ARIA
APPLIED RESEARCH IN ACTION

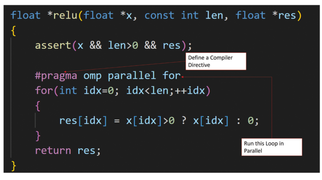
Automatically Parallelizing C/C++ code with Large Language Models

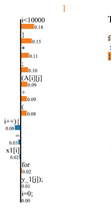
Using Large Language Model to automatically classify and specify OpenMP directives for the Sequential C/C++ code.

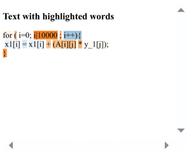
Amandeep Singh

Angela Demke Brown
ACADEMIC SUPERVISOR

Charles Boyd
INDUSTRY SUPERVISOR







PROJECT SUMMARY

Programmers have encountered significant challenges in effectively harnessing the capabilities of parallel architectures due to the inherent complexity of reasoning about parallel programs. Open Multi-Processing (OpenMP provides a way to unify code for serial and parallel application with its collection of compiler directives that mark sections of the code for parallel execution. However, parallelizing code using optimal OpenMP directives still requires developer expertise. Thus, with the advent of large Language Models capable of reasoning about code there has been much research into their ability to aid programmers in both identifying parallelizable code and then parallelizing it. However, the predictions of these models are sensitive to the non-semantic details of the input code, such as whitespace or variable naming convention.

In this study, we aim to investigate the impact of these code aesthetics on model performance. Specifically, we investigate finetuning and testing models on pre-formatted code according to the CLANG formatter. Additionally, we examine the effect of variable naming by finetuning a model on code with obfuscated variable names. We conduct these experiments on an Open-Source Large Language Model for Code, known as StarCoder and examine performance across two scenarios: identifying whether a given loop is parallelizable, and given a parallelizable loop identifying the correct type of OpenMP directive. In showing this we enhance the understanding of Large Language Models for parallelization's robustness to non-semantic changes in their inputs.

REFERENCES

1. A. a. B. J. a. C. J. Meade, "Challenges of evolving sequential to parallel code: An exploratory review," in IWPSE-EVOL'11 - Proceedings of the 12th International Workshop on Principles on Software Evolution, 2011.
2. P. P. P. S. H. A. Reed Milewicz, Negative Perceptions About the Applicability of Source-to-Source Compilers in HPC: A Literature Review, Springer International Publishing, 2021.
3. M. Hill, " A Primer on the meltdown & spectre hardware security design flaws and their important implications.," Computer Architecture Today, 2018.
4. A. a. A. A. Imam, "The Use of Natural Language Processing Approach for Converting Pseudo Code to C# Code," Journal of Intelligent Systems, p. 1388–1407, 2019.
5. R. H. a. Y. P. a. G. Oren, "Learning to Parallelize in a Shared-Memory Environment with Transformers," 2022.
6. D. G. a. S. R. a. S. Lu, "GraphCodeBERT: Pre-training Code Representations with Data Flow," in ICLR, 2021.
7. R. L. a. L. B. A. a. Y. Zi, "StarCoder: may the source be with you!," 2023.

AMD

Computer Science
UNIVERSITY OF TORONTO
Master of Science in Applied Computing