CS 540 High Performance Computing

Course Description

Covers the design, evaluation and use of high-performance processors, including instruction set architecture, pipelining, superscalar execution, instruction level parallelism, vector instructions, memory hierarchy, parallel computing including multi-core and GPU, and high-performance I/O. Special attention is given to the effective utilization of these features, including automated techniques, in the design and optimization of performance-driven software.

Course Objectives

To develop the skills required to implement high-performance software, including the interaction between algorithms, computer architecture and compilers. To learn techniques for analyzing the performance of programs and their interaction with the underlying hardware. To understand features of modern processors that affect performance and be able to use these features in the design and optimization of high-performance software. To utilize techniques to automatically implement, optimize, and adapt programs to different platforms.

Prerequisites

Students should have had undergraduate courses in data structures, discrete mathematics, linear algebra, and computer architecture. Students should be comfortable using the Unix Programming Environment and be able to read and modify C/C++ code.

Instructor

Constantine Katsinis
Office: 100H University Crossings
Phone: (215) 895-0966
Office Hours: Monday 3-5 pm in UC 100H
e-mail: katsinis@drexel.edu

Meeting Time

Thursday 06:30 pm - 09:20 pm, Rush 014 (and online)

Textbooks

There is no required text. However, we will use material from the books in the list below. Books #1 and #2 are freely available and will be accessible through the class web site.

1. Eijkhout, Introduction to High Performance Scientific Computing, 2e, 2014
2. Severance, High Performance Computing, an open textbook
3. Gerber, The Software Optimization Cookbook: High-Performance Recipes for IA-32 Platforms, 2e, 2008, From Drexel Library
4. Cormen, Leiserson, Rivest, Stein, Introduction to Algorithms, 3e
5. Bryant, O'Hallaron, Computer systems : a programmer's perspective, 2e.
6. Hager, Wellein, Introduction to High Performance Computing for Scientists and Engineers, 2011

Grading

1. Homework and labs 30%
2. Midterm exam 30%
3. Final exam 40%

All assignments must be completed alone unless otherwise stated. No late assignments will be accepted without prior approval.

Lectures

Le	Title	Topics - Readings
1	Introduction Three Divide and Conquer Algorithms.	Topics 1. Introduction 2. Three Problems/Three Algorithms a. Integer multiplication and Karatsuba's algorithm b. Matrix Multiplication and Strassen's algorithm c. Discrete Fourier transform (DFT) and the fast Fourier transform (FFT) Readings 1. Eijkhout, Intro Pages 3 and 4 2. Cormen, Chapter 3, Growth of Functions 3. Cormen, Chapter 4, Section 4, The recursion-tree method for solving recurrences 4. Cormen, Chapter 4, Section 5, The master method for solving recurrences 5. Master-theorem-1.pdf 6. Master-theorem-2.pdf 7. Introduction to the DFT.docx Presentation 1 Lecture-01-Intro-Algorithms Homework HW-1 1. Go through the development of the recursion tree in Cormen Figure 4.5 and explain all the associated analysis on page 90. 2. Do all the examples on Cormen pages 95 and 96. Explain why the theorem does not apply to the case T(n) = 2T(n/2) + nlgn. 3. Verify the result that T(n) = THETA(n^3) in slide 4 of file CSE548-lecture-3.pdf. 4. Work through all the mathematics of the FFT in file JJ-lec1p2.pdf, especially the details on page 3 and the analysis leading to the conclusion that T(N)=THETA(NlogN).
2	From Ops to Instructions.	Topics 1. Computer architecture - Single CPU 2. Assembly language 3. Analysis of assembly code Review 1 Bryant, Chapter 3, Machine-Level Representation of Programs 2 Bryant, Lectures on Machine-Level Programming Readings 1. Eijkhout, Chapter 1, Sequential Computing 2. Hager, Chapter 1, Modern Processors 3. Bryant, Chapter 3, Machine-Level Representation of Programs 4. Gerber, Chapter 5, Processor Architecture 5. Gerber, Chapter 14, Processor-Specific Optimizations 6. Performance Application Programming Interface (PAPI) a. Main: http://icl.cs.utk.edu/papi/ b. Documentation: http://icl.cs.utk.edu/projects/papi/wiki/Main_Page c. Programmer's reference: http://icl.cs.utk.edu/papi/docs/

		Presentation 1 Lecture-02-Machine-Organization Homework HW-2 1. Login to tux, navigate to /proc and review files cpuinfo and meminfo to find out the processor type and memory size. 2. Go to http://www.cpu-world.com/ and locate the tux processor type to obtain more information. 3. Determine the number of instructions executed in loop.c as a function of the array size n. 4. Use makefile to run the unoptimized and optimized versions of the following programs for several values of n and explain the output a. countloop.c b. timeloop.c Homework HW-2-Optional (no need to submit report) 1. Compile loop.c using commands in compile.txt (files test1.c and test2.c are similar and attempt to tell the compiler to use a register for variable i) 2. Explain the compiler output in files *.lst, focusing on the loop body. Program files • compile.txt • countloop.c • loop.c • Makefile • papi-test1.c • test1.c • test2.c • testloop.c • timeloop.c
3	Program Tuning and Optimization	Topics 1. Measuring program performance 2. Optimization blockers 3. Machine Independent Optimizations 4. Machine Dependent Optimizations Review 1 Bryant, Chapter 3, Machine-Level Representation of Programs 2 Bryant, Lectures on Machine-Level Programming 3 x86-64-Machine-Level-Programming 4 x86-Assembly-Language-Reference-Manual Readings 1. Hager, Chapter 2, Basic optimization techniques for serial code 2. Hager, Chapter 3, Data access optimization 3. Bryant, Chapter 5, Optimizing Program Performance 4. Severance, Chapter 2, Programming and Tuning Software Presentation 1 Lecture-03a-Data-Access-Optimization 2 Lecture-03b-25-optimization-CK Homework HW-3- Part-1 (based on Lecture-03a-Data-Access-Optimization) 1. Slide #23 (of 48), Case Study: Matrix Transpose, has the following two statements at the bottom: "+ Strided write more expensive than strided read because of write allocate. + Moving index i to inner loop changes access from strided writes to strided reads (flipped)". Use a diagram similar to the one in slide #12, Column Major Order, to justify these two statements. 2. Slide #42 (of 48), O(N3)/O(N2), has the following statement at the bottom: "Total memory traffic is N(N+1) words." Explain this statement. 3. Slide #46 (of 48), has the following statement in the middle paragraph: "This leads

		to an effective memory traffic of N(N/b+1) words." Explain this statement. Homework HW-3- Part-2 1. Use the programs in the files listed below to perform experiments measuring timing information and cycle counts for different versions of the combine function: combine1(), combine2(), combine3(), combine4(), combine5(), combine6(), combine7(). These functions can be found in file combine.c (possibly with a different name). 2. Use different versions of combine() in file combine.c along with vec.h and vec.c to create programs as illustrated in file co1.c. This file relies on the PAPI library and the functions in file counter.c 3. Explain in detail the operation of the functions in file counter.c 4. Your program, similar to co1.c, should report cycle counts and instruction counts for several values of the vector size used in all functions. Explain in detail why the numbers are different and relate them to the code of each version of the combine() function. 5. Use both integer and floating-point versions Program files • co1.c • vec.c • vec.h • counter.c • counter.h • combine.c • Makefile
4	Program Tuning and Optimization	Continued
5	Optimizing for the Memory Hierarchy. Midterm	Topics 1. Motivation for memory hierarchy 2. Basics of caches 3. Three C model Readings 1. Bryant, Chapter 6, The Memory Hierarchy 2. Severance, Chapter 1, Section 1, Memory Presentation 1 Lecture-05-Cache-Memory Homework HW-5 1. Examine the file cache-hw-nc.c to understand what it does exactly. Add comments in the code and create a separate description file explaining its functions. 2. Compile and run cache-hw-nc.c, record and store all its output, plot it and explain the plots. Program files • cache-hw-nc.c
6	Differential Equations and Linear Algebra	Topics 1. Ordinary and partial differential equations 2. Solving linear systems: direct methods 3. Sparse matrices 4. Numerical Linear Algebra: iterative methods Review 1. Eijkhout, Chapter 1, Sequential Computing 2. Eijkhout, Chapter 3, Computer Arithmetic 3. Severance, Chapter 1, Section 2, Floating-Point Numbers

		Readings 1. Eijkhout, Chapter 4, Numerical treatment of differential equations 2. Eijkhout, Chapter 5, Numerical linear algebra Presentation 1 Lecture-06a-OdePde 2 Lecture-06b-Direct-Methods 3 Lecture-06c-Iterative-Methods
7	Differential Equations and Linear Algebra	Continued Topics 1. Ordinary and partial differential equations 2. Solving linear systems: direct methods 3. Sparse matrices 4. Numerical Linear Algebra: iterative methods
8	Short Vector Instructions (SIMD Computation).	Topics 1. Overview of SSE 2. Example programs Readings 1. Gerber, Chapter 12, SIMD Technology 2. Gerber, Chapter 13, Automatic Vectorization 3. Gerber, Chapter 14, Processor-Specific Optimizations Presentation 1 Lecture-Week8-SIMD Homework HW-8 2 Compile the supplied C programs using this command: 3 gcc -mtune=opteron-sse3 -g -c -fverbose-asm -Wa,-adhln -msse3 -march=x86-64 -ftree-vectorize -ftree-vectorizer-verbose=7 -mfpmath=sse -O2 4 Provide a brief explanation of the use of each compiler option 5 Use the compiler option -ffast-math only if necessary 6 Show the assembly language listing and explain the use of the SSE instructions. If -ffast-math results in different assembly language programs, then show both listings and explain the effect of this option. Program files • test4f.c • test4h.c • test21.c • test9.c • e37.c • ti.c
9	GPUs	Topics 1. Introduction to GPU Architectures for High-Performance Computing Readings Presentation 1 Lecture-09-GPU
10	Holiday	

11		Topics
12	Final	

1 class	09/24/2015
2 class	10/01/2015
3 class	10/08/2015
4 class	10/15/2015
5 class	10/22/2015
6 class+mdt	10/29/2015
7 class	11/05/2015
8 class	11/12/2015
9 class	11/19/2015
10 NO class	11/26/2015
11 class	12/03/2015