Refining Compaction Offloading I/O Stack for LSM-based Key-Value Stores with SPDK

Honghyeon Yoo*, Hongsu Byun*, Sungyong Park†
Department of Computer Science and Engineering, *Sogang University*, Seoul, Republic of Korea
{yhh, byhs, parksy}@sogang.ac.kr

***Abstract*—We analyze the compaction I/O characteristics and the resulting I/O time amplification problem in log-structured merge-tree based key-value stores. We also propose an I/O stack refinement using SPDK and NVMe-oF for compaction offloading. Using our proposed I/O stack, preliminary experimental results show a 37.7% improvement in compaction throughput.**

***Index Terms*—Key-Value Store, LSM-tree, SPDK**

I. Introduction

LSM-tree based Key-Value Stores. The log-structured merge-tree based key-value store (LSM-KVS) is used not only as a high-performance database due to its high write performance, but also in a wide range of systems such as real-time data storage and processing like streaming, and furthermore as the backbone of distributed databases. On the other hand, LSM-KVS requires a compaction process that merge-sorts the inserted key-value pairs based on logs. Compaction delays can cause read performance degradation and I/O blocking, thus directly impacting the performance of LSM-KVS. Improving compaction in LSM-KVS has traditionally been a well-known topic, with numerous studies conducted ranging from improvements in compaction algorithms and scheduling to offloading across heterogeneous devices.

Network-based Compaction Offloading. In studies aimed at improving compaction, a recent *network-based compaction offloading* approach has emerged, which offloads compaction tasks between different servers connected via a network [1], [2]. The network-based compaction offloading approach has the advantage of accelerating compaction by utilizing the resources of other servers. However, it requires transmitting data over the network from the host server where the LSM-KVS data is stored to the remote server that performs the offloaded compaction. In other words, the I/O stack for network-based compaction offloading includes the additional network layer overhead along with the conventional compaction process.

Goals. NVMe over Fabrics (NVMe-oF) is a technology that extends the NVMe protocol over a network, providing low latency and high bandwidth similar to local devices for remote server NVMe devices. Noting the advantage of reducing the I/O stack when accessing remote server NVMe devices through NVMe-oF, we propose refining the I/O stack in network-based compaction offloading. This can ultimately improve both compaction and LSM-KVS performance.

Contributions. The contributions are as follows:

- We first analyzed the I/O characteristics of compaction to reveal the problem of I/O-induced delays in compaction and identified and the necessity to enhance the I/O stack.
- We designed compaction offloading using Intel SPDK [3] and NVMe-oF to dramatically reduce the I/O stack, and our preliminary experiments show a 37.2% reduction in compaction time and a 37.7% increase in throughput.

*Authors contributed equally to this work.
†S. Park is the corresponding author.

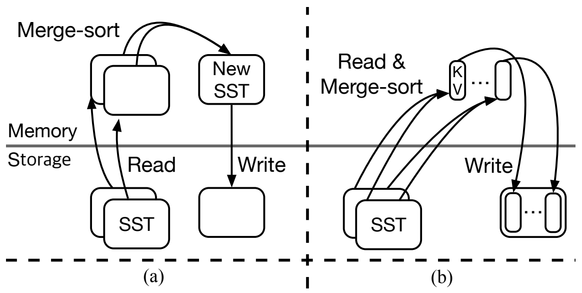

Fig. 1. Compaction's (a) logical execution process (b) actual implementation and execution process.

II. Background & Motivation

Figure 1(a) shows the anticipated logical process of compaction. It involves reading the victim SST file into memory, performing a merge-sort, and writing the new SST file to storage. This storage I/O requires one read and one write operation. Figure 1(b) shows the actual implementation as analyzed at the code level in RocksDB [4], a representative LSM-KVS. Merge-sort involves reading and writing the victim SST files for each key, necessitating I/O operations for every key-value pair subjected to compaction and consequently amplifying I/O time. RocksDB adopts this design to avoid resource waste that could occur when reading all SST files overlapping the key range of the victim SST file.

Figure 2 shows the actual I/O time spent during the compaction process as measured on RocksDB, compared to the ideal I/O time values assuming minimal I/O overhead. The ideal I/O time was calculated using sequential read/write results from the FIO benchmark [5] in our experimental environment. The detailed experimental environment is described in Section IV. The difference between compaction I/O time and ideal I/O time for RocksDB worsens as the compaction size increases, with an average of $7.6\times$ and a maximum of $15.9\times$ higher. We also found that I/O time accounts for 74% of the total compaction time when analysed with Perf [6]. This shows that the amplification of I/O time had a significant impact on compaction.

III. Design

We propose the following design to improve the I/O stack, assuming a situation where the host server runs the LSM-KVS instance directly and stores data, while the remote server only offloads compaction from the host.

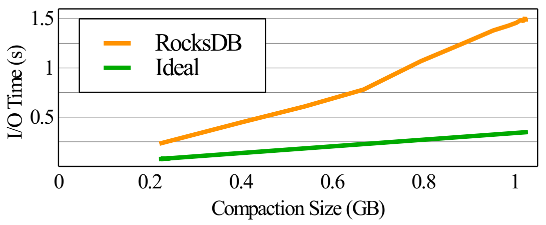

Fig. 2. Comparison of ideal I/O time and actual I/O time in RocksDB according to compaction size.

- **Typical Network-based Compaction Offloading:** Figure 3(a) shows a basic design for offloading compaction over a network. The host server and remote server use a Network FIle System to access data on the host server from the remote server. Both the host and remote servers go through the filesystem as well as an additional I/O stack at the network layer.
- **NVMe-oF-based Compaction Offloading:** Figure 3(b) shows the design of compaction offloading using SPDK and NVMe-oF. Using SPDK and NVMe-oF on the host and remote server to access NVMe devices can dramatically lighten the I/O stack by eliminating the need to traverse kernel space such as the filesystem. However, synchronization of concurrent data access needs to be managed.

IV. Preliminary Result

Experiment Setup. To verify whether reducing the I/O stack overhead with SPDK and NVMe-oF during compaction offloading can improve the compaction performance, we performed the following evaluation in two environments. In each environment, the host and remote server are the same hardware, and the server details are shown in Table I.

- **Baseline:** RocksDB v8.1.1 is used to implement compaction offloading. The host and remote server use the Network File System to share files.
- **NVMe-oF:** RocksDB v8.1.1 and SPDK v24.01 were used. RocksDB was run exclusively on the remote server, using host server-side storage device connected via NVMe-oF.

Baseline compaction offloading is fully implemented and experimented with, but NVMe-oF version was not fully implemented due to synchronisation issues. Therefore, for a fair comparison, we only compare compaction performance for both systems when running the FillRandom workload in db_bench for 16B-1024B (Key-Value) for 300 seconds.

Results. Table II shows the evaluation results. Compared to the baseline, NVMe-oF reduced the average time required for compaction by 37.2%, and the proportion of time spent on I/O also decreased. As a result, NVMe-oF achieved 37.7% higher compaction throughput compared to the baseline.

We measured network bandwidth usage and found that the peak usage with NVMe-oF exceeded twice the baseline, although it did not reach the maximum bandwidth (10 Gb/s).

TABLE I
Testbed Specifications.

CPU	Ryzen 9 7950X, 16 Cores, Up to 5.7 GHz
Memory	32 GB (16 GB × 2) DRAM
Storage	Samsung 970 EVO 1 TB
OS	Ubuntu 22.04.4 / Linux Kernel 5.10.0
Network	10 Gbps Ethernet

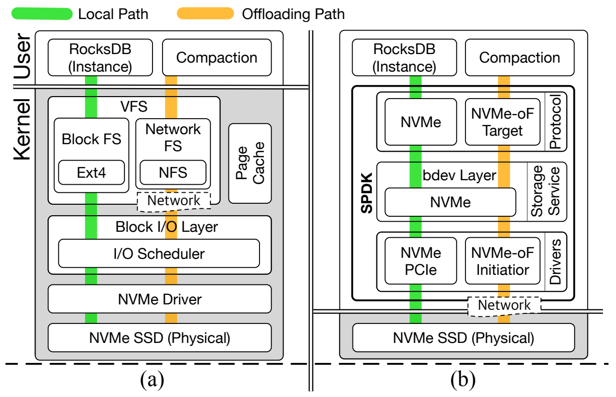

Fig. 3. I/O stack of compaction offloading using (a) NFS and (b) NVMe-oF.

This indicates that network bottlenecks in compaction offloading are not the main cause of performance degradation and that the problem lies in the I/O stack.

TABLE II
Evaluation Results.

	Baseline	**NVMe-oF**
Avg. I/O Time (sec)	1.542	0.968
Avg. I/O Ratio (%)	65.7	54.5
Compaction Throughput (MB/s)	119.4	164.51
Max. Network Bandwidth Usage (Gb/s)	3.1	6.5

V. Conclusion & Future Work

We analyzed the I/O characteristics of compaction in Log-structured Merge-tree based Key-Value Stores (LSM-KVS) and reveal that compaction delays are caused by the resulting I/O time amplification. To mitigate this I/O time amplification during compaction offloading, we propose an SPDK and NVMe-oF based compaction offloading I/O stack refinement. Preliminary experimental results show that our proposed method improves compaction throughput by 37.7%. For future work, we aim to fully integrate compaction offloading using SPDK and NVMe-oF, with the goal of enhancing LSM-KVS performance through the refined I/O stack.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. NRF-2021R1A2C2014386) (No. RS-2024-00416666).

References

[1] Q. Yu, C. Guo, J. Zhuang, V. Thakkar, J. Wang, and Z. Cao, "Caas-lsm: compaction-as-a-service for lsm-based key-value stores in storage disaggregated infrastructure," *Proceedings of the ACM on Management of Data*, vol. 2, no. 3, pp. 1–28, 2024.
[2] J. Kim, H. Yoo, S. Lee, H. Byun, and S. Park, "Coordinating compaction between lsm-tree based key-value stores for edge federation," in *2024 IEEE 17th International Conference on Cloud Computing (CLOUD)*. IEEE, 2024.
[3] 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)*. IEEE, 2017, pp. 154–161.
[4] Facebook, "Rocksdb: A persistent key-value store for fast storage environment," https://rocksdb.org, 2012.
[5] Axboe, J, "Github—axboe/fio: Flexible i/o tester," 2021. [Online]. Available: https://github.com/axboe/fio
[6] A. C. De Melo, "The new linux'perf'tools," in *Slides from Linux Kongress*, vol. 18, 2010, pp. 1–42.