Rapport d'avancement des travaux de thèse pour l'année 2010

Doctorant : Emna KALLEL

Directeur de thèse : Pr. Mohamed ABID

Titre : Compilation efficace pour les architectures multicœurs

1- Contexte

Depuis le lancement du premier ordinateur sur le marché, les demandes en capacité de calcul sont devenues de plus en plus importantes. Avec l'évolution des demandes de performances applicatives, les concepteurs de processeurs sont confrontés à un problème : L'augmentation des capacités informatiques est tributaire de la puissance, et le fait d'augmenter la puissance nécessite de gérer aussi les niveaux de dissipation. À cela s'ajoutent les demandes des industriels qui souhaitent des ordinateurs moins encombrants, des ordinateurs portables plus fins et plus légers, et un encombrement réduit pour les systèmes de bureau [1]. Ces verrous technologiques empêchent d'aller plus loin dans l'implantation efficace des processeurs mono-cœurs traditionnels. En fait, le schéma d'évolution des architectures subit actuellement un bouleversement : la multiplication des cœurs de processeur sur une même puce. Cette évolution technologique augmente les performances et la productivité dans des ordinateurs de plus petite taille capables d'exécuter simultanément plusieurs applications complexes et de réaliser davantage de tâches en moins de temps. En fait, les ordinateurs parallèles, ou multiprocesseurs permettent de diminuer le temps d'exécution des programmes car ils sont capables d'exécuter plusieurs instructions simultanément. Leur programmation n'est pas facile, en raison de la complexité du schéma d'exécution d'un programme. Deux approches sont utilisées en pratique :

- La programmation parallèle dite « de bas niveau », comme avec MPI[2]: le programmeur prend en charge la distribution des calculs sur les processeurs et décrit explicitement les communications entre les processeurs.
- La programmation parallèle dite « de haut niveau » : langage de haut niveau comme fortran est utilisé, comme avec OpenMP[5] ou High Performance Fortran HPF[4], et la distribution des calculs et la gestion des communications sont laissés à la charge du compilateur.

La présente thèse s'intéresse à la programmation parallèle de haut niveau en adaptant les langages de programmation et les compilateurs aux nouvelles architectures multi-cœurs à fin de bien exploiter cette montée en puissance.

2- Problématique

La diffusion des processeurs multi-cœur généralistes et spécialisés, tels les processeurs graphiques, permet de proposer des puissances de calcul potentielles extraordinaires. Cependant la programmation d'applications parallèles est beaucoup plus complexe que la programmation des applications séquentielles. Les problèmes rencontrés sont multiples et de différentes natures : algorithmique et extraction du parallélisme de l'application,

ordonnancement et placement des calculs sur les processeurs, gestion des accès concurrents aux données de la mémoire partagée, gestion des communications entre processeurs, etc. Aussi l'exploitation efficace de ces nouvelles architectures repose sur une programmation parallèle à multiples niveaux (parallélisme des tâches [6], des données [3] [7]) qui complexifient la mise en œuvre des applications.
L'objectif principale de cette thèse est de décharger le programmeur au maximum des problèmes rencontrés lors de la programmation parallèle.
Ainsi plusieurs approches ont été proposées dans ce sens. Par exemple, les approches de parallélisation du code séquentiel au niveau du parallélisme de données comme pluto[7] ou bien les environnements de réalisation haut niveau des applications parallèles comme le système accélérateur [8] qui fournit des opérations de donnée parallèle à haut niveau pour un langage de programmation impérative convenable, C#. Cependant, la conception des applications parallèles à haut niveau nécessite :

- Une méthodololologie de conception haut niveau qui permet d'exprimer le parallélisme de l'application.
- Une maitrise et analyse de l'application
- Une traduction du modèle de programmation haut niveau vers le bas niveau en adoptant un compilateur spécifique du modèle tout en gardant le niveau de parallélisme

A fin de mener bien la programmation parallèle haut niveau, une problématique majeur est de trouver un bon compromis entre le niveau d'abstraction du modèle de programmation et sa capacité à garantir une exécution parallèle efficace.

3- Travaux future

Le grand besoin de la performance et d'efficacité a conduit l'industrie informatique vers les systèmes multi-cœurs. Ces systèmes sont devenus la norme d'industrie dans presque tous les segments du marché informatique. Pour utiliser ces systèmes d'une manière efficace, plusieurs recherches ont été consacrées au développement des paradigmes de programmation explicitement parallèles, comme les nouvelles techniques de compilateur.
Dans cette thèse, nous avons étudié des différentes techniques et méthodologies de parallélisation automatique et optimisation du code.
Comme travail future, nous visons à proposer une méthodologie facile et efficace de programmation pour écrire les programmes parallèle au niveau compilateur en exploitant les directives de compilation.

Références

[1] wikipedia Online available at http://fr.wikipedia.org
[2] Message Passing Interface Forum. http://www.mpi-forum.org/
[3] Compiler support for general-purpose computation on GPUs, Yu-Te Lin · Peng-Sheng Chen Springer Science+Business Media, LLC 2008.
[4] B. Chapman, H. Zima, and P. Mehrotra. Extending HPF for advanced data-parallel applications. Parallel & Distributed Technology: Systems & Applications, Jan 1994.
[5] OpenMP Architecture Review Board. OpenMP Application Program Interface, version 3.0. OpenMP Architecture

[6] A. Lim, S. Liao, and M. Lam. Blocking and array contraction across arbitrarily nested loops using affine partitioning. In ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, pages 103–112, 2001.
[7] Uday Bondhugula, Albert Hartono, J. Ramanujam, P. Sadayappan “A Practical Automatic Polyhedral Parallelizer and Locality Optimizer” June 7–13, 2008, Tucson, Arizona, USA.
[8]. Tarditi D, Puri S, Oglesby J (2006) Accelerator: using data parallelism to program gpus for general purpose uses. In: The 12th international conference on architectural support for programming languages and operating systems, San Jose, California, USA, October 2006, pp 57–68