A Proposal of Automatic Parallelization using Transformer-based Large Language Models

Soratouch Pornmaneerattanatri
pornmaneerattanatri.so.pn8@is.naist.jp
Nara Institute of Science and Technology
Nara, Japan

Keichi Takahashi
Tohoku University
Sendai, Japan

Yutaro Kashiwa
Nara Institute of Science and Technology
Nara, Japan

Kohei Ichikawa
Nara Institute of Science and Technology
Nara, Japan

Hajimu Iida
Nara Institute of Science and Technology
Nara, Japan

1 INTRODUCTION

Modern computer hardware requires parallel programming to fully exploit its computational performance. Parallel programming demands deep knowledge of both hardware and software to bridge the two. To reduce the need for learning parallel programming, researchers have been developing various tools including automatic parallelization tools. These tools typically employ static analysis to identify loop patterns in source codes and transform parallelizable loops. However, codes that are manually parallelized often outperform those that are parallelized by automatic tools.

The newly emerged deep learning-based Natural Language Processing (NLP) models have set new benchmarks in NLP tasks, surpassing the average human score for the first time. This model, named transformer, is implemented with an attention mechanism as one of the layers, enabling developers to pre-train language understanding models to solve downstream tasks without the need for labeled data. Pre-training allows large sections of unlabeled articles, publications, and even conversations from various sources to be used as training data. The enormous success of transformer-based language models has inspired software engineering researchers to train language models on computer programming languages. The performance of these models on downstream software engineering tasks including code completion and code translation significantly exceeds that of previous studies.

2 PROPOSAL

We propose building a generative model for OpenMP directives trained on source code from public GitHub repositories utilizing CodeT5 / CodeT5+ [3], a transformer-based Large Language Model (LLM) designed for code understanding and generation. For this purpose, we collected 57,170 OpenMP-parallelized for-loops from GitHub. Since training LLMs is computationally demanding, we employ models pre-trained on C and C++ source codes, and use our collected dataset for fine-tuning. Figure 1 illustrates the two-part structure of our automatic parallelization approach, employing the LLMs. The first part, derived from our previous study, is the parallel for-loop classification model [2], which identifies parallelizable for-loops in the given source code. The main contribution of this study is the second part, which focuses on the OpenMP directive generation model.

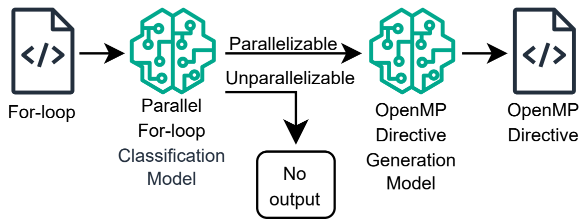
For-loop
Parallel For-loop Classification Model
Parallelizable
Unparallelizable
No output
OpenMP Directive Generation Model
OpenMP Directive

Figure 1: An automatic parallelization approach using LLM

3 EVALUATION

The effectiveness of this study is measured by the performance improvement of the source code modified by the tools. Various benchmarks providing both serial and OpenMP source codes, such as NAS Parallel Benchmarks, UK Mini-App Consortium Applications, and Mantevo Proxy Applications, will be used for this measurement. Benchmarks are utilized for their convenience, facilitating straightforward comparisons of runtime reductions between each approach and the original serial source code.

Clava is selected as the baseline model for the static analysis approach. This tool is recognized as a state-of-the-art C/C++ source-to-source compiler. One of the functions in Clava, Autopar [1], leverages static analysis to identify the parallelizable for-loops and annotates them with OpenMP directives. Although Autopar-Clava demonstrates exceptional results, it does not surpass the manual annotations by experts.

REFERENCES

[1] Hamid Arabnejad, João Bispo, Jorge G. Barbosa, and João M. P. Cardoso. 2018. AutoPar-Clava: An Automatic Parallelization source-to-source tool for C code applications. In *Proceedings of the 9th Workshop on Parallel Programming and RunTime Management Techniques for Manycore Architectures and 7th Workshop on Design Tools and Architectures for Multicore Embedded Computing Platforms*. 13–19.

[2] Soratouch Pornmaneerattanatri, Keichi Takahashi, Yutaro Kashiwa, Kohei Ichikawa, and Hajimu Iida. 2024. Parallelizable Loop Detection using Pre-trained Transformer Models for Code Understanding. In *Parallel and Distributed Computing, Applications and Technologies*. Springer Nature Singapore, 32–42.

[3] Yue Wang, Hung Le, Akhilesh Deepak Gotmare, Nghi D. Q. Bui, Junnan Li, and Steven C. H. Hoi. 2023. CodeT5+: Open Code Large Language Models for Code Understanding and Generation. *CoRR* abs/2305.07922 (2023).