# Spark实验部分

## Spark Streaming实验

【实验目的】

本实验旨在通过编写和执行基于Spark Streaming编程模型的wordcount程序，帮助学生深入理解Spark Streaming的工作原理，并学会使用Spark框架进行大规模数据处理。通过此实验，学生将能够掌握Spark编程的基本概念、编写简单的Spark程序以及运行它们在分布式环境中。

【实验内容】

基于Spark Streaming编程模型实现wordcount程序。编写Spark Streaming程序的基本步骤是：

1.创建SparkSession实例；

2.创建DataFrame表示从数据源输入的每一行数据；

3.DataFrame转换，类似于RDD转换操作；

4.创建StreamingQuery开启流查询；

5.调用StreamingQuery.awaitTermination()方法，等待流查询结束。

【实验环境】

1. 操作系统：Linux（Ubuntu）
2. 软件环境：Java JDK 1.8、Spark3.4
3. 硬件要求：至少1台计算机或虚拟机，建议配置至少4GB内存和100GB的硬盘空间用于安装Hadoop。
4. 网络连接：互联网连接，用于下载所需的软件和文档。

【实验步骤】

把wordcount程序写成单独一个scala文件，然后提交给spark执行；

1. 首先打开一个终端，输入以下命令:

cd /usr/local/spark/mycode

mkdir streaming

cd streaming

mkdir -p src/main/scala

cd src/main/scala

vim TestStructuredStreaming.scala

1. 用vim打开一个scala文件，在该文件输入以下内容:

import org.apache.spark.sql.functions.\_

import org.apache.spark.sql.SparkSession

object WordCountStructuredStreaming{

def main(args: Array[String]){

val spark = SparkSession.builder.appName("StructuredNetworkWordCount"). getOrCreate()

import spark.implicits.\_

val lines = spark.readStream.format("socket").option("host","localhost"). option("port",9999).load()

val words = lines.as[String].flatMap(\_.split(" "))

val wordCounts = words.groupBy("value").count()

val query = wordCounts.writeStream.outputMode("complete"). format("console").start()

query.awaitTermination()

}

}

1. 代码写好之后，退出终端，然后在/usr/lcoal/spark/mycode/streaming目录下创建simple.sbt文件:

cd /usr/local/spark/mycode/streaming

vim simple.sbt

打开vim编辑器以后，输入以下内容:

name := "Simple Project"

version := "1.0"

scalaVersion := "2.11.8"

libraryDependencies += "org.apache.spark" %% "spark-sql" % "2.0.0"

1. 通过执行sbt打包编译:

cd /usr/local/spark/mycode/streaming

/usr/local/sbt/sbt package

打包成功以后，就可以输入以下命令启动这个程序:

cd /usr/local/spark/mycode/streaming

/usr/local/spark/bin/spark-submit --class "WordCountStructuredStreaming" ./target/scala-2.11/simple-project\_2.11-1.0.jar

执行上输出程序之后，就开启了监听状态，当我们在终端输入“hello world，hello Beijing, hello world”之后，spark应用终端即可输出”hello”和”world”单词出现的次数。