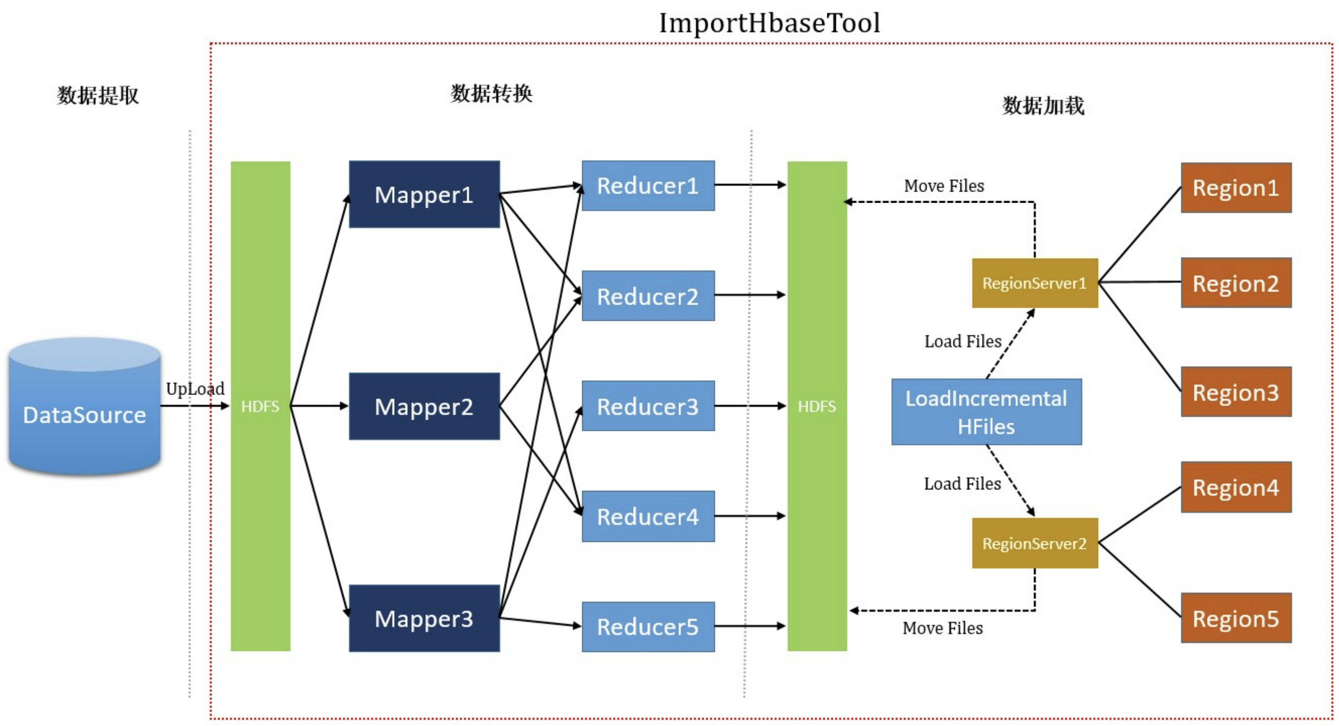
文档3-Hive2Hbase

1.基本原理

思路:

Bulkload 就是利用 HBase 的数据信息按照特定格式存储在 HDFS 内这一原理，直接在 HDFS 中生成持久化的 HFile 数据格式文件，然后上传至合适位置，即完成巨量数据快速入库的办法。



应用场景:

- 不影响线上业务的同时，巨量数据快速入库
- 适合场景HDFS数据快速导入

2.代码示例

2.1 pom引入

Pom文件

```
  <dependency>
     <groupId>org.apache.hbase</groupId>
       <artifactId>hbase-mapreduce</artifactId>
       <version>2.1.0-cdh6.3.2</version>
       <scope>compile</scope>
       <exclusions>
          <exclusion>
              <artifactId>hbase-server</artifactId>
              <groupId>org.apache.hbase</groupId>
           </exclusion>
       </exclusions>
  </dependency>

  <dependency>
     <groupId>org.apache.hbase</groupId>
     <artifactId>hbase-server</artifactId>
     <version>2.1.0-cdh6.3.2</version>
  </dependency>

  <dependency>
     <groupId>org.apache.phoenix</groupId>
     <artifactId>phoenix-core</artifactId>
     <version>5.0.0-HBase-2.0</version>
     <exclusions>
       <exclusion>
         <artifactId>hbase-server</artifactId>
         <groupId>org.apache.hbase</groupId>
       </exclusion>
     </exclusions>
   </dependency>
```

2.2 代码示例

MR Hbase BulkLoad

```
package com.baijiahulian.bdg;

import org.apache.commons.lang.StringUtils;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.hbase.Cell;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.KeyValue;
import org.apache.hadoop.hbase.TableName;
import org.apache.hadoop.hbase.client.*;
import org.apache.hadoop.hbase.io.ImmutableBytesWritable;
import org.apache.hadoop.hbase.mapreduce.HFileOutputFormat2;
import org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles;
import org.apache.hadoop.hbase.util.Bytes;
import org.apache.hadoop.hbase.util.MapReduceExtendedCell;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.security.UserGroupInformation;
import org.apache.hadoop.util.GenericOptionsParser;
import org.apache.phoenix.schema.types.PInteger;
import org.apache.phoenix.schema.types.PLong;

import java.io.IOException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.PrivilegedExceptionAction;

public class HbaseBulkLoadMRV3
{
    // ### ZK
    public static final String ZK = "al-bj-bigdata-inf-hbase-test03.inf.bdg.baijiahulian:2181,al-bj-bigdata-inf-
hbase-test02.inf.bdg.baijiahulian:2181,al-bj-bigdata-inf-hbase-test01.inf.bdg.baijiahulian:2181";

    // ###
    public static final String SPLIT_SYMBOL = "\001";

    // ###
    public static final String CF = "CF";

    // ###
    public static final Log LOG = LogFactory.getLog(HbaseBulkLoadMRV3.class);

    // ### Hive NULL
    public static final String HIVE_NULL_CONSTANT = "\\N";

    // ###
    public static final String EXEC_USER = "hbase";

    // ### MapClass
    public static class MapClass extends Mapper<LongWritable, Text, ImmutableBytesWritable, Cell>
    {
        // [0] - RowKey
        private String rowKey;

        // [1] - id
        private String id;

        // [2] -
        private String assistantNumber;

        // [3] -
        // 2020-06-25 19:00:41
        private String createTime;

        // [4] - userId
        private String recordType;

        // [5] -
        private String content;

        // [6] - ID
        private String targetUserId;

        // [7] -
        private String subclazzNumber;

        // [8] -
        private String assistantName;

        // Hbase
        private ImmutableBytesWritable outKey;
        private Cell cell;

        // phoenixLong
        PLong phoenixLong;
        PInteger pInteger;

        // MD5
        MessageDigest md;

        @Override
        protected void setup(Context context) throws IOException, InterruptedException {
            super.setup(context);
            try {
                md = MessageDigest.getInstance("MD5");
                phoenixLong = PLong.INSTANCE;
                pInteger = PInteger.INSTANCE;
            } catch (NoSuchAlgorithmException e) {
                e.printStackTrace();
            }
        }

        private String md5RowKey(String rowKey) {
            md.update(rowKey.getBytes());
            byte b[] = md.digest();
            int i;
            StringBuffer buf = new StringBuffer("");
            for (int offset = 0; offset < b.length; offset++) {
                i = b[offset];
                if (i < 0)
                    i += 256;
                if (i < 16)
                    buf.append("0");
                buf.append(Integer.toHexString(i));
            }
            return buf.toString().substring(8, 24);
        }

        public void map(LongWritable inKey, Text inValue, Context context) throws ClassCastException
        {
            // ###
            String[] infos = inValue.toString().split(SPLIT_SYMBOL);

            try
            {
                // ###
                id = infos[0];
                assistantNumber = infos[1];
                createTime = infos[2];
                recordType = infos[3];
                content = infos[4];
                targetUserId = infos[5];
                subclazzNumber = infos[6];
                assistantName = infos[7];

                // ###
                String msg = String.format("id=%s,rowkey=%s,assistantNumber=%s,createTime=%s,recordType=%s,
content=%s,targetUserId=%s,subclazzNumber=%s,assistantName=%s",
                        id,rowKey,assistantNumber,createTime,recordType,content,targetUserId,subclazzNumber,
assistantName);
                //
                if(LOG.isInfoEnabled()){
                    LOG.info(msg);
                }

                // ###
                if (!StringUtils.isBlank(id) && !StringUtils.equals(id, HIVE_NULL_CONSTANT))
                {
                    // rowkey
                    rowKey = md5RowKey(id);

                    // ### rowkey
                    outKey = new ImmutableBytesWritable(Bytes.toBytes(rowKey));
                    cell = new KeyValue(Bytes.toBytes(rowKey), Bytes.toBytes(CF), Bytes.toBytes("ID"),
phoenixLong.toBytes(Long.parseLong(id)));
                    context.write(outKey, new MapReduceExtendedCell(cell));

                    if (!(StringUtils.equals(assistantNumber, HIVE_NULL_CONSTANT)|| StringUtils.isBlank
(assistantNumber))) {
                        // ###
                        cell = new KeyValue(Bytes.toBytes(rowKey), Bytes.toBytes(CF), Bytes.toBytes
("ASSISTANT_NUMBER"), phoenixLong.toBytes(Long.parseLong(assistantNumber)));
                        context.write(outKey, new MapReduceExtendedCell(cell));
                    }
                    // ###
                    if (!(StringUtils.equals(createTime, HIVE_NULL_CONSTANT) || StringUtils.isBlank
(createTime))) {
                        cell = new KeyValue(Bytes.toBytes(rowKey), Bytes.toBytes(CF), Bytes.toBytes
("CREATE_TIME"), Bytes.toBytes(createTime));
                        context.write(outKey, new MapReduceExtendedCell(cell));
                    }

                    // ###
                    if (!(StringUtils.equals(recordType, HIVE_NULL_CONSTANT) || StringUtils.isBlank
(createTime))) {
                        cell = new KeyValue(Bytes.toBytes(rowKey), Bytes.toBytes(CF), Bytes.toBytes
("RECORD_TYPE"), pInteger.toBytes(Integer.parseInt(recordType)));
                        context.write(outKey, new MapReduceExtendedCell(cell));
                    }

                    // ###
                    if (!(StringUtils.equals(content, HIVE_NULL_CONSTANT) || StringUtils.isBlank(content))) {
                        cell = new KeyValue(Bytes.toBytes(rowKey), Bytes.toBytes(CF), Bytes.toBytes("CONTENT"),
Bytes.toBytes(content));
                        context.write(outKey, new MapReduceExtendedCell(cell));
                    }

                    // ### target user id
                    if (!(StringUtils.equals(targetUserId, HIVE_NULL_CONSTANT) || StringUtils.isBlank
(targetUserId))) {
                        cell = new KeyValue(Bytes.toBytes(rowKey), Bytes.toBytes(CF), Bytes.toBytes
("TARGET_USER_ID"), phoenixLong.toBytes(Long.parseLong(targetUserId)));
                        context.write(outKey, new MapReduceExtendedCell(cell));
                    }

                    // ### subclazzNumber
                    if (!(StringUtils.equals(subclazzNumber, HIVE_NULL_CONSTANT) || StringUtils.isBlank
(subclazzNumber))) {
                        cell = new KeyValue(Bytes.toBytes(rowKey), Bytes.toBytes(CF), Bytes.toBytes
("SUBCLAZZ_NUMBER"), phoenixLong.toBytes(Long.parseLong(subclazzNumber)));
                        context.write(outKey, new MapReduceExtendedCell(cell));
                    }

                    // ### assistantName
                    if (!(StringUtils.equals(assistantName, HIVE_NULL_CONSTANT) || StringUtils.isBlank
(assistantName))) {
                        cell = new KeyValue(Bytes.toBytes(rowKey), Bytes.toBytes(CF), Bytes.toBytes
("ASSISTANT_NAME"), Bytes.toBytes(assistantName));
                        context.write(outKey, new MapReduceExtendedCell(cell));
                    }
                }

            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }

    // ### Main
    public static void main(String[] args) throws Exception
    {
        // ###
        Configuration conf = new Configuration();
        conf.set("hbase.zookeeper.quorum", ZK);
        final String[] paramArgs = new GenericOptionsParser(conf, args).getRemainingArgs();

        Job hfileJob = Job.getInstance(conf);
        hfileJob.setJobName("HbaseBulkLoadDemoMR");
        hfileJob.setJarByClass(HbaseBulkLoadMRV3.class);
        hfileJob.setMapperClass(MapClass.class);
        hfileJob.setNumReduceTasks(0);

        hfileJob.setOutputKeyClass(ImmutableBytesWritable.class);
        hfileJob.setOutputKeyClass(MapReduceExtendedCell.class);
        FileInputFormat.addInputPath(hfileJob, new Path(paramArgs[0]));
        FileOutputFormat.setOutputPath(hfileJob, new Path(paramArgs[1]));

        // --------------------------------------------------
        // Hbase
        // --------------------------------------------------

        final Configuration hbaseConfiguration = HBaseConfiguration.create(conf);
        hbaseConfiguration.set("io.serializations", "org.apache.hadoop.io.serializer.WritableSerialization,org.
apache.hadoop.io.serializer.avro.AvroSpecificSerialization,org.apache.hadoop.io.serializer.avro.
AvroReflectSerialization,org.apache.hadoop.hbase.mapreduce.MutationSerialization,org.apache.hadoop.hbase.
mapreduce.ResultSerialization,org.apache.hadoop.hbase.mapreduce.KeyValueSerialization");

        final Connection connection = ConnectionFactory.createConnection(hbaseConfiguration);
        final TableName tableName = TableName.valueOf(paramArgs[2]);
        final Table clazzLessonTable = connection.getTable(tableName);
        HFileOutputFormat2.configureIncrementalLoad(hfileJob, clazzLessonTable, connection.getRegionLocator
(tableName));

        int result = hfileJob.waitForCompletion(true) ? 0 : 1;

        // ###
        UserGroupInformation ugi = UserGroupInformation.createRemoteUser(EXEC_USER);
        try {
            ugi.doAs(new PrivilegedExceptionAction<String>() {
                public String run() throws Exception {

                    Configuration conf2 = new Configuration();
                    conf2.set("hbase.zookeeper.quorum", ZK);
                    conf2.set("hadoop.security.authentication", "simple");

                    Configuration hbaseConfiguration2 = HBaseConfiguration.create(conf2);
                    hbaseConfiguration2.set("io.serializations", "org.apache.hadoop.io.serializer.
WritableSerialization,org.apache.hadoop.io.serializer.avro.AvroSpecificSerialization,org.apache.hadoop.io.
serializer.avro.AvroReflectSerialization,org.apache.hadoop.hbase.mapreduce.MutationSerialization,org.apache.
hadoop.hbase.mapreduce.ResultSerialization,org.apache.hadoop.hbase.mapreduce.KeyValueSerialization");

                    Connection connection2 = ConnectionFactory.createConnection(hbaseConfiguration2);
                    Admin admin2 = connection2.getAdmin();
                    TableName tableName2 = TableName.valueOf(paramArgs[2]);
                    Table clazzLessonTable2 = connection.getTable(tableName);

                    LoadIncrementalHFiles loader = new LoadIncrementalHFiles(hbaseConfiguration2);
                    loader.doBulkLoad(new Path(paramArgs[1]), admin2, clazzLessonTable2, connection2.
getRegionLocator(tableName2));
                    return "null";
                }
            });
        } catch (Exception ex) {
            ex.printStackTrace();
        }

        // ###
        System.exit(result);
    }
}
```

2.3 测试环境

- 测试环境用于进行test\beta测试
- 测试环境zk
    - al-bj-bigdata-inf-hbase-test03.inf.bdg.baijiahulian:2181;al-bj-bigdata-inf-hbase-test02.inf.bdg.baijiahulian:2181;al-bj-bigdata-inf-hbase-test01.inf.bdg.baijiahulian:2181

域名	IP
al-bj-bigdata-inf-hbase-test03.inf.bdg.baijiahulian	172.16.18.71

al-bj-bigdata-inf-hbase-test02.inf.bdg.baijiahulian	172.16.18.69
al-bj-bigdata-inf-hbase-test01.inf.bdg.baijiahulian	172.16.18.68

3.Spark版Hive2HBase

参见Hbase接入指南：wiki.baijia.com/x/DAI5Ag