Spectrum Scale - Doing more with less

September 2019

Mark Roberts
High Performance Computing
Physics Department

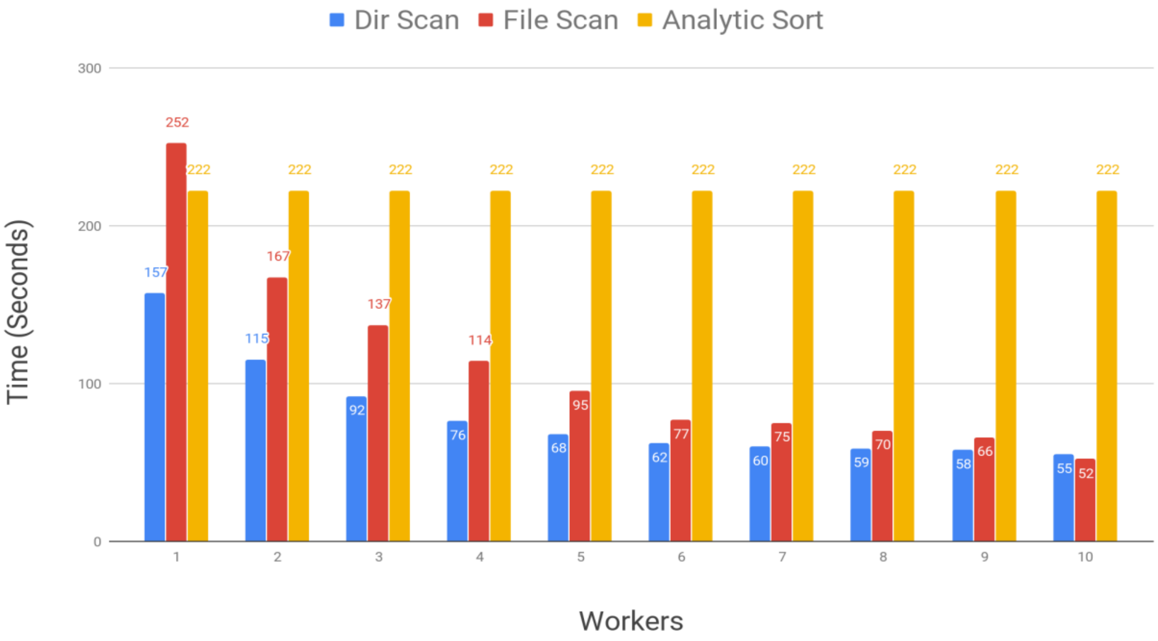
Introduction

- Need to do more with less
  - Data is increasing but budgets are reducing
  - Data capacity licensing
- Upcoming storage procurement
  - SC2019 Initial RFI document
  - Q1/2020 ITT Issued
  - Q3/2020 Contract placement
  - Q1/2021 SAT
  - Q2/2021 Data transfer completion

ILM Policy Engine - Phases

- Split into three phases
  - Directory scan
  - File scan
  - ***Analytical sort***
- Analytical sort
  - Single threaded and runs on calling node
    - Same time regardless of workers/threads
  - Possibly a long process time
    - Time is dictated by number of candidates

ILM Policy Engine – Phase Times



- ESS Nodes 5.0.3
- 75.37mil inodes
  - Dirs - 467k
  - Files - 75.25m
- 1 node scan
  - Dirs - 24.8%
  - Files - 39.9%
  - Sort - 35.3%
- 10 node scan
  - Dirs - 16.7%
  - Files - 15.8%
  - Sort - 67.5%

ILM Policy Command – Undocumented flag

```
/usr/lpp/mmfs/bin/mmapplypolicy fshome
    -I buckets
    -P /fsgroup/ILM/ilm-policy.txt
    -g /fshome/ilm-tmp
    -f /fsgroup/ilm-tmp/ILM
    -A <X> -a <X> -n <X> -m <X>
    -q --scope filesystem
    -N <nodes> 2>&1 | tee ilm-out-buckets.txt
```

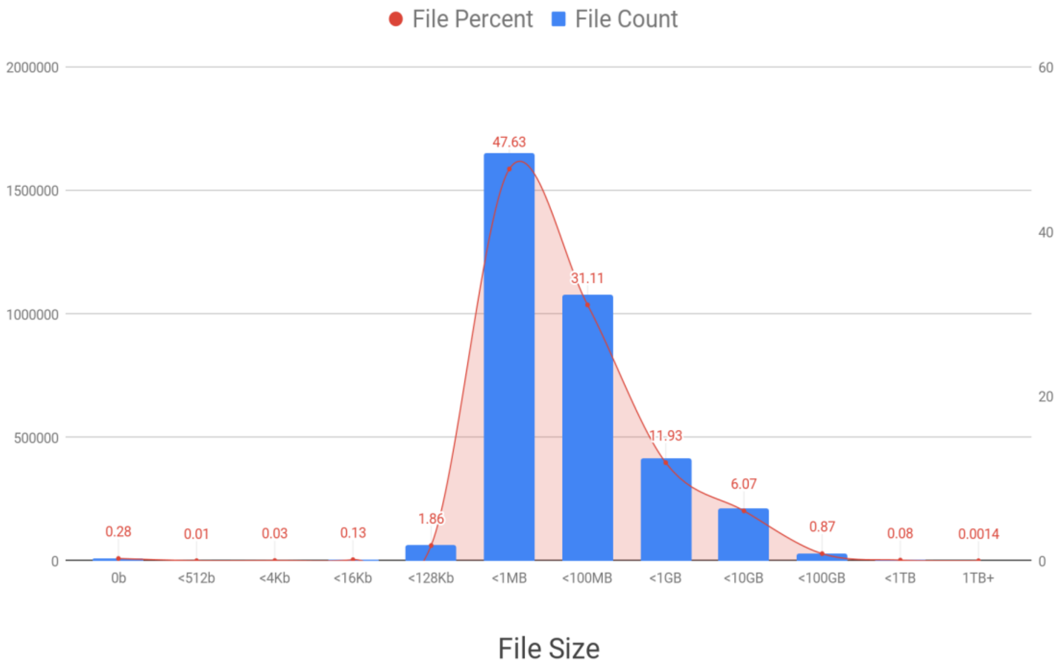
ILM Policy Command – “-l bucket” flag

- No aggregated output
  - Aggregate output files if needed
    - time cat /ptmp/bucket.* > ilm-buckets-all.txt
      - 16 seconds overhead
    - parallel cat command
      - Reduce time for large number of bucket files
- 10 nodes (including cat process)
  - AWE 123 secs from 329 secs (75.3mil inodes)
  - CSCS 121 secs from 378 secs (117mil inodes)

Why compression ?

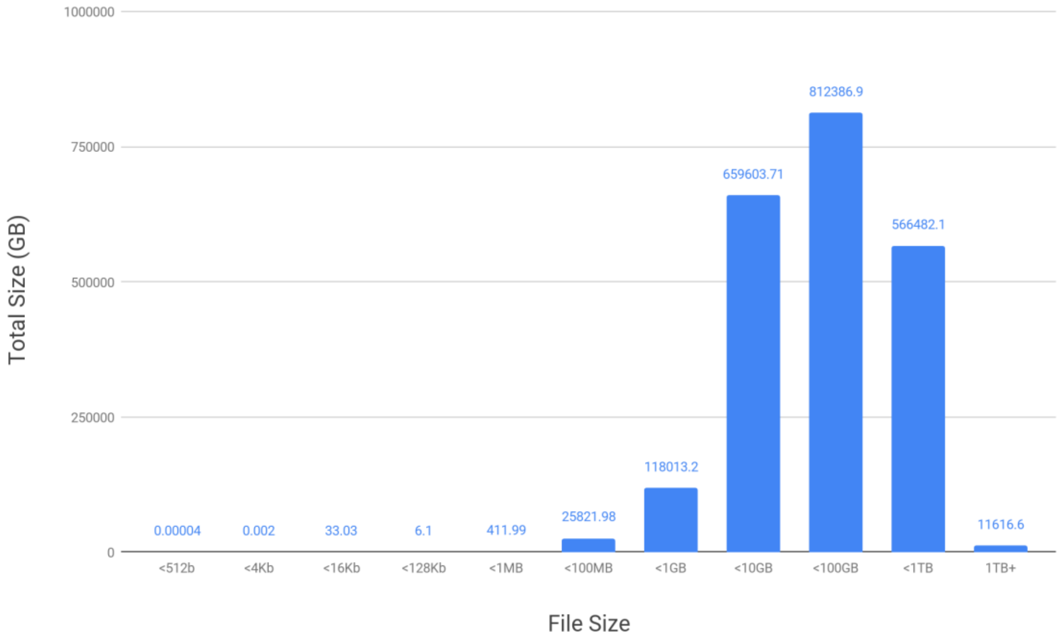
- Started evaluating after ESS 5.0.3 upgrade
- Looking at all our major data types
- First target - h5 files
  - Main workhorse format for physics codes
    - Restarts
    - Checkpoints
    - Visualisation
- Next targets … tar, text, log files etc
- Physics codes to use EAs – tag if data is compressible

Compression – h5 file count



- 4PB file system
- 75.37 million inodes total
- H5 file count
  - 2.2PB
  - 3.5 million inodes
  - 4.6% of total files

Compression – h5 file capacity



- h5 data -
  - Mix of 2D and 3D
- Majority not -
  - modified after 2 weeks
  - accessed after 3 months

Compression

File	Original Size (GB)	Lz4 Size (GB)	Z Size (GB)	Size Ratio (Z:Lz4)	Lz4 Time (Secs)	Z Time (Secs)	Time Ratio (Z:Lz4)
huge_3dEUL.h5	18	11	11	1	225.52	540.62	2.4
huge_3dALE.h5	16	12	12	0.92	215.29	862.02	4.0
long.log	20	7.5	4.2	0.56	674.17	1150.17	1.71

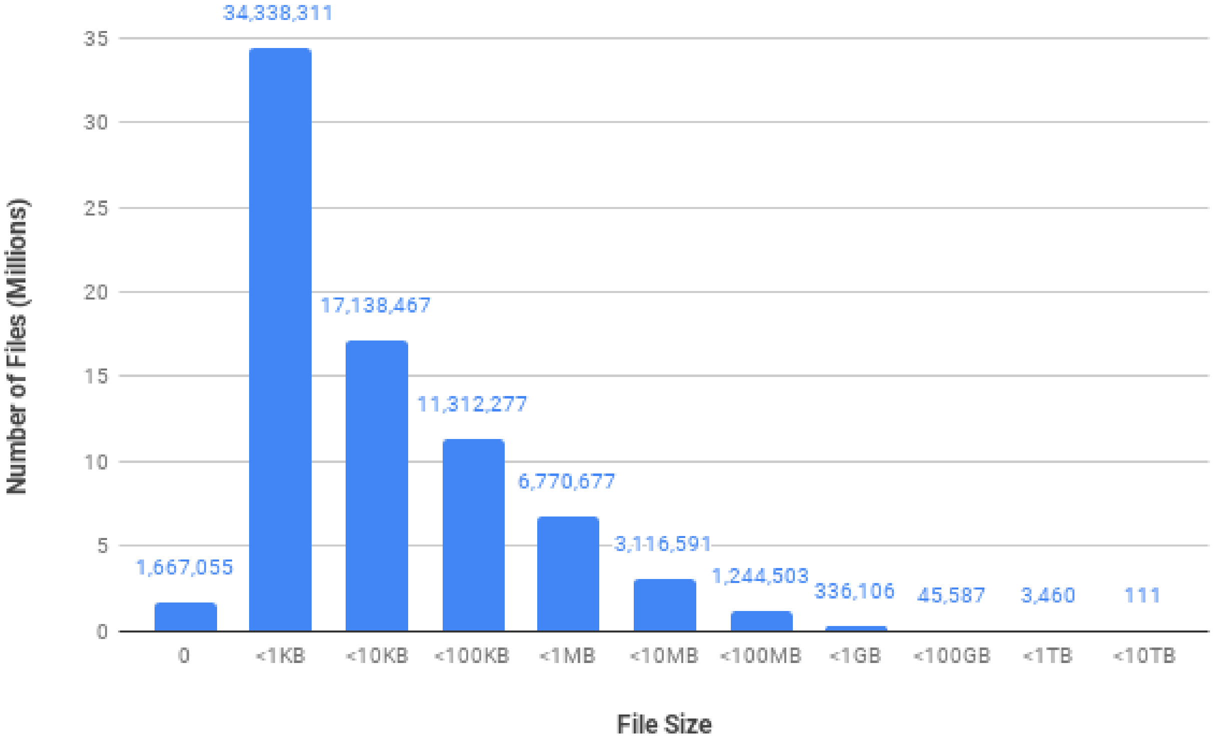
Compression - Summary

- Space reduction of 15-40% in lz4 tests
  - Estimated 300-800TB saving on 2.2PB
  - Compressed data is faster to read … fewer disk blocks
- Data is decompressed for backup/migration
  - Avoid by targeting old unmodified resident data
- Save space on “cold” data without resorting to HSM
  - Users dislike waiting for tape recall
  - Less data to recall when migrating to next storage platform
- Why not in-situ h5 compression ?

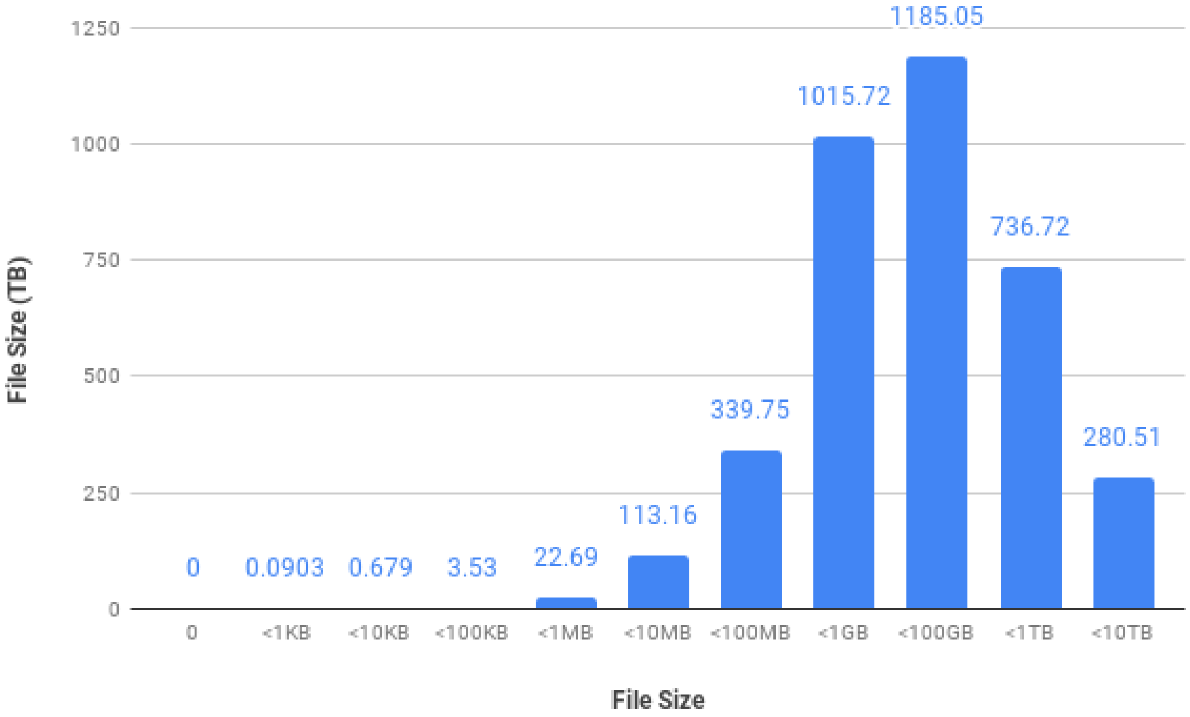
Duplicate File Analysis

- Identify duplicates
  - Reduce space
  - Reduce inode wastage
  - Reduce unnecessary tape usage
- Why ?
  - No deduplication (Spectrum Scale)
  - Users copy files across file system boundaries
  - We suspect it’s an issue … let’s find out !

File Size vs File Count



File Size vs File Capacity



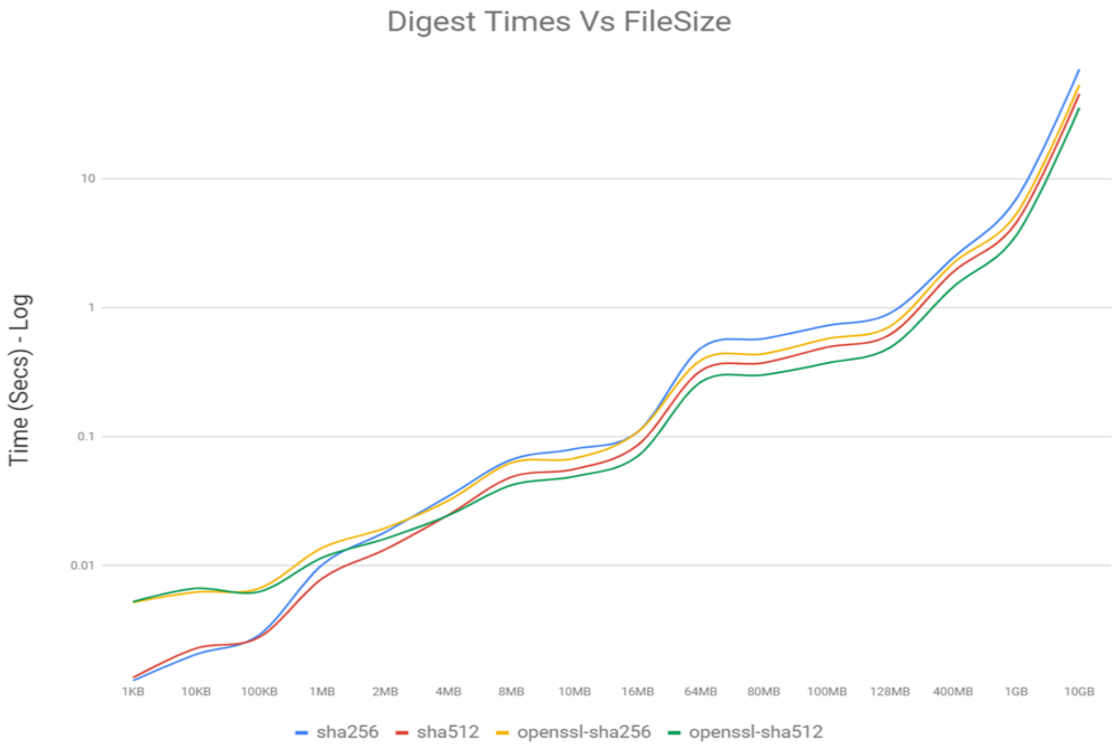
Duplicate File Analysis

- 99.49% of files < 100MB (480TB)
- 3.56PB data in 1.63 million files
- Only include resident/pre-migrated files
  - Do not include :
    - Zero sized Files
    - Ill-replicated files ?
    - Migrated files
    - Compressed files
      - Ideally decompress, compute and recompress at later date
    - mmbackup/snapshot areas

Duplicate File Analysis

- For file range >0 and <10MB
  - Accounts for 93.61% (72.67 million files) in 140.15TB
  - Check-summing serially could result in significant time
- If parallelised using multiple files over multiple cores
  - With 28 cores reduced total runtime of <12 hours could be achieved
- For larger files use first and last “X” MB of file
  - Reduces checksum time on huge files
  - If these match another file then –
    - Check file size
    - Break entire file up into chunks
    - Checksum each chunk
    - Create hash of checksums and compare

Duplicate File Analysis – Checksum Times



Duplicate File Analysis

- Varying checksum efficiency
  - Sha512 has good speed across all file sizes
  - OpenSSL-512 was faster at file size >10MB
    - For 10GB file size - OpenSSL-sha512 (35secs) versus sha512 (46secs)
- Store checksum information in EAs
  - `system.cksum=sha512`
  - `system.cksum.value=5c46b17ae4aec0bfece7855d52ddfbb2`
  - `system.cksum.epoch=<`*`epoch_date_of_checksum`*`>`
  - `system.cksum.fullcheck=1`
- Resolve atime effects when check-summing
  - Either store atime and restore after or mount with no atime

Future Plans

- Apply mmap fix (when available)
  - Used in our new ILM analysis workflow
- Auditing
  - Early testing
  - Not full coverage
- Clustered Watch Folders
  - External Kafka sink cluster
- Efficient cluster copy
  - mpiFileUtils

Thank you

Wayne Sawdon (IBM)
Marc Kaplan (IBM)
John Lewars (IBM)

Nick Macey (AWE)
Stephen Rollinson (AWE)
Dario Petrusic (CSCS)