LogStore: A Workload-aware, Adaptable Key-Value Store on Hybrid Storage Systems (Extended abstract)

Prashanth Menon*, Thamir M. Qadah◇, Tilmann Rabl†, Mohammad Sadoghi‡, Hans-Arno Jacobsen#
*School of Computer Science, Carnegie Mellon University
◇Umm Al-Qura University and Purdue University
†Database Systems and Information Management Group, TU Berlin
‡University of California, Davis
#Middleware Systems Research Group, University of Toronto, Canada

***Abstract*—Due to the recent explosion of data volume and velocity, a new array of lightweight key-value stores have emerged to serve as alternatives to traditional databases. The majority of these storage engines, however, sacrifice their read performance in order to cope with write throughput by avoiding random disk access when writing a record in favor of fast sequential accesses. But, the boundary between sequential vs. random access is becoming blurred with the advent of solid-state drives (SSDs).**

In this work, we propose our new key-value store, LogStore, optimized for hybrid storage architectures. Additionally, introduce a novel cost-based data staging model based on log-structured storage, in which recent changes are first stored on SSDs, and pushed to HDD as it ages while minimizing the read/write amplification for merging data from SSDs and HDDs. Furthermore, we take a holistic approach in improving both the read and write performance by dynamically optimizing the data layout, such as deferring and reversing the compaction process and developing an access strategy to leverage the strengths of each available medium in our storage hierarchy. Lastly, in our extensive evaluation, we demonstrate that LogStore achieves up to 6x improvement in throughput/latency over LevelDB, a state-of-the-art key-value store.

I. Introduction

Big data challenges are not characterized only by the large volume of data that has to be processed but also by a high rate of data production and consumption. The explosion in data volume and velocity is commonplace in a wide range of monitoring applications. In modern monitoring applications, many thousands of sensors continuously produce a multitude of readings that have to be stored at a high pace but have to also be readily available for continuous query processing. Solutions such as log-structured key-value storage systems that address the velocity problem often rely on modern storage hardware such as solid-state drives (SSDs) that provides faster access to persistent storage as the bridge the gap between sequential versus random access.

Existing solutions either use SSDs as the only storage medium or as a buffer to traditional hard-disk drives (HDDs) for improving the system's performance. Unfortunately, the usage of SSDs is not cost-effective in these cases because increase in the cost of using SSDs does not translate to an increase in performance with the same rate. For example, the cost of using SSDs in the cloud is $4\times$ the cost of HDDs and the performance increase is much lower than $4\times$.

In this work, we rethink the design of modern log-structured key-value (KV) stores. We present *LogStore*, a novel optimized hybrid storage architecture that serves as a key building block for distributed key-value stores in order to sustain high-velocity and high-volume data. In *LogStore*, we introduce a database staging mechanism using a novel, cost-based, log-structured storage system such that recent changes are first stored on SSDs, and as the data ages, it is pushed to HDD, while minimizing the read and write amplification for merging and compaction of data from SSDs and HDDs. We also ensure that all writes on both SSD and HDD are sequential in large block sizes. Furthermore, we develop a holistic approach to improve both read and write performance by dynamically optimizing the data layout based on the observed access patterns.

The contributions of *LogStore* are as follows: (1) An analytical cost model to estimate the performance of log-structured hybrid storage systems. The model accounts for access pattern and specific system characteristics, provides insights that guide the design of *LogStore* and reveals bottlenecks in LevelDB. (2) A new statistics-driven informed compaction process that retains only the hottest data on the SSD and evicts cold data to the HDD to achieve maximum throughput. (3) A new reversed compaction process that identifies hot data stored on HDD and migrates this data to the SSD through compaction. This technique also leverages statistics to remain adaptive to shifting workloads. (4) An optimization that enables faster write throughput by selectively deferring compactions based on access frequency. Reducing compaction execution offers faster overall throughput , and (5) a new compaction process that operates within a single level (termed staging compaction). This compaction process reduces the impact of having overlapping ranges of SSTs (which is unique to *LogStore*) on I/O performance of read operations.

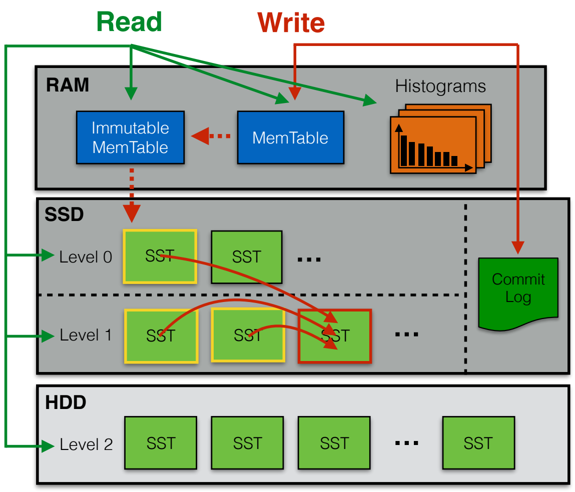

Fig. 1. *LogStore* architecture

II. LogStore Overview

Our proposed analytical model [3] clearly establishes a connection between the number of levels on each device, the access rates to each device and the throughput that can be expected of a hybrid storage system. Specifically, in a hybrid system that is bottlenecked by the HDD both read and write throughput can be improved by storing at most one level on HDD.

The architecture of *LogStore*, shown in Fig 1. Writes in *LogStore* are buffered in a Memtable and written out to a commit log. When the Memtable has reached a configurable size, it is converted into a read-only immutable Memtable. When this occurs, a new Memtable is created to handle new writes while the Immutable Memtable is simultaneously flushed to the first level as an SST. *LogStore* structures all SSTs into a series of three levels, the first two of which are on SSD while the last is on an HDD.

SSTs within a level are disjoint in the keys they store while SSTs across levels may overlap in key ranges, and often do in skewed workloads. *LogStore* does not size the levels such that they grow exponentially, but rather arranges the levels so that the total amount of data stored on the SSD (combined between Level-0 and Level-1) is a configurable fraction of the total amount of data. In most of our experiments, the SSD stores 50% of the total data.

The *LogStore* architecture has three main goals: (1) Store the hottest data on the SSD while evicting the coldest data to the HDD. (2) Perform as much of the I/O-intensive, preparatory work on the SSD as possible. (3) Ensure at most one seek for reads on HDD. The proposed optimizations strive to achieve these goals. The proposed informed compaction selects coldest SSTs for compaction to HDD. When some SSTs become hot again, the reverse compaction process bring them back to SSDs. *LogStore* avoids initiating frequent compactions under write-heavy workloads by relaxing the size constraints on levels, and allowing SSTs to overlap. Finally, *LogStore* uses staging compactions, which joins multiple overlapping SSTs and keep them on the same level to reduce the overhead of searching multiple SSTs for read operations. We refer the reader to our full paper [3] for detailed description of these optimizations.

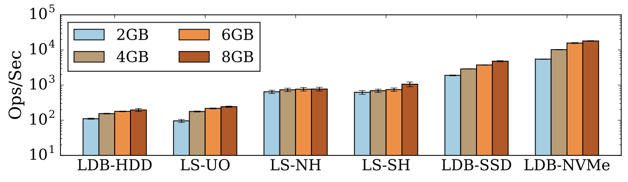

Fig. 2. Throughput for Workload-C with varying RAM sizes

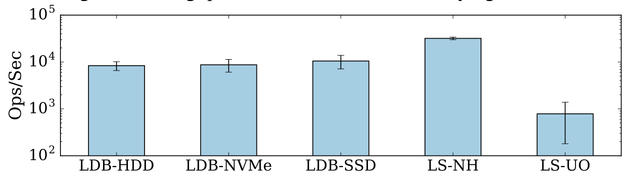

Fig. 3. Throughput for write-only workload

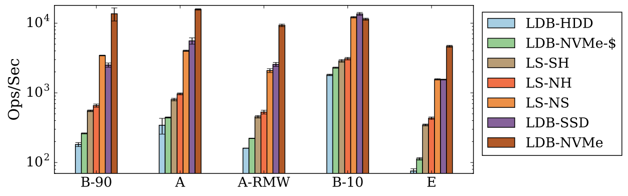

Fig. 4. Throughput for mixed read-write workloads

III. Evaluation

In this section, we highlight some of the interesting results from our evaluation of *LogStore* against *LevelDB* [1]. Extensive experiments are performed with various workloads based on YCSB [2] (i.e., A, B-10, B-19, C, A-RMW, and E), and use different configurations of *LogStore* (e.g., with NVMe and regular SSD). *LogStore* achieves 6× better throughput than *LevelDB* (*LogStore* in Fig. 2) running on HDD in read-only workloads, and up to 3.6× better throughput than *LevelDB* (Fig. 3) running on SSD/NVMe when executing a write-only workload. Across mixed read-write workloads, *LogStore* has 1.5-5.7× better throughput than *LevelDB* on HDD (Fig. 4).

IV. Conclusions

In this paper, we give an overview of *LogStore*, a new key-value store architecture that is workload-aware, dynamic, and designed to operate in a hybrid storage environment. *LogStore* implements informed, reverse, deferred, and staging compactions that are each driven by low-overhead statistics and a cost-benefit run-time analysis to make key-value stores use SSDs more effectively.

References

[1] LevelDB, a fast key-value storage library by google. https://code.google.com/p/leveldb/.

[2] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears. Benchmarking Cloud Serving Systems with YCSB. In *SoCC '10*, pages 143–154, 2010.

[3] P. Menon, T. Qadah, T. Rabl, M. Sadoghi, and H. A. Jacobsen. Logstore: A workload-aware, adaptable key-value store on hybrid storage systems. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2020.