GGen : Génération Aléatoire de Graphes Pour l'Ordonnancement

Swann Perarnau[1], Denis Trystram[1,2], Jean-Marc Vincent[1]

[1] Université de Grenoble
[2] Institut Universitaire de France
{swann.perarnau,denis.trystram,jean-marc.vincent}@imag.fr

Mots-clés : *ordonnancement, simulation, graphes.*

1 Introduction

Une grande partie des publications traitant d'ordonnancement contient une section simulation. Cette simulation permet de valider une implémentation, d'étudier un algorithme dans des conditions différentes du cadre d'une preuve théorique ou de comparer la performance de différents ordonnanceurs sur un même problème. Cette simulation repose sur deux types de données en entrée : des jeux de données *stencil* [4, 6], bien spécifiques et connus de la communauté, ainsi que des données générées aléatoirement.

Nous discutons dans cette présentation de GGen, un projet né en 2009 dans le but d'étudier l'influence des méthodes de génération aléatoire de données sur la qualité des simulations d'ordonnanceurs. Ce projet se concentre sur la génération de graphes orientés acycliques (DAGs), le modèle de tâches le plus utilisé en ordonnancement pour le parallélisme.

Au cours de ces deux années, GGen s'est doté d'une implémentation de référence des différentes méthodes de génération de DAGs les plus utilisées, d'une suite d'outils pour leur analyse et nous avons réalisé avec l'appui de Grid5000 de vastes campagnes de caractérisation des graphes générés pour étudier leur influence sur la simulation d'ordonnanceurs.

2 Pourquoi GGen ?

Pour comprendre la nature et la difficulté du problème, il est important de se rappeler qu'il n'existe aucun algorithme de génération aléatoire de DAG pouvant convenir à tous.

Si l'objectif de la génération est de fournir des structures de données variées, alors idéalement un générateur de graphes non isomorphes serait nécessaire. Malheureusement, un tel générateur est impossible en pratique : même l'énumération des graphes non isomorphes est un problème trop couteux en temps et en mémoire pour être réalisable. L'encyclopédie en ligne des séquences d'entiers donne une idée de la difficulté du problème : le nombre de DAGs non isomorphes différents n'est connu que pour des graphes de taille inférieure à 16 [5].

Si l'objectif de la génération est de fournir des données *réalistes*, alors n'importe quelle méthode choisie possédera des caractéristiques pouvant favoriser un ordonnanceur par rapport à un autre. C'est pourquoi il est nécessaire de disposer d'un cadre commun permettant d'analyser en détail ces méthodes et leur influence sur les performances des ordonnanceurs.

3 La boîte à outils GGen

Depuis sa création, GGen s'est petit à petit doté de différents outils pour la génération et l'analyse de graphes.

La première étape du projet a permis le développement d'une implémentation de référence de chacune des méthodes de génération couramment utilisée par la communauté [2, 3]. Cette implémentation est d'autant plus nécessaire que la plupart des algorithmes proposés dans la littérature ne sont pas suffisamment détaillés pour être implémentés en pratique. Cette implémentation s'appuie sur des bibliothèques reconnues pour leur qualité (Igraph pour la manipulation de graphes et GNU Scientific Library pour les générateurs aléatoires).

Une deuxième étape a apporté au projet un ensemble d'outils d'analyse et de modification des graphes générés. Ces algorithmes se concentrent sur les manipulations les plus courantes (ajout de poids sur les nœuds et les arêtes, transformations diverses) et l'analyse des caractéristiques les plus étudiées par la communauté, comme le chemin le plus long ou le diamètre.

Au cours de l'année 2010, un certain nombre d'analyses ont été effectuées dans le cadre du projet sur la plateforme expérimentale Grid5000. Plus de 1 million de graphes ont été caractérisés afin d'obtenir une idée précise du type de DAGs générés par les méthodes implémentées dans GGen. Ces résultats d'analyse sont en partie disponible dans une publication à Simutools 2010 [1].

4 Travaux en Cours

Le travail autour de GGen se concentre désormais sur les deux points suivants :

- la simulation d'algorithmes d'ordonnancement sur des jeux de données provenant du projet. Une étude approfondie de la performance d'ordonnanceurs pour systèmes distribués est en cours, avec notamment la recherche *d'inversions* : la possibilité qu'un changement de méthode de génération de données modifie une comparaison d'ordonnanceurs au point d'observer un renversement de leur performance.
- le développement d'un environnement automatique de caractérisation des graphes. Un tel outil permettrait facilement, à partir d'une méthode de génération et d'un algorithme d'ordonnancement, de rechercher des *points chauds* : des jeux de paramètres augmentant de manière significative la difficulté (pour l'ordonnanceur considéré) des graphes générés.

Références

[1] Daniel Cordeiro, Grégory Mounié, Swann Perarnau, Denis Trystram, Jean-Marc Vincent, and Frédéric Wagner. Random graph generation for scheduling simulations. In *Proceedings of 3rd International SIMUTools Conference*, 2010.

[2] Robert P. Dick, David L. Rhodes, and Wayne Wolf. TGFF : Task Graphs For Free. In *Proceedings of the 6th International Workshop on Hardware/Software Codesign*, pages 97–101, Washington, DC, USA, March 1998. IEEE Computer Society.

[3] Paul Erdős and Alfréd Rényi. On random graphs I. *Publicationes Mathematicae Debrecen*, 6 :290–297, 1959.

[4] Yu-Kwong Kwok and Ishfaq Ahmad. Benchmarking the task graph scheduling algorithms. In *Parallel Processing Symposium. IPPS/SPDP*, pages 531–537, April 1998.

[5] Neil James Alexander Sloane and Simon Plouffe. *The Encyclopedia of Integer Sequences*. Academic Press, 1995.

[6] Takao Tobita and Hironori Kasahara. A standard task graph set for fair evaluation of multiprocessor scheduling algorithms. *Journal of Scheduling*, 5(5) :379–394, 2002.