Integrate Master Data with Big Data using Oracle Table Access for Hadoop

Kuassi Mensah
Oracle Corporation
Redwood Shores, CA, USA

Keywords: Hadoop, BigData, Hive SQL, Spark SQL, HCatalog, StorageHandler

Introduction

The goal of Big Data Analytics is to furnish actionable information to help business decisions making. Example: *"Which of our products got a rating of four stars or higher, on social media in the last quarter?"* However, producing actionable information requires both Big Data (i.e., facts, usually stored in HDFS or NoSQL DBMS) as well as Master Data (i.e., products, customers and so on, usually stored in Oracle database). Accessing and integrating Master Data in Big Data analytics requires either an ETL copy or direct access from Hadoop or Spark.
Oracle Table Access for Hadoop[1] and Spark (OTA4H) turns remote Oracle database tables into Hadoop or Spark data sources. How does OTA4H works? Does it scale? How reliable and secure? This paper discusses the key features and benefits.

Oracle Table Access for Hadoop and Spark (OTA4H)

The Hadoop 2.x architecture furnishes a set of interfaces and abstraction layers which decouple the compute engines (Hive SQL, Spark, Big data SQL, Impala, Pig, MapReduce, etc), the cluster management (YARN), the computing resources (CPU, memory), and the data sources (HDFS, NoSQL, external tables). OTA4H is an Oracle Big Data Appliance feature which implements Apache Hadoop StorageHandler interface (InputFormat, SerDe) and allows direct, parallel, fast, consistent and secure access to Master Data in Oracle database using Hadoop/Spark query engines (Hive SQL, Spark SQL, and so on), as well as Hadoop/Spark APIs (Pig, MapReduce, and so on).

Example

Let HiveTab be a local Hive table and OracleTab an external table referring to a remote Oracle database table. The following Hive query[2] will perform a local JOIN between rows from HiveTab and selected[3] rows from Oracle, then return the firstname, lastname and the bonus of employees who have a salary greater than 70000 and received a bonus greater than 7.000".

```
Query="SELECT HiveTab.First_Name, HiveTab.Last_Name, OracleTab.bonus
      FROM HiveTab join OracleTab on (HiveTab.Emp_ID=OracleTab.Emp_ID)
      WHERE salary > 70000 and bonus > 7000;"
```

Here is the sequence of actions upon a Hadoop or Spark JOIN query or API call.

1. OTA4H receives parts of the execution plan related to the Oracle table.

[1] Just released as part of Oracle's Big Data Appliance 4.3 release.
[2] A simple JOIN operation, not a typical Big Data analytical query.
[3] To be more precise, the result set of an Oracle database sub-query.

2. It dynamically generates a number of database splits using the specified split pattern (discussed later). To ensure the consistency of the result set, all splits are tagged with the same Oracle database's System Commit Number a.k.a. SCN.
3. An Oracle SQL sub-query is generated for each split using, preferably, a secure connections (Kerberos, SSL, Oracle Wallet), and enforcing predicate pushdown, columns projection pushdown, or partition pruning, when appropriate.
4. Each split is processed by a Hadoop/Spark task (often a Map task with HiveSQL)). The consistency is enforced during the reading/scanning of the Oracle table, using the SCN captured during the split generation.
5. The matching rows from all tasks are returned to the query coordinator on Hadoop/Spark which performs the JOIN with rows from the local table.

Running Hive SQL with OTA4H

The demo scripts, Hadoop data, and Oracle database tables are available as part of BigDataLite 4.3 VirtualBox appliance.

Configuration

```
HIVE_AUXPATH="/opt/oracle/ota4h/jlib/ojdbc7.jar:/opt/oracle/ota4h/jlib/osh.jar:/opt
/oracle/ota4h/jlib/ucp.jar"
hive_init.hql
     add jar /opt/oracle/ota4h/jlib/osh.jar;
     add jar /opt/oracle/ota4h/jlib/ojdbc7.jar
     add jar /opt/oracle/ota4h/jlib/ucp.jar
```

Invocation/Execution

```
hive --auxpath ${HIVE_AUXPATH} -i hive_init.hql -e "${QUERY}"
```

Running Spark SQL with OTA4H

The compatibility of Spark SQL with Hive metastore allows running Spark against Hive tables (therefore external tables), using the same HiveQL queries. We are looking into further Spark integration with the Oracle database including SChemaRDD, DataFrames, and so on.

Configuration: here are the steps for running Spark SQL against Oracle database tables.

1. Add `ojdbc7.jar and osh.jar` to CLASSPATH in `/usr/lib/spark/bin/compute-classpath.sh`

   ```
   CLASSPATH="$CLASSPATH:/opt/oracle/ota4h/lib/osh.jar"
   CLASSPATH="$CLASSPATH:/opt/oracle/ota4h/lib/ojdbc7.jar"
   ```

2. Edit SPARK_HOME in `/usr/lib/spark/conf/spark-env.sh`

   ```
   export SPARK_HOME=/usr/lib/spark:/etc/hive/conf
   ```

3. Specify additional environment variables in /usr/lib/spark/conf/spark-env.sh.

   ```
   export DEFAULT_HADOOP=/usr/lib/hadoop
   export DEFAULT_HIVE=/usr/lib/hive
   export DEFAULT_HADOOP_CONF=/etc/hadoop/conf
   ```

```
export DEFAULT_HIVE_CONF=/etc/hive/conf
export HADOOP_HOME=${HADOOP_HOME:-$DEFAULT_HADOOP}
export HADOOP_HDFS_HOME=${HADOOP_HDFS_HOME:-${HADOOP_HOME}/../hadoop-hdfs}
export HADOOP_MAPRED_HOME=${HADOOP_MAPRED_HOME:-${HADOOP_HOME}/../hadoop-
mapreduce}
export HADOOP_YARN_HOME=${HADOOP_YARN_HOME:-${HADOOP_HOME}/../hadoop-yarn}
export HADOOP_CONF_DIR=${HADOOP_CONF_DIR:-$DEFAULT_HADOOP_CONF}
export HIVE_CONF_DIR=${HIVE_CONF_DIR:-$DEFAULT_HIVE_CONF}
CLASSPATH="$CLASSPATH:$HIVE_CONF_DIR"
CLASSPATH="$CLASSPATH:$HADOOP_CONF_DIR"

if [ "x" != "x$YARN_CONF_DIR" ]; then
  CLASSPATH="$CLASSPATH:$YARN_CONF_DIR"
fi
```

4. Make sure all Hadoop libs are added

```
CLASSPATH="$CLASSPATH:$HADOOP_HOME/client/*"
CLASSPATH="$CLASSPATH:$HIVE_HOME/lib/*"
CLASSPATH="$CLASSPATH:$($HADOOP_HOME/bin/hadoop classpath)"
```

Invocation/Execution

```
/usr/lib/spark/bin/spark-sql -e "${QUERY}"
```

Key OTA4H Features and Benefits

Beyond implementing the `StorageHandler` and `InputFormat` interfaces for the Oracle database, OTA4H furnishes unique optimizations in the areas of performance, scalability, high-availability and consistency.

Performance and Scalability

In order to fully exploit the capacity of Hadoop or Spark clusters and Oracle database servers, OTA4H furnishes the following optimizations.

- Splitter patterns. As of this release, OTA4H furnishes four split patterns: `SINGLE_SPLITTER, ROW_SPLITTER, BLOCK_SPLITTER, and PARTITION_SPLITTER`; these allow dividing the execution of the Oracle sub-query into several tasks which can be run in parallel. Each of these split patterns is fully described in the OTA4H doc (a chapter in BDA 4.3 doc).
- Optimized JDBC driver. OTA4H is packaged with a special JDBC driver which is instrumented to create Hadoop Writable types i.e.,Text and Date from SQL types -- without materializing intermediate Java objects.
- Connection caching. For each query, OTA4H connects to the Oracle database to compute the number of splits, checks whether partition pruning is possible, and so on. Each Hadoop or Spark task executes in a separate JVM but all tasks are not fired simultaneously; OTA4H retains and reuses those JVMs for the remaining tasks of the same query, thereby optimizing the overall query response time.
- Integration with Database Resident Connection Pooling (DRCP). Oracle DBAs may control the number of concurrent connections originating from Hadoop or Spark, using an RDBMS-side pool and setting a max connection limit. As documented, the JDBC URL specified in the external table's DDL must refer to DRCP through the "`:POOLED`" suffix in short JDBC URLs or "`(SERVER=POOLED)`" in long JDBC URLs.

» Column projection pushdown. By default[4], Hadoop or Spark queries retrieve only the specified columns from the Oracle database table. OTA4H enforces such projection during the rewriting of the sub-queries for each split.

» Predicate pushdown. By default[5], OTA4H incorporates the predicates in the WHERE clause when generating sub-queries for each split thereby ensuring that only the needed rows are retrieved from Oracle.

» Partition Pruning. When an external table is associated with a partitioned Oracle table using the `PARTITION_SPLITTER`, if the query can be resolved using only the relevant partitions, in other words, if predicate is a function of the partition key tasks. OTA4H generates splits only for the relevant partitions. The irrelevant partitions are skipped and not accessed. When the predicate does not refer to the Oracle table partitioning key, OTA4H generates as many tasks as partitions in the table.

Security

OTA4H ensures secure access to Oracle database through JDBC authentication, encryption and data integrity, and the JVM system properties.

» Simple and strong authentication. The value of *oracle.hcat.osh.authentication* property in the external table declaration i.e., `SIMPLE, KERBEROS, WALLET`[6] specifies the authentication type.
Example: `'oracle.hcat.osh.authentication' = 'KERBEROS'`. For Oracle wallets, the OraclePKI provider is used and OTA4H ships the required jars including *oraclepki.jar, osdt_cert.jar and osdt_core.jar*.
Depending on the type of authentication, additional properties must be set (see the OTA4H chapter in the BDA doc for more details).

» Encryption and Integrity. OTA4H enforces all *oracle.net.** and *oracle.jdbc.** table properties related to encryption and data integrity. The "JDBC-Thin Driver Support for Encryption and Integrity" section in the Oracle JDBC developer's guide[7] has more details.

» JVM system properties, Hive/Hadoop/Spark environment variables or configuration properties must also be set. Example: `java.security.krb5.conf` must be set using `HADOOP_OPTS` or `mapred.child.java.opts` in `mapred-site.xml` for child JVMs `or -Djava.security.krb5.conf=<path for kerberos config>;` Hive read the value of `KRB5_CONFIG`.

» Cloudera Manager may also be used to set all these properties.

High-Availability

RAC awareness. In RAC database environments, OTA4H inherits RAC support in Oracle JDBC. The Oracle database listener will route the Hadoop or Spark tasks connection requests to the available nodes either randomly or based on runtime load balancing (if enabled by the DBA), provided the RAC-aware service name is specified in the JDBC URL.

[4] hive.io.file.read.all.columns connection property set to false by default.
[5] As of Hive-1.1, hive.optimize.ppd configuration property is set to true by default.
[6] For this release, only WALLET SSL is supported; other SSL certificate containers including JKS, PKCS12, and so on will be supported in future releases.
[7] http://docs.oracle.com/database/121/JJDBC/clntsec.htm#JJDBC28295

Consistency

The maximum number of splits generated by OTA4H is controlled by oracle.hcat.osh.maxSplits however, the number of Hadoop or Spark tasks connecting simultaneously may be limited on the database-side by the DBA (DRCP's maxconnections) or on Hadoop/Spark side by `mapred.tasktracker.map.tasks.maximum` in `conf/mapred-site.xml` or `spark.dynamicAllocation.enabled`. In all cases, OTA4H ensures that the Hadoop or Spark tasks execute all the splits of the query as of the same Oracle database's System Commit Number (SCN).

Conclusion

Big Data Analytics requires the integration of Master Data with Big Data. Oracle Table Access for Hadoop and Spark (OTA4H), a new feature of Oracle Big Data Appliance allows direct access to Oracle tables by turning them into Hadoop or Spark data-sources. OTA4H is a fast, scalable, secure, and reliable implementation of Apache StorageHandler and InputFormat interfaces for the Oracle database.

Kuassi Mensah
Oracle Corporation
400 Oracle Parkway
94065, Redwood Shores, CA
USA

Phone: +1 415-8069937
Fax: +1 650-5067525
Email kuassi.mensah@oracle.com
Internet: @kmensah, db360.blogspot.com, https://www.facebook.com/kuassi.mensah and https://www.linkedin.com/in/kmensah