GPU Teaching Kit

Accelerated Computing

Lecture 1.1 – Course Introduction

Course Introduction and Overview

Course Goals

- **Learn how to program heterogeneous（异构）parallel computing systems and achieve**
  - High performance and energy-efficiency
  - Functionality(功能性) and maintainability(可维护性)
  - Scalability(可扩展性) across future generations
  - Portability(可移植性) across vendor devices
- **Technical subjects**
  - Parallel programming API, tools and techniques
  - Principles and patterns of parallel algorithms
  - Processor architecture features and constraints

Course Content

Module 1 Course Introduction	• Course Introduction and Overview • Introduction to Heterogeneous Parallel Computing • Portability and Scalability in Heterogeneous Parallel Computing
Module 2 Introduction to CUDA C	• CUDA C vs. CUDA Libs vs. OpenACC • Memory Allocation and Data Movement API Functions • Data Parallelism and Threads • Introduction to CUDA Toolkit
Module 3 CUDA Parallelism Model	• Kernel-Based SPMD Parallel Programming • Multidimensional Kernel Configuration • Color-to-Greyscale Image Processing Example • Blur Image Processing Example
Module 4 Memory Model and Locality	• CUDA Memories • Tiled Matrix Multiplication • Tiled Matrix Multiplication Kernel • Handling Boundary Conditions in Tiling • Tiled Kernel for Arbitrary Matrix Dimensions
Module 5 Kernel-based Parallel Programming	• Histogram (Sort) Example • Basic Matrix-Matrix Multiplication Example • Thread Scheduling • Control Divergence

Course Content

Module 6 Performance Considerations: Memory	• DRAM Bandwidth • Memory Coalescing in CUDA
Module 7 Atomic Operations	• Atomic Operations
Module 8 Parallel Computation Patterns (Part 1)	• Convolution • Tiled Convolution • 2D Tiled Convolution Kernel
Module 9 Parallel Computation Patterns (Part 2)	• Tiled Convolution Analysis • Data Reuse in Tiled Convolution
Module 10 Performance Considerations: Parallel Computation Patterns	• Reduction • Basic Reduction Kernel • Improved Reduction Kernel
Module 11 Parallel Computation Patterns (Part 3)	• Scan (Parallel Prefix Sum) • Work-Inefficient Parallel Scan Kernel • Work-Efficient Parallel Scan Kernel • More on Parallel Scan

Course Content

Module 12 Performance Considerations: Scan Applications	• Scan Applications: Per-thread Output Variable Allocation • Scan Applications: Radix Sort • Performance Considerations (Histogram (Atomics) Example) • Performance Considerations (Histogram (Scan) Example)
Module 13 Advanced CUDA Memory Model	• Advanced CUDA Memory Model • Constant Memory • Texture Memory
Module 14 Sparse Matrix Computation	• Parallel SpMV Using CSR • Padding and Transposition • Using a Hybrid Approach to Regulate Padding • Sorting and Partitioning for Regularization
Module 15 Merge Sort	• Sequential Merge Algorithm • Parallelization Approach • Co-Rank Function Implementation • Basic Parallel Merge Kernel • Tiled Merge Kernel • Circular-Buffer Merge Kernel
Module 16 Graph Search	• Breadth-First Search • Sequential BFS Function • Parallel BFS Function • Optimizations
Module 17 CUDA Python Using Numba	• CUDA Python using Numba

Teaching Arrangement

- **32 Lecture Hours**
- **16 Laboratory Hours (Important)**

Grade Components

- **In-class performance (70%)**
  - **Final written exam (70%)**
  - **Regular performance (30%)**
    - **Attendance (20%)**
    - **In-class quizzes (80%)**
- **Lab performance (30%)**
  - **Nvidia Certifications (50%)**
    - **2 NVIDIA certifications**
    - **2 corresponding quizzes.**
  - **4 assignments (50%)**

Students who miss more than three classes will be disqualified from participating in the final exam!!!

Textbooks, Reference Books, and Resources

- **Textbooks**
  - 大卫·B. 柯克(David B. Kirk),胡文美(Wen-mei W. Hwu)著
  - 大规模并行处理器程序设计（英文版·原书第3版）
  - 机械工业出版社, 2020年
- **Reference Books**
  - 高性能计算系列丛书·CUDA并行程序设计：GPU编程指南 [CUDA Programming:A Developer‘s Guide to Parallel Computing with GPUs], 机械工业出版社, 2014年
  - Professional Cuda C Programming，Wiley , 2014年
- **Resources**
  - https://docs.nvidia.com/cuda/doc/index.html

GPU Teaching Kit

Accelerated Computing

Lecture 1.2 – Course Introduction

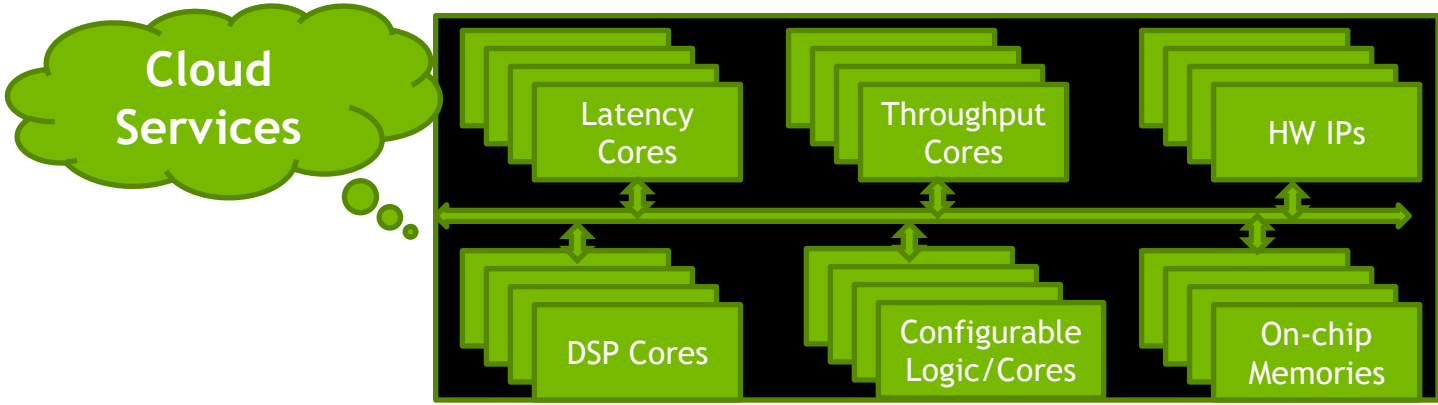
Introduction to Heterogeneous Parallel Computing

Objectives

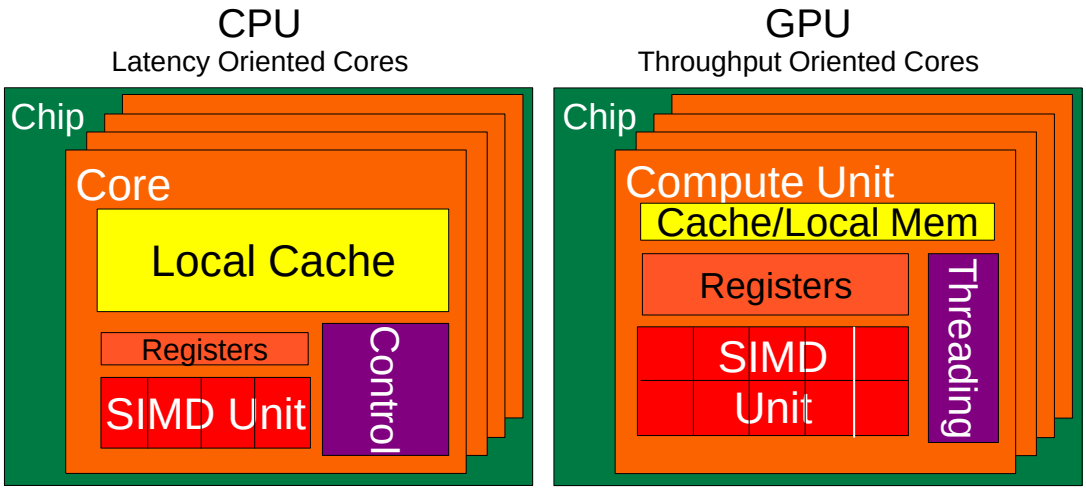
- To learn the major differences between latency devices (CPU cores) and throughput devices (GPU cores)
- To understand why winning applications increasingly use both types of devices

Heterogeneous Parallel Computing

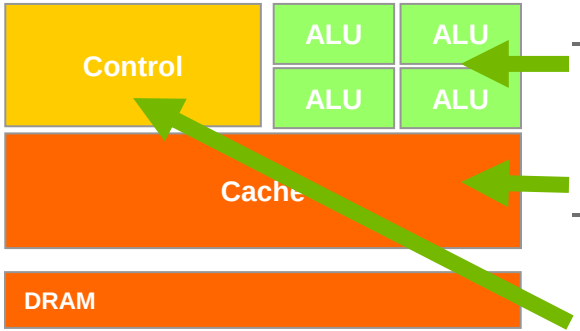
– Use the best match for the job (heterogeneity in mobile SOC(System on a Chip))



CPU and GPU are designed very differently

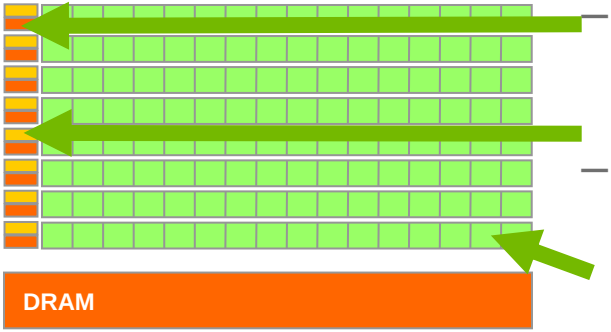


CPUs: Latency Oriented Design



- Powerful ALU
  - Reduced operation latency
- Large caches
  - Convert long latency memory accesses to short latency cache accesses
- Sophisticated control
  - Branch prediction for reduced branch latency
  - Data forwarding for reduced data latency

GPUs: Throughput Oriented Design

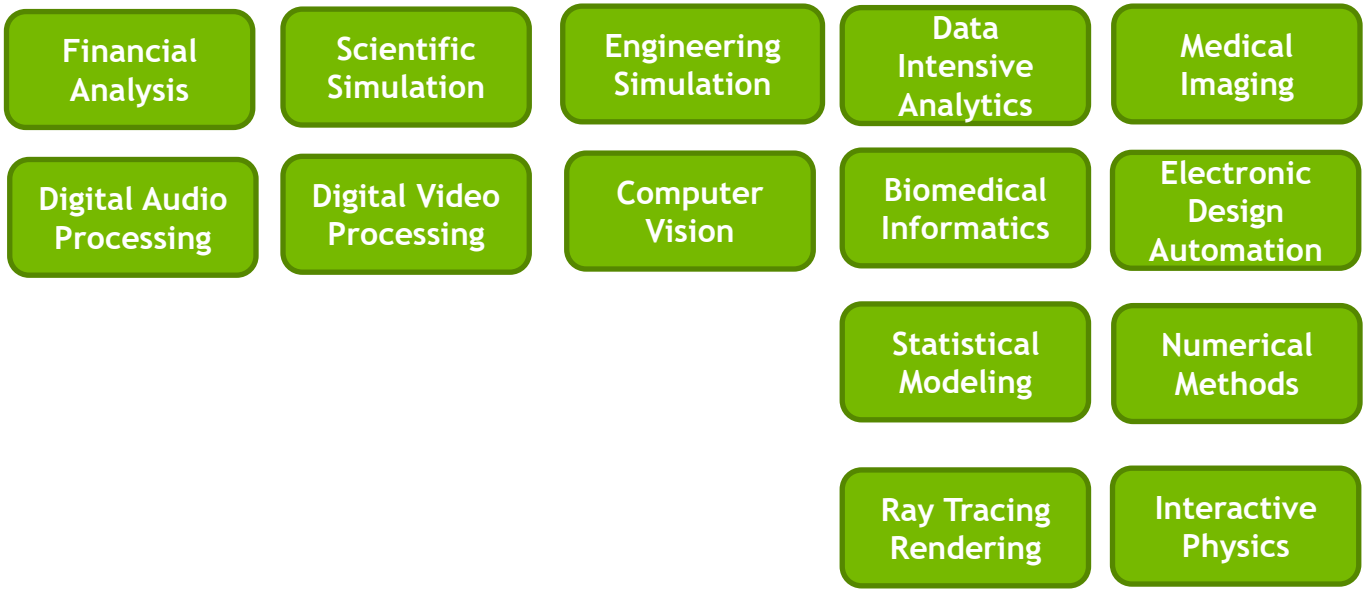


- Small caches
  - To boost memory throughput
- Simple control
  - No branch prediction
  - No data forwarding
- Energy efficient ALUs
  - Many, long latency but heavily pipelined for high throughput
- Require massive number of threads to tolerate latencies
  - Threading logic
  - Thread state

Winning Applications Use Both CPU and GPU

- CPUs for sequential parts where latency matters
  - CPUs can be 10X+ faster than GPUs for sequential code
- GPUs for parallel parts where throughput wins
  - GPUs can be 10X+ faster than CPUs for parallel code

Heterogeneous Parallel Computing in Many Disciplines



GPU Teaching Kit

Accelerated Computing

Lecture 1.3 – Course Introduction

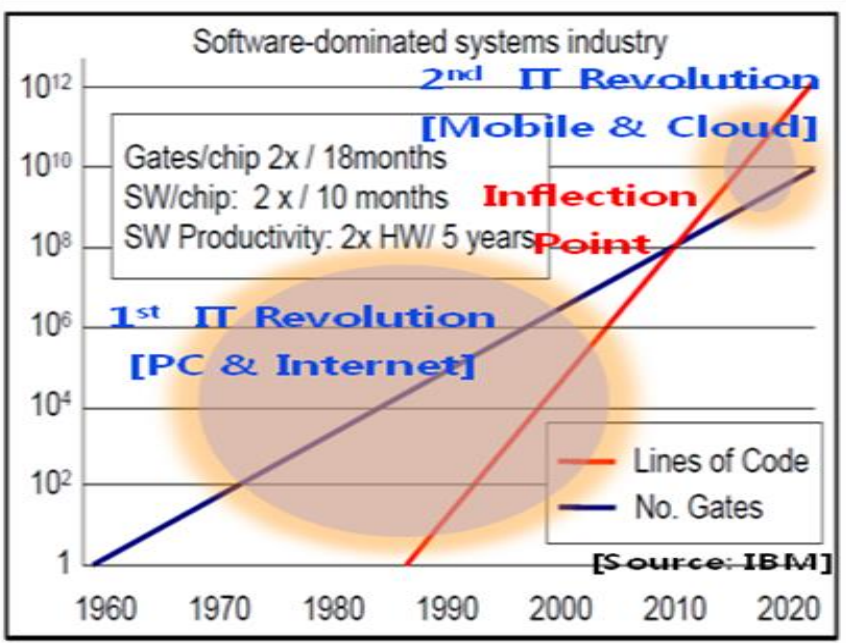
Portability and Scalability in Heterogeneous Parallel Computing

Objectives

– To understand the importance and nature of **scalability** and **portability** in parallel programming

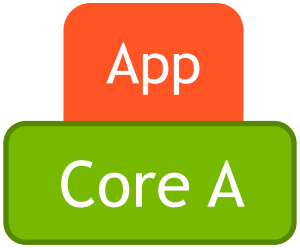
Software Dominates System Cost

- SW lines per chip increases at 2x/10 months
- HW gates per chip increases at 2x/18 months
- Future systems must <u>minimize software redevelopment</u>



Keys to Software Cost Control

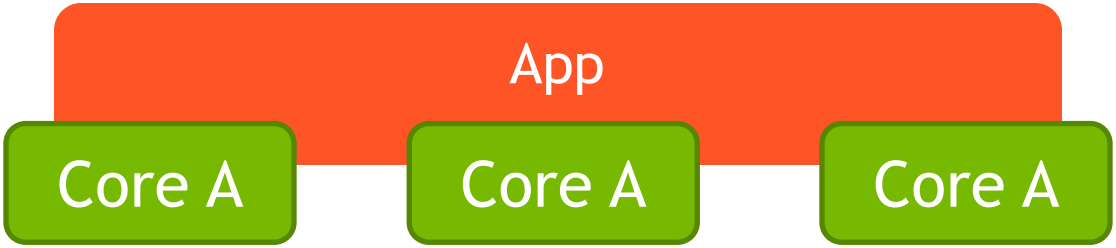
– Scalability



Keys to Software Cost Control

- Scalability
  - **The same application runs efficiently on new generations of cores**

Keys to Software Cost Control

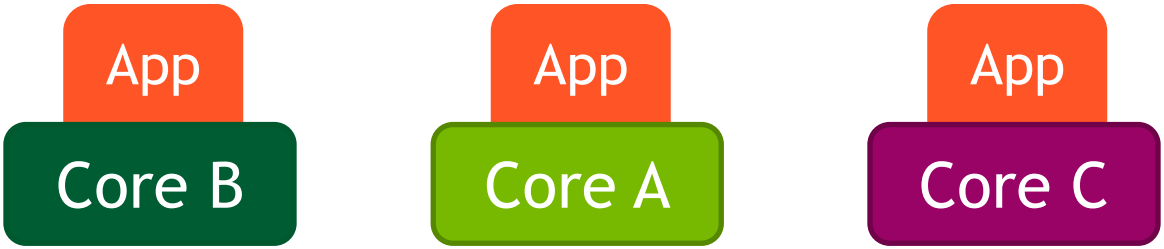


- Scalability
  - The same application runs efficiently on new generations of cores
  - **The same application runs efficiently on more of the same cores**

More on Scalability

- Performance growth with HW generations
  - Increasing number of compute units (cores)
  - Increasing number of threads
  - Increasing vector length
  - Increasing pipeline depth
  - Increasing DRAM burst size
  - Increasing number of DRAM channels
  - Increasing data movement latency

Keys to Software Cost Control



- Scalability
- **Portability**
  - The same application runs efficiently on different types of cores

Keys to Software Cost Control

- Scalability
- Portability
  - The same application runs efficiently on different types of cores
  - The same application runs efficiently on systems with different organizations and interfaces

More on Portability

- Portability across many different HW types
  - Across ISAs (Instruction Set Architectures) - X86 vs. ARM, etc.
  - Latency oriented CPUs vs. throughput oriented GPUs
  - Across parallelism models - SIMD vs. threading
  - Across memory models - Shared memory vs. distributed memory

GPU Teaching Kit

Accelerated Computing