High Performance by Exploiting Information Locality through Reverse Computing

Mouad Bahi

Thèse

High Performance by Exploiting Information Locality through Reverse Computing

Présentée et soutenue publiquement le 21 décembre 2011 par

Mouad Bahi

pour l'obtention du Doctorat de l'université Paris-Sud

Jury

Dr.	Christine	Eisenbeis	INRIA Saclay - Île-de-France	Directrice de thèse
Prof.	Claire	Hanen	Université Paris-Ouest Nanterre La Défense	Rapporteur
Dr.	Erven	Rohou	INRIA Rennes - Bretagne Atlantique	Rapporteur
Prof.	Jean-Luc	Gaudiot	University of California - Irvine	Examinateur
Prof.	Yannis	Manoussakis	Université Paris-Sud	Examinateur
Dr.	Claude	Tadonki	Mines ParisTech	Examinateur

INRIA Saclay - Île-de-France
Université Paris-Sud

Alchemy project
Ecole doctorale LRI

Contents

Ce rapport est une introduction au manuscrit de thèse intitulé "`High Performance by Exploiting Information Locality through Reverse Computing`". Il aborde le contexte et les principales problématiques de recherche auxquelles nous nous sommes intéressées. Il donne un aperçu des solutions proposées, puis présente l'organisation du manuscrit de thèse. Le manuscrit a eté rédigé en anglais et peut être consulté à la bibliothèque de l'université Paris-Sud XI.

1 Abstract

Les trois principales ressources du calcul sont le temps, l'espace et l'énergie, les minimiser constitue un des défis les plus importants de la recherche de la performance des processeurs. Cependant, la minimisation de l'une de ces ressources à tout prix nécessite une quantité disproportionnée des autres ressources. L'usage de la mémoire peut être réduit au prix d'une exécution plus lente du programme, et le temps de calcul peut être réduit au prix d'une augmentation de la consommation d'énergie. C'est pourquoi trouver un compromis entre ces trois facteurs est devenu le défi de la théorie de calcul.

Dans cette thèse, nous nous intéressons à un quatrième facteur qui est l'information. L'information a un impact direct sur ces trois facteurs, et nous montrons comment elle contribue ainsi à l'optimisation des performances.

Landauer [5] a montré que c'est la destruction - logique - d'information qui coûte de l'énergie, ceci est un résultat fondamental de la thermodynamique en physique. Sous cette hypothése, un calcul ne consommant pas d'énergie est donc un calcul qui ne détruit pas d'information. On peut toujours retrouver les valeurs d'origine et intermédiaires à tout moment du calcul, *le calcul est réversible.*

L'information peut être portée non seulement par une donnée mais aussi par le processus et les données d'entrée qui la génèrent. Quand un calcul est réversible, on peut aussi retrouver une information au moyen de données déjà calculées et du calcul inverse. Donc, *le calcul réversible améliore la localité de l'information.* Par exemple, pour l'instruction c: = a + b, l'informations dans *(a, b, c), (a, b), (a, c)* ou *(b, c)* est la même, donc pas besoin de garder les trois valeurs en vie en même temps car nous pouvons toujours recalculer la troixième valeur à partir des deux autres.$'a'$ peut être calculé à partir de *(b, c)* et $'b'$ à partir de *(a, c)* par une simple soustraction. Donc les valeurs qui portent la même information peuvent partager le même registre. Par conséquent on dit que *la localité de l'information peut optimiser l'espace de stockage.*

Pour tirer profit de la localité de l'information, nous étudions la conservation de l'information au cours d'un processus irrversible. Un calcul conventionnel est à priori un calcul irréversible et ne conserve pas l'information. L'addition de deux nombres, par exemple, détruit l'information, sauf si nous conservons l'un de ces nombres. Cela nous a conduit à la question quelle est la taille minimal de l'information, que nous devons garder afin de générer toutes les valeurs d'entrée, d'intermédiaire et de sortie, sans avoir besoin d'un espace mémoire supplémentaire ?. Comme l'avantage du calcul réversible est sa capacité à conserver l'information, nous abordons la question de rendre un programme réversible en terme d'espace mémoire.

La réversibilité des programmes a été étudié par Bennett [4]. Une première manière facile pour rendre un programme réversible est de sauvergarder l'historique de calcul - variables intermédiaires - au long de l'exécution. Cependant le problème d'effacer - oublier - cette information, appelé déchet, demeure. Ce déchet peut être utilisé comme une estimation de la consommation d'énergie intrinsques des programmes et le minimiser est

notre objectif. Bennett [4] a prouvé que si la sortie peut être calculé à partir de l'entrée, alors il existe un moyen réversible pour calculer l'entrée à partir de la sortie tout en éliminant le déchets de calcul. Cela peut-être au prix d'un espace de stockage additionel pour stocker toutes les états intermédiaires au cours du calcul. C'est pourquoi nous nous intréssons à étudier la complexité spatiale des programmes réversibles.

L'optimisation des performances en exploitant la localité de l'information peuvent provenir en diminuant la pression en registre et par conséquent en réduisant le spill code. Dans ce travail, nous revisitons le problème d'allocation de registres sous l'angle de calcul réversible. Tout en étant un très vieux problème en informatique, l'allocation de registres est toujours un problème préoccupant en architecture où le fossé entre la vitesse des processeurs et celle de la mémoire continue de se creuser. Dans l'allocation de registres, on peut utiliser la rematérialisation au lieu de spilling, autrement dit, on recalcule une valeur v localement à partir des valeurs encore stockées dans les registres au lieu de la garder en vie en mémoire. Par conséquent, l'hiérarchie mémoire est peu solicitée. Le recalcul est effectué de la même manière comme il est spécifié dans le programme, mais il y a une partie de l'information sur v portée par d'autres valeurs w qui ont été calculées directement ou indirectement de v, ce qui donne de nouvelles opportunités pour régénerer la valeur v: en recalculant v à l'envers de w. Par conséquent, l'une des questions que nous abordons dans cette thèse est de savoir si la rematérialisation via le calcul inverse - rematérialisation inverse - peut aider à l'amélioration de l'allocation de registres.

Cette approche, recalcul versus stockage, conduit également à augmenter significativement le parallélisme d'instructions (Cell BE), et le parallélisme de threads sur un multicore avec mémoire et/ou banc de registres partagés (GPU), dans lequel le nombre de threads dépend de manière importante du nombre de registres utilisés par un thread. Ainsi, l'ajout d'instructions du fait du calcul inverse pour la rematrialisation de certaines variables est largement compensé par le gain en paralllisme. Nos exprimentations sur le code de Lattice QCD porté sur un GPU Nvidia montrent un gain de performances atteignant 11%. nous montrons que la rematérialisation via le calcul inverse est moins couteuse en terme d'opérations ajoutées que la rematérialisation classique (recalcul directe).

2 Contribution

Le but de cette thèse est double:

1. Tout d'abord, nous abordons la question de rendre un programme réversibles en termes de compléxité spatiale. La complexité spatiale est la taille mémoire / nombre de registres nécéssaire pour effectuer un calcul dans les deux sens direct et inverse. Nous donnons une borne infèrieure de la complexité spatiale d'un DAG de calcul (graphe dirigé acyclique) avec oprations réversibles, ainsi qu'une heuristique visant à trouver le nombre minimum de registres requis pour une exécution directe et inverse d'un DAG. Nous définissons le déchet énergétique comme étant le nombre de regitres supplémentaires nécessaires ce calcul réversible. Nous avons effectué des expérimentations qui suggèrent que la taille du déchet n'est jamais plus de 50% de la taille du DAG pour un DAG avec des oprations unaires/binaires. Ce travail a été publié dans la conférence CASES'2009, International Conference on Compilers, Architecture, and Synthesis for Embedded Systems [1].

2. Deuxièmement, nous revisitons les problèmes d'allocation de registre de point de vue du calcul reversible et nous présentons une nouvelle technique de rematérialisation basée sur le recalcul inverse. Nous détaillons un algorithme heuristique pour effectuer la rematérialisation inverse et nous utilisons l'application LQCD (Lattice chromodynamique

quantique) pour démontrer qu'un gain important pouvant aller jusqu' 33% sur la pression en registre peut être obtenu. Nous montrons aussi comment le parallélisme d'instructions et le paralllisme de threads peuvent être améliorés en effectuant une allocation de registres avec du recalcul inverse. L'application de ces optimisations sur les processeurs graphiques GPU permet d'augmenter le nombre de threads par Streaming Multiprocesor (SM). Cela s'est fait sur le kernel du programme de simulation de la Lattice chromodynamique quantique (LQCD) où nous avons gagné 11% de performance. Ces résultats ont été publiés dans deux conférences internationales: ACM International Conference on Computing Frontiers- CF'2011 [3] et l'International Symposium on Computer Architecture and High Performance Computing - SBAC- PAD'2011 [2]. Ce travail a été choisi comme le Best Paper de la session Architecture, et a remporté le prix de Jùlio Salec Aude de la conference SBAC-PAD'2011.

3 Organisation du manuscrit de thèse

Après avoir présenté dans la section 1 le contexte, les problèmes de recherche auxquels nous nous sommes intéressés, et avoir présenté dans la section 2 nos contributions, nous présentons ici le plan de la thèse.

La partie 1 est consacré à expliquer en détail les concepts du calcul réversible et de la conservation de l'informations. Cette partie est composée des chapitres suivants: Chapitre 1 récapitule les travaux antrieurs relatifs à ces concepts et discute des différentes applications et approches du calcul réversible. Dans le chapitre 2, nous nous attaquons au problème d'inversement d'un programme et nous examinons le coût mémoire de la réversibilité.

La partie 2 de cette thèse décrit les diffrentes approches pour améliorer les performances des programmes en exploitant la localité de l'informations via le calcul inverse. Cette partie est organisée comme suit: Dans le chapitre 3, nous discutons les travaux antérieurs sur les techniques de l'allocation de registres liés à notre travail, dans l'objectif de réduire la pression en registre, . Dans le chapitre 4, nous présentons une nouvelle technique de rematérialisation basée basée sur le calcul inverse pour résoudre le problème d'accès à la mémoire. Les chapitres 5 et 6 adressent les problématiques du parallélisme d'instructions et de threads sur processeurs multi-coeur et montrent comment la rematerialisation inverse peut les améliorer en augmentant la disponibilité des ressources, typiquement les registres. Nous présentons ensuite les résultats de nos expérimentations sur Cell BE et NVIDIA GPU. Enfin, le chapitre 7 contient la conclusion de ce travail et propose des orientations possibles pour des recherches futures.

Bibliography

[1] Mouad Bahi and Christine Eisenbeis. Spatial complexity of reversibly computable dag. In *International Conference on Compilers, Architecture, and Synthesis for Embedded Systems - CASES*. ACM, 2009.

[2] Mouad Bahi and Christine Eisenbeis. High performance by exploiting information locality through reverse computing. In *International Symposium on Computer Architecture and High Performance Computing - SBAC-PAD*. IEEE Computer Society, 2011.

[3] Mouad Bahi and Christine Eisenbeis. Rematerialization-based register allocation through reverse computing. In *International Conference on Computing Frontiers - CF*. ACM, 2011.

[4] D. H. Bennett. *Logical reversibility of a computation*. IBM J. Res. Dev., 1973.

[5] R. Landauer. *Irreversibility and heat generation in the computing process*. IBM Journal of Research and Development, 1961.