Load Elimination in the Presence of Side Effects, Concurrency and Precise Exceptions

Christoph von Praun Florian Schneider Thomas Gross

To appear at the *16th Workshop on Languages and Compilers for Parallel Computing (LCPC03)*, College Station, TX, 2-4 October 2003

Abstract

Partial redundancy elimination can reduce the number of loads corresponding to field and array accesses in Java programs. The reuse of values loaded from memory at subsequent occurrences of load expressions must be done with care: Precise exceptions and the potential of side effects through method invocations and concurrent modifications in multi-threaded programs must be considered.

This work focuses on the effect of concurrency on the load optimization. Unlike previous approaches, our system determines accurate information about side effects and concurrency through a whole-program analysis. Partial redundancy elimination is extended to exploit this information and to broaden the optimization scope.

There are three main results: (1) Load elimination is effective even in the most conservative variant without side effect and concurrency analysis (avg. dynamic reduction of loads 21.1%, max. 55.6%). (2) Accurate side effect information can significantly increase the number of optimized expressions (avg. dynamic reduction of loads 26.4%, max. 66.1%). (3) Information about concurrency can make the optimization independent of the memory model, enables aggressive optimization across synchronization statements, and improves the number of optimization opportunities compared to an uninformed optimizer that is guided by a (weak) memory model (avg. dynamic reduction of loads 30.1%, max. 70.3%).