# Spark 调研

北京网感至察科技有限公司

2015年4月1日

## 运行环境安装

#### 操作系统

此系统要求运行于基于Unix的操作系统（Linux, Max Os等)之上，测试环境为Ubuntu 12.04 LTS.

#### 必备软件库与语言环境

Spark的运行环境要求JDK 1.6+, Scala 2.9+. 在缺省安装上述操作系统之后，运行

java -version

scala -version

检查JDK和Scala环境。若不符合要求，以下提供了JDK和Scala环境的安装步骤，对应的软件库为jdk1.7.0\_60和scala-2.10.5. sbt提供了类似Maven的项目构建功能，推荐安装。测试的Spark版本为对应Hadoop2.4的预编译版本Spark1.3.0-bin-hadoop2.4.

1. [可选] JDK 安装:

前往Oracle网站下载jdk安装包，假定为jdk1.7.0\_60.tar.gz, 下载路径为~/Downloads

运行：

cd /usr/lib/jvm/

sudo mv ~/Downloads/JDK.tar.gz ./ (需要提供管理员密码）

tar zxvf jdk1.7.0\_60.tar.gz

2. [可选] Scala安装：

运行：

cd /usr/local/lib

wget http://www.scala-lang.org/files/archive/scala-2.10.5.tgz

tar zxvf scala-2.10.5.tgz

3. Spark安装：

cd /usr/local/lib

wget http://www.apache.org/dyn/closer.cgi/spark/spark-1.3.0/spark-1.3.0-bin-hadoop2.4.tgz

tar zxvf spark-1.3.0-bin-hadoop2.4.tgz

4. [可选] sbt安装：

echo "deb https://dl.bintray.com/sbt/debian /" | sudo tee -a /etc/apt/sources.list.d/sbt.list

sudo apt-get update

sudo apt-get install sbt

5. 系统环境变量配置:

编辑~/.bashrc文件，在文件末尾添加：

export JAVA\_HOME="/usr/lib/jvm/jdk1.7.0\_60"

export SCALA\_HOME="/usr/local/lib/scala-2.10.5"

export SPARK\_HOME="/usr/local/lib/spark-1.3.0-bin-hadoop2.4/"

export PATH="$JAVA\_HOME/bin:$SPARK\_HOME/bin:$SCALA\_HOME/bin:$PATH"

## Spark配置与测试运行（单机）

以下以WordCount为例，测试Spark的交互运行和创建应用两种运行方式。

#### REPL交互运行

进入Spark Shell：

spark-shell

Spark中，RDD的创建有两种方式，分别是从输入流读取（比如HDFS file）和从其他RDD转化。假定待处理文件为/home/sid/test.txt, 文件内每行为一个单词。

在Spark Shell内运行：

val lines = sc.textFile("/home/sid/test.txt")

val count = lines.flatMap(line => line.split(" " )).map(word => (word, 1)).reduceByKey(\_+\_)

这里， flatMap, map, reduceByKey操作都生成了一个新的RDD，最终生成count，其形式为(String, Int)元组，对应每个单词出现的次数：

count.foreach(println(\_))

可以将每个元组依次输出。

#### 创建Spark应用

Spark应用通过spark-submit进行提交。编写scala程序时，需要导入 SparkContext和 SparkConf.

SimpleApp.scala程序：

import org.apache.spark.SparkContext

import org.apache.spark.SparkContext.\_

import org.apache.spark.SparkConf

object SimpleApp {

def main(args: Array[String]) {

val filename = "/home/sid/test.txt"

val conf = new SparkConf().setAppName("Word Count")

val sc = new SparkContext(conf)

val lines = sc.textFile(filename).cache()

val counts = lines.flatMap(line => line.split(" ")).map(word => (word, 1)).reduceByKey(\_+\_)

counts.foreach(println(\_))

}

}

Simple.sbt配置文件：

name := "Simple Project"

version := "1.0"

scalaVersion := "2.10.5"

libraryDependencies += "org.apache.spark" %% "spark-core" % "1.3.0"

运行sbt编译：

sbt package

编译结束后，生成jar文件，假定为simple-project\_2.10-1.0.jar.

利用spark-submit提交，提交时可以制定参数：

spark-submit –master local[4] simple-project\_2.10-1.0.jar

## Postgre SQL数据导入

首先安装PostgreSQL, 运行：

sudo apt-get install postgresql-9.1

假定数据库用户名postgres, 密码为123, 待导入的数据sql文件为gflux.sql，运行：

psql -U postgres -h localhost -d gflux -f gflux.sql

则sql文件里的数据可以导入至PostgreSQL数据库

## Spark中生成DataFrame

DataFrame是Spark中对关系型数据结构的抽象，可以类比为数据库中的一个表。Spark1.3支持创建DataFrame的方式有以下4种：结构化的数据文件(比如JSON); Hive表; 外部数据库(通过JDBC); 已有的RDD.

1. 下载对应的jdbc驱动

假定下载文件为postgresql-9.1-903.jdbc4.jar，文件存储至$SPARK\_HOME/lib文件夹下。

2. 指定Spark的classpath：

编辑$SPARK\_HOME/conf下spark\_env.sh文件，加入：

export SPARK\_CLASSPATH=.:$SPARK\_HOME/lib/postgresql-9.1-903.jdbc4.jar:$SPARK\_CLASSPATH

3. 生成DataFrame (以java为例，读取导入数据中里的item表)

**import** java.util.HashMap;

**import** java.util.Map;

**import** org.apache.spark.api.java.\*;

**import** org.apache.spark.SparkConf;

**import** org.apache.spark.sql.DataFrame;

**import** org.apache.spark.sql.SQLContext;

**public** **class** Test {

**public** **static** **void** **main**(String[] **args**) {

//创建SparkContext上下文

JavaSparkContext **sc** = **new** JavaSparkContext(**new** SparkConf().setAppName("JDBC demo"));

//创建SQLContext上下文

SQLContext **sqc** = **new** SQLContext(sc);

//创建选项字典

Map<String, String> **options** = **new** HashMap<String, String>();

String **user** = "postgres";

String **password** = "123";

options.put("url", "jdbc:postgresql://localhost:5432/gflux?user="

+ user + "&password=" + password);

options.put("dbtable", "item");

//生成DataFrame

DataFrame **jdbcDF** = sqc.load("jdbc", options);

//展示DataFrame

jdbcDF.show();

}

}

最后运行结果应为：

