# Chap11: Distributed & Parallel Computing for Deep Learning

National Tsing Hua University

2025 Fall Semester

# Outline

- Distributed Deep Learning
- Processing Acceleration for AI
- Communication and Memory Wall for AI
- Software Optimization for AI
- ML Systems

# Outline

- Distributed Deep Learning
- Processing Acceleration for AI
- Communication and Memory Wall for AI
- Software Optimization for AI
- ML Systems

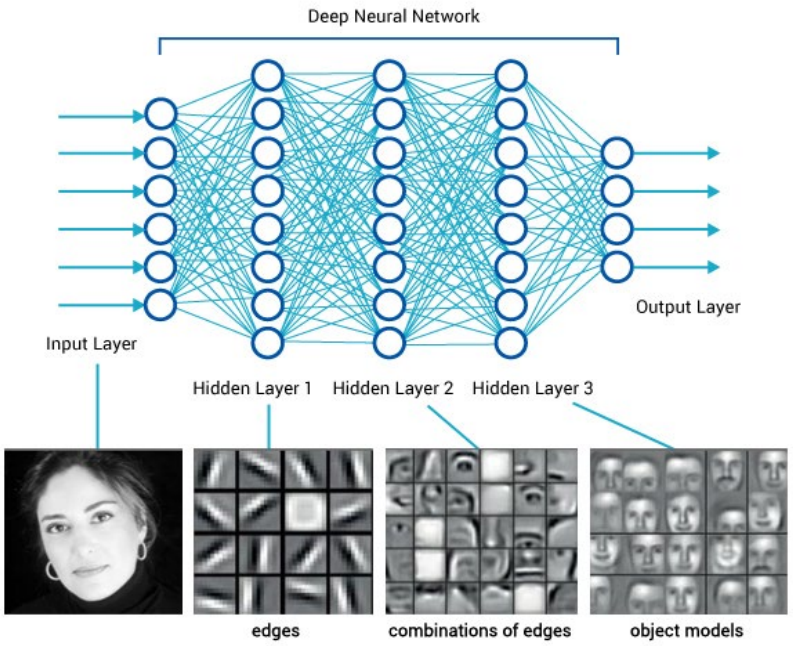
# What is Deep Learning?

- **AI**: it emphasizes the creation of intelligent machines that work and react like humans
- **Machine Learning**: it provides systems the ability to automatically learn and improve from experience without being explicitly programmed
- **Deep Learning**: a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks

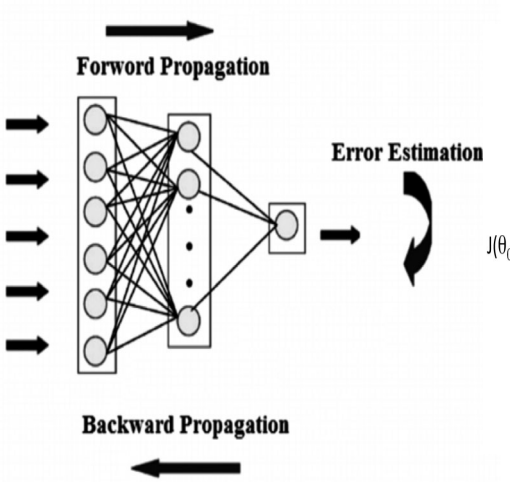

Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

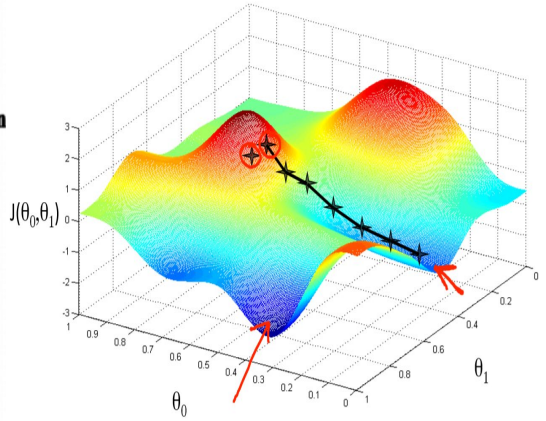
# What is Deep Learning?

- Based on universal approximation theorem
  - A model constructed with a **greedy layer-by-layer method**, such as the artificial neural network
  - Model must be trained iteratively by large set of training data using the gradient decent algorithm





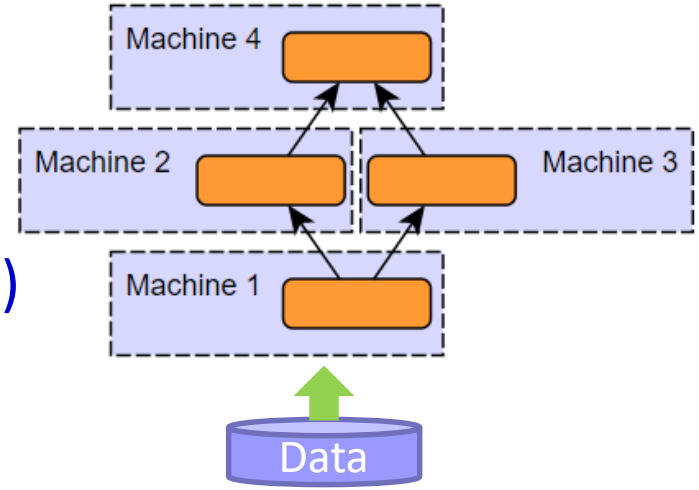


# How to Utilize Multiple Machines?

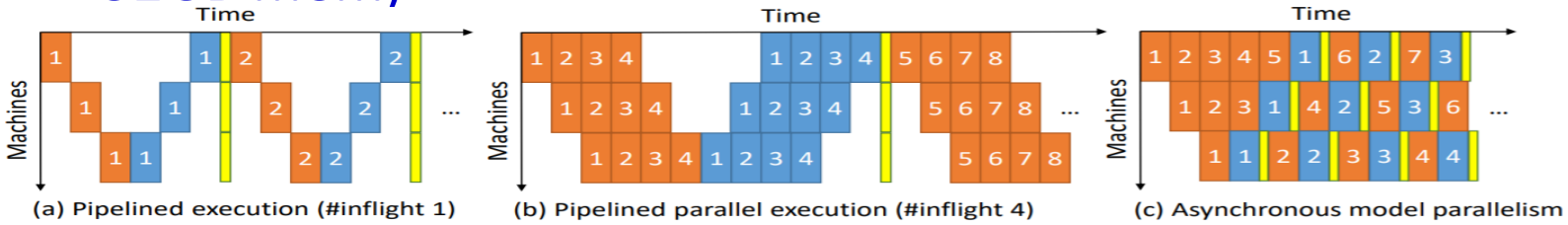
- We could utilize resources by…
  - Running multiple training jobs for different models
  - Running multiple training jobs with the same model, but different hyper-parameters
  - Running a single model training job across multiple machines ➔ **distributed training**
    - Fully utilize the resources of a system not just a single machine

# Model Parallelism



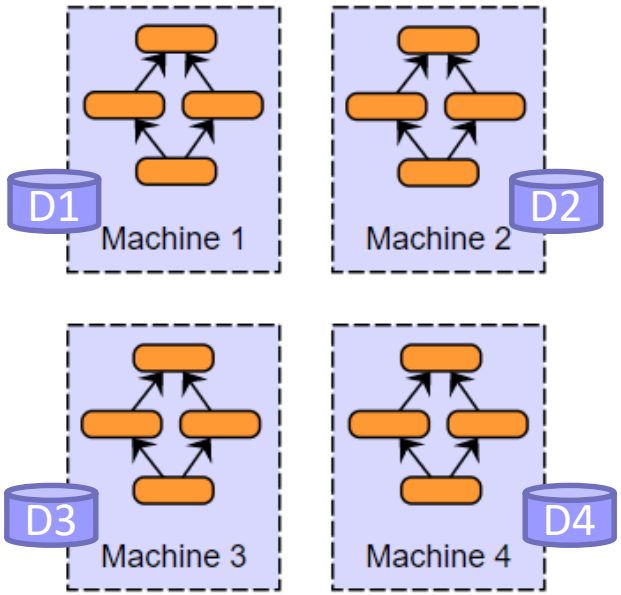

- Parallelization
  - Model is split across machines(GPUs)
  - The whole dataset is replicated
- Weakness
  - Harder to achieve good scalability

  due to synchronization point between layers (could be addressed by pipeline)
  - Model modification is equired if no shared memory
- Strength
  - More suitable on a single machine with multi-GPUs
  - The only solution when model cannot fit into a GPU (16 or 32GB mem)



# Data Parallelism

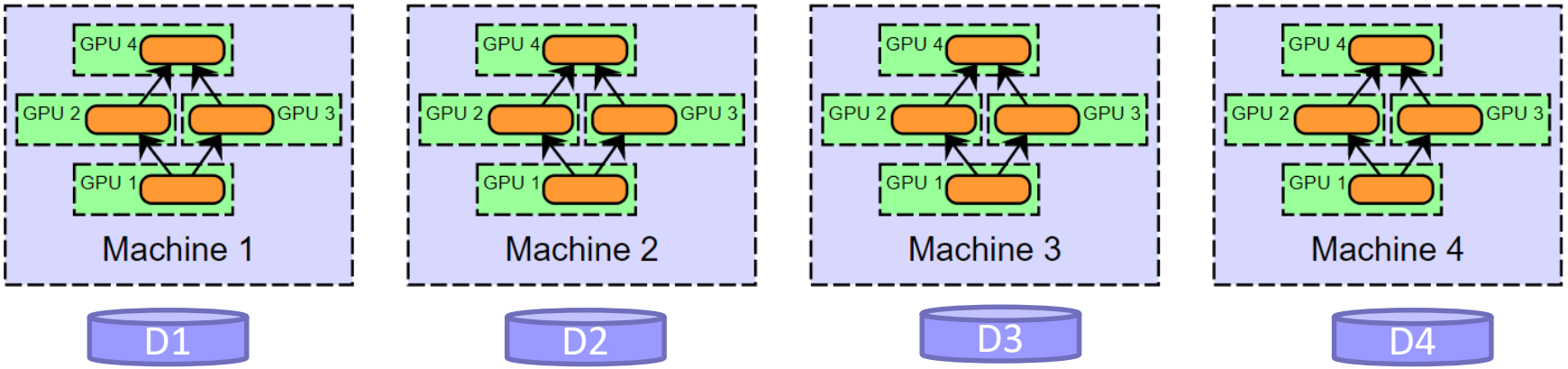
- Parallelization
  - Each machine (GPU) independently evaluate the whole model on a part of the dataset to compute gradient
  - Weight is updated by the average of gradients from all nodes
- Strength
  - Easier to achieve linear scale
  - Preferred approach for distributed systems
- Weakness
  - The whole model must fit into the memory of a node (GPU)




How to minimize the **communication overhead** of distributed stochastic gradient descent(SGD) is critical

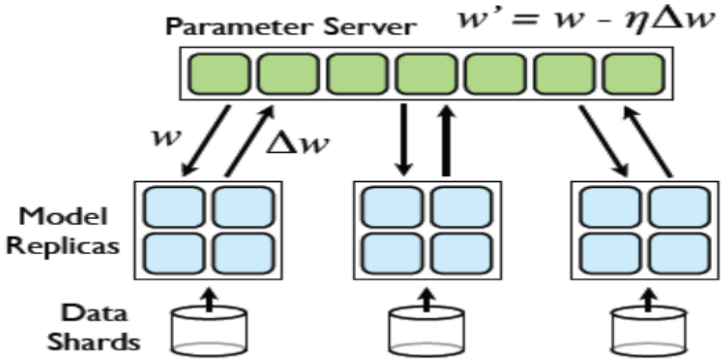
# Data + Model Parallelism

- Most commonly used solution in practice
  - Model parallelism is automated done by the compute framework
  - Data parallelism is controlled by programmers
    - Data partition
    - Parameter(weight) swapping

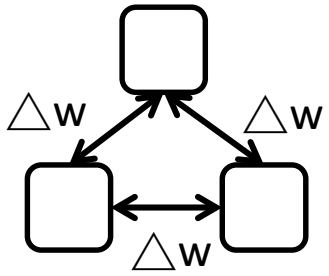


# Parameter Server vs. Allreduce

- Parameter Server (PS):
  - **De-centralized** across PS servers
  - Worker send gradient & receive weight
  - Support **both synchronized & asynchronized** SGD
  - # PS servers must be tuned
    - Too many ➔ more small messages
    - Too few ➔ network bottleneck




- Allreduce:
  - Peer to peer, **fully distributed**
  - Workers send gradient to each other, then compute weight by themselves
  - **Balanced communication load** across links
  - **Need to be synchronized SGD**

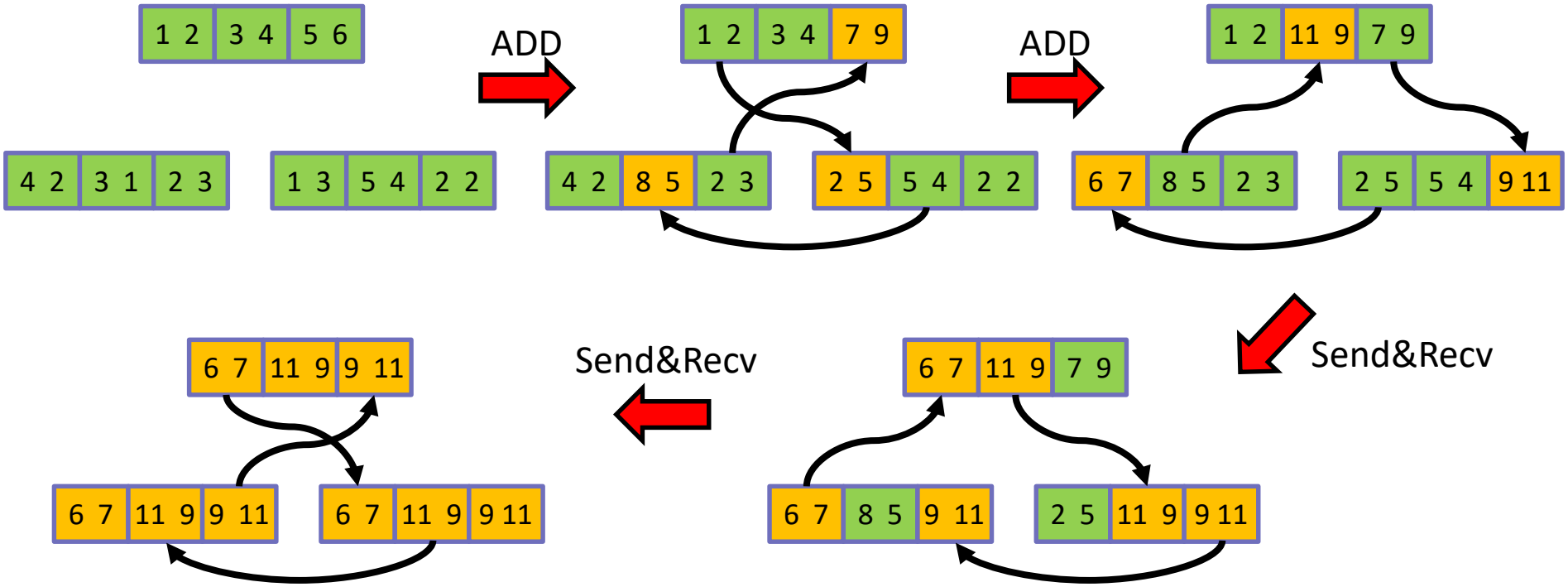


# Horovod

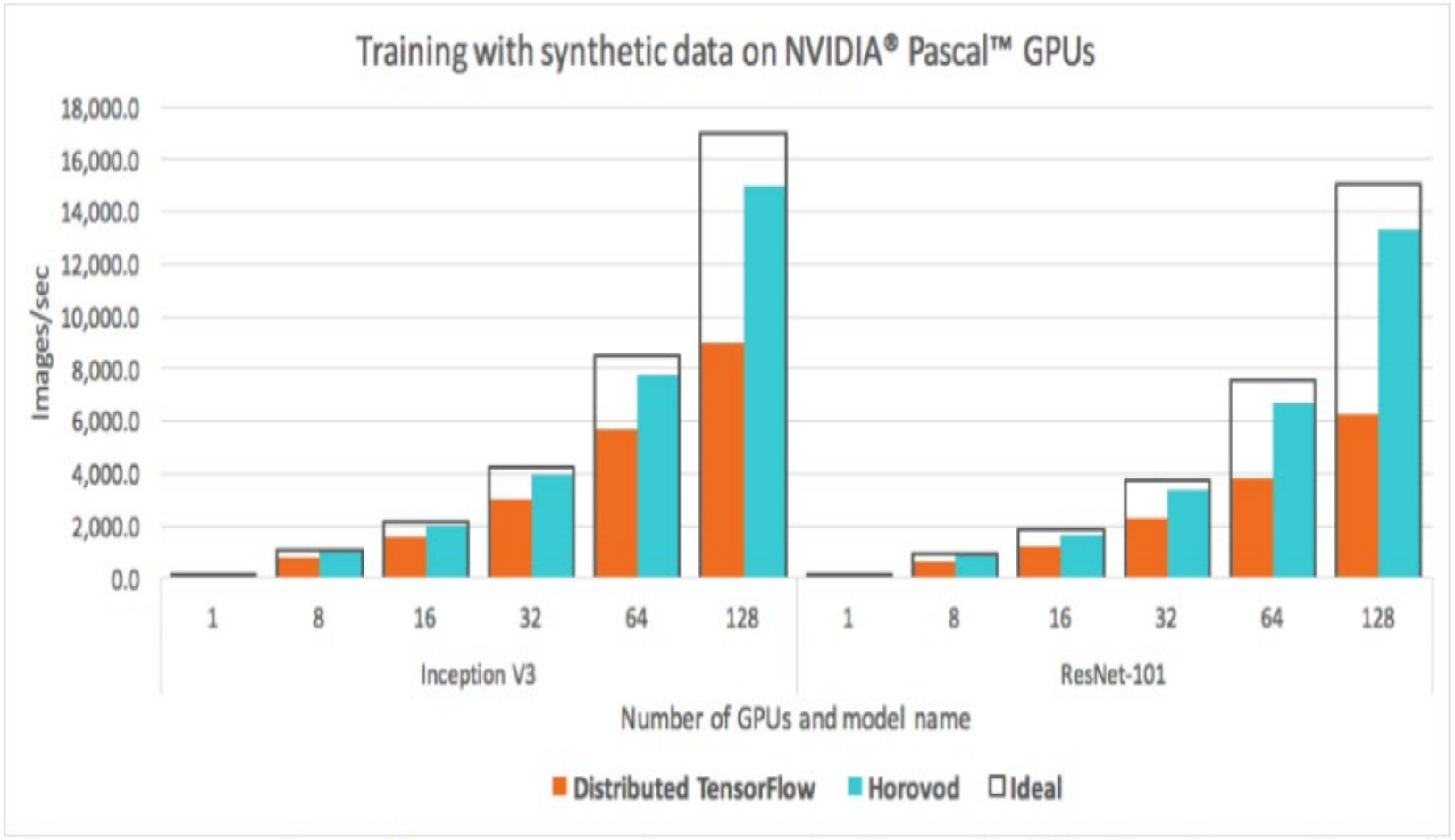

- Distributed training framework for
  - TensorFlow
  - Keras
  - PyTorch
- Separate infrastructure from ML computations
  - Executed like a traditional HPC parallel job
- Use bandwidth-optimal communication protocols
  - Implemented by HPC protocols: **MPI** and NVIDIA Collective Communications Library (NCCL)
  - Utilize RDMA (InfiniBand) if available
- Named after traditional Russian folk dance where participants dance in a circle with linked hands
- Introduction clip from UBER

# Horovod: Ring Allreduce

- An allreduce implementation that can full utilize P2P network bandwidth
  - 2*(N-1) iterations: N-1 Adds, N-1 Send&Recv

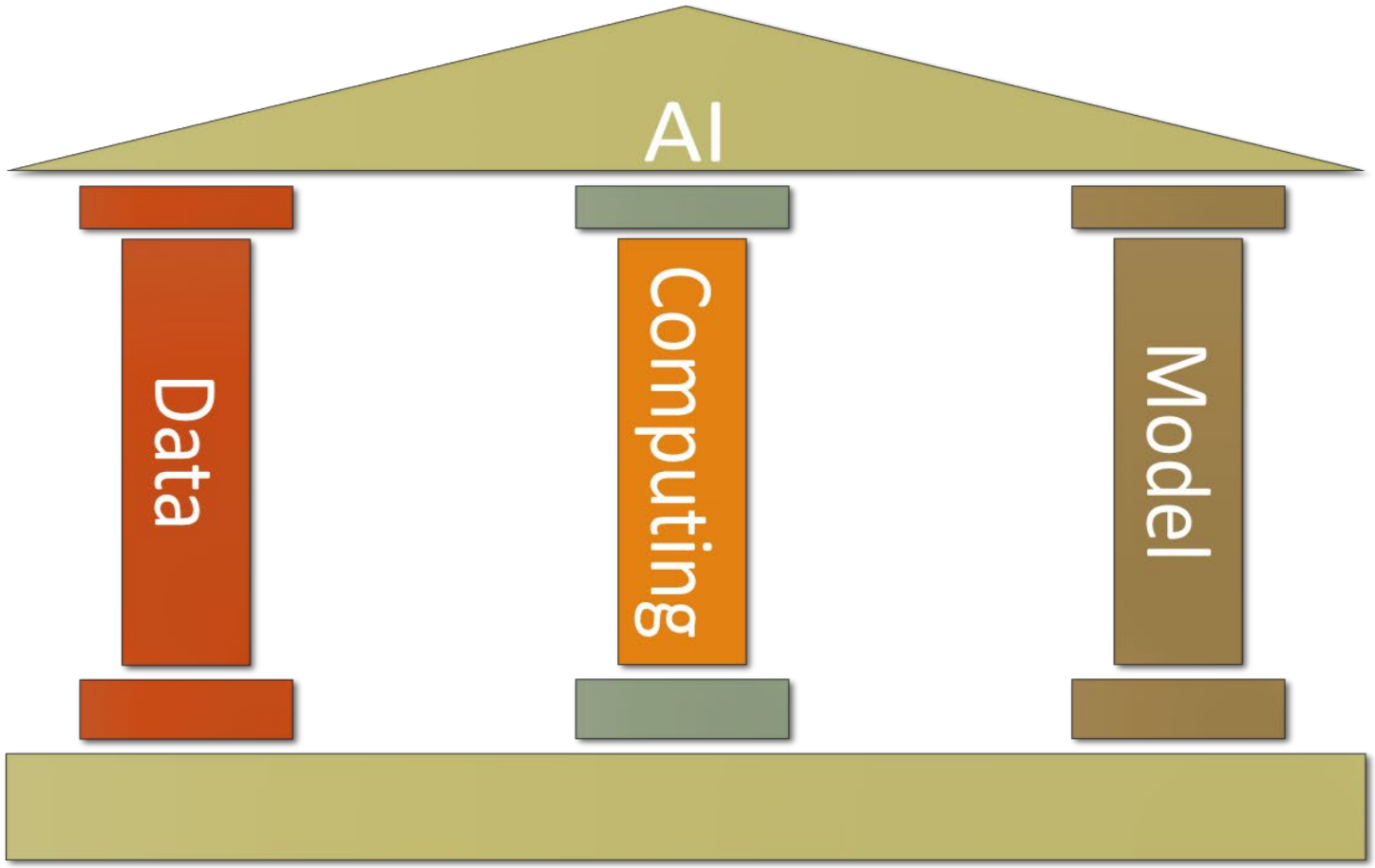


# Horovod


Training with synthetic data on NVIDIA® Pascal™ GPUs
Images/sec
18,000.0
16,000.0
14,000.0
12,000.0
10,000.0
8,000.0
6,000.0
4,000.0
2,000.0
0.0
1
8
16
32
64
128
1
8
16
32
64
128
Inception V3
ResNet-101
Number of GPUs and model name
Distributed TensorFlow
Horovod
Ideal

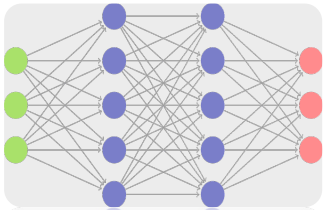
# Outline

- Distributed Deep Learning
- Processing Acceleration for AI
- Communication and Memory Wall for AI
- Software Optimization for AI
- ML Systems

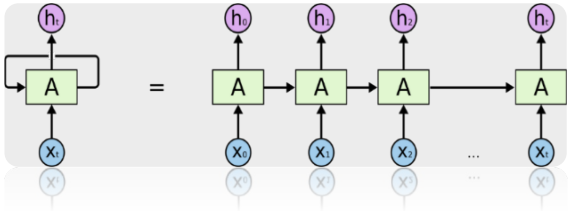



Compute is the most expensive and scarce resource

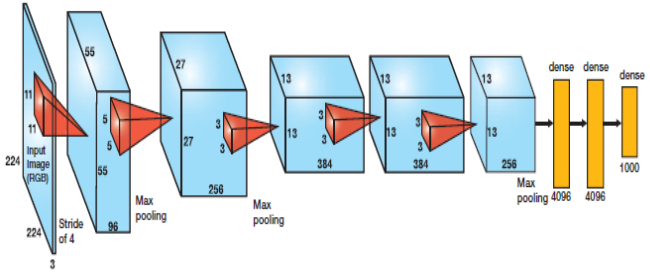
# Growing Demand for Computing



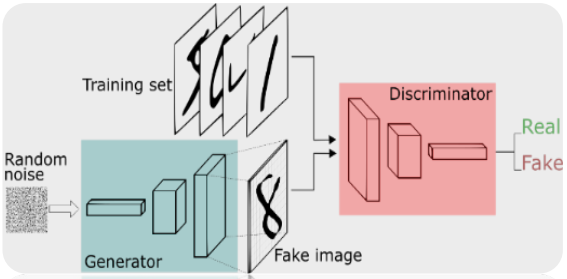
Feed forward fully connected network



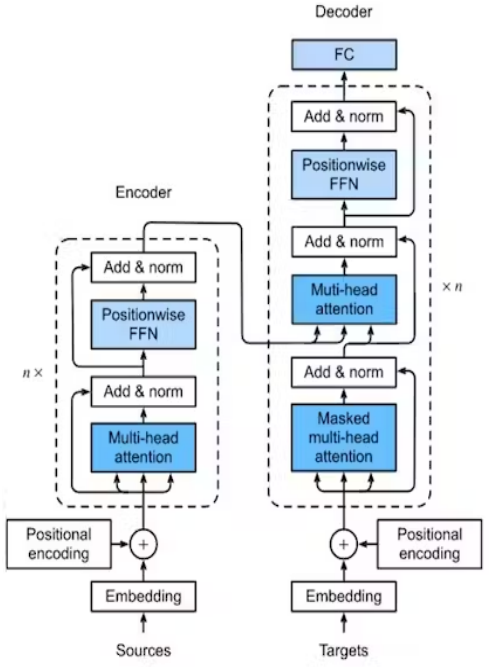

Recurrent neural network (RNN)




Convolution neural network (CNN)



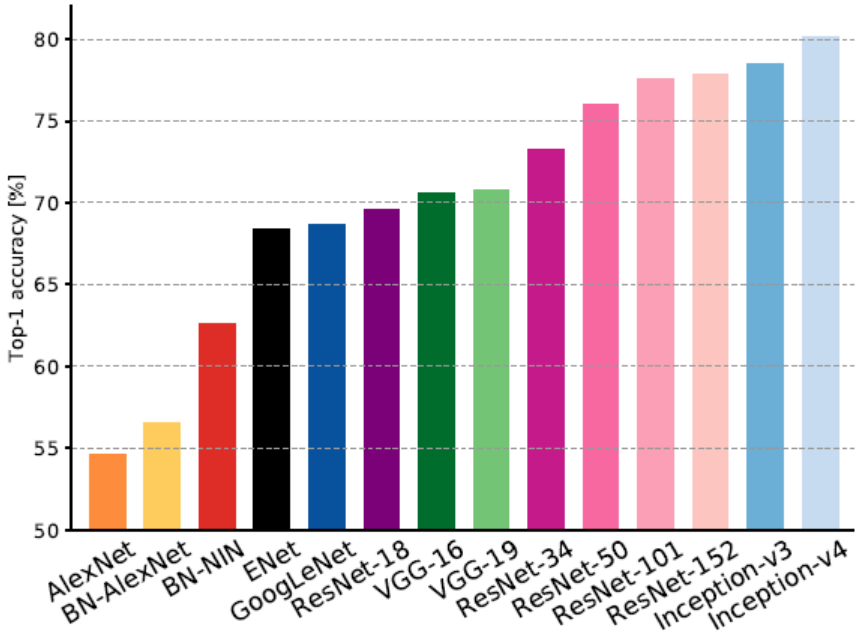

Generative Adversarial Neural network (GAN)

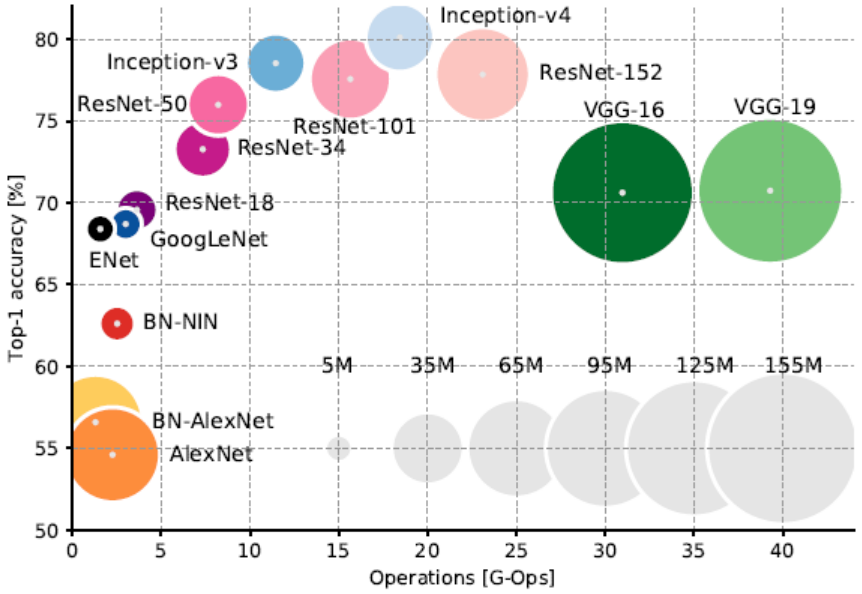



Large Language Model (LLM)

# Growing Demand for Computing

- Larger training dataset
- Larger model
- More train iterations
- More tuning parameters

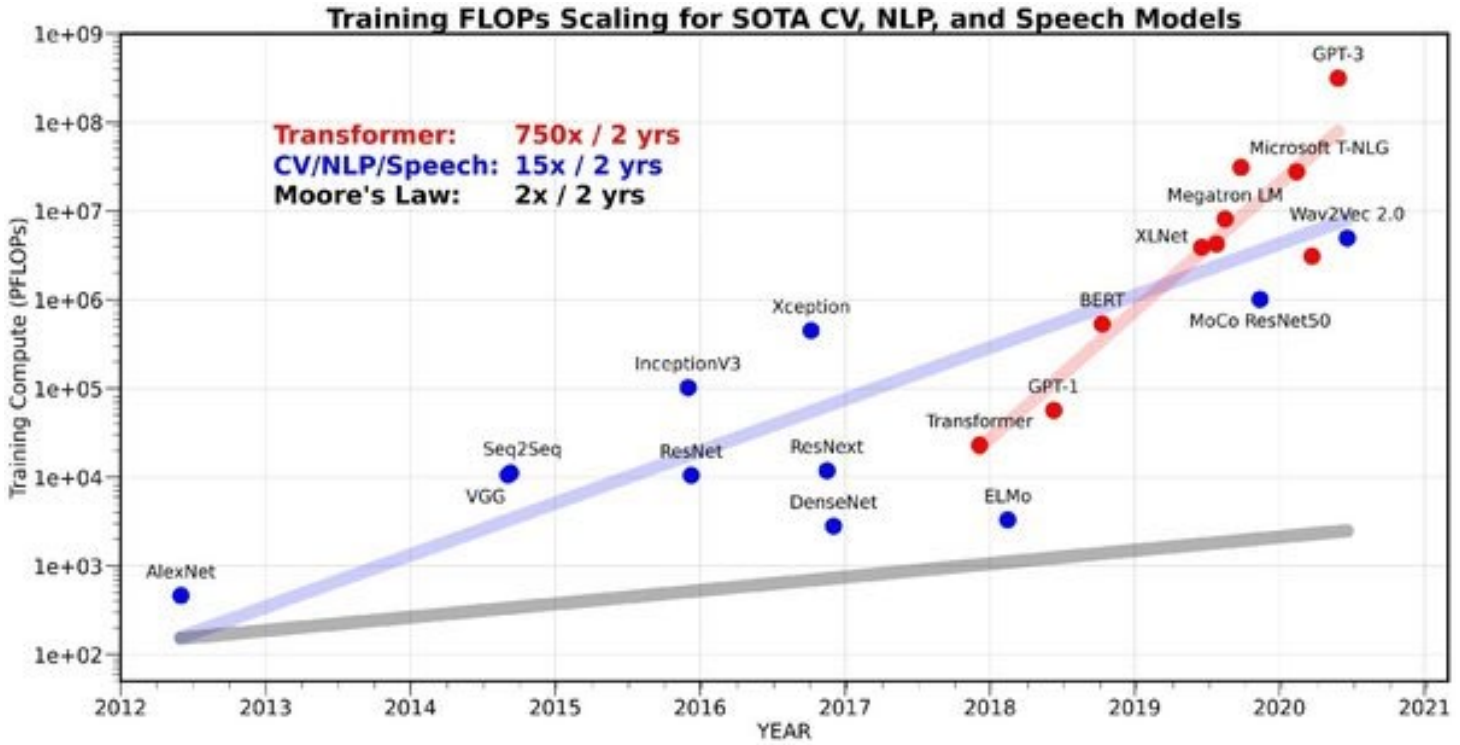



Source: Alfredo Canziani, "AN ANALYSIS OF DEEP NEURAL NETWORK MODELS FOR PRACTICAL APPLICATIONS".

# Growing Demand for Computing

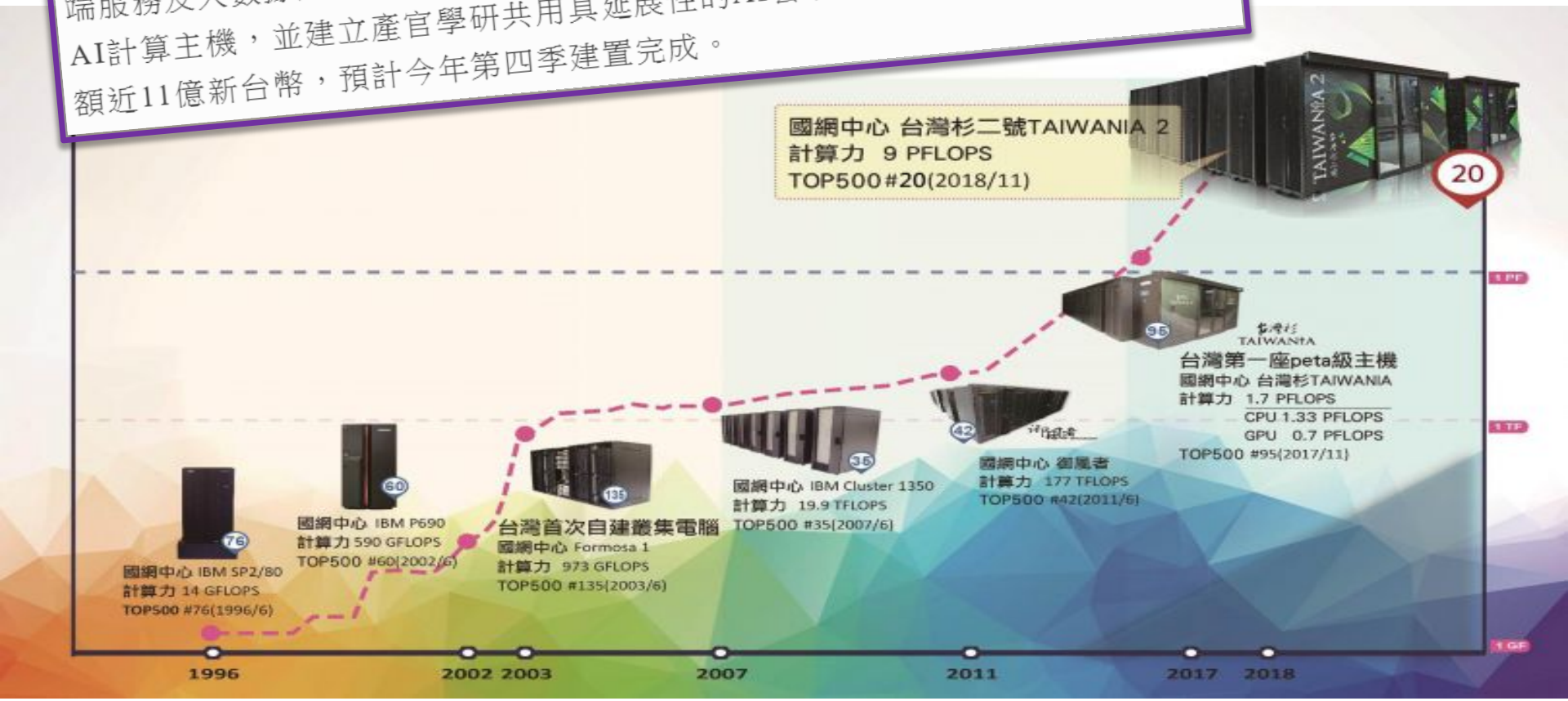
- 3.5 month doubling time since. (18 month double time for Moore's Law)
- 30K growth in 6 years



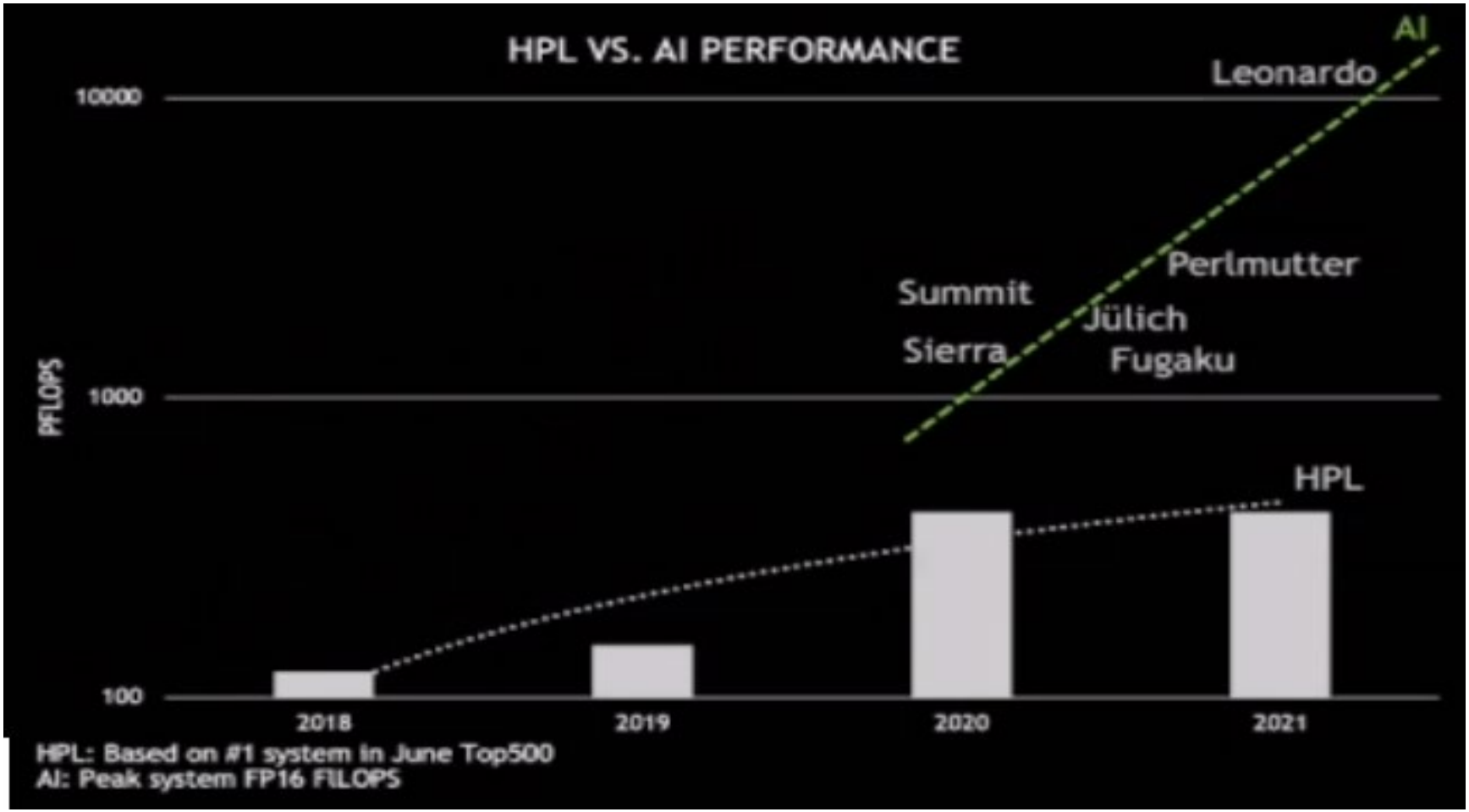

Source: openAI [https://openai.com/blog/ai-and-compute/]

# 台灣杉二號

【財訊快報／王宜弘報導】搶攻AI商機，台廠大團結！華碩( 2357 )、廣達( 2382 )以及台灣大( 3045 )結盟組成「台灣人工智慧A Team」，成軍後首戰告捷！週一(30日)三方共同宣布取得國家實驗研究院國家高速網路與計算中心「雲端服務及大數據運算設施暨整合式階層儲存系統建置案」，將協助建置新一代的AI計算主機，並建立產官學研共用具延展性的AI雲端大資料計算平台，建置總金額近11億新台幣，預計今年第四季建置完成。



# Supercomputers for AI



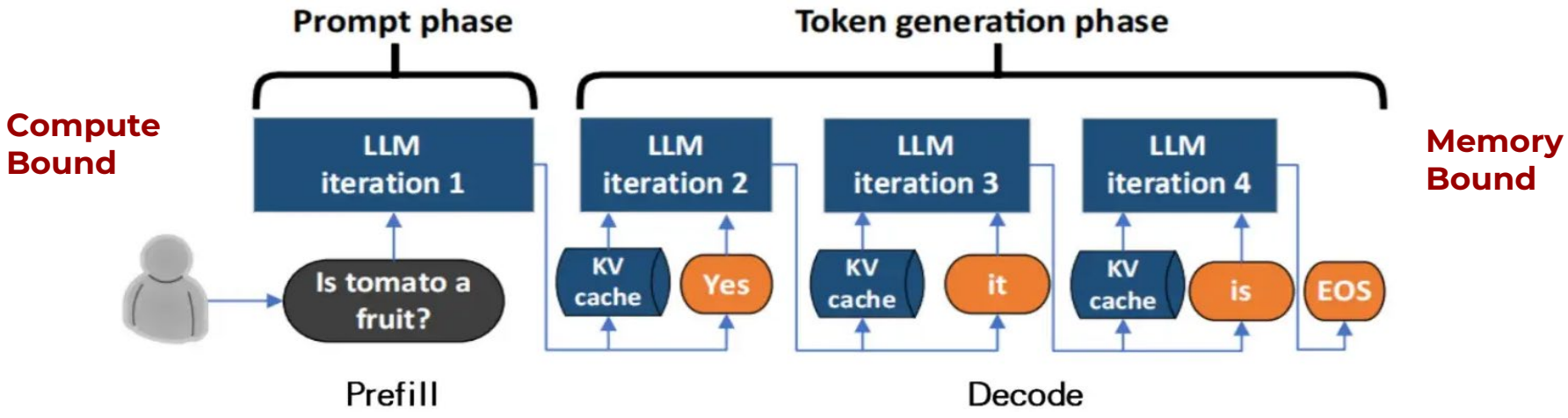
# ML Benchmark: MLPref (https://mlperf.org/)

- A benchmark suite measures how fast systems can train models to a target quality metric

| Area | Benchmark | Dataset | Quality Target | Reference Implementation Model |
|---|---|---|---|---|
| Vision | Image classification | ImageNet | 75.90% classification | ResNet-50 v1.5 |
| Vision | Object detection (light weight) | COCO | 23.0% mAP | SSD |
| Vision | Object detection (heavy weight) | COCO | 0.377 Box min AP and 0.339 Mask min AP | Mask R-CNN |
| Language | Translation (recurrent) | WMT English-German | 24.0 Sacre BLEU | NMT |
| Language | Translation (non-recurrent) | WMT English-German | 25.00 BLEU | Transformer |
| Language | NLP | Wikipedia 2020/01/01 | 0.712 Mask-LM accuracy | BERT |
| Commerce | Recommendation | 1TB Click Logs | 0.8025 AUC | DLRM |
| Research | Reinforcement learning | Go | 50% win rate vs. checkpoint | Mini Go (based on Alpha Go paper) |

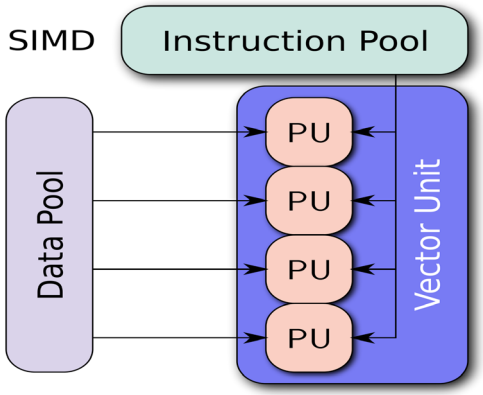
# LLM (Large Language Model) Inference Performance

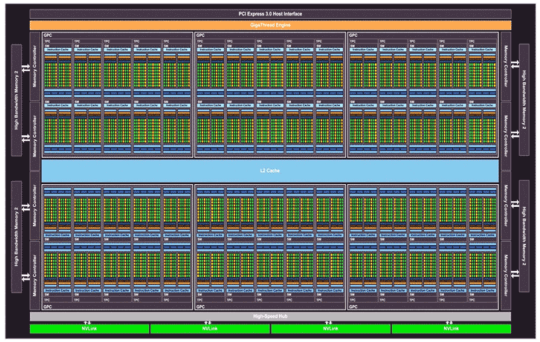
- Time to first token latency (TTFT): measuring the time taken for the LLM to output the first generated token to the user.
- Time per output token (TPOT): measuring the average latency between two subsequent generated tokens.



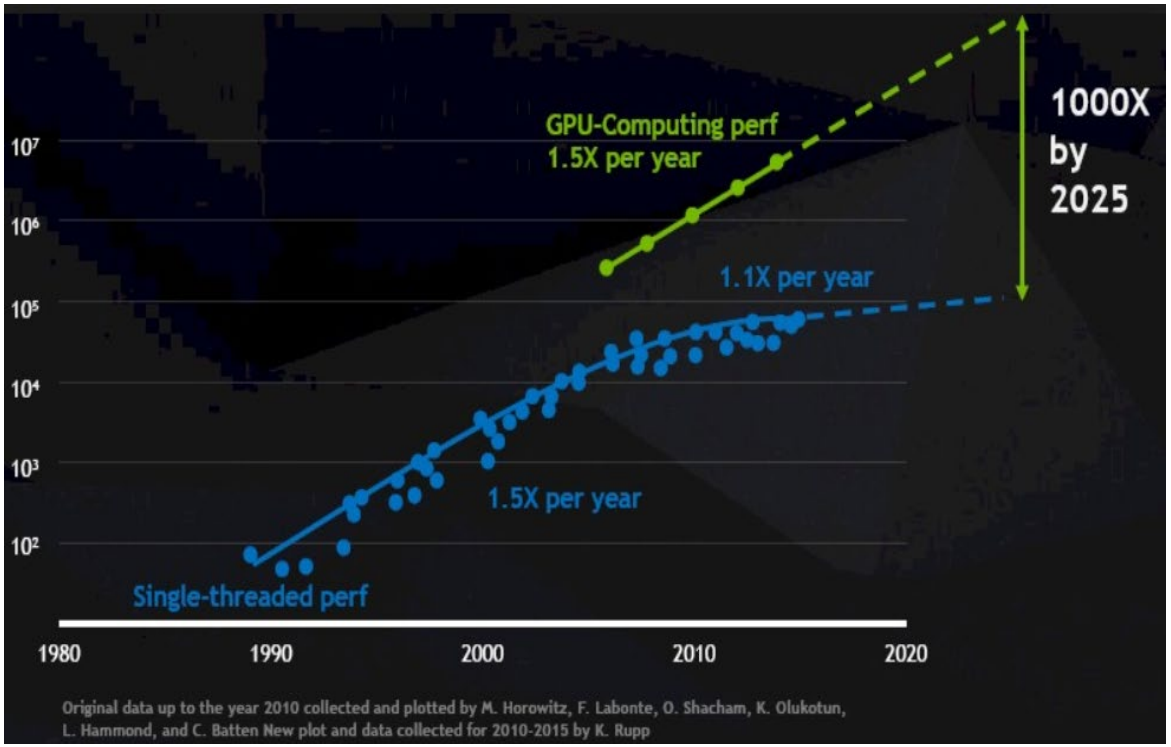
# Many-Core Processor: GPU

- Accelerator based on SIMD processor architecture



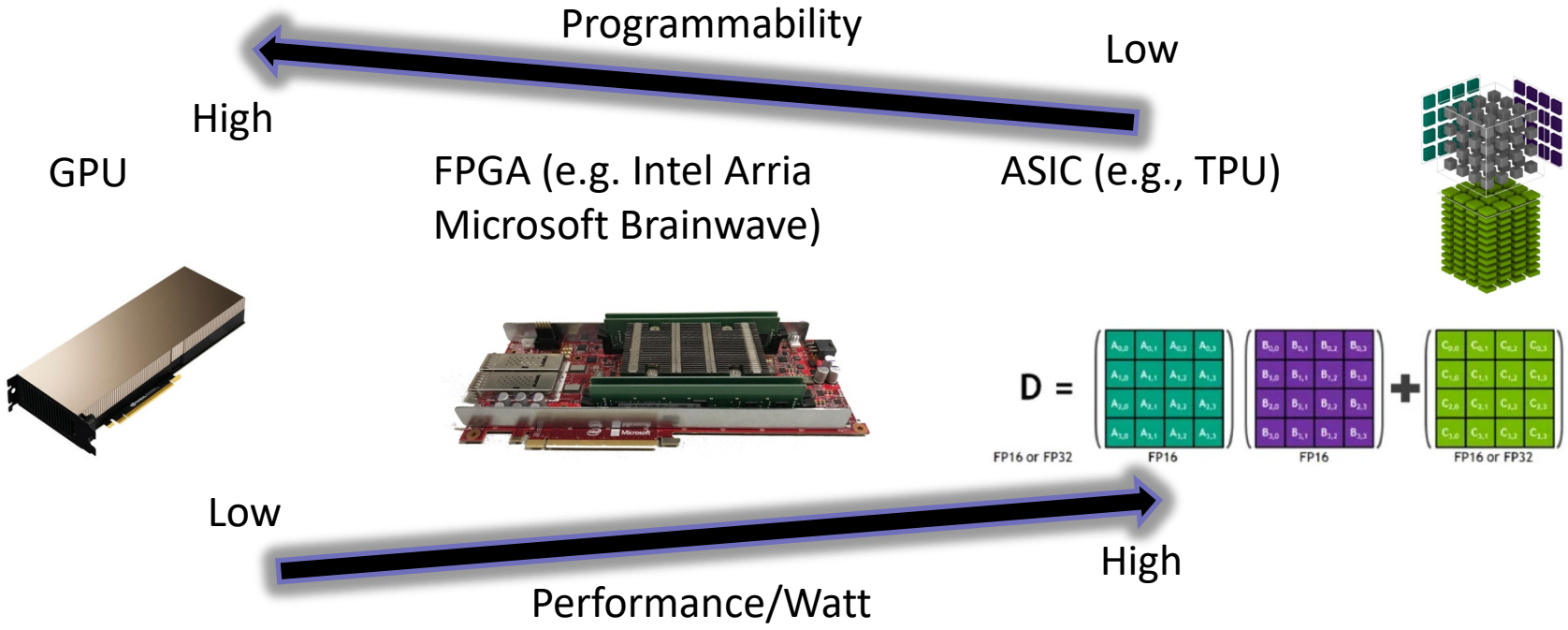
5,120 cores in a GPU




Images from Nvidia

# Specialized AI Chip

- Lower cost , Smaller size, Lower power consumption, Higher performance

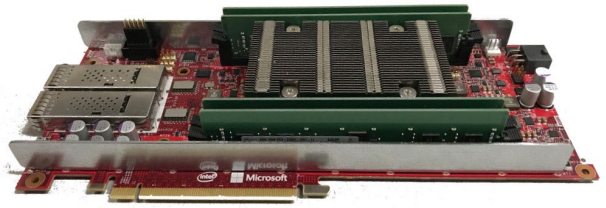


# AI Chip for Inference

- Co-design of the network structure and hardware architecture
  - AI Chip: dedicated “Tensor Accelerator”, like TensorCore
- Trade accuracy for energy & cost saving
  - model reduction, low precision computations
- Domain-specific, rather than application-specific
  - A new chip can be used more broadly across multiple applications by reconfiguration

Google Cloud TPU Pod (Hot Chips 2017)

Google’s TPU POD (ASIC)



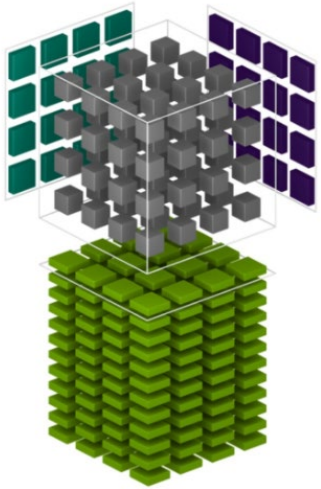
Microsoft's BrainWave (FPGA)

# Google Tensor Processing Unit (TPU)

- Specifically for deep learning (tensorflow framework)
- 30–80X higher performance-per-watt than contemporary CPUs and GPUs
  - Only for **reduced precision computation** (e.g. 8-bit precision)
  - Matrix Multiplier Unit: use a to achieve hundreds of thousands of **matric operation** in a single clock cycle
  - Systolic array: The ALUs perform **only multiplications and additions in fixed patterns**
- Reference
  - https://cloud.google.com/blog/big-data/2017/05/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu

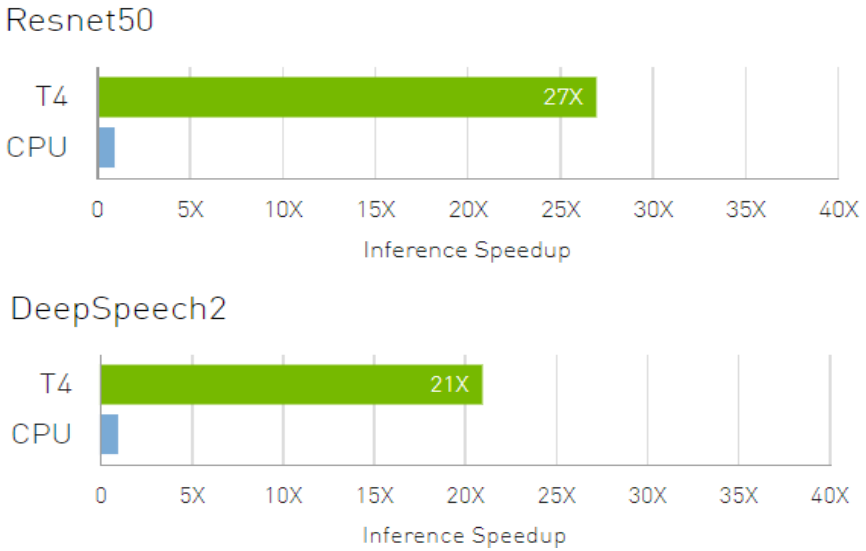
# TensorCore

- Supported after Volta architecture
  - 640 TensorCore in Tesla V100 ➔ 120 TFLOPS (16FLOPS on GPU core)
- Accelerate large matrix operations
  - perform mixed-precision matrix multiply and accumulate calculations in a single operation.
  - An common operation in most NN computations

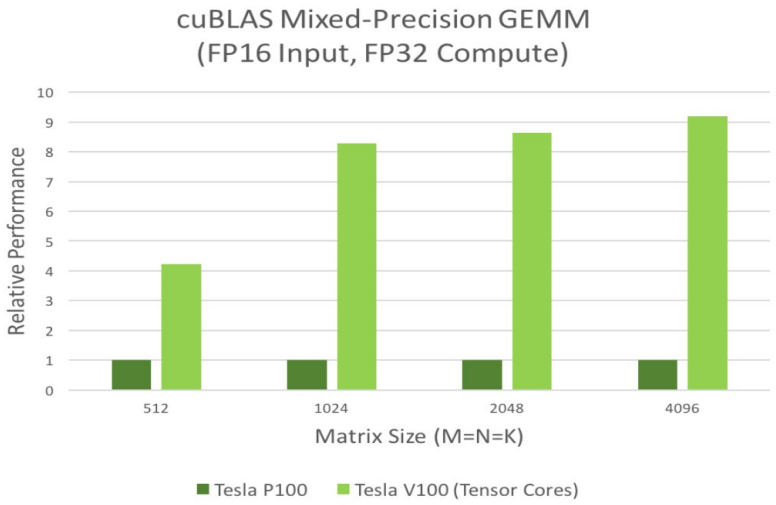


# TensorCore

- Enables massive increases in throughput and efficiency
  - T4 has the world's highest inference efficiency, up to 40X higher performance compared to CPUs with just 60% power consumption.
- Currently support in Caffe, MXNet, PyTorch, Theano, TensorFlow
  - But not for CNTK、Chainer、Torch




Chip-to-chip GPU-to-CPU speedups | NVIDIA Tesla T4 GPU vs Xeon Gold 6140 CPU



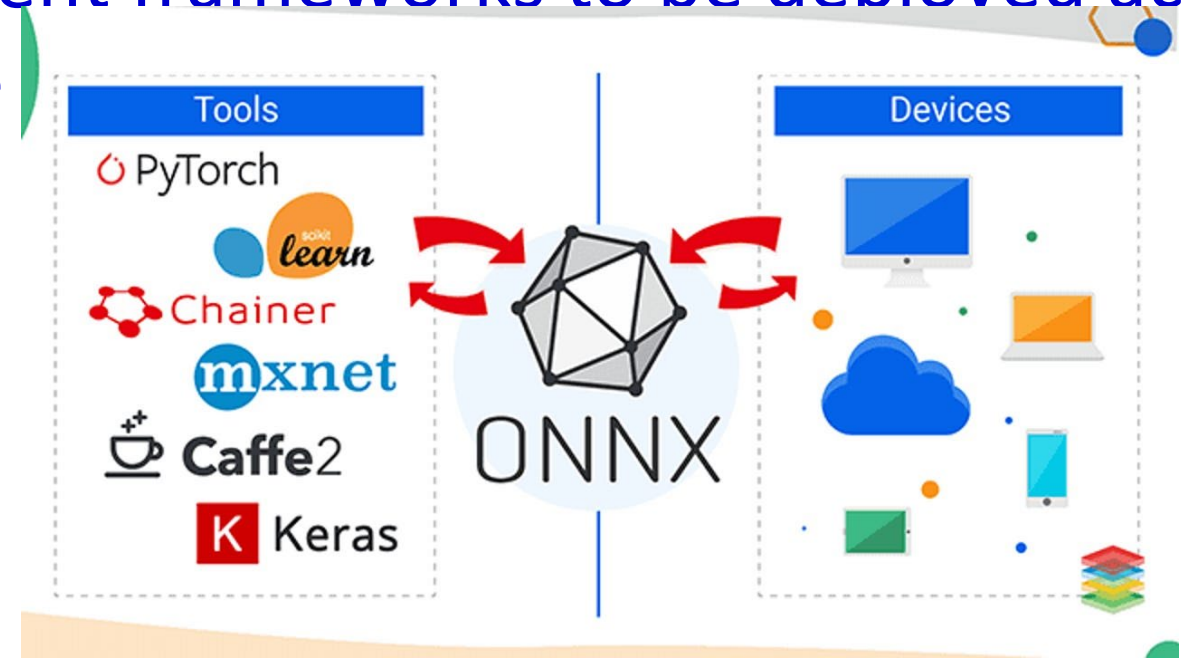

Input matrices are half precision, computation is single precision.

# AI Chip for Inference

- "memory wall" problem
  - Increase the capacity of the on-chip memory and brings it closer to the computing units
  - Compute-capable memory referred to as processing-in-memory (PIM)
- Lack of general software toolchain to efficiently translate different machine learning tasks and neural networks into executable binary codes, running on the AI chips
  - Neural network pruning, weight compression and dynamic quantization

# Interoperability

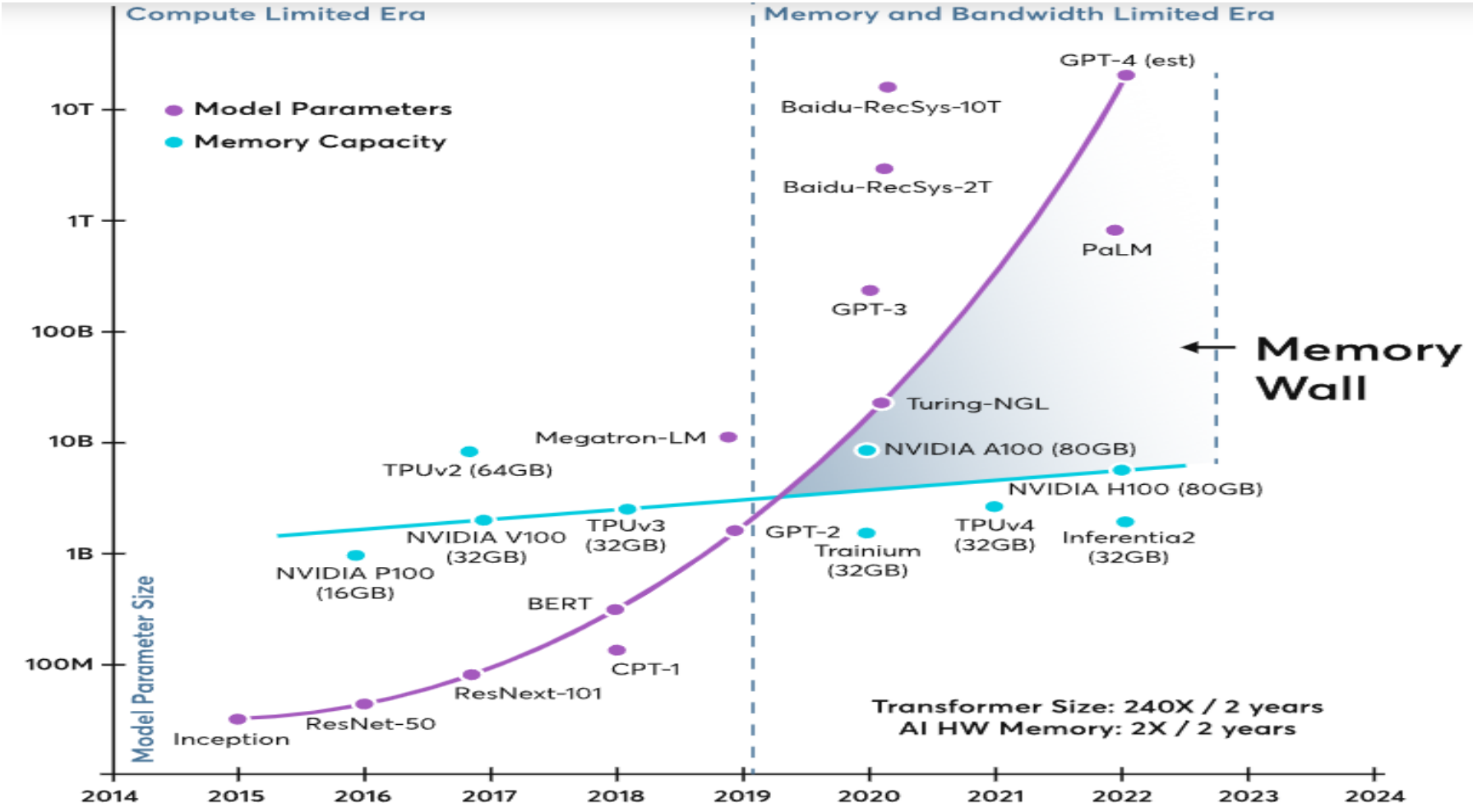
- ONNX is an interoperability layer that enables machine learning models trained using different frameworks to be deployed across a range



# Outline

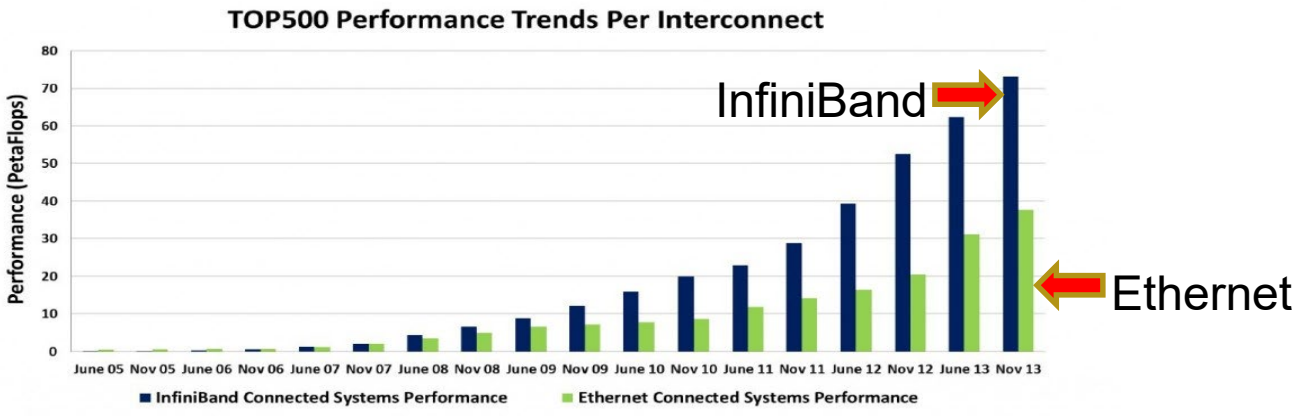
- Distributed Deep Learning
- Processing Acceleration for AI
- Communication and Memory Wall for AI
- Software Optimization for AI
- ML Systems

# Memory Wall

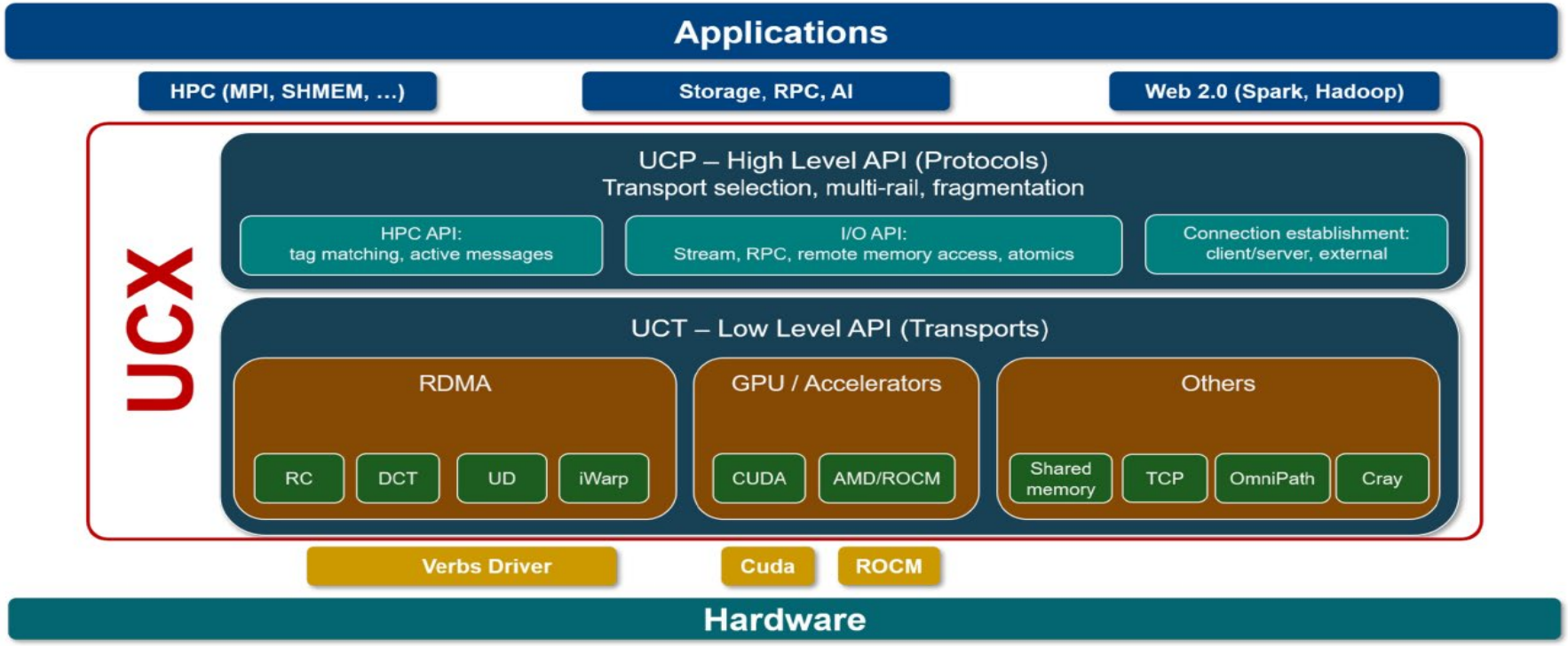

Compute Limited Era
Memory and Bandwidth Limited Era
Model Parameters
Memory Capacity
Model Parameter Size
10T
1T
100B
10B
1B
100M
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
GPT-4 (est)
Baidu-RecSys-10T
Baidu-RecSys-2T
PaLM
GPT-3
Turing-NGL
Megatron-LM
NVIDIA A100 (80GB)
TPUv2 (64GB)
NVIDIA H100 (80GB)
NVIDIA V100 (32GB)
TPUv3 (32GB)
GPT-2
Trainium (32GB)
TPUv4 (32GB)
Inferentia2 (32GB)
NVIDIA P100 (16GB)
BERT
CPT-1
ResNext-101
ResNet-50
Inception
Memory Wall
Transformer Size: 240X / 2 years
AI HW Memory: 2X / 2 years

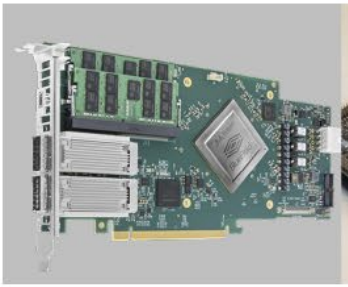
# Network Device: InfiniBand

- A computer network communications link used in high-performance computing featuring very high throughput
- It is the most commonly used interconnect in supercomputers
- Manufactured by Mellanox



# UCX High-level Overview



BlueField DPU

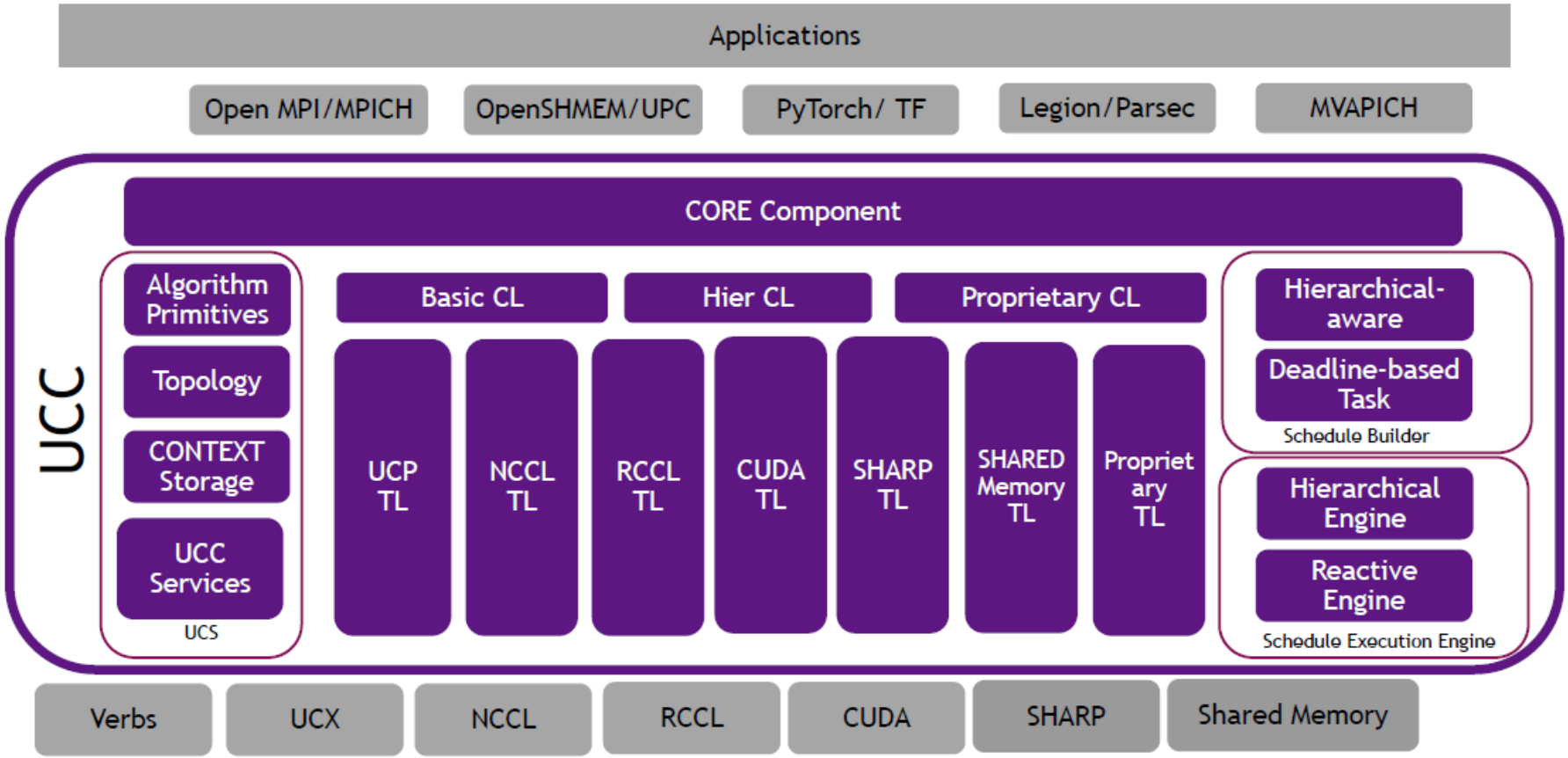
NVIDIA Jetson

Arm ThunderX2

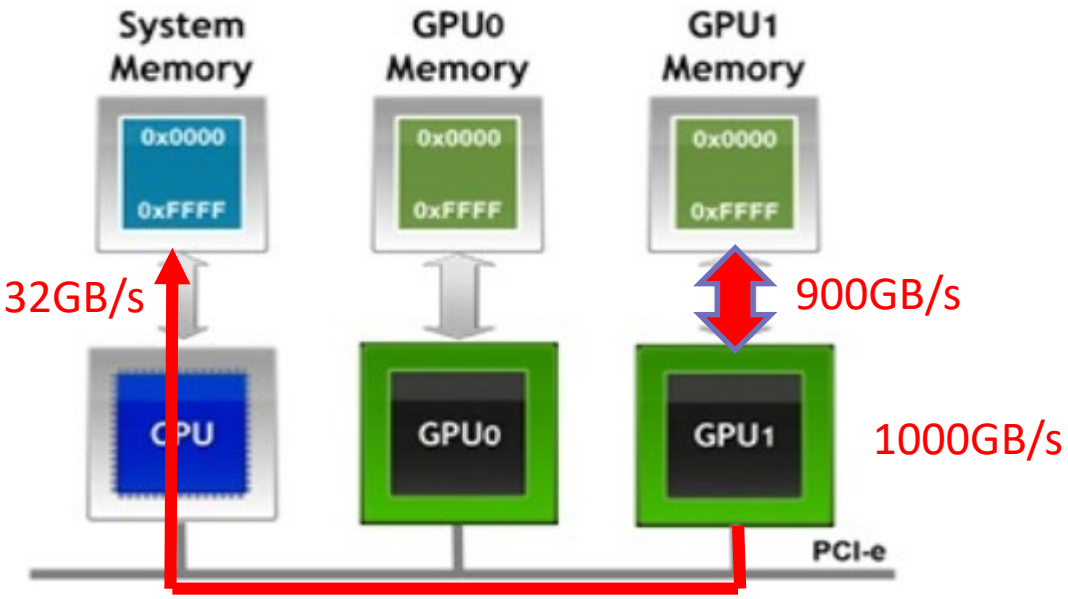
Odroid C2

N1 SDP

# UNIFIED COLLECTIVE COMMUNICATION (UCC) Architecture



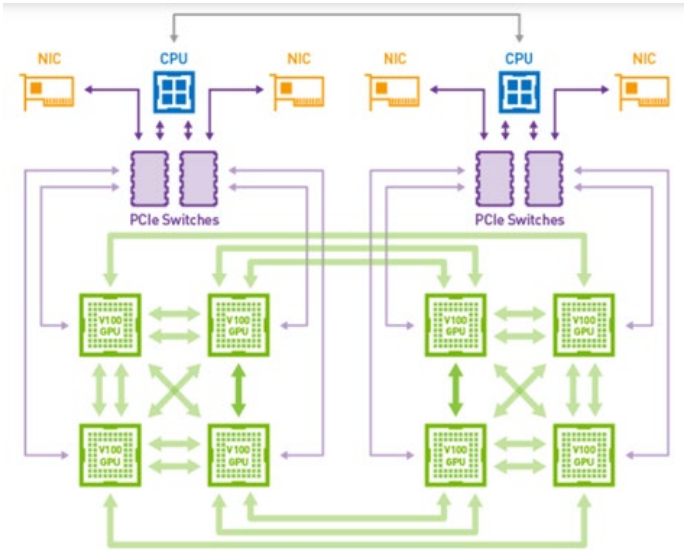
# GPU: Memory Access Bottleneck




- GPU is capable of processing 1,000GB/s data
- GPU internal memory access can reach 900GB/s
- But **PCI-E Gen4** only provide 32GB/s bandwidth

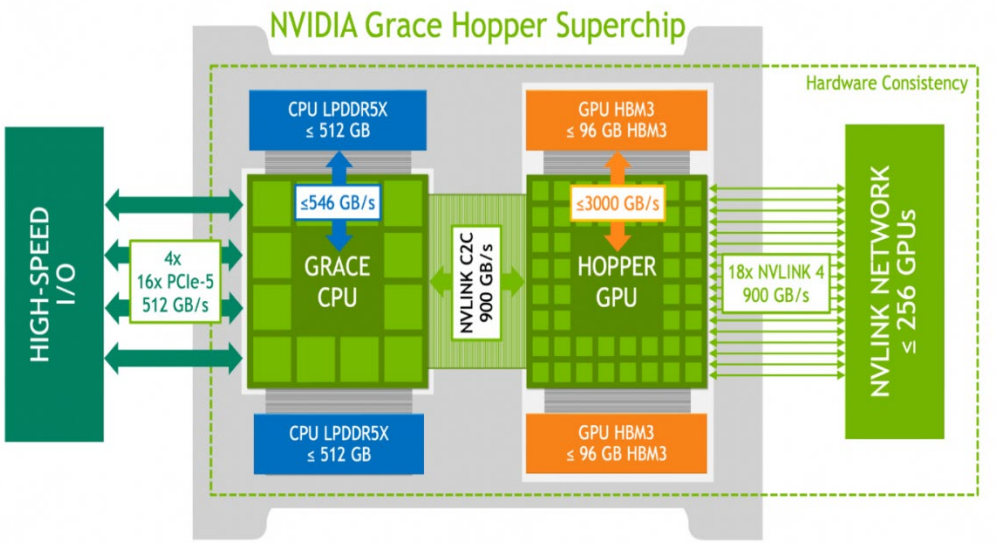
# NV-Link Fabric

- A high-bandwidth, energy-efficient interconnect that enables ultra-fast communication between GPUs
- NV-Link 5.0 Achieve 1.8TB/s bandwidth
  - 5 to 12 times faster than the PCIe (PCIe6.0 only has 242 GB/s)
- Use the SXM GPU module or NV-bridge

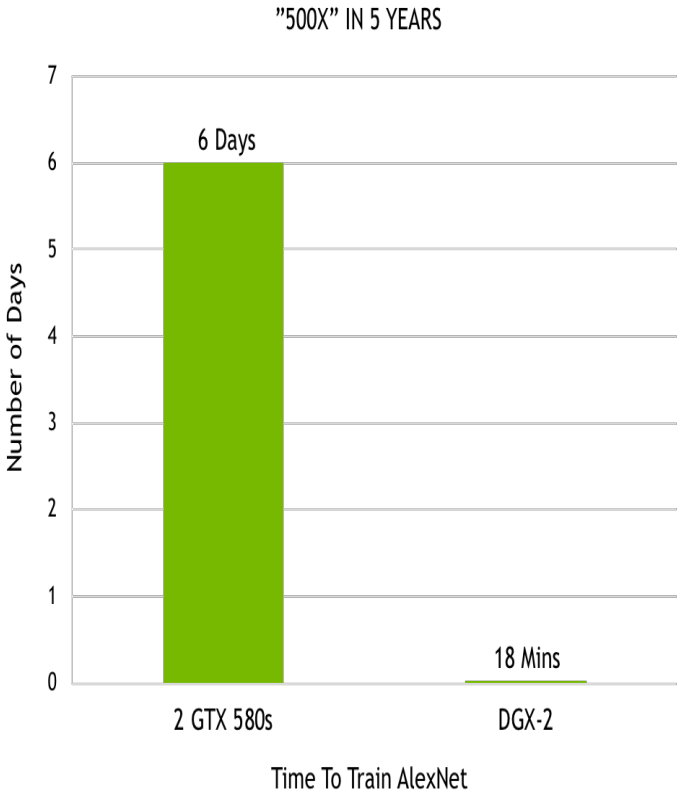



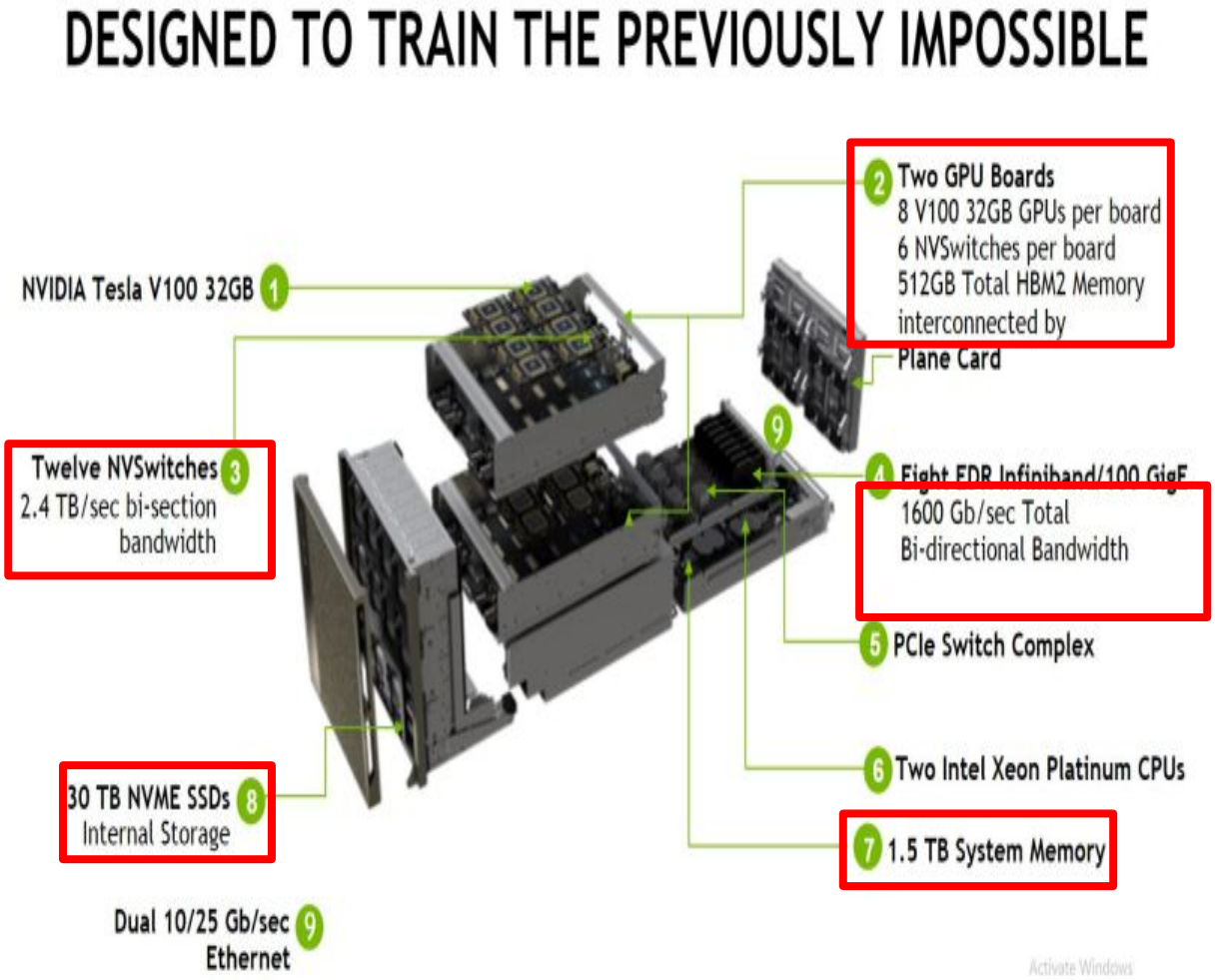
# NVIDIA Grace Hopper Superchip

- Massive Bandwidth for Compute Efficiency
- Using NVIDIA® NVLink®-C2C to deliver a CPU+GPU coherent memory model for accelerated AI and HPC applications.
- High-speed I/O
- HBM3 memory
- On-chip fabrics
- System-on-chip (SoC)
- ARM-based processors



# DGX-2 GPU Server



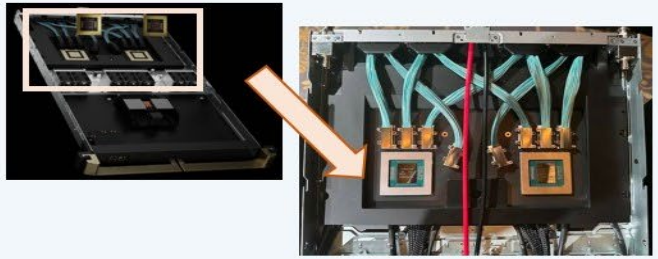

Images from Nvidia

# Nvidia 新一代 AI Server 架構大翻新

**DGX GB200 NVL72** 單一機架改採不同的設計方式，卡板彼此於機背後以高達**5,000**條的銅纜連接


## 2. Switch Tray(共9個)

每個Switch Tray包含 1片 Switch Board



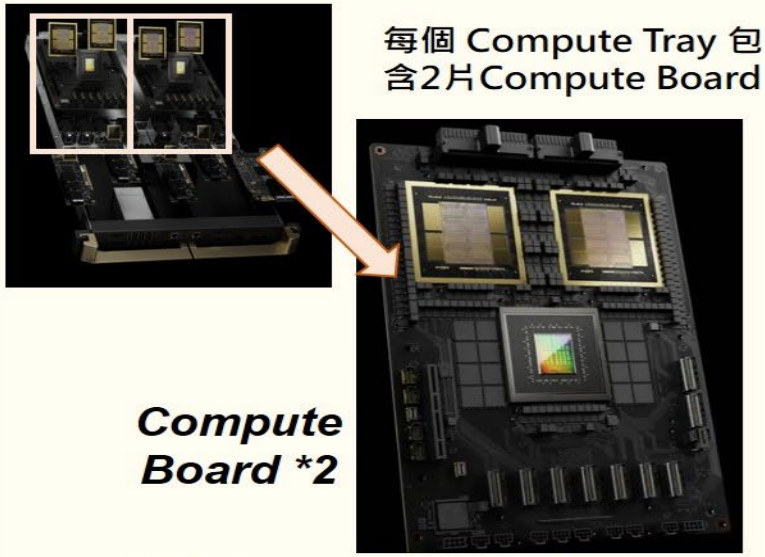
**Switch Board*1**

每片 Switch Board 包含2顆Switch晶片

第五代 **NVLink** 可擴充互連技術，提供 **Compute Board** 間快速流暢地的通訊

## 3. Compute Tray(共18個)

每個 Compute Tray 包含2片Compute Board



***Compute Board *2***

每片 **Compute Board**包含：2顆 **Blackwell GPU**與1顆**Grace CPU**

專為執行人工智慧、雲端和高效能運算 **(HPC)** 應用程式的現代資料中心而設計

## 1. Networking 網路平台

包含以下子系統：

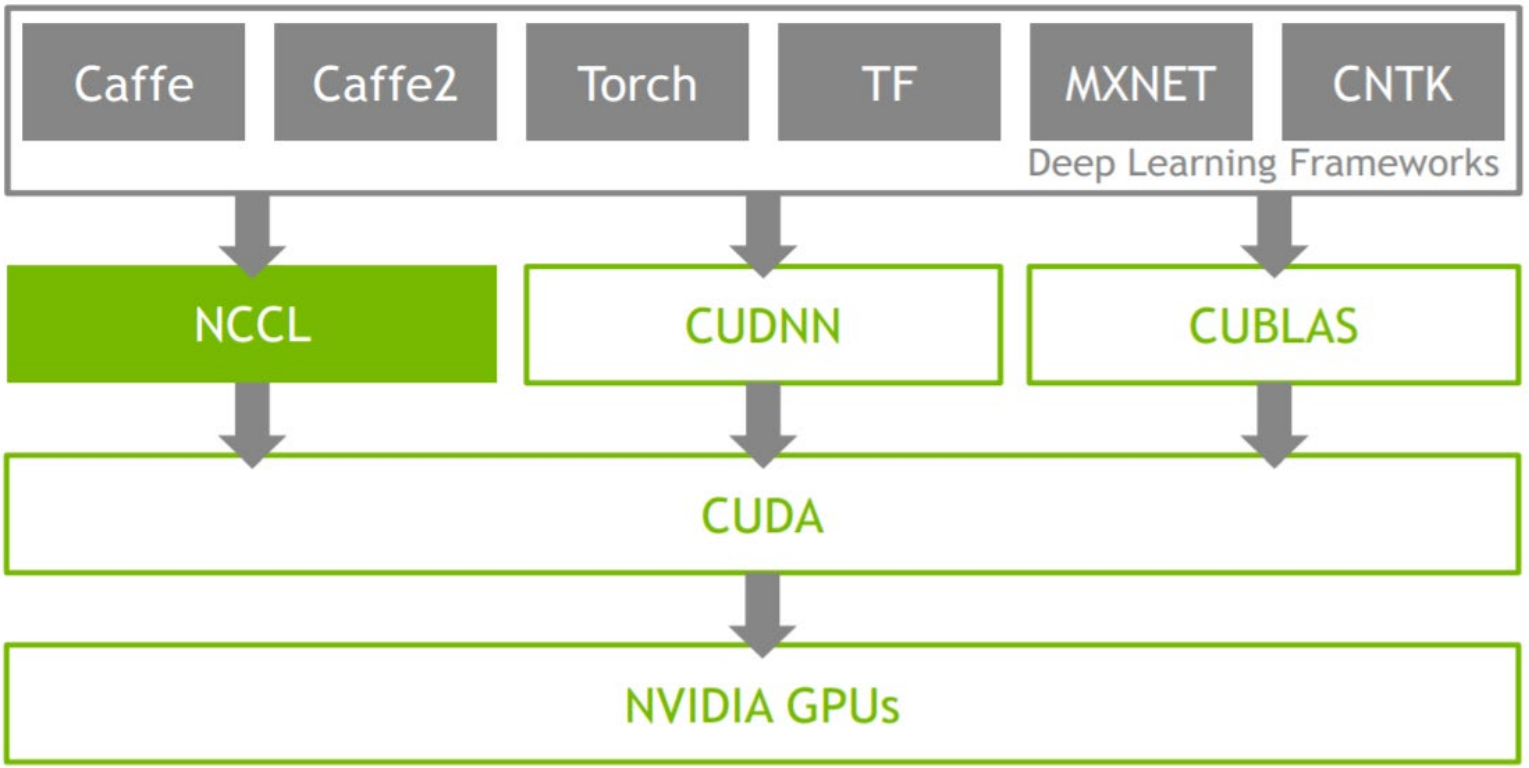
- Quantum-X800 InfiniBand
- Spectrum-X800
- BlueField-3 DPU

針對**AI Server**所推出的乙太網路，用於擴充數個機架，以及對外資料的傳輸

# Outline

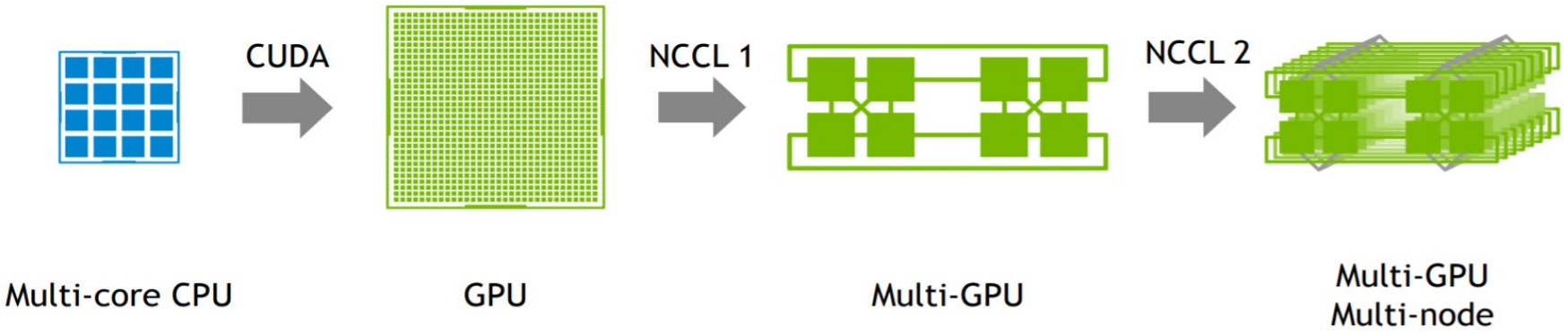
- Distributed Deep Learning
- Processing Acceleration for AI
- Communication and Memory Wall for AI
- Software Optimization for AI
- ML Systems

# CUDA Libraries for Deep Learning



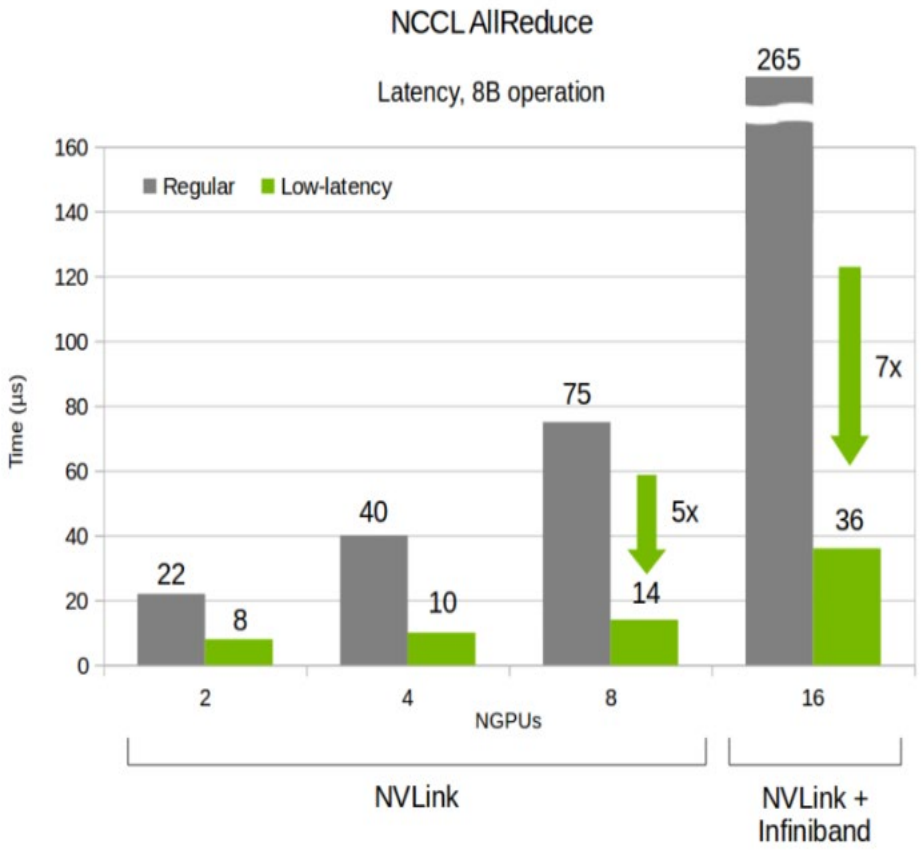
# NCCL

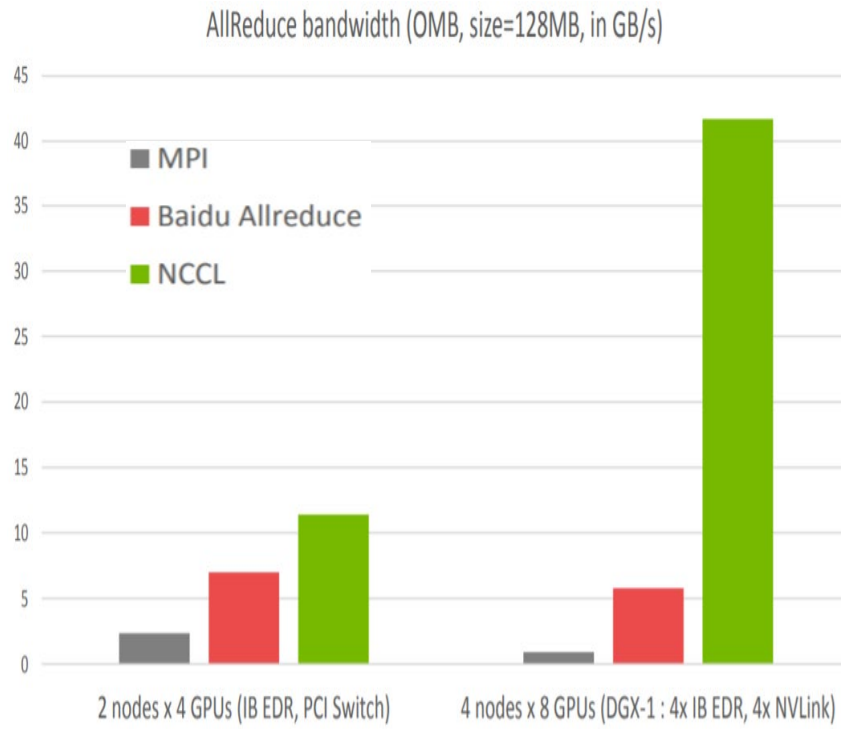
- NVIDIA Collective Communications Library
  - Optimized implementation of inter-GPU communication operations, such as **allreduce**
  - Deep learning frameworks can rely on NCCL's highly optimized, MPI compatible and topology aware routines, to take full advantage of all available GPUs within and across multiple nodes.
  - Optimized for **high bandwidth** and **low latency** over PCIe and NVLink high speed interconnect for intra-node communication and sockets and InfiniBand for inter-node communication



# NCCL

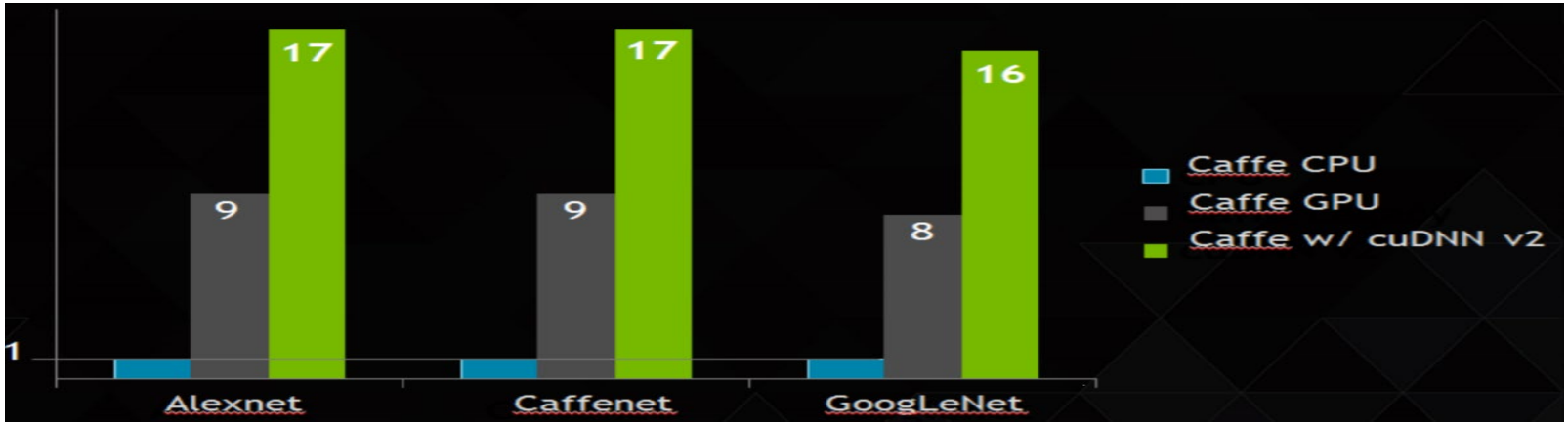
- Performance improvement on BW and Latency



# cuDNN

- Basic Deep Learning Subroutines:
  - E.g., convolutions, pooling, activation
  - Let user write a DNN application without custom CUDA code
- Flexible Layout
  - Handle any data layout
- Memory – Performance tradeoff
  - Good performance with minimal memory use, great performance with more memory use



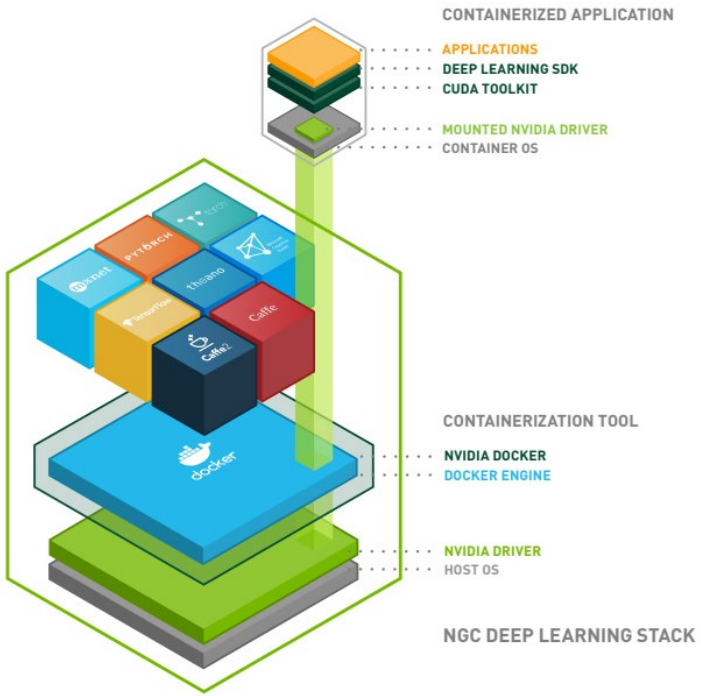
# cuBLAS

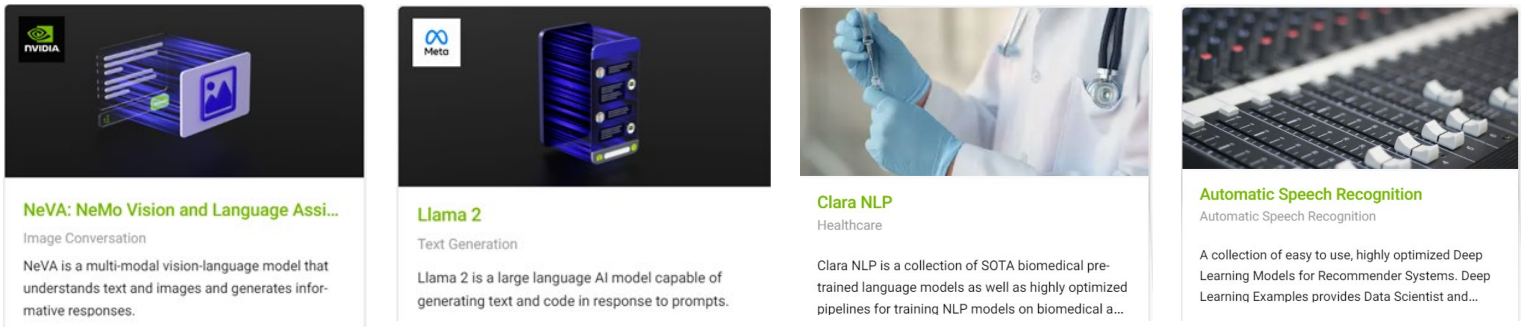
- BLAS: Basic Linear Algebra Subprograms
  - Defines a set of common functions for scalars, vectors, and matrices
    - E.g., cublaslasmax(): finds the smallest(first) index in a vector that is a maximum for that vector
  - Good for anything that uses heavy linear algebra computations
    - E.g., graphics, machine learning, computer vision, physical simulations

| numpy | math | cuBLAS (<T> is one of S, D, C, Z, H) |
|---|---|---|
| numpy.multiply(α, χ) | $(\lambda \mathbf{A})_{ij} = \lambda(\mathbf{A})_{ij}$ | cublas<T>gemm(α, χ) |
| numpy.multiply(χ, γ) | $(A \circ B)_{i,j} = (A)_{i,j}(B)_{i,j}.$ | cublas<T>gemm(χ, γ) |
| numpy.multiply(χ, A) | Aχ = C | cublas<T>gemm(χ, A) |
| numpy.multiply(A, B) | $C \leftarrow \alpha AB + \beta C$ | cublas<T>gemm(A, B) |
| | | |
| | | |

# Docker Container: NGC

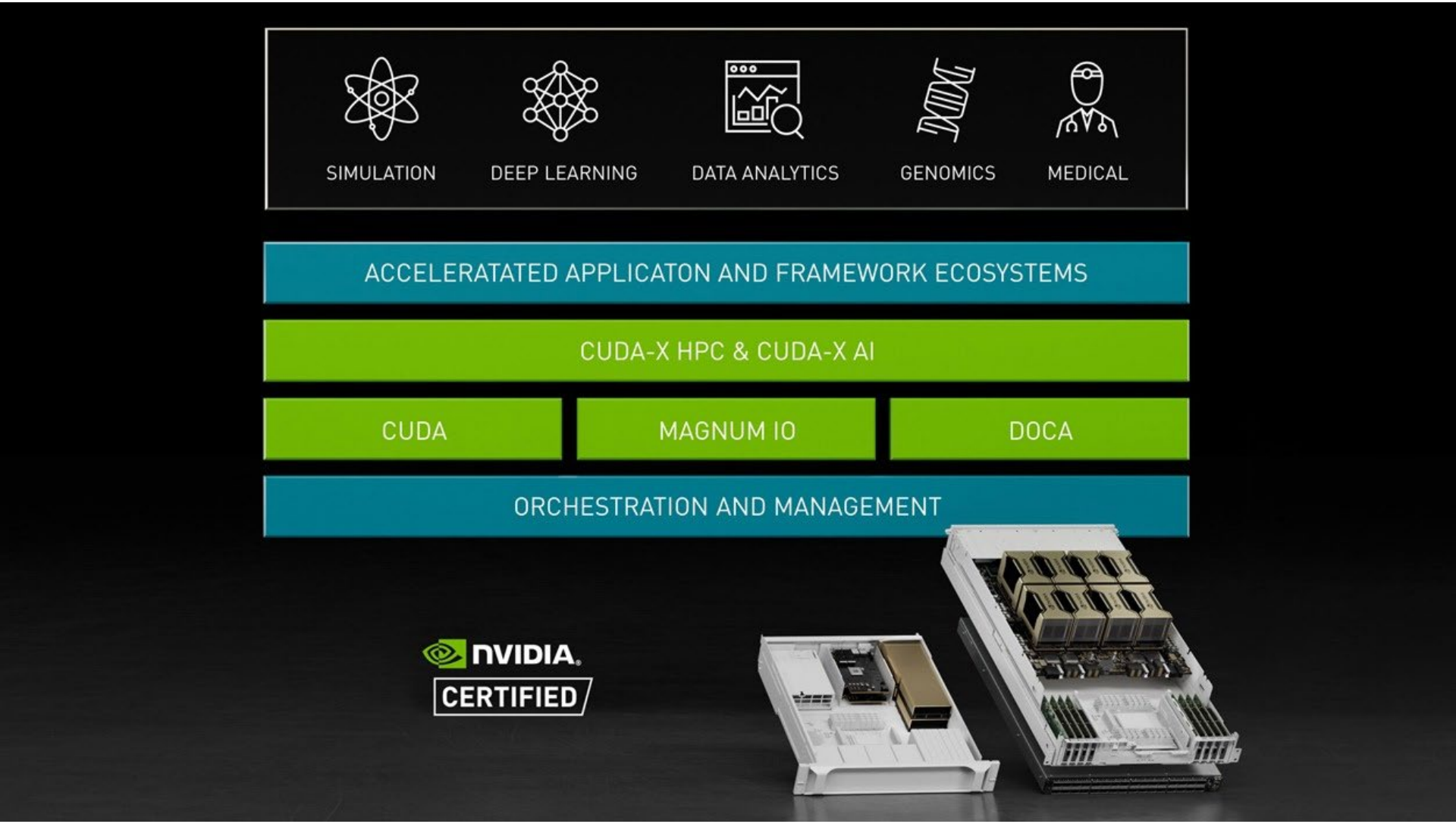
- The hub for GPU-optimized software for deep learning, machine learning, and HPC that provides containers, models, model scripts, and industry solutions
- Allow data scientists, developers and researchers can focus on building solutions and gathering insights faster.



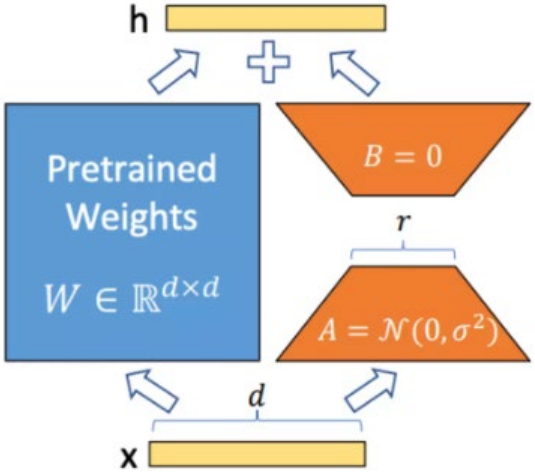
# Nvidia SDKs

Listen to what they said



# Memory Optimizations: LoRA

- LoRA[1]: A training method that accelerates the training of large models while consuming less memory
  - It adds pairs of rank-decomposition weight matrices (called update matrices) to existing weights, and only trains those newly added weights
  - Advantages:
    - Previous pretrained weights are kept frozen so the model is not as prone to catastrophic forgetting.
    - Rank-decomposition matrices have significantly fewer parameters than the original model

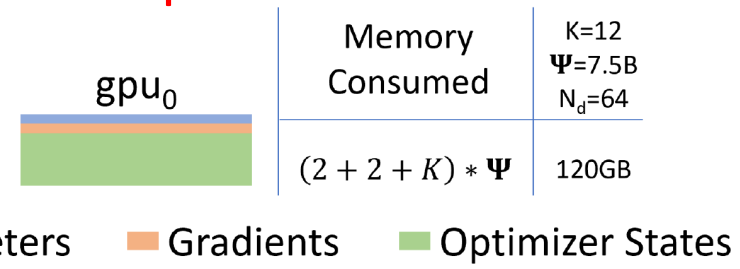



Train GPT-2 (original: 345M parameters) with <11M parameters.

[1]: LoRA: Low-Rank Adaptation of Large Language Models

# Memory Optimizations: Zero

- For large models, the majority of the memory is occupied by model states which include the optimizer states (such as momentum and variance in Adam), gradients, and model parameters
- ZeRO-DP aims to efficiently combine the idea of data parallel and model parallel, achieves the computation/communication efficiency of DP while achieving memory efficiency of MP
- ZeRO-DP optimize the memory consumption of model state by partition states on to devices




[1] ZeRO: Memory Optimizations Toward Training Trillion Parameter Models

# Memory Optimizations: Zero

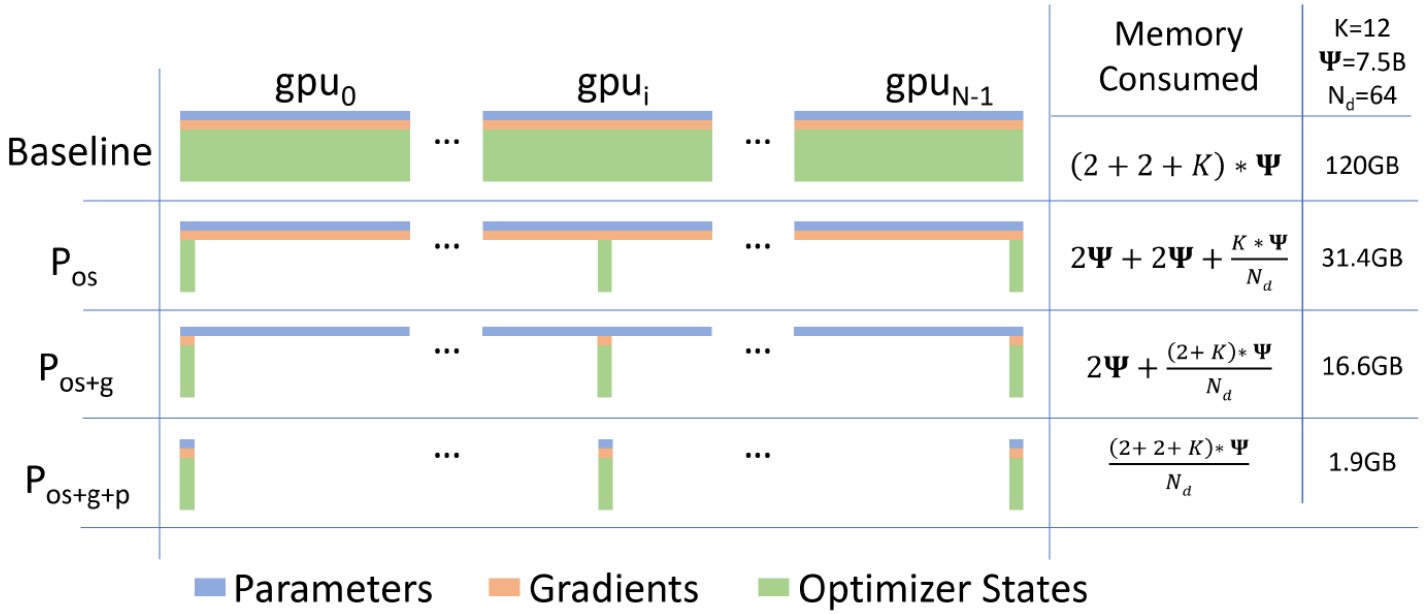
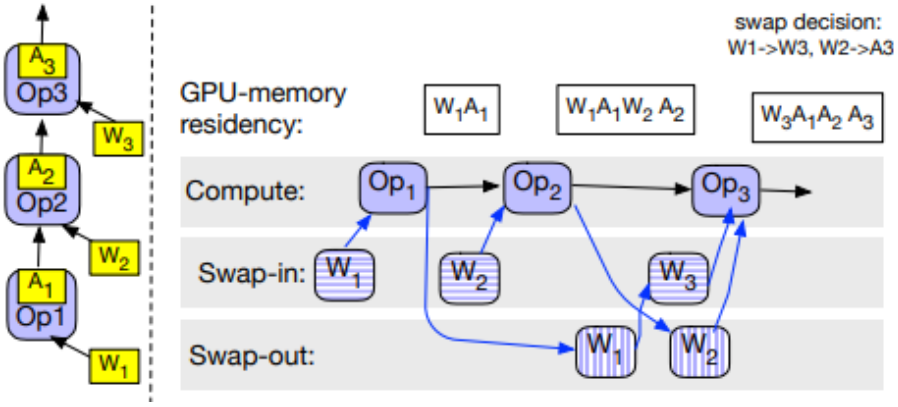



Figure 1: Comparing the per-device memory consumption of model states, with three stages of *ZeRO*-DP optimizations. $\Psi$ denotes model size (number of parameters), $K$ denotes the memory multiplier of optimizer states, and $N_d$ denotes DP degree. In the example, we assume a model size of $\Psi = 7.5B$ and DP of $N_d = 64$ with $K = 12$ based on mixed-precision training with Adam optimizer.

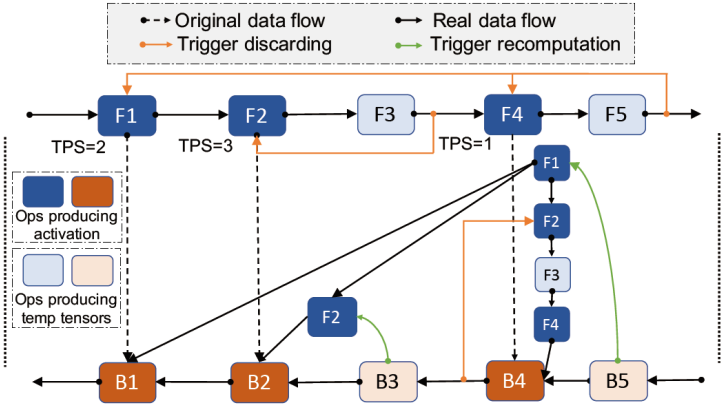
# Memory Optimizations

- Swapping
  - Use host memory as the swap space




SwapAdvisor: Pushing Deep Learning Beyond the GPU Memory Limit via Smart Swapping

- Re-computation
  - The values (e.g. activations) computed in forward propagation are used in backward




Melon: Breaking the Memory Wall for Resource-Efficient On-Device Machine Learning

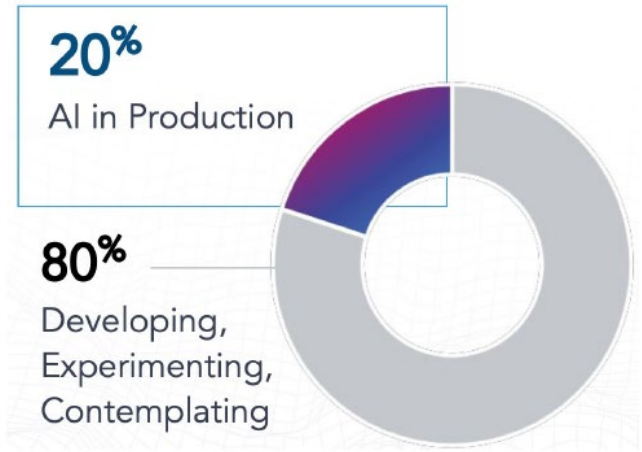
# Outline

- Distributed Deep Learning
- Processing Acceleration for AI
- Communication and Memory Wall for AI
- Software Optimization for AI
- ML Systems

# Gaps Between Education & Enterprise

| | Education | Enterprise |
|---|---|---|
| Objectives | Focus on Model Accuracy and use of Algorithm | Focus on Development/Integration with consideration of both Model Accuracy and Deployment Cost |
| Approach | Focus on Increasing Model Complexity for Better Accuracy | Keep Model Simple with Satisfied Accuracy |
| Data Source | Come from a Single or few Files | Come from multiple enterprise systems; need to be integrated, cross referenced, and summarized |
| Data Size | Typically Small to Medium | Range from Medium to Very Large |
| Data Characteristic | Typically 80% Clean | Start from raw data with 80% Noise |
| Tools & Dev. Env. | Limited tools, Standardized Env. | More tools + DevOps + Cloud + 3rd Party Solutions |
| Workflow | Fixed, Ad hoc | Agile, Dynamic |

# Growing AI Investments; Few Deployed at Scale



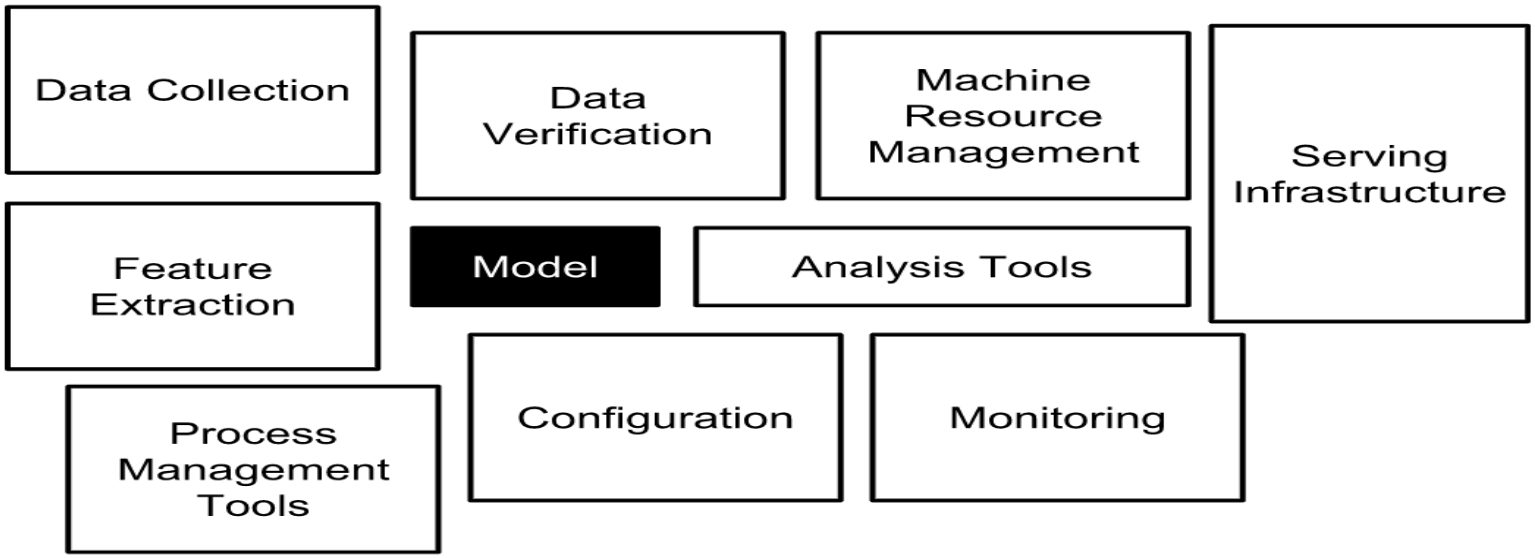

* Survey of 3073 AI-aware C-level Executives by Mckinsey


*Survey of 3073 AI-aware C-level Executives by Mckinsey

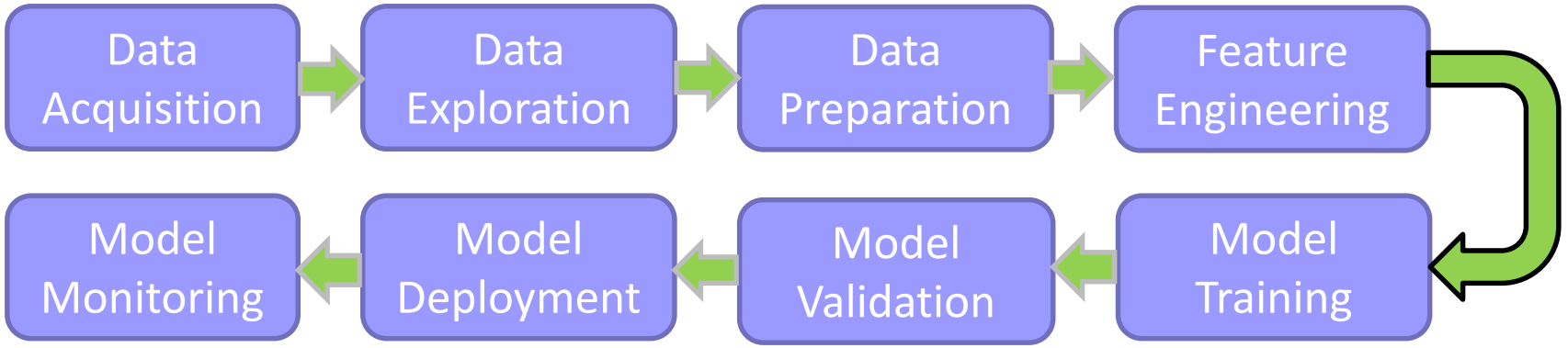
It is easy to get public model and dataset, but hard to deploy and operate the model in real world!!!

# ML System

- There is a lot more to AI/ML than just implementing an **algorithm** or a **technique**
  - Less than 5% of the production code is for ML model
- We need **a system** to support, optimize, and automate the whole process



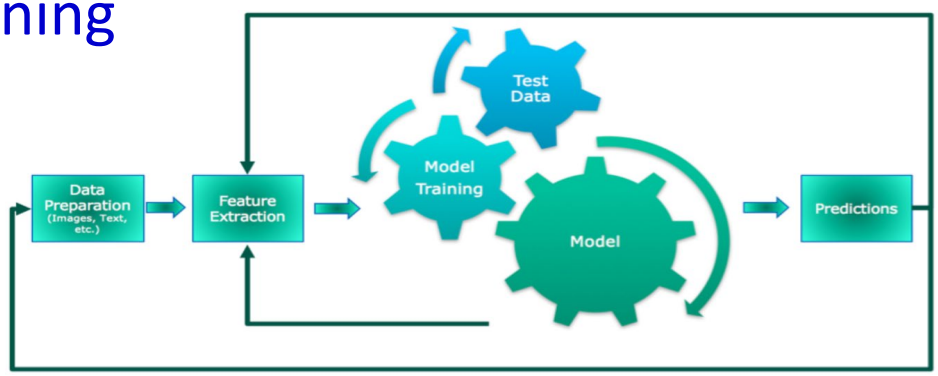
# ML Pipeline




- A machine learning pipeline is a way to **codify** and **automate** the **workflow** it takes to produce a machine learning mode
- Machine learning pipelines consist of **multiple sequential steps** that do everything from data extraction and preprocessing to model training and deployment

# ML Systems & MLOPs

- ML System: Designing and implementing the Systems that support ML models in real-world
  - Solve range of practical concerns that come with broader adoption
  - Optimize Hardware/Software systems for metrics beyond predictive accuracy
  - Foster a new systems machine learning research community at the intersection of the traditional systems and ML communities
- MLOPs: Continuous delivery and automation pipelines in machine learning



# New Community for ML Systems

- Designing and implementing the **Systems** that support ML models in real-world
  - Solve range of practical concerns that come with broader adoption
  - Optimize Hardware/Software systems for metrics beyond predictive accuracy
  - Foster a new systems machine learning research community at the intersection of the traditional systems and ML communities
- Research topics includes…
  - Efficient distributed training/inference
  - AI chip design & optimization
  - Systems to support the full machine learning lifecycle management
  - A lot more to be explored…