GSoC with Apache JCache Data store for Apache Gora

Kevin Ratnasekera, Software Engineer, WSO2

- About myself

- Software Engineer for WSO2 (kevin@wso2.com)
- Working as member of Integration technologies team
- Interests for Distributed systems
- Open source Fan
- Not related to Google or Hazelcast.

[1] http://wso2.com

- Agenda

- GSoC and Apache contribution.
- Apache Gora project.
- JCache data store for Apache Gora
- JCache API.
- Roadmap for Apache Gora.
- Conclusion.

- Google Summer of code

- How does GSoC work?
- GSoC statistics for 2016 program
  1,206 students
  178 open source organizations
  85.6% overall success rate
- ASF contribution
  ~50 students
  37 completed final evaluation

[1] https://developers.google.com/open-source/gsoc/resources/stats

- Apache software foundation

- 175 committees managing 294 community based projects
- 59 incubating podlings
- Active repos for ASF
  870 active repos maintained at github
  314 active Apache members at github

[1] https://projects.apache.org/
[2] https://github.com/apache
[3] https://people.apache.org/committer-index.html

- ASF as GSoC mentoring organization

- Considering 2010-2016 statistics
- Accepted students ~50 for each year
- Assigned mentors ~75 for each year
- One of the largest mentoring organizations

[1] www.slideshare.net/smarru/google-summer-of-code-at-apache-software-foundation

- Benefits to community.

- New contributors to the project.
- Long term contributors (committers/PMC members)
- New features/improvements/bug fixes to project.

- Apache Gora Project

- Data Persistence
  Abstract persistent layer for NoSQL, In memory data model, Persistence for Big data, Object to data store, Data store specific mappings
- Data Access
  Abstract Datastore API, Common interface for retrieval, alteration and query, Hide details on specific persistent data store implementation.
- MapReduce support
  Out of the box to run MR jobs over the Gora input data store, store results over the output data stores (Recently introduced Spark backend)

- Typical Gora usage

- Define persistent bean definition using Apache AVRO JSON schema.
- Compile the schema using Gora compiler.
- Create mapping file which maps between persistent bean to physical data store.
- Configure gora.properties to reflect data store properties.
- Create data store using DataStoreFactory

[1]https://gora.apache.org/current/tutorial.html

- Data Store API

```
public interface DataStore<K, T extends Persistent> {
  void initialize(Class<K> keyClass, Class<T> persistentClass, Properties properties);
  void setKeyClass(Class<K> keyClass);
  Class<K> getKeyClass();
  void setPersistentClass(Class<T> persistentClass);
  Class<T> getPersistentClass();
  String getSchemaName();
  void createSchema();
  void deleteSchema();
  void truncateSchema();
  boolean schemaExists();
  K newKey();
  T newPersistent();
  T get(K key);
  T get(K key, String[] fields);
  void put(K key, T obj);
  boolean delete(K key);
  long deleteByQuery(Query<K, T> query);
  Result<K, T> execute(Query<K, T> query);
  Query<K, T> newQuery();
  List<PartitionQuery<K, T>> getPartitions(Query<K, T> query) throws IOException;
  void flush();
  void setBeanFactory(BeanFactory<K,T> beanFactory);
  BeanFactory<K,T> getBeanFactory();
  void close();
}
```

- Writing a dataStore for Apache Gora.

- Implementation for 3 Abstract classes.

  DataStoreBase<K, T>
  QueryBase<K, T>
  ResultBase<K, T>

[1]https://cwiki.apache.org/confluence/display/GORA/Writing+a+new+DataStore+for+Gora+HOW_TO

- The need for Cache data store

- Limitations of Gora secret in memory store – MemStore
- Static ConcurrentSkipList map restricted to single instance per JVM, MemStore cannot be shared across JVMs (distributed)
- Reduce latency in persistent bean creation/retrieval from back-end database (repetitive reads)
- Caching layer irrespective backend persistent data store implementation (decoupled)

[1] http://events.linuxfoundation.org/sites/events/files/slides/deploying_gora_as_query_broker.pdf

- JCache API

- Standardize Caching API for Java platform. No more proprietary API’s.
- Common mechanism to create, access, update and remove data from caches.
- Doesn’t say anything about data distribution, network topology and wire level protocol etc.
- Implementation by different vendors,
  Ehcache, Infinispan, Hazelcast

- Why JCache?

- Portability between different Vendor implementations
- Developer productivity – learning curve is smaller.

- Fundamental differences

java.util.Map	javax.cache.Cache
Key Value based API	Key Value based API
Support Atomic updates	Support Atomic updates
Entries don’t get Expired/Evicted	Entries get Expired/Evicted
Entries stored on-heap	Entries stored anywhere
Store-By-Reference	Store-By-Value/ Store-by reference
	Integration with Loaders/writers
	Observation with Entry Listeners
	Statistics

[1] http://www.slideshare.net/DavidBrimley/jcache-its-finally-here

- JCache code sample

```
// explicitly retrieve the Hazelcast backed javax.cache.spi.CachingProvider
CachingProvider cachingProvider = Caching.getCachingProvider(
        "com.hazelcast.cache.HazelcastCachingProvider"
);

// retrieve the javax.cache.CacheManager
CacheManager cacheManager = cachingProvider.getCacheManager();

// create javax.cache.configuration.CompleteConfiguration subclass
CompleteConfiguration<Integer, User> config =
        new MutableConfiguration<~>();

// create cache on name users
Cache<Integer, User>  cache = cacheManager.createCache("users", config);

//example on cache add
cache.put(new Integer("1234"), new User(1234,"Kevin Ratnaskera"));
```

- JCache Cache Loader/Writer

- Integration with external resources.
- Handles Read through and write through caching for external resources.
- Register Loader/Writer and Read/Write through enabled at cache configuration.

```
public interface CacheLoader<K, V> {
    V load(K var1) throws CacheLoaderException;

    Map<K, V> loadAll(Iterable<? extends K> var1) throws CacheLoaderException;
}
```

```
public interface CacheWriter<K, V> {
    void write(Entry<? extends K, ? extends V> var1) throws CacheWriterException;

    void writeAll(Collection<Entry<? extends K, ? extends V>> var1) throws CacheWriterException;

    void delete(Object var1) throws CacheWriterException;

    void deleteAll(Collection<?> var1) throws CacheWriterException;
}
```

- JCache Cache Entry Listener

- Receives events related to cache entries (create,expiry, update, remove)
- Useful in distributed caches.
- Register at cache configuration.

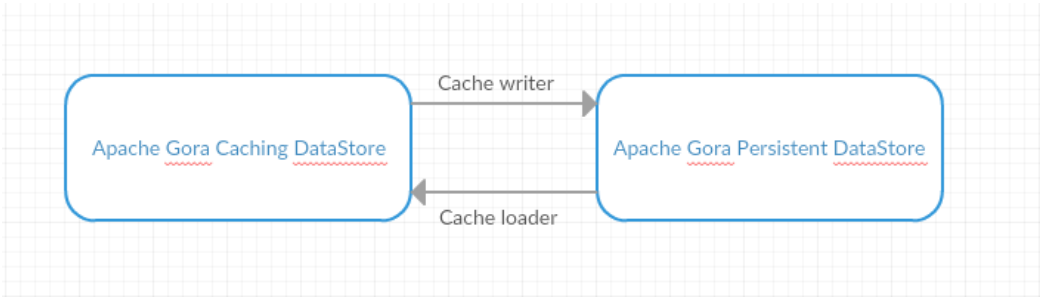
```
public interface CacheEntryRemovedListener<K, V> extends CacheEntryListener<K, V> {
    void onRemoved(Iterable<CacheEntryEvent<? extends K, ? extends V>> var1) throws CacheEntryListenerException;
}
```

- Hazelcast as JCache provider

- Apache license compliance
- Rich vendor specific additions such as
  Asynchronous operations
  Eviction
  Near cache
  Data distribution/partitioning exposed over vendor specific API

- Basic Design

- Implement cache as another data store exposing the same data store interface
- Cache data Store act as wrapper to persisting store delegating operations
- Make Persistent bean serializable.



- Configuration for caching data store

- Configuring persistent data store to expose over caching data store
- gora.properties

```
gora.datastore.default=org.apache.gora.hbase.store.HBaseStore

#JCache dataStore properties
gora.cache.datastore.default=org.apache.gora.jcache.store.JCacheStore
gora.datastore.jcache.provider=com.hazelcast.cache.impl.HazelcastServerCachingProvider
#gora.datastore.jcache.provider=com.hazelcast.client.cache.impl.HazelcastClientCachingProvider
#gora.datastore.jcache.hazelcast.config=hazelcast-client.xml
gora.datastore.jcache.hazelcast.config=hazelcast.xml
```

- Creating persistent data store instances which are exposed over the caching data store

```
//this dataStore talks to persistent store via the cache
dataStore = DataStoreFactory.getDataStore(Long.class, Pageview.class, new Configuration(), true);

//this dataStore talks  directly to persistent store
dataStore = DataStoreFactory.getDataStore(Long.class, Pageview.class, new Configuration(), false);

//this dataStore talks  directly to persistent store
dataStore = DataStoreFactory.getDataStore(Long.class, Pageview.class, new Configuration());
```

- Making Persistent data beans serializable

- Hazelcast as cache provider.
- Maintain data beans in serialized form inside caches.
- Need to preserve dirty state bytes as well as data.
- Two Approaches
  Using pure JAVA serialization, writing custom serializers.

- Pure Java Vs. Custom AVRO serializers

- Utf8, ByteBuffer and GenericData.Array are not in it s serializable form
- AVRO SpecificRecord class level fields instances Either should be declared as transient or implement serializable
- Rather not depend on another 3 rd party dependency for serialization.
- Custom serialiazer have freedom get extended from pluggable serializers from variety of methods

- Pure Java Vs. Custom AVRO serializers

```
private static final org.apache.avro.io.DatumWriter
        DATUM_WRITER$ = new org.apache.avro.specific.SpecificDatumWriter(SCHEMA$);
private static final org.apache.avro.io.DatumReader
        DATUM_READER$ = new org.apache.avro.specific.SpecificDatumReader(SCHEMA$);

/**
 * Writes AVRO data bean to output stream in the form of AVRO Binary encoding format. This will transform
 * AVRO data bean from its Java object form to it s serializable form.
 *
 * @param out java.io.ObjectOutput output stream to write data bean in serializable form
 */
@Override
public void writeExternal(java.io.ObjectOutput out)
        throws java.io.IOException {
  out.write(super.getDirtyBytes().array());
  DATUM_WRITER$.write(this, org.apache.avro.io.EncoderFactory.get()
          .directBinaryEncoder((java.io.OutputStream) out,
                  null));
}

/**
 * Reads AVRO data bean from input stream in it s AVRO Binary encoding format to Java object format.
 * This will transform AVRO data bean from it s serializable form to deserialized Java object form.
 *
 * @param in java.io.ObjectOutput input stream to read data bean in serializable form
 */
@Override
public void readExternal(java.io.ObjectInput in)
        throws java.io.IOException {
  byte[] __g__dirty = new byte[getFieldsCount()];
  in.read(__g__dirty);
  super.setDirtyBytes(java.nio.ByteBuffer.wrap(__g__dirty));
  DATUM_READER$.read(this, org.apache.avro.io.DecoderFactory.get()
          .directBinaryDecoder((java.io.InputStream) in,
                  null));
}
```

- Possible improvements

- Caching performance heavily depend on serialization/deserialization performance. Experiment with different serialization methods.
- Remove vendor specific Hazelcast JCache implementation (Eg :- Eviction policy – Not included JCache specification) from JCache data store.
- Ability to dynamically take any JCache provider.

[1] http://blog.hazelcast.com/comparing-serialization-methods

- Sample/Tutorial for JCache data store

- DistributedLogManager sample.
- Demonstrates standalone/distributed caching for data stores.

[1] https://issues.apache.org/jira/browse/GORA-484
[2] http://github.com/apache/gora/blob/master/gora-tutorial/src/main/java/org/apache/gora/tutorial/log/DistributedLogManager.java
[3] http://gora.apache.org/current/tutorial.html#jcache-caching-datastore

- References for project

- JCache store implementation [1]
- Documentation for project [2][3]

[1] https://issues.apache.org/jira/browse/GORA-409
[2] https://issues.apache.org/jira/browse/GORA-484
[3] http://gora.apache.org/current/gora-jcache.html

- Roadmap for Apache Gora

- REST API exposing data store functionalities. [1]
- Improve data store support.
  Eg:- Apache Kudu
- Different serialization frameworks other than AVRO. [2]
  Eg:- Apache thrift, Protocol buffers
- Different execution engine support. [3]
  Eg:- Apache Flink

[1] https://issues.apache.org/jira/browse/GORA-405
[2] https://issues.apache.org/jira/browse/GORA-279
[3] https://issues.apache.org/jira/browse/GORA-418

- Conclusion

- Contribute to Apache Gora
- Check Roadmap, Mailing lists, JIRA issues
- Join Apache GSoC effort
- Higher project acceptance/slot count for GSoC 2017

[1] https://issues.apache.org/jira/browse/gora
[2] http://gora.apache.org/mailing_lists.html
[3] https://developers.google.com/open-source/gsoc/timeline