HW for NN

2017 Edition

Jan Korous

2017-03-03

TOC

NN implementation on contemporary HW

From maths to computation

Example task

Classify 256x256 pixel 32-bit RGB images of cats and dogs.

Abstract optimization problem

Let set of all input images be denoted $\mathbb{I} = \{0,1\}^{32\times256\times256}$

For training set
$\mathbb{L} \in \{\mathbb{I} \times \{0,1\}\}$,
set of allowed solutions
$\mathbb{S} \in \{\mathbb{I} \to \{0,1\}\}$,
and aggregated training error
$E : \mathbb{S} \times \mathbb{I} \to \mathbb{R}$

find $f \in \mathbb{S}$ minimizing the error metric: $f = arg\min_{g\in\mathbb{S}} E(g)$.

$+$ (assumptions about relation of training and inference, ...)

Arbitrarily chosen solution subset

E. g. network of L hidden softplus layers with sigmoid output layer:

$$\vec{g} = \sum_{n=0}^{1} \left\{ \vec{e_n} \left\{ z \mapsto \frac{1}{1+e^{-z}} \right\} \circ \left\{ \vec{x} \mapsto w_{L+1,n,0} + \sum_{i=1}^{n_L} w_{L+1,n,i} x_i \right\} \right\} \circ \bigcirc_{k=1}^{L} \sum_{n=1}^{n_k} \left\{ \vec{e_n} \left\{ z \mapsto ln(1+e^z) \right\} \circ \left\{ \vec{x} \mapsto w_{k,n,0} + \sum_{i=1}^{n_{k-1}} w_{k,n,i} x_i \right\} \right\}$$

biases $w_{k,n,0}$ and weights $w_{k,n,i} \in \mathbb{R}$
$\vec{e_n} \in$ canon. base $\mathbb{E}^*$

Arbitrarily chosen error metric

E. g. cross-entropy:

$$E(g, \mathbb{L}) = -\frac{1}{|\mathbb{L}|} \sum_{\{\vec{x},\vec{y}\}\in\mathbb{L}} \left\{ \sum_{i=0}^{1} y_i ln(g_i(x)) + (1 - y_i) ln(1 - g_i(x)) \right\}$$

Arbitrarily chosen error minimization algorithm

Usually first order iterative gradient based method optimizing $g(\vec{w}) \in \mathbb{S}$.

E. g. stochastic gradient descent with constant step t:
$\vec{w}_{n+1} = \vec{w}_n - t \cdot (\nabla_{\vec{w}} E(g, \mathbb{B}))(\vec{w}_n)$
where $\mathbb{B} \in \mathbb{L}$, step $t > 0 \in \mathbb{R}$

stopping criterion:
$\|\nabla_{\vec{w}} E\| < \varepsilon$

Solution

- solve on symbolic level down to simple algebraic operations
- forget all analytical worries
  (space of solutions properties, existence of solution, convergence conditions...)
- numerical methods

Implementation

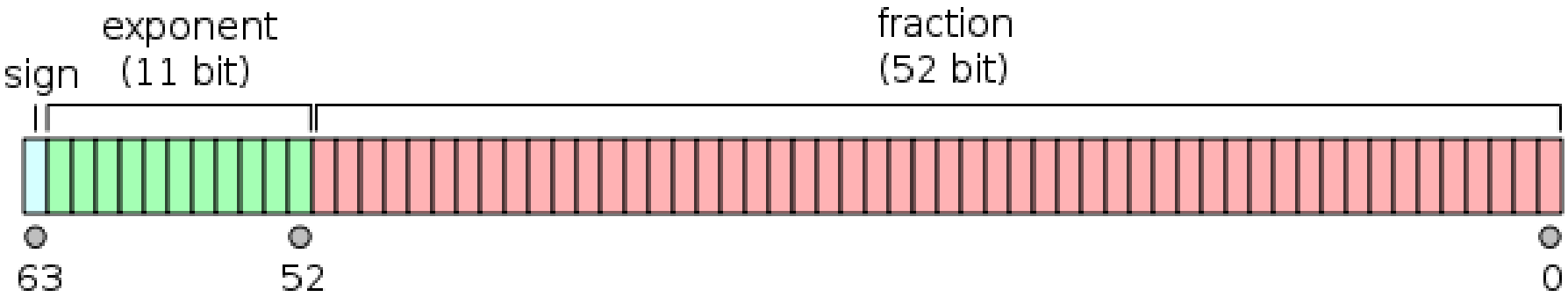
Software stack

- high-level framework (e. g. Keras)
- low level framework (e. g. TensorFlow, Theano)
- optimized linear algebra library (e. g. BLAS, CUDA - cuBLAS)
  - e. g. BLAS level 3 (1990): gemm (general matrix multiplication) function
- optimized hw instructions
  - e. g. fused multiplyadd (FMA) in GPU

Floating point arithmetics

IEEE 754 standard binary64 (AKA "double" AKA FP64)

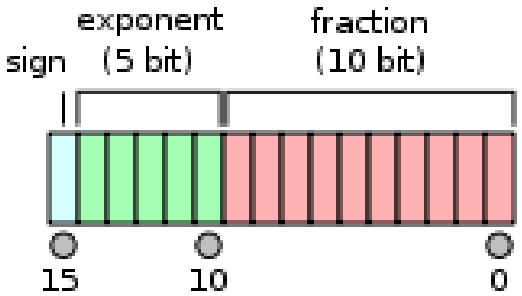
- Sign bit: 1 bit
- Exponent: 11 bits
- Significand precision: 53 bits (1.xyz)
- 15-17 significant decimal digits



Floating point arithmetics

IEEE 754 standard binary16 (AKA "half" AKA FP16)

- Sign bit: 1 bit
- Exponent: 5 bits
- Significand precision: 11 bits (1.xyz)
- "3.5" significant decimal digits



4x faster?

Not real numbers arithmetics

- limited precission
- rounding errors
- operations order dependency
- under/over-flows
- nothing like infinitesimal $\varepsilon > 0, \delta > 0$

NN precision sensitivity - intro

- numerical computations (3D physical rendering, simulations ...) usually use FP64 or FP32
- NN being succesfully trained on FP16 arithmetic
- NN being succesfully deployed on INT8 arithmetic
- robust algorithms? empirical evidence...

2016 overview

CPU or GPU?

Performance - benchmarks

Benchmarking State-of-the-Art Deep Learning Software Tools
Department of CS, Hong Kong Baptist University, 2017
https://arxiv.org/pdf/1608.07249.pdf

- Intel Core i7-3820, Intel Xeon E5-2630
- Nvidia GTX 980 (Maxwell), GTX 1080 (Pascal), Tesla K80 (Kepler)

conclusions

- "GPU is significantly faster." ($\approx$ 10x)
- No single best software tool fastest for all tasks.

Performance - benchmarks

Comparative Study of Deep Learning Software Frameworks
Robert Bosch LLC, 2016
`https://arxiv.org/pdf/1511.06435.pdf`

- Ubuntu 14.04
- Intel Xeon E5-1650 v2 3.50GHz 6Cores HT
- Nvidia GeForce GTX Titan X PCI-E

- modified LeNet, MNIST dataset
  GPU $\approx$ 10x faster
- AlexNet, ImageNet dataset
  GPU $\approx$ 30x faster
- Stacked autoencoders, MNIST dataset
  GPU $\approx$ 10x faster

Price

- Intel Core i7-6950X $1730
- Intel Core i7-6900K $1100
- NVidia Titan X (Pascal) $1200
- NVidia GTX 1080 - $600/700 (founders edition)

$\Rightarrow$ Both CPU and GPU price $\approx$ $1000.

Power consumption

- NVidia Titan X $\approx$ 250W
- NVidia GeForce GTX 1080 $\approx$ 180W
- Intel Core i7, Xeon $\approx$ 100-140W

GPU!

- price is similar
- GPU 10x faster (at least)
- GPU power consumption max 2x higher

...of course system still needs some CPU.

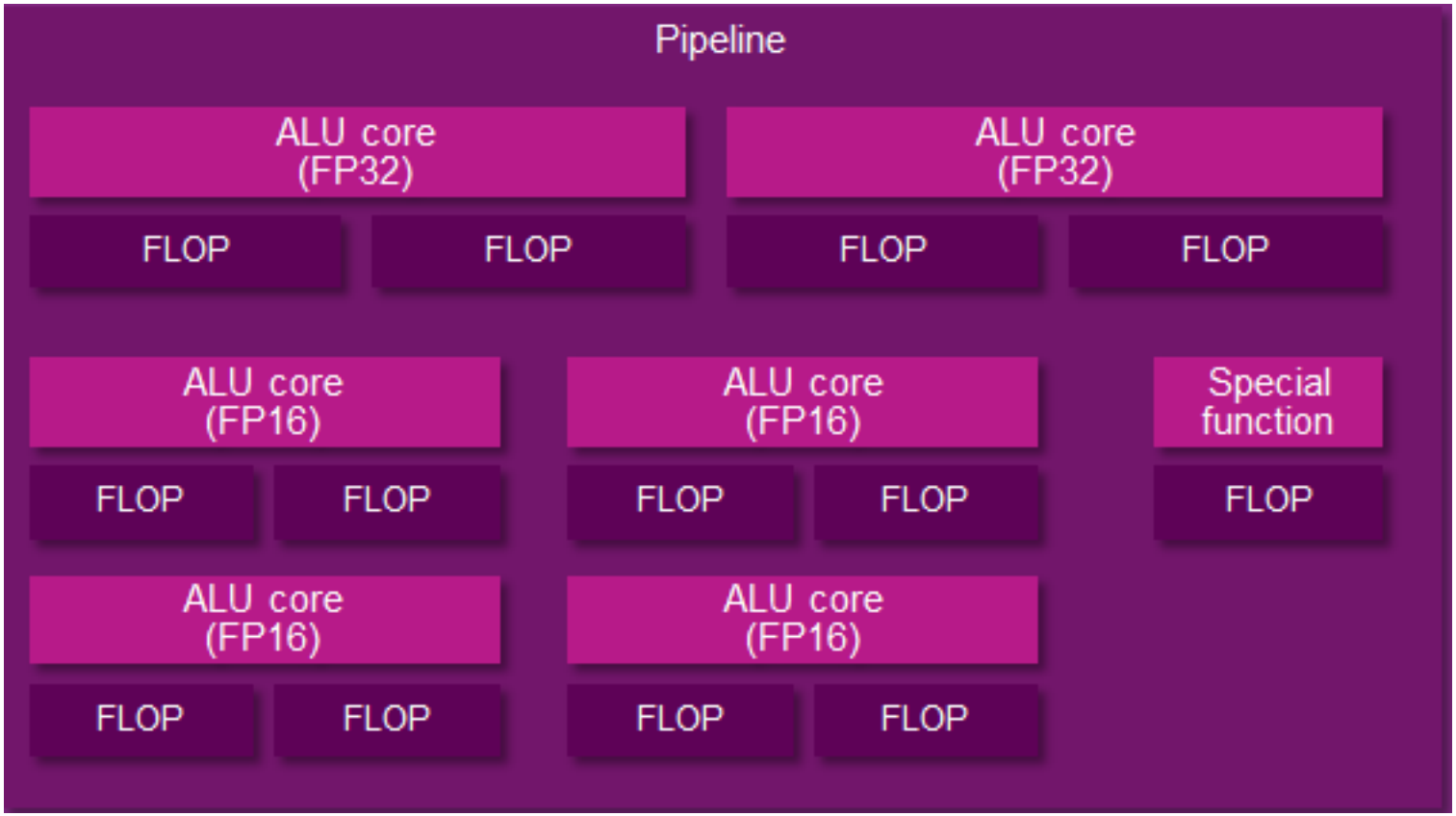
General GPU architecture

GeForce GTX 1080 Block Diagram

1 GPU $\approx$

- 10 streaming multi processor
- 10 x 100 computational core
- 10 x 100 x 10 specific ALU

(definitely NOT) CUDA Core Block Diagram



Different precision FP arithmetics implementation

- Different precision operations are potentially calculated by different ALUs.
- Different parts of chip (different transistors) are used for FP64 and FP32 operations.

GPU Parameters

- performance [TFLOPS] at optimal precission arithmetic [FP**]
- memory bandwidth [GB/s]
- memory size [GB]
- power consumption [W]
- GPU memory $\leftrightarrow$ CPU memory bandwidth [GB/s]
- ...

Crucial GPU Parameters?

- memory bandwidth [GB/s]
- arithmetic performance [TFLOPS] at FP16

For current GPUs bottleneck is memory bandwidth.

For decent utilization of ALUs performance magic is necessary.
Magic is system of memory caches similar to CPU.

Naive memory usage

GeForce GTX Titan Black (GK110 based)

- memory bandwidth ≈ 0.3TB/s
- FP32 performance ≈ 5 TFLOPS

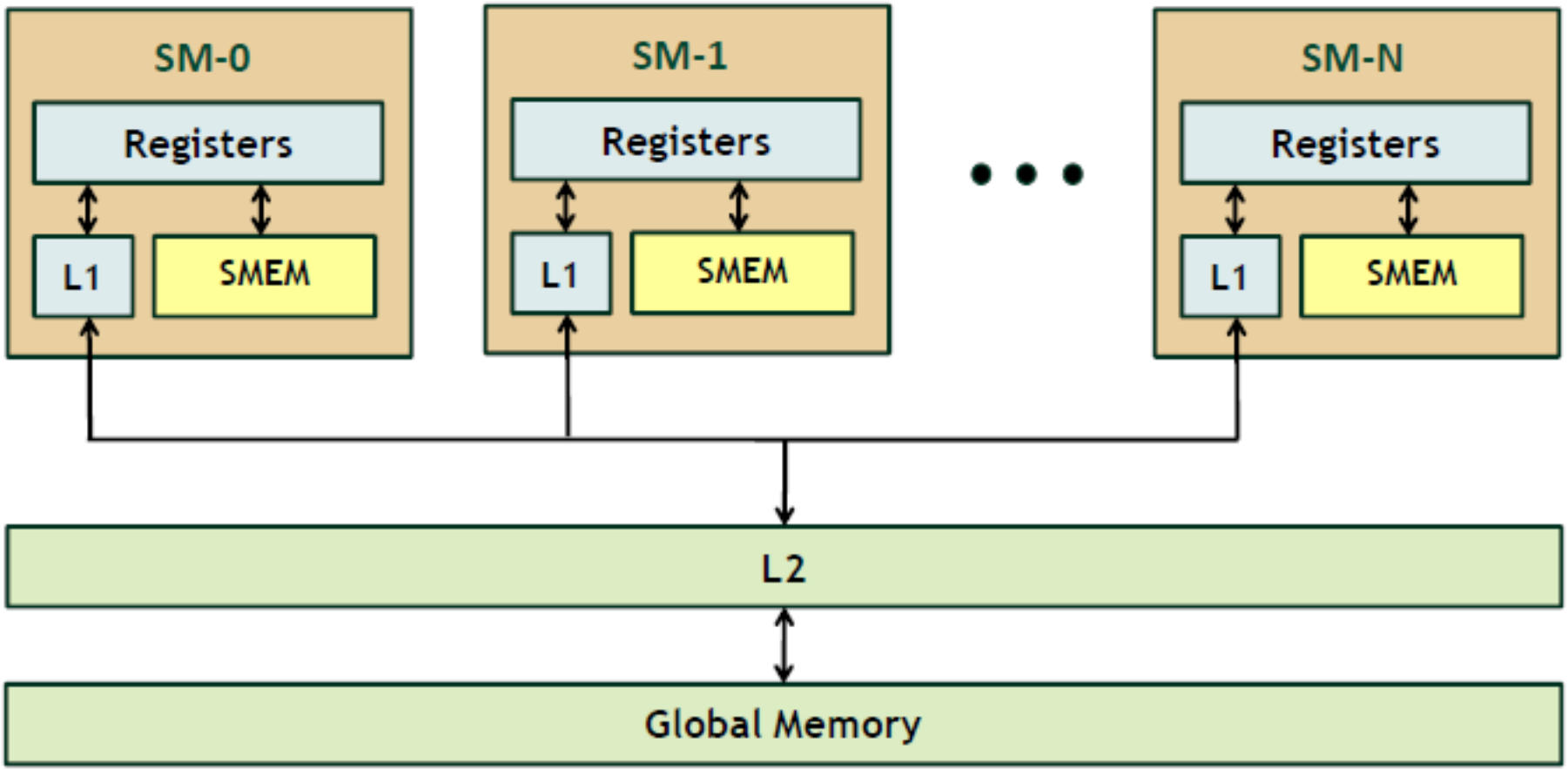
trivial FP32 $a + b$ with naive memory utilization

- necessary saturation input:
  $2 \times FP32 \times 5 \times 10^{12}/s = 40TB/s$
- necessary saturation output:
  $1 \times FP32 \times 5 \times 10^{12}/s = 20TB/s$

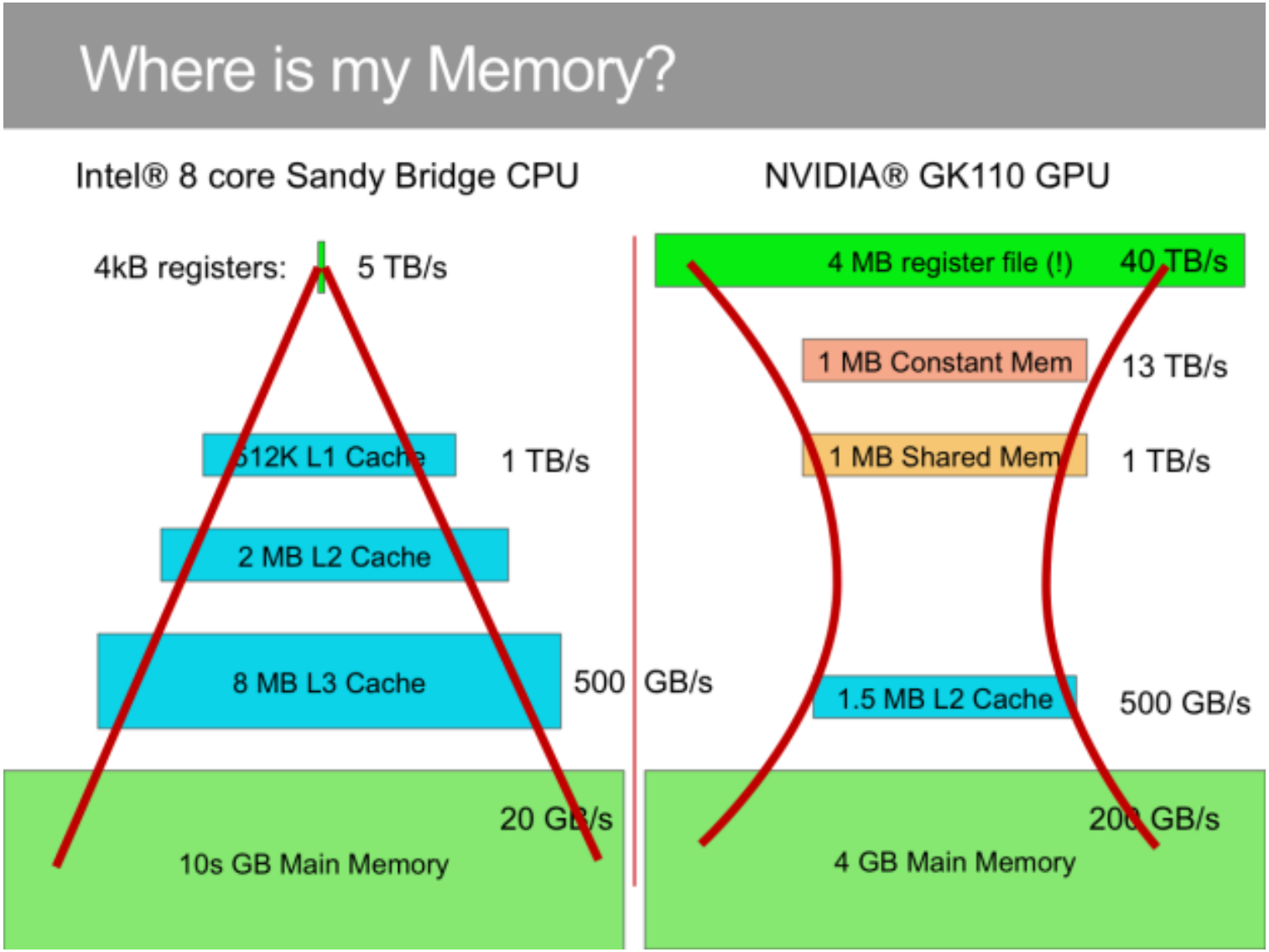
Operation with more input arguments will be even more limited by memory bandwidth.
If performance is plenty FP16 can really be 2x faster than FP32.

Real GPU memory hierarchy

NVidia GeForce GTX Titan (Fermi, GK110)



Real GPU memory hierarchy bandwidth



Magic

Source:
`http://courses.cms.caltech.edu/cs179/2015_lectures/cs179_2015_lec05.pdf`

- Compute throughput: 5 TFLOPS (FP16)
- Global memory bandwidth 336 GB/s (84 G-FP16/s)
- Shared memory bandwidth 3.4 TB/s (853 G-FP16/s)

"If you want to get beyond $\approx$ 900 GFLOPS, need to do multiple FLOPs per shared memory load."

"cuBLAS obtains about 4 TFLOPS on this GPU. Utilization is hard!"

NVidia beats AMD in ML market

Similar performance in hardware

- AMD Radeon Pro Duo
  16.4 TFLOPS @ FP32, HBM (512 GB/s), $1499, 350W
- NVidia GeForce GTX 1080
  8.2 TFLOPS @ FP32, GDDR5X (320 GB/s), $599, 180W

... ???

Low level APIs

- OpenCL
  - "Not to be confused with OpenGL"
  - "Framework for writing programs that execute across heterogeneous platforms consisting of CPUs, GPUs, DSPs, FPGAs and other processors or hardware accelerators."
  - developed, maintained by Khronos group (NVidia, AMD, Intel, Apple, Qualcomm, ... + 100)
- CUDA API
  - "parallel computing platform and programming model"
  - developed, maintained by NVidia

ML framework support

- OpenCL - Apache Singa, Wolfram Mathematica
- CUDA - Keras, TensorFlow, Theano, Torch, ...

- NVidia - both OpenCL and CUDA drivers
- AMD - OpenCL drivers only

Virtually no support for AMD GPUs by any relevant modern FOSS ML framework.

NVidia GPU options

Pascal architecture

Introduced 2016.

Two distinct chip categories:

- "pro" - GP100 chip - Tesla
- "consumer/gaming" - GP104 chip - GeForce

GP100 (Tesla)

CUDA core:

- memory bandwidth 540 (720) GB/s
- 64 x FP32 ALU
- 0 x FP16 ALU (!?)
  vec2 technology - identical operation for 2 inputs of 16bit on single FP32 ALU
  for suitable cases as fast as 2x FP16 ALUs

GP104 (GeForce)

CUDA core

- memory bandwidth 480 GB/s
- 128 x FP32 ALU
- 1 x vec2 FP32 ALU !!!
  FP16 performance only 2/128 FP32
- 2x performance in FP32 compared to GP100 constrained by memory bandwidth bottleneck

???

Tesla VS GeForce

18.7/21.2 TFLOPS @ FP16 (PCIe/NVLink)

- Tesla P100 PCI-E, 12GB (540GB/s) \$5,899 $\Rightarrow$ \$315/TFLOPS
- Tesla P100 PCI-E, 16GB (720GB/s) \$7,374 $\Rightarrow$ \$395/TFLOPS
- Tesla P100 SXM2, 16GB (720GB/s) \$9,428 $\Rightarrow$ \$445/TFLOPS

- GTX 1080: 9 TFLOPS @ FP32 (sic), 8GB (320 GB/s)
  \$599 $\Rightarrow$ \$70/TFLOPS (slow? small memory?)
- Titan X: 10.2 TFLOPS @ FP32 (sic), 12GB (480 GB/s)
  \$999 $\Rightarrow$ \$98/TFLOPS

???

Why Tesla?

- HBM2?
  - high memory bandwidth
  - 256 GB/s memory bandwidth per package, up to 8 GB per package
  - "slightly" faster than GDDR5X
- NVLink?
  - CPU-GPU, GPU-GPU interconnect
  - 20% theoretical speed-up to PCI-Express
  - TOP 500 super-computers (proprietary CPU...)
- Avoiding scaling overhead for absolute performance requirements?

2017 news

AMD ressurection

New CPU architecture "Zen"

- released March 2nd
- $\approx$ 50% instructions/clock increase
  (unique for CPU arch in last 5+ years)
- 1/2 - 2/3 price of comparable Intel CPUs
- "ML" branch prediction :-)

New GPU architecture "Vega"

- to be released Q2/2017
- implemented FP16, FP8 operations
- Vega effect: NVidia GTX 1080 \$599 $\rightarrow$ \$499
- GPU MI25
  - HBM2 memory
  - speculations about 25 TFLOPS @ FP16
  - (NVidia Titan X : 10.2 TFLOPS @ FP32)
  - $\Rightarrow$ 50TFLOPS @ FP8 ??? ...

Precission sensitivity of NN learning

- Deep Learning with Limited Numerical Precision (IBM, 2015)
  - "... empirical evaluation ... indicate that in most cases, 8-16 bits of precision is sufficient for back-propagation learning."
  - custom FPGA + stochastic rounding $\Rightarrow$ 12b fixed pt dec. num. are ok
  - alternatively start learning with 8bit, finish with 16bit float
  - https://arxiv.org/pdf/1502.02551.pdf
- Accelerating Deep Convolutional Networks using low-precision and sparsity (Intel, 2016)
  - custom FPGA - using CNN with 2bit weights
  - https://arxiv.org/pdf/1610.00324.pdf

AMD HIP

"HIP allows developers to convert CUDA code to portable C++."

"HIP is very thin and has little or no performance impact over coding directly in CUDA or hcc "HC" mode."

"The "hipify" tool automatically converts source from CUDA to HIP."

— https://github.com/GPUOpen-ProfessionalCompute-Tools/HIP

Radeon Open Compute Platform

"With the release of Radeon Instinct, ROCm can accelerate common deep-learning frameworks like Caffe, Torch 7, and TensorFlow on AMD hardware."

— techreport.com

TensorFlow XLA

- "domain-specific compiler for linear algebra that optimizes TensorFlow computations"
- using LLVM backend interface
- https://www.tensorflow.org/versions/master/experimental/xla/

TensorFlow XLA + ROCm $\stackrel{?}{=}$ NN @ AMD GPU

"User Guide for AMDGPU Back-end"

— http://llvm.org/docs/AMDGPUUsage.html

"Developing a new backend for XLA

...

Scenario 2: Non-CPU-like hardware with an existing LLVM backend"

— https://www.tensorflow.org

NVidia strikes back?

NVidia 1080 Ti

- announced Feb28, price $699, GP102 chip
- effectively obsoletes Titan X (only 1GB RAM benefit at $500 difference)
- FP32 performance $\approx$ 11 TFLOPS
- same design as GTX 1080 (GP104)
  performance @ FP16 is 1/64 performance @ FP32
- GDDR5X memory - bandwidth 484 GB/s

NVidia Volta

TBA

- new NVidia GPU architecture
- rumored to be released 2017-2018