DAOS Benchmarks and Findings

Rebecca George[1,2], Xinxin Mei[2], Jie Ren[1]
[1]William & Mary, [2]Jefferson Lab

Benchmarks

fio
- pvsync I/O engine, DFuse + POSIX

IOR
- DFS API

IO500
- DFS API vs DFuse + POSIX

mdtest
- DFS API

Interesting Errors

- metadata target filling (mdtest)
- clock set affinity fails (fio)
- NA_HOSTUNREACH errors (ior)
- Something alerting for admins in October

Benchmarks

Pool Info

```
Pool space info:
- Target(VOS) count:4064 → 127 storage nodes
- Storage tier 0 (SCM):
  Total size: 3.0 TB
  Free: 2.7 TB, min:655 MB, max:655 MB, mean:655 MB
- Storage tier 1 (NVMe):
  Total size: 97 TB
  Free: 97 TB, min:24 GB, max:24 GB, mean:24 GB
```

fio Benchmarks

Using pvsync I/O engine, DFuse POSIX container

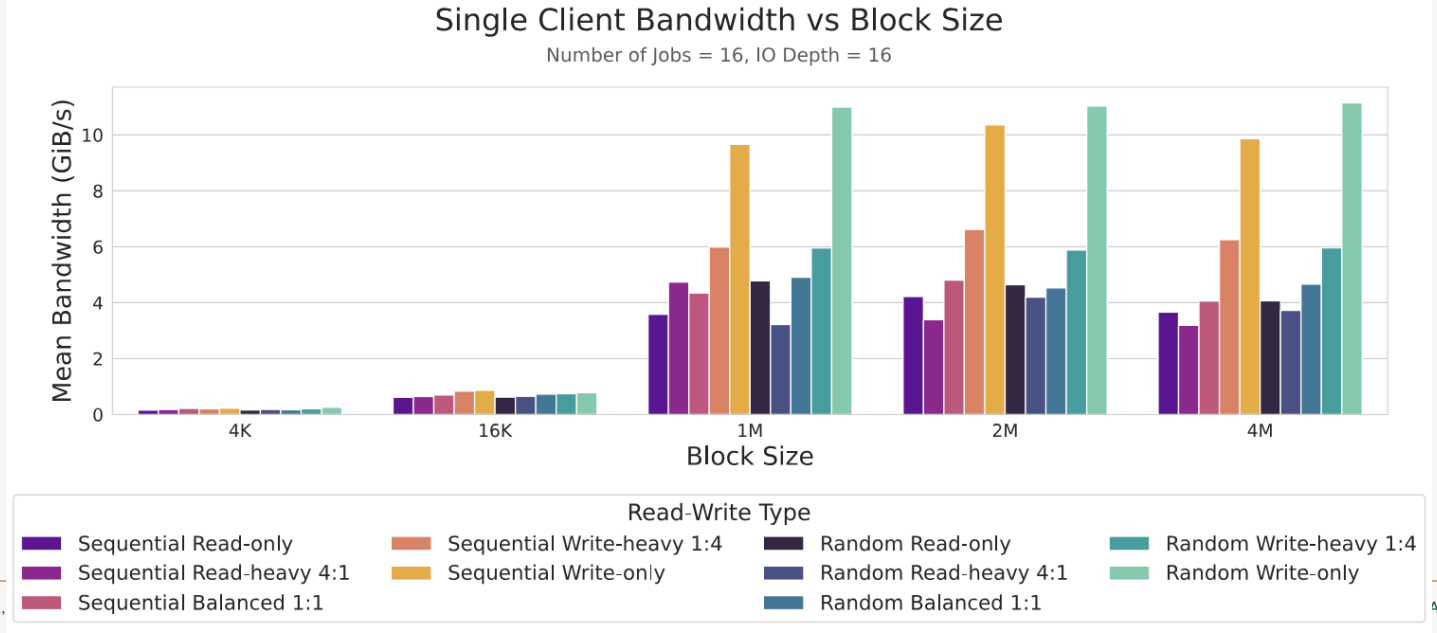
Parameters:

Access = read, rw (--rwmixread=80), rw, rw (--rwmixread=20), write, randread, randrw (--rwmixread=80), randrw, randrw (--rwmixread=20), randwrite

Block size = 4k, 16k, 1M, 2M, 4M

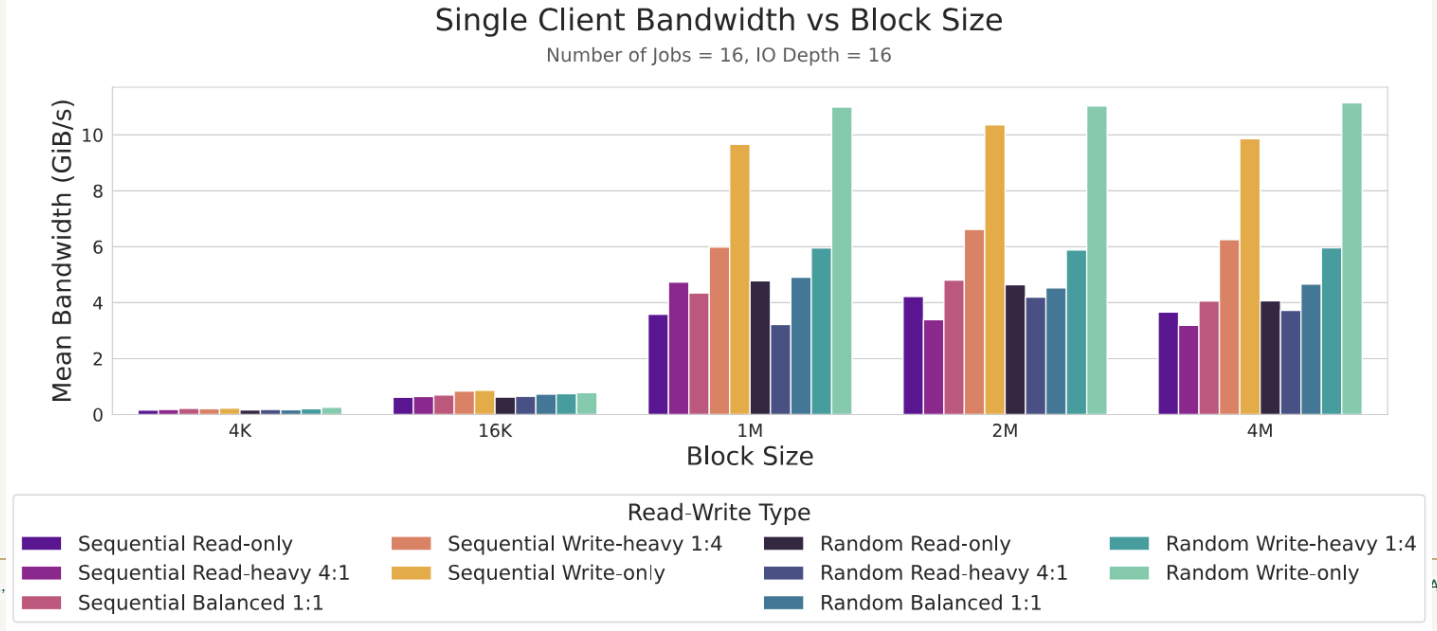
IO depth = 1, 4, 8, 16, 32, 64, 128

Jobs = 8, 16, 32, 64, 128

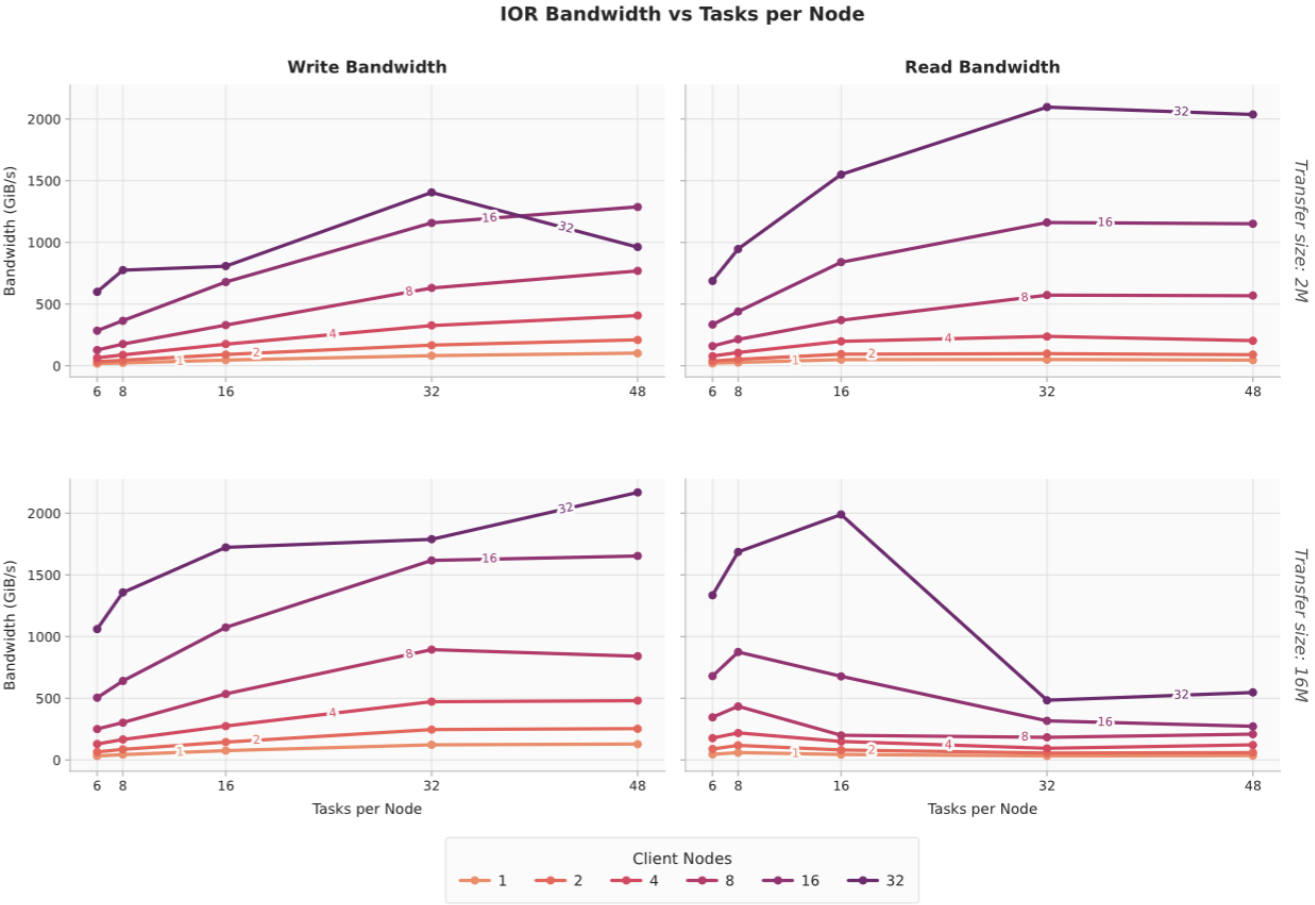


fio Benchmarks

- similar random/sequential access performance
- larger block sizes → higher bandwidths
- write bandwidth was much higher than read bandwidth



ior Benchmarks

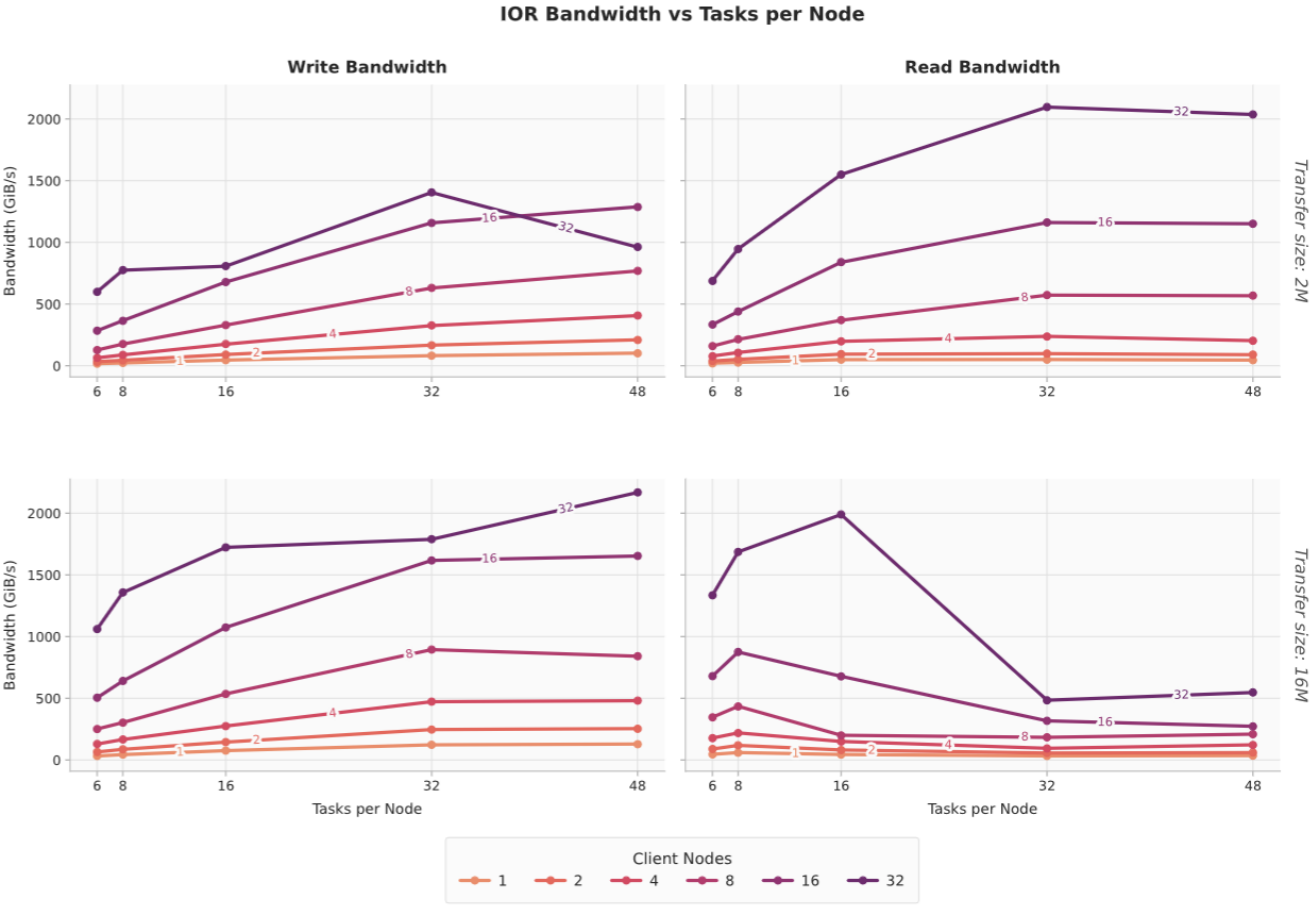


For transfer sizes of 2 MiB and 16 MiB , the IOR bandwidth vs number of tasks, using the DFS API

DFS API

Parameters:
Nodes: 1, 2, 4, 8, 16, 32
Block size: 128M
Transfer size: 16K 1M 2M 16M
Segments: 32
Ranks/Node: 6 8 16 32 48
Access: read, write

ior Benchmarks

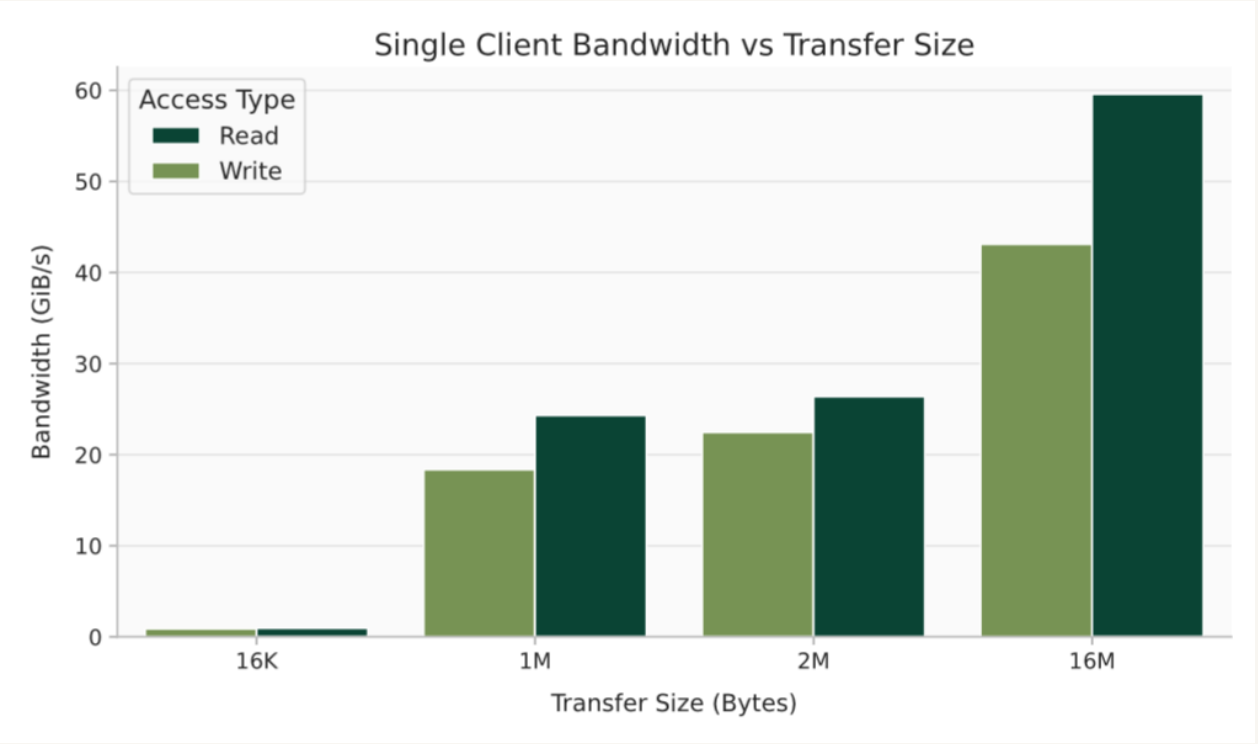


For transfer sizes of 2 MiB and 16 MiB , the IOR bandwidth vs number of tasks, using the DFS API

Read and write bandwidth increased as tasks increased, saturating around 32 tasks

At high task counts, sufficiently large transfer sizes reduced read bandwidth

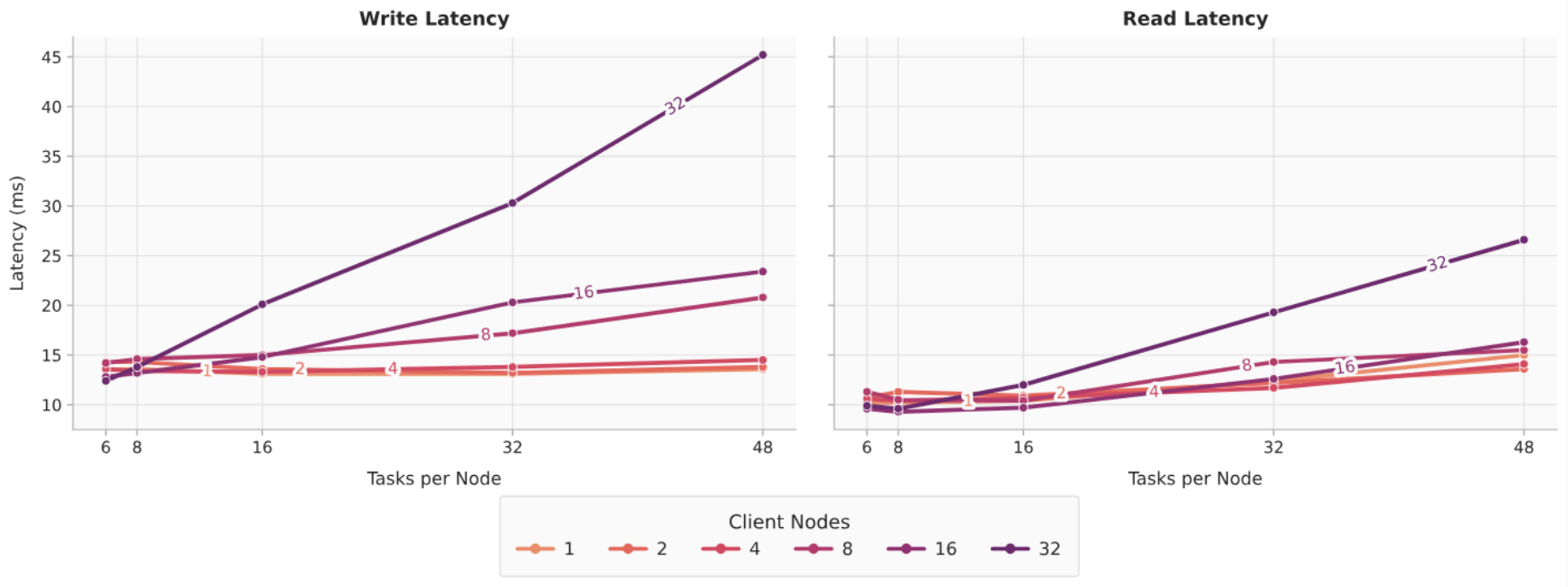
ior Benchmarks



For a single client, reading is slightly more performant than writing across transfer sizes

Sufficiently small transfer sizes (16 KiB) provide low throughput, both for a single client and all numbers of nodes and tasks per node tested

ior Benchmarks



Mean latency increases as the number of tasks increase at a more higher rate for writing than reading, for sufficiently large numbers of client nodes

IO500 Benchmarks

Test	DFS API	POSIX API
ior-easy-write	92.2 GiB/s	81.3 GiB/s
ior-easy-read	212.6 GiB/s	10.8 GiB/s
ior-hard-write	3.8 GiB/s	0.7 GiB/s
ior-hard-read	15.3 GiB/s	3.2 GiB/s
Bandwidth Score	32.6 GiB/s	6.5 GiB/s

Table 1. 8 node IO500 bandwidth

Test	DFS API	POSIX API
find	775.6 kIOPS	1.6 kIOPS
mdtest-easy-write	219.4 kIOPS	4.0 kIOPS
mdtest-hard-write	60.6 kIOPS	1.1 kIOPS
mdtest-hard-read	807.3 kIOPS	3.9 kIOPS
mdtest-easy-stat	1112.2 kIOPS	4.0 kIOPS
mdtest-hard-stat	1098.3 kIOPS	3.9 kIOPS
mdtest-easy-delete	301.0 kIOPS	3.4 kIOPS
mdtest-hard-delete	302.6 kIOPS	0.4 kIOPS
IOPS Score	417.7 kIOPS	2.2 kIOPS

Table 2. 8 node IO500 IOPS

DFS achieved ~5x higher bandwidths and ~190x higher IOPS than POSIX on a DFuse mount,

Interesting Errors

mdtest SCM target filling

When running mdtest, the job seems to fill up the storage space faster than expected, and a small number of targets fail to empty after the container is deleted.

This mdtest from io500 worked:

```
[mdtest-hard-write]
t_start         = 2026-02-15 06:26:58
exe             = ./mdtest --dataPacketType=timestamp -n 1000000 -t -w 3901 -e 3901 -P -G=-1622856514 -N 1 -F -
d /tmp/e2sar/rgeorge-io500/io500_data/mdtest-hard -x /home/rgeorge/io500_8326336_1-n/mdtest-hard.stonewall -C -
Y -W 300 --saveRankPerformanceDetails=/home/rgeorge/io500_8326336_1-n/mdtest-hard-write.csv -a DFS --
dfs.pool=e2sar --dfs.cont=rgeorge-io500 --dfs.dir_oclass=RP_2GX --dfs.oclass=RP_2G1
```

However, this mdtest command fails:

```
/home/rgeorge/ior/install/bin/mdtest '-a' 'DFS' '-d' '/mdtest' '-i' '2' '-n' '100000' '-u' '-F' '-w' '6144' '-
e' '6144' '--dfs.pool' 'e2sar2' '--dfs.cont' 'rgeorge-mdtest'
```

I thought 1 client node, with 48 cores, creating 100000 files per process, where files are 6 KiB big would mean :4,800,000 files x 6 KiB = 28,800,000 KiB = ~27.5 GiB of files (plenty of space)

mdtest SCM target filling

When running mdtest, the job seems to fill up the storage space faster than expected, and a small number of targets fail to empty after the container is deleted.

This mdtest from io500 worked:

```
[mdtest-hard-write]
t_start         = 2026-02-15 06:26:58
exe             = ./mdtest --dataPacketType=timestamp -n 1000000 -t -w 3901 -e 3901 -P -G=-1622856514 -N 1 -F -
d /tmp/e2sar/rgeorge-io500/io500_data/mdtest-hard -x /home/rgeorge/io500_8326336_1-n/mdtest-hard.stonewall -C -
Y -W 300 --saveRankPerformanceDetails=/home/rgeorge/io500_8326336_1-n/mdtest-hard-write.csv -a DFS --
dfs.pool=e2sar --dfs.cont=rgeorge-io500 --dfs.dir_oclass=RP_2GX --dfs.oclass=RP_2G1
```

previously 1024, raised to 6144 to hopefully become larger than data_thresh and avoid filling space

However, this mdtest command fails:

```
/home/rgeorge/ior/install/bin/mdtest '-a' 'DFS' '-d' '/mdtest' '-i' '2' '-n' '100000' '-u' '-F' '-w' '6144' '-
e' '6144' '--dfs.pool' 'e2sar2' '--dfs.cont' 'rgeorge-mdtest'
```

I thought 1 client node, with 48 cores, creating 100000 files per process, where files are 6 KiB big would mean :4,800,000 files x 6 KiB = 28,800,000 KiB = ~27.5 GiB of files (plenty of space)

mdtest SCM target filling

When running mdtest, the job seems to fill up the storage space faster than expected, and a small number of targets fail to empty after the container is deleted.

After the failure:

```
$ daos pool query e2sar
Pool 451a32a5-f5f3-4759-9935-5f26c0a2cee0, ntarget=4064,
disabled=160, leader=228, version=365, state=Degraded
Pool health info:
- Rebuild done, 0 objs, 0 recs
Pool space info:
- Target(VOS) count:3904
- Storage tier 0 (SCM):
 Total size: 2.9 TB
 Free: 2.6 TB, min:0 B, max:655 MB, mean:654 MB
- Storage tier 1 (NVMe):
 Total size: 93 TB
 Free: 93 TB, min:24 GB, max:24 GB, mean:24 GB
```

from this we see only one or a few targets fill up and stay full

Clock set affinity failures

fio with all three of these cases still gets the same error:

1. **fio** --cpus_allowed=$NUMA_CPU_CORES --cpus_allowed_policy=shared …
2. taskset -c $NUMA_CPU_CORES **fio** …
3. **fio** …

```
Successfully created container 897d3ac1-a794-4c0e-9eda-7d0d03f17465
  Container UUID : 897d3ac1-a794-4c0e-9eda-7d0d03f17465
  Container Label: rgeorge-fio_test-p1r1_SeqR-2M-16-128
  Container Type : POSIX
...
(launch fio command)
...
clock setaffinity failed: Invalid argument
clock setaffinity failed: Invalid argument
clock setaffinity failed: Invalid argument
clock setaffinity failed: Invalid argument
```

ior NA_HOSTUNREACH errors

With seemingly no pattern, the ior read + write tests in February intermittently returned NA_HOSTUNREACH errors, but the results align with trends from error-less data

```
Writing output to ior-results-rw_n-4_seg-32_8331886/rw_n-4_ppn-32_tx-1M.csv
Writing output to ior-results-rw_n-4_seg-32_8331886/rw_n-4_ppn-32_tx-1M.csv
02/15-19:48:47.42 x4019c6s2b0n0 DAOS[84790/84790/0] external ERR  # [841344.750038] mercury-
>rpc [error] /home/daos/pre/build/external/release/mercury/src/mercury_core.c:4479
hg_core_send_input_cb() NA callback returned error (NA_HOSTUNREACH)
02/15-19:48:47.52 x4019c6s3b0n0 DAOS[85740/85740/0] external ERR  # [841344.219472] mercury-
>rpc [error] /home/daos/pre/build/external/release/mercury/src/mercury_core.c:4479
hg_core_send_input_cb() NA callback returned error (NA_HOSTUNREACH)
02/15-19:48:47.57 x4019c6s3b0n0 DAOS[85722/85722/0] external ERR  # [841344.273194] mercury-
>rpc [error] /home/daos/pre/build/external/release/mercury/src/mercury_core.c:4479
hg_core_send_input_cb() NA callback returned error (NA_HOSTUNREACH)
...
Writing output to ior-results-rw_n-4_seg-32_8331886/rw_n-4_ppn-48_tx-16K.csv
Writing output to ior-results-rw_n-4_seg-32_8331886/rw_n-4_ppn-48_tx-16K.csv
02/15-19:50:28.35 x4019c6s0b0n0 DAOS[90374/90374/0] external ERR  # [841436.422321] mercury-
>rpc [error] /home/daos/pre/build/external/release/mercury/src/mercury_core.c:4479
hg_core_send_input_cb() NA callback returned error (NA_HOSTUNREACH)
...
```

October alerts to admins

In October 2025, something from our pool e2sar sent repeating errors to admins

After daos_user servers restarted during preventive maintenance,
There was increased message flow reporting errors,
According to the vendor, it was caused by something about the pool

Destroying and recreating the pool fixed the issue