International Journal of Advanced Smart Convergence Vol.9 No.3 253-259 (2020)
http://dx.doi.org/10.7236/IJASC.2020.9.3.253

IJASC 20-3-33

Implementation of Light-weight I/O Stack for NVMe-over-Fabrics

Sungyong Ahn

Assistant Professor, School of Computer Science and Engineering, Pusan National University, Korea
sungyong.ahn@pusan.ac.kr

Abstract

Most of today's large-scale cloud systems and enterprise data centers are distributing resources to improve scalability and resource utilization. NVMe-over-Fabric protocol allows submitting NVMe commands to a remote NVMe SSD through RDMA (Remote Direct Memory Access) network. It is attracting attention recently because it is possible to construct a disaggregation storage system with low latency through the protocol. However, the current I/O stack of NVMe-over-Fabric has an inefficient structure for maintaining compatibility with the traditional I/O stack. Therefore, in this paper, we propose a new mechanism to reduce I/O latency and CPU overhead by modifying I/O path of NVMe-over-Fabric to pass through legacy block layer. According to the performance evaluation results, the proposed mechanism is able to reduce the I/O latency and CPU overhead by up to 22% and 24% compared to the existing NVMe-over-Fabrics protocol, respectively.

Keywords: *Large-scale Cloud System, NVMe-over-Fabric, Storage Area Network, RDMA, NVMe SSDs*

1. Introduction

Recently, the resource disaggregation is widely used in a large-scale cloud system and enterprise data center for improving the resource scalability and utilization. Especially, storage disaggregation is very suitable for BigData workload in which data size increases rapidly by separating the storage node from the computation node [1]. Traditionally, disaggregated storage system based on HDDs (Hard Disk Drives) is established on iSCSI protocol which provides block-level access to remote storage device by carrying SCSI command over a network. However, iSCSI protocol is not suitable for high performance and low latency storage device such as NVMe SSD because its overhead is large enough to severely harm the I/O performance of NVMe SSDs while negligible for HDDs.

NVM Express (NVMe) [2] is the state-of-the-art storage interface for accessing non-volatile storage media such as SSDs. The legacy storage interface, such as SATA and SAS cannot take advantage of the internal parallelism of SSDs due to insufficient maximum bandwidth and single I/O queues. However, NVMe SSDs support multiple I/O queues so that the internal parallelism of SSDs can be utilized to the maximum. On the other hand, PCIe-based NVMe SSDs have much more limited number of devices than SATA/SAS that can be

Manuscript Received: August. 19, 2020 / Revised: August. 25, 2020 / Accepted: August. 29, 2020
Corresponding Author: sungyong.ahn@pusan.ac.kr
Tel: +82-51-510-2422, Fax: +82-51-517-2431
Assistant Professor, School of Computer Science and Engineering, Pusan National University, Korea

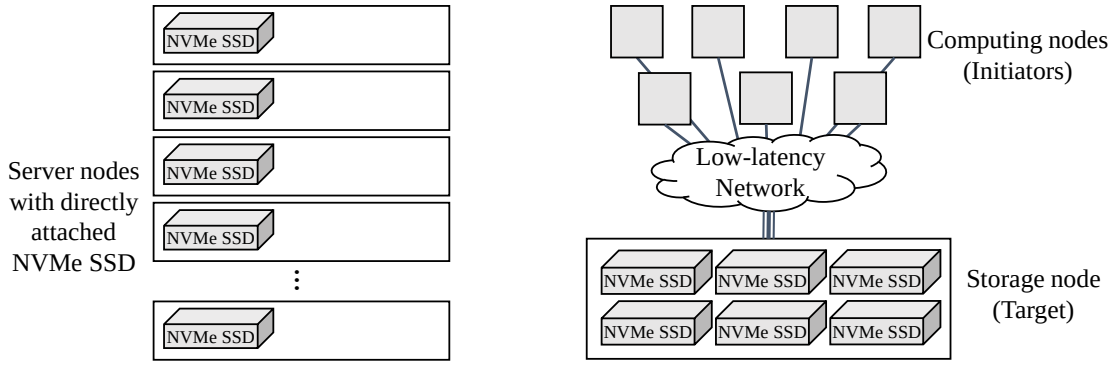


Figure 1. NVMe and NVMe-over-Fabrics

directly attached to a node. Therefore, NVMe-over-Fabrics [3][5] is recently proposed as a remote storage protocol for NVMe SSD. Because it transfers NVMe command directly to a remote NVMe SSD through RDMA (Remote Direct Memory Access) [6]. NVMe-over-Fabrics can reduce protocol overhead dramatically by removing legacy SCSI layer designed for hard disk drive and decreasing network delay and CPU overhead by using RDMA to transfer NVMe commands. According to [3], The goal of NVMe-over-Fabrics is to provide block level access to a remote NVMe SSD with no more than 10 microseconds of additional latency over a native NVMe SSD directly attached to a server.

However, current I/O stack of NVMe-over-Fabrics handles received NVMe commands via Linux block I/O layer to retain the compatibility with a legacy Linux block I/O stack. It increases the latency of remote NVMe SSDs and CPU overhead of storage node so that it has a bad effect on a scalability of storage node. Therefore, in this paper we propose a light-weight I/O stack for NVMe-over-Fabrics in which received NVMe commands are directly forwarded to NVMe SSD while bypassing the Linux block layer. In the proposed scheme, out of band NVMe command IDs are used to separate received NVMe commands from local NVMe commands while keeping the compatibility with legacy NVMe device driver. As a result, it reduces latency of remote NVMe SSDs and CPU overhead of storage node by 22% and 24%, respectively.

The remainder of this paper is organized as follows. First, Section 2 describes the background and related work. Then, we show the evaluation results for the overhead of NVMe-over-Fabrics protocol in Section 3. Section 4 describes the proposed mechanism for light-weight NVMe-over-Fabrics protocol. Experimental results are presented in Section 5. The conclusion is given in Section 6.

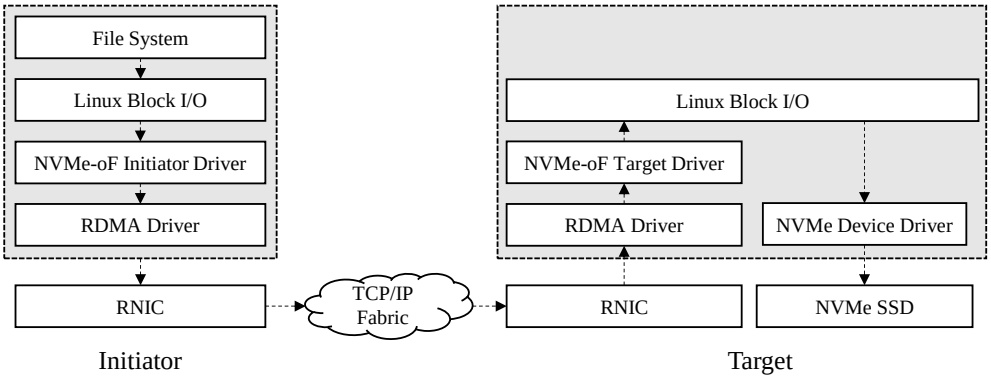


Figure 2. Overview of I/O stack for NVMe-over-Fabrics

2. Related Work

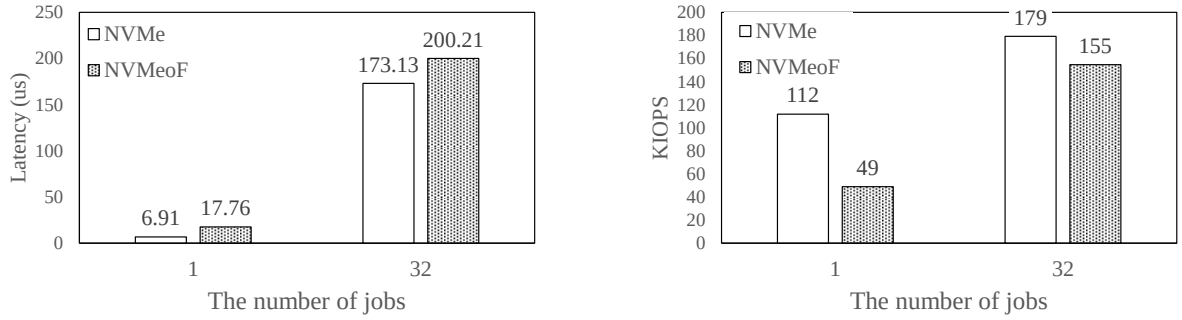
NVM Express (NVMe) [2] is high performance and scalable storage interface for PCIe based SSD. NAND flash memory based SSDs (Solid-state Drives) are recently very popular as a second storage device because of its various advantages such as low latency, high bandwidth, and low power consumption in comparison to HDDs (Hard-disk Drives). However, legacy storage interface, SATA/SAS have insufficient maximum bandwidth due to supporting single I/O queue so that it could not take full advantage of internal parallelism of SSDs. In contrast, NVMe has 65535 submission queues and completion queues and supports up to 64K command queueing for each, so it delivers high performance by fully utilizing the internal parallelism of SSD. Thus, NVMe SSD is recently more popular than the prior SSD based on SATA/SAS. However, since NVMe SSD requires a PCIe slot to be directly attached to a node, it has a great limitation on scalability. Therefore, using NVMe SSDs as a direct-attached storage (DAS) is not suitable for large-scale cloud systems or BigData systems where scalability of storage capacity is very important.

NVMe-over-Fabrics is new storage disaggregation protocol which transfers NVMe commands to NVMe SSD attached on the remote server through a high-speed network. Figure 1 shows the difference between direct attached NVMe SSD and disaggregation NVMe SSD using NVMe-over-Fabrics protocol. As you can see in Figure 1(b), the protocol is composed of a Target node on which NVMe SSD is directly attached and an Initiator node that accessing NVMe SSD remotely through NVMe-over-Fabrics protocol. Note that the RNIC (RDMA enabled Network Interface Card) should be installed on each node. An Initiator node can use the NVMe SSD on target node with NVMe-over-Fabrics protocols as if it is installed locally and, through this, the several benefits we can get are as follows: (1) the existing file systems and application can be used without any modification, (2) the number of storage devices is not restricted by the number of PCIe slots and can be expanded unlimitedly through a network. And (3) the storage management and monitoring can be performed more effectively by constructing a storage node separately form computing nodes.

There have been some studies for I/O scheduling in NVMe-over-Fabrics. In [7], Z. Guz et al. characterize the overhead of NVMe-over-Fabrics. According to the result of their research, NVMe-over-Fabrics causes only negligible performance degradation while iSCSI decrease I/O throughput by 20%. Also, J. Choi et al. have evaluated the performance of NVMe-over-Fabric protocol with shared-disk file systems [8] and PCIe JBOF (Just a Bunch Of Flash) equipped with 8 NVMe SSDs [9]. Moreover, as NVMe-over-Fabric received attention, some studies on applying NVMe-over-Fabric in various applications including big data and deep learning platforms was conducted. Y. Zhu et al. explore the use of NVMe storage disaggregation for support of deep neural networks [10]. D. Han et al. have improved the performance of HDFS, most popular Hadoop filesystem, by using NVMe-over-Fabric. According to evaluation results, the proposed architecture can improve the read throughput of HDFS by up to 2.55 times [11].

Table 1. Hardware and Software Setup

Component	Specification
CPU	Intel® Xeon® processor E5-2600 v4 x 2
Memory	64GB memory
NVMe SSD	Intel DC P4500 2.0TB
RNIC	Mellanox ConnectX-5 EN 100GbE
OS	*Linux 4.8*



(a) I/O latency (b) IOPS

Figure 3. I/O Performance Comparison between NVMe and NVMe-over-Fabrics

3. Motivation

Figure 2 briefly describes the legacy I/O stack between Initiator and Target node for NVMe-over-Fabrics protocols. As can be seen in the figure, a Target node receives NVMe command transmitted from Initiator node through NVMe-over-Fabrics protocols. However, the received NVMe command cannot be delivered to the NVMe device driver directly but is delivered to Linux block layer via NVMe-over-Fabrics Target driver. That is, to handle NVMe command sent by Initiator node, the new block I/O request should be built by NVMe-over-Fabric driver and deliver to NVMe device driver through a multi-queue block layer. In other words, NVMe-over-Fabrics I/O request goes through the block layer of a Target node unnecessarily. This mechanism causes several problems as follows: (1) the additional latency is occurred by a multi-queue block layer in a Target node. And (2) more CPU cycle of a Target node should be consumed to handle I/O request delivered through NVMe-over-Fabrics protocol. Consequently, as more NVMe SSDs are installed, CPU usage can become a bottleneck for storage node.

In this paper, we measured the latency of I/O requests which Initiator node submits to remote NVMe SSD through NVMe-over-Fabrics protocol. For the experiments, we configured remote NVMe SSD using two nodes and conducted 4KB random read workload with FIO benchmark [4]. The detailed the hardware and software setup is described in Table 1. Figure 3 shows the I/O performance evaluation results of NVMe and NVMeoF. According to the evaluation results of Figure 3(a), the latency of NVMe-over-Fabrics increases about 11 to 27us compared to the local NVMe SSD. Also, we can see that the IOPS decreases by 13~56% in the Figure 3(b). Considering the high performance of NVMe SSD, which has only a few microseconds of latency, it means that the performance overhead of NVMe-over-Fabrics is still large.

As mentioned above, a Target node of NVMe-over-Fabrics protocol uses the block layer to process the I/O request received from an Initiator node, which causes additional delay time and CPU cycle consumption. Therefore, in this paper, to reduce the latency and the CPU usage of a Target node, we propose light-weight I/O stack passing the NVMe command from NVMe-over-Fabrics driver to NVMe SSD device driver directly, not through the block layer.

4. Implementation

In this section, we propose the light-weight I/O stack for NVMe-over-Fabric with bypassing the block layer of Linux kernel. Figure 4 briefly shows the main difference between the proposed mechanism and the legacy I/O stack of NVMe-over-Fabrics target node. Note that, in the figure, the dotted line and the solid line

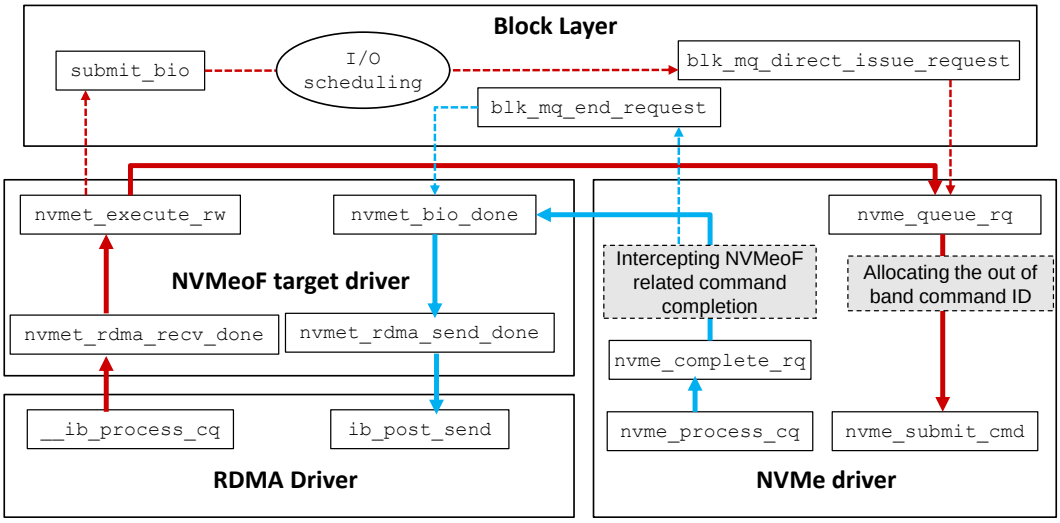


Figure 4. The proposed Light-weight I/O Stack for NVMe-over-Fabrics

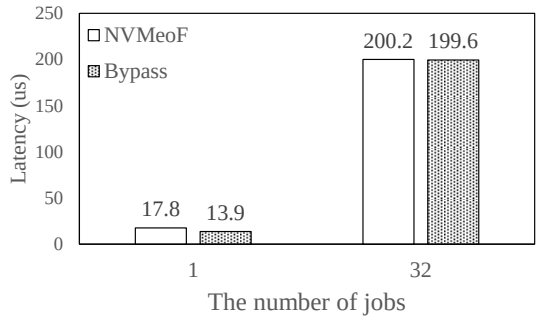
represents the I/O path of the legacy I/O stack and the proposed mechanism, respectively. As you can see in the figure, NVMe-over-Fabrics target driver submits I/O request delivered from the initiator node to the block layer by calling *submit_bio()* function in the legacy I/O stack. In other words, the legacy I/O stack suffers from the block layer overhead such as I/O scheduling and block I/O request handling. However, the proposed block layer bypassing mechanism submits the I/O request directly to NVMe device driver by calling *nvme_queue_rq()*. As a result, the overhead of Linux block layer can be removed form NVMe-over-Fabrics I/O stack in the proposed mechanism.

In addition, the out of band command ID mechanism is also invented to maintain compatibility with the existing block layer. In other to process local NVMe I/O requests simultaneously with NVMe-over-Fabrics I/O request it is necessary to distinguish the NVMe command transmitted from the NVMe-over-Fabrics Initiator during the I/O completion process of the NVMe device driver. To this end, when submitting the Initiator's NVMe command to the NVMe driver, a value outside the normal range is assigned for the NVMe command ID. Therefore, when a NVMe command is completed, you can check the command ID to determine whether the NVMe command is transmitted form Initiator or local application. If the command ID has normal range value the I/O request is handled normally by the block layer. If not, the I/O request is intercepted by NVMe-over-Fabrics driver and the result of I/O request is send to Initiator through the RDMA network. By using this technique, the block layer overhead can be effectively removed while maintaining compatibility with the existing block I/O stack.

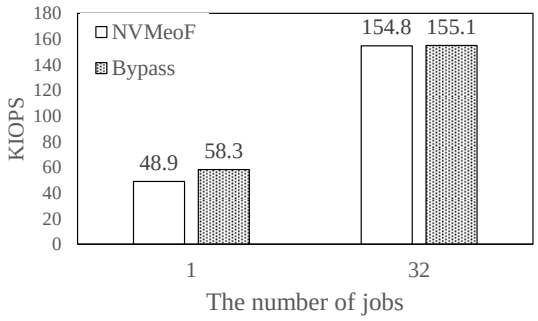
5. Experiment Results

We have implemented the proposed mechanism in Linux kernel 4.8 and evaluated it by using two Intel Xeon servers equipped with dual Xeon-E4 CPU, Mellanox ConnectX-5, and Intel NVMe SSD as described in Table 1.

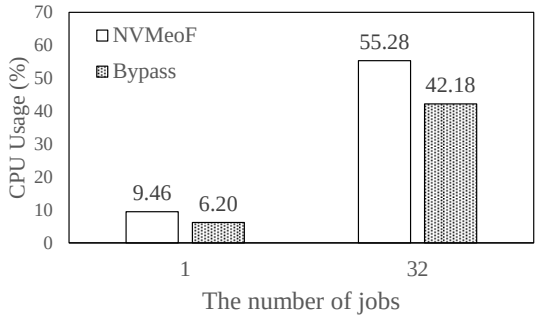
Figure 5 shows the I/O performance and CPU overhead of existing NVMe-over-Fabrics protocol (*NVMeoF*) and the newly proposed protocol (*Bypass*) measured with well-known I/O benchmark tools, FIO [4]. As can be seen from the Figure 5(a), the proposed protocol (*Bypass*) reduces the delay time by up to about 4us in comparison with *NVMeoF*, which is a 22% reduction in the overhead of the existing protocol. When the number of jobs is 32, there is almost no improvement in latency because the queuing delay increases



(a) I/O latency



(b) IOPS



(c) CPU usage of Target node

Figure 5. Performance Evaluation Results of the Proposed Light-weight NVMe-over-Fabrics Protocol

significantly as the I/O depth increases, and the latency reducing effect due to block layer bypassing is diluted. On the other hand, IOPS increases by up to 18% over the existing protocol in Figure 5(b).

Another advantage of the proposed protocol is the reduction of CPU overhead. Figure 5(c) shows the CPU usage consumed by NVMe-over-Fabrics protocol. It can be seen that the proposed mechanism reduces CPU overhead by up to 24% compared to the existing protocol while maintaining I/O performance. This means that more I/O requests of NVMe-over-Fabrics can be processed by a storage node with the limited computing power without I/O performance degradation. As a result, a storage node can provide higher scalability with the proposed mechanism.

6. Conclusion

NVMe-over-Fabrics is a storage networking protocol standard proposed to overcome the limitations of scalability of existing NVMe interface. Using this protocol, NVMe commands can be sent directly to the NVMe SSD mounted on a target node to perform I/O requests. However, in order to maintain compatibility with the block layer, the existing protocol does not directly transfer the received NVMe command to the NVMe driver, but unnecessarily passes through the block layer. This increases the I/O latency of NVMe-over-Fabrics and causes an increasing CPU overhead. Therefore, in this paper, we proposed a protocol that directly transfers NVMe commands to the NVMe device driver without going through the block layer while maintaining compatibility with the existing block layer. As a result of performance evaluation, the proposed mechanism reduces the I/O delay time and CPU overhead by up to 22% and 24% compared to the existing NVMe-over-Fabrics protocol, respectively.

Acknowledgement

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. 2018R1D1A3B07050034) and Pusan National University Research Grant, 2017

References

[1] A. Klimovic, C. Kozyrakis, E. Thereska, B. John, and S. Kumar, "Flash storage disaggregation," in *Proc. 11th European Conference on Computer Systems (EuroSys '16)*, pp. 1–15, April 18-21, 2016.
DOI: https://doi.org/10.1145/2901318.2901337

[2] NVM Express Base Specification Revision 1.4a, *https://nvmexpress.org/wp-content/uploads/NVM-Express-1_4a-2020.03.09-Ratified.pdf.*

[3] NVMe_Over_Fabrics, *https://nvmexpress.org/wp-content/uploads/NVMe_Over_Fabrics.pdf*

[4] Flexible I/O Tester, *https://github.com/axboe/fio.*

[5] NVM Express over Fabrics Specification Revision 1.1, *https://nvmexpress.org/wp-content/uploads/NVMe-over-Fabrics-1.1-2019.10.22-Ratified.pdf.*

[6] R. Mittal, A. Shpiner, A. Panda, E. Zahavi, A. Krishnamurthy, S. Ratnasamy, and S. Shenker, "Revisiting network support for RDMA," in *Proc. the 2018 Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '18)*, pp. 313–326, Aug. 20-25, 2018
DOI: https://doi.org/10.1145/3230543.3230557

[7] Z. Guz, H. Li, A. Shayesteh, and V. Balakrishnan, "NVMe-over-fabrics performance characterization and the path to low-overhead flash disaggregation," in *Proc. 10th ACM International Systems and Storage Conference*, pp. 1-9, May 22-24, 2017.
DOI: https://doi.org/10.1145/3078468.3078483

[8] J. Choi, H. Eom, and H. Yeom, "An Evaluation of Shared-Disk File Systems Using NVMe-over-Fabrics," in *Proc. Korea Software Congress 2017*, pp. 85-87, Dec. 20-22, 2017.

[9] J. Choi, H. Yeom, and H. Han, "An Evaluation of NVMe-over-Fabrics on PCIe JBOF SSDs," *Journal of KIISE*, Vol. 46, No. 6, pp. 499-505, June 2019.
DOI: https://doi.org/10.5626/JOK.2019.46.6.499

[10] Y. Zhu, W. Yu, B. Jiao, K. Mohror, A. Moody and F. Chowdhury, "Efficient User-Level Storage Disaggregation for Deep Learning," in Proc. *2019 IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 1-12, Sept. 23-26, 2019.
DOI: https://doi.org/10.1109/CLUSTER.2019.8891023

[11] D. Han and B. Nam, "Improving Access to HDFS using NVMeoF," in Proc. 2019 IEEE International Conference on Cluster Computing (CLUSTER), pp. 1-2, Sep. 23-26, 2019.
DOI: https://doi.org/10.1109/CLUSTER.2019.8890996