LevelDB Study
Introduction 3

2022. 07. 19

Presented by Min-guk Choi

koreachoi96@gmail.com

Previous Homework

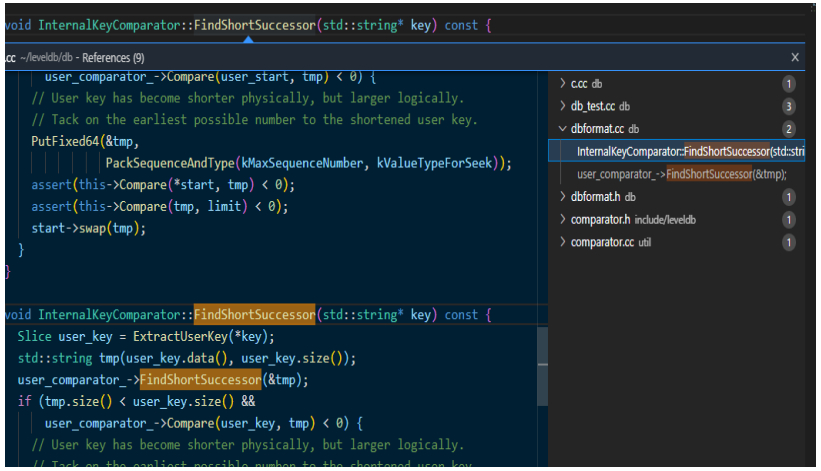
Google leveldb 멀티 레벨 구조

Google why leveldb multi level

The Log-Structured Merge-Tree (LSM-Tree)

Patrick O'Neil[1], Edward Cheng[2]
Dieter Gawlick[3], Elizabeth O'Neil[1]
To be published: Acta Informatica

ABSTRACT. High-performance transaction system applications typically insert rows in a History table to provide an activity trace; at the same time the transaction system generates log records for purposes of system recovery. Both types of generated information can benefit from efficient indexing. An example in a well-known setting is the TPC-A benchmark application, modified to support efficient queries on the History for account activity for specific accounts. This requires an index by account-id on the fast-growing History table. Unfortunately, standard disk-based index structures such as the B-tree will effectively double the I/O cost of the transaction to maintain an index such as this in real time, increasing the total system cost up to fifty percent. Clearly a method for maintaining a real-time index at low cost is desirable. The Log-Structured Merge-tree (LSM-tree) is a disk-based data structure designed to provide low-cost indexing for a file experiencing a high rate of record inserts (and deletes) over an extended period. The LSM-tree uses an algorithm that defers and batches index changes, cas-




```
mingu@sever:~/leveldb_release/build$ ./db_bench
LevelDB:    version 1.23
Date:       Fri Jul  8 21:38:00 2022
CPU:        16 * Intel(R) Core(TM) i7-10700K CPU @ 3.80GHz
CPUCache:   16384 KB
Keys:       16 bytes each
Values:     100 bytes each (50 bytes after compression)
Entries:    1000000
RawSize:    110.6 MB (estimated)
FileSize:   62.9 MB (estimated)
```

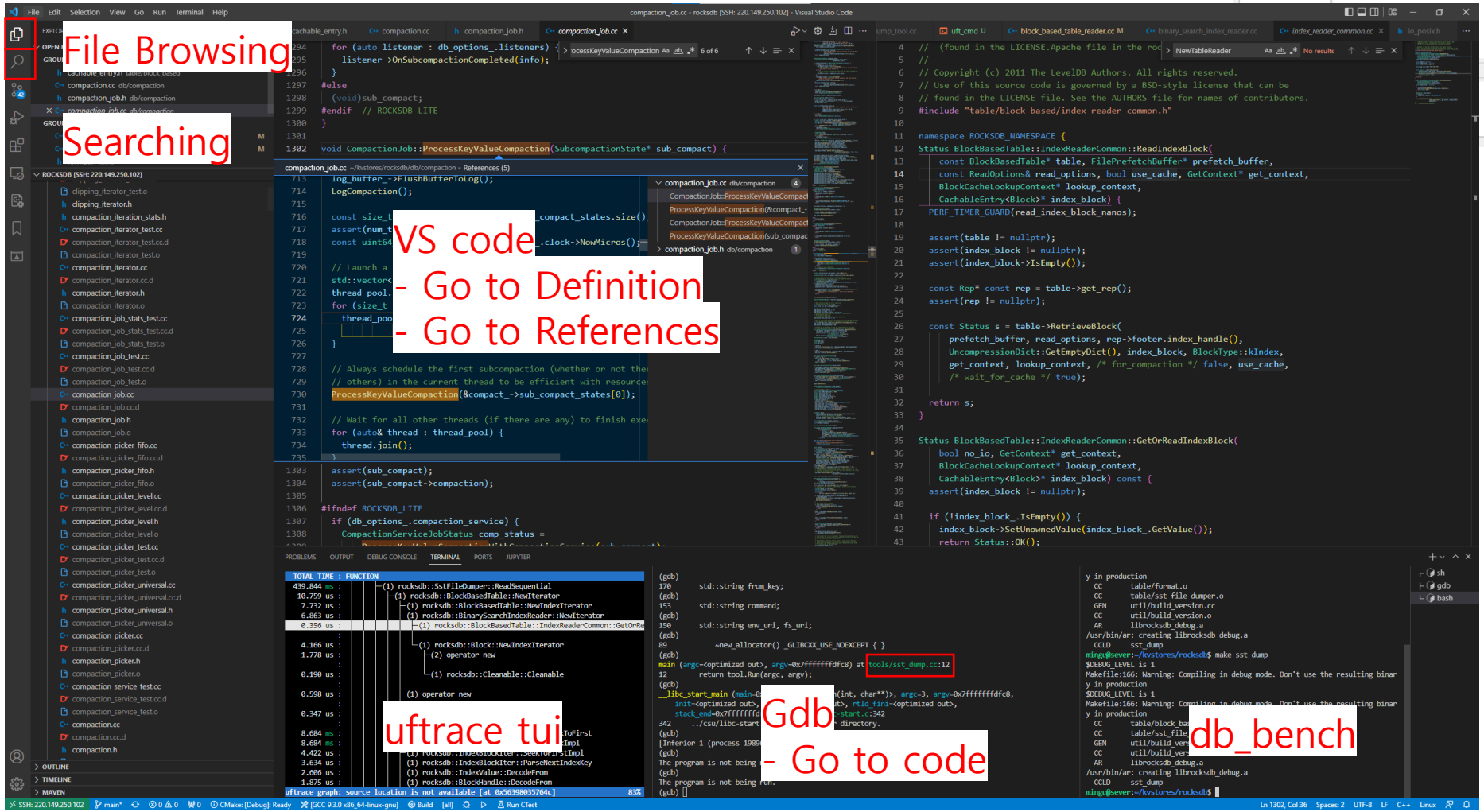
1. ***How to analyze LevelDB***
2. VS code
3. Understand
4. GDB
5. Uftrace

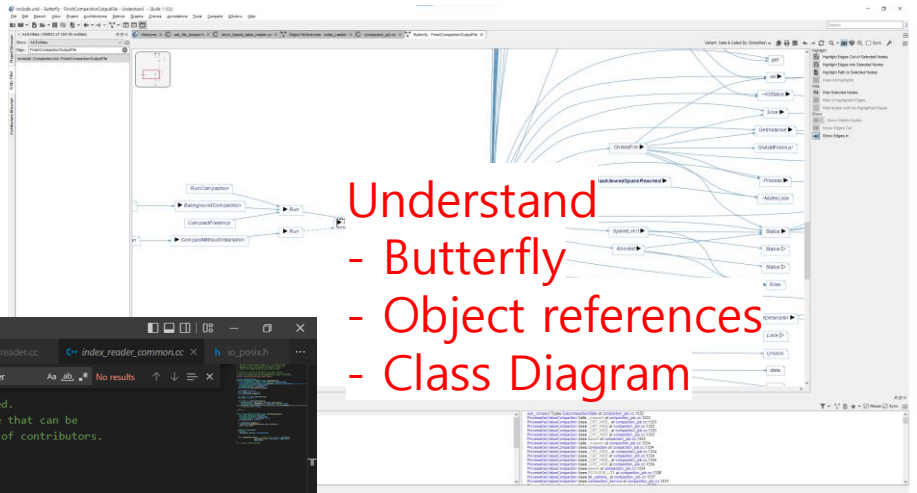
- How to analyze LevelDB
  - ✓ How to analyze LevelDB
  - ✓ How to use
  - ✓ What/When to use?
  - ✓ Where to start?
  - ✓ Source Code

How to analyze LevelDB

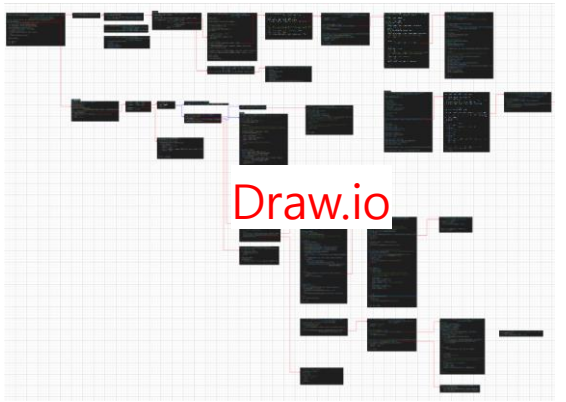
1. Do a research
   - Documents, lectures.
2. Remarks, Code
   - VS code, Understand
3. Code Tracing
   - GDB, Uftrace
4. Draw figures
   - Structure, class, flow
   - Draw.io, PPT
5. Write a markdown document
6. Prepare a 15-minute presentation

How I use

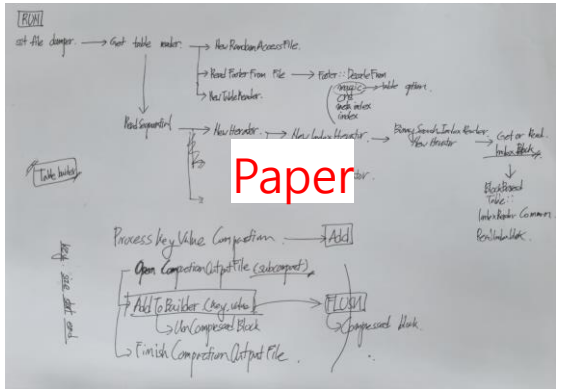




Understand
- Butterfly
- Object references
- Class Diagram



Draw.io



Paper

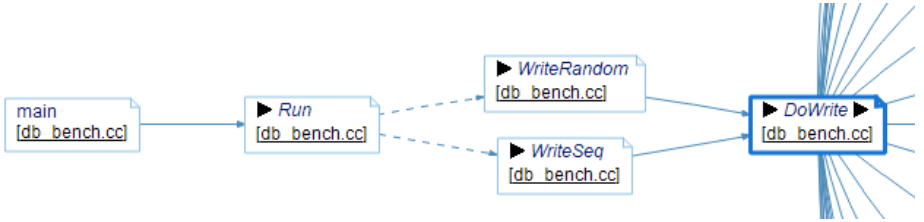
db_bench

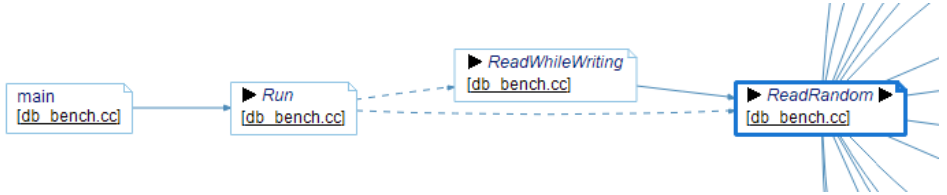
What/When to use?

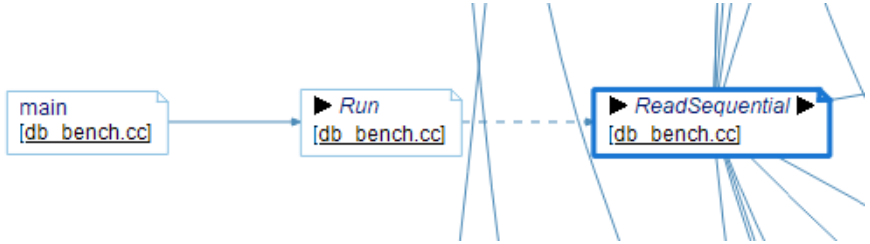
	1	2
Static Analysis Tool **- Remarks & Code**	VS code - Go to Definition - Go to References - Search - File Explorer	Understand - Class Diagram ~~- Object References~~ ~~- Butterfly~~
Dynamic Analysis Tool **- Code Flow/Tracing**	Uftrace - Record - Replay - Tracing	GDB - Break point - Step into / Next - Print / Display - Line by Line - Argument, Variable

Where to start?

- Start with basic operations
  - Put(Write) operations
    - db_bench –benchmarks=“fillseq, fillrandom”
    - db_bench.cc:DoWrite
  - Get(Read) operations
    - db_bench --benchmarks=“readrandom”
    - db_bench.cc:ReadRandom
  - Seek(Scan) operations
    - db_bench –benchmarks=“readseq”
    - db_bench.cc:ReadSequential







Source Code

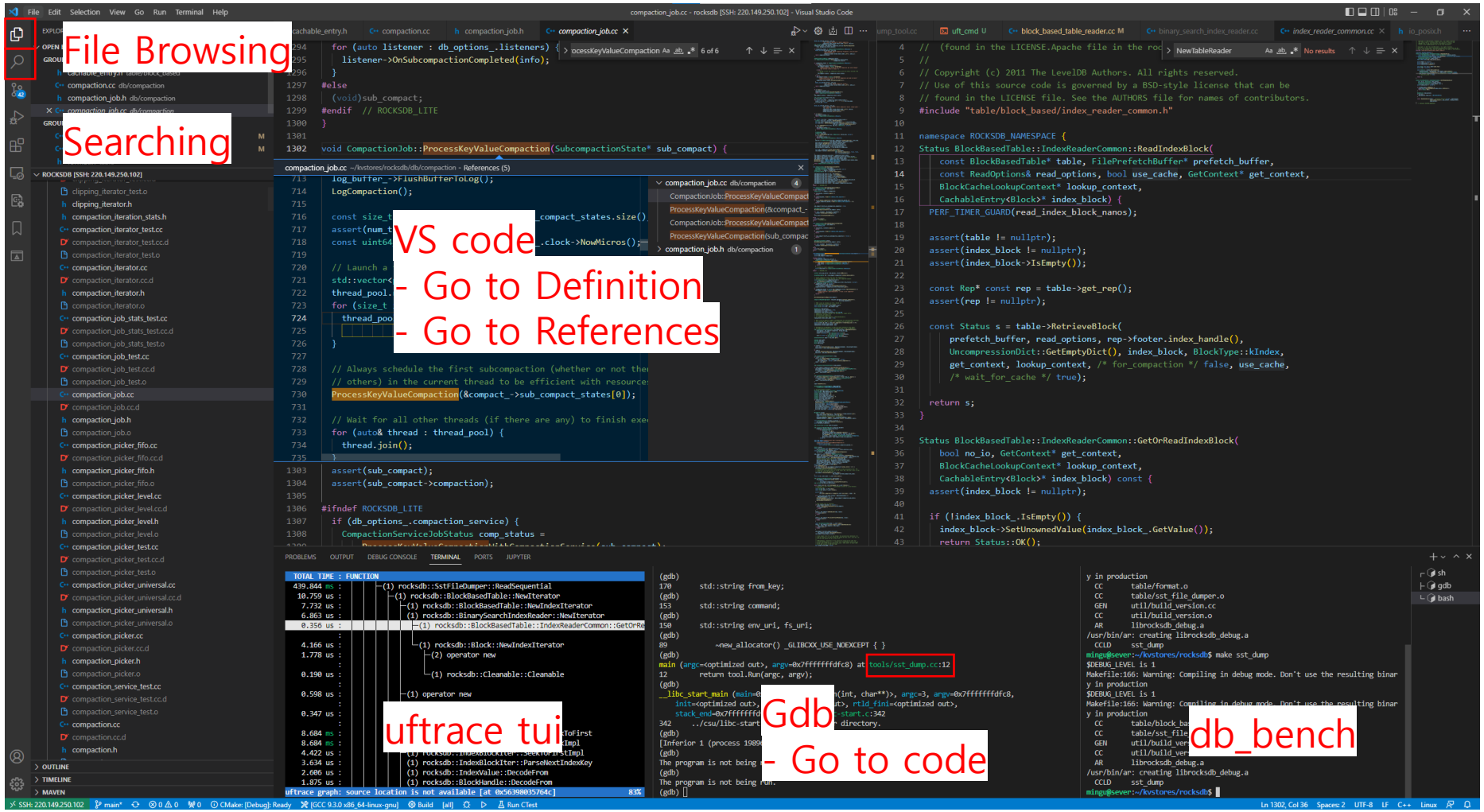
Subject	files
WAL/Manifest	version_set.h version_edit.h write_batich.cc db_impl.h Repair.cc
MemTable	Skiplist.h memtable.h db_impl.h Arena.h
Compaction	db_impl.h

Subject	files
SSTable	table/
Bloom Filter	c.cc Dbformat.cc filter_block.cc filter_block_test.cc filter_policy.h
Cache	cache.h table.cc table_cache.h hash.h db_impl.h

1. How to analyze LevelDB
2. ***VS code***
3. Understand
4. GDB
5. Uftrace

- VS Code
  - ✓ References

VS Code


File Browsing
Searching
VS code
- Go to Definition
- Go to References
uftrace tui
Gdb
- Go to code
db_bench

VS Code

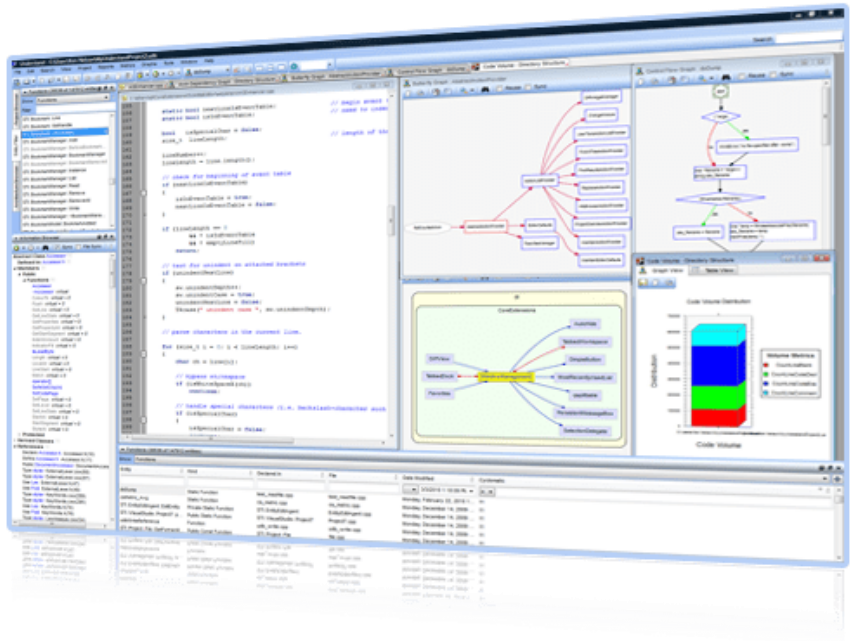
- References
  - 25 VS Code Productivity Tips and Speed Hacks
    - https://youtu.be/ifTF3ags0XI
  - 코딩시간을 절반으로 줄여주는 VSCode 9개 기능
    - https://youtu.be/mh-0twurNRE
  - 비쥬얼 스튜디오 코드 필수 단축키 정리 (Visual Studio Code 꿀템🍯🦋)
    - https://youtu.be/EVxCdenPbFs
  - Visual Studio Code 기본 사용법
    - https://youtu.be/K8qVH8V0VvY

1. How to analyze LevelDB
2. VS code
3. ***Understand***
4. GDB
5. Uftrace

- Understand
  - ✓ Introduction
  - ✓ Features
    - Class diagram
    - Object references
    - Control flow
    - Butterfly
  - ✓ Installation

Understand

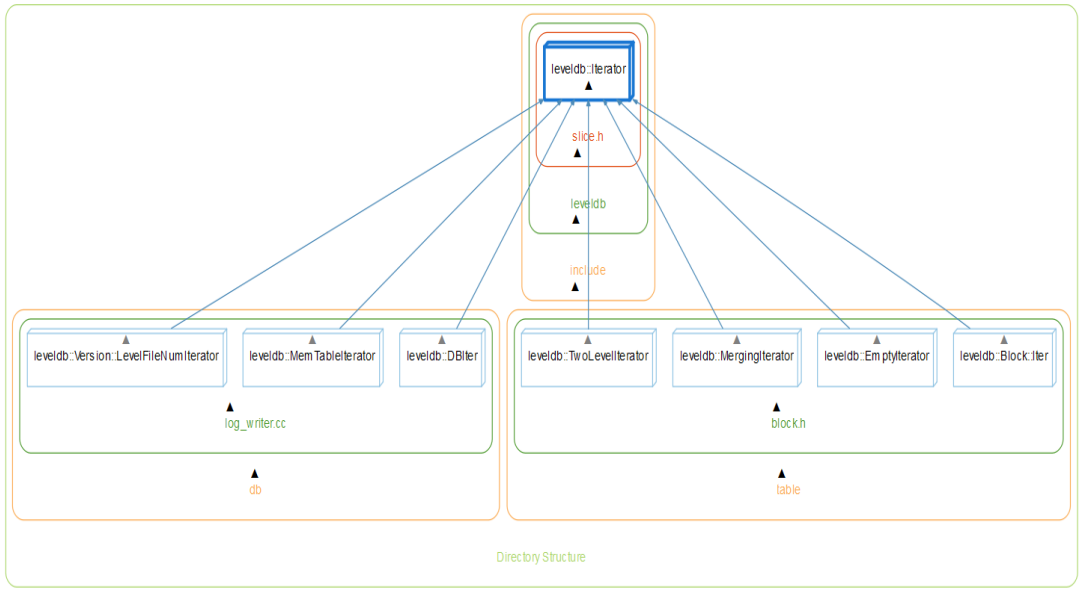
- Understand by SciTools
  - https://www.scitools.com/ (ENG)
  - https://www.slexn.com/understand/ (KOR)



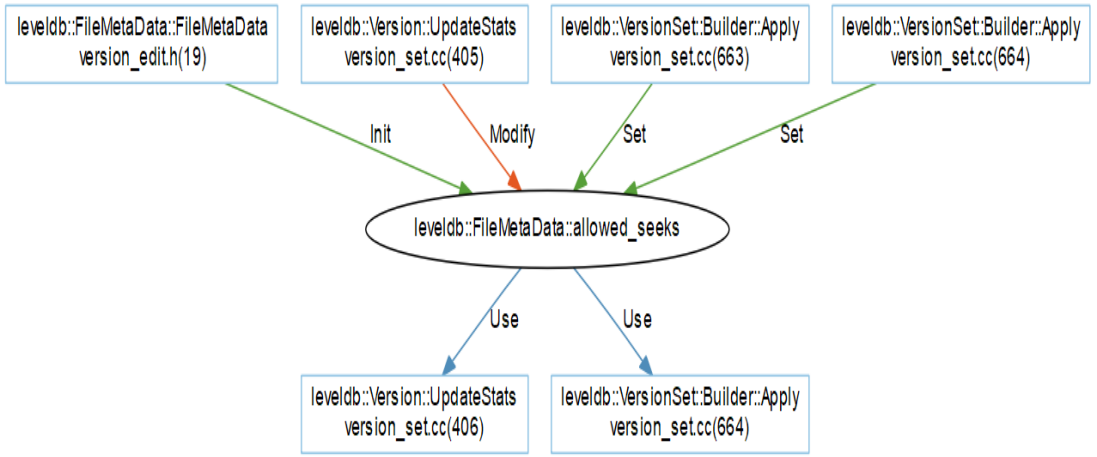
- Static analysis tool
  - understanding of complex open-source code
  - Graphing
    - code dependencies, code flows, function calls, and more

Understand

- LevelDB example



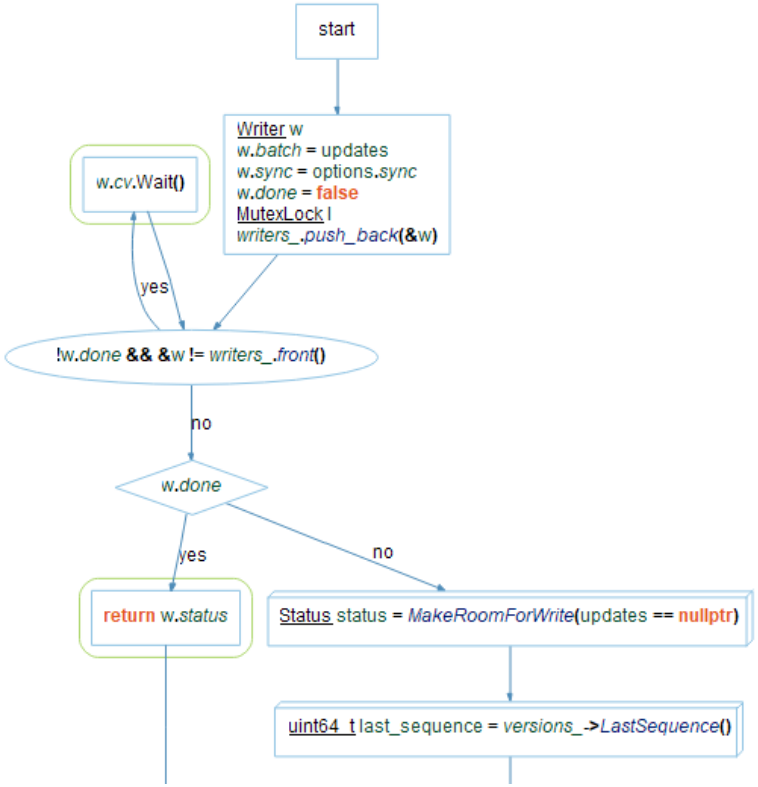
UML Class Diagram: Iterator



Object References

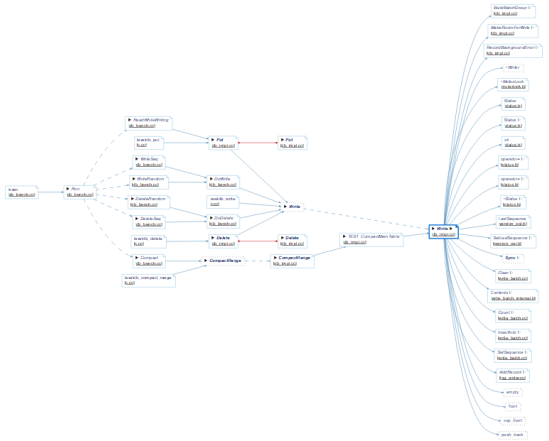
Understand

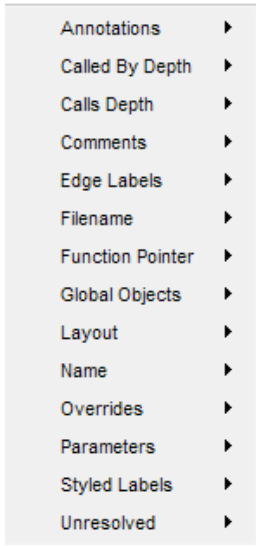
- LevelDB example
  - Control flow: db_impl.cc:Write
    - suggest to read code using VS code
    - Use “Go to Definition” (F12)

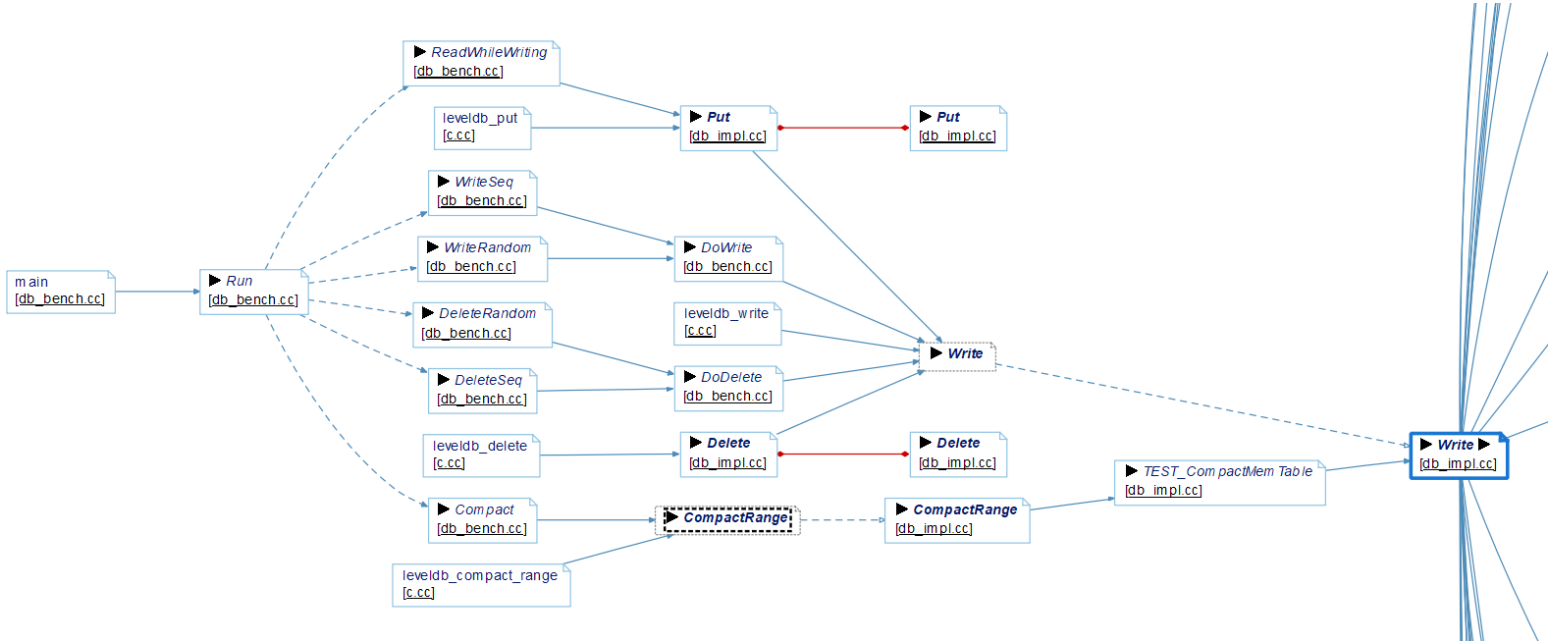


Understand

- LevelDB example
  - Butterfly: db_impl.cc:Get
    - Calls, called by

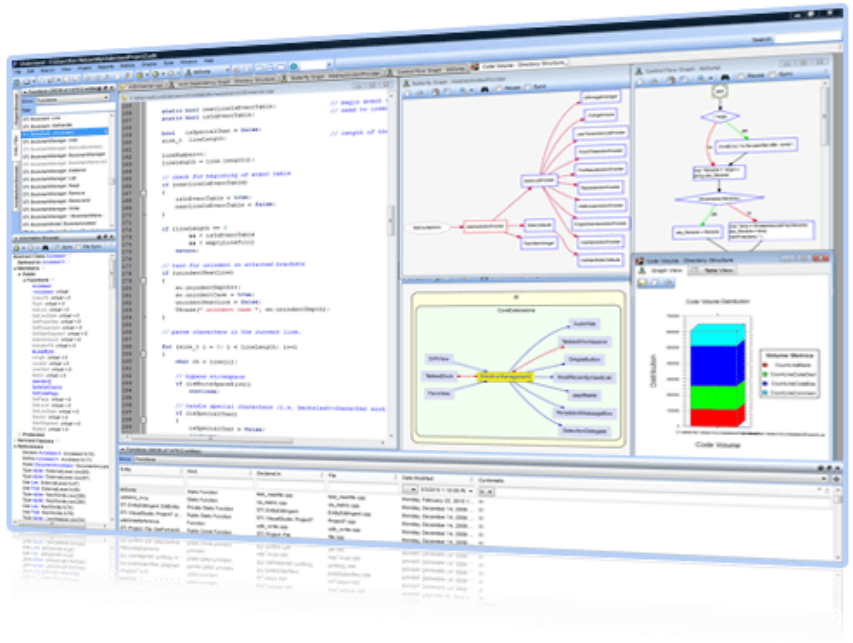






Understand

- How to install
  - Register with university email
    - https://licensing.scitools.com/register
  - Free trial link
    - https://www.scitools.com/pricing
  - Download link
    - https://licensing.scitools.com/download

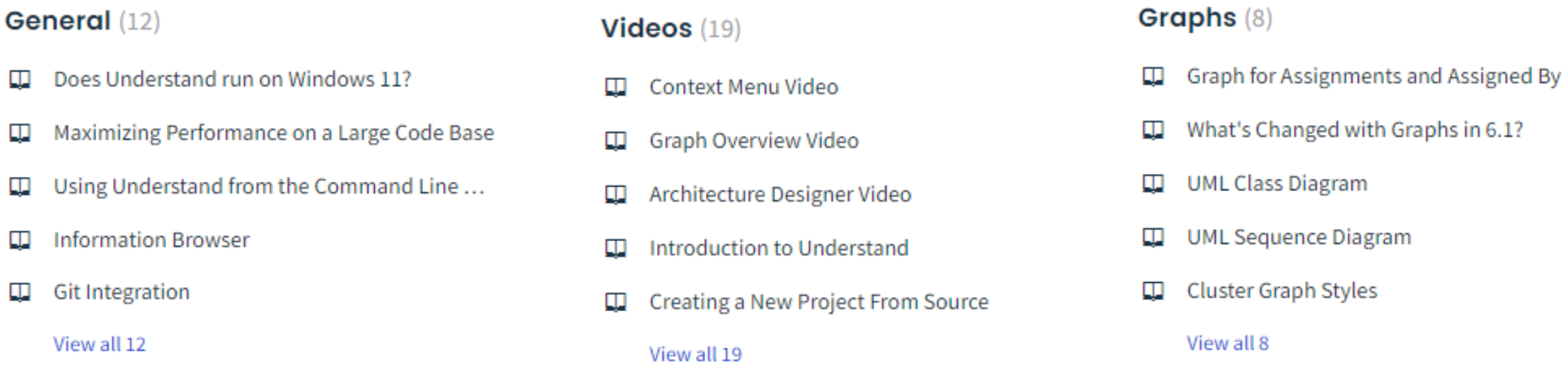


Free for Educational Use

Understand is free for students and teachers to use for educational purposes. Want to teach a class on code maintenance or need some metrics for your thesis? No problem, we got you covered. Learn More

Understand

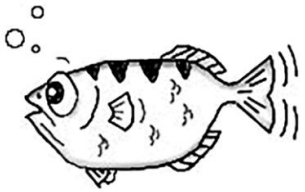
- How to use Understand?
  - https://support.scitools.com/support/solutions



1. How to analyze LevelDB
2. VS code
3. Understand
4. ***GDB***
5. Uftrace

- GDB
  - ✓ Introduction
  - ✓ Command
  - ✓ Example
    - Compaction Merge Iterator

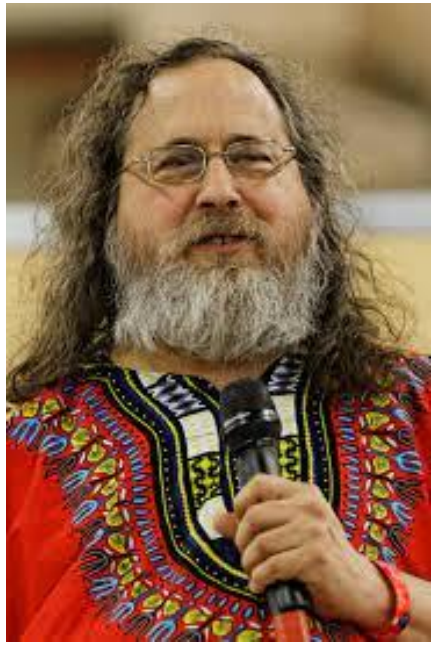
gdb



GDB
The GNU Project
Debugger

- What is GDB?
  - GNU Project debugger
  - see what is going on ‘inside' another program while it executes
  - see what another program was doing the moment it crashed.

- Key features
  - Start your program, specifying anything that might affect its behavior.
  - Make your program stop on specified conditions.
  - Examine what has happened when your program has stopped.
  - Change things in your program so you can experiment with correcting the effects of one bug and go on to learn about another.



GNU

gdb

- Run

  $ gdb <program>

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

- Break Point

  >b <file_name>:<line>

  > b <file_name>:<function>

  >i b

  >info break

  >d <breakpoint #>

  >delete <breakpoint #>

  >en <breakpoint #>

  >enable <breakpoint #>

  >dis <breakpoint #>

  >disable <breakpoint #>

gdb

- Process

  >*r* >*run*

  >*c* >*continue*

  >*n* >*next*

  >*s* >*step*

  >*fin* >*finish*

- Print

  >*p <val>* >*print <val>*

  >*display <val>*

  >*info locals*

  >*info variables*

  >*info f*

LevelDB Example: Merge Iterator

```
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   0x0000000000012213 in leveldb::DBImpl::DoCompactionWork(leveldb::DBImpl::CompactionState*)
at /home/mingu/leveldb_release/db/db_impl.cc:894
(gdb)
```

Open file in Editor (Ctrl + Click)

LevelDB Example: Merge Iterator

```
(gdb) r
Starting program: /home/mingu/leveldb_release/build/db_bench --benchmarks=fillrandom
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
LevelDB:    version 1.23
Date:       Fri Jul 15 15:57:24 2022
CPU:        16 * Intel(R) Core(TM) i7-10700K CPU @ 3.80GHz
CPUCache:   16384 KB
Keys:       16 bytes each
Values:     100 bytes each (50 bytes after compression)
Entries:    1000000
RawSize:    110.6 MB (estimated)
FileSize:   62.9 MB (estimated)
WARNING: Optimization is disabled: benchmarks unnecessarily slow
WARNING: Assertions are enabled; benchmarks unnecessarily slow
------------------------------------------------
[New Thread 0x7ffff79bd700 (LWP 7544)]
[New Thread 0x7ffff6eca700 (LWP 7545)]
[Switching to Thread 0x7ffff6eca700 (LWP 7545)]

Thread 3 "db_bench" hit Breakpoint 1, leveldb::DBImpl::DoCompactionWork (this=0x5555555dbc70, compact=0x7fffe8000c20)
    at /home/mingu/leveldb_release/db/db_impl.cc:894
894       int64_t imm_micros = 0;  // Micros spent doing imm_ compactions
(gdb)
```

LevelDB Example: Merge Iterator

```
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)
```

LevelDB Example: Merge Iterator

```
(gdb) display smallest
1: smallest = (leveldb::IteratorWrapper *) 0x0
(gdb) display child
2: child = (leveldb::IteratorWrapper *) 0x7fffe8010438
(gdb) n
150       for (int i = 0; i < n_; i++) {
1: smallest = (leveldb::IteratorWrapper *) 0x7fffe8010438
(gdb)
151         IteratorWrapper* child = &children_[i];
1: smallest = (leveldb::IteratorWrapper *) 0x7fffe8010438
2: child = (leveldb::IteratorWrapper *) 0x7fffe8010438
(gdb) display *smallest
3: *smallest = {iter_ = 0x7fffe8012110, valid_ = true, key_ = {
    data_ = 0x7fffe8012b70 '0' <repeats 13 times>, "198\001\225X\002", size_ = 24}}
(gdb) display *child
4: *child = {iter_ = 0x7fffe8012110, valid_ = true, key_ = {
    data_ = 0x7fffe8012b70 '0' <repeats 13 times>, "198\001\225X\002", size_ = 24}}
(gdb) n
152         if (child->Valid()) {
1: smallest = (leveldb::IteratorWrapper *) 0x7fffe8010438
2: child = (leveldb::IteratorWrapper *) 0x7fffe8010458
3: *smallest = {iter_ = 0x7fffe8012110, valid_ = true, key_ = {
    data_ = 0x7fffe8012b70 '0' <repeats 13 times>, "198\001\225X\002", size_ = 24}}
4: *child = {iter_ = 0x7fffe800ff50, valid_ = true, key_ = {
    data_ = 0x7fffe8012e30 '0' <repeats 14 times>, "40\001\213\206\001", size_ = 24}}
(gdb)
```

```
void MergingIterator::FindSmallest() {
  IteratorWrapper* smallest = nullptr;
  for (int i = 0; i < n_; i++) {
    IteratorWrapper* child = &children_[i];
    if (child->Valid()) {
      if (smallest == nullptr) {
        smallest = child;
      } else if (comparator_->Compare(child->key(),
        smallest = child;
      }
    }
  }
  current_ = smallest;
}
```

LevelDB Example: Merge Iterator

```
void MergingIterator::FindSmallest() {
  IteratorWrapper* smallest = nullptr;
  for (int i = 0; i < n_; i++) {
    IteratorWrapper* child = &children_[i];
    if (child->Valid()) {
      if (smallest == nullptr) {
        smallest = child;
      } else if (comparator_->Compare(child->key(),
        smallest = child;
      }
    }
  }
  current_ = smallest;
}
```

```
156             smallest = child;
3: *smallest = {iter_ = 0x7fffe800ff50, valid_ = true, key_ = {
    data_ = 0x7fffe8012e30 '0' <repeats 14 times>, "40\001\213\206\001", size_ = 24}}
4: *child = {iter_ = 0x7fffe80100a0, valid_ = true, key_ = {
    data_ = 0x7fffe800fa30 '0' <repeats 14 times>, "30\001\211\024\002", size_ = 24}}
(gdb)
150         for (int i = 0; i < n_; i++) {
3: *smallest = {iter_ = 0x7fffe80100a0, valid_ = true, key_ = {
    data_ = 0x7fffe800fa30 '0' <repeats 14 times>, "30\001\211\024\002", size_ = 24}}
```

LevelDB Example: Merge Iterator

```
void MergingIterator::FindSmallest() {
  IteratorWrapper* smallest = nullptr;
  for (int i = 0; i < n_; i++) {
    IteratorWrapper* child = &children_[i];
    if (child->Valid()) {
      if (smallest == nullptr) {
        smallest = child;
      } else if (comparator_->Compare(child->key(),
        smallest = child;
      }
    }
  }
  current_ = smallest;
}
```

```
160       current_ = smallest;
3: *smallest = {iter_ = 0x7fffe80100a0, valid_ = true, key_ = {
    data_ = 0x7fffe800fa30 '0' <repeats 14 times>, "30\001\211\024\002", size_ = 24}}
7: *current_ = {iter_ = 0x7fffe800ff50, valid_ = true, key_ = {
    data_ = 0x7fffe8012e30 '0' <repeats 14 times>, "40\001\213\206\001", size_ = 24}}
(gdb)
161     }
3: *smallest = {iter_ = 0x7fffe80100a0, valid_ = true, key_ = {
    data_ = 0x7fffe800fa30 '0' <repeats 14 times>, "30\001\211\024\002", size_ = 24}}
7: *current_ = {iter_ = 0x7fffe80100a0, valid_ = true, key_ = {
    data_ = 0x7fffe800fa30 '0' <repeats 14 times>, "30\001\211\024\002", size_ = 24}}
```

References

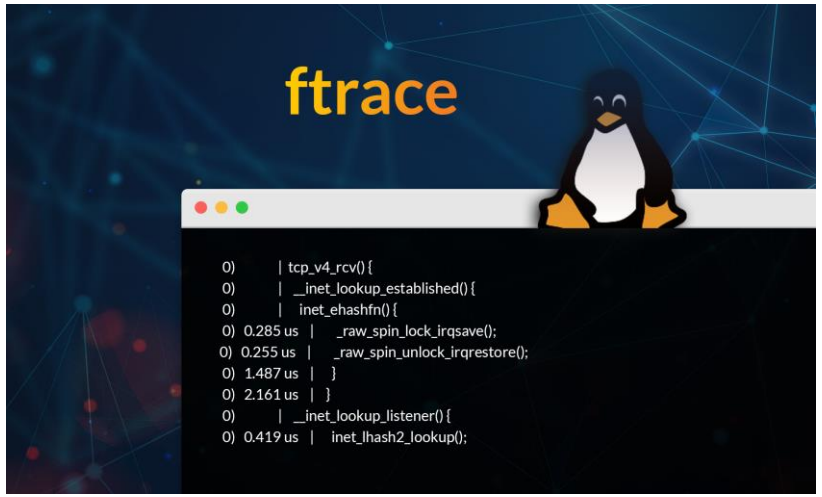
- KOR
  - gdb 디버거 사용법 및 다양한 기능 설명
    - https://edward0im.github.io/technology/2020/09/29/gdb/
  - gdb - 간단한 명령어/사용법/단축어 정리(cheat sheet)
    - https://dining-developer.tistory.com/13’

1. How to analyze LevelDB
2. VS code
3. Understand
4. GDB
5. ***Uftrace***

- Uftrace
  - ✓ Features
    - Record
    - Replay
    - Filter
    - TUI
  - ✓ Shell Script
  - ✓ Installation

uftrace

- Trace and analyze execution of a program written in C/C++
- Heavily inspired by the ftrace framework of the Linux kernel
- Supports user-space programs and kernel
- Various kind of commands and filters



Features

- Record
  - Runs 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

Replay: Write()

```
# DURATION     TID     FUNCTION
            [  5471] | leveldb::DBImpl::Write() {
            [  5471] |   leveldb::DBImpl::Writer::Writer()
   0.030 us [  5471] |     leveldb::Status::Status();
            [  5471] |     leveldb::port::CondVar::CondVar(
   0.164 us [  5471] |       std::condition_variable::condi
   0.296 us [  5471] |     } /* leveldb::port::CondVar::Co
   0.508 us [  5471] |   } /* leveldb::DBImpl::Writer::Writ
            [  5471] |   leveldb::MutexLock::MutexLock() {
            [  5471] |     leveldb::port::Mutex::Lock() {
            [  5471] |       std::mutex::lock() {
            [  5471] |         __gthread_mutex_lock() {
   0.040 us [  5471] |           __gthread_active_p();
   0.313 us [  5471] |           pthread_mutex_lock();
   1.758 us [  5471] |         } /* __gthread_mutex_lock *
   2.058 us [  5471] |       } /* std::mutex::lock */
   2.133 us [  5471] |     } /* leveldb::port::Mutex::Lock
   2.217 us [  5471] |   } /* leveldb::MutexLock::MutexLoc
            [  5471] |   std::deque::push_back() {
   0.029 us [  5471] |     std::move();
            [  5471] |     std::deque::emplace_back() {
   0.031 us [  5471] |       std::forward();
            [  5471] |       std::allocator_traits::constr
   0.029 us [  5471] |         std::forward();
```

- Record

  $ uftrace record ./db_bench benchmarks="fillrandom" \

  --num=100000

- Replay

  $ uftrace replay

  > /leveldb::DBImpl::Write

  $ uftrace replay –F leveldb::DBImpl::Write

Filter: Write()

```
# DURATION     TID     FUNCTION
            [  5471] | leveldb::DBImpl::Write() {
            [  5471] |   leveldb::DBImpl::Writer::Writer() {
   0.030 us [  5471] |     leveldb::Status::Status();
            [  5471] |     leveldb::port::CondVar::CondVar() {
   0.164 us [  5471] |       std::condition_variable::condition_variable();
   0.296 us [  5471] |     } /* leveldb::port::CondVar::CondVar */
   0.508 us [  5471] |   } /* leveldb::DBImpl::Writer::Writer */
            [  5471] |   leveldb::MutexLock::MutexLock() {
            [  5471] |     leveldb::port::Mutex::Lock() {
            [  5471] |       std::mutex::lock() {
            [  5471] |         __gthread_mutex_lock() {
   0.040 us [  5471] |           __gthread_active_p();
   0.313 us [  5471] |           pthread_mutex_lock();
   1.758 us [  5471] |         } /* __gthread_mutex_lock */
   2.058 us [  5471] |       } /* std::mutex::lock */
   2.133 us [  5471] |     } /* leveldb::port::Mutex::Lock */
   2.217 us [  5471] |   } /* leveldb::MutexLock::MutexLock */
            [  5471] |   std::deque::push_back() {
   0.029 us [  5471] |     std::move();
            [  5471] |     std::deque::emplace_back() {
   0.031 us [  5471] |       std::forward();
            [  5471] |       std::allocator_traits::construct() {
   0.029 us [  5471] |         std::forward();
```

No Filter

```
# DURATION     TID     FUNCTION
            [ 14234] | leveldb::DBImpl::Write() {
            [ 14234] |   leveldb::DBImpl::Writer::Writer() {
   0.311 us [ 14234] |     leveldb::port::CondVar::CondVar();
   0.493 us [ 14234] |   } /* leveldb::DBImpl::Writer::Writer */
   1.963 us [ 14234] |   leveldb::MutexLock::MutexLock();
            [ 14234] |   leveldb::DBImpl::MakeRoomForWrite() {
   0.262 us [ 14234] |     leveldb::VersionSet::NumLevelFiles();
   0.651 us [ 14234] |     leveldb::MemTable::ApproximateMemoryUsage();
   1.637 us [ 14234] |   } /* leveldb::DBImpl::MakeRoomForWrite */
   0.079 us [ 14234] |   leveldb::VersionSet::LastSequence();
            [ 14234] |   leveldb::DBImpl::BuildBatchGroup() {
   0.161 us [ 14234] |     leveldb::WriteBatchInternal::ByteSize();
   1.423 us [ 14234] |   } /* leveldb::DBImpl::BuildBatchGroup */
   0.315 us [ 14234] |   leveldb::WriteBatchInternal::SetSequence();
   0.221 us [ 14234] |   leveldb::WriteBatchInternal::Count();
   0.328 us [ 14234] |   leveldb::WriteBatchInternal::Contents();
            [ 14234] |   leveldb::log::Writer::AddRecord() {
            [ 14234] |     leveldb::log::Writer::EmitPhysicalRecord() {
   0.757 us [ 14234] |       leveldb::_GLOBAL__N_1::PosixWritableFile::Append();
   0.578 us [ 14234] |       leveldb::_GLOBAL__N_1::PosixWritableFile::Append();
```

Filter

Filiter

COMMON OPTIONS

-F *FUNC*, --filter=*FUNC* : Set filter to trace selected functions and their children functions. This option can be used more than once. See *FILTERS*.

-N *FUNC*, --notrace=*FUNC* : Set filter not to trace selected functions and their children functions. This option can be used more than once. See *FILTERS*.

-C *FUNC*, --caller-filter=*FUNC* : Set filter to trace callers of selected functions only. This option can be used more than once. See *FILTERS*.

-T *TRG*, --trigger=*TRG* : Set trigger on selected functions. This option can be used more than once. See *TRIGGERS*.

-D *DEPTH*, --depth *DEPTH* : Set trace limit in nesting level. See *FILTERS*.

-t *TIME*, --time-filter=*TIME* : Do not show functions which run under the time threshold. If some functions explicitly have the 'trace' trigger applied, those are always traced regardless of execution time. See *FILTERS*.

--no-libcall : Do not show library calls.

COMMON ANALYSIS OPTIONS

-H *FUNC*, --hide=*FUNC* : Set filter not to trace selected functions. It doesn't affect their subtrees, but hides only the given functions. This option can be used more than once. See *FILTERS*.

--kernel-full : Show all kernel functions and events occurred outside of user functions.

--kernel-only : Show kernel functions only without user functions.

--event-full : Show all (user) events outside of user functions.

--tid=*TID*[,*TID*,...] : Only print functions called by the given tasks. To see the list of tasks in the data file, you can use `uftrace report --task` or `uftrace info`. This option can also be used more than once.

Uftrace/doc/uftrace-replay.md

Tui: Write()

```
TOTAL TIME : FUNCTION
  11.538  s : (1) db_bench
   5.502  s : (85352) leveldb::DBImpl::Write
  26.224 ms :   ├─(85352) leveldb::DBImpl::Writer::Writer
  12.326 ms :   │ (85352) leveldb::port::CondVar::CondVar
            :   │
  39.723 ms :   ├─(85352) leveldb::MutexLock::MutexLock
            :   │
  72.355 ms :   ├─(85352) leveldb::DBImpl::MakeRoomForWrite
  10.459 ms :   │ ├─(85358) leveldb::VersionSet::NumLevelFiles
            :   │ │
  16.385 ms :   │ ├─(85355) leveldb::MemTable::ApproximateMemoryUsage
            :   │ │
   0.090 us :   │ ├─(3) leveldb::VersionSet::PrevLogNumber
            :   │ │
   0.100 us :   │ ├─(3) leveldb::VersionSet::NewFileNumber
            :   │ │
   6.257 us :   │ ├─(3) leveldb::LogFileName
   5.965 us :   │ │ (3) leveldb::MakeFileName
            :   │ │
  71.581 us :   │ ├─(3) leveldb::_GLOBAL__N_1::PosixEnv::NewWritableFile
   8.899 us :   │ │ (3) leveldb::_GLOBAL__N_1::PosixWritableFile::PosixWritableFile
   0.103 us :   │ │   ├─(3) leveldb::WritableFile::WritableFile
            :   │ │   │
   4.749 us :   │ │   ├─(3) leveldb::_GLOBAL__N_1::PosixWritableFile::IsManifest
   2.985 us :   │ │   │ (3) leveldb::_GLOBAL__N_1::PosixWritableFile::Basename
            :   │ │   │
   2.038 us :   │ │   └─(3) leveldb::_GLOBAL__N_1::PosixWritableFile::Dirname
            :   │ │
   0.093 us :   │ ├─(3) leveldb::log::Writer::~Writer
            :   │ │
  10.424 us :   │ ├─(3) leveldb::_GLOBAL__N_1::PosixWritableFile::~PosixWritableFile
   8.303 us :   │ │ (3) leveldb::_GLOBAL__N_1::PosixWritableFile::~PosixWritableFile
   6.650 us :   │ │   ├─(3) leveldb::_GLOBAL__N_1::PosixWritableFile::Close
uftrace graph: session 241c4796d8967ad9 (/home/mingu/leveldb_release/build/db_bench)
```

- Tui
  - Back-trace
  - Call Graph

- Run uftrace

  $ uftrace record ./db_bench – benchmarks=“fillrandom” \

  --num=100000

  $ uftrace tui

  >/ leveldb::DBImpl::Write

  >g

Tui: SkipList::Insert

```
========== 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
  4.824  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 :   │       ├─(4808728) 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
  5.508 ms :   │   ├─(100000) leveldb::Arena::AllocateAligned
  1.455 ms :   │   │ (890) leveldb::Arena::AllocateFallback
  1.384 ms :   │   │ (890) leveldb::Arena::AllocateNewBlock
           :   │   │
```

```
Help: (press any key to exit)

 ARROW          Navigation
 PgUp/Dn
 Home/End
 Enter          Fold/unfold graph or Select session
 G              Show (full) call graph
 g              Show call graph for this function
 R              Show uftrace report
 r              Show uftrace report for this function
 s              Sort by the next column in report
 I              Show uftrace info
 S              Change session
 O              Open editor
 c/e            Collapse/Expand direct children graph
 C/E            Collapse/Expand all descendant graph
 n/p            Next/Prev sibling
 u              Move up to parent
 l              Move to the longest executed child
 j/k            Move down/up
 z              Set current line to the center of screen
 /              Search
 </>/N/P        Search next/prev
 v              Show debug message
 f              Customize fields in graph or report mode
 h/?            Show this help
 q              Quit
```

Shell script

```
#!/bin/sh

default_func=""

if [ -z "$1" ]
then
    echo '[USAGE] : sh uft_cmd [COMMAND] [TARGET FUNC]'
    echo '[COMMAND] : replay, report, graph, tui'
    exit
fi

if [ -n "$2" ]
then
    func="-F ${2}"
else
    if [ -z "$default_func" ]
    then
        func=""
    else
        func="-F ${default_func}"
    fi
fi

uftrace $1 \
$func \
--no-libcall \
-N leveldb::MutexLock \
-N leveldb::ExtractUserKey \
-N leveldb::Arena::MemoryUsage \
-N __gthread_mutex_unlock \
-N __gthread_mutex_lock \
-N ^leveldb::Slice:: \
-N ^leveldb::port::Mutex \
-N ^leveldb::crc32c:: \
```

- Shell script
  - Easy to use filter
  - uftrace_script.sh

- Run Shell script

$ uftrace record ./db_bench benchmarks=“fillrandom” \
--num=100000

$ sh uftrace_script.sh replay

$ sh uftrace_script.sh tui

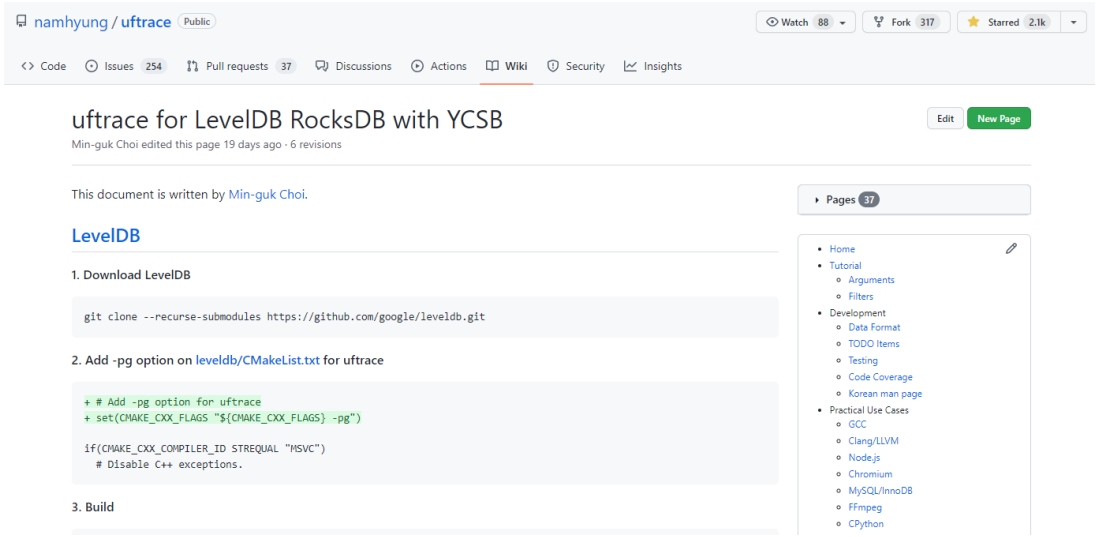
Install Guide

- Install uftrace
  - uftrace/INSTALL.md
    - https://github.com/namhyung/uftrace/blob/master/INSTALL.md
- Bind with leveldb
  - uftrace wiki – LevelDB/RocksDB (YCSB)
    - https://github.com/namhyung/uftrace/wiki/uftrace-for-LevelDB-RocksDB-with-YCSB
- Tutorial
  - https://uftrace.github.io/slide/#1
- Wiki
  - https://github.com/namhyung/uftrace/wiki

QUICK GUIDE

On Linux distros, following commands will build and install uftrace from source.

```
$ sudo misc/install-deps.sh    # optional for advanced features
$ ./configure                  # --prefix can be used to change install dir
$ make
$ sudo make install
```



Thank you