Whamcloud

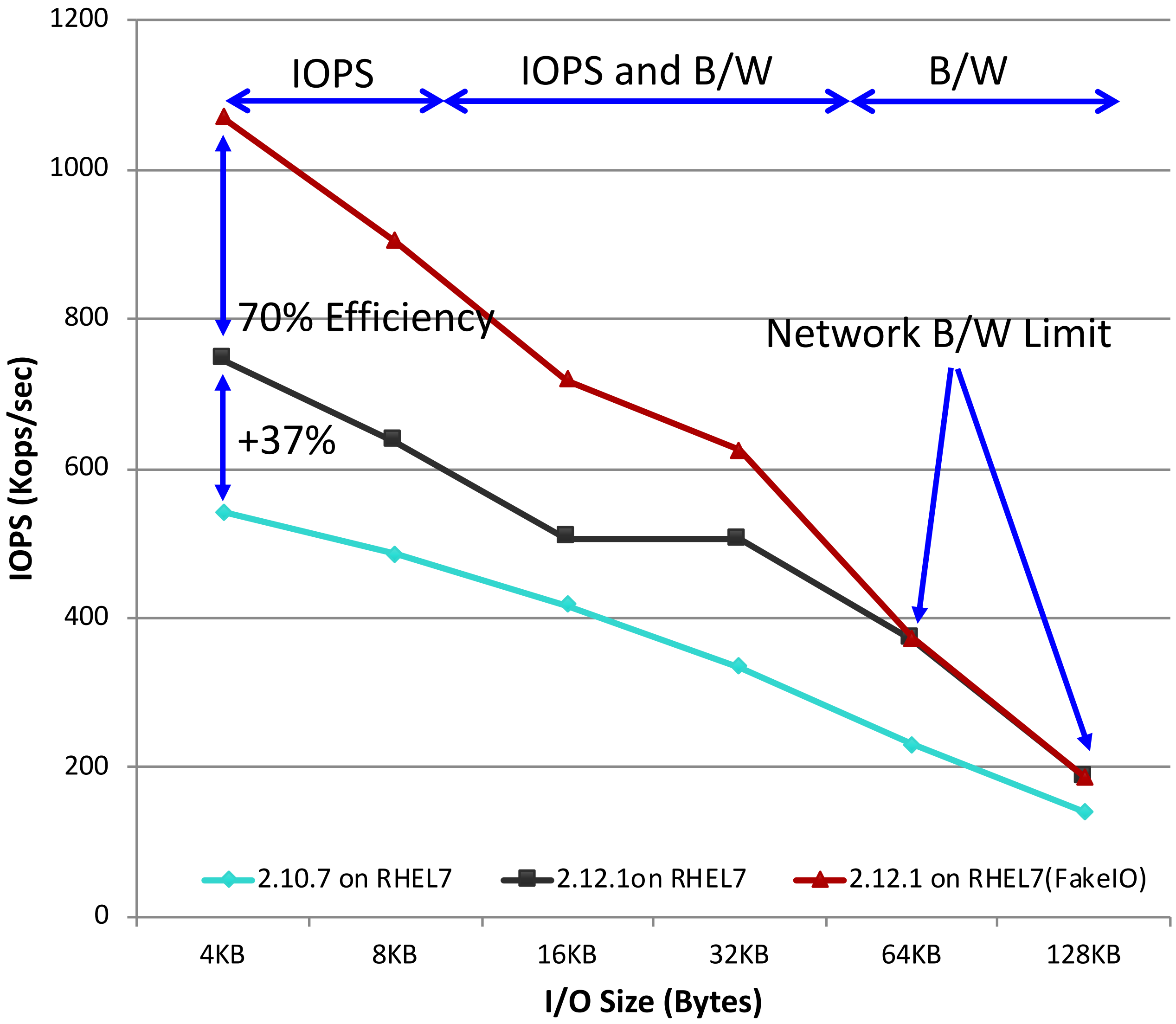
Lustre Optimizations and Improvements for Flash Storage

Shuichi Ihara

DDN® STORAGE

Random Read IOPS (N:1) Lustre 2.10.7 vs. 2.12.1

Whamcloud



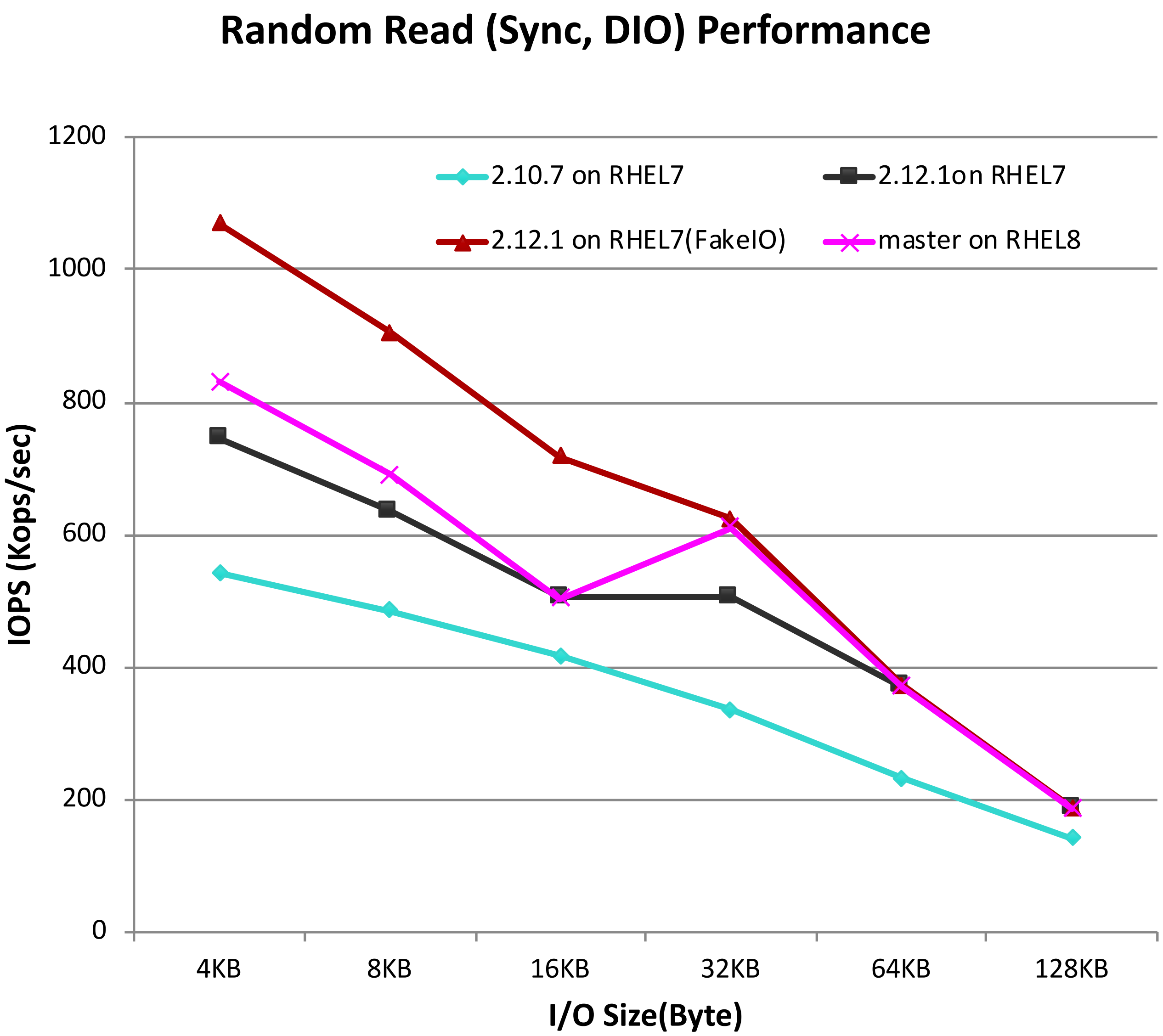
- Workload is fio with random O_DIRECT reads on different IO size
- 1x DDN AI200 with 20 x 1.6TB PCI NVMe devices
  - 2x vOSS (on Virtual Machine)
  - 8x CPU Cores and 64GB memory per OSS
  - InfiniBand EDR
- 32x clients
  - 2x 12 CPU cores, 128GB memory and InfiniBand FDR
- Lustre 2.12.1 is 37% higher 4K IOPS than 2.10.7
  - 70% of peak IOPS efficiency vs. RAM-only workload
- Lustre 2.12.1 is better than 2.10.7 at every IO size

Where do IOPS optimizations come from?

- Adding hardware resources (e.g. number of CPU cores) is easy way to improve IOPS
- IOPS efficiency (IOPS/core) improves performance with denser storage and less power

- LU-11164 `lvbo_*()` methods to reuse env (Lustre-2.12)
  - A server CPU optimization for small random I/O
  - “flame graph” pointed out 10% CPU of OSS for non-beneficial functions
  - Resulted in ~10% performance improvement
- LU-1757 Short I/O Support (Lustre-2.11/Lustre-2.12)
  - Send <= 4KB read/write data inline with RPC request instead of separate RDMA request
- LU-11347 Do not use pagecache for SSD I/O when read/write cache are disabled (Lustre-2.12)
  - Do not use kernel page cache for read/write RPCs on OSS if read/write caches are disabled
  - Use pre-allocated pages per thread to avoid page cache overhead
  - ~13% IOPS sustained improvement especially after memory reclaim triggered
  - Disable OSS read/write cache automatically if so

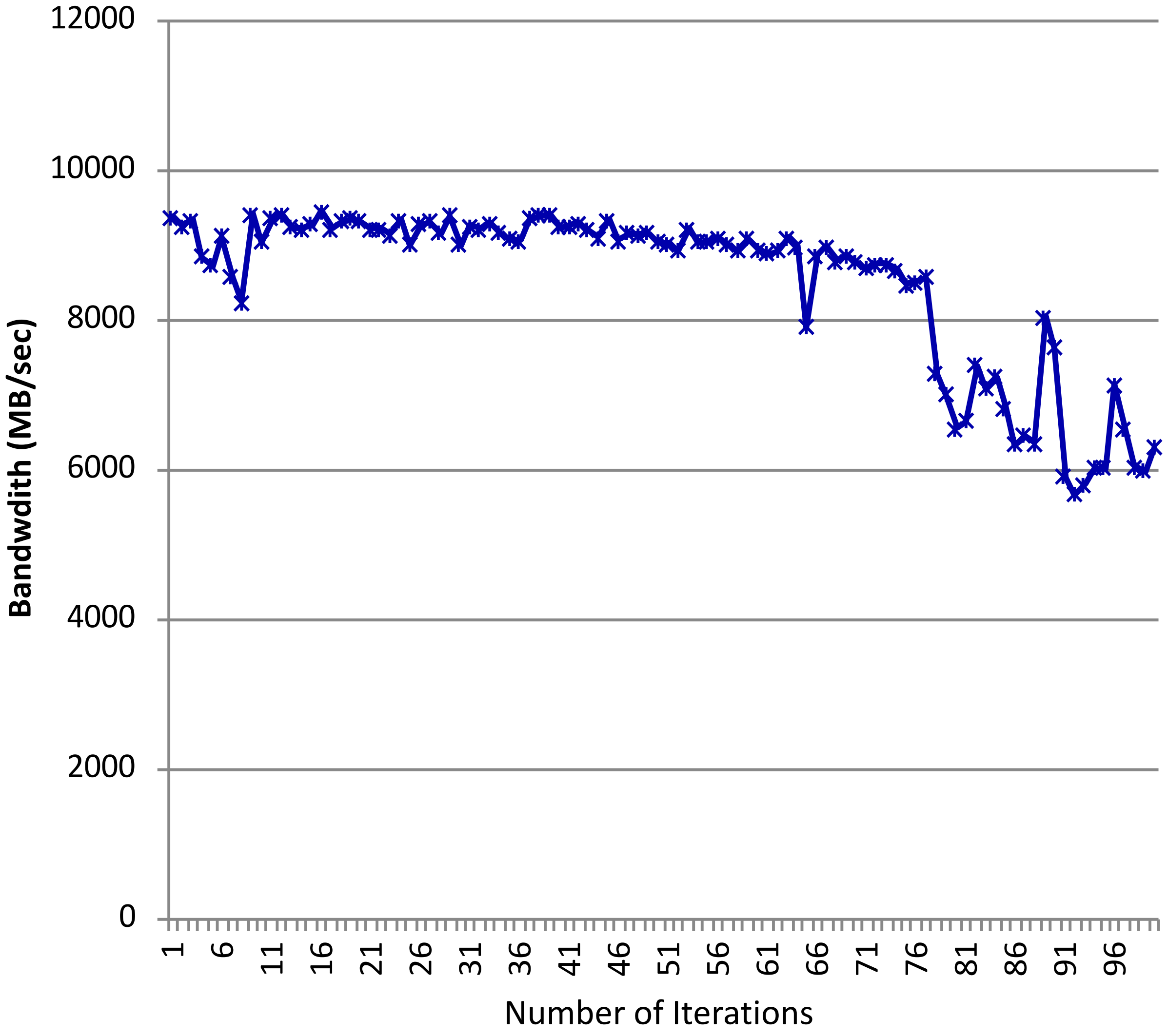
Experimental Setup and Evaluation of RHEL8 Server Kernel



- Keep same H/W configuration as before
- Replaced Lustre server kernel with RHEL8 (Beta)
- Apply Lustre patches to master branch to build
  - LU-11200 Centos7 arm64 server support
  - LU-11838 Support linux kernel version 4.18
- Another 10% IOPS gain on 4KB IO size
- Achieved 80% IOPS efficiency against fake-io
- Maximized available IOPS for 32KB IO size
- Need to investigate 16KB IO size for bottleneck

Performance Impact of FTL Garbage Collector

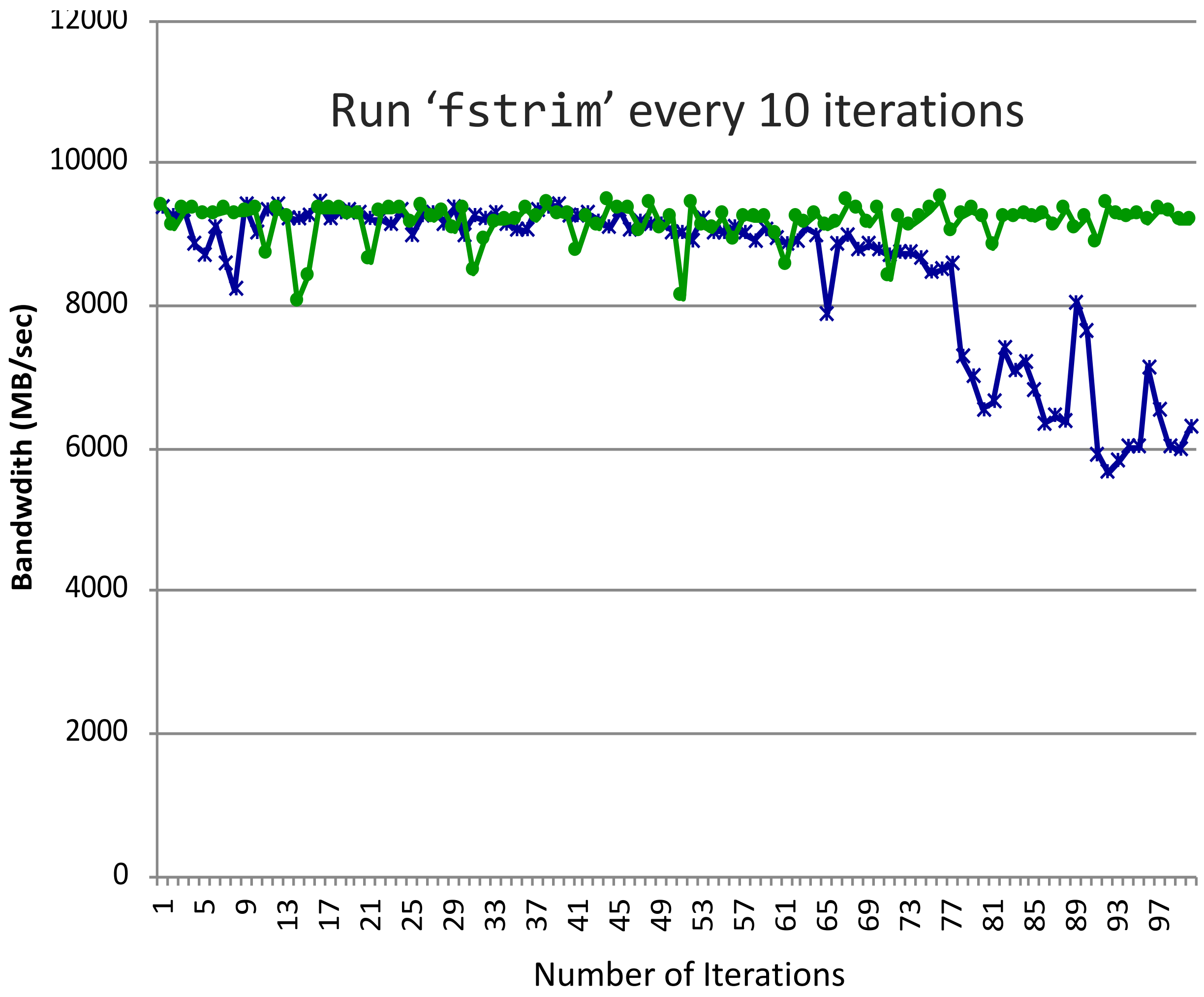
Continuous `obdfilter-survey` (9x Intel P3520 1.2TB)



- SSD/NVMe doesn’t allow overwriting the same cell unless it’s erased and free
- SSD controller runs background CG to free cells
  - Even if cell is erasable but not freed yet
- Background GC operations hurt performance if there is a continual foreground workload
- A simple test scenario ‘continuous `obdfilter-survey`’ demonstrates slowdown over time

TRIM (fstrim) to Lustre OSTs

Continuous `obdfilter-survey` (9x Intel P3520 1.2TB)



- `fstrim` to mount point of flash device
  - Discards all unused blocks in filesystem
  - Prevents unexpected GC by SSD controller
- ‘`fstrim`’ works against `ldiskfs`
- Patch “LU-11355 lustre: enable fstrim on lustre device” (Lustre-2.13)
  - Allows `fstrim` to OST mount point directly

```
oss# fstrim -v /mnt/lustre/ost/ost0000
```

- Can Integrate policy based ‘`fstrim`’ rather than continuous trim after each unlink
  - e.g. issue `fstrim` if there are less active IOs

Conclusions

- Lustre-2.12.x contains number of IOPS optimizations for flash devices
  - Demonstrates significantly better(+35%) Random Read IOPS and bandwidth than Lustre-2.10.x
- Newer Linux kernel for Lustre servers provides IOPS improvements
  - Performance numbers are encouraging
  - There are further possibilities to maximize CPU utilization and efficiency of IOPS/CPU core
- Move IOPS scaling by number of CPU cores forward
- On flash system for Lustre, TRIM needs to be considered
  - Background GC causes unexpected performance drops
  - Lustre supports `fstrim` to OST/MDT mount point directly
  - It can also be implemented by policy based discard