OJAI Query: How to force query execution to choose query service(Drill) path. (set OJAI configuration)

This article talks about how to force OJAI query to choose query service (Drill) path to execute.

Jan 12, 2018 · How do I

Task or use-case

Generally, the execution paths used depend on whether queries originate from the Drill SQL or OJAI. Queries originating from OJAI either directly access MapR-DB JSON or leverage the OJAI Query Service. Queries originating from Drill SQL leverage the full functionality of MapR Drill. MapR automatically chooses the query execution path. The automated strategy sometimes redirects the query to Query Services (for example, when reading a large amount of data or requires sorting a large data set , the total cost of which can turn out to be higher. If for rare cases, where the automated choice is more expensive, it is possible to force the execution path through Drill.

To determine the query execution path for OJAI Queries, you can use the following documentation section:
https://maprdocs.mapr.com/home/MapR-DB/Indexes/DeterminingQueryExecutionPath.html (https://maprdocs.mapr.com/home/MapR-DB/Indexes/DeterminingQueryExecutionPath.html)

This article provides an outline of a procedure to force the execution path through Drill.

Environment

MapR 6.0 and above

- mapr-core-6.0.0.20171109191718.GA (http://mapr-core-6.0.0.20171109191718.GA)-1.x86_64
- mapr-drill-internal-1.11.0.201711161142-1.noarch (http://mapr-drill-internal-1.11.0.201711161142-1.noarch)
- mapr-drill-1.11.0.201711161142-1.noarch (http://mapr-drill-1.11.0.201711161142-1.noarch)

And tested on MAPRSASL security enabled cluster.

Solution

Consider the following OJAI query:-

```
final Query query = connection.newQuery (http://connection.newQuery)()
        .select("_id", "customer_context.ecn")
         .where("{\"$gt\": {\"customer_context.ecn\":\"1010700\"}}")
         .where("{\"$lt\": {\"customer_context.ecn\":\"1011300\"}}")
         .setOption(OjaiOptions.OPTION_FORCE_DRILL, true)
         .limit(20000)
        .build();
```

setOption(OjaiOptions.OPTION_FORCE_DRILL, true) will force it to go through Drill:

To verify:-

```
OjaiQueryProperties oqp = ((OjaiQueryProperties) stream);
System.out.println (http://System.out.println)(oqp.getQueryPath (http://oqp.getQueryPath)().toString());
```

The outcome of this above should print **DRILL;** otherwise it prints **DIRECT.**

Alternatives

For running the OJAI query JVM should be launched with below **JVM options:**

```
export CLASSPATH=.:$MAPR_HOME/lib/*:$MAPR_HOME/conf
export JAVA_OPTS="$JAVA_OPTS -Ddrill.customAuthFactories=org.apache.drill.exec.rpc.security.maprsasl.MapRSaslFactory \
-Dzookeeper.sasl.client=false \
-Djava.security.auth.login.config=/opt/mapr/conf/mapr.login.conf";
$JAVA -cp .:`mapr clientclasspath` $JAVA_OPTS TWF2 1011280 1011300 N
```

Shell Script Code :

```
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.151-1.b12.el7_4.x86_64
echo "JAVA_HOME="$JAVA_HOME
export PWD=`pwd`

export MAPR_TICKETFILE_LOCATION=/tmp/maprticket_0
#export MAPR_IMPERSONATION_ENABLED=1
export MAPR_HOME=/opt/mapr
echo "MAPR_HOME="$MAPR_HOME
export MAPR_MAPREDUCE_MODE=default
echo "MAPR_MAPREDUCE_MODE="$MAPR_MAPREDUCE_MODE

export JAVAC=$JAVA_HOME/bin/javac
export JAVA=$JAVA_HOME/jre/bin/java
export JAVA_OPTS="$JAVA_OPTS -Ddrill.customAuthFactories=org.apache.drill.exec.rpc.security.maprsasl.MapRSaslFactory \
-Dzookeeper.sasl.client=false \
-Djava.security.auth.login.config=/opt/mapr/conf/mapr.login.conf"

rm *.class
$JAVAC *.java
echo "running command: "$JAVA $JAVA_OPTS TWF2

echo "MAPR_MAPREDUCE_MODE="$MAPR_MAPREDUCE_MODE

echo `mapr clientclasspath`

$JAVA -cp .:`mapr clientclasspath` $JAVA_OPTS TWF2 1011280 1011300 N
```

Java code:

```
import com.mapr.db.MapRDB (http://com.mapr.db.MapRDB);
import com.mapr.db.Table;
import org.ojai.store.QueryCondition;
import com.mapr.ojai.store.impl.OjaiOptions;
import org.ojai.Document (http://org.ojai.Document);
import org.ojai.DocumentStream (http://org.ojai.DocumentStream);
import org.ojai.store.Connection (http://org.ojai.store.Connection);
import org.ojai.store.DocumentStore (http://org.ojai.store.DocumentStore);
import org.ojai.store.DriverManager;
import org.ojai.store.Query;
import java.util.ArrayList (http://java.util.ArrayList);
import java.util.Arrays (http://java.util.Arrays);
import java.util.List (http://java.util.List);
import org.apache.log4j.BasicConfigurator (http://org.apache.log4j.BasicConfigurator);
import org.apache.log4j.Level;
import org.apache.log4j.Logger;
import java.util.Map (http://java.util.Map);
import com.mapr.db.impl.OjaiQueryProperties;
import java.util.concurrent.Callable (http://java.util.concurrent.Callable);
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.Future;

public class TWF2 {

  public static void main(final String[] args) throws Exception {

    System.out.println (http://System.out.println)("==== Start Application ===");

     //BasicConfigurator.configure (http://BasicConfigurator.configure)();
     //Logger.getRootLogger (http://Logger.getRootLogger)().setLevel(Level.TRACE (http://Level.TRACE));

    // Get command line parameters.
    String lowerAmt = args[0];
    String higherAmt = args[1];
    String doOutput = args[2];

    // Create an OJAI connection to MapR cluster
    final Connection connection = DriverManager.getConnection (http://DriverManager.getConnection)("ojai:mapr:");

    // Get an instance of OJAI DocumentStore
    final DocumentStore store = connection.getStore (http://connection.getStore)("/fas/event_data_new_4");

    // Build an OJAI query with the condition specified as a JSON string
    final Query query = connection.newQuery (http://connection.newQuery)()
        .select("_id", "customer_context.ecn")
         .where("{\"$gt\": {\"customer_context.ecn\":" + lowerAmt + " }}")
        .where("{\"$lt\": {\"customer_context.ecn\":" + higherAmt +"}}")
        .setOption(OjaiOptions.OPTION_FORCE_DRILL, true)
        .limit(20000)
 System.out.println (http://System.out.println)("query is" + query.toString (http://query.toString)());
   long amtQueryStart = System.currentTimeMillis (http://System.currentTimeMillis)();
   // fetch all OJAI Documents from this store
   final DocumentStream stream = store.findQuery (http://store.findQuery)(query);

   OjaiQueryProperties oqp = ((OjaiQueryProperties) stream);
    String idxused = oqp.getIndexUsed (http://oqp.getIndexUsed)();
    System.out.println (http://System.out.println)(oqp.getQueryPath (http://oqp.getQueryPath)().toString()); // DIRECT or DRILL
List<String> listIds = new ArrayList<String>();

   long amtQueryEnd = System.currentTimeMillis (http://System.currentTimeMillis)();
   int numRecs = 0;
   float totalAmt = 0;
   long amtOutputStart= System.currentTimeMillis (http://System.currentTimeMillis)();
   for (final Document userDocument : stream) {
     // Print the OJAI Document
     numRecs++;
     totalAmt = Float.parseFloat (http://Float.parseFloat)(userDocument.getString (http://userDocument.getString)("customer_context.ecn")) + totalAmt;
     listIds.add (http://listIds.add)(userDocument.getIdString (http://userDocument.getIdString)());
     if (doOutput.equals("Y")) {
         System.out.println (http://System.out.println)(userDocument.asJsonString (http://userDocument.asJsonString)());
         System.out.println (http://System.out.println)(userDocument.getIdString (http://userDocument.getIdString)());
     }
   }

   long amtOutputEnd = System.currentTimeMillis (http://System.currentTimeMillis)();

   System.out.println (http://System.out.println) ("================================================" );
   System.out.println (http://System.out.println) ("Query fetch start: " + amtQueryStart );
   System.out.println (http://System.out.println) ("Query fetch end: " + amtQueryEnd);
   System.out.println (http://System.out.println) ("Query fetch duration millisec: " + (amtQueryEnd - amtQueryStart));
   System.out.println (http://System.out.println) ("Query output start: " + amtOutputStart );
   System.out.println (http://System.out.println) ("Query output end: " + amtOutputEnd );
   System.out.println (http://System.out.println) ("Query output duration millisec: " + (amtOutputEnd - amtOutputStart));
   System.out.println (http://System.out.println) ("================================================" );
   System.out.println (http://System.out.println)("Lower bound query criterion: " + lowerAmt);
   System.out.println (http://System.out.println)("Higher bound query criterion: " + higherAmt);
  System.out.println (http://System.out.println)("Records fetched: " + numRecs);
   System.out.println (http://System.out.println)("Total dollar amount: " + totalAmt);
   System.out.println (http://System.out.println) ("================================================" );
}
```