Instructions de Lancement du Projet HadoopSparkProjetPMN

Prérequis

Dans mon cas j'ai IntelliJ IDEA installé sur Windows et Hadoop 3.3.2 et Spark 3.3.2 ansi que Scala 2.12.18 installés sur Ubuntu WSL (pendant les TP d'installation précédants) donc :

- Assurez-vous d'avoir Hadoop, Spark et Scala installées sur votre environnement, et notez bien leurs versions.
- Les chemins dans les scripts doivent être ajustés en fonction de votre configuration locale, veuillez trouver le détail dans ce qui suit:

Étape 1: Scripts Bash

Dans le dossier `bashFilesProjetPMN`, j'ai créé les fichiers bash necesssaires pour le lancement du projet et l'execution du .jar du projet dans spark.

Attention:

- Dans le fichier `run_spark.sh` :

> N'oubliez pas de remplacer `/chemin/local/data/*` par le chemin réel de vos fichiers de donnée, dans mon cas c'était:

```
hdfs dfs -put /home/simonlinux/datasetsProjetPMN/* /input/`
```

- Dans le fichier `copie_data_input.sh` :

> N'oubliez pas de remplacer `/chemin/vers/votre/application.jar` par le chemin réel de votre fichier JAR, dans mon cas c'était:

```
 /home/simonlinux/ProjetHadoopSpark-1.0-SNAPSHOT.jar
```

Étape 2: sur IntelliJ IDEA

Dans `pom.xml`, ajustez les versions de Hadoop, Spark et Scala selon votre installation, dans mon cas:

```
<properties>
    <encoding>UTF-8</encoding>
    <spark.version>3.3.2</spark.version>
    <scala.version>2.12.18</scala.version>
    <scala.compat.version>2.12</scala.compat.version>
</properties>
```

puis j'ai developpé le codes scala suivant le schema du projet suivant: ProjetHadoopSpark

```
└──src
    └──main
        └──scala
            ├──common
            │    ├──ApplicationProperties.scala
            │    └──Constants.scala
            ├──main
            │    ├──DataframeResult.scala
            │    └──Main.scala
            └──utils
                 ├──Utils.scala
                 └──UserDefinedFunction.scala
```

> Noubliez pas dans le fichier `ApplicationProperties.scala` de mettre à jour les valeurs des chemins d'entrée et sortie: inputPath et outputPath (où on va obtenir nos fichiers csv et parquet de sortie après traitement Spark) :

Rappel du contenu du fichier `ApplicationProperties.scala` :

```scala
package common

object ApplicationProperties {
    val inputPath = "/input/"
    val outputPath = "/output/"
    // On peut ajouter d'autres chemins ou propriétés si nécessaire
}
```

Ensuite, Générez votre `fichier.jar` via Maven (`Maven Projects > Lifecycle > clean > package`).

Étape 3 : Préparation des Fichiers et Scripts

- Assurez-vous d'avoir tous les fichiers `CSV` du projet dans `datasetsProjetPMN`.
- Placez votre `fichier.jar`, par exemple `ProjetHadoopSpark-1.0-SNAPSHOT.jar`, dans le répertoire approprié (voir plus haut).
- Assurez-vous que le dossier `bashFilesProjetPMN` contenant les `fichiers.sh` est dans le répertoire approprié de votre environnement d'exécution.

Remarques :

Après modification des scripts `.sh`, rendez-les à nouveau exécutables avec la commande :

```
chmod +x NomDeVotreScript.sh
```

Si vous modifiez vos scripts .sh sous Windows, ils peuvent contenir des caractères de fin de ligne Windows au lieu de Unix/Linux. Pour corriger cela, utilisez :

```
sudo apt-get update
sudo apt-get install dos2unix
dos2unix NomDeVotreScript.sh
```

Étape 3 : Lancement de l'Application

- Préparez votre environnement Hadoop :

```
sbin/start-dfs.sh
sbin/start-yarn.sh
```

- Exécutez `./scriptfinal.sh` depuis: `VotreCheminVers/bashFilesProjetPMN`. Le statut final doit indiquer `SUCCEEDED`, signifiant que votre application s'est terminée sans erreurs.

Importation des Fichiers de Sortie:

> Importez vos fichiers de sortie de HDFS vers votre système local avec les commandes :

```
hdfs dfs -get /output/result.csv /VotreChemin/local
hdfs dfs -get /output/result.parquet /VotreChemin/local
```

Simon BELFATMI
