Implémentation du Hachage parfait de classes dans une Machine virtuelle Java

F. Morandat

Java est un langage orienté objet en héritage simple, c'est a dire qu'une classe a au maximum une seule super classe directe. L'héritage simple se compile directement et efficacement. Les interfaces introduisent une notion de sous-typage multiple. Cependant l'invocation de méthodes typées par une interface soulève les mêmes problèmes que l'héritage multiple et, jusqu'à récemment, il n'existait pas de solution élégante et efficace pour implémenter ce mécanisme. Bien que les interfaces furent longtemps négligées par les programmeurs car elles étaient considérées trop lentes. Elles sont maintenant de plus en plus préconisées par de nombreux Frameworks (e.g. Eclipse) et par conséquent leurs nombre augmente, aussi bien dans le JDK que dans les applications utilisateurs. Cette augmentation crée un nouveau goulot d'étranglement (surtout sur les VM les plus modestes, qui ne virtualisent et dévirtualisent pas les appels). Ceci exacerbe encore les écarts entre les VM commerciales et les produits académiques.

Nous avons proposé récemment une technique appelée *hachage parfait de classes* [Ducournau, 2008, Ducournau and Morandat, 2011] et cherchons à vérifier sa viabilité dans une machine virtuelle complète. Cette technique est en temps constant et reste compatible avec les invariants utilisés par les machines virtuelles (VM).

L'objectif de ce stage est double. Tout d'abord implémenter dans une machine virtuelle Java (J3 [Geoffray et al., 2010] le hachage parfait de classes pour l'invocation de méthodes typées par une interface, ainsi que pour le test de sous-typage. Enfin évaluer dans les performances et le coût mémoire sur des programmes réalistes (DaCapo [Blackburn et al., 2006], SpecJVM, ...)

L'étudiant devra produire du code propre et montrer des capacités à travailler sur un gros projet. De bonnes connaissances en Java et C (manipulation de pointeurs) seront aussi nécessaires.

References

Stephen M. Blackburn, Robin Garner, Chris Hoffmann, Asjad M. Khan, Kathryn S. McKinley, Rotem Bentzur, Amer Diwan, Daniel Feinberg, Daniel Frampton, Samuel Z. Guyer, Martin Hirzel, Antony L. Hosking, Maria Jump, Han Bok Lee, J. Eliot B. Moss, Aashish Phansalkar, Darko Stefanovic, Thomas VanDrunen, Daniel von Dincklage, and Ben Wiedermann. The dacapo benchmarks: java benchmarking development and analysis. In Peri L. Tarr and William R. Cook, editors, *OOPSLA*, pages 169–190. ACM, 2006. ISBN 1-59593-348-4.

R. Ducournau. Perfect hashing as an almost perfect subtype test. *ACM Trans. Program. Lang. Syst.*, 30(6):1–56, 2008.

Roland Ducournau and Floréal Morandat. Perfect class hashing and numbering for object-oriented implementation. *Softw., Pract. Exper.*, 41(6):661–694, 2011.

Nicolas Geoffray, Gaël Thomas, Julia L. Lawall, Gilles Muller, and Bertil Folliot. Vmkit: a substrate for managed runtime environments. In Marc E. Fiuczynski, Emery D. Berger, and Andrew Warfield, editors, *VEE*, pages 51–62. ACM, 2010. ISBN 978-1-60558-910-7.