UPIR: Toward the Design of Unified Parallel Intermediate Representation for Parallel Programming Models

Anjia Wang
awang15@uncc.edu
University of North Carolina at Charlotte
Charlotte, North Carolina, USA

Xinyao Yi
xyi2@uncc.edu
University of North Carolina at Charlotte
Charlotte, North Carolina, USA

Yonghong Yan
yyan7@uncc.edu
University of North Carolina at Charlotte
Charlotte, North Carolina, USA

ABSTRACT
The complexity of heterogeneous computing architectures, as well as the demand for productive and portable parallel application development, have driven the evolution of parallel programming models to become more comprehensive and complex than before. Enhancing the conventional compilation technologies and software infrastructure to be parallelism-aware has become one of the main goals of recent compiler development. In this work, we propose the design of unified parallel intermediate representation (UIPR) for multiple parallel programming models and for enabling unified compiler transformation for the models. UIPR specifies three commonly used parallelism patterns (SPMD, data and task parallelism), data attributes and explicit data movement and memory management, and synchronization operations used in parallel programming. We demonstrate UIPR via a prototype implementation in the ROSE compiler for unifying IR for both OpenMP and OpenACC and in both C/C++ and Fortran, for unifying the transformation that lowers both OpenMP and OpenACC code to LLVM runtime, and for exporting UIPR to LLVM MLIR dialect. The fully extended paper of this abstract can be found from https://arxiv.org/abs/2209.10643.

CCS CONCEPTS
• Software and its engineering → Compilers.

KEYWORDS
Compiler transformation, Parallel intermediate representation, OpenMP, OpenACC, MLIR

ACM Reference Format:
Anjia Wang, Xinyao Yi, and Yonghong Yan. 2022. UIPR: Toward the Design of Unified Parallel Intermediate Representation for Parallel Programming Models. In International Conference on Parallel Architectures and Compilation Techniques (PACT ’22), October 8–12, 2022, Chicago, IL, USA. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3559009.3569646

1 INTRODUCTION
The past two decades have seen dramatically increased complexity of computer systems, including the significant increase of parallelism from 10s to 100s and 1000s computing units and cores, the wide adoption of heterogeneous architecture such as CPU, GPUs and vector units in a computer system, and the significant enhancement to the conventional memory hierarchy using new memory technologies such as 3D-stacked memory and NVRAM. Demands from users and applications for computing have also become high and diverse, ranging from computational science, large-scale data analysis, and artificial intelligence that adopts the computation-intensive deep neural network methods. Together they have driven the evolution of parallel programming models to become more comprehensive and complex with multifaceted goals including delivering portable performance across diverse architectures, being highly expressive for the wide ranges of users and applications, and allowing for high performance implementation and tools support. It is observed that existing parallel programming models share common parallelism functionality and use similar interfaces of essential capability for programming parallelism [2]. However, supporting these parallel models in one compiler often has to create language-dependent compiler passes of the same functionality for different models. We believe one of the barriers is the lack of language-independent abstraction of the fundamental entities and constructs for parallelism. This has hindered the research and development of parallelism-aware analysis and transformation across multiple programming models.

In this work, we propose the notion and specification of unified parallel intermediate representation (UIPR) to enable language-neutral parallelism-aware compilation. We create a prototype implementation in ROSE compiler, and demonstrate UIPR for unifying IR for offloading code in both OpenMP and OpenACC and in both C/C++ and Fortran. The demonstration includes a unified transformation that lowers OpenMP and OpenACC code to LLVM OpenMP runtime. UIPR is also implemented as LLVM MLIR dialect, thus the ROSE-based UIPR compiler is able to export the UIPR of a program to its MLIR dialect.

2 UNIFIED PARALLEL INTERMEDIATE REPRESENTATION (UIPR)
As existing parallel programming models share common parallelism functionality and similar interfaces of essential capability for programming parallelism [2], a language-independent abstraction of the fundamental entities and constructs for parallelism and their connections can be constructed in a unified intermediate representation serving as the backbone to enable unified and common parallelism-aware analysis and transformation. The UIPR design and specification include 1) the three commonly used parallelism patterns, namely single program multiple data (SPMD), data parallelism, and task parallelism including offloading tasks; 2) data
attributes and explicit data movement and memory management for assisting data-aware optimization for parallel programs; and 3) synchronization operations used in parallel programming for optimizing synchronization cost by the compiler. Table 1 shows the UPIR’s support and mapping for the language constructs of OpenMP and OpenACC.

| SPMD parallelism | UPIR  | OpenMP | OpenACC |
|------------------|-------|--------|---------|
| loop              | spmd  | teams, parallel | parallel |
| data parallelism  | loop  | distribute, for, simd | loop, gang 
| Async task        | task  | task, taskwait | async, wait |
| Data attributes   | data  | shared, private | data(copyin/out) 
| Synchronization   | sync  | barrier, atomic, critical | wait, atomic |

Table 1: Mapping of parallel programming model constructs with UPIR design

3 EVALUATION

Our prototype is implemented in ROSE source-to-source compiler [1]. Figure 1 shows how UPIR is generated from OpenMP and OpenACC source code, in both C and Fortran, and followed by a unified transformation. The UPIR is also implemented with LLVM TableGen to produce the UPIR dialects in MLIR, allowing the ROSE compiler support C/C++/Fortran, OpenMP, and OpenACC. Our experiments show that the UPIR compiler utilizes the unified transformation to compile both OpenMP and OpenACC programs. UPIR enables a unified compiler transformation for multiple parallel programming models.

Our experiments show that the UPIR compiler utilizes the unified transformation to compile both OpenMP and OpenACC programs. UPIR helps compilers conduct a unified transformation for multiple parallel programming models. We pick two offloading kernels for evaluation: AXPY and 2D stencil. The evaluation compared our prototype implementation of the UPIR to be exported to MLIR (Figure 2).

4 CONCLUSION

In this work, we present UPIR, a unified parallel intermediate representation used for representing parallelism of parallel programming models to assist parallelism-aware compiler analysis, transformation, and optimization. It is designed to support a wide-variety of parallel programming models and the prototype implementation in ROSE compiler support C/C++/Fortran, OpenMP, and OpenACC. UPIR enables a unified compiler transformation for multiple parallel programming models.

Our experiments show that the UPIR compiler utilizes the unified transformation to compile both OpenMP and OpenACC programs. UPIR helps compilers conduct a unified transformation for multiple parallel programming models. We pick two offloading kernels for evaluation: AXPY and 2D stencil. The evaluation compared our prototype implementation of the UPIR to be exported to MLIR (Figure 2).

3 EVALUATION

Figure 1: UPIR implementation in ROSE compiler to support C/C++/Fortran and OpenMP and OpenACC

Figure 2: AXPY in UPIR MLIR dialect, for OpenMP and OpenACC GPU Offloading

Figure 3: 2D stencil performance of UPIR compiler, LLVM, NVIDIA, and GCC compilers

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 1833332 and 2015254.