Chapter 9. Optimal Tiling for Minimizing Communication in Distributed Shared-Memory Multiprocessors

Anant Agarwal\textsuperscript{1}, David Kranz\textsuperscript{1}, Rajeev Barua\textsuperscript{2}, and Venkat Natarajan\textsuperscript{3}

\textsuperscript{1} Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139
\textsuperscript{2} Department of Electrical & Computer Engineering, University of Maryland, College Park, MD 20742, barua@eng.umd.edu
\textsuperscript{3} Wireless Systems Center, Semiconductor Products Sector Motorola

Summary. This paper presents a theoretical framework for automatically partitioning parallel loops and data arrays for cache-coherent NUMA multiprocessors to minimize both cache coherency traffic and remote memory references. While several previous papers have looked at hyperplane partitioning of iteration spaces to reduce communication traffic, the problem of deriving the optimal tiling parameters for minimal communication in loops with general affine index expressions has remained open. Our paper solves this open problem by presenting a method for deriving an optimal hyperparallelepiped tiling of iteration spaces for minimal communication in multiprocessors with caches. Our framework uses matrices to represent iteration and data space mappings and the notion of uniformly intersecting references to capture temporal locality in array references. We introduce the notion of data footprints to estimate the communication traffic between processors and use linear algebraic methods and lattice theory to compute precisely the size of data footprints. We show that the same theoretical framework can also be used to determine optimal tiling parameters for both data and loop partitioning in distributed memory multicomputers. We also present a heuristic for combined partitioning of loops and data arrays to maximize the probability that references hit in the cache, and to maximize the probability cache misses are satisfied by the local memory. We have implemented this framework in a compiler for Alewife, a distributed shared memory multiprocessor.

1. Introduction

Cache-based multiprocessors are attractive because they seem to allow the programmer to ignore the issues of data partitioning and placement. Because caches dynamically copy data close to where it is needed, repeat references to the same piece of data do not require communication over the network, and hence reduce the need for careful data layout. However, the performance of cache-coherent systems is heavily predicated on the degree of temporal locality in the access patterns of the processor. Loop partitioning for cache-coherent multiprocessors is an effort to increase the percentage of references that hit in the cache.

The degree of reuse of data, or conversely, the volume of communication of data, depends both on the algorithm and on the partitioning of work among the processors. (In fact, partitioning of the computation is often considered to be a facet of an algorithm.) For example, it is well known that a matrix...
multiply computation distributed to the processors by square blocks has a much higher degree of reuse than the matrix multiply distributed by rows or columns.

Loop partitioning can be done by the programmer, by the run time system, or by the compiler. Relegating the partitioning task to the programmer defeats the central purpose of building cache-coherent shared-memory systems. While partitioning can be done at run time (for example, see [1,2]), it is hard for the run time system to optimize for cache locality because much of the information required to compute communication patterns is either unavailable at run time or expensive to obtain. Thus compile-time partitioning of parallel loops is important.

This paper focuses on the following problem in the context of cache-coherent multiprocessors. Given a program consisting of parallel do loops (of the form shown in Fig. 2.1 in Sect. 2.1), how do we derive the optimal tile shapes of the iteration-space partitions to minimize the communication traffic between processors. We also indicate how our framework can be used for loop and data partitioning for distributed memory machines, both with and without caches.

1.1 Contributions and Related Work

This paper develops a unified theoretical framework that can be used for loop partitioning in cache-coherent multiprocessors, for loop and data partitioning in multicomputers with local memory, and for loop and data partitioning in cache coherent NUMA multiprocessors. The central contribution of this paper is a method for deriving an optimal hyperparallelepiped tiling of iteration spaces to minimize communication. The tiling specifies both the shape and size of iteration space tiles. Our framework allows the partitioning of doall loops accessing multiple arrays, where the index expressions in array accesses can be any affine function of the indices.

Our analysis uses the notion of uniformly intersecting references to categorize the references within a loop into classes that will yield cache locality. This notion helps specify precisely the set of references that have substantially overlapping data sets. Overlap produces temporal locality in cache accesses. A similar concept of uniformly generated references has been used in earlier work in the context of reuse and iteration space tiling [3,4].

The notion of data footprints is introduced to capture the combined set of data accesses made by references within each uniformly intersecting class. (The term footprint was originally coined by Stone and Thiebaut [5].) Then, an algorithm to compute precisely the total size of the data footprint for a given loop partition is presented. Precisely computing the size of the set of data elements accessed by a loop tile was itself an important and open problem. While general optimization methods can be applied to minimize the size of the data footprint and derive the corresponding loop partitions, we demonstrate several important special cases where the optimization problem