Hewlett Packard
Enterprise

GETTING THE MOST OUT OF MIRROR AND MIGRATE

Nathan Rutman

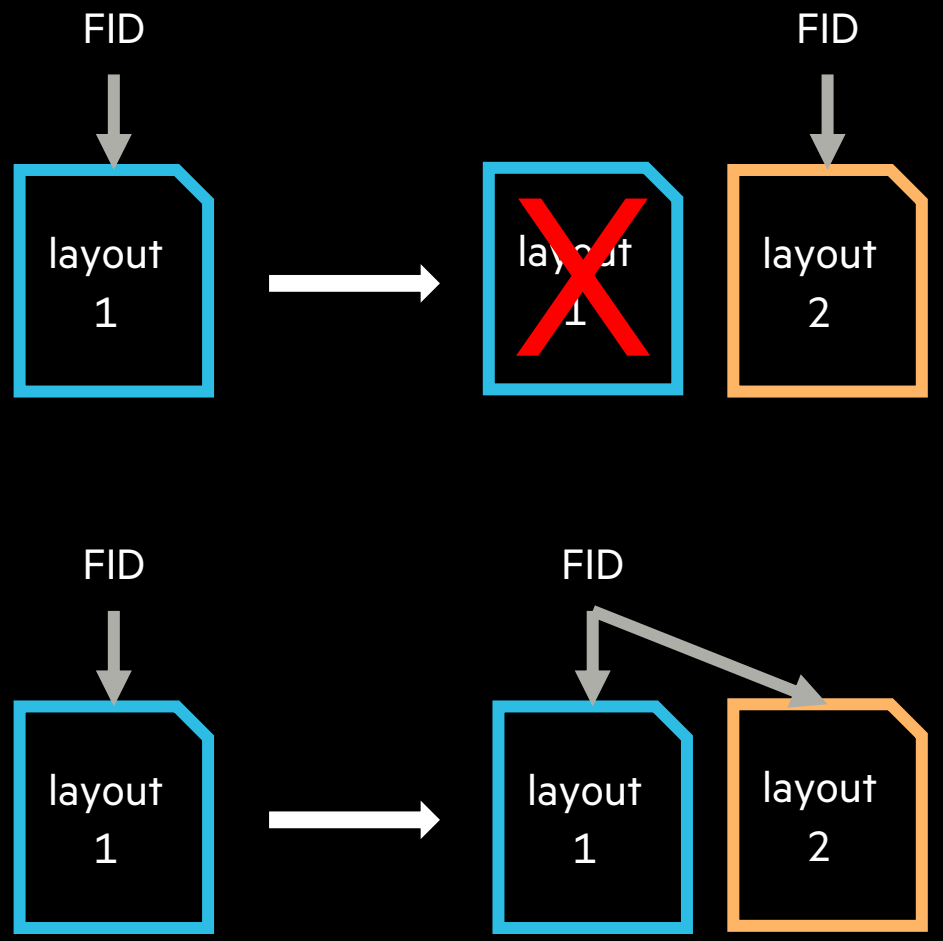
May 9, 2022

WHAT IS MIRROR/MIGRATE

- Layouts are fixed
- Mirrors are a type of layout = mapping of file extents to OST objects
- As verbs: migrate and mirror **move data**
- Namespace is *not* changed
- Metadata* is not changed
  - FID, owner, mode, ctime, etc.
  - *layout EA is

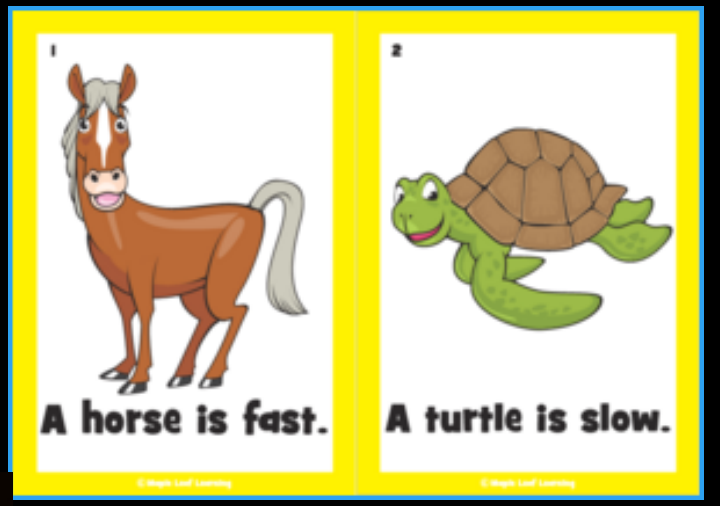
WHAT IS MIRROR/MIGRATE

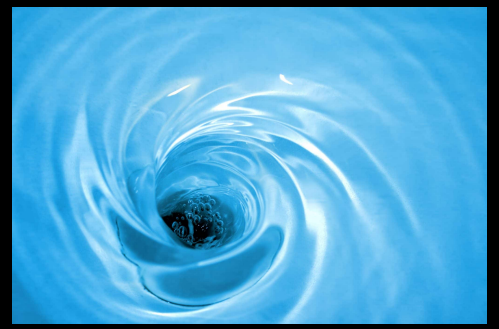
- “Move” means “copy”
  - Atomic layout change when done tells Lustre that data now lives elsewhere
- Migrate: copy, then destroy original data/layout
- Mirror: copy, and add the layout to the original



WHY MIRROR/MIGRATE

- Change striping pattern
  - Didn’t anticipate file size
  - Optimize for new code/processing
- Different OST types – eg. flash, disk
- Rebalance
- Drain





- Mostly, you care more about striping on large files. Mostly.

HOW MIRROR/MIGRATE

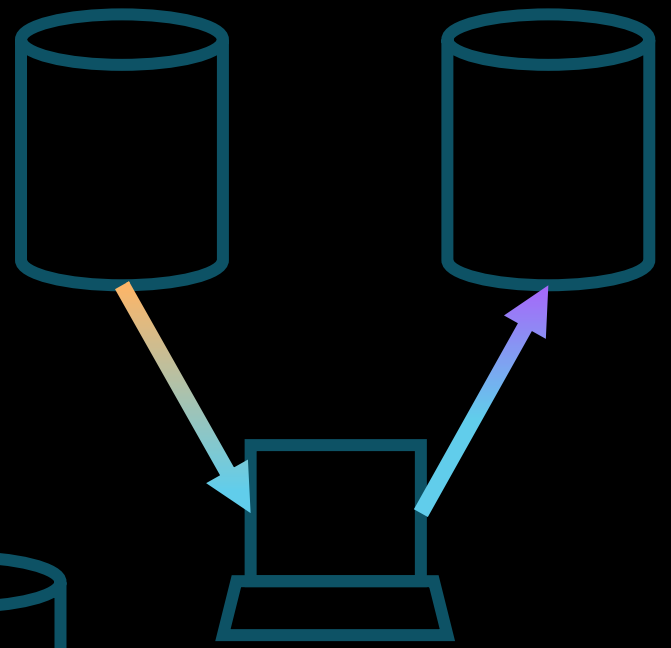
- lfs mirror create mirrorfile (new file)
- lfs mirror extend -N1 --pool flash mirrorfile (existing file)
- lfs mirror resync mirrorfile
- lfs migrate --pool flash monofile

SEE ALSO

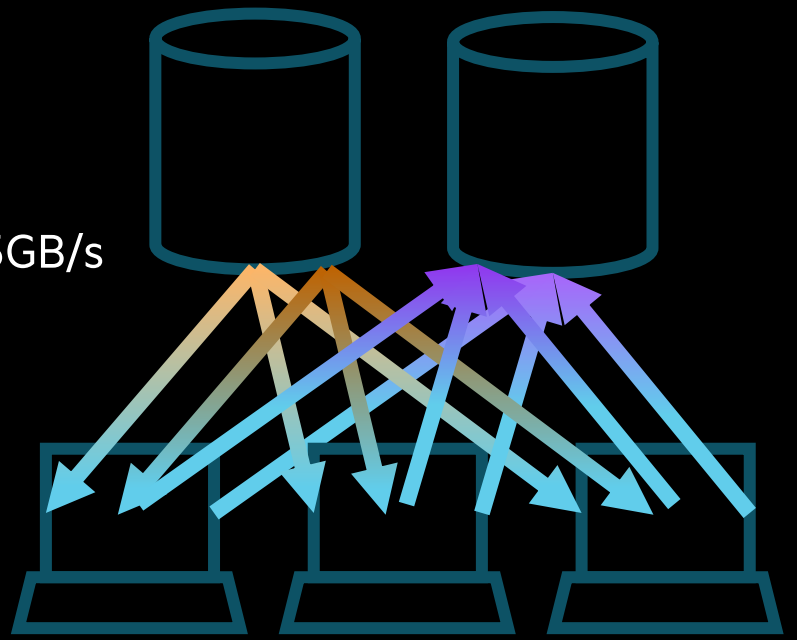
lfs(1), lfs-setstripe(1), lfs-mirror-create(1), lfs-mirror-extend(1), lfs-mirror-split(1), lfs-mirror-verify(1)

PERFORMANCE

- We want to move large files (most useful)
- But lfs operates locally – single client, single thread
- Copy operation has to both read and write
  - Migrate 1GB file costs 2GB of data transfer
  - Serialized
- = slow



- Need to parallelize
  - Across chunks of files – single thread limits – 5GB/s
  - Across nodes – single node limits – 25GB/s
  - Across files – overlap reads and writes



PARALLELIZE: THE ISSUES

- This is not what LFS does. Need to use llapi directly
- Migrate is “easy”
  - Get file data version
  - Use your favorite parallel file copy tool to a temp file with target layout
  - Verify data version + atomic swap layouts
  - Run this on multiple files simultaneously coordinated over multiple nodes
- Mirror is annoying
  - Requires read lease lock to resync a mirror (so writes can signal abort) LU-13668
  - But this can’t be held by more than one client at a time
  - So implement as migrate:
  - Break the mirror
  - Create a new temp file with the mirror layout
  - Copy the data
  - Verify data version + merge layouts

PARALLELIZE: THE ISSUES

- Specifying target layout
  - Parse all those LFS flags
  - Or read layout from a “template” = example file/dir
- Can’t add a plain-layout mirror
  - -E forces composite: *lfs setstripe -E eof -c -1 -p disk myfile*
  - Llapi version: force *layout->llot_is_composite* to true

    llapi_layout_comp_extent_get(layout, &start, &end);

    llapi_layout_comp_extent_set(layout, start, end);

MEASURING PERFORMANCE

- DD

  time $(for i in $(seq 1 99); do dd if=dirA/100G.${i} of=dirB/100G.${i} bs=64M iflag=direct,fullblock oflag=direct & done; wait)

- Truncate to create lots of large files quickly

  for i in $(seq 1 50); do truncate -s 100G 100G.${i}; done

- Live stack – threads in directIO

  for i in /proc/<pid>/task//stack; do cat $i; done | grep ll_direct | wc -l*
  27

- Open FDs

  ls -l /proc/<pid>/fd | wc -l; done

- Lustre stats
  - Extent stats: size of syscall

    8K - 16K : 39865 100 100 | 28799 100 100
  - RPC stats: size of actual Lustre RPCs

    1024: 15672 100 100 | 8484 100 100

```
lctl set_param llite.*.extents_stats=1; lctl set_param osc.*.rpc_stats=0   #clear
lctl get_param  llite.*.extents_stats
lctl get_param osc.*.rpc_stats
```

MEASURING PERFORMANCE: FIO

- Flexible IO paths: read, pread, readv, aio_read
- Multiple simultaneous ops: eg read/write
- Single node

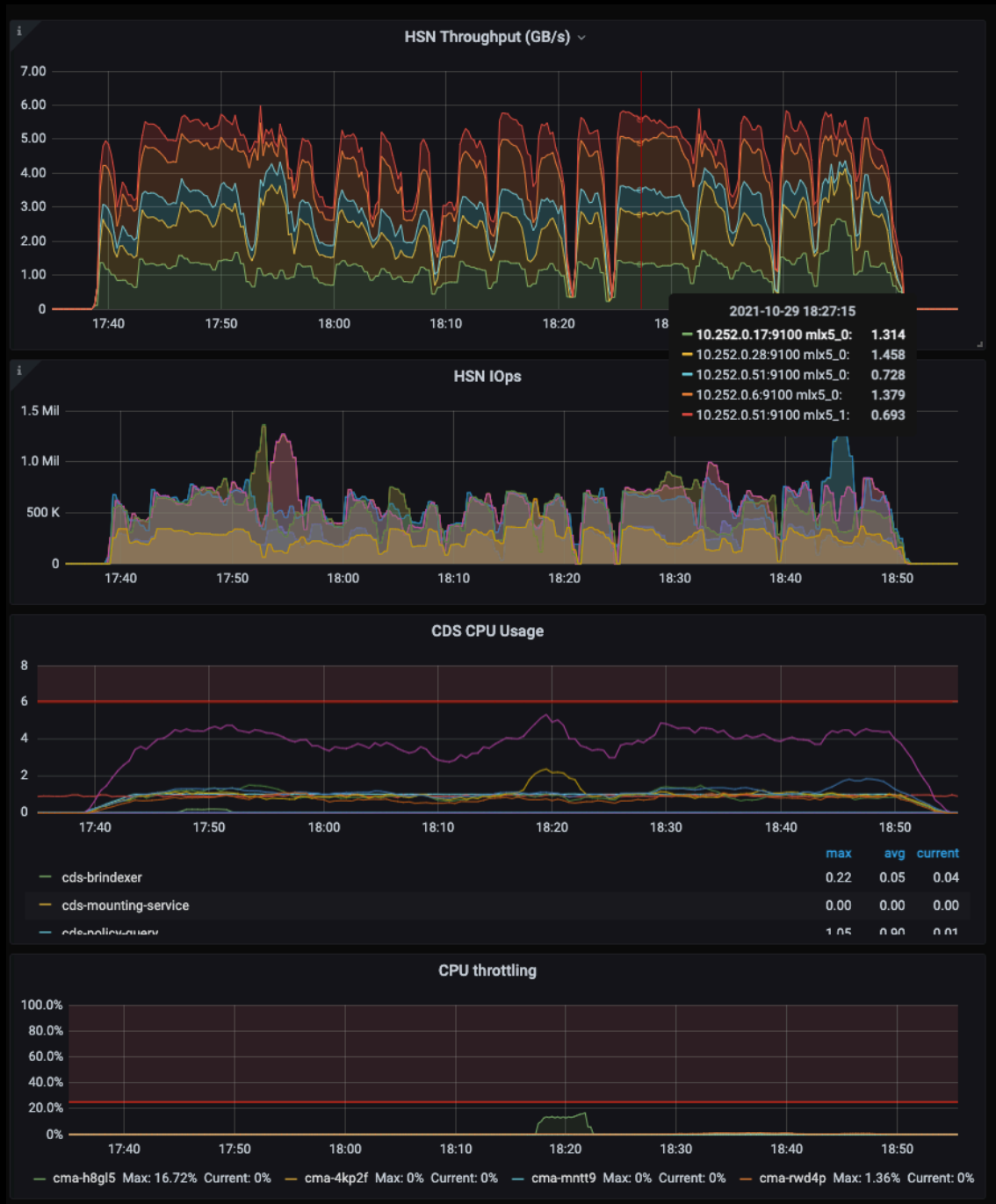
```
; -- start job file --
[global]
directory=/lus/nzrtest/fio/testdir
group_reporting
gtod_reduce=1
invalidate=1
thread
ioengine=sync
direct=1
iodepth=1
bs=16m
size=16g
numjobs=56

# use --section to run one at a time
[diow]
rw=write
exitall

[dior]
rw=read
exitall
; -- end job file --
```

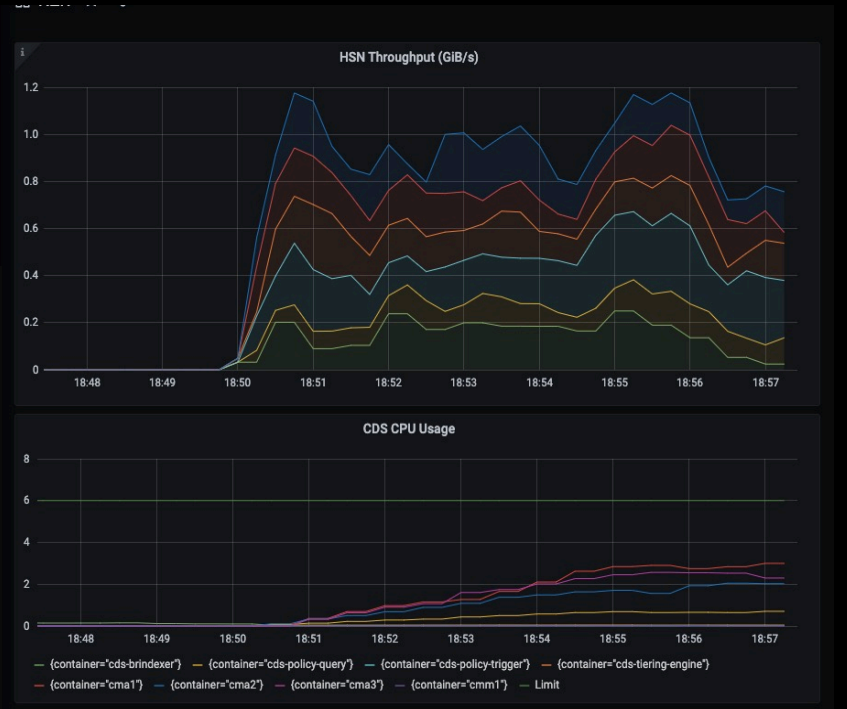
MEASURING PERFORMANCE: GRAFANA

- Prometheus + Grafana
  - No substitute for visual information
  - Anomalies are obvious
  - Timer captures avg MB/s; no nuance, no long tail info



OPTIMIZING PERFORMANCE: DIO AND FRIENDS

- Sendfile: kernel-to-kernel, no userspace copy
  - osc extents stats shows we're writing tiny chunks
- DIO
  - Faster IO, lower CPU
  - But must write sector-size multiples
  - Overshoot and truncate
- BIO
  - Use BIO for small files to avoid sector alignment
- IO-uring
  - Shared kernel/userspace async ring buffers
  - Libaio equivalence LU-13801

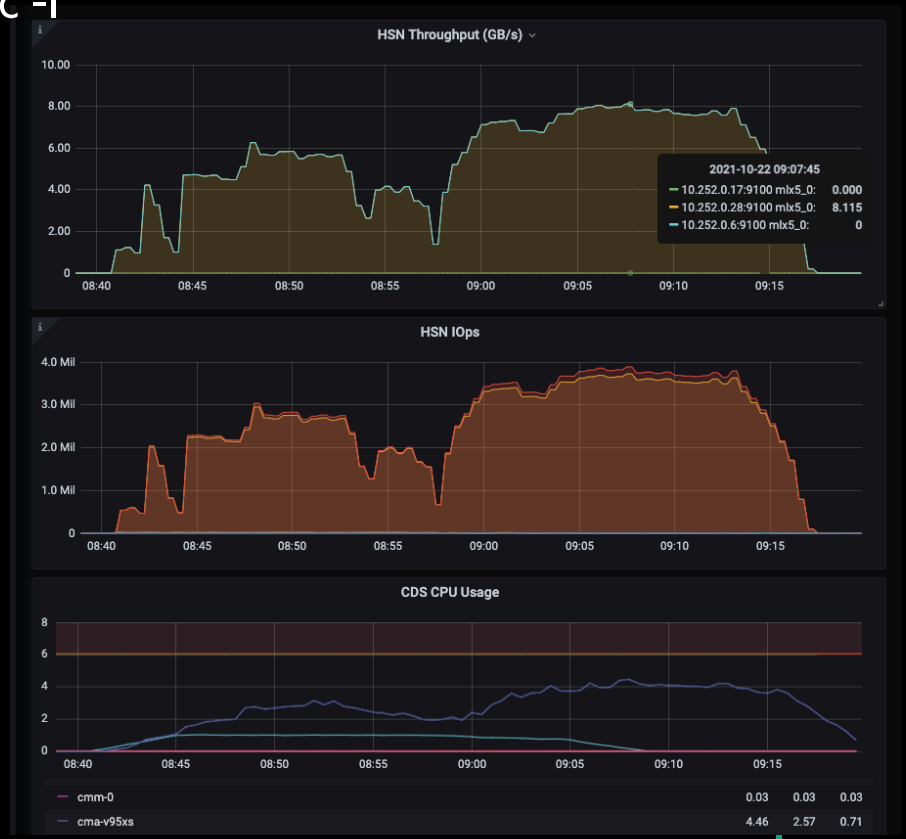


OPTIMIZING PERFORMANCE: THREAD COUNTS

- Use FIO to quickly find maximum thread count performance
  - In my case, 112 threads
- Avg RPCs in flight peak around 50-56 on flash OSTs and 40 on disk OSTs
- Most threads doing DIO simultaneously

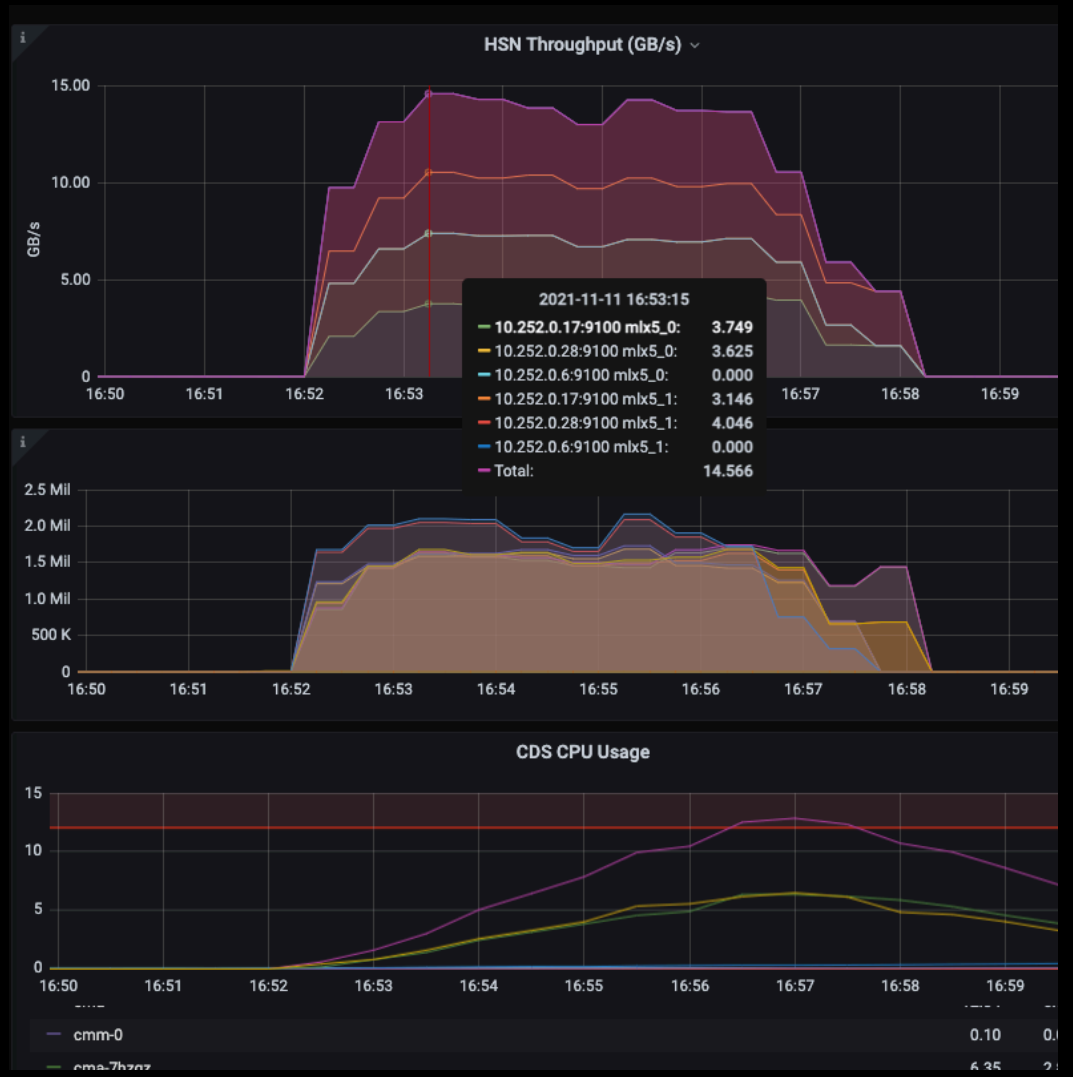
```
for i in /proc/$(pgrep fio)/task/*/stack; do cat $i; done | grep ll_direct | wc -l
108
```

	read			write		
in flight	rpcs	%	cum %	rpcs	%	cum %
47:	3245	6	51	1995	4	30
48:	3392	6	57	2220	4	35
49:	3716	7	64	2623	5	40
50:	3703	7	71	2758	5	46
51:	3546	6	78	2935	5	52
52:	3271	6	85	3070	6	58
53:	2916	5	90	3291	6	65
54:	2154	4	94	3297	6	71
55:	1576	3	97	3143	6	78
56:	1160	2	100	2970	6	84
57:	0	0	100	2770	5	89



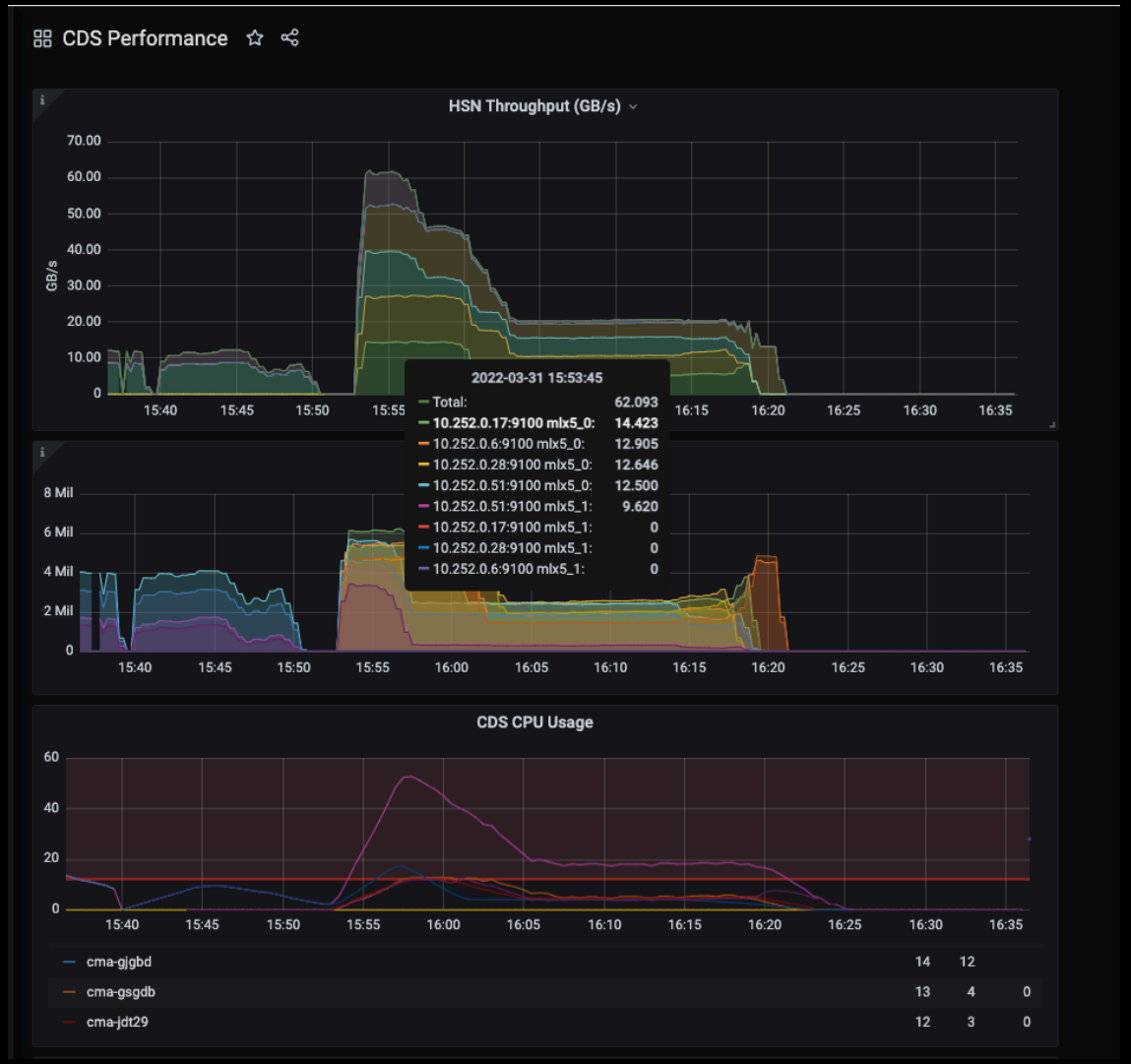
OPTIMIZING PERFORMANCE: CHUNK SIZE

- How should we break up a large file?
- Each node reads from 1 OST?
  - Eg. stripe count = 4, then read 1MB, skip 3MB
  - No OST contention, but perf is pretty bad
  - No 4/8/16MB reads
  - No readahead
  - Also, our use case is explicitly changing striping
- Read separate, large buffers on each node
  - Try to read 64MB, expecting read() will return less
  - Many clients : many OSTs



OPTIMIZING PERFORMANCE: FASTER LUSTRE SYSTEM

- Multirail running about 2x single rail, but not evenly loaded
- Multirail node needed more CPU



OPTIMIZING PERFORMANCE: LOAD BALANCING

- Load balancing problem is obvious from the graph
- Node with 2 nics finishes early, while other 3 nodes carry on
- With enough new work this doesn't matter
- Ideally measure per-node performance, and distribute work quantity based on speed

GOTCHAS

- RPCs in flight
- Lustre version
  - 2.15 server for LU-13668: open-for-read should not conflict with mirror creation
- CPU throttling
- One slow client
  - Bad hardware? Phantom load? IB firmware version? No…

THANK YOU

Nathan.Rutman@hpe.com