



Whamcloud

Lustre Performance Yesterday, Today and Tomorrow

Shuichi Ihara



Overview

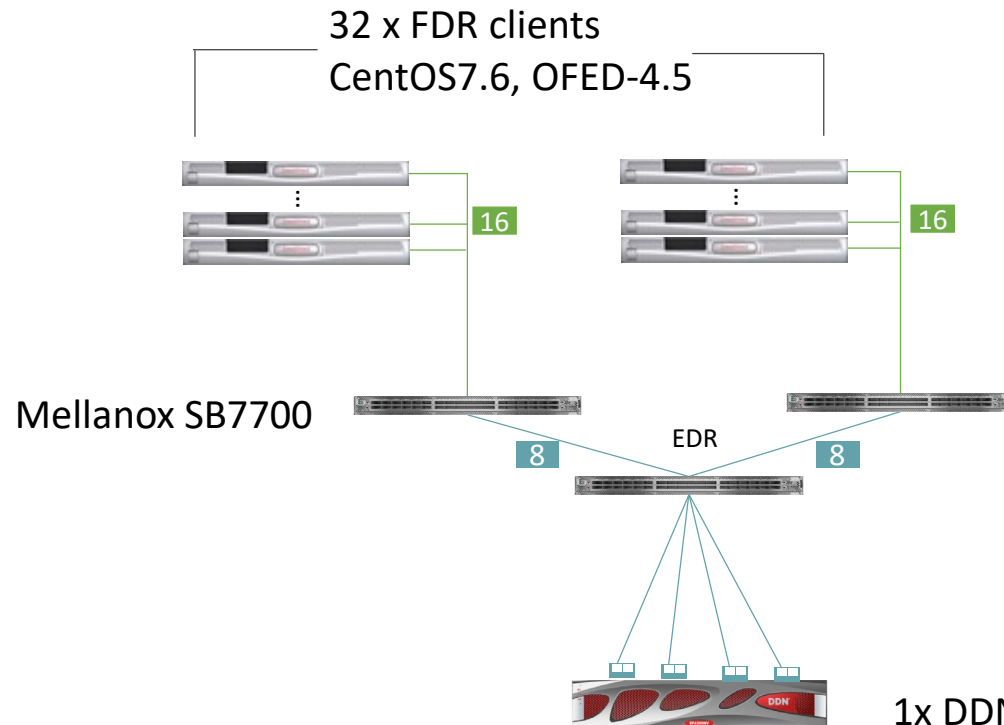
- ▶ Lustre continuously aims to improve performance, but it's hard to confirm because..
 - Many dependencies. (Not only Lustre version, but also kernel, OFED, Firmware version, etc.)
 - Need to understand test workload to see improvements
 - Same test workload, but it's tested by different users or different cluster (hardware)
 - Not all performance improvements mentioned on the Lustre roadmap (need to read changelog)
- ▶ Performance benchmark on same hardware with same configuration is important
 - Apples to apples testing with different Lustre branch
 - Tested three major Lustre versions (2.10.8, 2.12.2, and master)

We select and compare a number of interesting performance metrics on the different versions

Tested and Covered Workload

- ▶ **General Lustre Performance Comparisons on Flash System**
 - Aggregate Throughput
 - FPP (File Per Process), Strided SSF (Single Shared File) and Strided SSF Hard
 - 1MB IO Size on FPP and Strided and 47008 Byte IO size for Strided SSF Hard
 - Aggregate IOPS (4K Random Read)
 - Single OSS Performance
- ▶ **Single Client Performance Comparisons**
 - Single Thread
 - Single Client on High Network Bandwidth with Multi-rail
 - 4K Random Read IOPS
- ▶ **Metadata Operations for Small Files**

Benchmark Configuration



1x DDN AI400NV with 20 x 3.84TB SSD
Mixed 4 x OSS and MDS

► Test Conditions

- Keep same version on both server and client
- Same Lustre parameters on each tests

```
# lctl set_param obdfilter.*.brw_size=16
# lctl set_param osc.*.max_rpcs_in_flight=16
# lctl set_param osc.*.checksums=0
# lctl llite.*.max_read_ahead_mb=2048
```
- Enable new features if it improves performance



Whamcloud

General Lustre Performance Comparisons on Flash System

Aggregate Throughput Comparison



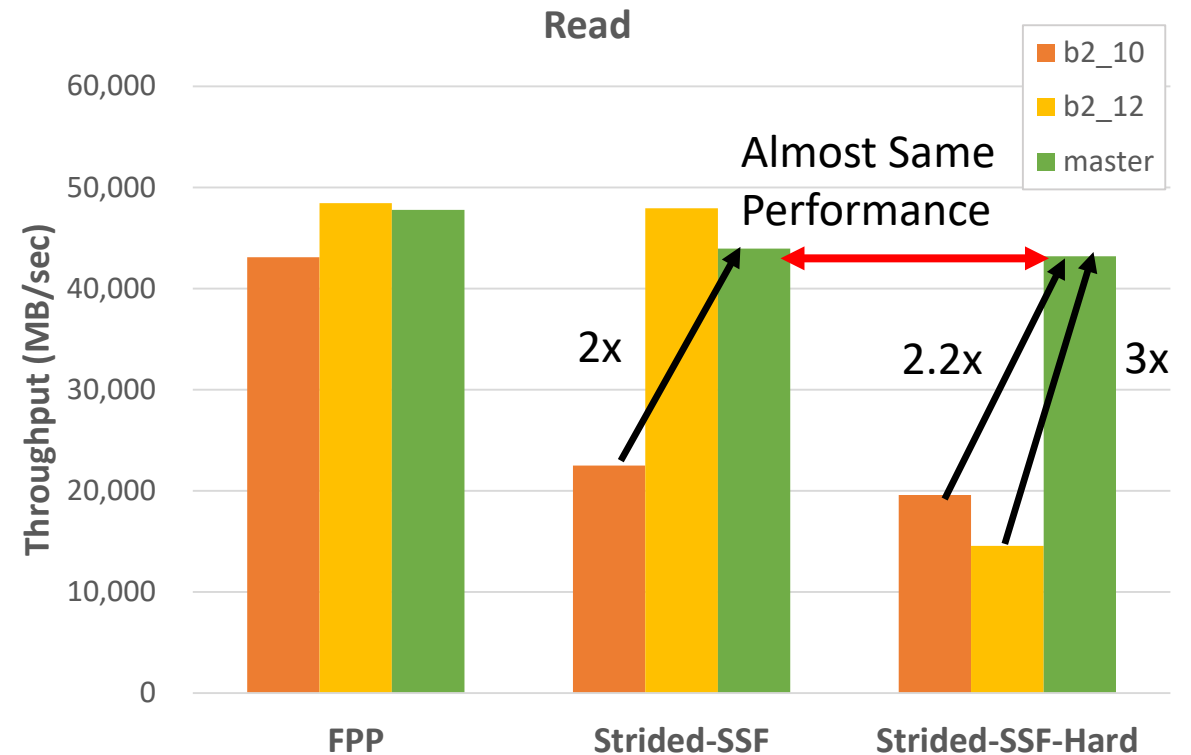
FPP (File Per Process), Strided-SSF (Single Shared File), Strided-SSF-Hard

Whamcloud

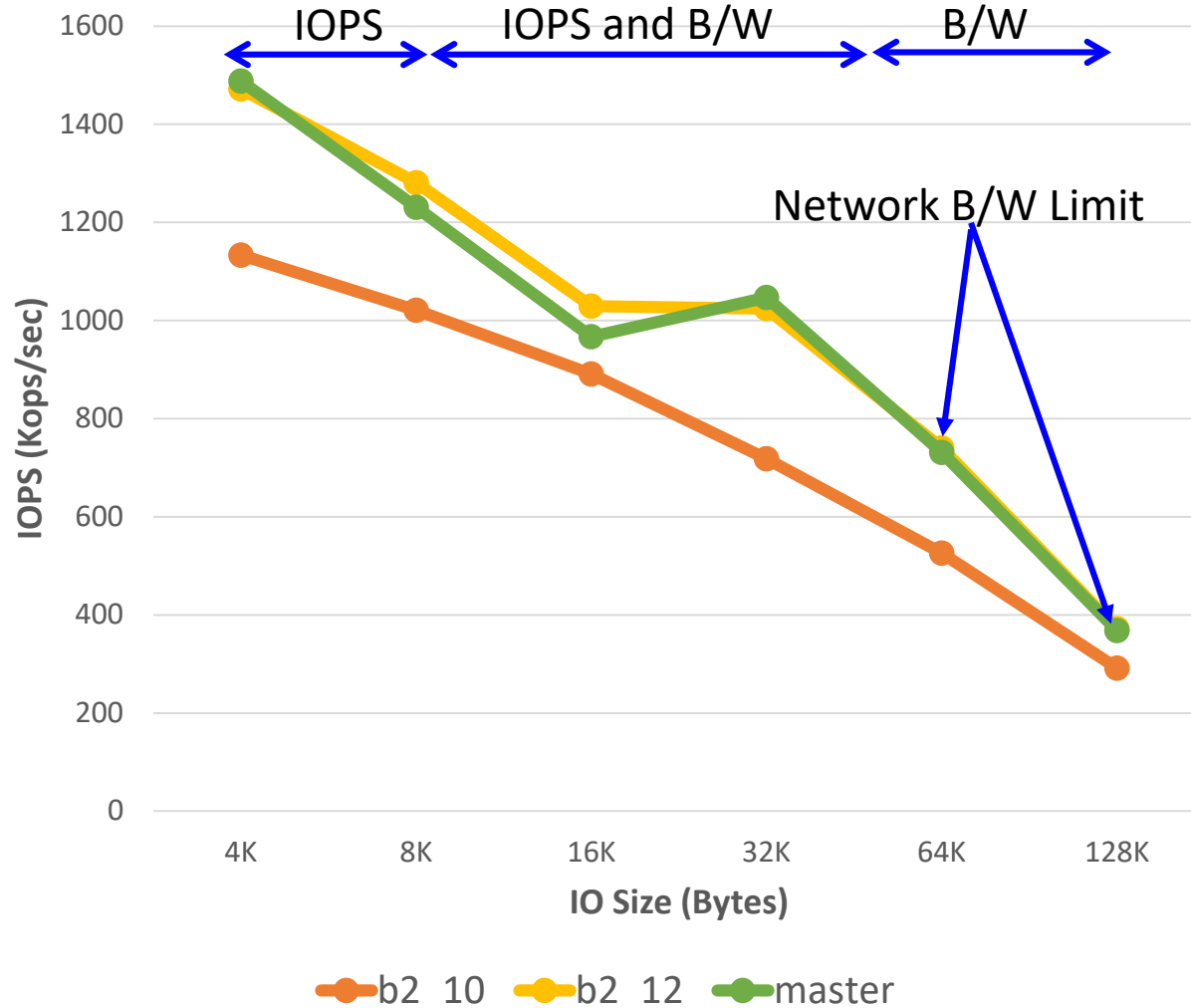
FPP: `ior -w/-r -s 6600 -i 1 -C -Q 1 -g -G 27 -k -e -t 1m -b 1m -F`

Strided-SSF: `ior -w/-r -s 6600 -i 1 -C -Q 1 -g -G 27 -k -e -t 1m -b 1m`

Strided-SSF-Hard: `ior -w/-r -s 132000 -i 1 -C -Q 1 -g -G 27 -ke -t 47008 -b 47008`

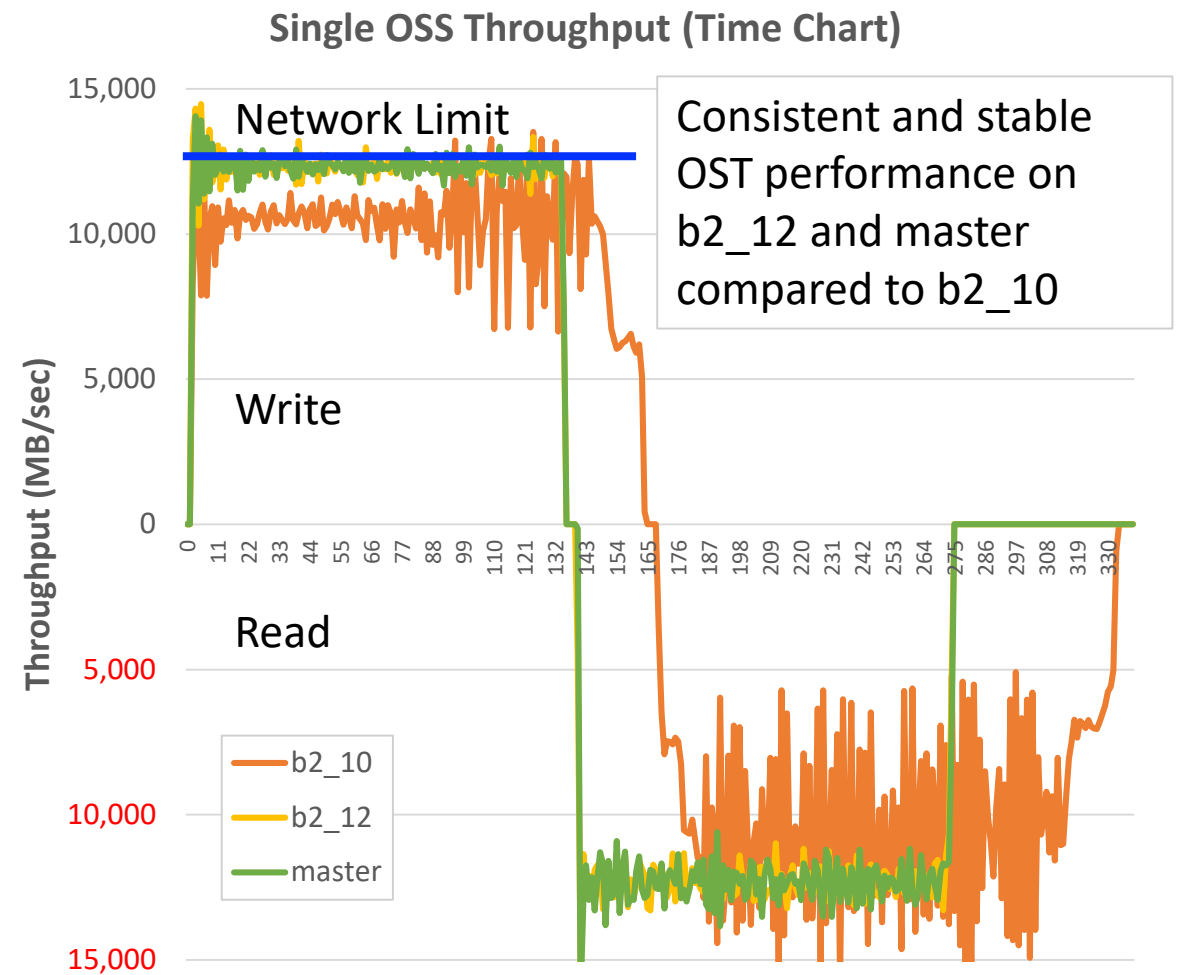
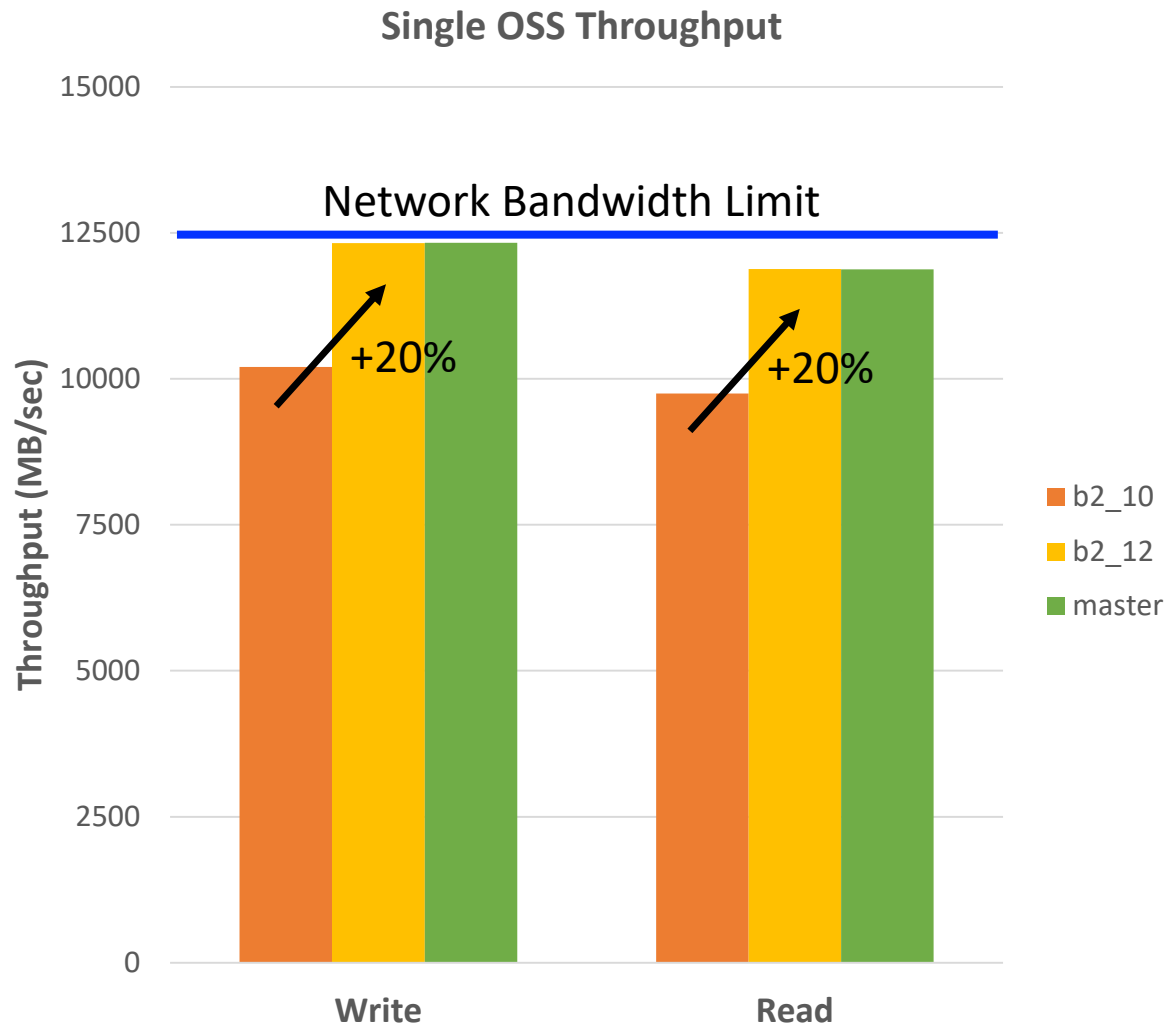


Aggregate IOPS (4K Random Read, O_DIRECT)



- ▶ Workload is fio with random O_DIRECT reads
 - Compare different IO sizes
- ▶ 1536 processes on 32 clients with random reads
- ▶ b2_12 and master 30% higher 4K IOPS vs. b2_10
- ▶ b2_12 and master branch saturate network
 - 4x IB-EDR bandwidth is 48GB/sec at 64K IO size

Single OSS Performance on Flash System Heavy Concurrent Access N:1 (32 clients, 768 processes)

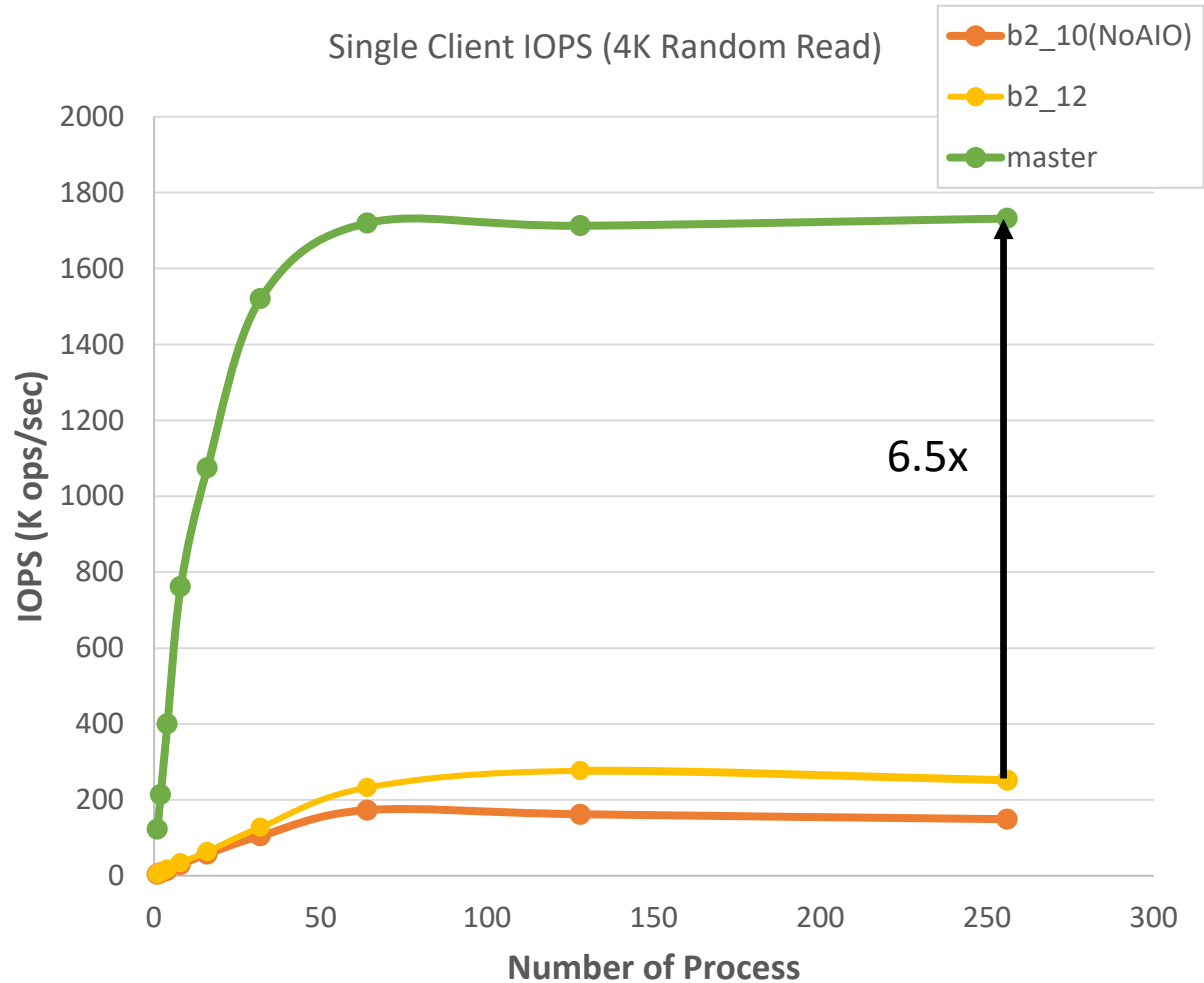




Whamcloud

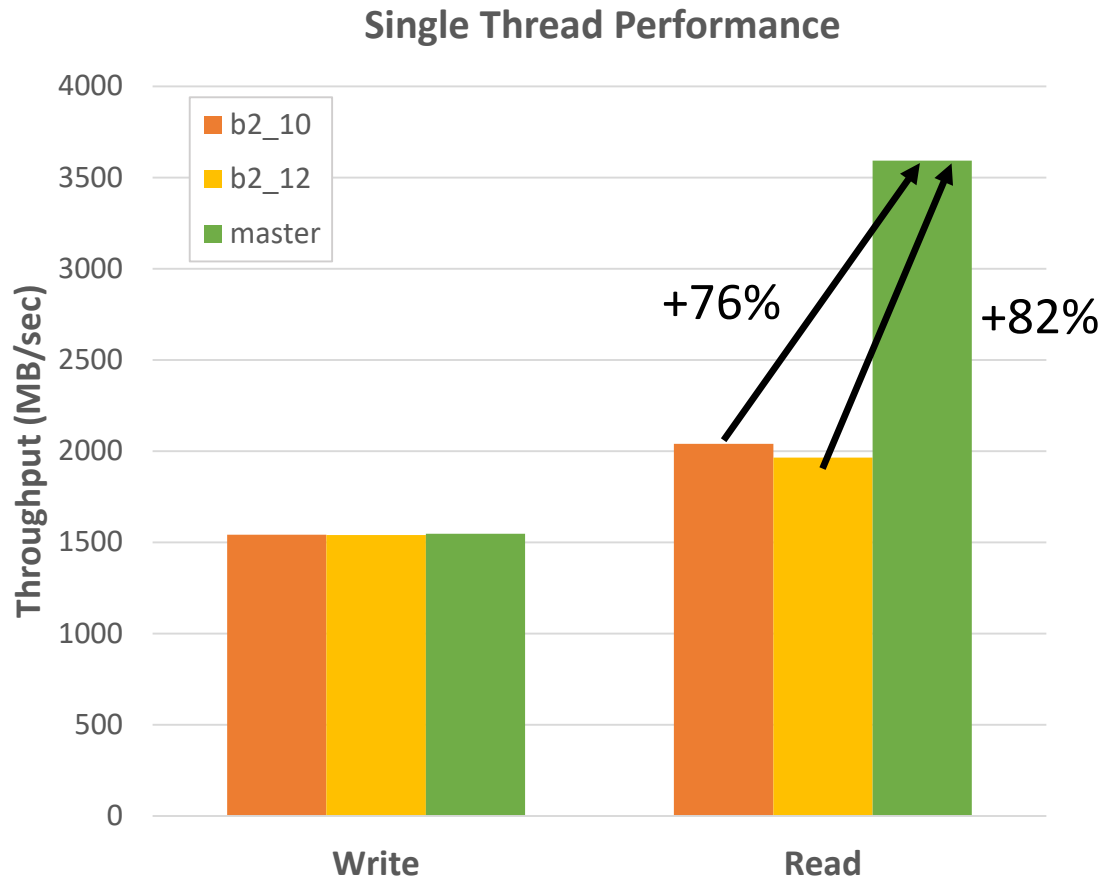
Lustre Client Performance Comparisons

Single client IOPS Scalability (4K random read, O_DIRECT/AIO)



- ▶ “fio” generates random O_DIRECT+AIO reads
 - Compare different number of processes on one client
- ▶ b2_10 doesn't allow O_DIRECT+AIO
- ▶ AIO works on b2_12 branch, but still “sync” mode
- ▶ master branch supports O_DIRECT+AIO
- ▶ master delivers 1.7M IOPS 4K random read on single client with sufficient storage performance

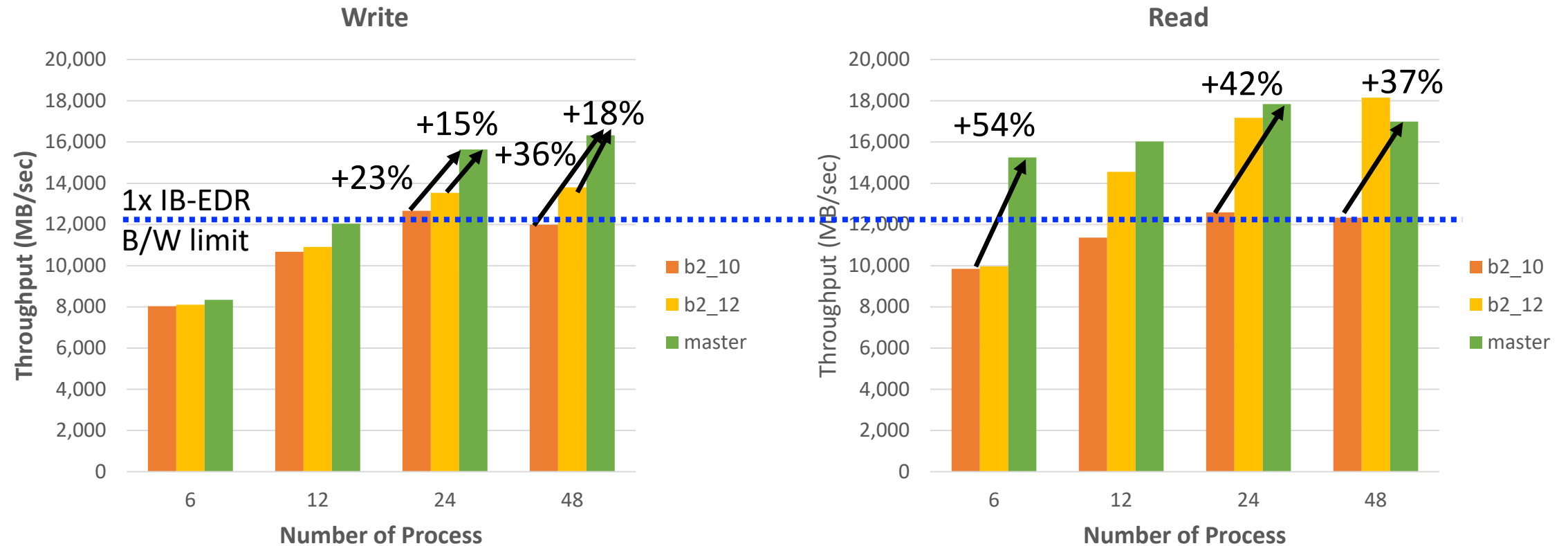
Single Thread Performance



- ▶ 1MB single stream to a large file
IOR -w (-r) -k -t 1m -b 384g -e -vv -o file
- ▶ Flashed cache between write and read
- ▶ Enabled Lustre file striping
lfs setstripe -c -1 -S 16M
- ▶ master 76-82% faster than 2.10/2.12 for read
- ▶ Write almost same speed on all branches

Single Client Performance with Multi-Rail

- ▶ High Bandwidth Network (2 x IB-EDR) Client with Lustre Multi-Rail
- ▶ File Per Process Workload (Aggregate 768GB files)
- ▶ Hidden Client performance bottleneck behind Single IB-EDR bandwidth



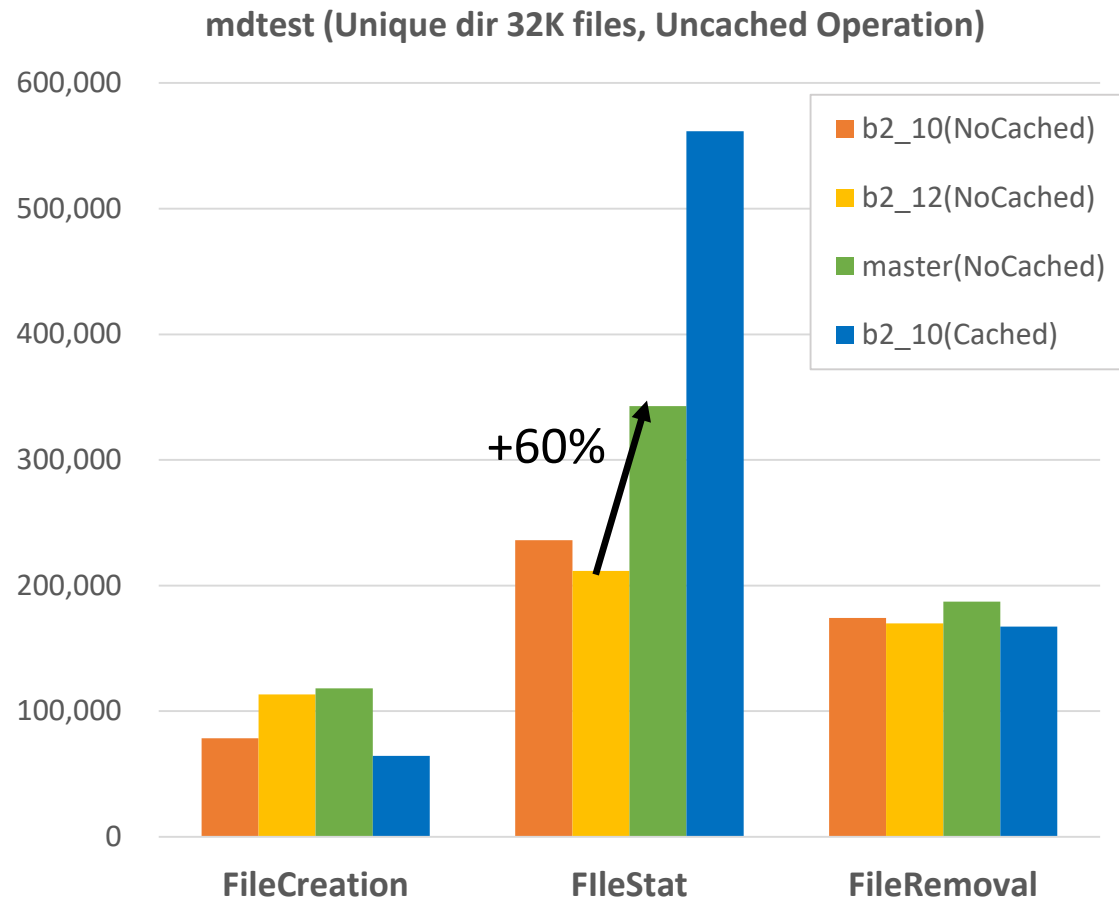


Whamcloud

Lustre Metadata Operations

Metadata Performance

Small File with Non-Cached Operations



- ▶ Workload is mdtest with create/stat/unlink
 - 11.5M x 32K files in unique directories
- ▶ 32 clients 768 processes generate workload
- ▶ No Cache aware mdtest

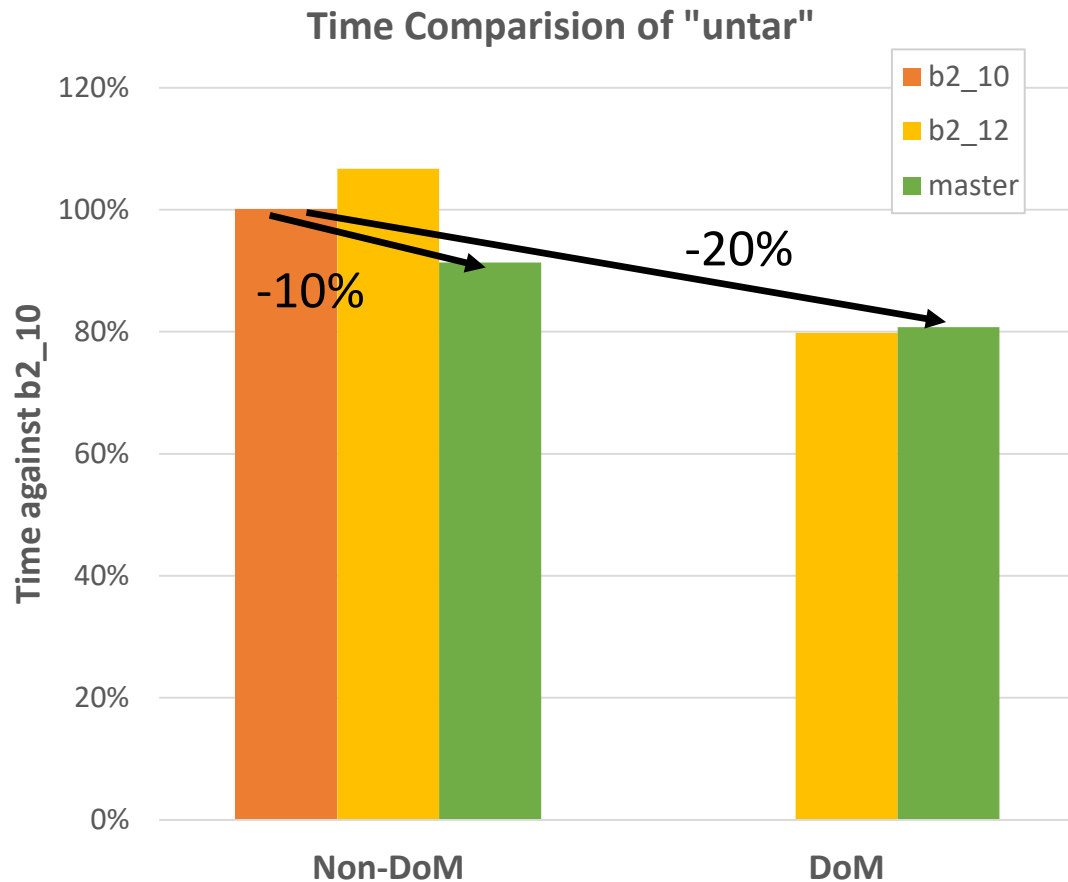
```
# mdtest -n15000 -Fvu -w 32k -e 32k -N24 -d /path
```

 - Client calls stat()/unlink() on remote created files.
 - Client cache at create is useless for next operations.
- ▶ Cache aware mdtest

```
# mdtest -n15000 -Fvu -w 32k -e 32k -d /path
```

 - Opencache can be valid for stat/unlink
 - Measured on b2_10 branch as a reference numbers
- ▶ master has 60% faster stat() than 2.10 and 2.12
- ▶ Unlink speed is almost same as cache aware mode

Single Client Small File Operation with DoM



- ▶ Speed "untar" operation
 - # tar xzf /lustre/linux-5.3-rc8.tgz -C /lustre/out
- ▶ Enabled DNE2 in all branch
 - # lfs setdirstripe -c 4 /lustre/out
 - # lfs setdirstripe -c 4 -D /lustre/out
- ▶ DoM for less than 16M files
 - # lctl set_param lod.*.dom_stripesize=16777216
 - # lfs setstripe -L mdt -E 16M /lustre/out
- ▶ master branch improves untar operation by 10%
- ▶ DoM can provide another 10% time reduction

Conclusions

- ▶ Apples to Apples comparison (different branch on same hardware) is very helpful
 - Tested major three branches (2.10.8, 2.12.2, and master) on same hardware
- ▶ master and b2_12 branch contains many performance improvements
 - Performance optimization for Flash system (both throughput and IOPS)
 - Client performance optimization (IOPS, bandwidth, reduced latency, remove overhead, etc.)
 - Metadata performance improvements for small files
- ▶ Future work
 - Add more performance metrics and different Lustre configurations (or hardware configurations. e.g. ethernet)
 - Found several performance bottlenecks in master that needs to be investigated



Whamcloud

Backup: Tested Lustre Branches

- ▶ b2_10 at v2_10_8
 - 1e117a3 New release 2.10.8
- ▶ b2_12 at v2_12_2-150-g2e0e446bca
 - 2e0e446 Revert "LU-11816 lnet: setup health timeout defaults"
- ▶ master at v2_12_57-22-g9b6b5e4798
 - 9b6b5e4 LU-11011 osc: add preferred checksum type support
 - Added the patches for LU-12518 and LU-4198