Proposition de recherche doctorale

Parallélisation d'un SAT-solveur

Mots clés :

- **Directeur de thèse** : Fabrice Kordon
- **Co-encadrant(s)** :
- **Unité de recherche** : Laboratoire d'informatique de Paris 6
- **Ecole doctorale** : École Doctorale Informatique, Télécommunications, Électronique de Paris
- **Domaine scientifique principal**: Divers

Résumé du projet de recherche (Langue 1)

{{{Contexte : problèmes SAT et multi-cœur}}} Avec l'intégration toujours plus grande, des systèmes informatiques dans des applications critiques (métro, voitures, centrales-nucléaires, blocs opératoires, etc), assurer la correction des programmes devient une nécessité. Si l'utilisation systématique de tests permet d'améliorer grandement la qualité des codes, elle ne peut réellement garantir leur correction A l'inverse, les techniques dites de model-checking, qui vérifient la validité du workflow applicatif (modélisé par un graphe d'états) par rapport à une propriété donnée, permettent de garantir la vali- dité des programmes. Ces techniques reposent, essentiellement, sur deux théories : celle basée sur les automates et celle fondée sur la satisfiabilité booléenne (SAT). Dans les deux théories l'ennemi com- mun est le problème bien connu de l'explosion combinatoire. En utilisant les automates, le problème se manifeste par une consommation mémoire démesuré, alors qu'en utilisant le SAT, le problème se manifeste sous la forme d'un temps de calcul qui peut être extrêmement long. Or, ces dernières années le nombre d'unités de calcul présentent dans les architectures multi-cœurs ne cesse d'augmenter. La puissance de calcul ainsi offerte ouvre de nouvelles perspectives pour la résolution de problèmes SAT. {{{L'existant en SAT-solving parallèle}}} Si l'utilisation d'architectures multi-cœurs a très vite semblé prometteuse c'est que la résolution de problème SAT se prête très bien à la parallélisation. En effet, en fixant la valeur de l'une des variables boolénnes propositionnelles u on obtient deux nouveaux problèmes disjoints : l'un associé à la valeur u = vrai, l'autre découlant de u = f aux. Ces deux nouveaux problèmes plus petits (puisque contenant, au moins, une variable de moins) peuvent alors se résoudre indépendamment. Partant de ce constat, de nombreux travaux de recherche ont été lancés pour tirer parti des nou- veaux multi-cœurs. Les deux concours internationaux de solveurs SAT comptent aujourd'hui 40% de solutions parallèles [1]. Parmi ces propositions, on distingue deux types d'approches : -* parallèlisation non collaborative [9] : le problème est découpé en un grand nombre de sous- problèmes plus petits correspondant à autant de combinaisons d'un sous-ensemble de variables. Chacun de ces problèmes est alors soumis indépendamment à une des instances du solveur. Cette approche présente l'intérêt de limiter au maximum les échanges et la synchronisation puisque chaque instance manipule ses propres structures de donnés. -* parallélisation collaborative [6, 2, 11] : à l'inverse de la parallélisation non collaborative, les différentes instances du solveur sont exécutés par des threads qui partagent les mêmes struc- tures de donnés. Ces derniers peuvent alors bénéficier de l'appantissages des autres threads. Plusieurs optimisations, dites de clause learning [10, 8], ont en effet été développés pour éviter d'explorer deux fois les mêmes sous-problèmes et optimiser la propagation des décisions. La parallélisation des solveurs SAT dans les multi-cœurs semblait donc être très prometteuse et les premiers résultats allaient dans ce sens. Cependant, les derniers résultats des concours (SAT competi- tion 2013) ont marqué un début de renversement de situation avec la victoire de solveur SAT appelé portfolio [12, 3]. Dans ces approches, plusieurs instances du problème sont soumises à différents al- gorithmes séquentiel de résolution de SAT. Le premier à donner la réponse stop l'analyse. Les bons résultats de ces solveurs, montrent qu'aujourd'hui les meilleurs algorithmes séquentiels sont plus effi- caces que les solutions parallèles. Les approches parallèles souffrent en effet d'un problème de partage de l'information. D'un coté, l'isolation des instances dans la parallélisation non collaborative empêche le partage des connais- sances, les empêchant ainsi de bénéficier des dernières avancés sur les heuristiques de décisions et sur l'apprentissage dynamique des différentes instances. À l'opposé, les problèmes liés à la synchronisa- tion et aux choix de l'information à partager constituent les goulots d'étranglement des approches collaboratives. En effet, une synchronisation naïve (par exemple, faisant intervenir des algorithmes bloquants) va faire exploser les temps de communications et d'attentes. Aussi, un partage non contrôlé de l'information entre les instances peut avoir un effet très néfaste sur le temps de calcul. Par exemple, une instance A qui communique une information à une instance B, qui doit la stocker et la traiter, alors qu'elle s'avère absolument inutile pour son future.

Résumé du projet de recherche (Langue 2)

L'objectif de cette thèse est d'imaginer puis d'intégrer à notre prototype des mécanismes permet- tant de limiter le coût lié à la synchronisation. En effet, les bonnes performances obtenues avec les 8 cœurs de la compétition, ne se maintiennent pas, dès lors que l'on utilise la capacité totale de la machine de test (40 cœurs). L'influence de la bande passante mémoire et des caches est significative sur les machines multicoeurs. Aux deux techniques de la proposition initiale, (1) utilisation de structure wait-free et (2) retar- dement du partage des clauses, nous envisageons aujourd'hui d'ajouter un mécanisme dynamique de selection des clauses partagés. En effet, une étude fine des espaces de recherche des threads, nous a mis en évidence une corrélation entre ces espaces et l'utilisation des clauses apprises. L'idée de cette troisième approche reprend le principe d'activation/désactivation des clauses développé dans [4], pour sélectionner les clauses à partager et ainsi diminuer le poids de la synchronisation.

Informations complémentaires (Langue 2)

[2] Y. Akashi and T. Yasumoto. Zennfork. SAT Challenge 2012, pages 17–19, 2012. [3] G. Audemard, B. Hoessen, S. Jabbour, J.-M. Lagniez, and C. Piette. Penelope, a parallel clause-freezer solver. SAT Challenge 2012, pages 43–44, 2012. [4] G. Audemard, J.-M. Lagniez, B. Mazure, and L. Saïs. On freezing and reactivating learnt clauses. In Pro- ceedings of the 14th International Conference on Theory and Application of Satisfiability Testing, SAT'11, pages 188–200, Berlin, Heidelberg, 2011. Springer-Verlag. [5] A. T. Clements, M. F. Kaashoek, and N. Zeldovich. Scalable address spaces using rcu balanced trees. In Proceedings of the seventeenth international conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS '12, pages 199–210, New York, NY, USA, 2012. ACM. [6] B.Kaufmann and M.Schneider.clasp, claspfolio, aspeed :Three solvers from the answer set solving collection potassco. SAT Challenge 2012, pages 17–19, 2012. [7] H. T. Kung and P. L. Lehman. Concurrent manipulation of binary search trees. ACM Trans. Database Syst., 5(3) :354–382, Sept. 1980. [8] K. Pipatsrisawat and A. Darwiche. On the power of clause-learning sat solvers with restarts. Principles and Practice of Constraint Programming-CP 2009, pages 654–668, 2009. [9] O. Roussel. Description of ppfolio 2012. SAT Challenge 2012, page 46, 2012. [10] L. Ryan. Efficient algorithms for clause-learning SAT solvers. PhD thesis, Theses (School of Computing Science)/Simon Fraser University, 2004. [11] T. Sonobe. Parallel cir minisat. SAT Challenge 2012, pages 38–39, 2012. [12] A. Wotzlaw, A. van der Grinten, E. Speckenmeyer, and S. Porschen. pfoliouzk : Solver description. SAT Challenge 2012, page 45, 2012.