

Spring 2008  
Georgia Tech, CSE 6220  
Tu/Th 1:35pm - 2:55pm  
Classroom: KACB Room 2447

## High Performance Computing

<http://www.cc.gatech.edu/~bader/COURSES/GATECH/CSE6220-Spring2008/>

**Instructor:** Prof. David A. Bader, KACB 1332, 404-385-0004, [bader@cc](mailto:bader@cc)

**Office Hours:** Tuesday/Thursday 3:00pm-4:00pm.

**Required Textbooks:**

- *Introduction to Parallel Computing*, Ananth Grama, Anshul Gupta, George Karypis, and Vipin Kumar, 2nd edition, Addison-Welsey, 2003.
- *Petascale Computing: Algorithms and Applications*, David A. Bader (Ed.), Chapman & Hall/CRC Computational Science Series, 2007.

**Course Description:** This course will introduce students to the design, analysis, and implementation, of highperformance computational science and engineering applications. The course focuses on advanced computer architectures, parallel algorithms, parallel languages, and performance-oriented computing. Students will develop knowledge and skills concerning:

- the key factors affecting performance of CSE applications, and
- mapping of applications to high-performance computing systems, and
- hardware/software co-design for achieving performance on real-world applications.

**Pre-requisites:** Proficiency in programming in a high level language such as C, C++, or FORTRAN. Basic knowledge of computer architecture and machine organization is also recommended. Students (from the Sciences, Engineering, and Computing) interested in algorithmic applications in science and engineering are encouraged to take this course.

**Grading:**

(35 %) Midterm  
(35 %) Final Project  
(20 %) Homework  
(10 %) Class participation

## CLASS POLICIES

1. Class announcements will be sent to the Georgia Tech T-Square mailing list, see <http://t-square.gatech.edu/>.
2. Please let me know as soon as possible if you will need to re-schedule an exam, or have any special needs during the semester.
3. Each student must read and abide by the Georgia Tech Academic Honor Code, see [www.honor.gatech.edu](http://www.honor.gatech.edu).
4. Plagiarizing is defined by Websters as “to steal and pass off (the ideas or words of another) as one’s own: use (another’s production) without crediting the source.” If caught plagiarizing, you will be dealt with according to the GT Academic Honor Code.
5. When working on homework, you may work with other students in the class. However; you must turn in separate copies of the homework with the following written on it: your name and the names of everyone with whom you collaborated.
6. Homework is due by the end of lecture on the given due date. Late homeworks will not be accepted without a legitimate excuse and approval from the instructor.
7. No collaboration is permitted on exams. The midterm exam will be in-class, closed-book exams. You will be allowed to take a “cheat sheet” (double-sided 8.5 x 11 sheet of paper) into each exam.
8. Unauthorized use of any previous semester course materials, such as tests, quizzes, homework, projects, and any other coursework, is prohibited in this course. Using these materials will be considered a direct violation of academic policy and will be dealt with according to the GT Academic Honor Code.
9. For any questions involving these or any other Academic Honor Code issues, please consult me, my teaching assistants, or [www.honor.gatech.edu](http://www.honor.gatech.edu).

## Coverage of Topics

The course provides an introduction to advanced computer architectures, parallel algorithms, parallel languages, and performance-oriented computing, and uses real-world case studies from computational science and engineering application domains. After an introduction to computational science and engineering applications, the course consists of four modules. The first focuses on describing the key characteristics of high-end computing architectures. This module provides the student with an appreciation of the underlying hardware systems. The second module covers parallel algorithms with particular attention given to the mapping of algorithms to architectures. The third module introduces the student to a variety of parallel programming languages and environments. Finally, the fourth module is a capstone and ties these concepts together for designing and implementing computational science and engineering applications that achieve superior performance. Each module develops the theoretic framework and complements the study with examples from real-world problems using implementations on modern computing systems. Several homework assignments and a course project will provide the students with an opportunity to write, analyze, and optimize, programs for high-end parallel computer systems.

### Computational Science and Engineering Introduction

1. Computational Science and Engineering Applications; characteristics and requirements
2. Review of Computational Complexity
3. Performance: metrics and measurements
4. Granularity and Partitioning
5. Locality: temporal/spatial/stream/kernel
6. Basic methods for parallel programming
7. Real-world case studies (drawn from multi-scale, multi-discipline applications)

### High-End Computer Systems

1. Memory Hierarchies
2. Multi-core Processors: Homogeneous and Heterogeneous
3. Shared-memory Symmetric Multiprocessors
4. Vector Computers
5. Distributed Memory Computers
6. Supercomputers and Petascale Systems
7. Application Accelerators / Reconfigurable Computing
8. Novel computers: Stream, multithreaded, and purpose-built

### Parallel Algorithms

1. Parallel models: ideal and real frameworks
2. Basic Techniques: Balanced Trees, Pointer Jumping, Divide and Conquer, Partitioning

3. Regular Algorithms: Matrix operations and Linear Algebra
4. Irregular Algorithms: Lists, Trees, Graphs
5. Randomization: Parallel Pseudo-Random Number Generators, Sorting, Monte Carlo techniques

## **Parallel Programming**

1. Revealing concurrency in applications
2. Task and Functional Parallelism
3. Task Scheduling
4. Synchronization Methods
5. Parallel Primitives (collective operations)
6. SPMD Programming (threads, OpenMP, MPI)
7. I/O and File Systems
8. Parallel Matlabs (Parallel Matlab, Star-P, Matlab MPI)
9. Partitioning Global Address Space (PGAS) languages (UPC, Titanium, Global Arrays)

## **Achieving Performance**

1. Measuring performance
2. Identifying performance bottlenecks
3. Restructuring applications for deep memory hierarchies
4. Partitioning applications for heterogeneous resources
5. Using existing libraries, tools, and frameworks

## Spring 2008, Tentative Course Schedule

Week	Date	Lec	Topic	Reading Assn.
1	8 Jan	1	Parallel Computing Introduction	Ch. 1
	10 Jan	2	Guest Lecture: Netezza Corp.	
2	15 Jan	3	Performance Metrics, Granularity, Locality	
	17 Jan	4	Shared Memory and Threads	
3	22 Jan	5	Memory Hierarchies	Ch. 2
	24 Jan	6	Multicore and Manycore	
4	29 Jan	7	Shared memory and symmetric multiprocessors	
	31 Jan	8	Distributed memory and clusters	
5	5 Feb	9	Petascale Computing: IBM and Cray systems	
	7 Feb	10	Application Accelerators: FPGAs, GPUs, Cell	
6	12 Feb	11	Grid Computing ( <i>Guest Lecturer: Z. Du</i> )	
	14 Feb	12	Parallel Algorithms: Ideal and Real Models	Ch. 3
7	19 Feb	13	Basic Algorithmic Techniques	Ch. 4
	21 Feb	14	Regular Algorithms	
8	26 Feb	15	Irregular Algorithms	
	28 Feb	16	<b>Midterm</b>	
9	4 Mar	17	MPI Programming ( <i>Guest Lecturer: Z. Du</i> )	Ch. 6
	6 Mar	18	MPI Programming ( <i>Guest Lecturer: Z. Du</i> )	
10	11 Mar	19	OpenMP Programming	Ch. 7
	13 Mar	20	No class meeting, attend SIAM PP08	
11	18 Mar	-	<b>Spring Break</b>	
	20 Mar	-	<b>Spring Break</b>	
12	25 Mar	21	PGAS Languages: UPC, CoF, Titanium	
	27 Mar	22	Adv. Parallel Programming ( <i>Guest Lecturer: Z. Du</i> )	
13	1 Apr	23	Dense Matrix Algorithms	Ch. 8
	3 Apr	24	Fast Fourier Transform	Ch. 13
14	8 Apr	25	Sparse Matrix Vector Multiplication	
	10 Apr	26	Sparse Graph Algorithms and Multithreaded Arch	
15	15 Apr	27	Project Presentations	
	17 Apr	28	Project Presentations	
16	22 Apr	29	Measuring Performance, Identifying Bottlenecks	
	24 Apr	30	Partitioning Applications for heterogeneous resources	

### **Additional Readings:**

- Grama, A. Gupta, G. Karypis, V. Kumar, *An Introduction to Parallel Computing, Design and Analysis of Algorithms: 2/e*, Addison-Wesley, 2003.
- G.E. Karniadakis, R.M. Kirby II, *Parallel Scientific Computing in C++ and MPI: A Seamless Approach to Parallel Algorithms and their Implementation*, Cambridge University Press, 2003.
- Wilkinson and M. Allen, *Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers, 2/E*, Prentice Hall, 2005.
- M.J. Quinn, *Parallel Programming in C with MPI and OpenMP*, McGraw-Hill, 2004.
- G.S. Almasi and A. Gottlieb, *Highly Parallel Computing, 2/E*, Addison-Wesley, 1994.
- J. Dongarra, I. Foster, G. Fox, W. Gropp, K. Kennedy, L. Torczon, A. White, editors, *The Sourcebook of Parallel Computing*, Morgan Kaufmann, 2002.