UNIVERSITÉ de VERSAILLES ST-QUENTIN EN YVELINES

Habilitation à diriger des recherches

Discipline :

INFORMATIQUE

Présentée par :

Sid-Ahmed-Ali TOUATI

Titre :

Méthodes d'optimisations de programmes bas niveau

On Backend Code Optimisation Methods

Soutenue le mercredi 30 juin 2010 devant le jury suivant :

Pr.	Jens Knoop	Université technique de Vienne	Rapporteur
Pr.	Jagannathan Ramanujam	Université de Louisiane	Rapporteur
Pr.	Denis Trystram	Institut polytechnique de Grenoble	Rapporteur
Dr.	Christian Bertin	Directeur de centre d'expertise chez STMicroelectronics	Examinateur
Dr.	Alain Darte	École normale supérieure de Lyon	Examinateur
Pr.	William Jalby	Université de Versailles St-Quentin en Yvelines	Examinateur
Pr.	Pascal Sainrat	Université Paul Sabatier de Toulouse	Président

Thèse d'habilitation préparée au sein des laboratoires PRiSM et de l'INRIA-Saclay

Préface

"... Was this is an example concerning foundational research based on rigorous mathematical and logical modelling and reasoning. I would like to single out as another example of the uniqueness and quality of the research of the applicant an example from another pole of the methodological spectrum applied by the applicant in his research, concerning practical experiments conducted in the search of hard problems, i.e. an example from the engineering pole of research. ... Moreover, I would like to add that the research work of the applicant is methodological sound, that it presents new and important scientific insights in theory and practice, and clearly demonstrate the ability of the applicant to contribute to the development and advancement of the field. ... "

Professor Jens Knoop
Head of the institute of computer science
Technical university of Vienna (Austria).

"... The habilitation thesis and Dr. Touati's work over the last ten years demonstrate that he is highly motivated and that he has the important ability to quickly grasp the central issues, and ask and answer the most important questions concerning research problems. His strong mathematical and reasoning abilities have added to his broad background for continued important research work. ..."

Professor Jagannathan Ramanujam
John E. and Beatrice L. Ritter distinguished professor
Louisiana State University (USA)

"... J'ai apprécié l'effort de Sid Touati *pour fournir un document complet et cohérent. Il est clair dans l'ensemble. Les résultats techniques sont introduits avec précision. La lecture est parfois aride, mais le texte est ponctué de nombreuses figures très claires. Les nombreux résultats sont obtenus par des techniques variées qui montrent un large spectre de connaissances et une bonne culture informatique. Sid* Touati *est aujourd'hui un spécialiste reconnu du domaine de la compilation de bas niveau sur les processeurs modernes. Le travail produit est de qualité, ce qui est attesté par de nombreuses publications (8 journaux internationaux parmi les meilleurs du domaine et une dizaine de conférences de bon niveau). Le travail est bien équilibré entre des analyses conceptuelles parfois sophistiquées (complexité, analyses d'algorithmes, preuves d'optimalité, heuristiques) et des réalisations expérimentales (production de codes en logiciels libres, disponibles sur le site web de Sid* Touati*). ..."*

Professor Denis Trystram
Distinguished professor at Grenoble Institute of Technology (France)

I am sincerely grateful to this honourable committee for providing me the title of "habilité à diriger des recherches". Many thanks to professors Trystram, Ramanujam and Knoop for their valuable reviews.

My first thinking after getting this title goes to my parents that were unable to attend today.

Then, I would like to thank professor William Jalby for his full support to me at the university. I must never forget the support of doctor Albert Cohen at INRIA.

I sincerely thank doctor Christine Eisenbeis, my first PhD advisor, because she helped me to go into this difficult but exciting field of research.

Thank you doctor Alain Darte for your detailed remarks every-time and everywhere to improve the quality of my research.

At the end, I am grateful to all my students that worked and are still working hard to finish their PhD.

After one decade of continuous effort in publishing, teaching, implementing, experimenting, advising, preparing projects, travelling, discussing, studying, thinking and reading, we arrive to the most popular time of a defence, which is the time of celebrating our success.

Discourse of Sid-Ahmed-Ali Touati
June 30th, 2010
Associate professor at the University of Versailles Saint-Quentin en Yvelines

Résumé

Ce manuscrit synthétise plus d'une décade de notre recherche académique sur le sujet d'optimisation de codes bas niveau, dont le but est une intégration dans un compilateur optimisant ou dans un outil d'optimisation semi-automatique. Dans les programmes bas niveau, les caractéristiques du processeur sont connues et peuvent être utilisées pour générer des codes plus *en harmonie* avec le matériel.

Nous commençons notre document par une vue générale sur le problème d'ordonnancement des phases de compilation. Actuellement, des centaines d'étapes de compilation et d'optimisation de codes existent; un problème fondamental et ouvert reste de savoir comment les combiner et les ordonner efficacement. Pour pallier rapidement cette difficulté, une stratégie du moindre effort consiste à appliquer une compilation itérative en exécutant successivement le programme avant de décider de la technique d'optimisation de code à employer et avec quels paramètres. Nous prouvons que l'approche de compilation itérative ne simplifie pas fondamentalement le problème, et l'utilisation de modèles statiques de performances reste un choix raisonnable.

Un problème classique de conflit entre deux étapes de compilation est celui qui lie l'allocation de registres et l'ordonnancement d'instructions. Nous montrons comment gérer efficacement cet antagonisme en séparant les contraintes de registres des contraintes d'ordonnancement d'instructions. Cela est possible grâce à la notion de saturation en registres (RS), qui est le besoin maximal en registres pour tous les ordonnancements possibles d'un graphe. Nous apportons une contribution formelle et une heuristique efficace, qui permettent la détection de contraintes de registres toujours vérifiées; ils peuvent par conséquent être négligées.

Nous introduisons la plate-forme SIRA, qui permet de garantir l'absence de code de vidage avant l'ordonnancement d'instructions. SIRA est un modèle basé sur la théorie des graphes permettant de borner le besoin maximal en registres pour tout pipeline logiciel, sans altérer, si possible, le parallélisme d'instructions. SIRA modélise les contraintes cycliques des registres dans différentes architectures possibles : avec plusieurs types de registres, avec tampons ou files d'attente, et avec des bancs de registres rotatifs. Nous apportons une heuristique efficace qui montre des résultats satisfaisants, que ce soit comme outil indépendant, ou comme passe intégrée dans un vrai compilateur.

Dans le contexte des processeurs exhibant des retards d'accès aux registres (VLIW, EPIC, DSP), nous attirons l'attention sur le problème qui peut survenir lorsque les contraintes de registres sont traitées avant l'ordonnancement d'instructions. Ce problème est la création de circuits négatifs ou nuls dans le graphe de dépendances de données. Nous montrons comment éliminer ces circuits indésirables dans le contexte de SIRA.

SIRA définit une relation formelle entre le nombre de registres alloués, le parallélisme d'instructions et le facteur de déroulage d'une boucle. Nous nous basons sur cette relation pour écrire un algorithme optimal qui minimise le facteur de déroulage tout en sauvegardant le parallélisme d'instructions et en garantissant l'absence de code de vidage. D'après nos connaissances, ceci est le premier résultat qui démontre que le compactage de la taille de code n'est pas un objectif antagoniste à l'optimisation des performances de code.

L'interaction entre la hiérarchie mémoire et le parallélisme d'instructions est un point central si l'on souhaite réduire le coût des latences d'opérations de chargement. Premièrement, notre étude pratique avec des micro-benchmarks montre que les processeurs superscalaires ayant une exécution dans le désordre ont un *bug* de performances dans leur mécanisme de désambiguation mémoire. Nous montrons ensuite qu'une vectorisation des opérations mémoire résoud ce problème pour des codes réguliers. Deuxièmement, nous étudions l'optimisation de préchargement de données pour des codes VLIW embarqués irréguliers.

Finalement, avec l'arrivée des processeurs multicœurs, nous observons que les temps d'exécution des programmes deviennent très variables. Afin d'améliorer la reproductibilité des résultats expérimentaux, nous avons conçu le *Speedup-Test*, un protocole statistique rigoureux. Nous nous basons sur des tests statistiques connus (tests de Shapiro-Wilk, F de Fisher, de Student, de Kolmogorov-Smirnov, de Wilcoxon-Mann-Whitney) afin d'évaluer si une accélération observée du temps d'exécution médian ou moyen est significative.

Mots clés : parallélisme d'instructions, ordonnancement d'instructions, allocation de registres, saturation en registres, pipeline logiciel, programmation linéaire, programmation linéaire en nombres entiers, hiérarchie mémoire, évaluation des performances des programmes, compilation, optimisation de code.

Abstract

This manuscript is a synthesis of our research effort since one full decade on the topic of low level code optimisation, devoted to an integration in a compiler backend or in a semi-automatic optimisation tool. At the backend level, processor characteristics are known and can be used to generate codes using the underlying hardware more efficiently.

We start our document by a global view on the phase ordering problem in optimising compilation. Nowadays, hundreds of compilation passes and code optimisation methods exist, but nobody knows exactly how to combine and order them efficiently. Consequently, a best effort strategy consists in doing an iterative compilation by successively executing the program to decide about the passes and optimisation parameters to apply. We prove that iterative compilation does not fundamentally simplify the problem, and using static performance models remains a reasonable choice.

A well known phase ordering dilemma between register allocation and instruction scheduling has been debated for long time in the literature. We show how to efficiently decouple register constraints from instruction scheduling by introducing the notion of register saturation (RS). RS is the maximal register need of all the possible schedules of a data dependence graph. We provide formal methods for its efficient computation, that allows to detect obsolete register constraints. Consequently, they can be neglected from the instruction scheduling process.

In order to guarantee the absence of spilling before instruction scheduling, we introduce the SIRA framework. It is a graph theoretical approach that bound the maximal register need for any subsequent software pipelining, while saving instruction level parallelism. SIRA model periodic register constraints in the context of multiple register types, buffers and rotating register files. We provide an efficient heuristic that show satisfactory results as a standalone tool, as well as an integrated compilation pass inside a real compiler.

In the context of processors with architecturally visible delays to access registers (VLIW, EPIC, DSP), we highlight an open problem that arises when register constraints are handled before instruction scheduling. This problem is the creation of non-positive cycles inside data dependence graphs. We show how to remove these undesirable cycles in the context of SIRA.

SIRA defines a formal relationship between the number of allocated registers, the instruction level parallelism and the loop unrolling factor. We use this relationship to write an optimal algorithm that minimises the unrolling factor while saving instruction level parallelism and guaranteeing the absence of spilling. As far as we know, this is the first result in the literature proving that code size compaction and code performance are not antagonistic optimisation objectives.

The interaction between memory hierarchy and instruction level parallelism is of crucial issue if we want to hide or to tolerate load latencies. Firstly, we practically demonstrate that superscalar out-of-order processors have a performance bug in their memory disambiguation mechanism. We show that a load/store vectorisation solves this problem for regular codes. For irregular codes, we study the combination of low level data pre-loading and prefetching, designed for embedded VLIW processors.

Finally, with the introduction of multicore processors, we observe that program execution times may be very variable in practice. In order to improve the reproducibility of the experimental results, we design the *Speedup-Test*, which is a rigorous statistical protocol. We rely on well known statistical tests (Shapiro-wilk's test, Fisher's F-test, Student's t-test, Kolmogorov-Smirnov's test, Wilcoxon-Mann-Whitney's test) to evaluate if an observed speedup of the average or the median execution time is significant.

Keywords : Instruction-Level Parallelism, Instruction Scheduling, Register Allocation, Register Saturation, Software Pipelining, Linear Programming, Integer Linear Programming, Memory Hierarchy, Program Performance Evaluation, Compilation, Code Optimisation

Contents