Software Design Document

Benchmark Module Apache Gora

Sheriffo Ceesay

GSoC 2019

28th May 2019

Introduction:

This GSoC project is to create a benchmark module for Apache Gora project. The system should be able to run benchmarks on the various data store modules of the project. Common benchmarks like the CRUD operations should be supported.

This design document covers in detail the designs used or planned to use in the implementation process. It follows the original proposal submitted in the GSoC selection process.

Purpose:

The main purpose of this document is to provide an in-depth explanation of the design of the Apache Gora benchmark module, created for the Apache Gora Project. The document is technical and hence intended for programmers which includes my mentors and the entire Gora community as a guide for the project implementation. Finally, the document could also be used as a guideline by engineers who would work on this module in future.

Scope:

The document provides a detailed description of the architecture of the benchmark module. It contains use cases and class diagrams to see how users and objects in the implementation interact respectively.

Definitions, Acronyms, Abbreviations

YCSB Yahoo! Cloud Service Benchmark
DataStore:- Apache Gora implementation of a NoSQL database. This can be MongoDB, HBase, e.t.c.
keyClass:-The type of the key for the value to be inserted in the datastore.
persistenceClass:-The type of the object to be inserted.

Design Consideration:

Assumption:

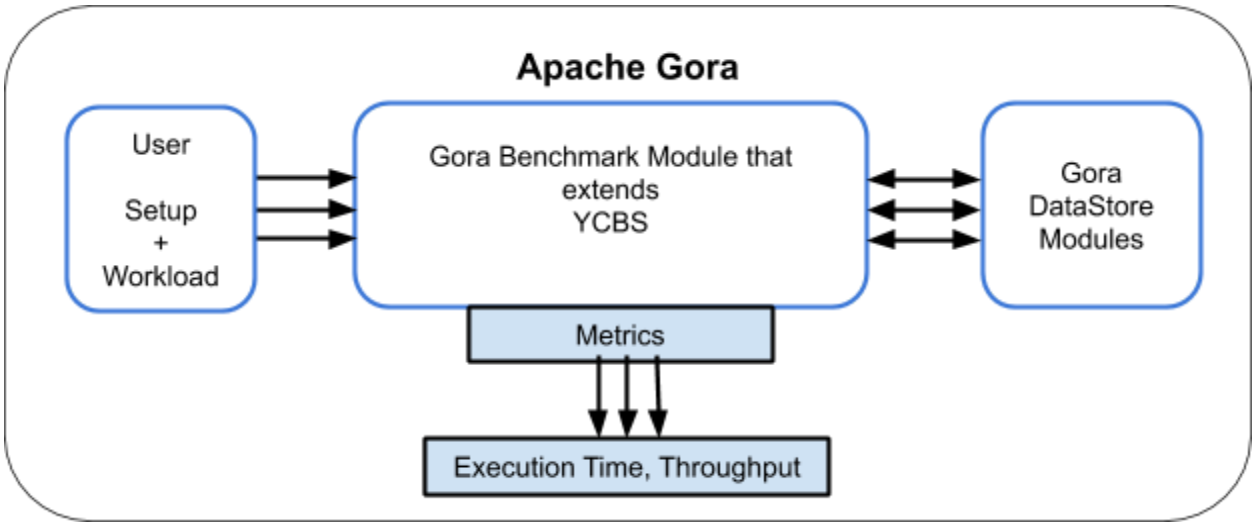
The benchmark-module is intended for technical users or engineers who want to measure and understand the efficiency of Apache Gora.

System Environment and Technologies Used

The module interacts with the entire Apache Gora project. The programming language used is Java. It also makes use of YCSB.

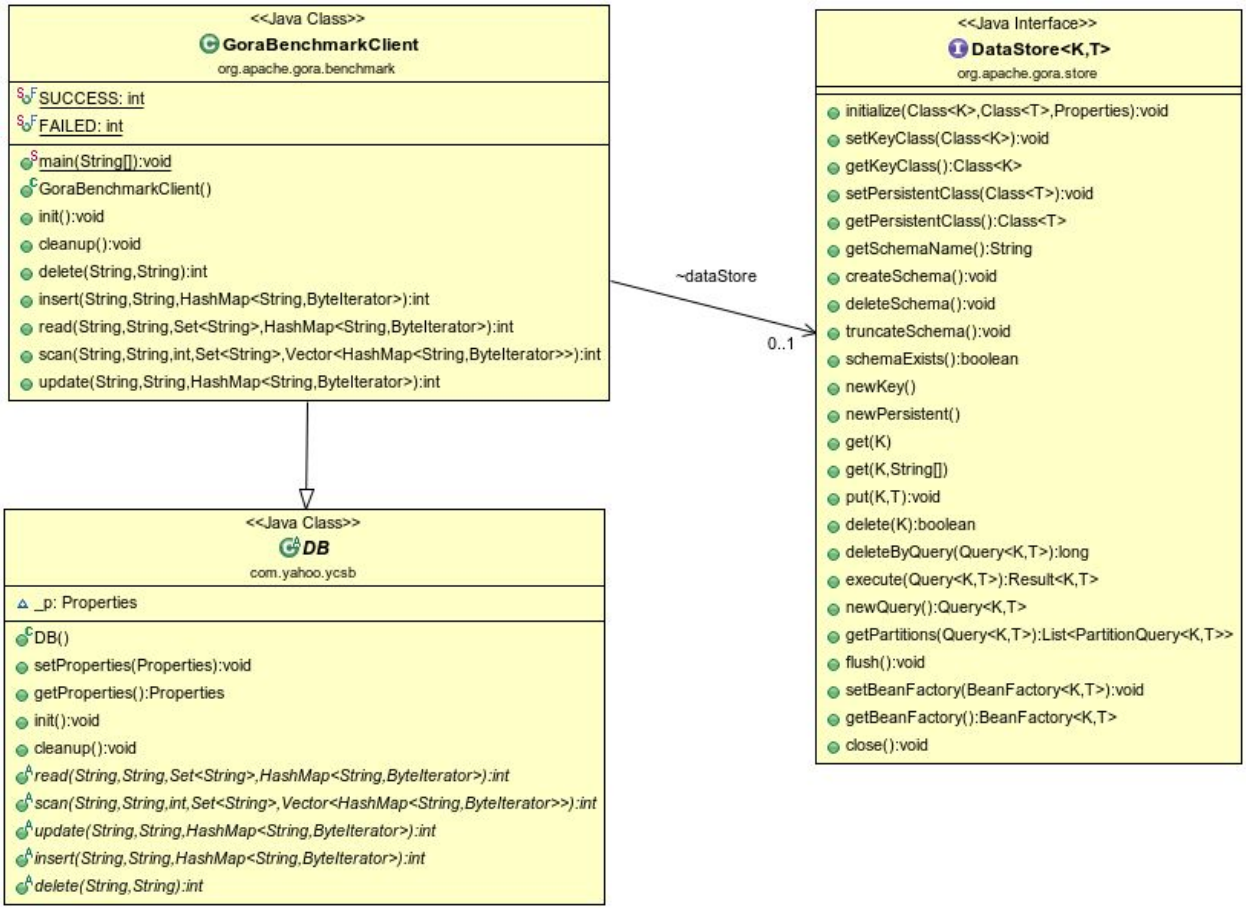
Architecture and System Design

The diagram below shows a high-level view of the actors to interact with the module. First, a user will set up the module and define workload using a configuration file. The workload is then executed using the benchmark module. The benchmark module runs the benchmark on the target data store and returns metrics upon completion.



Preliminary Class Diagram

This is the initial class diagram, however, I will update this I move on.



Sequence Of Operations:

1. An actor or a user configures the DataStore to benchmark. This is done by editing the gora.properties file.
2. An actor configures the workload to execute. Since we are using YCSB, we can use YCSB workload configuration mechanism to achieve this.
3. The default keyClass is java.lang.String and the default persistenceClass is generated.User.
4. Actor executes load operation to populate the DataStore with some dataset.
    a. By default, the load operation has 1000 records and 10 fields
    b. These settings can be changed using the workload settings.
5. Actor executes the transaction.

Extend YCSB and provide an implementation for Apache Gora.

YCSB is currently the defacto or state-of-the-art benchmark platform for NoSQL databases. It provides convenient configurable methods to generate and load

databases with synthetic data. It also provides mechanisms to run read, update, scan and delete transactions on the loaded dataset. At the end of each operation, various metrics are generated for analysis.

To extend YCSB the method on the second column of the table below must be implemented.

Mapping of Gora **DataStore** Interface to YCSB **DB** interface

Gora DataStore Interface	YCSB DB Interface
initialize(Class<K> keyClass, Class<T> persistentClass, Properties properties)	init()
T get(K key), get(K key,)	int read(String table, String key, Set<String> fields, Map<String, ByteIterator> result)
Result<K, T> execute(Query<K, T> query)	int scan(String table, String startkey, int recordcount, Set<String> fields, Vector<HashMap<String, ByteIterator>> result);
put(K key, T obj)	int update(String table, String key, Map<String, ByteIterator> values)
put(K key, T obj)	int insert(String table, String)
boolean delete(K key)	int delete(String table, String key);

Testing
The Java unit testing approach will be used to test the functionality.

References:
[Benchmark-Module Proposal]
[Apache Gora Project]
[YCSB Project]