Master Internship Proposal

Optimizing Programs with Sketch-Guided Polyhedral Compilation

Location	Team Inria CAMUS, ICube Laboratory, Strasbourg (Illkirch campus) or Team Inria CASH, LIP Laboratory, Lyon, France
Advisors	Thomas Kœhler, CNRS Christophe Alias, Inria

1 Context

Recent breakthroughs in domains such as image processing, physics simulation or machine learning require massive computing power. Optimizing programs is critical to make them faster, and reduce their memory and energy consumption. However, optimizing programs is challenging.

Modern compilers automatically apply many optimizations, but are too limited, often failing to achieve satisfying performance [1]. General-purpose compilers like Clang or GCC for C/C++ mostly fail to apply whole-program optimizations across function calls and loops. Domain-specific compilers such as Halide [2] for image processing and TVM [3] for machine learning enable more aggressive optimizations, but are restricted in their input and optimizations. Polyhedral compilers [4, 5] are effective across application domains, but, when fully automatic, rely on fragile heuristics.

As a result, skilled programmers often optimize libraries (e.g. BLAS or MKL for linear algebra) and applications by hand [6], but this is time consuming and risks introducing bugs. Rather than trying to replace programmers with elusively smart compilers, we aim to combine the strengths of manual and automatic approaches through *guided* (i.e. semi-automatic) optimization techniques.

Prior work in guided optimization often relies on step-by-step recipes: transformation scripts [7], scheduling APIs [2], rewriting strategies [8], or tactics [9]. The problem with optimization recipes is that they require programmers to think about *how* to achieve their optimizations in a framework that is likely unfamiliar (e.g. intermediate languages, polyhedrons, rewrite rules). Therefore, tools where built to alleviate the difficulty of writing optimization recipes. For example, Clint allows manipulating polyhedral schedules through interactive 2D diagrams [10], and interactive optimization assistants help programmers in writing recipes [11, 12]. Instead of programmers focusing on the *how*, we would like programmers to focus on declaratively describing the *what*.

Recent work introduces guided equality saturation [13], a semi-automatic term rewriting technique that scales by allowing human insight to guide the process at key points. Guides are concise and incomplete program sketches describing *what* key optimization steps look like, while an automatic search based on equality saturation instantiates the sketch and infers the *how*. This work was identified as a promising direction for future research by the MIT PL Review [14]. However, guided equality saturation is only applicable to optimizing functional programs using rewrite rules.

The goal of this internship is to enable sketch-guided optimization of imperative programs using polyhedral compilation techniques.

2 Objectives

- Design a language enabling programmers to write sketches that are satisfied by a family of C programs. This language should enable programmers to express key optimization insights and constraints, in a style similar to informal programs written on a piece of paper, or whiteboard.

- Implement or extend a polyhedral compiler such that it only generates programs that satisfy a given sketch. As regular polyhedral compilers, it should also take as input an initial program, preserve its semantics, and optimize some objective function. The initial program semantics and the objective function will enable resolving ambiguities in the user-provided sketch.
- Use this tool to reproduce well-known optimizations on benchmarks from PolyBench, Halide, TVM, or other sources. Ideally, demonstrate that sketch-guidance enables generating faster code than fully automatic polyhedral compilers thanks to programmer insight.

3 Required and Acquired Skills

The intern should come with:

- solid imperative programming skills
- interest in (or experience with) compilation and optimization
- willingness to learn (or knowledge of) C and other programming languages

The intern is expected to acquire:

- experience in basic academic skills: reviewing literature, conducting novel research, collaborating with other researchers, presenting research and communicating ideas
- greater understanding of compilers, program optimization and high performance applications
- the ability to combine theory (polyhedral model) with practice (high performance code)

4 Practical Aspects

If you are interested, send us an e-mail with a short statement of interest, questions you might have, the desired start date and duration of your internship (4-6 months), as well as a 1 page CV and your Master transcripts. If the internship goes well, there is potential to pursue a PhD in similar topics.

References

[1] Paul Barham and Michael Isard. "Machine Learning Systems are Stuck in a Rut". In: *Proceedings of the Workshop on Hot Topics in Operating Systems, HotOS 2019, Bertinoro, Italy, May 13-15, 2019*. ACM, 2019, pp. 177–183. URL: https://doi.org/10.1145/3317550.3321441.

[2] Jonathan Ragan-Kelley, Andrew Adams, Sylvain Paris, Marc Levoy, Saman P. Amarasinghe, and Frédo Durand. "Decoupling algorithms from schedules for easy optimization of image processing pipelines". In: *ACM Trans. Graph.* 31.4 (2012), 32:1–32:12. URL: https://doi.org/10.1145/2185520.2185528.

[3] Tianqi Chen et al. "TVM: An Automated End-to-End Optimizing Compiler for Deep Learning". In: *13th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2018, Carlsbad, CA, USA, October 8-10, 2018*. Ed. by Andrea C. Arpaci-Dusseau and Geoff Voelker. USENIX Association, 2018, pp. 578–594. URL: https://www.usenix.org/conference/osdi18/presentation/chen.

[4] Uday Bondhugula, Albert Hartono, Jagannathan Ramanujam, and Ponnuswamy Sadayappan.
"A practical automatic polyhedral parallelizer and locality optimizer". In: *Proceedings of the 29th ACM SIGPLAN Conference on Programming Language Design and Implementation*. 2008, pp. 101–113.

[5] Arun Thangamani, Vincent Loechner, and Stéphane Genaud.
"A Survey of General-purpose Polyhedral Compilers".
In: *ACM Trans. Archit. Code Optim.* (June 2024). Just Accepted. ISSN: 1544-3566.
URL: https://doi.org/10.1145/3674735.

[6] Thomas M. Evans, Andrew R. Siegel, Erik W. Draeger, Jack Deslippe, Marianne M. Francois, Timothy C. Germann, William E. Hart, and Daniel F. Martin. "A survey of software implementations used by application codes in the Exascale Computing Project".
In: *Int. J. High Perform. Comput. Appl.* 36.1 (2022), pp. 5–12.
URL: https://doi.org/10.1177/10943420211028940.

[7] Lénaıc Bagnères, Oleksandr Zinenko, Stéphane Huot, and Cédric Bastoul.
"Opening polyhedral compiler's black box".
In: *Proceedings of the 2016 International Symposium on Code Generation and Optimization, CGO 2016, Barcelona, Spain, March 12-18, 2016*.
Ed. by Björn Franke, Youfeng Wu, and Fabrice Rastello. ACM, 2016, pp. 128–138.
URL: https://doi.org/10.1145/2854038.2854048.

[8] Bastian Hagedorn, Johannes Lenfers, Thomas Kœhler, Xueying Qin, Sergei Gorlatch, and Michel Steuwer. "Achieving High Performance the Functional Way: Expressing High-Performance Optimizations as Rewrite Strategies".
In: *Communications of the ACM* 66.3 (2023), pp. 89–97.

[9] Amanda Liu, Gilbert Louis Bernstein, Adam Chlipala, and Jonathan Ragan-Kelley.
"Verified Tensor-Program Optimization via High-Level Scheduling Rewrites".
In: *Proc. ACM Program. Lang.* 6.POPL (Jan. 2022). URL: https://doi.org/10.1145/3498717.

[10] Oleksandr Zinenko, Stéphane Huot, and Cédric Bastoul.
"Visual program manipulation in the polyhedral model".
In: *ACM Transactions on Architecture and Code Optimization (TACO)* 15.1 (2018), pp. 1–25.

[11] Lorenzo Chelini, Martin Kong, Tobias Grosser, and Henk Corporaal.
"LoopOpt: Declarative Transformations Made Easy".
In: *SCOPES '21: 24th International Workshop on Software and Compilers for Embedded Systems, Eindhoven, The Netherlands, November 1 - 2, 2021*. Ed. by Sander Stuijk. ACM, 2021, pp. 11–16. URL: https://doi.org/10.1145/3493229.3493301.

[12] Yuka Ikarashi, Jonathan Ragan-Kelley, Tsukasa Fukusato, Jun Kato, and Takeo Igarashi.
"Guided Optimization for Image Processing Pipelines".
In: *IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2021, St Louis, MO, USA, October 10-13, 2021*.
Ed. by Kyle J. Harms, Jácome Cunha, Steve Oney, and Caitlin Kelleher. IEEE, 2021, pp. 1–5.
URL: https://doi.org/10.1109/VL/HCC51201.2021.9576341.

[13] Thomas Koehler, Andrés Goens, Siddharth Bhat, Tobias Grosser, Phil Trinder, and Michel Steuwer. "Guided Equality Saturation".
In: *Proc. ACM Program. Lang.* 8.POPL (2024), pp. 1727–1758.
URL: https://doi.org/10.1145/3632900.

[14] URL: https://plr.csail.mit.edu/.