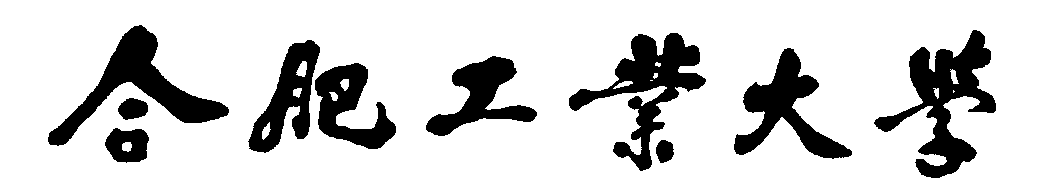
****

分布式程序设计

实验报告

|  |  |
| --- | --- |
| 专 业 班 级 | 计算机科学与技术21-1班 |
| 学生姓名及学号 | 2021214710-杨程锦 |
| 课程教学班号 | 0500020X--001 |
| 任 课 教 师 | 张赞 |
| 实验指导教师 | 张赞 |
| 实验地点 | 翡翠湖校区科教楼C座机房 |
| 2023～2024学年第 一 学期 | |

说 明

实验报告是关于实验教学内容、过程及效果的记录和总结，因此，应注意以下事项和要求：

1．每个实验单元在50页的篇幅内完成一份报告。“实验单元”指按照实验指导书规定的实验内容。若篇幅不够，可另附纸。

2、各实验的**预习部分的内容是进入实验室做实验的必要条件**，请按要求做好预习。

3．实验报告要求：**书写工整规范，语言表达清楚，数据和程序真实。理论联系实际，认真分析实验中出现的问题与现象，总结经验。**

4．参加实验的每位同学应独立完成实验报告的撰写，其中程序或相关的设计图纸也可以采用打印等方式粘贴到报告中。严禁抄袭或拷贝，否则，一经查实，按作弊论取，并取消理论课考试资格。

5．实验报告作为评定实验成绩的依据。

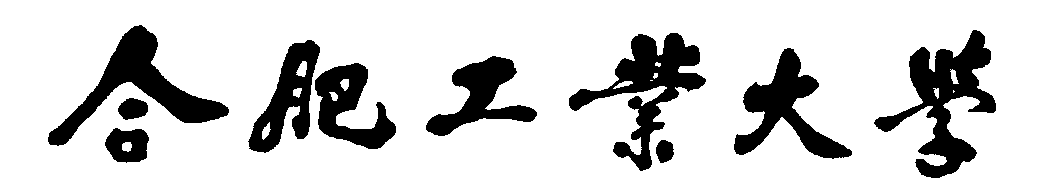
**实验序号及名称：实验 一**  Spark的本地安装

实验时间∶ 2023年12月 7日

|  |
| --- |
| **预习内容** |
| **一、实验目的和要求∶**  在本地虚拟机安装Spark |
| **二、实验任务∶**  在本地虚拟机中安装Spark并可以通过测试 |
| **三、实验准备方案，包括以下内容：**  1、安装VMware  2、安装Ubuntu 20.04.3 |

|  |
| --- |
| **实验内容** |
| **一、实验用仪器、设备：**  操作系统：Windows 10专业版  虚拟机：Ubuntu 20.04.3-desktop |
| **二、实验内容与步骤（过程及数据记录）：**  （一）、安装并配置Hadoop  1、创建一个名为Hadoop的用户  **sudo useradd -m hadoop -s /bin/bash**    2、设置密码  **sudo passwd hadoop**  3、为hadoop用户添加管理员权限  **sudo adduser hadoop sudo**  4、切换hadoop用户，更新apt以及安装vim  5、安装SSH serve  **sudo apt-get install openssh-server**  6、登陆主机并配置为无需密码登陆  **cd ~/.ssh/**  **ssh-keygen -t rsa**  **cat ./id\_rsa.pub >> ./authorized\_keys**  **ssh localhost**    7、解压JDK1.8  **cd /usr/lib**  **sudo mkdir jvm**  **cd ~**  **sudo tar -zxvf ./jdk-8u162-linux-x64.tar.gz -C/usr/lib/jvm**    8、配置环境变量  **cd ~**  **vim ~/.bashrc**  **按esc返回，输入：wq保存并退出**    **source ~/.bashrc**  **java -version**  文本  描述已自动生成  9、安装hadoop  **sudo tar -zxvf ./hadoop-3.3.1.tar.gz -C /usr/local**  **cd /usr/local/**  **ls**  **sudo mv ./hadoop-3.3.1/ ./hadoop**  **sudo chown -R hadoop ./hadoop**  **cd /usr/local/hadoop**  **./bin/hadoop version**    10、配置hadoop  **cd /usr/local/Hadoop**  **gedit ./etc/hadoop/core-site.xml**  将框中内容改为  **<configuration>**  **<property>**  **<name>hadoop.tmp.dir</name>**  **<value>file:/usr/local/hadoop/tmp</value>**  **<description>Abase for other temporary directories.</description>**  **</property>**  **<property>**  **<name>fs.defaultFS</name>**  **<value>hdfs://localhost:9000</value>**  **</property>**  **</configuration>**    将框中内容改为  **<configuration>**  **<property>**  **<name>dfs.replication</name>**  **<value>1</value>**  **</property>**  **<property>**  **<name>dfs.namenode.name.dir</name>**  **<value>file:/usr/local/hadoop/tmp/dfs/name</value>**  **</property>**  **<property>**  **<name>dfs.datanode.data.dir</name>**  **<value>file:/usr/local/hadoop/tmp/dfs/data</value>**  **</property>**  **</configuration>**    **cd /usr/local/hadoop**  **./bin/hdfs namenode -format**    11、开启NameNode和DataNode守护进程  **cd /usr/local/hadoop**  **./bin/hdfs namenode -format**  **jps**    12、访问Web界面http://localhost:9870    （二）安装并配置Spark  1、解压并配置Spark  **sudo tar -zxf ~/下载/spark-3.5.0-bin-without-hadoop.tgz -C /usr/local/**  **cd /usr/local**  **sudo mv ./spark-3.5.0-bin-without-hadoop/ ./spark**  **sudo chown -R hadoop:hadoop ./spark**  **cd /usr/local/spark**  **cp ./conf/spark-env.sh.template ./conf/spark-env.sh**  **vim ./conf/spark-env.sh**  在第一行添加：  **export SPARK\_DIST\_CLASSPATH=$(/usr/local/hadoop/bin/hadoop classpath)**    **vim ~/.bashrc**    在.bashrc 文件中添加如下内容:  **export HADOOP\_HOME=/usr/local/hadoop**  **export SPARK\_HOME=/usr/local/spark**  **export PYTHONPATH=$SPARK\_HOME/python:$SPARK\_HOME/python/lib/py4j-**  **0.10.9.7-src.zip:$PYTHONPATH**  **export PYSPARK\_PYTHON=python3**  **export PATH=$HADOOP\_HOME/bin:$SPARK\_HOME/bin:$PATH**  使环境变量生效  **source ~/.bashrc**  2、验证Spark  **cd /usr/local/spark**  **bin/run-example SparkPi**  **bin/run-example SparkPi 2>&1 | grep "Pi is"**    **bin/pyspark**    **3\*7**  **exit()**    **cd ~**  **vim test.py** |

|  |
| --- |
|  |
| **三、实验结果分析、思考题解答∶**  本实验成功的安装了Hadoop以及Spark，成功的运行了一些简单的实例。 |
| **四、感想、体会、建议∶**  通过本实验，我学会了一些基础的大数据工具的安装以及简单的使用，还对Linux的操作指令更加的娴熟了。 |
| **实验成绩∶**  **指导教师签名：**  年 月 日 |

****

分布式程序设计

实验报告

|  |  |
| --- | --- |
| 专 业 班 级 | 计算机科学与技术21-1班 |
| 学生姓名及学号 | 2021214710-杨程锦 |
| 课程教学班号 | 0500020X--001 |
| 任 课 教 师 | 张赞 |
| 实验指导教师 | 张赞 |
| 实验地点 | 翡翠湖校区科教楼C座机房 |
| 2023～2024学年第 一 学期 | |

说 明

实验报告是关于实验教学内容、过程及效果的记录和总结，因此，应注意以下事项和要求：

1．每个实验单元在50页的篇幅内完成一份报告。“实验单元”指按照实验指导书规定的实验内容。若篇幅不够，可另附纸。

2、各实验的**预习部分的内容是进入实验室做实验的必要条件**，请按要求做好预习。

3．实验报告要求：**书写工整规范，语言表达清楚，数据和程序真实。理论联系实际，认真分析实验中出现的问题与现象，总结经验。**

4．参加实验的每位同学应独立完成实验报告的撰写，其中程序或相关的设计图纸也可以采用打印等方式粘贴到报告中。严禁抄袭或拷贝，否则，一经查实，按作弊论取，并取消理论课考试资格。

5．实验报告作为评定实验成绩的依据。

**实验序号及名称：实验 二**  Spark编程

实验时间∶ 2023年12月 14日

|  |
| --- |
| **预习内容** |
| **一、实验目的和要求∶**  练习Spark编程基本操作 |
| **二、实验任务∶**  通过头歌实践平台联系Spark编程，进行基本的数据操作 |
| **三、实验准备方案，包括以下内容：**  阅读头歌实践教学平台的编程要求 |

|  |
| --- |
| **实验内容** |
| **一、实验用仪器、设备：**  头歌实践教学平台 |
| **二、实验内容与步骤（过程及数据记录）：**  第一关：  from pyspark import SparkContext  ################ Begin ################  # 创建SparkContext对象  sc = SparkContext("local", "count\_students")  # 读取文本文件  data = sc.textFile("/data/bigfiles/data.txt")  # 统计学生人数  num\_students = data.map(lambda line: line.split(",")[0]).distinct().count()  # 打印结果  print("\n该系共有学生：" + str(num\_students) + " 人")  # 关闭SparkContext对象  sc.stop()  ################ End ################  本关通过pyspark读取linux系统本地文件/data/bigfiles/data.txt，其中统计了某大学计算机系的成绩，最终输出该系的学生个数。  第二关：  from pyspark import SparkContext  ################ Begin ################  # 创建SparkContext对象  sc = SparkContext("local", "count\_courses")  # 读取文本文件  data = sc.textFile("/data/bigfiles/data.txt")  # 统计课程数量  num\_courses = data.map(lambda line: line.split(",")[1]).distinct().count()  # 打印结果  print("\n该系共有 {} 门课程".format(num\_courses))  # 关闭SparkContext对象  sc.stop()  ################ End ################  本关通过pyspark读取linux系统本地文件/data/bigfiles/data.txt，其中统计了某大学计算机系的成绩，最终输出该系开设的课程数。  第三关：  from pyspark import SparkContext  ################ Begin ################  # 创建SparkContext对象  sc = SparkContext("local", "Tom Average Score")  # 读取文本文件  lines = sc.textFile("file:///data/bigfiles/data.txt")  # 计算 Tom 同学的平均分  tom\_scores = lines.filter(lambda x: x.startswith("Tom")).map(lambda x: float(x.split(",")[2]))  average\_score = tom\_scores.mean()  # 打印结果  print("\nTom 同学的总成绩平均分为：%.1f 分" % average\_score)  # 关闭SparkContext对象  sc.stop()  ################ End ################  在 pyspark 中读取 Linux 系统本地文件 /data/bigfiles/data.txt，该数据集包含了某大学计算机系的成绩，要求计算 Tom 同学的总成绩平均分是多少（保留一位小数），直接打印输出计算的结果。  第四关：  from pyspark import SparkContext  ################ Begin ################  # 创建SparkContext对象  sc = SparkContext("local", "Courses Count")  # 读取文本文件  lines = sc.textFile("file:///data/bigfiles/data.txt")  # 统计各个同学选修课程数量  res = lines.map(lambda x:x.split(",")).map(lambda x:(x[0],1)) #学生每门课程都对应  each\_res = res.reduceByKey(lambda x,y: x+y) #按学生姓名获取每个学生的选课总数  # 打印结果  each\_res.foreach(print)  # 关闭SparkContext对象  sc.stop()  ################ End ################  在 pyspark 中读取 Linux 系统本地文件 /data/bigfiles/data.txt，该数据集包含了某大学计算机系的成绩，求每名同学选修的课程门数，直接打印输出计算的结果。  第五关：  from pyspark import SparkContext  ################ Begin ################  # 创建SparkContext对象  sc = SparkContext("local", "DataBase Students Count")  # 读取文本文件  lines = sc.textFile("file:///data/bigfiles/data.txt")  # 统计选修 DataBase 课程的人数  arr = lines.map(lambda x:x.split(",")).filter(lambda x:x[1]=="DataBase")  res = arr.count()  # 打印结果  print("\n选修 DataBase 课程的同学共有 %d 位" % res)  # 关闭SparkContext对象  sc.stop()  ################ End ################  在 pyspark 中读取 Linux 系统本地文件 /data/bigfiles/data.txt，该数据集包含了某大学计算机系的成绩，统计该系 DataBase 课程共有多少人选修，直接打印输出计算的结果。  第六关：  from pyspark import SparkContext  ################ Begin ################  # 创建SparkContext对象  sc = SparkContext("local", "Courses Average")  # 读取文本文件  lines = sc.textFile("file:///data/bigfiles/data.txt")  # 统计该系各门课程的平均分  res = lines.map(lambda x: x.split(",")).map(lambda x: (x[1], (int(x[2]), 1)))  temp = res.reduceByKey(lambda x, y: (x[0] + y[0], x[1] + y[1])) # 按课程名聚合课程总分和选课人数  # 课程总分/选课人数 = 平均分，保留一位小数  avg = temp.map(lambda x: (x[0], round(x[1][0] / x[1][1], 1)))  # 打印结果  avg.foreach(print)  # 关闭SparkContext对象  sc.stop()  ################ End ################  在 pyspark 中读取 Linux 系统本地文件 /data/bigfiles/data.txt，该数据集包含了某大学计算机系的成绩，统计该系各门课程的平均分是多少（保留一位小数），直接打印输出计算的结果。  第七关：  from pyspark import SparkContext  from pyspark.accumulators import AccumulatorParam  ################ Begin ################  # 创建SparkContext对象  sc = SparkContext("local", "DataBase Students Count")  # 创建累加器  data\_base\_students\_count = sc.accumulator(0)  # 定义累加器参数  class MyAccumulatorParam(AccumulatorParam):  def zero(self, value):  return value  def addInPlace(self, acc1, acc2):  return acc1 + acc2  # 读取文本文件  lines = sc.textFile("file:///data/bigfiles/data.txt")  # 使用累加器统计选修 DataBase 课程的人数  def count\_data\_base\_students(line):  global data\_base\_students\_count  if line.split(",")[1] == "DataBase":  data\_base\_students\_count += 1  # 使用 foreach 方法遍历 RDD 并应用累加器  lines.foreach(count\_data\_base\_students)  # 打印结果  print("\n共有 %d 名同学选修了 DataBase 课程" % data\_base\_students\_count.value)  # 关闭SparkContext对象  sc.stop()  ################ End ################ |

|  |
| --- |
|  |
| **三、实验结果分析、思考题解答∶**  第一关：    第二关：    第三关：    第四关：    第五关：    第六关：    第七关： |
| **四、感想、体会、建议∶**  通过本实验我对于Spark编程更加熟练了，对于python中的lambda表达式的运用更加娴熟，同样学会了累加器的操作，并且学到了使用foreach方法遍历RDD并应用累加器。 |
| **实验成绩∶**  **指导教师签名：**  年 月 日 |

**卡通人物

中度可信度描述已自动生成**

分布式程序设计

实验报告

|  |  |
| --- | --- |
| 专 业 班 级 | 计算机科学与技术21-1班 |
| 学生姓名及学号 | 2021214710-杨程锦 |
| 课程教学班号 | 0500020X--001 |
| 任 课 教 师 | 张赞 |
| 实验指导教师 | 张赞 |
| 实验地点 | 翡翠湖校区科教楼C座机房 |
| 2023～2024学年第 一 学期 | |

说 明

实验报告是关于实验教学内容、过程及效果的记录和总结，因此，应注意以下事项和要求：

1．每个实验单元在50页的篇幅内完成一份报告。“实验单元”指按照实验指导书规定的实验内容。若篇幅不够，可另附纸。

2、各实验的**预习部分的内容是进入实验室做实验的必要条件**，请按要求做好预习。

3．实验报告要求：**书写工整规范，语言表达清楚，数据和程序真实。理论联系实际，认真分析实验中出现的问题与现象，总结经验。**

4．参加实验的每位同学应独立完成实验报告的撰写，其中程序或相关的设计图纸也可以采用打印等方式粘贴到报告中。严禁抄袭或拷贝，否则，一经查实，按作弊论取，并取消理论课考试资格。

5．实验报告作为评定实验成绩的依据。

**实验序号及名称：实验 三**  出租车轨迹分析(Python)

实验时间∶ 2023年12月21 日

|  |
| --- |
| **预习内容** |
| **一、实验目的和要求∶**  通过Spark进行数据清洗和数据分析 |
| **二、实验任务∶**  利用正则表达式进行数据清洗  利用SparkSQL进行数据分析 |
| **三、实验准备方案，包括以下内容：**  阅读头歌实践教学平台的编程要求 |

|  |
| --- |
| **实验内容** |
| **一、实验用仪器、设备：**  头歌实践教学平台 |
| **二、实验内容与步骤（过程及数据记录）：**  第一关SparkSQL数据清洗：  # -\*- coding: UTF-8 -\*-  from pyspark.sql import SparkSession  if \_\_name\_\_ =='\_\_main\_\_':  spark = SparkSession.builder.  master("local").appName("demo").getOrCreate()  #\*\*\*\*\*\*\*\*\*\*begin\*\*\*\*\*\*\*\*\*\*#  df = spark.read.option("header",True).  option("delimiter","\t").csv("/root/data.csv")  df.createTempView("data")  spark.sql("""  select regexp\_replace(TRIP\_ID,'\\\W+','') as TRIP\_ID ,  regexp\_replace(CALL\_TYPE,'\\\W+','') as CALL\_TYPE ,  regexp\_replace(ORIGIN\_CALL,'\\\W+','') as ORIGIN\_CALL ,  regexp\_replace(TAXI\_ID,'\\\W+','') as TAXI\_ID ,  regexp\_replace(ORIGIN\_STAND,'\\\W+','') as ORIGIN\_STAND ,  regexp\_replace(TIMESTAMP,'\\\W+','') as TIMESTAMP ,  regexp\_replace(POLYLINE,'\\\W+','') as POLYLINE  from data  """).show()  #\*\*\*\*\*\*\*\*\*\*end\*\*\*\*\*\*\*\*\*\*#  spark.stop()  本关通过SparkSQL读取CSV文件，option参数说明：  header为true：将 CSV 第一行数据作为头部信息，换一句来说，就是将 CSV 的第一行数据作为 SparkSQL 表的字段  delimiter：分隔符，例如，CSV 文件默认以英文逗号进行字段分隔，那么 delimiter 为英文逗号，如果文件以分号进行字段分隔，那么 delimiter 为分号  使用SparkSQL内置字符串处理函数，使用正则表达式进行搜索与予预期结果匹配的文本。将出租车轨迹数据规整化,清洗掉多余的字符串，并使用 DataFrame.show() 打印输出。  第二关SparkSQL数据分析：  # -\*- coding: UTF-8 -\*-  from pyspark.sql import SparkSession  import json  if \_\_name\_\_ == '\_\_main\_\_' :  spark = SparkSession.builder.  master("local").appName("demo").getOrCreate()  #\*\*\*\*\*\*\*\*\*\*begin\*\*\*\*\*\*\*\*\*\*#  df = spark.read.option("header",True).  option("delimiter","\t").csv("/root/data2.csv")  df.createTempView("data")  spark.sql("select TRIP\_ID,CALL\_TYPE,  ORIGIN\_CALL, TAXI\_ID, ORIGIN\_STAND,  from\_unixtime(TIMESTAMP,'yyyy-MM-dd')  as TIME ,POLYLINE from data").show()  spark.udf.register("timeLen", lambda x: {  (len(json.loads(x)) - 1) \* 15 if len(json.loads(x)) > 0 else 8  })  spark.udf.register("startLocation", lambda x: {  str(json.loads(x)[0]) if len(json.loads(x)) > 0 else ""  })  spark.udf.register( "endLocation", lambda x: {  str(json.loads(x)[len(json.loads(x)) - 1])  if len(json.loads(x)) > 0 else ""  })  df.createTempView("data2")  res=spark.sql("select TRIP\_ID,CALL\_TYPE,ORIGIN\_CALL,  TAXI\_ID,ORIGIN\_STAND,from\_unixtime(TIMESTAMP,'yyyy-MM-dd')  as TIME, POLYLINE, timeLen(POLYLINE)  as TIMELEN, startLocation(POLYLINE)  as STARTLOCATION, endLocation(POLYLINE)  as ENDLOCATION from data2")  res.createTempView("data3")  res.show()  spark.sql("select CALL\_TYPE,TIME,count(1)  as NUM from data3 group by TIME,CALL\_TYPE  order by CALL\_TYPE,TIME").show()  #\*\*\*\*\*\*\*\*\*\*end\*\*\*\*\*\*\*\*\*\*#  本关使用SparkSQL进行数据分析，利用SparkSQL UDF对表中单行进行转换，以便为每行生成单个对应的输出值。用户自定义函数可以在Spark SQL中定义和注册为UDF，并且可以关联别名，这个别名可以在后面的SQL查询中使用。Spark SQL定义了UDF1到UDF22共22个类，UDF最多支持22 个输入参数。如果我们不想修改Apache Spark的源代码，对于需要超过22个输出参数的应用程序我们可以使用数组或结构作为参数来解决这个问题。 |

|  |
| --- |
|  |
| **三、实验结果分析、思考题解答∶**  第一关：    第二关： |
| **四、感想、体会、建议∶**  通过本实验，我对于SparkSQL的数据分析和数据清洗部分更加深入了解了，掌握了正则表达式清洗数据的技术以及掌握了利用SparkSQL UDF进行数据分析，通过此实验，我对于大数据技术的学习更加深入，更加了解了大数据的内涵了。 |
| **实验成绩∶**  **指导教师签名：**  年 月 日 |