Lab 6 - 2

29 Dec 2022
Parallel Programming

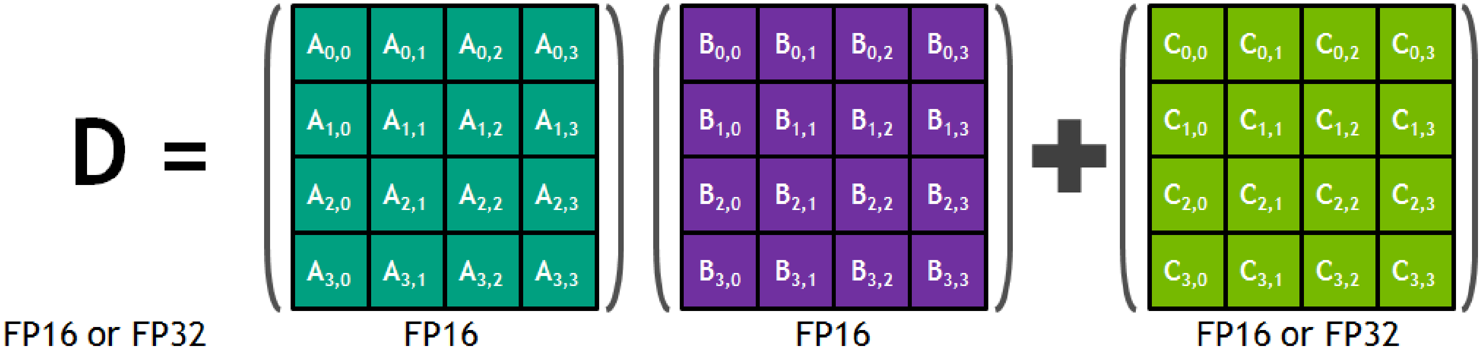
Outline

- Tensor Cores
- NCCL

Tensor Cores

Tensor Cores

- ➢ Specialized execution units designed for mixed precision operation
- ➢ Volta & Turing architecture
- ➢ Each tensor core performs 64 floating point FMA mixed-precision operation per clock



$$D = \left(\begin{matrix} A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\ A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} \end{matrix} \right) \left(\begin{matrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\ B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\ B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\ B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3} \end{matrix} \right) + \left(\begin{matrix} C_{0,0} & C_{0,1} & C_{0,2} & C_{0,3} \\ C_{1,0} & C_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,0} & C_{2,1} & C_{2,2} & C_{2,3} \\ C_{3,0} & C_{3,1} & C_{3,2} & C_{3,3} \end{matrix} \right)$$

CUDA Cores Performance

Volta & Turing

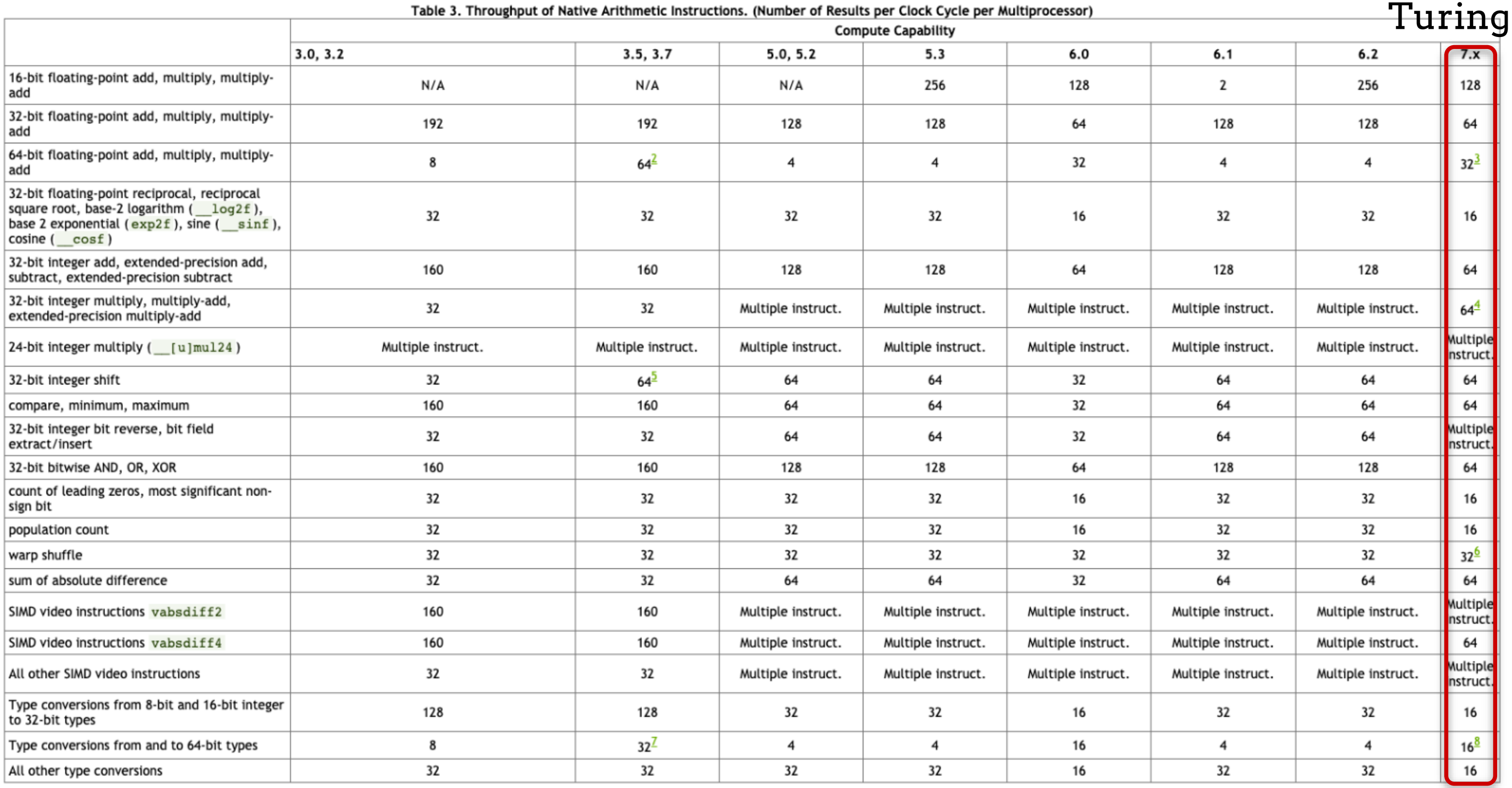
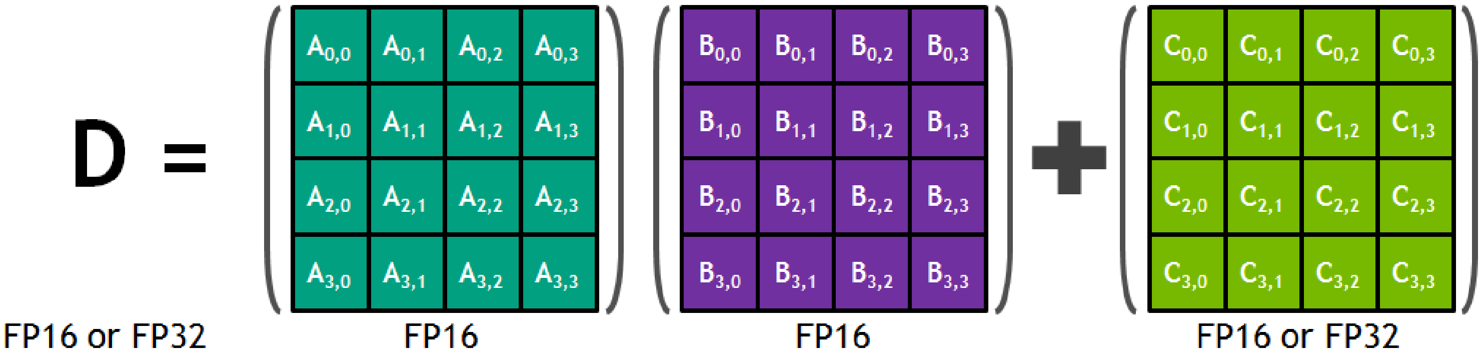


Table 3. Throughput of Native Arithmetic Instructions. (Number of Results per Clock Cycle per Multiprocessor)

	Compute Capability							
	3.0, 3.2	3.5, 3.7	5.0, 5.2	5.3	6.0	6.1	6.2	7.x
16-bit floating-point add, multiply, multiply-add	N/A	N/A	N/A	256	128	2	256	128
32-bit floating-point add, multiply, multiply-add	192	192	128	128	64	128	128	64
64-bit floating-point add, multiply, multiply-add	8	64[2]	4	4	32	4	4	32[3]
32-bit floating-point reciprocal, reciprocal square root, base-2 logarithm (`__log2f`), base 2 exponential (`exp2f`), sine (`__sinf`), cosine (`__cosf`)	32	32	32	32	16	32	32	16
32-bit integer add, extended-precision add, subtract, extended-precision subtract	160	160	128	128	64	128	128	64
32-bit integer multiply, multiply-add, extended-precision multiply-add	32	32	Multiple instruct.	Multiple instruct.	Multiple instruct.	Multiple instruct.	Multiple instruct.	64[4]
24-bit integer multiply (`__[u]mul24`)	Multiple instruct.	Multiple instruct.	Multiple instruct.	Multiple instruct.	Multiple instruct.	Multiple instruct.	Multiple instruct.	Multiple instruct.
32-bit integer shift	32	64[5]	64	64	32	64	64	64
compare, minimum, maximum	160	160	64	64	32	64	64	64
32-bit integer bit reverse, bit field extract/insert	32	32	64	64	32	64	64	Multiple instruct.
32-bit bitwise AND, OR, XOR	160	160	128	128	64	128	128	64
count of leading zeros, most significant non-sign bit	32	32	32	32	16	32	32	16
population count	32	32	32	32	16	32	32	16
warp shuffle	32	32	32	32	32	32	32	32[6]
sum of absolute difference	32	32	64	64	32	64	64	64
SIMD video instructions `vabsdiff2`	160	160	Multiple instruct.	Multiple instruct.	Multiple instruct.	Multiple instruct.	Multiple instruct.	Multiple instruct.
SIMD video instructions `vabsdiff4`	160	160	Multiple instruct.	Multiple instruct.	Multiple instruct.	Multiple instruct.	Multiple instruct.	64
All other SIMD video instructions	32	32	Multiple instruct.	Multiple instruct.	Multiple instruct.	Multiple instruct.	Multiple instruct.	Multiple instruct.
Type conversions from 8-bit and 16-bit integer to 32-bit types	128	128	32	32	16	32	32	16
Type conversions from and to 64-bit types	8	32[7]	4	4	16	4	4	16[8]
All other type conversions	32	32	32	32	16	32	32	16

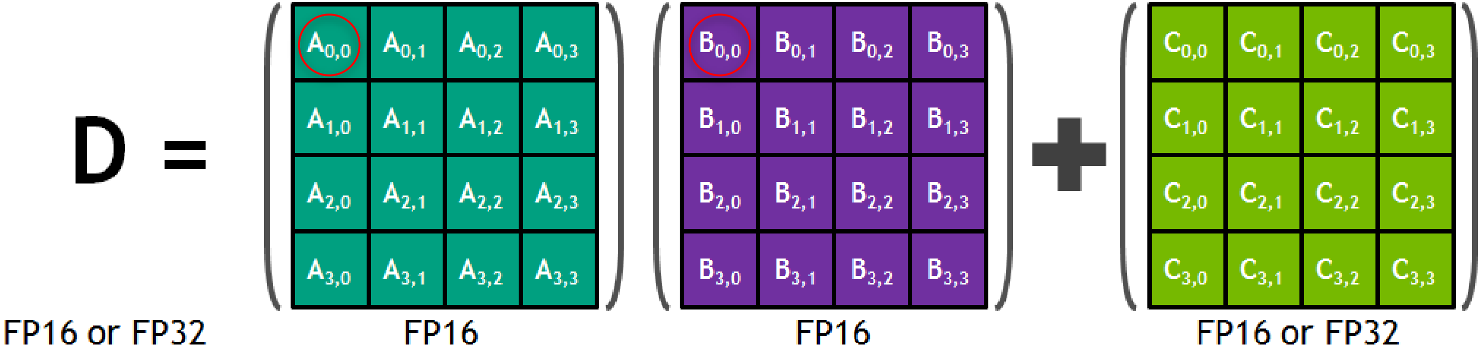
https://docs.nvidia.com/cuda/cuda-c-programming-guide/#arithmetic-instructions

Tensor Cores Performance



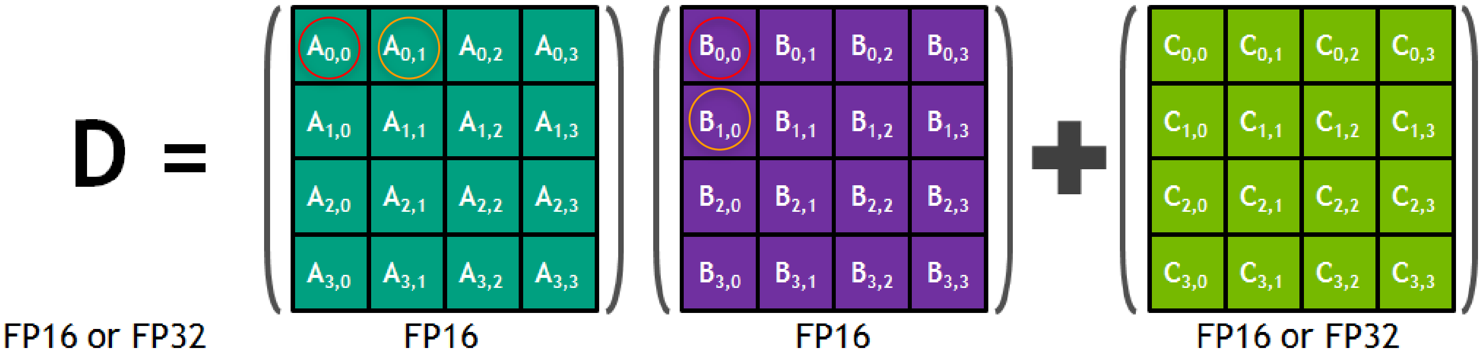
$$D = \begin{pmatrix} A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\ A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix} \begin{pmatrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\ B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\ B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\ B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3} \end{pmatrix} + \begin{pmatrix} C_{0,0} & C_{0,1} & C_{0,2} & C_{0,3} \\ C_{1,0} & C_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,0} & C_{2,1} & C_{2,2} & C_{2,3} \\ C_{3,0} & C_{3,1} & C_{3,2} & C_{3,3} \end{pmatrix}$$

Tensor Cores Performance



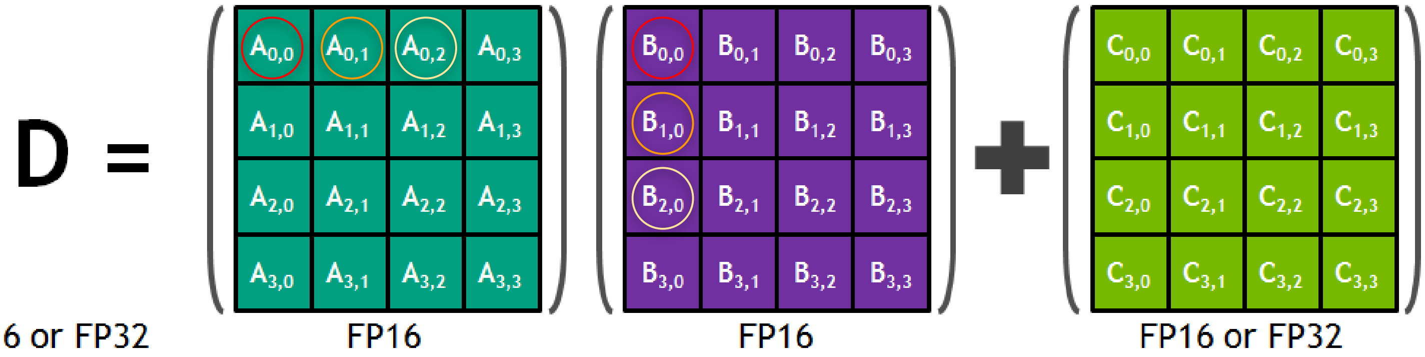
$$\underset{\text{FP16 or FP32}}{D} = \underset{\text{FP16}}{\begin{pmatrix} A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\ A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix}} \underset{\text{FP16}}{\begin{pmatrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\ B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\ B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\ B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3} \end{pmatrix}} + \underset{\text{FP16 or FP32}}{\begin{pmatrix} C_{0,0} & C_{0,1} & C_{0,2} & C_{0,3} \\ C_{1,0} & C_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,0} & C_{2,1} & C_{2,2} & C_{2,3} \\ C_{3,0} & C_{3,1} & C_{3,2} & C_{3,3} \end{pmatrix}}$$

Tensor Cores Performance



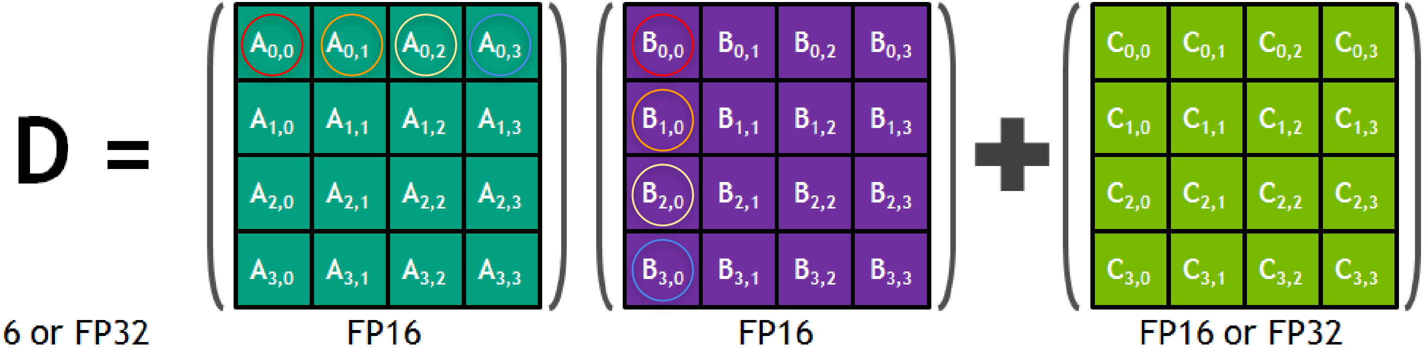
$$\underset{\text{FP16 or FP32}}{D} = \underset{\text{FP16}}{\begin{pmatrix} A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\ A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix}} \underset{\text{FP16}}{\begin{pmatrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\ B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\ B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\ B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3} \end{pmatrix}} + \underset{\text{FP16 or FP32}}{\begin{pmatrix} C_{0,0} & C_{0,1} & C_{0,2} & C_{0,3} \\ C_{1,0} & C_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,0} & C_{2,1} & C_{2,2} & C_{2,3} \\ C_{3,0} & C_{3,1} & C_{3,2} & C_{3,3} \end{pmatrix}}$$

Tensor Cores Performance



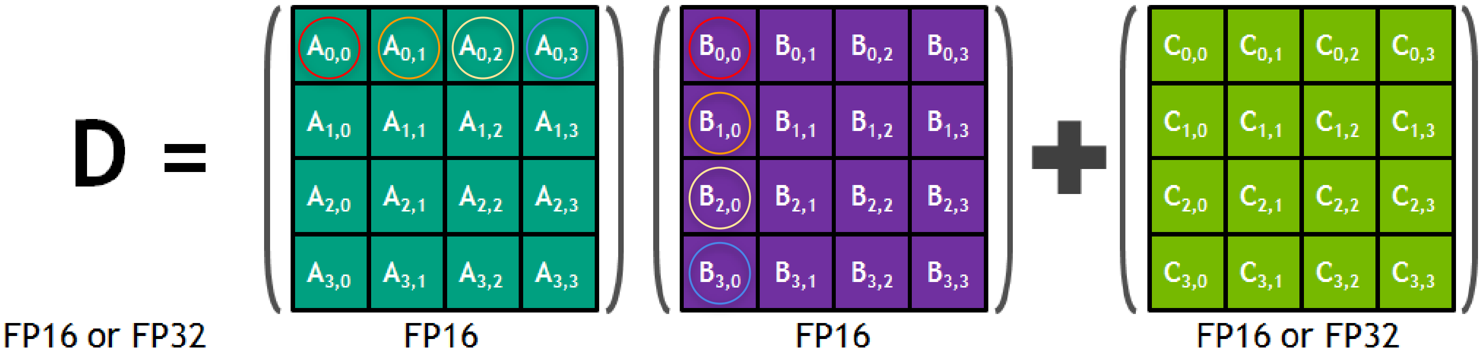
$$\underset{\text{FP16 or FP32}}{D} = \underset{\text{FP16}}{\begin{pmatrix} A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\ A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix}} \underset{\text{FP16}}{\begin{pmatrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\ B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\ B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\ B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3} \end{pmatrix}} + \underset{\text{FP16 or FP32}}{\begin{pmatrix} C_{0,0} & C_{0,1} & C_{0,2} & C_{0,3} \\ C_{1,0} & C_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,0} & C_{2,1} & C_{2,2} & C_{2,3} \\ C_{3,0} & C_{3,1} & C_{3,2} & C_{3,3} \end{pmatrix}}$$

Tensor Cores Performance



$$\underbrace{D}_{\text{FP16 or FP32}} = \underbrace{\begin{pmatrix} A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\ A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix}}_{\text{FP16}} \underbrace{\begin{pmatrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\ B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\ B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\ B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3} \end{pmatrix}}_{\text{FP16}} + \underbrace{\begin{pmatrix} C_{0,0} & C_{0,1} & C_{0,2} & C_{0,3} \\ C_{1,0} & C_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,0} & C_{2,1} & C_{2,2} & C_{2,3} \\ C_{3,0} & C_{3,1} & C_{3,2} & C_{3,3} \end{pmatrix}}_{\text{FP16 or FP32}}$$

Tensor Cores Performance



$$D = \begin{pmatrix} A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\ A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix} \begin{pmatrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\ B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\ B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\ B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3} \end{pmatrix} + \begin{pmatrix} C_{0,0} & C_{0,1} & C_{0,2} & C_{0,3} \\ C_{1,0} & C_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,0} & C_{2,1} & C_{2,2} & C_{2,3} \\ C_{3,0} & C_{3,1} & C_{3,2} & C_{3,3} \end{pmatrix}$$

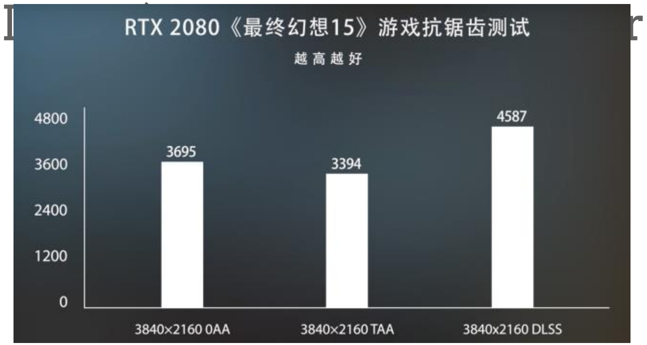
For each element, there are 4 multiplications and 4 addition (4 FMA)
*The performance per tensor core is 16 * 4 FMA per clock*

Performance Comparison (V100)

	Tensor Core	CUDA Core
Cores Per SM	8	64
operations throughput per SM per clock	64 x 8 x 2 = 1024	128

Use Cases

- Tensor Cores are already supported by many deep learning framework
  - Tensorflow
  - Pytorch
  - MXNet
  - Caffe2
  - CNTK
  - https://docs.nvidia.com/deeplearning/sdk/mixed-precision-training/index.html
- I r Sampling)



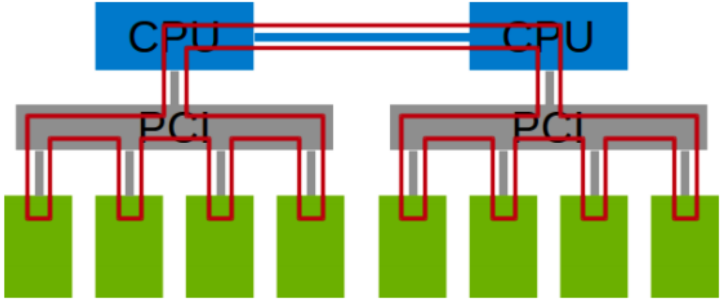
https://youtu.be/-Wg4jqp2cbc?t=795

NCCL

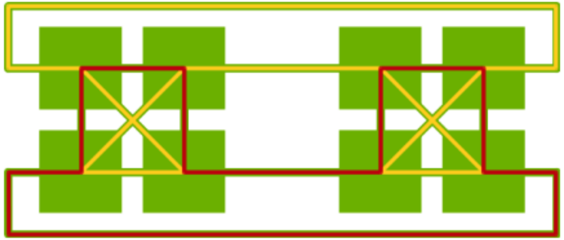
NCCL

- ➢ NVIDIA Collective Communications Library
- ➢ Pronounced “Nickel”
- ➢ Optimized collective communication library between CUDA devices
- ➢ Topology-aware
- ➢ Portability

Topology



PCIe / QPI : 1 unidirectional ring

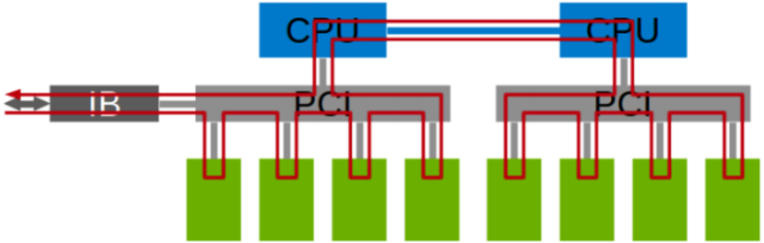


DGX-1 : 4 unidirectional rings

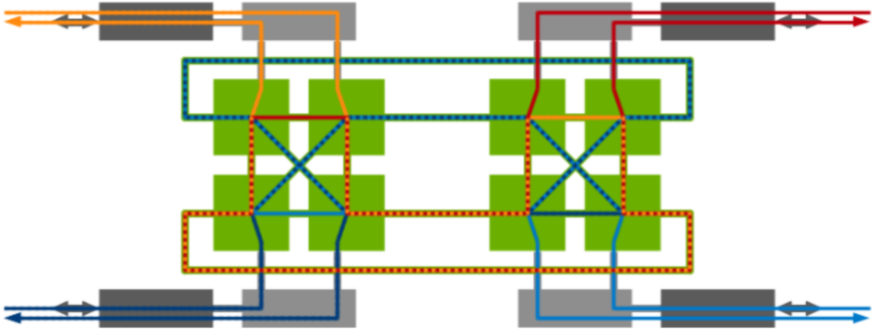
Various machines/motherboard have different topology

NCCL 2.0

- NCCL 1.0 is released in 2015
  - Limited to intra-node (8 cards)
- NCCL 2.x
  - Multi-node support
  - Improve the performance

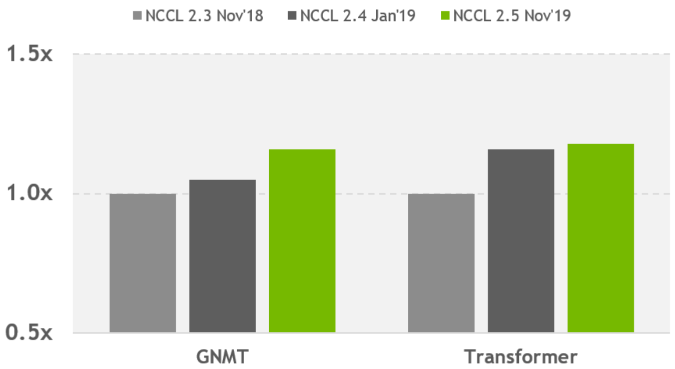


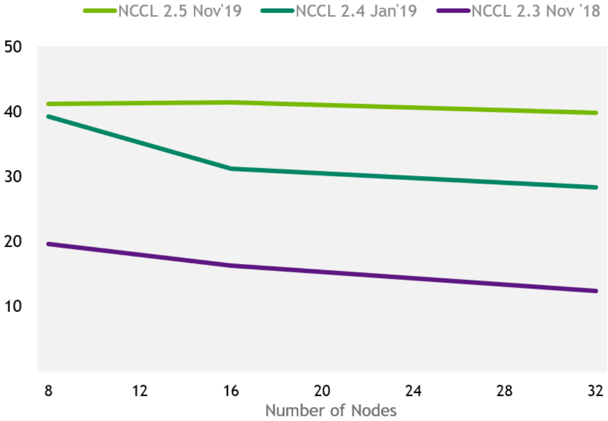
PCIe, Infiniband

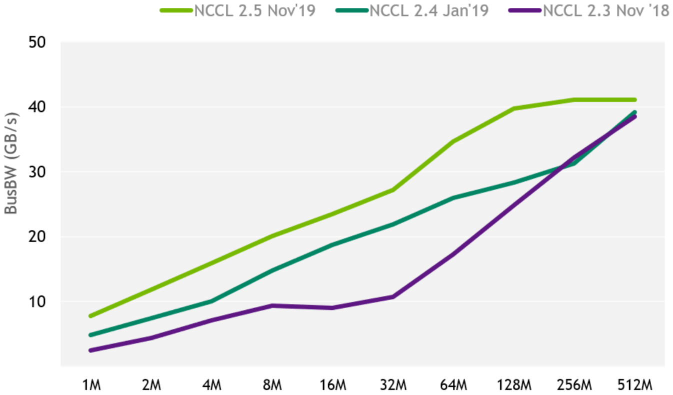


DGX-1 : NVLink, 4x Infiniband

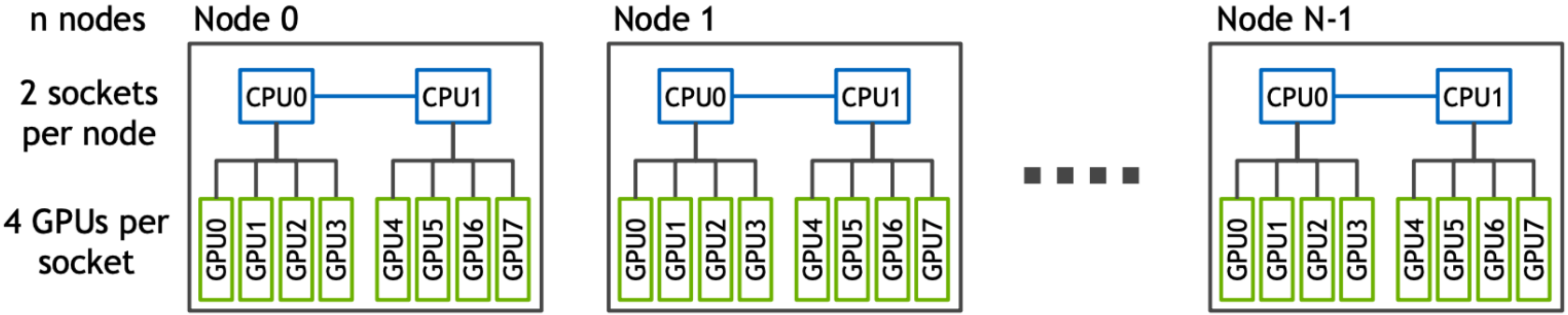
NCCL 2.X


Over18% speedup in a year

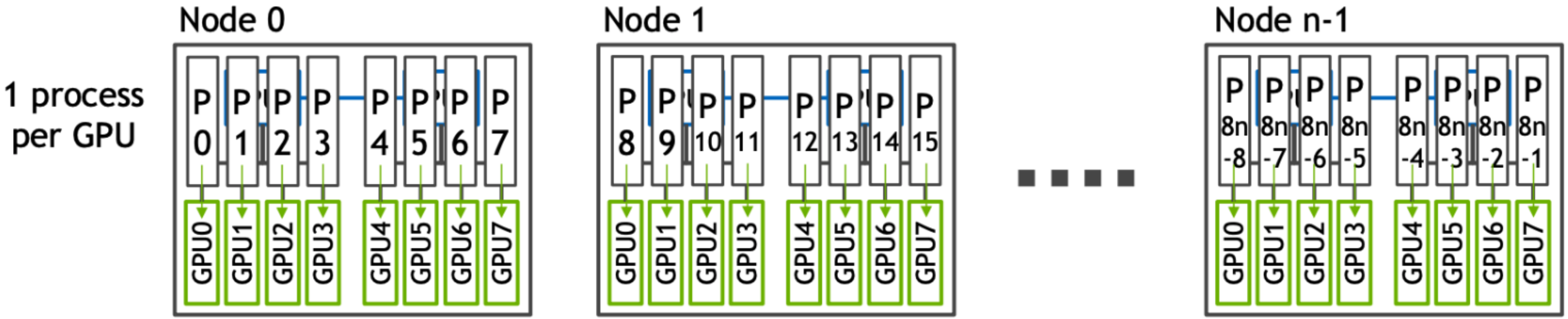

Allreduce bandwidth scaling (128MiB)


Allreduce Bandwidth

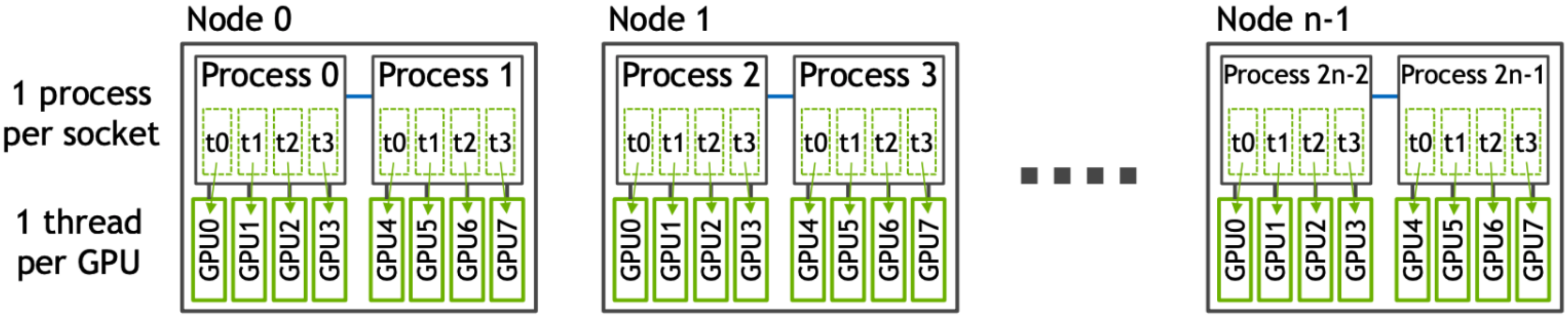
Processes, Threads and GPUs



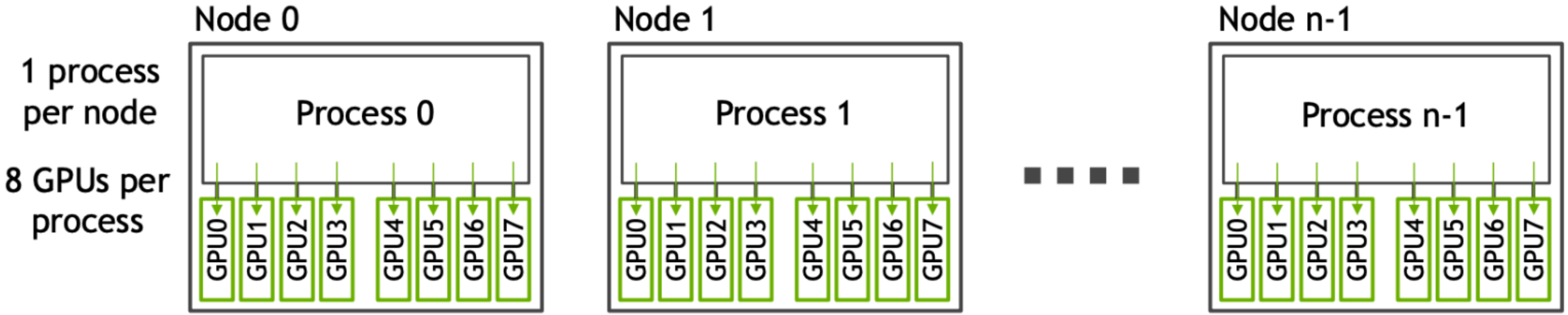
Processes, Threads and GPUs



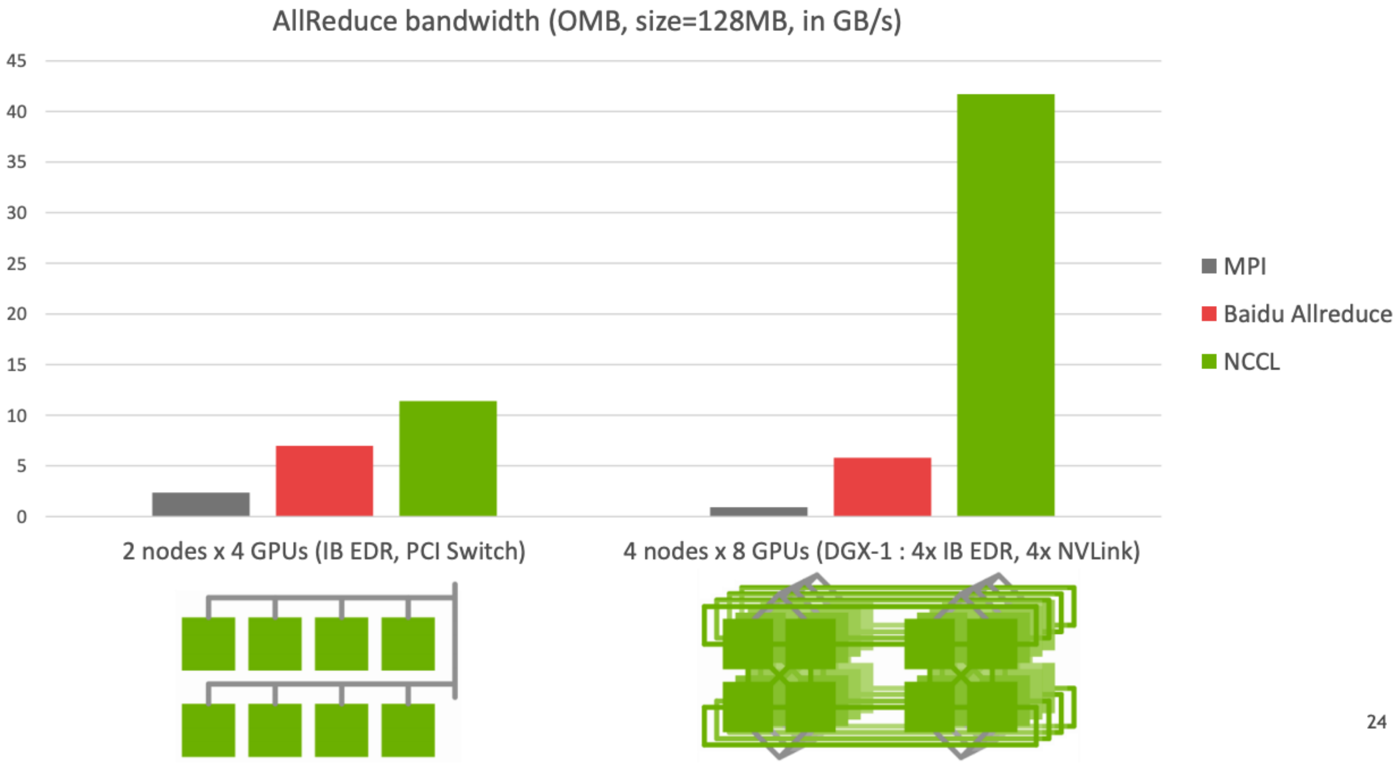
Processes, Threads and GPUs



Processes, Threads and GPUs



Performance



Reference

- CUDA Programming Guide
- Turing White Paper
- Volta White Paper
- 即刻灣 RTX 評測
- nccl-test
- NCCL 1.0 Slides
- NCCL 2.0 Slides
- NCCL Download