**在Windows下创建Spark应用程序**

1. **下载和配置Apache Spark**

* 下载地址：<http://spark.apache.org/downloads.html>

下载后解压至C:\，并将Spark目录更名为C:\spark

* 下载<http://public-repo-1.hortonworks.com/hdp-win-alpha/winutils.exe>

下载后移至C:\hadoop-2.7.3\bin

* 设置环境变量如下：

set HADOOP\_HOME = C:\hadoop-2.7.3

set SPARK\_HOME = C:\spark

set PATH = %PATH%;%SPARK\_HOME%\bin

**二、下载、安装和配置Maven/sbt（Java/Scala构建工具）**

* Maven下载地址：<https://maven.apache.org/download.cgi>

可选择Binary zip archive下载并解压至C:\apache-maven-3.3.9

* 设置环境变量如下：

set MAVEN\_HOME = C:\apache-maven-3.3.9

set PATH = %PATH%;%MAVEN\_HOME%\bin

* sbt下载地址：<http://www.scala-sbt.org/download.html>

下载后安装至C:\Program Files (x86)\sbt

* 设置环境变量如下：

set SBT\_HOME = C:\Program Files (x86)\sbt

set PATH = %PATH%;%SBT\_HOME%\bin

**三、用Scala创建App：SparkPi**

* 搭建App架构：

在工作目录下创建SparkPi子目录

在SparkPi下创建src子目录

在SparkPi/src下创建main子目录

在SparkPi/src/main下创建scala子目录

* 在SparkPi/src/main/scala目录下新建SparkPi.scala文件，内容如下：

/\*\* Import the spark and math packages \*/

import scala.math.random

import org.apache.spark.\_

/\*\* Computes an approximation to pi \*/

object SparkPi {

def main(args: Array[String]) {

/\*\* Create the SparkConf object \*/

val conf = new SparkConf().setAppName("Spark Pi")

/\*\* Create the SparkContext \*/

val spark = new SparkContext(conf)

/\*\* business logic to calculate Pi \*/

val slices = if (args.length > 0) args(0).toInt else 2

val n = math.min(100000L \* slices, Int.MaxValue).toInt // avoid overflow

val count = spark.parallelize(1 until n, slices).map { i =>

val x = random \* 2 - 1

val y = random \* 2 - 1

if (x\*x + y\*y < 1) 1 else 0

}.reduce(\_ + \_)

/\*\* Printing the value of Pi \*/

println("Pi is roughly " + 4.0 \* count / n)

/\*\* Stop the SparkContext \*/

spark.stop()

}

}

* 在SparkPi目录下新建sparkpi.sbt文件，内容如下：

name := "SparkPi Project"

version := "1.0"

scalaVersion := "2.10.4"

libraryDependencies += "org.apache.spark" %% "spark-core" % "1.2.1"

* 在SparkPi目录下运行：sbt package

首次运行时间较长（2016/10/04实测约40分钟），因为需要下载许多依赖文件包

* 在SparkPi目录下运行：

spark-submit --class SparkPi --master local[4] target/scala-2.10/sparkpi-project\_2.10-1.0.jar

在输出结果中可以看到Pi计算值

**四、用Python创建App：PythonPi**

* 搭建App架构：

在工作目录下创建PythonPi子目录

* 在PythonPi目录下新建PythonPi.py文件，内容如下：

#Import statements

import sys

from random import random

from operator import add

from pyspark import SparkContext

if \_\_name\_\_ == "\_\_main\_\_":

"""

Usage: pi [partitions]

"""

#Create the SparkContext

sc = SparkContext(appName="PythonPi")

#Run the calculations to estimate Pi

partitions = int(sys.argv[1]) if len(sys.argv) > 1 else 2

n = 100000 \* partitions

def f(\_):

x = random() \* 2 - 1

y = random() \* 2 - 1

return 1 if x \*\* 2 + y \*\* 2 < 1 else 0

#Create the RDD, run the transformations, and action to calculate Pi

count = sc.parallelize(xrange(1, n + 1), partitions).map(f).reduce(add)

#Print the value of Pi

print "Pi is roughly %f" % (4.0 \* count / n)

#Stop th SparkContext

sc.stop()

* 在PythonPi目录下运行：spark-submit --master local[4] PythonPi.py

在输出结果中可以看到Pi计算值

**五、用Java创建App：WordCount**

* 搭建App架构：

在工作目录下创建WordCount子目录

在WordCount下创建src子目录

在WordCount/src下创建main子目录

在WordCount/src/main下创建java子目录

* 在WordCount/src/main/java目录下新建WordCount.java文件，内容如下：

/\*\* Import the required classes \*/

import scala.Tuple2;

import org.apache.spark.SparkConf;

import org.apache.spark.api.java.JavaPairRDD;

import org.apache.spark.api.java.JavaRDD;

import org.apache.spark.api.java.JavaSparkContext;

import org.apache.spark.api.java.function.FlatMapFunction;

import org.apache.spark.api.java.function.Function2;

import org.apache.spark.api.java.function.PairFunction;

import java.util.Arrays;

import java.util.List;

import java.util.regex.Pattern;

/\*\* Setting up the class \*/

public final class WordCount {

private static final Pattern SPACE = Pattern.compile(" ");

public static void main(String[] args) throws Exception {

if (args.length < 1) {

System.err.println("Usage: JavaWordCount <file>");

System.exit(1);

}

/\*\* Create the SparkConf \*/

SparkConf sparkConf = new SparkConf().setAppName("WordCount");

/\*\* Create the SparkContext \*/

JavaSparkContext ctx = new JavaSparkContext(sparkConf);

/\*\* Create the RDDs and apply transformation and actions on them \*/

JavaRDD<String> lines = ctx.textFile(args[0], 1);

JavaRDD<String> words = lines.flatMap(

new FlatMapFunction<String, String>() {

@Override

public Iterable<String> call (String s) {

return Arrays.asList(SPACE.split(s));

}

}

);

/\*\* Mapping each word to a 1 \*/

JavaPairRDD<String, Integer> ones = words.mapToPair(

new PairFunction<String, String, Integer>() {

@Override

public Tuple2<String, Integer> call(String s) {

return new Tuple2<String, Integer>(s, 1);

}

}

);

/\*\* Adding up the values \*/

JavaPairRDD<String, Integer> counts = ones.reduceByKey(

new Function2<Integer, Integer, Integer>() {

@Override

public Integer call(Integer i1, Integer i2) {

return i1 + i2;

}

}

);

/\*\* Invoke an action to get it to return the values \*/

List<Tuple2<String, Integer>> output = counts.collect();

for (Tuple2<?,?> tuple : output) {

System.out.println(tuple.\_1() + ": " + tuple.\_2());

}

/\*\* Stop the SparkContext \*/

ctx.stop();

}

}

* 在WordCount目录下新建pom.xml文件，内容如下：

<project>

<groupId>edu.berkeley</groupId>

<artifactId>word-count</artifactId>

<modelVersion>4.0.0</modelVersion>

<name>Word Count</name>

<packaging>jar</packaging>

<version>1.0</version>

<dependencies>

<dependency> <!-- Spark dependency -->

<groupId>org.apache.spark</groupId>

<artifactId>spark-core\_2.10</artifactId>

<version>1.2.1</version>

</dependency>

</dependencies>

</project>

* 在WordCount目录下运行：mvn package

首次运行时间较长（2016/10/04实测约20分钟），因为需要下载许多依赖文件包

* 复制C:\spark\README.md至WordCount目录下
* 在WordCount目录下运行：

spark-submit --class WordCount --master local[4] target/word-count-1.0.jar README.md