

République Islamique de Mauritanie

Ministère de l'Enseignement Supérieur et de la Recherche
Scientifique



Université de Nouakchott Faculté des sciences et
Techniques Département Informatique Master ISDR

Rapport de Travail : MongoDB, Hadoop et Spark

Réalisé par :

**Fatimetou Mohamed Abderrahmane
Bagga C21095**

1. Introduction.....	3
2. Objectifs.....	3
3. Méthodologie et Déroulement.....	4
3.1. Travail 1 : MongoDB.....	4
3.2. Travail 2 : Hadoop.....	5
3.3. Travail 3 : Spark.....	5
4. Résultats.....	5
4.1. MongoDB.....	5
4.2. Hadoop.....	6
4.3. Spark.....	6
5. Conclusion.....	7

1.Introduction

Dans le cadre de l'étude des bases de données NoSQL et des systèmes de traitement de données massives, ce rapport présente trois travaux pratiques distincts.

Le premier porte sur l'installation et l'utilisation de **MongoDB**, une base de données NoSQL populaire, permettant la gestion flexible de données semi-structurées.

Le deuxième travail concerne l'installation et la configuration de **Hadoop**, un framework conçu pour le stockage distribué (via HDFS) et le traitement parallèle de grandes quantités de données grâce au paradigme MapReduce.

Enfin, le troisième travail se focalise sur **Apache Spark**, un moteur de calcul distribué moderne et performant, qui permet d'effectuer des traitements complexes en mémoire, bien plus rapidement que le modèle MapReduce traditionnel.

L'objectif de ces travaux est de comprendre les principes fondamentaux de ces technologies et de les appliquer à des cas pratiques, illustrant ainsi leur utilité dans le domaine du Big Data.

2. Objectifs

Travail 1 : MongoDB

L'objectif est d'apprendre à installer MongoDB sur Windows, à créer une base de données et une collection, puis à effectuer des opérations de type CRUD (création, lecture, mise à jour, suppression) sur les données stockées dans cette base.

Travail 2 : Hadoop

Ce travail vise à installer Hadoop sur une machine virtuelle Ubuntu et à effectuer des opérations de traitement de données en utilisant le système Hadoop en mode pseudo-distribué. Il inclut également l'exécution d'un programme MapReduce pour le comptage de mots dans un fichier texte.

Travail 3 : Spark

L'objectif est d'installer Apache Spark sur Ubuntu et d'apprendre à exécuter des programmes Spark pour traiter des données distribuées.
Ce travail inclut la lecture d'un fichier texte, le calcul des amis communs entre utilisateurs et l'affichage des résultats à l'aide des transformations RDD de Spark.

3. Méthodologie et Déroulement

3.1. Travail 1 : MongoDB

1. Installation de MongoDB sous Windows :

- Téléchargement de MongoDB depuis le lien fourni et extraction de l'archive dans le répertoire `C : \MongoDB`.
- Création des dossiers nécessaires pour le stockage des données (`C : \data\db`).
- Lancement du serveur MongoDB avec la commande `bin\mongod.exe`, puis lancement de la console cliente avec `bin\mongo.exe`.

2. Opérations CRUD sur MongoDB :

- Création d'une base de données nommée `info`.
- Création d'une collection `produite` et insertion de plusieurs documents contenant des informations sur différents produits électroniques (Macbook, DELL, Thinkpad).
- Exécution de différentes requêtes pour :
 - Récupérer tous les produits, un produit spécifique ou des produits avec des critères spécifiques (prix, nom, etc.).
 - Supprimer des produits à partir de critères définis (par fabricant ou par identifiant).

3.2. Travail 2 : Hadoop

1. Installation de Hadoop sur une VM Ubuntu :

- Installation d'Ubuntu sur VirtualBox et configuration du réseau en mode NAT.
- Activation de SSH sur la VM pour pouvoir se connecter à distance.

2. Installation de Java et Hadoop :

- Installation de Java 8, requis pour faire fonctionner Hadoop.
- Téléchargement et extraction de Hadoop sur la machine virtuelle, puis configuration des variables d'environnement.

3. Configuration du système Hadoop :

- Modification des fichiers de configuration (tels que `core-site.xml`, `hdfs-site.xml`, etc.) pour configurer Hadoop en mode pseudo-distribué.
- Formatage du système de fichiers HDFS avec la commande `hdfs namenode -`

format.

4. Démarrage et Test de Hadoop :

- Démarrage des services HDFS et YARN avec les commandes `start-dfs.sh` et `start-yarn.sh`.
- Vérification du bon fonctionnement des services en utilisant `jps` et `hdfs dfs`.

5. Exécution du programme MapReduce :

- Compilation du programme Java pour le comptage de mots (MapReduce).
- Utilisation de WinSCP pour transférer les fichiers nécessaires (programme `.jar` et fichier texte) sur la VM.
- Exécution du programme MapReduce et vérification des résultats.

3.3. Travail 3 : Spark

1. Installation et configuration de Spark sur une VM Ubuntu.
2. Répondre aux questions.
3. Installation de pySpark.
4. Ecriture et exécution d'un script complet `common_friends.py`.
5. Extraction de la réponse pour un cas concret ciblé (Mohamed et Sidi).
6. Création d'un dépôt GitHub avec `README.md`, `friends_common.txt` `common_friends.py`, `resultat_test.txt`.

4. Résultats

4.1. MongoDB :

- La création de la base de données et des documents s'est bien déroulée. Les requêtes de lecture ont permis d'extraire les produits selon différents critères, tels que le nom, le prix et le fabricant.
- La suppression des produits a également été effectuée avec succès en utilisant différents identifiants et critères de recherche.

4.2. Hadoop :

- L'installation de Hadoop et de Java sur la VM Ubuntu a été réalisée sans difficulté majeure.
- Après avoir configuré Hadoop et démarré les services, le test de la commande `jps` a confirmé que tous les services nécessaires (NameNode, DataNode, ResourceManager, NodeManager) étaient en cours d'exécution.
- Le programme MapReduce pour le comptage de mots a fonctionné comme prévu, et les résultats ont été correctement stockés et affichés dans HDFS.

4.3. Spark :

4.3.1. Répondre aux questions:

A. Expliquer le rôle de la fonction `pairs`

La fonction `pairs` a pour rôle de :

- Générer des couples d'utilisateurs, pour représenter chaque amitié sous forme de paire triée (ex. (1,2) au lieu de (2,1)) afin d'éviter les doublons inversés.
- Pour chaque ligne (`user`, `liste_d_amis`), elle crée pour chaque ami `friend` un couple (`user`, `friend`) trié.
- Elle associe à ce couple la liste complète des amis de `user` (pour pouvoir faire l'intersection et trouver les amis communs).

B. Compléter le code Spark (`MutualFriends.scala`)

```
//Main of our program
val data = sc.textFile("soc-LiveJournal1Adj.txt")
val data1 = data.map(x => x.split("\t")).filter(li => (li.size == 2))

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)
  }
}
```

```
val pairCounts = data1.flatMap(pairs)
    .reduceByKey((l1, l2) => l1.intersect(l2))

val p1 = pairCounts.map({ case ((u1, u2), mutualFriends) =>
    u1 + "\t" + u2 + "\t" + mutualFriends.mkString(",")
})

p1.saveAsTextFile("output")
```

C. Expliquer le second code Spark (extraction output1)

Le code a pour objectif de :

Parcourir le résultat p1 (la liste des amis communs pour chaque paire d'amis).

Sélectionner des couples précis, par exemple (0,4) et (20,22939).

Extraire leur liste d'amis communs et reformater pour l'enregistrer dans un fichier plus ciblé output1.

4.3.2. Résultats du mini-projet Spark : recherche des amis communs

- L'installation de Apache Spark et PySpark sur la VM Ubuntu a été réalisée sans difficulté majeure.
- Le fichier friends_common.txt a été préparé avec les données du graphe social représentant les utilisateurs et leurs amis.
- Le script PySpark common_friends.py a été développé pour :
Charger le fichier texte en RDD.
Transformer chaque ligne en tuple (utilisateur, liste des amis).
Générer toutes les paires triées (min(user_id, friend_id), max(user_id, friend_id)).
Grouper les paires et calculer l'intersection des ensembles d'amis pour obtenir les amis communs.
Filtrer spécifiquement la paire (1,2).
L'exécution du script via spark-submit common_friends.py s'est déroulée sans erreur et a produit le résultat attendu :
Enfin, le projet a été initialisé avec Git, puis poussé sur GitHub.

5. Conclusion

La mise en place conjointe de MongoDB, Hadoop et Spark dans ce projet a permis de démontrer la complémentarité de ces technologies dans un environnement Big Data. MongoDB a apporté une solution flexible pour le stockage de données semi-structurées, tandis

que Hadoop a assuré la répartition et la fiabilité du stockage distribué via HDFS. Enfin, Spark s'est révélé essentiel pour le traitement rapide des données et l'extraction d'informations pertinentes, en particulier dans l'implémentation du projet sur les amis communs.

Cette expérience confirme l'importance d'un écosystème technologique adapté pour relever les défis liés au volume, à la variété et à la vélocité des données, et ouvre la voie à des analyses plus avancées dans des contextes réels.

Annexes :

- **Annexe 1 : Commandes MongoDB utilisées**

```
C:\MongoDB>bin\mongo.exe
MongoDB shell version: 2.6.4
connecting to: test
> use info
switched to db info
> db
info
```

```
> db.produits.insert({
...   nom: "Macbook Pro",
...   fabricant: "Apple",
...   prix: 17435.99,
...   options: ["Intel Core i5", "Retina Display", "Long life battery"]
... })
WriteResult({ "nInserted" : 1 })
```

```
> db.produits.insert([ { nom: "DELL", fabricant: "Dell Technologies, Inc", prix: 1143, options: ["Intel(R) Core(tm) Ultra 9 285H", "SSD", "32 Go"] },
{ nom: "Thinkpad X230", fabricant: "Lenovo", prix: 114358.74, ultrabook: true, options: ["Intel Core i5", "SSD", "Long life battery"] } ])
BulkWriteResult({
  "writeErrors" : [ ],
  "writeConcernErrors" : [ ],
  "nInserted" : 2,
  "nUpserted" : 0,
  "nMatched" : 0,
  "nModified" : 0,
  "nRemoved" : 0,
  "upserted" : [ ]
})
```



```

> db.produits.find()
{ "_id" : ObjectId("6824cd49306e261ea80e9762"), "nom" : "Macbook Pro", "fabriquant" : "Apple", "prix" : 17435.99, "options" : [ "Intel Core i5", "Retina Display", "Long life battery" ] }
{ "_id" : ObjectId("6824cdde306e261ea80e9763"), "nom" : "DELL", "fabriquant" : "Dell Technologies, Inc", "prix" : 1143, "options" : [ "Intel® Core™ Ultra 9 285H", "SSD", "32 Go" ] }
{ "_id" : ObjectId("6824cdde306e261ea80e9764"), "nom" : "Thinkpad X230", "fabriquant" : "Lenovo", "prix" : 114358.74, "ultrabook" : true, "options" : [ "Intel Core i5", "Long life battery" ] }
{ "_id" : ObjectId("684e154224c9d9ba422524d7"), "nom" : "Macbook Pro", "fabriquant" : "Apple", "prix" : 17435.99, "options" : [ "Intel Core i5", "Retina Display", "Long life battery" ] }
{ "_id" : ObjectId("684e163d24c9d9ba422524d8"), "nom" : "DELL", "fabriquant" : "Dell Technologies, Inc", "prix" : 1143, "options" : [ "Intel(R) Core(tm) Ultra 9 285H", "SSD", "32 Go" ] }
{ "_id" : ObjectId("684e163d24c9d9ba422524d9"), "nom" : "Thinkpad X230", "fabriquant" : "Lenovo", "prix" : 114358.74, "ultrabook" : true, "options" : [ "Intel Core i5", "Long life battery" ] }
> db.produits.findOne()
{
  "_id" : ObjectId("6824cd49306e261ea80e9762"),
  "nom" : "Macbook Pro",
  "fabriquant" : "Apple",
  "prix" : 17435.99,
  "options" : [
    "Intel Core i5",
    "Retina Display",
    "Long life battery"
  ]
}
> db.produits.findOne({ nom: "Thinkpad X230" })._id
ObjectId("6824cdde306e261ea80e9764")
> db.produits.findOne({ _id: ObjectId("6824cdde306e261ea80e9764") })
{
  "_id" : ObjectId("6824cdde306e261ea80e9764"),
  "nom" : "Thinkpad X230",
  "fabriquant" : "Lenovo",
  "prix" : 114358.74,
  "ultrabook" : true,
  "options" : [
    "Intel Core i5",
    "SSD",
    "Long life battery"
  ]
}

```

Activer Windows
Accédez aux paramètres pour activer Windows.

```

> db.produits.find({ prix: { $gt: 13723 } })
{ "_id" : ObjectId("6824cd49306e261ea80e9762"), "nom" : "Macbook Pro", "fabriquant" : "Apple", "prix" : 17435.99, "options" : [ "Intel Core i5", "Retina Display", "Long life battery" ] }
{ "_id" : ObjectId("6824cdde306e261ea80e9764"), "nom" : "Thinkpad X230", "fabriquant" : "Lenovo", "prix" : 114358.74, "ultrabook" : true, "options" : [ "Intel Core i5", "Long life battery" ] }
{ "_id" : ObjectId("684e154224c9d9ba422524d7"), "nom" : "Macbook Pro", "fabriquant" : "Apple", "prix" : 17435.99, "options" : [ "Intel Core i5", "Retina Display", "Long life battery" ] }
{ "_id" : ObjectId("684e163d24c9d9ba422524d9"), "nom" : "Thinkpad X230", "fabriquant" : "Lenovo", "prix" : 114358.74, "ultrabook" : true, "options" : [ "Intel Core i5", "Long life battery" ] }
> db.produits.findOne({ ultrabook: true })
{
  "_id" : ObjectId("6824cdde306e261ea80e9764"),
  "nom" : "Thinkpad X230",
  "fabriquant" : "Lenovo",
  "prix" : 114358.74,
  "ultrabook" : true,
  "options" : [
    "Intel Core i5",
    "SSD",
    "Long life battery"
  ]
}
> db.produits.findOne({ nom: /Macbook/i })
{
  "_id" : ObjectId("6824cd49306e261ea80e9762"),
  "nom" : "Macbook Pro",
  "fabriquant" : "Apple",
  "prix" : 17435.99,
  "options" : [
    "Intel Core i5",
    "Retina Display",
    "Long life battery"
  ]
}
> db.produits.find({ nom: /^Macbook/ })
{ "_id" : ObjectId("6824cd49306e261ea80e9762"), "nom" : "Macbook Pro", "fabriquant" : "Apple", "prix" : 17435.99, "options" : [ "Intel Core i5", "Retina Display", "Long life battery" ] }
{ "_id" : ObjectId("684e154224c9d9ba422524d7"), "nom" : "Macbook Pro", "fabriquant" : "Apple", "prix" : 17435.99, "options" : [ "Intel Core i5", "Retina Display", "Long life battery" ] }

```

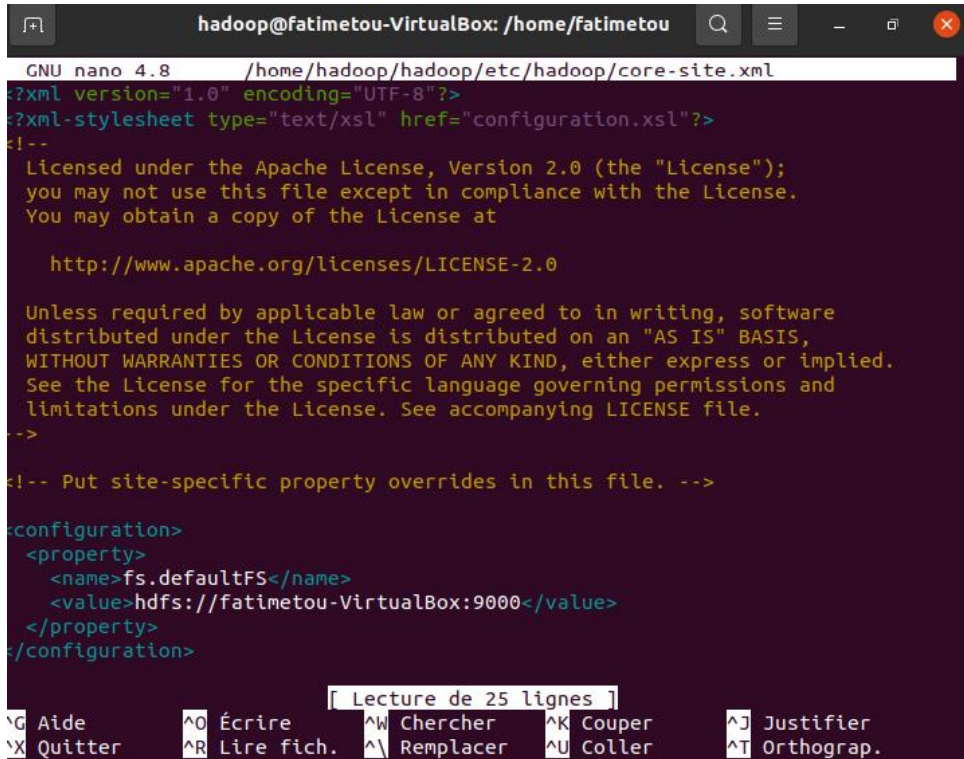
Activer Windows
Accédez aux paramètres pour activer Windows.

```

> db.produits.remove({ fabriquant: "Lenovo" })
WriteResult({ "nRemoved" : 2 })
> db.produits.remove({ _id: ObjectId("6824cdde306e261ea80e9764") })
WriteResult({ "nRemoved" : 0 })

```

- **Annexe 2 : Fichiers de configuration Hadoop**

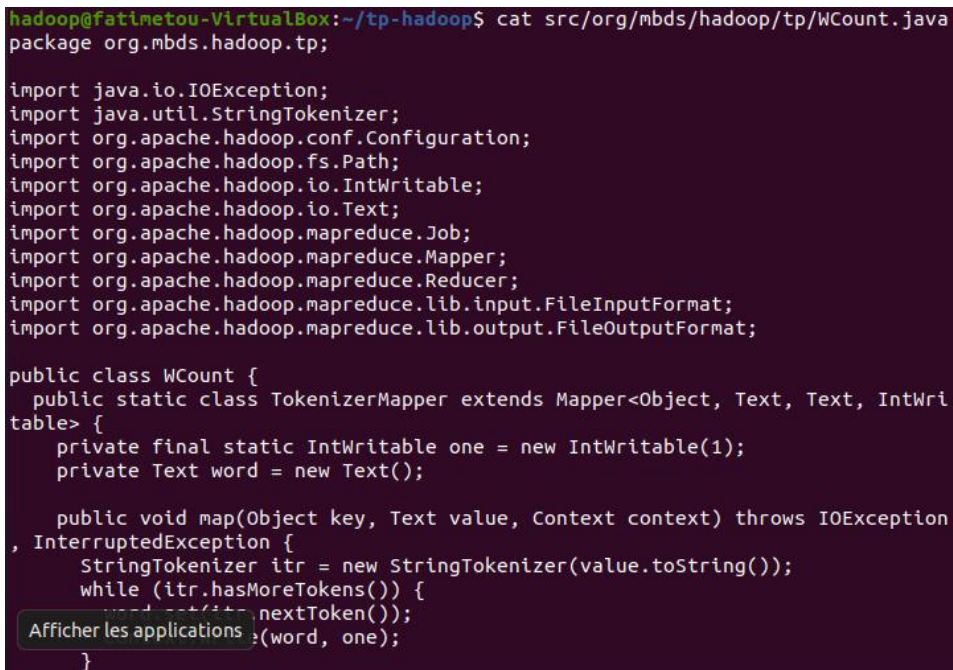


```
GNU nano 4.8 /home/hadoop/hadoop/etc/hadoop/core-site.xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->
<!-- Put site-specific property overrides in this file. -->
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://fatimetou-VirtualBox:9000</value>
  </property>
</configuration>
```

- **Annexe 3 : Code du programme MapReduce**



```
hadoop@fatimetou-VirtualBox:~/tp-hadoop$ cat src/org/mbds/hadoop/tp/WCount.java
package org.mbds.hadoop.tp;

import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WCount {
    public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable> {
        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();

        public void map(Object key, Text value, Context context) throws IOException, InterruptedException {
            StringTokenizer itr = new StringTokenizer(value.toString());
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, one);
            }
        }
    }
}
```

```

IntWritable> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values, Context context)
    throws IOException, InterruptedException {
        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        result.set(sum);
        context.write(key, result);
    }
}

public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
}

```

- **Annexe 4: Résultat d'exécution du programme common_friends.py**

```

25/06/29 20:01:55 INFO TaskSchedulerImpl: Killing all running tasks in stage 2:
Stage finished
25/06/29 20:01:55 INFO DAGScheduler: Job 1 finished: collect at /home/fatimetou
projet_spark/common_friends.py:35, took 5,032902 s
25/06/29 20:01:55 INFO TaskSchedulerImpl: Job 1 finished: collect at /home/fatimetou
projet_spark/common_friends.py:35, took 5,032902 s
25/06/29 20:01:55 INFO SparkContext: SparkContext is stopping with exitCode 0.
25/06/29 20:01:56 INFO SparkUI: Stopped Spark web UI at http://10.0.2.15:4040
25/06/29 20:01:56 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEn

```