REPUBLIQUE DU SENEGAL

Un peuple –un but –une foi

MINISTERE DE L'ENSEIGNEMENT SUPERIEURE DE LA RECHERCHE ET DE L'INNOVATION

Université Gaston Berger

UFR Institut Polytechnique de Saint-Louis

Rapport de TP : Traitement de Données avec MapReduce et Hadoop

Présenté par :

Malick DIOP **Code étudiant : P3084**

Professeur : Professeur Serigne Mboup

I. Introduction

Ce rapport détaille les étapes d'un projet de traitement de données à grande échelle en utilisant le framework MapReduce et la plateforme Hadoop. Il couvre l'introduction à ces technologies, la mise en place de l'environnement, l'exécution de tâches MapReduce simples, l'analyse approfondie des données, l'optimisation des performances et la gestion des erreurs. Chaque section fournit des informations approfondies et des conseils pratiques pour mener à bien ce type de projet.

II. Objectifs du TP

- Comprendre le fonctionnement de Hadoop et du modèle MapReduce.
- Déployer un environnement Hadoop.
- Implémenter et exécuter des programmes MapReduce.
- Analyser les résultats obtenus.

III. Présentation et fonctionnement de MapReduce et Hadoop

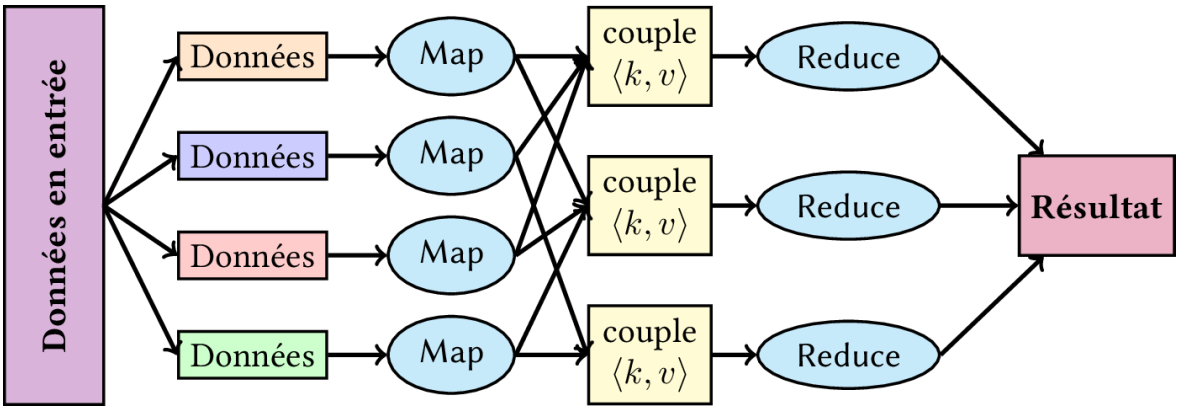
1. Présentation

MapReduce est un modèle de programmation pour le traitement de grands volumes de données en parallèle sur des grappes de serveurs. Il se compose de deux étapes principales : Map qui transforme les données en paires clé-valeur, et Reduce qui agrège ces paires pour produire un résultat final. Hadoop est une plateforme open-source qui implémente le modèle MapReduce et fournit un système de fichiers distribué (HDFS) pour stocker et traiter efficacement les données à grande échelle.

2. Fonctionnement

MapReduce est un modèle de programmation développé par Google pour le traitement et la génération de grandes quantités de données. Ce modèle est particulièrement efficace pour traiter des données réparties sur un cluster de machines. Le fonctionnement de MapReduce repose sur deux étapes principales : **Map** et **Reduce**.

- Étape de Map : L'étape de Map prend en entrée des paires clé-valeur et génère un ensemble intermédiaire de paires clé-valeur.
- Étape de Reduce : L'étape de Reduce prend les paires clé-valeur triées et groupées par clé produites par l'étape de Map, et les transforme en un ensemble final de paires clé-valeur.



IV. Installation et Configuration

- Tout d'abord, nous allons mettre en place le **cluster Hadoop** en utilisant **VIRTUALBOX** et **VAGRANT** c'*est-**à-dire l'environnement virtuel ou sera exécuté Hadoop.***
- Ensuite cloner le dépôt git qui contient le fichier de configuration. Grace a la commande suivante : **git clone** https://github.com/sopeKhadim/hadoopVagrant.git
- Maintenant on lance la machine virtuelle en exécutant la commande **VAGRANT UP** sur le répertoire où se trouve le fichier vagranfile. Cela permet de télécharger l'image de la machine, la configurer puis la démarrer.

```
LENOVO@DESKTOP-3LBOAEV MINGW64 ~/Downloads/hadoopVagrant-main (1)/hadoopVagrant-main (main)
$ vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Box 'SopeKhadim/hadoopVM' could not be found. Attempting to find and install...
    default: Box Provider: virtualbox
    default: Box Version: 2.0
==> default: Loading metadata for box 'SopeKhadim/hadoopVM'
    default: URL: https://vagrantcloud.com/api/v2/vagrant/SopeKhadim/hadoopVM
==> default: Adding box 'SopeKhadim/hadoopVM' (v2.0) for provider: virtualbox
    default: Downloading: https://vagrantcloud.com/SopeKhadim/boxes/hadoopVM/versions/2.0/providers/virtualbox/unknown/vagrant.box
==> default: Box download is resuming from prior download progress
    default:
==> default: Successfully added box 'SopeKhadim/hadoopVM' (v2.0) for 'virtualbox'!
==> default: Importing base box 'SopeKhadim/hadoopVM'...
==> default: Matching MAC address for NAT networking...
==> default: Checking if box 'SopeKhadim/hadoopVM' version '2.0' is up to date...
==> default: Setting the name of the VM: hadoopVagrant-main_default_1721953649304_44366
Vagrant is currently configured to create VirtualBox synced folders with
the `SharedFoldersEnableSymlinksCreate` option enabled. If the Vagrant
guest is not trusted, you may want to disable this option. For more
information on this option, please refer to the VirtualBox manual:
```

- Apres le démarrage, il faut se connecter avec la machine virtuelle en exécutant **VAGRANT SSH**

```
LENOVO@DESKTOP-3LBOAEV MINGW64 ~/Downloads/hadoopVagrant-main (1)/hadoopVagrant-main (main)
$ vagrant ssh
Last login: Sun Nov 28 11:04:44 2021 from 10.0.2.2
[vagrant@10 ~]$ ls
C:              metastore_db         mysql57-community-release-el7-10.noarch.rpm  mysql-connector-java-5.1.49.zip  sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz
derby.log       mysql-5.7.33         mysql-connector-java-5.1.48                  README.md                        wget-log
itversity-books mysql-5.7.33.tar.gz  mysql-connector-java-5.1.48.tar.gz            shareFolder
[vagrant@10 ~]$
```

V. Implémentation

- Le code source

compteurMot.py - C:/Users/LENOVO/Desktop/DAKHAR/prog python/compteurMot.py (3.12.3)

File Edit Format Run Options Window Help

```
# -*- coding: utf-8 -*-
from mrjob.job import MRJob
from mrjob.step import MRStep
import re

# Définition d'une expression régulière pour extraire les mots
WORD_RE = re.compile(r"[\w']+")

class WordCount1(MRJob):

    def steps(self):
        return [
            MRStep(mapper=self.mapper_get_words,
                   reducer=self.reducer_count_words)
        ]

    def mapper_get_words(self, _, line):
        # Découpage de la chaîne et itération sur chaque mot
        for word in WORD_RE.findall(line):
            yield (word.lower(), 1)

    def reducer_count_words(self, word, counts):
        # On définit une variable intermédiaire pour stocker le nombre d'occurrences
        yield (word, sum(counts))

if __name__ == '__main__':
    WordCount1.run()
```

Il faut d'abord exécuter la commande **PIP INSTALL MRJOB** pour installer la bibliothèque

MRJOB

- Lancer le cluster Hadoop **Start-dfs.sh**
  Start-yarn.sh
- Télécharger le fichier `ancien_figaro.txt` et le placer dans `/home/vagrant/shareFolder/`. Et copie les données dans HDFS :
  NDFS DFS -MKDIR -P /USER/HADOOP/
  NDFS DFS -PUT
  /HOME/VARGRANT/SHAREFOLDER/ANCIEN_FIGARO.TXT/USER/HADOOP/
- Executer le jop MapReduce
  `python WordCount.py -r hadoop`
  `hdfs:///user/hadoop/ancien_figaro.txt - o`
  `hdfs:///user/hadoop/output`
- **Enfin verifier les resultats**
  `hdfs dfs -ls /user/hadoop/output/`(sortie sur HDFS) `hdfs dfs -head /user/hadoop/output/part-00000`(afficher le contenu) `hdfs dfs -tail /user/hadoop/output/part-00000`(afficher le contenu)

```
C:\Users\LENOVO\Desktop\hadoopVagrant-main>pip install mrjob
Collecting mrjob
  Downloading mrjob-0.7.4-py2.py3-none-any.whl.metadata (7.3 kB)
Collecting PyYAML>=3.10 (from mrjob)
  Downloading PyYAML-6.0.1-cp312-cp312-win_amd64.whl.metadata (2.1 kB)
Downloading mrjob-0.7.4-py2.py3-none-any.whl (439 kB)
   ---------------------------------------- 439.6/439.6 kB 92.5 kB/s eta 0:00:00
Downloading PyYAML-6.0.1-cp312-cp312-win_amd64.whl (138 kB)
   ---------------------------------------- 138.7/138.7 kB 149.5 kB/s eta 0:00:00
Installing collected packages: PyYAML, mrjob
Successfully installed PyYAML-6.0.1 mrjob-0.7.4

[notice] A new release of pip is available: 24.0 -> 24.1.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

```
[vagrant@localhost shareFolder]$ python WordCount.py -r hadoop hdfs:///user/
hadoop/ancien_figaro.txt -o hdfs:///user/hadoop/output
No configs found; falling back on auto-configuration
No configs specified for hadoop runner
Looking for hadoop binary in /opt/hadoop/bin...
Found hadoop binary: /opt/hadoop/bin/hadoop
Using Hadoop version 3.2.1
Looking for Hadoop streaming jar in /opt/hadoop...
Found Hadoop streaming jar: /opt/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.2.1.jar
Creating temp directory /tmp/WordCount.vagrant.20240719.011203.407301
uploading working dir files to hdfs:///user/vagrant/tmp/mrjob/WordCount.vagrant.20240719.011203.407301/files/wd...
Copying other local files to hdfs:///user/vagrant/tmp/mrjob/WordCount.vagrant.20240719.011203.407301/files/
Running step 1 of 1...
  packageJobJar: [/tmp/hadoop-unjar6715198535264967943/] [] /tmp/streamjob8660867222631578809.jar tmpDir=null
  Connecting to ResourceManager at /0.0.0.0:8032
  Connecting to ResourceManager at /0.0.0.0:8032
  Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/vagrant/.staging/job_1721348138477_0001
  SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
  Total input files to process : 1
  SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
  SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
  number of splits:2
  SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
  Submitting tokens for job: job_1721348138477_0001
  Executing with tokens: []
  resource-types.xml not found
  Unable to find 'resource-types.xml'.
  Submitted application application_1721348138477_0001
  The url to track the job: http://localhost:8088/proxy/application_1721348138477_0001/
  Running job: job_1721348138477_0001
  Job job_1721348138477_0001 running in uber mode : false
   map 0% reduce 0%
   map 50% reduce 0%
   map 100% reduce 0%
   map 100% reduce 100%
  Job job_1721348138477_0001 completed successfully
  Output directory: hdfs:///user/hadoop/output
Counters: 55
        File Input Format Counters
```

VI. Conclusion

Ce TP a permis de se familiariser avec Hadoop et le modèle de programmation MapReduce. En déployant Hadoop, en implémentant un job MapReduce et en analysant les résultats, nous avons démontré la puissance de ce framework pour traiter des volumes de données massifs de manière efficace.