

# Team Compaction

좌오꾸와썬

E-Mail : erosbryant@dankook.ac.kr

강상우

E-Mail : aarom416@naver.com

발표: 박서영

E-Mail: lilianapsy@naver.com

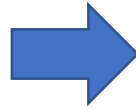
# Contents

- Discussions about the last experiments
  - Differences between the two experiments
- Compaction Code Flow
  - BGWork()
  - BackgroundCall()
  - BackgroundCompaction()

# db\_bench

- Experiment Setup

- Putty -> 서버



```
processor      : 19
vendor_id     : GenuineIntel
cpu family    : 6
model         : 151
model name    : 12th Gen Intel(R) Core(TM) i7-12700K
stepping      : 2
microcode     : 0x1f
cpu MHz       : 3600.000
cache size    : 25600 KB
```

Cpu정보

	total	used	free	shared	buff/cache	available
Mem:	67166	4660	51977	2	10528	61771
스왑:	2147	2	2144			

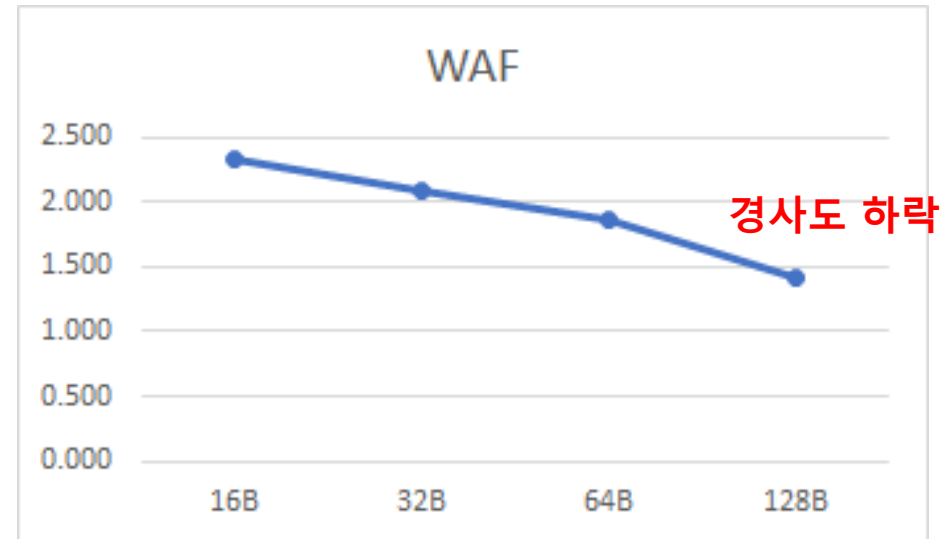
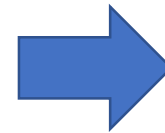
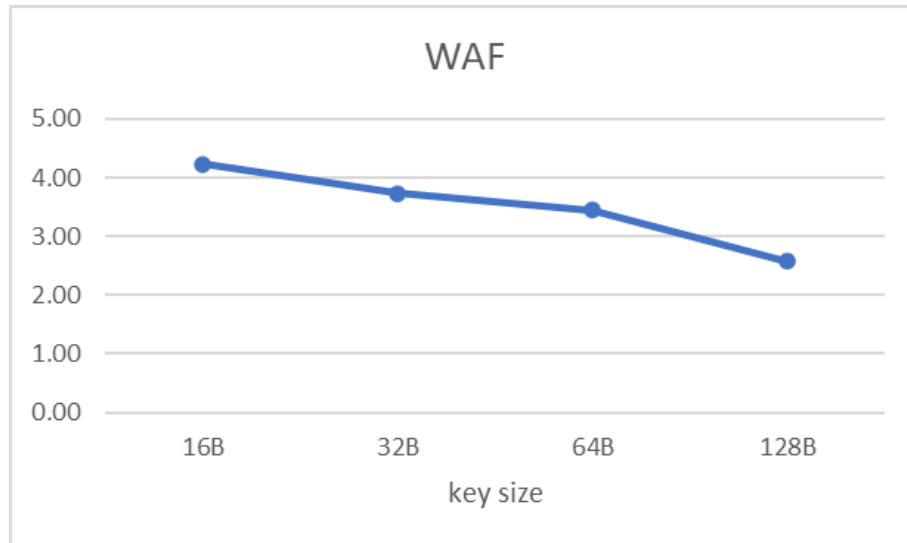
메모리정보

# db\_bench

- Various Key Size
  - 'fillrandom'
  - Num=1000000
  - Value=100 byte

16B	32B	64B	128B
2.259	1.962	1.866	1.408
2.336	2.008	1.866	1.408
2.319	2.182	1.866	1.408

the average of three values(key)



Re-measurement

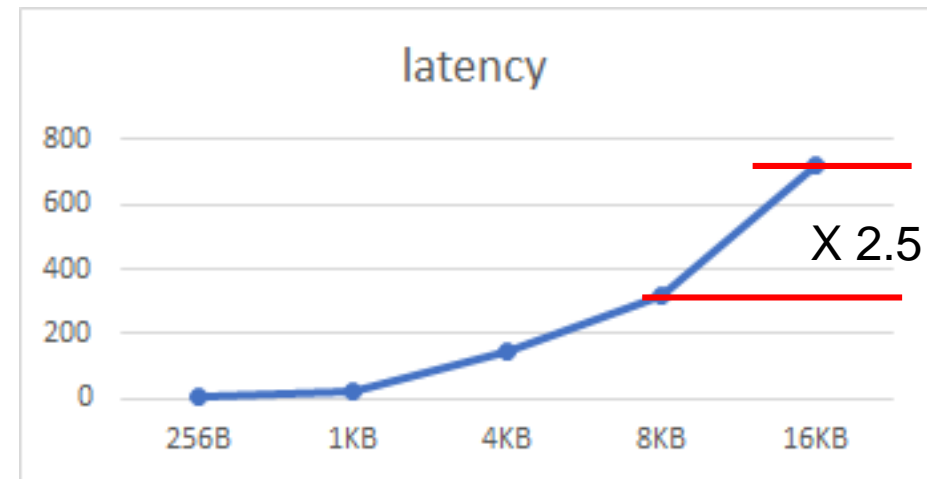
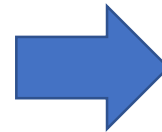
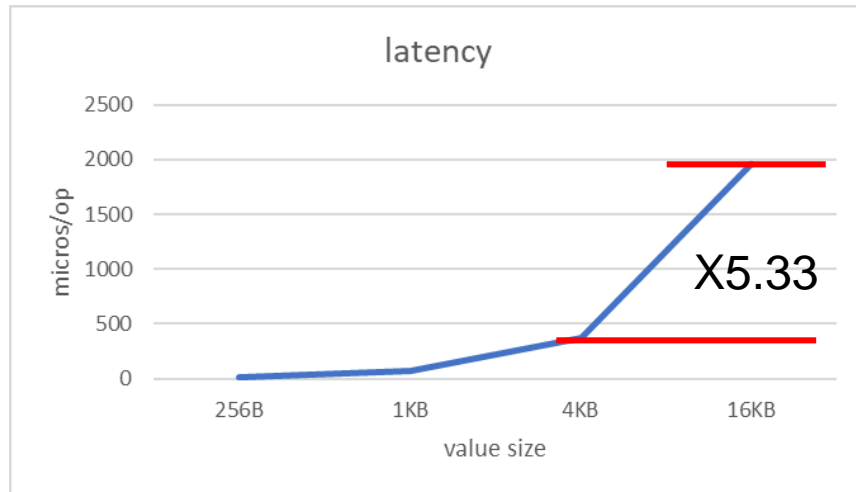
# db\_bench

- Various Value Size

- 'fillrandom'
- Num=1000000
- key=16 byte
- add segmentation

256B	1KB	4KB	8KB	16KB
4.632	25.804	142.405	316.174	717.444
4.785	25.631	144.432	316.442	715.414
4.768	27.343	143.815	317.211	715.982

the average of three values



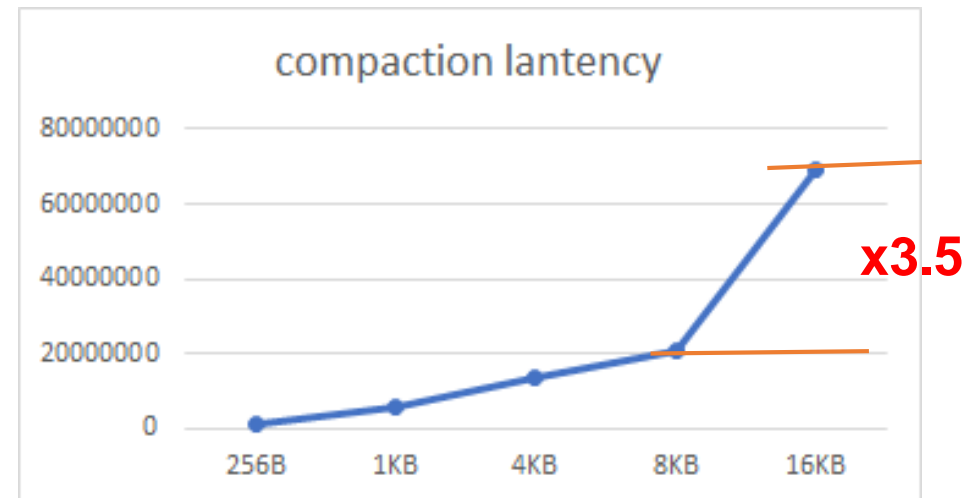
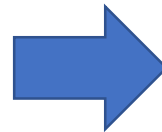
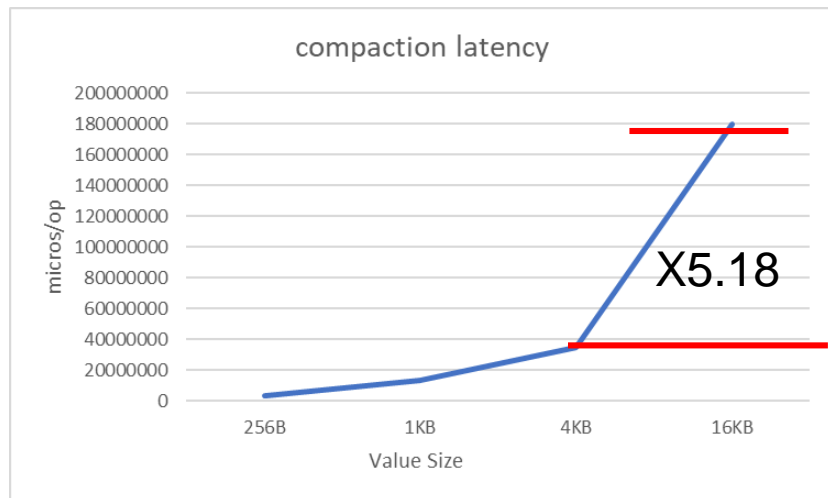
Re-measurement

# db\_bench

- Various Value Size
  - 'fillrandom'
  - Num=1000000
  - key=16 byte
  - add segmentation

256B	1KB	4KB	8KB	16KB
1284896	5368554	14033956	21291306	68876877
1231927	5107441	12875517	21225322	68835053
1371165	5958228	13298479	20485921	68815213

the average of three values



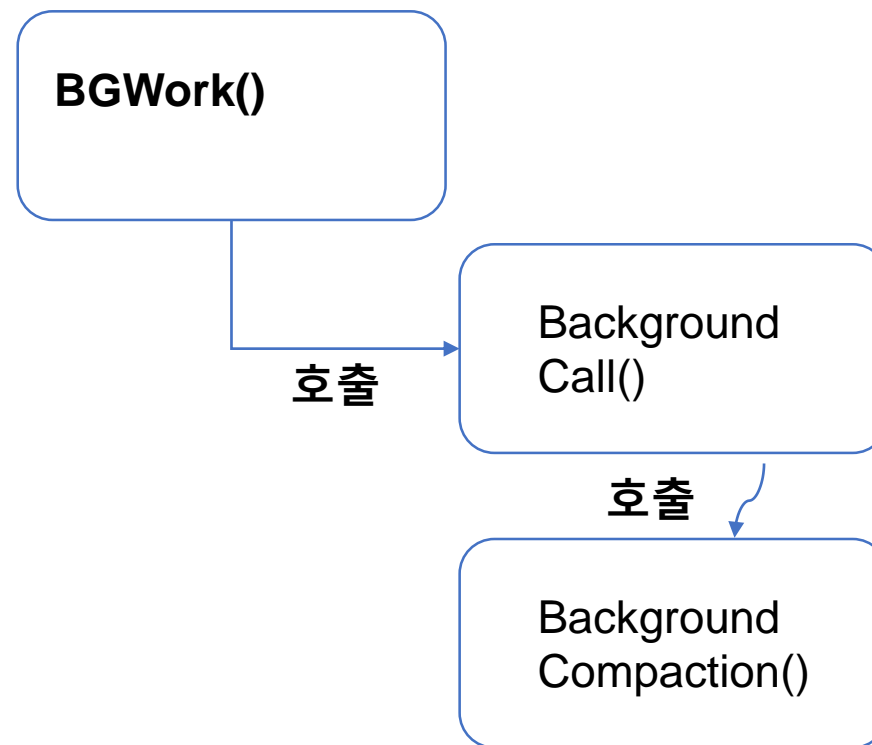
Re-measurement

# Uftrace record: --benchmarks="fillrandom", --num=10000

```
erosbryant@ErosBryant-computer:~/workspace/seoyoung/leveldb/build$ uftrace record ./db_bench --benchmarks="fillrandom" --num=10000
0
LevelDB: version 1.23
Date:    Sat Jul 30 22:11:10 2022
CPU:     20 * 12th Gen Intel(R) Core(TM) i7-12700K
CPUCache: 25600 KB
Keys:    16 bytes each
Values:  100 bytes each (50 bytes after compression)
Entries: 10000
RawSize: 1.1 MB (estimated)
FileSize: 0.6 MB (estimated)
WARNING: Optimization is disabled: benchmarks unnecessarily slow
WARNING: Assertions are enabled; benchmarks unnecessarily slow
-----
fillrandom :      66.111 micros/op;    1.7 MB/s
erosbryant@ErosBryant-computer:~/workspace/seoyoung/leveldb/build$ uftrace record ./db_bench --benchmarks="fillrandom" --num=10000
00
LevelDB: version 1.23
Date:    Sat Jul 30 22:11:18 2022
CPU:     20 * 12th Gen Intel(R) Core(TM) i7-12700K
CPUCache: 25600 KB
Keys:    16 bytes each
Values:  100 bytes each (50 bytes after compression)
Entries: 100000
RawSize: 11.1 MB (estimated)
FileSize: 6.3 MB (estimated)
WARNING: Optimization is disabled: benchmarks unnecessarily slow
WARNING: Assertions are enabled; benchmarks unnecessarily slow
-----
fillrandom :      58.257 micros/op;    1.9 MB/s
```

# Compaction Code Flow: BGWork()->BackgroundCall()->BackgroundCompaction()

```
g) 실행(R) 터미널(T) 도움말(H) • db_impl.cc - seoyoung [SSH: 220.149.250.124] - Visual Studio Code
G+ db_impl.cc 9+ •
leveldb > db > G+ db_impl.cc > ...
678 void DBImpl::BGWork(void* db) {
679     reinterpret_cast<DBImpl*>(db)->BackgroundCall();
680 }
681
682 void DBImpl::BackgroundCall() {
683     MutexLock l(&mutex_);
684     assert(background_compaction_scheduled_);
685     if (shutting_down_.load(std::memory_order_acquire)) {
686         // No more background work when shutting down.
687     } else if (!bg_error_.ok()) {
688         // No more background work after a background error.
689     } else {
690         BackgroundCompaction();
691     }
692
693     background_compaction_scheduled_ = false;
694
695     // Previous compaction may have produced too many files in a level,
696     // so reschedule another compaction if needed.
697     MaybeScheduleCompaction();
698     background_work_finished_signal_.SignalAll();
699 }
700
```





# Compaction tui: BackGroundCompaction()

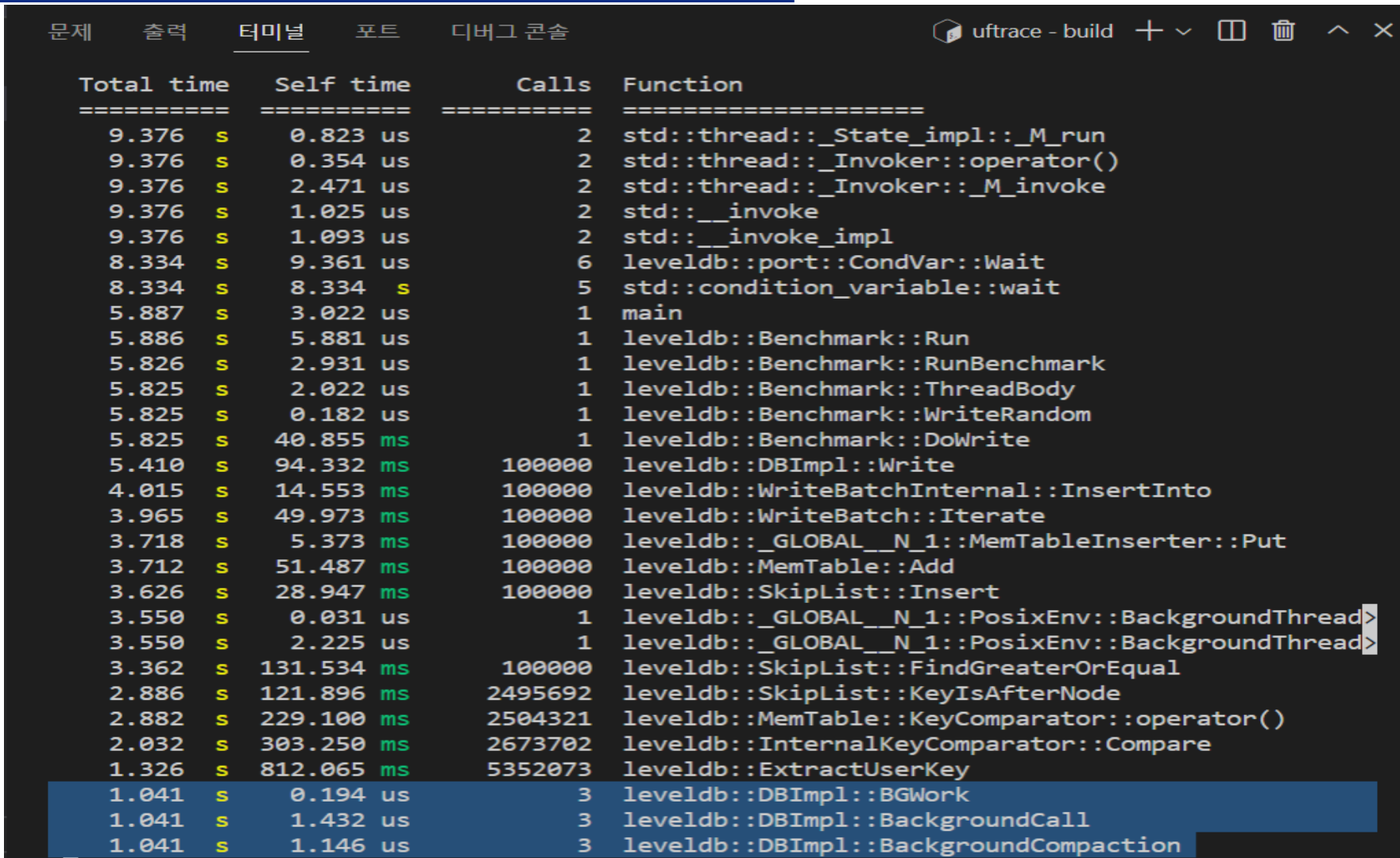
TOTAL TIME	FUNCTION
1.041 s	(3) leveldb::DBImpl::BGWork
1.041 s	(3) leveldb::DBImpl::BackgroundCall
2.625 us	(3) leveldb::MutexLock::MutexLock
2.399 us	(3) leveldb::port::Mutex::Lock
2.222 us	(3) std::mutex::lock
1.959 us	(3) __gthread_mutex_lock
0.078 us	(3) __gthread_active_p
0.354 us	(3) pthread_mutex_lock
0.481 us	(3) std::atomic::load
0.086 us	(3) std::operator&
0.085 us	(3) leveldb::Status::ok
1.041 s	(3) leveldb::DBImpl::BackgroundCompaction
0.093 us	(3) leveldb::port::Mutex::AssertHeld
1.041 s	(3) leveldb::DBImpl::CompactMemTable
0.092 us	(3) leveldb::port::Mutex::AssertHeld
8.840 us	(3) leveldb::VersionEdit::VersionEdit
0.407 us	(3) std::__cxx11::basic_string::basic_string
2.794 us	(6) std::vector::vector
2.322 us	(6) std::_Vector_base::_Vector_base
1.877 us	(6) std::_Vector_base::_Vector_impl::_Vector_impl
0.869 us	(6) std::allocator::allocator
0.146 us	(6) __gnu_cxx::new_allocator::new_allocator
0.173 us	(6) std::_Vector_base::_Vector_impl_data::_Vector_impl_data
1.638 us	(3) std::set::set
1.455 us	(3) std::_Rb_tree::_Rb_tree

uftrace graph: source location is not available [at 0x5643b9973040]

BGWork함수를  
기반으로  
여러 함수들이  
실행되는 것을  
알 수 있음

그 중, Background  
Compaction에 대해  
탐구.

# Compaction Code Flow: BackgroundCompaction()



Total time	Self time	Calls	Function
9.376 s	0.823 us	2	std::thread::_State_impl::_M_run
9.376 s	0.354 us	2	std::thread::_Invoker::operator()
9.376 s	2.471 us	2	std::thread::_Invoker::_M_invoke
9.376 s	1.025 us	2	std::__invoke
9.376 s	1.093 us	2	std::__invoke_impl
8.334 s	9.361 us	6	leveldb::port::CondVar::Wait
8.334 s	8.334 s	5	std::condition_variable::wait
5.887 s	3.022 us	1	main
5.886 s	5.881 us	1	leveldb::Benchmark::Run
5.826 s	2.931 us	1	leveldb::Benchmark::RunBenchmark
5.825 s	2.022 us	1	leveldb::Benchmark::ThreadBody
5.825 s	0.182 us	1	leveldb::Benchmark::WriteRandom
5.825 s	40.855 ms	1	leveldb::Benchmark::DoWrite
5.410 s	94.332 ms	100000	leveldb::DBImpl::Write
4.015 s	14.553 ms	100000	leveldb::WriteBatchInternal::InsertInto
3.965 s	49.973 ms	100000	leveldb::WriteBatch::Iterate
3.718 s	5.373 ms	100000	leveldb::_GLOBAL__N_1::MemTableInserter::Put
3.712 s	51.487 ms	100000	leveldb::MemTable::Add
3.626 s	28.947 ms	100000	leveldb::SkipList::Insert
3.550 s	0.031 us	1	leveldb::_GLOBAL__N_1::PosixEnv::BackgroundThread>
3.550 s	2.225 us	1	leveldb::_GLOBAL__N_1::PosixEnv::BackgroundThread>
3.362 s	131.534 ms	100000	leveldb::SkipList::FindGreaterOrEqual
2.886 s	121.896 ms	2495692	leveldb::SkipList::KeyIsAfterNode
2.882 s	229.100 ms	2504321	leveldb::MemTable::KeyComparator::operator()
2.032 s	303.250 ms	2673702	leveldb::InternalKeyComparator::Compare
1.326 s	812.065 ms	5352073	leveldb::ExtractUserKey
1.041 s	0.194 us	3	leveldb::DBImpl::BGWork
1.041 s	1.432 us	3	leveldb::DBImpl::BackgroundCall
1.041 s	1.146 us	3	leveldb::DBImpl::BackgroundCompaction

옆 사진을 보면  
세 함수가 총 1.041s 초씩  
동일하게 3번 호출되어  
실행되는 것을 알 수 있음.

(단, 각각의 self time은 다름)

# Compaction Function Graph & Code

## :BackgroundCompaction()

TOTAL TIME	FUNCTION
1.041 s	(3) leveldb::DBImpl::BackgroundCompaction
0.093 us	(3) leveldb::port::Mutex::AssertHeld
1.041 s	(3) leveldb::DBImpl::CompactMemTable
0.092 us	(3) leveldb::port::Mutex::AssertHeld
8.840 us	(3) leveldb::VersionEdit::VersionEdit
0.407 us	(3) std::__cxx11::basic_string::basic_string
2.794 us	(6) std::vector::vector
2.322 us	(6) std::_Vector_base::_Vector_base
1.877 us	(6) std::_Vector_base::_Vector_impl::_Vector_impl
0.869 us	(6) std::allocator::allocator
0.146 us	(6) __gnu_cxx::new_allocator::new_allocator
0.173 us	(6) std::_Vector_base::_Vector_impl_data::_V
1.638 us	(3) std::set::set
1.455 us	(3) std::_Rb_tree::_Rb_tree
1.250 us	(3) std::_Rb_tree::_Rb_tree_impl::_Rb_tree_impl
0.364 us	(3) std::allocator::allocator
0.075 us	(3) __gnu_cxx::new_allocator::new_allocator
0.075 us	(3) std::_Rb_tree_key_compare::_Rb_tree_key_compare
0.336 us	(3) std::_Rb_tree_header::_Rb_tree_header
0.088 us	(3) std::_Rb_tree_header::_M_reset
2.985 us	(3) leveldb::VersionEdit::Clear
0.285 us	(3) std::__cxx11::basic_string::clear

uftrace graph: source location is not available [at 0x5643b997316c]

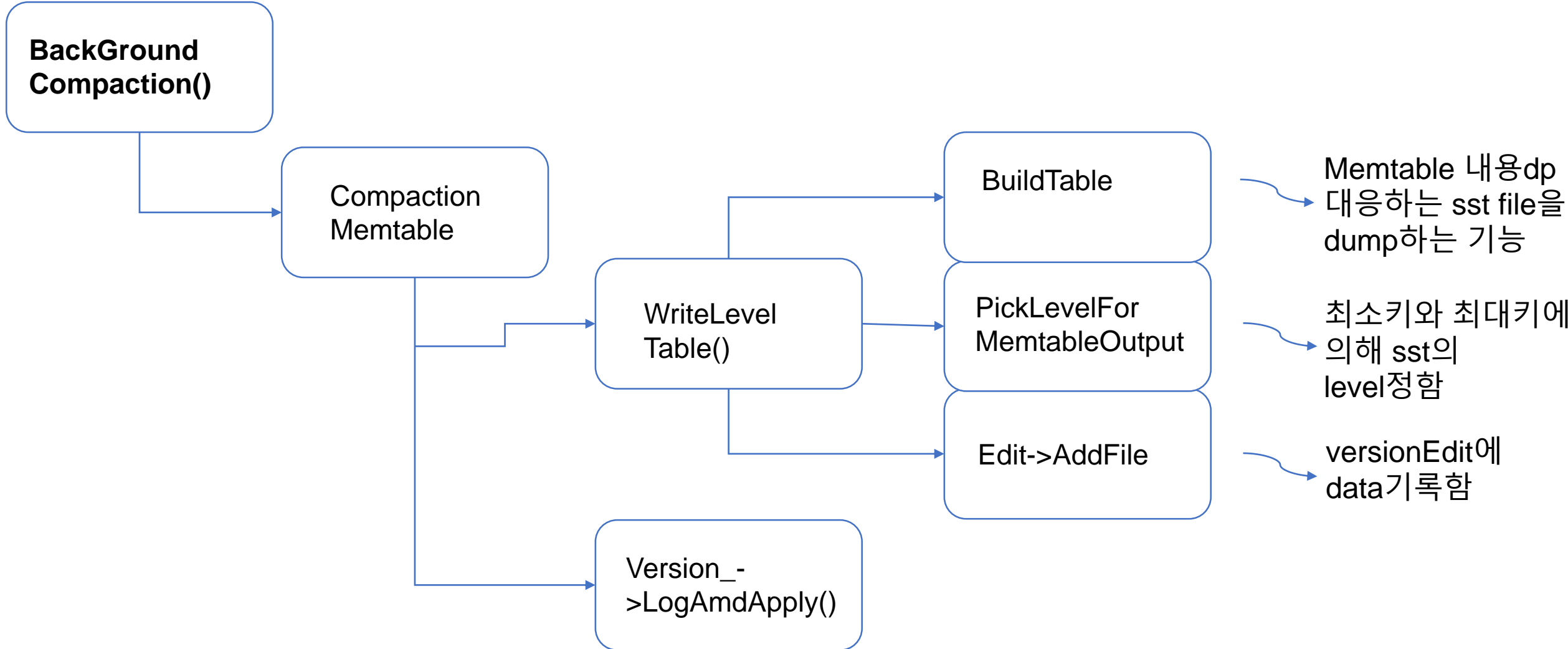
```

leveldb > db > db_impl.cc > BackgroundCompaction()
700 void DBImpl::BackgroundCompaction() {
701     mutex_.AssertHeld();
702     if (imm_ != nullptr) {
703         CompactMemTable();
704         return;
705     }
706     Compaction* c;
707     bool is_manual = (manual_compaction_ != nullptr);
708     InternalKey manual_end;
709     if (is_manual) {
710         ManualCompaction* m = manual_compaction_;
711         c = versions_>CompactRange(m->level, m->begin, m->end);
712         m->done = (c == nullptr);
713         if (c != nullptr) {
714             manual_end = c->input(0, c->num_input_files(0) - 1)->largest;
715         }
716         Log(options_.info_log,
717             "Manual compaction at level-%d from %s .. %s; will stop at %s\n",
718             m->level, (m->begin ? m->begin->DebugString().c_str() : "(begin)"),
719             (m->end ? m->end->DebugString().c_str() : "(end)"),
720             (m->done ? "(end)" : manual_end.DebugString().c_str()));
721     } else {
722         c = versions_>PickCompaction();
723     }
724 }

```

호출된 BackgroundCompaction()의  
그래프와 코드

# Compaction Code Flow: BackGroundCompaction()



# Thank you