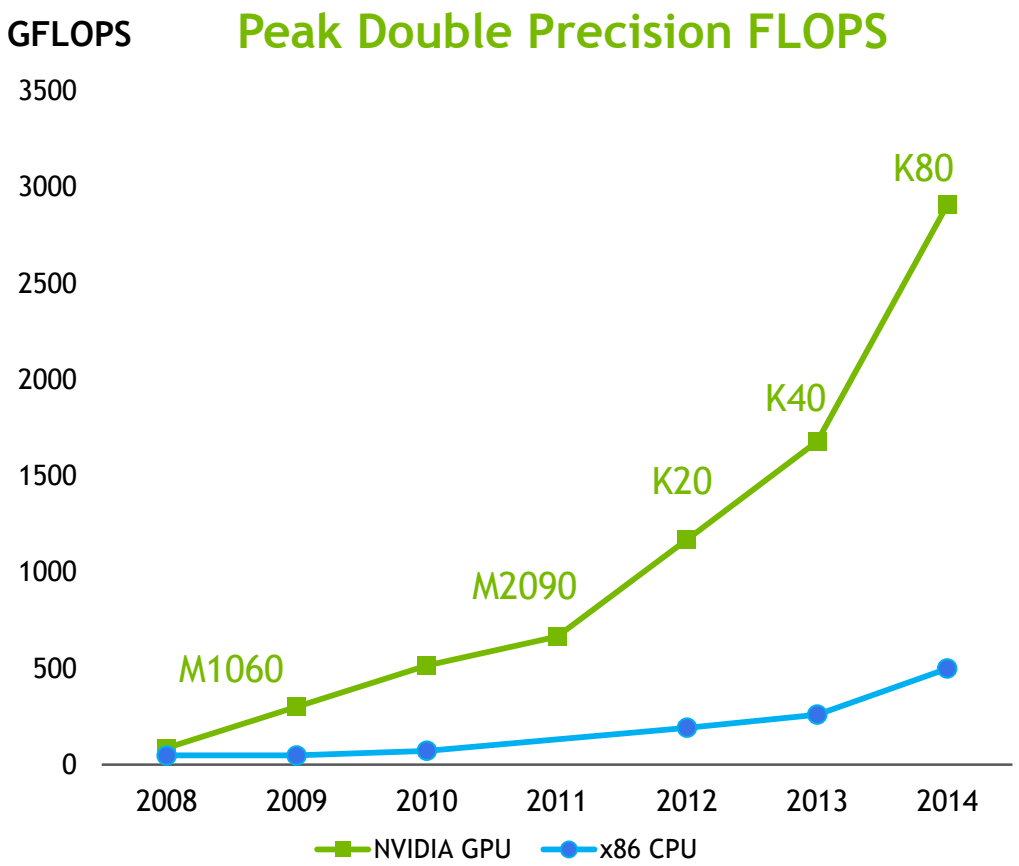
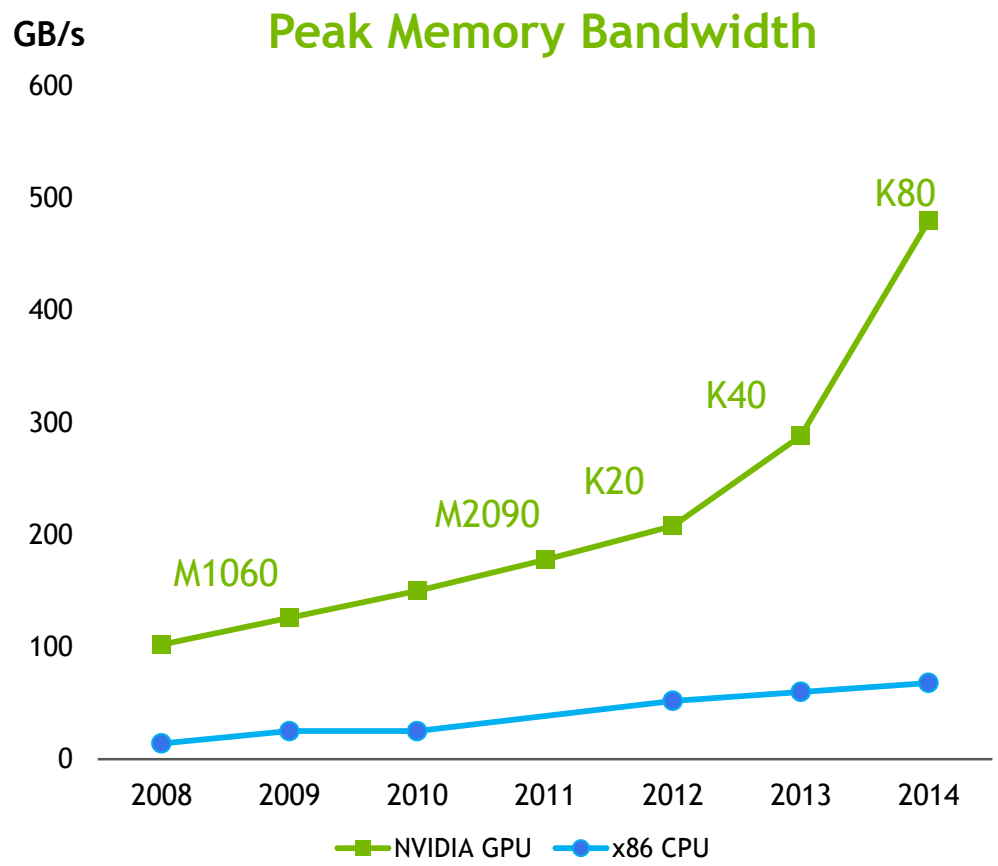
DL INFRA WITH NVIDIA GPU

Hyungon Ryu

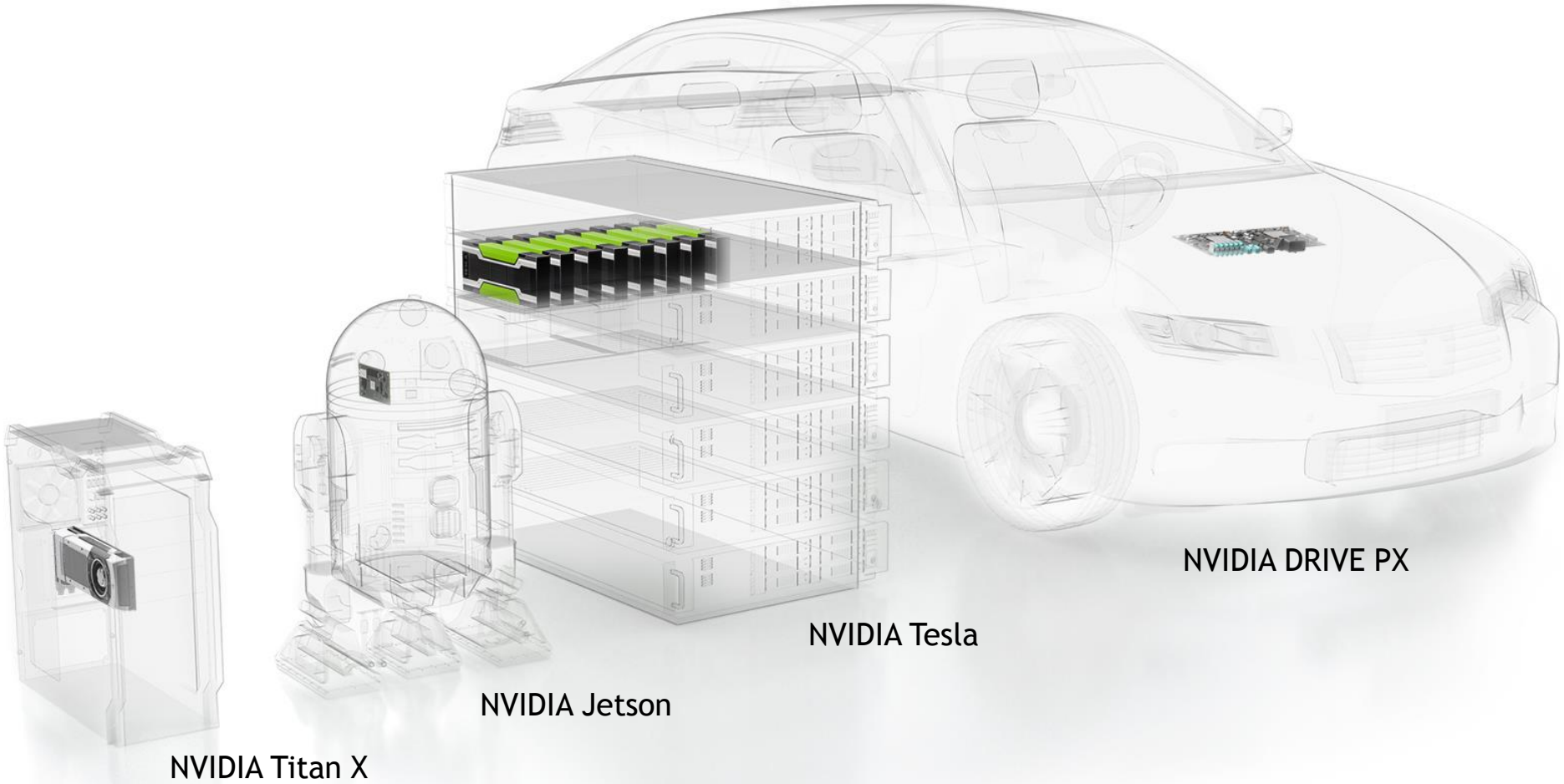
NVIDIA Sr. Solution Architect

PERFORMANCE LEAD CONTINUES TO GROW





DEEP LEARNING EVERYWHERE

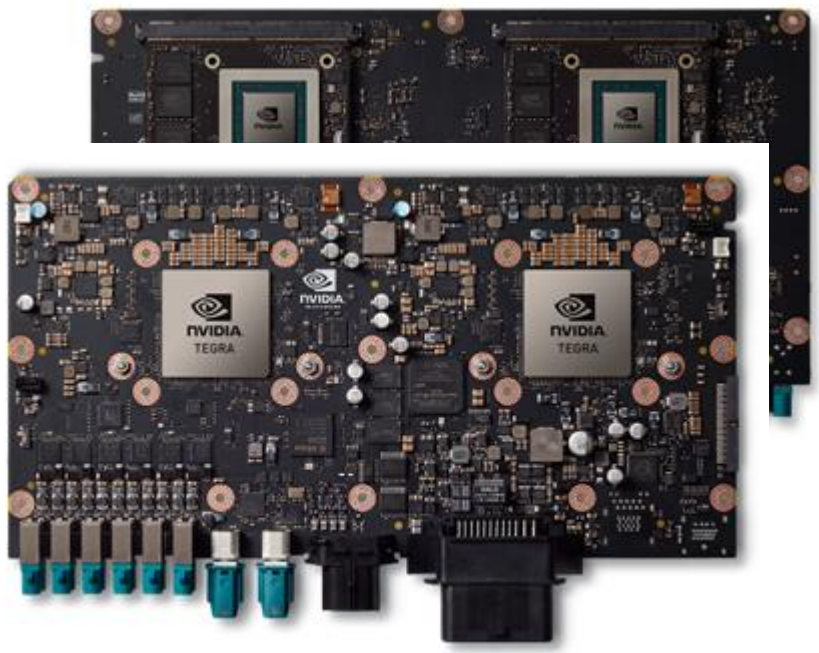


DRIVE PX 2

16 CPU cores | Pascal GPUs | 8TFLOPS

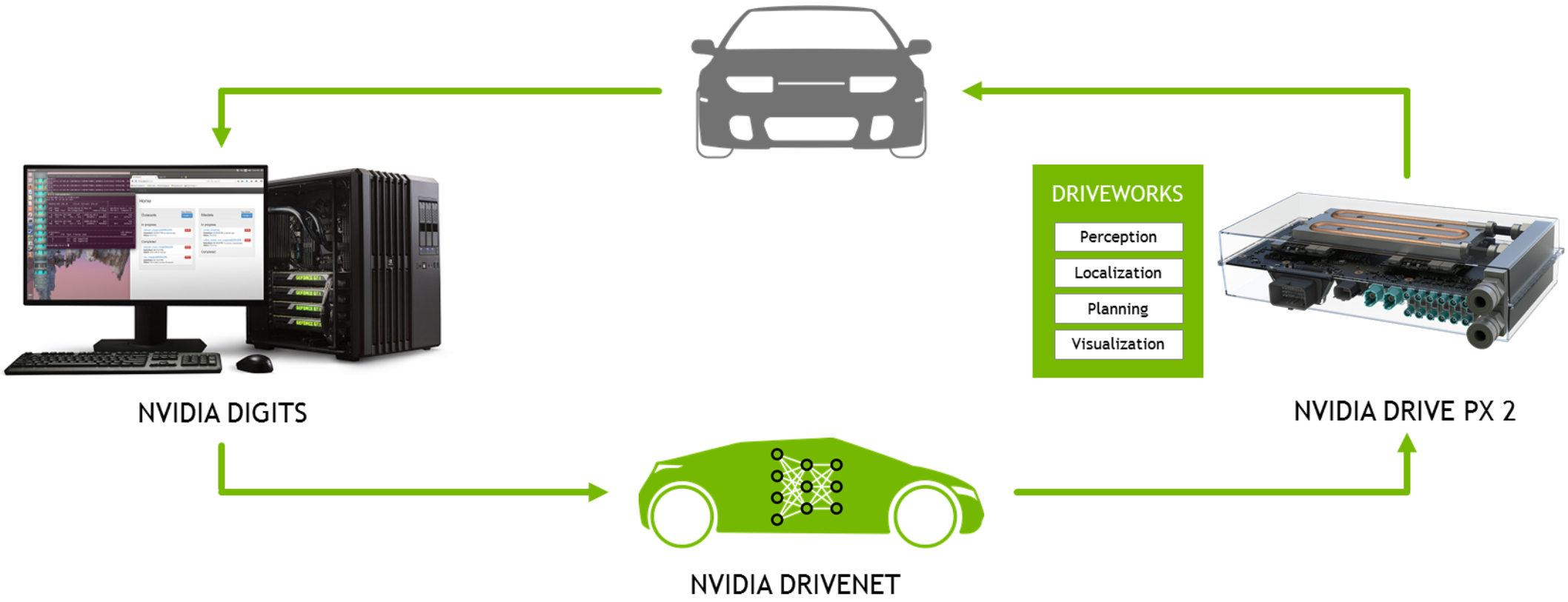
dGPU

CPU



http://blogs.nvidia.com/blog/2016/01/04/automotive-nvidia-drive-px-2/

END-TO-END DEEP LEARNING PLATFORM FOR SELF-DRIVING CARS



JETSON TX1

Quadro CPU cores | Maxwell GPU 256 cores | 4GB LPDDR4

5MP camera | 802.11ac Wi-Fi | 16GB eMMC 5.1 Flash Storage | USB 3.0 | HDMI

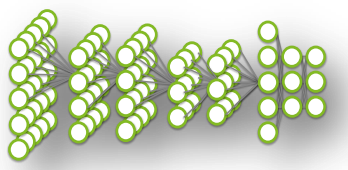
http://www.nvidia.com/object/jetson-tx1-dev-kit.html

END-TO-END DEEP LEARNING PLATFORM FOR EMBEDED

Jetson TX1 module

Training with Tesla M40

Jetson TX1 devkit



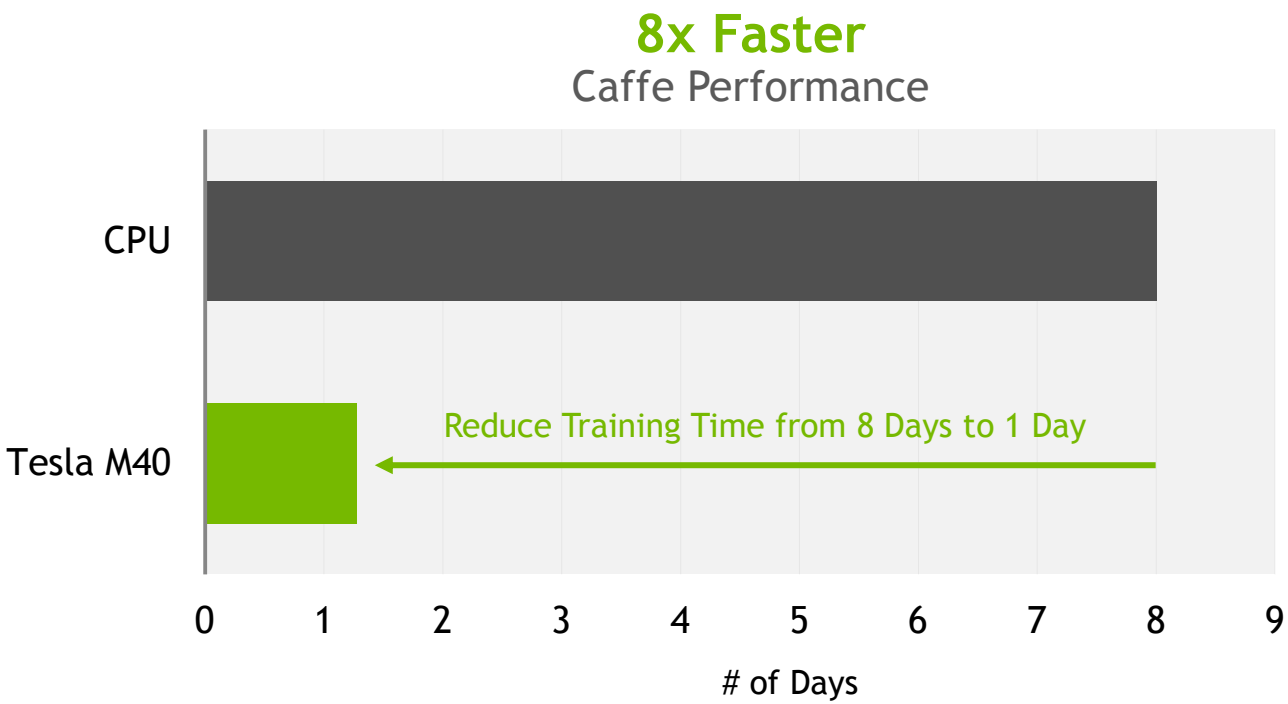
Trained parameter

TESLA PRODUCTS FOR TARGET MARKET

Market Segment	Key Feature	Recommendation
Supercomputing	• Double Precision Perf.	Tesla K80, K40
Deep Learning Training	• 12GB Memory per GPU • Single Precision Perf. per GPU • *Enable FP16*	Tesla M40
Accelerated Virtual Desktop	• # of Concurrent Connected Users • *GRID SDK for VDI* • *Support Hypervisor*	Tesla M60, M6
Hyperscale	• NVENC/NVDEC • Low power, Single Precision • Small form factor (LP type)	Tesla M4

TESLA M40

WORLD'S FASTEST ACCELERATOR FOR DEEP LEARNING



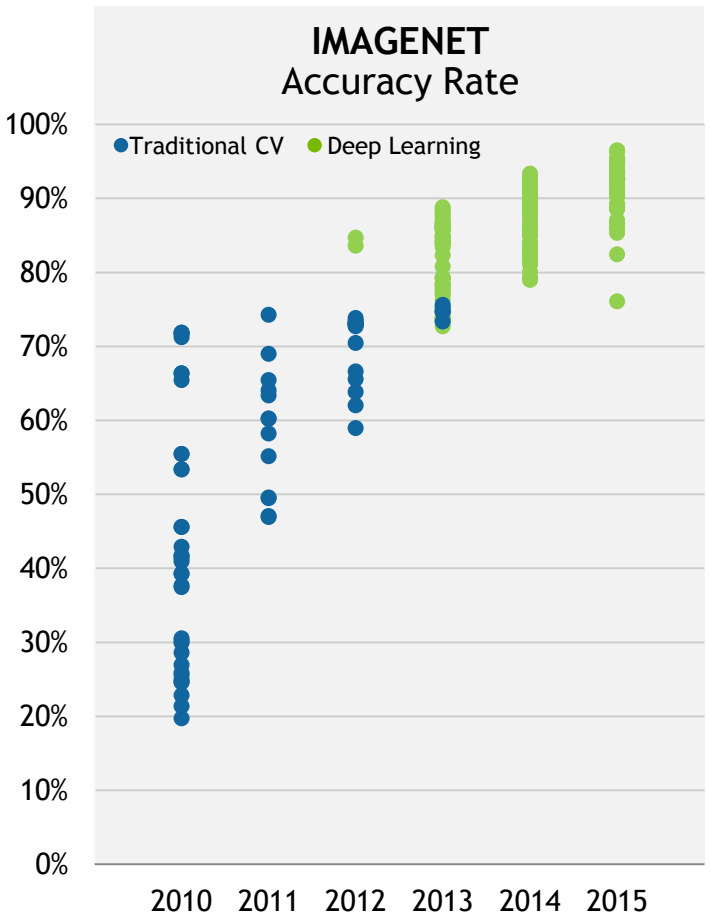
CUDA Cores	3072
Peak SP	7 TFLOPS
GDDR5 Memory	12 GB
Bandwidth	288 GB/s
Power	250W

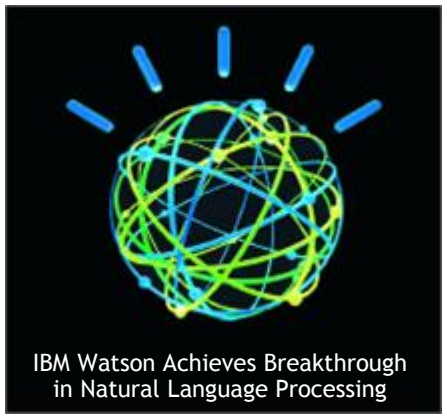
Caffe Benchmark: AlexNet training throughput based on 20 iterations, CPU: E5-2697v2 @ 2.70GHz. 64GB System Memory, CentOS 6.2

PRODUCT

구분	Tesla K40	Tesla K80	Tesla M4	Tesla M6	Tesla M40	Tesla M60
Architecture	Kepler	Kepler	Maxwell	Maxwell	Maxwell	Maxwell
# of GPU	1	2	1	1	1	2
# of Cores per GPU	2880	2496	1024	1536	3072	2048
# of Cores Per Board	2880	4992	1024	1536	3072	4096
SP Perf. per GPU	5.04TF	4.3TF	2.2TF	3.3TF	7.0TF	4.82TF
SP Perf. per Board	5.04TF	8.7TF	2.2TF	3.3TF	7.0TF	9.65TF
Mem per GPU	12GB	12GB	4GB	8GB	12GB	8GB
Mem per Board	12GB	24GB	4GB	8GB	12GB	16GB
Form Factor	Dual Slot	Dual Slot	LP Single Slot	Blade MXM	Dual Slot	Dual Slot
소비 전력	235W	300W	50/75W	100W	250W	300W
냉각	액티브/패시브	패시브	패시브	OEM	패시브	패시브

THE AI RACE IS ON





IBM Watson Achieves Breakthrough in Natural Language Processing

Facebook Launches Big Sur

Baidu Deep Speech 2 Beats Humans

TensorFlow

Google Launches TensorFlow

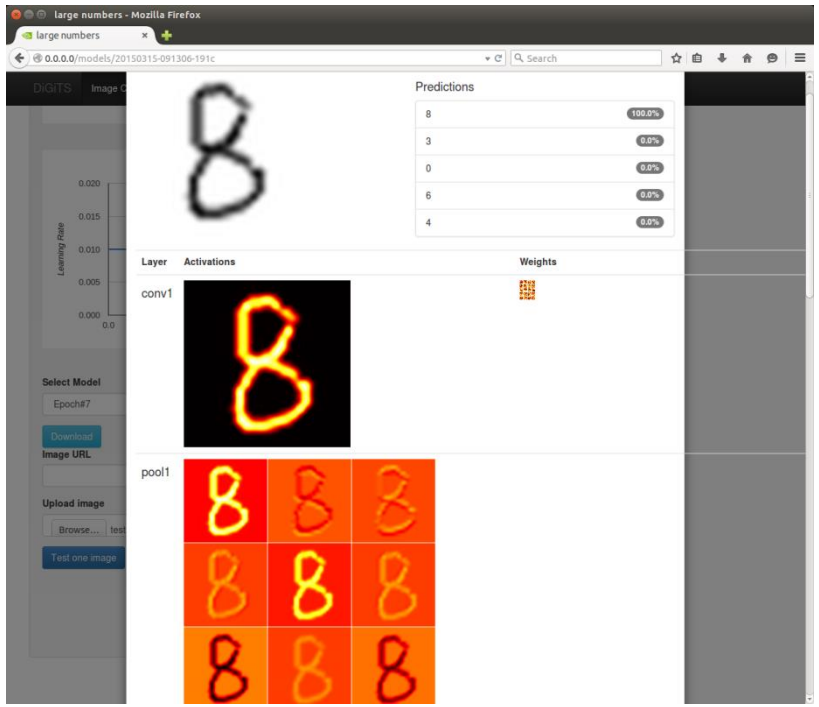
Toyota Invests $1B in AI Labs

Microsoft

Microsoft & U. Science & Tech, China Beat Humans on IQ

DIGITS

Deep Learning GPU Training System

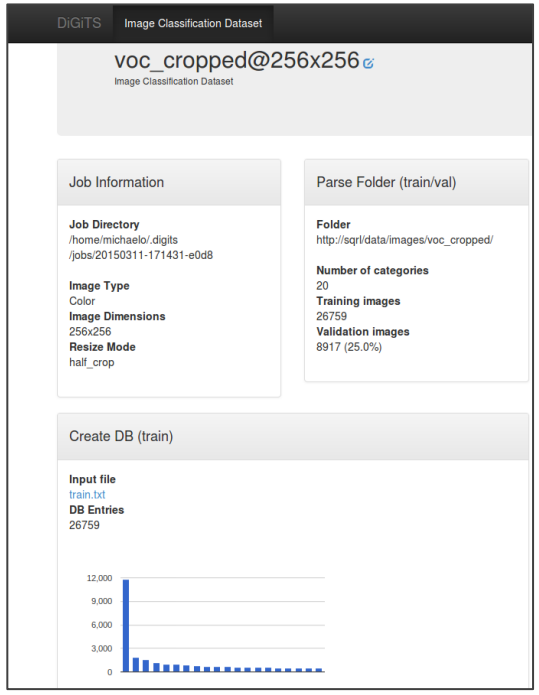


Who it is for

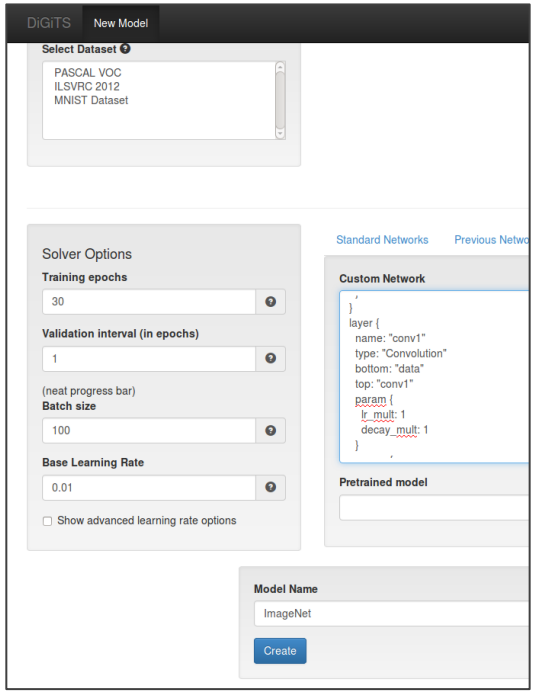
- Deep learning researchers
- Automotive
- Medical Researchers
- Finance
- Intelligent Video Analytics
- Web Companies
- Startups

DIGITS

Process Data

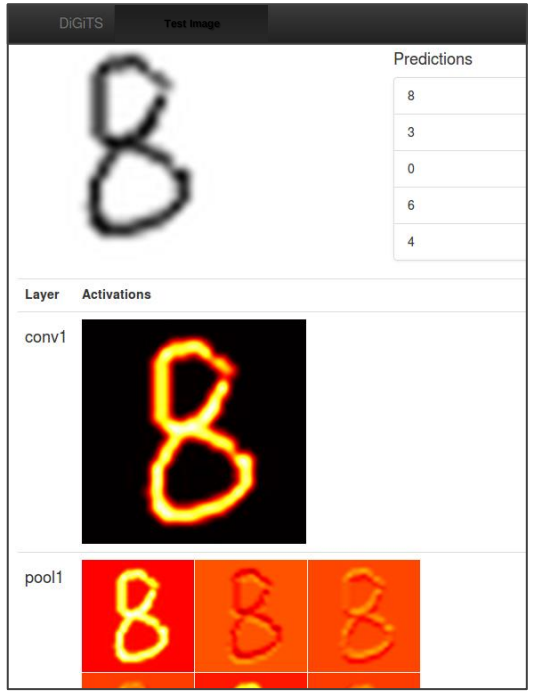


Configure DNN



Monitor Progress

Visualize Layers



DIABETIC RETINOPATHY

Affects 347 million people worldwide

- Leading cause of blindness among working age population in developed world
- Changes to blood vessels in the retina lead to aneurisms and fluid leaks

  If no treated early, can cause blindness

  Requires regular screenings
- Fundus photography with interpretation by trained physician

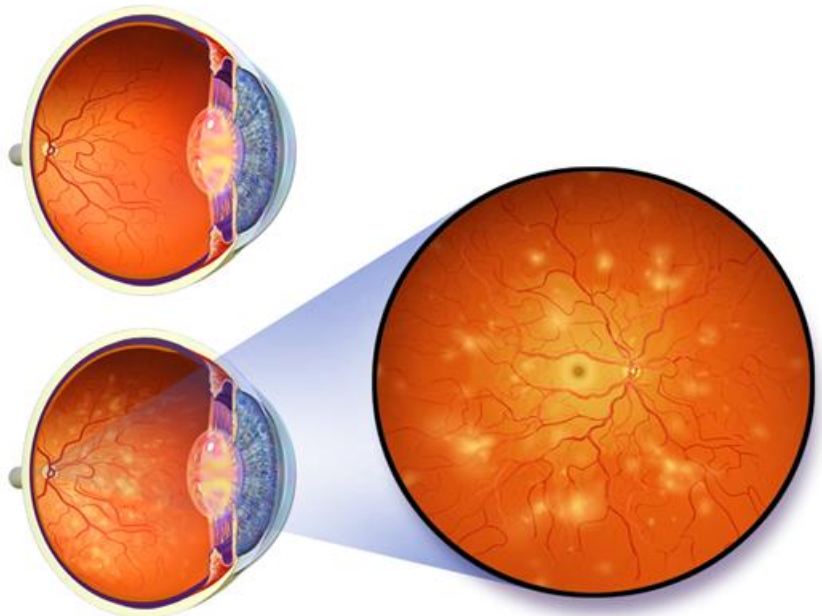


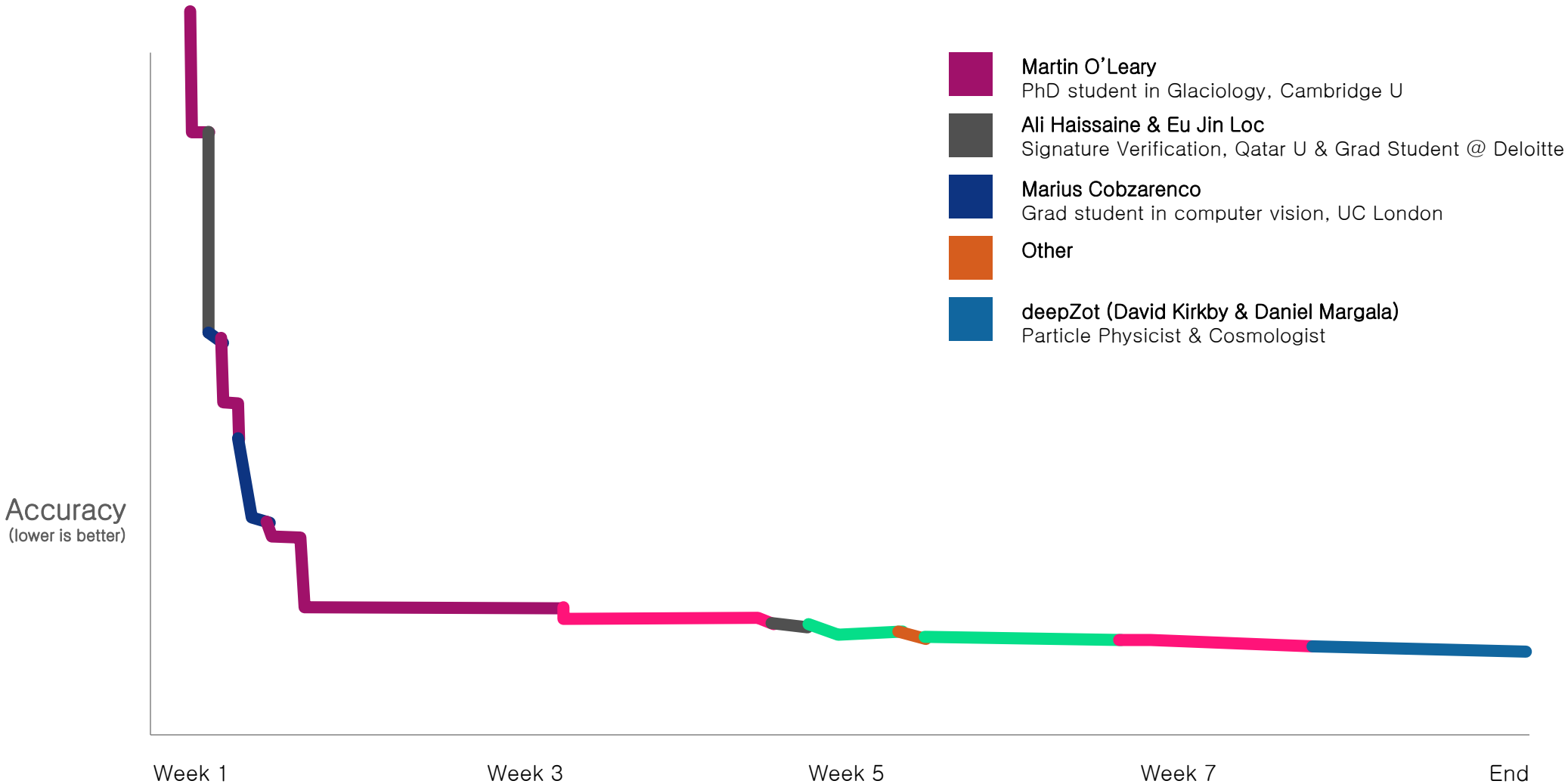
Image credit: Blausen.com staff. "Blausen gallery 2014". *Wikiversity Journal of Medicine*. DOI:10.15347/wjm/2014.010. ISSN 20018762.

KAGGLE DIABETIC RETINOPATHY CONTEST

$100,000 award sponsored by the California Healthcare Foundation

- Contestants provided 17,000 left/right images with score: 0 (healthy) to 4 (diseased)
- Typical clinician scores 0.83 (1.0 = perfect agreement with another clinician)
- 661 teams entered
- Winning score 0.84958
- 4 teams above 0.83

Strength of Community-based Data Science – “Mapping Dark Matter” results



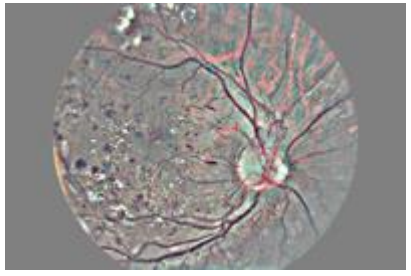
BENJAMIN GRAHAM – FINISHED #1!

Assistant Professor in Stats and Complexity, University of Warwick

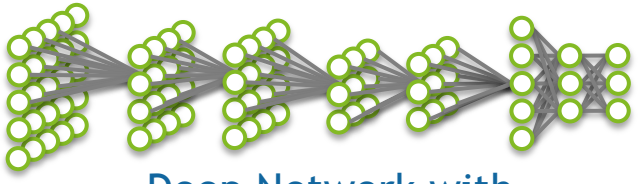
SparseConvNet
(written in CUDA)

NVIDIA CUDA

NVIDIA GPU



SparseConvNet



Deep Network with
fractional pooling

Random Forest
(scikits-learn)

ANTREAS ANTONIOU – FINISHED IN TOP 3RD

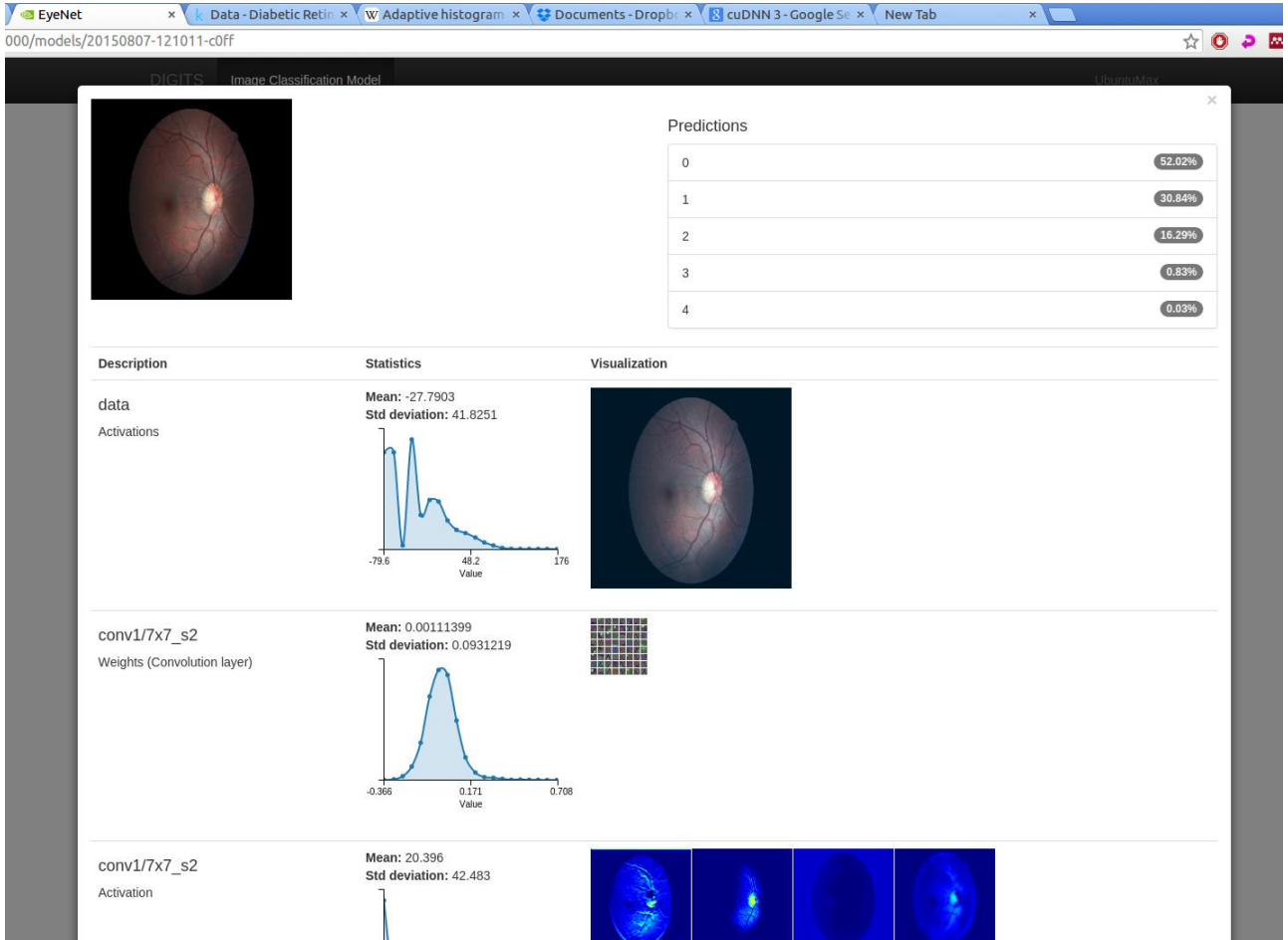
Master's Data Science student, University of Lancaster

NVIDIA DIGITS

NVIDIA cuDNN

NVIDIA CUDA

NVIDIA GPU



GPU Perf.

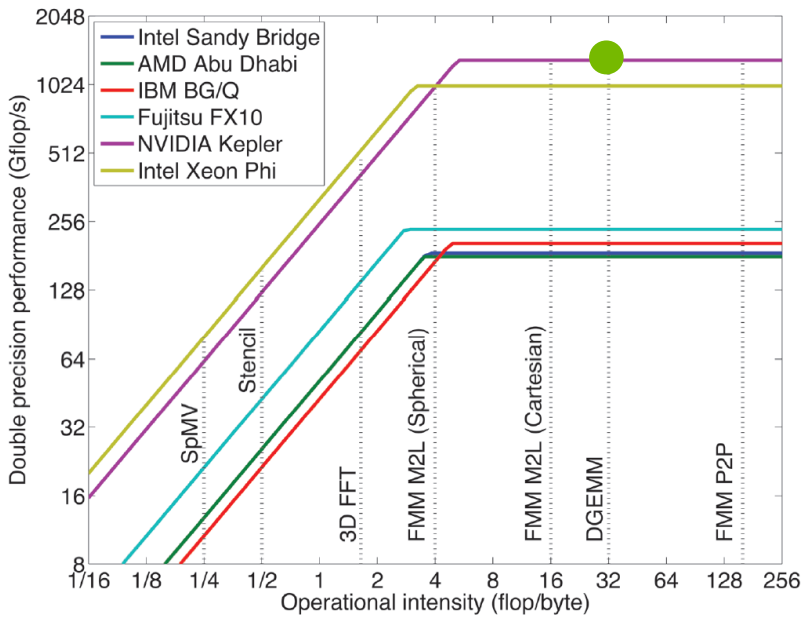
PERFORMANCE

	Clock (Ghz)	Core #	Socket #	total core	Inst #	Vec #	GFLOPS
CPU1	2.2	12	2	24	2	4	422.4
CPU2	3.6	8	2	16	2	4	460.8
CPU3	3.6	8	2	16	2	8	921.6
CPU4	2.2	12	2	24	2	8	844.8

Arch	Product	Clock (Mhz)	Core /SM	SM /chip	total core	GPU	FMA	GFLOPS (SP)	ratio	GFLOPS (DP)
mawell	M40	1,140	128	24	3,072	1	2	7,004	32	219
mawell	M60	1,180	128	16	2,048	2	2	9,667	32	302
kepler	K40	875	192	15	2,880	1	2	5,040	4	1,260
kepler	K80	875	192	14	2,688	2	2	9,408	4	2,352
fermi	C2075	1,150	16	30	480	1	2	1,104	2	552

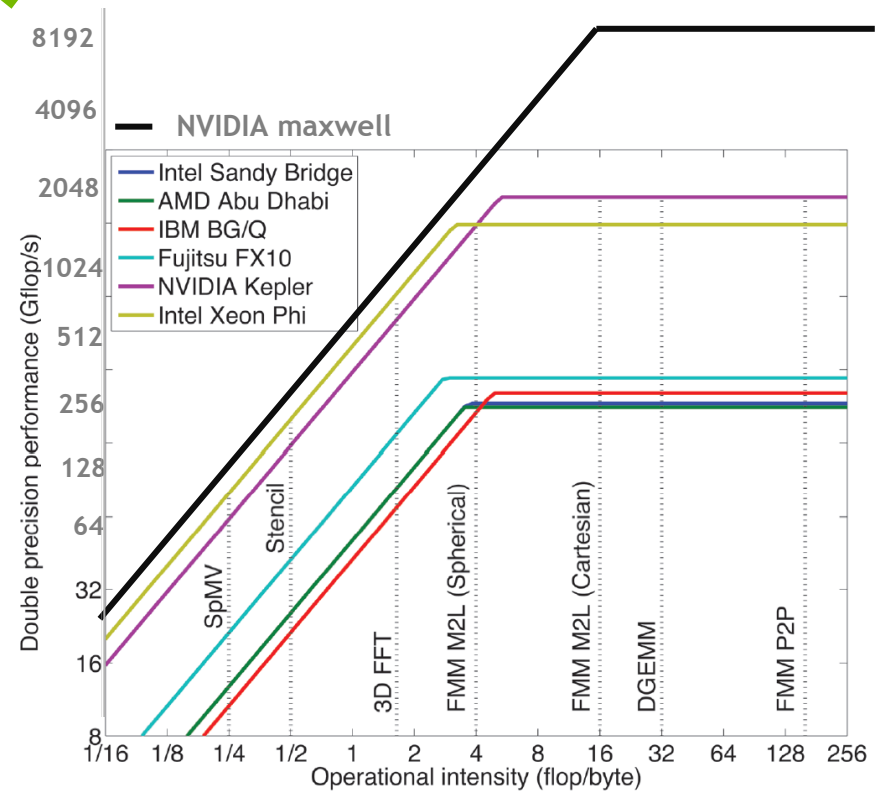
LOOPLINE

Double Precision



modify

Single Precision



THE ENGINE OF MODERN AI

EDUCATION

TORCH	CAFFE
NYU, facebook	Berkeley UNIVERSITY OF CALIFORNIA
THEANO	**MATCONVNET**
Université de Montréal	UNIVERSITY OF OXFORD
MOCHA.JL	**PURINE**
MIT Massachusetts Institute of Technology	NUS National University of Singapore
MINERVA	**MXNET***
NYU, Microsoft	UNIVERSITY of WASHINGTON, Carnegie Mellon University

BIG SUR	TENSORFLOW	WATSON	CNTK
facebook	Google	IBM	Microsoft

START-UPS

CHAINER	DL4J	KERAS	OPENDEEP
Preferred Networks	SKYMIND	SCHULTS LABORATORIES	VITRUVIAN

NVIDIA GPU PLATFORM

*U. Washington, CMU, Stanford, TuSimple, NYU, Microsoft, U. Alberta, MIT, NYU Shanghai

GPU-ACCELERATED DEEP LEARNING FRAMEWORKS

	CAFFE	TORCH	THEANO	CUDA-CONVNET2	KALDI
Description	Deep Learning Framework	Scientific Computing Framework	Math Expression Compiler	Deep Learning Application	Speech Recognition Toolkit
cuDNN	R2	R2	R2	--	--
Multi-GPU	✓	In Progress	In Progress	✓	✓ (nnet2)
Multi-CPU	✗	✗	✗	✗	✓ (nnet2)
License	BSD-2	GPL	BSD	Apache 2.0	Apache 2.0
Interface(s)	Text-based definition files, C++. Python, MATLAB	Python, Lua, MATLAB	Python	C++	C++, Shell scripts
Embedded (TK1)	✓	✓	✗	✗	✗

http://developer.nvidia.com/deeplearning

DL OPENSOURCE

framework	URLs
NV digits	Nvidia developer site github nvidia source site
Caffe	https://github.com/BVLC/caffe (official) https://github.com/NVIDIA/caffe (nvidia tree)
Torch7	http://deeplearning.net/software/theano/
Threano	http://deeplearning.net/software/theano/
KALDI	http://kaldi.sourceforge.net/
cuda-convnet	https://code.google.com/archive/p/cuda-convnet2/
nervana neon	https://github.com/NervanaSystems/neon
MXNET	https://github.com/dmlc/mxnet
MATCONVNET	http://www.vlfeat.org/matconvnet/
Mocha.JL	https://github.com/pluskid/Mocha.jl
purine	https://github.com/purine/purine2
Minerva	https://github.com/dmlc/minerva
Google tensorflow	https://www.tensorflow.org/
MS CNTK	https://cntk.codeplex.com/
MS DLTK	http://www.dmtk.io/
overfeat	http://cilvr.nyu.edu/doku.php?id=code:start
neuraltalk2	https://github.com/karpathy/neuraltalk2

DATASET & MEMORY FOR DL

IMAGENET DATASET

- Imagenet 2012 CLS-LOC dataset

type	size	# of image
Training Image	: 138GB	1,281,167
Validation Image	: 6.3GB	50,000
Teset Images	: 13GB	100,000

- Imagenet 2014 DET dataset

type	size	# of image
TrainingImage	: 47GB	456,567
Validataion Image	: 2.6GB	20,121
Test Image	: 5.1 GB	40,152

MSCOCO DATASET

- 2014 Training images [80K/13GB]
  2014 Validation images [40K/6.2GB]
  2014 Testing images [40K/6.2GB]
  2015 Testing images [80K/12.4G]

The man at bat readies to swing at the pitch while the umpire looks on.

A large bus sitting next to a very tall building.

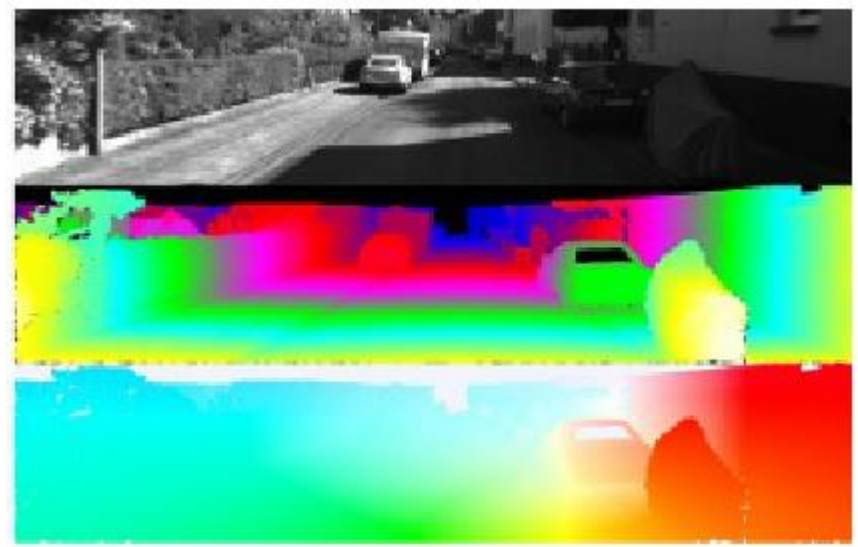
KITTI DATASET

- stereo, optical flow, visual odometry, 3D object detection and 3D tracking

- stereo/optical flow data set (2 GB)
- multi-view extension (17 GB)

- left color images of object data set (12 GB)
- right (stereo) (12 GB)

- point clouds(laser) (29 GB)



SELF-DRIVING CAR DATASET

George Hotz Comma.AI

- Hotz had accumulated some 100 hours of driving data to train his algorithms.
- An hour of driving generates roughly 300 gigabytes worth of data*.
- On his desk, Hotz has amassed 50 terabytes worth of hard drives packed with driving data.

*multiple camera + LIDAR + GPS + CAN data

Training

GPUMEM

Step4. GPUMem

cudaMemcpy

SYSMEM

Step3. lmDB

Memory DB for easy feeding

File

Step2. levelDB (optional)

Step1. PNG files

* lmdb & levelDB is not SQL based DBMS
  Key/Value-based DB (use DB as memptr)

Training

GPUMEM

Step4. GPUMem

cudaMemcpy

Step5. trained Parameters

cudaMemcpy

SYSMEM

Step3. lmDB

Memory DB for easy feeding

Step6. snapshot

Step7. snapshot(save)

File

Step2. levelDB (optional)

Step1. PNG files

* lmdb & levelDB is not SQL based DBMS
  Key/Value-based DB (use DB as memptr)

Inference

GPUMEM

Step6. GPUMem

cudaMemcpy

Step3. trained Parameters

cudaMemcpy

SYSMEM

Step5. lmDB

Memory DB for easy feeding

Step2. snapshot

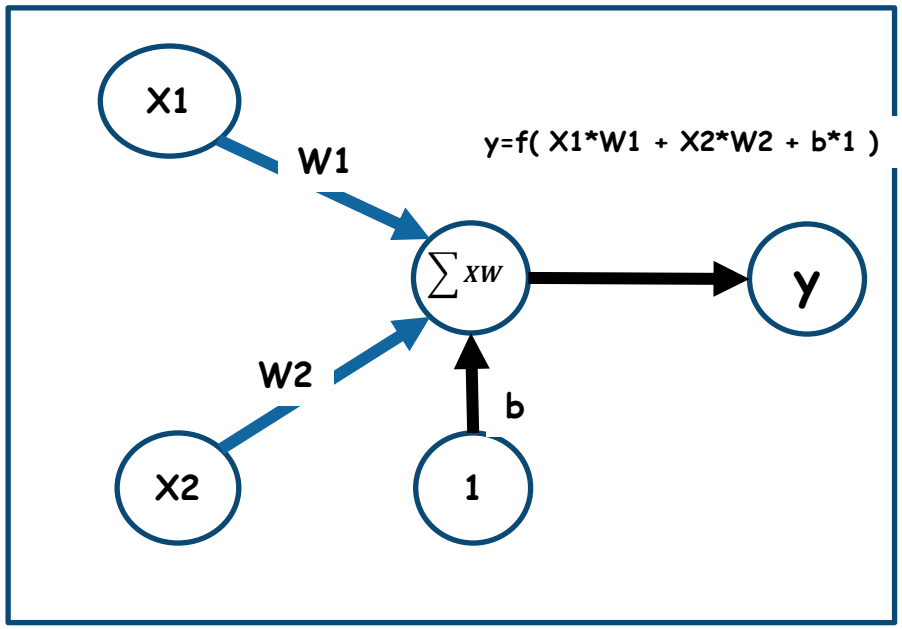
File/
Internet

Step4. PNG files

Step1. snapshot(saved)

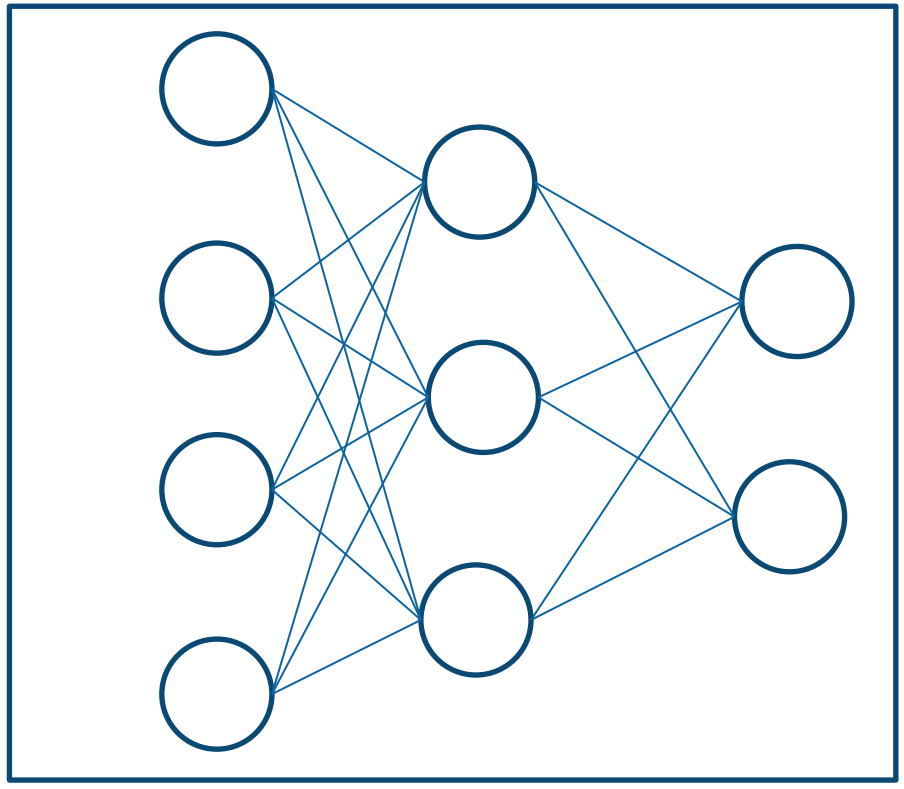
* lmdb & levelDB is not SQL based DBMS
  Key/Value-based DB (use DB as memptr)

PARAMETERS FOR ANN (PERCEPTRON)



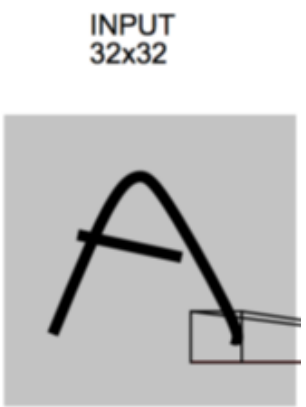
- Input : 2 (X1, X2)
- Output : 1 (y)
- Param : 3 (W1, W2,, b)
  - 2+1
- Connection : 2 = 2*1

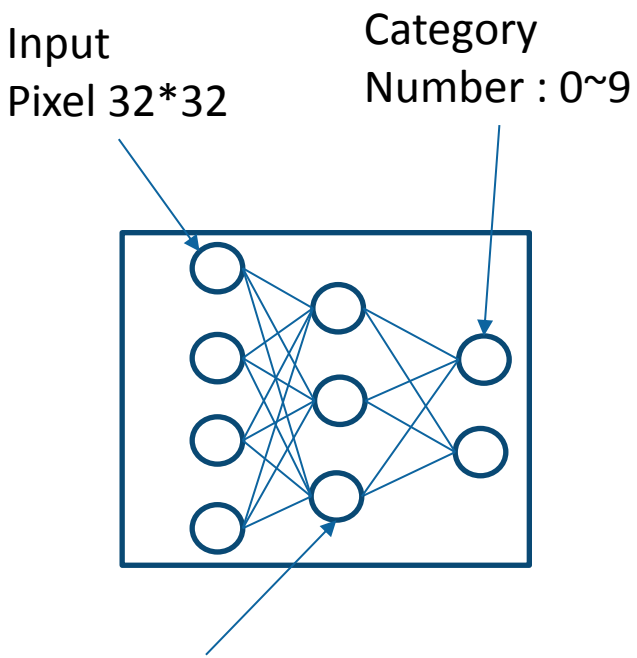
PARAM FOR ANN (MLP)



- Input : 4 (L11, L12,L13, L14)
- Hidden : 3 (L21, L22, L23)
- Output : 2 (L31, L32)
- Param : 23=15+8
  - L1 to L2 : (4+1) * 3=15
  - L2 to L3 : (3+1) * 2= 8
- Connection : 18 = 12 + 6
  - L1 to L2 : 4*3 = 12
  - L2 to L3 : 3*2= 6

MNIST PARAM WITH LEGACY ANN





- Input : 1024 = 32*32 pixel
- Hidden : 84 = (7*12)
- Output : 10 (0~9)
- Param : 86,960
  - (1024+1)*84 = 86,100
  - (84+1)*10 = 860
- Connection : 86,856
  - 1024*84=86,016
  - 84*10=840

0.3MB param
11MB dataset

70K images

CONVOLUTION

3	1	2
3	0	1
2	0	1

*

-1	0	-1
2	1	2
-1	0	-1

-3	0	-2
6	0	2
-2	0	1

+ →

2

```
FOR H
    FOR W
        FOR K
        SUM[H][W]=+Input[H-k][W-k]*K[K]
```

IMPLEMENTATION 1: 2D CONV AS A GEMV

Image

I1	I2	I3	I4	I5	I6
I7	I8	I9	I10	I11	I12
I13	I14	I15	I16	I17	I18
I19	I20	I21	I22	I23	I24
I25	I26	I27	I28	I29	I30
I31	I32	I33	I34	I35	I36

F1	F2	F3
F4	F5	F6
F7	F8	F9

Matrix

I1	I2	I3	I7	I8	I9	I13	I14	I15

Vec

F1
F2
F3
F4
F5
F6
F7
F8
F9

Boundary

IMPLEMENTATION 1: 2D CONV AS A GEMV

Image

I1	I2	I3	I4	I5	I6
I7	I8	I9	I10	I11	I12
I13	I14	I15	I16	I17	I18
I19	I20	I21	I22	I23	I24
I25	I26	I27	I28	I29	I30
I31	I32	I33	I34	I35	I36

F1	F2	F3
F4	F5	F6
F7	F8	F9

Matrix

I1	I2	I3	I7	I8	I9	I13	I14	I15
I2	I3	I4	I8	I9	I10	I14	I15	I16
I3	I4	I5	I9	I10	I11	I15	I16	I17
I9	I10	I11	I15	I16	I17	I21	I22	I23
I21	I22	I23	I27	I28	I29	I33	I34	I35
I28	I29	I30	I34	I35	I36	0	0	0

Boundary

Vec

F1
F2
F3
F4
F5
F6
F7
F8
F9

IMPLEMENTATION 1: 2D CONV AS A GEMV

Image

I1	I2	I3	I4	I5	I6
I7	I8	I9	I10	I11	I12
I13	I14	I15	I16	I17	I18
I19	I20	I21	I22	I23	I24
I25	I26	I27	I28	I29	I30
I31	I32	I33	I34	I35	I36

F1	F2	F3
F4	F5	F6
F7	F8	F9

Matrix

I1	I2	I3	I7	I8	I9	I13	I14	I15
I2	I3	I4	I8	I9	I10	I14	I15	I16
I3	I4	I5	I9	I10	I11	I15	I16	I17
I9	I10	I11	I15	I16	I17	I21	I22	I23
I21	I22	I23	I27	I28	I29	I33	I34	I35
I28	I29	I30	I34	I35	I36	0	0	0

Boundary

Vec

F1
F2
F3
F4
F5
F6
F7
F8
F9

CUDNN V1

- cuDNN provides high performance building blocks for deep neural network applications, including:
  - Forward and backward convolution routines, including cross-correlation, designed for convolutional neural nets
  - Arbitrary dimension ordering, striding, and sub-regions for 4d tensors means easy integration into any neural net implementation
  - Forward and backward paths for many common layer types such as pooling, ReLU, Sigmoid, softmax and Tanh
  - Tensor transformation functions
  - Context-based API allows for easy multithreading
  - Optimized for the latest NVIDIA GPU architectures
  - Supported on Windows, Linux and MacOS systems with Kepler, Maxwell or Tegra K1 GPUs.

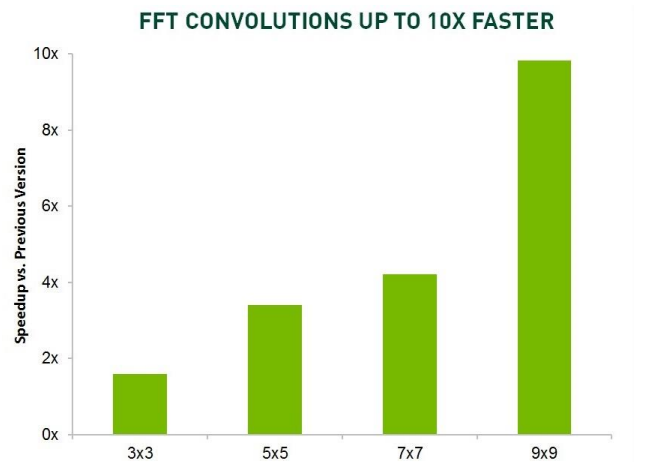
CUDNN V2

Support for 3D datasets. Community feedback desired

- OS X support
- Zero-padding of borders in pooling routines
- Parameter scaling
- Improved support for arbitrary strides
- Support for upcoming Tegra X1 via JIT compilation

CUDNN V3

- Train models up to 2x faster on Maxwell and Kepler-powered GPUs
  - 2D convolutions optimized specifically for Maxwell
  - FFT convolutions accelerate larger filter sizes (5x5 and better)
- Train up to 2x larger models with 16-bit floating point (FP16) data storage
- Heuristics to select the optimal algorithm for target hardware
- Support for 3D non-convolution layers
- LCN, LRN, and logarithmic softmax layer types
- Optional deterministic backprop



CUDNN V4

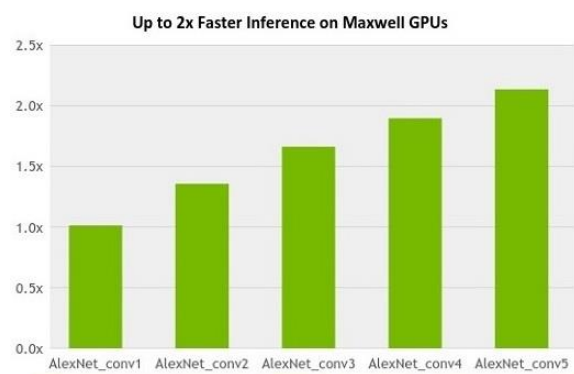
The new cuDNN 4 release delivers faster neural network training and is optimized for Maxwell GPUs.

•Train neural networks up to 14x faster using Google's Batch Normalization technique.

•Increase training and inference performance for convolutional layers up to 2x faster with new 2D tiled FFT algorithm.

•Accelerate inference performance for convolutional layers on small batch sizes up to 2x on Maxwell-architecture GPUs.

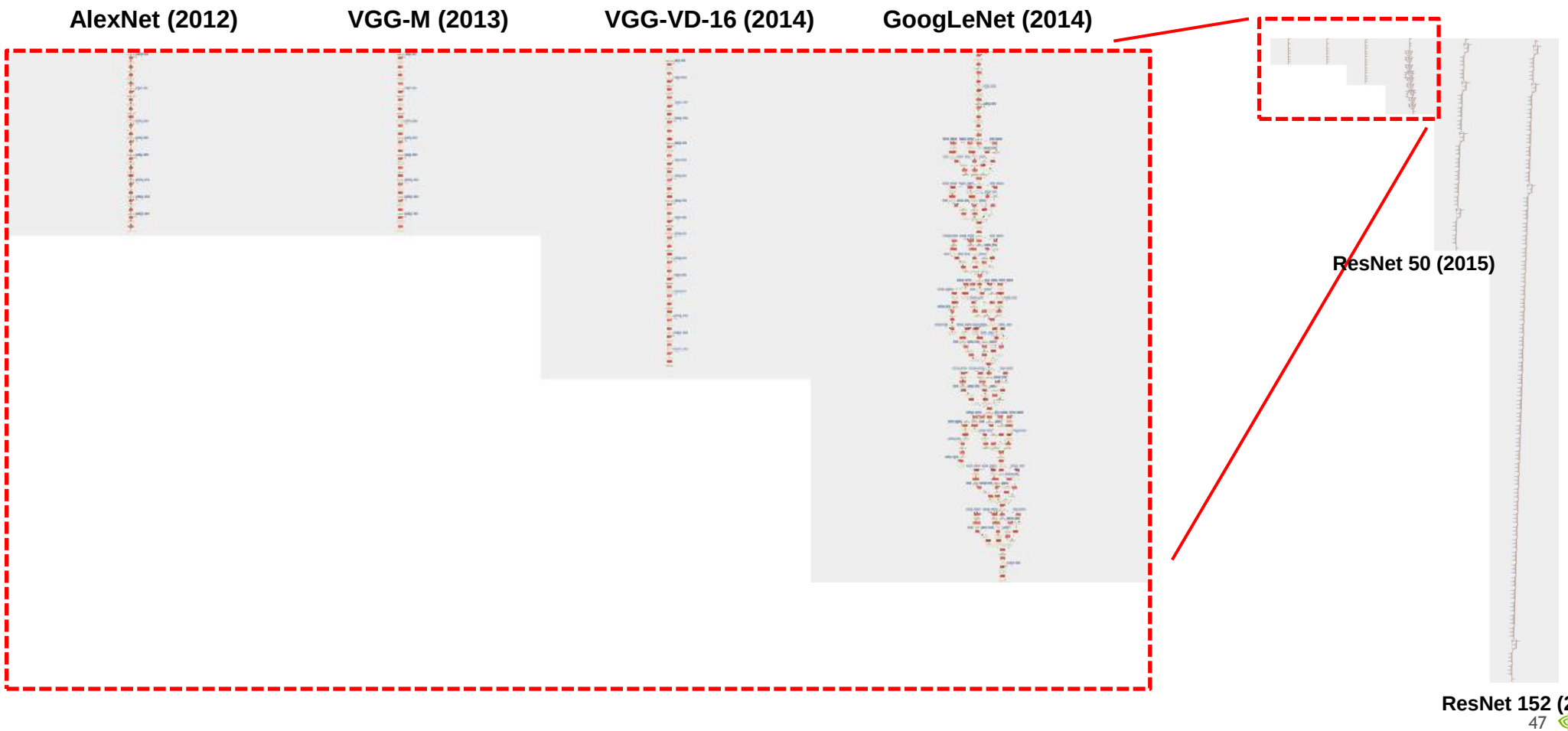
•Optimize for energy efficient inference with 10x better performance/Watt on Jetson TX1.



EVOLUTION OF CNN (2012~2014)

AlexNet (2012) **VGG-M (2013)** **VGG-VD-16 (2014)** **GoogLeNet (2014)**

EVOLUTION OF CNN (2015)



OF OPERATION

Layer1

input	Operation
W*H*C	W*H*C*K
224x224@3	Kernels K 11x11@192

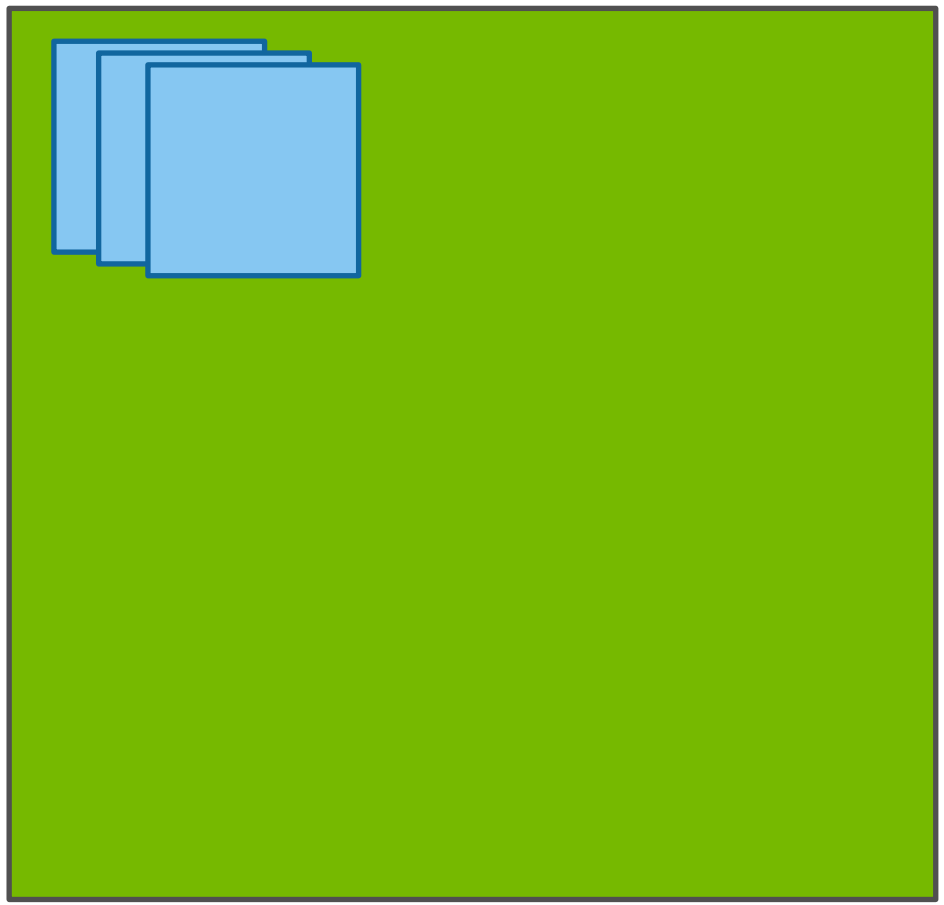
Increase # of Layer

manage input img
stide

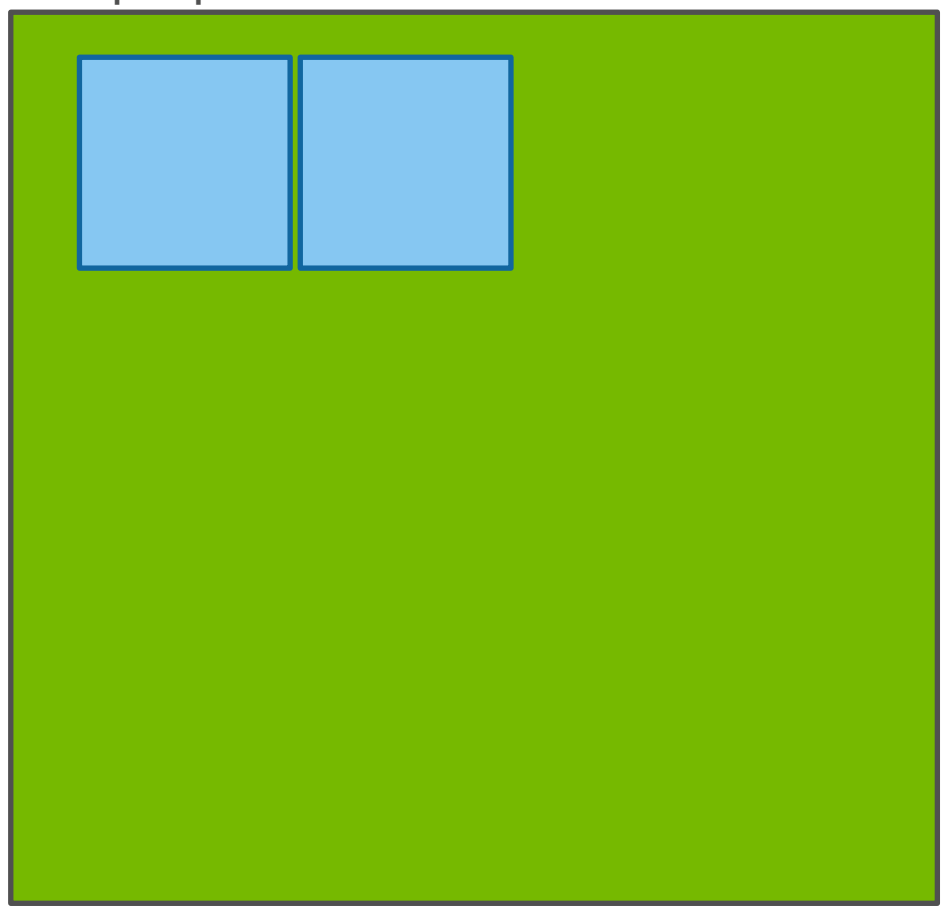
Manage kernel
Repeat kernel
Kernel group

STRIDE

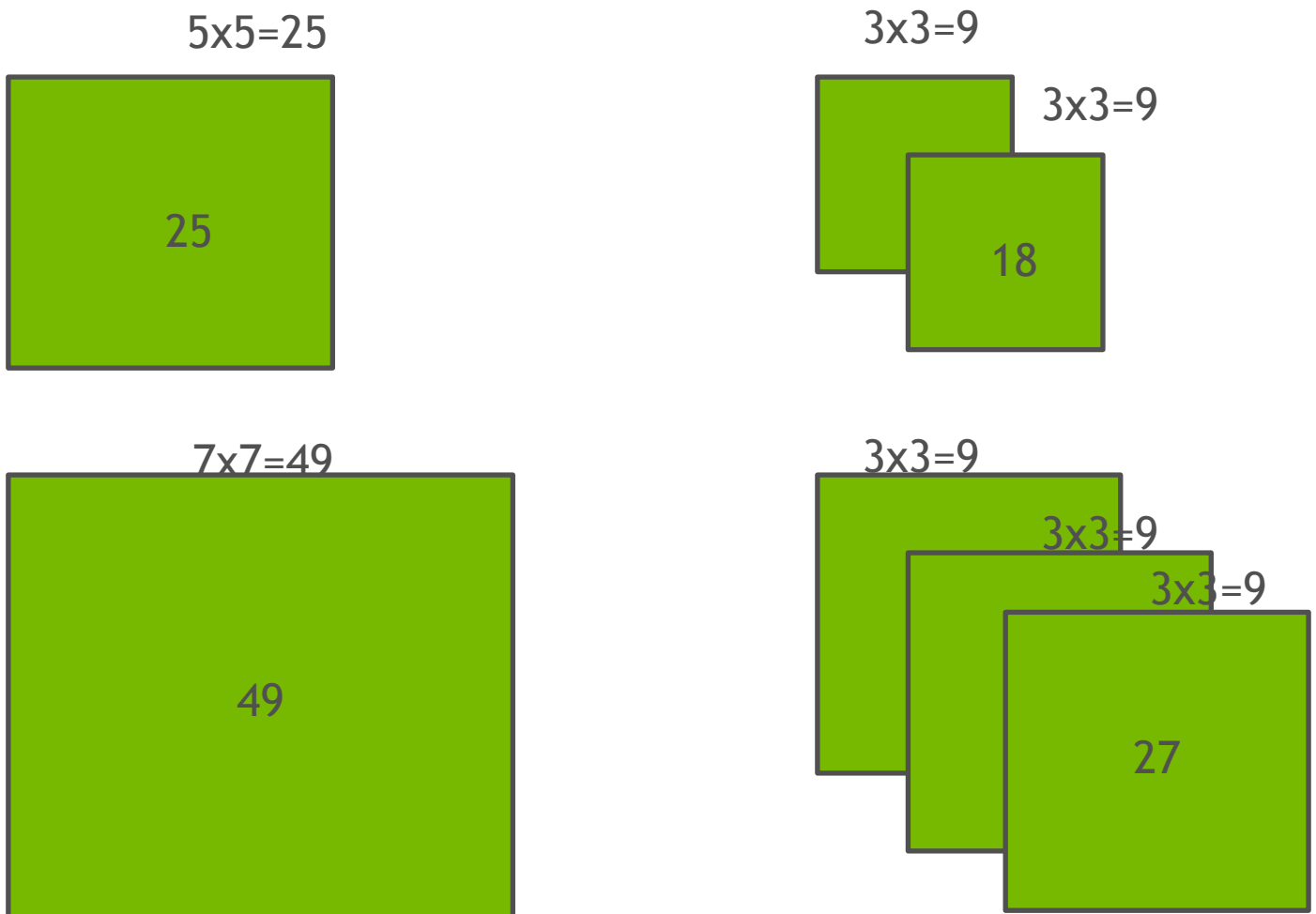
1 pixel movement



multiple pixel movement



RECEPTION (VGG)

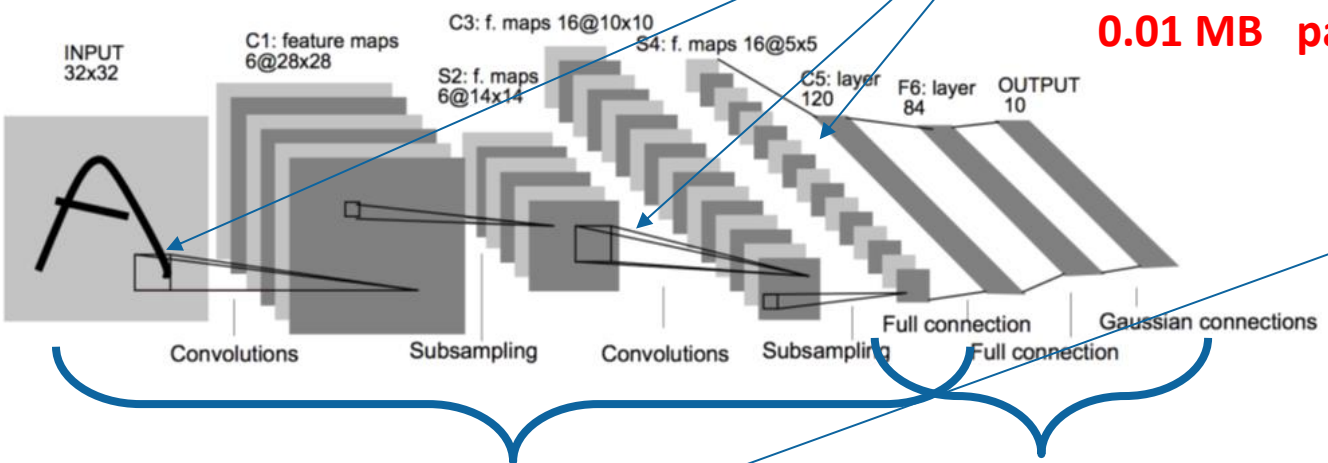


LENET5(1998)

Conv kernel

(5x5+1)*6 = 156
(5x5+1)*(3*6+4*9+5*1) = 1,534
(5x5+1)*(120) = 3,120

0.01 MB param



Neurons

In	1	32	32	1024
C1	6	28	28	4704
S2	6	14	14	1176
C3	16	10	10	1600
S4	16	5	5	400
C5	120	1	1	120
F6	84	1	1	84
Out	10	1	1	10
				9118

Conv

In	C1	S2	C3	S4	C5
1024	4704	1176	1600	400	120

49.3 MB param

ANN

In	hidden	out
120	84	10

0.04 MB param

Total 61 MB : Dataset 11MB + param 50M

ALEXNET(2012)

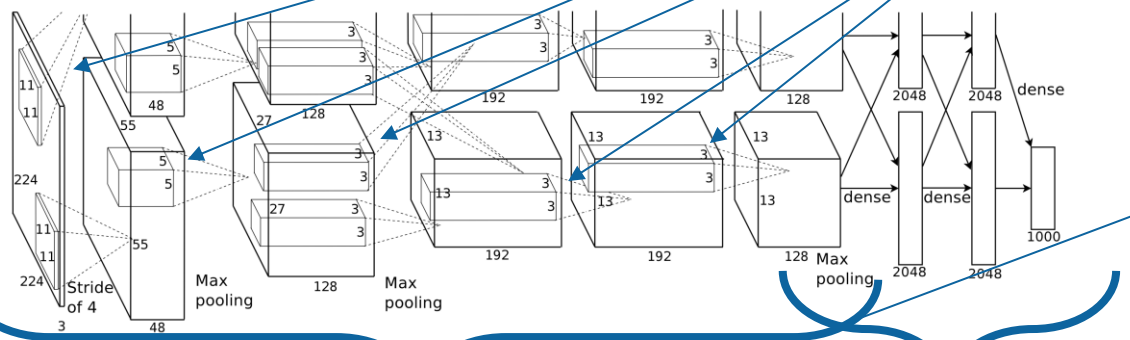
Conv kernel

(11x11+1)*96 = 11,712
(5x5+1)*(256) = 6,656
(3x3+1)*(384) = 3,840
(3x3+1)*(384) = 3,840
(3x3+1)*(256) = 2,560

0.1 MB param

Neurons

In	3	224	150,528
C1	96	55	290,400
C2	256	27	186,624
C3	384	13	64,896
C4	384	13	64,896
C5	256	13	43,264
F1	4096	1	4,096
F2	4096	1	4,096
Out	1000	1	1,000
			809,800



Conv

In	C1	C2	C3	C4	C5
150K	290K	186K	64K	64K	43K

436 MB param

ANN

In	hidden	hidden	out
43,264	4,096	4,096	1,000

0.7 MB param

Total 145GB : Dataset 145GB + param 436.8M

ALEXNET(7 LAYER) TO VGG(19 LAYER)

AlexNet

image
conv-64
conv-192
conv-384
conv-256
conv-256
FC-4096
FC-4096
FC-1000

image
conv-64
maxpool
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
FC-1000
softmax

image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
FC-1000
softmax

image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
conv-256
maxpool
conv-512
conv-512
conv-512
maxpool
conv-512
conv-512
conv-512
maxpool
FC-4096
FC-4096
FC-1000
softmax

image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
conv-256
conv-256
maxpool
conv-512
conv-512
conv-512
conv-512
maxpool
conv-512
conv-512
conv-512
conv-512
maxpool
FC-4096
FC-4096
FC-1000
softmax

19-layer

image
conv-64
conv-64
conv-128
conv-128
conv-256
conv-256
conv-256
conv-256
conv-512
conv-512
conv-512
conv-512
conv-512
conv-512
conv-512
conv-512
FC-4096
FC-4096
FC-1000

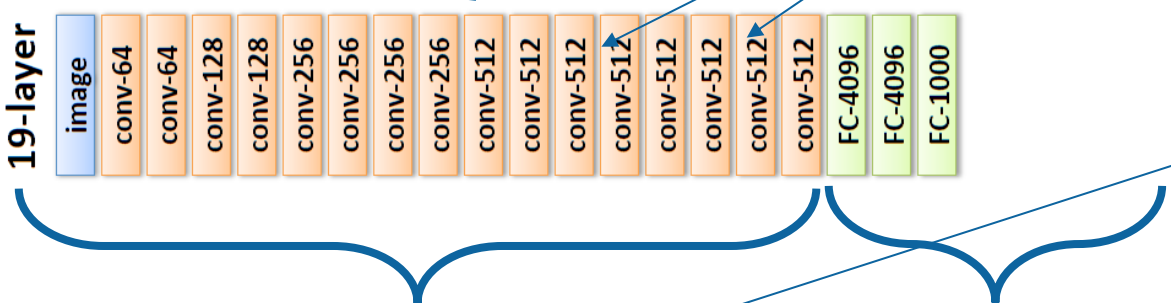
NVIDIA.

VGG(2014)

Conv kernel

(3x3+1)*(64) = 640
(3x3+1)*(128) = 1,280
(3x3+1)*(256) = 2,560
(3x3+1)*(512) = 5,120
(3x3+1)*(512) = 5,120
(3x3+1)*(512) = 5,120

0.1 MB param



Neurons

	ch	size	
input	3	224	150,528
Conv64	64	224	3,211,264
Conv64	64	224	3,211,264
Max	64	112	802,816
Conv128	128	112	1,605,632
Conv128	128	112	1,605,632
Max	128	56	401,408
Conv256	256	56	802,816
Conv256	256	56	802,816
Max	256	28	200,704
Conv512	512	28	401,408
Conv512	512	28	401,408
Max	512	14	100,352
Conv512	512	14	100,352
Conv512	512	14	100,352
Max	512	7	25,088
Conv512	512	7	25,088
Conv512	512	7	25,088
Max	512	1	512
FC4096	4,096	1	4,096
FC4096	4,096	1	4,096
FC1000	1,000	1	1,000

Conv

In	C64	C128	C256	C512	C512	C152
150K	802K	401K	200K	401K	100K	1k

547 MB param

ANN

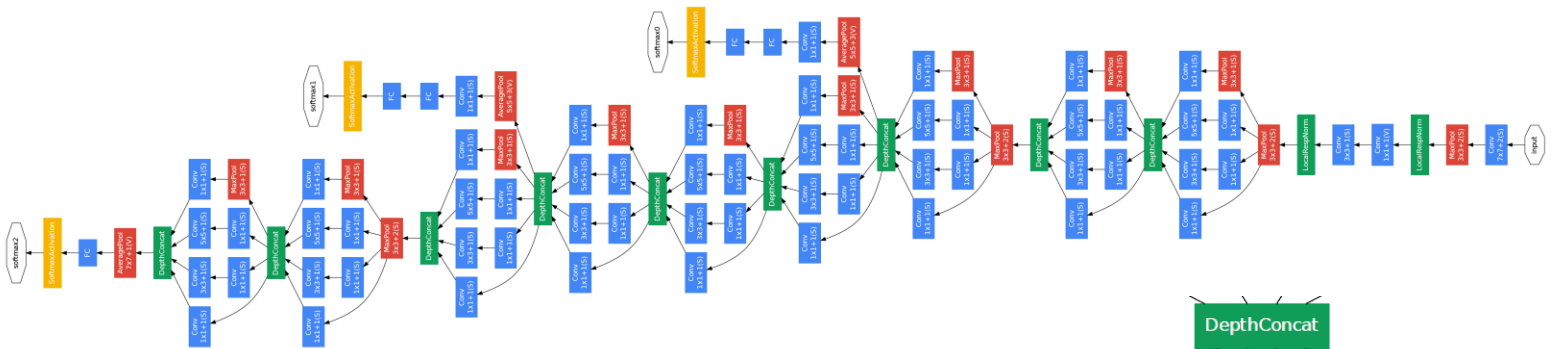
In	hidden	hidden	out
12,800	4,096	4,096	1,000

~1 MB param

Total 145GB : Dataset 145GB + param 548M

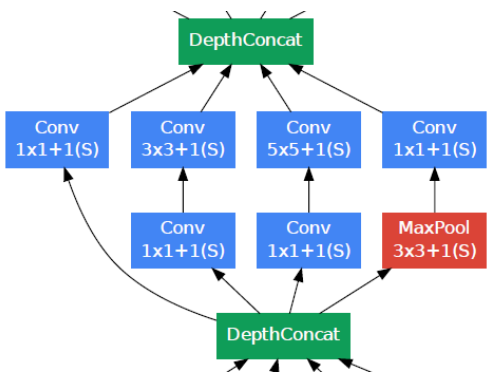
GOOGLENET(2014)

Our networks were trained using the DistBelief [4] distributed machine learning system using modest amount of model and data-parallelism. Although we used CPU based implementation only, a rough estimate suggests that the GoogLeNet network could be trained to convergence using few high-end GPUs within a week, the main limitation being the memory usage. Our training used asynchronous stochastic gradient descent with 0.9 momentum [17],



22 Layers = 3 + (9*2) + 1

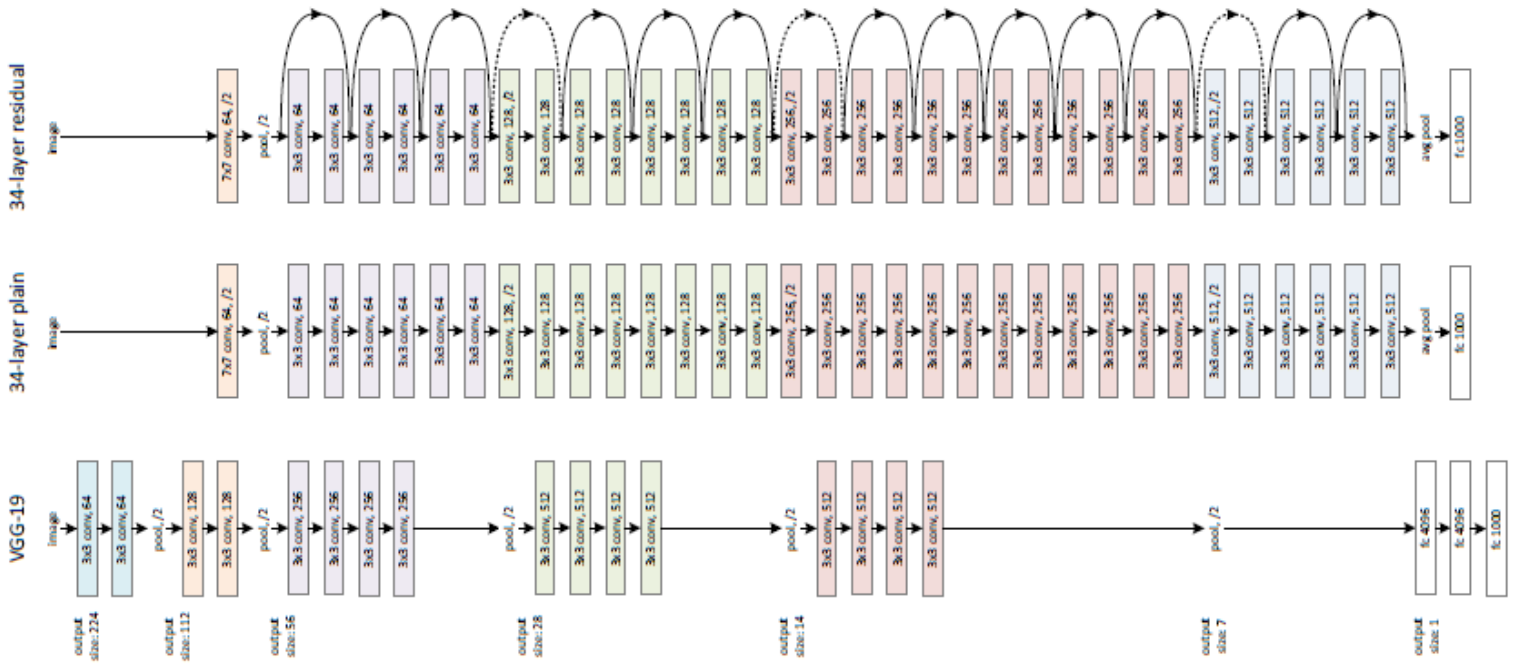
{ (5x5+1), (3x3+1), (1x1+1)



Neurons

	ch	size	
in	3	224	150,528
Conv1	64	112	802,816
max	64	56	200,704
Conv2	192	56	602,112
max	192	28	150,528
I 3a	256	28	200,704
I 3b	480	28	376,320
max	480	14	94,080
I 4a	512	14	100,352
I 4b	512	14	100,352
I 4c	512	14	100,352
I 4d	528	14	103,488
I 4e	832	14	163,072
max	832	7	40,768
I 5a	832	7	40,768
I 5b	1024	7	50,176
avg	1024	1	1,024
dropout	1024	1	1,024
linear	1000	1	1,000
softmax	1000	1	1,000

RESIDUAL DEEP NETWORK(2015)



Calculation

19 Layer	1.8* 10^9
34 Layer	3.6*10^9
50 Layer	3.8*10^9
101 Layer	7.6*10^9
152 layer	11.3*10^9

PCIE/GPU TOPOLOGY

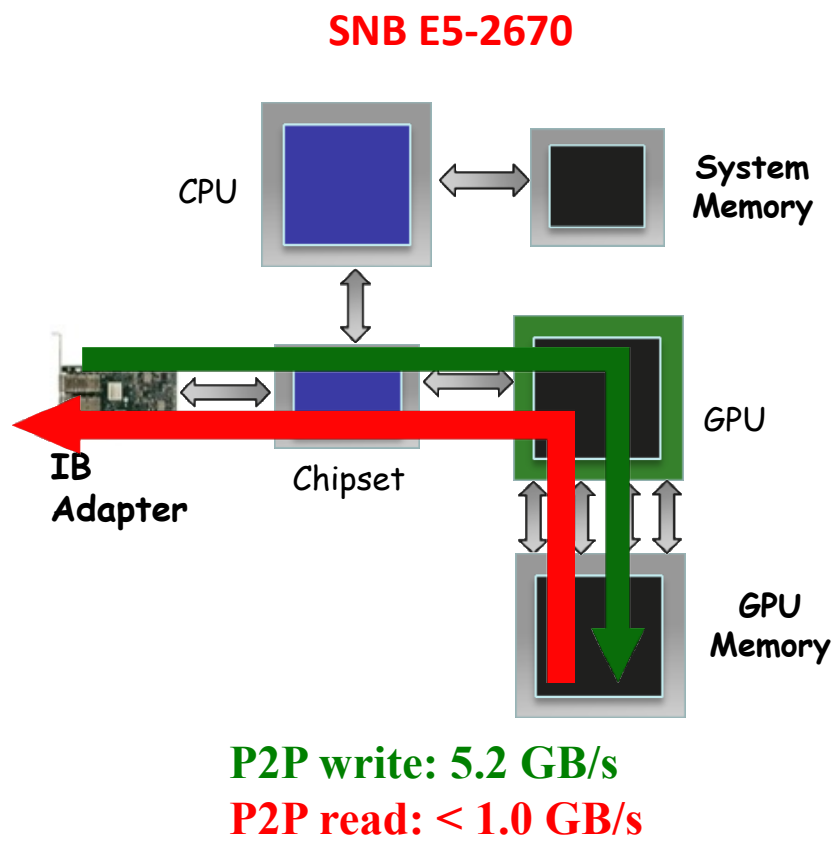
BIG SUR

FACEBOOK/QUANTA

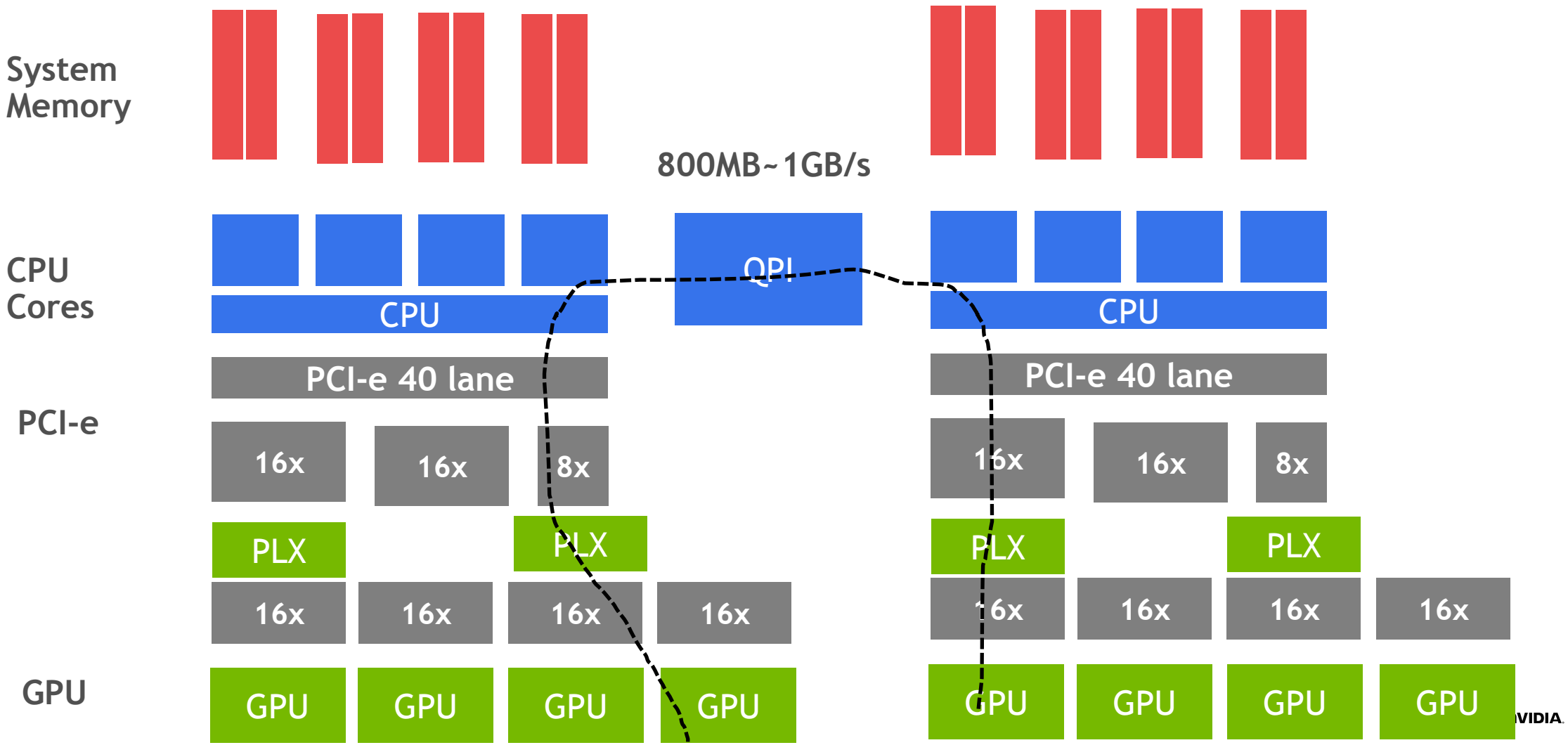
HISTORY/MOTIVATION

Xeon ICHs are crippled

- Westmere: read 1.5GB/s, write ~peak
- SNB: read 800MB/s, write 5.2GB/s
- IVB: read 3.7GB/s, write 6.1GB/s
- IVB+PLX:



PCI-E TOPOLOGY

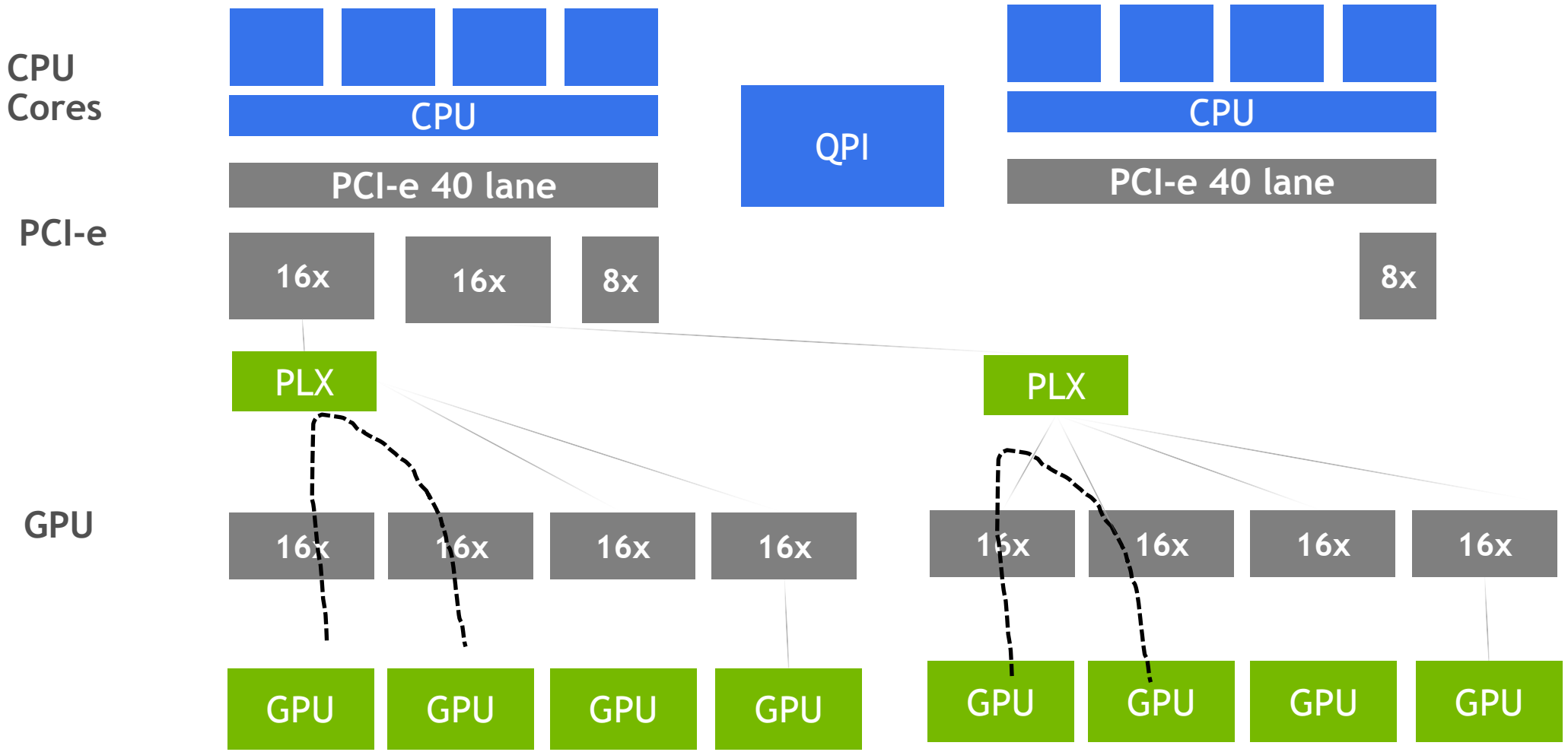


NVIDIA-SMI TOPO

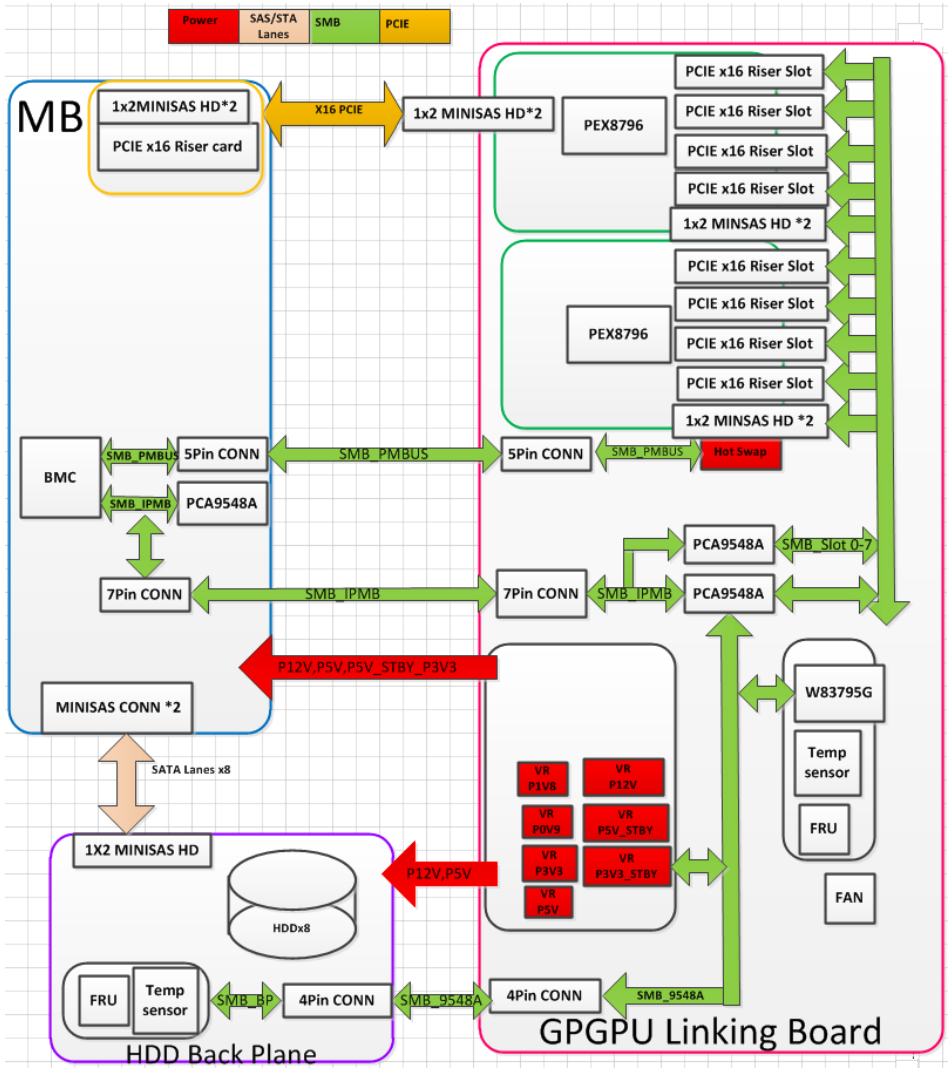
```
[hryu@ivb122 ~]$ nvidia-smi topo -m

      0     1     2     3     4     5     6     7   mlx5_0 CPU Affinity
GPU0   X    PIX   PXB   PXB   SOC   SOC   SOC   SOC   PHB   0-9
GPU1  PIX    X    PXB   PXB   SOC   SOC   SOC   SOC   PHB   0-9
GPU2  PXB   PXB    X    PIX   SOC   SOC   SOC   SOC   PHB   0-9
GPU3  PXB   PXB   PIX    X    SOC   SOC   SOC   SOC   PHB   0-9
GPU4  SOC   SOC   SOC   SOC    X    PIX   PHB   PHB   SOC   10-19
GPU5  SOC   SOC   SOC   SOC   PIX    X    PHB   PHB   SOC   10-19
GPU6  SOC   SOC   SOC   SOC   PHB   PHB    X    PIX   SOC   10-19
GPU7  SOC   SOC   SOC   SOC   PHB   PHB   PIX    X    SOC   10-19
mlx5_0 PHB  PHB   PHB   PHB   SOC   SOC   SOC   SOC    X
```

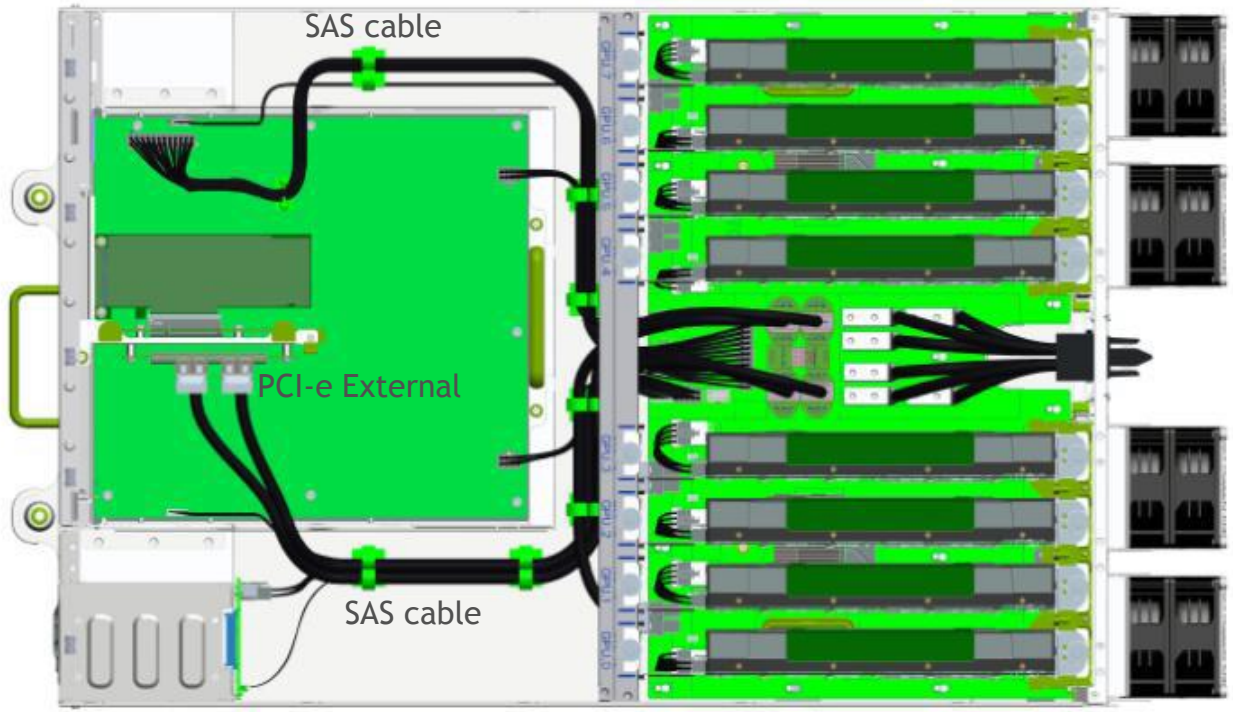
X = Self
SOC = Path traverses a socket-level link (e.g. QPI)
PHB = Path traverses a PCIe host bridge
PXB = Path traverses multiple PCIe internal switches
PIX = Path traverses a PCIe internal switch

PCI-E TOPOLOGY

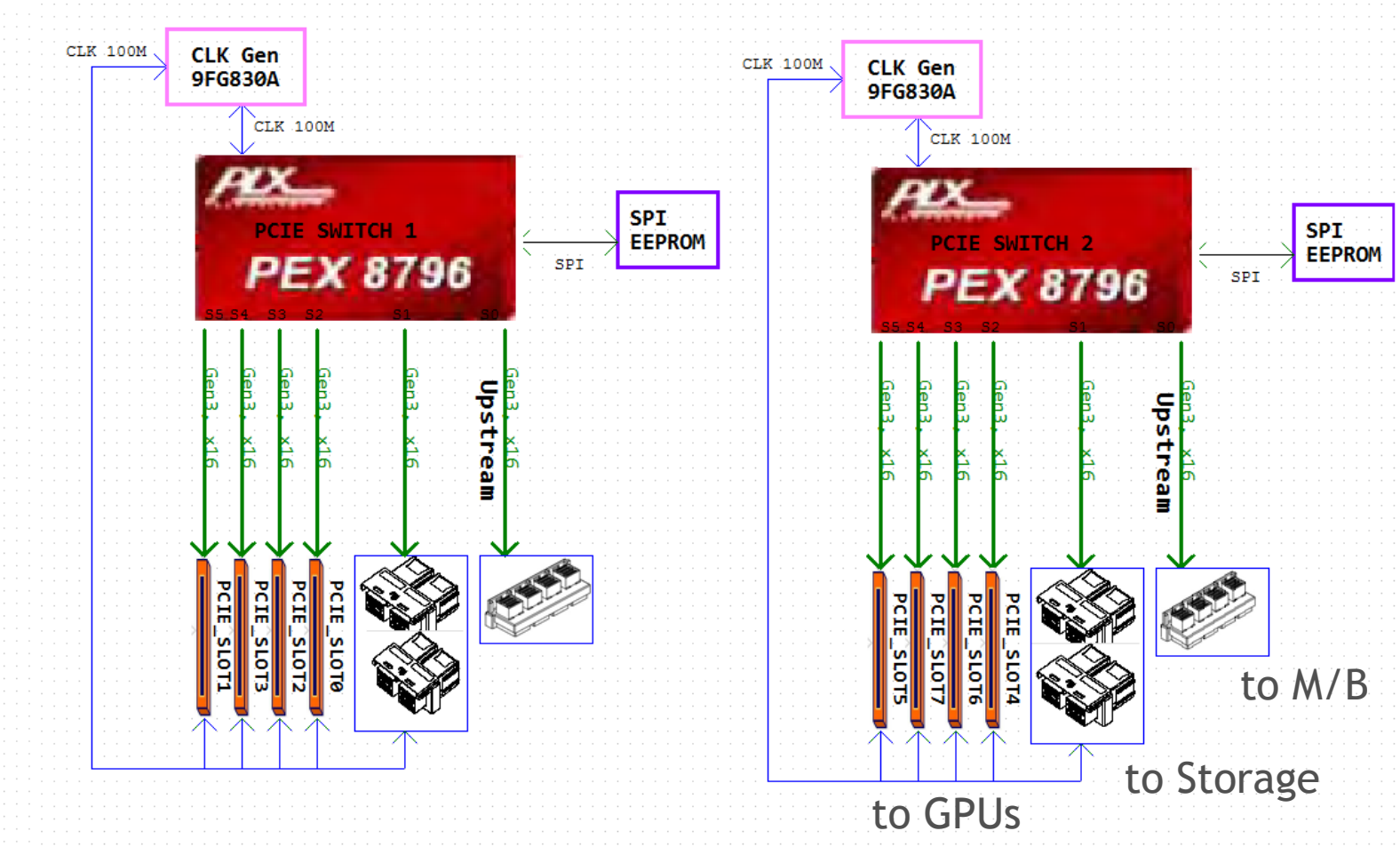


BLOCK DIAGRAM





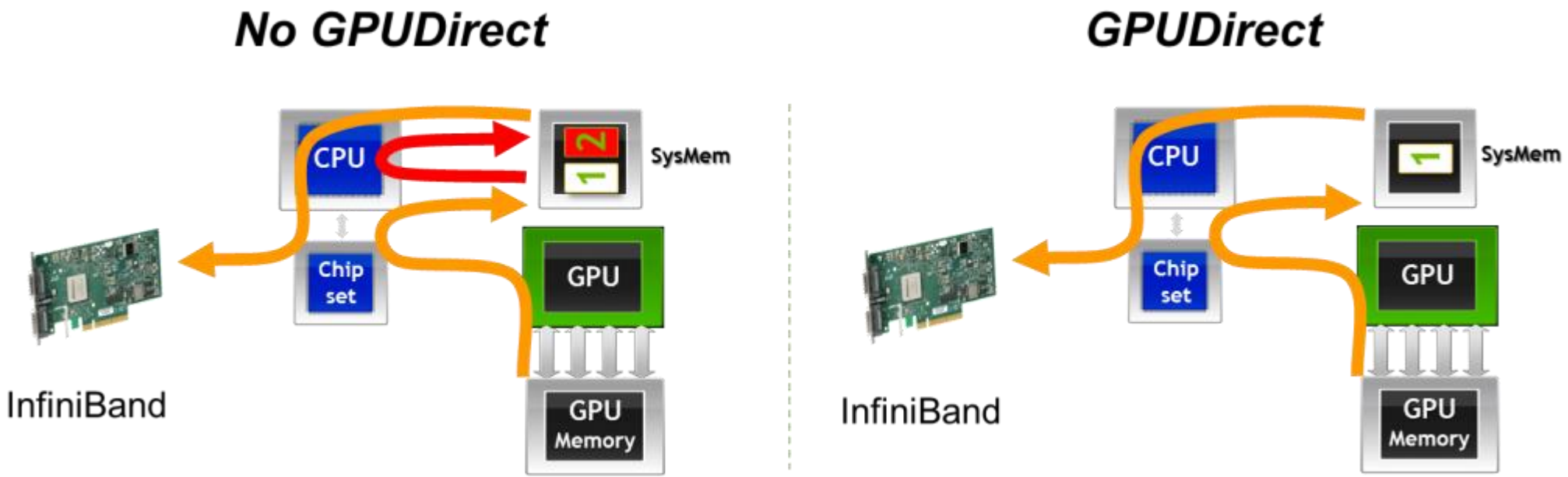
PCI-E PLX S/W


CLK 100M
CLK Gen
9FG830A
CLK 100M
PCIE SWITCH 1
PEX 8796
S5 S4 S3 S2 S1 S0
SPI
SPI
EEPROM
Gen3 x16
Gen3 x16
Gen3 x16
Gen3 x16
Gen3 x16
Gen3 x16
Upstream
PCIE_SLOT1
PCIE_SLOT3
PCIE_SLOT2
PCIE_SLOT0
CLK 100M
CLK Gen
9FG830A
CLK 100M
PCIE SWITCH 2
PEX 8796
S5 S4 S3 S2 S1 S0
SPI
SPI
EEPROM
Gen3 x16
Gen3 x16
Gen3 x16
Gen3 x16
Gen3 x16
Gen3 x16
Upstream
PCIE_SLOT5
PCIE_SLOT7
PCIE_SLOT6
PCIE_SLOT4
to M/B
to Storage
to GPUs

GPUDIRECT

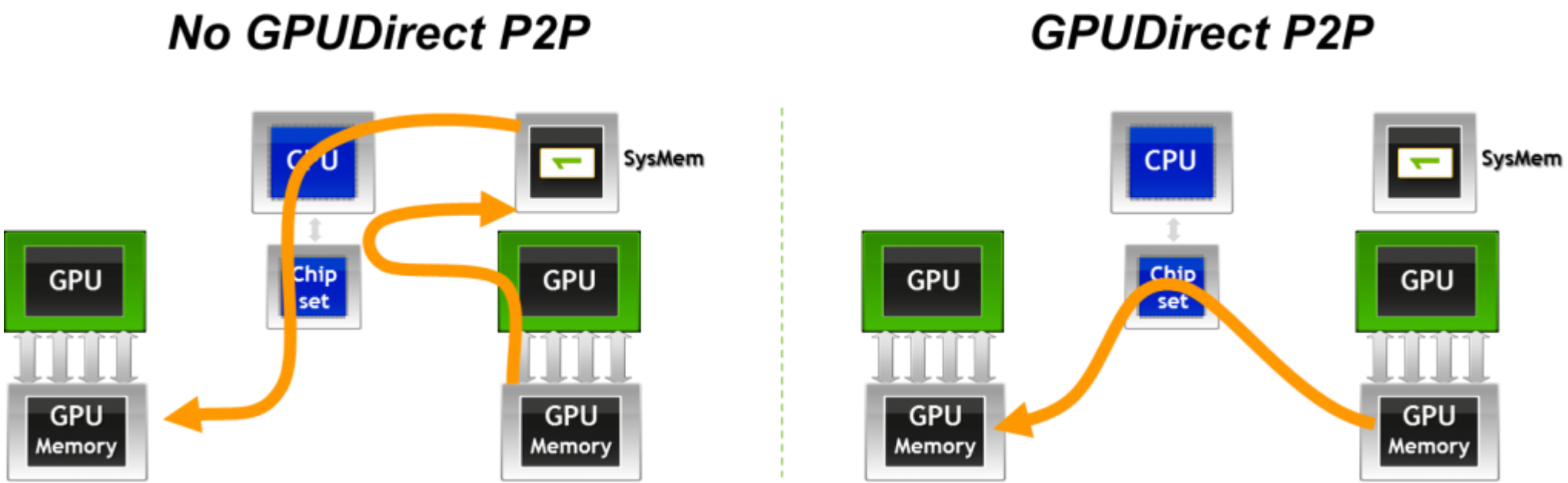
GPUDIRECT

Pinned memory Share



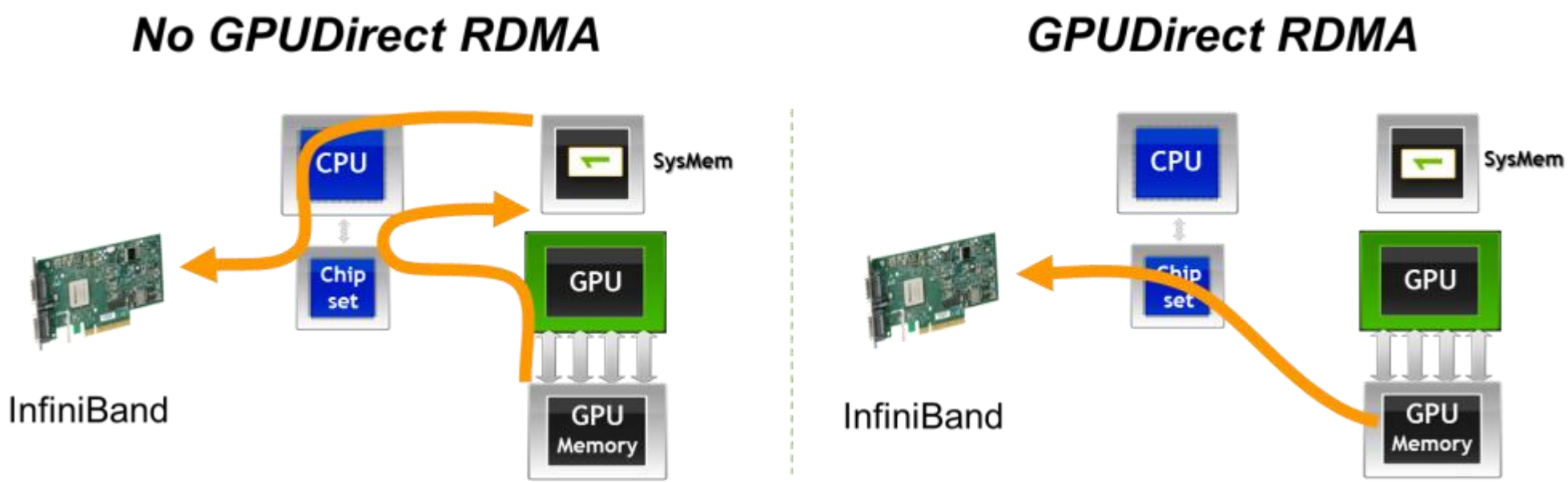
GPUDIRECT P2P

PCI-e DMA support



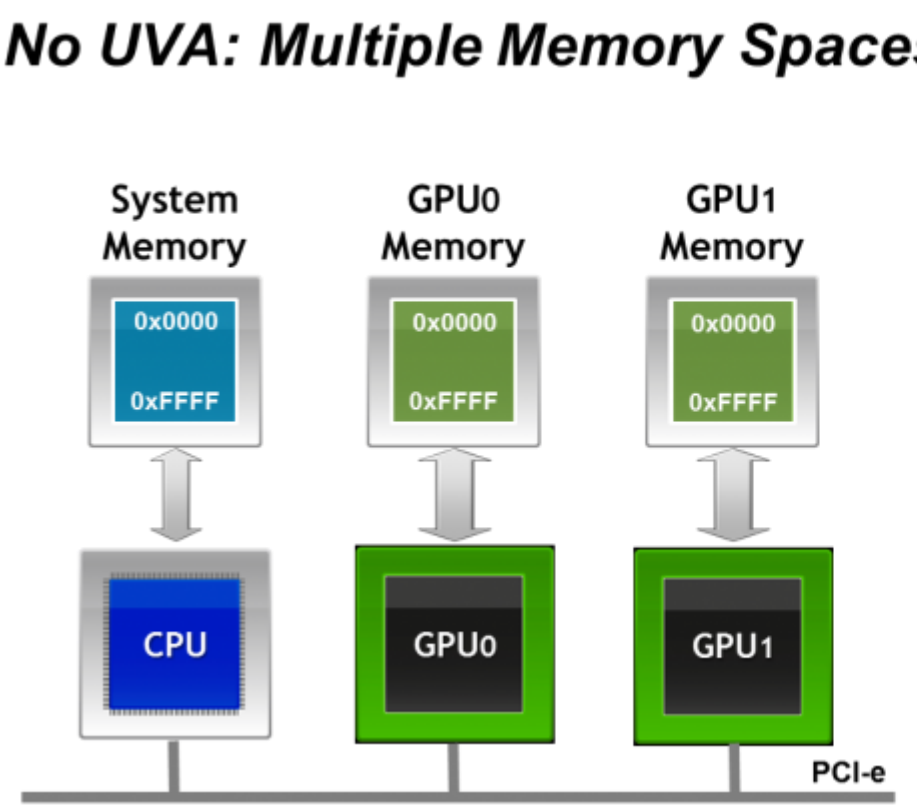
GPUDIRECT RDMA

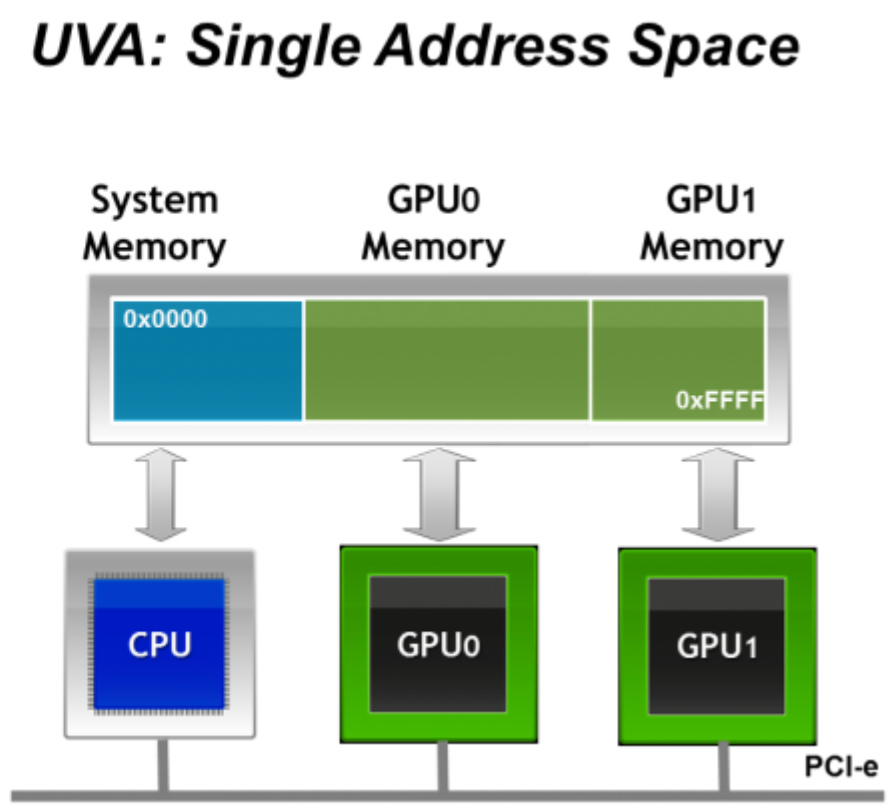
DMA + 3rd party P2P



UVA

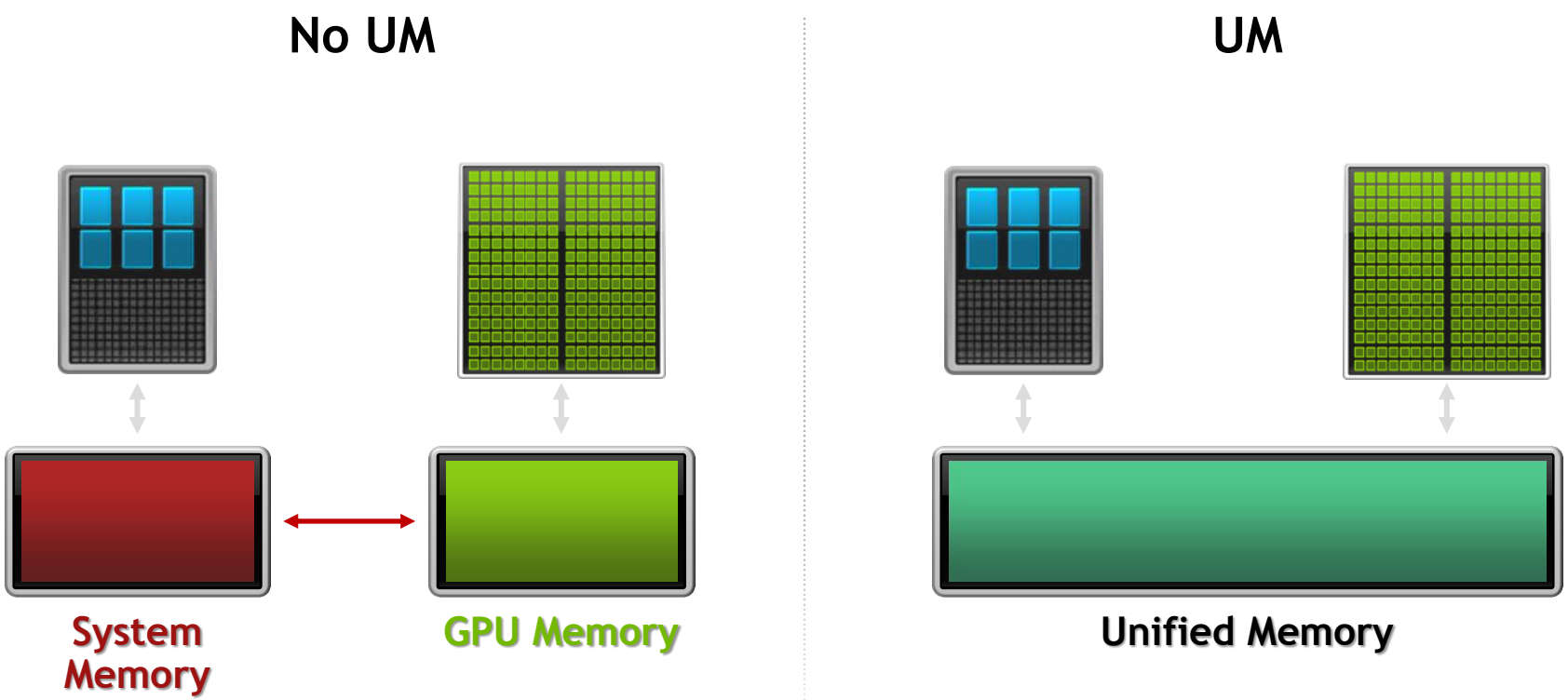
Unified Virtual Address





UM

Unified Memory



PCI BAR BACKGROUND

- BAR=Base Address Register
  - PCI resource
  - up to 6 32bits regs
  - 64bits BAR uses 2 regs
  - physical address range
  - prefetchable ~ cachable [danger here]
- most GPUs expose 3 BARs
- BAR1 space varies:
  - K20/40c=256MB
  - K40m=16GB [new!]

```
$ lspci -vv -s 1:0.0
01:00.0 3D controller: NVIDIA Corporation Device 1024 (rev a1)
        Subsystem: NVIDIA Corporation Device 0983
        Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr-
Stepping- SERR- FastB2B- DisINTx-
        Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort-
<MAbort- >SERR- <PERR- INTx-
        Latency: 0
        Interrupt: pin A routed to IRQ 86
        Region 0: Memory at fa000000 (32-bit, non-prefetchable) [size=16M]
        Region 1: Memory at b0000000 (64-bit, prefetchable) [size=256M]
        Region 3: Memory at c0000000 (64-bit, prefetchable) [size=32M]

$ nvidia-smi
…
 BAR1 Memory Usage
        Total                           : 256 MiB
        Used                            : 2 MiB
        Free                            : 254 MiB
```

PLATFORM-RELATED LIMITATIONS

PCIE topology and NUMA effects:

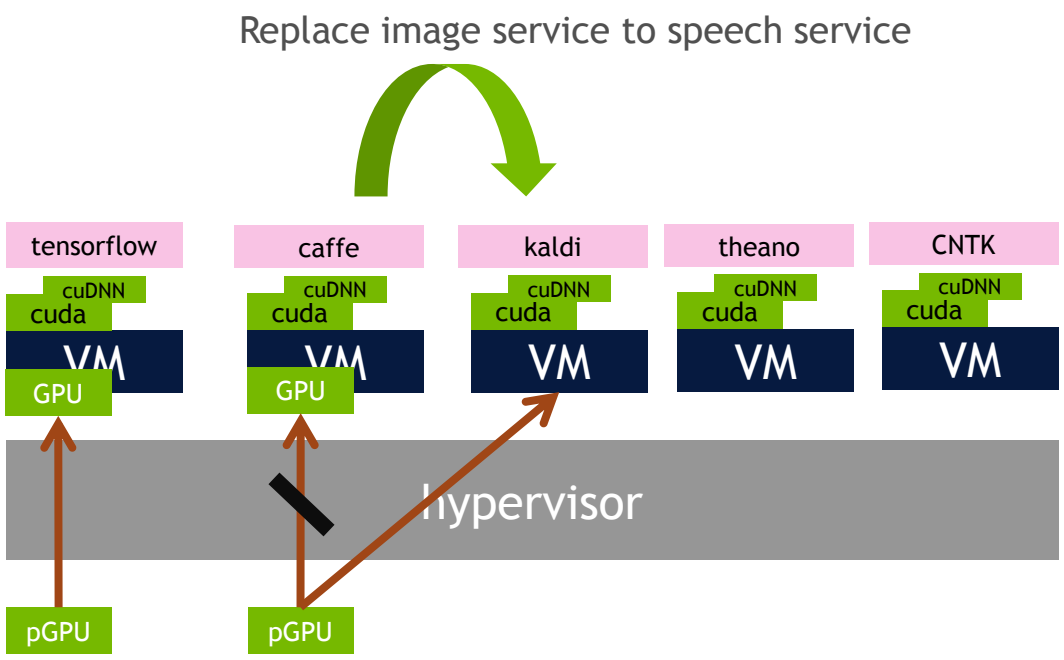
- Number and type of traversed chipsets/bridges/switches
  - GPU memory reading BW most affected
    - Sandy Bridge Xeon severely limited
  - On older chipsets, writing BW affected too
- Crossing inter-CPU bus huge bottleneck
  - currently allowed though

GPUDIRECTRDMA ON MELLANOX IB BOARDS

- Mellanox is de-facto standard for Cluster interconnect
- GPUDiretRDMA support early prototype started in Feb 2013
  - demoed at GTC 2013 on MVAPICH2 1.9
- rework for Mellanox SW stack
  - May 2013 - Dec 2013
  - shipping since 1/2014 MOFED 2.1
  - OpenMPI >= 1.7.4
  - MVAPICH2 2.0b

GPU Tools & Monitoring

GPU VIRTUALIZATON FOR DL



VM Sinarios

Support DL Scientists
Deploy TF VM with 1EA M40 for new employee

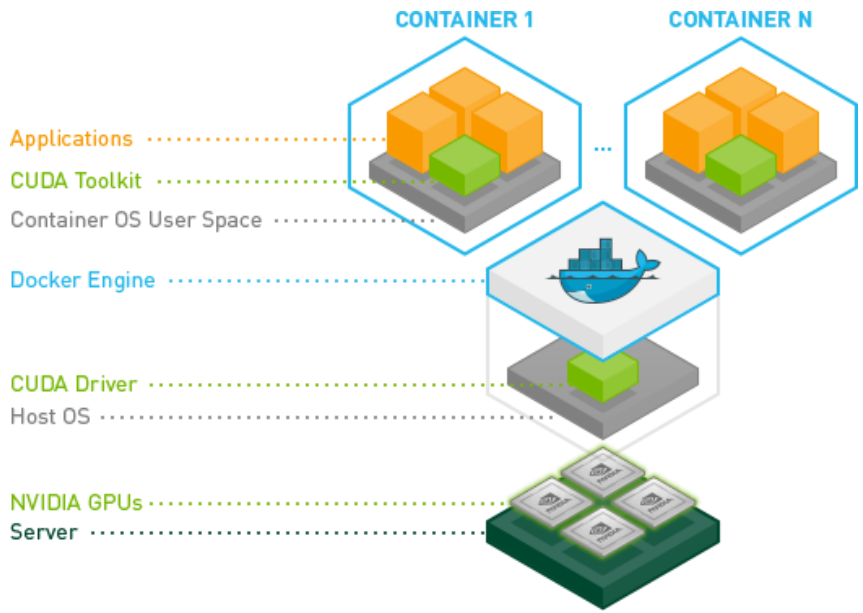
Support On-demand scalability for DL inference
Add 100 CCUs for speech service

NV-DOCKER

build and run NVIDIA Docker images.

Containerizing GPU applications provides several benefits, among them:

- **Reproducible builds**
- **Ease of deployment**
- **Isolation of individual devices**
- **Run across heterogeneous driver/toolkit environments**
- **Requires only the NVIDIA driver to be installed**
- **Enables "fire and forget" GPU applications**
- **Facilitate collaboration**



https://github.com/NVIDIA/nvidia-docker

APACHE MESOS

Native support for GPUs

- https://reviews.apache.org/r/44360/

  This script can be used to install the Nvidia GDK for Cuda 7.5 on a mesos development machine. The purpose of the Nvidia GDK is to provide all the necessary header files (nvml.h) and library files (libnvidia-ml.so) necessary to build mesos with Nvidia GPU support.

- https://reviews.apache.org/r/44361/

  This is the initial commit to begin adding native support for GPUs in Mesos. This initial version will only include support for Nvidia GPUs that can be managed by the Nvidia Management Library (nvml).

- https://reviews.apache.org/r/44363/

  This is the first isolator to fall under the category of cgroup devices.

- https://reviews.apache.org/r/44366/

  Currently we can either infer the number of GPUs to offer as resources from the --nvidia_gpus flag, or pass the amount explicity as a "gpus" resource. In the future, we will generalize this via autodiscovery of gpus and support for gpus types other than Nvidia.

GPU MON/MGM TOOL

- NVIDIA-healthmon
- Nvidia-smi
- Nvml library
- CUDA SDK
  - P2platencytest
  - Bandwidthtest
- NCCL

NVIDIA-HEALTHMON

nvidia-healthmon

```
 . . .
System Results: 61 success, 0 errors, 0 warnings, 72 did not run
One or more tests didn't run.
```

nvidia-healthmon -e

```
 . . .
System Results: 93 success, 0 errors, 0 warnings, 40 did not run
One or more tests didn't run.
```

nvidia-healthmon -L

```
    . . .

GPU 7:
   PCI Bus ID: 0000:8A:00.0
   Name: Tesla K80
   UUID: GPU-85578e93-4eaf-847d-e22a-f6525382e868
```

NVIDIA-HEALHMON HELP

```
Options include:
    [-h | --help]: Print usage
    [-H | --verbose-help]: Print detailed usage
    [-v | --verbose]: Enable verbose output
    [-V | --version]: Prints the version number
    [-q | --quick]: Execute a subset of tests
    [-e | --extended]: Execute the complete test suite
    [-i | --id]: Target a specific GPU
    [-L | --list-devices]: List all the GPUs attached
    [-c | --config]: Path to the configuration file
    [-d | --debug]: Use this flag to enable debug logging, and specify the log path
    [-l | --log-file]: Write all output to a file instead of STDOUT
    [-t | --timeout]: Maximum time allowed to run, in seconds, per test;
                      the default is 120. A value of 0 disables this feature.
```

NVIDIA-SMI/NVML

```
+-----------------------------------------------------+
| NVIDIA-SMI 352.68     Driver Version: 352.68         |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla K80           On   | 0000:06:00.0     Off |                    0 |
| N/A   36C    P8    28W / 149W |     55MiB / 11519MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla K80           On   | 0000:07:00.0     Off |                    0 |
| N/A   33C    P8    40W / 149W |     55MiB / 11519MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   2  Tesla K80           On   | 0000:0A:00.0     Off |                    0 |
| N/A   34C    P8    29W / 149W |     55MiB / 11519MiB |      0%      Default |

   .  . .

+-------------------------------+----------------------+----------------------+
|   7  Tesla K80           On   | 0000:8A:00.0     Off |                    0 |
| N/A   35C    P8    40W / 149W |     55MiB / 11519MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID  Type  Process name                               Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

NVIDIA-SMI HELP

```
 -pm,  --persistence-mode=   Set persistence mode: 0/DISABLED, 1/ENABLED
 -e,   --ecc-config=         Toggle ECC support: 0/DISABLED, 1/ENABLED
 --gom=                      Set GPU Operation Mode: 0/ALL_ON, 1/COMPUTE, 2/LOW_DP
 --query-supported-clocks=   List of supported clocks.
                                Call --help-query-supported-clocks for more info.
 -ac   --applications-clocks= Specifies <memory,graphics> clocks as a pair (e.g. 2000,800) that defines GPU's
                               speed in MHz while running applications on a GPU.
    -rac  --reset-applications-clocks : Resets the applications clocks to the default values.
    -acp  --applications-clocks-permission=   Toggles permission requirements for -ac and -rac commands: 0/UNRESTRICTED, 1/RESTRICTED
-pl   --power-limit=        Specifies maximum power management limit in watts
```

NVIDIA-SMI EXAMPLE FOR CLOCK

nvidia-smi -i 0 --query-supported-clocks=graphics,memory --format=csv

```
graphics [MHz], memory [MHz]
875 MHz, 2505 MHz
862 MHz, 2505 MHz
849 MHz, 2505 MHz
836 MHz, 2505 MHz
823 MHz, 2505 MHz
810 MHz, 2505 MHz
797 MHz, 2505 MHz
784 MHz, 2505 MHz
771 MHz, 2505 MHz
758 MHz, 2505 MHz
745 MHz, 2505 MHz
732 MHz, 2505 MHz
719 MHz, 2505 MHz
705 MHz, 2505 MHz
692 MHz, 2505 MHz
679 MHz, 2505 MHz
666 MHz, 2505 MHz
653 MHz, 2505 MHz
640 MHz, 2505 MHz
627 MHz, 2505 MHz
614 MHz, 2505 MHz
601 MHz, 2505 MHz
588 MHz, 2505 MHz
575 MHz, 2505 MHz
562 MHz, 2505 MHz
324 MHz, 324 MHz
```

CUDA SDK

- simpleP2P
- p2pBandwidthLatencyTest
- BandwidthTest

GPU BW TEST

```
Running on...

 Device 0: Tesla K80
 Quick Mode

 Host to Device Bandwidth, 1 Device(s)
 PINNED Memory Transfers
   Transfer Size (Bytes)  Bandwidth(MB/s)
   33554432               7687.8

 Device to Host Bandwidth, 1 Device(s)
 PINNED Memory Transfers
   Transfer Size (Bytes)  Bandwidth(MB/s)
   33554432               7717.5

 Device to Device Bandwidth, 1 Device(s)
 PINNED Memory Transfers
   Transfer Size (Bytes)  Bandwidth(MB/s)
   33554432               140721.2

Result = PASS
```

GPU P2P TEST

```
Checking for multiple GPUs...
CUDA-capable device count: 8
> GPU0 = "      Tesla K80" IS  capable of Peer-to-Peer (P2P)
> GPU1 = "      Tesla K80" IS  capable of Peer-to-Peer (P2P)
> GPU2 = "      Tesla K80" IS  capable of Peer-to-Peer (P2P)
> GPU3 = "      Tesla K80" IS  capable of Peer-to-Peer (P2P)
> GPU4 = "      Tesla K80" IS  capable of Peer-to-Peer (P2P)
> GPU5 = "      Tesla K80" IS  capable of Peer-to-Peer (P2P)
> GPU6 = "      Tesla K80" IS  capable of Peer-to-Peer (P2P)
> GPU7 = "      Tesla K80" IS  capable of Peer-to-Peer (P2P)

Checking GPU(s) for support of peer to peer memory access...
> Peer access from Tesla K80 (GPU0) -> Tesla K80 (GPU1) : Yes
> Peer access from Tesla K80 (GPU0) -> Tesla K80 (GPU2) : Yes
> Peer access from Tesla K80 (GPU0) -> Tesla K80 (GPU3) : Yes
> Peer access from Tesla K80 (GPU0) -> Tesla K80 (GPU4) : No
> Peer access from Tesla K80 (GPU0) -> Tesla K80 (GPU5) : No
> Peer access from Tesla K80 (GPU0) -> Tesla K80 (GPU6) : No
> Peer access from Tesla K80 (GPU0) -> Tesla K80 (GPU7) : No
```

P2P BW/ LATENCY TEST

Bidirectional P2P=Disabled Bandwidth Matrix (GB/s)

D\D	0	1	2	3	4	5	6	7
0	85.71	9.11	8.45	8.43	14.01	15.80	15.68	15.70
1	8.79	72.15	8.68	8.47	13.85	14.83	14.93	14.88
2	8.53	8.57	72.20	9.05	13.34	14.21	14.99	14.77
3	8.09	7.66	8.50	77.38	13.72	14.65	14.37	15.30
4	14.21	14.10	14.05	14.14	72.28	8.37	13.93	14.11
5	15.65	15.32	14.94	15.20	8.41	81.31	14.09	14.05
6	14.26	14.07	14.17	14.21	14.03	14.32	72.10	8.46
7	14.28	14.11	14.19	14.22	14.01	14.57	8.34	81.08

Bidirectional P2P=Enabled Bandwidth Matrix (GB/s)

D\D	0	1	2	3	4	5	6	7
0	86.80	19.36	17.49	18.62	14.94	16.47	15.73	15.68
1	19.39	85.12	17.58	18.63	15.40	16.19	15.46	15.67
2	17.55	17.59	81.16	17.67	14.76	15.80	15.59	15.58
3	18.68	18.66	18.73	85.01	15.17	16.07	15.37	15.39
4	15.76	15.64	15.50	15.71	81.25	17.63	17.39	17.38
5	16.38	16.54	16.31	16.45	17.62	85.50	17.36	17.38
6	15.82	15.73	15.01	15.03	17.39	17.39	80.94	17.57
7	15.65	15.63	15.69	15.73	17.39	17.39	17.64	81.15

P2P=Disabled Latency Matrix (us)

D\D	0	1	2	3	4	5	6	7
0	5.96	28.51	28.70	28.58	27.85	27.35	27.67	28.22
1	29.21	4.87	28.38	29.46	28.17	27.85	26.90	26.80
2	29.82	28.38	4.91	30.12	27.65	28.70	27.26	27.30
3	31.70	30.58	30.06	5.84	29.13	29.91	28.68	28.12
4	29.20	28.02	28.86	28.98	5.59	27.54	26.44	26.64
5	29.17	28.07	28.46	29.11	26.99	5.55	26.00	25.95
6	28.33	27.52	27.57	28.45	26.35	26.90	4.70	25.61
7	28.07	27.56	27.51	28.47	26.29	26.97	25.52	4.72

P2P=Enabled Latency Matrix (us)

D\D	0	1	2	3	4	5	6	7
0	5.89	8.13	8.87	9.36	29.89	30.43	29.11	29.00
1	9.34	4.92	8.82	9.70	27.43	27.76	26.81	27.13
2	9.37	8.39	4.92	9.79	27.66	27.94	27.63	27.31
3	9.54	8.40	8.23	5.81	29.12	30.21	28.20	28.11
4	29.12	28.53	28.43	28.97	5.59	9.12	7.27	7.23
5	29.48	28.01	28.36	28.89	7.76	5.75	7.28	7.22
6	28.18	27.57	27.60	29.05	7.66	9.16	4.73	7.21
7	28.11	27.50	28.14	28.50	7.67	9.18	7.21	4.71

NCCL LIB

Optimized primitives for collective multi-GPU communication

- NCCL (pronounced "Nickel") is a stand-alone library of standard collective communication routines, such as all-gather, reduce, broadcast, etc., that have been optimized to achieve high bandwidth over PCIe. NCCL supports an arbitrary number of GPUs installed in a single node and can be used in either single- or multi-process (e.g., MPI) applications.
  - all-reduce
  - all-gather
  - reduce-scatter
  - reduce
  - broadcast

- https://github.com/NVIDIA/nccl

NCCL LIB

Easy installation

Download src

```
[hryu@ivb121 nccl]$ git clone https://github.com/NVIDIA/nccl.git
```

cuda compiler & gcc

```
[hryu@ivb121 nccl]$ module load cuda/7.5
[hryu@ivb121 nccl]$ which nvcc
/shared/apps/cuda/CUDA-v7.5.18/bin/nvcc
[hryu@ivb121 nccl]$ module load gcc/4.8.4
```

Build lib

```
[hryu@ivb121 nccl]$ make -j 20
Grabbing  src/nccl.h                > build/include/nccl.h
Compiling src/libwrap.cu            > build/obj/libwrap.o
Compiling src/core.cu               > build/obj/core.o
Compiling src/all_gather.cu         > build/obj/all_gather.o
Compiling src/all_reduce.cu         > build/obj/all_reduce.o
Compiling src/broadcast.cu          > build/obj/broadcast.o
Compiling src/reduce.cu             > build/obj/reduce.o
Compiling src/reduce_scatter.cu     > build/obj/reduce_scatter.o
Linking   build/lib/libnccl.so.1.0.2
```

Build sample

```
[hryu@ivb121 nccl]$ make -j 20 test
Building  test/single/all_gather_test.cu > build/test/single/all_gather_test
Building  test/single/all_reduce_test.cu > build/test/single/all_reduce_test
Building  test/single/broadcast_test.cu > build/test/single/broadcast_test
Building  test/single/reduce_test.cu > build/test/single/reduce_test
Building  test/single/reduce_scatter_test.cu > build/test/single/reduce_scatter_test
```

NCCL RESULT

reduce

```
[hryu@ivb121 single]$ ./reduce_test 10485760  2 0 1
# Using devices
#   Rank  0 uses device  0 [0x06] Tesla K80
#   Rank  1 uses device  1 [0x07] Tesla K80

#                                                     out-of-place                    in-place
#      bytes             N    type      op  root    time  algbw  busbw      res     time  algbw  busbw      res
    10485760      10485760    char     sum    0    1.564   6.71   6.71    0e+00    1.579   6.64   6.64    0e+00
    10485760      10485760    char     sum    1    1.525   6.87   6.87    0e+00    1.541   6.80   6.80    0e+00
    10485760      10485760    char    prod    0    1.749   6.00   6.00    0e+00    1.783   5.88   5.88    0e+00
    10485760      10485760    char    prod    1    1.741   6.02   6.02    0e+00    1.767   5.93   5.93    0e+00
    10485760      10485760    char     max    0    1.529   6.86   6.86    0e+00    1.552   6.76   6.76    0e+00
    10485760      10485760    char     max    1    1.519   6.90   6.90    0e+00    1.538   6.82   6.82    0e+00
    10485760      10485760    char     min    0    1.528   6.86   6.86    0e+00    1.546   6.78   6.78    0e+00
    10485760      10485760    char     min    1    1.517   6.91   6.91    0e+00    1.535   6.83   6.83    0e+00
    10485760       2621440     int     sum    0    1.524   6.88   6.88    0e+00    1.545   6.79   6.79    0e+00
    10485760       2621440     int     sum    1    1.513   6.93   6.93    0e+00    1.530   6.85   6.85    0e+00
    10485760       2621440     int    prod    0    1.486   7.06   7.06    0e+00    1.501   6.99   6.99    0e+00
    10485760       2621440     int    prod    1    1.478   7.09   7.09    0e+00    1.490   7.04   7.04    0e+00
    10485760       2621440     int     max    0    1.520   6.90   6.90    0e+00    1.539   6.81   6.81    0e+00
    10485760       2621440     int     max    1    1.512   6.93   6.93    0e+00    1.525   6.88   6.88    0e+00
    10485760       2621440     int     min    0    1.593   6.58   6.58    0e+00    1.622   6.47   6.47    0e+00
    10485760       2621440     int     min    1    1.587   6.61   6.61    0e+00    1.602   6.54   6.54    0e+00
```

NCCL RESULT

reduce (QPI limit)

```
[hryu@ivb121 single]$ ./reduce_test 10485760  2 0 4
# Using devices
#   Rank  0 uses device  0 [0x06] Tesla K80
#   Rank  1 uses device  4 [0x85] Tesla K80

#                                                     out-of-place                    in-place
#      bytes             N    type      op  root    time  algbw  busbw      res     time  algbw  busbw      res
    10485760      10485760    char     sum    0    3.521   2.98   2.98    0e+00    3.557   2.95   2.95    0e+00
    10485760      10485760    char     sum    1    2.749   3.81   3.81    0e+00    2.657   3.95   3.95    0e+00
    10485760      10485760    char    prod    0    3.898   2.69   2.69    0e+00    3.922   2.67   2.67    0e+00
    10485760      10485760    char    prod    1    3.367   3.11   3.11    0e+00    3.403   3.08   3.08    0e+00
    10485760      10485760    char     max    0    2.938   3.57   3.57    0e+00    2.966   3.54   3.54    0e+00
    10485760      10485760    char     max    1    2.641   3.97   3.97    0e+00    2.627   3.99   3.99    0e+00
    10485760      10485760    char     min    0    3.032   3.46   3.46    0e+00    3.060   3.43   3.43    0e+00
    10485760      10485760    char     min    1    2.529   4.15   4.15    0e+00    2.559   4.10   4.10    0e+00
    10485760       2621440     int     sum    0    3.235   3.24   3.24    0e+00    3.238   3.24   3.24    0e+00
    10485760       2621440     int     sum    1    2.645   3.96   3.96    0e+00    2.671   3.93   3.93    0e+00
    10485760       2621440     int    prod    0    5.034   2.08   2.08    0e+00    4.870   2.15   2.15    0e+00
    10485760       2621440     int    prod    1    3.955   2.65   2.65    0e+00    3.970   2.64   2.64    0e+00
    10485760       2621440     int     max    0    3.075   3.41   3.41    0e+00    3.120   3.36   3.36    0e+00
    10485760       2621440     int     max    1    2.705   3.88   3.88    0e+00    2.722   3.85   3.85    0e+00
    10485760       2621440     int     min    0    3.116   3.36   3.36    0e+00    3.138   3.34   3.34    0e+00
    10485760       2621440     int     min    1    2.737   3.83   3.83    0e+00    2.764   3.79   3.79    0e+00
```

NCCL RESULT

broadcast

```
[hryu@ivb121 single]$ ./broadcast_test 10485760  2 0 1
# Using devices
#   Rank  0 uses device  0 [0x06] Tesla K80
#   Rank  1 uses device  1 [0x07] Tesla K80

#      bytes             N    type  root     time  algbw  busbw    delta
    10485760      10485760    char     0    0.976  10.74  10.74    0e+00
    10485760      10485760    char     1    0.967  10.84  10.84    0e+00
    10485760       2621440     int     0    0.991  10.58  10.58    0e+00
    10485760       2621440     int     1    0.987  10.62  10.62    0e+00
    10485760       5242880    half     0    0.980  10.70  10.70    0e+00
    10485760       5242880    half     1    0.974  10.77  10.77    0e+00
    10485760       2621440   float     0    0.992  10.57  10.57    0e+00
    10485760       2621440   float     1    0.986  10.64  10.64    0e+00
    10485760       1310720  double     0    0.957  10.96  10.96    0e+00
    10485760       1310720  double     1    0.951  11.03  11.03    0e+00
    10485760       1310720   int64     0    0.959  10.93  10.93    0e+00
    10485760       1310720   int64     1    0.954  10.99  10.99    0e+00
    10485760       1310720  uint64     0    0.962  10.90  10.90    0e+00
    10485760       1310720  uint64     1    0.954  10.99  10.99    0e+00
```

DNN training Benchmark

DL ARITHMETIC

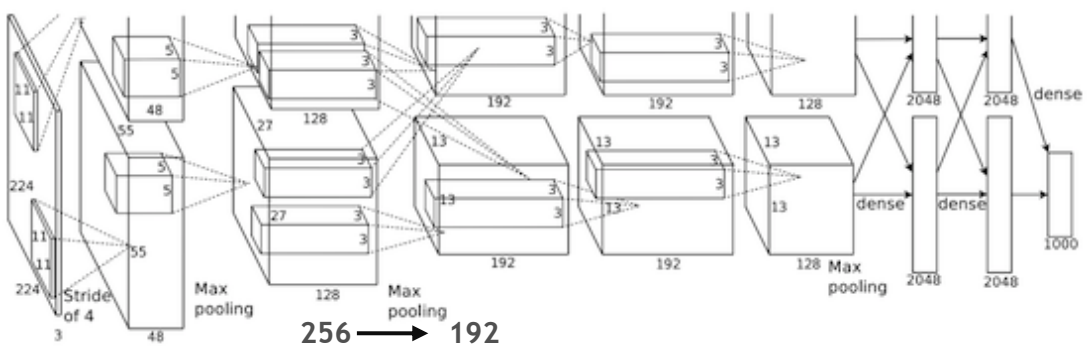
- FCN(ANN) → dot product
- Image : convolution
  - Convolution → Image Process with filter
  - → SGEMV, SGEMM
  - → FFT
- Speech : RNN(LSTM, GRU)
- NLP : cuSpase

TRAINING REQUIREMENT

- Large Batch is important to speed up & multiGPU (data parallel)
- multiGPU scale up for training
  - CV2, NV-CAFFE, NV-TORCH, kaldi
  - TF, CNTK, neon,
- Multi-node Scale up for training
  - CNTK, TF(internel),
- Multi Model(job)
  - DIGITS, TF serving
- AELXNET is still reference

ALEXNET BENCHMARK

- There are multiple version of ALEXNET
  - Alexnet-Original (original 2012, 2 path on 2 gpu cuda-convet)
  - Alexnet-caffe (caffe provide script, 1 gpu on caffe)
  - ALEXNET-OWT(One Weird Trick) (2014, reduce parameter on cuda-convet2)



96 → 64

96, 256, 384, 384, 256 Alexent-2012

64, 192, 384, 384, 256 Alexent-2014

CONVNET-BENCHMARK

https://github.com/soumith/convnet-benchmarks

Alexnet

Input 128x3x224x224

Library	Class	Time (ms)	forward (ms)	backward (ms)
CuDNN[R4]-fp16 (Torch)	cudnn.SpatialConvolution	71	25	46
Nervana-neon-fp16	ConvLayer	78	25	52
CuDNN[R4]-fp32 (Torch)	cudnn.SpatialConvolution	81	27	53
Nervana-neon-fp32	ConvLayer	87	28	58
fbfft (Torch)	fbnn.SpatialConvolution	104	31	72
TensorFlow	conv2d	151	34	117
Chainer	Convolution2D	177	40	136
cudaconvnet2*	ConvLayer	177	42	135
CuDNN[R2] *	cudnn.SpatialConvolution	231	70	161
Caffe (native)	ConvolutionLayer	324	121	203
Torch-7 (native)	SpatialConvolutionMM	342	132	210
CL-nn (Torch)	SpatialConvolutionMM	963	388	574
Caffe-CLGreenTea	ConvolutionLayer	1442	210	1232

CONVNET-BENCHMARK

https://github.com/soumith/convnet-benchmarks

VGG

Input 64x3x224x224

Library	Class	Time (ms)	Forward (ms)	Backward (ms)
Nervana-neon-fp16	ConvLayer	254	82	171
Nervana-neon-fp32	ConvLayer	320	103	217
CuDNN[R4]-fp16 (Torch)	cudnn.SpatialConvolution	471	140	331
CuDNN[R4]-fp32 (Torch)	cudnn.SpatialConvolution	529	162	366
Chainer	Convolution2D	885	251	632
TensorFlow	conv2d	982	191	791
fbfft (Torch)	SpatialConvolutionCuFFT	1092	355	737
cudaconvnet2*	ConvLayer	1229	408	821
CuDNN[R2] *	cudnn.SpatialConvolution	1099	342	757
Caffe	ConvolutionLayer	1068	323	745
Torch-7 (native)	SpatialConvolutionMM	1105	350	755
CL-nn (Torch)	SpatialConvolutionMM	3437	875	2562
Caffe-CLGreenTea	ConvolutionLayer	5620	988	4632

CONVNET-BENCHMARK

https://github.com/soumith/convnet-benchmarks

GoogleNet

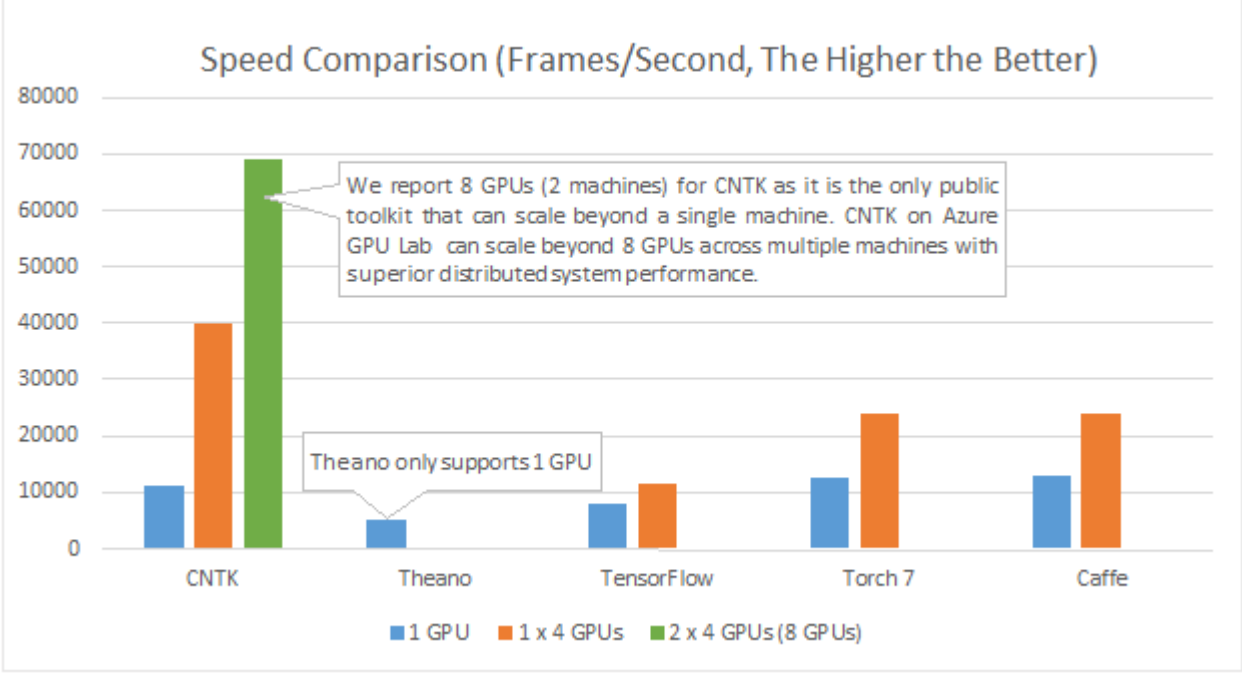
Input 64x3x224x224

Library	Class	Time (ms)	forward (ms)	backward (ms)
Nervana-neon-fp16	ConvLayer	230	72	157
Nervana-neon-fp32	ConvLayer	270	84	186
CuDNN[R4]-fp16 (Torch)	cudnn.SpatialConvolution	462	112	349
CuDNN[R4]-fp32 (Torch)	cudnn.SpatialConvolution	470	130	340
Chainer	Convolution2D	687	189	497
TensorFlow	conv2d	905	187	718
Caffe	ConvolutionLayer	1935	786	1148
CL-nn (Torch)	SpatialConvolutionMM	7016	3027	3988
Caffe-CLGreenTea	ConvolutionLayer	9462	746	8716

CNTK BENCHMARK

https://github.com/Alexey-Kamenev/Benchmarks

8196 minibatch **Input 8196x3x224x224**



CTC(SPEECH) BENCHMARK

https://github.com/baidu-research/warp-ctc

Maxwell GPU

T=150, L=40, A=28	warp-ctc	Eesen	Theano
N=1	3.1 ms	.5 ms	67 ms
N=16	3.2 ms	6 ms	94 ms
N=32	3.2 ms	12 ms	119 ms
N=64	3.3 ms	24 ms	153 ms
N=128	3.5 ms	49 ms	231 ms

T=150, L=20, A=5000	warp-ctc	Eesen	Theano
N=1	7 ms	40 ms	120 ms
N=16	9 ms	619 ms	385 ms
N=32	11 ms	1238 ms	665 ms
N=64	16 ms	2475 ms	1100 ms
N=128	23 ms	4950 ms	2100 ms

Maxwell CPU

T=150, L=40, A=28	warp-ctc	Stanford -CTC	Theano
N=1	2.6 ms	13 ms	15 ms
N=16	3.4 ms	208 ms	180 ms
N=32	3.9 ms	416 ms	375 ms
N=64	6.6 ms	832 ms	700 ms
N=128	12.2 ms	1684 ms	1340 ms

T=150, L=20, A=5000	warp-ctc	Stanford-CTC	Theano
N=1	21 ms	31 ms	850 ms
N=16	37 ms	496 ms	10800 ms
N=32	54 ms	992 ms	22000 ms
N=64	101 ms	1984 ms	42000 ms
N=128	184 ms	3968 ms	86000 ms

DNN inference Benchmark

INFERENCE REQUIREMENT

- Don’t need Large Batch
  - Just check 1 image or multiple images batch 1,2,4,8,16,32,64
- Perf/W is also important

Deep Learning S/W development

감사합니다.