UPMC
SORBONNE UNIVERSITÉS

CoDel - 5I451

TP 02 – Mise en œuvre du MapReduce.

Julien Sopena

Après après avoir étudié dans le premier TP, le fonctionnement de la plate-forme *Hadoop* à l'aide d'exemples fournis, nous allons maintenant mettre en œuvre le modèle *MapReduce* pour résoudre plusieurs types de problèmes. Ainsi, autour du calcul de nombre Π, nous verrons un exemple de calcul numérique et un exemple d'analyse de fichier. L'indexation et la parallélisation de calculs sont en effet très propices à l'utilisation du modèle *MapReduce*.

Remarque:

Ce TP suppose que vous ayez réglé votre environnement de développement et d'exécution comme indiqué dans le premier TP. Il utilise les mêmes sources (disponibles dans le repertoire `/Vrac/Sopena/PSIA-TP_Hadoop`) et notamment l'archive .

Exercice 1 : Calcul de Π en *MapReduce*

Nous allons commencer par calculer la valeur de Π grace à une méthode de Monte Carlo. Cette approche probabiliste du calcul de Π considère :

— un repère orthonormé $(O, \vec{i}, \vec{j})$;
— un cercle de diamètre d centré sur O l'origine du repère ;
— un carré de coté d centré sur O.

L'aire A_{cercle} du cercle peut s'exprimer :

$$A_{cercle} = \Pi * rayon^2 \Leftrightarrow A_{cercle} = \Pi * \frac{d^2}{4} \Leftrightarrow \Pi = 4 * \frac{A_{cercle}}{d^2} \Leftrightarrow \Pi = 4 * \frac{A_{cercle}}{A_{carre}}$$

En discrétisant par un nombre suffisamment grand de points l'aire du carré, on peut obtenir une bonne approximation du rapport des aires et donc de la valeur de Π. Ainsi, la méthode consiste à tirer aléatoirement des points se situant dans l'espace $[-\frac{d}{2}, \frac{d}{2}] * [-\frac{d}{2}, \frac{d}{2}]$, puis à calculer la proportion des points appartenant au cercle de diamètre d. La valeur de Π s'obtient alors en multipliant par 4 ce ratio.

Pour savoir si un point P de coordonnées $[x, y]$ appartient au cercle, il suffit de calculer sa distance par l'origine du repère ($[0, 0]$) en utilisant le théorème de Pythagore. Ainsi $Distance(O, P) = \sqrt{x^2 + y^2}$. Si cette distance est supérieure à d, le point n'est pas dans le cercle.

Application au *Hadoop*

Chaque *map* va prendre un nombre fixe de points choisis aléatoirement et équitablement répartis dans le carré et déterminer pour chaque point s'il est dans le cercle ou pas. Les maps transmettent leur résultats aux reduces qui additionneront le nombre de points présents dans le cercle. Le programme appelant le job doit ensuite parcourir l'ensemble des résultats produits par les reduces, les additionner et calculer Π selon la formule décrite ci-dessus. Les maps choisissent leur points aléatoirement en initialisant une séquence de Halton. Une séquence de Halton est une séquence de réels compris entre 0 et 1. Le diamètre du cercle ne peut dans ce cas pas dépasser 1. Pour éviter que les maps choisissent tous les mêmes points, la graine qui initialise la séquence de Halton doit être propre à chaque map.

Question 1

Comment peut-on faire pour passer une graine différente à chaque map ?

Question 2

Identifiez les types des clés et valeurs des entrées des maps.

Question 3

Sachant que les maps calculent si un point est dans le cercle ou pas, quelle sera le type de sortie des clés et valeurs des maps ?

Question 4

Sachant que les reduces comptent en fonction de la sortie des maps, le nombre total de points qui sont dans le cercle, quel sera le type de sortie des clés et valeurs des reduces ?

Question 5

Liez chaque type à une classe de l'API *Hadoop*.

Question 6

Écrivez en JAVA le code d'un *map*. Pour cette question, comme pour les suivantes, il est fortement déconseillé d'utiliser les classes dépréciées de l'API.

Question 7

Écrivez en JAVA le code d'un *reduce*.

Question 8

Écrivez un programme JAVA *MapReduce* qui prendra en paramètre le nombre de maps, le nombre de reduces ainsi que le nombre de points par map. Le programme devra configurer les fichiers d'entrée des maps, soumettre le job, récupérer et fusionner les résultats des reduces, calculer Π et afficher sa valeur.

Pour faire cet exercice, vous utiliserez le squelette MyPiEstimator.java qui vous a été fourni dans avec les sources du TP, ainsi que la classe HaltonSequence.java.

Conseil : Désactiver l'option exécution spéculative des maps et des reduces qui est activée par défaut grâce aux lignes de code suivantes :

```
conf.setBoolean("mapred.map.tasks.speculative.execution",false);
conf.setBoolean("mapred.reduce.tasks.speculative.execution", false);
```

Question 9

Le programme vous donne la possibilité de paramétrer le nombre de reduces. Est-ce vraiment nécessaire ?

Exercice 2 : Un peu de poésie, dans ce nombre de brute.

Si, comme on l'a vu dans l'exercice précédent, l'approche *MapReduce* peut servir de modèle pour paralléliser la résolution de problèmes numériques, elle trouve aussi toute son efficacité dans les problèmes textuels. Elle est ainsi utilisée par les moteurs de recherche pour indexer les pages web.

Dans cet exercice nous allons concevoir un programme *MapReduce* qui compte le nombre de lettre des mots d'un texte. Ainsi, nous voulons :

1. prendre en entrée : un fichier contenant un texte.
2. retourner en sortie : une suite numérique, où chaque chiffre correspond dans l'ordre au nombre de lettre des mots du textes (sans tenir compte de la ponctuation).

Par exemple si en entrée on a un fichier contenant le texte suivant :

```
Super! Voici mes premiers pas avec Hadoop.
```

La séquence numérique de sortie devra être :

```
5538346
```

Question 1

Définissez le traitement des tâches *maps* et *reduces* qui permettent de mettre en œuvre l'approche *MapReduce* pour résoudre ce problème.

Sachant que l'on aura qu'une instance de la tâche reduce, vous chercherez a maximiser la parallélisation du traitement.

Indication : Il est possible de surcharger la fonction `cleanup` des classes `Mapper` et `Reducer`. Cette méthode est exécutée lorsque la tâche se termine.

Question 2

Implémentez dans *Hadoop* de votre approche. Pour la tester, vous utiliserez le fichier pourrez scinder le fichier poeme.txt se trouvant dans le repertoire */Vrac/Sopena/PSIA-TP_Hadoop/data*. Vous pouvez, le cas échéant, scinder ce fichier en plusieurs sous-fichiers.

Exercice 3 : *Hadoop*, et si on utilisait vraiment plusieurs machines

Question 1

Vous testerez en dernier lieu les différents programmes *Hadoop* produits ci-dessus avec plusieurs machines en configurant le fichier conf/slaves. Coordonnez-vous avec d'autres binômes pour utiliser leurs machines.

