ECOLE NORMALE SUPÉRIEURE DE
L'ENSEIGNEMENT TECHNIQUE DE MOHAMMEDIA

UNIVERSITÉ HASSAN II DE CASABLANCA

ENSET

المدرسة العليا لأساتذة التعليم التقني
المحمدية
جامعة الحسن الثاني بالدار البيضاء

DEPARTEMENT MATHEMATIQUES ET INFORMATIQUE

Filière Ingénieur :

« Ingénierie Informatique : Big Data et Cloud Computing »
II-BDCC2

Big Data: Architectures de stockages

Rapport du TP

Traitement de données HDFS avec Map Reduce

Réalisé par :

- MOUAKKAL Nouhayla

Encadré par :

- Pr BOUSSELHAM Abdelmjid

Année Universitaire : 2023 - 2024

Sommaire:

Introduction

Le traitement de grandes quantités de données est une tâche essentielle dans l'ère numérique actuelle. Hadoop MapReduce est un modèle de programmation utilisé pour traiter et générer de grands ensembles de données avec un algorithme parallèle distribué sur un cluster. Ce rapport détaille deux exercices réalisés dans le cadre d'un projet de traitement de données massives à l'aide de Hadoop MapReduce. Le premier exercice porte sur l'analyse des données de vente pour calculer les ventes totales par ville, et le second sur l'analyse des fichiers journaux Web pour compter les requêtes par adresse IP

Exercice 1 : Traitement de Données de Vente

I. Objectif :

L'objectif de cet exercice est d'analyser un ensemble de données de ventes stockées dans un fichier texte et de calculer le total des ventes par ville. Chaque ligne du fichier de données contient des informations sur une vente, y compris la date, la ville, le produit et le prix.

II. La Tâche 1 : Calcul du Total des Ventes par Ville

Code du Driver

Le code du Driver initialise le job Hadoop, configure les classes Mapper et Reducer, et spécifie les formats d'entrée et de sortie.

```java
public class Driver {
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf);
        job.setJarByClass(Driver.class);
        job.setMapperClass(JobMapper.class);
        job.setReducerClass(JobReduce.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(DoubleWritable.class);
        job.setInputFormatClass(TextInputFormat.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Code du Mapper

Le Mapper extrait la ville et le prix de chaque ligne de vente et émet une paire clé-valeur avec la ville comme clé et le prix comme valeur.

```java
public class JobMapper extends Mapper<LongWritable, Text,Text,DoubleWritable> {
    @Override
    protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text, DoubleWritable>.Context
context) throws IOException, InterruptedException {
        String[] tokens = value.toString().split(" ");
        if (tokens.length == 4) {
            String city = tokens[1];
            double price = Double.parseDouble(tokens[3]);
            context.write(new Text(city), new DoubleWritable(price));
        }
    }
}
```

Code du Reducer

Le Reducer agrège les prix pour chaque ville et émet le total des ventes par ville.

```java
public class JobReducer extends Reducer<Text, DoubleWritable,Text,DoubleWritable> {
    @Override
    protected void reduce(Text key, Iterable<DoubleWritable> values, Reducer<Text, DoubleWritable, Text,
DoubleWritable>.Context context) throws IOException, InterruptedException {
        double totalSales = 0.0;
        for (DoubleWritable value : values) {
            totalSales += value.get();
        }
        context.write(key, new DoubleWritable(totalSales));
    }
}
```

```
T14s@DESKTOP-J9PA2CS MINGW64 ~/Desktop/docker-hadoop (master)
$ docker cp ventes.txt 152a40d62b55:/opt/hadoop-3.2.1
Successfully copied 2.56kB to 152a40d62b55:/opt/hadoop-3.2.1

T14s@DESKTOP-J9PA2CS MINGW64 ~/Desktop/docker-hadoop (master)
$ docker cp tp2_hdfs-1.0-SNAPSHOT.jar 152a40d62b55:/opt/hadoop-3.2.1
Successfully copied 15.9kB to 152a40d62b55:/opt/hadoop-3.2.1

T14s@DESKTOP-J9PA2CS MINGW64 ~/Desktop/docker-hadoop (master)
$ docker exec -it 152a40d62b55 bash
root@152a40d62b55:/# cd opt
root@152a40d62b55:/opt# ls
hadoop  hadoop-3.2.1
root@152a40d62b55:/opt# cd hadoop-3.2.1
root@152a40d62b55:/opt/hadoop-3.2.1# ls
LICENSE.txt  NOTICE.txt  README.txt  bin  etc  include  lib  libexec  logs  sbin  share  tp2_hdfs-1.0-SNAPSHOT.jar  ventes.txt
```

Exécution et Résultats

Test avec une list des ventes

```
root@152a40d62b55:/opt/hadoop-3.2.1# cat ventes.txt
2024-03-18 Rabat Laptop 1200
2024-03-18 Rabat Tablet 500
2024-03-18 Rabat Smartphone 800
2024-03-18 Casablanca Smartphone 800
2024-03-19 Marrakech Tablet 500
2024-03-19 Marrakech Headphones 100
2024-03-20 Tanger Laptop 1100
2024-03-20 Tanger Tablet 450
2024-03-21 Sale Smartphone 750
2024-03-21 Sale Headphones 90
2024-03-22 Rabat Laptop 1050
2024-03-22 Rabat Tablet 480
2024-03-23 Marrakech Smartphone 850
2024-03-23 Casablanca Laptop 1150
2024-03-24 Tanger Tablet 520
2024-03-24 Tanger Headphones 95
2024-03-25 Sale Laptop 1120
2024-03-25 Sale Tablet 470
2024-03-26 Rabat Smartphone 770
2024-03-26 Casablanca Headphones 85
root@152a40d62b55:/opt/hadoop-3.2.1#
```

Execution (Avec l'année 2024 comme paramètre)

```
hadoop jar tp2_hdfs-1.0-SNAPSHOT.jar
com.mouakkal.ex1.job2.Driver /TP2-HADOOP/ventes.txt /TP2-
HADOOP/ex1_job2_output 2024
```

```
root@152a40d62b55:/opt/hadoop-3.2.1# hdfs dfs -cat /TP2-HADOOP/ex1_job2_output/part-r-00000
2024-03-30 17:29:43,391 INFO sasl.SaslDataTransferClient: SASL encryption trust check: local
Casablanca      2035.0
Marrakech       1450.0
Rabat   4800.0
Sale    2430.0
Tanger  2165.0
```

III. La Tâche 2 : Calcul du Total des Ventes des Produits par Ville pour une Année Donnée

Code du Driver

Similaire à la tâche précédente, mais avec des ajustements pour inclure le filtrage par année.

```java
public class Driver {
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf);
        job.setJarByClass(Driver.class);
        job.setMapperClass(JobMapper.class);
        job.setReducerClass(JobReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        job.setInputFormatClass(TextInputFormat.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Code du Mapper et Reducer :

Les Mapper et Reducer sont similaires à ceux de la tâche 1, avec une logique ajoutée pour filtrer les ventes par année.

2. Source code du Mapper

```
public class JobMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
    @Override
    protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text, IntWritable>.Context context) throws IOException,
InterruptedException {
        String word = value.toString();
        String[] informations = word.split(" ");
        String ipAdd = informations[0].split(" -- ")[0];
        context.write(new Text(ipAdd),new IntWritable(1));
    }
}
```

3. Source code du Reducer

```
public class JobReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
    @Override
    protected void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable, Text, IntWritable>.Context context) throws
IOException, InterruptedException {
        int counter=0;
        Iterator<IntWritable> iterator = values.iterator();
        while (iterator.hasNext()){
            counter+=iterator.next().get();
        }
        context.write(key,new IntWritable(counter));
    }
}
```

Exécution et Résultats :

Test avec une liste des ventes

```
root@152a40d62b55:/opt/hadoop-3.2.1# cat ventes.txt
2024-03-18 Rabat Laptop 1200
2024-03-18 Rabat Tablet 500
2024-03-18 Rabat Smartphone 800
2024-03-18 Casablanca Smartphone 800
2024-03-19 Marrakech Tablet 500
2024-03-19 Marrakech Headphones 100
2024-03-20 Tanger Laptop 1100
2024-03-20 Tanger Tablet 450
2024-03-21 Sale Smartphone 750
2024-03-21 Sale Headphones 90
2024-03-22 Rabat Laptop 1050
2024-03-22 Rabat Tablet 480
2024-03-23 Marrakech Smartphone 850
2024-03-23 Casablanca Laptop 1150
2024-03-24 Tanger Tablet 520
2024-03-24 Tanger Headphones 95
2024-03-25 Sale Laptop 1120
2024-03-25 Sale Tablet 470
2024-03-26 Rabat Smartphone 770
2024-03-26 Casablanca Headphones 85
root@152a40d62b55:/opt/hadoop-3.2.1#
```

Le job est exécuté avec l'année 2024 comme paramètre pour filtrer les ventes de cette année spécifique.

```
hadoop jar tp2_hdfs-1.0-SNAPSHOT.jar com.mouakkal.ex1.job2.Driver /TP2-HADOOP/ventes.txt /TP2-HADOOP/ex1_job2_output 2024
```

```
root@152a40d62b55:/opt/hadoop-3.2.1# hdfs dfs -cat /TP2-HADOOP/ex1_job2_output/part-r-00000
2024-03-30 17:29:43,391 INFO sasl.SaslDataTransferClient: SASL encryption trust check: local
Casablanca      2035.0
Marrakech       1450.0
Rabat   4800.0
Sale    2430.0
Tanger  2165.0
```

Exercice 2 : Analyse des Fichiers Journaux Web

I. Objectif

L'objectif est de compter le nombre total de requêtes et le nombre de requêtes réussies (code de réponse HTTP 200) par adresse IP à partir d'un fichier journal Web.

Code du Driver

Le Driver configure le job Hadoop pour traiter les fichiers journaux Web.

```
public class Driver {
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf);
        job.setJarByClass(Driver.class);
        job.setMapperClass(JobMapper.class);
        job.setReducerClass(JobReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        job.setInputFormatClass(TextInputFormat.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Code du Mapper

Le Mapper extrait l'adresse IP de chaque ligne du journal et émet une paire clé-valeur avec l'adresse IP comme clé et 1 comme valeur pour compter les requêtes.

```
public class JobMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
    @Override
    protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text,
IntWritable>.Context context) throws IOException, InterruptedException {
        String word = value.toString();
        String[] informations = word.split(" ");
        String ipAdd = informations[0].split(" -- ")[0];
        context.write(new Text(ipAdd),new IntWritable(1));
    }
}
```

Code du Reducer

Le Reducer agrège les compteurs pour chaque adresse IP et émet le nombre total de requêtes par adresse IP.

```
public class JobReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
    @Override
    protected void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable,
Text, IntWritable>.Context context) throws IOException, InterruptedException {
        int counter=0;
        Iterator<IntWritable> iterator = values.iterator();
        while (iterator.hasNext()){
            counter+=iterator.next().get();
        }
        context.write(key,new IntWritable(counter));
    }
}
```

Résultat et exécution :

Test avec un fichier log

```
T14s@DESKTOP-J9PA2CS MINGW64 ~/Desktop/docker-hadoop (master)
$ docker cp log.txt 152a40d62b55:/opt/hadoop-3.2.1
Successfully copied 3.07kB to 152a40d62b55:/opt/hadoop-3.2.1

T14s@DESKTOP-J9PA2CS MINGW64 ~/Desktop/docker-hadoop (master)
$ docker exec -it 152a40d62b55 bash
root@152a40d62b55:/# cd /opt/hadoop-3.2.1
root@152a40d62b55:/opt/hadoop-3.2.1# cat log.txt
192.168.1.1 -- [12/May/2023:15:30:45 +0000] "GET /page1 HTTP/1.1" 200 1234
192.168.1.2 -- [12/May/2023:15:31:02 +0000] "GET /page2 HTTP/1.1" 404 567
192.168.1.1 -- [12/May/2023:15:32:10 +0000] "GET /page1 HTTP/1.1" 200 789
192.168.1.3 -- [12/May/2023:15:32:35 +0000] "GET /page3 HTTP/1.1" 200 987
192.168.1.1 -- [12/May/2023:15:33:12 +0000] "GET /page2 HTTP/1.1" 404 543
192.168.1.2 -- [12/May/2023:15:33:45 +0000] "GET /page1 HTTP/1.1" 200 876
192.168.1.1 -- [12/May/2023:15:34:18 +0000] "GET /page3 HTTP/1.1" 200 765
192.168.1.3 -- [12/May/2023:15:34:56 +0000] "GET /page2 HTTP/1.1" 404 432
192.168.1.1 -- [12/May/2023:15:35:25 +0000] "GET /page1 HTTP/1.1" 200 654
192.168.1.2 -- [12/May/2023:15:36:01 +0000] "GET /page2 HTTP/1.1" 200 543
192.168.1.3 -- [12/May/2023:15:36:42 +0000] "GET /page3 HTTP/1.1" 200 432
192.168.1.1 -- [12/May/2023:15:37:18 +0000] "GET /page1 HTTP/1.1" 404 321
192.168.1.2 -- [12/May/2023:15:37:57 +0000] "GET /page2 HTTP/1.1" 200 210
192.168.1.3 -- [12/May/2023:15:38:38 +0000] "GET /page3 HTTP/1.1" 200 109
192.168.1.1 -- [12/May/2023:15:39:21 +0000] "GET /page1 HTTP/1.1" 200 876
192.168.1.2 -- [12/May/2023:15:40:05 +0000] "GET /page2 HTTP/1.1" 200 765
192.168.1.3 -- [12/May/2023:15:40:50 +0000] "GET /page3 HTTP/1.1" 200 654
192.168.1.1 -- [12/May/2023:15:41:36 +0000] "GET /page1 HTTP/1.1" 200 543
192.168.1.2 -- [12/May/2023:15:42:23 +0000] "GET /page2 HTTP/1.1" 200 432
192.168.1.3 -- [12/May/2023:15:43:11 +0000] "GET /page3 HTTP/1.1" 200 321
root@152a40d62b55:/opt/hadoop-3.2.1#
```

Le job est exécuté avec la commande suivante, spécifiant les chemins d'entrée et de sortie :

```
hadoop jar tp2_hdfs-1.0-SNAPSHOT.jar com.mouakkal.ex2.Driver /TP2-HADOOP/log.txt /TP2-HADOOP/ex2_output
```

```
root@152a40d62b55:/opt/hadoop-3.2.1# hdfs dfs -copyFromLocal log.txt /TP2-HADOOP
2024-03-30 18:06:04,207 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
root@152a40d62b55:/opt/hadoop-3.2.1# hadoop jar tp2_hdfs-1.0-SNAPSHOT.jar com.errami.ex2.Driver /TP2-HADOOP/log.txt /TP2-HADOOP/ex2_output
2024-03-30 18:19:39,673 INFO client.RMProxy: Connecting to ResourceManager at resourcemanager/172.22.0.3:8032
2024-03-30 18:19:39,846 INFO client.AHSProxy: Connecting to Application History server at historyserver/172.22.0.5:10200
2024-03-30 18:19:40,030 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface
 ToolRunner to remedy this.
2024-03-30 18:19:40,132 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/root/.staging/job_1
2024-03-30 18:19:40,374 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2024-03-30 18:19:40,490 INFO input.FileInputFormat: Total input files to process : 1
2024-03-30 18:19:40,537 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2024-03-30 18:19:40,563 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2024-03-30 18:19:40,573 INFO mapreduce.JobSubmitter: number of splits:1
2024-03-30 18:19:40,719 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2024-03-30 18:19:40,739 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1711816744481_0004
2024-03-30 18:19:40,739 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-03-30 18:19:40,932 INFO conf.Configuration: resource-types.xml not found
2024-03-30 18:19:40,932 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2024-03-30 18:19:41,217 INFO impl.YarnClientImpl: Submitted application application_1711816744481_0004
2024-03-30 18:19:41,257 INFO mapreduce.Job: The url to track the job: http://resourcemanager:8088/proxy/application_1711816744481_0004/
2024-03-30 18:19:41,258 INFO mapreduce.Job: Running job: job_1711816744481_0004
2024-03-30 18:19:48,365 INFO mapreduce.Job: Job job_1711816744481_0004 running in uber mode : false
2024-03-30 18:19:48,368 INFO mapreduce.Job:  map 0% reduce 0%
2024-03-30 18:19:55,741 INFO mapreduce.Job:  map 100% reduce 0%
2024-03-30 18:19:59,781 INFO mapreduce.Job:  map 100% reduce 100%
2024-03-30 18:20:00,840 INFO mapreduce.Job: Job job_1711816744481_0004 completed successfully
2024-03-30 18:20:00,946 INFO mapreduce.Job: Counters: 54
```

Résultats du nombre de requêtes réussies :

```
root@152a40d62b55:/opt/hadoop-3.2.1# hdfs dfs -cat /TP2-HADOOP/ex2_output/part-r-00000
2024-03-30 19:20:30,475 INFO sasl.SaslDataTransferClient: SASL encryption trust check:
192.168.1.1     8
192.168.1.2     6
192.168.1.3     6
root@152a40d62b55:/opt/hadoop-3.2.1#
```

Conclusion

En conclusion, Ce TP a démontré l'utilisation efficace de Hadoop MapReduce pour traiter des ensembles de données massives dans différents contextes : analyse des données de vente et analyse des fichiers journaux Web. Les exercices ont illustré la capacité de MapReduce à diviser les tâches de traitement de données en sous-tâches parallèles, permettant une gestion efficace et rapide des données volumineuses. Ce rapport souligne l'importance de Hadoop MapReduce comme outil puissant pour les analystes de données et les ingénieurs de données dans diverses applications industrielles et de recherche.