Scalable Data Systems: Introduction of RocksDB

CZ4123

Contents

- What is RocksDB
- Why RocksDB
- Some Demos

1. What is RocksDB

RocksDB is an **embeddable** persistent key-value store based on **LSM-tree** that provides **key-value store** and **read-write functions**.

1. What is RocksDB

RocksDB is built upon Google's **LevelDB** and is developed and maintained by **Meta**. Over the past years, it has become a standard for embeddable key-value stores.

Today RocksDB runs in production at Meta, Microsoft, Netflix, Uber.

2. Why RocksDB

- **High Performance**

  RocksDB uses a log structured engine, written in C++, for maximum performance.
- **Optimized for Fast Storage**

  RocksDB is optimized for fast, low latency storage such as high-speed disk drives.
- **Adaptable**

  RocksDB is adaptable to different workloads.
- **Basic and Advanced Database Operations**

  RocksDB provides basic operations such as open and close a database, read, write to more advanced operations such as merging and compaction filter.

1. Prerequisite

1. Linux Operating System (Ubuntu 22.04)
2. Compression libraries
    - zlib - a library for data compression
    - bzip2 - a library for data compression
    - lz4 - a library for extremely fast data compression
    - snappy - a library for fast data compression
    - zstandard - Fast real-time compression algorithm

    ```
    sudo apt install libsnappy-dev zlib1g-dev libbz2-dev libbz2-dev liblz4-dev libzstd-dev
    ```

3. Gflags (A library that handles command line flags processing.)

    ```
    sudo apt install libgflags-dev
    ```

2. Install RocksDB

1. Install dependencies

   ```
   sudo apt install libsnappy-dev zlib1g-dev libbz2-dev libbz2-dev liblz4-dev libzstd-devlib gflags-dev
   ```

2. Clone RocksDB repository

   ```
   git clone https://github.com/facebook/rocksdb.git
   ```

   ```
   cd ./rocksdb
   ```

3. Compile RocksDB

   ```
   make static_lib
   ```

   This is a recommended option. It will compile RocksDB static library, librocksdb.a, in release mode.

3. RocksDB Demos

Demo1: write & read

demo_wr.cc: Write several keys and read out

Demo2: deletion

demo_d.cc: Write several keys and delete one key

Demo3: range read

demo_seek.cc: Read all k-v pairs

& read k-v pairs within a range

Demo4: View generated files

3. RocksDB Demos

Modify makefile

```
gedit ./examples/Makefile
```

```
demo_wr: librocksdb demo_wr.cc
        $(CXX) $(CXXFLAGS) $@.cc -o$@ ../librocksdb.a -I../include -O2 -std=c++17
        $(PLATFORM_LDFLAGS) $(PLATFORM_CXXFLAGS) $(EXEC_LDFLAGS)
demo_d: librocksdb demo_d.cc
        $(CXX) $(CXXFLAGS) $@.cc -o$@ ../librocksdb.a -I../include -O2 -std=c++17
        $(PLATFORM_LDFLAGS) $(PLATFORM_CXXFLAGS) $(EXEC_LDFLAGS)
demo_seek: librocksdb demo_seek.cc
        $(CXX) $(CXXFLAGS) $@.cc -o$@ ../librocksdb.a -I../include -O2 -std=c++17
        $(PLATFORM_LDFLAGS) $(PLATFORM_CXXFLAGS) $(EXEC_LDFLAGS)
demo_insert: librocksdb demo_insert.cc
        $(CXX) $(CXXFLAGS) $@.cc -o$@ ../librocksdb.a -I../include -O2 -std=c++17
        $(PLATFORM_LDFLAGS) $(PLATFORM_CXXFLAGS) $(EXEC_LDFLAGS)
```

3.1. Write/Read Experiment

demo_wr.cc

Open DB

> Status s = DB::Open(options, **kDBPath**, &db);
>
> assert(s.ok());

Close DB

> db->Close();

Options

> options.OptimizeLevelStyleCompaction();
>
> options.create_if_missing = true;
>
> options.level_compaction_dynamic_level_bytes = false;

```cpp
int main() {
  DB* db;
  Options options;
  // Set RocksDB option
  options.OptimizeLevelStyleCompaction();
  options.level_compaction_dynamic_level_bytes = false;
  // create the DB if it's not already present
  options.create_if_missing = true;

  // open DB
  Status s = DB::Open(options, kDBPath, &db);
  assert(s.ok());

  // Put key-value
  s = db->Put(rocksdb::WriteOptions(), "Key 0", "value 0000");
  s = db->Put(rocksdb::WriteOptions(), "Key 1", "value 0001");

  // Get key-value
  std::string opt;

  s = db->Get(rocksdb::ReadOptions(), "Key 0", &opt);
  std::cout << "Key 0: " << opt << std::endl;
  s = db->Get(rocksdb::ReadOptions(), "Key 1", &opt);
  std::cout << "Key 1: " << opt << std::endl;

  // close DB
  db->Close();
  delete db;

  return 0;
}
```

3.1. Write/Read Experiment

demo_wr.cc

Put

```
s = db->Put(key, value);
```

Get

```
s = db->Get(key, &value);
```

Run demo_wr

```
make ./demo_wr
./demo_wr
```

```cpp
int main() {
  DB* db;
  Options options;
  // Set RocksDB option
  options.OptimizeLevelStyleCompaction();
  options.level_compaction_dynamic_level_bytes = false;
  // create the DB if it's not already present
  options.create_if_missing = true;

  // open DB
  Status s = DB::Open(options, kDBPath, &db);
  assert(s.ok());

  // Put key-value
  s = db->Put(rocksdb::WriteOptions(), "Key 0", "value 0000");
  s = db->Put(rocksdb::WriteOptions(), "Key 1", "value 0001");

  // Get key-value
  std::string opt;

  s = db->Get(rocksdb::ReadOptions(), "Key 0", &opt);
  std::cout << "Key 0: " << opt << std::endl;
  s = db->Get(rocksdb::ReadOptions(), "Key 1", &opt);
  std::cout << "Key 1: " << opt << std::endl;

  // close DB
  db->Close();
  delete db;

  return 0;
}
```

3.2. Deletion Experiment

demo_d.cc

Delete

```
s = db->Delete(key);
```

Run demo_d

```
make ./demo_d
./demo_d
```

```cpp
// open DB
Status s = DB::Open(options, kDBPath, &db);
assert(s.ok());

// Put key-value
s = db->Put(rocksdb::WriteOptions(), "Key 2", "value 0002");

// Get Key 2
std::string opt;

s = db->Get(rocksdb::ReadOptions(), "Key 2", &opt);
std::cout << "Key 2: " << opt << std::endl;

// Delete Key 2
s = db->Delete(rocksdb::WriteOptions(), "Key 2");

// Get Key 2
s = db->Get(rocksdb::ReadOptions(), "Key 2", &opt);

std::cout << "Get Key 2 after Delete: " << std::endl;
if(s.ok()){
  std::cout << "Key 2: " << opt << std::endl;
}else{
  std::cout << "The value has deleted." << std::endl;
}
```

3.3. Range Read Experiment

demo_seek.cc

Iterator

> rocksdb::Iterator* it

Read ALL inserted k-v pairs

> it->SeekToFirst();
> it->Next();
> it->Valid();

```cpp
// open DB
Status s = DB::Open(options, kDBPath, &db);
assert(s.ok());

// Put key-value
s = db->Put(rocksdb::WriteOptions(), "Key 3", "value 0003");
s = db->Put(rocksdb::WriteOptions(), "Key 4", "value 0004");
s = db->Put(rocksdb::WriteOptions(), "Key 5", "value 0005");
s = db->Put(rocksdb::WriteOptions(), "Key 6", "value 0006");
s = db->Put(rocksdb::WriteOptions(), "Key 7", "value 0007");
s = db->Put(rocksdb::WriteOptions(), "Key 8", "value 0008");
s = db->Put(rocksdb::WriteOptions(), "Key 9", "value 0009");

// Read all k-v pairs
rocksdb::Iterator* it = db->NewIterator(rocksdb::ReadOptions());

std::cout << "\nRead all k-v pairs: " << std::endl;

for (it->SeekToFirst(); it->Valid(); it->Next()) {
  std::cout << it->key().ToString() << ": " << it->value().ToString() << std::endl;
}

// Check for any errors found during the scan
assert(it->status().ok());
```

3.3. Range Read Experiment

demo_seek.cc

Iterator

rocksdb::Iterator* it

Read ALL inserted k-v pairs

it->SeekToFirst();
it->Next();
it->Valid();

What is the output?

3 – 9 or 0 – 9 ?

```
// open DB
Status s = DB::Open(options, kDBPath, &db);
assert(s.ok());

// Put key-value
s = db->Put(rocksdb::WriteOptions(), "Key 3", "value 0003");
s = db->Put(rocksdb::WriteOptions(), "Key 4", "value 0004");
s = db->Put(rocksdb::WriteOptions(), "Key 5", "value 0005");
s = db->Put(rocksdb::WriteOptions(), "Key 6", "value 0006");
s = db->Put(rocksdb::WriteOptions(), "Key 7", "value 0007");
s = db->Put(rocksdb::WriteOptions(), "Key 8", "value 0008");
s = db->Put(rocksdb::WriteOptions(), "Key 9", "value 0009");

// Read all k-v pairs
rocksdb::Iterator* it = db->NewIterator(rocksdb::ReadOptions());

std::cout << "\nRead all k-v pairs: " << std::endl;

for (it->SeekToFirst(); it->Valid(); it->Next()) {
  std::cout << it->key().ToString() << ": " << it->value().ToString() << std::endl;
}

// Check for any errors found during the scan
assert(it->status().ok());
```

3.3. Range Read Experiment

demo_seek.cc

Read all inserted k-v pairs which key∈[start, limit)

- it->Seek(start);
- it->Next();
- it->key().ToString() < limit;

```
// Read all k-v pairs within a window
std::string start = "Key 4";
std::string limit = "Key 8";

std::cout << "\nRead all k-v pairs within [Key 4, Key 8]: " << std::endl;

for (it->Seek(start);
     it->Valid() && it->key().ToString() < limit;
     it->Next()) {
    std::cout << it->key().ToString() << ": " << it->value().ToString() << std::endl;
}

assert(it->status().ok());
```

3.3. Range Read Experiment

demo_seek.cc

Read all inserted k-v pairs in REVERSE order

```
it->SeekToLast();
it->Prev();
```

```
// Print all k-v pairs in reverse order
std::cout << "\nPrint all k-v pairs in reverse order: " << std::endl;

for (it->SeekToLast(); it->Valid(); it->Prev()) {
  std::cout << it->key().ToString() << ": " << it->value().ToString() << std::endl;
}

assert(it->status().ok());
```

Run demo_seek

```
cd ./examples
make ./demo_seek
./demo_seek
```

3.4. View generated files

LDB Tool

Print the **manifest file** to view the high-level information of the LSM-tree.

```
DEBUG_LEVEL=0 make ldb
./ldb manifest_dump --path="/path_to_the_file"
```

SST_Dump

Print the information of **SST files**.

```
DEBUG_LEVEL=0 make sst_dump
./sst_dump --file=/path_to_the_file.sst --show_properties
```

THANK YOU