République Algérienne Démocratique & Populaire
Ministère de l'Enseignement Supérieur & de la Recherche Scientifique
Ecole Supérieure en Informatique 08 mai 1945 de Sidi Bel Abbes
Année universitaire : 2023-2024 **Semestre 1**
3 SC : ISI & SIW **Module : Big Data**

TP 1 : Hadoop

Installation :

Nous allons utiliser tout au long de ce TP trois conteneurs représentant respectivement un noeud maître (Namenode) et deux noeuds esclaves (Datanodes).

Après le lancement de Docker, ouvrir la ligne de commande et effectuer le travail suivant :

1. Télécharger l'image docker liliasfaxi/spark-hadoop:hv-2.7.2 uploadée sur dockerhub.
2. Créer les trois conteneurs à partir de l'image téléchargée. Pour cela:
   2.1. Créer un réseau bridge nommé hadoop qui permettra de relier les trois conteneurs.
   2.2.Créer et lancer les trois conteneurs :
   - Le conteneur hadoop-master expose les ports 50070, 8088 et 16010.
   - Les conteneurs hadoop-slave1et hadoop-slave2 exposent le port 8042.
3. Entrer dans le conteneur master pour commencer à l'utiliser.

Le résultat de cette exécution sera le suivant:

```
root@hadoop-master:~#
```

Vous vous retrouverez dans le shell du namenode, et vous pourrez ainsi manipuler le cluster à votre guise. La première chose à faire, une fois dans le conteneur, est de lancer hadoop et yarn. Un script est fourni pour cela, appelé start-hadoop.sh. Lancer ce script.

Premiers pas avec Hadoop :

- Créer un répertoire dans HDFS, appelé input.
- Nous allons utiliser le fichier **purchases.txt**[1] comme entrée pour le traitement MapReduce. Ce fichier se trouve déjà sous le répertoire principal de votre machine master.
- Charger le fichier **purchases** dans le répertoire input que vous avez créé.
- Afficher le contenu du répertoire **input**.
- Afficher les dernières lignes du fichier **purchases.**

Map Reduce :

Présentation :

Un Job Map-Reduce se compose principalement de deux types de programmes:

[1] https://github.com/CodeMangler/udacity-hadoop-course/raw/master/Datasets/purchases.txt.gz

- **Mappers :** permettent d'extraire les données nécessaires sous forme de clé/valeur, pour pouvoir ensuite les trier selon la clé.
- **Reducers :** prennent un ensemble de données triées selon leur clé, et effectuent le traitement nécessaire sur ces données (somme, moyenne, total...).

Wordcount :

Nous allons tester un programme MapReduce grâce à un exemple très simple, le WordCount. Le Wordcount permet de calculer le nombre de mots dans un fichier donné, en décomposant le calcul en deux étapes :

- L'étape de Mapping, qui permet de découper le texte en mots et de délivrer en sortie un flux textuel, où chaque ligne contient le mot trouvé, suivi de la valeur 1 (pour dire que le mot a été trouvé une fois).
- L'étape de Reducing, qui permet de faire la somme des 1 pour chaque mot, pour trouver le nombre total d'occurrences de ce mot dans le texte.

Commençons par créer un projet Maven dans IntelliJ. Définir les valeurs suivantes pour votre projet:

- **GroupId:** hadoop.mapreduce
- **ArtifactId:** wordcount

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

```
<dependencies>
    <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-common</artifactId>
        <version>2.7.2</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.apache.hadoop/hadoop-mapreduce-client-core -->
    <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-mapreduce-client-core</artifactId>
        <version>2.7.2</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.apache.hadoop/hadoop-hdfs -->
```

```
        <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>
```

- Créer un package **tp1** sous le répertoire **src/main/java**.
- Créer la classe **TokenizerMapper** qui représente la classe MAP.
- Créer la classe **IntSumReducer** qui représente la classe REDUCE.
- Enfin, créer la classe **WordCount** qui représente la classe Driver.

Lancer Map Reduce sur le cluster :

Dans votre projet IntelliJ :

- Générer le fichier **jar** de l'application.
- Copier le fichier **jar** créé dans le conteneur **master**.
- Revenir au shell du conteneur master et lancer le job map reduce sur le fichier purchases.txt que vous aviez préalablement chargé dans le répertoire input de HDFS..
- Afficher le contenu du fichier généré.