SL-01 8.7 查询的管理和监控

8.7.1 管理和监控的方法

8.7.2 一个监控的实例



SL-02 8.7.1 管理和监控的方法



- 查询时返回的StreamingQuery()对象可用于对查询进行管理和监控。
- 在8.2节的实例中，返回的query对象就是StreamingQuery()，

1. 这个对象包含recentProgress、lastProgress、status等多个属性，
2. 其中，recentProgress可用于返回最近的多个StreamingQueryProgresss构成的数组，返回的数组大小可以使用 spark.sql.streaming.numRecentProgressUpdates来定义；
3. lastProgress可返回最近的Streaming-QueryProgress；
4. status可返回当前查询的状态。

SL-03 8.7.1 管理和监控的方法



- 可以在单个SparkSession中启动任意数量的查询，它们都会共享相同的集群资源并且并行运行。
- 此外，可以使用8.2节的实例中返回的Spark的stream()函数，来获取StreamingQueryManager对象，用于管理当前活动的查询。如果在一个程序内启动了多个查询，则不能使用单个query的awaitTermination()来让程序阻塞直到查询结束，因为这会使得一个查询结束后，程序立刻退出，导致其他查询无法继续，

  此时应当结合使用StreamingQueryManager的

  awaitAnyTermination()函数、

  active属性和resetTerminated()函数，

  直到所有查询运行完成才退出程序。
- 其中，awaitAnyTermination()会在任意一个查询结束后停止阻塞；active会返回当前上下文所有还在活动的查询；只要还有活动的查询存在，就可以使用resetTerminated()忽略已经结束的查询，并使awaitAnyTermination()可以继续用于阻塞程序。

SL-04 8.7.2 一个监控的实例



```
#!/usr/bin/env python3

from pprint import pprint
import time

from pyspark.sql import SparkSession
from pyspark.sql.functions import split
from pyspark.sql.functions import explode
```

SL-05 8.7.2 一个监控的实例



```
if __name__=='__main__':
    spark=SparkSession  \
        .builder    \
        .appName('StructuredNetworkWordCount')  \
        .getOrCreate()

    spark.sparkContext.setLogLevel('WARN')

    lines=spark \
        .readStream \
        .format('socket')   \
        .option('host','localhost') \
        .option('port',9999)    \
        .load()
```

SL-06 8.7.2 一个监控的实例



```
words=lines.select(
    explode(
        split(lines.value," ")
    ).alias('word')
)

wordCounts=words.groupBy('word').count()

query=wordCounts    \
    .writeStream    \
    .outputMode('complete') \
    .format('console')  \
    .queryName('write_to_console')  \
    .trigger(processingTime='8 seconds')    \
    .start()
```

SL-07 8.7.2 一个监控的实例



```
while True:
    if query.lastProgress:
        if query.lastProgress['numInputRows']>0:
            pprint(query.lastProgress)

        pprint(query.status)

        time.sleep(5)
```

SL-08 8.7.2 一个监控的实例

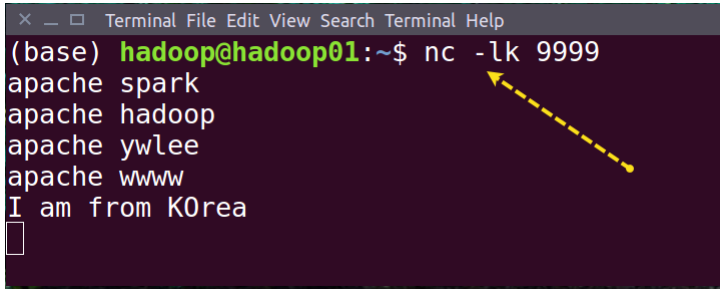


在执行StructuredNetworkWordCountWithMonitor.py之前，需要启动HDFS，启动HDFS的命令如下：

```
$ cd /usr/local/hadoop
$ sbin/start-dfs.sh
```

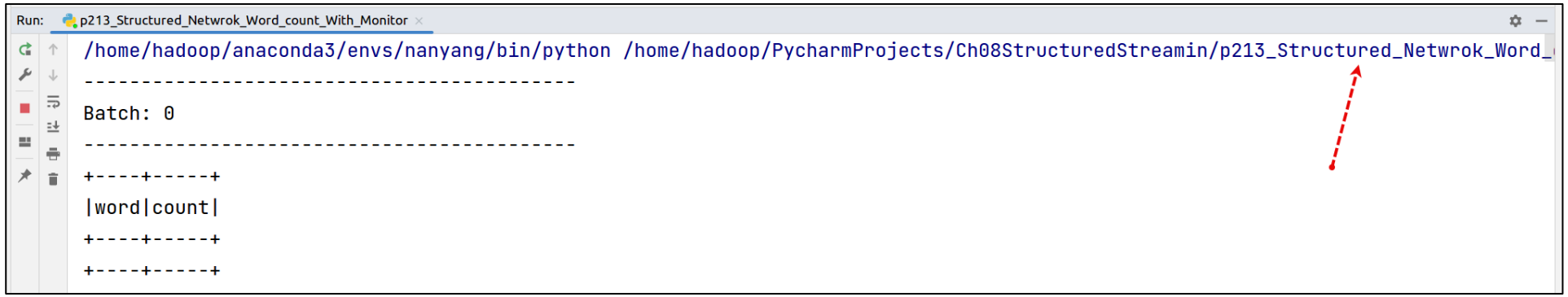
新建一个终端（记作“数据源终端”），输入如下命令：

```
$ nc -lk 9999
```



再新建一个终端（记作“流计算终端”），执行如下命令：

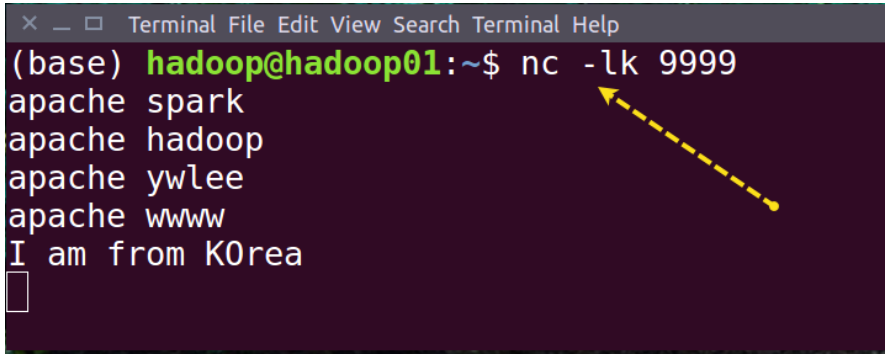
```
$ cd /usr/local/spark/mycode/structuredstreaming/monitor
$ /usr/local/spark/bin/spark-submit StructuredNetworkWordCountWithMonitor.py
```



SL-09 8.7.2 一个监控的实例



为了模拟文本数据流，可以在数据源终端内不断输入一行行英文语句，nc程序会把这些数据发送到StructuredNetworkWordCountWithMonitor.py程序进行处理，比如，输入如下数据：



SL-10 8.7.2 一个监控的实例



在“流计算终端”内观察程序执行结果，可以看出，与StructuredNetworkWordCount.py相比，在执行StructuredNetworkWordCountWithMonitor.py以后的输出结果中，新增了类似如下的状态信息：

```
/home/hadoop/anaconda3/envs/nanyang/bin/python /home/ha
-------------------------------------------
Batch: 0
-------------------------------------------
+----+-----+
|word|count|
+----+-----+
+----+-----+

{'isDataAvailable': True,
 'isTriggerActive': True,
 'message': 'Processing new data'}
{'isDataAvailable': True,
 'isTriggerActive': True,
 'message': 'Processing new data'}
{'isDataAvailable': True,
 'isTriggerActive': True,
 'message': 'Processing new data'}
```

```
-------------------------------------------
Batch: 1
-------------------------------------------
+------+-----+
|  word|count|
+------+-----+
| ywlee|    1|
|apache|    3|
| spark|    1|
|hadoop|    1|
+------+-----+

{'batchId': 1,
 'durationMs': {'addBatch': 13312,
                'commitOffsets': 66,
                'getBatch': 0,
                'latestOffset': 0,
                'queryPlanning': 79,
                'triggerExecution': 13523,
                'walCommit': 60},
 'id': '2826ef31-ab5d-4de3-98ca-4668b44ba26c',
 'inputRowsPerSecond': 0.07525398218989088,
 'name': 'write_to_console',
 'numInputRows': 3,
 'processedRowsPerSecond': 0.22184426532574134,
 'runId': 'c3540d94-4156-49df-b7b4-b9733b65f3f6',
 'sink': {'description': 'org.apache.spark.sql.execution.streaming.ConsoleTable$@367c2c79',
```

SL-11 8.8 本章小结

1. 本章首先介绍了Structured Streaming的基本概念以及它与Spark SQL、Spark Streaming的区别。
2. Structured Streaming与Spark Streaming相似，可以用来处理数据流，并且整合了Spark SQL的DataFrame/Dataset来处理结构化的数据流。
3. 然后，通过一个实例演示了编写Structured Streaming程序的基本步骤，并介绍了输入源、输出模式和接收器。Spark的设计可以保证端到端的完全一致性，所以本章重点介绍了Spark如何处理容错，并通过一个实例来观察迟到数据处理的详细结果。
4. 本章最后介绍了查询的管理和监控方法。