TDLOG – séance n° 5
Concurrence & interopérabilité : TP

Xavier Clerc – xavier.clerc@enseignants.enpc.fr
Cédric Doucet – cedric.doucet@inria.fr
Thierry Martinez – thierry.martinez@inria.fr

21 octobre 2015

À rendre au plus tard le 18 novembre 2015

L'objectif de cette séance de TP est d'écrire un programme *Python* analysant des données au format json réparties dans de nombreux fichiers répartis dans plusieurs répertoires.

1 Version séquentielle

1.1 Données

Les données utilisées pour ce TP sont des données de l'IGN disponibles en *open data* à l'adresse :

`http://professionnels.ign.fr/geofla`

Elles sont disponibles au format json[1] sur les clefs USB mises à disposition durant le TP. Le fichier `data.zip` contient 37 répertoires (nommés `00`, `01`, *etc.*) contenant, à l'exception du dernier, 1000 fichiers au format json. Chaque fichier contient des données sur une commune de France.
Chaque fichier de commune contient des informations telles que :

- sa région (champ `"nom_region"`) ;
- son département (champ `"nom_dept"`) ;
- son statut (champ `"statut"`) ;
- son nombre d'habitants en milliers (champ `"population"`).

1.2 Extraction d'information

L'objetif de ce TP est de charger les données de l'ensemble des fichiers afin de pouvoir répondre aux questions suivantes :

1. D'après une conversion effectuée par `opendatasoft.com`.

1. combien existe-t-il de statuts de communes ?
2. pour chaque statut, combien de communes ont ce statut ?
3. pour chaque département, quel est le nombre d'habitants ?
4. pour chaque région, quel est le nombre d'habitants ?
5. quel est le nombre moyen de communes par département ?
6. quel est le nombre moyen de départements par région ?
7. quel est le nombre moyen de communes par région ?

2 Version parallèle

Une fois la version séquentielle programmée, il est intéressant de proposer une version parallèle afin de réduire les temps de traitement. Il faut dans un premier temps identifier les parties du programme qui peuvent être exécutées en parallèle, puis utiliser des modules de la libraririe de *Python* pour paralléliser certains traitements.

Ensuite, il est important de s'assurer que : (i) les deux versions du programmes donnent des résultats équivalents et (ii) la version parallèle est plus rapide. Deux points d'attention sur la question des performances :

1. l'accélération due à la parallélisation est (le plus souvent) inférieure au nombre de cœurs (du fait des portions de code non parallélisées, des communications ajoutées, *etc.*) ;
2. le temps d'exécution d'un programme sollicitant beaucoup le système de fichiers est **très** dépendant des *caches* du disque et du système d'exploitation.

Pour tenir compte du second point, il faut prendre garde à ne mesurer les temps d'exécution qu'*à chaud* (*i. e.* avec les caches remplis des informations pertinentes). Dans notre cas, on ne mesurera le temps qu'après 3 ou 4 exécutions, et on fera attention à ne pas avoir en arrière plan une application susceptible de *perturber* l'exécution de notre programme (*p. ex.* antivirus, système de *backup*).