Proposition de thèse de doctorat en informatique

Vers des logiciels plus sobres en énergie avec de nouveaux modèles de coûts pour les compilateurs optimisants

Résumé. Une informatique plus sobre en énergie passe par l'obtention de logiciels consommant moins tout en offrant de bonnes performances. De tels logiciels nécessitent des compilateurs optimisants capables de générer un code binaire adapté à la machine cible et efficace en termes de temps et d'énergie. Le compilateur doit donc posséder un modèle de coût donnant une fonction objectif pilotant la recherche d'un optimum. L'objectif de cette thèse est de concevoir un tel modèle de coût couplant le temps à l'énergie, et de l'intégrer aux passes d'optimisations des compilateurs.

Mots-clefs. Compilation, optimisation, énergie, machine learning, modèles de coûts, architecture des ordinateurs.

Contexte. L'énergie consommée par le renouvellement des serveurs informatiques pénalise fortement les gains de consommation obtenus par l'utilisation de serveurs plus modernes : la fabrication d'un serveur consomme une énergie équivalente à plusieurs années de son utilisation, sans compter les ressources consommées comme l'eau ou les minéraux, parfois critiques et non renouvelables.

Le projet *Gabian* de Mines Paris étudie l'allongement de la durée de vie de ces serveurs informatiques en utilisant une plateforme expérimentale d'une centaines de serveurs hétérogènes déclassés. Considérés comme des déchets, ils peuvent donc être altérés et modifiés pour mener différents types d'expériences sur la consommation énergétique des applications, leur efficacité en temps, ainsi que sur le comportement des différents composants matériels quand ces derniers vieillissent.

Équipée de sondes permettant de mesurer la consommation énergétique et la température, la plateforme offre un terrain expérimental concret pour étudier finement le comportement des logiciels et leurs améliorations, notamment sur les couches basses composées du système d'exploitation et des compilateurs.

Gabian, en allongeant la durée de vie des machines, doit contribuer à une informatique plus sobre en énergie et en ressources minérales, en évaluant l'impact de ce prolongement sur l'efficacité en temps et en énergie, ainsi que sur la fiabilité des applications, en proposant de nouvelles approches pour améliorer ces différents aspects.

Problématique. L'architecture matérielle sur laquelle le code binaire généré par un compilateur va s'exécuter est très complexe, avec plusieurs types de parallélisme disponibles – pipelines, unités vectorielles, plusieurs cœurs, un ou plusieurs GPU – ainsi qu'une hiérarchie mémoire profonde où chaque niveau a une taille, une bande passante et une latence très différentes. Cette architecture est mal exploitée par les compilateurs car mal modélisée [7].

D'autre part, il n'existe actuellement pas de modèles de coûts intégrés aux compilateurs indiquant la consommation énergétique des instructions, notamment en relation avec les accès mémoires qui ont un impact fort sur celle-ci. Même si la vitesse du code et sa consommation énergétique semblent corrélées, leur relation est encore mal comprise et nécessite un travail expérimental conséquent pour aboutir à un modèle analytique précis.

Dans les compilateurs, les *back-ends* actuels, responsables de la génération du code assembleur final après toutes les transformations, utilisent des modèles de coûts d'exécution très simples pour espérer aboutir au code le plus rapide possible, avec comme métrique le nombre de cycles processeurs [11]. Cette approche est trop simple pour permettre une prédiction pertinente du comportement réel du code généré : il manque par exemple la prise en compte des effets de cache ou de TLB.

Les *middle-ends* de ces compilateurs, responsables des optimisations, parallélisations et restructurations du code ne sont généralement pas dirigés par des modèles de coûts communs aux différentes passes : les opérations de

transformation reposent principalement sur une approche heuristique propre à chaque type de transformation, où les approximations faites sont parfois très éloignées de ce qui va réellement se passer lorsque le code final sera exécuté. Ainsi, une passe peut tout à fait prendre des décisions ayant un impact négatif sur le potentiel d'optimisation d'autres passes ayant lieu avant ou après.

L'obtention de codes efficaces en temps ou en énergie passe donc par un réexamen et une évolution des modèles de coûts utilisés par les compilateurs optimisants.

Objectif. Ce travail de thèse doit contribuer à l'amélioration des modèles de coûts dirigeant les décisions d'optimisation des compilateurs, à la fois pour le temps et pour l'énergie.

Après avoir pris connaissance de l'état de l'art du domaine (voir par exemple [1, 8, 5, 9, 6, 4, 2]) et avoir étudié la manière dont ces modèles de coût sont actuellement utilisés dans des compilateurs comme PIPS [13, 3], LLVM et GCC, l'amélioration de ces modèles va nécessiter de bien comprendre les différents paramètres impactant le temps d'exécution d'un code et sa consommation d'énergie sur une architecture spécifique de machine (types de processeurs, types de GPU, tailles et bandes passantes des mémoires disponibles). Ces paramètres seront différents d'une machine à l'autre, ce qui va certainement rendre nécessaire la conception d'un modèle paramétrique pouvant être adapté à chaque machine. Cette adaptation implique que le code binaire généré sera propre à une machine particulière et devra donc être compilé spécifiquement pour cette machine.

L'utilisation d'un modèle paramétrique commun à toutes les passes nécessitera peut-être des transformations profondes au niveau de la structure des compilateurs actuels, afin que les optimisations soient dirigées par une même fonction objectif. Plusieurs approches utilisant le *machine learning* ont été proposées [1, 12] et devront également être explorées, notamment sur les aspects paramétriques pouvant impliquer la nécessité d'un nouvel apprentissage pour chaque type de machine.

La conception des modèles ainsi que les tests utiliseront la plateforme expérimentale du projet *Gabian* de Mines Paris constituée d'une centaine de serveurs hétérogènes équipés de sondes matérielles de consommation électrique et de température. Les compteurs intégrés aux processeurs devront également être utilisés.

Une implémentation expérimentale d'un modèle de coût temps/énergie pourra être effectuée dans un compilateur, par exemple pour diriger une passe d'optimisation comme la *distribution de boucles* implémentée par Mines Paris dans GCC [10].

Profil du candidat. Notions sur les compilateurs, l'architecture des ordinateurs et le machine learning. Langages C, C++. Master 2 ou diplôme d'ingénieur en informatique. Bon niveau en anglais à l'oral et à l'écrit.

Localisation. Centre de recherche en informatique (CRI), Mines Paris, Université PSL, Campus Pierre Laffite, Sophia-Antipolis, France.

Encadrement. Georges-André Silber `georges-andre.silber@minesparis.psl.eu`, maître de conférences et Fabien Coelho, professeur.

Candidature. CV, notes, lettre de motivation et lettres de recommendations à envoyer à l'adresse email ci-dessus. Un entretien sera effectué pour les candidats sélectionnés. Date de début de la thèse le 01/10/2023, date limite de candidature le 15/5/2023.

Références

[1] Riyadh Baghdadi et al. « A Deep Learning Based Cost Model for Automatic Code Optimization ». In : *Proceedings of Machine Learning and Systems*. T. 3. 2021, p. 181-193. URL : `https://proceedings.mlsys.org/paper/2021/file/3def184ad8f4755ff269862ea77393dd-Paper.pdf`.

[2] Lorenz Braun et al. « A Simple Model for Portable and Fast Prediction of Execution Time and Power Consumption of GPU Kernels ». In : *ACM Trans. Archit. Code Optim.* 18.1 (jan. 2021). URL : `https://doi.org/10.1145/3431731`.

[3] Vincent Dornic, Pierre Jouvelot et David K. Gifford. « Polymorphic Time Systems for Estimating Program Complexity ». In : *ACM Lett. Program. Lang. Syst.* 1.1 (mars 1992), p. 33-45. URL : https://doi.org/10.1145/130616.130620.

[4] Sunpyo Hong et Hyesoon Kim. « An Analytical Model for a GPU Architecture with Memory-Level and Thread-Level Parallelism Awareness ». In : *Proceedings of the 36th Annual International Symposium on Computer Architecture*. ISCA '09. Association for Computing Machinery, 2009, p. 152-163.

[5] Dejice Jacob, Phil Trinder et Jeremy Singer. « Pricing Python Parallelism : A Dynamic Language Cost Model for Heterogeneous Platforms ». In : *Proceedings of the 16th ACM SIGPLAN International Symposium on Dynamic Languages*. DLS 2020. Virtual, USA : Association for Computing Machinery, 2020, p. 29-42. URL : https://doi.org/10.1145/3426422.3426979.

[6] Abhinav Jangda et Uday Bondhugula. « An Effective Fusion and Tile Size Model for PolyMage ». In : *ACM Transactions on Programming Languages and Systems* 42.3 (déc. 2020). URL : https://doi.org/10.1145/3404846.

[7] Charles E. Leiserson et al. « There's plenty of room at the Top : What will drive computer performance after Moore's law ? » In : *Science* 368 (6495 5 juin 2020). URL : https://science.sciencemag.org/content/368/6495/eaam9744.

[8] David Leopoldseder et al. « A Cost Model for a Graph-Based Intermediate-Representation in a Dynamic Compiler ». In : *Proceedings of the 10th ACM SIGPLAN International Workshop on Virtual Machines and Intermediate Languages*. VMIL 2018. Boston, MA, USA : Association for Computing Machinery, 2018, p. 26-35. URL : https://doi.org/10.1145/3281287.3281290.

[9] Michail Papadimitriou et al. « Automatically Exploiting the Memory Hierarchy of GPUs through Just-in-Time Compilation ». In : *Proceedings of the 17th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*. VEE 2021. Virtual, USA : Association for Computing Machinery, 2021, p. 57-70. URL : https://doi.org/10.1145/3453933.3454014.

[10] Sebastian Pop et Georges-André Silber. *Loop distribution. GNU Compiler Collection (GCC)*. Free Software Foundation, Inc. 2006. URL : https://github.com/gcc-mirror/gcc/blob/master/gcc/tree-loop-distribution.cc.

[11] R.H. Saavedra et A.J. Smith. « Performance characterization of optimizing compilers ». In : *IEEE Transactions on Software Engineering* 21.7 (1995), p. 615-628. URL : https://ieeexplore.ieee.org/document/392982.

[12] Zheng Wang et Michael O'Boyle. « Machine Learning in Compiler Optimization ». In : *Proceedings of the IEEE* 106.11 (nov. 2018).

[13] Lei Zhou. « Complexity Estimation in the PIPS Parallel Programming Environment ». In : *Proceedings of CONPAR92/VAPPV*. Laboratoire de l'informatique du parallélisme, école normale supérieure de Lyon, Lyon, France, 1992.