

2025 Winter RocksDB Study

# 2025 Winter RocksDB Study

## 2<sup>nd</sup> week

**Hojin Shin, Guangxun Zhao**

<http://sslab.dankook.ac.kr/>, <https://sslab.dankook.ac.kr/~choijm>

Presentation by Hojin Shin  
hojin03s@dankook.ac.kr

# Contents

1. How to Analyze RocksDB
2. Research Topic (recommended)
3. Future Schedule
4. QnA

# How to analyze RocksDB

## ■ Recommended Approaches

- ✓ 1) Read and write researches
  - Documents, Lectures, Papers (Top-tier conference)
  - Conference: EuroSys, OSDI, FAST, ATC, VLDB, SIGMOD ...
- ✓ 2) Remarks and Code
  - VS code, Terminal ...
- ✓ 3) Code Tracing
  - GDB, Uftrace
- ✓ 4) Draw figures
  - Structure, Class, Code flow with Draw.io and PPT
- ✓ 5) Write a markdown document
  - Using github
- ✓ 6) Draw a evaluation figures
  - Python, Excel
- ✓ 7) Prepare 15-minute presentation

## Key-Value Store 분석 (RocksDB 중점)

*Dankook Univ. Embedded Lab.*

*Shin Hojin*

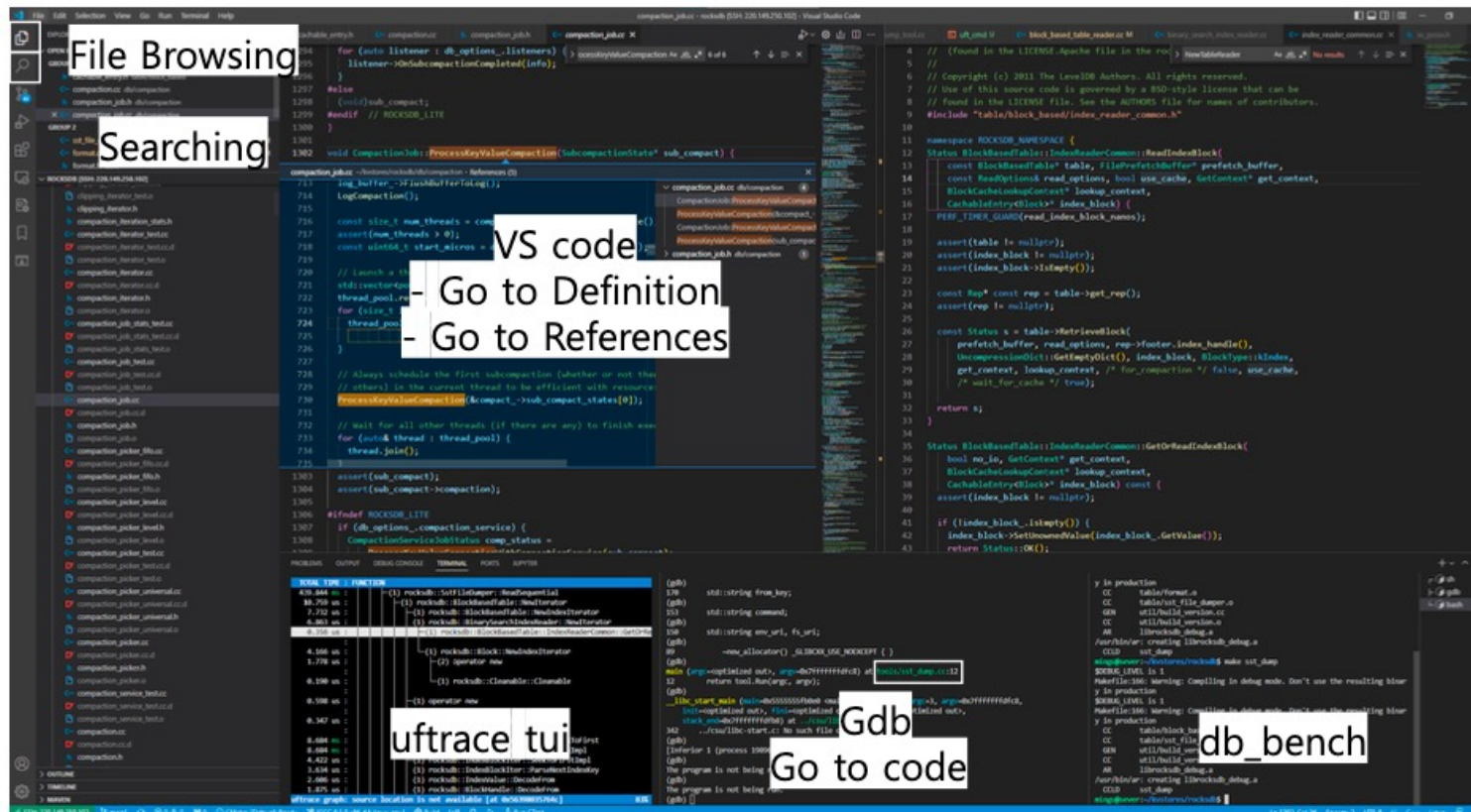


## 목차

1. Overview .....	5
1.1 Introduction .....	5
1.2 High-level Architecture .....	5
1.3 Features .....	5
2. RocksDB 디렉토리 구조 (주요 파일만 표시) .....	9
2.1 CACHE .....	9
2.2 DB .....	9
2.3 ENV .....	10
2.4 FILE .....	10
2.5 MEMTABLE .....	10
2.6 TABLE .....	10
2.7 UTIL .....	11
3. RocksDB basic operation .....	11
3.1 Opening a database .....	11
3.2 Status (error control) .....	11
3.3 Closing a database .....	11
3.4 Reads .....	11
3.5 Writes .....	11
3.6 Concurrency .....	12
3.7 Column family - DB 를 논리적으로 분리하는 방법 .....	12
3.8 Iterator .....	12
3.9 Prefix Seek .....	13

# Analyze Tool Example (by min-guk)

- VS code



# Analyze Tool Example (by min-guk)

## ■ GDB

### ✓ Run

- \$ gdb <program>
- \$ gdb --args <program> <arg1> <arg2> ...

### ✓ Process

- > r > run
- > c > continue
- > n > next
- > s > step
- > fin > finish

```
mingu@server:~/leveldb_release/build$ gdb --args ./db_bench --benchmarks="fillrandom"
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./db_bench...
(gdb) b db_impl.cc:894
Breakpoint 1 at 0x12213: file /home/mingu/leveldb_release/db/db_impl.cc line 894.
(gdb) i b
Num Type Disp Enb Address What
1 breakpoint keep y 0x00000000000012213 in leveldb::DBImpl::DoCompactionWork(leveldb::DBImpl::CompactionState*)
at /home/mingu/leveldb_release/db/db_impl.cc:894
(gdb)
```

Open file in Editor (Ctrl + Click)

```
1013 input->Next();
(gdb) s
leveldb::(anonymous namespace)::MergingIterator::Next (this=
0x555555590fee <leveldb::(anonymous namespace)::TwoLevelIterator::value() const+96>)
at /home/mingu/leveldb_release/table/merger.cc:55
55 void Next() override {
(gdb) n
56 assert(Valid());
(gdb)
63 if (direction_ != kForward) {
(gdb)
77 current_->Next();
(gdb)
78 FindSmallest();
(gdb) s
leveldb::(anonymous namespace)::MergingIterator::FindSmallest (
this=0x555555590f8c <leveldb::(anonymous namespace)::TwoLevelIterator::key() const+96>)
at /home/mingu/leveldb_release/table/merger.cc:148
148 void MergingIterator::FindSmallest() {
(gdb)
```

# Analyze Tool Example (by min-guk)

## Uftrace

### ✓ Record

- Run a program and saves the trace data

### ✓ Replay

- Show program execution in the trace data

### ✓ Graph/Tui

- Show function call graph in the trace data

### ✓ Filter

```
# DURATION    TID     FUNCTION
[ 5471] [ 5471] leveldb::DBImpl::Write() {
[ 5471] [ 5471]   leveldb::DBImpl::Writer::Writer() {
[ 5471] [ 5471]     leveldb::Status::Status();
[ 5471] [ 5471]     leveldb::port::CondVar::CondVar() {
[ 5471] [ 5471]       std::condition_variable::condition_variable();
[ 5471] [ 5471]     } /* leveldb::port::CondVar::CondVar */
[ 5471] [ 5471]   } /* leveldb::DBImpl::Writer::Writer */
[ 5471] [ 5471]   leveldb::MutexLock::MutexLock() {
[ 5471] [ 5471]     leveldb::port::Mutex::Lock() {
[ 5471] [ 5471]       std::mutex::lock() {
[ 5471] [ 5471]         __pthread_mutex_lock() {
[ 5471] [ 5471]           __pthread_active_p();
[ 5471] [ 5471]           pthread_mutex_lock();
[ 5471] [ 5471]         } /* __pthread_mutex_lock */
[ 5471] [ 5471]       } /* std::mutex::lock */
[ 5471] [ 5471]     } /* leveldb::port::Mutex::Lock */
[ 5471] [ 5471]   } /* leveldb::MutexLock::MutexLock */
[ 5471] [ 5471]   std::deque::push_back() {
[ 5471] [ 5471]     std::move();
[ 5471] [ 5471]     std::deque::emplace_back() {
[ 5471] [ 5471]       std::forward();
[ 5471] [ 5471]       std::allocator_traits::construct() {
[ 5471] [ 5471]         std::forward();
```

No Filter

```
# DURATION    TID     FUNCTION
[ 14234] [ 14234] leveldb::DBImpl::Write() {
[ 14234] [ 14234]   leveldb::DBImpl::Writer::Writer() {
[ 14234] [ 14234]     leveldb::port::CondVar::CondVar() {
[ 14234] [ 14234]       } /* leveldb::DBImpl::Writer::Writer */
[ 14234] [ 14234]   } /* leveldb::DBImpl::Writer::Writer */
[ 14234] [ 14234]   leveldb::MutexLock::MutexLock();
[ 14234] [ 14234]   leveldb::DBImpl::MakeRoomForWrite() {
[ 14234] [ 14234]     leveldb::VersionSet::NumLevelFiles();
[ 14234] [ 14234]     leveldb::MemTable::ApproximateMemoryUsage();
[ 14234] [ 14234]   } /* leveldb::DBImpl::MakeRoomForWrite */
[ 14234] [ 14234]   leveldb::VersionSet::LastSequence();
[ 14234] [ 14234]   leveldb::DBImpl::BuildBatchGroup() {
[ 14234] [ 14234]     leveldb::WriteBatchInternal::ByteSize();
[ 14234] [ 14234]   } /* leveldb::DBImpl::BuildBatchGroup */
[ 14234] [ 14234]   leveldb::WriteBatchInternal::SetSequence();
[ 14234] [ 14234]   leveldb::WriteBatchInternal::Count();
[ 14234] [ 14234]   leveldb::WriteBatchInternal::Contents();
[ 14234] [ 14234]   leveldb::log::Writer::AddRecord() {
[ 14234] [ 14234]     leveldb::log::Writer::EmmitPhysicalRecord() {
[ 14234] [ 14234]       leveldb::GLOBAL_N_1::PosixWritableFile::Append();
[ 14234] [ 14234]     }
[ 14234] [ 14234]   }
[ 14234] [ 14234]   leveldb::GLOBAL_N_1::PosixWritableFile::Append();
```

Filter

```
----- Back-trace -----
5.188 s: (100000) leveldb::SkipList::Insert
5.188 s: (100000) leveldb::MemTable::Add
5.188 s: (100000) leveldb::GLOBAL_N_1::MemTableInserter::Put
5.188 s: (100000) leveldb::WriteBatch::Iterate
5.188 s: (100000) leveldb::WriteBatchInternal::InsertInto
5.188 s: (100000) leveldb::DBImpl::Write
5.188 s: (100000) leveldb::Benchmark::DoWrite
5.188 s: (100000) leveldb::Benchmark::WriteRandom
5.188 s: (100000) leveldb::Benchmark::ThreadBody

----- Call Graph -----
5.188 s: (100000) leveldb::SkipList::Insert
4.424 s: (100000) leveldb::SkipList::FindGreaterOrEqual
13.450 ms: (100000) leveldb::SkipList::GetMaxHeight
473.144 ms: (2495692) leveldb::SkipList::Node::Next
4.121 s: (2495692) leveldb::SkipList::KeyIsAfterNode
3.903 s: (2404364) leveldb::MemTable::KeyComparator::operator()
993.576 ms: (4888728) leveldb::GetLengthPrefixedSlice
2.544 s: (2404364) leveldb::InternalKeyComparator::Compare
492.141 ms: (2404364) leveldb::GLOBAL_N_1::ByteWiseComparatorImpl::Compare
25.035 ms: (100000) leveldb::SkipList::RandomHeight
16.364 ms: (133759) leveldb::Random::OneIn
4.255 ms: (133759) leveldb::Random::Next
11.891 ms: (100026) leveldb::SkipList::GetMaxHeight
30.790 ms: (100000) leveldb::SkipList::NewNode
5.588 ms: (100000) leveldb::Arena::AllocateAligned
1.455 ms: (890) leveldb::Arena::AllocateFallback
1.384 ms: (890) leveldb::Arena::AllocateNewBlock
```



# Research Topic (Recommended)

## ■ Research Topic

- ✓ 1) Write (flush, WAL and compaction)
  - GearDB (FAST'19), WOKV (ICCCBDA'18), Reducing WAF (ICDE'22)
- ✓ 2) Read (bloom filter and cache)
  - ElasticBF (FAST'19), BloomStore (MSST'12), Rosetta (SIGMOD'20)
- ✓ 3) Key-value store structure (Tiered Storage and PMem)
  - NovelLSM (FAST'18), MatrixKV (ATC'20), SpanDB (FAST'21), PRISM (ASPLOS'23)

## ■ Advanced Research Topic

- ✓ 1) Key-value separation
  - Wisckey (FAST'16), FenceKV (TPDS'22)
- ✓ 2) In-memory index optimization
  - JellyFish (Middleware'20), ListDB (OSDI'22), S3 (VLDB'19)
- ✓ 3) Learned Index
  - Bourbon (OSDI'20)

# Future Schedule

## ■ Study Schedule (4 weeks)

Week 3 (25.2.4)	<ul style="list-style-type: none"><li>• RocksDB Benchmark evaluation (Flush, Compaction, WAL, Bloom filter, Cache)<ul style="list-style-type: none"><li>• Analyze the reasons for performance degradation or improvement</li></ul></li></ul>
Week 4 (25.2.11)	<ul style="list-style-type: none"><li>• Analyze the code relevant to the topic chosen by each team</li><li>• Visualize the code using structure diagram and code flow</li></ul>
Week 5 (25.2.18)	<ul style="list-style-type: none"><li>• Document the results from the previous week's work (using Github wiki)<ul style="list-style-type: none"><li>• Propose optimization ideas based on the analysis</li></ul></li></ul>
Week 6 (25.2.25)	<ul style="list-style-type: none"><li>• Organize and refine areas that were incomplete in the documentation process</li><li>• Implement the proposed optimization ideas and present the experimental results</li></ul>



# 2025 Winter RocksDB Study

## 2<sup>nd</sup> week

**Hojin Shin, Guangxun Zhao**

<http://sslab.dankook.ac.kr/>, <https://sslab.dankook.ac.kr/~choijm>

# Thank You

## Q & A ?

Presentation by Hojin Shin  
hojin03s@dankook.ac.kr