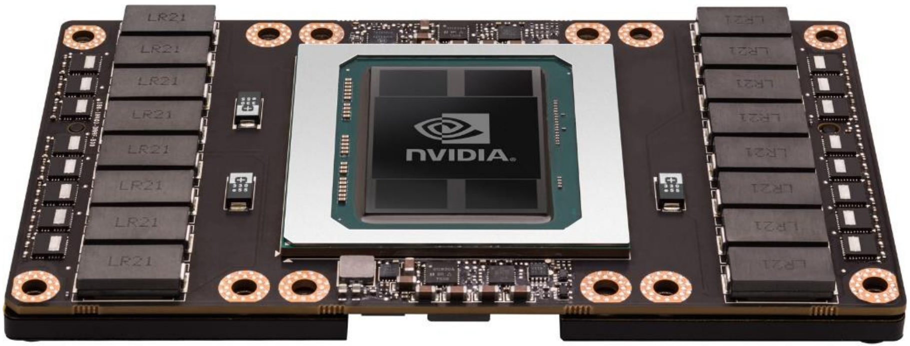
Inception and Evolution of GPUs

A Brief Story of Graphics Processing Units



Albert García García

agarcia @ dtic.ua.es

NVIDIA 3DPERCEPTIONLAB

Contents

Introduction

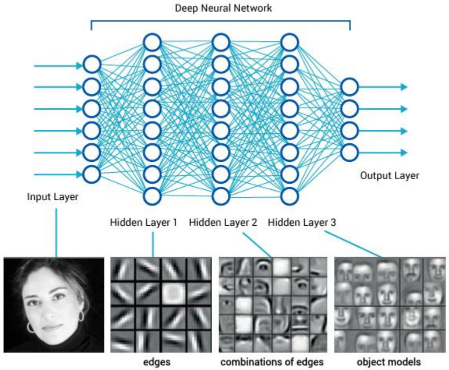
Introduction

Need for compute power

AR/VR

Self-Driving Cars

Deep Learning



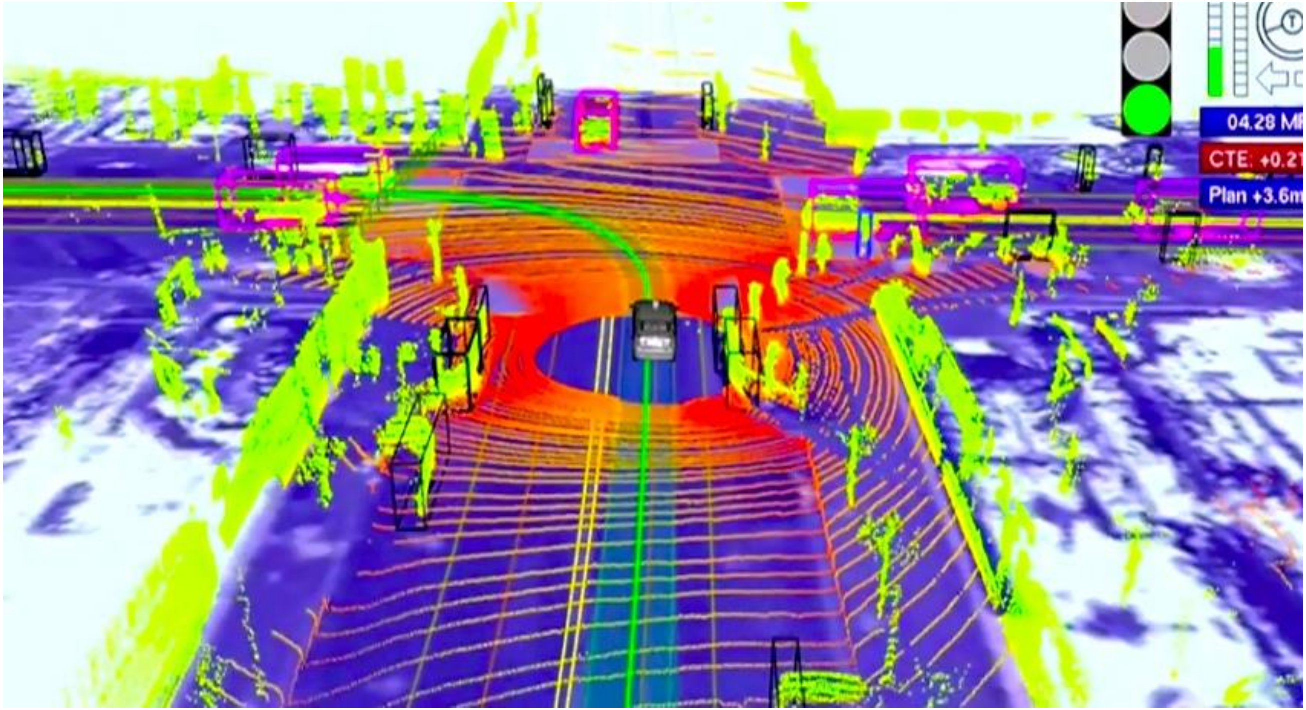
Introduction

Augmented Reality (AR)

Introduction
Virtual Reality (VR)

Introduction

Self-Driving Cars

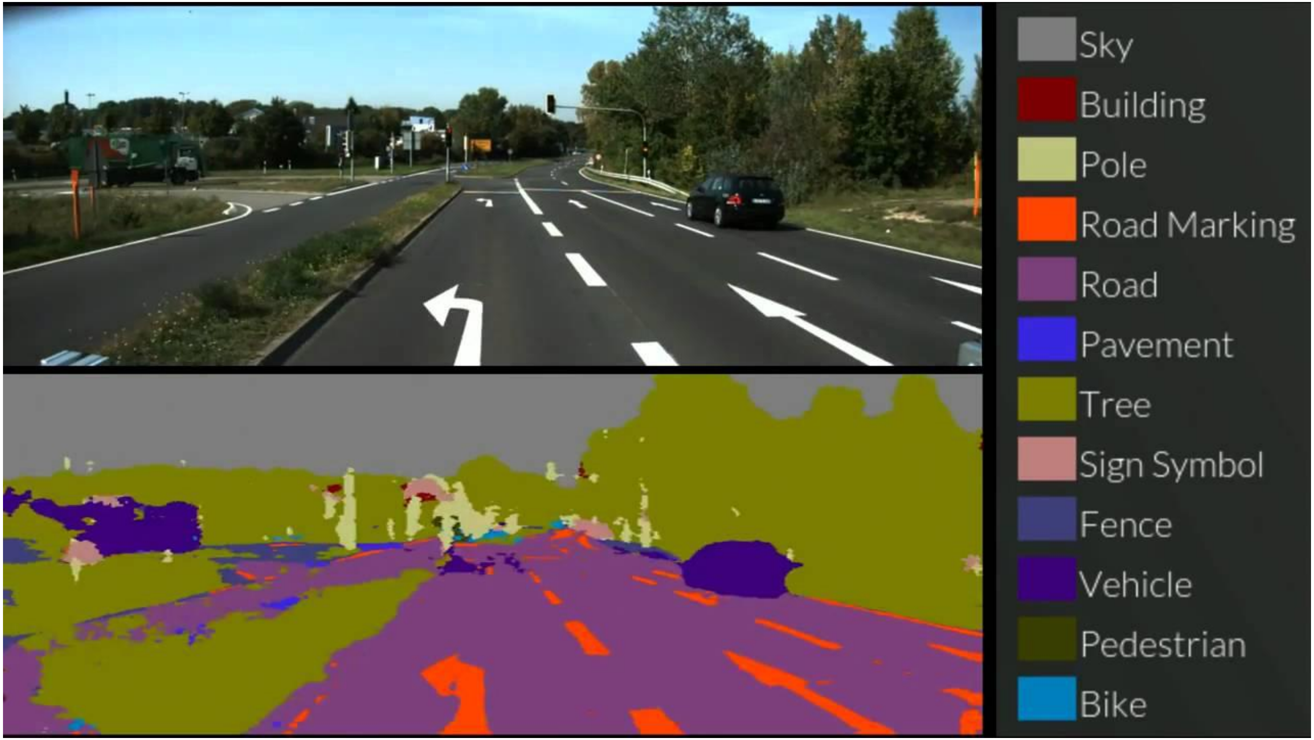


Introduction

Self-Driving Cars

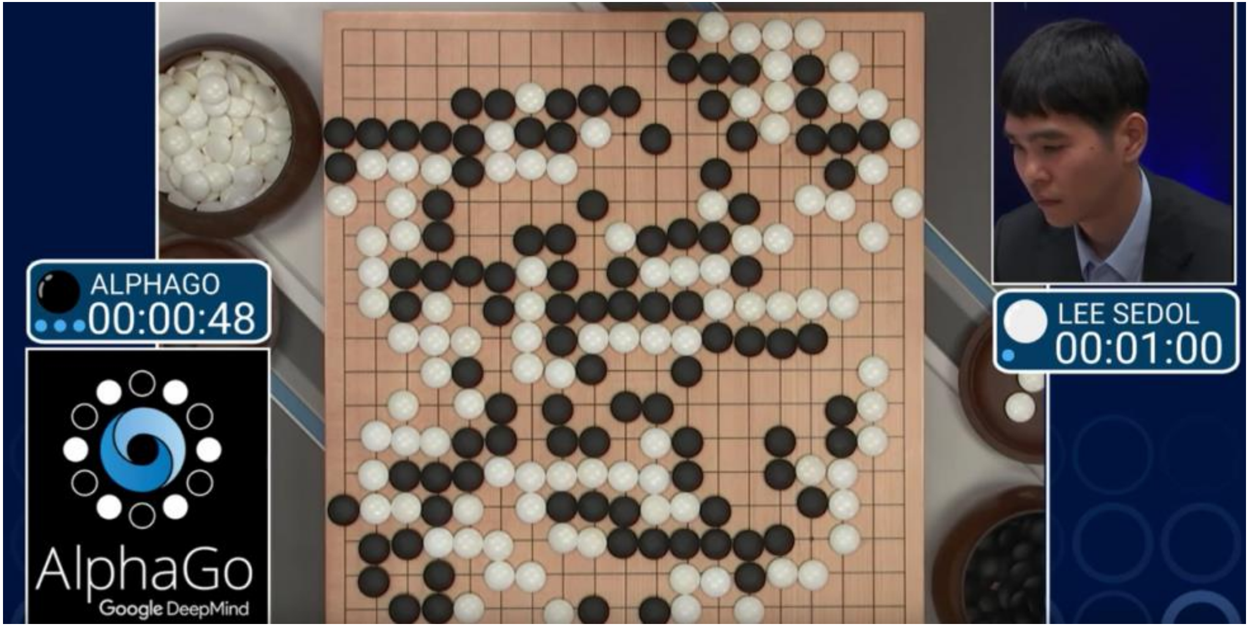
Introduction

Self-Driving Cars



Introduction

Deep Learning



Introduction

I am AI

https://www.youtube.com/watch?v=GiZ7kyrwZGQ

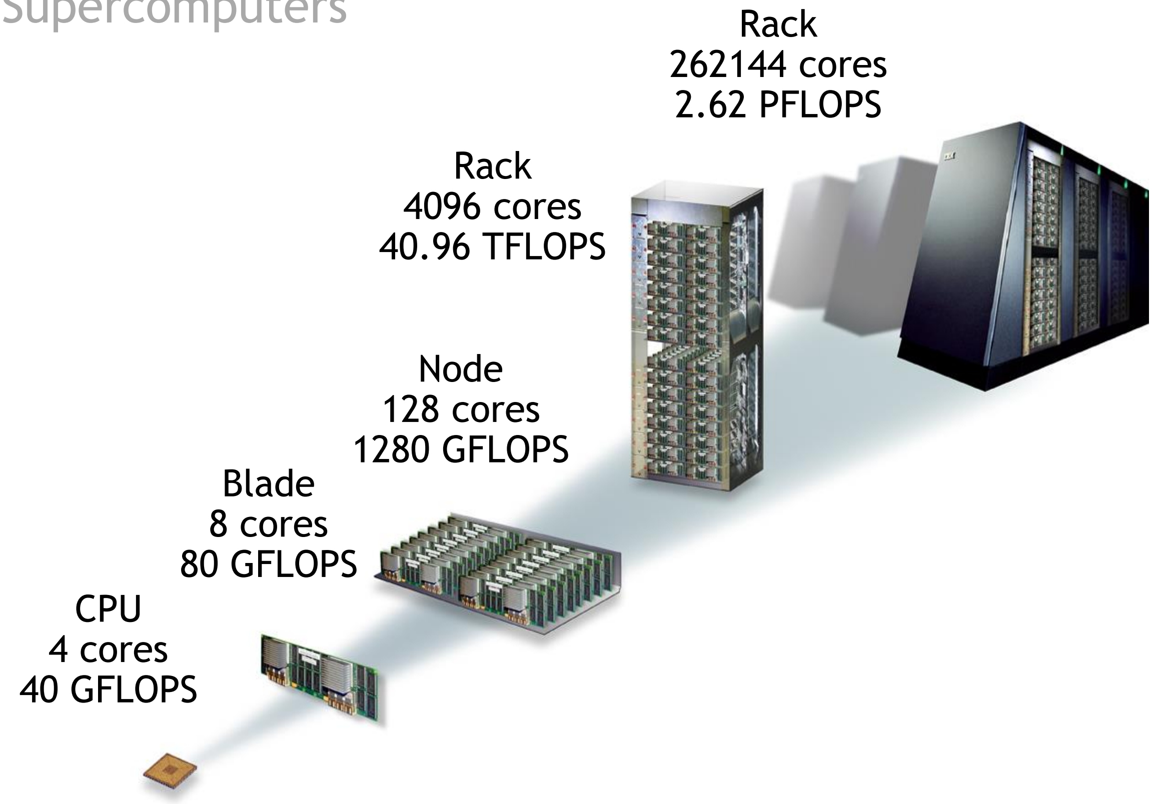
Introduction

How to handle all that data?

If you were to plow a field, what would you choose?

A) One big, fast, and cutting-edge tractor
B) A thousand chickens pecking on the ground

Introduction
Supercomputers



Introduction

Supercomputer performance

- Most used metric is FLOPS (Floating Point Operations Per Second):
  - MFLOPS (MegaFLOPS) = 1 000 000 FLOPS
  - GFLOPS (Giga FLOPS) = 1 000 000 000 FLOPS
  - TFLOPS (Tera FLOPS) = 1 000 000 000 000 FLOPS
  - PFLOPS (Peta FLOPS) = 1 000 000 000 000 000 FLOPS
  - EFLOPS (Exa FLOPS) = 1 000 000 000 000 000 000 FLOPS
- How many FLOPS for the most powerful supercomputer on Earth?
- How much power does that supercomputer draw?

Introduction

Supercomputer performance (TOP500)

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National Supercomputing Center in Wuxi China	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway NRCPC	10,649,600	93,014.6	125,435.9	15,371
2	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3,120,000	33,862.7	54,902.4	17,808
3	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560,640	17,590.0	27,112.5	8,209
4	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1,572,864	17,173.2	20,132.7	7,890

Introduction
MareNostrum 4

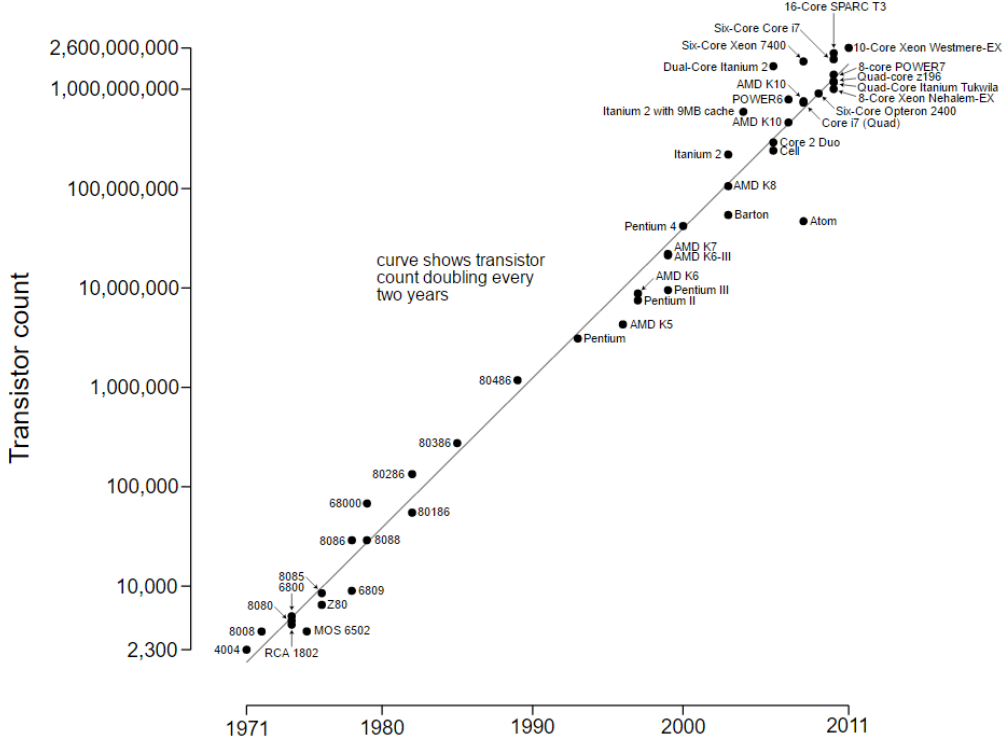
Introduction

GPUs or massive parallelism for all

Moore’s Law

Moore's Law

Approximately, every two years, transistor count doubles

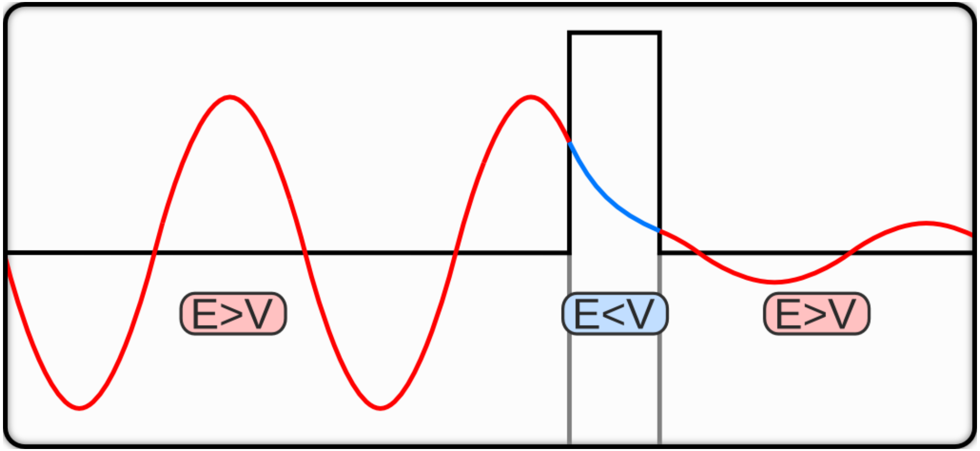


NVIDIA

3DPERCEPTIONLAB

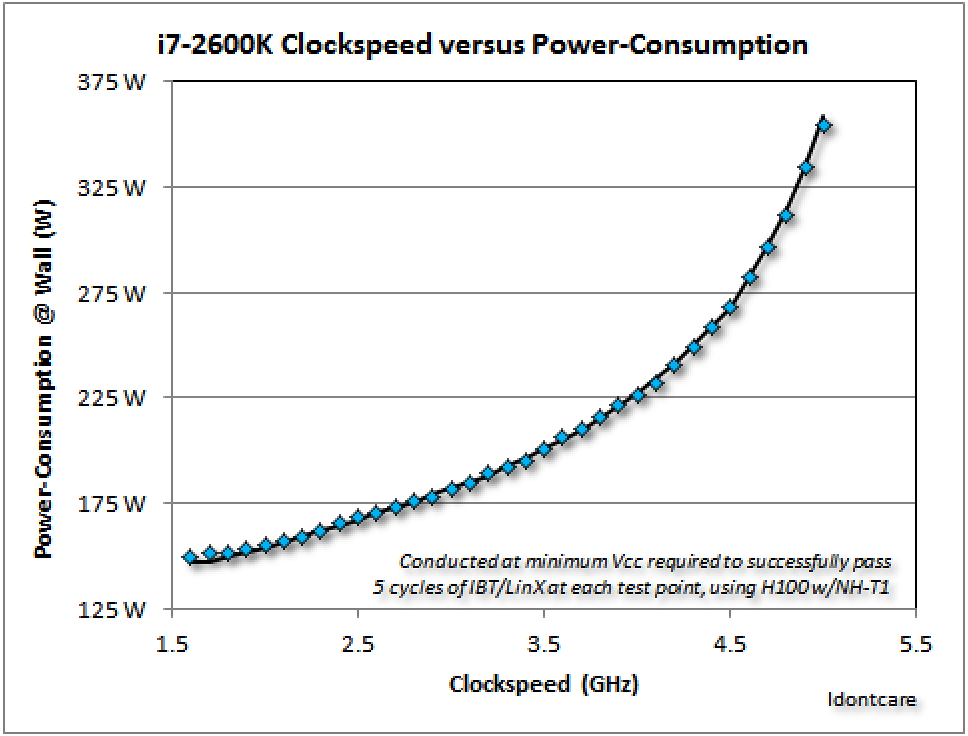
Moore’s Law

Problem: transistor size



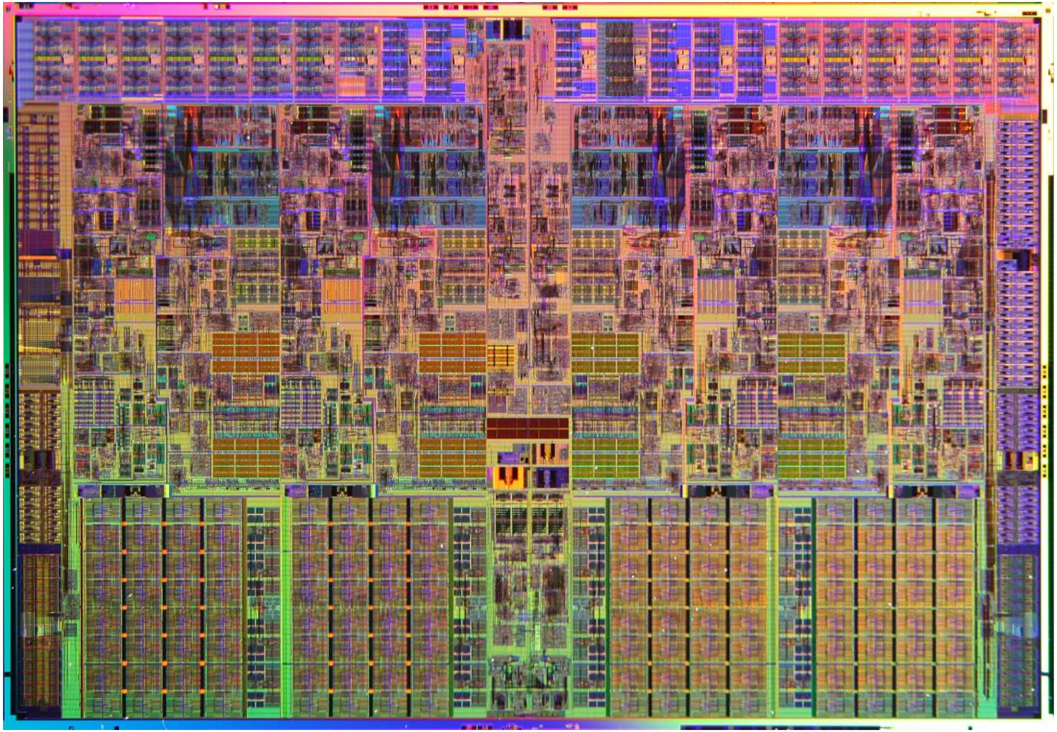
Moore’s Law

Problem: power draw



Moore's Law

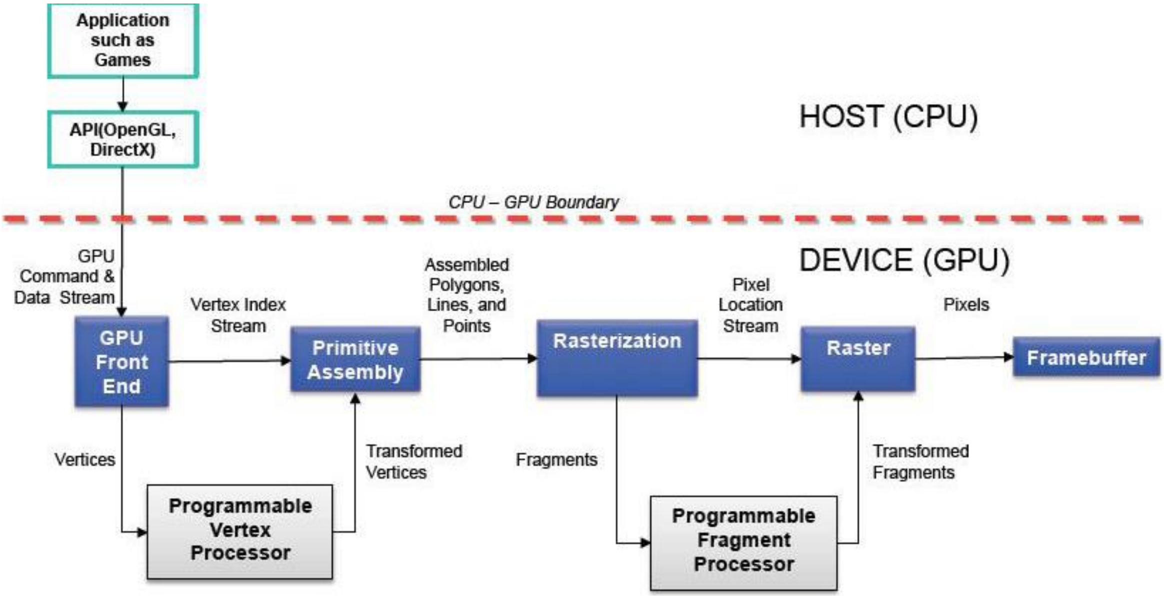
How to keep scaling? Multicore



The Graphics Pipeline

The Graphics Pipeline

OpenGL / DirectX



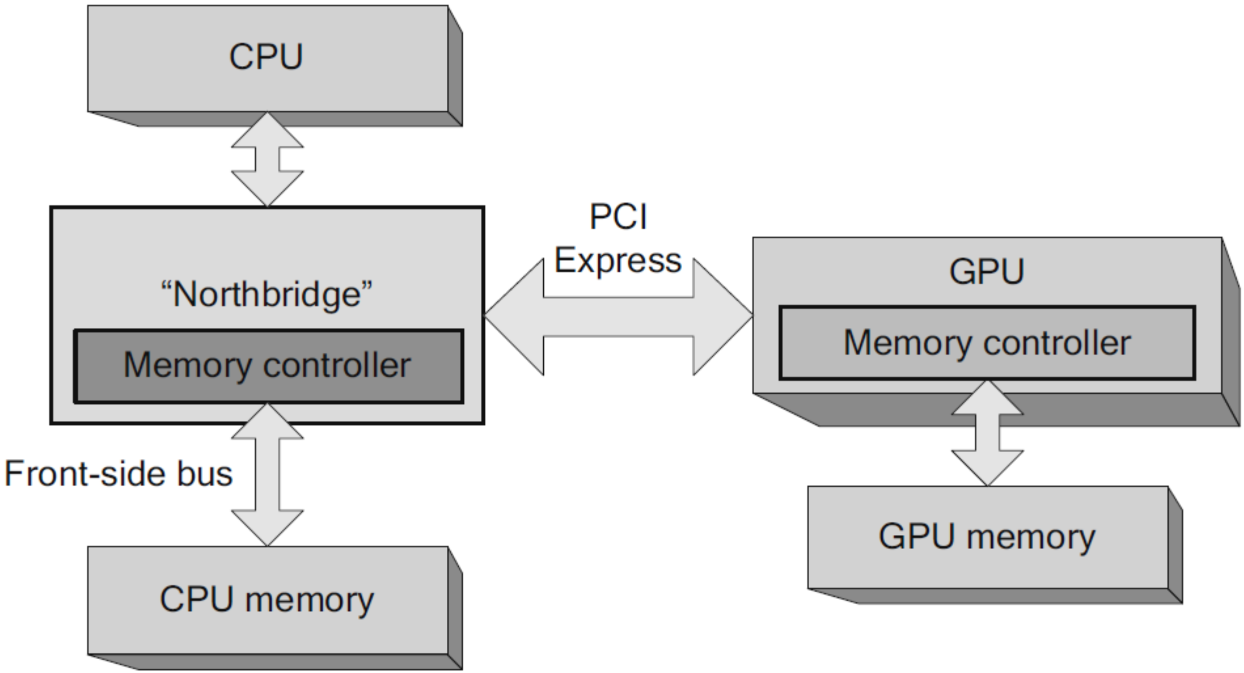
The Graphics Processing Unit

The Graphics Processing Unit (GPU)

It all started with the GeForce 256

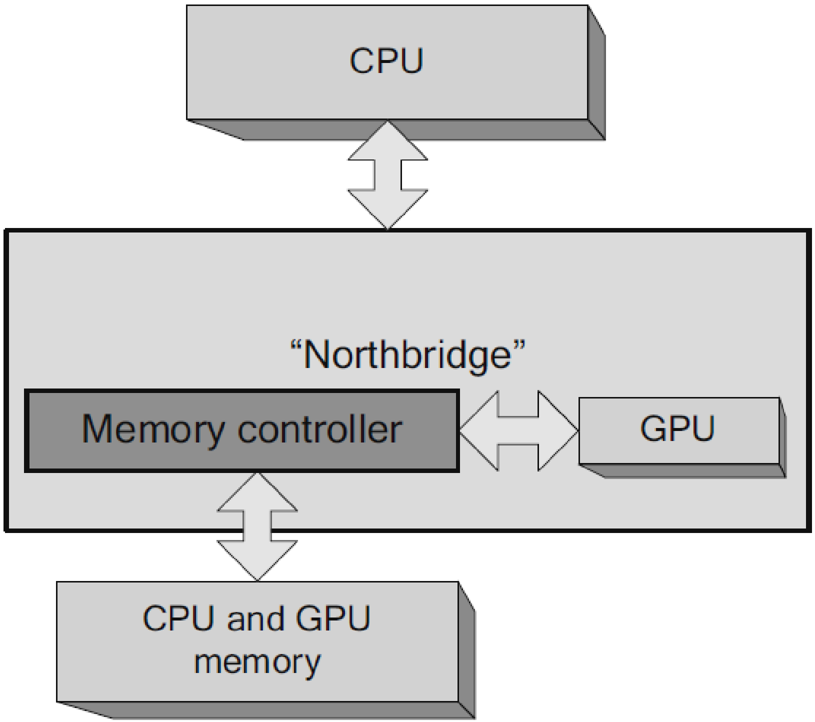
The Graphics Processing Unit (GPU)

Typical CPU / GPU system



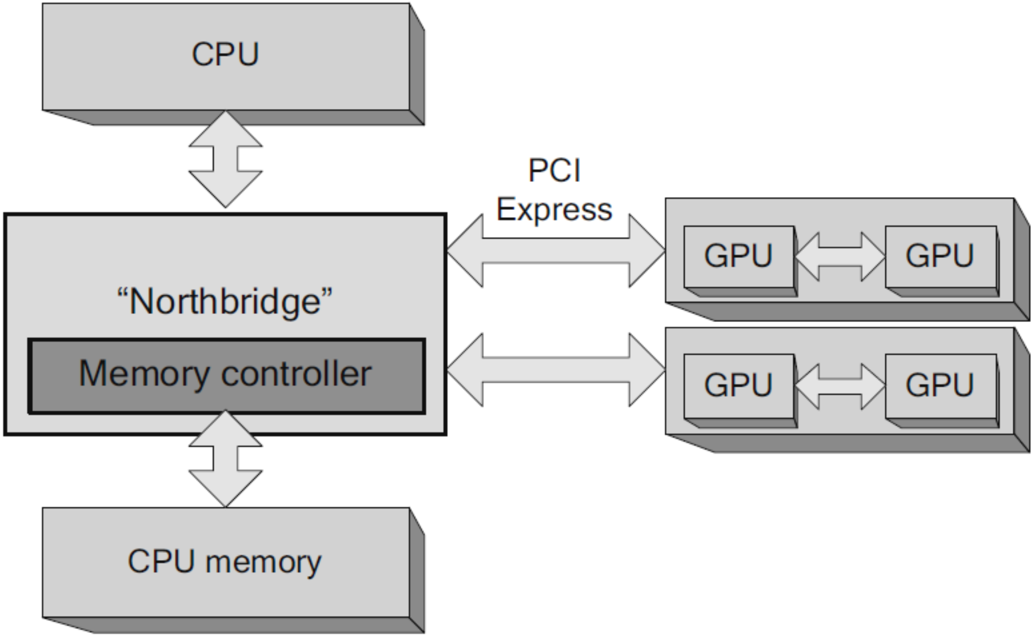
The Graphics Processing Unit (GPU)

Integrated CPU / GPU system



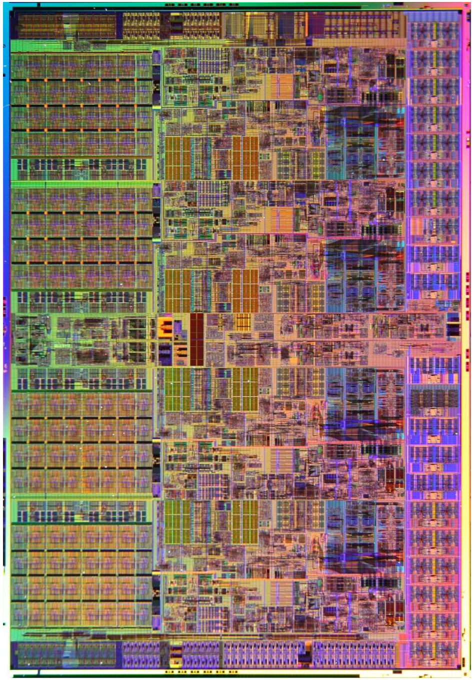
The Graphics Processing Unit (GPU)

CPU / multi-GPU system

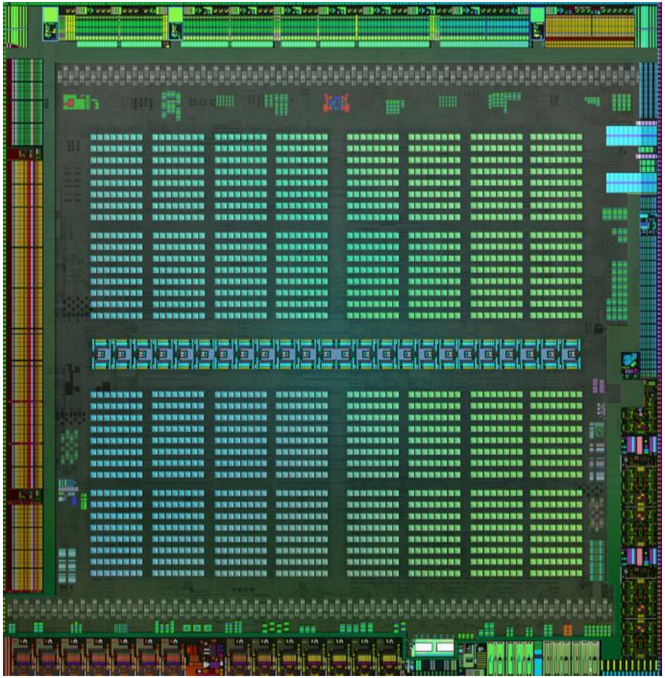


The Graphics Processing Unit (GPU)

The essence of the GPU at a glance



Intel i7

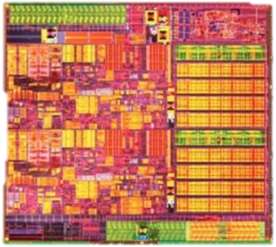


GTX 480

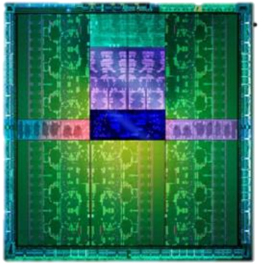
The Graphics Processing Unit (GPU)

CPU vs. GPU (different goals, different efficiency)

CPU	GPU
Optimized for latency	Optimized for throughput
Complex caching	“Simple” caching
Complex control	Simple control
1690 pJ / FLOP	140 pJ / FLOP



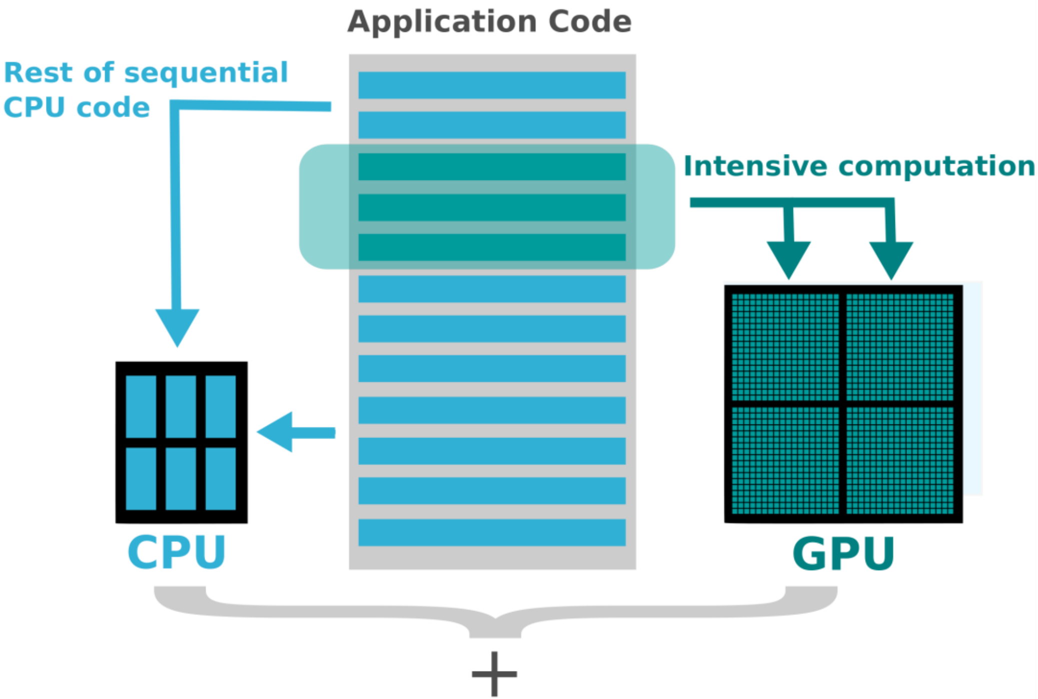
Westmere
32 nm



Kepler
28 nm

The Graphics Processing Unit (GPU)

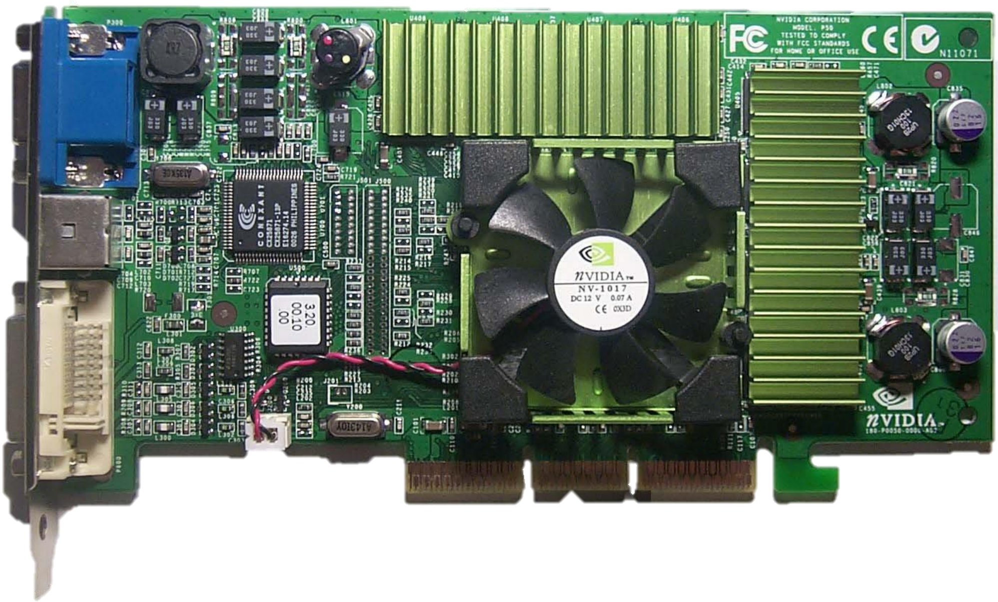
Heterogeneous computing



GPGPU First Steps

GPGPU First Steps

GeForce 3 and programmable vertex/pixel shaders



GPGPU First Steps

GeForce 3 and programmable vertex/pixel shaders

GPGPU First Steps

Fool the GPU using graphics APIs (OpenGL / DirectX)

$$Y = Y + alpha * X$$

NVIDIA 3DPERCEPTIONLAB

GPGPU First Steps

Fool the GPU using graphics APIs (OpenGL / DirectX)

```
void saxpy(float* Y, float* X, int alpha, int n){
        for (int i = 0; i < n; ++i)
                Y[i] = Y[i] + alpha * X[i];
}
```

GPGPU First Steps

Fool the GPU using graphics APIs (OpenGL / DirectX)

```
float saxpy (
        float2 coords : TEXCOORD0,
        uniform sampler2D textureY,
        uniform sampler2D textureX,
        uniform float alpha ) : COLOR
{
        float result;
        float yval=y_old[i];
        float y = tex2D(textureY,coords);
        float xval=x[i];
        float x = tex2D(textureX,coords);
        y_new[i]=yval+alpha*xval;
        result = y + alpha * x;
        return result;
}
```

GPGPU First Steps

Progress was cut down because of certain limitations

– Steep learning curve (OpenGL/DirectX) and program translation

– Need to learn shading languages (e.g., Cg or GLSL)

– No guarantees on FP32 (float) or FP64 (double) support

– Strict limitations on memory access patterns (reading and writing)

– Lack of development and debugging tools

– Limited resources in general: memory, clock speed, flexibility…

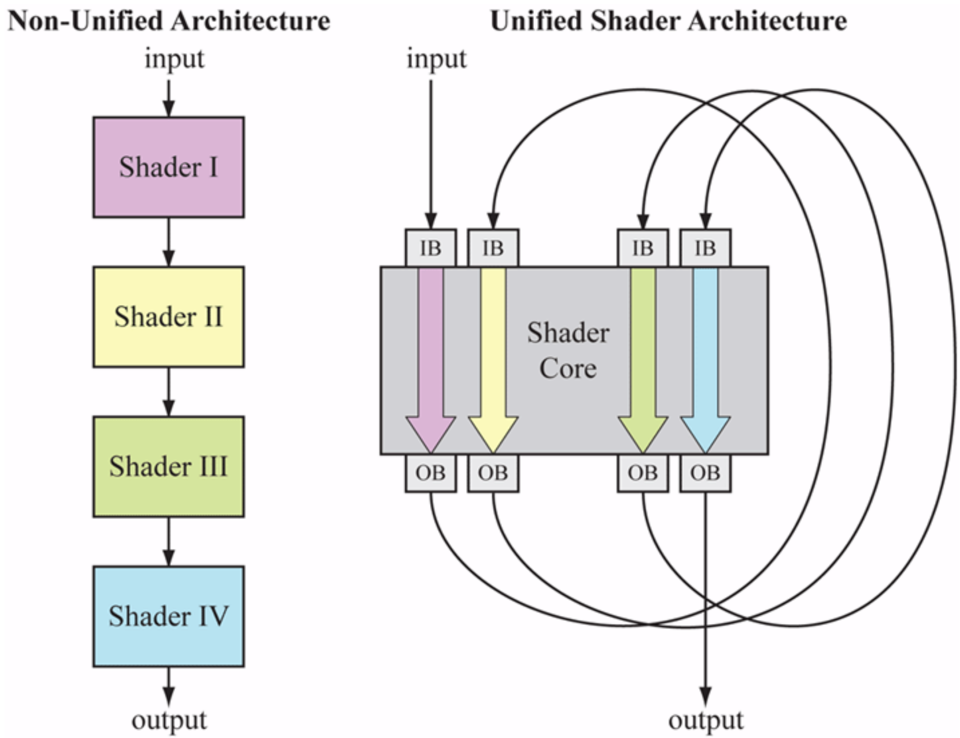
CUDA Architecture

CUDA Architecture

GeForce 8800 GTX

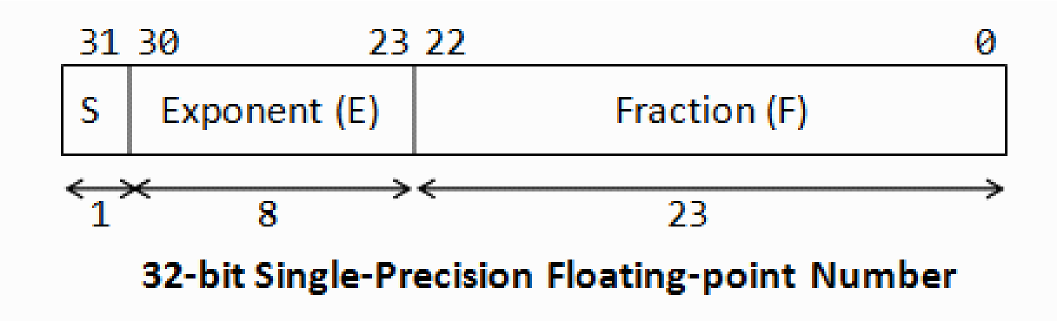
CUDA Architecture

Unified shaders



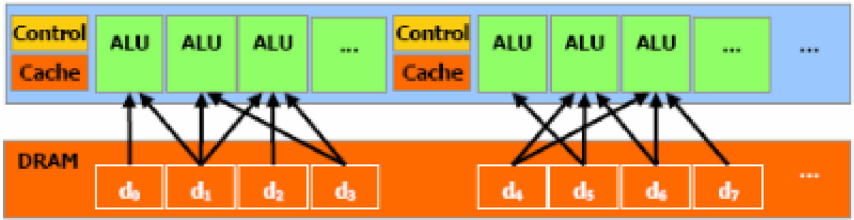
CUDA Architecture

Single precision floating point (FP32)

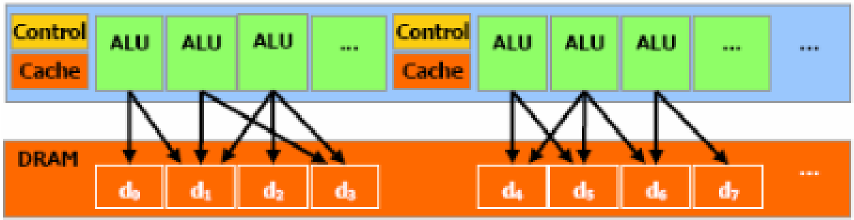


CUDA Architecture

Uncommon (for graphics) memory access patterns



Gather



Scatter

CUDA Architecture
Platform and ecosystem

- Brand new hardware architecture
- Specialized driver for GPU computing
- Flexible and “friendly” programming language (based on C++)
- Development environment
  - NVCC compiler
  - Visual Studio integration
  - Visual Studio Nsight for debugging
- Documentation, tons of documentation and tutorials!
- Collaboration with academia (donations, support…)

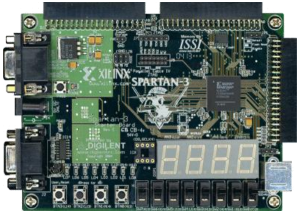
Conclusion

Conclusion

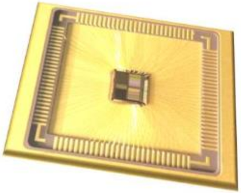
Programmability vs. efficiency

CPU

FPGA



ASIC



Efficiency

Development effort / flexibility

Conclusion

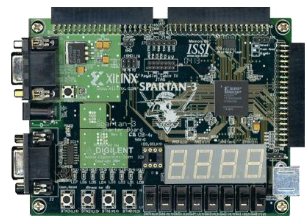
Programmability vs. efficiency

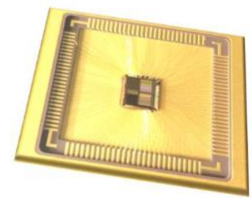
GPU

CPU

FPGA

ASIC



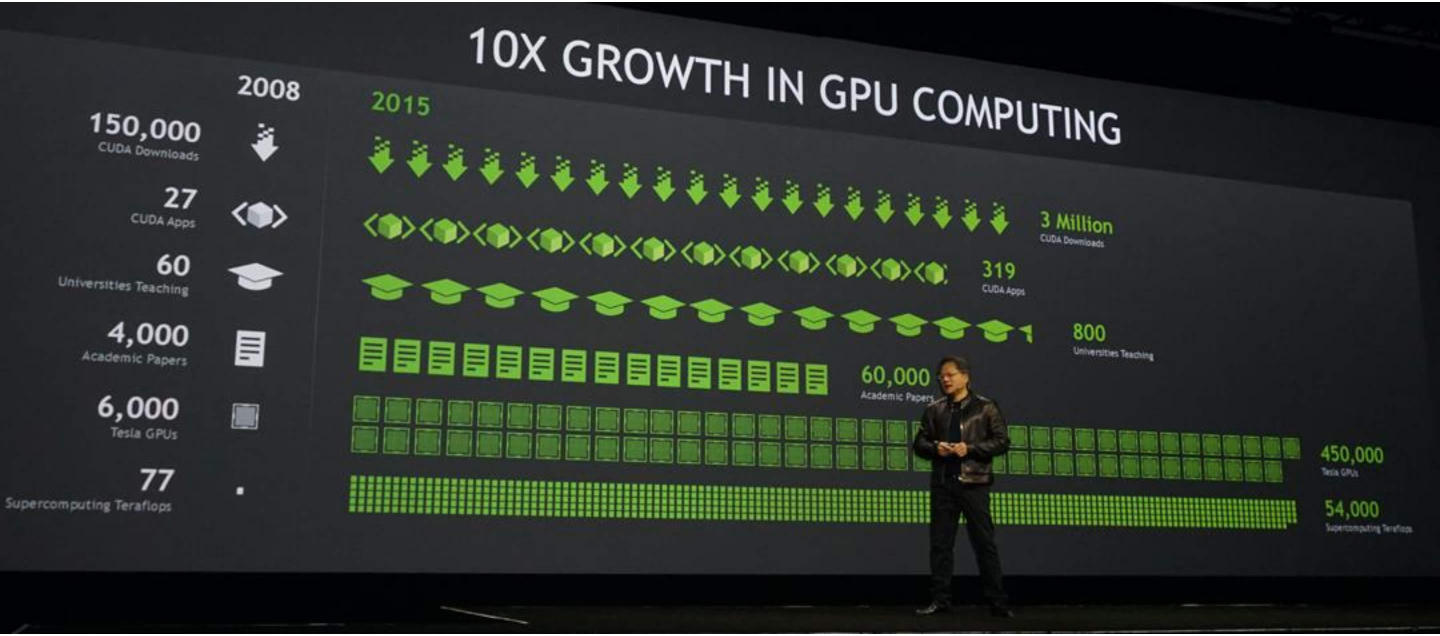


Efficiency

Development effort / flexibility

Conclusion

Meteoric ascent



Inception and Evolution of GPUs

A Brief Story of Graphics Processing Units

Thanks for your attention!

They have also been done in collaboration with SERGIO ORTS ESCOLANO!

Albert García García

agarcia @ dtic.ua.es