12/11/2024

Why GPU?

Objective

- To get an overview of historical development of microprocessor
- To understand how GPU could scale computational power

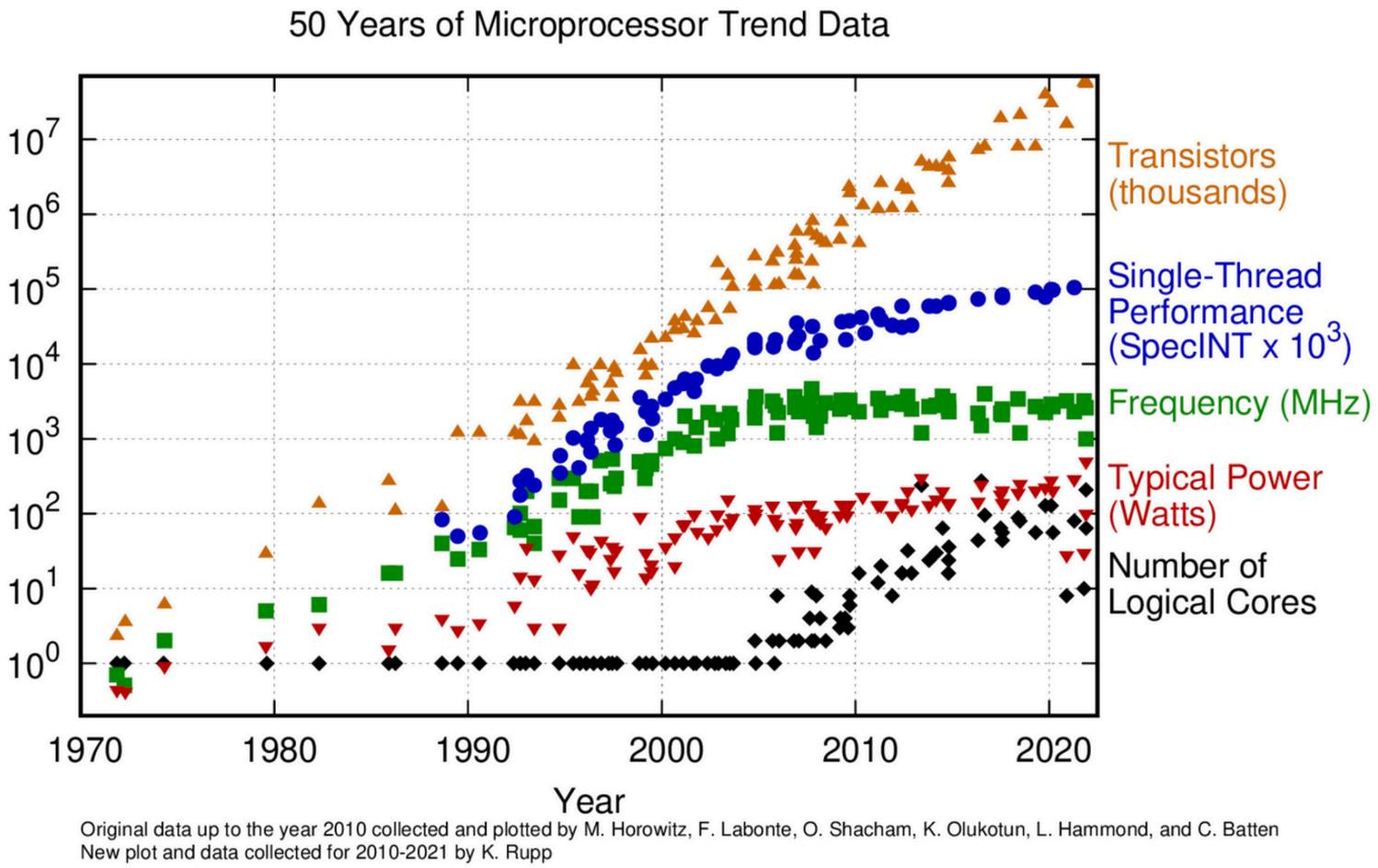
Moore’s Law

The number of transistors we can place on the integrated circuit doubles approximately every 2 years.

(observation and a forecast was made in 1965) ---- by Gordon E. Moore(cofounder of intel)

Every time the transistor density becomes double, it becomes faster

Moore’s Law



In 1970 there were around 1000 transistors placed in an integrated chip whereas in 2010 there were nearly 1 billion transistors placed in a chip.

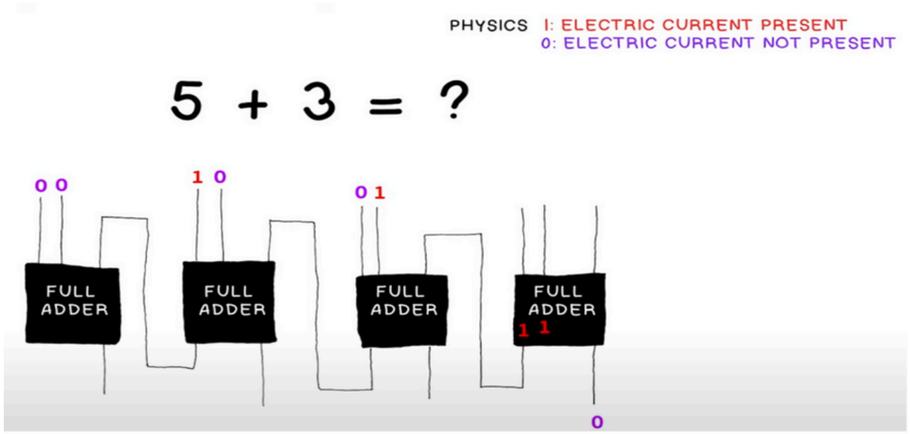
Why transistors?

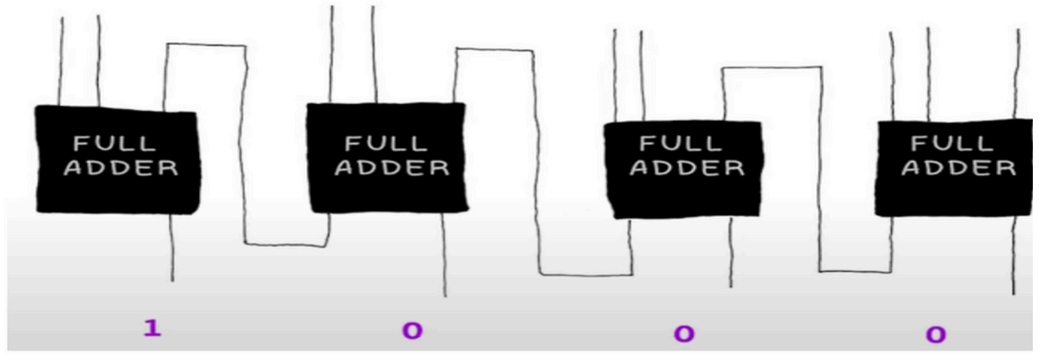
The reason behind computer being so powerful is transistors.Computer are built with number of chips called Integrated Circuits which is built out of huge number of transistors and extremely small conducting lines for electric current

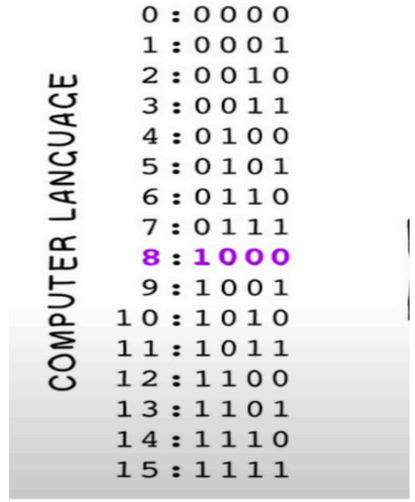
We can use these transistors or manipulate them to either let electric current flow through a specific conducting line or not.

Basic Operation

Basic addition of 5 and 3 as 8 which is represented by 1000 in the computer language.

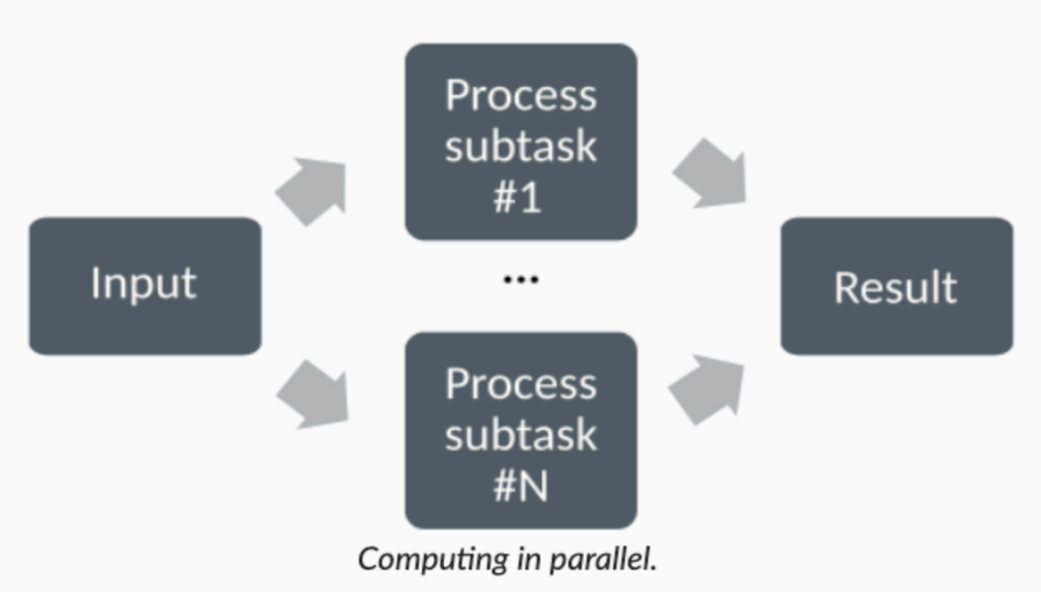






Computing In Parallel

Parallel computing involves dividing a computational problem into smaller, independent subtasks.These subtasks are processed simultaneously by multiple processing units to reduce the overall computation time.



Computing in parallel.

i.e. like assembling burgers in a fast-food restaurant.

Challenges in Parallel Processing

Data Dependency --- Tasks may depend on the results of others, causing delays.

Load Balancing --- Uneven distribution of tasks can lead to some processors being idle

Communication Overhead --- Transferring data between processors can be time-consuming.

Synchronization Issues --- Ensuring tasks are completed in the correct order can be complex.

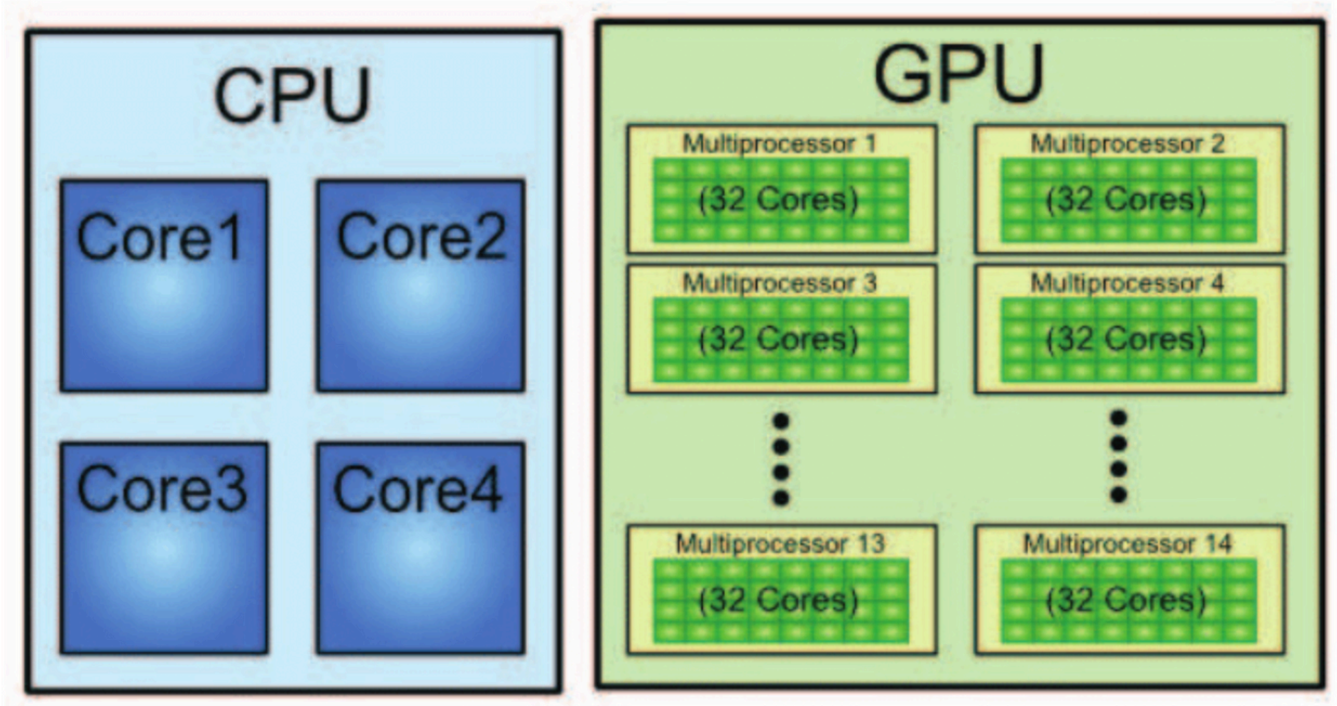
Scalability --- Not all problems scale well with more processors.

Graphics Processing Units(GPUs)

GPUs are specialized accelerators initially developed for graphic processing tasks.Over time, they've become essential in High-Performance Computing (HPC).

Fast Fourier Transform and Matrix Multiplication operations are highly optimized for execution on GPUs . These tasks leverage the parallel processing capabilities of GPUs, resulting in significant performance improvements over traditional CPU-based computations

Architecture



Typical CPU's architecture vs. a typical GPU's architecture (source http://blog.goldenhelix.com/)

Host Vs Device

GPU-enabled systems require a heterogeneous programming model that involves both CPU and GPU, where the CPU and its memory are referred to as the host, and the GPU and its memory as the device

Use case for CPU-GPU interaction

Developers write code for the CPU, which may include tasks for the GPU like rendering graphics or complex calculations.

The CPU sends commands and data to the GPU via the GPU driver, which manages their communication and task execution.

The GPU processes the commands, executing shaders or calculations as needed.

The GPU sends the output, like images or 3D models, back to the CPU through the GPU driver.

The CPU uses this output to update the screen or continue processing.

Example

if you are writing cuda program you will see a lot of instances where you will see code like this

// Copy data from host to device in CUDA, where d=device(GPU) and h=host(CPU)
cudaMemcpy(d_a, h_a, size, cudaMemcpyHostToDevice);
cudaMemcpy(d_b, h_b, size, cudaMemcpyHostToDevice);

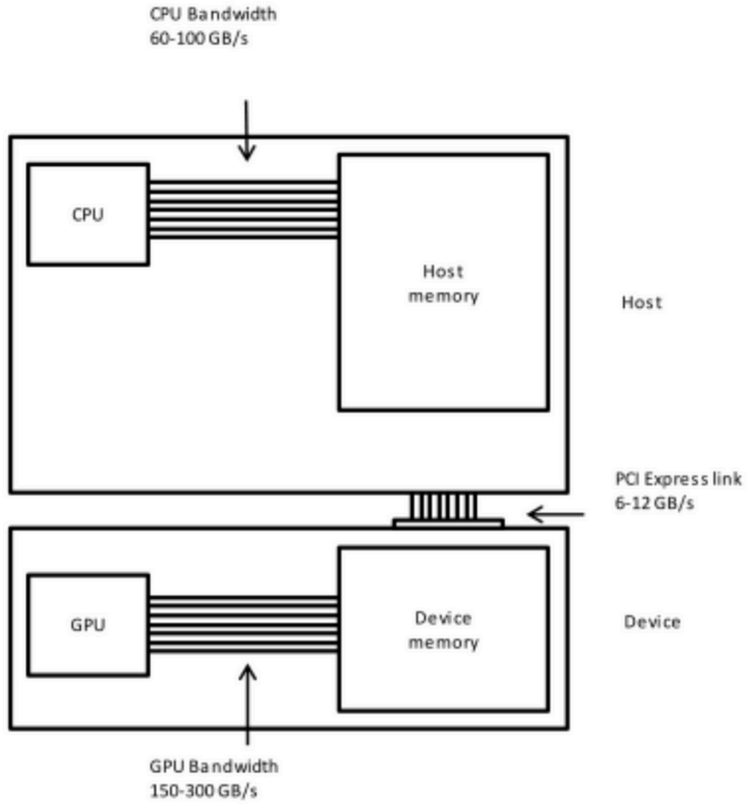


Figure adapted from the Carpentry GPU Programming lesson.

Host (CPU): Prepares data and manages the workflow, similar to the CPU tasks in the diagram.

Device (GPU): Executes parallel tasks, processing data simultaneously across multiple threads.

A look at top-500 list

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,699,904	1,206.00	1,714.81	22,786
2	**Aurora** - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel DOE/SC/Argonne National Laboratory United States	9,264,128	1,012.00	1,980.01	38,698
3	**Eagle** - Microsoft NDv5, Xeon Platinum 8480C 48C 2GHz, NVIDIA H100, NVIDIA Infiniband NDR, Microsoft Azure Microsoft Azure United States	2,073,600	561.20	846.84	
4	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
5	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	2,752,704	379.70	531.51	7,107

Snapshot from the TOP500 list from June, 2024.

Why GPUs?

Speed

GPU computing can significantly accelerate many types of scientific workloads.

Improved energy efficiency

Compared to CPUs, GPUs can perform more calculations per watt of power consumed, which can result in significant energy savings

Cost-effectiveness

GPUs can be more cost-effective than traditional CPU-based systems for certain workloads.

Limitations

1. Only for certain workloads – Not all workloads can be efficiently parallelized and accelerated on GPUs. Certain types of workloads, such as those with irregular data access patterns or high branching behavior, may not see significant performance improvements on GPUs.

2. Steeper learning curve – GPU computing could require specialized skills in GPU programming and knowledge of GPU architecture, leading to a steeper learning curve compared to CPU programming.

Key takeaways

GPUs are accelerators for some types of tasks

Highly parallilizable compute-intensive tasks are suitable for GPUs

New programming skills are needed to use GPUs efficiently