第三期任务：Spark 实战

一、机器情况

操作系统：Windows 7 Service Pack 1 64 位专业版
CPU：Inter(R) Core(TM) i7-4770 @ 3.40GHz 四核八线程
内存：8.00 GB
硬盘：2 TB(ST2000DM 001-1CH164 SCSI Disk Device)

二、前期安装与环境配置

安装 Spark 之前需要先安装好 Scala、SBT、GitHub 以及 Java，本次任务中使用的各版本情况与下载地址如下：

http://www.scala-lang.org/download/ scala-2.11.6.msi http://www.scala-sbt.org/download.html sbt-0.13.8.msi http://msysgit.github.io/ Git-1.9.5-preview20150319.exe http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html jdk-8u45-windows-x64.exe

安装好之后需要设置一些环境变量，首先要向 PATH 中添加以下几项：

C:\Program Files (x86)\scala\bin;
C:\Program Files (x86)\sbt\bin;
C:\Program Files (x86)\Git\bin;
C:\Program Files\Java\jdk1.8.0_45\bin;

之后添加新的环境变量如下：

SCALA_HOME C:\Program Files (x86)\scala\ SBT_HOME C:\Program Files (x86)\sbt\ JAVA_HOME C:\Program Files\Java\jdk1.8.0_45 CLASSPATH C:\Program Files\Java\jdk1.8.0_45\lib;

至此，Spark 依赖的程序和变量都已经具备，可以开始安装了。

三、下载与安装 Spark

本次任务安装的是 Spark 1.3.1 版本，具体地址和文件信息如下：

http://spark.apache.org/downloads.html
1.3.1 (Apr 17 2015)
Source Code
spark-1.3.1.tgz

把 spark-1.3.1.tgz 解压缩之后，以管理员模式运行 CMD.exe，并来到解压的根目录下。特别要注意，整个文件路径不可以有空格！

在解压缩后的根目录里执行命令：sbt package，这会将 Spark 所有依赖的 lib 下载到本地的 C:\Users\你的用户名\.ivy2\cache 里面。完成之后，Spark 依赖的库已经具备，执行命令：sbt assembly，这会将 Spark 依赖的库和 Spark 本身 assemble 成很大的 jar 包。这两步加起来要一小时左右，依赖网速以及机器速度，最终下载的所有 lib 文件放在附录里。

此后你会在 assembly\target\scala-2.11.6 下找到 spark-assembly-1.3.1-hadoop1.0.4.jar、spark-assembly_2.11.6-1.3.1.jar 这两个 jar 包。

上述步骤完成之后，就可以在 bin 目录下执行命令：spark-shell，可以成功进入 shell，进行操作。

本人在下载 lib 的过程中遇到几次如下的报错，可能是因为某些服务器暂时链接不上，输入 r 重试即可，一般重试一次就可以执行下去。所以安装过程还是需要有人全程值守的。

[error] ({https://github.com/ScrapCodes/sbt-pom-reader.git#ignore_artifact_id}sbt-pom-reader/*: update) sbt.ResolveException: download failed: org.jsoup#jsoup;1.6.1!jsoup.jar

[error] download failed: commons-logging#commons-logging;1.1.1!commons-logging.jar

Project loading failed: (r)etry, (q)uit, (l)ast, or (i)gnore?

四、编程

使用 Scala 语言完成编程，代码见 MyCode.txt。

4.1、读取数据

原始数据以 txt 格式存储，所以直接使用 sc.textFile()函数读取即可。由于数据分成了两部分，再使用 sc.textFile() union sc.textFile()的方式把数据拼接在一起。

数据的每一行作为一个单位进行读取，由于 UserID(即电话号码)太长无法以 Int 类型存储，所以程序中所有数据均使用 Double 类型。StartTime 的格式是 03-15-11-05-09，由于没有找到详细的说明，个人根据所有数据的情况对它的理解是“月份-日期-小时-分钟-秒钟”，研究的问题里只需要用到月份和日期，所以忽略具体时间信息，存储数据时该字段存储为(Double)315。

4.2、数据分析

一共对 9 个问题进行了研究，程序中每个问题的最终答案放在变量 ques1、ques2…ques9 里，输出的结果放在 Result1.xlsx 和 Result2-Q6Q7.xlsx 的不同工作表里。由于第 6、第 7 题的答案太大，所以将这两个题单独放在了一个 EXCEL 里。

具体问题内容如下：

Q1：统计每大类业务数据总量；

Q2：统计每大类业务中每小类业务总量；

Q3：统计每小区业务总量；

Q4：统计每小区业务平均速率（总业务数量/总业务时长）；

Q5：统计每用户业务总量；

Q6：统计每用户不同大类业务使用次数和该类业务总量；

Q7：统计每用户不同小类业务使用次数和该类业务总量；

Q8：统计每天业务总量；

Q9：统计每天业务平均速率（总业务数量/总业务时长）；

对这些问题的处理思路如下：

Q1：把 BussinessType.A 设为 Key，与 DataUsage 组成 Tuple2，使用 reduceByKey(_+_)获得每个大类业务的数据总量。

Q2：由于小类的编号都是以大类编号开头，所以直接使用 BussinessType.B 即可，处理方式与 Q1 相同，把 BussinessType.B 设为 Key 即可。

Q3：同 Q1，把 CommunityID 设为 Key 即可。

Q4：首先要获得每小区的业务总时长，同 Q3，把 Tuple2 中的 DataUsage 换成 Duration 即可。之后把这两个 RDD 通过 join 函数映射在一起，再利用自行设计的 devideKey 函数求出总业务数量/总业务时长 的值作为平均速率。

Q5：同 Q1，把 UserID 设为 Key 即可。

Q6：利用 eachUser 函数把每条记录中的 UserID、BussinessType.A、DataUsage 抽取出来，并将 UserID 设为 Key。使用 groupByKey 把每个用户的信息合并成一条，再利用 countKey 函数进行统计。

Q7：同 Q6，把 BussinessType.A 换成 BussinessType.B 即可。

Q8：同 Q1，把 StartTime 设为 Key 即可。

Q9：同 Q4，并利用 Q8 中的结果。

程序运行顺畅，速度很快，体现了 Spark 的优秀性能。