Intro to CUDA: GPU Architectures

Dr Paul Richmond

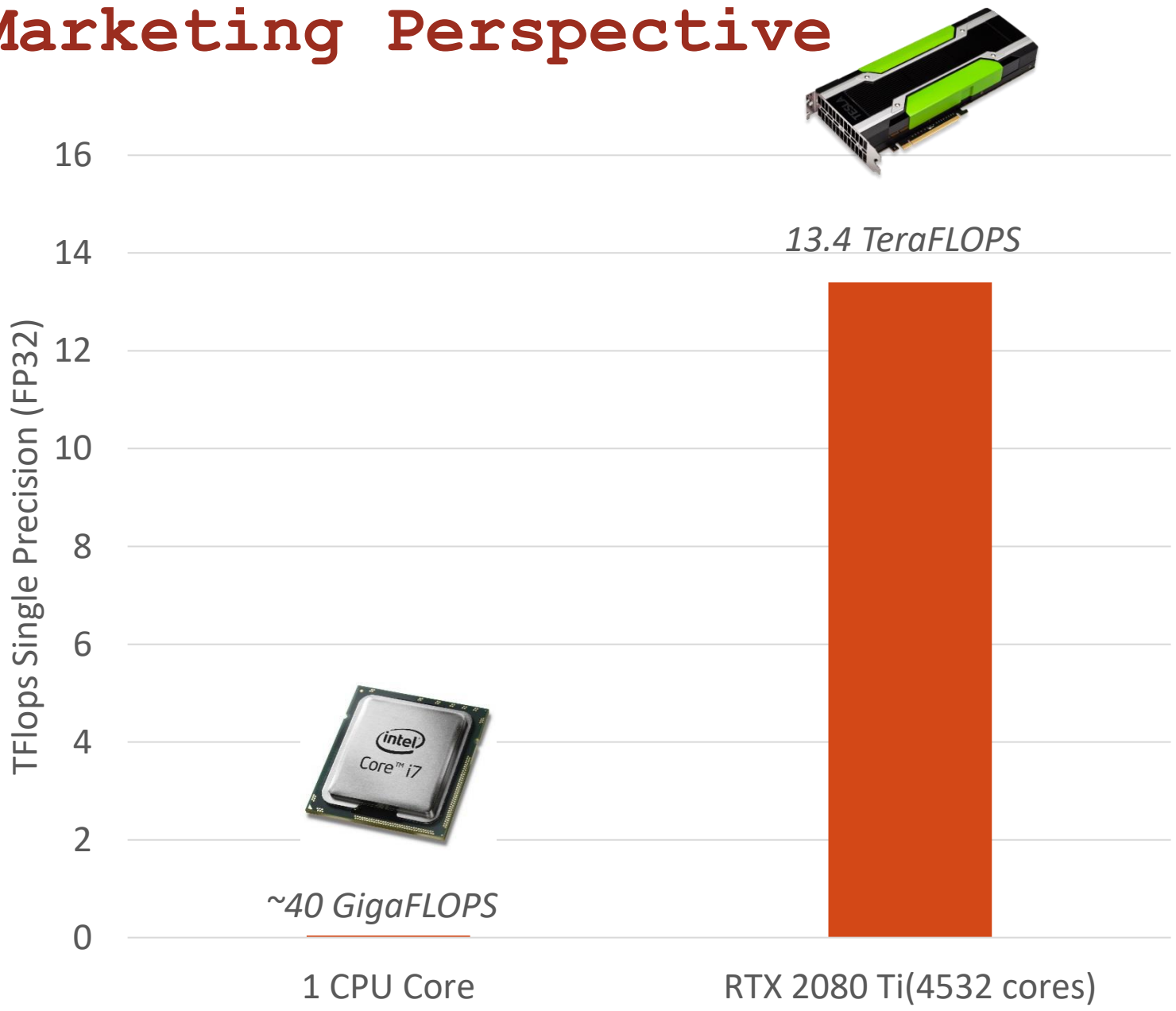
Institute of Computing for Climate Science (ICCS)

https://iccs.cam.ac.uk/

UNIVERSITY OF CAMBRIDGE

- ❑ **Introduction to GPU Performance**
- ❑ Parallelism and Micro-processor Design
- ❑ CPUs vs GPUs
- ❑ GPU Hardware and Accelerated Systems Design
- ❑ Programming GPUs

Marketing Perspective



4.9 hours *CPU* time

vs.

1 minute *GPU* time

UNIVERSITY OF CAMBRIDGE

Reality

python - GPU is slower several ti

stackoverflow.com/questions/61649311/gpu-is-slower-several-times-than-cpu-how-to-improve-the-code

stackoverflow Products Customers Use cases Search... Log in Sign up

Home

PUBLIC

Stack Overflow

Tags

Users

Jobs

TEAMS What's this?

Free 30 Day Trial

GPU is slower several times than CPU, how to improve the code?

Ask Question

machine learning - Why is my GP

stackoverflow.com/questions/41948406/why-is-my-gpu-slower-than-cpu-when-training-lstm-rnn-models

stackoverflow Products Customers Use cases Search... Log in Sign up

Home

PUBLIC

Stack Overflow

Tags

Users

Jobs

TEAMS What's this?

Free 30 Day Trial

Why is my GPU slower than CPU when training LSTM/RNN models?

Ask Question

Asked 3 years, 3 months ago Active 8 months ago Viewed 23k times

29

6

My machine has the following spec:

CPU: Xeon E5-1620 v4

GPU: Titan X (Pascal)

Ubuntu 16.04

Nvidia driver 375.26

CUDA tookit 8.0

cuDNN 5.1

I've benchmarked on the following Keras examples with Tensorflow as the backed reference:

The Overflow Blog

Podcast 236: A glitch in the Matrix

Good coders borrow, great coders steal

Featured on Meta

Creative Commons Licensing UI and Data Updates

Meta escalation/response process update

The problem with 100x speedups

- GPU is **not** magic
- GPU **and** CPU Optimisation is not easy
- GPUs done right
  - ~5-10x speedup of memory bound HPC applications
  - More energy efficiency for HPC
  - Utilisation of readily available desktop performance
  - Performance portability

- ❑ Introduction to GPU Performance
- ❑ **Parallelism and Micro-processor Design**
- ❑ CPUs vs GPUs
- ❑ GPU Hardware and Accelerated Systems Design
- ❑ Programming GPUs

Ubiquity of Parallelism

Mobile

8 cores

CPU

~16 cores

GPU

~16 cores

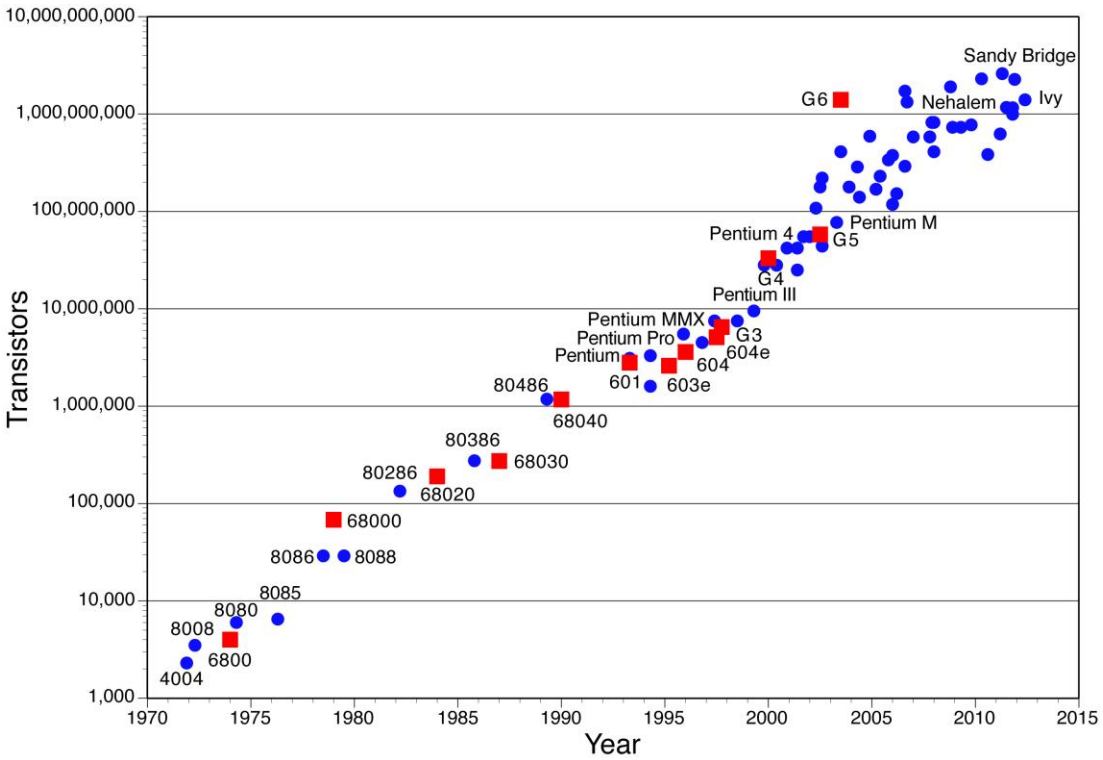
Supercomputer

8,699,904 cores

- Parallel computing is **already here**, and it is here to stay
- Trend is **increasing numbers** of simpler lower-power cores

Transistors != performance

- Moores Law: A doubling of transistors every couple of years
  - Not a law actually an observation
  - Doesn't actually say anything about performance
- Future of Moore's Law
  - Moore's law is dead!
  - A bright future for Moore's Law



UNIVERSITY OF CAMBRIDGE

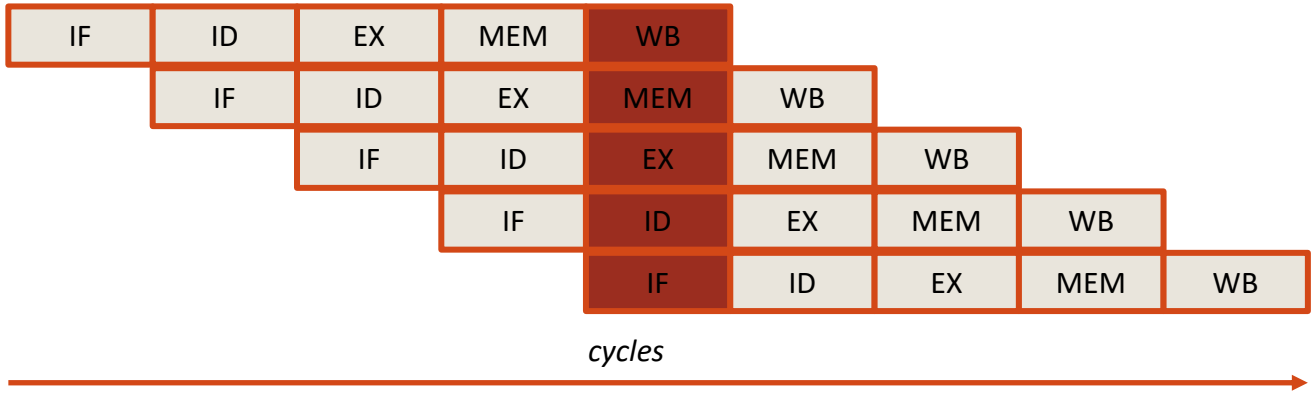
Dennard Scaling

"As transistors get smaller their power density stays constant"

$$\text{Power} = \text{Frequency} \times \text{Voltage}^2$$

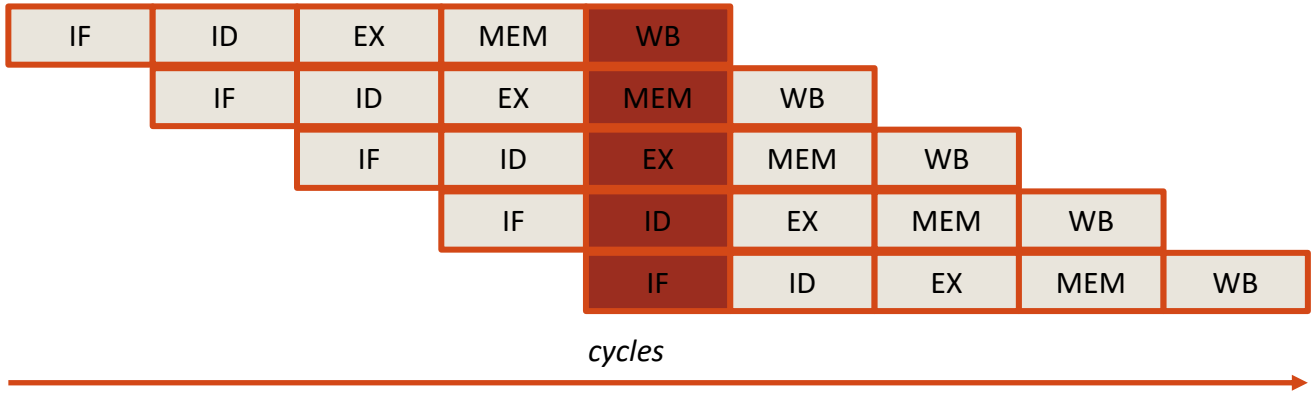
- Performance improvements for CPUs traditionally realised by increasing frequency
- Decrease voltage to maintain a steady power
  - Only works so far
- Increase Power
  - Disastrous implications for cooling

Instruction Level Parallelism



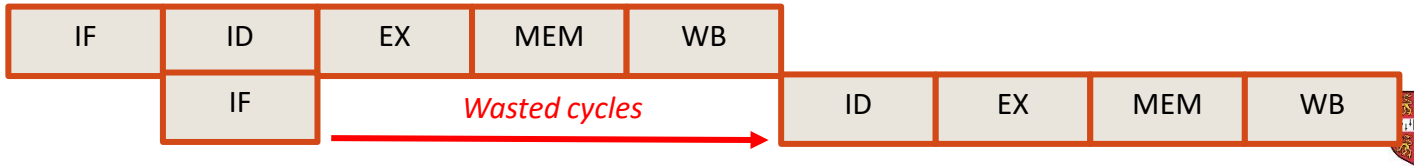
- ❑Transistors used to build more complex architectures
- ❑Use pipelining to overlap instruction execution

Instruction Level Parallelism



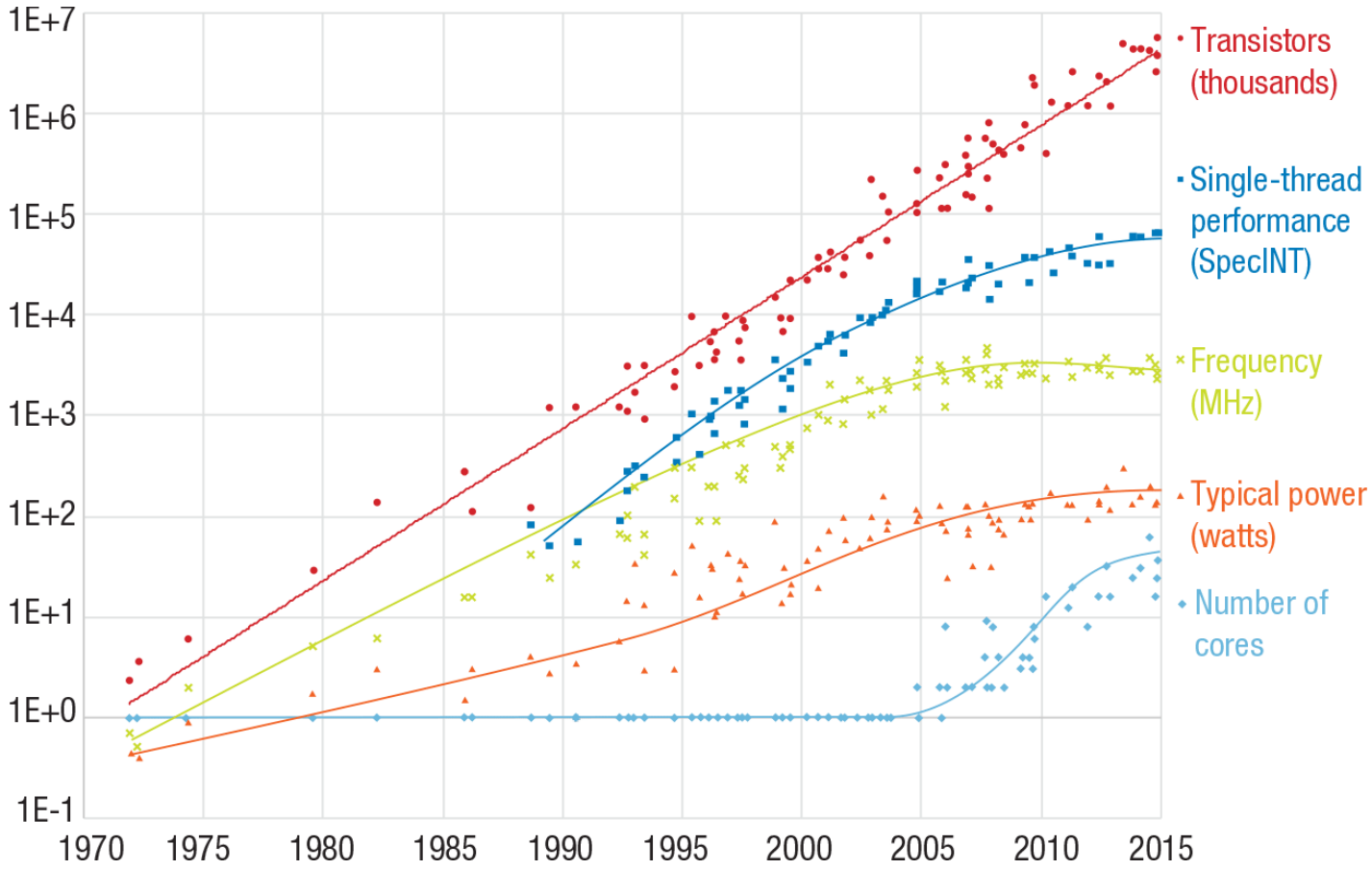
- Transistors used to build more complex architectures
- Use pipelining to overlap instruction execution

```
add 1 to R1
copy R1 to R2
```



UNIVERSITY OF CAMBRIDGE

Golden Era of Performance

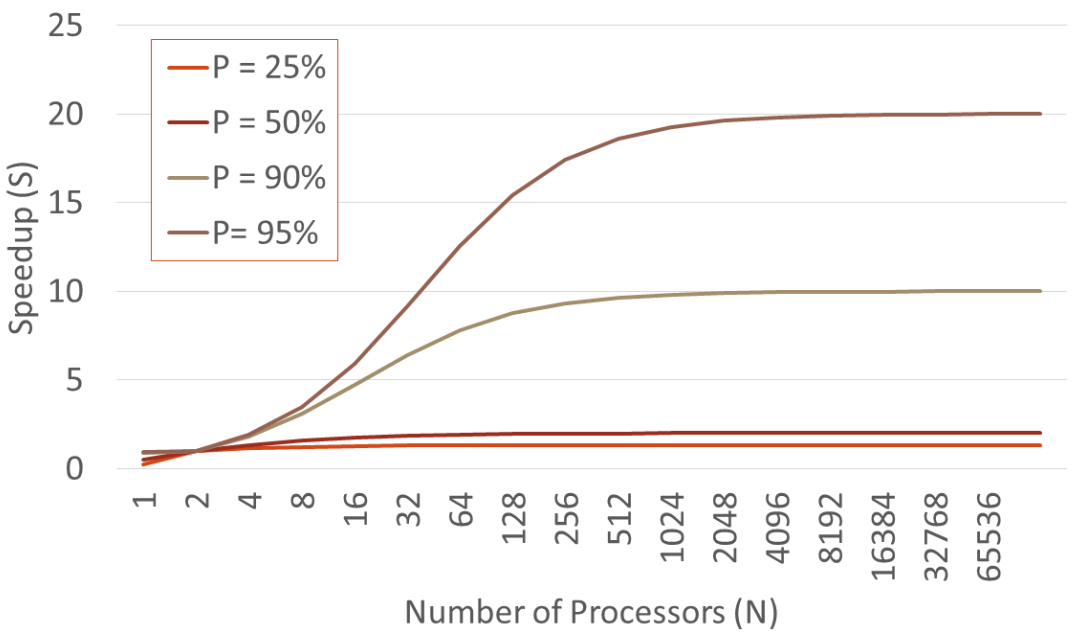


Adapting to Thrive in a New Economy of Memory Abundance, K Bresniker et al.

- 90s saw great improvements to single CPU performance
  - 1980s to 2002: 100% performance increase every 2 years
  - 2002 to now: ~40% every 2 years

Why More Cores?

$$Speedup\ (S) = \frac{1}{\frac{P}{N} - (1 - P)}$$



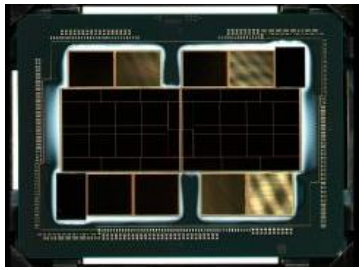
- ❑Partial parallelism does not solve the problem (Amdahl's law)
- ❑Use extra transistors for multi/many core parallelism
  - ❑More operations per clock cycle
  - ❑Power can be kept low
  - ❑Processor designs can be simple – shorter pipelines (RISC)
- ❑**Introduces software constraints….**

UNIVERSITY OF CAMBRIDGE

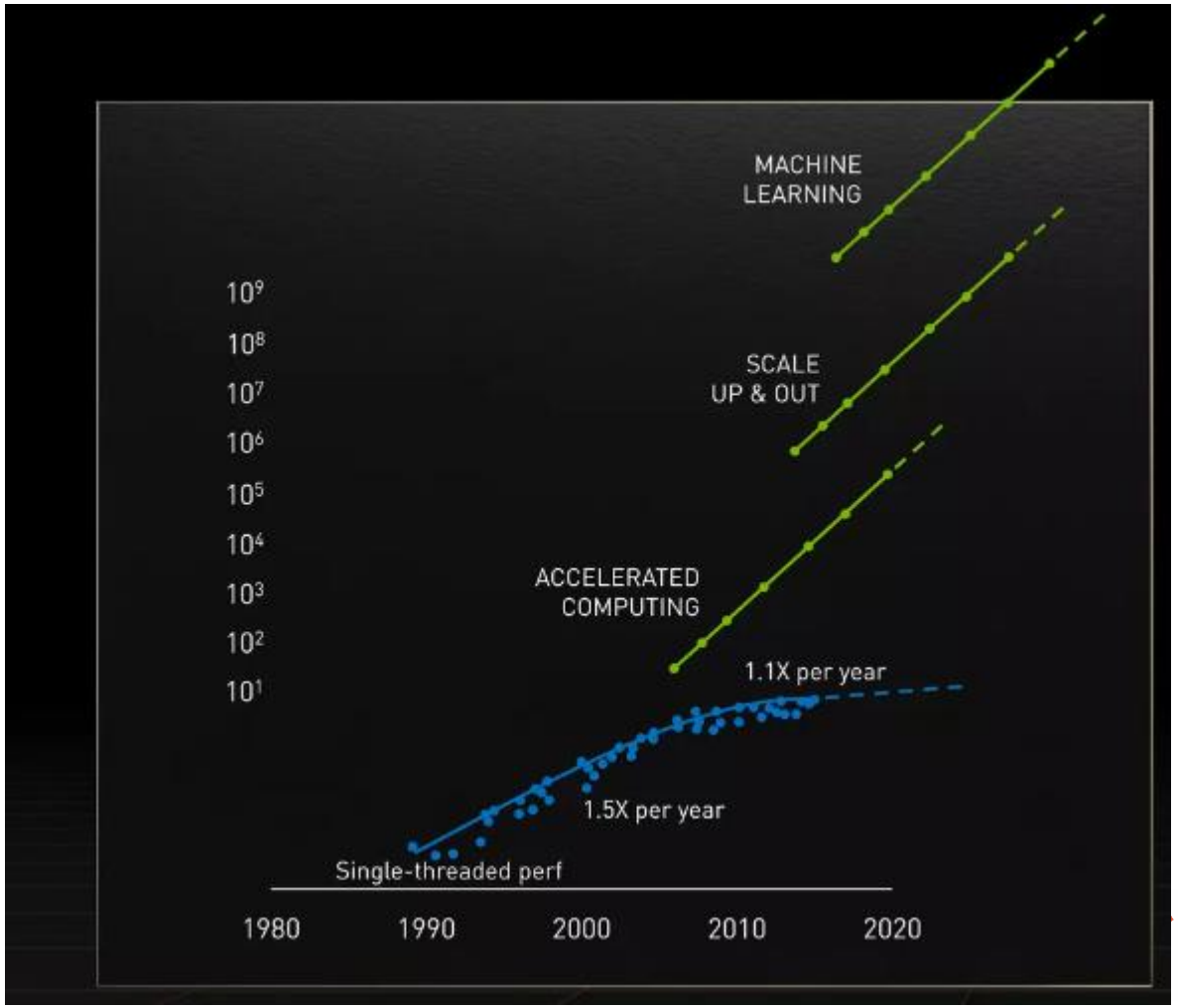
- ❑Introduction to GPU Performance
- ❑Parallelism and Micro-processor Design
- ❑**CPUs vs GPUs**
- ❑GPU Hardware and Accelerated Systems Design
- ❑Programming GPUs

GPUs and Many Core Designs

- Take the idea of multiple cores to the extreme (many cores)
- Dedicate more die space to compute
  - At the expense of branch prediction, out of order execution, etc.
- Simple, Lower Power and Highly Parallel
  - Very effective for HPC and ML applications



From GTC 2022 Keynote Talk, NVIDIA CEO Jensen Huang



Latency vs. Throughput

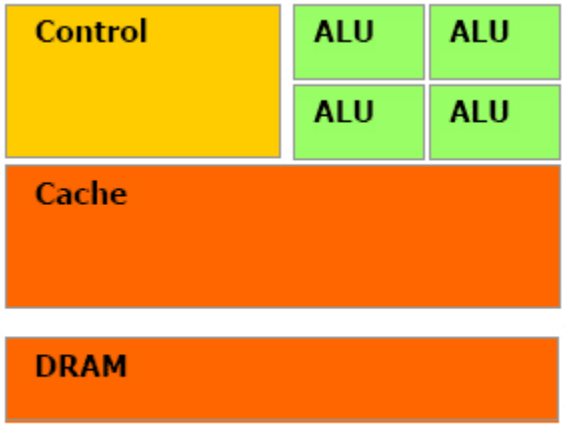
- Latency: The time required to perform some action
  - Measured in units of time
- Throughput: The number of actions executed per unit of time
  - Measured in units of what is produced

- E.g. An assembly line manufactures GPUs. It takes 8 hours to manufacture a GPU but the assembly line can manufacture 100 GPUs per day.

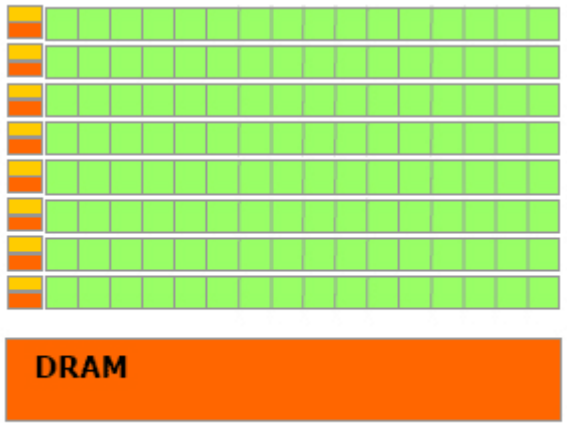
CPU vs GPU

- CPU
  - Latency oriented
  - Optimised for serial code performance
  - Good for single complex tasks
- GPU
  - Throughput oriented
  - Massively parallel architecture
  - Optimised for performing many similar tasks simultaneously (data parallel)

CPU vs GPU



CPU



GPU

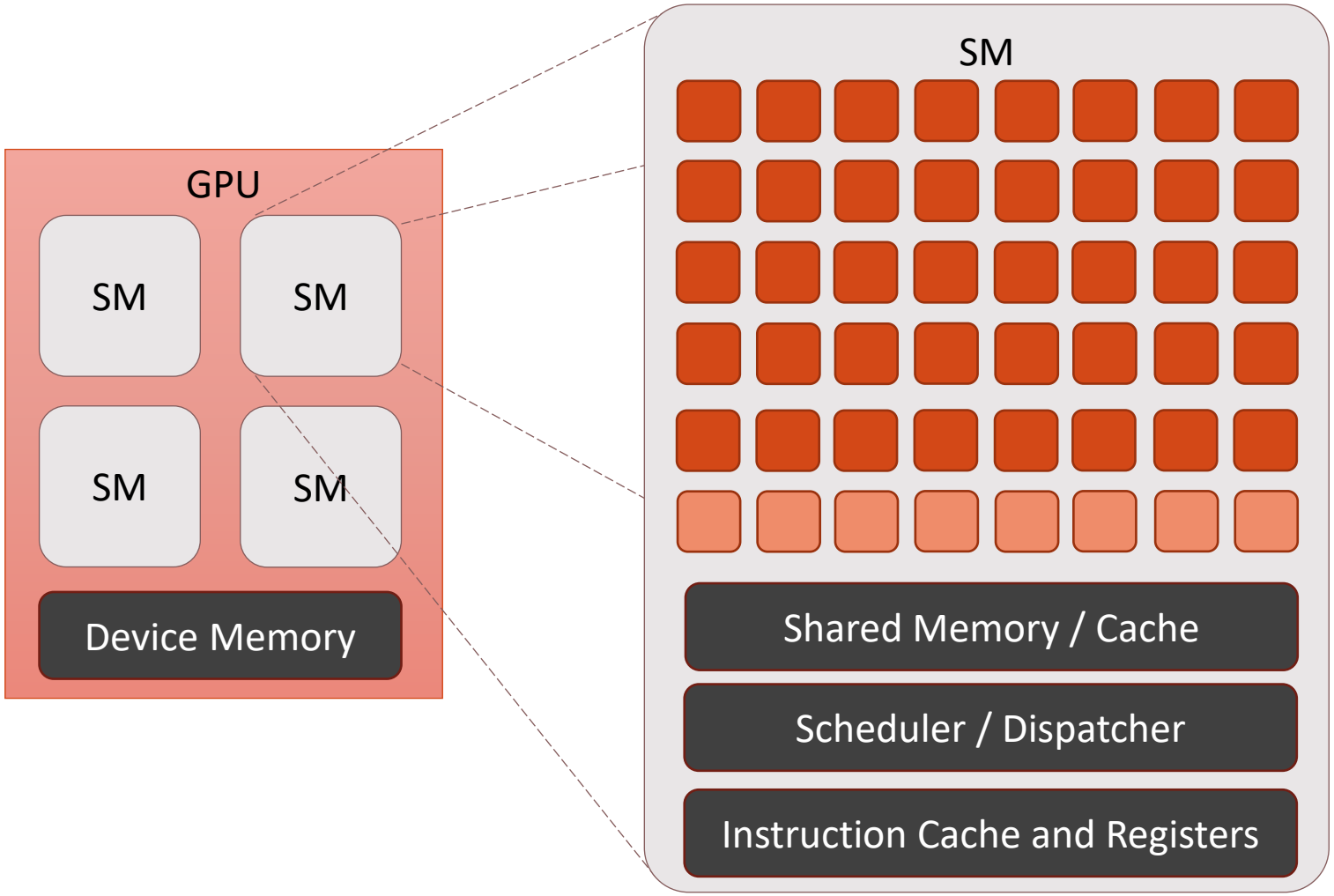
- Large Cache
  - Hide long latency memory access
- Powerful Arithmetic Logical Unit (ALU)
  - Low Operation Latency
- Complex Control mechanisms
  - Branch prediction etc.

- Small cache
  - But faster memory throughput
- Energy efficient ALUs
  - Long latency but high throughput
- Simple control
  - No branch prediction

UNIVERSITY OF CAMBRIDGE

- ❑Introduction to GPU Performance
- ❑Parallelism and Micro-processor Design
- ❑CPUs vs GPUs
- ❑**GPU Hardware and Accelerated Systems Design**
- ❑Programming GPUs

Simplistic (NVIDIA) GPU Hardware Model



- NVIDIA GPUs have a 2-level hierarchy
  - Each Streaming Multiprocessor (SM) has multiple vector (CUDA) cores
  - The number of SMs varies across different hardware implementations
  - The design of SMs varies between GPU families
  - The number of cores per SM varies between GPU families

UNIVERSITY OF CAMBRIDGE

NVIDIA vs AMD vs Intel GPUs

Form Factor	H100 SXM	
FP64	34 teraFLOPS	
FP64 Tensor Core	67 teraFLOPS	
FP32	67 teraFLOPS	
TF32 Tensor Core	989 teraFLOPS*	
BFLOAT16 Tensor Core	1,979 teraFLOPS	
FP16 Tensor Core	1,979 teraFLOPS	
FP8 Tensor Core	3,958 teraFLOPS	
INT8 Tensor Core	3,958 TOPS*	
GPU memory	80GB	
GPU memory bandwidth	3.35TB/s	
Decoders	7 NVDEC 7 JPEG	7 NVDEC 7 JPEG
Max thermal design power (TDP)	Up to 700W (configurable)	300-350W (configurable)
Multi-Instance GPUs	Up to 7 MIGS @ 10GB each	
Form factor	SXM	PCIe Dual-slot air-cooled
Interconnect	NVLink: 900GB/s PCIe Gen5: 128GB/s	NVLINK: 600GB/s PCIe Gen5: 128GB/s
Server Options	NVIDIA-Certified Systems with 4 or 8 GPUs NVIDIA DGX™ H100 with 8 GPUs	Partner and NVIDIA-Certified Systems with 1-8 GPUs
NVIDIA AI Enterprise	Add-on	Included

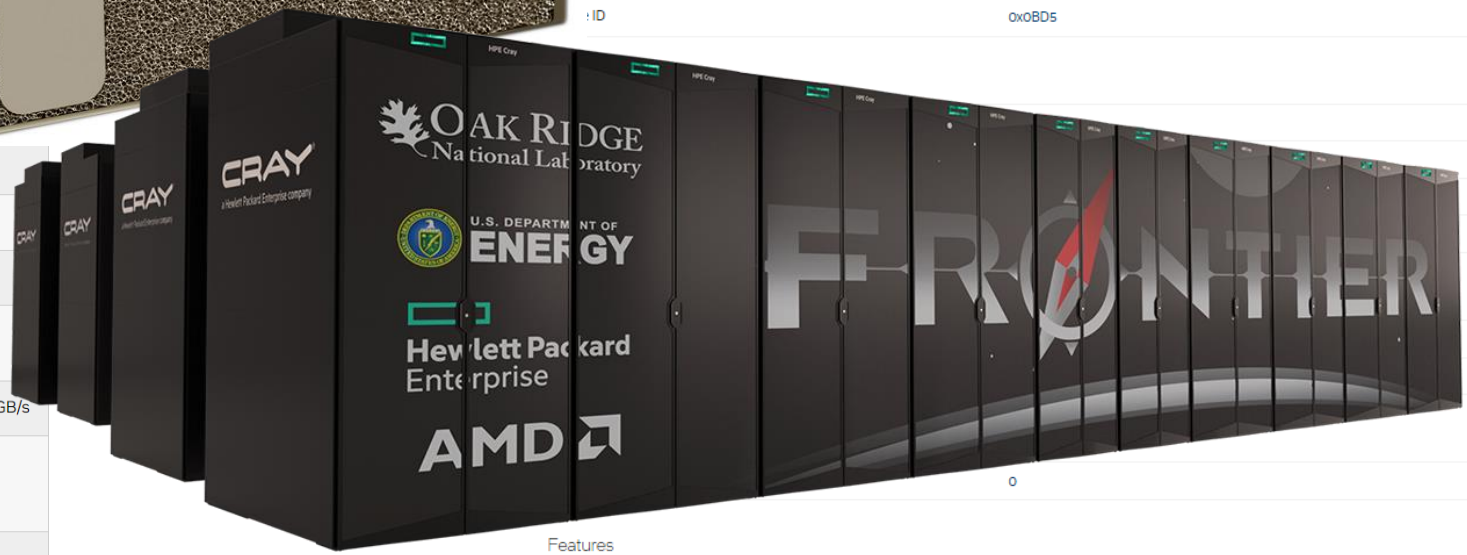
Essentials

Export specifications

Product Collection	Intel® Data Center GPU Max Series
Code Name	Products formerly Ponte Vecchio
Microarchitecture	Xe-HPC
Marketing Status	Launched
Launch Date	Q1'23
ted Discontinuance	Jan 2026
nty Period	3 yrs
onditions	Server/Enterprise
se	Artificial Intelligence, High Performance Computing

Specifications

es	128
acing Units	128
K Matrix Extensions (Intel® XMX) Engines	1024
tor Engines	1024
ics Max Dynamic Clock	1600 MHz
ics Base Clock	900 MHz
K Link Maximum Frequency	53 Gbps
	600 W
press Configurations ‡	Gen 5 x16
ID	0x0BD5



0

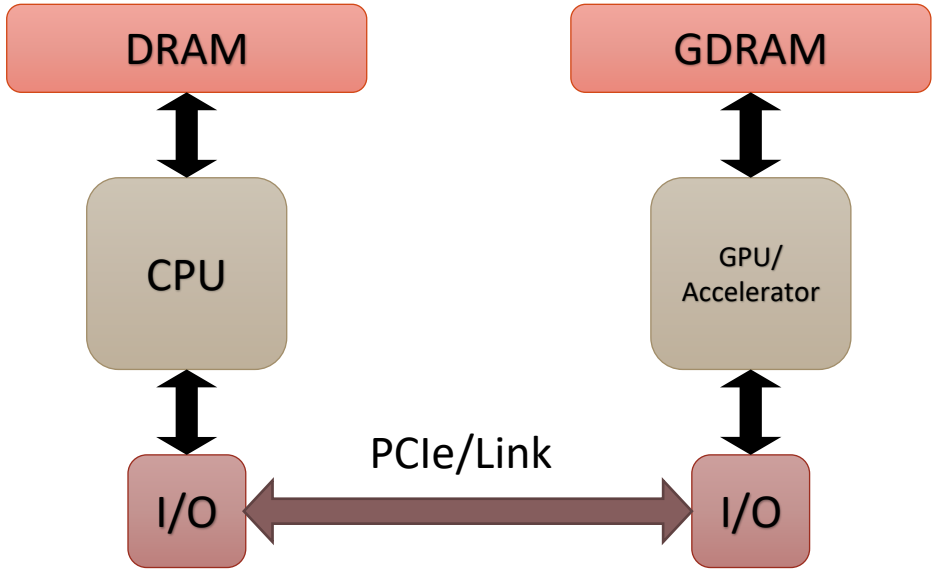
Features

H.264 Hardware Encode/Decode	No

Arms race ...

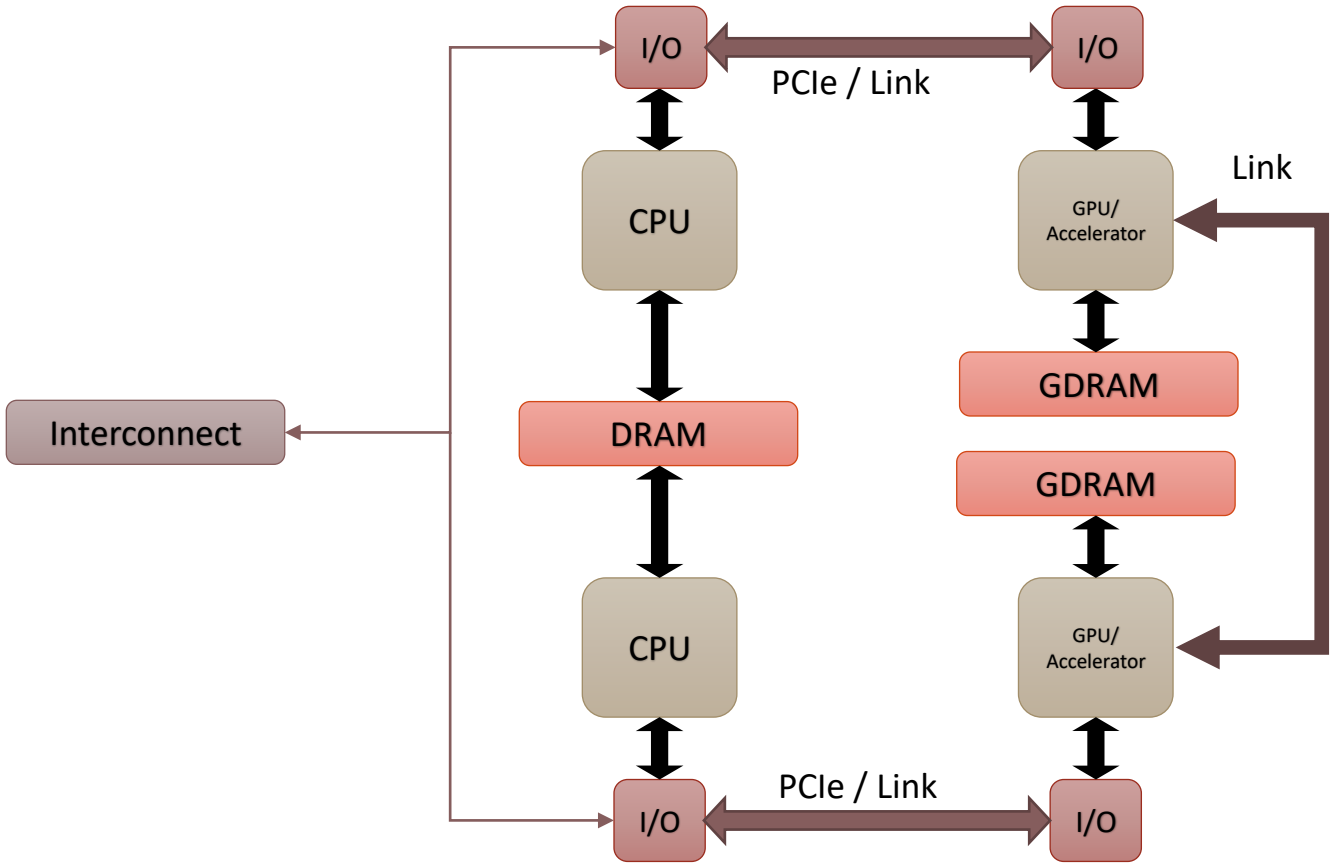
Accelerated Systems

- CPUs and GPUs are used together
  - GPUs cannot be used instead of CPUs
  - GPUs perform compute heavy parts
- Communication is via a link
  - PCIe 5.0: 128 GB/sec throughput
  - Specialist P2P Links/Switch: e.g. NVLINK(v5 for H100): 900GB/sec



Multi GPU Systems

- Can have multiple CPUs and Accelerators within each "Shared Memory Node"
  - CPUs share physical memory but accelerators do not!

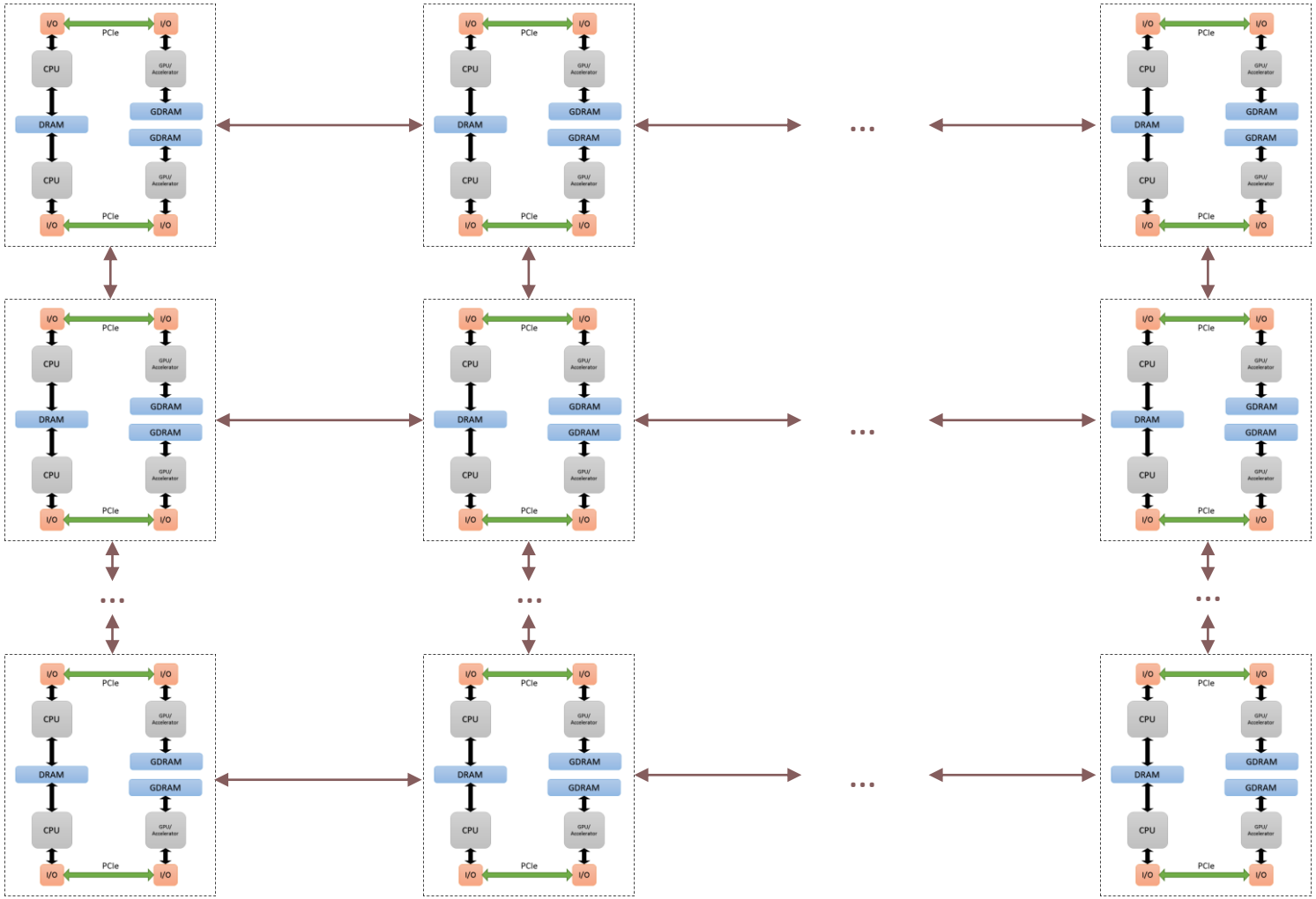


GPU Workstation

- For example 2 multi core CPUs + 4 GPUS
- Make sure your case and power supply are upto the job!

UNIVERSITY OF CAMBRIDGE

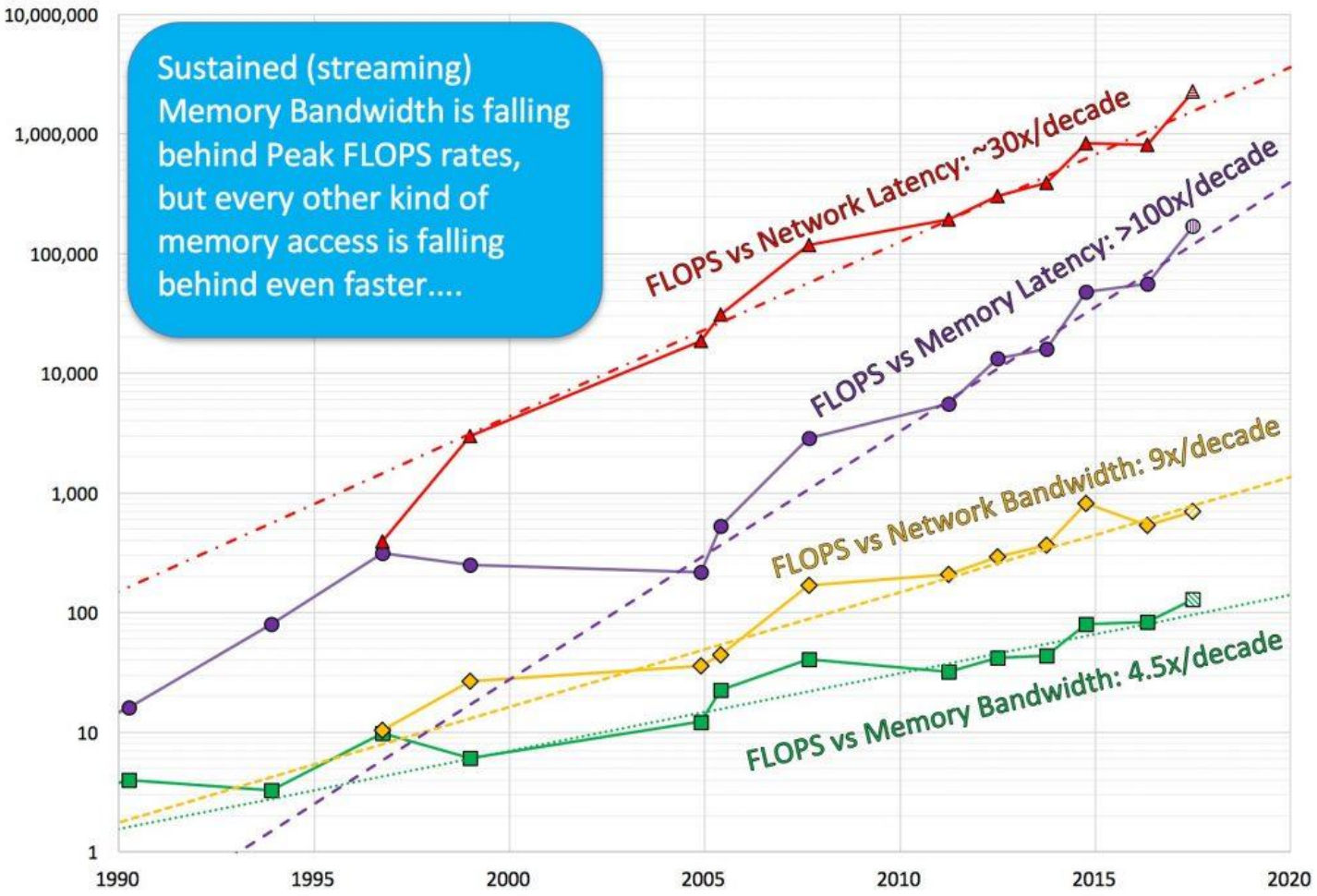
Accelerated Supercomputers



UNIVERSITY OF CAMBRIDGE

HPC Observations

- Improvements at individual computer node level are greatest
  - Better parallelism
  - Hybrid processing
  - 3D fabrication
- Communication costs are increasing
  - Memory per core is reducing
- Throughput > Latency

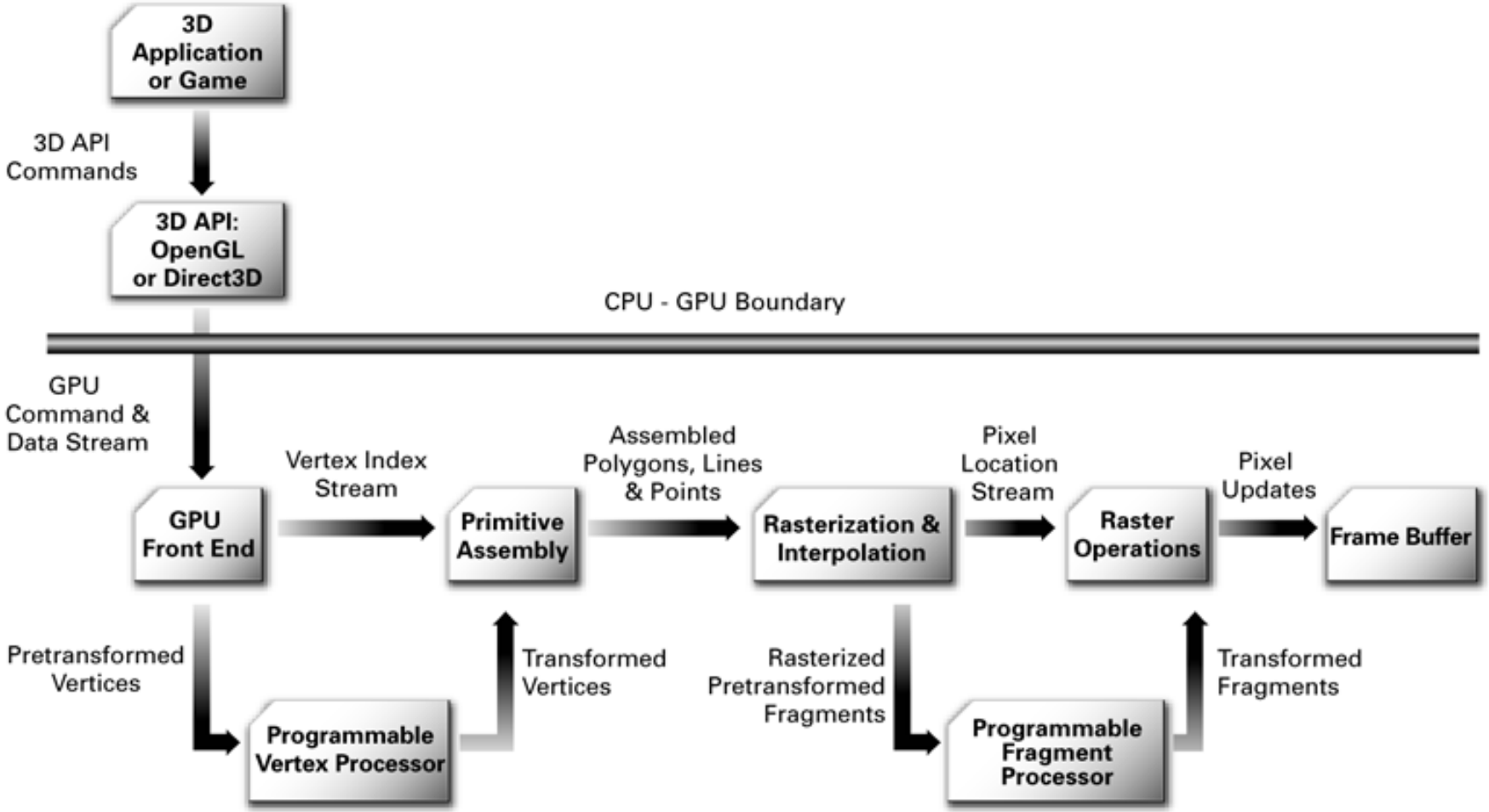


http://sc16.supercomputing.org/2016/10/07/sc16-invited-talk-spotlight-dr-john-d-mccalpin-presents-memory-bandwidth-system-balance-hpc-systems/

UNIVERSITY OF CAMBRIDGE

- ❑Introduction to GPU Performance
- ❑Parallelism and Micro-processor Design
- ❑CPUs vs GPUs
- ❑GPU Hardware and Accelerated Systems Design
- ❑**Programming GPUs**

The Graphics Pipeline



Source: *NVidia Cg Users Manual*

UNIVERSITY OF CAMBRIDGE

GPGPU

- General Purpose computation on Graphics Hardware
  - First termed by Mark Harris (NVIDIA) in 2002
  - Recognised the use of GPUs for non graphics applications
- Requires mapping a problem into graphics concepts
  - Data into textures (images)
  - Computation into shaders
- Later unified processors were used rather than fixed stages
  - 2006: GeForce 8 series

Unified Processors and CUDA

- Compute Unified Device Architecture (CUDA)
  - First released in 2006/7
- Targeted new bread of unified “streaming multiprocessors”
- C like programming for GPUs
  - No computer graphics: General purpose programming model
  - Revolutionised GPU programming for general purpose use

luci fish

Level of Control vs Portability

- Trade off between
  - Ease of Use
  - Level of Control
  - Portability

- Libraries and APIs
  - Abstract GPU
- Accelerated standard languages,
  - Easy to use but restricted control
- Incremental Portable Optimisation (Directives)
  - More control over data movement and HOW to parallelise
  - Increased portability
- Platform Specialisation Languages
  - Lowest level of control (and difficulty)
  - Need architectural awareness

PyTorch

C++ Standard Parallelism

OpenMP

NVIDIA CUDA

UNIVERSITY OF CAMBRIDGE

Standard Language Parallelism (Fortran)

```
integer :: n, i
do (i = 1: n)
        y(i) = a*x(i)+y(i)
enddo
```

Standard Language Parallelism (Fortran)

```
integer :: n, i
do concurrent (i = 1: n)
        y(i) = a*x(i)+y(i)
enddo
```

Directive based GPU programming

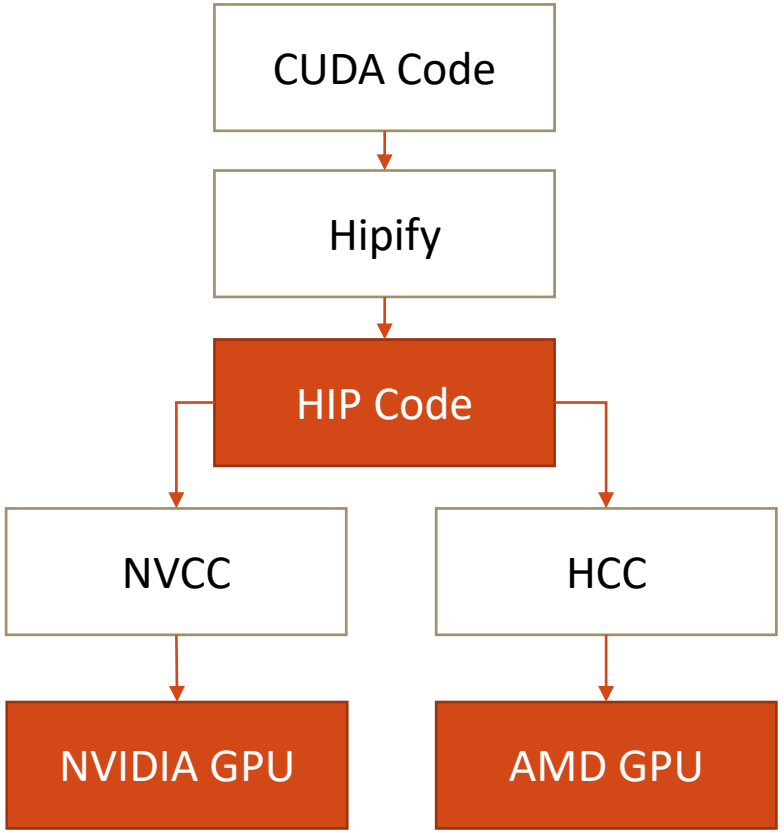
- GPU Accelerated Directives (OpenACC)
  - Helps compiler auto generate code for the GPU
  - Very similar to OpenMP
  - *Pros: Performance portability, limited understanding of hardware required*
  - *Cons: Limited fine grained control of optimisation*
- OpenMP 4.0
  - GPU offload for parallelism
  - *Pros: Platform and hardware independent, write once*
  - *Cons: Difficult to obtain high performance or use cutting edge features*

```
#pragma omp target data map (to: c[0:N], b[0:N]) map(tofrom: a[0:N])
#pragma omp target teams distribute parallel for
for (j=0; j<N; j++){
    a[j] = b[j]+scalar*c[j];
}
```

UNIVERSITY OF CAMBRIDGE

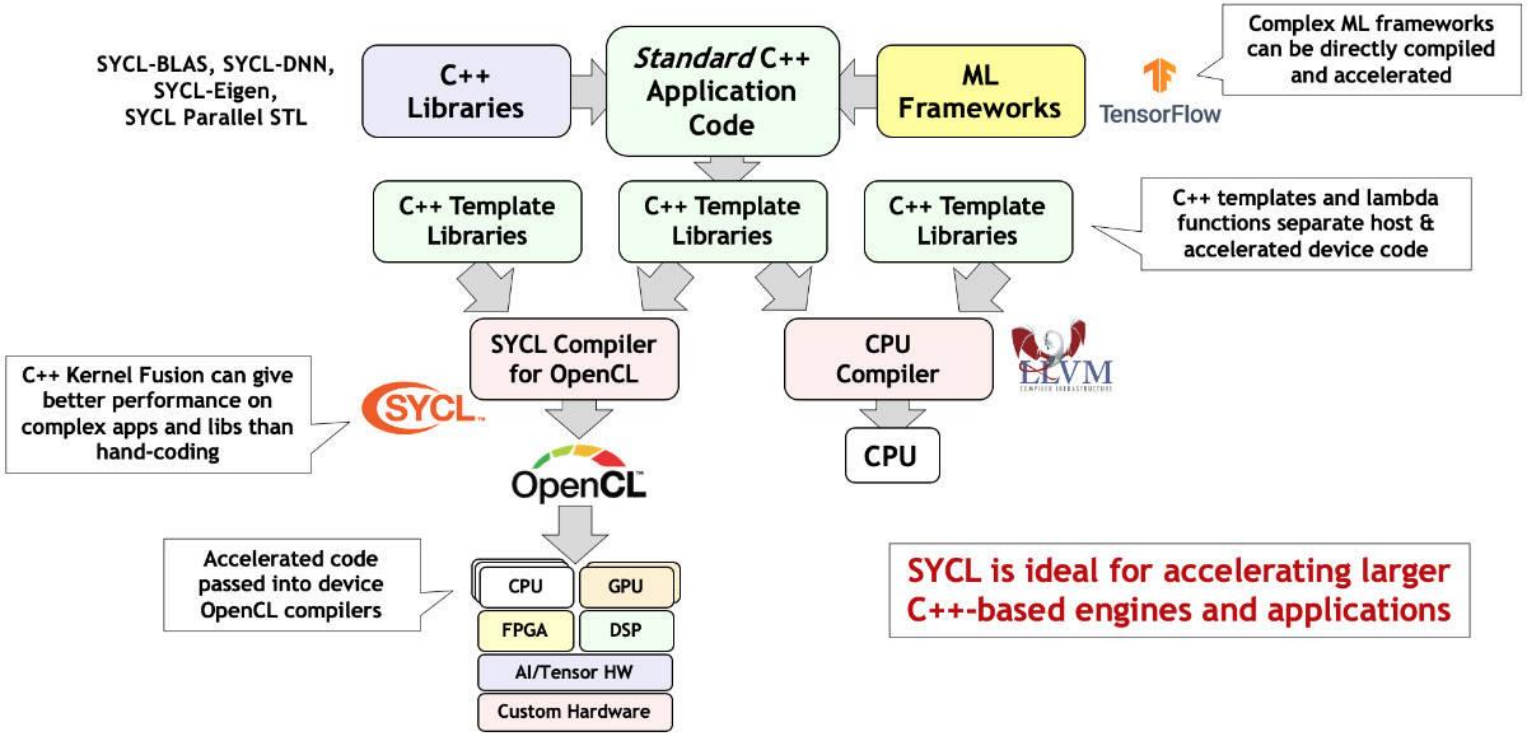
ROCm and HIP

- Radeon Open Compute (ROCm)
  - Platform and runtime for Gpu compute
  - AMD open equivalent of CUDA
- Heterogeneous-Compute Interface for Portability (HIP)
  - C++ interface
  - One to one replacement for CUDA
  - HIP source to source conversion tools
- *Pros: Can run on AMD and NVIDIA GPU hardware*
- *Cons: Subset of the CUDA language*



OpenCL and SYCL

- OpenCL: Multiple architecture support (CPUs/GPUs/FPGA)
  - Lower level than CUDA
  - Portability diminished if code is “targeted”
- SYCL: Based on modern C++ (C++17)
  - Performance portable
  - Implementations supported by different vendors (e.g. hipSYCL)



https://www.khronos.org/sycl/

Why a course on CUDA?

- ❑Excellent support and documentation
- ❑Readily available hardware (supported on consumer GPUs)
- ❑Hardware knowledge help in understanding GPU performance at high levels of abstraction
- ❑Programming concepts map easily to other approaches (e.g. HIP)

Summary

- GPUs are better suited to parallel tasks than CPUs
- Accelerators are typically not used alone, but work in tandem with CPUs
- GPU hardware is constantly evolving
- GPU accelerated systems scale from simple workstations to large-scale supercomputers
- Many approaches to GPU programming each with trade off