GPGPU Programming

Fall-2019

Instructor Information

Instructor	Email	Office Location & Hours
Dr. Ayaz ul Hassan Khan	ayazhk@gmail.com	TBA

General Information

Description

Learn how to program heterogeneous parallel computing systems and achieve: high performance and energy-efficiency, functionality and maintainability, scalability across future generations, parallel programming API, tools and techniques, principles and patterns of parallel algorithms, processor architecture features and constraints. This is a hands on programming course. We will be using Google Colaboratory environment for programming in CUDA C and pyCUDA which is free to use with one NVIDIA GPU access. So, you don't need to have personal GPU machine to take this course.

Expectations and Goals

- List the major difference between latency devices (CPU cores) and throughput devices (GPU cores)
- State the importance and nature of scalability and portability in parallel programming
- Recognize the main venues and developer resources for GPU computing, write CUDA programs
- Estimate memory bandwidth requirements, analyze the problem and exploit data parallelism
- Evaluate and analyze the implementation of parallel numerical algorithms
- Evaluate some valuable tools and resources from the CUDA toolkit
- Recognize the effectiveness of related parallel programming model: OpenACC

Course Materials

Required Text

- Programming Massively Parllalel Processors: A Hands-on Approach, 2nd Edition by David Kirk and Wen-Mei Hwu
- CUDA by Example: An Introduction to General – Purpose GPU Programming, Jason Sanders and Edward Kandrot

Reference Text

- CUDA C Programming Guide, NVIDIA
- https://nvidia.qwiklab.com/
- www.openacc.org, http://www.nvidia.com/object/cuda_home_new.htm l
- Lecture Notes, Handouts, and Selected Research Papers

Pre-Requisites: Computer Programming, Computer Architecture

Brief List of Topics to be covered:

Topic
Introduction to Heterogeneous Parallel Computing
Introduction to CUDA C and pyCUDA
CUDA Parallelism Model
Memory Model and Locality
Kernel – Based Parallel Programming
Performance Considerations: Memory
Parallel Computation Patterns: Stencil, Reduction, Scan
Related Programming Models: MPI, OpenACC

About Course Instructor:

Dr. Ayaz ul Hassan Khan received his BS degree from NED-Pakistan, MS degree in Computer Science from LUMS-Pakistan and PhD degree in Computer Science and Engineering with the specialization in Parallel Computing from King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia. He has 14 years of experience in both industry and academics. He has hands-on experience in Database programming, Network Programming, Embedded Systems Programming, and Parallel Programming. As part of his PhD dissertation work, he has got extensive experience in optimizing parallel programs for Graphics Processing Units (GPUs) using CUDA. He has explored several optimizations for CUDA programs that are helpful for the efficient implementation of parallel numerical algorithms for solving linear systems that is a key issue in the design of large scale simulation systems. In addition to that he has also got hands on experience in setting up GPU servers including the installations of hardware, operating system, remote access setup, run-time profilers, CUDA compiler and debugging tools. Moreover, he has used a new innovative approach for writing code translators to implement RT-CUDA Compiler. This approach is based on parse tree modification generated through the grammar instead of applying actions into the grammars itself. The applied approach is more flexible and modular approach to ease the implementation of code transformations and new optimizations in future. His current areas of interest include Parallel and Distributed Computing, High Performance Computing, Computer Architecture, Operating Systems, Deep Learning and Big Data Analytics. He has published 8 journals and 11 conference papers/posters in the field of his research areas in recent years. Check out the following for details: https://sites.google.com/site/ayazresearch/

Related Links:

- https://www.nvidia.com/en-us/gtc/
- https://developer.nvidia.com/cuda-zone
- https://developer.nvidia.com/cuda-toolkit
- https://www.openacc.org/
- https://www.open-mpi.org/

Programming a graphics processing unit (GPU) seems like a distant world from Java programming. This is understandable, because most of the use cases for Java are not applicable to GPUs.Â You still need to understand how the GPU is programmed, but you can approach GPGPU in a more Java-friendly way. Moreover, Aparapi provides an easy way to bind OpenGL contexts to the OpenCL layer underneathâ€”thus enabling the data to stay entirely on the video cardâ€”and thereby avoid memory latency issues. General Purpose Computing on GPUs (GPGPU). Other than rendering graphics, a GPU (Graphics Processing Unit) can also be used for general-purpose computing. A GPU usually has a lot more cores than a CPU but the speed of each core in GPU is slower than that of CPU.Â OpenCL (Open computing language) is a framework for writing programs which can directly run on GPUs or CPUs or other kind of computing processors. OpenCL is based on C99 (a past version of C programming language). Why use GPUs? Rolling your own GPGPU apps. GPGPU/Streaming Languages. BrookGPU. Streams.Â AMD + Stream Processing Future. Sequoia â€“ Programming the memory system. GPGPU on the HD 2900XT. GPGPU on the HD 2900XT, cont. GPGPU on the HD 2900XT, cont. Performance basics for GPGPU â€“ HD 2900XT. What is protein folding. Protein folding in the cell. GPGPU. Multi-GPU Programming. Georgii Evtushenko. Follow.Â Therefore, Iâ€™ll highlight some general multi-GPU programming principles before diving into communication specific details. Basics of multi-GPU programming. Avoiding multi-GPU support. First of all, why and when do you need to invest your time into multi-GPU support? There is no reason to waste your time unless you need to accelerate a particular instance of your application. Another reason for multi-GPU programming is memory limitations. Recent papers in GPGPU (General Purpose GPU) Programming. Papers. People. An Optimized GPU-Accelerated Route Planning of Multi-UAV Systems Using Simulated Annealing. Save to Library. Download. by Adnan Ozsoy. â€¢ 2. GPU Computing, GPGPU (General Purpose GPU) Programming. OptimizaciÃ³n en Paralelo Basada en Poblaciones Usando GPGPU. Save to Library. Download.Â We are witnessing an increasing adoption of GPUs for performing general purpose computation, which is usually known as GPGPU.