Spark streaming

刘磊

2020 年 12 月

设置环境变量

设置环境变量 PYTHONPATH，使得 Python3 可以找到 pyspark 包。将下面的内容添加

到环境变量配置文件~/.bashrc

```
export PYTHONPATH=$SPARK_HOME/python/:$SPARK_HOME/python/lib/py4j-0.10.7-src.zip:$PYTHONPATH
```

本次实验将把代码写为脚本运行，所以需要将 PYSPARK_DRIVER_PYTHON 的值由

jupyter 改为/usr/bin/python3，将 PYSPARK_DRIVER_PYTHON_OPTS 注释掉。

修改以后的内容如下图所示

```
# Pyspark
# export PYSPARK_DRIVER_PYTHON=jupyter
export PYSPARK_DRIVER_PYTHON=/usr/bin/python3
# export PYSPARK_DRIVER_PYTHON_OPTS='notebook'
export PYSPARK_PYTHON=python3
export PYTHONPATH=$SPARK_HOME/python/:$SPARK_HOME/python/lib/py4j-0.10.7-src.zip:$PYTHONPATH
```

使环境变量生效

```
source ~/.bashrc
```

测试

在终端中运行 python3，导入 pyspark。

```
python3
from pyspark import SparkContext
```

```
lei@ubuntu:~$ python3
Python 3.6.9 (default, Oct  8 2020, 12:12:24)
[GCC 8.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from pyspark import SparkContext
```

没有报错说明设置成功。

文件流

监控一个目录，若有新文件，Spark 就开始处理这个文件。在 Pyspark 工作目录中创建目录 streaming，并在其中创建目录 logfile。

```
cd ~/pyspark-workspace
mkdir streaming
cd streaming
mkdir logfile
```

进入 logfile 目录，创建文件 FileStreaming.py，写入下面的代码。

```
from pyspark import SparkContext, SparkConf
from pyspark.streaming import StreamingContext

conf = SparkConf()
conf.setAppName('TestDStream')
conf.setMaster('local[2]')
sc = SparkContext(conf = conf)
ssc = StreamingContext(sc, 10)
lines = ssc.textFileStream('file:///home/lei/pyspark-
workspace/streaming/logfile')
words = lines.flatMap(lambda line: line.split(' '))
wordCounts = words.map(lambda x : (x,1)).reduceByKey(lambda a,b:a+b)
wordCounts.pprint()
ssc.start()
ssc.awaitTermination()
```

提交任务

```
spark-submit FileStreaming.py
```

在~/pyspark-workspace/streaming/logfile 目录下新建一个文件 log.txt，写入一些内容保存，就可以在监听窗口中显示词频统计结果

```
-------------------------------------------
Time: 2020-12-01 19:46:50
-------------------------------------------
('hadoop', 1)
('hello', 2)
('spark', 1)
```

套接字流

监听一个端口，如果端口收到数据 spark 就进行处理。在

~/pyspark-workspace/streaming/logfile 目录下新建文件 NetworkWordCount.py，

写入以下代码

```
from pyspark import SparkContext
from pyspark.streaming import StreamingContext

sc = SparkContext("local[2]", appName="NetworkWordCount")
ssc = StreamingContext(sc, 1)
lines = ssc.socketTextStream("localhost", 9999)
words = lines.flatMap(lambda line: line.split(" "))
wordCounts = words.map(lambda x:(x, 1)).reduceByKey(lambda a,b:a+b)
wordCounts.pprint()
ssc.start()
ssc.awaitTermination()
```

新打开一个终端作为 nc 窗口，启动 nc 程序

```
nc -lk 9999
```

从此刻开始，在这个终端中输入的内容将被发送到 9999 端口。例如

```
lei@ubuntu:~/pyspark-workspace/streaming/logfile$ nc -lk 9999
red red green red blue
```

再另一个终端中执行如下代码启动流计算

```
spark-submit NetworkWordCount.py
```

该脚本将读取本地计算机端口 9999 接收发送到该套接字的任何内容。流计算终端将显示单词

的统计结果。

```
-------------------------------------------
Time: 2020-12-01 23:16:07
-------------------------------------------
('green', 1)
('red', 3)
('blue', 1)
```

再在 nc 窗口输入一些词

```
red green blue
```

流计算终端将显示单词的统计结果为

```
-------------------------------------------
Time: 2020-12-01 23:16:22
-------------------------------------------
('green', 1)
('red', 1)
('blue', 1)
```

可以看到两次输入的数据是分别统计的，并没有聚合。

全局聚合

在~/pyspark-workspace/streaming/logfile 目录下新建文件

StatefulStreamingWordCount.py，写入以下代码

```
from pyspark import SparkContext
from pyspark.streaming import StreamingContext

sc = SparkContext("local[2]", "StatefulNetworkWordCount")
ssc = StreamingContext(sc, 1)

# Create checkpoint for local StreamingContext
ssc.checkpoint("checkpoint")

# Define updateFunc: sum of the (key, value) pairs
def updateFunc(new_values, last_sum):
    return sum(new_values) + (last_sum or 0)

lines = ssc.socketTextStream("localhost", 9999)

# Calculate running counts
#   Line 1: Split lines in to words
#   Line 2: count each word in each batch
#   Line 3: Run `updateStateByKey` to running count
running_counts = lines.flatMap(lambda line: line.split(" "))\
                  .map(lambda word: (word, 1))\
                  .updateStateByKey(updateFunc)
running_counts.pprint()
```

```
ssc.start()
ssc.awaitTermination()
```

用上节同样的方法运行并发送数据，查看结果有什么不同。