光环毕业设计-新闻网站日志分析系统

**序: 网站流量统计基础指标**

1. 页面浏览量(pv)

是网站页面被点击的次数，如果同一个用户点击了10次，会生成10条日志，pv的值就是10。

1. 访问者数(uv)

是统计日志中浏览器cookie的个数，如果一个用户使用chrome访问了3次，又用IE访问了3次，访问者数就是2次，因为不同浏览器之间的cookie是不共享的。

1. 访问Ip数(ip)

跟访问者数类似，是对访客的ip数进行去重，同1个ip访问3次记做1次。

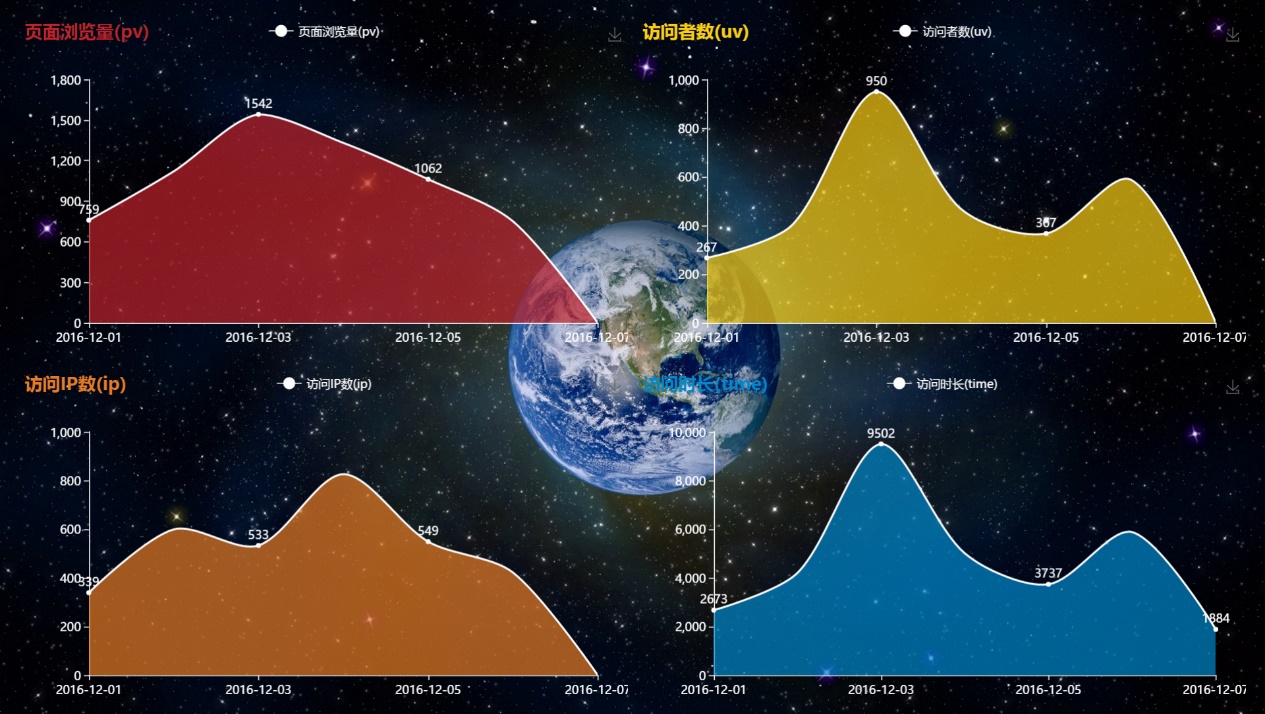
1. 访问时长(time)

是指访客的停留时间，同一cookie的会话按30分钟切分，最后一页按10秒计算。

毕业设计首页

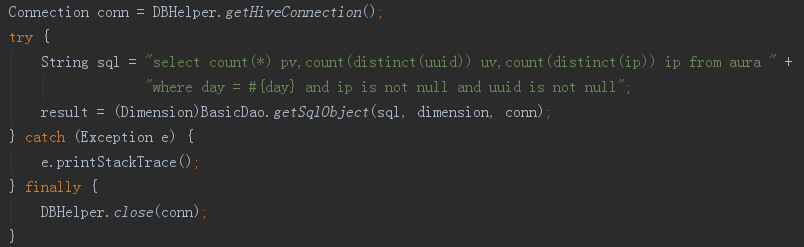


1. **离线计算框(Hive,MapReduce)**

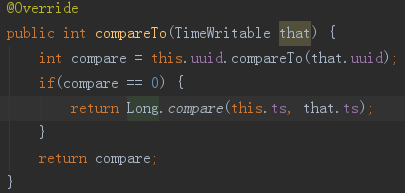


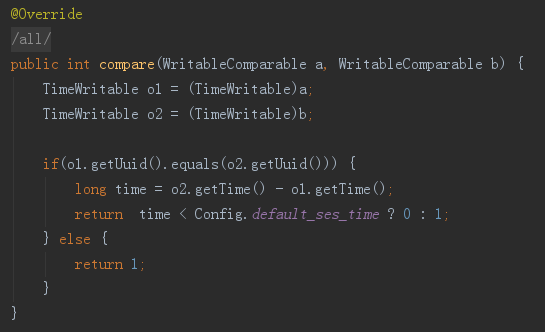
刚开班的时候咱们讲的是Hadoop体系的Hive和MapReduce。唐刚老师说过：“能用Hive的时候尽量使用Hive因为使用sql语句写程序比较简单，如果要计算的内容比较复杂，sql语句表达不出来，那就使用MapReduce，所以Hive和Mapreduce是要结合使用的”。在这里前3个指标pv，uv，ip是可以用Hive实现的，最后一个访问时长用sql语句表达不出来，所以用MapReduce来实现。

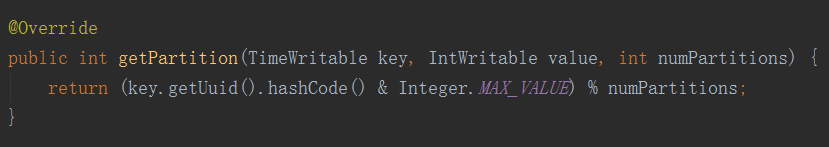
Hive代码片段 com.aura.java.hive.HiveAnalysis



MapReduce代码片段 com.aura.java.mapreduce.\*







1. **流式计算框架 (Storm,Spark Streaming)**

1. Storm



在Hive和MapReduce之后，咱们又学了Storm实时计算。相比而言，MapReduce是处理咱们的历史数据，而Storm是处理咱们实时产生的数据，一个MapReduce任务是有开始和结束的，而Storm任务开始运行后，如果不手工杀掉，是会一直运行下去的。

在这里Storm是实时读取Kafka里的数据，用的是storm.kafka.KafkaSpout,这个老师文档里有提到，而Kafka是和Flume结合使用，用的是org.apache.flume.sink.kafka.KafkaSink这个老师文档里也有。现在界面上没有数据，需要启动Storm程序，然后把日志传给Flume,使用命令:

flume-ng avro-client --conf conf -H ha01 -p 41414 -F /logs/aura.log

现在咱们就能看到访客国家分布的情况，而且数据每秒在实时的变化。而国家分布式通过ip库得到的，输入ip返回对应的国家和省份。

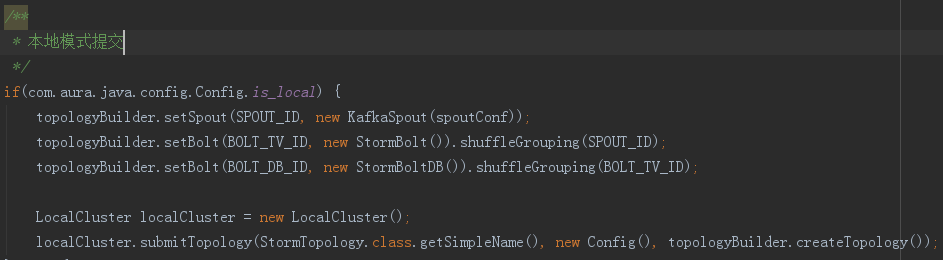
1. Spark Streaming

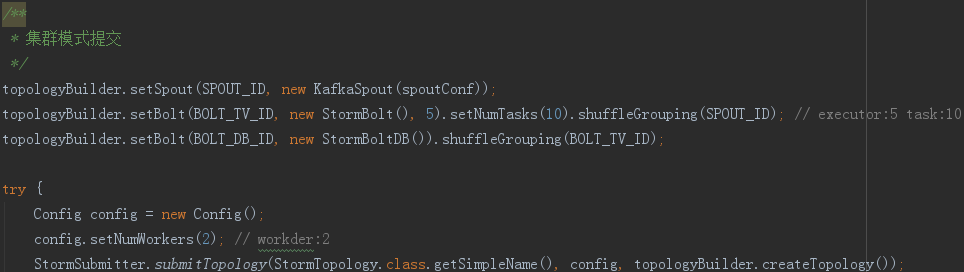


除了Storm之外咱们还学习了另一个实时性比较强的框架叫Spark Streaming,相比而言Strom是真正的流式计算，而Spark Streaming是小批量实时计算，也就是说Spark Streaming也是批处理，只不过他切分的时间很短，计算速度足够快。

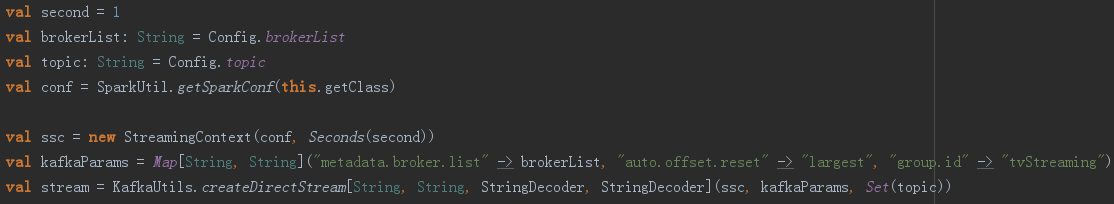
如每秒需要处理5000条数据，Strom从一秒开始到这一秒结束之间的过程有可能已经算完4000多条了，而Spark Streaming是等前一秒结束才计算这一秒的数据，所以从实时性上Storm要略强于Spark Streaming 而Spark Streaming的安装比较简单，公司做数据分析几乎都会使用Spark,只要安装了Spark 就可以直接使用Spark Streaming不需要再安装其他组件，另外Spark Streaming可以使用scala语言，实现相同的功能代码量要少于Strom的Java语言。

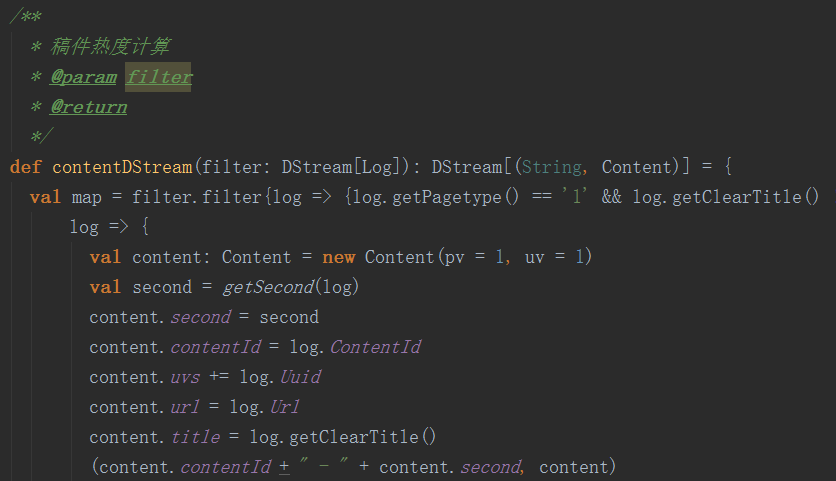
Storm代码片段 com.aura.java.storm.\*



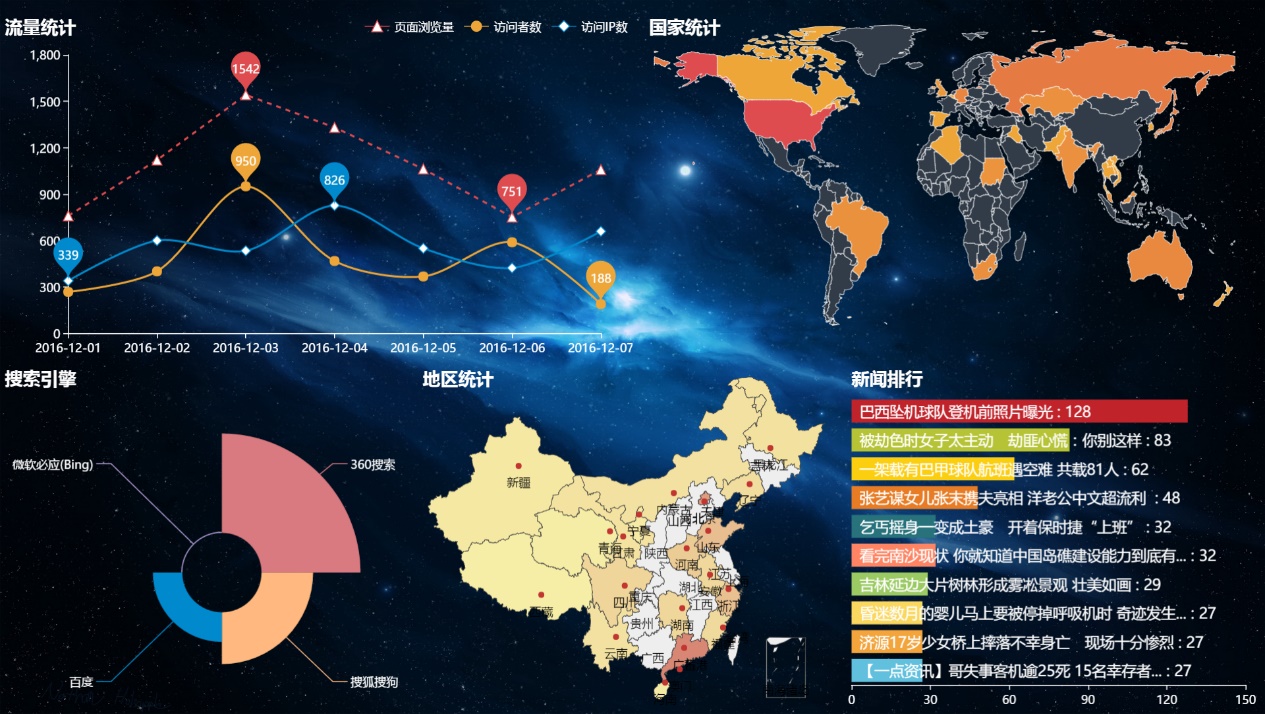


Spark Streaming代码片段 com.aura.scala.spark.streaming.\*





**三. 内存计算框架(Spark Core)**

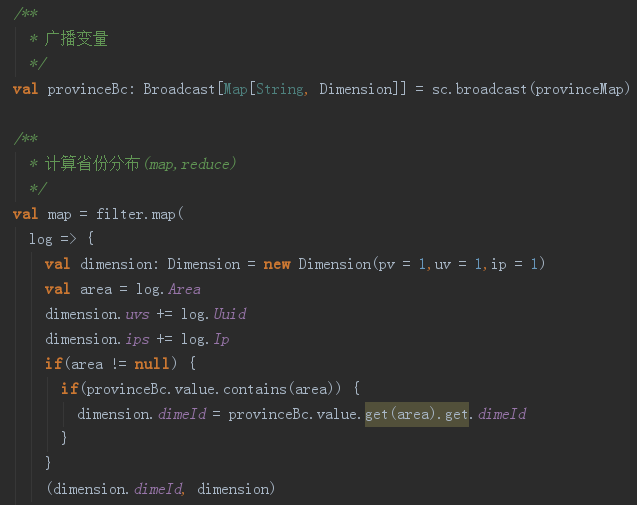


在Storm之后，咱们又学习了Spark Core内存计算框架，Spark Core也是一个批处理计算，只不过他的计算速度比MapReduce快，而这里的流量统计，就是刚才咱们提到的，pv，uv，ip的统计，国家分布和地区分布式通过ip库来获取，新闻排行是计算受访最高的10条新闻稿件，搜索引擎是通过，refer来获取的比如我通过百度来访问咱们学校的网站，点击进入光环国际首页，从这个页面上document.referrer可以得到上一个跳转的页面（百度搜索的页面）

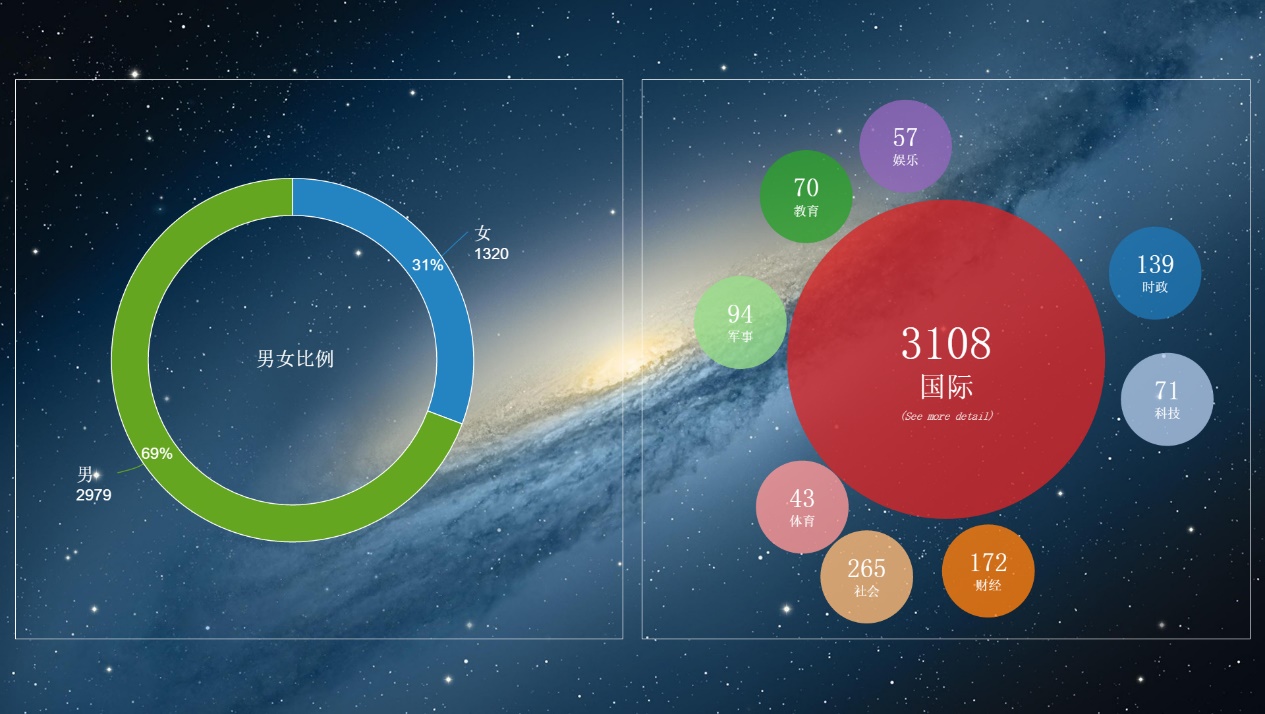


如上图所示，判断refer的特征域名中包含baidu.com并且?后的内容包括wd=就认为是通过百度搜索引擎跳转过来的，其余搜索引擎同理。

Spark Core代码片段 com.aura.scala.spark.core.\*



1. **机器学习(Spark MlLib)**



刚才咱们提到的情况都是，根据日志里面有的数据进行统计和分析，比如流量统计，搜索引擎统计，国家地区统计等，而现在要统计的内容是日志里没有的，比如男女比例和新闻频道的分类，去网站看新闻不需要登录，所以没有注册信息就得不到性别。而新闻所属频道虽然可以通过url里获取到。但是其中包含非该频道的稿件以及花边新闻内容太多，不能正确获取该新闻的频道分类情况。

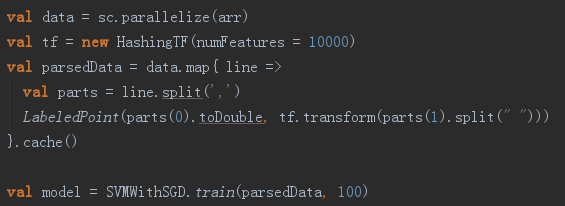
统计这部分数据可以用Spark MlLib中机器学习的分类算法来实现，性别分类用的是支持向量机算法，他是一个有监督学习，首先通过训练集生成模型，然后对日志数据进行分析，在分析过程中用模型来预测性别。

之前演示的支持向量机算法是一个二分类算法，而新闻频道的算法涉及到多个分类，这里用到了Spark MlLib的另一个算法叫贝叶斯分类算法，这个算法也是一个有监督学习也是生成模型后对日志数据进行频道分类预测。

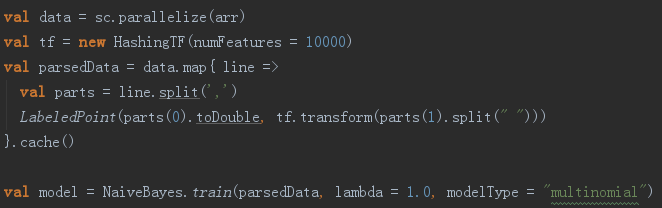
在Spark MlLib中有分类算法有很多中，比如我们在做频道分类预测的时候，一开始使用了随机森林算法，我们将训练集的一部分作为测试集来查看预测结果的准确率，后来发现朴素贝叶斯在预测这类数据准确率比随机森林决策树的准确率高，而二分类算法中也有逻辑回归和支持向量机二种算法可以供选择，一般情况下就是看哪个算法的准确率高就采用哪钟算法来实现。

**Spark MlLib代码片段 com.aura.scala.spark.mllib.\***

1. 支持向量积算法



1. 贝叶斯分类算法



1. **交互式查询(Impala,Kudu)**



1. Impala + HDFS Parquet

董老师讲过Impala和Presto是交互式查询，唐刚老师也提过Impala的实时性很高，相比而言Storm的实时性指的是，当前查看的数据是几秒之前产生的，而Impala的实时性是指查询数据量大的时候所需要的时间短。

之前咱们演示的MapReduce,Hive,Spark,Storm都是有一个分析程序，将HDFS或者Kafka里的日志数据进行分析，然后将分析结果写入MySql。而Impala是在Web程序中通过JDBC直接连接Impala查询HDFS里的Parquet格式的日志数据。因为他查询速度非常快，查询TB级的数据，只需要秒级的响应时间。

这里演示的是一个字符云，是指用户在搜索引擎是输入哪些关键词跳转到该网站的情况，简称新闻热搜词。

Impala的交互式查询有2点好处

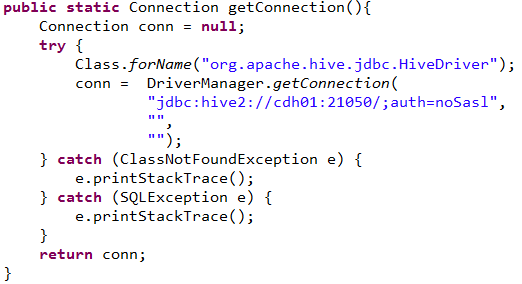
1 ) 开发比较简单，不需要写分析程序，直接通过Web页面查询。

2 ) 跨时段查询uv，ip等需要归并的数据结果更准确，比如用Spark来做日志分析，网站第一天的访问者数是2000，第二天是3000，如果跨时段查2天的数据，只能是2000+3000=5000，这个数据是不准确的，因为有一些用户是两天都访问过该网站的，只凭2000和3000这2个数字是不能把这类访客去重的，而Impala是每次查询都直接对日志里的cookie和ip字段进行count(distinct(...))是可以将这类用户去重的。

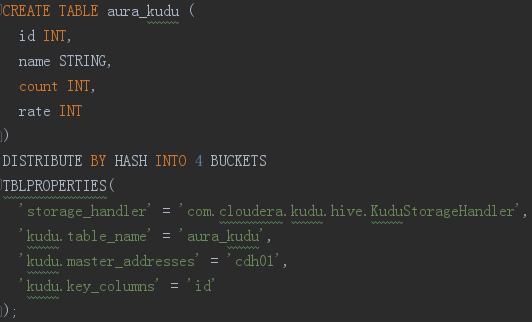
1. Impala + Kudu

之前咱们演示的都是用户日志数据，在企业里除了日志数据常见的还有爬虫抓取的数据，对指定的一些网站每小时进行抓取，如果遇到重复的Url就更新文章的标题和摘要，这类数据存在HDFS里是不适合的，因为HDFS不支持对文件内容进行修改，如果用Imapla进行修改操作，需要用到董老师上次课提到的新型键值数据库Kudu。Impala+Kudu的使用也很简单，只要建表的时候将storage\_handler指定为com.cloudera.kudu.hive.KuduStorageHandler就可以在Impala里使用update语句对数据进行修改了。

Impala使用JDBC连接



Kudu建表语句



**同学提问：为什么Impala的查询速度比Hive快这么多**

答: 原因有以下五点

1. Java前端与C++处理后端，效率要高于单独使用JAVA。

1. 基于内存进行计算，速度上优于Hive中使用MapReduce使用磁盘来计算。
2. 支持Parquet列式存储（这一点Hive近期也支持了）
3. 直接使用C++访问HDFS里的文件，绕过了原本的JAVA接口。
4. 使用Code Generation（代码生成）对CPU指令进行优化。

源码分享

<https://github.com/anlei-cdh/AuraAnalysis>

<https://github.com/anlei-cdh/AuraWeb>

光环国际-大数据实战1607班 安雷