Rapport développement d'application map reduce

Introduction

Le but de cet atelier est de développer une application Big Data capable de traiter les données issues de la plateforme GDELT. Ces données, qui représentent des événements mondiaux, doivent être analysées pour :

- Calculer le nombre total d'événements recensés par pays.
- Fournir une répartition des événements par type d'événement pour chaque pays.

L'application sera déployée sur un cluster Hadoop/HDFS de 10 nœuds, configuré à l'aide de Docker.

Objectifs

1. Télécharger et comprendre les données issues de la plateforme GDELT.
2. Configurer un cluster Hadoop de 10 nœuds avec Docker pour simuler un environnement distribué.
3. Développer une application MapReduce pour analyser les données.
4. Exécuter et tester l'application sur le cluster Hadoop.
5. Généralisation
6. Difficultés rencontrées et conclusion

Étape 1 : Compréhension des données GDELT

Description des données

Les données GDELT se trouvent à GDELT Data et sont au format CSV. Chaque ligne représente un événement, avec plus de 50 colonnes. Voici les colonnes importantes pour notre projet :

- **EventCode** (colonne 26) : Type d'événement (par exemple, 10 = protestations, 20 = conflits armés).

- **ActionGeo_CountryCode** (colonne 51) : Le nom du pays où l'événement a eu lieu.

Action :

1. Télécharge un fichier GDELT récent (exemple : `20231201.export.CSV`).
2. Étudie la structure des colonnes en ouvrant le fichier dans un tableur ou un éditeur de texte.

Étape 2 : Configuration du cluster Hadoop (10 nœuds)

Utilisation de Docker

Nous utiliserons l'image Docker **BigDataEurope Hadoop** pour configurer un cluster avec :

- 1 **Namenode** : Gère le système de fichiers HDFS.
- 10 **Datanodes** : Stockent les données et exécutent les calculs.

Commandes pour configurer le cluster :

Télécharger l'image docker uploadée sur dockerhub:

```shell
docker pull liliasfaxi/spark-hadoop:hv-2.7.2
```

Créer les 11 contenaires à partir de l'image téléchargée. Pour cela:

1. Créer un réseau qui permettra de relier les 11 contenaires:

```shell
docker network create --driver=bridge hadoop
```

2. Lancer le namenode

```shell
docker run -itd --net=hadoop -p 50070:50070 -p 8088:8088 -p 707 --name hadoop-master --hostname hadoop-master liliasfaxi/spark-
hadoop:hv-2.7.2
```

3. Configurer le nombre de datanodes

SHELL

```
 vi $HADOOP_HOME/etc/hadoop/slaves

hadoop-slave1
hadoop-slave2
hadoop-slave3
hadoop-slave4
hadoop-slave5
hadoop-slave6
hadoop-slave7
hadoop-slave8
hadoop-slave9
hadoop-slave10
```

4. Lancer les 10 Datanodes :

SHELL

```
docker run -itd -p 8040:8042 --net=hadoop --name hadoop-slave1 --hostname hadoop-slave1 liliasfaxi/spark-hadoop:hv-2.7.2
1d9c4ff20cd16cadcbd60b657f94e589b2356ccadc3a308e4a863548c62e064f

docker run -itd -p 8041:8042 --net=hadoop --name hadoop-slave2 --hostname hadoop-slave2 liliasfaxi/spark-hadoop:hv-2.7.2
a21ec4aad3b2425eb91ad38916a94a0a38f31bf41d396ca024d8b6de75eedd87

docker run -itd -p 8042:8042 --net=hadoop --name hadoop-slave3 --hostname hadoop-slave3 liliasfaxi/spark-hadoop:hv-2.7.2
1fef704cca3de702a991a8b19ab1978ae0c583488793c727ba19e351d0a29790

docker run -itd -p 8043:8042 --net=hadoop --name hadoop-slave4 --hostname hadoop-slave4 liliasfaxi/spark-hadoop:hv-2.7.2
9cfb01f90de72ebbe1dda9650fc95b0471cef9f1d4bab0129a479b3c23a8a941

docker run -itd -p 8044:8042 --net=hadoop --name hadoop-slave5 --hostname hadoop-slave5 liliasfaxi/spark-hadoop:hv-2.7.2
eeec221ec4f1f0c22a73c57c09c58d9261d544eae38f92246fbd48e36458af8f

docker run -itd -p 8045:8042 --net=hadoop --name hadoop-slave6 --hostname hadoop-slave6 liliasfaxi/spark-hadoop:hv-2.7.2
f2d84ab62a4d723bf1fe9d983843498cc90f59680d8e59bfe855de9313560912

docker run -itd -p 8046:8042 --net=hadoop --name hadoop-slave7 --hostname hadoop-slave7 liliasfaxi/spark-hadoop:hv-2.7.2
2698706b5e8d7e5e0c86c6363f5af20354048c155d3b0785c75241645d71e6e3

docker run -itd -p 8047:8042 --net=hadoop --name hadoop-slave8 --hostname hadoop-slave8 liliasfaxi/spark-hadoop:hv-2.7.2
9404f677298ab0cb57477318d550961caf65a24a0e52f71e655295bc2aac3ead

docker run -itd -p 8048:8042 --net=hadoop --name hadoop-slave9 --hostname hadoop-slave9 liliasfaxi/spark-hadoop:hv-2.7.2
3d12b09e4a361a9051a3d7d08538a157aa70b7fa19be2b283ce483a33cf6aa43

docker run -itd -p 8049:8042 --net=hadoop --name hadoop-slave10 --hostname hadoop-slave10 liliasfaxi/spark-hadoop:hv-2.7.2
b4d382ec0e161c0d849db57147e024bbe34e3a36a3395c1b0c4f4b0616613274
```

5. Entrer dans le contenaire master pour commencer à l'utiliser.

SHELL

```
docker exec -it hadoop-master bash
root@hadoop-master:~#
```

6. Lancer le script start-hadoop.sh

SHELL

```
./start-hadoop.sh

Starting namenodes on [hadoop-master]
hadoop-master: Warning: Permanently added 'hadoop-master,172.19.0.2' (ECDSA) to the list of known hosts.
hadoop-master: namenode running as process 3619. Stop it first.
hadoop-slave2: Warning: Permanently added 'hadoop-slave2,172.19.0.10' (ECDSA) to the list of known hosts.
hadoop-slave1: Warning: Permanently added 'hadoop-slave1,172.19.0.11' (ECDSA) to the list of known hosts.
hadoop-slave7: Warning: Permanently added 'hadoop-slave7,172.19.0.16' (ECDSA) to the list of known hosts.
hadoop-slave4: Warning: Permanently added 'hadoop-slave4,172.19.0.13' (ECDSA) to the list of known hosts.
hadoop-slave5: Warning: Permanently added 'hadoop-slave5,172.19.0.14' (ECDSA) to the list of known hosts.
hadoop-slave3: Warning: Permanently added 'hadoop-slave3,172.19.0.12' (ECDSA) to the list of known hosts.
hadoop-slave6: Warning: Permanently added 'hadoop-slave6,172.19.0.15' (ECDSA) to the list of known hosts.
hadoop-slave8: Warning: Permanently added 'hadoop-slave8,172.19.0.17' (ECDSA) to the list of known hosts.
hadoop-slave9: Warning: Permanently added 'hadoop-slave9,172.19.0.18' (ECDSA) to the list of known hosts.
hadoop-slave10: Warning: Permanently added 'hadoop-slave10,172.19.0.19' (ECDSA) to the list of known hosts.
```

8. Vérifie que le cluster est actif :

- Accède à l'interface web HDFS : http://localhost:50070.

Hadoop | Overview | Datanodes | Datanode Volume Failures | Snapshot | Startup Progress | Utilities

Overview 'hadoop-master:9000' (active)

Started:	Sat Dec 28 17:40:14 UTC 2024
Version:	2.7.2, rUnknown
Compiled:	2016-05-27T18:05Z by root from Unknown
Cluster ID:	CID-b721bea8-93cb-45f0-9023-dff705808b00
Block Pool ID:	BP-195763961-172.17.0.3-1550840521902

Summary

Security is off.

Safemode is off.

1138 files and directories, 804 blocks = 1942 total filesystem object(s).

Heap Memory used 78.34 MB of 353.5 MB Heap Memory. Max Heap Memory is 889 MB.

Non Heap Memory used 57.75 MB of 58.81 MB Commited Non Heap Memory. Max Non Heap Memory is -1 B.

• Vérifie que les 10 nœuds sont opérationnels:
http://localhost:8040
http://localhost:8041
http://localhost:8042
http://localhost:8043
http://localhost:8044
http://localhost:8045
http://localhost:8046
http://localhost:8047
http://localhost:8048
http://localhost:8049

Logged in as: dr.who

- ResourceManager
- NodeManager
  - Node Information
  - List of Applications
  - List of Containers
- Tools

NodeManager information

Total Vmem allocated for Containers	16.80 GB
Vmem enforcement enabled	false
Total Pmem allocated for Container	8 GB
Pmem enforcement enabled	false
Total VCores allocated for Containers	8
NodeHealthyStatus	true
LastNodeHealthTime	Sat Dec 28 22:46:16 UTC 2024
NodeHealthReport	
Node Manager Version:	2.7.2 from Unknown by root source checksum c63f7cc71b8f63249e35126f0f7492d on 2016-05-27T18:16Z
Hadoop Version:	2.7.2 from Unknown by root source checksum d0fda26633fa762bff87ec759ebe689c on 2016-05-27T18:05Z

Étape 3 : Développement de l'application MapReduce

L'application analysera les données GDELT et produira pour chaque pays

- Le nombre total d'événements recensés.
- La répartition des événements par type.

Ouvrir le chier pom.xml, et ajouter les dépendances suivantes pour Hadoop, HDFS et Map Reduce:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0   http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>hadoop.mapreduce</groupId>
    <artifactId>CountTypeEvent</artifactId>
    <version>1.0</version>

    <properties>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <dependencies>

       <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-common</artifactId>
            <version>2.7.2</version>
        </dependency>

        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-mapreduce-client-core</artifactId>
            <version>2.7.2</version>
        </dependency>

        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-hdfs</artifactId>
            <version>2.7.2</version>
        </dependency>

        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-mapreduce-client-common</artifactId>
            <version>2.7.2</version>
        </dependency>

    </dependencies>

</project>
```

Créer un package tn.insat.tp sous le répertoire src/main/java puis créer la classe `GDELTMapper`

Le mapper lit chaque ligne du fichier CSV, extrait le code pays et le type d'événement, puis les émet sous la forme :
(Pays:Type, 1)

Exemple de code (Java) :

```java
package tn.insat;

import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import java.io.IOException;
public class GDELTMapper extends Mapper<Object, Text, Text, IntWritable> {
    private Text countryEvent = new Text();
    private final static IntWritable one = new IntWritable(1);
    public void map(Object key, Text value, Context context) throws IOException, InterruptedException {
        String[] fields = value.toString().split("\\t"); // Les données GDELT sont tabulées
        if (fields.length > 51 && !fields[51].isEmpty()) {
            String country = fields[51];
            String eventType = fields[26];
            countryEvent.set(country + "-" + eventType);
            context.write(countryEvent, one);
        }
    }
}
```

Créer la classe `GDELTReducer`

Le reducer agrège les occurrences pour chaque clé (Pays:Type) et produit le total :

(Pays:Type, Total)

Exemple de code (Java) :

```java
package tn.insat;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

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

Créer la classe `GDELTDriver`

Le driver configure le job Hadoop, spécifie le mapper/reducer et lance le traitement.

Exemple de code (Java) :

```java
package tn.insat;

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.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import tn.insat.GDELTMapper;
import tn.insat.GDELTReducer;

public class GDELTDriver {
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "GDELT Event Analysis");
        job.setJarByClass(GDELTDriver.class);
        job.setMapperClass(GDELTMapper.class);
        job.setReducerClass(GDELTReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        // Utilisez TextInputFormat pour les fichiers texte
        job.setInputFormatClass(TextInputFormat.class);

        // Définir les chemins d'entrée et de sortie
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        // Lancer le job
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Étape 4 : Exécution du programme

Créer une configuration Maven avec la ligne de commande: **package install**

Lancer la configuration. Un fichier CountTypeEvent-1.0.jar sera créé dans le répertoire target du projet

1. Copier le fichier jar créé dans le contenaire master.

```shell
docker cp target/CountTypeEvent-1.0.jar hadoop-master:/root
```

Charge les données GDELT sur HDFS :

```shell
docker exec -it hadoop-master bash

hdfs dfs -mkdir /input

hdfs dfs -put /path/to/20231201.export.CSV /input/
```

Lance le job Hadoop :

```shell
hadoop jar CountTypeEvent-1.0.jar tn.insat.GDELTDriver /input /output
```

Vérifie les résultats :

```shell
hdfs dfs -tail /output/part-r-00000
```

Résultat attendu

Pour chaque pays, le rapport inclura des lignes comme :

```shell
US-023  25

US-045  100

FR-060  19

FR-095  13

CA-079  28

CA-035  26
```

Étape 5 : Généralisation

Télécharger le fichier html ayant l'ensemble des fichiers csv et extraire les liens fichiers zip dans un fichier .txt

On crée un script download_html.sh

```shell
#!/bin/bash

# URL de base pour les fichiers
base_url="http://data.gdeltproject.org/events/"

# Télécharger le contenu de la page web contenant les fichiers
wget -q -O index.html ${base_url}

# Extraire les liens des fichiers ZIP depuis la page HTML
grep -oP 'HREF="\K[^"]+\.zip' index.html > files.txt
```

Télécharger les fichier csv via le files.txt en créant un script download_csv.sh

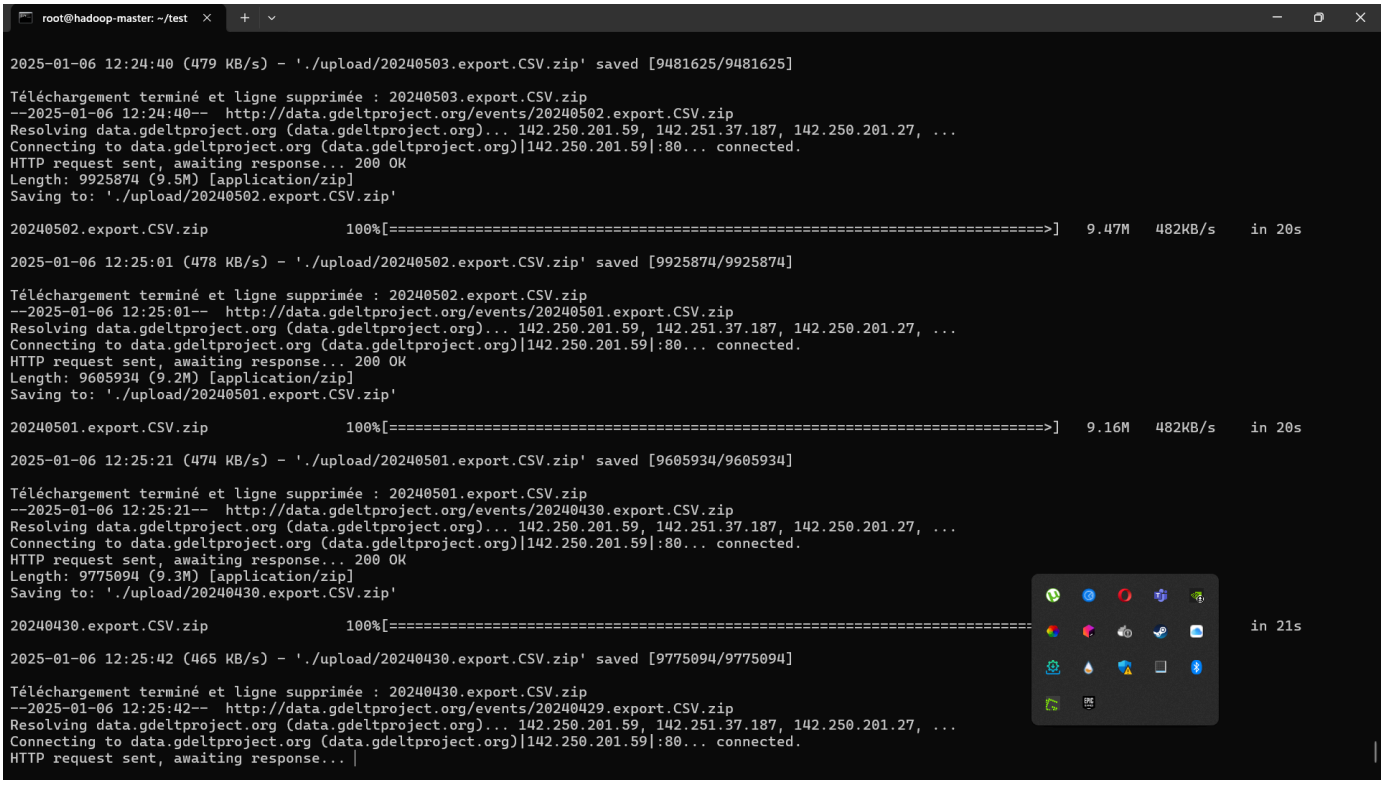
SHELL

```shell
#!/bin/bash

base_url="http://data.gdeltproject.org/events/"
files="files.txt"

# Télécharger chaque fichier ZIP et supprimer la ligne correspondante après téléchargement
while read -r file; do
    # Télécharger le fichier
    if wget "${base_url}${file}" -P ./upload; then
        # Supprimer la ligne correspondante du fichier si le téléchargement a réussi
        sed -i "/^${file}$/d" "$files"
        echo "Téléchargement terminé et ligne supprimée : $file"
    else
        echo "Échec du téléchargement : $file"
    fi
done < "$files"
```

Exécuter le script `./download_csv.sh`



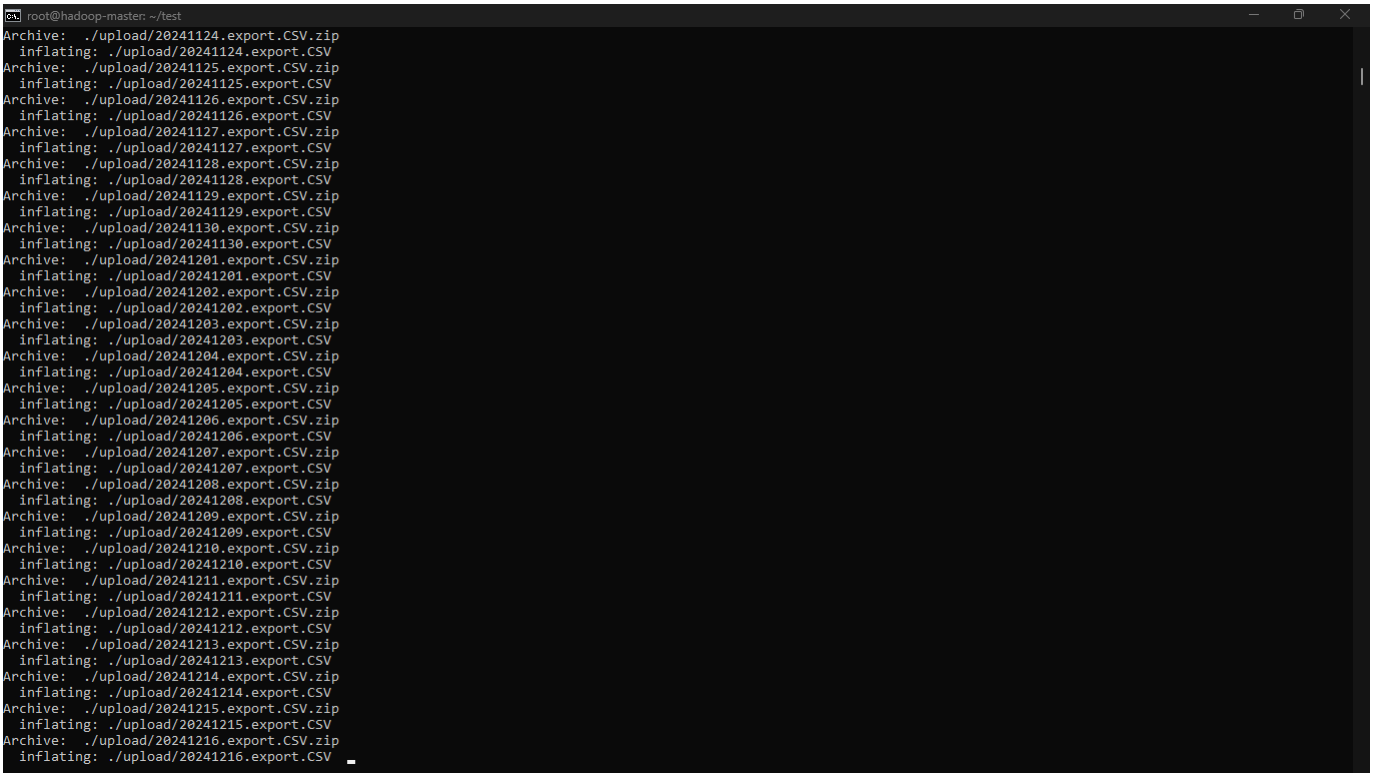
`unziper` les fichiers dans `upload` en créant un script unzip.sh

SHELL

```shell
#!/bin/bash

# unzip
for file in ./upload/*.zip; do
    unzip "$file" -d ./upload
done
```

Exécuter le script `./unzip.sh`

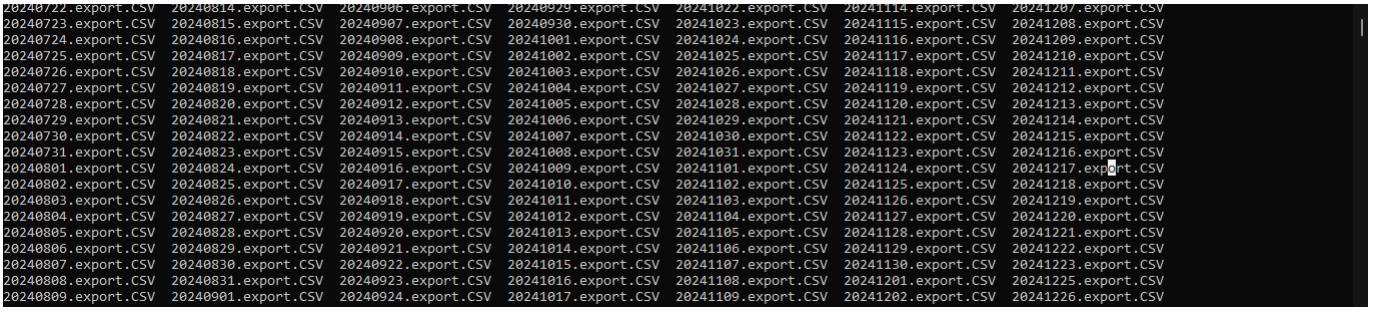


Copier les fichiers CSV dans input

SHELL

```
hadoop fs -put ./upload/*.CSV input
```

Vérifier le contenu de input `hadoop fs -ls input`



Exécuter le job

SHELL

```
hadoop jar CountTypeEvent-1.0.jar tn.insat.GDELTDriver input output
```

```
25/01/06 11:08:03 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1736160723346_0003
25/01/06 11:08:03 INFO impl.YarnClientImpl: Submitted application application_1736160723346_0003
25/01/06 11:08:03 INFO mapreduce.Job: The url to track the job: http://hadoop-master:8088/proxy/application_173616072
25/01/06 11:08:03 INFO mapreduce.Job: Running job: job_1736160723346_0003
25/01/06 11:08:47 INFO mapreduce.Job: Job job_1736160723346_0003 running in uber mode : false
25/01/06 11:08:47 INFO mapreduce.Job:  map 0% reduce 0%
25/01/06 11:10:30 INFO mapreduce.Job:  map 1% reduce 0%
25/01/06 11:10:32 INFO mapreduce.Job:  map 2% reduce 0%
25/01/06 11:10:33 INFO mapreduce.Job:  map 3% reduce 0%
25/01/06 11:10:36 INFO mapreduce.Job:  map 5% reduce 0%
25/01/06 11:10:37 INFO mapreduce.Job:  map 6% reduce 0%
25/01/06 11:10:38 INFO mapreduce.Job:  map 8% reduce 0%
25/01/06 11:10:39 INFO mapreduce.Job:  map 9% reduce 0%
25/01/06 11:10:40 INFO mapreduce.Job:  map 10% reduce 0%
25/01/06 11:10:41 INFO mapreduce.Job:  map 11% reduce 0%
25/01/06 11:10:42 INFO mapreduce.Job:  map 12% reduce 0%
25/01/06 11:10:43 INFO mapreduce.Job:  map 13% reduce 0%
25/01/06 11:10:49 INFO mapreduce.Job:  map 14% reduce 0%
25/01/06 11:10:52 INFO mapreduce.Job:  map 15% reduce 0%
25/01/06 11:10:57 INFO mapreduce.Job:  map 16% reduce 5%
25/01/06 11:11:00 INFO mapreduce.Job:  map 17% reduce 5%
25/01/06 11:11:01 INFO mapreduce.Job:  map 18% reduce 5%
25/01/06 11:11:02 INFO mapreduce.Job:  map 19% reduce 5%
25/01/06 11:11:03 INFO mapreduce.Job:  map 20% reduce 5%
25/01/06 11:11:04 INFO mapreduce.Job:  map 22% reduce 5%
25/01/06 11:11:05 INFO mapreduce.Job:  map 24% reduce 7%
25/01/06 11:11:06 INFO mapreduce.Job:  map 26% reduce 7%
25/01/06 11:11:07 INFO mapreduce.Job:  map 27% reduce 7%
25/01/06 11:11:08 INFO mapreduce.Job:  map 28% reduce 7%
25/01/06 11:11:09 INFO mapreduce.Job:  map 28% reduce 9%
25/01/06 11:11:12 INFO mapreduce.Job:  map 29% reduce 9%
25/01/06 11:11:15 INFO mapreduce.Job:  map 29% reduce 10%
25/01/06 11:11:18 INFO mapreduce.Job:  map 30% reduce 10%
25/01/06 11:11:19 INFO mapreduce.Job:  map 31% reduce 10%
25/01/06 11:11:20 INFO mapreduce.Job:  map 32% reduce 10%
25/01/06 11:11:21 INFO mapreduce.Job:  map 33% reduce 10%
25/01/06 11:11:24 INFO mapreduce.Job:  map 34% reduce 11%
25/01/06 11:11:29 INFO mapreduce.Job:  map 35% reduce 11%
25/01/06 11:11:30 INFO mapreduce.Job:  map 36% reduce 11%
25/01/06 11:11:31 INFO mapreduce.Job:  map 37% reduce 11%
25/01/06 11:11:32 INFO mapreduce.Job:  map 38% reduce 11%
25/01/06 11:11:33 INFO mapreduce.Job:  map 40% reduce 12%
25/01/06 11:11:34 INFO mapreduce.Job:  map 41% reduce 12%
25/01/06 11:11:35 INFO mapreduce.Job:  map 41% reduce 13%
```

Vérifier les sorties au niveau du dossier output

SHELL

```
root@hadoop-master:~# hadoop fs -ls output
Found 2 items
-rw-r--r--   2 root supergroup          0 2025-01-06 11:13 output/_SUCCESS
-rw-r--r--   2 root supergroup    3732043 2025-01-06 11:13 output/part-r-00000
```

vérifier si on a le résultat attendu

```
root@hadoop-master:~# hadoop fs -tail -f output/part-r-00000

ZI-0871 4
ZI-0872 1
ZI-0873 1
ZI-0874 142
ZI-090  1223
ZI-091  13
ZI-092  5
ZI-093  7
ZI-100  835
ZI-101  24
ZI-1014 46
ZI-1031 1
ZI-1041 5
ZI-1043 37
ZI-1044 6
ZI-105  10
ZI-1053 37
ZI-1056 3
ZI-110  1268
ZI-111  1191
ZI-112  2068
ZI-1121 26
ZI-1122 9
ZI-1123 27
ZI-1124 33
ZI-1125 1
ZI-113  25
ZI-114  369
ZI-115  7
ZI-116  8
ZI-120  1027
ZI-121  5
ZI-122  5
ZI-1231 2
ZI-1233 12
ZI-124  29
ZI-1241 34
ZI-1243 34
ZI-1244 10
ZI-1246 1
ZI-125  120
ZI-127  19
```

Etape 6 difficultés rencontrées et conclusion

Difficultés rencontrées :

L'une des principales difficultés rencontrées lors de l'étape 5 était la gestion du grand nombre de fichiers CSV disponibles sur la plateforme GDELT. Avec des milliers de fichiers à traiter, il a été nécessaire de se concentrer sur un sous-ensemble des fichiers plutôt que de tous les télécharger et traiter en une seule fois. Pour cela, j'ai téléchargé et extrait les liens de fichiers CSV en les organisant dans un fichier texte, puis j'ai utilisé des scripts pour télécharger les fichiers ZIP et les extraire avant de les charger sur HDFS. Ce processus a permis d'éviter de surcharger le cluster Hadoop avec un trop grand nombre de fichiers simultanément.

Conclusion :

En conclusion, l'atelier a permis de mettre en place une application Big Data fonctionnelle pour analyser les données mondiales issues de la plateforme GDELT. Grâce à l'utilisation de Hadoop et Docker, nous avons pu simuler un environnement distribué sur 10 nœuds, ce qui a facilité le traitement des grandes quantités de données. Le projet a été une excellente occasion d'explorer le traitement parallèle et les principes du Big Data tout en surmontant des défis liés à la gestion de données massives.