Spark Streaming

设置环境变量

修改~/.bashrc　注释掉jupyter，notebook。再指定PYSPARK_DRIVER_PYTHON和PYTHONPATH

```
# pyspark
# export PYSPARK_DRIVER_PYTHON=jupyter
export PYSPARK_DRIVER_PYTHON=/usr/bin/python3
# export PYSPARK_DRIVER_PYTHON_OPTS='notebook'
export PYSPARK_PYTHON=python3
# spark streaming
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

```
chen@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
>>>
```

文件流

监控一个目录，若有新文件，Spark就会处理这个文件。但是如果修改文件内容，Spark是不会发现的。

设置监控目录为

```
/home/chen/pyspark-workspace/streaming/logfile
```

在logfile目录中创建文件FileStreaming.py

```
#!/usr/bin/env python
# coding=utf-8
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/chen/pyspark-
workspace/streaming/logfile') # 目录提前存在
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
```

在监控目录下创建文件test2.txt，写入一些内容。

保存退出后，就会在另一个终端显示

```
Time: 2020-12-01 19:58:30
-------------------------------------------
('this', 1)
('is', 1)
('test', 1)
('sentence.', 1)
('a', 1)
('second', 1)

-------------------------------------------
Time: 2020-12-01 19:58:40
-------------------------------------------
```

套接字流

监听一个端口，如果端口收到数据，spark就会处理。

1. 首先打开9999端口等待传输数据

```
nc -lk 9999
```

2. 在监控目录下新建 文件NetworkWordCount.py

```
#!/usr/bin/env python
# coding=utf-8
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()
```

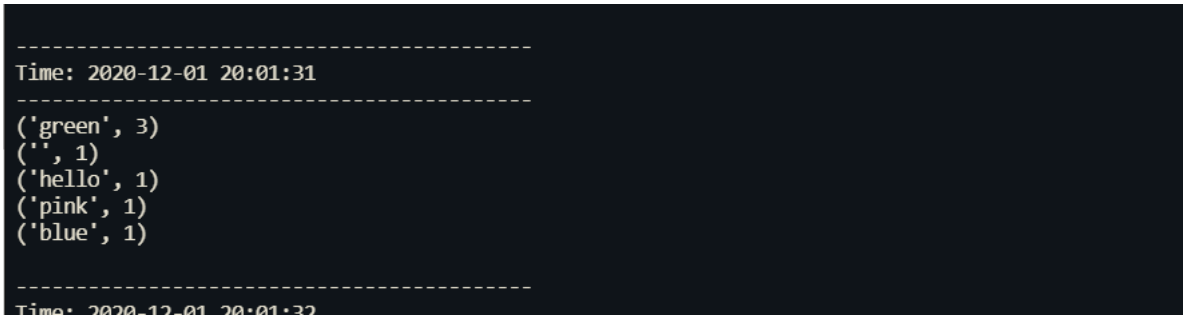
3. 提交

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

4. 写入一些数据

```
chen@ubuntu:~$ nc -lk 9999
green hello pink green green blue
```

5. 另一个端口显示一次的处理结果



全局聚合

为了使实时处理得到累加的效果，做以下处理：

1. 首先打开9999端口等待传输数据

```
nc -lk 9999
```

2. 在监控目录下新建 文件 StatefulStreamingWordCount.py

```
#!/usr/bin/env python
# coding=utf-8
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)).up
running_counts.pprint()
ssc.start()
ssc.awaitTermination()
```

3. 提交

```
spark-submit StatefulStreamingWordCount.py
```

4. 写入一些数据

5. 在另一个终端可以看到累加的效果

```
------------------------------------------
('green', 2)
('red', 2)

------------------------------------------
Time: 2020-12-01 21:07:19
------------------------------------------
('green', 2)
('red', 2)

------------------------------------------
Time: 2020-12-01 21:07:20
------------------------------------------
('green', 2)
('red', 2)

------------------------------------------
Time: 2020-12-01 21:07:21
------------------------------------------
('green', 2)
('red', 2)
```