链滴

使用 spark 从 kafka 消费数据写入 hive 动态分区表（一）

作者：ludengke95

原文链接：https://ld246.com/article/1557503765084

来源网站：链滴

使用spark从kafka消费数据写入hive动态分区表

最近因为业务需求，需要把kafka内的数据写入hive动态分区表，进入kafka的数据是保证不会重复的同一条业务数据仅会进入kafka一次。这就保证数据到了hive基本不会发生update操作，可以对hive行统计生成静态表的形式将统计数据写入mysql。咱也不说那么多废话了，开整。

直接写入

从kafka取出数据转化成bean对象，根据业务要求将数据过滤，清洗，拿到最终的RDD，直接写入hiv分区表。（PS：我就不贴全部代码，仅仅贴出主要代码，作为一个有职业道德的程序员，职业道德不许我这么做。哈哈哈，其实是我怂，我这么大条，万一泄露公司机密就OVER了）

RDD写入HIVE动态分区表，因为我需要向两个动态分区表写数据，所以加了type字段

```
private static void writeHive(SparkSession sparkSession,JavaRDD rdd,String type){

        Dataset<Row> writerTradeData = sparkSession.createDataFrame(rdd,type.equals("trade"
?ApiTradePlus.class:ApiTradeOrderPlus.class);
        writerTradeData.createOrReplaceTempView("tmp"+type);
        String sql = "";
        if(type.equals("trade")){
            sql = "insert into tmp_trade `partition(tradedate)` select * from "+"tmp"+type;
        }
        else if(type.equals("order")){
            sql = "insert into tmp_trade_order `partition(tradedate)` select * from "+"tmp"+type;
        }
        sparkSession.sql(sql);
    }
```

别忘记最重要的hive操作，默认是不支持动态分区表需要开启配置。以hive.exec.开头的几个参数，两个开启动态分区表的配置，后三个是设置分区数上限，我是按照日期分的，一不留神就容易超出分

表数上限，所以设置的稍微大了一点点。

```
SparkConf sparkConf = SparkFactory.getDefaultSparkConf()
                .set("spark.executor.cores", "4")
                .set("spark.ui.port", "30004")
                .set("hive.exec.dynamic.partition.mode","nonstrict")
                .set("hive.exec.dynamic.partition","true")
                .set("hive.exec.max.dynamic.partitions.pernode","100000")
                .set("hive.exec.max.dynamic.partitions","100000")
                .set("hive.exec.max.created.files","100000")
                .setAppName("KafkaToHiveStreaming");
```

哦，对了还有这个我写的sparkConf工厂类，生成默认配置，速率设置了100，因为有15个分区，10执行一次，所以执行一次就会消费1.5W的消息，自我觉得应该能够10s应该够了。

```
public static SparkConf getDefaultSparkConf() {

        return new SparkConf()
                .set("spark.executor.memory", "6g")
                .set("spark.shuffle.file.buffer", "1024k")
                .set("spark.reducer.maxSizeInFlight", "128m")
                .set("spark.shuffle.memoryFraction", "0.3")
                .set("spark.streaming.stopGracefullyOnShutdown", "true")
                .set("spark.streaming.kafka.maxRatePerPartition", "100")
                .set("spark.serializer", KryoSerializer.class.getCanonicalName())
                .registerKryoClasses(SERIALIZER_CLASS);
    }
```

接下来激动人心的时刻到了。把任务提交到spark集群，当我满心欢喜提上去之后，打开监控页面，瞬间就凉了一半，提交的第一个任务，执行了整整10分钟，我已经不忍心截图了，悲伤辣么大。 old_sweat

10分钟处理了1.5Wkafka消息，写入数据库2W左右，因为业务需求,需要特殊记录写两次，位于的分不一样。

曲线救国

先写入hive非分区表，然后再通过hive将非分区表的数据迁移到分区表上。

直接写分区表的速度真的是慢的可以，如果放到线上，我估计就要被离职了。为了我Money还是要加呀。所以想出了这个曲线救国的路线，其实刚开始是想写hadoop文件，然后使用外部表，外部表数源选择hadoop文件，然后通过insert into select * from导入。转念一想不如直接写非分区表，这样直接生成格式化号的hadoop文件，还不用外部表。

所以原来的一个spark任务分成了两个spark任务：

1. SparkStreaming任务：负责消费kafka数据写入非分区表。
2. SparkSql任务：负责将hive的非分区表的数据迁移到动态分区表。

SparkStreaming关键代码如下：

```
private static void writeHive(SparkSession sparkSession,JavaRDD rdd,String time,String type){

        Dataset<Row> writerTradeData = sparkSession.createDataFrame(rdd,type.equals(TABLE
API_TRADE)?ApiTradeWithoutOrder.class:ApiTradeOrderWithTime.class);
```

```
        writerTradeData.createOrReplaceTempView("tmp"+time);
        String sql = "";
        String selectSql = null;
        /**
         * select 的顺序要求与hive中desc表的字段顺序一致
         */
        if(type.equals(TABLE_API_TRADE)){
            selectSql = " select 全部字段(最好不要用*) from tmp"+time;
            sql = String.join(" ","insert into ",TABLE_API_TRADE) ;
        }
        else if(type.equals(TABLE_API_TRADE_ORDER)) {
            selectSql=" select 全部字段(最好不要用*) from tmp" + time;
            sql = String.join(" ","insert into ",TABLE_API_TRADE_ORDER) ;
        }
        sparkSession.sql(String.join(" ",sql,selectSql));
    }
```

简单说明一下，这个地方time是干啥的，其实这个对应分区表的分区字段，还对代码进行了简单的优，毕竟公司最近在实行阿里编码规范。这个写非分区表是真的快，嗖嗖嗖的。1.5W/7s，这个7s还包了清洗过滤数据的过程，基本没变。

SparkSql的关键代码：

```
private static void deal() {
        String insertSql = " insert into ";
        String partitionSql = " partition(tradedate) ";
        String tradeSelectSql = " select 全部字段(最好不要用*) from " + TABLE_API_TRADE;
        String orderSelectSql = " select 全部字段(最好不要用*) from " + TABLE_API_TRADE_ORDER

        /**
         * 1. 迁移数据 insert into select from
         * 2. 重建非分区表 truncate table
         * point: select 的顺序要求与hive中desc表的字段顺序一致tradeSelectSql,orderSelectSql
         */
        session.sql("USE ".concat(HIVE_DB));
        session.sql(String.join(" ",insertSql, API_TRADE_PARTITIONED_TABLE_NAME,partitionSql,t
adeSelectSql));
        session.sql(String.join(" ",insertSql, API_TRADE_ORDER_PARTITIONED_TABLE_NAME,partit
onSql,orderSelectSql));
        session.sql("truncate table" + TABLE_API_TRADE);
        session.sql("truncate table" + TABLE_API_TRADE_ORDER);
    }
```

说白了，这个就是使用insert into partition(分区字段) select * from

hadoop崩了

当我满心欢喜的执行SparkSql任务的时候，又有噩耗发生了，hadoop竟然扛不住hive表数据迁移，非分区表写入到分区表，数据总量大约2000W左右吧，分了1.8W个task，在执行到0.8W的时候，hadoop第一个NameNode崩了，1.4W的时候第二个NameNode崩了，我感觉我也崩了，哎，花了10钟重启全部集群，还是规划好每次写入数据的条数吧，最后调整到了600W进行一次hive数据迁移。

hive就会搞事情

你以为迁移完成就完了嘛？NO！NO！NO！我不得统计hive数据的条数呀，很不幸的是hive有个配，默认不不开启，所以你的select count(1) from table 和上帝一样给你开了个玩笑，返回0，我忙活这么久你告诉我一条都没写进去。感觉自己和蔡徐坤一样，会唱，会跳，会rap，会打篮球，就是不写程序。乖乖的在hive-env.sh中加了

```
# hive shell sql 爆内存溢出可以增大这个值
export HADOOP_HEAPSIZE=4096
```

重点来了（敲黑板，记到小本本上）

重点来了，因为数据中可能存在\t\r\n等鬼东西，所以做了替换，并且在hive中的字段分割符使用\00，我就不信了用户数据的东西还有\001。最恶心的是\r\n会影响hive的数据行数，是不是听上去有点逼？来我告诉你，hive默认的行分割符是\n，所以如果你的字段里面包含\n，那么恭喜你中奖了，数会出现裂变（不知道在这里合适不合适）一条变两条，贼开心，厉害点的会变成10条，我就出现了1变0的操作。cold_sweat，是谁录入的，让我出来打一顿，缺少社会主义的毒打。

其实吧，这个事要解决很简单，我使用\011或者其他的作为行分割符不就行了嘛，但是很不幸的告诉，hive暂时仅支持\n作为行分割符，难受的一批，乖乖的清洗数据去吧。

标题：使用spark从kafka消费数据写入hive动态分区表（二）

作者：ludengke95

地址：http://www.ludengke95.info/articles/2019/05/20/1558346695894.html