An Environment for Portable High Productivity High Performance Computing on GPUs/Accelerators

PIs: P. Sadayappan (The Ohio State University) and J. Ramanujam (Louisiana State University)

The emergence of multi-core processors as the computing engine in all commodity platforms presents an enormous software development crisis. For over two decades, sequential software applications have enjoyed the "free ride" of performance improvement with each new processor generation. However, due to insurmountable heat dissipation challenges, chip manufacturers were forced to abandon clock rate increases, and to focus on multi-core architectures. The reality today is that existing and new applications must be changed if they are to experience any performance benefits from newer generations of processors.

The emergence of GPUs as cost-effective and powerful processors for general-purpose computing has raised interest in their use for many scientific and engineering applications but further exacerbates the software development challenges. Although the drafting of the OpenCL standard is is a very promising development towards portable programming of GPUs and accelerators, developing high-performance applications in OpenCL is even more complicated than programming general-purpose multicore processors using programming models like OpenMP.

In this project, we are developing a programming environment for easing the development of portable high-performance applications for GPUs and accelerators – by automatic generation of OpenCL code from annotated C programs provided by the user. The proposed work is motivated by recent advances in *polyhedral-based approaches* for powerful transformations of affine computations that have enabled the development of the Pluto automatic parallelization/optimization system.

Approach. The Pluto polyhedral optimization system will serve as a powerful core engine for a compiler-driven auto-transformation system for GPUs/accelerators. In this project, we are pursuing a number of key technical advances that will lead to a practical and highly-advanced Pluto-based OpenCL transformer. Our work addresses the following major topics:

- Effective Multi-level Tiling: Transforming loop nests from standard sequential C to OpenCL is essentially an exercise in effective multi-level tiling with multi-level parallelism. We are developing effective algorithms for multi-level tiling and parallelization.
- Exploiting SIMD instruction sets: We will implement an effective vectorization capability (based on the polyhedral framework, unlike current vendor compilers) to exploit the performance gains from use of SIMD instructions.
- Scratchpad Memory Management: Effective use of on-chip scratchpad memories is critical for achieving high-performance with GPUs/Accelerators. We are developing effective strategies for automatic movement of data between global memory and local memory in the generated OpenCL code.
- Optimizing Irregular Computations: Complementing the polyhedral transformation framework for affine computations, we are developing data locality optimization strategies based on the inspector/executor paradigm. One aspect of this is *optimizing recurrent access patterns.* The overheads with an inspector/executor paradigm can be greatly reduced if the inspector code can be hoisted outside loops within which access patterns repeat. We are developing effective approaches for compile-time identification of loops with recurrent access patterns.