JDEV2015
MapReduce et Spark pour le calcul scientifique

Pierre Senellart (pierre.senellart@telecom-paristech.fr)

2 et 3 juillet 2015

Le but de cet atelier est de se familiariser avec les paradigmes de programmation de MapReduce (via son implémentation dans Hadoop) et de Spark. Le site de l'atelier comporte un lien vers une machine virtuelle Virtualbox comprenant une installation de Hadoop et de Spark prête à l'emploi. En raison de limitations liées à l'environnement de l'atelier, ces installations fonctionnent en mode « pseudo-distribué » : les données et calculs sont sur une seule et même machine, mais l'accès aux données et l'exécution du calcul se fait de la même manière qu'en mode distribué, avec des accès réseaux simulés par une connexion à `localhost`.

1 Prise en main de la machine virtuelle

Si possible, augmenter la mémoire allouée à la machine virtuelle (2 Go serait idéal). Démarrer la machine virtuelle (en cas d'impossibilité de démarrage graphique, mettre à jour votre version de Virtualbox en suivant les instructions de la page `https://www.virtualbox.org/wiki/Downloads`). Le système est un Debian Linux, avec un utilisateur préconfiguré, `user`, dont le mot de passe est `jdev2015`. Cet utilisateur est *sudoer* mais il ne devrait pas être nécessaire d'utiliser les droits d'administrateur au cours de cet atelier.

Une fois la machine démarrée, utiliser le script `start_hadoop.sh` dans un terminal (raccourci sur le bureau) pour lancer le serveur Hadoop au nom de l'utilisateur `user`. Vous pouvez consulter les URL suivantes pour des informations sur la configuration HDFS (système de fichier distribué) et sur les jobs MapReduce, respectivement : `http://localhost:50070/` et `http://localhost:8088/`.

Vous pouvez tester l'installation HDFS en tapant :

```
hadoop fs -ls /
```

dans un terminal. Vous devriez voir le contenu du répertoire racine du système de fichiers HDFS. Vous pouvez utiliser `hadoop fs` pour manipuler en ligne de commande les fichiers sous HDFS. Voir la documentation avec `hadoop fs -help`.

Nous utiliserons le langage Python pour ce TP. Nous recommandons l'utilisation d'Eclipse avec le package PyDev, préinstallé dans la machine virtuelle, mais il est également possible d'utiliser un éditeur de texte et de lancer les jobs Python en ligne de commande.

2 Premiers exemples MapReduce et Spark

Dans l'espace de travail Eclipse (`$HOME/workspace`) vous sont fournis deux projets Python dont le but est, en MapReduce et en Spark, d'extraire les mots ayant plus de 100 occurrences dans la

version originale de *Alice aux pays des merveilles*. Lire en détail les codes des deux programmes pour comprendre leur fonctionnement.

Schématiquement, un programme MapReduce est formé de trois composants principaux :

— un *mapper*, qui traite l'entrée pour produire des paires clef/valeur ;

— un *reducer*, qui agrège d'une certaine manière les valeurs associées à la même clef ;

— un *driver*, qui lance le programme MapReduce en soumettant mapper et reducer à l'environnement d'exécution.

Cela est simplifié, car un vrai programma MapReduce pourra avoir plusieurs mappers ou reducers. Il est également courant qu'un driver fasse appel à une séquence d'étapes Map/Reduce. À l'issue de chaque étape, le résultat partiel est sérialisé sur HDFS et doit donc être relu à l'étape suivante. MapReduce s'occupe de manière transparente de la distribution du calcul des mappers et des reducers, et du regroupement (*shuffle*) nécessaire entre les deux.

Spark fonctionne en s'appuyant sur la notion de *Resilient Distributed Dataset* (RDD), des objets abstraits correspondant à un ensemble de traitement à effectuer sur des données. Chaque opérateur de Spark va construire un nouveau RDD à partir des RDD existants, mais ceux-ci ne seront matérialisés qu'à l'appel d'un ordre `.collect()` ou similaire. Spark s'occupe de distribuer les calculs effectués par chaque opérateur, et de faire communiquer les opérateurs entre eux pour minimiser les surcoûts liés à des sérialisations/déserialisations successives.

Lancer les jobs MapReduce et Spark : dans Eclipse, le driver MapReduce peut être lancé comme un programme ordinaire, et le programme Spark peut être lancé grâce à un *External Tool* préconfiguré (dans la barre d'outils, le bouton à droite du bouton permettant de lancer un programme classiquement). Comparer les sorties, et jeter un œil aux messages d'information et de debug.

La version d'Hadoop configurée présente un bug qui entraîne l'impossibilité de supprimer des fichiers HDFS dans certaines circonstances, cf. `https://issues.apache.org/jira/browse/HDFS-8179`. *Pour contourner ce problème si vous y êtes confronté, remplacer les appels :*

```
hadoopy.rmr(output_path)
```

ou similaires par :

```
hadoopy.rmr("-skipTrash %s"%output_path)
```

Vous pouvez consulter la documentation du module Python pour Hadoop/MapReduce sur `http://hadoopy.readthedocs.org/` et de Spark sur `https://spark.apache.org/docs/latest/programming-guide.html`.

3 Vos propres programmes MapReduce et Spark

Il est temps de passer à l'élaboration de vos propres programmes MapReduce et Spark, pour comparer ces deux technologies. Nous vous proposons un cas d'utilisation ; cependant, si vous avez un calcul (relativement simple) à réaliser sur un jeu de données qui vous tient à cœur, vous pouvez également tenter de l'implémenter en MapReduce et Spark.

Créez de nouveaux projets pour vos programmes, et inspirez-vous tout au long du développement des exemples fournis.

Cas d'utilisation : calcul de PageRank dans Simple English Wikipedia

PageRank est la technique qu'ont proposé les fondateurs de Google, Brin et Page, pour associer un score aux pages du Web. L'idée de PageRank est la suivante : *les pages importantes sur le Web sont les pages étant pointées par des pages importantes*. Plus généralement, le score de PageRank est une

mesure utile à calculer dans tout graphe orienté, et peut révéler des informations sur l'importance et la centralité des nœuds de ce graphe.

PageRank est défini comme la probabilité qu'un *surfeur aléatoire* effectuant une marche aléatoire sur le Web en suivant les liens uniformément au hasard (et, avec une faible probabilité, effectuant un saut vers une autre page du Web choisie uniformément au hasard) se retrouve sur une page donnée dans un point distant du futur (une fois que la *mesure d'équilibre* de la chaîne de Markov a été atteinte).

Étant donné un graphe orienté de matrice d'adjacence G ($G(i,j)$ vaut 1 s'il y a un lien de i vers j, 0 sinon), le PageRank des nœuds du graphe peut être calculé de la manière suivante :

1. Normaliser G pour que chaque ligne somme à 1.
2. Soit u le vecteur uniforme de somme 1, soit v égal à u.
3. Répéter jusqu'à convergence (par exemple différence relative inférieure à 1% entre les versions successives de v) :
   — $v := (1-d)^t Gv + du$ (avec par exemple $d = \frac{1}{4}$).

Vous trouverez sur la page de l'atelier un jeu de données formé du graphe de la version Simple English (voir `http://simple.wikipedia.org/`) de Wikipedia. Ce jeu de données est formé d'un ensemble de titre d'articles (un par ligne) et d'un ensemble d'arêtes, décrit par un fichier dont chaque ligne est de la forme :

```
A B1,C1 B2,C2 ... Bn,Cn
```

où `A` est l'index d'un article (le fichier des titres d'articles les donnant dans l'ordre), `B1`, ..., `Bn` sont des index d'articles pointés par `A`, et `C1`, ..., `Cn` sont le nombre de liens de `A` à l'article en question.

En utilisant MapReduce (respectivement, Spark), le but est de calculer le PageRank de l'ensemble des nœuds du jeu de données, et de trier le résultat par PageRank décroissant.

En particulier vous devrez :

— charger le jeu de données dans HDFS, sous un format lisible par Hadoop et Spark (le format `SequenceFile` est recommandé pour Hadoop, il peut être produit avec `hadoop.writetb`, cf. `http://hadoopy.readthedocs.org/en/latest/tutorial.html` ; pour Spark il est possible d'utiliser directement le format texte) ;

— écrire la multiplication matricielle sous la forme d'un job MapReduce ou d'une succession d'opérateurs Spark ;

— écrire la structure générale du programme faisant appel à ces jobs jusque convergence ;

— trier et interpréter le résultat.

Quel est l'article le plus important dans Simple English Wikipedia ?