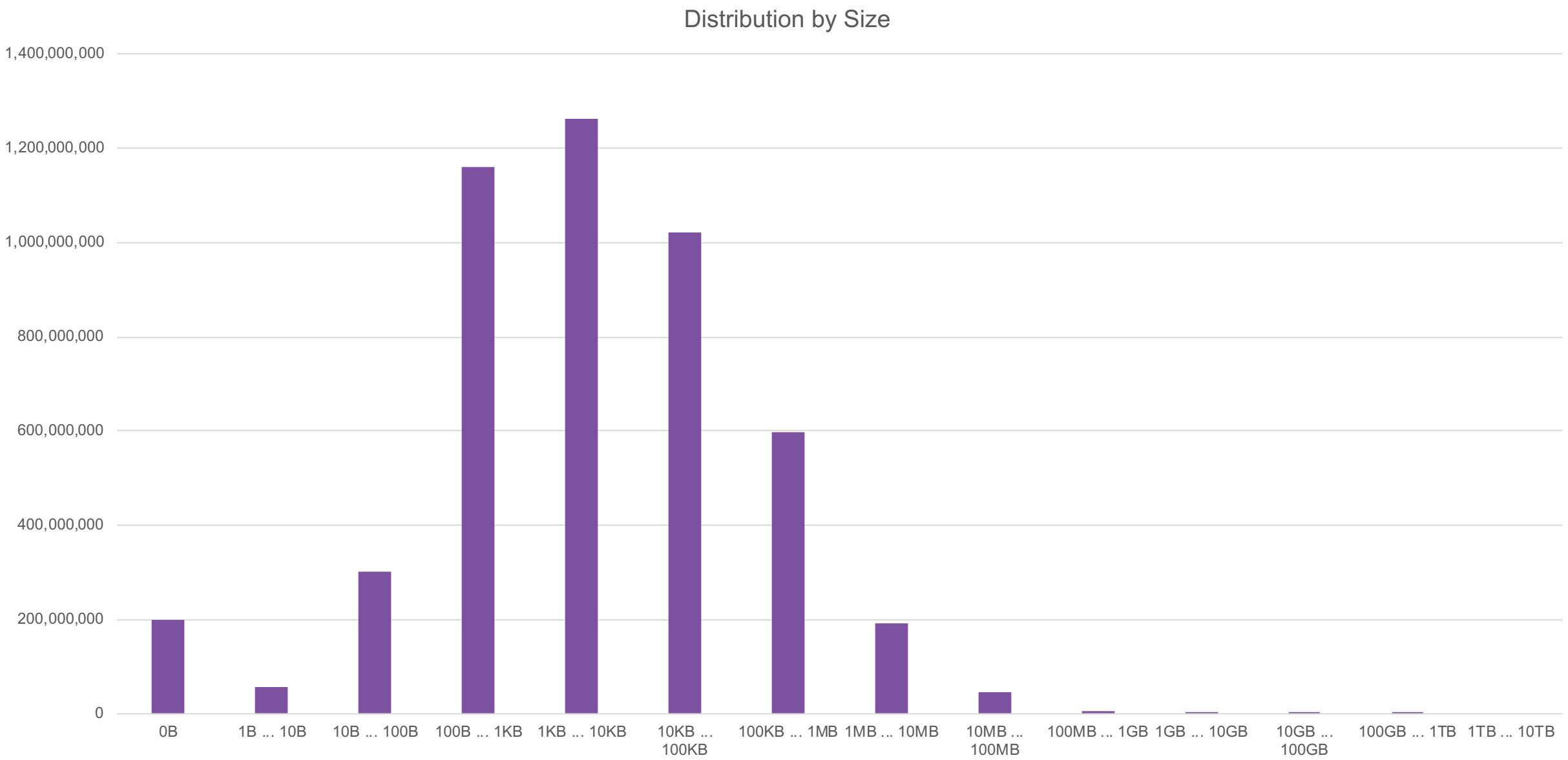
48
countries

Nuance HPC Environment

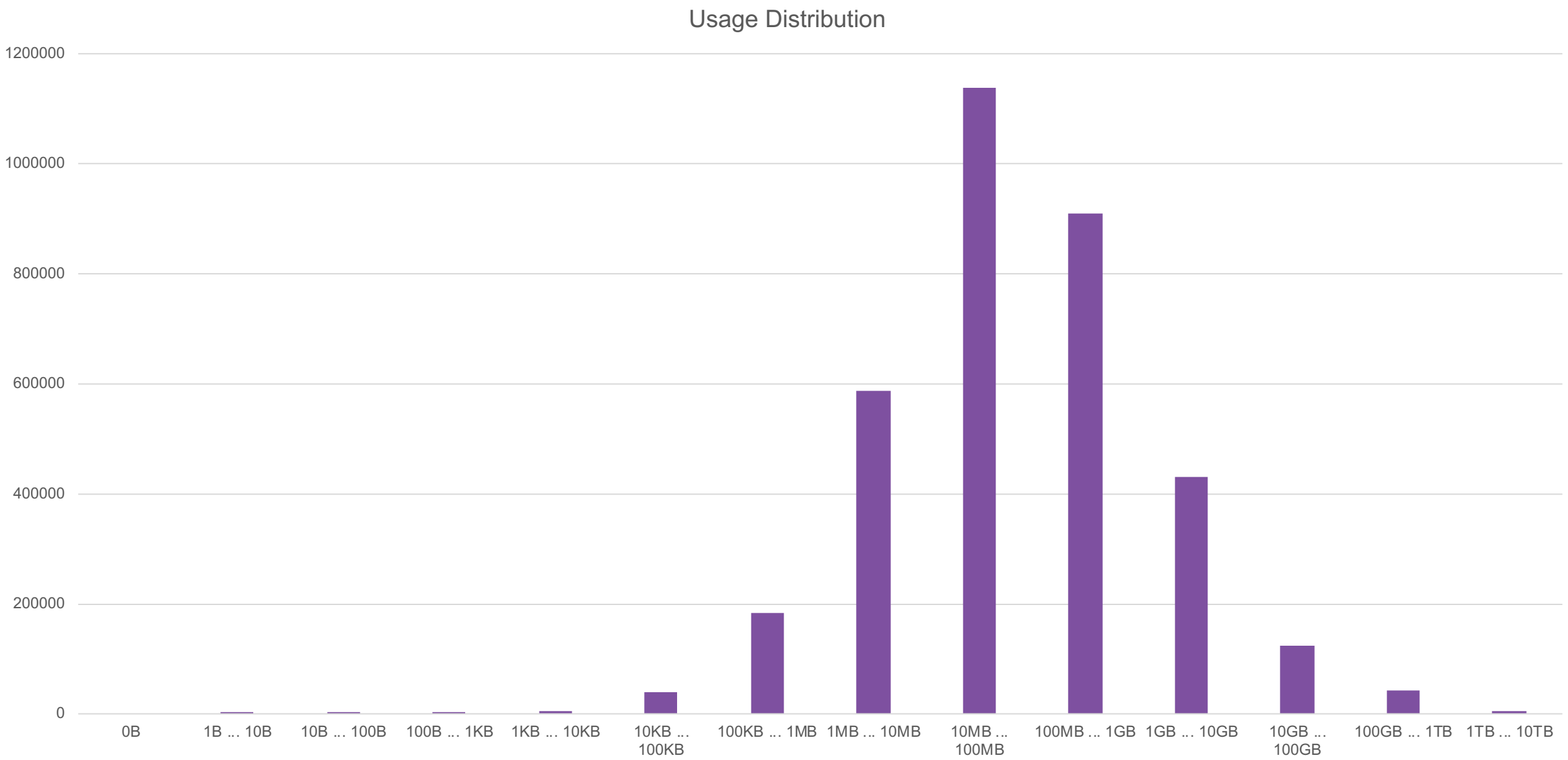
Nuance HPC Environment

- Compute
  - Approximately 1600 Nodes (x86 Linux 6.8, 7.5+
  - 200+ GPU Nodes (mix of types, x86 Linux)
  - 2 main compute “grids” – UGE Job Scheduler
  - 10G Ethernet
- Storage
  - 3 Spectrum Scale Clusters
  - SAN based storage
  - “Roll your own” NSD servers
  - CEPH (Object and File)
- Spectrum Scale
  - Clusters/Filesystems are 7+ years old (started at GPFS 3.2)
    - Current clusters are all 4.2.3-17 (CentOS 6.X dependency on some clients)
  - 2+ PB SAN storage (20+ file systems)
  - 3PB ESS GL4 5.2.3.4-1 (Scale 5.0.3.2)
  - Approx 4 billion files (avg file size <10k)

HPC File Distribution



HPC File Distribution



AFM Migration

Migration Goals/Constraints

- Goals
  - Retire legacy SAN storage and servers
  - Reduce datacenter space
  - Reduce power, cooling
  - Reduce number of file systems, move to filesets
  - Encrypt data (Spectrum Scale + SKLM)
- Constraints
  - Move off high speed SAN storage by 7/1 (did not meet this, but not due to AFM)
  - Move off slower/older SAN storage by YE 2019 (may make it?)
  - Minimal user disruption (data unavailable to users)
- How can we do this?
  - rsync (or a variation of this)
  - AFM

Target Environment

- ESS GL4, 2PB (now 3PB)
  - Initially running ESS 5.3.2-1. (Scale 5.0.2.2)
  - 2 file systems: 600TB, 1100TB
    - Needed to be Scale 4.2.3 compatible
  - Spectrum Scale Encryption (2 SKLM servers, primary/backup)
  - IO nodes 4x40GBE bond

- AFM
  - 4 gateway nodes
  - 128gb/24 core 10GE
  - Scale 5.0.3.3

AFM Initial Setup

- Designate AFM Gateway nodes
  - `mmchnode --gateway -N Node1,Node2`

- Create AFM relationship (Cache fileset)
  - `mmcrfileset fs1 fileset1 -p afmtarget=gpfs:///gpfs/source-gpfs1 -p afmmode=ro --inode-space=new`
  - `mmlinkfileset fs1 fileset1 -J /gpfs/fs1/fileset1`

AFM behavior and setup considerations

- Leave the AFM cache in RO mode while syncing with home fileset
- Prefetch – how much, metadata only
- Gateway nodes are the bottleneck for your data transfer
  - Network memory, threads
- Consider upgrading the AFM gateway nodes to the latest 5.0.3 release (5.0.3.3 as of this presentation)
- Consider having at least one extra AFM node for prefetch operations apart from dedicated gateways

AFM Data Migration

- AFM will “pull over” data when needed unless prefetched – Don’t need to prefetch

- Prefetch data
  - `mmafmctl fs1 prefetch -j fileset1 --local --gateway afmgw03 --directory /gpfs/fs1/fileset1`
  - This command walks the entire directory tree and prefetches all files – lengthy process
  - Reports progress

- Prefetch metadata only
  - `mafmctl fs1 prefetch -j fileset1 --gateway afmgw03 --local  --readdir-only --directory /gpfs/fs1/fileset1`
  - This command walks the entire directory tree and prefetches metadata only
  - DOES NOT report progress

Moving users over to the new fileset

- Many possible ways to do this, this is an example only – will require at least a brief outage
- On the source (AFM Home) cluster
  - Unmount existing file system on all node (don't forget about the remote mounts!)
  - Consider changing mount point of old file system to RO and changing the mount point
    - `mmchfs filesystem1 –T /gpfs/filesystem1_old –o ro`
  - Replace existing mount point with sym-link to new fileset/cluster
    - `ln –s /gpfs/filesystem1 -> /gpfs/new_fluster/fs1/fileset1`
- On the target (AFM Cache) cluster
  - Shutdown AFM: `mmafmctl fs1 stop –j fileset1 (may not be needed)`
  - Unlink the fileset: `mmunlinkfileset fs1 fileset1`
  - Change the AFM mode to local-updates: `mmchfileset fs1 fileset1 –p afmMode=lu`
  - Relink fileset: `mmlinkfileset fs1 fileset1 –J /gpfs/fs1/fileset1`

What happens now?

- Users now are accessing data on the new fileset/filesystem
  - Any new files/changes occur on the new fileset only, old file system is a reference
  - Do a final metadata/data prefetch to pull over any data not already in the “cache”
  - Check for uncached files using policy scan

- Policy scan to check for uncached files
  - AFM runs this if you ever disable the AFM relationship
  - Can take a LONG time to run and needs sufficient space to hold lists of files
  - Simply telling AFM you want to disable the relationship will trigger a check for uncached files
  - Running policy scan is a simpler process

Example policy scan

— `mmapplypolicy /gpfs/fs1/fileset1 --scope fileset -P uncached.pol -L 1 -N afmgw01,afmgw02,afmgw03,afmgw04 -I defer -f /gpfs/fs2/temp/fileset1 --iscan-by-number`

```
define([vc],[CASE WHEN ($1) IS NULL THEN '-1' ELSE VARCHAR(($1)) END])
define([val],[vc(($1)) || ' '])
define([regular],[((NAME IS NOT NULL) AND (BLOCKSIZE!=0))])
define([orphan],[((NAME IS NULL) OR (BLOCKSIZE=0))])
RULE EXTERNAL LIST 'dirs' ESCAPE '%'
RULE 'dirsRule' LIST 'dirs' DIRECTORIES_PLUS FOR FILESET (‘fileset1') WHERE (regular) AND
((NOT RegEx(misc_attributes,['^[^D]*$|z|j'])) OR (NOT RegEx(misc_attributes,['^[^D]*$|z|u'])))
RULE EXTERNAL LIST 'files' ESCAPE '%'
RULE 'filesRule' LIST 'files' DIRECTORIES_PLUS FOR FILESET (‘fileset1') WHERE (regular) AND
(NOT RegEx(misc_attributes,['[DXazu]']) AND SUBSTR(mode,1,1) NOT IN ( 'b', 'c', 'p', 's'))
RULE EXTERNAL LIST 'orphans' ESCAPE '%'
RULE 'orphansRule' LIST 'orphans' directories_plus SHOW('i='||varchar(inode)||' '||varchar(mode)||'
nlink='||varchar(NLINK)) WHERE (orphan)
```


Things to watch out for

- Beware high-IO operations run out of AFM cache – Refresh waiters
  - Waiting 0.7120 sec since 12:48:28, monitored, thread 48270 PCacheRpcMsgHandlerThread: on ThCond 0x7F925C000E10 (PCacheMsgCondVar), reason 'Wait for refresh to complete'
- Consider disabling refresh after you are sure all files/metadata is caches from the source
  - CAUTION; This stops AFM from checking for changes on the source!
  - `mmchfileset fs1 fileset1 -p afmDirLookupRefreshInterval=disable -p afmDirOpenRefreshInterval=disable -p afmFileOpenRefreshInterval=disable -p afmFileLookupRefreshInterval=disable`
- Watch out for "dirty" files and directories – AFM will not refresh these in LU mode
  - Fixed in 5.0.4.X release with "afmReaddirOnce" option – or ask for efix

```
tspcacheutil /gpfs/fs1/fileset1/data/users/
inode: ino=4504963 gen=196387551 uid=0 gid=5474 size=32768 mode=0200042775 nlink=289
       ctime=1569244778.793916000 mtime=1569244778.774146000
       cached 1  hasState 1  local 0  complete 1
       create 0  setattr  0  dirty 1
pcache: parent ino=4504960 foldval=0x93F05ECA nlink=2
remote: ino=143360 size=16384 nlink=281 fhsize=24 version=0
        ctime=1567686147.115945000 mtime=1567686147.115932000
```


So how well did it work?

- Overall migration process went smoothly
- Less overall user downtime than rsync-based process (well mostly)
- Did discover some AFM bugs/limitation early, fixed quickly by development
  - Shout out to Venkateswara (Venkat) Puvvada – AFM Development lead
- Performance on high-IO cases is poor, needed to disable refresh intervals
  - Some undocumented options that may help, need to contact development
- If your applications are using “mmap” to read files (“git” does this a lot) then make sure the clients are at level 4.3.2.17 or 5.0.3.2 or later – or uncached read operations will return incorrect data.
  - APAR https://www-01.ibm.com/support/docview.wss?uid=isg1IJ17053

