Assignment 5: NoSQL

CSE 511: Data Processing at Scale - Spring 2023

Available: 04/11/2023 **Due Date**: 04/21/2023 11:59PM **Points:** 100

Introduction

RocksDB is an embeddable persistent key-value store for fast storage environments. It uses a log structured database engine, written entirely in C++ for maximum performance. Over the years, RocksDB has become a popular choice for a wide range of applications, including large-scale data processing, real-time analytics, and high-throughput transaction processing. It is used by companies such as Facebook, and LinkedIn to power their critical data-driven applications. From database storage engines such as MyRocks to application data caching to embedded workloads, RocksDB can be used for a variety of data needs. It provides basic operations such as opening and closing a database, reading and writing to more advanced operations such as merging and compaction filters. For this assignment, we will be exploring the more basic functionalities of this Key-Value Store.

Problem Statement

In the scope of this assignment, we will use `subreddits.csv` of the Reddit database from the previous assignments. The `subreddits.csv` is also available here. Also in this assignment, only high-level steps and expectations are mentioned. You are expected to explore the necessary requirements to solve the problem statement.

Step 0: Creating the setup for RocksDB (0 pts)

In the scope of this assignment, you **MUST** use RocksDB 7.10.2 Hence the first and most important step is to familiarize yourself with RocksDB and create a working setup for RocksDB. No Docker containers will be provided for this assignment.

The RocksDB library provides a persistent key-value store. Keys and values are arbitrary byte arrays. The keys are ordered within the key-value store according to a user-specified comparator function `rocksdb`. You can refer here for some help on installation.

Step 1: Loading the Data (25 pts)

RocksDB supports various write operations such as `Put`, `Merge`, and `WriteBatch`. These operations are optimized for high performance, and have features such as memtable,

write-ahead log, and compaction that allow for efficient and persistent write operations, making RocksDB a popular choice for applications that require high write throughput and low latency.

What to implement?

You need to load the data from the subreddits.csv file into RocksDB. However, the csv has multiple columns and RocksDB is a Key Value Store. The data has to be modified in some way for things to work. We will take advantage of the id for the data being unique for every row and the header of the csv file for this.

To store the data, you need to follow the format of the key being "<id_value>_<column_name>" where id_value is the value of the column id for the row and column_name is the name of the column. The value for this key will be the data in the column. Refer the example below for more details.

description	id
A CSE 511 example value	sp_23511

In this example, the key value pairs for the row will be:

Key: sp_23511_description value: A CSE 511 example value
Key: sp_23511_id value: sp_23511

***Note**: As shown in the example, make sure to have a key-value pair for the ID column as well.*

Step 2: Basic Operations

Your task is to implement the following operations for RocksDB. Each operation carries equal weightage. You can find a basic explanation for some basic operation here which will help you implement this assignment. You can also explore some existing examples here.

1. **Read (25 pts)**

   Read operations in RocksDB involve fetching data from the database and providing it to the application. The database provides various read operations like `Get`, `MultiGet`, and `Iterator` to fetch data. The Get operation returns the value associated with the given key, while MultiGet returns multiple values for a set of keys.

   What to implement?

   You must implement MultiGet in this stage of the assignment. The criterias for what you need to return can be found in the template code.

2. **Iterator (25 pts)**

   In RocksDB, an `iterator` is used to traverse through a subset of keys and values in a database. It allows for efficient sequential or random access to the data and supports forward and backward iteration. Iterators can be used to perform operations like range queries, prefix scans, and finding the next/previous key. However, care must be taken to use iterators correctly and avoid undefined behaviour.

   What to implement?

   You must implement Iterator in this stage of the assignment. The criterias for what you need to return can be found in the template code.

3. **Delete (25 pts)**

   `Delete` operation in RocksDB involves marking a key-value pair as deleted rather than physically deleting the data from the database. This is because RocksDB uses an LSM (Log-Structured Merge) tree-based storage engine which compacts data during background processes. The actual physical deletion of data is done during compaction. When a delete operation is performed, the key-value pair is marked with a special tombstone marker and will not be returned in future read operations.

   What to implement?

   You must implement Delete in this stage of the assignment. The criterias for what you need to return can be found in the template code.

***Note**: If you are using M1/M2 setups, we advise you to work on a multi-architectural docker you can create on your own or provided in the previous assignments to avoid system-specific issues.*

Grading

- The assignment is **auto-graded**.
- As part of this assignment, a tester template code has been provided to you. It is highly recommended that you expand upon this code and create your own test cases to thoroughly test the functionality of the code. **We will not be sharing any pre-made test cases for this assignment.**
- Your test case file will **not** be graded and there **is no extra credit** on this assignment.
- You **MUST** adhere to the requirements and the output schema such that your script does not fail the auto grader.

Submission Requirements & Guidelines

- Submit the assignment following the below guidelines
- This is an **individual** assignment

- Maintain your code on the **provided private GitHub repository for assignment-5** in the [SPRING-2023] [CSE511] Data Processing at Scale Organization). Make sure you don't use any other repository.
- **What to submit on Canvas?**
    - One C++ file (`assignment5.cc`)
- **What to maintain on GitHub?**
    - All relevant code files, no data (csv) files.
    - You need to decide on the relevance of each file to the compilation of your code and commit it accordingly.
- **Naming format:**
    a. You **MUST** name your cpp file as `assignment5.cc`

Submission Policies

- Late submissions will *absolutely not* be graded (unless you have verifiable proof of emergency). It is much better to submit partial work on time and get partial credit than to submit late for no credit.
- Every student needs to *work independently* on this exercise. We encourage high-level discussions among students to help each other understand the concepts and principles. However, a code-level discussion is prohibited, and plagiarism will directly lead to the failure of this course. We will use anti-plagiarism tools to detect violations of this policy.