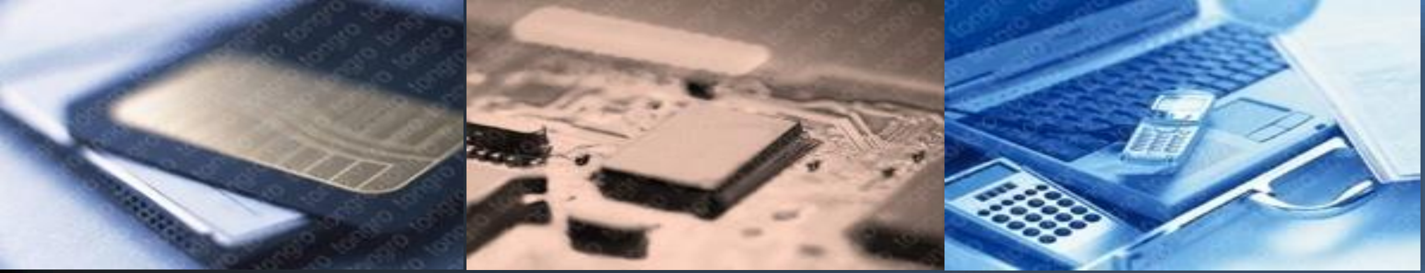
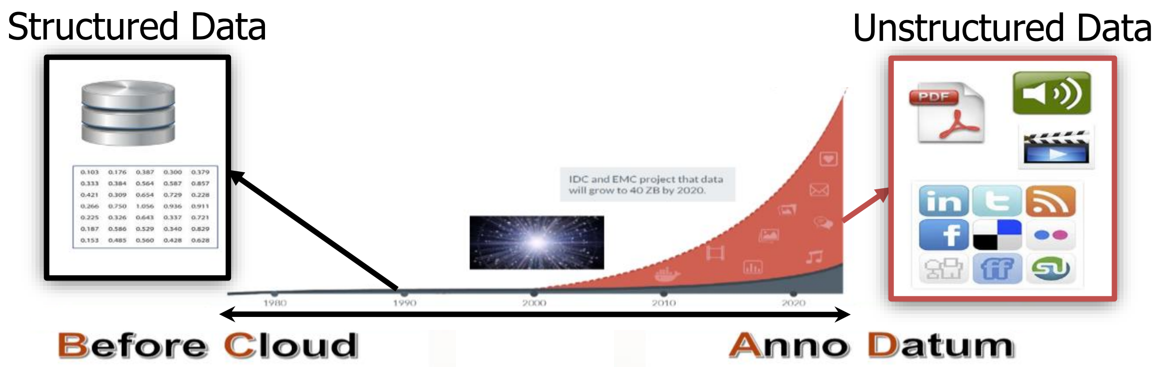


Install LevelDB

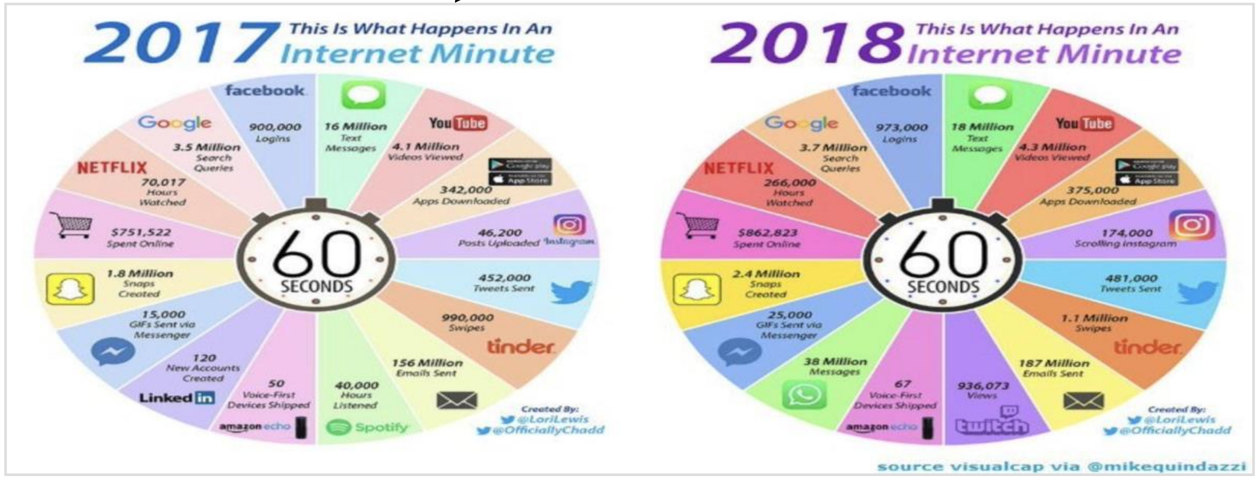
오명훈
snt2426@gmail.com

Key-Value Store

- Data Growth
  - Unstructured Data는 미리 정의된 데이터 모델이 없거나, 정의된 방식으로 정리되지 않음



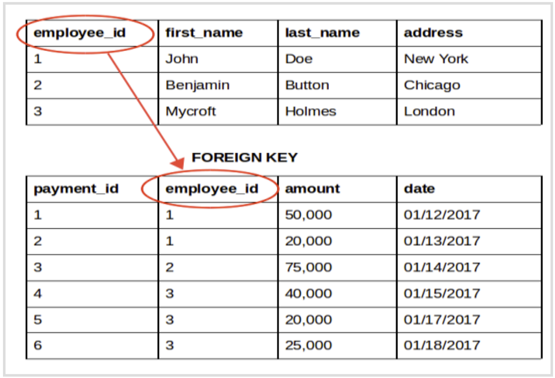
< Unstructured Data의 증가 >



< What happens in an Internet Minute >

Key-Value Store

- Key-Value Stores are Common
  - Unstructured Data의 증가로 RDBMS 모델(SQL DB)의 대안으로 Key-Value Store 모델(NoSQL DB)을 활용

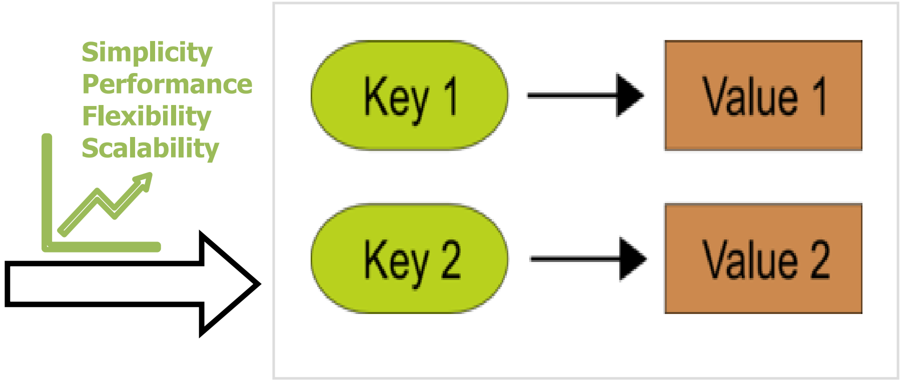


employee_id	first_name	last_name	address
1	John	Doe	New York
2	Benjamin	Button	Chicago
3	Mycroft	Holmes	London

FOREIGN KEY

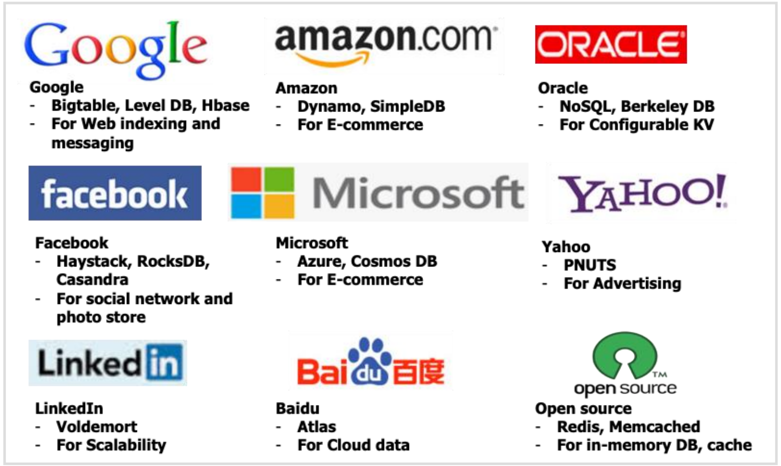
payment_id	employee_id	amount	date
1	1	50,000	01/12/2017
2	1	20,000	01/13/2017
3	2	75,000	01/14/2017
4	3	40,000	01/15/2017
5	3	20,000	01/17/2017
6	3	25,000	01/18/2017

< RDBMS: Employees and Payments Tables >



< Key-Value Store >

  - 현재 다양한 기업에서 용도에 따라 Key-Value Store를 활용



Google
- Bigtable, Level DB, Hbase
- For Web indexing and messaging

Amazon
- Dynamo, SimpleDB
- For E-commerce

Oracle
- NoSQL, Berkeley DB
- For Configurable KV

Facebook
- Haystack, RocksDB, Casandra
- For social network and photo store

Microsoft
- Azure, Cosmos DB
- For E-commerce

Yahoo
- PNUTS
- For Advertising

LinkedIn
- Voldemort
- For Scalability

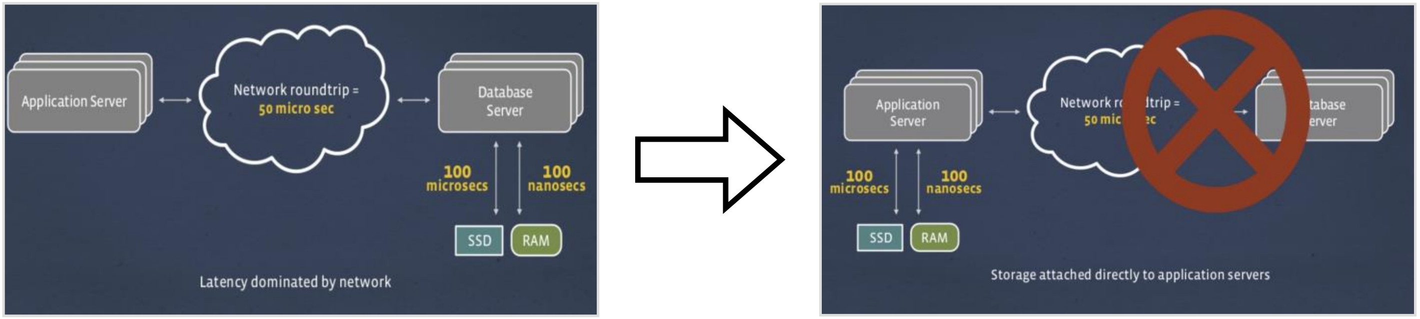
Baidu
- Atlas
- For Cloud data

Open source
- Redis, Memcached
- For in-memory DB, cache

< Prevalence of Key-Value Store >

Key-Value Store

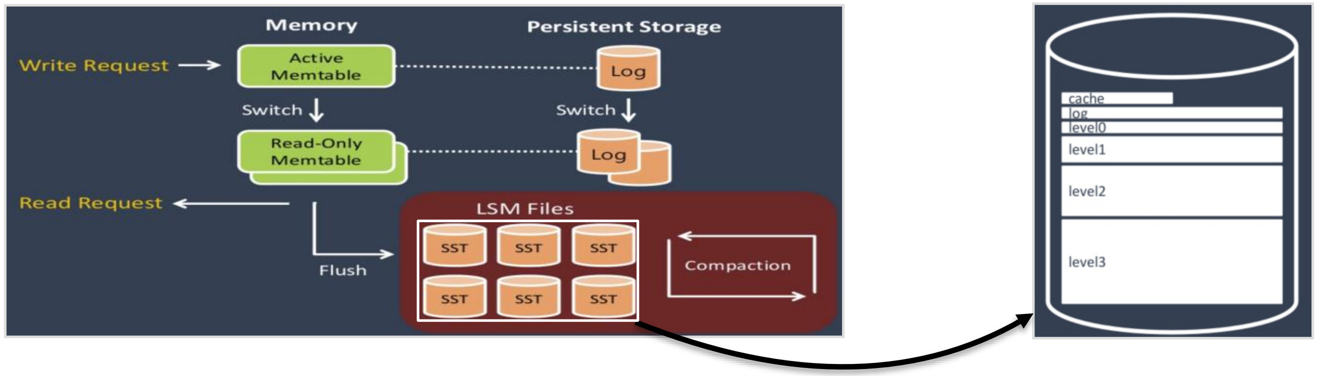
- LevelDB, RocksDB
  - Google, Facebook
  - Most popular embedded NoSQL database



Application Server
Network roundtrip = 50 micro sec
Database Server
100 microsecs
100 nanosecs
SSD
RAM
Latency dominated by network
Storage attached directly to application servers

< Embedded database >

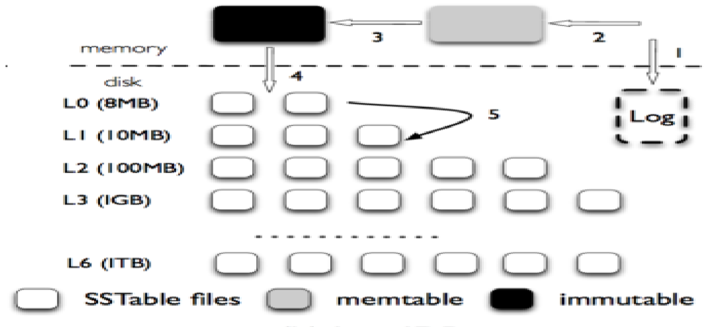
  - Persistent Key-Value Store
  - Optimized for storage (Disk, SSD)



< Database Architecture >

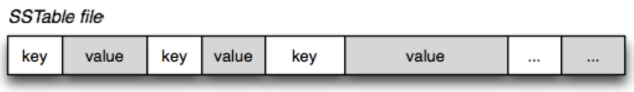
Key-Value Store

- DB Implementation
  - Log Structured Merge tree architecture

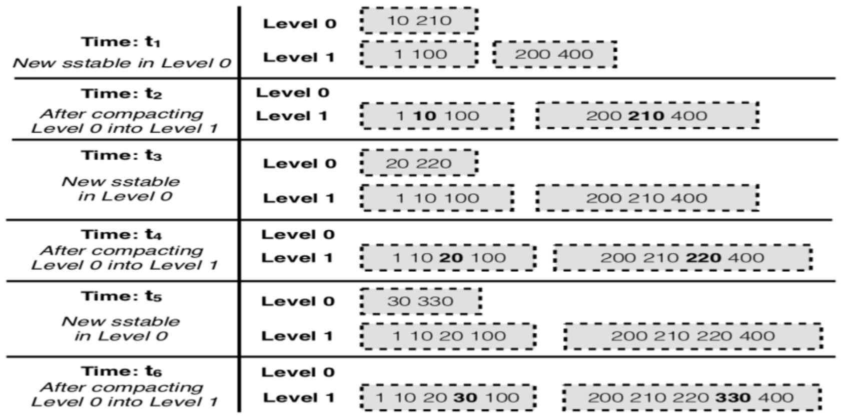


(b) LevelDB

  - Log structured
    - Only append, Only Sequential write
  - Overlapped Data
  - Merge (Compaction)



< Sorted String Table file>



Time: t_1 *New sstable in Level 0*	**Level 0**	10 210	
	Level 1	1 100	200 400
Time: t_2 *After compacting Level 0 into Level 1*	**Level 0**		
	Level 1	1 **10** 100	200 **210** 400
Time: t_3 *New sstable in Level 0*	**Level 0**	20 220	
	Level 1	1 10 100	200 210 400
Time: t_4 *After compacting Level 0 into Level 1*	**Level 0**		
	Level 1	1 10 **20** 100	200 210 **220** 400
Time: t_5 *New sstable in Level 0*	**Level 0**	30 330	
	Level 1	1 10 20 100	200 210 220 400
Time: t_6 *After compacting Level 0 into Level 1*	**Level 0**		
	Level 1	1 10 20 **30** 100	200 210 220 **330** 400

< L0 -> L1 Compaction >

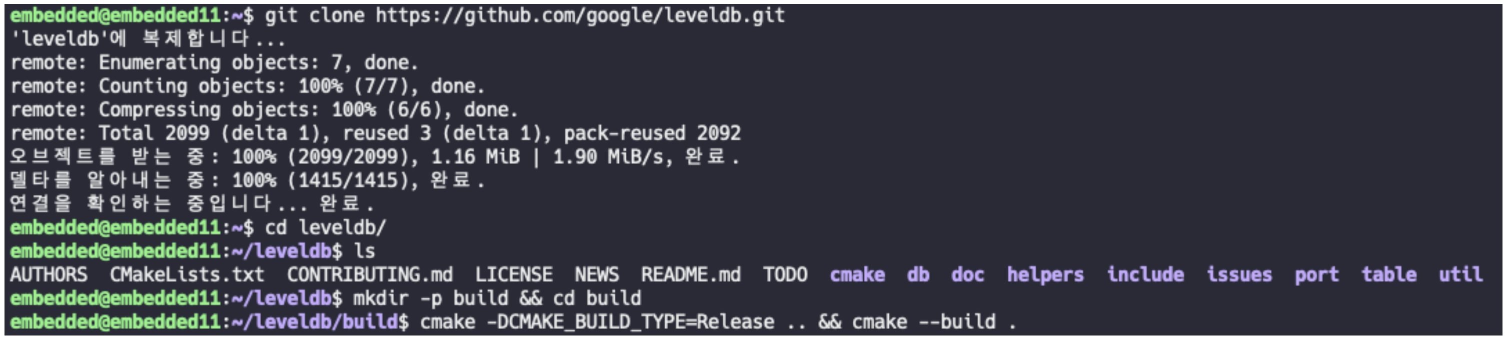
Key-Value Store

- LevelDB
  - http://leveldb.org

LevelDB

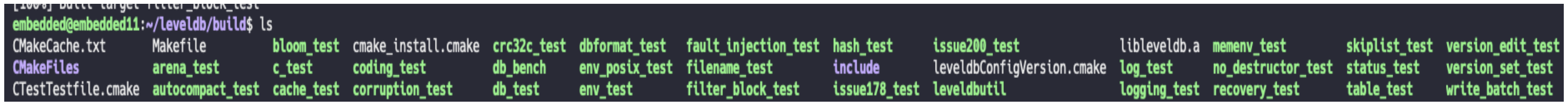
A light-weight, single-purpose library for persistence with bindings to many platforms.

  - LevelDB build


```
embedded@embedded11:~$ git clone https://github.com/google/leveldb.git
'leveldb'에 복제합니다...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 2099 (delta 1), reused 3 (delta 1), pack-reused 2092
오브젝트를 받는 중: 100% (2099/2099), 1.16 MiB | 1.90 MiB/s, 완료.
델타를 알아내는 중: 100% (1415/1415), 완료.
연결을 확인하는 중입니다... 완료.
embedded@embedded11:~$ cd leveldb/
embedded@embedded11:~/leveldb$ ls
AUTHORS  CMakeLists.txt  CONTRIBUTING.md  LICENSE  NEWS  README.md  TODO  cmake  db  doc  helpers  include  issues  port  table  util
embedded@embedded11:~/leveldb$ mkdir -p build && cd build
embedded@embedded11:~/leveldb/build$ cmake -DCMAKE_BUILD_TYPE=Release .. && cmake --build .
```

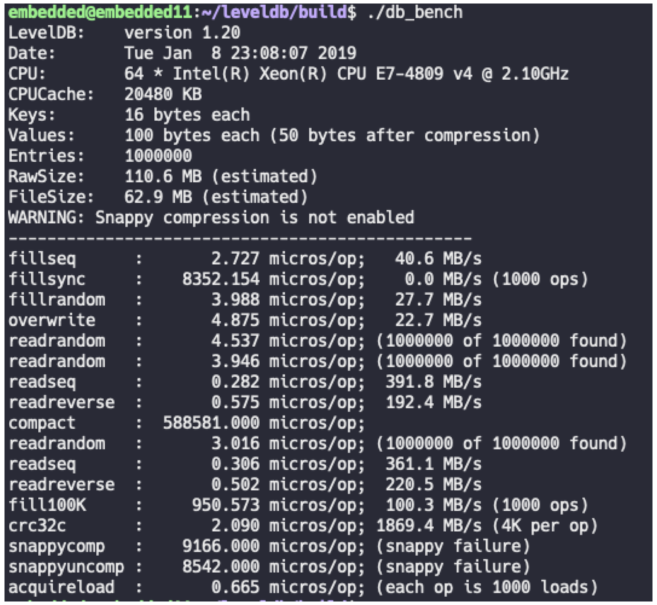
Key-Value Store

- LevelDB

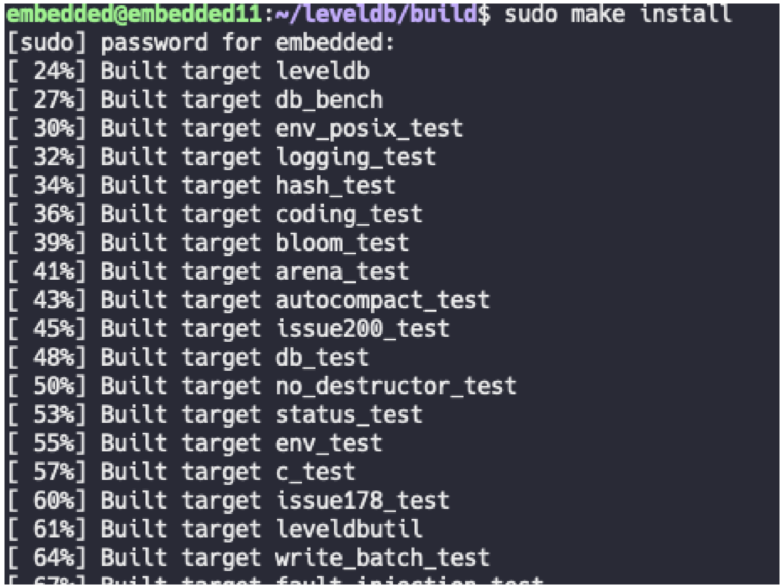


```
[100%] Built target filter_block_test
embedded@embedded11:~/leveldb/build$ ls
CMakeCache.txt      Makefile          bloom_test  cmake_install.cmake  crc32c_test  dbformat_test  fault_injection_test  hash_test     issue200_test               libleveldb.a  memenv_test          skiplist_test  version_edit_test
CMakeFiles          arena_test        c_test      coding_test          db_bench     env_posix_test filename_test         include       leveldbConfigVersion.cmake  log_test      no_destructor_test   status_test    version_set_test
CTestTestfile.cmake autocompact_test  cache_test  corruption_test      db_test      env_test       filter_block_test     issue178_test leveldbutil                 logging_test  recovery_test        table_test     write_batch_test
```

LevelDB install

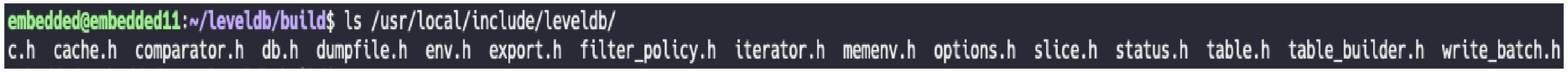


```
embedded@embedded11:~/leveldb/build$ ./db_bench
LevelDB:    version 1.20
Date:       Tue Jan  8 23:08:07 2019
CPU:        64 * Intel(R) Xeon(R) CPU E7-4809 v4 @ 2.10GHz
CPUCache:   20480 KB
Keys:       16 bytes each
Values:     100 bytes each (50 bytes after compression)
Entries:    1000000
RawSize:    110.6 MB (estimated)
FileSize:   62.9 MB (estimated)
WARNING: Snappy compression is not enabled
------------------------------------------------
fillseq      :       2.727 micros/op;   40.6 MB/s
fillsync     :    8352.154 micros/op;    0.0 MB/s (1000 ops)
fillrandom   :       3.988 micros/op;   27.7 MB/s
overwrite    :       4.875 micros/op;   22.7 MB/s
readrandom   :       4.537 micros/op; (1000000 of 1000000 found)
readrandom   :       3.946 micros/op; (1000000 of 1000000 found)
readseq      :       0.282 micros/op;  391.8 MB/s
readreverse  :       0.575 micros/op;  192.4 MB/s
compact      :  588581.000 micros/op;
readrandom   :       3.016 micros/op; (1000000 of 1000000 found)
readseq      :       0.306 micros/op;  361.1 MB/s
readreverse  :       0.502 micros/op;  220.5 MB/s
fill100K     :     950.573 micros/op;  100.3 MB/s (1000 ops)
crc32c       :       2.090 micros/op; 1869.4 MB/s (4K per op)
snappycomp   :    9166.000 micros/op; (snappy failure)
snappyuncomp :    8542.000 micros/op; (snappy failure)
acquireload  :       0.665 micros/op; (each op is 1000 loads)
```



```
embedded@embedded11:~/leveldb/build$ sudo make install
[sudo] password for embedded:
[ 24%] Built target leveldb
[ 27%] Built target db_bench
[ 30%] Built target env_posix_test
[ 32%] Built target logging_test
[ 34%] Built target hash_test
[ 36%] Built target coding_test
[ 39%] Built target bloom_test
[ 41%] Built target arena_test
[ 43%] Built target autocompact_test
[ 45%] Built target issue200_test
[ 48%] Built target db_test
[ 50%] Built target no_destructor_test
[ 53%] Built target status_test
[ 55%] Built target env_test
[ 57%] Built target c_test
[ 60%] Built target issue178_test
[ 61%] Built target leveldbutil
[ 64%] Built target write_batch_test
[ 67%] Built target fault_injection_test
```

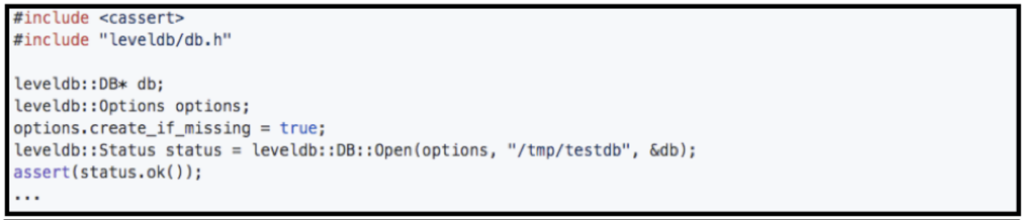
LevelDB install2



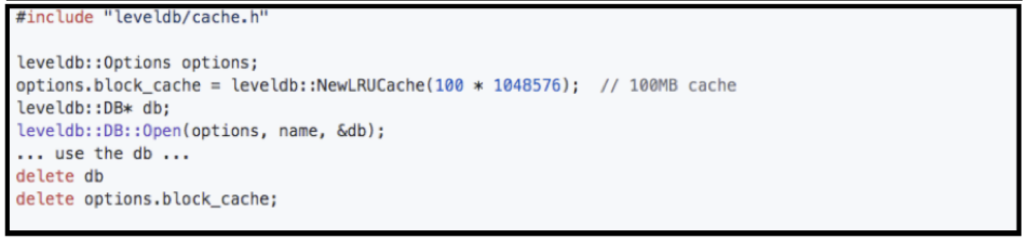
```
embedded@embedded11:~/leveldb/build$ ls /usr/local/include/leveldb/
c.h  cache.h  comparator.h  db.h  dumpfile.h  env.h  export.h  filter_policy.h  iterator.h  memenv.h  options.h  slice.h  status.h  table.h  table_builder.h  write_batch.h
```

Key-Value Store

- LevelDB Library
  - DB Open

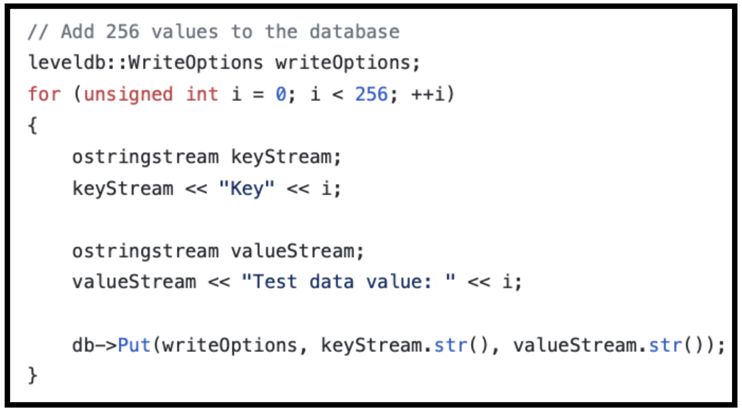


```
#include <cassert>
#include "leveldb/db.h"

leveldb::DB* db;
leveldb::Options options;
options.create_if_missing = true;
leveldb::Status status = leveldb::DB::Open(options, "/tmp/testdb", &db);
assert(status.ok());
...
```



```
#include "leveldb/cache.h"

leveldb::Options options;
options.block_cache = leveldb::NewLRUCache(100 * 1048576);  // 100MB cache
leveldb::DB* db;
leveldb::DB::Open(options, name, &db);
... use the db ...
delete db
delete options.block_cache;
```

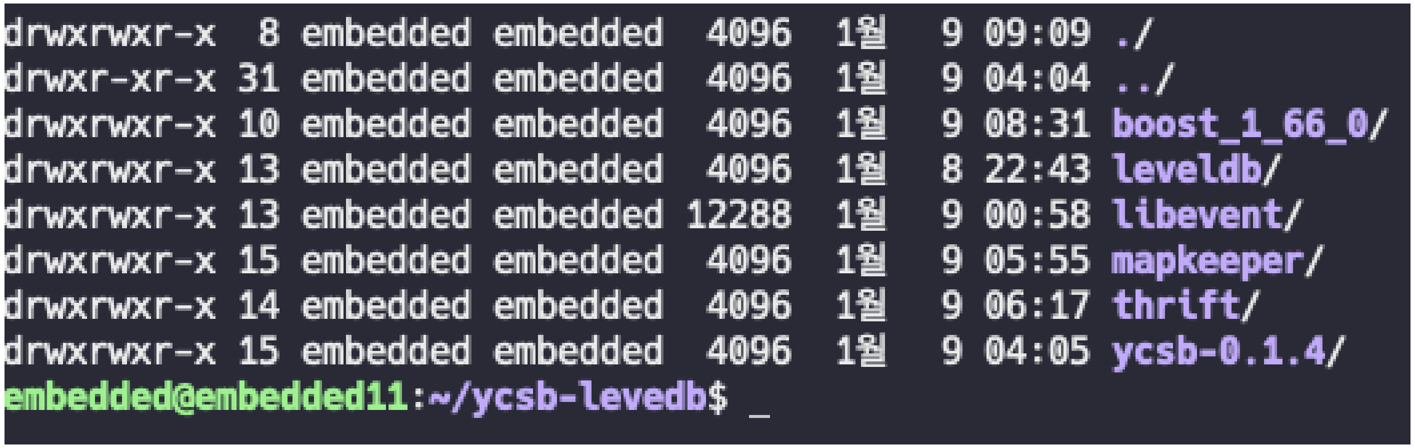
  - Data Put



```
// Add 256 values to the database
leveldb::WriteOptions writeOptions;
for (unsigned int i = 0; i < 256; ++i)
{
    ostringstream keyStream;
    keyStream << "Key" << i;

    ostringstream valueStream;
    valueStream << "Test data value: " << i;

    db->Put(writeOptions, keyStream.str(), valueStream.str());
}
```

Key-Value Store

- YCSB (Yahoo! Cloud Serving Benchmark)



```
drwxrwxr-x  8 embedded embedded  4096  1월   9 09:09 ./
drwxr-xr-x 31 embedded embedded  4096  1월   9 04:04 ../
drwxrwxr-x 10 embedded embedded  4096  1월   9 08:31 boost_1_66_0/
drwxrwxr-x 13 embedded embedded  4096  1월   8 22:43 leveldb/
drwxrwxr-x 13 embedded embedded 12288  1월   9 00:58 libevent/
drwxrwxr-x 15 embedded embedded  4096  1월   9 05:55 mapkeeper/
drwxrwxr-x 14 embedded embedded  4096  1월   9 06:17 thrift/
drwxrwxr-x 15 embedded embedded  4096  1월   9 04:05 ycsb-0.1.4/
embedded@embedded11:~/ycsb-levedb$ _
```

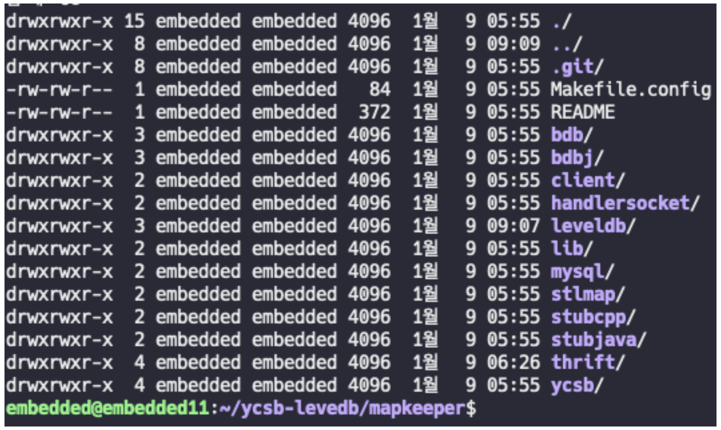
- cd libevent
- mkdir build && cd build && cmake ..
- make && sudo make install

- cd boost_1_66_0
- ./bootstrap.sh —prefix=/usr/local && sudo ./b2 install

- cd thrift
- ./configure --prefix=/usr/local --with-cpp --with-boost-libdir=/usr/local/lib
- Make && sudo make install

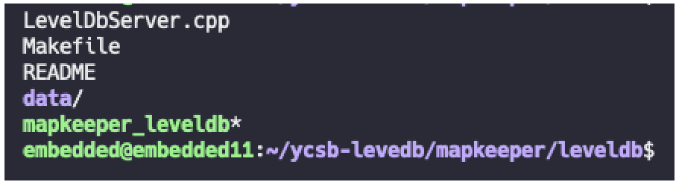
Key-Value Store

- YCSB (Yahoo! Cloud Serving Benchmark)



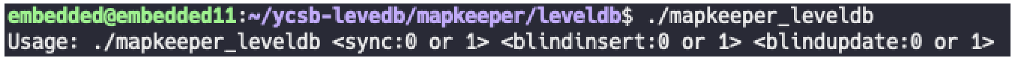
```
drwxrwxr-x 15 embedded embedded 4096  1월   9 05:55 ./
drwxrwxr-x  8 embedded embedded 4096  1월   9 09:09 ../
drwxrwxr-x  8 embedded embedded 4096  1월   9 05:55 .git/
-rw-rw-r--  1 embedded embedded   84  1월   9 05:55 Makefile.config
-rw-rw-r--  1 embedded embedded  372  1월   9 05:55 README
drwxrwxr-x  3 embedded embedded 4096  1월   9 05:55 bdb/
drwxrwxr-x  3 embedded embedded 4096  1월   9 05:55 bdbj/
drwxrwxr-x  2 embedded embedded 4096  1월   9 05:55 client/
drwxrwxr-x  2 embedded embedded 4096  1월   9 05:55 handlersocket/
drwxrwxr-x  3 embedded embedded 4096  1월   9 09:07 leveldb/
drwxrwxr-x  2 embedded embedded 4096  1월   9 05:55 lib/
drwxrwxr-x  2 embedded embedded 4096  1월   9 05:55 mysql/
drwxrwxr-x  2 embedded embedded 4096  1월   9 05:55 stlmap/
drwxrwxr-x  2 embedded embedded 4096  1월   9 05:55 stubcpp/
drwxrwxr-x  2 embedded embedded 4096  1월   9 05:55 stubjava/
drwxrwxr-x  4 embedded embedded 4096  1월   9 06:26 thrift/
drwxrwxr-x  4 embedded embedded 4096  1월   9 05:55 ycsb/
embedded@embedded11:~/ycsb-levedb/mapkeeper$
```

- cd mapkeeper
- cd thrift && make
- cd .. && cd leveldb
- make



```
LevelDbServer.cpp
Makefile
README
data/
mapkeeper_leveldb*
embedded@embedded11:~/ycsb-levedb/mapkeeper/leveldb$
```

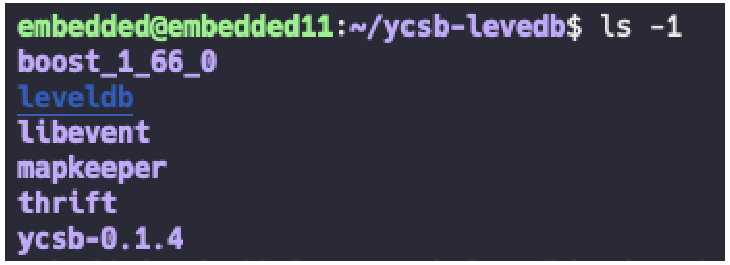
- ./mapkeeper_leveldb 0 0 0



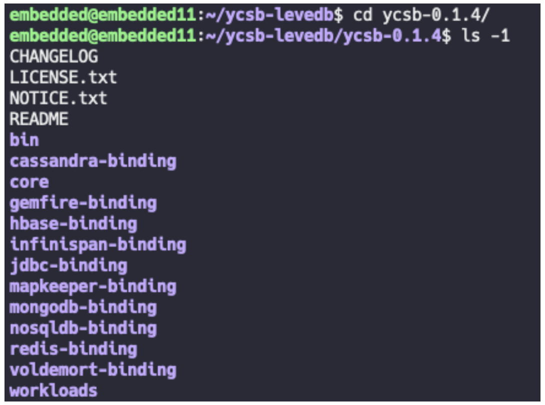
```
embedded@embedded11:~/ycsb-levedb/mapkeeper/leveldb$ ./mapkeeper_leveldb
Usage:  ./mapkeeper_leveldb <sync:0 or 1> <blindinsert:0 or 1> <blindupdate:0 or 1>
```

Key-Value Store

- YCSB (Yahoo! Cloud Serving Benchmark)

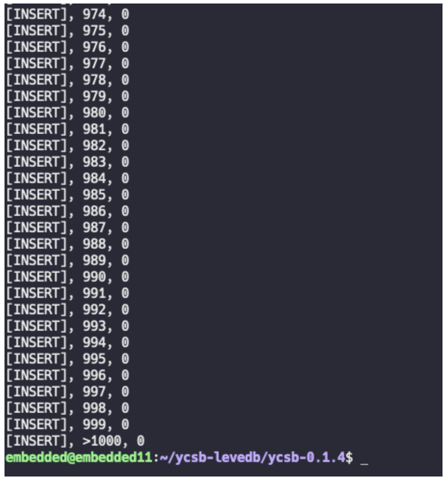


```
embedded@embedded11:~/ycsb-levedb$ ls -1
boost_1_66_0
leveldb
libevent
mapkeeper
thrift
ycsb-0.1.4
```

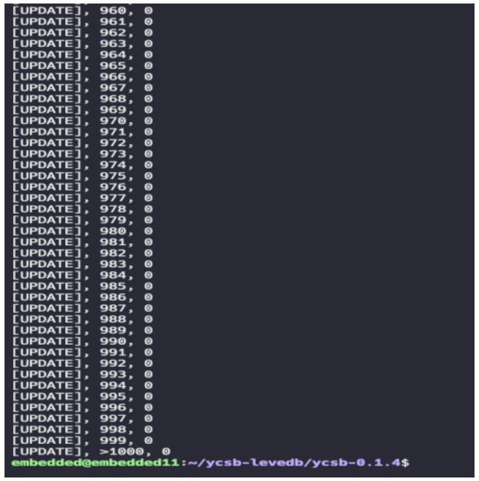


```
embedded@embedded11:~/ycsb-levedb$ cd ycsb-0.1.4/
embedded@embedded11:~/ycsb-levedb/ycsb-0.1.4$ ls -1
CHANGELOG
LICENSE.txt
NOTICE.txt
README
bin
cassandra-binding
core
gemfire-binding
hbase-binding
infinispan-binding
jdbc-binding
mapkeeper-binding
mongodb-binding
nosqldb-binding
redis-binding
voldemort-binding
workloads
```

- cd ycsb-0.1.4
- ./bin/ycsb load mapkeeper -s -P workloads/workloada
- ./bin/ycsb run mapkeeper -s -P workloads/workloada



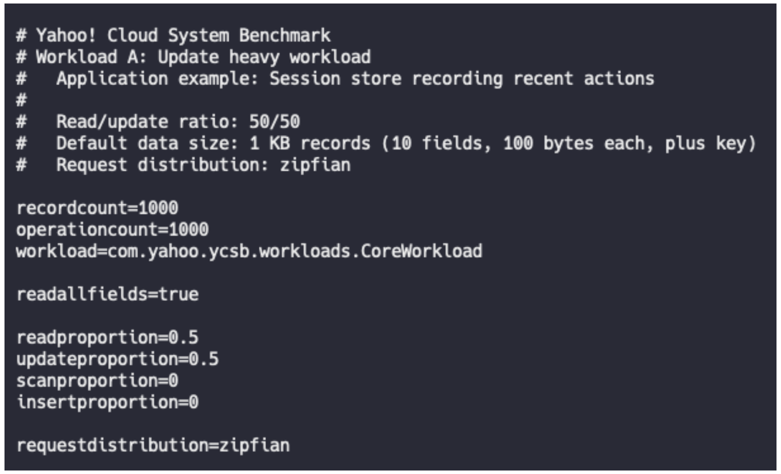
```
[INSERT], 974, 0
[INSERT], 975, 0
[INSERT], 976, 0
[INSERT], 977, 0
[INSERT], 978, 0
[INSERT], 979, 0
[INSERT], 980, 0
[INSERT], 981, 0
[INSERT], 982, 0
[INSERT], 983, 0
[INSERT], 984, 0
[INSERT], 985, 0
[INSERT], 986, 0
[INSERT], 987, 0
[INSERT], 988, 0
[INSERT], 989, 0
[INSERT], 990, 0
[INSERT], 991, 0
[INSERT], 992, 0
[INSERT], 993, 0
[INSERT], 994, 0
[INSERT], 995, 0
[INSERT], 996, 0
[INSERT], 997, 0
[INSERT], 998, 0
[INSERT], 999, 0
[INSERT], >1000, 0
embedded@embedded11:~/ycsb-levedb/ycsb-0.1.4$ _
```



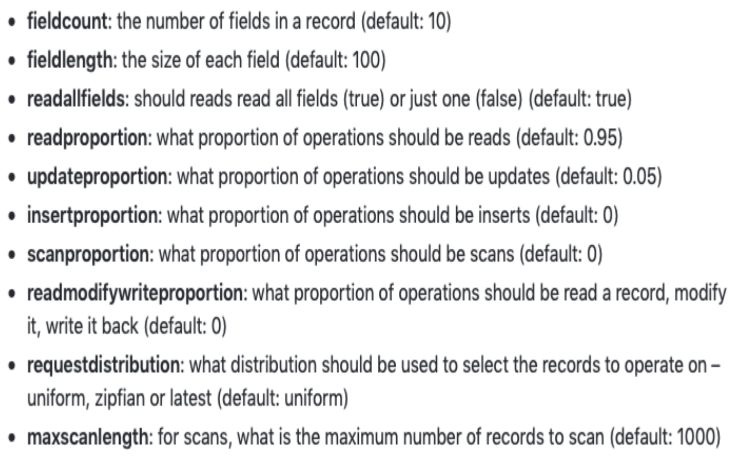
```
[UPDATE], 960, 0
[UPDATE], 961, 0
[UPDATE], 962, 0
[UPDATE], 963, 0
[UPDATE], 964, 0
[UPDATE], 965, 0
[UPDATE], 966, 0
[UPDATE], 967, 0
[UPDATE], 968, 0
[UPDATE], 969, 0
[UPDATE], 970, 0
[UPDATE], 971, 0
[UPDATE], 972, 0
[UPDATE], 973, 0
[UPDATE], 974, 0
[UPDATE], 975, 0
[UPDATE], 976, 0
[UPDATE], 977, 0
[UPDATE], 978, 0
[UPDATE], 979, 0
[UPDATE], 980, 0
[UPDATE], 981, 0
[UPDATE], 982, 0
[UPDATE], 983, 0
[UPDATE], 984, 0
[UPDATE], 985, 0
[UPDATE], 986, 0
[UPDATE], 987, 0
[UPDATE], 988, 0
[UPDATE], 989, 0
[UPDATE], 990, 0
[UPDATE], 991, 0
[UPDATE], 992, 0
[UPDATE], 993, 0
[UPDATE], 994, 0
[UPDATE], 995, 0
[UPDATE], 996, 0
[UPDATE], 997, 0
[UPDATE], 998, 0
[UPDATE], 999, 0
[UPDATE], >1000, 0
embedded@embedded11:~/ycsb-levedb/ycsb-0.1.4$
```

Key-Value Store

- YCSB (Yahoo! Cloud Serving Benchmark)



```
# Yahoo! Cloud System Benchmark
# Workload A: Update heavy workload
#   Application example: Session store recording recent actions
#
#   Read/update ratio: 50/50
#   Default data size: 1 KB records (10 fields, 100 bytes each, plus key)
#   Request distribution: zipfian

recordcount=1000
operationcount=1000
workload=com.yahoo.ycsb.workloads.CoreWorkload

readallfields=true

readproportion=0.5
updateproportion=0.5
scanproportion=0
insertproportion=0

requestdistribution=zipfian
```



- **fieldcount**: the number of fields in a record (default: 10)
- **fieldlength**: the size of each field (default: 100)
- **readallfields**: should reads read all fields (true) or just one (false) (default: true)
- **readproportion**: what proportion of operations should be reads (default: 0.95)
- **updateproportion**: what proportion of operations should be updates (default: 0.05)
- **insertproportion**: what proportion of operations should be inserts (default: 0)
- **scanproportion**: what proportion of operations should be scans (default: 0)
- **readmodifywriteproportion**: what proportion of operations should be read a record, modify it, write it back (default: 0)
- **requestdistribution**: what distribution should be used to select the records to operate on – uniform, zipfian or latest (default: uniform)
- **maxscanlength**: for scans, what is the maximum number of records to scan (default: 1000)

- vi workload/workloada
  - Workload A: Update heavy workload
  - Workload B: Read mostly workload
  - Workload C: Read only
  - Workload D: Read latest workload
  - Workload E: Short ranges
  - Workload F: Read-modify-write
- Implementing New Workloads
  - https://github.com/brianfrankcooper/YCSB/wiki

Q&A