HOMEWORK
4. Practice 2

쥬용지에

arashio1111@gmail.com

Contents:

1. 과제 설명
2. 기능 소개
3. 결과 분석

4. Practice 2

```
[Load] $ ./db_bench --benchmarks="fillrandom" --use_existing_db=0
[A] $ ./db_bench --benchmarks="readseq" --use_existing_db=1
[B] $ ./db_bench --benchmarks="readrandom" --use_existing_db=1
[C] $ ./db_bench --benchmarks="seekrandom" --use_existing_db=1
```

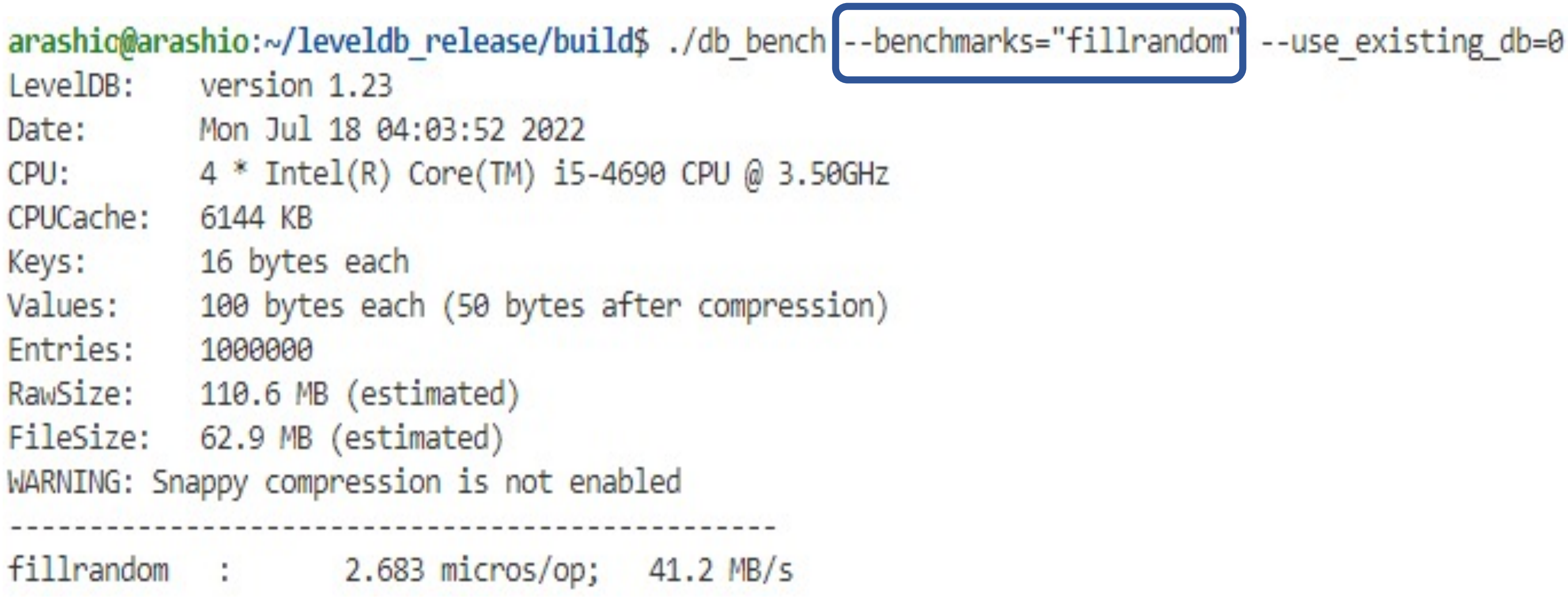
Note - Before running A, B, and C, run db_load benchmark.

Q1. Which user key-value interface does each benchmark use? (Put, Get, Iterator, ...)

Q2. Compare throughput and latency of each benchmark and explain why.

Q1. Which user key-value interface does each benchmark use?

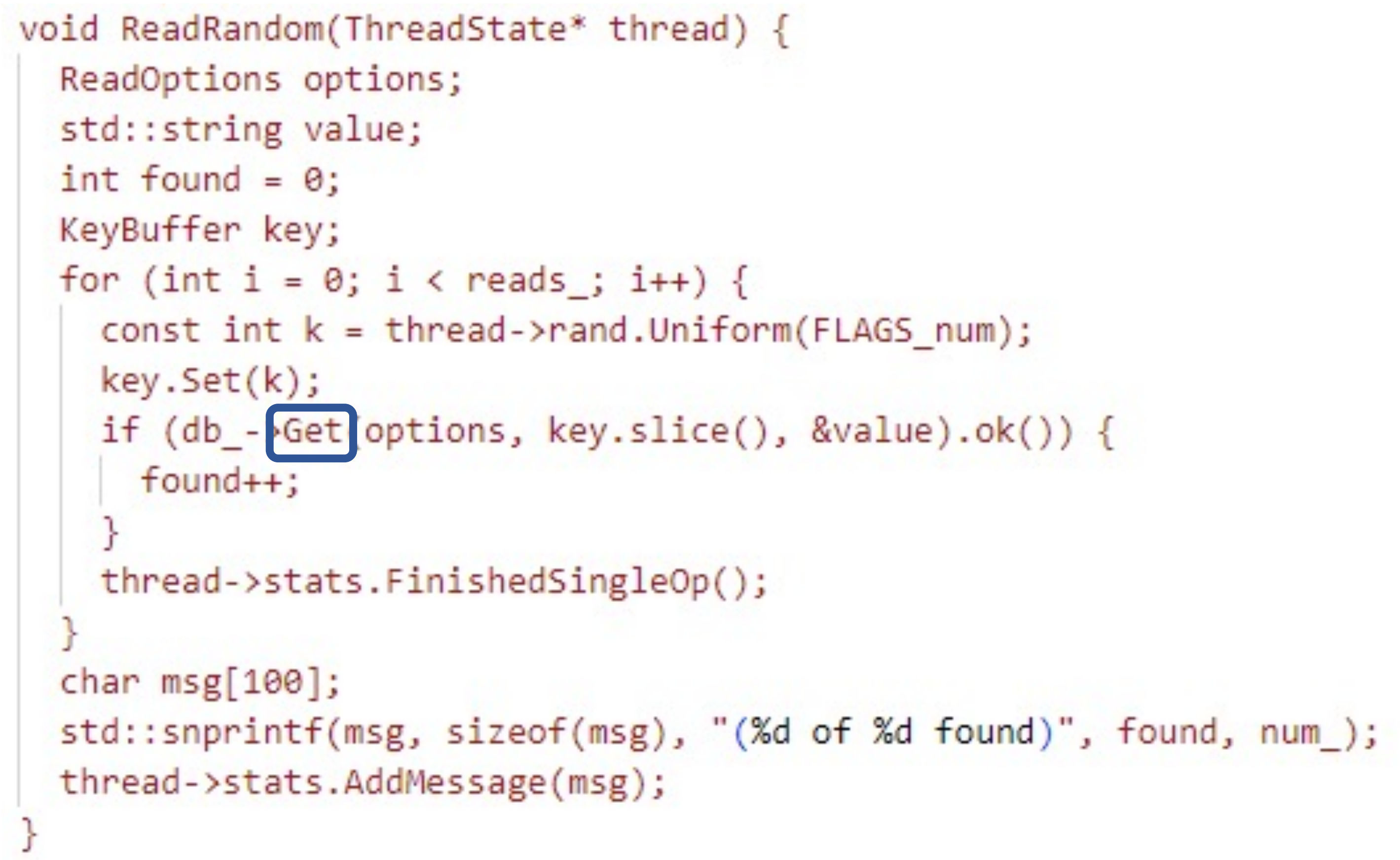
$ ./db_bench --benchmarks="fillrandom" --use_existing_db=0



```
arashio@arashio:~/leveldb_release/build$ ./db_bench --benchmarks="fillrandom" --use_existing_db=0
LevelDB:    version 1.23
Date:       Mon Jul 18 04:03:52 2022
CPU:        4 * Intel(R) Core(TM) i5-4690 CPU @ 3.50GHz
CPUCache:   6144 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: Snappy compression is not enabled
------------------------------------------------
fillrandom   :       2.683 micros/op;   41.2 MB/s
```

Q1. Which user key-value interface does each benchmark use?

$ ./db_bench --benchmarks="readrandom" --use_existing_db=1

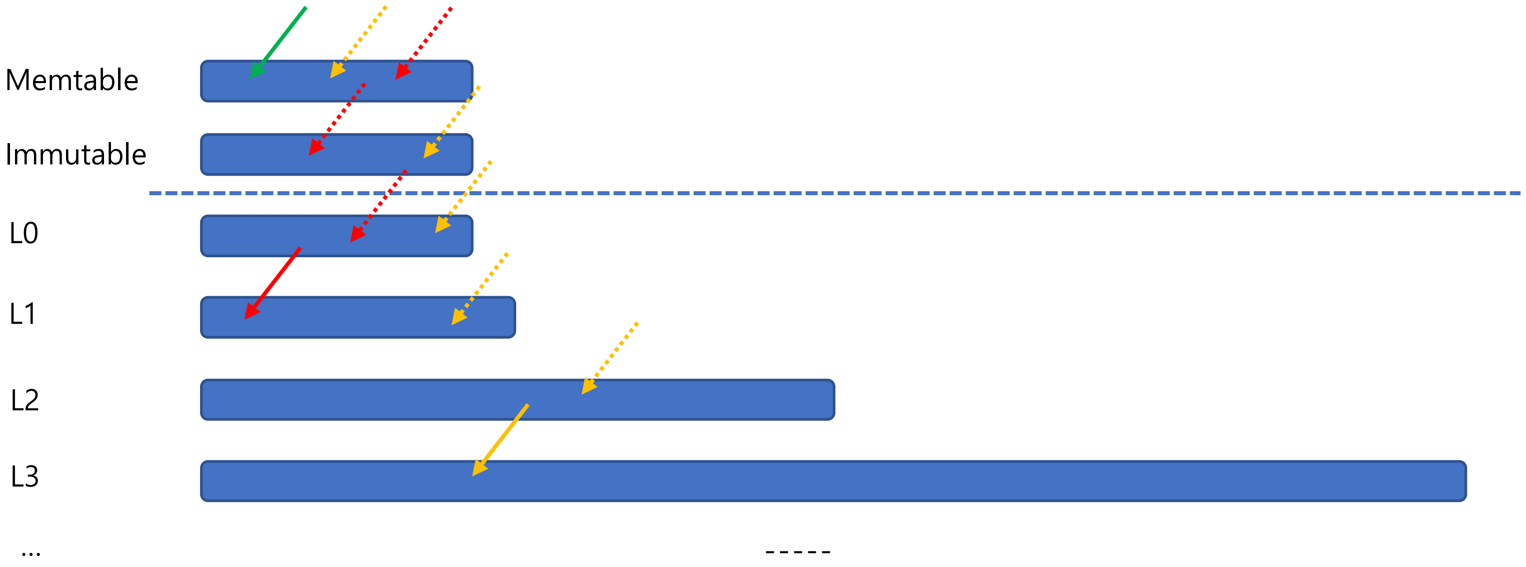


```
void ReadRandom(ThreadState* thread) {
  ReadOptions options;
  std::string value;
  int found = 0;
  KeyBuffer key;
  for (int i = 0; i < reads_; i++) {
    const int k = thread->rand.Uniform(FLAGS_num);
    key.Set(k);
    if (db_->Get(options, key.slice(), &value).ok()) {
      found++;
    }
    thread->stats.FinishedSingleOp();
  }
  char msg[100];
  std::snprintf(msg, sizeof(msg), "(%d of %d found)", found, num_);
  thread->stats.AddMessage(msg);
}
```

Q1. Which user key-value interface does each benchmark use?

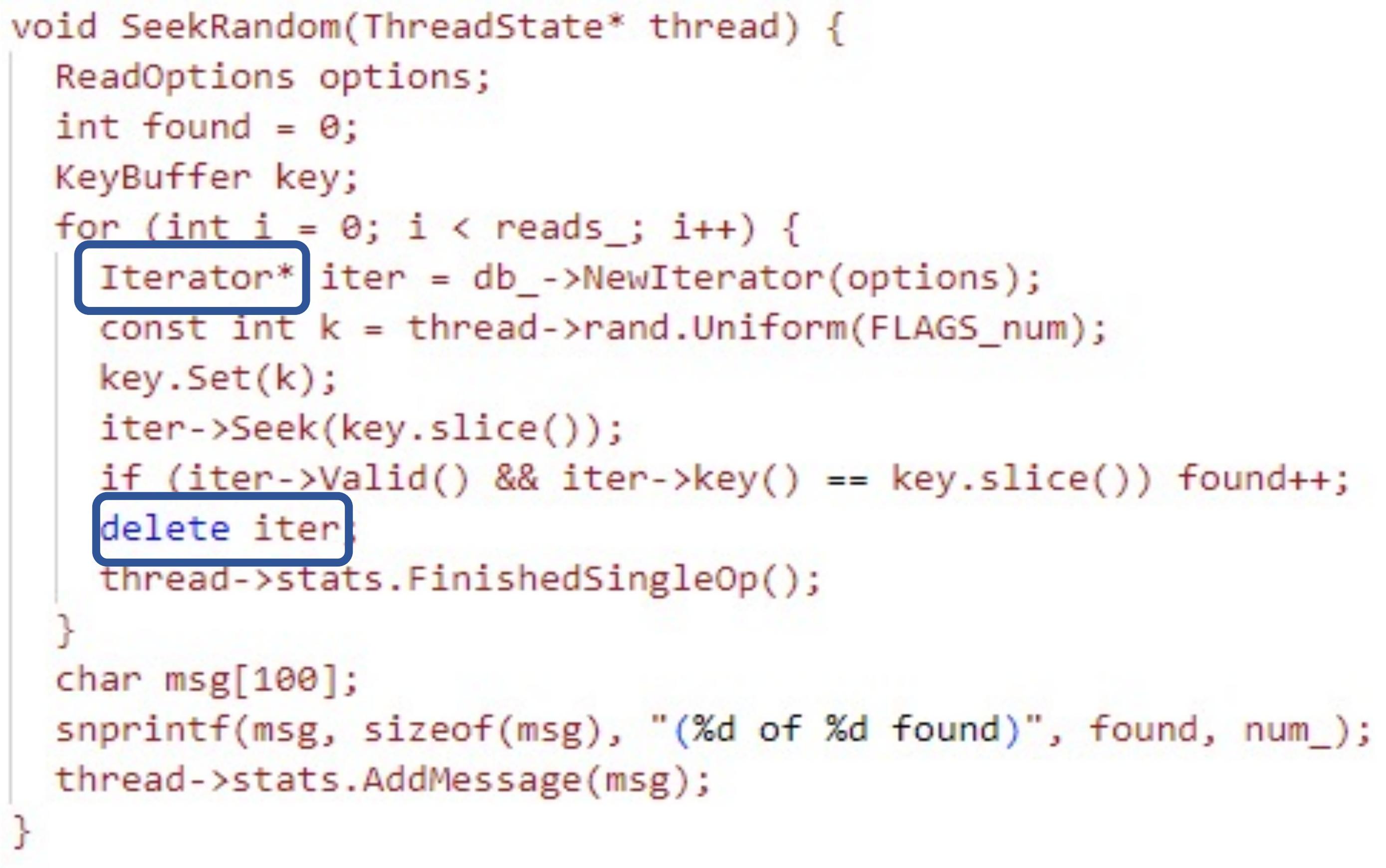
Readrandom
= get (random Key) * N

Random read



Q1. Which user key-value interface does each benchmark use?

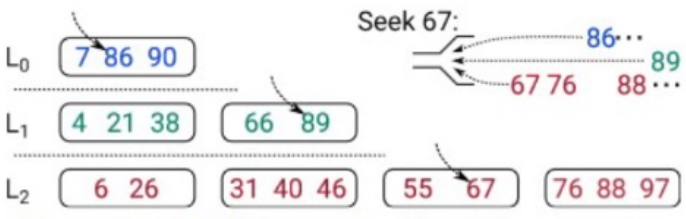
$ ./db_bench --benchmarks=“seekrandom" --use_existing_db=1



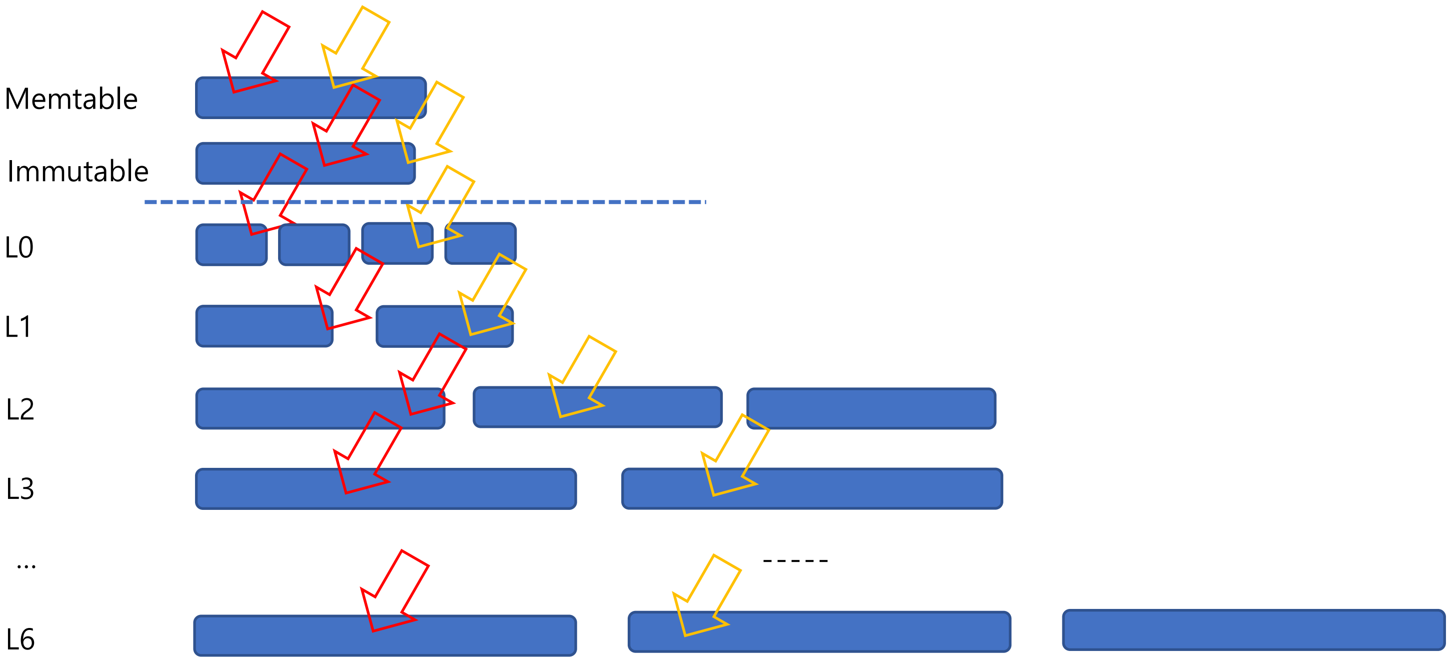
```
void SeekRandom(ThreadState* thread) {
  ReadOptions options;
  int found = 0;
  KeyBuffer key;
  for (int i = 0; i < reads_; i++) {
    Iterator* iter = db_->NewIterator(options);
    const int k = thread->rand.Uniform(FLAGS_num);
    key.Set(k);
    iter->Seek(key.slice());
    if (iter->Valid() && iter->key() == key.slice()) found++;
    delete iter;
    thread->stats.FinishedSingleOp();
  }
  char msg[100];
  snprintf(msg, sizeof(msg), "(%d of %d found)", found, num_);
  thread->stats.AddMessage(msg);
}
```

Q1. Which user key-value interface does each benchmark use?

Seekrandom
=seek (random Key) * N

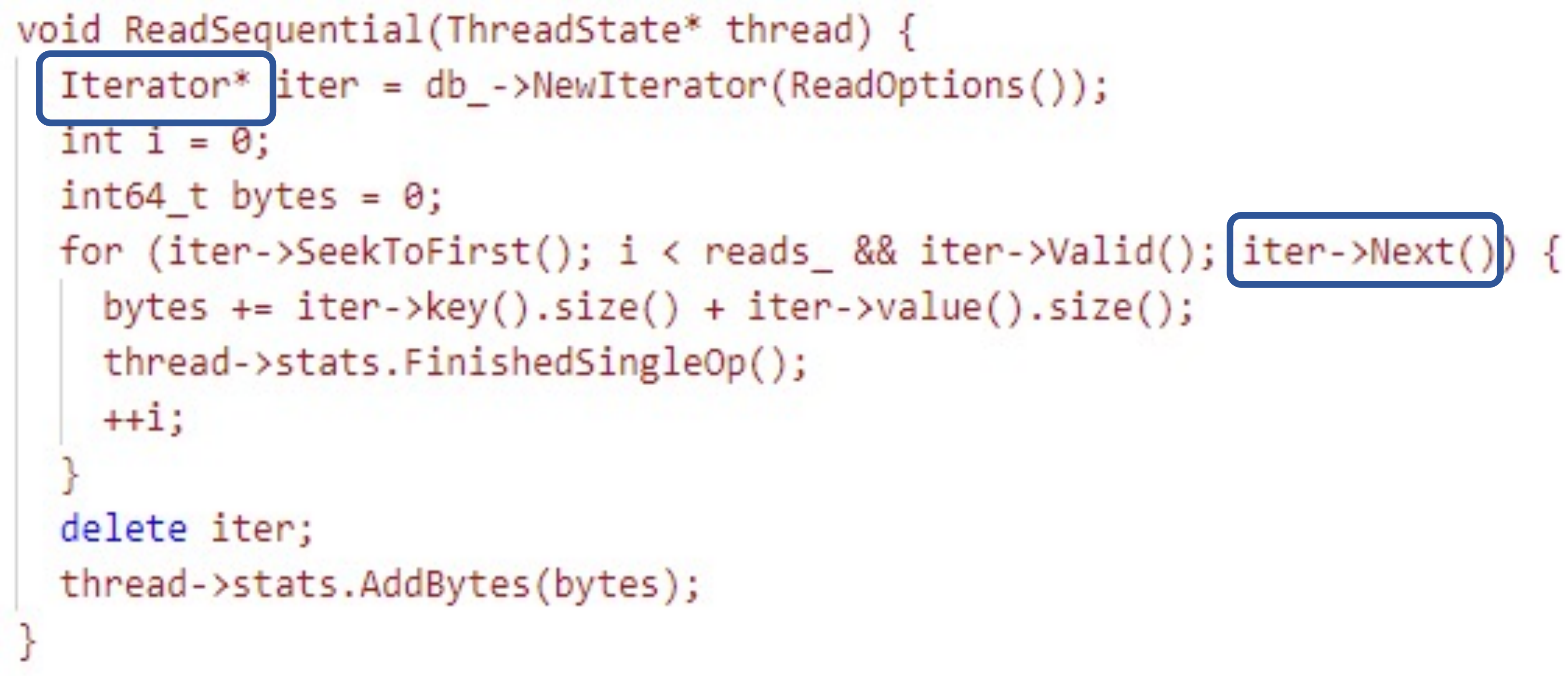


Random read



Q1. Which user key-value interface does each benchmark use?

```
$ ./db_bench --benchmarks="readseq" --use_existing_db=1
```

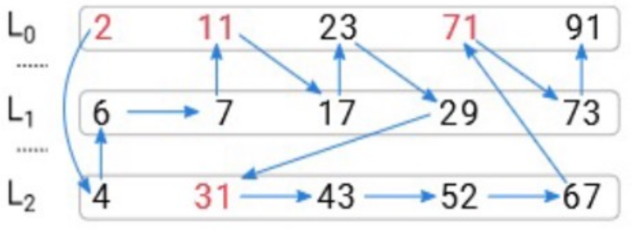


```
void ReadSequential(ThreadState* thread) {
  Iterator* iter = db_->NewIterator(ReadOptions());
  int i = 0;
  int64_t bytes = 0;
  for (iter->SeekToFirst(); i < reads_ && iter->Valid(); iter->Next()) {
    bytes += iter->key().size() + iter->value().size();
    thread->stats.FinishedSingleOp();
    ++i;
  }
  delete iter;
  thread->stats.AddBytes(bytes);
}
```

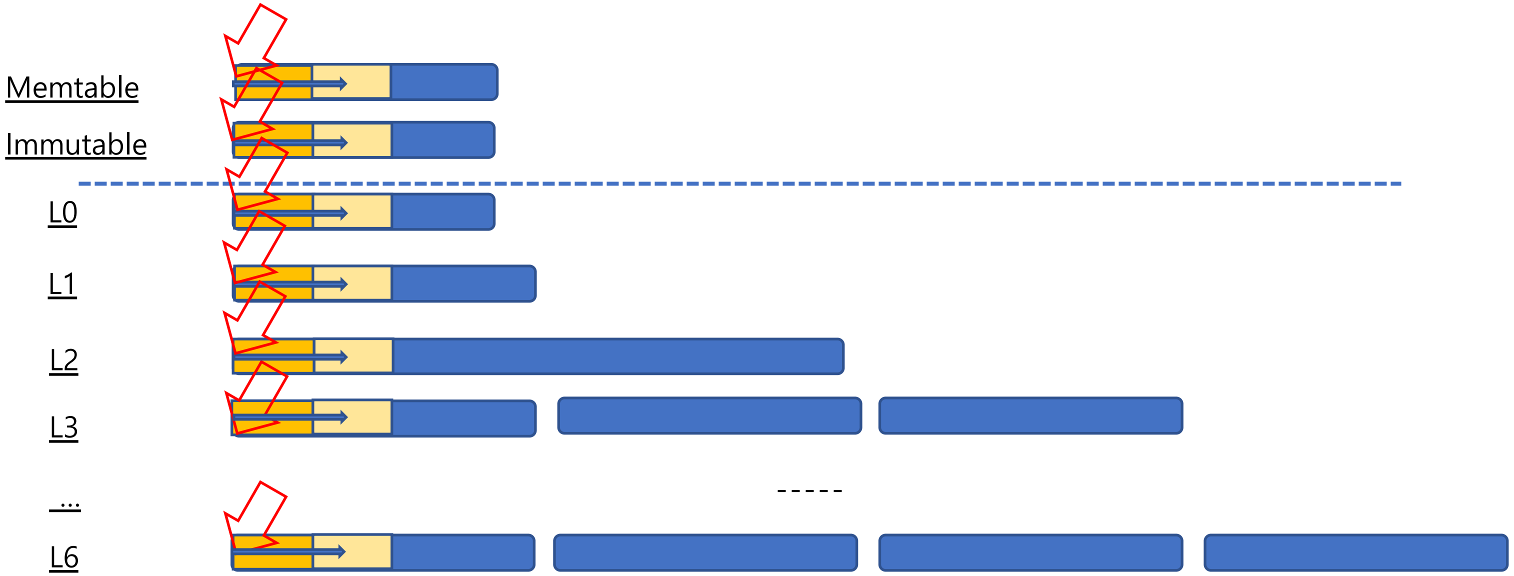
Q1. Which user key-value interface does each benchmark use?

Readseq
= Iterator->SeekToFirst()
Iterator->next() *N

Sequential read



Wenshao Zhong, REMIX: Efficient Range Query for LSM-trees, FAST '21



Memtable
Immutable
L0
L1
L2
L3
...
L6

Q2. Compare latency of each benchmark and explain why.

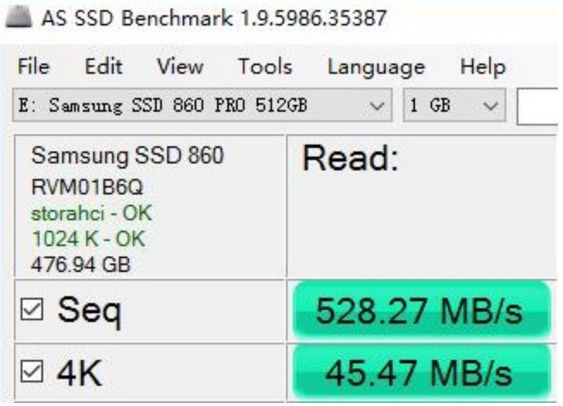
	read seq	read random	seek random
User interface	Iterator next()	Get()	Iterator seek()
I/O	sequential read	random read	random read
Latency	0.102 micros/op	1.482 micros/op	1.742 micros/op

Q2. Compare latency of each benchmark and explain why.

Readrandom and seekrandom are random reads.

- ✓ seekrandom needs to query the highest level every time.
- ✓ seekrandom will be slower than readrandom.

Readseq is read sequentially



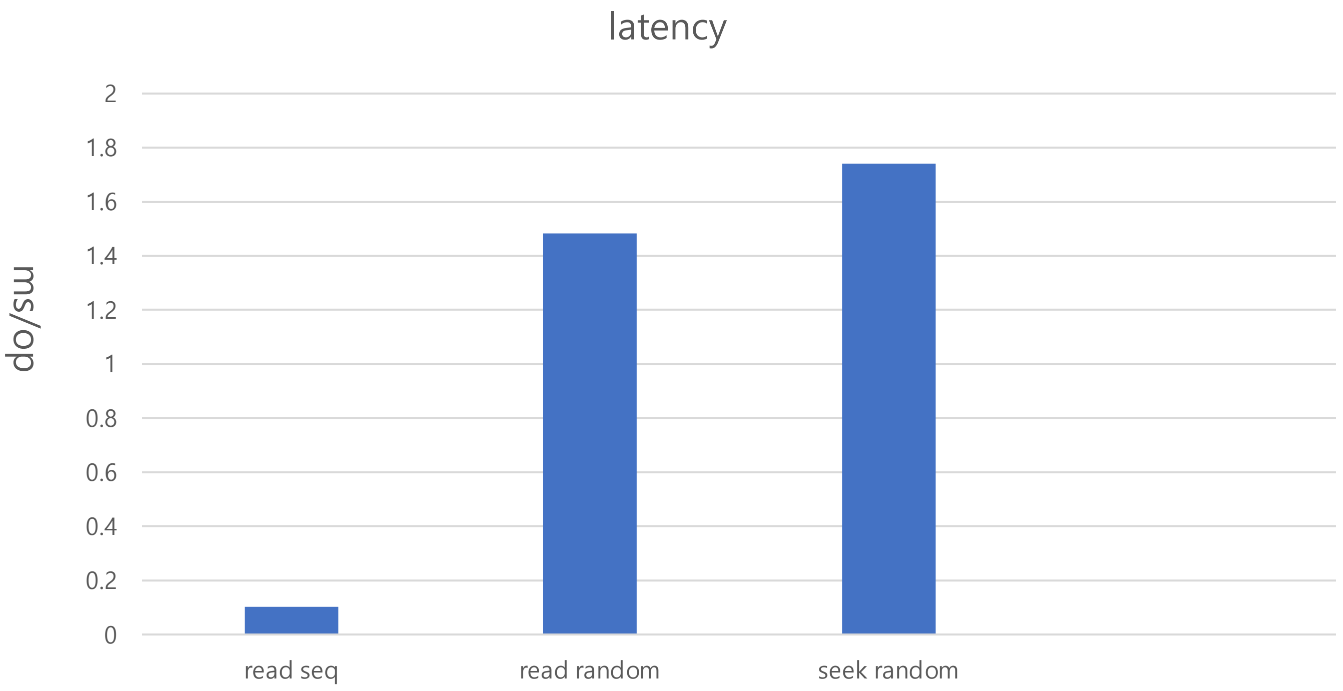
AS SSD Benchmark 1.9.5986.35387

File Edit View Tools Language Help

E: Samsung SSD 860 PRO 512GB | 1 GB

Samsung SSD 860
RVM01B6Q
storahci - OK
1024 K - OK
476.94 GB

	Read:
☑ Seq	528.27 MB/s
☑ 4K	45.47 MB/s



Q&A

감사합니다

Thank you~!