PROJECT REPORT

Empirical evaluation of various RDSA technologies

CS546 – Parallel and distributed processing

November 22nd, 2019

Deepak Kumar Yadav - A20447332

Gurunath Reddy - A20447076

1. Introduction & Background Information

Large scale supercomputers and data centers often require exchanging data and messages among each other. A popular technology is called "Remote Data Storage Access" (RDSA). This technology enables a remote computer to directly access and modify the contents of local storage devices. This functionality greatly enhances the performance of data exchange, in comparison to traditional message passing interfaces. There are many software's that enable storage over fabric software, such as NVMe over fabric, SSD over fabric, NVRAM over fabric.

NVMe (non-volatile memory express) is a host controller interface and storage protocol created to accelerate the transfer of data between enterprise and client systems and solid-state drives (SSDs) over a computer's high-speed Peripheral Component Interconnect Express (PCIe) bus. This enables it to have high performance when compared to SSD or HDD. Below bar graph shows the bandwidth of the available 3 storages.[7]

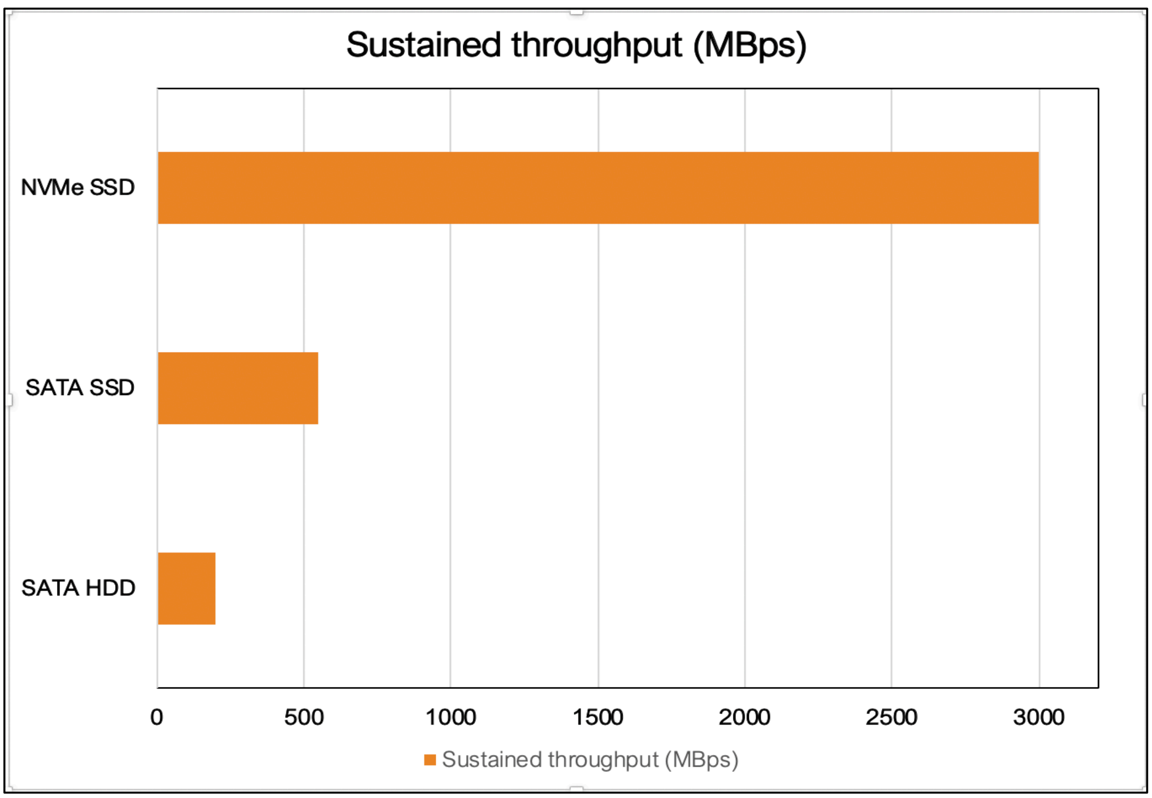

Figure 1.1

NVMe-oF uses the NVMe protocol to connect hosts to storage across a network fabric. NVMe-oF extends the benefits of NVMe across the data center, expanding the distance over which NVMe devices can be reached. And, depending on the fabric used and the existing infrastructure, it may not require a hardware or software upgrade.

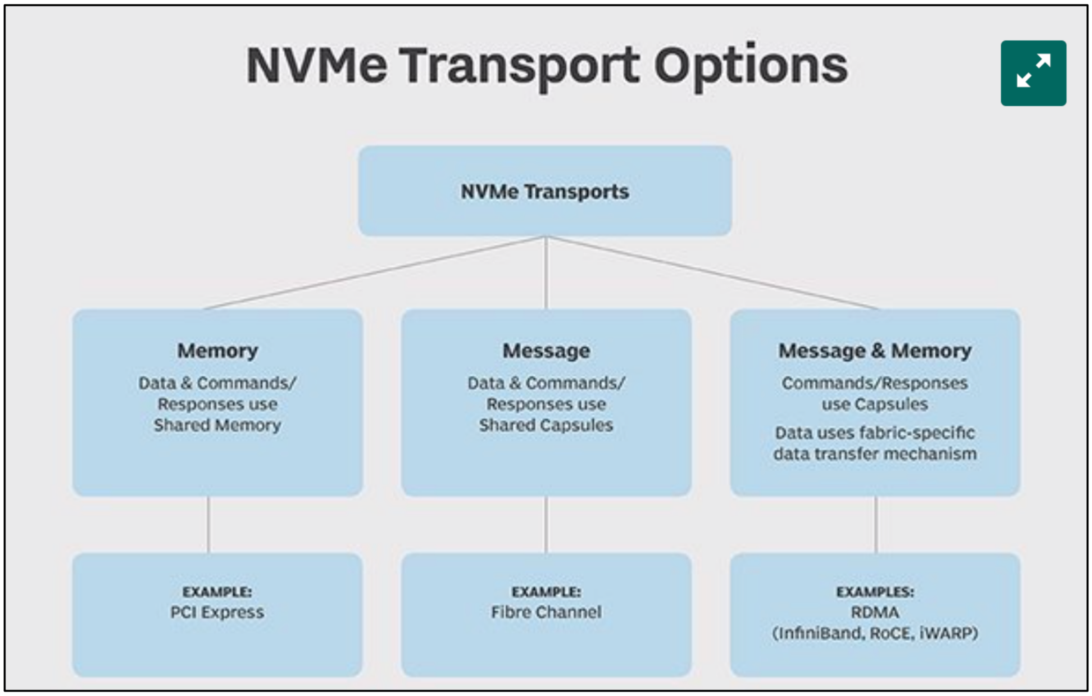

Figure 1.2

NVMe-oF lets an NVMe host communicate with a network-connected NVMe storage device, mapping commands and responses to the hosts shared memory. With it, NVMe-based devices can be accessed over greater distances, and more devicescan be looped in. One NVMe-oF advantage is it makes distributed data centers with all the capabilities of NVMe a reality[8].

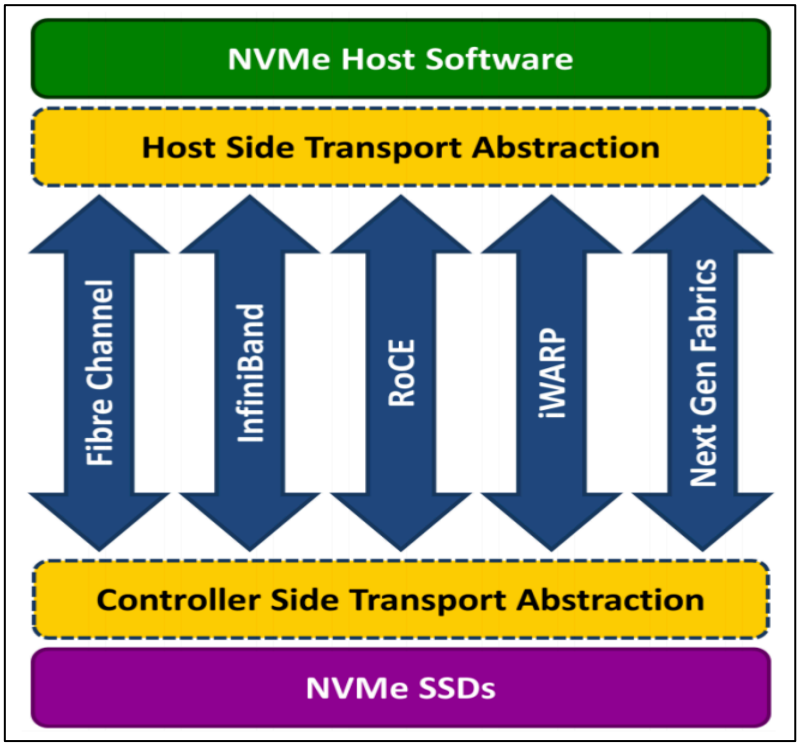

Figure 1.3

In a local NVMe implementation, NVMe commands and responses are mapped to shared memory in a host over the PCIe interface. However, fabrics are built on the concept of sending and receiving messages without shared memory between the end points. The NVMe fabric message transports are designed to encapsulate NVMe commands and responses into a message-based system by using "capsules" that include one or more NVMe commands or responses. The capsules or combination of capsules and data are independent of the specific fabric technology and are sent and received over the desired fabric technology.

For NVMe over Fabrics, the entire NVMe multi-queue model is maintained, using normal NVMe submission queues and completion queues, but encapsulated over a message-based transport. The NVMe I/O queue pair (submission and completion) is designed for multi-core CPUs, and this low-latency efficient design is maintained in NVMe over Fabrics.

2. Problem Statement

Evaluation of over fabric solution with different workloads for different operations (read and write) using single and multiple drives.

3. Solution

3.1. Metric

NVMe SSD (NVM Express Solid-State Drive) over PCIe (Peripheral Component Interconnect Express) has become popular in client and data center markets since it provides high throughput and low latency than traditional SATA SSDs. As SSD capacity increases, complex internal SSD operations such as garbage collection, error correction, etc. are needed and this can induce performance variations [1]. In a storage system, a predictable I/O latency which is one of the most important factors is always desired, however, the SSD performance variations make the I/O latency prediction harder.

3.2. Design

Below is the architecture design for the solution. It has one initiator and two targets that are connected using RoCE fabric.

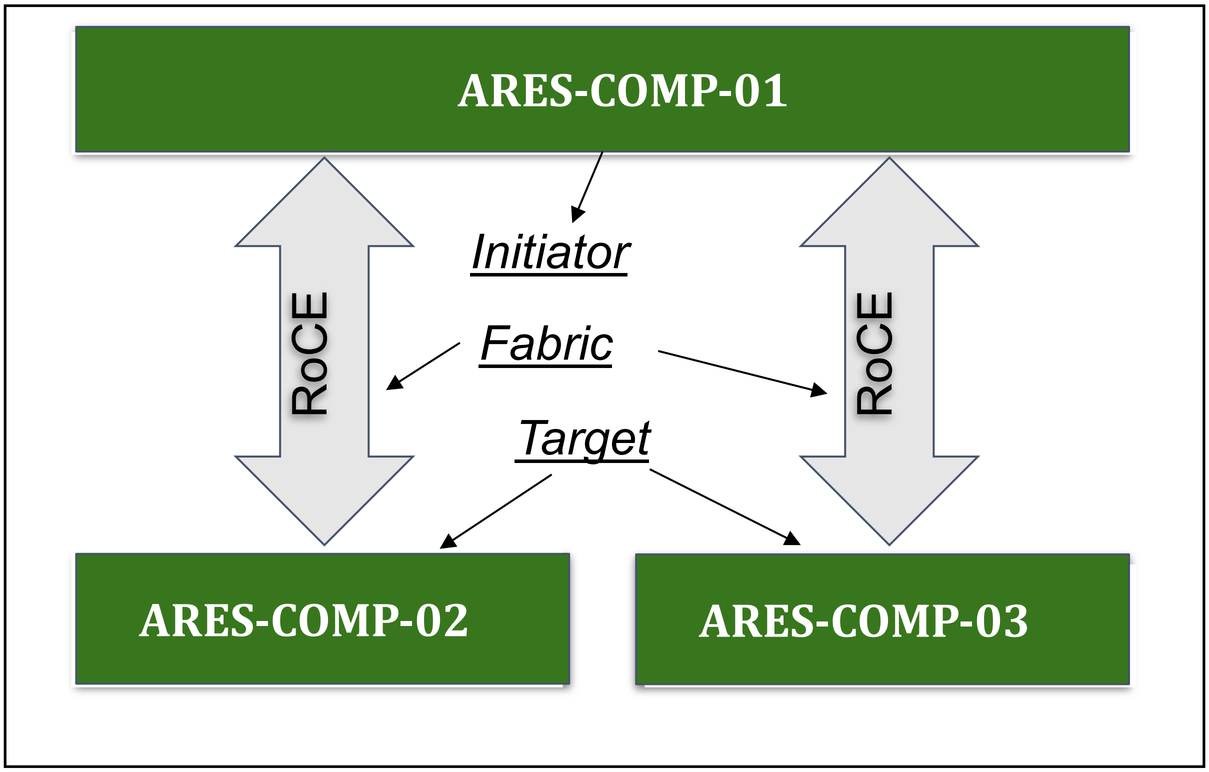

Figure 3.2.1

3.3. I/O Phase Detection

3.3.1. Use Cases

There are several use cases for NVMe over Fabrics. One can easily imagine a storage system comprised of many NVMe devices, using NVMe over Fabrics with either an RDMA or Fibre Channel interface, making a complete end-to-end NVMe storage solution. This system would provide extremely high performance while maintaining the very low latency available via NVMe.

Another implementation would use NVMe over Fabrics to achieve the low latency while connected to a storage subsystem that uses more traditional protocols internally to handle I/O to each of the SSDs in that system. This would gain the benefits of the simplified host software stack and lower latency over the wire, while taking advantage of existing storage subsystem technology.

3.3.2. Source Code

NVMe CLI[5] is an option available on Linux operating system to interact with available NVMe SSDs. In the current implementation, shell scripts which internally call nvme commands[6] where used to benchmark, interact with NVMe devices. Below, is a screen shot of the used shell scripts. However, please refer the codes attached in blackboard for all the codes used within the project.

nvme_1_client_read_10G_write_1G.sh

```
#!/bin/bash

# Initialize variables / read from CLI
DEVICE="/dev/nvme0n1"
START_BLOCK=0
BLOCK_COUNT=0
# 1GB = 1073741824
DATA_SIZE_1G=1073741824
DATA_SIZE_10G=10737418240
METADATA_SIZE=64
READ_OUTPUT_FILE="read_from_nvme_target.txt"
WRITE_OUTPUT_FILE="write_from_nvme_target.txt"

# ---------------- Connection
# This will create a connection to a remote system specified by --traddr and --trsvcid
# trasport = rdma(N/W fabric is RDMA N/W i.e., ROCA, iWARP, Infiniband ) | fc(Fibre #channel n/w) | loo
echo "Starting connection"
    nvme connect --transport=rdma --traddr=192.168.1.1

# ----------------  Get Namespace ID
# Assuming no Namespace changes are to be done (Attach / Create / Format)
# We can get the namespace id for the nvme block device
    nvme get-ns-id ${DEVICE}

echo "Read 10GB and Write 1GB simultaneously STARTED"
./nvme_reader.sh "$START_BLOCK" "$BLOCK_COUNT" "$METADATA_SIZE" "$DATA_SIZE_10G" "$READ_OUTPUT_FILE" &
./nvme_writer.sh "$START_BLOCK" "$BLOCK_COUNT" "$METADATA_SIZE" "$DATA_SIZE_1G" "$WRITE_OUTPUT_FILE"
echo "Read and write Completed"

echo "Disconnect"
# ---------------- Disconnect
    nvme disconnect
```

Figure 3.3.2.1

3.4 Implementational Details

RDMA can be implemented over different link layer protocols. It was originally associated with InfiniBand [2], requiring special-purpose interconnect. Two recent Ethernet- base alternatives – RoCE and iWARP – provide a more cost-efficient solution that is expected to increase RDMA adoption in the data center domain. RoCE (RDMA over

Converged Ethernet) [3] is based on InfiniBand transport over Ethernet and uses RDMA over Ethernet and UDP/IP frames. iWARP (interent Wide Area RDMA Protocol) [4] uses RDMA over a connection-oriented transport such as TCP.

3.4.1 Test cases

To evaluate the performance of different operations we have designed the below test cases which involves two NVMe operations (read and write) that have been tested with different workloads (1GB, 10GB). The below workloads have been tested.

- **One client trying to read 1GB**
  In this we have one client trying to read 1GB.
- **One client trying to read 10GB**
  In this we have one client trying to read 10GB.
- **One client trying to write 1GB**
  In this we have one client trying to write 1GB.
- **One client trying to write 10GB**
  In this we have one client trying to write 10GB.
- **Multiple clients trying to read 1GB in parallel**
  In this we have two clients trying to read 1GB data simultaneously.
- **Multiple clients trying to write 10GB in parallel**
  In this we have two clients trying to read 10GB data simultaneously.
- **Multiple clients trying to write 1GB in parallel**
  In this we have two clients trying to write 1GB data simultaneously.
- **Multiple clients trying to write 10GB in parallel**
  In this we have two clients trying to write 10GB data simultaneously.
- **Read 1GB from 2 drives in parallel and read 10GB from 2 drives in parallel.**
- **One client tries to read 10GB data from target-1 and another client tries to read data from client-2**.
- **Write 1GB from 2 drives in parallel and read 10GB from 2 drives in parallel.**
- **One client tries to write 10GB data from target-1 and another client tries to write data from client-2.**
- **One client trying to read 1GB and another client trying to write 1GB in parallel.**
- **One client trying to read 10GB and another client trying to write 10GB in parallel.**
- **One client trying to read 1GB and another client trying to write 10GB in parallel.**
- **One client trying to read 10GB and another client trying to write 1GB in parallel.**

In all the above test cases we capture the latency in microseconds. Every operation specified above will be executed 5 times and then the we compute the average from all the 5 outputs. This is done for consistency purposes and also to eliminate any deviations.

3.4.2 Hardware

Hardware used for the implementation are three nodes from ARES cluster. Figure 3**.4.2.1** has the hardware level details of the machines **ares-comp-01** is the initiator node which communicates with **target-1** for most of the test cases. **Target-2** is used for few of the test cases.

ITEM	DESCRIPTION (INITIATOR)	DESCRIPTION (TARGET-1)	DESCRIPTION (TARGET-2)
Server Name	ares-comp-01	ares-comp-02	ares-comp-03
CPU	Dual Intel(R) Xeon Scalable Silver 4114 @ 2.20GHz		
Memory (GB)	96		
Operating System	Cent OS		
Storage	Samsung 960 Evo 250GB NVMe SSD		
NIC	25 Gbps Mellanox ConnectX4		

Figure 3.4.2.1

3.5. Experimental and Evaluation Results

For the test scenarios described above we have captured the latency and below values were obtained. For every operation we have 5 values are obtained as we run the same operation 5 times to were generated.

Figure-3.5.1 shows the values captured when single read operation and single read operation was performed for data sizes 1GB and 10GB.

Figure-3.5.2 show the values captured when read and write had equal load of 1GB and 10GB and also when read and write had mixed (unequal) workloads.

Read 1GB	5709
	5662
	5717
	5742
	5727
	5711
Read 10G	5604
	5681
	5666
	5635
	5613
	5639
Write 1GB	5603
	5586
	5559
	5486
	5547
	5556
Write 10GB	5653
	5633
	5600
	5612
	5528
	5605

Figure-3.5.1

Read Write 1G	5701	1911
	5673	1761
	5796	5673
	5701	1794
	5693	1924
	5712	2612
Read Write 10G	5699	5803
	5684	5573
	5641	5642
	5664	5693
	5646	1898
	5666	4921
Read 1G Write 10Gb	5619	5717
	5655	5626
	5675	5723
	5703	5654
	5714	5588
	5673	5661
Read 10G wtite 1Gb	5603	5721
	5635	5545
	5653	5716
	5638	5679
	5643	5673
	5634	5666

Figure-3.5.2

Figure-3.5.3 shows the values captured when we have more than one client trying to perform the same operation with equal loads in parallel.

Figure-3.5.4 shows the values captured when we have more than one nvme drive available for interaction i.e., one client is trying to read/write data from ***nvme-drive-1*** and the other client from the same initiator is trying to read/write data from ***nvme-drive-2*** as we can see in the test case the load was equal.

Read 1GB parallel	5238	5740
	5611	159
	5699	144
	5621	1482
	5745	175
	5582	1540
Read 10GB parallel	5745	144
	5738	177
	5710	163
	5733	1888
	5697	161
	5724	506
Write 1GB parallel	2375	5668
	5617	131
	5595	5595
	5638	131
	5581	5590
	4961	3423
Write 10GB parallel	5022	6806
	5617	1761
	5639	96
	5583	95
	5633	99
	5498	1771

Figure-3.5.3

Read 1GB 2 drives	5733	5796
	5796	5728
	5884	5672
	5668	5706
	5929	5581
	5802	5696
Write 1GB 2 drives	5629	55399
	5655	54230
	57015	5614
	142	5673
	5689	137
	14826	24210
Read 10GB 2 drives	5825	5711
	5681	5845
	5781	5696
	5836	5598
	5755	5726
	5775	5715
Write 10GB 2 drives	183	5647
	155	5639
	5722	183
	5650	178
	188	5594
	2379	3448

Figure-3.5.4

From the above tables we obtained the below graphs. We have clubbed the data into graphs based on operation being performed and drivers involved.

Figure 3.5.5, Figure 3.5.6, shows the average latency for read/write operations with single and multiple clients with a single nvme driver. Latenc-1 shows the latency for the second parallel read/write operation.

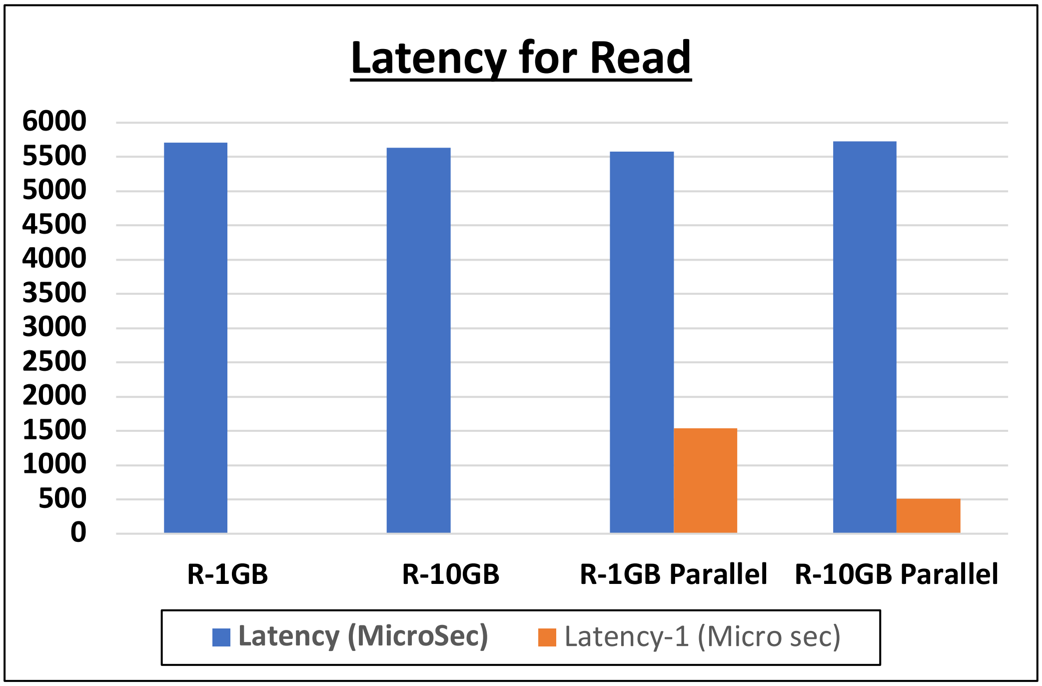

Figure 3.5.5

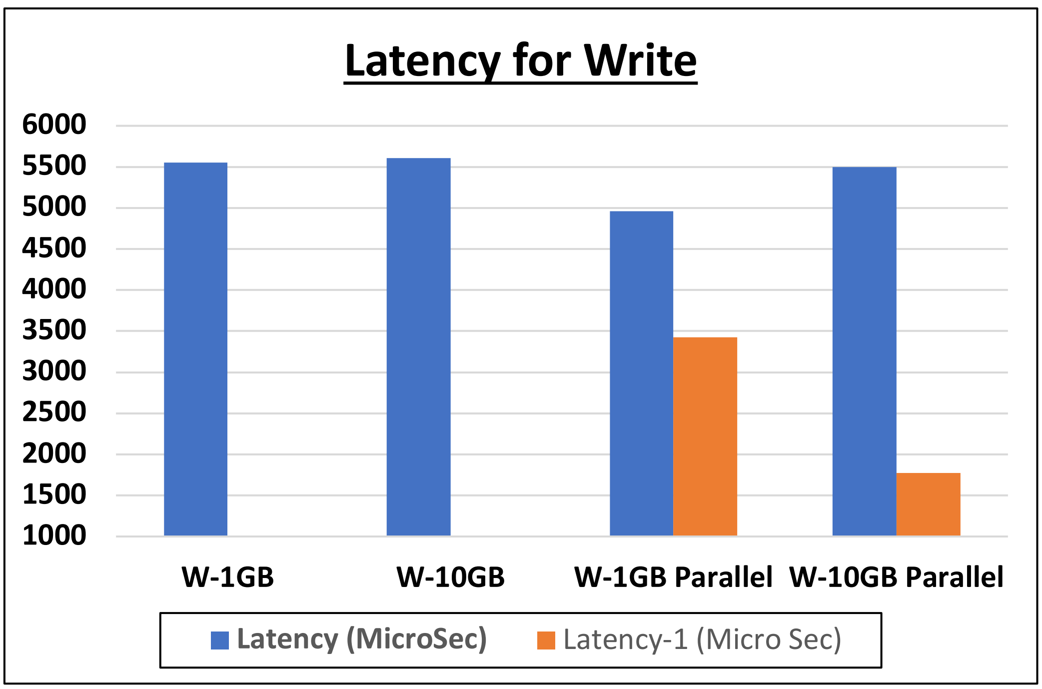

Figure 3.5.6

Figure 3.5.7 shows the latency for read and write operations performing in parallel contacting single NVMe drive. **Figure 3.5.8** shows the read operation using multiple NVMe drives.

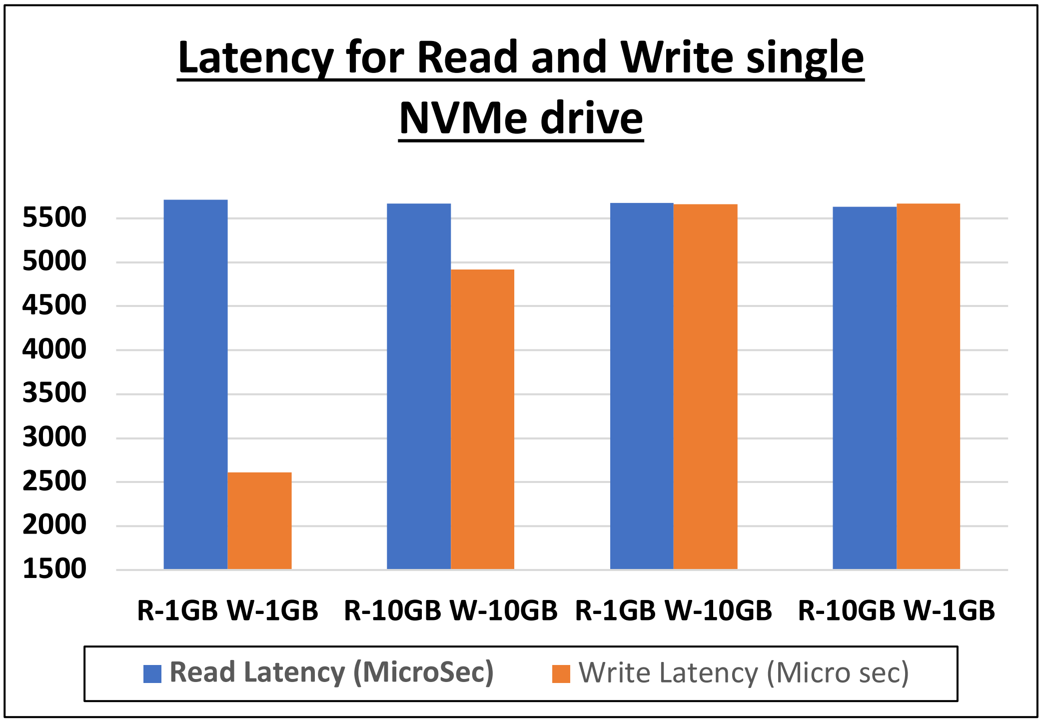

Figure 3.5.7

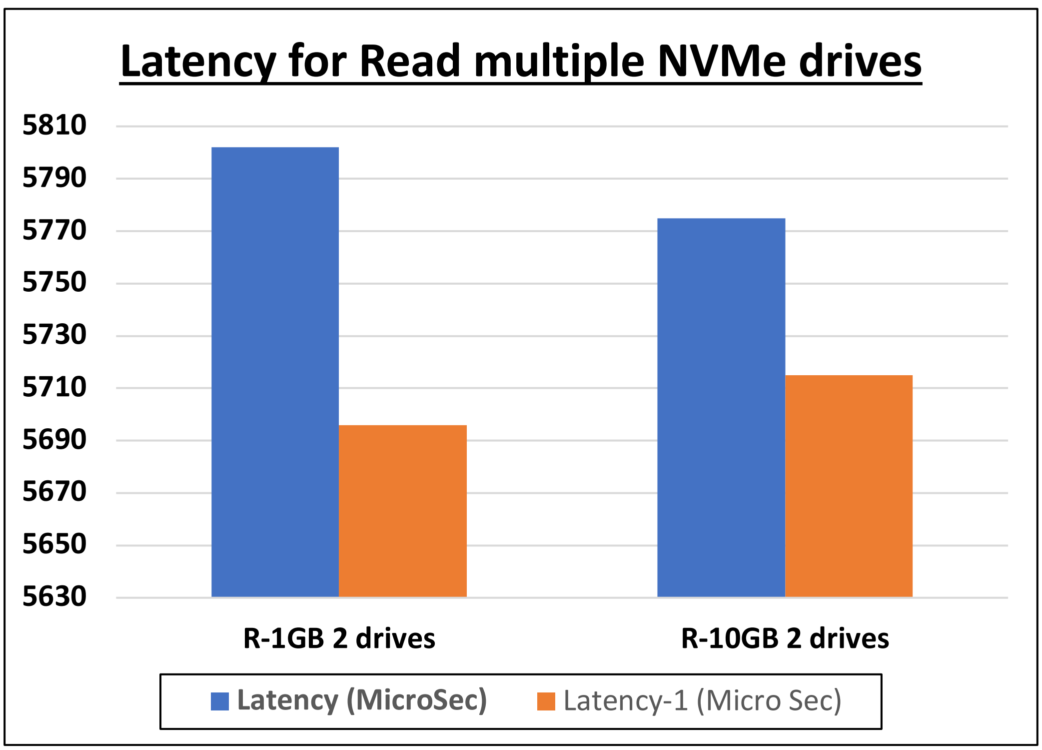

Figure 3.5.8

Figure 3.5.9 shows the write operation using two NVMe drives.

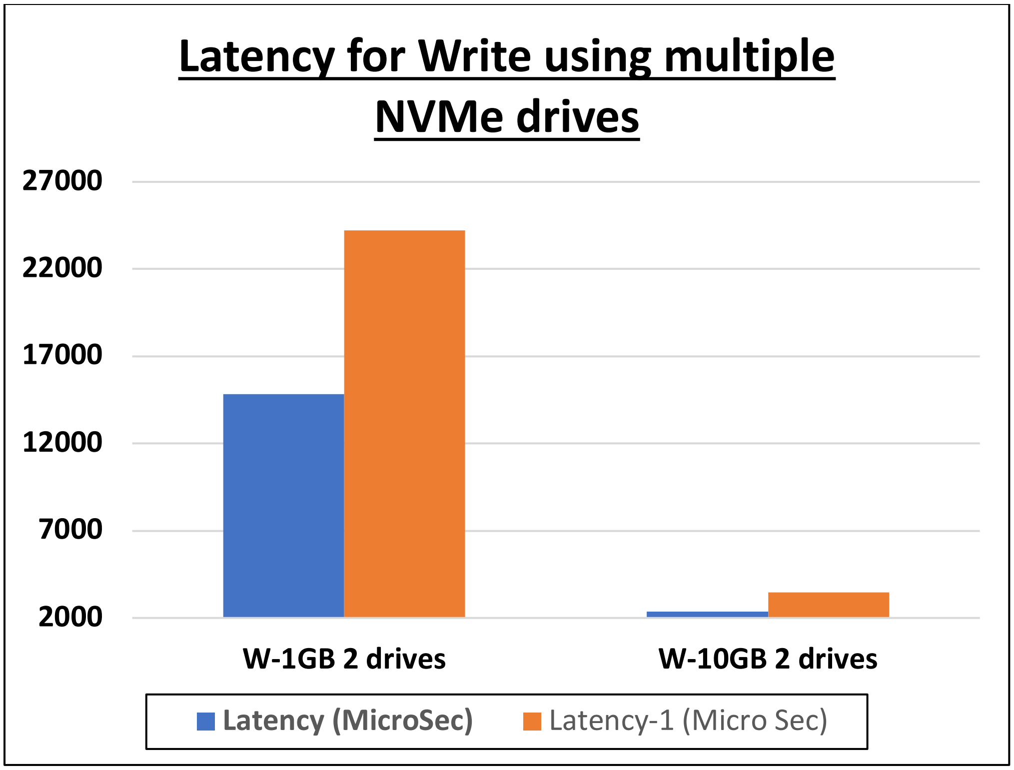

Figure 3.5.9

4. Conclusions

NVMe over Fabrics is poised to extend the low-latency efficient NVMe block storage protocol over fabrics to provide large-scale sharing of storage over distance. NVMe over Fabrics maintains the architecture and software consistency of the NVMe protocol across different fabric types, providing the benefits of NVMe regardless of the fabric type or the type of non-volatile memory used in the storage target. The next few years will be very exciting for the industry!

- For single drive NVMe write operations are faster than read operations.
- For multiple drives NVMe read operations are faster than write operations.
- Network / cache can influence the read operation. Below figure shows that consecutive reads are faster

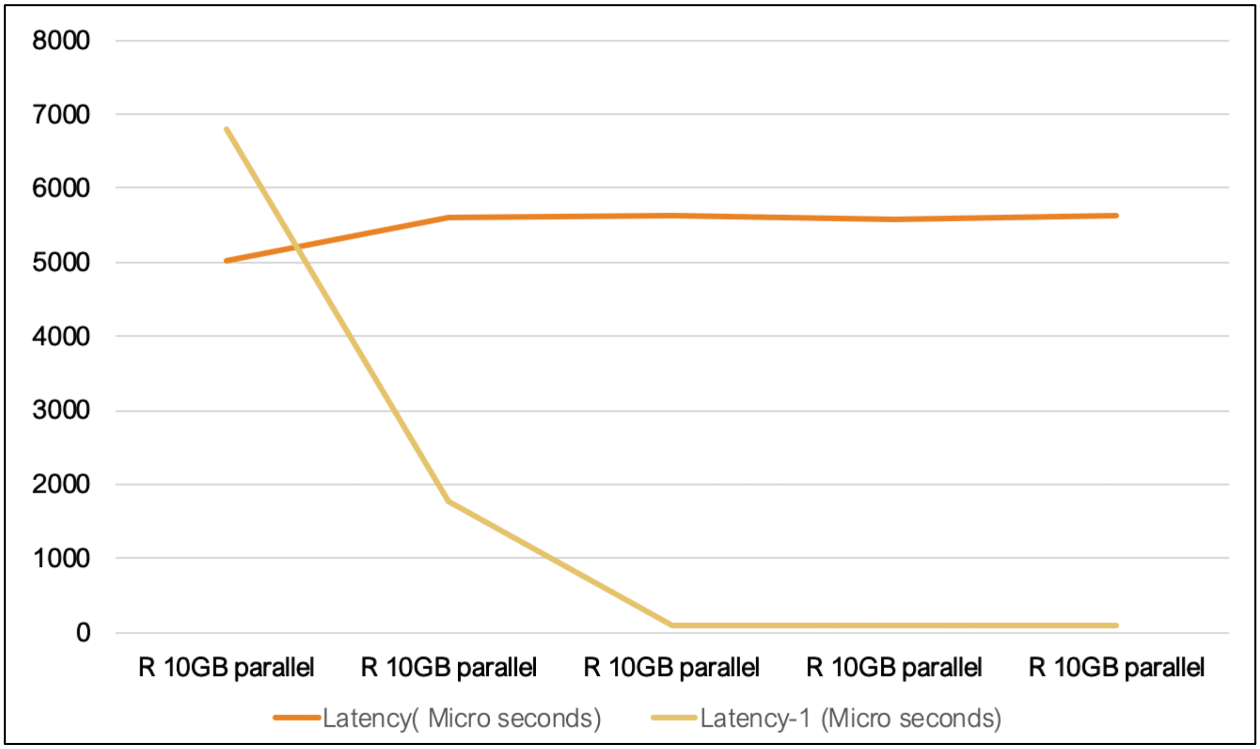

Figure 4.1

5. References.

[1] Mingzhe Hao, Gokul Soundararajan, Deepak Kenchammana- Hosekote, Andrew A.Chien, and Haryadi S.Gunawi, "The Tail at Store: A Revelation from Millions of Hours of Disk and SSD Deployments," in FAST '16.

[2] Jiuxing Liu, Jiesheng Wu, Sushmitha P Kini, Pete Wyckoff, and Dhabaleswar K Panda. High performance RDMA-based MPI implementation over InfiniBand. In Proceedings of the 17th annual international conference on Supercomputing, pages 295–304. ACM, 2003.

[3] David Cohen, Thomas Talpey, Arkady Kanevsky, Uri Cummings, Michael Krause, Renato Recio, Diego Crupnicoff, Lloyd Dickman, and Paul Grun. Remote direct memory access over the converged enhanced ethernet fabric: Evaluating the options. In High Performance Interconnects, 2009. HOTI 2009. 17th IEEE Symposium on, pages 123–130. IEEE, 2009.

[4] RDMA Consortium. Architectural specifications for RDMA over TCP/IP. Technical report.

[5] NVME CLI git hub https://github.com/linux-nvme/nvme-cli

[6] NVME CLI Commands man list https://www.mankier.com/package/nvme-cli

[7] Comparison in between HDD vs SSD vs NVMe SSD https://unihost.com/help/nvme-vs-ssd-vs-hdd-overview-and-comparison/

[8] NVM Express. NVM Express over Fabric 1.0. http://www.nvmexpress.org/, 2016.