Cloud Innovator
MEGAZONE

Megazone is delivering values through the business expertise and experiences built from the Platform Service, Digital Marketing and Digital Service areas

OpenACC
More Science, Less Programming

# Fundamentals of Parallel Computing & Programming with NVIDIA Architecture

Matilda Lab / Hybrid AI Center / Megazone Cloud Corp.
Sungho Kim, Ph.D.

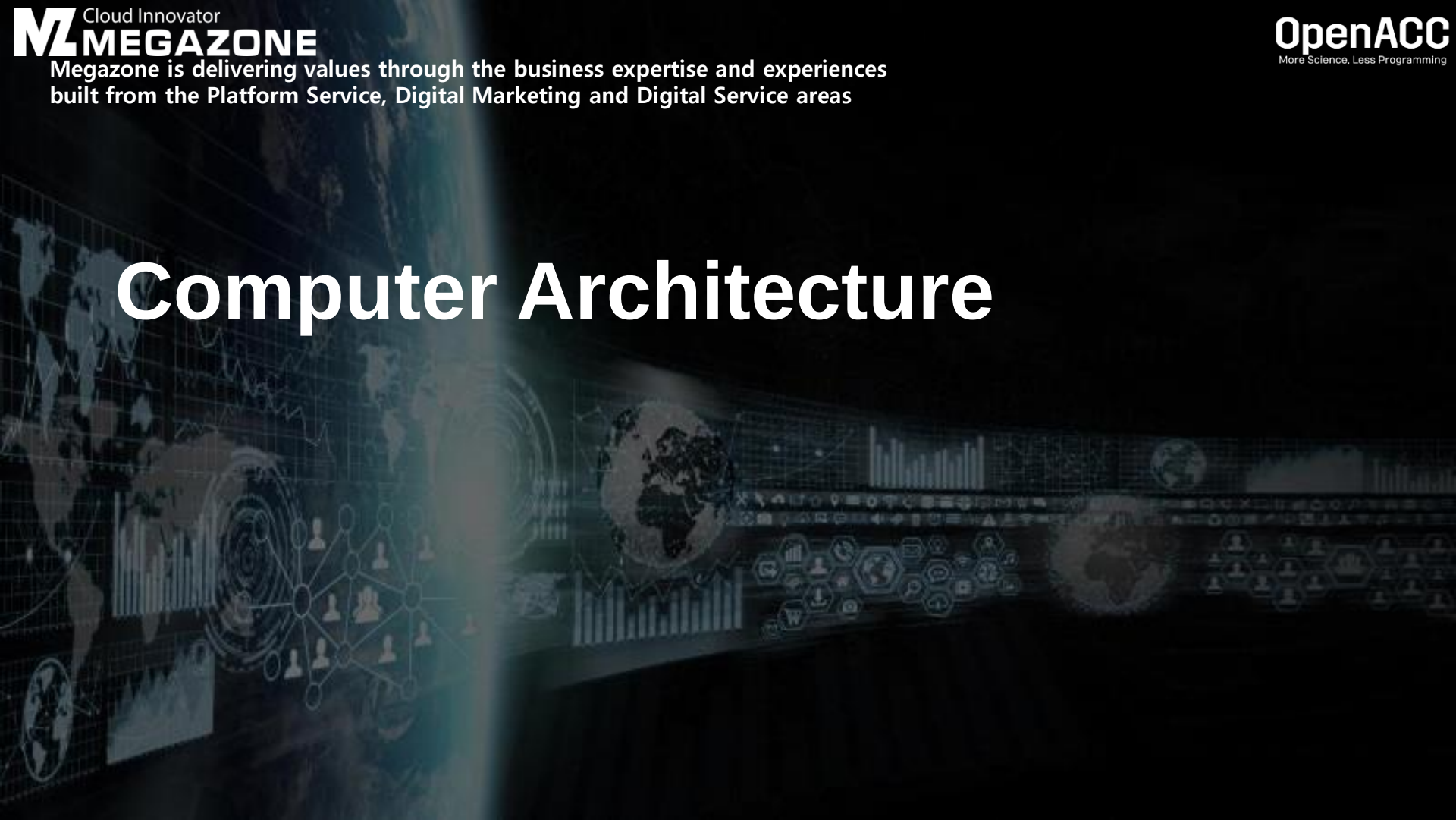
2023. 12.

# Contents

Cloud Innovator
MEGAZONE

Megazone is delivering values through the business expertise and experiences built from the Platform Service, Digital Marketing and Digital Service areas

OpenACC
More Science, Less Programming



# Computer Architecture

Cloud Innovator
MEGAZONE

Megazone is delivering values through the business expertise and experiences built from the Platform Service, Digital Marketing and Digital Service areas

OpenACC
More Science, Less Programming

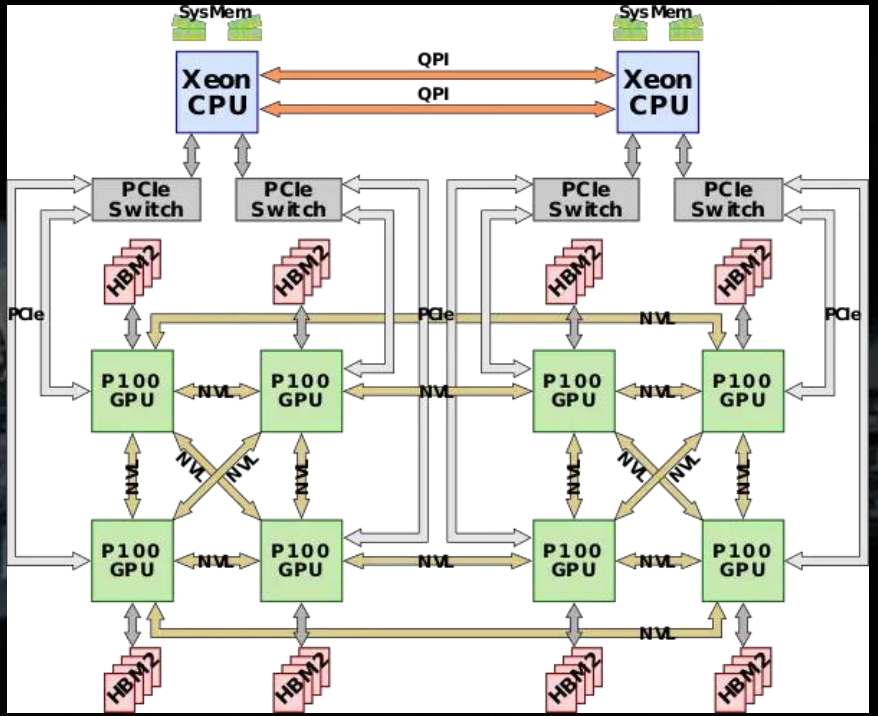
# NVIDIA HGX Server Architecture

Codename : Hopper Architecture

https://developer.nvidia.com/ko-kr/blog/nvidia-hopper-architecture-in-depth/

# HGX Server Overall Architecture

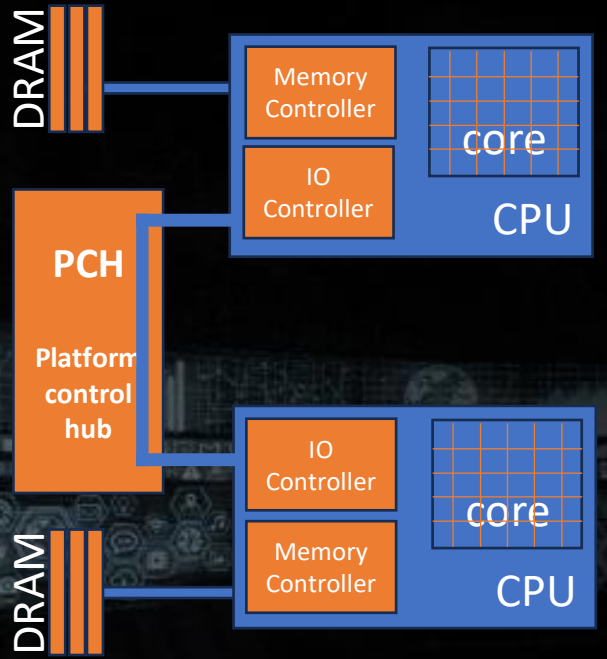
- Intel or ARM CPU
- DDR and HBM Memory
- PCIE switch
- Nvidia’s NVLink
- Infiniband
- NVM SSD



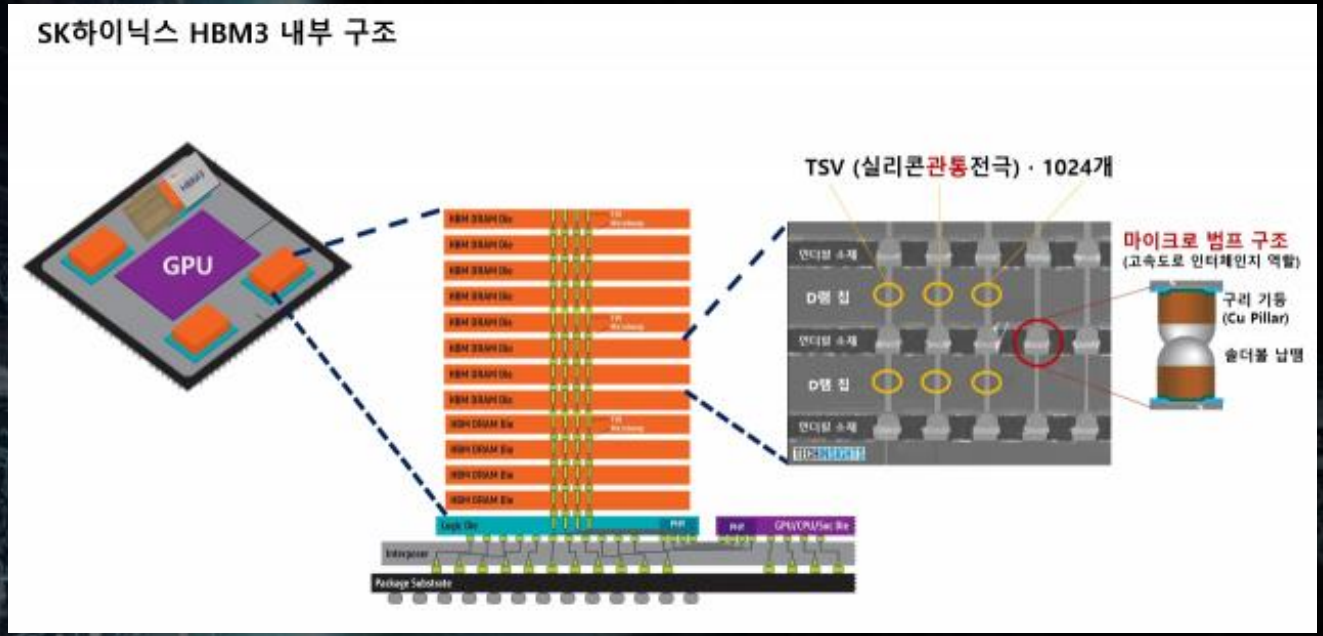
# CPU-Memory Connection

- DRAM-CPU-IO connection
  - DDR4→DDR5
  - DDR5→HBM
  - CPU-to-CPU : QPI/NUMA

- CPU-Memory Connection
  - DDR5 = 4000~8000MT/s x 64bit width / 8 bits/byte = 32GB/s ~ 64 GB/s
  - DDR4 = 1600~3200MT/s x 64bit width / 8 bits/byte = 12.8GB/s ~ 25.6GB/s



# HBM(High Bandwidth Memory) and H100 GPU core




- HBM3E=9.2 Gbps x 1024bit = 1.2TB/s
  - 24GB / 8 high-cube
  - 36GB / 12 high-cube

# PCIE Switch Chip and GPUs

- PCI Express Gen 5x16 lane interface, providing **128 GB/sec total bandwidth**
  - 64 GB/sec in each direction

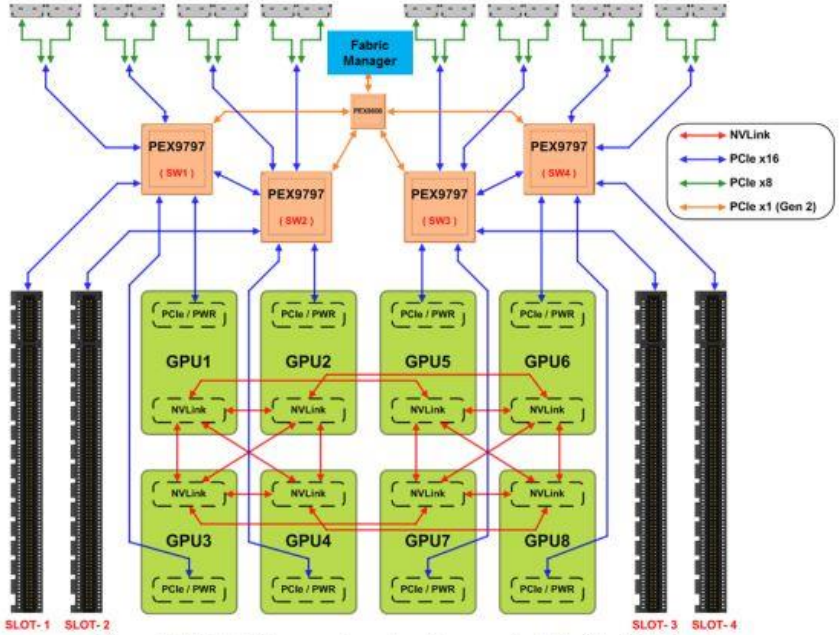
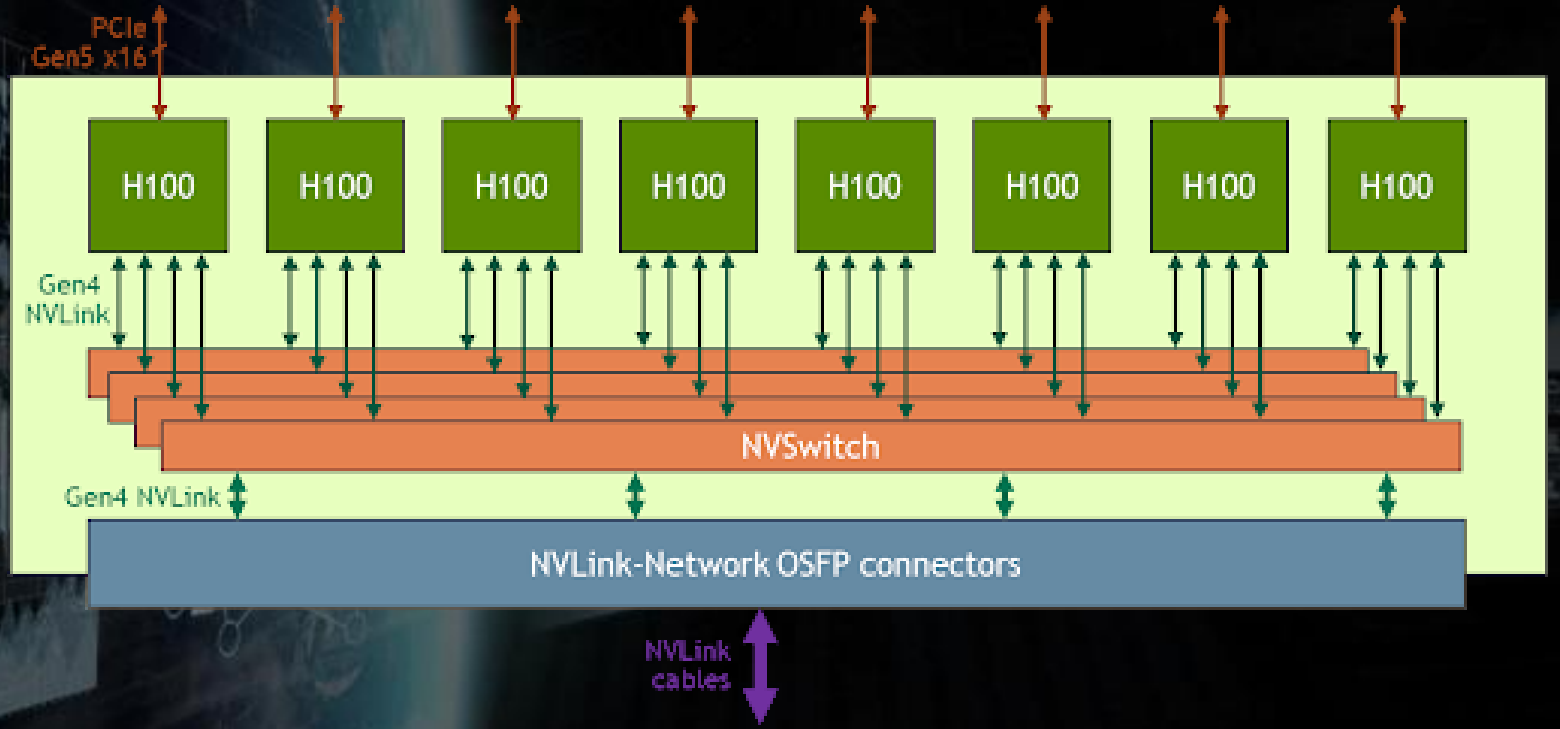



*Figure 5 NVLink in Hyper-cube and cable-connected, flexible PCIe Topology*

# GPU, NVLink, PCIE Connection

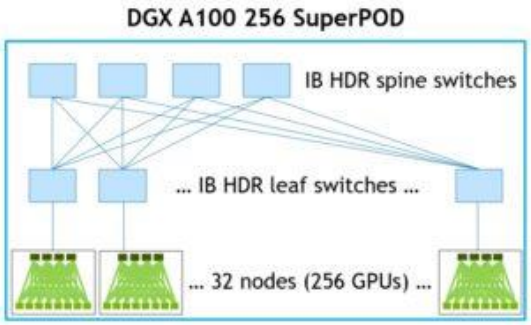


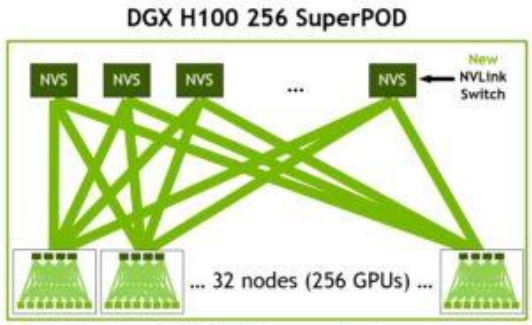
# 4th Gen NVLink and 3rd Gen NVSwitch

- New NVLink provides **7x the bandwidth of PCIe Gen 5**
  - Operating at **900 GB/sec total bandwidth** for multi-GPU I/O and shared memory accesses
    - A100 to provide 600 GB/sec total bandwidth.
- Each NVSwitch inside a node provides 64 ports
  - Total switch throughput increases to **13.6 Tbits/sec** (2nd Gen : 7.2 Tbits/sec)
- New NVSwitch provides **hardware acceleration of collective operations** with multicast and NVIDIA SHARP in-network reductions.
  - Accelerated collectives include **write broadcast (all_gather), reduce_scatter, and broadcast atomics**.
  - In-fabric multicast and reductions provide up to 2x throughput gain while using NCCLo n A100.
  - NVSwitch acceleration of collectives significantly reduces the load on SMs for collective communications.

# New NVLink Switch System

- NVLink Switch System supports up to 256 GPUs.
  - The connected nodes can deliver **57.6 TBs of all-to-all bandwidth** and can supply an incredible 1 exaFLOP of FP8 sparse AI compute.
  - The NVLink Network interconnect in **2:1 tapered fat tree topology** enables a staggering 9x increase in bisection bandwidth.




| | A100 SuperPod | | | H100 SuperPod | | | Speedup | |
|---|---|---|---|---|---|---|---|---|
| | Dense PFLOP/s | Bisection [GB/s] | Reduce [GB/s] | Dense PFLOP/s | Bisection [GB/s] | Reduce [GB/s] | Bisection | Reduce |
| 1 DGX / 8 GPUs | 2.5 | 2,400 | 150 | 16 | 3,600 | 450 | **1.5x** | **3x** |
| 32 DGXs / 256 GPUs | 80 | 6,400 | 100 | 512 | 57,600 | 450 | **9x** | **4.5x** |

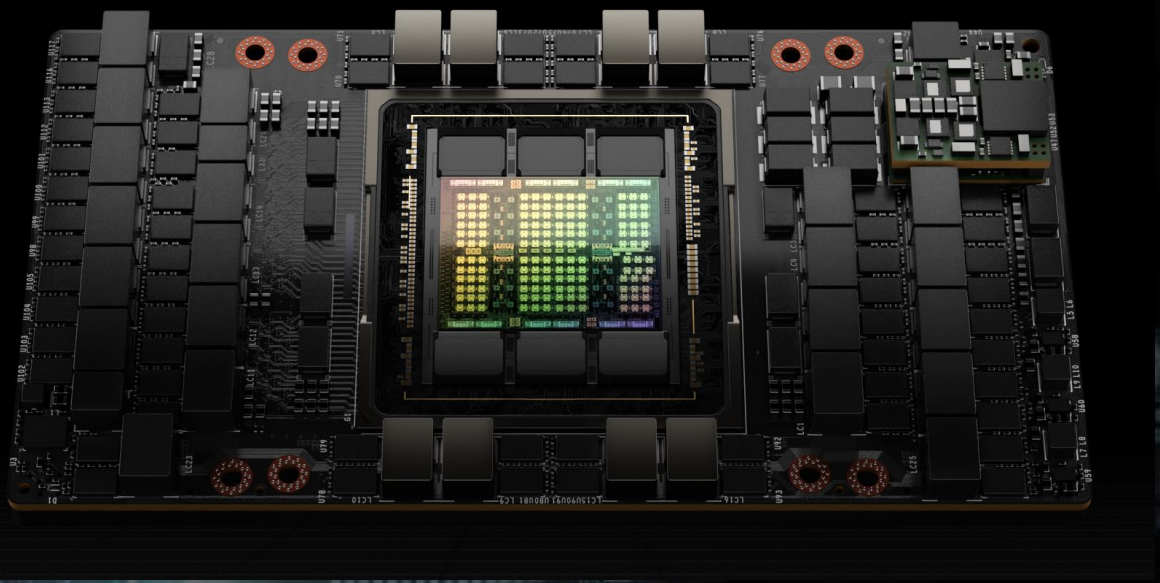
Cloud Innovator
MEGAZONE

Megazone is delivering values through the business expertise and experiences built from the Platform Service, Digital Marketing and Digital Service areas

OpenACC
More Science, Less Programming

# NVIDIA H100 Architecture

# NVIDIA H100 Tensor core GPU



- TSMC 4nm 제조공정
- AI, HPC, 데이터 분석을 위한 데이터 센터용 워크로드 실행
- 그래픽 처리 기능 없음
  - SXM5 및 PCIe H100 모두 단 2개의 TPC 만이 그래픽 기능 지원

# HOPPER H100 TENSOR CORE GPU

80B Transistors, TSMC 4N


2nd Gen Multi-Instance GPU
Confidential Computing
PCIe Gen5

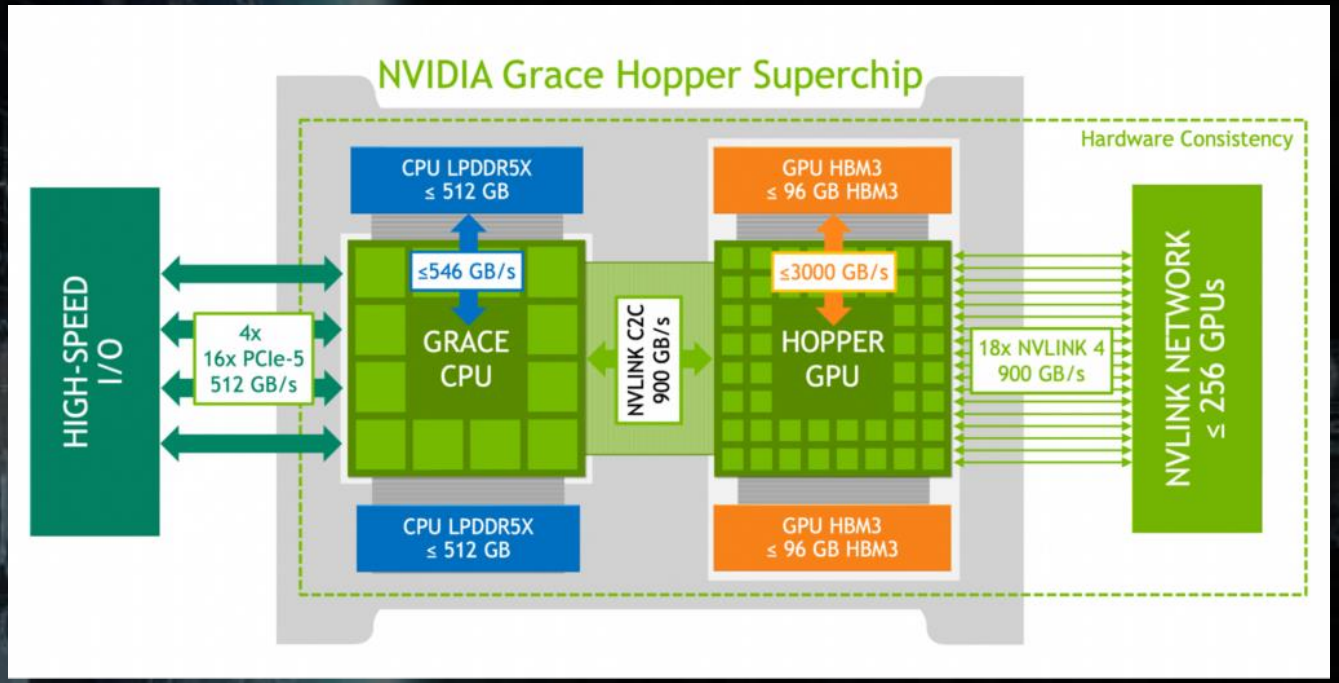
New Memory System
World's First HBM3 DRAM
Larger 50MB L2

132 SMs
2x Performance per Clock
4th Gen Tensor Core
Thread Block Clusters

4th Gen NVLink
900 GB/s total bandwidth
New SHARP support
NVLink Network

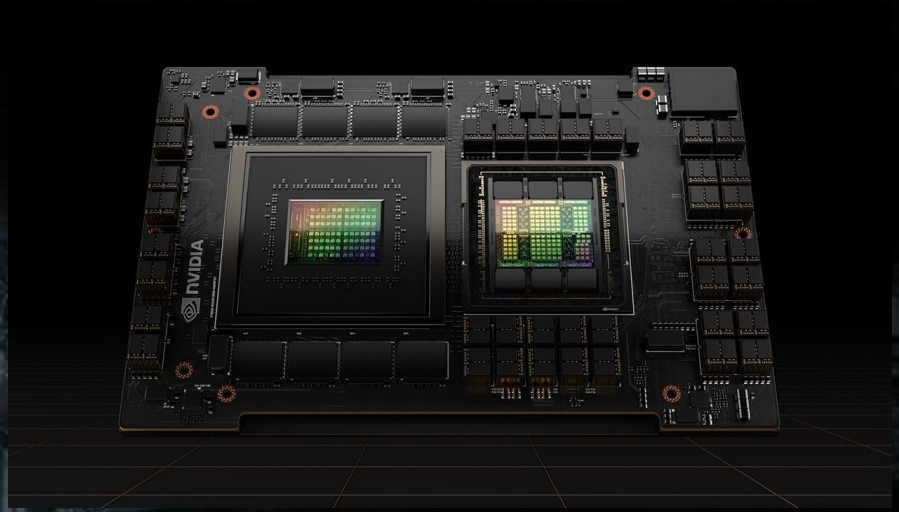
NVIDIA

# Grace Hopper Superchip with ARM CPU



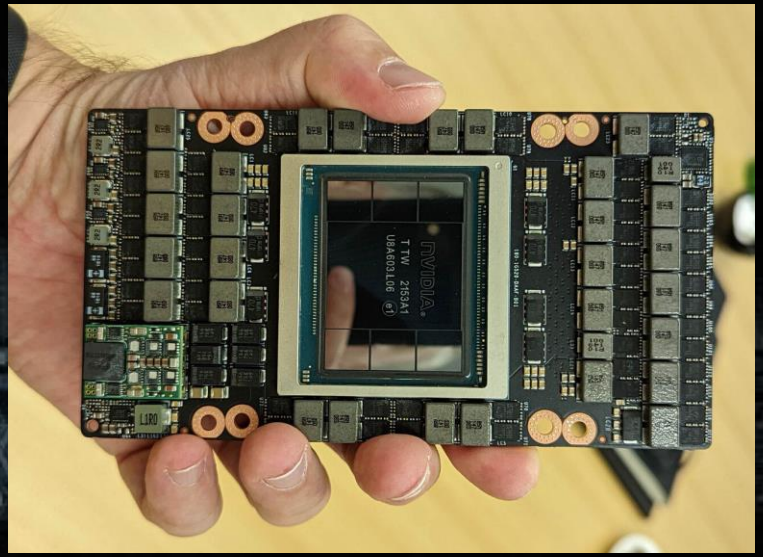
# NVIDIA GH100 Spec

- GH100(Grace Hopper)
  - 8개의 GPC
  - 72개의 TPC(GPC당 TPC 9개)
  - TPC당 2개의 SM = 144개
  - GPU당 144개의 SM
  - SM당 128개의 FP32 CUDA 코어
    - 전체 GPU당 18432개의 FP32 CUDA 코어
  - SM당 4개의 4세대 Tensor 코어
    - 전체 GPU당 576개
  - HBM3 또는 HBM2e 스택 6개
    - 512비트 메모리 컨트롤러 12개
  - 60MB L2 캐시
  - 4세대 NVLink 및 PCIe Gen 5



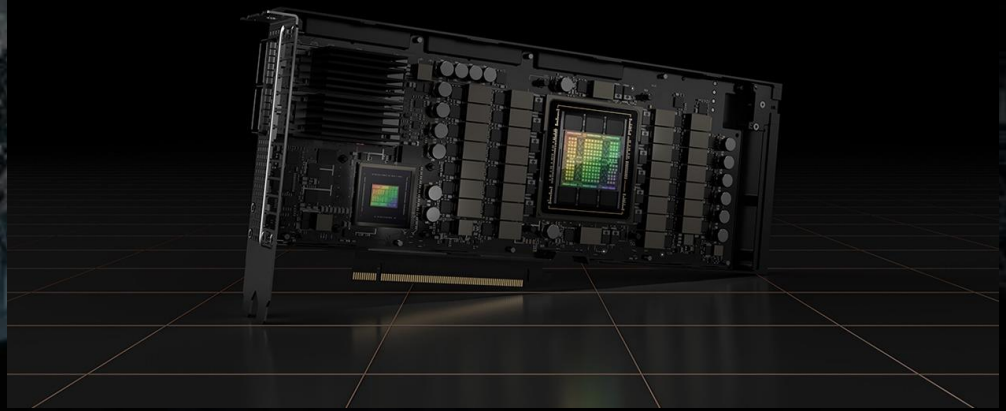
# NVIDIA H100 GPU SXM5 Formfactor Spec

- SXM5 Board NVIDIA H100 GPU
  - 8개의 GPC
  - 66개의 TPC
  - TPC당 2개의 SM = 132개
  - GPU당 132개의 SM
  - SM당 128개의 FP32 CUDA 코어
    - 전체 GPU당 16896개의 FP32 CUDA 코어
  - SM당 4개의 4세대 Tensor 코어
    - GPU당 528개
  - 80GB HBM3, HBM2e 스택 5개
    - 512비트 메모리 컨트롤러 10개
  - 50MB L2 캐시
  - 4세대 NVLink 및 PCIe Gen 5

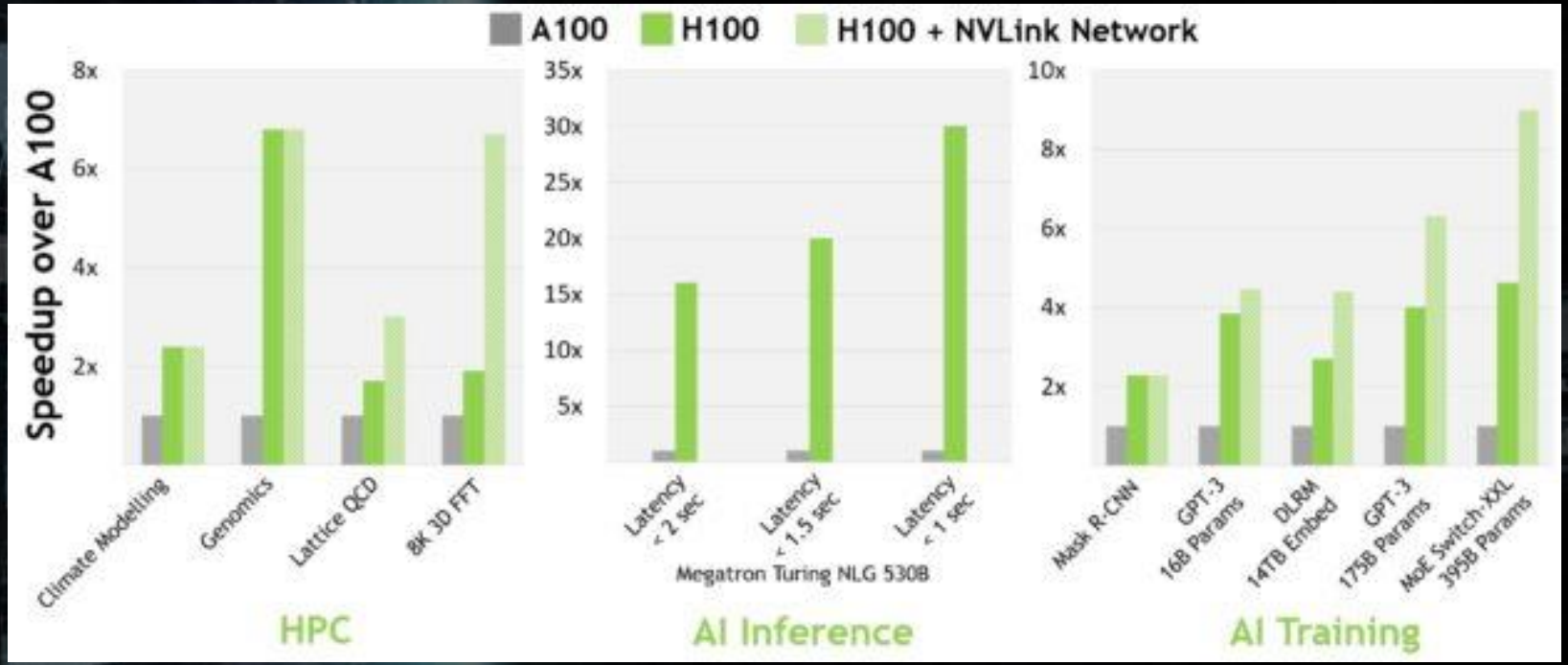


# NVIDIA H100 GPU PCIe Gen 5 Spec

- PCIe Gen5 Board NVIDIA H100 GPU
  - 7개 또는 8개의 GPC,
  - 57개의 TPC
  - TPC당 2개의 SM = 114개
  - GPU당 114개의 SM
  - SM당 128개의 FP32 CUDA 코어
    - 전체 GPU당 14592개의 FP32 CUDA 코어
  - SM당 4개의 4세대 Tensor 코어
    - GPU당 456개
  - 80GB HBM2e, HBM2e 스택 5개
    - 512비트 메모리 컨트롤러 10개
  - 50MB L2 캐시
  - 4세대 NVLink 및 PCIe Gen 5



# Performance Comparison H100 with A100



# NVIDIA H100 TC 성능 사양

| | NVIDIA H100 SXM5[1] | NVIDIA H100 PCIe[1] |
|---|---|---|
| Peak FP64[1] | 30TFLOPS | 24TFLOPS |
| Peak FP64 Tensor core[1] | 60TFLOPS | 48TFLOPS |
| Peak FP32[1] | 60TFLOPS | 48TFLOPS |
| Peak FP16[1] | 120TFLOPS | 96TFLOPS |
| Peak BF16[1] | 120TFLOPS | 96TFLOPS |
| Peak TF32 Tensor core[1] | 500TFLOPS \| 1000TFLOPS[2] | 400TFLOPS \| 800TFLOPS[2] |
| Peak FP16 Tensor core[1] | 1000TFLOPS \| 2000TFLOPS[2] | 800TFLOPS \| 1600TFLOPS[2] |
| Peak BF16 Tensor core[1] | 1000TFLOPS \| 2000TFLOPS[2] | 800TFLOPS \| 1600TFLOPS[2] |
| Peak FP8 Tensor core[1] | 2000TFLOPS \| 4000TFLOPS[2] | 1600TFLOPS \| 3200TFLOPS[2] |
| Peak INT8 Tensor core[1] | 2000TOPS \| 4000TOPS[2] | 1600TOPS \| 3200TOPS[2] |

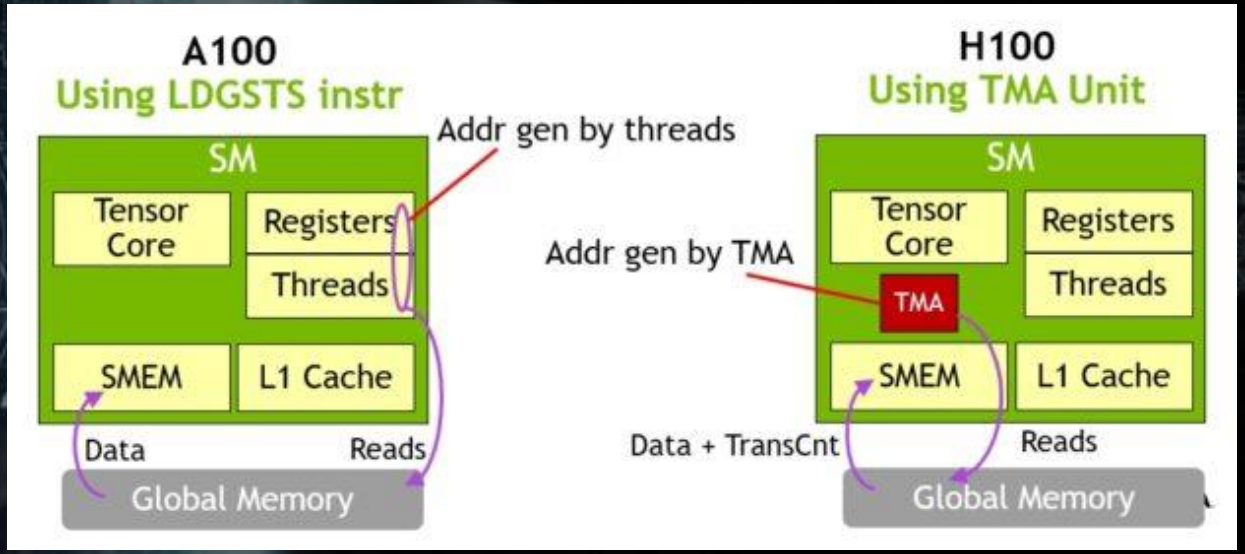
1. H100의 예비 성능 추정치
2. Sparsity 기능을 사용한 실질적인 TFLOPS 및 TOPS

# H100 SM Architecture

- 4th Gen Tensor Core
  - SM당 속도 향상, 늘어난 SM 수, H100의 더 높은 클럭, A100에 비해 칩 간 속도가 최대 6배
  - 2x Faster FP32 & FP64 FMA per SM, 4x Faster FP8
  - 2x Performance with Sparsity in Deep Learning
  - 7x Faster than A100 with New DPX Instruction Set
- 3x Faster than A100 on IEEE64 and FP32 with 2x clock speed per SM and increased SM and faster clock
- 1.33x larger than A100 on 256KB L1 Data cache / Shared Memory
- New Tensor Memory Accelerator(TMA)
  - Fully asynchronous data movement between global and shared memory
  - TMA는 Asynchronous Copy between Thread Block Cluster
  - Asynchronous Transaction Barrier for Atomic Data movement and Synchronization
- New Thread Block Cluster
  - Turn locality into Efficiency
- Direct Communication for load/store between SM with Distributed Shared memory

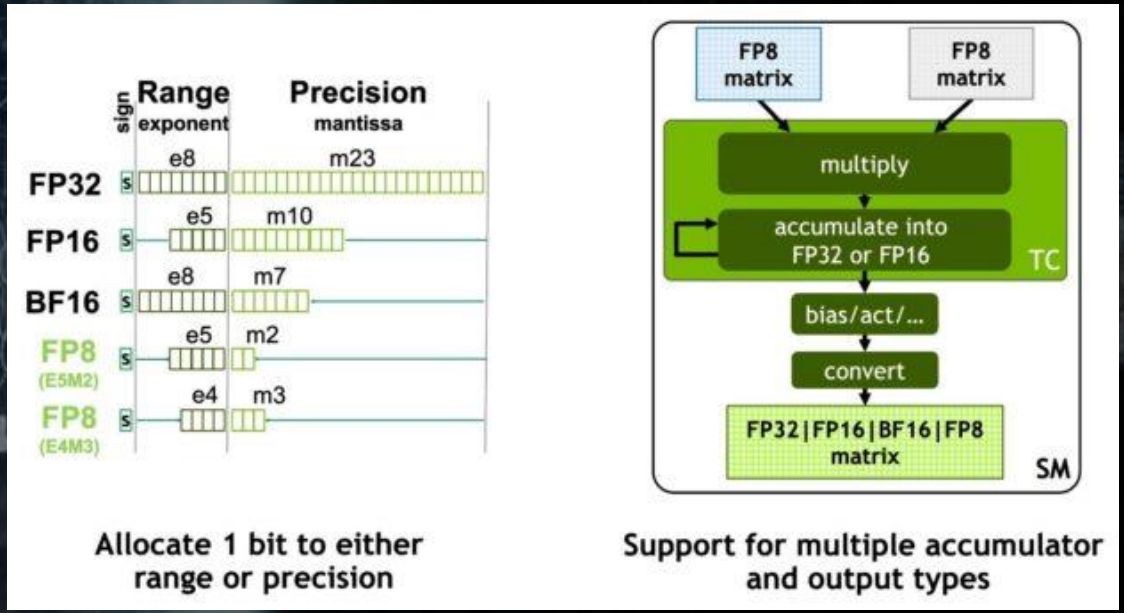
# New Tensor Memory Accelerator(TMA)

- Fully asynchronous data movement between global and shared memory
  - TMA: Asynchronous Copy between Thread Block Cluster
  - Asynchronous Transaction Barrier for Atomic Data movement and Synchronization



# NVIDIA Hopper FP8 Data Format

- New FP8 Data format
  - E4M3 : Sign 1bit, Exponent 4bit, Mantissa 3bit
  - E5M2 : Sign 1bit, Exponent 5bit, Mantissa 2bit

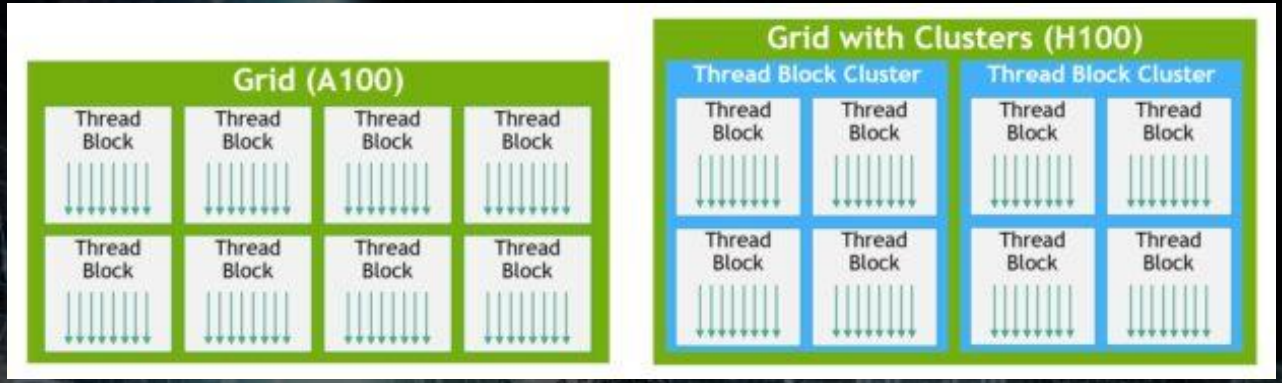


# H100 Performance


| TFLOPS 단위 측정 | A100 | A100 Sparsity | H100 SXM5 | H100 SXM5 Sparsity | H100 SXM5 속도 향상 (A100 대비) |
|---|---|---|---|---|---|
| FP8 TC | | | 2000 | 4000 | 6.4배 |
| FP16 | 78 | | 120 | | 1.5배 |
| FP16 TC | 312 | 624 | 1000 | 2000 | 3.2배 |
| BF16 TC | 312 | 624 | 1000 | 2000 | 3.2배 |
| FP32 | 19.5 | | 60 | | 3.1배 |
| TF32 TC | 156 | 312 | 500 | 1000 | 3.2배 |
| FP64 | 9.7 | | 30 | | 3.1배 |
| FP64 TC | 19.5 | | 60 | | 3.1배 |
| INT8 TC | 624TOPS | 1248TOPS | 2000 | 4000 | 3.2배 |

- A100의 SM 108개보다 22% 증가한 132개의 SM
- 새로운 4세대 Tensor 코어 덕분에 2배 더 빠른 각각의 H100 SM 속도
- 각 Tensor 코어 내에서 새로운 FP8 형식 및 관련 트랜스포머 엔진으로 2배 개선
- 또한 H100의 클럭 주파수 증가로 성능이 약 1.3배 향

# Thread Block Cluster

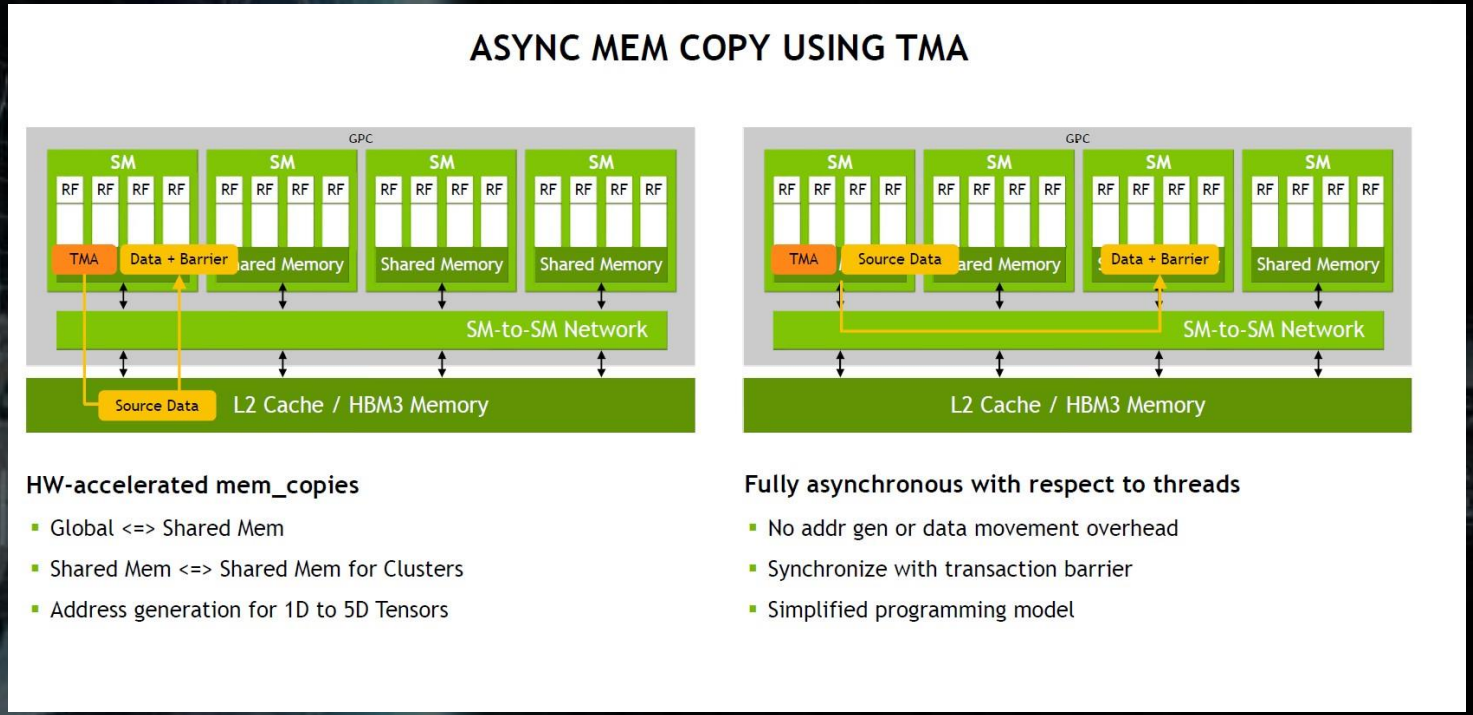




- GPC는 항상 물리적으로 근접하게 함께 있는 하드웨어 계층 구조의 SM
- GPC의 SM과 SM 간의 전용 네트워크는 클러스터의 스레드 사이에서 빠르게 데이터를 공유

# Asynchronous execution

to enable more overlap of data movement, computation, and synchronization.




**HW-accelerated mem_copies**

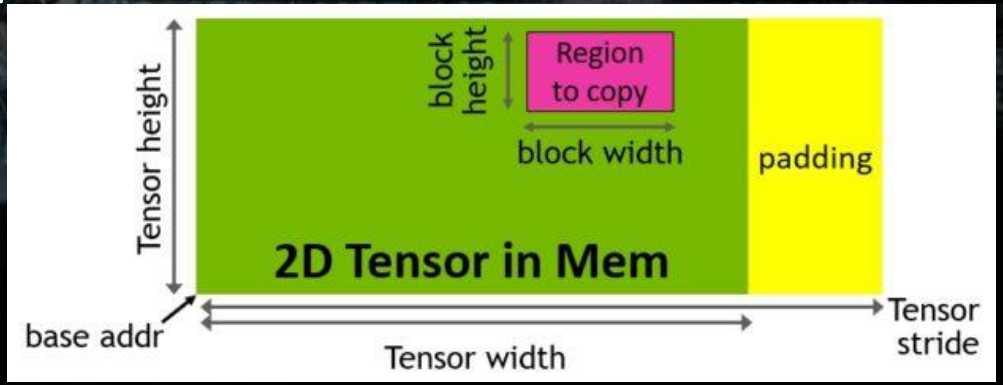
- Global <=> Shared Mem
- Shared Mem <=> Shared Mem for Clusters
- Address generation for 1D to 5D Tensors

**Fully asynchronous with respect to threads**

- No addr gen or data movement overhead
- Synchronize with transaction barrier
- Simplified programming model

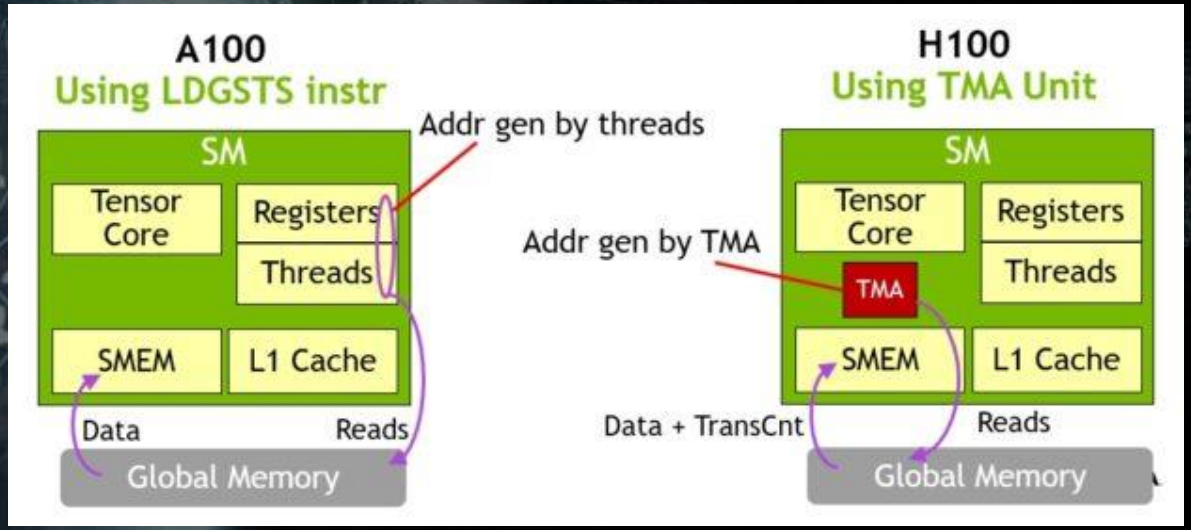
# TMA(*Tensor Memory Accelerator*)

- Data fetch efficiency is improved  with TMA
  - can transfer large blocks of data and multidimensional tensors from global memory to shared memory and back again.
- A single thread in a warp is elected to issue an asynchronous TMA operation (cuda::memcpy_async) to copy a tensor.
- As a result, multiple threads can wait on a cuda::barrier for completion of the data transfer.



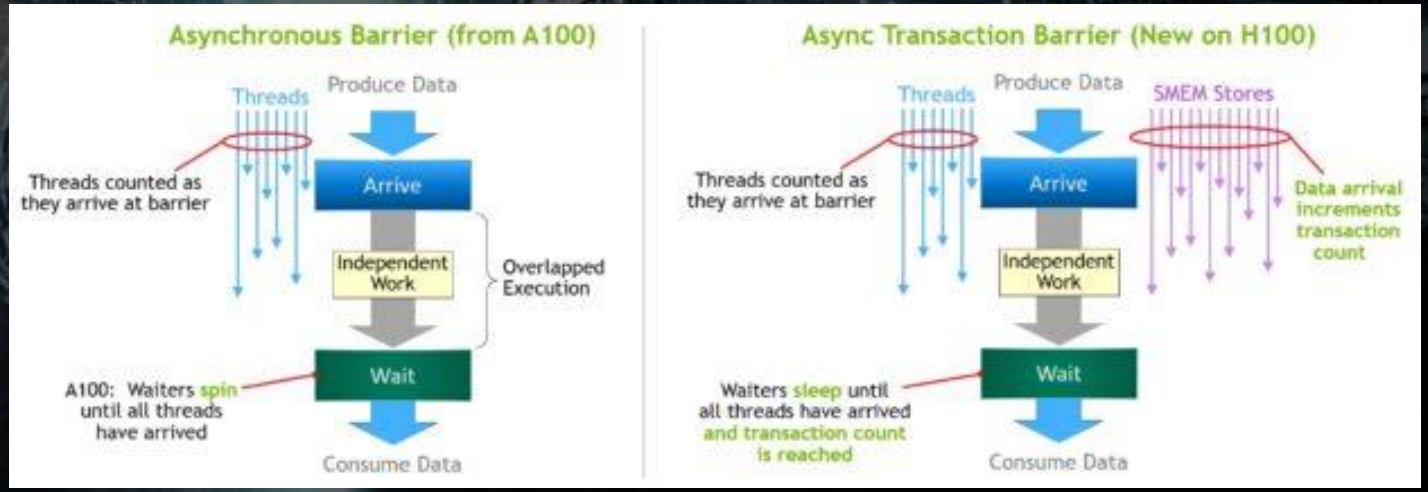
# New Tensor Memory Accelerator(TMA)

- Fully asynchronous data movement between global and shared memory
  - TMA: Asynchronous Copy between Thread Block Cluster
  - Asynchronous Transaction Barrier for Atomic Data movement and Synchronization
  - *LoadGlobalStoreShared* Instruction set



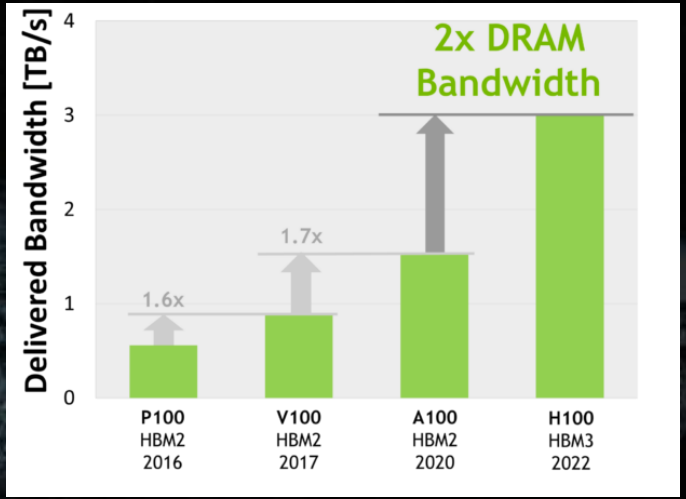
# Asynchronous transaction barrier

- First, threads signal ***Arrive*** when they are done producing their portion of the shared data. This ***Arrive*** is non-blocking so that the threads are free to execute other independent work.
- Eventually, the threads need the data produced by all the other threads. At this point, they do a ***Wait***, which blocks them until every thread has signaled ***Arrive***.
- New for NVIDIA Hopper is the ability for *waiting* threads to sleep until all other threads arrive.

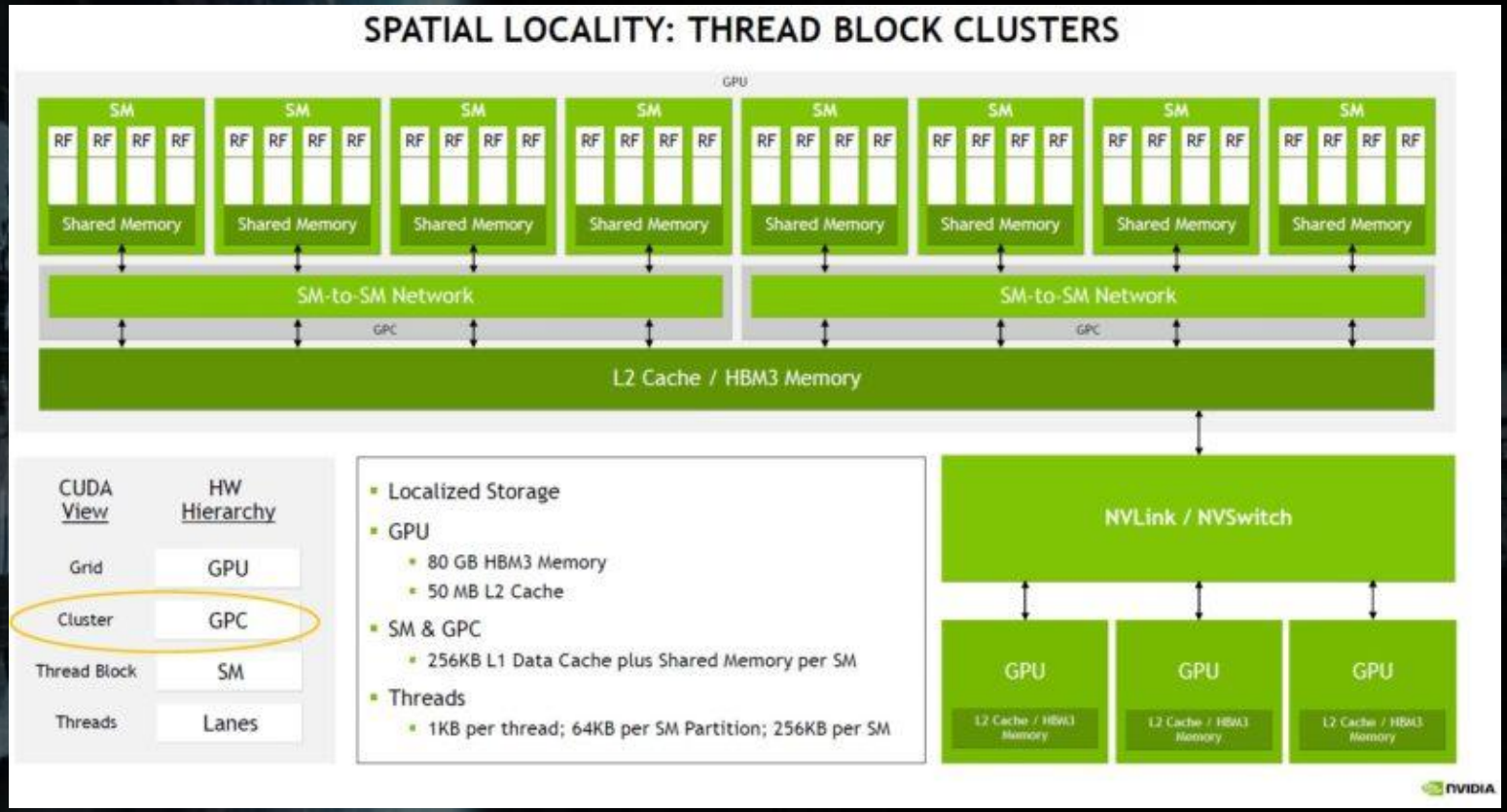


# H100 HBM3 and HBM2e DRAM

- The H100 SXM5 GPU raises the bar considerably by supporting
  - 80 GB (five stacks) of fast HBM3 memory,
  - delivering over 3 TB/sec of memory bandwidth,
  - effectively a 2x increase over the memory bandwidth of A100 that was launched just two years ago.
- The PCIe H100 provides 80 GB of fast HBM2e with over 2 TB/sec of memory bandwidth.



# Spatial Locality : Thread Block Clusters



# Spec Summary

| GPU 기능 | NVIDIA A100 | NVIDIA H100 SXM5[1] | NVIDIA H100 PCIe[1] |
|---|---|---|---|
| GPU 아키텍처 | NVIDIA Ampere | NVIDIA Hopper | NVIDIA Hopper |
| GPU 보드 폼 팩터 | SXM4 | SXM5 | PCIe Gen 5 |
| SM | 108 | 132 | 114 |
| TPC | 54 | 66 | 57 |
| SM당 FP32 코어 수 | 64 | 128 | 128 |
| GPU당 FP32 코어 수 | 6912 | 16896 | 14592 |
| SM당 FP64 코어 수(Tensor 제외) | 32 | 64 | 64 |
| CPU당 FP64 코어 수(Tensor 제외) | 3456 | 8448 | 7296 |
| SM당 INT32 코어 수 | 64 | 64 | 64 |
| GPU당 INT32 코어 수 | 6912 | 8448 | 7296 |
| SM당 Tensor 코어 수 | 4 | 4 | 4 |
| GPU당 Tensor 코어 수 | 432 | 528 | 456 |
| GPU 부스트 클럭<br>(H100의 경우 최종 확정되지 않음)[3] | 1410MHz | 최종 확정되지 않음 | 최종 확정되지 않음 |
| TDP[1] | 400W | 700W | 350W |
| 트랜지스터 | 542억 개 | 800억 개 | 800억 개 |
| GPU 다이 크기 | 826mm2 | 814mm2 | 814mm2 |
| TSMC 제조 프로세스 | 7nm N7 | NVIDIA에 맞춤화된 4N | NVIDIA에 맞춤화된 4N |

# Spec Summary

| GPU 기능 | NVIDIA A100 | NVIDIA H100 SXM5[1] | NVIDIA H100 PCIe[1] |
|---|---|---|---|
| 최대 FP8 Tensor TFLOPS(FP16 누산 포함)[1] | 해당 없음 | 2000/4000[2] | 1600/3200[2] |
| 최대 FP8 Tensor TFLOPS(FP32 누산 포함)[1] | 해당 없음 | 2000/4000[2] | 1600/3200[2] |
| 최대 FP16 Tensor TFLOPS(FP16 누산 포함)[1] | 312/624[2] | 1000/2000[2] | 800/1600[2] |
| 최대 FP16 Tensor TFLOPS(FP32 누산 포함)[1] | 312/624[2] | 1000/2000[2] | 800/1600[2] |
| 최대 BF16 Tensor TFLOPS(FP32 누산 포함)[1] | 312/624[2] | 1000/2000[2] | 800/1600[2] |
| 최대 TF32 Tensor TFLOPS[1] | 156/312[2] | 500/1000[2] | 400/800[2] |
| 최대 FP64 Tensor TFLOPS[1] | 19.5 | 60 | 48 |
| 최대 INT8 Tensor TOPS[1] | 624/1248[2] | 2000/4000[2] | 1600/3200[2] |
| 최대 FP16 TFLOPS(Tensor 외)[1] | 78 | 120 | 96 |
| 최대 BP16 TFLOPS(Tensor 외)[1] | 39 | 120 | 96 |
| 최대 FP32 TFLOPS(Tensor 외)[1] | 19.5 | 60 | 48 |
| 최대 FP64 TFLOPS(Tensor 외)[1] | 9.7 | 30 | 24 |
| 최대 INT32 TOPS[1] | 19.5 | 30 | 24 |

# Spec Summary

| GPU 기능 | NVIDIA A100 | NVIDIA H100 SXM5[1] | NVIDIA H100 PCIe[1] |
|---|---|---|---|
| 텍스처 유닛 | 432 | 528 | 456 |
| 메모리 인터페이스 | 5120비트 HBM2 | 5120비트 HBM3 | 5120비트 HBM2e |
| 메모리 크기 | 40GB | 80GB | 80GB |
| 메모리 데이터 속도 | 1215MHz DDR | 최종 확정되지 않음 | 최종 확정되지 않음 |
| 메모리 대역폭[1] | 1555GB/sec | 3000GB/sec | 2000GB/sec |
| L2 캐시 크기 | 40MB | 50MB | 50MB |
| SM당 공유 메모리 크기 | 최대 164KB 구성 가능 | 최대 228KB 구성 가능 | 최대 228KB 구성 가능 |
| SM당 레지스터 파일 크기 | 256KB | 256KB | 256KB |
| GPU당 레지스터 파일 크기 | 27648KB | 33792KB | 29184KB |

# Compute Capability

| Data Center GPU | NVIDIA V100 | NVIDIA A100 | NVIDIA H100 |
|---|---|---|---|
| GPU architecture | NVIDIA Volta | NVIDIA Ampere | NVIDIA Hopper |
| Compute capability | 7.0 | 8.0 | 9.0 |
| Threads / warp | 32 | 32 | 32 |
| Max warps / SM | 64 | 64 | 64 |
| Max threads / SM | 2048 | 2048 | 2048 |
| Max thread blocks (CTAs) / SM | 32 | 32 | 32 |
| Max thread blocks / thread block clusters | N/A | N/A | 16 |
| Max 32-bit registers / SM | 65536 | 65536 | 65536 |
| Max registers / thread block (CTA) | 65536 | 65536 | 65536 |
| Max registers / thread | 255 | 255 | 255 |
| Max thread block size (# of threads) | 1024 | 1024 | 1024 |
| FP32 cores / SM | 64 | 64 | 128 |
| Ratio of SM registers to FP32 cores | 1024 | 1024 | 512 |
| Shared memory size / SM | Configurable up to 96 KB | Configurable up to 164 KB | Configurable up to 228KB |

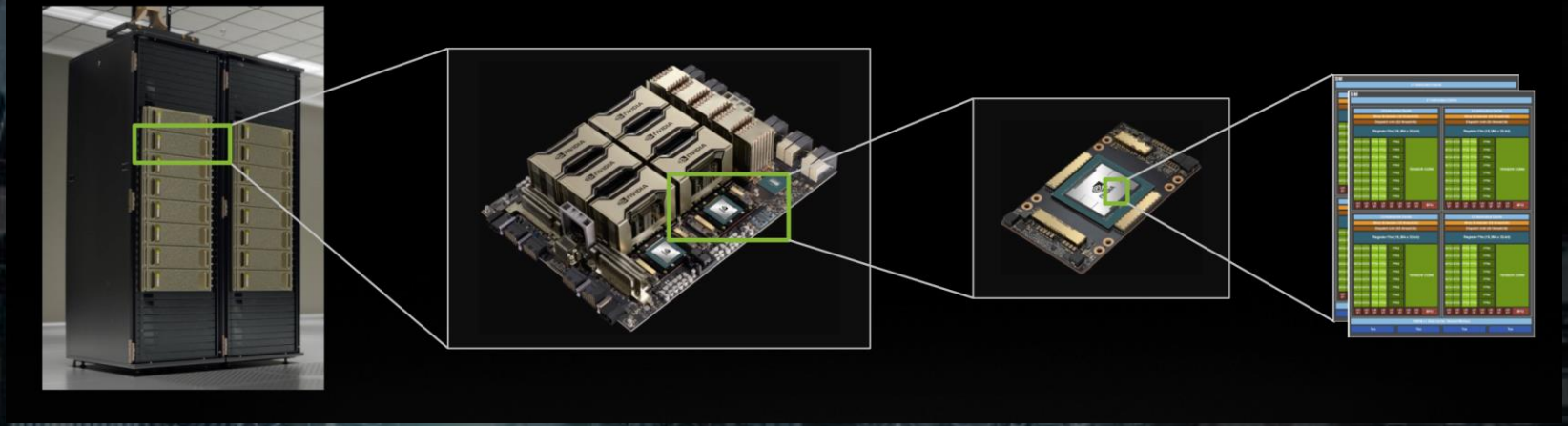

Pascal Streaming Multiprocessor (GP104)
Instruction Cache
SM Processing Block (Partition)
Instruction Buffer
Warp Scheduler
2x Dispatch Unit
Register File
FP32 / INT32
8x LD/ST
SFU
Textrure/L1 Cache
4x Texture Units
Texture/L1 Cache
4x Texture Units
Shared Memory
Turing Streaming Multiprocessor (TU104)
SM Processing Block (Partition)
L0 Instruction Cache
Warp Scheduler
Dispatch Unit
Register File
INT32
FP32
2x Tensor Cores 2nd Gen
4x LD/ST
SFU
L1 Data Cache/Shared Memory
4x Texture Units
RT Core 1st Generation
2x FP64
Ampere Streaming Multiprocessor (GA104)
FP32 / INT32
FP32
Tensor Core 3rd Gen
RT Core 2nd Generation
SM
L1 Instruction Cache
L0 Instruction Cache
Warp Scheduler (32 thread/clk)
Dispatch Unit (32 thread/clk)
Register File (16,384 x 32-bit)
INT32
FP32
FP64
LD/ST
SFU
TENSOR CORE 4th GENERATION
Tensor Memory Accelerator
256 KB L1 Data Cache / Shared Memory
Tex

# NVIDIA HPC SDK

Installation and Basic Setting

https://developer.nvidia.com/hpc-sdk/

https://docs.nvidia.com/hpc-sdk/index.html



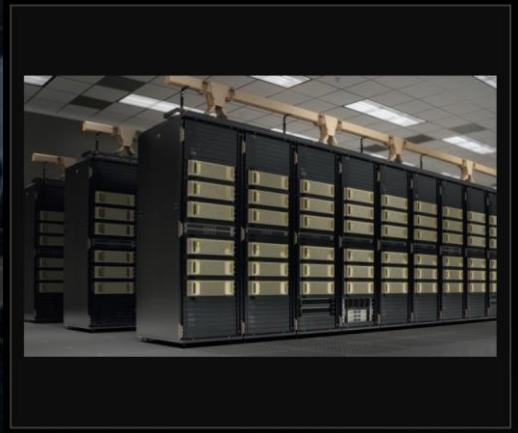
Multi-System Rack
Unlimited Scale

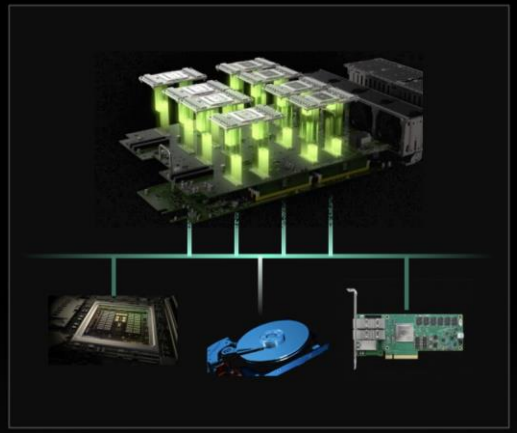
Multi-GPU System
8 GPUs

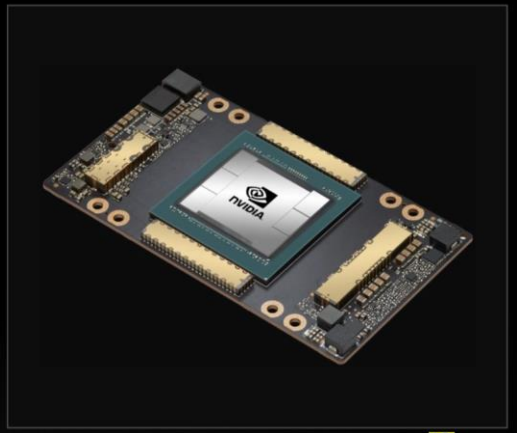
Multi-SM System
144 SMs

Multi-Core SM
2048 Threads

# CUDA Platform: Targets Each level of the Hierarchy







## System Scope

Fabric Management
Data Center Operations
Deployment
Monitoring
Compatibility
Security

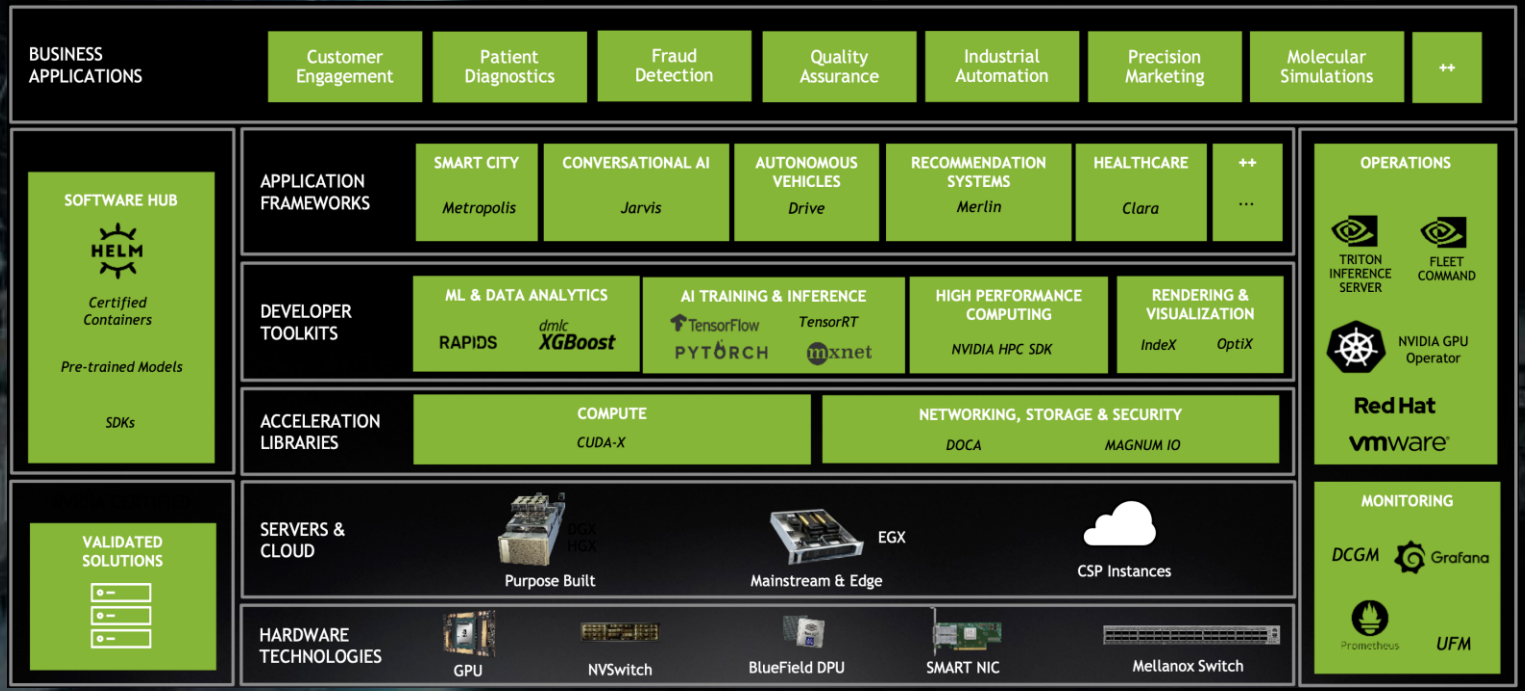
## Multi-GPU System

GPU-FIRECT
NVLINK
LIBRARIES
UNIFIED MEMORY
ARM
MIG

## Multi-SM System

CUDA C++
OPENACC
STANDARD LANGUAGES
SYNCRONIZATION
PRECISION

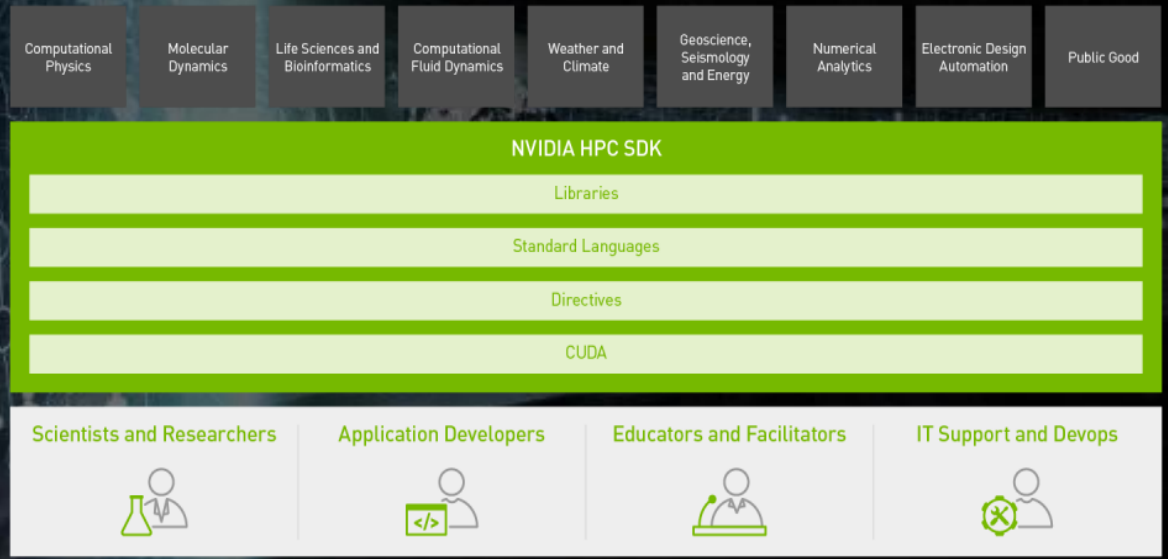
# Accelerated Platform



# NVIDIA HPC SDK

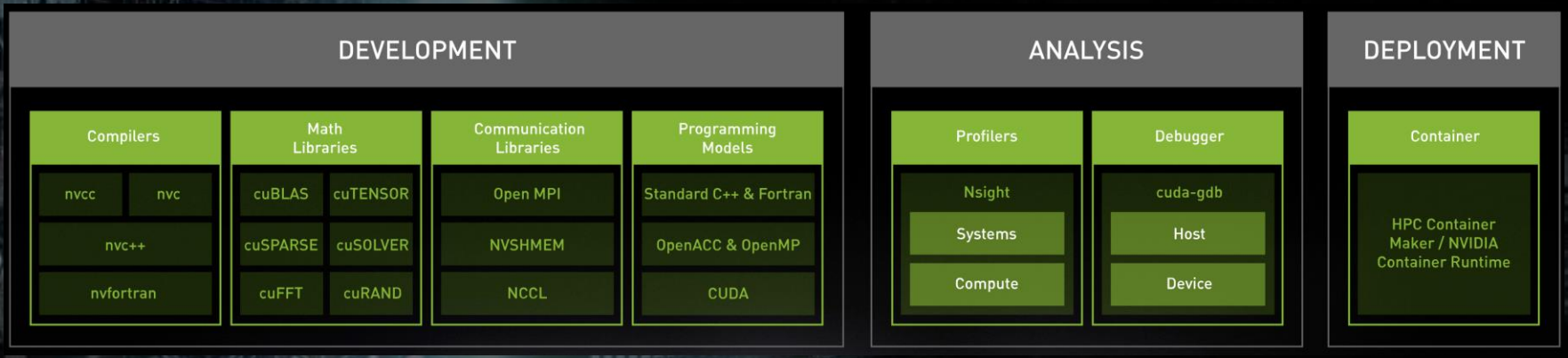
## A Comprehensive Suite of Fortran, C, and C++ Development Tools and Libraries

- The NVIDIA HPC SDK is a comprehensive toolbox for GPU accelerating HPC modeling and simulation applications.
- It includes the C, C++, and Fortran compilers, libraries, and analysis tools necessary for developing HPC applications on the NVIDIA platform.



# NVIDIA HPC SDK

- Download at developer.nvidia.com/hpc-sdk



# NVIDIA HPC SDK

- The NVIDIA HPC SDK is a comprehensive suite of compilers, libraries and tools essential to maximizing developer productivity and the performance and portability of HPC applications.
  - The NVIDIA HPC SDK C, C++, and Fortran compilers support GPU acceleration of HPC modeling and simulation applications with standard C++ and Fortran, OpenACC directives, and CUDA.
  - GPU-accelerated math libraries maximize performance on common HPC algorithms, and optimized communications libraries enable standards-based multi-GPU and scalable systems programming.
  - Performance profiling and debugging tools simplify porting and optimization of HPC applications, and containerization tools enable easy deployment on-premises or in the cloud.
- From computational science to AI, GPU-accelerated applications are delivering groundbreaking scientific discoveries. And popular languages like C, C++, Fortran, and Python are being used to develop, optimize, and deploy these applications.

# Lab Exercise

- nvaccelinfo
- nvidia-smi
- Visit  developer.nvidia.com/gpu-accelerated-libraries

# CUDA Architecture

- GPU Architecture finding
  - $ nvaccelinfo
  - Default Target : ccxx

```
mzc01-sunkim317 — ubuntu@ip-10-0-7-147: ~/test — ssh -p 22 -o IdentityFile yes -i ~/.ssh/hpc-gpu-aws.pem ubuntu@ec2-43-202-64-70.ap-northeast-2.co
a.out  test.f90
ubuntu@ip-10-0-7-147:~/test$ nvaccelinfo

CUDA Driver Version:           12000
NVRM version:                  NVIDIA UNIX x86_64 Kernel Module  525.85.12  Sat Jan 28 02:10:06 UTC 2023

Device Number:                 0
Device Name:                   Tesla V100-SXM2-16GB
Device Revision Number:        7.0
Global Memory Size:            16935419904
Number of Multiprocessors:     80
Concurrent Copy and Execution: Yes
Total Constant Memory:         65536
Total Shared Memory per Block: 49152
Registers per Block:           65536
Warp Size:                     32
Maximum Threads per Block:     1024
Maximum Block Dimensions:      1024, 1024, 64
Maximum Grid Dimensions:       2147483647 x 65535 x 65535
Maximum Memory Pitch:          2147483647B
Texture Alignment:             512B
Clock Rate:                    1530 MHz
Execution Timeout:             No
Integrated Device:             No
Can Map Host Memory:           Yes
Compute Mode:                  default
Concurrent Kernels:            Yes
ECC Enabled:                   Yes
Memory Clock Rate:             877 MHz
Memory Bus Width:              4096 bits
L2 Cache Size:                 6291456 bytes
Max Threads Per SMP:           2048
Async Engines:                 2
Unified Addressing:            Yes
Managed Memory:                Yes
Concurrent Managed Memory:     Yes
Preemption Supported:          Yes
Cooperative Launch:            Yes
Default Target:                cc70
ubuntu@ip-10-0-7-147:~/test$
```

# Architecture List

| Virtual Architecture Feature List | Architecture Target List | |
|---|---|---|
| compute_50, compute_52, compute_53 | sm_50, sm_52, sm_53 | Maxwell Arch Support |
| compute_60, compute_61, compute_62 | sm_60, sm_61, sm_62 | Pascal Arch Support |
| compute_70, compute_72 | sm_70, sm_71 | Volta Arch Support |
| compute_75 | sm_75 | Turing Arch Support |
| compute_80, compute_86, compute_87 | sm_80, sm_86, sm_87 | Ampere Arch Support |
| compute_89 | sm_89 | Ada Arch Support |
| compute_90, compute_90a | sm_90, sm_90a | Hopper Arch Support |

# NVIDIA HPC SDK

- System Requirements
  - NVIDIA GPU(s)
    - Pascal (sm60), Volta (sm70), Turing (sm75), Ampere (sm80), or Hopper (sm90) CUDA driver version >= 440.33 for CUDA 10.2
  - Docker 19.03 or later which includes support for the --gpus option, or Singularity version 3.4.1 or later
  - For older Docker versions, use nvidia-docker >= 2.0.3
- Running the NVIDIA HPC SDK

```
$ docker run --gpus all -it --rm nvcr.io/nvidia/nvhpc:23.5-devel-cuda_multi-ubuntu20.04
$ cd /opt/nvidia/hpc_sdk/Linux_x86_64/23.5/examples/OpenACC/samples
$ make all
```

- Running with Docker

```
$ docker run --gpus all -it --rm -v $(pwd):/host_pwd -w /host_pwd nvcr.io/nvidia/nvhpc:23.5-devel-cuda_multi-ubuntu20.04
```

- https://docs.nvidia.com/hpc-sdk/hpc-sdk-container/index.html

# End-user Environment Settings

- In csh, use these commands:

```
% setenv NVARCH `uname -s`_`uname -m`
% setenv NVCOMPILERS /opt/nvidia/hpc_sdk
% setenv MANPATH "$MANPATH":$NVCOMPILERS/$NVARCH/23.5/compilers/man
% set path = ($NVCOMPILERS/$NVARCH/23.5/compilers/bin $path)
```

- In bash, sh, or ksh, use these commands:

```
$ NVARCH=`uname -s`_`uname -m`; export NVARCH
$ NVCOMPILERS=/opt/nvidia/hpc_sdk; export NVCOMPILERS
$ MANPATH=$MANPATH:$NVCOMPILERS/$NVARCH/23.5/compilers/man; export MANPATH
$ PATH=$NVCOMPILERS/$NVARCH/23.7/compilers/bin:$PATH; export PATH
```

- Once the 64-bit compilers are available, you can make the OpenMPI commands and man pages accessible using these commands.

```
% set path = ($NVCOMPILERS/$NVARCH/23.7/comm_libs/mpi/bin $path)
% setenv MANPATH "$MANPATH":$NVCOMPILERS/$NVARCH/23.5/comm_libs/mpi/man
```

- And the equivalent in bash, sh, and ksh:

```
$ export PATH=$NVCOMPILERS/$NVARCH/23.5/comm_libs/mpi/bin:$PATH
$ export MANPATH=$MANPATH:$NVCOMPILERS/$NVARCH/23.5/comm_libs/mpi/man
```

# Break

- https://developer.nvidia.com/gpu-accelerated-libraries

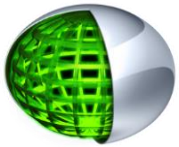
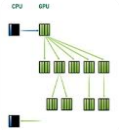
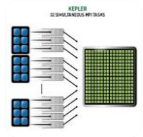
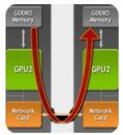
Cloud Innovator
MEGAZONE

Megazone is delivering values through the business expertise and experiences built from the Platform Service, Digital Marketing and Digital Service areas

OpenACC
More Science, Less Programming

# Introduction to the CUDA Platform

# CUDA Parallel Computing Platform

| | | | |
|---|---|---|---|
| Programming Approaches | Libraries | OpenACC Directives | Programming Languages |
| | "Drop-in" Acceleration | Easily Accelerate Apps | Maximum Flexibility |
| Development Environment |  | Nsight IDE<br>Linux, Mac and Windows<br>GPU Debugging and Profiling | CUDA-GDB debugger<br>NVIDIA Visual Profiler |
| Open Compiler Tool Chain |  | Enables compiling new languages to CUDA platform, and CUDA languages to other architectures | |
| Hardware Capabilities | SMX | Dynamic Parallelism | HyperQ | GPUDirect |
| |  |  |  |  |

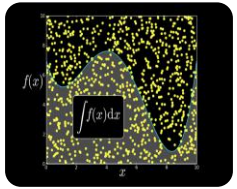
# 3 Ways to Accelerate Applications

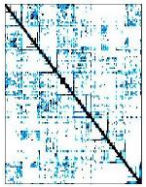
# Libraries: Easy, High-Quality Acceleration

- **Ease of use:** Using libraries enables GPU acceleration without in-depth knowledge of GPU programming
- **"Drop-in":** Many GPU-accelerated libraries follow standard APIs, thus enabling acceleration with minimal code changes
- **Quality:** Libraries offer high-quality implementations of functions encountered in a broad range of applications
- **Performance:** NVIDIA libraries are tuned by experts

# Some GPU-accelerated Libraries


NVIDIA cuBLAS


NVIDIA cuRAND

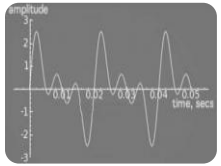

NVIDIA cuSPARSE


NVIDIA NPP

GPU VSIPL
Vector Signal Image Processing

CULA tools
GPU Accelerated Linear Algebra

MAGMA
Matrix Algebra on GPU and Multicore


NVIDIA cuFFT

ROGUE WAVE SOFTWARE
IMSL Library

ArrayFire Matrix Computations

CUSP
Sparse Linear Algebra

Thrust
C++ STL Features for CUDA

# 3 Steps to CUDA-accelerated application

- **Step 1:** Substitute library calls with equivalent CUDA library calls

  `saxpy ( … )` ► `cublasSaxpy ( … )`

- **Step 2:** Manage data locality

  - with CUDA: `cudaMalloc(), cudaMemcpy(), etc.`
  - with CUBLAS: `cublasAlloc(), cublasSetVector(), etc.`

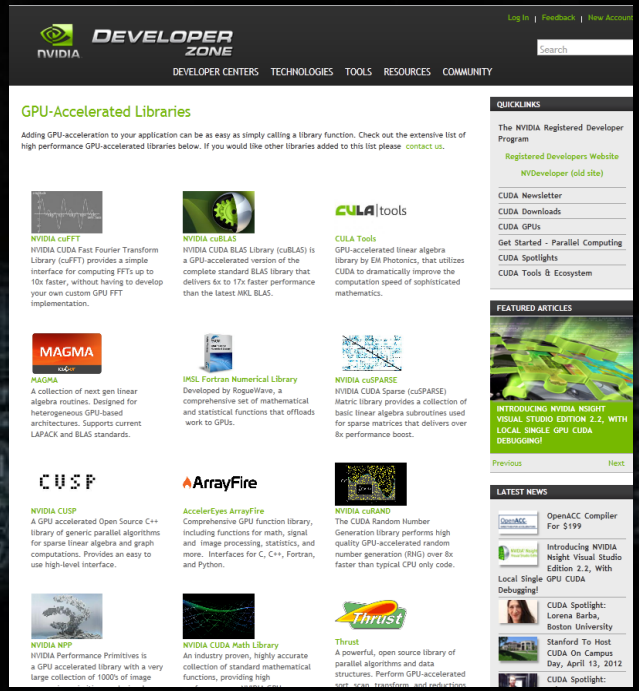
- **Step 3:** Rebuild and link the CUDA-accelerated library

```
nvcc myobj.o –lcublas
```

# Explore the CUDA (Libraries) Ecosystem

- CUDA Tools and Ecosystem described in detail on NVIDIA Developer Zone:

  developer.nvidia.com/cuda-tools-ecosystem

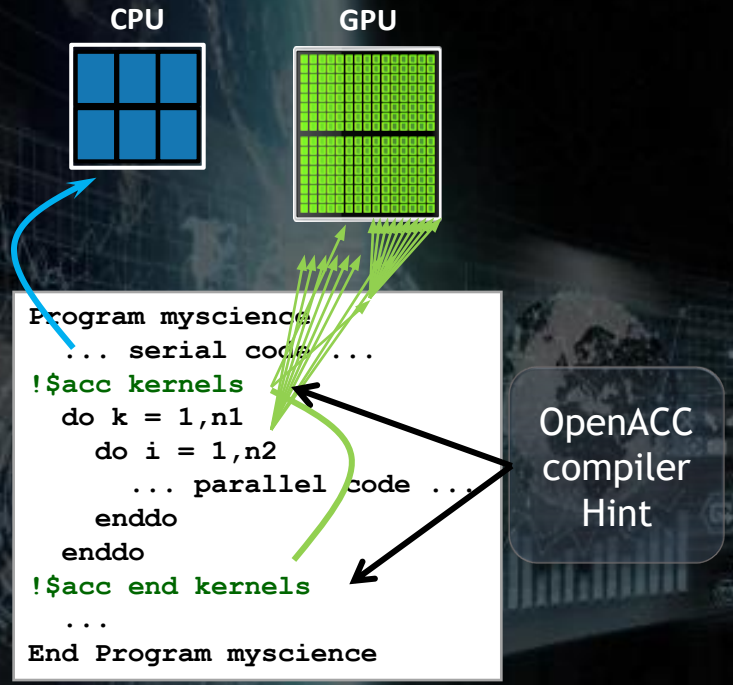


# 3 Ways to Accelerate Applications

Applications

| Libraries | OpenACC Directives | Programming Languages |
| --- | --- | --- |
| "Drop-in" Acceleration | Easily Accelerate Applications | Maximum Flexibility |

# OpenACC Directives




Simple Compiler hints

Compiler Parallelizes code

Works on many-core GPUs & multicore CPUs

# OpenACC

## The Standard for GPU Directives

OpenACC
DIRECTIVES FOR ACCELERATORS

- **Easy:** Directives are the easy path to accelerate compute intensive applications
- **Open:** OpenACC is an open GPU directives standard, making GPU programming straightforward and portable across parallel and multi-core processors
- **Powerful:** GPU Directives allow complete access to the massive parallel power of a GPU

# Directives: Easy & Powerful

## Real-Time Object Detection

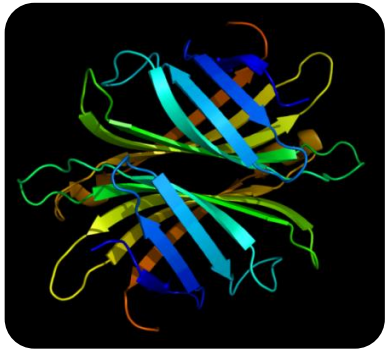
Global Manufacturer of Navigation Systems


## Valuation of Stock Portfolios using Monte Carlo

Global Technology Consulting Company


## Interaction of Solvents and Biomolecules

University of Texas at San Antonio

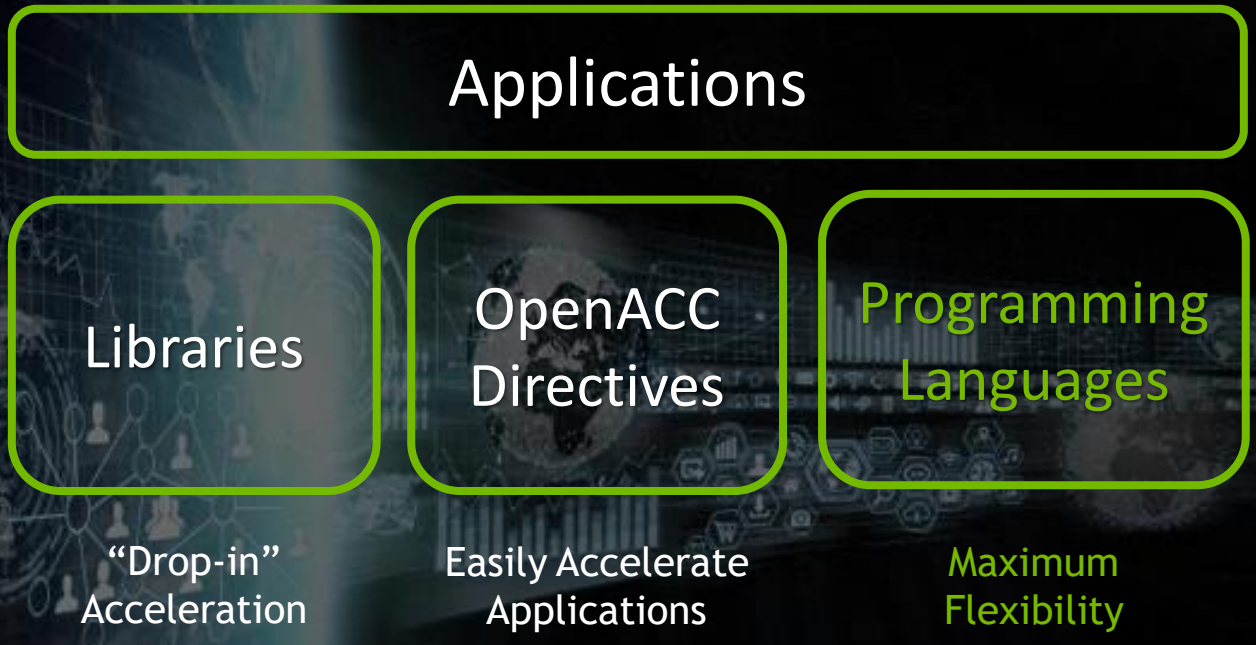


**5x** in 40 Hours **2x** in 4 Hours **5x** in 8 Hours

"Optimizing code with directives is quite easy, especially compared to CPU threads or writing CUDA kernels. The most important thing is avoiding restructuring of existing code for production applications."

-- Developer at the Global Manufacturer of Navigation Systems

# 3 Ways to Accelerate Applications



# GPU Programming Languages

| | |
|---|---|
| Numerical analytics | MATLAB, Mathematica, LabVIEW |
| Fortran | OpenACC, CUDA Fortran |
| C | OpenACC, CUDA C |
| C++ | Thrust, CUDA C++ |
| Python | PyCUDA, Copperhead |
| F# | Alea.cuBase |

# Learn More on;

These languages are supported on all CUDA-capable GPUs.

You might already have a CUDA-capable GPU in your laptop or desktop PC!

CUDA C/C++
http://developer.nvidia.com/cuda-toolkit

GPU.NET
http://tidepowerd.com

Thrust C++ Template Library
http://developer.nvidia.com/thrust

MATLAB
http://www.mathworks.com/discovery/matlab-gpu.html

CUDA Fortran
http://developer.nvidia.com/cuda-toolkit

Mathematica
http://www.wolfram.com/mathematica/new-in-8/cuda-and-opencl-support/

PyCUDA (Python)
http://mathema.tician.de/software/pycuda

# Getting Started

- Download CUDA Toolkit & SDK: www.nvidia.com/getcuda
- Nsight IDE (Eclipse or Visual Studio): www.nvidia.com/nsight
- Programming Guide/Best Practices: docs.nvidia.com
- Questions:
  - NVIDIA Developer forums: devtalk.nvidia.com
  - Search or ask on: www.stackoverflow.com/tags/cuda
- General: www.nvidia.com/cudazone

# Lab Exercises

- Login Account & PW
- module
- echo $ENV
- echo $PATH
- nvcc –version
- nvfortran –verision