



LESSON 11: Hardware (and Frameworks)

CARSTEN EIE FRIGAARD

SPRING 2022



"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E." — Mitchell (1997).

L11: Hardware (and Frameworks)

Agenda

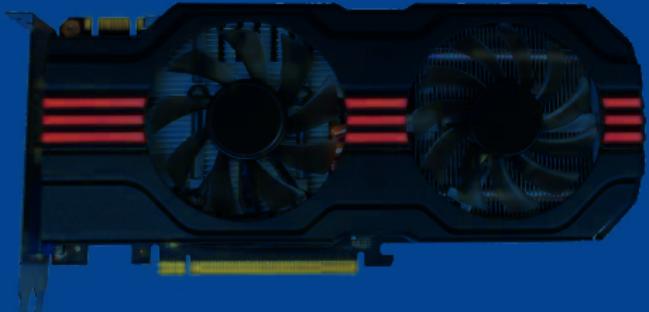
- ▶ Frameworks for Machine Learning,
- ▶ Hardware for Machine Learning,
 - ▶ CPUs,
 - ▶ GPUs,
 - ▶ TPUs,
 - ▶ Exotic Hardware,
 - ▶ Coin-mining, low-hash-rate.

FRAMEWORKS FOR MACHINE LEARNING

Frameworks for Machine Learning

name	note	
Sci-Kit Learn	(INRIA)	
Keras	interface	
Tensorflow	Google	
Pytorch	Facebook	
CNTK	Microsoft	
Apache MXNet	Amazon	
Core ML	Apple	
Caffe	(Berkeley)	
H2O	??	
Shogun	??	

HARDWARE FOR MACHINELEARNING



Hardware for Machine Learning

Methods and Terminology (SKIP most except μ, η)

Objective:

Why optimize using 'application specific' hardware?

▶ Effectiveness:

▶ cost of purchasing/operating systems,

$$\mu = \text{FLOPS}/\$$$

$$\eta = \text{FLOPS}/\text{Watt}$$

▶ cut-down developer waiting time,

▶ make modelling iterations fast (say minutes).

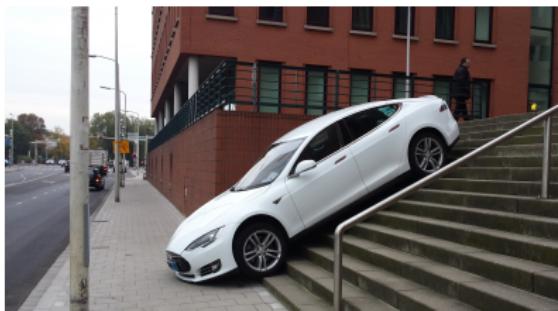
▶ Big-data:

▶ enable training on x-large data.

▶ Real-time constraints:

▶ inference (on visual data) in real-time,

▶ low-power constraints.



Hardware for Machine Learning

Methods and Terminology

Heterogeneous computing: systems that use more than one kind of processor or cores, say CPU + GPU.

Cluster computing: (loosely) or tightly connected computers that work together; can be viewed as a single system.

HPC: High-performance computing; a '**super-computer**' with high level of performance (vs. general-purpose computer).



Flynn's taxonomy:

SISD: single instruction, single data,

SIMD/SIMT: single instruction, multiple data/threads (data parallel); **this one is our objective,**

(**MISD:** multiple instructions, single data (fault tolerance)),

(**MIMD:** multiple instructions, multiple data (distributed)).

Methods and Terminology

Amdahl's law

What speedup can we expect when optimizing/parallelizing a program?

$$S = \frac{1}{(1 - p) + p/s}$$

where

S : total program speedup,

p : fraction of the program that can be run in parallel,

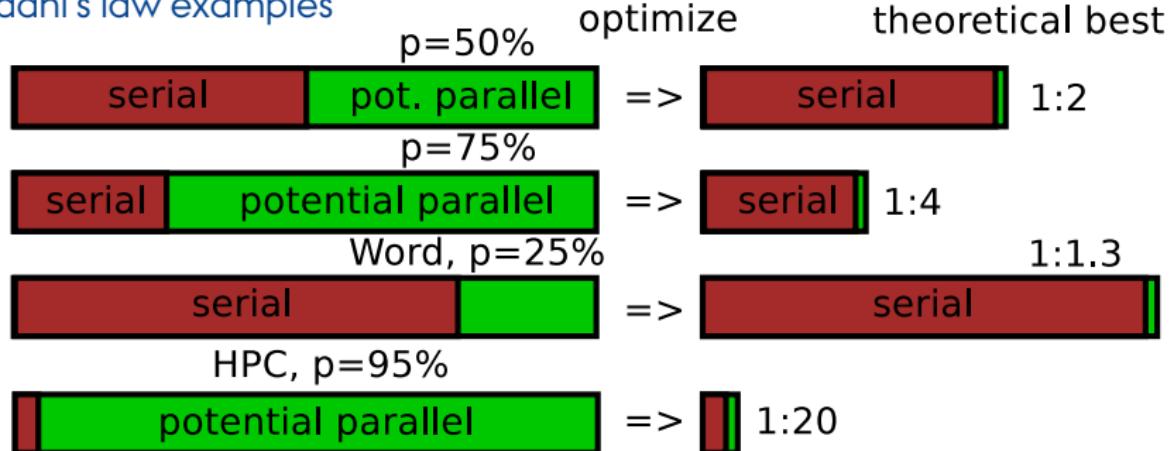
s : parallel fraction speedup factor.

If we take $s = \infty$ the theoretical max. total speedup will be

$$S_{\max} = \frac{1}{(1 - p) + p/\infty} = \frac{1}{1 - p}$$

Methods and Terminology

Amdahl's law examples



And there is seldom need to optimize unless $S_{\max} \gg 2$.

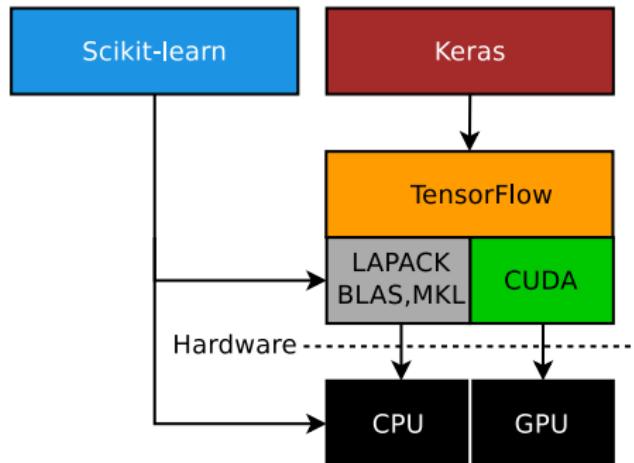
A **1:5 speedup** is a good, ad-hoc compromise btw. a fast program and a costly optimization dev. phase.

What does Donald E. Knuth say about premature optimization?

"Root of all evil!"



RESUMÉ: Keras and Tensorflow



GP-GPU: General-Purpose Graphics Processing Unit...or just **GPU**.

CUDA: Compute Unified Device Architecture, API for SIMD/SIMT on GPU,

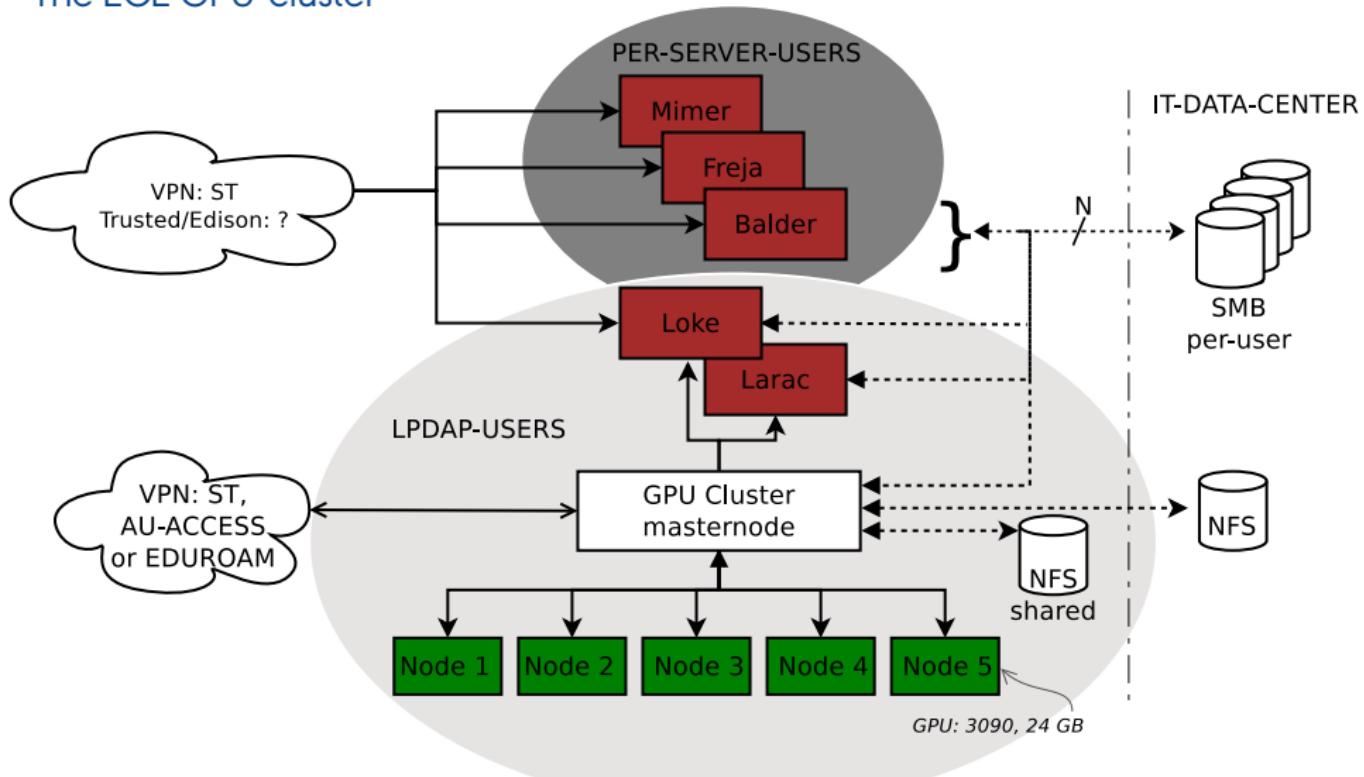
LAPACK: Linear Algebra Package, numerical linear algebra lib, with roots in FORTRAN 77,

BLAS: Basic Linear Algebra Subprograms; vector/matrix lib,

MKL: Math Kernel Library; fast Intel-arch optimized math lib.

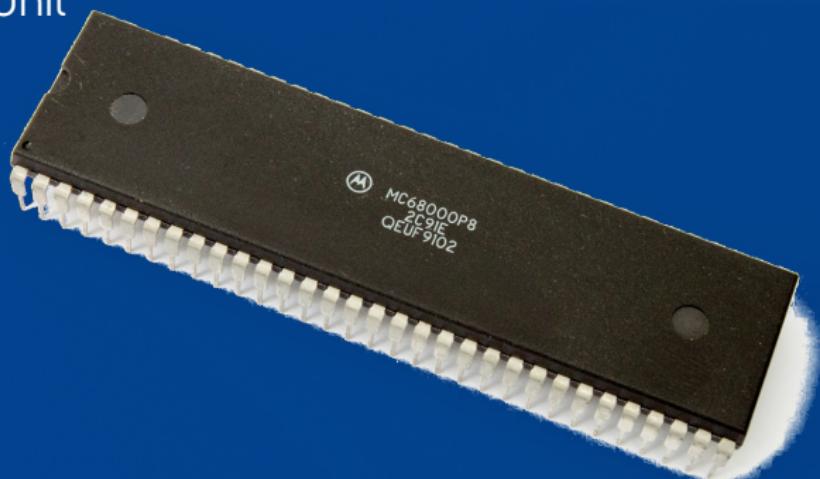
High-Performance-Computing (HPC)

The ECE GPU-cluster



CPUS

Central Processing Unit



CPUs

Build Tensorflow from source (SKIP)

- ▶ for specific architecture, say ARM,
 - ▶ or for HPC optimization for all CPU feature
- > lscpu

```
Architecture: x86_64
Model name: Intel(R) Core(TM) i7-6600U CPU @ 2.60GHz
Flags: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr
      pge dts acpi mmx fxsr fma..sse sse2..sse4_1sse4_2..
```



Using Docker and pulling TF from GIT + a lot of scripting!

```
> git clone https://github.com/tensorflow/tensorflow
> git checkout 1.12
(lots of scripting and pain..)
> bazel build -copt=-mfma -copt=-msse4.2 tensorflow
```

Howtos:

<https://www.tensorflow.org/install/source>

<https://www.pugetsystems.com/labs/hpc/Build-TensorFlow-CPU-with-MKL-and-Anaconda-Python-3-6-using-a-Docker-Container-1133>

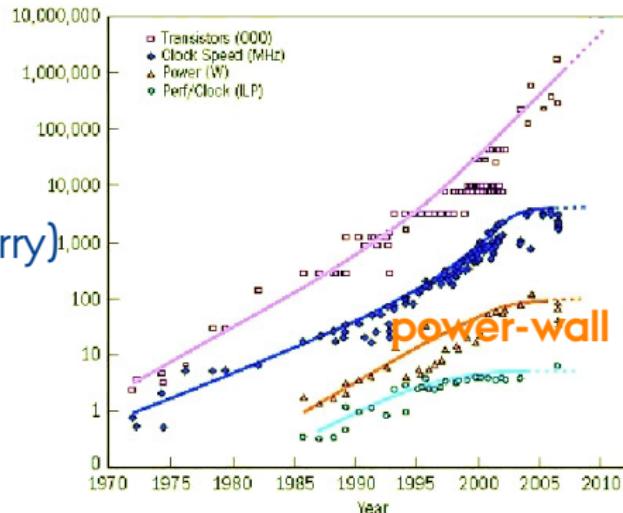
<https://www.pugetsystems.com/labs/hpc/Build-TensorFlow-GPU-with-CUDA-9-1-MKL-and-Anaconda-Python-3-6-using-a-Docker-Container-1134>

CPUs

SIMD/Data parallel compute using CPU

CPU architectures:

- ▶ i386/AMD64,ARM (Raspberry)
- ▶ all CPU types: lots of cores; end of Moore's Law, 'Singularity University', problem w. exp. growth?



Scikit-learn problem with SIMD:

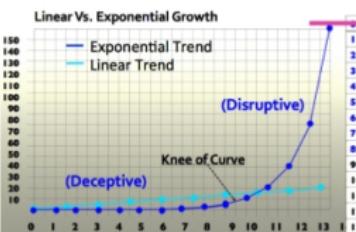
- ▶ no ML-algo GPU hardware enabled,
- ▶ many ML-algo not even multicore aware!

Keras/TensorFlow:

- ▶ CPU optimization of TF possible,
- ▶ (GPU enabled TF possible, using CUDA + cudNN).

Exponential Technologies

- Artificial Intelligence
- Robotics
- Biotech
- Manufacturing
- Computation / Networks
- Synthetic Biology
- Digital Medicine

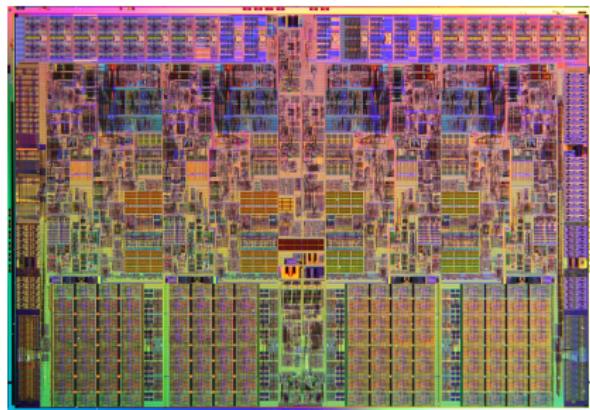


GP-GPUS

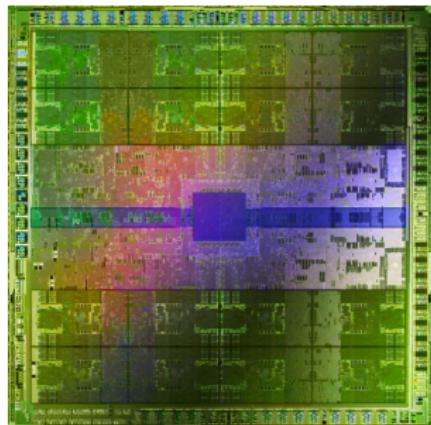
General-Purpose Graphical Processing Unit



CPUs vs GPUs



Nehalem CPU
die size: $\sim 700 \text{ mm}^2$
transistors: $\sim 2.3 \cdot 10^9$



Fermi GPU
die size: 520 mm^2
transistors: $\sim 3 \cdot 10^9$

Nvidia arch. transistor counts

Pascal	$\sim 15 \cdot 10^9$
Turing	$\sim 19 \cdot 10^9$
Volta	$\sim 21 \cdot 10^9$
Ampere	$\sim 28 \cdot 10^9$ (GPU 3090, 8nm)

CPUs vs GPUs

So many transistors, but how many for the ALUs/FPUs?

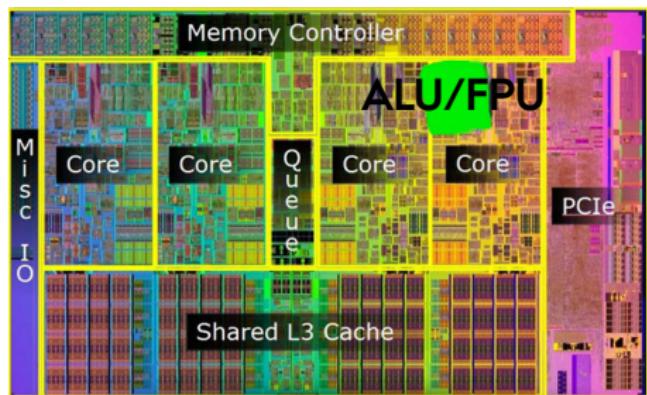
What makes GPUs such excellent HW for Machine Learning?

ALU: Arithmetic logic unit

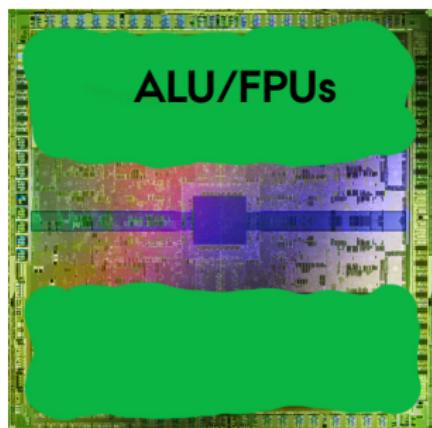
FPU: Floating-point unit

Memory Controller: six controllers on GPU,

CPU: lots of speculative execution; waste of transistors.



CPU (type?)
with one ALU/FPU marked



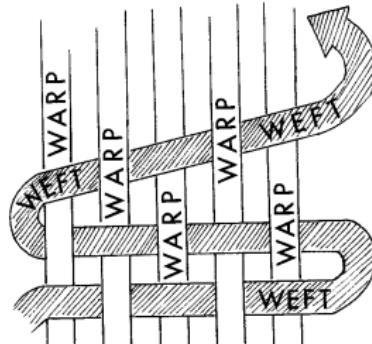
Fermi GPU
ALUs/FPUs all over

GPUs

Fundamental Problems with the GPUs Hardware (SKIP most, except WARP)

GPUs had several *Achilles heels* related to its hardware, many of them addressed in the latest Vola V100 architecture:

- ▶ coding problems graphically, now CUDA,
- ▶ no STACK, now added,
- ▶ no CACHE (or Texture only), now both L₁ and L₂ cache,
- ▶ distinct GPU memory, now UNIFIED memory,
- ▶ SIMT WARP-bunch of 32-threads, now true SIMD,



GPUs

GPU architecture: Core Design, Streaming Multiprocessors



GPUs

Core Design for a SM
(Streaming Multiprocessor)

Volta SM design
(new gen. GPU):

FP64/32:FPUs
INT: ALUs
TENSOR CORES: ?

Ampere, RTX, 3090:
Raytracingkerner: ?



GEFORCE RTX 3090	
NVIDIA CUDA® kerner	10496
Høj CPU-hastighed (GHz)	1.70
Normal CPU-hastighed (GHz)	1.40

Annotations highlight specific components:

- Raytracingkerner 2. generation:** Circled in green, referring to the second-generation ray tracing kernels.
- Tensor Cores 3. generation:** Circled in green, referring to the third-generation tensor cores.
- Standard hukommelseskonfiguration:** Standard memory configuration.
- 24 GB GDDR6X:** Circled in green, indicating the total memory capacity.
- Hukommelsesgrænsefladens bredde:** Width of the memory interface.
- 384-bit:** Bit width of the memory interface.

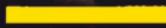
GPUs

Demo: RTX and raytracing

2070 S 99 %
VRAM 5,461 MB
M.CLOCK 7001 MHz
C.CLOCK 1920 MHz
PWR 212.4 W
TEMP. 67 °C
32GB RAM 11,561 MB
I7 9700K 15 %
CLOCK 5000 MHz
TEMP. 51 °C
low 0.1% 55
low 1% 58
FPS avg 67

FRAME TIME 15.6 ms

FPS 65 fps

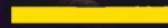


Original

2070 S 98 %
VRAM 5,453 MB
M.CLOCK 7001 MHz
C.CLOCK 1920 MHz
PWR 210.5 W
TEMP. 67 °C
32GB RAM 11,564 MB
I7 9700K 22 %
CLOCK 5000 MHz
TEMP. 56 °C
low 0.1% 54
low 1% 56
FPS avg 65

FRAME TIME 16.3 ms

FPS 61 fps

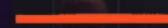


Sharpen

2070 S 99 %
VRAM 5,277 MB
M.CLOCK 7001 MHz
C.CLOCK 1980 MHz
PWR 205.6 W
TEMP. 66 °C
32GB RAM 11,559 MB
I7 9700K 15 %
CLOCK 5000 MHz
TEMP. 58 °C
low 0.1% 31
low 1% 31
FPS avg 36

FRAME TIME 31.0 ms

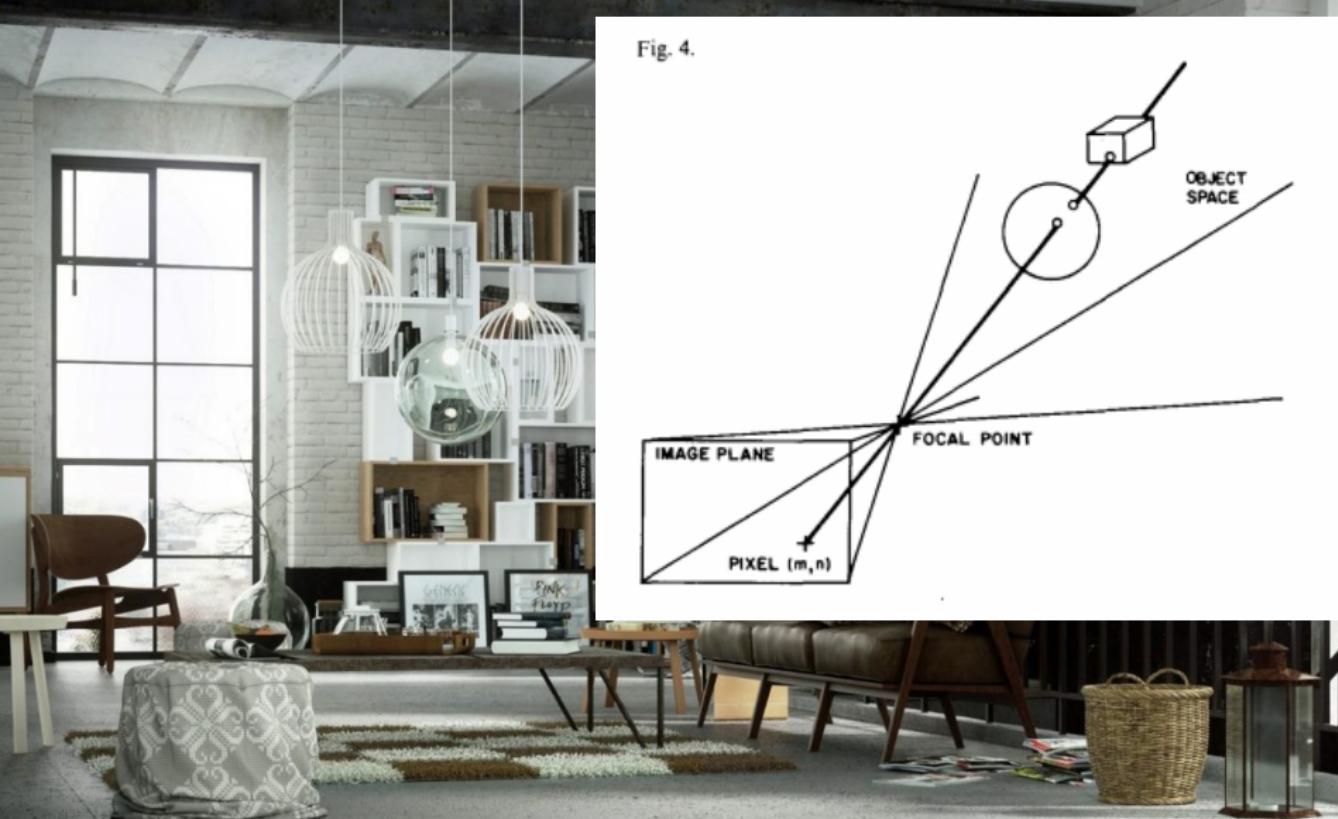
FPS 31 fps



Ray Tracing+Sharpen

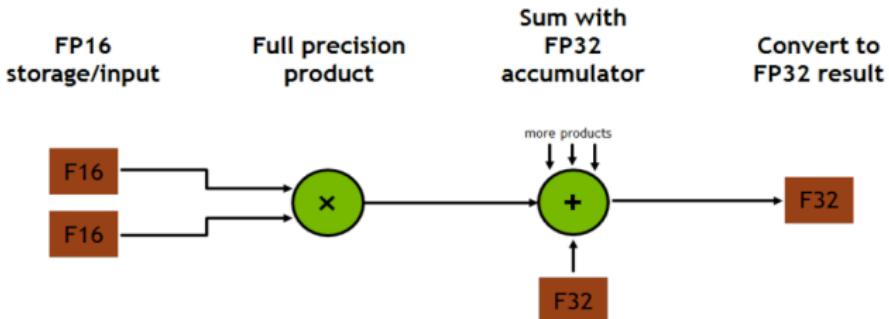
GPUs

Tensor cores: Raytracing vs Rasterization on GPUs



GPUs

Tensor Cores
(SKIP most)



$$D = \begin{pmatrix} A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\ A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix}_{\text{FP16 or FP32}} \begin{pmatrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\ B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\ B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\ B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3} \end{pmatrix}_{\text{FP16}} + \begin{pmatrix} C_{0,0} & C_{0,1} & C_{0,2} & C_{0,3} \\ C_{1,0} & C_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,0} & C_{2,1} & C_{2,2} & C_{2,3} \\ C_{3,0} & C_{3,1} & C_{3,2} & C_{3,3} \end{pmatrix}_{\text{FP16 or FP32}}$$

Tesla V100's Tensor Cores deliver up to **125 Tensor TFLOPS** for training and inference applications.

[volta-architecture-whitepaper.pdf]

Analog: DSD: half-adder, full-adder, ripple-carry-adder with n-bit multiplier based on adders (scales n-bit²)..

HPC Top500

Best High-Performance Computer

[<https://www.top500.org/list/2007/06/>]

[https://en.wikipedia.org/wiki/List_of_Nvidia_graphics_processing_units]



HOME	LISTS ▾	STATISTICS ▾	RESOURCES
------	---------	--------------	-----------

Home » Lists » TOP500 » June 2007

JUNE 2007

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	BlueGene/L - eServer Blue Gene Solution, IBM DOE/NNSA/LLNL United States	131,072	280.6	367.0	1,433
2	Jaguar - Cray XT4/XT3, Cray/HPE DOE/SC/Oak Ridge National Laboratory United States	23,016	101.7	119.3	
3	Red Storm - Sandia/ Cray Red Storm, Opteron 2.4 GHz dual core, Cray/HPE NNSA/Sandia National Laboratories United States	26,544	101.4	127.4	

GPUs

Turing Tensor Core GPU (this is not a commercial!)

GPU Acceleration Core Mainstream

NVIDIA T4 enterprise GPUs supercharge the world's most trusted mainstream servers, bla bla...

Its low-profile, 70-watt (W) design is designed for enterprise data centers, making it ideal for demanding workloads including machine learning, deep learning, and virtual desktop infrastructure (VDI). It also offers the efficiency of smaller PCIe form factors for maximum system options and convenience.

Inference Performance

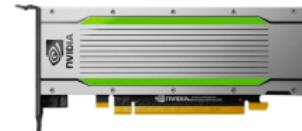


Comparisons made of one NVIDIA Tesla T4 GPU and servers with a dual-socket Xeon Gold 6140 CPU.

Training Performance



Comparison made between dual NVIDIA Tesla T4 GPUs and servers with a dual-socket Xeon Gold 6140 CPU.



SPECIFICATIONS

GPU Architecture	NVIDIA Turing
NVIDIA Turing Tensor Cores	320
NVIDIA CUDA® Cores	2,560
Single-Precision	8.1 TFLOPS
Mixed-Precision (FP16/FP32)	65 TFLOPS
INT8	130 TOPS
INT4	260 TOPS
GPU Memory	16 GB GDDR6 300 GB/sec
ECC	Yes
Interconnect Bandwidth	32 GB/sec
System Interface	x16 PCIe Gen3
Form Factor	Low-Profile PCIe
Thermal Solution	Passive
Compute APIs	CUDA, NVIDIA TensorRT™, ONNX

GPUs

When is the GPU faster than the CPU for NN?

GPU slower for CPU for a three-layer NN + MNIST, why?

- ▶ GPU needs a reasonable amount of trainable parameters + data to beat the CPU!

`model.summary():`

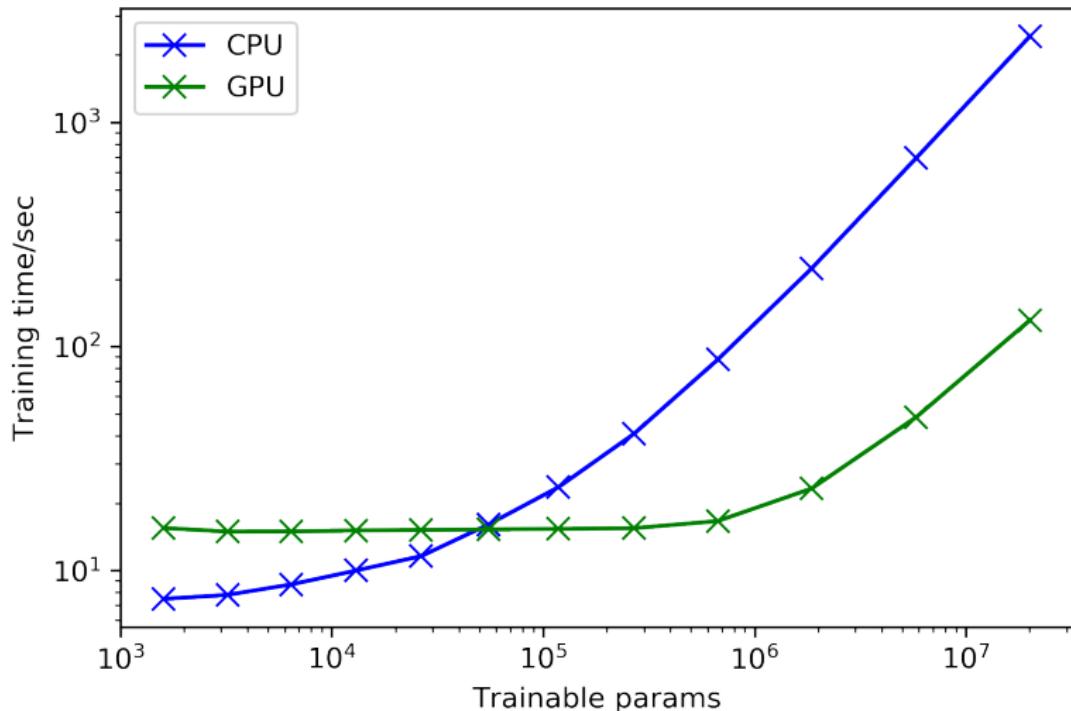
```
1 # i=12, n=4096
2 #
3 # Layer (type)          Output Shape       Param #
4 # =====
5 # dense_46 (Dense)     (None, 4096)        3215360
6 #
7 # dropout_31 (Dropout) (None, 4096)        0
8 #
9 # dense_47 (Dense)     (None, 4096)        16781312
10 #
11 # dropout_32 (Dropout) (None, 4096)        0
12 #
13 # dense_48 (Dense)     (None, 10)          40970
14 # =====
15 # Total params: 20,037,642
16 # Trainable params: 20,037,642
17 # Non-trainable params: 0
```

GPUs

Actual test on the GPU-server

CPU vs GPU on MNIST for a three layer NN with dropout...

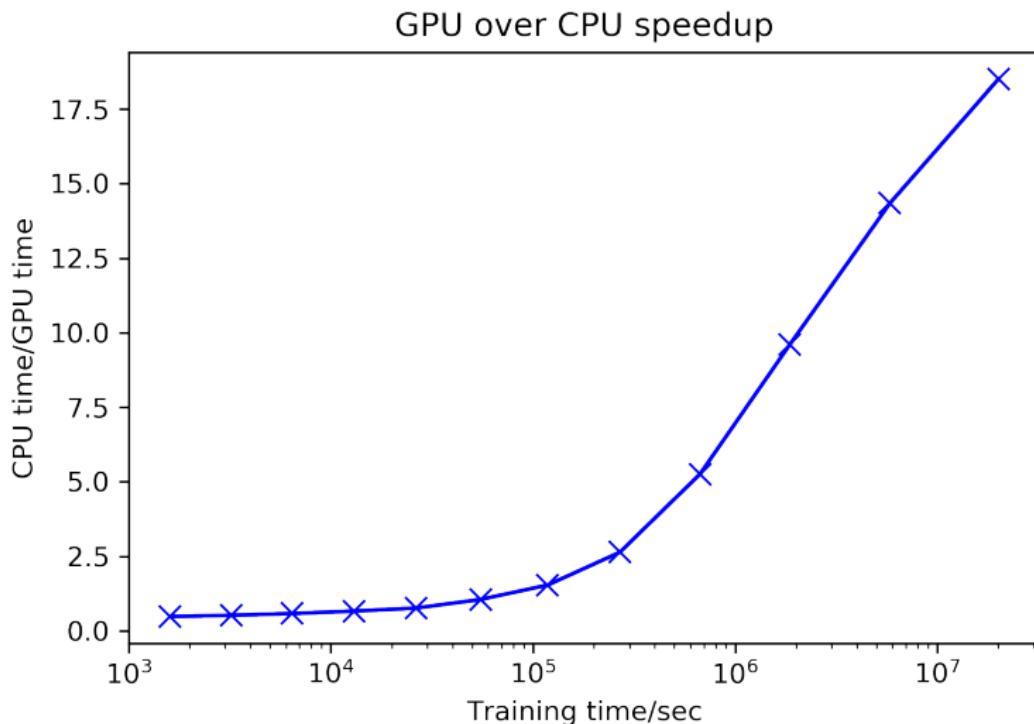
Training time-vs-Trainable params



GPUs

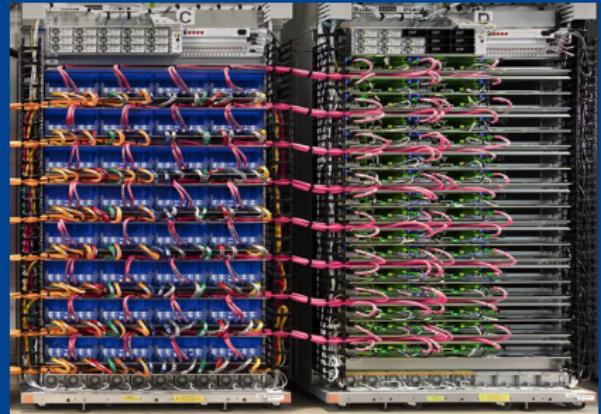
Actual test on the GPU-server

CPU vs GPU on MNIST for a three layer NN with dropout...



TPUS

Tensor Processing Units



TPUs

Tensor Processing Units

Custom ASICs by Google



Cloud TPU v3

Launched in 2018

Inference and training



Dev Board

A development board to quickly prototype on-device ML products. Scale from prototype to production with a removable system-on-module (SoM).

→ Datasheet
→ Get started guide

\$149.99

Buy



TPU v2

Launched in 2015

Inference only

Launched in 2017
Inference and training

Dev board with Google Edge TPU ML accelerator coprocessor

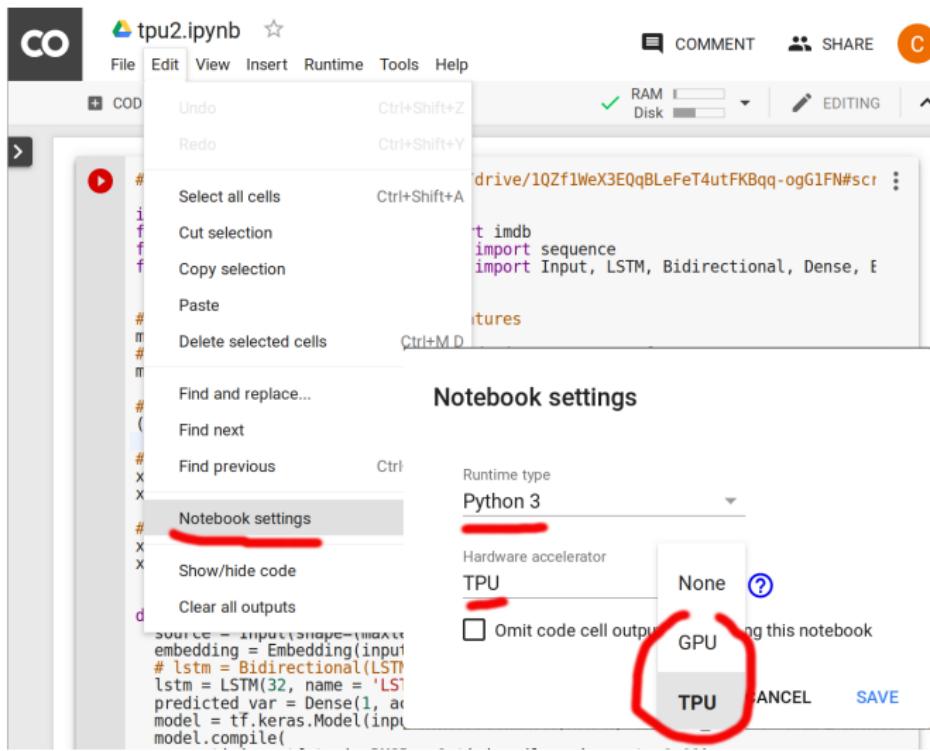
[\[https://coral.withgoogle.com/products/dev-board/\]](https://coral.withgoogle.com/products/dev-board/)



TPUs

Access to TPUs (SKIP)

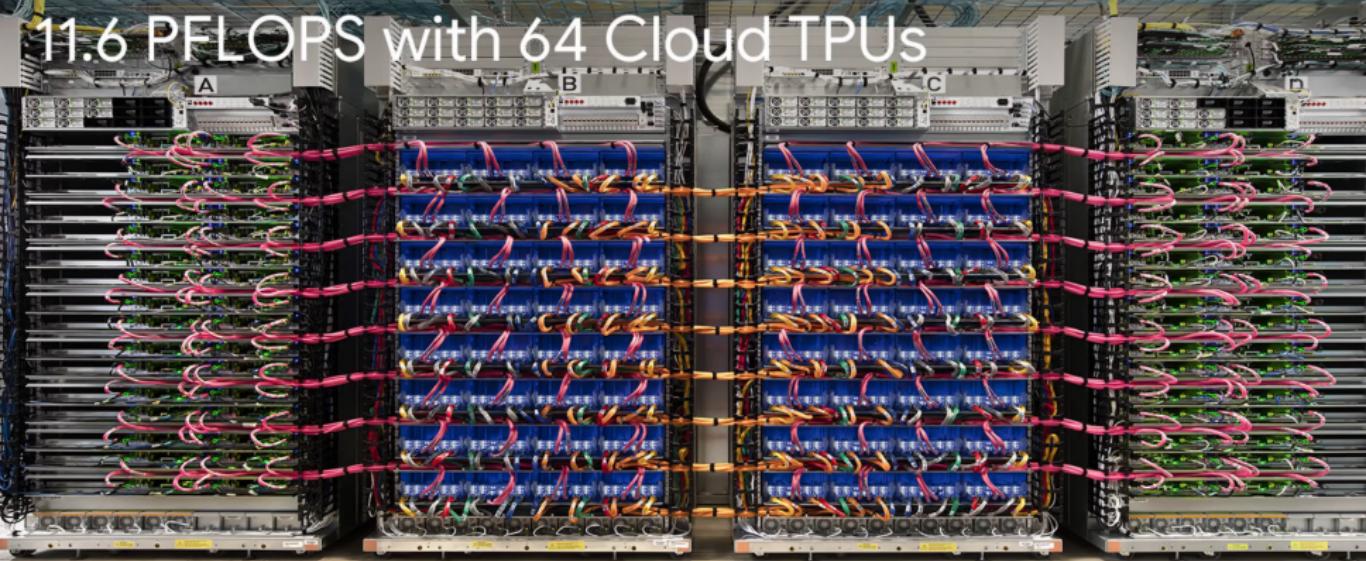
Free Jupyter Notebook environment with access to TPUs:
<https://colab.research.google.com>



TPUs

TPU Cloud

TPU v2 Pod: Google's HPC cluster for ML
11.6 PFLOPS with 64 Cloud TPUs



[<https://storage.googleapis.com/nexttpu/index.html>]

TPUs vs GPUs

Performance, TPUs vs GPUs, who wins? (SKIP most)

- ▶ Huge advantage for TPU performance-per-watt,
- ▶ Colab performance:
inconclusive (TPU part does not work yet),
- ▶ TPU only for inference?

	K80 2012	TPU 2015	P40 2016
Inferences/Sec <10ms latency	1/13 TH	1X	2X
Training TOPS	6 FP32	NA	12 FP32
Inference TOPS	6 FP32	90 INT8	48 INT8
On-chip Memory	16MB	24 MB	11 MB
Power	300W	75W	250W
Bandwidth	320 GB/S	34 GB/S	350 GB/S

[<https://www.extremetech.com/computing/247403-nvidia-claims-pascal-gpus-challenge-googles-tensorflow-tpu-updated-benchmarks>]

EXOTIC HARDWARE

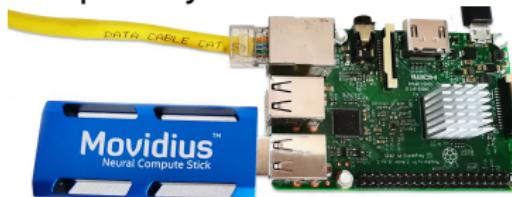


Exotic Hardware

- ▶ Intel Phi multicore CPU, 64 i386 cores:



- ▶ Raspberry PIs + Intel Movidius stick



- ▶ FPGAs



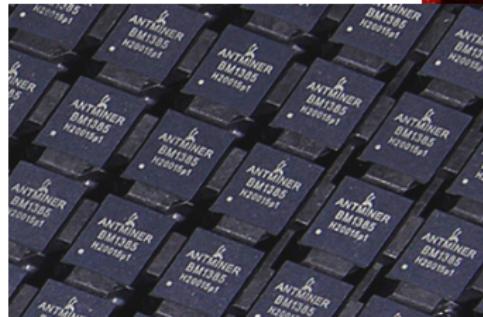
ASICs vs GPUs

Bitcoin-mining and Low-hash-rate GPUs

AntMiner S7: based on ASICs.

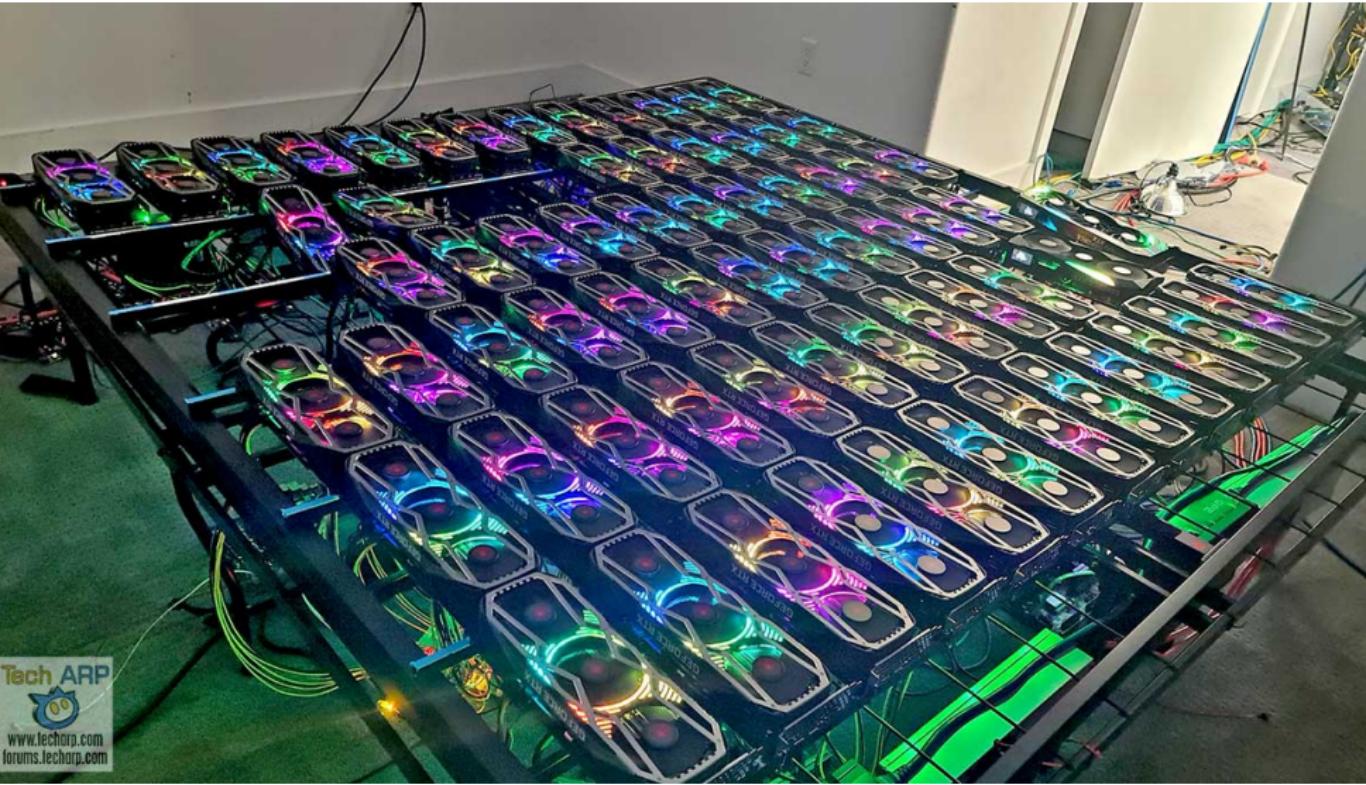
Some specs:

- ▶ hash rate: 4.8 THash/s
- ▶ chips per unit: 162 x BM1385
- ▶ power consumption: **1210 W**
- ▶ power efficiency: 0.25 W/GHash
- ▶ price: \$479.95 ~ 2880,- DKK
- ▶ production: **0.16 bitcoin/month**



ASICs vs GPUs

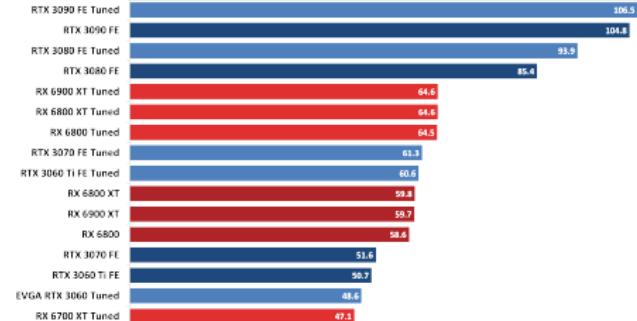
78 x GeForce RTX 30180 Mining Rig..



ASICs vs GPUs

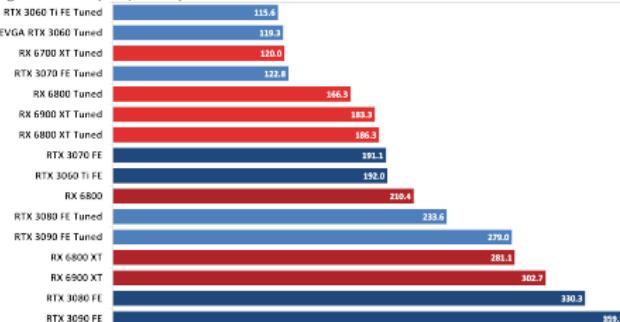
Coin-mining and Low-hash-rate GPUs

GPU Mining Performance
Mining Hash Rate (ETH, MH/s)



tom'sHARDWARE

GPU Mining Performance
Mining Power Use (ETH, Watts)



tom'sHARDWARE

