Master MIAGE
Construction d'Applications Réparties – Année 2017/18
TP 2 - Compteur Parallèle

Giuseppe Lipari

1 Compteur Parallèle

Objectif : Compter le nombre d'occurrences des mots dans un texte en utilisant plusieurs threads (en Java).

Il faut concevoir un programma Java que prend en entrée un fichier texte et trouve le mot le plus fréquent dans le texte.

1.1 Compteur séquentiel

On va d'abord réaliser une version séquentielle.

On commence par écrire une fonction qui lit un fichier ligne par ligne, et produit un `ArrayList` de Strings. Pour découper une ligne en mots, on peut utiliser la méthode `split()` de la classe `String`, comme dans l'exemple suivant :

```
String txt = new String("this is a test");
String[] result = txt.split("\\s");
for (int x=0; x<result.length; x++)
   // utilise result[x]
```

Les mots sont mémorisés dans un `HashMap<String, Integer>` qui associe à chaque mot le nombre d'occurrences trouvées. Enfin, le programme imprime le mot avec le plus grand nombre d'occurrences.

Tester la fonction pour compter les occurrences du mot avec des tests de régression (JUnit).

1.2 Compteur multi-threaded

Écrire le programme de manière multithreadé. Le programme prend en entré sur la ligne de commande le nombre de threads (variable `nthreads`) et le nom du fichier.

(Vous pouvez utiliser le même projet Java, en créant une deuxième classe avec une méthode `main()`. Comme ça, vous pouvez réutiliser une partie du code déjà écrit précédemment).

Il faut maintenant écrire une fonction qui lit le fichier ligne par ligne, et produit `nthreads` ArrayLists, une pour chaque thread. Chaque thread traites les ligne affecté à lui et produit le résultat dans une `HashMap`.

Question est-ce que on peut utiliser un seul `HashMap` pour tous les thread ?

Écrire de tests de régression pour les fonctions principales.

2 Serveur pour compter les mots

Écrire un serveur qui fourni le service de comptage.

Le serveur attends la connexion d'un client. Quand la connexion est établi, le client envoie une séquence de lignes de texte, terminé par la ligne :

:::END:::

Le serveur lance `N` threads qui font le comptage en parallèle, et il réponds avec un message qui contient le mot le plus fréquent. Le nombre `N` de threads est un paramètre à passer sur la ligne de commande du serveur.

Coder également un client qui prend en paramètre sur la ligne de commande le nom d'un fichier texte, lit le fichier et l'envoie au serveur ligne par ligne, en terminant le message avec la ligne :::`END`:::.

Vérifiez le bon fonctionnement du client et du serveur.

Notes :

1. Il est possible de coder le serveur en utilisant directement le threads, ou le `ExecutorService` vu en cours.
2. Le serveur doit être capable de servir plusieurs clients au même temps.

3 Rendu

Rendre un fichier .zip qui contient

— Un fichier `readme.txt` qui contient

1. vos noms
2. un résumé de ce que vous êtes réussi à faire
3. des instructions pour lancer le programme

— Un ou plusieurs fichiers `.jar` qui contiennent les quatre programmes (version séquentielle, version multi-threadé, serveur et client). Il doit être possible de lancer les programmes de la ligne de commande, par exemple :
`java -jar myprogram.jar arguments`

— Le code source.