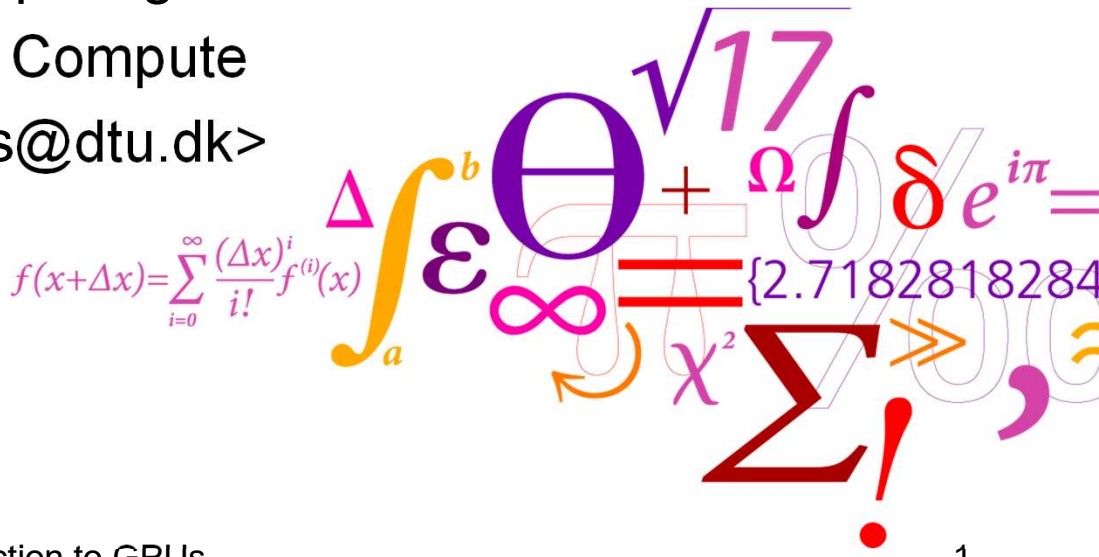
Introduction to GPUs

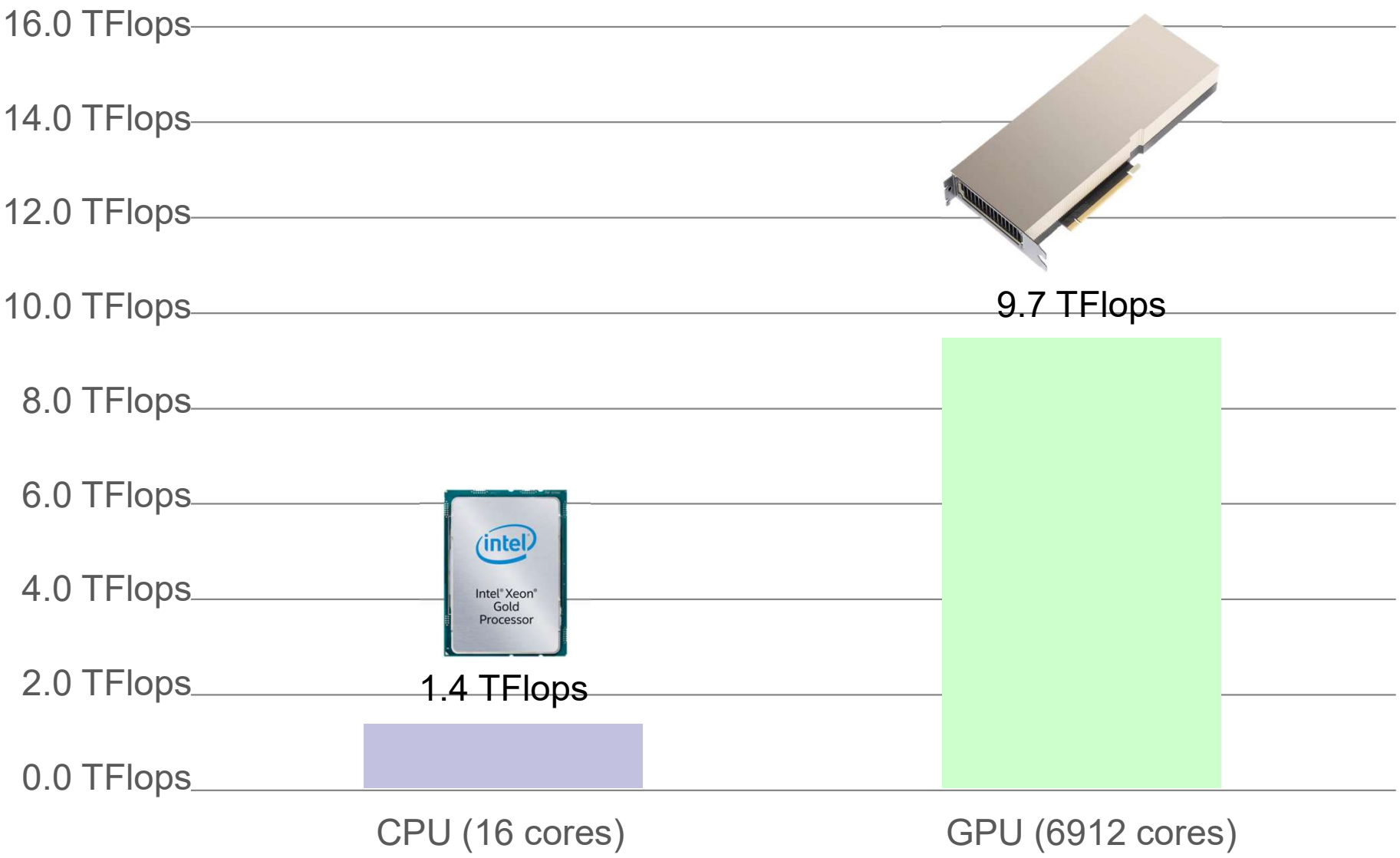
Hans Henrik Brandenborg Sørensen
DTU Computing Center
DTU Compute
<hhbs@dtu.dk>



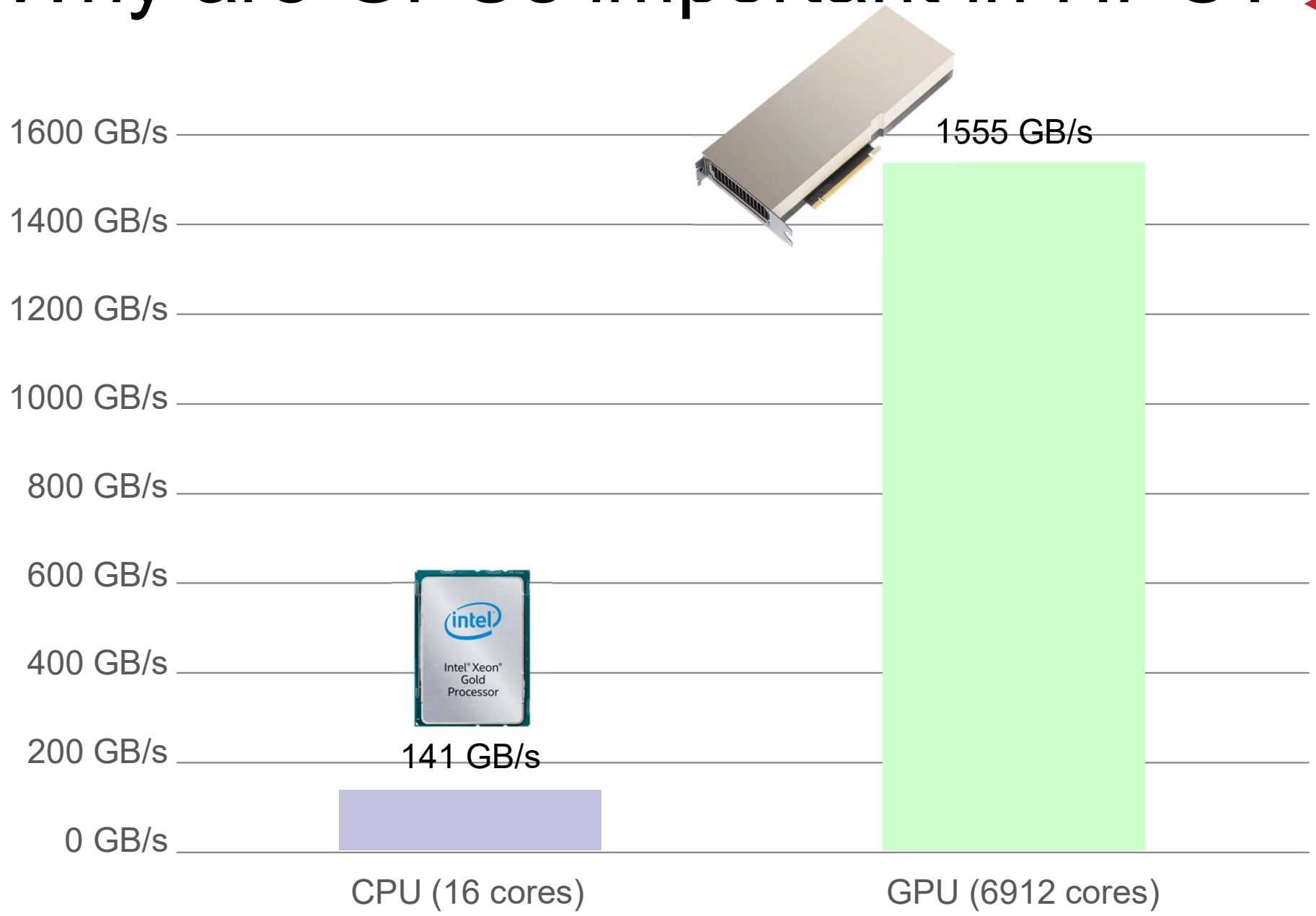
Outline

- Why are GPUs important for you?
- Why are GPUs different from CPUs?
- GPUs as accelerators
- Tensor Cores
- Performance / scratching the surface

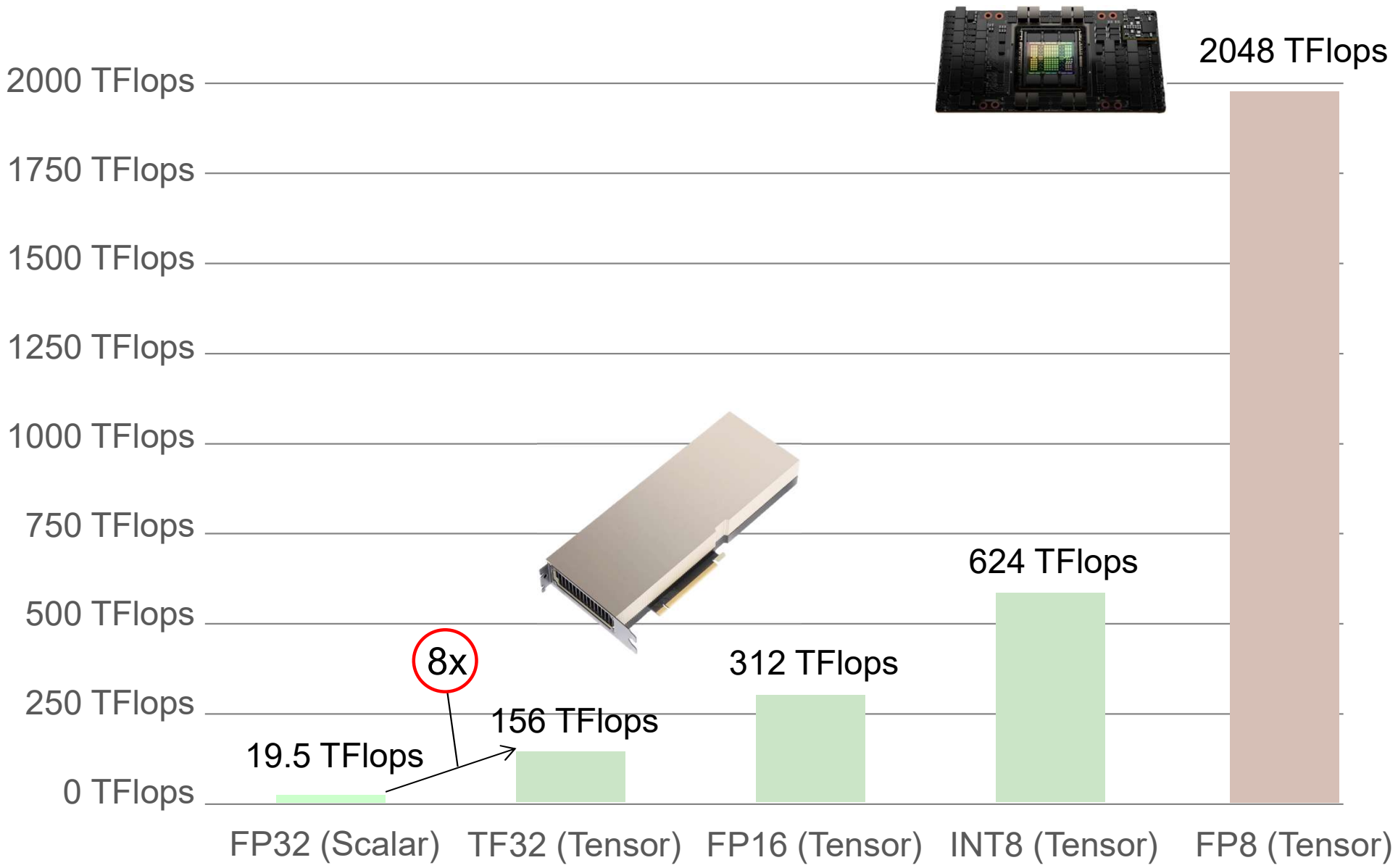
Why are GPUs important in HPC?



Why are GPUs important in HPC?



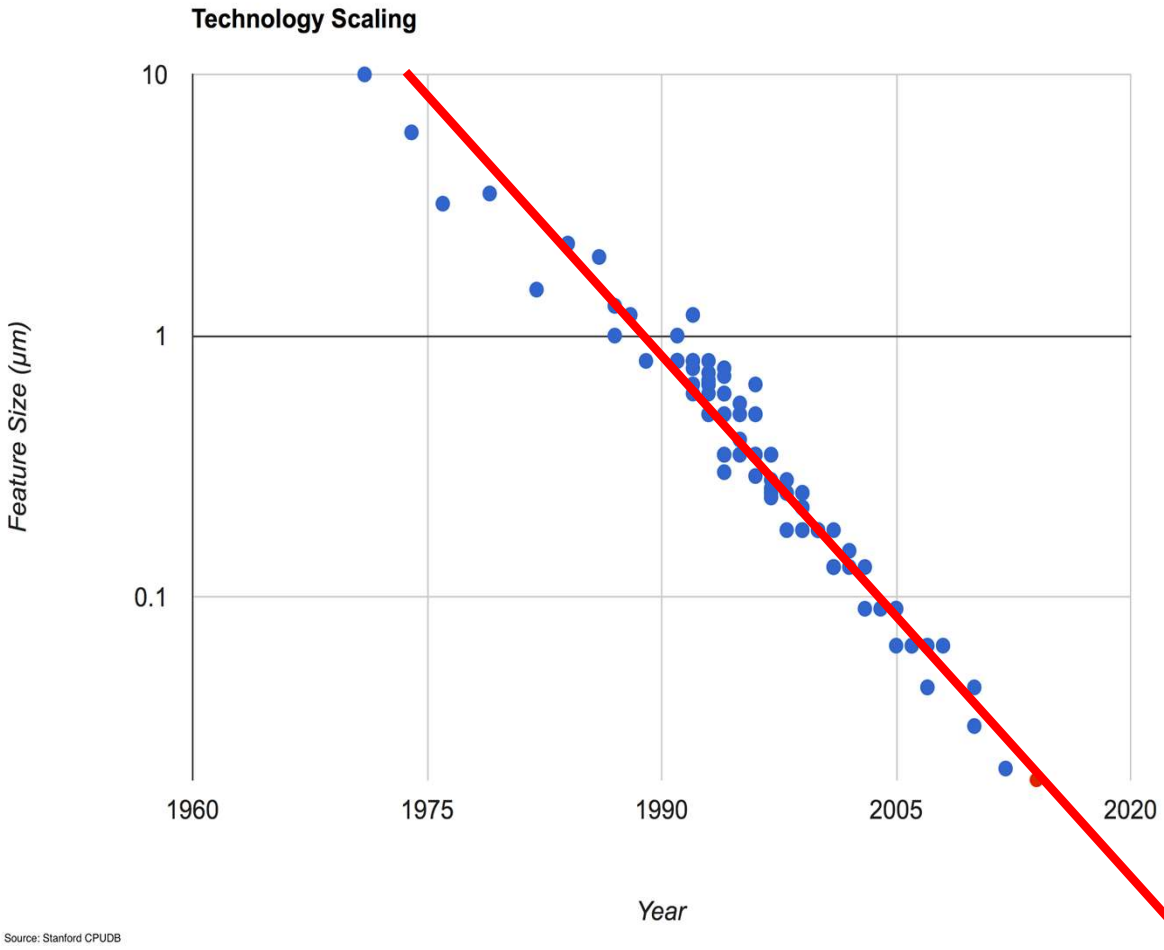
Why are GPUs important for you?



Why are GPUs different from CPUs?

The good news

- Transistor size over the years; still decreases

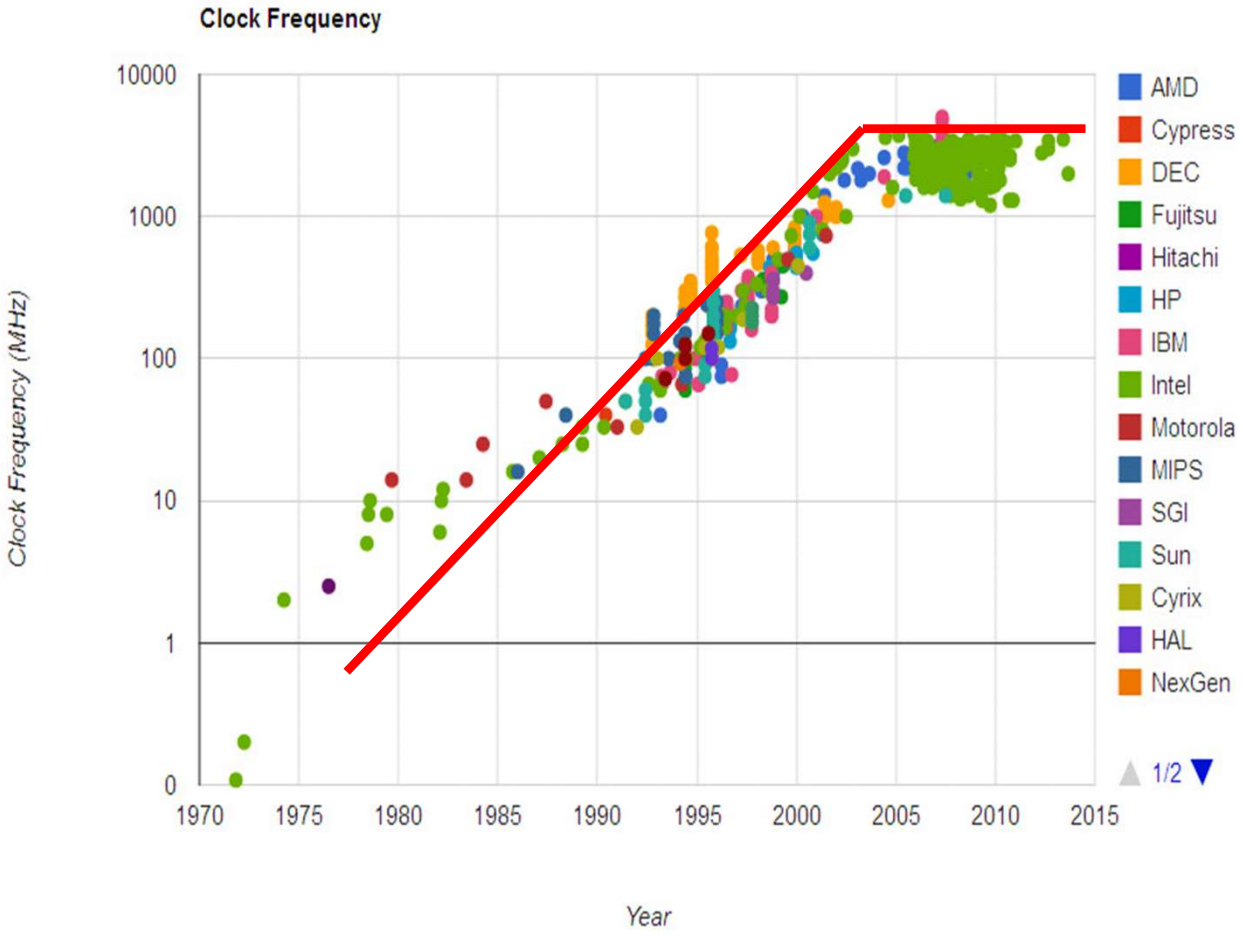


- Smaller
- Faster
- Less power
- More on chip

Currently best: 4 nm (Hopper

The bad news

- Clock speed over the years; stagnated since 2005



- Increasing over many years
- However, over the last decade the clocks speeds have essentially remained constant

Why not increase clock speed?

- Power = Frequency x Voltage^2
- Heat produced depends on power

Why not increase clock speed?

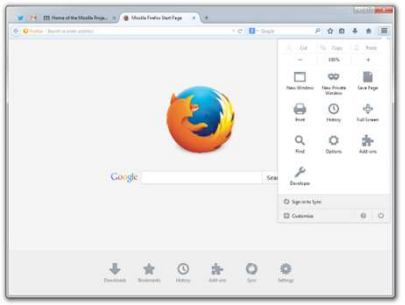
- Power = Frequency x Voltage^2
- Heat produced depends on power
- “Nard scaling”
  - Reduce transistor voltage to counter higher clock speed
  - Broke down in 2005 because of weakened current in the wires (need to distinguish 0 and 1)

Why not increase clock speed?

- Power = Frequency x Voltage^2
- Heat produced depends on power
- “Nard scaling”
  - Reduce transistor voltage to counter higher clock speed
  - Broke down in 2005 because of weakened current in the wires (need to distinguish 0 and 1)
- What matters today: # operations per Watt!
- Trade-off favors slower simpler cores
  - More operations per watt
  - Frequency kept low

CPUs (traditionally)

- Usual task of traditional CPUs
  - Desktop applications / OS
    - Lightly threaded
    - Lots of branches
    - Lots of (indirect) memory accesses



Mac OS

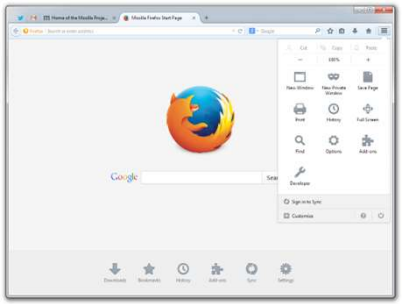
Windows 10

Microsoft Office

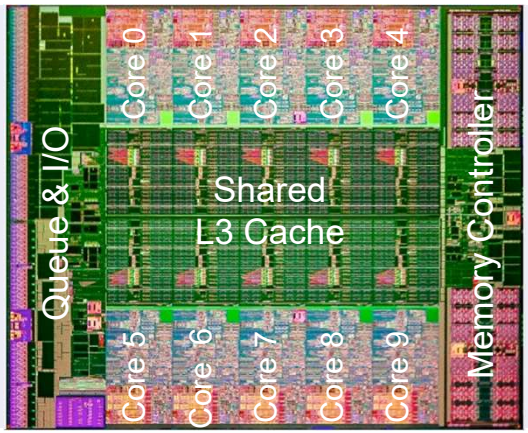
- CPUs try to minimize the time to complete a particular task – often to support user interaction!

CPUs (traditionally)

- Usual task of traditional CPUs
  - Desktop applications / OS
    - Lightly threaded
    - Lots of branches
    - Lots of (indirect) memory accesses



- CPUs try to minimize the time to complete a particular task – often to support user interaction!
- Complex control hardware
  - \+ Flexibility in performance
  - \+ Lightly parallel
  - − Expensive in terms of power



GPUs (traditionally)

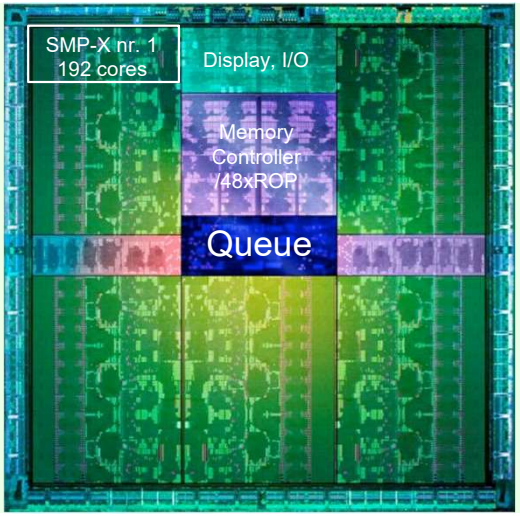
- GPUs are designed to compute pixels – fast!
  - Rendering video games in real-time
  - Play HD movies on smart phones
  - Render visual effects for movies…

- More concerned about the number of pixels per second than the latency of any particular pixel!

GPUs (traditionally)

- GPUs are designed to compute pixels – fast!
  - Rendering video games in real-time
  - Play HD movies on smart phones
  - Render visual effects for movies…

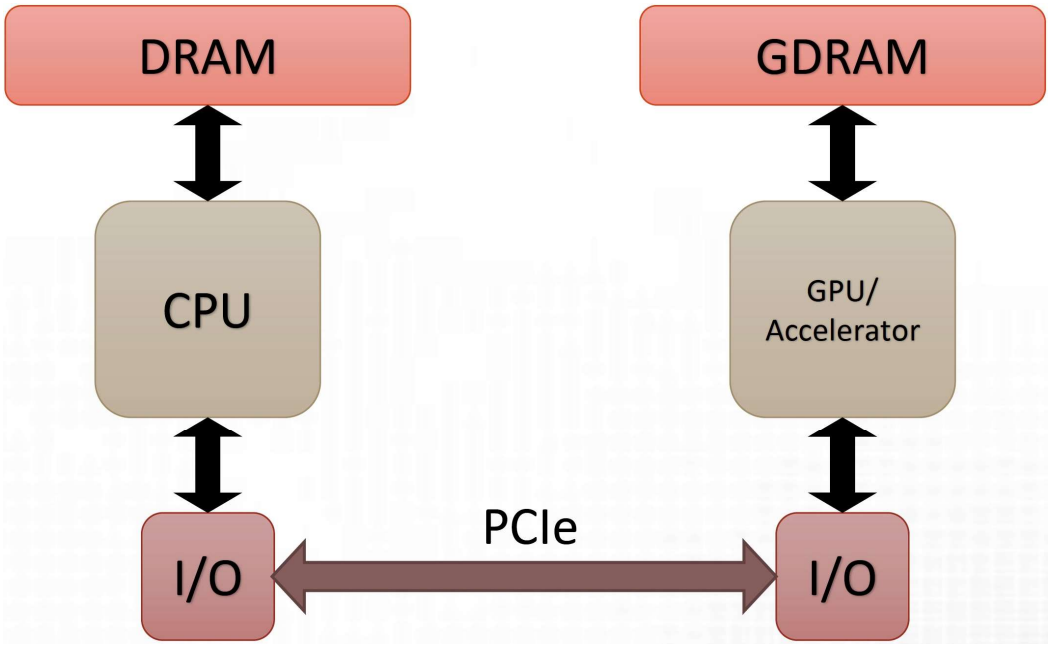
- More concerned about the number of pixels per second than the latency of any particular pixel!
- Simpler control hardware
  - \+ More transistors for computation
  - \+ Power efficient
  - − Highly parallel
  - − More restrictive in performance



GPUs as accelerators

GPUs as accelerators

- Problem: Still require OS, I/O, and scheduling
- Solution: “Hybrid system”
  - CPU provides management
  - Accelerators such as GPUs provide compute power

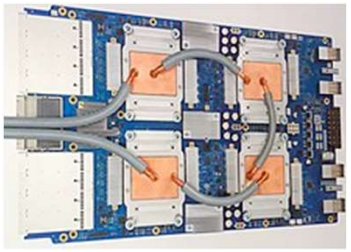


Types of accelerators

- GPUs
  - HPC high-end versions – Tesla branch
  - DP downgraded versions – Titan branch
  - Gamer versions – RTX branch

- Custom many-core processors
  - Japan / China

- TPUs (Tensor Processing Units)
  - Google AI accelerator – application-specific integrated circuit (ASIC)



Accelerators in Top500

DTU

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	8,730,112	1,102.00	1,685.65	21,100
2	**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
3	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	1,110,144	151.90	214.35	2,942
4	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148.60	200.79	10,096
5	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94.64	125.71	7,438

1st exascale

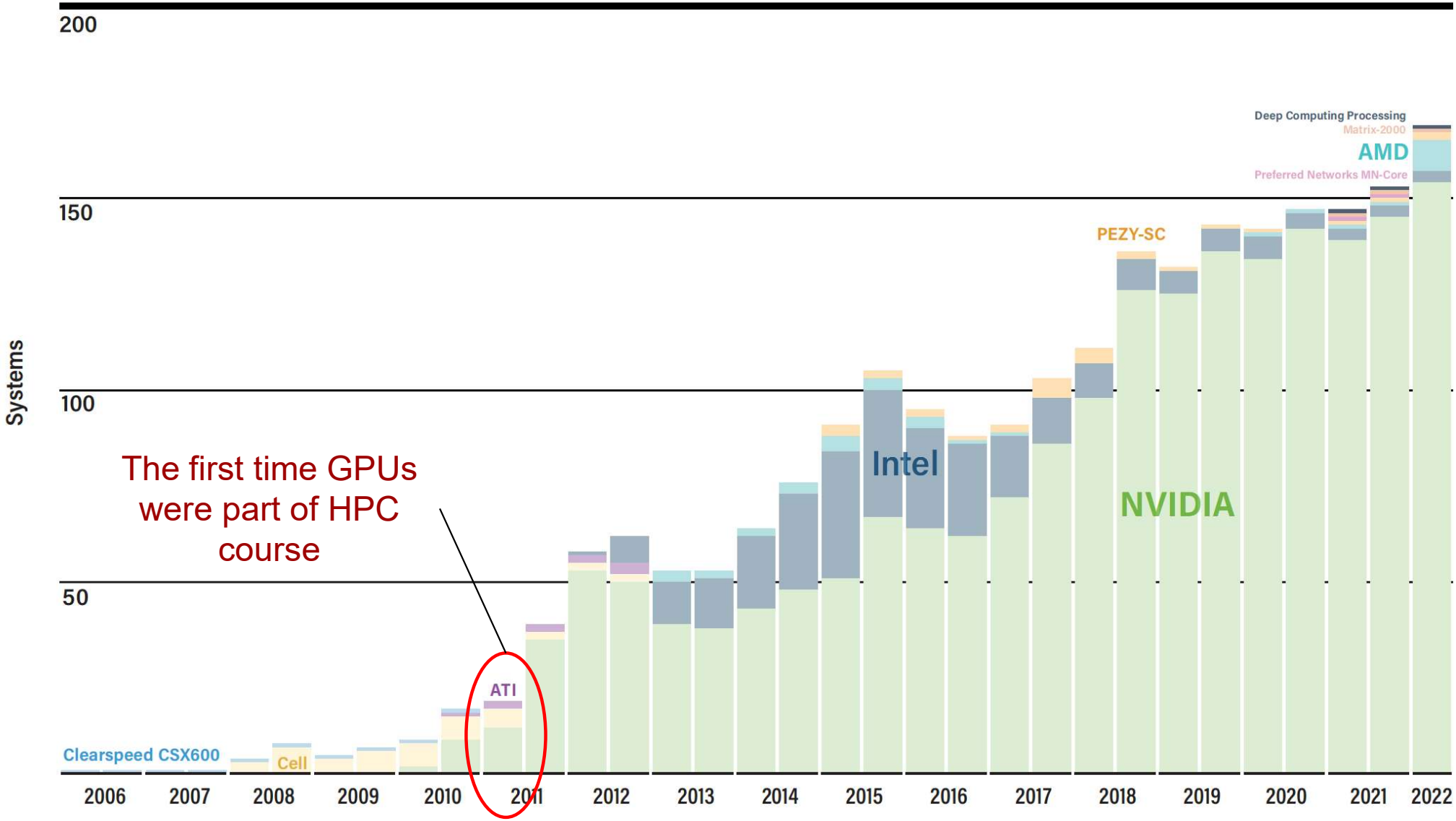
AMD Instinct (MI250X)

AMD Instinct (MI250X)

Nvidia Tesla (Volta)

Nvidia Tesla (Volta)

Accelerators in Top500



The first time GPUs were part of HPC course

Nvidia GPU architectures

- Five generations of high-performance GPUs:

	# GPUs	Name	Year	Architecture	CUDA cap.	CUDA cores	Clock MHz	Mem GiB	SP peak GFlops	DP peak GFlops	Peak GB/s
Kepler	5	Tesla K40c	2013	GK110B (Kepler)	3.5	2880	745 / 875	11.17	4291 / 5040	1430 / 1680	288
	8	Tesla K80c (dual)	2014	GK210 (Kepler)	3.7	2496	562 / 875	11.17	2796 / 4368	932 / 1456	240
Pascal	8	*TITAN X	2016	GP102 (Pascal)	6.1	3584	1417 / 1531	11.90	10157 / 10974	317.4 / 342.9	480
Volta	22	Tesla V100	2017	GV100 (Volta)	7.0	5120	1380	15.75	14131	7065	898
	12	Tesla V100-SXM2	2018	GV100 (Volta)	7.0	5120	1530	31.72	15667	7833	898
Ampere	6	Tesla A100-PCIE	2020	GA100 (Ampere)	8.0	6912	1410	39.59	19492	9746	1555
Hopper	-	Tesla H100-SXM5	2022	GH100 (Hopper)	9.0	8448	1650	79.18	38984	19492	3110

Source: http://www.hpc.dtu.dk/?page_id=2129

#Cores, Mem, and GB/s keep going up!

Clock freq. level off!

Peak increases 2x every ~3 years!

What is CUDA?

DTU

- [**C**ompute **U**nified **D**evice **A**rchitecture]
- A parallel computing standard and API proposed by Nvidia for general-purpose computations on CUDA-enabled GPUs
  - Priority #1: Make things easy (Sell GPUs)
  - Priority #2: Get performance
- Result: Simple to get started, but..
  - Requires expert knowledge to get best performance
- Scalable
- Well documented and free to use (!)

CUDA toolkit

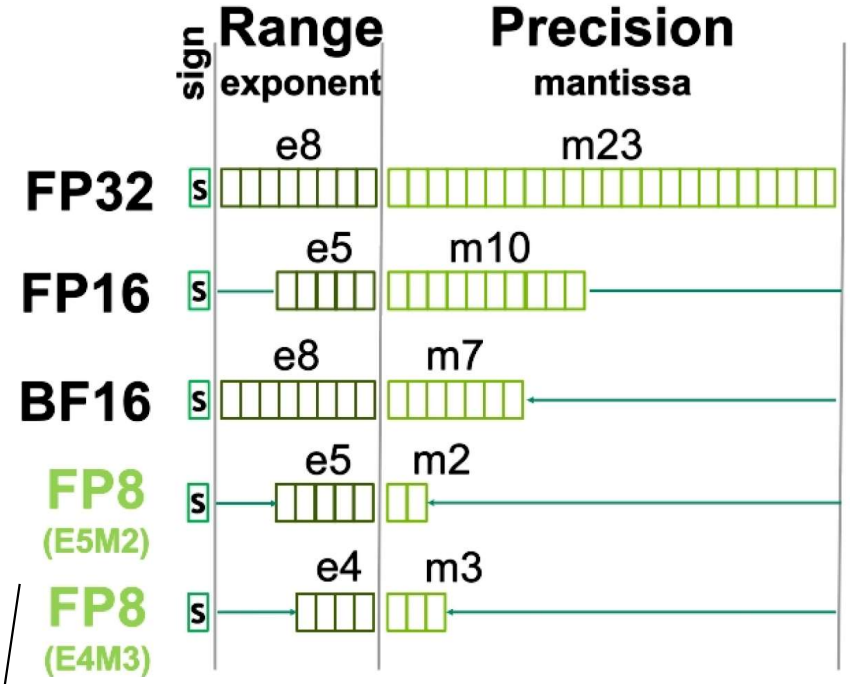
DTU

- The CUDA toolkit comprises a full framework of compiler, profiler, low- and high-level APIs, and highly tuned GPU libraries, e.g.
  - cuBLAS – CUDA Basic Linear Algebra Subroutines library
  - cuFFT – CUDA Fast Fourier Transform library
  - cuRAND – CUDA Random Number Generation library
  - cuSOLVER – CUDA based collection of dense and sparse direct solvers
  - cuSPARSE – CUDA Sparse Matrix library
  - cuDNN – CUDA Deep Neural Network library
  - …

Tensor Cores

Floating-point formats

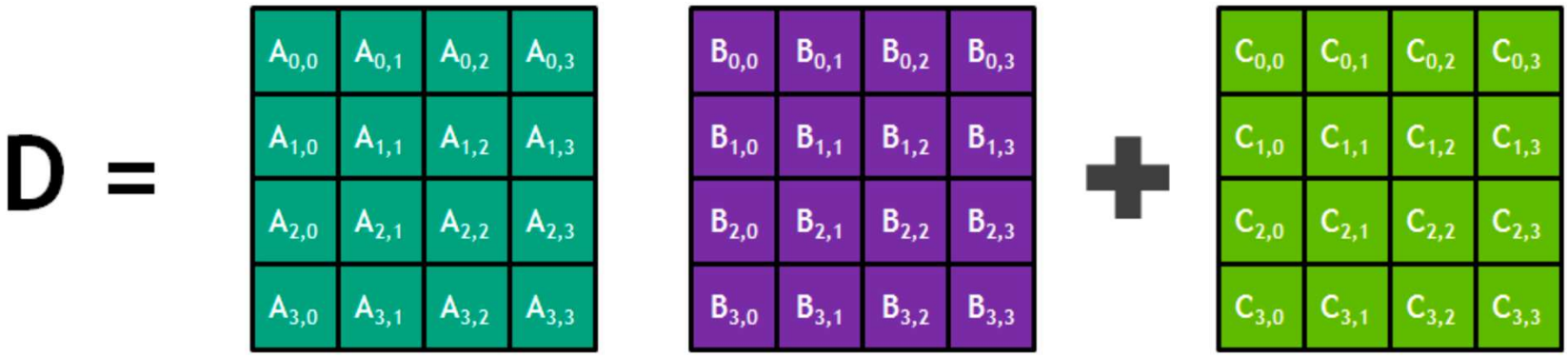
- Standard way to represent real numbers
  - Double (FP64), single (FP32), half (FP16), quarter (FP8)
- How floating-point numbers work
  - Exponent; determines range of values
  - Mantissa; determines the relative precision



FP8 Formats for Deep Learning (2022)
https://arxiv.org/abs/2209.05433

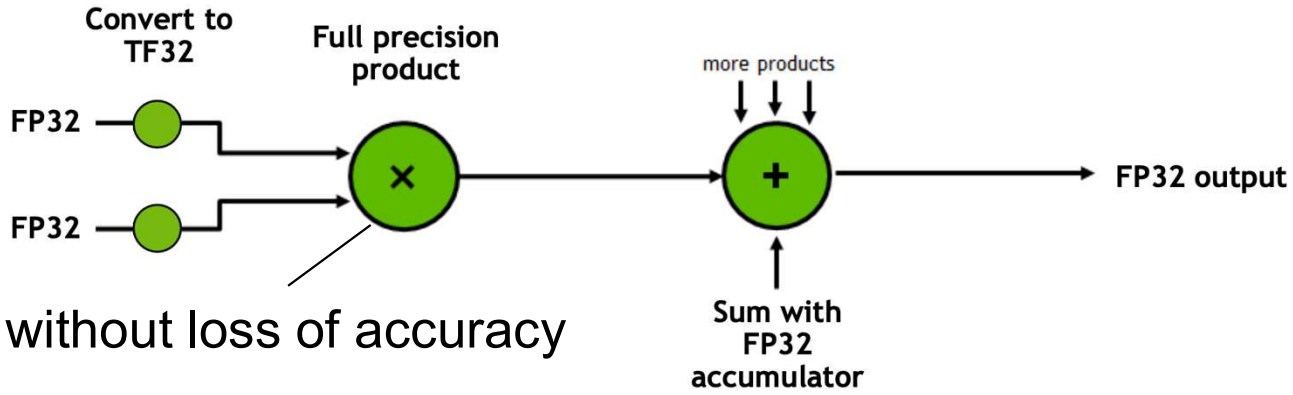
Tensor cores – what are they?

- Specialized math hardware units
  - Performs one mixed precision GEMM per cycle
    - V100 (FP16): 4 x 4 x 4
    - A100 (FP16): 8 x 4 x 8
    - H100 (FP8): 8 x 4 x 32
  - 8-16x higher math throughput (32x with sparsity) and saves 30% in power!
  - Same memory bandwidth pressure

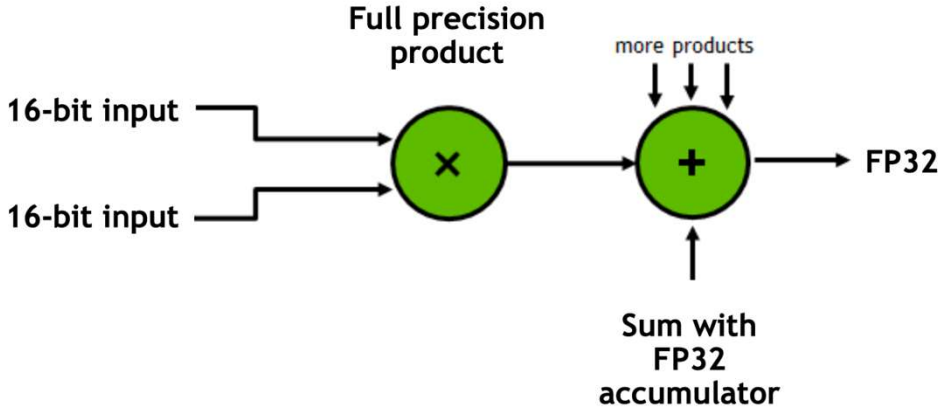


Flavors of Tensor cores

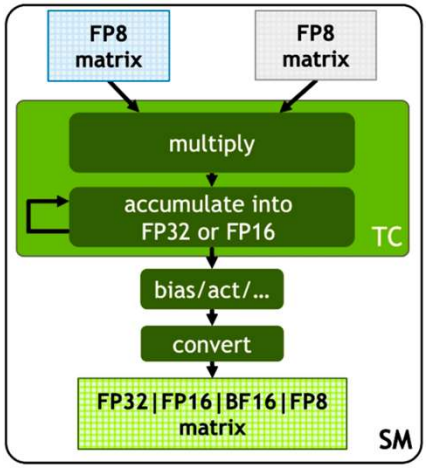
A100 TF32 mode (default)



V100 / A100 FP16 / BF16 mode

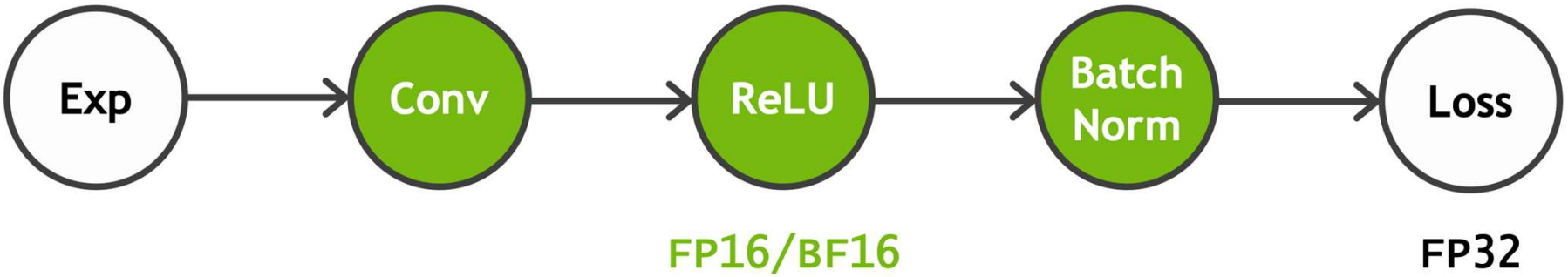


H100 FP8 mode (Automatic)



Mixed precision training

- Combines single-precision (FP32) with lower precision (FP16 / FP8) when training a network
  - Use lower precision where applicable (e.g. convolutions, matrix multiplies,...)
  - Keep certain operations in FP32
- Achieves the same accuracy as FP32 training using all the same hyper-parameters



Benefits of mixed precision

- Accelerates math-intensive operations with specialized hardware (GPU Tensor Cores)
  - FP16/BF16 have 16x higher throughput than FP32
- Accelerates memory-intensive operations
  - 16-bit: half the number of bytes to be read/written
- Reduces memory requirements
  - 16-bit: reduce storage of tensors by half
  - Enables training of larger models, larger mini-batches, larger inputs
- Hopper accelerates further using 8-bit formats

Automatic mixed precision (AMP)

- Nvidia AMP automates all the key considerations
  - Layer selection
    - Decides which operations compute in FP32 vs. FP16
  - Weight storage
    - Keeps model weights and updates in FP32
  - Loss scaling
    - Retains small gradient magnitudes for FP16 results
- Supported now in all mayor frameworks

PyTorch	Native support available in PT 1.6+ and NVIDIA Container 20.06+. Documentation can be found here: https://pytorch.org/docs/stable/amp.html https://pytorch.org/docs/stable/notes/amp_examples.html

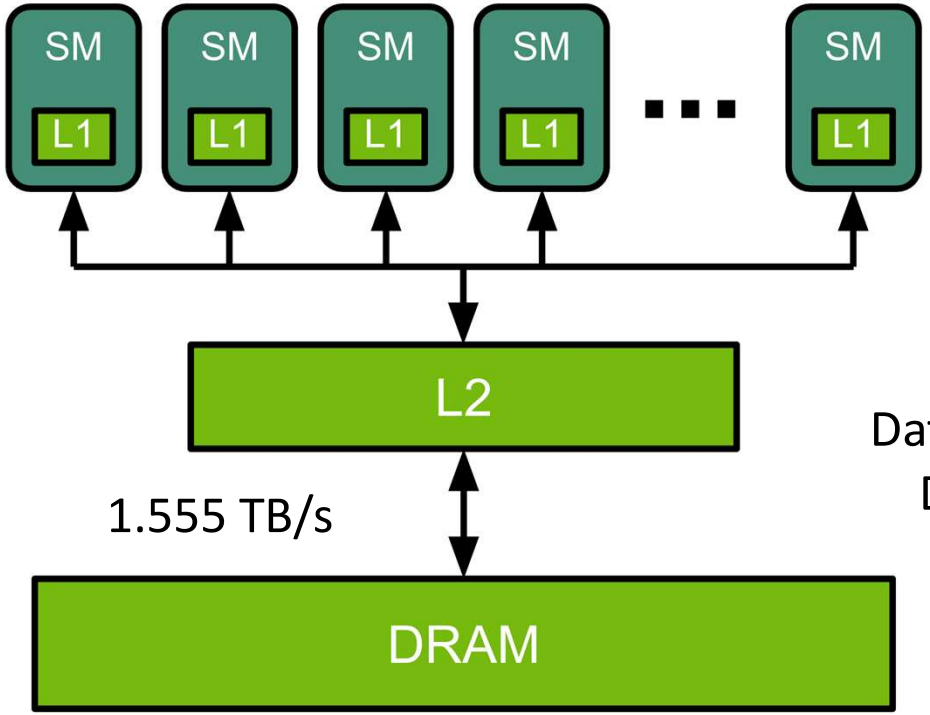
Performance

GPU performance basics

- Computing is done in parallel on SMs

19.5 TFlops for FP32 / 156 TFlops for TF32

A100:



108 Streaming multiprocessors
64 FP32 cores per SM
4 TF32 Tensor cores per SM

Data and instructions are accessed from DRAM through the shared L2 cache

GPU performance basics

- Compute (math) bound
  - Limited by # flops * time per flop

GFlops = # floating point operations / 10^9 / runtime

- Memory bound
  - Limited by # bytes moved / bandwidth

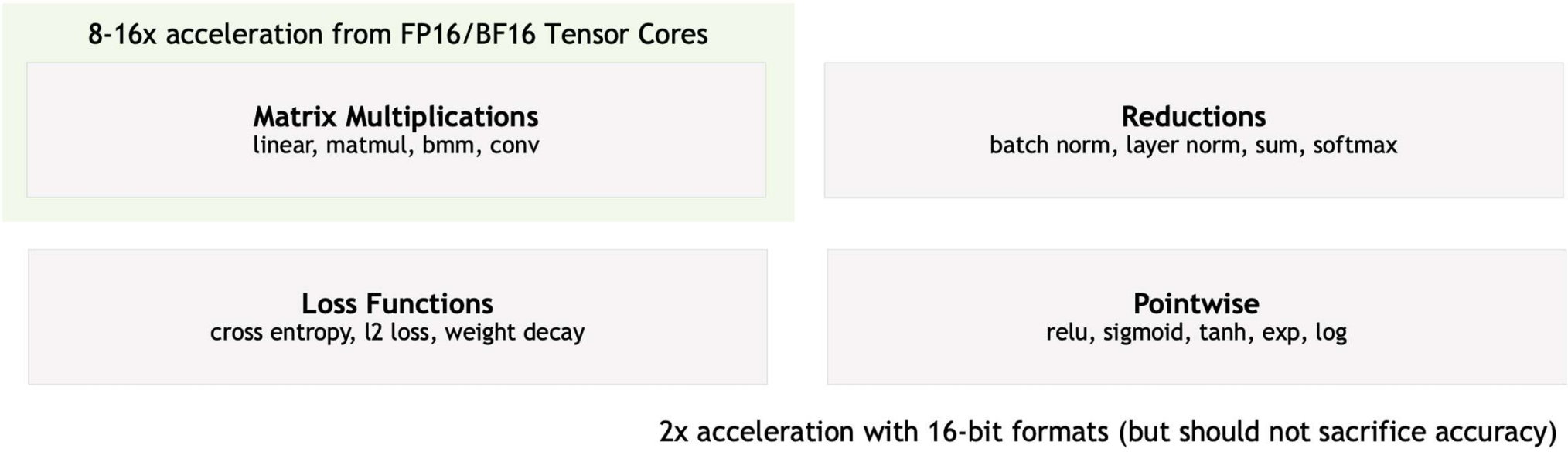
Bandwidth = (Bytes_read + Bytes_written) / 10^9 / runtime

- Arithmetic intensity (GFlops / Bandwidth) vs.

$$R_{FP16} = \frac{\text{peak performance}}{\text{max bandwidth}} = \frac{312\ \text{TFlops/s}}{1.555\ \text{TB/s}} = 201\ \text{Flops/B}$$

Speed-up from Tensor Cores

- Compute bound
  - Benifits from Tensor Cores: 8 – 16x speed-up
- Memory bound
  - Benifits from FP16 format: 2x speed-up



Ensure training is on GPUs

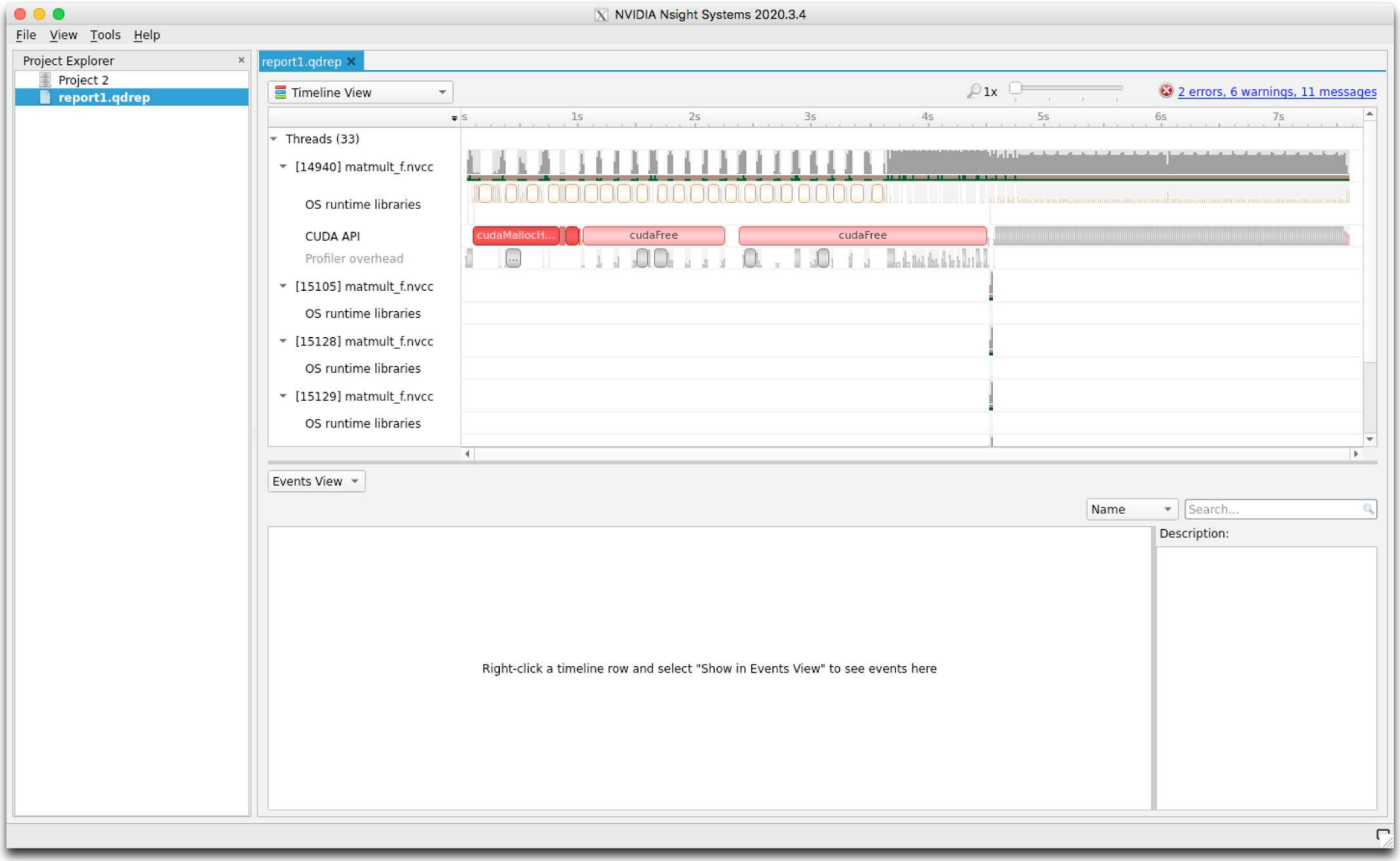
- Did your training time improve?
- Check use by hand
  - NVIDIA Nsight Compute (profiler for e.g. CUDA)

```
$ nv-nsight-cu-cli python train.py

> ==PROF== Profiling “volta_fp16_s884cudnn" - 1: 0%....50%....100% -
10 passes
> ...
```

  - NVIDIA Deep Learning Profiler `dlprof`

```
# Wrap training command line with DLPROF
$ dlprof --mode=pytorch python train.py

> Total Wall Clock Time (ns): 25812693595          # Time spent
> Total GPU Time (ns): 19092416468                 # Time spent on GPU
> Total Tensor Core Kernel Time (ns): 10001024991 # Time spent on TCs
```

NVIDIA Nsight systems



Ensure use of Tensor Cores / AMP

- Check with profilers
  - Kernel names that use Tensor Cores tend to look like

```
> ==PROF== Profiling "volta_fp16_s884cudnn...
> ==PROF== Profiling "ampere_h16816gemm...
> ==PROF== Profiling "cutlass_tensorop_f16_s1688gemm_f16_...
```

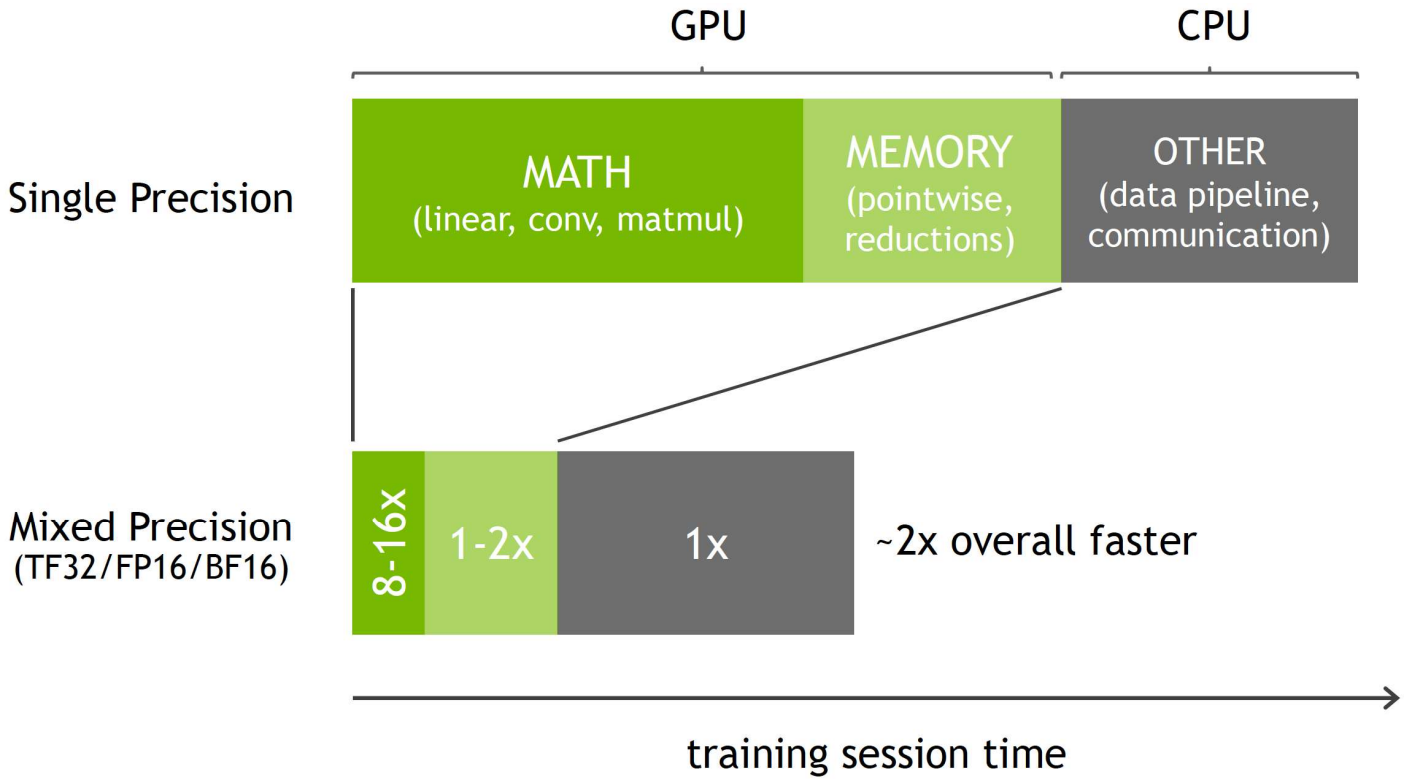
  - NVIDIA Deep Learning Profiler `dlprof`

```
> Total Tensor Core Kernel Time (ns): 10001024991 # Time spent on TCs
```

- Best performance if
  - Compute bound layers (gemms/convs) dominate training
  - Good parameter choices (dimensions multiples of 8)
  - Linear / conv layers large enough (GEMM dims > 128)

Overall training performance

- Be aware of overall performance bottleneck
  - Tensor cores speed up only part of the (GPU) work
  - The remaining parts may become the bottleneck



End of talk