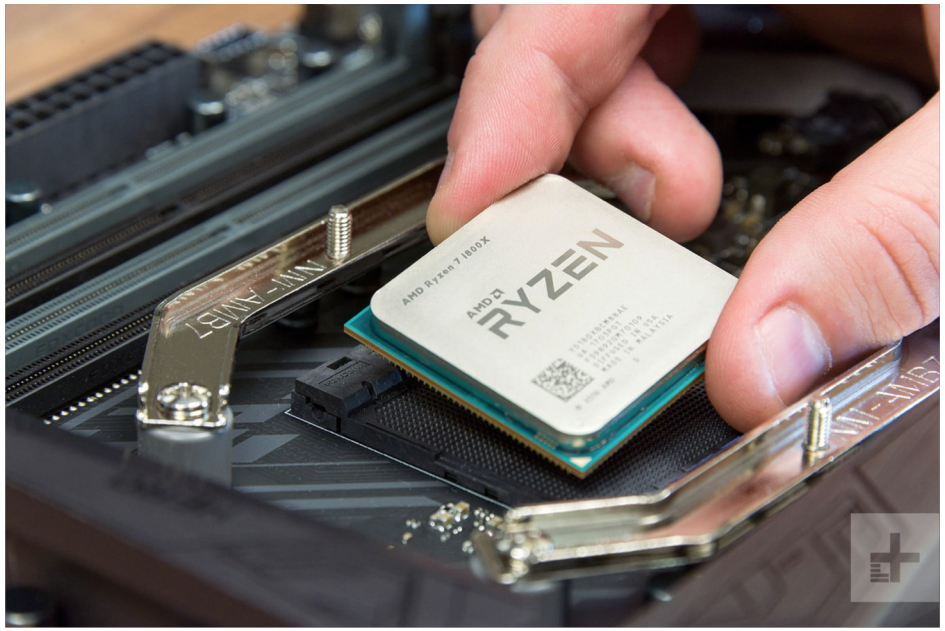
Deep Learning Hardware

Zhixin Piao
ShanghaiTech University

Hardware

CPU or GPU?

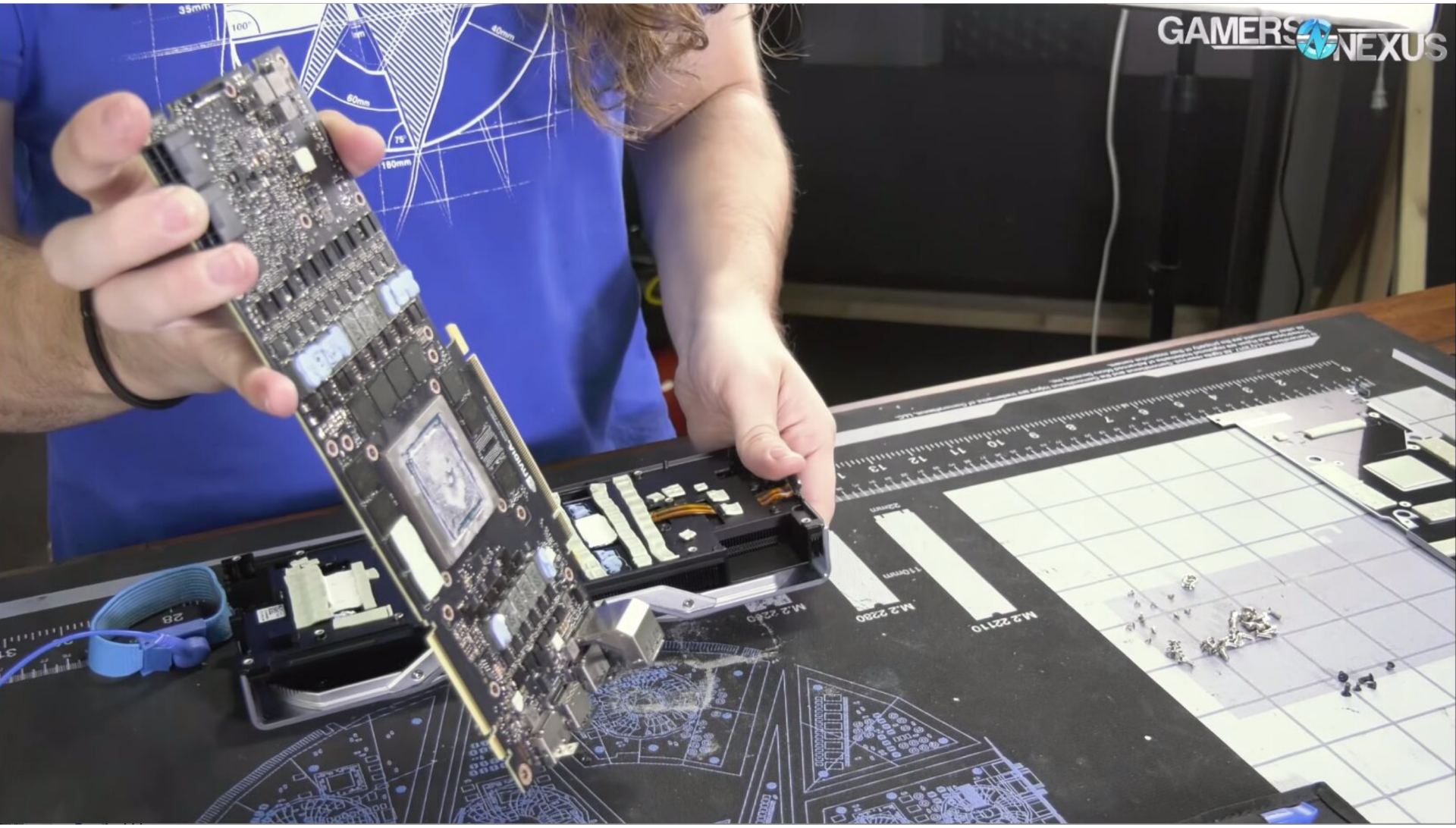
AMD RYZEN 1800X



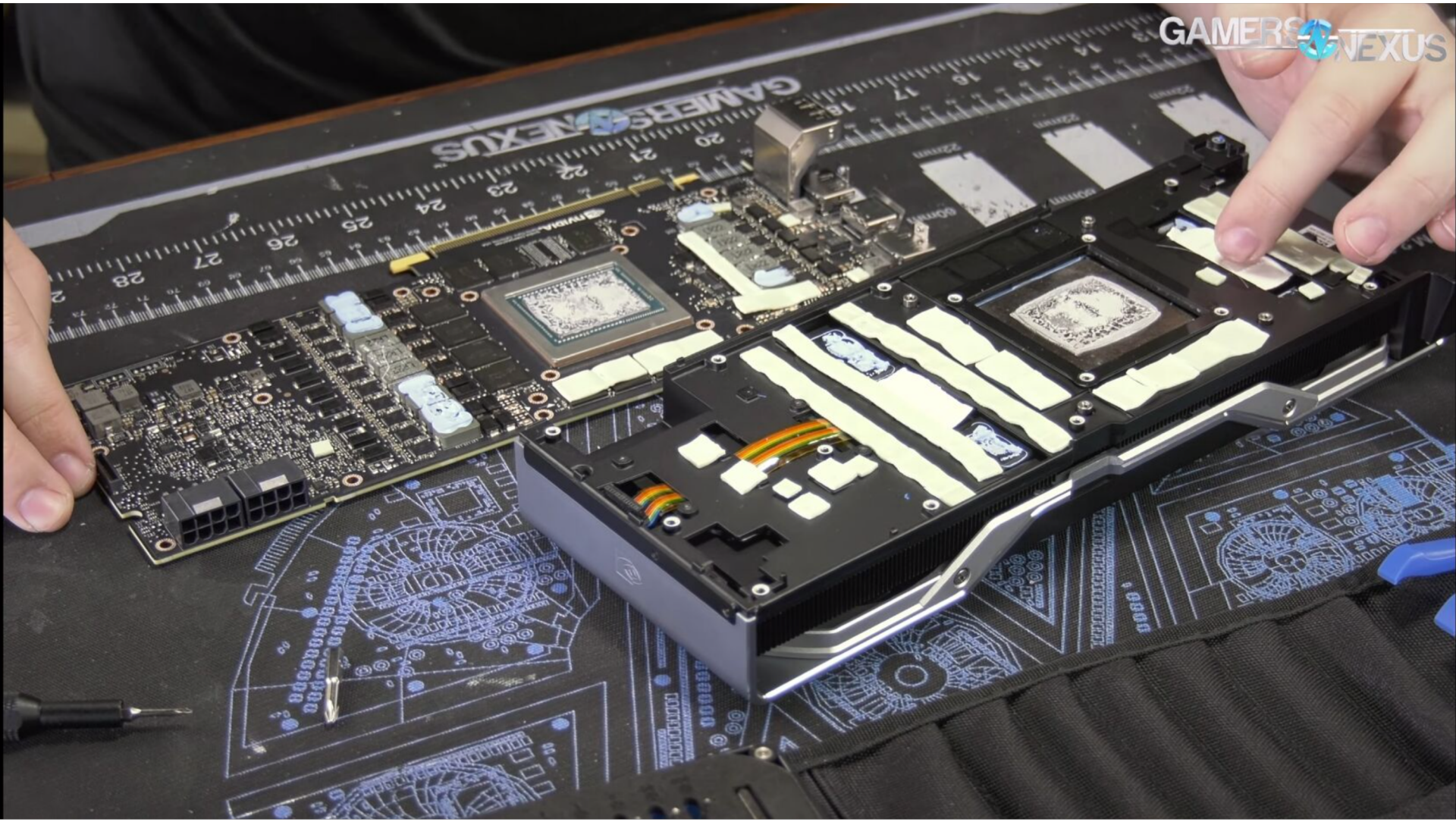
Bigger is stronger?

MSI GTX 1080Ti

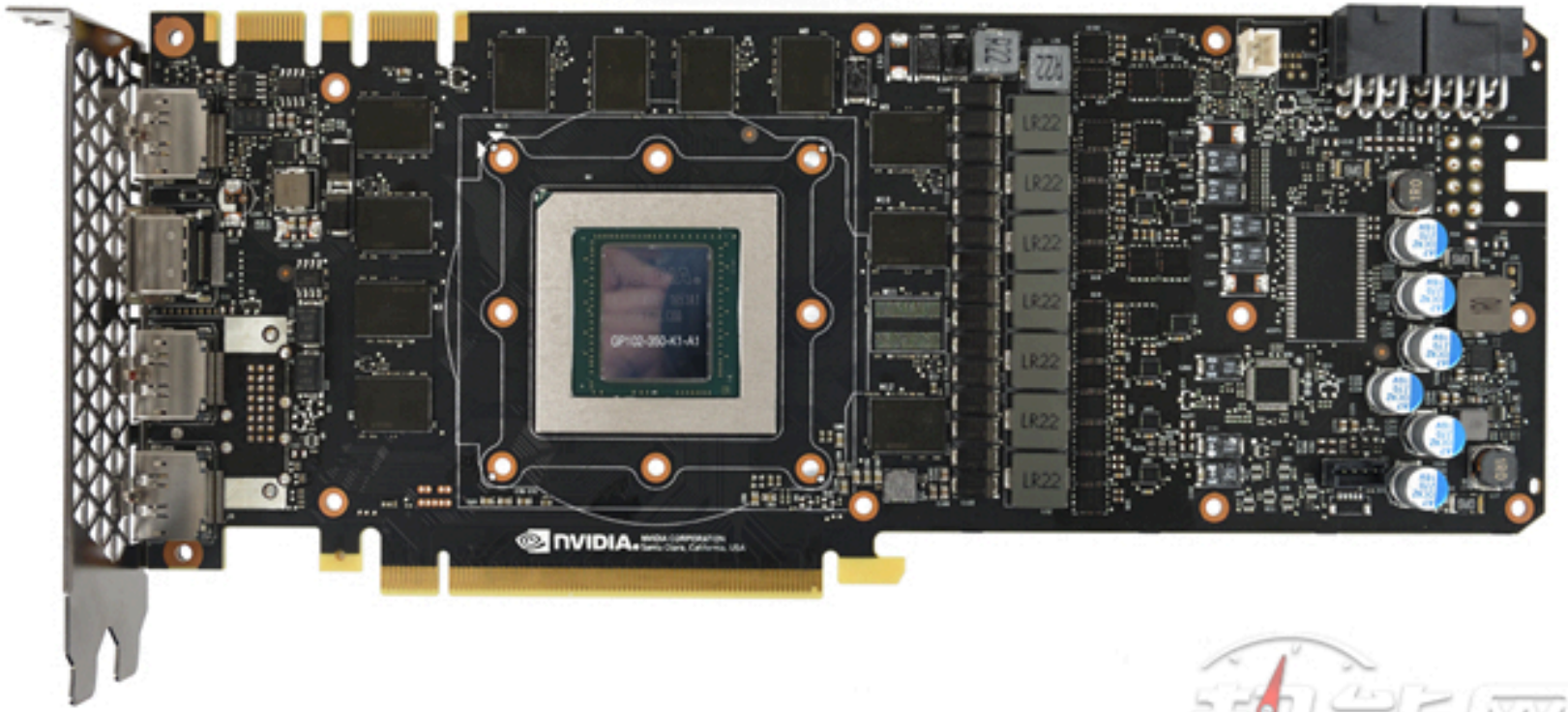
Howerver…



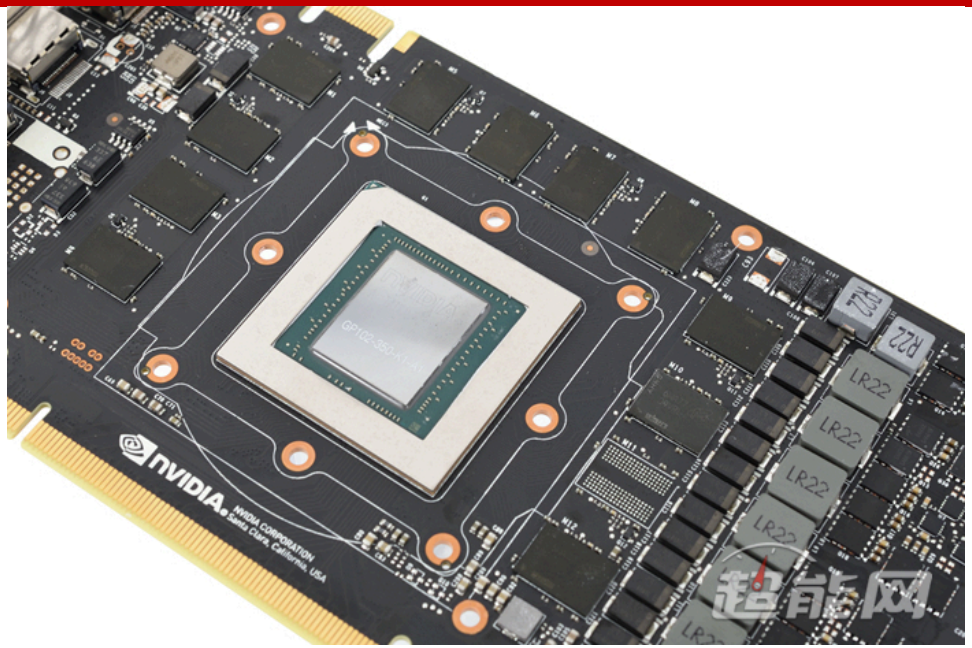
Howerver…



GTX 1080Ti

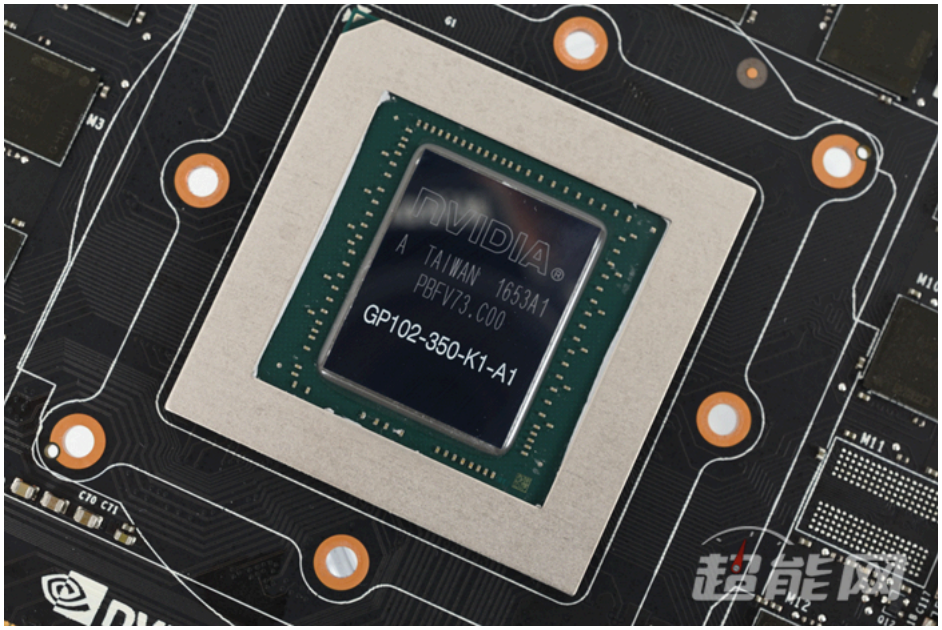


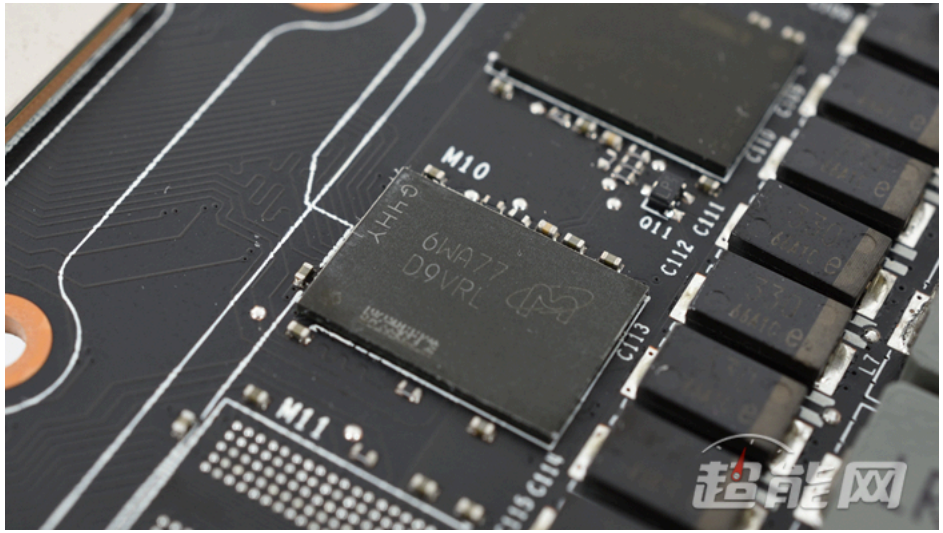
GTX 1080Ti



GPU

GDDR5X



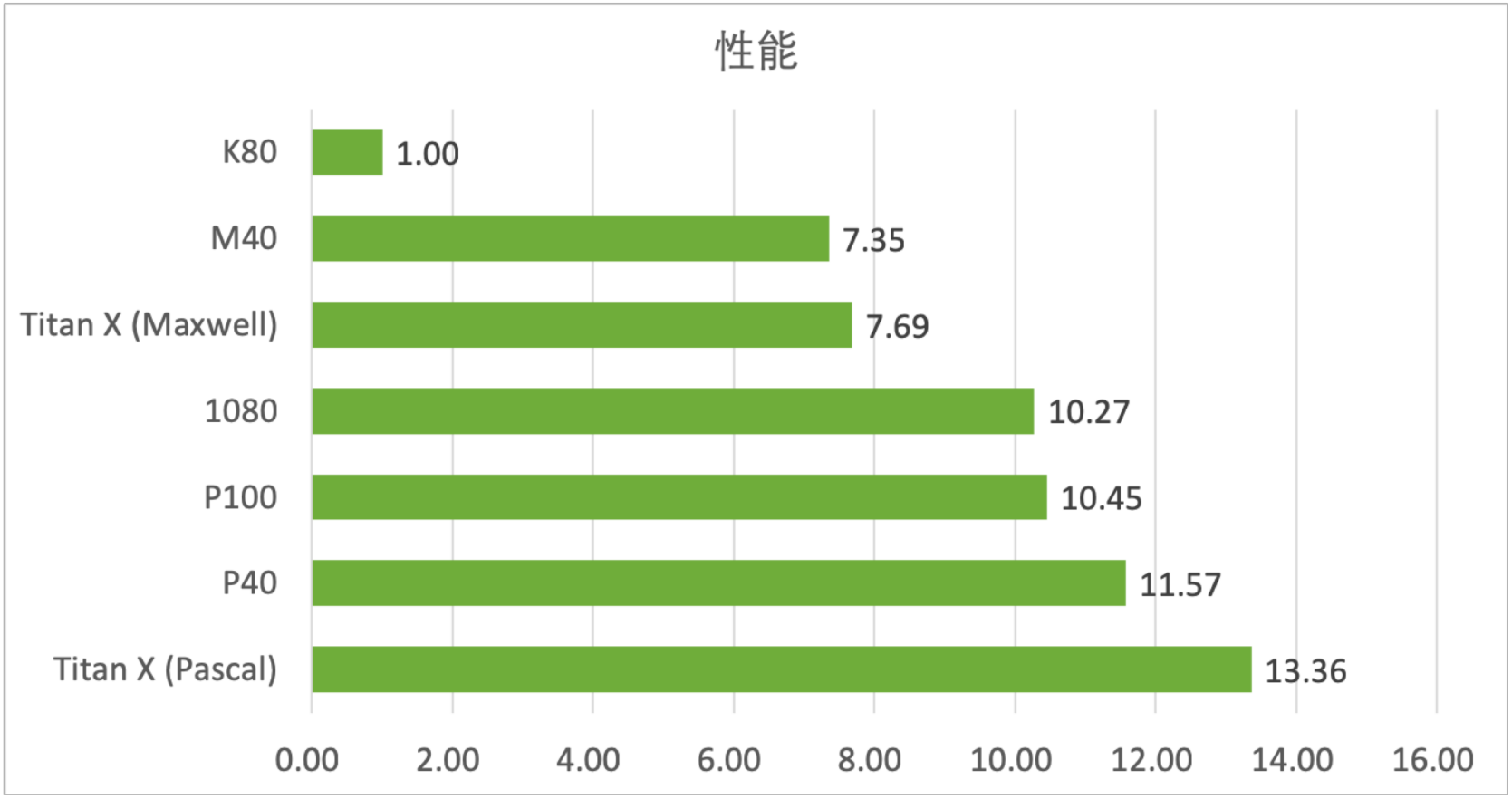


8 Titan X Server

GPU GFLOPS

gpu	gpu cores name	cuda cores number	GFLOPS (fp32)	GFLOPS (fp64)	GFLOPS (fp16)
gtx 1080	GP104	2560	8228-8873	257-277	128-139
gtx 1080ti	GP102	3584	10609-11340	332-354	166-177
titan xp	GP102	3840	10790-12150	337-380	169-190
p100	GP100	3584	9519–10609	4036–4670	18700
p40	GP102	3840	10007–11758	312.7–367.4	183.7
v100	GV100	5120	14899	7450	112000
titan v	GV100	5120	13800	6900	27600
rtx 2080	TU102	2944	10068	314.6	20137
rtx 2080ti	TU102	4352	13448	420.2	26895

GPU Performance for Deep Learning



HDD	Read(MB/s)	Write(MB/s)
Seq	122.841	123.240
512K	50.543	86.188
4K	0.698	2.206
4KQD32	1.984	2.210

SATA3 SSD	Read(MB/s)	Write(MB/s)
Seq	468.222	445.604
512K	362.094	412.903
4K	37.131	127.713
4KQD32	378.698	336.179

AI /public	Read(MB/s)	Write(MB/s)
Seq	675.907	452.696
512K	608.076	438.356
4K	75.350	62.919
4KQD32	225.899	205.622

AI /p300	Read(MB/s)	Write(MB/s)
Seq	816.587	709.142
512K	1113.500	641.202
4K	26.062	7.334
4KQD32	238.806	25.632

Network Speed Acceleration with IB and HSE

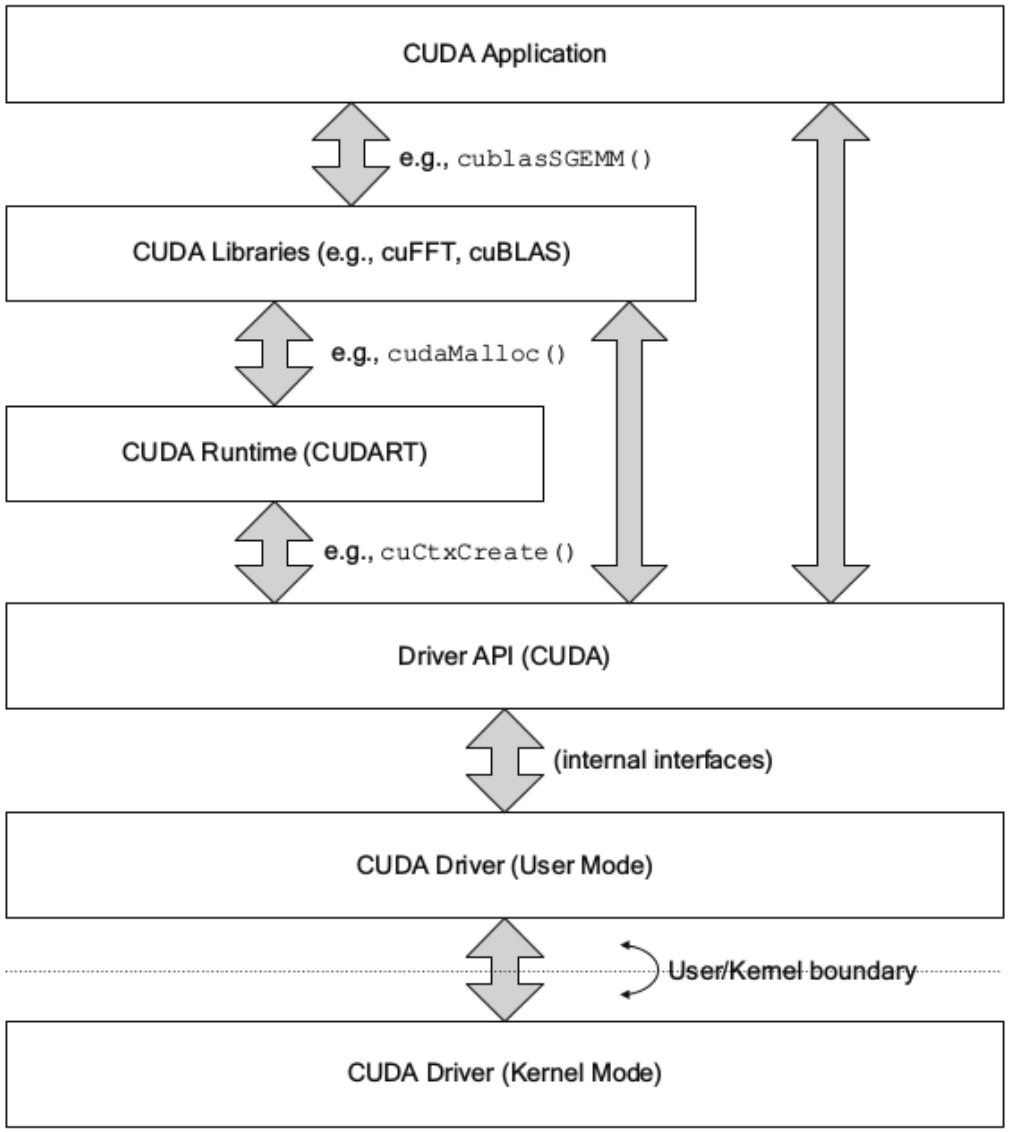
Ethernet (1979 -)	10 Mbit/sec
Fast Ethernet (1993 -)	100 Mbit/sec
Gigabit Ethernet (1995 -)	1000 Mbit /sec
ATM (1995 -)	155/622/1024 Mbit/sec
Myrinet (1993 -)	1 Gbit/sec
Fibre Channel (1994 -)	1 Gbit/sec
InfiniBand (2001 -)	2 Gbit/sec (1X SDR)
10-Gigabit Ethernet (2001 -)	10 Gbit/sec
InfiniBand (2003 -)	8 Gbit/sec (4X SDR)
InfiniBand (2005 -)	16 Gbit/sec (4X DDR)
	24 Gbit/sec (12X SDR)
InfiniBand (2007 -)	32 Gbit/sec (4X QDR)
40-Gigabit Ethernet (2010 -)	40 Gbit/sec
InfiniBand (2011 -)	54.6 Gbit/sec (4X FDR)
InfiniBand (2012 -)	2 x 54.6 Gbit/sec (4X Dual-FDR)
25-/50-Gigabit Ethernet (2014 -)	25/50 Gbit/sec
100-Gigabit Ethernet (2015 -)	100 Gbit/sec
Omni-Path (2015 -)	100 Gbit/sec
InfiniBand (2015 -)	100 Gbit/sec (4X EDR)
InfiniBand (2016 -)	200 Gbit/sec (4X HDR)

100 times in the last 15 years

组别	序号	主机名	功能	带内管理	IB网络	IPMI带外管理
	0	node100	admin 管理节点	10.10.10.100	12.12.12.100	10.10.20.100
GPU集群	1	node01	计算节点	10.10.10.101	12.12.12.101	10.10.20.101
	2	node02	计算节点	10.10.10.102	12.12.12.102	10.10.20.102
	3	node03	计算节点	10.10.10.103	12.12.12.103	10.10.20.103
	4	node04	计算节点	10.10.10.104	12.12.12.104	10.10.20.104
	5	node05	计算节点	10.10.10.105	12.12.12.105	10.10.20.105
	6	node06	计算节点	10.10.10.106	12.12.12.106	10.10.20.106
	7	node07	计算节点	10.10.10.107	12.12.12.107	10.10.20.107
	8	node08	计算节点	10.10.10.108	12.12.12.108	10.10.20.108
	9	node09	计算节点	10.10.10.109	12.12.12.109	10.10.20.109
	10	node10	计算节点	10.10.10.110	12.12.12.110	10.10.20.110
	11	node11	计算节点	10.10.10.111	12.12.12.111	10.10.20.111
	12	node12	计算节点	10.10.10.112	12.12.12.112	10.10.20.112
	13	node13	计算节点	10.10.10.113	12.12.12.113	10.10.20.113
	14	node14	计算节点	10.10.10.114	12.12.12.114	10.10.20.114
	15	node15	计算节点	10.10.10.115	12.12.12.115	10.10.20.115
	16	node16	计算节点	10.10.10.116	12.12.12.116	10.10.20.116
	17	node17	计算节点	10.10.10.117	12.12.12.117	10.10.20.117
	18	node18	计算节点	10.10.10.118	12.12.12.118	10.10.20.118
	19	node19	计算节点	10.10.10.119	12.12.12.119	10.10.20.119
	20	node20	计算节点	10.10.10.120	12.12.12.120	10.10.20.120
	21	node21	计算节点	10.10.10.121	12.12.12.121	10.10.20.121
	22	node22	计算节点	10.10.10.122	12.12.12.122	10.10.20.122
	23	node23	计算节点	10.10.10.123	12.12.12.123	10.10.20.123
	24	node24	计算节点	10.10.10.124	12.12.12.124	10.10.20.124
	25	node25	计算节点	10.10.10.125	12.12.12.125	10.10.20.125
	26	node26	计算节点	10.10.10.126	12.12.12.126	10.10.20.126

Driver

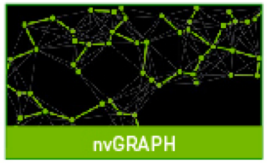
CUDA (Compute Unified Device Architecture) is a parallel computing platform and application programming interface (API) model created by Nvidia.

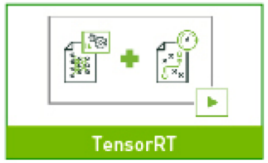


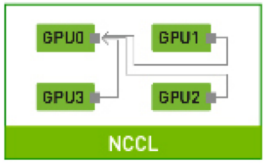
CUDA Libraries

DOMAIN-SPECIFIC


cuDNN


nvGRAPH

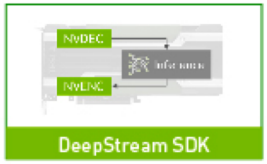

TensorRT



NCCL

VISUAL PROCESSING


NVIDIA NPP



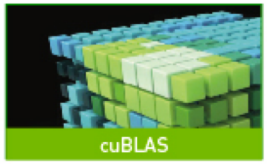
DeepStream SDK

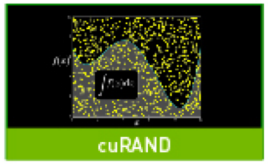

NVIDIA CODEC SDK

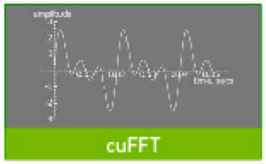

IndeX Framework

LINEAR ALGEBRA

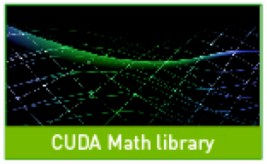

cuSPARSE


cuBLAS

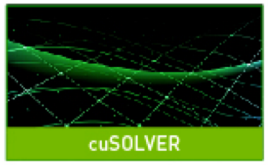

cuRAND


cuFFT

MATH ALGORITHMS

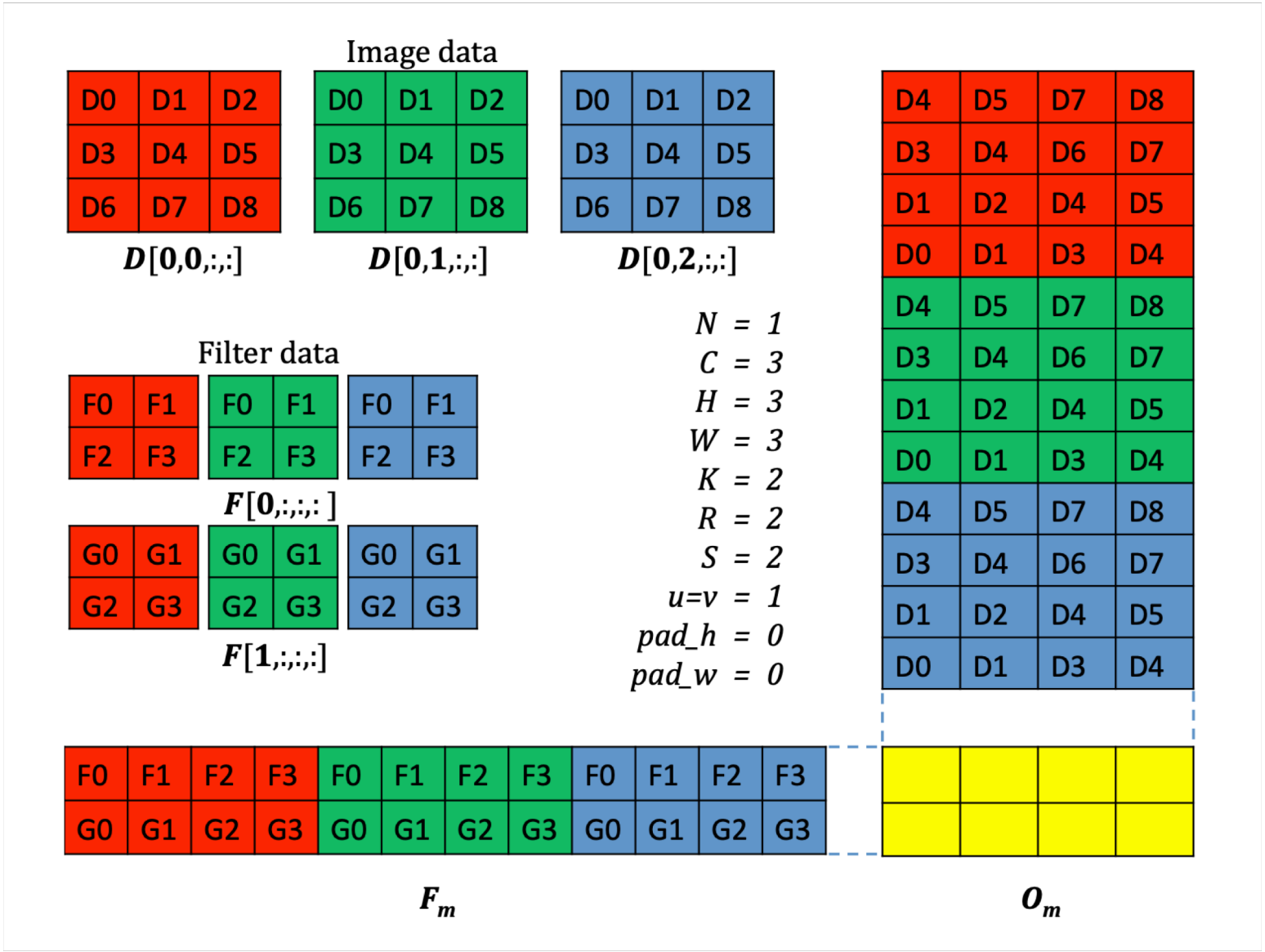

CUDA Math library


AmgX


cuSOLVER

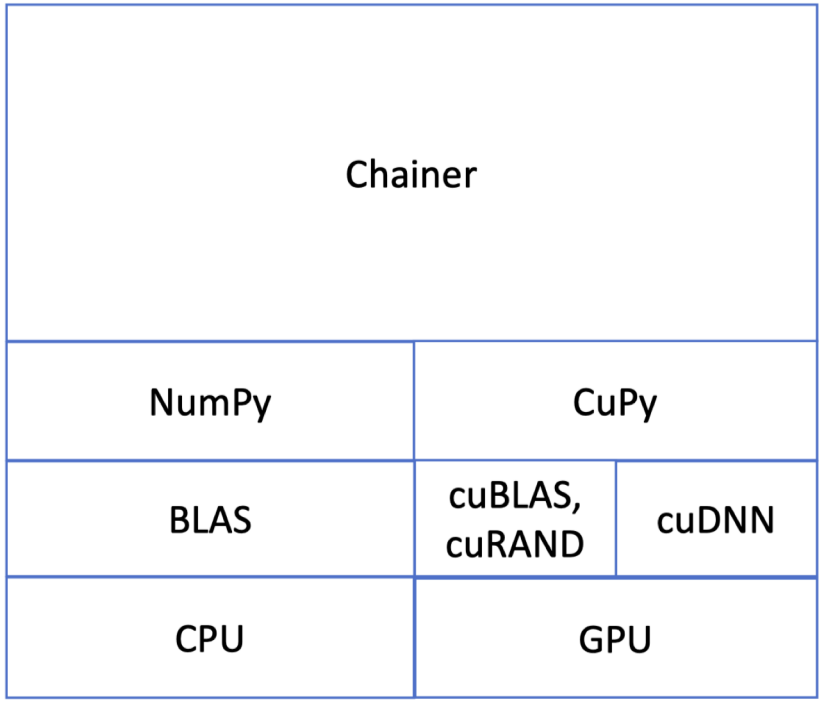
THRUST LIBRARY

e.g. Convolution lowering in cuDNN



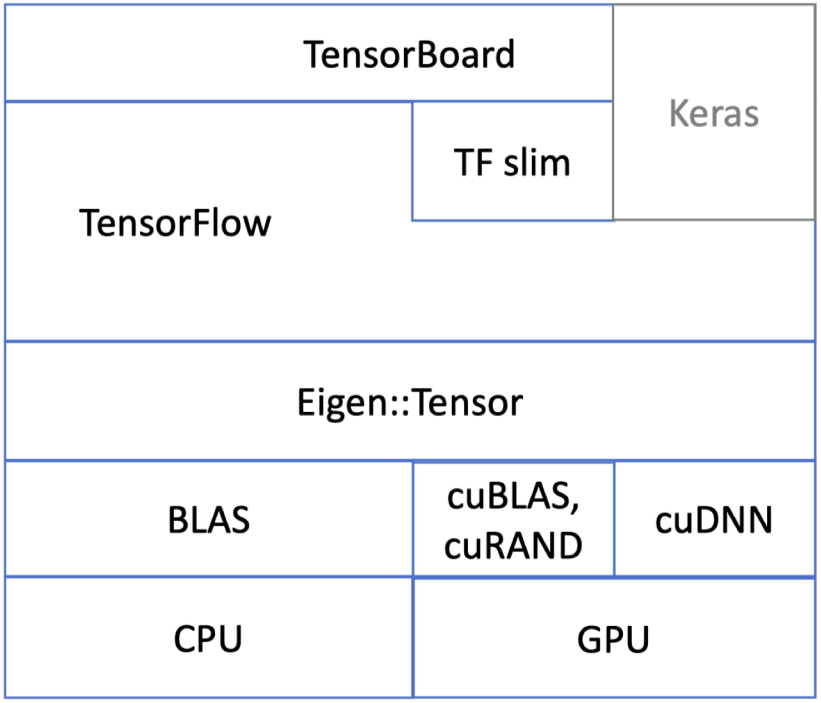
Technology stack of **Chainer**

name
Graphical visualization
Machine learning workflow management
Computational graph management
Multi-dimensional array processing
Numerical computation
Computational device



Technology stack of **TensorFlow**

name
Graphical visualization
Machine learning workflow management
Computational graph management
Multi-dimensional array processing
Numerical computation
Computational device

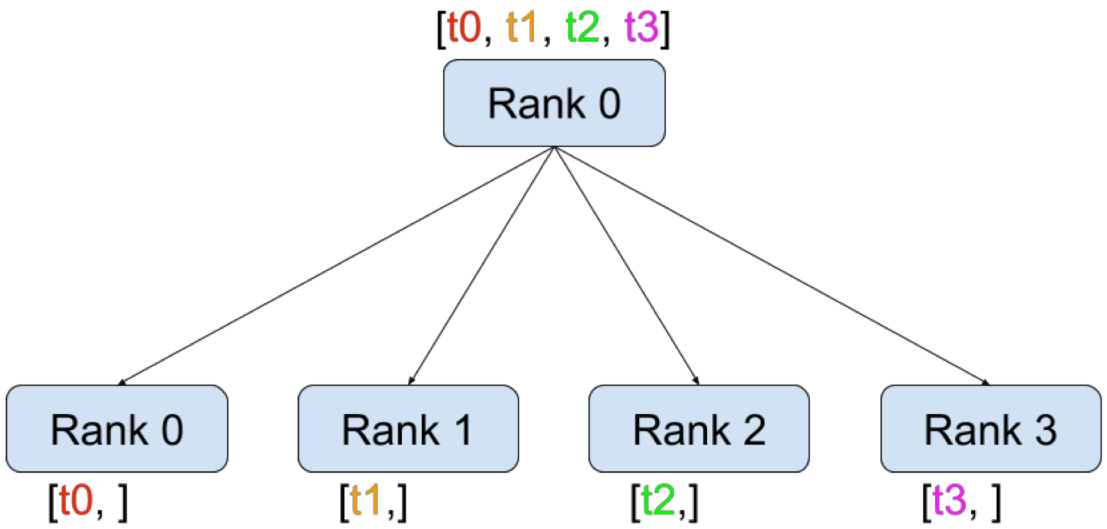


Docker Container 内置环境

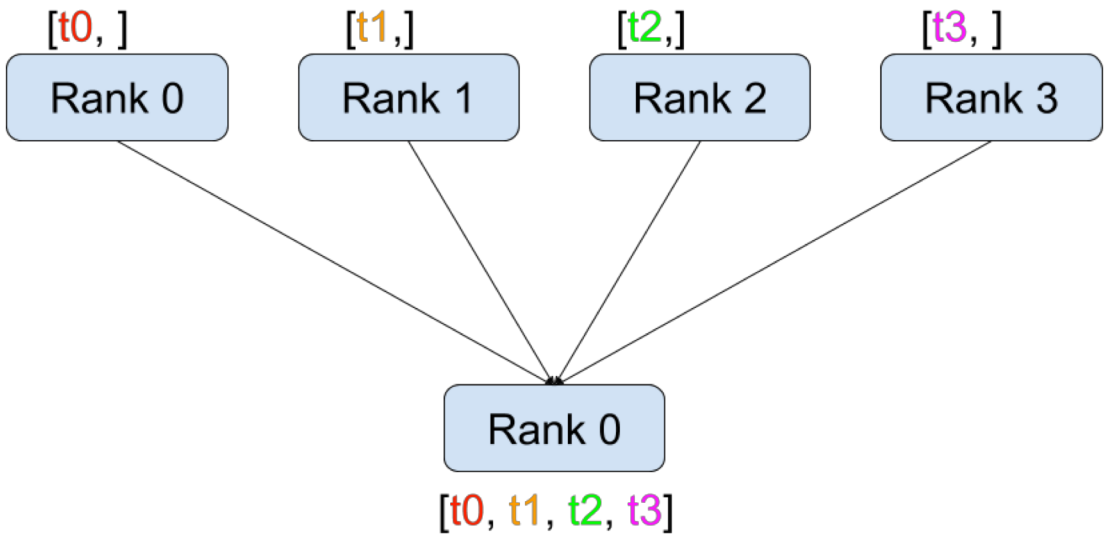
环境	版本	环境	版本
ubuntu	16.04.4 LTS	nvidia driver	396.44 (不可更改)
cuda	9.0.176	cudnn	7.1.4
python	3.6.6	tensorflow	1.9.0
pytorch	0.4.0	mxnet	1.2.1
keras	2.2.2	lasagne	0.2.dev1
cntk	2.5.1	chainer	4.3.1
caffe	1.0.0	caffe2	0.8.2
torch	7	sonnet	1.23
theano	1.0.1		

Deep Learning Distribution

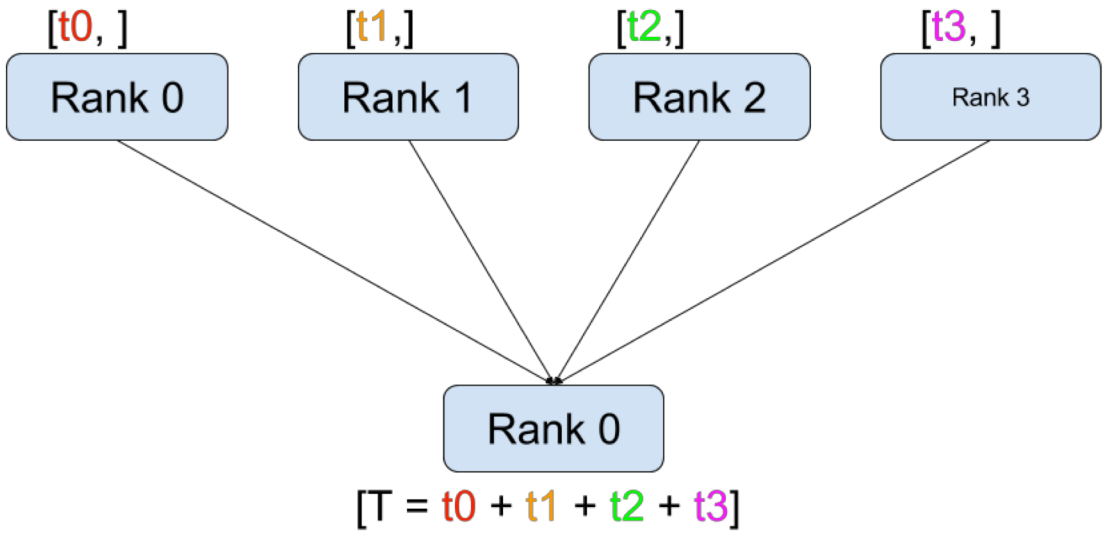
Scatter



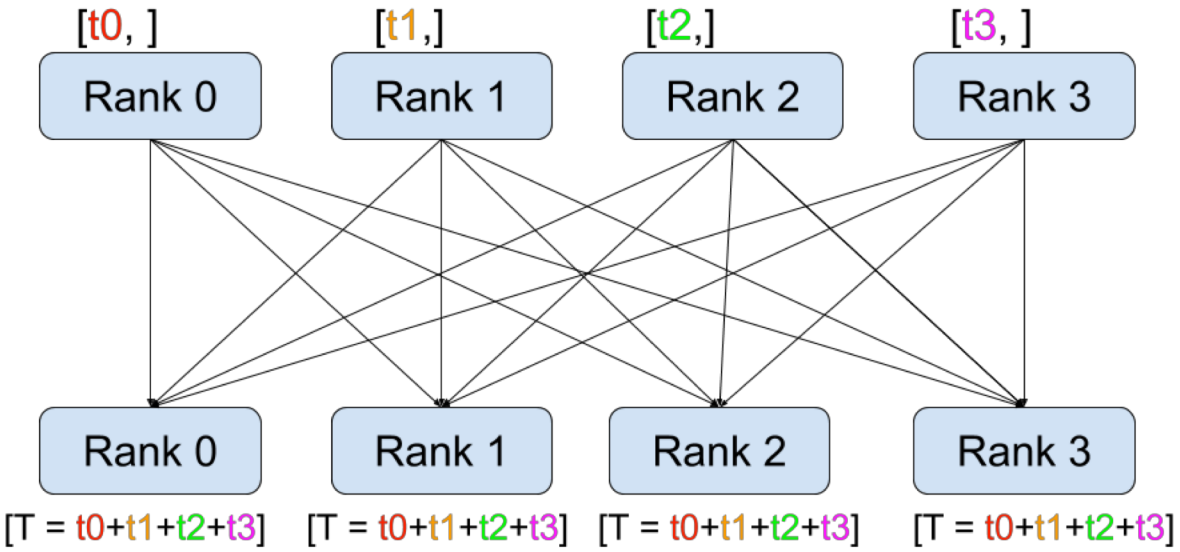
Gather



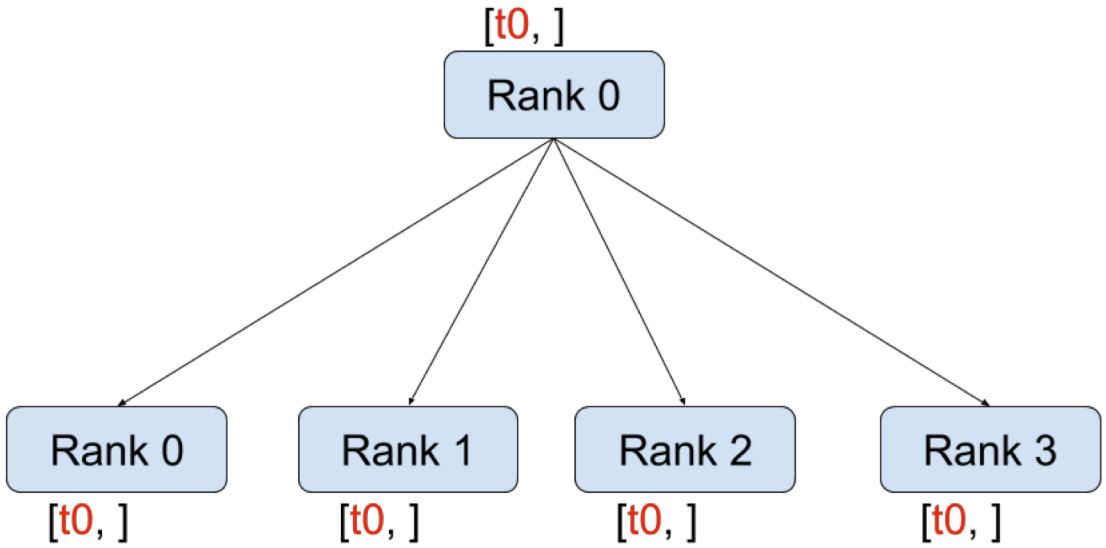
Reduce



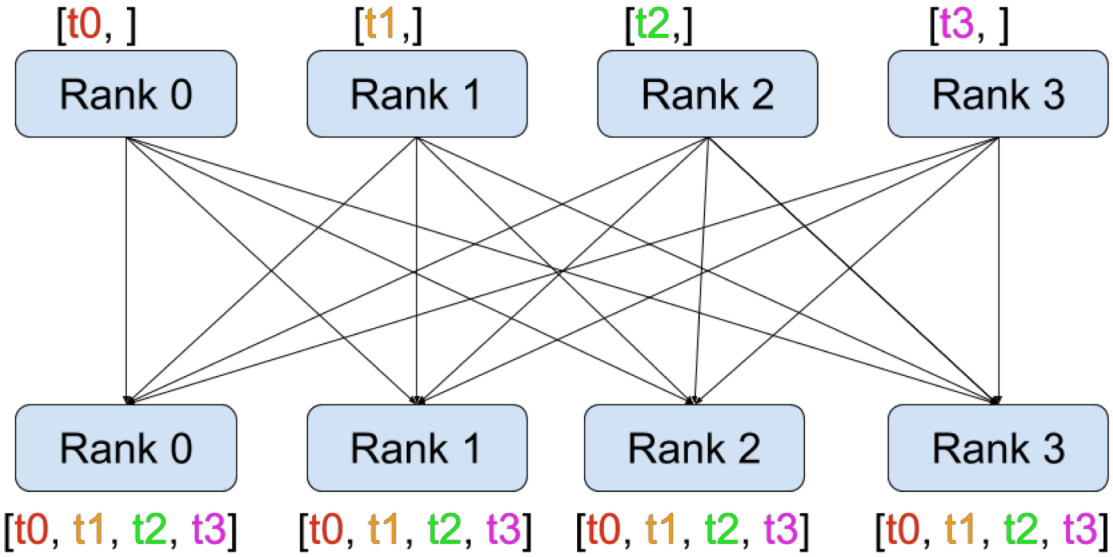
All Reduce



Broadcast



All-Gather



torch.nn.parallel.DistributedDataParallel

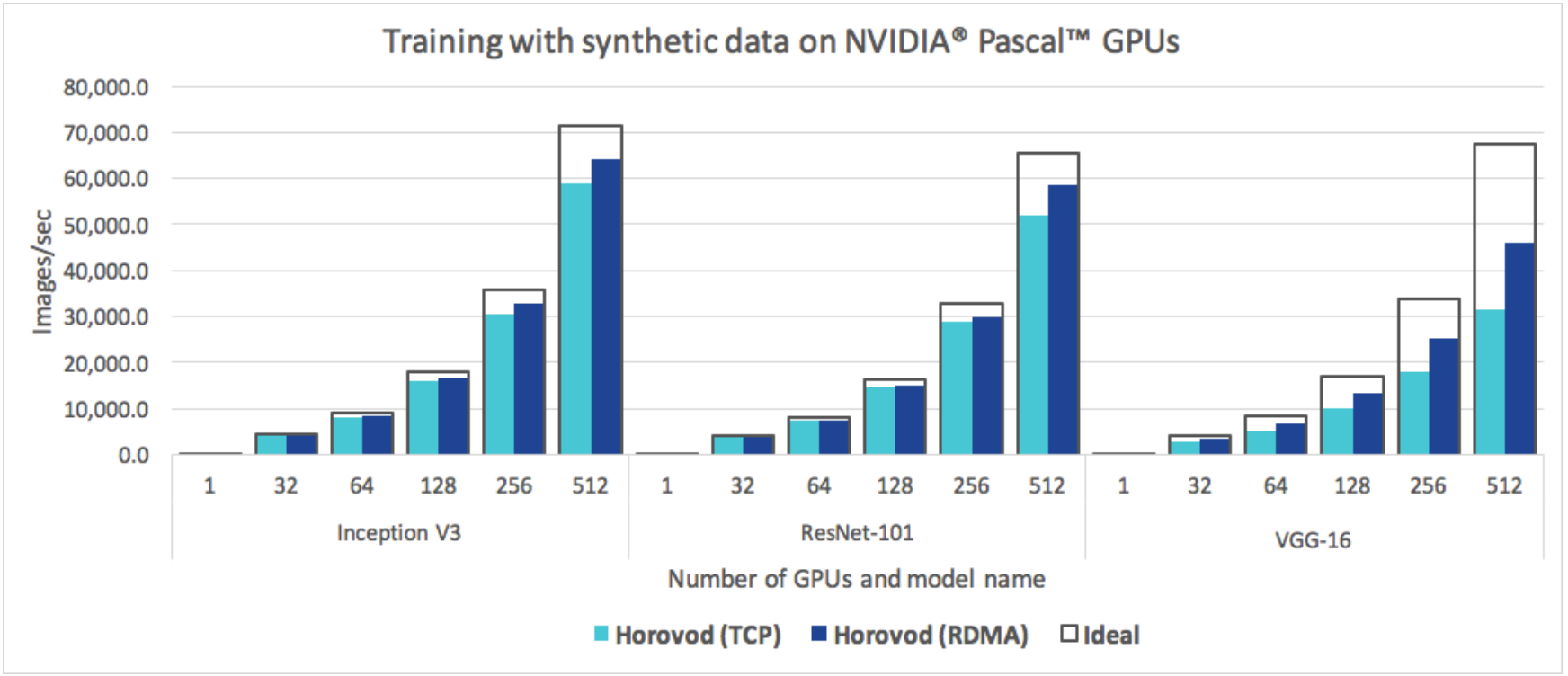
Example:

```
>>> torch.distributed.init_process_group(world_size=4, init_method='...')
>>> net = torch.nn.DistributedDataParallel(model)
```

Note

Parameters are never broadcast between processes. The module performs an all-reduce step on gradients and assumes that they will be modified by the optimizer in all processes in the same way. Buffers (e.g. BatchNorm stats) are broadcast from the module in process of rank 0, to all other replicas in the system in every iteration.


HOROVOD


Training with synthetic data on NVIDIA® Pascal™ GPUs
80,000.0
70,000.0
60,000.0
50,000.0
40,000.0
30,000.0
20,000.0
10,000.0
0.0
Images/sec
1
32
64
128
256
512
Inception V3
ResNet-101
VGG-16
Number of GPUs and model name
Horovod (TCP)
Horovod (RDMA)
Ideal

Running Horovod

The example commands below show how to run distributed training. See the Running Horovod page for more instructions, including RoCE/InfiniBand tweaks and tips for dealing with hangs.

1. To run on a machine with 4 GPUs:

```
$ mpirun -np 4 \
    -H localhost:4 \
    -bind-to none -map-by slot \
    -x NCCL_DEBUG=INFO -x LD_LIBRARY_PATH -x PATH \
    -mca pml ob1 -mca btl ^openib \
    python train.py
```

2. To run on 4 machines with 4 GPUs each:

```
$ mpirun -np 16 \
    -H server1:4,server2:4,server3:4,server4:4 \
    -bind-to none -map-by slot \
    -x NCCL_DEBUG=INFO -x LD_LIBRARY_PATH -x PATH \
    -mca pml ob1 -mca btl ^openib \
    python train.py
```

tonysy / **Distributed-Tutorial**

Watch 1 | Unstar 1 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Insights

Tutorial for Distributed Deep Learning on Serveral Frameworks,e.g. Keras, Tensorflow, PyTorch

1 commit | 1 branch | 0 releases | 1 contributor

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

tonysy first commit — Latest commit `33252f0` 6 hours ago

examples	first commit	6 hours ago
.gitignore	first commit	6 hours ago
README.md	first commit	6 hours ago

README.md

Install Softwares to support distributed training on AI cluster

本教程基于horovod分布式训练框架，支持keras, tensorflow和pytorch

以下相关版本已经过测试：

- keras: 2.2.2
- tensorflow: 1.10.0
- pytorch: 0.4.0

硬件环境

本教程针对ShanghaiTech AI集群