First Course Handout

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

Prerequisite:

- Exposure to CS 220 (Computer Organization) and CS 330 (Operating Systems) (or equivalent non-IITK courses) is desirable.
- Programming maturity (primarily C/C++/Java) is desirable.

Lecture Hours: MonFri 8:00-9:00 AM Tue 9:00-10:00 AM in KD 101

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. This course will discuss programming language abstractions with architecture-aware development to learn to write scalable parallel programs. This is not a “programming tips and tricks” course.

We will have 4–6 assignments to use the concepts learned in class and appreciate the challenges in extracting performance.

Course Contents: The course will focus on a subset of the following topics.

- Challenges in parallel programming, correctness, and performance errors, understanding performance, and performance models
- Exploiting spatial and temporal locality with caches, analytical cache miss analysis
- Performance bottleneck analysis: PAPI counters, using performance analysis tools
- Dependence Testing, Loop Transformations
- Shared-memory programming and Pthreads
- Compiler vectorization: vector ISA, auto-vectorizing compiler, vector intrinsics
- Core OpenMP, Advanced OpenMP, Heterogeneous programming with OpenMP
- Parallel Programming Models and Patterns
- Intel Threading Building Blocks
- GPU Architecture and CUDA Programming

- Shared-memory Synchronization
- Concurrent Data Structures

We may add new, drop existing, or reorder topics depending on progress and class feedback. The course may also involve reading and critiquing related research papers.

Policies:

- Please be on time for class.
- Please try to avoid using laptops and/or mobile devices in class; they are distracting for both the instructor and other students who want to concentrate.
- You are allowed up to **two late** days to submit your assignment, with a 25% penalty for each day.

Evaluation:

Assignments	40%
Midsem	30%
Endsem	30%

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

References:

- Computer Systems: A Programmer's Perspective - R. Bryant and D. O'Hallaron
- Compilers: Principles, Techniques, and Tools - A. Aho, M. Lam, R. Sethi, and J. Ullman
- Computer Architecture: A Quantitative Approach - J. Hennessy and D. Patterson
- Optimizing Compilers for Modern Architectures - R. Allen and K. Kennedy
- Automatic Parallelization: An Overview of Fundamental Compiler Techniques - Samuel P. Midkiff
- An Introduction to Parallel Programming - Peter S. Pacheco
- Programming Massively Parallel Processors: A Hands-on Approach - David B. Kirk and Wen-mei W. Hwu
- A Primer on Memory Consistency and Cache Coherence - Vijay Nagarajan, Daniel J. Sorin, Mark D. Hill and David A. Wood

- The Art of Multiprocessor Programming - Maurice Herlihy and Nir Shavit

  We may read and discuss related materials and research papers, which we will announce in class.