Université de Sherbrooke

Algorithmes de dames parallélisés

Projet de cours

Par :
Jonathan Beaudette

Remis à :
Gabriel Girard

IFT630
Processus concurrents et parallélisme

27/04/2012

Description du projet

Dans le cadre du cours IFT 630, Processus concurrents et parallélisme, nous devions faire un projet de session théorique ou pratique. J'ai choisi de faire un projet pratique. Mon projet consiste à paralléliser des algorithmes séquentiels d'intelligence artificielle jouant au jeu de dames. Pour mettre en pratique ces algorithmes, j'ai trouvé un jeu de dames à code source libre nous permettant de sélectionner les joueurs artificiels du jeu parmi une dizaine d'algorithmes séquentiels. Mon but était de choisir certains de ses algorithmes, de les paralléliser et ensuite de les tester contre leur confrère séquentiel.

Contenu du projet

Le dossier remis comprend : premièrement, ce rapport; deuxièmement, un fichier jar pour l'exécution du projet; finalement, le code source du projet. Vous trouverez le code source des algorithmes modifiés dans le répertoire « EBFCheckers_Source_v1.26\source\checkersPlayer ». Les trois algorithmes modifiés sont : « BadCheckersPlayer_JB.java », « EFCheckersPlayer_JB.java » et « HAL900_JB.java ».

Déroulement du projet

Avant de commencer ce projet, j'ai passé plusieurs heures à chercher un jeu de dames à code source libre, flexible et fonctionnel. Une fois trouvé, j'ai passé du temps à comprendre la logique de l'application. Par la suite, j'ai analysé les algorithmes disponibles avant de choisir lesquels j'allais modifier. Une fois parallélisé, j'ai testé les algorithmes contre eux-mêmes et je me suis même amusé à jouer contre eux. Pour le développement de ce projet, j'ai créé un projet Netbeans à partir du code source existant.

Travail accompli

Parmi les algorithmes disponibles, il y en avait qui ne valait simplement pas la peine d'être parallélisé. Par exemple : « Basic AI », cet algorithme compare seulement les coups du premier niveau de l'arbre. « Random AI », celui-ci choisi le prochain coup de façon aléatoire. Les noms des algorithmes que j'ai modifiés contiennent le suffixe « _JB ». J'ai donc parallélisé les algorithmes suivants :

- **The Bad Checkers Player :**

Cet algorithme qui, évidemment, n'est pas très bon utilise la méthode de recherche « iterative deepening » avec l'algorithme « minimax ». Ce joueur utilise tout le temps qu'il lui est alloué pour parcourir l'arbre le plus profondément possible. Pour paralléliser l'algorithme, je crée un thread pour chaque premier coup possible. Donc, la recherche dans chaque sou arbres à partir du deuxième niveau se fais en parallèle. Chaque thread retourne un objet de type « Future » qui permet au maître d'attendre que le résultat des threads soit disponible. Pour comparer le joueur original et ma version parallèle, je calcule le temps moyen pour parcourir l'arbre à une profondeur de huit niveaux. Avec ma machine à quatre cœurs, l'algorithme parallèle est deux à cinq fois plus rapide que le séquentiel.

- **EFCheckers :**

Un algorithme très rapide qui parcourt toujours l'arbre à sept niveaux de profondeur. J'ai premièrement modifié cet algorithme pour qu'il consiste d'une fonction récursive au lieu d'être sept boucles imbriqués, de cette façon, je pouvais facilement modifier le niveau de profondeur parcouru. Encore une fois, le parcours de chaque sou arbres à partir du deuxième niveau est effectué dans un thread différent. Par contre, cette fois, les threads ne retournent aucune valeur. À la place, ils affectent le résultat de leurs calculs à une variable globale, si et seulement si leur valeur est plus grande que la valeur maximum. Pour assurer l'exclusion mutuelle, cette affectation est faite à partir d'une méthode synchronisée. Le processus maître utilise un join pour attendre que les processus travailleurs aient terminé. Le parcours de l'arbre au niveau sept se fait entre deux et trois fois plus rapidement avec la version parallèle que la version séquentiel.

- **HAL900 :**

Cet algorithme modifie dynamiquement le niveau de profondeur qu'il parcourt par rapport au nombre de nœuds qu'il a parcourus dans les tours précédents. En parallélisant encore une fois la recherche de chaque sou arbres, l'algorithme réussi souvent à parcourir un niveau de profondeur de plus que l'algorithme séquentiel.

Problèmes rencontrés

- **Intelligence artificielle :**

Une des difficultés de ce projet, pour moi, a été mon manque de connaissance en intelligence artificielle. Comme je n'ai pas encore suivi le cours d'intelligence artificielle, je n'étais pas familier avec

plusieurs des termes et des techniques utilisés par les joueurs artificiels de l'application. J'ai dû apprendre des concepts de base en intelligence artificielle pour réussir à paralléliser des algorithmes.

- **Temps maximum d'un tour :**

Cette application permet à l'utilisateur d'ajuster le temps de calcul alloué aux algorithmes à chaque tour. La difficulté pour quelques algorithmes était d'obtenir les résultats des différents threads travailleurs avant que le temps alloué ne soit fini.

Exécution et utilisation de l'application

- **Exécuter** :

  Pour faire rouler l'application, simplement exécutez la commande suivante : « java -jar "CheckersProject.jar" » en console.

- **Sélectionner les joueurs :**

  À partir de l'application, vous pouvez sélectionner les joueurs en appuyant sur la touche « s » ou en cliquant sur « Settings », « Player Setup » dans le menu. Vous pourrez choisir les joueurs à partir de la fenêtre affichée. Les noms de joueur terminant par mes initiales, JB, sont les algorithmes que j'ai parallélisés.

- **Configurer le temps :**

  Il est possible de configurer le temps maximum qu'un joueur peut prendre pour calculer son coup. Par contre, si le joueur n'effectue pas son coup dans le temps alloué, alors l'application choisira un déplacement aléatoire. Pour configurer le temps de maximum par tour, appuyer sur la touche « o » ou choisissez « Settings » et « Game Options » à partir du menu. La fenêtre des options s'affichera et vous pourrez modifier le paramètre « Turn Time ».

- **Démarrer une partie :**

  La dernière étape est de démarrer la partie. Vous n'avez qu'à appuyer sur la touche « n » ou choisir « File », « New Game » dans le menu. Il ne vous reste qu'à jouer ou à observer la partie si les deux joueurs sont artificiels.