Senior Design

LLMs in Compiler Optimization: Challenges and Future Direction

Zarif Sadman and Apan Qasem

Recently, large language models (LLMs) have become highly effective tools, capable of handling a wide range of tasks in natural language processing and software engineering. Language models like Code Llama and Codex exhibit a good understanding of code-based tasks, such as code generation, translation, and completion of unfinished code. However, their potential applications in high performance computing (HPC) performance modeling have not been thoroughly investigated yet. While LLMs, with their capability to understand and generate code while analyzing intricate patterns, hold great promise, they rely on the availability of large volumes of training data, in the order of 10^{10} tokens. Creating robust and dynamic datasets that accurately represent real-world workloads and system behavior is a major challenge, primarily due to the enormous cost in computation time on production HPC systems. This scarcity of representative tokens may limit the development and validation of LLM models for performance modeling, optimization, and tuning. This study reviews the state-of-the-art in LLM-driven performance modeling, highlights current challenges, and proposes potential solutions.

Digital Object Identifier 10.1109/MPULS.2025.3526517
Date of current version: 27 February 2025.

Recent contributions

Leather and Cummins [5] highlighted the role of deep learning and reinforcement learning for complex optimization tasks, envisioning fully automated compilers. Cummins et al. [1] demonstrated the effectiveness of a 7B-parameter LLM in optimizing low level virtual machine (LLVM) IR for reducing code size, outperforming conventional approaches without iterative compilations. Building on this, Grubisic et al. [2] incorporated compiler-generated feedback for guided optimization, improving performance through Fast Feedback techniques. Cummins et al. [3] introduced the Meta LLM Compiler, based on Code Llama, pre-trained on extensive LLVM IR and assembly code datasets, achieving significant autotuning potential without repetitive compilations. Additionally, Chen et al. [4] explored LLMs in

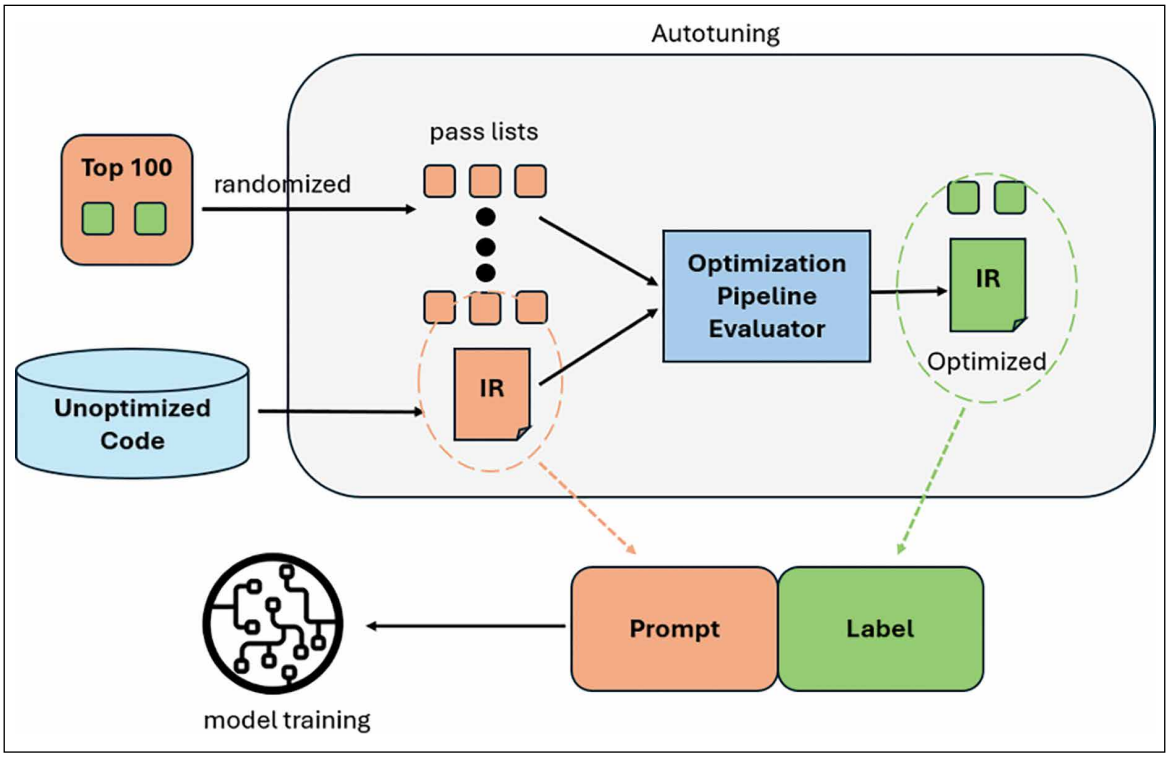

Figure 1. Workflow for training LLMs in compiler optimization. (Image adapted from [3].)

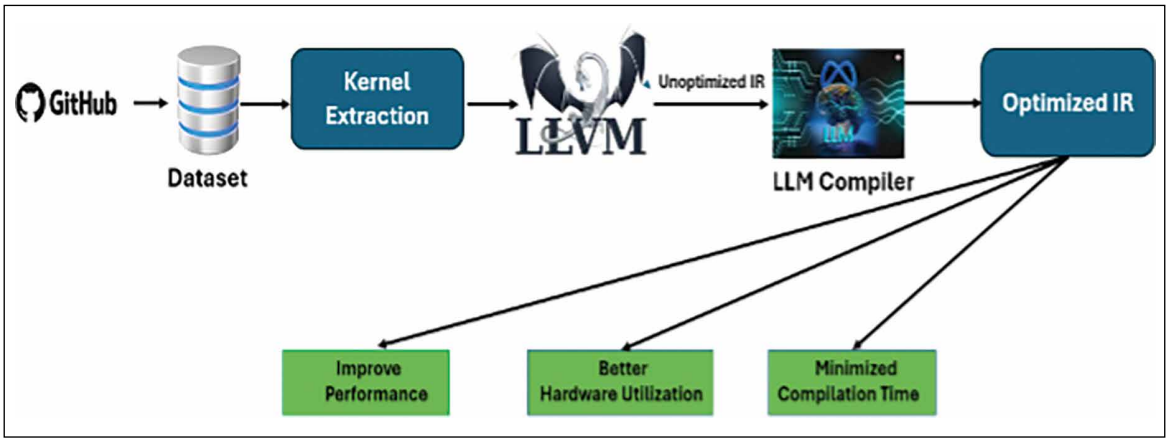

Figure 2. LLM inference workflow for compiler optimization.

HPC tasks, identifying challenges like the scarcity of specialized datasets and the complexity of scaling workloads.

LLM training and inference

In the training process (Figure 1), the input includes unoptimized code, which is compiled using various random optimization passes. Each pass generates a corresponding binary size and optimized output code. The optimization pass achieving the smallest binary size is selected, and this optimal pass sequence is paired with its respective optimized intermediate representation (IR) and the original code to create training examples. During inference (Figure 2), source code benchmarks are used to build datasets of computational routines, which are then converted into unoptimized IR using LLVM. The LLM compiler predicts optimized

IR based on learned strategies from the training phase. This optimized IR leads to better performance, efficient hardware utilization, and reduced compilation time.

Future work and challenges

In our future work, we focus on utilizing dynamic runtime data, such as execution time and memory usage, in both training and inference processes to achieve adaptive and context-aware optimization. This approach moves beyond static data, leveraging runtime metrics to tailor optimizations to actual execution behavior for improved efficiency. However, significant challenges remain, including the limited availability of datasets with dynamic runtime information, the complexity of scaling HPC workloads across diverse architectures, and the resource-intensive nature of training and fine-tuning LLMs. Addressing these challenges is essential for advancing the synergy between LLMs and HPC, unlocking their full potential for dynamic and scalable optimizations. ■

References

[1] C. Cummins et al., "Large language models for compiler optimization," 2023, *arXiv:2309.07062*.

[2] D. Grubisic et al., "Compiler generated feedback for large language models," 2024, *arXiv:2403.14714*.

[3] C. Cummins et al., "Meta large language model compiler: Foundation models of compiler optimization," 2024, *arXiv:240702524*.

[4] L. Chen et al., "The landscape and challenges of HPC research and LLMs," 2024, *arXiv:2402.02018*.

[5] H. Leather and C. Cummins, "Machine learning in compilers: Past, present and future," in *Proc. Forum Specification Design Lang. (FDL)*, Kiel, Germany, Sep. 2020, pp. 1–8, doi: 10.1109/FDL50818.2020.9232934.

Zarif Sadman *is currently pursuing the Ph.D. degree with the Department of Computer Science, Texas State University, San Marcos, TX, USA, specializing in compiler optimization.*

Apan Qasem *received the Ph.D. degree from Rice University, Houston, TX, USA. He has been a visiting professor at the School of Informatics, University of Edinburgh, and a visiting scholar at AMD Research. He is currently a professor with the Computer Science Department, Texas State University, San Marcos, TX, USA. He leads the Compilers Research Laboratory, where he and his students are working on a number of projects in the broad area of high-performance computing with an emphasis on developing intelligent compiler techniques for improving programmer productivity, performance, and energy efficiency. His research has received funding from the National Science Foundation (including the CAREER Award in 2013), the Department of Energy, the Department of Health and Human Services (DHHS), the Semiconductor Research Consortium (SRC), AMD, NVIDIA, Intel, and IBM (including the Faculty Award in 2008). He has coauthored over 80 peer-reviewed publications, including five that won best paper awards. He won the Presidential Distinction Award for Excellence in Teaching at Texas State in 2012 and 2024 and has been selected as a Favorite Professor by the Alpha Chi National Honor Society on four occasions. He also serves as the department's Associate Chair.*