

Compte Rendu Lab2

Programmation avec l'API HDFS et MapReduce

Nom et Prénom : AMRANI Hamza

Filière : IDF

I. Démarrer le Cluster Hadoop

Pour le démarrage du cluster Hadoop, voir TP1 et TP2.

N.B :

Voir les comptes rendus du TP1 et TP2 pour les détails complets de cette partie (démarrage des conteneurs, accès au master, démarrage de Hadoop et YARN).

II. Programmation avec l'API HDFS

N.B :

La partie HDFS (HadoopFileStatus, ReadHDFS, WriteHDFS) a déjà été réalisée dans le TP2. Voir le compte rendu du TP2 pour ces exemples.

III. Programmation avec l'API MapReduce

L'objectif de cette partie est de simuler l'exemple WordCount. Le job permet de compter le nombre d'occurrences de chaque mot présent dans un fichier texte.

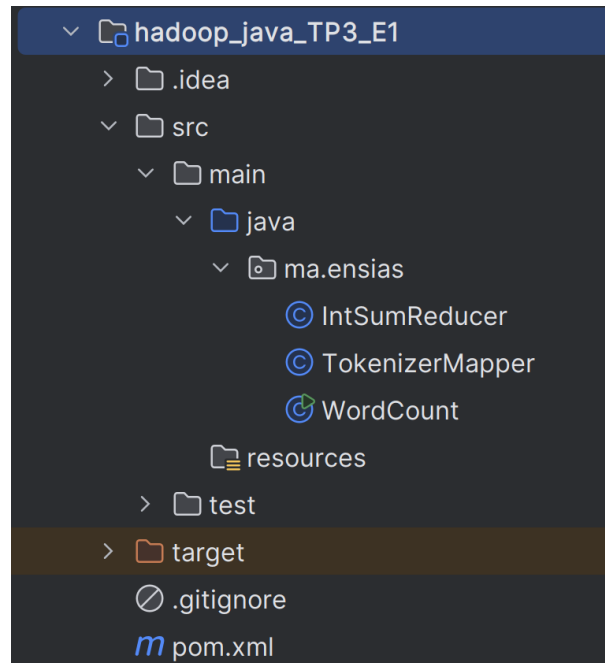
Principe :

- **Phase de Mapping :** le texte est découpé en mots. Pour chaque mot, génération d'une paire (mot, 1)
- **Phase de Reducing :** les paires sont regroupées par mot et agrégées pour obtenir le nombre total d'occurrences

1. Structure du projet

Projet 1 : hadoop_java_TP3_E1

Organisation du projet Maven avec les trois classes principales :



Le projet contient :

- **TokenizerMapper** : classe Mapper pour découper le texte en mots
- **IntSumReducer** : classe Reducer pour additionner les occurrences
- **WordCount** : classe principale pour lancer le job

2. Compilation et déploiement

Compiler le projet avec Maven :

```
mvn clean package
cp target/hadoop-hdfs-WordCount.jar ~/documents/Big\ Data/
hadoop_project/
```

```
hamza@Hamza-amr MINGW64 ~/Documents/Big Data/hadoop_project/hadoop_java_TP3_E1
$ mvn clean package
[INFO] Scanning for projects...
[INFO]
[INFO] -----< ma.ensias:hadoop_java_TP3_E1 >-----
[INFO] Building hadoop_java_TP3_E1 1.0-SNAPSHOT
[INFO] from pom.xml
[INFO] -----[ jar ]-----
[INFO] Building jar: C:\Users\hamza\Documents\Big Data\hadoop_project\hadoop_java_TP3_E1\target\hadoop-hdfs-WordCount.jar
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 6.337 s
[INFO] Finished at: 2025-10-11T07:51:11+02:00
[INFO] -----
hamza@Hamza-amr MINGW64 ~/Documents/Big Data/hadoop_project/hadoop_java_TP3_E1
$ cp target/hadoop-hdfs-WordCount.jar ~/documents/Big\ Data/hadoop_project/
```

3. Exécution du job WordCount

Lancer le job MapReduce :

```
hadoop jar /shared_volume/hadoop-hdfs-WordCount.jar /user/root/web_input/alice.txt /user/root/output_wordcount
```

```
root@hadoop-master: ~
root@hadoop-master:~# hadoop jar /shared_volume/hadoop-hdfs-wordCount.jar /user/root/web_input/alice.txt /user/root/output_wordcount
2025-10-11 05:58:12,598 INFO client.RMProxy: Connecting to ResourceManager at hadoop-master/172.19.0.2:8032
2025-10-11 05:58:12,765 INFO client.AHSProxy: Connecting to Application History server at localhost/127.0.0.1:10200
2025-10-11 05:58:13,335 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2025-10-11 05:58:13,356 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/root/.staging/job_1760146326434_0001
2025-10-11 05:58:14,133 INFO input.FileInputFormat: Total input files to process : 1
2025-10-11 05:58:14,260 INFO mapreduce.JobSubmitter: number of splits:1
2025-10-11 05:58:14,321 INFO Configuration.deprecation: yarn.resourcemanager.system-metrics-publisher.enabled is deprecated. Instead, use yarn.system-metrics-publisher.enabled
2025-10-11 05:58:14,441 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1760146326434_0001
2025-10-11 05:58:14,442 INFO mapreduce.JobSubmitter: Executing with tokens: []
2025-10-11 05:58:14,646 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'
2025-10-11 05:58:15,442 INFO impl.YarnClientImpl: Submitted application application_1760146326434_0001
2025-10-11 05:58:15,489 INFO mapreduce.Job: The url to track the job: http://hadoop-master:8088/proxy/application_1760146326434_0001/
2025-10-11 05:58:15,489 INFO mapreduce.Job: Running job: job_1760146326434_0001
2025-10-11 05:58:24,626 INFO mapreduce.Job: Job job_1760146326434_0001 running in uber mode : false
2025-10-11 05:58:24,628 INFO mapreduce.Job: map 0% reduce 0%
2025-10-11 05:58:31,698 INFO mapreduce.Job: map 100% reduce 0%
2025-10-11 05:58:33,999 INFO mapreduce.Job: map 100% reduce 100%
2025-10-11 05:58:34,006 INFO mapreduce.Job: Job job_1760146326434_0001 completed successfully
```

Le job s'exécute avec succès :

- Map : 100%
- Reduce : 100%
- Statut : **Job completed successfully**

4. Vérification via l'interface YARN

Accéder à l'interface web pour suivre l'exécution :

```
http://localhost:8088
```

ID	User	Name	Application Type	Queue	Application Priority	Start Time	Launch Time	Finish Time	State	Final Status	Running Containers	Allocated CPU V-Cores	Allocated Memory MB	Reserved CPU V-Cores	Reserved Memory MB	% of Queue	% of Cluster	Progress	Tracking UI	Blacklisted Nodes
application_1760146326434_0001	root	word count	MAPREDUCE	default	0	Sat Oct 11 07:58:15 +0200 2025	Sat Oct 11 07:58:16 +0200 2025	Sat Oct 11 07:58:35 +0200 2025	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	0.0	0.0		History	0

Le job apparaît avec le statut **FINISHED** et **SUCCEEDED**.

5. Résultats du WordCount

Vérifier les fichiers de sortie et afficher les résultats :

```
# Lister les fichiers de sortie
hdfs dfs -ls /user/root/output_wordcount

# Afficher les 20 premiers résultats
hdfs dfs -cat /user/root/output_wordcount/part-r-00000 | head -20

# Afficher les 10 mots les plus fréquents
hdfs dfs -cat /user/root/output_wordcount/part-r-00000 | sort -t$'\t' -k2 -nr | head -10

# Compter le nombre de mots uniques
hdfs dfs -cat /user/root/output_wordcount/part-r-00000 | wc -l
```

```
root@hadoop-master: ~
root@hadoop-master:~# hdfs dfs -ls /user/root/output_wordcount
Found 2 items
-rw-r--r--  2 root supergroup          0 2025-10-11 05:58 /user/root/output_wordcount/_SUCCESS
-rw-r--r--  2 root supergroup    52500 2025-10-11 05:58 /user/root/output_wordcount/part-r-00000
root@hadoop-master:~# hdfs dfs -cat /user/root/output_wordcount/part-r-00000 | head -20
" 1
"'TIS 1
"'Tis 1
"--SAID 1
"Come 1
"Coming 1
"Edwin 1
"French, 1
"HOW 1
"He's 1
"How 1
"I 8
"I'll 2
"Keep 1
"Let 1
"Such 1
"THEY 1
"There 2
"There's 1
"Too 1
cat: Unable to write to output stream.
root@hadoop-master:~# hdfs dfs -cat /user/root/output_wordcount/part-r-00000 | sort -t$'\t' -k2 -nr | head -10
the 1515
and 721
to 719
a 608
of 500
she 483
said 414
in 350
it 346
was 327
root@hadoop-master:~# hdfs dfs -cat /user/root/output_wordcount/part-r-00000 | wc -l
5464
```

Résultats principaux :

- **5464 mots uniques** identifiés dans le fichier alice.txt
- Les mots les plus fréquents : "the" (1515), "and" (721), "to" (719)
- Taille du fichier de sortie : 52500 octets

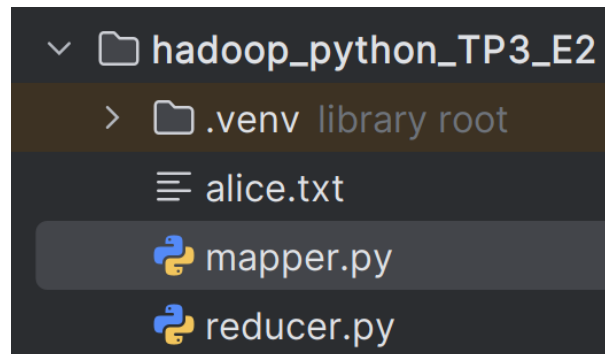
6. MapReduce avec Python

L'objectif de cette partie est de réaliser le même exemple WordCount en utilisant Python avec Hadoop Streaming. Cette approche permet d'utiliser des scripts Python comme mapper et reducer sans avoir à écrire de code Java.

3.6.1 Structure du projet

Projet 2 : `hadoop_python_TP3_E2`

Organisation du projet avec les fichiers Python :



Le projet contient :

- **mapper.py** : script Python pour découper le texte en mots
- **reducer.py** : script Python pour additionner les occurrences
- **alice.txt** : fichier texte d'entrée (déjà présent dans HDFS)

N.B :

Les fichiers Python sont déjà présents dans le dossier partagé `/shared_volume/hadoop_python_TP3_E2/`.

3.6.2 Créer le Mapper

Le fichier `mapper.py` lit les lignes du fichier texte et émet une paire (mot, 1) pour chaque mot.

Tester localement le mapper :

```
cat alice.txt | python mapper.py
```

```
hamza@Hamza-amr MINGW64 ~/Documents/Big Data/hadoop_project/hadoop_python_TP3_E2
$ cat alice.txt | python mapper.py
PROJECT 1
GUTENBERG 1
AND 1
DUNCAN 1
RESEARCH 1
SHAREWARE 1
```

3.6.3 Créer le Reducer

Le fichier `reducer.py` reçoit les paires triées et agrège les comptages pour chaque mot

Tester localement la chaîne complète (mapper | sort | reducer) :

```
cat alice.txt | python mapper.py | sort -k1,1 | python reducer.py
```

```
hamza@Hamza-amr MINGW64 ~/Documents/Big Data/hadoop_project/hadoop_python_TP3_E2
$ cat alice.txt | python mapper.py | sort -k1,1 | python reducer.py
" ' 1
"'TIS 1
"'Tis 1
"--SAID 1
"Come 1
"Coming 1
```

3.6.4 Localiser le JAR Hadoop Streaming

Avant d'exécuter le job avec Hadoop, il faut localiser le fichier JAR Hadoop Streaming :

```
find / -name 'hadoop-streaming*.jar'
```

```
root@hadoop-master: ~
root@hadoop-master:~# find / -name 'hadoop-streaming*.jar'
/usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.2.0.jar
/usr/local/hadoop/share/hadoop/tools/sources/hadoop-streaming-3.2.0-sources.jar
/usr/local/hadoop/share/hadoop/tools/sources/hadoop-streaming-3.2.0-test-sources.jar
```

Nous utiliserons le premier fichier JAR pour exécuter notre job.

3.6.5 Exécution avec Hadoop Streaming

Lancer le job MapReduce avec Hadoop Streaming :

```
hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.2.0.jar \
  -files /shared_volume/hadoop_python_TP3_E2/mapper.py,/
  shared_volume/hadoop_python_TP3_E2/reducer.py \
  -mapper "python3 mapper.py" \
  -reducer "python3 reducer.py" \
  -input /user/root/web_input/alice.txt \
  -output /user/root/output_python
```

```

root@hadoop-master:~# hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.2.0.jar \
  -files /shared_volume/hadoop_python_TP3_E2/mapper.py,/shared_volume/hadoop_python_TP3_E2/reducer.py \
  -mapper "python3 mapper.py" \
  -reducer "python3 reducer.py" \
  -input /user/root/web_input/alice.txt \
  -output /user/root/output_python
packageJobJar: [/tmp/hadoop-unjar3655848609933018432/] [] /tmp/streamjob6702744710351404866.jar tmpDir=null
2025-10-11 10:53:45,830 INFO client.RMProxy: Connecting to ResourceManager at hadoop-master/172.19.0.2:8032
2025-10-11 10:53:45,971 INFO client.AHSProxy: Connecting to Application History server at localhost/127.0.0.1:10200
2025-10-11 10:53:45,997 INFO client.RMProxy: Connecting to ResourceManager at hadoop-master/172.19.0.2:8032
2025-10-11 10:53:45,997 INFO client.AHSProxy: Connecting to Application History server at localhost/127.0.0.1:10200
2025-10-11 10:53:46,373 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/root/.staging/
2025-10-11 10:53:47,430 INFO mapred.FileInputFormat: Total input files to process : 1
2025-10-11 10:53:47,539 INFO mapreduce.JobSubmitter: number of splits:2
2025-10-11 10:53:47,592 INFO Configuration.deprecation: yarn.resourcemanager.system-metrics-publisher.enabled is deprecated. Instead,
2025-10-11 10:53:47,696 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1760146326434_0002
2025-10-11 10:53:47,698 INFO mapreduce.JobSubmitter: Executing with tokens: []
2025-10-11 10:53:47,833 INFO conf.Configuration: resource-types.xml not found
2025-10-11 10:53:47,833 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2025-10-11 10:53:48,312 INFO impl.YarnClientImpl: Submitted application application_1760146326434_0002
2025-10-11 10:53:48,342 INFO mapreduce.Job: The url to track the job: http://hadoop-master:8088/proxy/application_1760146326434_0002/
2025-10-11 10:53:48,343 INFO mapreduce.Job: Running job: job_1760146326434_0002
2025-10-11 10:53:54,487 INFO mapreduce.Job: Job job_1760146326434_0002 running in uber mode : false
2025-10-11 10:53:54,489 INFO mapreduce.Job: map 0% reduce 0%
2025-10-11 10:54:01,583 INFO mapreduce.Job: map 100% reduce 0%
2025-10-11 10:54:07,625 INFO mapreduce.Job: map 100% reduce 100%
2025-10-11 10:54:07,632 INFO mapreduce.Job: Job job_1760146326434_0002 completed successfully
2025-10-11 10:54:07,709 INFO mapreduce.Job: Counters: 54

```

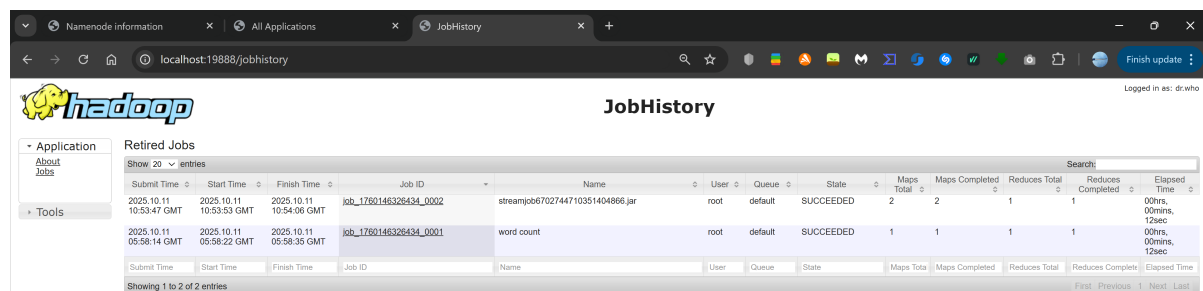
Le job s'exécute avec succès :

- Map : 100%
- Reduce : 100%
- Statut : **Job completed successfully**

7. Vérification via l'interface JobHistory

Accéder à l'interface web JobHistory pour consulter l'historique des jobs :

<http://localhost:19888/jobhistory>



The screenshot shows the Hadoop JobHistory web interface. The title is "JobHistory". On the left, there is a sidebar with "Application" and "Tools". The main content area shows a table of "Retired Jobs". The table has columns for Submit Time, Start Time, Finish Time, Job ID, Name, User, Queue, State, Maps Total, Maps Completed, Reduces Total, Reduces Completed, and Elapsed Time. Two jobs are listed:

Submit Time	Start Time	Finish Time	Job ID	Name	User	Queue	State	Maps Total	Maps Completed	Reduces Total	Reduces Completed	Elapsed Time
2025.10.11 10:53:47 GMT	2025.10.11 10:53:53 GMT	2025.10.11 10:54:06 GMT	job_1760146326434_0002	streamjob6702744710351404866.jar	root	default	SUCCEEDED	2	2	1	1	00hrs, 00mins, 19secs
2025.10.11 05:58:14 GMT	2025.10.11 05:58:22 GMT	2025.10.11 05:58:35 GMT	job_1760146326434_0001	word count	root	default	SUCCEEDED	1	1	1	1	00hrs, 00mins, 19secs

Showing 1 to 2 of 2 entries

Le job apparaît dans l'historique.

8. Résultats du WordCount Python

Vérifier les fichiers de sortie et afficher les résultats :

```

# Lister les fichiers de sortie
hdfs dfs -ls /user/root/output_python

# Afficher les 20 premiers résultats
hdfs dfs -cat /user/root/output_python/part-00000 | head -20

```

```
root@hadoop-master: ~  
root@hadoop-master:~# hdfs dfs -ls /user/root/output_python  
Found 2 items  
-rw-r--r-- 2 root supergroup 0 2025-10-11 10:54 /user/root/output_python/_SUCCESS  
-rw-r--r-- 2 root supergroup 52500 2025-10-11 10:54 /user/root/output_python/part-00000  
root@hadoop-master:~# hdfs dfs -cat /user/root/output_python/part-00000 | head -20  
"  
"TIS 1  
"Tis 1  
"--SAID 1  
"Come 1  
"Coming 1  
"Edwin 1  
"French, 1  
"HOW 1  
"He's 1  
"How 1  
"I 8  
"I'll 2  
"Keep 1  
"Let 1  
"Such 1  
"THEY 1  
"There 2  
"There's 1  
"Too 1
```

9. Comparaison Java vs Python

Avantages de Python avec Hadoop Streaming :

- Développement rapide avec des scripts simples
- Tests locaux faciles sans infrastructure Hadoop
- Pas de compilation nécessaire
- Utilisation de bibliothèques Python riches (numpy, pandas, etc.)

Avantages de Java :

- Meilleures performances pour les gros volumes de données
- Intégration native avec l'écosystème Hadoop
- Support complet des fonctionnalités avancées de MapReduce

IV. Conclusion

Dans ce Lab, nous avons appris à :

- Programmer avec l'API MapReduce en Java
- Implémenter l'algorithme WordCount
- Utiliser Hadoop Streaming avec Python
- Initialiser et gérer un dépôt Git/GitHub

Ces compétences sont essentielles pour le traitement distribué de données massives avec Hadoop.