文档2-Hbase2Hive

1. 基本原理

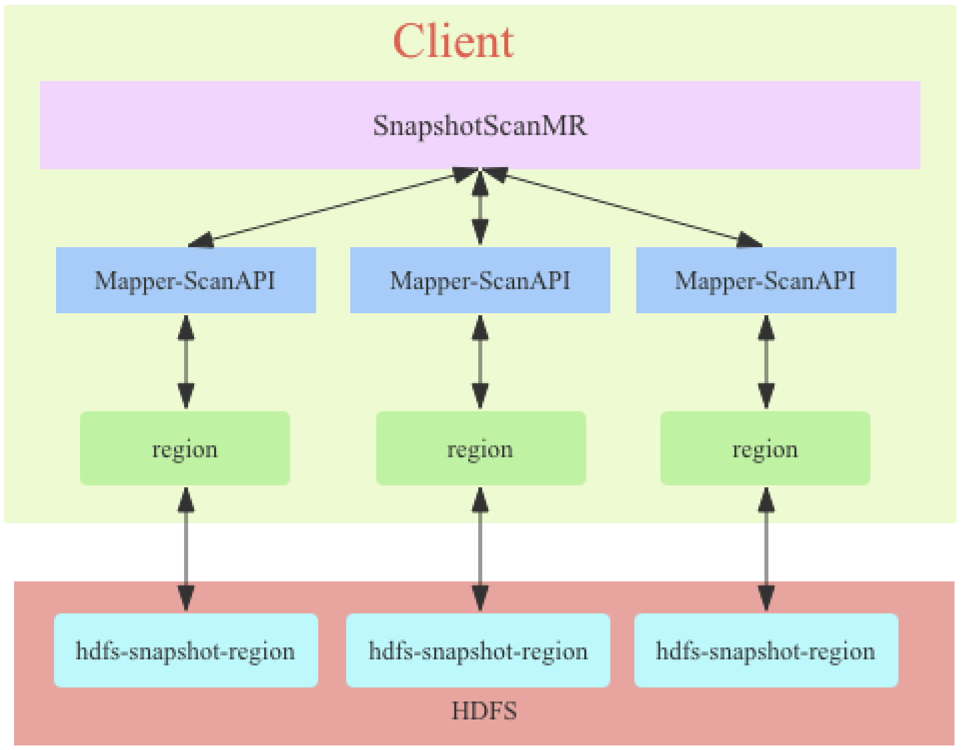
思路:

离线数据获取不影响当前Hbase业务，如果使用传统的数据扫描，将会对于Hbase RegionServer产生较大的性能压力，所以采取**SnapShot**方式扫描Hbase存储的数据

SnapshotScanMR直接会在客户端打开Region扫描HDFS上的文件，不需要发送Scan请求给RegionServer，再由RegionServer扫描HDFS上的文件。

优点:

- 减小对RegionServer的影响，SnapshotScanMR这种绕过RegionServer的实现方式最大限度的减小了对集群中其他业务的影响。
- 极大的提升了扫描效率。SnapshotScanMR相比TableScanMR在扫描效率上会有2倍～N倍的性能提升



应用场景:

- 数据从Hbase导出至HDFS或Hive

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

SnapShotMR

```
package com.baijiahulian.bdg;

import org.apache.commons.lang.ArrayUtils;
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.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.fs.Trash;
import org.apache.hadoop.hbase.Cell;
import org.apache.hadoop.hbase.CellScanner;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.TableName;
import org.apache.hadoop.hbase.client.*;
import org.apache.hadoop.hbase.io.ImmutableBytesWritable;
import org.apache.hadoop.hbase.mapred.TableMap;
import org.apache.hadoop.hbase.mapred.TableMapReduceUtil;
import org.apache.hadoop.hbase.util.Bytes;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.security.UserGroupInformation;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

import java.io.IOException;
import java.security.PrivilegedExceptionAction;
```

```java
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.UUID;

/**
 * ----------------
 * :
 * MR-HbaseDemo-SnapShotHbase
 * <p>
 * :
 * param1:Hbase TableName
 * param2:Hive OutPutPath
 * ---------------
 */
public class ScanSnapShotMR extends ToolRunner implements Tool {
    // ###
    public static final Log LOG = LogFactory.getLog(ScanSnapShotMR.class);

    // ### (Hbase)
    private static final String HDFS_FS_DEFAULTFS = "fs.defaultFS";
    private static final String HDFS_FS_DEFAULTFS_VALUE = "hdfs://xxxx:8020/";

    // ### Hbase-ZKHbaseZK
    private static final String HBASE_NAME = "hbase.zookeeper.quorum";
    private static final String HBASE_VALUE = "xxxx1:2181;xxxx2:2181;xxxx3:2181";

    // ### Hbase-
    private static final String HBASE_ROOT_DIR = "hbase.rootdir";
    private static final String HBASE_ROOT_DIR_VALUE = "hdfs://xxxx:8020/hbase";

    // ### AppMaster
    private static final String YARN_APP_MAPREDUCE_AM_STAGING_DIR = "yarn.app.mapreduce.am.staging-dir";

    // ### Hbase-io.serializations
    private static final String HDFS_IO_SERIALIZATIONS = "io.serializations";
    private static final String HDFS_IO_SERIALIZATIONS_VALUE = "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";

    // ###
    private static final String HBASE_CF = "info";

    // ### StageDir
    private static final String STAGE_DIR_PREFIX = "hdfs://xxxx:8020/tmp/hbase/stagedir/";

    // ### tmpDir
    private static final String TEP_DIR_PREFIX = "hdfs://xxxx:8020/tmp/hbase/region_tmp/";

    // ### trash stage dir
    private static final String STAGE_TRASH_DIR_PREFIX = "hdfs://xxxx:8020/user/hbase_tmp/.Trash/";

    /**
     * SnapShotMapper
     */
    static class TableSnapshotMapper extends MapReduceBase implements TableMap<Text, NullWritable> {
        // ###
        private Text mkey = new Text();

        public void map(ImmutableBytesWritable key, Result result,
                        OutputCollector<Text, NullWritable> collector, Reporter reporter)
                throws IOException {
            // ###
            StringBuffer stringBuffer = new StringBuffer();
            CellScanner cellScanner = result.cellScanner();

            SimpleDateFormat format = new SimpleDateFormat("yyyyMMdd");
            boolean flag = false;
            stringBuffer.append("7");
            // rowkeyCellScanner
            while (cellScanner.advance()) {
                Cell cell = cellScanner.current();
                String qualifier = Bytes.toString(cell.getQualifierArray(), cell.getQualifierOffset(), cell.
getQualifierLength());
                String value = Bytes.toString(cell.getValueArray(), cell.getValueOffset(), cell.
getValueLength());
                try {
                    if ("task_dt".equals(qualifier) && (format.parse(value).getTime() + 14 * 24 * 60 * 60 *
1000) > System.currentTimeMillis()) {
                        flag = true;
                    }
                } catch (ParseException e) {
                    e.printStackTrace();
                }
                if ("biz_type".equals(qualifier)) {
                    continue;
                }
                stringBuffer.append("\u0001" + value);
            }

            if (flag && stringBuffer.toString().split("\u0001").length == 21) {
                mkey.set(stringBuffer.toString());
                collector.collect(mkey, NullWritable.get());
            }
        }
    }

    public int run(String[] args) throws Exception {
        int result = -1;

        Admin admin = null;
        Connection connection = null;
        String stagingDir = StringUtils.EMPTY;
        String trashStagingDir = StringUtils.EMPTY;
        FileSystem fs = null;
        Configuration hbaseConfiguration = null;

        String snapShotName = StringUtils.EMPTY;

        try {
            // ------------------------------------
            // STEP 1. output_dir
            //
            // ------------------------------------

            // ###
            Configuration config = new Configuration();
            setConf(config);

            // ### output_dir
            Path path = new Path(args[1]);
            fs = path.getFileSystem(config);

            // ###
            if (fs.exists(path)) {
                LOG.info("[ScanSnapShotMR] - output_dir is exists");
            }

            // ###
            Trash trashTmp = new Trash(fs, config);
           /* if (trashTmp.moveToTrash(path)) {
                LOG.info("Moved to trash: " + path);
            }*/

            // ------------------------------------
            // STEP 2. HbaseSnapShot
            // ------------------------------------
            hbaseConfiguration = HBaseConfiguration.create(config);
            hbaseConfiguration.set(HDFS_IO_SERIALIZATIONS, HDFS_IO_SERIALIZATIONS_VALUE);
            hbaseConfiguration.set(HBASE_ROOT_DIR, HBASE_ROOT_DIR_VALUE);
            String tableName = args[0];
            //
            connection = ConnectionFactory.createConnection(hbaseConfiguration);
            admin = connection.getAdmin();
            if (admin == null) {
                LOG.error("[ScanSnapShotMR] - Hbase Admin Error");
                return -1;
            }

            //
            boolean exists = admin.tableExists(TableName.valueOf(tableName));
            if (!exists) {
                LOG.error("[ScanSnapShotMR] - Hbase tableName = " + tableName + " is not exists.");
                return -1;
            }

            // snapshot
            snapShotName = tableName.split(":")[1] + "_snapshot" + UUID.randomUUID().toString().replace("-",
"");

            //
            LOG.info("[ScanSnapShotMR] - snapShotName = " + snapShotName);

            try {
                admin.snapshot(snapShotName, TableName.valueOf(tableName));
            } catch (Exception ex) {
                LOG.error("[ScanSnapShotMR] - CloneSnapShot Error", ex);
                return -1;
            }

            // ------------------------------------
            // STEP 3. HbaseSnapShot
            // ------------------------------------
            stagingDir = STAGE_DIR_PREFIX + UUID.randomUUID().toString() + "/";
            trashStagingDir = STAGE_TRASH_DIR_PREFIX + stagingDir;
            hbaseConfiguration.set(YARN_APP_MAPREDUCE_AM_STAGING_DIR, stagingDir);

            // ### job
            JobConf jobConf = new JobConf(hbaseConfiguration);

            // ### SnapShot
            Path tmpPath = new Path(TEP_DIR_PREFIX + tableName.replaceAll(":", "_") + UUID.randomUUID().
toString() + "/");
            Path outPutPath = new Path(args[1]);

            // ### MR
            jobConf.setMapperClass(TableSnapshotMapper.class);
            jobConf.setMapOutputKeyClass(Text.class);
            jobConf.setMapOutputValueClass(NullWritable.class);
            jobConf.setJarByClass(ScanSnapShotMR.class);

            // ------------------------------------
            // STEP 4. MR
            // ------------------------------------
            // ###
            TableMapReduceUtil.initTableSnapshotMapJob(snapShotName,
                    HBASE_CF, TableSnapshotMapper.class, Text.class,
                    NullWritable.class, jobConf, false, tmpPath);

            // ### outputformate
            FileOutputFormat.setOutputPath(jobConf, outPutPath);
            TableMapReduceUtil.addDependencyJars(jobConf);

            // ### job
            RunningJob job = JobClient.runJob(jobConf);
            job.waitForCompletion();

            // ### result
            result = (job.isSuccessful() == true) ? 1 : 0;

        } catch (Exception ex) {
            LOG.error("[ScanSnapShotMR] - ex = {}", ex);
            result = -1;
        } finally {
            try {
                // ###
                Path path = new Path(stagingDir);
                fs = path.getFileSystem(hbaseConfiguration);
                Trash trashTmp = new Trash(fs, hbaseConfiguration);
                // ###
                if (fs.exists(path)) {
                   /* if (trashTmp.moveToTrash(path)) {
                        LOG.info("Moved StagingDir to trash: " + path);
                    }*/
                }
                // ### snapshot
                LOG.info("[ScanSnapShotMR] - SnapShotName = " + snapShotName);
                admin.deleteSnapshot(snapShotName);

                // ###
                admin.close();
                connection.close();
            } catch (Exception ex2) {
                LOG.error("[ScanSnapShotMR] - finally ex = {}", ex2);
                result = -1;
            }
        }

        // ###
        return result;
    }

    public void setConf(Configuration configuration) {
        // ### Hbase
        configuration.set(HBASE_NAME, HBASE_VALUE);

        // ###
        configuration.set(HDFS_FS_DEFAULTFS, HDFS_FS_DEFAULTFS_VALUE);
    }

    public Configuration getConf() {
        return null;
    }

    /**
     * [main] - Main
     * 1.1Hbase
     * 2.2
     *
     * @param args
     * @throws Exception
     */
    public static void main(final String[] args) throws Exception {

        // ###
        if (ArrayUtils.isEmpty(args)) {
            LOG.error("[ScanSnapShotMR] - args is not empty!");
            System.exit(-1);
        }

        // ###
        if (args.length < 2 || args.length > 2) {
            LOG.error("[ScanSnapShotMR] - args size must be 2 ; 1) Hbase Table Name 2) Hbase Table Output Dir");
        }

        // ###
        for (int i = 0; i < args.length; i++) {
            if (LOG.isInfoEnabled()) {
                LOG.info("[ScanSnapShotMR] - arg(" + i + ")=" + args[i]);
            }
        }

        // ###
        ToolRunner.run(HBaseConfiguration.create(), new ScanSnapShotMR(), args);
    }
}
```

2.3 配置说明

2.3.1 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

- 测试环境HDFS
    - private static final String HDFS_FS_DEFAULTFS_VALUE = "hdfs://al-bj-bigdata-inf-hbase-test01.inf.bdg.baijiahulian:8020/";
    - private static final String STAGE_DIR_PREFIX = "hdfs://al-bj-bigdata-inf-hbase-test01.inf.bdg.baijiahulian:8020/tmp/hbase/stagedir/";
    - private static final String HBASE_ROOT_DIR_VALUE = "hdfs://al-bj-bigdata-inf-hbase-test01.inf.bdg.baijiahulian:8020/hbase";
    - private static final String TEP_DIR_PREFIX = "hdfs://al-bj-bigdata-inf-hbase-test01.inf.bdg.baijiahulian:8020/tmp/hbase/region_tmp/";
    - private static final String STAGE_TRASH_DIR_PREFIX = "hdfs://al-bj-bigdata-inf-hbase-test01.inf.bdg.baijiahulian:8020/user/hbase_tmp/.Trash/";

2.3.2 线上环境

- 线上环境用于进行prod上线
- 具体的配置