RT4 · Big Data Class Labs

Lab 2 · Batching and Streaming with Spark

Adam Lahbib · M. Sofiene Barka · Mohamed Rafraf

Apr. 20, 2023 @ INSAT

Contents

1 Introduction

In this lab we will be using Spark to perform batch and streaming processing. We will be using the Spark Shell to perform batch processing and the Spark Streaming API to perform streaming processing.

2 Definitions

2.1 Batch Processing

Batch processing is the execution of a series of programs in a single job. The programs are executed in a specific order, and the output of one program is used as the input to the next program. Batch processing is typically used for large amounts of data that are processed infrequently. Batch processing is also known as offline processing.

2.2 Streaming Processing

Streaming processing is the execution of a series of programs in a single job. The programs are executed in a specific order, and the output of one program is used as the input to the next program. Streaming processing is typically used for large amounts of data that are processed infrequently. Streaming processing is also known as offline processing.

2.3 Spark

Spark is a unified analytics engine for large-scale data processing. It provides high-level APIs in Java, Scala, Python, and R, and an optimized engine that supports general execution graphs. It also supports a rich set of higher-level tools including Spark SQL for SQL and structured data processing, MLlib for machine learning, GraphX for graph processing, and Spark Streaming.

3 Lab Walkthrough

3.1 Starting the containers

```
~ ❯  docker run -itd --net=hadoop -p 50070:50070 -p 8088:8088 -p 7077:7077 -p 16010:16010 \
            --name hadoop-master --hostname hadoop-master \
            liliasfaxi/spark-hadoop:hv-2.7.2
b623f5aee695f023e6044c1f07023670f1ca88da8faae09032fa5fd0176ca07b
~ ❯ docker run -itd -p 8040:8042 --net=hadoop \
        --name hadoop-slave1 --hostname hadoop-slave1 \
              liliasfaxi/spark-hadoop:hv-2.7.2
41d62ed45dabb699af5ea2ca11519dd6a0f43dcd184589d3d453754ece14bb11
~ ❯  docker run -itd -p 8041:8042 --net=hadoop \
        --name hadoop-slave2 --hostname hadoop-slave2 \
              liliasfaxi/spark-hadoop:hv-2.7.2
1b91c156a3475cb86a6a3c288a6caa8d6d22eee7e5522f10aa83fc89d32c3b18
```

3.2 Starting Hadoop & YARN

```
~ ❯ docker exec -it hadoop-master bash
root@hadoop-master:~# ./start-hadoop.sh


Starting namenodes on [hadoop-master]
hadoop-master: Warning: Permanently added 'hadoop-master,172.24.0.2' (ECDSA) to the list of known hosts.
hadoop-master: starting namenode, logging to /usr/local/hadoop/logs/hadoop-root-namenode-hadoop-master.out
hadoop-slave2: Warning: Permanently added 'hadoop-slave2,172.24.0.4' (ECDSA) to the list of known hosts.
hadoop-slave1: Warning: Permanently added 'hadoop-slave1,172.24.0.3' (ECDSA) to the list of known hosts.
hadoop-slave2: starting datanode, logging to /usr/local/hadoop/logs/hadoop-root-datanode-hadoop-slave2.out
hadoop-slave1: starting datanode, logging to /usr/local/hadoop/logs/hadoop-root-datanode-hadoop-slave1.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: Warning: Permanently added '0.0.0.0' (ECDSA) to the list of known hosts.
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-root-secondarynamenode-hadoop-master.out


starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn--resourcemanager-hadoop-master.out
hadoop-slave2: Warning: Permanently added 'hadoop-slave2,172.24.0.4' (ECDSA) to the list of known hosts.
hadoop-slave1: Warning: Permanently added 'hadoop-slave1,172.24.0.3' (ECDSA) to the list of known hosts.
hadoop-slave2: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-root-nodemanager-hadoop-slave2.out
hadoop-slave1: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-root-nodemanager-hadoop-slave1.out
```

3.3 Checking Daemons on Main Node

```
root@hadoop-master:~# jps
155 NameNode
348 SecondaryNameNode
508 ResourceManager
1055 Jps
root@hadoop-master:~#
```

3.4 Checking Daemons on both Worker Nodes

```
~ ❯ docker exec -it hadoop-slave1 bash                                    took 1h 5m 3s  1.42  4.55G at 03:42  59%
root@hadoop-slave1:~# jps
169 NodeManager
475 Jps
63 DataNode
root@hadoop-slave1:~# exit
~ ❯ docker exec -it hadoop-slave2 bash                                    took 4s  1.41  4.55G at 03:43  59%
root@hadoop-slave2:~# jps
168 NodeManager
62 DataNode
415 Jps
root@hadoop-slave2:~#
```

3.5 Testing Spark with Spark Shell

3.5.1 Touching a file and putting it on HDFS

```
root@hadoop-master:~# vi file1.txt
root@hadoop-master:~# cat file1.txt
Hello Spark Wordcount!
Hello Hadoop Also :)
root@hadoop-master:~# hadoop fs -put file1.txt
root@hadoop-master:~#
```

3.5.2 Spark Shell from Main Node

```
root@hadoop-master:~# spark-shell
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
23/04/14 14:48:36 WARN yarn.Client: Neither spark.yarn.jars nor spark.yarn.archive is set, falling back to uploading libraries under SPARK_HO
ME.
23/04/14 14:51:17 WARN metastore.ObjectStore: Failed to get database global_temp, returning NoSuchObjectException
Spark context Web UI available at http://172.24.0.2:4040
Spark context available as 'sc' (master = yarn, app id = application_1681479486268_0002).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.2.0
      /_/

Using Scala version 2.11.8 (OpenJDK 64-Bit Server VM, Java 1.8.0_191)
Type in expressions to have them evaluated.
Type :help for more information.

scala> Display all 646 possibilities? (y or n)

scala>
```

3.5.3 Running a Scala Script

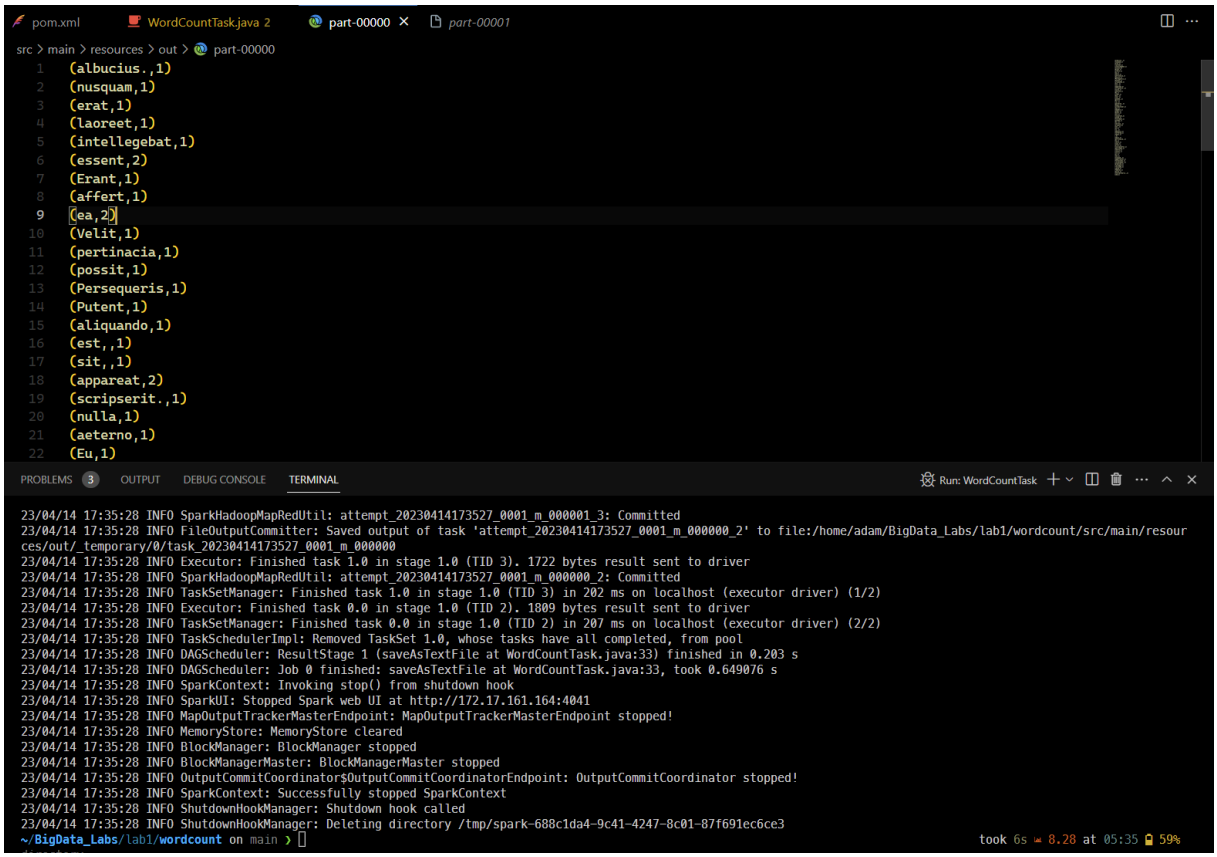
Output:

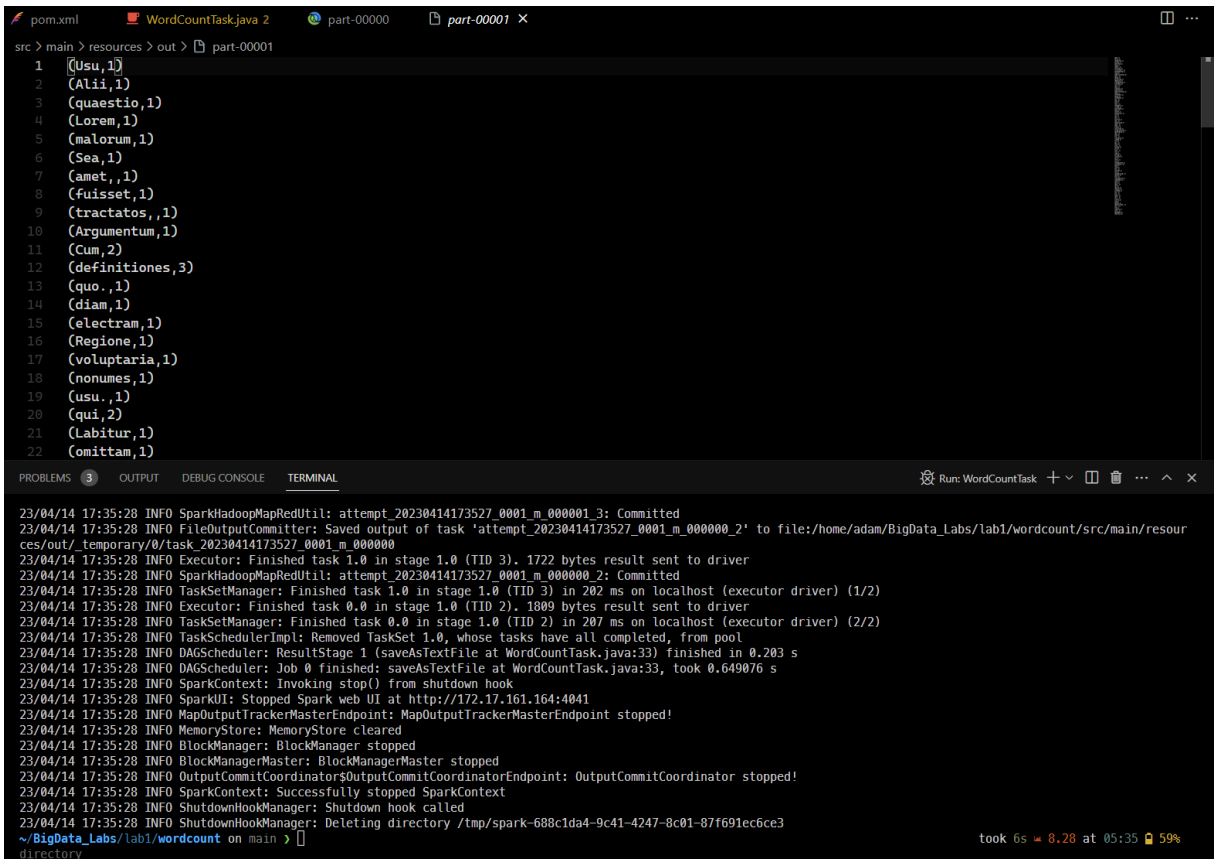
```
scala>

scala> val lines = sc.textFile("file1.txt")
lines: org.apache.spark.rdd.RDD[String] = file1.txt MapPartitionsRDD[1] at textFile at <console>:24

scala> val words = lines.flatMap(_.split("\\s+"))
words: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at flatMap at <console>:26

scala> val wc = words.map(w => (w, 1)).reduceByKey(_ + _)
wc: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[4] at reduceByKey at <console>:28

scala> wc.saveAsTextFile("file1.count")

scala> 23/04/14 16:08:31 WARN netty.Dispatcher: Message RemoteProcessDisconnected(172.24.0.4:43910) dropped. RpcEnv already stopped.
23/04/14 16:08:31 WARN netty.Dispatcher: Message RemoteProcessDisconnected(172.24.0.4:43910) dropped. RpcEnv already stopped.
root@hadoop-master:~# hadoop fs -get file1.count
root@hadoop-master:~# cd file1.count/
root@hadoop-master:~/file1.count# cat part-0000
cat: part-0000: No such file or directory
root@hadoop-master:~/file1.count# cat part-00000
(Hello,2)
(Wordcount!,1)
root@hadoop-master:~/file1.count# cat part-00001
(Spark,1)
(:),1)
(Also,1)
(Hadoop,1)
root@hadoop-master:~/file1.count#
```

The yielded directory file1.count is the result of the word count operation performed on the file1.txt file on each two of the worker nodes.

3.6 Spark Batch in JAVA

3.6.1 Local





3.6.2 Cluster

In local mode

This mode is for debugging and testing. It runs the Spark driver program in the same process as the Spark shell. The driver program is responsible for running the Spark application's main function, distributing the application's code and data to the Spark executors, and monitoring their execution. The driver program also runs the SparkContext, which is the entry point to Spark functionality. The driver program runs on the local machine, and the executors run on the worker nodes.

```
root@hadoop-master:~# hadoop fs -tail output/part-00000
.2,87)
(375.38,73)
(291.13,65)
(446.42,77)
(452.49,81)
(253.31,87)
(178.68,66)
(172.95,89)
(424.55,93)
(105.64,94)
(114.97,68)
(Arlington,40348)
(24.97,79)
(195.09,90)
(22.02,73)
(469.7,88)
(186.23,83)
(384.3,81)
(64.6,90)
(449.9,83)
(466.39,63)
(2012-04-10,11215)
(408.73,68)
(138.75,85)
(28.66,89)
(33.24,86)
(153.29,68)
(239.24,88)
(155.43,85)
(173.56,65)
(427.43,98)
(203.89,84)
```

```
root@hadoop-master:~# hadoop fs -tail output/part-00001
7)
(138.69,89)
(257.3,75)
(481.51,77)
(427.86,72)
(478.53,80)
(319.84,88)
(236.4,87)
(56.64,74)
(171.44,79)
(345.27,83)
(496.97,86)
(266.09,89)
(303.98,93)
(33.94,71)
(333.86,67)
(65.95,75)
(268.1,94)
(131.11,74)
(118.76,111)
(458.22,83)
(113.4,77)
(398.76,80)
(61.02,74)
(348.08,77)
(100.93,82)
(261.02,82)
(308.37,99)
(53.9,78)
(61.31,76)
(162.59,71)
(71.83,83)
(483.4,92)
(144.17,88)
(11.29,88)
```

In YARN mode

This mode is for production use. It runs the Spark driver program on one of the worker nodes in the cluster. The driver program is responsible for running the Spark application's main function, distributing the application's code and data to the Spark executors, and monitoring their execution. The driver program also runs the SparkContext, which is the entry point to Spark functionality. The driver program runs on the local machine, and the executors run on the worker nodes.

```
root@hadoop-master:~# hadoop fs -tail output2/part-00000
.2,87)
(375.38,73)
(291.13,65)
(446.42,77)
(452.49,81)
(253.31,87)
(178.68,66)
(172.95,89)
(424.55,93)
(105.64,94)
(114.97,68)
(Arlington,40348)
(24.97,79)
(195.09,90)
(22.02,73)
(469.7,88)
(186.23,83)
(384.3,81)
(64.6,90)
(449.9,83)
(466.39,63)
(408.73,68)
(2012-04-10,11215)
(138.75,85)
(28.66,89)
(33.24,86)
(153.29,68)
(239.24,88)
(155.43,85)
(173.56,65)
(427.43,98)
(203.89,84)
(120.7,76)
(318.02,71)
(458.43,84)
```

```
root@hadoop-master:~# hadoop fs -tail output2/part-00001
7)
(138.69,89)
(257.3,75)
(481.51,77)
(427.86,72)
(478.53,80)
(319.84,88)
(236.4,87)
(56.64,74)
(171.44,79)
(345.27,83)
(496.97,86)
(266.09,89)
(303.98,93)
(33.94,71)
(333.86,67)
(65.95,75)
(268.1,94)
(131.11,74)
(118.76,111)
(458.22,83)
(113.4,77)
(398.76,80)
(61.02,74)
(348.08,77)
(100.93,82)
(261.02,82)
(308.37,99)
(53.9,78)
(61.31,76)
(162.59,71)
(71.83,83)
(483.4,92)
(144.17,88)
(11.29,88)
```

3.7 Spark Streaming

Spark Streaming API is used to process streaming data. It is built on top of Spark Core and Spark SQL. Spark Streaming API provides a high-level abstraction called DStream (Discretized Stream) that represents a continuous stream of data. DStream is a sequence of RDDs (Resilient Distributed Datasets). RDD is a collection of elements that are partitioned across the nodes of the cluster that can be operated in parallel. Spark Streaming API provides a set of operations that can be performed on DStream. Spark

Streaming API can be used to process data from various sources such as Kafka, Flume, Kinesis, and TCP sockets.

3.7.1 Locally

```
23/04/14 21:09:59 INFO TaskSetManager: Finished task 0.0 in stage 248.0 (TID 246) in 9 ms on localhost (executor driver) (3/3)
23/04/14 21:09:59 INFO TaskSchedulerImpl: Removed TaskSet 248.0, whose tasks have all completed, from pool
23/04/14 21:09:59 INFO DAGScheduler: ResultStage 248 (print at Stream.java:31) finished in 0.007 s
23/04/14 21:09:59 INFO BlockManagerInfo: Removed broadcast_105_piece0 on 172.17.161.164:35833 in memory (size: 2.1 KB, free: 837.6 MB)
23/04/14 21:09:59 INFO BlockManagerInfo: Removed broadcast_109_piece0 on 172.17.161.164:35833 in memory (size: 2.1 KB, free: 837.6 MB)
23/04/14 21:09:59 INFO ContextCleaner: Cleaned shuffle 52
23/04/14 21:09:59 INFO DAGScheduler: Job 124 finished: print at Stream.java:31, took 0.035963 s
-------------------------------------------
Time: 1681502999000 ms
-------------------------------------------
(world!,1)
(Hello,1)

23/04/14 21:09:59 INFO JobScheduler: Finished job streaming job 1681502999000 ms.0 from job set of time 1681502999000 ms
23/04/14 21:09:59 INFO JobScheduler: Total delay: 0.058 s for time 1681502999000 ms (execution: 0.052 s)
23/04/14 21:09:59 INFO ShuffledRDD: Removing RDD 244 from persistence list
23/04/14 21:09:59 INFO BlockManagerInfo: Removed broadcast_112_piece0 on 172.17.161.164:35833 in memory (size: 2.1 KB, free: 837.6 MB)
23/04/14 21:09:59 INFO BlockManager: Removing RDD 244
23/04/14 21:09:59 INFO MapPartitionsRDD: Removing RDD 243 from persistence list
23/04/14 21:09:59 INFO BlockManager: Removing RDD 243
23/04/14 21:09:59 INFO ContextCleaner: Cleaned shuffle 56
```

```
~ ❯ nc -lk 9999                                                    ✘ INT took 39m 52s  2.37  2
Hello world!
```

3.7.2 Cluster

We've typed `hello world!` as well!

```
23/04/14 21:10:06 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 0 ms
23/04/14 21:10:06 INFO ShuffleBlockFetcherIterator: Getting 0 non-empty blocks out of 0 blocks
23/04/14 21:10:06 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 0 ms
23/04/14 21:10:06 INFO Executor: Finished task 1.0 in stage 276.0 (TID 275). 1091 bytes result sent to driver
23/04/14 21:10:06 INFO Executor: Finished task 2.0 in stage 276.0 (TID 276). 1091 bytes result sent to driver
23/04/14 21:10:06 INFO TaskSetManager: Finished task 0.0 in stage 276.0 (TID 274) in 4 ms on localhost (executor driver) (
23/04/14 21:10:06 INFO TaskSetManager: Finished task 2.0 in stage 276.0 (TID 276) in 5 ms on localhost (executor driver) (
23/04/14 21:10:06 INFO TaskSetManager: Finished task 1.0 in stage 276.0 (TID 275) in 5 ms on localhost (executor driver) (
23/04/14 21:10:06 INFO TaskSchedulerImpl: Removed TaskSet 276.0, whose tasks have all completed, from pool
23/04/14 21:10:06 INFO DAGScheduler: ResultStage 276 (print at Stream.java:31) finished in 0.003 s
23/04/14 21:10:06 INFO DAGScheduler: Job 138 finished: print at Stream.java:31, took 0.010922 s
-------------------------------------------
Time: 1681503006000 ms
-------------------------------------------
(world!,1)
(Hello,1)

23/04/14 21:10:06 INFO JobScheduler: Finished job streaming job 1681503006000 ms.0 from job set of time 1681503006000 ms
23/04/14 21:10:06 INFO JobScheduler: Total delay: 0.030 s for time 1681503006000 ms (execution: 0.025 s)
23/04/14 21:10:06 INFO ShuffledRDD: Removing RDD 272 from persistence list
23/04/14 21:10:06 INFO BlockManager: Removing RDD 272
23/04/14 21:10:06 INFO MapPartitionsRDD: Removing RDD 271 from persistence list
23/04/14 21:10:06 INFO BlockManager: Removing RDD 271
23/04/14 21:10:06 INFO MapPartitionsRDD: Removing RDD 270 from persistence list
```

4 Conclusion

Code Repository: https://github.com/adamlahbib/BigData_Labs