First Course Handout

Course Title: Programming for Performance
Course No: CS 610
Credits: 3-0-0-0-[9]

Prerequisite:

- Exposure to CS 210 (Computer Organization), CS 330 (Operating Systems), CS 335 (Compiler Design), and CS 422 (Computer Architecture) (or equivalent for non-IITK courses) is desirable.
- Programming maturity (primarily C/C++/Java) is desirable.

Lecture Hours: WF 10:35-11:50 AM (online, asynchronous)
Discussion Hours: W 10:30-11:30 AM (online, synchronous)

Course Objective: To obtain good performance, one needs to write correct but scalable parallel programs using programming language abstractions like threads. In addition, the developer needs to be aware of and utilize many architecture-specific features like vectorization to extract the full performance potential. In this course, we will discuss programming language abstractions with architecture-aware development to learn to write scalable parallel programs.

This course will involve programming assignments to use the concepts learnt in class and appreciate the challenges in extracting performance.

Course Contents: The course will primarily focus on the following topics.

1. Introduction: Challenges in parallel programming, correctness and performance errors, understanding performance, performance models
2. Exploiting spatial and temporal locality with caches, analytical cache miss analysis
3. Compiler transformations: Dependence analysis, Loop Transformations
4. Shared-memory programming and Pthreads
5. Compiler vectorization: vector ISA, auto-vectorizing compiler, vector intrinsics, assembly
6. OpenMP: Core OpenMP, Advanced OpenMP, Heterogeneous programming with OpenMP
7. Parallel Programming Models and Patterns

8. Intel Threading Building Blocks
9. GPGPU programming: GPU architecture and CUDA Programming
10. Performance bottleneck analysis: PAPI counters, Using performance analysis tools

Optional topics

11. Heterogeneous Programming with OpenMP
12. Fork-Join Parallelism
13. Concurrent data structures
14. Shared-memory synchronization
15. Memory consistency models
16. Transactional memory

Evaluation:

Class participation/quizzes/paper critiques	10%
Assignments	40%
Mid-sem	20%
End-sem	30%

- This is a tentative allocation
    - Might change allocations slightly depending on the strength of the class
- Grading will be relative

References:

1. Optimizing Compilers for Modern Architectures - R. Allen and K. Kennedy
2. Automatic Parallelization: An Overview of Fundamental Compiler Techniques - Samuel P. Midkiff
3. An Introduction to Parallel Programming - Peter S. Pacheco
4. Parallel Computer Architecture: A Hardware/Software Approach - D. Culler, J, Singh with A Gupta
5. Intel Threading Building Blocks: Outfitting C++ for Multi-core Processor Parallelism - J. Reindeers
6. Programming Massively Parallel Processors: A Hands-on Approach - David B. Kirk and Wen-mei W. Hwu

7. The Art of Multiprocessor Programming - Maurice Herlihy and Nir Shavit
8. Introduction to Parallel Computing - A. Grama, A Gupta, G Karypis, and V Kumar

We will also distribute relevant handouts and research papers.