GPU History
and
GPU Architecture

Overview

1. Processing Unit Measurements
   - Latency
   - Throughput
     - Flops

2. Data Parallelism
   - SISD
   - SIMD
   - SPMD
   - MPMD

Overview

3. From CPU to GPU
   - https://www.youtube.com/watch?v=6N_sJKw8sqw&feature=youtu.be
4. GPU History
   - https://www.youtube.com/watch?v=6N_sJKw8sqw&feature=youtu.be
5. NVIDIA and GPGPU
6. CUDA

Overview

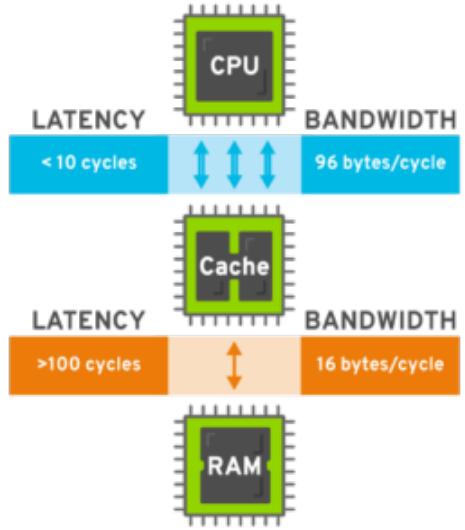
4. NVIDA GPU Models (Architectures or codename)
   - http://haanjack.github.io/cuda/2016/03/31/cuda-processor.html
   - http://donghyun53.net/nvidia-gpu-%EC%95%84%ED%82%A4%ED%85%8D%EC%B2%98-%EB%B3%80%EC%B2%9C%EC%82%AC-%EC%83%81%ED%8E%B8/
   - Tesla (2008)
   - Fermi (2010)
   - Kepler (2012)
   - MaxWell(2014)
   - Pascal (2016)
   - Volta (2018)
   - Turing (2018)
   - Ampere (2020)

Overview

8. Future of GPU
    - TSMC
    - SAMSUNG
    - NVIDIA
    - ARM

1. PROCESSING UNIT MEASUREMENTS

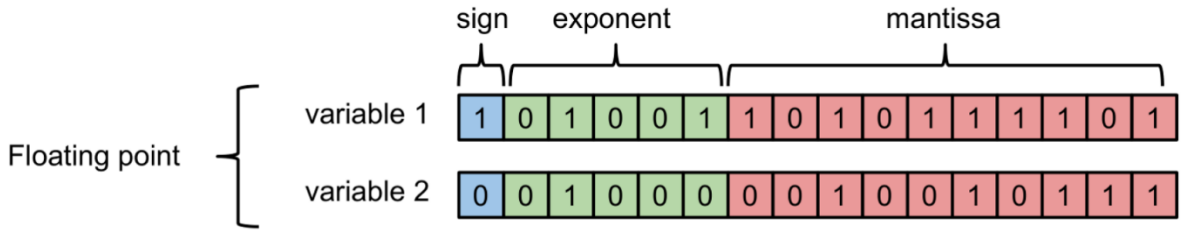
1.Latency (Time)
- The time required to perform some action(request)
- Latency unit(단위): ms (or ns)
- CPU

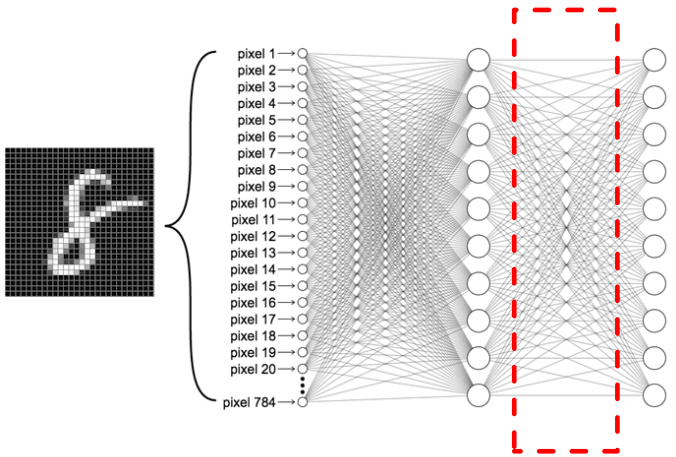


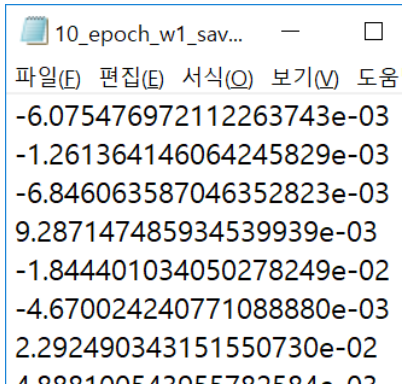
2.Throughput
- The number of actions executed per unit of time
- Throughput unit: MB/s, GB/s, Mbps, FLOPS, …..
- GPU

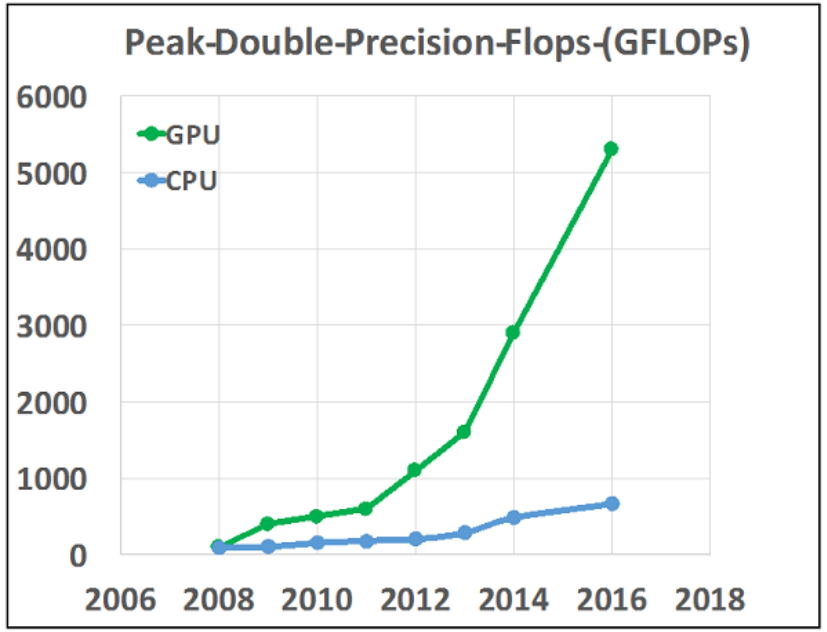
1. PROCESSING UNIT MEASUREMENTS

3. FLOPS(Float point Operations Per Second)



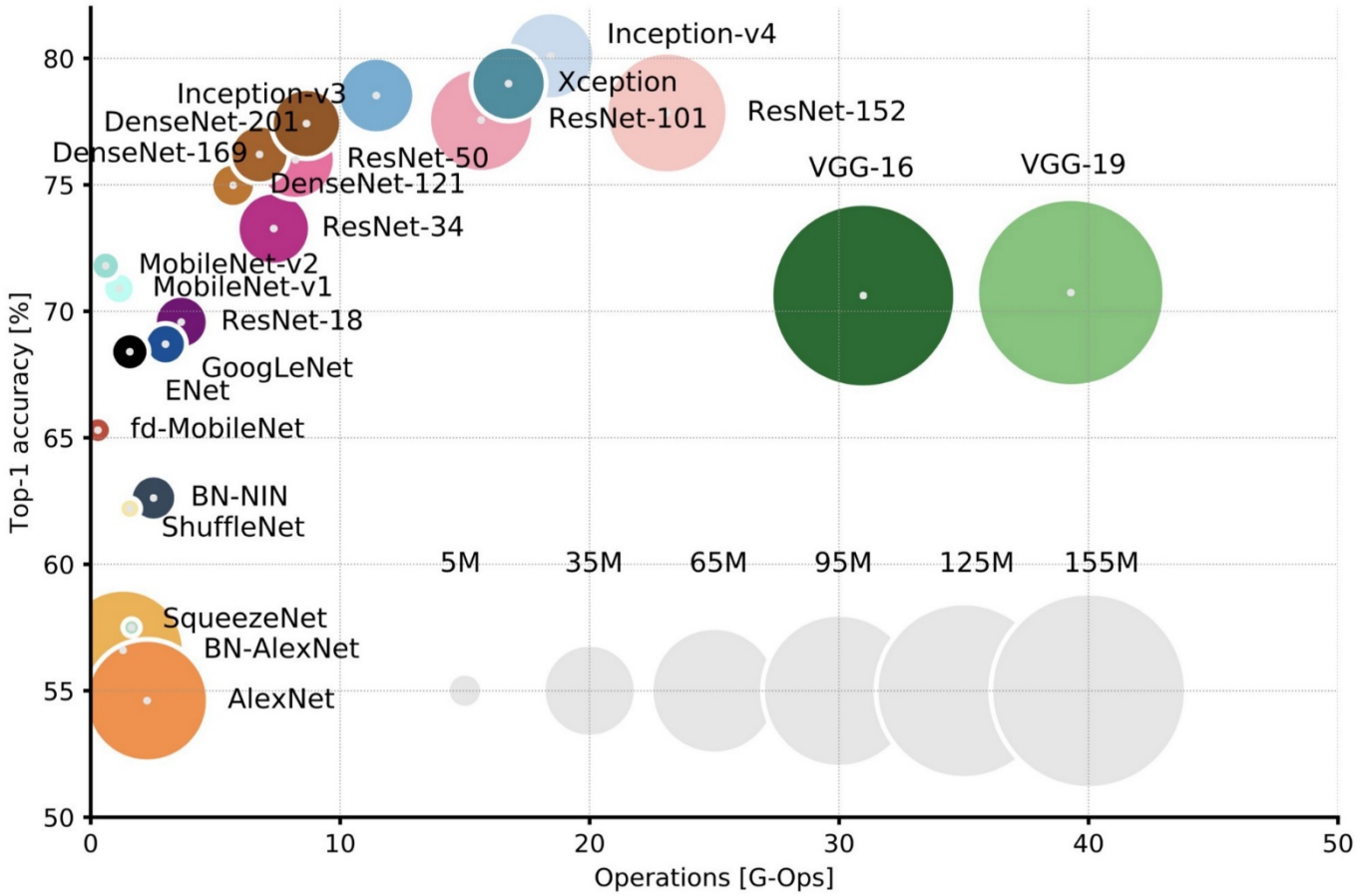






<MNIST 10 epoches second layer weights>

1. PROCESSING UNIT MEASUREMENTS



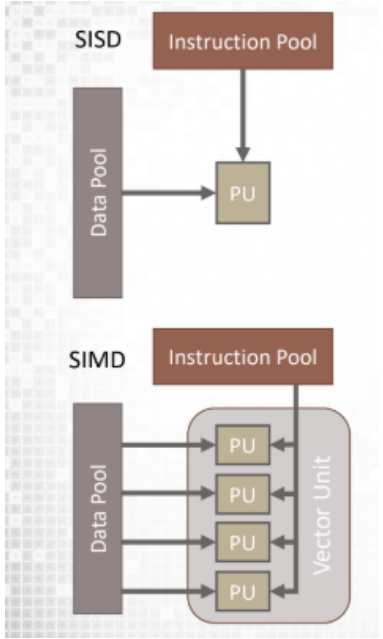
https://arxiv.org/pdf/1810.00736.pdf

2.DATA PARALLELISM

- We can classify an architecture based on instructions and data (Flynn's Taxonomy)
  - Instructions
    - Single Instruction (SI)
    - Multiple Instruction (MI)
    - Single Program (SP)
    - Multiple Program (MP)
  - Data
    - Single Data (SD)
    - Multiple Data (MD)

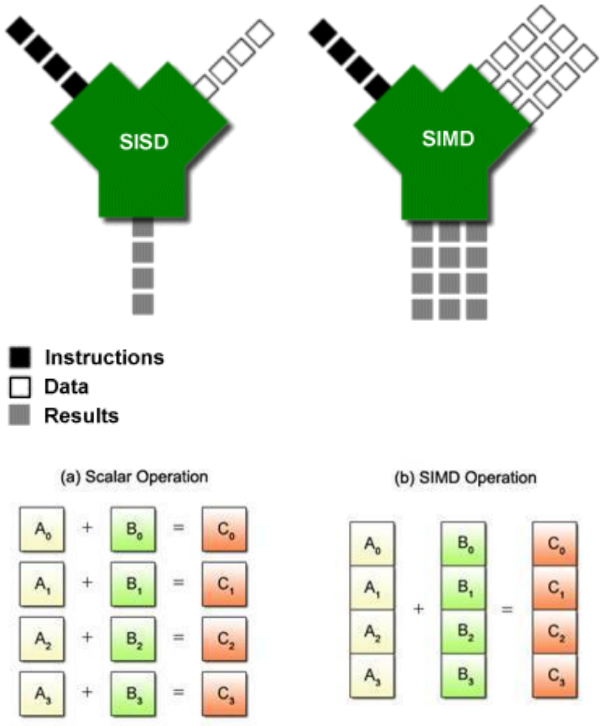
https://en.wikipedia.org/wiki/Flynn%27s_taxonomy

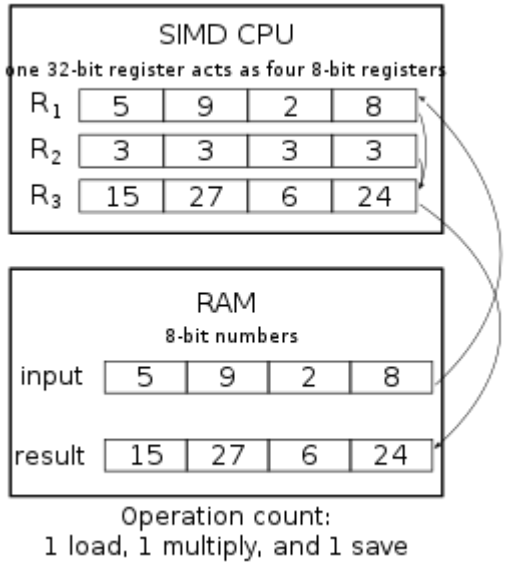
2.DATA PARALLELISM



※PU: Processing Unit

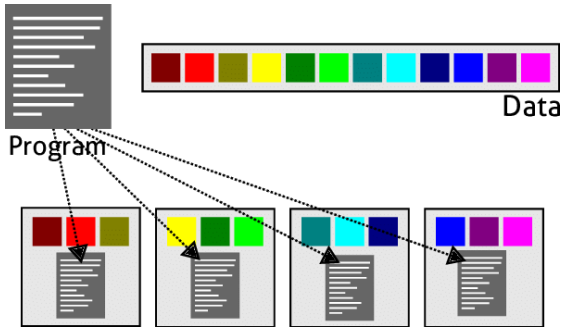
- SISD
  - Single Instruction Single Data
  - Before Pentium CPU, SISD
- SIMD
  - Single Instruction Multiple Data
  - Multiple processing elements performing the same operation simultaneously
  - Modern CPUs have SIMD instructions





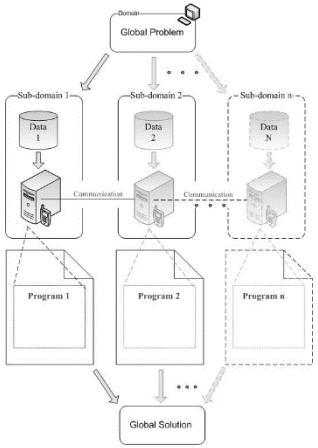
Operation count:
1 load, 1 multiply, and 1 save

2.DATA PARALLELISM



<SPMD>

- SPMD
  - Single Program Multiple Data
  - Multiple autonomous processors simultaneously executing a program on different data
  - Program execution can have an independent path for each data point.



<MPMD>

- MPMD
  - Multiple Program Multiple Data
  - Multiple autonomous processors simultaneously executing at least two independent programs.
  - Typically client & host programming models fit this description

2.DATA PARALLELISM

[SPMD – MNIST example]

```
(x_train, t_train), (x_test, t_test) = load_mnist(normalize=True, one_hot_label=True)
```

x_train=

784

[[0,0,0, ……..,0,0],
[0,0,0, ……..,0,0],
.
.
[0,0,0, ……,0,0]]

이미지 정보

60000

```
x_batch = x_train[batch_mask]
```

```
grad = network.gradient(x_batch, t_batch)
```

GPU

```
Epoch: 0028 cost = 0.008190597
Epoch: 0029 cost = 0.009955806
Epoch: 0030 cost = 0.007922430
Learning Finished!
Accuracy: 0.9798
Label:  [4]
Prediction:  [4]
Completed in  40.17236089706421
```

CPU

```
Epoch: 0028 cost = 0.010043771
Epoch: 0029 cost = 0.008661602
Epoch: 0030 cost = 0.008352582
Learning Finished!
Accuracy: 0.9832
Label:  [0]
Prediction:  [0]
Completed in  263.5923902988434
```

2.DATA PARALLELISM

※ Facebook이 아래와 같은 data parallelism 구조를 기반으로 ImageNet 학습을 1시간만에 끝냈다고 함

Accurate, Large Minibatch SGD:
Training ImageNet in 1 Hour

Priya Goyal Piotr Dollár Ross Girshick Pieter Noordhuis
Lukasz Wesolowski Aapo Kyrola Andrew Tulloch Yangqing Jia Kaiming He

Facebook

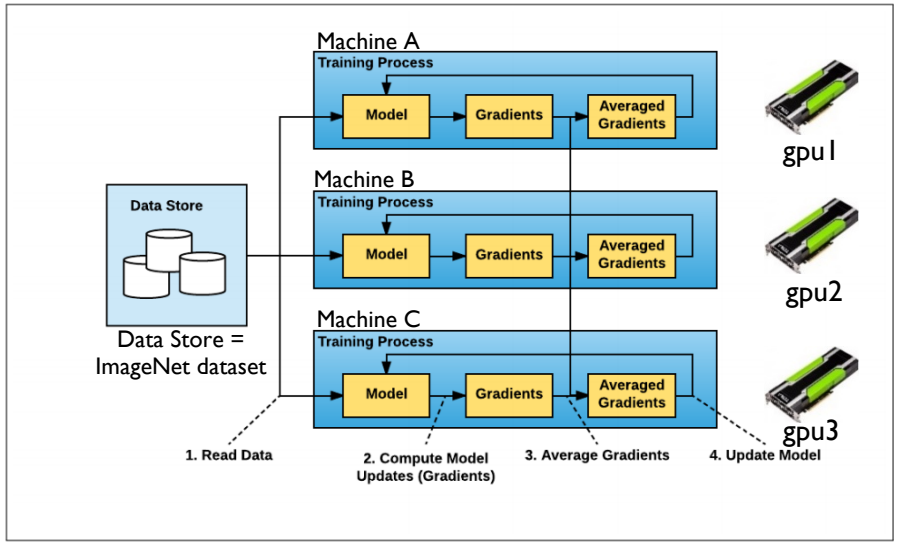


Figure 2: The "data parallel" approach to distributed training involves splitting up the data and training on multiple nodes in parallel. In synchronous cases, the gradients for different batches of data are calculated separately on each node but averaged across nodes to apply consistent updates to the model copy in each node.

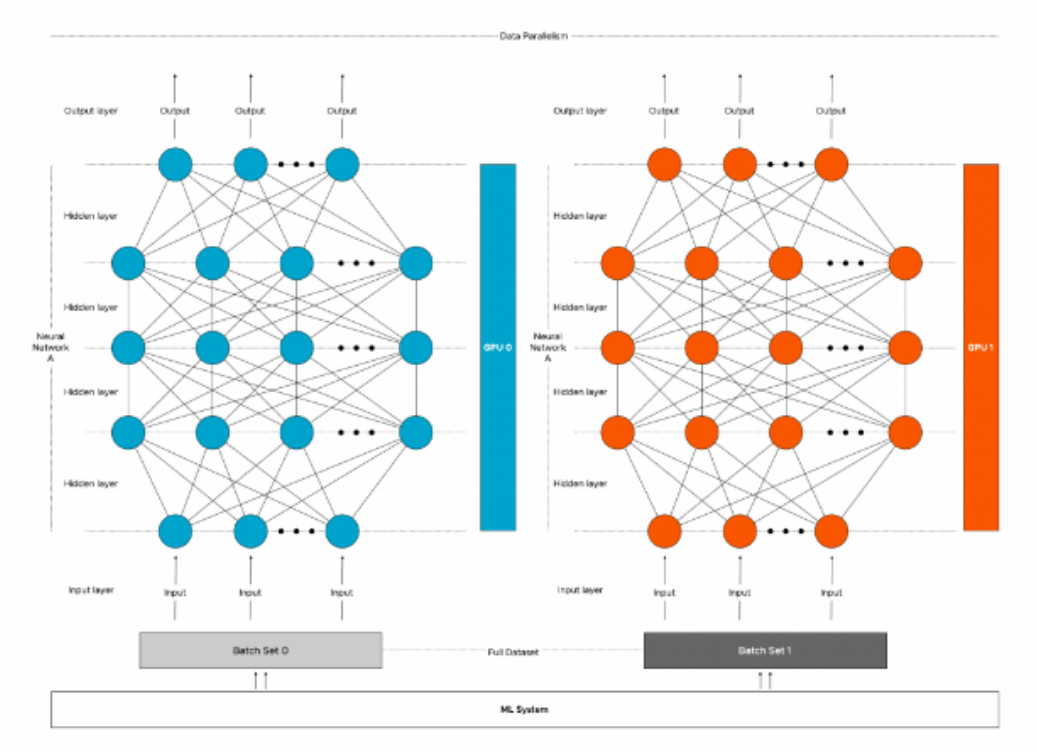
1. 딥러닝 모델 (ex: ResNet50)을 복제해서 각 machine에 올려준다.
2. 데이터 3등분 분할 후 각각의 machin에 있는 모델에 할당해준다.
3. 각 mahine에 모델에서 training 과정을 진행하고 gradient descen로 업데이트 하는 것이 아니라, 각각의 machine에서 얻은 gradient 값들을 합산하여 평균을 낸다.

$$\frac{\theta_j := \theta_j - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} + \theta_j := \theta_j - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} + \theta_j := \theta_j - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}}{3}$$

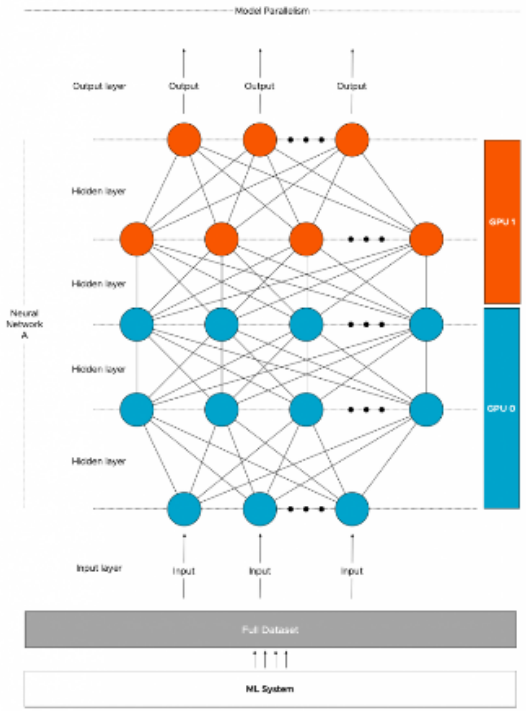
4. 각각의 machine에서 평균 값들을 이용해 backpropagation을 하여 parameter를 update하게 된다.

2.DATA PARALLELISM

Data parallelism



Model parallelism

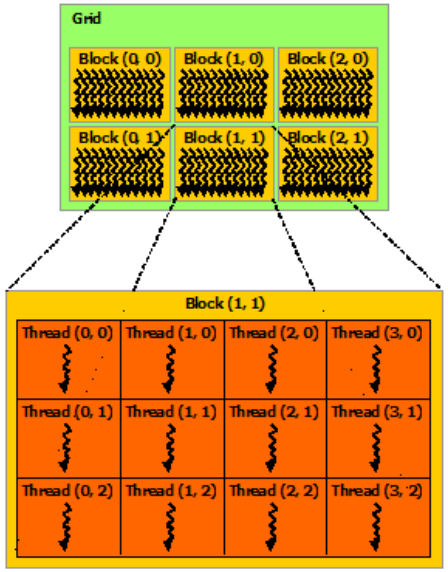


"Maybe model sequentiality would be a better name for this mode as it primarily is using devices in sequential order. More than often a device is idling, waiting to receive the data from another device"

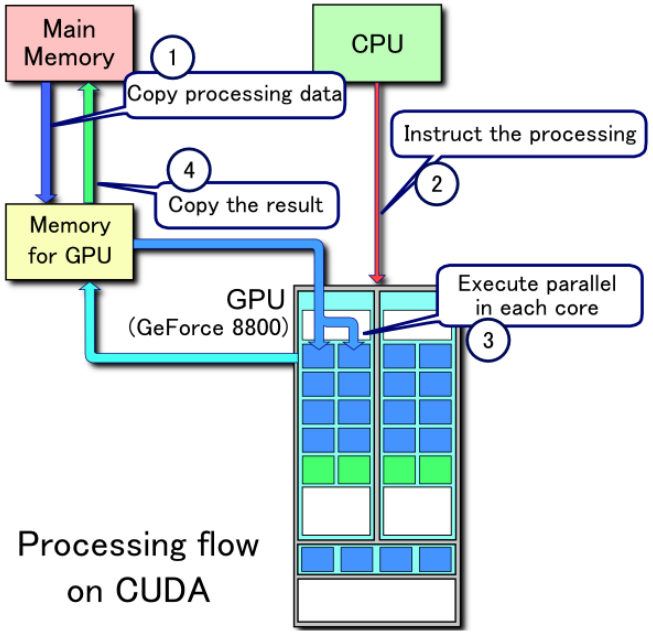
https://frankdenneman.nl/2020/02/19/multi-gpu-and-distributed-deep-learning/

2.DATA PARALLELISM

- What taxonomy best describes data parallelism with a GPU?
  - Obvious Answer: SIMD
  - Less Obvious answer: SPMD
  - Slightly confusing answer: SIMT (Single Instruction Multiple Threading)



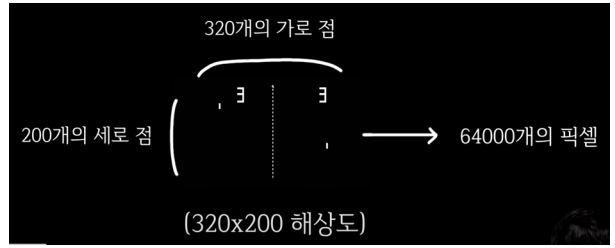
- Grid: GPU
- Block: SM (Streaming Multi-processor)
- Thread: core



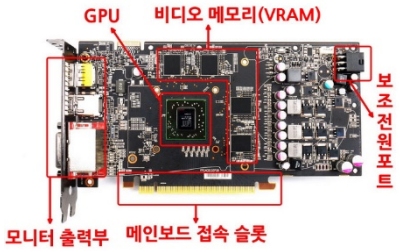
3. FROM CPU TO GPU

- DOS
- CPU

Resolution: 320x200 =



중요하지 않은 일



<GPU board>

+

<GPU cooler>

=

<Dedicated graphics card>

4. GPU HISTORY

<1993-Intel Pentium>

- Moore's laws
- IC (integrated Circuit)
- 1993
- Intel Pentium

<1993-NVIDIA>

- 1993
- Jensen Huang
- NVIDIA

<1996 -Voodoo>

- 1996
- 3dfx
- Voodoo

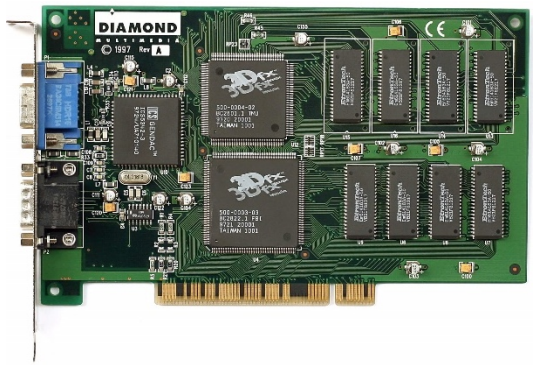
- 1996
- Voodoo
- Quake – First 3d shooting game

4. GPU HISTORY

<2D game>

<1998 Starcraft – 3D game>

4. GPU HISTORY



Voodoo Reference Card

$200

Modified Voodoo Card

$425

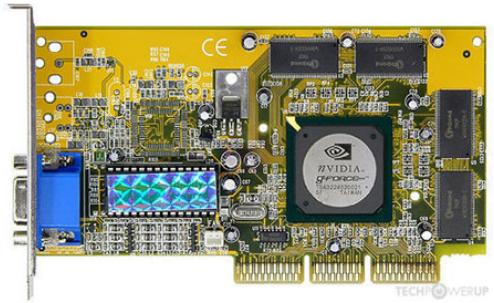
4. GPU HISTORY

NVIDIA GEFORCE GTX

VS

AMD RADEON

Radeon FreeSync | Radeon Overlay | Radeon Chill | Radeon Wattman | Radeon ReLive | AMD's Eyefinity



- 1999
- First Geforce model
- Geforce 2 Series

- 2002
- 3dfx 3D technology
- NVIDIA reference card
- Geforce 4 Series

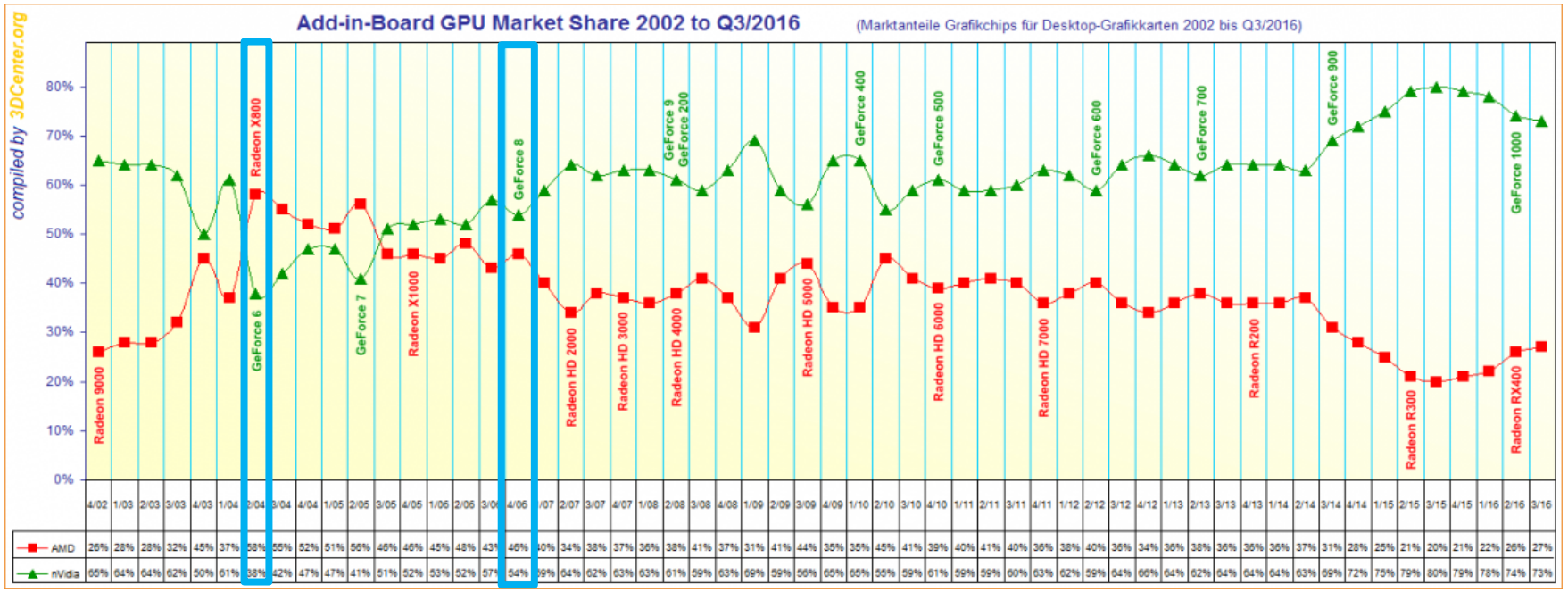
- 2004
- Geforce 6 Series
- Too much Power: 400W

- 2004
- ATI Radeon X800
- Power: 240W

4. GPU HISTORY

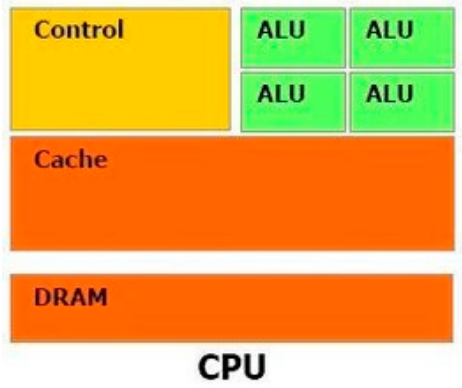
- 2005
- Geforce 7 series

- 2006
- Geforce 8 series

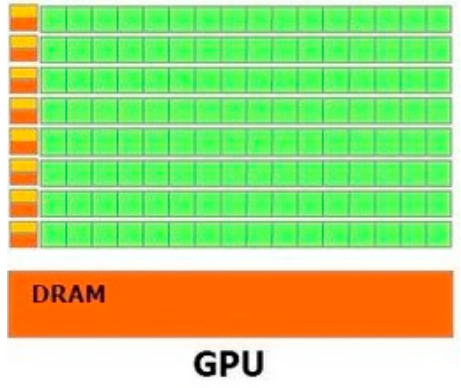


<GPU market share from 2002 to 2016 3/4>

5. NVIDA AND GPGPU



[CPU – Matrix multiplication]



[GPU가 행렬을 연산 할 때]

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

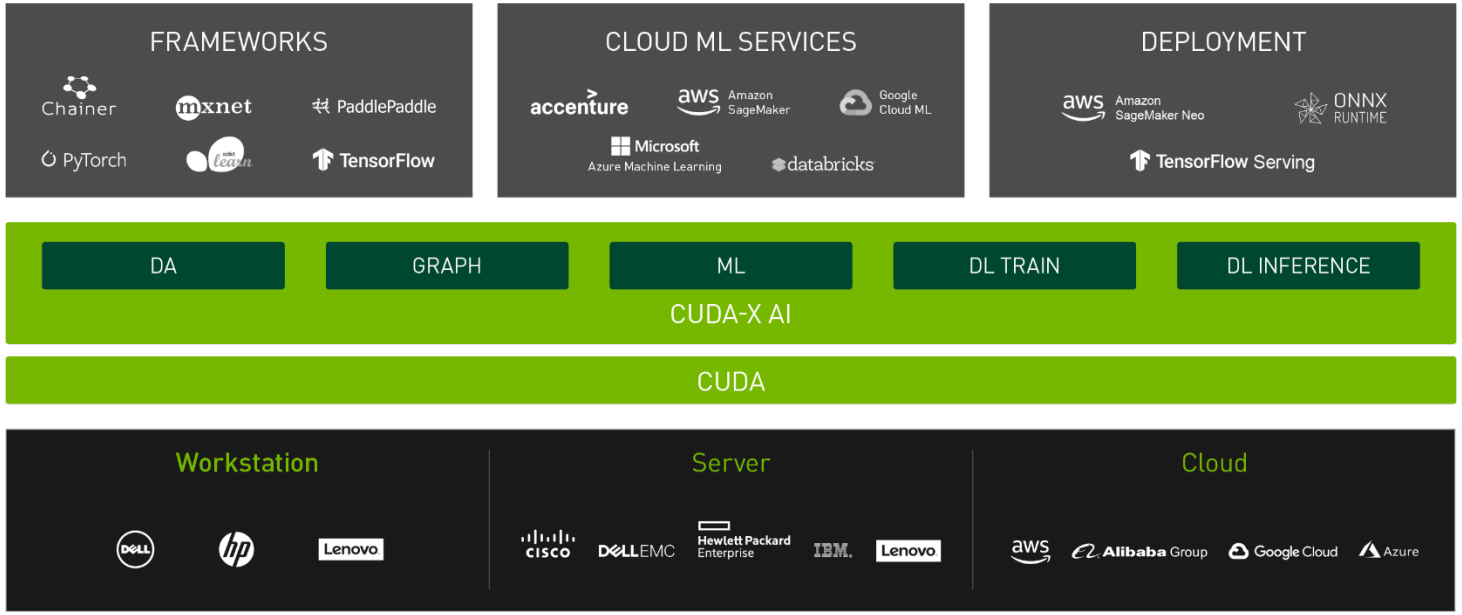
$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} (1*5) + (2*7) & (1*6) + (2*8) \\ (3*5) + (4*7) & (3*6) + (4*8) \end{bmatrix}$$

GPGPU

General Purpose computation on Graphic Processor Units

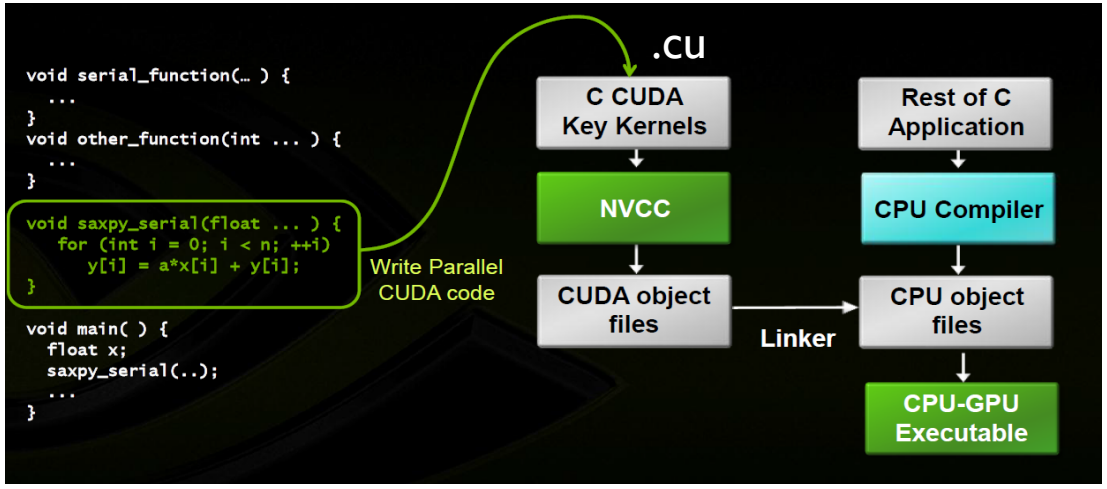
6. CUDA

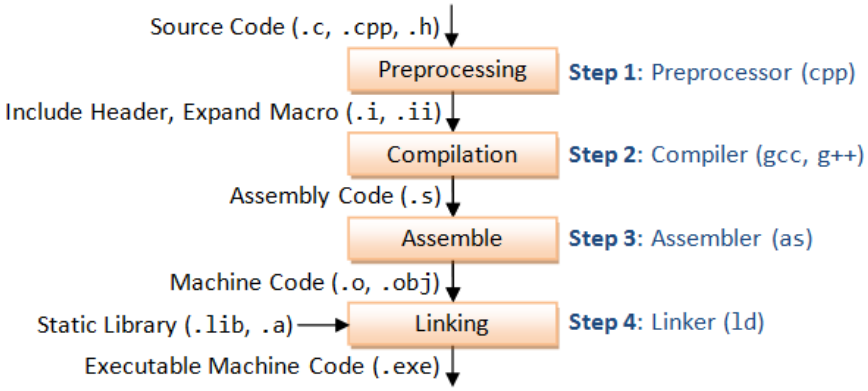

NVIDIA
CUDA

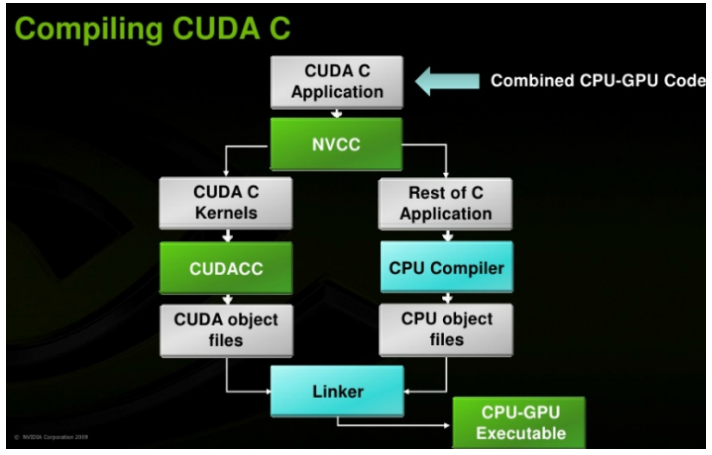

FRAMEWORKS
Chainer
mxnet
PaddlePaddle
PyTorch
learn
TensorFlow
CLOUD ML SERVICES
accenture
aws Amazon SageMaker
Google Cloud ML
Microsoft Azure Machine Learning
databricks
DEPLOYMENT
aws Amazon SageMaker Neo
ONNX RUNTIME
TensorFlow Serving
DA
GRAPH
ML
DL TRAIN
DL INFERENCE
CUDA-X AI
CUDA
Workstation
DELL
hp
Lenovo
Server
cisco
DELL EMC
Hewlett Packard Enterprise
IBM
Lenovo
Cloud
aws
Alibaba Group
Google Cloud
Azure

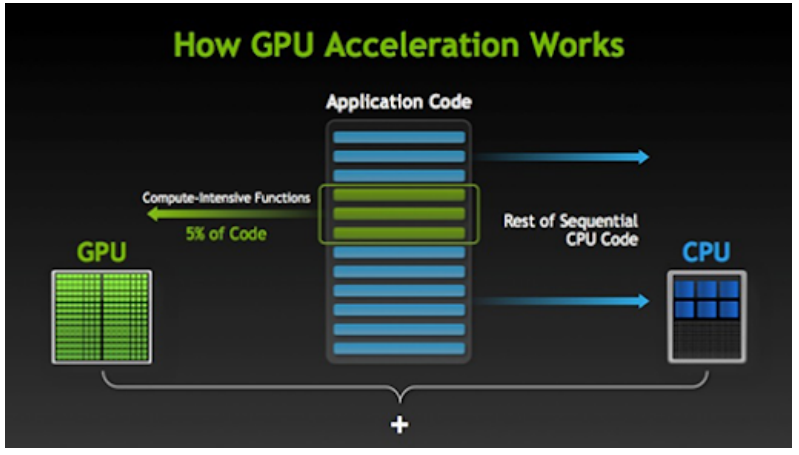
6. CUDA

※NVCC: Nvidia Cuda Compiler

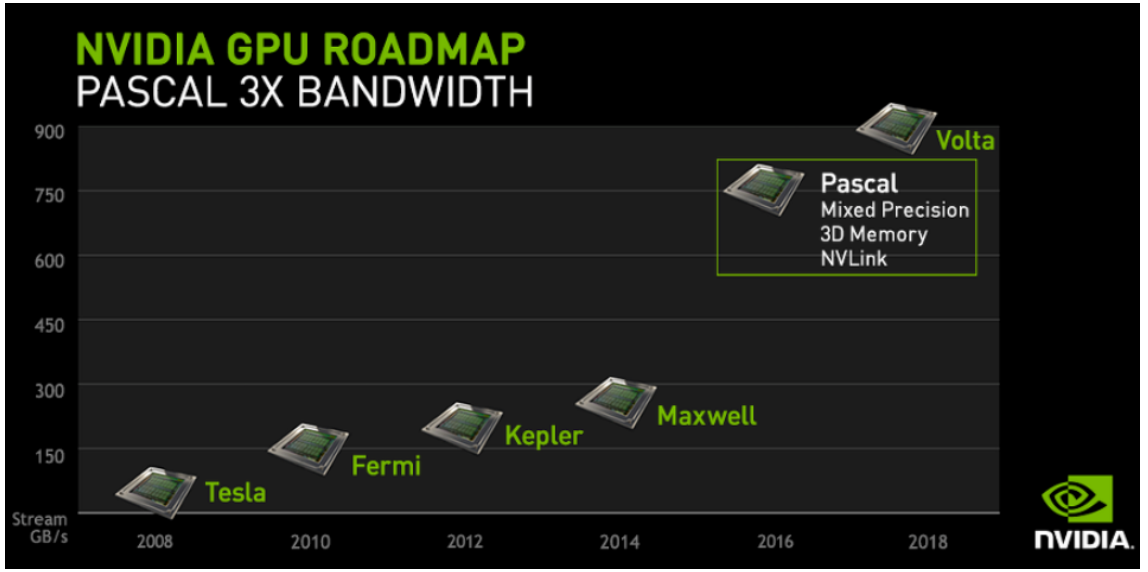




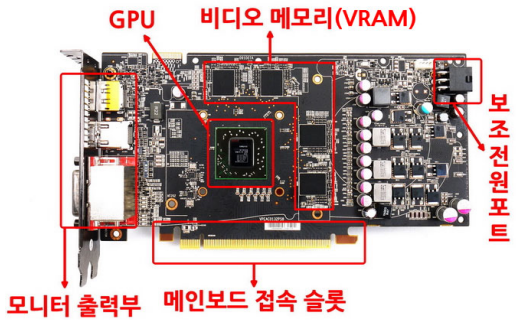




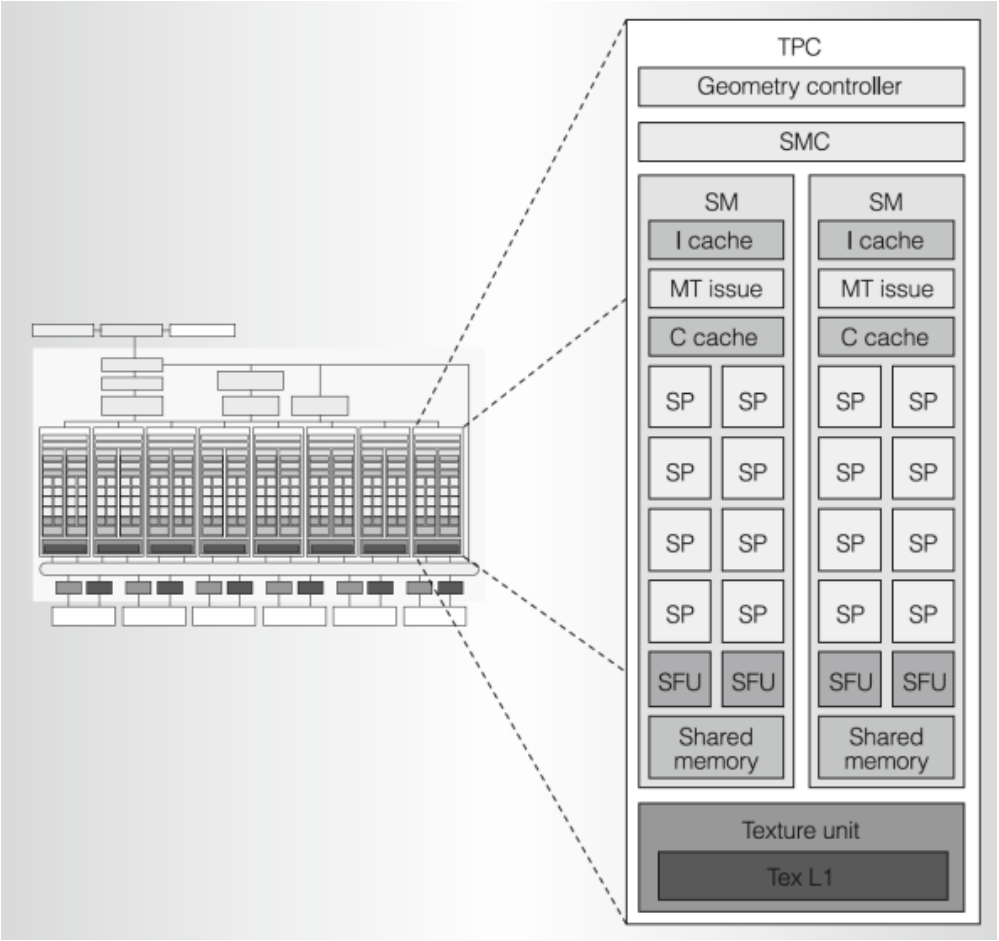
7. NVIDA GPU MODELS (ARCHITECTURES)



- Geforce (GTX)
  - Gaming
  - GPGPU
- Quadro
  - Graphic

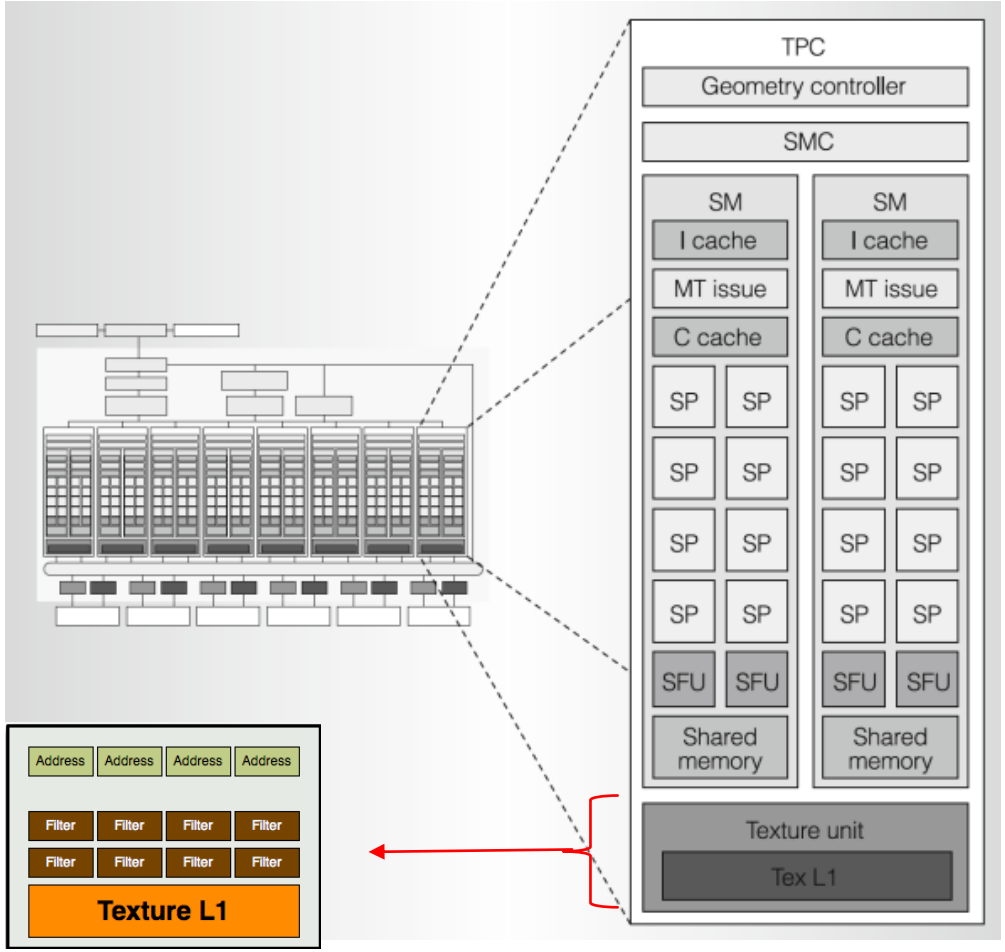


7-1. NVIDA GPU MODELS (TESLA)



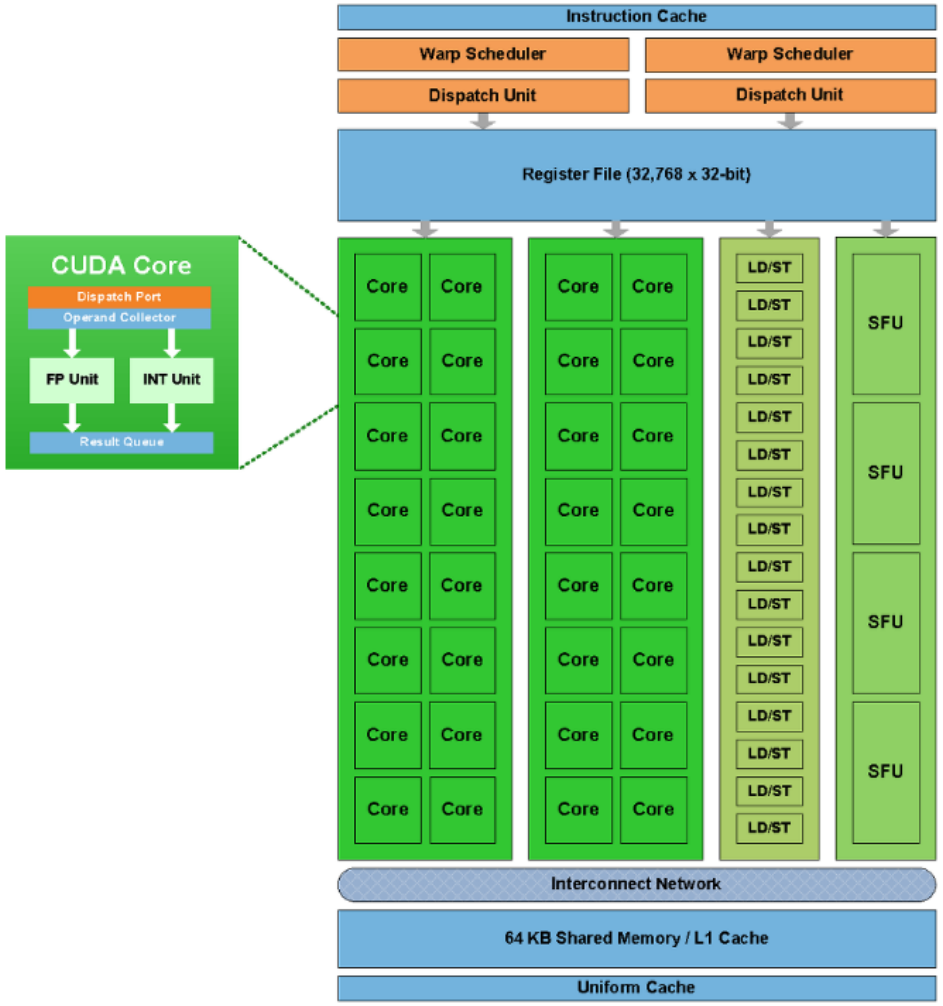
- Geforce 8, 9, 200 (2006, 2008, 2008)
- Geometry controller
- SMC (SM controller)
- SM (Streaming Multiprocessor)
  - Multiple vector "CUDA" cores
  - SP (Stream Processor)
    - ALU
  - SFU (Special Function Unit)
    - Transcendental function
    - 4 Floating Point (FP) units (multipliers)
  - Shared memory
    - Thread
    - SIMT (Single Instruction Multi-Threading)

7-1. NVIDA GPU MODELS (TESLA)



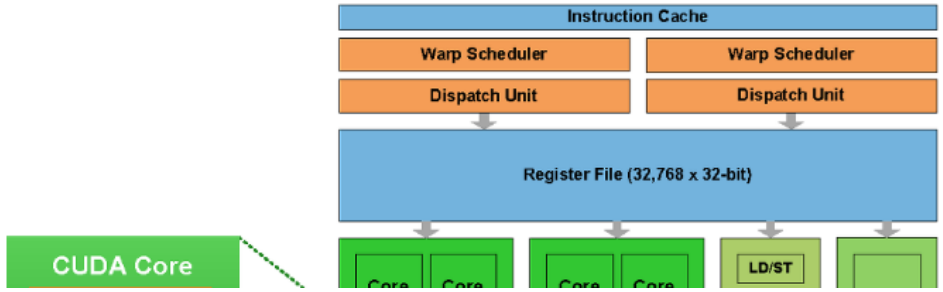
- Texture unit
  - Apply texture to the pixels
  - Texture mapping unit
    - Texture Address Units (TA)
      - Texel
    - Texture Filtering (TF)

7-2. NVIDA GPU MODELS (FERMI)



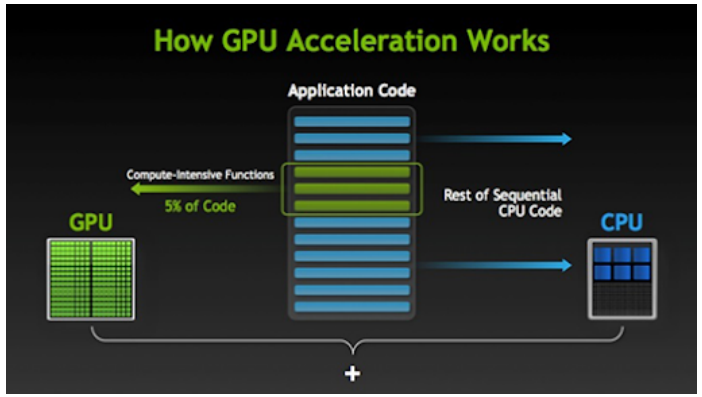
- Geforce 400, 500 (2010, 2010)
  - Control Units
    - Instruction Cache
    - Warp Scheduler
    - Dispatch
  - Processing Units
    - SP (CUDA Core)
    - SFU
  - Memory Units
    - Register File
    - Shared Memory (L1 Cache)

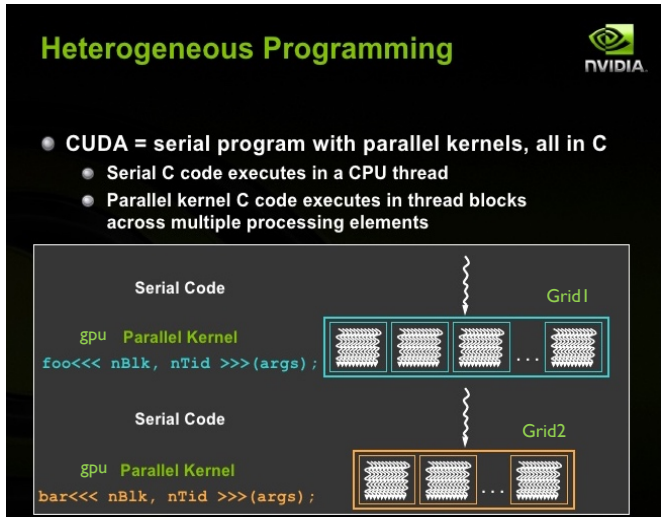
7-2. NVIDA GPU MODELS (FERMI)

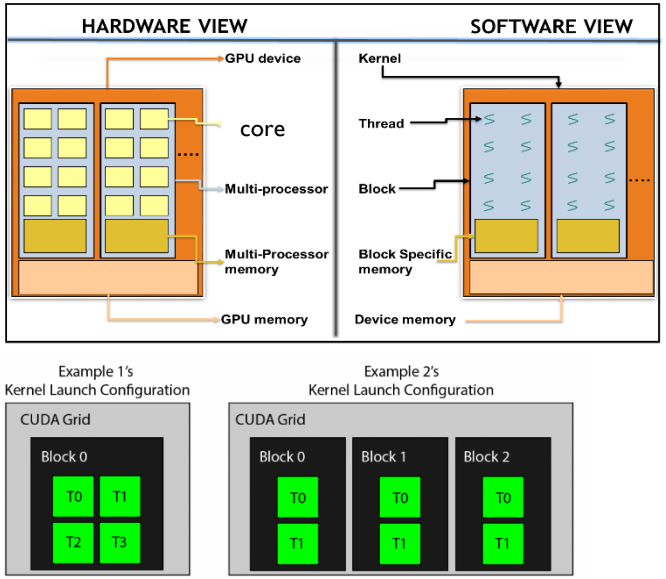


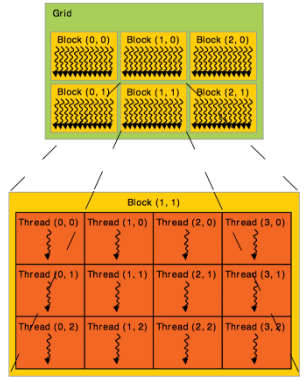
[Control Units]

- Instruction Cache
- Warp Scheduler
- Dispatch

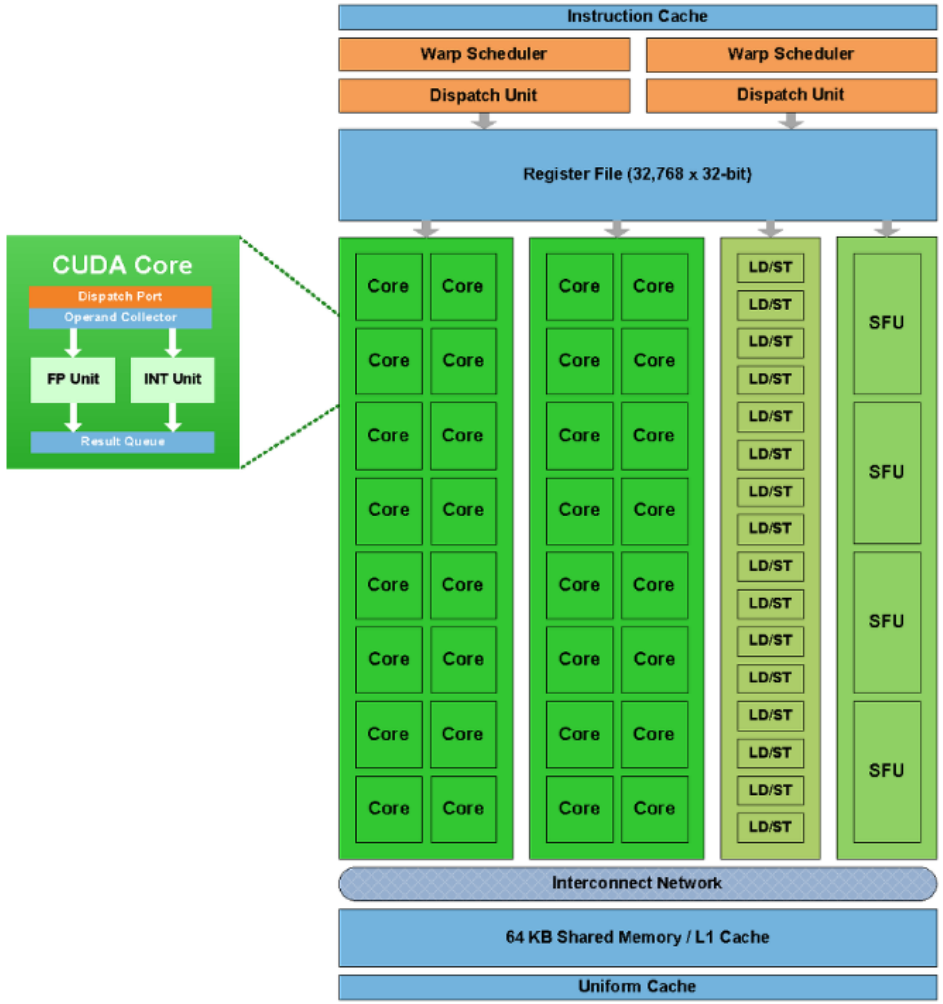








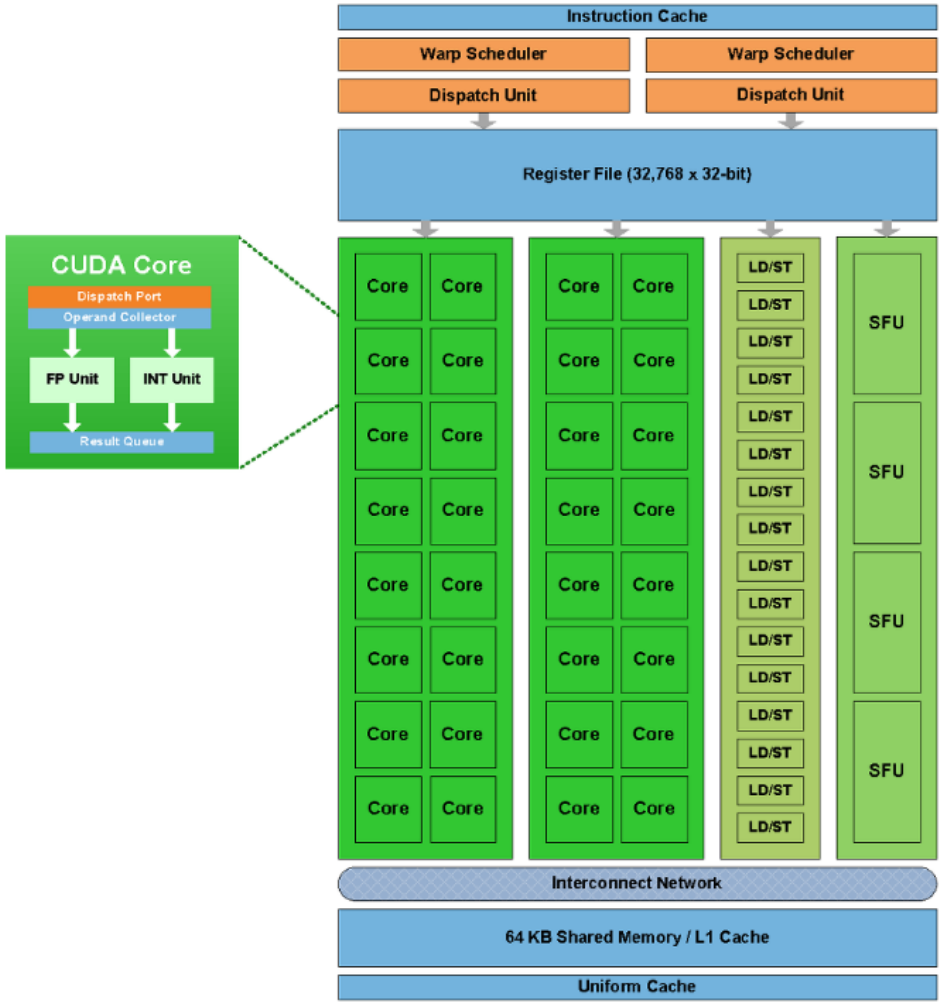
7-2. NVIDA GPU MODELS (FERMI)



[Processing Units]

- SM
  - CUDA core (SP)
    - 32 SP = 4 X Tesla SP
  - 4 SPU
  - 16 LD/ST
    - https://docs.nvidia.com/gameworks/content/developertools/desktop/analysis/report/cudaexperiments/kernellevel/pipeutilization.htm

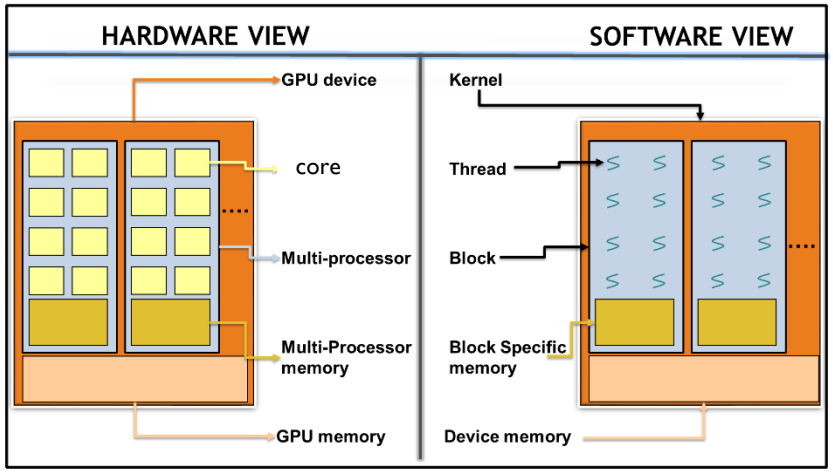
7-2. NVIDA GPU MODELS (FERMI)

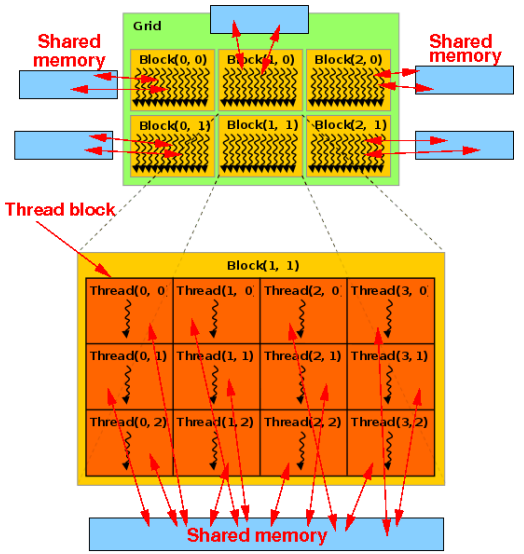


[Memory Units]

- Register File
  - Set of registers for CUDA cores
  - Flexibility

- Shared Memory (L1 Cache)
  - Sharing CUDA block





7-2. NVIDA GPU MODELS (FERMI WITH ALEXNET)

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

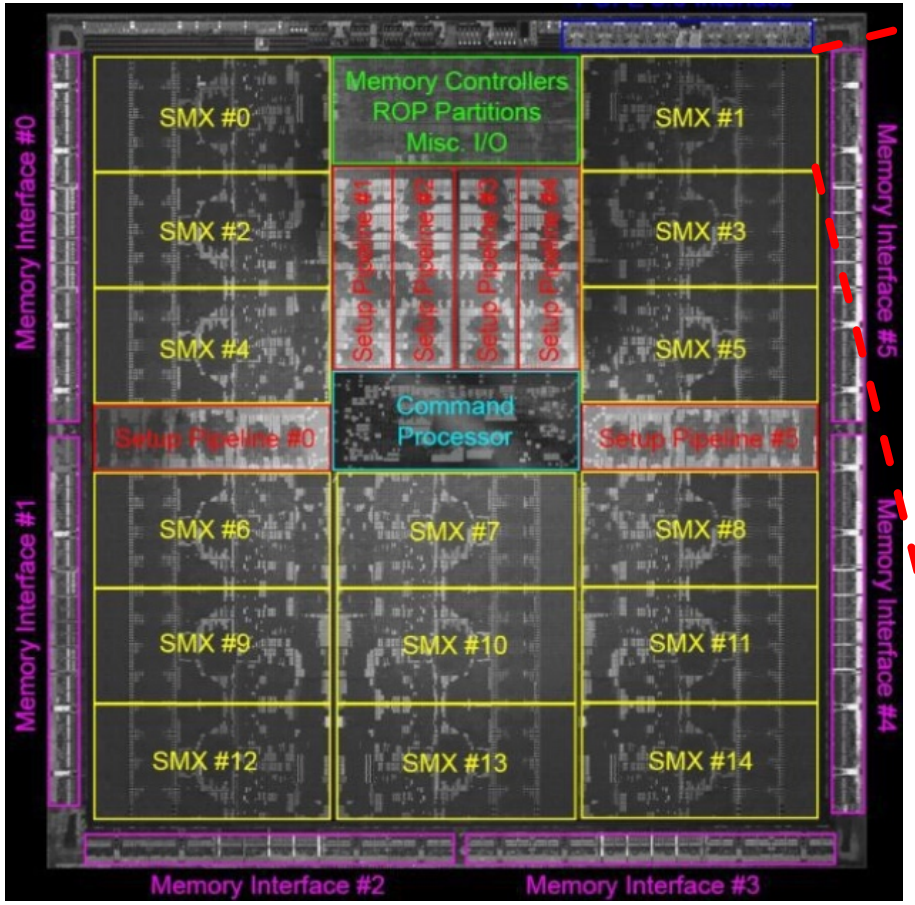
Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

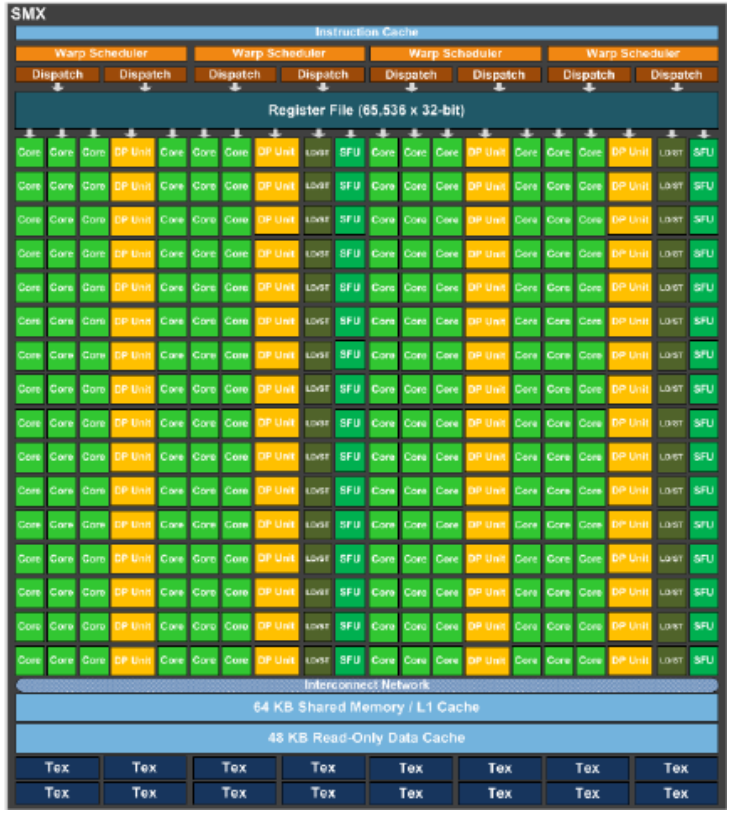
3.2 Training on Multiple GPUs

A single GTX 580 GPU has only 3GB of memory, which limits the maximum size of the networks that can be trained on it. It turns out that 1.2 million training examples are enough to train networks which are too big to fit on one GPU. Therefore we spread the net across two GPUs. Current GPUs

그래픽 카드 모델명	GPU				그래픽 메모리				TDP (W)	출고 가격 ($)
	이름 (공정) (면적)	CUDA:TMU:ROP (RE, PE)	클럭 (코어) (셰이더) (MHz)	L2 캐시 (KB)	버스 (bit)	규격	클럭 (전송률) (MHz) (Mbps)	용량 (GB)		
데스크탑용 제품군										
GTX 580		512:64:48 (4, 16)	772 (1544)	768	384		1002 (4008)	1.5 3	244	499

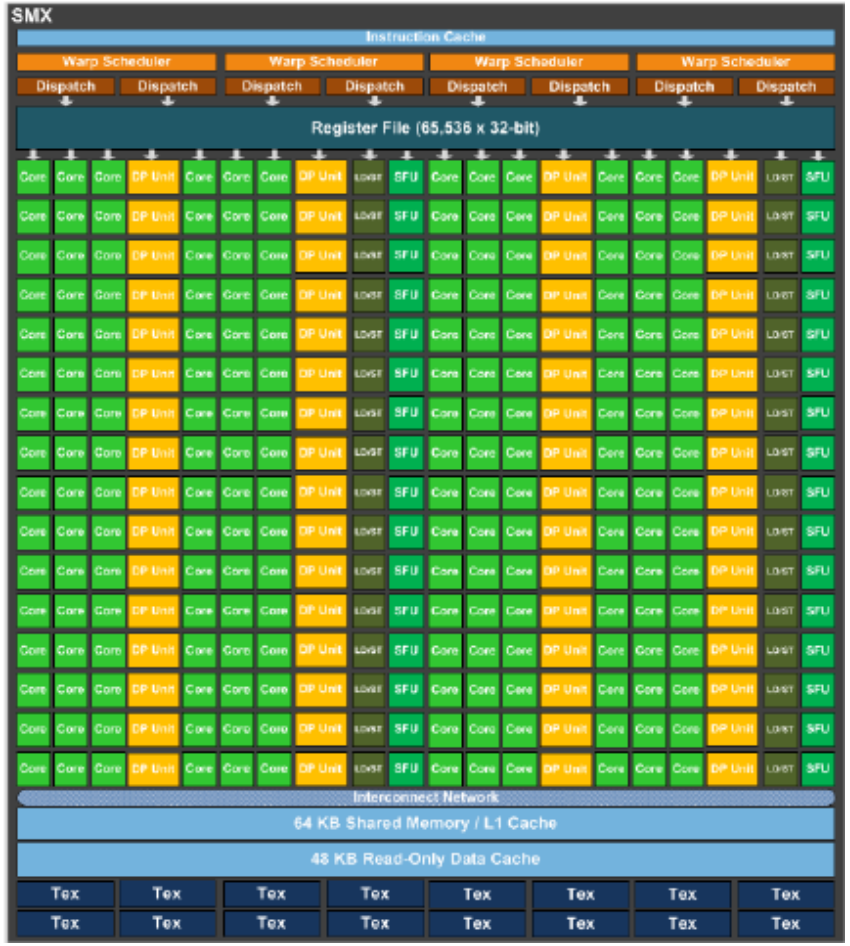
7-3. NVIDA GPU MODELS (KEPLER)





- Geforce 600, 700, TITAN (2012, 2013)
- SM → SMX
- Same speed

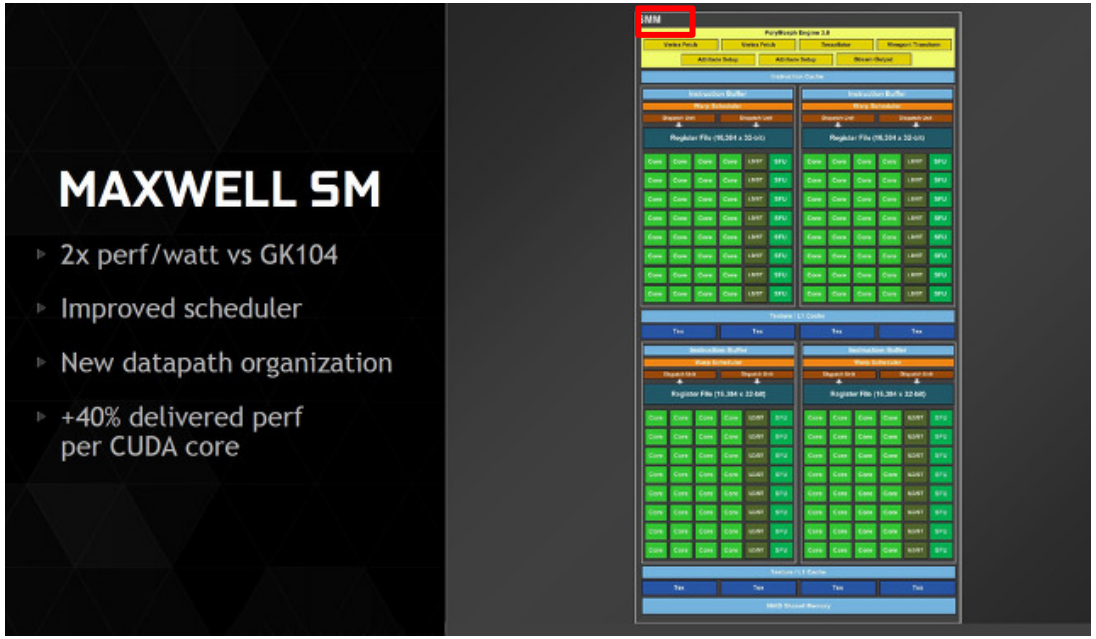
7-3. NVIDA GPU MODELS (KEPLER)

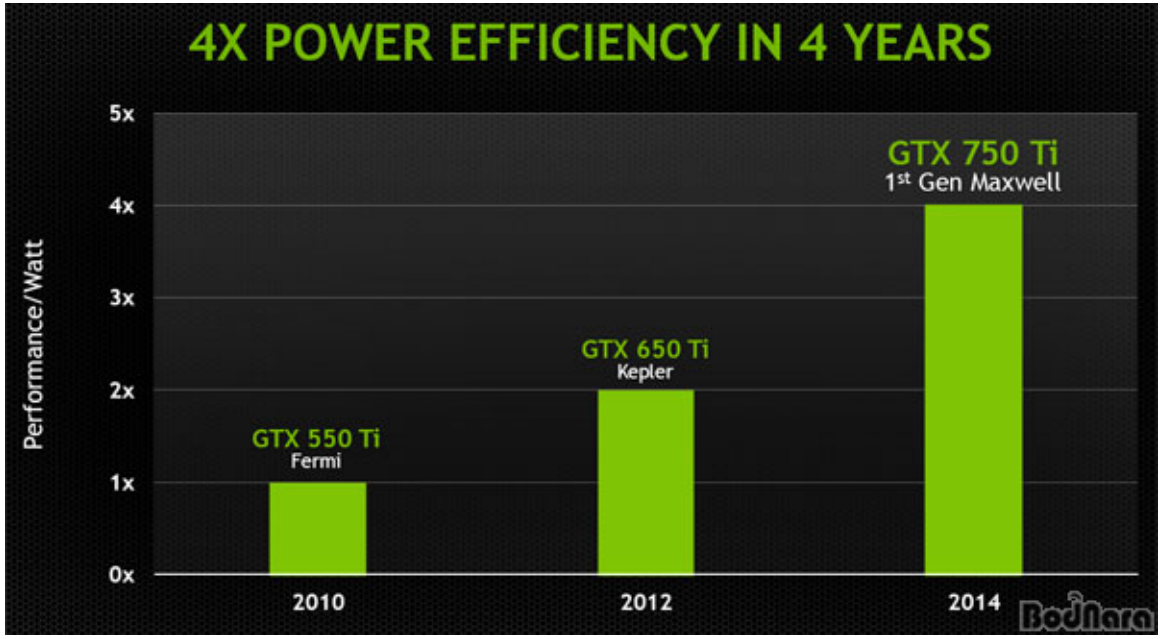


- HPC (High Performance Computing)
  - DP units (for 64bit floating point computation)
  - More warp scheduler
  - More Dispatch unit
  - 255 registers
  - SMX – 8 instructions (= 4 warp scheduler X 2 Dispatch unit)

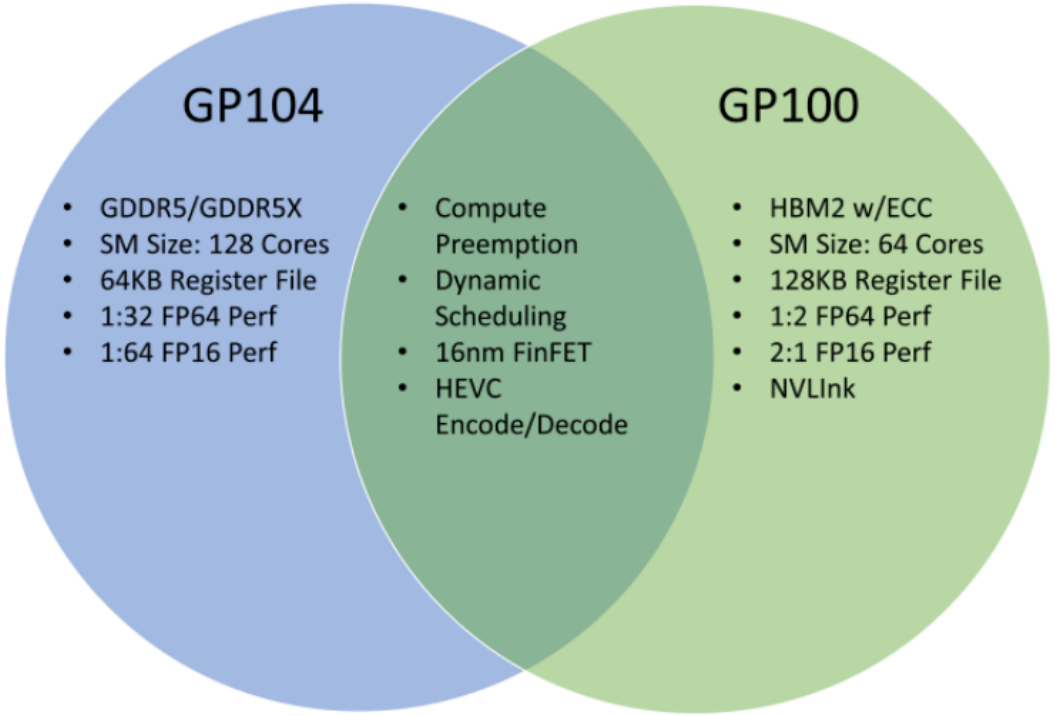
7-4. NVIDA GPU MODELS (MAXWELL)

- Geforce 700 (일부), 900, TITAN X (2014)
- (Still) 28nm
- Low Power SoC (System on Chip) technology
- SMX → SMM





7-5. NVIDA GPU MODELS (PASCAL)

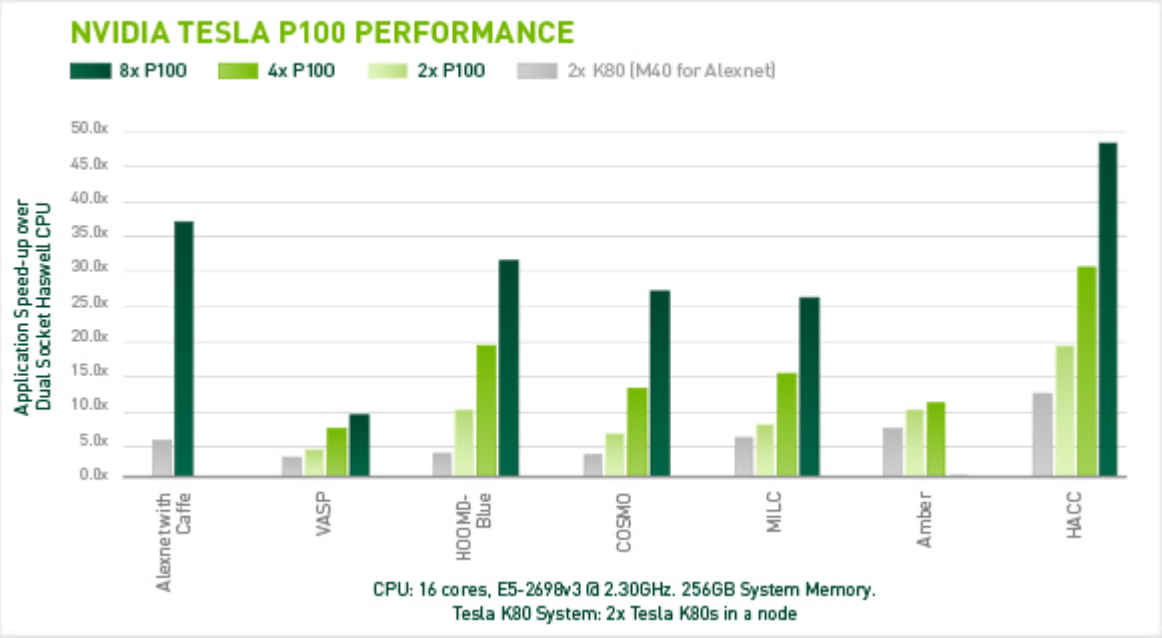


- Geforce 10, TITAN X, TITAN XP (2016)
- 2014 CNN architectures
  - GoogLeNet
  - VGG
- Pascal (GP)
  - GP 104 (for HPC)
  - GP 100 (for graphics)
- 2016 NVIDIA GTC (GPU Technology Conference)
  - AI, Deep Learning

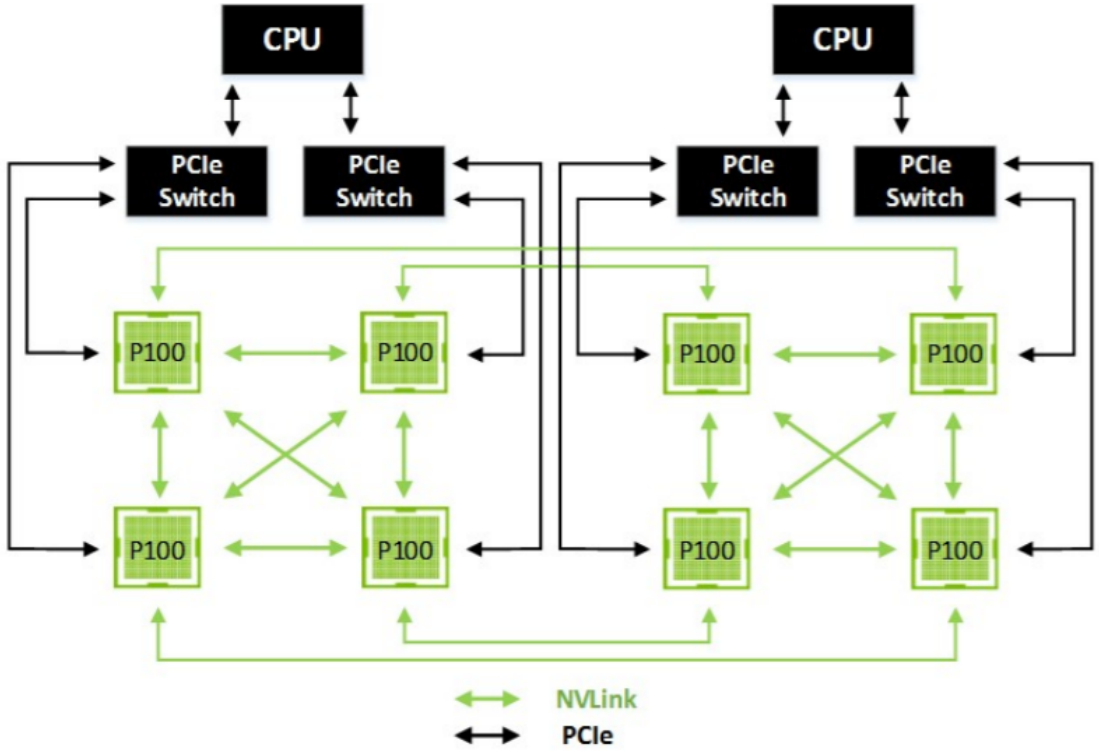
2016 NVIDIA GTC → https://www.youtube.com/watch?v=HyXRBNchE2A&feature=emb_logo

7-5. NVIDA GPU MODELS (PASCAL)

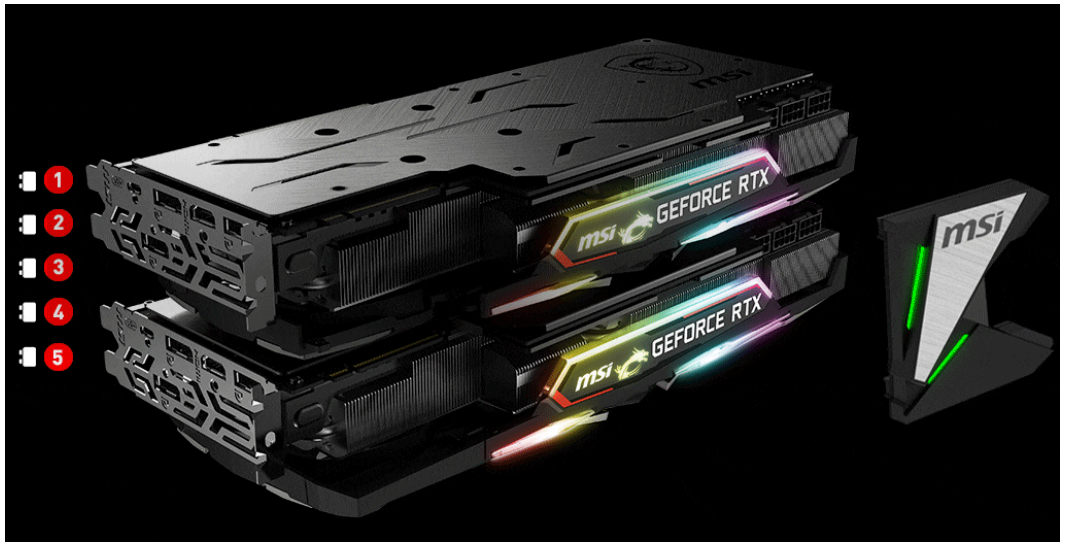
- 16nm
- 16 FP



7-5. NVIDA GPU MODELS (PASCAL)

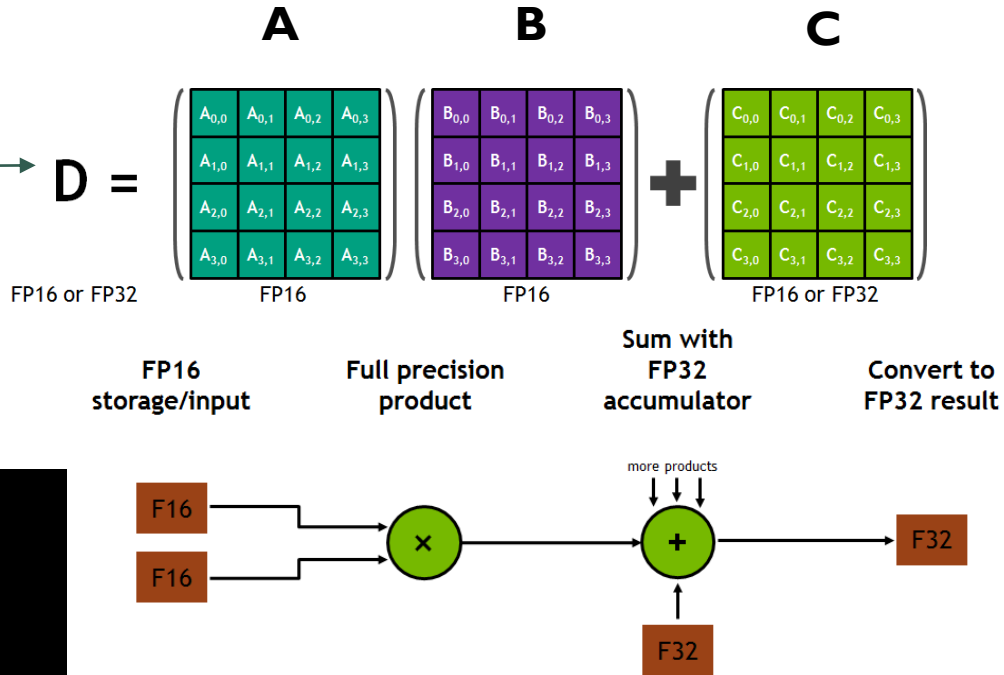


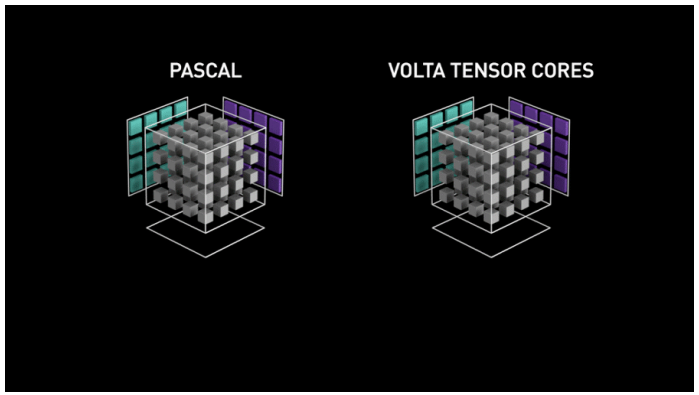
- Muli-GPU and accelerator
  - NVLink



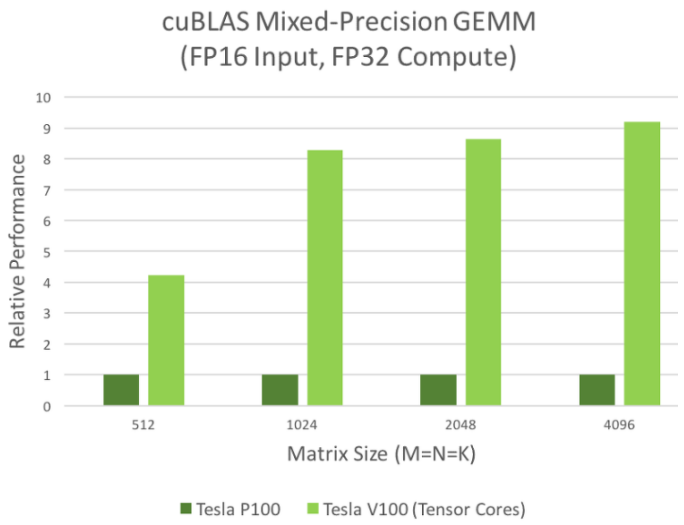
7-6. NVIDA GPU MODELS (VOLTA)

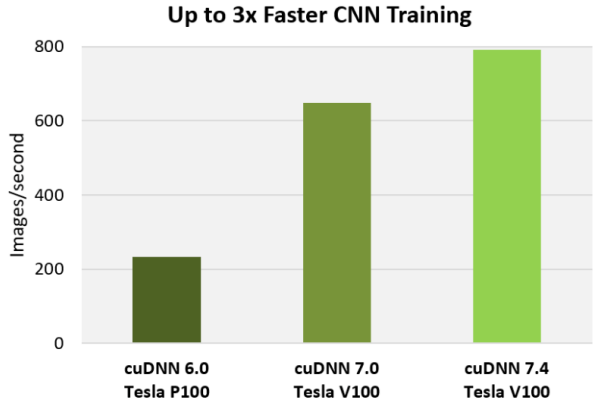
- TITANV (2017)
- 12nm
- CUDA core → FP32 core
- 8bit INT core
- Tensor core



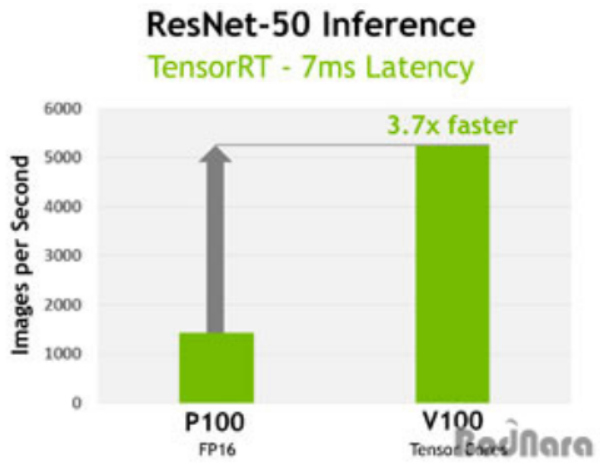


7-6. NVIDA GPU MODELS (VOLTA)



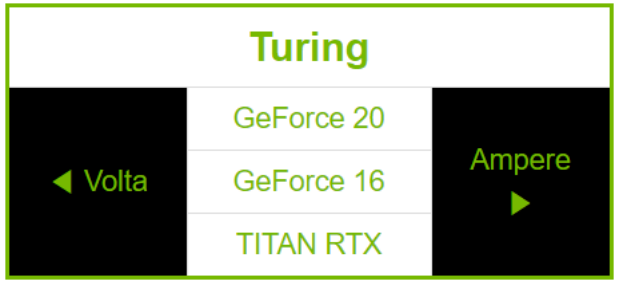


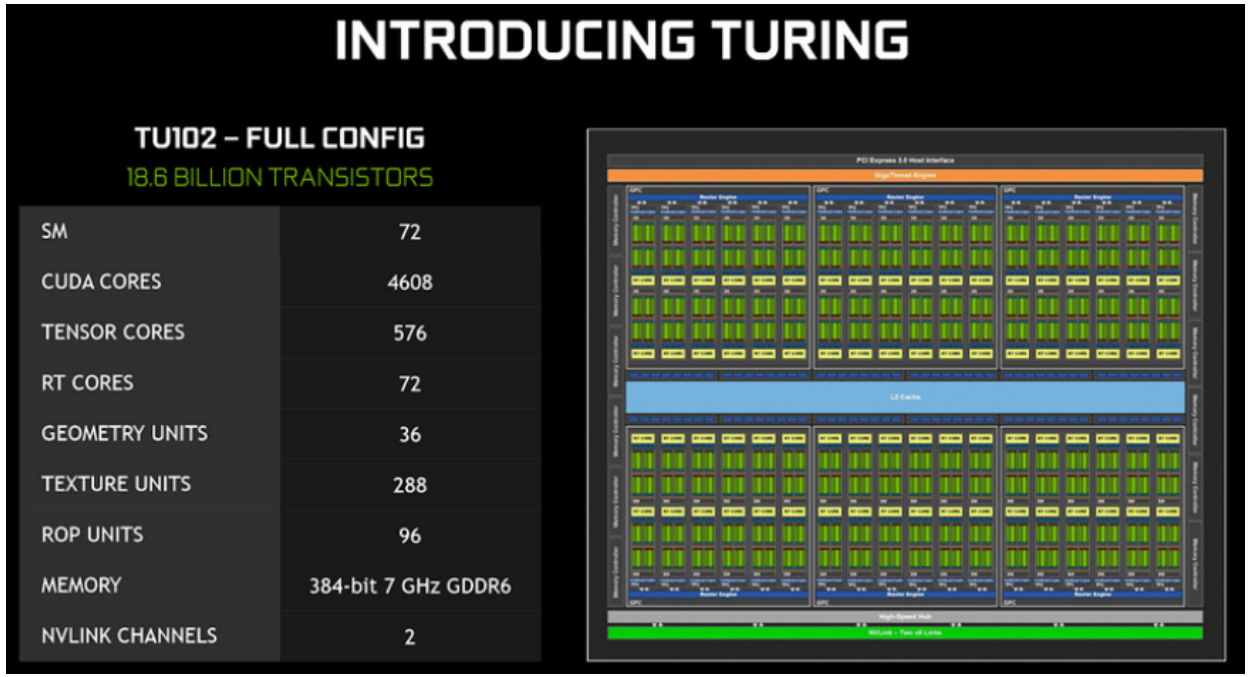
TensorFlow performance (images/sec), Tesla P100 + cuDNN 6 (FP32) on 17.12 NGC container, Tesla V100 + cuDNN 7.0 (Mixed) on 18.02 NGC container, Telsa V100 + cuDNN 7.4 (Mixed) on 18.10 NGC container, ResNet-50, Batch Size: 128



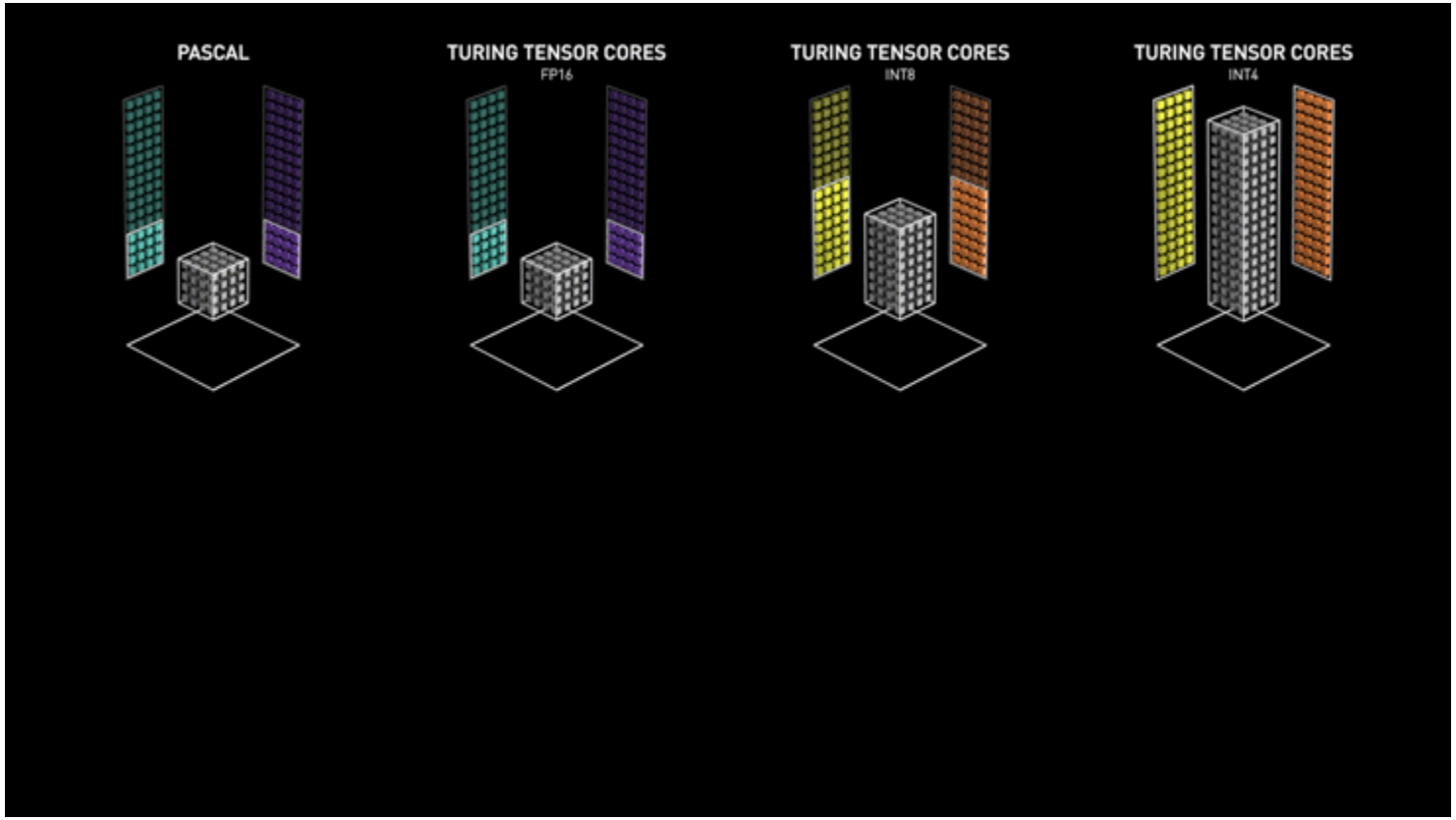
7-7. NVIDA GPU MODELS (TURING)

- Geforce 20, 16, TITAN RTX (2018)
- (Still) 12nm
- 4 INT
- RT core

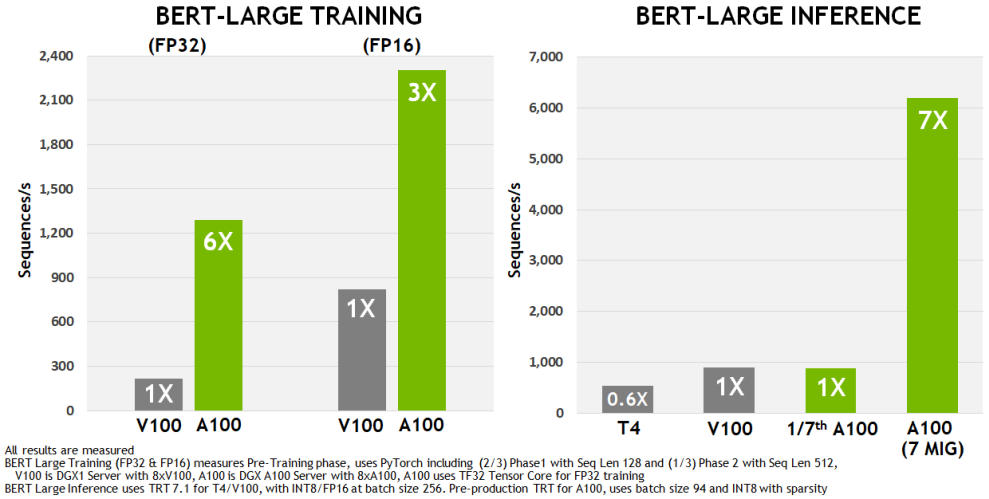




7-7. NVIDA GPU MODELS (TURING)

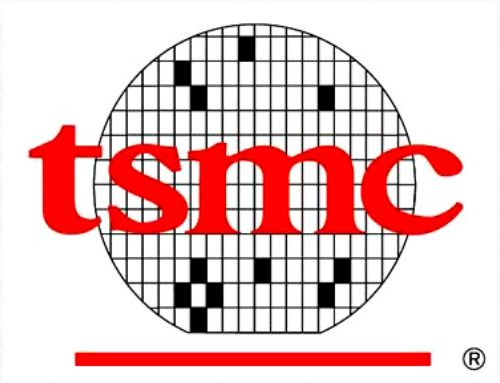


7-8. NVIDA GPU MODELS (AMPERE)



https://89douner.tistory.com/159?category=913897

8. Future of GPU



Thanks for listening !

https://www.youtube.com/watch?v=98Xis1W1mMk

Standford 강의