



Lustre Data on MDT – an early look

LUG 2018 - April 2018

[Argonne National Laboratory](#)

Frank Leers

DDN Performance Engineering

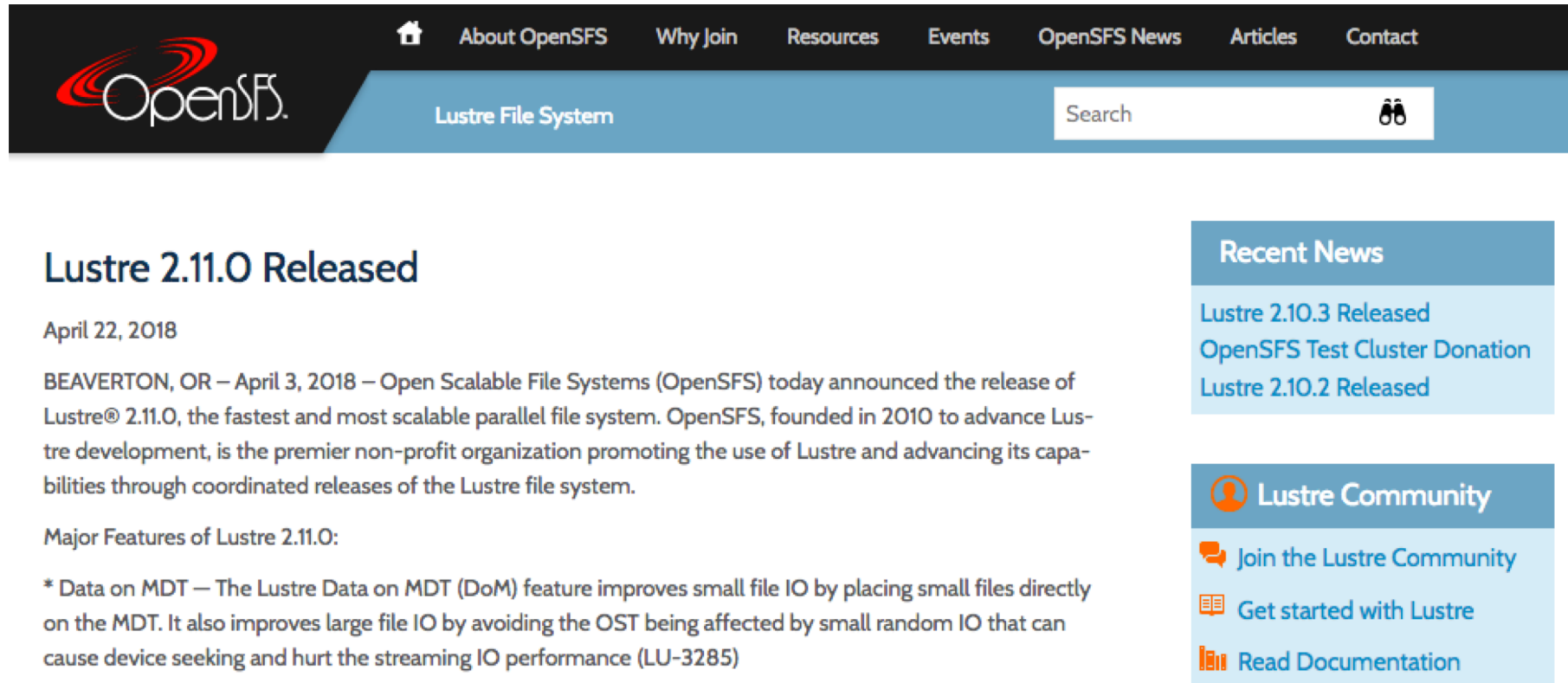
Agenda

- ▶ DoM Overview
- ▶ Practical Usage
- ▶ Performance Investigation
 - Test Environment
 - Test Cases
- ▶ Next Steps



Data on MDT Overview

DoM Overview – Release Announcement



The screenshot shows the OpenSFS website header with navigation links: Home, About OpenSFS, Why Join, Resources, Events, OpenSFS News, Articles, and Contact. Below the header is a blue banner with the text "Lustre File System" and a search bar. The main content area features a large heading "Lustre 2.11.0 Released" with a date "April 22, 2018". The text below the heading states: "BEAVERTON, OR – April 3, 2018 – Open Scalable File Systems (OpenSFS) today announced the release of Lustre® 2.11.0, the fastest and most scalable parallel file system. OpenSFS, founded in 2010 to advance Lustre development, is the premier non-profit organization promoting the use of Lustre and advancing its capabilities through coordinated releases of the Lustre file system." Below this is a section titled "Major Features of Lustre 2.11.0:" followed by a bullet point: "* Data on MDT — The Lustre Data on MDT (DoM) feature improves small file IO by placing small files directly on the MDT. It also improves large file IO by avoiding the OST being affected by small random IO that can cause device seeking and hurt the streaming IO performance (LU-3285)". On the right side of the page, there are two sidebars. The top sidebar is titled "Recent News" and lists three items: "Lustre 2.10.3 Released", "OpenSFS Test Cluster Donation", and "Lustre 2.10.2 Released". The bottom sidebar is titled "Lustre Community" and lists three items: "Join the Lustre Community", "Get started with Lustre", and "Read Documentation".

Lustre 2.11.0 Released

April 22, 2018

BEAVERTON, OR – April 3, 2018 – Open Scalable File Systems (OpenSFS) today announced the release of Lustre® 2.11.0, the fastest and most scalable parallel file system. OpenSFS, founded in 2010 to advance Lustre development, is the premier non-profit organization promoting the use of Lustre and advancing its capabilities through coordinated releases of the Lustre file system.

Major Features of Lustre 2.11.0:

- * Data on MDT — The Lustre Data on MDT (DoM) feature improves small file IO by placing small files directly on the MDT. It also improves large file IO by avoiding the OST being affected by small random IO that can cause device seeking and hurt the streaming IO performance (LU-3285)

Recent News

- Lustre 2.10.3 Released
- OpenSFS Test Cluster Donation
- Lustre 2.10.2 Released

Lustre Community

- Join the Lustre Community
- Get started with Lustre
- Read Documentation

“The Lustre Data-on-MDT (DoM) feature improves small file IO by placing small files on the MDT. DoM also improves large file IO by avoiding the OST being affected by small random IO that can cause device seeking and hurt the streaming IO performance.”

DoM Overview – Benefits and Definition

► Optimize small file I/O

- Avoid OST RPC overhead

“The Data On MDT (DOM) project aims to improve small file performance by allowing the data for small files to be placed only on the MDT, so that additional RPCs and I/O overhead can be eliminated, and performance correspondingly improved. “

- Take advantage of flash-based MDT for small(er) files

“Since the MDT storage is typically configured as high-IOPS RAID-1+0 and optimized for small IO, Data-on-MDT will also be able to leverage this faster storage.”

- Separates small-block workload from HDD-based sequential large(er)-block

► Works with DNE, PFL

► Feature parity (or better) with other PFS' similar feature

- i.e. Spectrum Scale *large* inodes, which can hold ~3400 bytes of file data in a 4KB inode



DoM – Practical Usage

Functionality – dom_stripesize

Functionality – dom_stripesize – Defaults / Changing / Query

The LOD parameter **dom_stripesize** is used to control per MDS maximum size for a DoM component on a given MDT.

- Default dom_stripesize is 1MB
- Minimum dom_stripesize is 64KB
- Maximum dom_stripesize is 1GB

Changing dom_stripesize:

```
# lctl set_param lod.*MDT<index>*.dom_stripesize=<value in bytes>
```

```
# lctl conf_param lod.*MDT<index>*.dom_stripesize=<value in bytes>
```

Checking dom_stripesize:

```
# lctl get_param lod.*MDT<index>*.dom_stripesize
```


Functionality – dom_stripesize cont'd. – Change Example

Changing dom_stripesize

```
# clush -bgmds 'lctl set_param -n lod.*MDT000*.dom_stripesize=65536'
```

```
# clush -bgmds 'lctl get_param -n lod.*MDT000*.dom_stripesize'
```

```
-----  
mds[0-1] (2)  
-----
```

```
65536  
65536  
65536  
65536
```

```
# clush -bgmds 'lctl set_param -n lod.*MDT000*.dom_stripesize=32768'
```

```
#      ← no complaints
```

```
# clush -bgmds 'lctl get_param -n lod.*MDT000*.dom_stripesize'
```

```
-----  
mds[0-1] (2)  
-----
```

```
1048576  
1048576  
1048576  
1048576
```

Note: We didn't see an error when attempting to set dom_stripesize lower, i.e. 32KB – but this results in the value getting reset to the default of 1MB. See example above.

The next "lfs setstripe -L mdt ..." operation shows that the value is indeed now set to 1MB

Also, setting dom_stripesize to a non-64KB-aligned value will round down to the nearest 64KB boundary.

Functionality – dom_stripesize cont'd. – more Change Examples

128KB and 4MB are valid. > 1GB is not supported

```
# lctl set_param lod.*MDT000*.dom_stripesize=131072
# lctl get_param -n lod.*MDT000*.dom_stripesize
131072            (128KB)

# lctl set_param lod.*MDT000*.dom_stripesize=$((1048576*4))
# lctl get_param -n lod.*MDT000*.dom_stripesize
4194304            (4MB)
```

```
# lctl set_param lod.*MDT000*.dom_stripesize=$((1048576*1024))
# lctl get_param -n lod.*MDT000*.dom_stripesize
1073741824        (1024MB)                    OK, 1GB works!

# lctl set_param lod.*MDT0000*.dom_stripesize=$((1048576*1025))
error: set_param: setting /proc/fs/lustre/lod/domfs-MDT0000-mdtlov/dom_stripesize=1074790400:
Numerical result out of range

# lctl get_param -n lod.*MDT0000*.dom_stripesize
1073741824        (1GB)                    1025MB is too Big, but 1GB value retained
```

Functionality – dom_stripesize cont'd. – turning it off

Disable for all MDTs, then verify

```
# clush -bgmds 'lctl set_param lod.*MDT000*.dom_stripesize=0'
# clush -bgmds 'lctl get_param -n lod.*MDT000*.dom_stripesize'
-----
mds[0-1] (2)
-----
0
0
0
0
```

Can we create a new DoM-backed file when dom_stripesize is zero?

```
# lfs setstripe -L mdt -E1M DNE.2c.file1
lfs setstripe: cannot create composite file 'DNE.2c.file1': Invalid argument
```

No!

Functionality – dom_stripesize cont'd. - procfs

There are procfs entries:

```
# clush -bgmds 'find /proc/fs/lustre/ -name stripesize'
-----
mds0
-----
/proc/fs/lustre/lod/domfs-MDT0000-mdtlov/stripesize
/proc/fs/lustre/lod/domfs-MDT0001-mdtlov/stripesize
/proc/fs/lustre/lod/domfs-MDT0002-mdtlov/stripesize
/proc/fs/lustre/lod/domfs-MDT0003-mdtlov/stripesize
/proc/fs/lustre/lov/domfs-clilov-ffff883f76242000/stripesize
-----
mds1
-----
/proc/fs/lustre/lod/domfs-MDT0004-mdtlov/stripesize
/proc/fs/lustre/lod/domfs-MDT0005-mdtlov/stripesize
/proc/fs/lustre/lod/domfs-MDT0006-mdtlov/stripesize

# clush -bgmds 'cat /proc/fs/lustre/lod/domfs-MDT000*-mdtlov/dom_stripesize'
-----
mds[0-1] (2)
-----
0
0
0
0
```

Read-Only – echoing a value into stripesize doesn't effect change

We heard from James Simmons on Tuesday, that /proc entries for lustre are going away, so

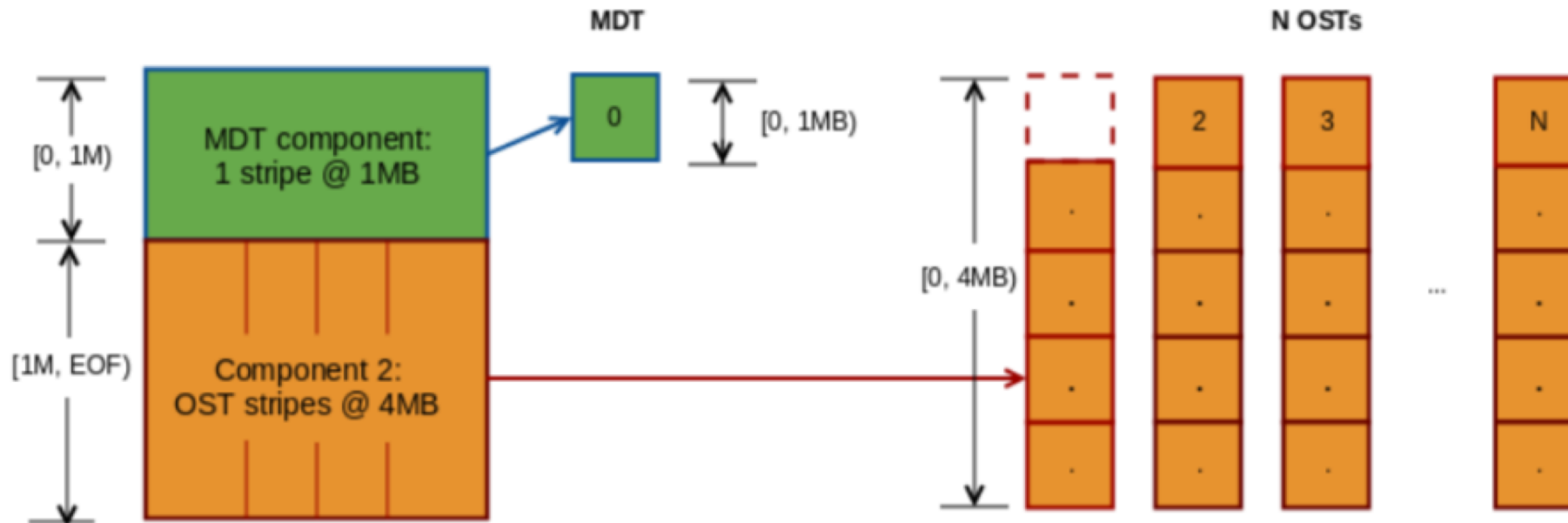
Components and Striping

DoM Overview – Component Layout

► PFL for Data on MDT

- DoM layout is organized as a Progressive File Layout, which occupies the first entry describing the MDT stripe. The PFL semantics enable a DoM file to grow into OSTs.
- In the example diagram below, Component 1 is a 1MB stripe, component 2 covers 1MB -> EOF, and is striped over all available OSTs.

```
# lfs setstripe -L mdt -E 1M -E -1 -S 4M fubar
```



Usage – lfs (setstripe) options used with DoM

LFS-SETSTRIPE(1)

Lustre Utilities

LFS-SETSTRIPE(1)

NAME

lfs setstripe - set striping pattern of a file or directory default

SYNOPSIS

lfs setstripe [--component-end|-E end1] [STRIPE_OPTIONS] [{--component-end|-E end2} [STRIPE_OPTIONS] ...] <filename>

Create a file with the specified composite layout.

Each component defines the stripe pattern of the file in the range of [start, end].

The first component implicitly starts at offset 0, and all later components start at the end of previous extent.

The -E option is used to specify the end offset of each component, and it also indicates the following

STRIPE_OPTIONS are for this component. The end offset of -1 or EOF indicates the component extends to end of file.

<STRIPE_OPTIONS for DoM>

-L, --layout <layout type>

The type of stripe layout, can be RAID0, released or **mdt**. It is raid0 by default.

The mdt type allows place the first component of the file on the MDT where the inode is located.

This is used with composite file layouts and can be defined as first component only.

The stripe_size of MDT part is always equal to the component size.

There is also per-MDT parameter lod.dom_stripe_size to limit maximum size of DoM stripe which can be changed with lctl set_param command, (e.g. lctl set_param lod.*.dom_stripe_size=0)

Usage – lfs setstripe no-no's, and ... rules

```
# lfs setstripe -E 1G -L mdt -E 1G -L mdt -E 1G -L mdt foo
```

error: 'mdt' layout can be only the first one

```
# lfs setstripe -E 1G -L mdt -c 2 foo
```

Option 'stripe-count' can't be specified with Data-on-MDT component: 2

```
# lfs setstripe -E 1G -L mdt -S 2M foo
```

Option 'stripe-size' can't be specified with Data-on-MDT component: 2147483648

Guidelines and Restrictions

- The layout of a DoM file is stored on disk as a *composite layout* and is a special case of a PFL file.
- For DoM files, the file layout is composed of the first special component which is placed on MDT and rest of components on OSTs.
- The first component has MDT layout which is a new layout type, and is placed on an MDT in MDT object data blocks.
- MDT component is always a single stripe, with size equal to the component size - no `-S` (stripe size) or `-c` (stripe count) arguments.
- The rest of the components are placed over OSTs as usual with RAID0 layout.
- The OST components are not instantiated until file growth beyond the limit of MDT layout.
- If a DoM layout is set on an existing directory, all subsequent files created will inherit this layout.

Usage – lfs find is DoM-aware! - Examples:

Find all files with DoM layout

```
# lfs find -L mdt . | wc -l  
3200362
```

Find DoM files with a 1M stripe size

```
# lfs find DOM-7-MDT -L mdt -S 1M | head -8  
DOM-7-MDT  
DOM-7-MDT/mdtestdir  
DOM-7-MDT/mdtestdir/#test-dir.1  
DOM-7-MDT/mdtestdir/#test-dir.1/mdtest_tree.95.0  
DOM-7-MDT/mdtestdir/#test-dir.1/mdtest_tree.95.0/file.mdtest.95.3412  
DOM-7-MDT/mdtestdir/#test-dir.1/mdtest_tree.95.0/file.mdtest.95.2269  
DOM-7-MDT/mdtestdir/#test-dir.1/mdtest_tree.95.0/file.mdtest.95.7863  
DOM-7-MDT/mdtestdir/#test-dir.1/mdtest_tree.95.0/file.mdtest.95.1351
```

Find DoM directories with stripe size less than 180K

```
# lfs find -L mdt -S -180K -type d .  
./128K  
./64K
```

```
# lfs getstripe DOM-7-MDT/mdtestdir/#test-  
dir.1/mdtest_tree.95.0/file.mdtest.95.3412  
  
DOM-7-MDT/mdtestdir/#test-  
dir.1/mdtest_tree.95.0/file.mdtest.95.3412  
lcm_layout_gen: 1  
lcm_mirror_count: 1  
lcm_entry_count: 1  
lcme_id: 1  
lcme_mirror_id: 0  
lcme_flags: init  
lcme_extent.e_start: 0  
lcme_extent.e_end: 1048576  
lmm_stripe_count: 0  
lmm_stripe_size: 1048576  
lmm_pattern: mdt  
lmm_layout_gen: 0  
lmm_stripe_offset: 0
```

Let's see a few lfs setstripe usage examples

File example – data only on MDT – 1MB

```
# lfs setstripe -L mdt -E 1M DoM.1c.file1
```

1. 0-1MB on DoM MDT

DoM.1c.file1

```
lcm_layout_gen: 1
lcm_mirror_count: 1
lcm_entry_count: 1
lcme_id: 1
lcme_mirror_id: 0
lcme_flags: init
lcme_extent.e_start: 0
lcme_extent.e_end: 1048576
lmm_stripe_count: 0
lmm_stripe_size: 1048576
lmm_pattern: mdt
lmm_layout_gen: 0
lmm_stripe_offset: 0
```

what if we write past the 1MB

EOF?

```
# dd if=/dev/zero of=DoM.1c.file1 bs=1M count=10
dd: error writing 'DoM.1c.file1': No data available
2+0 records in
1+0 records out
1048576 bytes (1.0 MB) copied, 0.00286129 s, 366 MB/s
```

```
# ls -lh DoM.1c.file1
-rw-r--r-- 1 root root 1.0M Apr 13 13:42 DoM.1c.file1
```

File example – data on 1MDT for 1MB, then restricted to 1 OST

lfs setstripe -L mdt -E 1M -E -1 -S 1M DoM.2c.file1

1. 0-1MB on DoM MDT
2. 1MB-EOF – restrict to 1 OST at 1MB stripe

```
DoM.2c.file1
lcm_layout_gen: 2
lcm_mirror_count: 1
lcm_entry_count: 2
  lcme_id: 1
  lcme_mirror_id: 0
  lcme_flags: init
  lcme_extent.e_start: 0
  lcme_extent.e_end: 1048576
    lmm_stripe_count: 0
    lmm_stripe_size: 1048576
    lmm_pattern: mdt
    lmm_layout_gen: 0
    lmm_stripe_offset: 0
  -----
  lcme_id: 2
  lcme_mirror_id: 0
  lcme_flags: 0
  lcme_extent.e_start: 1048576
  lcme_extent.e_end: EOF
    lmm_stripe_count: 1
    lmm_stripe_size: 1048576
    lmm_pattern: raid0
    lmm_layout_gen: 0
    lmm_stripe_offset: -1
```

```
DoM.2c.file1
lcm_layout_gen: 3
lcm_mirror_count: 1
lcm_entry_count: 2
  lcme_id: 1
  lcme_mirror_id: 0
  lcme_flags: init
  lcme_extent.e_start: 0
  lcme_extent.e_end: 1048576
    lmm_stripe_count: 0
    lmm_stripe_size: 1048576
    lmm_pattern: mdt
    lmm_layout_gen: 0
    lmm_stripe_offset: 0
  -----
  lcme_id: 2
  lcme_mirror_id: 0
  lcme_flags: init
  lcme_extent.e_start: 1048576
  lcme_extent.e_end: EOF
    lmm_stripe_count: 1
    lmm_stripe_size: 1048576
    lmm_pattern: raid0
    lmm_layout_gen: 0
    lmm_stripe_offset: 8
    lmm_objects:
      - 0: { l_ost_idx: 8, l_fid: [0x100080000:0x349639:0x0] }
```

...write some data, get an object

File example – data on 1MDT for 1GB, then stripe across all OSTs

```
# lfs setstripe -L mdt -E 1G -E -1 -S 4M -c -1 DoM.2c.file2
```

DoM.2c.file2

```
lcm_layout_gen: 2
lcm_mirror_count: 1
lcm_entry_count: 2
lcme_id: 1
lcme_mirror_id: 0
lcme_flags: init
lcme_extent.e_start: 0
lcme_extent.e_end: 1073741824
lmm_stripe_count: 0
lmm_stripe_size: 1073741824
lmm_pattern: mdt
lmm_layout_gen: 0
lmm_stripe_offset: 0
```

```
lcme_id: 2
lcme_mirror_id: 0
lcme_flags: init
lcme_extent.e_start: 1073741824
lcme_extent.e_end: EOF
lmm_stripe_count: 14
lmm_stripe_size: 4194304
lmm_pattern: raid0
lmm_layout_gen: 0
lmm_stripe_offset: 10
lmm_objects:
- 0: { l_ost_idx: 10, l_fid: [0x1000a0000:0x349338:0x0] }
- 1: { l_ost_idx: 0, l_fid: [0x100000000:0x345c66:0x0] }
- 2: { l_ost_idx: 7, l_fid: [0x100070000:0x349679:0x0] }
- 3: { l_ost_idx: 3, l_fid: [0x100030000:0x3475e9:0x0] }
- 4: { l_ost_idx: 9, l_fid: [0x100090000:0x346129:0x0] }
- 5: { l_ost_idx: 2, l_fid: [0x100020000:0x3471a9:0x0] }
- 6: { l_ost_idx: 13, l_fid: [0x1000d0000:0x346829:0x0] }
- 7: { l_ost_idx: 5, l_fid: [0x100050000:0x364209:0x0] }
- 8: { l_ost_idx: 11, l_fid: [0x1000b0000:0x348749:0x0] }
- 9: { l_ost_idx: 6, l_fid: [0x100060000:0x345409:0x0] }
- 10: { l_ost_idx: 12, l_fid: [0x1000c0000:0x349899:0x0] }
- 11: { l_ost_idx: 1, l_fid: [0x100010000:0x34b789:0x0] }
- 12: { l_ost_idx: 8, l_fid: [0x100080000:0x34963a:0x0] }
- 13: { l_ost_idx: 4, l_fid: [0x100040000:0x34d338:0x0] }
```

1. 0-1GB on DoM MDT
2. 1GB-EOF – across all OSTs at 4MB stripes

14 OSTs preset in this filesystem

DoM + PFL example – 4 components: DoM, 1/4/all OSTs

lfs setstripe -E 1M -L mdt -E 64M -c1 -S 1M -E 8G -c4 -E -1 -c -1 -S 4M DNE.PFL.file1

```
DoM.PFL.file1
lcm_layout_gen: 6
lcm_mirror_count: 1
lcm_entry_count: 4
lcme_id: 1
lcme_mirror_id: 0
lcme_flags: init
lcme_extent.e_start: 0
lcme_extent.e_end: 1048576
lmm_stripe_count: 0
lmm_stripe_size: 1048576
lmm_pattern: mdt
lmm_layout_gen: 0
lmm_stripe_offset: 0
-----
lcme_id: 2
lcme_mirror_id: 0
lcme_flags: init
lcme_extent.e_start: 1048576
lcme_extent.e_end: 67108864
lmm_stripe_count: 1
lmm_stripe_size: 1048576
lmm_pattern: raid0
lmm_layout_gen: 0
lmm_stripe_offset: 10
lmm_objects:
- 0: { l_ost_idx: 10, l_fid: [0x1000a0000:0x349339:0x0] }
-----
lcme_id: 3
lcme_mirror_id: 0
lcme_flags: init
lcme_extent.e_start: 67108864
lcme_extent.e_end: 8589934592
lmm_stripe_count: 4
lmm_stripe_size: 1048576
lmm_pattern: raid0
lmm_layout_gen: 0
lmm_stripe_offset: 0
lmm_objects:
- 0: { l_ost_idx: 0, l_fid: [0x100000000:0x345c67:0x0] }
- 1: { l_ost_idx: 7, l_fid: [0x100070000:0x34967a:0x0] }
- 2: { l_ost_idx: 3, l_fid: [0x100030000:0x3475ea:0x0] }
- 3: { l_ost_idx: 9, l_fid: [0x100090000:0x34612a:0x0] }
```

```
lcme_id: 4
lcme_mirror_id: 0
lcme_flags: init
lcme_extent.e_start: 8589934592
lcme_extent.e_end: EOF
lmm_stripe_count: 14
lmm_stripe_size: 4194304
lmm_pattern: raid0
lmm_layout_gen: 0
lmm_stripe_offset: 13
lmm_objects:
- 0: { l_ost_idx: 13, l_fid: [0x1000d0000:0x34682a:0x0] }
- 1: { l_ost_idx: 5, l_fid: [0x100050000:0x36420a:0x0] }
- 2: { l_ost_idx: 11, l_fid: [0x1000b0000:0x34874a:0x0] }
- 3: { l_ost_idx: 6, l_fid: [0x100060000:0x34540a:0x0] }
- 4: { l_ost_idx: 12, l_fid: [0x1000c0000:0x34989a:0x0] }
- 5: { l_ost_idx: 1, l_fid: [0x100010000:0x34b78a:0x0] }
- 6: { l_ost_idx: 8, l_fid: [0x100080000:0x34963b:0x0] }
- 7: { l_ost_idx: 4, l_fid: [0x100040000:0x34d339:0x0] }
- 8: { l_ost_idx: 2, l_fid: [0x100020000:0x3471aa:0x0] }
- 9: { l_ost_idx: 10, l_fid: [0x1000a0000:0x34933a:0x0] }
- 10: { l_ost_idx: 0, l_fid: [0x100000000:0x345c68:0x0] }
- 11: { l_ost_idx: 7, l_fid: [0x100070000:0x34967b:0x0] }
- 12: { l_ost_idx: 3, l_fid: [0x100030000:0x3475eb:0x0] }
- 13: { l_ost_idx: 9, l_fid: [0x100090000:0x34612b:0x0] }
```

1. 0-1MB on DoM MDT
2. 1-64MB on 1 stripe of 1MB stripes
3. 64M-8GB on 4 OSTs at default stripes
4. 8GB-EOF across all OSTs at 4MB stripes



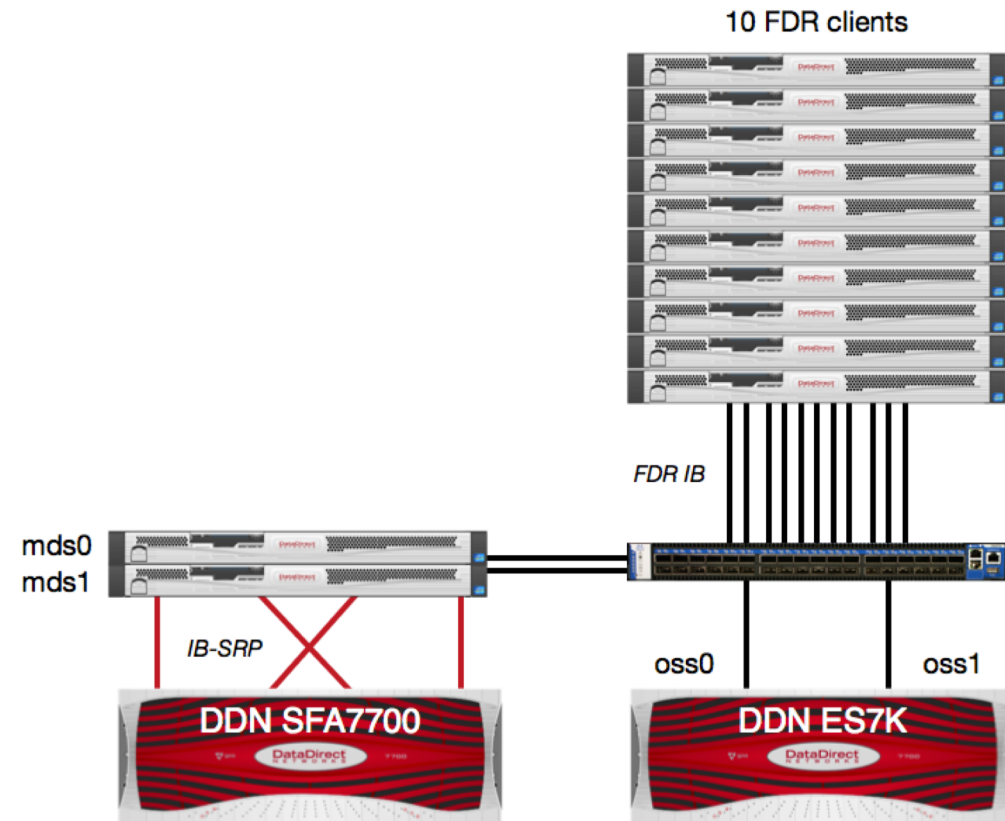
Testing – Environment, Configuration

- Collection of hardware that we tested on – description and diagrams
- Software versions and filesystem information

Test Environment - Hardware

Hardware Configuration

- Metadata
 - SFA7700-IB
 - 2 Dell R640, 2 CX-3 ea.
 - 8 RAID6 4+2 SSD MDT LUNs
 - 1 RAID1 SSD MGT LUN
- Data
 - ES7K
 - 2 VM OSS
 - 14 RAID6 8+2 NLSAS OST LUNs
- Clients
 - 10 Dell R620, 1 FDR LNET ea.
- FDR Infiniband fabric



Test Environment – Software Stack

- MDS R640 servers
 - EXAScaler **HPC** CentOS 4.0.0-r2.lustre_upstream
 - lustre-2.11.50_51_g8d37637-1.el7.x86_64
 - kernel 3.10.0-693.21.1.el7.x86_64.lustre
- ES7K vOSS
 - EXAScaler **SFA** CentOS 4.0.0-r2.lustre_upstream
 - lustre-2.11.50_51_g8d37637-1.el7.x86_64
 - kernel 3.10.0-693.21.1.el7.x86_64.lustre
- Clients
 - lustre-client-2.11.50_51_g8d37637-1.el7.centos.x86_64
 - kernel 3.10.0-693.el7.x86_64
- Common
 - CentOS Linux release 7.4.1708 (Core)
- SFA platforms
 - SFAOS 3.1.3.0 41240
- Tuning
 - osc.*.max_pages_per_rpc=16M
 - osc.*.max_rpcs_in_flight=32

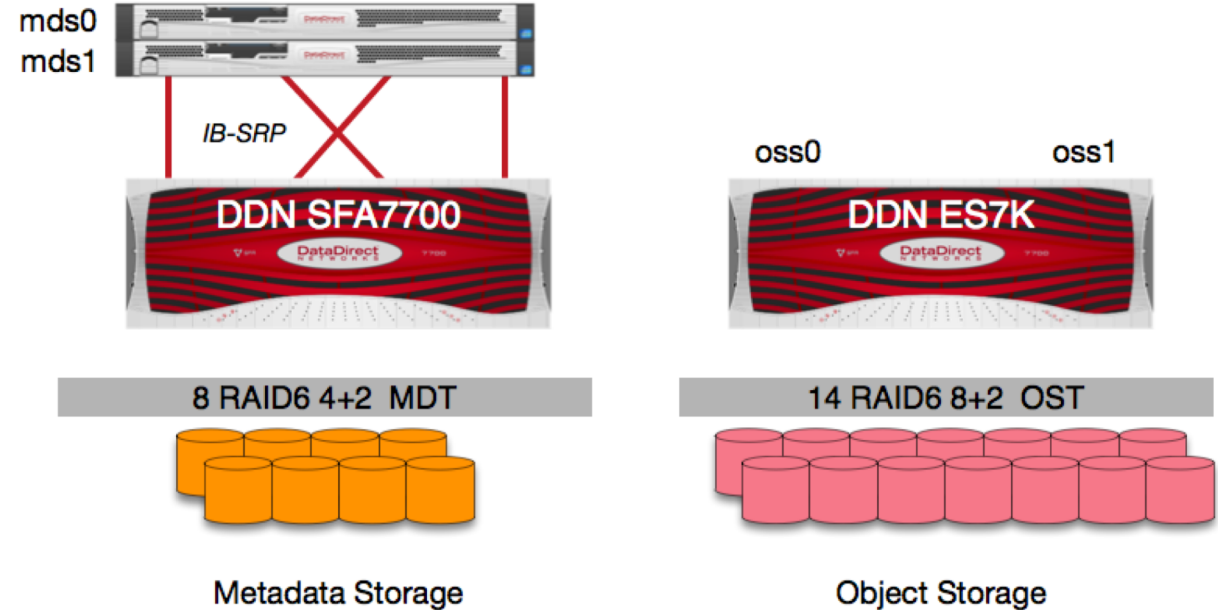


Test Case I mdtest

- **1a:** SSD non-DoM MDT + HDD OST – mdtest w/file size scaling, (no DoM)
- **1b:** SSD DoM MDT – mdtest w/file size scaling, no OST I/O

Test Case 1 mdtest – storage allocation

- Metadata – MDT
 - 8 RAID6 4+2 SSD MDT LUNs
- Data – OST
 - 14 RAID6 8+2 NLSAS OST LUNs



Test case 1a: mdtest – non-DoM MDT, HDD OST

Test case 1a: mdtest - non-DoM MDT + SAS OST – details

Set up directories. Create 1 per MDT using *lfs mkdir -l <index>*

```
for i in 0 1 2 3 4 5 6 7; do lfs mkdir -i ${i} MDT${i}; done
```

This leaves us with

```
# for i in 0 1 2 3 4 5 6 7
> do index=`lfs getdirstripe --mdt-index MDT${i}`
> echo "directory MDT${i} has master index ${index}"
> done
```

```
directory MDT0 has master index 0
directory MDT1 has master index 1
directory MDT2 has master index 2
directory MDT3 has master index 3
directory MDT4 has master index 4
directory MDT5 has master index 5
directory MDT6 has master index 6
directory MDT7 has master index 7
```

8 directories, each linked to one of our 8 MDTs

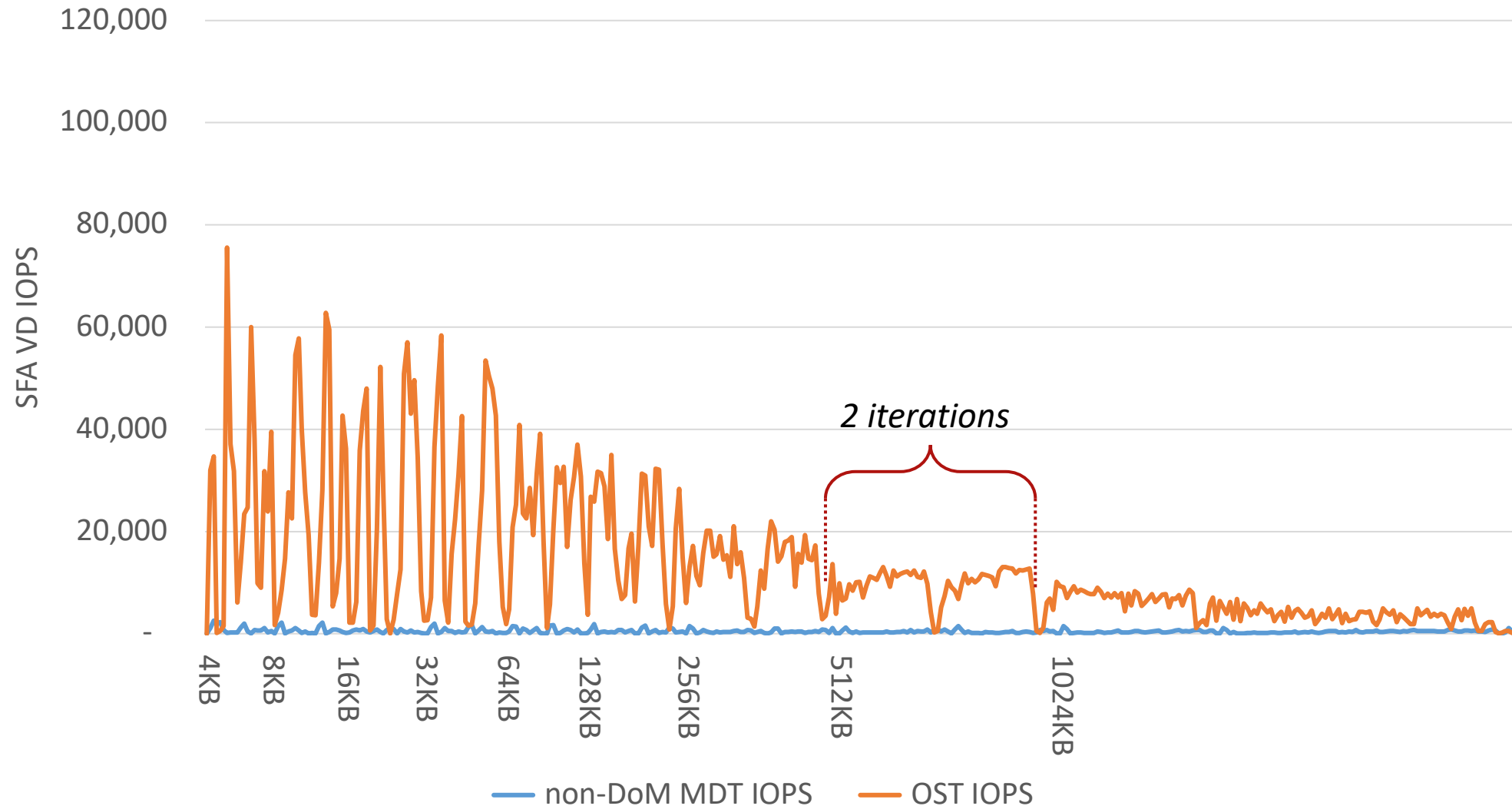
mdtest run parameters for 10 clients, 16PPN

```
mpirun --machinefile 10.txt \
--map-by ppr:16:node \
```

```
mdtest -v \
-i 2 \
-n 10000 \
-w ${FILESZ} \
-u -F -C -T -r \
-d /domfs/MDT0/mdtestdir@
/domfs/MDT1/mdtestdir@
/domfs/MDT2/mdtestdir@
/domfs/MDT3/mdtestdir@
/domfs/MDT4/mdtestdir@
/domfs/MDT5/mdtestdir@
/domfs/MDT6/mdtestdir@
/domfs/MDT7/mdtestdir
```

Test case 1a: mdtest - non-DoM MDT + SAS OST – IOPS

mdtest: scale bytes written (-w <bytes>)
8 * non-DoM MDT + 14 NLSAS OST - SFA VD IOPS



Test case 1b: mdtest – DoM MDT only

Test case 1b: mdtest - DoM MDT – details

Setup directories

```
# for i in 0 1 2 3 4 5 6 7
```

```
> do
```

```
> lfs mkdir -i ${i} DOM-${i}-MDT
```

```
> lfs setstripe -L mdt --component-end 1M DOM-${i}-MDT
```

```
> done
```

lfs mkdir with MDT index

lfs setstripe for MDT-only DoM component, 1MB max

Leaves us with ...

```
# lfs getstripe DOM-?-MDT
```

```
DOM-0-MDT
```

```
.
```

```
.
```

```
DOM-7-MDT
```

```
    lcme_extent.e_end:    1048576
```

```
    lcme_extent.e_start:  0
```

```
    lcme_flags:           0
```

```
    lcme_id:              N/A
```

```
    lcme_mirror_id:       N/A
```

```
lcm_entry_count:         1
```

```
lcm_layout_gen:          0
```

```
lcm_mirror_count:        1
```

```
    stripe_count:         0
```

```
    stripe_size:          1048576
```

```
    pattern:              mdt
```

```
    stripe_offset:        -1
```

mdtest run parameters for 10 clients, 16PPN

```
mpirun --machinefile 10.txt \
```

```
--map-by ppr:16:node \
```

```
mdtest -i 2 -v \
```

```
-n 10000 \
```

```
-w ${FILESZ} \
```

```
-u -F -C -T -r \
```

```
-d /domfs/DOM-0-MDT/mdtestdir@
```

```
/domfs/DOM-1-MDT/mdtestdir@
```

```
/domfs/DOM-2-MDT/mdtestdir@
```

```
/domfs/DOM-3-MDT/mdtestdir@
```

```
/domfs/DOM-4-MDT/mdtestdir@
```

```
/domfs/DOM-5-MDT/mdtestdir@
```

```
/domfs/DOM-6-MDT/mdtestdir@
```

```
/domfs/DOM-7-MDT/mdtestdir
```


Test case 1b: mdtest - MDT + SAS OST – run excerpt

File size is 8192

-- started at 04/16/2018 22:19:27 --

mdtest-1.9.3 was launched with 160 total task(s)
on 10 node(s)

Command line used: mdtest -v -i 2 -n 10000 -w 8192

-u -F -C -T -r -d /domfs/DOM-0-MDT/mdtestdir@

/domfs/DOM-1-MDT/mdtestdir@/domfs/DOM-2-MDT/mdtestdir@

/domfs/DOM-3-MDT/mdtestdir@/domfs/DOM-4-MDT/mdtestdir@

/domfs/DOM-5-MDT/mdtestdir@/domfs/DOM-6-MDT/mdtestdir@

/domfs/DOM-7-MDT/mdtestdir

V-1: Entering parse_dirpath...

V-1: Entering valid_tests...

barriers : True

collective_creates : False

create_only : True

dirpath(s):

/domfs/DOM-0-MDT/mdtestdir

/domfs/DOM-1-MDT/mdtestdir

/domfs/DOM-2-MDT/mdtestdir

/domfs/DOM-3-MDT/mdtestdir

/domfs/DOM-4-MDT/mdtestdir

/domfs/DOM-5-MDT/mdtestdir

/domfs/DOM-6-MDT/mdtestdir

/domfs/DOM-7-MDT/mdtestdir

dirs_only : False

read_bytes : 0

read_only : False

first : 1

files_only : True

iterations : 2

items_per_dir : 0

last : 0

leaf_only : False

items : 10000

nstride : 0

pre_delay : 0

remove_only : False

random_seed : 0

stride : 1

shared_file : False

time_unique_dir_overhead: False

stat_only : True

unique_dir_per_task : True

write_bytes : 8192

sync_file : False

depth : 0

V-1: Entering display_freespace...

V-1: Entering show_file_system_size...

Path: /domfs/DOM-0-MDT

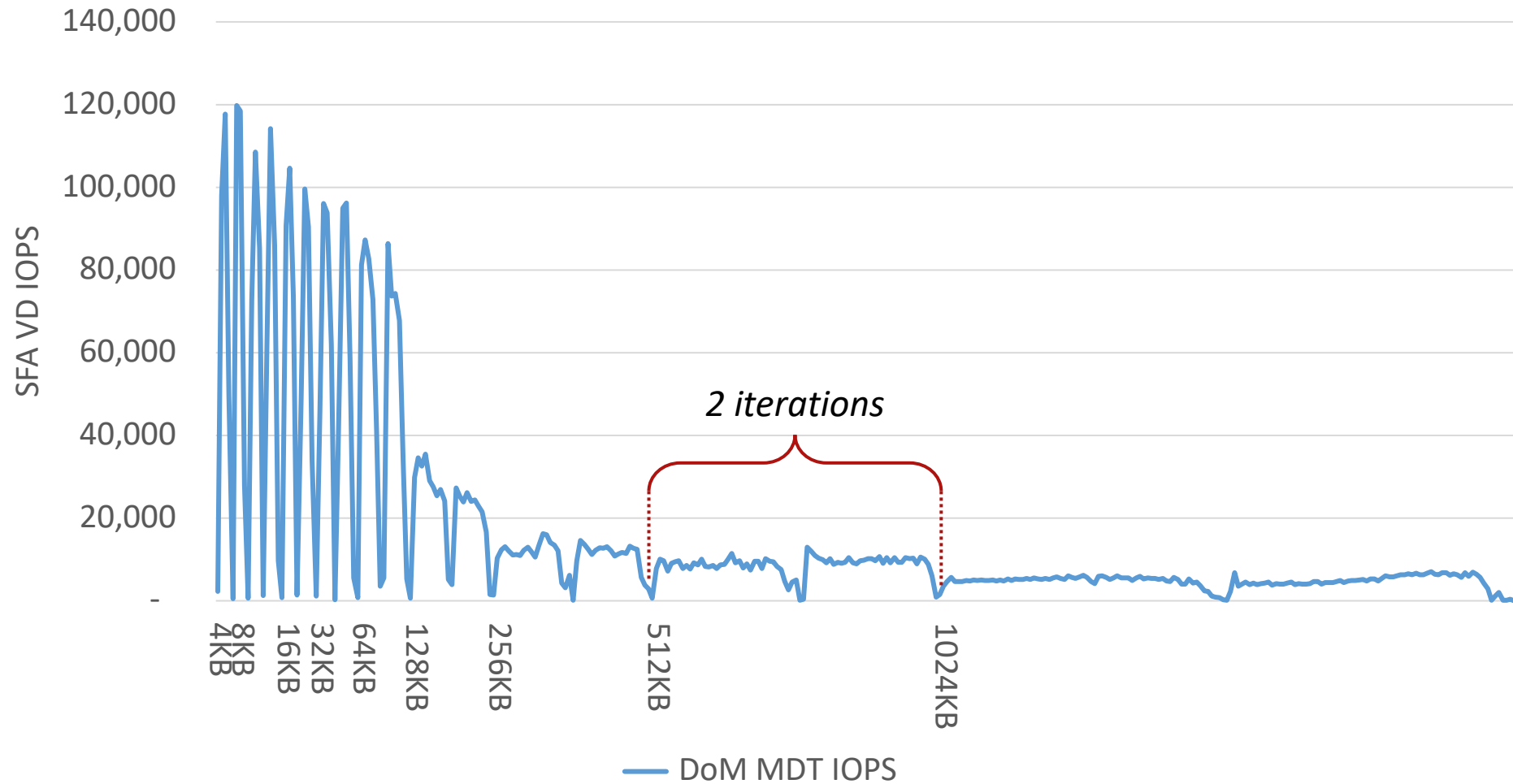
FS: 396.5 TiB Used FS: 0.0% Inodes: 400.2 Mi Used Inodes:
0.1%

160 tasks, 1600000 files

Operation	Duration	Rate
-----	-----	----
V-1: main: * iteration 1 *		
V-1: Entering create_remove_directory_tree, currDepth = 0...		
V-1: Entering create_remove_directory_tree, currDepth = 1...		
.		
.		
.		

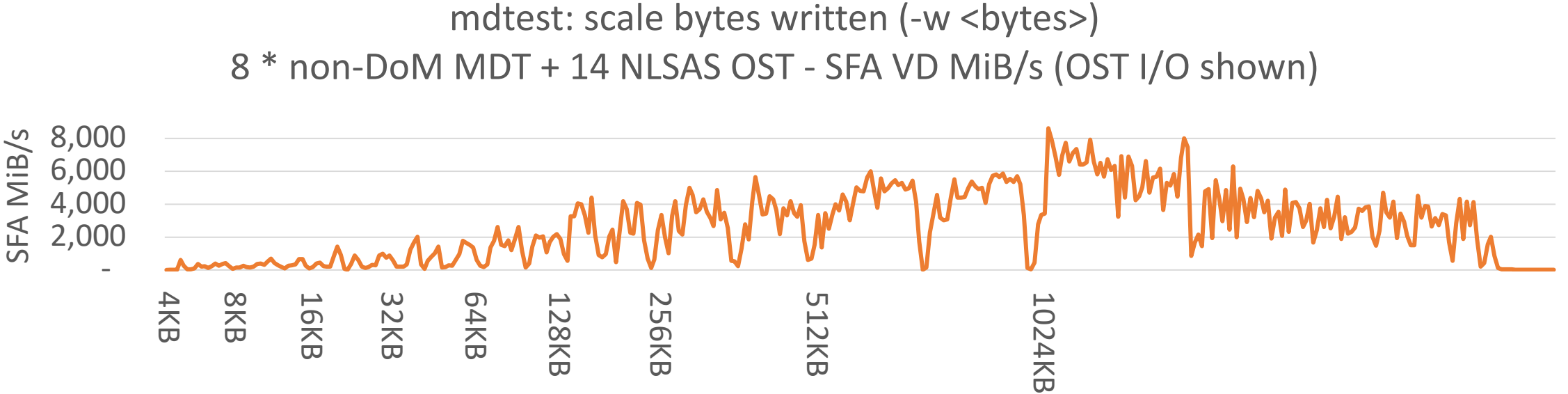
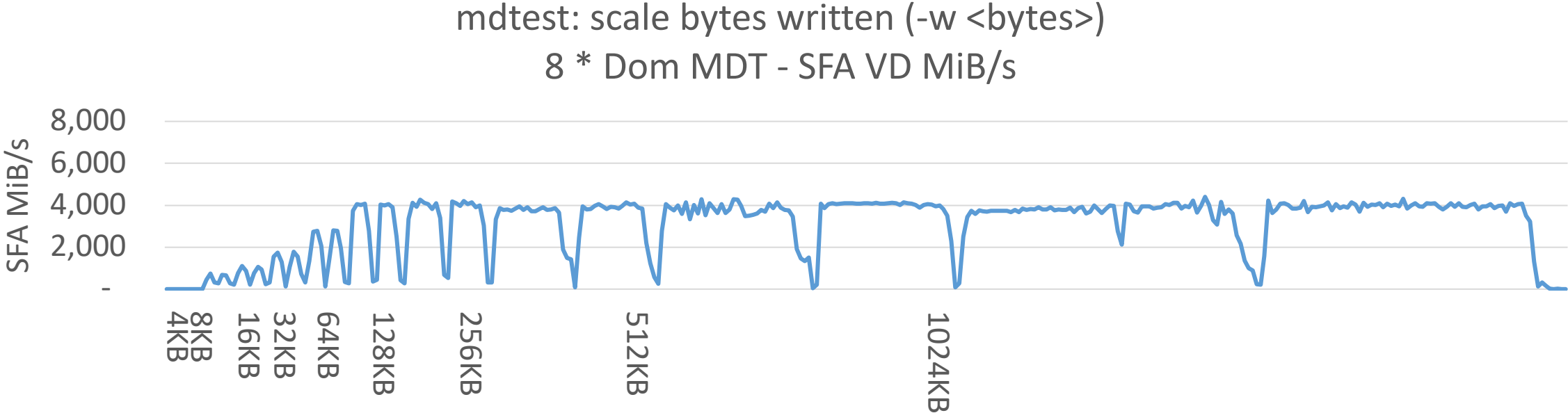
Test case 1b: mdtest - DoM-only MDT – IOPS

mdtest: scale bytes written (-w <bytes>)
8 * 1MB DoM MDT, no OST - SFA VD IOPS

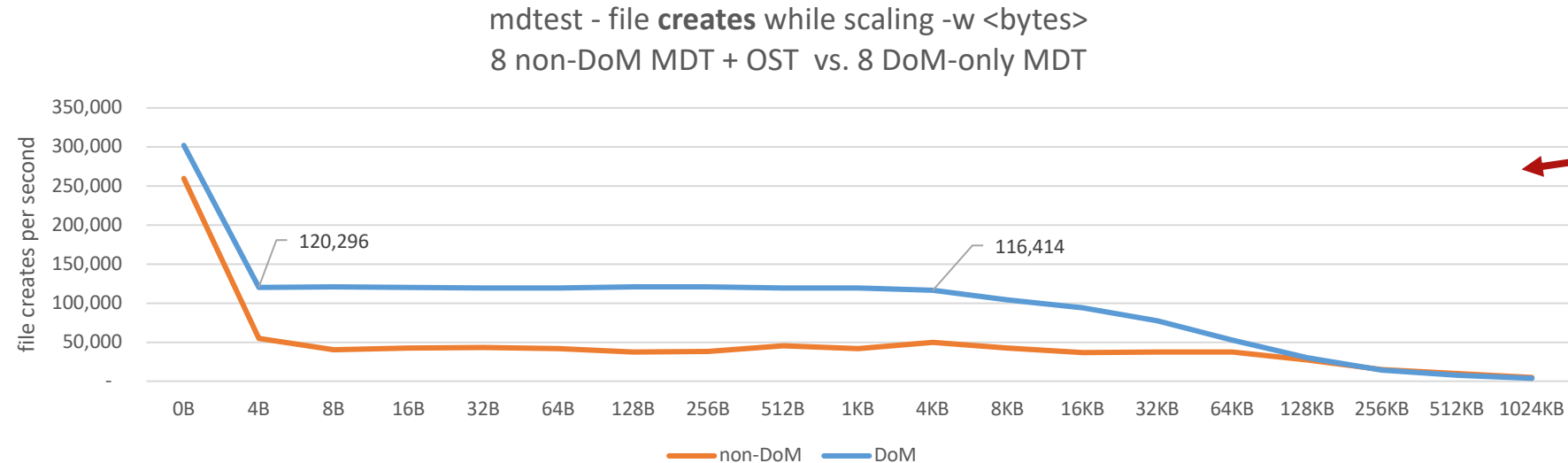


Test cases 1a and 1b: Side-by-Side

Test case 1a and 1b: mdtest - DoM MDT vs. non-DoM MDT+OST - **B/W**



Test cases 1a and 1b – file create and stat operations

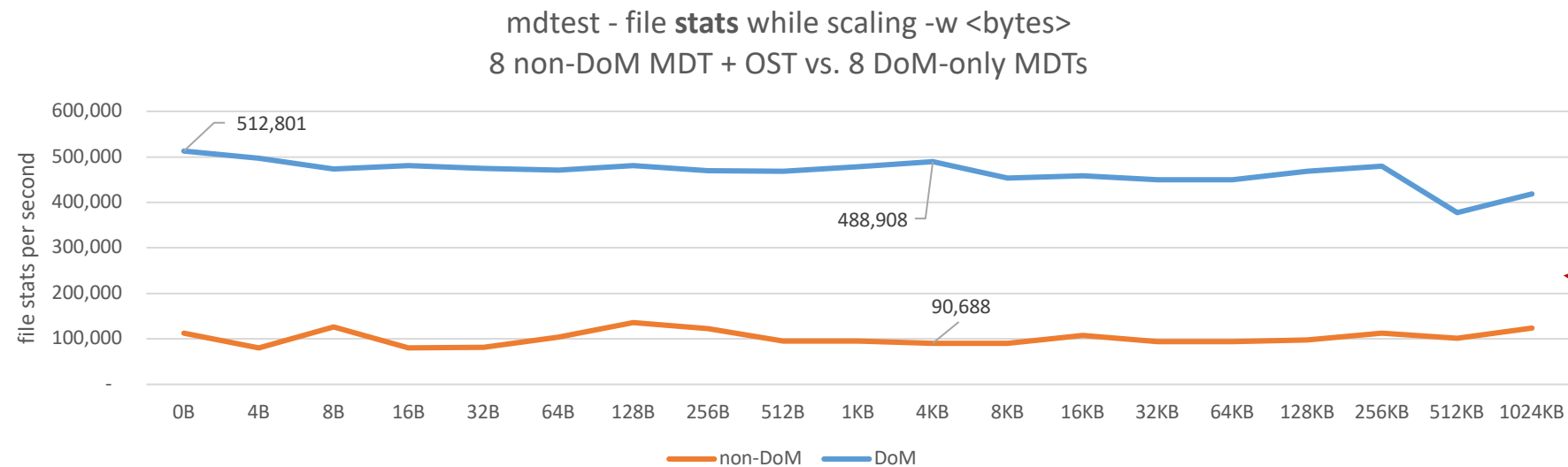


file create operations

Max Summary for DoM run

SUMMARY: (of 2 iterations)

Operation	Max
File creation	120784.307
File stat	504216.064
File read	0.000
File removal	271487.848
Tree creation	72.891
Tree removal	39.345



file stat operations



Test Case 2 – IOR

- 4KB, 8KB, 32KB random read/write against:
 - **2a:** *8 RAID6 8+2 SAS OST, no DoM
 - **2b:** 8 DoM MDT

Test case 2a - Setup

Test case 2a – IOR on OST - small block random, 1GB files

- ▶ **Instantiation and striping of 160 directories over 8 OSTs**
- ▶ **stripe count=1, component end=1G, stripe size=1MB, rotate through 2 sets of 10**

```
for i in `seq 0 9`; do lfs mkdir -i 0 ${i}; lfs setstripe -E 1G -c1 -S1G -o0 ${i}; done
for i in `seq 10 19`; do lfs mkdir -i 1 ${i}; lfs setstripe -E 1G -c1 -S1G -o1 ${i};done
for i in `seq 20 29`; do lfs mkdir -i 2 ${i}; lfs setstripe -E 1G -c1 -S1G -o2 ${i};done
for i in `seq 30 39`; do lfs mkdir -i 3 ${i}; lfs setstripe -E 1G -c1 -S1G -o3 ${i};done

for i in `seq 40 49`; do lfs mkdir -i 4 ${i}; lfs setstripe -E 1G -c1 -S1G -o7 ${i};done
for i in `seq 50 59`; do lfs mkdir -i 5 ${i}; lfs setstripe -E 1G -c1 -S1G -o8 ${i};done
for i in `seq 60 69`; do lfs mkdir -i 6 ${i}; lfs setstripe -E 1G -c1 -S1G -o9 ${i};done
for i in `seq 70 79`; do lfs mkdir -i 7 ${i}; lfs setstripe -E 1G -c1 -S1G -o10 ${i};done

for i in `seq 80 89`; do lfs mkdir -i 0 ${i}; lfs setstripe -E 1G -c1 -S1G -o0 ${i};done
for i in `seq 90 99`; do lfs mkdir -i 1 ${i}; lfs setstripe -E 1G -c1 -S1G -o1 ${i};done
for i in `seq 100 109`; do lfs mkdir -i 2 ${i}; lfs setstripe -E 1G -c1 -S1G -o2 ${i};done
for i in `seq 110 119`; do lfs mkdir -i 3 ${i}; lfs setstripe -E 1G -c1 -S1G -o3 ${i};done

for i in `seq 120 129`; do lfs mkdir -i 4 ${i}; lfs setstripe -E 1G -c1 -S1G -o7 ${i};done
for i in `seq 130 139`; do lfs mkdir -i 5 ${i}; lfs setstripe -E 1G -c1 -S1G -o8 ${i};done
for i in `seq 140 149`; do lfs mkdir -i 6 ${i}; lfs setstripe -E 1G -c1 -S1G -o9 ${i};done
for i in `seq 150 159`; do lfs mkdir -i 7 ${i}; lfs setstripe -E 1G -c1 -S1G -o10 ${i};done
```

- ▶ **IOR runs are 10 client, 16 PPN, unique directory**
- ▶ **4KB random read example below**
- ▶ **all caches dropped between runs**

```
# mpirun --machinefile dom-10 --map-by ppr:16:node IOR -z -e -r -k -E -F -u -t4k -b1g -o /domfs/ior
```


Test case 2a – IOR on OST - small block random + 1M sequential

Representative example of an OST Directory Component

Before File creation...

```
# lfs getstripe 0
0
lcm_layout_gen:      0
lcm_mirror_count:    1
lcm_entry_count:     1
  lcme_id:            N/A
  lcme_mirror_id:     N/A
  lcme_flags:         0
  lcme_extent.e_start: 0
  lcme_extent.e_end:  1073741824
    stripe_count:     1
    stripe_size:      1073741824
    pattern:          raid0
    stripe_offset:    0
```

directory




...and Directory + File Component After Write

```
# lfs getstripe 0
0
lcm_layout_gen:      0
lcm_mirror_count:    1
lcm_entry_count:     1
  lcme_id:            N/A
  lcme_mirror_id:     N/A
  lcme_flags:         0
  lcme_extent.e_start: 0
  lcme_extent.e_end:  1073741824
    stripe_count:     1
    stripe_size:      1073741824
    pattern:          raid0
    stripe_offset:    0

0/ior.00000000
lcm_layout_gen:      1
lcm_mirror_count:    1
lcm_entry_count:     1
  lcme_id:            1
  lcme_mirror_id:     0
  lcme_flags:         init
  lcme_extent.e_start: 0
  lcme_extent.e_end:  1073741824
    lmm_stripe_count:  1
    lmm_stripe_size:   1073741824
    lmm_pattern:       raid0
    lmm_layout_gen:    0
    lmm_stripe_offset: 0
    lmm_objects:
      - 0: { l_ost_idx: 0, l_fid: [0x100000000:0x15c:0x0] }
```

1 fie



*with
lmm_object*

Test case 2b – IOR on DoM - small block random, 1GB files

- ▶ IOR runs are 10 client, 16 PPN, unique directory
- ▶ Instantiation of 160 directories over 8 MDTs – 20 per
- ▶ stripe count implied, component end=1G, stripe size implied, rotate through 2 sets 8 MDTs

```
for i in `seq 0 9`; do lfs mkdir -i 0 ${i}; done
for i in `seq 10 19`; do lfs mkdir -i 1 ${i}; done
for i in `seq 20 29`; do lfs mkdir -i 2 ${i}; done
for i in `seq 30 39`; do lfs mkdir -i 3 ${i}; done
for i in `seq 40 49`; do lfs mkdir -i 4 ${i}; done
for i in `seq 50 59`; do lfs mkdir -i 5 ${i}; done
for i in `seq 60 69`; do lfs mkdir -i 6 ${i}; done
for i in `seq 70 79`; do lfs mkdir -i 7 ${i}; done

for i in `seq 80 89`; do lfs mkdir -i 0 ${i}; done
for i in `seq 90 99`; do lfs mkdir -i 1 ${i}; done
for i in `seq 100 109`; do lfs mkdir -i 2 ${i}; done
for i in `seq 110 119`; do lfs mkdir -i 3 ${i}; done
for i in `seq 120 129`; do lfs mkdir -i 4 ${i}; done
for i in `seq 130 139`; do lfs mkdir -i 5 ${i}; done
for i in `seq 140 149`; do lfs mkdir -i 6 ${i}; done
for i in `seq 150 159`; do lfs mkdir -i 7 ${i}; done

for i in `seq 0 159`; do lfs setstripe -E 1G -L mdt ${i}; done
```

```
# mpirun --machinefile dom-10 --map-by ppr:16:node IOR -z -e (-w | -r) -k -E -F -u -t4k -b1g -o /domfs/ior
```

Test case 2b – IOR on DoM - small block random + 1M sequential

Representative example of a DoM Directory Component

Before File Creation...

```
# lfs getstripe 0
0
lcm_layout_gen:      0
lcm_mirror_count:    1
lcm_entry_count:     1
  lcme_id:            N/A
  lcme_mirror_id:     N/A
  lcme_flags:         0
  lcme_extent.e_start: 0
  lcme_extent.e_end:  1073741824
    stripe_count: 0      stripe_size: 1073741824
    pattern:      mdt      stripe_offset: -1
```

directory



...and Directory + File Component After Write

```
# lfs getstripe 0
0
lcm_layout_gen:      0
lcm_mirror_count:    1
lcm_entry_count:     1
  lcme_id:            N/A
  lcme_mirror_id:     N/A
  lcme_flags:         0
  lcme_extent.e_start: 0
  lcme_extent.e_end:  1073741824
    stripe_count: 0      stripe_size: 1073741824
    pattern:      mdt      stripe_offset: -1

0/ior.00000000
lcm_layout_gen:      1
lcm_mirror_count:    1
lcm_entry_count:     1
  lcme_id:            1
  lcme_mirror_id:     0
  lcme_flags:         init
  lcme_extent.e_start: 0
  lcme_extent.e_end:  1073741824
    lmm_stripe_count: 0
    lmm_stripe_size:  1073741824
    lmm_pattern:      mdt
    lmm_layout_gen:   0
    lmm_stripe_offset: 0
```

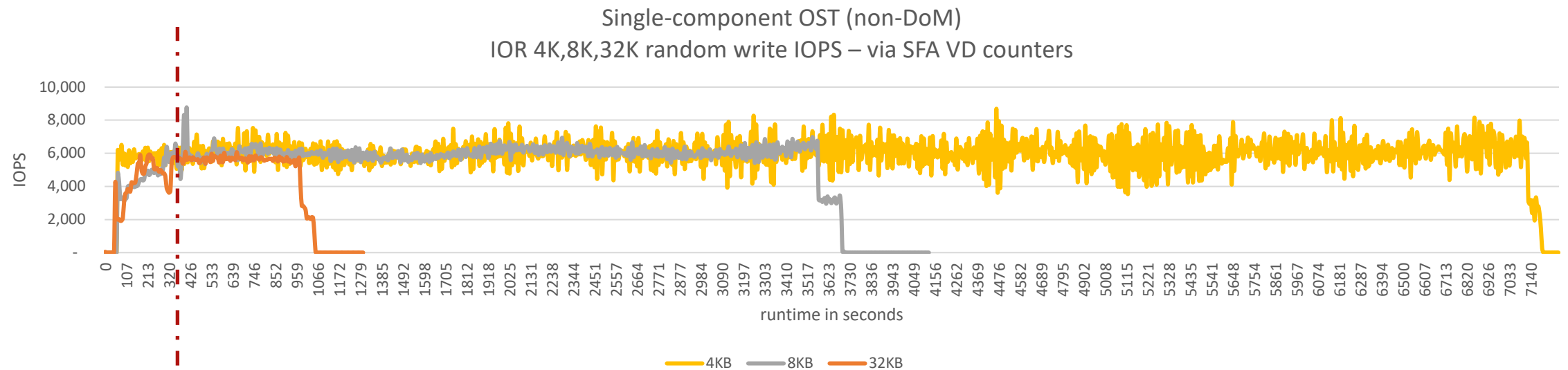
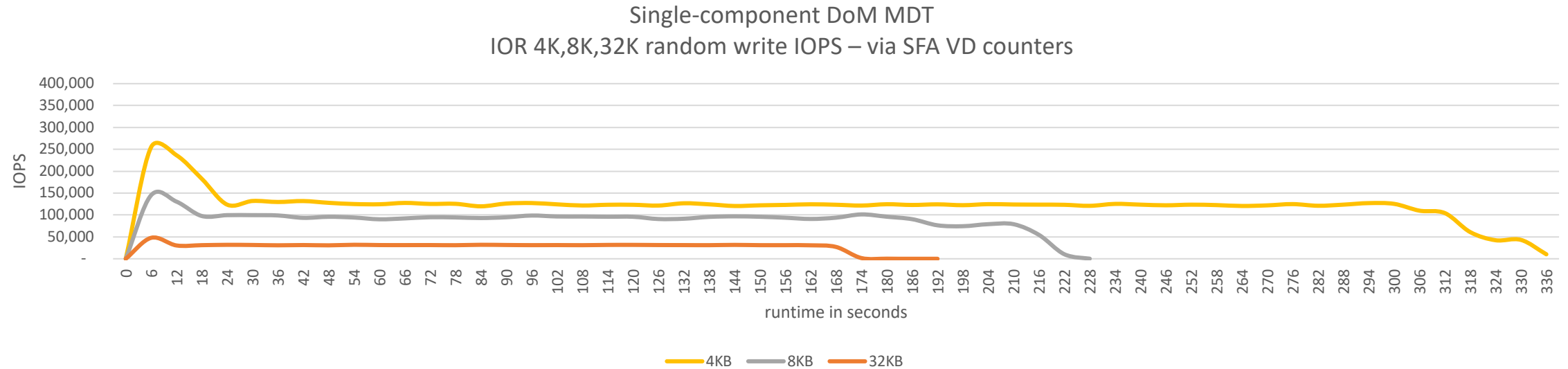
1 file



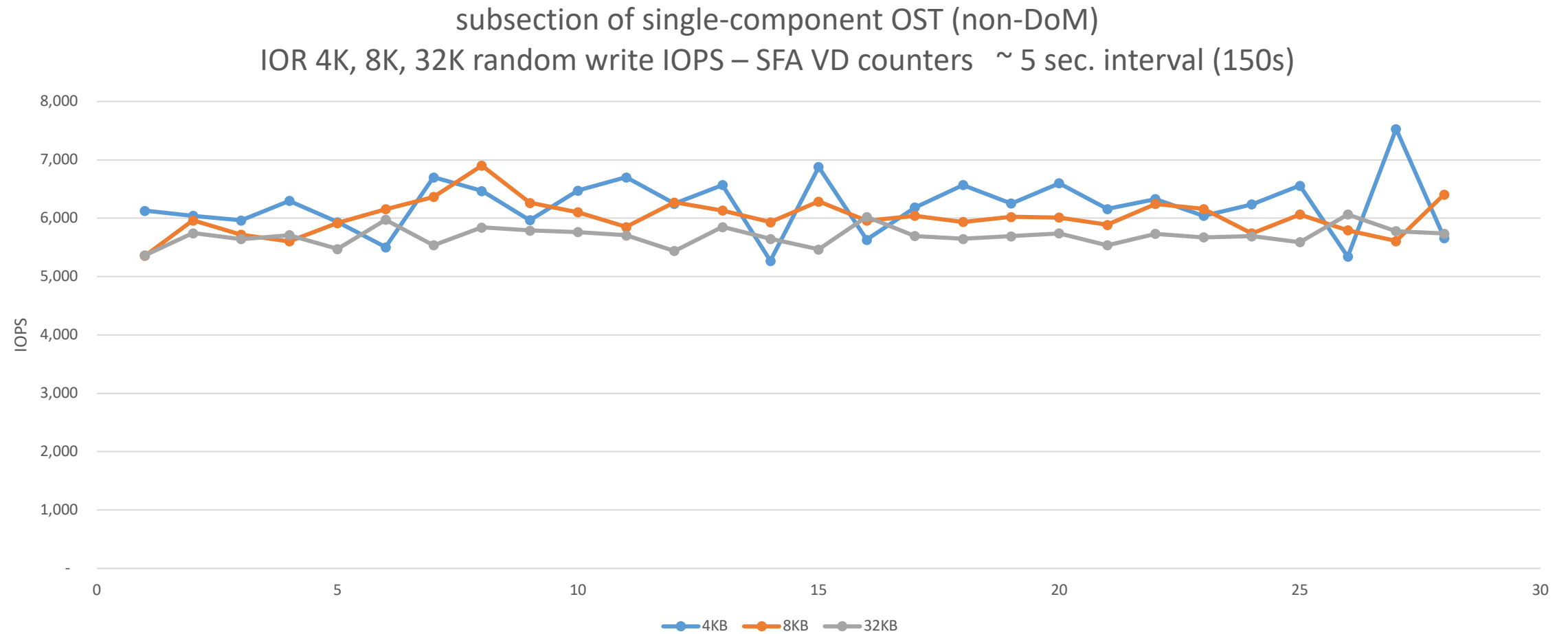
Note that there is no lmm object present for a MDT DoM file stripe entry

Test cases 2a and 2b: Side-by-Side

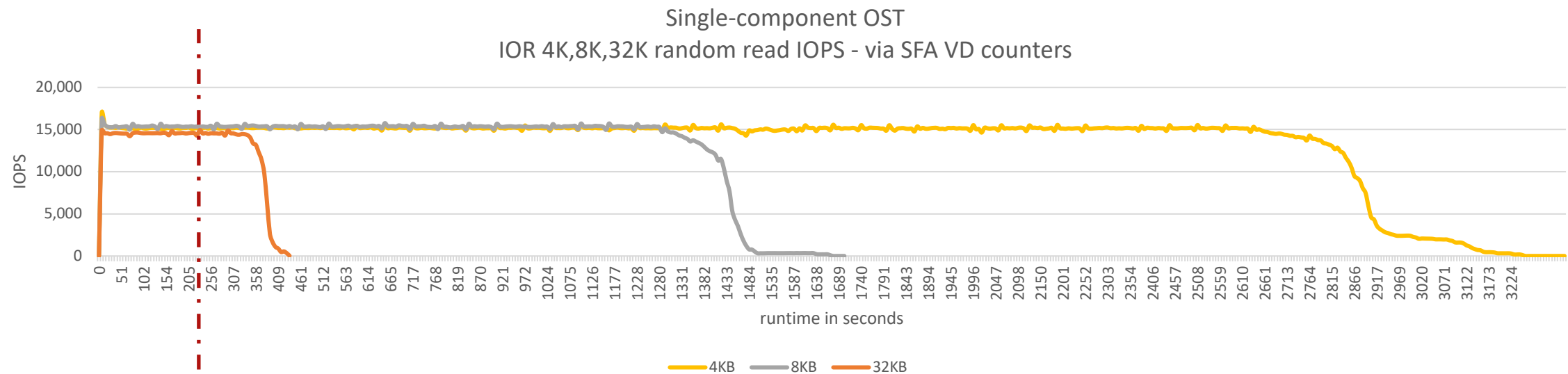
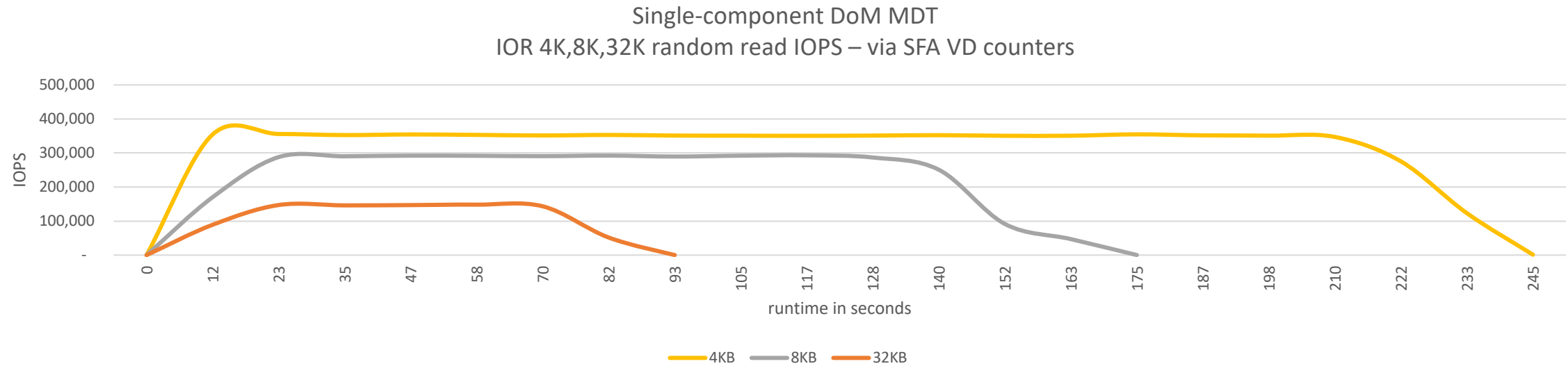
Test case 2 – IOR Write IOPS – DoM MDT vs. OST



Test case 2 – OST IOR Write IOPS – subsection zoom-in peek



Test case 2 – IOR Read IOPS – DoM MDT vs. OST





Test Case 3

- Workload Isolation Benefits. Compare:
 - **a:** Sequential read from HDD OSTs – standalone baseline workload (Background workload)
 - **b:** Start background sequential read from OSTs ... introduce 2nd workload (IOR write) - same OSTs
 - **c:** Start Background sequential read from OSTs ... introduce 2nd workload (IOR write) - DoM

Test case 3: Workload Isolation – details

3 Workloads - Description

- a. Background workload (IOR sequential read) target directories in root of lustre filesystem with default striping. Pre-create files.
- b. Foreground disruptive OST workload (IOR sequential write) target directories
240 directories with ***lfs setstripe -E 1G -o \${i}*** over **14 OSTs** (4 clients 8 PPN)
- c. Foreground DoM workload (IOR sequential write) target directories
Similar to test case 2, but 240 directories, with ***lfs setstripe -E 1G -L mdt \${i}*** over **8 MDTs** (6 clients 40PPN)

IOR Run Description:

Background/Control OST-only 1MB sequential read

```
$ mpirun --machinefile dom-4 --map-by ppr:8:node IOR -e -r -k -E -F -t 1m -b64g -vv -o /domfs/ior
```

Foreground/OST-only 1MB sequential write

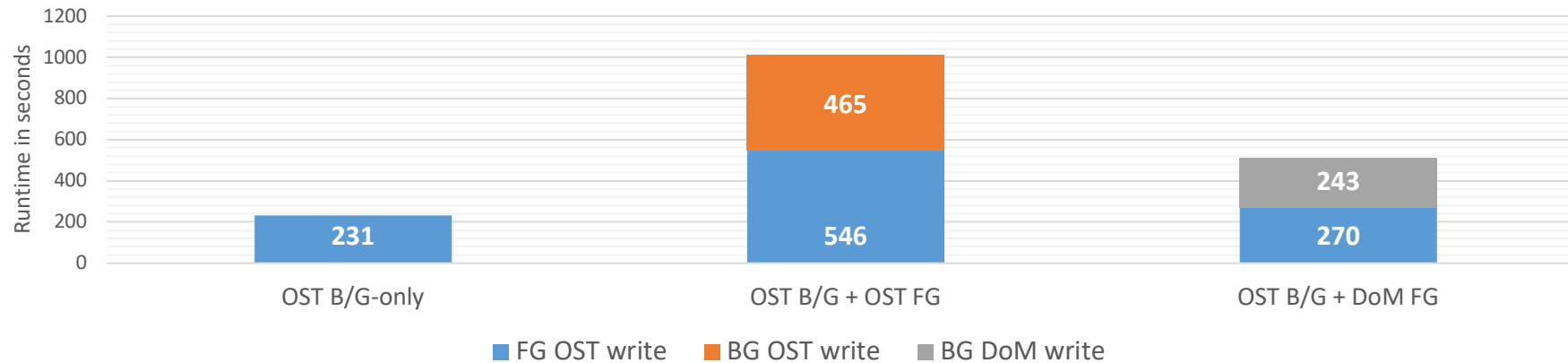
```
$ clients=6;rs=40;ppn=40;np=$((ppn*clients));mpirun --oversubscribe --machinefile dom-havoc \
--map-by node --np $np IOR -i10 -d10 -e -w -E -F -u -t1m -b1g -vv -o /domfs/havoc-O/ior
```

Foreground/DoM-only 1MB sequential write

```
$ clients=6;rs=40;ppn=40;np=$((ppn*clients));mpirun --oversubscribe --machinefile dom-havoc \
--map-by node --np $np IOR -i4 -d10 -e -w -E -F -u -t1m -b1g -vv -o /domfs/havoc-D/ior
```

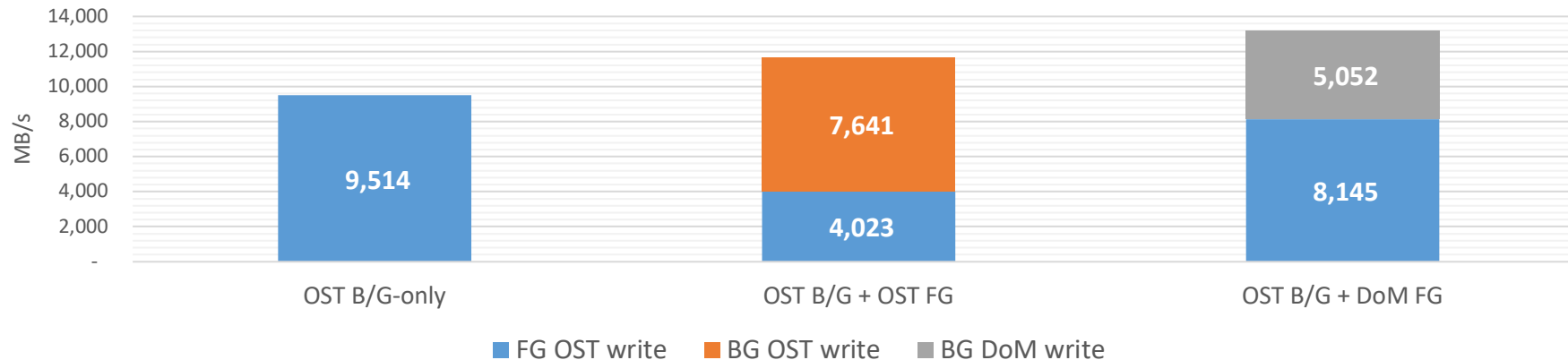
Test case 3 – Workload Isolation Benefits

Runtime Gains from DoM workload Separation
Foreground vs. Background+Foreground vs. Background+DoM



Workload Residence	Runtime
OST Only	211s
OST B/G + OST F/G	1011s
OST B/G + DoM F/G	513s

IOR BW Gains from DoM workload Separation
Foreground vs. Background+Foreground vs. Background+DoM



Workload Residence	MB/s
OST Only	9,514
OST B/G + OST F/G	11,664
OST B/G + DoM F/G	13,197



Looking Ahead

- Migration of files from MDT to OST by request or automatically
 - Performance enhancements
 - Read-ahead for small files, return data in open()
 - Client - first write size detect will change layout
-
- Performance testing with DNE2 striped directories
 - Large-scale performance investigation with a diverse workload
 - Benefits of diverting DoM RPCs to MDS on an all-flash lustre filesystems
 - Small file size ranges – further studies with real codes

References – Documentation and Planned Enhancements

- [Data on MDT](#)
- [Data on MDT High Level Design](#)
- [DoM User Guide](#)
- [DoM Performance Optimization](#)
- [Data on MDT Phase II](#)
- [Lock-ahead - Request extent locks from userspace](#)
- [Better DOM+DNE integration](#)

Acknowledgements

Suichi Ihara

Rahul Deshmukh

Carlos Thomaz

Thank You!

Keep in touch with us.



sales@ddn.com



9351 Deering Avenue
Chatsworth, CA 91311



[@ddn_limitless](https://twitter.com/ddn_limitless)



1.800.837.2298
1.818.700.4000



[company/datadirect-networks](https://www.linkedin.com/company/datadirect-networks)

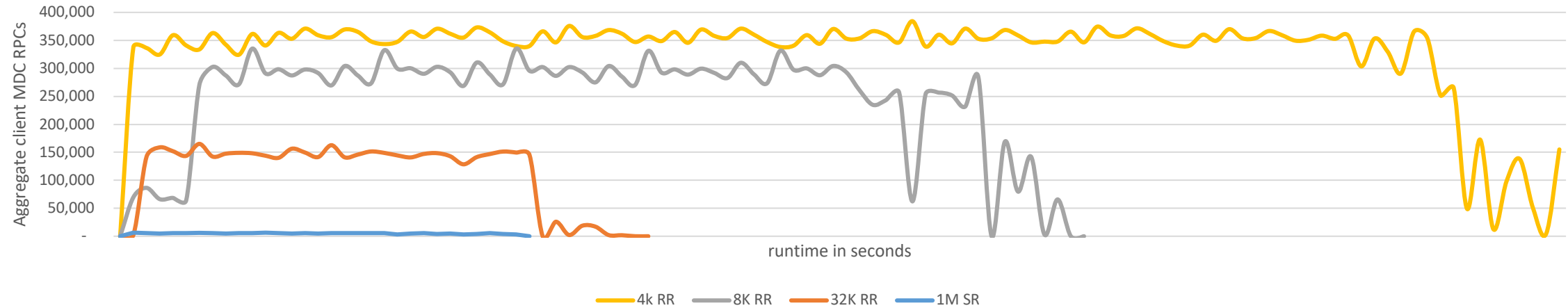


Appendix

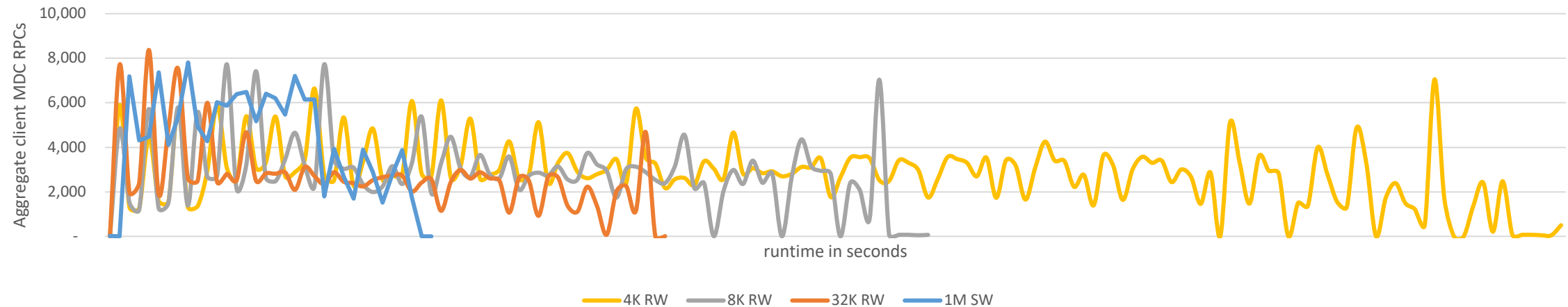
- RPC Graph
- IOR Write throughput
- LBUG during lfs migrate

Test case 2a – IOR on DoM – Read / Write MDC RPCs at client

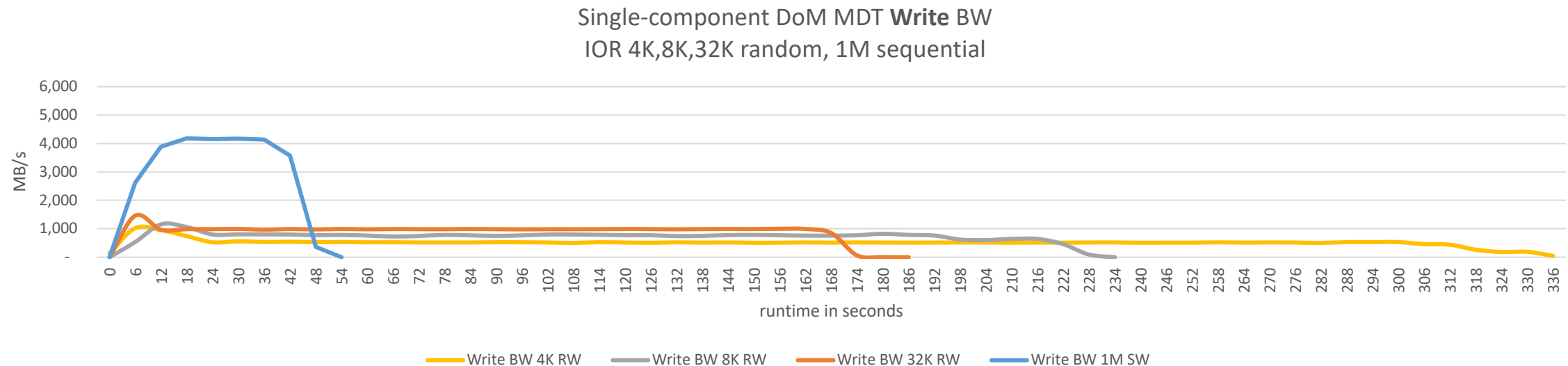
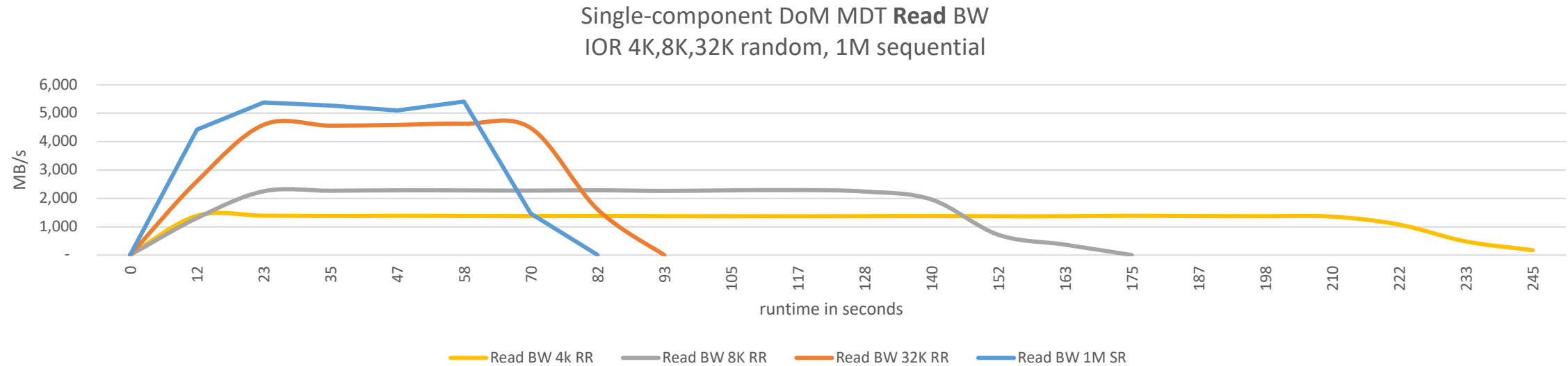
Single-component DoM MDT - Aggr. Read MDC RPCs
during IOR 4K,8K,32K random, 1M sequential runs



Single-component DoM MDT - Aggr. Write MDC RPCs
during IOR 4K,8K,32K random, 1M sequential runs



Test case 2 – IOR Write Throughput – DoM MDT vs. OST



LBUG seen while using lfs migrate on a DoM file

```
lfs getstripe test-1M.3
test-1M.3
  lcm_layout_gen:      1
  lcm_mirror_count:    1
  lcm_entry_count:     1
    lcme_id:           1
    lcme_mirror_id:    0
    lcme_flags:        init
    lcme_extent.e_start: 0
    lcme_extent.e_end: 1048576
    lmm_stripe_count:  0
    lmm_stripe_size:   1048576
    lmm_pattern:       mdt
    lmm_layout_gen:    0
    lmm_stripe_offset: 0

# lfs migrate -L mdt -E 4M test-1M.3

Message from syslogd@ at Apr 15 21:38:31 ...
kernel:LustreError: 4366:0:(mdc_dev.c:1346:mdc_req_attr_set()) LBUG

Message from syslogd@s at Apr 15 21:38:31 ...
kernel:Kernel panic - not syncing: LBUG
packet_write_wait: Connection to 10.x.x.x port 22: Broken pipe
```

```
# ls -l
total 1049460
-rw-r--r-- 1 root root 1073741824 Apr 22 09:31 10.testfile

# lfs getstripe 10.testfile
10.testfile
  lcm_layout_gen:      1
  lcm_mirror_count:    1
  lcm_entry_count:     1
    lcme_id:           1
    lcme_mirror_id:    0
    lcme_flags:        init
    lcme_extent.e_start: 0
    lcme_extent.e_end: 1073741824
    lmm_stripe_count:  0
    lmm_stripe_size:   1073741824
    lmm_pattern:       mdt
    lmm_layout_gen:    0
    lmm_stripe_offset: 0

# lfs migrate -E 64M -c 1 -E 256M -c 4 -E -1 -c -1 10.testfile

Message from syslogd@ at Apr 16 09:32:02 ...
kernel:LustreError: 2634:0:(mdc_dev.c:1346:mdc_req_attr_set()) LBUG
```

LU-10910 - LBUG with "lfs migrate -c 1 <domfile>"