Langages de programmation 2 : Projet Java

Yves Roggeman *

17 novembre 2015

Résumé

Ceci est un des deux énoncés de l'épreuve de première session qui se déroule en janvier. Il sert de base à l'épreuve orale ; l'évaluation finale porte sur l'acquisition des concepts démontrée à cette occasion. Le but de cet exercice de programmation est donc de démontrer une connaissance approfondie et un usage adéquat des constructions du langage de programmation `Java`, en particulier de l'usage des primitives de parallélisme (liées aux *Treads*). L'évaluation portera donc essentiellement sur la pertinence des choix effectués dans l'écriture : la codification, la présentation et l'optimisation du programme justifiée par la maîtrise des mécanismes mis en œuvre lors de la compilation et l'exécution du code.

Le problème posé consiste à paralléliser un algorithme de tri récursif en faisant s'exécuter en parallèle les sous-appels récursifs qu'il contient, lorsque c'est possible. Il y a donc lieu de transcrire, d'adapter une version purement séquentielle en faisant bon usage des possibilités de parallélisme de Java et en identifiant les parties maximales du code qui ne sont pas mutuellement exclusives, afin d'optimiser le temps d'exécution espéré (si la JVM et son environnement permettent un parallélisme effectif et efficace).

1 L'algorithme à paralléliser

L'algorithme choisi pour illustrer la démarche est celui du tri d'un tableau par fusion. La fusion de deux tableaux triés consiste à construire pas à pas un tableau trié contenant l'union des deux tableaux donnés. C'est une opération linéaire (en $O(n)$), proportionnelle au nombre total d'éléments.

Le tri par fusion consiste, à chaque étape, à simplement fusionner deux demi-tableaux triés préalablement. Ces deux demi-tableaux auront été triés de la même façon, récursivement. La fusion étant linéaire et la découpe en deux conduisant à une profondeur $\log_2 n$, l'algorithme de tri par fusion a une performance (non parallèle) en temps de $O(n.\log n)$. Par contre, il requiert un espace de travail supplémentaire total de la taille du tableau à trier : $O(n)$; celui-ci sert à sauvegarder provisoirement (et localement) les valeurs situées dans la zone de destination de la fusion qui pourraient être écrasées prématurément.

Pour vous aider, voici un exemple en `C++` d'implantation séquentielle de cet algorithme.

Programme 1 – Algorithme de tri par fusion

```
1  template <typename T>
2  void MergeSort (T v[], const std::size_t n) {
3    if (n > 1) {
4      const std::size_t m = n/2;
5      MergeSort(v, m);
6      MergeSort(v+m, n-m);
7      T *w = new T[m];
8      std::size_t i = 0, j = 0, k = 0;
9      for (; i < m; ++i) w[i] = v[i];
10     while (j < m && i < n) v[k++] = v[i] < w[j] ? v[i++] : w[j++];
11     while (j < m) v[k++] = w[j++];
12     delete[] w;
13   }
14 }
```

Lors de la parallélisation de cet algorithme en `Java`, n'hésitez pas à permuter éventuellement certaines instructions qui peuvent l'être, afin d'optimiser l'effet potentiel de cette parallélisation.

*Université libre de Bruxelles (ULB) <yves.roggeman@ulb.ac.be>

2 Réalisation

Il vous est demandé d'écrire en `Java` un fichier (package anonyme) contenant une méthode `main` contenant plusieurs appels de test d'invocation de ce tri. Afin de vérifier le bon parallélisme, vous veillerez à produire quelques messages permettant de tracer l'exécution. Pour le reste, la forme exacte et les choix des méthodes et des classes nécessaires à la réalisation de l'implantation demandée sont laissés à votre choix.

D'une manière générale, la concision, la précision, la lisibilité (clarté du texte source), l'efficacité (pas d'opérations inutiles ou inadéquates) et le juste choix des syntaxes typiques de `Java` seront des critères essentiels d'appréciation. De brefs commentaires dans le code source sont souhaités pour éclairer les choix de codification.

Votre travail doit être réalisé pour le jeudi 17 décembre 2015 à 18 heures au plus tard. Vous remettrez tout votre travail empaqueté en un seul fichier compacté (« .zip » ou autre) via le site du cours (INFO-F-202) sur l'Université virtuelle (http ://uv.ulb.ac.be/). Ceux-ci devront contenir en commentaire vos matricule, nom, prénom et année d'études. Le jour de l'examen, vous viendrez avec une version imprimée — un *listing* — de ces divers fichiers. Une impression du résultat d'une exécution du programme est également demandée.