Summary of LSM Tree Survey

Prepared By
MD.Omer Danish
Student ID: 1505053

Submitted to
Md. Tarikul Islam Papon
Lecturer

Department of Computer Science
Bangladesh University of Engineering and Technology

Most of the modern systems are using The Log-Structured Merge-tree (LSM) in the storage layer to a great extent. Authors of this paper provide a general taxonomy to classify the literature of LSM-trees and discuss their strengths and trade-offs.This design of LSM tree brings a number of advantages.Authors of this paper survey these recent research efforts on improving LSM-trees. They also survey some representative LSM-based NoSQL systems.

This paper contains mainly four topics.**Firstly** this paper is reviewing history of LSM-trees and LSM-tree implementations at present. **Secondly** it presents a taxonomy of the proposed LSM-tree improvements and surveys the existing work using that taxonomy. **Thirdly** it surveys some representative LSM-based NoSQL systems, focusing on their storage layers. **Fourthly** it reflects on the result of this survey, identifying several outages and opportunities for future work on LSM-based storage systems.

LSM tree maintains out-of-place update structure.This design improves write performance and can also simplify the recovery process by not overwriting old data.The major problem of this design is that read performance is sacrificed since a record may be stored in any of multiple locations. The LSM-tree addressed the problems of log structured tree by designing a merge process which is integrated into the structure itself, providing high write performance with bounded query performance and space utilization.

Today's LSM-tree implementations commonly exploit the immutability of disk components to simplify concurrency control and recovery. Leveling merge policy of LSM Tree optimizes for query performance since there are fewer components to search in the LSM-tree. The tiering merge policy is more write optimized since it reduces the merge frequency.There are two well-known optimizations that are used by most LSM-tree implementations today.One is Bloom filter and second is Partioning.Partioning is to range-partition the disk components of LSM-trees into multiple (usually fixed-size) small partitions.The LSM-tree is highly tunable. For example, by changing the merge policy from leveling to tiering, one can greatly improve write performance with only a small negative impact on point lookup queries due to the Bloom filter

Despite the popularity of LSM-trees the basic LSM-tree design suffers from various draw backs and insufficiencie. These are Write Amplification,Merge Operations,Hardware,Special Workloads,Auto-Tuning. Secondary Indexing.Based on these major issues of the basic LSM-tree design,authors of the paper present a taxonomy of LSM-tree improvements.One Way to optimize write amplification is tiering with horizontal or al grouping.Another way to optimize write performance skip-tree that proposes a merge skipping idea or Exploiting Data Skew.Write amplification for skewed update workloads are reduced where some hot keys are updated frequently. The basic idea is to separate hot keys from cold keys in the memory component so that only cold keys are flushed to disk.

The paper reviews some existing work that improves the implementation of

merge operations, including improving merge performance, minimizing buffer cache misses, and eliminating write stalls.

This paper reviews the LSM-tree improvements proposed for different hardware platforms, including large memory, multi core,SSD/NVM, and native storage.

This reviews LSM-tree improvements that target special workloads like temporal data, small data, semi-sorted data, and append-mostly data to achieve better performance. The improvements(LSM-trie,SlimDB) presented in the paper each target a specialized workload.

The authors of this paper reviews some research task that are done to provide auto tuning of LSM trees.These works are Auto-Tuning,Parameter Tuning,Tuning Merge Policies,Dynamic Bloom Filter Memory Allocation,Optimizing Data Placement.
The authors have discussed LSM-based secondary indexing techniques to support efficient query processing, including index structures,index maintenance, statistics collection, and distributed indexing.These indexing techniques are Index Structures,Index Maintenance,Statistics Collection,Distributed Indexing .

Authors also survey five representative LSM-based open source NoSQL systems namely LevelDB, RocksDB , Cassandra, HBase, and AsterixDB. LevelDB is an LSM-based key-value store that supports a simple key-value interface including puts, gets, and scans. Apache HBase is a distributed data storage system in the Hadoop ecosystem that is based on a master-slave architecture. Apache AsterixDB is an open-source Big Data Management System (BDMS) that aims to manage massive amounts of semi-structured data efficiently.

At the end of the paper the authors briefly discusses some future research directions what they get from the results of their survey.The fields are Thorough Performance Evaluation,Partitioned Tiering Structure,Hybrid Merge Policy,Minimizing Performance Variance,Towards Database Storage Engines.The authors presented everything systematically.Reading this survey paper has been a great experience.