Compiler Optimization for Parallel Recursive Traversals of K-d Trees

Samyam Rajbhandari[1] Jinsung Kim[1]
Sriram Krishnamoorthy[2] Louis-Noël Pouchet[1] Fabrice Rastello[3]
Robert J. Harrison[4] P. Sadayappan[1]

[1] Ohio State University, USA [2] Pacific Northwest National Laboratory, USA [3] INRIA, France [4] Stony Brook University, USA

Abstract

There has been considerable work towards implementing loop transformations for enhancing data locality, including loop fusion, permutation, tiling, etc., and many loop transformations are implemented in the optimization passes of production compilers such as GNU gcc, IBM xlc, Intel icc, Nvidia nvcc etc. However, there has been relatively little prior research on compiler-directed data locality optimization for recursive programs. Our work is motivated by the need for compiler optimization to enhance the performance of the production scientific application framework MADNESS (Multiresolution Adaptive Numerical Environment for Scientific Simulation), an environment for the solution of integral and differential equations in many dimensions.

A MADNESS user writes a program using high-level operators on functions over space (i.e., these functions are the "variables" of the MADNESS program). Examples of MADNESS operators on functions are addition, multiplication, convolution, and differentiation. Functions over space (the MADNESS programs variables) are internally represented using k-d trees, refined to a tree depth based on the desired numerical precision for the computation. The implementation of MADNESS operators has a recursive specification over the structure of the k-d tree representation of the produced result variable, suitably traversing the k-d tree representations of the input operand variables. The k-d trees representing MADNESS variables are typically much larger than cache. Therefore, the execution of each MADNESS operator generally requires a considerable amount of data movement across nodes and within the memory/cache hierarchy of nodes on a distributed-memory cluster. In the current MADNESS implementation, the recursive functions corresponding to the operators are executed in a nested fork-join fashion, incurring significant overhead. We develop domain-specific compiler support to optimize parallel execution of MADNESS programs.

Loop fusion is a key program transformation for data locality optimization that is implemented in production compilers. But optimizing compilers for imperative languages currently cannot exploit fusion opportunities across a set of recursive tree traversal computations with producer-consumer relationships. We develop a compile-time approach to dependence characterization and program transformation to enable fusion across recursively specified traversals over k-d trees. We present a source-to-source code transformation framework to automatically generate fused composite recursive operators from an input program containing a sequence of primitive recursive operators. We use the new compiler framework to implement fused operators in MADNESS, demonstrating significant performance improvement.