IIT Kharagpur | IIT Madras | IIT Goa | IIT Palakkad

Applied Accelerated Artificial Intelligence

NVIDIA

Introduction to AI Accelerators

GPUs

Satyadhyan Chickerur, PhD
Professor
School of Computer Science and Engineering
KLE Technological University
NVIDIA DLI Ambassador/Instructor

National Supercomputing Mission

Centre for Development of Advanced Computing

Agenda

- What are AI Accelerators
- Where are they used
- How do they work
- One view of AI Accelerators
- Second View of AI Accelerators
- GPUs
- CPU vs. Parallel vs. GPU
- PARAM Shivay and DGX I
- Benefits

What are Artificial Intelligence (AI) Accelerators?

An AI accelerator is high-performance specialized hardware that is optimized for AI workloads such as neural networks, machine learning, and other data-intensive or sensor-driven processes.

The three main types are

1. Central Processing Unit (CPU)

2. Graphics Processing Unit (GPU)

3. Field-Programmable Gate Arrays (FPGA)/Application-Specific Integrated Circuit (ASIC)

Where are they used

We can divide AI Accelerators into two groups (based on where we use them):

- Data centres
- Edge Devices

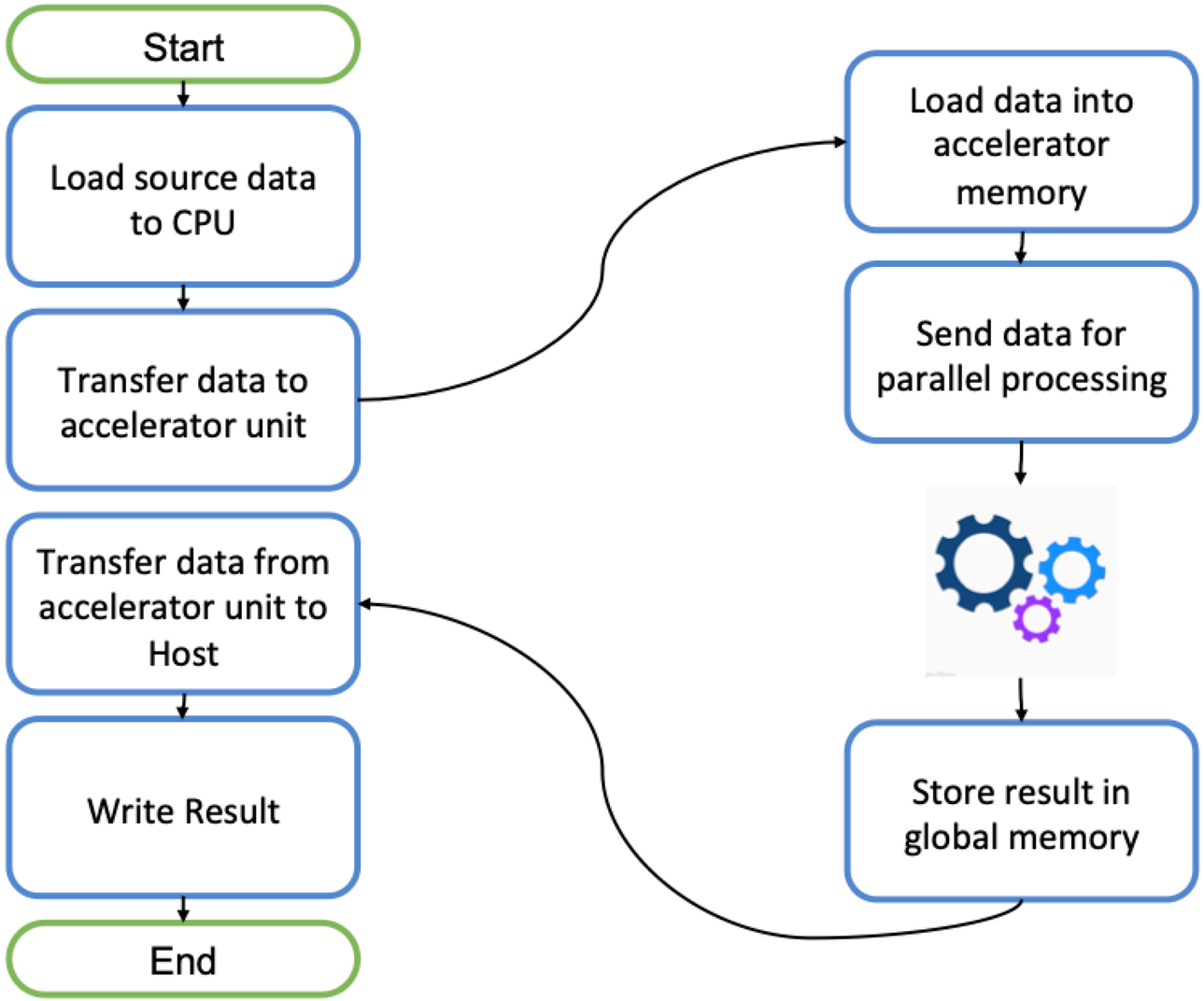
Data Centres

Source:https://medium.com/swlh/how_to_move_a_data_center_4ab553cd131d

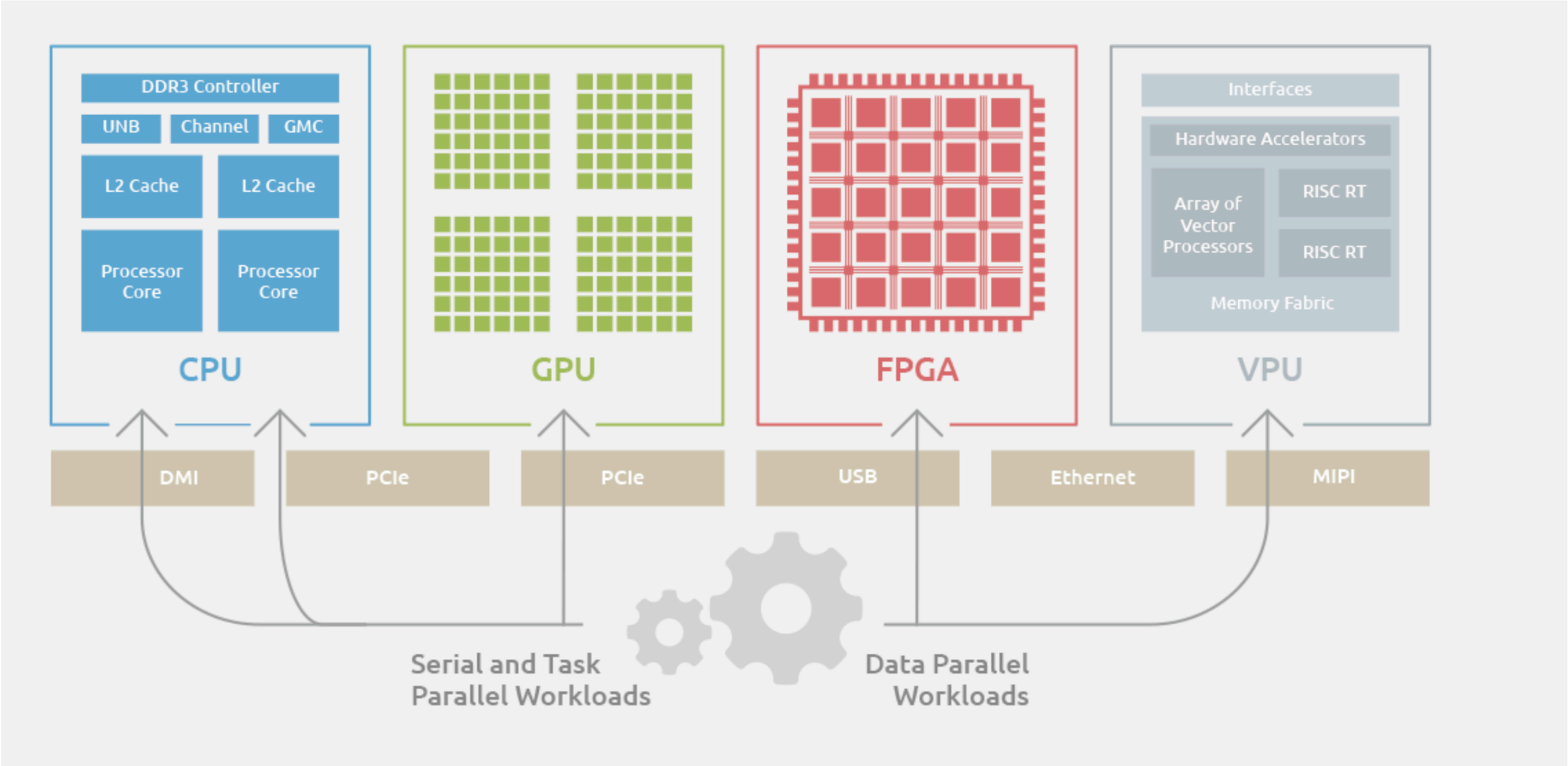
Edge

Source: https://thinkrobotics.in/products/nvidia-jetson-nano?variant=33399250714710&utm_medium=product_sync&utm_source=google&utm_content=sag_organic&utm_campaign=sag_organic&gclid=Cj0KCQiAxc6PBhCEARIsAH8Hff1InEn2zlWq3SbMoC0okpYEgqVMV_CBj6dWu1blABrTmfcr5eHYoa4aAr_IEALw_wcB

How do they work



One View of AI Accelerators



Source : https://www.adlinktech.com/en/gpu-computing

Demo 1

- CPU related info

1. lscpu : displays CPU information
2. cat /proc/cpuinfo

- GPU related info

1. nvidia-smi : displays GPU information
2. gpustat : GPU information

```
u18@slave-node:~$ lscpu
Architecture:            x86_64
CPU op-mode(s):          32-bit, 64-bit
Byte Order:              Little Endian
CPU(s):                  8
On-line CPU(s) list:     0-7
Thread(s) per core:      2
Core(s) per socket:      4
Socket(s):               1
NUMA node(s):            1
Vendor ID:               GenuineIntel
CPU family:              6
Model:                   60
Model name:              Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz
Stepping:                3
CPU MHz:                 2440.303
CPU max MHz:             3900.0000
CPU min MHz:             800.0000
BogoMIPS:                6783.61
Virtualization:          VT-x
L1d cache:               32K
L1i cache:               32K
L2 cache:                256K
L3 cache:                8192K
NUMA node0 CPU(s):       0-7
```

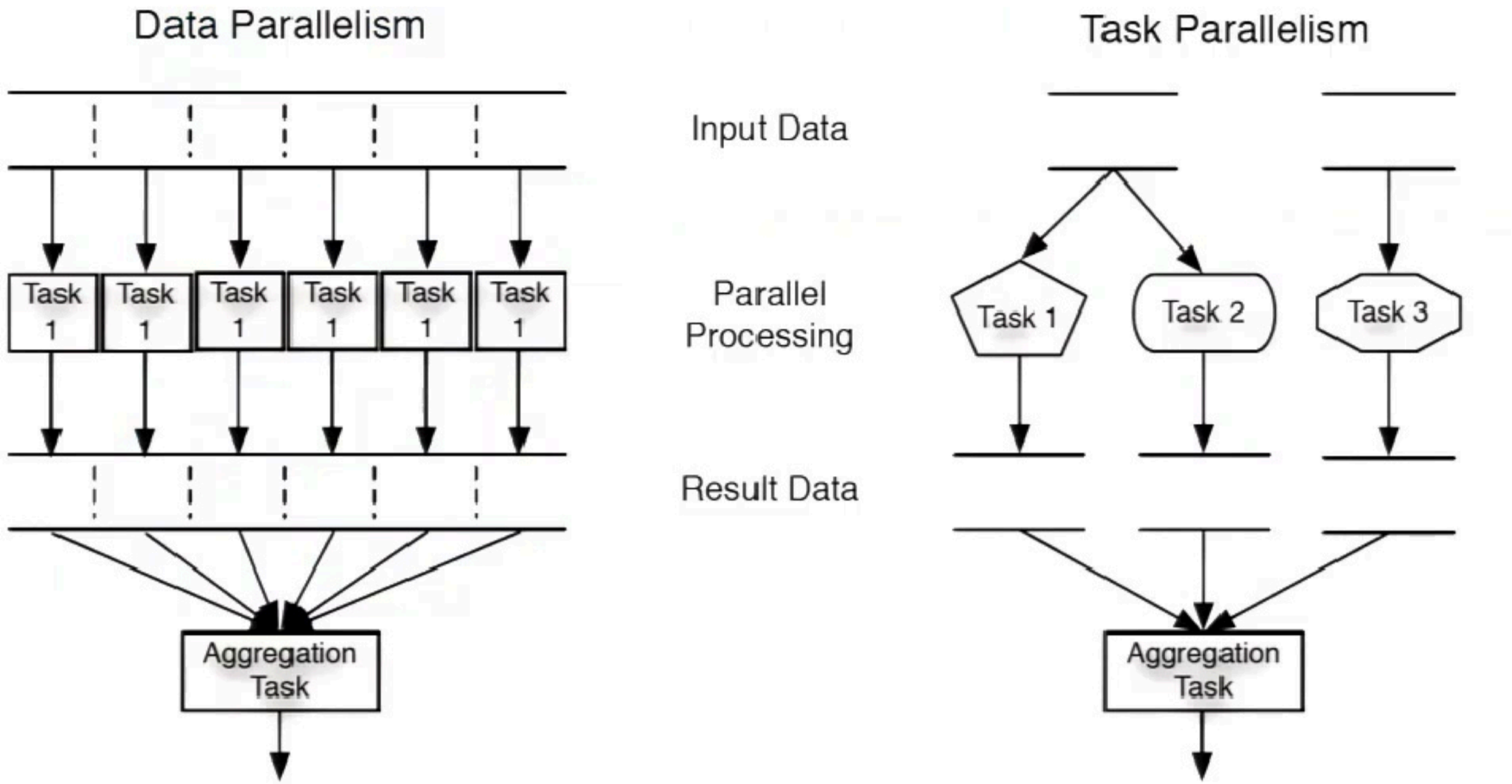
```
u18@slave-node:~$ nvidia-smi
Wed Feb  2 10:32:44 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.86       Driver Version: 470.86       CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Quadro K5200        Off  | 00000000:01:00.0 Off |                  Off |
| 26%   40C    P8    13W / 150W |    168MiB /  8126MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A      5957      G   /usr/lib/xorg/Xorg                 87MiB |
|    0   N/A  N/A      6671      G   /usr/bin/gnome-shell               58MiB |
|    0   N/A  N/A      8841      G   /usr/lib/firefox/firefox           13MiB |
|    0   N/A  N/A     12649      G   /usr/lib/firefox/firefox            1MiB |
+-----------------------------------------------------------------------------+
```

Source : https://www.adlinktech.com/en/gpu-computing

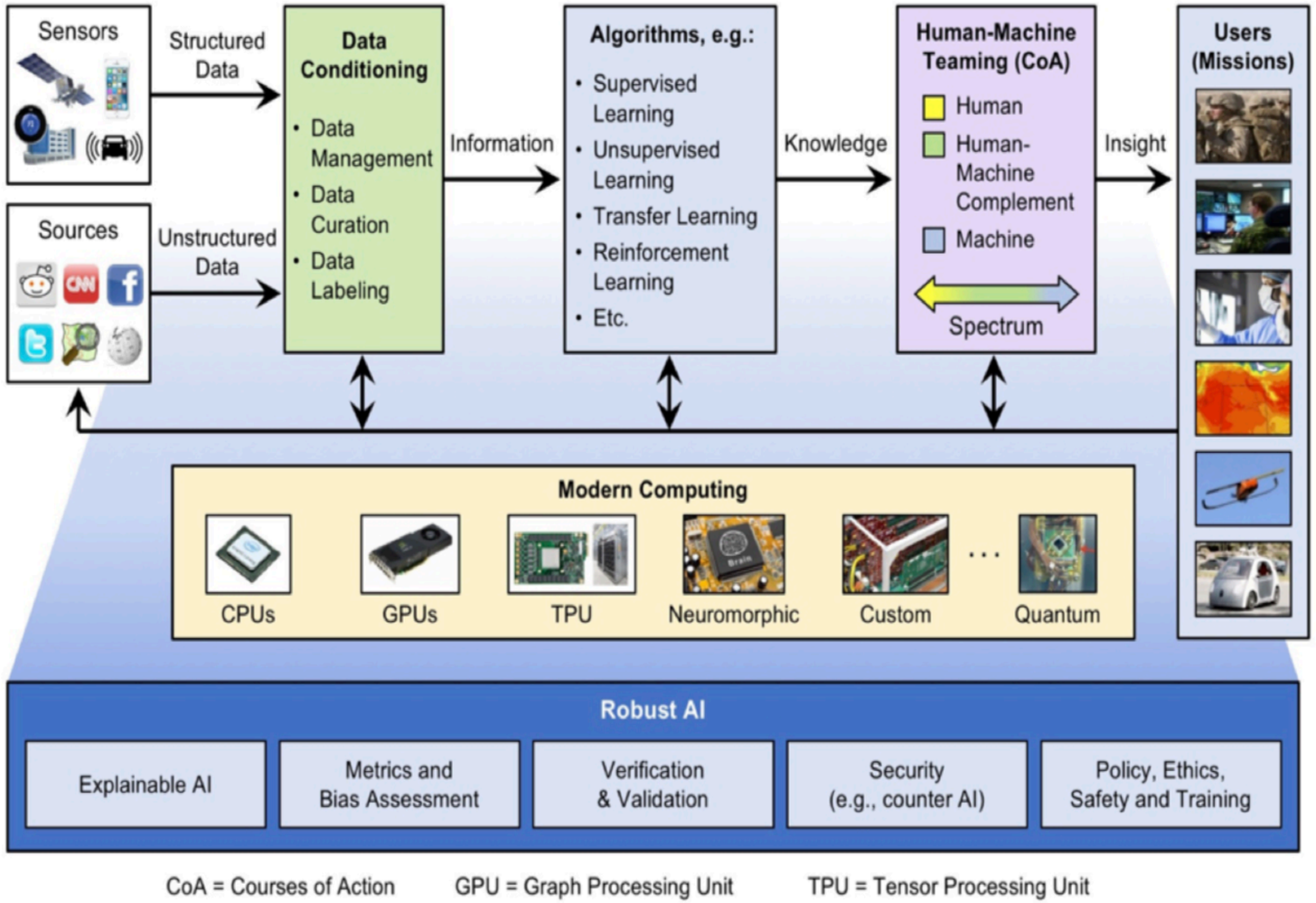
Task and Data Parallelism

Task parallelism vs Data parallelism



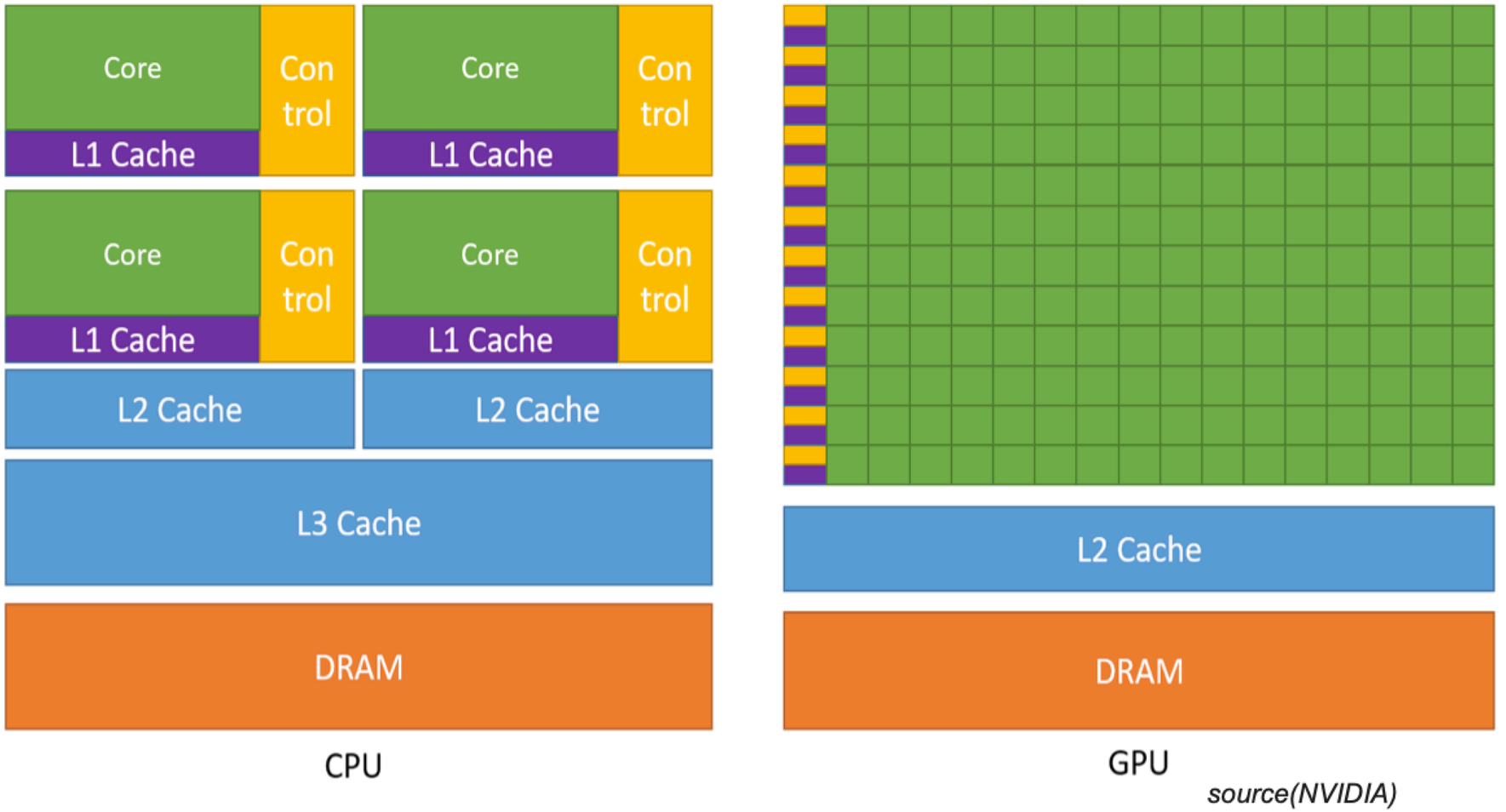
http://master.cmc.msu.ru/files/03_Parallel_Programming_Patterns.pdf

Second View of AI Accelerators



AI Accelerator Survey and Trends
Albert Reuther, Peter Michaleas, Michael Jones, Vijay Gadepally, Siddharth Samsi, and Jeremy Kepner
MIT Lincoln Laboratory Supercomputing Center

GPUs



source(NVIDIA)

Gpus – For the Data Center

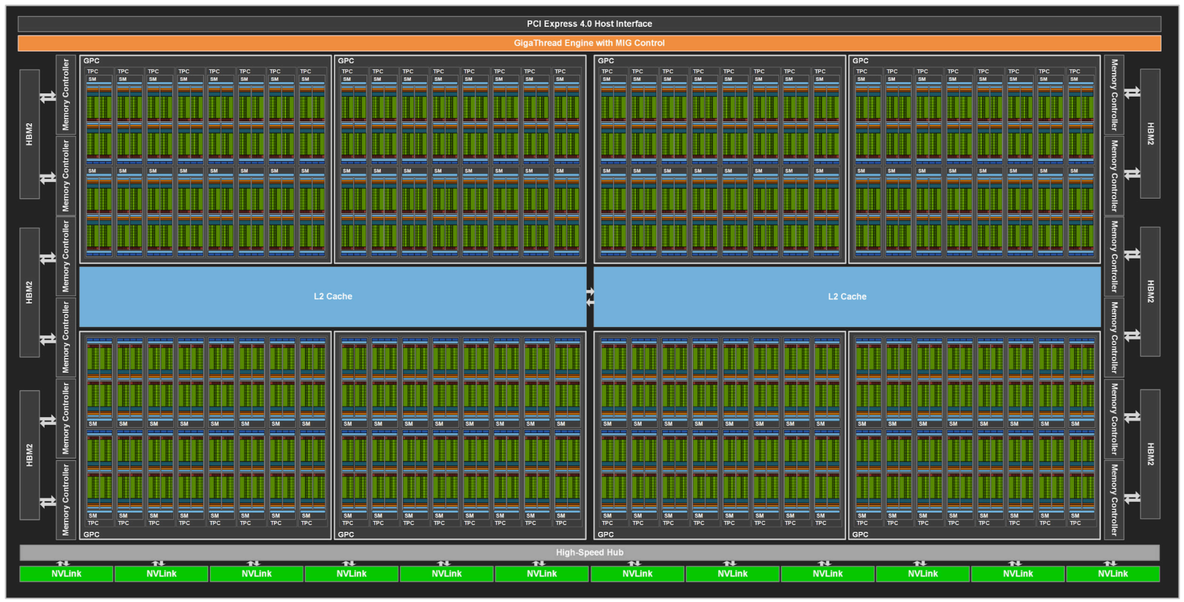


Figure 6. GA100 Full GPU with 128 SMs (A100 Tensor Core GPU has 108 SMs)

For the Edge

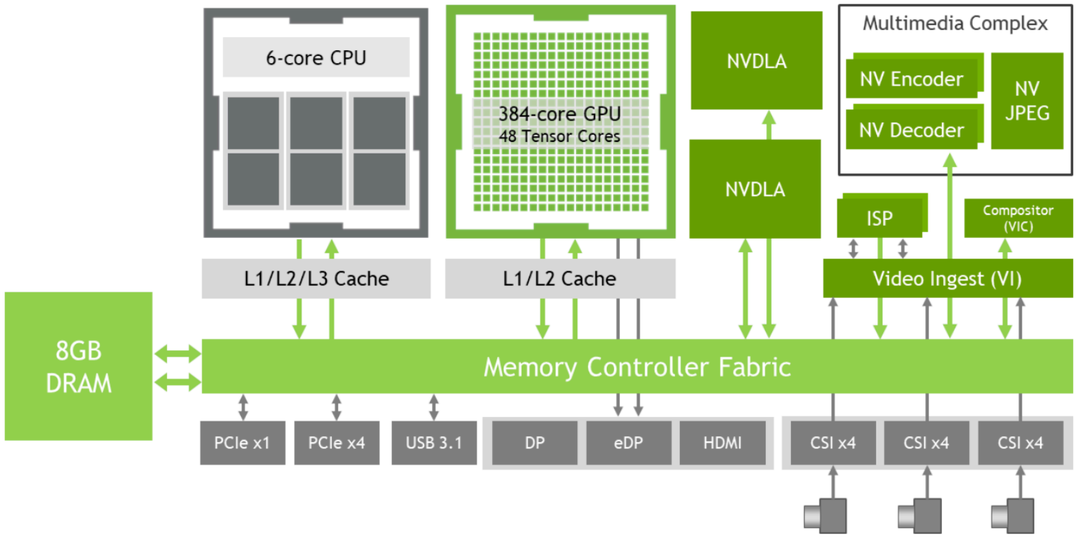
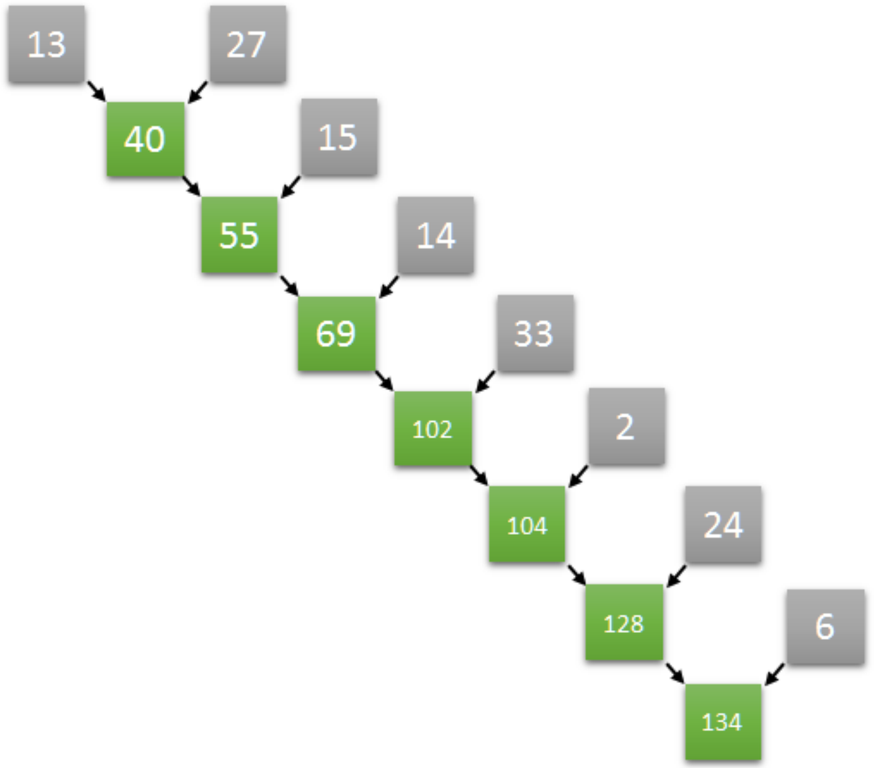


Figure 2: Block diagram of Jetson Xavier NX processor engines including high-speed I/O and memory fabric.

Source : NVIDIA

CPU vs GPU Processing

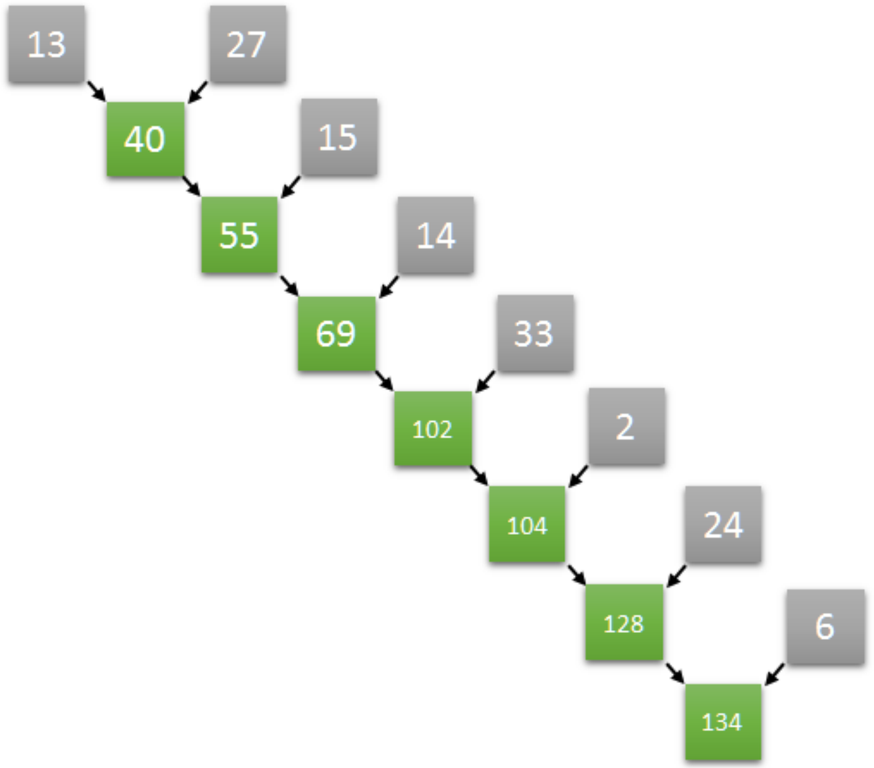
- Sequential reduction : (((((((13+27)+15)+14)+33)+2)+24)+6)



Source: https://www.eximiaco.tech/en/2019/06/10/implementing-parallel-reduction-in-cuda/

CPU vs Parallel vs GPU Processing

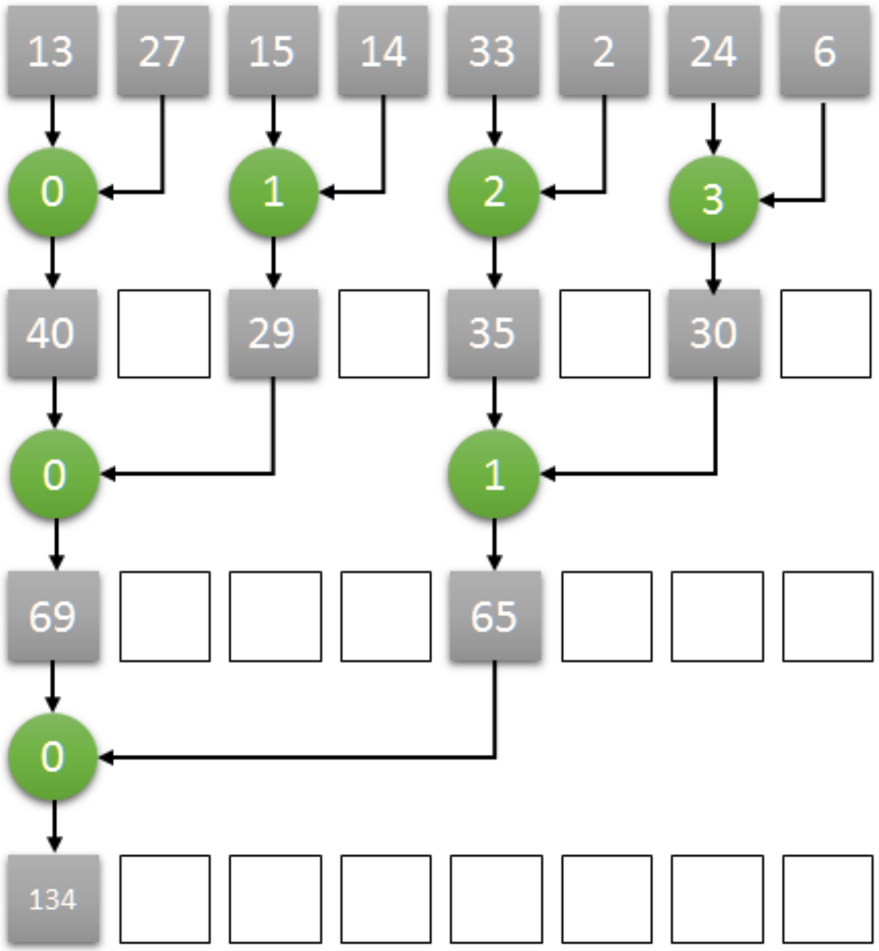
- Parallel reduction : ((13+27)+(15+14))+((33+2)+(24+6))



Source: https://www.eximiaco.tech/en/2019/06/10/implementing-parallel-reduction-in-cuda/

CPU vs Parallel vs GPU Processing

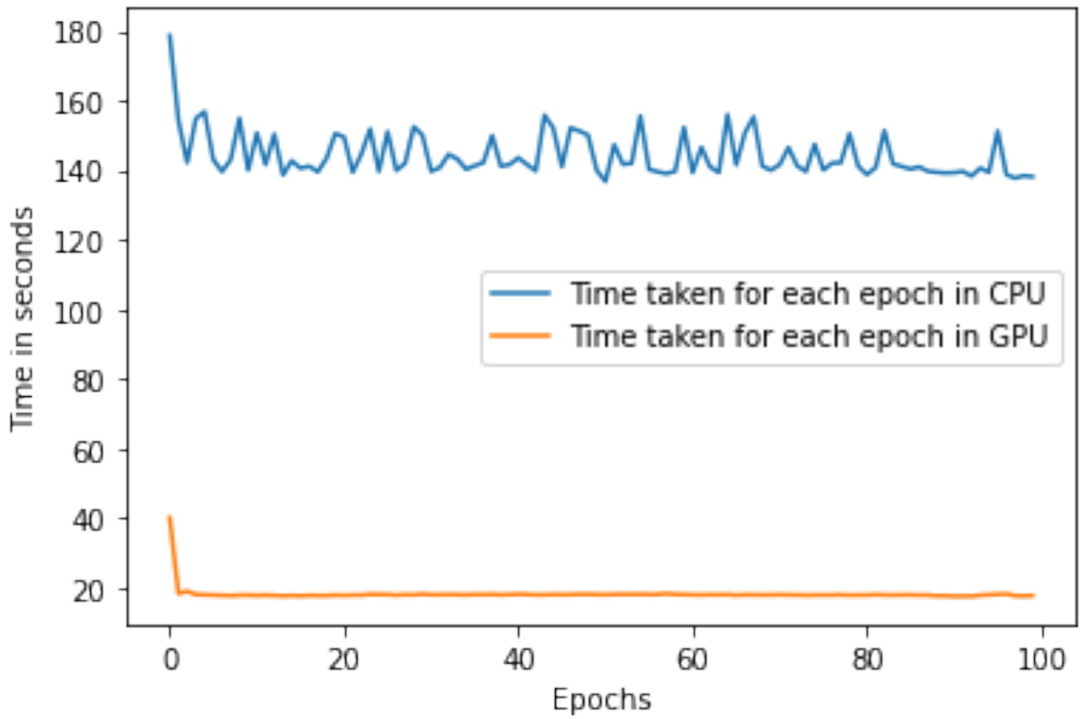
- Parallel Reduction with a GPU



- Assuming N as the number of the elements in an array, we start N/2 threads, one thread for every two elements
- Each thread computes the sum of the corresponding two elements, storing the result at the position of the first one.
- Iteratively, each step:
  - the number of threads halved (for example, starting with 4, then 2, then 1)
  - doubles the step size between the corresponding two elements (starting with 1, then 2, then 4)
- after some iterations, the reduction result will be stored in the first element of the array.

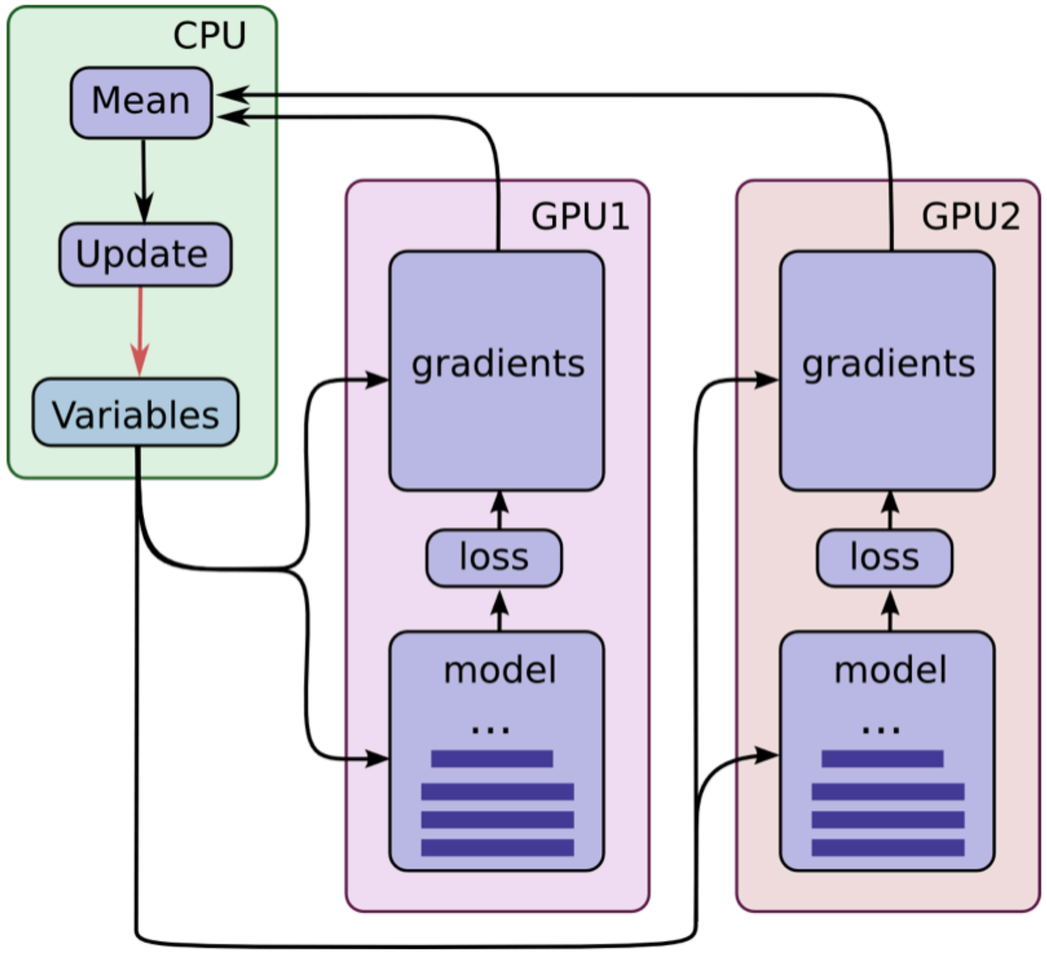
Source: https://www.eximiaco.tech/en/2019/06/10/implementing-parallel-reduction-in-cuda/

CPU vs. GPU Processing – Training time (Classification Problem)



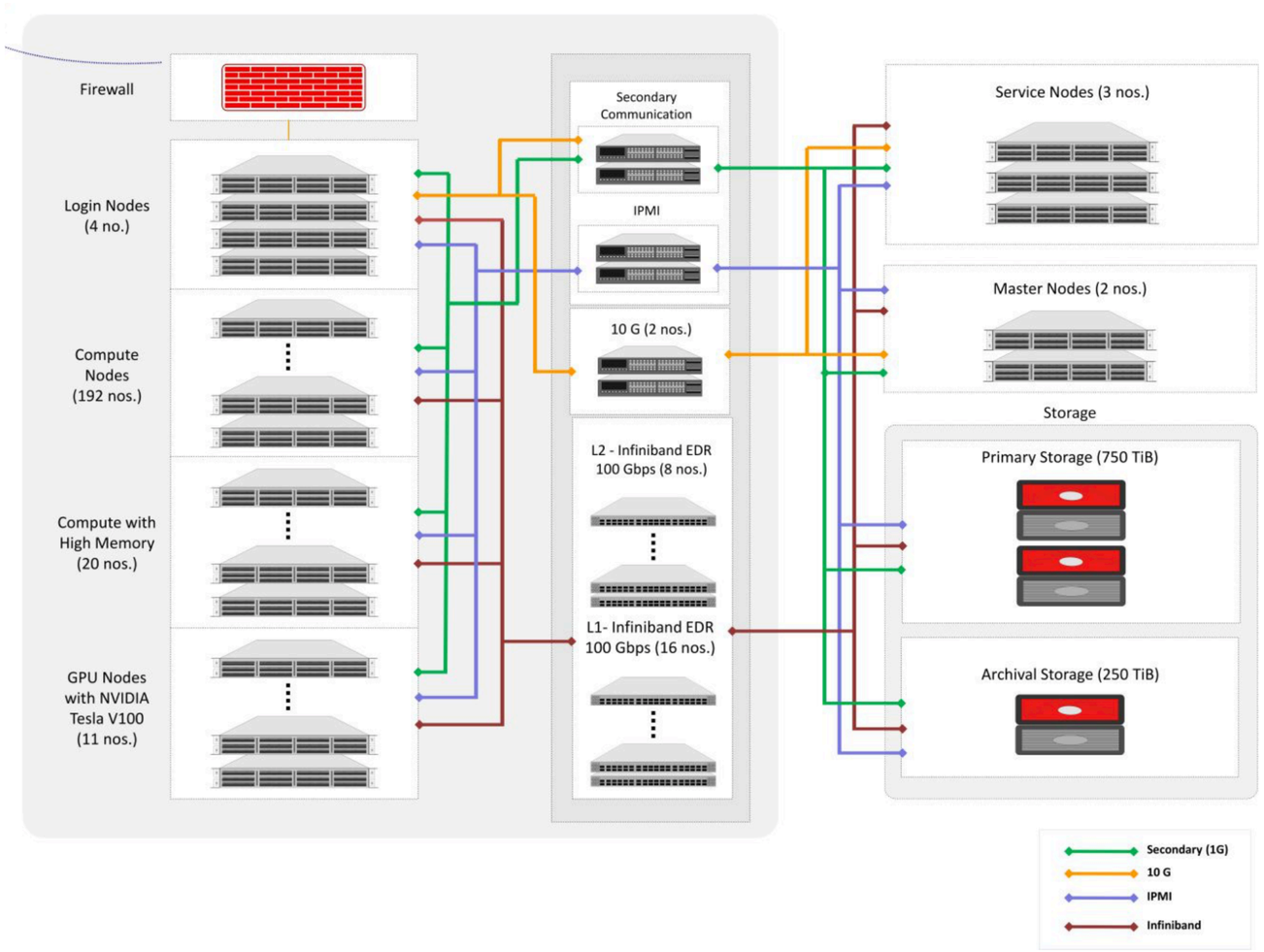
No of Epochs	Time taken for Epochs in CPU (sec)	Time taken for Epochs in GPU(sec)
1	154.26	18.21
2	142.15	18.84
3	154.84	17.92
4	156.78	17.85
5	142.98	17.77
6	139.54	17.69
7	143.02	17.58
8	154.93	17.72
9	139.99	17.76
10	150.66	17.65
11	141.59	17.72
12	150.30	17.65
13	138.58	17.51
14	142.64	17.62
15	140.51	17.52
16	141.09	17.65
17	139.49	17.61
18	143.42	17.61
...	...	...
79	141.08	17.75
80	138.68	17.77
81	140.72	17.88
82	151.33	17.78
83	141.76	17.74
84	141.02	17.80
85	140.29	17.80
86	140.89	17.74
87	139.63	17.73
88	139.36	17.53
89	139.15	17.56
90	139.25	17.41
91	139.62	17.44
92	138.27	17.44
93	140.55	17.72
94	139.36	17.83
95	151.21	17.97
96	138.70	18.07
97	137.66	17.57
98	138.35	17.51
99	138.08	17.61

Train a convolutional neural network on multiple GPU with TensorFlow.



Source:https://wizardforcel.gitbooks.io/tensorflow-examples-aymericdamien/content/6.2_multigpu_cnn.html

PARAM Shivay – Super Computer of 837 TFLOPS



CPU only Compute Nodes

- 192 Nodes
- 7680 Cores
- Compute power of Rpeak 589 TFLOPS
- Each Node with
  - 2 X Intel Xeon Skylake 6148, 20 cores, 2.4 GHz, processors
  - 192 GB memory
  - 480 GB SSD

High Memory Compute Nodes

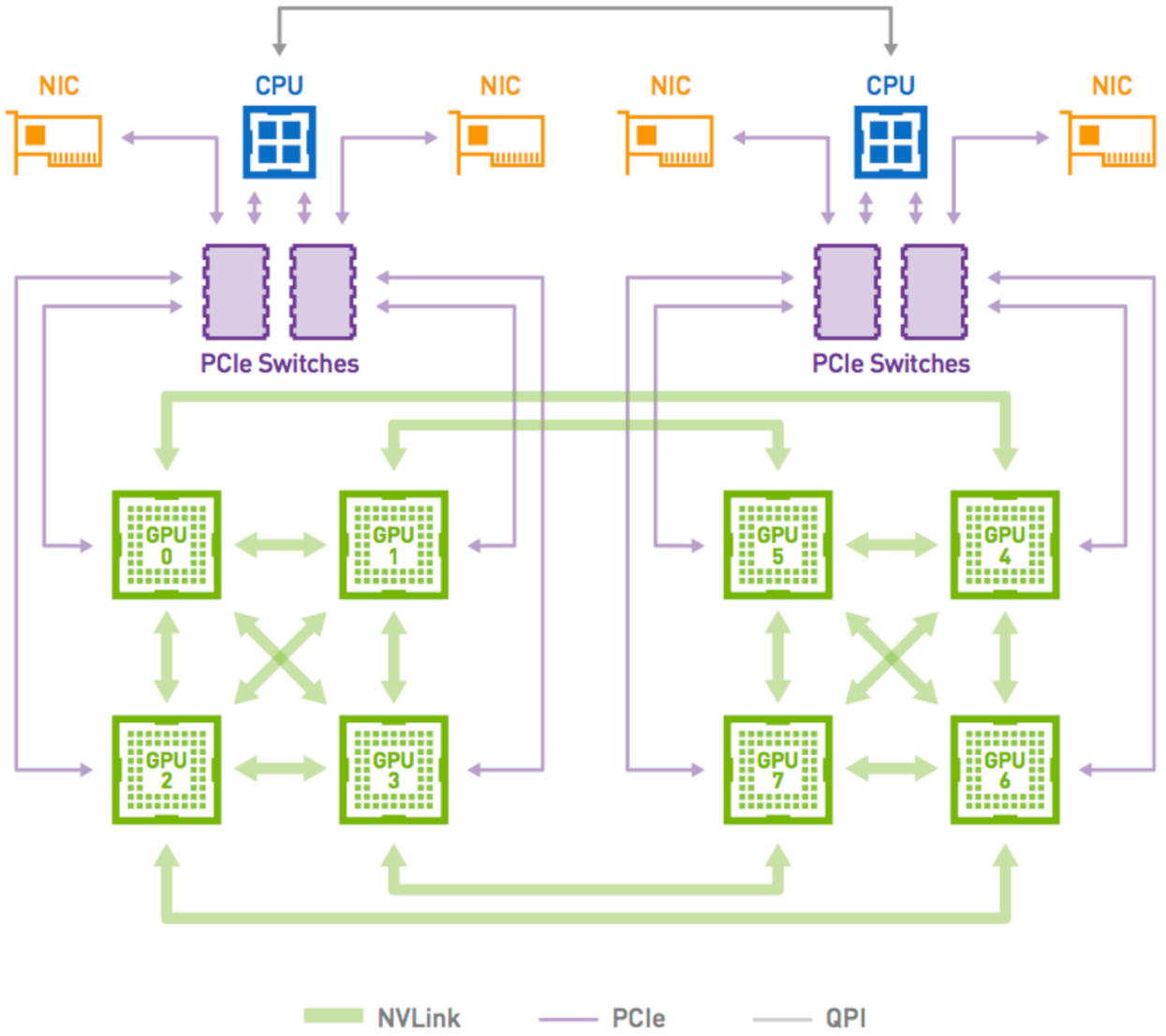
- 20 Nodes
- 800 Cores
- Compute power of Rpeak 61 TFLOPS
- Each Node with
  - 2 X Intel Xeon Skylake 6148, 20 Cores, 2.4 GHz,Processors
  - 768 GB Memory
  - 480 GB SSD

GPU Compute Nodes

- 11 Nodes
- 440 CPU cores
- Compute power of Rpeak 187 TFLOPS
- Each Node with
  - 2 X Intel Xeon Skylake 6148, 20 Cores, 2.4 GHz, Processors
  - 192 GB Memory
  - 480 GB SSD
  - 2xNvidia V100 PCIe accelerator cards each with 5120 CUDA cores,16 GB HBM2

Source:https://nsmindia.in/node/155#5

DGX – I

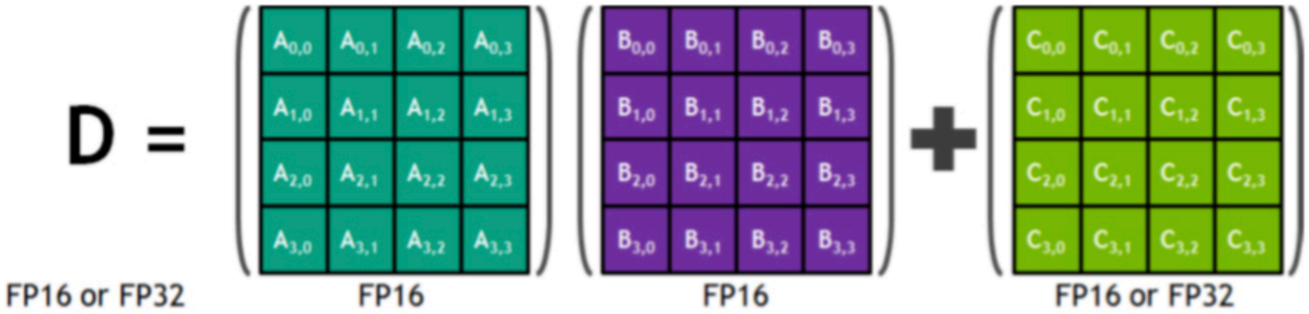


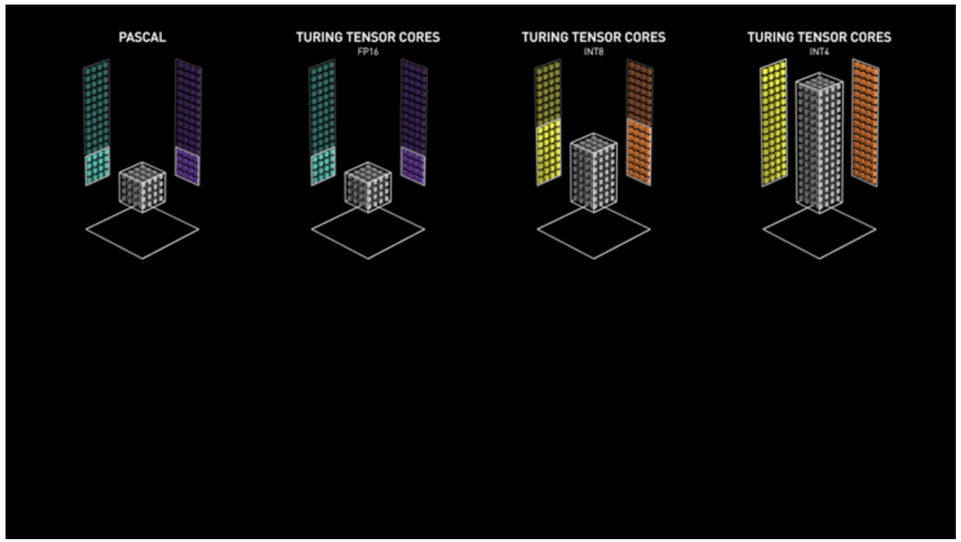
GPUs	8xTesla V100
GPU Memory	256 GB (32 GB/GPU)
CPU	Dual 20-core Intel Xeon E5-2698 v4 2.2 GHz
NVIDIA CUDA Cores	40,960
NVIDIA Tensor Cores (on V100 based systems)	5,120
System Memory	512 GB 2.133 GHz DDR4 RDIMM
Storage	4x1.92 TB SSD RAID-0
Network	Dual 10 GbE

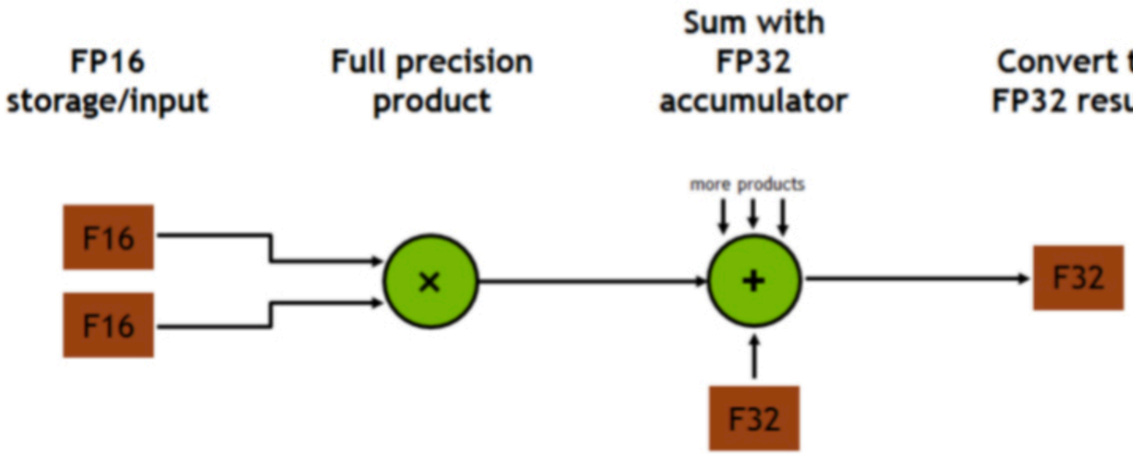
Source:NVIDIA

DGX – I – Tensor Cores

- The Tesla V100 GPU contains 640 Tensor Cores: 8 per SM (Streaming Multiprocessor).
- Each Tensor Core provides a 4x4x4 matrix processing array which performs the operation D = A * B + C, where A, B, C and D are 4x4 matrices. The matrix multiply inputs A and B are FP16 matrices, while the accumulation matrices C and D may be FP16 or FP32 matrices.







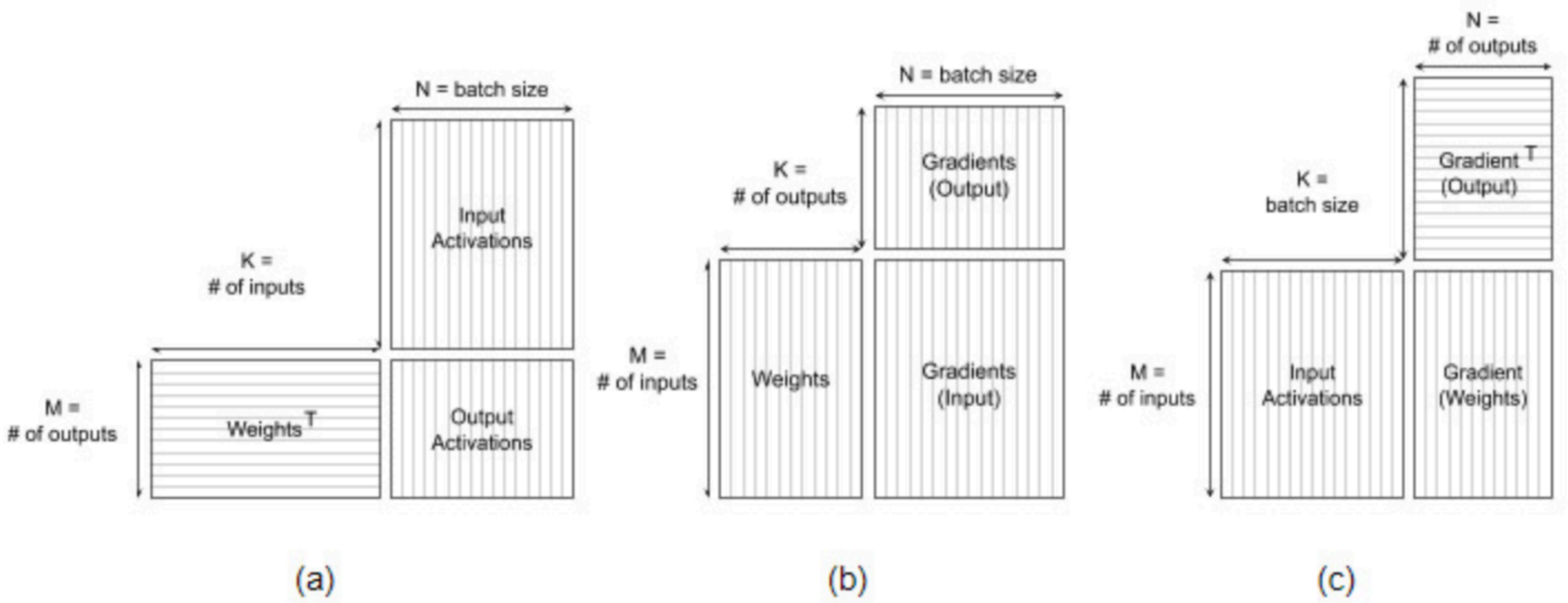
Source:NVIDIA

DGX – I – Tensor Cores

- One arithmetic operation that holds high importance is matrix multiplication.
- Multiplying two 4×4 matrices involves 64 multiplications and 48 additions.
- Convolution and Multiplication are the areas where the new cores shine.
- Typically, the notion is that CUDA cores are slower, but offer more significant precision. Whereas a Tensor cores are lightning fast, however lose some precision along the way.
- All Tensor core does is that it accelerates the speed of matrix multiplication.
- Tensor Cores are able to multiply two fp16 matrices 4x4 and add the multiplication product fp32 matrix (size: 4x4) to accumulator (that is also fp32 4x4 matrix).
- *General Matrix Multiplication (GEMM)*
- Instead of needing to use many CUDA cores and more clocks to accomplish the same task it can be done in a single clock cycle causing a dramatic speed up in machine learning applications.

Source:NVIDIA

DGX – I – Tensor Cores



Equivalent matrix multiplies for (a) forward propagation, (b) activation gradient calculation, and (c) weight gradient calculation of a fully-connected layer.

- Tensor Cores in CUDA Libraries : cuBLAS uses Tensor Cores to speed up GEMM computations (GEMM is the BLAS term for a matrix-matrix multiplication); cuDNN uses Tensor Cores to speed up both convolutions and recurrent neural networks (RNNs)
- CUDA − WMMA (Warp Matrix Multiply-Accumulate)

Source:NVIDIA

DGX – I – Tensor Cores

- DLSS -- Deep Learning Super Sampling
- Basic premise is simple: render a frame at low-ish resolution and when finished, increase the resolution of the end result so that it matches the native screen dimensions of the monitor (e.g. render at 1080p, then resize it to 1400p). That way you get the performance benefit of processing fewer pixels, but still get a nice looking image on the screen.

Source: https://www.techspot.com/article/2049-what-are-tensor-cores/

Demo 2

- Tensor core vs. Gpu cores performance

```
chpc@LAPTOP-TDCSOMDA:~/wmma_tensorcore_sample/project/project$ ./kernel
[*] Initializing Matrix...
[+]    A: 4096 x 4096
[+]    B: 4096 x 4096
[+]    C: 4096 x 4096
[*] Computing D = A * B + C on GPU without Tensor Cores...
[+] GPU(without Tensor Cores) Elapsed Time: 1409.558716 ms
[+] TFLOPS: 0.10
[*] Computing D = A * B + C on GPU with Tensor Cores...
[+] GPU(with Tensor Cores) Elapsed Time: 1386.487549 ms
[+] TFLOPS: 0.10
```

Source: https://github.com/wzsh/wmma_tensorcore_sample

Benefits

- Sustainable
- Speed
- Scalability
- Heterogeneous Architecture
- Overall Efficiency

Thank You