NUTANIX

Storage Performance Review *for Hypervisors*

Dr Felipe Franciosi

AHV Engineering Lead

OCTOBER 2019 | KVM FORUM LYON

Disclaimer

This presentation and the accompanying oral commentary may include express and implied forward-looking statements, including but not limited to statements concerning our business plans and objectives, product features and technology that are under development or in process and capabilities of such product features and technology, our plans to introduce product features in future releases, the implementation of our products on additional hardware platforms, strategic partnerships that are in process, product performance, competitive position, industry environment, and potential market opportunities. These forward-looking statements are not historical facts, and instead are based on our current expectations, estimates, opinions and beliefs. The accuracy of such forward-looking statements depends upon future events, and involves risks, uncertainties and other factors beyond our control that may cause these statements to be inaccurate and cause our actual results, performance or achievements to differ materially and adversely from those anticipated or implied by such statements, including, among others: failure to develop, or unexpected difficulties or delays in developing, new product features or technology on a timely or cost-effective basis; delays in or lack of customer or market acceptance of our new product features or technology; the failure of our software to interoperate on different hardware platforms; failure to form, or delays in the formation of, new strategic partnerships and the possibility that we may not receive anticipated results from forming such strategic partnerships; the introduction, or acceleration of adoption of, competing solutions, including public cloud infrastructure; a shift in industry or competitive dynamics or customer demand; and other risks detailed in our Form 10-Q for the fiscal quarter ended April 30, 2017, filed with the Securities and Exchange Commission. These forward- looking statements speak only as of the date of this presentation and, except as required by law, we assume no obligation to update forward- looking statements to reflect actual results or subsequent events or circumstances. Any future product or roadmap information is intended to outline general product directions, and is not a commitment, promise or legal obligation for Nutanix to deliver any material, code, or functionality. This information should not be used when making a purchasing decision. Further, note that Nutanix has made no determination as to if separate fees will be charged for any future product enhancements or functionality which may ultimately be made available. Nutanix may, in its own discretion, choose to charge separate fees for the delivery of any product enhancements or functionality which are ultimately made available.

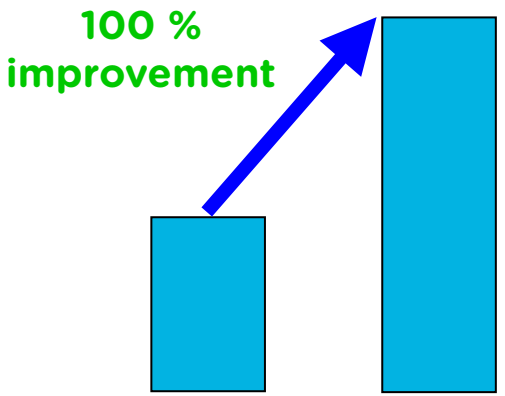
Certain information contained in this presentation and the accompanying oral commentary may relate to or be based on studies, publications, surveys and other data obtained from third-party sources and our own internal estimates and research. While we believe these third-party studies, publications, surveys and other data are reliable as of the date of this presentation, they have not independently verified, and we make no representation as to the adequacy, fairness, accuracy, or completeness of any information obtained from third-party sources.

Performance Overhead

Performance Overhead

-25%

5 X
FASTER



Storage Performance Overhead

Depends on:

- Direction of I/O (read / write)
- Size of I/O (..., 4K, ..., 1M, ...)
- Sequence of I/O (sequential / random)
- Queue Depth (# of I/Os outstanding)
  - Batched I/Os
  - Number of threads
  - QD per thread
- Handling Fairness (multiple sources)
- Submission interface (libaio, io_uring, spdk)
- I/O Duration (sustained performance)
- Temperature [1]
- Noise / Vibration [2]
- Interrupt distribution
  - Depending on CPU utilisation
- Cache contention (O_DIRECT)
- NUMAness
- Lock contention
- Backend (Null / HDD / SSD / Net)

[1] "... why some (might) like it hot" dl.acm.org/citation.cfm?id=2254778

[2] "Shouting in the Datacenter" youtu.be/tDacjrSCeq4

Agenda

1 Measuring Storage Performance

2 Analysing Virtualisation Overhead
(and a brief Hypervisor Analysis)

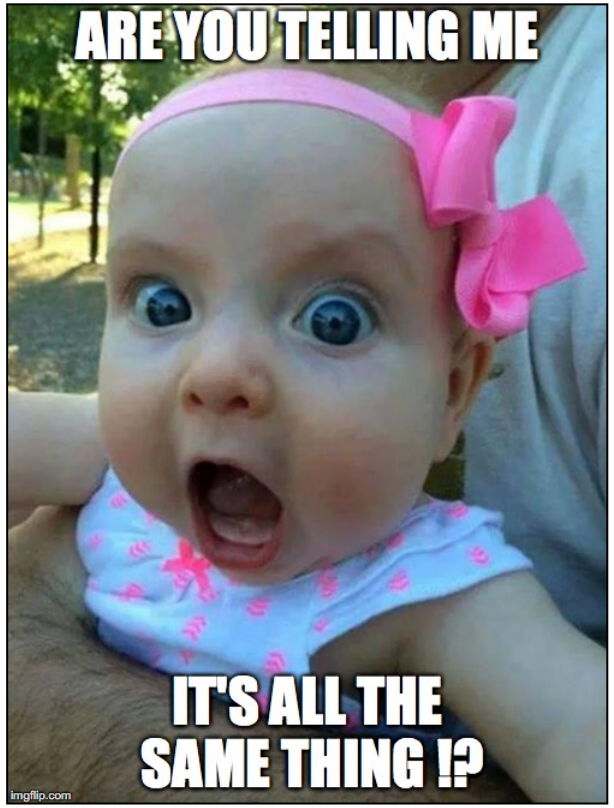
3 Conclusions, Thoughts, and Extras

Measuring Storage Performance

Measuring Storage Performance

What are we really measuring?

- Bandwidth or Throughput (MB/s)
- IOPS (reqs/s)
- Latency (us)



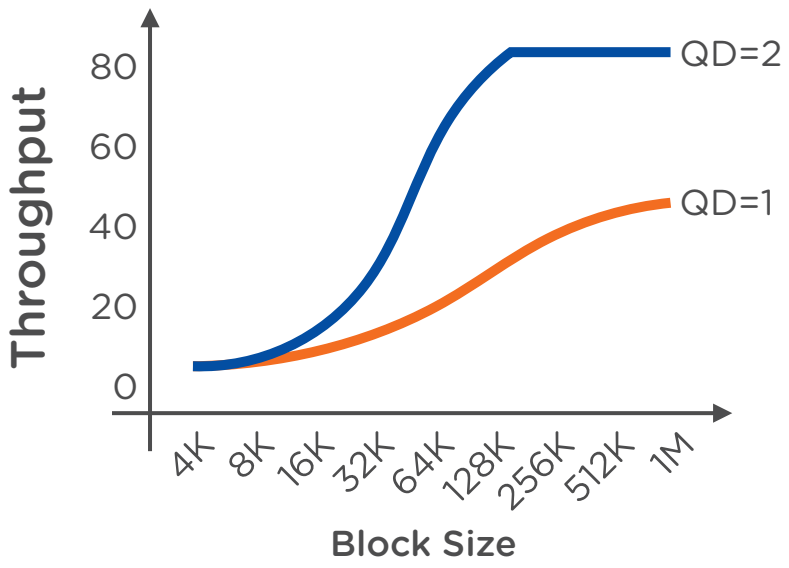
(MB/s) $\frac{\text{data}}{\text{time}}$

(reqs/s) $\frac{\text{data}}{\text{time}}$

(us/req) $\frac{\text{time}}{\text{data}}$

Measuring Storage Performance

Measuring Throughput

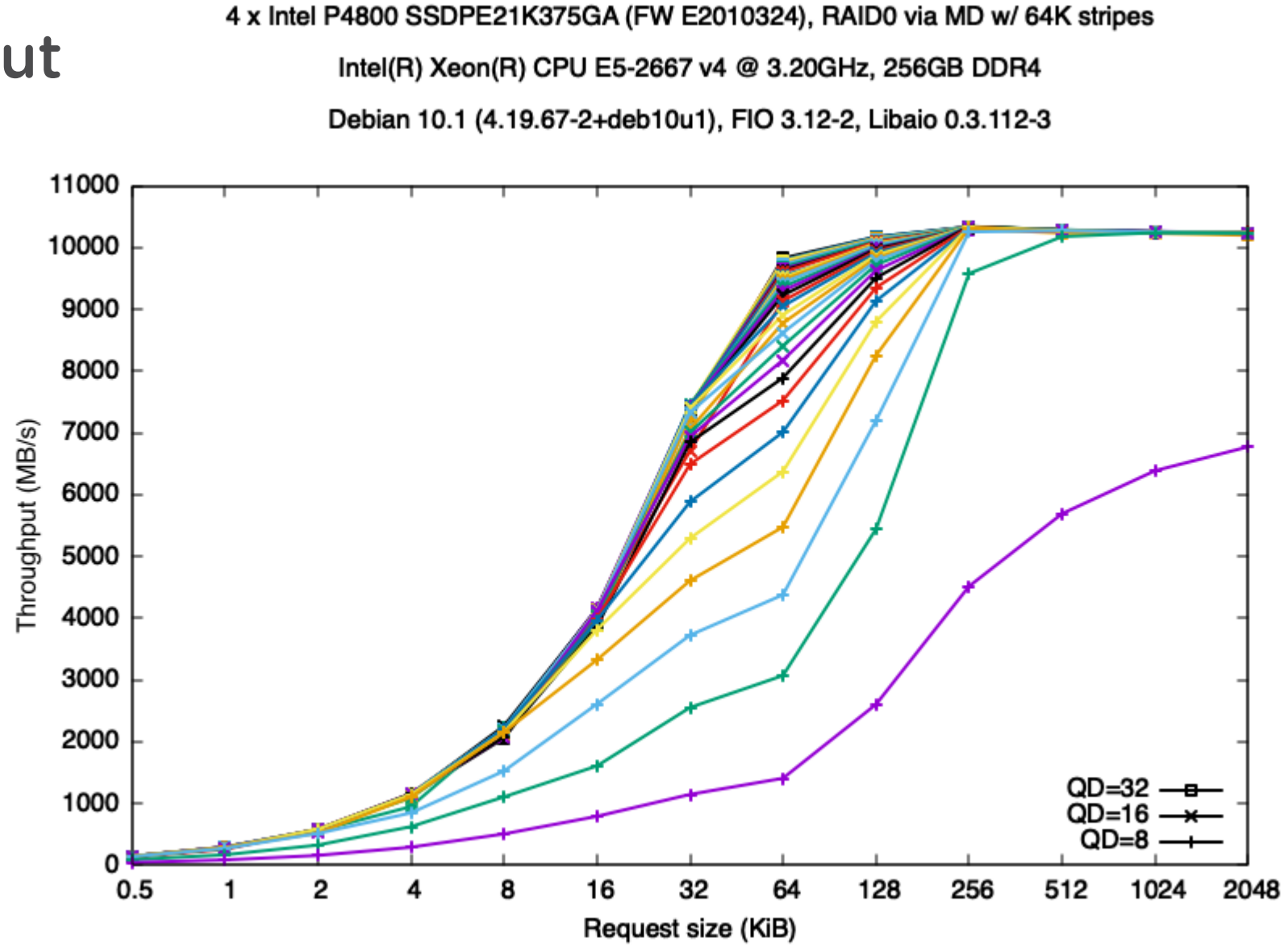


- Usually associated with sequential I/O (and large requests)
  - When access pattern is known, data can be transferred in bulk
  - Transferring contiguous large datasets favours HDD
- Modern SSDs are not saturated with QD 1
  - Worth measuring higher QDs, in steps of 1
  - Doesn't *normally* saturate CPUs, but it can
  - Pinning should be observed to avoid NUMAness
- Plotting the graph
  - Y-Axis linear scale (MB/s)
  - X-Axis log scale (KiB)
  - Series varying QD (1 or more CPUs)

Measuring Storage Performance

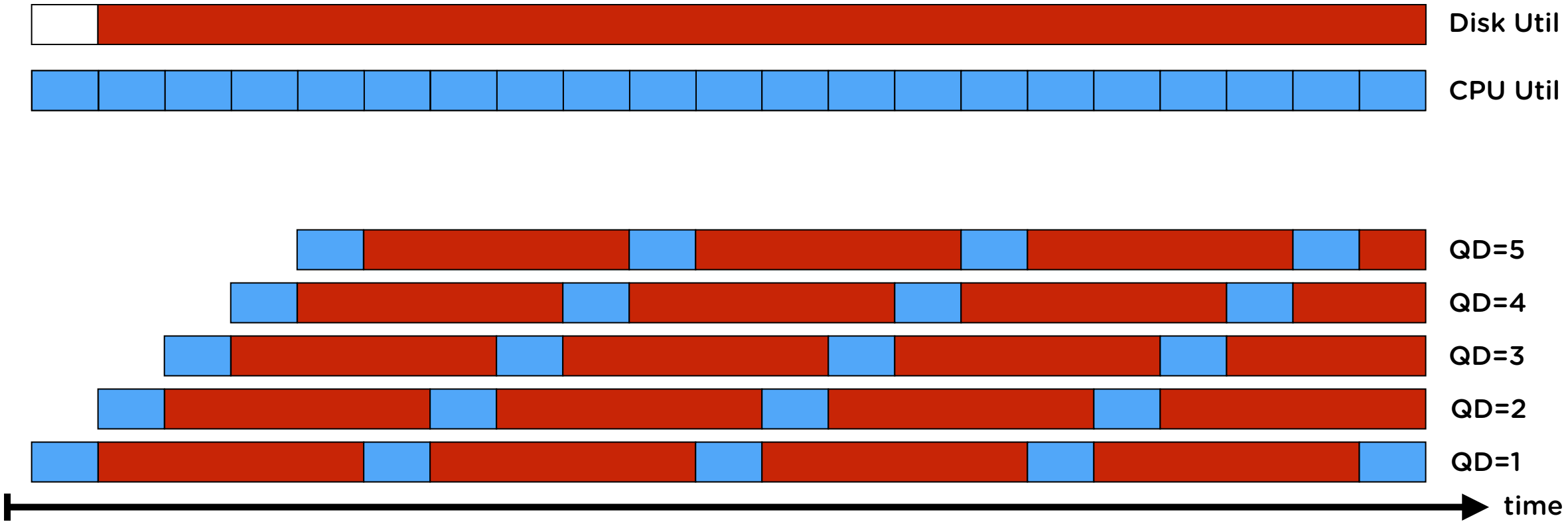
Measuring Throughput

- Direction: Read
- Sequence: Random
- Interface: libaio
- Req size: 512 B ... 2 MiB
- Num threads: 1
  - QD/thread: 1 ... 32
- Thread pinned to CPU 8 (in drive's NUMA node)



Measuring Storage Performance

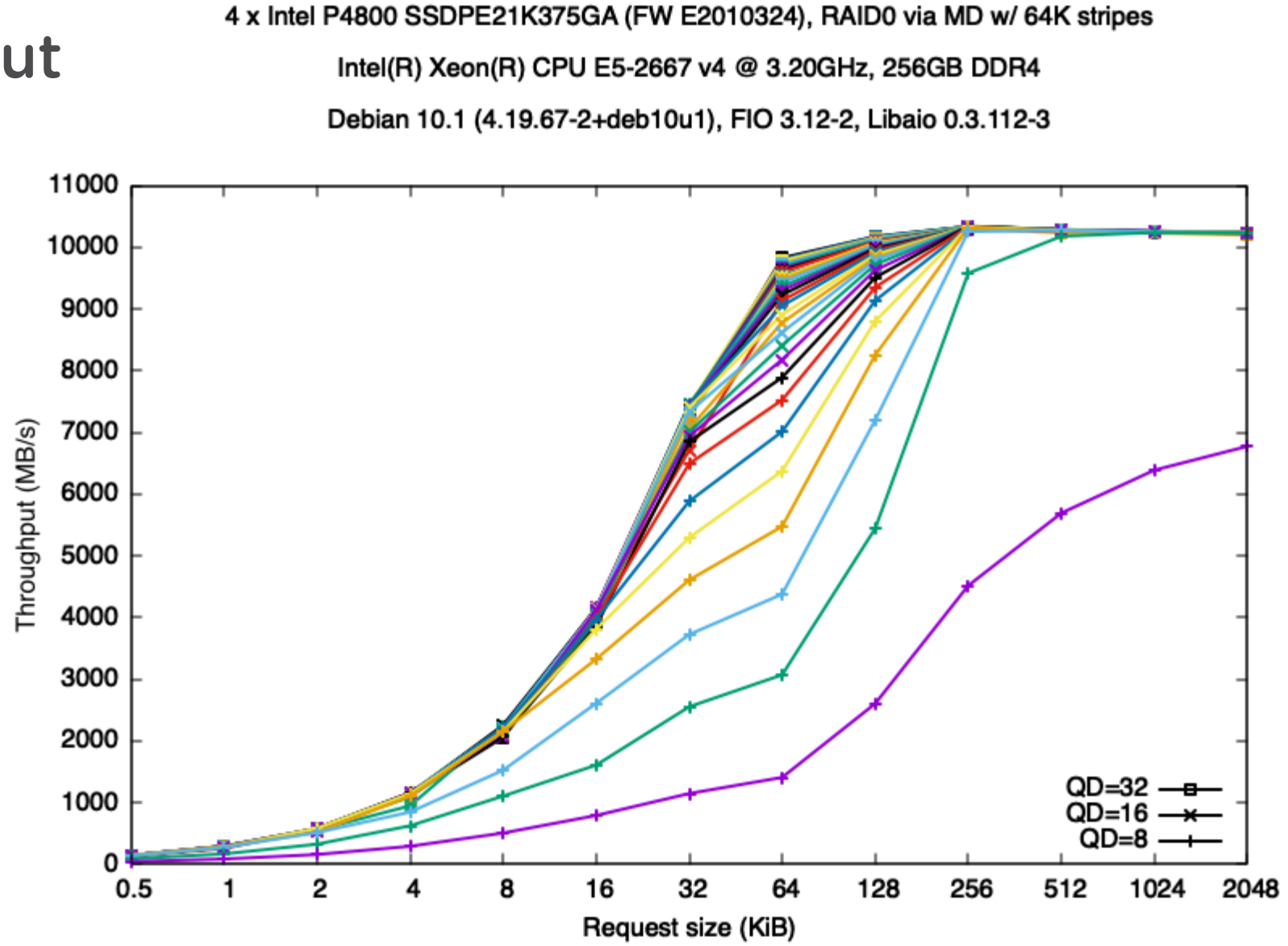
Saturating CPUs and NVMe (when CPU usage is inefficient)



Measuring Storage Performance

Measuring Throughput

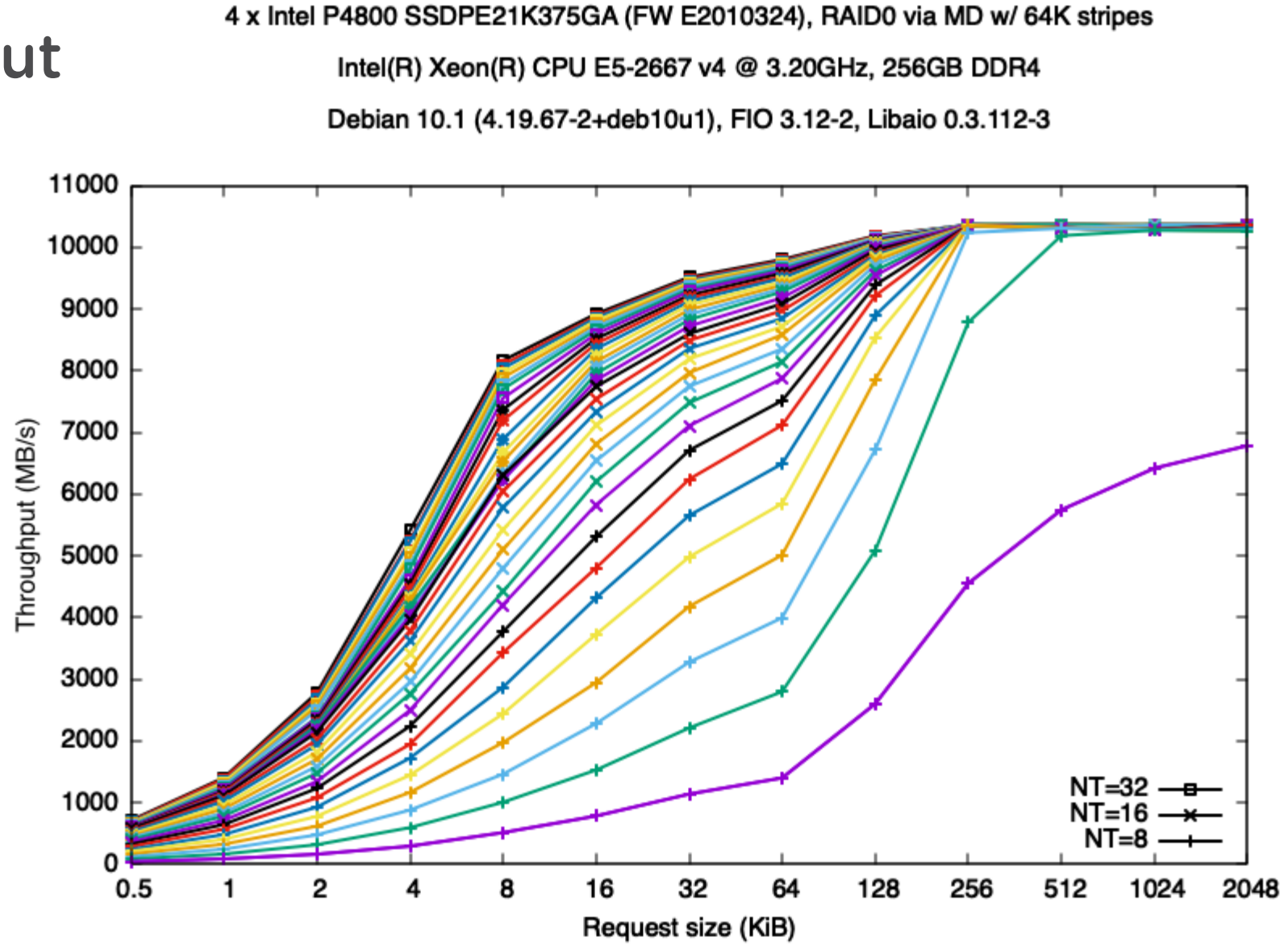
- Direction: Read
- Sequence: Random
- Interface: libaio
- Req size: 512 B ... 2 MiB
- Num threads: 1
  - QD/thread: 1 ... 32
- Thread pinned to CPU 8 (in drive's NUMA node)



Measuring Storage Performance

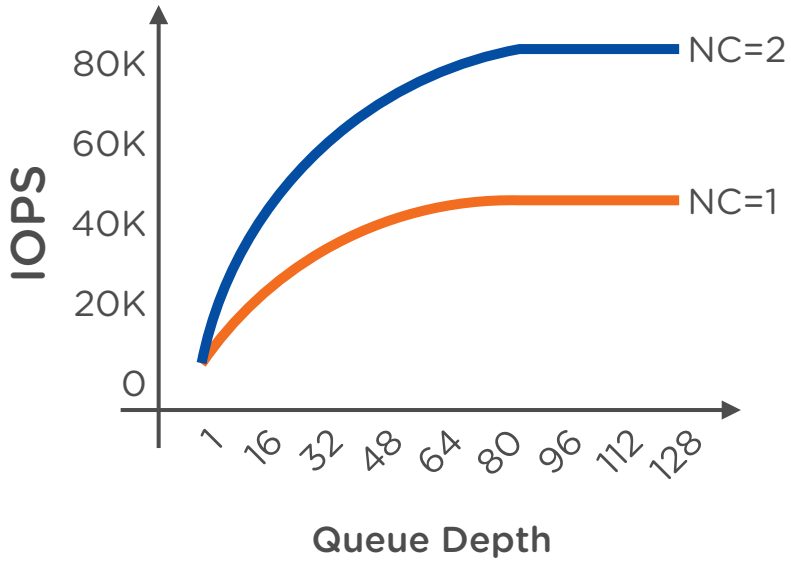
Measuring Throughput

- Direction: Read
- Sequence: Random
- Interface: libaio
- Req size: 512 B ... 2 MiB
- Num threads: 1 ... 32
  - QD/thread: 1
- Threads pinned to CPUs 8-15 (in drive's NUMA node)



Measuring Storage Performance

Measuring IOPS

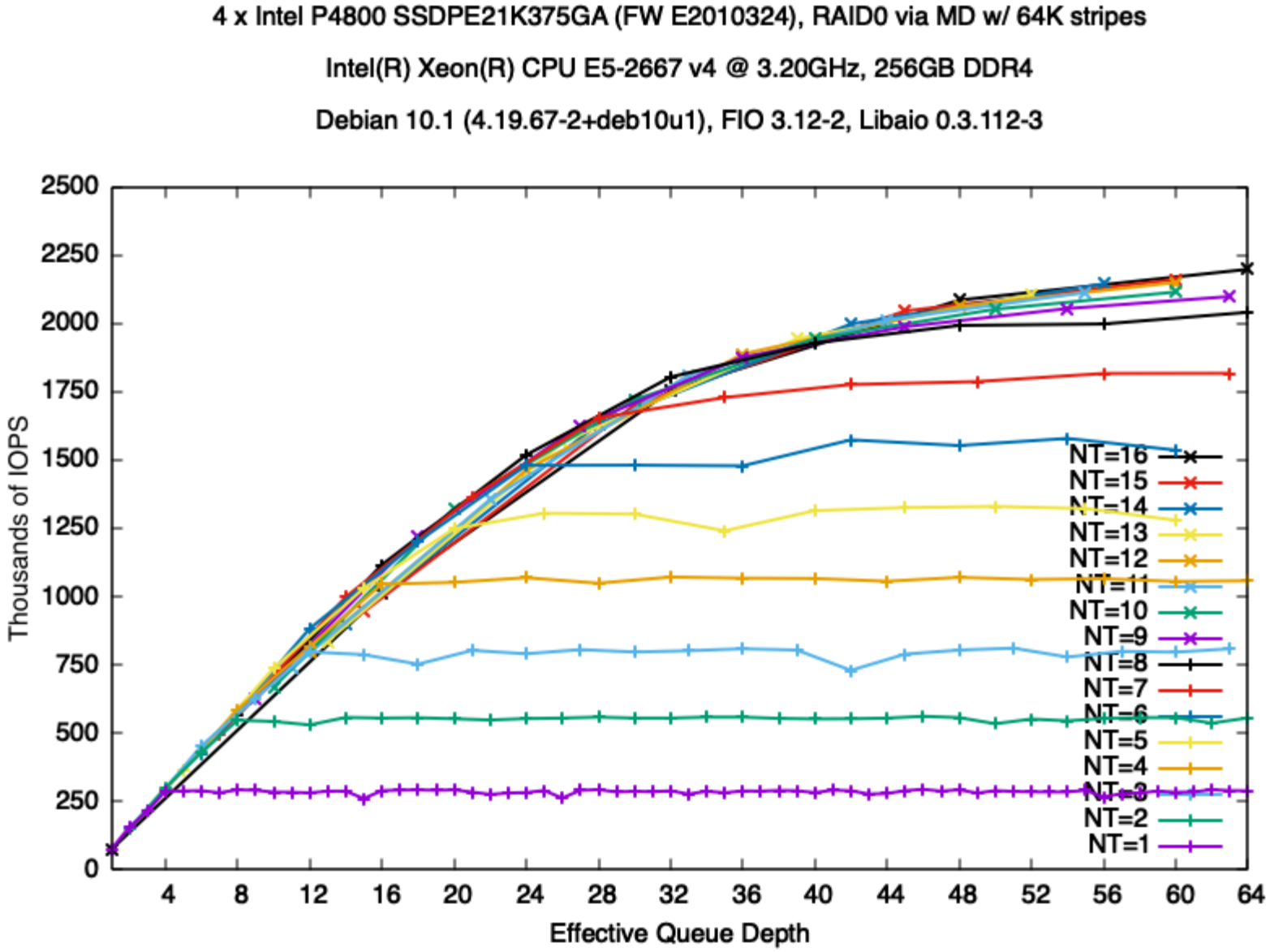


- Usually associated with random I/O (and small requests)
  - When access pattern is unknown, data cannot be transferred in bulk
  - Transferring small datasets randomly disfavours HDD
- Modern SSDs are not saturated with 1 CPU (using kernel + libaio)
  - Worth measuring with more CPUs, in steps of 1
  - Leaner drivers (SPDK, io_uring) are much more efficient
  - Pinning should be observed to avoid NUMAness
- Plotting the graph
  - Y-Axis linear scale (reqs/s)
  - X-Axis linear scale (QD)
  - Series varying number of CPUs (NC)
  - Fixed request size (4 or 8 KiB)

Measuring Storage Performance

Measuring IOPS

- Direction: Read
- Sequence: Random
- Interface: libaio
- Req size: 4 KiB
- Num threads: 1 ... 16
  - QD/thread: 1 ... 64
- Thread(s) pinned to CPUs 8-15 (in drive's NUMA node)



Analysing Virtualisation Overhead

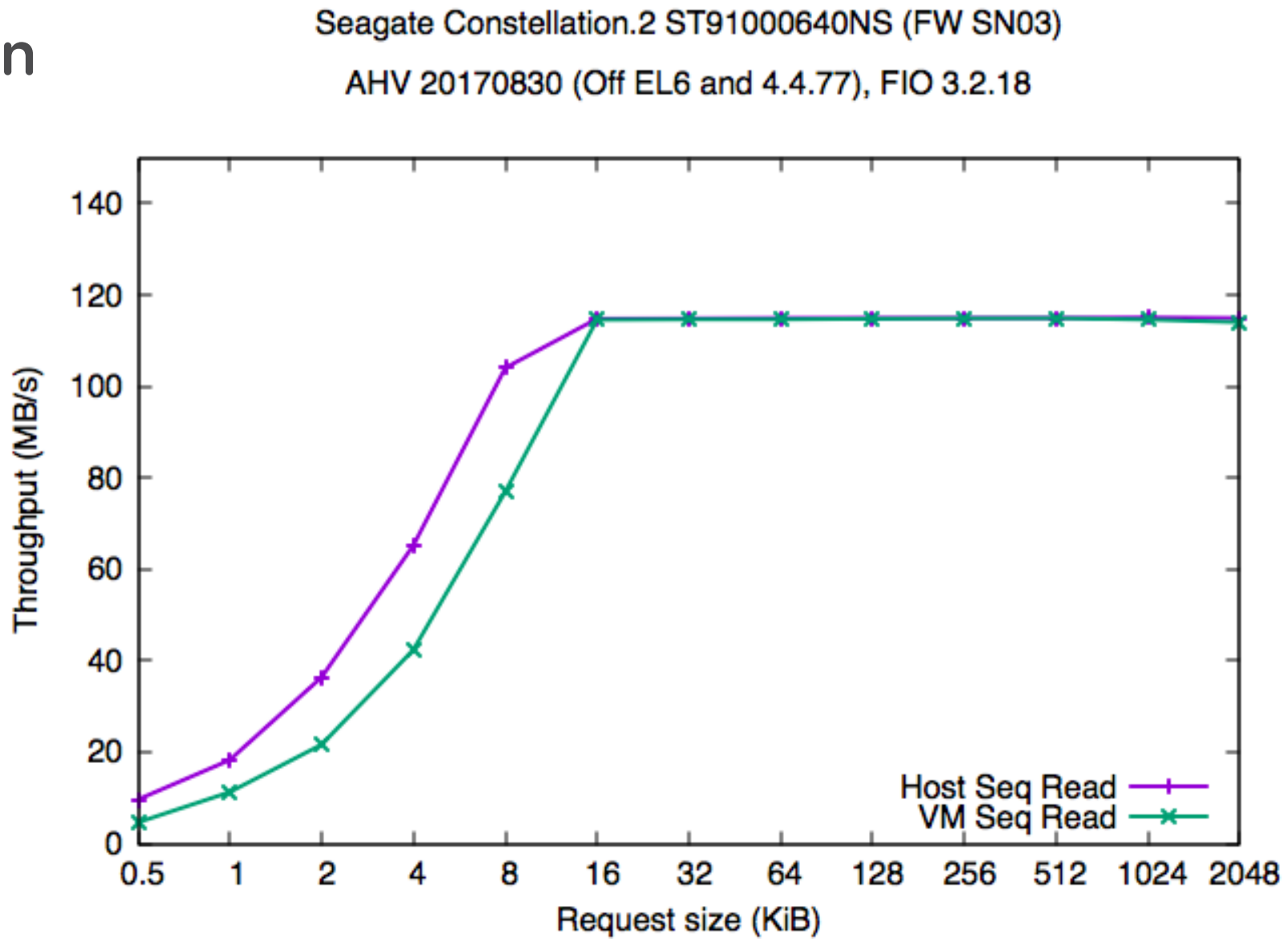
Analysing Virtualisation Overhead

Overhead Illustration

- **Measuring Throughput**
  - Mechanical drive
  - Sequential reads
  - QD = 1
  - Req size: 512 B ... 2 MiB

And from a VM ?

- Debian 9.4 VM (FIO 3.2.18)
- Host with Qemu 2.6
- Disk over virtio-scsi

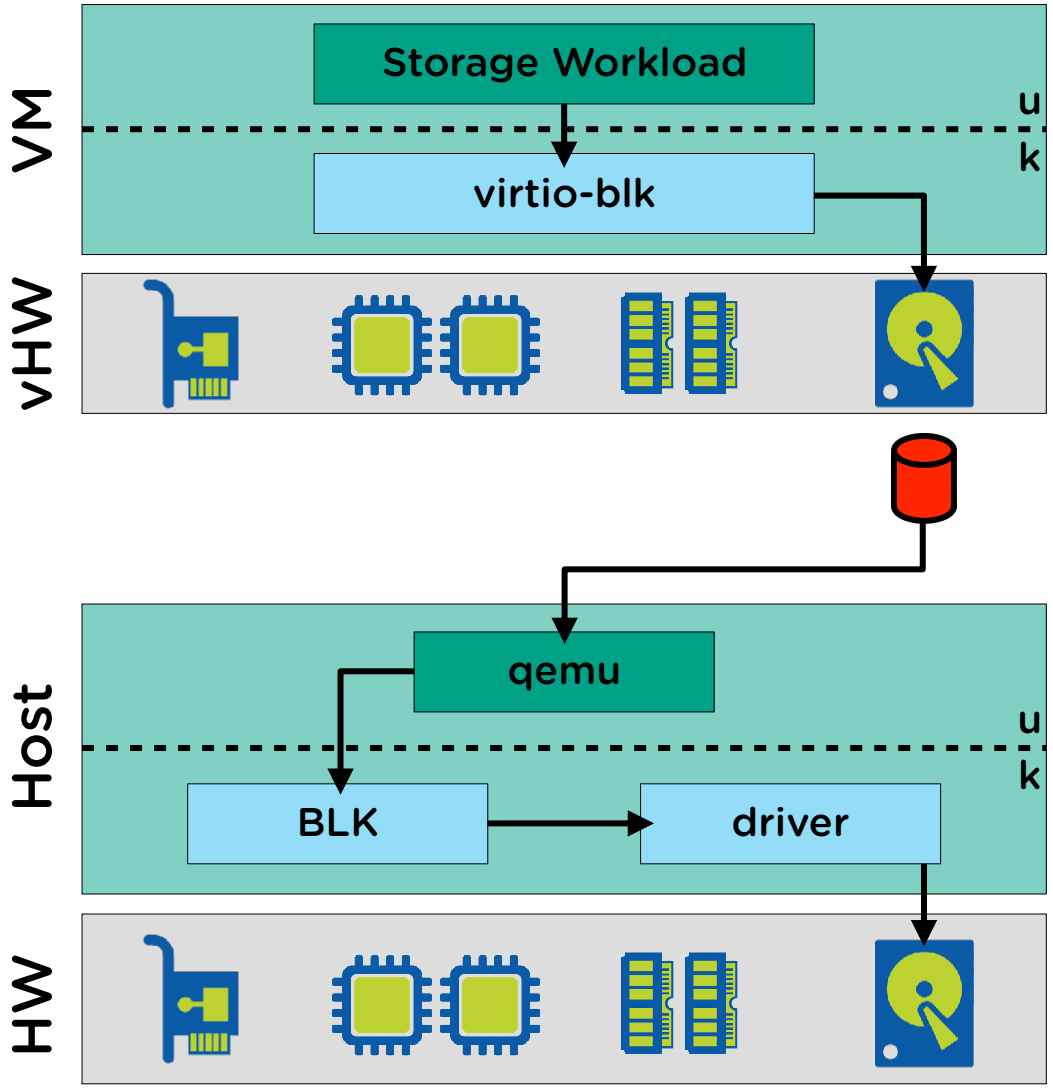


Analysing KVM + Qemu

1 VM on Qemu 3.1.0

w/ 1 vDisk on virtio-blk

Hypervisor Analysis: KVM + Qemu



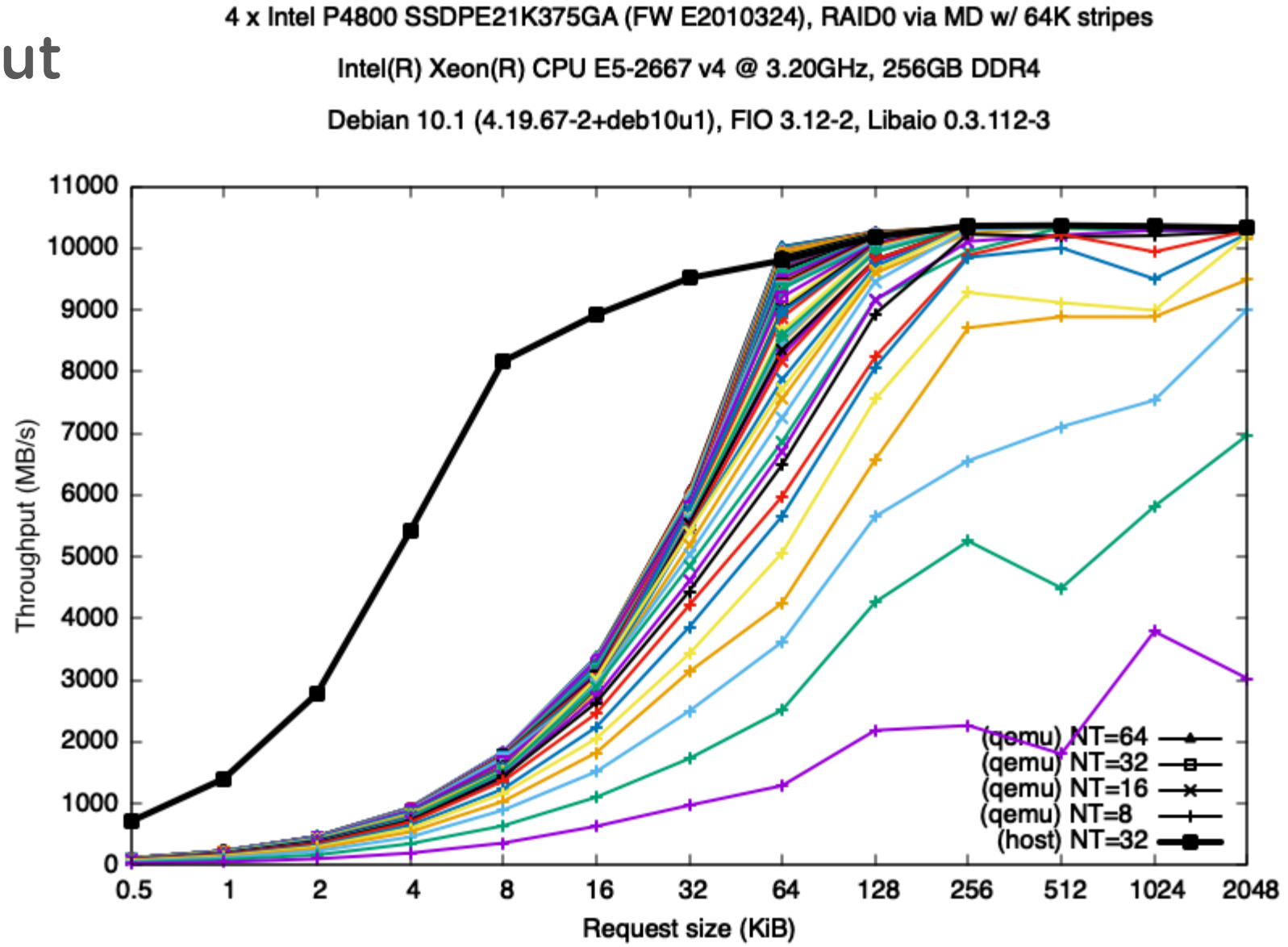
Typical virtio-blk deployment

- One controller per disk presented to VM
- Disks are block devices
- One controller cannot use multiple I/O threads
  - The I/O thread bottlenecks on CPU
  - In order to scale, VM requires more controllers (and more virtual disks and more I/O threads)

Hypervisor Analysis: KVM + Qemu

Measuring Throughput

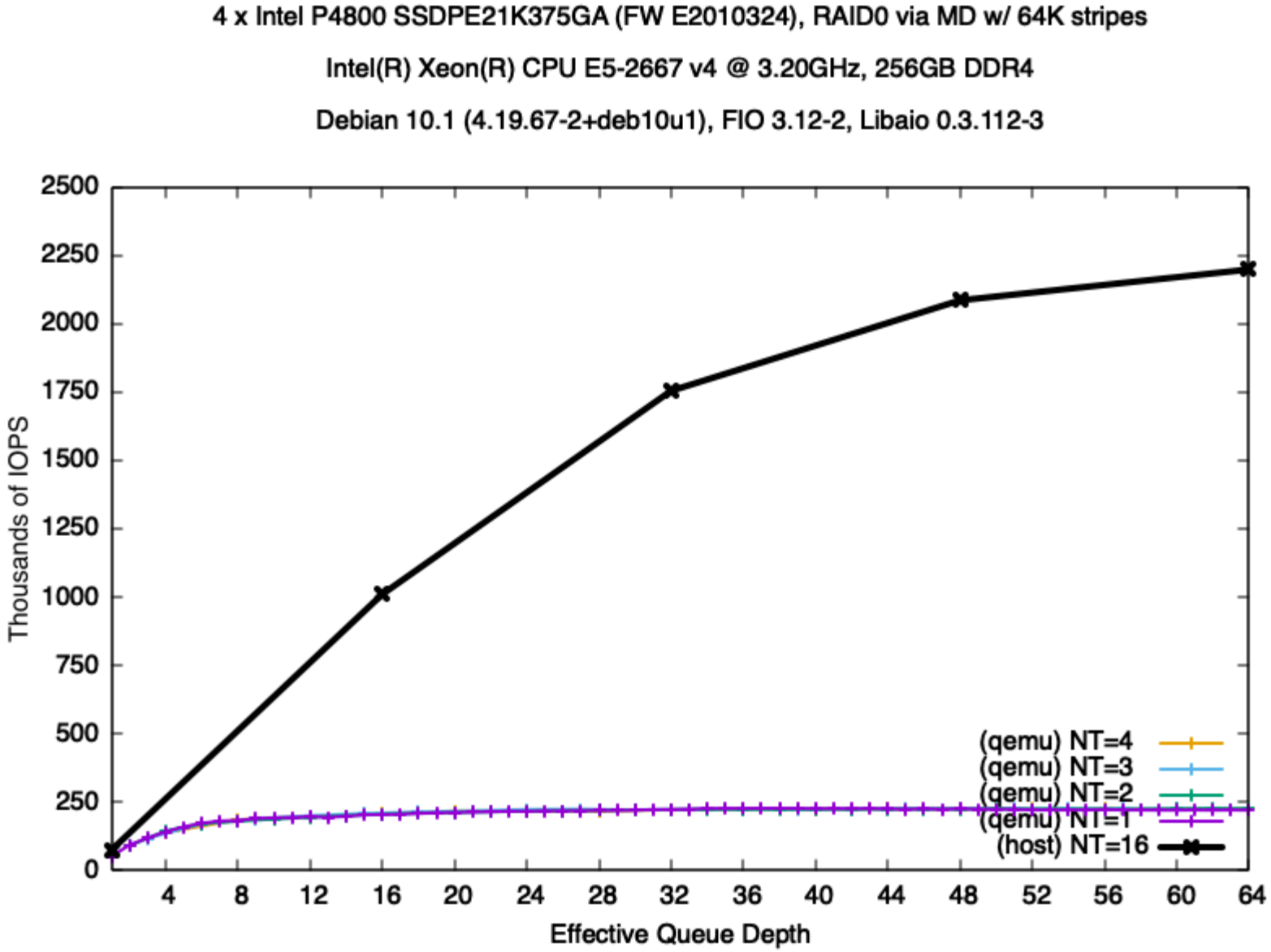
- Direction: Read
- Sequence: Random
- Interface: libaio
- Req size: 512 B ... 2 MiB
- Num threads: 32
  - QD/thread: 1
- Thread(s) pinned to CPUs 8-15 (in drive's NUMA node)
- VM (Qemu 3.1.0, virtio-blk)
  - Num threads: 1 ... 64
  - QD/thread: 1
  - VM pinned to CPUs 8-15



Hypervisor Analysis: KVM + Qemu

Measuring IOPS

- Direction: Read
- Sequence: Random
- Interface: libaio
- Req size: 4 KiB
- Num threads: 16
  - QD/thread: 1 ... 64
- Threads pinned to CPUs 8-15 (in drive's NUMA node)
- VM (Qemu 3.1.0, virtio-blk)
  - Num threads: 1 ... 4
  - QD/thread: 1 ... 64
  - VM pinned to CPUs 8-15

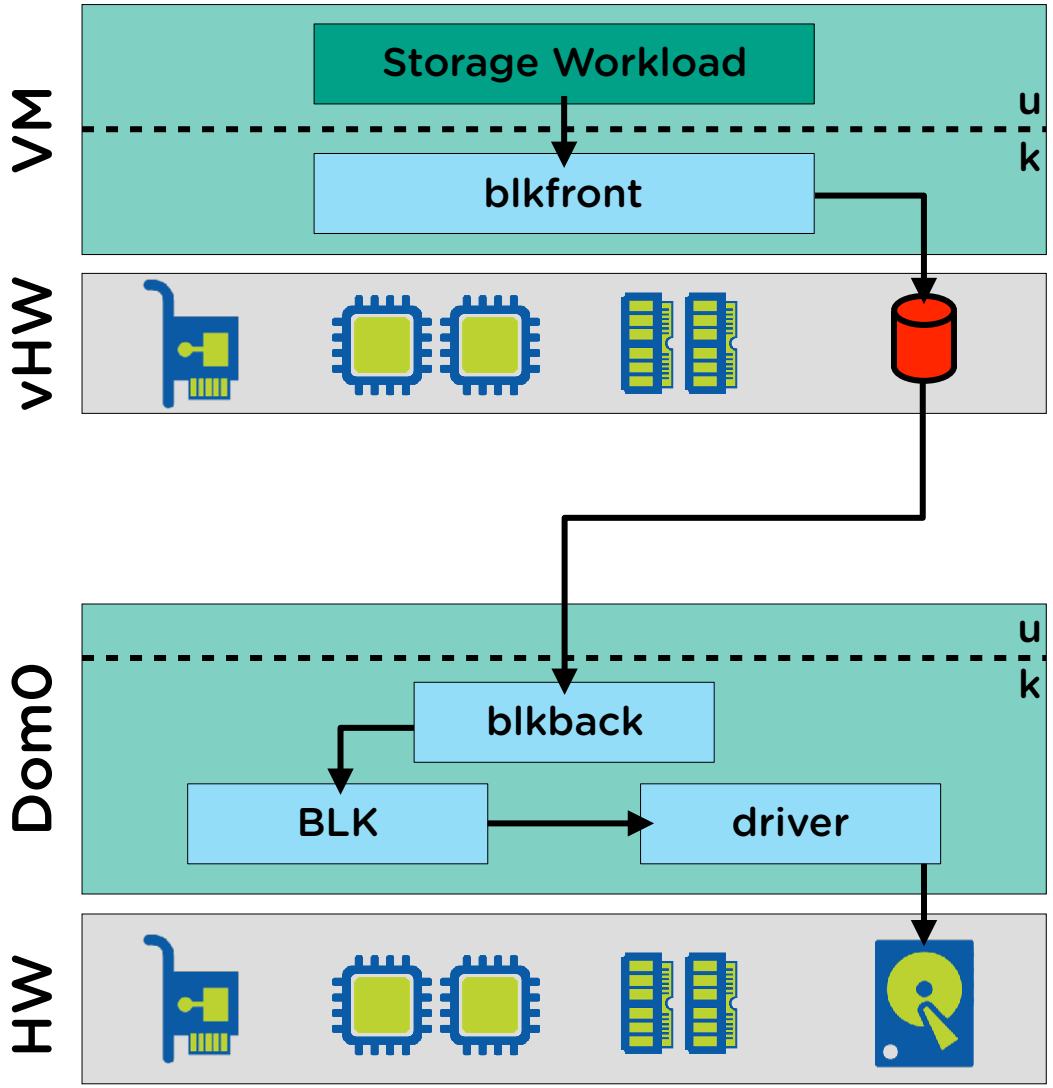


Analysing Xen + Blkback

1 VM on Qemu 3.1.0

1 vDisk on Blkfront/Blkback

Hypervisor Analysis: Xen + Blkback



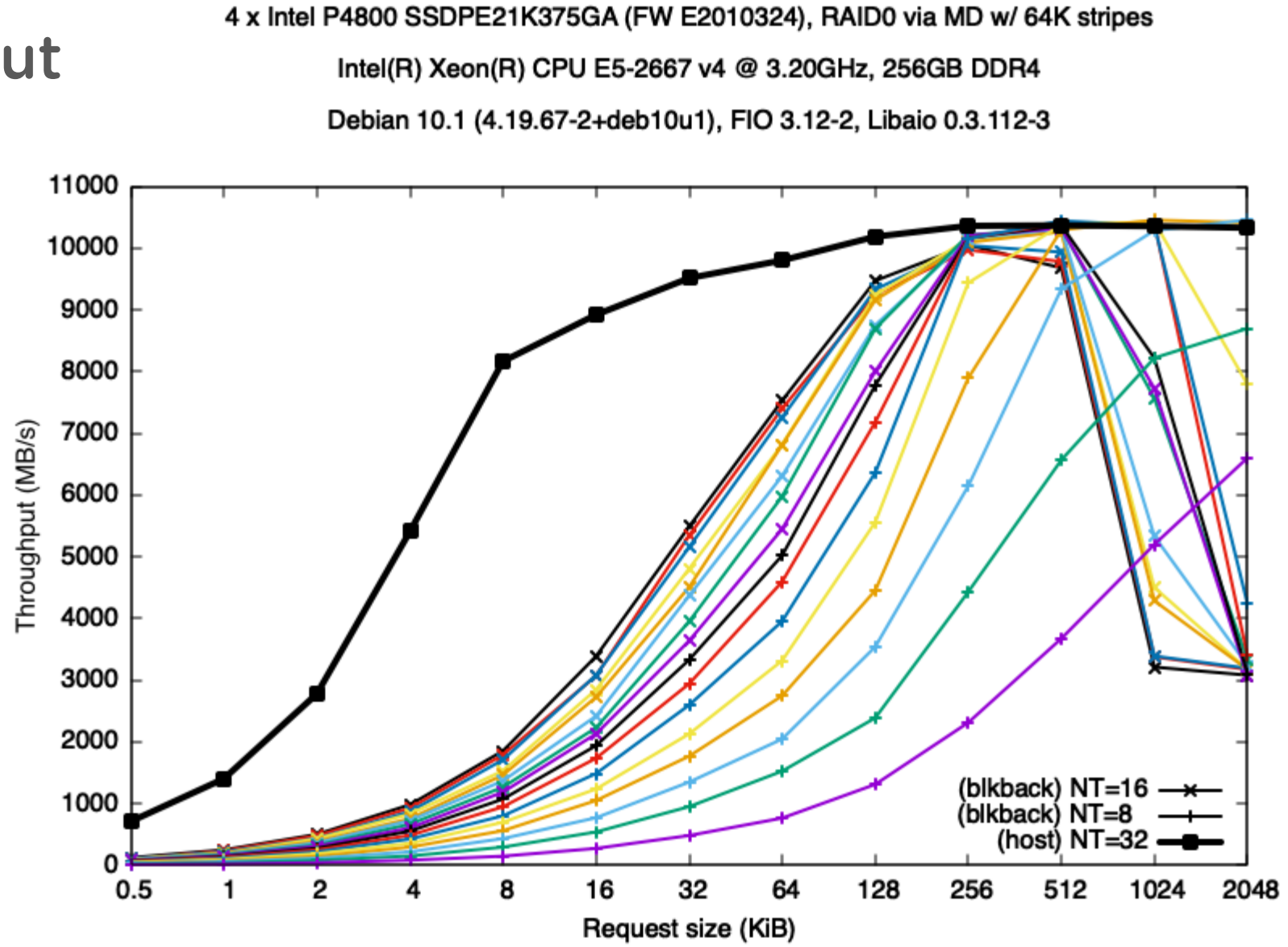
Typical blkback deployment

- Disks are block devices on XenBus
- Blkback has a queue for each guest vCPU
  - This allows performance to scale with VM size
  - But it eats a lot of CPU on Domain 0
- The Xen block ring interface has design deficiencies
  - Limitations on the number of outstanding requests
    - Multi-page rings can help
  - Limitations on the request size
    - Indirect descriptors can help
- Xen requires memory to granted by frontend and mapped by backend. This contributes to overhead.
  - Persistent grants can help

Hypervisor Analysis: Xen + Blkback

Measuring Throughput

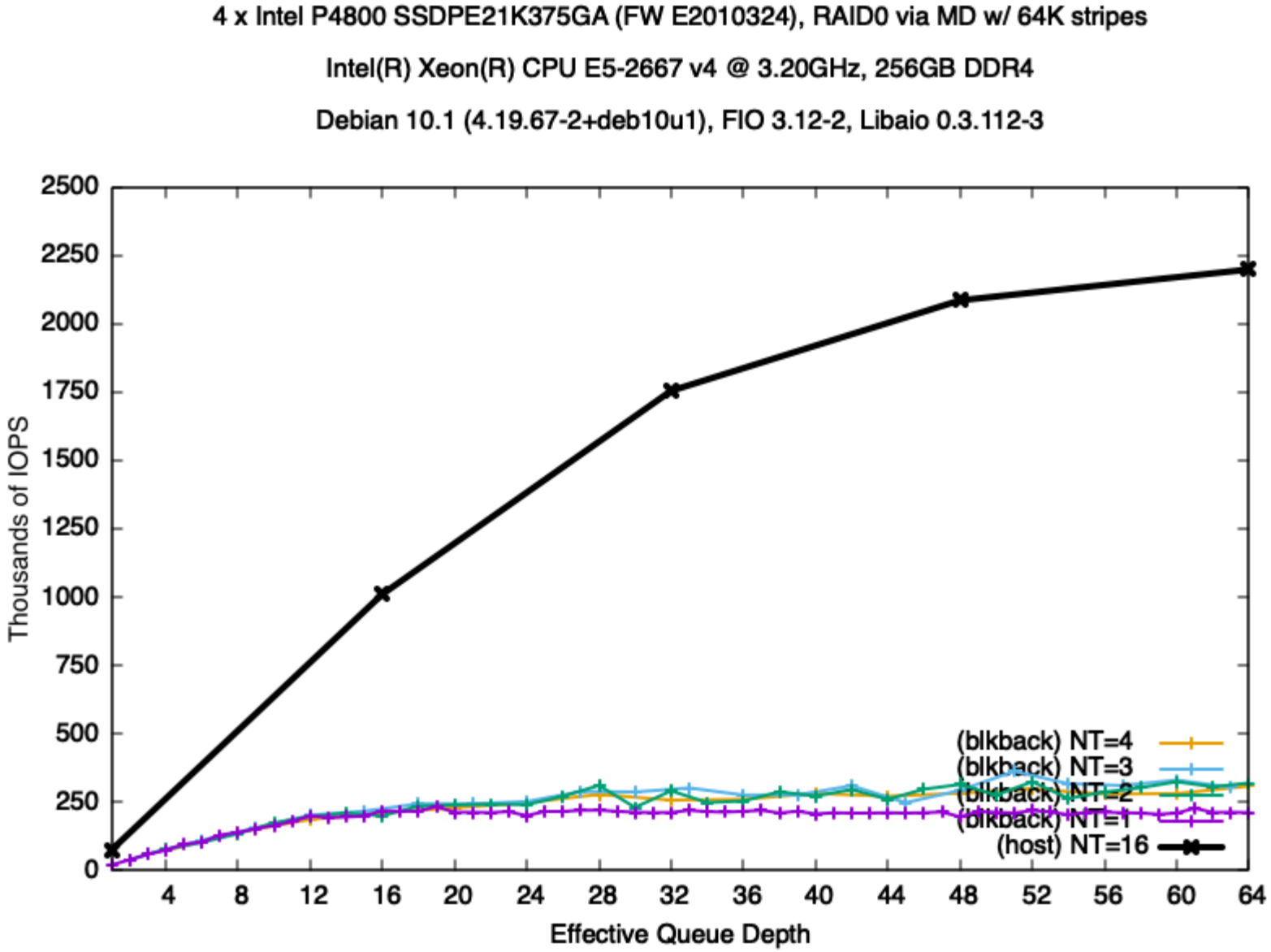
- Direction: Read
- Sequence: Random
- Interface: libaio
- Req size: 512 B ... 2 MiB
- Num threads: 1 ... 32
  - QD/thread: 1
- Thread(s) pinned to CPUs 8-15 (in drive's NUMA node)
- VM (Qemu 3.1.0, blkback)
  - Num threads: 1 ... 16
  - QD/thread: 1
  - VM pinned to CPUs 8-15



Hypervisor Analysis: Xen + Blkback

Measuring IOPS

- Direction: Read
- Sequence: Random
- Interface: libaio
- Req size: 4 KiB
- Num threads: 16
  - QD/thread: 1 ... 64
- Threads pinned to CPUs 8-15 (in drive's NUMA node)
- VM (Qemu 3.1.0, blkback)
  - Num threads: 1 ... 4
  - QD/thread: 1 ... 64
  - VM pinned to CPUs 8-15



Conclusions, Thoughts, and Extras

Conclusions, Thoughts, and Extras

- Low-latency, high-throughput storage is hard to virtualise
  - Traditional kernel datapaths consume too much CPU
  - Datapath from application to storage must be more **efficient**

- KVM
  - Emulators have direct access to VM memory by default
  - VirtIO + MQ can work great if emulator is parallel and efficient
  - NVMe is an attractive solution due to industry support on standard

- Xen
  - Is too focused on security, not allowing VM memory access by default
  - Xen's ring format should be revisited to align with modern device models
  - Blkback is efficient (entirely in-kernel datapath), but apparently has contention problems

Conclusions, **Thoughts**, and Extras

- How can hypervisors be more efficient?
  - **First thing**: avoid legacy kernel datapaths (ie. libaio, read()/write() syscalls)
    - There are more attractive solutions with io_uring or SPDK
  - **Second thing**: pursue hardware offload or software host polling
    - Hardware offload means that VFs can be passed-through to VMs
    - Host polling means that:
      - One process* handles I/O from all VMs on host
      - This process polls VMs' submission queues in a tight loop
      - VMs don't have to kick the hypervisor (saves on VM EXITs)
      - Hypervisor doesn't require IRQs from devices (saves on ctx switches)
    - Additionally, VMs that care about performance can also poll
      - VMs don't require IRQs from hypervisor (saves on ctx switches)

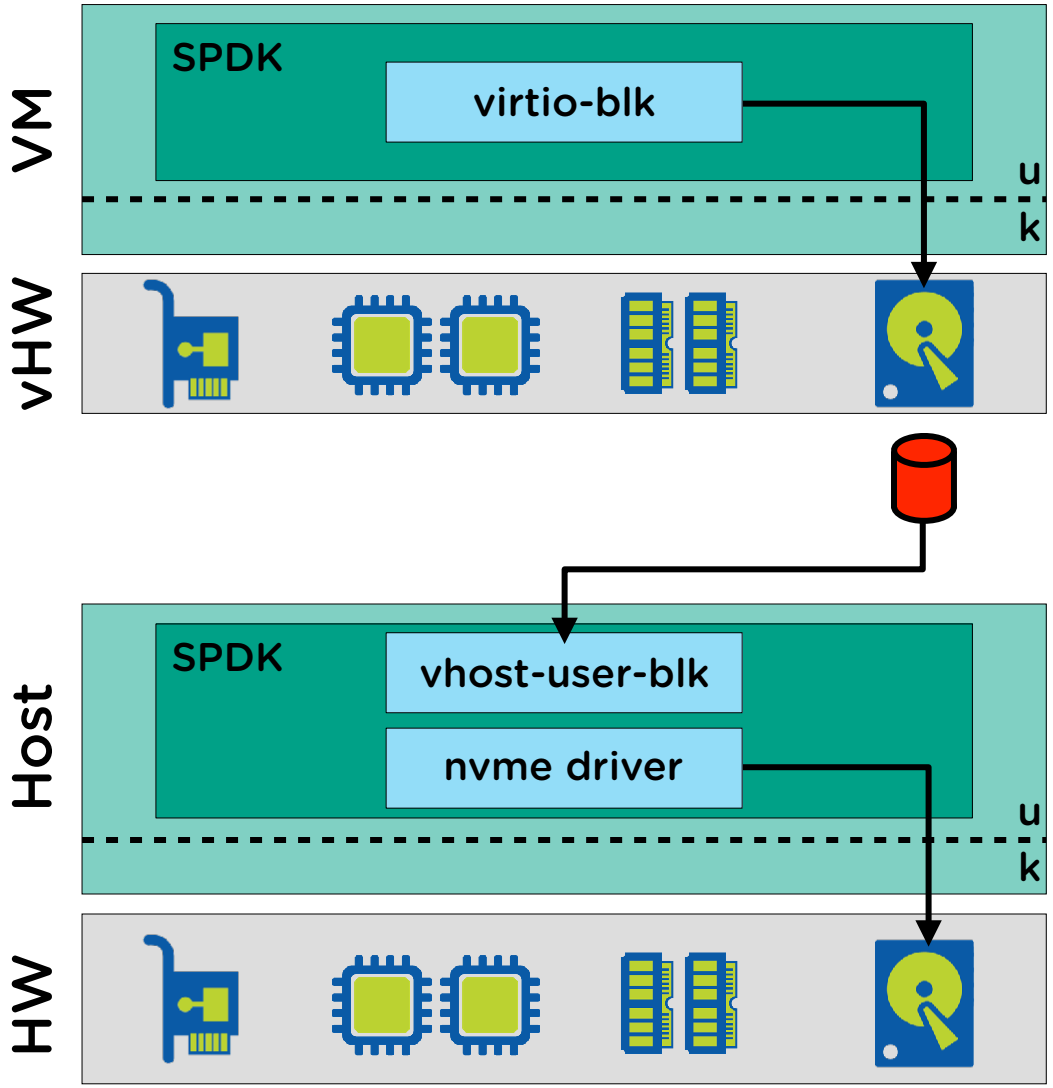
* Multi-tenant hosts (or similar setups) could have a poller per tenant

EXTRAS: Analysing KVM + Qemu

1 VM on Qemu 3.1.0

1 vDisk on SPDK (vhost-user-blk)

Hypervisor Analysis: KVM + Qemu + SPDK



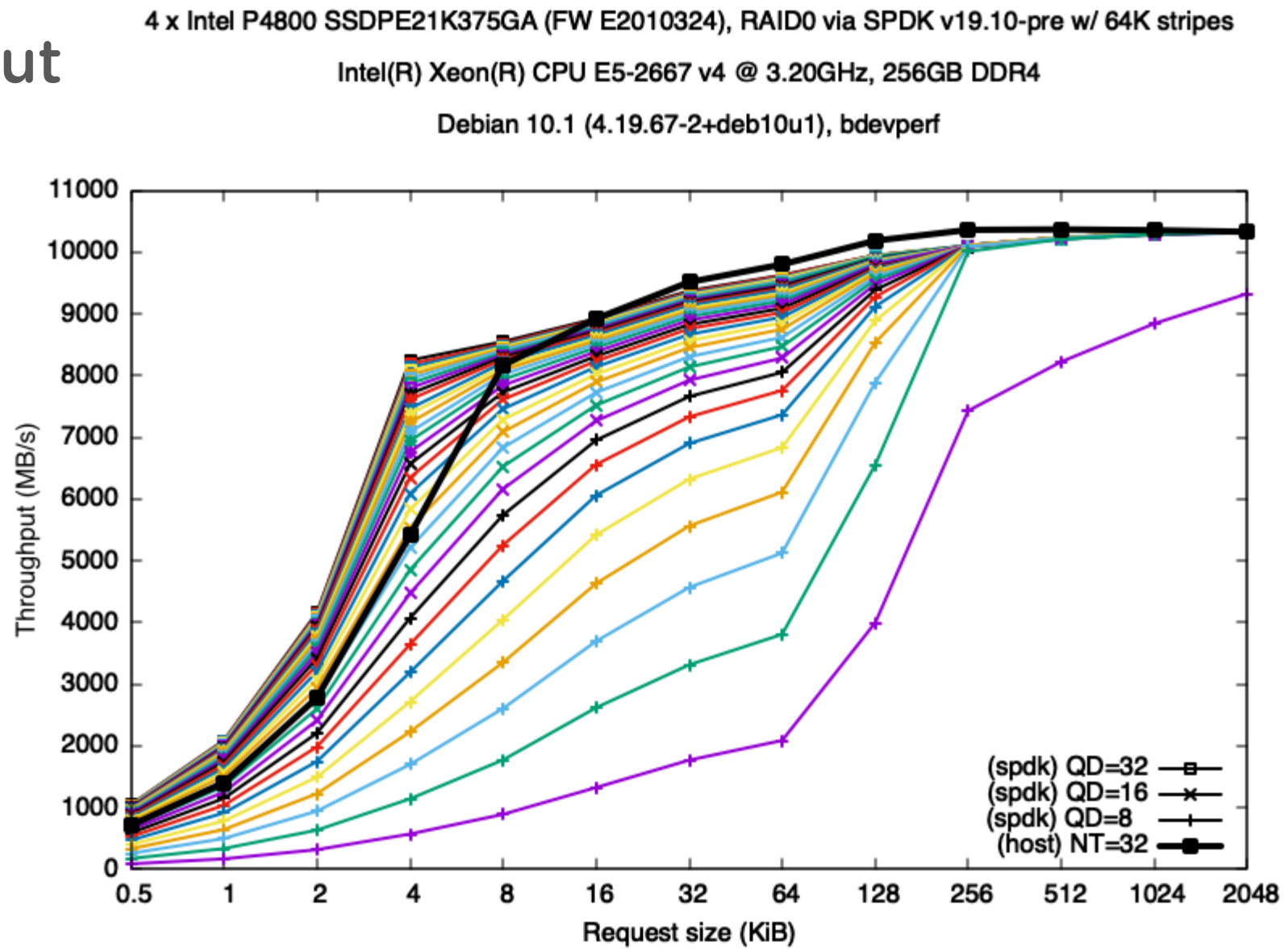
SPDK virtio-blk deployment

- Qemu responsible for configuring virtio-blk device
- Datapath (ie. VQs) offloaded to SDPK via vhost
- One SPDK application per host
  - Many VMs can be driven by one/few thread which:
  - Polls VMs' VQs (no VM EXITs)
  - Directly accesses NVMe devices
  - Polls NVMes CQs (no IRQs)
  - IRQs the guests for completions
    (if guest is polling, no IRQs are required)
- Drawback: NVMes dedicated to this application
  - Totally OK for hypervisors (NVMes are for VM data)

Baseline using SPDK

Measuring Throughput

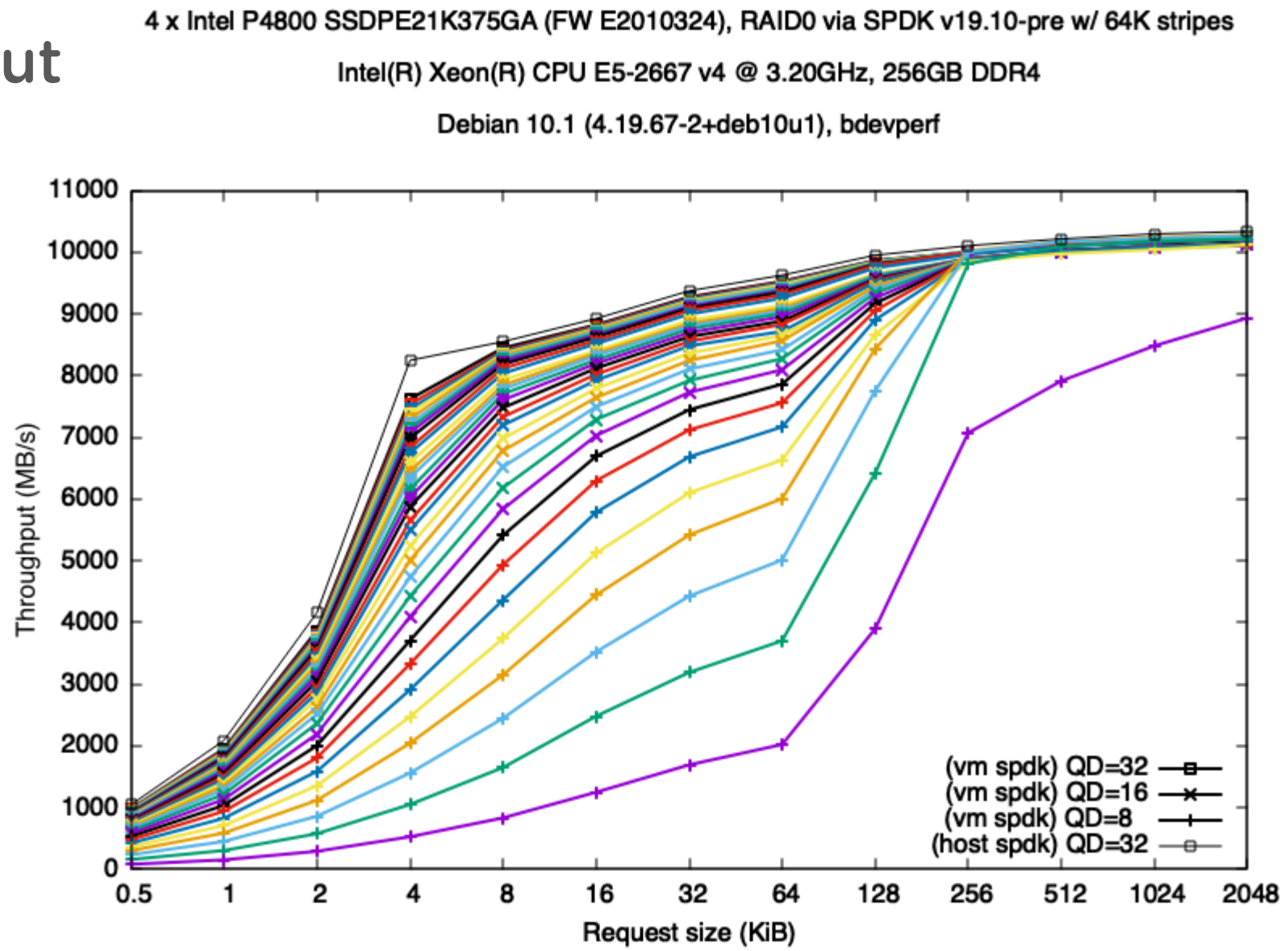
- Direction: Read
- Sequence: Random
- Interface: SPDK
- Req size: 512 B ... 2 MiB
- Num threads: 1
  - QD/thread: 1 ... 32
- Thread pinned to CPUs 8 (in drive's NUMA node)



Hypervisor Analysis: KVM + Qemu + SPDK

Measuring Throughput

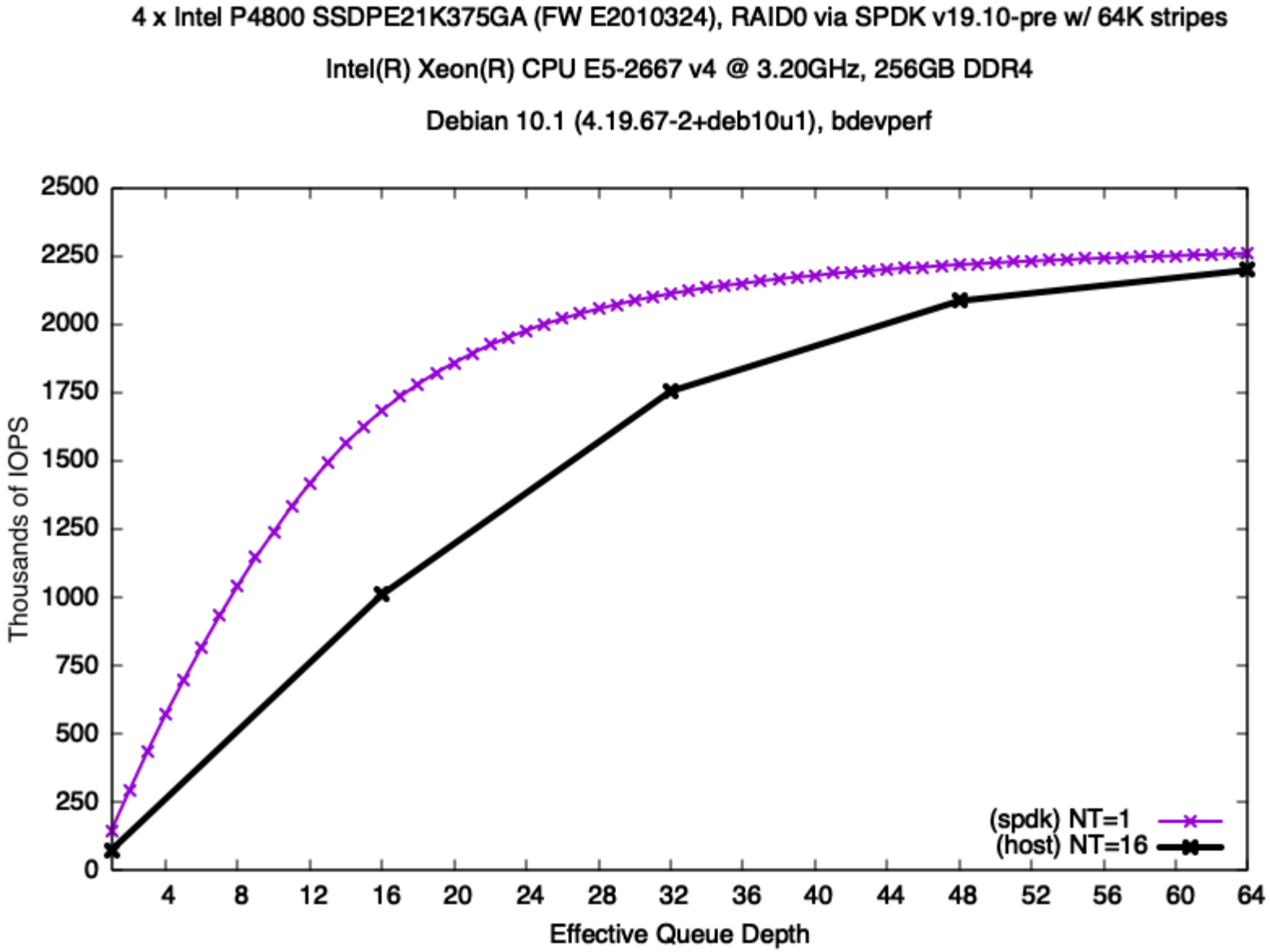
- Direction: Read
- Sequence: Random
- Interface: SPDK
- Req size: 512 B ... 2 MiB
- Num threads: 1
  - QD/thread: 32
- Thread pinned to CPUs 8 (in drive's NUMA node)
- VM (Qemu 3.1.0, SPDK)
  - Num threads: 1
  - QD/thread: 1 ... 32
  - VM pinned to CPUs 8-15



Baseline using SPDK

Measuring IOPS

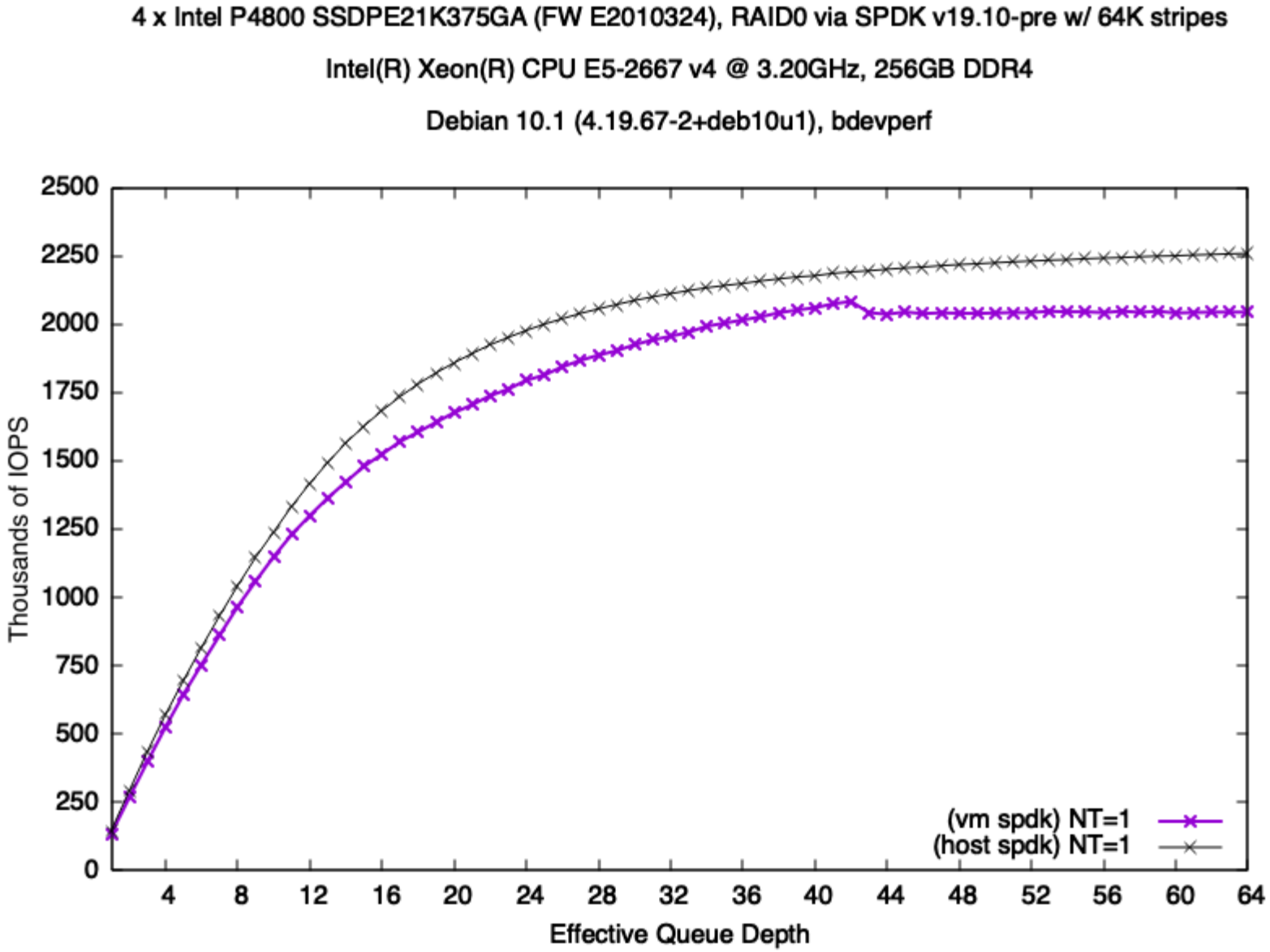
- Direction: Read
- Sequence: Random
- Interface: SPDK
- Req size: 4 KiB
- Num threads: 1
  - QD/thread: 1 ... 64
- Thread pinned to CPU 8 (in drive's NUMA node)



Hypervisor Analysis: KVM + Qemu + SPDK

Measuring IOPS

- Direction: Read
- Sequence: Random
- Interface: SPDK
- Req size: 4 KiB
- Num threads: 1
  - QD/thread: 1 ... 64
- Thread pinned to CPU 8 (in drive's NUMA node)
- VM (Qemu 3.1.0, SPDK)
  - Num threads: 1
  - QD/thread: 1 ... 64
  - VM pinned to CPUs 8-15



Thank you