An Early Case Study with Multi-Tenancy Support in SPDK's NVMe-over-Fabric Designs

Darren Ng, Charles Parkinson, Andrew Lin, Arjun Kashyap, and Xiaoyi Lu
{dng350,cparkinson2,alin85,akashyap5,xiaoyi.lu}@ucmerced.edu
University of California, Merced

ABSTRACT

Resource disaggregation is prevalent in datacenters since it provides high resource utilization when compared to servers dedicated to either compute, memory, or storage. NVMe-over-Fabrics (NVMe-oF) is the standardized protocol used for accessing disaggregated storage over the network. Currently, the NVMe-oF specification lacks any semantics to prioritize I/O requests based on different application needs. Since applications have varying goals — latency-sensitive or throughput-critical I/O — we need to design efficient schemes in order to allow applications to specify the type of performance they wish to achieve. Furthermore, with additional tenants, we need to provide the respective specified performance optimizations that each application requests, regardless of congestion. This is a challenging problem, as the current NVMe specification lacks semantics. to support multi-tenancy. Our early study research poster brings awareness to the ways in which the we can bring multi-tenancy support to the NVMe-oF specification.

1 INTRODUCTION

The term performance has different meanings for applications since some applications optimize for latency while others optimize for throughput. Hence, the underlying storage system/runtime needs to be made aware of whether the application I/O is latency-sensitive or throughput-critical. Since the underlying storage runtime is not multi-tenant aware, it treats all the I/O requests with the same priority. The problem is exacerbated by the fact that the NVMe-oF protocol, a popular remote storage protocol, does not support passing application hints to the storage system about their performance goals.

Some work has been done to add priority and Quality-of-Service (QoS) support to local SSD accesses [1–4] while others have implemented the same for disaggregated storage [5, 6]. ReFlex [5] uses a highly customized storage disaggregation stack to enable priority support while our design adds priority design to the popular NVMe-oF protocol used for storage disaggregation. blk-switch [6] simulates the priority in Intel SPDK [7] by varying the niceness value of the different application processes and does not tag individual application I/O requests. Thus, there is still a gap to truly support priority schemes in the userspace NVMe-oF runtime with multi-tenancy support.

We take a preliminary step in studying necessities of multi-tenancy support in SPDK's NVMe-oF protocol and illustrate the early benefits of supporting multi-tenancy. Unlike previous work, we move our implementation into the userspace. In the future, we hope to create an extensive evaluation of multi-tenancy support for NVMe-oF.

2 MOTIVATION

In this section we would like to answer the question - *What are the benefits of multi-tenancy support?* In datacenters and HPC, it is common for a multi-tenancy environment to be highly desirable and allows devices to serve more applications and users at once. Considering that different applications may not optimize for the same performance goals, SPDK's NVMe-oF will not be able to deliver desired performance respective to each application. By supporting multi-tenancy within SPDK, each application will be able to achieve a specified performance optimization strategy regardless of other concurrent applications.

For this early study, we choose to augment Intel SPDK with priority schemes and multi-tenant support since it is a widely used userspace storage library in HPC systems. For instance, Intel Distributed Asynchronous Object Storage (DAOS) which is a widely popular exascale storage stack [8–10], internally uses SPDK [11]. Another question that presents itself - *What factors of existing SPDK NVMe-oF protocol violates multi-tenancy support?* In our early observations, we have framed two major factors that prevent it multi-tenancy: computation order and semantic data passing. The main requirement of multi-tenancy support is to provide the respective performance optimization that is specified by an application and we review them in this section.

Computation order. The NVMe-oF target processes each request in the order received. Thus an application with low latency objectives may be required to wait behind a very large number of requests from a high throughput application. As shown in Figure 1, we can observe multiple requests from two different applications of different performance objectives and how a latency-sensitive application may have long latencies for its requests. Thus the multi-tenancy requirement is not upheld. Latency-sensitive requests can also disrupt the throughput-critical requests as the target must now switch between processing two different types of requests within one batch of throughput-critical requests. This can negatively impact throughput-critical requests for high-throughput applications which similarly will not uphold multi-tenancy requirements.

Another issue is that SPDK sends too much completion notification packets to optimize for high-throughput applications. Each I/O request requires the NVMe-oF target to deliver a request completion

SC '23, November 12–17, 2023, Denver, CO

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

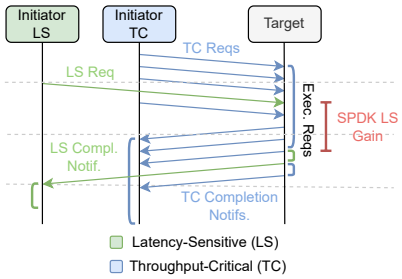

Figure 1: Baseline SPDK timeline of request completions of two different initiators.

notification. For high-throughput applications, these large number of request completion notifications consume CPU processing and can reduce throughput. The reduction in throughput becomes more visible when multiple applications begin concurrent requests.

Semantic Data Passing. The SPDK library does not support any priority schemes that would allow the storage system to discern I/O requests. Thus, the traditional userspace NVMe-oF storage runtimes can neither interpret nor handle the differential requests to maximize the specific performance goal the application may desire. This can only be achieved by providing the semantic data within each request that denotes the optimization strategy of the application. In order to bridge this gap we need to enable priority semantic data passing in SPDK-based NVMe-oF runtime to support multi-tenancy.

3 EVALUATION

Experimental Setup: Our test environment uses Chameleon Cloud [12] storage_nvme nodes to provide 10/25Gbps Ethernet experiments. We use an AMD EPYC 7352 2.3GHz processor with 24 cores and 256 GB RAM.

Performance Analysis: In our early experiments we show results for throughput on two different interconnects and the throughput and latency concurrency. We compare the default SPDK numbers running the perf benchmark with a queue depth of 1 and 128 as the baseline against latency sensitive and throughput critical initiators from our design, respectively.

It is observed that there are two factors that are needed to achieve performance improvement: 1) proper window size selection and 2) network speed.

Window Size Selection: In our design, we define window size as the number of requests coalesced into one completion notification network packet. Figure 2 shows throughput and latency of our design over increasing window sizes. In this experiment we observe performance results with multi-tenancy using one throughput-critical initiator and one latency-sensitive initiator. Our design achieves a peak throughput gain of 35.9% at a window size of 32, and over all window sizes, achieves an average improvement in throughput by 20.6% over SPDK. This is also in tandem with maintaining a low latency with the difference in latency at -4.5%. Overall, we can see that a large window size (64) may not always be optimal and a small window size (<32) will not maximize throughput.

Network Performance Impact: In this subsection, we observe results with 10 and 25 Gbps Ethernet speeds and how they influence

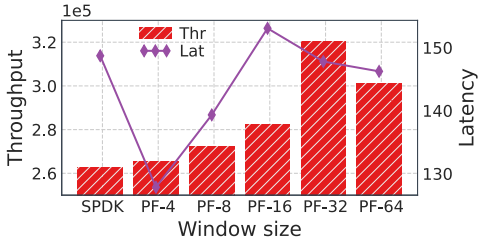

Figure 2: Throughput of one NVMe-oF initiator sending throughput-critical requests concurrently with one sending latency-sensitive requests. Varying window sizes are shown (PF-4/8/16/32/64).

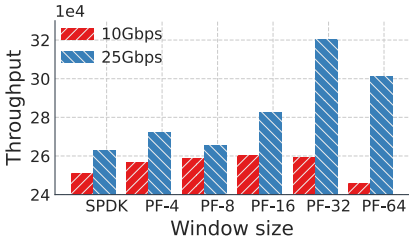

Figure 3: Throughput for 10 and 25 Gbps while sending a throughput-critical write workload.

performance. Figure 3 shows performance gain from the baseline SPDK to our design with varying window sizes. For factor 2), we can see that both SPDK and our design are hindered in performance when using 10Gbps speeds. In this case, the network becomes the bottleneck as it is already saturated. If we observe the 25Gbps Ethernet, we are able to increase throughput as window size increases. For a window size of 64 at 10Gbps, the completion notification packets begin to observe more time before being received by the initiator, exacerbated by network bottleneck, and may negatively impact performance.

4 CONCLUSION AND FUTURE WORK

Our early study design provides the ability for workloads to specify their I/O optimization strategies — latency-sensitive or throughput-critical — and makes the NVMe-oF runtime multi-tenant aware. Our design shows the benefit of coalescing request completions for userspace NVMe-oF runtime. In the future, we will co-design more HPC and datacenter applications with the proposed design schemes to demonstrate the benefits.

5 ACKNOWLEDGEMENTS

This work was supported in part by an NSF research grant OAC #2321123 and a DOE research grant DE-SC0024207. Results presented in this paper were obtained using the Chameleon testbed supported by the National Science Foundation

REFERENCES

[1] S. Kim, H. Kim, J. Lee, and J. Jeong, "Enlightening the I/O Path: A Holistic Approach for Application Performance," in *Proceedings of the 15th Usenix Conference on File and Storage Technologies*, FAST'17, (USA), p. 345–358, USENIX Association, 2017.

[2] J. Zhang, M. Kwon, D. Gouk, S. Koh, C. Lee, M. Alian, M. Chun, M. T. Kandemir, N. S. Kim, J. Kim, and M. Jung, "Flashshare: Punching through Server Storage

Stack from Kernel to Firmware for Ultra-Low Latency SSDs," in *Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation*, OSDI'18, (USA), p. 477–492, USENIX Association, 2018.

[3] H.-J. Kim, Y.-S. Lee, and J.-S. Kim, "NVMeDirect: A User-space I/O Framework for Application-specific Optimization on NVMe SSDs," in *8th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 16)*, (Denver, CO), USENIX Association, June 2016.

[4] A. Tai, I. Smolyar, M. Wei, and D. Tsafrir, "Optimizing Storage Performance with Calibrated Interrupts," *ACM Transactions on Storage (TOS)*, vol. 18, no. 1, pp. 1–32, 2022.

[5] A. Klimovic, H. Litz, and C. Kozyrakis, "ReFlex: Remote Flash Local Flash," in *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '17, (New York, NY, USA), p. 345–359, Association for Computing Machinery, 2017.

[6] J. Hwang, M. Vuppalapati, S. Peter, and R. Agarwal, "Rearchitecting linux storage stack for μs latency and high throughput," in *15th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 21)*, pp. 113–128, 2021.

[7] Z. Yang, J. R. Harris, B. Walker, D. Verkamp, C. Liu, C. Chang, G. Cao, J. Stern, V. Verma, and L. E. Paul, "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, IEEE, 2017.

[8] J. Liu, Q. Koziol, G. F. Butler, N. Fortner, M. Chaarawi, H. Tang, S. Byna, G. K. Lockwood, R. Cheema, K. A. Kallback-Rose, D. Hazen, and M. Prabhat, "Evaluation of HPC Application I/O on Object Storage Systems," in *2018 IEEE/ACM 3rd International Workshop on Parallel Data Storage Data Intensive Scalable Computing Systems (PDSW-DISCS)*, pp. 24–34, 2018.

[9] J. Soumagne, J. Henderson, M. Chaarawi, N. Fortner, S. Breitenfeld, S. Lu, D. Robinson, E. Pourmal, and J. Lombardi, "Accelerating HDF5 I/O for Exascale Using DAOS," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 4, pp. 903–914, 2022.

[10] M. S. Breitenfeld, N. Fortner, J. Henderson, J. Soumagne, M. Chaarawi, J. Lombardi, and Q. Koziol, "Daos for extreme-scale systems in scientific applications," 2017.

[11] Intel, "DAOS: Revolutionizing High-Performance Storage." https://www.intel.com/content/www/us/en/high-performance-computing/daos-high-performance-storage-brief.html, 2023.

[12] Chameleon Cloud, "Chameleon Cloud." https://www.chameleoncloud.org/, 2023.