CES Data Scientist, Télécom ParisTech

Index inversé en MapReduce et en Spark

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

9 November 2015

Le but de ce TP est d'implémenter la construction d'un index inversé en MapReduce et en Spark, donc de permettre de paralléliser et passer à l'échelle ce qui a été fait à la session précédente.

Pour évaluation de la session, vous devez envoyer à Pierre Senellart <pierre.senellart@telecom-paristech.fr> vos programmes Python au plus tard le 6 décembre novembre (pénalités en cas de rendu après cette date, et aucun rendu accepté à compter du 13 décembre).

Nous réutilisons l'installation de la machine virtuelle utilisée pour la session précédente.

1 Configuration préliminaire

Pour corriger le problème de mauvais fonctionnement de Spyder, taper « `sudo apt-get remove python-matplotlib` » dans un terminal (le mot de passe demandé est `cesds2015`).

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)
```

Pour permettre à Spark de directement lire des tables HBase il faut ajouter une variable d'environnement à la configuration, avec :

```
export SPARK_CLASSPATH=
  /usr/local/stow/hbase-1.1.2/lib/hbase-common-1.1.2.jar:
  /usr/local/stow/hbase-1.1.2/lib/hbase-client-1.1.2.jar:
  /usr/local/stow/hbase-1.1.2/lib/hbase-server-1.1.2.jar:
  /usr/local/stow/hbase-1.1.2/lib/guava-12.0.1.jar:
  /usr/local/stow/hbase-1.1.2/lib/htrace-core-3.1.0-incubating.jar:
  /usr/local/stow/hbase-1.1.2/lib/protobuf-java-2.5.0.jar:
  /usr/local/stow/hbase-1.1.2/lib/hbase-protocol-1.1.2.jar:
  /usr/local/stow/spark-1.4.0-bin-hadoop2.6/lib/
  spark-examples-1.4.0-hadoop2.6.0.jar
```

(sans espace ni retour à la ligne). Pour rendre ce changement permanent dans tous vos terminaux, vous pouvez mettre cette commande dans un fichier `.bashrc` dans votre compte (un tel fichier est disponible sur le site pédagogique).

2 Tutoriel de Hadoopy

Lisez attentivement l'ensemble du tutoriel de l'interface Hadoopy `http://www.hadoopy.com/en/latest/tutorial.html`. Vous pourrez par la suite également utiliser la documentation de l'interface de programmation, disponible à `http://www.hadoopy.com/en/latest/api.html`.

3 De la table simplewiki à un fichier HDFS

Hadoopy gère encore mal la lecture directe depuis une table HBase (ça n'est pas un problème avec l'interface Hadoop Java). Commencez donc par écrire un script Python mettant l'intégralité de la table HBase contenant le contenu de Simple English Wikipedia dans un fichier sur HDFS au format obtenu par l'appel `hadoopy.writedb`.

4 Construction de l'index inversé

Implémentez la construction de l'index inversé en MapReduce avec l'interface Python Hadoopy. Le fichier d'entrée est le fichier que vous venez de créer sur HDFS. La sortie doit aussi être placée sur HDFS, dans votre répertoire personnel. Pour faciliter le débogage :

— Commencez par utiliser `hadoopy.launch_local` plutôt que `hadoopy.launch` qui simule une exécution MapReduce au sein du même processus

— Limitez vous au 1000 premières entrées de la table simplewiki, en modifiant le script que vous avez créé à la question précédente à cet effet.

Une fois que votre programme fonctionne, enlevez ces deux restrictions et tester.

5 Chargement du résultat dans HBase

Écrivez un programme Python pour charger le fichier résultat dans HBase, pour produire une table au même format que celle obtenue au TP précédent. Fixez `batch_size=1000` pour forcer des batchs de taille raisonnable. Vérifier que le contenu de la table semble correct.

6 Spark

Programmez maintenant la construction de l'index inversé à l'aide de Spark.

Pour exécuter un programme Spark Python, faire : `spark-submit monprogramme.py`

On obtiendra un contexte Spark `sc` avec : `sc=SparkContext()`

Se reporter à la documentation de Spark (`http://spark.apache.org/docs/latest/programming-gu` `html`) pour les fonctions s'appliquant à un contexte Spark et à un RDD.

Dans un premier temps, pour simplifier, vous pouvez, comme pour MapReduce, lire l'entrée dans un fichier HDFS et produire la sortie sur HDFS. Dans un second temps, lisez l'entrée directement depuis HBase (voir ci-dessous) et écrire la sortie directement dans HBase (à l'aide de l'action `forEachPartition` d'un RDD, qui est exécutée une fois pour chaque partition du RDD).

Pour lire une table HBase (nommée `simplewiki`) comme un RDD (formé pour chaque ligne, de la clef et de la colonne `wiki:text`), utilisez les instructions suivantes :

```
hbaseConfig={"hbase.mapreduce.inputtable":"simplewiki",\
    "hbase.mapreduce.scan.columns":"wiki:text"}

table=sc.newAPIHadoopRDD(
```

```
'org.apache.hadoop.hbase.mapreduce.TableInputFormat',
'org.apache.hadoop.hbase.io.ImmutableBytesWritable',
'org.apache.hadoop.hbase.client.Result',
keyConverter=\
  "org.apache.spark.examples.pythonconverters.ImmutableBytesWritableToStringConverter",
valueConverter=\
  "org.apache.spark.examples.pythonconverters.HBaseResultToStringConverter",
conf=hbaseConfig)
```