实验三:Spark-SQL 分布式数据库部署

22121630 汪江豪

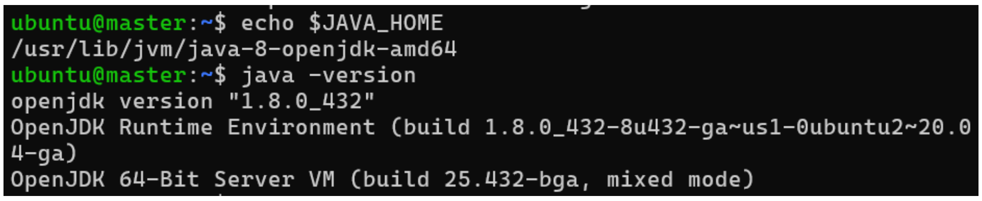
实验内容：

实验数据集记录了 2016 年以来 50 个州和华盛顿特区的出生率和相关数据。根据要求进行相关查询：

1. 按洲和年份分组查询，出生婴儿的男女比例。
   输出格式：(State，Year，MRatio, FRatio)
2. 按年份分组查询，查询母亲受教育程度与平均生育年龄的关系。
   输出格式：(Year，Education_Level_Code, Average_Age_of_Mother)
3. 按年份分组查询，查询平均生育年龄与婴儿平均体重的关系。
   输出格式：(Year，Average_Age_of_Mother， Average_Birth_Weight)

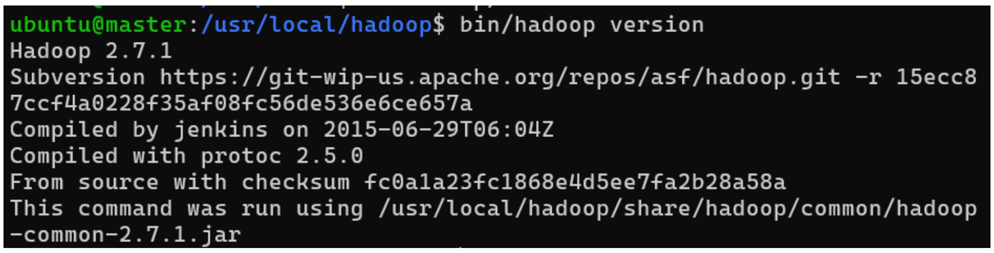
实验步骤：

1. 执行准备工作：
   Sudo apt-get update; sudo apt-get upgrade
   Sudo apt-get install openssh-server vim
   Cd .ssh; ssh-keygen -t rsa -C “2814388011@qq.com”
2. 安装 java
   Sudo apt-get install openjdk-8-jdk openjdk-8-jre
3. 为 java 设置环境变量
   Sudp vim ~/.bashrc
   在第一行添加
   export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
   保存并退出后应用设置
   Source ~/.bashrc
   验证 java 安装成功



```
ubuntu@master:~$ echo $JAVA_HOME
/usr/lib/jvm/java-8-openjdk-amd64
ubuntu@master:~$ java -version
openjdk version "1.8.0_432"
OpenJDK Runtime Environment (build 1.8.0_432-8u432-ga~us1-0ubuntu2~20.0
4-ga)
OpenJDK 64-Bit Server VM (build 25.432-bga, mixed mode)
```

4. 安装 hadoop 和 spark，可以到官网下载，或者将本地下载好的安装包通过 scp 命令远程传输到相应主机。
5. 解压到指定目录,修改权限和名称，方便后续访问
   Sudo tar -zxf Hadoop-2.7.1.tar.gz -C /usr/local
   Sudo tar -zxf spark-2.1.0-bin-with-hive.tgz -C /usr/local
   Cd /usr/local
   Sudo chown -R ubuntu:root ./spark-2.1.0-bin-h27hive
   Sudo chown -R ubuntu:root ./Hadoop-2.7.1
   Mv spark-2.1.0-bin-h27hive spark
   Mv Hadoop-2.7.1 hadoop
   验证 hadoop 安装成功


```
ubuntu@master:/usr/local/hadoop$ bin/hadoop version
Hadoop 2.7.1
Subversion https://git-wip-us.apache.org/repos/asf/hadoop.git -r 15ecc8
7ccf4a0228f35af08fc56de536e6ce657a
Compiled by jenkins on 2015-06-29T06:04Z
Compiled with protoc 2.5.0
From source with checksum fc0a1a23fc1868e4d5ee7fa2b28a58a
This command was run using /usr/local/hadoop/share/hadoop/common/hadoop
-common-2.7.1.jar
```

6. 配置 spark 环境并验证安装

   Cd ../spark/conf

   Cp spark-env.sh.template spark-env.sh

   Sudo vim spark-env.sh

   第一行添加

   export SPARK_DIST_CLASSPATH=$(/usr/local/hadoop/bin/hadoop classpath)

   保存并退出

   ../bin/run-example SparkPi

   运行显示出圆周率计算结果，表明 spark 安装成功
7. 配置 hadoop 集群环境

   Cd ../hadoop/etc/Hadoop/

   修改相应的 slaves,core-site.xml,hdfs-site.xml,mapred-site.xml(要先将模板拷贝一份出来),yarn-site.xml 文件。至此，配置完成。保存实例，另外创建两个该实例作为 slave01 和 slave02.
8. 连接三台主机

   三个主机，修改主机名，记录相互内网 ip，实现 ssh 免密访问

   Sudo vim /etc/hostname，分别修改为 master slave01，slave02(主机名不可更改，否则会与之前配置文件不符出错)

   Sudo vim /etc/hosts，分别添加三台机器的内网 ip [主机名]

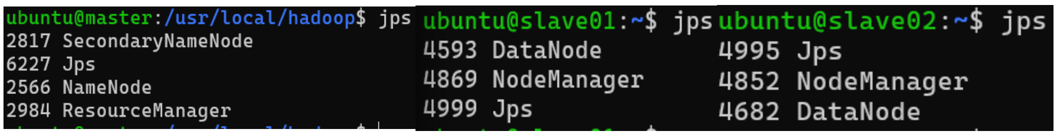
   分别重启，应用配置。重启后分别输入 ssh 相互的主机名，验证三台机器之间可以互相实现免密登录。
9. 启动 hadoop 集群

   在 master 主机中，cd /usr/local/Hadoop

   先进行格式化：bin/hdfs namenode -format

   再启动脚本：sbin/start-all.sh

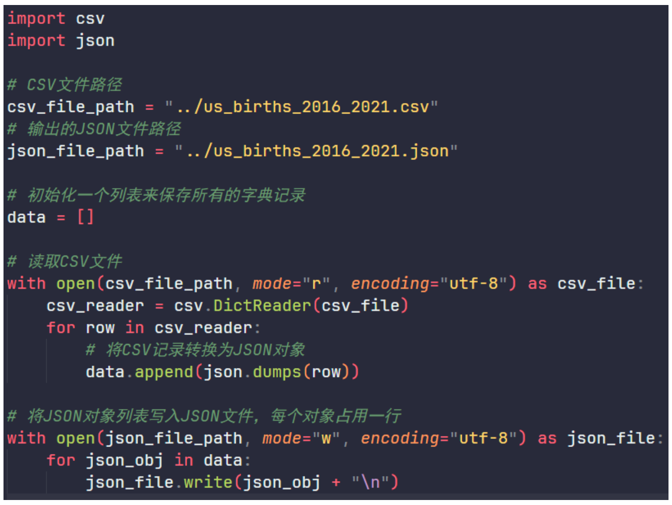
   验证 hadoop 集群是否成功启动，三台机器分别输入 jps 显示如下：


```
ubuntu@master:/usr/local/hadoop$ jps
2817 SecondaryNameNode
6227 Jps
2566 NameNode
2984 ResourceManager
```
```
ubuntu@slave01:~$ jps
4593 DataNode
4869 NodeManager
4999 Jps
```
```
ubuntu@slave02:~$ jps
4995 Jps
4852 NodeManager
4682 DataNode
```

表明 hadoop 集群成功启动。

10. 创建目录并上传数据集

    先将数据集 us_births_2016_2021.csv 转换为.json 格式文件(各个表项之间不能有,否则导入数据库会出现异常)。Csv 转 json 的 python 脚本如下：



```python
import csv
import json

# CSV文件路径
csv_file_path = "../us_births_2016_2021.csv"
# 输出的JSON文件路径
json_file_path = "../us_births_2016_2021.json"

# 初始化一个列表来保存所有的字典记录
data = []

# 读取CSV文件
with open(csv_file_path, mode="r", encoding="utf-8") as csv_file:
    csv_reader = csv.DictReader(csv_file)
    for row in csv_reader:
        # 将CSV记录转换为JSON对象
        data.append(json.dumps(row))

# 将JSON对象列表写入JSON文件，每个对象占用一行
with open(json_file_path, mode="w", encoding="utf-8") as json_file:
    for json_obj in data:
        json_file.write(json_obj + "\n")
```

再通过 scp 从本地传到 master 主机，家目录下，命名为 us.json 再上传到 hdfs 系统

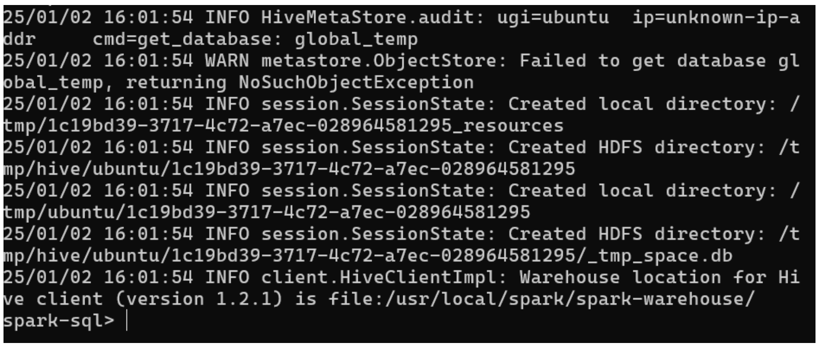
创建目录：bin/hdfs dfs -mkdir -p /user/ubuntu(目录名不能错)

上传数据集：bin/hdfs dfs -put ~/us.json

11. 启动 spark-sql 数据库

Cd ../spark

Bin/spark-sql



```
25/01/02 16:01:54 INFO HiveMetaStore.audit: ugi=ubuntu  ip=unknown-ip-a
ddr     cmd=get_database: global_temp
25/01/02 16:01:54 WARN metastore.ObjectStore: Failed to get database gl
obal_temp, returning NoSuchObjectException
25/01/02 16:01:54 INFO session.SessionState: Created local directory: /
tmp/1c19bd39-3717-4c72-a7ec-028964581295_resources
25/01/02 16:01:54 INFO session.SessionState: Created HDFS directory: /t
mp/hive/ubuntu/1c19bd39-3717-4c72-a7ec-028964581295
25/01/02 16:01:54 INFO session.SessionState: Created local directory: /
tmp/ubuntu/1c19bd39-3717-4c72-a7ec-028964581295
25/01/02 16:01:54 INFO session.SessionState: Created HDFS directory: /t
mp/hive/ubuntu/1c19bd39-3717-4c72-a7ec-028964581295/_tmp_space.db
25/01/02 16:01:54 INFO client.HiveClientImpl: Warehouse location for Hi
ve client (version 1.2.1) is file:/usr/local/spark/spark-warehouse/
spark-sql> |
```

显示如图，表明成功启动 spark-sql。

12. 导入 us.json 数据集

Create table us using org.apache.spark.sql.json options(path “us.json”)

可以查看表是否创建成功：select * from us limit 50,显示前 50 行

13. 开始 sql 查询：

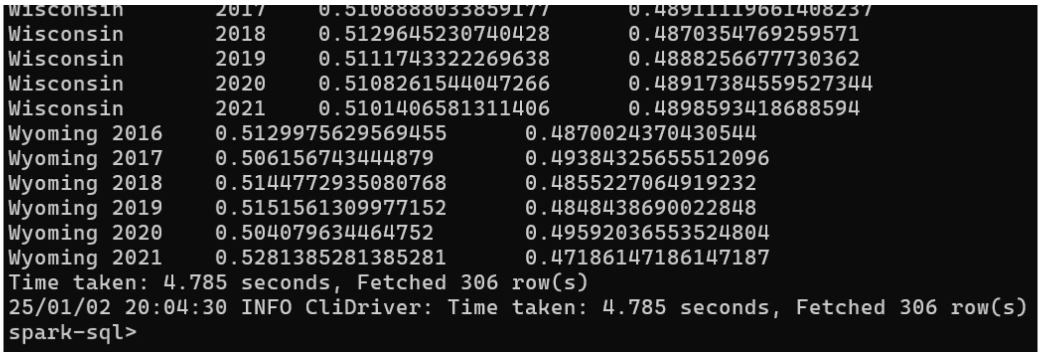
(1) 按洲和年份分组查询，出生婴儿的男女比例

输出格式：(State，Year，MRatio，FRatio)

语句如下：(标识符有空格的要用反引号``)

```
select State,
       Year,
       sum(case when Gender = "M" then `Number of Births` else 0 end) /
     sum(`Number of Births`) as MRatio,
       sum(case when Gender = "F" then `Number of Births` else 0 end) /
     sum(`Number of Births`) as FRatio
       from us group by State, Year order by State, Year;
```

结果如图：



```
Wisconsin       2017    0.5108888033859177      0.4891111966140823
Wisconsin       2018    0.5129645230740428      0.4870354769259571
Wisconsin       2019    0.5111743322269638      0.4888256677730362
Wisconsin       2020    0.5108261544047266      0.48917384559527344
Wisconsin       2021    0.5101406581311406      0.4898593418688594
Wyoming 2016    0.5129975629569455      0.4870024370430544
Wyoming 2017    0.506156743444879       0.49384325655512096
Wyoming 2018    0.5144772935080768      0.4855227064919232
Wyoming 2019    0.5151561309977152      0.4848438690022848
Wyoming 2020    0.504079634464752       0.49592036553524804
Wyoming 2021    0.5281385281385281      0.47186147186147187
Time taken: 4.785 seconds, Fetched 306 row(s)
25/01/02 20:04:30 INFO CliDriver: Time taken: 4.785 seconds, Fetched 306 row(s)
spark-sql>
```

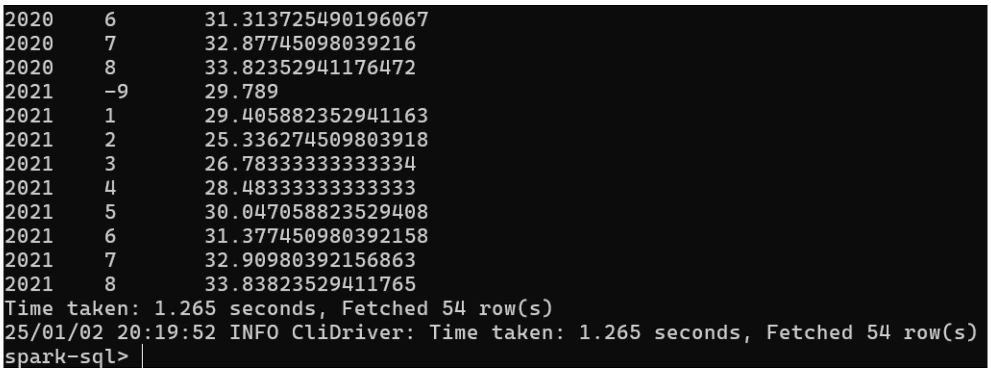
(2) 按年份分组查询，查询母亲受教育程度与平均生育年龄的关系

输出格式：(Year，Education_Level_Code, Average_Age_of_Mother)

语句如下：

```
select
Year,
`Education Level Code`
AVG(`Average Age of Mother (years)`) as Average_Age_of_Mother
from us group by Year, `Education Level Code`
order by Year, `Education Level Code`;
```

结果如图：



```
2020    6       31.313725490196067
2020    7       32.87745098039216
2020    8       33.82352941176472
2021    -9      29.789
2021    1       29.405882352941163
2021    2       25.336274509803918
2021    3       26.78333333333334
2021    4       28.48333333333333
2021    5       30.047058823529408
2021    6       31.377450980392158
2021    7       32.90980392156863
2021    8       33.83823529411765
Time taken: 1.265 seconds, Fetched 54 row(s)
25/01/02 20:19:52 INFO CliDriver: Time taken: 1.265 seconds, Fetched 54 row(s)
spark-sql> |
```

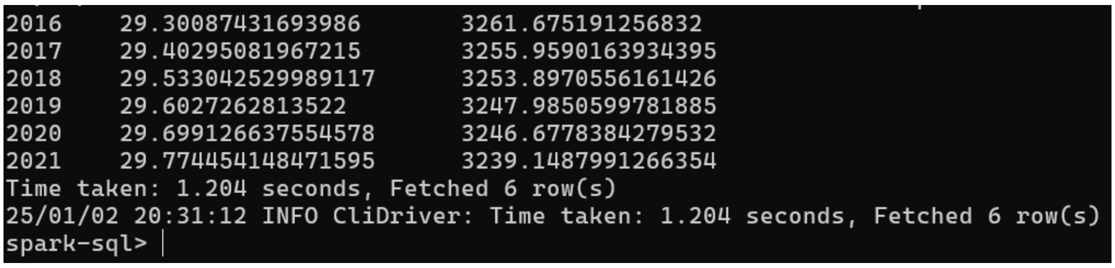
(3)按年份分组查询，查询平均生育年龄与婴儿平均体重的关系。

输出格式：(Year，Average_Age_of_Mother， Average_Birth_Weight)

语句如下：

```
select
    Year,
    AVG(`Average Age of Mother (years)`) as Average_Age_of_Mother,
    AVG(`Average Birth Weight (g)`) as Average_Birth_Weight
from us group by Year order by Year;
```

结果如图：



```
2016    29.30087431693986       3261.675191256832
2017    29.40295081967215       3255.9590163934395
2018    29.533042529989117      3253.8970556161426
2019    29.6027262813522        3247.9850599781885
2020    29.699126637554578      3246.6778384279532
2021    29.774454148471595      3239.1487991266354
Time taken: 1.204 seconds, Fetched 6 row(s)
25/01/02 20:31:12 INFO CliDriver: Time taken: 1.204 seconds, Fetched 6 row(s)
spark-sql> |
```