Boosting Compaction Performance of LSM-tree-based KV Stores in Multi-Near-Data Processing Systems

Hui Sun, Qiang Wang
Anhui University
Hefei, China
{sunhui,e19201078}@ahu.edu.cn

Yinliang Yue*, Yuhong Zhao*
Chinese Academy of Sciences
89 Minzhuang Road, Beijing, China
{yueyinliang,zhaoyuhong}@iie.ac.cn

Song Fu
University of North Texas
Denton, Texas, USA
Song.Fu@unt.edu

CCS CONCEPTS

• **Computer systems organization → Embedded software; Heterogeneous (hybrid) systems; Embedded hardware**.

KEYWORDS

Near-Data Processing, LSM-Tree, Data layout, Parallel Computing, Load Balancing

ACM Reference Format:
Hui Sun, Qiang Wang, Yinliang Yue*, Yuhong Zhao, and Song Fu. 2018. Boosting Compaction Performance of LSM-tree-based KV Stores in Multi-Near-Data Processing Systems . In *50th International Conference on Parallel Processing (ICPP 2021), August 9-12, 2021, State, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

The LSM-tree [1] has been the primary data structure of key-value stores (KV stores), which comprises one memory component and multiple components on disk. But its compaction may lead to serious write amplification, which is a performance bottleneck. Under random-write-intensive workloads, there can be long-time compaction that blocks user-level write/read operations. The compaction process may also lead to much data movement and high CPU costs. Random-write I/Os are prevalent in real-world workloads; thus, the bottleneck slows down user-level performance.

With the advancement of embedded devices' computing power, a near-data processing (NDP) model has attracted attention in academia and industry. This model offloads computing tasks to storage devices to reduce data movement between the storage device to the host-side CPU and save system resources (I/O resources and CPU resources). NDP systems have been developed to improve the performance of KV stores [2]. However, there is no studies have focused on near-data computing clusters. In this paper, we investigate the use of near-data computing clusters to enhance the compaction performance of KV stores, named MStore that consists of multiple NDP devices.

*Yinliang Yue and Yuhong Zhao are corresponding authors

ICPP 2021, August 9-12, 2021, State, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06...$15.00
https://doi.org/10.1145/1122445.1122456

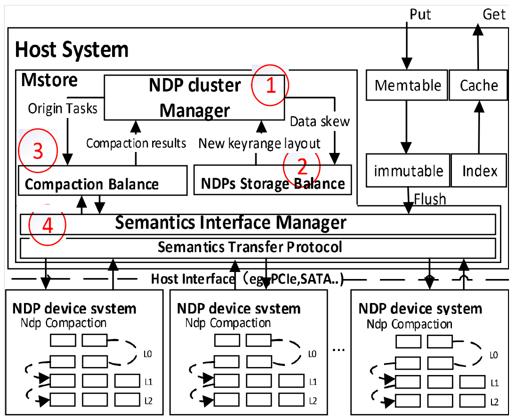


Figure 1: Overview system of MStore

2 OVERVIEW SYSTEM

As shown in Fig. 1, MStore includes an NDP-cluster management module, an NDP-cluster storage balancing module, an NDP-cluster computing load balancing module, a data transmission protocol, and a semantic interface between the host- and device-side subsystems. We mainly optimized the write performance.

2.1 Data organization and distribution

We propose a multi-column LSM-tree structure to divide the dataset into multiple key ranges. Each NDP device that manages one or more key ranges performs compaction on its own managed data. Key ranges on each device can be dynamically adjusted to lessen data skew. In addition, data in the same column of a multi-column LSM-tree are independent; thus, data migration is avoided among NDP devices in the process of compaction. Some key ranges belong to an NDP device at a time, which is expressed by a variable named owner. When the owner is -1, it means that the key range, which previously belonged to device NDP_i, is not currently belonged to device NDP_i. The key-range possess other information, *e.g.*, hot/cold feature and key-range sizes.

2.2 Data load balancing

Key-range Adjustment. MStore distributes workload on each NDP device by adjusting its assigned key ranges. The key-range adjustment can achieve two goals, (1) avoiding data storage skew and (2) minimizing the impact on the read performance. Thus, the

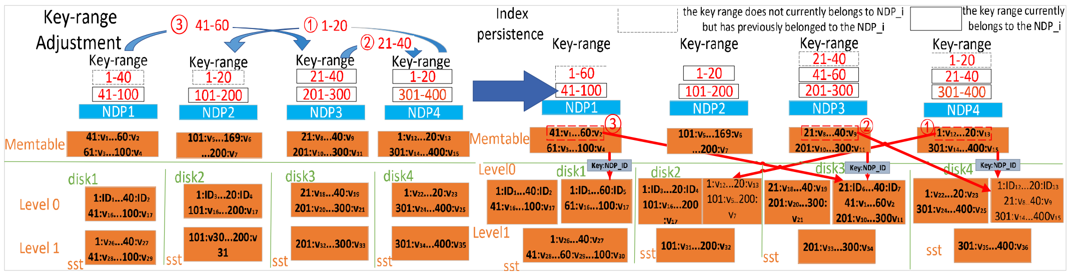


Figure 2: An example of Key-range adjustment

overlapping areas in the LSM-tree should be minimized. For example, at time t1, the distribution of each key range is presented on the left hand of Fig. 2. Three cases require key-range adjustment.

Case 1. MStore detects that device NDP_4 needs to move part of key ranges to device NDP_ 2. Device NDP_4 has two key ranges. If 1-20 is moved to NDP_2, it will not extend the overlapping area and increase the overhead of read operations. Thus, moving 1-20 has the highest priority. Since moving key ranges to NDP_2 extends the overlapping area, moving key ranges 301-400 has the lowest priority. Case 2. Device NDP_3 moves a portion of key ranges to device NDP_4. When 21-40 is moved to device NDP_4, the overlapping area will not extend. However, querying the data of key range 21-40 requires an additional NDP device (NDP_4). Thus, moving 21-40 has the second-highest priority. Similarly, moving 201-300 has the lowest priority. Case 3. Device NDP_1 should move part of the key ranges to device NDP_3. Since key range 1-40 does not belong to NDP_1, key range 41-60 conducts migration even though key range 41-100 has the lowest priority.

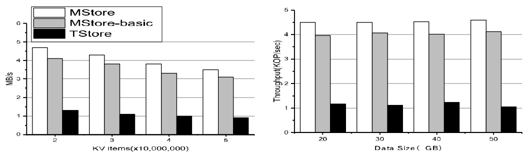
Index persistence of the overlapping area. After key-range adjustment, the host distributes the corresponding data to each NDP device according to the new key-range distribution. An NDP device needs to insert <key:NDP_ID> in the previously owned key range. As shown in Fig. 2, the key range is represented by a dotted box. These key ranges contain expired data, and then <key:NDP_ID> overwrites the expired data in compaction. In addition, <key:NDP_ID> also plays a vital role in the read performance, *i.e.*, it helps the host distinguish the real values or index.

2.3 Computation balancing

According to the multi-key range-based data layout (see Section 2.1), each NDP device can select a compaction task from its own managed dataset, avoiding compacted data movement between NDP devices. However, in asynchronous mode, the workload of each NDP compaction may vary significantly; thus, extra task balancing is designed to balance workloads on NDP devices, which also enhances resources utilization on NDP devices.

3 EVALUATION

An NDP SSD is not available in the test. We use a PC with an E6700 core as a host and four embedded development boards as NDP devices forming a cluster. Each board has two Cortex A72 cores and four Cortex A53 cores. Data transmission between the host and the NDP device cluster is implemented through an Ethernet switch. We compared the performance of MStore with MStore-basic, and TStore [2] (a dynamic task offloading NDP-enabled KV store). In MStore-basic, the storage balancing and the compaction computation balancing are disabled when studying the write performance.



(a) DB_Bench (b) YCSB-C

Figure 3: Performance under write-intensive workloads with different data sizes.

We use DB_Bench and YCSB-C as benchmark programs. DB_Bench is placed in LevelDB while YCSB-C simulates random-write workloads. In Fig. 3, MStore achieves performance improvement by 3.88x and 3.5x than TStore under DB_bench and YCSB-C, respectively. Compared with MStore-basic, MStore's load balancing method contributes 14.0% performance improvement on average.

4 CONCLUSION

We propose a near-data processing cluster named MStore to improve the write performance of KV stores. According to the data organization structure, we design a dynamic key-range adjustment method to balance data storage load and a compaction-aware computation balancing method to alleviate the 'cannikin law' caused by uneven compaction on NDP devices due to data skew in the NDP cluster. We evaluate the performance of MStore and compare it with TStore and MStore-basic on a tested platform.

REFERENCES

[1] P. O'Neil, E. Cheng, D. Gawlick, and E. O'Neil. 1996. The log-structured merge-tree (LSM-tree). *Acta Informatica* 33, 4 (1996), 351–385.
[2] Hui Sun, Wei Liu, Jianzhong Huang, Song Fu, Zhi Qiao, and Weisong Shi. 2019. Near-Data Processing-Enabled and Time-Aware Compaction Optimization for LSM-tree-based Key-Value Stores. In *Proceedings of the 48th International Conference on Parallel Processing*. 1–11.