Compiler Optimization For Scientific Applications

CS 491/591 Section 2
Brian T. Smith, UNM, CS Dept.
Spring, 2003

Introduction

• Lecturer
  – Brian T. Smith, Prof., CS Dept.
    • Galles Bldg, Room 2001, 2nd floor of the showroom
      1601 Central Ave. NE (at Pine St.)
    • 277-8249 (receptionist)
    • bsmith@ahpcc.unm.edu, smith@cs.unm.edu
    • CS dept. office  FEC 330A
    • Office hours at CS dept. office
      – Tues/Thurs, 1:45PM-3:30PM, FEC 330A, 277-4565
Textbook And Reference Books

• Textbook:
  Optimizing Compilers For Modern Architectures
  Randy Allen and Ken Kennedy, Morgan-Kaufmann, 2002
  ISBN-1-55860-286-0
  • Errata file for textbook
    – posted on the Handouts section of the web site:
      http://www.ahpcc.unm.edu/~bsmith/CS491_591_OPT/CS491_591_OPT.html

• Reference and related books
  Computer Architectures: A Quantitative Approach
  John Hennessy and David Patterson,

  Modern Processor Design: Fundamentals Of Superscalar Processors
  Beta Edition, John Shen and Mikko Lipasti,

Lectures Topics (From 2nd Edition)

• Chapter 1, Compiler Challenges For High-Performance Architectures
  • What are the concepts of:
    – pipelining, vector operations, superscalar processors, VLIW processors, processor parallelism, memory hierarchy

• Chapter 2, Dependence: Theory And Practice
  • Dependence and its properties
  • Simple dependence testing
  • Parallelization and vectorization
Topics Continued

• Chapter 4, Preliminary Transformations
  • Gaining information needed
  • Loop normalization
  • Data flow analysis
  • Induction-variable exposure

• Chapter 5, Enhancing Fine Grain Parallelism
  • Loop interchange
  • Scalar expansion
  • Scalar and array renaming
  • Node splitting
  • Recognition of reductions
  • Index-set splitting
  • Run-time symbolic resolution
  • Loop skewing

• Chapter 6, Creating Coarse-Grained Parallelism
  • single loop methods
  • perfect loop nests
  • imperfectly nested loops
  • Packaging of parallelism

• Chapter 7, Handling Control Flow
  • if-conversion
  • control dependence
Topics Continued

• If time permits, some topics in:
  – Chapter 9, Managing Cache
    • loop interchange for spatial locality
    • blocking
    • cache management in complex loop nests
    • software prefetching
  – Chapter 8, Improving Register Usage
    • scalar register allocation
    • scalar replacement
    • unroll-and-jam
    • loop interchange and register reuse
    • loop fusion for register reuse
    • complex loop nests

Slides -- Credit

• Most of the slide material is based on slides prepared by Kennedy and Allen
  • I will modify and annotate them to help clarify points
  • On the other hand, I will make precis of their slide material -- there is far too much to cover in the semester with the background that you have
  • This material is also available for you to download directly if you want it (it is very large)
Assignments, Quizzes, And Final Exam

- See the schedule of topics for the dates of midterm quizzes and the final project
- Midterm quizzes
  - in class, after a collection of chapters are finished
  - brief review in the class period before each quiz
  - two quizzes (maybe)
- Final Project
  - to be determined

Homework Assignments and Grading

- Some exercises from each section are assigned
- Completed and handed-in by the beginning of the class listed in the schedule
- Grading:
  - 30% for midterm quizzes
  - 30% for the assigned homework exercises
  - 40% for the final project
What Is Compiler Optimization?

- Block diagram for a compiler:

  - Compilation phases:
    - source-to-source transformations of the intermediate language
    - performs:
      - code preparation for code generation
      - optimization (vectorization, parallelization)
      - nowadays some/many of these phases are machine independent

Allen/Kennedy Thesis

- Conflicts in language design:
  - easy to use and write clear maintainable code
  - versus
    - capable of readily compiled to highly efficient code
      - a language becomes unused when the production codes do not run at near peak performance (what users expect)

- Their compilation experience has been on codes that are production scientific codes
  - efficiency is paramount to remain competitive
Allen/Ken… Thesis Continued

• Their thesis is that:
  • the crux of the issue to create efficient code is using data dependence
  – the original emphasis (long history back to the early 60s) was on loops, vectors, and arrays
    – clearly this was were efficiency improvements paid off
  • was the special emphasis of vector processors
  – now, it is the issue for every current processor
    – superscalar, VLIW, hyperthreaded, multi-processors and multi-computers
    – even important in cache and in machines with memory hierarchies
• Thus, data dependence is a fundamental compiler analysis tool
  • the book/course is a presentation of the theory and practice of data dependence

Dependence-Based Computation

• Vectorization and parallelization require a deeper analysis than optimization for scalar machines
  – Must be able to determine whether two accesses to the same array might be to the same location
• Dependence is the theory that makes this possible
  – There is a dependence between two statements if they might access the same location, there is a path from one to the other, and one access is a write
• Dependence has other applications
  – Memory hierarchy management—restructuring programs to make better use of cache and registers
Then, Why Do We Care?

• To complete the computation faster
  • primarily, reduce the elapsed time of the computation
• To solve problems that could otherwise not be solved because:
  – it would take too long (days, months, even years) -- the weather example
  – the resources are not available in a single processor
    » with more processors, there is more memory, more disk, more …, more of every computational resource
• Areas and examples:
  – weather prediction, information recover and discovery, drug analysis, rapid response to threats, modeling fluid flows, manufacturing process, drug discovery, system modeling avoiding expensive prototypes, etc

Objective Of This Course

• Introduce you to the ideas and concepts of data dependence in programs
• Prepare you for future architectures:
  • They will be parallel at least in terms of the chip
  • They may be parallel in terms of your personal machine (WS)
  • They certainly will be parallel in terms of future computational frontiers
• Because the optimization tools will not be always there:
  • Give you the knowledge of how to write your program so that your programs will run efficiently
  • But balance that with the need to avoid writing architecture-specific optimized code