



Rapport de Travaux Pratiques — Big DataE

Exploration des technologies MongoDB,
Hadoop et Spark



Demandé par : Dr.Mohamed El Moustapha El Arby

Elaboré par : Khaita Loudaa C19006

I. Introduction

Ce rapport présente le travail réalisé dans le cadre de la matière **Big Data**, au cours du semestre, à travers trois travaux pratiques majeurs. Ces TPs permettent de découvrir, d'explorer et de mettre en œuvre des technologies fondamentales pour le traitement des données massives, illustrant concrètement les concepts étudiés en cours.

Le premier TP est consacré à **MongoDB**, une base de données NoSQL orientée document, utilisée pour manipuler des données semi-structurées et effectuer des opérations CRUD. Le deuxième TP porte sur **Hadoop**, un framework distribué permettant de stocker et traiter de grands volumes de données grâce au modèle **MapReduce**.

Le troisième TP est dédié à **Apache Spark**, d'abord sous forme de TD pour s'initier au traitement distribué de graphes sociaux, puis sous forme de TP complet mettant en pratique le calcul d'amis communs sur un graphe social simulé.

Ces activités visent à développer une compréhension concrète et appliquée des outils du Big Data, en reliant théorie et pratique, et en nous préparant à relever les défis du traitement des données à grande échelle.

II. Tp N°1 : Bases de données NoSQL « MongoDB »

Objectif du TP MongoDB :

L'objectif de cette partie du projet est de découvrir les concepts de base de MongoDB à travers l'installation, la configuration et l'utilisation de son serveur et de son client sous Windows. Elle consiste à créer une base de données, manipuler des collections et des documents via des opérations CRUD (Create, Read, Update, Delete), et à exécuter des requêtes pour interagir avec les données de manière pratique.

1. Installation de MongoDB sous Windows :

- Téléchargez la version zip de MongoDB pour Windows (64-bit zip).



- Extraire l'archive zip dans le dossier C:\MongoDB (elle doit contenir un répertoire bin\).

Nom	Modifié le	Type	Taille
Install-Compass	15/05/2025 1:19 PM	Script Windows Powe...	2 Ko
mongod	15/05/2025 1:19 PM	Application	63 024 Ko
mongod.pdb	15/05/2025 1:20 PM	Fichier PDB	101 7532 Ko
mongos	15/05/2025 1:20 PM	Application	37 065 Ko
mongos.pdb	15/05/2025 1:20 PM	Fichier PDB	57 4876 Ko
vc_redist.x64	15/05/2025 1:20 PM	Application	24 722 Ko

- Créez les dossiers C:\data et C:\data\db (c'est là que MongoDB stocke vos données et d'autres choses).

Nom	Modifié le
db	15/05/2025 2:14

2. Serveur mongo :

Dans une ligne de commande: lancez le serveur : bin\mongod.exe

```

C:\Windows\System32\cmd.exe > mongosh mongodb://127.0.0.1 > + >
Microsoft Windows [version 10.0.22621.4317]
(c) Microsoft Corporation. Tous droits réservés.

C:\mongodb-win32-x86_64-windows-7.0.20\bin>mongod.exe
{"t": {"$date": "2025-05-15T14:08:33.757+00:00"}, "s": "I", "c": "NETWORK", "id": 4915701, "ctx": "thread1", "msg": "Initialized wire specification", "attr": {"spec": {"incomingExternalClient": {"minWireVersion": 0, "maxWireVersion": 21}, "incomingInternalClient": {"minWireVersion": 0, "maxWireVersion": 21}, "outgoing": {"minWireVersion": 6, "maxWireVersion": 21}, "isInternalClient": true}}}
{"t": {"$date": "2025-05-15T14:08:36.087+00:00"}, "s": "I", "c": "CONTROL", "id": 23285, "ctx": "thread1", "msg": "Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
{"t": {"$date": "2025-05-15T14:08:36.091+00:00"}, "s": "I", "c": "NETWORK", "id": 4648602, "ctx": "thread1", "msg": "Implicit TCP FastOpen in use"}
{"t": {"$date": "2025-05-15T14:08:36.097+00:00"}, "s": "I", "c": "REPL", "id": 5123008, "ctx": "thread1", "msg": "Successfully registered PrimaryOnlyService", "attr": {"service": "TenantMigrationDonorService", "namespace": "config.tenantMigrationDonors"}}
{"t": {"$date": "2025-05-15T14:08:36.097+00:00"}, "s": "I", "c": "REPL", "id": 5123008, "ctx": "thread1", "msg": "Successfully registered PrimaryOnlyService", "attr": {"service": "TenantMigrationRecipientService", "namespace": "config.tenantMigrationRecipients"}}
{"t": {"$date": "2025-05-15T14:08:36.098+00:00"}, "s": "I", "c": "CONTROL", "id": 5945603, "ctx": "thread1", "msg": "Multi threading initialized"}
{"t": {"$date": "2025-05-15T14:08:36.098+00:00"}, "s": "I", "c": "TENANT_M", "id": 7091600, "ctx": "thread1", "msg": "Starting TenantMigrationAccessBlockerRegistry"}
{"t": {"$date": "2025-05-15T14:08:36.099+00:00"}, "s": "I", "c": "CONTROL", "id": 4615611, "ctx": "initandlisten", "msg": "MongoDB starting", "attr": {"pid": 17304, "port": 27017, "dbPath": "C:/data/db/", "architecture": "64-bit", "host": "Khaita"}}
{"t": {"$date": "2025-05-15T14:08:36.100+00:00"}, "s": "I", "c": "CONTROL", "id": 23398, "ctx": "initandlisten", "msg": "Target operating system minimum version", "attr": {"targetMinOS": "Windows 7/Windows Server 2008 R2"}}
{"t": {"$date": "2025-05-15T14:08:36.100+00:00"}, "s": "I", "c": "CONTROL", "id": 23403, "ctx": "initandlisten", "msg": "Build Info", "attr": {"buildInfo": {"version": "7.0.20", "gitVersion": "cf29fc744f8ee2ac9245f2845f29c6a706dc375a", "modules": [], "allocator": "tcmalloc", "environment": {"distmod": "windows", "distarch": "x86_64", "target_arch": "x86_64"}}, "os": {"name": "Microsoft Windows 10", "version": "10.0 (build 22621)"}, "options": {}}}
{"t": {"$date": "2025-05-15T14:08:36.101+00:00"}, "s": "I", "c": "CONTROL", "id": 21951, "ctx": "initandlisten", "msg": "Options set by command line", "attr": {"options": {}}}
{"t": {"$date": "2025-05-15T14:08:36.105+00:00"}, "s": "I", "c": "STORAGE", "id": 22315, "ctx": "initandlisten", "msg": "Opening WiredTiger", "attr": {"config": "create,cache_size=7507M,session_max=33000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,remove=true,path=journal,compressor=snappy),builtin_extension_config=(zstd=(compression_level=6)),file_m

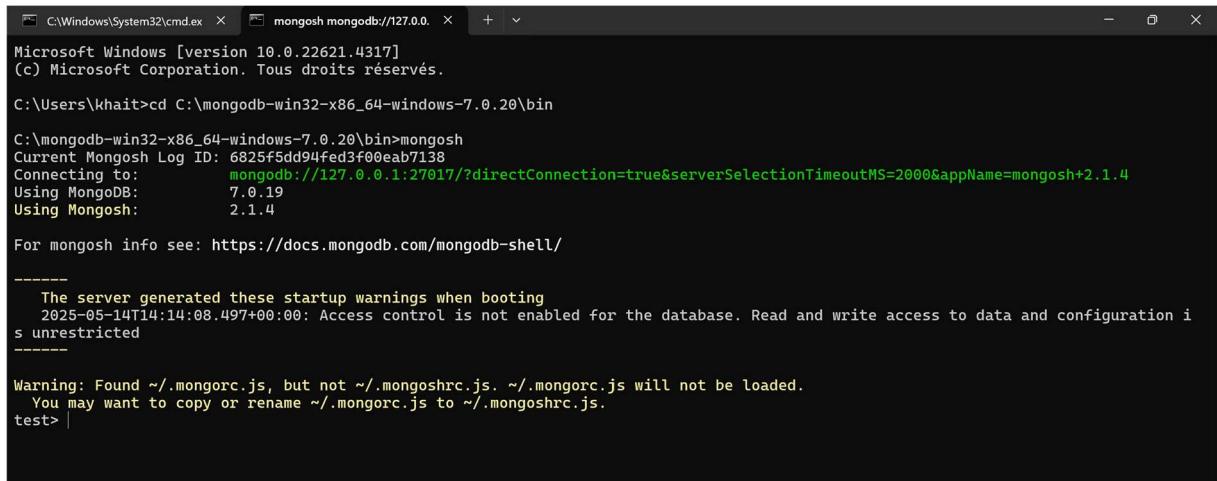
```

3. Client mongo :

Ce que nous venons de lancer c'est le serveur

Pour interagir avec ce serveur, il nous faut un client. Pour l'instant, nous allons lancer nos requêtes au serveur au travers de la console mongo.

Dans une ligne de commande lancez l'exécutable bin\mongosh



```
C:\Windows\System32\cmd.exe  mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.1.4
Microsoft Windows [version 10.0.22621.4317]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\khait>cd C:\mongodb-win32-x86_64-windows-7.0.20\bin
C:\mongodb-win32-x86_64-windows-7.0.20\bin>mongosh
Current Mongosh Log ID: 6825f5dd94fed3f00eab7138
Connecting to:      mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.1.4
Using MongoDB:      7.0.19
Using Mongosh:      2.1.4

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

-----
The server generated these startup warnings when booting
2025-05-14T14:14:08.497+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
-----
Warning: Found ~/.mongorc.js, but not ~/.mongoshrc.js. ~/.mongorc.js will not be loaded.
You may want to copy or rename ~/.mongorc.js to ~/.mongoshrc.js.
test> |
```

Exercice 01

Création d'une BD Mongo et opération CRUD

- Créer une base de données nommée info et vérifiez qu'elle est sélectionnée.

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

- Créer une nouvelle collection nommée produits et y insérer le document suivant :

(nom: Macbook Pro

fabriquant: Apple

prix: 17435,99

options: Intel Core i5

Retina Display

Long life battery)

```

info> db.produits.insertOne({
...   nom: "Macbook Pro",
...   fabriquant: "Apple",
...   prix: 17435.99,
...   options: ["Intel Core i5", "Retina Display", "Long life battery"]
... })
{
  acknowledged: true,
  insertedId: ObjectId('6825fa2094fed3f00eab7139')
}
info> |

```

c. Rajouter deux autres documents dans produits :

```

(nom: DELL
fabriquant: Dell Technologies, Inc
prix: 1143
options: Intel® Core™ Ultra 9 285H
SSD
32 Go)
(nom: Thinkpad X230
fabriquant: Lenovo
prix: 114358,74
ultrabook: true
options: Intel Core i5
SSD
Long life battery)

```

```

info> db.produits.insertMany([
...   {
...     nom: "DELL",
...     fabriquant: "Dell Technologies, Inc",
...     prix: 1143,
...     options: ["Intel® Core™ Ultra 9 285H", "SSD", "32 Go"]
...   },
...   {
...     nom: "Thinkpad X230",
...     fabriquant: "Lenovo",
...     prix: 114358.74,
...     ultrabook: true,
...     options: ["Intel Core i5", "SSD", "Long life battery"]
...   }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6825fa9c94fed3f00eab713a'),
    '1': ObjectId('6825fa9c94fed3f00eab713b')
  }
}
info> |

```

d. Effectuez les requêtes de lecture suivantes:

- Récupérer tous les produits.

```
info> db.produits.find().pretty()
[
  {
    _id: ObjectId('6825fa2094fed3f00eab7139'),
    nom: 'Macbook Pro',
    fabriquant: 'Apple',
    prix: 17435.99,
    options: [ 'Intel Core i5', 'Retina Display', 'Long life battery' ]
  },
  {
    _id: ObjectId('6825fa9c94fed3f00eab713a'),
    nom: 'DELL',
    fabriquant: 'Dell Technologies, Inc',
    prix: 1143,
    options: [ 'Intel® Core™ Ultra 9 285H', 'SSD', '32 Go' ]
  },
  {
    _id: ObjectId('6825fa9c94fed3f00eab713b'),
    nom: 'Thinkpad X230',
    fabriquant: 'Lenovo',
    prix: 114358.74,
    ultrabook: true,
    options: [ 'Intel Core i5', 'SSD', 'Long life battery' ]
  }
]
info>
```

- Récupérer le premier produit

```
info> db.produits.findOne()
{
  _id: ObjectId('6825fa2094fed3f00eab7139'),
  nom: 'Macbook Pro',
  fabriquant: 'Apple',
  prix: 17435.99,
  options: [ 'Intel Core i5', 'Retina Display', 'Long life battery' ]
}
info> |
```

- Trouver l'id du Thinkpad et faites la requête pour récupérer ce produit avec son id.

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

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

- Récupérer les produits dont le prix est supérieur à 13723 DA

```
info> db.produits.find({ prix: { $gt: 13723 } }).pretty()
[
  {
    _id: ObjectId('6825fa2094fed3f00eab7139'),
    nom: 'Macbook Pro',
    fabriquant: 'Apple',
    prix: 17435.99,
    options: [ 'Intel Core i5', 'Retina Display', 'Long life battery' ]
  },
  {
    _id: ObjectId('6825fa9c94fed3f00eab713b'),
    nom: 'Thinkpad X230',
    fabriquant: 'Lenovo',
    prix: 114358.74,
    ultrabook: true,
    options: [ 'Intel Core i5', 'SSD', 'Long life battery' ]
  }
]
info> |
```

- Récupérer le premier produit ayant le champ ultrabook à true

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

- Récupérer le premier produit dont le nom contient Macbook

```
[info> db.produits.findOne({ nom: { $regex: "Macbook", $options: "i" } })
{
  _id: ObjectId('6825fa2094fed3f00eab7139'),
  nom: 'Macbook Pro',
  fabriquant: 'Apple',
  prix: 17435.99,
  options: [ 'Intel Core i5', 'Retina Display', 'Long life battery' ]
}
info>
```

- Récupérer les produits dont le nom commence par Macbook

```
[info> db.produits.find({ nom: { $regex: "^Macbook", $options: "i" } }).pretty()
[
  {
    _id: ObjectId('6825fa2094fed3f00eab7139'),
    nom: 'Macbook Pro',
    fabriquant: 'Apple',
    prix: 17435.99,
    options: [ 'Intel Core i5', 'Retina Display', 'Long life battery' ]
  }
]
info> |
```

- Supprimer le produit dont le fabricant est Lenovo.

```
[info> db.produits.deleteOne({ fabriquant: "Lenovo" })
{ acknowledged: true, deletedCount: 1 }
info> |
```

- Supprimer le Lenovo X230 en utilisant uniquement son id.

Insérer de nouveau pour qu'on puisse le supprimer d'une autre façon.

```
[info> db.produits.insertOne({
...   nom: "Thinkpad X230",
...   fabriquant: "Lenovo",
...   prix: 114358.74,
...   ultrabook: true,
...   options: ["Intel Core i5", "SSD", "Long life battery"]
... })
{
  acknowledged: true,
  insertedId: ObjectId('6826417394fed3f00eab713c')
}
info> |
```

```

`info> db.produits.findOne({ nom: "Thinkpad X230" })
{
  _id: ObjectId('6826417394fed3f00eab713c'),
  nom: 'Thinkpad X230',
  fabriquant: 'Lenovo',
  prix: 114358.74,
  ultrabook: true,
  options: [ 'Intel Core i5', 'SSD', 'Long life battery' ]
}
info> db.produits.deleteOne({ _id: ObjectId("6826417394fed3f00eab713c") })
{ acknowledged: true, deletedCount: 1 }
info> |

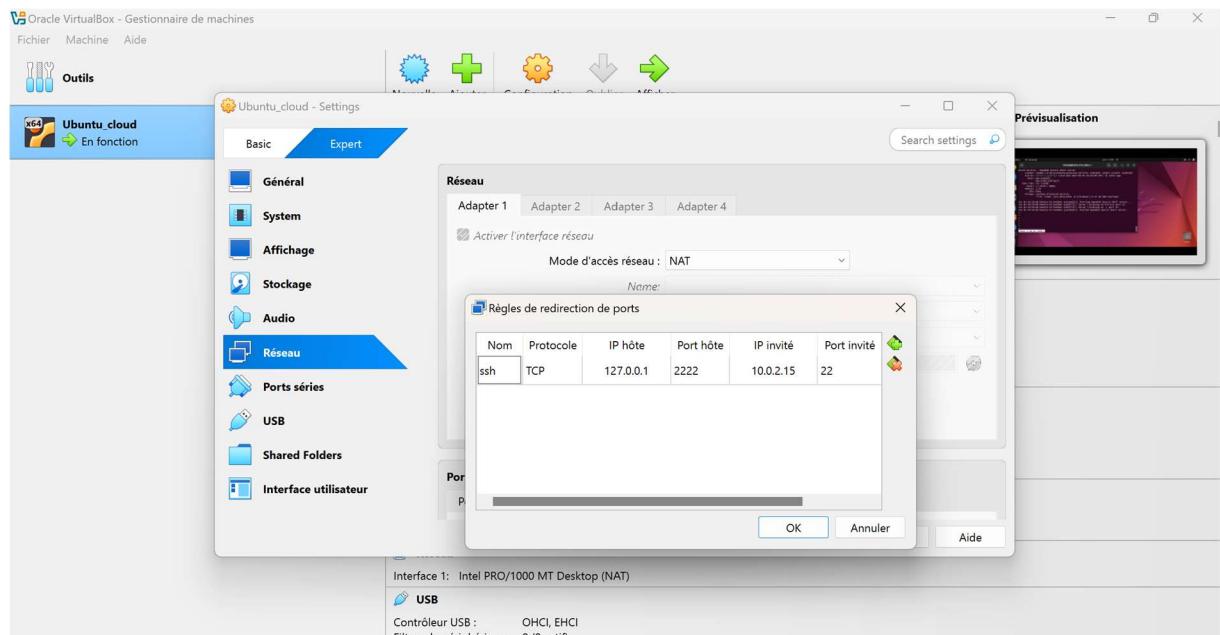
```

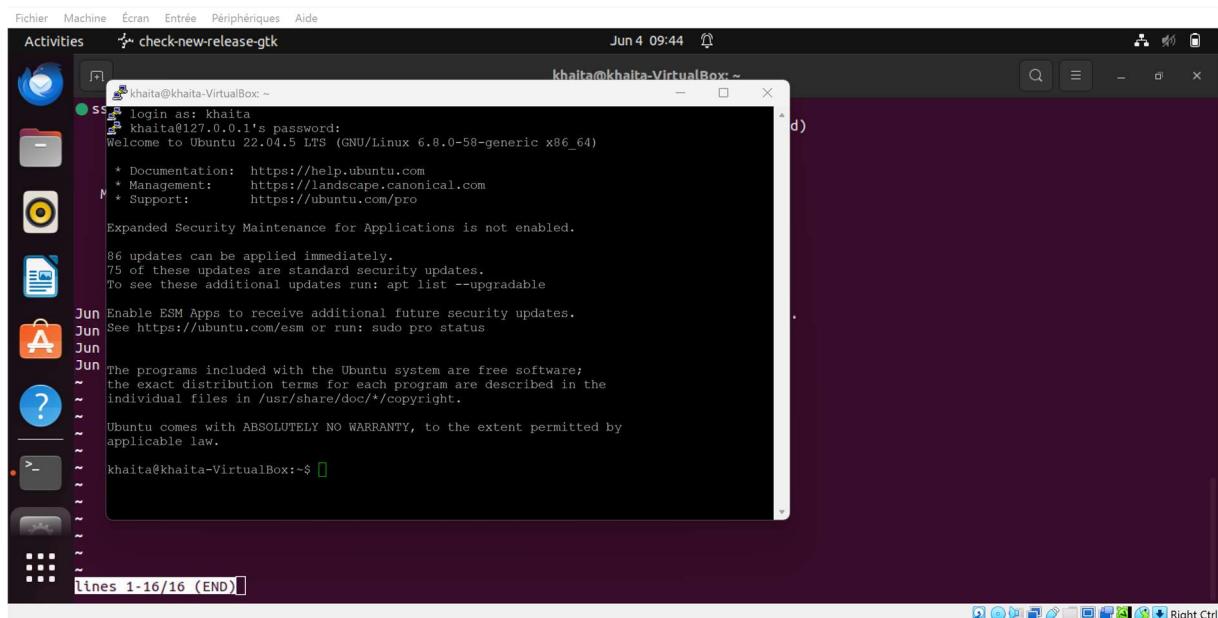
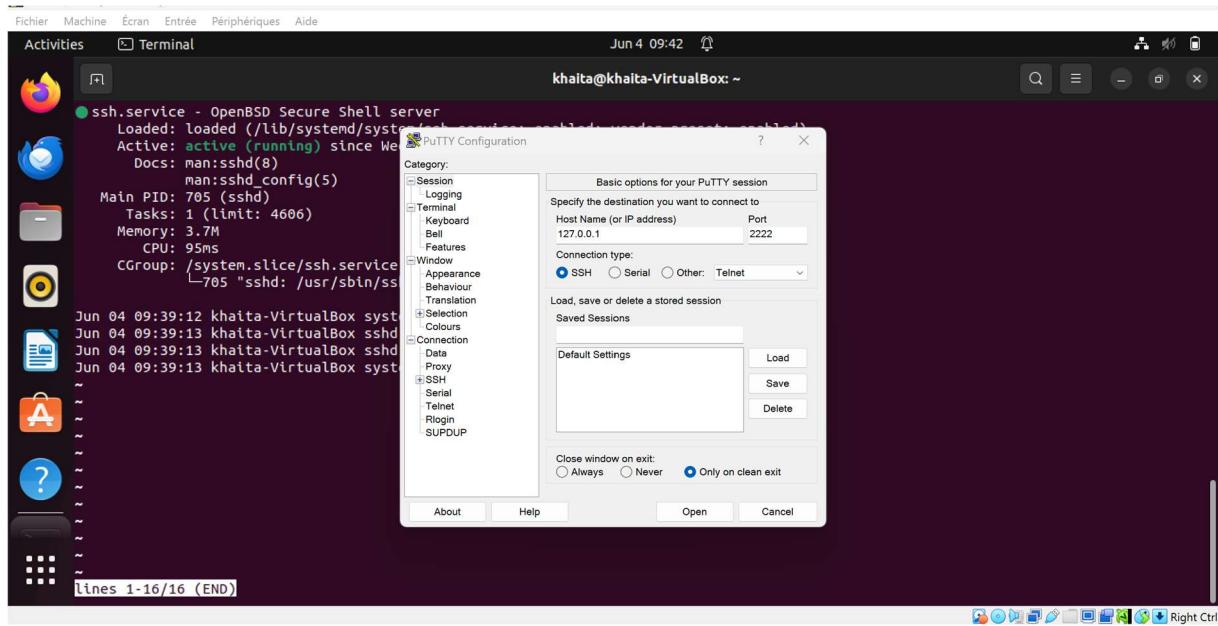
III. Tp N° 2 : Framework HADOOP

Objectif du TP Hadoop (WordCount) :

Ce TP Hadoop a pour objectif de mettre en place un environnement de traitement Big Data en configurant Hadoop en mode pseudo-distribué sur un système Linux. Il s'agit d'installer Hadoop, de préparer le système de fichiers distribué (HDFS), de compiler un programme Java utilisant MapReduce, et de l'exécuter pour compter les occurrences des mots d'un fichier texte. Chaque étape vise à illustrer le fonctionnement des composants principaux d'Hadoop (NameNode, DataNode, YARN) et à montrer comment ils collaborent pour traiter des données de manière parallèle et distribuée. Ce rapport décrit les différentes étapes nécessaires pour atteindre cet objectif.

L'accès à la virtuel machine d'après Putty





1. Creation de nouveau user « hduse »

Un utilisateur nommé 'hduse' a été créé pour isoler l'environnement Hadoop. Les commandes suivantes ont été utilisées :

Sudo add user hduse

Sudo usermod –aG sudo hduse

2. Prérequis et installation

Téléchargement et extraction d'Hadoop :

```
wget https://downloads.apache.org/hadoop/common/hadoop-3.3.6/hadoop-3.3.6.tar.gz  
tar -xvzf hadoop-3.3.6.tar.gz  
mv hadoop-3.3.6 hadoop
```

Installation de Java 8 :

```
sudo apt install openjdk-8-jdk -y
```

Définir JAVA_HOME dans hadoop-env.sh :

```
nano hadoop/etc/hadoop/hadoop-env.sh  
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

3. Configuration de Hadoop

.bashrc :

ajout des variables JAVA_HOME et HADOOP_HOME

hadoop-env.sh :

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

core-site.xml :

```
<configuration>  
  <property>  
    <name>fs.defaultFS</name>  
    <value>hdfs://localhost:9000</value>  
  </property>  
</configuration>
```

hdfs-site.xml :

```
<configuration>  
  <property>  
    <name>dfs.replication</name>  
    <value>1</value>  
  </property>  
</configuration>
```

mapred-site.xml :

```
<configuration>  
  <property>  
    <name>mapreduce.framework.name</name>  
    <value>yarn</value>
```

```
</property>  
</configuration>
```

yarn-site.xml :

```
<configuration>  
  <property>  
    <name>yarn.nodemanager.aux-services</name>  
    <value>mapreduce_shuffle</value>  
  </property>  
</configuration>
```

4. Préparation des répertoires

Création des répertoires de stockage :

```
~/hadoopdata/hdfs/namenode  
~/hadoopdata/hdfs/datanode
```

5. Formatage et démarrage du système Hadoop

Formatage du NameNode : `hdfs namenode -format`

Démarrage des démons : `sbin/start-dfs.sh` et `sbin/start-yarn.sh`

Vérification : `jps`

6. Exécution du programme MapReduce

Création des dossiers et ajout du fichier dans HDFS :

```
hdfs dfs -mkdir /user  
hdfs dfs -mkdir /user/hduse  
hdfs dfs -put poeme.txt /user/hduse
```

Compilation :

```
javac -classpath $(hadoop classpath) -d . WCount.java
```

Création du jar :

```
jar -cvf wcount.jar *
```

Exécution du job :

```
hadoop jar wcount.jar WCount /user/hduse/poeme.txt /user/hduse/output
```

Récupération des résultats :

```
hdfs dfs -cat /user/hduse/output/part-r-00000
```

IV. TD N° 3 : Spark

Objectif du TD Spark :

L'objectif de ce TD est de se familiariser avec les concepts de base de Spark et de la programmation distribuée à travers l'analyse d'un mini-graphe social. Il s'agit de :

- Générer les paires d'amis et leurs listes associées de manière normalisée.
- Appliquer des opérations MapReduce pour calculer les amis communs entre utilisateurs.
- Comprendre et manipuler des RDD dans Spark en Scala.
- Filtrer et formater les résultats pour des paires d'utilisateurs spécifiques.

Ce TD prépare aux notions essentielles pour la réalisation des TPs plus avancés, en consolidant les bases du traitement distribué des graphes sous Spark.

Input :

Nous considérons pour ce TD un mini-graphe dans le fichier ‘soc-LiveJournal1Adj.txt’ :

	<u>Utilisateur</u>	<u>Amis</u>
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

Ici, « **Utilisateur** » est un identifiant entier unique correspondant à un utilisateur, et « **Amis** » est une liste d'identifiants uniques séparés par des virgules, correspondant aux amis de cet utilisateur.

À noter que les amitiés sont mutuelles (c'est-à-dire que les arêtes sont non orientées) : si A est ami avec B, alors B est également ami avec A.

Output :

La sortie doit contenir une ligne par paire d'utilisateurs, sous le format suivant :

<User_A><TAB><User_B><TAB><Liste des amis communs>

où <User_A> et <User_B> sont des identifiants uniques correspondant aux utilisateurs A et B (A et B sont amis).

<Liste des amis communs> est une liste d'identifiants uniques séparés par des virgules correspondant à la liste des amis communs des utilisateurs A et B.

1) *Explication du rôle de la function pairs :*

Elle reçoit en entrée un utilisateur avec sa liste d'amis, et génère :

- Toutes les paires possibles formées entre cet utilisateur et chacun de ses amis.
- Des paires normalisées (toujours dans l'ordre (min, max)) pour éviter les doublons dans le traitement (par exemple, elle crée uniquement (1,2) et jamais (2,1)).
- Pour chaque paire générée, elle associe la liste complète des amis de l'utilisateur courant.

```
//generate ((user1, user2), friendList) for pair counts
def pairs(str: Array[String]) = {
    val users = str(1).split(",")
    val user = str(0)
    val n = users.length
    for(i <- 0 until n) yield {
        val pair = if(user < users(i)) {
            (user,users(i))
        } else {
            (users(i),user)
        }
        (pair, users) }}}
```

2) *On considère le code Spark en Scala (MutualFriends.scala) à compléter :*

```
//Main of our program
---1-- val data = sc.textFile("soc-LiveJournal1Adj.txt")
---2-- val data1= data.map(x => x.split("\t")).filter(li => li.size == 2)
---3-- val pairCounts = data1.flatMap(pairs).reduceByKey((a, b) => a.intersect(b))
---4-- val p1= pairCounts.map { case ((user1, user2), friends) =>
    s"$user1\t$user2\t${friends.mkString(",")}"  }
---5-- p1.saveAsTextFile("output")
```

3) Explication du rôle du code Sparks en Scala suivant :

Rôle global du code :

Sélectionne des paires précises.

- Extrait leurs amis communs sous forme simple.
- Produit un fichier compact avec uniquement ces résultats.

Code avec commentaires :

```
// Initialise une chaîne vide qui servira à accumuler les résultats formatés sous forme texte
var ans=""

// -----1---
// Sélectionne la paire (0,4) dans p1, récupère les amis communs, les transforme en tableau
val p2=p1.map(x=>x.split("\t"))
    .filter(x => (x.size == 3))
    .filter(x=>(x(0)=="0"&&x(1)=="4"))
    .flatMap(x=>x(2).split(","))
    .collect()

// -----2---
// Ajoute à la chaîne ans la ligne formatée des amis communs de la paire (0,4)
ans=ans+"0"+"\t"+"4"+"\t"+p2.mkString(",")+"\n"

// -----3---
// Sélectionne la paire (20,22939), récupère ses amis communs sous forme de tableau
val p3=p1.map(x=>x.split("\t"))
    .filter(x => (x.size == 3))
    .filter(x=>(x(0)=="20"&&x(1)=="22939"))
    .flatMap(x=>x(2).split(","))
    .collect()

// -----4---
// Ajoute à la chaîne ans la ligne formatée des amis communs de la paire (20,22939)
ans=ans+"20"+"\t"+"22939"+"\t"+p3.mkString(",")+"\n"
```

```

// ----5---

// Transforme la chaîne ans en un RDD Spark pour la sauvegarde
val answer=sc.parallelize(Seq(ans))

// ----6---

// Sauvegarde le résultat sous forme de fichier texte dans le répertoire output1
answer.saveAsTextFile("output1")

// Main of our program

// Charge le fichier contenant les données du graphe social (liste d'adjacence)
val data = sc.textFile("soc-LiveJournal1Adj.txt")

// Sépare chaque ligne par tabulation et garde les lignes valides avec 2 colonnes (user et liste
// d'amis)
val data1 = data.map(x => x.split("\t"))
    .filter(li => (li.size == 2))

// Génère des paires (userA, userB) avec la liste des amis, puis calcule les amis communs
// (intersection)
val pairCounts = data1.flatMap(pairs)
    .reduceByKey({ case (param1, param2) => (param1.intersect(param2)) })

// Met en forme le résultat : userA <TAB> userB <TAB> liste des amis communs
val p1 = pairCounts.map({ case ((param1, param2), param3) =>
    param1 + "\t" + param2 + "\t" + param3.mkString(",")
})

// Sauvegarde les résultats globaux dans le dossier output
p1.saveAsTextFile("output")

```

```

// ---- Partie filtrage des paires spécifiques ----

// Initialise une chaîne pour accumuler les résultats ciblés
var ans = ""

// Pour la paire (0,4)
val p2 = p1.map(x => x.split("\t"))
    .filter(x => (x.size == 3))
    .filter(x => (x(0) == "0" && x(1) == "4"))
    .flatMap(x => x(2).split(","))
    .collect()
ans = ans + "0" + "\t" + "4" + "\t" + p2.mkString(",") + "\n"

// Pour la paire (20,22939)
val p3 = p1.map(x => x.split("\t"))
    .filter(x => (x.size == 3))
    .filter(x => (x(0) == "20" && x(1) == "22939"))
    .flatMap(x => x(2).split(","))
    .collect()
ans = ans + "20" + "\t" + "22939" + "\t" + p3.mkString(",") + "\n"

// Pour la paire (1,29826)
val p4 = p1.map(x => x.split("\t"))
    .filter(x => (x.size == 3))
    .filter(x => (x(0) == "1" && x(1) == "29826"))
    .flatMap(x => x(2).split(","))
    .collect()
ans = ans + "1" + "\t" + "29826" + "\t" + p4.mkString(",") + "\n"

// Pour la paire (19272,6222)
val p5 = p1.map(x => x.split("\t"))
    .filter(x => (x.size == 3))

```

```

.filter(x => (x(0) == "19272" && x(1) == "6222"))
.flatMap(x => x(2).split(","))
.collect()

ans = ans + "6222" + "\t" + "19272" + "\t" + p5.mkString(",") + "\n"

// Pour la paire (28041,28056)
val p6 = p1.map(x => x.split("\t"))
.filter(x => (x.size == 3))
.filter(x => (x(0) == "28041" && x(1) == "28056"))
.flatMap(x => x(2).split(","))
.collect()

ans = ans + "28041" + "\t" + "28056" + "\t" + p6.mkString(",") + "\n"

// Transforme la chaîne ans en un RDD
val answer = sc.parallelize(Seq(ans))

// Sauvegarde les résultats filtrés dans output1
answer.saveAsTextFile("output1")

```

V. TP N° 3 : Spark

Objectif du TP Spark :

Ce rapport présente les étapes réalisées dans le cadre d'un TP Spark qui consiste à calculer les amis communs entre deux utilisateurs (Sidi et Mohamed) dans un graphe social. Le TP se déroule selon deux modes d'exécution :

- Exécution interactive via le shell PySpark
- Exécution d'un fichier Python structuré via spark-submit

Le projet final est hébergé sur GitHub :

☞ <https://github.com/KhaitaLoudaa/tp-spark-amis-communs>

1. Exécution interactive dans Spark

Étape 1 : Démarrage de Spark

Nous avons lancé PySpark avec :

pyspark

Étape 2 : Chargement des données

Chargement du fichier texte contenant les données :

```
data = sc.textFile("///home/Khaita/spark-tp/mini_tp_spark.txt")
data.collect()
```

The screenshot shows a terminal window titled "Ubuntu_cloud [En fonction] - Oracle VirtualBox". The terminal session starts with a welcome message from the Spark context, followed by the Python version and configuration details. It then executes a command to read a file named "mini_tp_spark.txt" and collect its contents into a list. The output lists seven names and their friend IDs.

```

Ubuntu_cloud [En fonction] - Oracle VirtualBox
Fichier Machine Écran Entrée Périphériques Aide
Activities Terminal Jun 20 15:46
khaita@khaita-VirtualBox: ~/spark-tp
ses where applicable
Welcome to
version 3.4.1
Using Python version 3.10.12 (main, May 27 2025 17:12:29)
Spark context Web UI available at http://10.0.2.15:4040
Spark context available as 'sc' (master = local[*], app id =
local-1750434094563).
SparkSession available as 'spark'.
>>> data = sc.textFile("file:///home/khaita/spark-tp/mini_tp_
spark.txt")
^[[C
Traceback (most recent call last):
  File "/opt/spark/python/pyspark/context.py", line 378, in s
ignal_handler
    raise KeyboardInterrupt
KeyboardInterrupt
>>> data = sc.textFile("file:///home/khaita/spark-tp/mini_tp_
spark.txt")
>>> data.collect()
[Stage 0:>
['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']
>>> 
```

Étape 3 : Génération des paires d'amis

```
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, set(friends)))
    return pairs
```

```

pair_rdd = data.flatMap(generate_pairs)

for pair, friends in pair_rdd.collect():

    print(f"Paire: {pair} -> Amis: {', '.join(friends)}")

```

```

Fichier Machine Écran Entrée Périphériques Aide
Activities Terminal Jun 20 16:49
khaita@khaita-VirtualBox: ~/spark-tp
>>>
>>>
>>> for line in data.collect():
...     print(line)
...
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
>>> 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])) # (min,
max)
...         pairs.append((pair, set(friends)))
...     return pairs
...
>>> pair_rdd = data.flatMap(generate_pairs)
>>> for pair, friends in pair_rdd.collect():
...     print(f"Paire: {pair} -> Amis: {', '.join(friends)}")
...
Paire: ('1', '2') -> Amis: 4, 3, 2
Paire: ('1', '3') -> Amis: 4, 3, 2

```

Étape 4 : Filtrage des amis communs pour la paire (1, 2)

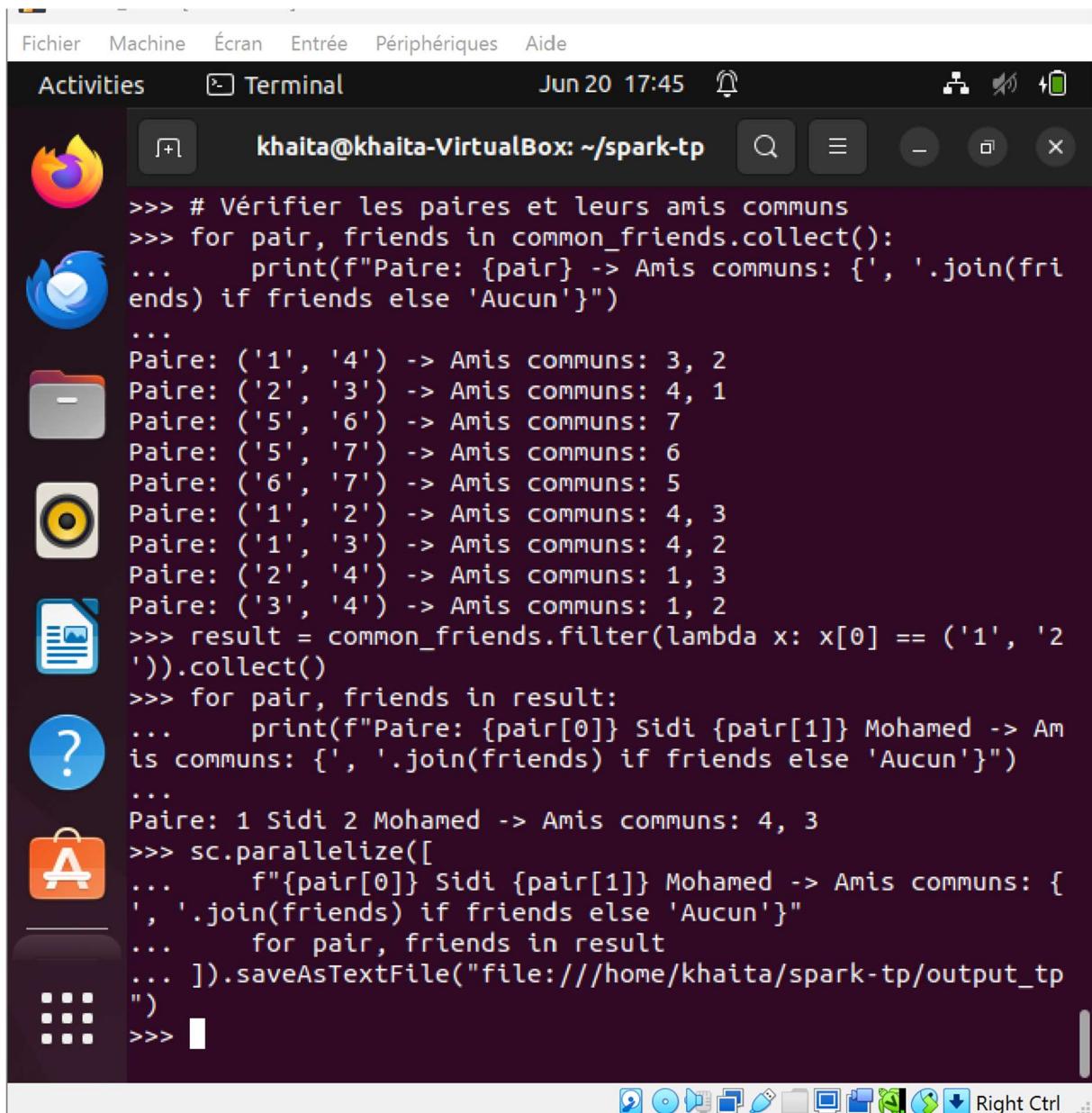
```

result = common_friends.filter(lambda x: x[0] == ('1', '2')).collect()

for pair, friends in result:

    print(f"Paire: {pair[0]} Sidi {pair[1]} Mohamed -> Amis communs: {', '.join(friends) if friends else 'Aucun'}")

```



```
Fichier Machine Écran Entrée Périphériques Aide
Activities Terminal Jun 20 17:45
khaita@khaita-VirtualBox: ~/spark-tp
>>> # Vérifier les paires et leurs amis communs
>>> for pair, friends in common_friends.collect():
...     print(f"Paire: {pair} -> Amis communs: {', '.join(fri
ends) if friends else 'Aucun'}")
...
Paire: ('1', '4') -> Amis communs: 3, 2
Paire: ('2', '3') -> Amis communs: 4, 1
Paire: ('5', '6') -> Amis communs: 7
Paire: ('5', '7') -> Amis communs: 6
Paire: ('6', '7') -> Amis communs: 5
Paire: ('1', '2') -> Amis communs: 4, 3
Paire: ('1', '3') -> Amis communs: 4, 2
Paire: ('2', '4') -> Amis communs: 1, 3
Paire: ('3', '4') -> Amis communs: 1, 2
>>> result = common_friends.filter(lambda x: x[0] == ('1', '2')).collect()
>>> for pair, friends in result:
...     print(f"Paire: {pair[0]} Sidi {pair[1]} Mohamed -> Am
is communs: {', '.join(friends) if friends else 'Aucun'}")
...
Paire: 1 Sidi 2 Mohamed -> Amis communs: 4, 3
>>> sc.parallelize([
...     f"{pair[0]} Sidi {pair[1]} Mohamed -> Amis communs: {',
'.join(friends) if friends else 'Aucun'}"
...     for pair, friends in result
... ]).saveAsTextFile("file:///home/khaita/spark-tp/output_tp
")
>>> 
```

2. Exécution d'un fichier Python structuré via spark-submit

Après validation des codes en mode interactif :

- Création du fichier tp_spark.py contenant l'ensemble des étapes.

```

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, set(friends)))
    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"Pair: {pair[0]} Sidi {pair[1]} Mohamed -> Amis communs: {', '.join(friends) if friends else 'Aucun'}")

sc.parallelize([
    f"Pair: {pair[0]} Sidi {pair[1]} Mohamed -> Amis communs: {', '.join(friends) if friends else 'Aucun'}"
    for pair, friends in result
]).saveAsTextFile("output_tp")

sc.stop()

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location M-U Undo
^X Exit ^R Read File ^V Replace ^U Paste ^J Justify ^L Go To Line M-E Redo
M-A Set Mark M-D Copy

```

- Supprimer output.tp et lancer le programme :

```

Fichier Machine Écran Entrée Périphériques Aide
Activities Terminal Jun 20 20:20 khalta@khalta-VirtualBox: ~/spark-tp
khalta@khalta-VirtualBox:~/spark-tp$ ls
mini_tp_spark.txt  output_tp  tp_spark.py
khalta@khalta-VirtualBox:~/spark-tp$ rm -r output_tp
khalta@khalta-VirtualBox:~/spark-tp$ /opt/spark/bin/spark-submit tp_spark.py
25/06/20 20:19:45 WARN Utils: Your hostname, khalta-VirtualBox resolves to a loopback address: 127.0.1.1; using 10.0.2.15 instead (on interface enp0s3)
25/06/20 20:19:45 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
25/06/20 20:19:46 INFO SparkContext: Running Spark version 3.4.1
25/06/20 20:19:46 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
25/06/20 20:19:47 INFO ResourceUtils: =====
25/06/20 20:19:47 INFO ResourceUtils: No custom resources configured for spark.driver.
25/06/20 20:19:47 INFO ResourceUtils: =====
25/06/20 20:19:47 INFO SparkContext: Submitted application: AmisCommunsTP
25/06/20 20:19:47 INFO ResourceProfile: Default ResourceProfile created, executor resources: Map(cores -> name: cores, amount: 1, script: , vendor: , memory -> name: memory, amount: 1024, script: , vendor: , offHeap -> name: offHeap, amount: 0, script: , vendor: ), task resources: Map(cpu -> name: cpus, amount: 1.0)
25/06/20 20:19:47 INFO ResourceProfile: Limiting resource is cpu
25/06/20 20:19:47 INFO ResourceProfileManager: Added ResourceProfile id: 0
25/06/20 20:19:47 INFO SecurityManager: Changing view acls to: khalta
25/06/20 20:19:47 INFO SecurityManager: Changing modify acls to: khalta
25/06/20 20:19:47 INFO SecurityManager: Changing view acls groups to:
25/06/20 20:19:47 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: khalta; groups with view permissions: EMPTY; users with modify permissions: khalta; groups with modify permissions: EMPTY
25/06/20 20:19:47 INFO Utils: Successfully started service 'sparkDriver' on port 41337.
25/06/20 20:19:47 INFO SparkEnv: Registering MapOutputTracker
25/06/20 20:19:48 INFO SparkEnv: Registering BlockManagerMaster
25/06/20 20:19:48 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology information

```

```

Fichier Machine Écran Entrée Périphériques Aide
Activities Terminal Jun 20 20:21
khaita@khaita-VirtualBox: ~/spark-tp
25/06/20 20:19:47 INFO SecurityManager: Changing acls to: khaita
25/06/20 20:19:47 INFO SecurityManager: Changing view acls groups to:
25/06/20 20:19:47 INFO SecurityManager: Changing modify acls groups to:
25/06/20 20:19:47 INFO SecurityManager: authentication disabled; ui acls disabled; users with view permissions: khaita; groups with view permissions: EMPTY; users with modify permissions: khaita; groups with modify permissions: EMPTY
25/06/20 20:19:47 INFO Utils: Successfully started service 'sparkDriver' on port 41337.
25/06/20 20:19:47 INFO SparkEnv: Registering MapOutputTracker
25/06/20 20:19:48 INFO SparkEnv: Registering BlockManagerMaster
25/06/20 20:19:48 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology information
25/06/20 20:19:48 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
25/06/20 20:19:48 INFO SparkEnv: Registering BlockManagerMasterHeartbeat
25/06/20 20:19:48 INFO DiskBlockManager: Created local directory at /tmp/blockmgr-570556c1-c531-467a-8006-5feadaf6930
25/06/20 20:19:48 INFO MemoryStore: MemoryStore started with capacity 366.3 MiB
25/06/20 20:19:48 INFO SparkEnv: Registering OutputCommitCoordinator
25/06/20 20:19:48 INFO JettyUtils: Start Jetty 0.0.0.0:4040 for SparkUI
25/06/20 20:19:48 INFO Utils: Successfully started service 'SparkUI' on port 4040.
25/06/20 20:19:49 INFO Executor: Starting executor ID driver on host 10.0.2.15
25/06/20 20:19:49 INFO Executor: Starting executor with user classpath (userClassPathFirst = false): ''
25/06/20 20:19:49 INFO Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on port 45249.
25/06/20 20:19:49 INFO NettyBlockTransferService: Server created on 10.0.2.15:45249
25/06/20 20:19:49 INFO BlockManager: Using org.apache.spark.storage.RandomBlockReplicationPolicy for block replication policy
25/06/20 20:19:49 INFO BlockManagerMaster: Registering BlockManager BlockManagerId(driver, 10.0.2.15, 45249, None)
25/06/20 20:19:49 INFO BlockManagerMasterEndpoint: Registering block manager 10.0.2.15:45249 with 366.3 MiB RAM, BlockManagerId(driver, 10.0.2.15, 45249, None)
25/06/20 20:19:49 INFO BlockManagerMaster: Registered BlockManager BlockManagerId(driver, 10.0.2.15, 45249, None)
25/06/20 20:19:49 INFO BlockManager: Initialized BlockManager: BlockManagerId(driver, 10.0.2.15, 45249, None)
Paire: 1 StdIn 2 Mohamed -> Amis communs: 4, 3
khaita@khaita-VirtualBox:~/spark-tp$ 

```

3. Structuration en projet GitHub

- Création du fichier README.md documentant le projet.
- Organisation des fichiers :

SPARK-TP/

```

├── tp_spark.py
├── mini_tp_spark.txt
└── output_tp/
    └── README.md

```

Upload sur GitHub :

KhaitaLoudaa / tp-spark-amis-communs

Type to search

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

tp-spark-amis-communs Public

main 1 Branch 0 Tags

Go to file Add file Code

About

TP Spark pour calcul des amis communs entre deux utilisateurs

Activity 0 stars 0 watching 0 forks

Releases

No releases published Create a new release

README

Voici le lien du projet : <https://github.com/KhaitaLoudaa/tp-spark-amis-communs>

VI. Conclusion

Les travaux réalisés dans ce TP ont permis d'explorer concrètement des technologies majeures du Big Data et de comprendre leurs usages respectifs dans le traitement des données massives.

À travers la manipulation de **MongoDB**, nous avons appris à gérer des données semi-structurées et à appliquer des opérations CRUD dans un environnement NoSQL.

Avec **Hadoop**, nous avons expérimenté le modèle **MapReduce** et les principes du traitement distribué, en apprenant à exécuter des tâches sur des volumes de données simulés.

Enfin, grâce à **Spark**, nous avons découvert une approche plus interactive et rapide du traitement parallèle, en particulier à travers l'analyse de graphes sociaux et le calcul d'amis communs.

Ce TP nous a apporté une richesse d'informations pratiques : il nous a permis de relier théorie et pratique, de comprendre les différences et complémentarités entre ces outils, et de mieux appréhender les défis liés au Big Data.

Les compétences acquises à travers ces exercices nous préparent à manipuler des projets plus complexes et à aborder des cas réels d'analyse de données volumineuses.