SL-01 슬라이드 1: 8.2 编程实践 - 基本步骤

好，理论部分我们就先讲到这里。现在开始实践环节！

编写 Structured Streaming 程序的基本步骤包括：

1. **导入pyspark模块 (Import pyspark modules)**
2. **创建SparkSession对象 (Create SparkSession object)**
3. **创建输入数据源 (Create input data source)**
4. **定义流计算过程 (Define stream computation process)**
5. **启动流计算并输出结果 (Start stream computation and output results)**

这五个步骤是标准流程，每个 Structured Streaming 程序都要经过这些步骤。就像做菜一样，有固定的步骤，但可以调整具体的配料和做法。

等一下我们会通过一个具体的例子，一步一步地演示每个步骤。

SL02 슬라이드 2: 实例任务介绍

现在我们来看一个具体的任务：

任务描述： 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。

这是一个经典的**词频统计 (Word Count)** 任务！

虽然看起来简单，但它展示了流处理的核心概念：

- **实时数据接收**
- **数据转换处理**
- **聚合计算**
- **结果输出**

这个例子会让大家对流处理有直观的理解。

我们会看到数据如何实时到达，程序如何实时处理，结果如何实时更新。

就像新闻直播一样，每当有新的语句到达，词频统计结果就会立即更新！

SL-03 슬라이드 3: 步骤1 - 导入模块

好，我们开始第一步：**导入pyspark模块**。

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

让我解释一下每一行：

第一行：from pyspark.sql import SparkSession

- 这是创建 Spark 应用程序的入口点

第二行：from pyspark.sql.functions import split

- split 函数用于拆分字符串，比如把 "hello world" 拆分成 ["hello", "world"]

第三行：from pyspark.sql.functions import explode

- explode 函数用于展开数组(배열)内的所有元素，把一行变成多行
- 比如把 int numbers[5] = {1, 2, 3, 4, 5};

为什么需要这两个函数？

因为我们要把英文语句拆分成单词，然后统计每个单词的频率。

比如 "apache spark" 这一行，我们要：

1. 用 split 拆分成 ["apache", "spark"]
2. 用 explode 展开成两行：一行是 "apache"，一行是 "spark"

SL-04 슬라이드 4: 步骤2 - 创建SparkSession

第二步：**创建SparkSession对象**。

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

让我们分解这段代码：(**fēn jiè zhè duàn dài mǎ)**

第一行：if __name__ == "__main__":

- 这是Python的标准写法，确保代码只在直接运行时执行

SparkSession 创建：

- SparkSession.builder - 使用建造者模式 (Builder Pattern)
- .appName("StructuredNetworkWordCount") - 给应用起个名字
- .getOrCreate() - 获取现有session或创建新的

最后一行：spark.sparkContext.setLogLevel('WARN')

- 设置日志级别为WARN，这样可以减少输出的信息，让结果更清晰

这就像开车前要先发动引擎一样，SparkSession 就是我们程序的"引擎"。

发动(fā dòng) :개시하다, 시작하다 (운동, 작전 등을)

SL-05 슬라이드 5: 步骤3 - 创建输入数据源

第三步：**创建输入数据源**。

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

这段代码做了什么？

.readStream - 这告诉Spark我们要读取流数据，不是静态文件

.format("socket") - 数据源类型是socket连接

.option("host", "localhost") - 主机地址是本机(**zhǔ jī dì zhǐ shì běn jī)**

.option("port", 9999) - 端口号是9999

.load() - 加载数据源

简单来说，我们的程序会**监听本机的9999端口**，

等待有人向这个端口发送文本数据。

这就像在家里等快递 —— 我们告诉快递员地址（localhost）和门牌号（9999），

然后等着接收包裹 **bāo guǒ**（数据）。

等会儿我们会用 nc 命令来模拟发送数据到这个端口。

- **等**会儿 (děng huì er) - 잠시 후에
- **命令** (mìng lìng) - 명령어
- 来 (lái) - ~하기 위해
- **模拟** (mó nǐ) - 시뮬레이션하다
- **发送** (fā sòng) - 전송하다
- **端口** (duān kǒu) - 포트

SL-06 슬라이드 6: 步骤4 - 定义流计算过程

第四步：**定义流计算过程**。这是最核心的部分！

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

让我们一行一行分析：

第一部分 - 数据转换：(部分 (bù fēn))

- lines.value - 获取接收到的文本内容
- split(lines.value, " ") - 按空格拆分成单词数组
- explode(...) - 把数组展开成多行
- .alias("word") - 给列起个名字叫"word"

第二部分 - 聚合计算： jù hé jì suàn

- words.groupBy("word") - 按单词分组
- .count() - 计算每组的数量

举个例子：

- 输入："apache spark"
- split后：["apache", "spark"]
- explode后：两行数据，一行是"apache"，一行是"spark"
- groupBy后：按单词分组
- count后：apache=1次，spark=1次

这就是我们的数据处理流水线（data processing pipeline）！

SL-07 슬라이드 7: 步骤5 - 启动流计算

最后一步：**启动流计算**并**输出结果**。

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

让我解释每个参数：

.writeStream - 写入流，开始输出结果

.outputMode("complete") - 输出模式是"完整"，每次输出所有结果

*append：只输出新增的行，不会更新或删除已输出的行(새 결과만 추가)

*update：只输出自上次触发以来发生变更的行(新增、更新)(변경된 결과만 출력)

.format("console") - 输出到控制台，方便我们看结果

.trigger(processingTime="8 seconds") - 每8秒触发一次处理

.start() - 启动流处理

query.awaitTermination() - 等待流处理结束（通常是手动停止）

输出模式有三种：

- **complete** - 每次输出完整结果表
- **append** - 只输出新增的行
- **update** - 只输出有更新的行

我们选择 complete 模式，这样每次都能看到完整的词频统计结果。

8秒的触发间隔意味着每8秒更新一次结果，这样我们有足够时间输入测试数据。

- **间隔** (jiàn gé) - 간격, 인터벌
- **这样** (zhè yàng) - 이렇게, 그러면
- **时间** (shí jiān) – 시간
- **结果** (jié guǒ) - 결과
- **足够** (zú gòu) - 충분한
- **测试数据** (cè shì shù jù) - 테스트 데이터

SL-08 슬라이드 8: 程序运行准备

现在我们把所有代码写入文件 StructuredNetworkWordCount.py。

运行前的准备工作：

1. 启动HDFS： (jps 명령어로 동작 확인 :QuorumPeerMain은 zookeeper)

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

为什么要启动HDFS？

因为Structured Streaming需要checkpoint目录来保存状态信息。

2. 开启数据源终端：

```
$ nc -lk 9999
```

nc 是 netcat 命令，-l 表示监听，-k 表示保持连接。

这个命令会在9999端口等待连接，我们可以在这里输入文本数据。

3. 开启流计算终端：

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

现在我们有三个终端：

- **HDFS终端** - 运行Hadoop
- **数据源终端** - 用nc发送数据
- **[PyCharm출력화면]流计算终端** - 运行Spark程序

就像三个演员在同一个舞台上合作演出！

- 端口 (duān kǒu)
- 舞台表演 (wǔ tái biǎo yǎn) - 무대 공연

SL-09 슬라이드 9: 运行结果演示

最后我们来看运行效果！

在数据源终端输入：

```
apache spark
apache hadoop
```

在流计算终端看到输出：[처음에 에러는 무시하고 밑에 __main__ 나오면 터미널에서 입력 후에 대기함

Batch: 0

```
+------+-----+
| word|count|
+------+-----+
|apache|    1|
| spark|    1|
+------+-----+
```

Batch: 1

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

大家看到了吗？

第一批处理：输入"apache spark"，得到apache=1，spark=1 **第二批处理**：输入"apache hadoop"，apache的计数变成2（累计），hadoop新出现计数为1

这就是**实时流处理**的效果！每当我们输入新数据，词频统计就会实时更新。

注意几个关键点：

1. **累计统计** - apache出现了2次
2. **实时更新** - 每8秒更新一次结果
3. **完整输出** - 每次显示所有单词的统计

这就是我们今天学习的 Structured Streaming 的威力！

```
# 실행 방법: 터미널에서
# nc -lk 9999 을 입력후에 다음 내용을 입력
# apache spark
```

```
# apache hadoop

# 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')

# 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()

# 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()
```

总结

好的，今天的课程到这里就结束了。让我们回顾一下学到的核心内容：

1. **基本概念**：数据流 = 无界表
2. 两**种处理模型**：微批处理（100ms级）和持续处理（ms级）
3. **编程五步骤**：导入→创建session→数据源→计算逻辑→启动输出
4. **实践案例**：词频统计的完整实现

课后作业：请大家尝试修改触发时间间隔，观察对结果的影响。下节课我们将学习更多的输入源和输出格式。

如果有任何问题，记得通过邮件联系我。谢谢大家！