Proposition de recherche doctorale

Etude de transformations et d'optimisations de code parallèle statique ou dynamique pour architecture manycore

Mots clés :

- **Directeur de thèse** : albert COHEN
- **Co-encadrant(s)** :
- **Unité de recherche** : Laboratoire inconnu!
- **Ecole doctorale** : École Doctorale Informatique, Télécommunications, Électronique de Paris
- **Domaine scientifique principal**: Divers

Résumé du projet de recherche (Langue 1)

{{{Contexte}}} L'évolution courante des architectures de processeurs impacte directement le calcul haute performance et, ainsi, la topologie des futures générations de supercalculateurs. En effet, une transition s'effectue : les processeurs utilisés jusqu'à présent possèdent quelques cœurs de calculs dits généralistes (de l'ordre d'une dizaine) tandis que les matériels spécialisés comportent plusieurs centaines de cœurs simplifiés. La première catégorie tend à augmenter le nombre de cœurs et, surtout, à élargir les unités de calcul, permettant ainsi d'augmenter les performances de calcul. En revanche, les matériels spécialisés demandent une programmation massivement parallèle et commencent à proposer des cœurs de plus en plus complexes. Ainsi, la programmation parallèle multi-niveaux (SIMD - Single Instruction Multiple Data et multithread) devient de plus en plus importante et cruciale pour atteindre des performances intéressantes. Mais cette programmation ne doit pas entraver l'exploitation efficace de plusieurs milliers de cœurs, comme c'est déjà le cas dans les supercalulateurs de classe pétaflopique. Ainsi, il convient de proposer des méthodes permettant d'exploiter de façon efficace et dynamique les unités de calcul complexe dans un environnement massivement parallèle. {{{Objectif de la thèse}}} Le but de cette thèse est de concevoir et d'implémenter des transformations de code permettant une exploitation efficace des futures architectures « manycore » dans un contexte massivement parallèle. Ainsi, en tenant compte de l'aspect parallèle d'une application, il sera alors possible de définir et tester des optimisations visant à exploiter cette sémantique pour tirer profit des unités de calcul vectorielle (type SIMD). {{{Déroulement de la thèse}}} Le plan de travail de cette thèse sera le suivant : après une étude bibliographique sur les solutions existantes permettant d'exploiter des unités fonctionnelles vectorielles (vectorisation, transformation de code, …) ainsi que la programmation flexible massivement parallèle (MPI, OpenMP, OpenCL, …), la première étape sera de concevoir une transformation visant à adapter la granularité d'un code massivement multithread en fonction de l'architecture cible. S'appuyant sur la norme OpenCL, cette transformation devra prendre en compte les contraintes architecturales et les contraintes d'ordonnancement afin de produire le code le plus efficace possible. Un point de départ pourrait être la transformation Deep Jam, publiée à la conférence PACT en 2005. Dans un second temps, il conviendra d'évaluer cette transformation sur plusieurs benchmarks représentatifs (solveurs CFD, MHD, transport) et de proposer des améliorations en vue d'une exploitation dans un supercalculateur hybride. Par la suite, plusieurs pistes seront à traiter. Cette optimisation devra être accompagnée d'autres transformations facilitant son application ainsi qu'un ou plusieurs modèles de coût permettant une adaptation dynamique de son comportement : la prise en compte de la charge de travail et de la bande passante mémoire disponible sur les architectures cibles sera nécessaire pour adapter la granularité des threads exécutés. Nous pouvons résumer le plan de travail ainsi : -*5 mois de bibliographie sur l'architecture des processeurs haute performance ainsi que sur le modèle d'exécution multithreading et sur les algorithmes de vectorisation -*6 mois sur la mise en place d'une première transformation permettant l'exploitation de plusieurs plateformes d'exécution à partir d'un code massivement multithread. Un point de départ de programmation massivement parallèle pourra être alors le modèle de programmation OpenCL. -*3 mois sur l'évaluation et l'optimisation dirigée par les applications et benchmarks internes. Ces évaluations se feront sur un supercalculateur hybride CPUs/GPUs. -*6 mois sur l'évolution potentielle vers un modèle dynamique plus intelligent. -*4 mois sur la prise en compte d'une analyse statique pour améliorer les décisions du runtime et ainsi augmenter les performances