High-Performance Compilers For Parallel Computing
This work covers everything necessary to build a competitive, advanced compiler for parallel or high-performance computers. It starts with a review of basic terms and algorithms such as graphs, trees and matrix algebra. Methods focus on analysis and synthesis, where analysis extracts information from the source program. The various restrictions and problems caused by different languages commonly used in such machines are shown.

This is a modern classic. If you develop compilers for high performance computing (HPC), this must be on your shelves. It has a heavy emphasis on Fortran, the workhorse of HPC, but is widely applicable to other languages as well. The majority of the book’s content reflects the large majority of processors that carry today’s HPC load: sequential, von Neumann engines, even the computing ensemble as a whole has lots of them. This book’s real contribution is in its analysis of the loops that process arrays. Wolfe presents a number of ways to characterize dependencies, using quantitative techniques that go well beyond the graph-based presentations elsewhere. By casting the dependency problem in terms of integer programming or linear algebra, Wolfe make huge bodies of problem-solving knowledge available to the compiler developer. He also uses these quantitative terms to give new insight into loop transformations that the reader may already understand. Despite the irreplaceable value of this book, I found it maddening to read. For some reason, the periods dropped off of nearly every sentence in the text. Then, just when I got used to that quirk, the periods...
appeared again. Their coming and going was so irregular throughout the book that I never really got into a steady reading rhythm. A much worse problem appeared throughout the first chapter, though. It's an exceptional discussion of matrix multiplication, a staple of performance computing. The chapter presents it again and again, to demonstrate differences in looping constructs and the organization of memory access. That part of the discussion was great. The problem is that it's wrong - a systematic error, in seemingly every example, replaced the scalar multiplication at the core of the algorithm with addition.

Download to continue reading...


Dmca