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), CS 330 (Operating Systems) and CS 335 (Compiler Design) (or equivalent non-IITK courses) is desirable.
- Programming maturity (primarily C/C++/Java) is desirable.

Lecture Hours: WedFri 10:30-12:00 PM in KD 102

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 language-level and architecture-specific features like dependences and 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 five or more assignments to use the concepts learned in class and appreciate the challenges in extracting performance.

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

1. Challenges in parallel programming, correctness, and performance errors, understanding performance, and performance models
2. Exploiting spatial and temporal locality with caches, analytical cache miss analysis
3. Dependence Testing, Loop Transformations
4. Shared-memory programming and Pthreads
5. Compiler vectorization: vector ISA, auto-vectorizing compiler, vector intrinsics
6. 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. Shared-memory Synchronization

11. Concurrent Data Structures
12. Performance bottleneck analysis: PAPI counters, Using performance analysis tools

Optional topics

13. Heterogeneous Programming with OpenMP
14. Fork-Join Parallelism

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.

Evaluation:

Class participation/quizzes/paper critiques	5%
Assignments	40%
Midsem	25%
Endsem	30%

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

References:

1. Computer Systems: A Programmer's Perspective - R. Bryant and D. O'Hallaron
2. Computer Architecture: A Quantitative Approach - J. Hennessy and D. Patterson
3. Optimizing Compilers for Modern Architectures - R. Allen and K. Kennedy
4. Automatic Parallelization: An Overview of Fundamental Compiler Techniques - Samuel P. Midkiff
5. An Introduction to Parallel Programming - Peter S. Pacheco
6. Intel Threading Building Blocks: Outfitting C++ for Multi-core Processor Parallelism - J. Reindeers
7. Programming Massively Parallel Processors: A Hands-on Approach - David B. Kirk and Wen-mei W. Hwu
8. Pro TBB – Michael Voss, Rafael Asenjo, and James Reinders

We will also distribute relevant handouts and research papers.