8.2 编写Structured Streaming程序的基本步骤

编写Structured Streaming程序的基本步骤包括:

① 导入pyspark模块

② 创建SparkSession对象

③ 创建输入数据源

④ 定义流计算过程

⑤ 启动流计算并输出结果

SL -02 8.2 编写Structured Streaming程序的基本步骤



实例任务

一个包含很多行英文语句的数据流源源不断到达，

Structured Streaming程序对每行英文语句进行拆分，

并统计每个单词出现的频率

SL-03 8.2 编写Structured Streaming程序的基本步骤



1.步骤1：导入pyspark模块

导入PySpark模块，代码如下

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

由于程序中需要用到拆分字符串和展开数组内的所有单词的功能，所以引用了来自pyspark.sql.functions里面的split和explode函数。

SL-04　8.2 编写Structured Streaming程序的基本步骤



2.步骤2：创建SparkSession对象

创建一个SparkSession对象，代码如下

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

 spark.sparkContext.setLogLevel('WARN')
```

SL-05 8.2 编写Structured Streaming程序的基本步骤



3．步骤3：创建输入数据源

创建一个输入数据源

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

SL-06 8.2 编写Structured Streaming程序的基本步骤



4.步骤4：定义流计算过程

定义相关的查询语句

```
words = lines.select(
        explode(
            split(lines.value, " ")
        ).alias("word")
    )
    wordCounts = words.groupBy("word").count()
```

SL-07 8.2 编写Structured Streaming程序的基本步骤



5.步骤5：启动流计算并输出结果

执行流计算

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

    query.awaitTermination()
```

SL-09 8.2 编写Structured Streaming程序的基本步骤



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/
$ /usr/local/spark/bin/spark-submit StructuredNetworkWordCount.py
```

8.2 编写Structured Streaming程序的基本步骤



为了模拟文本数据流，可以在“数据源终端”内用键盘不断敲入一行行英文语句，nc程序会把这些数据发送给StructuredNetworkWordCount.py程序进行处理，比如输入如下数据

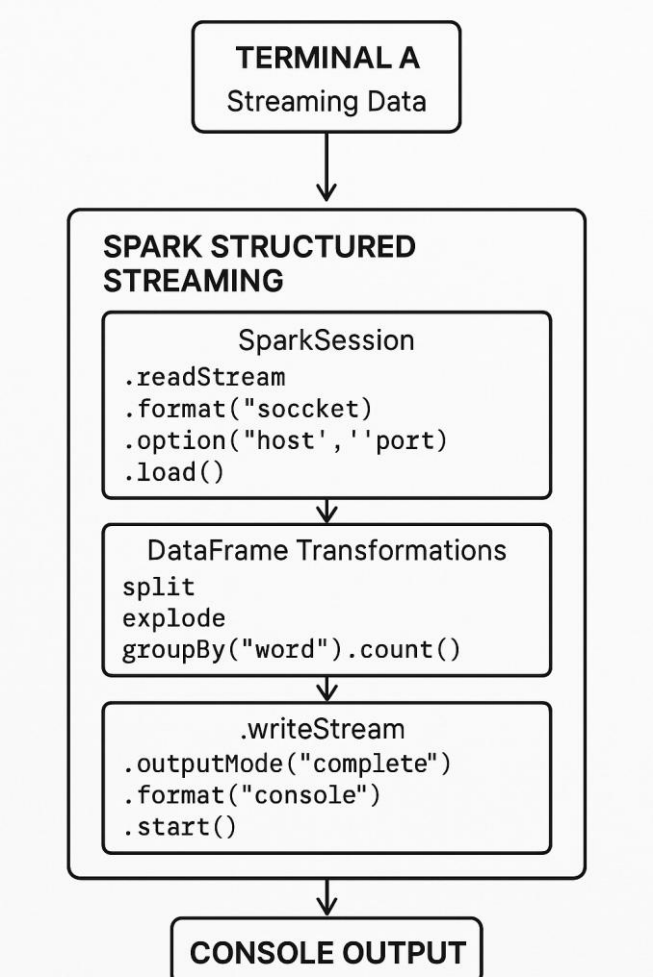
```
apache spark
apache hadoop
```

则在“流计算终端”窗口内会输出类似以下的结果信息

```
-------------------------------------------
Batch: 0
-------------------------------------------
+------+-----+
|  word|count|
+------+-----+
|apache|    1|
| spark|    1|
+------+-----+
```

```
-------------------------------------------
Batch: 1
-------------------------------------------
+------+-----+
|  word|count|
+------+-----+
|apache|    2|
| spark|    1|
|hadoop|    1|
+------+-----+
```

SL-08 8.2 编写Structured Streaming程序的基本步骤



nc –lk 9999

```
(base) hadoop@hadoop01:/usr/bin$ nc -lk 9999
apache spark
apache hadoop
ywlee korea professor
hello every one my name is Yangweon
i study apache pyspark
zookeeper is good
number is the sequence
yangweon is a
what is batch
good good good
```

```
+----+-----+
|word|count|
+----+-----+
+----+-----+

-------------------------------------------
Batch: 1
-------------------------------------------
+---------+-----+
|     word|count|
+---------+-----+
|    korea|    1|
|professor|    1|
|    ywlee|    1|
|   apache|    2|
|    spark|    1|
|   hadoop|    1|
+---------+-----+
```

8.2 编写Structured Streaming程序的基本步骤



```
# 步骤1. 导入pyspark模块
from pyspark.sql import SparkSession
from pyspark.sql.functions import split
from pyspark.sql.functions import explode

# 步骤2. 创建SparkSession对象
if __name__ == '__main__':
    spark = SparkSession \
        .builder \
        .appName("StructuredNetworkWordCount") \
        .getOrCreate()

    print(__name__)
    spark.sparkContext.setLogLevel('WARN')
```

8.2 编写Structured Streaming程序的基本步骤



```
# 步骤3. 创建输入数据源
    lines = spark \
        .readStream \
        .format('socket') \
        .option('host', 'localhost') \
        .option('port', 9999) \
        .load()

# 步骤4： 定义流计算过程
    words = lines.select(
       explode(
           split(lines.value, " ")
      ).alias("word")
    )
    wordCounts = words.groupBy("word").count()
```

8.2 编写Structured Streaming程序的基本步骤



```
# 步骤5： 启动流计算并输出结果
    query=wordCounts \
        .writeStream \
        .outputMode('complete') \
        .format('console')  \
        .trigger(processingTime='8 seconds') \
        .start()
    query.awaitTermination()

spark.stop()
```