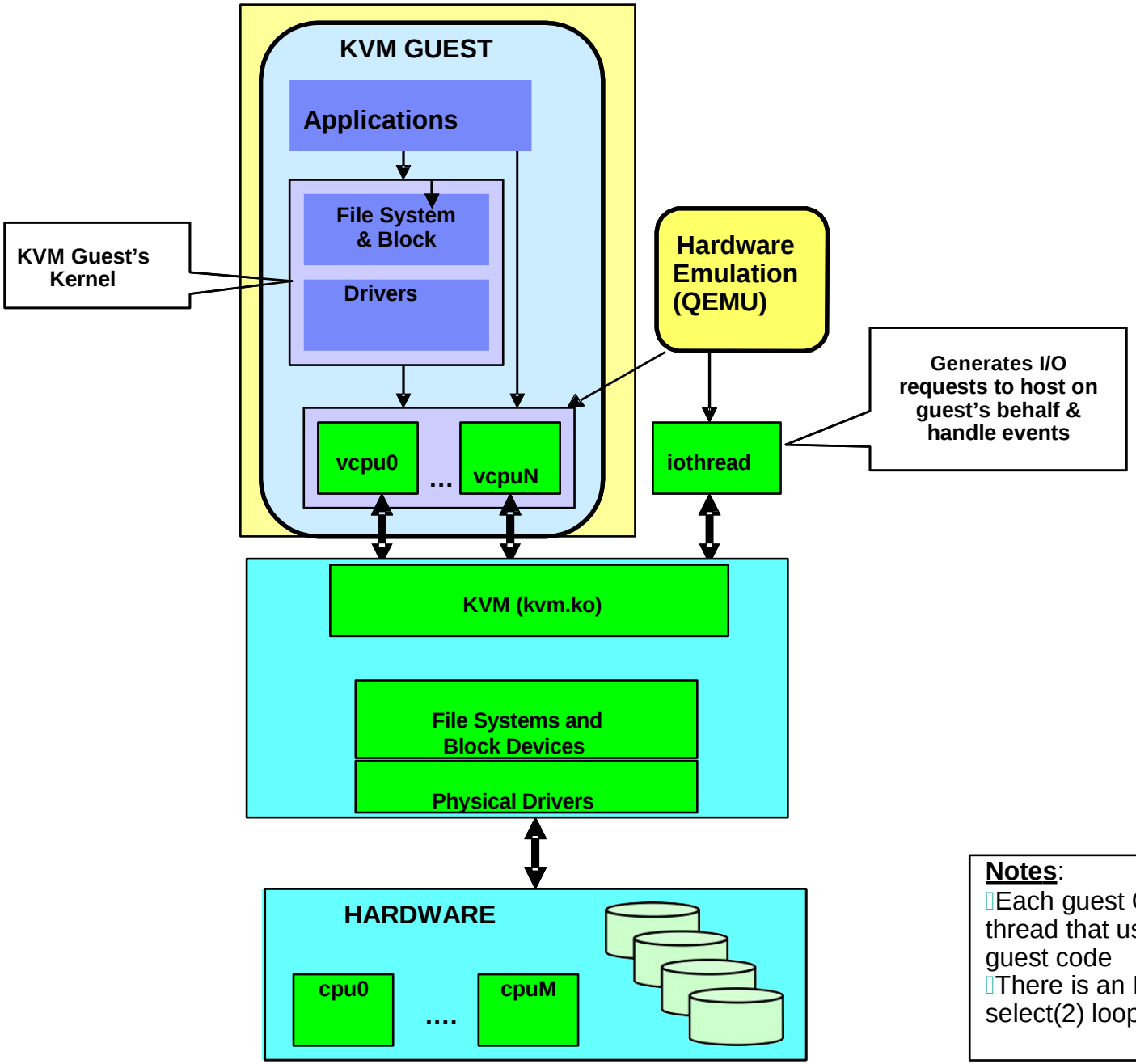
QEMU Disk IO
Which performs Better:
Native or threads?

Pradeep Kumar Surisetty
Red Hat, Inc.
devconf.cz, February 2016

Outline

- KVM IO Architecture
- Storage transport choices in KVM
- Virtio-blk Storage Configurations
- Performance Benchmark tools
- Challenges
- Performance Results with Native & Threads
- Limitations
- Future Work

KVM I/O Architecture



Notes:

- Each guest CPU has a dedicated vcpu thread that uses kvm.ko module to execute guest code
- There is an I/O thread that runs a select(2) loop to handle events

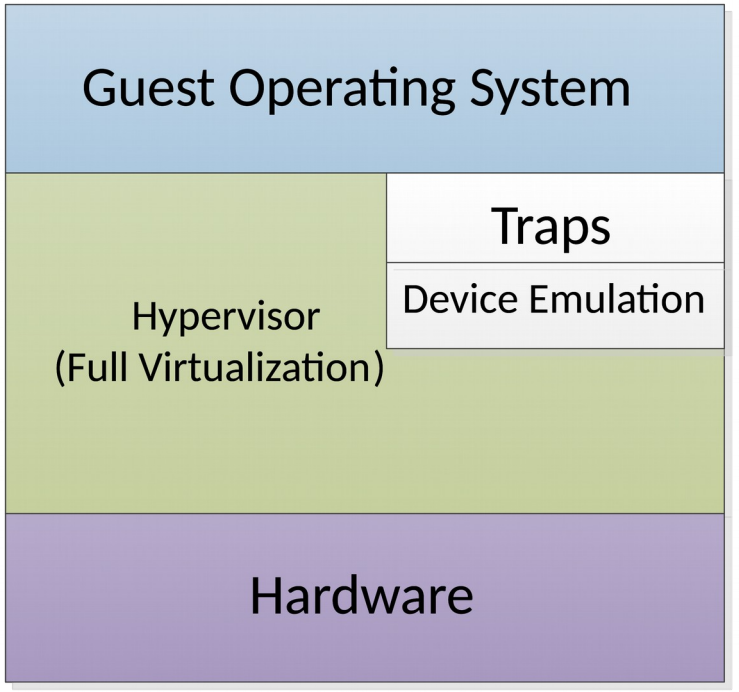
Storage transport choices in KVM

- **Full virtualization : IDE, SATA, SCSI**
  - Good guest compatibility
  - Lots of trap-and-emulate, bad performance
- **Para virtualization: virtio-blk, virtio-scsi**
  - Efficient guest ↔ host communication through virtio ring buffer (virtqueue)
  - Good performance
  - Provide more virtualization friendly interface, higher performance.
  - In AIO case, io_submit() is under the global mutex

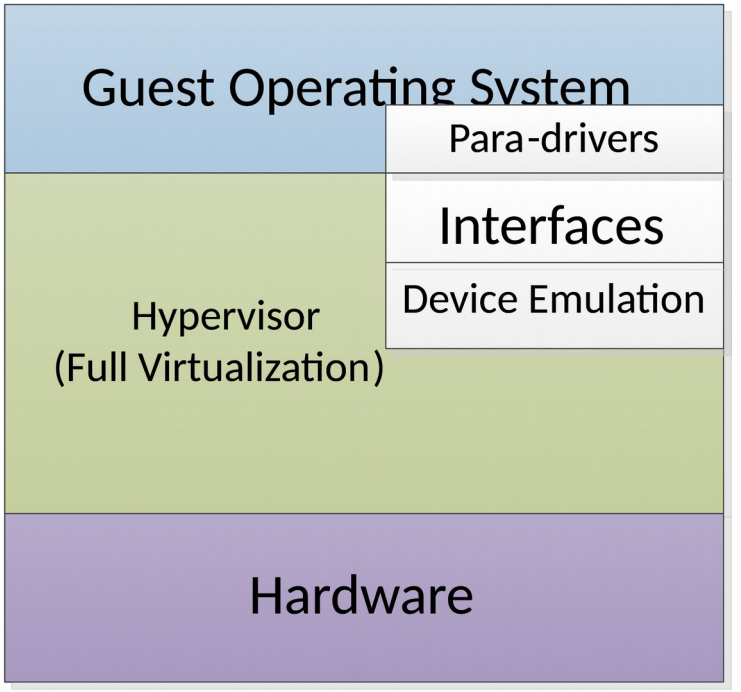
Storage transport choices in KVM

- **Device assignment (Passthrough)**
  - Pass hardware to guest, high-end usage, high performance
  - Limited Number of PCI Devices
  - Hard for Live Migration

Storage transport choices in KVM

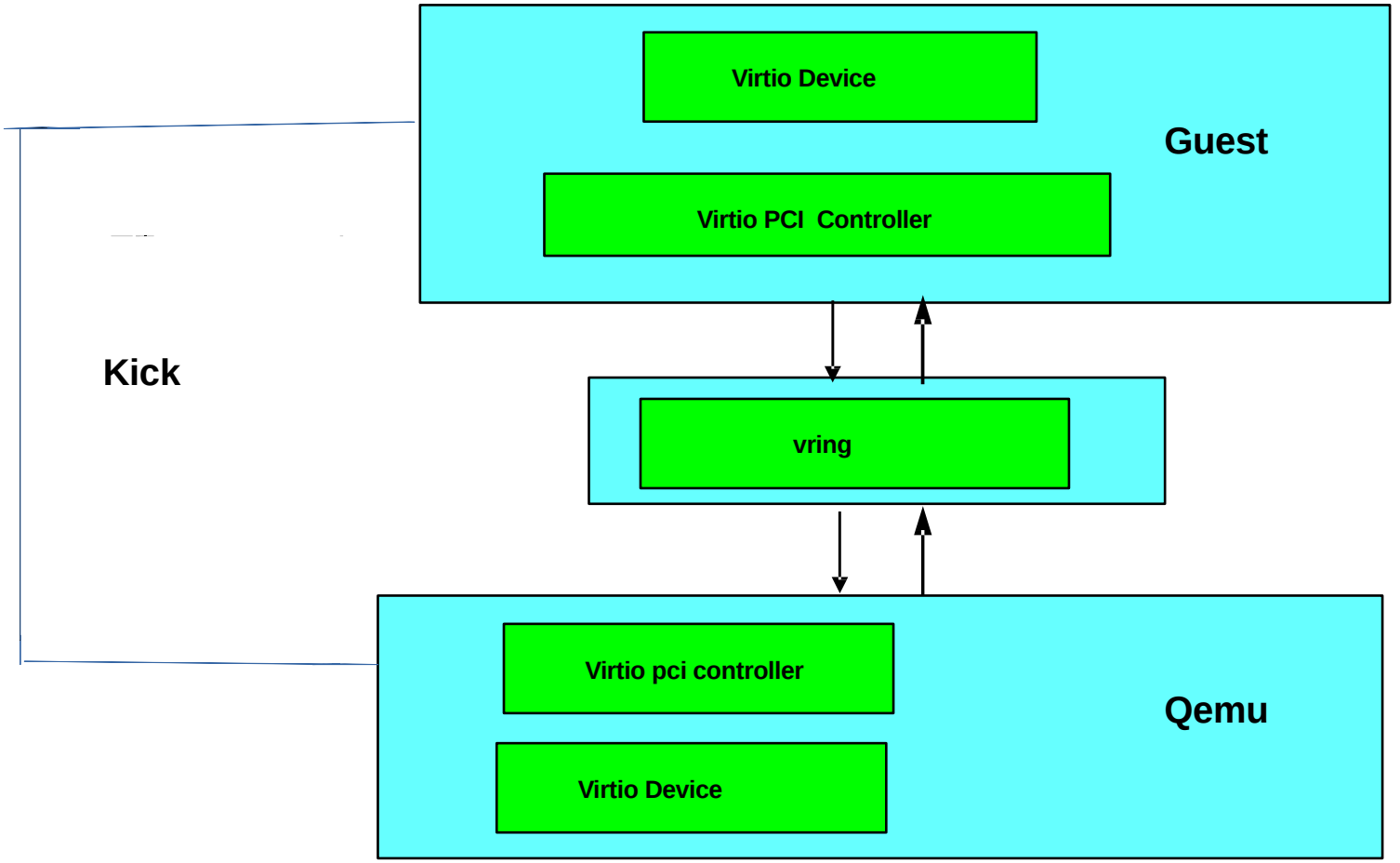


Full virtualization



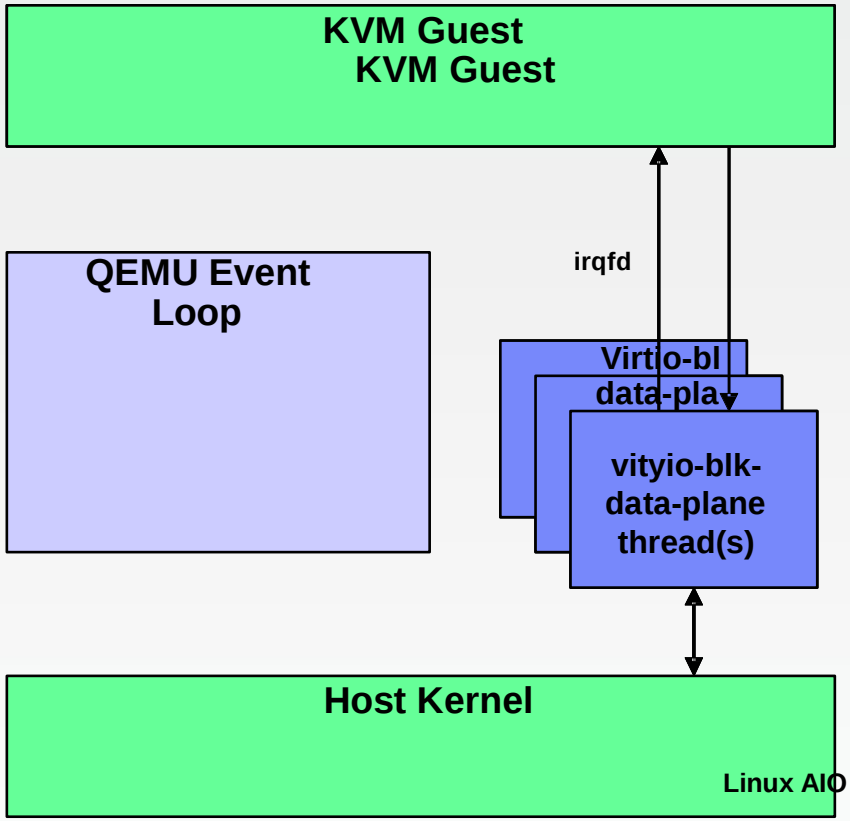
Para-virtualization

Ring buffer with para virtualization

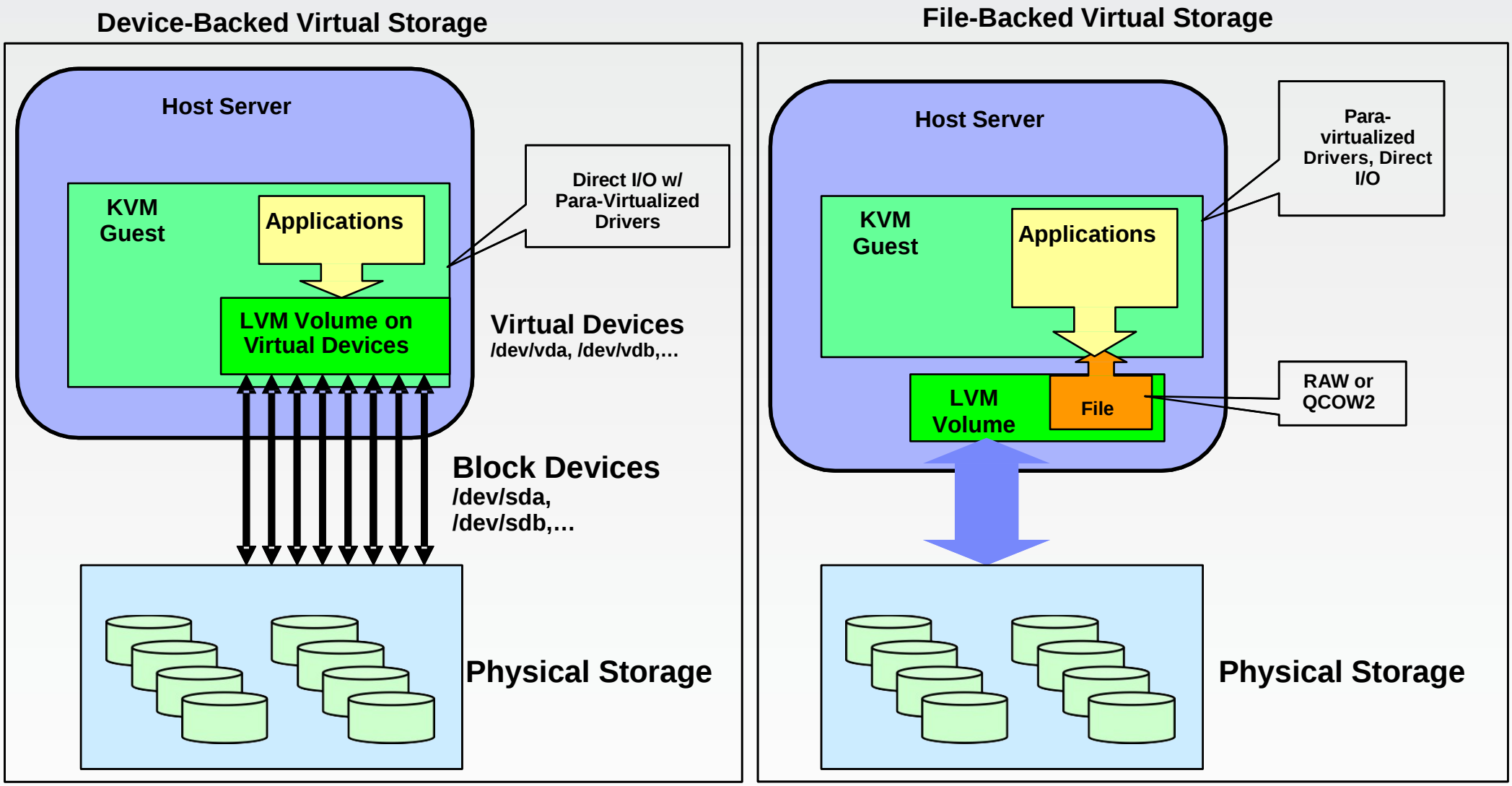


Virtio-blk-data-plane:

- Accelerated data path for para-virtualized block I/O driver
- Threads are defined by -object othread,iothread=<id> and the user can set up arbitrary device->iothread mappings (multiple devices can share an iothread)
- *No need to acquire **big QEMU lock***



Virtio-blk Storage Configurations



Openstack:
Libvirt: AIO mode for disk devices

1) Asynchronous IO (AIO=Native)

Using io_submit calls

2) Synchronous (AIO=Threads)

pread64, pwrite64 calls

Default Choice in Openstack is aio=threads*

Ref: https://specs.openstack.org/openstack/novaspecs/specs/mitaka/approved/libvirt-aio-mode.html

*** Before solving this problem**

Example XML

- </disk>

  <disk type='file' device='disk'>

    <driver name='qemu' type='qcow2' cache='none' io='**native**'/>

    <source file='/home/psuriset/xfs/vm2-native-ssd.qcow2'/>

    <target dev='vdb' bus='virtio'/>

    <address type='pci' domain='0x0000' bus='0x00' slot='0x06' function='0x0'/>

  </disk>

- <disk type='file' device='disk'>

    <driver name='qemu' type='qcow2' cache='none' io='**threads**'/>

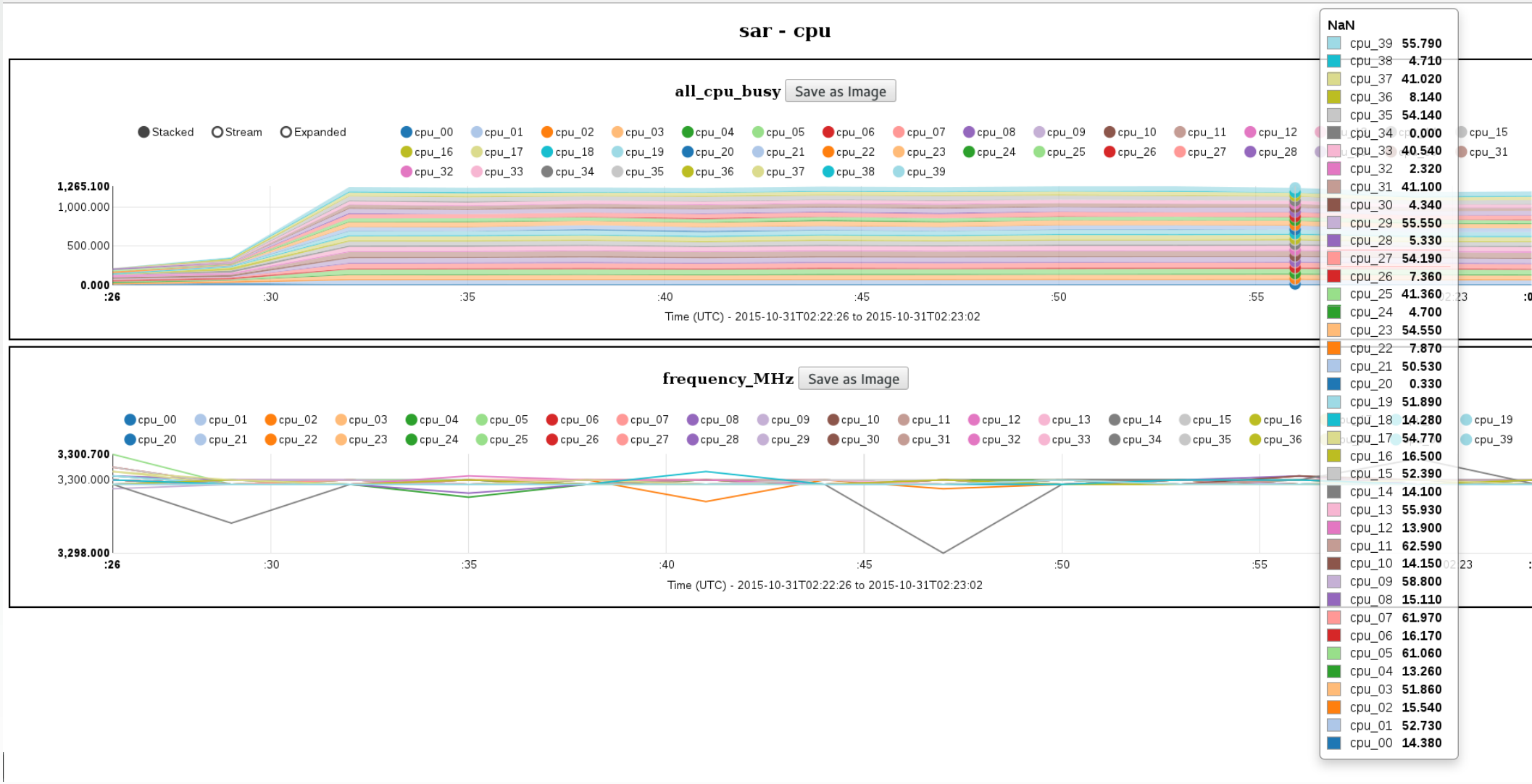
    <source file='/home/psuriset/xfs/vm2-threads-ssd.qcow2'/>

    <target dev='vdc' bus='virtio'/>

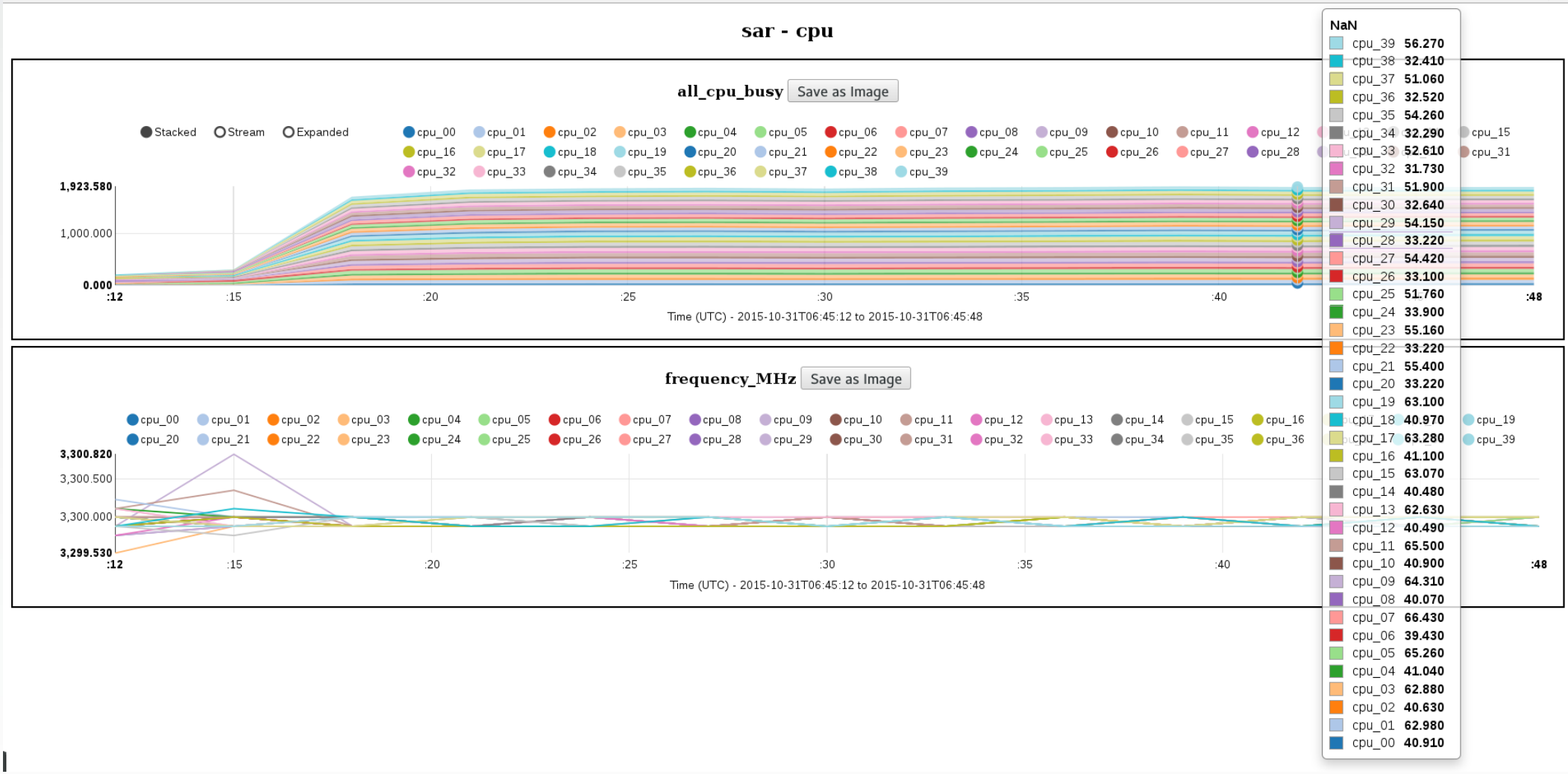
    <address type='pci' domain='0x0000' bus='0x00' slot='0x07' function='0x0'/>

  </disk>

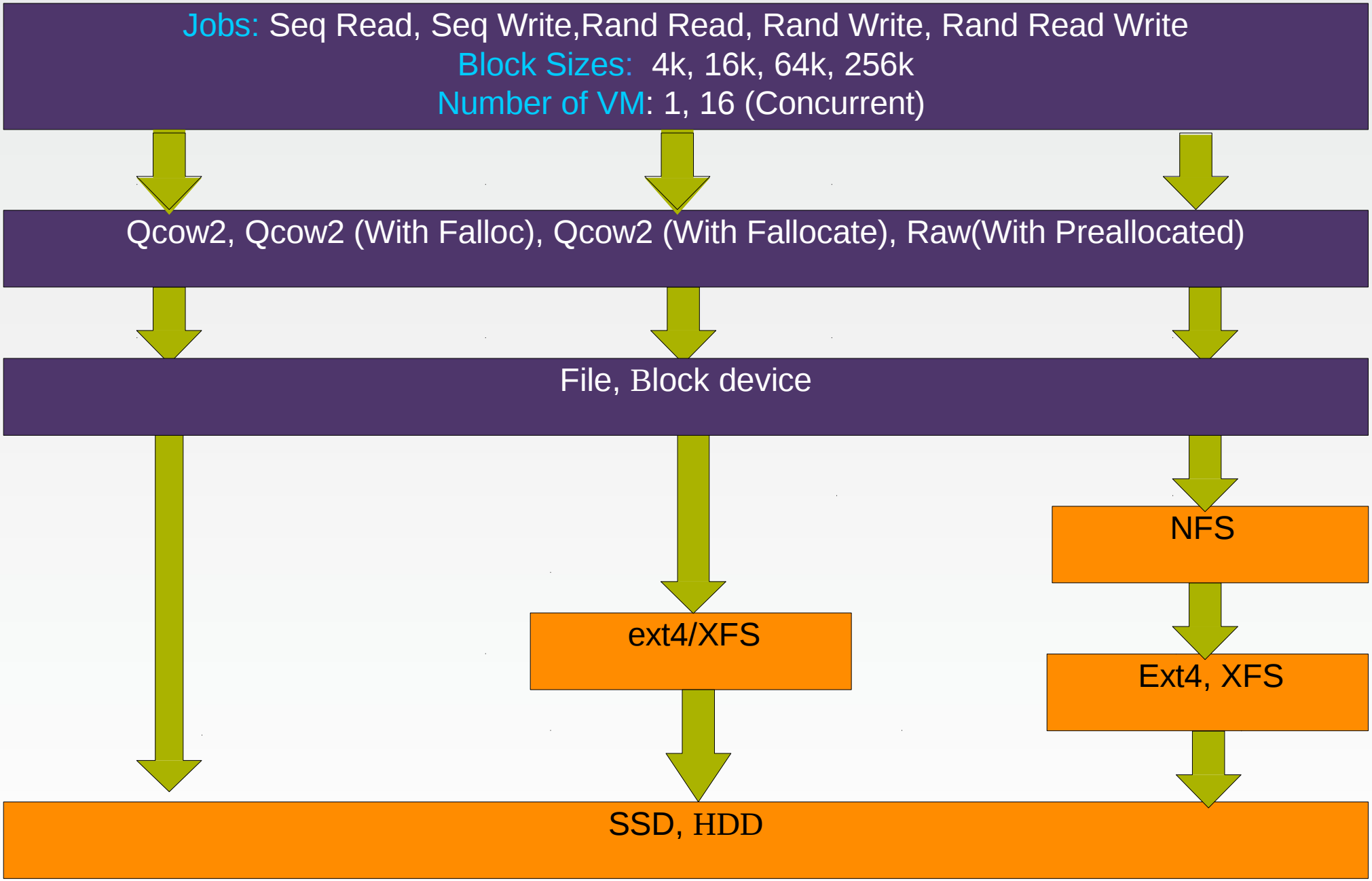
CPU Usage with aio=Native



CPU Usage with aio=Threads



Multiple layers Evaluated with virtio-blk



Test Environment

Hardware

- **2 x Intel(R) Xeon(R) CPU E5-2690 v2 @ 3.00GHz**
- **256 GiB memory @1866MHz**
- **1 x 1 TB NVMe PCI SSD**
- **1 x 500 GB HDD**

Software

- **Host: RHEL 7.2 :3.10.0-327**
- **Qemu: 2.3.0-31 + AIO Merge Patch**
- **VM: RHEL 7.2**

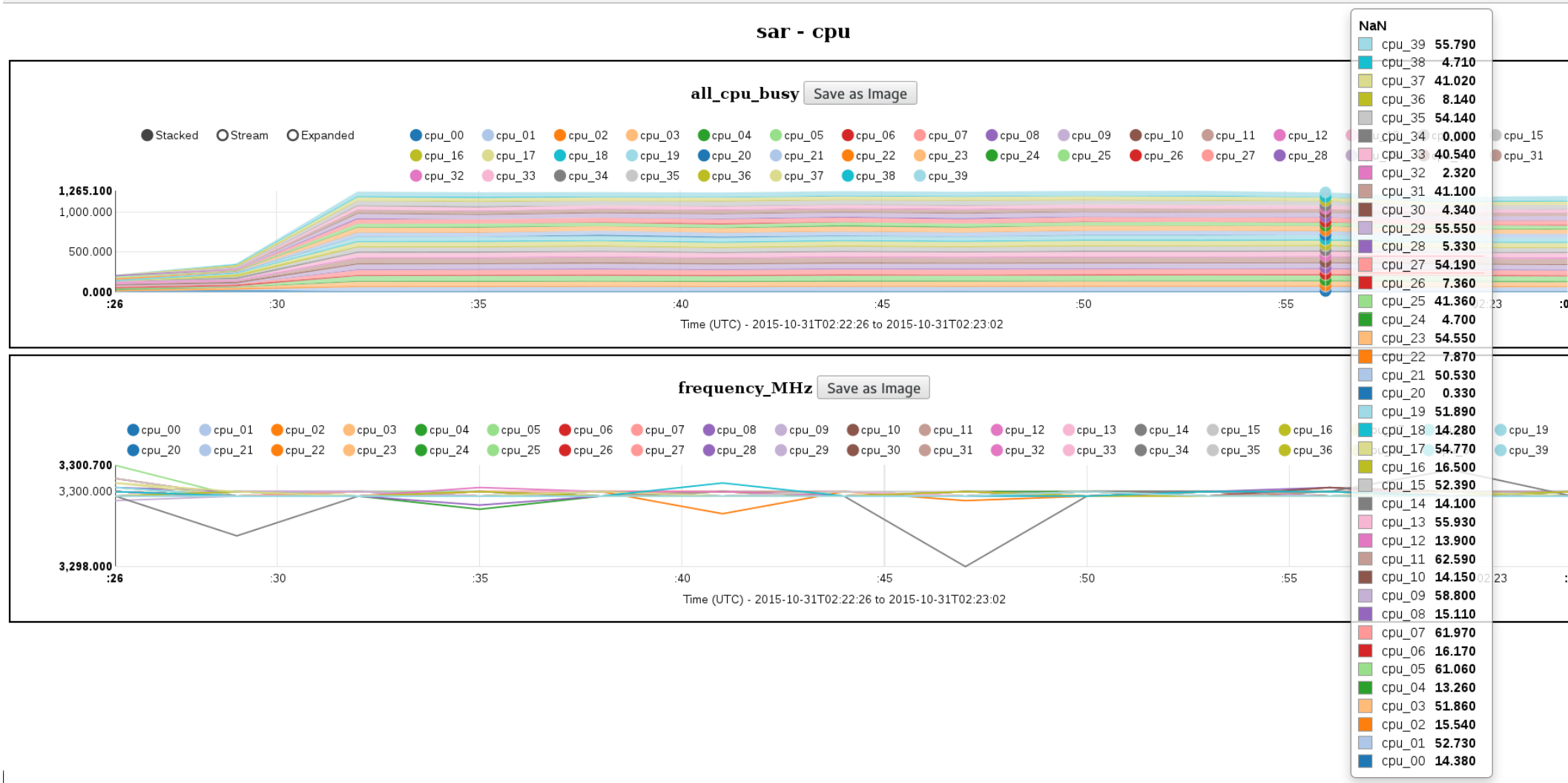
Tools
What is Pbench?

pbench (perf bench) aims to:

- Provide easy access to benchmarking & performance tools on Linux systems
- Standardize the collection of telemetry and configuration Information
- Automate benchmark execution
- Output effective visualization for analysis allow for ingestion into elastic search

Pbench Continued...

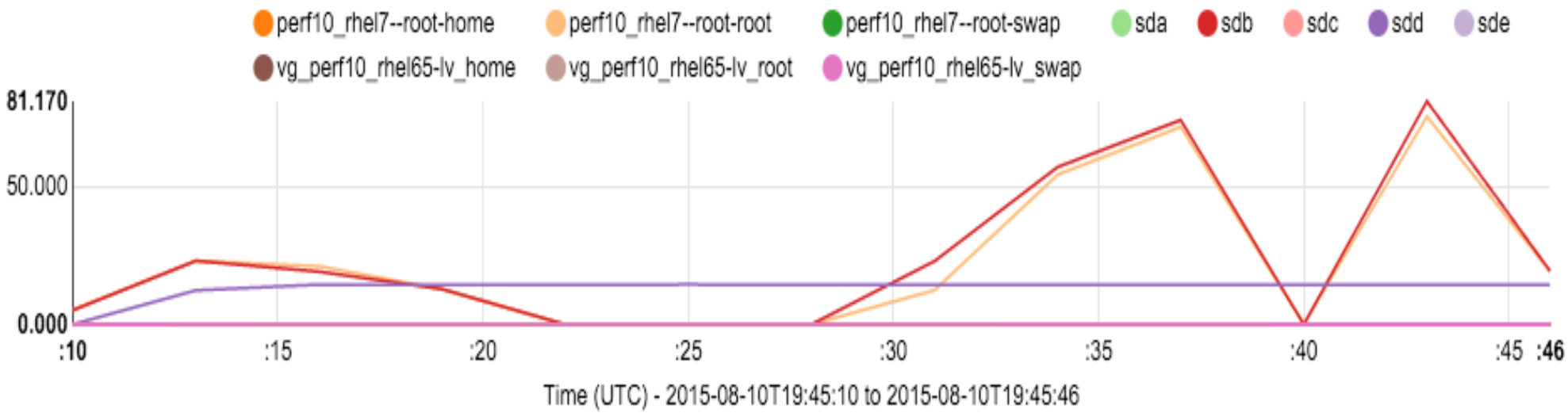
Tool visualization:



Pbench Continued ..

tool visualization:

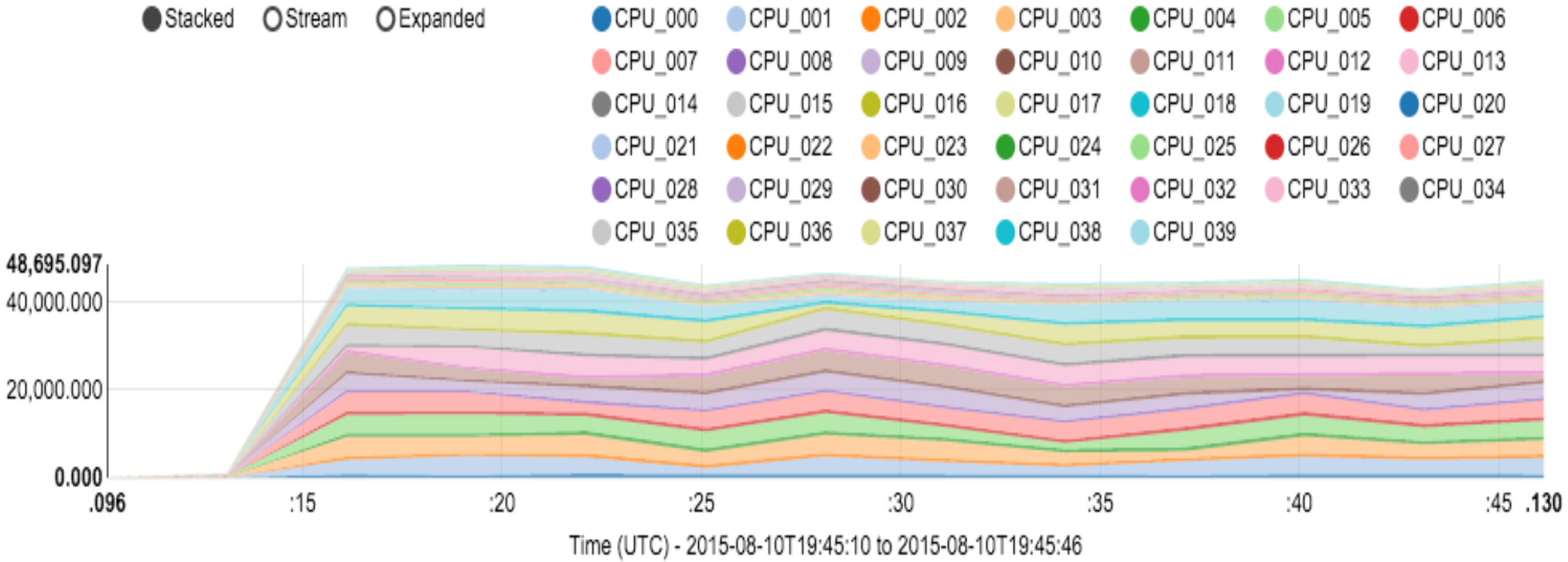
iostat tool, disk request size:



Pbench Continued ..

tool visualization:

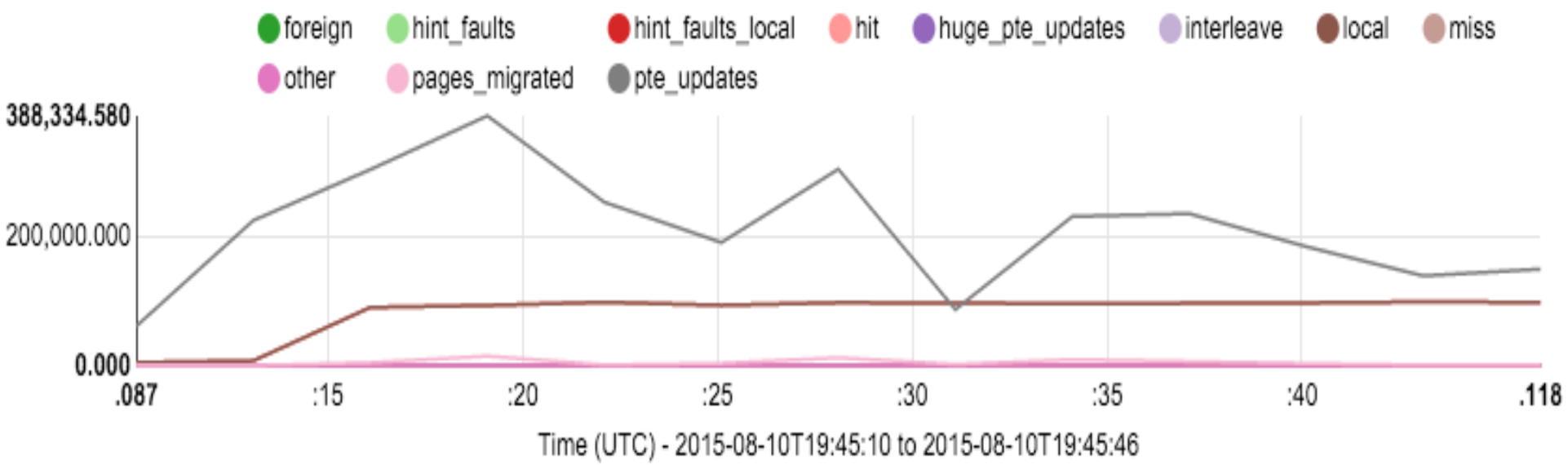
proc-interrupts tool, function call interrupts/sec:



Pbench Continued ..

tool visualization:

proc-vmstat tool, numa stats: entries in /proc/vmstat which begin with “numa_” (delta/sec)



Pbench Continued ..

pbench benchmarks

example: fio benchmark

pbench_fio --config=baremetal-hdd
runs a default set of iterations:
[read,rand-read]*[4KB, 8KB….64KB]
takes 5 samples per iteration and compute avg, stddev
handles start/stop/post-process of tools for each iteration
other fio options:
--targets=<devices or files>
--ioengine=[sync, libaio, others]
--test-types=[read,randread,write,randwrite,randrw]
--block-sizes=[<int>,[<int>]] (in KB)

FIO: Flexible IO Tester

- **IO type**

  Defines the io pattern issued to the file(s). We may only be reading sequentially from this file(s), or we may be writing randomly. Or even mixing reads and writes, sequentially or Randomly

- **Block size**

  In how large chunks are we issuing io? This may be a single value, or it may describe a range of block sizes.

- **IO size**

  How much data are we going to be reading/writing

- **IO Engine**

  How do we issue io? We could be memory mapping the file, we could be using regular read/write, we could be using splice, async io, syslet, or even SG (SCSI generic sg)

- **IO depth**

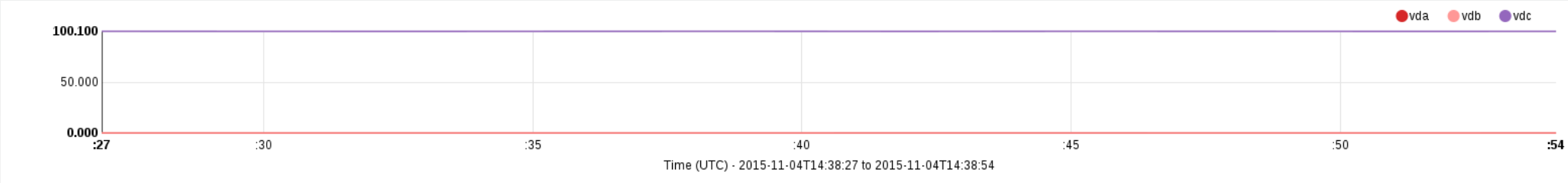
  If the io engine is async, how large a queuing depth do we want to maintain?

- **IO Type**

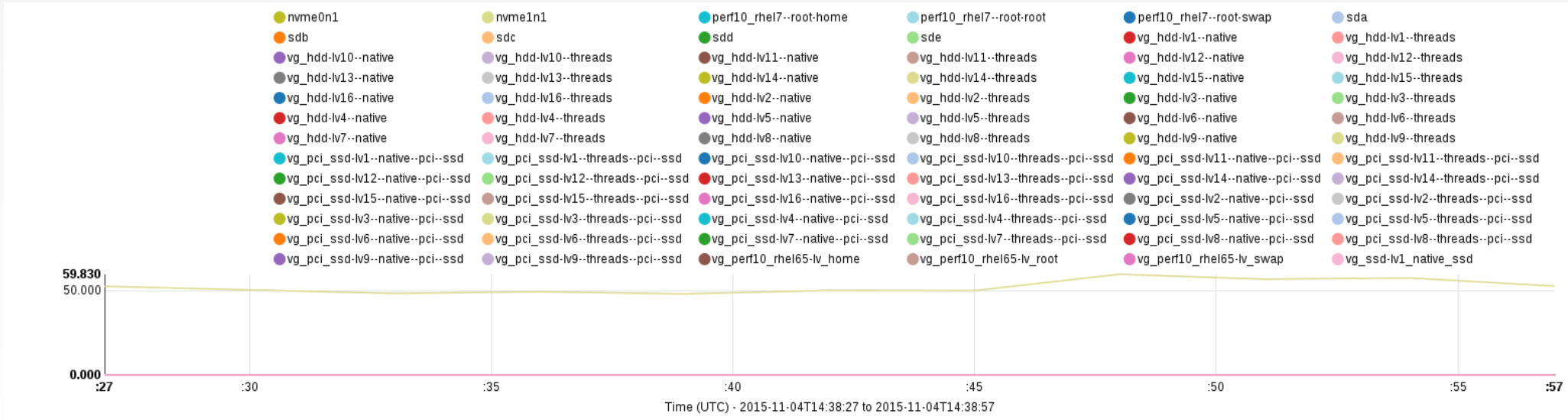
  Should we be doing buffered io, or direct/raw io?

Guest & Host iostat during 4k seq read with aio=native

Guest



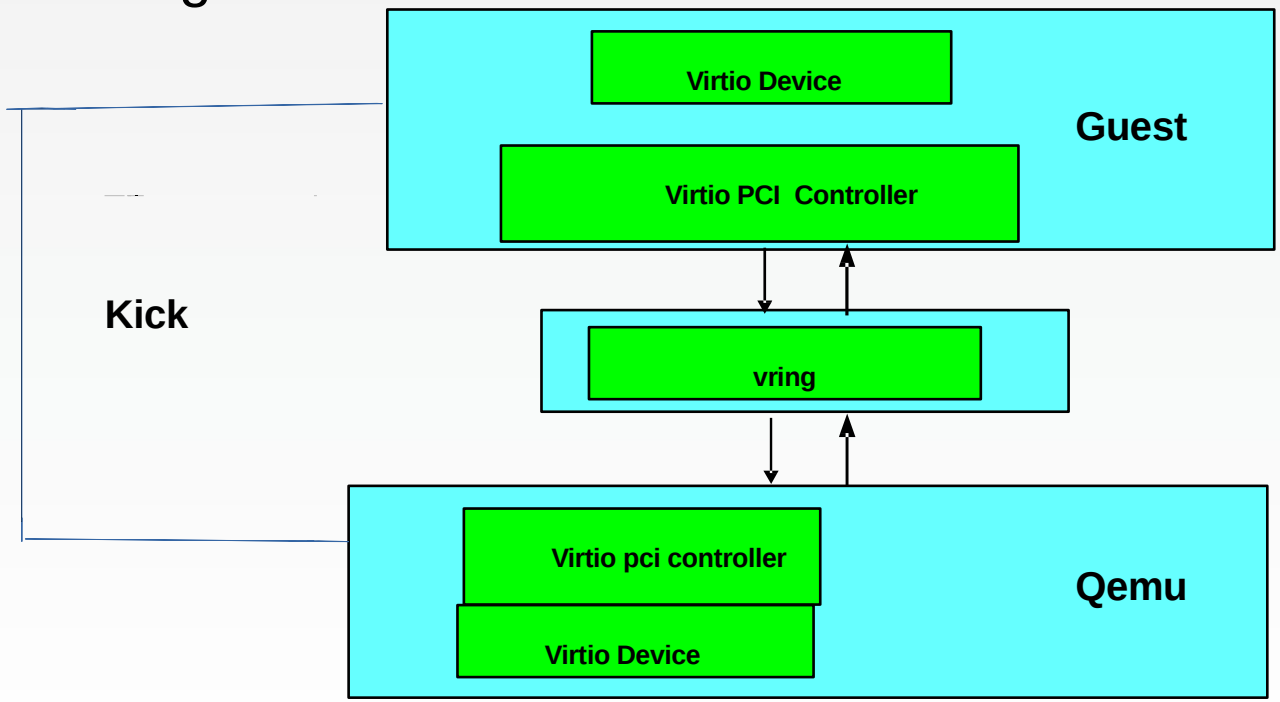
Host



AIO Native

- Aio=native uses Linux AIO io_submit(2) for read and write requests and Request completion is signaled using eventfd.
- Virtqueue kicks are handled in the iothread. When the guest writes to the virtqueue kick hardware register the kvm.ko module signals the ioeventfd which the main loop thread is monitoring.
- Requests are collected from the virtqueue and submitted (after write request merging) either via aio=threads or aio=native.
- Request completion callbacks are invoked in the main loop thread and an interrupt is injected into the guest.

.



Challenges for Read with aio=native

.

- virtio-blk does ***not* merge** read requests in qemu-kvm. It only merges write requests.
- QEMU submits each 4 KB request through a separate io_submit() call.
- Qemu would submit only 1 request at a time though Multiple requests to process
- Batching method was implemented for both virtio-scsi and virtio-blk-data-plane disk

Batch Submission

What is I/O batch submission

- Handle more requests in one single system call(io_submit), so calling number of the syscall of io_submit can be decrease a lot

Abstracting with generic interfaces

- bdrv_io_plug( ) / bdrv_io_unplug( )
- merged in fc73548e444ae3239f6cef44a5200b5d2c3e85d1

  (virtio-blk: submit I/O as a batch)

Performance Comparison Graphs

Results

Test Specifications	Single VM Results	Multiple VM (16) Results
Disk: SSD **FS:** None (used LVM) **Image**: raw **Preallocated**: yes	aio=threads has better performance with LVM. 4K read,randread performance is 10-15% higher. 4K write is 26% higher.	Native & threads perform equally in most cases but native does better in few cases.
Disk: SSD **FS**: EXT4 **Image:** raw **Preallocated**: yes	Native performs well with randwrite, write, and randread-write. Threads 4K read is 10-15% Higher.. 4K randread is 8% higher.	Both have similar results. 4K seq reads: threads 1% higher.
Disk: SSD **FS:** XFS **Image**: Raw **Preallocated**: yes	aio=threads has better performance	Native & threads perform equally in most cases but native does better in few cases. Threads better in seq writes
Disk: SSD **FS:** EXT4 **Image**: raw **Preallocated**: yes **NFS** : yes	Native performs well with randwrite, write and randread-write. Threads do well with 4K/16K read, randread by 12% higher.	Native & threads perform equally in most cases but native does better in few cases.

Test Specifications	Single VM Results	Multiple VM (16) Results
Disk: SSD **FS:** XFS **Image:** raw **Preallocated**: yes **NFS** : yes	Native performs well with all tests except read & randread tests where threads perform better.	Native performs well with all tests.
Disk: SSD **FS:** EXT4 **Image:** qcow2 **Preallocated**: no	Native does well with all tests. Threads outperform native <10% for read and randread.	Native is better than threads in most cases. Seq reads are 10-15% higher with native.
Disk: SSD **FS:** XFS **Image:** qcow2 **Preallocated**: no	Native performs well with all tests except seq read which is 6% higher	Native performs better than threads except seq write, which is 8% higher
Disk: SSD **FS:** EXT4 **Image:** qcow2 **Preallocated**: with falloc (using qemu-img)	Native is optimal for almost all tests. Threads slightly better (<10%) for seq reads.	Native is optimal with randwrite, write and randread-write. Threads have slightly better performance for read and randread.
Disk: SSD **FS:** XFS **Image:** qcow2 **Preallocate**: with falloc	Native is optimal for write and randread-write. Threads better (<10%) for read and randread.	Native is optimal for all tests. Threads is better for seq writes.
Disk: SSD **FS:** EXT4 **Image:** qcow2	Native performs better for randwrite, write, randread-write. Threads does better for read and randread. 4K,16K read,randread is 12% higher.	Native outperforms threads.

Test Specifications	Single VM Results	Multiple VM (16) Results
Disk: SSD **FS:** XFS **Image:** qcow2 **Preallocated**: with fallocate	Native is optimal for randwrite, write and randread Threads better (<10%) for read	Native optimal for all tests. Threads optimal for randread, and 4K seq write.
Disk: HDD **FS:** No. Used LVM **Image**: raw **Preallocated**: yes	Native outperforms threads in all tests.	Native outperforms threads in all tests.
Disk: HDD **FS:** EXT4 **Image**: raw **Preallocated**: yes	Native outperforms threads in all tests.	Native outperforms threads in all tests.
Disk: HDD **FS:** XFS **Image**: raw **Preallocated**: yes	Native outperforms threads in all tests.	Native is optimal or equal in all tests.
Disk: HDD **FS:** EXT4 **Image**: qcow2 **Preallocated**: no	Native is optimal or equal in all test cases except randread where threads is 30% higher.	Native is optimal except for 4K seq reads.
Disk: HDD **FS:** XFS **Image**: qcow2 **Preallocated**: no	Native is optimal except for seq writes where threads is 30% higher.	Native is optimal except for 4K seq reads.
Disk: HDD **FS:** EXT4 **Image**: qcow2	Native is optimal or equal in all cases except randread where threads is 30% higher.	Native is optimal or equal in all tests except 4K read.

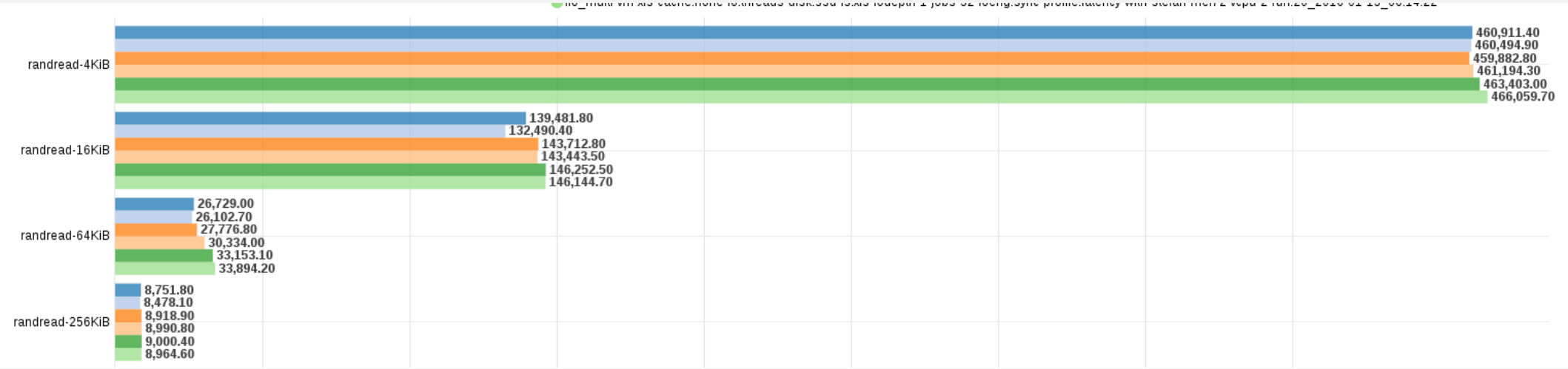
Test Specifications	Single VM Results	Multiple VM (16) Results
Disk: HDD **FS:** XFS **Image**: qcow2 **Preallocated**: with falloc (using qemu-img)	Native is optimal or equal in all cases except seq write where threads is 30% higher.	Native is optimal or equal in all cases except for 4K randread.
Disk: HDD **FS:** EXT4 **Image**: qcow2 **Preallocated**: with fallocate	Native is optimal or equal in all tests.	Native is optimal or equal in all tests except 4K randread where threads is 15% higher.
Disk: HDD **FS:** XFS **Image**: qcow2 **Preallocated**: with fallocate	Native is optimal in all tests except for seq write where threads is 30% higher.	Nativs is better. Threads has slightly better performance(<3-4%), excluding randread where threads is 30% higher.

Performance Graphs

1. Disk: SSD, Image: raw, Preallocated: yes, VMs: 16

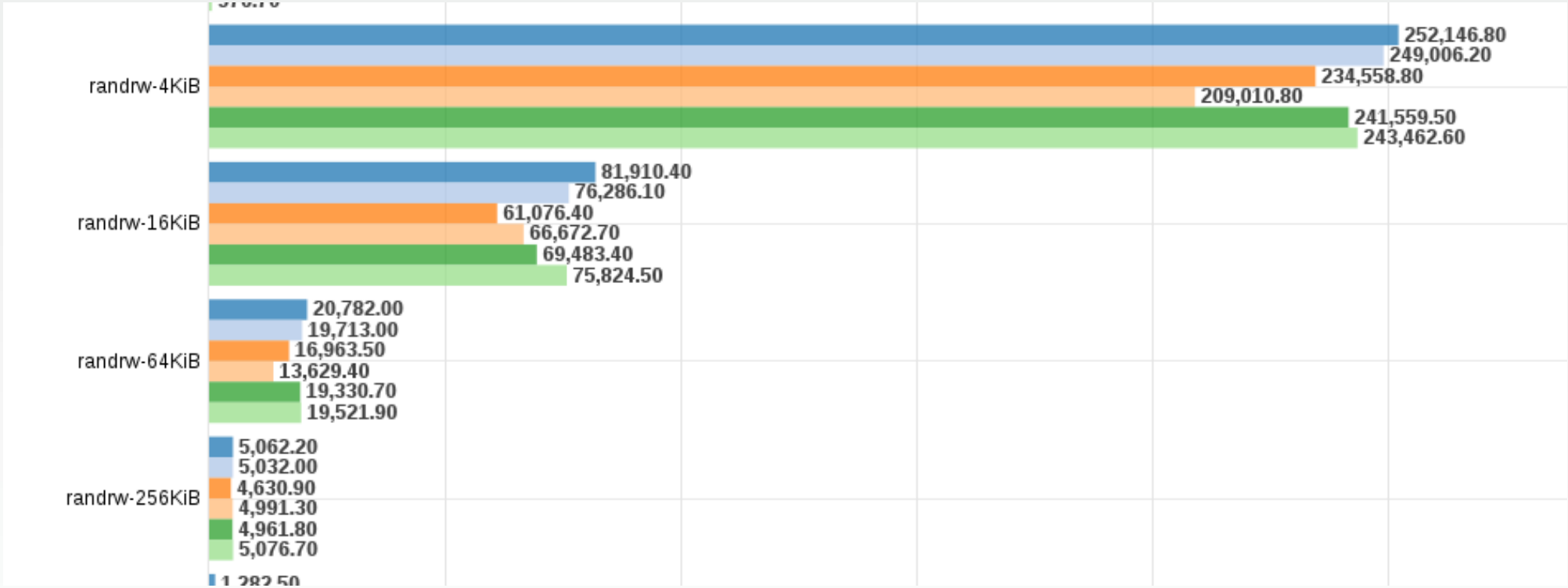
FS: no.Used LVM, aio=native
FS: No.Used LVM, aio=threads
FS: EXT4, aio=native
FS: EXT4, aio=threads
FS: XFS, aio=native
FS: XFS, aio=threads

RandRead



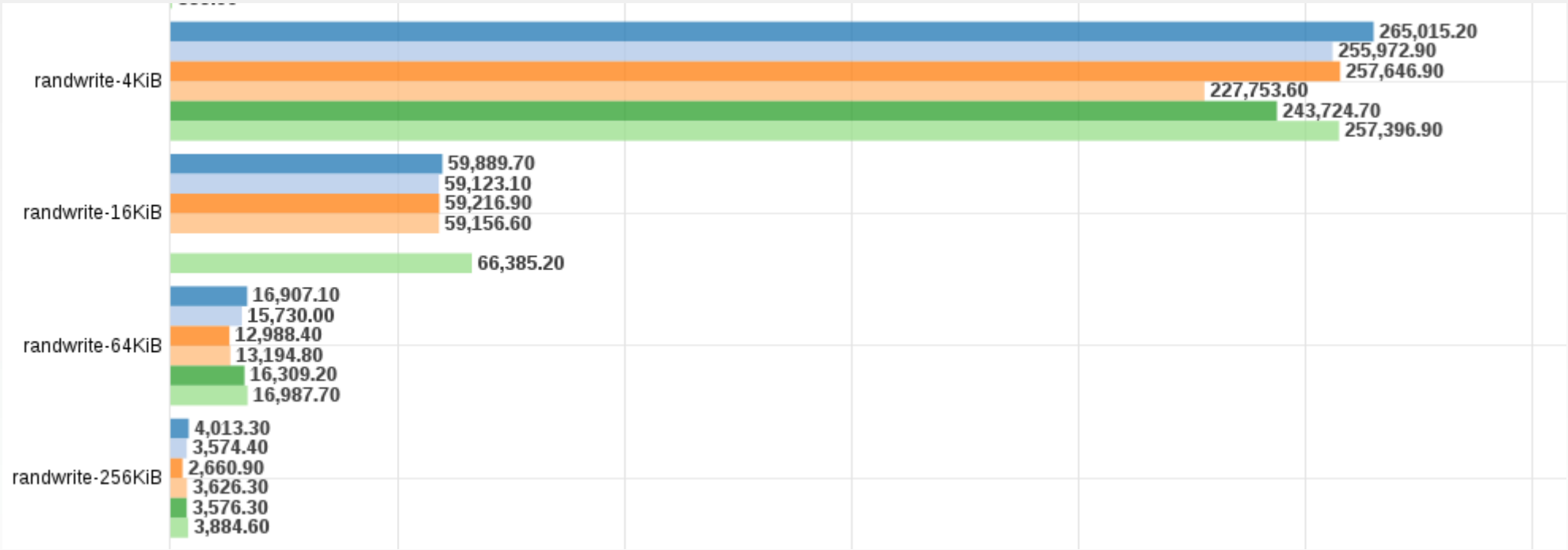
RandReadWrite

FS: no.Used LVM, aio=native
FS: No.Used LVM, aio=threads
FS: EXT4, aio=native
FS: EXT4, aio=threads
FS: XFS, aio=native
FS: XFS, aio=threads



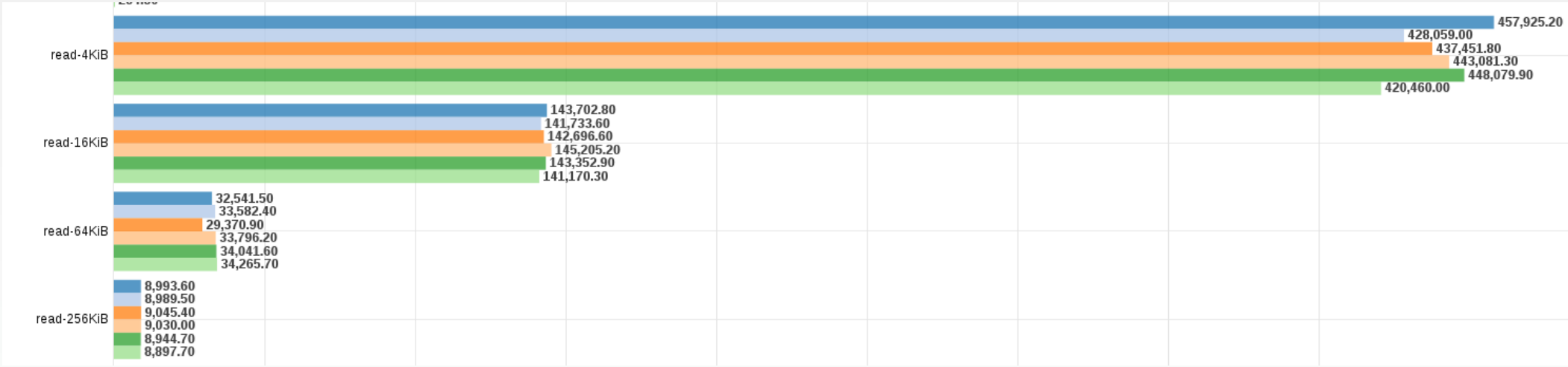
RandWrite

FS: no.Used LVM, aio=native
FS: No.Used LVM, aio=threads
FS: EXT4, aio=native
FS: EXT4, aio=threads
FS: XFS, aio=native
FS: XFS, aio=threads



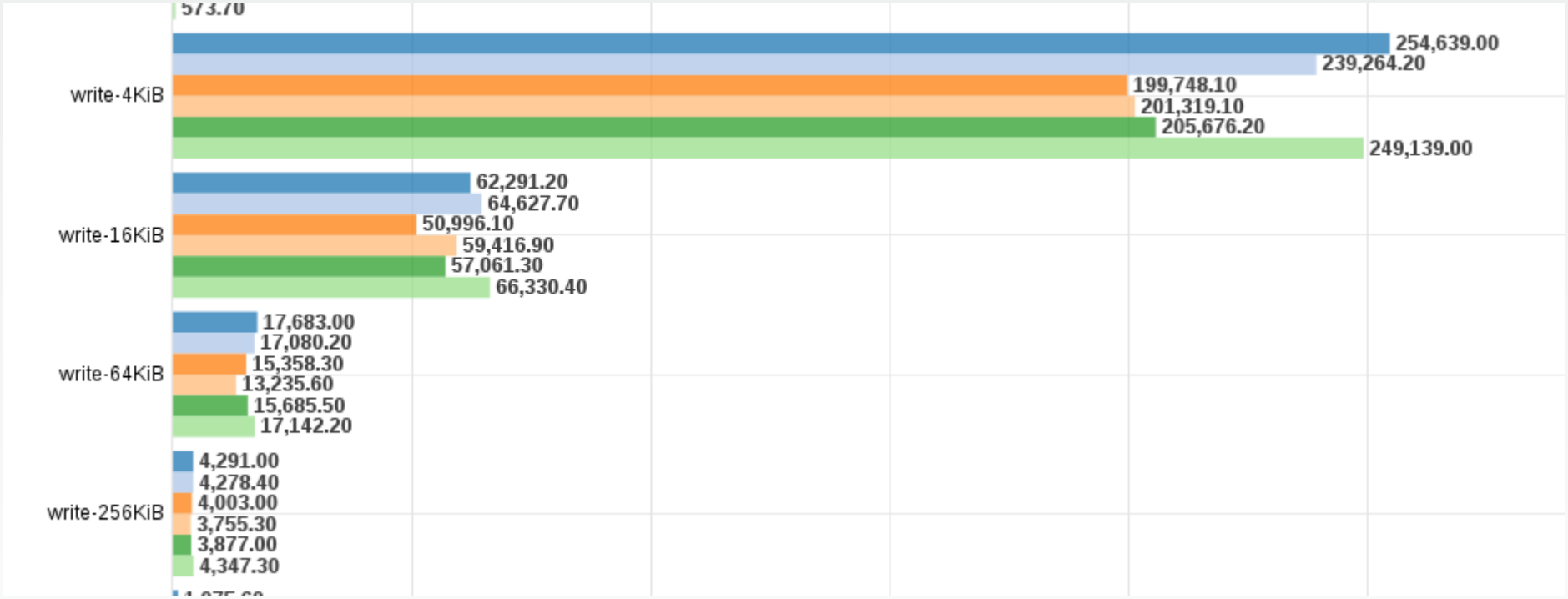
FS: no.Used LVM, aio=native
FS: No.Used LVM, aio=threads
FS: EXT4, aio=native
FS: EXT4, aio=threads
FS: XFS, aio=native
FS: XFS, aio=threads

Seq Read



Seq Write

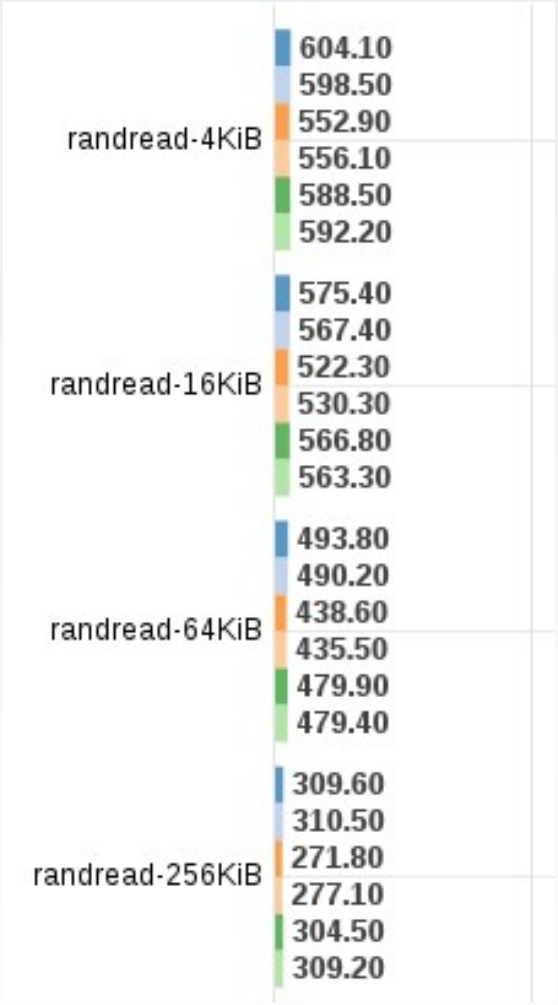
FS: no.Used LVM, aio=native
FS: No.Used LVM, aio=threads
FS: EXT4, aio=native
FS: EXT4, aio=threads
FS: XFS, aio=native
FS: XFS, aio=threads



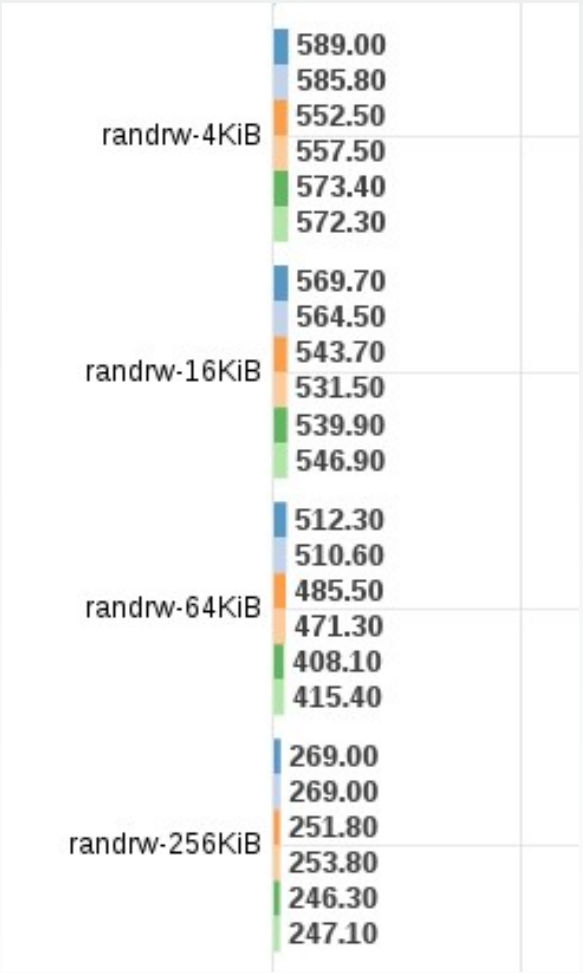
2. Disk: HDD, Image: raw, Preallocated: yes, VMs: 16

FS: no.Used LVM, aio=native
FS: No.Used LVM, aio=threads
FS: EXT4, aio=native
FS: EXT4, aio=threads
FS: XFS, aio=native
FS: XFS, aio=threads

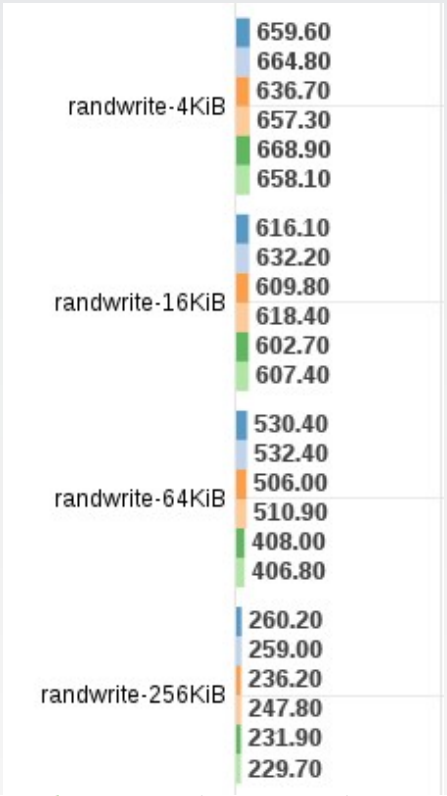
RandRead



RandReadWrite

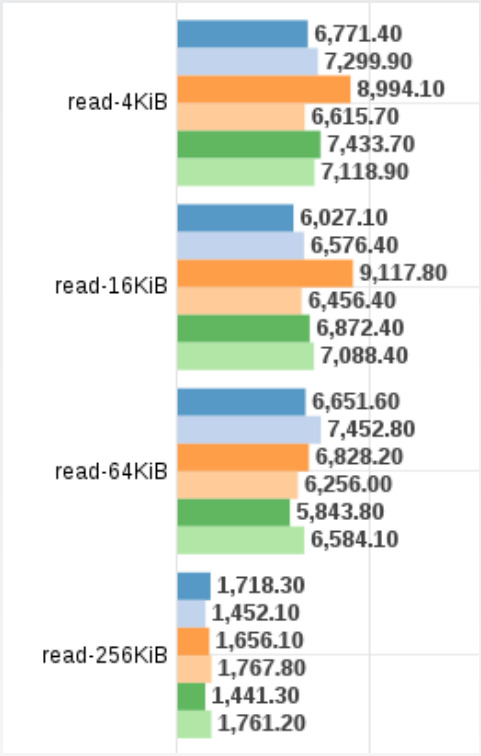


RandWrite

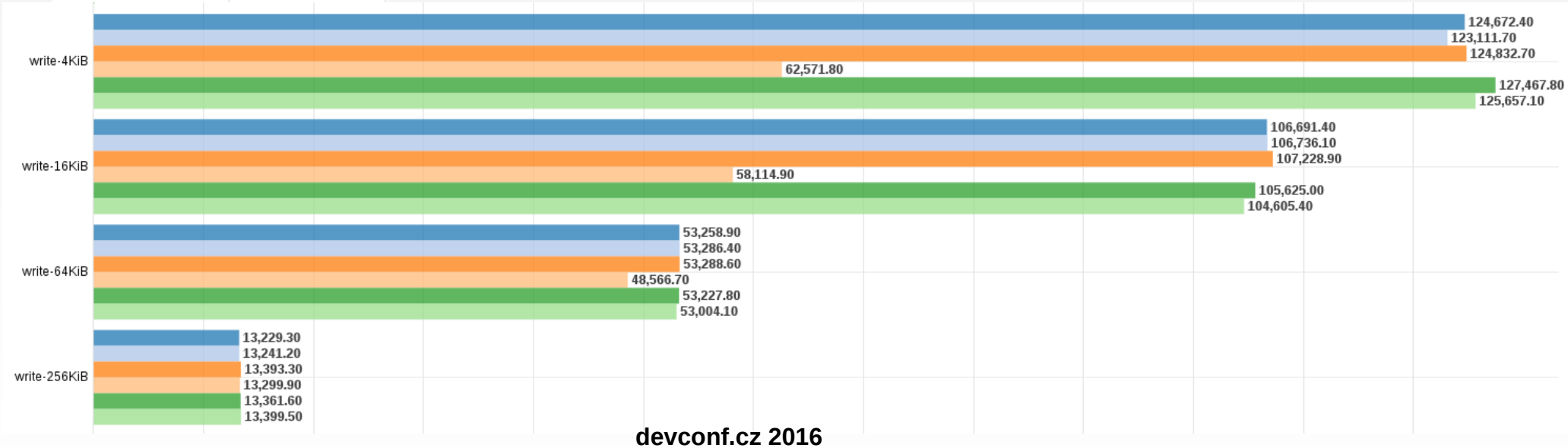


FS: no.Used LVM, aio=native
FS: No.Used LVM, aio=threads
FS: EXT4, aio=native
FS: EXT4, aio=threads
FS: XFS, aio=native
FS: XFS, aio=threads

Seq Read



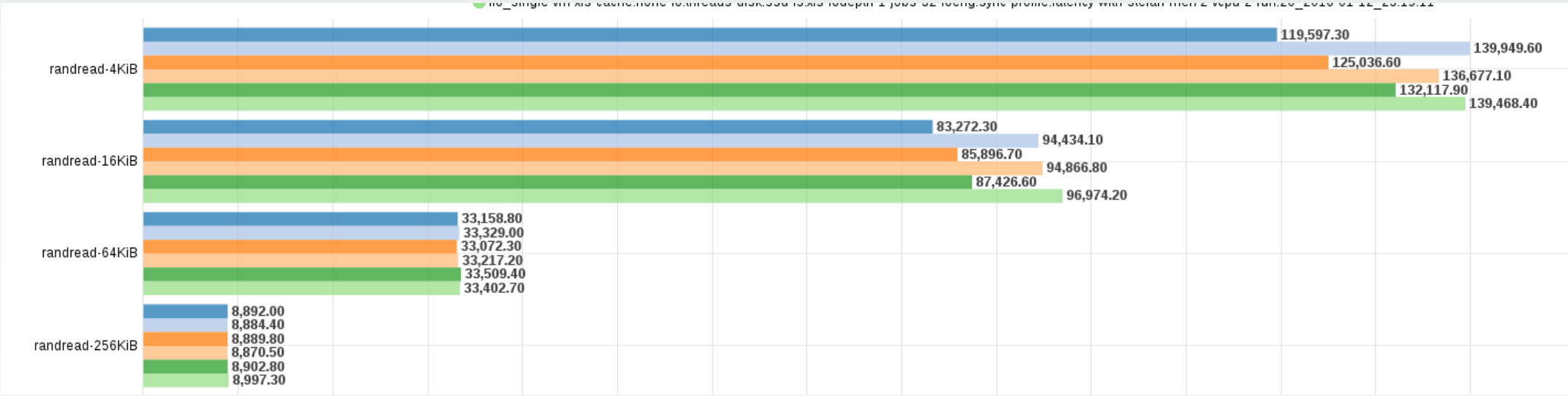
Seq Write



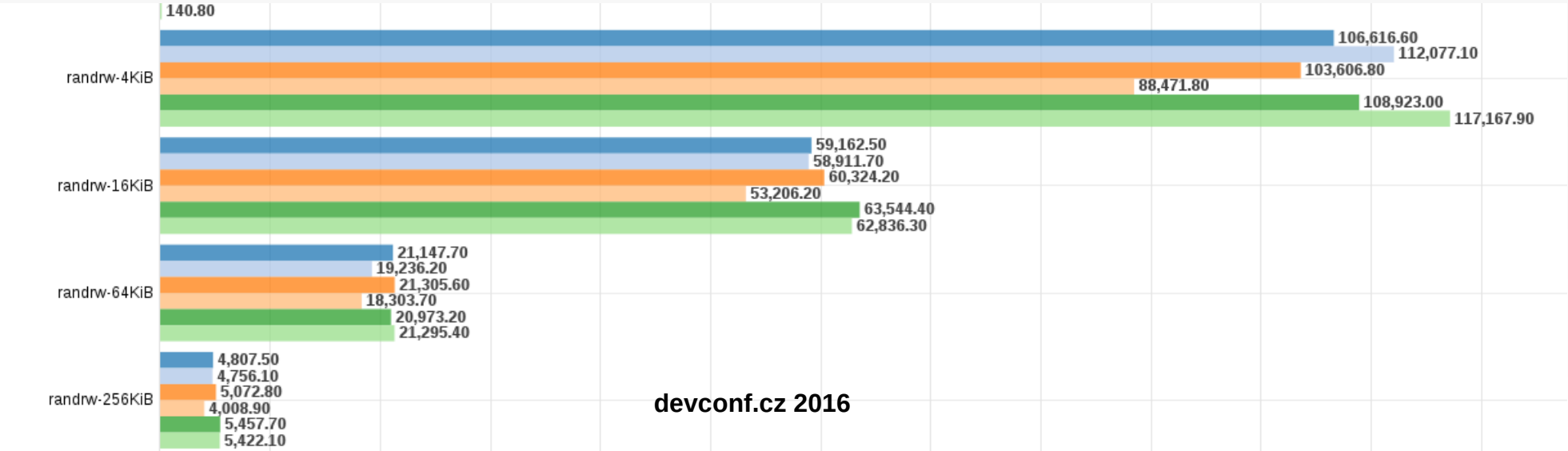
3. Disk: SSD, Image: raw, Preallocated: yes, VMs: 1

FS: no.Used LVM, aio=native
FS: No.Used LVM, aio=threads
FS: EXT4, aio=native
FS: EXT4, aio=threads
FS: XFS, aio=native
FS: XFS, aio=threads

RandRead

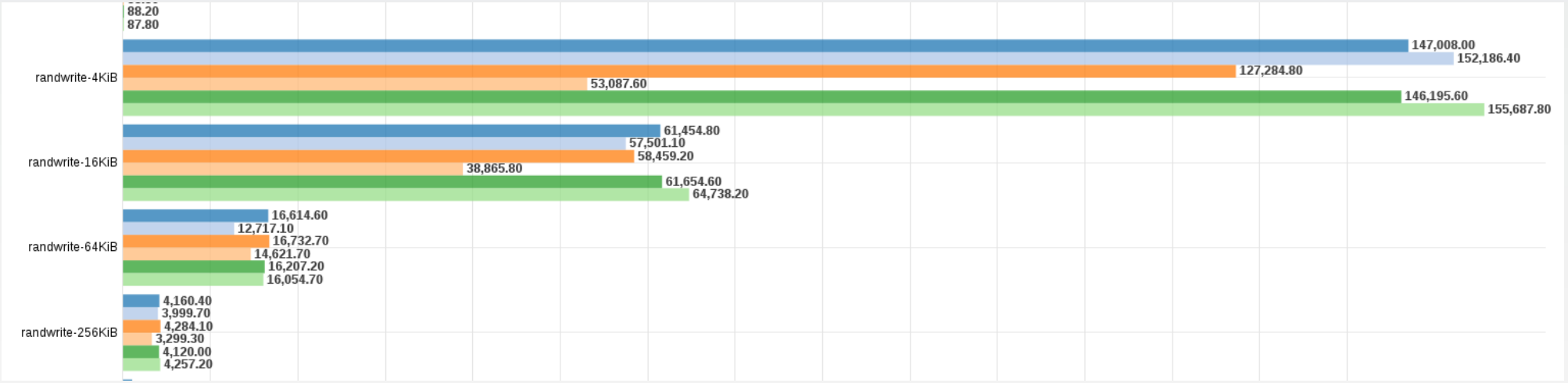


RandRead Write

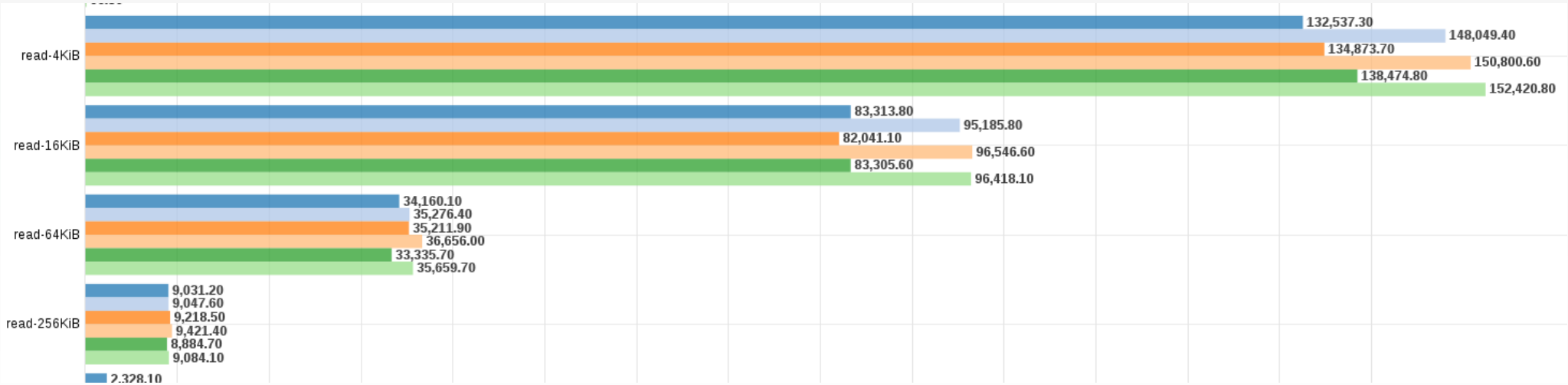


RandRead Write

FS: no.Used LVM, aio=native
FS: No.Used LVM, aio=threads
FS: EXT4, aio=native
FS: EXT4, aio=threads
FS: XFS, aio=native
FS: XFS, aio=threads

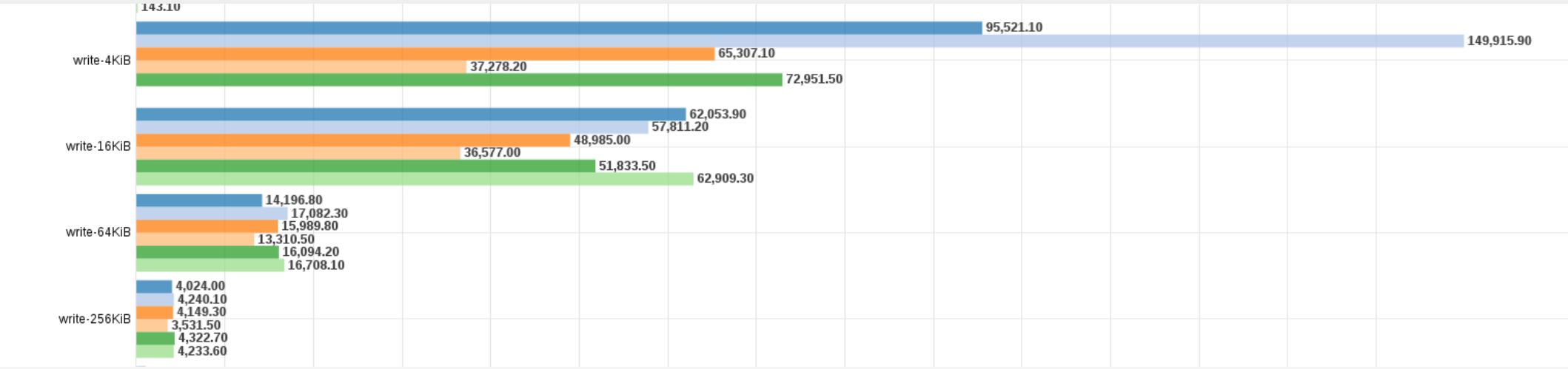


Seq Read



Seq Write

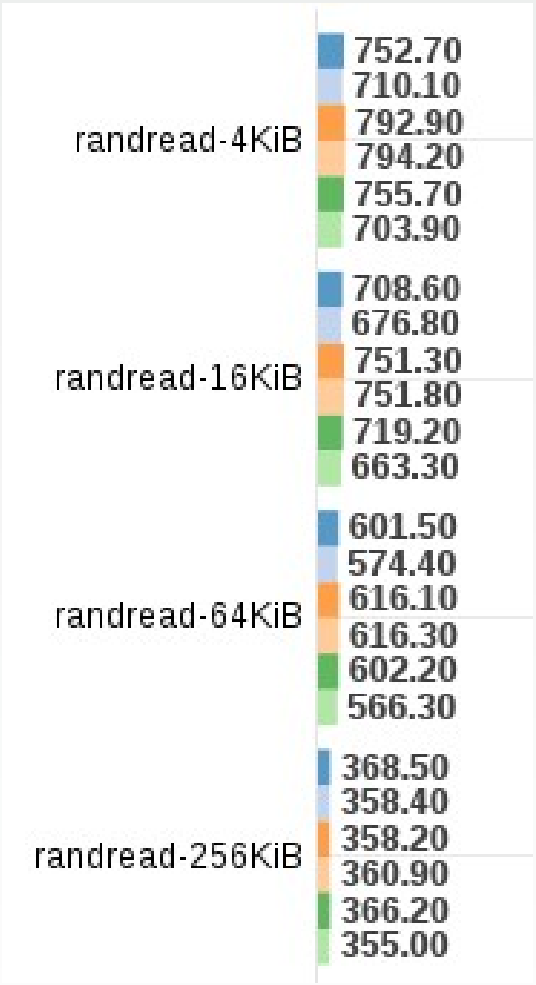
FS: no.Used LVM, aio=native
FS: No.Used LVM, aio=threads
FS: EXT4, aio=native
FS: EXT4, aio=threads
FS: XFS, aio=native
FS: XFS, aio=threads



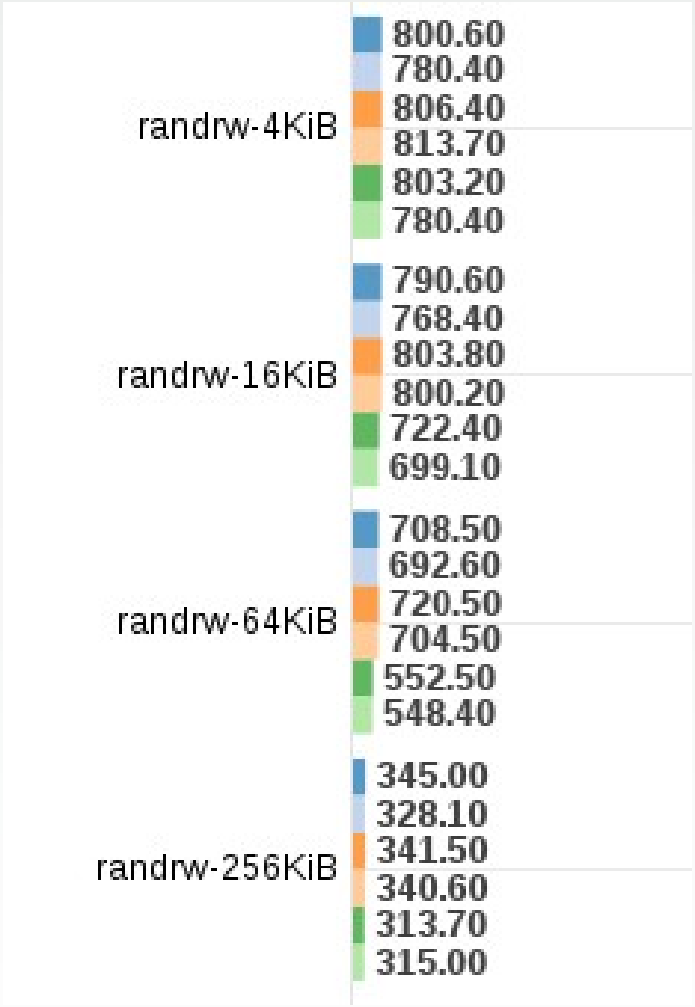
4. Disk: HDD, Image: raw, Preallocated: yes, VMs: 1

FS: no.Used LVM, aio=native
FS: No.Used LVM, aio=threads
FS: EXT4, aio=native
FS: EXT4, aio=threads
FS: XFS, aio=native
FS: XFS, aio=threads

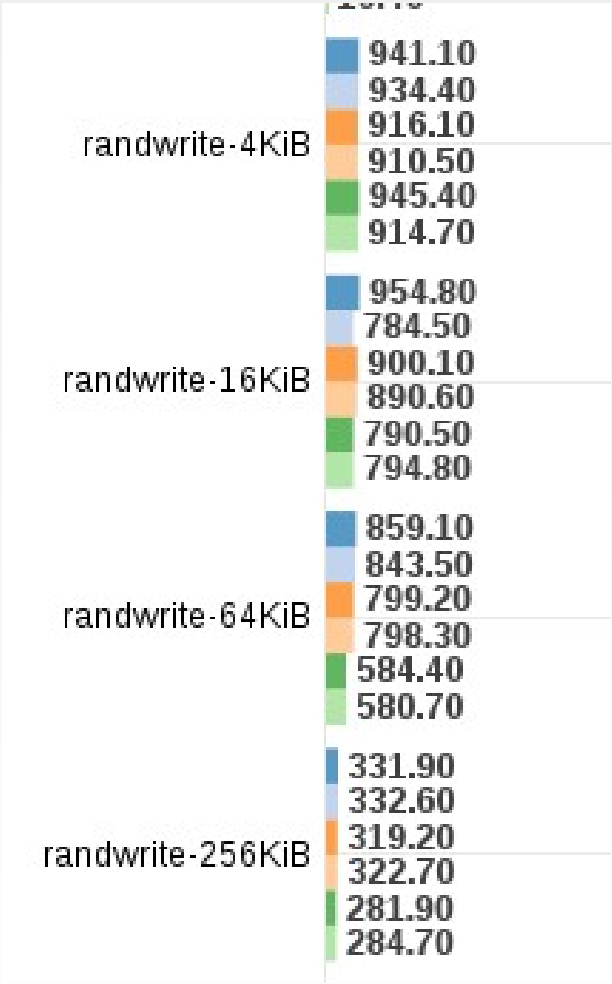
Rand Read



Rand Read Write

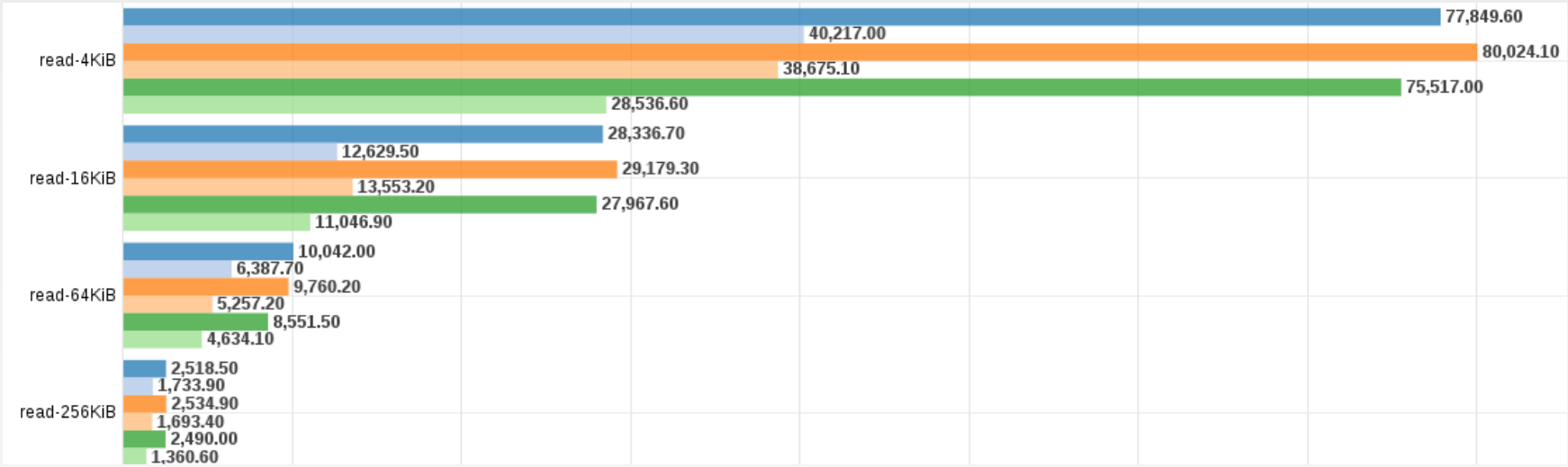


Rand Write

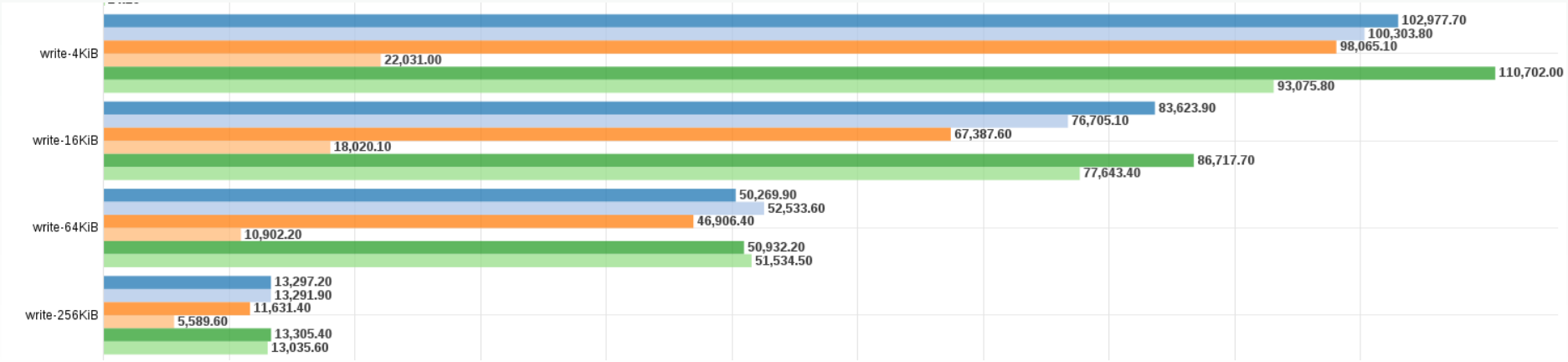


FS: no.Used LVM, aio=native
FS: No.Used LVM, aio=threads
FS: EXT4, aio=native
FS: EXT4, aio=threads
FS: XFS, aio=native
FS: XFS, aio=threads

Seq Read



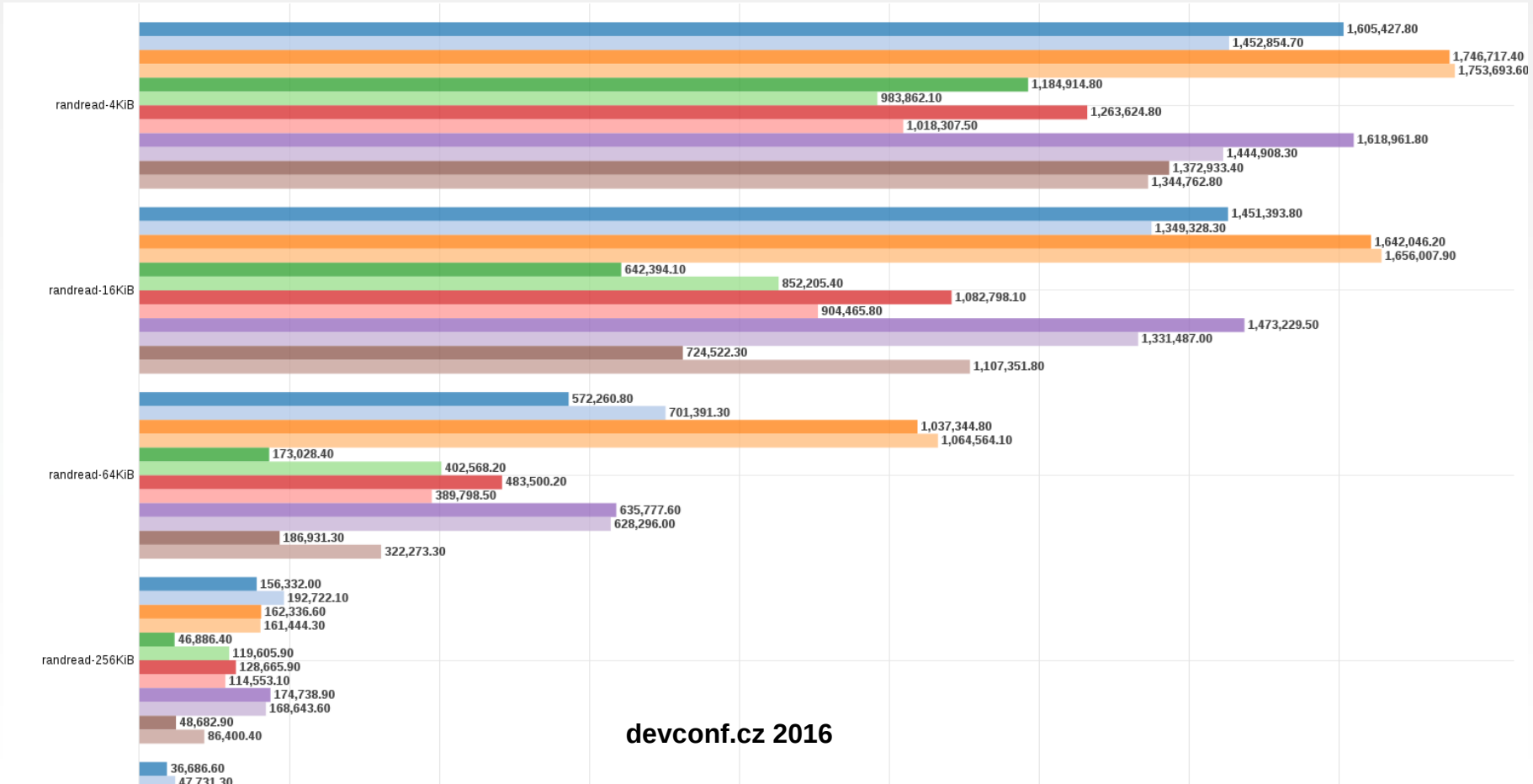
Seq Write



5. Disk: SSD, Image: qcow2, VMs: 16

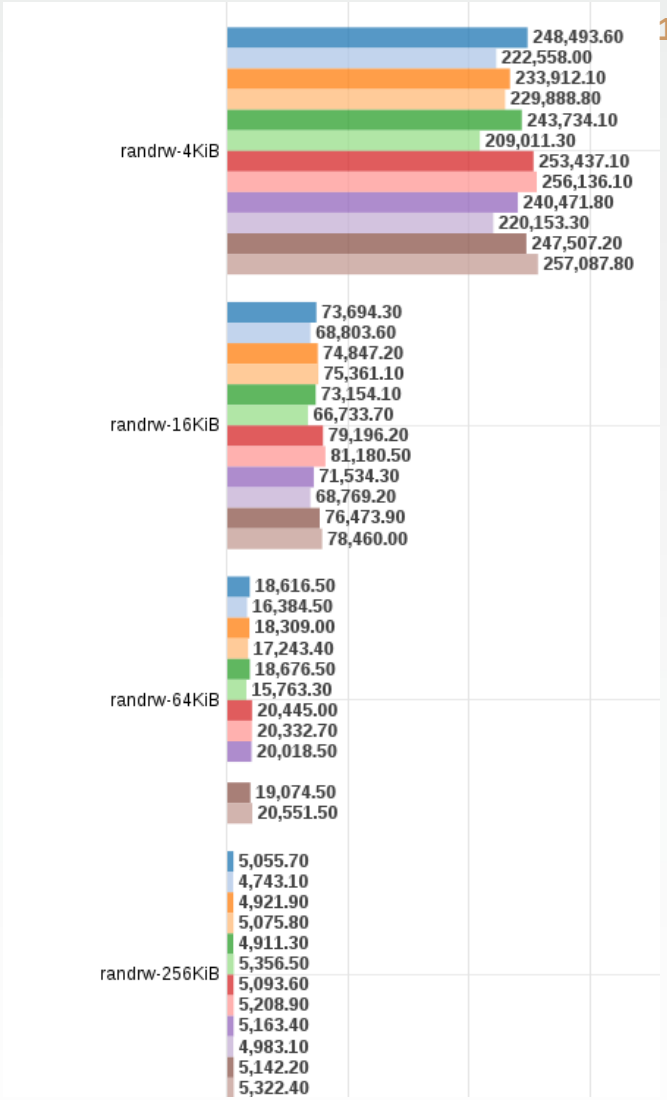
1 FS: EXT4, aio=native, Img: qcow2
2 FS: EXT4, aio=threads, Img: qcow2
3 FS: XFS, aio=native, Img: qcow2
4 FS: XFS, aio=threads, Img: qcow2
5 FS: EXT4, aio=native, Img: qcow2, Prealloc: Falloc
6 FS: EXT4, aio=threads, Img: qcow2, Prealloc: Falloc
7 FS: XFS, aio=native, Img: qcow2, Prealloc: Falloc
8 FS: XFS, aio=threads, Img: qcow2, Prealloc: Falloc
9 FS: EXT4, aio=native, Img: qcow2, Prealloc: Fallocate
10 FS: EXT4, aio=threads, Img: qcow2, Prealloc: Fallocate
11 FS: XFS, aio=native, Img: qcow2, Prealloc: Fallocate
12 FS: XFS, aio=threads, Img: qcow2, Prealloc: Fallocate

RandRead

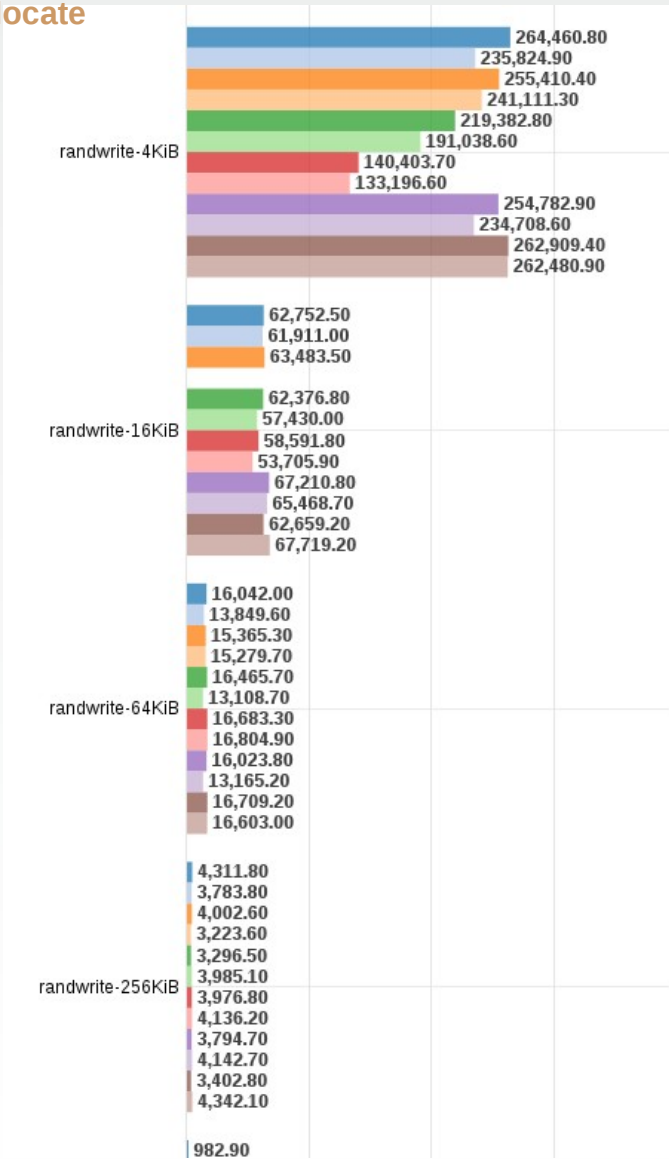


1 FS: EXT4, aio=native, Img: qcow2
2 FS: EXT4, aio=threads, Img: qcow2
3 FS: XFS, aio=native, Img: qcow2
4 FS: XFS, aio=threads, Img: qcow2
5 FS: EXT4, aio=native, Img: qcow2, Prealloc: Falloc
6 FS: EXT4, aio=threads, Img: qcow2, Prealloc: Falloc
7 FS: XFS, aio=native, Img: qcow2, Prealloc: Falloc
8 FS: XFS, aio=threads, Img: qcow2, Prealloc: Falloc
9 FS: EXT4, aio=native, Img: qcow2, Prealloc: Fallocate
10 FS: EXT4, aio=threads, Img: qcow2, Prealloc: Fallocate
11 FS: XFS, aio=native, Img: qcow2, Prealloc: Fallocate
12 FS: XFS, aio=threads, Img: qcow2, Prealloc: Fallocate

RandReadWrite

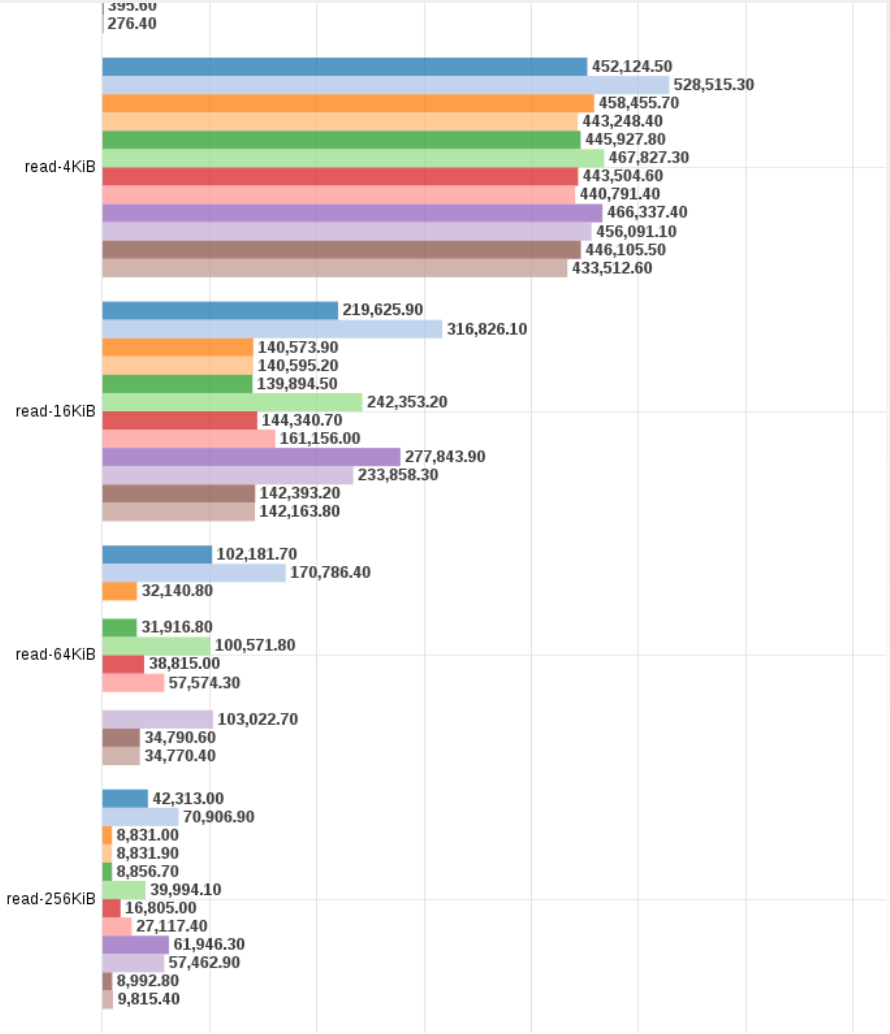


RandWrite

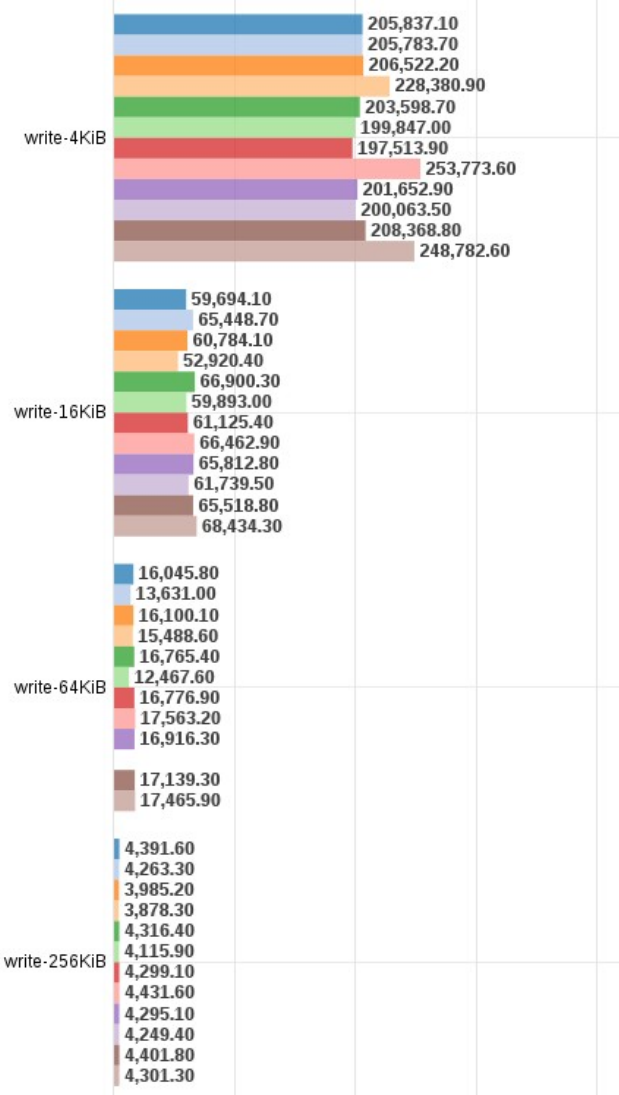


1 FS: EXT4, aio=native, Img: qcow2
2 FS: EXT4, aio=threads, Img: qcow2
3 FS: XFS, aio=native, Img: qcow2
4 FS: XFS, aio=threads, Img: qcow2
5 FS: EXT4, aio=native, Img: qcow2, Prealloc: Falloc
6 FS: EXT4, aio=threads, Img: qcow2, Prealloc: Falloc
7 FS: XFS, aio=native, Img: qcow2, Prealloc: Falloc
8 FS: XFS, aio=threads, Img: qcow2, Prealloc: Falloc
9 FS: EXT4, aio=native, Img: qcow2, Prealloc: Fallocate
10 FS: EXT4, aio=threads, Img: qcow2, Prealloc: Fallocate
11 FS: XFS, aio=native, Img: qcow2, Prealloc: Fallocate
12 FS: XFS, aio=threads, Img: qcow2, Prealloc: Fallocate

Seq Read

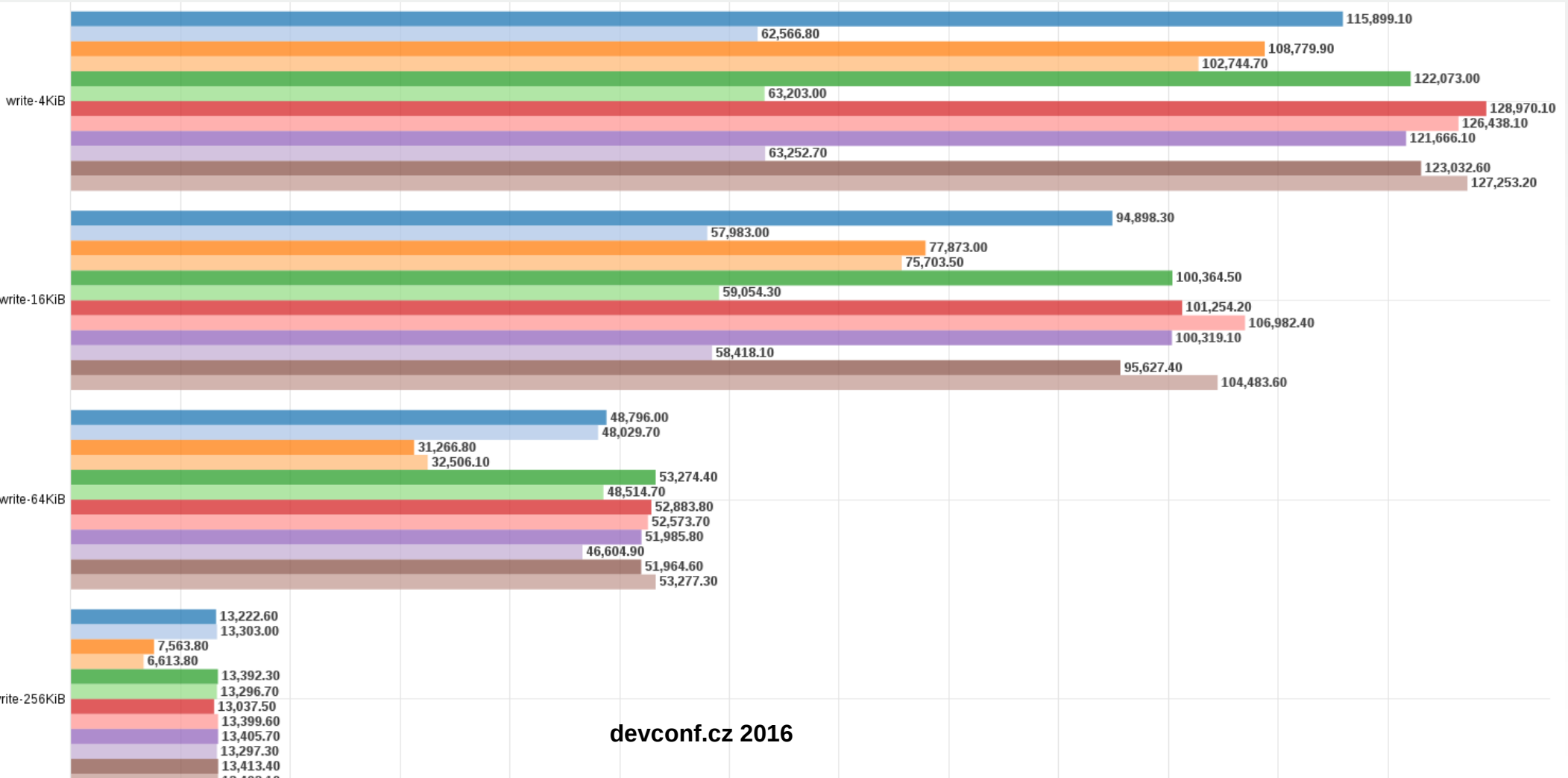


SeqWrite



6. Disk: HDD, Image: qcow2, VMs: 16

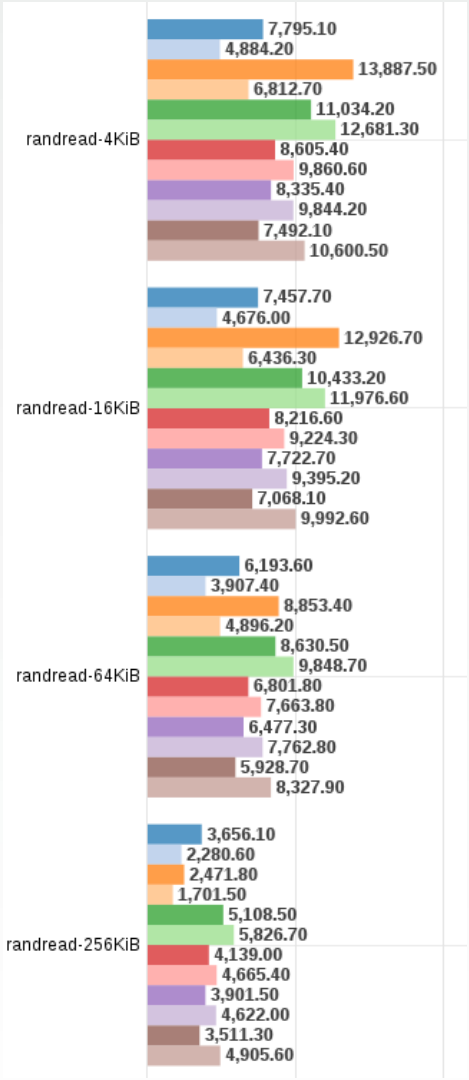
1 FS: EXT4, aio=native, Img: qcow2
2 FS: EXT4, aio=threads, Img: qcow2
3 FS: XFS, aio=native, Img: qcow2
4 FS: XFS, aio=threads, Img: qcow2
5 FS: EXT4, aio=native, Img: qcow2, Prealloc: Falloc
6 FS: EXT4, aio=threads, Img: qcow2, Prealloc: Falloc
7 FS: XFS, aio=native, Img: qcow2, Prealloc: Falloc
8 FS: XFS, aio=threads, Img: qcow2, Prealloc: Falloc
9 FS: EXT4, aio=native, Img: qcow2, Prealloc: Fallocate
10 FS: EXT4, aio=threads, Img: qcow2, Prealloc: Fallocate
11 FS: XFS, aio=native, Img: qcow2, Prealloc: Fallocate
12 FS: XFS, aio=threads, Img: qcow2, Prealloc: Fallocate



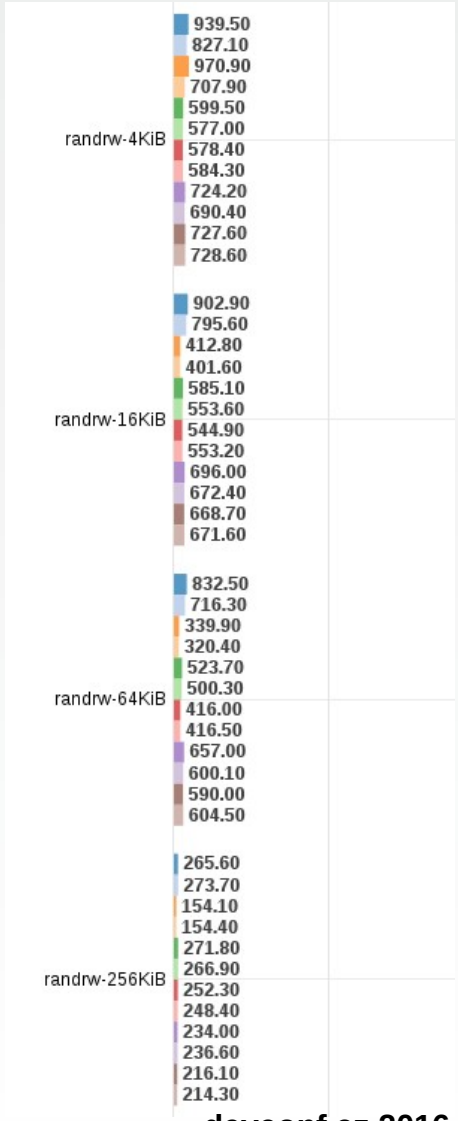
devconf.cz 2016

1 FS: EXT4, aio=native, Img: qcow2
2 FS: EXT4, aio=threads, Img: qcow2
3 FS: XFS, aio=native, Img: qcow2
4 FS: XFS, aio=threads, Img: qcow2
5 FS: EXT4, aio=native, Img: qcow2, Prealloc: Falloc
6 FS: EXT4, aio=threads, Img: qcow2, Prealloc: Falloc
7 FS: XFS, aio=native, Img: qcow2, Prealloc: Falloc
8 FS: XFS, aio=threads, Img: qcow2, Prealloc: Falloc
9 FS: EXT4, aio=native, Img: qcow2, Prealloc: Fallocate
10 FS: EXT4, aio=threads, Img: qcow2, Prealloc: Fallocate
11 FS: XFS, aio=native, Img: qcow2, Prealloc: Fallocate
12 FS: XFS, aio=threads, Img: qcow2, Prealloc: Fallocate

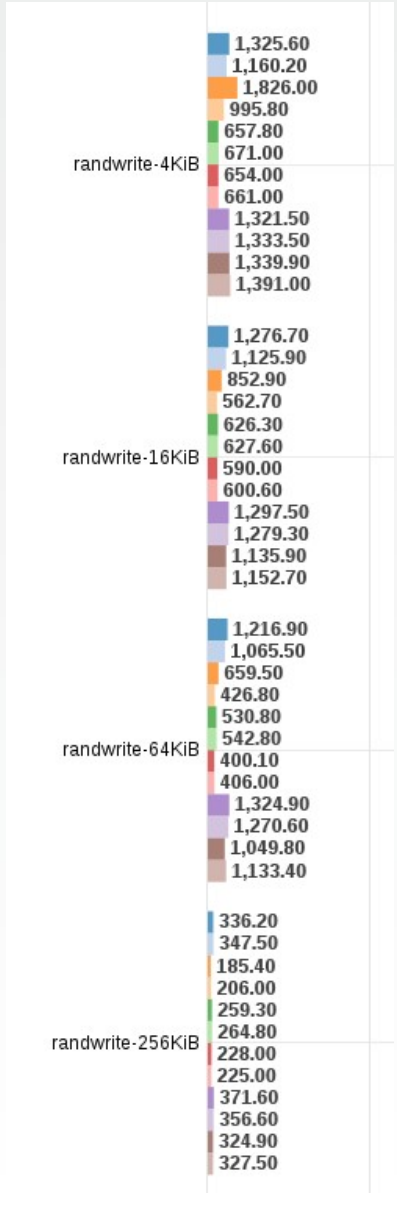
RandRead



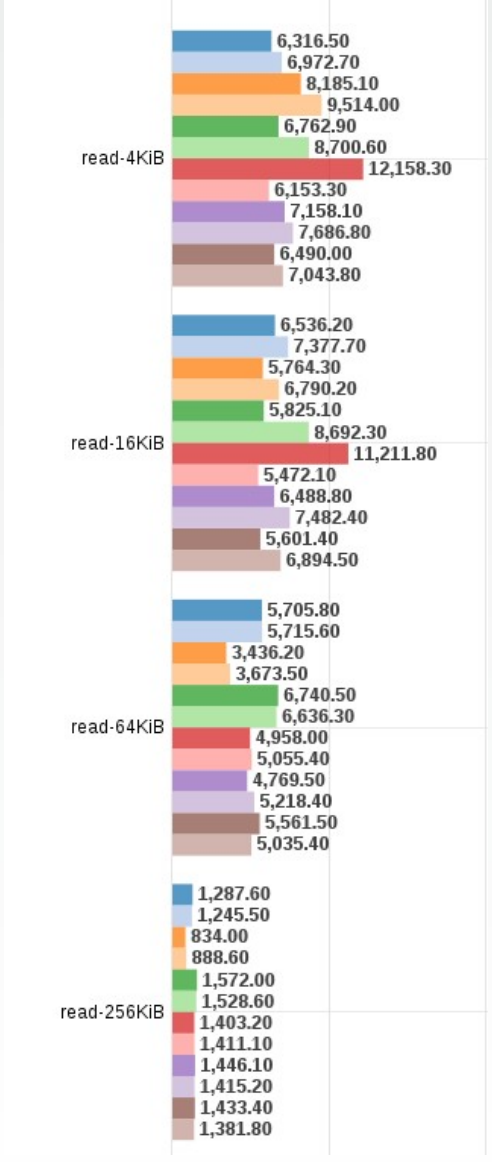
Rand Read Write



Rand Write

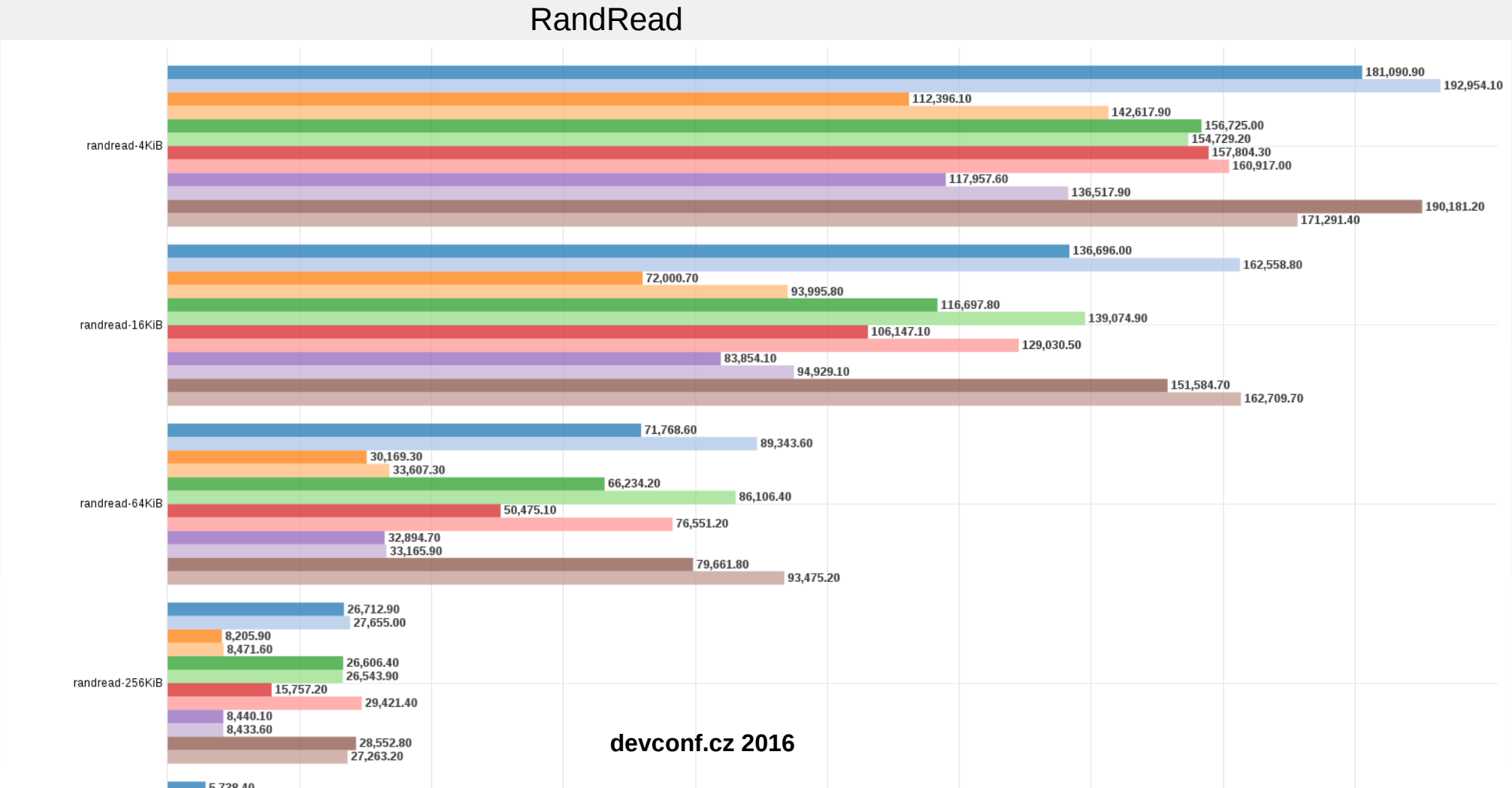


Seq Read



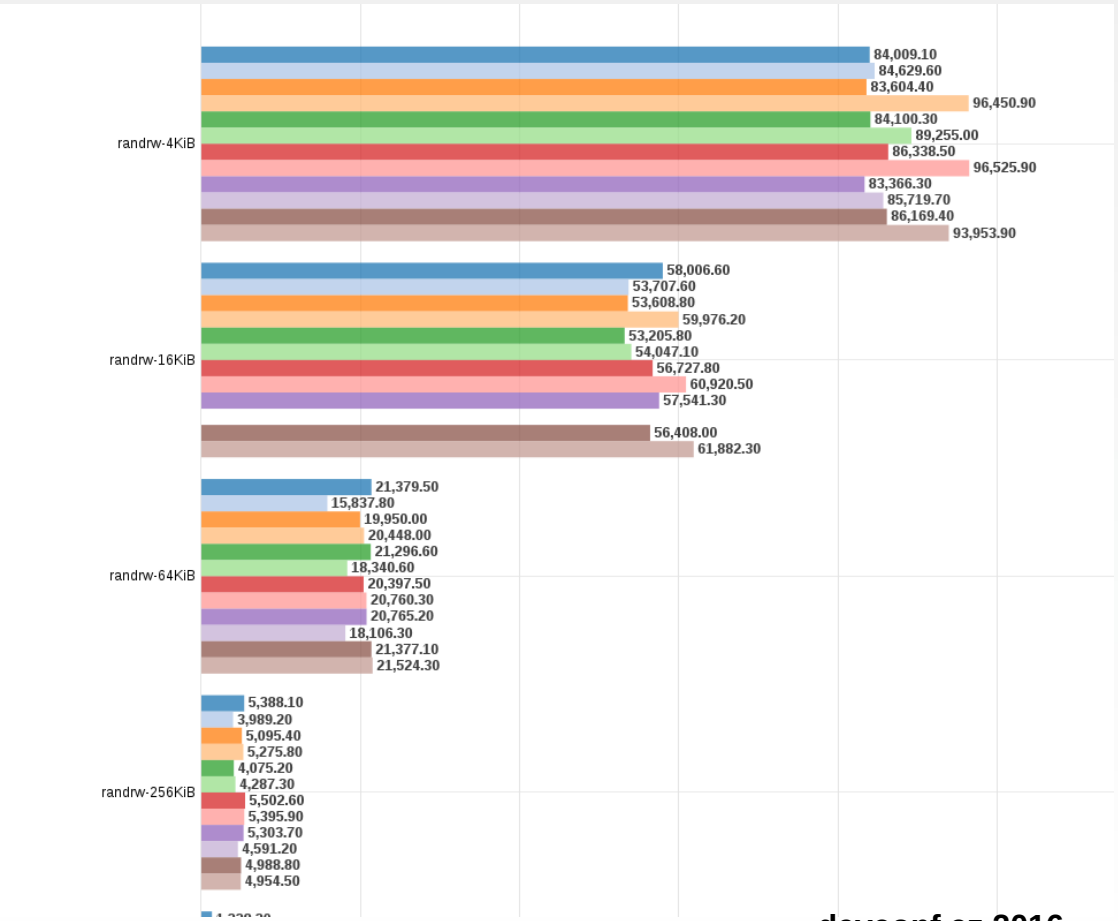
7. Disk: SSD, Image: qcow2, VMs: 1

1 FS: EXT4, aio=native, Img: qcow2
2 FS: EXT4, aio=threads, Img: qcow2
3 FS: XFS, aio=native, Img: qcow2
4 FS: XFS, aio=threads, Img: qcow2
5 FS: EXT4, aio=native, Img: qcow2, Prealloc: Falloc
6 FS: EXT4, aio=threads, Img: qcow2, Prealloc: Falloc
7 FS: XFS, aio=native, Img: qcow2, Prealloc: Falloc
8 FS: XFS, aio=threads, Img: qcow2, Prealloc: Falloc
9 FS: EXT4, aio=native, Img: qcow2, Prealloc: Fallocate
10 FS: EXT4, aio=threads, Img: qcow2, Prealloc: Fallocate
11 FS: XFS, aio=native, Img: qcow2, Prealloc: Fallocate
12 FS: XFS, aio=threads, Img: qcow2, Prealloc: Fallocate

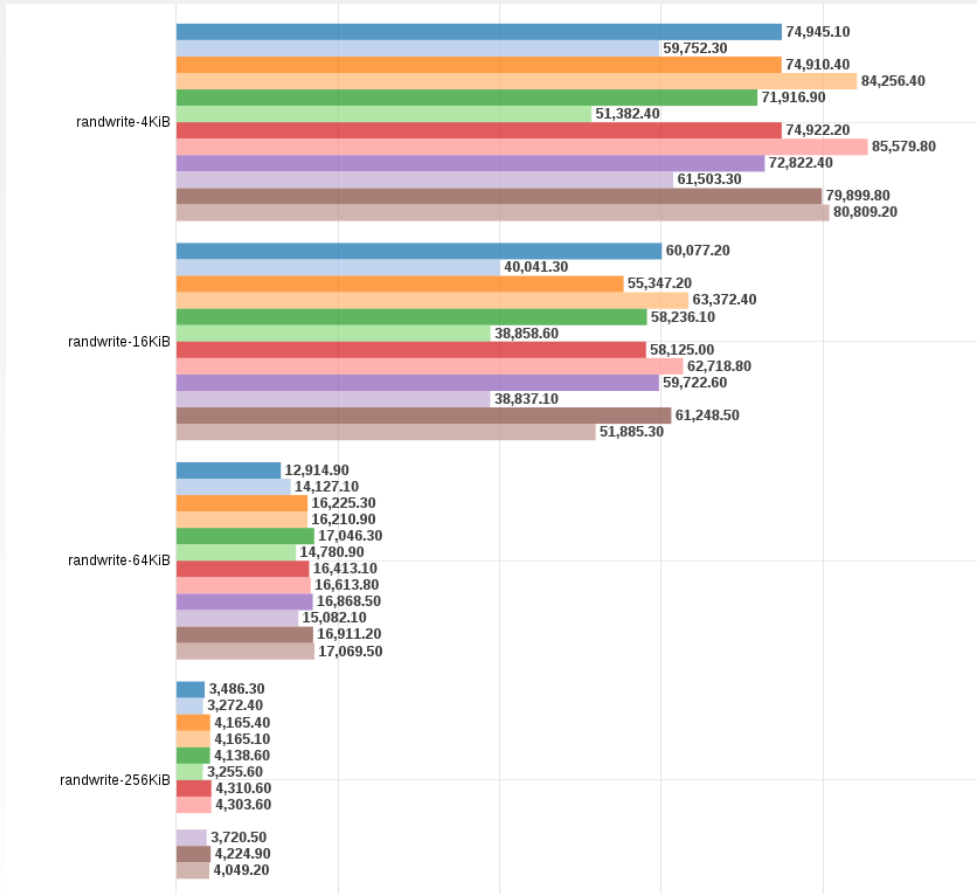


1 FS: EXT4, aio=native, Img: qcow2
2 FS: EXT4, aio=threads, Img: qcow2
3 FS: XFS, aio=native, Img: qcow2
4 FS: XFS, aio=threads, Img: qcow2
5 FS: EXT4, aio=native, Img: qcow2, Prealloc: Falloc
6 FS: EXT4, aio=threads, Img: qcow2, Prealloc: Falloc
7 FS: XFS, aio=native, Img: qcow2, Prealloc: Falloc
8 FS: XFS, aio=threads, Img: qcow2, Prealloc: Falloc
9 FS: EXT4, aio=native, Img: qcow2, Prealloc: Fallocate
10 FS: EXT4, aio=threads, Img: qcow2, Prealloc: Fallocate
11 FS: XFS, aio=native, Img: qcow2, Prealloc: Fallocate
12 FS: XFS, aio=threads, Img: qcow2, Prealloc: Fallocate

RandReadWrite

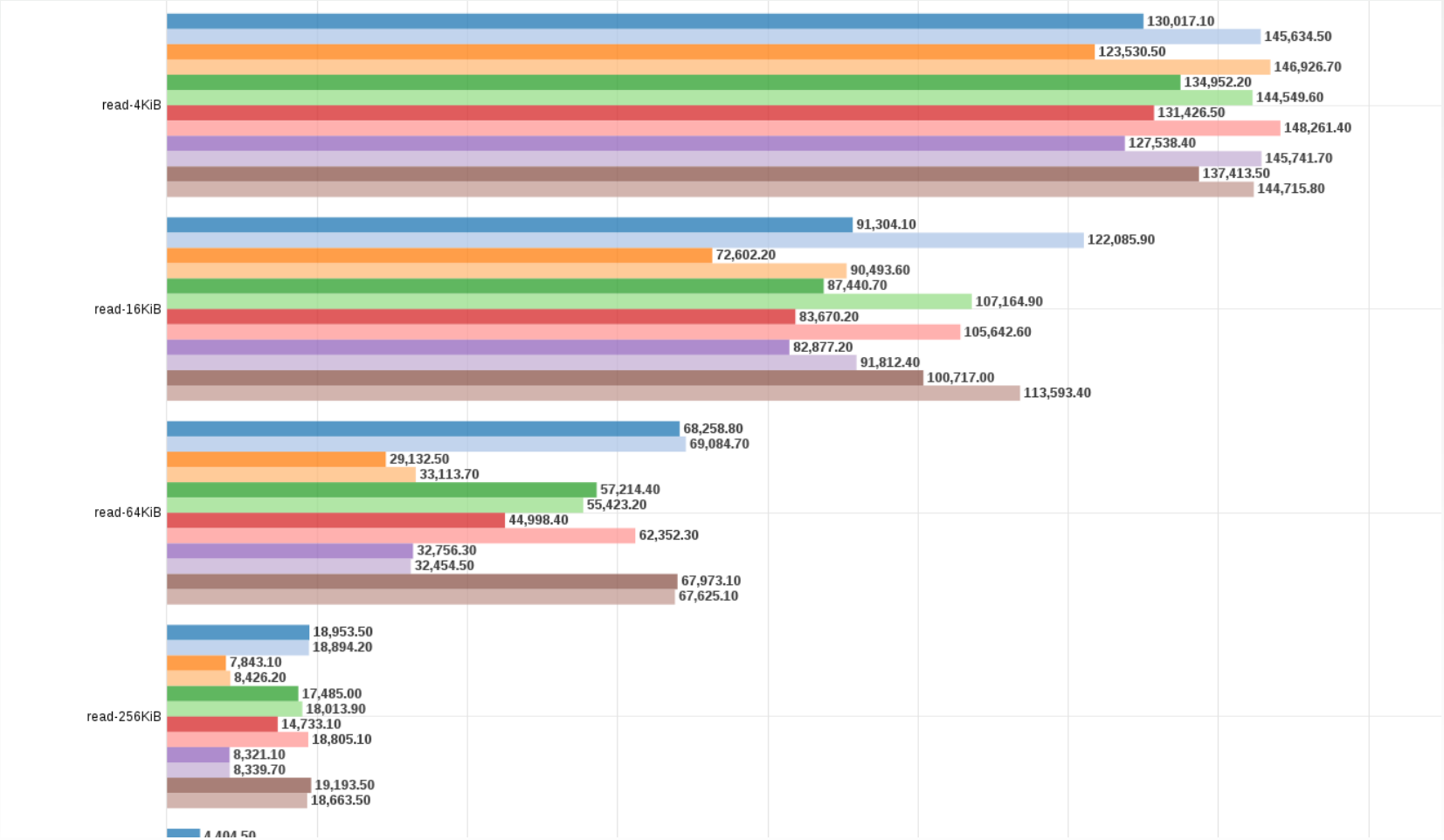


RandWrite



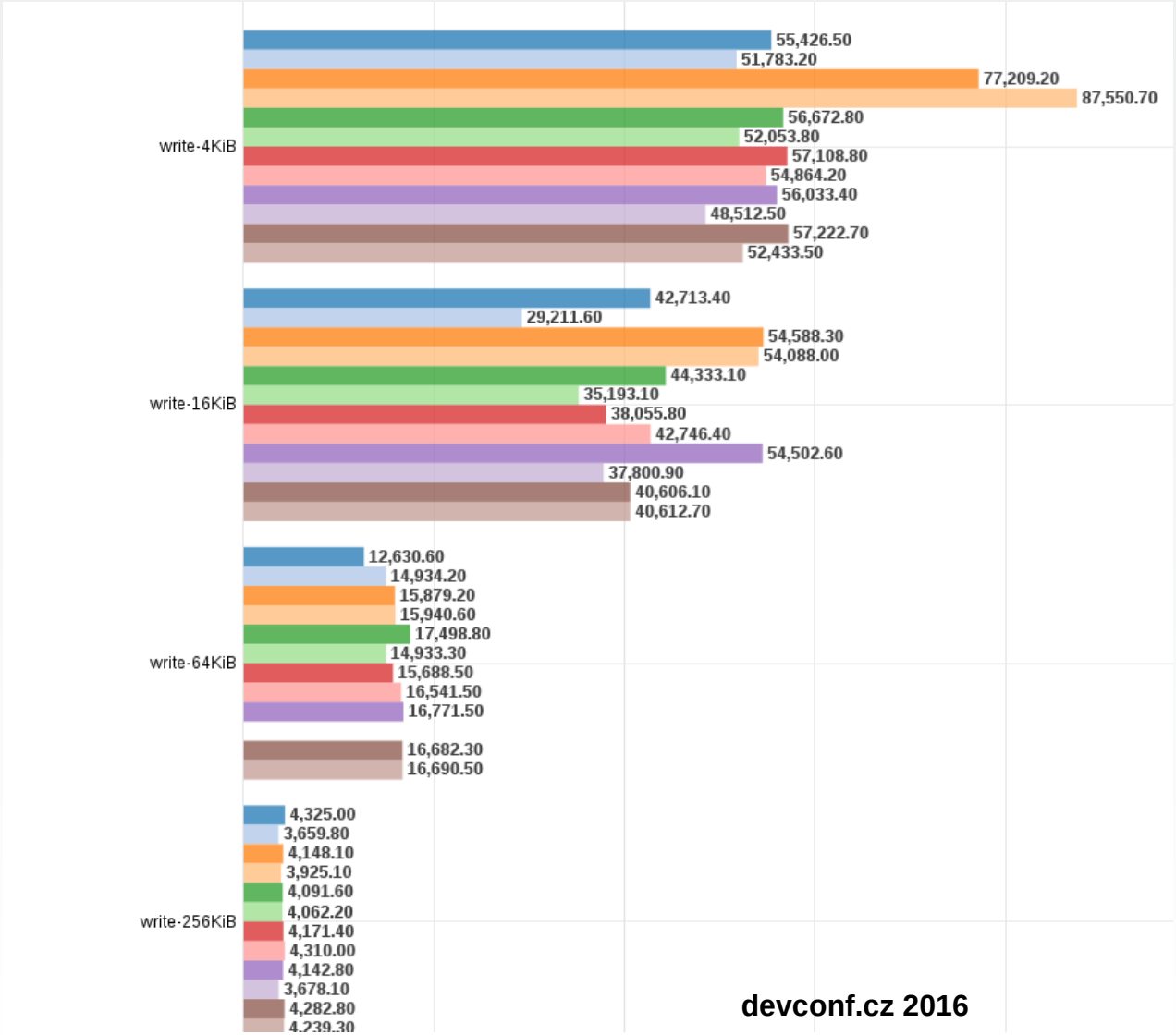
Seq Read

1 FS: EXT4, aio=native, Img: qcow2
2 FS: EXT4, aio=threads, Img: qcow2
3 FS: XFS, aio=native, Img: qcow2
4 FS: XFS, aio=threads, Img: qcow2
5 FS: EXT4, aio=native, Img: qcow2, Prealloc: Falloc
6 FS: EXT4, aio=threads, Img: qcow2, Prealloc: Falloc
7 FS: XFS, aio=native, Img: qcow2, Prealloc: Falloc
8 FS: XFS, aio=threads, Img: qcow2, Prealloc: Falloc
9 FS: EXT4, aio=native, Img: qcow2, Prealloc: Fallocate
10 FS: EXT4, aio=threads, Img: qcow2, Prealloc: Fallocate
11 FS: XFS, aio=native, Img: qcow2, Prealloc: Fallocate
12 FS: XFS, aio=threads, Img: qcow2, Prealloc: Fallocate



Seq Write

1 FS: EXT4, aio=native, Img: qcow2
2 FS: EXT4, aio=threads, Img: qcow2
3 FS: XFS, aio=native, Img: qcow2
4 FS: XFS, aio=threads, Img: qcow2
5 FS: EXT4, aio=native, Img: qcow2, Prealloc: Falloc
6 FS: EXT4, aio=threads, Img: qcow2, Prealloc: Falloc
7 FS: XFS, aio=native, Img: qcow2, Prealloc: Falloc
8 FS: XFS, aio=threads, Img: qcow2, Prealloc: Falloc
9 FS: EXT4, aio=native, Img: qcow2, Prealloc: Fallocate
10 FS: EXT4, aio=threads, Img: qcow2, Prealloc: Fallocate
11 FS: XFS, aio=native, Img: qcow2, Prealloc: Fallocate
12 FS: XFS, aio=threads, Img: qcow2, Prealloc: Fallocate

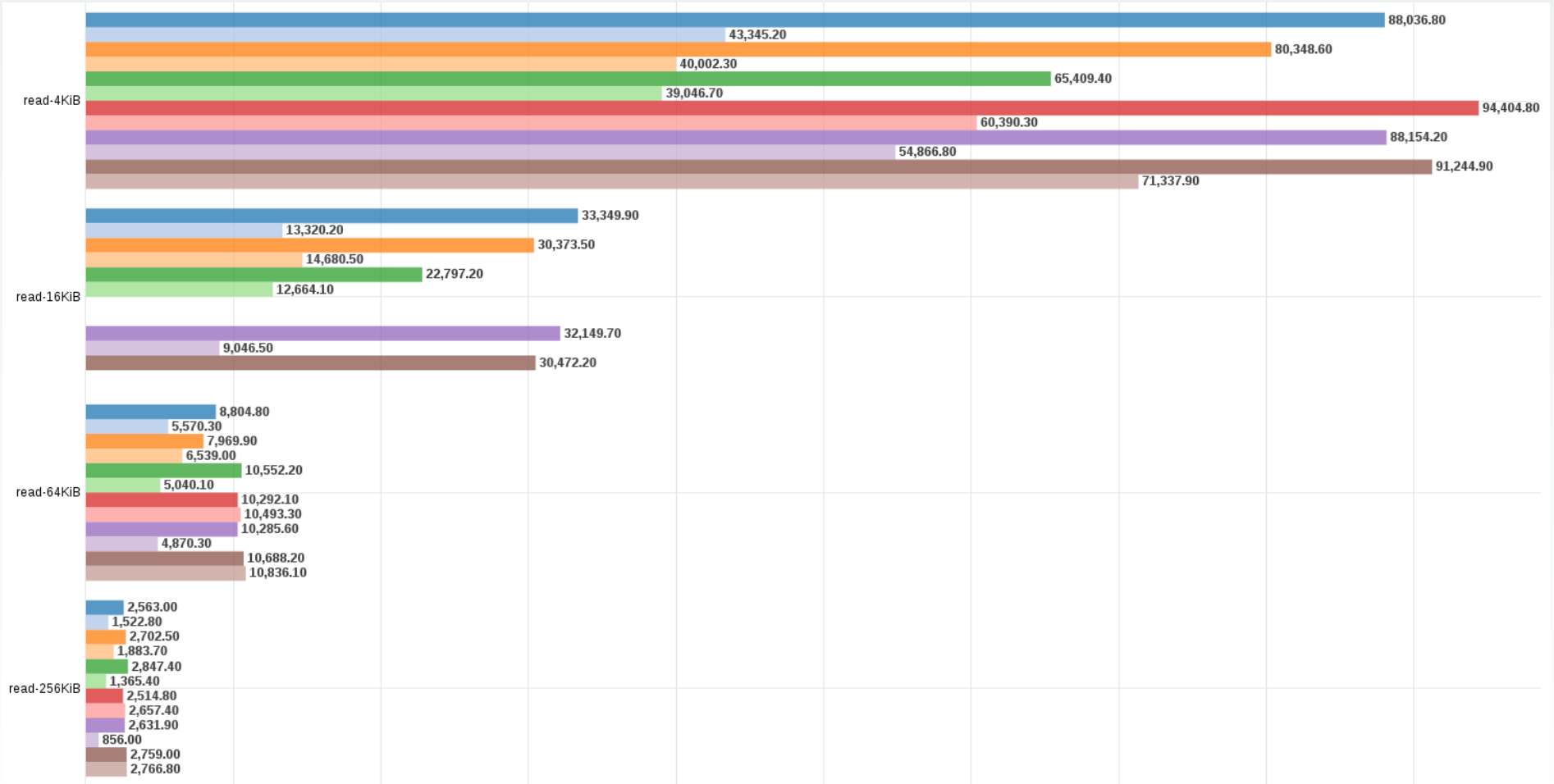


devconf.cz 2016

8. Disk: HDD, Image: qcow2, VMs: 1

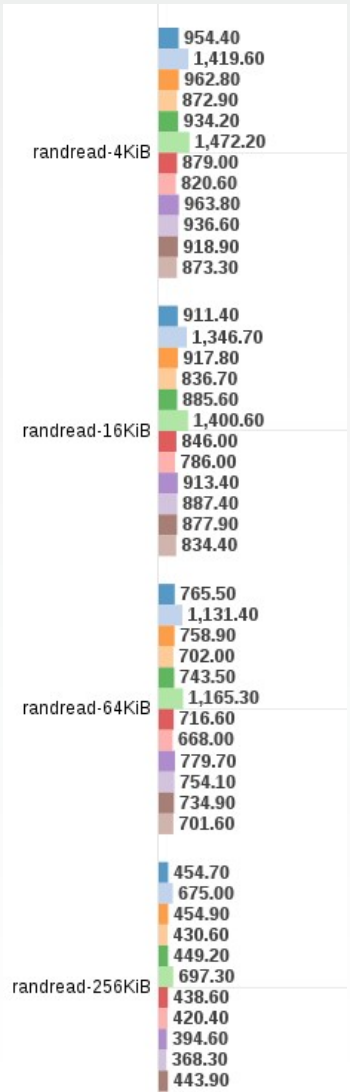
1 FS: EXT4, aio=native, Img: qcow2
2 FS: EXT4, aio=threads, Img: qcow2
3 FS: XFS, aio=native, Img: qcow2
4 FS: XFS, aio=threads, Img: qcow2
5 FS: EXT4, aio=native, Img: qcow2, Prealloc: Falloc
6 FS: EXT4, aio=threads, Img: qcow2, Prealloc: Falloc
7 FS: XFS, aio=native, Img: qcow2, Prealloc: Falloc
8 FS: XFS, aio=threads, Img: qcow2, Prealloc: Falloc
9 FS: EXT4, aio=native, Img: qcow2, Prealloc: Fallocate
10 FS: EXT4, aio=threads, Img: qcow2, Prealloc: Fallocate
11 FS: XFS, aio=native, Img: qcow2, Prealloc: Fallocate
12 FS: XFS, aio=threads, Img: qcow2, Prealloc: Fallocate

Seq Read

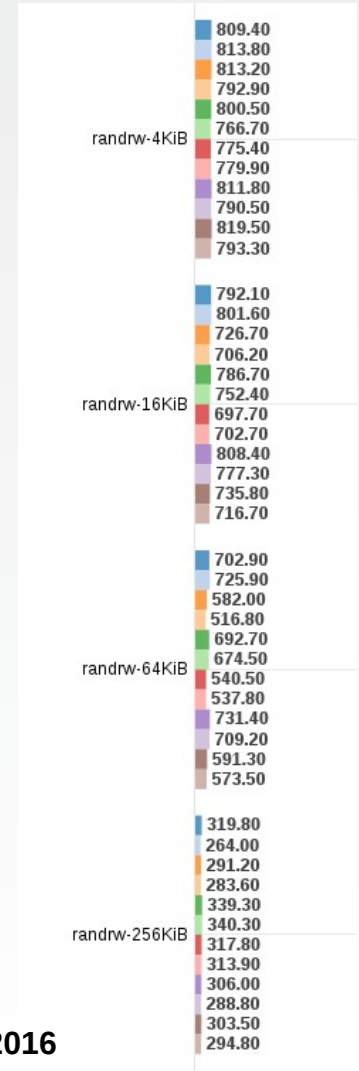


1 FS: EXT4, aio=native, Img: qcow2
2 FS: EXT4, aio=threads, Img: qcow2
3 FS: XFS, aio=native, Img: qcow2
4 FS: XFS, aio=threads, Img: qcow2
5 FS: EXT4, aio=native, Img: qcow2, Prealloc: Falloc
6 FS: EXT4, aio=threads, Img: qcow2, Prealloc: Falloc
7 FS: XFS, aio=native, Img: qcow2, Prealloc: Falloc
8 FS: XFS, aio=threads, Img: qcow2, Prealloc: Falloc
9 FS: EXT4, aio=native, Img: qcow2, Prealloc: Fallocate
10 FS: EXT4, aio=threads, Img: qcow2, Prealloc: Fallocate
11 FS: XFS, aio=native, Img: qcow2, Prealloc: Fallocate
12 FS: XFS, aio=threads, Img: qcow2, Prealloc: Fallocate

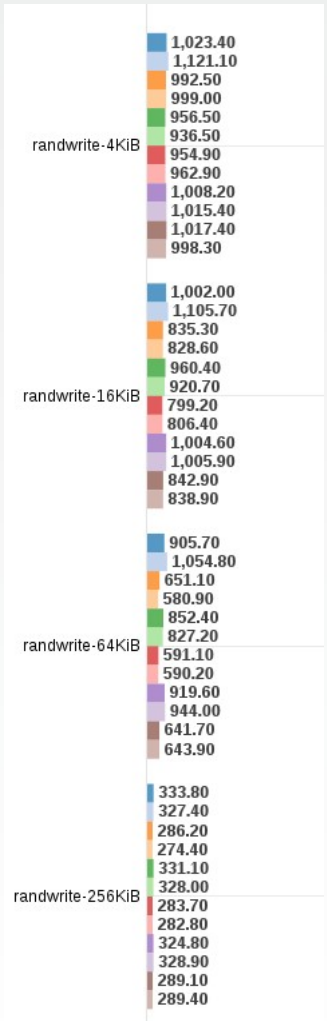
Rand Read



Rand Read Write

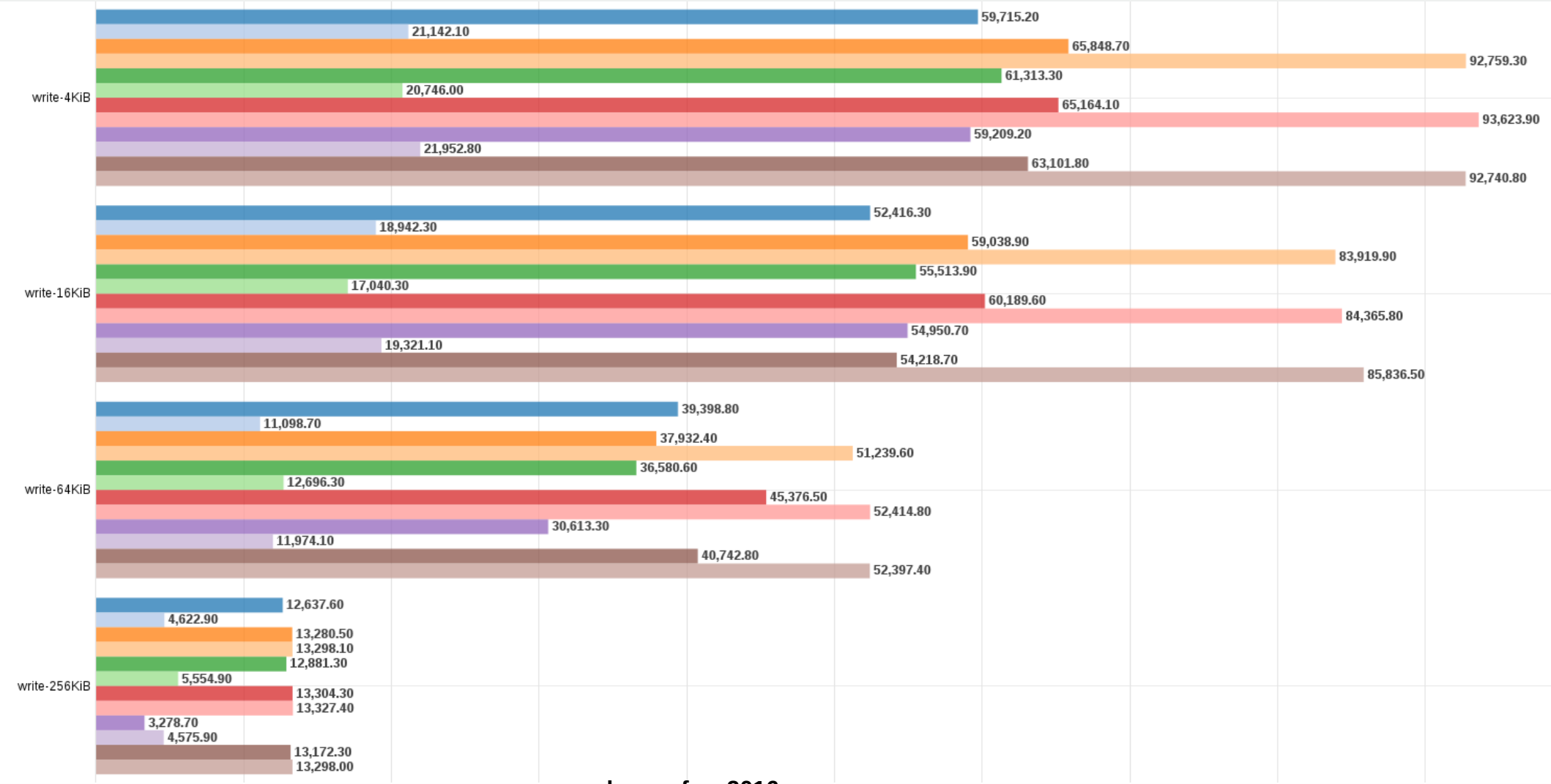


Rand Write



Seq Write

1 FS: EXT4, aio=native, Img: qcow2
2 FS: EXT4, aio=threads, Img: qcow2
3 FS: XFS, aio=native, Img: qcow2
4 FS: XFS, aio=threads, Img: qcow2
5 FS: EXT4, aio=native, Img: qcow2, Prealloc: Falloc
6 FS: EXT4, aio=threads, Img: qcow2, Prealloc: Falloc
7 FS: XFS, aio=native, Img: qcow2, Prealloc: Falloc
8 FS: XFS, aio=threads, Img: qcow2, Prealloc: Falloc
9 FS: EXT4, aio=native, Img: qcow2, Prealloc: Fallocate
10 FS: EXT4, aio=threads, Img: qcow2, Prealloc: Fallocate
11 FS: XFS, aio=native, Img: qcow2, Prealloc: Fallocate
12 FS: XFS, aio=threads, Img: qcow2, Prealloc: Fallocate



8. Disk: SSD, Image: raw, NFS: yes, VMs: 1

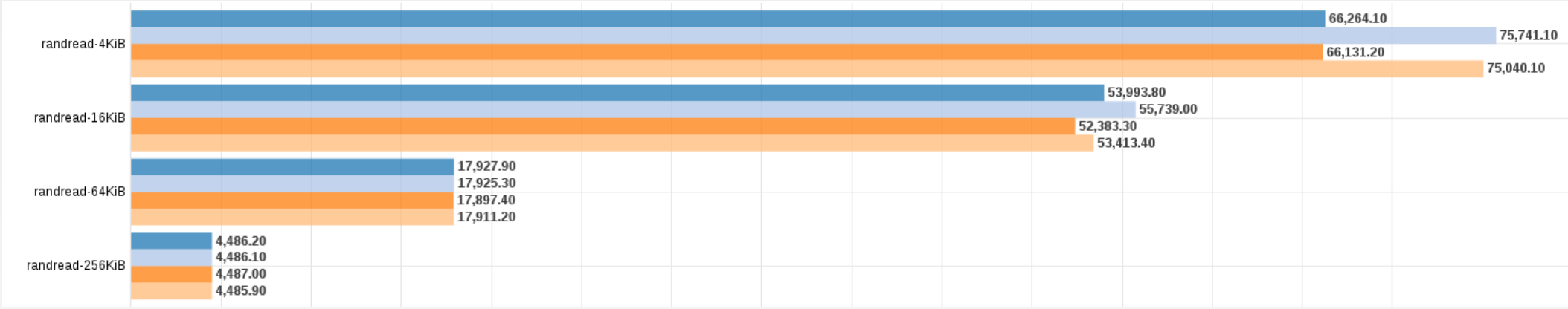
1.FS: EXT4, **aio**=native

2. FS: EXT4, **aio**=threads

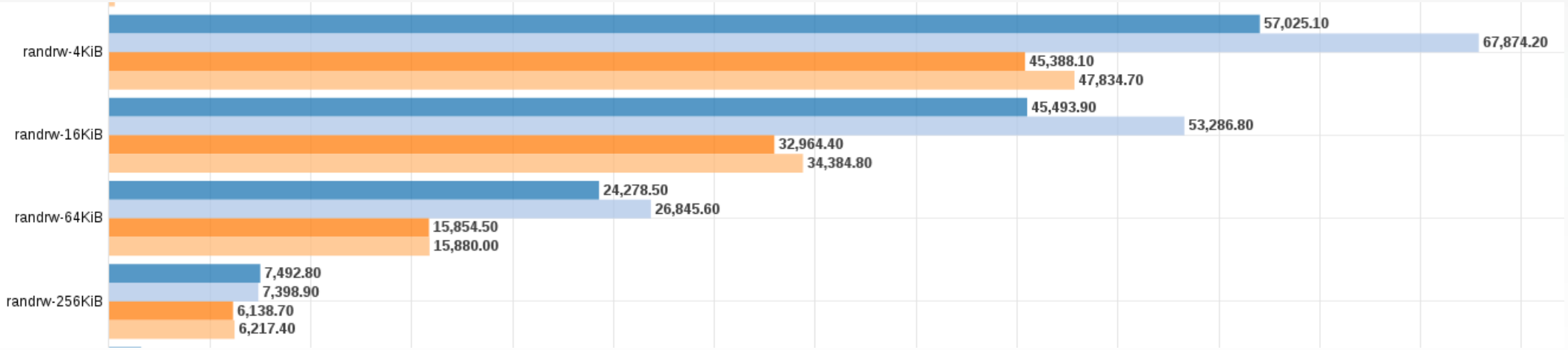
3. FS: XFS, **aio**=native

4. FS: **XFS, aio=**threads

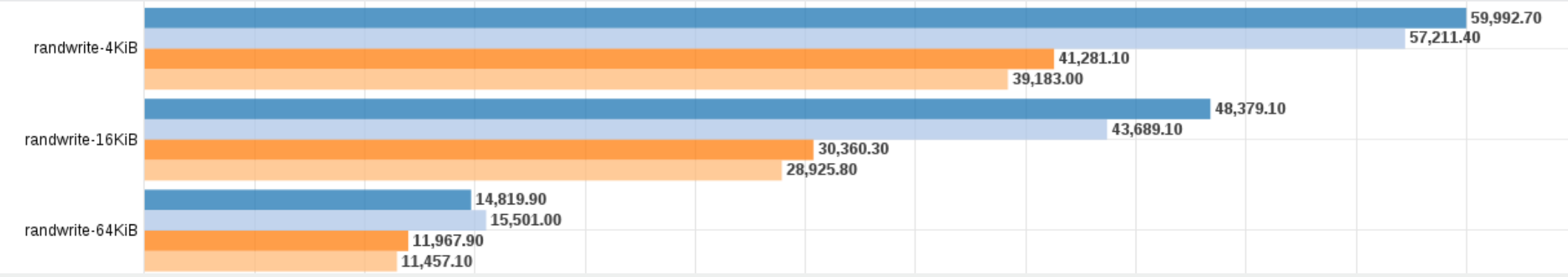
Rand Read



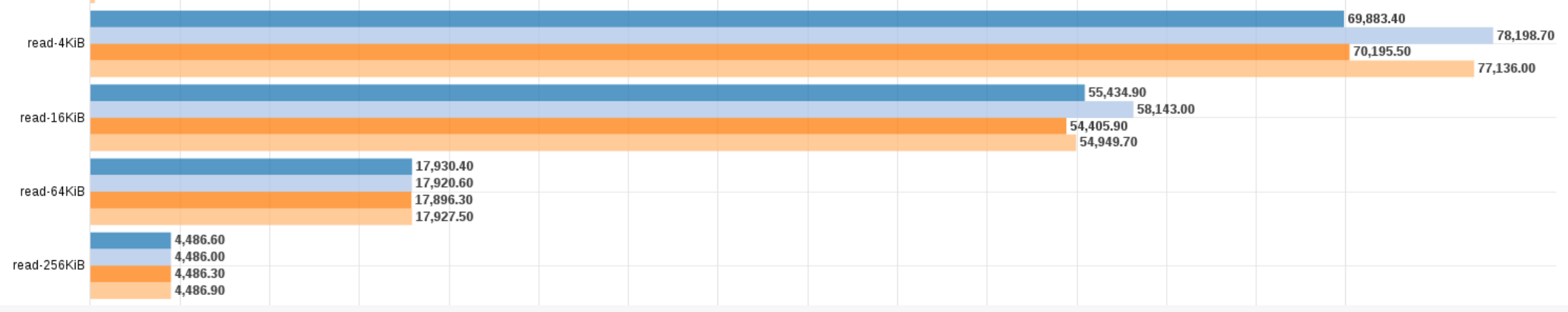
Rand Read Write



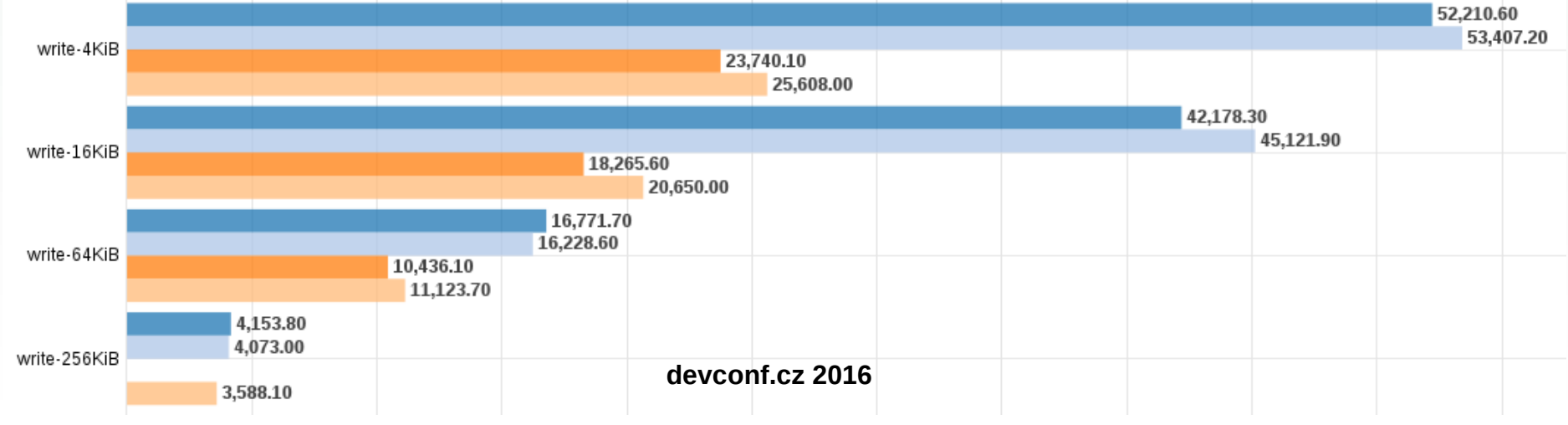
Rand Write



Seq Read



Seq Write



https://review.openstack.org/#/c/232514/7/specs/mitaka/approved/libvirt-aio-mode.rst,cm

Performance Brief

- https://access.redhat.com/articles/2147661

Conclusion & Limitations

- Throughput increased a lot because IO thread takes fewer CPU to submit I/O
- AIO=Native is Preferable choice with few limitations.
- Native AIO can block the VM if the file is not fully allocated and is therefore not recommended for use on sparse files.
- Writes to sparsely allocated files are more likely to block than fully preallocated files. Therefore it is recommended to only use aio=native on fully preallocated files, local disks, or logical volumes.

Future work

Evaluate Virtio Data Plane Performance
Reduce cpu utilization for aio=threads and consider

Questions

References

- Stefan Hajnoczi Optimizing the QEMU Storage Stack, Linux Plumbers 2010
- Asias He, Virtio-blk Performance Improvement, KVM forum 2012
- Khoa Huynch: Exploiting The Latest KVM Features For Optimized Virtualized Enterprise Storage Performance, LinuxCon2012
- Pbench: http://distributed-system-analysis.github.io/pbench/

  https://github.com/distributed-system-analysis/pbench
- FIO: https://github.com/axboe/fio/

Special Thanks
to
Andrew Theurer
Stefan Hajnoczj

Thanks

Irc: #psuriset

Blog: psuriset.com