TP n° 2 Spark

Nom prénom binôme 1

Nom prénom binôme 2 (Pas de trînome sauf groupe impaire).

I. Spark et Hadoop

Relancer les containers dockers que vous avez : docker start hadoop-master hadoop-worker1 hadoop-worker2

Aller sur le container master : docker exec -it hadoop-master bash puis lancer hadoop : ./start-hadoop.sh

Verifier que les processus background sont bien lancés avec la commande : jps.

Quels sont les processus qui tournent sur le container master ?

Faire de même avec les workers : docker exec -it hadoop-worker1 bash. Quels sont les processus qui tournent sur le worker ?

II. Premier pas avec Spark

Créer un fichier dans le container master nommé fichier1.txt avec le contenu suivant :

Salut salut Spark ! On compte les mots.

Salut salut Hadoop ! On va compter les mots.

Mettre ce fichier dans le dossier input sur le HDFS.

Pour lancer spark, il faut taper la commande : spark-shell. Quel est le langage utilisé ici ?

Faire un MapReduce pour compter les mots du fichier1.txt que vous avez crée en utilisant Scala. Vous pouvez vous appuyer sur ChatGPT pour générer ce code. Avant de l'exécuter, merci de m'appeler pour venir vérifier. (4 lignes suffissent normalement).

III. Lancer un wordcount à travers python et Spark

Fichier Python pour lancer un wordcount sur le fichier demandé via spark.

```
from pyspark.sql import SparkSession

# Initialize SparkSession
spark = SparkSession.builder \
    .appName("WordCount") \
    .getOrCreate()

# Load the text file
lines = spark.sparkContext.textFile("fichier1.txt")

# Split each line into words
words = lines.flatMap(lambda line: line.split())

# Map each word to a tuple (word, 1) and then reduce by key to count occurrences
word_counts = words.map(lambda word: (word, 1)).reduceByKey(lambda a, b: a + b)

# Save the word counts to a text file
word_counts.saveAsTextFile("fichier_count")

# Stop the SparkSession
spark.stop()
```

Pour lancer le code : spark-submit word_count.py

Récupérer les fichiers de sorties crée et vérifier la bonne exécution du wordcount. Insérer ici les screenshots des résultats.

Regarder sur http://localhost:8088/ l'état des jobs et le temps d'exécution. Commenter en comparant par rapport à celui pris par un MapReduce vu lors du dernier TP.

Comme précisé en cours, il y a deux modes pour lancer le code avec spark submit sur le cluster ou en local sur votre machine.

Quels sont les deux commandes différentes pour lancer le code en local et sur le container docker ? Tester les deux commandes. Est-ce que vous voyez un changement de vitesse d'exécution ?

IV. SparkStreaming

On va maintenant utiliser une autre fonctionnalité de Spark, celle de la gestion de flux de données. Nous allons lancer un « chat » dans notre container pour lui envoyer des messages et intercepter ces messages pour les process en temps réel avec Spark.

Dans un premier temps configurer un chat sur deux terminal en mode envoie et ecoute sur le port 9999 à l'aide de netcat.

Le code script python pour configurer hadoop-streaming :

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

# Initialisation de la session Spark
spark = SparkSession \
    .builder \
    .appName("NetworkWordCount") \
    .getOrCreate()

# Configuration de la lecture en continu à partir du socket localhost:9999
lines = spark \
    .readStream \
    .format("socket") \
    .option("host", "localhost") \
    .option("port", 9999) \
    .load()

# Diviser les lignes en mots
words = lines.select(
    explode(split(lines.value, " ")).alias("word")
)
```

```
# Compter le nombre d'occurrences de chaque mot
wordCounts = words.groupBy("word").count()

# Écrire les résultats continus sur la console, rafraîchissant chaque seconde
query = wordCounts \
    .writeStream \
    .outputMode("complete") \
    .format("console") \
    .trigger(processingTime="1 second") \
    .start()

# Attendre la terminaison du traitement en continu
query.awaitTermination()
```

Pour lancer le code : spark-submit streaming_word_count.py

Où s’execute hadoopstreaming ? Parler de l’intervalle de temps pour le traitement des données.

V. Bonus Kafka

Lancer kafka à l'aide de ./start-kafka-zookeeper.sh vérifier que le job s'exécute bien. Créer un tropic puis un producer et un consumer et envoyer des messages pour voir leurs retours.