SDC 20

BY Developers FOR Developers

Storage Developer Conference
September 22-23, 2020

Improving NVMe/TCP Performance by Enhancing Software and Hardware

Sagi Grimberg, Lightbits
Anil Vasudevan, Intel

Notices and Disclaimers

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to www.intel.com/benchmarks.

Performance results are based on testing as of date disclosed in the system configuration and may not reflect all publicly available security updates. See configuration disclosure for details. No product or component can be absolutely secure.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. Check with your system manufacturer or retailer or learn more at www.intel.com.

Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.

Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice Revision #20110804.

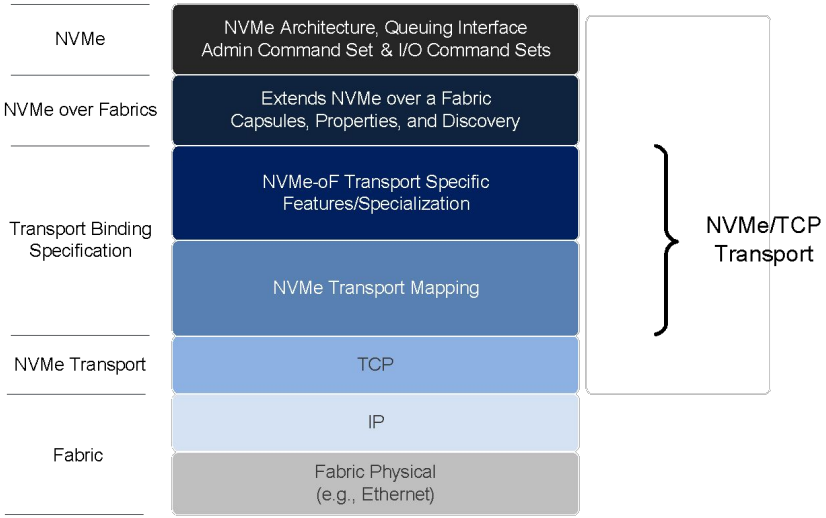
Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

Intel, the Intel logo, Intel Atom®, Xeon and Xeon logos, are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

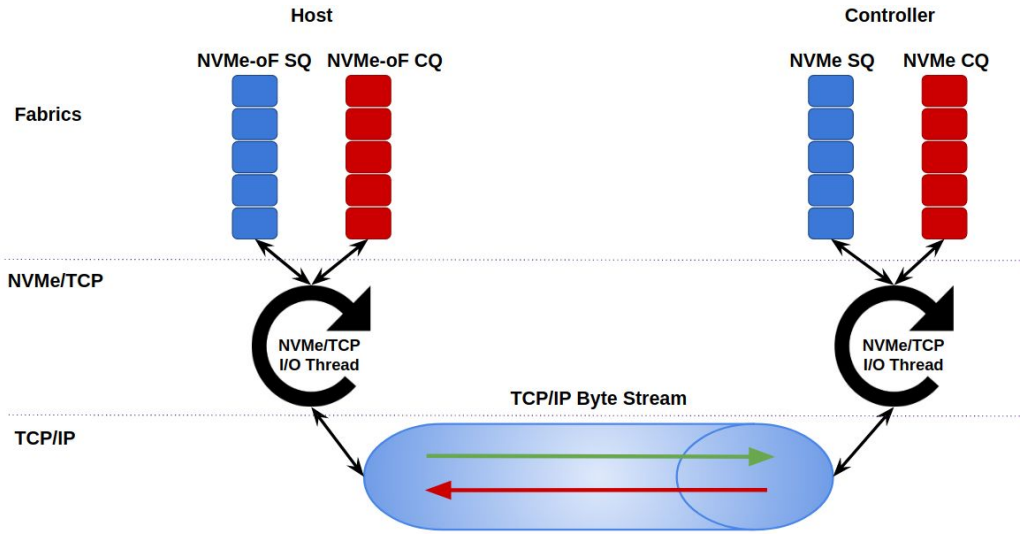
NVMe/TCP (short) Intro

- NVMe/TCP is the standard transport binding to run NVMe on top of standard TCP/IP networks
- Standard NVMe multi-queue interface runs on top of TCP sockets
- Same NVMe command set, encapsulated over NVMe/TCP PDUs



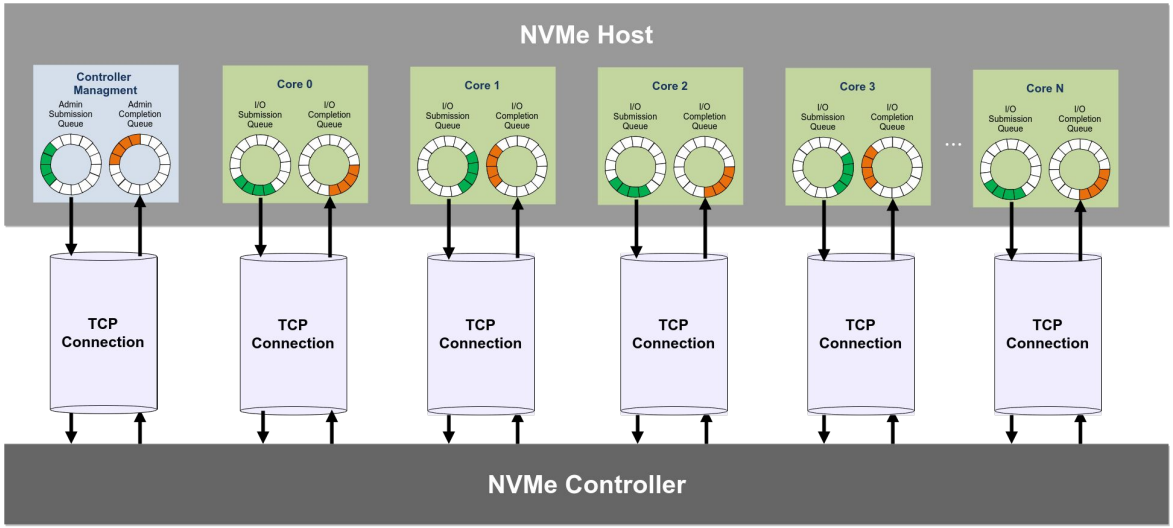
NVMe/TCP (short) Intro

- Each NVMe queue-pair is mapped to a bidirectional TCP connection
- Commands and data-transfer are processed by a dedicated context

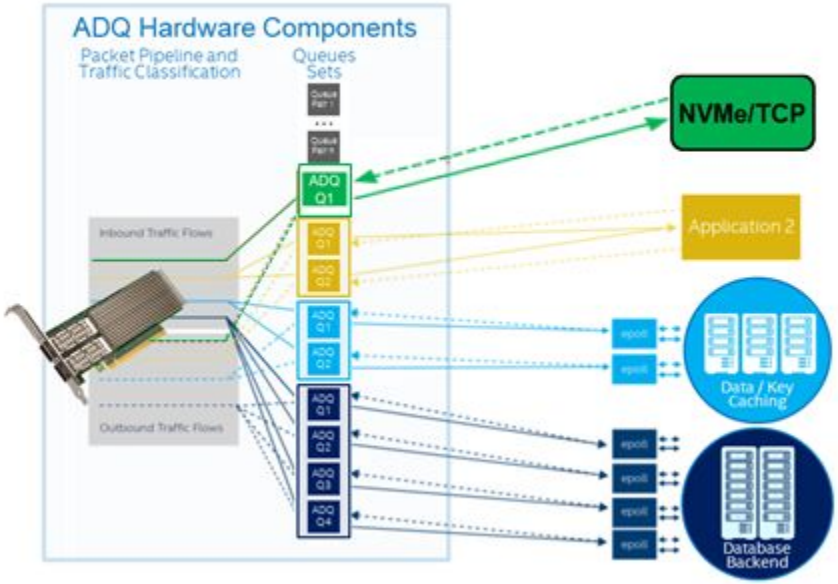


NVMe/TCP Queue Mapping

- Each NVMe queue normally mapped to a dedicated CPU core
  - But not necessarily
- No controller-wide serialization



Application Device Queues (ADQ)



ADQ Basics

- Filters application traffic to a dedicated set of queues
- Application threads of execution are connected to specific queues within the ADQ queue set
- Bandwidth control of egress (Tx) network traffic per application

	Capability
Application	Align Application Threads and ADQ's
Kernel	Busy Polling Device Queues (e.g. epoll(), recv(), poll()) Symmetric Queuing for receive and transmit Queue Identification for Applications HW accelerated Application receive traffic steering configuration HW accelerated Application transmit traffic shaping configuration
Driver	Steering and signaling optimizations
NIC HW	Application specific traffic steering and queuing Application transmit traffic shaping

Latency Contributors

- **Serialization** - Lightweight, only on a per-queue basis (and hctx, sockets etc) - scales pretty well
- **Context Switching** - 2 at a minimum contributed by the driver
- **Memory copy** - Only on RX, not a huge contributor (sometimes is on high load)
- **Interrupts** - Definitely impactful, LRO/GRO/Adaptive-moderation can mitigate a bit, but latency is less consistent
- **socket overhead** - Exists, but not huge, mostly around small size RX/TX
- **Affinitization** - Definitely a contributor if not affinitized correctly
- **Cache pollution** - Has some, not excessive
- **Head-of-Line blocking** - Can be apparent in mixed workloads

Host Direct-IO

- **User issues issues direct file/block I/O** (*ignoring the rest of the stack*)
- **nvme_tcp_queue_rq prepares NVMe/TCP PDU and place it in a queue**
- **nvme_tcp_io_work context picks up I/O and process it**
- **I/O completes, controller sends back data/completion to the host**
- **NIC generates interrupt**
- **NAPI is triggered**
- **nvme_tcp_data_ready is triggered**
- **nvme_tcp_io_work context is triggered, processing and completing the I/O**
- **user context completes I/O**

Host Direct-IO

- **User issues issues direct file/block I/O** (*ignoring the rest of the stack*)
- **nvme_tcp_queue_rq prepares NVMe/TCP PDU and place it in a queue**

Context-Switch

- **nvme_tcp_io_work context picks up I/O and process it**
- **I/O completes, controller sends back data/completion to the host**
- **NIC generates interrupt**

Soft-IRQ

- **NAPI is triggered**
- **nvme_tcp_data_ready is triggered**

Context-Switch

- **nvme_tcp_io_work context is triggered, processing and completing the I/O**
- **user context completes (via io_getevents)**

Mixed Workload Optimization

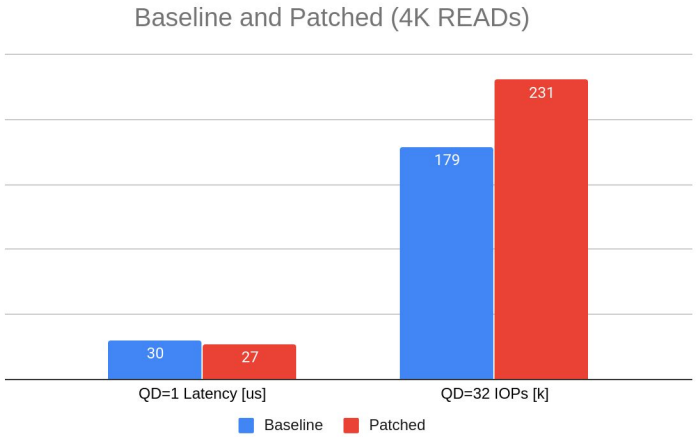
- **Linux block layer allows for multiple queue maps**
  - Default: normal set of HW queues
  - Read: Dedicated queues for Reads
  - Poll: Dedicated queues for polling application and RWF_HIPRI I/O
- **Eliminate Head-of-Line blocking of small reads vs. large writes**
  - Send Reads on dedicated read queues, and writes on default queues
- **Added support for multiple queue maps and plugging into the block layer**

Test: 16 readers issuing synchronous 4K reads, 1 unbound writer issuing 1M writes @QD=32

	READ IOPs [k]	READ Ave Latency [us]	READ 99.99% latency [us]
Baseline	80.4	396	14222
Patched	171	181.5	1811

Affinity Optimizations

- **Linux grew capability to split different I/O types to different queue maps**
- **optimize queue io_cpu assignment for multiple queue maps**
  - use separated alignment for different queue maps (read/default/polling)
  - calculate each queue map alignment individually
  - Especially important for Read and Poll queue maps



Low QD latency optimizations - TX path

- **Eliminate NVMe/TCP context switch when queuing a request**
  - **Prepare NVMe/TCP and process directly from nvme_tcp_queue_rq**
  - **Network send might_sleep, so need to convert hctx locking to srcu**
  - **Serialize of two contexts of the same queue is required**
    - Introduce a mutex
    - Only if the queue is empty
    - Only if the queue mapped CPU matches the running cpu
- **Socket priority**
  - Steers egress traffic to the preferred NIC queue set

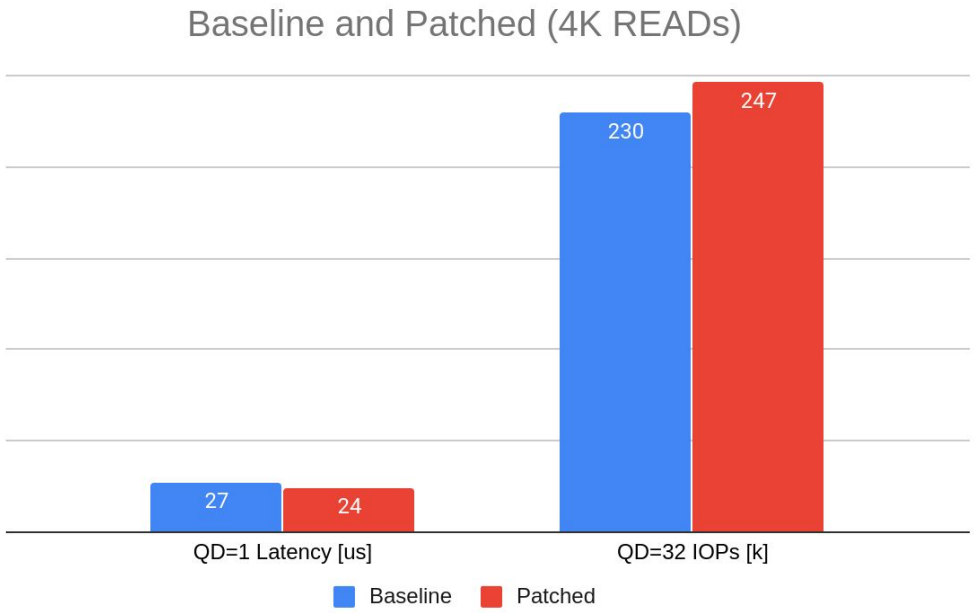
Low QD latency optimizations - RX path

- **Linux grew a polling interface for latency sensitive I/O**
  - Submit with RWF_HIPRI
  - Poll for completion (also via io_uring IORING_SETUP_IOPOLL)

- **We add nvme_tcp_poll and plug it into blk_poll interface**
  - Add dedicated queues for polling (connect options)
  - nvme_tcp_poll calls sk_busy_loop

- **Skip RX data_ready context switch if application is polling at the same time**
  - Mostly true if NIC moderation is working well
  - If device can hold off interrupts more aggressively it works very well

Low QD latency optimizations - Results



Target Optimizations

- **Assign I/O threads based on *so_incoming_cpu***
  - Avoid unnecessary context switches
  - But may end up with uneven balancing in the system
- **Group multiple connections to a single I/O context**
  - Each context is processing a NIC HW queue
- **Allow for more polling friendly heuristics**
  - Polling budget based on poll groups
  - Less context switching and fewer interrupts
  - Still maintain Fairness per connection
- **Socket priority**
  - Steers egress traffic to the preferred NIC queue set

ADQ improvements

Traffic Isolation - Direct NVMe traffic to its dedicated queue set

- Inbound:
  - Dedicated queue-set configuration (tc-mqprio)
  - Traffic Filtering (tc-flower)
  - Queue selection (RSS/Flow Director)
- Outbound:
  - Set Socket priority
  - Extensions to Transmit Packet Steering (XPS)

Value

- No noisy traffic from neighbor workloads
- Opportunity to customize network parameters for a specific workload

ADQ improvements

Minimizing Context switching and Interrupts overhead

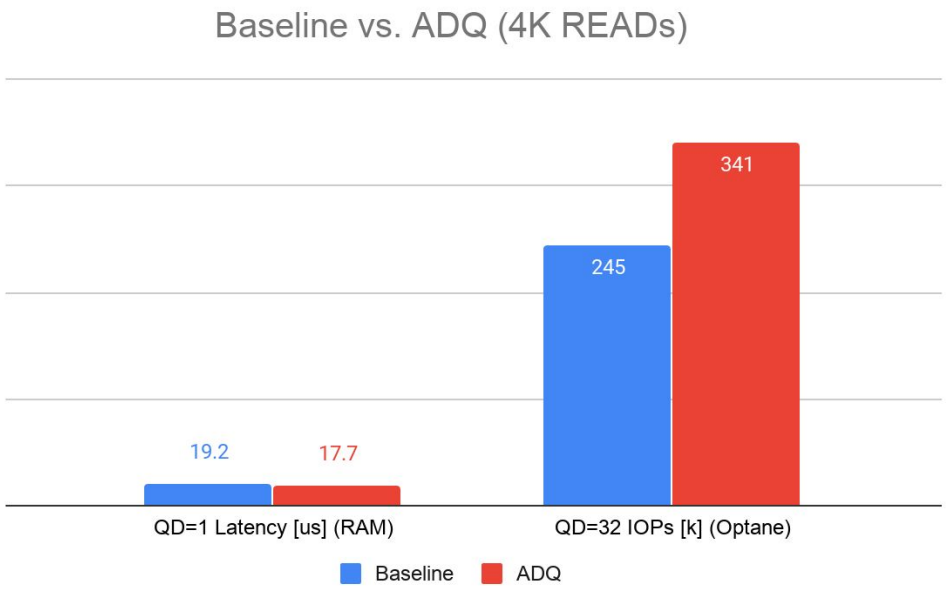
- Busy polling on dedicated queue set
  - Drain network completions in application context
  - Process NVMe completions directly in application context
- Handle Request/Response in application context
  - keeps the application thread active - no redundant context switch
- Grouping multiple NVMe/TCP queues to a single NIC HW queue
  - Streamlines sharing of a NIC HW queue - no redundant context switch

Value

- Reducing CPU utilization
- Lowering Latency

ADQ Measurements

- Comparing NVMe/TCP with ADQ enabled vs. ADQ Disabled
- Platform is Cascade-Lake

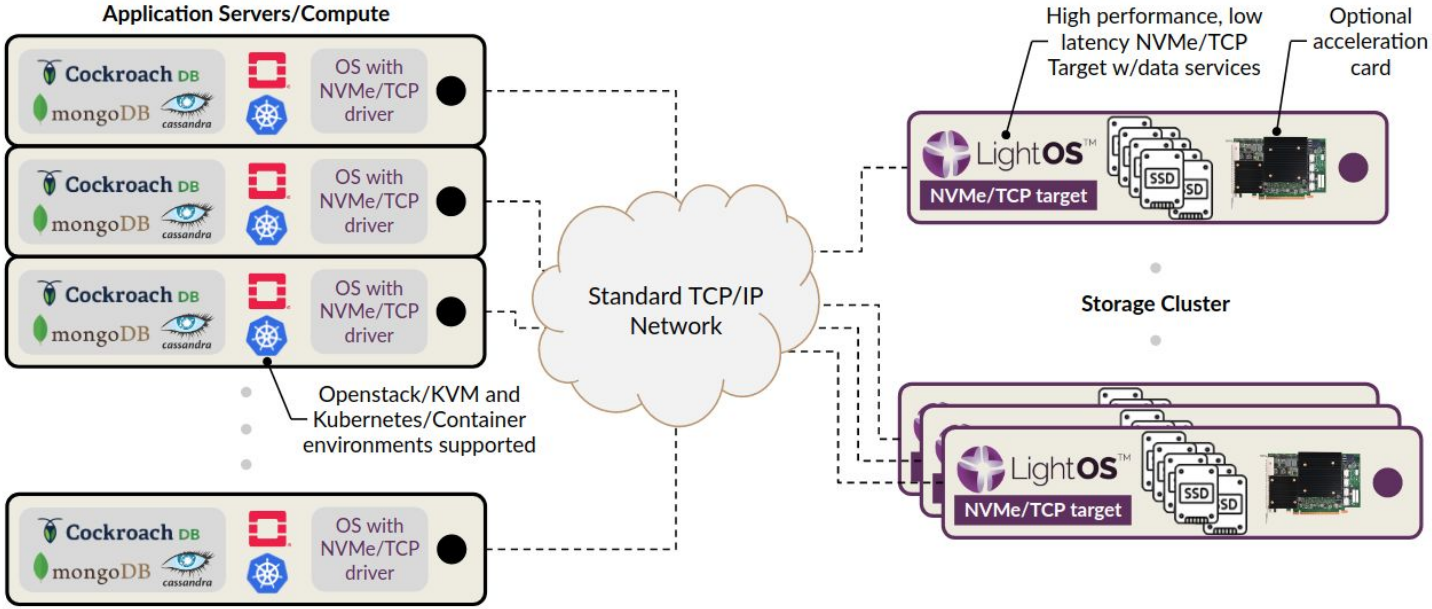


LightOS

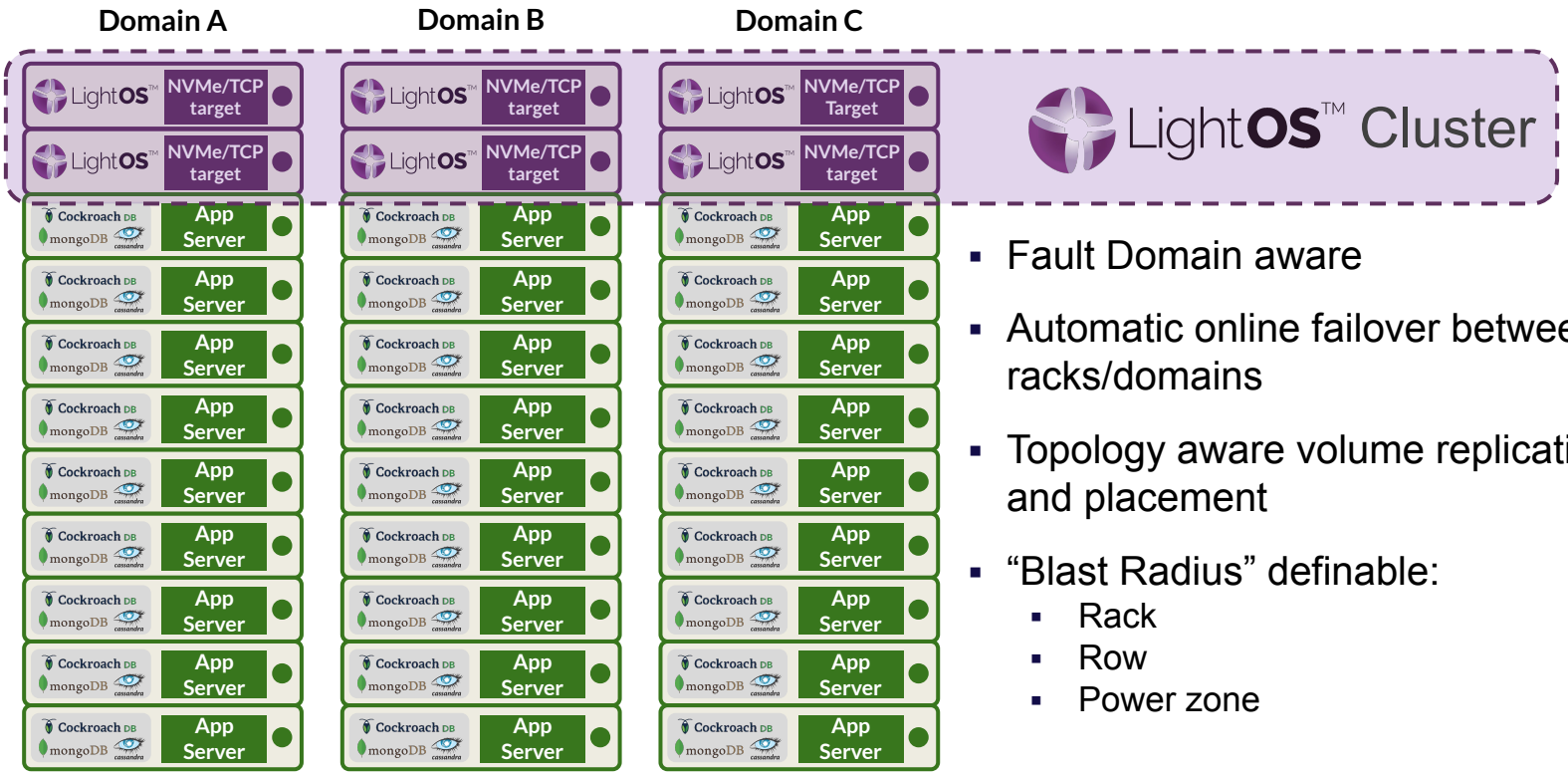
Simple, Available, Reliable, Scalable and Low latency

High Performance Distributed SDS

Standard servers, NICs and SSDs, optional hardware accelerator



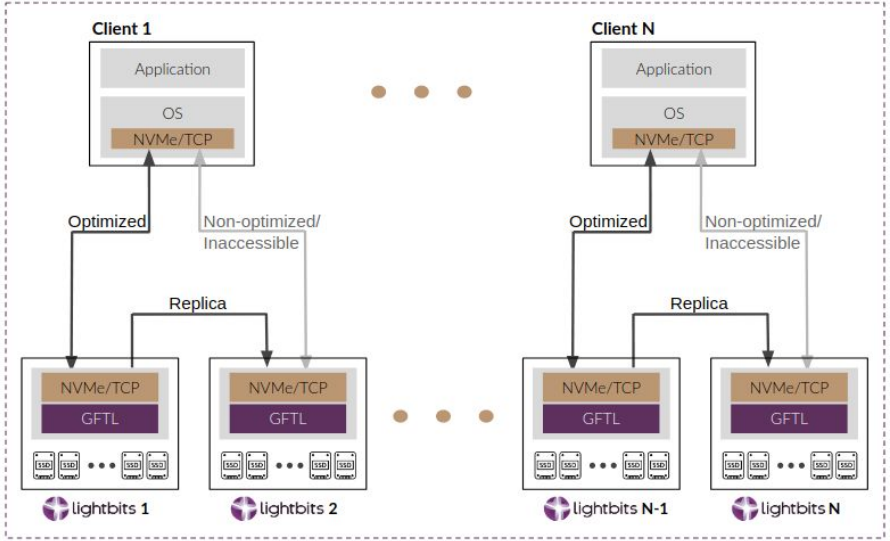
Multi-Rack, Scale-Out



- Fault Domain aware
- Automatic online failover between racks/domains
- Topology aware volume replication and placement
- “Blast Radius” definable:
  - Rack
  - Row
  - Power zone

High Availability

- **SSD Failure Protection with RAID**
- **Volume Protection Policies**
  - 1/2/3 way sync replication
- **NVMe Multipathing**
  - Asymmetric Namespace Access
- **NIC Bonding/Teaming Support**
- **All services highly available**
  - API Service
  - NVMe Discovery Service



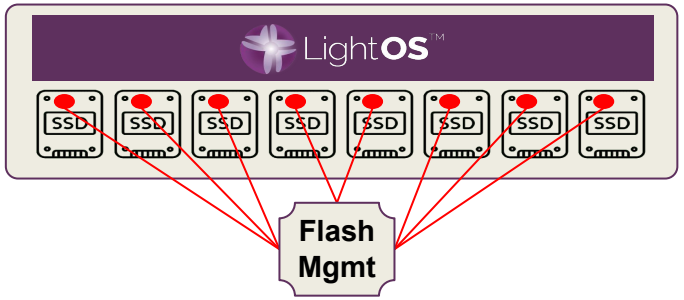
Rich Data Services

Data Services

- Logical Volumes
- Thin-Provisioning
- Inline Compression
- Online Capacity Expansion
- SSD hotplug support
- Online Volume Resize
- Erasure Coding

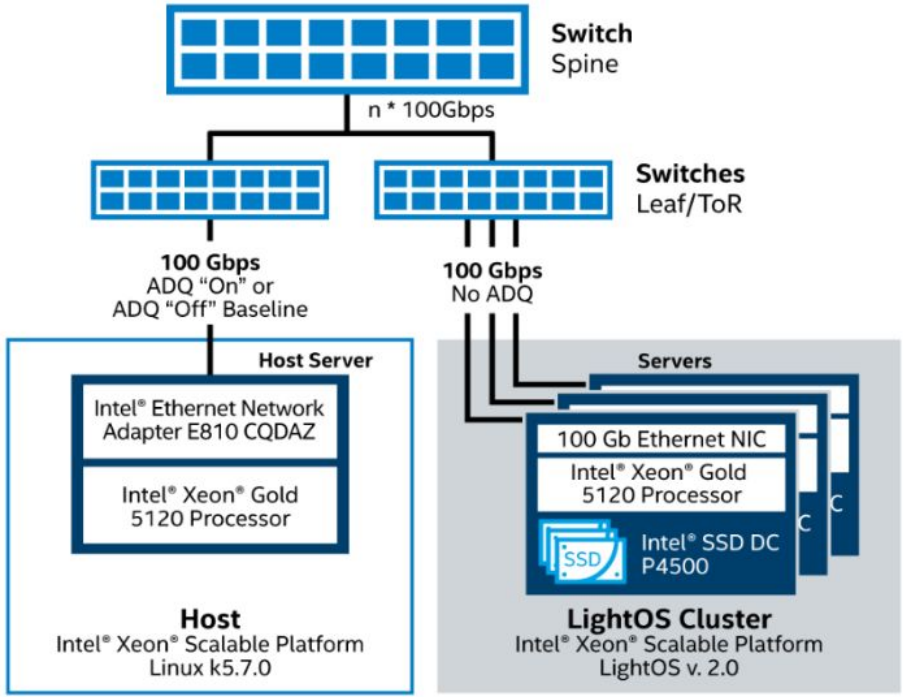
Flash Management

- Endurance improvements (QLC support)
- Consistent Low Latency



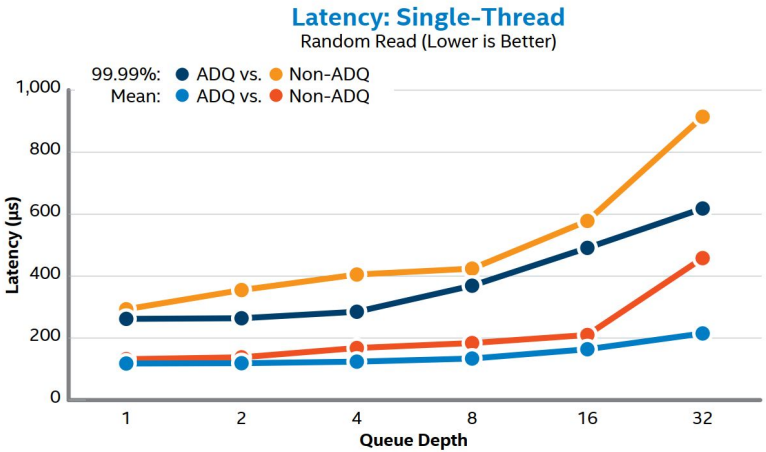
Performance with ADQ Host

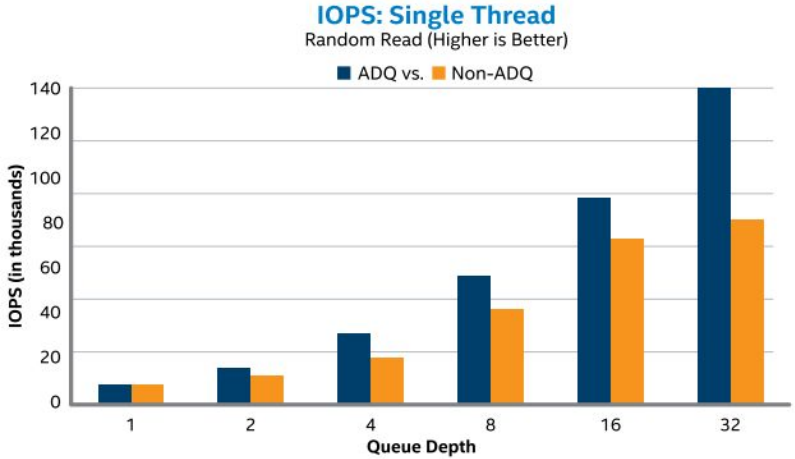
Test Setup



Performance with ADQ Host

Single Thread

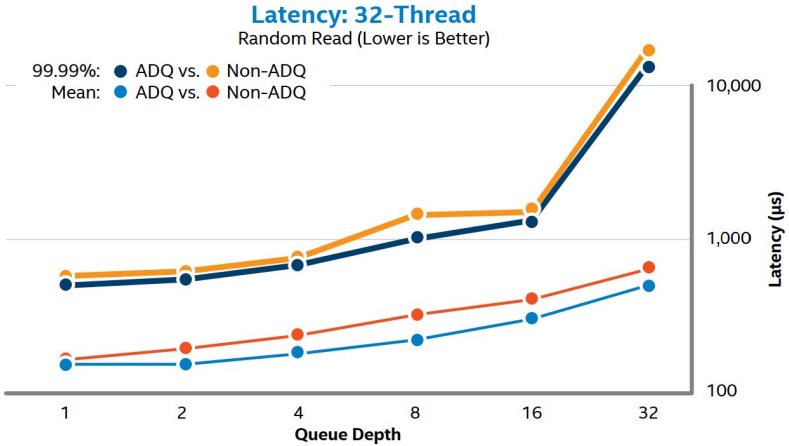


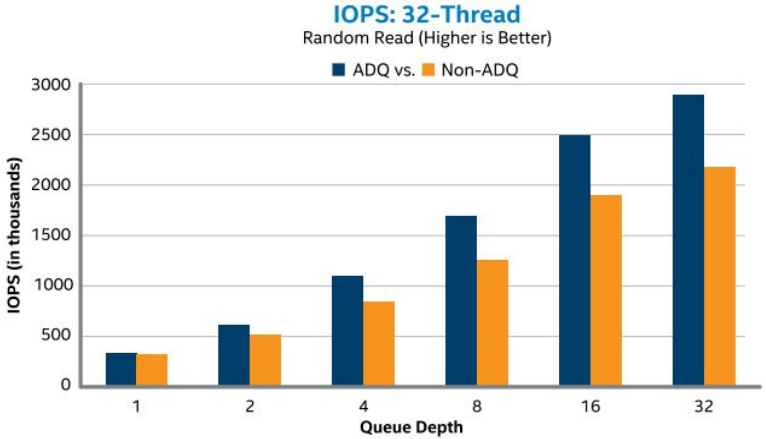


- ***Up to 30% improvement in 99.99% tail latency***
- ***Up to 50% improvement in average latency***
- ***Up to 70% improvement in IOPs***

Performance with ADQ Host

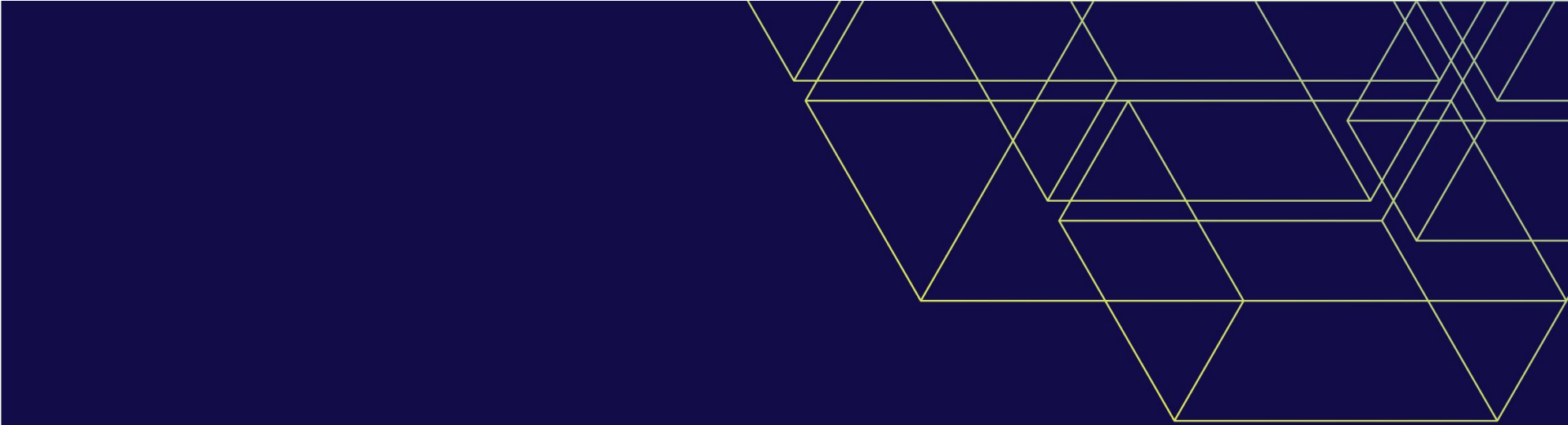
Multi Thread





- ***Up to 30% improvement in 99.99% tail latency***
- ***Up to 30% improvement in average latency***

- ***Up to 30% improvement in IOPs***



Please take a moment to rate this session.

Your feedback matters to us.

Questions?

Test Configuration (Slide 24)

	System Under Test	LightOS Cluster
Test By	Lightbits Labs	Lightbits Labs
Test Date	July 15, 2020	July 15, 2020
Platform	Supermicro SYS-2029U-TN24R4T	Supermicro SYS-2028U-TN24R4T+
# Nodes	1	1 to 3
# Sockets	2	1
CPU	Intel® Xeon® Gold 5120 Processor @ 2.2 GHz	Intel® Xeon® Processor E5-2648L v4 @ 1.8 GHz
Cores/Socket, Threads/Socket	14 cores/socket, 28 threads/socket	14 cores/socket, 28 threads/socket
Microcode	0x2000065	0xb000036
Hyper-Threading	On	On
Turbo	On	On
BIOS Version	3.2	American Megatrends Inc. (3.1c)
System DDR Mem Config: slots/cap/run-speed	16 slots/16 GB/2133 MT/s DDR4	16 slots/16 GB/2133 MT/s DDR4
Total Memory/Node (DDR+DCPMM)	256 GB	256 GB
Storage - Boot	128 GB SATADOM-SL 3ME3	128 GB SATADOM-SL 3ME3
Storage - Application Drives	N/A	8x Intel® SSD DC P4500
Network Adapter	1x Intel® Ethernet Network Adapter E810-CQDA2 @ 100Gbps	**Single-Node:** 1x Intel® Ethernet Network Adapter E810-CQDA2 @ 100Gbps **Multi-Node:** Add 2 x Mellanox ConnectX-4 EN Ethernet Adapter @ 100Gbps
PCH	N/A	N/A
Other Hardware (Accelerator)	N/A	N/A
OS	CentOS 7.7	LightOS version 2.0 (CentOS 7.7)
Kernel	5.7.0+.x86_64	4.14.189_00172587149ee079f0f16_rel_lb-7.x86_64
Workload and version	FIO 3.20	N/A
NVME/TCP with ADQ Patch	Pull request until put into the main branch: All upstream	N/A
Compiler	N/A	N/A
Driver	1.0.4-1.x86_64, firmware version: 1.40 0x80003ab8 1.2735.0, iproute-4.11.0-25.el7_7.2.x86_64	1.0.4-1.x86_64, firmware version: 1.40 0x80003ab8 1.2735.0, iproute-4.11.0-25.el7_7.2.x86_64
NVMe/TCP Settings	MTU set to 1500 Connected to targets with 32 Polling queues	MTU set to 1500
LightOS Settings	N/A	Default
Network Switches	**Host Leaf:** Accton 7712-32X/AOS **Spine:** Mellanox MSN2700-CS2F	**Cluster Leaf:** Accton 7712-32X/AOS **Spine:** Mellanox MSN2700-CS2F
SSD Pool	N/A	8x Intel® SSD DC P4500 1 TB (2.5" U.2)

Test Configuration (Slide 24)

	ADQ "Off" Baseline	ADQ "On"
System Settings		
Interrupt Moderation	adaptive-rx	rx_usecs=0 tx_usecs=50
IRA Balance	Off	Off
Interrupt Affinitization	Linear	Linear
ADQ Settings		
Epoll Busy Poll	N/A	N/A
Socket Option for NAPI ID	N/A	N/A
TC-Mqprio Hardware Offload and Shaper	None	On
TC- Cloud Filter Enabling with TC-flower	None	On
Symmetric Queueing	Off	On