Fouille de données et Medias Sociaux

M2 DAC

TME 5. Spark

Ce TME a pour objectif de se familiariser avec le système de traitement de données Spark. Spark permet la définition de traitements parallèles simplifiée par rapport au populaire Hadoop MapReduce, et améliore significativement les performances dans de nombreux cas, notamment pour les processus itératifs. Dans un premier temps, nous considèrerons l'execution d'un programme exemple sur le système Spark, afin de se familiariser avec la commande de soumission de process. Ensuite, nous proposons de définir un programme simple : le fameux WordCount traité dans le TME précédent avec Hadoop MapReduce. Enfin, nous proposons de nous intéresser à l'apprentissage d'un classifieur linéaire avec Spark.

1 Environnement de travail

Afin de mettre en place l'environnement de travail pour Spark, tapez les commandes suivantes sur la console de votre machine:

```
mkdir /tmp/FDMS
cp /Vrac/lamprier/FDMS/spark-1.5.2-bin-hadoop2.4.tgz /tmp/FDMS
cd /tmp/FDMS
tar zxvf spark-1.5.2-bin-hadoop2.4.tgz
export SPARK_HOME=/tmp/FDMS/spark-1.5.2-bin-hadoop2.4
cd $SPARK_HOME
mkdir /tmp/FMDS/spark-events
echo spark.eventLog.enabled true >> conf/spark-defaults.conf
echo spark.eventLog.dir file:/tmp/FDMS/spark-events >> conf/spark-defaults.conf
echo spark.history.fs.logDirectory file:/tmp/FDMS/spark-events >> conf/spark-defaults.conf
./sbin/start-history-server.sh
```

Une fois ces commandes entrées, vous devez avoir un répertoire de travail *$SPARK_HOME* pointant sur spark ainsi qu'une répertoire */tmp/FMDS/spark-events* visant à contenir les logs d'execution. La dernière commande lance un serveur permettant d'analyser ces logs. Dans un navigateur, ouvrez l'URL *http://localhost:18080/*. Vous devez avoir accès à ce serveur qui liste les différentes applications qui ont été lancées sur spark (pour le moment aucune).

2 Prise en Main de Spark

La commande *spark − submit* permet de soumettre un processus à Spark. Testez la commande suivante qui lance un exemple de processus calculant une approximation de π en 10 iterations:

```
$SPARK_HOME/bin/spark-submit --master local examples/src/main/python/pi.py 20
```

Si tout se passe bien, cette commande permet de lancer le programme exemple JavaSparkPi et affiche le resultat. L'option *–master local* indique à Spark de lancer le programme en local en utilisant un thread unique (par defaut). Vous pouvez tester *–master local[N]* avec différents nombres de threads en parallèle N et observer ce qu'il se passe grâce au serveur d'analyse de logs. Notez qu'ici on ne lance le processus qu'en local mais que Spark est prévu pour être lancé sur n'importe quel type de cluster.

3 Un premier programme Spark: WordCount

Il s'agit dans un premier temps de s'interesser à un programme simple que l'on connait bien: le fameux WordCount discuté en cours sous la forme d'un programmme MapReduce. L'idée est alors de transposer le programme en Spark pour se familiariser avec les librairies de la plateforme.
Pour le développement de WordCount, il s'agit de :

- Construire un RDD à partir des lignes d'un fichier texte passé en entrée
- Eventuellement transformer ce RDD pour obtenir un RDD liste de mots
- Pour chaque mot, emettre un couple <mot,1>
- Appliquer une reduction par clé permettant de sommer les 1 de tous les couples de même clé (on pourrait de la même manière utiliser la fonction *countByKey* de Spark).

Note : la page *https://spark.apache.org/docs/latest/programming-guide.html* donne un bon aperçu des fonctions disponibles sous Spark.

4 Classification linéaire

Nous proposons de considérer un problème de classification de pages Web. Nous nous intéresserons aux jeu de données situé à l'adresse */Vrac/lamprier/FDMS/WebKB*, que vous copierez chez vous. Ce jeu de données contient notamment dans son repertoire *content* quatre fichiers correspondant à des pages de différentes universités américaines. Les pages sont identifiées par leur adresse Web, disposent d'un vecteur de caractéristiques binaires et d'une classe d'appartenance. Il s'agit d'aprendre un classifieur linéaire capable de classer ces pages dans leur classe de réference.
Etant donné le nombre de classes possibles, nous proposons de définir autant de classifieurs binaire que de classes. La fonction $f_c(x)$ suivante doit alors retourner un score positif si x appartient à la classe c, négatif sinon:

$$f_c(x) =< w_c, x > +b_c$$

Pour l'apprentissage de cette fonction nous considèrerons un coût de classification type SVM (HingeLoss) :

$$L(f_c(x), y) = max(0, 1 - f_c(x) \times y)$$

Une fois l'apprentissage effectué sur un sous-ensemble des données, il s'agit d'évaluer les performances du classifieur sur un ensemble de test.

Note: Afin d'éviter des biais dûs à la sur-représentation des éléments négatifs dans l'apprentissage, nous proposons de *sampler* les éléments négatifs de manière à n'en considérer qu'une sous-partie, dont la taille est égale au nombre de représentants pour la classe considérée.

5 Diffusion: IC

Proposer une parallèlisation Spark de l'apprentissage du modèle diffusion IC.