实验四 Spark Streaming

零、服务器集群&Spark注意事项

1.可使用的服务器集群

从lab2开始，我们上线了新的服务器集群，目前有两个集群(两个集群不互通，大家可以选择随意选择在集群一或二上实验^ ^)：

集群一：ip地址：`10.103.9.11`，可用机器：01，03-06。登录集群一01的命令：`ssh xxx@10.103.9.11` （也就是和以下实验指导书的内容完全相同）

集群二：ip地址：`10.103.10.156`，可用机器：01-04。登录集群二01的命令：`ssh xxx@10.103.10.156 -p 8001`（**由于进入该集群的端口并非默认端口，所以在 `ssh` 指令后面一定要用 `-p` 要加上端口号！**）

其中 xxx为学号，默认密码也为学号。

推荐学号尾号为奇数的同学使用集群一，学号尾号为偶数的同学使用集群二，以减轻服务器集群的负担。

2.实验中的注意事项

很多同学在使用完 `spark-shell` 后直接关闭窗口或 `ctrl+z` 挂起，未正常退出，导致spark进程仍在运行，使得服务器资源不足。这就会导致其他同学在进行实验时出现以下情况：

```
scala> textFile.first()
[Stage 0:>          (0 + 0) / 1]23/08/13 22:09:04 WARN TaskSchedulerImpl:
Initial job has not accepted any resources; check your cluster UI to ensure
that workers are registered and have sufficient resources
```

所以大家在正常完成实验之后不要直接关闭进程窗口，**请务必输入 `:quit` 从而正常退出 `spark` 进程。**

此外，请大家合理安排实验时间，在临近deadline时也很可能由于做实验的人较多导致集群资源不足。

一、实验目标

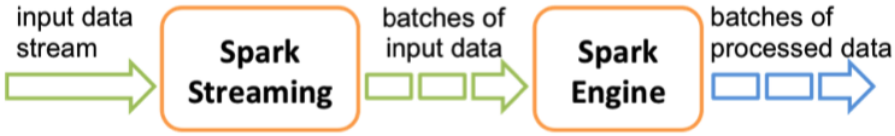
本次实验旨在帮助理解实时数据流处理方法。具体如下：

- 学习Spark Streaming分布式处理框架，理解流式数据处理概念；
- 实现简单流式数据产生、接收与处理。

二、Spark Streaming概述

Spark Streaming 是核心 Spark API 的扩展，可实现可扩展、高吞吐量、可容错的实时数据流处理。数据可以从诸如 Kafka，Flume，Kinesis 或 TCP 套接字等众多来源获取，并且可以使用由高级函数（如 map，reduce，join 和 window）开发的复杂算法进行流数据处理。最后，处理后的数据可以被推送到文件系统，数据库和实时仪表板。而且，还可以在数据流上应用 Spark 提供的机器学习和图处理算法。

在内部，它的工作原理如图所示。Spark Streaming 接收实时输入数据流，并将数据切分成批，然后由 Spark 引擎对其进行处理，最后生成“批”形式的结果流。Spark Streaming 将连续的数据流抽象为 discretized stream 或 DStream。可以从诸如 Kafka，Flume 和 Kinesis 等来源的输入数据流中创建 DStream，或者通过对其他 DStream 应用高级操作来创建。在内部，DStream 由一个 RDD 序列表示。



三、流式数据产生、接收与处理

1. 使用 netcat 指令产生测试数据流（2分）

netcat 是网络工具中的瑞士军刀，它能通过 TCP 和 UDP 在网络中读写数据。通过 与其他工具重定向，可以实现很多复杂的功能，如端口扫描、流式数据生成或保存、文件传 输等。本次实验主要用 netcat 生成流式数据。

首先，我们需要开启两个终端，一个用于生成流式数据，另一个用于接收流式数据。在 两个窗口中分别执行下面的指令（把 11009 改成学号的后四位或五位，避免端口冲突）

```
# terminal_1
thumm01:~$ nc localhost 11009

# terminal_2
thumm01:~$ nc -lk 11009
```

接着输入字符串，这个字符串会被这一端的 netcat 以流数据的形式发送到 对应的端口，并在另一端被 netcat 接收并输出。这是一个最简单的聊天软件:)

```
# terminal_1
thumm01:~$ nc localhost 11009
Hello World

# terminal_2
thumm01:~$ nc -lk 11009
Hello World
```

此外，netcat 还可以与重定向符号”>”结合直接将文件数据转成网络流发送到另一端

```
# terminal_1
thumm01:~$ vim test.txt
thumm01:~$ cat test.txt
spark streaming test
thumm01:~$ nc localhost 11009 < test.txt

# terminal_2
thumm01:~$ nc -lk 11009
spark streaming test
```

如果右边改成 nc -lk 11009 > test_recvd.txt 就可以把接受的流保存成文件，完成了一次文件传输

2. 使用 Spark Streaming 接收流数据（2分）

前面使用了 netcat 接收流数据并打印，接下来使用 SparkStream 接收流数据，为后 续流数据处理做准备。首先使用 netcat 交互地生成流式数据

```
thumm01:~$ nc -lk 11009
```

然后在另一个终端启动 spark-shell

```
thumm01:~$ spark-shell
//spark-shell 启 动 信 息

// 导 入 相 关 SparkStreaming 包
scala> import org.apache.spark.streaming.{Seconds, StreamingContext}
import org.apache.spark.streaming.{Seconds, StreamingContext}

// 创 建 流 式
scala> val ssc = new StreamingContext(sc, Seconds(1))
ssc: org.apache.spark.streaming.StreamingContext =
org.apache.spark.streaming.StreamingContext@2114955c

// 监听 thumm01 的 11009 端口
scala> val lines = ssc.socketTextStream("thumm01", 11009)
lines: org.apache.spark.streaming.dstream.ReceiverInputDStream[String] =
org.apache.spark.streaming.dstream.SocketInputDStream@105fa381

scala> lines.print() // 打 印 接 收 到 的 信 息
scala> ssc.start() // 启 动 流 数 据 接 受 与 处 理
... // 每 隔 一 段 时 间 刷 新 一 次 ， 如 果 接 收 到 信 息 就 输 出 ， 没 有 接 受 就 只 输 出 时 间
scala> ssc.stop () // 停 止 流 数 据 接 收
```

在 netcat 端输入 “hello world”, 回车，在 spark-shell 端会出现下面的输出，表示接收到信息:

```
scala> -------------------------------------------
Time: 1669966519000 ms
-------------------------------------------
hello world

[Stage 2:>                                                          (0 + 1) / 1]
```

Spark Streaming 需要输入 ssc.stop() 停止流数据的接收，而 ctrl + c 不会让程序退出， 这点需要注意。从上面的流程，我们可以看到 Spark Streaming 处理流式数据的过程是先定义处理流程，然后启动任务。

四、使用Spark Streaming做词频统计

0. 词频统计要求

以下要求对下述两个任务都适用。

输出格式

为了方便进行正确性的验证，请将你词频统计的输出结果保证按照ASCII字典序从小到大的格式进行输出。每行一个按照 `<word> <count>` 的形式输出，代表该词汇出现的次数。

以 `wc_dataset.txt` 为例，你需要保证输出结果为：

```
'tis 5
Again 5
Alabama 15
Alleghenies 5
Almighty 5
America 25
American 20
And 60
But 20
California 5
Carolina 5
Catholics 5
Colorado 5
Constitution 5
Continue 5
Declaration 5
Emancipation 5
Five 5
Free 5
From 10
Gentiles 5
Georgia 15
Go 5
God 5
God's 15
Hampshire 5
Happiness 5
I 75
In 10
Independence 5
Instead 5
It 20
Jews 5
Land 5
Let 45
Liberty 5
Life 5
Lookout 5
Lord 5
Louisiana 5
Mississippi 20
Mountain 10
My 5
Negro 65
```

```
Negro's 5
New 15
Nineteen 5
No 5
Now 25
One 15
Pennsylvania 5
Pilgrim's 5
Proclamation 5
Protestants 5
Rights 5
Rockies 5
Some 5
South 10
Stone 5
Tennessee 5
Thank 5
The 10
There 5
This 25
We 50
When 10
With 15
York 10
You 5
a 63875
abide 100
able 140
... // 后续词汇较多，予以省略
```

提交报告时附带中间过程文件

你需要在你提交的实验报告压缩包当中提供你运行词频统计期间，中间过程文件夹的截图，并提取出最终结果文件夹的具体文件。最终结果应包括以下内容（其中文件夹命名为原始实验记录，不需要与此处相同）：

```
output
|-- _SUCCESS
|-- part-00000
|-- part-00001
|-- ...
```

1. 使用 map+reduceByKey 做词频统计（3分）

当我们能接收到流数据以后，下一步就是对流数据进行处理，这里还是做最经典的词频统计任务。在一个终端启动 spark-shell，输入下面指令

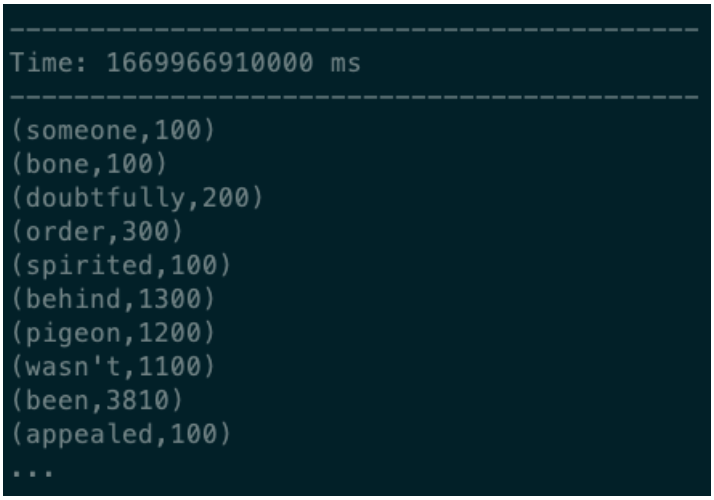
```
import org.apache.spark.streaming._ // 导入数据包
val ssc = new StreamingContext(sc, Seconds(5)) // 创建流数据上下文，每隔5秒创建一个流式RDD
val lines = ssc.socketTextStream("thumm01", 11009) // 监听 thumm01 的端口
val result = lines.flatMap(_.split(" " )).map(w => (w, 1)).reduceByKey(_ + _) // 词频统计
result.print() // 输出词频统计结果
ssc.start() // 启动任务 //
ssc.stop() 最后退出的时候输入
```

在指令执行过程中，程序会输出很多错误，这个可以不用管，因为生成流数据的那端还没开。

在另一个终端产生数据，这里使用 netcat 将 wc_dataset.txt 作为数据源生成流：

```
thumm01:~$ nc -l 11009 < /home/dsjxtjc/wc_dataset.txt
```

这个时候，Spark Streaming 端就会显示词频统计的结果：



这个时候我们就完成了统计 5 秒内词频的功能， 对这个程序做一定的修改， 就能实现统计累计词频，并按照单词的字典序从小到大输出的功能。

2. 更换RDD函数组合做词频统计（3分）

上一题使用的是map+reduceByKey来实现词频统计，请换用另一种RDD函数组合实现（可以参考实验三），对比词频统计结果。

四、报告提交要求

请严格按照以下要求提交实验报告。

1. 将命令、关键代码（文本）、结果截图放入报告，实验报告需为pdf 格式，连同代码文件一同打包成压缩文件（命名为`学号_姓名_实验一.*`，例如：`2021200000_张三_实验一.zip`），最后提交到网络学堂。压缩文件中文件目录应为：

```
.
└── 学号_姓名_实验二.pdf
└── code
    └── code_file1.py
    └── code_file2.py
    └── ...
```

2. 迟交作业一周以内，以50% 比例计分；一周以上不再计分。另一经发现抄袭情况（包括往届），零分处理。