Challenges in Code Optimization of Parallel Programs

Vivek Sarkar

Rice University

Abstract. Code optimization has a rich history that dates back over half a century, and includes deep innovations that arose in response to changing trends in hardware and programming languages. These innovations have contributed significantly to programmer productivity by reducing the effort that programmers spend on hand-implementing code optimizations and by enabling code to be more portable. Often these innovations were accompanied by *paradigm shifts* in the foundations of compilers led by the introduction of new ideas such as interprocedural whole program analysis, coloring-based register allocation, static single assignment form, array dependence analysis, pointer alias analysis, loop transformations, adaptive profile-directed optimizations, and dynamic compilation.

In this talk, we claim that the current multicore trend in the computer industry is forcing a new paradigm shift in compilers to address the challenge of *code optimization of parallel programs*, regardless of whether the parallelism is implicit or explicit in the programming model. All computers — embedded, mainstream, and high-end — are now being built from multicore processors with little or no increase in clock speed per core. This trend poses multiple challenges for compilers for future systems as the number of cores per socket continues to grow, and the cores become more heterogeneous. In addition, compilers have to keep pace with a proliferation of new parallel languages and libraries.

To substantiate our claim, we first highlight some of the anomalies that arise when classical techniques from sequential code optimization are applied to parallel code. We then examine the historical foundations of code optimization including intermediate representations (IR's), abstract execution models, legality and cost analyses of IR transformations and identify paradigm shifts that will be necessary to support optimization of parallel code. We pay special attention to memory consistency models and their impact on code optimization. Finally, we summarize the approach to code optimization of parallel programs being taken in the Habanero Multicore Software Research project at Rice University.

O. de Moor and M. Schwartzbach (Eds.): CC 2009, LNCS 5501, p. 1, 2009.