Programmation OpenCL sur architectures hybrides

Raymond Namyst
équipe-projet LaBRI-Inria « Runtime »

Mots clés : Calcul parallèle, Accélérateur, GPU, multicoeur, OpenCL.

Contexte et objectifs du travail

Les ordinateurs multicoeurs équipés d'accélérateurs réalisent une percée remarquable dans le paysage du calcul haute performance. Parmi les machines parallèles les plus puissantes au monde (selon le classement www.top500.org), une sur deux est équipée d'accélérateurs. Cette récente évolution vers des architectures hétérogènes pose des problèmes de portabilité des applications, qui doivent souvent être composées de codes développés au moyen de langages différents (e.g. OpenMP, CUDA, Intel TBB, OpenCL, etc.)

Pour tenter d'uniformiser la façon de programmer des accélérateurs de calcul, le consortium Khronos a donné naissance au standard de programmation OpenCL en 2008. OpenCL fournit en effet un langage indépendant du matériel permettant d'exprimer du parallélisme massif de type SIMD, et de nombreux compilateurs permettent de générer du code pour la quasi-totalité des accélérateurs existants.

Récemment, des travaux de recheche ont proposé des techniques permettant soit de généraliser un programme OpenCL mono-accélérateur à des machines multi-accélérateurs, soit de répartir dynamiquement la charge de calcul au sein de programmes OpenCL multi-accélérateurs. Toutefois, ces approches se heurtent encore à un problème épineux : les programmes OpenCL sont presque tous optimisés pour un type d'accélérateur donné. Autrement dit, un programme OpenCL destiné à s'exécuter sur un GPU n'est pas du tout codé de la même façon (layout des données, utilisation de la mémoire locale, etc.) qu'un programme OpenCL destiné à s'exécuter sur un coprocesseur Intel Xeon Phi. La faute à un langage encore trop proche du matériel.

Le travail consistera à étudier les articles mentionnés ci-dessous, et à en estimer les limites, en terme de portabilité des performances, sur les architectures hybrides contemporaines. Il s'agira ensuite de conduire des petites expérimentations simples mettant en lumière les différences de codage OpenCL entre un GPU et un Xeon Phi sur un noyau de calcul d'interaction NBODY. On essaiera d'imaginer une version générique de code OpenCL qui pourrait être "automatiquement" dérivée en code optimisé par un spécialiseur de code. On s'attachera en particulier à voir comment gérer de manière transparente un agencement (layout) différent en fonction de l'accélérateur sous-jacent.

Bibliographie

1. SnuCL : An OpenCL Framework for Heterogeneous CPU/GPU Clusters, Jungwon Kim, Sangmin Seo, Jun Lee, Jeongho Nah, Gangwon Jo, and Jaejin Lee, Proceedings of the 26th ACM international conference on Supercomputing, Pages 341-352, 2012.
2. Transparent CPU-GPU Collaboration for Data-Parallel Kernels on Heterogeneous Systems, Janghaeng Lee, Mehrzad Samadi, Yongjun Park, Scott Mahlke, Proceedings of the 22nd international conference on Parallel architectures and compilation techniques, Pages 245-256, 2013.

Liens utiles

— Préconisation d'Intel concernant la programmation OpenCL sur Xeon Phi : OpenCL Design and Programming Guide for the Intel Xeon Phi Coprocessor