Module 8: Advance HBase

Practical Doc

1. Start HBase shell, Run below command in HBase shell:

   create 'customers','info'

2. Copy the custs file on HDFS:

   hdfs dfs -put custs

3. Run the below commands in the terminal (web console):

HADOOP_CLASSPATH=`${HBASE_HOME}/bin/hbase classpath` hadoop jar /opt/cloudera/parcels/CDH/lib/hbase/hbase-server-1.2.0-cdh5.11.1.jar importtsv -Dimporttsv.separator=, -Dimporttsv.bulk.output=output -Dimporttsv.columns=HBASE_ROW_KEY,info:id,info:fname,info:lname,info:age,info:prof customers custs

HADOOP_CLASSPATH=`${HBASE_HOME}/bin/hbase classpath` hadoop jar /opt/cloudera/parcels/CDH/lib/hbase/hbase-server-1.2.0-cdh5.11.1.jar completebulkload output customers

4. Run Hbase shell, run below command. You should get all the rows which got loaded n the table:

   scan 'customers'

5) Below are the codes that you can run in the cloud lab by providing the CLASSPATH of all required Hadoop and HBase jars:

- Hadoop Jars Path:
  a. /opt/cloudera/parcels/CDH/lib/hbase/lib/
  b. /opt/cloudera/parcels/CDH/lib/hadoop/lib/

- HBase Jars Path:
  a. opt/cloudera/parcels/CDH/lib/hbase/lib/

Example Java Command:

- Compile: javac -cp .:/opt/cloudera/parcels/CDH/lib/hbase/lib/*:/opt/cloudera/parcels/CDH/lib/hadoop/lib/*:/opt/cloudera/parcels/CDH/lib/hadoop/client/* Example.java
- Execute: java -cp .:/opt/cloudera/parcels/CDH/lib/hbase/lib/*:/opt/cloudera/parcels/CDH/lib/hadoop/lib/*:/opt/cloudera/parcels/CDH/lib/hadoop/client/* ExampleClass

```
*****************************************
GetListExample Example of retrieving data from HBase using lists of Get instances
*****************************************
package hbase;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.KeyValue;
import org.apache.hadoop.hbase.client.Get;
import org.apache.hadoop.hbase.client.HTable;
import org.apache.hadoop.hbase.client.Result;
import org.apache.hadoop.hbase.util.Bytes;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class GetListExample {

 public static void main(String[] args) throws IOException {
  Configuration conf = HBaseConfiguration.create();
 conf.set("hbase.zookeeper.quorum", "ip-20-0-31-249.ec2.internal");
 conf.set("hbase.zookeeper.property.clientPort", "2181");
  HTable table = new HTable(conf, "customers");

  // GetListExample
  byte[] cf1 = Bytes.toBytes("info");
  byte[] qf1 = Bytes.toBytes("fname");
  byte[] qf2 = Bytes.toBytes("lname");
  byte[] row1 = Bytes.toBytes("4005000");
  byte[] row2 = Bytes.toBytes("4009000");
```

```
List<Get> gets = new ArrayList<Get>();

Get get1 = new Get(row1);
get1.addColumn(cf1, qf1);
gets.add(get1);

Get get2 = new Get(row2);
get2.addColumn(cf1, qf1);
gets.add(get2);

Get get3 = new Get(row2);
get3.addColumn(cf1, qf2);
gets.add(get3);

Result[] results = table.get(gets);

System.out.println("First iteration...");
for (Result result : results) {
 String row = Bytes.toString(result.getRow());
 System.out.print("Row: " + row + " ");
 byte[] val = null;
 if (result.containsColumn(cf1, qf1)) {
  val = result.getValue(cf1, qf1);
  System.out.println("Value: " + Bytes.toString(val));
 }
 if (result.containsColumn(cf1, qf2)) {
  val = result.getValue(cf1, qf2);
  System.out.println("Value: " + Bytes.toString(val));
 }
}
```

```
   System.out.println("Second iteration...");
   for (Result result : results) {
    for (KeyValue kv : result.raw()) {
      System.out.println("Row: " + Bytes.toString(kv.getRow()) +
        " Value: " + Bytes.toString(kv.getValue()));
    }
   }
   // GetListExample
 }
}
```

****

PutListExample Example inserting data into HBase using a list

****


```
create 'testtable','colfam1'
```

```
package hbase;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.client.HTable;
import org.apache.hadoop.hbase.client.Put;
import org.apache.hadoop.hbase.util.Bytes;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class PutListExample {
```

```
public static void main(String[] args) throws IOException {

Configuration conf = HBaseConfiguration.create();

conf.set("hbase.zookeeper.quorum", "ip-20-0-31-249.ec2.internal");
conf.set("hbase.zookeeper.property.clientPort", "2181");

HTable table = new HTable(conf, "testtable");



List<Put> puts = new ArrayList<Put>();


Put put1 = new Put(Bytes.toBytes("row1"));

put1.add(Bytes.toBytes("colfam1"), Bytes.toBytes("qual1"),

Bytes.toBytes("val1"));

puts.add(put1);


Put put2 = new Put(Bytes.toBytes("row2"));

put2.add(Bytes.toBytes("colfam1"), Bytes.toBytes("qual1"),

Bytes.toBytes("val2"));

puts.add(put2);


Put put3 = new Put(Bytes.toBytes("row2"));

put3.add(Bytes.toBytes("colfam1"), Bytes.toBytes("qual2"),

Bytes.toBytes("val3"));

puts.add(put3);


table.put(puts);


}

}
```

****

RowFilterExample Example using a filter to select specific rows

****

```
package hbase;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.client.HTable;
import org.apache.hadoop.hbase.client.Result;
import org.apache.hadoop.hbase.client.ResultScanner;
import org.apache.hadoop.hbase.client.Scan;
import org.apache.hadoop.hbase.filter.BinaryComparator;
import org.apache.hadoop.hbase.filter.CompareFilter;
import org.apache.hadoop.hbase.filter.Filter;
import org.apache.hadoop.hbase.filter.RegexStringComparator;
import org.apache.hadoop.hbase.filter.RowFilter;
import org.apache.hadoop.hbase.filter.SubstringComparator;
import org.apache.hadoop.hbase.util.Bytes;

import java.io.IOException;

public class RowFilterExample {

 public static void main(String[] args) throws IOException {
  Configuration conf = HBaseConfiguration.create();
 conf.set("hbase.zookeeper.quorum", "ip-20-0-31-249.ec2.internal");
 conf.set("hbase.zookeeper.property.clientPort", "2181");
  HTable table = new HTable(conf, "customers");

  Scan scan = new Scan();
  scan.addColumn(Bytes.toBytes("info"), Bytes.toBytes("fname"));
```

```
Filter filter1 = new RowFilter(CompareFilter.CompareOp.LESS_OR_EQUAL,
  new BinaryComparator(Bytes.toBytes("4000010")));
scan.setFilter(filter1);
ResultScanner scanner1 = table.getScanner(scan);

System.out.println("Scanning table #1...");

for (Result res : scanner1) {
  System.out.println(res);
}
scanner1.close();

Filter filter2 = new RowFilter(CompareFilter.CompareOp.EQUAL,
  new RegexStringComparator("40000 ?(09|12|15)"));
scan.setFilter(filter2);
ResultScanner scanner2 = table.getScanner(scan);

System.out.println("Scanning table #2...");

for (Result res : scanner2) {
  System.out.println(res);
}
scanner2.close();

Filter filter3 = new RowFilter(CompareFilter.CompareOp.EQUAL,
  new SubstringComparator("5555"));
scan.setFilter(filter3);
ResultScanner scanner3 = table.getScanner(scan);

System.out.println("Scanning table #3...");
```

```
   for (Result res : scanner3) {
     System.out.println(res);
   }
   scanner3.close();

 }
}
```

****

Other Filters

****


```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.HColumnDescriptor;
import org.apache.hadoop.hbase.HTableDescriptor;
import org.apache.hadoop.hbase.KeyValue;
import org.apache.hadoop.hbase.client.Get;
import org.apache.hadoop.hbase.client.HBaseAdmin;
import org.apache.hadoop.hbase.client.HTable;
import org.apache.hadoop.hbase.client.Put;
import org.apache.hadoop.hbase.client.Result;
import org.apache.hadoop.hbase.client.ResultScanner;
import org.apache.hadoop.hbase.client.Scan;
import org.apache.hadoop.hbase.filter.BinaryComparator;
import org.apache.hadoop.hbase.filter.CompareFilter;
import org.apache.hadoop.hbase.filter.FamilyFilter;
import org.apache.hadoop.hbase.filter.Filter;
```

```
import org.apache.hadoop.hbase.filter.QualifierFilter;
import org.apache.hadoop.hbase.filter.SubstringComparator;
import org.apache.hadoop.hbase.filter.ValueFilter;
import org.apache.hadoop.hbase.util.Bytes;

import java.io.IOException;
import java.util.Random;

public class FamilyFilterExample {
        private static Configuration conf = HBaseConfiguration.create();
          conf.set("hbase.zookeeper.quorum", "ip-20-0-31-249.ec2.internal");
          conf.set("hbase.zookeeper.property.clientPort", "2181");

        public static void disableTable(String table) throws IOException {
                HBaseAdmin admin =  new HBaseAdmin(conf);
                admin.disableTable(table);
        }

        public static void dropTable(String table) throws IOException {
                HBaseAdmin admin =  new HBaseAdmin(conf);
                if (admin.tableExists(table)) {
                                disableTable(table);
                                admin.deleteTable(table);
                }
        }

        public static void createTable(String table, String... colfams)
                        throws IOException {
                HBaseAdmin admin =  new HBaseAdmin(conf);
                HTableDescriptor desc = new HTableDescriptor(table);
                for (String cf : colfams) {
```

```
                HColumnDescriptor coldef = new HColumnDescriptor(cf);
                desc.addFamily(coldef);
            }

            admin.createTable(desc);
        }

        public static void fillTable(String table, int startRow, int endRow, int numCols,
                int pad, boolean setTimestamp, boolean random,String... colfams) throws
IOException {
                HTable tbl = new HTable(conf, table);
                Random rnd = new Random();
                for (int row = startRow; row <= endRow; row++) {
                        for (int col = 0; col < numCols; col++) {
                                Put put = new Put(Bytes.toBytes("row-" + padNum(row, pad)));
                                for (String cf : colfams) {
            String colName = "col-" + padNum(col, pad);
            String val = "val-" + (random ?
            Integer.toString(rnd.nextInt(numCols)) :
            padNum(row, pad) + "." + padNum(col, pad));
            if (setTimestamp) {
            put.add(Bytes.toBytes(cf), Bytes.toBytes(colName),
                                col, Bytes.toBytes(val));
                                } else {
                                put.add(Bytes.toBytes(cf), Bytes.toBytes(colName),
                                Bytes.toBytes(val));
                                        }
                                }
                                tbl.put(put);
                        }
                }
```

```
                tbl.close();
        }

        public static String padNum(int num, int pad) {
                String res = Integer.toString(num);
                if (pad > 0) {
                while (res.length() < pad) {
                res = "0" + res;
                }
                }
                return res;
                }


 public static void main(String[] args) throws IOException {
  Configuration conf = HBaseConfiguration.create();
  conf.set("hbase.zookeeper.quorum", "ip-20-0-31-249.ec2.internal");
 conf.set("hbase.zookeeper.property.clientPort", "2181");
  FamilyFilterExample.dropTable("testtable");
  FamilyFilterExample.createTable("testtable", "colfam1", "colfam2", "colfam3", "colfam4");
  System.out.println("Adding rows to table...");
  FamilyFilterExample.fillTable("testtable", 1, 10, 2, -1,false,false,"colfam1", "colfam2", "colfam3",
"colfam4");

  HTable table = new HTable(conf, "testtable");

  System.out.println("*****************************************************************");
  System.out.println("****using a filter to include only specific column families****");
  System.out.println("*****************************************************************");
```

```
Filter filter1 = new FamilyFilter(CompareFilter.CompareOp.LESS,
  new BinaryComparator(Bytes.toBytes("colfam2")));

Scan scan = new Scan();
scan.setFilter(filter1);
ResultScanner scanner = table.getScanner(scan);
System.out.println("Scanning table... ");
for (Result result : scanner) {
  System.out.println(result);
}
scanner.close();

Get get1 = new Get(Bytes.toBytes("row-5"));
get1.setFilter(filter1);
Result result1 = table.get(get1);
System.out.println("Result of get(): " + result1);

Filter filter2 = new FamilyFilter(CompareFilter.CompareOp.EQUAL,
  new BinaryComparator(Bytes.toBytes("colfam3")));
Get get2 = new Get(Bytes.toBytes("row-5"));
get2.addFamily(Bytes.toBytes("colfam1"));
get2.setFilter(filter2);
Result result2 = table.get(get2);
System.out.println("Result of get(): " + result2);


System.out.println("*****************************************************************");
System.out.println("***using a filter to include only specific column qualifiers***");
System.out.println("*****************************************************************");

Filter filter3 = new QualifierFilter(CompareFilter.CompareOp.LESS_OR_EQUAL,
```

```
                new BinaryComparator(Bytes.toBytes("col-2")));

        Scan scan1 = new Scan();
        scan1.setFilter(filter3);
        ResultScanner scanner1 = table.getScanner(scan1);

System.out.println("Scanning table... ");

for (Result result : scanner1) {
        System.out.println(result);
        }
        scanner1.close();

        Get get = new Get(Bytes.toBytes("row-5"));
get.setFilter(filter3);
Result result = table.get(get);
System.out.println("Result of get(): " + result);

System.out.println("*****************************************************************");
System.out.println("******************using the value based filter******************");
System.out.println("*****************************************************************");

Filter filter = new ValueFilter(CompareFilter.CompareOp.EQUAL, new SubstringComparator(".0"));

        Scan scan11 = new Scan();
        scan11.setFilter(filter);
        ResultScanner scanner11 = table.getScanner(scan11);
System.out.println("Results of scan:");
for (Result result11 : scanner11) {
        for (KeyValue kv : result11.raw()) {
                System.out.println("KV: " + kv + ", Value: " + Bytes.toString(kv.getValue()));
```

```
            }
    }
    scanner11.close();

    Get get11 = new Get(Bytes.toBytes("row-5"));
    get11.setFilter(filter);
    Result result11 = table.get(get11);
    System.out.println("Result of get: ");
    for (KeyValue kv : result11.raw()) {
            System.out.println("KV: " + kv + ", Value: " +
            Bytes.toString(kv.getValue()));
            }
  }
}
```