Génération automatique de tâches parallèles sur architecture hybride: application à la Chromo-dynamique quantique

Sujet de Mémoire de Master 2

1er décembre 2011

Mots-clés : compilation, génération de code parallèle, calcul haute-performance, parallélisme
Lieu de thèse : Équipe Satanas, LaBRI, Bordeaux.
Encadrant Denis BARTHOU
Contact : 05 24 57 41 16 ; denis.barthou@labri.fr

1 Contexte scientifique

Les architectures parallèles les plus performantes actuellement utilisent plusieurs formes de parallélisme et sont hybrides, mélangeant des multicoeurs et des cartes graphiques. L'adaptation des codes applicatifs à ces nouvelles architectures se heurte à plusieurs difficultés. D'une part, l'évolution rapide des architectures et la grande diversité des modèles de programmation parallèle (OpenMP, OpenHMPP, les PGAS, OpenCL, ...) ne permet pas de garantir que l'effort d'adaptation d'un code à une machine sera valable pour la génération suivante. D'autre part cette adaptation nécessite des modifications importantes de structures de contrôle et de données du programme, ainsi qu'une adaptation du parallélisme. Enfin, pour des applications ayant de grands besoins en puissance de calcul, l'obtention de performances exaflopiques passe par la mise au point de nouveaux algorithmes et ses adaptations à l'architecture sont autant de frein à la mise au point de nouveaux codes.

L'objectif du projet ANR PetaQCD [1], fruit d'une collaboration entre physiciens et informaticiens à laquelle participent des enseignants/chercheurs du LaBRI, est de développer des outils logiciels de génération de code dédiés aux simulations de la chromo-dynamique quantique (QCD). Ces simulations sont un outil essentiel pour améliorer la connaissance de la matière à l'échelle subnucléaire. Le coeur de ces simulations consiste en une résolution de système linéaire creux, que l'on peut retrouver dans d'autres applications de calcul hautes performances. Nous avons proposé dans le projet de partir d'une représentation plus riche sémantiquement du problème, en partant des formules et des algorithmes, dans un langage symbolique mathématique. Ce langage permet de ne pas dépendre d'un langage parallèle particulier, de prouver des propriétés sur les algorithmes utilisés et de tester rapidement différentes idées d'algorithmes. Le compilateur QIRAL développé pour ce langage, le compile en un code exécutable. Le parallélisme est pour l'instant exprimé uniquement au niveau des boucles, comme en OpenMP, pour une machine à mémoire partagée. Pour une machine hybride utilisant multicoeurs et GPUs, un parallélisme de tâches reste à faire. Dans l'équipe SATANAS, en particulier dans l'EPI INRIA Runtime, a été développé un runtime, StarPU [2], facilitant l'exécution de tâches parallèles sur des architectures hybrides. Le but du mémoire est donc de restructurer le parallélisme et les structures de données afin de générer automatiquement un graphe de tâches parallèles à l'aide de StarPU.

2 Sujet

L'objectif de ce mémoire est dans un premier temps de proposer une méthode pour la génération de graphe de tâches parallèles à partir de la représentation du calcul parallèle manipulant des tableaux multidimensionnels. Cette restructuration du parallélisme, dépendant de l'architecture, consiste à choisir les structures de données et à partitionner le calcul en tâches parallèles. La principale difficulté de cette de génération de code repose sur le choix des structures de données. On pourra s'inspirer des travaux passés[3, 4] basés sur le modèle polyhédrique pour en dériver un premier algorithme et proposer un modèle de performance lié au volume de données communiquées entre les tâches. Lorsque plusieurs solutions peuvent convenir, aussi bien concernant les structures de données que le graphe de tâches, on générera autant de versions de codes différentes.

La second partie du travail consistera à réaliser cette génération de code dans le compilateur développé dans le projet ANR et à analyser les performances sur des architectures hybrides multicoeurs/GPUs. Enfin, on pourra étendre la méthode pour des architectures multinoeuds (plusieurs noeuds de calculs, composés chacun de multicoeurs et de GPUs), en utilisant la bibliothèque de communication MPI (Message Passing Interface) au travers du runtime StarPU.

Références

[1] Projet petaqcd. `www.petaqcd.org`.

[2] Cédric Augonnet, Samuel Thibault, Raymond Namyst, and Pierre-André Wacrenier. StarPU : A Unified Platform for Task Scheduling on Heterogeneous Multicore Architectures. *Concurrency and Computation : Practice and Experience, Special Issue : Euro-Par 2009*, 23 :187–198, February 2011.

[3] Qingda Lu, Christophe Alias, Uday Bondhugula, Thomas Henretty, Sriram Krishnamoorthy, J. Ramanujam, Atanas Rountev, P. Sadayappan, Yongjian Chen, Haibo Lin, and Tin-fook Ngai. Data layout transformation for enhancing data locality on nuca chip multiprocessors. In *Proceedings of the 2009 18th International Conference on Parallel Architectures and Compilation Techniques*, pages 348–357, Washington, DC, USA, 2009. IEEE Computer Society.

[4] Philippe Clauss and Benoît Meister. Automatic memory layout transformations to optimize spatial locality in parameterized loop nests. *SIGARCH Comput. Archit. News*, 28 :11–19, March 2000.