Name: Sheriffo Ceesay
Project Name: Benchmark Module for Apache Gora
Issue Link: https://issues.apache.org/jira/browse/GORA-532
Email: sneceesay77@gmail.com

Introduction and Background

The ability to use Object Oriented Programming to interact with relational databases using Object Relational Model frameworks such as Hibernate and Apache Open JPA has greatly simplified their integration. With the advent to big data where there is no common standard or strict schema definition, it becomes impossible or challenging to deploy traditional ORM approaches to big data stores.

Apache Gora is an opensource framework which aims to give users an easy-to-use in-memory data model and persistence for big data frameworks with data store specific mappings. The overall goal for Apache Gora is to become the standard data representation and persistence framework for big data by providing easy to use Java API for accessing data agnostic of where the data is stored. It uses Apache Avro for data serialisation and depends on mapping files specific to each datastore.

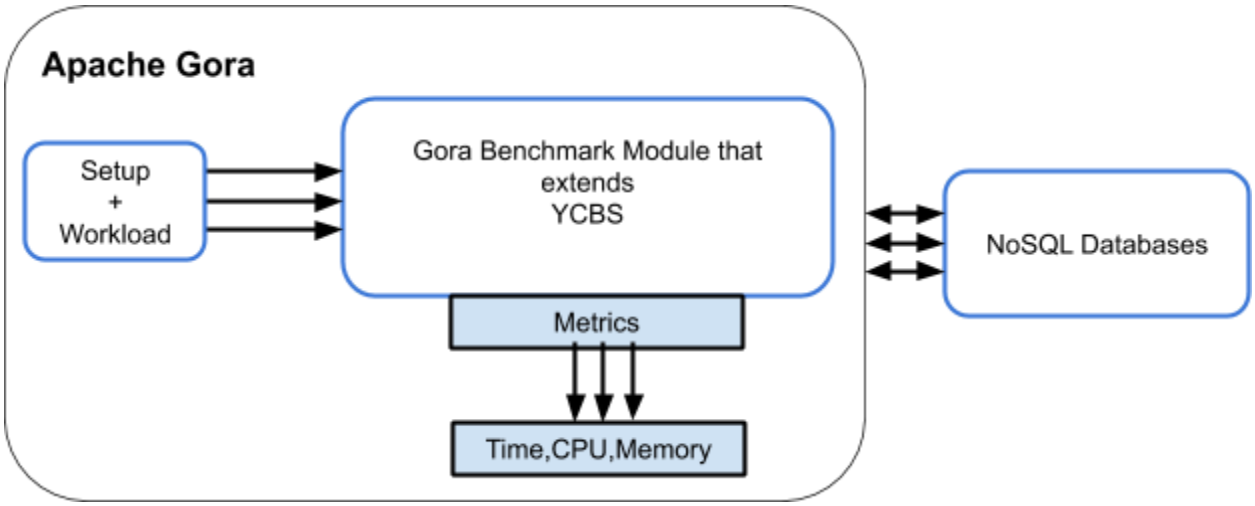
Apache Gora supports persisting data to various NoSQL store types. It supports column stores like HBase and Cassandra, key-value stores like Aerospike and Amazon DynamoDB and document stores like MongoDB and CouchDB. It also supports multi-model NoSQL databases like OrientDB. There is support for Apache Hadoop and Apache Spark for analysing data.

Benchmarking is the process of measuring the performance of a system in order to get insight into its functionality and general performance characteristics. This is normally achieved by executing representative workloads. Metrics like execution time, storage and CPU usage are often collected for analysis. Example of a benchmark standard in the RDBMS world is the TPC-DS benchmark [1]. Recently development in NoSQL and big data systems prompted researchers and engineers to come up with various benchmarking frameworks. Popular examples of these systems are Yahoo!'s Cloud Serving Benchmark YCSB++ [2], Apache Accumulo Benchmarking [3] and HiBench [4], Spark Bench [5].

The Problem Statement

The idea is to develop a Benchmark module that will help to identify and understand the various performance characteristics of Apache Gora. It will also help to identify the overhead incurred by Gora compared to the use of native NoSQL systems. This will help in fixing bug and aid performance improvement. The performance characteristics may range from execution time and resource utilisation. The proposed module could be used to benchmark and compare native implementation vs Apache Gora implementation.

Proposed Architecture and Solution



There are three main components in the above diagram.

1. Setup will configure the benchmarking module to communicate with Gora. Workloads are the programs or operation that will be executed on the Gora platform. We will include CRUD operational some stress test operations like sorting. The pipeline will also include functionality to generate synthetic data and load them into the datastore for the workloads to use.

2. Benchmark Module: This would be the main component. The module will send operations to Gora and metrics will be connected after the completion of an operation. For a start, results can be written to an organised text file.
3. Apache Gora: A setup of Gora targeting the installed NoSQL datastores.

Since Gora has implemented a variety of NoSQL database implementation, I think, we may not be able to include all these NoSQL databases in this project. So we could

target one or two databases in our first implementation. Subsequently, we could replicate this approach to other databases in the future.

Proposed Implementation idea

The implementation basically has to implement the methods defined in the YCSBs DB abstract class [7]. Therefore for each NoSQL database implemented by Gora, the solution would entail using Gora API to define a concrete implementation for the methods defined below. See sample skeletal code below and the mapping of YCSB DB and DataStore Interfaces.

```
public class GoraBenchmarkClient extends  DB{
        public GoraBenchmarkClient(){}

        init(...) // handles connecting to the database and any setup issue
        read(...) //read a single record
        scan(...) //read a range of records
        update(...) //update record(s)
        insert(...) //Insert a record
        delete(...) //delete a single record
}
```

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);

Deliverables

1. Gora-Benchmark module integrated with YCBS that can be used to run workloads `on Gora and return metrics like execution time. The following benchmark operations will be included in this work.
    a. read()
    b. scan()
    c. update()
    d. delete()
    e. sortOperation() → suggested by Kevin
2. A comparison of Gora vs Native NoSQL implementation. Example Gora-MongoDB implementation vs native MongoDB implementation.
3. Documentation on the usage of the system.
4. Reusable scripts for running the benchmarks.
5. A paper to submit to ApacheConf

Timeline

Task	Start	End
1. Community Bonding Period a. Understanding the internals of Apache Gora. b. Setup a local environment c. Explore Apache Gora source code d. Clarify and furnish the requirements and design I have already started looking into Gora internals, following tutorials, understanding the code base and the collaboration methods. I successfully, followed the LogAnalyser tutorial using HBase as the backend and modified the settings to use a local MongoDB installation as well. Currently, I am attempting to fix https://issues.apache.org/jira/projects/GORA/issues/GORA-565	Now	26th May
Develop the basic functionality of the core module, test and improve.	27 May	23 June
Phase 1 Evaluation	24 June	28 June

Improve the basic functionality by adding more benchmark operations, compare Gora and Native implementations.	29 June	21 July
Phase 2 Evaluation	22 July	26 July
Finalise the functionality and provide documentation	27 July	18 Aug
Final Week: Submit final work for evaluation	19 Aug	26 Aug

Availability

This project is related to what I am currently doing my PhD so I will be able to dedicate 30 hrs/week hours to this. I have already discussed this with my supervisor and he welcomed the idea.
I may have two possible trips as indicated in the table below.

15 - 17 July	Annual CS School trip to The Burn	Confirmed
3 - 12 August	I may attend a conference in Japan	Not yet confirmed

Contribution to Apache Gora

https://issues.apache.org/jira/browse/GORA-564

About Me

I am a third year PhD student at the University of St Andrews, UK. My research focuses on Benchmarking and Performance Modelling of Big Data Applications. I have worked extensively with various Benchmarking frameworks like HiBench[2], BigDataBench[3] and read a lot of papers about their approach and implementation. I have also developed a custom benchmarking tool for MapReduce jobs which obtains the execution time of all sub-phases e.g. (Read, Map, Collect, Spill, Merge, Shuffle, Reduce and Write).

I wrote a paper in 2017 that focuses on simplifying the deploying of benchmark tools titled Plug and Play Bench: Simplifying Benchmarking using Containers [6].

I am conversant with MapReduce and Apache Spark. I am currently modelling the performance of data flow with cycles pattern. Apache Spark is an example of such a pattern.

As part of my master's degree program, I did a six months internship with Christie National Health Service in Manchester. My main responsibility was to investigate and

gauge the possibility of moving or replicating their various RDBM systems to a NoSQL solution and provide a simple interface that would help them aid this movement. I solved this challenge by merging and dumping the selected databases into MongoDB. This project, including the source codes, are totally propriety.

Finally, I had always wanted to actively contribute to open source projects and I am sure. this is a great opportunity for me to get started.

Reference

[1] Nambiar, R.O. and Poess, M., 2006, September. The making of TPC-DS. In *Proceedings of the 32nd international conference on Very large databases* (pp. 1049-1058). VLDB Endowment.

[2] Cooper, B.F., Silberstein, A., Tam, E., Ramakrishnan, R. and Sears, R., 2010, June. Benchmarking cloud serving systems with YCSB. In *Proceedings of the 1st ACM symposium on Cloud computing* (pp. 143-154). ACM.

[3] Sen R, Farris A, Guerra P. Benchmarking apache accumulo bigdata distributed table store using its continuous test suite. In2013 IEEE International Congress on Big Data 2013 Jun 27 (pp. 334-341). IEEE.

[4] Huang S, Huang J, Dai J, Xie T, Huang B. The HiBench benchmark suite: Characterization of the MapReduce-based data analysis. In2010 IEEE 26th International Conference on Data Engineering Workshops (ICDEW 2010) 2010 Mar 1 (pp. 41-51). IEEE.

[5] Li, M., Tan, J., Wang, Y., Zhang, L. and Salapura, V., 2015, May. Sparkbench: a comprehensive benchmarking suite for in-memory data analytic platform spark. In *Proceedings of the 12th ACM International Conference on Computing Frontiers*(p. 53). ACM.

[6] Ceesay, S., Barker, A. and Varghese, B., 2017, December. Plug and play bench: Simplifying big data benchmarking using containers. In *2017 IEEE International Conference on Big Data (Big Data)* (pp. 2821-2828). IEEE. (https://ieeexplore.ieee.org/abstract/document/8258249)

[7] https://github.com/brianfrankcooper/YCSB/wiki/Adding-a-Database