



Rapport de Travaux Pratiques — Big Data

Ce rapport présente l'exploration et la mise en œuvre de technologies fondamentales pour le traitement des données massives.

Demandé par : Dr.Mohamed El Moustapha El Arby

Elaboré par : Khaita Loudaa C19006



Introduction aux Technologies Big Data



MongoDB : Base de Données NoSQL

Découverte et manipulation de données semi-structurées via des opérations CRUD et des requêtes.



Hadoop : Framework Distribué

Stockage et traitement de grands volumes de données grâce au modèle MapReduce.



Apache Spark : Traitement Distribué

Initiation au traitement de graphes sociaux et calcul d'amis communs.

TP N°1 : Bases de données NoSQL « MongoDB »

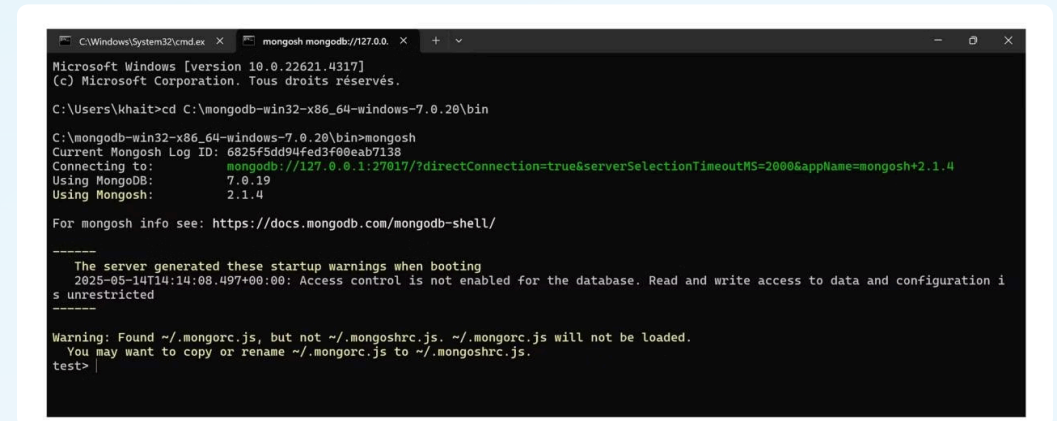
Objectif du TP MongoDB

Découvrir les concepts de base de MongoDB : installation, configuration et utilisation du serveur et du client sous Windows. Créer une base de données, manipuler des collections et des documents via des opérations CRUD, et exécuter des requêtes pour interagir avec les données.



Installation et Configuration

Téléchargement et extraction de MongoDB, création des dossiers de stockage C:\data et C:\data\db. Lancement du serveur (mongod.exe) et du client (mongosh) via la ligne de commande pour interagir avec la base de données.



TP N°1 : Exercice 01 - Opérations CRUD

Création d'une base de données nommée "info" et d'une collection "produits".

```
test> use info
switched to db info
info> db
info
info> |
```

Quelques Requêtes

- Récupérer tous les produits.
- Récupérer le premier produit.
- Trouver et récupérer un produit par son ID.

```
info> db.produits.findOne({ _id: ObjectId("6825fa9c94fed3f00eab713b") })
{
  _id: ObjectId('6825fa9c94fed3f00eab713b'),
  nom: 'Thinkpad X230',
  fabricant: 'Lenovo',
  prix: 114358.74,
  ultrabook: true,
  options: [ 'Intel Core i5', 'SSD', 'Long life battery' ]
}
info> |
```

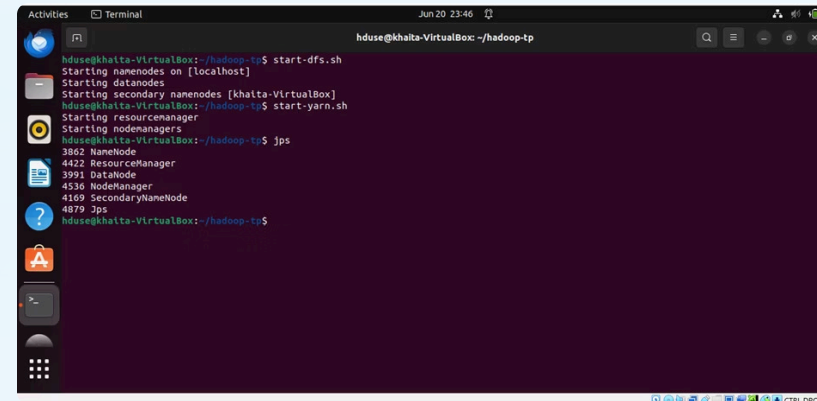
TP N°2 : Framework HADOOP (WordCount)

Objectif du TP Hadoop

Mettre en place un environnement Big Data en configurant Hadoop en mode pseudo-distribué sur Linux. Installer Hadoop, préparer HDFS, compiler un programme Java MapReduce, et l'exécuter pour compter les occurrences de mots.

Etapes clés

- Création de l'utilisateur 'hduse'.
- Téléchargement et extraction d'Hadoop.
- Installation de Java 8 et configuration de JAVA_HOME.
- Configuration des fichiers core-site.xml, hdfs-site.xml, mapred-site.xml, yarn-site.xml.
- Préparation des répertoires HDFS.
- Formatage du NameNode et démarrage des démons.



```
hduse@khaita-VirtualBox: ~/hadoop-tp
hduse@khaita-VirtualBox:~/hadoop-tp$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [khaita-VirtualBox]
hduse@khaita-VirtualBox:~/hadoop-tp$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
hduse@khaita-VirtualBox:~/hadoop-tp$ jps
3802 NameNode
4422 ResourceManager
3991 DataNode
4536 NodeManager
4169 SecondaryNameNode
4879 Jps
hduse@khaita-VirtualBox:~/hadoop-tp$
```

TP N°2 : Exécution MapReduce

Exécution du programme MapReduce pour le comptage de mots :

Préparation des Données

Création des dossiers HDFS et ajout du fichier "poeme.txt".

Compilation et Création du JAR

Compilation du code Java (WCount.java) et création du fichier JAR (wcount.jar).

Exécution du Job

Lancement du job MapReduce via la commande `hadoop jar`.

Récupération des Résultats

Affichage des résultats du comptage de mots depuis HDFS.

```
Machine Écran Entrée Périphériques Aide
ities Terminal

4879 Jps
hduse@khaïta-VirtualBox:~/hadoop-tp$ hadoop jar wcount.jar WordCount /input /output
2025-06-20 23:47:08,848 INFO client.DefaultHadoopConf: Exception in thread "main" org.apache.hadoop.mapreduce.lib.output.FileOutputFormat: The output file /output/tpout already exists
at org.apache.hadoop.mapreduce.lib.output.FileOutputFormat.(FileOutputFormat.java:100)
at org.apache.hadoop.mapreduce.JobSubmitter.submitJob(JobSubmitter.java:400)
at org.apache.hadoop.mapreduce.JobSubmitter.submitJob(JobSubmitter.java:375)
at org.apache.hadoop.mapreduce.Job$11.run(Job.java:1111)
at org.apache.hadoop.mapreduce.Job$11.run(Job.java:1111)
at java.security.AccessController.doPrivileged(AccessController.java:390)
at javax.security.auth.Subject.doAs(Subject.java:422)
at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1642)
at org.apache.hadoop.mapreduce.Job.submit(Job.java:1111)
at org.apache.hadoop.mapreduce.Job.waitForCompletion(Job.java:1141)
at WCount.main(WCount.java:59)
at sun.reflect.NativeMethodAccessorImpl.invoke0(NativeMethodAccessorImpl.java:-2)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
at java.lang.reflect.Method.invoke(Method.java:498)
at org.apache.hadoop.util.RunJar.run(RunJar.java:209)
at org.apache.hadoop.util.RunJar.main(RunJar.java:116)

hduse@khaïta-VirtualBox:~/hadoop-tp$ hdfs dfs -ls /output
Big 2
Bonjour 2
Data 2
Hadoop 1
puissant 1
hduse@khaïta-VirtualBox:~/hadoop-tp$
```

Made with GAMMA

TD : Spark et Graphes Sociaux

Objectif du TD Spark

Familiarisation avec Spark et la programmation distribuée via l'analyse d'un mini-graphe social. Générer des paires d'amis, calculer les amis communs avec MapReduce, et manipuler des RDD en Scala.

TP N°3 : Spark et Graphes Sociaux

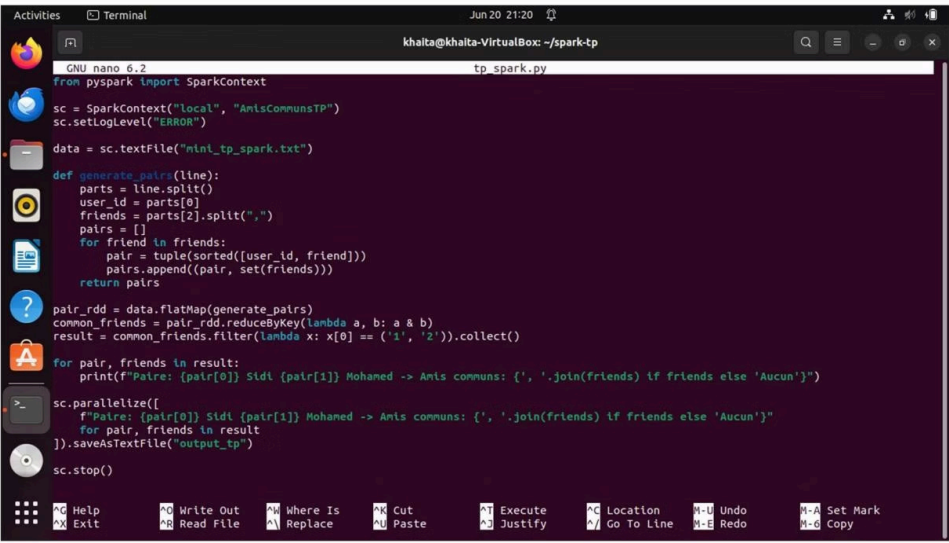
Données du Graphe Social

Utilisation du fichier 'mini_tp_spark.txt' avec des utilisateurs et leurs listes d'amis mutuels.

| | | |
|---|---------|-------|
| 1 | Sidi | 2,3,4 |
| 2 | Mohamed | 1,3,4 |
| 3 | Ali | 1,2,4 |
| 4 | Samir | 1,2,3 |
| 5 | Karim | 6,7 |
| 6 | Youssef | 5,7 |
| 7 | Amine | 5,6 |

Exécution Interactive et Structurée

Calcul des amis communs entre deux utilisateurs (Sidi et Mohamed) via un fichier Python structuré (exécuté avec spark-submit).



```
Activities Terminal Jun 20 21:20 khalta@khalta-VirtualBox: ~/spark-tp
GNU nano 6.2 tp_spark.py
from pyspark import SparkContext
sc = SparkContext("local", "AmisCommunsTP")
sc.setLogLevel("ERROR")
data = sc.textFile("mini_tp_spark.txt")
def generate_pairs(line):
    parts = line.split()
    user_id = parts[0]
    friends = parts[2].split(",")
    pairs = []
    for friend in friends:
        pair = tuple(sorted([user_id, friend]))
        pairs.append(pair)
    return pairs
pair_rdd = data.flatMap(generate_pairs)
common_friends = pair_rdd.reduceByKey(lambda a, b: a & b)
result = common_friends.filter(lambda x: x[0] == ('1', '2')).collect()
for pair, friends in result:
    print(f"Patre: {pair[0]} Sidi {pair[1]} Mohamed -> Amis communs: {' '.join(friends) if friends else 'Aucun'}")
sc.parallelize([
    f"Patre: {pair[0]} Sidi {pair[1]} Mohamed -> Amis communs: {' '.join(friends) if friends else 'Aucun'}"
    for pair, friends in result
]).saveAsTextFile("output_tp")
sc.stop()
```



```
db.produits.findOne({ nom: "Thinkpad X230" })

: ObjectId('6826417394fed3f00eab713c'),
: 'Thinkpad X230',
riquant: 'Lenovo',
x: 114358.74,
raboook: true,
ions: [ 'Intel Core i5', 'SSD', 'Long life battery' ]

db.produits.deleteOne({ _id: ObjectId("6826417394fed3f00eab713c") })
nowledged: true, deletedCount: 1 }
|
```

```
Terminal
Jun 20 23:48
hduse@khalita-VirtualBox: ~/hadoop-tp

79 Jps
hduse@khalita-VirtualBox:~/hadoop-tp$ hadoop jar WCount.jar WCount /user/hduse/poeme.txt /user/hduse/output
25-06-20 23:47:08,848 INFO client.DefaultHARMFaloverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
ception in thread "main" org.apache.hadoop.mapred.FileAlreadyExistsException: Output directory hdfs://localhost:9000/user
ut already exists
at org.apache.hadoop.mapreduce.lib.output.FileOutputFormat.checkOutputSpecs(FileOutputFormat.java:164)
at org.apache.hadoop.mapreduce.JobSubmitter.checkSpecs(JobSubmitter.java:277)
at org.apache.hadoop.mapreduce.JobSubmitter.submitJobInternal(JobSubmitter.java:143)
at org.apache.hadoop.mapreduce.Job$11.run(Job.java:1676)
at org.apache.hadoop.mapreduce.Job$11.run(Job.java:1675)
at java.security.AccessController.doPrivileged(Native Method)
at javax.security.auth.Subject.doAs(Subject.java:422)
at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1899)
at org.apache.hadoop.mapreduce.Job.submit(Job.java:1675)
at org.apache.hadoop.mapreduce.Job.waitForCompletion(Job.java:1696)
at WCount.main(WCount.java:59)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
at java.lang.reflect.Method.invoke(Method.java:498)
at org.apache.hadoop.util.RunJar.run(RunJar.java:328)
at org.apache.hadoop.util.RunJar.main(RunJar.java:241)
hduse@khalita-VirtualBox:~/hadoop-tp$ hdfs dfs -cat /user/hduse/output/part-r-00000
0 2
njour 2
ta 2
doop 1
ssant 1
hduse@khalita-VirtualBox:~/hadoop-tp$
```

Khaitaloudaa / tp-spark-amis-communs

Issues Pull requests Actions Projects Wiki Security Insights Settings

spark-amis-communs Public

Pin Watch Fork Star

1 Branch 0 Tags

Go to file Add file Code

About

TP Spark pour calcul des amis com entre deux utilisateurs

Activity

0 stars

0 watching

0 forks

Releases

No releases published

Create a new release

| Commit | Message | Author | Time |
|----------------|----------|--------------|-------------|
| Initial commit | TP Spark | Khaitaloudaa | 2 hours ago |
| Initial commit | TP Spark | Khaitaloudaa | 2 hours ago |
| Initial commit | TP Spark | Khaitaloudaa | 2 hours ago |
| Initial commit | TP Spark | Khaitaloudaa | 2 hours ago |
| Initial commit | TP Spark | Khaitaloudaa | 2 hours ago |

Conclusion et Perspectives

Ces travaux pratiques ont permis d'explorer concrètement les technologies majeures du Big Data : MongoDB pour les données semi-structurées, Hadoop pour le traitement distribué MapReduce, et Spark pour l'analyse rapide de graphes sociaux.

Le projet final est hébergé sur GitHub, démontrant l'application des compétences acquises.

Voir le Projet GitHub