MINES PARIS | PSL

PhD Proposal in Computer Science

Towards more energy-efficient software with new cost models for optimizing compilers

Abstract. More energy-efficient computing requires software that consumes less power while delivering good performance. Such software requires optimizing compilers able to generate binary code adapted to the target machine and efficient in terms of time and energy. The compiler must therefore have a cost model giving an objective function that drives the search for an optimum. The objective of this thesis is to design such a cost model coupling time and energy, and to integrate it to the compiler optimization passes.

Keywords. Compilation, optimization, energy, machine learning, cost models, computer architecture.

Context. The energy consumed by the renewal of computer servers strongly penalizes the gains of consumption obtained by the use of more modern servers: the manufacture of a server consumes an energy equivalent to several years of its use, without counting the consumed resources such as water and minerals, sometimes critical and non renewable.

The *Gabian* project of Mines Paris studies the extension of the life span of these computer servers by using an experimental platform of hundred decommissioned heterogeneous servers. Considered as waste, they can be altered and modified to conduct different types of experiments on the energy consumption of applications, their time efficiency, as well as on the behavior of the different hardware components as they age.

Equipped with sensors to measure energy consumption and temperature, the platform offers a concrete experimental ground to study in detail the behavior of software and their improvements, especially on the lower layers composed of the operating system and compilers.

Gabian, by extending the life of machines, must contribute to a computing more efficient in energy and mineral resource consumption, by evaluating the impact of this extension on time and energy efficiency, as well as on the reliability of applications, by proposing new approaches to improve these different aspects.

Problem. The hardware architecture on which the binary code generated by a compiler will run is very complex, with many types of parallelism available – pipelines, vector units, multiple cores, one or more GPUs – as well as a deep memory hierarchy where each level has a very different size, bandwidth, and latency. This architecture is poorly exploited by compilers because it is poorly modeled [7].

On the other hand, there are currently no cost models integrated in compilers indicating the energy consumption of instructions, especially in relation to memory accesses which have a strong impact on it. Even if the speed of the code and its energy consumption seem to be correlated, their relationship is still poorly understood and requires significant experimental work to obtain an accurate analytical model.

In compilers, the current *back-ends*, responsible for the generation of the final assembly code after all transformations, use very simple execution cost models to hope to achieve the fastest possible code, with as metric the number of processor cycles [11]. This approach is too simple to allow a relevant prediction of the real behavior of the generated code: for example, it lacks the consideration of cache or TLB effects.

The *middle-ends* of these compilers, responsible for optimizations, parallelizations and restructurings of the code, are generally not directed by cost models common to the different passes: the transformation operations are mainly based on a heuristic approach specific to each type of transformation, where the approximations made are sometimes very far from what will really happen when the final code is executed. Thus, one pass can make decisions that have a negative impact on the optimization potential of other passes taking place before or after.

Obtaining time or energy efficient code therefore requires a re-examination and evolution of the cost models used by optimizing compilers.

Objective. This thesis work should contribute to the improvement of cost models driving compiler optimization decisions, both for time and energy.

After reviewing the state of the art in the field (see for example [1, 8, 5, 9, 6, 4, 2]) and studying how these cost models are currently used in compilers such as PIPS [12, 3], LLVM, and GCC, the improvement of these models will require a good understanding of the different parameters impacting the execution time of a code and its energy consumption on a specific machine architecture (types of processors, types of GPUs, sizes and bandwidths of the available memories). These parameters will differ from one machine to another, which will certainly make it necessary to design a parametric model that can be adapted to each machine. This adaptation implies that the generated binary code will be specific to a particular machine and will therefore have to be compiled specifically for that machine.

The use of a common parametric model for all runs may require deep transformations in the structure of current compilers, so that optimizations are driven by the same objective function. Several approaches using *machine learning* have been proposed citeBaghdadiR2021,ZhengW2018 and will also need to be explored, especially on parametric aspects that may imply the need for new learning for each type of machine.

The design of the models as well as the tests will use the experimental platform of the *Gabian* project of Mines Paris, which consists of a hundred heterogeneous servers equipped with hardware probes of power consumption and temperature. The counters integrated in the processors will also be used.

An experimental implementation of a time/energy cost model could be done in a compiler, for example to run an optimization pass like the *loop distribution* implemented by Mines Paris in GCC [10].

Skills expected and prerequisites. Good knowledge in compilers, computer architecture, and machine learning. The candidate must hold an engineering degree or a master of science. Proficiency in english, scientific writing in particular, is required.

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

Supervision. Georges-André Silber `georges-andre.silber@minesparis.psl.eu`, senior lecturer, and Fabien Coelho, professor.

Application. CV, grades, statement of purpose, and letters of recommandation to send to the above email address. Start date of the thesis: 01/10/2023. Application limit: 15/5/2023.

References

[1] Riyadh Baghdadi et al. “A Deep Learning Based Cost Model for Automatic Code Optimization”. In: *Proceedings of Machine Learning and Systems*. Vol. 3. 2021, pp. 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, and David K. Gifford. “Polymorphic Time Systems for Estimating Program Complexity”. In: *ACM Lett. Program. Lang. Syst.* 1.1 (Mar. 1992), pp. 33–45. URL: https://doi.org/10.1145/130616.130620.

[4] Sunpyo Hong and 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, pp. 152–163.

[5] Dejice Jacob, Phil Trinder, and 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, pp. 29–42. URL: https://doi.org/10.1145/3426422.3426979.

[6] Abhinav Jangda and Uday Bondhugula. “An Effective Fusion and Tile Size Model for PolyMage”. In: *ACM Transactions on Programming Languages and Systems* 42.3 (Dec. 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 June 5, 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, pp. 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, pp. 57–70. URL: https://doi.org/10.1145/3453933.3454014.

[10] Sebastian Pop and 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 and A.J. Smith. “Performance characterization of optimizing compilers”. In: *IEEE Transactions on Software Engineering* 21.7 (1995), pp. 615–628. URL: https://ieeexplore.ieee.org/document/392982.

[12] 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.