CS 6170 : ADVANCED HIGH PERFORMANCE COMPUTING

Semester Hours:	3.0	*Contact Hours:* 3
Coordinator:	Robert Green	
Text:	Programming Massively Parallel Processors: A Hands-on Approach 3rd Edition	
Author(s):	Kirk and Hwu	
Year:	Morgan Kaufmann, 2016	

SPECIFIC COURSE INFORMATION

Catalog Description:

The course introduces high performance computing techniques and their relevant implementation issues for exploring advanced computing environments Co-processors. The architecture of the advanced computing environments, advanced parallel algorithms, and performance issues are discussed. Hands-on projects are considered for various practical algorithms and applications. Prerequisites: CS 5170, or permission of instructor.

Course Type: **ELECTIVE**

SPECIFIC COURSE GOALS

- I am able to compare the architectural differences in Co-processors.
- I am able to redesign existing computing algorithms suitable for Co-processor environments.
- I am able to explain advanced HPC performance issues.
- I am able to implement accelerator-based programs for a practical application.

LIST OF TOPICS COVERED

- Review on Parallel Computing
  - Overview of multi-processor and multi-threading
  - Traditional parallel computing
- Introduction to GPU Computing
  - Architecture of modern GPUs
  - Parallel programming languages and models
- GPU: Data Parallel Computing and Scalable Parallel Execution

    - Data parallelism
    - Device global memory and data transfer
    - Kernel functions and threading
    - Thread optimization, multidimensional data
    - Synchronization, transparent scalability, latency tolerance
- GPU: Memory and Data Locality
    - Memory type, memory access efficiency
    - Tiling for reduced memory traffic
    - Memory as a limiting factor to parallelism
- GPU: Performance and Numerical Considerations
    - Global memory bandwidth
    - Warps and SIMD hardware, dynamic partitioning resources, thread granularity
    - Representations, accuracy, linear solvers and numerical stability
- GPU: Parallel Algorithms and Parallel Patterns
    - Applications case studies (e.g., visualization, machine learning)
- GPU: Parallel Programming and Computational Thinking
- Introduction to Xeon Phi
    - Architecture from the programmer's perspective
    - Software tools
- Xeon Phi: Programming Models
    - Native application
    - Offload programming models, multiple coprocessors
    - MPI applications and heterogeneous clustering
- Xeon Phi: Porting Applications and Parallel Scalability
    - Reliance on complier and libraries
    - Cross-compilation of user applications, performance expectations
    - Vectorization, multi-threading, task parallelization
- Xeon Phi: Optimization
    - Finding bottlenecks with VTune amplifier
    - MPI diagnostics using Trace Analyzer and Collector
    - MKL, scalar optimization, thread parallelism

  - Memory access and cache utilization
  - Data persistence and PCIe traffic
  - MPI applications on clusters with co-processors

EXAMPLE PROJECTS

- **Graph Engine on Single Desktop Computer**
  - GraphChi (https://github.com/GraphChi/graphchi-cpp)
  - Using novel graph representations and some optimization techniques, GraphChi performs as well as cluster-based graph processing engines. In this project, GraphChi is extended to GPUs to enhance the performance.
- **Graph Processing Engine**
  - CuSha (https://github.com/farkhor/CuSha)
  - CuSha is a state-of-the-art graph processing engine on GPUs. However, its main weakness is the inefficiency for supporting graph queries (e.g., finding the neighbor nodes of some randomly chosen nodes). This project studies the source of the inefficiency and addresses it by proposing new graph representations or optimizations.
- **Exploring Multiple GPUs**
  - Multi-GPU graph processing is challenging due to the difficulty of balanced graph partitioning and efficient inter-GPU communication handling. In this project, CuSha is extended to support multiple-GPUs.
- **Approximate Computing on GPUs**
  - SAGE (http://dl.acm.org/citation.cfm?id=2540711)
  - Approximate computing sacrifices accuracy for speed and power saving. SAGE is a framework to do approximate computing on GPUs. In this project, SAGE is explored for image processing and machine learning algorithms.
- **Processing Big Graphs on Xeon Phis**
  - In this project, Xeon Phis processing is considered for processing very large graphs. Widely-used graph algorithm (e.g., BFS) will be implemented on a single machine and on multiple machines with Xeon Phis.

RECOMMENDED REFERENCES

- CUDA Handbook: A Comprehensive Guide to GPU Programming, by Wilt, Addison-Wesley Professional, 2013
- CUDA by Example: An Introduction to General-Purpose GPU Programming, by Sanders and Kandrot, Addison-Wesley Professional, 2010
- Parallel Programming and Optimization with Intel Xeon Phi Coprocessor 2nd Ed., by Vladimirov, Asai, and Karpusenko, Colfax International, 2015