Spark SQL & Spark Hive编程开发， 并和Hive执行效率对比

Spark SQL也公布了很久，今天写了个程序来看下Spark SQL、Spark Hive以及直接用Hive执行的效率进行了对比。以上测试都是跑在YARN上。

首先我们来看看我的环境：

1. 3台DataNode，2台NameNode，每台机器20G内存，24核
2. 数据都是lzo格式的，共336个文件，338.6 G
3. 无其他任务执行

三个测试都是执行

```
select count(*), host, module
from ewaplog
group by host, module
order by host, module;
```

下面我们先来看看Spark SQL核心的代码（关于Spark SQL的详细介绍请参见Spark官方文档，这里我就不介绍了。）：

```
/**
 * User: 过往记忆
 * Date: 14-8-13
 * Time: 下午23:16
 * bolg:
 * 本文地址：/archives/1090
 * 过往记忆博客，专注于hadoop、hive、spark、shark、flume的技术博客，大量的干货
```

```
 * 过往记忆博客微信公共帐号：iteblog_hadoop
 */

public static class Entry implements Serializable {
        private String host;
        private String module;
        private String content;

        public Entry(String host, String module, String content) {
            this.host = host;
            this.module = module;
            this.content = content;
        }

        public String getHost() {
            return host;
        }

        public void setHost(String host) {
            this.host = host;
        }

        public String getModule() {
            return module;
        }

        public void setModule(String module) {
            this.module = module;
        }

        public String getContent() {
            return content;
        }

        public void setContent(String content) {
            this.content = content;
        }

        @Override
        public String toString() {
            return "[" + host + "₩t" + module + "₩t" + content + "]";
        }
}

.......
```

```
JavaSparkContext ctx = ...
JavaSQLContext sqlCtx = ...
JavaRDD<Entry> stringJavaRDD = ctx.textFile(args[0]).map(
    new Function<String, Entry>() {
        @Override
        public Entry call(String str) throws Exception {
           String[] split = str.split("₩u0001");
           if (split.length < 3) {
              return new Entry("", "", "");
           }

           return new Entry(split[0], split[1], split[2]);
        }
});

JavaSchemaRDD schemaPeople = sqlCtx.applySchema(stringJavaRDD, Entry.class);
schemaPeople.registerAsTable("entry");
JavaSchemaRDD teenagers = sqlCtx.sql("select count(*), host, module " +
           "from entry " +
           "group by host, module " +
           "order by host, module");

List<String> teenagerNames = teenagers.map(new Function<Row, String>() {
    public String call(Row row) {
        return row.getLong(0) + "₩t" +
             row.getString(1) + "₩t" + row.getString(2);
    }
}).collect();

for (String name : teenagerNames) {
        System.out.println(name);
}
```

Spark Hive核心代码：

```
/**
 * User: 过往记忆
 * Date: 14-8-23
 * Time: 下午23:16
 * bolg:
 * 本文地址：/archives/1090
 * 过往记忆博客，专注于hadoop、hive、spark、shark、flume的技术博客，大量的干货
 * 过往记忆博客微信公共帐号：iteblog_hadoop
```

```
 */
JavaHiveContext hiveContext =....;
JavaSchemaRDD result = hiveContext.hql("select count(*), host, module " +
            "from ewaplog " +
            "group by host, module " +
            "order by host, module");
List<Row> collect = result.collect();
for (Row row : collect) {
   System.out.println(row.get(0) + "₩t" + row.get(1) + "₩t" + row.get(2));
}
```

大家可以看到Spark Hive核心代码里面的SQL语句和直接在Hive上面执行一样，在执行这个代码的时候，需要确保ewaplog存在。而且在运行这个程序的时候需要依赖Hive的一些jar包，需要依赖Hive的元数据等信息。对Hive的依赖比较大。而Spark SQL直接读取lzo文件，并没有涉及到Hive，相比Spark Hive依赖性这方便很好。Spark SQL直接读取lzo文件，然后将数据存放在RDD中，applySchema方法将JavaRDD转换成JavaSchemaRDD，我们来看看文档是怎么来描述的

> At the core of this component is a new type of RDD, SchemaRDD. SchemaRDDs are composed Row objects along with a schema that describes the data types of each column in the row. A SchemaRDD is similar to a table in a traditional relational database. A SchemaRDD can be created from an existing RDD, Parquet file, a JSON dataset, or by running HiveQL against data stored in Apache Hive.

转换成JavaSchemaRDD之后，我们可以用用registerAsTable将它注册到表中，之后就可以通过JavaSQLContext的sql方法来执行相应的sql语句了。

用Maven编译完上面的程序之后，放到Hadoop集群上面运行：

```
iteblog@Spark $ spark-submit --master yarn-cluster
                --jars lib/spark-sql_2.10-1.0.0.jar
                --class SparkSQLTest
                --queue queue1
                ./spark-1.0-SNAPSHOT.jar
                /home/wyp/test/*.lzo
```

分别经过了20分钟左右的时间，Spark SQL和Spark Hive都可以运行完，结果如下：

```
39511517 bokingserver1 CN1_hbase_android_client
59141803 bokingserver1 CN1_hbase_iphone_client
```

39544052 bokingserver2 CN1_hbase_android_client
59156743 bokingserver2 CN1_hbase_iphone_client
23472413 bokingserver3 CN1_hbase_android_client
35251936 bokingserver3 CN1_hbase_iphone_client
23457708 bokingserver4 CN1_hbase_android_client
35262400 bokingserver4 CN1_hbase_iphone_client
19832715 bokingserver5 CN1_hbase_android_client
51003885 bokingserver5 CN1_hbase_iphone_client
19831076 bokingserver6 CN1_hbase_android_client
50997314 bokingserver6 CN1_hbase_iphone_client
30526207 bokingserver7 CN1_hbase_android_client
50702806 bokingserver7 CN1_hbase_iphone_client
54844214 bokingserver8 CN1_hbase_android_client
88062792 bokingserver8 CN1_hbase_iphone_client
54852596 bokingserver9 CN1_hbase_android_client
88043401 bokingserver9 CN1_hbase_iphone_client
54864322 bokingserver10 CN1_hbase_android_client
88041583 bokingserver10 CN1_hbase_iphone_client
54891529 bokingserver11 CN1_hbase_android_client
88007489 bokingserver11 CN1_hbase_iphone_client
54613917 bokingserver12 CN1_hbase_android_client
87623763 bokingserver12 CN1_hbase_iphone_client

为了比较基于Spark的任务确实比基于Mapreduce的快，我特意用Hive执行了同样的任务，如下：

```
hive> select count(*), host, module from ewaplog
    > group by host, module order by host, module;

Job 0: Map: 2845  Reduce: 364   Cumulative CPU: 17144.59 sec
HDFS Read: 363542156311 HDFS Write: 36516 SUCCESS
Job 1: Map: 1  Reduce: 1   Cumulative CPU: 4.82 sec
HDFS Read: 114193 HDFS Write: 1260 SUCCESS
Total MapReduce CPU Time Spent: 0 days 4 hours 45 minutes 49 seconds 410 msec
OK
39511517 bokingserver1 CN1_hbase_android_client
59141803 bokingserver1 CN1_hbase_iphone_client
39544052 bokingserver2 CN1_hbase_android_client
59156743 bokingserver2 CN1_hbase_iphone_client
23472413 bokingserver3 CN1_hbase_android_client
35251936 bokingserver3 CN1_hbase_iphone_client
23457708 bokingserver4 CN1_hbase_android_client
35262400 bokingserver4 CN1_hbase_iphone_client
```

```
19832715 bokingserver5 CN1_hbase_android_client
51003885 bokingserver5 CN1_hbase_iphone_client
19831076 bokingserver6 CN1_hbase_android_client
50997314 bokingserver6 CN1_hbase_iphone_client
30526207 bokingserver7 CN1_hbase_android_client
50702806 bokingserver7 CN1_hbase_iphone_client
54844214 bokingserver8 CN1_hbase_android_client
88062792 bokingserver8 CN1_hbase_iphone_client
54852596 bokingserver9 CN1_hbase_android_client
88043401 bokingserver9 CN1_hbase_iphone_client
54864322 bokingserver10 CN1_hbase_android_client
88041583 bokingserver10 CN1_hbase_iphone_client
54891529 bokingserver11 CN1_hbase_android_client
88007489 bokingserver11 CN1_hbase_iphone_client
54613917 bokingserver12 CN1_hbase_android_client
87623763 bokingserver12 CN1_hbase_iphone_client
Time taken: 1818.706 seconds, Fetched: 24 row(s)
```

从上面的显示我们可以看出，Hive执行同样的任务用了30分钟，而Spark用了20分钟，也就是省了1/3的时间，还是很快的。在运行的过程中，我发现Spark消耗内存比较大，在程序运行期间，三个子节点负载很高，整个队列的资源消耗了一半以上。我想如果集群的机器数据更多的话，Spark的运行速度应该还会有一些提升。好了今天就说到这，欢迎关注本博客。