Spark Streaming TP

Réalisé par :
Raoui Zainab- Hmamed Abdelmounaime - Charid Basma -
Agbalou Reda- Islah Zineb - Ait Brik Sara

Objective

The objective of this exercise sheet is to help you understand and apply Spark Streaming, a powerful real-time processing framework in the Apache Spark ecosystem. By the end of these exercises, you should be able to:

- Comprehend the fundamentals of Spark Streaming.
- Perform basic Spark Streaming operations.
- Work with more complex Spark Streaming tasks.

Here you will find some basic concepts and some ready-to-use configuration to help you start.

TP Toolbox

Before starting the exercises, ensure you have the following tools and environments set up:

- **Apache Spark**: Install and configure Apache Spark on your system. (version: 3.1.1)
- **Scala/Python**: Familiarity with Scala or Python programming languages. (scalaSDK: 3.1.1, Scala: 2.12)
- **Java OpenJDK 8**

Configuration Tips

We understand that the configuration process can be challenging, and therefore, we would like to offer some helpful tips

- use IntelliJ and gradle as a build automation tool
- add scala and Big data tool plugins to IntelliJ
- Start a new project with a spark template
- and you are ready to take of.

TP Content

In this TP, we will tackle three exercises with varying levels of difficulty, ranging from easy to hard.

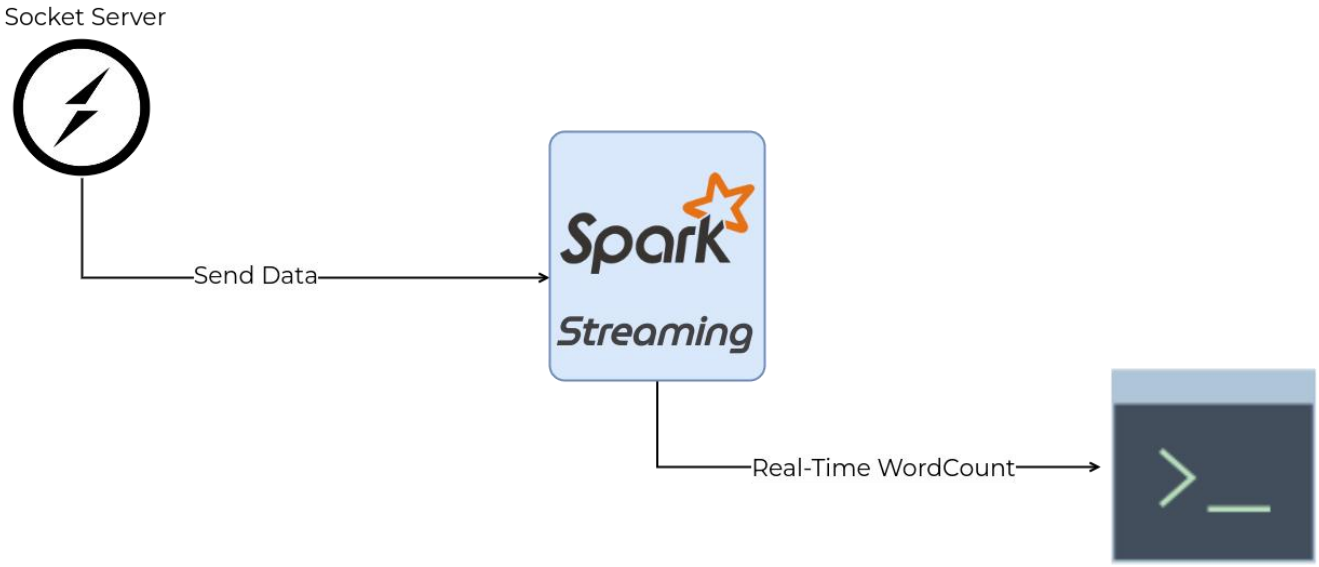
- **Exercise 1**: Gain a fundamental understanding of Spark Streaming and its core concepts.
- **Exercise 2**: Perform basic Spark Streaming operations.
- **Exercise 3**: Work on more complex Spark Streaming tasks.

Real-Time Word Count with Spark Streaming

Objective

In this example, you will learn how to use Spark Streaming to perform real-time word count on a stream of text data. Follow the steps below to create a Spark Streaming application.

Exercise overview



Exercise Step by step

0.1 Set up the Project

Create a new project directory for your Spark Streaming application.

0.2 Create a Data Source

Simulate a data source by setting up a socket server. Open a terminal and run the following command to start the server:

```
$ nc -lk 9999
```

This command starts a socket server on port 9999, which you will use to send text data to your Spark Streaming application.

If you are using Windows, check this link you will find how to start a socket server.

0.3 Create Your Spark Streaming Application

0.3.1 Scala

```scala
import org.apache.spark.SparkConf
import org.apache.spark.streaming.{Seconds, StreamingContext}

object WordCountStreaming {
  def main(args: Array[String]): Unit = {

    /* Your code goes here
        1 --> Create a SparkConf / SparkSession and set the application name
        2 --> Create a StreamingContext with a batch interval of x second
        3 --> Create a DStream that reads data from the socket source
        4 --> Split the lines into words, and count word occurrences
        5 --> Print the word count to the console
        6 --> Start the streaming context
        7 --> Wait for the streaming context to terminate
    */
  }
}
```

0.3.2 Python

```python
from pyspark import SparkConf
from pyspark.streaming import StreamingContext

def main():
    /* Your code goes here
        1 --> Create a SparkConf / SparkSession and set the application name
        2 --> Create a StreamingContext with a batch interval of x second
        3 --> Create a DStream that reads data from the socket source
        4 --> Split the lines into words, and count word occurrences
        5 --> Print the word count to the console
        6 --> Start the streaming context
        7 --> Wait for the streaming context to terminate
    */

if __name__ == "__main__":
    main()
```

0.4 Run the Spark Streaming Application

Compile and run your Spark Streaming application from your IDEA or using the spark-submit command for Scala and Python applications.

0.5 Send Data to the Socket Server

In the terminal where you started the **nc** command (step 2), you can now type or paste text data. This data will be sent to the Spark Streaming application, and you will see real-time word counts printed on the console.

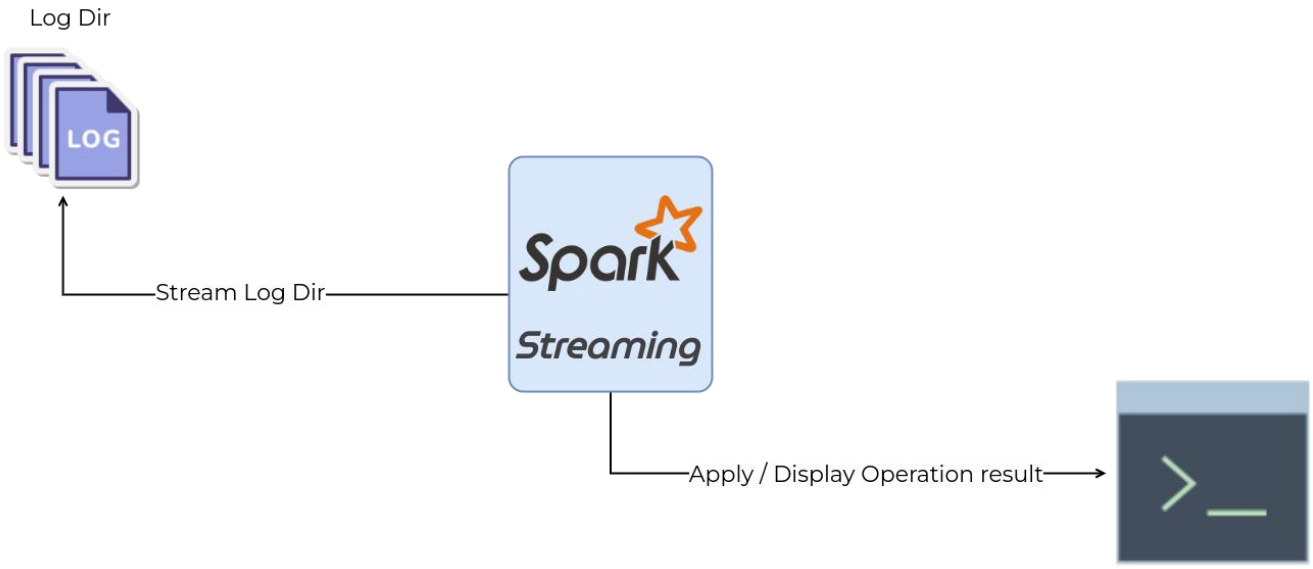
That's it! You've created a simple Spark Streaming application that counts words in real-time from a socket data source. This example illustrates the basics of Spark Streaming.

Real-Time Log File Analysis with Spark Streaming

Objective

In this example, you will learn how to use Spark Streaming for real-time analysis of log files generated by web servers. Follow the steps below to create a Spark Streaming application for log file analysis.

Exercise overview



Exercise Step by step

0.1 Set up the Project

Create a new project directory for your Spark Streaming application.

0.2 Create a Data Source

You can choose to either generate a stream of log data for this example or use a sample log file. The log data should represent web server access events.

Here you will find log files example to use.

Caution: Try copy one file at the time while the program is running to see the real-time processes.

0.3 Create Your Spark Streaming Application

0.3.1 Scala

```scala
import org.apache.spark.sql.SparkSession
import  org.apache.spark.sql.types.{StructField,  StructType}

object LogFileStreaming {
  def main(args: Array[String]): Unit = {

    /* Your code goes here
        1 --> Create a SparkConf / SparkSession and set the application name
        2 --> Create a schema based on the log files content
        3 --> Create a DStream that reads data from the logs dir
        4 --> Create a temp view on top of the that DStream
        5 --> Apply transformations (i.e extract the numbers of errors in those log file)
        6 --> Display the results in real-time
        7 --> Start the streaming context
        8 --> Wait for the streaming context to terminate
    */
  }
}
```

0.3.2 Python

```python
from pyspark import SparkConf
from pyspark.streaming import StreamingContext

def main():
    /* Your code goes here
        1 --> Create a SparkConf / SparkSession and set the application name
        2 --> Create a schema based on the log files content
        3 --> Create a DStream that reads data from the logs dir
        4 --> Create a temp view on top of the that DStream
        5 --> Apply transformations (i.e extract the numbers of errors in those log file)
        6 --> Display the results in real-time
        7 --> Start the streaming context
        8 --> Wait for the streaming context to terminate
    */

if __name__ == "__main__":
    main()
```

0.4 Run the Spark Streaming Application

Compile and run your Spark Streaming application from your IDEA or using the spark-submit command for Scala and Python applications.

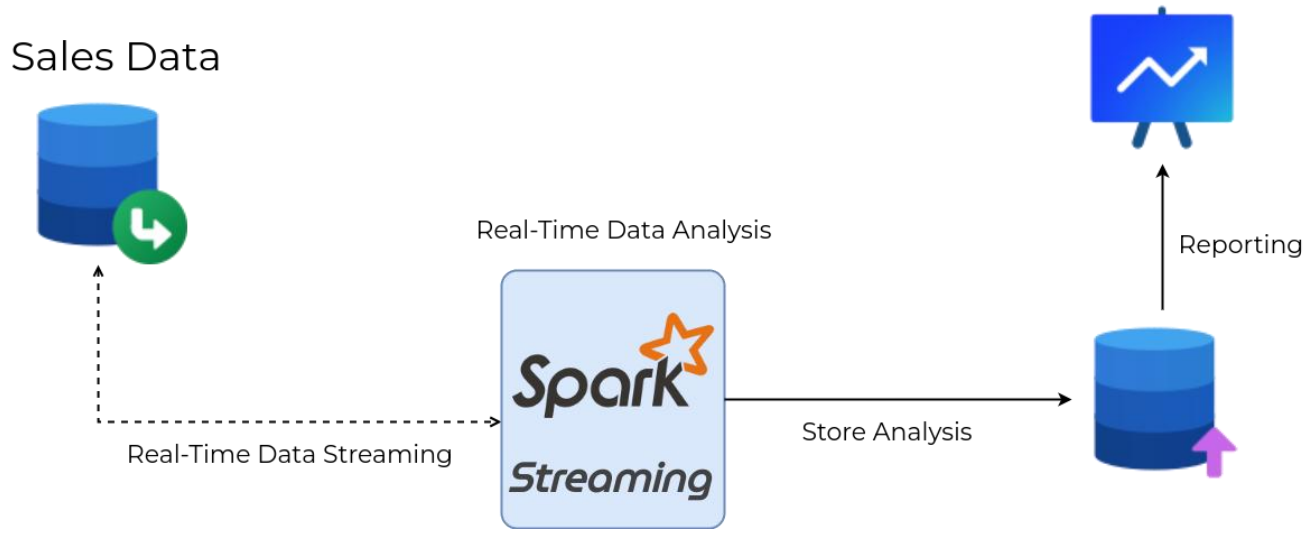
This example demonstrates how Spark Streaming can be used for real-time log file analysis, which is valuable for monitoring web server logs and generating insights in real-time.

Real-Time Sales Analysis

Objective

The objective of this project is to perform real-time sales analysis by streaming data from a MySQL database, analyzing it using Spark Streaming, and storing the results in a different database that can be connected to Power BI for visualization.

1 Exercise overview



2 Exercise Step by step

2.1 Data Source and MySQL Setup

- Download the sales data-set from here
- Install and run a MySQL Server (use docker)

```
$ docker run --name mysql-server -p 3306:3306 -e MYSQL_ROOT_PASSWORD=root -d mysql:latest
```

- Access MySQL server shell

```
$ docker exec -it mysql-server bash
$ mysql -uroot -proot
```

- In the MySQL console create a database

```
CREATE DATABASE salesDB;
```

- Connect to the salesDB from IntellJ and run the queries that you will find with the data sets to create the tables
- From the IntelliJ DataBase Panel, import the csv data each one to its corresponding table.

2.2 Spark Streaming Application

- Create a Spark Streaming application in Scala or Python.
- Set up your SparkConf and StreamingContext.
- Use a library like JDBC to connect to the MySQL database.
- Stream the sales data from the MySQL database in real-time

  Tips

  - Implement a Custom Receiver for receiving data from the concerned data source.
  - Use **receiverStream** to stream data from the database.
- Perform some analysis on our sales data to uncover valuable insights
- Reporting

2.3 Run the Spark Streaming Application

- Ensure that your Spark Streaming application runs continuously to provide real-time data updates.
- Add some rows of data in different table and see if the result after the analysis changes.

This Exercise involves setting up a data pipeline that streams data from a MySQL database, processes it in real-time with Spark Streaming.

Expand your knowledge (Optional)

- **Real-Time Data Storage**: As your Spark Streaming application processes data, store the real-time analysis results in the selected database.
- **Power BI Integration**: Set up a connection between the database where you store the real-time results and Power BI. You can use Power BI's connectors to connect to various databases.
- **Monitoring and Alerts**: Implement a monitoring system to detect and handle issues with data processing or database connections