Base de données NOSQL et Big Data

TP2 :Le traitement batch avec Hadoop Streaming

Objectifs du TP :

Ce TP a pour objectif de vous introduire à l'utilisation de Hadoop Streaming pour implémenter des tâches de traitement de données à l'aide de MapReduce en Python. Vous acquerrez les compétences nécessaires pour développer des programmes Map et Reduce en Python, ainsi que pour exécuter des jobs MapReduce avec Hadoop Streaming.

Partie 1 : Comprendre Hadoop Streaming

Hadoop Streaming est un outil qui permet d'exécuter des tâches MapReduce avec des programmes personnalisés dans des langages autres que Java, tels que Python, Perl, Ruby, etc. Cette approche simplifiée implique que Hadoop appelle un exécutable dans le cadre d'un travail, plutôt que d'utiliser un fichier JAR dans un environnement Java.

Le processus fonctionne en utilisant des flux d'entrée et de sortie entre les tâches de mappage (mapper) et de réduction (reducer) sous la forme de lignes de texte. Les programmes personnalisés lisent à partir de l'entrée standard (STDIN) et écrivent sur la sortie standard (STDOUT). Il est essentiel de comprendre comment STDIN et STDOUT sont utilisés en Python pour pouvoir utiliser l'API de flux Hadoop.

Le "mapper" est le programme qui exécute la fonction de mappage, tandis que le "reducer" est le programme qui exécute la fonction de réduction. Les lignes de sortie standard du processus de mise en correspondance sont converties en paires clé/valeur en les séparant par le premier caractère de tabulation ("\t", le caractère de séparation par défaut mais modifiable). Ces paires clé/valeur sont ensuite envoyées à l'entrée standard du réducteur pour un traitement ultérieur. Enfin, le réducteur écrit sur la sortie standard, qui est la sortie finale du programme.

Partie 2 : Exécution d'un job avec Hadoop Streaming

Dans cet exemple, nous allons voir le code du **Word Count** pour compter combien de fois chaque mot apparaît dans un ensemble de fichiers texte. Le programme se compose d'une fonction de mappage (mapper) et d'une fonction de réduction (reducer). Initialement, nous allons tester le code localement sur de petits fichiers avant de l'utiliser dans une tâche de streaming MapReduce. Il est beaucoup plus facile de déboguer les erreurs de codage sur votre ordinateur personnel que sur le cluster.

Mapper : mapperWC.py

Le programme mapper lit chaque ligne d'entrée, découpe les mots et émet un couple clé-valeur pour chaque mot, ou la clé est le mot et la valeur est 1. Le programme renvoie (écrit) ces paires clé-valeur sur STDOUT. Un délimiteur (par défaut, '\t' est le délimiteur) doit être utiliser entre la clé et la valeur pour le traitement correct d'une paire clé-valeur lors de l'étape de Shuffle.

En utilisant votre environnement de développement Python, créez un fichier mapperWC.py. Il est préférable de saisir ce code manuellement plutôt que de le copier/coller pour mieux comprendre ce que fait le code.

```
# mapperwc.py
import sys

for line in sys.stdin:
    # Assurez-vous que line est une chaîne de caractères
    if isinstance(line, str):
        words = line.split()
        for word in words:
            print("{0}\t1".format(word))
```

Le programme reducer combine les paires clé-valeur en additionnant le nombre d'occurrences de chaque mot, puis renvoie le résultat sur la sortie standard (STDOUT). Pour ce faire, créez un fichier reducerWC.py et entrez le code suivant :

```
# reducerwc.py
import sys
current_word = None
current_count = 0
word = None

for line in sys.stdin:
    line = line.strip()
    word, count = line.split('\t', 1)
    try:
        count = int(count)
    except ValueError:
        continue
    if current_word == word:
        current_count += count
    else:
        if current_word:
            print('{}\t{}'.format(current_word, current_count))
        current_word = word
        current_count = count

if current_word == word:
    print('{}\t{}'.format(current_word, current_count))
```

Execution du programme en local:

C'est une bonne idée de tester le code localement sur de petits fichiers avant de l'intégrer dans un travail de streaming MapReduce. Vous pouvez effectuer des tests locaux en utilisant la méthode de pipeline typique de style Unix. Pour ce faire, combinez l'entrée standard (stdin) et la sortie standard (stdout) de votre script Python (mapperwc.py) en utilisant des commandes de redirection de flux. Assurez-vous d'avoir un fichier (par exemple "input.txt") pour exécuter vos tests.

Ouvrez PowerShell (ou un autre terminal, en fonction de votre système d'exploitation) et saisissez la commande suivante : type 'C:\users\21627\Desktop\Nouveau dossier\input.txt | python mapperwc.py | python reducerwc.py

```
C:\Users\21627\Desktop\Nouveau dossier>echo "Ceci est un exemple de texte dans le fichier input.txt." > "C:\Users\21627\Desktop
\Nouveau dossier\input.txt"

C:\Users\21627\Desktop\Nouveau dossier>type "C:\Users\21627\Desktop\Nouveau dossier\input.txt" | python "C:\Users\21627\Desktop
\Nouveau dossier\mapperwc.py" | python "C:\Users\21627\Desktop\Nouveau dossier\reducerwc.py"
"Ceci    1
est      1
un       1
exemple  1
de       1
texte    1
dans     1
le       1
fichier  1
input.txt."     1
```

Vous pouvez faire le même test sur votre container namenode.

Execution du programme en local:

Une fois que le code a été exécuté localement avec succès, l'étape suivante consiste à le déployer sur le cluster à l'aide de Hadoop Streaming. Pour ce faire, suivez les étapes suivantes1. :

1.Transférer les scripts dans le cluster Hadoop:

docker cp mapperwc.py namenode:/mapperwc.py

docker cp reducerwc.py namenode:/reducerwc.py

```
C:\Users\21627\Desktop\Nouveau dossier>docker cp mapperwc.py namenode:/mapperwc.py
Successfully copied 2.05kB to namenode:/mapperwc.py

C:\Users\21627\Desktop\Nouveau dossier>
```

```
C:\Users\21627\Desktop\Nouveau dossier>docker cp reducerwc.py namenode:/reducerwc.py
Successfully copied 2.56kB to namenode:/reducerwc.py
```

2.Entrer dans le container du namenode :

docker exec -it namenode bash

```
C:\Users\21627\Desktop\Nouveau dossier>docker exec -it namenode bash
```

3.Créer des fichiers textes et les transferer dans HDFS :

echo "Hello World" > input/f1.txtecho "Hello World" > input/f1.txt

echo "Hello Docker" > input/f2.txt

echo "Hello Hadoop" > input/f3.txt

echo "Hello MapReduce" > input/f4.txt

```
C:\Users\21627\Desktop\Nouveau dossier>docker exec -it namenode bash
root@b64f55210ee0:/# echo "Hello World" > input/f1.txtecho "Hello World" > input/f1.txt
root@b64f55210ee0:/# echo "Hello Docker" > input/f2.txt
root@b64f55210ee0:/# echo "Hello Hadoop" > input/f3.txt
root@b64f55210ee0:/# echo "Hello MapReduce" > input/f4.txt
```

hadoop fs -mkdir -p /input

hdfs dfs -put ./input/* input

hdfs dfs -ls /

```
root@b64f55210ee0:/# hadoop fs -mkdir -p  /input
root@b64f55210ee0:/# hdfs dfs -put ./input/* input
put: `input/f1.txt': File exists
put: `input/f2.txt': File exists
put: `input/f3.txt': File exists
put: `input/f4.txt': File exists
root@b64f55210ee0:/# hdfs dfs -ls /
Found 8 items
drwxr-xr-x   - root supergroup          0 2023-10-22 20:26 /TPs
drwxr-xr-x   - root supergroup          0 2023-10-22 23:25 /data
drwxr-xr-x   - root supergroup          0 2023-10-22 20:13 /dossier
drwxr-xr-x   - root supergroup          0 2023-11-13 17:03 /input
drwxr-xr-x   - root supergroup          0 2023-10-09 18:55 /rmstate
drwxr-xr-x   - root supergroup          0 2023-10-22 16:37 /root
drwxr-xr-x   - root supergroup          0 2023-10-24 15:12 /user
drwxr-xr-x   - root supergroup          0 2023-10-22 16:27 /var
root@b64f55210ee0:/#
```

4.Executer le programme MapReduce :

Pour cela, vous devez d’abord localiser le fichier JAR Hadoop Streaming

find / -name ‘hadoop-streaming*.jar’

les scripts doivent être eexécutables :

chmod u+x mapperwc.py

chmod u+x reducerwc.py

```
root@b64f55210ee0:/# find / -name 'hadoop-streaming*.jar'
root@b64f55210ee0:/# chmod u+x mapperwc.py
root@b64f55210ee0:/# chmod u+x reducerwc.py
root@b64f55210ee0:/#
```

Pour exécuter le programme MapReduce, il faut lancer la commande suivante:

```
root@b64f55210ee0:/# hadoop jar /opt/hadoop-3.2.1/share/hadoop/tools/lib/hadoop-streaming-3.2.1.jar \
> -files mapperwc.py,reducerwc.py \-files mapperwc.py,reducerwc.py \
> -mapper "python mapperwc.py" \
> -reducer "python reducerwc.py" \
> -input input \
> -output output
```

5. **Voir Les résultats :**

Pour voir les resultats, vous pouvez lister les fichiers de sortie et les afficher:

hdfs dfs -ls output hdfs ds -cat output/part-r-00000

```
"Ceci   1
est     1
un      1
exemple 1
de      1
texte   1
dans    1
le      1
fichier 1
input.txt."     1
```