Faster Tree Walking in Lustre

LAD 2019

Nathan Rutman


nrutman@cray.com

Outline

- Why does it matter
- A deep investigation
- Are we focused on the right thing?
- How can we make it better

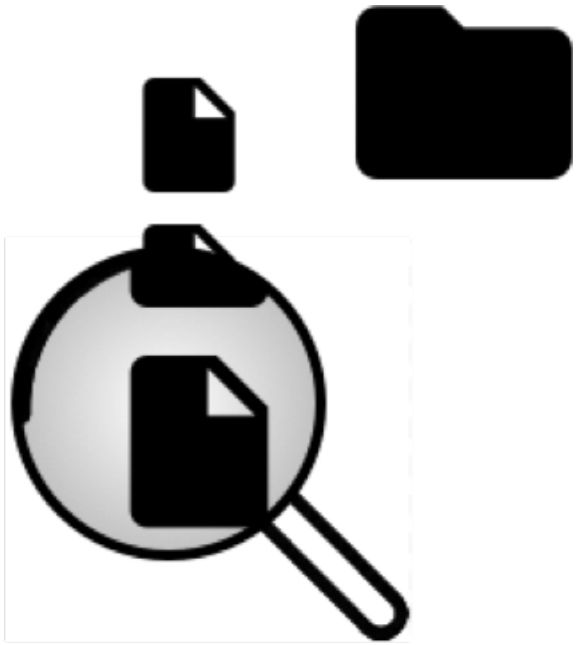
Why?

- Find
  - User search
  - Policies
  - Filesystem stats (eg size distributions)
- Populate DB – but DB really is just for faster find
- Changelogs are (historically) slow / incremental / buggy / incomplete
  - still need to stat, getxattr, fid2path
  - need to occasionally re-sync from tree
- ls –l -- maybe?
  - is it still too slow?
  - statahead 100K dir: 25 seconds down to 3 seconds

Tree traversal

- D stat dir (ldlm_ibits_enqueue->getattr)
- D stat dir? (ldlm_glimpse_enqueue)
- D opendir (ldlm_ibits_enqueue->open)
  - Dx readdir (mds_readpage)
    - Fxx stat dirent (ldlm_ibits_enqueue->getattr)
    - Fxx size osc (ldlm_glimpse_enqueue)
- D closedir (mds_close->close)
- Note that ordering above may change (e.g. open-read-close-stat)

- per Dir: 2x ibits_enqueue, glimpse, mds_readpage, mds_close
- per File: ibits_enqueue, glimpse on osc



mdtest

Pure stat; **no treewalk**, single client

mdtest -n 150000 -u -C -f=20 -R -d /lus/nathan/mdt1/02

	cached	drop mdc	drop osc	drop mdc+osc	drop mdc+ osc+mdt	drop mdc+ mdt
Directory stat	112,119	11931	127,902	11411	5222	5404
File stat	60,020	9526	9929	3854	3164	8003
File stat (ms)	16	105	101	259	316	124

- mdc ibits_enqueue time = osc ibits_enqueue time
- no statahead; sequential mdt rpc then osc rpc
- forcing server to stat from disk vs mem – not a big impact 16-18%
- mdtest does not measure the **larger** directory traversal times

fprof

Treewalk, 16 clients x 8 procs

10M empty dirs or files

	objects/s
Directory	30,545
File	217,139
1 file per dir	60,000

- stats are 7x faster than directory reads
- effective scanning rate is gated by directory reads
- file stats are almost free for small dirs
  (30k+30k in the same time as 30k)

Estimating times

File system objects: 147814 (133k Files + 14k Dirs)
Time elapsed: 9.686485334s
Index rate (obj/sec): 14781

mdc.testfs-MDT0001-mdc-ffff8a77f16a2000.stats=

	File/Dir	samples	units	min	max	sum	calc usec	avg	% of time
req_waittime		205258	[usec]	31	36014	22070734		107.53	100%
req_active		205258	[reqs]	1	17	539203		2.63	
ldlm_glimpse_enqueue	D	14255	[reqs]	1	1	14255			
ldlm_ibits_enqueue	F+2D	162211	[reqs]	1	1	162211	14935488	84.64	68%
mds_close	D	14396	[usec]	37	12192	792288	async	55.04	4%
mds_readpage	D	14396	[usec]	161	15890	6342958		440.61	29%
osc ldlm_glimpse_enqueue	F	54838	[reqs]	27	12107	2835723		51.71	0%

- No direct measure of ldlm* times; assume waittime=processing time and ldlm times are the remainder after mds* (maybe roughly right?)
  - 1/3 time for readdir
  - 2/3 time for getattr
- With dirs only, each readpage takes 4306us??

Caching effects on treewalk

single client

multiple processes

- *lctl set_param ldlm.namespaces.testfs-MDT0001-*.lru_size=clear*
  - *echo 3 > /proc/sys/vm/drop_caches* only has an impact on MDT
- Client caching can skip getattr, mds_readpage
  - client cached: 49,271 files/s
  - client MDC LRU clear: 14,781 files/s
  - client and server clear: 8,694 files/s
- Clearing osc locks forces ldlm_glimpse_enqueue – but doesn't affect rate much
  - clear osc locks, but NOT mdc: 36,953 files/s
  - no statahead (llite.testfs*.statahead_stats)
    - statahead is triggered (& useful) only for single-threaded process
  - DoM, and Lazy SoM will have no affect?
- Why is server caching helpful? MDT devices are capable of Miops
  - Latency, cache logic overhead?

MDTEST showed small effect of server cache (20%) vs 70% here – difference is readpage

many processes request stats in parallel

Layout info

- *ls –l* or I/O on a file forces the client to get the layout
- But it doesn't seem to be cached, at least not as an EA
- *getfattr lustre.lov* does cache it
- *lfs getstripe* costs 6 RPCs

```
[root@c-lmo216 ~]# ls -l /mnt/testfs/nathan/mdt1/test.txt
-rw-r--r-- 1 root root 45 Aug 27 10:29 /mnt/testfs/nathan/mdt1/test.txt
[root@c-lmo216 ~]# lctl get_param mdc.testfs-MDT0001*.stats
req_waittime               2 samples [usec] 349 446 795 320717
req_active                 2 samples [reqs] 1 1 2 2
ldlm_ibits_enqueue         2 samples [reqs] 1 1 2 2
[root@c-lmo216 ~]# getfattr -R -d -m 'trusted.lov' -P -e hex /mnt/testfs/nathan/mdt1/test.txt
trusted.lov=0xd00bd10b…

[root@c-lmo216 ~]# lctl get_param mdc.testfs-MDT0001*.stats
req_waittime               3 samples [usec] 349 516 1311 586973
req_active                 3 samples [reqs] 1 1 3 3
ldlm_ibits_enqueue         3 samples [reqs] 1 1 3 3
```

- LU-11367 llapi_get_lum_file_fd() should get stat+layout in single RPC

Improvement ideas – low impact

Stats are already fast! Not much to be gained.

- Improve client statahead
  - long-lived threads
  - aggressive statahead at opendir time
- Lsom to elminate OST glimpse LU-11554
- DoM to eliminate OST glimpse
- Layout should be free with stat LU-11367

Improvement ideas – medium impact

Warm the MDS cache

- Read the directory into server memory after opendir, assuming read is coming
- Stat all files in a dir after readdir to get them into cache (server statahead)
- Read all subdirectories on opendir/readdir patterns

Save a lookup

- Client creates dentries from readdir for local lookup
  - see statahead HLD section 3.5, or LU-31 HLD “List Lock” discussion

Improvement ideas – high impact

unlock server IOPS with complex RPCs

- readdir+ stats files and returns all info with readdir LU-23
- opendir returns first readdir page in open response
  - like “short read”
  - increases latency of open, but saves readdir roundtrip
- opendir with readdir+/statx and create dentries+inodes
  - return all file MD at opendir
  - populate client dentry and inode caches with this info
  - take directory update lock to insure dir doesn’t change
  - no locks on any of the dentries or inodes; ok for statx with AT_STATX_DONT_SYNC

A Hint

What can I do today with this investigation and potential enhancements?

- Tree walking is slow, but stats are fast
- So use stats without tree walking
  - File/dir creates, deletes, renames affect mtime of parent dir
  - Dirs are 1/10 of files
  - Check dir mtimes against a database
  - Shows you which dirs to re-check for updates
  - Doesn’t see file mtime-only changes

THANK YOU

Questions?