Topics

- Compute Internals
- CPU Internals
- Arm and Intel CPUs
- GPU Computing

Learning Objectives

- Describe the different types of hardware implementations for AI computing such as Edge, Embedded, Cloud and On-Premise.
- Describe the difference between CPU and GPU processing
- Write and execute code and compare differences between CPU and GPUs

AI Hardware

Computing Implementations

Cloud – Data is external to your network and accessed over HTTP

Edge – Access Data at the Edge of your Network or from the Device

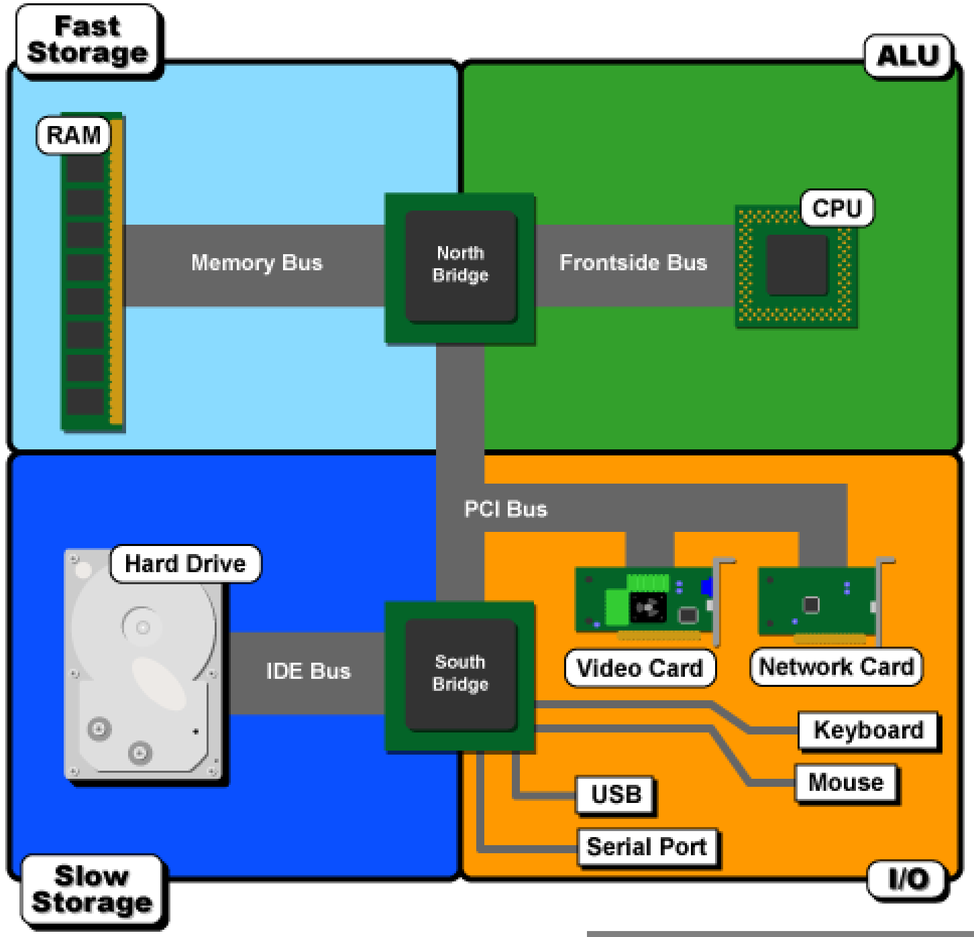
On-Premise Data is located within your local network

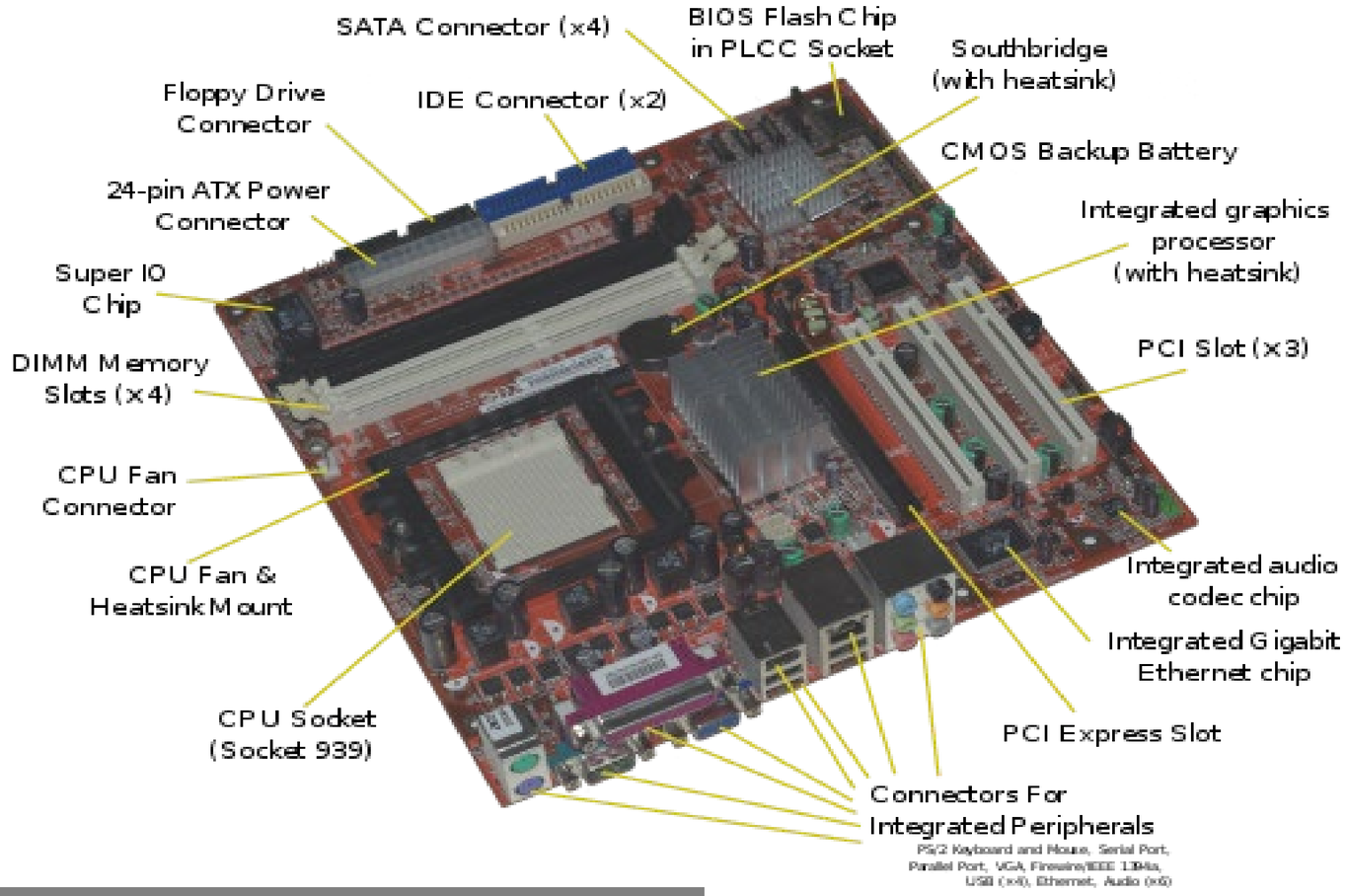
AIoT – Artificial Intelligence of Things – AI and Internet of Things combined

Embedded-Data is local on the machine

Compute Internals

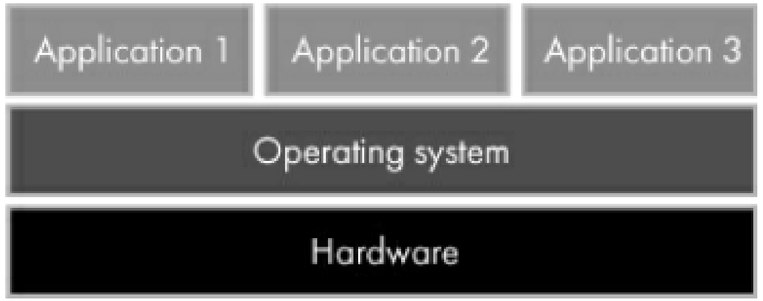
Computer Internal





Operating System

Layer between Hardware and Software



Justice, Matthew. How Computers Really Work (p. 196). No Starch Press. Kindle Edition.

Operating System Kernel

An operating system kernel is responsible for managing memory, facilitating device I/O, and providing a set of system services for applications. The kernel allows multiple programs to run in parallel and share hardware resources. It is the core part of an operating system, but it alone provides no way for end users to interact with the system.

Operating systems also include non-kernel components that are needed for a system to be of use. This includes the shell, a user interface for working with the kernel. The terms shell and kernel are part of a metaphor for operating systems, where the OS is thought of as a nut or seed. The kernel is at the core; a shell surrounds it.

Justice, Matthew. How Computers Really Work (pp. 195-196). No Starch Press. Kindle Edition.

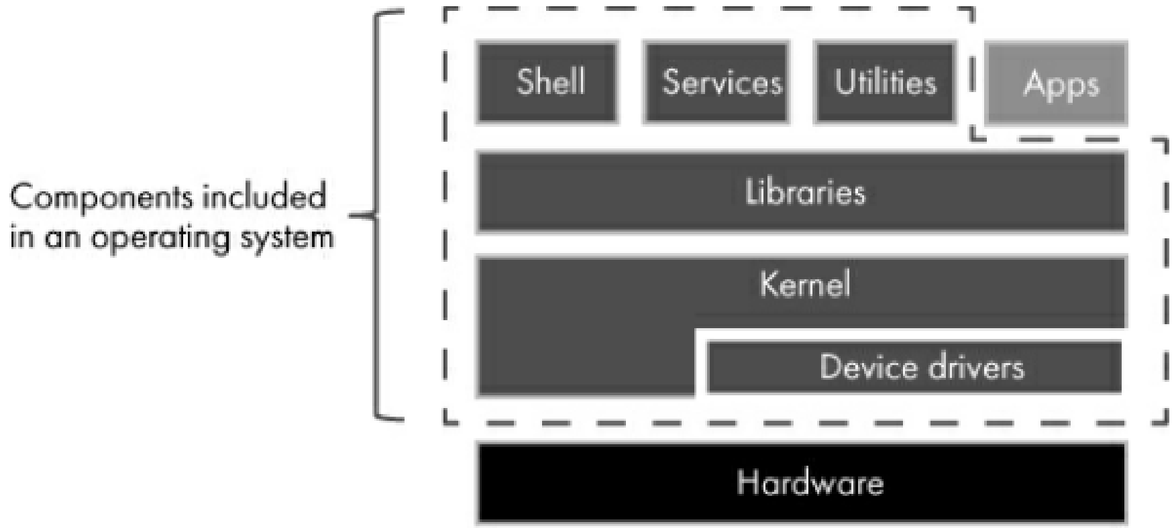
Hardware Interface

The shell can be either a command line interface (CLI) or a graphical user interface (GUI). Some examples of shells are the Windows shell GUI (including the desktop, Start menu, taskbar, and File Explorer), and the Bash shell CLI found on Linux and Unix systems.

When it comes to interacting with hardware, the kernel acts in partnership with device drivers. A device driver, or simply driver, is software designed to interact with specific hardware. An operating system's kernel needs to work with a wide variety of hardware, so rather than designing the kernel to know how to interact with every hardware device in the world, software developers implement the code for specific devices in device drivers. Operating systems typically include a set of device drivers for common hardware and also provide a mechanism for installing additional drivers.

Justice, Matthew. How Computers Really Work (p. 196). No Starch Press. Kindle Edition.

Components in an Operating System

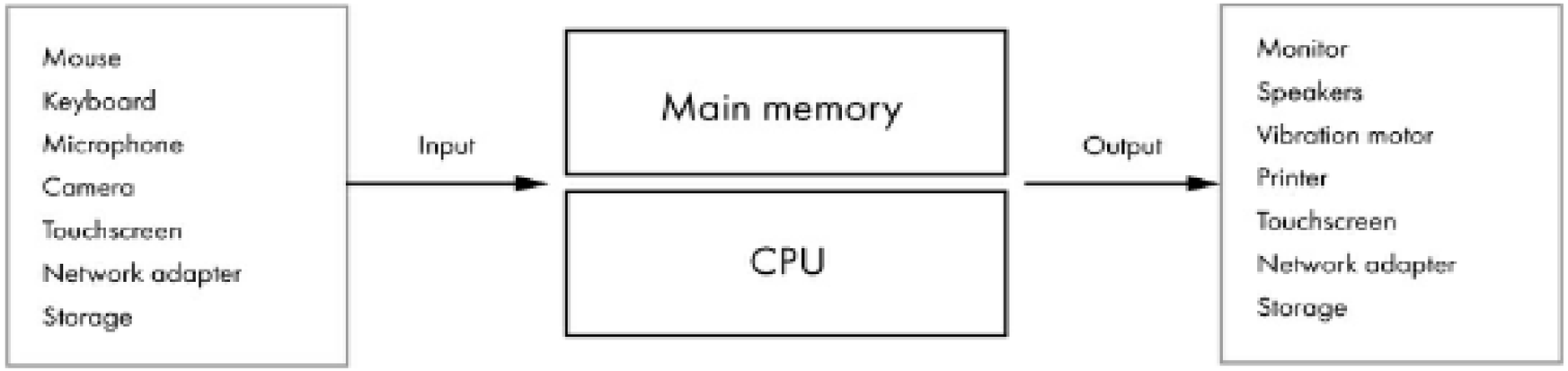


Justice, Matthew. How Computers Really Work No Starch Press. Kindle Edition.

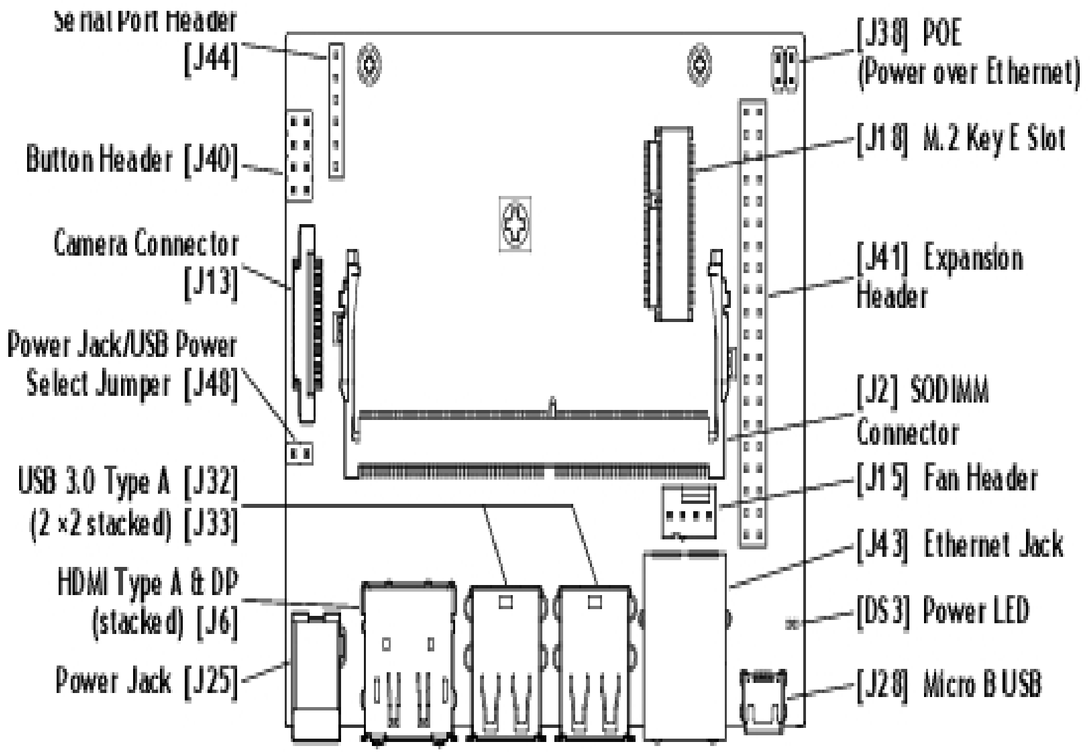
Input/Output Operations

Computers need to interact with external environment

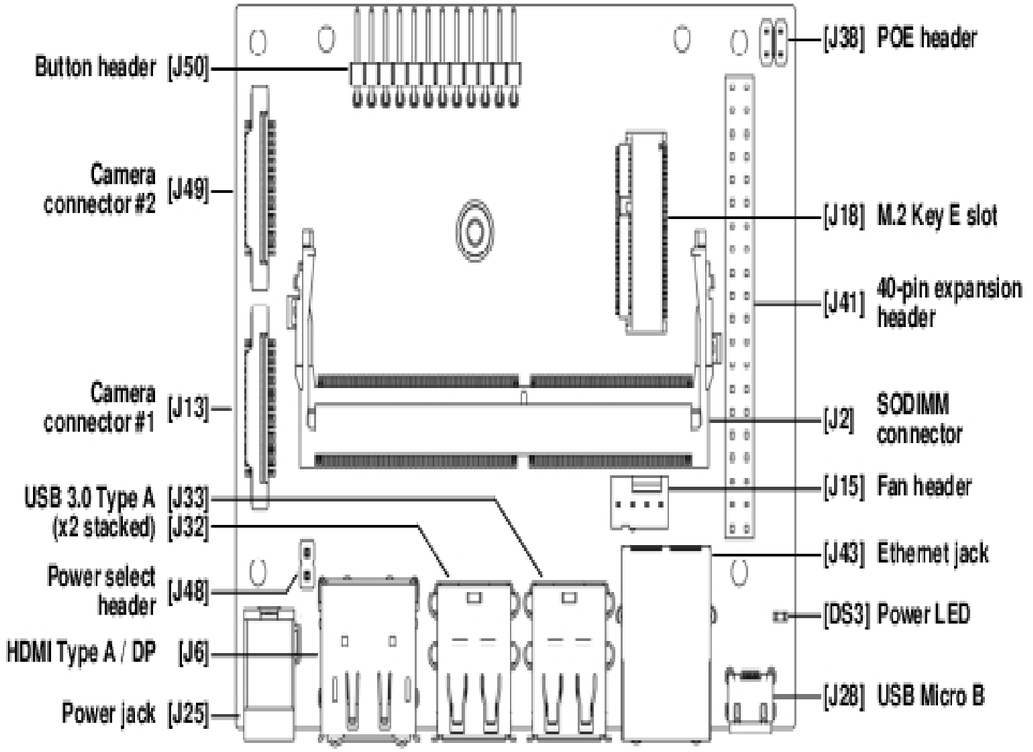
- I/O Devices
- OS has software device drivers for each device



Nano Carrier Board Layout



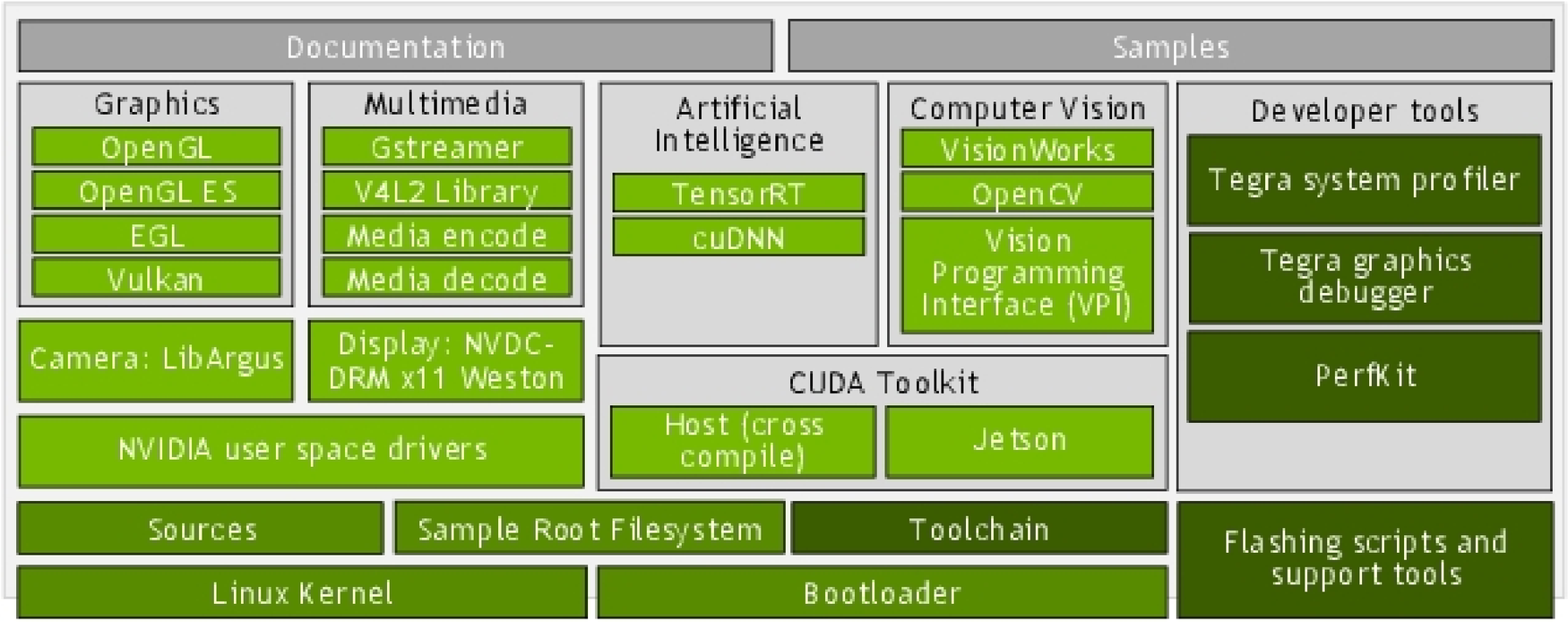
Top View



Bottom View

ref:
https://docs.NVIDIA.com/jetson/l4t/index.html#page/Tegra%20Linux%20Driver%20Package%20Development%20Guide/hw_setup.html#wwpID0E0YR0HA

Jetson Board Support Architecture



Jetson Board Support Architecture

Package Descriptions

Jetson Board Support Package

- **Linux Kernel**: A UNIX-like computer operating system kernel mostly used for mobile devices.
- **Sample Root Filesystem derived from Ubuntu**: A sample root filesystem of the Ubuntu distribution. Helps you create root filesystems for different configurations.
- **Toolchain**: A set of development tools chained together by stages that runs on various architectures.
- Bootloader: Boot software boot for initializing the system on the chip.
- **Sources**: Source code for kernel and multimedia applications.
- **NVIDIA® Jetson™ User Space Drivers**: Provides the NVIDIA drivers that run in the user space.
- Flashing Support Scripts and Tools: A set of scripts and associated tools to assist in flashing your system.
- Sample Source Code and Applications: Sample source code for developing embedded applications for the Jetson platform.

For additional descriptions, see: https://docs.NVIDIA.com/jetson/l4t/index.html#page/Tegra%20Linux%20Driver%20Package%20Development%20Guide/overview.html#

Processors

Processors

- CPU – Central Processing Units
- GPU – Graphics Processing Units
- FPGA – Field Programmable Gate Arrays
- ASIC – Application Specific Integrated Circuits

Parallel Computing – Flynn's Taxonomy*

SISD- Single Instruction Single Data computing system is a uniprocessor machine that executes single instruction on a single data stream (uniprocessor – earlier CPUs)

SIMD- Single Instruction Multiple Data is a computing system with several identical processors each with local memory and work under the control of a single instruction stream (Modern CPUs & GPUs).

MISD- multiple instruction, single data – different operators on same data (none available commercially)

MIMD- Multiple instruction, multiple data (Sun/IBM SMP – Symmetric Multi-Processing Machines)

More info:

https://www.geeksforgeeks.org/difference-between-simd-and-mimd/

Processors

SIMD -Single Instruction Multiple Data.	MIMD -Multiple Instruction Multiple Data.
Less memory.	Large memory.
Less cost	Higher cost
Single decoder.	Multiple decoders.
Latent or tacit synchronization.	Accurate or explicit synchronization.
Synchronous programming.	Asynchronous programming.
Simple in complexity	More complex
Less efficient for performance	More efficient for performance

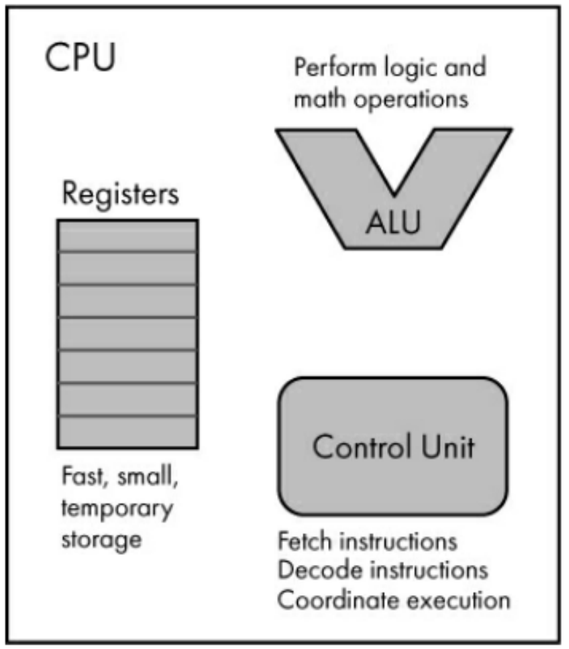
Components of a CPU

CPU

Registers

ALU

Control Unit



Multicore CPU

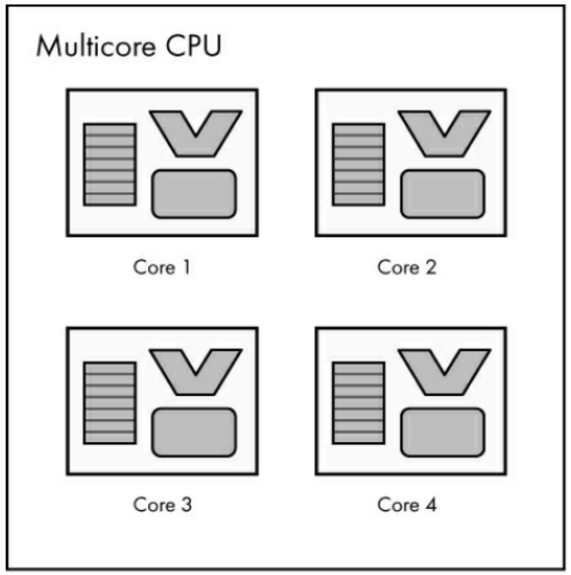


Figure 7-9: A four-core CPU—each core has its own registers, ALU, and control unit

CPU Cores

PHYSICAL AND LOGICAL CORES Not all cores are equally capable of parallelism. A physical core is a hardware implementation of a core within a CPU. Logical cores represent the ability of a single physical core to run multiple threads at once (one thread per logical core). Intel refers to this capability as hyper-threading.

A computer with two physical cores, each with two logical cores, has a total of four logical cores and run four threads at once. Logical cores cannot achieve the full parallelism of physical cores.

Justice, Matthew. How Computers Really Work (p. 203). No Starch Press. Kindle Edition.

Cloud vCore consideration

You decide to reserve a virtual instance with 4 vCores:

How many physical processors are available?

How many logical cores are available?

	CISC (Complex Instruction Set Computer)	RISC (Reduced Instruction Set Computer)
1	Emphasis on hardware	Emphasis on software
2	Includes multi-clock complex instructions	Single-clock, reduced instruction only
3	Small code sizes	Typically larger code sizes
4	Many addressing modes	Few addressing modes.
5	An easy compiler design	A complex compiler design.
6	Pipelining does not function correctly here because of complexity in instructions.	Pipelining is not a major problem and this option speeds up the processors.

Ref:
https://www.microcontrollertips.com/risc-vs-cisc-architectures-one-better/#:~:text=One%20of%20the%20major%20differences%20between%20RISC%20and,and%20CISC%20emphasizes%20efficiency%20in%20instructions%20per%20program.
https://cs.stanford.edu/people/eroberts/courses/soco/projects/risc/risccisc/

ARM Processor

The Arm architecture is one of the most popular processor architectures in the world today, with several billion Arm-based devices shipped every year.

There are three architecture profiles: A, R and M.

A-profile (Applications)	R-profile (Real-time)	M-profile (Microcontroller)
• High performance	• Targeted at systems with real-time requirements.	• Smallest/lowest power. Small, highly power-efficient devices.
• Designed to run a complex operating system, such as Linux or Windows.	• Commonly found in networking equipment, and embedded control systems.	• Found at the heart of many IoT devices.

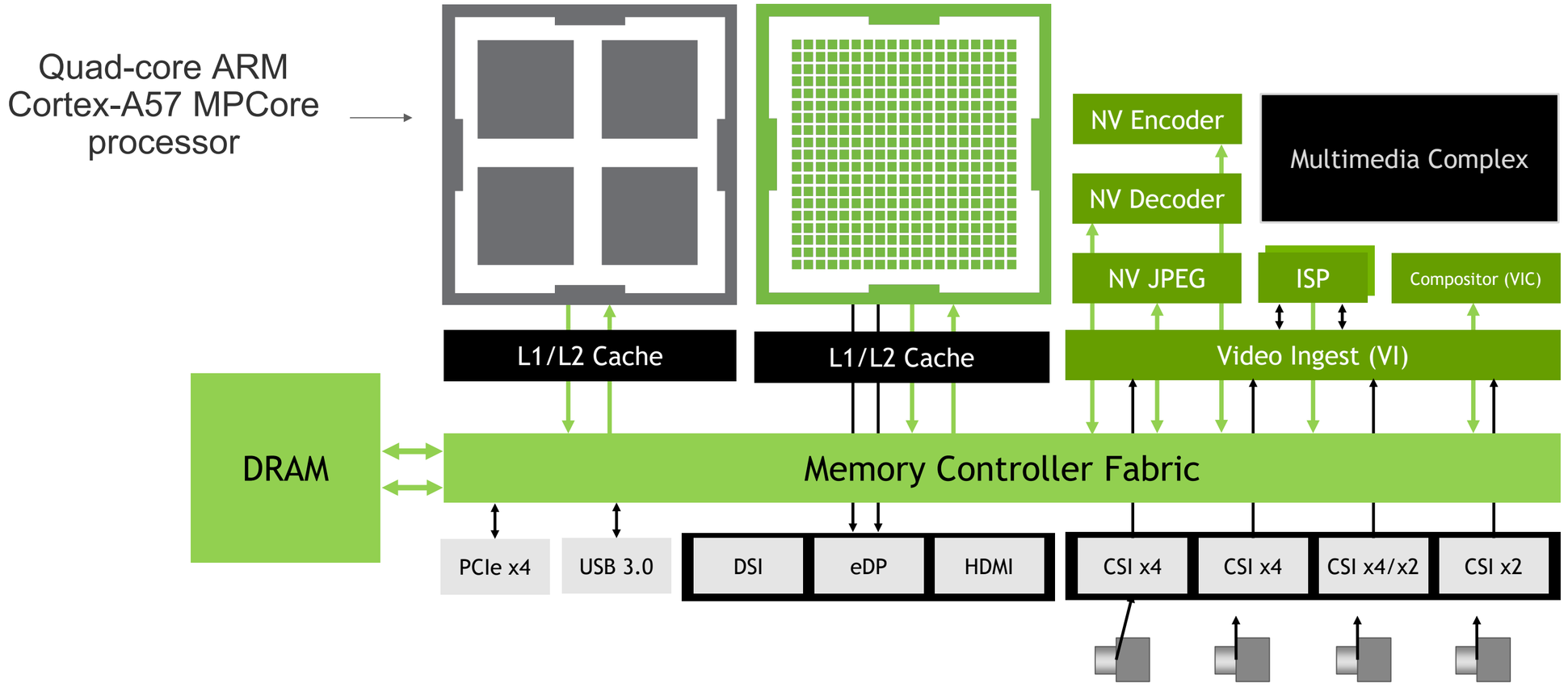
These three profiles allow Arm architecture to be tailored to the needs of different use cases, while still sharing several base features.

ARM vs Intel CPUs

ARM vs. Intel

type	ARM	Intel
starting from	RISC	CISC
speed	moderate	important
power consumption	important	moderate
compiler	important	moderate

Jetson Nano Block Diagram



GPUs

GPU vs CPU

Mythbusters Demo GPU versus CPU - YouTube

CPU

Latency – beginning to end duration of performing a single computation (*Tuomanen, 2018)

CPUs are engineered to reduce the latency of a single computation.

Computations are sequential

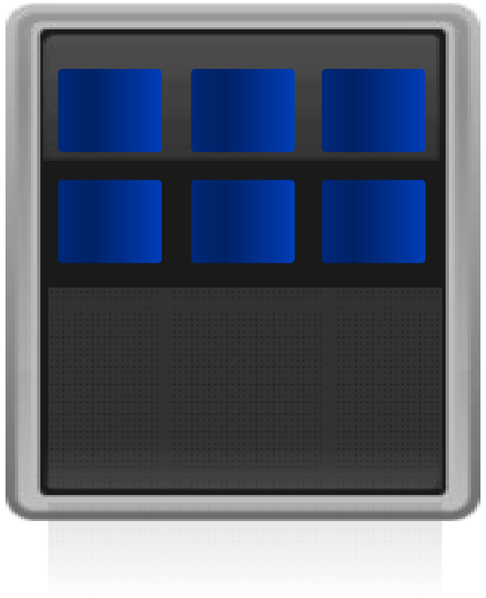
Fewer cores that are restricted with the number of processes it can compute and it handles those processes very fast

*Hands-On GPU Programming with Python and CUDA

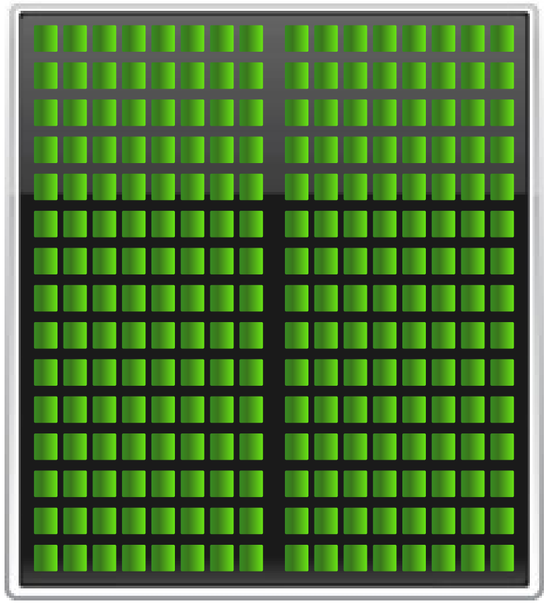
Accelerated Computing

10x Performance & 5x Energy Efficiency for HPC

CPU
Optimized for
Serial Tasks



GPU Accelerator
Optimized for
Parallel Tasks



CPU Strengths

- Very large main memory
- Very fast clock speeds
- Latency optimized via large caches
- Small number of threads can run very quickly

CPU Weaknesses

- Relatively low memory bandwidth
- Cache misses very costly
- Low performance/watt

GPU

Each core is much simpler than that of the CPU and each core on its own is not as fast as a CPU

It’s the number of cores that make a difference. GPU has hundreds to thousands of cores compared to a CPU which has 1 to 6 cores

Computations are done in parallel, asynchronously.

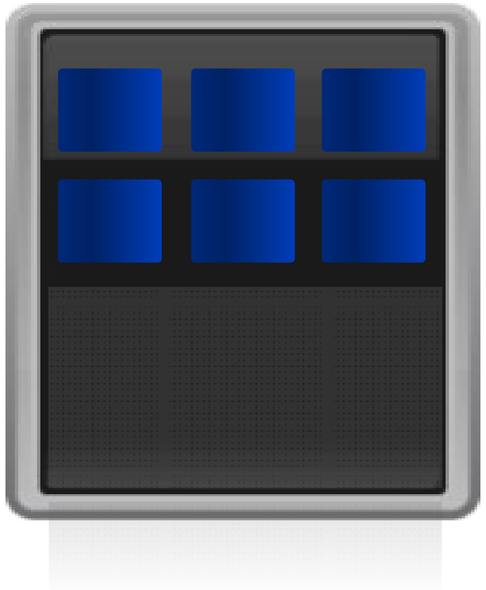
Still relies on CPU for managing and passing data

Programs must be rewritten to enable parallel processing

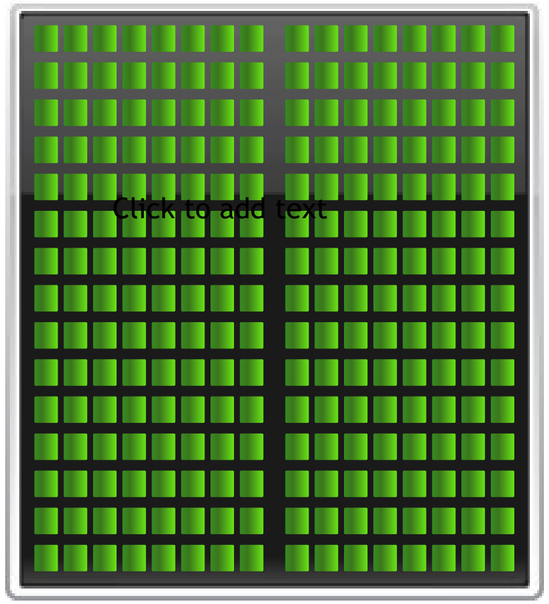
Accelerated Computing

10x Performance & 5x Energy Efficiency for HPC

CPU
Optimized for
Serial Tasks



GPU Accelerator
Optimized for
Parallel Tasks



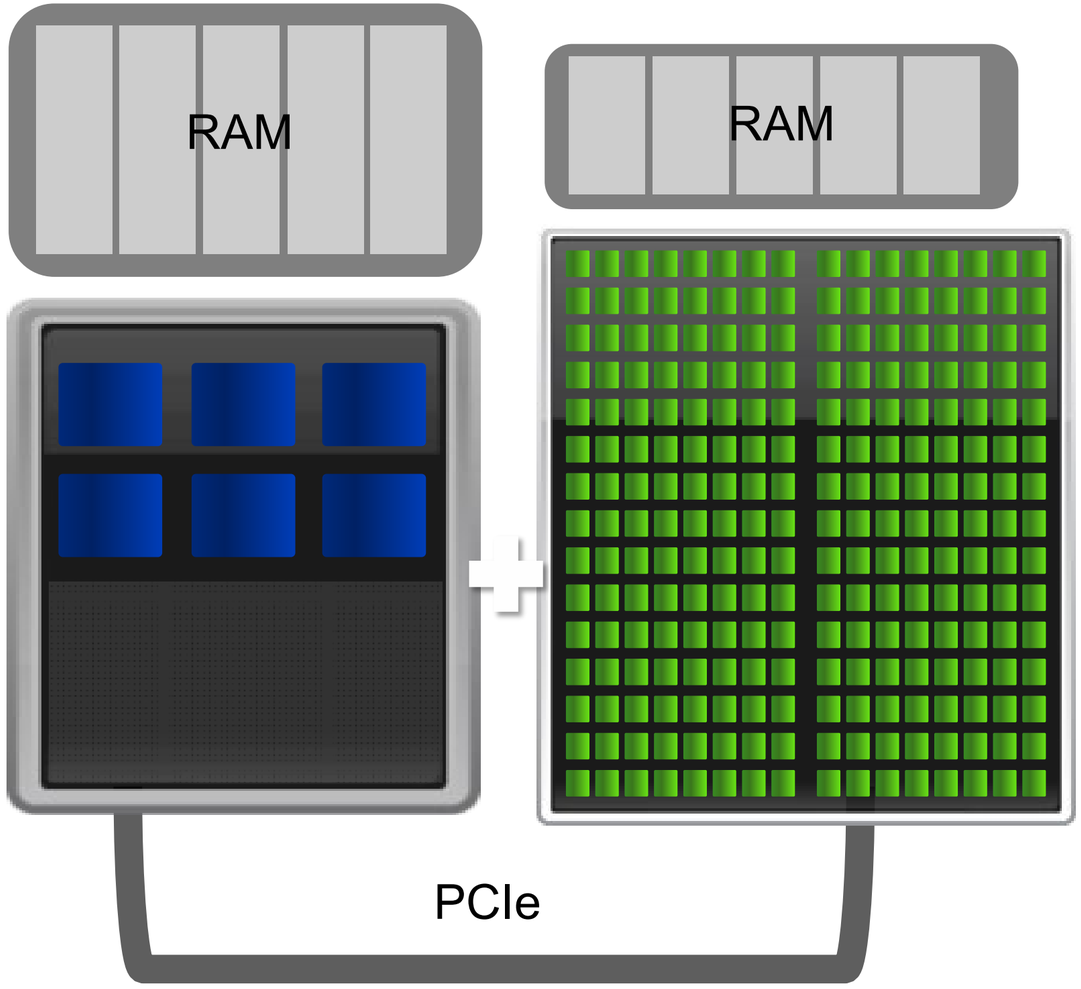
GPU Strengths

- High bandwidth main memory
- Significantly more compute resources
- Latency tolerant via parallelism
- High throughput
- High performance/watt

GPU Weaknesses

- Relatively low memory capacity
- Low per-thread performance

Accelerator Nodes



CPU and GPU have distinct memories

- CPU generally larger and slower
- GPU generally smaller and faster

CPU and GPU communicate via PCIe

- Data must be copied between these memories over PCIe
- PCIe Bandwidth is much lower than either memories

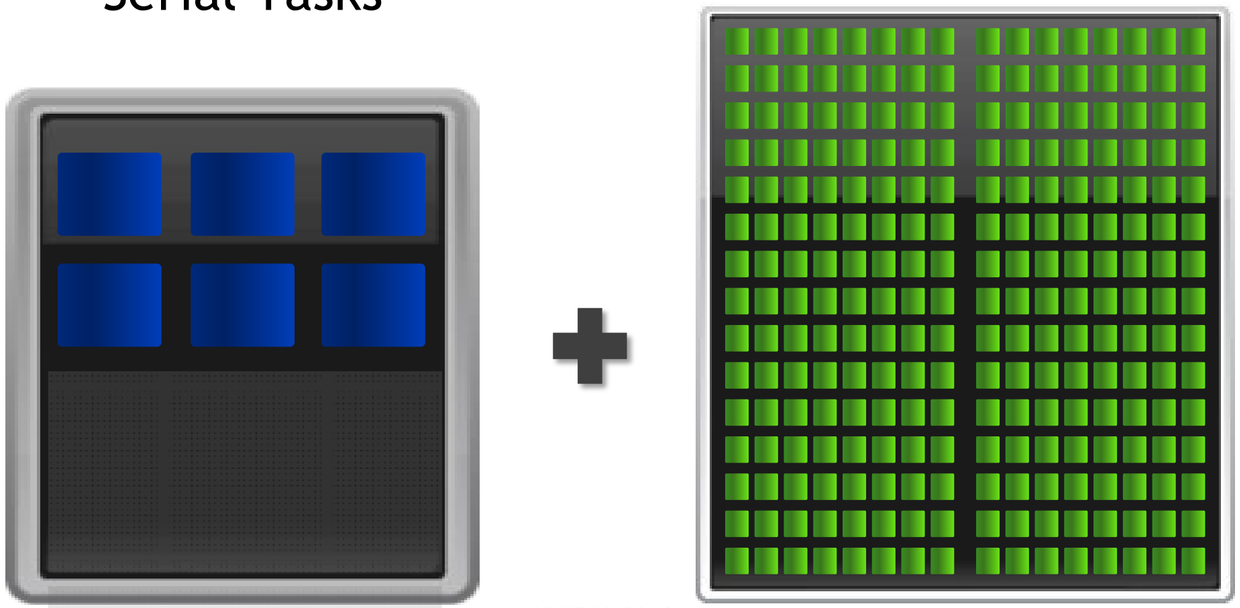
Emerging Tech - Nvlink

Accelerated Computing

10x Performance & 5x Energy Efficiency for HPC

CPU
Optimized for
Serial Tasks

GPU Accelerator
Optimized for
Parallel Tasks

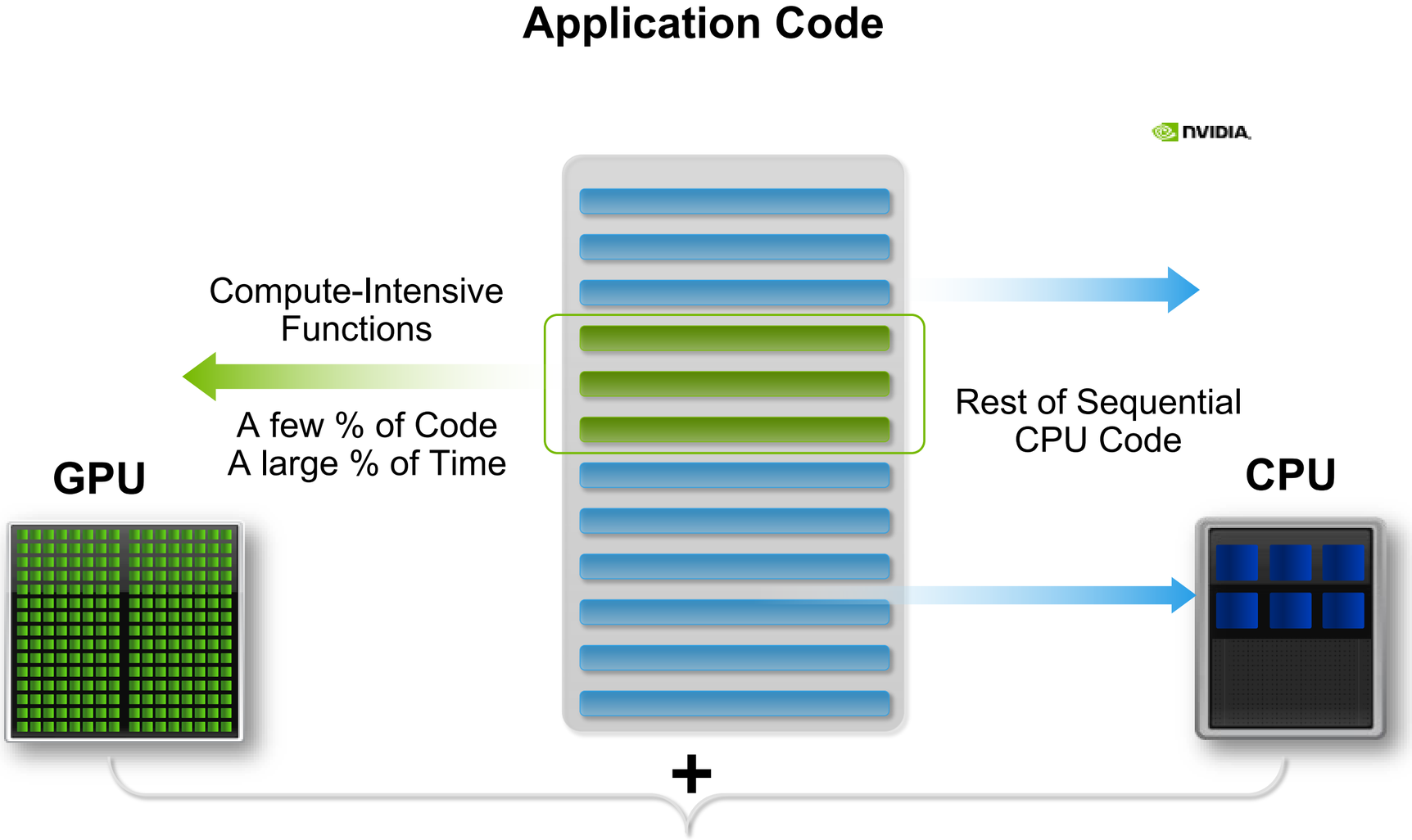


Slide adapted from NVIDIA Accelerated Computing Teaching Kit

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

What is Heterogeneous Programming?



Amdahl’s Law (*Tuomanen, 2018)

$$\text{Speedup} = \frac{1}{(1 - p) + p / N}$$

p= proportion of code that can be parallelizable

N= number of GPU cuda cores

*Hands-On GPU Programming with Python and CUDA

Speed vs Throughput

Speed

Throughput

*Images from Wikimedia Commons via Creative Commons

Which is better depends on your needs…

Edge AI and Robotics Teaching Kit

Thank You