

Lecture 15

GPU Architectures

이재진

서울대학교 데이터사이언스대학원

서울대학교 공과대학 컴퓨터공학부

<http://aces.snu.ac.kr/~jlee>



Key Performance Factors

- Parallelism
 - Amount data and computation processed at a time
- Clock frequency
 - Processing speed
- Memory bandwidth
 - Amount of data transferred at a time
- Memory latency
 - Time for each data element to be transferred
- Different applications are sensitive to different performance factors
- Different architectures address these factors in different ways



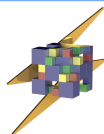
CPUs with regards to the Performance Factors

- Parallelism
 - CPUs have multiple cores, where each has ILP
- Clock frequency
 - CPUs maximize clock frequency
 - But, power wall
- Memory bandwidth
 - DDR memory has limited bandwidth
- Memory latency
 - DDR memory has long latency
 - CPUs strive to hide the latency through:
 - On-chip caches to stage data
 - Out-of-order execution
 - Simultaneous multithreading



Accelerators

- Graphics Processing Units (GPUs) have evolved for the gaming market
 - A GPU has many number-crunching cores
 - GPU vendors, such as NVIDIA and AMD, have tailored existing GPU architectures to the HPC market
 - Software ecosystem is important
- GPUs now firmly established in HPC industry



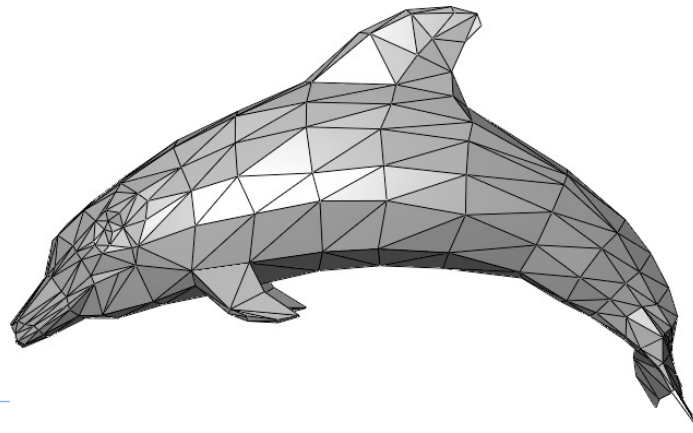
GPUs with regards to the Performance Factors

- Parallelism
 - GPUs dedicate much more space to compute at the expense of caches, controllers, etc.
 - Much higher extent of parallelism than CPUs (many simple cores)
- Clock frequency
 - GPUs typically have lower clock-frequency than CPUs
 - But, GPUs achieve high performance through parallelism
- Memory bandwidth
 - GPUs use high bandwidth memory
 - GDDR or HBM2 stacked memory
- Memory latency
 - Similar to DDR
 - GPUs hide latency through hardware multithreading



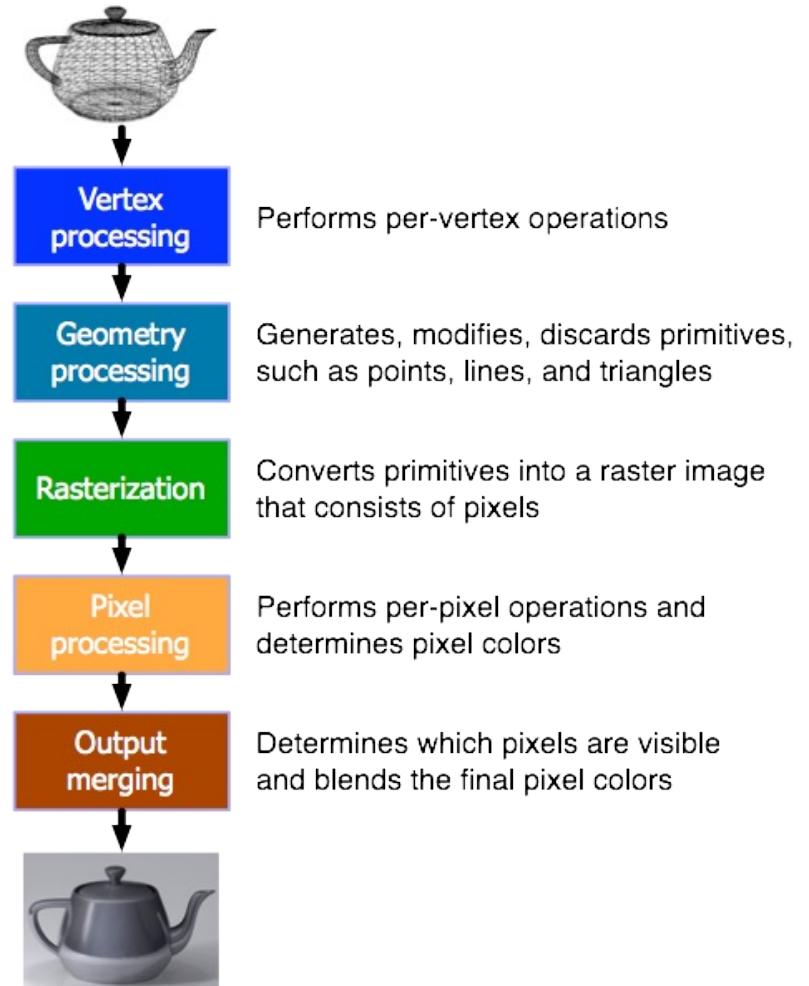
Rendering

- The process of generating an image from a 3D model
- 3D modeling
 - The process of developing a mathematical representation (i.e., model) of any three-dimensional surface of an object
 - Polygon or triangle meshes



Rendering (Graphics) Pipeline

- Generates 2D images from 3D models
- Supported by commodity hardware
- Some stages are programmable
 - Vertex
 - Geometry
 - Pixel
 - ...

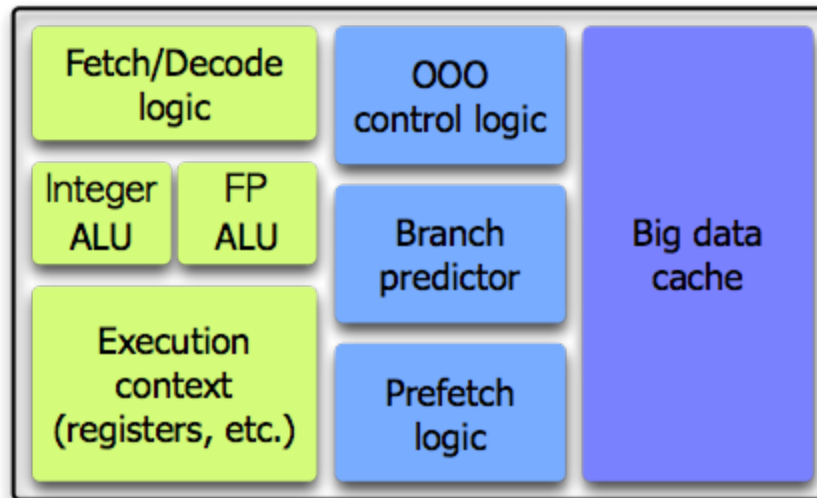


Shaders

- Computer program that runs on the GPU and its purpose is to execute one of the programmable stages of the rendering pipeline

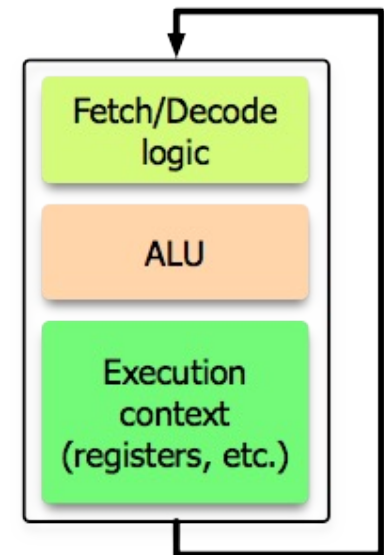


General-purpose CPU



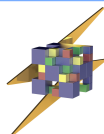
Shader Cores

- Very simple, programmable
- No architecture components that make a single instruction stream run fast
- Logical graphics pipeline
 - Vertex shader, Geometry shader, Pixel shader, etc.
 - Graphics pipeline stages are implemented by visiting the programmable shader core several times



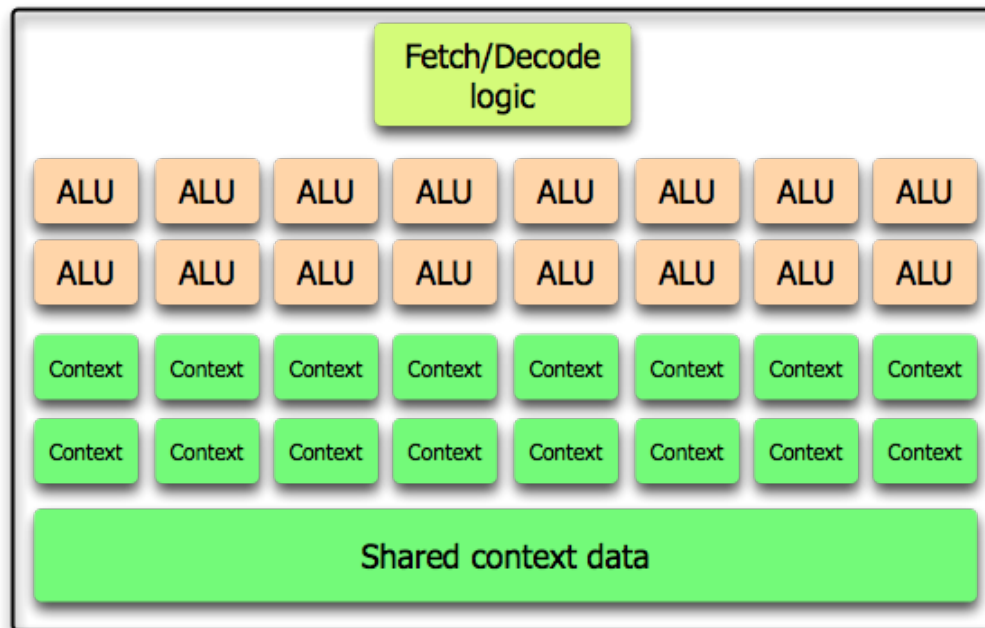
Characteristics of GPU Applications

- Data independence between triangles and pixels
- Millions of triangles and pixels
- Can perform massively parallel processing



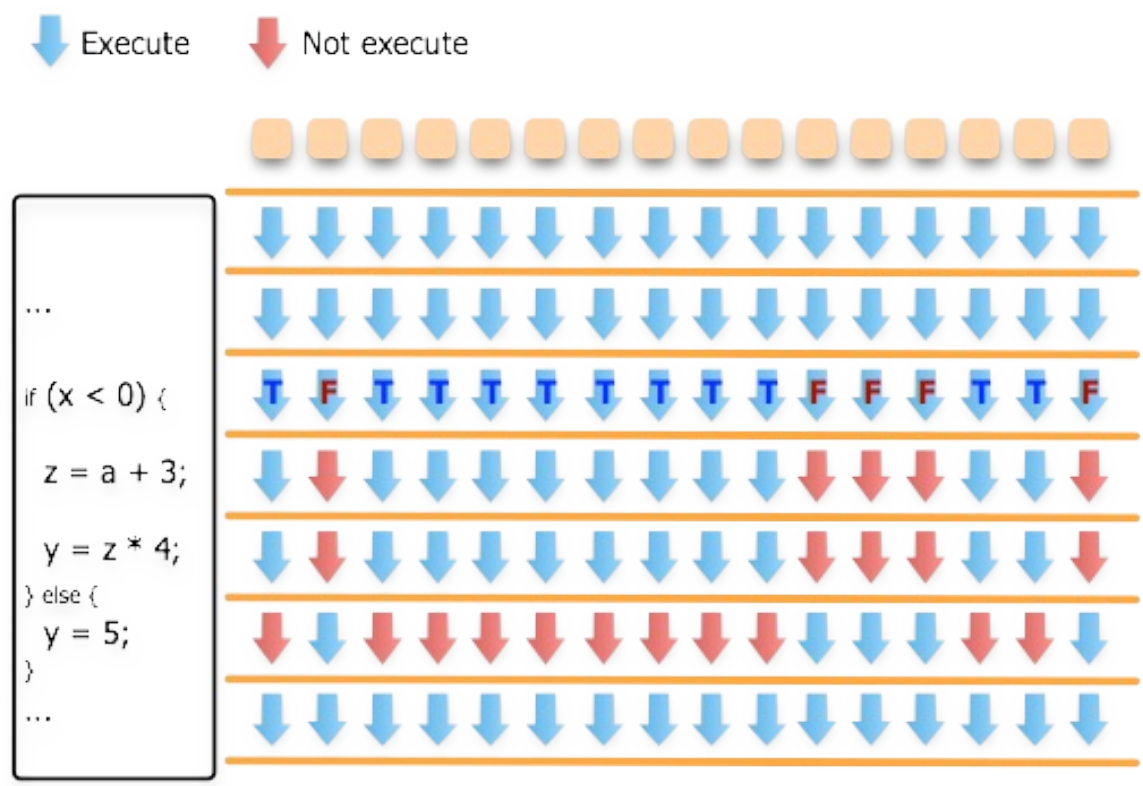
Exploiting Massive Parallelism

- Streaming multiprocessors
 - SIMD processing across many ALUs
 - A single program counter across multiple ALUs
- SIMT
 - Single Instruction, Multiple Thread



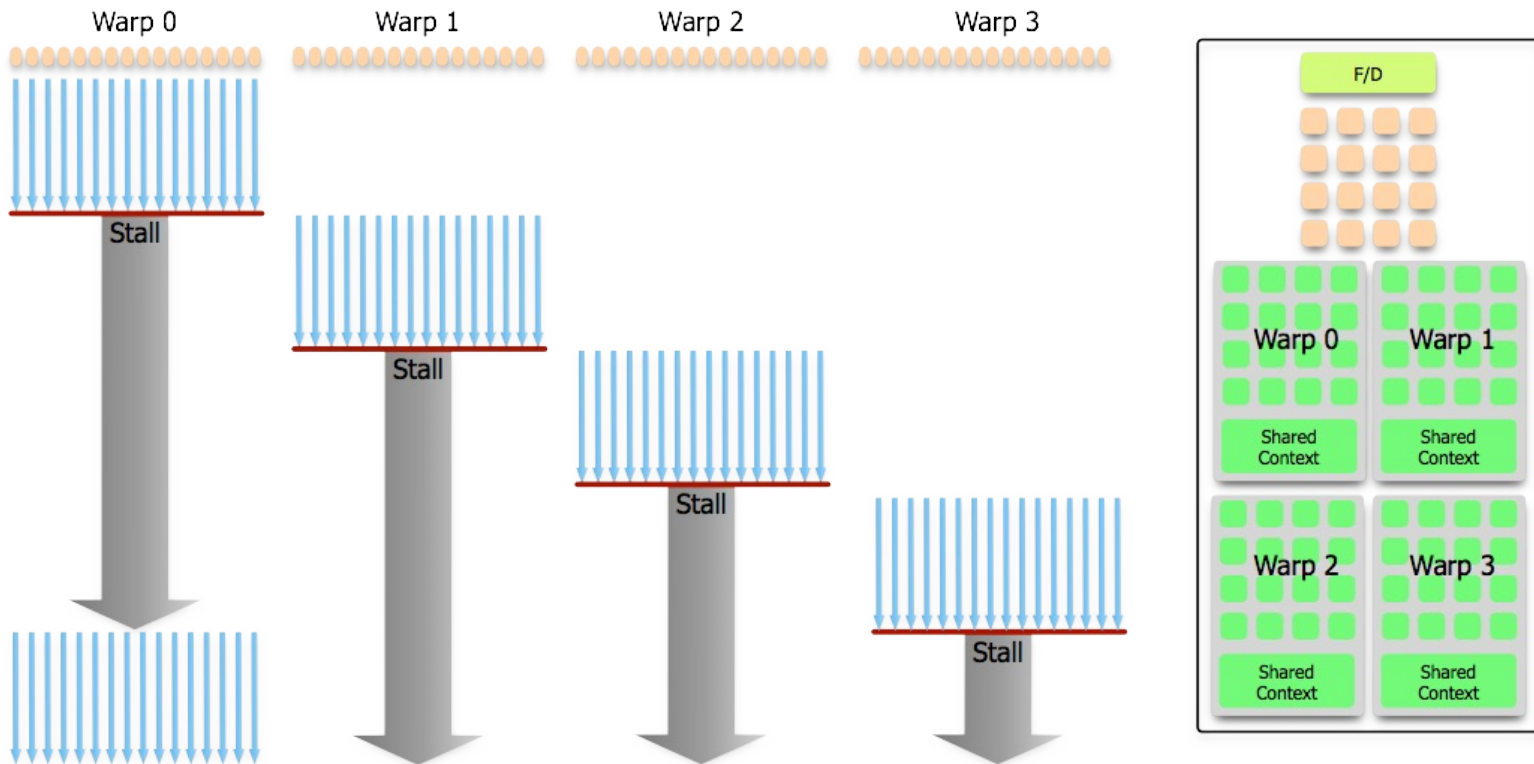
Executing Branches

- Conditional execution in a warp
- Hardware automatically handles divergence



Hardware Context Switch

- To avoid stalls caused by high latency operations
- A single SM can run more than one warp
 - A warp is a group of instruction execution instances and is the unit of context switch
 - All threads in a warp execute the same instruction at a time

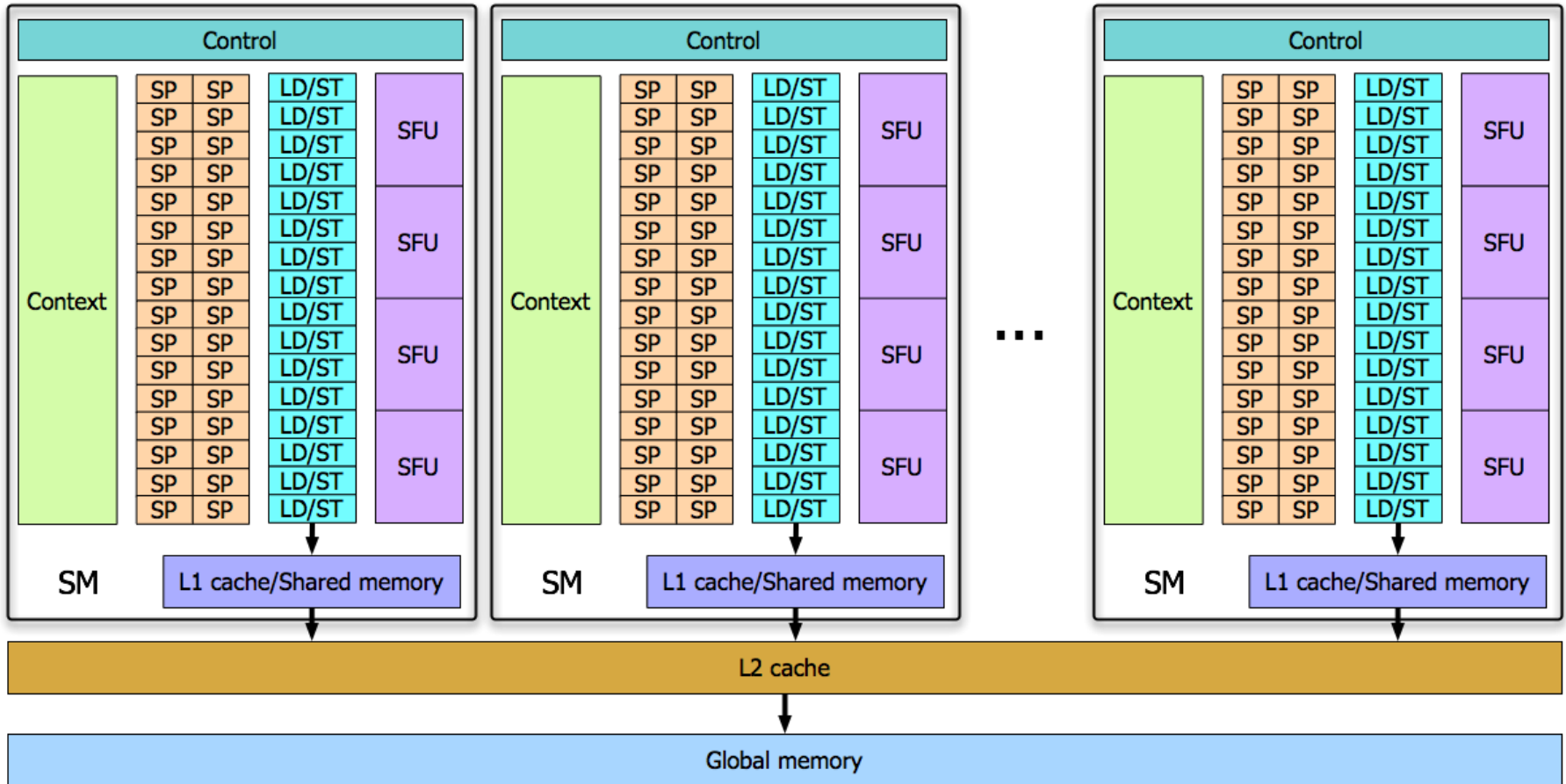


Multiple Streaming Multiprocessors

- Different streaming multiprocessors (SMs) execute different shaders
- N SMs
 - N simultaneous instruction streams



The Overall Picture



GPU Summary

- Exploits massive data parallelism
 - Many simple compute units
 - SIMD (a single program counter)
 - Many threads and no synchronization between threads (for graphics applications)
- Hardware context switch
 - To tolerate high latencies



Tensor Cores

- Tensor Cores give the Tesla V100 accelerator a peak throughput 12 times the 32-bit floating point throughput of the previous-generation Tesla P100
 - Enable AI programmers to use mixed precision to achieve higher throughput without sacrificing accuracy
- Tensor Cores are already supported for DL training in many DL frameworks including TensorFlow, PyTorch, MXNet, and Caffe2



Tensor Cores (cont'd)

- Programmable matrix-multiply-and-accumulate units
- The Tesla V100 GPU contains 640 Tensor Cores: 8 per SM
- Each Tensor Core provides a $4 \times 4 \times 4$ matrix processing array which performs the operation $D = A \times B + C$
 - A, B, C and D are 4×4 matrices
 - Inputs A and B are FP16 matrices
 - The accumulation matrices C and D may be FP16 or FP32 matrices

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

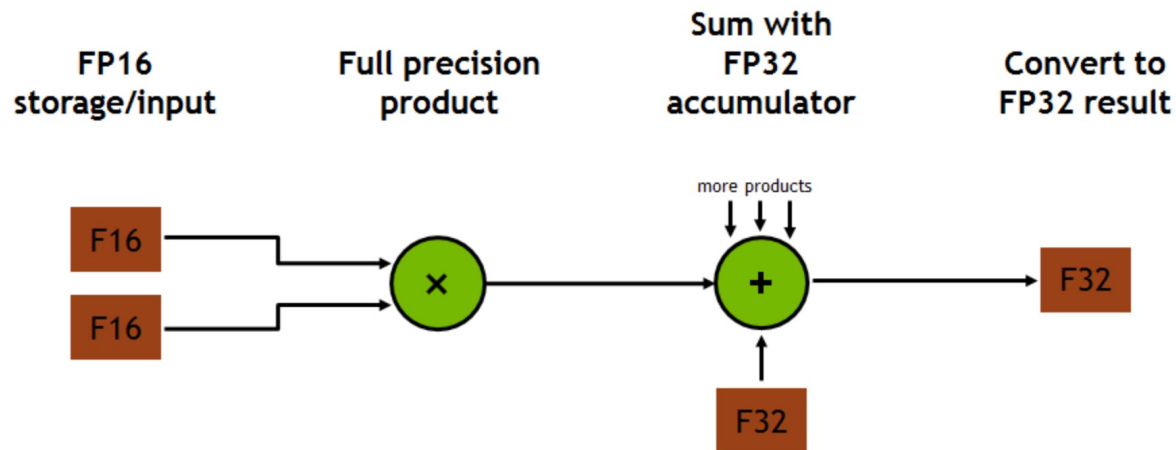
FP16 or FP32
FP16
FP16
FP16 or FP32

<https://developer.nvidia.com/blog/programming-tensor-cores-cuda-9/>



Architecture

- Each Tensor Core performs 64 floating point FMA mixed-precision operations per clock
 - Tensor Cores operate on FP16 input data with FP32 accumulation
 - FP16 input multiply with full-precision product
 - FP32 accumulate



<https://developer.nvidia.com/blog/programming-tensor-cores-cuda-9/>



Performance

- 8 Tensor Cores in an SM
 - A total of 1024 floating point operations per clock
 - Multiplication and addition
 - 8× increase in throughput for deep learning applications per SM compared to Pascal GP100 using standard FP32 operations
 - Total 12× increase in throughput for the Volta V100 GPU compared to the Pascal P100 GPU

	Tesla V100	RTX 2080*
Double precision peak performance	7 TFLOPS	300 GFLOPS
Single precision peak performance	14 TFLOPS	10 TFLOPS
Half precision peak performance using tensor core	112 TFLOPS	80 TFLOPS



CUDA libraries for Tensor Cores

- cuBLAS and cuDNN
 - cuBLAS uses Tensor Cores to speed up GEMM computations
 - cuDNN uses Tensor Cores to speed up both CNNs and RNNs
- Intrinsic in CUDA programming model

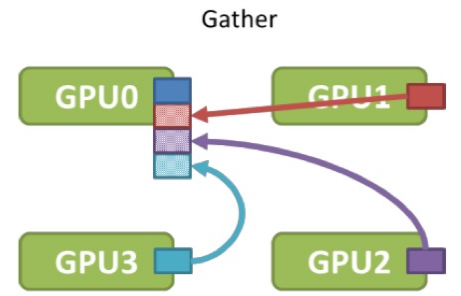
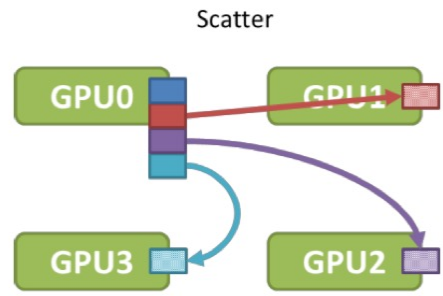
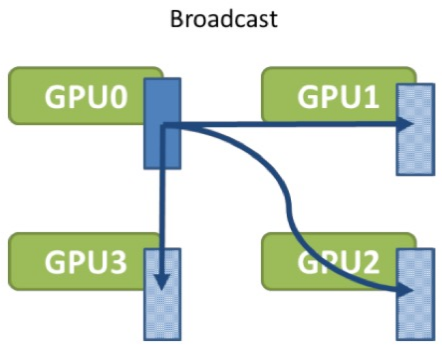


NCCL

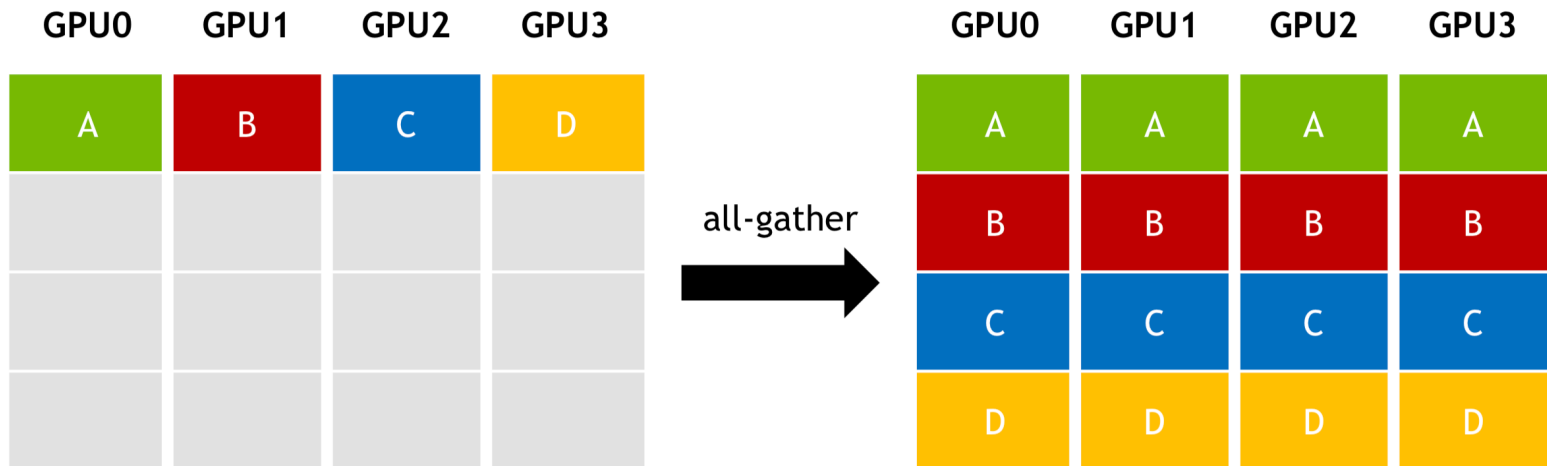
- NVIDIA Collective Communication Library
- Between GPUs in a node, or across nodes
- Similar to MPI collective communication API
 - Broadcast
 - Scatter
 - Gather
 - All-Gather
 - All-to-All
 - Reduce
 - All-Reduce



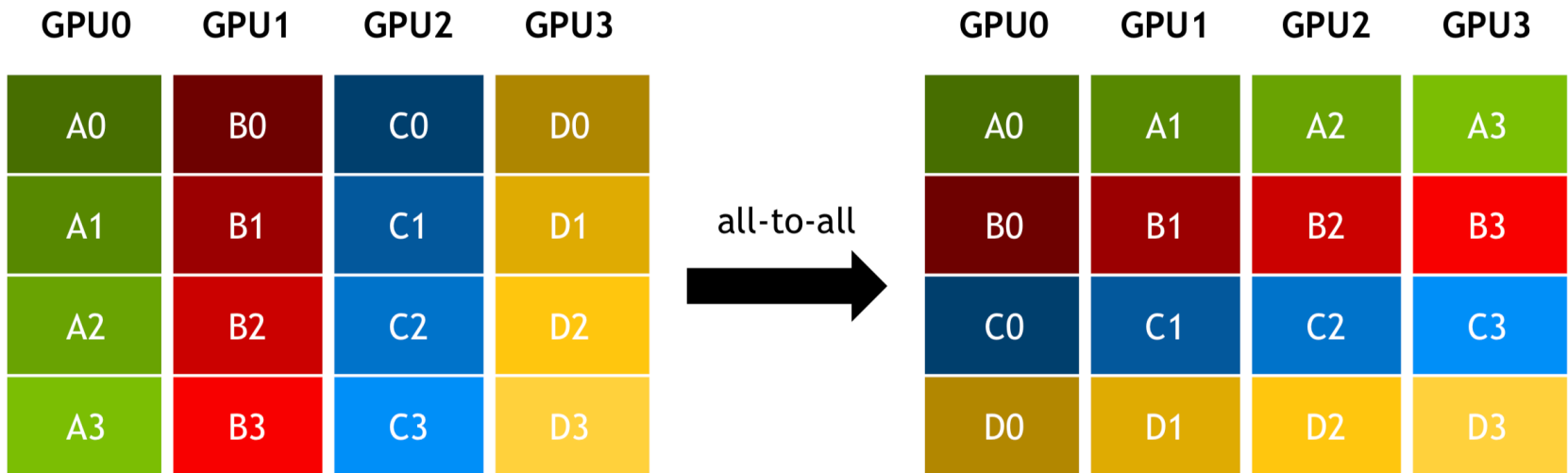
Broadcast, Scatter, and Gather



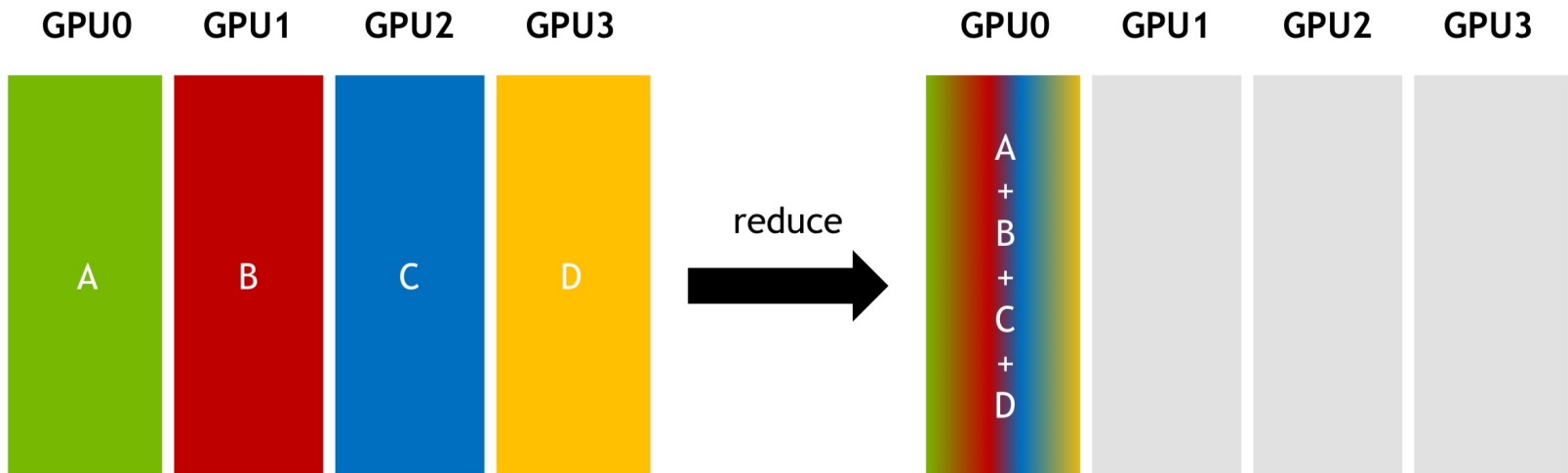
All-Gather



All-to-All



Reduce



All-Reduce

