Impact of Commodity Networks on Storage Disaggregation with NVMe-oF

Arjun Kashyap[2(✉)], Shashank Gugnani[1], and Xiaoyi Lu[2]

[1] The Ohio State University, Columbus, USA
gugnani.2@osu.edu
[2] University of California Merced, Merced, USA
{akashyap5,xiaoyi.lu}@ucmerced.edu

Abstract. NVMe-based storage is widely used for various data-intensive applications due to their high bandwidth and low access latency. The NVMe-over-Fabrics (NVMe-oF) protocol specification provides efficient remote access to NVMe-SSDs over storage networking fabrics. NVMe-oF provides the opportunity to make storage disaggregation practical by reducing the cost of remote access. Unfortunately, the performance characteristics of different NVMe-oF networking protocols are not well understood. In this paper, we propose a four dimensional (network protocol, I/O pattern, I/O size, and number of cores) evaluation methodology to understand NVMe-oF performance over various commodity networks. We conduct comprehensive microbenchmark analyses using the user-space Intel SPDK library to compare the TCP, IPoIB, RoCE, and RDMA transports. Our analysis reveals interesting, and often counter-intuitive insights and performance tradeoffs among the different transports. We find that InfiniBand with native RDMA is able to deliver the best performance among all tested networking protocols in most experiments. Contrary to expectation, IPoIB could achieve better CPU utilization and lower tail-latency for large I/O operations in some of our experiments. We believe that our analysis helps gain insight into the deployment implications of NVMe-oF in datacenters.

Keywords: NVMe-over-Fabrics · SPDK · Disaggregated storage · Performance characterization

1 Introduction

Storage disaggregation is gaining popularity in cloud datacenters recently [13–15] due to the ability to scale and manage storage and compute results with increased flexibility and better resource utilization. A large body of work [4,5,14–16] has looked at how storage systems can be efficiently designed to take advantage of disaggregated storage. Other works [10,13,26] have looked at improving the network costs of disaggregation. Taking this trend into consideration, it is imperative to gain a better understanding of the impact of commodity networks on storage disaggregation.

Fast NVMe [18] drives are becoming ubiquitous in datacenters. Their low latency and inherent parallelism make them a perfect fit for cloud storage. The recently

This work was supported in part by NSF research grant CCF #1822987.
This work was done by the authors A. Kashyap and X. Lu while at Ohio State University.

F. Wolf and W. Gao (Eds.): Bench 2020, LNCS 12614, pp. 41–56, 2021.
https://doi.org/10.1007/978-3-030-71058-3_3

proposed NVMe over Fabrics (NVMe-oF) standard [19] allows fast remote access to NVMe devices over various network transports. NVMe-oF has overtaken Serial Attached SCSI (SAS) and Serial AT Attachment (SATA) as the state-of-the-art approach to access storage remotely. The standard was designed to leverage the increased parallelism available in modern solid state drives (SSDs) and networks, such as InfiniBand (IB) and Remote Direct Memory Access (RDMA) over Converged Ethernet (RoCE). Studies [2,3] have shown that NVMe-oF has extremely low overhead at the application level compared to traditional I/O paths. However, the standard allows several network protocols to be used for remote I/O and prior work has not considered the breadth of available protocols.

We believe that there is a lack of studies which evaluate the tradeoffs of different networks and try to gain an in-depth understanding of their performance characteristics on storage disaggregation. This is even more important when considering NVMe-oF because the network overhead constitutes a significant portion of remote access latency. Protocols such as SAS and SATA were designed for spinning disks where data storage was significantly slower than network I/O. Therefore, it was not beneficial to study the impact of network protocols until now. For cloud vendors, there is a lack of sufficient information to make informed decisions on the network protocol to deploy in datacenters. For instance, the tail latency characteristics of networks are not known. Therefore, it is hard to choose which network to use to achieve a given latency service level objective. We seek to solve this information gap through an in-depth performance characterization.

In this paper, we propose a four dimensional (network protocol, I/O pattern, I/O size, and number of cores) evaluation methodology to understand NVMe-oF over various commodity networks. We conduct a set of comprehensive microbenchmark analyses using the userspace Intel SPDK library to compare the TCP, IP-over-IB/IPoIB, RoCE, and RDMA transports. Our analysis reveals interesting insights and performance tradeoffs between the different transports.

To summarize, the contributions of our work are as follows:

- An extensive four dimensional (network protocol, I/O pattern, I/O size, and number of cores) characterization methodology of NVMe-oF protocols
- In-depth NVMe-oF protocol performance characterization and tradeoff analysis using the userspace Intel SPDK library
- The community can leverage the findings in this paper to make informed decisions on the transport type to deploy for NVMe-oF based on latency, bandwidth, and CPU utilization constraints of the system in datacenters

We conduct extensive experiments on a local cluster with NVMe-SSDs and several networks connecting each server. Overall, our findings reveal interesting insights – RoCE is able to achieve bandwidth similar to IPoIB and IB for most of the write workloads despite having only 40 Gbps network card when compared to 100 Gbps card for InfiniBand. This shows that moderate network bandwidth is sufficient to achieve good performance for write operations on current-generation NVMe-SSDs. Overall, we find that InfiniBand with native RDMA can deliver the best performance among all tested networking protocols in most experiments.

Contrary to expectation, IPoIB could achieve better CPU utilization and lower tail-latency for large I/O operations in some of our experiments. IPoIB has similar CPU

utilization to TCP with ∼45% CPU time in user mode and ∼50% in kernel/system mode but its bandwidth are similar to RoCE and IB whose CPU utilization is ∼92% in user mode. Apart from TCP, the other three network protocols - RoCE, IPoIB, and IB have similar average latency but varying tail latency depending upon the I/O size. Increasing the number of cores on client and target for NVMe-oF protocol does not change the throughput for TCP, RoCE, and IB transports for random read operation but raises the latency for all transports.

The rest of this paper is organized as follows. Section 2 discusses background on storage disaggregation, NVMe-oF, and SPDK. Section 3 presents the characterization methodology we propose, Sect. 4 presents our in-depth experimental analysis, and Sect. 5 summarizes our analysis with a discussion. Section 6 discusses related work and Sect. 7 presents the conclusion and future work.

2 Background

In this section, we discuss storage disaggregation, the NVMe-oF standard, and the SPDK library.

2.1 Storage Disaggregation and NVMe-oF

Storage disaggregation is a commonly used technique to improve resource utilization in cloud environments. In this approach, storage devices are physically separated from compute servers and are typically accessed over the network using a remote access protocol. This separation allows applications to only provision storage in an ad-hoc manner and scale it up or down depending on utilization. Disaggregation comes at the cost of the network overhead of remote access and indeed a lot of research [10,13,26] has looked into limiting the performance overheads of remote access. Nevertheless, the flexibility and resource savings of disaggregation are worth the overhead.

The NVMe-oF protocol specification [19] allows remote access of NVMe devices over different transports/network protocols. It allows fast access to any remote NVMe device via NVMe block storage protocol over the network as seen in Fig. 1. NVMe-oF currently supports three types of transports:

* NVMe over Fabrics using RDMA (Infiniband, RoCE, and iWARP)
* NVMe over Fabrics using Fibre Channel
* NVMe over Fabrics using Ethernet (TCP)

The specification terms NVMe-oF client/initiator as the node which contains drivers to send NVMe commands over the transport and NVMe-oF target as the node which contains drivers for local NVMe storage device (connected via PCI-e) and different transports.

2.2 Intel SPDK

Intel SPDK [25] is a userspace library that provides NVMe drivers, storage protocols (iSCSI target, NVMe-oF target, and vhost-scsi target), and storage services (blobstore

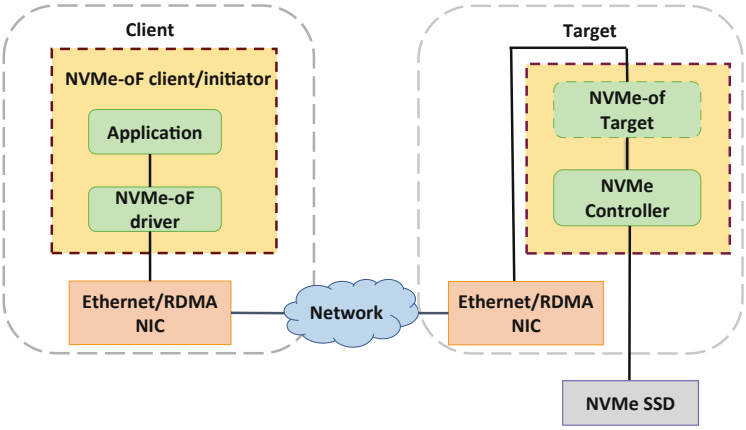


Fig. 1. NVMe-oF architecture

and block device abstraction layer). It uses polling rather than interrupts to complete asynchronous I/O operations via queue pairs. It provides a userspace NVMe-oF application which consists of NVMe-oF target [7] and NVMe-oF client/initiator drivers. The client is responsible for establishing a connection and submitting I/O requests (similar to local NVMe devices) over the network to an NVMe subsystem exposed by an NVMe-oF target. Apart from avoiding costly syscalls and data copy overheads, SPDK is also able to scale and avoid synchronization overheads by pining its connections to CPU cores.

3 Characterization Methodology

Our methodology involves evaluating NVMe-oF across four dimensions - transport type or network protocol, I/O pattern or workload, number of cores involved in I/O, and I/O size as seen in Fig. 2. The network transport under consideration are TCP, RoCE, IPoIB, and IB as these are widely available in today's datacenters. We did not consider iWARP and fibre channel because we did not have access to these networks. Moreover, they are not widely used in datacenters. I/O pattern studied are sequential/random read/write and read-write and I/O sizes are varied from 1KB to 1MB. Number of cores represent the threads (one thread is pinned to one core) performing I/O at NVMe-oF client and NVMe-oF target side. I/O pattern, I/O size, and the number of cores represent most of the workloads different applications generate while reading or writing from/to NVMe-SSDs.

For benchmarking NVMe-oF we choose SDPK's NVMe-oF implementation [25] as it provides both a user-space NVMe-oF target and client and is the current state-of-the-art approach. The SPDK NVMe-oF target uses NVMe driver in polled-mode for I/Os. Being in user-space, it helps us avoid software processing overheads and obtain the actual latency of I/O transactions. The metrics reported are throughput/bandwidth, average and tail (p50, p90, p99, p999, and p9999) latency, client CPU utilization, and SSD parallelism or concurrency. These metrics were chosen as they help in quantifying

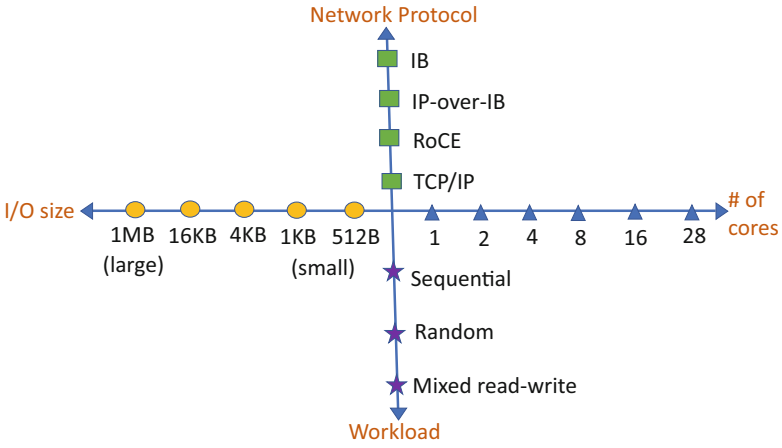


Fig. 2. Four-dimensional characterization approach

the performance and efficiency of various network transports and are typically the most important for cloud vendors and applications. Throughput, bandwidth, and latency are measured using the NVMe perf tool [20] as it has the capability to generate various workloads (sequential/random read/write and read-write), vary queue depth, I/O size, pin threads to CPU cores, and specify different transport types (ethernet and RDMA).

In our experiments, first we find out the maximum bandwidth of each transport type by trying out different combinations of I/O size, workload, and queue depth while performing I/Os. The observation here was that higher queue depth always provided higher bandwidth keeping all other factors same. Hence, all of our experiments are performed at a queue depth of 128. Even at a queue depth of 128, the system is not able to achieve its peak bandwidth when only one core is involved in I/O at both NVMe-oF client and target side. To fully utilize/saturate the network and SSD, we differ the number of cores and noticed bandwidth scaling with number of cores for IPoIB, though not linearly. Once we obtain the maximum throughput of all four transport types, we decide to study the variation in maximum bandwidth according to I/O pattern or workload as workload type has a profound impact on the bandwidth. Keeping other factors, like number of cores and queue depth constant, we evaluated the change in bandwidth for 1 KB and 4 KB I/O sizes. We choose these I/O sizes as it is representative of the typical size of objects in systems, such as key-value stores. Efficiency of the network transports is determined by the latency and CPU utilization at peak bandwidth. Average, p90, p99, p999, and p9999 latencies for each transport were noted when system has maximum throughput. Again, the increase in latency is proportional to the I/O size. The CPU utilization of a transport was broken down into the time spent by an I/O operation in user/system/idle mode. NVMe-SSDs support multiple levels (channel, plane, chip, and die) of parallelism. We calculate the concurrency/parallelism each transport could attain by varying the queue depth. This helps us to learn how well each network transport can exploit the parallelism within an NVMe-SSD. These metrics could allow a user of NVMe-oF protocol to determine a suitable transport type for their application based on its operating environment and latency constraints.

4 Evaluation

In this section, we conduct experiments on four networking protocols TCP, RoCE, IPoIB, and IB (native RDMA) to evaluate the performance of SPDK's NVMe-oF storage protocol over RDMA and Ethernet fabrics. We use the perf [20] tool of SPDK to obtain bandwidth, throughput, and latency results across various dimensions- I/O pattern, I/O size, and number of cores. The number of cores indicate the total threads (each thread pinned to one core) involved in performing I/O operations at NVMe-oF client and target side. The I/O sizes considered for the study range from 512 B to 1 MB. The I/O pattern or workload considered are sequential read, sequential write, random read, random write, read-write and random read-write. Like other studies [21,22] the ratio of reads to write in both read-write and random read-write operations were 70:30.

For all the experiments, the queue depth is set to 128 unless specified otherwise because this value of queue depth provides maximum throughput. The configuration of NVMe-oF target [8] is done by JSON-RPC interface provided by SPDK and the perf tool is run on the client. Each experiment is performed three times and the average value is reported.

4.1 Experimental Setup

Our cluster consists of two Linux servers directly connected, each equipped with a two-socket, 28-core Intel broadwell CPU (E5-2680v4@2.40 GHz), 512 GB DRAM, an Ethernet (1 Gbps) NIC, a ConnectX-3 RoCE (40 Gbps) NIC, and a ConnectX-5 IB-EDR (100 Gbps) NIC. One server has an Intel DC P3700 NVMe-SSD attached via PCIe and acts as the NVMe-oF target while the other server is used as the NVMe-oF client performing I/O reads and writes over the network. We use SPDK v20.04.1 for all of our evaluations. In our experiments, the legend *TCP* refers to TCP over 1 GbE, *RoCE* refers to RoCE over 40 GbE, *IPoIB* refers to IP-over-InfiniBand over a 100Gb IB card, and *IB* refers to native RDMA over a 100 Gb IB card.

4.2 Bandwidth Evaluation

First, we wish to discover the peak throughput for all transports as we vary the number of cores or workload or I/O size. Thus, we vary one of the factors and keep the others fixed. The number of cores/threads are set to 28 as maximum bandwidth is achieved when all 28 cores are performing I/O operations. We study the variation in bandwidth with respect to I/O size in Fig. 3 for random read operation with number of threads/cores set to 28 (full subscription) at both NVMe-oF target and client. The peak bandwidth for this configuration is 3599 MiB/s at 1 MB I/O size for IB transport and 3611 MiB/s for local SSDs. IPoIB has a slightly lower peak bandwidth of 3426 MiB/s. An interesting observation is that RoCE, IPoIB, and IB are able to almost achieve their respective peak bandwidths after I/O size crosses 4 KB. Thus, small I/O sizes are sufficient to achieve near peak bandwidth of an NVMe SSD over the network. TCP transport has the lowest peak bandwidth at 466 MiB/s which is 13% of the peak bandwidth achieved by other transports. This is because of the low bandwidth of the ethernet NIC.

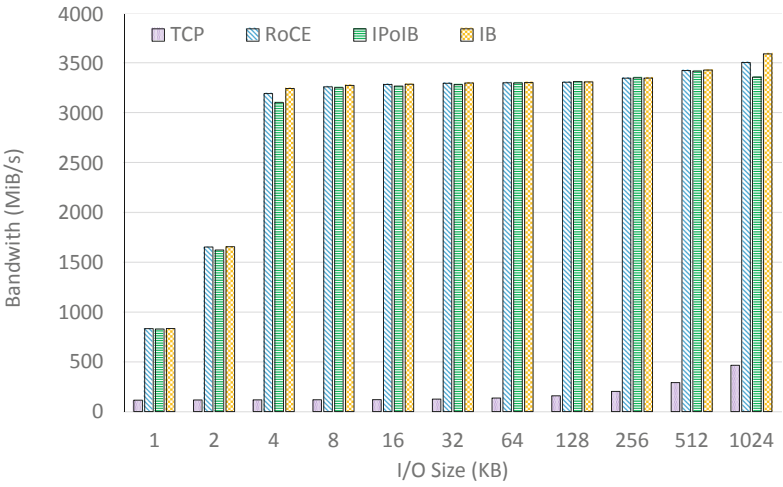


Fig. 3. Bandwidth with varied I/O sizes at full subscription for random read I/O

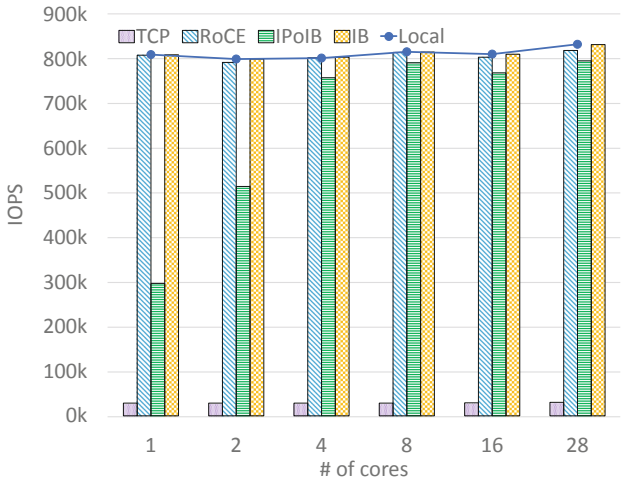
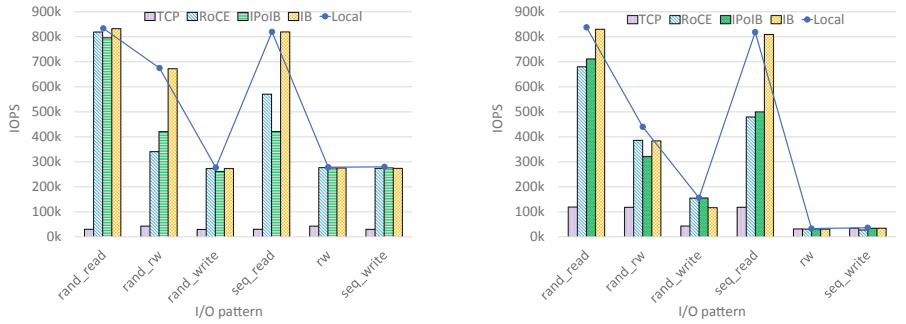


Fig. 4. Throughput with varied cores/threads for 4KB random read I/O

Second, we discuss the effects of number of cores/threads on throughput. For this experiment, we equally vary the number of threads performing I/O at the NVMe-oF client and target between 1–28 and measure the bandwidth for random read I/O at 4KB. We observe from Fig. 4 that the throughput/IOPS does not scale with the increase in the number of cores for TCP transport type and remains fixed at maximum throughput for RoCE and IB. On the other hand, IPoIB' throughput increases until 4 cores and then reaches its peak value. The low throughput of IPoIB for only 1 or 2 cores is due to the execution of kernel code in IPoIB network stack that dominates the total I/O time

decreasing the overall throughput. This places RoCE and IB above IPoIB in terms of throughput as they need only a single core to reach maximum throughput.

Next, we analyse the impact of I/O pattern or workload (sequential/random read-/write) for all transport types over 1 KB and 4 KB I/O sizes at full subscription. In Fig. 5a for 4 KB I/O size the maximum IOPS achieved by writes (275 k IOPS) is roughly 33% of the maximum IOPS achieved by reads (832 k IOPS) for IB transport type. This low bandwidth of write workload is not due to network protocol/transport type. We verified this by running the same I/O pattern on local NVMe-SSD and found that the write bandwidth is bounded by the SSD rather than the transport type. It can also been seen from Fig. 5a that the throughput provided by NVMe over TCP is constant across workloads indicating the network protocol is the bottleneck while IB maintains the throughput close to that of local SSDs for all workloads. Surprisingly, from Fig. 5b the maximum throughput of read-write (70:30) and/or sequential write workload (35 k IOPS) at 1 KB I/O size is 4% of the maximum IOPS achieved by random read operations (830 k IOPS). Again, we verify that this peculiar behaviour is not dependent upon transport type but rather is present while performing I/Os on local NVMe SSDs as well due to write amplification at smaller sized I/Os. From Fig. 5 we can observe that random read I/O pattern has the highest throughput on average and we choose this workload for majority of our experiments.



(a) 4KB I/O size

(b) 1KB I/O size

Fig. 5. Throughput/IOPS variation with respect to I/O pattern or workload at full subscription (28 cores/threads). For random read-write (rand_rw) and read-write (rw) workloads the read to write ratio was 70:30

4.3 Latency Evaluation

In latency analysis we study the impact of transport types on latency and the overhead they introduce. We first compare average, p90, p99 and p9999 latency of local NVMe-SSDs and NVMe-oF for all four transport types with 512 B (the minimum block size of a SSD) and 1 MB I/O sizes at full subscription. This shows us the overhead incurred by both small and large I/Os due to the transport type. As expected, from Table 1, the average latency of TCP transport is approximately 20 times that of average latency of

RoCE/IPoIB/IB transport types. Similarly, the p99 and p9999 latency of TCP transport is roughly 30 times that of p99 latency of RoCE/IPoIB/IB transport. In fact, in terms of bandwidth IB is the closest to local NVMe-SSDs for small (Table 2) and large sized I/Os whereas in terms of latency IPoIB adds the least amount of overhead at peak utilization for large I/Os. This could be related to the configuration of various transports in SPDK library. For small I/O requests, IPoIB incurs a higher overhead with its tail latency being 1.5× of IB.

Table 1. Latency at peak bandwidth and full subscription for 1MB random read I/O

	Bandwidth (MiB/s)	Avg Latency (s)	p90 (s)	p99 (s)	p9999 (s)
Local	3626.10	0.96	1.1	1.33	1.40
TCP	466.60	19.44	28.72	29.23	29.27
RoCE	3512.67	0.99	1.16	1.71	1.75
IPoIB	3366.23	1.05	1.12	1.35	1.42
IB	3599.87	0.97	1.15	1.60	1.64

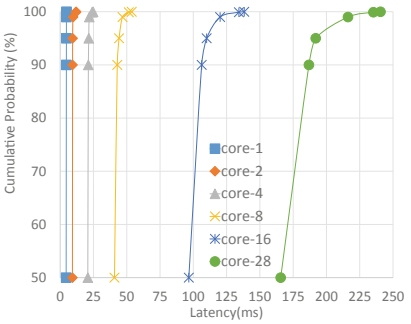
Table 2. Latency at full subscription for 512 bytes random read I/O. RoCE was not reported to due to testbed limitations

	Bandwidth (MiB/s)	Avg Latency (ms)	p90 (ms)	p99 (ms)	p9999 (ms)
Local	403.67	4.30	4.61	4.71	5.78
TCP	111.85	15.65	25.26	28.94	34.94
IPoIB	401.07	4.36	5.03	6.10	9.02
IB	402.97	4.34	4.63	4.77	6.07

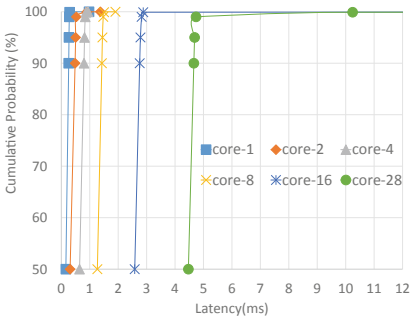
Similar to bandwidth, we next examine the variation in latency with respect to the number of NVMe-oF client and target cores performing I/Os for all the transport types. From the cumulative distribution function (CDF) plot of latencies in Fig. 6 and 7 one can see the latency increases with increase in number of cores irrespective of the transport type and IB has the latency closet to local SSDs for all the cores. RoCE and IPoIB have quite similar latency distribution except p9999 latency of RoCE being unusally high (1146 ms). From Fig. 6c, d, Table 1, and Table 2 we can observe that the tail latency of IPoIB and IB are dependent upon the I/O size. For small and medium sized I/Os IB provides lower tail latency and for larger I/Os IPoIB produces slightly better tail latency.

4.4 CPU Utilization Study

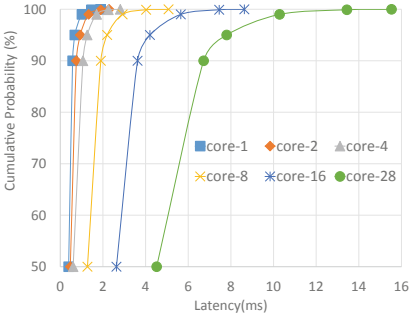
In this experiment, we measure the NVMe-oF client CPU utilization for all four transport types performing 1 MB random read I/O at full subscription using Linux sysstat tools [23]. This helps us measure the efficiency of the various transport types initiating I/O operations at the client. One can see from Fig. 8 that interrupt based TCP and



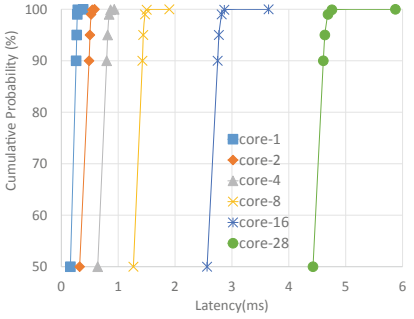
(a) TCP



(b) RoCE. The p9999 latency (1146 ms) for 28 cores is omitted from the graph.



(c) IPoIB



(d) IB

Fig. 6. Latency CDF for various cores for each transport type at 4 KB random read I/O

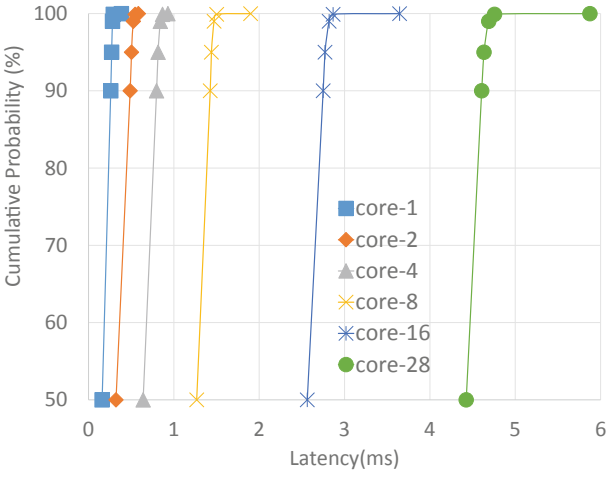


Fig. 7. Latency CDF for various cores for local SSD at 4 KB random read I/O

IPoIB have lower user mode CPU utilization when compared to polling based RoCE and IB. Similar CPU utilization is observed for other workloads too. Table 1 shows us the bandwidth and latency of IPoIB and IB transport for this configuration. Though IB has higher bandwidth when compared to IPoIB, the latencies (p99 and p9999) of IB is slightly higher (1.15×) than IPoIB. This indicates that IPoIB is more efficient at large I/Os despite being an interrupt based network protocol, which may imply that running IB in event based mode could be considered for large I/O sizes rather than polling mode.

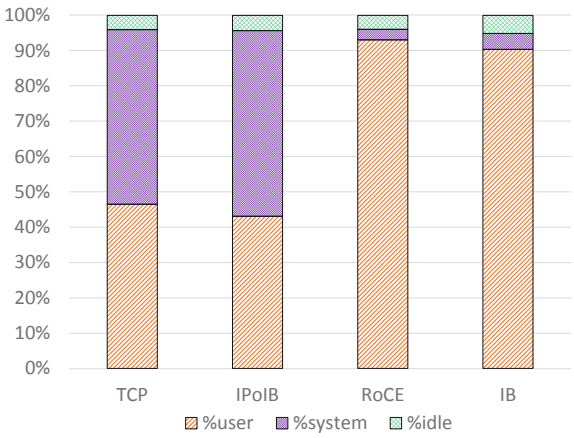


Fig. 8. NVMe-oF client CPU utilization at full subscription for 1 MB random read I/O

4.5 SSD Concurrency Study

To find the parallelism each transport type could achieve, we first measure the throughput or IOPS of local and remote NVMe SSDs for varying queue depth at 4 KB workloads when only 1 CPU core performs I/O operations. The concurrency/parallelism at any queue depth for a specific workload type is calculated by dividing the throughput of that transport type by the throughput of local NVMe SSD at queue depth of 1. For example, the concurrency of an IB transport type for sequential read operations is computed by dividing the each of the IOPS attained by IB transport type for queue depths 1 to 128 by the throughput of a local NVMe SSD for a similar workload at queue depth of 1. Similarly, the concurrency for all network transports and workloads (sequential/random read/write) are calculated. In Fig. 9 and 10 we can see that concurrency for read workloads increases with queue depth whereas write concurrency does not change after queue depth of 8. In terms of network transport, IB is able to reach higher parallelism for both sequential/random read workloads than rest of the transports, followed by RoCE. IPoIB is only able to achieve maximum concurrency of 2.3 for sequential read I/O at a queue depth of 128 which is significantly lower from the maximum concurrencies of IB and RoCE, 6.3 and 4.6 respectively.

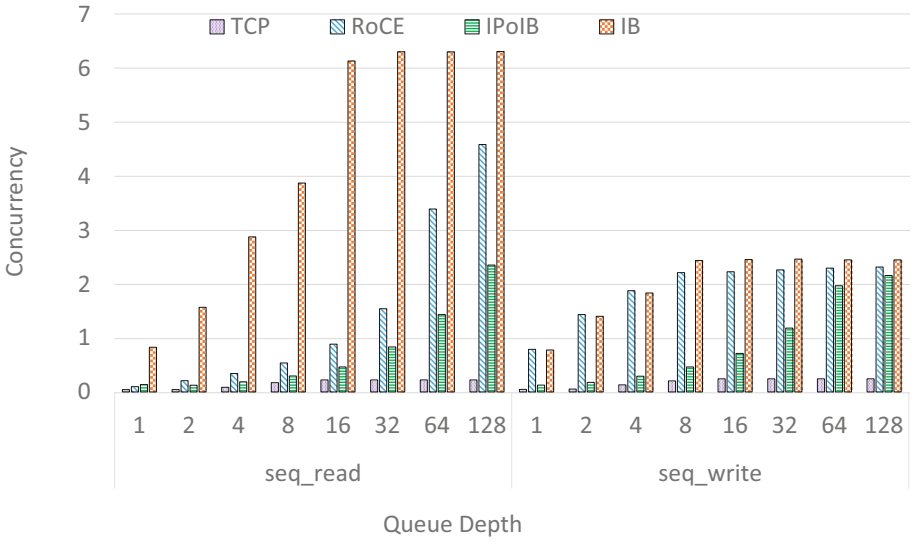


Fig. 9. Concurrency achieved by network transports for varying queue depth for 4 KB sequential workload

5 Discussion

Tradeoffs. Our performance characterization and analysis of user-space NVMe-oF protocol specification is done across four dimensions and various metrics (throughput, latency, CPU utilization, and concurrency). Table 3 provides a summary of the tradeoffs between different transports for evaluated metrics. In terms of throughput/bandwidth and average/tail latencies IB network transport delivers very efficient performance among all the protocols. IB with RDMA was also able to obtain 2.7× higher concurrency than IPoIB transport. IPoIB had better CPU utilization and tail latencies at maximum throughput. There might be two reasons for this. One is that the performance of the tested SSD is limited, which prevents our experiments from achieving higher performance with native RDMA. The achievable performance with IPoIB on IB EDR is sufficient to saturate the SSD's performance. The other reason is that the software implementation in SPDK may not exploit the best configurations with RDMA protocol, which leaves rooms for further performance improvements.

RoCE was similar to IB and IPoIB in terms of bandwidth, concurrency, and average latency but fared slightly worse in terms of tail latency and client CPU utilization. A general trend observed across all network transports was that latency increased as we increased the number of cores performing I/Os. Though IPoIB and IB (RDMA) are faster and have higher performance than RoCE and TCP (ethernet), the current-generation networking infrastructure in datacenters is typically built on top of traditional ethernet. The costs of laying down a new RDMA-capable network would be expensive. To reduce the deployment cost, RoCE could be used on top of the ethernet

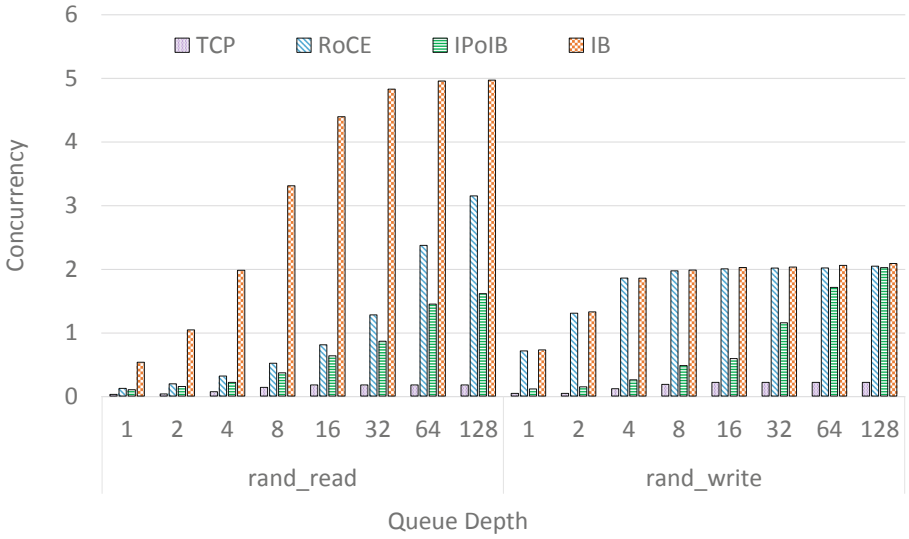


Fig. 10. Concurrency achieved by network transports for varying queue depth for 4 KB random workload

infrastructure by adding specialized NICs to the nodes. Hence, NVMe-over-RoCE may be a good tradeoff between cost and efficiency.

Other Metrics. Despite having a low bandwidth ethernet card (1 Gbps) for NVMe over TCP, the throughput of TCP transport is quite low due to execution of kernel space code during communication. On the contrary, kernel-space based IPoIB (100 Gbps NIC) shows good performance, which implies that kernel overhead is minimal at maximum throughput and when the network bandwidth is high. Although we did not evaluate the kernel NVMe-oF driver, our results indicate that kernel overhead may not have significant impact on throughput, if not latency. We leave a complete evaluation for future work. We also expect the type of SSD to have a profound impact on the metrics across all dimensions due to different driver/controller software and storage technology (NAND flash-based and next generation 3D XPoint [6]). The SSD we evaluated suffers from write amplification effects. Therefore, all networks (except TCP) were able to saturate the SSD write bandwidth, but only IB was able to saturate the read bandwidth. Newer Optane drives do not suffer from garbage collection and write amplification overheads and will thus be harder to saturate for writes. We expect the type of network and its bandwidth to be even more important for such SSDs.

Table 3. Summary of tradeoffs between transports for each evaluated metric. As TCP bandwidth is lower than SSD bandwidth on our cluster, we do not consider it directly. IPoIB is representative of TCP performance because it supports TCP over a faster IB network.

Metric	RoCE	IPoIB	IB
Bandwidth (Fig. 3)	✓	✓	✓
Throughput (Fig. 4)	✓		✓
Average Latency (Table 1 & 3)	✓	✓	✓
Tail Latency (Fig. 6 & Table 3)			✓
Client CPU Utilization (Fig. 8)		✓	
Concurrency (Fig. 9 & 10)			✓
Cost-Effectiveness	✓		

6 Related Work

Previous works [2,3] do not study user-space SPDK's NVMe-oF protocol implementation rather they study NVMe-oF driver present in Linux kernel and compare it with similar protocols like iSCSI [10,11]. Yang et al. [25] showed the inefficiencies present in the Linux kernel NVMe-oF drivers when compared to userspace NVMe-oF drivers. Given that NVMe over TCP is relatively new only two studies [22,26] exist as far as we know which evaluate NVMe/TCP in user-space but no comparison exists with other network tranpsorts. NVMe over TCP is important for the datacenters which currently do not support RDMA-capable networks yet.

Some work which does evaluate disaggregated flash storage in [1–3,9,12,14,15,17,21,24] only use RoCE/iWARP rather than InfiniBand interconnect. Klimovic et al. in [14,15] implement a software system for remote flash access only over TCP/IP and in [13] describe an architecture for disaggregated flash system that optimizes remote flash access by tuning system settings. [9] evaluates NVMe-oF on ARM SoC whereas our characterization is done on x86 architecture. [24] studies NVMe and NVMe-oF for docker storage drivers and containerized applications. Other than [4], which implement HDFS over Fabric on top of NVMe-oF, none of the previous work extensively evaluate an important networking protocol IPoIB and IB or study various workload types like sequential/random reads/writes or NVMe-oF client CPU utilization.

Our work is first-of-a-kind to comprehensively study and characterize SPDK's user-space NVMe-oF protocol for 4 different networking/transport protocols, namely TCP, IPoIB, RoCE, and IB, across numerous dimensions, including number of cores, I/O size, workloads, concurrency, and client-side CPU utilization.

7 Conclusion and Future Work

In this paper, we present an in-depth analysis of the tradeoffs and performance characteristics of network protocols on storage disaggregation. Our characterization of NVMe-over-Fabrics over Ethernet and RDMA was performed across four dimensions

- networking protocols (TCP, RoCE, IPoIB, and IB), I/O size, number of cores, and I/O pattern. We found that native RDMA with IB was able to deliver the best performance among all tested networking protocols in most experiments. IB with RDMA can typically deliver lower latency and higher throughput whereas IB with IPoIB offers lower CPU utilization. Native RDMA with IB or RoCE can achieve higher SSD concurrency. We believe that our analysis helps gain insight into the deployment implications of NVMeoF in datacenters. In the future, we plan to study additional protocols, such as iWARP and fibre channel, and types of NVMe-SSDs. We also plan to evaluate various transport types on real world applications like key-value stores and distributed filesystems.

References

1. Chelsio Communications. NVM Express over Fabrics (2014). http://www.chelsio.com/wp-content/uploads/resources/NVM_Express_Over_Fabrics.pdf
2. Guz, Z., (Huan) Li, H., Shayesteh, A., Balakrishnan, H.: NVMe-over-Fabrics performance characterization and the path to low-overhead flash disaggregation. In: Proceedings of the 10th ACM International Systems and Storage Conference, SYSTOR 2017 (2017)
3. Guz, Z., (Huan) Li, H., Shayesteh, A., Balakrishnan, V.: Performance characterization of NVMe-over-fabrics storage disaggregation. ACM Trans. Storage **14**(4), 1–18 (2018)
4. Han, D., Nam, B.: Improving access to HDFS using NVMeoF. In: 2019 IEEE International Conference on Cluster Computing (CLUSTER), pp. 1–2 (2019)
5. IBM Research - Zurich. Crail (2017). http://www.crail.io/
6. Intel. Intel Optane Memory. https://www.intel.com/OptaneMemory
7. Intel. SPDK NVMe over Fabrics Target (2017). https://spdk.io/doc/nvmf.html
8. Intel. SPDK NVMe over Fabrics Target Programming Guide (2017). https://spdk.io/doc/nvmf.html
9. Jia, Y., Anger, E., Chen, F.: When NVMe over fabrics meets arm: performance and implications. In: 2019 35th Symposium on Mass Storage Systems and Technologies (MSST), pp/134–140 (2019)
10. Joglekar, A., Kounavis, M.E., Berry, F.L.: A scalable and high performance software ISCSI implementation. In: Proceedings of the 4th Conference on USENIX Conference on File and Storage Technologies, FAST 2005, vol. 4, p. 20, USA. USENIX Association (2005)
11. Khosravi, H.M., Joglekar, A., Iyer, R.: Performance characterization of iSCSI processing in a server platform. In: 2005 24th IEEE International Performance, Computing, and Communications Conference, PCCC 2005, pp. 99–107 (2005)
12. Kim, J., Fair, D.: How ethernet RDMA protocols iWARP and RoCE support NVMe over fabrics (Ethernet Storage Forum) (2016). https://www.snia.org/sites/default/files/ESF/How_Ethernet_RDMA_Protocols_Support_NVMe_over_Fabrics_Final.pdf
13. Klimovic, A., Kozyrakis, C., Thereska, E., John, B., Kumar, S.: Flash storage disaggregation. In Proceedings of the Eleventh European Conference on Computer Systems, EuroSys 2016 (2016)
14. Klimovic, A., Litz, H., Kozyrakis, C.: ReFlex: remote flash ≈ local flash. In: Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2017, pp. 345–359 (2017)
15. Klimovic, A., Wang, Y., Stuedi, P., Trivedi, A., Pfefferle, J., Kozyrakis, C.: Pocket: elastic ephemeral storage for serverless analytics. In: Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation, OSDI 2018, pp. 427–444, USA. USENIX Association (2018)

16. Mickens, J., et al.: Blizzard: fast, cloud-scale block storage for cloud-oblivious applications. In: 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2014), pp. 257–273 (2014)
17. Dave M., Metz, J.: Under the Hood with NVMe over fabrics. In: Proceedings of the Ethernet Storage Forum (2015)
18. NVM Express (2011). https://nvmexpress.org/
19. NVMe-over-Fabrics Specification (2019). https://nvmexpress.org/developers/nvme-of-specification/
20. SPDK NVMe perf Benchmark. https://github.com/spdk/spdk/tree/master/examples/nvme/perf
21. SPDK 20.04 NVMe-oF RDMA Performance Report. https://ci.spdk.io/download/performance-reports/SPDK_rdma_perf_report_2004.pdf
22. SPDK 20.04 NVMe-oF TCP Performance Report. https://ci.spdk.io/download/performance-reports/SPDK_tcp_perf_report_2004.pdf
23. Linux sysstat. https://github.com/sysstat/sysstat
24. Xu, Q., et al.: Performance analysis of containerized applications on local and remote storage. In: International Conference on Massive Storage Systems and Technology (2017)
25. Yang, Z., et al.: SPDK: a development kit to build high performance storage applications. In: 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), pp. 154–161 (2017)
26. Yang, Z., Wan, Q., Cao, G., Latecki, K.: uNVMe-TCP: a user space approach to optimizing NVMe over fabrics TCP transport. In: Hsu, C.-H., Kallel, S., Lan, K.-C., Zheng, Z. (eds.) IOV 2019. LNCS, vol. 11894, pp. 125–142. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-38651-1_13