BROADCOM®

White Paper

NVMe over Fabrics Performance

Stingray™-Based Storage Appliance

NVMe-oF Overview

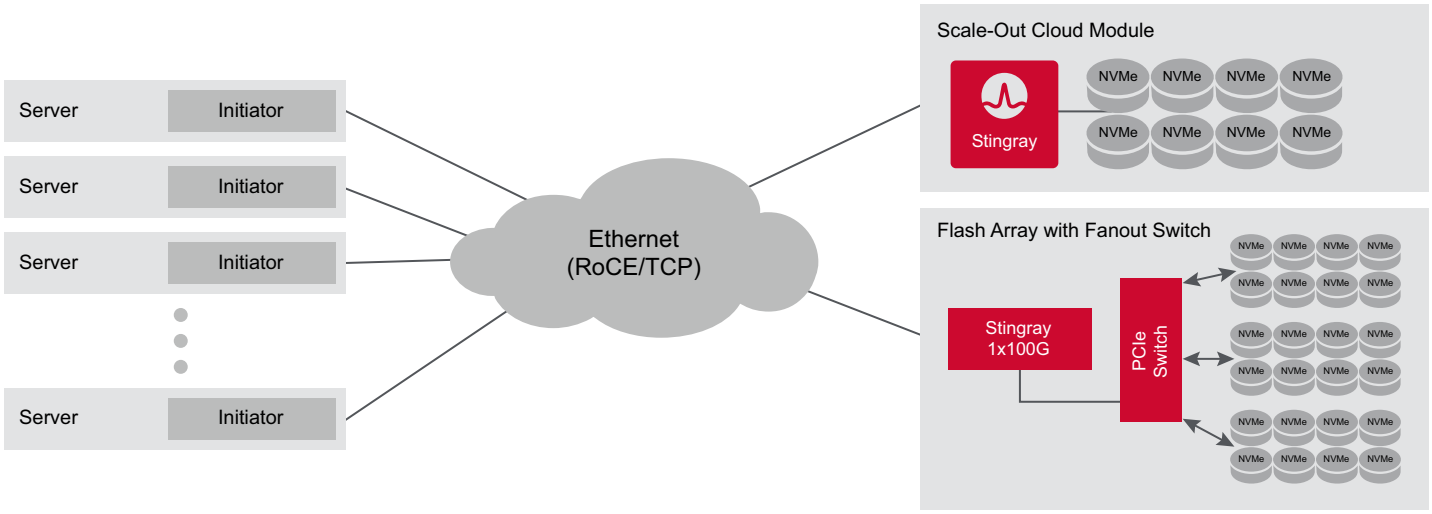
The introduction of solid state drives (SSD) in 1999 engendered a new performance class of data storage not previously available. This provided a major step forward in the theoretical performance of storage that had been dominated for a generation by hard disk drive (HDD) spinning media. However, while those drives showed great promise of what the future of flash offered, overall system performance was still hampered by flash device technology, interconnect, and existing storage protocols. In 2007, FusionIO and similar companies introduced the first truly high-performance flash drives based on PCI Express (PCIe). This generation of flash overcame some of the limitations of early solid state drives, such as lower reliability and wear levels. Combined with PCIe's higher bandwidth and lower latency, this enabled very high random read 4 KB I/O per second (IOPS) performance up to 10 times faster than the existing standard serial attached SCSI (SAS)-based SSDs at the time. These technology advancements provided the first truly high-performance, locally connected flash storage in a server and proved to be very successful.

Over the past 10 years, the performance and reliability of flash have increased significantly to the point that newer PCIe Gen 3-based drives can achieve 500K+ IOPS performance per drive. However, one major limitation of this technology was scale. As scale-out applications and virtualization advanced over the past 10 years, directly connected server flash increasingly became a performance bottleneck. As applications or virtual machines (VMs) started running across physical servers, performance was limited to the drives still housed in a server. With locally attached, non-connected storage, it was impossible to share the data stored on those drives between physical servers. This created a scenario where there was either too much flash storage in a server or not enough. Neither situation was optimal and caused a lot of inefficiency for storage usage. Companies were forced to provision more storage than typically required, increasing capex. This of course still left the problem of scale, so the issue of sharing storage remained unsolved.

To address this problem, in 2016 the NVMe Express consortium approved the first revolutionary standard in a decade: NVMe over Fabrics (NVMe-oF). NVMe-oF is a protocol that allows access to remote NVME drives with performance and latency comparable to locally attached PCIe-based NVMe drives. Fabrics is supported by a number of transport protocols including Ethernet, Infiniband, and Fibre Channel. Fabrics enables storage to be disaggregated from compute nodes, allowing maximum flexibility for next generation servers and applications. New architectures based around composable infrastructure are being developed and deployed to leverage this new ability to connect compute and storage in a flexible and cost-efficient manner.

This paper demonstrates Broadcom's NVMe-oF-based Stingray™ data center system-on-a-chip (SoC) designed to support native NVMe-oF using both RoCE (RDMA over Converged Ethernet) or TCP/IP.

Figure 1: Stingray Data Center SoC



The Broadcom Stingray SoC integrates a full-featured 100 Gb/s RDMA Network Interface (RNIC) based on the same technology as the Whitney+ and Thor RNICs, a high-performance octal-core A72 Arm-based CPU subsystem running at an industry-leading 3 GHz, and a high-bandwidth hardware accelerator capable of up to 100 Gb/s for functions like RAID 5/6, Crypto, and De-dupe.

NVMe over Fabrics

NVMe over Fabrics enables remote NVMe storage to act and perform like locally attached drives. This provides economies of scale for disaggregated storage and compute nodes.

Performance Setup and Objective

This paper focuses on NVMe-oF performance using the Stingray PS1100R 100G storage controller in comparison with local directly-attached storage (DAS). RoCE is used as NVMe transport in this version of the report and future versions will include results for TCP. The goal of this paper is to emphasize the maturity level of NVMe-oF as well as the stability and performance of the Stingray NVMe-oF implementation (in both IOPS and bandwidth). Additionally this paper compares the Stingray solution against an x86-based storage target running Fabrics.

Tools

The document presents performance data gathered using single or multiple initiators. The target side always uses SPDK and exports the drives that are present in the storage target chassis. On the initiator side, several approaches are used for measuring performance:

- fio using libaio engine (going through the initiator kernel stack)
- fio using SPDK engine (using the SPDK fio plugin, thus mostly sidestepping the kernel except for basic fio operation)
- SPDK perf utility

Using the libaio engine and going through the initiator kernel stack can introduce variability in the results depending on the kernel being used, along with more dependency on the CPU performance of the initiator. On the other hand, the SPDK results (either by using the SPDK fio plugin or the SPDK perf utility) demonstrated more consistent results. The results presented in this paper are based on the SPDK initiator.

Setup

Three network configurations were tested. The first two use a Stingray PS1100R storage adapter and the third option uses an x86-based storage target. The three setups are described as follows:

1. A setup with no congestion, where Stingray is directly connected to a single initiator.

   This simple setup is used as a baseline for the performance data since it avoids latencies or congestion on the ports when going through a 100G Ethernet switch.

2. A setup where Stingray (in a JBOF with 8 NVMe drives) is connected to one or more SPDK initiators through 100G Ethernet switch.

3. A setup where an x86-based storage target (running SPDK) is connected to one or more SPDK initiators through 100G Ethernet switch.

Setups 2 and 3 are basically the same with the exception of the target box: the second setup uses a Stingray-based JBOF and the third setup uses an x86-based server. Other common aspects of the latter two setups are as follows:

- A 100G Ethernet switch connecting the target to the initiators. PFC is enabled on the switch.
- Eight x86-based initiators. PFC is enabled on the initiators.
- The targets use identical SSD drives.

NOTE: For additional details on the hardware and software used during testing, see Setup Details.

In the case of multiple initiators, the test run is simultaneous and the throughput or IOPS results are the sum of the individual initiator IOPS or throughput numbers. The individual initiator results can vary from run to run since the performance distribution is dependent on the overall system (including the switch and its congestion control mechanisms). However, the total throughput is fairly consistent.

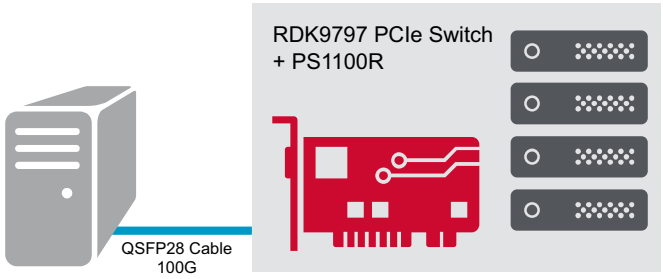
The latency is reported only for the full setup and includes the Ethernet switch. Latency was measured from the average values obtained from one initiator for queue depths 1, 8, and 32 using the SPDK perf utility.

Single-Initiator Performance (RoCE)

Topology

For the single-initiator performance, a single x86 server with a Broadcom P1100p PCIe NIC card (with RoCE support) is connected to a Stingray PS1100R storage adapter connected to a Broadcom RDK9797 PCIe switch and four HGST SN200 or four Intel 900P NVMe drives.

Figure 2: Single-Initiator Performance Topology



SPDK Initiator (perf)

Testing was performed using SPDK on both the target and the initiator. On the initiator side, the SPDK perf application was used for the performance measurements. See Example Test Commands for the perf commands used. Generally this configuration results in optimal performance.

The results were measured using Broadcom's PS1100R 1.2.7.0 release. Unless otherwise noted, the following tests were run on the PS1100R system. See Setup Details for additional details.

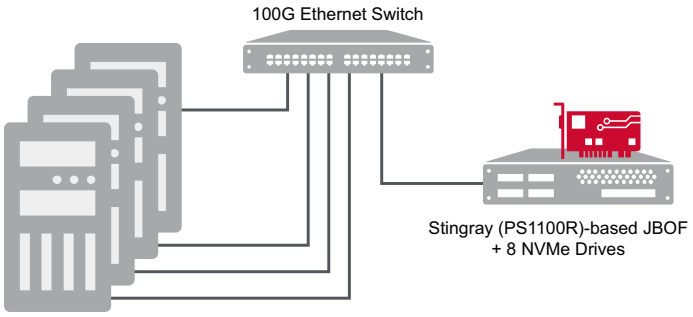
Test Type	Measurement	Drives Used	Result
4K Random Read	Bandwidth (IOPS)	4x HGST	2811 KIOPS
4K Random Write	Bandwidth (IOPS)	4x 900P	2074 KIOPS
4K Random Read	Latency (µs)	1x 900P	15.07 µs
4K Random Write	Latency (µs)	1x 900P	21.50 µs
128K Random Read	Bandwidth (Gb/s)	4x HGST	91.16 Gb/s
128K Random Write	Bandwidth (Gb/s)	4x 900P	73.52 Gb/s
128K Random Write	Bandwidth (Gb/s)	4x HGST + 4x 900P (SST)	81.33 Gb/s

Multiple-Initiator Performance (RoCE)

Topology

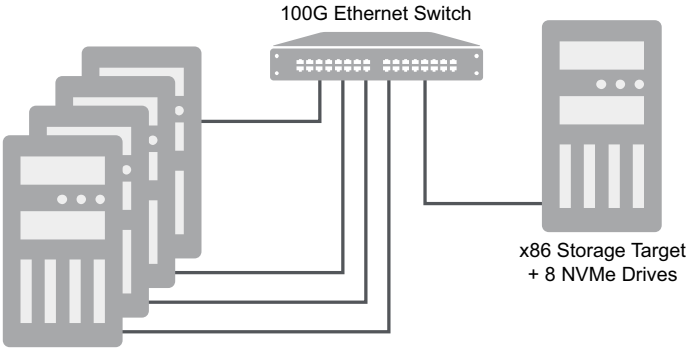
For the multiple-initiator performance, eight x86 initiators using Broadcom P225p cards are connected to a Stingray-based JBOF with PS1100R and eight Dell MZWLL800HEHP NVMe SSD drives. PFC/CC is enabled on the switch and the endpoints.

Figure 3: Multiple-Initiator Performance Stingray Topology



For comparison with Stingray, an x86-based storage target is used while the rest of the setup is the same as in the previous Stingray case.

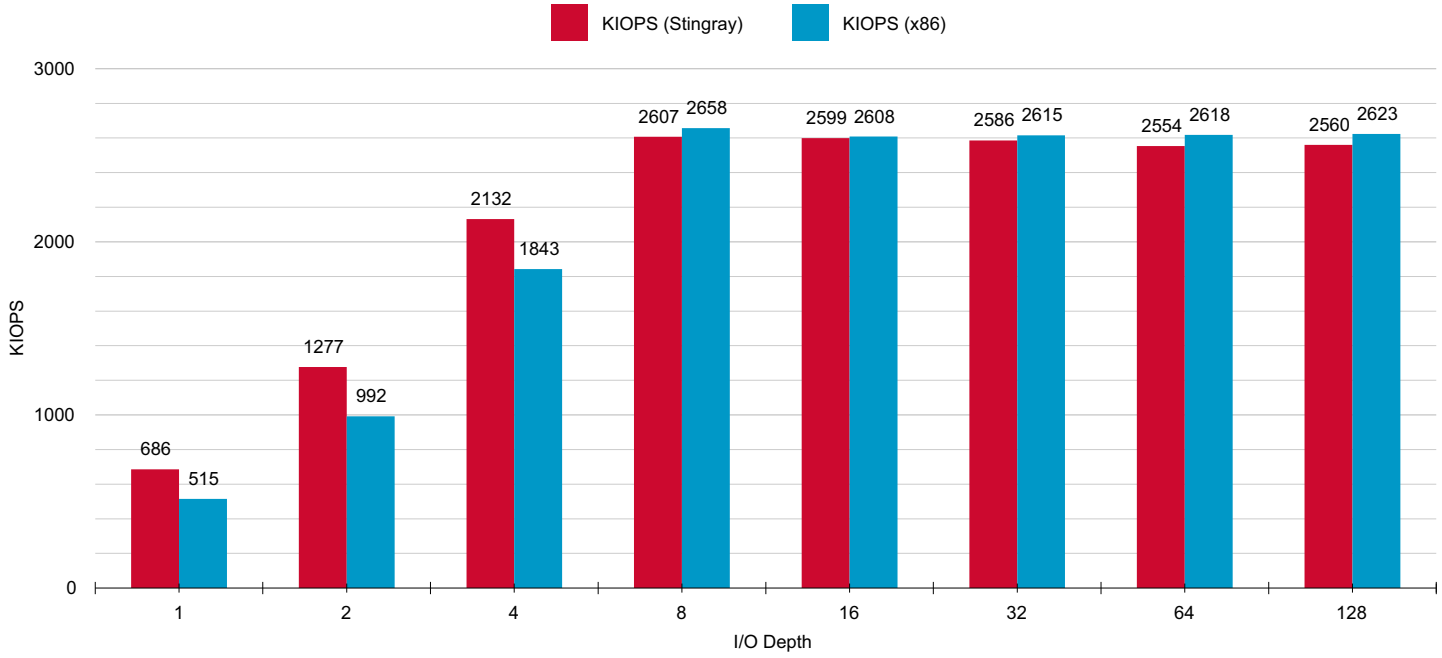
Figure 4: Multiple-Initiator Performance x86 Topology



IOPS Test Results

As seen in Figure 5, once there is sufficient I/O depth (8 in this case) a performance rate of ~2.6 MIOPS is achieved. The results are comparable (within a few percentage points) to the x86-based SPDK target using the same drives. For I/O depths less than 8, Stingray typically performs better than x86.

Figure 5: Random Read (4K): KIOPS vs. I/O Depth



Bandwidth Test Results

Sequential reads almost reach the maximum theoretical bandwidth for both Stingray and x86. The sequential writes show bandwidth slightly above 8.5 GB/s (~70 Gb/s). In this particular setup, the drives are rated at 1000 MB/s for sequential write so the total number for sequential writes appears to be bound by the drive speed.

Figure 6: Sequential Read: Bandwidth vs. Block Size

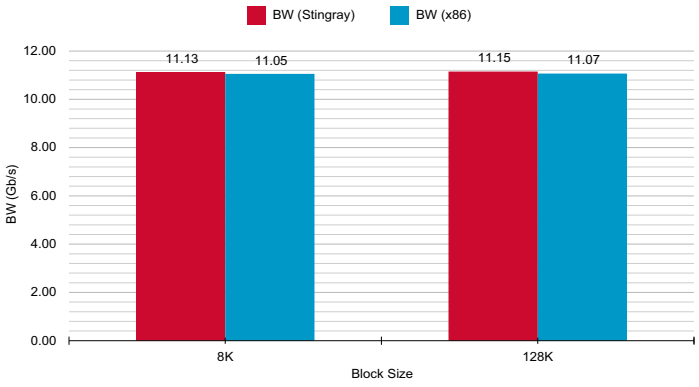
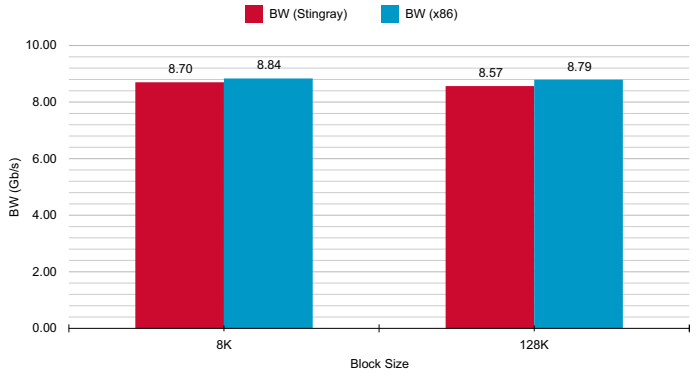
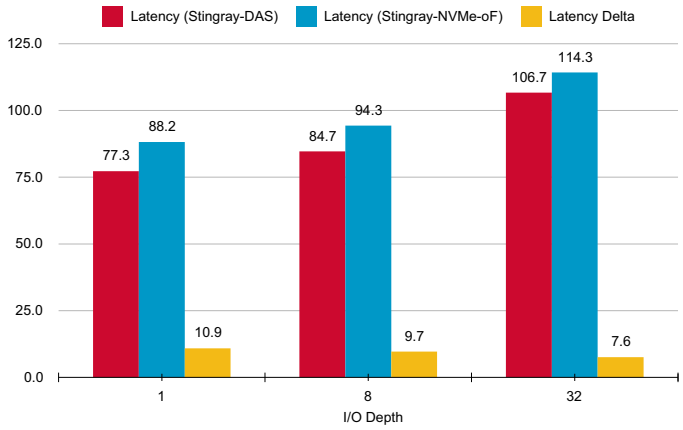


Figure 7: Sequential Write: Bandwidth vs. Block Size



Latency Results

Figure 8: Random Read Latency: Local (direct-attach) vs. NVMEoF



Based on Figure 8, the additional average latency added by NVMe-oF is within 10 µs. In real applications, such as high performance in-memory databases, typical latencies run in the millisecond range[1]. The latencies shown above are representative of multi-initiator systems with data center network congestion. With typical application performance measured in milliseconds, the small amount of incremental NVMe-oF latency demonstrated above should have a negligible impact on application performance.

In addition to the scenario with no traffic (shown in Figure 8), latency in the presence of background traffic coming from a single 25G initiator was measured. The main reason to measure latency with background traffic is to replicate a more realistic scenario within a data center. A disaggregated based rack architecture tries to maximize the efficiency of a storage target by accessing all drives in the target. This creates enough I/Os to sufficiently stress each drive. The results for this scenario are shown in Figure 9 through Figure 11.

1. https://scalegrid.io/blog/comparing-in-memory-databases-redis-vs-mongodb-percona-memory-engine/

Figure 9: Average Random Read Latency (iodepth=1)

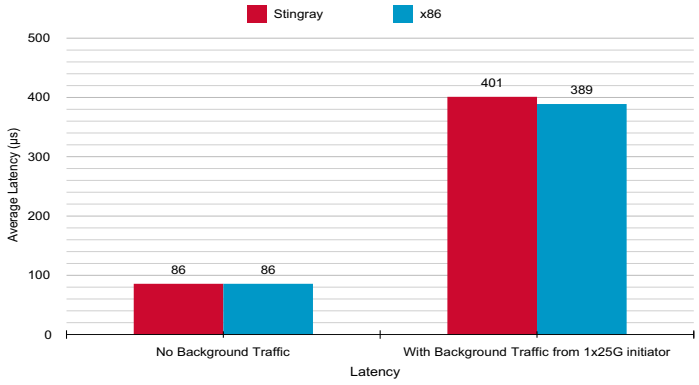


Figure 10: Average Random Read Latency (iodepth=8)

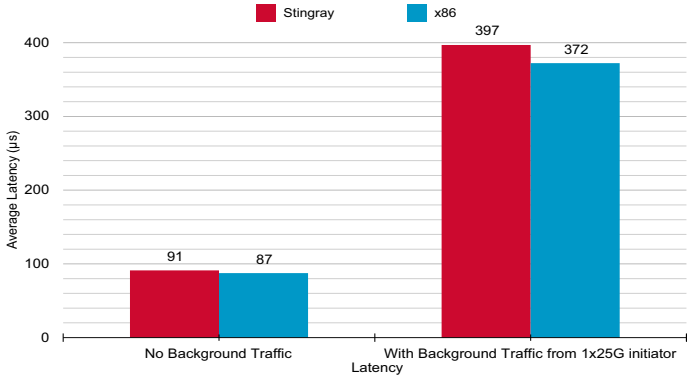
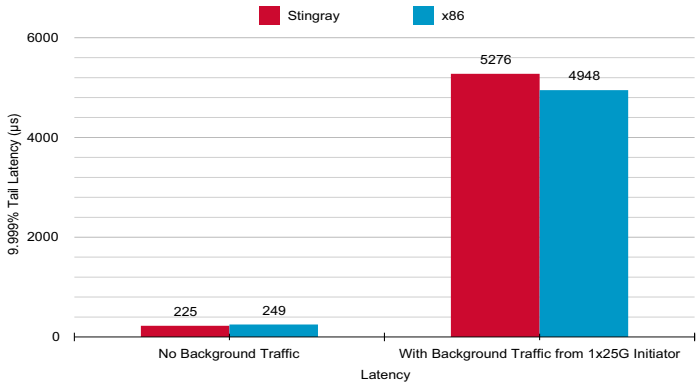


Figure 11: 99.999% Tail Latency (iodepth=8)



Conclusion

The primary goal of this paper is to demonstrate that Broadcom's Stingray SoC solution provides market-leading performance and latency, enabling a new class of next-generation Fabric-based Bunch of Flash (FBOF) appliances.

Key findings include:

- The Broadcom Stingray SoC provides NVMe flash performance and bandwidth in line with x86 solutions while integrating significantly more functionality:
  - An integrated 100G RNIC engine.
  - Octal 3 GHz Arm A72 cores.
  - Hardware accelerators for RAID/Crypto and De-dupe.
- Stingray delivers up to 2.6 MIOPS in an eight-drive setup, similar to x86 target performance.
  - Achieved at a much lower power: 35W (Stingray) versus 155W (E5-6134 + NIC).
- The Fabric latency adder is small in comparison to the overall read/write latency. The vast majority of applications using NVMe-oF storage are therefore expected to perform just as well as with direct-attached NVMe drives.
  - NVMe-oF latency of ~10 µs in the multi-initiator setup.
- Testing was completed with the generally available Broadcom SDK.
  - The product is ready for deployment.
  - Various switches have been tested in our lab including Arista, Dell, and Mellanox, demonstrating interoperability with commonly used data center switches.

Setup Details

Topology and Initiator Configuration

Three topologies are used during the test procedure:

1. Direct-connect of single initiator with Stingray storage target. For this topology, the following x86 hardware is used:

CPU	Intel Core i7-7820X Skylake-X 8-Core 3.2 GHz max
NIC	Broadcom P1100p 100G NIC

Along with the following kernel and driver configuration:

Kernel	4.14.0+ (4.14.0 with SPDK stability patches)
L2 Driver	bnxt_en - 1.9.2-214.0.111.0
RoCE Driver	bnxt_re - 214.0.109.0
NIC Firmware	214.0.111.0/pkg 214.0.111.0
Fio	fio-2.20
NVMe	1.1.89.g2ae8
SPDK / DPDK	18.01/17.11.0

2. A more general setup where the Stingray storage target is connected to the 100G switch and multiple initiators:
3. x86 storage target (using identical NVMe drives as in topology 2):

For topologies 2 and 3, eight initiators are used, each with the following HW configuration:

CPU	Intel Xeon Gold 5115 CPU 10-core 3.7 GHz max.
NIC	Broadcom P225p 2x25G NIC

The following SW configuration was used on the initiators:

Kernel	Ubuntu 4.17.19-041719-generic
L2 Driver	bnxt_en - 1.9.2-214.1.39.0
RoCE Driver	bnxt_re - 214.1.40.0
NIC Firmware	214.1.38.0/pkg 214.1.42.0
Fio	fio-3.12
NVMe	1.7
SPDK/DPDK	v18.10.1/18.08.0

Storage Target Configuration

The following software is used on the Stingray-based target used in topologies 1 and 2:

Kernel	4.14.79+gf2991e23f24b
L2 Driver (bnxt_en)	1.9.2-214.1.58.0
RoCE Driver (bnxt_re)	214.1.73.4
NIC Firmware	214.1.73.3
NVMe	1.6
SPDK/DPDK	v18.04/17.11.2

For the x86 storage target used in topology 3, the following HW configuration is used:

CPU	2x Intel Xeon Gold 6134 CPU 8-core 3.7 GHz max.
NIC	Broadcom P1100p 1x100G NIC

Along with the following software configuration:

Kernel	4.17.19-041719-generic
L2 Driver (bnxt_en)	1.9.2-214.1.58.0
RoCE Driver (bnxt_re)	214.1.73.4
NIC Firmware	214.1.73.3
NVMe	1.7
SPDK/DPDK	v18.04/18.02

NVMe Drives

For topology 1:

Make	HGST Ultrastar
Model	SN20
Capacity	1600 GB
Form Factor	U.2 2.5-inch Drive w/ StarTech U.2 to PCIe Adapter (PEX4SFF8639)
Sequential Read (128K)	3350 MB/s
Sequential Write (128K)	2100 MB/s
Random Read IOPs (4 KB)	835,000
Random Write IOPs (4 KB)	200,000
Read Latency	Not Specified
Write Latency	20 µs

Make	Intel
Model	Optane SSD 900P
Capacity	280 GB
Form Factor	HH-HL Add-in Card
Sequential Read (128K)	2500 MB/s
Sequential Write (128K)	2000 MB/s
Random Read IOPs (4 KB)	550,000
Random Write IOPs (4 KB)	500,000
Read Latency	10 µs
Write Latency	10 µs

For topologies 2 and 3:

Make	Samsung
Model	MZWLL800HEHP
Capacity	800 GB
Form Factor	U.2
Sequential Read (128K)	3300 MB/s
Sequential Write (128K)	1000 MB/s
Random Read IOPs (4 KB)	800,000
Random Write IOPs (4 KB)	160,000
Read Latency	90 µs
Write Latency	30 µs

A secure format is done on all the drives before starting testing, for example:

```
nvme format /dev/nvme0 --ses=1
```

The drives are not preconditioned afterwards since system performance is being tested instead of drive performance. Hence, being at steady-state drive performance was deemed not required. This also allows us to format the drives on demand if the performance seems irregular. In practice, there was a minimal difference in performance.

Ethernet Switch

Two 100G switches were used for testing: the Mellanox SN2100 (16-port) and Dell Z9100 (32-port).

Example Test Commands

SPDK perf

```
4K Random Read
perf -r "trtype:RDMA adrfam:IPv4
traddr:192.168.1.10 trsvcid:1023" -q 128 -s 4096
-w randread -d 2048 -t 60 -c 0xffff

4K Random Write
perf -r "trtype:RDMA adrfam:IPv4
traddr:192.168.1.10 trsvcid:1023" -q 128 -s 4096
-w randwrite -d 2048 -t 60 -c 0xffff

4K Random Read Latency
perf -r "trtype:RDMA adrfam:IPv4
traddr:192.168.1.10 trsvcid:1023
subnqn:nqn.2016-06.io.spdk:cnode0" -q 1 -s 4096 -
w randread -d 2048 -t 60 -c 0x1

4K Random Write Latency
perf -r "trtype:RDMA adrfam:IPv4
traddr:192.168.1.10 trsvcid:1023
subnqn:nqn.2016-06.io.spdk:cnode0" -q 1 -s 4096 -
w randwrite -d 2048 -t 60 -c 0x1
```

fio (Using SPDK Plugin)

When using fio, best performance was achieved using the SPDK fio plugin. We used a file-based fio configuration and a file example for one test is provided. Other tests are performed with minimal change in the configuration.

4K Random Read fio randread.fio 4K Random Write fio randwrite.fio 4K Random Read Latency fio randwrite.fio 4K Random Write Latency fio randwrite.fio	# cat randread.fio [global] rw=randread numjobs=1 bs=4k runtime=3600 ioengine=/root/spdk/examples/nvme/fio_plugin/fio_plugin direct=1 iodepth=128 time_based group_reporting norandommap=1 thread=1 [disk1] filename=trtype=RDMA adrfam=IPv4 traddr=192.168.1.218 trsvcid=1023 ns=1 [disk2] filename=trtype=RDMA adrfam=IPv4 traddr=192.168.1.218 trsvcid=1023 ns=2 [disk3] filename=trtype=RDMA adrfam=IPv4 traddr=192.168.1.218 trsvcid=1023 ns=3 [disk4] filename=trtype=RDMA adrfam=IPv4 traddr=192.168.1.218 trsvcid=1023 ns=4 [disk5] filename=trtype=RDMA adrfam=IPv4 traddr=192.168.1.218 trsvcid=1023 ns=5 [disk6] filename=trtype=RDMA adrfam=IPv4 traddr=192.168.1.218 trsvcid=1023 ns=6 [disk7] filename=trtype=RDMA adrfam=IPv4 traddr=192.168.1.218 trsvcid=1023 ns=7 [disk8] filename=trtype=RDMA adrfam=IPv4 traddr=192.168.1.218 trsvcid=1023 ns=8

BROADCOM®