Aerospike Datastore Implementation

Project	Implement Aerospike Datastore
Product	Apache Gora
Name	Nishadi Kirielle
University	University of Moratuwa, Sri Lanka
Email	ndimeshi.12@cse.mrt.ac.lk
Github	https://github.com/nishadi
LinkedIn	https://www.linkedin.com/in/nishadikirielle

About Me

I am a Computer Science and Engineering undergraduate from University of Moratuwa, Sri Lanka. Currently I am in the final phase of my final year research project which is to improve query performance of in-memory databases using bitmaps. My interested areas involve Cloud Computing, In-memory Databases, NoSQL and Big data. In addition, I have an in-depth knowledge in Java, Maven and Git.

Open source contributions

- WSO2 App Cloud

  My contribution for WSO2 App Cloud product involves creating custom application, researching on a solution for the Kubernetes session affinity problem, configuring a load balancer for App Cloud and enabling HTTPS support in App Cloud Kubernetes cluster.

- WSO2 PPaaS

  This project contribution was to create an artifact migration tool to migrate artifacts of WSO2 PPaaS 4.0.0 to WSO2 PPaaS 4.1.x.

Other:

- Bit-Oriented Analytics Platform

  I have engaged in this project as my final year project to develop and evaluate a cluster-based, bit-oriented analytics platform (storage engine) designed to manage large, fast-growing volumes of data and provide fast query performance when used for OLAP and other query-intensive applications. In this project I have developed an extension to Apache spark SQL to incorporate bitmap compression for data manipulations.

Background

Apache Gora is an in-memory data model that facilitates object to datastore mappings providing a data representation and persistence framework for big data. As it currently supports for persisting objects to various database models such as column stores like Apache Hbase, Apache Cassandra and key value stores, this project aims to extend its capability to provide support for Aerospike database.

Aerospike [1] is a key value store having a distributed NoSQL database. The key value store operations associate with records (RDBMS rows), namespaces (RDBMS databases), sets (RDBMS tables) and bins (RDBMS columns). Each record contains a unique key and one or more bins containing the values. The data model used in Aerospike is schemaless [2].

Proposed Solution

The expected approach is to start with designing and implementing datastore. As per the modular architecture in Apache Gora code base, the newly implementing module to support for the Aerospike datastore will be added as a new module named with gora-aerospike.

In order to interact with Aerospike server, Aerospike has provided the support for clients in several languages. Aerospike Java client can be used for the purpose of interacting with the server in implementing the Gora Aerospike datastore module.

blocked URL

Configuration

In configuring Gora to use the Aerospike client, the basic configuration needs to be updated to use the Aerospike backend via the properties file. In addition, the Aerospike server specific IP addresses and ports need to be provided via the properties files.

```
gora.datastore.default=org.apache.gora.aerospike.store.AerospikeStore


#Aerospike dataStore properties
gora.aerospikestore.server.ip=localhost
gora.aerospikestore.server.port=3000
```

Implementation

In the implementation of the new module, the basic consideration is on the datastore class, which is named AerospikeStore in this module. It is extended from the DataStoreBase class and with this base class, all of the basic functionality needed to implement for the new data store are implemented.

```
public class AerospikeStore<K,T extends PersistentBase> extends DataStoreBase<K,T> {
}
```

1. Initializing the connection with Aerospike server

   Initializing the connection, an Aerospike client needs to be created specifying the IP address and port of one or more seed nodes in the cluster. If the user specifies a single seed node, the client first connects to the specified seed node and then discovers the rest of the cluster. If the user specifies multiple seed nodes, then the client iterated through the array of seed nodes until it successfully gets connected to one of the seed nodes and then discovers the cluster. Further, in the scenarios where the user connects an Aerospike enterprise servers which may require user authentication, the user has the ability to provide user authentication details. [3] All of these initial details can be provided though the properties file.

2. Reading, writing and deleting data records from the Aerospike server

   Reading, writing and deleting data records from the server can be achieved via overriding the corresponding methods in the data store base. In Aerospike client, to write records there are several options including writing with a policy, writing a single record and writing multiple records [4]. To delete a value of a given key, setting the corresponding bin(column) value to NULL is the provided approach by Aerospike server. In reading, Aerospike provides ways to read all bins or specified bins from the server. So that it can be implemented via Gora module.

3. Executing queries with Aerospike backend

As per in reading, writing and deleting, for querying as well, the corresponding methods in the data store base can be overridden. Then in order to query the Aerospike server, the java client can be used as it has facilitated the query defining and executing. To execute the queries Aerospike clients query API can be invoked.

4. Terminating the connection with Aerospike server

For the termination of the connection to the server, the same approach can be used that is used in initializing the connection.

As per the final step of the Gora Aerospike datastore module, usage guide will be documented on how to use the new module.

Results for Apache Community

With the completion of this project, the Apache community gets access to use Aerospike database backend with Apache Gora product.

Deliverables

1. A new maven module to provide support for Aerospike backend including following basic functionalities and test cases.
    a. Initializing the connection with Aerospike server
    b. Writing and deleting data from the Aerospike server
    c. Executing queries with Aerospike backend
    d. Terminating the connection with Aerospike server
2. Documentation for the the usage and functionality of the module
3. An additional sample for gora-tutorial module to understand the usage of the datastore depending on the requirement.

Scheduling

Activity	Start Time	End Time
Community Bonding Period : Learn the organization, community, guidelines and best practices. Finalize requirements.	04-05-2017	29-05-2017
Design the gora-aerospike-mapping.xml	30-05-2017	06-06-2016
Improve the implementation of put method in the prototype, implement get, delete methods. Write test cases for the implemented functionality	07-06-2017	18-06-2016
Discuss the completed functionality of the current implementation and the progress with mentor to get feedbacks and suggestions	19-06-2017	25-06-2017
Midterm Evaluations	26-06-2017	30-06-2017
Implement query execution functionality	01-07-2017	12-07-2017
Complete the gora-aerospike module and test the functionality	13-07-2017	20-07-2017
Documenting the usage of gora-aerospike module and extended time period for any project risks.	21-07-2017	28-07-2017
Final Evaluations	29-07-2017	29-08-2017

Community Engagement

In familiarizing with the project and the community, I have communicated via the Apache Gora dev-mailing list and Apache Gora JIRA to discuss the details regarding the project idea. Further I have developed a prototype of the gora-aerospike module which has the ability to connect with Aerospike server and put data records into it.

https://github.com/nishadi/gora/blob/gora-aerospike/gora-aerospike/src/main/java/org/apache/gora/aerospike/store/AerospikeStore.java

References

[1]."Aerospike High Performance NoSQL Database", Aerospike, 2017. [Online]. Available: http://www.aerospike.com/. [Accessed: 29- Mar- 2017].

[2]. "Data Model Aerospike", Aerospike.com, 2017. [Online]. Available: http://www.aerospike.com/docs/architecture/data-model.html. [Accessed: 29- Mar- 2017].

[3]. "Connecting Aerospike", Aerospike.com, 2017. [Online]. Available: http://www.aerospike.com/docs/client/java/usage/connect. [Accessed: 29- Mar- 2017].

[4]. "Key Value Store Aerospike", Aerospike.com, 2017. [Online]. Available: http://www.aerospike.com/docs/client/java/usage/kvs/write.html. [Accessed: 29- Mar- 2017].