Lustre performance on DDN SFA18KE

September 23rd, 2019
Gaël DELBARY
TOPICS

- T1KF project
- Hardware Storage
- Performance clarification
- Page cache revival
- Performance in real life
- Overstriping
T1KF Project – Key numbers

► T1KF: Flash Parallel Filesystem (Q4 2019) for the next exascale supercomputers

► Target:
  - First store storage level (same Lustre FS as STORE) with Lustre Pool
  - 1 TB/s in write sequential (5% of times)
  - Mounted by all supercomputers
  - Data migration with RobinHood (via lfs migrate)
  - Hide complexity to end users

► Usage:
  - Checkpoint restart files (mostly sequential writes)
  - Final files (mostly sequential writes)
  - Data post-processing (sequential writes + random read)
T1KF Project – Flash required?

▶ Reasons:
  - Low footprint (limit to 5 racks)
  - Random read
  - High write throughput

▶ External things to monitor:
  - Weight (density is not light)
  - Power (SSDs consume energy)

▶ Collaboration with DDN engineering

▶ Goal: Evaluate Lustre performance with SSD drives
Hardware storage

DDN SFA18KE with SFAOS 11.5 GA
- Dual controllers
- Skylake processors
- 784 GB of RAM
- 8x Infiniband Mellanox EDR ConnectX5 dual-port
- QEMU inside for Embedded
- 8x VMs per couplet

2x Declustered RAID pool (56x SSD HGST SS530)
- 2x hotspares
- 27x usable disks (RAID5 inside means 24 for write)

Tested configuration: 1x VM (OSS inside)
- 75 GB of RAM
- 4x CPU cores (8x virtual with hyperthreading)
- 1x EDR Infiniband card
Common parameters

- Devices formatted with no lazy initialization
- CentOS 7.6 (ioscheduler: none)
- MOFED 4.5.1
- Lustre 2.12.2 (Idiskfs backend)
- Lustre clients tuning (no checksums, max_pages_per_rpc=512)
- FIO 3.11 (direct=1, ioengine=libaio)
- IOR 3.2.1 (8 EDR clients)
- EDR Infiniband network
Performance clarification

▶ Lustre performance is often bad?
  - Sure?
  - Investigations needed before affirmation 😊

▶ Lustre can perform if under layers are, like:
  - Network
  - Internal PCIe architecture
  - I/O devices
  - CPU resources
  - Kernel I/O layers
  ...
  - Lustre layers (focused today)
LUG 2018 talk: Lustre server on kernel 4.12.8
- LU-10942 and LU-11071 based

Summary:
Reducing latency = better throughput

Not perfect solution:
- throughput fluctuation
- cpu usage for memory reclaim and not for I/O

LU-11347 rescue
LU-11347: Do not use pagecache for SSD I/O when read/write cache are disabled

Requirements (for 2.12.2):
- lctl set_param osd-ldiskfs.*.read_cache_enable=0
- lctl set_param osd-ldiskfs.*.writethrough_cache_enable=0
- echo 0 > /sys/block/”device”/queue/rotational
LU-11347: pagecache passthrough off

- I/O done by ofd_commitrw (15%)
- Find_or_create_page (59%)
- I/O is running? Bottleneck idea?
Page cache revival: LU-11347 light deep dive

LU-11347: pagecache passthrough on

- I/O done by ofd_commitrw (69%)
- Time past in block layers
- Block device is the bottleneck
Page cache revival: LU-11347 light deep dive in real life

LU-11347: pagecache passthrough off: ior write sequential (files per process)
- Do you see something?
**Il_ost_io**: Lustre processes to “run” I/O on LUN
- Too much `find_or_create_page` (0.88%, too low)
- Direct reclaims occurs
- Same graph as the first one
- All ll_ost_io* have been aggregated
LU-11347: pagecache passthrough off: ior write sequential (files per process)
- Do you see something?
Page cache revival: LU-11347 light deep dive in real life

- 4x lustre network processes are running
- Ofd_commitrw are significant
- I/O bottleneck
Performance in real life: reproductibility

Focus: IOR File per Process write sequential workload
ior -F -D 60 -k -E -w -v -t 2m -b 64g

![IOR File Per Process write sequential workload chart]

Performance drop 30% lost
Performance in real life: Analysis

► Switch to block device (FIO on raw device, don’t do this in production)
► Same performance drop (follow IOR graph)
► Backend disks seems slower
  - Storage array is fine
  - SSD drives state?

► Garbage collector is running
  - Write amplification mechanism

► Solution: planned or synchronous fstrim (-o discard)
  - Erase nand cells which are not anymore used by FS
  - LU-11355 to run manually trim operation on OST target
Performance in real life: SSD choice criteria

**XSR methodology**
- Goal: monitor SSD capability to regain « nominal » performance

<table>
<thead>
<tr>
<th>SSD Type</th>
<th>2 hours 128KB sequential write</th>
<th>2 hours 128KB random write</th>
<th>2 hours 128KB sequential write</th>
</tr>
</thead>
<tbody>
<tr>
<td>3DWPD Hitachi SS530</td>
<td>1h40</td>
<td>47 minutes</td>
<td>1h10</td>
</tr>
<tr>
<td>1DWPD Samsung PM1643</td>
<td>1h10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3DWPD Samsung PM1643 Low power</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Recovery time, shorter is better
Overstriping: LU-9846

Patrick Farrel LUG’19 slides:

Requirements:
- LU-11796: Remove unnecessary assert
- LU-9846: Add overstriping support
- Landed for Lustre 2.13

Main idea: more than one stripe per ost
Main target: single shared file (Lustre killer)

Patrick F. modesty: “Basic change is trivial: Remove explicit checks preventing this”
Result: +866, -125 code lines (huge work)
Summary

- LU-11347 helps for write sequential workload
- LU-9846 add performance for single shared files
- Flamegraph: good tool to troubleshoot and understand what is going on low layers
  http://www.brendangregg.com/flamegraphs.html

- Community efforts are efficient and bring Lustre features/performance

- Todo for T1KF:
  - Benchmarks at scale (Infiniband topology to improve)
  - Kernel CentOS 8
  - IO500
Questions?