Evolution of Development Priorities in Key-value Stores Serving Large-scale Applications: The RocksDB Experience

FAST 2021

Siying Dong (Speaker), Andrew Kryczka, Yanqin Jin
Software Engineer, Facebook Inc

Michael Stumm
University of Toronto
Agenda

Design Goals of RocksDB

Evolution of Resource Optimization Targets

Lessons on Integrity Checking
Design Goals of RocksDB
RocksDB
Persistent Key-Value Store Library
RocksDB Design Goals

<table>
<thead>
<tr>
<th>For Modern Hardware</th>
<th>Versatile</th>
<th>Single Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSDs, Multicore CPUs and serve high throughput</td>
<td>Serve wide range of applications</td>
<td>Doesn’t handle replication, load balancing, etc</td>
</tr>
</tbody>
</table>
RocksDB Timeline

2012
- Fork from LevelDB
- Add Multithreaded Compactions

2013
- Open Source, Serious Production Use Cases
- Component Pluggability

2014
- Support Backups

2015
- Support Transactions
- Support Bulk Loading

2016
- MyRocks (MySQL engine) in production at Parse
  - Support DeleteRange()

2017
- Experimented caching on NVM

2018
- Rocksandra (Apache Cassandra engine) in production at Instagram
  - Start to optimize for disaggregated storage

2019
- Parallel I/O within MultiGet()

2020
- Partial Support of user-defined timestamps
Evolution of Resource Optimization Targets
First Optimization Goal: Write Amplification
Focusing on Write Amplification in Engine Layer

Often Hundreds
Write amplification by storage engine

Close to 1
Write amplification by file systems

Between 1 and 3
SSD’s internal write amplification

Source:
Focusing on Write Amplification in Engine Layer

Often Hundreds
Write amplification by storage engine

Close to 1
Write amplification by file systems

Between 1 and 3
SSD’s internal write amplification

Source:
Data Structure:
Log-Structured Merge-Tree
Lesson

Space efficiency is the bottleneck for most applications using SSDs.
Resource Utilization for ZippyDB and MyRocks Use Cases

Flash endurance vs. CPU

Read bandwidth vs. Space
Good News

LSM-Tree is good for space efficiency too (with some optimizations)
Lesson

Reducing CPU Overhead is becoming more important.
CPU/Space Balance can often be shifted through hardware configuration
But it’s hard to balance CPU and SSD.

It’s time to focus on disaggregated Storage.
It’s hard to balance CPU and SSD
RocksDB on Disaggregated Storage
Recap Performance Optimization Targets

• Initial Optimization Targets are Write Amplification, and it is indeed important.

• Majority of use cases are bounded by SSD space.

• Reducing CPU overheads is becoming more important for efficiency

• Now working on disaggregated storage to achieve balanced CPU and SSD usage.
Typical Way of Using RocksDB

Replication through Paxos

Host 1

Host 2

Host 3

DB

DB

DB
Typical Way of Using RocksDB

Replication through Paxos

Host 1 → DB

Host 2 → DB

Host 3 → DB

Copy files to build a new replica
Typical Way of Using RocksDB

Replication through Paxos
Typical Way of Using RocksDB

Host 1

Host 2

Host 3

Host 4

Host 5

Backup Storage

backup

restore
Typical Way of Using RocksDB

Host 1

Host 2

Host 3

Host 4

Host 5

Backup Storage

backup

restore
Typical Way of Using RocksDB

Replication through Paxos

Copy files to build a new replica

Backup Storage

Host 1

Host 2

Host 3

Host 4

Host 5

backup

restore
Typical Way of Using RocksDB

Replication through Paxos

Host 1

Host 2

Host 3

Host 4

Host 5

Copy files to build a new replica

Paxos

backup

restore

Backup Storage

Read the paper for lessons on large-scale systems
Minimum Integrity Check: Block Checksum

RocksDB

File System and Storage Device

Write block checksum when writing a block

Check block checksum when reading a block
Lesson

It is beneficial to detect data corruptions earlier, rather than eventually.
Detectable corruptions can be corrected by copying a healthy replica

Replication through Paxos

Host 1

Host 2

Host 3

Host 4

Corrupted

Copy a new replica
Delayed corruption detection risks data loss
Unchecked Data Copying Risk Data Loss

Copy a new replica for load balance
Unchecked Data Copying Risk Data Loss

Host 1

Host 2

Host 3

Host 4

Copy a new replica for load balance

corrupted
Unchecked Data Copying Risk Data Loss

Copy a new replica for load balance

Copy another new replica for load balance
Unchecked Data Copying Risk Data Loss

Host 1

Host 4

Host 5

DB

corrupted

DB

corrupted

DB

corrupted
Keep Full SST File Checksums

- **Host 1**: DB (corrupted)
- **Host 2**: DB
- **Host 3**: DB
- **Host 4**: DB

SST Files + their checksums

File checksum mismatch
Reject
Keep Full SST File Checksums

Backup Storage

Check full file checksums

backup

restore

DB

Check full file checksums
Lesson

CPU corruption does happen, though very rarely, and sometimes cannot be handled by data replication.
CPU corruption is silent and can cause data loss

- Host 1: DB\n  - Corrupted by CPU/memory

- Host 2: DB
  - Silent corruption

- Host 3: DB
  - Silent corruption

- Host 4: DB
  - Silent corruption

- Host 5: DB
  - Silent corruption

Copy a new replica for load balance

Copy another new replica for load balance
Use per K/V Checksum to Mitigate CPU corruption

Client

[Write]
Pass and check K/V Checksums

Memtables

Per K/V checksums

[Read]
Check K/V Checksums

[SSTable]

[Memtable Flush] Check K/V checksums

[Compaction] check K/V checksums

SSTable

Block checksums

[Read]
Check K/V Checksums

Block Cache

K/V checksums

Block checksums

SSTable
CPU Corruption can cause handoff issues

- RocksDB
  - WAL Log Entry + checksum
    - Bitflip when copying the buffer
  - Bytes + Checksums
    - Reliable Storage System
CPU Corruption can cause handoff issues

Handoff checksum can make the protection end-to-end
Recap Lessons on Integrity Checking

- RocksDB users handle integrity errors by tossing back replica and building a new one, so delayed corruption detection risks data loss.

- Full file checksums can help prevent corrupted copy from spreading.

- CPU/memory corruption does happen, although very rarely.

- Per KV checksum and handoff checksums could mitigate damage of CPU/memory corruption.
More in the paper

- More Production Data
- Thoughts in Newer Storage Technologies
- Lessons on serving large-scale systems
  - Resource management
  - WAL treatment
  - Rate-limit file deletions
  - Data format compatibility
  - Managing configurations
  - Replication and Backup Supports
- More lessons on failure handling
- Lessons on key-value interface
Thank you!
Questions?