BlockListDB: Progressive Compaction from Fine-Grained Lists to Coarse-Grained Blocks

Submission No. 232

ABSTRACT

Most of byte-addressable key-value stores designed for NVMM leverage the in-place structural modification operations (SMOs) to insert, update, or delete keys through 8-byte pointer updates. While this in-place SMOs help mitigate the write amplification problem, excessive in-place updates lead to decreased spatial locality, resulting in degraded read performance. In this study, we propose SkipBlockLists optimized for NVMM as an intermediate layer for serializing byte-addressable in-memory SkipList indexes into SSTables optimized for block devices. SkipBlockLists manage multiple keys within a single node, combining the advantages of SkipLists and B-trees. BlockListDB effectively reduces the amount of coarse-grained IO by using in-place SMO, while improving locality within node through copy-on-write. Through extensive performance study, we show that BlockListDB outperforms other state-of-the-art LSM trees by a large margin.

CCS CONCEPTS

• **Information systems → Key-value stores**.

KEYWORDS

Non-Volatile Main Memory, Log-Structured Merge-Tree, Key-Value Store

ACM Reference Format:
Submission No. 232 . 2025. BlockListDB: Progressive Compaction from Fine-Grained Lists to Coarse-Grained Blocks . In *Proceedings of ACM SIGMOD/PODS International Conference on Management of Data (SIGMOD '25).* ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

The LSM tree employs an in-memory index known as MemTable to buffer key-value pairs [15]. When a substantial amount of data accumulates in the buffer to efficiently utilize disk bandwidth, it's flushed to disk as a single file named SSTable. As the flushed SSTable overlaps with previously flushed ones in terms of key ranges, compaction (merge-sort) is executed in the background. Due to the characteristics of block devices, Copy-on-Write (CoW) is commonly utilized to merge-sort SSTables, resulting in many key-value pairs being rewritten to disk. Consequently, this write amplification problem degrades compaction performance, and if merge-sort performance is poor, the number of overlapping SSTables increases, which hurts search performance. To mitigate these challenges, the *write stall* mechanism has been employed, temporarily pausing client insertions.

Various studies have been conducted to reduce write amplification in key-value stores [9, 19]. In particular, many research efforts have explored performing merge-sort using small IO granularities such as cacheline or XPLine in byte-addressable NVMM. Radical designs, such as FlatStore [3], have been proposed, arguing that recovery is feasible if data is logged in NVMM, even if the index is stored only in DRAM.

In contrast, ListDB [10] proposes a more conservative technique to gradually transform append-only log entries into SkipList nodes, eliminating the need for the entire index to reside in memory. Additionally, ListDB minimizes write amplification by preventing key-value pairs from being rewritten even when SkipLists are repeatedly merge-sorted, as it implements an in-place SkipList merge-sort algorithm. However, although the in-place merge-sort algorithm, called *Zipper compaction*, effectively reduces write amplification, they suffer from low locality since the positioning on NVMM is determined by the order of key-value insertions, regardless of key proximity. Moreover, there is a significant drawback in terms of metadata management if the index points to individual key-value pairs (as in FlatStore [3] or Prism [18]), or if individual key-value pairs are managed as independent nodes in the index. Especially when the size of key-value pairs is small, the size of metadata becomes larger than the data itself.

With the discontinuation of Optane DCPMM, it is anticipated that the next-generation NVMM will be either smaller-capacity NVDIMMs or Memory Semantic SSDs that employ small DRAM as internal buffers. Therefore, it is rational to consider NVMM as a storage tier positioned between DRAM and block devices. This necessitates the design of a NVMM layer that enables easy transformation into SSTables.

This study introduces a novel data structure called SkipBlockLists, which serves as an intermediate index layer between byte-addressable in-memory SkipLists and SSTables optimized for block devices. SkipBlockLists inherits the advantages of lock-free SkipLists because it adopts the same structural modification operations (SMOs) as lock-free SkipLists. While SkipLists inherently suffer from poor spatial locality and require a significant amount of metadata since they need one or more pointers for each key-value pair, SkipBlockLists improve spatial locality by storing multiple key-value pairs in an array within a single node, and thereby they reduce the number of nodes of the linked list.

Based on this SkipBlockLists, this study introduces a new layer for NVMM between DRAM and block device storage. LSM-trees utilize data structures optimized for DRAM and block devices respectively, but significant serialization and deserialization costs occur when moving data from memory to storage, limiting the benefits of low latency NVMM. Specifically, fine-grained SkipLists, where one key-value pair is stored per node, are employed in DRAM, while arrays of tens of MBs in size are utilized in coarse-grained block devices. For the NVMM layer, SkipBlockLists in their intermediate

SIGMOD '25, June 22–27, 2025, Berlin, Germany

Submission Summary

Conference Name
SIGMOD International Conference on Management of Data (2025)

Track Name
Round 2 Paper Submissions

Paper ID
232

Paper Title
BlockListDB: Progressive Compaction from Fine-Grained Lists to Coarse-Grained Blocks

Abstract
Most of byte-addressable key-value stores designed for NVMM leverage the in-place structural modification operations (SMOs) to insert, update, or delete keys through 8-byte pointer updates. While this in-place SMOs help mitigate the write amplification problem, excessive in-place updates lead to decreased spatial locality, resulting in degraded read performance. In this study, we propose SkipBlockLists optimized for NVMM as an intermediate layer for serializing byte-addressable in-memory SkipList indexes into SSTables optimized for block devices. SkipBlockLists manage multiple keys within a single node, combining the advantages of SkipLists and B-trees. By using SkipBlockLists as an intermediate layer, BlockListDB demonstrates superior performance compared to ListDB. BlockListDB effectively reduces the amount of coarse-grained IO by using in-place SMO, while improving locality within node through copy-on-write. Through extensive performance study, we show that BlockListDB outperforms other state-of-the-art LSM trees by a large margin.

Created
4/8/2024, 6:43:35 PM

Last Modified
4/8/2024, 10:48:07 PM

Authors
Juwon Chung (Sungkyunkwan University) <juwon2277@gmail.com> ✓ **A**
Wonbae Kim (Kakao) <zwigul@gmail.com> ✓
Sangeun Chae (Sunkyunkwan University) <andrewchae48@gmail.com> ✓
Beomseok Nam (Sungkyunkwan University) <bnam@skku.edu> ✓ **A**

Primary Subject Area
Data Management Systems -> Storage, indexing, and physical database design

Secondary Subject Areas

Data Management Systems -> Database systems on emerging hardware

Submission Questions Response

1. Conflicts

I have read the "Conflicts of Interest" policy in the Call for Research Papers and will ensure that:

(a) every author has an account on CMT, and

(b) every author will have entered PC and domain conflicts prior to the submission deadline.

(c) every author has provided an ORCID

****EACH author needs to enter their domain conflicts and individual conflicts with PC members by accessing these options on their CMT account menu****

Agreement accepted

2. Novelty

Our submission satisfies the "Duplicate Submission and Novelty Requirements" explained in the Call for Research Papers.

We understand that, to enforce this requirement, high-level metadata of submissions (title, abstract, list of authors), may be shared with the Program Chairs / Editors of other conferences and journals.

Agreement accepted

3. Double Anonymous

We understand that all research track submissions need to be appropriately anonymized according to the double-anonymization rules.

Agreement accepted

4. Online posting

We understand that to avoid compromising the double-anonymity requirement, authors should refrain from publicizing and uploading versions of their submitted manuscripts to pre-publication servers, such as arXiv, and other online forums very close to the time of submission or during the reviewing period. If a version of a submission already resides on a pre-publication server, such as arXiv, dated more than 2 weeks prior to the submission deadline, the authors do not need to remove it before submitting to SIGMOD.

Agreement accepted

5. Authorship

We understand that if the submission undergoes revision or is accepted, no new authors can be added, and no authors can be removed.

Agreement accepted

6. Research Involving Human Subjects

We understand that, as a published ACM author, we are subject to all ACM Publications Policies, including ACM's new Publications Policy on Research Involving Human Participants and Subjects.

Agreement accepted

7. Artifact Availability Requirements

SIGMOD strives to establish a culture where sharing research artifacts (data, results, code, and scripts) is the norm rather than an exception. Although it is not mandatory for acceptance, providing this extra material can help reviewers evaluate your work more thoroughly.

Please include a link with your materials in the text box provided below at the time of submission. The link and materials should preserve anonymity. For example, this may be an anonymized GitHub repository using https://anonymous.4open.science/. You may want to make sure that the link you provide is not indexed by search engines. On GitHub, you can do so by adding the following to the page head: <meta name="robots" content="noindex">.

*This link is expected to be live within *one* week from submission. Reviewers that may need to look at the materials will not do so earlier than that. Note, that we do not expect a fully polished submission in terms of automatically reproducing results, but rather a reasonably clean version of the state of the code when submitting the paper. Please, do not use a shortened link which traces who accesses it.*

In the event that you are not able to submit your code, data, scripts, and notebooks please explain in the text box provided below why this is the case.

[Not Answered]

8. Artifact Anonymity Confirmation

We understand that artifacts must also satisfy double-anonymity requirements, just like the main submission. We understand that violation of anonymity in artifacts will lead to rejection.

Agreement accepted