Eliminating Conflict Misses for Tiled Codes

Gabriel Rivera, Chau-Wen Tseng

Department of Computer Science
University of Maryland
College Park, MD 20742

Tiling is a powerful compiler technique for exploiting data locality in scientific codes. However, previous research has shown conflict misses occurring due to caches with limited associativity can significantly degrade the performance of tiled codes. Two approaches for avoiding conflict misses are 1) carefully picking tile sizes, and 2) copying tiles to contiguous buffers at run time. In this research, we improve the flexibility and performance of existing approaches through intra-variable padding, as well as provide extensive experimental evaluation using a matrix multiply case study.

Previous techniques for avoiding conflict misses include picking the largest non-conflicting square tile [1] and some rectangular tile generated using the Euclidean GCD algorithm [2]. By varying the problem size of matrix multiply, we discover many sizes where both algorithms are forced to select tiles small relative to the cache size. As a result, the amount of loop overhead degraded the performance of the tiled code, sometimes by 20% or more on a Sun UltraSparc. We provide a simple padding algorithm for solving this problem.

Our padding heuristic simply examines tiles selected when the target array column is padded from 0 to 7 elements. The pad producing the "best" tile size is chosen. If possible, array declarations are modified. Otherwise the array is first precopied to a padded array.

We experimentally evaluate the performance of matrix multiply for problem sizes between 100 to 400 for different versions of tiling on a Sun UltraSparc, using both timings and cache simulations. Results show padding can significantly improve the performance of tiling, avoiding pathological array sizes which force small tiles. The overhead of precopying the added array is under 3% of the execution time, particularly for larger problem sizes. Experiments also show that copying tiles to contiguous buffers is quite effective, and larger tiles should be selected than proposed in previous papers. Overall, we find padding used in conjunction with tile size selection to be a useful compiler transformation for eliminating conflicts in tiled codes.

References

[1] S. Coleman and K. S. McKinley. Tile size selection using cache organization and data layout. In *Proceedings of the SIGPLAN '95 Conference on Programming Language Design and Implementation*, La Jolla, CA, June 1995.

[2] M. Lam, E. Rothberg, and M. E. Wolf. The cache performance and optimizations of blocked algorithms. In *Proceedings of the Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IV)*, Santa Clara, CA, April 1991.