Aggressive Loop Fusion for Improving Locality and Parallelism

Jingling Xue

Programming Languages and Compilers Group,
School of Computer Science and Engineering,
University of New South Wales,
Sydney, NSW 2052, Australia

Abstract. Existing loop fusion algorithms fuse loop nests only when the dependences in the loop nests are not violated. This paper presents a new algorithm that is capable of fusing loop nests in the presence of fusion-preventing anti-dependences. We eliminate all these violated dependences by automatic array copying. In this work, such an aggressive loop fusion strategy is applied to a Jacobi program. The performance of such iterative methods is typically limited by the speed of the memory system. Fusing the two loop nests in the Jacobi program into one reduces data cache misses, and consequently, improves the performance results of both sequential and parallel versions of the Jacobi program, as validated by our experimental results on an HP AlphaServer SC45 supercomputer.

1 Introduction

Due to the increasing performance mismatch between processors and main memories, modern computer systems are equipped with increasingly more levels of caches (e.g., three levels in the Intel IA-64 processors) to prevent performance degradation. However, caches help speed up only those programs that exhibit good data locality. For programs that do not reuse data, their execution times are limited by the poor latency and bandwidth values of the main memory. Therefore, cache-conscious programs are important for CPU-intensive applications, where the most computations are carried out inside loop nests.

There has been a great deal of work on the exploitation of cache locality for performance enhancement. For example, the design of LAPACK is influenced by efficiency considerations in the presence of caches. The main motivation of LAPACK was to recast the algorithms in EISPACK and LINPACK into blocked versions in terms of calls to BLAS [1]. In parallel with the development of LAPACK, compiler researchers have successfully automated many of the loop transformations, such as loop tiling or blocking [9][11][12] (for generating blocked algorithms), loop fusion and loop distribution [10], used in LAPACK in a compiler.

However, one fundamental limitation of existing loop transformations is that they are dependence-preserving and thus inapplicable when the data dependences in the program are violated. In [13], we introduced a new loop fusion
Aggressive Loop Fusion for Improving Locality and Parallelism

A compiler algorithm that allows arbitrary loop nests with affine loop bounds and array subscript expressions to be fused. In the fused program, all fusion-preventing flow (i.e., true) and output dependences are eliminated by loop tiling and all fusion-preventing anti-dependences by automatic array copying. Such an aggressive loop fusion strategy has two important benefits. First, by fusing the two loop nests that cannot be fused conventionally, we are able to exploit the data reuse across the two loop nests. Second, by creating perfect loop nests that cannot be obtained conventionally, we are able to exploit the data reuse within perfect loop nests by further applying loop tiling to these perfect nests. In [13], we demonstrated that our aggressive loop fusion can improve program performance significantly on uniprocessors with cache memories. In this paper, we show that our aggressive loop fusion can also improve the performance of parallel applications running on multi-processor computer systems. Our example is an MPI program that uses the Jacobi method to solve the Helmholtz equation. Iterative solvers for partial differential equations (PDEs) such as Jacobi are typically implemented using global sweeps over the whole data set. As a result, their performance is limited by the speed of the memory system. Improving the cache performance of iterative solvers is absolutely essential to achieving good performance for these solvers on modern computer systems. We report and analyse the performance results of our Jacobi application before and after loop fusion is applied. The fused program yields improved performance due to improved data locality and also slightly reduced message communication cost.

Like Gauss-Seidel and SOR (Successive Over-Relaxation) methods, Jacobi is a classic iterative solver for PDEs. These solvers are still important today because they are useful either as models for more complex methods or as building blocks from which more advanced methods, such as multigrid, can be constructed. This paper is not concerned with designing fast iterative solvers. Instead, the thesis of this work is that an aggressive loop fusion strategy can improve the performance of parallel applications for which the existing loop fusion is inapplicable. One future work is to apply our technique to multigrid methods.

The rest of this paper is organised as follows. Section 2 introduces an algorithm that fuses loop nests in the presence of violated anti-dependences. In Section 3, we apply this algorithm to transform a Jacobi program consisting of two loop nests into one perfect loop nest. Section 4 presents and analyses our experimental results on uniprocessor and multi-processor systems. Section 5 compares with the related work. Section 6 concludes the paper.

2 An Aggressive Loop Fusion Algorithm

We consider array-dominated programs consisting of multiple loop nests whose loop bounds and array subscript expressions are affine expressions of the surrounding loop variables. The fusion of two perfect loop nests is legal iff all dependences from the first (i.e., the lexically earlier) nest to the second nest are not reversed in the fused program [10, p. 315]. The dependences that are reversed are known as the fusion-preventing dependences. There are three kinds of fusion-