# Spark实验部分

## 实验一 Spark安装与配置实验

【实验目的】

通过本实验，学习如何在Linux环境下配置大数据分析引擎Spark的安装与使用，为后续的Spark大数据处理和分析工作做好准备。

【实验内容】

1. 安装Linux系统（建议通过虚拟机）。
2. 在Linux系统配置Hadoop基础环境。
3. 安装Java环境
4. 安装Spark (选择和Hadoop集成的安装包)

【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Spark。
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】

1. 用浏览器访问spark.apache.org，打开documentation下面latest release页面，阅读Spark概述。下载[spark-3.4.2-bin-hadoop3.tgz](https://www.apache.org/dyn/closer.lua/spark/spark-3.4.2/spark-3.4.2-bin-hadoop3.tgz)
2. tar解压到home目录某文件下: tar -xzvf spark-3.4.2-bin-hadoop3.tgz
3. 参考文档说明，选择单机模式(standalone deplogy mode)，spark也可以单独运行。
4. bin目录下./spark-shell运行。

若分开安装Hadoop和Spark，请参考https://dblab.xmu.edu.cn/blog/804/

# Spark实验部分

## 实验二 WordCount实验

【实验目的】

本实验旨在通过编写和执行基于Spark编程模型的WordCount程序，帮助学生深入理解Spark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】

1. Scala程序实现WordCount，掌握RDD 和函数式编程思想。
2. 使用命令行执行WordCount程序。
3. 使用Eclipse编译、打包WordCount程序。
4. 查看程序执行结果。

【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。

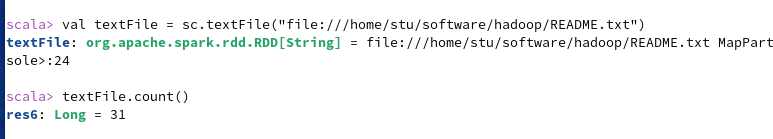
【实验步骤】

1. 使用命令行执行scale代码，注意双引号格式，在spark-shell中读取Linux系统本地文件**file:///home/stu/software/hadoop/README.txt, 统计**“Hadoop”单词的个数。

**scala> val sc = new SparkContext( )**

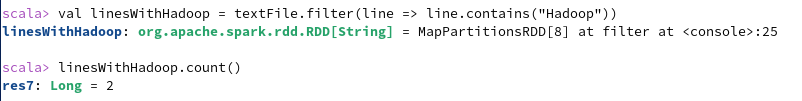
**scala> val textFile = sc.textFile(“file:///home/stu/software/hadoop/README.txt”)**

**scala> textFile.count()**

****

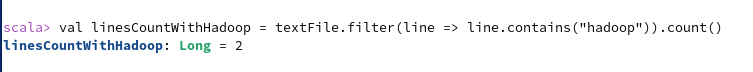
**scala> val linesWithHadoop = textFile.filter(line => line.contains(“Hadoop”))**

**scala> linesWithHadoop.count()**



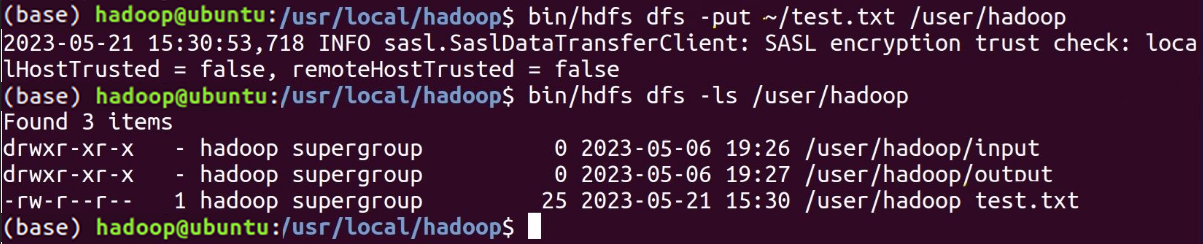
**上面两条语句可以通过链式操作合并为：**

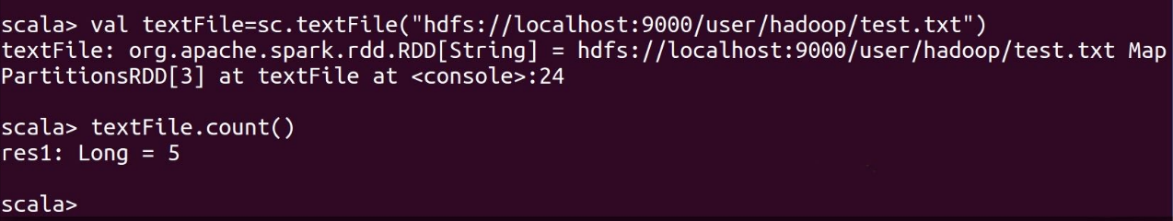
**scala> val linesCountWithHadoop = textFile.filter(line => line.contains(“Hadoop”)).count()**



1. 在spark-shell中读取HDFS系统文件“/user/hadoop/test.txt”（如果该文件不存在，请先创建），然后，统计出文件的行数；

**scala>** val lines = sc.textFile("hdfs://localhost:9000/user/hadoop/test.txt")





1. 使用命令行来实现wordcount，即统计文件中各个单词的计数：

**scala> val sc = new SparkContext( )**

**scala> val textFile = sc.textFile(“file:///home/stu/software/hadoop/README.txt”)**

**scala> val wordCounts = textFile.flatmap(line => line.split(“ ”)).map(word => (word,1)).reduceByKey((a,b) => a+b)**

**scala> wordCount.collect()**

**scala> wordCount.foreach(println)**

1. 用scale语言实现wordcount

WordCount程序参考代码1：

import java.io.File

import scala.io.Source

object WordCountApp {  
  
 def main(args: Array[String]): Unit = {  
 //文件路径  
 val filePath = "file:///home/stu/software/hadoop/README.txt "  
 val codec = "utf-8"  
 //打开文件  
 val file = Source.fromFile(filePath, codec)   
 val wc = file.getLines().flatMap(\_.split("\t")).toList.map((\_, 1)).groupBy((\_.\_1)).mapValues(\_.size)

println(wc)  
 // 关闭文件  
 file.close()  
 }  
}

WordCount程序参考代码2：

import org.apache.spark.rdd.RDD

import org.apache.spark.{SparkConf, SparkContext}

/\*\* \* Scala原生实现wordcount \*/

object WordCount {

def main(args: Array[String]): Unit = {

val list = List("cw is cool", "wc is beautiful", "andy is beautiful", "mike is cool")

/\*\* \* 第一步，将list中的元素按照分隔符这里是空格拆分，然后展开 \* 先map(\_.split(" "))将每一个元素按照空格拆分 \* 然后flatten展开 \* flatmap即为上面两个步骤的整合 \*/

val res0 = list.map(\_.split(" ")).flatten

val res1 = list.flatMap(\_.split(" "))

println("第一步结果")

println(res0)

println(res1)

/\*\* \* 第二步是将拆分后得到的每个单词生成一个元组 \* k是单词名称，v任意字符即可这里是1 \*/

val res3 = res1.map((\_, 1))

println("第二步结果")

println(res3)

/\*\* \* 第三步是根据相同的key合并 \*/

val res4 = res3.groupBy(\_.\_1)

println("第三步结果")

println(res4)

/\*\* \* 最后一步是求出groupBy后的每个key对应的value的size大小，即单词出现的个数 \*/

val res5 = res4.mapValues(\_.size)

println("最后一步结果")

println(res5.toBuffer)

}

}

1. 试一试，是否还有其他的实现方式？

# Spark实验部分

## 实验三 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。