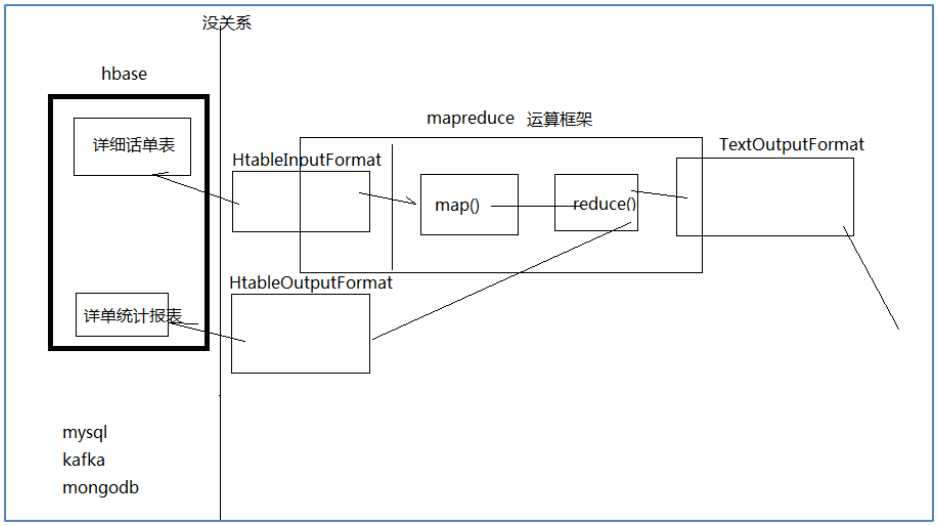
HBase 高级编程

目录

1、HBase 结合 MapReduce

为什么需要用 MapReduce 去访问 HBase 的数据？——加快分析速度和扩展分析能力
MapReduce 访问 HBase 数据作分析一定是在离线分析的场景下应用



1.1、HBaseToHDFS

从 HBase 中读取数据，分析之后然后写入 HDFS，代码实现：

```
package com.ghgj.mapreduce;
```

```
import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.client.Result;
import org.apache.hadoop.hbase.client.Scan;
import org.apache.hadoop.hbase.io.ImmutableBytesWritable;
import org.apache.hadoop.hbase.mapreduce.TableMapReduceUtil;
import org.apache.hadoop.hbase.mapreduce.TableMapper;
import org.apache.hadoop.hbase.util.Bytes;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class HbaseReader {

    public static String user_info = "user_info";

    static class HdfsSinkMapper extends TableMapper<Text, NullWritable> {

        @Override
        protected void map(ImmutableBytesWritable key, Result value,
                Context context) throws IOException, InterruptedException {
            byte[] bytes = key.copyBytes();
            String rowkey = new String(bytes);
            byte[] usernameBytes =
value.getValue("base_info".getBytes(),"name".getBytes());
            String username = Bytes.toString(usernameBytes);
            context.write(new Text(rowkey + "\t" + username),
                    NullWritable.get());
        }
    }

    static class HdfsSinkReducer extends
            Reducer<Text, NullWritable, Text, NullWritable> {
        @Override
        protected void reduce(Text key, Iterable<NullWritable> values,
                Context context) throws IOException, InterruptedException {
            context.write(key, NullWritable.get());
        }
    }
```

```
        public static void main(String[] args) throws Exception {
            Configuration conf = HBaseConfiguration.create();
            System.setProperty("HADOOP_USER_NAME", "root");
            conf.set("hbase.zookeeper.quorum",
"hadoop01:2181,hadoop02:2181,hadoop03:2181,hadoop04:2181,hadoop05:2181");
            Job job = Job.getInstance(conf);
            job.setJarByClass(HbaseReader.class);
            Scan scan = new Scan();
            TableMapReduceUtil.initTableMapperJob(user_info, scan,
                    HdfsSinkMapper.class, Text.class, NullWritable.class, job);
            job.setReducerClass(HdfsSinkReducer.class);
            FileOutputFormat.setOutputPath(job, new Path("/hbasetest/output"));
            job.setOutputKeyClass(Text.class);
            job.setOutputValueClass(NullWritable.class);
            job.waitForCompletion(true);
        }
}
```

1.2、HDFSToHBase

从 hdfs 从读入数据，处理之后写入 hbase，代码实现：

```
    package com.ghgj.mapreduce;

    import java.io.IOException;

    import org.apache.hadoop.conf.Configuration;
    import org.apache.hadoop.fs.Path;
    import org.apache.hadoop.hbase.HBaseConfiguration;
    import org.apache.hadoop.hbase.HColumnDescriptor;
    import org.apache.hadoop.hbase.HTableDescriptor;
    import org.apache.hadoop.hbase.TableName;
    import org.apache.hadoop.hbase.client.HBaseAdmin;
    import org.apache.hadoop.hbase.client.Mutation;
    import org.apache.hadoop.hbase.client.Put;
    import org.apache.hadoop.hbase.io.ImmutableBytesWritable;
    import org.apache.hadoop.hbase.mapreduce.TableMapReduceUtil;
    import org.apache.hadoop.hbase.mapreduce.TableReducer;
    import org.apache.hadoop.io.LongWritable;
    import org.apache.hadoop.io.NullWritable;
    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;

public class HbaseSinkerTest {

    public static String TABLE_NAME = "flowbean";

    static class HbaseSinkMrMapper extends Mapper<LongWritable, Text, Text, NullWritable> {
        @Override
        protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
            String line = value.toString();
            String[] fields = line.split("\t");
            String phone = fields[0];
            String url = fields[1];
            context.write(new Text(phone+"\t"+url), NullWritable.get());
        }
    }

    static class HbaseSinkMrReducer extends TableReducer<Text, NullWritable, ImmutableBytesWritable> {
        @Override
        protected void reduce(Text key, Iterable<NullWritable> values,
                Context context) throws IOException, InterruptedException {
    //        Put put = new Put(key.getPhone().getBytes());
            String[] keys = key.toString().split("\\t");
            Put put = new Put(keys[0].getBytes());
            put.add("f1".getBytes(), "url".getBytes(), keys[1].getBytes());
            context.write(new ImmutableBytesWritable(keys[0].getBytes()), put);
        }
    }

    public static void main(String[] args) throws Exception {
        Configuration conf = HBaseConfiguration.create();
        conf.set("hbase.zookeeper.quorum", "hadoop01:2181,hadoop02:2181,hadoop03:2181,hadoop04:2181,hadoop05:2181");

        // 以下这段代码为创建表的代码，表名 flowbean， 列簇叫 f1
        HBaseAdmin hBaseAdmin = new HBaseAdmin(conf);
        boolean tableExists = hBaseAdmin.tableExists(TABLE_NAME);
        if (tableExists) {
            hBaseAdmin.disableTable(TABLE_NAME);
            hBaseAdmin.deleteTable(TABLE_NAME);
```

```
                }
                HTableDescriptor desc = new HTableDescriptor(TableName.valueOf(TABLE_NAME));
                HColumnDescriptor hColumnDescriptor = new HColumnDescriptor("f1".getBytes());
                desc.addFamily(hColumnDescriptor);
                hBaseAdmin.createTable(desc);

                Job job = Job.getInstance(conf);
                job.setJarByClass(HbaseSinker.class);
                job.setMapperClass(HbaseSinkMrMapper.class);
                job.setReducerClass(HbaseSinkMrReducer.class);
                TableMapReduceUtil.initTableReducerJob(TABLE_NAME, HbaseSinkMrReducer.class, job);
                FileInputFormat.setInputPaths(job, new Path("/data/data.txt"));
                job.setMapOutputKeyClass(Text.class);
                job.setMapOutputValueClass(NullWritable.class);
                job.setOutputKeyClass(ImmutableBytesWritable.class);
                job.setOutputValueClass(Mutation.class);
                job.waitForCompletion(true);
            }
}
```

2、HBase 和 MySQL 进行数据互导

2.1、MySQL 数据导入到 HBase

下面是命令：

```
sqoop import --connect jdbc:mysql://hadoop01/mytest --username root --password root --table student --hbase-create-table --hbase-table studenttest --column-family name --hbase-row-key id
```

其中会报错，说 Exception in thread "main" java.lang.NoSuchMethodError: org.apache.hadoop.hbase.HTableDescriptor.addFamily(Lorg/apache/hadoop/hbase/HColumnDescriptor;)V 是由于版本不兼容引起，我们可以通过事先创建好表就可以使用了。

请使用下面的命令：

```
sqoop import --connect jdbc:mysql://hadoop01/mytest --username root --password root --table student --hbase-table studenttest1 --column-family name --hbase-row-key id
```

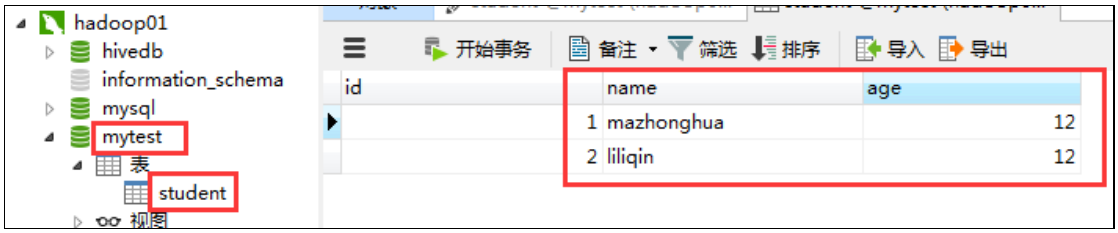
--hbase-create-table 自动在 hbase 中创建表

--column-family name 指定列簇名字

--hbase-row-key id 指定 rowkey 对应的 mysql 当中的键

看效果：

Mysql 数据：



导入的过程日志：

```
                Total time spent by all maps in occupied slots (ms)=30772
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=30772
                Total vcore-milliseconds taken by all map tasks=30772
                Total megabyte-milliseconds taken by all map tasks=31510528
        Map-Reduce Framework
                Map input records=2
                Map output records=2
                Input split bytes=197
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=478
                CPU time spent (ms)=4150
                Physical memory (bytes) snapshot=245022720
                Virtual memory (bytes) snapshot=1709907968
                Total committed heap usage (bytes)=37859328
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=0
16/12/19 00:37:40 INFO mapreduce.ImportJobBase: Transferred 0 bytes in 48.3893 seconds (0 bytes/sec)
16/12/19 00:37:40 INFO mapreduce.ImportJobBase: Retrieved 2 records.
[root@hadoop01 sqoop-1.4.6.bin]#
```

最后看 hbase 中的数据：

```
hbase(main):004:0> scan 'student'
ROW                             COLUMN+CELL
 1                              column=name:age, timestamp=1482136657425, value=12
 1                              column=name:name, timestamp=1482136657425, value=mazhonghua
 2                              column=name:age, timestamp=1482136655693, value=12
 2                              column=name:name, timestamp=1482136655693, value=liliqin
2 row(s) in 0.4150 seconds

hbase(main):005:0>
```

2.2、HBase 数据导入到 MySQL

目前没有直接的命令将 HBase 中的数据导出到 MySQL，但是可以先将 HBase 中的数据导出到 HDFS 中，再将数据导出 MySQL

替代方案：

先将 HBase 的数据导入到 HDFS 或者 Hive，然后再将数据导入到 MySQL

3、HBase 整合 Hive

3.1、原理

Hive 与 HBase 利用两者本身对外的 API 来实现整合，主要是靠 HBaseStorageHandler 进行通信，利用 HBaseStorageHandler，Hive 可以获取到 Hive 表对应的 HBase 表名，列簇以及列，InputFormat 和 OutputFormat 类，创建和删除 HBase 表等。

Hive 访问 HBase 中表数据，实质上是通过 MapReduce 读取 HBase 表数据，其实现是在 MR 中，使用 HiveHBaseTableInputFormat 完成对 HBase 表的切分，获取 RecordReader 对象来读取数据。

对 HBase 表的切分原则是一个 Region 切分成一个 Split，即表中有多少个 Regions，MapReduce 中就有多少个 Map。

读取 HBase 表数据都是通过构建 Scanner，对表进行全表扫描，如果有过滤条件，则转化为 Filter。当过滤条件为 RowKey 时，则转化为对 RowKey 的过滤，Scanner 通过 RPC 调用 RegionServer 的 next()来获取数据

3.2、准备 HBase 表和数据

创建 HBase 表：

```
create 'mingxing',{NAME => 'base_info',VERSIONS => 1},{NAME => 'extra_info',VERSIONS => 1}
```

插入准备数据：

```
put 'mingxing','rk001','base_info:name','huangbo'
put 'mingxing','rk001','base_info:age','33'
put 'mingxing','rk001','extra_info:math','44'
put 'mingxing','rk001','extra_info:province','beijing'
put 'mingxing','rk002','base_info:name','xuzheng'
put 'mingxing','rk002','base_info:age','44'
put 'mingxing','rk003','base_info:name','wangbaoqiang'
put 'mingxing','rk003','base_info:age','55'
put 'mingxing','rk003','base_info:gender','male'
put 'mingxing','rk004','extra_info:math','33'
put 'mingxing','rk004','extra_info:province','tianjin'
put 'mingxing','rk004','extra_info:children','3'
put 'mingxing','rk005','base_info:name','liutao'
put 'mingxing','rk006','extra_info:name','liujialing'
```

3.3、Hive 端操作

进入 Hive 客户端，需要进行一下参数设置：

```
指定 hbase 所使用的 zookeeper 集群的地址：默认端口是 2181，可以不写
set hbase.zookeeper.quorum=hadoop03:2181,hadoop04:2181,hadoop05:2181;
指定 hbase 在 zookeeper 中使用的根目录
set zookeeper.znode.parent=/hbase;
加入指定的处理 jar
add jar /home/hadoop/apps/apache-hive-2.3.2-bin/lib/hive-hbase-handler-2.3.2.jar;
```

创建基于 HBase 表的 hive 表：

所有列簇：

```
create external table mingxing(rowkey string, base_info map<string, string>, extra_info map<string, string>)
row format delimited fields terminated by '\t'
stored by 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
with serdeproperties ("hbase.columns.mapping" = ":key,base_info:,extra_info:")
tblproperties("hbase.table.name"="mingxing","hbase.mapred.output.outputtable"="mingxing");
```

部分列簇部分列：

```
create external table mingxing1(rowkey string, name string, province string)
row format delimited fields terminated by '\t'
stored by 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
with serdeproperties ("hbase.columns.mapping" = ":key,base_info:name,extra_info:province")
tblproperties("hbase.table.name"="mingxing","hbase.mapred.output.outputtable"="mingxing");
```

org.apache.hadoop.hive.hbase.HBaseStorageHandler：处理 hive 到 hbase 转换关系的处理器

hbase.columns.mapping：定义 hbase 的列簇和列到 hive 的映射关系

hbase.table.name：hbase 表名

3.4、验证

查询语句：

```
select * from mingxing;
select count(*) from mingxing;
select rowkey,base_info['name'] from mingxing;
select rowkey,extra_info['province'] from mingxing;
select rowkey,base_info['name'], extra_info['province'] from mingxing;
```