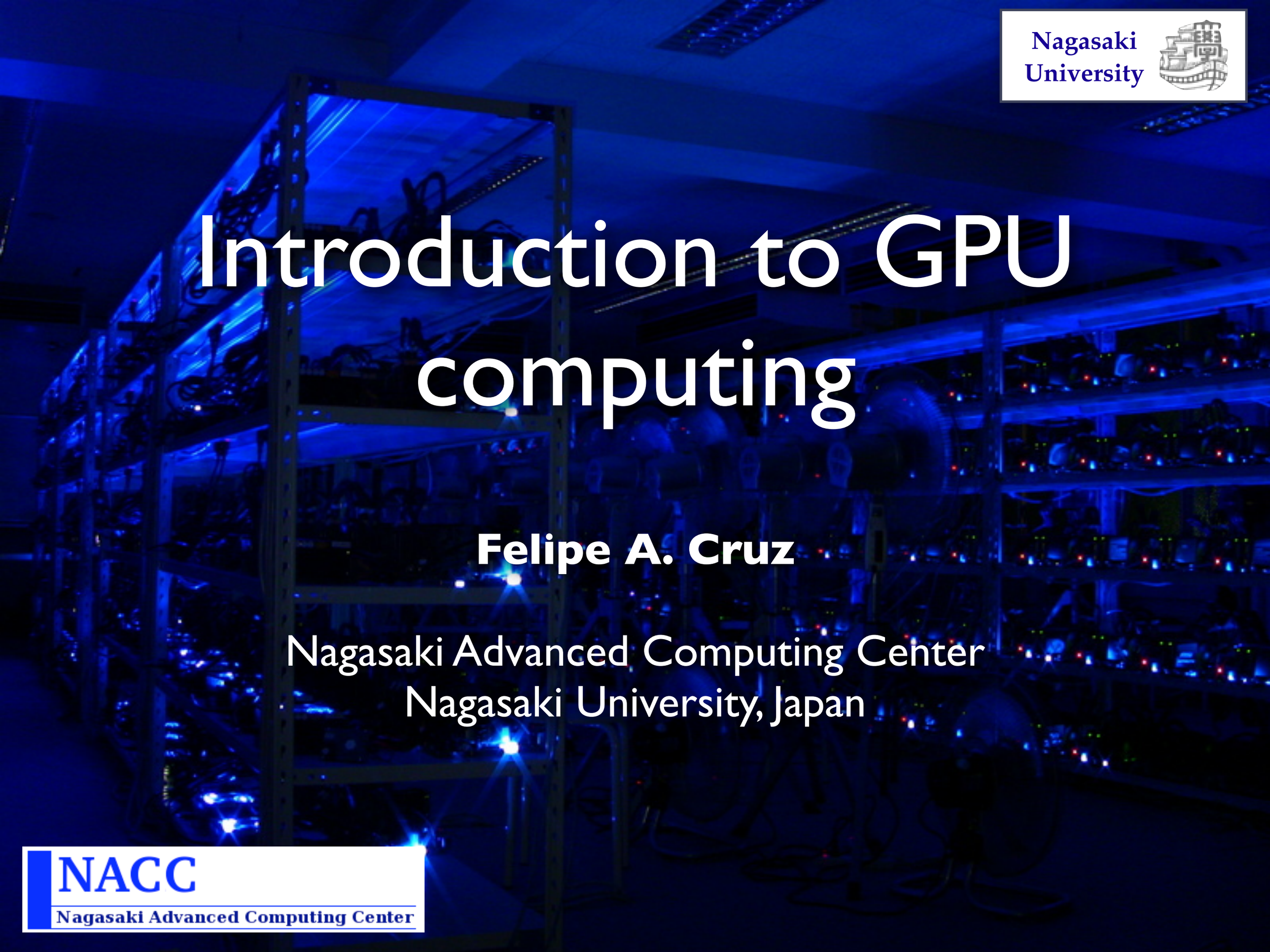
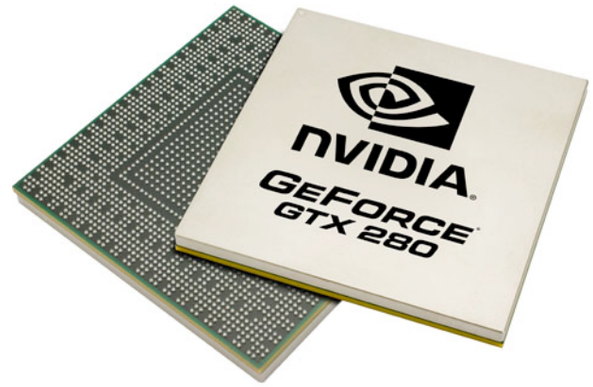


Introduction to GPU computing

Felipe A. Cruz

Nagasaki Advanced Computing Center
Nagasaki University, Japan

The **GPU** evolution



- The **Graphic Processing Unit** (**GPU**) is a processor that was **specialized** for processing graphics.

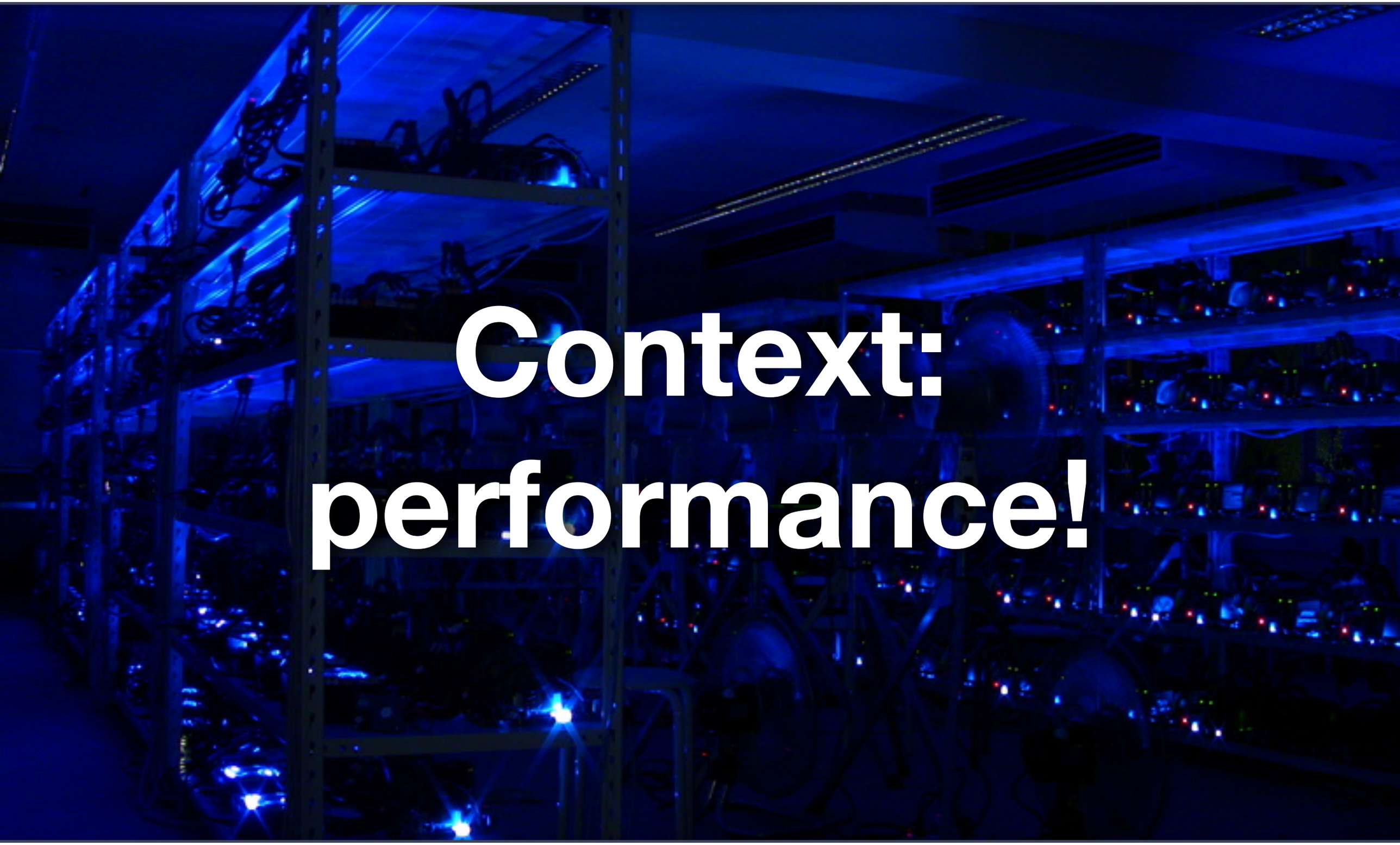
- The GPU has recently **evolved** towards a **more flexible** architecture.

- **Opportunity**: We can implement ***any algorithm***, not only graphics.

- **Challenge**: obtain **efficiency** and **high performance**.

Tesla card

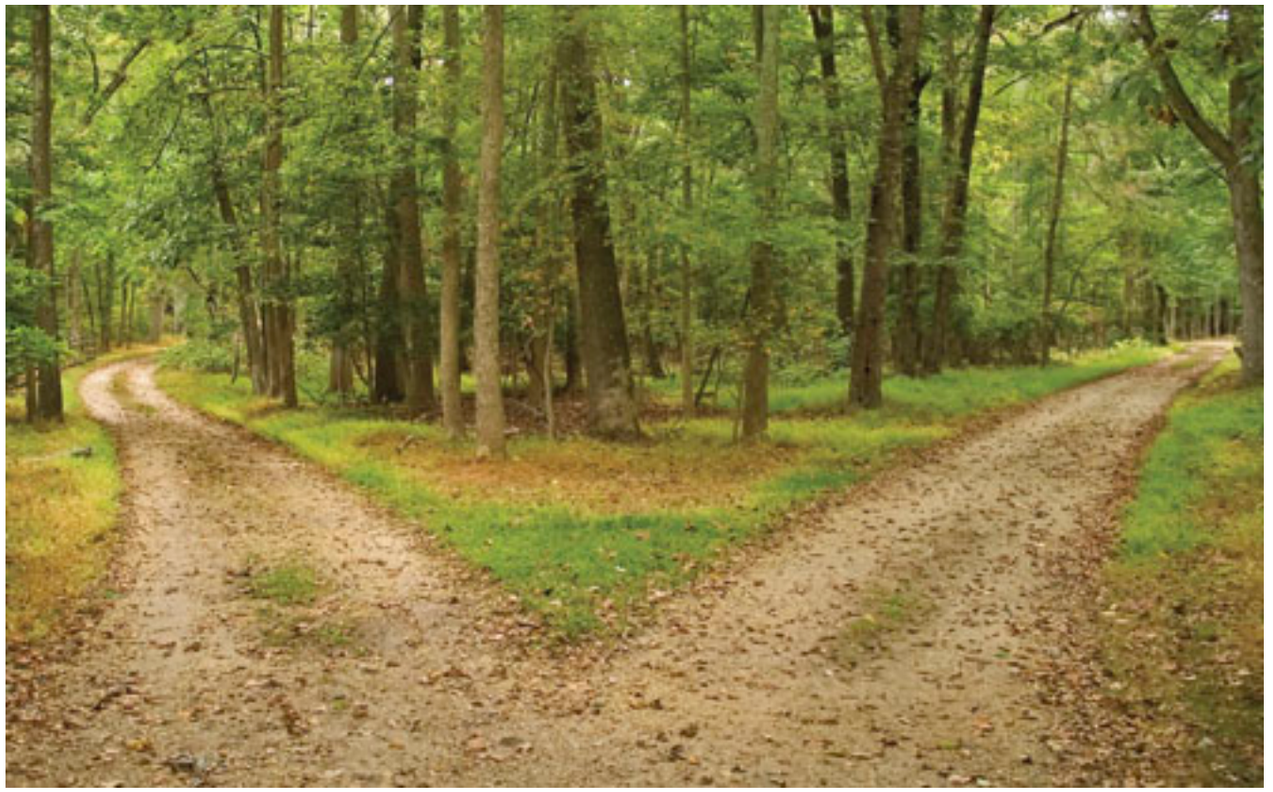
Tesla S1070: 4 cards



Context: performance!

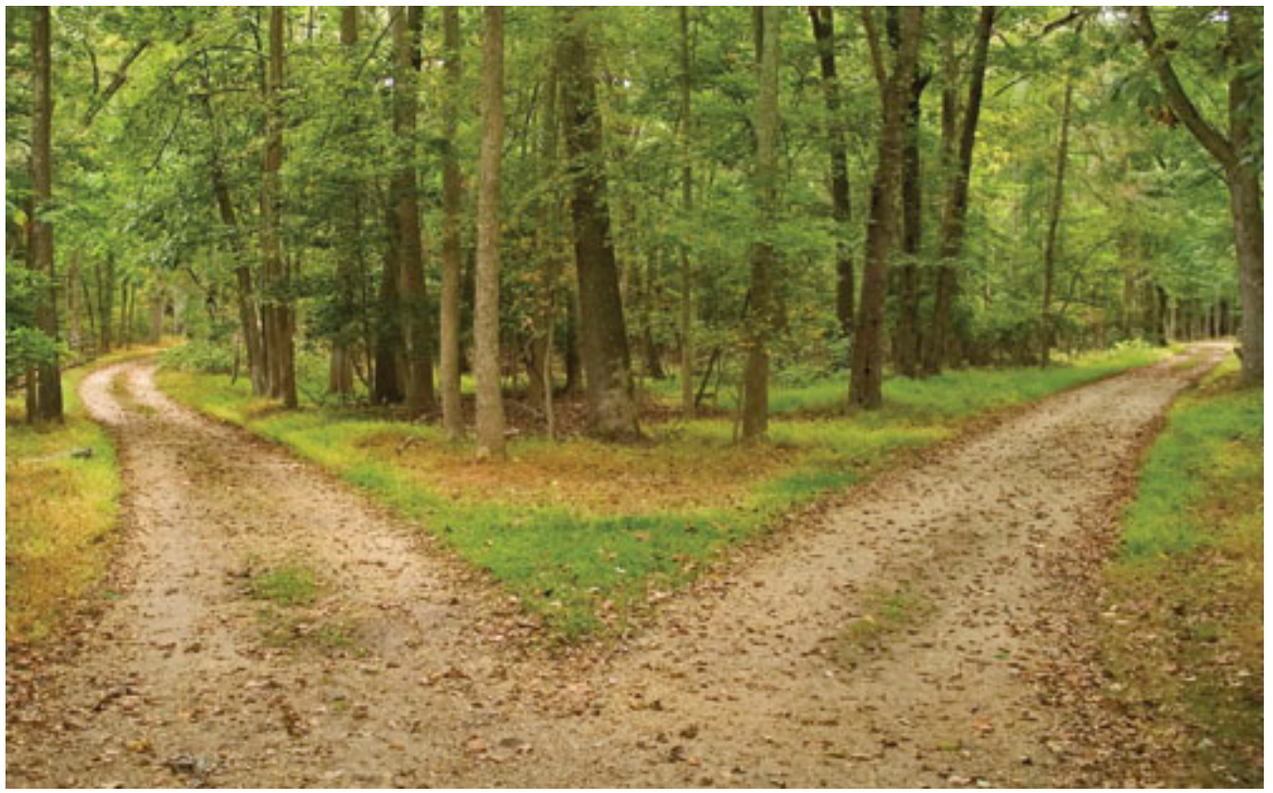
Felipe A. Cruz

Paths to performance



Paths to performance

Faster processors

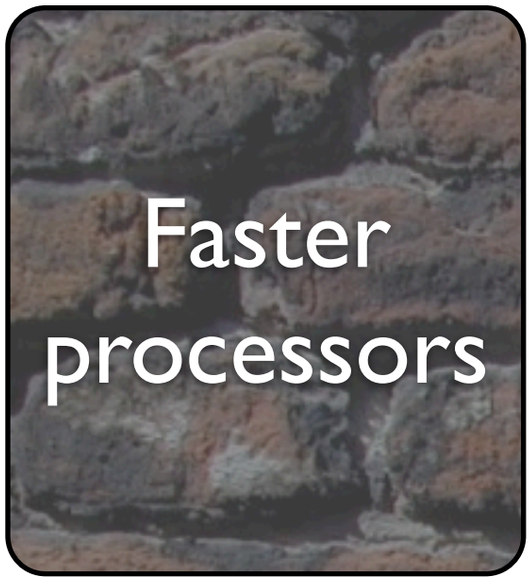


Important method for **improving performance**.

Processors speed improved from **1 MHz** (1980s) to over **4GHz** (2000s)

Over **1,000 fold** faster clock rates in **30 years**!

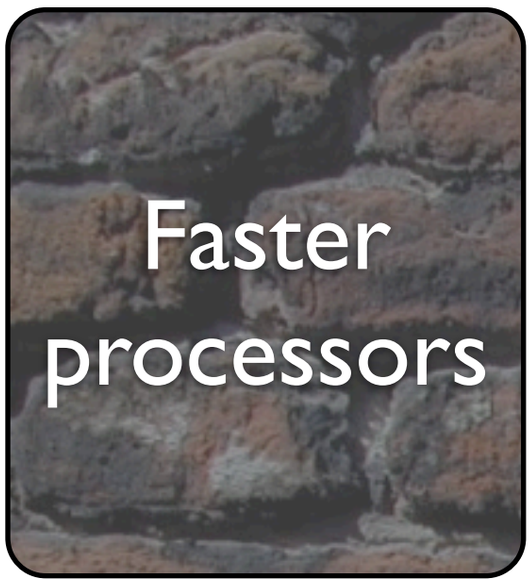
Paths to performance



Memory wall: large gap between memory and processor speed.

Power wall: higher clock speeds require exponential power increase.

Paths to performance

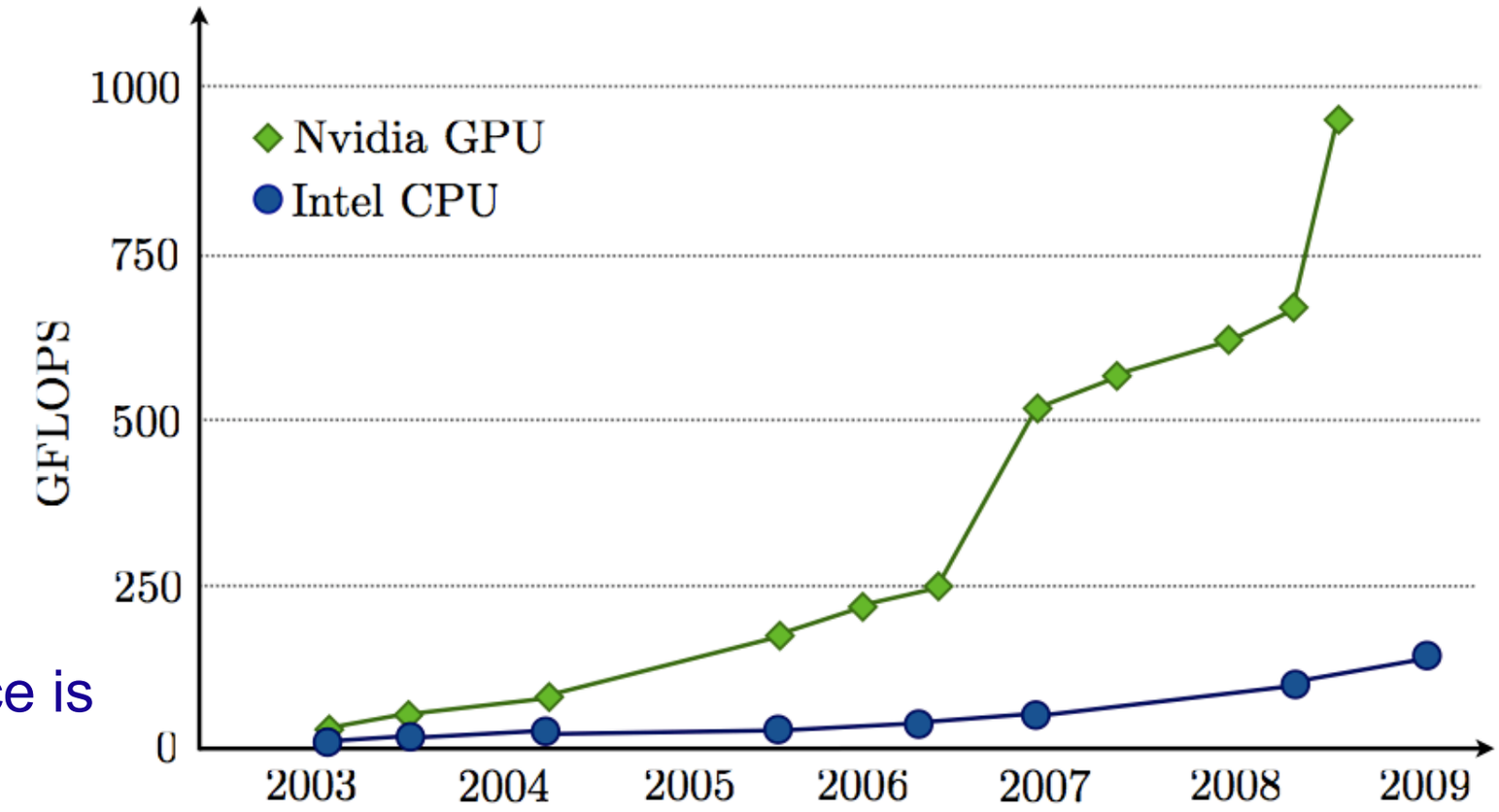


Processors with many components that ***may execute*** in parallel.

Benefit from ***aggregated performance***.

Paths to performance

- There are two ways to obtain a fast computing system:
  - Fast processor (**CPU**)
  - Use concurrency (**GPU**)
- For both, CPU and GPU, serial performance is not increasing.
- Multicore and GPU performance is based on concurrency...
- GPU many times **faster** than CPU!



NACC
Nagasaki Advanced Computing Center

Nagasaki University

After the buzz...

Felipe A. Cruz

Why accelerators?

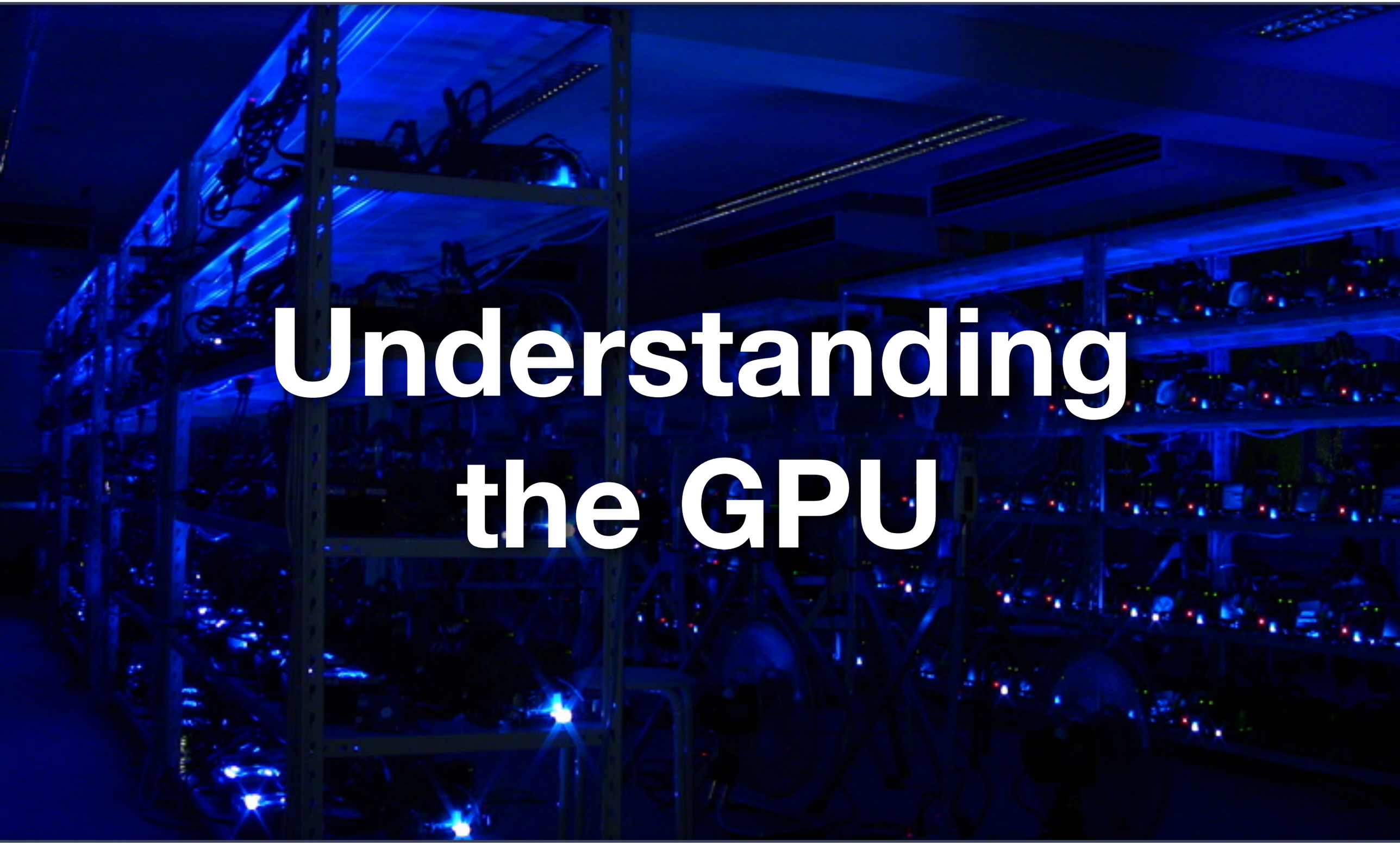
- Accelerators have been around since the 70s! but they were very expensive...
- GPUs are **commodity** hardware, powerful, and cheap.
- New GPU generation are released every **two years**.
- **Uni-processor speed is not doubling** every two years anymore!
- GPUs are competitive alternatives in the search for performance.

Can *I* get 100x?

- ***Some*** algorithms **can** get such speedups.
- It mostly **depends** on the **non-parallel** part (the one that does not accelerate with a GPU!).
- **Complex applications** make use of many algorithms!
- **Redesign** your application to be parallel-friendly is most important!
- The **future is parallel**. Let's **get ready**!

GPGPU = general purpose?

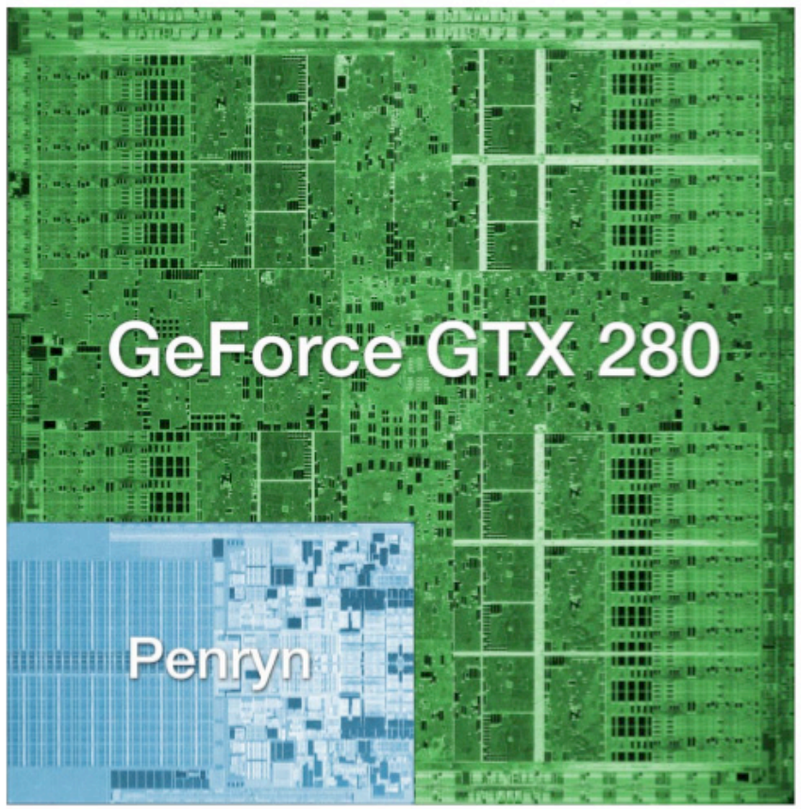
- **Technically**: yes, you can program anything!
- Practically: we want **performance**...
- GPUs are **specialized hardware**: **not flexible** (as a CPU)
- **CPU + GPU** = combination of flexible and performance.



Understanding the GPU

Felipe A. Cruz

What is a GPU?



- **G**raphics **P**rocessing **U**nit (GPU)
- Most computers have one
- **Simple** and **regular** design!
- **Billions** of transistors
- Performance:
  - **~1 Teraflop** (Single precision)
  - **~500 Gflops** (Double precision)
- Also: A **heater** for winter!

How it works:

Many scooters

Sport car

How it works:

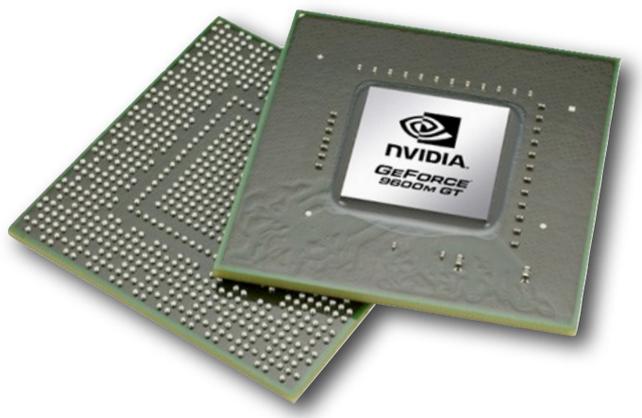
Many scooters

Deliver many packages
within a reasonable timescale

Sport car

Deliver a package as
soon as possible

How it works:



High **throughput**
and
reasonable **latency**

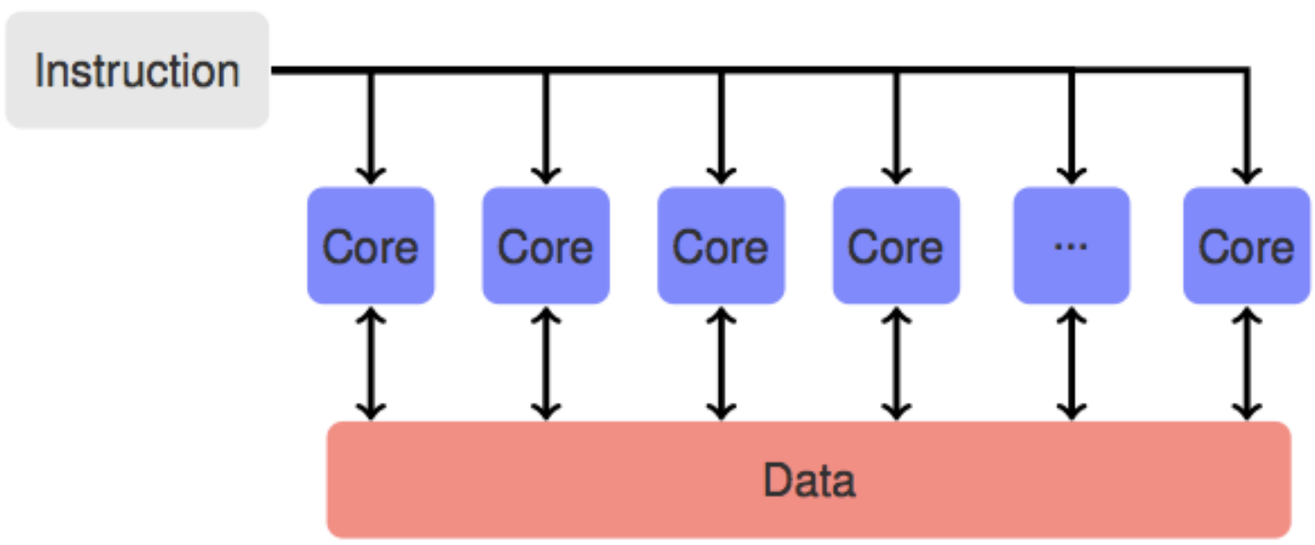
Compute many jobs
within a reasonable timescale

Low latency
and
reasonable throughput

Compute a job as
fast as possible

What are “GPU cores”?

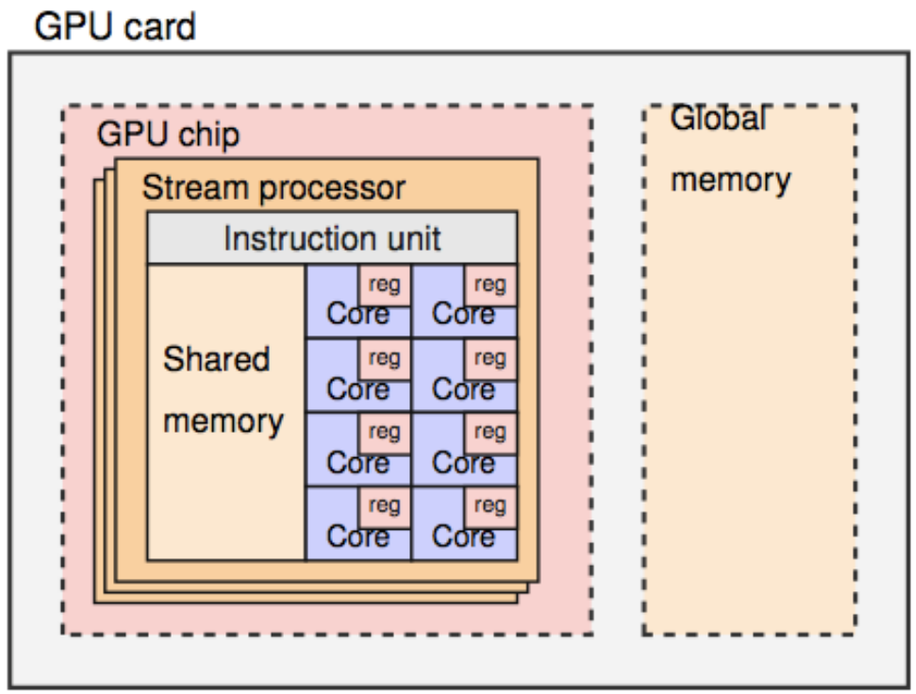
- Not the same than a CPU core!
- Basic component on a GPU is a ‘**stream processor**’, or SIMD processor (**S**ingle **I**nstruction **M**ultiple **D**ata)



SIMD: Same instruction for all cores! but each can operate over different data!

GPU architecture

Sketch of the GPU architecture:



The GPU spec:

- **~1 Teraflop** single precision
- **~.5 Teraflop** double precision
- Shared memory: **16 KB to 48 KB**
- L1 cache: **16 KB to 48 KB**
- *1 Teraflop = 10^12 floating point operations

Obtain **performance by concurrency**!
MIMD: Independent stream-processors.

GPU issues

- Massively parallel algorithm needed!
  - high concurrency and simple data pattern
- Balance computation and data movement!
  - Core is much faster than data-path
  - Keep the GPU busy!
- Latency of data movement!
  - Multi-threading

Obtaining performance

- **Design** algorithms **for parallelism**:
  - fine grained and coarse grained!
- **Map** algorithm **to the architecture**:
  - **SIMD** at stream processor.
  - **MIMD** at GPU chip.
  - Consider **memory hierarchy**.
  - Others: **heterogeneous**, **distributed systems...**