Interface reconstruction algorithms in 2D and 3D for many-core and multi-core computer architectures

Marianne M. Francois[†], **Li-Ta Lo**[‡], **Christopher Sewell**[‡]

[†] T-3: Fluid Dynamics and Solid Mechanics, Los Alamos National Laboratory
(mmfran@lanl.gov)

[‡] CCS-7: Applied Computer Science, Data Science at Scale Team, Los Alamos National Laboratory

ABSTRACT

With the increasing heterogeneity of high-performance computing hardware and the multitude of vendor-specific hardware, a major challenge to computational physicists is to work in close collaboration with computer scientists to develop portable and efficient algorithms and software on these novel computer architectures. In this work, we have written a single portable and efficient code, named PINION, using NVIDIA's Thrust library to perform interface reconstruction in two-dimensions and three-dimensions on structured Cartesian meshes. Interface reconstruction is a technique commonly used in volume tracking methods for simulations of multi-material flows. The Thrust library provides a high-level interface to program on GPUs as well as multi-core CPUs because it supports CUDA, OpenMP, and Intel Threading Building Blocks. However, the Thrust library has a simplistic data model and only employs one-dimensional vectors, making it challenging to perform multi-dimensional physics-based simulations. Therefore, we have designed two-dimensional and three-dimensional mesh data structures that are mapped to the one-dimensional vectors used by Thrust. Since our mesh data structures present an interface using familiar terminology (such as cells, vertices, and edges), they are simple to use in physics algorithms. With these new data structures in place, we have implemented a recursive volume-of-fluid initialization algorithm and standard piecewise interface reconstruction algorithms in two-dimensions and in three-dimensions, and have tested them on multiple architectures. Our work provides a single implementation of these algorithms that can be compiled to multiple backends (including multi-core CPUs, NVIDIA GPUs, and Intel Xeon Phi coprocessors), making efficient use of the available parallelism on each. Performance results will be shown to illustrate the utility of this approach.