

# Query-driven compaction in LSM-trees

Shubham Kaushik  
Boston University  
USA

\*\*\*\*  
Boston University  
USA

\*\*\*\*  
Boston University  
USA

## ABSTRACT

In modern data systems, particularly for write-intensive workloads, log-structured merge (LSM) trees have become the most widely used technique. The idea of out-of-place updates, which logically invalidate keys that may exist on multiple levels, helps LSM-trees excel for write-heavy workloads. However, current LSM-tree designs do not capitalize on the sort-merge operations performed for the range queries, which leads to carrying out almost the same amount of work for each range query, even when they are same or overlapping. This inefficiency can result in the wastage of CPU cycles and unnecessary I/O operations in worst-case scenarios due to presence of logically invalid keys.

In this paper, we present a query-driven compaction strategy that removes the invalid (logically deleted) keys from the LSM-tree and flushes back the valid keys filtered by sort-merge performed during a range query. We conduct performance experiments with the help of a custom implementation in one of the popular LSM-based data stores, RocksDB.

## ACM Reference Format:

Shubham Kaushik, \*\*\*\*, and \*\*\*\*. 2023. Query-driven compaction in LSM-trees. In *Proceedings of ACM Conference (Conference'17)*. ACM, New York, NY, USA, 1 page. <https://doi.org/10.1145/nnnnnnnn.nnnnnnn>

## 1 INTRODUCTION

**Write heavy workloads with LSM.** The rising research in the field of robotics and IoT devices, combined with the integration of artificial intelligence and machine learning, has resulted in the generation of vast amounts of data. This data often requires real-time processing and exhibits a write-heavy nature. Data stores like RocksDB and LevelDB have been designed to efficiently handle such workloads. These data stores rely on the technique of **log-structured merge (LSM)** trees, an efficient data structure tailored for managing write-heavy workloads. The fundamental concept underlying LSM trees is of out-of-place updates, which logically invalidate keys instead of performing in-place updates.

**Compaction.** The LSM trees are composed of multiple levels, each of which is a sorted run of key-value pairs. The compactions are performed to merge the sorted runs from the lower levels into the higher levels. The process is triggered when the size of a level exceeds a certain threshold. It helps in removing the stale data from LSM and making room for new data in lower levels.

**Range queries.** The LSM trees are optimized for write-heavy workloads, but it also supports range queries. The range queries are performed by merging the sorted runs from multiple levels and filtering out the keys that have been logically invalidated. This process is also called **sort-merge**, which is similar to the compaction.

**Problem.**

## 2 LSM BACKGROUND

## 3 PROBLEM

## 4 SOLUTION NAME

## 5 EVALUATION

## 6 RELATED WORK

## 7 CONCLUSION

---

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

*Conference'17, July 2017, Washington, DC, USA*

© 2023 Association for Computing Machinery.

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...\$15.00

<https://doi.org/10.1145/nnnnnnnn.nnnnnnn>