TP 1 : Hadoop et HDFS

idir.benouaret@epita.fr

Introduction

Pour cette première séance de TP, vous allez travailler sur Hadoop, utiliser et programmer HDFS (Hadoop Distributed File System). C'est l'espace de stockage des fichiers distribués. Il ressemble à l'arbre et système de fichiers Unix et il y a des commandes très similaires pour manipuler les fichiers et les dossiers. Il existe aussi des APIs (Java et Python par exemple) qui permettent d'écrire des programmes de traitement des fichiers HDFS.

Le but de ce premier TP est d'apprendre les concepts et les commandes de bases afin de bien gérer les fichiers sur HDFS. Dans un second temps, vous utiliserez une API python pour la gestion des fichiers.

Exercices

Exercice I : Install party

Warning :

- La procédure d'installation décrite ci-dessous concerne les distributions Linux.
- Si vous êtes sur MacOs, la procédure devrait être similaire avec quelques exceptions. Par exemple votre fichier de configuration est .profile et non pas .bashrc
- Si vous êtes sur Windows : qu'attendez vous pour changer ?

Q1 – Vérifier que Java (11) est bien installé sur votre machine. Dans un terminal : `java -version`.

Q2 – Sinon : `sudo apt-get install openjdk-11-jdk`

Q3 – Configurer les variables d'environnement Java dans votre `.bashrc`

Sur ma machine perso, ça ressemble à ceci :

```
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
export PATH=$PATH:$JAVA_HOME/bin
```

Q4 – Télécharger la dernière version de Hadoop ici : `https://hadoop.apache.org/releases.html`.

Q5 – Extraire, copier le dossier dans votre répertoire home. Le renommer en simplement `hadoop` :

```
tar -zxvf ~/hadoop-3.3.6.tar.gz
mv hadoop-3.3.6 hadoop
```

Maintenant il est temps de configurer hadoop. Il faut suivre les étapes suivantes

1. Se mettre sur le répertoire dans lequel se trouvent les fichiers de configurations de hadoop : `cd hadoop/etc/hadoop`
2. Ouvrir le fichier `hadoop-env.sh`. Il est primordial de spécifier la variable d'environnement Java. Ajouter `export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64` (c'est exactement la même ligne que vous avez dans votre .bashrc)
3. Ouvrir le fichier `core-site.xml` et y mettre la configuration suivante :

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

4. Ouvrir le fichier `hdfs-site.xml` et y mettre la configuration suivante :

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
</configuration>
```

5. ouvrir le fichier `mapred-site.xml` et y mettre la configuration suivante :

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
  <property>
    <name>mapreduce.application.classpath</name>
    <value>$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/*:$HADOOP_MAPRED_HOME/
        share/hadoop/mapreduce/lib/*</value>
  </property>
</configuration>
```

6. ouvrir le fichier `yarn-site.xml` et y mettre la configuration suivante :

```
<configuration>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.nodemanager.env-whitelist</name>

    <value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,
        CLASSPATH_PREP END_DISTCACHE,HADOOP_YARN_HOME,HADOOP_MAPRED_HOME</value>
  </property>
</configuration>
```

Q6 – Vérifier que ssh et pdsh sont installés. Sinon :

```
sudo apt-get install ssh
sudo apt-get install pdsh
```

pdsh est un utilitaire qui permet d'envoyer la même ligne de commande sur plusieurs machines en parallèle. Très utile pour administrer un parc de serveurs Linux.

Q7 – Vérifiez que vous avez un accès `ssh` à votre localhost. quelle est la réaction de `ssh localhost` sur votre machine ?

Si jamais vous n'avez pas de clé ssh, suivre les étapes suivantes :

1. Créer votre clé publique ssh : `ssh-keygen -t rsa -P '' -f  /.ssh/id_rsa`
2. `cat  /.ssh/id_rsa.pub » /.ssh/authorized_keys` (Cela va copier votre clé publique dans le fichier des clés autorisées pour ne plus vous redemander de taper un mot de passe)
3. `chmod 0600  /.ssh/authorized_keys`

4. Tester `ssh localhost` puis exit quand ça marche

Q8 – Encore des configs (les toutes dernières) : dans votre .bashrc

```
export HADOOP_HOME=~/hadoop/
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
export HADOOP_STREAMING=$HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.2.3.jar
export HADOOP_LOG_DIR=$HADOOP_HOME/logs
export PDSH_RCMD_TYPE=ssh
```

Q9 – **Ne pas oublier de faire :** `source .bashrc`

Normalement, tout est prêt. Il faut maintenant formater le système de fichiers HDFS. Cette opération n'est à faire qu'une seule fois sinon il faudra dire adieu à vos données qui y sont stockées :

Q10 – Tapez la commande : `hdfs namenode -format`

Lancement de HDFS

vous êtes prêts à lancer le cluster hadoop : taper la commande `start-dfs.sh`. Ceci lance HDFS. Tapez la commande `jps` pour vérifier. Différents scripts bash pour lancer les services sont disponibles dans le répertoire sbin de hadoop. Par exemple, `start-all.sh` qui lance tous les services (déconseillé dans un vrai environnement de déploiement).

Il ne faut pas oublier d'arrêter tous les services, une fois qu'on a finit de les utiliser : `stop-all.sh`. C'est souvent la cause de certains bugs.

Exercice II : Manipulations sur HDFS

La partie intéressante du TP commence maintenant avec l'utilisation du système de fichier distribués HDFS.

Q1 – Voici quelques commandes de bases à découvrir. Elles vous seront certainement familières parce qu'elles sont similaires aux commandes Unix que vous connaissez déjà.

1. `hdfs dfs -ls` : Oups, il est fort possible que vous soyez confronté à votre première erreur sur hadoop.

Ce qui se passe est tout à fait normal. L'adresse par défaut de votre répertoire HDFS est `/user/votre_login` qui n'existe pas encore

2. `hdfs dfs -ls /` : par contre, cette commande affiche ce qu'il y a à la racine de votre répertoire HDFS. La commande n'affiche rien pour le moment car votre répertoire est vide.
3. Pour information. Il n'existe pas de commande équivalente à `cd`. En effet, la notion de répertoire courant n'existe pas dans HDFS. Donc à chaque fois il faudra mettre le chemin complet. Prenez cette habitude.
4. Il est maintenant temps de créer votre répertoire : `hdfs dfs -mkdir -p /user/votre_login`
5. Vérifiez que la création est bien faite : normalement `hdfs dfs -ls` fonctionne maintenant et renvoie le listing du répertoire (qui est encore vide)
6. `hdfs dfs -mkdir dossier` : crée un dossier dans votre espace HDFS, c'est à dire `/user/votre_login/dossier`. Faites des tests en créant quelques répértoires.
7. Vérifiez leurs existences avec `hdfs dfs -ls`

Q2 – Ouvrez un terminal, créez un nouveau document texte appelé `bonjour.txt`, mettez du texte dedans.

1. `hdfs dfs -put bonjour.txt` : cette commande copie un fichier local sur HDFS. Sans mettre de chemin, cela sera copié à la racine (`/user/votre_login`)
2. `hdfs dfs -put bonjour.txt dossier` copie le fichier sur HDFS dans le répértoire dossier
3. `hdfs dfs -cat bonjour.txt` : affiche le contenu
4. `hdfs dfs -tail bonjour.txt` : affiche le dernier Ko du fichier
5. `hdfs dfs -head bonjour.txt` : affiche le premier Ko du fichier
6. Supprimez ce fichier de HDFS par : `hdfs dfs -rm bonjour.txt`. Vérifiez qu'il n'est plus présent
7. Remettez à nouveau ce fichier par : `hdfs dfs -copyFromLocal bonjour.txt`
8. `hdfs dfs -chmod go+w bonjour.txt` (vérifiez son propriétaire, son groupe et ses droits avec hdfs dfs -ls)
9. `hdfs dfs -mv bonjour.txt dossier/bonjour.txt` (vérifiez avec `hdfs dfs -ls -R`)
10. `hdfs dfs -get dossier/bonjour.txt hello.txt` : transfère le fichier de HDFS vers votre machine locale en lui changeant son nom. Attention, cette commande ne serait pas à faire avec de vrai mégadonnées, à moins que votre machine dispose d'un espace de stockage infini :)
11. `hdfs dfs -cp dossier/bonjour.txt dossier/salut.txt` (vérifiez)
12. `hdfs dfs -count -h /dossier` : affiche le nombre de sous-dossiers, fichiers et octets occupés dans /dossier. Cette commande correspond à peu près à du -h dans Unix
13. `hdfs dfs -touchz toto.txt` : crée un fichier sur HDFS

La documentation de toutes les commandes est sur cette page : `https://hadoop.apache.org/docs/r2.8.4/hadoop-project-dist/hadoop-common/FileSystemShell.html`

Passez un peu de temps à regarder les commandes que vous n'avez pas encore tester.

Pour aller plus loin, je vous conseille la lecture de ce blog
`https://bradhedlund.com/2011/09/10/understanding-hadoop-clusters-and-the-network/`

Pour nommer les fichers dans un environnement mixte (local et HDFS), on peut utiliser la notation URI : hdfs :///nomcomplet. Par exemple hdfs dfs -ls hdfs :///dossier. On peut ainsi distinguer `file:///tmp/toto` (local) de `hdfs:///tmp/toto` (hdfs).

Exercice III : État du cluster

Les services Hadoop génèrent des pages web automatiquement pour permettre de suivre le fonctionnement. Cliquez sur ce lien : `http://localhost:9870` pour vous y connecter.

La page que vous voyez propose plusieurs liens vers différents services Hadoop. On va s'intéresser à HDFS pour ce premier TP. Accèdez avec votre navigateur à http ://localhost :9870 ; ça amène sur la page Overview. Dans le tableau Summary vous avez l'espace total, l'espace utilisé, l'espace libre, avec des valeurs en %, des liens vers les DataNodes vivants, morts ou en train de se désactiver (decommissioning nodes). Comme on est en mode pseudo-distribué vous remarquerez qu'il y a un seul datanode.

Tout en haut, il y a une barre verte avec différents liens. Cliquez sur Datanodes. Vous voyez la capacité et la charge de chaque DataNode. La colonne Last Contact indique le nombre de secondes depuis le précédent « battement de cœur » (heartbeat) envoyé par le Datanode au Namenode, c'est le terme employé pour un signal périodique de bon fonctionnement. Il y a un contact toutes les 3 secondes. Un Datanode est considéré comme mort lorsqu'il n'a pas donné signe de vie depuis 10 minutes. Vous pouvez rafraîchir pour voir l'évolution.

Toujours en haut, il y a un bouton 'Utilities' avec un item Browse the file system. Vous pouvez parcourir l'arbre des fichiers HDFS. Par exemple, descendez dans /user/votrelogin, il y a un message d'erreur tout en haut.

Exercice IV : Programmation Python avec l'API HDFS

Dans cette partie, vous allez programmer en langage Python. Utilisez votre IDE préféré et vérifiez que Python3 est bien installé sur votre machine. Si ce n'est pas le cas, installez Python3. Nous allons travailler avec une librairie python pour interagir avec *hdfs*. Pour l'installer tapez `pip3 install hdfs`. La documentation de la librairie est sur cette page web :
`https://hdfscli.readthedocs.io/en/latest/quickstart.html`. C'est une API qui permet de créer un connexion à HDFS à partir d'un programme Python.

Télécharger le fichier `arbres.csv` (moodle), et *jouer* avec l'api Hdfs :

- API reference :
  `https://hdfscli.readthedocs.io/en/latest/quickstart.html#python-bindings`
- connexion :
  `https://hdfscli.readthedocs.io/en/latest/api.html#hdfs.client.InsecureClient`

Listes non exhaustives des tâches à réaliser

Q1 – Créer un fichier `arbres.csv` depuis l'API

Q2 – Écrire un programme python qui écrit le fichier `arbres.csv` (local) dans votre fichier

`arbres.csv` (hdfs)

Q3 – Accès et affichage d'un fichier HDFS dans un programme Python.

Q4 – Afficher le contenu d'un répertoire.

Q5 – Afficher le fichier `arbres.csv` ligne par ligne

Q6 – Supprimer un fichier depuis l'API