2015/04/15

DataDirect™ NETWORKS
INFORMATION IN MOTION™

OSD-Btrfs, A Novel Lustre OSD based on Btrfs

Shuichi Ihara
DataDirect Networks, Inc

Li Xi
DataDirect Networks, Inc

Why Btrfs for Lustre?

- Lustre today uses ldiskfs(ext4) or, more recently, ZFS as backend file system
- Ldiskfs(ext4)
  - ldiskfs(ext4) is used widely and exhibits well-tuned performance and proven reliability
  - But, its technical limitations are increasingly apparent, it is missing important novel features, lacks scalability, while offline fsck is extremely time-consuming
- ZFS
  - ZFS is an extremely feature-rich file system
  - but, for use as Lustre OSD further performance optimization is required
  - Licensing and potential legal issues have prevented a broader distribution
- Btrfs
  - Btrfs shares many features and goals with ZFS
  - Btrfs is becoming sufficiently stable for production usage
  - Btrfs is available through the major mainstream Linux distributions
  - Pachless server support. Simple Lustre setup and easy maintenance

Attractive Btrfs Features for Lustre (1)

DataDirect NETWORKS
INFORMATION IN MOTION

- Features that improve OSD internally
  - Integrated multiple device support: RAID 0/1/10/5/6
    - RAID 0 results in less OSTs per OSS and simplifies management, while improving the performance characteristics of a single OSD
    - RAID 1/10/5/6 improves reliability of OSDs
  - Dynamic inode allocation
    - No need to worry about formatting the MDT with wrong option to cause insufficient inode number
  - crc32c checksums on data and metadata
    - Improves OSD reliability
  - Online filesystem check and very fast offline filesystem check (planed)
    - Drives are getting bigger, but not faster, which leads to increasing fsck times

Attractive Btrfs Features for Lustre (2)

DataDirect NETWORKS
INFORMATION IN MOTION

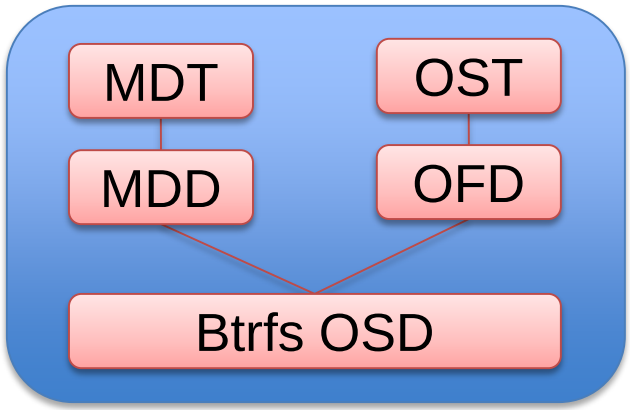
- Features which enable implementations of Lustre features
  - Writable snapshots, read-only snapshots
  - Subvolumes (separate internal filesystem roots)
  - Transparent Compression (zlib and LZO)
    - Enables Lustre to choose whether to compress data or not according to compression rates, access frequencies, backup policies or explicit instructions from applications
  - Send/receive (binary diff between a pair of subvolumes and then replay)
  - Out-of-band data deduplication
  - Reflink: Allows for files to be copied and modified, with only the modifications taking up additional storage space.

Lustre on Btrfs: Issues

- No user/group/project quota
  - Quota is important for Lustre file systems shared by lots of users or organizations
  - Per-subvolume quota in Btrfs might not be sufficient for some use cases
  - It is possible (though might be difficult) to add in the future
- Offline fsck tool is still under development
  - Btrfs is mostly self-healing and can recover from broken root trees at mount time
  - However, users *want* to run fsck in certain cases
- Performance issues
  - Concurrency: lock contention of Btrfs trees (extent tree/sub-volume trees)
  - Scalability: performance at tens of millions of files and hundreds of TBs
- RAS Features
  - Reliability and code stabilization

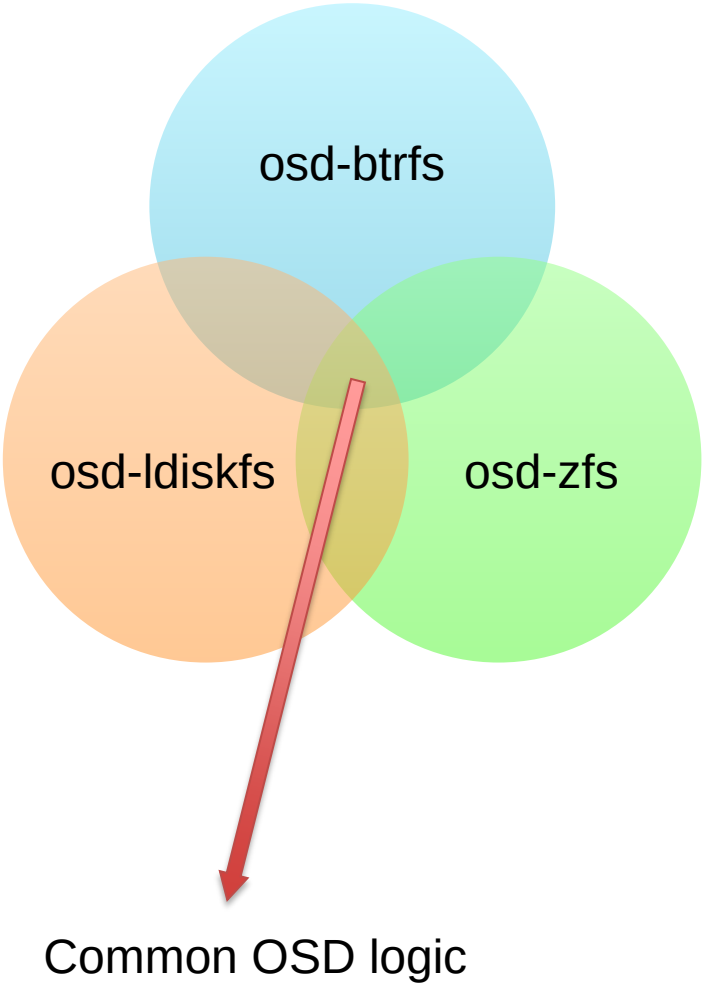
Motivation for Btrfs OSD

- Primary goal: Implement a Btrfs-OSD prototype
  - Proof of Concept study to implement basic OSD interfaces for Btrfs
  - Run benchmarks to ensure Btrfs is sufficiently scalable as Lustre backend file system
  - Evaluate performance tunings and enhancements necessary for a functional Btrfs-OSD
  - Evaluate features required to fully support existing and future Lustre features



Implementation of OSD-Btrfs

- Ldiskfs/ZFS/Btrfs OSD share a fair amount of code
- Btrfs shares more code with ldiskfs, rather than ZFS, since interfaces are fairly similar
- Common code portions could be implemented as functions in a shared library to reduce code redundancy



OSD-Btrfs Implementation (1)

- Patches are based on RHEL 7 kernel (3.10)
  - REHL 7(Centos 7)is likely to be used as the main distribution in the next couple of years
  - We back ported Btrfs patches from upstream Linux kernel to RHEL 7 kernel, but didn't obtain significant performance improvement
- Build system
  - All 'btrfs' in the code is renamed to 'lbtrfs' to avoid conflicts with Linux kernel
  - lbtrfs module is built separately (similar to ldiskfs)
  - Two additional RPMs are generated: lustre-osd-btrfs and lustre-osd-btrfs-mount
- Utility support for formatting/mounting OSD with Btrfs has been added
  - RAID features of Btrfs are available to the OSD

OSD-Btrfs Implementation (2)

- Object Index
  - Internal to the OSD, an object index table is used to map the File Identifier(FID) to the local inode number.
  - A new item type in Btrfs is added to map FID to Btrfs inode ID/generation
  - This implementation introduces less metadata overhead when compared to directory-based implementation
  - But it requires changes to btrfsck to support the new item
- Transaction callback
  - Lustre needs transaction callback to release OSD internal resources after a transaction is committed
  - We have applied a patch to Btrfs to enable transaction callback
- Object Operations
  - Interfaces for creation/deletion/write/read/… objects are added by patching Btrfs
- Item numbers of transactions
  - Item numbers required by different operation types need to be calculated carefully

OSD-Btrfs Implementation (3)

- Zero Copy I/O
  - Filesystem buffers are accessed directly in order to bypass memory copies to achieve good streaming I/O
  - Pages are locked during the access of buffers
  - Btrfs codes are patched to avoid releasing locked pages in this process
- Free inode number in Btrfs is missing because inodes are allocated dynamically
  - Lustre needs sufficient free inode numbers to pre-create objects
  - We have applied a patch to estimate free inode numbers according to currently unused blocks and free metadata space
  - Object pre-creation could still fail during operation because actual data exhausts the available space
  - Possible solution: reserve inodes for object pre-creation?

Future plans

- Btrfs OSD functional enhancements
  - Add Xattr operations of objects
  - Update btrfsck due to disk format change of object Index
- Regression and stability testing
- Contributions to the Btrfs and Lustre communities
- Evaluation of Btrfs-OSD for MDT

Benchmark Set-up

- Primary focused on throughput performance
- Test Configuration
  - SFA7700, 80 x 7200RPM 4TB NL-SAS
    - 8 x RAID6(8D+2P)
    - 8 x OST(4 OST per OSS)
  - 2 x OSS and 1 x MDS
    - 2 x Intel Xeon CPU E5-2680v2
    - 128GB Memory
    - FDR Infiniband HCA
  - 32 x Clients
    - 2 x Intel Xeon CPU E5-2660v2
    - 128GB Memory
    - FDR Infiniband HCA
  - RHEL7.1
  - Lustre-2.7.50 (master branch)
  - Mellanox OFED 2.4.0

Setup Lustre with OSD-Btrfs

- Install Lustre RPMs except lustre-osd-btrfs-* RPMs
- Create MDT on ldiskfs and mount it
- Format OST with OSD-Btrfs
  - Simple syntax

    # mkfs.lustre --mgsnode=10.128.0.167@o2ib --ost \
    --backfstype=lbtrfs --index=0 --fsname=lustre /dev/ost0
  - Format OST With Btrfs RAID options

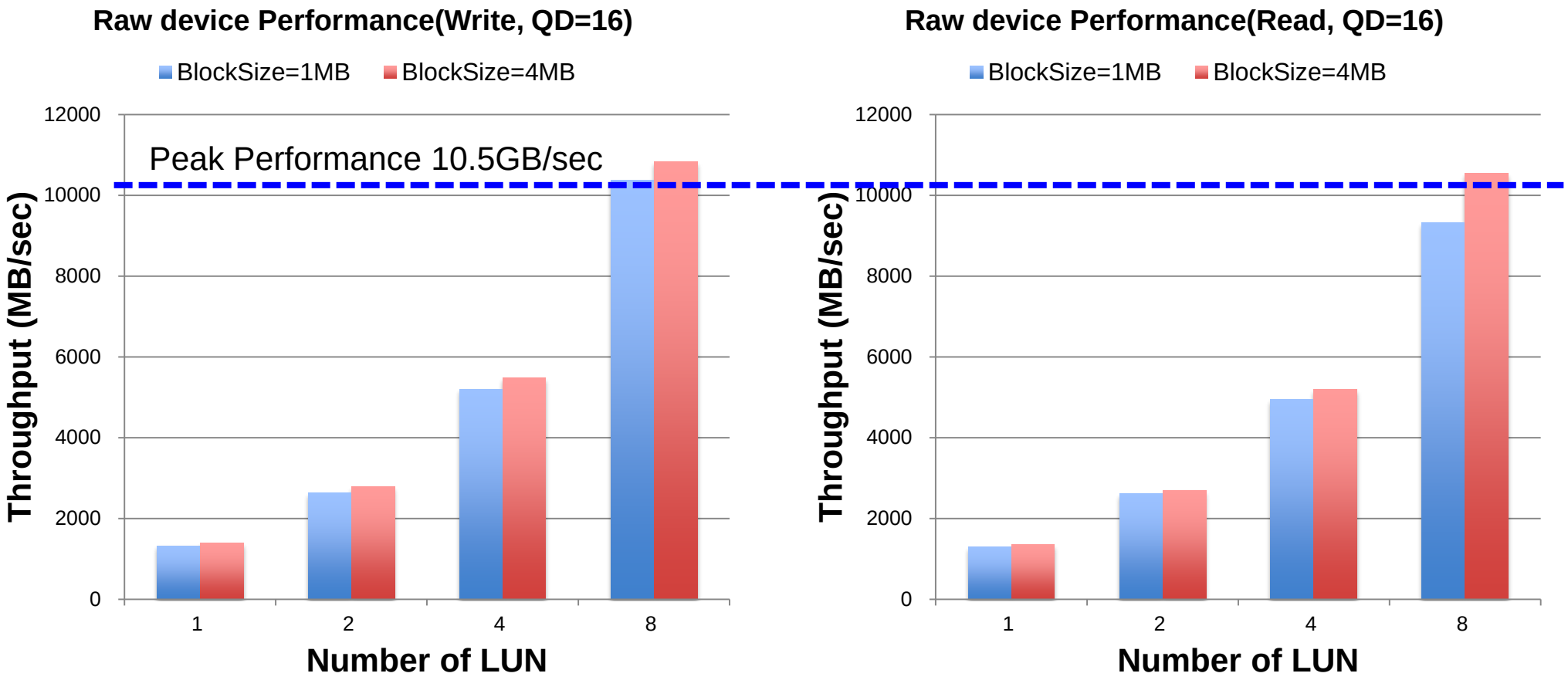
    # mkfs.lustre --mgsnode=10.128.0.167@o2ib --ost \
    --backfstype=lbtrfs --index=0 --fsname=lustre \
    --mkfsoptions="-m raid0 -d raid0” /dev/mapper/ost0 /dev/mapper/ost1
  - Mount OST

    # mount –t lustre /dev/mapper/ost0

Hardware performance

Two servers, 80xNL-SAS Drives

- xdd to raw devices
  - Sequential Write and Read with O_DIRECT (“-dio -seek sequential”)



Preliminary Performance Results(1)

2 OSSs, 8 OSTs (80xNL-SAS)

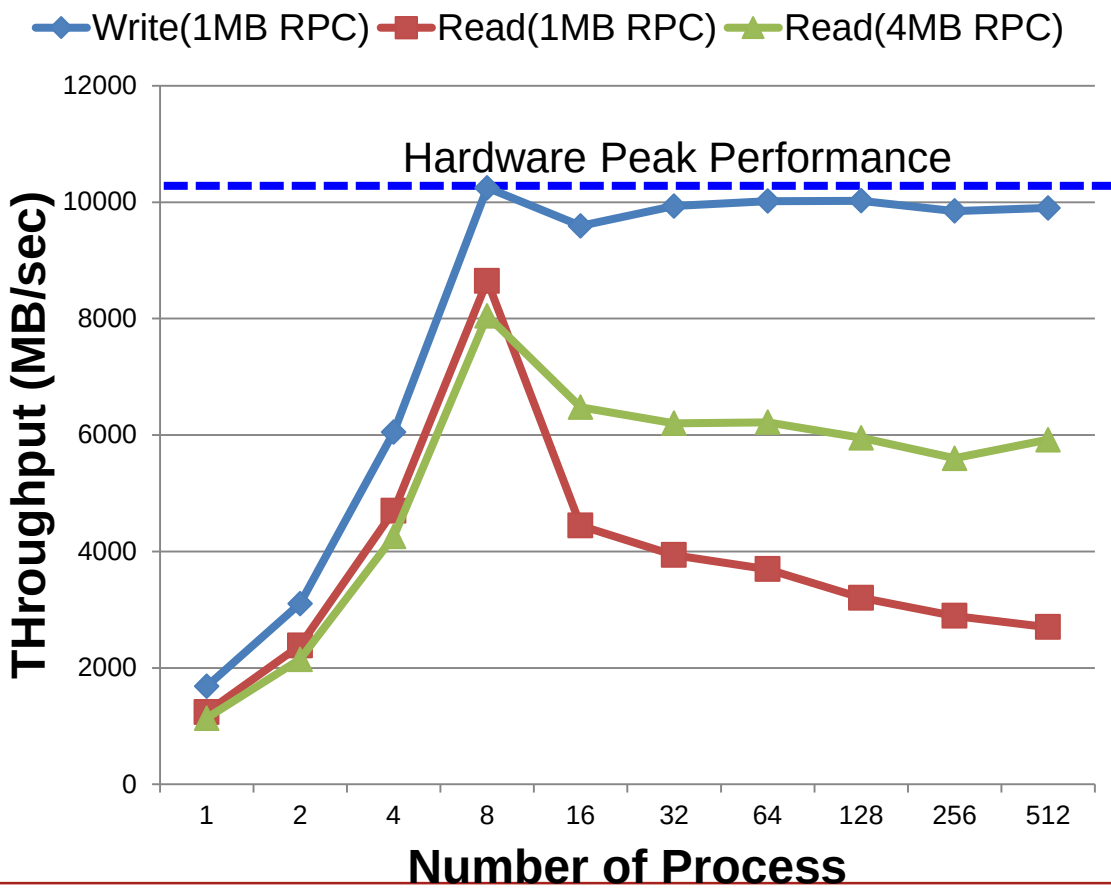
DataDirect NETWORKS
Information in Motion

IOR -w -k -t 1m -b ${size}g -vv -e -g -F –o /lustre/file (Write)

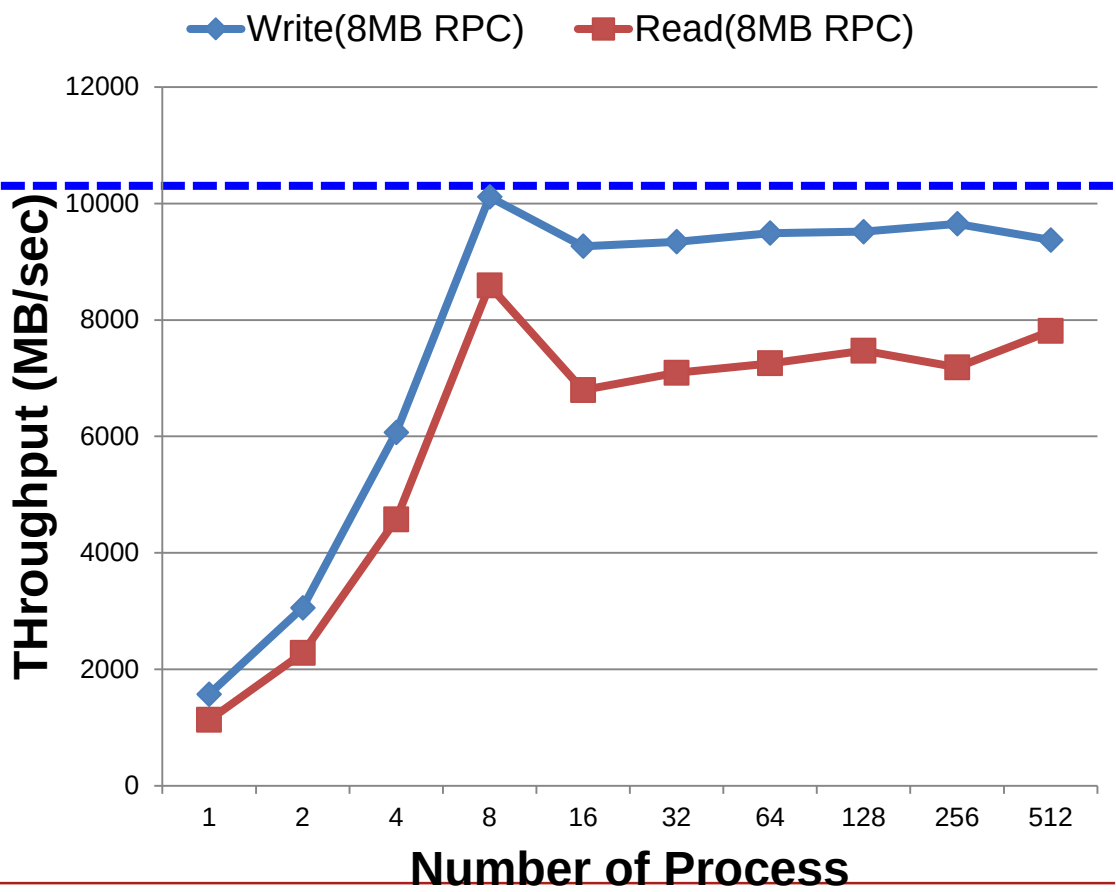
Flush all page caches on OSS and clients after write test.

IOR -r -k -t 1m -b ${size}g -vv -e -g -F –o /lustre/file (Read)

IOR (FPP, Sequential, 1MB/4MB RPC)

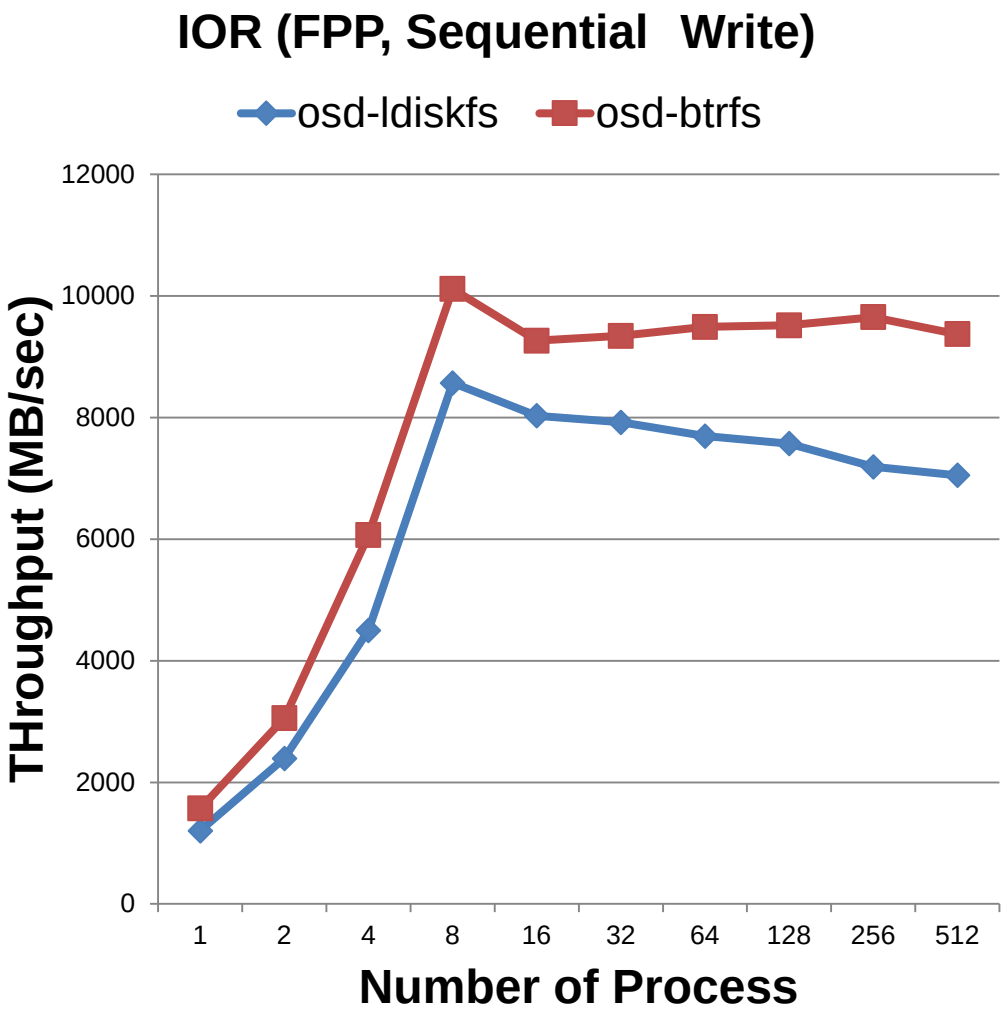


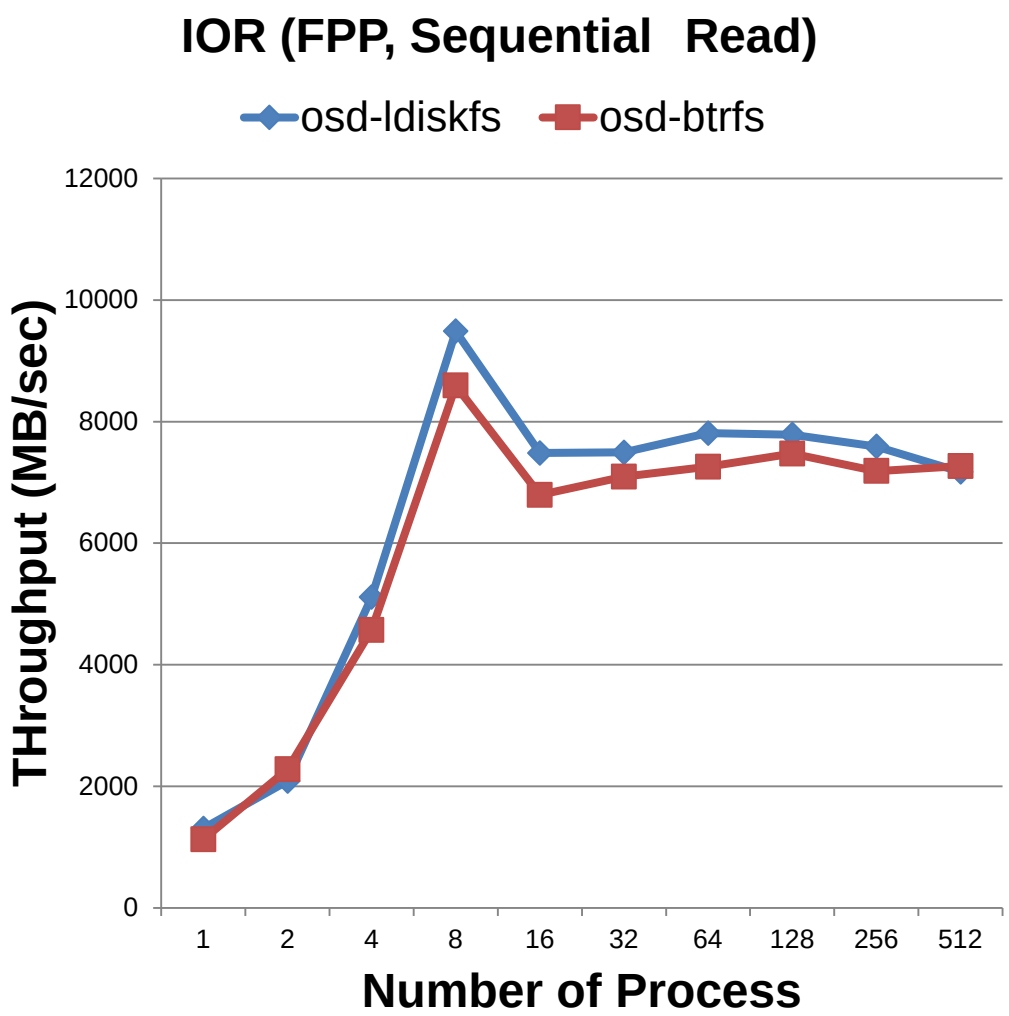
IOR (FPP, Sequential, 8MB RPC)



Preliminary Performance Results(2) OSD-ldiskfs and OSD-Btrfs

DataDirect NETWORKS
INFORMATION IN MOTION

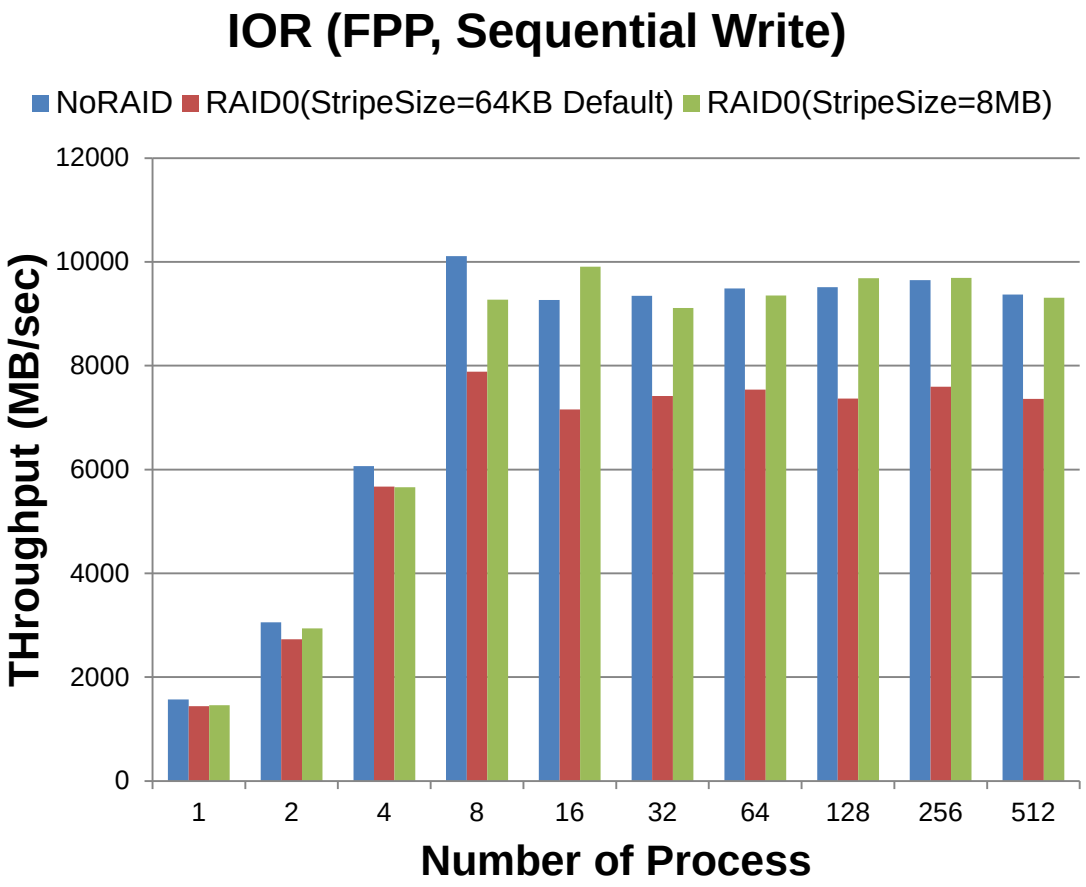


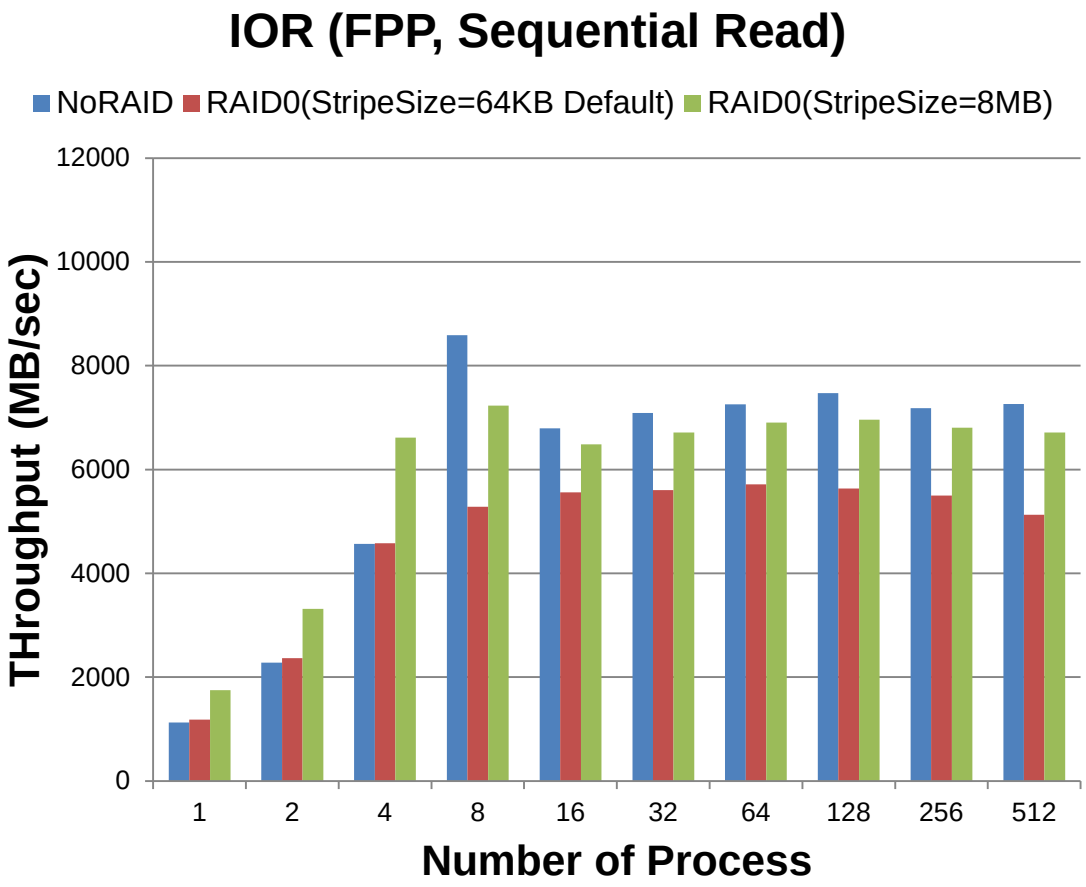


Preliminary Performance Results(3)

Btrfs-RAID0, 2 OSSs, 4 OST(80xNL-SAS)

- Setup Btrfs-RAID0 top of RAID6 volumes
  - Two RAID6 volumes are striped by Bbtrfs-RAID0 (8 x OST -> 4 x OST)





Btrfs's volume stripe size(BTRFS_STRIPE_LEN) is fixed (64KB). Just increased it to 8MB.

Conclusions

- We designed a Btrfs-OSD and implemented a prototype based on the existing Lustre OSD framework
- Initial benchmark results are attractive and encouraging, despite the fact that we have spent very little time on performance optimization yet
- More fundamental benchmarks do determine the usefulness of Btrfs-OSD as OST device are necessary (e.g. file creation, file deletion, file re-writing)