Simulation d'exécution de programmes parallèles

Rapport intermédiaire de travail de recherche
de Master en Informatique de l'Université de Strasbourg

Olga Pigareva
M1 SIL

2 mars 2019

Table des matières

0.1 La présentation de la problématique dans son contexte

0.1.1 Introduction

Le sujet de mon travail d'étude et de recherche aborde une domaine très actuel dans le monde numérique, la programmation parallèle. La parallélisation de tâches et de processus permet d'augmenter la performance et faire l'exécution d'un programme beaucoup plus rapide. C'est pourquoi la programmation parallèle reste un sujet très actuel, même si elle était inventée il y a déjà plusieurs années.

La programmation parallèle comprend la résolution d'un problème simultanément par plusieurs processus. Cela implique l'utilisation d'un seul ressource par plusieurs processus, ce qui nécessite une synchronisation. De sa part, la synchronisation peut amener à un interblocage qui peut avoir les conséquences indésirables (catastrophiques) pour un programme.
Mon travail fait partie d'une recherche visant à caractériser ces interblocages. Dans les parties suivantes, j'expliquerai ce que signifient ces termes.

0.1.2 Programmes parallèles

Les programmes parallèles permettent de créer les activités qui peuvent exécuter les instructions indépendamment. Souvent ces activités ont besoin d'acceder aux mêmes ressources. Pour ne pas avoir les résultats imprévus lors de modification de données par les processus différents, les ressources sont synchronisées. Cela veut dire que les processus peuvent acceder les données chacun à son tour. Pour attendre la libération de données ou la fin d'instructions d'un processus, un programme parallèle utilise les horloges qui sont les barrières de synchronisation.

0.1.3 Interblocage

Le problème d'interblocage apparaît quand chaque processus attends une autre à se terminer et finalement aucun peut acceder à la ressource. Par conséquence, le programme reste bloquer.
Par exemple, sur la Figure 1, le processus P1 attends la fin de processus P2 pour acceder à R1 et P2, de sa part, attends la fin de P1 pour acceder à R2. Tous les deux processus restent bloquer.

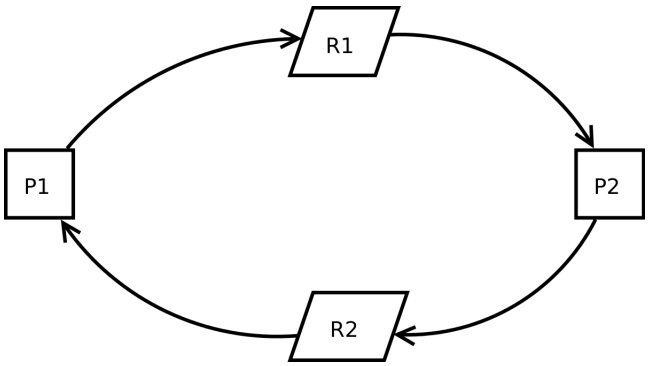

FIGURE 1 – Exemple d'interblocage

Souvent pour les programmes avec plusieurs processus et plusieurs ressources, il est difficile de controller l'absence d'interblocages, surtout avec la liberté et puissance offertes par la synchronisation. D'où l'intérêt de reconnaître ces interblocages automatiquement le plus tôt possible, à savoir, éventuellement, lors de la compilation du programme.

0.1.4 Problème

Est-il possible de définir un interblocage pendant l'étape de compilation ?

0.1.5 Objectifs

Pour répondre à cette question, l'idée est de commencer par créer une version simplifiée d'un langage parallèle en prenant un langage X10 comme un exemple. Le compilateur doit reconnaître juste les instructions qu'on a besoin pour créer plusieurs activités parallèles (boucles, conditions, création de nouveaux processus et horloges).

Après avoir implémenté un langage qui permet de créer plusieurs activités parallèles avec la synchronisation, il doit être possible de visualiser l'état de toutes les activités au moment d'interblocages. on se donne un choix entre simulation d'exécution de programmes parallèles et afficher un message si l'execution se bloquer

0.2 La liste des tâches déjà effectuées

0.2.1 Le travail réalisé

— Installation de librairies et IDE pour la langage de programmation **X10** sur mon ordinateur.
Premièrement, il fallait de préparer l'environnement de travail.
— Lecture de documentation et plus de documents sur l'interblocage et programmes parallèles.
— Exécution de différentes programmes test X10 pour comprendre le problème et voir comment il fonctionne.
— Implementation d'un compilateur inspiré par le langage X10 à l'aide de Lexer et parser (Lex et Yacc).
— Génération d'un arbre de la syntaxe abstraite (AST en anglais).

0.3 Le planning prévisionnel des tâches à réaliser

mars Implementation de simulation d'exécution du programme et captures de traces d'exécution.

avril Implementation de visualisation des traces d'execution ou de systèmes mathématique permettant détecter un interblocage.

mai Rédiger un rapport avec le bilan de résultats atteins.

0.4 Il peut aussi contenir tout ce qui apparaîtra dans votre rapport final.

0.5 Motivation