SI2-SSI: Accelerating the Pace of Research through Implicitly Parallel Programming

David I. August
august@cs.princeton.edu
and
David Walker
dpw@cs.princeton.edu

Today, two trends conspire to slow down the pace of science, engineering and academic progress in general. First, an increasing number of disciplines rely on computation to process massive data sets and to perform computationally intensive simulations. Second, processor speeds are no longer increasing so performance-critical programs need to exploit parallelism to harness computing trends. Unfortunately, doing so with today’s software-development tools takes advanced training in computer science and even with such training is extremely difficult, error-prone and time consuming.

Our work brings the performance promises of multicore architectures to scientists, engineers and everyone else who needs them. More specifically, after years of experience with programming languages and compilers research, and after working with numerous scientists on real applications, the PIs believe that a new approach, the Implicitly Parallel Programming Paradigm (IPP), will be able to meet the needs of scientists and engineers in a cost-effective manner. Unlike other proposed solutions to the parallelism problem, IPP is domain-independent, performance portable and highly efficient. Most importantly, it will be easy-to-use by everyday scientists who are not specially trained in parallel programming.

IPP allows users to write ordinary sequential programs and then to augment them with semantic specifications that expand (or abstract) the set of sequential program behaviors. This capacity for abstraction will provide parallelizing compilers with the flexibility to more aggressively optimize programs than would otherwise be possible. In fact, it will enable effective parallelization techniques where they were impossible before. The language design and compiler implementation will be accompanied by formal semantic analysis that will be used to judge the correctness of compiler transformations, provide a foundation for about reasoning programs, and guide the creation of static analysis and defect detection algorithms. Moreover since existing programs and languages can be viewed as (degenerately) implicitly parallel, decades of investment in human expertise, languages, compilers, methods, tools, and applications is preserved. In particular, it will be possible to upgrade old legacy programs or libraries from slow sequential versions without overhauling the entire system architecture, but merely by adding a few auxiliary specifications. Conceptually, IPP restores an important layer of abstraction, freeing programmers to write high-level code, designed to be easy to understand, rather than low-level code, architected according to the specific demands of a particular parallel machine.

In work by the PIs, IPP appears to represent a breakthrough. Using semantic specifications consisting of 60 lines of code for the 500,000 lines composing SPEC CINT2000, a suite of notoriously sequential programs, IPP methods, applied manually and systematically to approximate the result of the proposed work, obtained a geometric mean speedup of 5.5x on 32 processors, exceeding the historic performance expectation of 5.2x for the 32-core processor generation. Preliminary field work also showed that a programmer using IPP could, over the course of one day, create an executable that outperforms those resulting from person-years of effort by domain-expert scientists.

How To Engage Us: We are providing direct and immediate benefit to researchers in several areas of science by working with researchers in other areas to study, analyze and develop high-value scientific applications. Optimized code from our work has already been delivered to researchers, improving the rate at which they can study scientific questions. In particular, a GPU parallelization of a computational biology program named KNNImpute and a multi-core parallelization of a computational physics program named packing have been provided to the original users of these applications. Our compiler infrastructure operates over ANSI C/C++ and we would be interested in discovering new, important, moderately sized applications within emerging scientific disciplines. If you are eager to work with us, please do not hesitate to email the PIs.