OPENFABRICS ALLIANCE

2021 OFA Virtual Workshop

HOW TO EFFICIENTLY PROVIDE SOFTWARE-DEFINED STORAGE USING SMARTNICS

Jonas Pfefferle, Nikolas Ioannou, Jose Castanos, Bernard Metzler

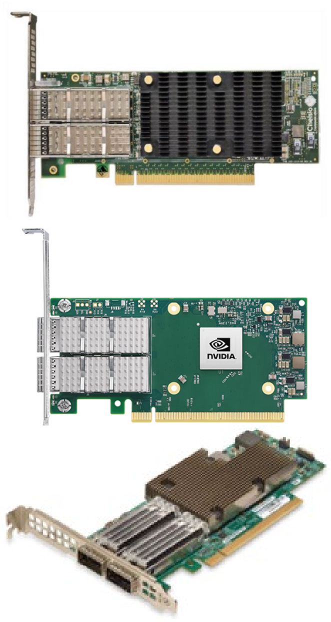
IBM Research

MOTIVATION

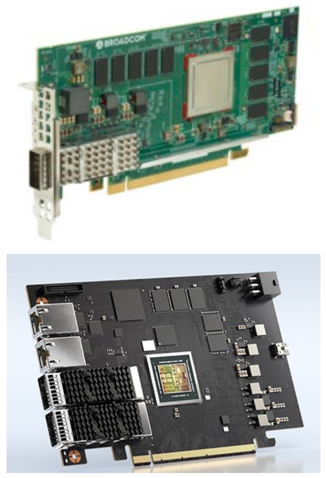
- **SmartNICs are the state-of-the-art solution to provide network and storage virtualization in cloud environments**
- **Leading cloud providers use custom SmartNIC designs like AWS Nitro or Azure SmartNIC**
- **SmartNICs provide isolation, security and increased performance**
  => more energy and cost efficient
- **Recently a new set of commodity SmartNIC products have become available**
  For example: NVIDIA BlueField, Broadcom Stingray or Pensando DSC
- **Include broad set of storage and network virtualization options**
  - Overlay networks e.g. VXLAN
  - Embedded switch
  - NVMe emulation
  - Virtio-queue support (block and network)
  - Encryption
  - Packet filters / deep packet inspection
- **Can we use a commodity SmartNIC to provide transparent storage virtualization with Ceph as the storage backend?**

SMARTNIC LANDSCAPE

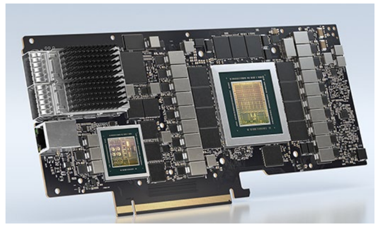
"Smart"



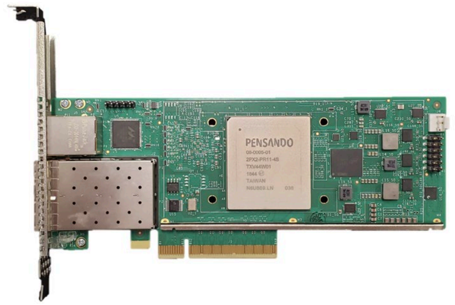
RDMA, NFV, ToE, NVMe-oF, iSCSI, …



NIC + Arm



NIC + GPU



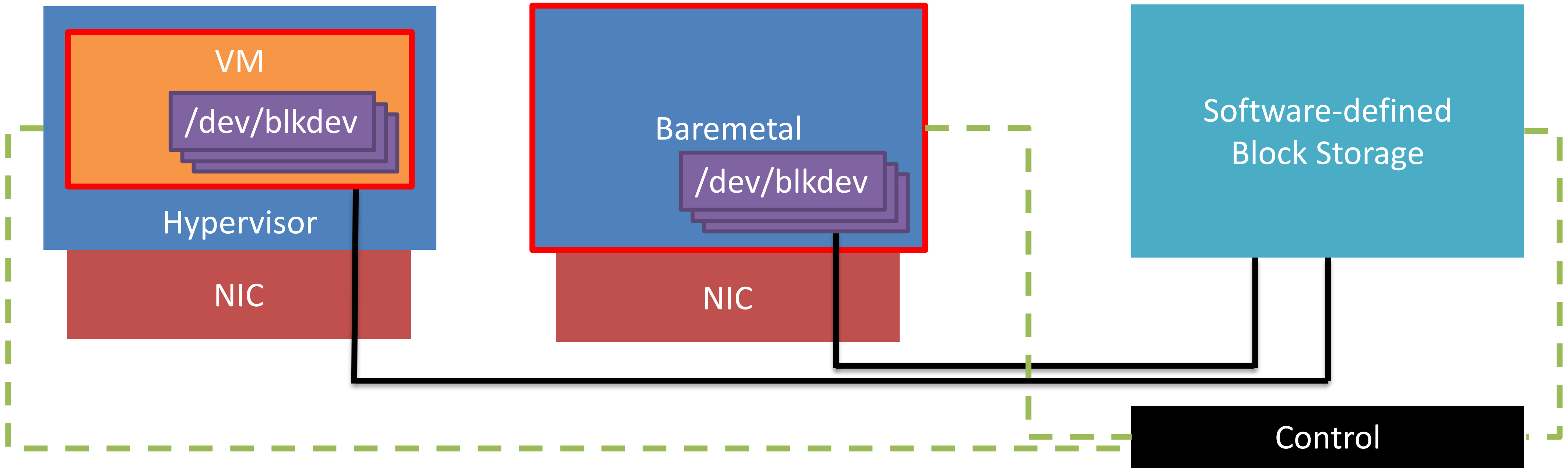
P4 programmable NIC + Arm

FPGA

Programmability

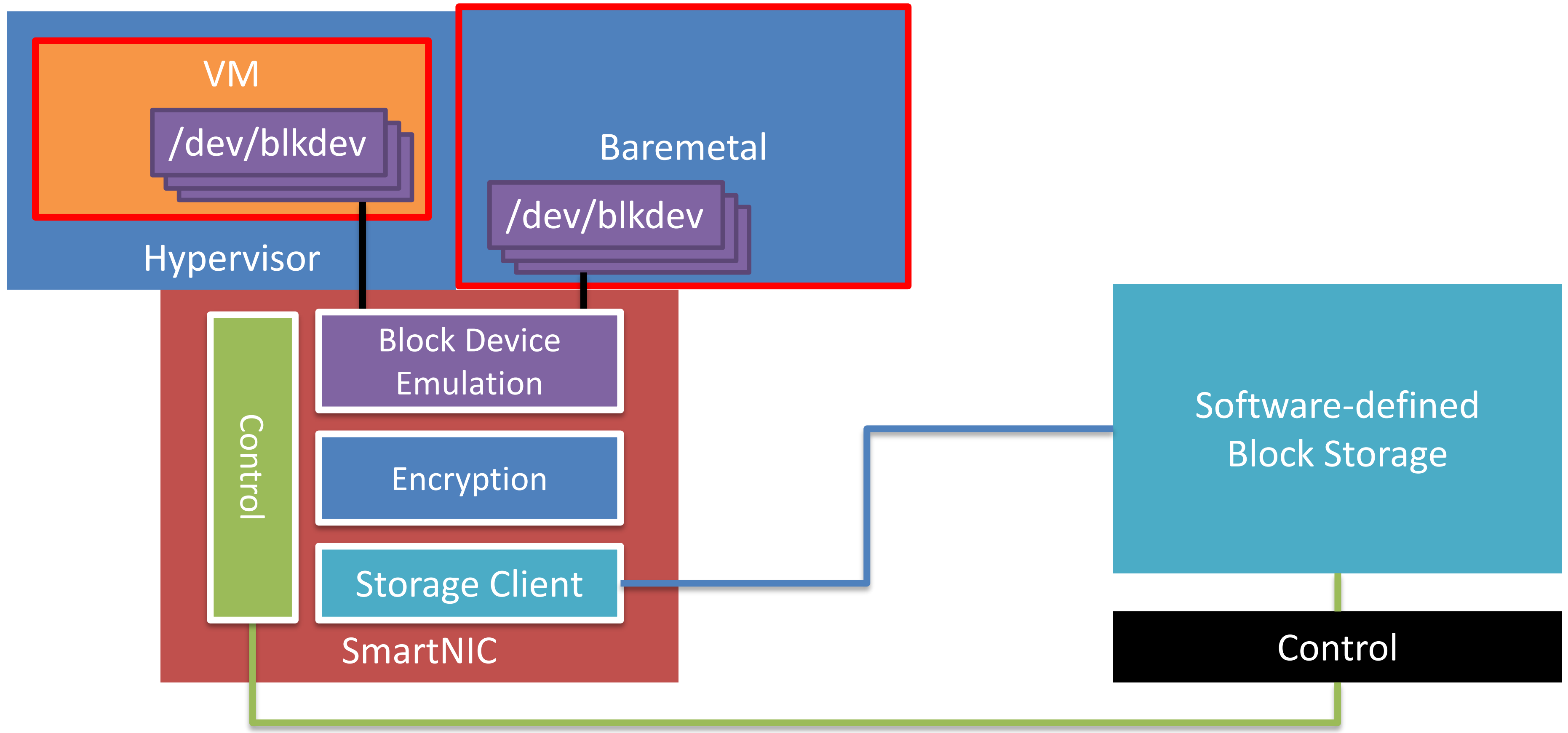
STORAGE VIRTUALIZATION

- **Goal: Transparent software-defined block storage for baremetal and virtual machines in cloud environments**
- **Requirements:**
  - Transparent block device emulation to the host (no special storage driver on the host)
  - For VM: datapath without the involvement of the hypervisor
  - Control plane that allows adding and removing block devices to the host (outside of red box)
  - Data encryption with key management

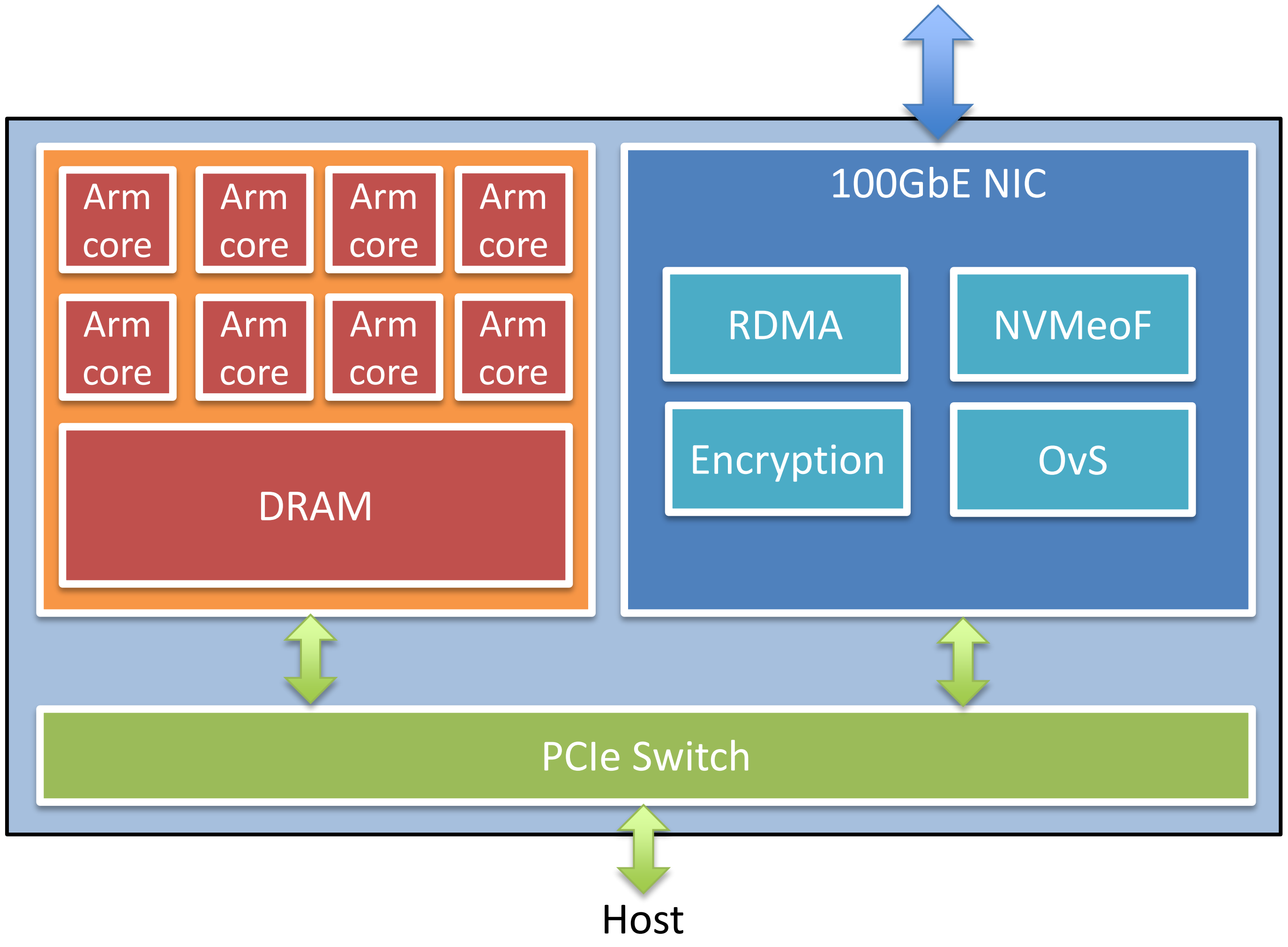


Perfect fit for SmartNICs?

STORAGE VIRTUALIZATION ON SMARTNIC

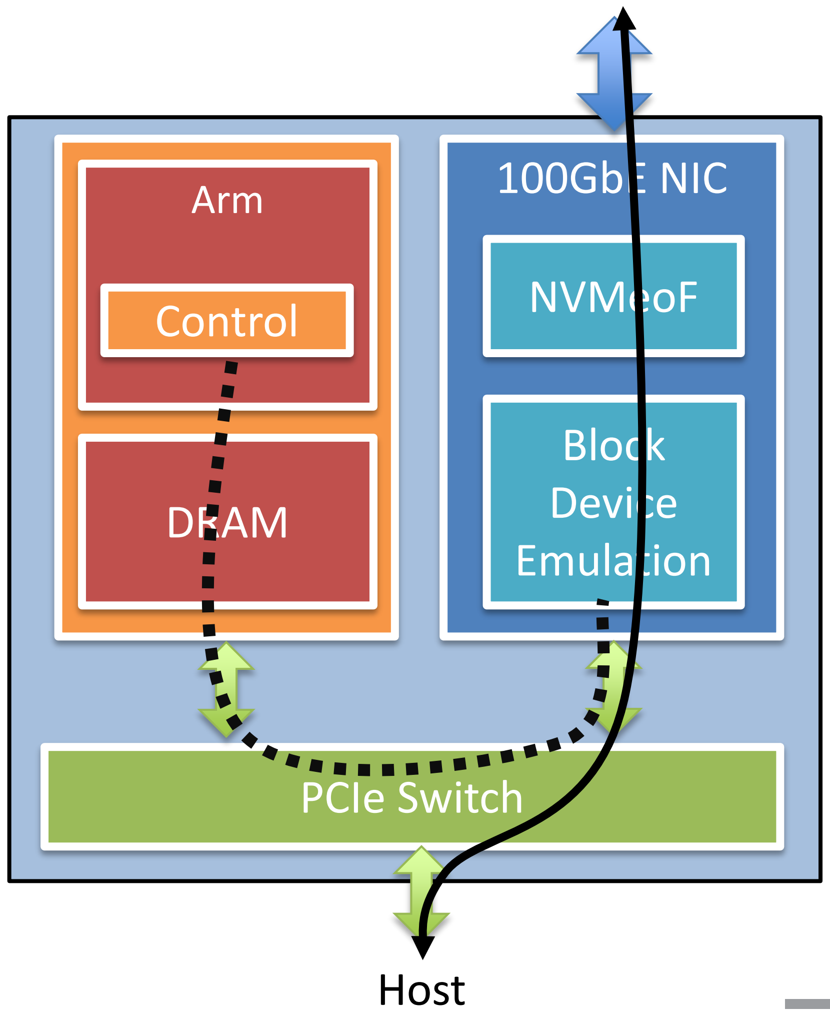


SMARTNIC: NIC + ARM

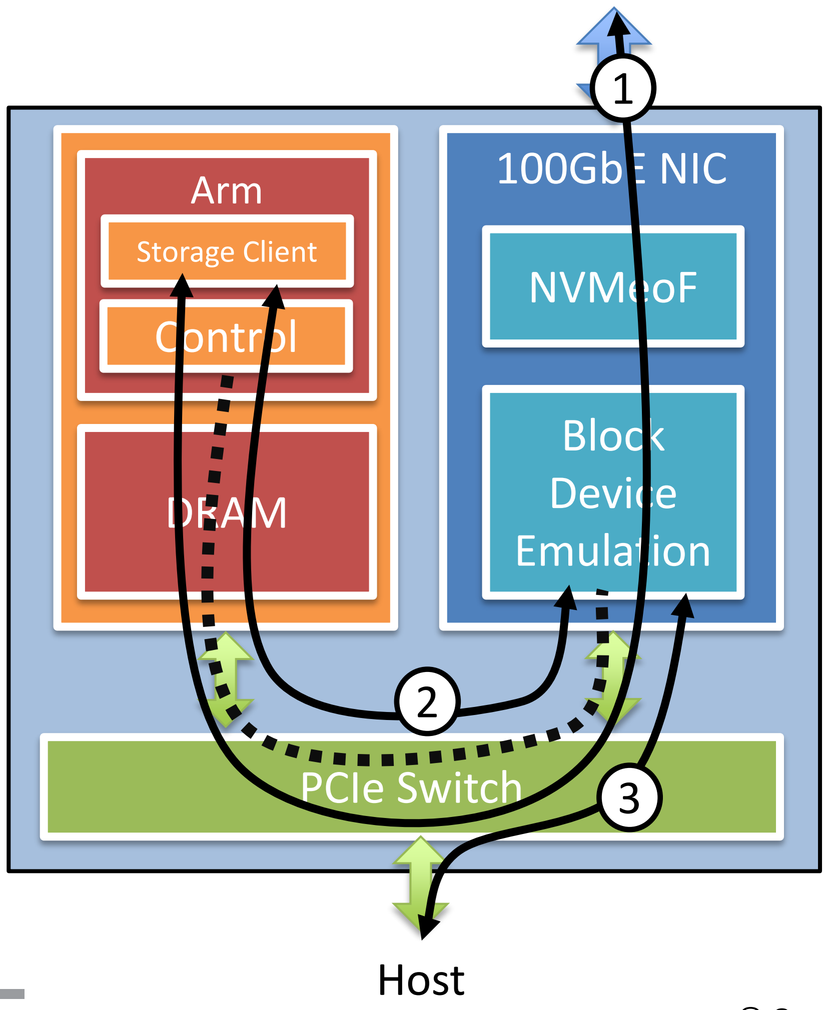


BLOCK DEVICE EMULATION

Fully Offloaded Block Device Emulation

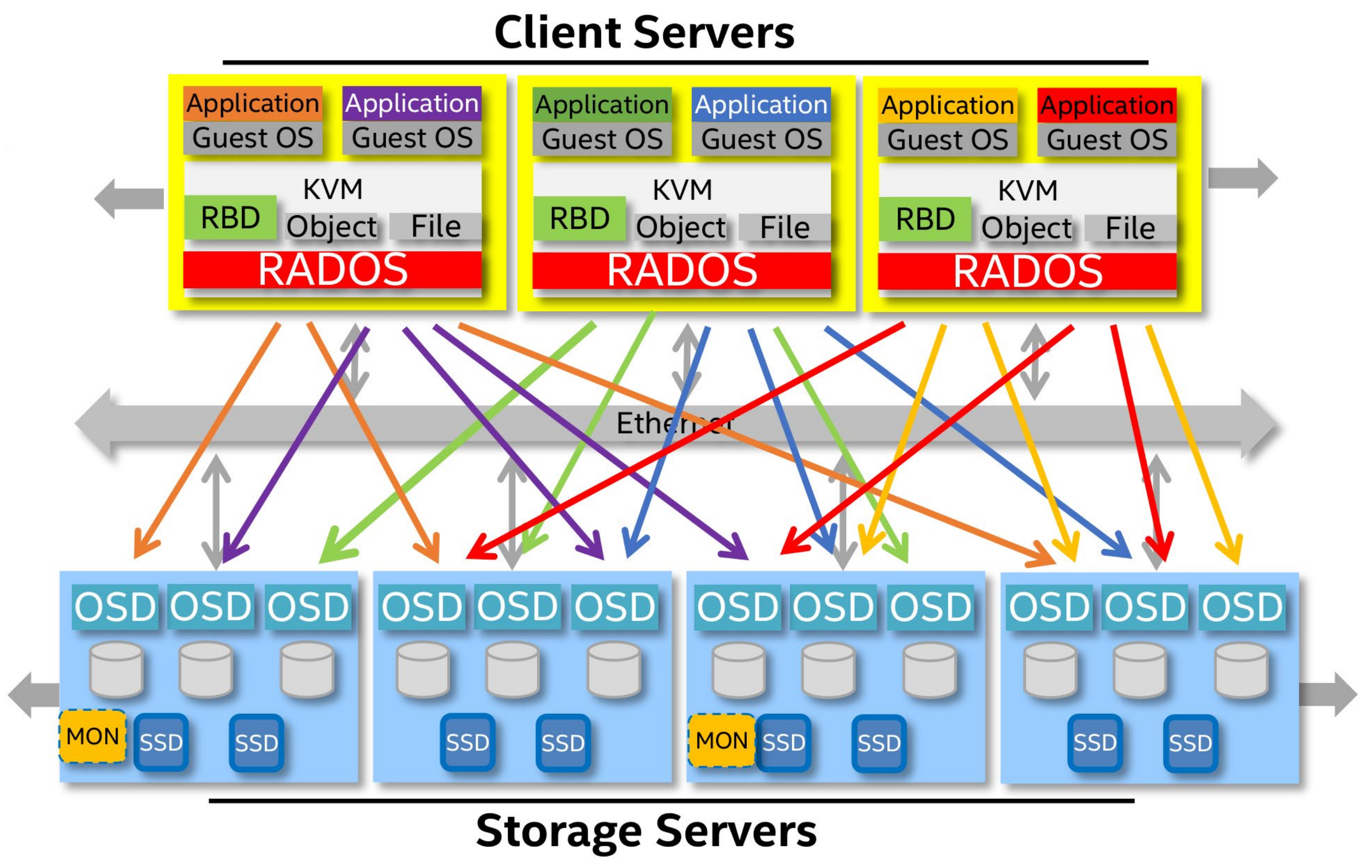


Block Device Emulation with Storage Client on Arm



CEPH

- Open-source, massively scalable, software-defined storage system
- Builds on Reliable Autonomic Distributed Object Store (RADOS)
- Offers **object** (RGW), **block** (RBD) and **file** (CEPH FS) API in a single unified storage cluster

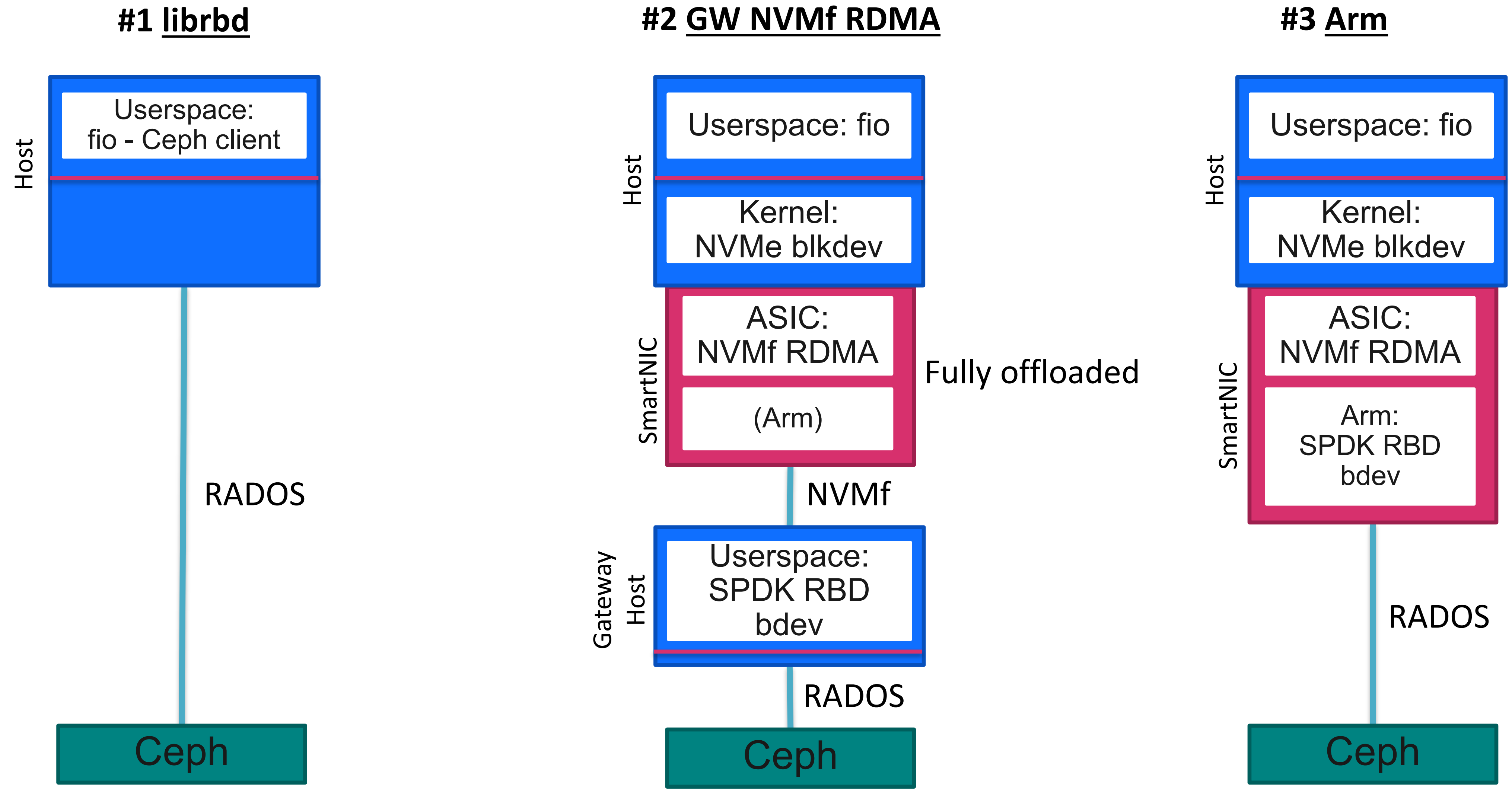


Source: https://de.slideshare.net/LarryCover/ceph-open-source-storage-software-optimizations-on-intel-architecture-for-cloud-workloads

OPENFABRICS ALLIANCE

EVALUATION

CONFIGURATIONS



SETUP AND BASELINE NETWORK PERFORMANCE

Hosts: Ceph server and clients

- 2x Intel(R) Xeon(R) CPU E5-2697 v4
- 1TB DDR4
- Ubuntu 20.04 – Linux kernel 5.5
- Mellanox ConnectX-5 100GbE*
- TCP Performance
  - RTT: **42.59usec @16KiB**
  - IOPS: 1 thread: 291.4K @16KiB; 8 threads: 717.1K @16KiB
  - Throughput: 1 thread: 43.2Gbit/s; 8 threads: **94.6Gbit/s**

*Server only

NVIDIA BlueField 2 Arm (MBF2M516A)

- 8x ArmV8 A72 cores @2Ghz
- 16GB DDR4
- CentOS 7.6 – Linux kernel 4.20
- Dual-port 100GbE
- TCP Performance
  - RTT: **116usec @16KiB**
  - IOPS: 4 threads: 331.1K @16KiB; 8 threads: 287.6K @16KiB
  - Throughput: 1 thread: 19.1Gbit/s; 4 threads: 47.8Gbit/s; 8 threads: **52.8Gbit/s**

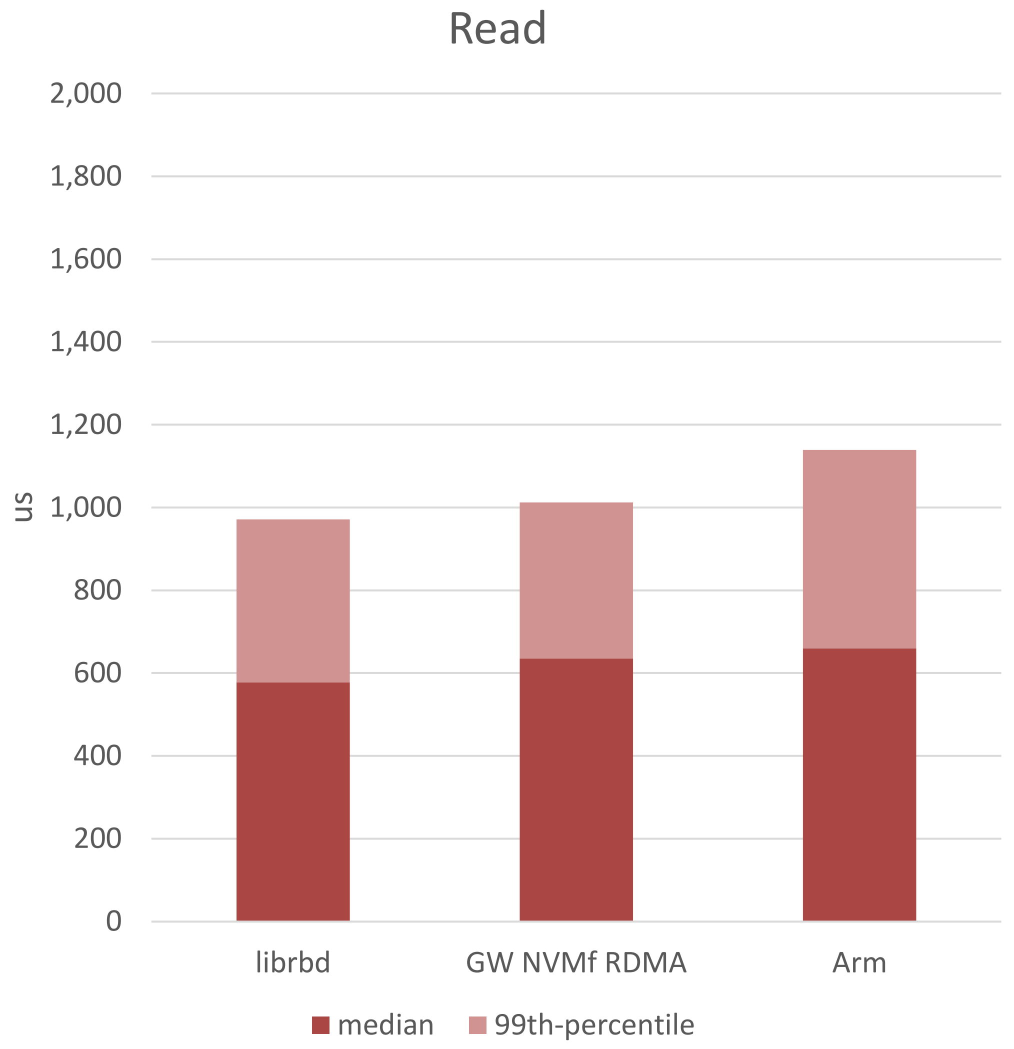
CEPH SETUP

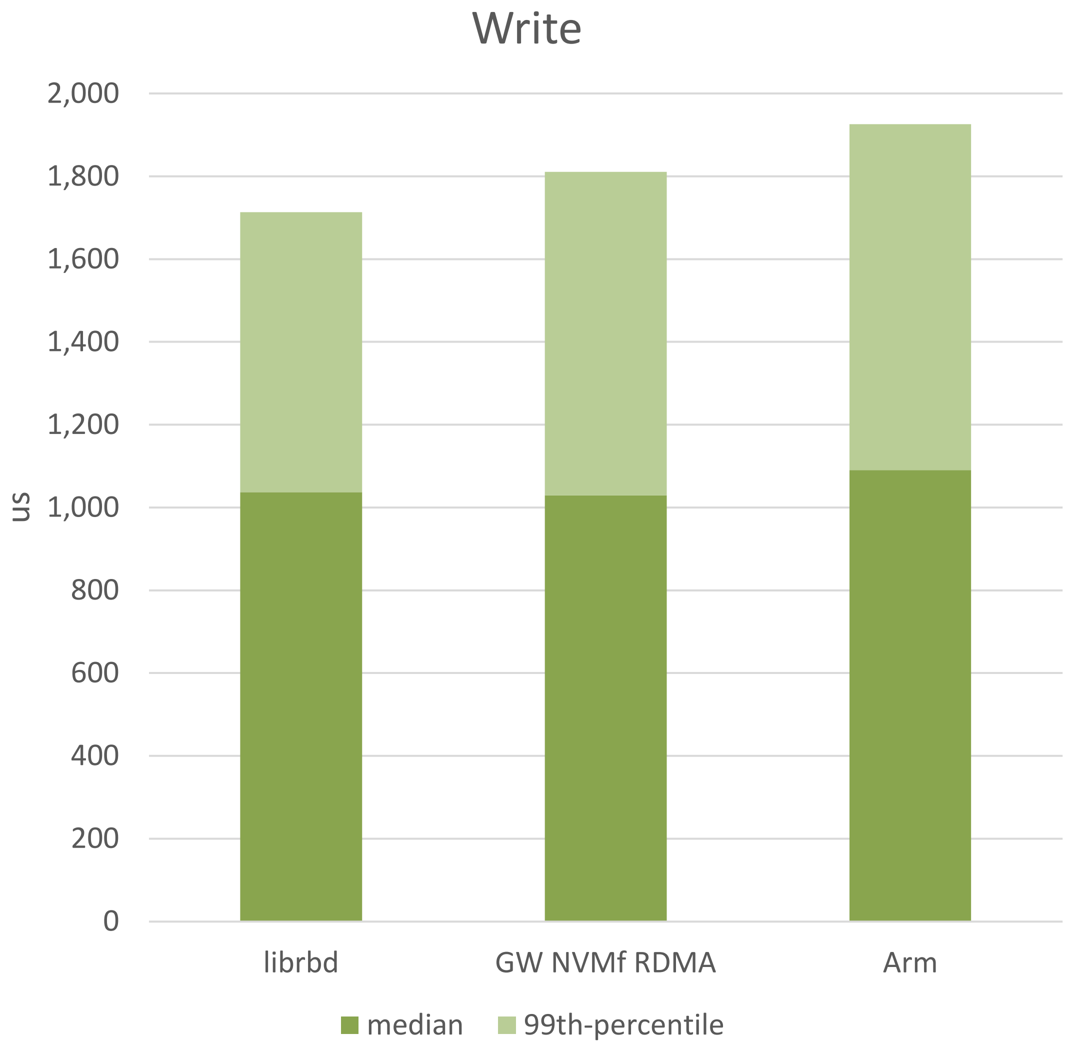
- **Ceph Octopus**
- **2 storage servers with 2 OSDs each = 4 OSDs total**
- **1x NVMe Samsung PM1725a per storage server (fio – blkdev)**

Read Latency:	93.71usec @16KiB
Write Latency:	17.6usec @16KiB
Read IOPS:	392K @16KiB
Write IOPS:	181K @16KiB
Read Throughput:	6314MiB/s
Write Throughput:	3189MiB/s

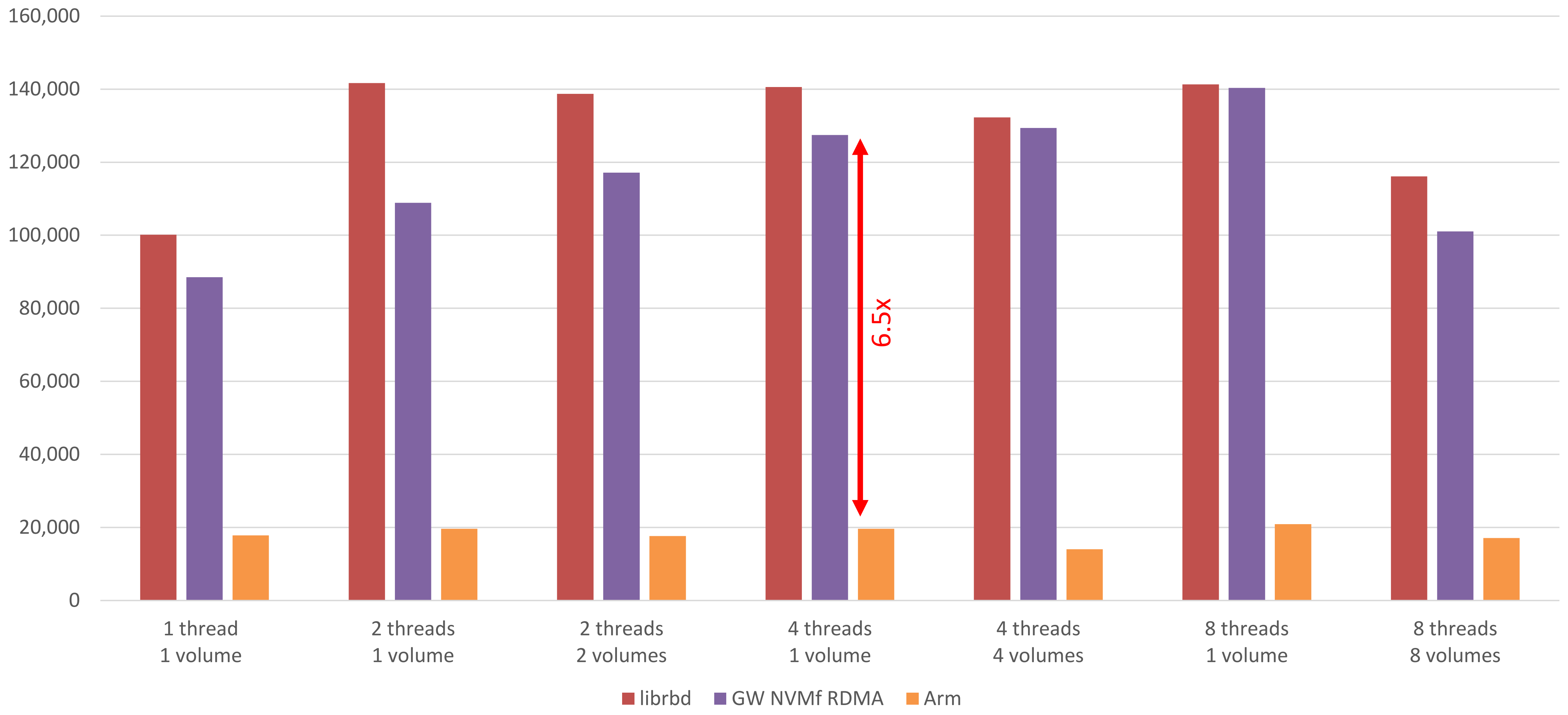
- **No replication: objective gateway and BlueField performance**
- **Default object size of 4MiB**
- **32 Ceph RBD images each 100GB**

LATENCY QD1@16KIB

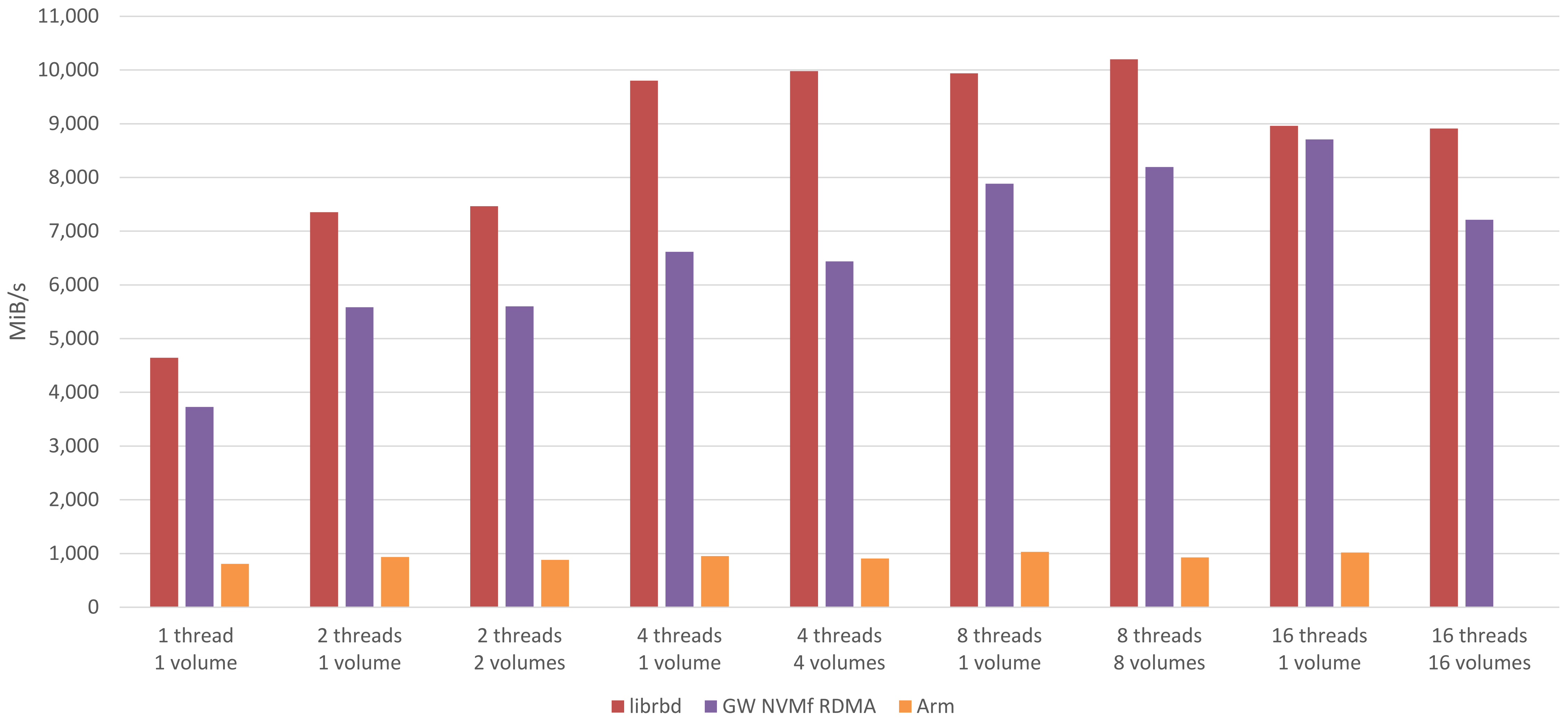




READ IOPS QD128@16KIB



READ THROUGHPUT QD16@1MIB



SUMMARY AND OUTLOOK

- **Embedded Arm on current generation commodity SmartNICs not fast enough for complex data path operations at line speed**
- **Gateway solution can be feasible but at the cost of extra compute resources and additional network hops**
  - Needs multipath NVMeoF for fault tolerance
- **Possible solutions for librbd on SmartNIC:**
  - More programmable SmartNIC, e.g. FPGA solution => downside complexity of programming
  - (Partial) protocol offload onto ASIC, e.g. TCP, RADOS => ASIC space is expensive which protocols to pick?
  - Optimized librbd / faster Arm cores => power requirements?

OPENFABRICS ALLIANCE

2021 OFA Virtual Workshop

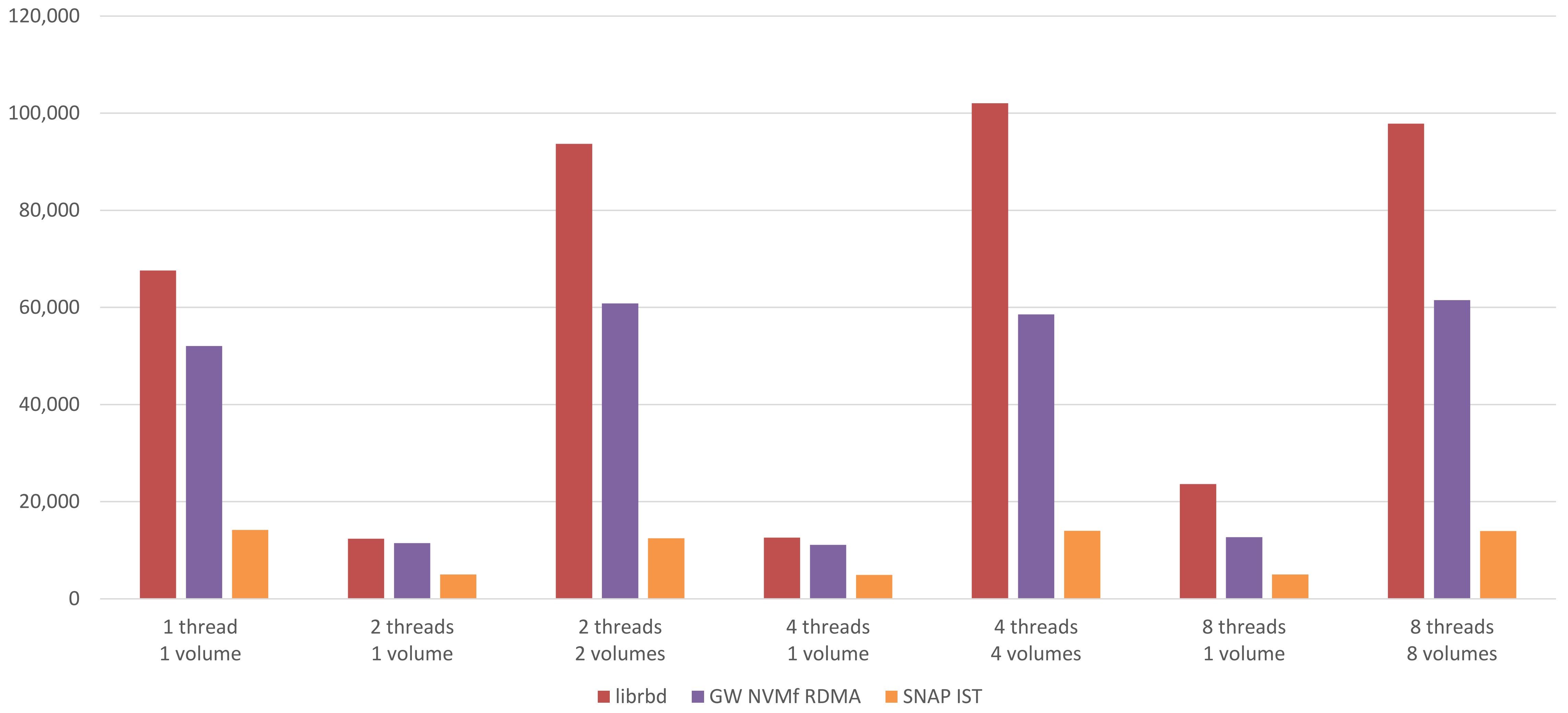
THANK YOU

Jonas Pfefferle, Nikolas Ioannou, Jose Castanos, Bernard Metzler

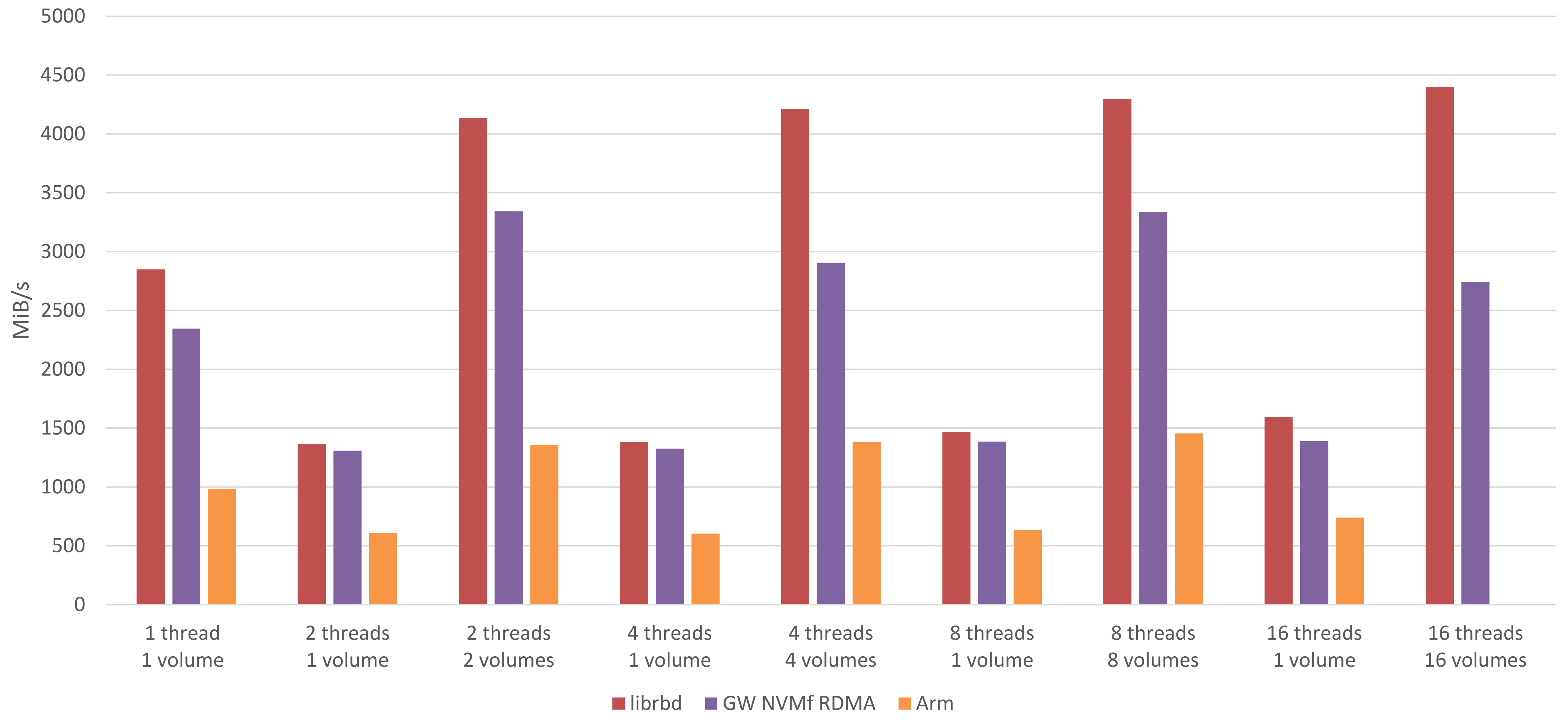
IBM Research

BACKUP

WRITE IOPS QD128@16KIB



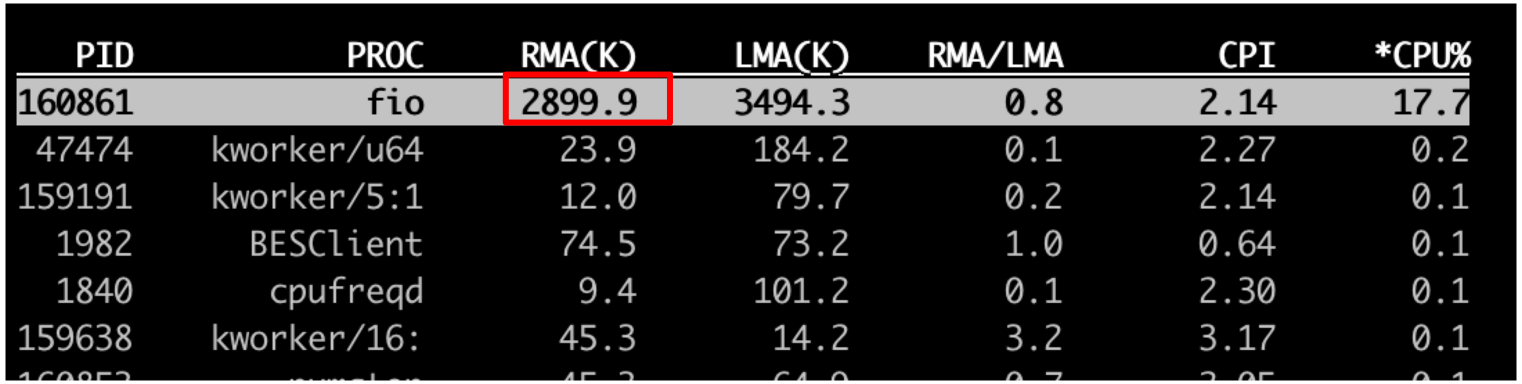
WRITE THROUGHPUT QD16@1MIB



CEPH LIBRBD PERFORMANCE

- **Evaluate client side Ceph performance => librbd**
- **Find bottlenecks**
- **Tune configuration options**
- **All benchmarks are single thread read IOPS QD128@16KiB**
- **Findings have been applied to SmartNIC evaluation above (except those that do not apply to SmartNIC configuration like NUMA)**
- **Test setup:**
  - 2x Intel(R) Xeon(R) CPU E5-2690@2.90GHz
  - 256GB RAM
  - 4x Samsung 960Pro 1TB
  - Mellanox ConnectX5 – 100GbE
  - Ceph Octopus 40osds on 5 machines (2 osds per drive), no replication, 64 volumes each 200GB

NUMATOP – RANDOM READ QD128 @16KIB

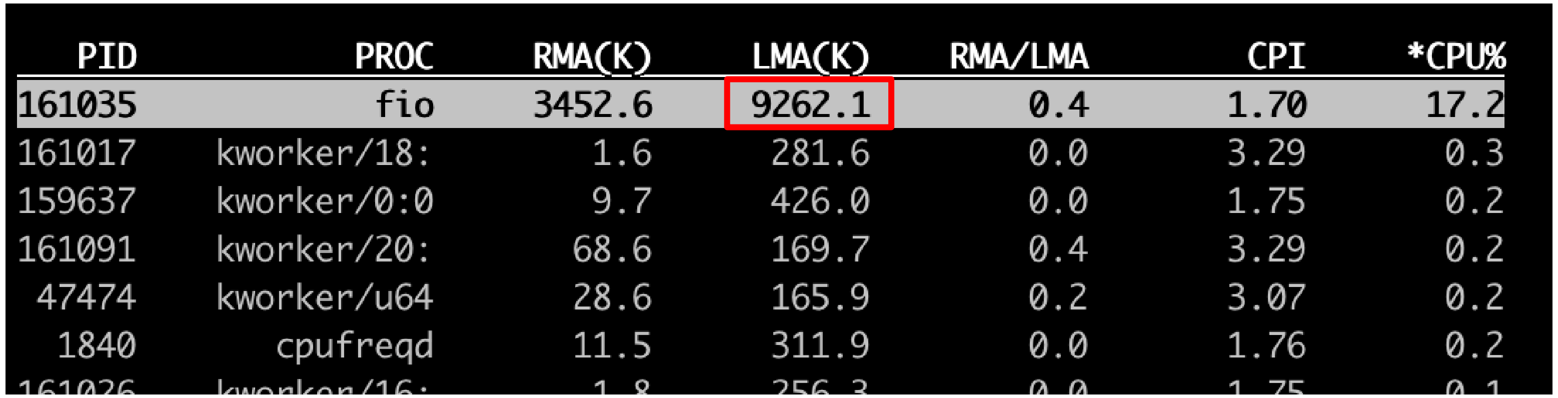


```
   PID           PROC    RMA(K)     LMA(K)    RMA/LMA        CPI     *CPU%
160861            fio    2899.9     3494.3        0.8       2.14      17.7
 47474     kworker/u64     23.9      184.2        0.1       2.27       0.2
159191     kworker/5:1     12.0       79.7        0.2       2.14       0.1
  1982       BESClient     74.5       73.2        1.0       0.64       0.1
  1840        cpufreqd      9.4      101.2        0.1       2.30       0.1
159638     kworker/16:     45.3       14.2        3.2       3.17       0.1
```

64K IOPS

=> 45% of memory accesses are remote

with numactl:

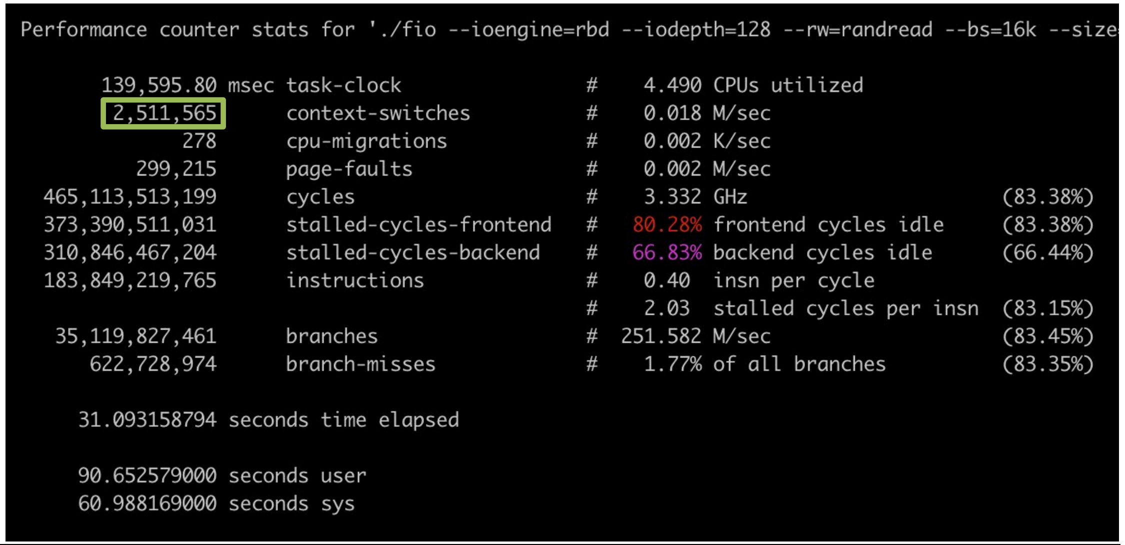


```
   PID           PROC    RMA(K)     LMA(K)    RMA/LMA        CPI     *CPU%
161035            fio    3452.6     9262.1        0.4       1.70      17.2
161017     kworker/18:      1.6      281.6        0.0       3.29       0.3
159637     kworker/0:0      9.7      426.0        0.0       1.75       0.2
161091     kworker/20:     68.6      169.7        0.4       3.29       0.2
 47474     kworker/u64     28.6      165.9        0.2       3.07       0.2
  1840        cpufreqd     11.5      311.9        0.0       1.76       0.2
```

90K IOPS

=> 27% of memory accesses are remote

PERF STAT - RANDOM READ QD128 @16KIB

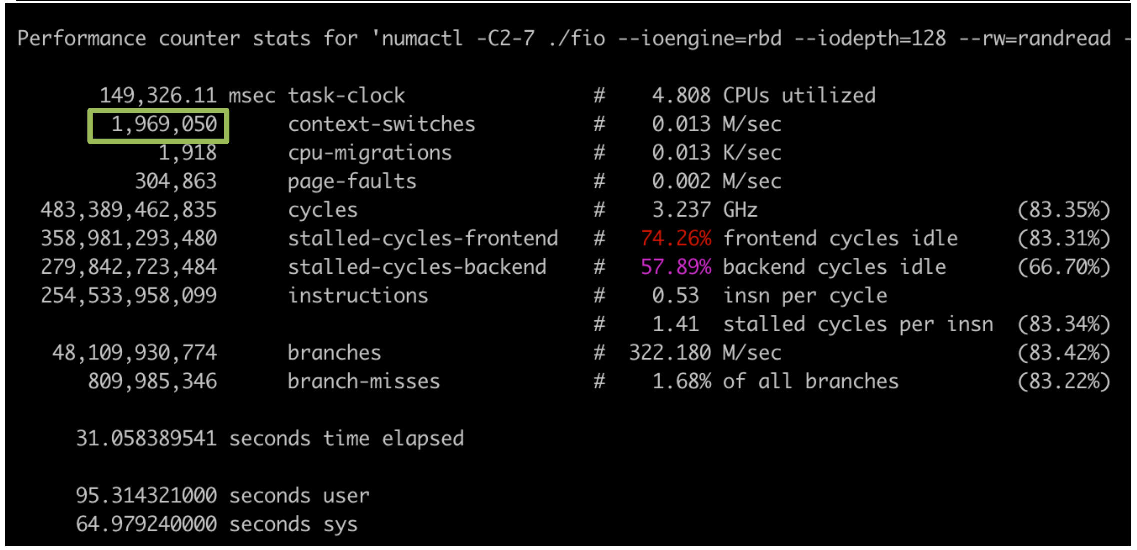


```
Performance counter stats for './fio --ioengine=rbd --iodepth=128 --rw=randread --bs=16k --size

      139,595.80 msec task-clock                #    4.490 CPUs utilized
       2,511,565      context-switches          #    0.018 M/sec
             278      cpu-migrations            #    0.002 K/sec
         299,215      page-faults               #    0.002 M/sec
 465,113,513,199      cycles                    #    3.332 GHz                      (83.38%)
 373,390,511,031      stalled-cycles-frontend   #   80.28% frontend cycles idle     (83.38%)
 310,846,467,204      stalled-cycles-backend    #   66.83% backend cycles idle      (66.44%)
 183,849,219,765      instructions              #    0.40  insn per cycle
                                                #    2.03  stalled cycles per insn  (83.15%)
  35,119,827,461      branches                  #  251.582 M/sec                    (83.45%)
     622,728,974      branch-misses             #    1.77% of all branches          (83.35%)

    31.093158794 seconds time elapsed

    90.652579000 seconds user
    60.988169000 seconds sys
```



```
Performance counter stats for 'numactl -C2-7 ./fio --ioengine=rbd --iodepth=128 --rw=randread -

      149,326.11 msec task-clock                #    4.808 CPUs utilized
       1,969,050      context-switches          #    0.013 M/sec
           1,918      cpu-migrations            #    0.013 K/sec
         304,863      page-faults               #    0.002 M/sec
 483,389,462,835      cycles                    #    3.237 GHz                      (83.35%)
 358,981,293,480      stalled-cycles-frontend   #   74.26% frontend cycles idle     (83.31%)
 279,842,723,484      stalled-cycles-backend    #   57.89% backend cycles idle      (66.70%)
 254,533,958,099      instructions              #    0.53  insn per cycle
                                                #    1.41  stalled cycles per insn  (83.34%)
  48,109,930,774      branches                  #  322.180 M/sec                    (83.42%)
     809,985,346      branch-misses             #    1.68% of all branches          (83.22%)

    31.058389541 seconds time elapsed

    95.314321000 seconds user
    64.979240000 seconds sys
```

64K IOPS

1,860,518 operations
=> **1.34** context-switches / operation

With numactl:
90K IOPS

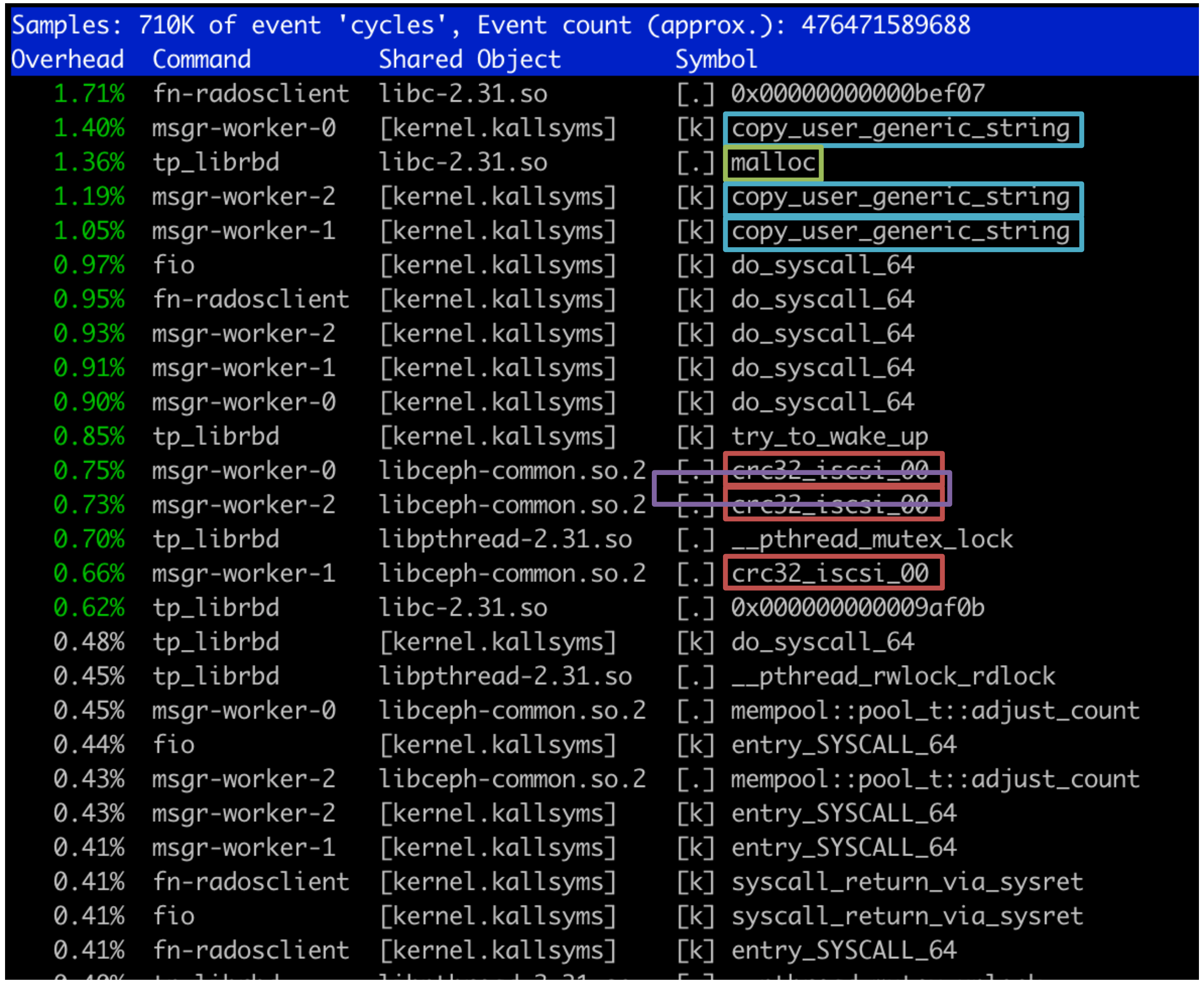
2,740,627 operations
=> **0.72** context-switches / operation

PERF – RANDOM READ QD128 @16KIB

With numactl:
90K IOPS

malloc

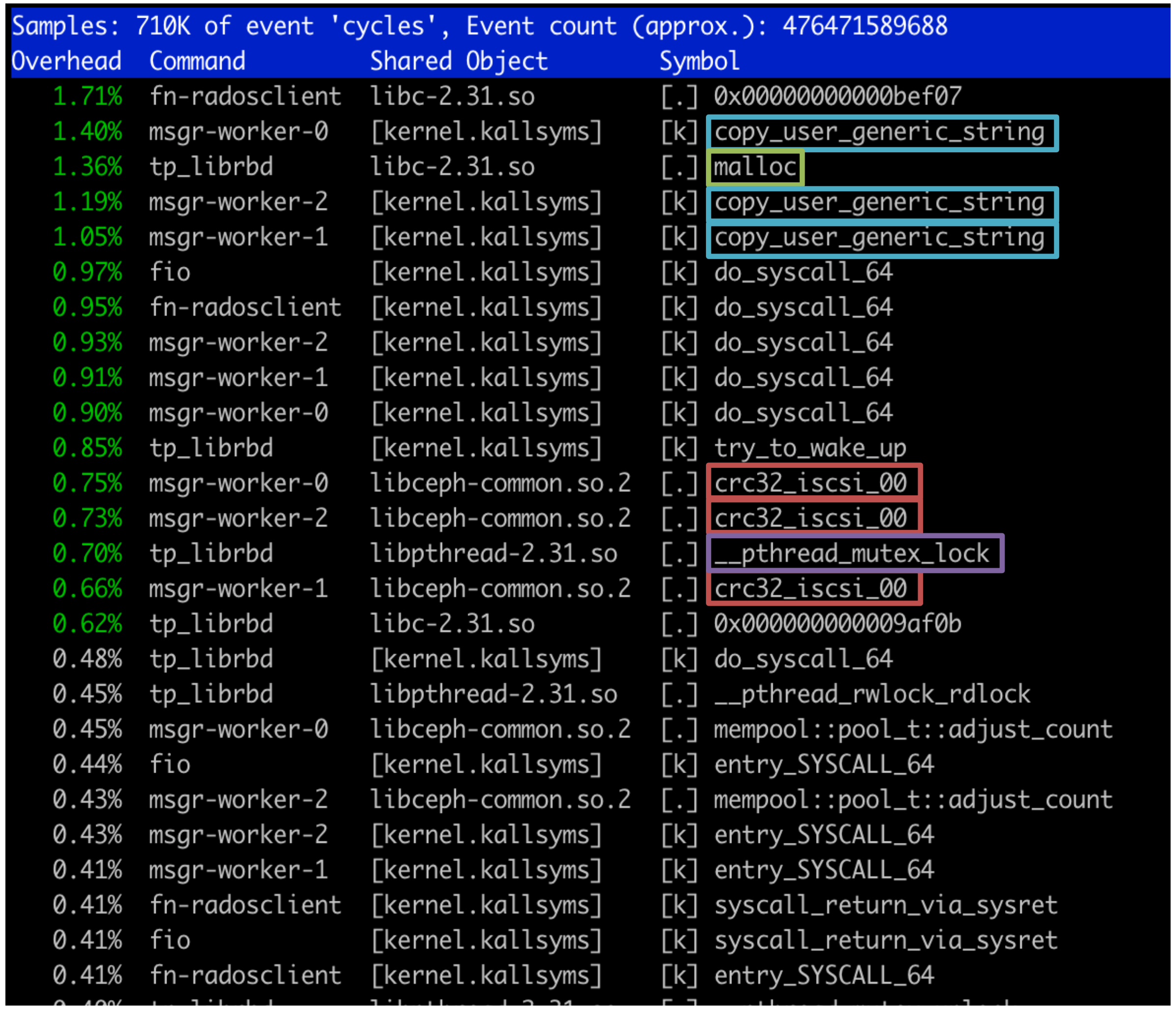
- librbd introduced jemalloc as default allocator a few years back
- However ***neither*** the official Ubuntu packages or the official Ceph packages are compiled with jemalloc support



```
Samples: 710K of event 'cycles', Event count (approx.): 476471589688
Overhead  Command       Shared Object        Symbol
   1.71%  fn-radosclient  libc-2.31.so         [.] 0x00000000000bef07
   1.40%  msgr-worker-0   [kernel.kallsyms]    [k] copy_user_generic_string
   1.36%  tp_librbd       libc-2.31.so         [.] malloc
   1.19%  msgr-worker-2   [kernel.kallsyms]    [k] copy_user_generic_string
   1.05%  msgr-worker-1   [kernel.kallsyms]    [k] copy_user_generic_string
   0.97%  fio             [kernel.kallsyms]    [k] do_syscall_64
   0.95%  fn-radosclient  [kernel.kallsyms]    [k] do_syscall_64
   0.93%  msgr-worker-2   [kernel.kallsyms]    [k] do_syscall_64
   0.91%  msgr-worker-1   [kernel.kallsyms]    [k] do_syscall_64
   0.90%  msgr-worker-0   [kernel.kallsyms]    [k] do_syscall_64
   0.85%  tp_librbd       [kernel.kallsyms]    [k] try_to_wake_up
   0.75%  msgr-worker-0   libceph-common.so.2  [.] crc32_iscsi_00
   0.73%  msgr-worker-2   libceph-common.so.2  [.] crc32_iscsi_00
   0.70%  tp_librbd       libpthread-2.31.so   [.] __pthread_mutex_lock
   0.66%  msgr-worker-1   libceph-common.so.2  [.] crc32_iscsi_00
   0.62%  tp_librbd       libc-2.31.so         [.] 0x000000000009af0b
   0.48%  tp_librbd       [kernel.kallsyms]    [k] do_syscall_64
   0.45%  tp_librbd       libpthread-2.31.so   [.] __pthread_rwlock_rdlock
   0.45%  msgr-worker-0   libceph-common.so.2  [.] mempool::pool_t::adjust_count
   0.44%  fio             [kernel.kallsyms]    [k] entry_SYSCALL_64
   0.43%  msgr-worker-2   libceph-common.so.2  [.] mempool::pool_t::adjust_count
   0.43%  msgr-worker-2   [kernel.kallsyms]    [k] entry_SYSCALL_64
   0.41%  msgr-worker-1   [kernel.kallsyms]    [k] entry_SYSCALL_64
   0.41%  fn-radosclient  [kernel.kallsyms]    [k] syscall_return_via_sysret
   0.41%  fio             [kernel.kallsyms]    [k] syscall_return_via_sysret
   0.41%  fn-radosclient  [kernel.kallsyms]    [k] entry_SYSCALL_64
```

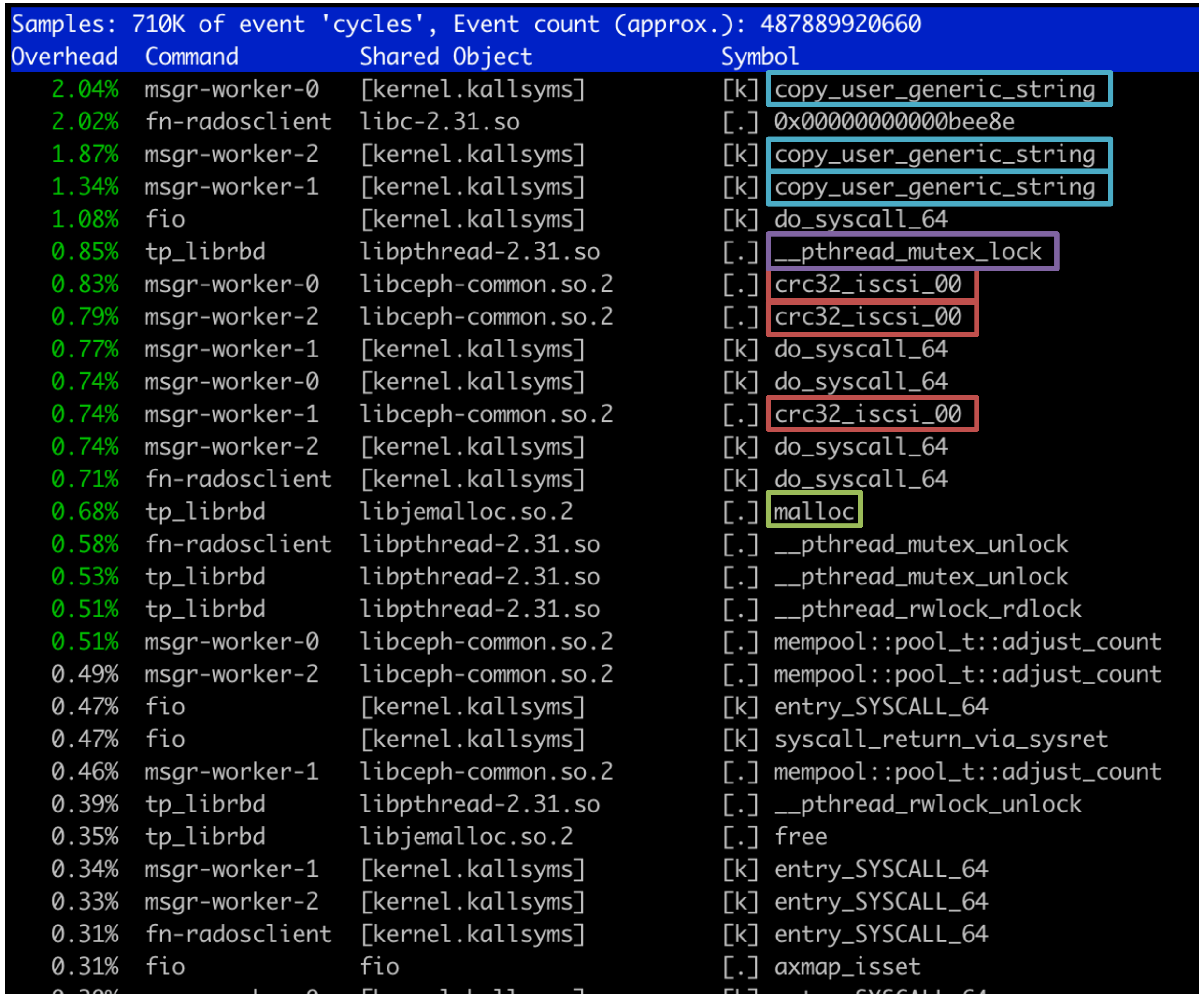
PERF – RANDOM READ QD128 @16KIB

With numactl: **90K IOPS**



```
Samples: 710K of event 'cycles', Event count (approx.): 476471589688
Overhead  Command       Shared Object         Symbol
   1.71%  fn-radosclient libc-2.31.so          [.] 0x00000000000bef07
   1.40%  msgr-worker-0  [kernel.kallsyms]     [k] copy_user_generic_string
   1.36%  tp_librbd      libc-2.31.so          [.] malloc
   1.19%  msgr-worker-2  [kernel.kallsyms]     [k] copy_user_generic_string
   1.05%  msgr-worker-1  [kernel.kallsyms]     [k] copy_user_generic_string
   0.97%  fio            [kernel.kallsyms]     [k] do_syscall_64
   0.95%  fn-radosclient [kernel.kallsyms]     [k] do_syscall_64
   0.93%  msgr-worker-2  [kernel.kallsyms]     [k] do_syscall_64
   0.91%  msgr-worker-1  [kernel.kallsyms]     [k] do_syscall_64
   0.90%  msgr-worker-0  [kernel.kallsyms]     [k] do_syscall_64
   0.85%  tp_librbd      [kernel.kallsyms]     [k] try_to_wake_up
   0.75%  msgr-worker-0  libceph-common.so.2   [.] crc32_iscsi_00
   0.73%  msgr-worker-2  libceph-common.so.2   [.] crc32_iscsi_00
   0.70%  tp_librbd      libpthread-2.31.so    [.] __pthread_mutex_lock
   0.66%  msgr-worker-1  libceph-common.so.2   [.] crc32_iscsi_00
   0.62%  tp_librbd      libc-2.31.so          [.] 0x000000000009af0b
   0.48%  tp_librbd      [kernel.kallsyms]     [k] do_syscall_64
   0.45%  tp_librbd      libpthread-2.31.so    [.] __pthread_rwlock_rdlock
   0.45%  msgr-worker-0  libceph-common.so.2   [.] mempool::pool_t::adjust_count
   0.44%  fio            [kernel.kallsyms]     [k] entry_SYSCALL_64
   0.43%  msgr-worker-2  libceph-common.so.2   [.] mempool::pool_t::adjust_count
   0.43%  msgr-worker-2  [kernel.kallsyms]     [k] entry_SYSCALL_64
   0.41%  msgr-worker-1  [kernel.kallsyms]     [k] entry_SYSCALL_64
   0.41%  fn-radosclient [kernel.kallsyms]     [k] syscall_return_via_sysret
   0.41%  fio            [kernel.kallsyms]     [k] syscall_return_via_sysret
   0.41%  fn-radosclient [kernel.kallsyms]     [k] entry_SYSCALL_64
```

With numactl + jemalloc: **100K IOPS**



```
Samples: 710K of event 'cycles', Event count (approx.): 487889920660
Overhead  Command        Shared Object         Symbol
   2.04%  msgr-worker-0  [kernel.kallsyms]     [k] copy_user_generic_string
   2.02%  fn-radosclient libc-2.31.so          [.] 0x00000000000bee8e
   1.87%  msgr-worker-2  [kernel.kallsyms]     [k] copy_user_generic_string
   1.34%  msgr-worker-1  [kernel.kallsyms]     [k] copy_user_generic_string
   1.08%  fio            [kernel.kallsyms]     [k] do_syscall_64
   0.85%  tp_librbd      libpthread-2.31.so    [.] __pthread_mutex_lock
   0.83%  msgr-worker-0  libceph-common.so.2   [.] crc32_iscsi_00
   0.79%  msgr-worker-2  libceph-common.so.2   [.] crc32_iscsi_00
   0.77%  msgr-worker-1  [kernel.kallsyms]     [k] do_syscall_64
   0.74%  msgr-worker-0  [kernel.kallsyms]     [k] do_syscall_64
   0.74%  msgr-worker-1  libceph-common.so.2   [.] crc32_iscsi_00
   0.74%  msgr-worker-2  [kernel.kallsyms]     [k] do_syscall_64
   0.71%  fn-radosclient [kernel.kallsyms]     [k] do_syscall_64
   0.68%  tp_librbd      libjemalloc.so.2      [.] malloc
   0.58%  fn-radosclient libpthread-2.31.so    [.] __pthread_mutex_unlock
   0.53%  tp_librbd      libpthread-2.31.so    [.] __pthread_mutex_unlock
   0.51%  tp_librbd      libpthread-2.31.so    [.] __pthread_rwlock_rdlock
   0.51%  msgr-worker-0  libceph-common.so.2   [.] mempool::pool_t::adjust_count
   0.49%  msgr-worker-2  libceph-common.so.2   [.] mempool::pool_t::adjust_count
   0.47%  fio            [kernel.kallsyms]     [k] entry_SYSCALL_64
   0.47%  fio            [kernel.kallsyms]     [k] syscall_return_via_sysret
   0.46%  msgr-worker-1  libceph-common.so.2   [.] mempool::pool_t::adjust_count
   0.39%  tp_librbd      libpthread-2.31.so    [.] __pthread_rwlock_unlock
   0.35%  tp_librbd      libjemalloc.so.2      [.] free
   0.34%  msgr-worker-1  [kernel.kallsyms]     [k] entry_SYSCALL_64
   0.33%  msgr-worker-2  [kernel.kallsyms]     [k] entry_SYSCALL_64
   0.31%  fn-radosclient [kernel.kallsyms]     [k] entry_SYSCALL_64
   0.31%  fio            fio                   [.] axmap_isset
```

CEPH OPTIONS

rbd_disable_zero_copy_writes

- Default true because buffer should not be changed while “owned” by librbd, i.e. if client writes into buffer => CRC error
- Well behaved client should not touch buffer

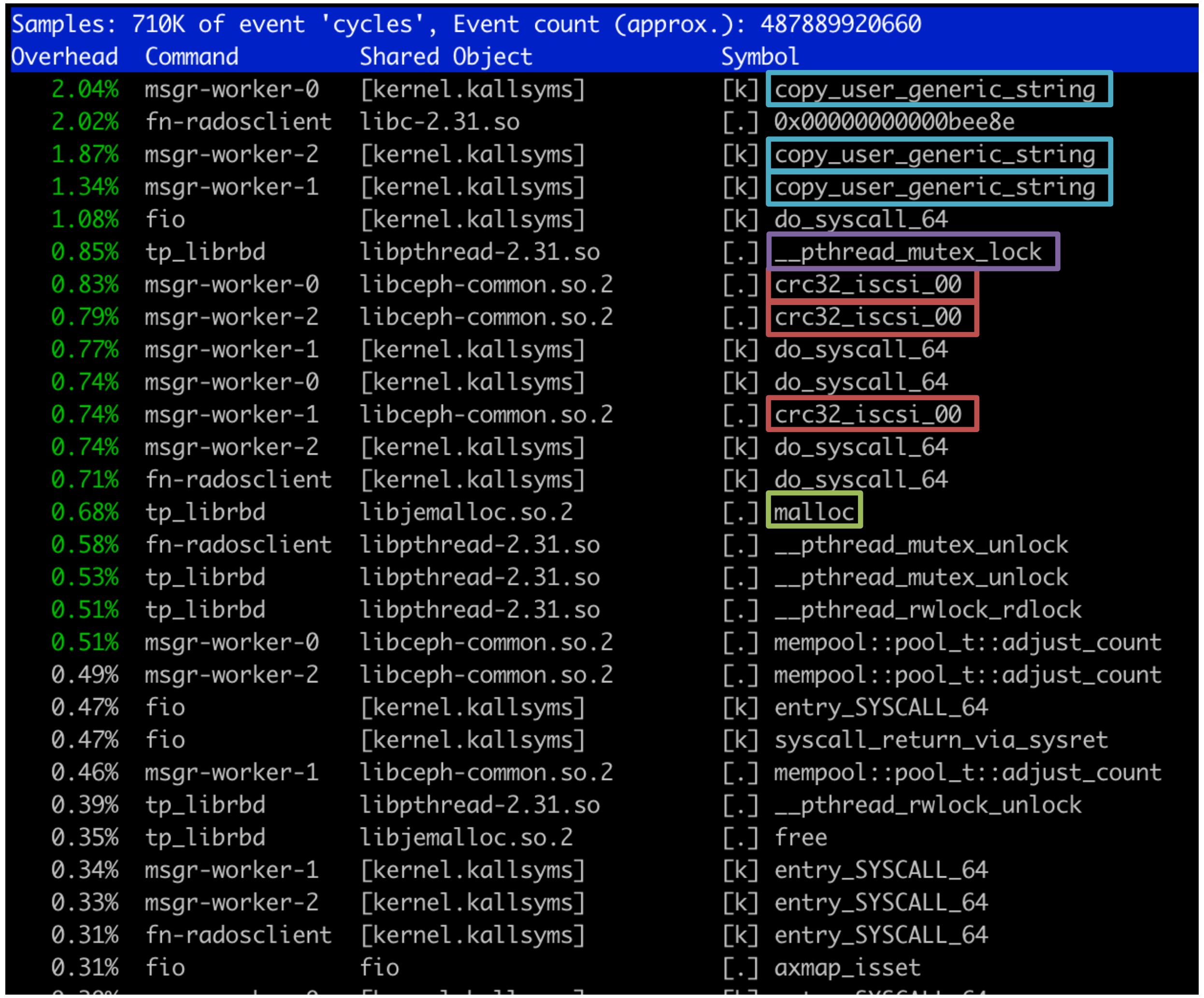
rbd_cache

- Client local cache with default size of 32MB
- Enabled by default
- Decreased write performance on fast backends
- Introduces additional copies on read

=> ***Disabled*** in all tests

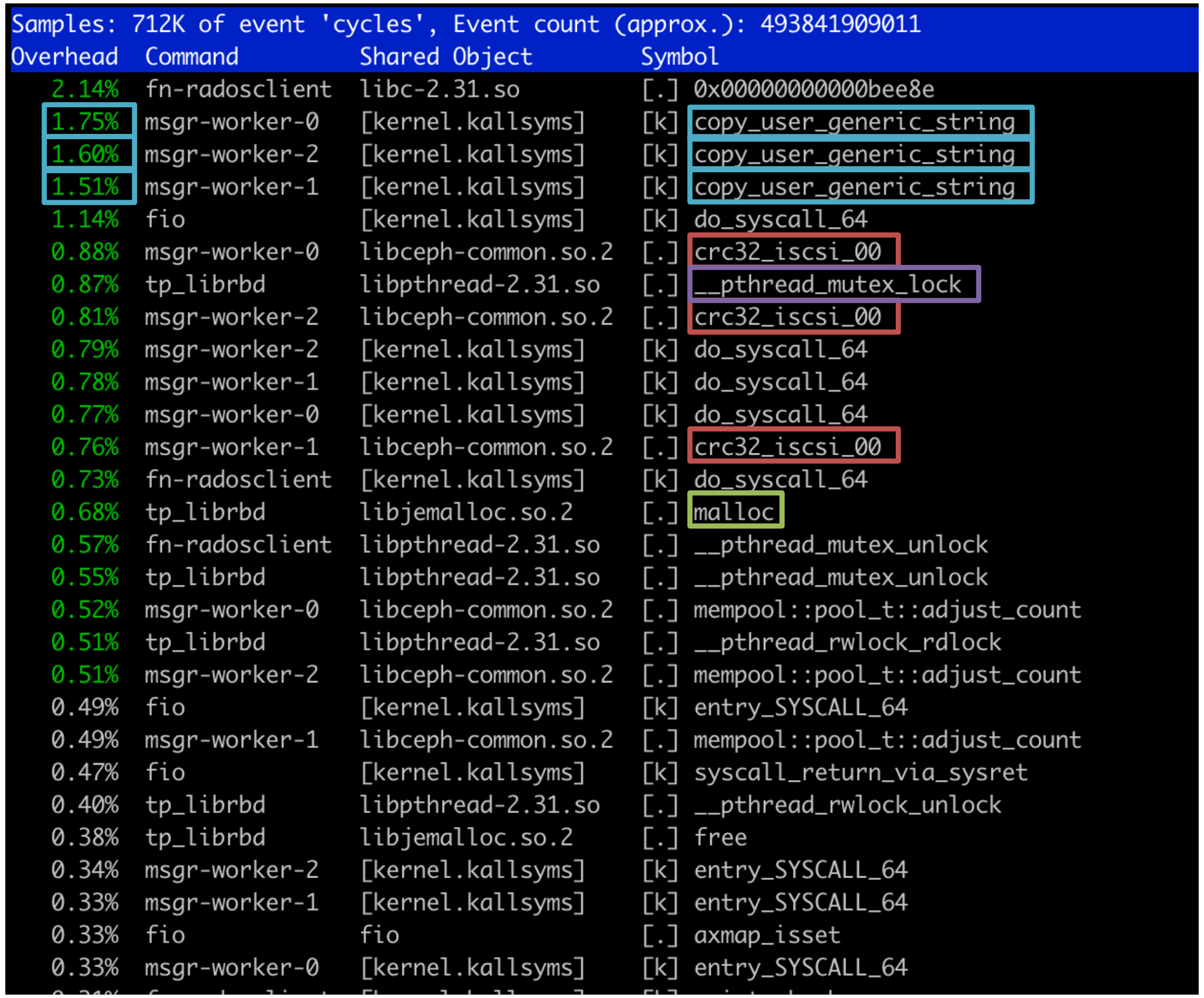
PERF – RANDOM READ QD128 @16KIB

numactl + jemalloc: **100K IOPS**



```
Samples: 710K of event 'cycles', Event count (approx.): 487889920660
Overhead  Command         Shared Object          Symbol
   2.04%  msgr-worker-0   [kernel.kallsyms]      [k] copy_user_generic_string
   2.02%  fn-radosclient  libc-2.31.so           [.] 0x00000000000bee8e
   1.87%  msgr-worker-2   [kernel.kallsyms]      [k] copy_user_generic_string
   1.34%  msgr-worker-1   [kernel.kallsyms]      [k] copy_user_generic_string
   1.08%  fio             [kernel.kallsyms]      [k] do_syscall_64
   0.85%  tp_librbd       libpthread-2.31.so     [.] __pthread_mutex_lock
   0.83%  msgr-worker-0   libceph-common.so.2    [.] crc32_iscsi_00
   0.79%  msgr-worker-2   libceph-common.so.2    [.] crc32_iscsi_00
   0.77%  msgr-worker-1   [kernel.kallsyms]      [k] do_syscall_64
   0.74%  msgr-worker-0   [kernel.kallsyms]      [k] do_syscall_64
   0.74%  msgr-worker-1   libceph-common.so.2    [.] crc32_iscsi_00
   0.74%  msgr-worker-2   [kernel.kallsyms]      [k] do_syscall_64
   0.71%  fn-radosclient  [kernel.kallsyms]      [k] do_syscall_64
   0.68%  tp_librbd       libjemalloc.so.2       [.] malloc
   0.58%  fn-radosclient  libpthread-2.31.so     [.] __pthread_mutex_unlock
   0.53%  tp_librbd       libpthread-2.31.so     [.] __pthread_mutex_unlock
   0.51%  tp_librbd       libpthread-2.31.so     [.] __pthread_rwlock_rdlock
   0.51%  msgr-worker-0   libceph-common.so.2    [.] mempool::pool_t::adjust_count
   0.49%  msgr-worker-2   libceph-common.so.2    [.] mempool::pool_t::adjust_count
   0.47%  fio             [kernel.kallsyms]      [k] entry_SYSCALL_64
   0.47%  fio             [kernel.kallsyms]      [k] syscall_return_via_sysret
   0.46%  msgr-worker-1   libceph-common.so.2    [.] mempool::pool_t::adjust_count
   0.39%  tp_librbd       libpthread-2.31.so     [.] __pthread_rwlock_unlock
   0.35%  tp_librbd       libjemalloc.so.2       [.] free
   0.34%  msgr-worker-1   [kernel.kallsyms]      [k] entry_SYSCALL_64
   0.33%  msgr-worker-2   [kernel.kallsyms]      [k] entry_SYSCALL_64
   0.31%  fn-radosclient  [kernel.kallsyms]      [k] entry_SYSCALL_64
   0.31%  fio             fio                    [.] axmap_isset
```

numactl + jemalloc + rbd_disable_zero_copy_writes=false: **110K IOPS**



```
Samples: 712K of event 'cycles', Event count (approx.): 493841909011
Overhead  Command         Shared Object          Symbol
   2.14%  fn-radosclient  libc-2.31.so           [.] 0x00000000000bee8e
   1.75%  msgr-worker-0   [kernel.kallsyms]      [k] copy_user_generic_string
   1.60%  msgr-worker-2   [kernel.kallsyms]      [k] copy_user_generic_string
   1.51%  msgr-worker-1   [kernel.kallsyms]      [k] copy_user_generic_string
   1.14%  fio             [kernel.kallsyms]      [k] do_syscall_64
   0.88%  msgr-worker-0   libceph-common.so.2    [.] crc32_iscsi_00
   0.87%  tp_librbd       libpthread-2.31.so     [.] __pthread_mutex_lock
   0.81%  msgr-worker-2   libceph-common.so.2    [.] crc32_iscsi_00
   0.79%  msgr-worker-2   [kernel.kallsyms]      [k] do_syscall_64
   0.78%  msgr-worker-1   [kernel.kallsyms]      [k] do_syscall_64
   0.77%  msgr-worker-0   [kernel.kallsyms]      [k] do_syscall_64
   0.76%  msgr-worker-1   libceph-common.so.2    [.] crc32_iscsi_00
   0.73%  fn-radosclient  [kernel.kallsyms]      [k] do_syscall_64
   0.68%  tp_librbd       libjemalloc.so.2       [.] malloc
   0.57%  fn-radosclient  libpthread-2.31.so     [.] __pthread_mutex_unlock
   0.55%  tp_librbd       libpthread-2.31.so     [.] __pthread_mutex_unlock
   0.52%  msgr-worker-0   libceph-common.so.2    [.] mempool::pool_t::adjust_count
   0.51%  tp_librbd       libpthread-2.31.so     [.] __pthread_rwlock_rdlock
   0.51%  msgr-worker-2   libceph-common.so.2    [.] mempool::pool_t::adjust_count
   0.49%  fio             [kernel.kallsyms]      [k] entry_SYSCALL_64
   0.49%  msgr-worker-1   libceph-common.so.2    [.] mempool::pool_t::adjust_count
   0.47%  fio             [kernel.kallsyms]      [k] syscall_return_via_sysret
   0.40%  tp_librbd       libpthread-2.31.so     [.] __pthread_rwlock_unlock
   0.38%  tp_librbd       libjemalloc.so.2       [.] free
   0.34%  msgr-worker-2   [kernel.kallsyms]      [k] entry_SYSCALL_64
   0.33%  msgr-worker-1   [kernel.kallsyms]      [k] entry_SYSCALL_64
   0.33%  fio             fio                    [.] axmap_isset
   0.33%  msgr-worker-0   [kernel.kallsyms]      [k] entry_SYSCALL_64
```

THREADS, THREADS AND MORE THREADS

rbd_op_threads

- Default 1
- Used for librbd::thread_pool
- All I/O is submitted to ioqueue associated with thread_pool
- ***No performance improvements*** seen with >1 threads

ms_async_op_threads

- Messenger threads handle all messages from librbd/librados to osds/mon/mgr
- Default of ***3 threads*** seem to be a ***sweetspot*** for a single process
  => no significant improvement increasing to 4 or more threads
  - 1 Thread: 44K IOPS
  - 2 Threads: 85K IOPS
  - 3 Threads: 110K IOPS
  - 4 Threads: 112K IOPS

Note: New Ceph version (v16.0.0 not released) based on boost asio
librados_thread_count(2) and client_asio_thread_count(2)
=> alpha performance ~83K IOPS

SPECTRE/MELTDOWN MITIGATIONS

Context switches due to socket operations

- **0.55 context switches/operation**

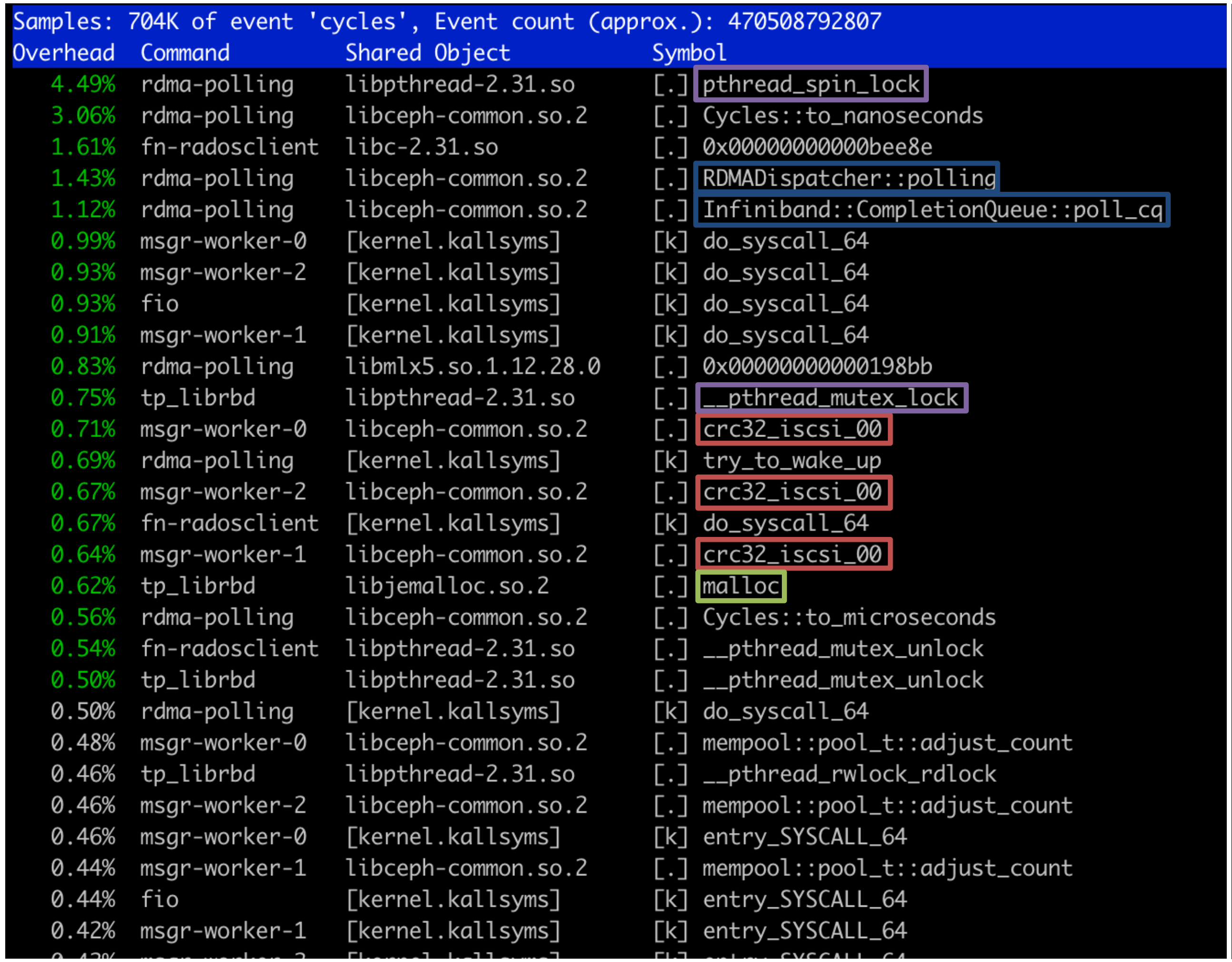
Spectre/Meltdown mitigations makes context switches expensive (Intel mostly)
=> Disable Spectre/Meltdown mitigations

- Kernel command line = *“mitigations=off”*

110K IOPS => **115K IOPS**

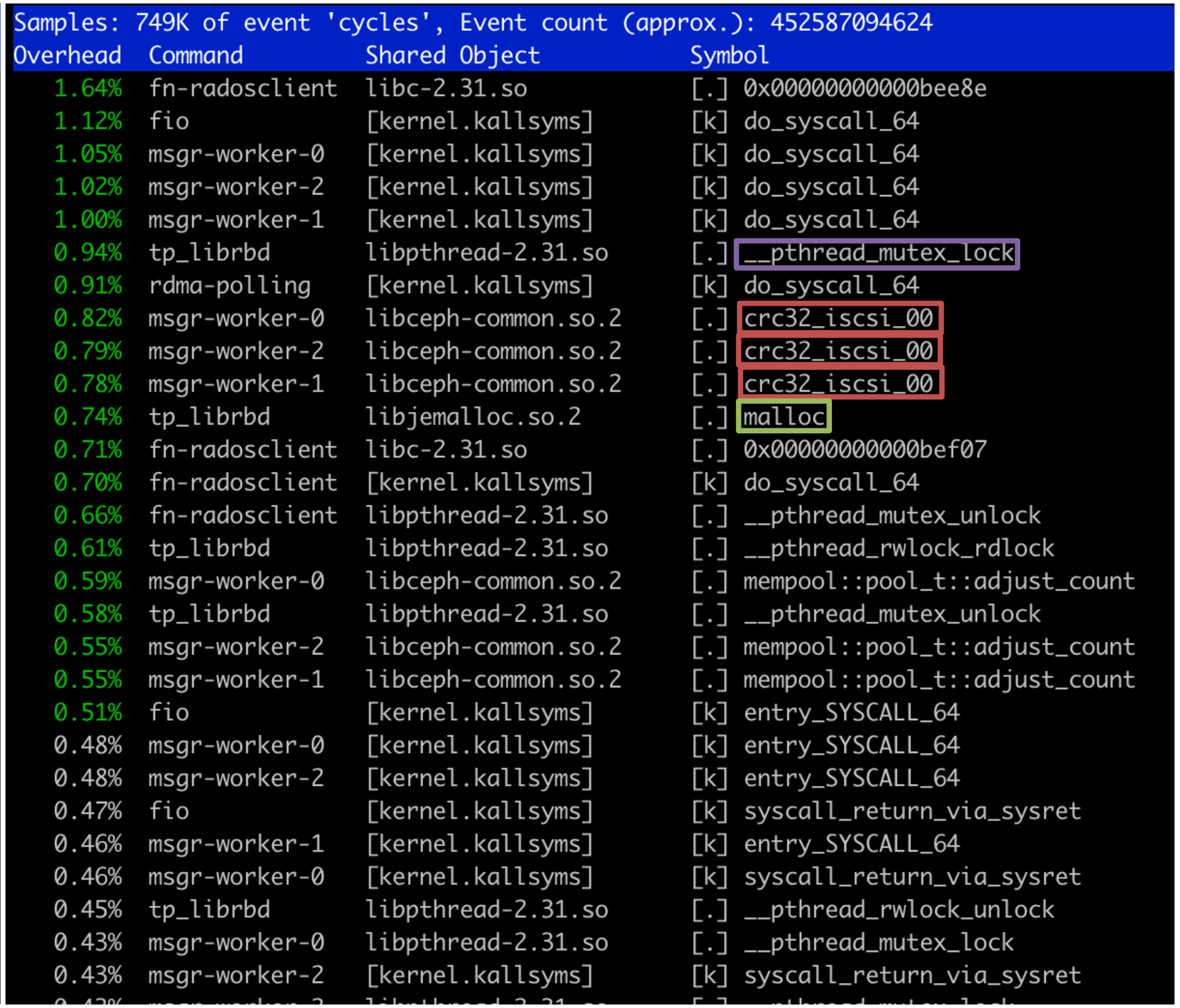
RDMA

With RDMA enabled: **110K IOPS**



```
Samples: 704K of event 'cycles', Event count (approx.): 470508792807
Overhead  Command       Shared Object         Symbol
   4.49%  rdma-polling  libpthread-2.31.so    [.] pthread_spin_lock
   3.06%  rdma-polling  libceph-common.so.2   [.] Cycles::to_nanoseconds
   1.61%  fn-radosclient libc-2.31.so         [.] 0x00000000000bee8e
   1.43%  rdma-polling  libceph-common.so.2   [.] RDMADispatcher::polling
   1.12%  rdma-polling  libceph-common.so.2   [.] Infiniband::CompletionQueue::poll_cq
   0.99%  msgr-worker-0 [kernel.kallsyms]     [k] do_syscall_64
   0.93%  msgr-worker-2 [kernel.kallsyms]     [k] do_syscall_64
   0.93%  fio           [kernel.kallsyms]     [k] do_syscall_64
   0.91%  msgr-worker-1 [kernel.kallsyms]     [k] do_syscall_64
   0.83%  rdma-polling  libmlx5.so.1.12.28.0  [.] 0x00000000000198bb
   0.75%  tp_librbd     libpthread-2.31.so    [.] __pthread_mutex_lock
   0.71%  msgr-worker-0 libceph-common.so.2   [.] crc32_iscsi_00
   0.69%  rdma-polling  [kernel.kallsyms]     [k] try_to_wake_up
   0.67%  msgr-worker-2 libceph-common.so.2   [.] crc32_iscsi_00
   0.67%  fn-radosclient [kernel.kallsyms]    [k] do_syscall_64
   0.64%  msgr-worker-1 libceph-common.so.2   [.] crc32_iscsi_00
   0.62%  tp_librbd     libjemalloc.so.2      [.] malloc
   0.56%  rdma-polling  libceph-common.so.2   [.] Cycles::to_microseconds
   0.54%  fn-radosclient libpthread-2.31.so   [.] __pthread_mutex_unlock
   0.50%  tp_librbd     libpthread-2.31.so    [.] __pthread_mutex_unlock
   0.50%  rdma-polling  [kernel.kallsyms]     [k] do_syscall_64
   0.48%  msgr-worker-0 libceph-common.so.2   [.] mempool::pool_t::adjust_count
   0.46%  tp_librbd     libpthread-2.31.so    [.] __pthread_rwlock_rdlock
   0.46%  msgr-worker-2 libceph-common.so.2   [.] mempool::pool_t::adjust_count
   0.46%  msgr-worker-0 [kernel.kallsyms]     [k] entry_SYSCALL_64
   0.44%  msgr-worker-1 libceph-common.so.2   [.] mempool::pool_t::adjust_count
   0.44%  fio           [kernel.kallsyms]     [k] entry_SYSCALL_64
   0.42%  msgr-worker-1 [kernel.kallsyms]     [k] entry_SYSCALL_64
```

RDMA + ms_async_rdma_polling_us=0: **120K IOPS**



```
Samples: 749K of event 'cycles', Event count (approx.): 452587094624
Overhead  Command        Shared Object        Symbol
   1.64%  fn-radosclient libc-2.31.so         [.] 0x00000000000bee8e
   1.12%  fio            [kernel.kallsyms]    [k] do_syscall_64
   1.05%  msgr-worker-0  [kernel.kallsyms]    [k] do_syscall_64
   1.02%  msgr-worker-2  [kernel.kallsyms]    [k] do_syscall_64
   1.00%  msgr-worker-1  [kernel.kallsyms]    [k] do_syscall_64
   0.94%  tp_librbd      libpthread-2.31.so   [.] __pthread_mutex_lock
   0.91%  rdma-polling   [kernel.kallsyms]    [k] do_syscall_64
   0.82%  msgr-worker-0  libceph-common.so.2  [.] crc32_iscsi_00
   0.79%  msgr-worker-2  libceph-common.so.2  [.] crc32_iscsi_00
   0.78%  msgr-worker-1  libceph-common.so.2  [.] crc32_iscsi_00
   0.74%  tp_librbd      libjemalloc.so.2     [.] malloc
   0.71%  fn-radosclient libc-2.31.so         [.] 0x00000000000bef07
   0.70%  fn-radosclient [kernel.kallsyms]    [k] do_syscall_64
   0.66%  fn-radosclient libpthread-2.31.so   [.] __pthread_mutex_unlock
   0.61%  tp_librbd      libpthread-2.31.so   [.] __pthread_rwlock_rdlock
   0.59%  msgr-worker-0  libceph-common.so.2  [.] mempool::pool_t::adjust_count
   0.58%  tp_librbd      libpthread-2.31.so   [.] __pthread_mutex_unlock
   0.55%  msgr-worker-2  libceph-common.so.2  [.] mempool::pool_t::adjust_count
   0.55%  msgr-worker-1  libceph-common.so.2  [.] mempool::pool_t::adjust_count
   0.51%  fio            [kernel.kallsyms]    [k] entry_SYSCALL_64
   0.48%  msgr-worker-0  [kernel.kallsyms]    [k] entry_SYSCALL_64
   0.48%  msgr-worker-2  [kernel.kallsyms]    [k] entry_SYSCALL_64
   0.47%  fio            [kernel.kallsyms]    [k] syscall_return_via_sysret
   0.46%  msgr-worker-1  [kernel.kallsyms]    [k] entry_SYSCALL_64
   0.46%  msgr-worker-0  [kernel.kallsyms]    [k] syscall_return_via_sysret
   0.45%  tp_librbd      libpthread-2.31.so   [.] __pthread_rwlock_unlock
   0.43%  msgr-worker-0  libpthread-2.31.so   [.] __pthread_mutex_lock
   0.43%  msgr-worker-2  [kernel.kallsyms]    [k] syscall_return_via_sysret
```

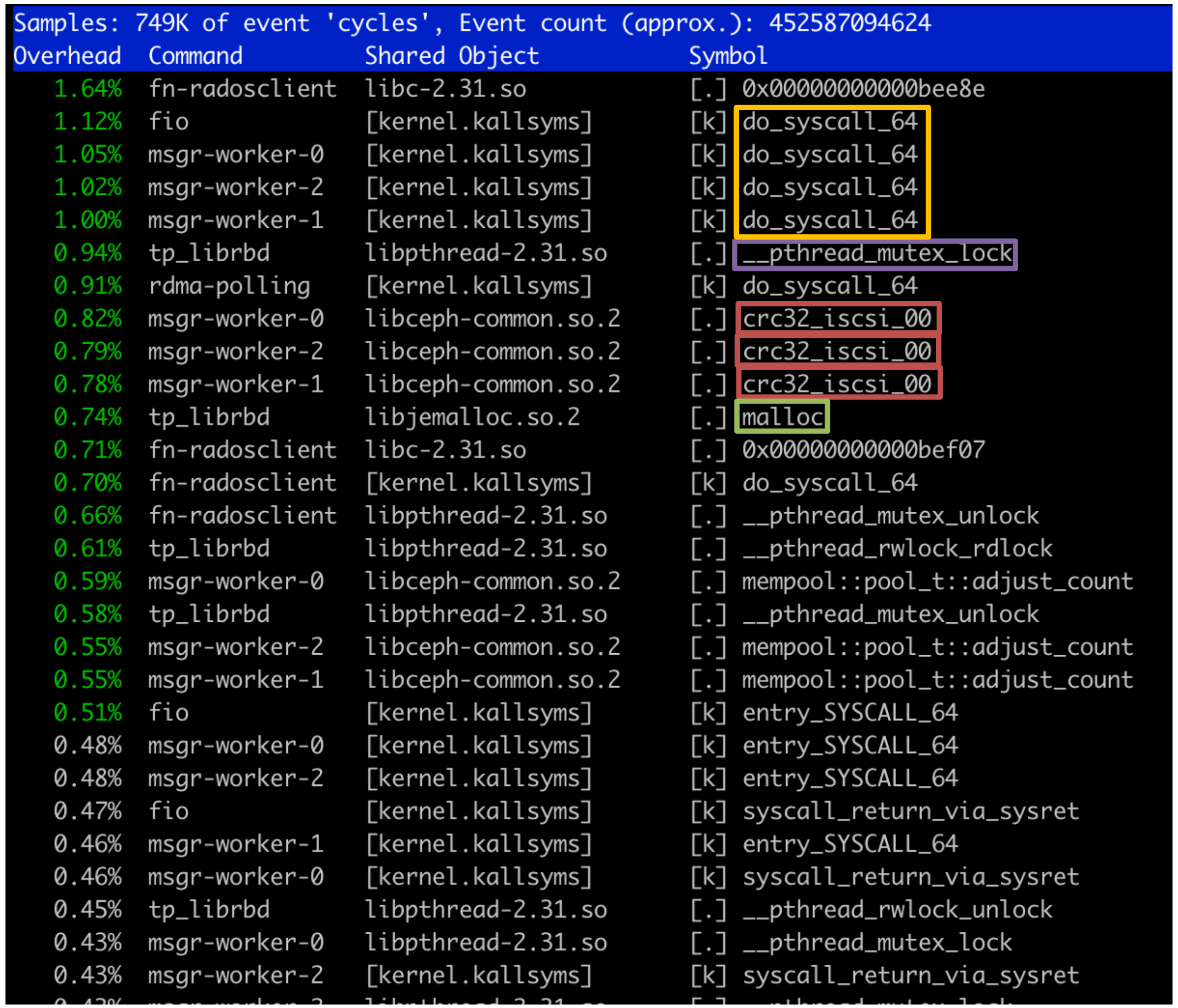
RDMA CONTINUED

RDMA + ms_async_rdma_polling_us=0

120K IOPS

Top symbols system calls

- EventCenter::process_events uses EventEpoll driver, which uses epoll to dispatch events => read, write syscalls
- Locking to protect shared datastructures between msgr workers => futex
- RDMADispatcher / RDMAConnectedSocketImpl uses eventfd for dispatching => read/write syscalls



```
Samples: 749K of event 'cycles', Event count (approx.): 452587094624
Overhead  Command          Shared Object           Symbol
   1.64%  fn-radosclient   libc-2.31.so            [.] 0x00000000000bee8e
   1.12%  fio              [kernel.kallsyms]       [k] do_syscall_64
   1.05%  msgr-worker-0    [kernel.kallsyms]       [k] do_syscall_64
   1.02%  msgr-worker-2    [kernel.kallsyms]       [k] do_syscall_64
   1.00%  msgr-worker-1    [kernel.kallsyms]       [k] do_syscall_64
   0.94%  tp_librbd        libpthread-2.31.so      [.] __pthread_mutex_lock
   0.91%  rdma-polling     [kernel.kallsyms]       [k] do_syscall_64
   0.82%  msgr-worker-0    libceph-common.so.2     [.] crc32_iscsi_00
   0.79%  msgr-worker-2    libceph-common.so.2     [.] crc32_iscsi_00
   0.78%  msgr-worker-1    libceph-common.so.2     [.] crc32_iscsi_00
   0.74%  tp_librbd        libjemalloc.so.2        [.] malloc
   0.71%  fn-radosclient   libc-2.31.so            [.] 0x00000000000bef07
   0.70%  fn-radosclient   [kernel.kallsyms]       [k] do_syscall_64
   0.66%  fn-radosclient   libpthread-2.31.so      [.] __pthread_mutex_unlock
   0.61%  tp_librbd        libpthread-2.31.so      [.] __pthread_rwlock_rdlock
   0.59%  msgr-worker-0    libceph-common.so.2     [.] mempool::pool_t::adjust_count
   0.58%  tp_librbd        libpthread-2.31.so      [.] __pthread_mutex_unlock
   0.55%  msgr-worker-2    libceph-common.so.2     [.] mempool::pool_t::adjust_count
   0.55%  msgr-worker-1    libceph-common.so.2     [.] mempool::pool_t::adjust_count
   0.51%  fio              [kernel.kallsyms]       [k] entry_SYSCALL_64
   0.48%  msgr-worker-0    [kernel.kallsyms]       [k] entry_SYSCALL_64
   0.48%  msgr-worker-2    [kernel.kallsyms]       [k] entry_SYSCALL_64
   0.47%  fio              [kernel.kallsyms]       [k] syscall_return_via_sysret
   0.46%  msgr-worker-1    [kernel.kallsyms]       [k] entry_SYSCALL_64
   0.46%  msgr-worker-0    [kernel.kallsyms]       [k] syscall_return_via_sysret
   0.45%  tp_librbd        libpthread-2.31.so      [.] __pthread_rwlock_unlock
   0.43%  msgr-worker-0    libpthread-2.31.so      [.] __pthread_mutex_lock
   0.43%  msgr-worker-2    [kernel.kallsyms]       [k] syscall_return_via_sysret
   0.42%  msgr-worker-2    libpthread-2.31.so      [.] __pthread_mutex_lock
```

RDMA VS TCP

CPU utilization

- RDMA expected to be lower than TCP
- 1 fio process
  - 3 messenger threads
  - 1 rbd operations thread
- *<u>Total CPU utilization</u>*
  - ***RDMA:*** 15% of 32 logical CPUs => **4.8 CPUs**
  - ***TCP:*** 7% of 32 logical CPUs => **2.2 CPUs**
- RDMA messenger threads are polling =>
  polling threshold can be changed but hurts performance

Max machine IOPS

- ***RMDA:*** 8 processes/volumes => ***265K IOPS***
- ***TCP:*** 8 processes/volumes => ***270K IOPS***

LIBRBD RDMA PROBLEMS AND SOLUTIONS

Problems

- Ceph network abstraction: streaming (socket)
  - Ceph RDMA implements socket API
  - Dispatching with eventfd => context switches
  - Adds unnecessary copies
- General event processing via ePoll => context switches

Solutions

- Implement RDMA networking at Ceph message layer
- Reduce number of threads and dispatching
- Shared memory instead of kernel for events where possible (dynamic)

SPDK AND LIBRBD

- Pinned reactor threads for event processing in SPDK
- Librbd threads inherit thread affinity of reactor threads
  => all 4 librbd threads run on same core!
- Unpinning the threads increases performance >5x
  - 19.8K IOPS **=> 107K IOPS**
- Every reactor thread creates new librados io context => 4 new threads per image per reactor thread