ECE 569 – Spring 2019

High Performance Computing: Technology, Architecture, and Algorithms (3 credits)

Description of Course: ECE 569 stresses the need for and the design of high-performance computing (HPC) systems. HPC is more than just for achieving high performance - it is a compelling vision for how computation can seamlessly scale from a single processor to virtually limitless computing power. The single enabling force for HPC is the use of parallelism. The market demands general-purpose processors that deliver high single threaded performance as well as multi-core throughput for a wide variety of workloads on client, server, and high-performance computing (HPC) systems. This pressure has given us almost three decades of progress toward higher complexity and higher clock rates. Each new generation of process technology requires ever more heroic measures to improve transistor characteristics; each new core microarchitecture must work disproportionately harder to find and exploit instruction-level parallelism (ILP). New commodity parallel computing devices, bring the originally elite high performance computing into the reach of general public. To program and accelerate applications on the new high performance computing devices, we must understand both the computational architecture and the principles of program optimization. This course will (a) provide an overview of existing High-Performance Computing (HPC) software and hardware, (b) present basic software design patterns for high performance parallel computing, (c) introduce CUDA for parallel computing on the Graphics Processing Unit (GPU), (d) compare GPU programming model with the OpenMP and the Message Passing Interface (MPI) standard for leveraging parallelism on a cluster. The approach is hands-on, the students are expected to use the lecture information, a series of assignments and a final project to emerge at the end of the class with parallel programing knowledge that can be immediately applied to their research projects. Throughout the parallel programing with CUDA focused lectures and assignments students will learn the f fundamental Kernel optimization, Memory optimization, and CPU-GPU interaction optimization strategies. We will cover the following specific techniques:

- Improve access pattern to reduce wasted transactions: coalescing
- Reduce redundant access: shared memory, bank conflicts,
- Caching or Non-caching?
- Texture and Constant Caches
- Strive for perfect coalescing (data transpose, padding, 1-thread-per-task vs 1-warp-per-task)
- Finding out bottleneck in memory utilization and memory throughput utilization techniques,
- Latency optimization and Latency hiding techniques
- Instruction optimization: Arithmetic Instruction Throughputs
- Profiling assisted optimization: occupancy, warp utilization, bandwidth optimization
- Control Flow optimization, divergent branches:
  Overlapped execution using streams
- Minimizing CPU-GPU data transfer
- Streams and Async API
- Pinned (non-pageable) memory
- Overlap kernel and memory copy

Feel free to contact **Dr. Ali Akoglu (akoglu@ece.arizona.edu)** if you have any question.