UNIVERSITÉ LIBRE DE BRUXELLES ULB

FACULTÉ DES SCIENCES
DÉPARTEMENT D'INFORMATIQUE

Langages de programmation 2 : Projet Java

Yves Roggeman *

Année académique 2016–2017

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 *threads*). L'évaluation portera donc essentiellement sur la pertinence des choix effectués dans l'écriture : la codification, la présentation et l'optimisation du programme justifiées par la maîtrise des mécanismes mis en œuvre lors de la compilation et l'exécution du code.

Le problème posé consiste à implanter un algorithme de tri parallèle particulier en attribuant à différents *threads* les opérations parallélisables. La performance finale effective dépendra non seulement de la JVM et de son environnement qui déterminera le degré de parallélisation réelle en fonction du nombre de processeurs ou de cœurs qui lui sont accessibles, mais également des choix effectués par le programmeur dans la traduction de l'algorithme en `Java`. Tel est donc le l'enjeu du problème posé : choisir une implantation qui sera la plus efficace dans un environnement idéal.

Il vous est demandé de vous limiter aux outils de parallélisation élémentaires disponibles dans le noyau de `Java`, *i.e.* dans `java.lang`, donc pas ceux de `java.util.concurrent`, par exemple.

1 L'algorithme de tri pair-impair

Cet algorithme vise à trier un tableau de n valeurs ; n est un paramètre. Pour l'implantation de test, nous supposerons qu'il s'agit simplement de nombres entiers (type **`int`**) à trier par ordre croissant.

On suppose que l'on peut lancer au maximum p processus en parallèle ; p est un autre paramètre de l'algorithme. Chaque processus sera responsable d'un bloc de $\frac{n}{p}$ données contiguës du tableau, identifiées, par exemple, par l'indice de la première. On suppose pour le raisonnement que $p|n$, mais en pratique, ce sera $\lceil \frac{n}{p} \rceil$ données pour $(n \bmod p)$ processeurs et $\lfloor \frac{n}{p} \rfloor$ données pour les autres, quantités réparties en alternance. En cours de travail, chaque processus devra toutefois être capable d'accéder temporairement au double de données : les siennes et celles d'un des blocs voisins immédiats.

On suppose les processus identifiés par un indice $0, 1 \ldots (p-1)$ dans l'ordre des blocs de données qu'ils traiteront, ce qui les répartit en deux groupes : les processeurs « pairs » et les « impairs » en fonction de la parité de leur indice.

À son initialisation, chaque processus commence par trier ses propres données par un algorithme classique. Dans notre implantation, on peut utilise un simple tri par insertion linéaire ou toute autre méthode de la bibliothèque, car ce choix n'aura aucune influence sur la suite. Toutefois, le travail des différents processus peut et doit s'exécuter complètement en parallèle.

Ensuite, l'algorithme fait fusionner les données des processus avec celles d'un de leurs voisins immédiats. En alternance,

- Étape paire : les processus d'indice pair (d'indice $2k$) fusionnent partiellement leurs données propres avec celles de leur voisin de droite ou suivant (d'indice $2k+1$) pour ne conserver que les premières valeurs (les plus petites), tandis que ce voisin de droite effectue la même fusion partielle de ses données avec celles du premier pour ne conserver que les dernières valeurs (les plus grandes).
- Étape impaire : les processus d'indice impair (d'indice $2k+1$) font de même avec leur voisin suivant (d'indice$2k+2$).

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

Chacune des étapes permet de faire travailler en parallèle tous les couples de processus et chaque processus agit toujours, en écriture, sur le même bloc de données. Au sein d'un couple de processus, le travail est également possible en parallèle si l'on veille à dupliquer localement certaines données le temps d'une fusion ; on peut se limiter à une zone de travail supplémentaire de taille égale à celle de ses propres données.

Selon la parité de p, le nombre de processus, le premier bloc de données et/ou le dernier peuvent rester inchangés à une des étapes (il n'y a pas de processus de fusion lancé pour eux).

On démarre par une étape paire. Après p étapes, le tableau complet est trié. Il s'agit en effet d'un tri analogue à l'algorithme « cocktail », mais complètement parallélisé. Sa complexité temporelle est linéaire : $O(p.\frac{n}{p}) = O(n)$. En espace, il demande un espace de travail total de même valeur : $O(n)$.

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 lundi 14 novembre 2016 à 10 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.