TP - Map-Reduce

Rappels Unix (*''Le terminal, c'est pas facile...''*)

Mais si... Quelques rappels en vrac :

`cd dir`	se déplacer vers le répertoire `dir` (qui doit exister dans le répertoire courant).
`cd ..`	se déplacer dans le répertoire parent.
`mkdir dir`	créer le répertoire `dir` dans le répertoire courant.
`pwd`	afficher le répertoire courant.
`ls`	afficher le contenu du répertoire courant.
`python script.py`	exécuter le fichier python `script.py` (qui doit exister dans le répertoire courant !).
`rm script.py`	supprimer le fichier `script.py`.
`rm -R dir`	supprimer le répertoire `dir`.
`echo "bonjour"`	Afficher bonjour à l'écran.
`cat file`	Afficher le contenu du fichier `file` à l'écran.

Premier programme Map-Reduce : comptage

Pour la première partie de ce TP nous allons implémenter le *hello world* du monde Map-Reduce: le comptage de mots.

Dans un premier temps on utilise le pipe de liniux pour enchainer les traitements à la manière de ce qui est fait dans un programme distribué.

Récupérer les données

Nous allons utiliser des livres électroniques accessibles sur le projet Gutenberg. Téléchargez les 3 livres suivants (au format texte, lien *Plain Text UTF-8*) :

- The Outline of Science, Vol. 1 (of 4) by J. Arthur Thomson
- The Notebooks of Leonardo Da Vinci
- Ulysses by James Joyce

Map

Nous allons écrire le *mapper* pour notre tâche de comptage. Le rôle du mapper est d'émettre le couple (**mot**,1) pour chaque **mot** du document qui lui est donné en entrée. Nous allons écrire ce

mapper en python, le document sera donné via l'entrée standard et l'émission du couple (clé,valeur) consistera à afficher

```
clé valeur
```

à l'écran.

Quelques rappels et infos utiles sur python :

`print "bonjour"`	affiche bonjour à l'écran (i.e. écrit bonjour sur la *sortie standard*.
`s = "Longtemps je me suis"` `L = s.split(" ")`	la méthode `.split(" ")` découpe une chaîne de caractère en mots. On obtient donc `L = ["Longtemps", "je", "me", "suis"].`
`line.strip()`	supprime le retour à la ligne à la fin de la chaîne de caractères `line`

`import sys`
`for line in sys.stdin:`
`  print line`
	affiche chaque ligne de l'entrée standard à l'écran.

Vous avez maintenant tous les outils pour écrire le *mapper*. Utilisez pycharm (pycharm.sh) ou geany (clic droit sur le bureau -> Debian -> Applications -> Programming) pour écrire le mapper en python que vous sauvegarderez sous le nom `mapper.py` dans le répertoire `comptage`. Utilisez les commandes suivantes pour vérifier que votre *mapper* fonctionne correctement :

```
mbds@mbds:~/TP1/comptage$ echo "longtemps je me suis je me suis je me" | python mapper.py
longtemps 1
je 1
me 1
suis 1
je 1
me 1
suis 1
je 1
me 1
mbds@mbds:~/TP1/comptage$ echo "" | python mapper.py
mbds@mbds:~/TP1/comptage$ echo " " | python mapper.py
```

- Il est normal que les deux dernières commandes n'affichent rien, si ce n'est pas le cas il y a un problème dans votre code :

    - Si la deuxième commande affiche quelque chose, vous n'avez pas supprimé les retours à la ligne
    - Si la troisième commande affiche quelque chose, vous devez modifier votre code pour ne pas émettre de couple (clé,valeur) pour le mot vide (i.e. "").
- `cmd1 | cmd2` permet de faire un *pipe*, c'est à dire de d'utiliser ce que retourne `cmd1` comme entrée de `cmd2` via l'entrée standard.

Shuffle

L'étape de *shuffle* consiste à rassembler les couples (clé,valeur) par clés, pour nos tests il suffira de trier la sortie du *mapper* en utilisant la commande unix

```
mbds@mbds:~/TP1/comptage$ echo "longtemps je me suis je me suis je me" | python mapper.py | sort
je 1
je 1
je 1
longtemps 1
me 1
me 1
me 1
suis 1
suis 1
```

Cette phase de map/reduce sera gérée par hadoop, donc pas de code à écrire pour le *shuffler*.

Reduce

Le rôle du *reducer* pour la tache de comptage est de sommer les valeurs correspondant à une même clé est d'émettre le couple (clé,somme). L'entrée donnée au *reducer* est une liste de (clé,valeur) triée par clés. Chaque ligne de l'entrée contiendra un couple `clé valeur` séparé par un espace comme dans l'exemple ci dessus. Pour cet exemple le *reducer* devra afficher :

```
mbds@mbds:~/TP1/comptage$ echo "longtemps je me suis je me suis je me" | python mapper.py | sort| python reducer.py
je 3
longtemps 1
me 3
suis 2
```

Ici encore l'entrée sera donné via l'entrée standard et l'émission du couple (clé,valeur) consistera à l'afficher à l'écran (i.e. l'écrire sur la sortie standard).

Ecrivez le *reducer* en python, sauvegardez votre programme sous le nom `reducer.py` et tester le avec les commandes suivantes :

```
mbds@mbds:~/TP1/comptage$ cat /usr/share/doc/python-
```

```
matplotlib/README.txt | python mapper.py | sort | python
reducer.py
: 2
:: 1
* 2
*** 6
# 3
10.3.9 1
2.5 1
2.6 1
2D 1
8.4.x 1
a 5
across 1
Agg 1
(ala 1
all 1
also 1
and 6
any 1
Apple's 1

...

which 1
wish 2
with 1
WXAgg 3
wxPython 1
X 1
X. 1
you 5
You 1
your 1
```

Comptage Mot le plus fréquent

En vous inspirant des programmes précédents écrivez un mapper et un reducer pour déterminer et afficher le mot le plus fréquent dans les textes passés en input du programme.