IITB Summer Internship 2017

Software Requirement Specification

Performance Improvement of MongoDB and MySQL using RocksDB Engine and subsequent integration into OpenEdX

Principal Investigator

Prof D B Phatak

Project In-charge

Mr. Nagesh Karmali

Project Mentors

Miss. Firuza Aibara

Mr. Nithin S

Project Members

Abhishek Chattopadhyay

Ekta Agrawal

Simran Goyal

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

EdX is a MOOC(Massive Open Online Course) platform started by MIT and Harvard in collaboration for revolutionizing online education. EdX is among the most popular MOOC's in the world because of the advance features it provides for making online education of student fruitful and more effective worldwide. It also provides the author an effective way to design a course.
OpenEdX is the open source platform based on the EdX platform and is created on the Django framework (written in python). RocksDB, a storage engine has been presented by Facebook Inc. as a better alternative to InnoDB, both in terms of throughput and reduced storage space requirements.
The project deals with integration of MongoRocks and MyRocks, built by integrating RocksDB into MongoDB and MySQL, respectively, in the OpenEdX framework and run benchmark tests to check for its feasibility and viability.

1.1 Purpose

This purpose of this Software Requirement Specification (SRS) document is to provide a glance into the feasibility of improvement in the OpenEdX framework (Ficus Release, 2016-17) and subsequently, the entire system of IITBombayX, using MySQL and MongoDB along with the RocksDB engine developed at Facebook Inc.

1.2 Document Conventions

In general this document prioritizes in writing the requirements of the system and analyzing in details the tools being provided to its users. Every requirement is having its own priority (non conflicting). In addition few figures are also being provided to make requirements more clear to the reader.

1.3 Intended Audience and Reading Suggestions

This document is intended for any individual user, developer, tester, project manager or document writer that needs to understand the basic system architecture and its specifications. The potential use for each reader is elucidated as follows:

Developer: The developer who wants to read, modify or add new requirements into the existing program may need first to consult this document and update the requirements in an appropriate manner so as not to change the actual purpose of the system or make the system inconsistent with the actual requirement and can be successfully passed to next phase of the development.

User: The user of this program reviews the diagram and the specification provided in the document and check to determine whether the software has all the suitable requirements and if the software developer has the implemented all of them.

Tester: The tester needs this document to validate that the initial requirements of this programs actually corresponds to correctly executable program.

1.4 Product Scope

The OpenEdX system or platform provides an easy and effective way to build MOOC's for students and others to opt for free online courses. The benefits of this is that the traditional learning methods (like classrooms with black boards) are upgraded with online interactive support using computers as a tool to provide all kind of courses in interactive manner.
It also provides an interactive platform for the author to make/design a high quality online course with different kinds of grading systems and other facilities.

Minimizing space amplification is important to efficient hardware use because storage space is the bottleneck in production environments. SSDs process far fewer reads/s and writes/s during peak times under InnoDB than what the hardware is capable of. The throughput level under InnoDB is low, not because of any bottleneck on the SSD or the processing node — e.g., CPU utilization remains below 40% — but because the query rate per node is low. RocksDB provides an efficient solution to this issue, under production conditions. As a result it may be possible to use the same engine enabled database system for the OpenEdX framework, thus enabling more efficient usage of disk space and faster execution time.

1.5 References

The reader can refer the following sites to acquire the basic understanding of the interface:

1. http://cidrdb.org/cidr2017/papers/p82-dong-cidr17.pdf (Paper on RocksDB and benchmark tests)
2. https://github.com/facebook/mysql-5.6/wiki/MyRocks-advantages-over-InnoDB
3. https://code.facebook.com/posts/190251048047090/myrocks-a-space-and-write-optimized-mysql-database/

2. Case Study

2.1 Current perspective and need for efficient systems.

- OpenEdX currently uses MySQL to store majority of the coursework details and MongoDB to store the user details.
- Both of these Databases are run with the InnoDB engine at their core. The space amplification caused by InnoDB causes a lot of storage space (around 40 ~50%) of a Hard Drive to get wasted, mainly due to page fragmentation
- Compressed InnoDB provides static page compression thus, further wasting space.

2.2 Why use RocksDB?

- RocksDB scales to run on servers with many CPU cores.

- RocksDB makes efficient use of storage,more Input Output operations Per Second (*IOPS*),improved compression and less write wear.
- RocksDB has a flexible architecture to allow for innovation.
- RocksDB supports IO-bound,in-memory,and write-once workloads.

2.3 Benchmark Performance Review

- Based on experimental evaluations of MySQL with RocksDB as the embedded storage engine (using TPC-C and LinkBench benchmarks) and based on measurements taken from production databases, we show that RocksDB uses less than half the storage that InnoDB uses, yet performs well and in many cases even better than the B-tree-based InnoDB storage engine.
- This can be seen in the fact that, outside of Facebook,both MongoDB and Sherpa,Yahoo's largest distributed data source use RocksDB. Further,RocksDB is used by LinkedIn and by Netflix.
- In a typical production MySQL environment at Facebook, SSDs process far fewer reads/s and writes/s during peak times under InnoDB than what the hardware is capable of.

3. Parameters of Improvement over InnoDB

RocksDB is based on Log-Structured Merge-Trees (LSM-trees). The LSM-tree was originally designed to minimize random writes to storage as it never modifies data in place,but only appends data to files located in stable storage node is low.

3.1 Need for Log-Structured Merge trees.

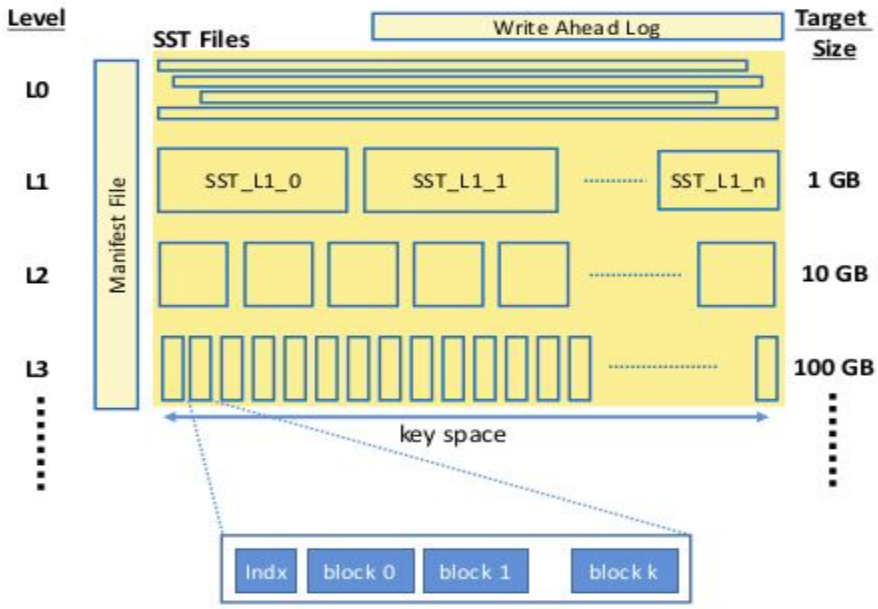
Whenever data is written to the LSM-tree, it is added to an in-memory write buffer called *mem-table*, implemented as a skiplist having O(log n) insertions and searches. At the sametime, the data is appended to a Write Ahead Log (WAL) for recovery purposes.

3.1.1 Reasons for better throughput

In the general form described in the "differential files" paper—when writing is expensive,instead of trying to immediately write,defer into a "differential layer".Reads consult both layers.Periodically layers are merged efficiently,to make sure there are not too many layers—because each layer consulted decreases read performance.This is the conceptual basis of all sorted map write-deferral strategies.

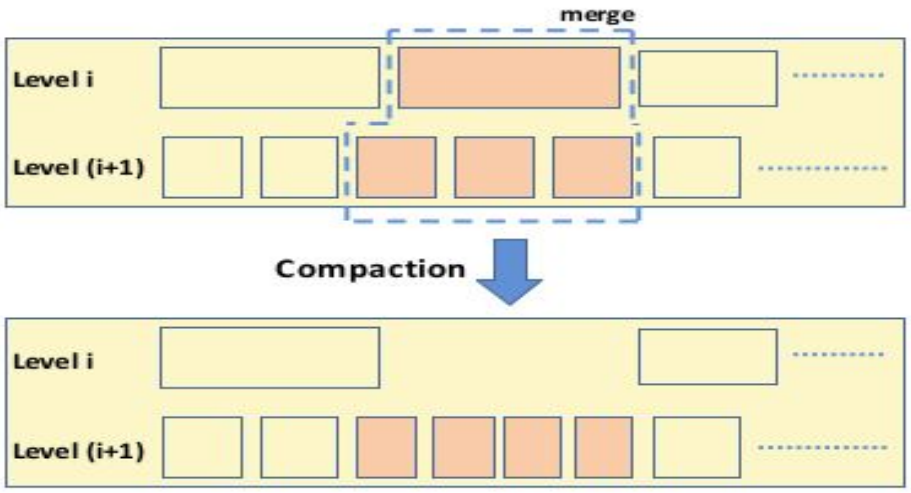
After a write, if the size of the mem-table reaches a predetermined size, then (i) the current WAL and mem-table become immutable, and a new WAL and mem-able are allocated for capturing subsequent writes,

(ii) the contents of the mem-table are flushed out to a "Sorted Sequence Table" (SST) data file, and upon completion, (iii) the WAL and mem-table containing the data just flushed are discarded.



SST file organization. Each level maintains a set of SST files. Each SST file consists of unaligned 16KB locks with an index block identifying the other blocks within the SST. Level 0 is treated differently in that its SST files have overlapping key ranges, while the SST files at the other level have non-overlapping key ranges. A manifest file maintains a list of all SST files and their key ranges to assist lookups.

SSTs are organized into a sequence of levels of increasing size, Level-0 – Level-N,where each level will have multiple SSTs.When the number of files in Level-0 exceeds a threshold (e.g., 4), then the Level-0 SSTs are merged with the Level-1 SSTs that have overlapping key ranges; when completed, all of the merge sort input (L0 and L1) files are deleted and replaced by new (merged) L1 files.The merging process is called **compaction** because it removes data marked as deleted and data that has been over-written (if it is no longer needed).



Compaction. The contents of a selected level-i SST file is merged with those SST files at level i+1 that have key ranges overlapping with the key range of the level-i SST. The shaded SST files in the top part of the figure are deleted after the merge process. The shaded SST files in the bottom part of the figure are new files created from the compaction process. The compaction process removes data that has become obsolete; i.e.,data that has been marked as deleted and data that has been overwritten (if they are no longer needed for snapshots).

A single Manifest File maintains a list of SSTs at each level, their corresponding key ranges, and some other metadata. It is maintained as a log to which changes to the SST information are appended.The search for a key occurs at each successive level until the key is found or it is determined that the key is not present in the last level.At each of these successive levels,three binary searches are necessary. The first search locates the target SST by using the data in the Manifest File. The second search locates the target data block within the SST file by using the SST's index block. The final search looks for the key within the data block.

3.2 Space Amplification

Space amplification is the ratio of the size of the database to the size of the data in the database.Compression decreases space-amp.It is increased by fragmentation with a B-Tree and old versions of rows with an LSM.

RocksDB uses two strategies to reduce space amplification: (i) adapting the level sizes to the size of the data, and(ii) applying a number of compression strategies.

3.2.1 Dynamic level size adaptation

If we dynamically adjust the size of each level to be 1/10-th the size of the data on the next level, then space amplification will be reduced to less than 1.111....The level size multiplier is a tunable parameter within an LSM-tree. Above, we assumed it is 10. The larger the size multiplier is, the lower the space amplification the and read amplification, but the higher the write amplification.

3.2.2 Compression

The compression strategies used by RocksDB are mentioned in brief:

Key prefix encoding: Prefix encoding is applied on keys by not writing repeated prefixes of previous keys. We have found this reduces space requirements by 3% – 17% in practice, depending on the data workload.

Sequence ID garbage collection: The sequence ID of a key is removed if it is older than the oldest snapshot needed for multiversion concurrency control.In practice, this optimization reduces space requirements from between 0.03% and 23%.

Tiered compression: Compression generally decreases the amount of storage space required, but increases CPU overheads, since data has to be compressed and decompressed. The stronger the compression, the higher the CPU overhead.In various use cases, applying strong compression to the last level saves an additional 15%–30% in storage space over using lightweight compression only.

Bloom filters: Bloom filters are effective in reducing I/O operations and attendant CPU overheads, but at the cost of somewhat increased memory usage since the filter (typically) requires 10 bits per key.The last-level bloom filter is large (~9X as large as all lower-level Bloom filters combined) and the space it would consume in the memory-based caches would prevent the caching of other data that would be being accessed.

Prefix Bloom filters: Bloom filters do not help with range queries.A prefix Bloom filter was developed that helps with range queries, based on the observation that many range queries are often over a prefix; e.g., the userid part of a (userid,timestamp) key or postid of a (postid,likerid) key. We allow users to define prefix extractors to deterministically extract a prefix part of the key from which we construct a Bloom filter.

Data compression: RocksDB currently supports several compression algorithms, including LZ, Snappy, zlib, and Zstandard. Each level can be configured to use any or none of these compression algorithms. Compression is applied on a per-block basis. Depending on the composition of the data, weaker compression algorithms can reduce space requirements down to as low as 40%, and stronger algorithms down to as low as 25%, of their original sizes on production Facebook data.

Dictionary-Based Compression: A data dictionary can be used to further improve compression. Data dictionaries can be particularly important when small data blocks are used, as smaller blocks typically yield lower compression ratios. The dictionary makes it possible for smaller blocks to benefit from more context. Experimentally, we have found that a data dictionary can reduce space requirements by an additional 3%.

4. Impact on Project

Our main focus is the improvement in the database of the OpenEdX platform .It may be possible to reduce the storage space required by an installation by efficiently integrating RocksDB into the core of the current system. .

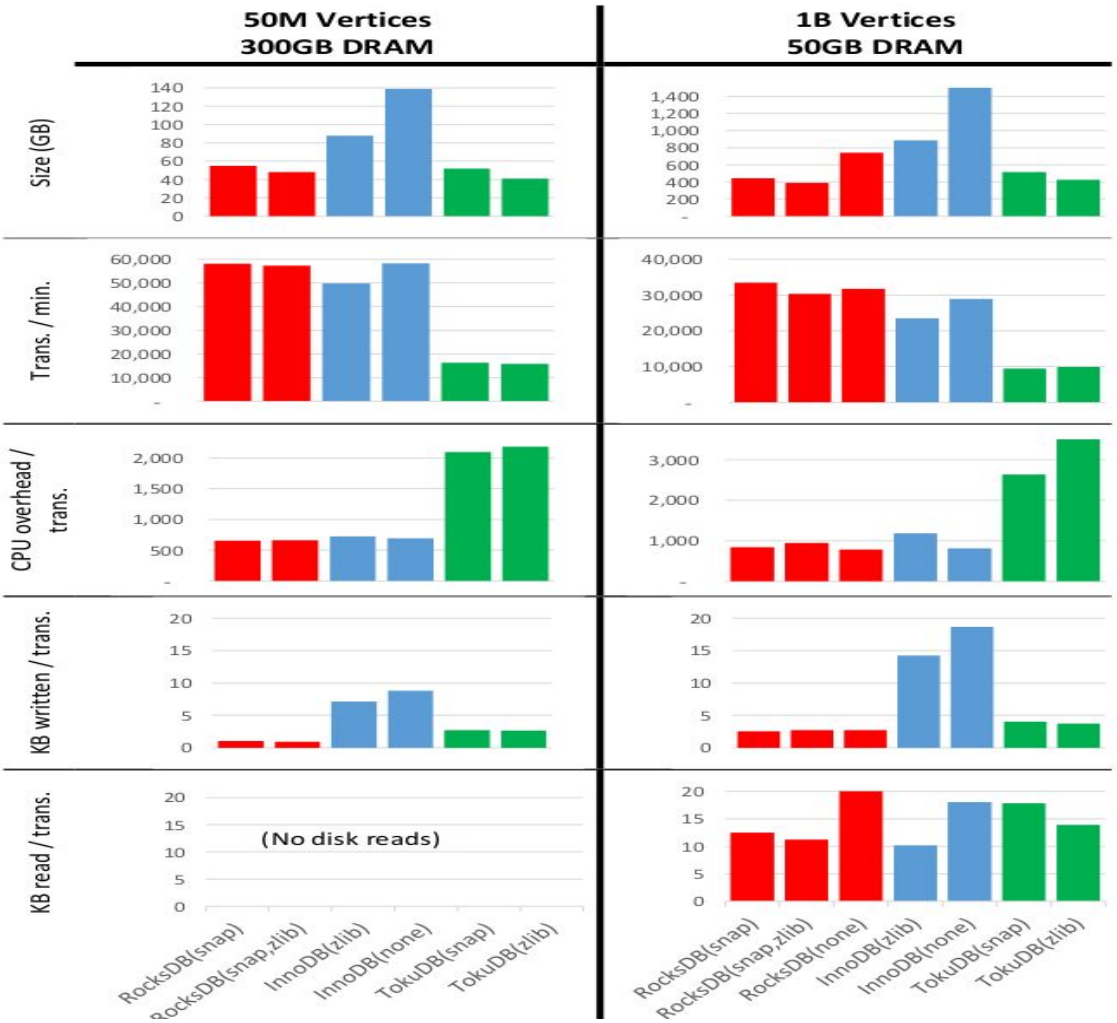
The performance analysis of RocksDB in Facebook's benchmark tests gave the following results (Benchmark tests results attached in Appendix A)

- **Space usage:** RocksDB with compression uses less storage space than any of the alternatives considered; without compression, it uses less than half as much storage space as InnoDB without compression.
- **Transaction throughput**: RocksDB exhibits higher throughput than all the alternatives considered: 3%- 16% better than InnoDB.
- **CPU overhead:** When stronger compression is used, RocksDB exhibits less than 20% higher CPU overhead per transaction compared to InnoDB with no compression
- **Write Volume:** The volume of data written per transaction in RocksDB is less than 20% of the volume of data written by InnoDB.
- **Read Volume:** The volume of data read per read transaction in RocksDB is 20% higher than InnoDB when no compression is used, and between 10% and 22% higher when compression is used.

The existing system of OpenEdX being based on InnoDB, may suffer from these drawbacks. RocksDB, if possible to be integrated in OpenEdX, shall make the platform lighter and improve performance efficiency.

5. Appendix

Benchmarking :The Benchmark tests were done with respect to RocksDB and InnoDB on two platforms, Link Benchmark and TPC-C Benchmarks.
The following graphs provide an idea with regards to the tests run.



Link-Bench Benchmark: *Statistics gathered from the 24th hour of 24 hour runs with 16 concurrent clients for 3 different storage engines: RocksDB (from Facebook MySQL 5.6)shown in red, InnoDB (from MySQL 5.7.10) shown in blue, and TokuDB (Percona Server 5.6.26-74.0) shown in green, configured to use the compression scheme(s) listed in brackets. (Sync-on-commit was disabled, binlog/oplog and redo logs were enabled.)*

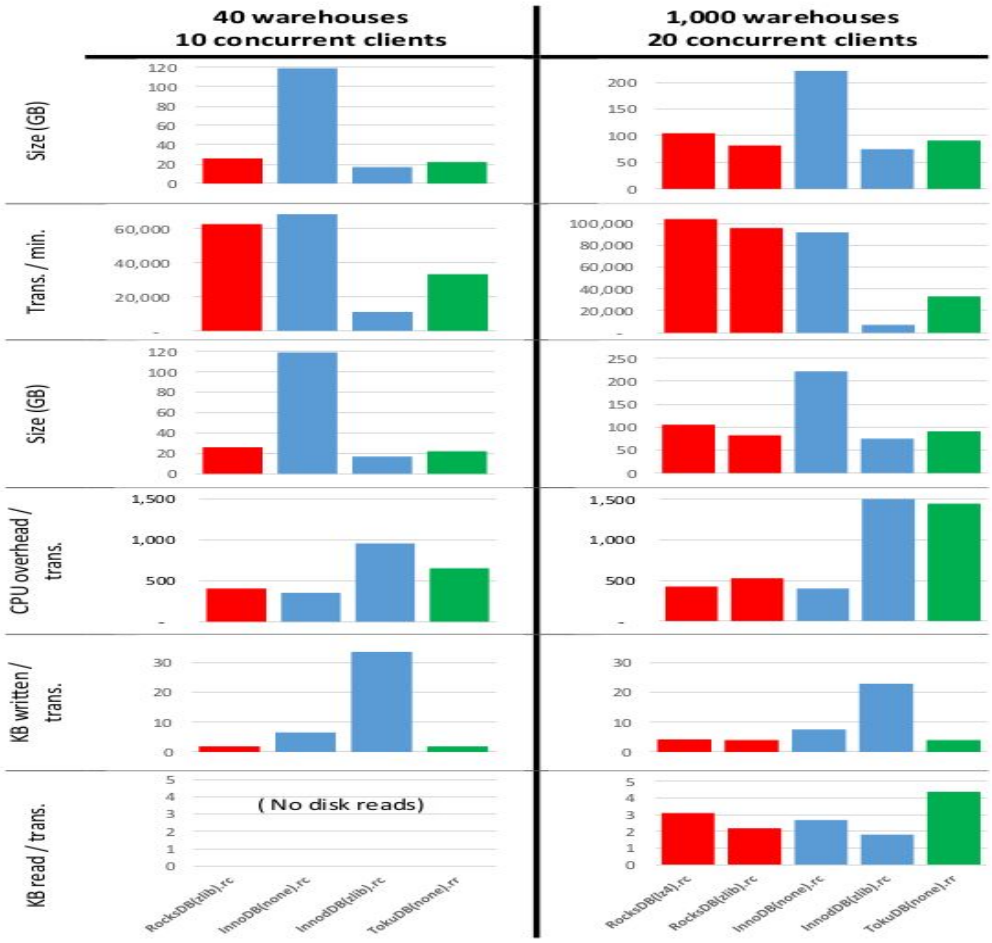
System setup: *The hardware consisted of an Intel Xeon E5-2678v3 CPU with 24-cores/48-HW-threads running at 2.50GHz,256GB of RAM, and roughly 5T of fast NVMe SSD provided via3 devices configured as SW RAID 0. The operating system was Linux 4.0.9-30*

Left hand side graphs: *Statistics from LinkBench configured to store 50M vertices, which fits entirely in DRAM.*

Right hand side graphs: *Statistics from LinkBench configured to store 1B vertices, which does not fit in memory after constraining DRAM memory to 50GB: all but 50GB of RAM was m locked by a background process so the database software, OS page cache and other monitoring processes had to share the 50GB. The MyRocks block cache was set to 10GB.*

The graphs depicts the quality of service achieved by the different storage engines. Specifically, it shows the 99th percentile latencies for read and write requests on both vertices and edges in the LinkBench database. The behavior of RocksDB is an order of magnitude better than the behaviour of all the other alternatives considered.

Comparative study between InnoDB and RocksDB by Facebook against TCP-P Benchmark



Left hand side: *Configuration of 40 warehouses and 10 concurrent clients. The database fits entirely in memory. The statistics were gathered over the entire 15th hour after 14 hours of operation.*
Right hand side: *Configuration of 1,000 warehouses and 20 concurrent clients. The statistics were gathered over the entire 12th hour after 11 hours of operation. The transaction isolation levels used are marked as "rc" for READ COMMITTED or "rr" for REPEATABLE READ.*

The figure clearly shows that RocksDB is not only competitive on OLTP workloads, but generally has higher transaction throughput while requiring significantly less storage space than the alternatives. RocksDB writes out less data per transaction than all the other configurations tested, yet reads only marginally more and requires only marginally more CPU overhead per transaction.