TP 1 : Hadoop et HDFS

idir.benouaret@epita.fr

Introduction

Apache Hadoop est un framework open-source pour stocker et traiter les données volumineuses sur un cluster. Il est utilisé par un grand nombre de contributeurs et utilisateurs.

Pour cette première séance de TP, vous allez travailler sur Hadoop, utiliser et programmer sur 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**

Nous allons utiliser docker pour installer une image de hadoop. Les étapes d'installation suivantes sont largement adaptées de la page Lilia Sfaxi, elles-même basées sur le projet GitHub de Kai Liu

Installation

1. Télécharger l'image docker uploadée sur dockerhub :
   - `docker pull liliasfaxi/hadoop-cluster:latest`
2. Créer les trois contenaires à partir de l'image téléchargée. Pour cela :

(a) Créer un réseau qui permettra de relier les trois contenaires :

- `docker network create -driver=bridge hadoop`

(b) Créer et lancer les trois contenaires (les instructions -p permettent de faire un mapping entre les ports de la machine hôte et ceux du contenaire) :

- `docker run -itd -net=hadoop -p 9870:9870 -p 8088:8088 -p 7077:7077 -p 16010:16010 -name hadoop-master -hostname hadoop-master liliasfaxi/hadoop-cluster:latest`
- `docker run -itd -p 8040:8042 -net=hadoop -name hadoop-worker1 -hostname hadoop-worker1 liliasfaxi/hadoop-cluster:latest`
- `docker run -itd -p 8041:8042 -net=hadoop -name hadoop-worker2 -hostname hadoop-worker2 liliasfaxi/hadoop-cluster:latest`

(c) 2.3. Vérifier que les trois contenaires tournent bien en lançant la commande : `docker ps`

3. Entrer dans le contenaire master pour commencer à l'utiliser. Ce conteneur joue le rôle du **NameNode** et s'appelle `hadoop-master`

- `docker exec -it hadoop-master bash`

Lancement de HDFS

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 contenaire, est de lancer **hadoop** et **yarn**. Un script est fourni pour cela, appelé start-hadoop.sh. Lancer ce script.

- `./start-hadoop.sh`

Exercice II : **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.

Exercice III : **Manipulations sur HDFS**

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

1. `hdfs dfs -ls` : il est fort possible que vous soyez confronté à votre première erreur sur hdfs. Ce qui se passe est tout à fait normal. L'adresse par défaut de votre répertoire HDFS est `/user/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 le répertoire : `hdfs dfs -mkdir -p /user/root`
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/root/dossier`. Faites des tests en créant quelques répértoires.
7. Vérifiez leurs existences avec `hdfs dfs -ls`

Q2 – 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/root`)
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 IV : **Programmation Python avec l'API HDFS**

Dans cette partie, vous allez programmer en langage Python. 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