Vérification automatisée de structures de données réparties pour des processeurs multicœur

Présentation du sujet

Les spécificités des processeurs multicœur imposent de développer des structures de données réparties, capables d'assurer un certain nombre de propriétés, essentielles à leur bon fonctionnement : absence d'inter-blocages, équité, cohérence des données, etc.

Habituellement, les concepteurs de telles structures de données prouvent, à la main, leur bon fonctionnement. En guise d'exemple d'une telle preuve on pourra lire un article présentant une implantation de *skip list* par Herlihy, Lev, Luchangco et Shavit [1]. Cependant, en général, en raison d'un très grand nombre de configurations possibles, appréhender complètement le fonctionnement d'un système réparti est une tâche très complexe pour un humain. La confiance qu'on peut avoir en de telles preuves est donc limitée, d'autant plus pour des structures de données plus complexes qu'une simple liste (files de priorité, arbres, etc [2]).

Si avoir une totale confiance dans une implantation particulière d'une structure de données pour les processeurs multicœur est aujourd'hui très difficile, il existe tout de même des solutions pour acquérir une confiance raisonnable en celle-ci. Il est notamment possible d'utiliser des outils de vérification automatique, qui travaillent généralement par exploration exhaustive de toutes les configurations (d'un modèle, généralement à évènements discrets : automate, réseau d'automates, réseau de Petri) d'un système – c'est-à-dire, ici, une implantation d'une structure de données, associée à un processeur. Il existe un certains nombre de tels outils, par exemple, SPIN [3], CADP [4], LTSMin [5], LoLA [6], ITSTools [7]... chacun ayant ses spécificités lui permettant de vérifier plus ou moins efficacement certains types de systèmes et de propriétés.

L'objectif de ce stage sera de rechercher, à l'aide d'outils de vérification automatique, des bugs dans des implantations de structures de données pour les processeurs multicœur. Dans un premier temps l'étudiant pourra introduire volontairement des bugs dans des structures de données simples, afin de se familiariser avec les techniques de modélisation et d'appréhender les spécificités des différents outils. Il cherchera ensuite à prouver, à l'aide d'outils de vérification automatique, l'existence de bugs dans des structures de données plus complexes, que des spécialistes des systèmes répartis estiment être potentiellement incorrectes, sans avoir pu le démontrer.

Éléments de bibliographie

1. Herlihy, Lev, Luchangco et Shavit. *A Provably Correct Scalable Concurrent Skip List.* OPODIS'06.
2. Gramoli. *More than you ever wanted to know about synchronization: synchrobench, measuring the impact of the synchronization on concurrent algorithms.* PPOPP'15.
3. `http://spinroot.com/spin/whatispin.html`
4. `http://cadp.inria.fr/`
5. `http://fmt.cs.utwente.nl/tools/ltsmin/`
6. `http://home.gna.org/service-tech/lola/`
7. `http://ddd.lip6.fr/itstools.php`

Encadrement

Le stage, encadré par Loïg Jezequel (Université de Nantes) et Corentin Travers (ENSEIRB-MATMECA), aura lieu à Nantes, dans l'équipe STR (`https://ls2n.fr/equipe/str/`) du LS2N (`http://ls2n.fr/`).