Towards an Abstraction-Friendly Programming Model for High Productivity and High Performance Computing

Chunhua Liao, Daniel J. Quinlan and Thomas Panas

Lawrence Livermore National Laboratory
Livermore, CA 94551
{liao6,quinlan1,panas2}@llnl.gov

General purpose languages, such as C++, permit the construction of various high level abstractions to hide redundant, low level details and accelerate programming productivity. Example abstractions include functions, data structures, classes, templates and so on. However, the use of abstractions significantly impedes static code analyses and optimizations, including parallelization, applied to the abstractions' complex implementations. As a result, there is a common perception that performance is inversely proportional to the level of abstraction. On the other hand, programming large scale, possibly heterogeneous high-performance computing systems is notoriously difficult and programmers are less likely to abandon the help from high level abstractions when solving real-world, complex problems. Therefore, the need for programming models balancing both programming productivity and execution performance has reached a new level of criticality.

We are exploring a novel abstraction-friendly programming model in order to support high productivity and high performance computing. We believe that standard or domain-specific semantics associated with high level abstractions can be exploited to aid compiler analyses and optimizations, thus helping achieving high performance without losing high productivity. We encode representative abstractions and their useful semantics into an abstraction specification file. In the meantime, an accessible, source-to-source compiler infrastructure (the ROSE compiler) is used to facilitate recognizing high level abstractions and utilizing their semantics for more optimization opportunities. Our initial work has shown that recognizing abstractions and knowing their semantics within a compiler can dramatically extend the applicability of existing optimizations, including automatic parallelization. Moreover, a new set of optimizations have become possible within an abstraction-friendly and semantics-aware programming model.

In the future, we will apply our programming model to more large scale applications. In particular, we plan to classify and formalize more high level abstractions and semantics which are relevant to high performance computing. We will also investigate better ways to allow language designers, library developers and programmers to communicate abstraction and semantics information with each other.