

POCA — TP noté n°3

Java concurrence

Jean-Baptiste Yunès
Jean-Baptiste.Yunes@u-paris.fr

3 décembre 2024

Rendu final

L'intégralité du code produit, les tests, etc doit être fourni à la date indiquée par l'enseignant. Il est important de ne fournir que le code source, et de prendre bien soin que rien ne manque. Le code sera fourni sous la forme d'une archive `tar.gz` ou `zip` (pas autre chose). Le code devra comprendre un fichier `README.txt` contenant le nom et le prénom (dans cet ordre!) de l'élève concerné. Les modalités de rendu seront données par l'enseignant mais se feront sur `moodle.u-paris.fr` dans le cours POCA et sous aucune autre forme.

Généralités

Dans ce TP, on vise à utiliser diverses techniques de concurrence pour accélérer un calcul. Le calcul consiste à compter les occurrences d'un mot dans plusieurs textes. Sont donnés deux classes utiles :

FileScanner qui contient essentiellement une méthode permettant de compter l'occurrence d'un mot dans un fichier.

Utils qui contient une constante (le mot à chercher), une méthode permettant de récupérer une liste de fichiers à traiter et une méthode permettant de mesurer le temps d'exécution d'un `Runnable`.

On peut les utiliser ainsi pour obtenir le nombre d'occurrence d'un mot donné dans un fichier :

```
public class Test {  
    public static void main(String[] args) {
```

```
// Directory that contains the files to parse
var paths = Uutils.init("/home/truc");

// Example
var f = new FileScanner(paths.get(0).toFile());
System.out.println(f.count(Uutils.WORD,true));
}
}
```

Il est demandé de ne pas pas modifier une quelconque de ces classes. Le code source de ces classes ainsi que le fichier texte sont à récupérer sur moodle ou sur la page de l'enseignant. Chaque question devra être résolue dans une méthode de nom `test n` , où n est le numéro de la question. On devra donc écrire quelque chose comme :

```
public class Test {
    public static void main(String[] args) {
        // Directory that contains the files to parse
        var paths = Uutils.init("/home/blabla");

        test1(paths);
    }
    public static void test1(List<Path> paths) {
        Uutils.measure("message", () -> {
            var f = new FileScanner(paths.get(0).toFile());
            System.out.println(f.count(Uutils.WORD,true));
        });
    }
}
```

1 Question n°1

Calculer le temps nécessaire pour compter le nombre de fois qu'apparaît le mot donné sans concurrence, si possible en utilisant un `Stream`.

2 Question n°2

Résoudre le problème en utilisant un `Stream` parallèle.

3 Question n°3

Multithreader en utilisant un `ExecutorService` avec des `Callable<Long>` et en récupérant les valeurs des `Future<Long>`. Attention, on ne peut pas créer un `ExecutorService` utilisant trop de threads natifs (leur nombre est limité)...

4 Question n°4

Puisqu'on ne peut pas créer trop de `Thread` natifs, on souhaite résoudre la question précédente en utilisant des `Virtual Threads` qui eux ne sont pas limités en nombre. Les `Threads` utiliseront une `ArrayBlockingQueue<Long>` (de taille limitée et pas trop grande) pour communiquer leurs résultats au threads principal.

5 Question n°5

Réécrire le code précédent en utilisant un `ExecutorService` et une `ArrayBlockingQueue<Long>`.

6 Question n°6

Résoudre le problème à l'aide d'une `RecursiveTask<Long>`.

7 Question n°7

Résoudre le problème à l'aide de `CompletableFuture<Long>` de sorte à obtenir un seul `CompletableFuture` qui permettra d'obtenir le résultat final.

8 Question n°8

Produire un fichier texte de nom `time.txt` contenant les résultats des différentes mesures.